aws

Panduan Developerr

Amazon API Gateway

Amazon API Gateway: Panduan Developerr

Table of Contents

Apa itu Amazon API Gateway?

> **Note**
>
> Pengalaman konsol API Gateway yang didesain ulang sekarang tersedia. Untuk tutorial tentang cara menggunakan konsol untuk membuat REST API, lihatMemulai dengan konsol REST API.

Amazon API Gateway adalah AWS layanan untuk membuat, menerbitkan, memelihara, memantau, dan mengamankan REST, HTTP, dan WebSocket API dalam skala apa pun. Pengembang API dapat membuat API yang mengakses AWS atau layanan web lainnya, serta data yang disimpan di AWSCloud. Sebagai pengembang API Gateway API, Anda dapat membuat API untuk digunakan dalam aplikasi klien Anda sendiri. Atau Anda dapat membuat API Anda tersedia untuk pengembang aplikasi pihak ketiga. Untuk informasi selengkapnya, lihat the section called “Siapa yang menggunakan API Gateway?”.

API Gateway membuat API RESTful yang:

- Berbasis HTTP.
- Aktifkan komunikasi client-server stateless.
- Menerapkan metode HTTP standar seperti GET, POST, PUT, PATCH, dan DELETE.

Untuk informasi selengkapnya tentang API REST API Gateway API dan API HTTPthe section called “Memilih antara REST API dan HTTP API ”, lihatBekerja dengan HTTP API,the section called “Gunakan API Gateway untuk membuat REST API”,, danthe section called “Buat dan konfigurasikan”.

API Gateway membuat WebSocket API yang:

- Patuhi WebSocketprotokol, yang memungkinkan komunikasi full-duplex stateful antara klien dan server.
- Rutekan pesan masuk berdasarkan konten pesan.

Untuk informasi selengkapnya tentang API Gateway WebSocket API, lihat the section called “Gunakan API Gateway untuk membuat WebSocket API” danthe section called “Tentang WebSocket API”.

Topik

Arsitektur API Gateway

Diagram berikut menunjukkan arsitektur API Gateway.

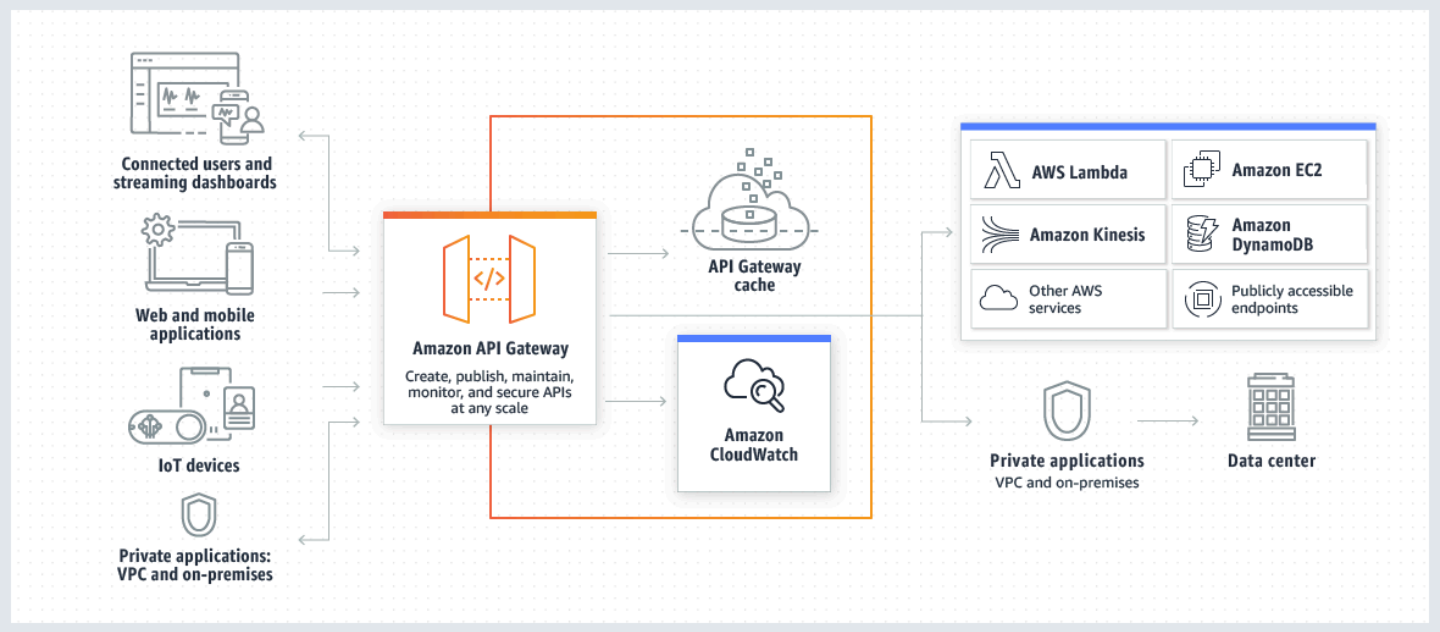


Diagram ini menggambarkan bagaimana API yang Anda buat di Amazon API Gateway memberi Anda atau pelanggan pengembang pengalaman developer yang terintegrasi dan konsisten untuk membuat aplikasi tanpa AWS server. API Gateway menangani semua tugas yang terlibat dalam menerima dan memproses hingga ratusan ribu panggilan API bersamaan. Tugas-tugas ini termasuk manajemen lalu lintas, otorisasi dan kontrol akses, pemantauan, dan manajemen versi API.

API Gateway bertindak sebagai “pintu depan” bagi aplikasi untuk mengakses data, logika bisnis, atau fungsionalitas dari layanan backend Anda, seperti beban kerja yang berjalan di Amazon Elastic Compute Cloud (Amazon EC2), kode yang berjalan, aplikasi web apa pun, atau aplikasi komunikasi AWS Lambda real-time.

Fitur API Gateway

Amazon API Gateway menawarkan fitur-fitur seperti berikut:

- Support untuk stateful (WebSocket) dan stateless (HTTP dan REST) API.
- Mekanisme otentikasi yang kuat dan fleksibel, seperti AWS Identity and Access Management kebijakan, fungsi otorisasi Lambda, dan kumpulan pengguna Amazon Cognito.
- Penerapan rilis Canary untuk meluncurkan perubahan dengan aman.
- CloudTraillogging dan pemantauan penggunaan API dan perubahan API.
- CloudWatch akses logging dan eksekusi logging, termasuk kemampuan untuk mengatur alarm. Untuk informasi selengkapnya, lihat the section called “Metrik CloudWatch” dan the section called “Metrik”.
- Kemampuan untuk menggunakan AWS CloudFormation template untuk mengaktifkan pembuatan API. Untuk informasi selengkapnya, lihat Referensi Jenis Sumber Daya Amazon API Gateway dan Referensi Jenis Sumber Daya Amazon API Gateway V2.
- Support untuk nama domain kustom.
- Integrasi dengan AWS WAFuntuk melindungi API Anda dari eksploitasi web umum.
- Integrasi dengan AWS X-Rayuntuk memahami dan memprioritaskan latensi kinerja.

Untuk daftar lengkap rilis fitur API Gateway, lihatRiwayat dokumen.

Kasus penggunaan API Gateway

Topik

- Gunakan API Gateway untuk membuat REST API
- Gunakan API Gateway untuk membuat API HTTP
- Gunakan API Gateway untuk membuat WebSocket API
- Siapa yang menggunakan API Gateway?

Gunakan API Gateway untuk membuat REST API

API API Gateway REST terdiri dari sumber daya dan metode. Resource adalah entitas logis yang dapat diakses aplikasi melalui jalur sumber daya. Metode sesuai dengan permintaan REST API yang dikirimkan oleh pengguna API Anda dan respons yang dikembalikan ke pengguna.

Misalnya, `/incomes` bisa menjadi jalur sumber daya yang mewakili pendapatan pengguna aplikasi. Sumber daya dapat memiliki satu atau lebih operasi yang ditentukan oleh kata kerja HTTP yang sesuai seperti GET, POST, PUT, PATCH, dan DELETE. Kombinasi jalur sumber daya dan operasi mengidentifikasi metode API. Misalnya, `POST /incomes` metode dapat menambahkan pendapatan yang diperoleh oleh penelepon, dan `GET /expenses` metode dapat menanyakan biaya yang dilaporkan yang dikeluarkan oleh penelepon.

Aplikasi tidak perlu tahu di mana data yang diminta disimpan dan diambil dari backend. Di API REST API Gateway, frontend dienkapsulasi oleh permintaan metode dan respons metode. API berinteraksi dengan backend melalui permintaan integrasi dan respons integrasi.

Misalnya, dengan DynamoDB sebagai backend, pengembang API menyiapkan permintaan integrasi untuk meneruskan permintaan metode masuk ke backend yang dipilih. Penyiapan mencakup spesifikasi tindakan DynamoDB yang sesuai, peran dan kebijakan IAM yang diperlukan, dan transformasi data input yang diperlukan. Backend mengembalikan hasilnya ke API Gateway sebagai respons integrasi.

Untuk merutekan respons integrasi ke respons metode yang sesuai (dari kode status HTTP yang diberikan) ke klien, Anda dapat mengonfigurasi respons integrasi untuk memetakan parameter respons yang diperlukan dari integrasi ke metode. Anda kemudian menerjemahkan format data keluaran backend ke format frontend, jika perlu. API Gateway memungkinkan Anda untuk menentukan skema atau model untuk payload untuk memfasilitasi pengaturan template pemetaan tubuh.

API Gateway menyediakan fungsionalitas manajemen REST API seperti berikut ini:

- Support untuk membuat SDK dan membuat dokumentasi API menggunakan ekstensi API Gateway ke OpenAPI
- Pelambatan permintaan HTTP

Gunakan API Gateway untuk membuat API HTTP

HTTP API memungkinkan Anda membuat RESTful API dengan latensi lebih rendah dan biaya lebih rendah daripada REST API.

Anda dapat menggunakan API HTTP untuk mengirim permintaan ke AWS Lambda fungsi atau ke titik akhir HTTP yang dapat dirutekan secara publik.

Misalnya, Anda dapat membuat API HTTP yang terintegrasi dengan fungsi Lambda di backend. Saat klien memanggil API Anda, API Gateway mengirimkan permintaan ke fungsi Lambda dan mengembalikan respons fungsi ke klien.

HTTP API mendukung otorisasi OpenID Connect dan OAuth 2.0. Mereka datang dengan dukungan bawaan untuk berbagi sumber daya lintas asal (CORS) dan penerapan otomatis.

Untuk mempelajari selengkapnya, lihat the section called “Memilih antara REST API dan HTTP API ”.

Gunakan API Gateway untuk membuat WebSocket API

Dalam WebSocket API, klien dan server dapat saling mengirim pesan kapan saja. Server backend dapat dengan mudah mendorong data ke pengguna dan perangkat yang terhubung, menghindari kebutuhan untuk menerapkan mekanisme pemungutan suara yang kompleks.

Misalnya, Anda dapat membuat aplikasi tanpa server menggunakan API Gateway WebSocket API dan mengirim serta menerima pesan AWS Lambda ke dan dari pengguna individu atau grup pengguna di ruang obrolan. Atau Anda dapat memanggil layanan backend seperti, Amazon AWS Lambda Kinesis, atau titik akhir HTTP berdasarkan konten pesan.

Anda dapat menggunakan API Gateway WebSocket API untuk membangun aplikasi komunikasi real-time yang aman tanpa harus menyediakan atau mengelola server apa pun untuk mengelola koneksi atau pertukaran data skala besar. Kasus penggunaan yang ditargetkan mencakup aplikasi waktu nyata seperti berikut ini:

- Aplikasi obrolan
- Dasbor real-time seperti ticker saham
- Peringatan dan notifikasi waktu nyata

API Gateway menyediakan fungsionalitas manajemen WebSocket API seperti berikut ini:

- Pemantauan dan pembatasan koneksi dan pesan

- Menggunakan AWS X-Ray untuk melacak pesan saat mereka melakukan perjalanan melalui API ke layanan backend
- Integrasi mudah dengan titik akhir HTTP/HTTPS

Siapa yang menggunakan API Gateway?

Ada dua jenis pengembang yang menggunakan API Gateway: pengembang API dan pengembang aplikasi.

Pengembang API membuat dan menerapkan API untuk mengaktifkan fungsionalitas yang diperlukan di API Gateway. Pengembang API harus menjadi pengguna di AWS akun yang memiliki API.

Pengembang aplikasi membangun aplikasi yang berfungsi untuk memanggil AWS layanan dengan menjalankan API REST WebSocket atau yang dibuat oleh pengembang API di API Gateway.

Pengembang aplikasi adalah pelanggan pengembang API. Pengembang aplikasi tidak perlu memiliki AWS akun, asalkan API tidak memerlukan izin IAM atau mendukung otorisasi pengguna melalui penyedia identitas federasi pihak ketiga yang didukung oleh federasi identitas kumpulan pengguna Amazon Cognito. Penyedia identitas tersebut termasuk Amazon, kumpulan pengguna Amazon Cognito, Facebook, dan Google.

Membuat dan mengelola API Gateway API

Pengembang API bekerja dengan komponen layanan API Gateway untuk manajemen API, bernama`apigateway`, untuk membuat, mengonfigurasi, dan menerapkan API.

Sebagai pengembang API, Anda dapat membuat dan mengelola API dengan menggunakan konsol API Gateway, yang dijelaskan dalamMemulai dengan API Gateway, atau dengan memanggilReferensi API. Ada beberapa cara untuk memanggil API ini. Mereka termasuk menggunakan AWS Command Line Interface (AWS CLI), atau dengan menggunakan AWS SDK. Selain itu, Anda dapat mengaktifkan pembuatan API dengan AWS CloudFormationtemplate atau (dalam kasus REST API dan HTTP API)Bekerja dengan ekstensi API Gateway ke OpenAPI.

Untuk daftar Wilayah tempat API Gateway tersedia, serta titik akhir layanan kontrol terkait, lihat Titik Akhir dan Kuota Amazon API Gateway.

Memanggil API Gateway API

Pengembang aplikasi bekerja dengan komponen layanan API Gateway untuk eksekusi API, bernama`execute-api`, untuk memanggil API yang dibuat atau diterapkan di API Gateway. Entitas

pemrograman yang mendasarinya diekspos oleh API yang dibuat. Ada beberapa cara untuk memanggil API semacam itu. Untuk mempelajari lebih lanjut, lihat Memanggil REST API di Amazon API Gateway dan Memanggil API WebSocket .

Mengakses API Gateway

Anda dapat mengakses Amazon API Gateway dengan cara berikut:

- AWS Management Console— AWS Management Console Menyediakan antarmuka web untuk membuat dan mengelola API. Setelah menyelesaikan langkah-langkahnyaPrasyarat, Anda dapat mengakses konsol API Gateway dihttps://console.aws.amazon.com/apigateway.
- AWSSDK — Jika Anda menggunakan bahasa pemrograman yang AWS menyediakan SDK, Anda dapat menggunakan SDK untuk mengakses API Gateway. SDK menyederhanakan otentikasi, terintegrasi dengan mudah dengan lingkungan pengembangan Anda, dan menyediakan akses ke perintah API Gateway. Untuk informasi lebih lanjut, lihat Alat untuk Amazon Web Services.
- API Gateway API V1 dan V2 API — Jika Anda menggunakan bahasa pemrograman yang tidak tersedia untuk SDK, lihat Referensi API Amazon API Gateway Versi 1 dan Referensi API Amazon API Gateway Versi 2.
- AWS Command Line Interface – Untuk informasi selengkapnya, lihat Menyiapkan dengan AWS Command Line Interface di AWS Command Line Interface Panduan Pengguna.
- AWS Tools for Windows PowerShell— Untuk informasi selengkapnya, lihat Menyiapkan AWS Tools for Windows PowerShell di Panduan AWS Tools for Windows PowerShell Pengguna.

Bagian dari infrastruktur AWS tanpa server

Bersama dengan AWS Lambda, API Gateway membentuk bagian yang menghadap aplikasi dari infrastruktur tanpa AWS server. Untuk mempelajari lebih lanjut tentang memulai dengan tanpa server, lihat Panduan Pengembang Tanpa Server.

Agar aplikasi dapat memanggil AWS layanan yang tersedia untuk umum, Anda dapat menggunakan Lambda untuk berinteraksi dengan layanan yang diperlukan dan mengekspos fungsi Lambda melalui metode API di API Gateway. AWS Lambdamenjalankan kode Anda pada infrastruktur komputasi yang sangat tersedia. Ini melakukan eksekusi dan administrasi sumber daya komputasi yang diperlukan. Untuk mengaktifkan aplikasi tanpa server, API Gateway mendukung integrasi proxy yang efisien dengan AWS Lambda dan titik akhir HTTP.

Cara memulai dengan Amazon API Gateway

Untuk pengenalan Amazon API Gateway, lihat berikut ini:

- Mulai, yang menyediakan panduan untuk membuat API HTTP.
- Tanah tanpa server, yang menyediakan video instruksional.
- Happy Little API Shorts, yang merupakan serangkaian video instruksional singkat.

Konsep Amazon API Gateway

API Gateway

API Gateway adalahAWSlayanan yang mendukung hal berikut:

- Membuat, menyebarkan, dan mengelolaRESTantarmuka pemrograman aplikasi (API) untuk mengekspos titik akhir HTTP backend,AWS Lambdafungsi, atau lainnyaAWSlayanan.
- Membuat, menyebarkan, dan mengelolaWebSocketAPI untuk mengeksposAWS Lambdafungsi atau lainnyaAWSlayanan.
- Memanggil metode API terbuka melalui frontend HTTP dan WebSocket endpoint.

API REST API Gateway

Kumpulan sumber daya HTTP dan metode yang terintegrasi dengan backend HTTP endpoint, fungsi Lambda, atau lainnyaAWSlayanan. Anda dapat menyebarkan koleksi ini dalam satu atau lebih tahap. Biasanya, sumber daya API diatur dalam pohon sumber daya sesuai dengan logika aplikasi. Setiap sumber daya API dapat mengekspos satu atau beberapa metode API yang memiliki kata kerja HTTP unik yang didukung oleh API Gateway. Untuk informasi selengkapnya, lihat the section called “Memilih antara REST API dan HTTP API ”.

API HTTP API Gateway

Kumpulan rute dan metode yang terintegrasi dengan backend HTTP endpoint atau fungsi Lambda. Anda dapat menyebarkan koleksi ini dalam satu atau lebih tahap. Setiap rute dapat mengekspos satu atau beberapa metode API yang memiliki kata kerja HTTP unik yang didukung oleh API Gateway. Untuk informasi selengkapnya, lihat the section called “Memilih antara REST API dan HTTP API ”.

API WebSocket API Gateway

Kumpulan rute WebSocket dan tombol rute yang terintegrasi dengan endpoint HTTP backend, fungsi Lambda, atau lainnyaAWSlayanan. Anda dapat menyebarkan koleksi ini dalam satu atau

lebih tahap. Metode API dipanggil melalui koneksi WebSocket frontend yang dapat Anda kaitkan dengan nama domain kustom terdaftar.

Penyebaran API

Snapshot pada suatu titik waktu API Gateway Anda. Agar tersedia untuk digunakan klien, penyebaran harus dikaitkan dengan satu atau lebih tahapan API.

Pengembang API

KlasterAWSakun yang memiliki penerapan API Gateway (misalnya, penyedia layanan yang juga mendukung akses terprogram).

Titik akhir API

Nama host untuk API di API Gateway yang digunakan ke Wilayah tertentu. Nama host adalah bentuk`{api-id}.execute-api.{region}.amazonaws.com`. Jenis endpoint API berikut ini didukung:

- Titik akhir API yang dioptimalkan
- Titik akhir API privat
- Titik akhir API Regional

Kunci API

String alfanumerik yang digunakan API Gateway untuk mengidentifikasiaplikasipengembang yang menggunakan REST atau WebSocket API Anda. API Gateway dapat menghasilkan kunci API atas nama Anda, atau Anda dapat mengimpornya dari file CSV. Anda dapat menggunakan kunci API bersama-sama denganOtorisasi Lambdaataupaket penggunaanuntuk mengontrol akses ke API Anda.

LihatTitik akhir API.

Pemilik API

LihatPengembang API.

Tahap API

Referensi logis ke status siklus hidup API Anda (misalnya, 'dev', 'prod', 'beta', 'v2'). Tahapan API diidentifikasi oleh API ID dan nama stage.

Pengembang aplikasi

Pembuat aplikasi yang mungkin atau mungkin tidak memilikiAWSakun dan berinteraksi dengan API yang Anda, pengembang API, telah diterapkan. Pengembang aplikasi adalah pelanggan Anda. Pengembang aplikasi biasanya diidentifikasi olehKunci API.

URL Callback

Ketika klien baru terhubung ke melalui koneksi WebSocket, Anda dapat memanggil integrasi di API Gateway untuk menyimpan URL callback klien. Anda kemudian dapat menggunakan URL callback untuk mengirim pesan ke klien dari sistem backend.

Portal Pengembang

Aplikasi yang memungkinkan pelanggan Anda mendaftar, menemukan, dan berlangganan produk API Anda (paket penggunaan API Gateway), mengelola kunci API mereka, dan melihat metrik penggunaannya untuk API Anda.

Titik akhir API yang dioptimalkan

Nama host default API Gateway API yang disebarkan ke Wilayah yang ditentukan saat menggunakan distribusi CloudFront untuk memfasilitasi akses klien biasanya dari seluruhAWSWilayah. Permintaan API dialihkan ke CloudFront Point of Presence (POP) terdekat, yang biasanya meningkatkan waktu koneksi untuk klien yang beragam secara geografis.

LihatTitik akhir API.

Permintaan integrasi

Antarmuka internal rute API WebSocket atau metode REST API di API Gateway, di mana Anda memetakan badan permintaan rute atau parameter dan badan permintaan metode ke format yang diperlukan oleh backend.

Tanggapan integrasi

Antarmuka internal rute API WebSocket atau metode REST API di API Gateway, di mana Anda memetakan kode status, header, dan muatan yang diterima dari backend ke format respons yang dikembalikan ke aplikasi klien.

Template pemetaan

Skrip diKecepatan Template Bahasa (VTL)yang mengubah tubuh permintaan dari format data frontend ke format data backend, atau yang mengubah tubuh respon dari format data backend ke

format data frontend. Pemetaan template dapat ditentukan dalam permintaan integrasi atau dalam respon integrasi. Mereka dapat referensi data dibuat tersedia pada runtime sebagai konteks dan tahap variabel.

Pemetaan bisa sesederhanatransformasi identitasyang melewati header atau badan melalui integrasi apa adanya dari klien ke backend untuk permintaan. Hal yang sama berlaku untuk respon, di mana muatan diteruskan dari backend ke klien.

permintaan metode

Antarmuka publik metode API di API Gateway yang mendefinisikan parameter dan badan yang harus dikirim oleh pengembang aplikasi untuk mengakses backend melalui API.

Respon metode

Antarmuka publik REST API yang mendefinisikan kode status, header, dan model tubuh yang diharapkan pengembang aplikasi dalam respons dari API.

Integrasi Mock

Dalam integrasi tiruan, respons API dihasilkan dari API Gateway secara langsung, tanpa perlu backend integrasi. Sebagai pengembang API, Anda memutuskan bagaimana API Gateway merespons permintaan integrasi tiruan. Untuk ini, Anda mengonfigurasi permintaan integrasi metode dan respons integrasi untuk mengaitkan respons dengan kode status tertentu.

Model

Sebuah skema data yang menentukan struktur data permintaan atau respon payload. Sebuah model diperlukan untuk menghasilkan SDK API yang diketik dengan kuat. Hal ini juga digunakan untuk memvalidasi muatan. Sebuah model nyaman untuk menghasilkan template pemetaan sampel untuk memulai pembuatan template pemetaan produksi. Meskipun berguna, model tidak diperlukan untuk membuat template pemetaan.

API privat

LihatTitik akhir API privat.

Titik akhir API privat

Endpoint API yang terpapar melalui endpoint VPC antarmuka dan memungkinkan klien mengakses sumber daya API pribadi dengan aman di dalam VPC. API pribadi diisolasi dari internet publik, dan hanya dapat diakses menggunakan endpoint VPC untuk API Gateway yang telah diberikan akses.

Integrasi privat

Jenis integrasi API Gateway untuk klien untuk mengakses sumber daya di dalam VPC pelanggan melalui endpoint REST API pribadi tanpa mengekspos sumber daya ke internet publik.

Integrasi proksi

Konfigurasi integrasi API Gateway yang disederhanakan. Anda dapat mengatur integrasi proxy sebagai integrasi proxy HTTP atau integrasi proxy Lambda.

Untuk integrasi proxy HTTP, API Gateway meneruskan seluruh permintaan dan respons antara frontend dan backend HTTP. Untuk integrasi proxy Lambda, API Gateway mengirimkan seluruh permintaan sebagai masukan ke fungsi Lambda backend. API Gateway kemudian mengubah output fungsi Lambda menjadi respons HTTP frontend.

Dalam REST API, integrasi proxy paling sering digunakan dengan sumber daya proxy, yang diwakili oleh variabel jalur serakah (misalnya,`{proxy+}`) dikombinasikan dengan catch-allANYmetode.

Membuat cepat

Anda dapat menggunakan quick create untuk menyederhanakan pembuatan HTTP API. Pembuatan cepat membuat API dengan integrasi Lambda atau HTTP, rute default catch-all, dan tahap default yang dikonfigurasi untuk men-deploy perubahan secara otomatis. Untuk informasi selengkapnya, lihat the section called “Buat HTTP API dengan menggunakanAWSCLI”.

Titik akhir API Regional

Nama host API yang disebarkan ke Wilayah yang ditentukan dan dimaksudkan untuk melayani klien, seperti instans EC2, dalam hal yang samaAWSWilayah. Permintaan API ditargetkan langsung ke API Gateway API khusus Wilayah tanpa melalui distribusi CloudFront apa pun. Untuk permintaan di wilayah, endpoint Regional melewati perjalanan pulang pergi yang tidak perlu ke distribusi CloudFront.

Selain itu, Anda dapat menerapkanPerutean berbasis latensipada endpoint Regional untuk menyebarkan API ke beberapa Wilayah menggunakan konfigurasi titik akhir API Regional yang sama, atur nama domain khusus yang sama untuk setiap API yang digunakan, dan konfigurasikan data DNS berbasis latensi di Route 53 untuk merutekan permintaan klien ke Wilayah yang memiliki latensi terendah.

LihatTitik akhir API.

Rute

Rute WebSocket di API Gateway digunakan untuk mengarahkan pesan masuk ke integrasi tertentu, sepertiAWS Lambdafungsi, berdasarkan isi pesan. Ketika Anda menentukan WebSocket API Anda, Anda menentukan kunci rute dan backend integrasi. Kunci rute adalah atribut di badan pesan. Ketika tombol rute dicocokkan dalam pesan masuk, backend integrasi dipanggil.

Rute default juga dapat diatur untuk kunci rute yang tidak cocok atau untuk menentukan model proxy yang melewati pesan melalui apa adanya komponen backend yang melakukan routing dan memproses permintaan.

Permintaan rute

Antarmuka publik metode API WebSocket di API Gateway yang mendefinisikan badan yang harus dikirim oleh pengembang aplikasi dalam permintaan untuk mengakses backend melalui API.

Jawaban rute

Antarmuka publik API WebSocket yang mendefinisikan kode status, header, dan model tubuh yang diharapkan pengembang aplikasi dari API Gateway.

Paket penggunaan

SEBUAHpaket penggunaanmenyediakan klien API yang dipilih dengan akses ke satu atau lebih digunakan REST atau WebSocket API. Anda dapat menggunakan paket penggunaan untuk mengkonfigurasi throttling dan batas kuota, yang diberlakukan pada kunci API klien individu.

Koneksi WebSocket

API Gateway mempertahankan koneksi persisten antara klien dan API Gateway itu sendiri. Tidak ada koneksi persisten antara API Gateway dan integrasi backend seperti fungsi Lambda. Layanan backend dipanggil sesuai kebutuhan, berdasarkan isi pesan yang diterima dari klien.

Memilih antara REST API dan HTTP API

REST API dan HTTP API keduanya merupakan produk RESTful API. REST API mendukung lebih banyak fitur daripada API HTTP, sedangkan API HTTP dirancang dengan fitur minimal sehingga dapat ditawarkan dengan harga lebih murah. Pilih REST API jika Anda memerlukan fitur seperti kunci API, pembatasan per klien, validasi permintaan, AWS WAF integrasi, atau titik akhir API pribadi. Pilih HTTP API jika Anda tidak memerlukan fitur yang disertakan dengan REST API.

Bagian berikut merangkum fitur inti yang tersedia di REST API dan HTTP API.

Jenis titik akhir

Jenis titik akhir mengacu pada titik akhir yang dibuat API Gateway untuk API Anda. Untuk informasi selengkapnya, lihat the section called “Pilih jenis titik akhir API”.

Jenis titik akhir	API REST	API HTTPS
Dioptimalkan tepi	✓	
Regional	✓	✓
Pribadi	✓	

Keamanan

API Gateway menyediakan sejumlah cara untuk melindungi API Anda dari ancaman tertentu, seperti aktor jahat atau lonjakan lalu lintas. Untuk mempelajari selengkapnya, lihat the section called “Melindungi” dan the section called “Melindungi”.

Fitur keamanan	API REST	API HTTPS
Otentikasi TLS timbal balik	✓	✓
Sertifikat untuk otentikasi backend	✓	
AWS WAF	✓	

Otorisasi

API Gateway mendukung beberapa mekanisme untuk mengontrol dan mengelola akses ke API Anda. Lihat informasi yang lebih lengkap di the section called “Kontrol akses” dan the section called “Kontrol akses”.

Opsi otorisasi	API REST	API HTTPS
IAM	✓	✓

Opsi otorisasi	API REST	API HTTPS
Kebijakan sumber daya	✓	
Amazon Cognito	✓	✓ [1]
Otorisasi khusus dengan suatu fungsi AWS Lambda	✓	✓
Token Web JSON (JWT) [2]		✓

[1] Anda dapat menggunakan Amazon Cognito dengan otorisasi JWT.

[2] Anda dapat menggunakan otorisasi Lambda untuk memvalidasi JWT untuk REST API.

Manajemen API

Pilih REST API jika Anda memerlukan kemampuan manajemen API seperti kunci API dan pembatasan tarif per klien. Lihat informasi selengkapnya di the section called “Mendistribusikan”, the section called “Nama domain kustom”, dan the section called “Nama domain kustom”.

Fitur	API REST	API HTTPS
Domain kustom	✓	✓
Kunci API	✓	
Pembatasan tarif per klien	✓	
Pelambatan penggunaan per klien	✓	

Pengembangan

Saat Anda mengembangkan API Gateway API, Anda memutuskan sejumlah karakteristik API Anda. Karakteristik ini bergantung pada kasus penggunaan API Anda. Untuk informasi lebih lanjut, lihat the section called “Kembangkan” dan the section called “Mengembangkan”.

Fitur	API REST	API HTTPS
Konfigurasi CORS	✓	✓
Doa uji	✓	
Caching	✓	
Penerapan yang dikendalikan pengguna	✓	✓
Penerapan otomatis		✓
Tanggapan gateway khusus	✓	
Penerapan rilis kenari	✓	
Minta validasi	✓	
Minta transformasi parameter	✓	✓
Minta transformasi tubuh	✓	

Pemantauan

API Gateway mendukung beberapa opsi untuk mencatat permintaan API dan memantau API Anda. Lihat informasi yang lebih lengkap di the section called “Memantau” dan the section called “Memantau”.

Fitur	API REST	API HTTPS
CloudWatch Metrik Amazon	✓	✓
Akses log ke CloudWatch Log	✓	✓
Akses log ke Amazon Data Firehose	✓	
Log eksekusi	✓	

Fitur	API REST	API HTTPS
AWS X-Ray menelusuri	✓	

Integrasi

Integrasi menghubungkan API Gateway API Anda ke sumber daya backend. Lihat informasi yang lebih lengkap di the section called “Integrasi” dan the section called “Integrasi”.

Fitur	API REST	API HTTPS
Titik akhir HTTP publik	✓	✓
AWS layanan	✓	✓
AWS Lambda fungsi	✓	✓
Integrasi pribadi dengan Network Load Balancers	✓	✓
Integrasi pribadi dengan Application Load Balancers		✓
Integrasi pribadi dengan AWS Cloud Map		✓
Integrasi tiruan	✓	

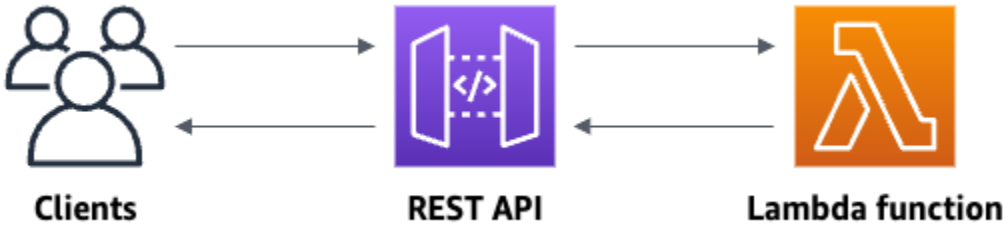
Memulai dengan konsol REST API

Dalam latihan memulai ini, Anda membuat REST API tanpa server menggunakan konsol API Gateway REST API. API tanpa server memungkinkan Anda fokus pada aplikasi Anda alih-alih menghabiskan waktu Anda menyediakan dan mengelola server. Latihan ini harus memakan waktu kurang dari 20 menit untuk diselesaikan, dan dimungkinkan dalam Tingkat AWS Gratis.

Pertama, Anda membuat fungsi Lambda menggunakan konsol Lambda. Selanjutnya, Anda membuat REST API menggunakan konsol API Gateway REST API. Kemudian, Anda membuat metode API

dan mengintegrasikannya dengan fungsi Lambda menggunakan integrasi proxy Lambda. Terakhir, Anda menerapkan dan menjalankan API Anda.

Saat Anda menjalankan REST API, API Gateway merutekan permintaan ke fungsi Lambda Anda. Lambda menjalankan fungsi dan mengembalikan respons ke API Gateway. API Gateway kemudian mengembalikan respons itu kepada Anda.



Untuk menyelesaikan latihan ini, Anda memerlukan pengguna Akun AWS dan AWS Identity and Access Management (IAM) dengan akses konsol. Untuk informasi selengkapnya, lihat Prasyarat untuk memulai dengan API Gateway.

Topik

- Langkah 1: Membuat fungsi Lambda
- Langkah 2: Buat REST API
- Langkah 3: Buat integrasi proxy Lambda
- Langkah 4: Menerapkan API Anda
- Langkah 5: Panggil API Anda
- (Opsional) Langkah 6: Bersihkan

Langkah 1: Membuat fungsi Lambda

Anda menggunakan fungsi Lambda untuk backend API Anda. Lambda menjalankan kode Anda hanya saat diperlukan dan menskalakan secara otomatis, dari beberapa permintaan per hari hingga ribuan per detik.

Untuk latihan ini, Anda menggunakan fungsi Node.js default di konsol Lambda.

Untuk membuat fungsi Lambda

1. Masuk ke konsol Lambda di https://console.aws.amazon.com/lambda.
2. Pilih Buat fungsi.

3. Di bagian Informasi dasar, untuk Nama fungsi, masukkan **`my-function`**.
4. Pilih Buat fungsi.

Kode fungsi Lambda default akan terlihat mirip dengan berikut ini:

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

Anda dapat memodifikasi fungsi Lambda untuk latihan ini, selama respons fungsi selaras dengan format yang dibutuhkan API Gateway.

Ganti body respons default (`Hello from Lambda!`) dengan`The API Gateway REST API console is great!`. Ketika Anda memanggil fungsi contoh, ia mengembalikan `200` respons ke klien, bersama dengan respons yang diperbarui.

Langkah 2: Buat REST API

Selanjutnya, Anda membuat REST API dengan sumber daya root (`/`).

Untuk membuat REST API

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Lakukan salah satu dari berikut:
   - Untuk membuat API pertama Anda, untuk REST API, pilih Build.
   - Jika Anda pernah membuat API sebelumnya, pilih Create API, lalu pilih Build for REST API.
3. Untuk nama API, masukkan**`my-rest-api`**.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Tetap tetapkan jenis endpoint API ke Regional.
6. Pilih Buat API.

Langkah 3: Buat integrasi proxy Lambda

Selanjutnya, Anda membuat metode API untuk REST API di root resource (/) dan mengintegrasikan metode tersebut dengan fungsi Lambda Anda menggunakan integrasi proxy. Dalam integrasi proxy Lambda, API Gateway meneruskan permintaan masuk dari klien langsung ke fungsi Lambda.

Untuk membuat integrasi proxy Lambda

1. Pilih / sumber daya, lalu pilih Create method.
2. Untuk jenis Metode, pilihANY.
3. Untuk jenis Integrasi, pilih Lambda.
4. Aktifkan integrasi proxy Lambda.
5. Untuk fungsi Lambda, masukkan**`my-function`**, lalu pilih fungsi Lambda Anda.
6. Pilih metode Buat.

Langkah 4: Menerapkan API Anda

Selanjutnya, Anda membuat penerapan API dan mengaitkannya dengan sebuah panggung.

Untuk men-deploy API Anda

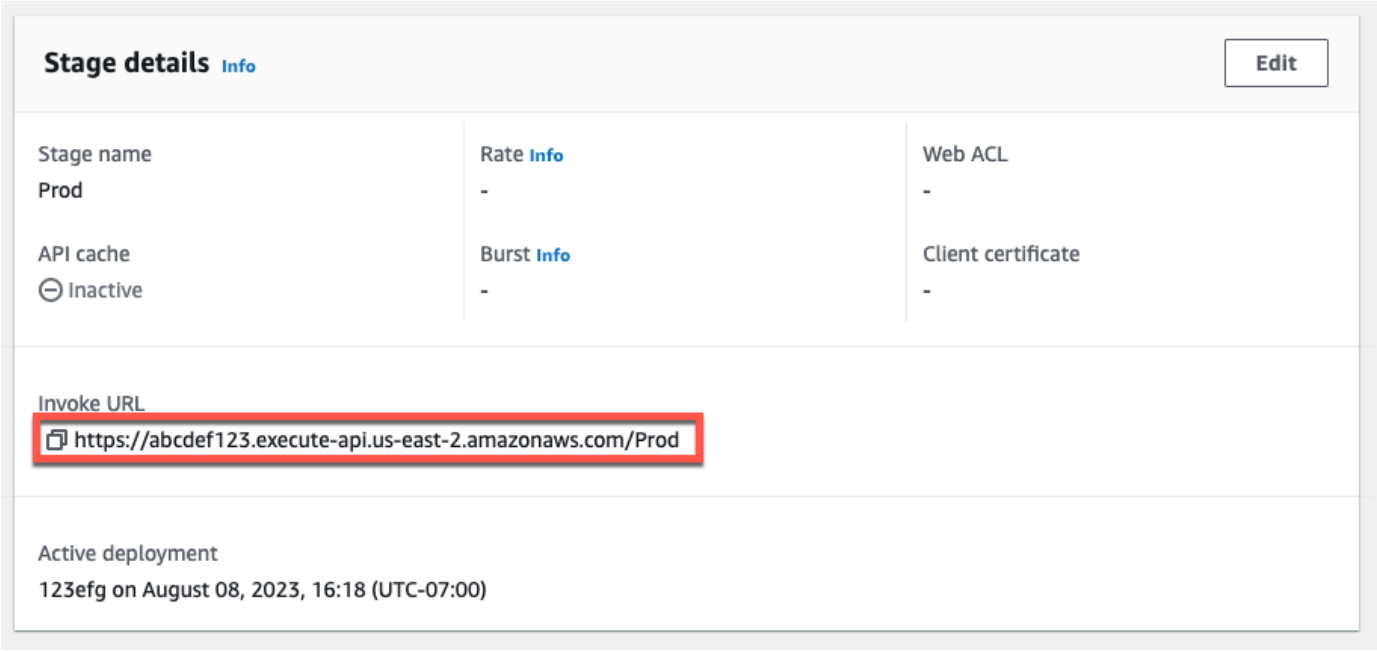
1. Pilih Deploy API.
2. Untuk Stage, pilih New stage.
3. Untuk nama Panggung, masukkan**`Prod`**.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Pilih Deploy.

Sekarang klien dapat menghubungi API Anda. Untuk menguji API Anda sebelum menerapkannya, Anda dapat memilih metode APAPUN secara opsional, menavigasi ke tab Uji, lalu pilih Uji.

Langkah 5: Panggil API Anda

Untuk menjalankan API Anda

1. Dari panel navigasi utama, pilih Stage.
2. Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda.



3. Masukkan URL pemanggilan di browser web.

   URL lengkap akan terlihat seperti`https://abcdef123.execute-api.us-east-2.amazonaws.com/Prod`.

   Browser Anda mengirimkan `GET` permintaan ke API.
4. Verifikasi respons API Anda. Anda akan melihat teks `"The API Gateway REST API console is great!"` di browser Anda.

(Opsional) Langkah 6: Bersihkan

Untuk mencegah biaya yang tidak perlu untuk AndaAkun AWS, hapus sumber daya yang Anda buat sebagai bagian dari latihan ini. Langkah-langkah berikut menghapus REST API, fungsi Lambda, dan sumber daya terkait.

Untuk menghapus REST API

1. Di panel Sumber Daya, pilih tindakan API, Hapus API.
2. Di kotak dialog Delete API, masukkan konfirmasi, lalu pilih Hapus.

Untuk menghapus fungsi Lambda Anda

1. Masuk ke konsol Lambda di https://console.aws.amazon.com/lambda.

2. Pada halaman Fungsi, pilih fungsi Anda. Pilih Actions (Tindakan), Delete (Hapus).
3. Di kotak dialog Hapus 1 fungsi, masukkan**`delete`**, lalu pilih Hapus.

Untuk menghapus grup log fungsi Lambda Anda

1. Buka Halaman grup log di konsol Amazon CloudWatch.
2. Pada halaman Grup log, pilih grup log fungsi Anda (`/aws/lambda/my-function`). Kemudian, untuk Tindakan, pilih Hapus grup log.
3. Di kotak dialog Hapus grup log, pilih Hapus.

Untuk menghapus peran eksekusi fungsi Lambda Anda

1. Buka halaman Peran dari konsol IAM.
2. (Opsional) Pada halaman Peran, di kotak pencarian, masukkan**`my-function`**.
3. Pilih peran fungsi Anda (misalnya,`my-function-`*`31exxmpl`*), lalu pilih Hapus.
4. Di Hapus**`my-function-`*`31exxmpl`***? kotak dialog, masukkan nama peran, lalu pilih Hapus.

> **Tip**
>
> Anda dapat mengotomatiskan pembuatan dan pembersihan AWS sumber daya dengan menggunakan AWS CloudFormation atau AWS Serverless Application Model ()AWS SAM. Untuk beberapa contoh AWS CloudFormation template, lihat contoh template untuk API Gateway di repositori awsdocs GitHub .

Prasyarat untuk memulai dengan API Gateway

Sebelum Anda menggunakan Amazon API Gateway untuk pertama kalinya, selesaikan tugas-tugas berikut.

Daftar Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk secara online.

   Anda akan diminta untuk menerima panggilan telepon dan memasukkan kode verifikasi pada keypad telepon sebagai bagian dari prosedur pendaftaran.

   Saat Anda mendaftar Akun AWS, Pengguna root akun AWS dibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya dalam akun. Sebagai praktik terbaik keamanan, tetapkan akses administratif ke pengguna administratif, dan hanya gunakan pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

AWS mengirimi Anda email konfirmasi setelah proses pendaftaran selesai. Anda dapat melihat aktivitas akun saat ini dan mengelola akun dengan mengunjungi https://aws.amazon.com/ dan memilih Akun Saya.

Membuat pengguna administratif

Setelah Anda mendaftarAkun AWS, amankanPengguna root akun AWS, aktifkanAWS IAM Identity Center, dan buat pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Mengamankan Pengguna root akun AWS Anda

1. Masuk ke AWS Management Console sebagai pemilik akun dengan memilih Pengguna root dan memasukkan alamat email Akun AWS Anda. Pada halaman berikutnya, masukkan kata sandi Anda.

Untuk bantuan masuk menggunakan pengguna root, silakan lihat Masuk sebagai pengguna root dalam Panduan Pengguna AWS Sign-In.

2. Mengaktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

   Untuk petunjuk, silakan lihat Mengaktifkan perangkat MFA virtual untuk pengguna root Akun AWS Anda (konsol) dalam Panduan Pengguna IAM.

Membuat pengguna administratif

1. Aktifkan Pusat Identitas IAM.

   Untuk petunjuk, lihat Mengaktifkan AWS IAM Identity Center di Panduan AWS IAM Identity Center Pengguna.

2. Di Pusat Identitas IAM, berikan akses administratif ke pengguna administratif.

   Untuk tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, lihat Mengkonfigurasi akses pengguna dengan default Direktori Pusat Identitas IAM di Panduan AWS IAM Identity Center Pengguna.

Masuk sebagai pengguna administratif

- Untuk masuk dengan pengguna Pusat Identitas IAM, gunakan URL masuk yang dikirim ke alamat email saat Anda membuat pengguna Pusat Identitas IAM.

  Untuk bantuan masuk menggunakan pengguna Pusat Identitas IAM, silakan lihat Masuk ke portal akses AWS dalam Panduan Pengguna AWS Sign-In.

Memulai dengan API Gateway

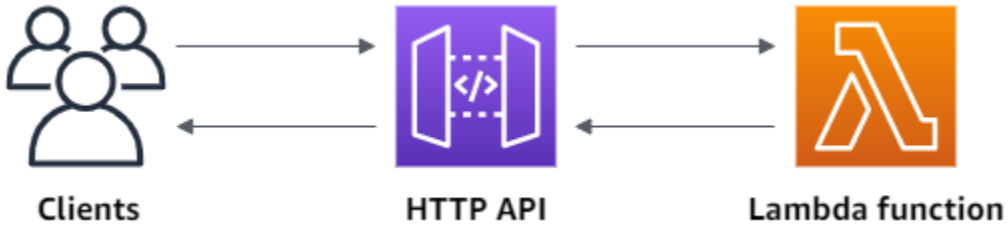
> **Note**
>
> Pengalaman konsol API Gateway yang didesain ulang sekarang tersedia. Untuk tutorial tentang cara menggunakan konsol untuk membuat REST API, lihatMemulai dengan konsol REST API.

Dalam latihan memulai ini, Anda membuat API tanpa server. API tanpa server memungkinkan Anda fokus pada aplikasi Anda, alih-alih menghabiskan waktu untuk menyediakan dan mengelola server. Latihan ini membutuhkan waktu kurang dari 20 menit untuk diselesaikan, dan dimungkinkan dalam Tingkat AWS Gratis.

Pertama, Anda membuat fungsi Lambda menggunakan konsol. AWS Lambda Selanjutnya, Anda membuat API HTTP menggunakan konsol API Gateway. Kemudian, Anda menjalankan API Anda.

> **Note**
>
> Latihan ini menggunakan API HTTP untuk kesederhanaan. API Gateway juga mendukung REST API, yang mencakup lebih banyak fitur. Untuk mempelajari selengkapnya, lihat the section called “Memilih antara REST API dan HTTP API ”.

Saat Anda memanggil API HTTP, API Gateway merutekan permintaan ke fungsi Lambda Anda. Lambda menjalankan fungsi Lambda dan mengembalikan respons ke API Gateway. API Gateway kemudian mengembalikan respons kepada Anda.



Untuk menyelesaikan latihan ini, Anda memerlukan AWS akun dan AWS Identity and Access Management pengguna dengan akses konsol. Untuk informasi selengkapnya, lihat Prasyarat.

Topik

Langkah 1: Membuat fungsi Lambda

Anda menggunakan fungsi Lambda untuk backend API Anda. Lambda menjalankan kode Anda hanya saat diperlukan dan menskalakan secara otomatis, dari beberapa permintaan per hari hingga ribuan per detik.

Untuk contoh ini, Anda menggunakan fungsi Node.js default dari konsol Lambda.

Untuk membuat fungsi Lambda

1. Masuk ke konsol Lambda di https://console.aws.amazon.com/lambda.
2. Pilih Buat fungsi.
3. Untuk Nama fungsi, masukkan **`my-function`**.
4. Pilih Buat fungsi.

Fungsi contoh mengembalikan `200` respon ke klien, dan teks`Hello from Lambda!`.

Anda dapat memodifikasi fungsi Lambda, selama respons fungsi selaras dengan format yang dibutuhkan API Gateway.

Kode fungsi Lambda default akan terlihat mirip dengan berikut ini:

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

Langkah 2: Buat API HTTP

Selanjutnya, Anda membuat API HTTP. API Gateway juga mendukung REST WebSocket API dan API, tetapi HTTP API adalah pilihan terbaik untuk latihan ini. REST API mendukung lebih banyak fitur daripada API HTTP, tetapi kami tidak memerlukan fitur tersebut untuk latihan ini. HTTP API dirancang dengan fitur minimal sehingga dapat ditawarkan dengan harga lebih murah. WebSocket API mempertahankan koneksi persisten dengan klien untuk komunikasi dupleks penuh, yang tidak diperlukan untuk contoh ini.

HTTP API menyediakan endpoint HTTP untuk fungsi Lambda Anda. API Gateway merutekan permintaan ke fungsi Lambda Anda, lalu mengembalikan respons fungsi ke klien.

Untuk membuat API HTTP

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Lakukan salah satu dari berikut:
   - Untuk membuat API pertama Anda, untuk HTTP API, pilih Build.
   - Jika Anda telah membuat API sebelumnya, pilih Buat API, lalu pilih Build for HTTP API.
3. Untuk Integrasi, pilih Tambahkan integrasi.
4. Pilih Lambda.
5. Untuk fungsi Lambda, masukkan. **`my-function`**
6. Untuk nama API, masukkan**`my-http-api`**.
7. Pilih Selanjutnya.
8. Tinjau rute yang dibuat API Gateway untuk Anda, lalu pilih Berikutnya.
9. Tinjau tahap yang dibuat API Gateway untuk Anda, lalu pilih Berikutnya.
10. Pilih Create (Buat).

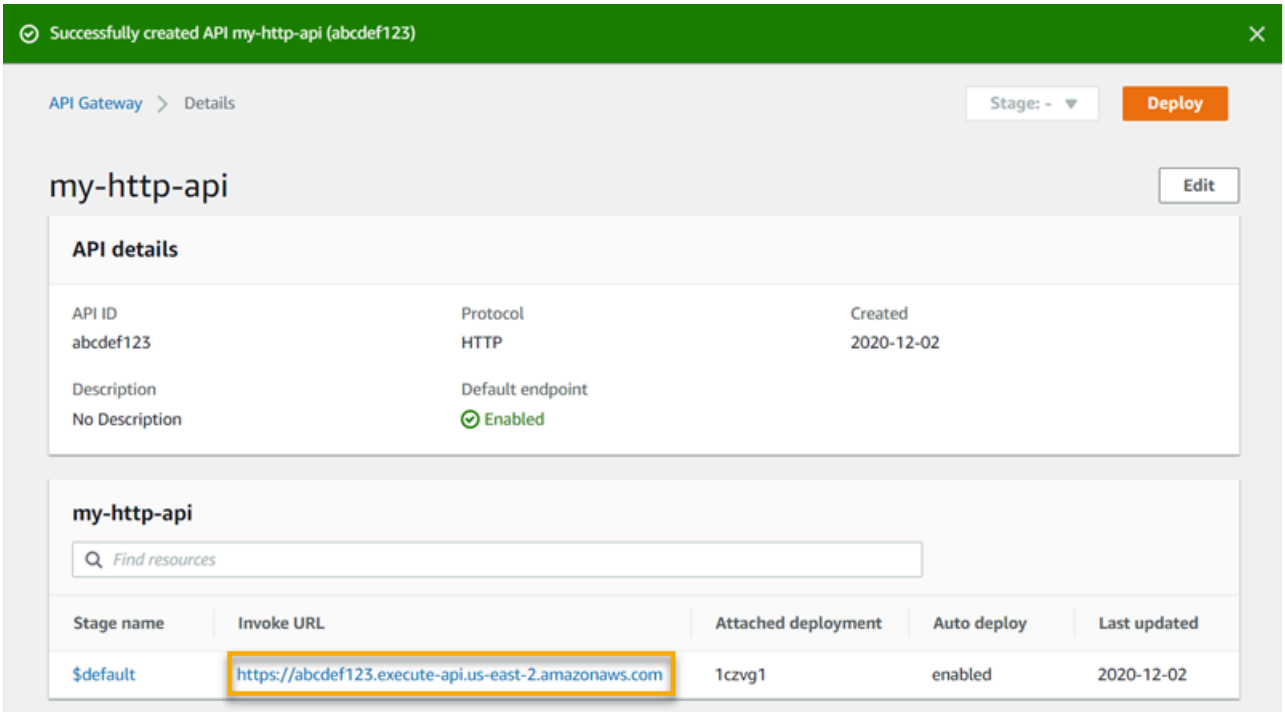
Sekarang Anda telah membuat API HTTP dengan integrasi Lambda yang siap menerima permintaan dari klien.

Langkah 3: Uji API Anda

Selanjutnya, Anda menguji API Anda untuk memastikan bahwa itu berfungsi. Untuk mempermudah, gunakan browser web untuk menjalankan API Anda.

Untuk menguji API Anda

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API Anda.
3. Perhatikan URL pemanggilan API Anda.



4. Salin URL pemanggilan API Anda, dan masukkan di browser web. Tambahkan nama fungsi Lambda Anda ke URL pemanggilan Anda untuk memanggil fungsi Lambda Anda. Secara default, konsol API Gateway membuat rute dengan nama yang sama dengan fungsi Lambda Anda. `my-function`

   URL lengkap akan terlihat seperti`https://abcdef123.execute-api.us-east-2.amazonaws.com/my-function`.

   Browser Anda mengirimkan `GET` permintaan ke API.
5. Verifikasi respons API Anda. Anda akan melihat teks `"Hello from Lambda!"` di browser Anda.

(Opsional) Langkah 4: Bersihkan

Untuk mencegah biaya yang tidak perlu, hapus sumber daya yang Anda buat sebagai bagian dari latihan memulai ini. Langkah-langkah berikut menghapus API HTTP Anda, fungsi Lambda Anda, dan sumber daya terkait.

Untuk menghapus API HTTP

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pada halaman API, pilih API. Pilih Actions (Tindakan), lalu pilih Delete (Hapus).
3. Pilih Delete (Hapus).

Untuk menghapus fungsi Lambda

1. Masuk ke konsol Lambda di https://console.aws.amazon.com/lambda.
2. Pada halaman Fungsi, pilih fungsi. Pilih Actions (Tindakan), lalu pilih Delete (Hapus).
3. Pilih Delete (Hapus).

Untuk menghapus grup log fungsi Lambda

1. Di CloudWatch konsol Amazon, buka halaman Grup log.
2. Pada halaman Grup log, pilih grup log fungsi (`/aws/lambda/my-function`). Pilih Tindakan, lalu pilih Hapus grup log.
3. Pilih Delete (Hapus).

Untuk menghapus peran eksekusi fungsi Lambda

1. Di AWS Identity and Access Management konsol, buka halaman Peran.
2. Pilih peran fungsi, misalnya,`my-function-31exxmpl`.
3. Pilih Hapus peran.
4. Pilih Ya, hapus.

Anda dapat mengotomatiskan pembuatan dan pembersihan AWS sumber daya dengan menggunakan AWS CloudFormation atau. AWS SAM Misalnya AWS CloudFormation template, lihat contoh AWS CloudFormation template.

Langkah selanjutnya

Untuk contoh ini, Anda menggunakan AWS Management Console untuk membuat HTTP API sederhana. HTTP API memanggil fungsi Lambda dan mengembalikan respons ke klien.

Berikut ini adalah langkah selanjutnya saat Anda terus bekerja dengan API Gateway.

- Konfigurasikan jenis integrasi API tambahan, termasuk:
  - Titik akhir HTTP
  - Sumber daya pribadi dalam VPC, seperti layanan Amazon ECS
  - AWSlayanan seperti Amazon Simple Queue ServiceAWS Step Functions, dan Kinesis Data Streams
- Kontrol akses ke API Anda
- Aktifkan pencatatan untuk API Anda
- Konfigurasikan throttling untuk API Anda
- Konfigurasikan domain khusus untuk API Anda

Untuk mendapatkan bantuan terkait Amazon API Gateway dari komunitas, lihat Forum Diskusi API Gateway. Ketika Anda memasuki forum ini, AWS mungkin mengharuskan Anda untuk masuk.

Untuk mendapatkan bantuan dengan API Gateway langsung dariAWS, lihat opsi dukungan di halaman AWS Support.

Lihat juga Pertanyaan yang sering diajukan (FAQ), atau hubungi kami langsung.

Tutorial dan lokakarya Amazon API Gateway

Tutorial dan lokakarya berikut menyediakan latihan langsung untuk membantu Anda mempelajari API Gateway.

Tutorial REST API

- Bangun API REST API Gateway dengan integrasi Lambda
- Tutorial: Buat REST API dengan mengimpor contoh
- Membangun API REST API Gateway dengan integrasi HTTP
- Tutorial: Membangun REST API dengan integrasi pribadi API Gateway
- Tutorial: Membangun API API Gateway REST dengan AWS integrasi
- Tutorial: Buat `Calc` REST API dengan dua integrasi AWS layanan dan satu integrasi non-proxy Lambda
- Tutorial: Membuat REST API sebagai proxy Amazon S3 di API Gateway
- Tutorial: Membuat REST API sebagai proxy Amazon Kinesis di API Gateway
- Tutorial: Membangun REST API pribadi

HTTP API tutorial

- Tutorial: Bangun API CRUD dengan Lambda dan DynamoDB
- Tutorial: Membangun API HTTP dengan integrasi pribadi ke layanan Amazon ECS

WebSocket Tutorial API

- Tutorial: Membangun aplikasi obrolan tanpa server dengan WebSocket API, Lambda, dan DynamoDB

Lokakarya

- Membangun aplikasi web tanpa server
- CI/CD untuk aplikasi tanpa server
- Lokakarya keamanan tanpa server
- Manajemen identitas tanpa server, otentikasi dan otorisasi

- Lokakarya Amazon API Gateway

Tutorial API REST API Amazon API Gateway

Tutorial berikut menyediakan latihan langsung untuk membantu Anda mempelajari API API Gateway REST API.

Topik

- Bangun API REST API Gateway dengan integrasi Lambda
- Tutorial: Buat REST API dengan mengimpor contoh
- Membangun API REST API Gateway dengan integrasi HTTP
- Tutorial: Membangun REST API dengan integrasi pribadi API Gateway
- Tutorial: Membangun API API Gateway REST dengan AWS integrasi
- Tutorial: Buat Calc REST API dengan dua integrasi AWS layanan dan satu integrasi non-proxy Lambda
- Tutorial: Membuat REST API sebagai proxy Amazon S3 di API Gateway
- Tutorial: Membuat REST API sebagai proxy Amazon Kinesis di API Gateway
- Tutorial: Membangun REST API pribadi

Bangun API REST API Gateway dengan integrasi Lambda

Untuk membangun API dengan integrasi Lambda, Anda dapat menggunakan integrasi proxy Lambda atau integrasi non-proxy Lambda.

Dalam integrasi proxy Lambda, input ke fungsi Lambda terintegrasi dapat dinyatakan sebagai kombinasi header permintaan, variabel jalur, parameter string kueri, dan isi. Selain itu, fungsi Lambda dapat menggunakan pengaturan konfigurasi API untuk memengaruhi logika eksekusinya. Untuk pengembang API, menyiapkan integrasi proxy Lambda itu sederhana. Selain memilih fungsi Lambda tertentu di wilayah tertentu, Anda tidak memiliki banyak hal lain yang harus dilakukan. API Gateway mengonfigurasi permintaan integrasi dan respons integrasi untuk Anda. Setelah disiapkan, metode API terintegrasi dapat berkembang dengan backend tanpa memodifikasi pengaturan yang ada. Ini dimungkinkan karena pengembang fungsi Lambda backend mem-parsing data permintaan yang masuk dan merespons dengan hasil yang diinginkan ke klien ketika tidak ada yang salah atau merespons dengan pesan kesalahan ketika terjadi kesalahan.

Dalam integrasi non-proxy Lambda, Anda harus memastikan bahwa input ke fungsi Lambda disediakan sebagai payload permintaan integrasi. Ini menyiratkan bahwa Anda, sebagai pengembang API, harus memetakan data input apa pun yang diberikan klien sebagai parameter permintaan ke dalam badan permintaan integrasi yang tepat. Anda mungkin juga perlu menerjemahkan badan permintaan yang disediakan klien ke dalam format yang dikenali oleh fungsi Lambda.

Topik

- Tutorial: Bangun API REST Hello World dengan integrasi proxy Lambda
- Tutorial: Bangun API REST API Gateway dengan integrasi proxy Lambda lintas akun
- Tutorial: Membangun API REST API Gateway dengan integrasi non-proxy Lambda

Tutorial: Bangun API REST Hello World dengan integrasi proxy Lambda

Integrasi proxy Lambda adalah jenis integrasi API Gateway API yang ringan dan fleksibel yang memungkinkan Anda mengintegrasikan metode API — atau seluruh API — dengan fungsi Lambda. Fungsi Lambda dapat ditulis dalam bahasa apa pun yang didukung Lambda. Karena ini adalah integrasi proxy, Anda dapat mengubah implementasi fungsi Lambda kapan saja tanpa perlu menerapkan ulang API Anda.

Dalam tutorial ini, Anda melakukan hal-hal berikut:

- Buat “Halo, Dunia!” Lambda berfungsi sebagai backend untuk API.
- Buat dan uji “Halo, Dunia!” API dengan integrasi proxy Lambda.

Topik

- Buat “Halo, Dunia!” Fungsi Lambda
- Buat “Halo, Dunia!” API
- Menerapkan dan menguji API

Buat “Halo, Dunia!” Fungsi Lambda

Untuk membuat “Halo, Dunia!” Fungsi Lambda di konsol Lambda

1. Masuk ke konsol Lambda di https://console.aws.amazon.com/lambda.
2. Pada bilah AWS navigasi, pilih Wilayah (misalnya, AS Timur (Virginia N.)).

> **Note**
>
> Perhatikan wilayah tempat Anda membuat fungsi Lambda. Anda akan membutuhkannya saat membuat API.

3. Pilih Fungsi di panel navigasi.
4. Pilih Buat fungsi.
5. Pilih Penulis dari scratch.
6. Di bagian Informasi dasar, lakukan hal berikut:
   a. Dalam nama Fungsi, masukkan**`GetStartedLambdaProxyIntegration`**.
   b. Untuk Runtime, pilih runtime Node.js atau Python terbaru yang didukung.
   c. (Opsional) Di bagian Izin, luaskan Ubah peran eksekusi default. Untuk daftar dropdown peran eksekusi, pilih Buat peran baru dari templat AWS kebijakan.
   d. Dalam nama Peran, masukkan**`GetStartedLambdaBasicExecutionRole`**.
   e. Biarkan bidang Policy templates kosong.
   f. Pilih Buat fungsi.
7. Di bawah Kode fungsi, di editor kode sebaris, salin/tempel kode berikut:

   Node.js

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

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

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

Python

```
import json


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

    greeter = 'World'

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

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

    try:
```

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

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

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

    return res
```

8. Pilih Deploy.

Buat “Halo, Dunia!” API

Sekarang buat API untuk “Hello, World!” Lambda berfungsi dengan menggunakan konsol API Gateway.

Untuk membuat “Halo, Dunia!” API

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Jika ini adalah pertama kalinya Anda menggunakan API Gateway, Anda akan melihat halaman yang memperkenalkan Anda ke fitur layanan. Di bawah REST API, pilih Build. Saat munculan Create Example API muncul, pilih OK.

   Jika ini bukan pertama kalinya Anda menggunakan API Gateway, pilih Buat API. Di bawah REST API, pilih Build.
3. Untuk nama API, masukkan**`LambdaProxyAPI`**.

4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Tetap tetapkan jenis endpoint API ke Regional.
6. Pilih Buat API.

Setelah membuat API, Anda membuat sumber daya. Biasanya, sumber daya API diatur dalam pohon sumber daya sesuai dengan logika aplikasi. Untuk contoh ini, Anda membuat sumber daya / helloworld.

Untuk membuat sumber daya

1. Pilih sumber daya/, lalu pilih Buat sumber daya.
2. Matikan sumber daya Proxy.
3. Pertahankan jalur Sumber Daya sebagai`/`.
4. Untuk Nama sumber daya, masukkan **`helloworld`**.
5. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).
6. Pilih Buat sumber daya.

Dalam integrasi proxy, seluruh permintaan dikirim ke fungsi Lambda backend apa adanya, melalui metode catch-all `ANY` yang mewakili metode HTTP apa pun. Metode HTTP yang sebenarnya ditentukan oleh klien pada waktu berjalan. `ANY`Metode ini memungkinkan Anda menggunakan penyiapan metode API tunggal untuk semua metode HTTP yang didukung:`DELETE`,`GET`,`HEAD`,`OPTIONS`,`PATCH`,`POST`, dan`PUT`.

Untuk membuat **`ANY`** metode

1. Pilih sumber daya /helloworld, lalu pilih Create method.
2. Untuk jenis Metode, pilih APAPUN.
3. Untuk jenis Integrasi, pilih fungsi Lambda.
4. Aktifkan integrasi proxy Lambda.
5. Untuk fungsi Lambda, pilih Wilayah AWS tempat Anda membuat fungsi Lambda Anda, lalu masukkan nama fungsi.
6. Untuk menggunakan nilai batas waktu default 29 detik, tetap aktifkan batas waktu default. Untuk menetapkan batas waktu kustom, pilih Batas waktu default dan masukkan nilai batas waktu antara `50` dan milidetik. `29000`

7. Pilih metode Buat.

Menerapkan dan menguji API

Untuk men-deploy API Anda

1. Pilih Deploy API.
2. Untuk Stage, pilih New stage.
3. Untuk nama Panggung, masukkan**`test`**.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Pilih Deploy.
6. Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda.

Gunakan browser dan cURL untuk menguji API dengan integrasi proxy Lambda

Anda dapat menggunakan browser atau cURL untuk menguji API Anda.

Untuk menguji `GET` permintaan hanya menggunakan parameter string kueri, Anda dapat mengetikkan URL untuk `helloworld` sumber daya API ke dalam bilah alamat browser. Sebagai contoh: https://***`r275xc9bmd`***.execute-api.***`us-east-1`***.amazonaws.com/test/helloworld?name=John&city=Seattle

Untuk metode lain, Anda harus menggunakan utilitas pengujian REST API yang lebih canggih, seperti POSTMAN atau cURL. Tutorial ini menggunakan cURL. Contoh perintah cURL di bawah ini mengasumsikan bahwa cURL diinstal pada komputer Anda.

Untuk menguji API yang Anda gunakan menggunakan cURL:

1. Buka jendela terminal.
2. Salin perintah cURL berikut dan tempel ke jendela terminal, dan ganti URL pemanggilan dengan yang Anda salin di langkah sebelumnya dan tambahkan **`/helloworld`** ke akhir URL.

> **Note**
>
> Jika Anda menjalankan perintah di Windows, gunakan sintaks ini sebagai gantinya:

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

a. Untuk memanggil API dengan parameter string kueri`?greeter=John`:

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

b. Untuk memanggil API dengan parameter header`greeter:John`:

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

c. Untuk memanggil API dengan badan`{"greeter":"John"}`:

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

Dalam semua kasus, outputnya adalah respons 200 dengan badan respons berikut:

```
Hello, John!
```

Tutorial: Bangun API REST API Gateway dengan integrasi proxy Lambda lintas akun

Anda sekarang dapat menggunakan AWS Lambda fungsi dari AWS akun yang berbeda sebagai backend integrasi API Anda. Setiap akun dapat berada di wilayah mana pun di mana Amazon API Gateway tersedia. Ini memudahkan untuk mengelola dan berbagi fungsi backend Lambda secara terpusat di beberapa API.

Di bagian ini, kami menunjukkan cara mengonfigurasi integrasi proxy Lambda lintas akun menggunakan konsol Amazon API Gateway.

Buat API untuk integrasi Lambda lintas akun API Gateway

Untuk membuat API

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Jika ini adalah pertama kalinya Anda menggunakan API Gateway, Anda akan melihat halaman yang memperkenalkan Anda ke fitur layanan. Di bawah REST API, pilih Build. Saat munculan Create Example API muncul, pilih OK.

   Jika ini bukan pertama kalinya Anda menggunakan API Gateway, pilih Buat API. Di bawah REST API, pilih Build.
3. Untuk nama API, masukkan**`CrossAccountLambdaAPI`**.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Tetap tetapkan jenis endpoint API ke Regional.
6. Pilih Buat API.

Buat fungsi integrasi Lambda di akun lain

Sekarang Anda akan membuat fungsi Lambda di akun yang berbeda dari yang Anda buat contoh API.

Membuat fungsi Lambda di akun lain

1. Masuk ke konsol Lambda di akun yang berbeda dari akun tempat Anda membuat API Gateway API.
2. Pilih Buat fungsi.
3. Pilih Penulis dari scratch.
4. Di bawah Tulis dari awal, lakukan langkah berikut:
   a. Untuk nama Fungsi, masukkan nama.
   b. Dari daftar drop-down Runtime, pilih runtime Node.js yang didukung.
   c. Di bagian Izin, perluas Pilih atau buat peran eksekusi. Anda dapat membuat peran atau memilih peran yang ada.
   d. Pilih fungsi Buat untuk melanjutkan.
5. Gulir ke bawah ke panel kode Fungsi.

6. Masukkan implementasi fungsi Node.js darithe section called “Tutorial: Hello World API dengan integrasi proxy Lambda”.
7. Pilih Deploy.
8. Perhatikan ARN lengkap untuk fungsi Anda (di sudut kanan atas panel fungsi Lambda). Anda akan membutuhkannya saat membuat integrasi Lambda lintas akun Anda.

Konfigurasikan integrasi Lambda lintas akun

Setelah Anda memiliki fungsi integrasi Lambda di akun yang berbeda, Anda dapat menggunakan konsol API Gateway untuk menambahkannya ke API Anda di akun pertama Anda.

> **Note**
>
> Jika Anda mengonfigurasi otorisasi lintas wilayah dan lintas akun, `sourceArn` yang ditambahkan ke fungsi target harus menggunakan wilayah fungsi, bukan wilayah API.

Setelah membuat API, Anda membuat sumber daya. Biasanya, sumber daya API diatur dalam pohon sumber daya sesuai dengan logika aplikasi. Untuk contoh ini, Anda membuat sumber daya / helloworld.

Untuk membuat sumber daya

1. Pilih sumber daya/, lalu pilih Buat sumber daya.
2. Matikan sumber daya Proxy.
3. Pertahankan jalur Sumber Daya sebagai/.
4. Untuk Nama sumber daya, masukkan **`helloworld`**.
5. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).
6. Pilih Buat sumber daya.

Setelah Anda membuat sumber daya, Anda membuat `GET` metode. Anda mengintegrasikan `GET` metode dengan fungsi Lambda di akun lain.

Untuk membuat **`GET`** metode

1. Pilih sumber daya /helloworld, lalu pilih Create method.
2. Untuk tipe Metode, pilih GET.

3. Untuk jenis Integrasi, pilih fungsi Lambda.
4. Aktifkan integrasi proxy Lambda.
5. Untuk fungsi Lambda, masukkan ARN lengkap fungsi Lambda Anda dari Langkah 1.

   Di konsol Lambda, Anda dapat menemukan ARN untuk fungsi Anda di sudut kanan atas jendela konsol.
6. Saat Anda memasukkan ARN, string `aws lambda add-permission` perintah akan muncul. Kebijakan ini memberikan akses akun pertama Anda ke fungsi Lambda akun kedua Anda. Salin dan tempel string `aws lambda add-permission` perintah ke AWS CLI jendela yang dikonfigurasi untuk akun kedua Anda.
7. Pilih metode Buat.

Anda dapat melihat kebijakan terbaru untuk fungsi Anda di konsol Lambda.

(Opsional) Untuk melihat kebijakan Anda yang diperbarui

1. Masuk ke AWS Management Console dan buka AWS Lambda konsol di https://console.aws.amazon.com/lambda/.
2. Pilih fungsi Lambda Anda.
3. Pilih Izin.

   Anda akan melihat `Allow` kebijakan dengan `Condition` klausa di mana dalam `AWS:SourceArn` adalah ARN untuk metode API `GET` Anda.

Tutorial: Membangun API REST API Gateway dengan integrasi non-proxy Lambda

Dalam panduan ini, kami menggunakan konsol API Gateway untuk membangun API yang memungkinkan klien memanggil fungsi Lambda melalui integrasi non-proxy Lambda (juga dikenal sebagai integrasi khusus). Untuk informasi selengkapnya tentang AWS Lambda dan fungsi Lambda, lihat Panduan AWS Lambda Pengembang.

Untuk memfasilitasi pembelajaran, kami memilih fungsi Lambda sederhana dengan pengaturan API minimal untuk memandu Anda melalui langkah-langkah membangun API Gateway API dengan integrasi kustom Lambda. Bila perlu, kami menjelaskan beberapa logika. Untuk contoh lebih rinci tentang integrasi kustom Lambda, lihat. Tutorial: Buat `Calc` REST API dengan dua integrasi AWS layanan dan satu integrasi non-proxy Lambda

Sebelum membuat API, siapkan backend Lambda dengan membuat fungsi Lambda di, dijelaskan selanjutnya. AWS Lambda

Topik

Buat fungsi Lambda untuk integrasi non-proxy Lambda

> **Note**
>
> Membuat fungsi Lambda dapat mengakibatkan biaya ke akun Anda AWS .

Pada langkah ini, Anda membuat “Halo, Dunia!” -seperti fungsi Lambda untuk integrasi kustom Lambda. Sepanjang panduan ini, fungsi ini disebut. `GetStartedLambdaIntegration`

Implementasi fungsi `GetStartedLambdaIntegration` Lambda ini adalah sebagai berikut:

Node.js

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

console.log('Loading function');

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

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

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

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

Python

```
import json

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


def lambda_handler(event, context):
    print(event)
    # parse the input for the name, city, time, and day property values
    try:
        if event['name']:
            name = event['name']
    except KeyError:
        name = 'you'
    try:
        if event['city']:
            city = event['city']
    except KeyError:
        city = 'World'
    try:
        if event['time'] in times:
```

```
            time = event['time']
        else:
            time = 'day'
    except KeyError:
        time = 'day'
    try:
        if event['day'] in days:
            day = event['day']
        else:
            day = ''
    except KeyError:
        day = ''
    # Generate a greeting
    greeting = 'Good ' + time + ', ' + name + ' of ' + \
        city + '.' + ['', ' Happy ' + day + '!'][day != '']
    # Log the greeting to CloudWatch
    print(greeting)

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

Untuk integrasi kustom Lambda, API Gateway meneruskan input ke fungsi Lambda dari klien sebagai badan permintaan integrasi. `event`Objek dari fungsi handler Lambda adalah input.

Fungsi Lambda kami sederhana. Ini mem-parsing `event` objek input untuk`name`,, `city`, `time`, dan `day` properti. Kemudian mengembalikan salam, sebagai objek JSON dari`{"message":greeting}`, ke penelepon. Pesannya ada dalam `"Good [morning|afternoon|day], [`*`name`*`|you] in [`*`city`*`|World]. Happy `*`day`*`!"` pola. Diasumsikan bahwa input ke fungsi Lambda adalah dari objek JSON berikut:

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

Lihat informasi selengkapnya di Panduan Developer AWS Lambda.

Selain itu, fungsi mencatat eksekusinya ke Amazon CloudWatch dengan meneleponconsole.log(...). Ini berguna untuk melacak panggilan saat men-debug fungsi.

Untuk mengizinkan `GetStartedLambdaIntegration` fungsi mencatat panggilan, tetapkan peran IAM dengan kebijakan yang sesuai untuk fungsi Lambda untuk membuat CloudWatch aliran dan menambahkan entri log ke aliran. Konsol Lambda memandu Anda untuk membuat peran dan kebijakan IAM yang diperlukan.

Jika Anda menyiapkan API tanpa menggunakan konsol API Gateway, seperti saat mengimpor API dari file OpenAPI, Anda harus secara eksplisit membuat, jika perlu, dan menyiapkan peran dan kebijakan pemanggilan untuk API Gateway untuk menjalankan fungsi Lambda. Untuk informasi selengkapnya tentang cara menyiapkan peran pemanggilan dan eksekusi Lambda untuk API Gateway API, lihat. Kontrol akses ke API dengan izin IAM

Dibandingkan dengan`GetStartedLambdaProxyIntegration`, fungsi Lambda untuk integrasi proxy Lambda, fungsi Lambda untuk integrasi kustom `GetStartedLambdaIntegration` Lambda hanya mengambil masukan dari badan permintaan integrasi API Gateway API. Fungsi ini dapat mengembalikan output dari setiap objek JSON, string, angka, Boolean, atau bahkan blob biner. Fungsi Lambda untuk integrasi proxy Lambda, sebaliknya, dapat mengambil input dari data permintaan apa pun, tetapi harus mengembalikan output dari objek JSON tertentu. `GetStartedLambdaIntegration`Fungsi untuk integrasi kustom Lambda dapat memiliki parameter permintaan API sebagai input, asalkan API Gateway memetakan parameter permintaan API yang diperlukan ke badan permintaan integrasi sebelum meneruskan permintaan klien ke backend. Agar hal ini terjadi, pengembang API harus membuat template pemetaan dan mengonfigurasinya pada metode API saat membuat API.

Sekarang, buat fungsi `GetStartedLambdaIntegration` Lambda.

Untuk membuat fungsi **`GetStartedLambdaIntegration`** Lambda untuk integrasi kustom Lambda

1. Buka AWS Lambda konsol di https://console.aws.amazon.com/lambda/.
2. Lakukan salah satu dari cara berikut:
   - Jika halaman selamat datang muncul, pilih Mulai Sekarang dan kemudian pilih Buat fungsi.
   - Jika halaman daftar Lambda > Fungsi muncul, pilih Buat fungsi.
3. Pilih Tulis dari awal.
4. Di panel Penulis dari awal, lakukan hal berikut:
   a. Untuk Nama, masukkan **`GetStartedLambdaIntegration`** sebagai nama fungsi Lambda.
   b. Untuk Runtime, pilih runtime Node.js atau Python terbaru yang didukung.

c. (Opsional) Di bagian Izin, luaskan Ubah peran eksekusi default. Untuk daftar dropdown peran eksekusi, pilih Buat peran baru dari templat AWS kebijakan.

d. Untuk nama Peran, masukkan nama untuk peran Anda (misalnya,**`GetStartedLambdaIntegrationRole`**).

e. Untuk templat Kebijakan, pilih Izin microservice sederhana.

f. Pilih Buat fungsi.

5. Di panel Configure function, di bawah Kode fungsi lakukan hal berikut:

a. Salin kode fungsi Lambda yang tercantum di awal bagian ini dan tempel di editor kode sebaris.

b. Tinggalkan pilihan default untuk semua bidang lain di bagian ini.

c. Pilih Deploy.

6. Untuk menguji fungsi yang baru dibuat, pilih tab Uji.

a. Untuk Nama peristiwa, masukkan **`HelloWorldTest`**.

b. Untuk Event JSON, ganti kode default dengan yang berikut ini.

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

c. Pilih Uji untuk menjalankan fungsi. Hasil Eksekusi: bagian yang berhasil ditampilkan. Perluas Detail dan Anda melihat output berikut.

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

Outputnya juga ditulis ke CloudWatch Log.

Sebagai latihan sampingan, Anda dapat menggunakan konsol IAM untuk melihat peran IAM (`GetStartedLambdaIntegrationRole`) yang dibuat sebagai bagian dari pembuatan fungsi Lambda. Terlampir pada peran IAM ini adalah dua kebijakan inline. Satu menetapkan izin paling dasar untuk eksekusi Lambda. Ini memungkinkan memanggil CloudWatch sumber daya apa

pun dari akun Anda di wilayah tempat fungsi Lambda dibuat. CloudWatch `CreateLogGroup` Kebijakan ini juga memungkinkan pembuatan CloudWatch stream dan peristiwa logging untuk fungsi `HelloWorldForLambdaIntegration` 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:*"
            ]
        }
    ]
}
```

Dokumen kebijakan lainnya berlaku untuk memanggil AWS layanan lain yang tidak digunakan dalam contoh ini. Anda dapat melewatkannya untuk saat ini.

Terkait dengan peran IAM adalah entitas tepercaya, yaitu`lambda.amazonaws.com`. Inilah hubungan kepercayaannya:

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

```
  ]
}
```

Kombinasi hubungan kepercayaan ini dan kebijakan sebaris memungkinkan fungsi Lambda memanggil fungsi untuk mencatat peristiwa ke `console.log()` Log. CloudWatch

Jika Anda tidak menggunakan AWS Management Console untuk membuat fungsi Lambda, Anda harus mengikuti contoh ini untuk membuat peran dan kebijakan IAM yang diperlukan dan kemudian secara manual melampirkan peran ke fungsi Anda.

Buat API dengan integrasi non-proxy Lambda

Dengan fungsi Lambda (`GetStartedLambdaIntegration`) yang dibuat dan diuji, Anda siap mengekspos fungsi tersebut melalui API Gateway API. Untuk tujuan ilustrasi, kami mengekspos fungsi Lambda dengan metode HTTP generik. Kami menggunakan badan permintaan, variabel jalur URL, string kueri, dan header untuk menerima data input yang diperlukan dari klien. Kami mengaktifkan validator permintaan API Gateway untuk API untuk memastikan bahwa semua data yang diperlukan didefinisikan dan ditentukan dengan benar. Kami mengonfigurasi template pemetaan untuk API Gateway untuk mengubah data permintaan yang disediakan klien menjadi format yang valid seperti yang dipersyaratkan oleh fungsi Lambda backend.

Untuk membuat API dengan integrasi non-proxy Lambda

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Jika ini adalah pertama kalinya Anda menggunakan API Gateway, Anda akan melihat halaman yang memperkenalkan Anda ke fitur layanan. Di bawah REST API, pilih Build. Saat munculan Create Example API muncul, pilih OK.

   Jika ini bukan pertama kalinya Anda menggunakan API Gateway, pilih Buat API. Di bawah REST API, pilih Build.
3. Untuk nama API, masukkan**`LambdaNonProxyAPI`**.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Tetap tetapkan jenis endpoint API ke Regional.
6. Pilih Buat API.

Setelah membuat API Anda, Anda membuat sumber daya/{city}. Ini adalah contoh sumber daya dengan variabel jalur yang mengambil masukan dari klien. Kemudian, Anda memetakan variabel jalur ini ke input fungsi Lambda menggunakan template pemetaan.

Untuk membuat sumber daya

1. Pilih Buat sumber daya.
2. Matikan sumber daya Proxy.
3. Pertahankan jalur Sumber Daya sebagai`/`.
4. Untuk Nama sumber daya, masukkan **`{city}`**.
5. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).
6. Pilih Buat sumber daya.

Setelah membuat sumber daya/{city} Anda, Anda membuat `ANY` metode. Kata kerja `ANY` HTTP adalah placeholder untuk metode HTTP yang valid yang dikirimkan klien pada waktu berjalan. Contoh ini menunjukkan bahwa `ANY` metode dapat digunakan untuk integrasi kustom Lambda serta untuk integrasi proxy Lambda.

Untuk membuat **`ANY`** metode

1. Pilih sumber daya/{city}, lalu pilih Create method.
2. Untuk jenis Metode, pilih APAPUN.
3. Untuk jenis Integrasi, pilih fungsi Lambda.
4. Matikan integrasi proxy Lambda.
5. Untuk fungsi Lambda, pilih Wilayah AWS tempat Anda membuat fungsi Lambda Anda, lalu masukkan nama fungsi.
6. Untuk menggunakan nilai batas waktu default 29 detik, tetap aktifkan batas waktu default. Untuk menetapkan batas waktu kustom, pilih Batas waktu default dan masukkan nilai batas waktu antara `50` dan milidetik. `29000`
7. Pilih metode Buat.

Setelah membuat `ANY` metode Anda, Anda mengaktifkan validator permintaan untuk variabel jalur URL, string kueri, dan header untuk memastikan bahwa semua data yang diperlukan didefinisikan dan ditentukan dengan benar. Untuk contoh ini, Anda membuat parameter string `time` kueri dan `day` header.

Untuk mengaktifkan validator permintaan

1. Pada tab Permintaan metode, di bawah Pengaturan permintaan metode, pilih Edit.

2. Untuk validator Permintaan, pilih Validasi parameter string kueri dan header.
3. Pilih parameter string kueri URL dan lakukan hal berikut:
    a. Pilih Tambahkan string kueri.
    b. Untuk Nama, masukkan **`time`**.
    c. Aktifkan Diperlukan.
    d. Tetap caching dimatikan.
4. Pilih header permintaan HTTP dan lakukan hal berikut:
    a. Pilih Tambahkan header.
    b. Untuk Nama, masukkan **`day`**.
    c. Aktifkan Diperlukan.
    d. Tetap caching dimatikan.
5. Pilih Simpan.

Setelah mengaktifkan validator permintaan, Anda mengonfigurasi permintaan integrasi untuk `ANY` metode dengan menambahkan templat pemetaan tubuh untuk mengubah permintaan yang masuk menjadi payload JSON, seperti yang dipersyaratkan oleh fungsi Lambda backend.

Untuk mengkonfigurasi permintaan integrasi

1. Pada tab Permintaan integrasi, di bawah pengaturan permintaan Integrasi, pilih Edit.
2. Untuk Request body passthrough, pilih Bila tidak ada templat yang ditentukan (disarankan).
3. Pilih template Pemetaan.
4. Pilih Tambahkan templat pemetaan.
5. Untuk jenis Konten, masukkan**`application/json`**.
6. Untuk badan Template, masukkan kode berikut:

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

7. Pilih Simpan.

Uji menjalankan metode API

Konsol API Gateway menyediakan fasilitas pengujian bagi Anda untuk menguji pemanggilan API sebelum diterapkan. Anda menggunakan fitur Uji konsol untuk menguji API dengan mengirimkan permintaan berikut:

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

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

Dalam permintaan pengujian ini, Anda akan mengatur `ANY` ke`POST`, mengatur `{city}` ke`Seattle`, menetapkan `Wednesday` sebagai nilai `day` header, dan menetapkan `"John"` sebagai nilai`callerName`.

Untuk menguji **`ANY`** metode

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Untuk jenis Metode, pilih`POST`.
3. Untuk Path, di bawah kota, masuk**`Seattle`**.
4. Untuk string Query, masukkan**`time=morning`**.
5. Untuk Header, masukkan**`day:Wednesday`**.
6. Untuk Badan Permintaan, masukkan**`{ "callerName": "John" }`**.
7. Pilih Uji.

Verifikasi bahwa muatan respons yang dikembalikan adalah sebagai berikut:

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

Anda juga dapat melihat log untuk memeriksa bagaimana API Gateway memproses permintaan dan respons.

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

```
Thu Aug 31 01:07:25 UTC 2017 : Method response headers: {X-Amzn-Trace-
Id=sampled=0;root=1-59a7614d-373151b01b0713127e646635, Content-Type=application/json}
Thu Aug 31 01:07:25 UTC 2017 : Successfully completed execution
Thu Aug 31 01:07:25 UTC 2017 : Method completed with status: 200
```

Log menampilkan permintaan masuk sebelum pemetaan dan permintaan integrasi setelah pemetaan. Ketika pengujian gagal, log berguna untuk mengevaluasi apakah input asli benar atau template pemetaan berfungsi dengan benar.

Terapkan API

Doa tes adalah simulasi dan memiliki keterbatasan. Misalnya, ia melewati mekanisme otorisasi apa pun yang diberlakukan pada API. Untuk menguji eksekusi API secara real time, Anda harus menerapkan API terlebih dahulu. Untuk menerapkan API, Anda membuat panggung untuk membuat snapshot API pada saat itu. Nama stage juga mendefinisikan path dasar setelah nama host default API. Sumber daya root API ditambahkan setelah nama panggung. Saat memodifikasi API, Anda harus menerapkannya kembali ke tahap baru atau yang sudah ada sebelum perubahan diterapkan.

Untuk menerapkan API ke panggung

1. Pilih Deploy API.
2. Untuk Stage, pilih New stage.
3. Untuk nama Panggung, masukkan**`test`**.

   > **Note**
   >
   > Input harus berupa teks yang dikodekan UTF-8 (yaitu, tidak terlokalisasi).

4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Pilih Deploy.

Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda. Pola umum URL dasar API adalah`https://`*`api-id`*`.`*`region`*`.amazonaws.com/`*`stageName`*. Misalnya, URL dasar API (`beags1mnid`) yang dibuat di `us-west-2` wilayah dan diterapkan ke `test` panggung adalah`https://beags1mnid.execute-api.us-west-2.amazonaws.com/test`.

Uji API dalam tahap penerapan

Ada beberapa cara Anda dapat menguji API yang diterapkan. Untuk permintaan GET yang hanya menggunakan variabel jalur URL atau parameter string kueri, Anda dapat memasukkan URL sumber daya API di browser. Untuk metode lain, Anda harus menggunakan utilitas pengujian REST API yang lebih canggih, seperti POSTMAN atau cURL.

Untuk menguji API menggunakan cURL

1. Buka jendela terminal di komputer lokal Anda yang terhubung ke internet.
2. Untuk mengujiPOST `/Seattle?time=evening`:

   Salin perintah cURL berikut dan tempel ke jendela 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"
   }'
   ```

   Anda harus mendapatkan respons yang sukses dengan muatan berikut:

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

   Jika Anda POST mengubah PUT permintaan metode ini, Anda mendapatkan respons yang sama.

Hapus

Jika Anda tidak lagi membutuhkan fungsi Lambda yang Anda buat untuk panduan ini, Anda dapat menghapusnya sekarang. Anda juga dapat menghapus sumber daya IAM yang menyertainya.

> **Warning**
>
> Jika Anda berencana untuk menyelesaikan penelusuran lain dalam seri ini, jangan hapus peran eksekusi Lambda atau peran pemanggilan Lambda. Jika Anda menghapus fungsi Lambda yang diandalkan API Anda, API tersebut tidak akan berfungsi lagi. Menghapus fungsi

Lambda tidak dapat dibatalkan. Jika Anda ingin menggunakan fungsi Lambda lagi, Anda harus membuat ulang fungsi tersebut.
Jika Anda menghapus sumber daya IAM yang diandalkan oleh fungsi Lambda, fungsi Lambda tersebut tidak akan berfungsi lagi, dan API apa pun yang bergantung pada fungsi tersebut tidak akan berfungsi lagi. Menghapus sumber daya IAM tidak dapat dibatalkan. Jika Anda ingin menggunakan sumber daya IAM lagi, Anda harus membuat ulang sumber daya.

Untuk menghapus fungsi Lambda

1. Masuk ke AWS Management Console dan buka AWS Lambda konsol di https://console.aws.amazon.com/lambda/.
2. Dari daftar fungsi, pilih GetHelloWorld, pilih Tindakan, lalu pilih Hapus fungsi. Saat diminta, pilih Hapus lagi.
3. Dari daftar fungsi, pilih GetHelloWithName, pilih Tindakan, lalu pilih Hapus fungsi. Saat diminta, pilih Hapus lagi.

Untuk menghapus sumber daya IAM terkait

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Dari Detail, pilih Peran.
3. Dari daftar peran, pilih API GatewayLambdaExecRole, pilih Tindakan Peran, lalu pilih Hapus Peran. Saat diminta, pilih Ya, Hapus.
4. Dari Detail, pilih Kebijakan.
5. Dari daftar kebijakan, pilih API GatewayLambdaExecPolicy, pilih Tindakan Kebijakan, lalu pilih Hapus. Saat diminta, pilih Hapus.

Tutorial: Buat REST API dengan mengimpor contoh

Anda dapat menggunakan konsol Amazon API Gateway untuk membuat dan menguji REST API sederhana dengan integrasi HTTP untuk PetStore situs web. Definisi API telah dikonfigurasi sebelumnya sebagai file OpenAPI 2.0. Setelah memuat definisi API ke dalam API Gateway, Anda dapat menggunakan konsol API Gateway untuk memeriksa struktur dasar API atau cukup menerapkan dan menguji API.

PetStore Contoh API mendukung metode berikut bagi klien untuk mengakses situs backend HTTP dari. `http://petstore-demo-endpoint.execute-api.com/petstore/pets`

> **Note**
>
> Tutorial ini menggunakan endpoint HTTP sebagai contoh. Saat Anda membuat API sendiri, kami sarankan Anda menggunakan titik akhir HTTPS untuk integrasi HTTP Anda.

- `GET /`: untuk akses baca sumber daya root API yang tidak terintegrasi dengan titik akhir backend apa pun. API Gateway merespons dengan ikhtisar PetStore situs web. Ini adalah contoh dari jenis `MOCK` integrasi.
- `GET /pets`: untuk akses baca ke sumber daya API yang terintegrasi dengan `/pets` sumber daya backend `/pets` yang diberi nama serupa. Backend mengembalikan halaman hewan peliharaan yang tersedia di file. PetStore Ini adalah contoh dari jenis `HTTP` integrasi. URL dari titik akhir integrasi adalah`http://petstore-demo-endpoint.execute-api.com/petstore/pets`.
- `POST /pets`: untuk akses tulis ke `/pets` sumber daya API yang terintegrasi dengan sumber daya backend`/petstore/pets`. Setelah menerima permintaan yang benar, backend menambahkan hewan peliharaan yang ditentukan ke PetStore dan mengembalikan hasilnya ke pemanggil. Integrasi juga`HTTP`.
- `GET /pets/{petId}`: untuk akses baca ke hewan peliharaan yang diidentifikasi oleh `petId` nilai sebagaimana ditentukan sebagai variabel jalur dari URL permintaan masuk. Metode ini juga memiliki tipe `HTTP` integrasi. Backend mengembalikan hewan peliharaan tertentu yang ditemukan di file. PetStore URL titik akhir HTTP backend adalah`http://petstore-demo-endpoint.execute-api.com/petstore/pets/`*n*, di mana n bilangan bulat sebagai pengidentifikasi hewan peliharaan yang ditanyakan.

API mendukung akses CORS melalui `OPTIONS` metode jenis `MOCK` integrasi. API Gateway mengembalikan header yang diperlukan yang mendukung akses CORS.

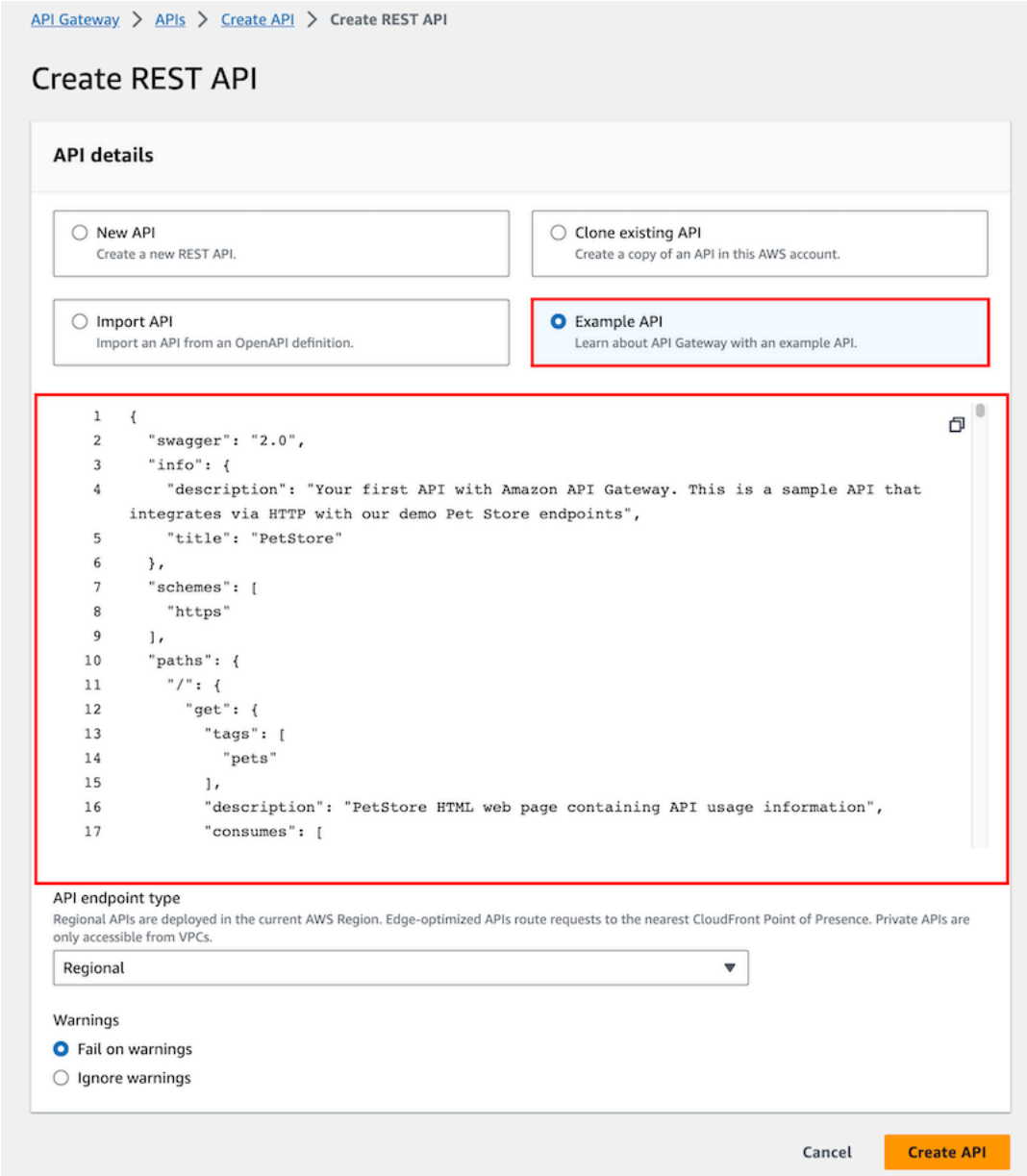
Prosedur berikut memandu Anda melalui langkah-langkah untuk membuat dan menguji API dari contoh menggunakan API Gateway Console.

Untuk mengimpor, membangun, dan menguji contoh API

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Lakukan salah satu dari berikut:

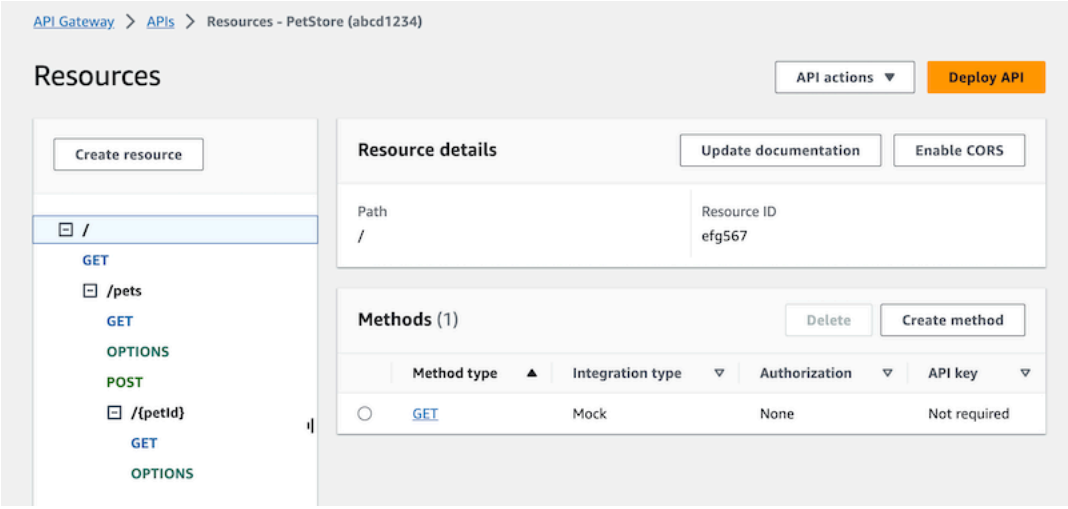
- Untuk membuat API pertama Anda, untuk REST API, pilih Build.
- Jika Anda pernah membuat API sebelumnya, pilih Create API, lalu pilih Build for REST API.

3. Di bawah Create REST API, pilih Example API lalu pilih Create API untuk membuat contoh API.



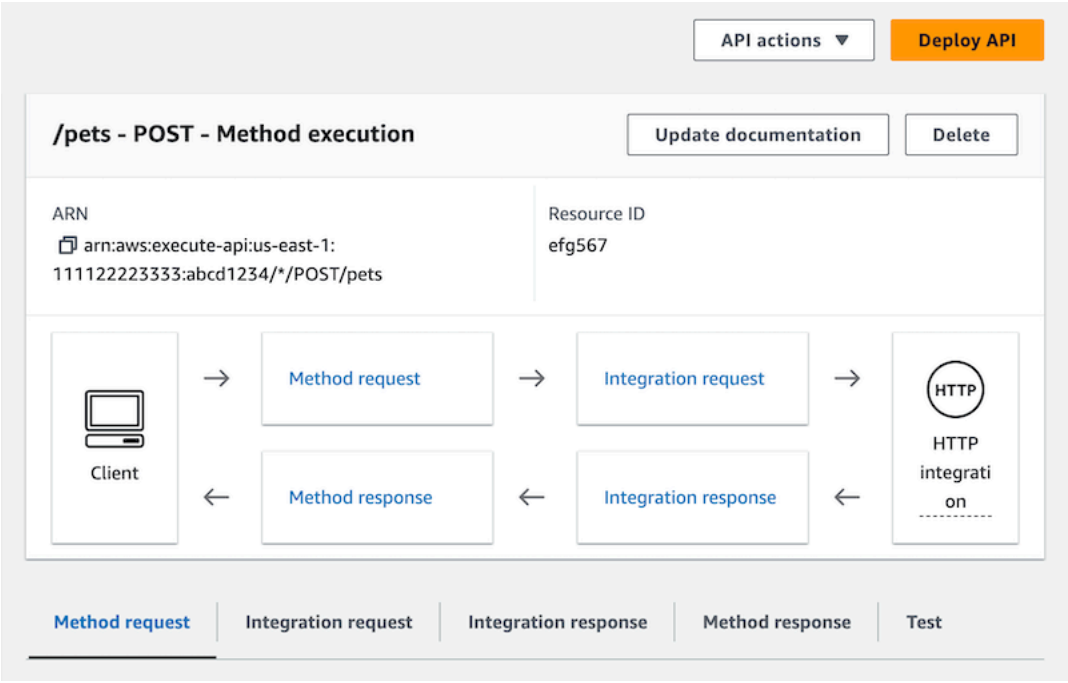
Anda dapat menggulir ke bawah definisi OpenAPI untuk detail contoh API ini sebelum memilih Create API.

4. Di panel navigasi utama, pilih Resources. API yang baru dibuat ditampilkan sebagai berikut:



Panel Resources menunjukkan struktur API yang dibuat sebagai pohon node. Metode API yang didefinisikan pada setiap sumber daya adalah tepi pohon. Ketika sumber daya dipilih, semua metodenya tercantum dalam tabel Metode di sebelah kanan. Ditampilkan dengan setiap metode adalah jenis metode, jenis integrasi, jenis otorisasi, dan persyaratan kunci API.

5. Untuk melihat detail metode, untuk memodifikasi pengaturannya, atau untuk menguji pemanggilan metode, pilih nama metode dari daftar metode atau pohon sumber daya. Di sini, kami memilih `POST /pets` metode sebagai ilustrasi:



Panel eksekusi Metode yang dihasilkan menyajikan tampilan logis dari struktur dan perilaku metode yang dipilih (`POST /pets`).

Permintaan Metode dan respons Metode mewakili antarmuka API dengan frontend, dan permintaan Integrasi serta respons Integrasi mewakili antarmuka API dengan backend.

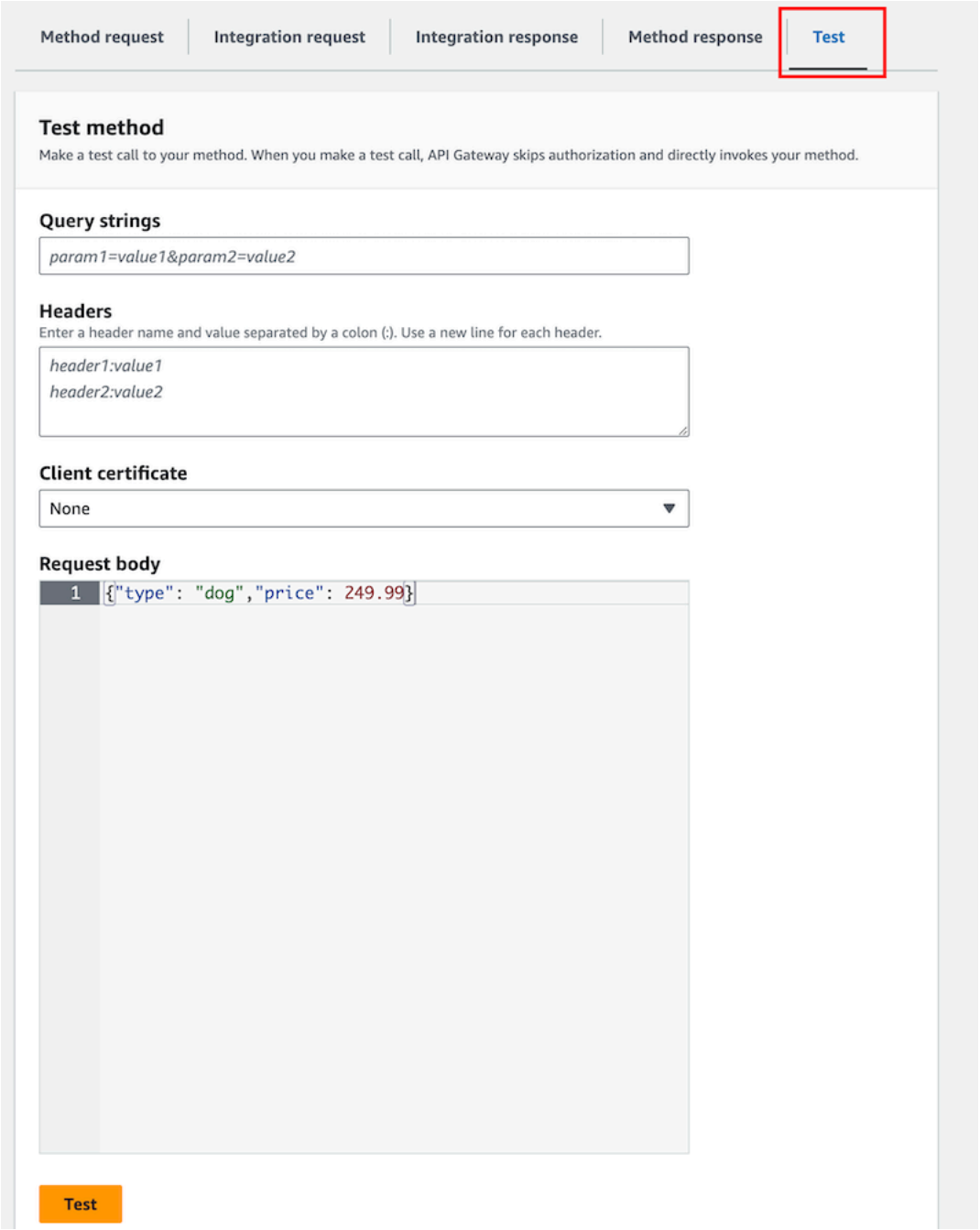
Klien menggunakan API untuk mengakses fitur backend melalui permintaan Metode. API Gateway menerjemahkan permintaan klien, jika perlu, ke dalam formulir yang dapat diterima oleh backend dalam permintaan Integrasi sebelum meneruskan permintaan masuk ke backend. Permintaan yang ditransformasikan dikenal sebagai permintaan integrasi. Demikian pula, backend mengembalikan respons ke API Gateway dalam respons Integrasi. API Gateway kemudian merutekan ke Method Response sebelum mengirimnya ke klien. Sekali lagi, jika perlu, API Gateway dapat memetakan data respons backend ke formulir yang diharapkan oleh klien.

Untuk `POST` metode pada sumber daya API, payload permintaan metode dapat diteruskan ke permintaan integrasi tanpa modifikasi, jika payload permintaan metode memiliki format yang sama dengan payload permintaan integrasi.

Permintaan `GET /` metode menggunakan tipe `MOCK` integrasi dan tidak terikat pada titik akhir backend nyata. Respons Integrasi yang sesuai diatur untuk mengembalikan halaman HTML statis. Ketika metode dipanggil, API Gateway hanya menerima permintaan dan segera mengembalikan respons integrasi yang dikonfigurasi ke klien melalui respons Metode. Anda dapat menggunakan integrasi tiruan untuk menguji API tanpa memerlukan titik akhir backend. Anda juga dapat menggunakannya untuk menyajikan respons lokal, yang dihasilkan dari template pemetaan badan respons.

Sebagai pengembang API, Anda mengontrol perilaku interaksi frontend API Anda dengan mengonfigurasi permintaan metode dan respons metode. Anda mengontrol perilaku interaksi backend API Anda dengan menyiapkan permintaan integrasi dan respons integrasi. Ini melibatkan pemetaan data antara metode dan integrasi yang sesuai. Untuk saat ini, kami fokus pada pengujian API untuk memberikan pengalaman end-to-end pengguna.

6. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
7. Misalnya, untuk menguji `POST /pets` metode, masukkan **`{"type": "dog","price": 249.99}`** payload berikut ke dalam badan Permintaan, lalu pilih Uji.



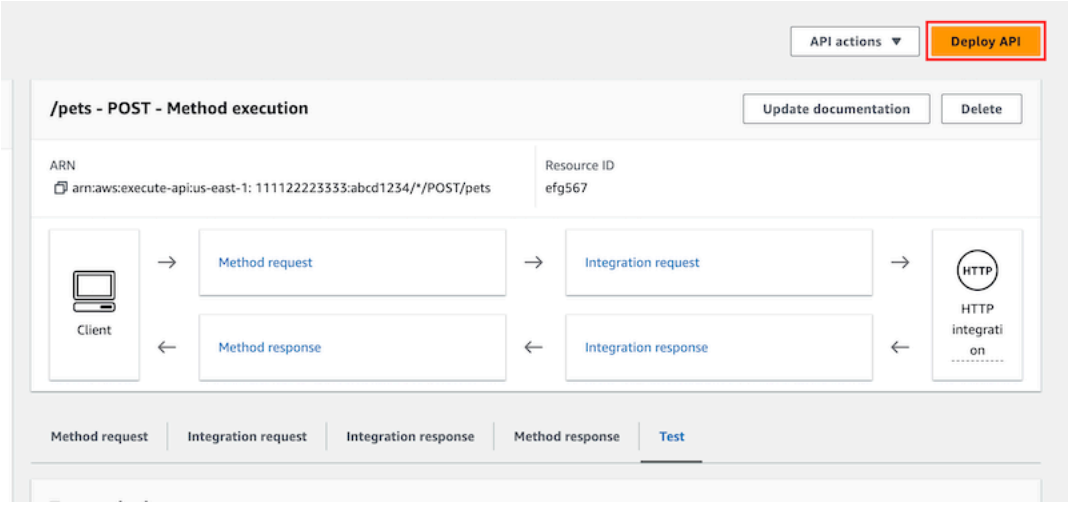
Input menentukan atribut hewan peliharaan yang ingin kita tambahkan ke daftar hewan peliharaan di situs PetStore web.

8. Hasil ditampilkan sebagai berikut:

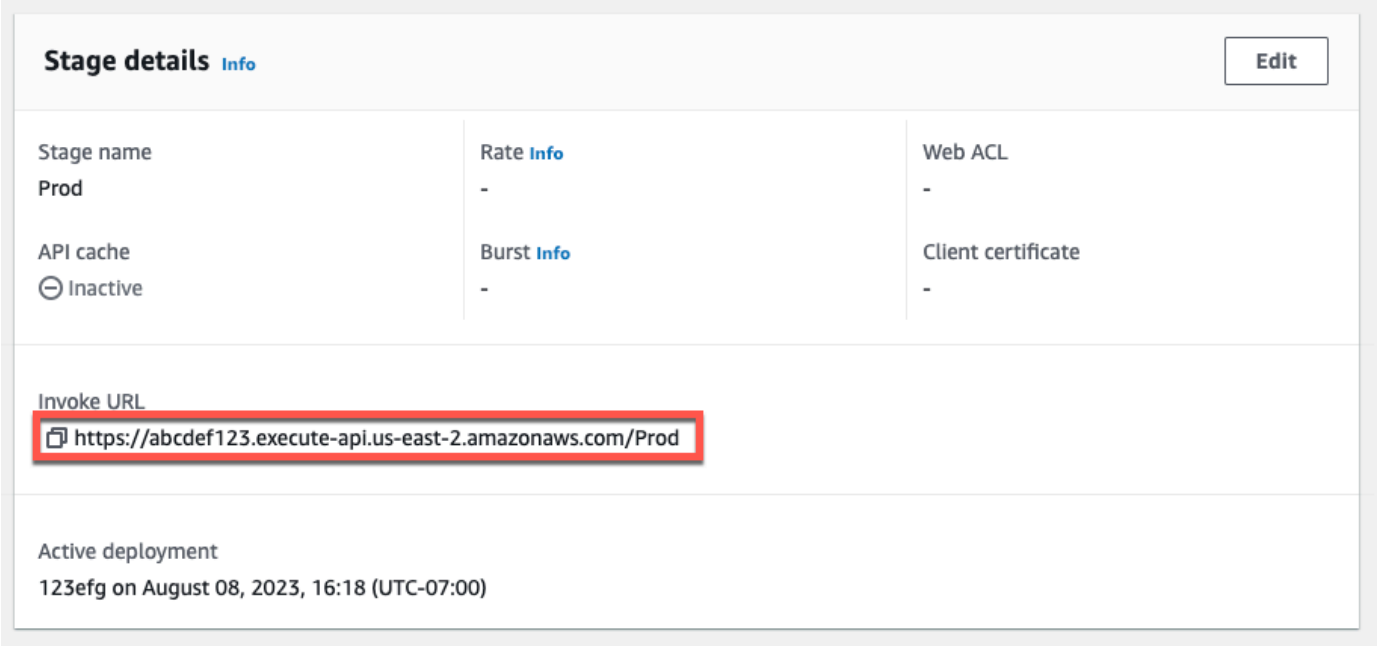
Entri Log output menunjukkan perubahan status dari permintaan metode ke permintaan integrasi, dan dari respons integrasi ke respons metode. Ini dapat berguna untuk memecahkan masalah kesalahan pemetaan yang menyebabkan permintaan gagal. Dalam contoh ini, tidak ada pemetaan yang diterapkan: payload permintaan metode diteruskan melalui permintaan integrasi ke backend dan, demikian pula, respons backend diteruskan melalui respons integrasi ke respons metode.

Untuk menguji API menggunakan klien selain test-invoke-request fitur API Gateway, Anda harus terlebih dahulu menerapkan API ke tahap.

9. Untuk menerapkan API sampel, pilih Deploy API.



10. Untuk Stage, pilih New stage, dan kemudian enter**`test`**.
11. (Opsional) Untuk Deskripsi, masukkan deskripsi.
12. Pilih Deploy.
13. Di panel Tahapan yang dihasilkan, di bawah Detail tahap, URL Invoke menampilkan URL untuk memanggil permintaan metode API. `GET /`



14. Pilih ikon salin untuk menyalin URL pemanggilan API Anda, lalu masukkan URL pemanggilan API Anda di browser web. Respons yang berhasil mengembalikan hasil, yang dihasilkan dari template pemetaan dalam respons integrasi.

15. Di panel navigasi Tahapan, perluas tahap pengujian, pilih GET on`/pets/{petId}`, lalu salin nilai URL Invoke dari. `https://api-id.execute-api.region.amazonaws.com/test/pets/{petId}` `{petId}`singkatan dari variabel jalur.

    Rekatkan nilai URL Invoke (diperoleh pada langkah sebelumnya) ke bilah alamat browser, ganti `{petId}` dengan, misalnya`1`, dan tekan Enter untuk mengirimkan permintaan. Respons 200 OK harus kembali dengan muatan JSON berikut:

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

    Memanggil metode API seperti yang ditunjukkan dimungkinkan karena jenis Otorisasi disetel ke. `NONE` Jika `AWS_IAM` otorisasi digunakan, Anda akan menandatangani permintaan menggunakan protokol Signature Version 4 (SigV4). Untuk contoh permintaan seperti itu, lihatthe section called "Tutorial: Membangun API dengan integrasi non-proxy HTTP".

Membangun API REST API Gateway dengan integrasi HTTP

Untuk membangun API dengan integrasi HTTP, Anda dapat menggunakan integrasi proxy HTTP atau integrasi kustom HTTP. Sebaiknya gunakan integrasi proxy HTTP, bila memungkinkan, untuk pengaturan API yang efisien sambil menyediakan fitur serbaguna dan canggih. Integrasi kustom HTTP dapat menarik jika perlu untuk mengubah data permintaan klien untuk backend atau mengubah data respon backend untuk klien.

Topik

- Tutorial: Membangun REST API dengan integrasi proxy HTTP
- Tutorial: Membangun REST API dengan integrasi non-proxy HTTP

Tutorial: Membangun REST API dengan integrasi proxy HTTP

Integrasi proxy HTTP adalah mekanisme sederhana, kuat, dan serbaguna untuk membangun API yang memungkinkan aplikasi web mengakses beberapa sumber daya atau fitur dari titik akhir HTTP terintegrasi, misalnya seluruh situs web, dengan pengaturan yang efisien dari satu metode API. Dalam integrasi proxy HTTP, API Gateway meneruskan permintaan metode yang dikirimkan klien

ke backend. Data permintaan yang diteruskan mencakup header permintaan, parameter string kueri, variabel jalur URL, dan payload. Titik akhir HTTP backend atau server web mem-parsing data permintaan yang masuk untuk menentukan respons yang dikembalikan. Integrasi proxy HTTP membuat klien dan backend berinteraksi secara langsung tanpa intervensi dari API Gateway setelah metode API disiapkan, kecuali untuk masalah yang diketahui seperti karakter yang tidak didukung, yang tercantum di dalamnya. the section called “Catatan penting”

Dengan sumber daya proxy yang mencakup semua`{proxy+}`, dan `ANY` kata kerja catch-all untuk metode HTTP, Anda dapat menggunakan integrasi proxy HTTP untuk membuat API dari satu metode API. Metode ini mengekspos seluruh rangkaian sumber daya HTTP yang dapat diakses publik dan operasi situs web. Ketika server web backend membuka lebih banyak sumber daya untuk akses publik, klien dapat menggunakan sumber daya baru ini dengan penyiapan API yang sama. Untuk mengaktifkan ini, pengembang situs web harus berkomunikasi dengan jelas kepada pengembang klien apa sumber daya baru dan operasi apa yang berlaku untuk masing-masing.

Sebagai pengantar singkat, tutorial berikut menunjukkan integrasi proxy HTTP. Dalam tutorial, kita membuat API menggunakan konsol API Gateway untuk berintegrasi dengan PetStore situs web melalui sumber daya proxy generik`{proxy+}`, dan membuat placeholder metode HTTP dari. `ANY`

Topik

- Membuat API dengan integrasi proxy HTTP menggunakan konsol API Gateway
- Uji API dengan integrasi proxy HTTP

Membuat API dengan integrasi proxy HTTP menggunakan konsol API Gateway

Prosedur berikut memandu Anda melalui langkah-langkah untuk membuat dan menguji API dengan sumber daya proxy untuk backend HTTP menggunakan konsol API Gateway. Backend HTTP adalah `PetStore` situs web (`http://petstore-demo-endpoint.execute-api.com/petstore/pets`) dariTutorial: Membangun REST API dengan integrasi non-proxy HTTP, di mana tangkapan layar digunakan sebagai alat bantu visual untuk mengilustrasikan elemen UI API Gateway. Jika Anda baru menggunakan konsol API Gateway untuk membuat API, Anda mungkin ingin mengikuti bagian itu terlebih dahulu.

Untuk membuat API

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.

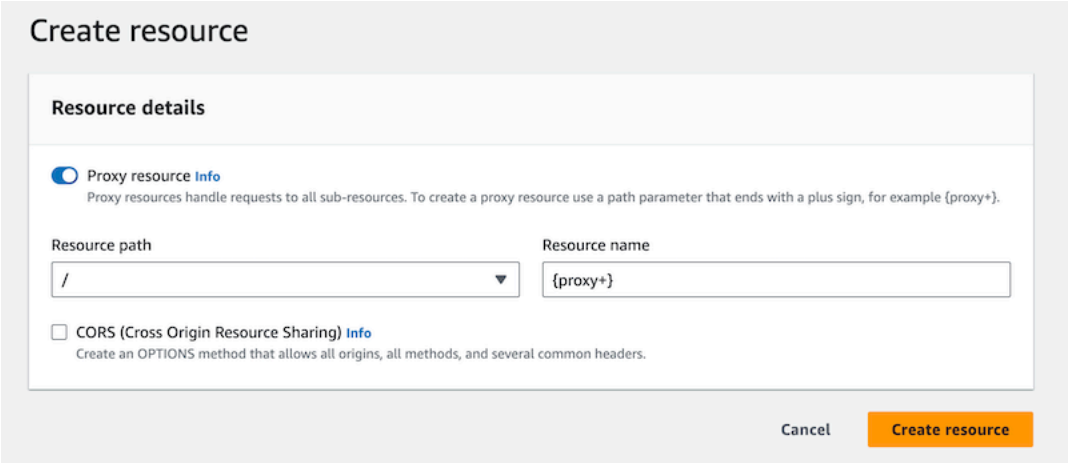
2. Jika ini adalah pertama kalinya Anda menggunakan API Gateway, Anda akan melihat halaman yang memperkenalkan Anda ke fitur layanan. Di bawah REST API, pilih Build. Saat munculan Create Example API muncul, pilih OK.

   Jika ini bukan pertama kalinya Anda menggunakan API Gateway, pilih Buat API. Di bawah REST API, pilih Build.
3. Untuk nama API, masukkan**HTTPProxyAPI**.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Tetap tetapkan jenis endpoint API ke Regional.
6. Pilih Buat API.

Pada langkah ini, Anda membuat jalur sumber daya proxy dari`{proxy+}`. Ini adalah placeholder dari salah satu titik akhir backend di bawah. `http://petstore-demo-endpoint.execute-api.com/` Misalnya, bisa jadi`petstore`,`petstore/pets`, dan`petstore/pets/{petId}`. API Gateway membuat `ANY` metode saat Anda membuat `{proxy+}` sumber daya dan berfungsi sebagai placeholder untuk salah satu kata kerja HTTP yang didukung pada waktu berjalan.

Untuk membuat sumber daya/{proxy+}

1. Pilih API Anda.
2. Di panel navigasi utama, pilih Resources.
3. Pilih Buat sumber daya.
4. Aktifkan sumber daya proxy.
5. Pertahankan jalur Sumber Daya sebagai`/`.
6. Untuk Nama sumber daya, masukkan **`{proxy+}`**.
7. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).
8. Pilih Buat sumber daya.



Pada langkah ini, Anda mengintegrasikan ANY metode dengan titik akhir HTTP backend, menggunakan integrasi proxy. Dalam integrasi proxy, API Gateway meneruskan permintaan metode yang dikirimkan klien ke backend tanpa intervensi dari API Gateway.

Untuk membuat **`ANY`** metode

1. Pilih sumber daya/{proxy+}.
2. Pilih metode APAPUN.
3. Di bawah simbol peringatan, pilih Edit integrasi. Anda tidak dapat menerapkan API yang memiliki metode tanpa integrasi.
4. Untuk jenis Integrasi, pilih HTTP.
5. Aktifkan integrasi proxy HTTP.
6. Untuk metode HTTP, pilih APAPUN.
7. Untuk URL Endpoint, masukkan**`http://petstore-demo-endpoint.execute-api.com/{proxy}`**.
8. Pilih Simpan.

Uji API dengan integrasi proxy HTTP

Apakah permintaan klien tertentu berhasil tergantung pada hal berikut:

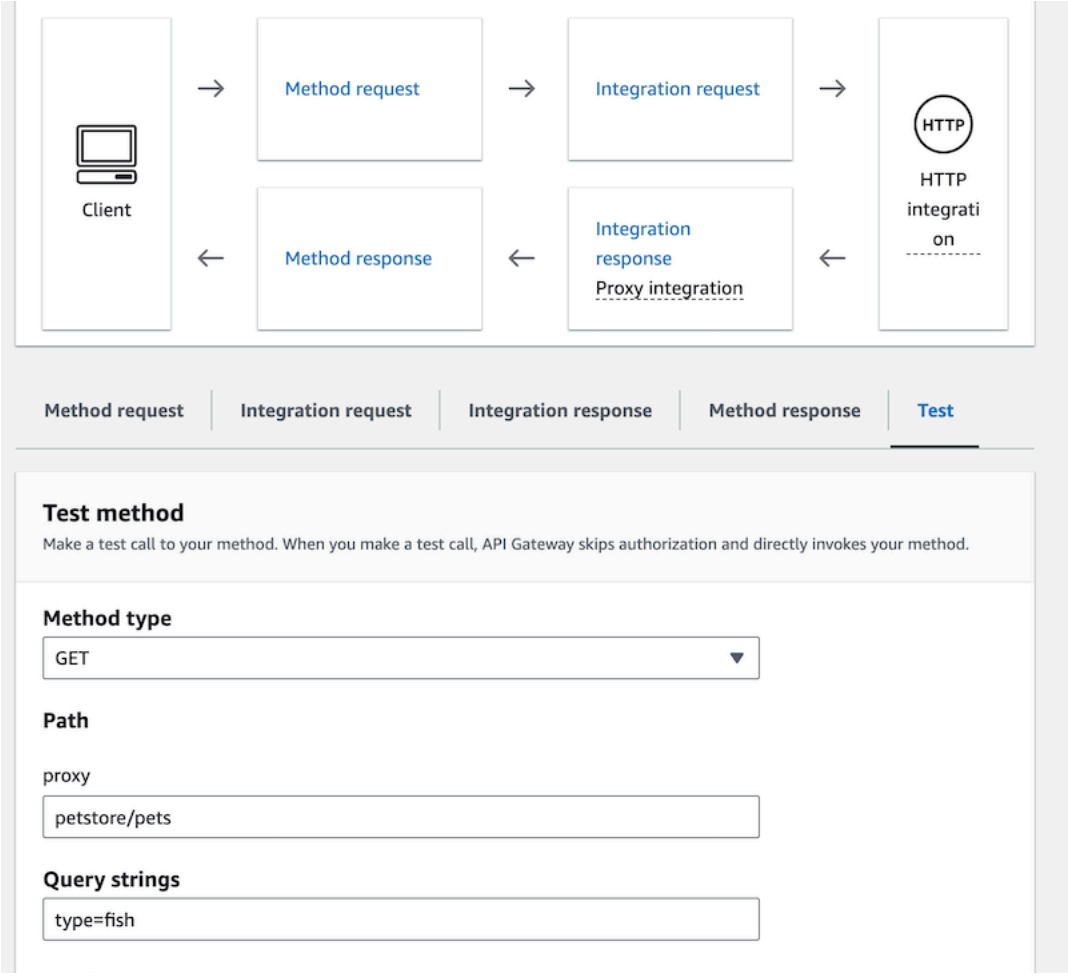
- Jika backend telah membuat titik akhir backend yang sesuai tersedia dan, jika demikian, telah memberikan izin akses yang diperlukan.
- Jika klien memberikan masukan yang benar.

Misalnya, PetStore API yang digunakan di sini tidak mengekspos `/petstore` sumber daya. Dengan demikian, Anda mendapatkan `404 Resource Not Found` respons yang berisi pesan kesalahan`Cannot GET /petstore`.

Selain itu, klien harus dapat menangani format output backend untuk mengurai hasil dengan benar. API Gateway tidak memediasi untuk memfasilitasi interaksi antara klien dan backend.

Untuk menguji API yang terintegrasi dengan PetStore situs web menggunakan integrasi proxy HTTP melalui sumber daya proxy

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Untuk jenis Metode, pilih`GET`.
3. Untuk Path, di bawah proxy, masukkan**`petstore/pets`**.
4. Untuk string Query, masukkan**`type=fish`**.
5. Pilih Uji.



Karena situs web backend mendukung `GET /petstore/pets?type=fish` permintaan, ia mengembalikan respons yang berhasil serupa dengan yang berikut:

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

Jika Anda mencoba meneleponGET /petstore, Anda mendapatkan 404 respons dengan pesan kesalahanCannot GET /petstore. Ini karena backend tidak mendukung operasi yang ditentukan. Jika Anda meneleponGET /petstore/pets/1, Anda mendapatkan 200 OK respons dengan muatan berikut, karena permintaan didukung oleh PetStore situs web.

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

Anda juga dapat menggunakan browser untuk menguji API Anda. Terapkan API Anda dan kaitkan ke panggung untuk membuat URL Invoke API Anda.

Untuk men-deploy API Anda

1. Pilih Deploy API.
2. Untuk Stage, pilih New stage.

3. Untuk nama Panggung, masukkan**`test`**.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Pilih Deploy.

Sekarang klien dapat menghubungi API Anda.

Untuk menjalankan API Anda

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API Anda.
3. Di panel navigasi utama, pilih Stage.
4. Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda.

   Masukkan URL pemanggilan API Anda di browser web.

   URL lengkap akan terlihat seperti`https://abcdef123.execute-api.us-east-2.amazonaws.com/test/petstore/pets?type=fish`.

   Browser Anda mengirimkan `GET` permintaan ke API.
5. Hasilnya harus sama dengan yang dikembalikan saat Anda menggunakan Test di konsol API Gateway.

Tutorial: Membangun REST API dengan integrasi non-proxy HTTP

Dalam tutorial ini, Anda membuat API dari awal menggunakan konsol Amazon API Gateway. Anda dapat menganggap konsol sebagai studio desain API dan menggunakannya untuk cakupan fitur API, bereksperimen dengan perilakunya, membangun API, dan menerapkan API Anda secara bertahap.

Topik

- Buat API dengan integrasi kustom HTTP
- (Opsional) Parameter permintaan peta

Buat API dengan integrasi kustom HTTP

Bagian ini memandu Anda melalui langkah-langkah untuk membuat sumber daya, mengekspos metode pada sumber daya, mengonfigurasi metode untuk mencapai perilaku API yang diinginkan, dan untuk menguji dan menerapkan API.

Pada langkah ini, Anda membuat API kosong. Dalam langkah-langkah berikut, Anda membuat sumber daya dan metode untuk menghubungkan API Anda ke `http://petstore-demo-endpoint.execute-api.com/petstore/pets` titik akhir, menggunakan integrasi HTTP non-proxy.

Untuk membuat API

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Jika ini adalah pertama kalinya Anda menggunakan API Gateway, Anda akan melihat halaman yang memperkenalkan Anda ke fitur layanan. Di bawah REST API, pilih Build. Saat munculan Create Example API muncul, pilih OK.

   Jika ini bukan pertama kalinya Anda menggunakan API Gateway, pilih Buat API. Di bawah REST API, pilih Build.
3. Untuk nama API, masukkan**`HTTPNonProxyAPI`**.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Tetap tetapkan jenis endpoint API ke Regional.
6. Pilih Buat API.

Pohon Resources menunjukkan sumber daya root (`/`) tanpa metode apa pun. Dalam latihan ini, kita akan membangun API dengan integrasi kustom HTTP dari PetStore situs web (http://petstore-demo-endpoint.execute-api.com/petstore/pets.) Untuk tujuan ilustrasi, kami akan membuat `/pets` sumber daya sebagai anak dari root dan mengekspos metode GET pada sumber daya ini bagi klien untuk mengambil daftar item Hewan Peliharaan yang tersedia dari situs web. PetStore

Untuk membuat sumber daya /pets

1. Pilih sumber daya/, lalu pilih Buat sumber daya.
2. Matikan sumber daya Proxy.
3. Pertahankan jalur Sumber Daya sebagai`/`.
4. Untuk Nama sumber daya, masukkan **`pets`**.
5. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).
6. Pilih Buat sumber daya.

Pada langkah ini, Anda membuat `GET` metode pada sumber daya /pets. `GET`Metode ini terintegrasi dengan situs `http://petstore-demo-endpoint.execute-api.com/petstore/pets` web. Opsi lain untuk metode API termasuk yang berikut:

- POST, terutama digunakan untuk membuat sumber daya anak.
- PUT, terutama digunakan untuk memperbarui sumber daya yang ada (dan, meskipun tidak disarankan, dapat digunakan untuk membuat sumber daya anak).
- DELETE, digunakan untuk menghapus sumber daya.
- PATCH, digunakan untuk memperbarui sumber daya.
- HEAD, terutama digunakan dalam skenario pengujian. Ini sama dengan GET tetapi tidak mengembalikan representasi sumber daya.
- OPTIONS, yang dapat digunakan oleh penelepon untuk mendapatkan informasi tentang opsi komunikasi yang tersedia untuk layanan target.

Untuk metode HTTP permintaan integrasi, Anda harus memilih salah satu yang didukung oleh backend. Untuk `HTTP` atau`Mock integration`, masuk akal bahwa permintaan metode dan permintaan integrasi menggunakan kata kerja HTTP yang sama. Untuk jenis integrasi lainnya, permintaan metode kemungkinan akan menggunakan kata kerja HTTP yang berbeda dari permintaan integrasi. Misalnya, untuk memanggil fungsi Lambda, permintaan integrasi harus digunakan `POST` untuk memanggil fungsi, sedangkan permintaan metode dapat menggunakan kata kerja HTTP apa pun tergantung pada logika fungsi Lambda.

Untuk membuat **`GET`** metode pada sumber daya /pets

1. Pilih sumber daya /pets.
2. Pilih metode Buat.
3. Untuk tipe Metode, pilih GET.
4. Untuk jenis Integrasi, pilih Integrasi HTTP.
5. Matikan integrasi proxy HTTP.
6. Untuk metode HTTP, pilih GET.
7. Untuk URL Endpoint, masukkan**`http://petstore-demo-endpoint.execute-api.com/petstore/pets`**.
8. Untuk penanganan Konten, pilih Passthrough.
9. Tetap aktifkan batas waktu default.

10. Pilih metode Buat.

PetStore Situs web ini memungkinkan Anda untuk mengambil daftar `Pet` item berdasarkan jenis hewan peliharaan (seperti “Dog” atau “Cat”) pada halaman tertentu. Ini menggunakan parameter string `type` dan `page` query untuk menerima input tersebut. Dengan demikian, kita harus menambahkan parameter string kueri ke permintaan metode dan memetakannya ke dalam string kueri yang sesuai dari permintaan integrasi.

Untuk menambahkan parameter string kueri ke **`GET`** metode

1. Pada tab Permintaan metode, di bawah Pengaturan permintaan metode, pilih Edit.
2. Pilih parameter string kueri URL, lalu lakukan hal berikut:
    a. Pilih Tambahkan string kueri.
    b. Untuk Nama, masukkan **`type`**
    c. Tetap Diperlukan dan Caching dimatikan.

    Ulangi langkah sebelumnya untuk membuat string kueri tambahan dengan nama**`page`**.
3. Pilih Simpan.

Klien sekarang dapat menyediakan jenis hewan peliharaan dan nomor halaman sebagai parameter string kueri saat mengirimkan permintaan. Parameter input ini harus dipetakan ke dalam parameter string kueri integrasi untuk meneruskan nilai input ke PetStore situs web kami di backend.

Untuk memetakan parameter input ke permintaan Integrasi

1. Pada tab Permintaan integrasi, di bawah Pengaturan permintaan integrasi, pilih Edit.
2. Pilih parameter string kueri URL, lalu lakukan hal berikut:
    a. Pilih Tambahkan parameter string kueri.
    b. Untuk Nama, masukkan **`type`**.
    c. Untuk Dipetakan dari, masukkan **`method.request.querystring.type`**
    d. Tetap caching dimatikan.
    e. Pilih Tambahkan parameter string kueri.
    f. Untuk Nama, masukkan **`page`**.

g. Untuk Dipetakan dari, masukkan **`method.request.querystring.page`**

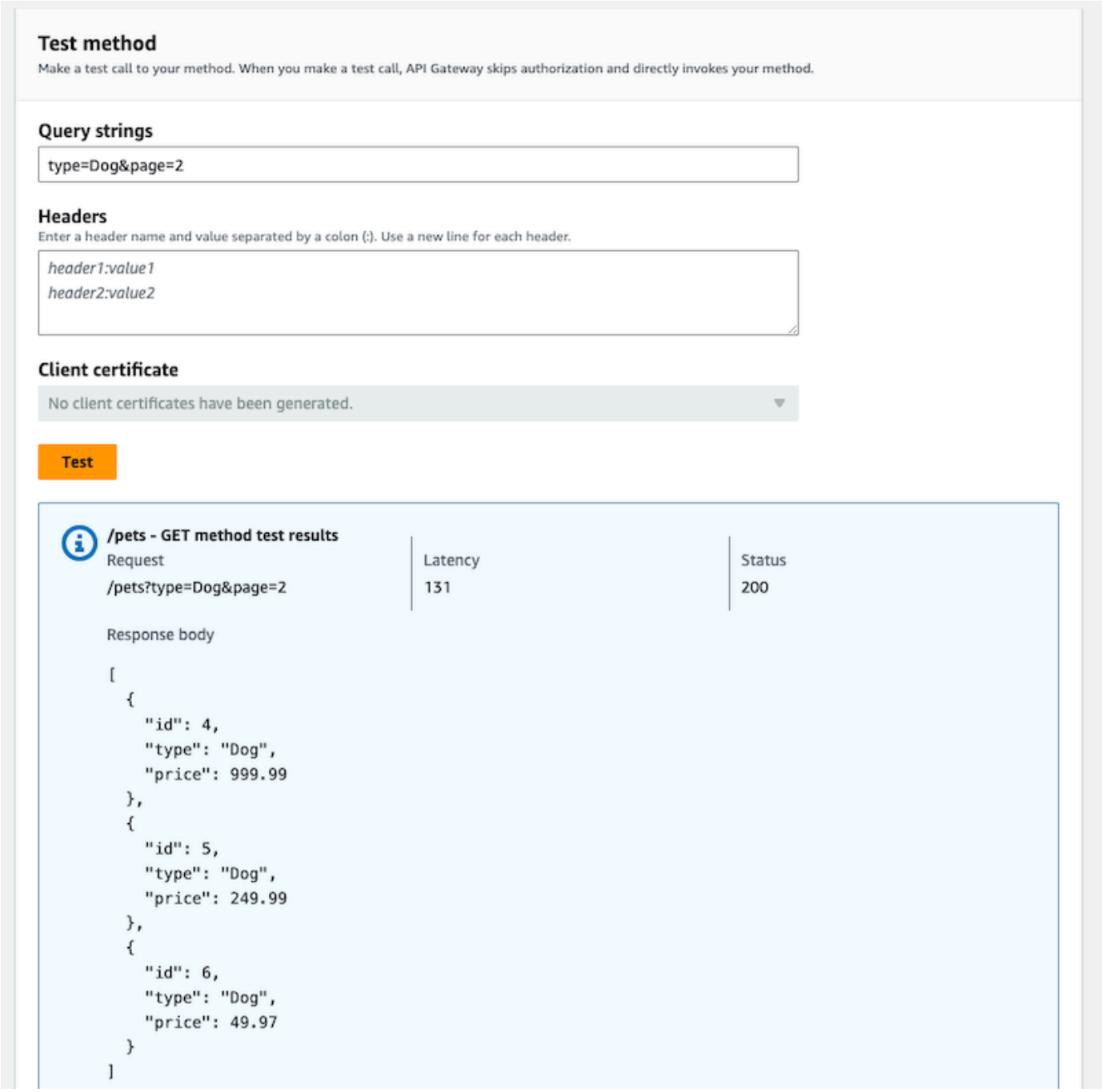
h. Tetap caching dimatikan.

3. Pilih Simpan.

Untuk menguji API

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Untuk string Query, masukkan**`type=Dog&page=2`**.
3. Pilih Uji.

   Hasilnya mirip dengan yang berikut:



Sekarang setelah pengujian berhasil, kami dapat menerapkan API untuk membuatnya tersedia untuk umum.

4. Pilih Deploy API.
5. Untuk Stage, pilih New stage.
6. Untuk nama Panggung, masukkan**`Prod`**.
7. (Opsional) Untuk Deskripsi, masukkan deskripsi.

8. Pilih Deploy.
9. (Opsional) Di bawah detail Tahap, untuk Invoke URL, Anda dapat memilih ikon salin untuk menyalin URL pemanggilan API Anda. Anda dapat menggunakan ini dengan alat seperti Postman dan cURL untuk menguji API Anda.

Jika Anda menggunakan SDK untuk membuat klien, Anda dapat memanggil metode yang diekspos oleh SDK untuk menandatangani permintaan. Untuk detail implementasi, lihat AWS SDK yang Anda pilih.

> **Note**
>
> Saat perubahan dilakukan pada API Anda, Anda harus menerapkan ulang API agar fitur baru atau yang diperbarui tersedia sebelum menjalankan URL permintaan lagi.

(Opsional) Parameter permintaan peta

Parameter permintaan peta untuk API Gateway API

Tutorial ini menunjukkan cara membuat parameter jalur `{petId}` pada URL permintaan metode API untuk menentukan ID item, memetakannya ke parameter `{id}` jalur di URL permintaan integrasi, dan mengirim permintaan ke titik akhir HTTP.

> **Note**
>
> Jika Anda memasukkan huruf yang salah, seperti huruf kecil, bukan huruf besar, ini akan menyebabkan kesalahan nanti dalam penelusuran.

Langkah 1: Buat sumber daya

Pada langkah ini, Anda membuat sumber daya dengan parameter jalur {PeTiD}.

Untuk membuat sumber daya {PeTiD}

1. Pilih sumber daya /pets, lalu pilih Create resource.
2. Matikan sumber daya Proxy.
3. Untuk jalur Sumber Daya, pilih /pets/.

4. Untuk Nama sumber daya, masukkan **`{petId}`**.

   Gunakan kurawal kurawal (`{ }`) di sekitar `petId` sehingga /pets/ {PeTiD} ditampilkan.
5. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).
6. Pilih Buat sumber daya.

Langkah 2: Buat dan uji metode

Pada langkah ini, Anda membuat `GET` metode dengan parameter `{petId}` jalur.

Untuk mengatur metode GET

1. Pilih sumber daya/{PeTiD}, lalu pilih Create method.
2. Untuk tipe Metode, pilih GET.
3. Untuk jenis Integrasi, pilih Integrasi HTTP.
4. Matikan integrasi proxy HTTP.
5. Untuk metode HTTP, pilih GET.
6. Untuk URL Endpoint, masukkan **`http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}`**
7. Untuk penanganan Konten, pilih Passthrough.
8. Tetap aktifkan batas waktu default.
9. Pilih metode Buat.

Sekarang Anda memetakan parameter `{petId}` path ke parameter `{id}` path di endpoint HTTP.

Untuk memetakan parameter **`{petId}`** jalur

1. Pada tab Permintaan integrasi, di bawah Pengaturan permintaan integrasi, pilih Edit.
2. Pilih parameter jalur URL.
3. API Gateway membuat parameter jalur untuk permintaan integrasi bernama PeTiD. Ini tidak akan berfungsi untuk backend Anda. Titik akhir HTTP digunakan `{id}` sebagai parameter jalur. Ubah nama **`id`**PeTiD menjadi.

   Ini memetakan parameter jalur permintaan metode `petId` ke parameter jalur permintaan integrasi`id`.

4. Pilih Simpan.

Sekarang Anda menguji metodenya.

Untuk menguji metode

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Di bawah Path for PeTiD, masukkan. **4**
3. Pilih Uji.

   Jika berhasil, badan Response menampilkan yang berikut:

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

Langkah 3: Menyebarkan API

Pada langkah ini, Anda menerapkan API sehingga Anda dapat mulai memanggilnya di luar konsol API Gateway.

Untuk menerapkan API

1. Pilih Deploy API.
2. Untuk Stage, pilih Prod.
3. (Opsional) Untuk Deskripsi, masukkan deskripsi.
4. Pilih Deploy.

Langkah 4: Uji API

Pada langkah ini, Anda pergi ke luar konsol API Gateway dan menggunakan API Anda untuk mengakses titik akhir HTTP.

1. Di panel navigasi utama, pilih Stage.
2. Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda.

Seharusnya terlihat seperti ini:

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

3. Masukkan URL ini di kotak alamat tab browser baru dan tambahkan `/pets/4` ke URL sebelum Anda mengirimkan permintaan Anda.
4. Browser akan mengembalikan yang berikut:

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

Langkah selanjutnya

Anda dapat menyesuaikan API lebih lanjut dengan mengaktifkan validasi permintaan, mengubah data, atau membuat respons gateway khusus.

Untuk menjelajahi lebih banyak cara untuk menyesuaikan API Anda, lihat tutorial berikut:

- Untuk informasi selengkapnya tentang validasi permintaan, lihatSiapkan validasi permintaan dasar di API Gateway.
- Untuk informasi tentang cara mengubah payload permintaan dan respons, lihatMengatur transformasi data di API Gateway.
- Untuk informasi tentang cara membuat respons gateway khusus, lihat,Menyiapkan respons gateway untuk REST API menggunakan konsol API Gateway.

Tutorial: Membangun REST API dengan integrasi pribadi API Gateway

Anda dapat membuat API Gateway API dengan integrasi pribadi untuk memberikan pelanggan Anda akses ke sumber daya HTTP/HTTPS dalam Amazon Virtual Private Cloud (Amazon VPC). Sumber daya VPC tersebut adalah titik akhir HTTP/HTTPS pada instans EC2 di belakang Network Load Balancer di VPC. Network Load Balancer merangkum sumber daya VPC dan merutekan permintaan yang masuk ke sumber daya yang ditargetkan.

Saat klien memanggil API, API Gateway terhubung ke Network Load Balancer melalui tautan VPC yang telah dikonfigurasi sebelumnya. Tautan VPC dienkapsulasi oleh sumber daya API Gateway dari. VpcLink Ini bertanggung jawab untuk meneruskan permintaan metode API ke sumber daya VPC dan mengembalikan respons backend ke pemanggil. Untuk pengembang API, a `VpcLink` secara fungsional setara dengan titik akhir integrasi.

Untuk membuat API dengan integrasi pribadi, Anda harus membuat yang baru`VpcLink`, atau memilih yang sudah ada, yang terhubung ke Network Load Balancer yang menargetkan sumber daya VPC yang diinginkan. Anda harus memiliki izin yang sesuai untuk membuat dan mengelola file. `VpcLink` Anda kemudian menyiapkan metode API dan mengintegrasikannya `VpcLink` dengan mengatur salah satu `HTTP` atau `HTTP_PROXY` sebagai tipe integrasi, menyetel `VPC_LINK` sebagai jenis koneksi integrasi, dan menyetel `VpcLink` pengenal pada integrasi `connectionId`.

> **Note**
>
> Network Load Balancer dan API harus dimiliki oleh akun yang samaAWS.

Untuk segera mulai membuat API untuk mengakses sumber daya VPC, kami berjalan melalui langkah-langkah penting untuk membangun API dengan integrasi pribadi, menggunakan konsol API Gateway. Sebelum membuat API, lakukan hal berikut:

1. Buat sumber daya VPC, buat atau pilih Network Load Balancer di bawah akun Anda di wilayah yang sama, dan tambahkan instans EC2 yang menghosting sumber daya sebagai target Network Load Balancer. Untuk informasi selengkapnya, lihat Menyiapkan Network Load Balancer untuk integrasi pribadi API Gateway.
2. Berikan izin untuk membuat tautan VPC untuk integrasi pribadi. Untuk informasi selengkapnya, lihat Berikan izin untuk membuat tautan VPC.

Setelah membuat sumber daya VPC dan Network Load Balancer dengan resource VPC yang dikonfigurasi dalam grup targetnya, ikuti petunjuk di bawah ini untuk membuat API dan mengintegrasikannya dengan resource VPC melalui a in a private integration. `VpcLink`

Untuk membuat API dengan integrasi pribadi

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.

2. Jika ini adalah pertama kalinya Anda menggunakan API Gateway, Anda akan melihat halaman yang memperkenalkan Anda ke fitur layanan. Di bawah REST API, pilih Build. Saat munculan Create Example API muncul, pilih OK.

   Jika ini bukan pertama kalinya Anda menggunakan API Gateway, pilih Buat API. Di bawah REST API, pilih Build.
3. Buat REST API Regional atau yang dioptimalkan untuk tepi.
4. Pilih API Anda.
5. Pilih metode Create, dan kemudian lakukan hal berikut:

   a. Untuk jenis Metode, pilih`GET`.
   b. Untuk jenis Integrasi, pilih tautan VPC.
   c. Aktifkan integrasi proxy VPC.
   d. Untuk metode HTTP, pilih`GET`.
   e. Untuk tautan VPC, pilih [Gunakan variabel panggung] dan masukkan kotak teks **`${stageVariables.vpcLinkId}`** di bawah ini.

      Anda menentukan variabel `vpcLinkId` stage setelah menerapkan API ke tahap dan menetapkan nilainya ke ID. `VpcLink`
   f. Untuk URL Endpoint, masukkan URL, misalnya,`http://myApi.example.com`.

      Di sini, nama host (misalnya,`myApi.example.com`) digunakan untuk mengatur `Host` header permintaan integrasi.
   g. Pilih metode Buat.

      Dengan integrasi proxy, API siap digunakan. Jika tidak, Anda perlu melanjutkan untuk mengatur respons metode dan respons integrasi yang sesuai.
6. Pilih Deploy API, lalu lakukan hal berikut:

   a. Untuk Stage, pilih New stage.
   b. Untuk nama Panggung, masukkan nama panggung.
   c. (Opsional) Untuk Deskripsi, masukkan deskripsi.
   d. Pilih Deploy.
7. Di bawah bagian Detail tahap, perhatikan URL Panggilan yang dihasilkan. Anda membutuhkannya untuk menjalankan API. Sebelum melakukan itu, Anda harus mengatur variabel `vpcLinkId` tahap.

8. Di panel Tahapan, pilih tab Variabel tahap, lalu lakukan hal berikut:

   a. Pilih Kelola variabel, lalu pilih Tambahkan variabel tahap.

   b. Untuk Nama, masukkan **`vpcLinkId`**.

   c. Untuk Nilai, masukkan IDVPC_LINK, misalnya,*`gix6s7`*.

   d. Pilih Simpan.

      Menggunakan variabel stage, Anda dapat dengan mudah beralih ke link VPC yang berbeda untuk API dengan mengubah nilai variabel stage.

Tutorial: Membangun API API Gateway REST dengan AWS integrasi

Baik Bangun API REST API Gateway dengan integrasi Lambda topik Tutorial: Bangun API REST Hello World dengan integrasi proxy Lambda maupun topik menjelaskan cara membuat API Gateway API untuk mengekspos fungsi Lambda terintegrasi. Selain itu, Anda dapat membuat API Gateway API untuk mengekspos AWS layanan lain, seperti Amazon SNS, Amazon S3, Amazon Kinesis, dan bahkan. AWS Lambda Ini dimungkinkan oleh `AWS` integrasi. Integrasi Lambda atau integrasi proxy Lambda adalah kasus khusus, di mana pemanggilan fungsi Lambda diekspos melalui API Gateway API.

Semua AWS layanan mendukung API khusus untuk mengekspos fitur-fiturnya. Namun, protokol aplikasi atau antarmuka pemrograman cenderung berbeda dari layanan ke layanan. API Gateway API dengan `AWS` integrasi memiliki keuntungan menyediakan protokol aplikasi yang konsisten bagi klien Anda untuk mengakses berbagai AWS layanan.

Dalam panduan ini, kami membuat API untuk mengekspos Amazon SNS. Untuk contoh lebih lanjut tentang mengintegrasikan API dengan AWS layanan lain, lihatTutorial dan lokakarya Amazon API Gateway.

Berbeda dengan integrasi proxy Lambda, tidak ada integrasi proxy yang sesuai untuk layanan lainAWS. Oleh karena itu, metode API terintegrasi dengan satu AWS tindakan. Untuk fleksibilitas lebih, mirip dengan integrasi proxy, Anda dapat mengatur integrasi proxy Lambda. Fungsi Lambda kemudian mem-parsing dan memproses permintaan untuk tindakan lain. AWS

API Gateway tidak mencoba lagi saat titik akhir habis. Pemanggil API harus menerapkan logika coba lagi untuk menangani batas waktu titik akhir.

Panduan ini dibangun di atas instruksi dan konsep di. Bangun API REST API Gateway dengan integrasi Lambda Jika Anda belum menyelesaikan panduan itu, kami sarankan Anda melakukannya terlebih dahulu.

Topik

Prasyarat

Sebelum memulai panduan ini, lakukan hal berikut:

1. Selesaikan langkah-langkah dalam Prasyarat untuk memulai dengan API Gateway.
2. Buat API baru bernama`MyDemoAPI`. Untuk informasi selengkapnya, lihat Tutorial: Membangun REST API dengan integrasi non-proxy HTTP.
3. Terapkan API setidaknya sekali ke tahap bernama`test`. Untuk informasi selengkapnya, lihat Menerapkan API diBangun API REST API Gateway dengan integrasi Lambda.
4. Selesaikan langkah-langkah lainnyaBangun API REST API Gateway dengan integrasi Lambda.
5. Buat setidaknya satu topik di Amazon Simple Notification Service (Amazon SNS). Anda akan menggunakan API yang diterapkan untuk mendapatkan daftar topik di Amazon SNS yang terkait dengan AWS akun Anda. Untuk mempelajari cara membuat topik di Amazon SNS, lihat Membuat Topik. (Anda tidak perlu menyalin topik ARN yang disebutkan di langkah 5.)

Langkah 1: Buat peran eksekusi proxy AWS layanan

Untuk mengizinkan API menjalankan tindakan Amazon SNS, Anda harus memiliki kebijakan IAM yang sesuai yang dilampirkan ke peran IAM.

Untuk membuat peran eksekusi proxy AWS layanan

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Peran.
3. Pilih Create role (Buat peran).
4. Pilih AWSlayanan di bawah Pilih jenis entitas tepercaya, lalu pilih API Gateway dan pilih Izinkan API Gateway untuk mendorong CloudWatch log ke Log.
5. Pilih Berikutnya, lalu pilih Berikutnya.
6. Untuk nama Peran**`APIGatewaySNSProxyPolicy`**, masukkan, lalu pilih Buat peran.
7. Dalam daftar Peran, pilih peran yang baru saja Anda buat. Anda mungkin perlu menggulir atau menggunakan bilah pencarian untuk menemukan peran.
8. Untuk peran yang dipilih, pilih tab Tambahkan izin.
9. Pilih Lampirkan kebijakan dari daftar dropdown.
10. Di bilah pencarian, masukkan **`AmazonSNSReadOnlyAccess`** dan pilih Tambahkan izin.

> **Note**
>
> Tutorial ini menggunakan kebijakan terkelola untuk kesederhanaan. Sebagai praktik terbaik, Anda harus membuat kebijakan IAM Anda sendiri untuk memberikan izin minimum yang diperlukan.

11. Perhatikan ARN Peran yang baru dibuat, Anda akan menggunakannya nanti.

Langkah 2: Buat sumber daya

Pada langkah ini, Anda membuat sumber daya yang memungkinkan proxy AWS layanan berinteraksi dengan AWS layanan.

Untuk membuat sumber daya

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API Anda.
3. Pilih sumber daya root,/, yang diwakili oleh satu garis miring (/), lalu pilih Buat sumber daya.
4. Matikan sumber daya Proxy.

5. Pertahankan jalur Sumber Daya sebagai`/`.
6. Untuk Nama sumber daya, masukkan **`mydemoawsproxy`**.
7. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).
8. Pilih Buat sumber daya.

Langkah 3: Buat metode GET

Pada langkah ini, Anda membuat metode GET yang memungkinkan proxy AWS layanan berinteraksi dengan AWS layanan.

Untuk membuat **`GET`** metode

1. Pilih sumber daya /mydemoawsproxy, lalu pilih Create method.
2. Untuk jenis metode, pilih GET.
3. Untuk jenis Integrasi, pilih Layanan AWS.
4. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat topik Amazon SNS Anda.
5. Untuk Layanan AWS, pilih Amazon SNS.
6. Biarkan AWSsubdomain kosong.
7. Untuk metode HTTP, pilih GET.
8. Untuk jenis tindakan, pilih Gunakan nama tindakan.
9. Untuk nama Tindakan, masukkan**`ListTopics`**.
10. Untuk peran Eksekusi, masukkan peran ARN untuk. **`APIGatewaySNSProxyPolicy`**
11. Pilih metode Buat.

Langkah 4: Tentukan pengaturan metode dan uji metode

Anda sekarang dapat menguji `GET` metode Anda untuk memverifikasi bahwa metode tersebut telah diatur dengan benar untuk mencantumkan topik Amazon SNS Anda.

Untuk menguji **`GET`** metode

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Pilih Uji.

   Hasilnya menampilkan respons yang mirip dengan berikut ini:

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

Langkah 5: Menyebarkan API

Pada langkah ini, Anda menerapkan API sehingga Anda dapat memanggilnya dari luar konsol API Gateway.

Untuk menerapkan API

1. Pilih Deploy API.
2. Untuk Stage, pilih New stage.
3. Untuk nama Panggung, masukkan**`test`**.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Pilih Deploy.

Langkah 6: Uji API

Pada langkah ini, Anda pergi ke luar konsol API Gateway dan menggunakan proxy AWS layanan Anda untuk berinteraksi dengan layanan Amazon SNS.

1. Di panel navigasi utama, pilih Stage.
2. Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda.

   Seharusnya terlihat seperti ini:

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

3. Masukkan URL ke dalam kotak alamat tab browser baru.
4. Tambahkan `/mydemoawsproxy` sehingga URL terlihat seperti ini:

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

   Jelajahi URL. Informasi yang mirip dengan berikut ini harus ditampilkan:

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

Langkah 7: Membersihkan

Anda dapat menghapus sumber daya IAM yang diperlukan proxy AWS layanan untuk bekerja.

> **Warning**
>
> Jika Anda menghapus sumber daya IAM yang diandalkan oleh proxy AWS layanan, proxy AWS layanan tersebut dan API apa pun yang mengandalkannya tidak akan berfungsi lagi. Menghapus sumber daya IAM tidak dapat dibatalkan. Jika Anda ingin menggunakan sumber daya IAM lagi, Anda harus membuatnya kembali.

Untuk menghapus sumber daya IAM terkait

1. Buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Di area Detail, pilih Peran.
3. Pilih ApiGateway AWSProxyExecRole, lalu pilih Tindakan Peran, Hapus Peran. Saat diminta, pilih Ya, Hapus.
4. Di area Detail, pilih Kebijakan.
5. Pilih ApiGateway AWSProxyExecPolicy, lalu pilih Tindakan Kebijakan, Hapus. Saat diminta, pilih Hapus.

Anda telah mencapai akhir dari panduan ini. Untuk diskusi lebih rinci tentang membuat API sebagai proxy AWS layanan, lihatTutorial: Membuat REST API sebagai proxy Amazon S3 di API Gateway,Tutorial: Buat `Calc` REST API dengan dua integrasi AWS layanan dan satu integrasi non-proxy Lambda, atauTutorial: Membuat REST API sebagai proxy Amazon Kinesis di API Gateway.

Tutorial: Buat **`Calc`** REST API dengan dua integrasi AWS layanan dan satu integrasi non-proxy Lambda

Tutorial integrasi non-proxy Memulai menggunakan `Lambda Function` integrasi secara eksklusif. `Lambda Function`integrasi adalah kasus khusus dari jenis `AWS Service` integrasi yang melakukan banyak pengaturan integrasi untuk Anda, seperti secara otomatis menambahkan izin berbasis sumber daya yang diperlukan untuk menjalankan fungsi Lambda. Di sini, dua dari tiga integrasi menggunakan `AWS Service` integrasi. Dalam jenis integrasi ini, Anda memiliki lebih banyak kontrol, tetapi Anda harus melakukan tugas secara manual seperti membuat dan menentukan peran IAM yang berisi izin yang sesuai.

Dalam tutorial ini, Anda akan membuat fungsi `Calc` Lambda yang mengimplementasikan operasi aritmatika dasar, menerima dan mengembalikan input dan output berformat JSON. Kemudian Anda akan membuat REST API dan mengintegrasikannya dengan fungsi Lambda dengan cara berikut:

1. Dengan mengekspos `GET` metode pada `/calc` sumber daya untuk menjalankan fungsi Lambda, memasok input sebagai parameter string kueri. (`AWS Service`integrasi)
2. Dengan mengekspos `POST` metode pada `/calc` sumber daya untuk menjalankan fungsi Lambda, memasok input dalam payload permintaan metode. (`AWS Service`integrasi)

3. Dengan mengekspos `/calc/{operand1}/{operand2}/{operator}` sumber daya `GET` bersarang untuk menjalankan fungsi Lambda, memasok input sebagai parameter jalur. (`Lambda Function`integrasi)

Selain mencoba tutorial ini, Anda mungkin ingin mempelajari file definisi OpenAPI untuk `Calc` API, yang dapat Anda impor ke API Gateway dengan mengikuti petunjuk di. the section called “OpenAPI”

Topik

- Buat peran IAM yang dapat diasumsikan
- Buat fungsi Calc Lambda
- Uji fungsi Calc Lambda
- Buat Calc API
- Integrasi 1: Buat GET metode dengan parameter kueri untuk memanggil fungsi Lambda
- Integrasi 2: Buat POST metode dengan payload JSON untuk memanggil fungsi Lambda
- Integrasi 3: Buat GET metode dengan parameter jalur untuk memanggil fungsi Lambda
- Definisi OpenAPI dari contoh API terintegrasi dengan fungsi Lambda

Buat peran IAM yang dapat diasumsikan

Agar API dapat menjalankan fungsi `Calc` Lambda, Anda harus memiliki peran IAM yang dapat diasumsikan API Gateway, yang merupakan peran IAM dengan hubungan tepercaya berikut:

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

Peran yang Anda buat harus memiliki izin Lambda InvokeFunction. Jika tidak, pemanggil API akan menerima `500 Internal Server Error` respons. Untuk memberikan izin ini kepada peran ini, Anda akan melampirkan kebijakan IAM berikut:

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

Inilah cara untuk mencapai semua ini:

Buat peran IAM yang dapat diasumsikan API Gateway

1. Masuk ke konsol IAM.
2. Pilih Peran.
3. Pilih Buat Peran.
4. Di Pilih jenis entitas tepercaya, pilih Layanan AWS .
5. Di bawah Pilih layanan yang akan menggunakan di peran ini, pilih Lambda.
6. Pilih Berikutnya: Izin.
7. Pilih Buat Kebijakan.

   Jendela konsol Create Policy baru akan terbuka. Di jendela itu, lakukan hal berikut:

   a. Di tab JSON, ganti kebijakan yang ada dengan yang berikut:

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

```
    ]
}
```

b. Pilih Tinjau kebijakan.

c. Di bawah Kebijakan Peninjauan, lakukan hal berikut:

i. Untuk Nama, ketik nama seperti **`lambda_execute`**.

ii. Pilih Buat Kebijakan.

8. Di jendela konsol Create Role asli, lakukan hal berikut:

a. Di bawah Lampirkan kebijakan izin, pilih **`lambda_execute`** kebijakan Anda dari daftar tarik-turun.

Jika kebijakan tidak terlihat dalam daftar, pilih tombol refresh di bagian atas daftar. (Jangan me-refresh halaman browser!)

b. Pilih Selanjutnya: Tanda.

c. Pilih Berikutnya: Tinjauan.

d. Untuk nama Peran, ketikkan nama seperti**`lambda_invoke_function_assume_apigw_role`**.

e. Pilih Buat peran.

9. Pilih **`lambda_invoke_function_assume_apigw_role`** dari daftar peran Anda.

10. Pilih tab Trust relationship.

11. Pilih Edit trust relationship (Edit Hubungan Kepercayaan).

12. Ganti kebijakan yang ada dengan yang berikut:

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

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

13. Pilih Perbarui Kebijakan Kepercayaan.
14. Catat peran ARN untuk peran yang baru saja Anda buat. Anda akan membutuhkannya nanti.

Buat fungsi **`Calc`** Lambda

Selanjutnya Anda akan membuat fungsi Lambda menggunakan konsol Lambda.

1. Di konsol Lambda, pilih Buat fungsi.
2. Pilih Penulis dari Scratch.
3. Untuk Nama, masukkan **`Calc`**.
4. Untuk Runtime, pilih runtime Node.js atau Python terbaru yang didukung.
5. Pilih Buat fungsi.
6. Salin fungsi Lambda berikut di runtime pilihan Anda dan tempelkan ke editor kode di konsol Lambda.

   Node.js

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

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

  const res = {};
  res.a = Number(event.a);
  res.b = Number(event.b);
  res.op = event.op;
  if (isNaN(event.a) || isNaN(event.b)) {
```

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

  return res;
};
```

Python

```
import json


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

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

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

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

    return res
```

7. Di bawah Peran eksekusi, pilih Pilih peran yang ada.
8. Masukkan peran ARN untuk **`lambda_invoke_function_assume_apigw_role`** peran yang Anda buat sebelumnya.
9. Pilih Deploy.

Fungsi ini membutuhkan dua operan (adanb) dan operator (op) dari parameter `event` input. Input adalah objek JSON dari format berikut:

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

Fungsi ini mengembalikan hasil yang dihitung (c) dan masukan. Untuk input yang tidak valid, fungsi mengembalikan nilai null atau string “Invalid op” sebagai hasilnya. Outputnya adalah format JSON berikut:

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

Anda harus menguji fungsi di konsol Lambda sebelum mengintegrasikannya dengan API di langkah berikutnya.

Uji fungsi **`Calc`** Lambda

Berikut cara menguji `Calc` fungsi Anda di konsol Lambda:

1. Pilih tab Uji.
2. Untuk nama acara pengujian, masukkan**`calc2plus5`**.
3. Ganti definisi acara pengujian dengan yang berikut:

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

4. Pilih Simpan.
5. Pilih Uji.
6. Perluas Hasil eksekusi: berhasil. Anda akan melihat yang berikut ini:

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

```
}
```

Buat **`Calc`** API

Prosedur berikut menunjukkan cara membuat API untuk fungsi `Calc` Lambda yang baru saja Anda buat. Di bagian berikutnya, Anda akan menambahkan sumber daya dan metode ke dalamnya.

Untuk membuat API

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Jika ini adalah pertama kalinya Anda menggunakan API Gateway, Anda akan melihat halaman yang memperkenalkan Anda ke fitur layanan. Di bawah REST API, pilih Build. Saat munculan Create Example API muncul, pilih OK.

   Jika ini bukan pertama kalinya Anda menggunakan API Gateway, pilih Buat API. Di bawah REST API, pilih Build.
3. Untuk nama API, masukkan**`LambdaCalc`**.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Tetap tetapkan jenis endpoint API ke Regional.
6. Pilih Buat API.

Integrasi 1: Buat **`GET`** metode dengan parameter kueri untuk memanggil fungsi Lambda

Dengan membuat `GET` metode yang meneruskan parameter string kueri ke fungsi Lambda, Anda mengaktifkan API untuk dipanggil dari browser. Pendekatan ini dapat berguna, terutama untuk API yang memungkinkan akses terbuka.

Setelah membuat API, Anda membuat sumber daya. Biasanya, sumber daya API diatur dalam pohon sumber daya sesuai dengan logika aplikasi. Untuk langkah ini, Anda membuat sumber daya /calc.

Untuk membuat sumber daya /calc

1. Pilih Buat sumber daya.
2. Matikan sumber daya Proxy.
3. Pertahankan jalur Sumber Daya sebagai`/`.

4. Untuk Nama sumber daya, masukkan **`calc`**.
5. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).
6. Pilih Buat sumber daya.

Dengan membuat `GET` metode yang meneruskan parameter string kueri ke fungsi Lambda, Anda mengaktifkan API untuk dipanggil dari browser. Pendekatan ini dapat berguna, terutama untuk API yang memungkinkan akses terbuka.

Dalam metode ini, Lambda mengharuskan `POST` permintaan digunakan untuk memanggil fungsi Lambda apa pun. Contoh ini menunjukkan bahwa metode HTTP dalam permintaan metode frontend dapat berbeda dari permintaan integrasi di backend.

Untuk membuat **`GET`** metode

1. Pilih sumber daya /calc, lalu pilih Create method.
2. Untuk tipe Metode, pilih GET.
3. Untuk jenis Integrasi, pilih Layanan AWS.
4. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat fungsi Lambda Anda.
5. Untuk Layanan AWS, pilih Lambda.
6. Biarkan AWS subdomain kosong.
7. Untuk metode HTTP, pilih POST.
8. Untuk tipe Action, pilih Use path override. Opsi ini memungkinkan kita untuk menentukan ARN dari tindakan Invoke untuk menjalankan fungsi kita. `Calc`
9. Untuk Path override, masukkan**`2015-03-31/functions/arn:aws:lambda:`*`us-east-2`*`:`*`account-id`*`:function:Calc/invocations`**. Untuk *`account-id`*, masukkan Akun AWS ID Anda. Untuk*`us-east-2`*, masukkan Wilayah AWS tempat Anda membuat fungsi Lambda Anda.
10. Untuk peran Eksekusi, masukkan peran ARN untuk. **`lambda_invoke_function_assume_apigw_role`**
11. Jangan mengubah pengaturan cache Credential dan batas waktu default.
12. Pilih metode Buat.

Sekarang Anda mengatur parameter kueri untuk metode GET pada sumber daya /calc sehingga dapat menerima input atas nama fungsi Lambda backend.

Untuk mengatur parameter string kueri

1. Pada tab Permintaan metode, di bawah Pengaturan permintaan metode, pilih Edit.
2. Untuk validator Permintaan, pilih Validasi parameter string kueri dan header. Pengaturan ini akan menyebabkan pesan kesalahan kembali jika klien tidak menentukan parameter yang diperlukan.
3. Pilih parameter string kueri URL dan lakukan hal berikut:

   a. Pilih Tambahkan string kueri.

   b. Untuk Nama, masukkan **`operand1`**.

   c. Aktifkan Diperlukan.

   d. Tetap caching dimatikan.

   Ulangi langkah yang sama dan buat string kueri bernama **`operand2`** dan string kueri bernama**`operator`**.
4. Pilih Simpan.

Sekarang, Anda membuat template pemetaan untuk menerjemahkan string kueri yang disediakan klien ke payload permintaan integrasi seperti yang dipersyaratkan oleh fungsi. `Calc` Template ini memetakan tiga parameter string kueri yang dideklarasikan dalam permintaan Metode ke dalam nilai properti yang ditunjuk dari objek JSON sebagai masukan ke fungsi Lambda backend. Objek JSON yang diubah akan dimasukkan sebagai payload permintaan integrasi.

Untuk memetakan parameter input ke permintaan integrasi

1. Pada tab Permintaan integrasi, di bawah Pengaturan permintaan integrasi, pilih Edit.
2. Untuk Request body passthrough, pilih Bila tidak ada templat yang ditentukan (disarankan).
3. Pilih template Pemetaan.
4. Pilih Tambahkan templat pemetaan.
5. Untuk jenis Konten, masukkan**`application/json`**.
6. Untuk badan Template, masukkan kode berikut:

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

```
}
```

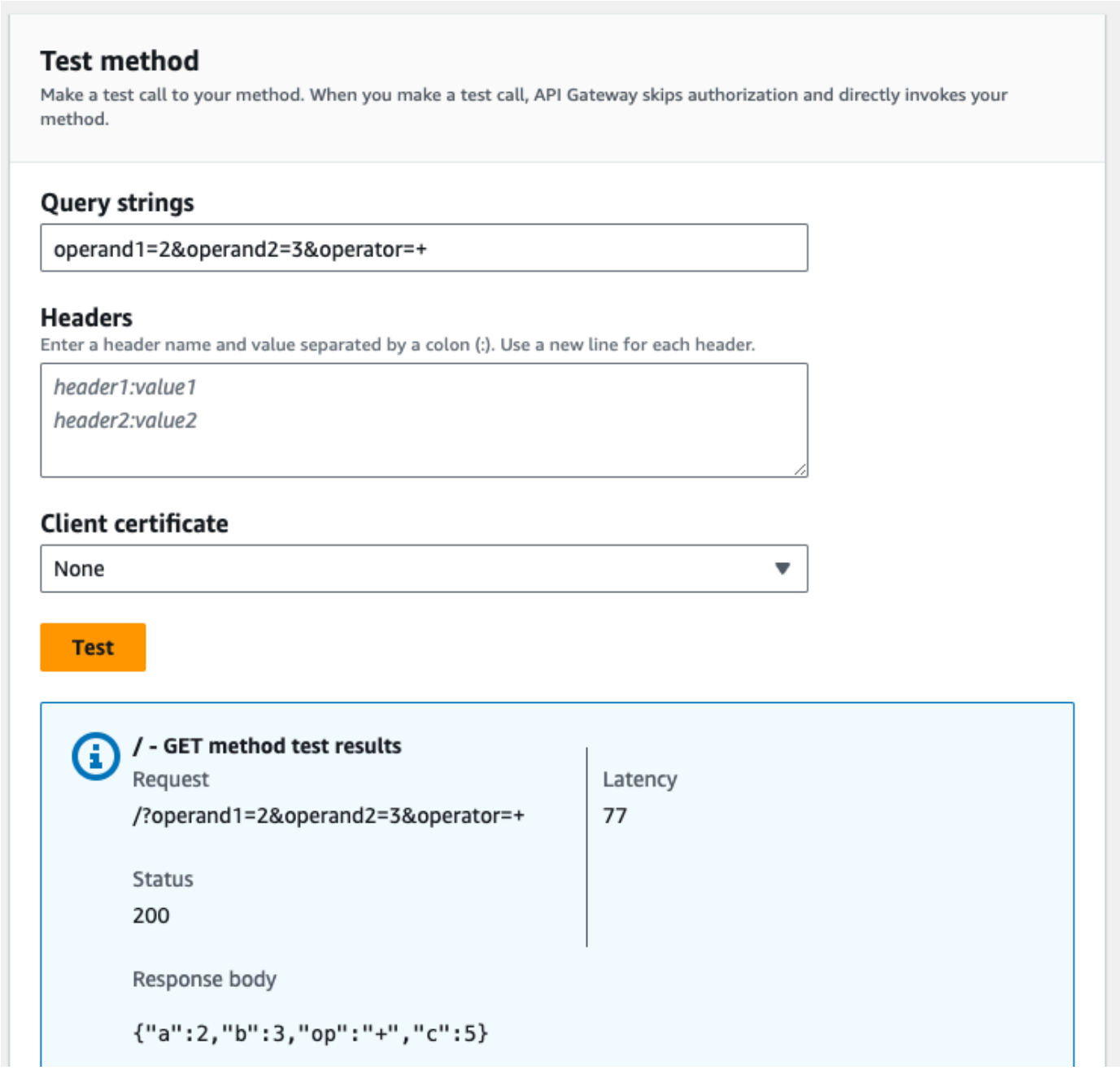
7. Pilih Simpan.

Anda sekarang dapat menguji `GET` metode Anda untuk memverifikasi bahwa metode tersebut telah diatur dengan benar untuk menjalankan fungsi Lambda.

Untuk menguji **`GET`** metode

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Untuk string Query, masukkan**`operand1=2&operand2=3&operator=+`**.
3. Pilih Uji.

   Hasilnya akan terlihat mirip dengan ini:



Integrasi 2: Buat **POST** metode dengan payload JSON untuk memanggil fungsi Lambda

Dengan membuat POST metode dengan payload JSON untuk memanggil fungsi Lambda, Anda membuatnya sehingga klien harus memberikan input yang diperlukan ke fungsi backend di badan permintaan. Untuk memastikan bahwa klien mengunggah data input yang benar, Anda akan mengaktifkan validasi permintaan pada payload.

Untuk membuat **`POST`** metode dengan payload JSON

1. Pilih sumber daya /calc, lalu pilih Create method.
2. Untuk jenis Metode, pilih POST.
3. Untuk jenis Integrasi, pilih Layanan AWS.
4. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat fungsi Lambda Anda.
5. Untuk Layanan AWS, pilih Lambda.
6. Biarkan AWS subdomain kosong.
7. Untuk metode HTTP, pilih POST.
8. Untuk tipe Action, pilih Use path override. Opsi ini memungkinkan kita untuk menentukan ARN dari tindakan Invoke untuk menjalankan fungsi kita. `Calc`
9. Untuk Path override, masukkan**`2015-03-31/functions/arn:aws:lambda:`*`us-east-2`*`:`*`account-id`*`:function:Calc/invocations`**. Untuk *`account-id`*, masukkan Akun AWS ID Anda. Untuk*`us-east-2`*, masukkan Wilayah AWS tempat Anda membuat fungsi Lambda Anda.
10. Untuk peran Eksekusi, masukkan peran ARN untuk. **`lambda_invoke_function_assume_apigw_role`**
11. Jangan mengubah pengaturan cache Credential dan batas waktu default.
12. Pilih metode Buat.

Sekarang Anda membuat model input untuk menggambarkan struktur data input dan memvalidasi badan permintaan yang masuk.

Untuk membuat model input

1. Di panel navigasi utama, pilih Model.
2. Pilih Buat model.
3. Untuk Nama, masukkan **`input`**.
4. Untuk jenis Konten, masukkan**`application/json`**.

   Jika tidak ada jenis konten yang cocok ditemukan, validasi permintaan tidak dilakukan. Untuk menggunakan model yang sama terlepas dari jenis konten, masukkan**`$default`**.
5. Untuk skema Model, masukkan model berikut:

```
{
```

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

6. Pilih Buat model.

Anda sekarang membuat model output. Model ini menjelaskan struktur data dari output yang dihitung dari backend. Ini dapat digunakan untuk memetakan data respons integrasi ke model yang berbeda. Tutorial ini bergantung pada perilaku passthrough dan tidak menggunakan model ini.

Untuk membuat model keluaran

1. Pilih Buat model.
2. Untuk Nama, masukkan **`output`**.
3. Untuk jenis Konten, masukkan**`application/json`**.

   Jika tidak ada jenis konten yang cocok ditemukan, validasi permintaan tidak dilakukan. Untuk menggunakan model yang sama terlepas dari jenis konten, masukkan**`$default`**.
4. Untuk skema Model, masukkan model berikut:

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

5. Pilih Buat model.

Anda sekarang membuat model hasil. Model ini menjelaskan struktur data dari data respons yang dikembalikan. Ini mereferensikan skema input dan output yang ditentukan dalam API Anda.

Untuk membuat model hasil

1. Pilih Buat model.
2. Untuk Nama, masukkan **`result`**.
3. Untuk jenis Konten, masukkan**`application/json`**.

   Jika tidak ada jenis konten yang cocok ditemukan, validasi permintaan tidak dilakukan. Untuk menggunakan model yang sama terlepas dari jenis konten, masukkan**`$default`**.
4. Untuk skema Model, masukkan model berikut dengan *`restapi-id`* Anda. *`Restapi-id`* Anda tercantum dalam tanda kurung di bagian atas konsol dalam alur berikut: `API Gateway > APIs > LambdaCalc (`*`abc123`*`).`

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

Anda sekarang mengonfigurasi permintaan metode metode POST Anda untuk mengaktifkan validasi permintaan pada badan permintaan yang masuk.

Untuk mengaktifkan validasi permintaan pada metode POST

1. Di panel navigasi utama, pilih Sumber daya, lalu pilih `POST` metode dari pohon sumber daya.
2. Pada tab Permintaan metode, di bawah Pengaturan permintaan metode, pilih Edit.
3. Untuk validator Permintaan, pilih Validasi isi.
4. Pilih Request body, lalu pilih Add model.
5. Untuk jenis Konten, masukkan**`application/json`**.

Jika tidak ada jenis konten yang cocok ditemukan, validasi permintaan tidak dilakukan. Untuk menggunakan model yang sama terlepas dari jenis konten, masukkan**`$default`**.

6. Untuk Model, pilih input.
7. Pilih Simpan.

Anda sekarang dapat menguji `POST` metode Anda untuk memverifikasi bahwa metode tersebut telah diatur dengan benar untuk menjalankan fungsi Lambda.

Untuk menguji **`POST`** metode

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Untuk badan Permintaan, masukkan payload JSON berikut.

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

3. Pilih Uji.

   Anda akan melihat output berikut:

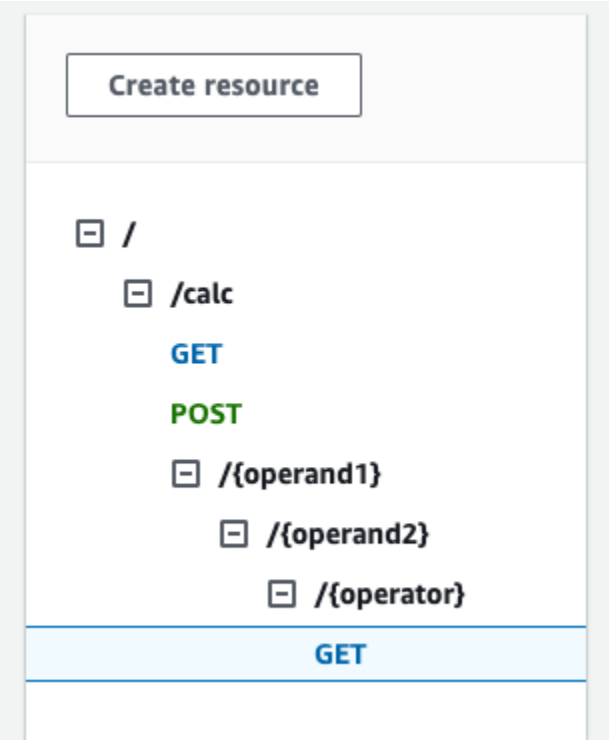
```
{
  "a": 1,
  "b": 2,
  "op": "+",
  "c": 3
}
```

Integrasi 3: Buat **`GET`** metode dengan parameter jalur untuk memanggil fungsi Lambda

Sekarang Anda akan membuat `GET` metode pada sumber daya yang ditentukan oleh urutan parameter jalur untuk memanggil fungsi Lambda backend. Nilai parameter jalur menentukan data

input ke fungsi Lambda. Anda akan menggunakan template pemetaan untuk memetakan nilai parameter jalur masuk ke payload permintaan integrasi yang diperlukan.

Struktur sumber daya API yang dihasilkan akan terlihat seperti ini:



Untuk membuat sumber daya/{operand1}/{operand2}/{operator}

1. Pilih Buat sumber daya.
2. Untuk jalur Sumber Daya, pilih`/calc`.
3. Untuk Nama sumber daya, masukkan **`{operand1}`**.
4. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).
5. Pilih Buat sumber daya.
6. Untuk jalur Sumber Daya, pilih`/calc/{operand1}/`.
7. Untuk Nama sumber daya, masukkan **`{operand2}`**.
8. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).
9. Pilih Buat sumber daya.
10. Untuk jalur Sumber Daya, pilih`/calc/{operand1}/{operand2}/`.
11. Untuk Nama sumber daya, masukkan **`{operator}`**.
12. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).

13. Pilih Buat sumber daya.

Kali ini Anda akan menggunakan integrasi Lambda bawaan di konsol API Gateway untuk menyiapkan integrasi metode.

Untuk mengatur integrasi metode

1. Pilih sumber daya/{operand1}/{operand2}/{operator}, lalu pilih Create method.
2. Untuk tipe Metode, pilih GET.
3. Untuk jenis Integrasi, pilih Lambda.
4. Matikan integrasi proxy Lambda.
5. Untuk fungsi Lambda, pilih Wilayah AWS tempat Anda membuat fungsi Lambda dan masukkan. **`Calc`**
6. Tetap aktifkan batas waktu default.
7. Pilih metode Buat.

Anda sekarang membuat template pemetaan untuk memetakan tiga parameter jalur URL, dideklarasikan ketika sumber daya /calc/ {operand1}/{operand2}/{operator} dibuat, ke dalam nilai properti yang ditunjuk di objek JSON. Karena jalur URL harus dikodekan URL, operator divisi harus ditentukan sebagai pengganti. `%2F` / Template ini menerjemahkan `%2F` ke dalam `'/'` sebelum meneruskannya ke fungsi Lambda.

Untuk membuat template pemetaan

1. Pada tab Permintaan integrasi, di bawah Pengaturan permintaan integrasi, pilih Edit.
2. Untuk Request body passthrough, pilih Bila tidak ada templat yang ditentukan (disarankan).
3. Pilih template Pemetaan.
4. Untuk jenis Konten, masukkan**`application/json`**.
5. Untuk badan Template, masukkan kode berikut:

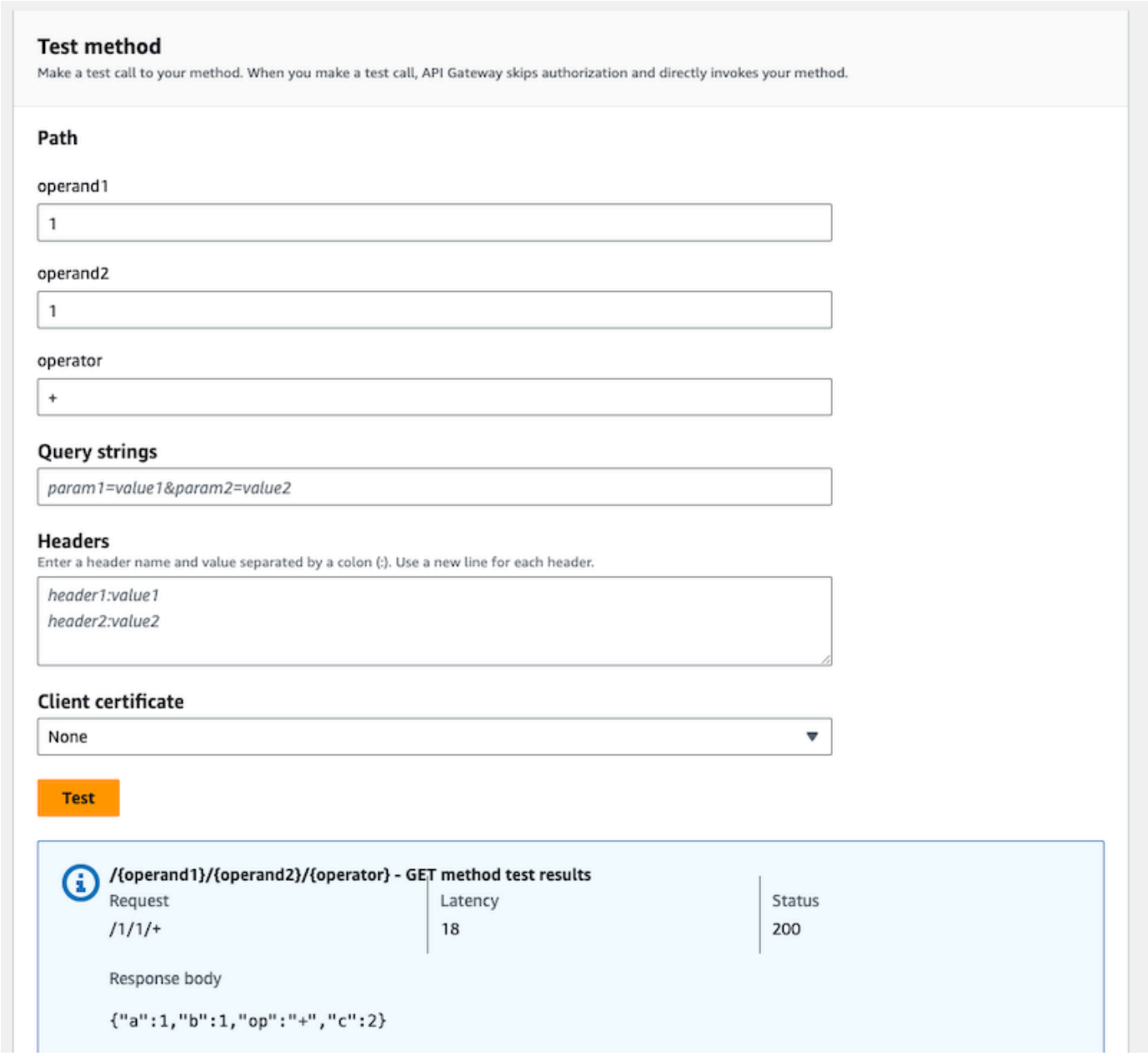
```
{
   "a": "$input.params('operand1')",
   "b": "$input.params('operand2')",
   "op":
 #if($input.params('operator')=='%2F')"/"#{else}"$input.params('operator')"#end
}
```

6. Pilih Simpan.

Anda sekarang dapat menguji `GET` metode Anda untuk memverifikasi bahwa itu telah diatur dengan benar untuk menjalankan fungsi Lambda dan meneruskan output asli melalui respons integrasi tanpa pemetaan.

Untuk menguji **`GET`** metode

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Untuk Jalan, lakukan hal berikut:
   a. Untuk operand1, masukkan. **1**
   b. Untuk operand2, masukkan. **1**
   c. Untuk operator, masukkan**+**.
3. Pilih Uji.
4. Hasilnya akan terlihat seperti ini:



Selanjutnya, Anda memodelkan struktur data payload respons metode setelah `result` skema.

Secara default, badan respons metode diberi model kosong. Ini akan menyebabkan badan respons integrasi dilewatkan tanpa pemetaan. Namun, ketika Anda membuat SDK untuk salah satu bahasa tipe kuat, seperti Java atau Objective-C, pengguna SDK Anda akan menerima objek kosong sebagai hasilnya. Untuk memastikan bahwa klien REST dan klien SDK menerima hasil yang diinginkan, Anda harus memodelkan data respons menggunakan skema yang telah ditentukan. Di sini Anda akan mendefinisikan model untuk badan respons metode dan untuk membuat template pemetaan untuk menerjemahkan badan respons integrasi ke dalam badan respons metode.

Untuk membuat respons metode

1. Pada tab Respons Metode, di bawah Respons 200, pilih Edit.
2. Di bawah Badan respons, pilih Tambahkan model.
3. Untuk jenis Konten, masukkan**`application/json`**.
4. Untuk Model, pilih hasil.
5. Pilih Simpan.

Menyetel model untuk badan respons metode memastikan bahwa data respons akan dilemparkan ke `result` objek SDK tertentu. Untuk memastikan bahwa data respons integrasi dipetakan sesuai, Anda memerlukan template pemetaan.

Untuk membuat template pemetaan

1. Pada tab Respons integrasi, di bawah Default - Respons, pilih Edit.
2. Pilih template Pemetaan.
3. Untuk jenis Konten, masukkan**`application/json`**.
4. Untuk Buat template, pilih hasil.
5. Ubah template pemetaan yang dihasilkan agar sesuai dengan yang berikut:

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

6. Pilih Simpan.

Untuk menguji template pemetaan

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.

2. Untuk Jalan, lakukan hal berikut:

   a. Untuk operand1, masukkan. **1**

   b. Untuk operand2, masukkan. **2**

   c. Untuk operator, masukkan**+**.

3. Pilih Uji.

4. Hasilnya akan terlihat seperti berikut:

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

Pada titik ini, Anda hanya dapat memanggil API menggunakan fitur Uji di konsol API Gateway. Untuk membuatnya tersedia bagi klien, Anda harus menerapkan API Anda. Selalu pastikan untuk menerapkan ulang API Anda setiap kali Anda menambahkan, memodifikasi, atau menghapus sumber daya atau metode, memperbarui pemetaan data, atau memperbarui pengaturan tahap. Jika tidak, fitur atau pembaruan baru tidak akan tersedia untuk klien API Anda. sebagai berikut:

Untuk menerapkan API

1. Pilih Deploy API.

2. Untuk Stage, pilih New stage.

3. Untuk nama Panggung, masukkan**`Prod`**.

4. (Opsional) Untuk Deskripsi, masukkan deskripsi.

5. Pilih Deploy.

6. (Opsional) Di bawah detail Tahap, untuk Invoke URL, Anda dapat memilih ikon salin untuk menyalin URL pemanggilan API Anda. Anda dapat menggunakan ini dengan alat seperti Postman dan cURL untuk menguji API Anda.

> **Note**
>
> Selalu terapkan ulang API Anda setiap kali Anda menambahkan, memodifikasi, atau menghapus sumber daya atau metode, memperbarui pemetaan data, atau memperbarui pengaturan tahap. Jika tidak, fitur atau pembaruan baru tidak akan tersedia untuk klien API Anda.

Definisi OpenAPI dari contoh API terintegrasi dengan fungsi Lambda

OpenAPI 2.0

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

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

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

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

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

```
        "a",
        "b",
        "op"
      ],
      "properties": {
        "a": {
          "type": "number"
        },
        "b": {
          "type": "number"
        },
        "op": {
          "type": "string",
          "description": "binary op of ['+', 'add', '-', 'sub', '*', 'mul', '%2F',
'div']"
        }
      },
      "title": "Input"
    },
    "Output": {
      "type": "object",
      "properties": {
        "c": {
          "type": "number"
        }
      },
      "title": "Output"
    },
    "Result": {
      "type": "object",
      "properties": {
        "input": {
          "$ref": "#/definitions/Input"
        },
        "output": {
          "$ref": "#/definitions/Output"
        }
      },
      "title": "Result"
    }
  },
  "x-amazon-apigateway-request-validators": {
    "Validate body": {
      "validateRequestParameters": false,
```

```
      "validateRequestBody": true
    },
    "Validate query string parameters and headers": {
      "validateRequestParameters": true,
      "validateRequestBody": false
    }
  }
}
```

Tutorial: Membuat REST API sebagai proxy Amazon S3 di API Gateway

Sebagai contoh untuk menampilkan menggunakan REST API di API Gateway ke proxy Amazon S3, bagian ini menjelaskan cara membuat dan mengonfigurasi REST API untuk mengekspos operasi Amazon S3 berikut:

- Paparkan GET di sumber daya root API untuk mencantumkan semua bucket Amazon S3 pemanggil.
- Paparkan GET pada resource Folder untuk melihat daftar semua objek di bucket Amazon S3.
- Paparkan GET pada resource Folder/Item untuk melihat atau mengunduh objek dari bucket Amazon S3.

Anda mungkin ingin mengimpor API sampel sebagai proxy Amazon S3, seperti yang ditunjukkan pada. Definisi OpenAPI dari API sampel sebagai proxy Amazon S3 Sampel ini berisi metode yang lebih terbuka. Untuk petunjuk tentang cara mengimpor API menggunakan definisi OpenAPI, lihat. Mengkonfigurasi REST API menggunakan OpenAPI

> **Note**
>
> Untuk mengintegrasikan API Gateway API Anda dengan Amazon S3, Anda harus memilih wilayah tempat layanan API Gateway dan Amazon S3 tersedia. Untuk ketersediaan wilayah, lihat Titik Akhir dan Kuota Amazon API Gateway.

Topik

- Siapkan izin IAM untuk API untuk menjalankan tindakan Amazon S3
- Buat sumber daya API untuk mewakili sumber daya Amazon S3

- Paparkan metode API untuk mencantumkan bucket Amazon S3 pemanggil
- Mengekspos metode API untuk mengakses bucket Amazon S3
- Mengekspos metode API untuk mengakses objek Amazon S3 dalam bucket
- Definisi OpenAPI dari API sampel sebagai proxy Amazon S3
- Panggil API menggunakan klien REST API

Siapkan izin IAM untuk API untuk menjalankan tindakan Amazon S3

Untuk mengizinkan API menjalankan tindakan Amazon S3, Anda harus memiliki kebijakan IAM yang sesuai yang dilampirkan ke peran IAM.

Untuk membuat peran eksekusi proxy AWS layanan

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Peran.
3. Pilih Buat peran.
4. Pilih AWSlayanan di bawah Pilih jenis entitas tepercaya, lalu pilih API Gateway dan pilih Izinkan API Gateway untuk mendorong CloudWatch log ke Log.
5. Pilih Berikutnya, lalu pilih Berikutnya.
6. Untuk nama Peran**`APIGatewayS3ProxyPolicy`**, masukkan, lalu pilih Buat peran.
7. Dalam daftar Peran, pilih peran yang baru saja Anda buat. Anda mungkin perlu menggulir atau menggunakan bilah pencarian untuk menemukan peran.
8. Untuk peran yang dipilih, pilih tab Tambahkan izin.
9. Pilih Lampirkan kebijakan dari daftar dropdown.
10. Di bilah pencarian, masukkan **`AmazonS3FullAccess`** dan pilih Tambahkan izin.

> **Note**
>
> Tutorial ini menggunakan kebijakan terkelola untuk kesederhanaan. Sebagai praktik terbaik, Anda harus membuat kebijakan IAM Anda sendiri untuk memberikan izin minimum yang diperlukan.

11. Perhatikan ARN Peran yang baru dibuat, Anda akan menggunakannya nanti.

Buat sumber daya API untuk mewakili sumber daya Amazon S3

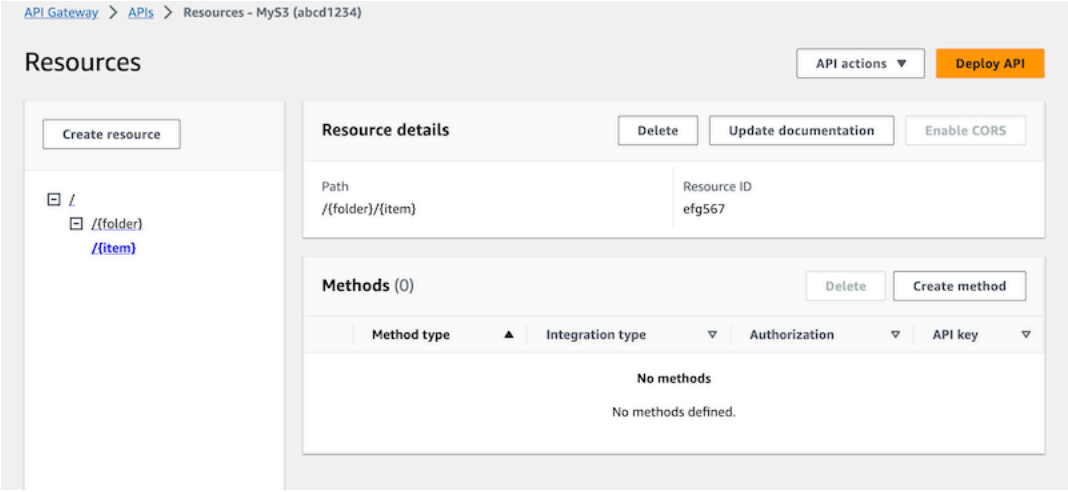
Kami akan menggunakan sumber daya root (/) API sebagai wadah bucket Amazon S3 pemanggil yang diautentikasi. Kami juga akan membuat `Folder` dan `Item` sumber daya untuk mewakili bucket Amazon S3 tertentu dan objek Amazon S3 tertentu, masing-masing. Nama folder dan kunci objek akan ditentukan, dalam bentuk parameter jalur sebagai bagian dari URL permintaan, oleh pemanggil.

> **Note**
>
> Saat mengakses objek yang kunci objeknya termasuk / atau karakter khusus lainnya, karakter harus dikodekan URL. Misalnya, `test/test.txt` harus dikodekan ke. `test%2Ftest.txt`

Untuk membuat sumber daya API yang mengekspos fitur layanan Amazon S3

1. Saat Wilayah AWS Anda membuat bucket Amazon S3, buat API bernama myS3. Sumber daya root API ini (/) mewakili layanan Amazon S3. Pada langkah ini, Anda membuat dua sumber daya tambahan/{folder} dan/{item}.
2. Pilih sumber daya root API, lalu pilih Buat sumber daya.
3. Matikan sumber daya Proxy.
4. Untuk jalur Sumber Daya, pilih/.
5. Untuk Nama sumber daya, masukkan **`{folder}`**.
6. Biarkan CORS (Cross Origin Resource Sharing) tidak dicentang.
7. Pilih Buat sumber daya.
8. Pilih sumber daya/{folder}, lalu pilih Buat sumber daya.
9. Gunakan langkah-langkah sebelumnya untuk membuat sumber daya turunan dari/{folder} bernama**`{item}`**.

   API final Anda akan terlihat mirip dengan yang berikut ini:



Paparkan metode API untuk mencantumkan bucket Amazon S3 pemanggil

Mendapatkan daftar bucket Amazon S3 dari penelepon melibatkan pemanggilan tindakan GET Service di Amazon S3. Pada sumber daya root API, (/), buat metode GET. Konfigurasikan metode GET untuk diintegrasikan dengan Amazon S3, sebagai berikut.

Untuk membuat dan menginisialisasi metode API **`GET /`**

1. Pilih sumber daya/, lalu pilih Create method.
2. Untuk jenis metode, pilih GET.
3. Untuk jenis Integrasi, pilih Layanan AWS.
4. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat bucket Amazon S3.
5. Untuk Layanan AWS, pilih Amazon Simple Storage Service.
6. Biarkan AWSsubdomain kosong.
7. Untuk metode HTTP, pilih GET.
8. Untuk tipe Action, pilih Use path override. Dengan penggantian jalur, API Gateway meneruskan permintaan klien ke Amazon S3 sebagai permintaan gaya jalur API Amazon S3 REST API yang sesuai, di mana sumber daya Amazon S3 diekspresikan oleh jalur sumber daya pola. `s3-host-name/bucket/key` API Gateway menyetel `s3-host-name` dan meneruskan klien yang ditentukan `bucket` dan `key` dari klien ke Amazon S3.
9. Untuk Path override, masukkan/.
10. Untuk peran Eksekusi, masukkan peran ARN untuk. **`APIGatewayS3ProxyPolicy`**
11. Pilih metode Buat.

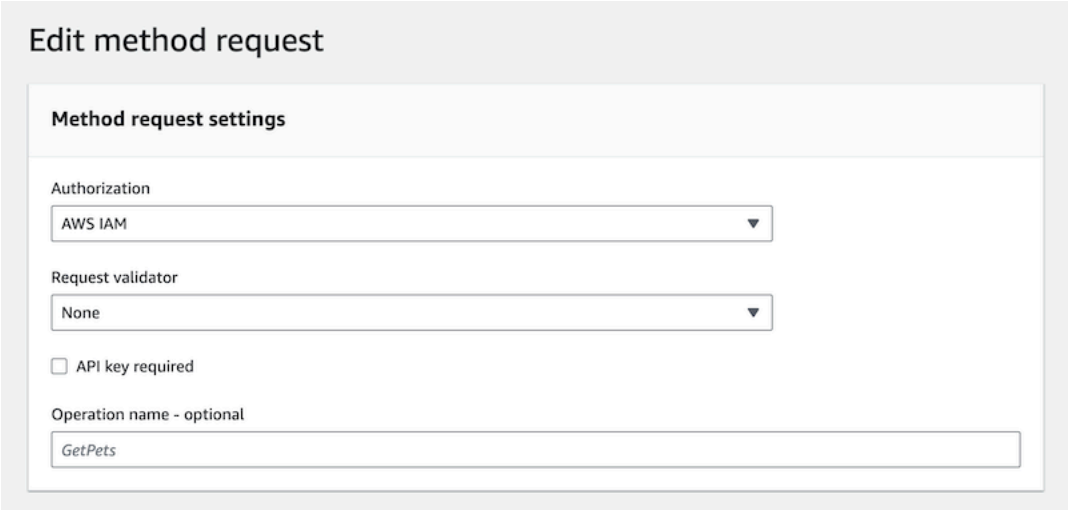
Pengaturan ini mengintegrasikan `GET https://your-api-host/stage/` permintaan frontend dengan backend. `GET https://your-s3-host/`

> **Note**
>
> Setelah penyiapan awal, Anda dapat mengubah pengaturan ini di halaman Permintaan Integrasi metode.

Untuk mengontrol siapa yang dapat memanggil metode API kami ini, kami mengaktifkan flag otorisasi metode dan mengaturnya ke`AWS_IAM`.

Untuk mengaktifkan IAM untuk mengontrol akses ke metode GET/

1. Pada tab Permintaan metode, di bawah Pengaturan permintaan metode, pilih Edit.
2. Untuk Otorisasi, dari menu tarik-turun, pilih. `AWS_IAM`



3. Pilih Simpan.

Agar API kami mengembalikan respons dan pengecualian yang berhasil dengan benar kepada penelepon, mari kita mendeklarasikan tanggapan 200, 400, dan 500 di Method Response. Kami menggunakan pemetaan default untuk 200 tanggapan sehingga tanggapan backend dari kode status yang tidak dideklarasikan di sini akan dikembalikan ke pemanggil sebagai 200 respons.

Untuk mendeklarasikan tipe respons untuk metode **`GET /`**

1. Pada tab Respons Metode, di bawah Respons 200, pilih Edit.

2. Pilih Add header dan lakukan hal berikut:

   a. Untuk nama Header, masukkan**`Content-Type`**.

   b. Pilih Tambahkan header.

   Ulangi langkah-langkah ini untuk membuat **`Timestamp`** header dan **`Content-Length`** header.

3. Pilih Simpan.
4. Pada tab Respons metode, di bawah Respons metode, pilih Buat respons.
5. Untuk kode status HTTP, masukkan 400.

   Anda tidak menetapkan header apa pun untuk respons ini.

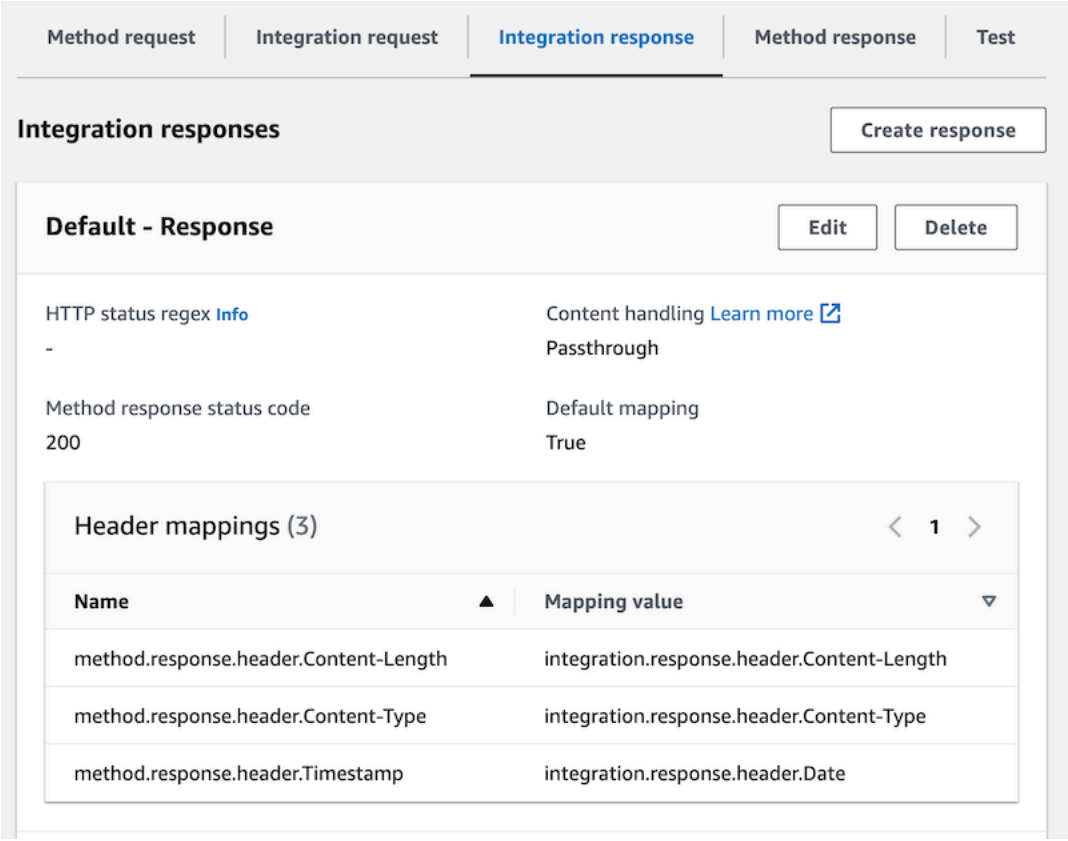
6. Pilih Simpan.
7. Ulangi langkah-langkah berikut untuk membuat respons 500.

   Anda tidak menetapkan header apa pun untuk respons ini.

Karena respons integrasi yang berhasil dari Amazon S3 mengembalikan bucket list sebagai payload XHTML dan respons metode default dari API Gateway mengembalikan payload JSON, kita harus memetakan nilai parameter header Content-Type backend ke mitra frontend. Jika tidak, klien akan menerima `application/json` untuk tipe konten ketika badan respons sebenarnya adalah string XHTML. Prosedur berikut menunjukkan cara mengaturnya. Selain itu, kami juga ingin menampilkan parameter header lainnya kepada klien, seperti Tanggal dan Panjang Konten.

Untuk mengatur pemetaan header respons untuk metode GET/

1. Pada tab Respons integrasi, di bawah Default - Respons, pilih Edit.
2. Untuk header Content-Length, masukkan **`integration.response.header.Content-Length`** untuk nilai pemetaan.
3. Untuk header Content-Type, masukkan **`integration.response.header.Content-Type`** untuk nilai pemetaan.
4. Untuk header Timestamp, masukkan **`integration.response.header.Date`** untuk nilai pemetaan.
5. Pilih Simpan. Hasilnya akan terlihat mirip dengan yang berikut:

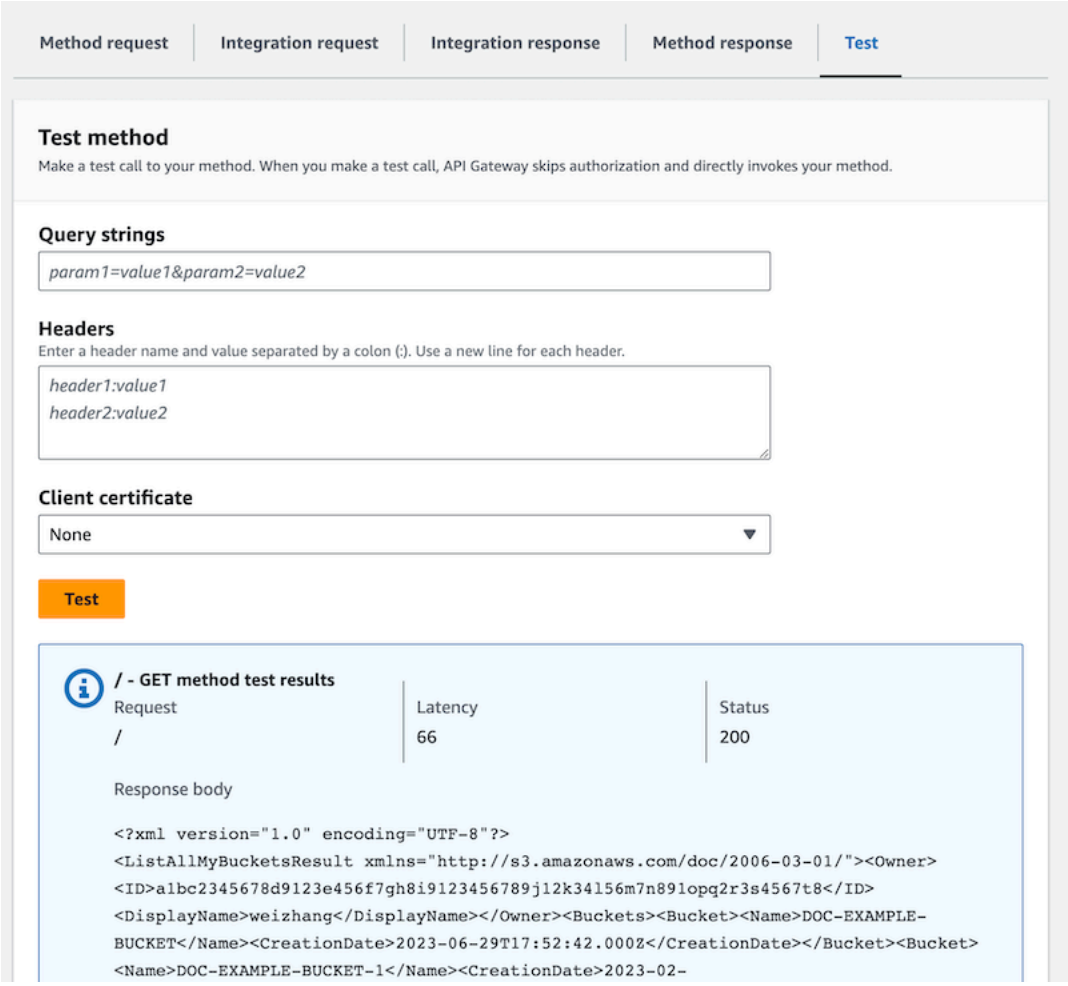


6. Pada tab Respons Integrasi, di bawah Respons integrasi, pilih Buat respons.
7. Untuk regex status HTTP, masukkan. **`4\d{2}`** Ini memetakan semua kode status respons HTTP 4xx ke respons metode.
8. Untuk kode status respons Metode, pilih**`400`**.
9. Pilih Buat.
10. Ulangi langkah-langkah berikut untuk membuat respons integrasi untuk respons metode 500. Untuk regex status HTTP, masukkan. **`5\d{2}`**

Sebagai praktik yang baik, mari kita uji API yang telah kita konfigurasikan sejauh ini.

Untuk menguji **`GET /`** metode

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Pilih Uji. Hasilnya akan terlihat seperti gambar berikut:



Mengekspos metode API untuk mengakses bucket Amazon S3

Untuk bekerja dengan bucket Amazon S3, kami mengekspos `GET` metode pada sumber daya/{folder} untuk mencantumkan objek dalam bucket. Instruksi mirip dengan yang dijelaskan dalamPaparkan metode API untuk mencantumkan bucket Amazon S3 pemanggil. Untuk metode lainnya, Anda dapat mengimpor contoh API di sini,Definisi OpenAPI dari API sampel sebagai proxy Amazon S3.

Untuk mengekspos metode GET pada sumber daya folder

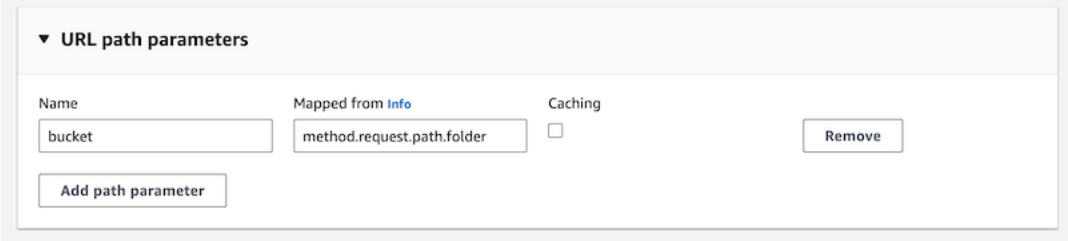
1. Pilih sumber daya/{folder}, lalu pilih Create method.
2. Untuk jenis metode, pilih GET.
3. Untuk jenis Integrasi, pilih Layanan AWS.
4. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat bucket Amazon S3.
5. Untuk Layanan AWS, pilih Amazon Simple Storage Service.

6. Biarkan AWSsubdomain kosong.
7. Untuk metode HTTP, pilih GET.
8. Untuk tipe Action, pilih Use path override.
9. Untuk Path override, masukkan**`{bucket}`**.
10. Untuk peran Eksekusi, masukkan peran ARN untuk. **`APIGatewayS3ProxyPolicy`**
11. Pilih metode Buat.

Anda mengatur parameter `{folder}` jalur di URL titik akhir Amazon S3. Anda perlu memetakan parameter `{folder}` jalur permintaan metode ke parameter `{bucket}` jalur permintaan integrasi.

Untuk memetakan **`{folder}`** ke **`{bucket}`**

1. Pada tab Permintaan integrasi, di bawah Pengaturan permintaan integrasi, pilih Edit.
2. Pilih parameter jalur URL, lalu pilih Tambahkan parameter jalur.
3. Untuk Nama, masukkan **`bucket`**.
4. Untuk Dipetakan dari, masukkan**`method.request.path.folder`**. Konfigurasi akan terlihat mirip dengan yang berikut ini:



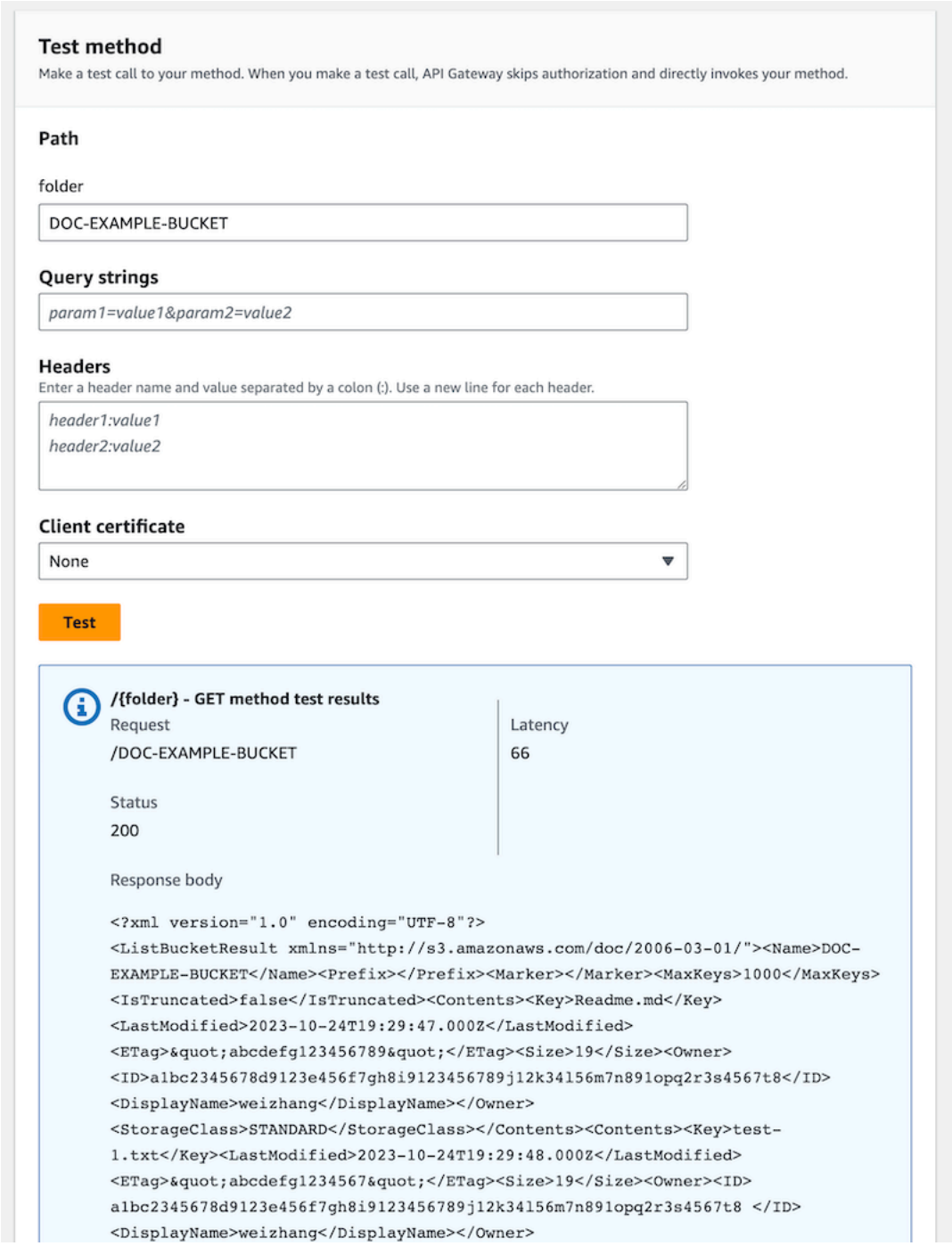
5. Pilih Simpan.

Sekarang, Anda menguji API Anda.

Untuk menguji **`/{folder} GET`** metode.

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Di bawah Path, untuk folder, masukkan nama bucket Anda.
3. Pilih Uji.

   Hasil tes akan berisi daftar objek di bucket Anda.



Mengekspos metode API untuk mengakses objek Amazon S3 dalam bucket

Amazon S3 mendukung tindakan GET, DELETE, HEAD, OPTIONS, POST, dan PUT untuk mengakses dan mengelola objek dalam bucket tertentu. Dalam tutorial ini, Anda mengekspos `GET` metode pada `{folder}/{item}` sumber daya untuk mendapatkan gambar dari ember. Untuk

aplikasi `{folder}/{item}` sumber daya lainnya, lihat contoh API,Definisi OpenAPI dari API sampel sebagai proxy Amazon S3.

Untuk mengekspos metode GET pada sumber daya item

1. Pilih sumber daya/{item}, lalu pilih Create method.
2. Untuk jenis metode, pilih GET.
3. Untuk jenis Integrasi, pilih Layanan AWS.
4. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat bucket Amazon S3.
5. Untuk Layanan AWS, pilih Amazon Simple Storage Service.
6. Biarkan AWSsubdomain kosong.
7. Untuk metode HTTP, pilih GET.
8. Untuk tipe Action, pilih Use path override.
9. Untuk penggantian Path, masukkan {bucket}/{object}.
10. Untuk peran Eksekusi, masukkan peran ARN untuk. **`APIGatewayS3ProxyPolicy`**
11. Pilih metode Buat.

Anda mengatur parameter `{folder}` dan `{item}` jalur di URL titik akhir Amazon S3. Anda perlu memetakan parameter jalur permintaan metode ke parameter jalur permintaan integrasi.

Dalam langkah ini, Anda melakukan hal berikut:

- Petakan parameter `{folder}` jalur permintaan metode ke parameter `{bucket}` jalur permintaan integrasi.
- Petakan parameter `{item}` jalur permintaan metode ke parameter `{object}` jalur permintaan integrasi.

Untuk memetakan **`{folder}`** ke **`{bucket}`** dan **`{item}`** ke **`{object}`**

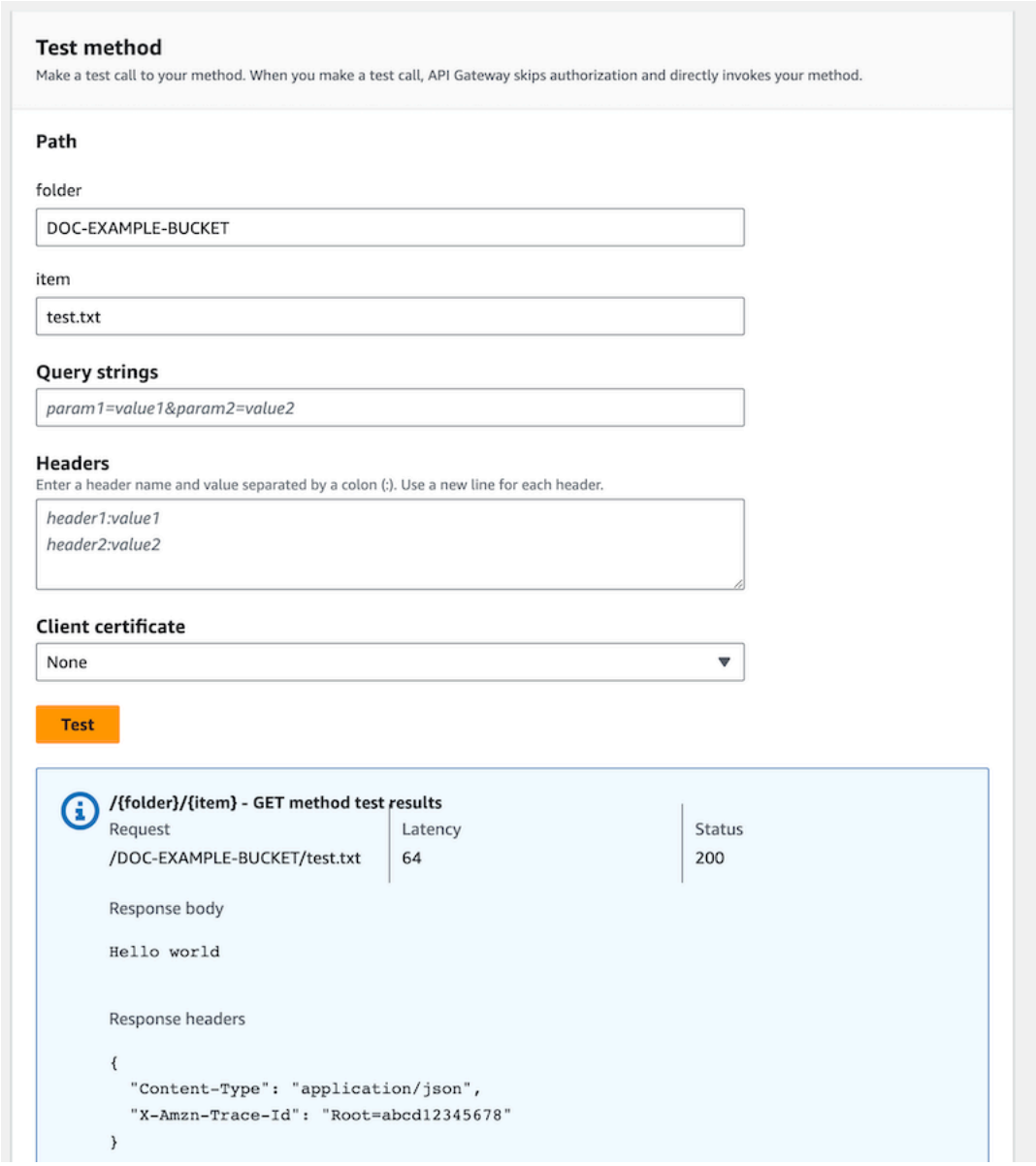
1. Pada tab Permintaan integrasi, di bawah Pengaturan permintaan integrasi, pilih Edit.
2. Pilih parameter jalur URL.
3. Pilih Tambahkan parameter jalur.
4. Untuk Nama, masukkan **`bucket`**.
5. Untuk Dipetakan dari, masukkan**`method.request.path.folder`**.

6. Pilih Tambahkan parameter jalur.
7. Untuk Nama, masukkan **`object`**.
8. Untuk Dipetakan dari, masukkan**`method.request.path.item`**.
9. Pilih Simpan.

Untuk menguji **`/{folder}/{object} GET`** metode.

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Di bawah Path, untuk folder, masukkan nama bucket Anda.
3. Di bawah Path, untuk item, masukkan nama item.
4. Pilih Uji.

   Badan respons akan berisi isi item.



Permintaan mengembalikan teks biasa (“Hello world”) dengan benar sebagai konten file yang ditentukan (test.txt) di bucket Amazon S3 yang diberikan (DOC-EXAMPLE-BUCKET).

Untuk mengunduh atau mengunggah file biner, yang di API Gateway dianggap sebagai hal apa pun selain konten JSON yang dikodekan utf-8, diperlukan pengaturan API tambahan. Ini diuraikan sebagai berikut:

Untuk mengunduh atau mengunggah file biner dari S3

1. Daftarkan jenis media dari file yang terpengaruh ke API binaryMediaTypes. Anda dapat melakukan ini di konsol:

a. Pilih pengaturan API untuk API.
b. Di bawah Jenis media biner, pilih Kelola jenis media.
c. Pilih Tambahkan jenis media biner, lalu masukkan jenis media yang diperlukan, misalnya,`image/png`.
d. Pilih Simpan perubahan untuk menyimpan pengaturan.

2. Tambahkan header `Content-Type` (untuk diunggah) dan/atau `Accept` (untuk diunduh) ke permintaan metode untuk meminta klien menentukan jenis media biner yang diperlukan dan memetakannya ke permintaan integrasi.
3. Setel Penanganan Konten ke `Passthrough` dalam permintaan integrasi (untuk diunggah) dan dalam respons integrasi (untuk diunduh). Pastikan tidak ada template pemetaan yang ditentukan untuk jenis konten yang terpengaruh. Untuk informasi selengkapnya, lihat Perilaku Passthrough Integrasi dan Pilih Templat Pemetaan VTL.

Batas ukuran muatan adalah 10 MB. Lihat Kuota API Gateway untuk mengonfigurasi dan menjalankan REST API.

Pastikan file di Amazon S3 memiliki jenis konten yang benar yang ditambahkan sebagai metadata file. Untuk konten media yang dapat disederhanakan, `Content-Disposition:inline` mungkin juga perlu ditambahkan ke metadata.

Untuk informasi selengkapnya tentang dukungan biner di API Gateway, lihatKonversi jenis konten di API Gateway.

Definisi OpenAPI dari API sampel sebagai proxy Amazon S3

Definisi OpenAPI berikut menjelaskan API yang berfungsi sebagai proxy Amazon S3. API ini berisi lebih banyak operasi Amazon S3 daripada API yang Anda buat dalam tutorial. Metode berikut diekspos dalam definisi OpenAPI:

- Paparkan GET di sumber daya root API untuk mencantumkan semua bucket Amazon S3 pemanggil.
- Paparkan GET pada resource Folder untuk melihat daftar semua objek di bucket Amazon S3.
- Paparkan PUT pada sumber daya Folder untuk menambahkan bucket ke Amazon S3.
- Paparkan DELETE pada resource Folder untuk menghapus bucket dari Amazon S3.
- Paparkan GET pada resource Folder/Item untuk melihat atau mengunduh objek dari bucket Amazon S3.

- Paparkan PUT pada resource Folder/Item untuk mengunggah objek ke bucket Amazon S3.
- Paparkan HEAD pada resource Folder/Item untuk mendapatkan metadata objek di bucket Amazon S3.
- Paparkan DELETE pada resource Folder/Item untuk menghapus objek dari bucket Amazon S3.

Untuk petunjuk tentang cara mengimpor API menggunakan definisi OpenAPI, lihat. Mengkonfigurasi REST API menggunakan OpenAPI

Untuk petunjuk tentang cara membuat API serupa, lihatTutorial: Membuat REST API sebagai proxy Amazon S3 di API Gateway.

Untuk mempelajari cara menjalankan API ini menggunakan Postman, yang mendukung otorisasi AWS IAM, lihat. Panggil API menggunakan klien REST API

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2016-10-13T23:04:43Z",
    "title": "MyS3"
  },
  "host": "9gn28ca086.execute-api.{region}.amazonaws.com",
  "basePath": "/S3",
  "schemes": [
    "https"
  ],
  "paths": {
    "/": {
      "get": {
        "produces": [
          "application/json"
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            },
            "headers": {
              "Content-Length": {
```

```
              "type": "string"
            },
            "Timestamp": {
              "type": "string"
            },
            "Content-Type": {
              "type": "string"
            }
          }
        },
        "400": {
          "description": "400 response"
        },
        "500": {
          "description": "500 response"
        }
      },
      "security": [
        {
          "sigv4": []
        }
      ],
      "x-amazon-apigateway-integration": {
        "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
        "responses": {
          "4\\d{2}": {
            "statusCode": "400"
          },
          "default": {
            "statusCode": "200",
            "responseParameters": {
              "method.response.header.Content-Type":
"integration.response.header.Content-Type",
              "method.response.header.Content-Length":
"integration.response.header.Content-Length",
              "method.response.header.Timestamp":
"integration.response.header.Date"
            }
          },
          "5\\d{2}": {
            "statusCode": "500"
          }
        },
        "uri": "arn:aws:apigateway:us-west-2:s3:path//",
```

```
          "passthroughBehavior": "when_no_match",
          "httpMethod": "GET",
          "type": "aws"
        }
      }
    },
    "/{folder}": {
      "get": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "folder",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            },
            "headers": {
              "Content-Length": {
                "type": "string"
              },
              "Date": {
                "type": "string"
              },
              "Content-Type": {
                "type": "string"
              }
            }
          },
          "400": {
            "description": "400 response"
          },
          "500": {
            "description": "500 response"
          }
        },
```

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

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

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

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

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

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

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

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

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

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

```
        },
        "400": {
          "description": "400 response"
        },
        "500": {
          "description": "500 response"
        }
      },
      "security": [
        {
          "sigv4": []
        }
      ],
      "x-amazon-apigateway-integration": {
        "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
        "responses": {
          "4\\d{2}": {
            "statusCode": "400"
          },
          "default": {
            "statusCode": "200"
          },
          "5\\d{2}": {
            "statusCode": "500"
          }
        },
        "requestParameters": {
          "integration.request.path.object": "method.request.path.item",
          "integration.request.path.bucket": "method.request.path.folder"
        },
        "uri": "arn:aws:apigateway:us-west-2:s3:path/{bucket}/{object}",
        "passthroughBehavior": "when_no_match",
        "httpMethod": "DELETE",
        "type": "aws"
      }
    }
  },
  "securityDefinitions": {
    "sigv4": {
      "type": "apiKey",
      "name": "Authorization",
      "in": "header",
      "x-amazon-apigateway-authtype": "awsSigv4"
```

```
    }
  },
  "definitions": {
    "Empty": {
      "type": "object",
      "title": "Empty Schema"
    }
  }
}
```

OpenAPI 3.0

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "MyS3",
    "version" : "2016-10-13T23:04:43Z"
  },
  "servers" : [ {
    "url" : "https://9gn28ca086.execute-api.{region}.amazonaws.com/{basePath}",
    "variables" : {
      "basePath" : {
        "default" : "S3"
      }
    }
  } ],
  "paths" : {
    "/{folder}" : {
      "get" : {
        "parameters" : [ {
          "name" : "folder",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "400" : {
            "description" : "400 response",
            "content" : { }
          },
          "500" : {
```

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

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

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

```
        },
        "passthroughBehavior" : "when_no_match",
        "type" : "aws"
      }
    },
    "delete" : {
      "parameters" : [ {
        "name" : "folder",
        "in" : "path",
        "required" : true,
        "schema" : {
          "type" : "string"
        }
      } ],
      "responses" : {
        "400" : {
          "description" : "400 response",
          "content" : { }
        },
        "500" : {
          "description" : "500 response",
          "content" : { }
        },
        "200" : {
          "description" : "200 response",
          "headers" : {
            "Date" : {
              "schema" : {
                "type" : "string"
              }
            },
            "Content-Type" : {
              "schema" : {
                "type" : "string"
              }
            }
          },
          "content" : {
            "application/json" : {
              "schema" : {
                "$ref" : "#/components/schemas/Empty"
              }
            }
          }
```

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

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

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

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

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

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

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

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

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

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

Panggil API menggunakan klien REST API

Untuk memberikan end-to-end tutorial, kami sekarang menunjukkan cara memanggil API menggunakan Postman, yang mendukung otorisasi AWS IAM.

Untuk memanggil API proxy Amazon S3 kami menggunakan Postman

1. Menerapkan atau menerapkan ulang API. Catat URL dasar API yang ditampilkan di sebelah Invoke URL di bagian atas Stage Editor.
2. Luncurkan Postman.
3. Pilih Otorisasi dan kemudian pilih`AWS Signature`. Ketik ID Kunci Akses dan Kunci Akses Rahasia pengguna IAM Anda ke dalam bidang AccessKeydan SecretKeyinput, masing-masing. Ketik AWS wilayah tempat API Anda digunakan di kotak teks AWSWilayah. `execute-api`Ketik kolom input Nama Layanan.

Anda dapat membuat sepasang kunci dari tab Security Credentials dari akun pengguna IAM Anda di IAM Management Console.

4. Untuk menambahkan bucket yang diberi nama `apig-demo-5` ke akun Amazon S3 Anda di wilayah ini*`{region}`*:

   > **Note**
   >
   > Pastikan nama bucket harus unik secara global.

   a. Pilih PUT dari daftar metode drop-down dan ketik URL metode (`https://`*`api-id`*`.execute-api.`*`aws-region`*`.amazonaws.com/`*`stage`*`/`*`folder-name`*

   b. Tetapkan nilai `Content-Type` header sebagai`application/xml`. Anda mungkin perlu menghapus header yang ada sebelum menyetel jenis konten.

   c. Pilih item menu Body dan ketik fragmen XHTML berikut sebagai badan permintaan:

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

   d. Pilih Kirim untuk mengirimkan permintaan. Jika berhasil, Anda harus menerima `200 OK` respons dengan muatan kosong.

5. Untuk menambahkan file teks ke ember, ikuti petunjuk di atas. Jika Anda menentukan nama bucket **`apig-demo-5`** for `{folder}` dan nama file **`Readme.txt`** for `{item}` di URL dan memberikan string teks **`Hello, World!`** sebagai isi file (sehingga menjadikannya payload permintaan), permintaan menjadi

   ```
   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!
```

Jika semuanya berjalan dengan baik, Anda harus menerima `200 OK` respons dengan muatan kosong.

6. Untuk mendapatkan konten `Readme.txt` file yang baru saja kita tambahkan ke `apig-demo-5` bucket, lakukan permintaan GET seperti berikut ini:

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

Jika berhasil, Anda harus menerima `200 OK` respons dengan string `Hello, World!` teks sebagai payload.

7. Untuk mencantumkan item di `apig-demo-5` bucket, kirimkan permintaan berikut:

```
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
```

Jika berhasil, Anda harus menerima `200 OK` respons dengan payload XHTML yang menampilkan satu item dalam bucket yang ditentukan, kecuali jika Anda menambahkan lebih banyak file ke bucket sebelum mengirimkan permintaan ini.

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Name>apig-demo-5</Name>
    <Prefix></Prefix>
    <Marker></Marker>
```

```
    <MaxKeys>1000</MaxKeys>
    <IsTruncated>false</IsTruncated>
    <Contents>
        <Key>Readme.txt</Key>
        <LastModified>2016-10-15T06:26:48.000Z</LastModified>
        <ETag>"65a8e27d8879283831b664bd8b7f0ad4"</ETag>
        <Size>13</Size>
        <Owner>
            <ID>06e4b09e9d...603addd12ee</ID>
            <DisplayName>user-name</DisplayName>
        </Owner>
        <StorageClass>STANDARD</StorageClass>
    </Contents>
</ListBucketResult>
```

> **Note**
>
> Untuk mengunggah atau mengunduh gambar, Anda perlu mengatur penanganan konten ke CONVERT_TO_BINARY.

Tutorial: Membuat REST API sebagai proxy Amazon Kinesis di API Gateway

Halaman ini menjelaskan cara membuat dan mengonfigurasi REST API dengan integrasi AWS tipe untuk mengakses Kinesis.

> **Note**
>
> Untuk mengintegrasikan API Gateway API Anda dengan Kinesis, Anda harus memilih wilayah di mana layanan API Gateway dan Kinesis tersedia. Untuk ketersediaan wilayah, lihat Titik Akhir Layanan dan Kuota.

Untuk tujuan ilustrasi, kami membuat contoh API untuk memungkinkan klien melakukan hal berikut:

1. Daftar aliran yang tersedia pengguna di Kinesis
2. Membuat, mendeskripsikan, atau menghapus aliran tertentu

3. Membaca catatan data dari atau menulis catatan data ke dalam aliran yang ditentukan

Untuk menyelesaikan tugas-tugas sebelumnya, API mengekspos metode pada berbagai sumber daya untuk menjalankan yang berikut ini, masing-masing:

1. `ListStreams`Tindakan dalam Kinesis
2. Tindakan `CreateStream``DescribeStream`, atau `DeleteStream` tindakan
3. Tindakan `GetRecords` atau `PutRecords` (termasuk`PutRecord`) dalam Kinesis

Secara khusus, kami membangun API sebagai berikut:

- Paparkan metode HTTP GET pada `/streams` sumber daya API dan integrasikan metode tersebut dengan ListStreamstindakan di Kinesis untuk mencantumkan aliran di akun pemanggil.
- Paparkan metode HTTP POST pada `/streams/{stream-name}` sumber daya API dan integrasikan metode tersebut dengan CreateStreamtindakan di Kinesis untuk membuat aliran bernama di akun pemanggil.
- Paparkan metode HTTP GET pada `/streams/{stream-name}` sumber daya API dan integrasikan metode tersebut dengan DescribeStreamtindakan di Kinesis untuk mendeskripsikan aliran bernama di akun pemanggil.
- Paparkan metode HTTP DELETE pada `/streams/{stream-name}` sumber daya API dan integrasikan metode tersebut dengan DeleteStreamtindakan di Kinesis untuk menghapus aliran di akun pemanggil.
- Paparkan metode HTTP PUT pada `/streams/{stream-name}/record` sumber daya API dan integrasikan metode dengan PutRecordtindakan di Kinesis. Ini memungkinkan klien untuk menambahkan catatan data tunggal ke aliran bernama.
- Paparkan metode HTTP PUT pada `/streams/{stream-name}/records` sumber daya API dan integrasikan metode dengan PutRecordstindakan di Kinesis. Ini memungkinkan klien untuk menambahkan daftar catatan data ke aliran bernama.
- Paparkan metode HTTP GET pada `/streams/{stream-name}/records` sumber daya API dan integrasikan metode tersebut dengan GetRecordstindakan di Kinesis. Hal ini memungkinkan klien untuk daftar catatan data dalam aliran bernama, dengan iterator shard tertentu. Sebuah iterator shard menentukan posisi shard dari mana untuk mulai membaca catatan data secara berurutan.
- Paparkan metode HTTP GET pada `/streams/{stream-name}/sharditerator` sumber daya API dan integrasikan metode tersebut dengan GetShardIteratortindakan di Kinesis. Metode pembantu ini harus diberikan pada `ListStreams` tindakan di Kinesis.

Anda dapat menerapkan instruksi yang disajikan di sini untuk tindakan Kinesis lainnya. Untuk daftar lengkap tindakan Kinesis, lihat Referensi API Amazon Kinesis.

Alih-alih menggunakan konsol API Gateway untuk membuat API sampel, Anda dapat mengimpor API sampel ke API Gateway menggunakan API Gateway Import API. Untuk informasi tentang cara menggunakan API Impor, lihatMengkonfigurasi REST API menggunakan OpenAPI.

Membuat peran dan kebijakan IAM untuk API untuk mengakses Kinesis

Untuk mengizinkan API menjalankan tindakan Kinesis, Anda harus memiliki kebijakan IAM yang sesuai yang dilampirkan ke peran IAM.

Untuk membuat peran eksekusi proxy AWS layanan

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.
2. Pilih Peran.
3. Pilih Buat peran.
4. Pilih AWS layanan di bawah Pilih jenis entitas tepercaya, lalu pilih API Gateway dan pilih Izinkan API Gateway untuk mendorong CloudWatch log ke Log.
5. Pilih Berikutnya, lalu pilih Berikutnya.
6. Untuk nama Peran**`APIGatewayKinesisProxyPolicy`**, masukkan, lalu pilih Buat peran.
7. Dalam daftar Peran, pilih peran yang baru saja Anda buat. Anda mungkin perlu menggulir atau menggunakan bilah pencarian untuk menemukan peran.
8. Untuk peran yang dipilih, pilih tab Tambahkan izin.
9. Pilih Lampirkan kebijakan dari daftar dropdown.
10. Di bilah pencarian, masukkan **`AmazonKinesisFullAccess`** dan pilih Tambahkan izin.

> **Note**
>
> Tutorial ini menggunakan kebijakan terkelola untuk kesederhanaan. Sebagai praktik terbaik, Anda harus membuat kebijakan IAM Anda sendiri untuk memberikan izin minimum yang diperlukan.

11. Perhatikan ARN Peran yang baru dibuat, Anda akan menggunakannya nanti.

Mulai membuat API sebagai proxy Kinesis

Gunakan langkah-langkah berikut untuk membuat API di konsol API Gateway.

Untuk membuat API sebagai proxy AWS layanan untuk Kinesis

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Jika ini adalah pertama kalinya Anda menggunakan API Gateway, Anda akan melihat halaman yang memperkenalkan Anda ke fitur layanan. Di bawah REST API, pilih Build. Saat munculan Create Example API muncul, pilih OK.

   Jika ini bukan pertama kalinya Anda menggunakan API Gateway, pilih Buat API. Di bawah REST API, pilih Build.
3. Pilih API Baru.
4. Dalam nama API, masukkan**`KinesisProxy`**. Simpan nilai default untuk semua bidang lainnya.
5. (Opsional) Untuk Deskripsi, masukkan deskripsi.
6. Pilih Buat API.

Setelah API dibuat, konsol API Gateway menampilkan halaman Resources, yang hanya berisi sumber daya root (`/`) API.

Daftar aliran dalam Kinesis

Kinesis mendukung `ListStreams` tindakan dengan panggilan REST API berikut:

```
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>

{
   ...
}
```

Dalam permintaan REST API di atas, tindakan ditentukan dalam parameter `Action` kueri. Atau, Anda dapat menentukan tindakan di `X-Amz-Target` header, sebagai gantinya:

```
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
{
   ...
}
```

Dalam tutorial ini, kita menggunakan parameter query untuk menentukan tindakan.

Untuk mengekspos tindakan Kinesis di API, tambahkan `/streams` resource ke root API. Kemudian atur `GET` metode pada sumber daya dan integrasikan metode dengan `ListStreams` aksi Kinesis.

Prosedur berikut menjelaskan cara membuat daftar aliran Kinesis dengan menggunakan konsol API Gateway.

Untuk membuat daftar aliran Kinesis dengan menggunakan konsol API Gateway

1. Pilih `/` sumber daya, lalu pilih Buat sumber daya.
2. Untuk Nama sumber daya, masukkan **`streams`**.
3. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).
4. Pilih Buat sumber daya.
5. Pilih `/streams` sumber daya, lalu pilih Create method, lalu lakukan hal berikut:

    a. Untuk tipe Metode, pilih GET.

    > **Note**
    >
    > Kata kerja HTTP untuk metode yang dipanggil oleh klien mungkin berbeda dari kata kerja HTTP untuk integrasi yang diperlukan oleh backend. Kami memilih `GET` di sini, karena aliran daftar secara intuitif merupakan operasi BACA.

    b. Untuk jenis Integrasi, pilih AWS layanan.

    c. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat aliran Kinesis Anda.

    d. Untuk Layanan AWS, pilih Kinesis.

e. Biarkan AWS subdomain kosong.

f. Untuk metode HTTP, pilih POST.

> **Note**
>
> Kami memilih `POST` di sini karena Kinesis mengharuskan `ListStreams` tindakan dipanggil dengannya.

g. Untuk tipe Tindakan, pilih Gunakan nama tindakan.

h. Untuk nama Tindakan, masukkan**`ListStreams`**.

i. Untuk peran Eksekusi, masukkan ARN untuk peran eksekusi Anda.

j. Pertahankan default Passthrough untuk Penanganan Konten.

k. Pilih metode Buat.

6. Pada tab Permintaan integrasi, di bawah Pengaturan permintaan integrasi, pilih Edit.
7. Untuk Request body passthrough, pilih Bila tidak ada templat yang ditentukan (disarankan).
8. Pilih parameter header permintaan URL, lalu lakukan hal berikut:

   a. Pilih Tambahkan parameter header permintaan.

   b. Untuk Nama, masukkan **`Content-Type`**.

   c. Untuk Dipetakan dari, masukkan**`'application/x-amz-json-1.1'`**.

   Kami menggunakan pemetaan parameter permintaan untuk mengatur `Content-Type` header ke nilai statis `'application/x-amz-json-1.1'` untuk menginformasikan Kinesis bahwa input adalah versi tertentu dari JSON.
9. Pilih Templat pemetaan, lalu pilih Tambahkan templat pemetaan, dan lakukan hal berikut:

   a. Untuk Content-Type, masukkan. **`application/json`**

   b. Untuk badan Template, masukkan**`{}`**.

   c. Pilih Simpan.

   ListStreamsPermintaan mengambil muatan dari format JSON berikut:

```
{
    "ExclusiveStartStreamName": "string",
```

```
    "Limit": number
}
```

Namun, propertinya opsional. Untuk menggunakan nilai default, kami memilih payload JSON kosong di sini.

10. Uji metode GET pada sumber daya /streams untuk menjalankan tindakan `ListStreams` di Kinesis:

    Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.

    Pilih Tes untuk menguji metode Anda.

    Jika Anda telah membuat dua aliran bernama “MyStream” dan “YourStream” di Kinesis, pengujian yang berhasil mengembalikan respons 200 OK yang berisi muatan berikut:

```
{
     "HasMoreStreams": false,
     "StreamNames": [
         "myStream",
         "yourStream"
     ]
}
```

Buat, jelaskan, dan hapus aliran di Kinesis

Membuat, mendeskripsikan, dan menghapus aliran di Kinesis melibatkan pembuatan permintaan Kinesis REST API berikut, masing-masing:

```
POST /?Action=CreateStream HTTP/1.1
Host: kinesis.region.domain
...
Content-Type: application/x-amz-json-1.1
Content-Length: PayloadSizeBytes

{
    "ShardCount": number,
```

```
    "StreamName": "string"
}
```

```
POST /?Action=DescribeStream HTTP/1.1
Host: kinesis.region.domain
...
Content-Type: application/x-amz-json-1.1
Content-Length: PayloadSizeBytes

{
    "StreamName": "string"
}
```

```
POST /?Action=DeleteStream HTTP/1.1
Host: kinesis.region.domain
...
Content-Type: application/x-amz-json-1.1
Content-Length: PayloadSizeBytes

{
    "StreamName":"string"
}
```

Kita dapat membangun API untuk menerima input yang diperlukan sebagai payload JSON dari permintaan metode dan meneruskan payload ke permintaan integrasi. Namun, untuk memberikan lebih banyak contoh pemetaan data antara permintaan metode dan integrasi, serta respons metode dan integrasi, kami membuat API kami agak berbeda.

Kami mengeksposGET,POST, dan metode `Delete` HTTP pada to-be-named `Stream` sumber daya. Kami menggunakan variabel `{stream-name}` jalur sebagai pengganti sumber daya aliran dan mengintegrasikan metode API ini dengan Kinesis,, dan tindakan `DescribeStream``CreateStream`, masing-masing. `DeleteStream` Kami mengharuskan klien meneruskan data input lain sebagai header, parameter kueri, atau muatan permintaan metode. Kami menyediakan template pemetaan untuk mengubah data ke payload permintaan integrasi yang diperlukan.

Untuk membuat sumber daya {stream-name}

1. Pilih sumber daya /streams, lalu pilih Buat sumber daya.
2. Matikan sumber daya Proxy.
3. Untuk jalur Sumber Daya, pilih`/streams`.
4. Untuk Nama sumber daya, masukkan **`{stream-name}`**.
5. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).
6. Pilih Buat sumber daya.

Untuk mengonfigurasi dan menguji metode GET pada sumber daya aliran

1. Pilih sumber daya/{stream-name}, lalu pilih Create method.
2. Untuk tipe Metode, pilih GET.
3. Untuk jenis Integrasi, pilih AWS layanan.
4. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat aliran Kinesis Anda.
5. Untuk Layanan AWS, pilih Kinesis.
6. Biarkan AWS subdomain kosong.
7. Untuk metode HTTP, pilih POST.
8. Untuk tipe Tindakan, pilih Gunakan nama tindakan.
9. Untuk nama Tindakan, masukkan**`DescribeStream`**.
10. Untuk peran Eksekusi, masukkan ARN untuk peran eksekusi Anda.
11. Pertahankan default Passthrough untuk Penanganan Konten.
12. Pilih metode Buat.
13. Di bagian Permintaan integrasi, tambahkan parameter header permintaan URL berikut:

    ```
    Content-Type: 'x-amz-json-1.1'
    ```

    Tugas mengikuti prosedur yang sama untuk mengatur pemetaan parameter permintaan untuk `GET /streams` metode tersebut.
14. Tambahkan template pemetaan badan berikut untuk memetakan data dari permintaan `GET /streams/{stream-name}` metode ke permintaan `POST /?Action=DescribeStream` integrasi:

```
{
    "StreamName": "$input.params('stream-name')"
}
```

Template pemetaan ini menghasilkan payload permintaan integrasi yang diperlukan untuk `DescribeStream` tindakan Kinesis dari nilai parameter jalur permintaan `stream-name` metode.

15. Untuk menguji `GET /stream/{stream-name}` metode untuk menjalankan `DescribeStream` tindakan di Kinesis, pilih tab Uji.
16. Untuk Path, di bawah nama aliran, masukkan nama aliran Kinesis yang ada.
17. Pilih Uji. Jika tes berhasil, respons 200 OK dikembalikan dengan muatan yang mirip dengan berikut ini:

```
{
  "StreamDescription": {
    "HasMoreShards": false,
    "RetentionPeriodHours": 24,
    "Shards": [
      {
        "HashKeyRange": {
          "EndingHashKey": "680564733841876926926749214863536422 90",
          "StartingHashKey": "0"
        },
        "SequenceNumberRange": {
          "StartingSequenceNumber":
 "49559266461454070523309915164834022007924120923395850242"
        },
        "ShardId": "shardId-000000000000"
      },
      ...
      {
        "HashKeyRange": {
          "EndingHashKey": "340282366920938463463374607431768211455",
          "StartingHashKey": "272225893536750770770699685945414569164"
        },
        "SequenceNumberRange": {
          "StartingSequenceNumber":
 "49559266461543273504104037657400164881014714369419771970"
        },
        "ShardId": "shardId-000000000004"
      }
```

```
    ],
    "StreamARN": "arn:aws:kinesis:us-east-1:12345678901:stream/myStream",
    "StreamName": "myStream",
    "StreamStatus": "ACTIVE"
  }
}
```

Setelah menerapkan API, Anda dapat membuat permintaan REST terhadap metode API ini:

```
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
```

Untuk mengkonfigurasi dan menguji metode POST pada sumber daya aliran

1. Pilih sumber daya/{stream-name}, lalu pilih Create method.
2. Untuk jenis Metode, pilih POST.
3. Untuk jenis Integrasi, pilih AWS layanan.
4. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat aliran Kinesis Anda.
5. Untuk Layanan AWS, pilih Kinesis.
6. Biarkan AWS subdomain kosong.
7. Untuk metode HTTP, pilih POST.
8. Untuk tipe Tindakan, pilih Gunakan nama tindakan.
9. Untuk nama Tindakan, masukkan**`CreateStream`**.
10. Untuk peran Eksekusi, masukkan ARN untuk peran eksekusi Anda.
11. Pertahankan default Passthrough untuk Penanganan Konten.
12. Pilih metode Buat.
13. Di bagian Permintaan integrasi, tambahkan parameter header permintaan URL berikut:

```
Content-Type: 'x-amz-json-1.1'
```

Tugas mengikuti prosedur yang sama untuk mengatur pemetaan parameter permintaan untuk `GET /streams` metode tersebut.

14. Tambahkan template pemetaan badan berikut untuk memetakan data dari permintaan `POST /streams/{stream-name}` metode ke permintaan `POST /?Action=CreateStream` integrasi:

```
{
    "ShardCount": #if($input.path('$.ShardCount') == '') 5 #else
 $input.path('$.ShardCount') #end,
    "StreamName": "$input.params('stream-name')"
}
```

Dalam template pemetaan sebelumnya, kami menetapkan `ShardCount` ke nilai tetap 5 jika klien tidak menentukan nilai dalam payload permintaan metode.

15. Untuk menguji `POST /stream/{stream-name}` metode untuk menjalankan `CreateStream` tindakan di Kinesis, pilih tab Uji.
16. Untuk Path, di bawah nama aliran, masukkan nama aliran Kinesis baru.
17. Pilih Uji. Jika tes berhasil, respons 200 OK dikembalikan tanpa data.

Setelah menerapkan API, Anda juga dapat membuat permintaan REST API terhadap metode POST pada sumber daya Stream untuk menjalankan `CreateStream` tindakan di 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
}
```

Konfigurasikan dan uji metode DELETE pada sumber daya aliran

1. Pilih sumber daya/{stream-name}, lalu pilih Create method.

2. Untuk jenis Metode, pilih DELETE.
3. Untuk jenis Integrasi, pilih AWS layanan.
4. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat aliran Kinesis Anda.
5. Untuk Layanan AWS, pilih Kinesis.
6. Biarkan AWS subdomain kosong.
7. Untuk metode HTTP, pilih POST.
8. Untuk tipe Tindakan, pilih Gunakan nama tindakan.
9. Untuk nama Tindakan, masukkan**`DeleteStream`**.
10. Untuk peran Eksekusi, masukkan ARN untuk peran eksekusi Anda.
11. Pertahankan default Passthrough untuk Penanganan Konten.
12. Pilih metode Buat.
13. Di bagian Permintaan integrasi, tambahkan parameter header permintaan URL berikut:

    ```
    Content-Type: 'x-amz-json-1.1'
    ```

    Tugas mengikuti prosedur yang sama untuk mengatur pemetaan parameter permintaan untuk `GET /streams` metode tersebut.
14. Tambahkan template pemetaan badan berikut untuk memetakan data dari permintaan `DELETE /streams/{stream-name}` metode ke permintaan integrasi yang sesuai dari`POST /?Action=DeleteStream`:

    ```
    {
        "StreamName": "$input.params('stream-name')"
    }
    ```

    Template pemetaan ini menghasilkan input yang diperlukan untuk `DELETE /streams/{stream-name}` tindakan dari nama jalur URL yang disediakan klien. `stream-name`
15. Untuk menguji `DELETE /stream/{stream-name}` metode untuk menjalankan `DeleteStream` tindakan di Kinesis, pilih tab Uji.
16. Untuk Path, di bawah nama aliran, masukkan nama aliran Kinesis yang ada.
17. Pilih Uji. Jika tes berhasil, respons 200 OK dikembalikan tanpa data.

Setelah menerapkan API, Anda juga dapat membuat permintaan REST API berikut terhadap metode DELETE pada sumber daya Stream untuk memanggil `DeleteStream` tindakan di 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

{}
```

Dapatkan catatan dari dan tambahkan catatan ke aliran di Kinesis

Setelah Anda membuat aliran di Kinesis, Anda dapat menambahkan catatan data ke aliran dan membaca data dari aliran. Menambahkan catatan data melibatkan pemanggilan PutRecordsatau PutRecordtindakan dalam Kinesis. Yang pertama menambahkan beberapa catatan sedangkan yang terakhir menambahkan satu catatan ke aliran.

```
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"
}
```

atau

```
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"
}
```

Di sini, `StreamName` mengidentifikasi aliran target untuk menambahkan catatan. `StreamName`,`Data`, dan `PartitionKey` diperlukan data input. Dalam contoh kita, kita menggunakan nilai default untuk semua data input opsional dan tidak akan secara eksplisit menentukan nilai untuk mereka dalam input ke permintaan metode.

Membaca data dalam Kinesis sama dengan memanggil tindakan: 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
}
```

Di sini, aliran sumber dari mana kita mendapatkan catatan ditentukan dalam `ShardIterator` nilai yang diperlukan, seperti yang ditunjukkan dalam tindakan Kinesis berikut untuk mendapatkan iterator pecahan:

```
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"
}
```

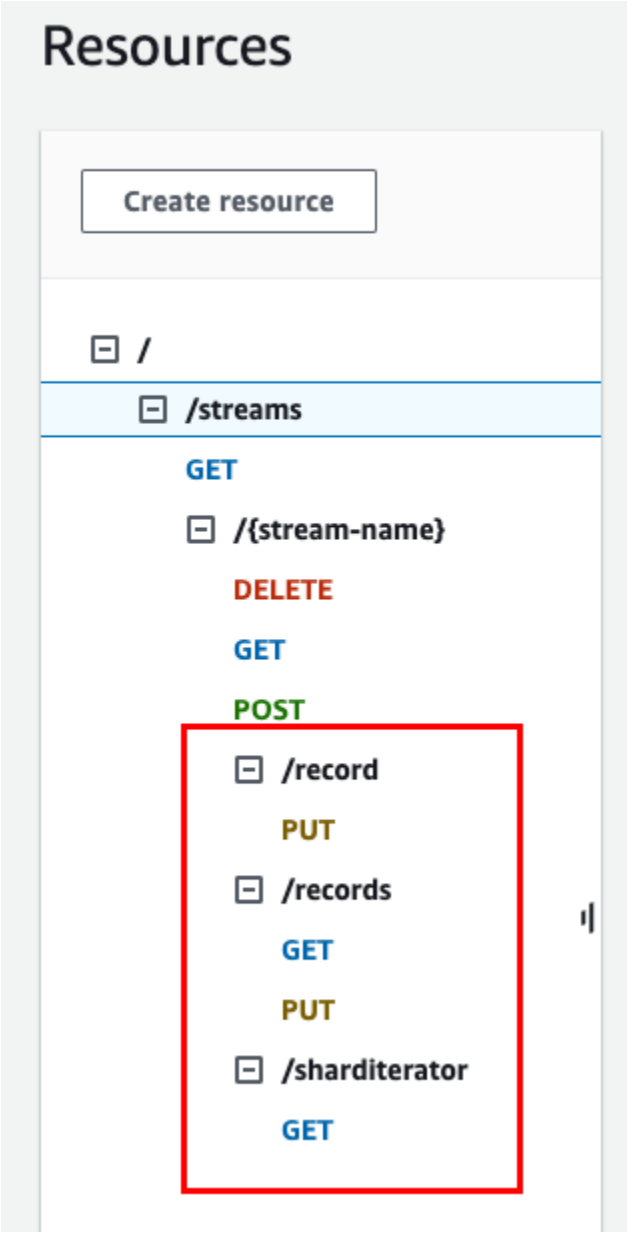
Untuk `PutRecords` tindakan `GetRecords` dan, kami mengekspos PUT metode `GET` dan, masing-masing, pada `/records` sumber daya yang ditambahkan ke sumber daya aliran bernama ()/ `{stream-name}`. Demikian pula, kami mengekspos `PutRecord` tindakan sebagai PUT metode pada `/record` sumber daya.

Karena `GetRecords` tindakan mengambil sebagai input `ShardIterator` nilai, yang diperoleh dengan memanggil aksi `GetShardIterator` helper, kami mengekspos metode `GET` helper pada `ShardIterator` resource (). `/sharditerator`

Untuk membuat sumber daya /record, /records, dan /sharditerator

1. Pilih sumber daya/{stream-name}, lalu pilih Buat sumber daya.
2. Matikan sumber daya Proxy.
3. Untuk jalur Sumber Daya, pilih`/{stream-name}`.
4. Untuk Nama sumber daya, masukkan **`record`**.
5. Tetap nonaktifkan CORS (Cross Origin Resource Sharing).
6. Pilih Buat sumber daya.

7. Ulangi langkah sebelumnya untuk membuat /records dan sumber daya /sharditerator. API final akan terlihat seperti berikut:



Empat prosedur berikut menjelaskan cara mengatur masing-masing metode, cara memetakan data dari permintaan metode ke permintaan integrasi, dan cara menguji metode.

Untuk mengatur dan menguji **`PUT /streams/{stream-name}/record`** metode yang akan dipanggil **`PutRecord`** di Kinesis:

1. Pilih /record, lalu pilih Create method.

2. Untuk jenis Metode, pilih PUT.
3. Untuk jenis Integrasi, pilih AWS layanan.
4. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat aliran Kinesis Anda.
5. Untuk Layanan AWS, pilih Kinesis.
6. Biarkan AWS subdomain kosong.
7. Untuk metode HTTP, pilih POST.
8. Untuk tipe Tindakan, pilih Gunakan nama tindakan.
9. Untuk nama Tindakan, masukkan**`PutRecord`**.
10. Untuk peran Eksekusi, masukkan ARN untuk peran eksekusi Anda.
11. Pertahankan default Passthrough untuk Penanganan Konten.
12. Pilih metode Buat.
13. Di bagian Permintaan integrasi, tambahkan parameter header permintaan URL berikut:

    ```
    Content-Type: 'x-amz-json-1.1'
    ```

    Tugas mengikuti prosedur yang sama untuk mengatur pemetaan parameter permintaan untuk `GET /streams` metode tersebut.
14. Tambahkan template pemetaan badan berikut untuk memetakan data dari permintaan `PUT /streams/{stream-name}/record` metode ke permintaan integrasi yang sesuai dari`POST /?Action=PutRecord`:

    ```
    {
        "StreamName": "$input.params('stream-name')",
        "Data": "$util.base64Encode($input.json('$.Data'))",
        "PartitionKey": "$input.path('$.PartitionKey')"
    }
    ```

    Template pemetaan ini mengasumsikan bahwa payload permintaan metode adalah dari format berikut:

    ```
    {
       "Data": "some data",
       "PartitionKey": "some key"
    }
    ```

Data ini dapat dimodelkan dengan skema JSON berikut:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "PutRecord proxy single-record payload",
  "type": "object",
  "properties": {
      "Data": { "type": "string" },
      "PartitionKey": { "type": "string" }
  }
}
```

Anda dapat membuat model untuk menyertakan skema ini dan menggunakan model untuk memfasilitasi pembuatan template pemetaan. Namun, Anda dapat membuat template pemetaan tanpa menggunakan model apa pun.

15. Untuk menguji `PUT /streams/{stream-name}/record` metode, atur variabel `stream-name` jalur ke nama aliran yang ada, berikan muatan format yang diperlukan, lalu kirimkan permintaan metode. Hasil yang berhasil adalah `200 OK` respons dengan muatan format berikut:

```
{
  "SequenceNumber": "49559409944537880850133345460169886593573102115167928386",
  "ShardId": "shardId-000000000004"
}
```

Untuk mengatur dan menguji **`PUT /streams/{stream-name}/records`** metode yang akan dipanggil **`PutRecords`** di Kinesis

1. Pilih sumber daya /records, lalu pilih Create method.
2. Untuk jenis Metode, pilih PUT.
3. Untuk jenis Integrasi, pilih AWS layanan.
4. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat aliran Kinesis Anda.
5. Untuk Layanan AWS, pilih Kinesis.
6. Biarkan AWS subdomain kosong.

7. Untuk metode HTTP, pilih POST.
8. Untuk tipe Tindakan, pilih Gunakan nama tindakan.
9. Untuk nama Tindakan, masukkan**`PutRecords`**.
10. Untuk peran Eksekusi, masukkan ARN untuk peran eksekusi Anda.
11. Pertahankan default Passthrough untuk Penanganan Konten.
12. Pilih metode Buat.
13. Di bagian Permintaan integrasi, tambahkan parameter header permintaan URL berikut:

```
Content-Type: 'x-amz-json-1.1'
```

Tugas mengikuti prosedur yang sama untuk mengatur pemetaan parameter permintaan untuk `GET /streams` metode tersebut.

14. Tambahkan template pemetaan berikut untuk memetakan data dari permintaan `PUT /streams/{stream-name}/records` metode ke permintaan integrasi yang sesuai dari`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
    ]
}
```

Template pemetaan ini mengasumsikan bahwa payload permintaan metode dapat dimodelkan oleh skema JSON berikut:

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

Anda dapat membuat model untuk menyertakan skema ini dan menggunakan model untuk memfasilitasi pembuatan template pemetaan. Namun, Anda dapat membuat template pemetaan tanpa menggunakan model apa pun.

Dalam tutorial ini, kami menggunakan dua format payload yang sedikit berbeda untuk menggambarkan bahwa pengembang API dapat memilih untuk mengekspos format data backend ke klien atau menyembunyikannya dari klien. Satu format adalah untuk PUT `/streams/{stream-name}/records` metode (di atas). Format lain digunakan untuk PUT `/streams/{stream-name}/record` metode (dalam prosedur sebelumnya). Dalam lingkungan produksi, Anda harus menjaga kedua format tetap konsisten.

15. Untuk menguji PUT `/streams/{stream-name}/records` metode, atur variabel `stream-name` jalur ke aliran yang ada, berikan payload berikut, dan kirimkan permintaan metode.

```
{
    "records": [
        {
            "data": "some data",
            "partition-key": "some key"
        },
        {
            "data": "some other data",
            "partition-key": "some key"
        }
    ]
}
```

Hasil yang berhasil adalah respons 200 OK dengan muatan yang mirip dengan output berikut:

```
{
  "FailedRecordCount": 0,
  "Records": [
    {
      "SequenceNumber": "49559409944537880850133345460167468741933742152373764162",
      "ShardId": "shardId-000000000004"
    },
    {
      "SequenceNumber": "49559409944537880850133345460168677667753356781548470338",
      "ShardId": "shardId-000000000004"
    }
  ]
}
```

Untuk mengatur dan menguji **`GET /streams/{stream-name}/sharditerator`** metode yang dipanggil **`GetShardIterator`** di Kinesis

`GET /streams/{stream-name}/sharditerator`Metode ini adalah metode pembantu untuk memperoleh iterator shard yang diperlukan sebelum memanggil metode. `GET /streams/{stream-name}/records`

1. Pilih sumber daya /sharditerator, lalu pilih Create method.
2. Untuk tipe Metode, pilih GET.
3. Untuk jenis Integrasi, pilih AWS layanan.
4. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat aliran Kinesis Anda.
5. Untuk Layanan AWS, pilih Kinesis.
6. Biarkan AWS subdomain kosong.
7. Untuk metode HTTP, pilih POST.
8. Untuk tipe Tindakan, pilih Gunakan nama tindakan.
9. Untuk nama Tindakan, masukkan**`GetShardIterator`**.
10. Untuk peran Eksekusi, masukkan ARN untuk peran eksekusi Anda.
11. Pertahankan default Passthrough untuk Penanganan Konten.
12. Pilih metode Buat.

`GetShardIterator`Tindakan tersebut membutuhkan masukan dari suatu ShardId nilai. Untuk meneruskan `ShardId` nilai yang disediakan klien, kami menambahkan parameter `shard-id` kueri ke permintaan metode, seperti yang ditunjukkan pada langkah berikut.

13. Pada tab Permintaan metode, untuk pengaturan permintaan Metode, pilih Edit.
14. Pilih parameter string kueri URL, lalu lakukan hal berikut:

    a. Pilih Tambahkan string kueri.

    b. Untuk Nama, masukkan **`shard-id`**.

    c. Tetap Diperlukan dan Caching dimatikan.

    d. Pilih Simpan.

    Dalam template pemetaan berikut, kita menetapkan nilai parameter `shard-id` query ke nilai `ShardId` properti dari payload JSON sebagai masukan untuk tindakan `GetShardIterator` di Kinesis.

15. Di bagian Permintaan integrasi, tambahkan template pemetaan berikut untuk menghasilkan input yang diperlukan (`ShardIddanStreamName`) ke `GetShardIterator` tindakan dari `shard-id` dan `stream-name` parameter permintaan metode. Selain itu, template pemetaan juga ditetapkan `ShardIteratorType` `TRIM_HORIZON` sebagai default.

    ```
    {
        "ShardId": "$input.params('shard-id')",
        "ShardIteratorType": "TRIM_HORIZON",
        "StreamName": "$input.params('stream-name')"
    }
    ```

16. Menggunakan opsi Uji di konsol API Gateway, masukkan nama aliran yang ada sebagai nilai variabel `stream-name` Path, setel string **`shard-id`** Kueri ke `ShardId` nilai yang ada (mis.,`shard-000000000004`), dan pilih Uji.

    Muatan respons yang berhasil mirip dengan output berikut:

    ```
    {
      "ShardIterator": "AAAAAAAAAAFYVN3VlFy..."
    }
    ```

    Catat `ShardIterator` nilainya. Anda membutuhkannya untuk mendapatkan catatan dari aliran.

Untuk mengkonfigurasi dan menguji **`GET /streams/{stream-name}/records`** metode untuk menjalankan **`GetRecords`** tindakan di Kinesis

1. Pilih sumber daya /records, lalu pilih Create method.
2. Untuk tipe Metode, pilih GET.
3. Untuk jenis Integrasi, pilih AWS layanan.
4. Untuk Wilayah AWS, pilih Wilayah AWS tempat Anda membuat aliran Kinesis Anda.
5. Untuk Layanan AWS, pilih Kinesis.
6. Biarkan AWS subdomain kosong.
7. Untuk metode HTTP, pilih POST.
8. Untuk tipe Tindakan, pilih Gunakan nama tindakan.
9. Untuk nama Tindakan, masukkan**`GetRecords`**.
10. Untuk peran Eksekusi, masukkan ARN untuk peran eksekusi Anda.
11. Pertahankan default Passthrough untuk Penanganan Konten.
12. Pilih metode Buat.

    `GetRecords`Tindakan tersebut membutuhkan masukan dari suatu `ShardIterator` nilai. Untuk meneruskan `ShardIterator` nilai yang disediakan klien, kami menambahkan parameter `Shard-Iterator` header ke permintaan metode, seperti yang ditunjukkan pada langkah-langkah berikut.
13. Pada tab Permintaan metode, untuk pengaturan permintaan Metode, pilih Edit.
14. Pilih header permintaan HTTP, lalu lakukan hal berikut:

    a. Pilih Tambahkan header.

    b. Untuk Nama, masukkan **`Shard-Iterator`**.

    c. Tetap Diperlukan dan Caching dimatikan.

    d. Pilih Simpan.
15. Di bagian Permintaan integrasi, tambahkan template pemetaan badan berikut untuk memetakan nilai parameter `Shard-Iterator` header ke nilai `ShardIterator` properti payload JSON untuk tindakan `GetRecords` di Kinesis.

```
{
    "ShardIterator": "$input.params('Shard-Iterator')"
}
```

16. Menggunakan opsi Uji di konsol API Gateway, masukkan nama aliran yang ada sebagai nilai variabel `stream-name` Path, setel `Shard-Iterator` Header ke `ShardIterator` nilai yang diperoleh dari uji coba `GET /streams/{stream-name}/sharditerator` metode (di atas), dan pilih Uji.

    Muatan respons yang berhasil mirip dengan output berikut:

    ```
    {
      "MillisBehindLatest": 0,
      "NextShardIterator": "AAAAAAAAAAF...",
      "Records": [ ... ]
    }
    ```

Definisi OpenAPI dari API sampel sebagai proxy Kinesis

Berikut ini adalah definisi OpenAPI untuk API sampel sebagai proxy Kinesis yang digunakan dalam tutorial ini.

OpenAPI 3.0

```
{
  "openapi": "3.0.0",
  "info": {
    "title": "KinesisProxy",
    "version": "2016-03-31T18:25:32Z"
  },
  "paths": {
    "/streams/{stream-name}/sharditerator": {
      "get": {
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          },
          {
            "name": "shard-id",
            "in": "query",
```

```
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/GetShardIterator",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"ShardId\": \"$input.params('shard-
id')\",\n    \"ShardIteratorType\": \"TRIM_HORIZON\",\n    \"StreamName\":
 \"$input.params('stream-name')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    },
    "/streams/{stream-name}/records": {
      "get": {
        "parameters": [
          {
```

```
            "name": "stream-name",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          },
          {
            "name": "Shard-Iterator",
            "in": "header",
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/GetRecords",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"ShardIterator\": \"$input.params('Shard-
Iterator')\"\n}"
          },
```

```
        "passthroughBehavior": "when_no_match",
        "httpMethod": "POST"
      }
    },
    "put": {
      "parameters": [
        {
          "name": "Content-Type",
          "in": "header",
          "schema": {
            "type": "string"
          }
        },
        {
          "name": "stream-name",
          "in": "path",
          "required": true,
          "schema": {
            "type": "string"
          }
        }
      ],
      "requestBody": {
        "content": {
          "application/json": {
            "schema": {
              "$ref": "#/components/schemas/PutRecordsMethodRequestPayload"
            }
          },
          "application/x-amz-json-1.1": {
            "schema": {
              "$ref": "#/components/schemas/PutRecordsMethodRequestPayload"
            }
          }
        },
        "required": true
      },
      "responses": {
        "200": {
          "description": "200 response",
          "content": {
            "application/json": {
              "schema": {
                "$ref": "#/components/schemas/Empty"
```

```
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/PutRecords",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\",\n    \"Records\": [\n          {\n            \"Data\":
 \"$util.base64Encode($elem.data)\",\n           \"PartitionKey\":
 \"$elem.partition-key\"\n          }#if($foreach.hasNext),#end\n    ]\n}",
            "application/x-amz-json-1.1": "{\n  \"StreamName\":
 \"$input.params('stream-name')\",\n  \"records\" : [\n    {\n        \"Data
\" : \"$elem.data\",\n        \"PartitionKey\" : \"$elem.partition-key\"\n
  }#if($foreach.hasNext),#end\n  ]\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    },
    "/streams/{stream-name}": {
      "get": {
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          }
```

```
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/DescribeStream",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      },
      "post": {
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
```

```
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/CreateStream",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"ShardCount\": 5,\n    \"StreamName\":
 \"$input.params('stream-name')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      },
      "delete": {
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
```

```
            "headers": {
              "Content-Type": {
                "schema": {
                  "type": "string"
                }
              }
            },
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          },
          "400": {
            "description": "400 response",
            "headers": {
              "Content-Type": {
                "schema": {
                  "type": "string"
                }
              }
            },
            "content": {}
          },
          "500": {
            "description": "500 response",
            "headers": {
              "Content-Type": {
                "schema": {
                  "type": "string"
                }
              }
            },
            "content": {}
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/DeleteStream",
          "responses": {
            "4\\d{2}": {
```

```
              "statusCode": "400",
              "responseParameters": {
                "method.response.header.Content-Type":
 "integration.response.header.Content-Type"
              }
            },
            "default": {
              "statusCode": "200",
              "responseParameters": {
                "method.response.header.Content-Type":
 "integration.response.header.Content-Type"
              }
            },
            "5\\d{2}": {
              "statusCode": "500",
              "responseParameters": {
                "method.response.header.Content-Type":
 "integration.response.header.Content-Type"
              }
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    },
    "/streams/{stream-name}/record": {
      "put": {
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
```

```
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/PutRecord",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\",\n    \"Data\": \"$util.base64Encode($input.json('$.Data'))\",\n
 \"PartitionKey\": \"$input.path('$.PartitionKey')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    },
    "/streams": {
      "get": {
        "responses": {
          "200": {
            "description": "200 response",
            "content": {
              "application/json": {
```

```
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/ListStreams",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    }
  },
  "components": {
    "schemas": {
      "Empty": {
        "type": "object"
      },
      "PutRecordsMethodRequestPayload": {
        "type": "object",
        "properties": {
          "records": {
            "type": "array",
            "items": {
              "type": "object",
              "properties": {
                "data": {
                  "type": "string"
```

```
                },
                "partition-key": {
                  "type": "string"
                }
              }
            }
          }
        }
      }
    }
  }
}
```

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2016-03-31T18:25:32Z",
    "title": "KinesisProxy"
  },
  "basePath": "/test",
  "schemes": [
    "https"
  ],
  "paths": {
    "/streams": {
      "get": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        },
        "x-amazon-apigateway-integration": {
```

```
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/ListStreams",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    },
    "/streams/{stream-name}": {
      "get": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        },
```

```
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/DescribeStream",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      },
      "post": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/CreateStream",
```

```
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"ShardCount\": 5,\n    \"StreamName\":
 \"$input.params('stream-name')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      },
      "delete": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            },
            "headers": {
              "Content-Type": {
                "type": "string"
              }
            }
          },
```

```
          "400": {
            "description": "400 response",
            "headers": {
              "Content-Type": {
                "type": "string"
              }
            }
          },
          "500": {
            "description": "500 response",
            "headers": {
              "Content-Type": {
                "type": "string"
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/DeleteStream",
          "responses": {
            "4\\d{2}": {
              "statusCode": "400",
              "responseParameters": {
                "method.response.header.Content-Type":
"integration.response.header.Content-Type"
              }
            },
            "default": {
              "statusCode": "200",
              "responseParameters": {
                "method.response.header.Content-Type":
"integration.response.header.Content-Type"
              }
            },
            "5\\d{2}": {
              "statusCode": "500",
              "responseParameters": {
                "method.response.header.Content-Type":
"integration.response.header.Content-Type"
              }
            }
          },
```

```
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    },
    "/streams/{stream-name}/record": {
      "put": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/PutRecord",
          "responses": {
            "default": {
              "statusCode": "200"
```

```
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\",\n    \"Data\": \"$util.base64Encode($input.json('$.Data'))\",\n
 \"PartitionKey\": \"$input.path('$.PartitionKey')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    },
    "/streams/{stream-name}/records": {
      "get": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "type": "string"
          },
          {
            "name": "Shard-Iterator",
            "in": "header",
            "required": false,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
```

```
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/GetRecords",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"ShardIterator\": \"$input.params('Shard-
Iterator')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      },
      "put": {
        "consumes": [
          "application/json",
          "application/x-amz-json-1.1"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "Content-Type",
            "in": "header",
            "required": false,
            "type": "string"
          },
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "type": "string"
          },
```

```
          {
            "in": "body",
            "name": "PutRecordsMethodRequestPayload",
            "required": true,
            "schema": {
              "$ref": "#/definitions/PutRecordsMethodRequestPayload"
            }
          },
          {
            "in": "body",
            "name": "PutRecordsMethodRequestPayload",
            "required": true,
            "schema": {
              "$ref": "#/definitions/PutRecordsMethodRequestPayload"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/PutRecords",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\",\n    \"Records\": [\n          {\n            \"Data\":
 \"$util.base64Encode($elem.data)\",\n            \"PartitionKey\":
 \"$elem.partition-key\"\n          }#if($foreach.hasNext),#end\n    ]\n}",
```

```
            "application/x-amz-json-1.1": "{\n  \"StreamName\":
 \"$input.params('stream-name')\",\n  \"records\" : [\n    {\n        \"Data
\" : \"$elem.data\",\n        \"PartitionKey\" : \"$elem.partition-key\"\n
  }#if($foreach.hasNext),#end\n  ]\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    },
    "/streams/{stream-name}/sharditerator": {
      "get": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "type": "string"
          },
          {
            "name": "shard-id",
            "in": "query",
            "required": false,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/GetShardIterator",
```

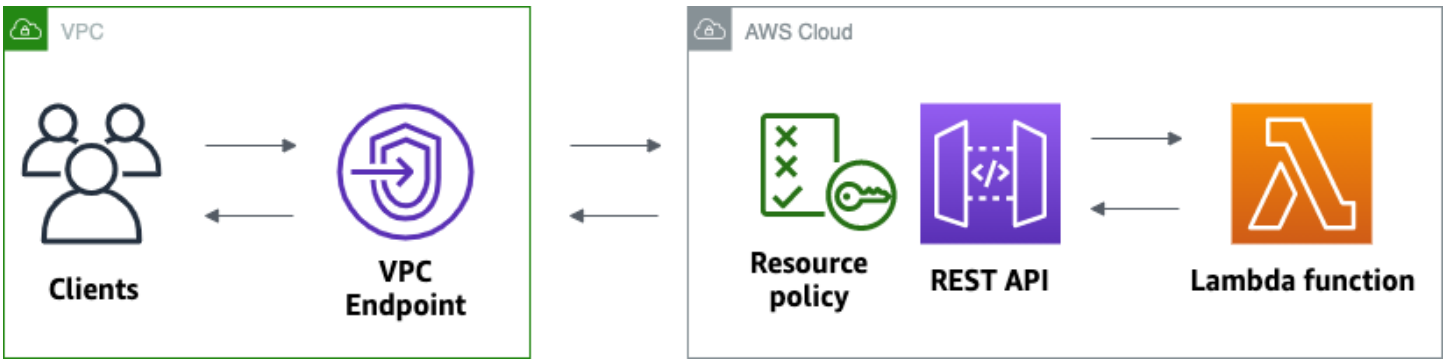
```
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"ShardId\": \"$input.params('shard-
id')\",\n    \"ShardIteratorType\": \"TRIM_HORIZON\",\n    \"StreamName\":
 \"$input.params('stream-name')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    }
  },
  "definitions": {
    "Empty": {
      "type": "object"
    },
    "PutRecordsMethodRequestPayload": {
      "type": "object",
      "properties": {
        "records": {
          "type": "array",
          "items": {
            "type": "object",
            "properties": {
              "data": {
                "type": "string"
              },
              "partition-key": {
                "type": "string"
              }
            }
          }
        }
      }
    }
  }
```

```
}
```

Tutorial: Membangun REST API pribadi

Dalam tutorial ini, Anda membuat REST API pribadi. Klien dapat mengakses API hanya dari dalam VPC Amazon Anda. API diisolasi dari internet publik, yang merupakan persyaratan keamanan umum.

Tutorial ini memakan waktu sekitar 30 menit untuk menyelesaikannya. Pertama, Anda menggunakan AWS CloudFormation template untuk membuat VPC Amazon, titik akhir VPC, AWS Lambda fungsi, dan meluncurkan instans Amazon EC2 yang akan Anda gunakan untuk menguji API Anda. Selanjutnya, Anda menggunakan AWS Management Console untuk membuat API pribadi dan melampirkan kebijakan sumber daya yang memungkinkan akses hanya dari titik akhir VPC Anda. Terakhir, Anda menguji API Anda.



Untuk menyelesaikan tutorial ini, Anda memerlukan AWS akun dan AWS Identity and Access Management pengguna dengan akses konsol. Untuk informasi selengkapnya, lihat Prasyarat.

Dalam tutorial ini, Anda menggunakan AWS Management Console. Untuk AWS CloudFormation template yang membuat API ini dan semua resource terkait, lihat template.yaml.

Topik

- Langkah 1: Buat dependensi
- Langkah 2: Buat API pribadi
- Langkah 3: Buat metode dan integrasi
- Langkah 4: Lampirkan kebijakan sumber daya
- Langkah 5: Menerapkan API Anda
- Langkah 6: Verifikasi bahwa API Anda tidak dapat diakses publik
- Langkah 7: Hubungkan ke instans di VPC Anda dan panggil API Anda

Langkah 1: Buat dependensi

Unduh dan unzip AWS CloudFormation template ini. Anda menggunakan template untuk membuat semua dependensi untuk API pribadi Anda, termasuk VPC Amazon, titik akhir VPC, dan fungsi Lambda yang berfungsi sebagai backend API Anda. Anda membuat API pribadi nanti.

Untuk membuat AWS CloudFormation tumpukan

1. Buka AWS CloudFormation konsol di https://console.aws.amazon.com/cloudformation.
2. Pilih Buat tumpukan kemudian pilih Dengan sumber daya baru (standar).
3. Untuk Tentukan templat, pilih Unggah file templat.
4. Pilih template yang Anda unduh.
5. Pilih Berikutnya.
6. Untuk nama Stack, masukkan **`private-api-tutorial`** dan kemudian pilih Berikutnya.
7. Untuk opsi Konfigurasi tumpukan, pilih Berikutnya.
8. Untuk Kemampuan, akui bahwa AWS CloudFormation dapat membuat sumber daya IAM di akun Anda.
9. Pilih Buat tumpukan.

AWS CloudFormation menyediakan dependensi untuk API Anda, yang dapat memakan waktu beberapa menit. Ketika status AWS CloudFormation tumpukan Anda adalah CREATE_COMPLETE, pilih Output. Perhatikan ID titik akhir VPC Anda. Anda membutuhkannya untuk langkah-langkah selanjutnya dalam tutorial ini.

Langkah 2: Buat API pribadi

Anda membuat API pribadi untuk mengizinkan hanya klien dalam VPC Anda untuk mengaksesnya.

Untuk membuat API pribadi

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih Create API, lalu untuk REST API, pilih Build.
3. Untuk nama API, masukkan**`private-api-tutorial`**.

4. Untuk jenis endpoint API, pilih Private.
5. Untuk ID titik akhir VPC, masukkan ID titik akhir VPC dari Output tumpukan Anda. AWS CloudFormation
6. Pilih Buat API.

Langkah 3: Buat metode dan integrasi

Anda membuat `GET` metode dan integrasi Lambda untuk menangani `GET` permintaan ke API Anda. Saat klien memanggil API Anda, API Gateway mengirimkan permintaan ke fungsi Lambda yang Anda buat di Langkah 1, lalu mengembalikan respons ke klien.

Untuk membuat metode dan integrasi

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API Anda.
3. Pilih sumber daya/, lalu pilih Create method.
4. Untuk jenis Metode pilih`GET`.
5. Untuk jenis Integrasi, pilih fungsi Lambda.
6. Aktifkan integrasi proxy Lambda. Dengan integrasi proxy Lambda, API Gateway mengirimkan peristiwa ke Lambda dengan struktur yang ditentukan, dan mengubah respons dari fungsi Lambda Anda menjadi respons HTTP.
7. Untuk fungsi Lambda, pilih fungsi yang Anda buat dengan AWS CloudFormation template di Langkah 1. Nama fungsi dimulai dengan**`private-api-tutorial`**.
8. Pilih metode Buat.

Langkah 4: Lampirkan kebijakan sumber daya

Anda melampirkan kebijakan sumber daya ke API Anda yang memungkinkan klien untuk menjalankan API Anda hanya melalui titik akhir VPC Anda. Untuk lebih membatasi akses ke API Anda, Anda juga dapat mengonfigurasi kebijakan titik akhir VPC untuk titik akhir VPC Anda, tetapi itu tidak diperlukan untuk tutorial ini.

Untuk melampirkan kebijakan sumber daya

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API Anda.

3. Pilih Kebijakan sumber daya, lalu pilih Buat kebijakan.
4. Masukkan kebijakan berikut. Ganti ***VPCeID dengan ID*** titik akhir VPC Anda dari Output tumpukan Anda. AWS CloudFormation

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "execute-api:/*",
            "Condition": {
                "StringNotEquals": {
                    "aws:sourceVpce": "vpceID"
                }
            }
        },
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "execute-api:/*"
        }
    ]
}
```

5. Pilih Simpan perubahan.

Langkah 5: Menerapkan API Anda

Selanjutnya, Anda menerapkan API Anda untuk membuatnya tersedia bagi klien di Amazon VPC Anda.

Untuk menerapkan API

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API Anda.
3. Pilih Deploy API.
4. Untuk Stage, pilih New stage.

5. Untuk nama Panggung, masukkan**`test`**.
6. (Opsional) Untuk Deskripsi, masukkan deskripsi.
7. Pilih Deploy.

Sekarang Anda siap untuk menguji API Anda.

Langkah 6: Verifikasi bahwa API Anda tidak dapat diakses publik

Gunakan `curl` untuk memverifikasi bahwa Anda tidak dapat menjalankan API dari luar VPC Amazon Anda.

Untuk menguji API Anda

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API Anda.
3. Di panel navigasi utama, pilih Tahapan, lalu pilih tahap pengujian.
4. Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda. URL terlihat seperti`https://abcdef123.execute-api.us-west-2.amazonaws.com/test`. Titik akhir VPC yang Anda buat di Langkah 1 memiliki DNS pribadi yang diaktifkan, sehingga Anda dapat menggunakan URL yang disediakan untuk menjalankan API Anda.
5. Gunakan curl untuk mencoba menjalankan API Anda dari luar VPC Anda.

   ```
   curl https://abcdef123.execute-api.us-west-2.amazonaws.com/test
   ```

   Curl menunjukkan bahwa titik akhir API Anda tidak dapat diselesaikan. Jika Anda mendapatkan respons yang berbeda, kembali ke Langkah 2, dan pastikan Anda memilih Private untuk jenis endpoint API Anda.

   ```
   curl: (6) Could not resolve host: abcdef123.execute-api.us-west-2.amazonaws.com/
   test
   ```

Selanjutnya, Anda terhubung ke instans Amazon EC2 di VPC Anda untuk menjalankan API Anda.

Langkah 7: Hubungkan ke instans di VPC Anda dan panggil API Anda

Selanjutnya, Anda menguji API Anda dari dalam VPC Amazon Anda. Untuk mengakses API pribadi Anda, Anda terhubung ke instans Amazon EC2 di VPC, lalu menggunakan curl untuk menjalankan

API Anda. Anda menggunakan Systems Manager Session Manager untuk terhubung ke instans Anda di browser.

Untuk menguji API Anda

1. Buka konsol Amazon EC2 di https://console.aws.amazon.com/ec2/.
2. Pilih Instans.
3. Pilih contoh bernama private-api-tutorialyang Anda buat dengan AWS CloudFormation template di Langkah 1.
4. Pilih Connect dan kemudian pilih Session Manager.
5. Pilih Connect untuk meluncurkan sesi berbasis browser ke instans Anda.
6. Di sesi Session Manager Anda, gunakan curl untuk menjalankan API Anda. Anda dapat menjalankan API karena Anda menggunakan instance di VPC Amazon Anda.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/test
```

Verifikasi bahwa Anda mendapatkan responsHello from Lambda!.

Anda berhasil membuat API yang hanya dapat diakses dari dalam VPC Amazon Anda dan kemudian memverifikasi bahwa itu berfungsi.

Langkah 8: Membersihkan

Untuk mencegah biaya yang tidak perlu, hapus sumber daya yang Anda buat sebagai bagian dari tutorial ini. Langkah-langkah berikut menghapus REST API dan AWS CloudFormation tumpukan Anda.

Untuk menghapus REST API

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pada halaman API, pilih API. Pilih tindakan API, pilih Hapus API, lalu konfirmasikan pilihan Anda.

Untuk menghapus AWS CloudFormation tumpukan

1. Buka AWS CloudFormation konsol di https://console.aws.amazon.com/cloudformation.
2. Pilih AWS CloudFormation tumpukan Anda.
3. Pilih Hapus dan kemudian konfirmasikan pilihan Anda.

Langkah selanjutnya: Otomatiskan dengan AWS CloudFormation

Anda dapat mengotomatiskan pembuatan dan pembersihan semua AWS sumber daya yang terlibat dalam tutorial ini. Untuk contoh AWS CloudFormation template lengkap, lihat template.yaml.

Tutorial API HTTP Gateway

Tutorial berikut menyediakan latihan langsung untuk membantu Anda mempelajari tentang API API Gateway HTTP.

Topik

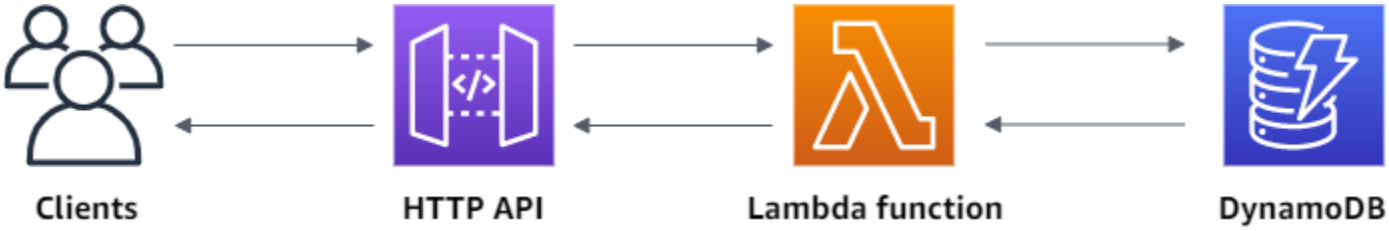
- Tutorial: Bangun API CRUD dengan Lambda dan DynamoDB
- Tutorial: Membangun API HTTP dengan integrasi pribadi ke layanan Amazon ECS

Tutorial: Bangun API CRUD dengan Lambda dan DynamoDB

Dalam tutorial ini, Anda membuat API tanpa server yang membuat, membaca, memperbarui, dan menghapus item dari tabel DynamoDB. DynamoDB adalah layanan basis data NoSQL terkelola penuh yang memberikan performa yang cepat dan dapat diprediksi dengan skalabilitas sempurna. Tutorial ini membutuhkan waktu sekitar 30 menit untuk menyelesaikannya, dan Anda dapat melakukannya dalam TingkatAWS Gratis.

Pertama, Anda membuat tabel DynamoDB menggunakan konsol DynamoDB. Kemudian Anda membuat fungsi Lambda menggunakanAWS Lambda konsol. Selanjutnya, Anda membuat API Gateway menggunakan konsol API Gateway. Terakhir, Anda menguji API Anda.

Saat Anda memanggil API HTTP Anda, API Gateway merutekan permintaan tersebut ke fungsi Lambda Anda. Fungsi Lambda berinteraksi dengan DynamoDB, dan mengembalikan respons ke API Gateway. API Gateway kemudian mengembalikan respons kepada Anda.



Untuk menyelesaikan latihan ini, Anda memerlukanAWS akun danAWS Identity and Access Management pengguna dengan akses konsol. Untuk informasi selengkapnya, lihat Prasyarat.

Dalam tutorial ini, Anda menggunakanAWS Management Console. UntukAWS SAM template yang membuat API ini dan semua sumber daya terkait, lihat `template.yaml`.

Topik

- Langkah 1: Buat Tabel DynamoDB
- Langkah 2: Buat fungsi Lambda
- Langkah 3: Buat API HTTP
- Langkah 4: Buat rute
- Langkah 5: Buat Integrasi
- Langkah 6: Lampirkan integrasi Anda ke rute
- Langkah 7: Uji API Anda
- Langkah 8: Membersihkan
- Langkah selanjutnya: Otomatiskan denganAWS SAM atauAWS CloudFormation

Langkah 1: Buat Tabel DynamoDB

Anda menggunakan tabel DynamoDB untuk menyimpan data untuk API Anda.

Setiap item memiliki ID unik, yang kami gunakan sebagai kunci partisi untuk tabel.

Untuk membuat tabel DynamoDB

1. Buka konsol DynamoDB di https://console.aws.amazon.com/dynamodb/.
2. Pilih Buat tabel.

3. Untuk Nama tabel, masukkan **`http-crud-tutorial-items`**.
4. Untuk kunci Partisi, masukkan**`id`**.
5. Pilih Buat tabel.

Langkah 2: Buat fungsi Lambda

Anda membuat fungsi Lambda untuk backend API Anda. Fungsi Lambda ini membuat, membaca, memperbarui, dan menghapus item dari DynamoDB. Fungsi ini menggunakan event dari API Gateway untuk menentukan cara berinteraksi dengan DynamoDB. Untuk mempermudah tutorial ini menggunakan fungsi Lambda tunggal. Sebagai praktik terbaik, Anda harus membuat fungsi terpisah untuk setiap rute.

Untuk membuat fungsi Lambda

1. Masuk ke konsol Lambda di https://console.aws.amazon.com/lambda.
2. Pilih Buat fungsi.
3. Untuk Nama fungsi, masukkan **`http-crud-tutorial-function`**.
4. Di bawah Izin pilih Ubah peran eksekusi default.
5. Pilih Buat peran baru dari templatAWS kebijakan.
6. Untuk Nama peran, masukkan **`http-crud-tutorial-role`**.
7. Untuk template Kebijakan, pilih**`Simple microservice permissions`**. Kebijakan ini memberikan izin fungsi Lambda untuk berinteraksi dengan DynamoDB.

> **Note**
>
> Tutorial ini menggunakan kebijakan terkelola untuk kesederhanaan. Sebagai praktik terbaik, Anda harus membuat kebijakan IAM Anda sendiri untuk memberikan izin minimum yang diperlukan.

8. Pilih Buat fungsi.
9. Buka`index.mjs` di editor kode konsol, dan ganti isinya dengan kode berikut. Pilih Deploy untuk memperbarui fungsi Anda.

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
```

```
  DynamoDBDocumentClient,
  ScanCommand,
  PutCommand,
  GetCommand,
  DeleteCommand,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});

const dynamo = DynamoDBDocumentClient.from(client);

const tableName = "http-crud-tutorial-items";

export const handler = async (event, context) => {
  let body;
  let statusCode = 200;
  const headers = {
    "Content-Type": "application/json",
  };

  try {
    switch (event.routeKey) {
      case "DELETE /items/{id}":
        await dynamo.send(
          new DeleteCommand({
            TableName: tableName,
            Key: {
              id: event.pathParameters.id,
            },
          })
        );
        body = `Deleted item ${event.pathParameters.id}`;
        break;
      case "GET /items/{id}":
        body = await dynamo.send(
          new GetCommand({
            TableName: tableName,
            Key: {
              id: event.pathParameters.id,
            },
          })
        );
        body = body.Item;
        break;
```

```
      case "GET /items":
        body = await dynamo.send(
          new ScanCommand({ TableName: tableName })
        );
        body = body.Items;
        break;
      case "PUT /items":
        let requestJSON = JSON.parse(event.body);
        await dynamo.send(
          new PutCommand({
            TableName: tableName,
            Item: {
              id: requestJSON.id,
              price: requestJSON.price,
              name: requestJSON.name,
            },
          })
        );
        body = `Put item ${requestJSON.id}`;
        break;
      default:
        throw new Error(`Unsupported route: "${event.routeKey}"`);
    }
  } catch (err) {
    statusCode = 400;
    body = err.message;
  } finally {
    body = JSON.stringify(body);
  }

  return {
    statusCode,
    body,
    headers,
  };
};
```

Langkah 3: Buat API HTTP

API HTTP menyediakan titik akhir HTTP untuk fungsi Lambda Anda. Dalam langkah ini, Anda membuat API kosong. Pada langkah-langkah berikut, Anda mengonfigurasi rute dan integrasi untuk menghubungkan API dan fungsi Lambda Anda.

Untuk membuat API HTTP

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih Buat API, dan kemudian untuk HTTP API, pilih Build.
3. Untuk nama API, masukkan**`http-crud-tutorial-api`**.
4. Pilih Selanjutnya.
5. Untuk Konfigurasi rute, pilih Berikutnya untuk melewati pembuatan rute. Anda membuat rute nanti.
6. Periksa tahap yang dibuat API Gateway untuk Anda, lalu pilih Berikutnya.
7. Pilih Create (Buat).

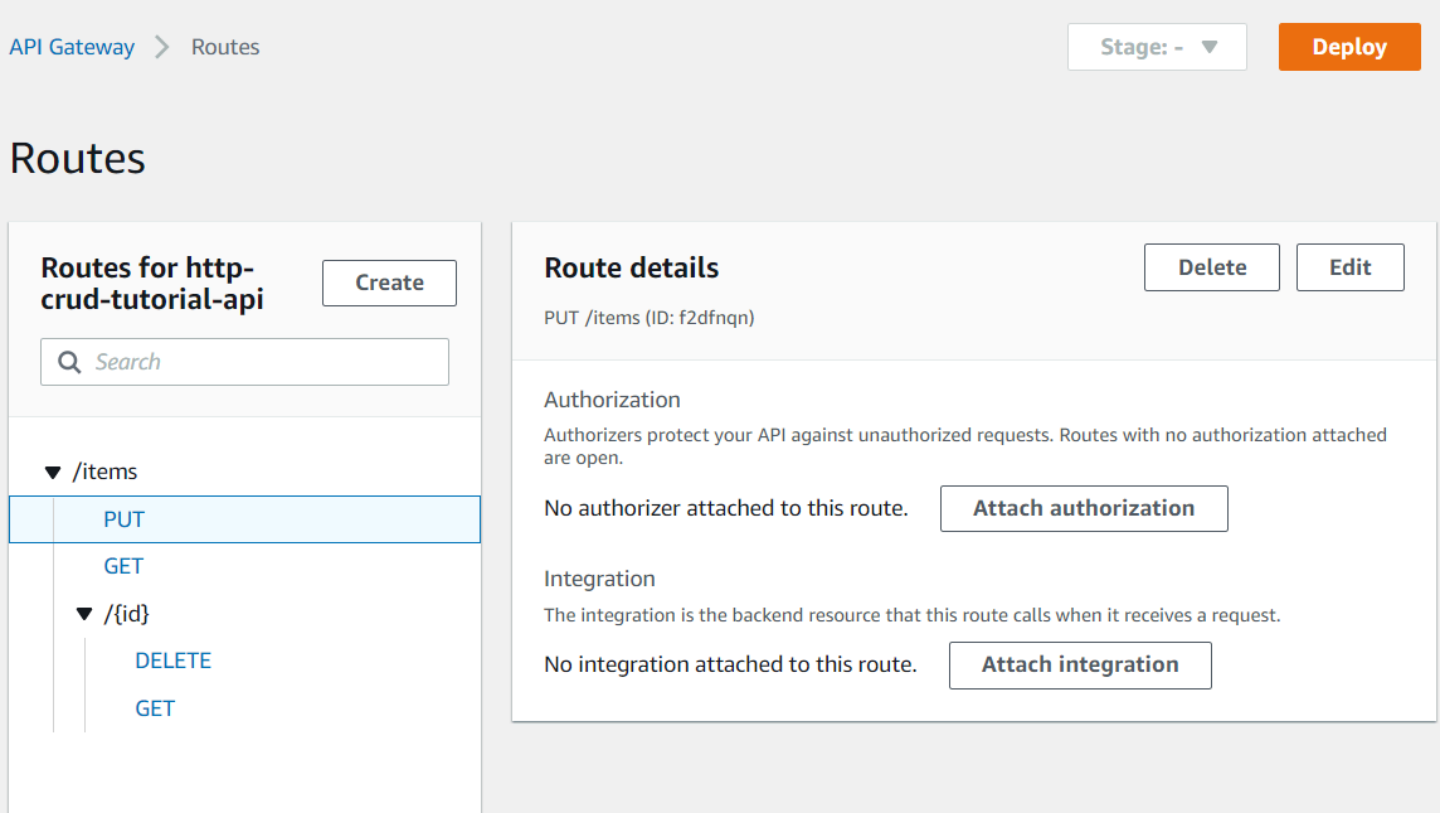
Langkah 4: Buat rute

Rute adalah cara untuk mengirim permintaan API yang masuk ke sumber daya backend. Rute terdiri dari dua bagian: metode HTTP dan jalur sumber daya, misalnya,`GET /items`. Untuk contoh API ini, kita membuat empat rute:

- `GET /items/{id}`
- `GET /items`
- `PUT /items`
- `DELETE /items/{id}`

Untuk membuat rute

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API Anda.
3. Pilih Rute.
4. Pilih Create (Buat).
5. Untuk Metode, pilih**`GET`**.
6. Untuk jalan, masukkan**`/items/{id}`**. `{id}`Di akhir jalur adalah parameter jalur yang diambil API Gateway dari jalur permintaan saat klien membuat permintaan.
7. Pilih Create (Buat).
8. Ulangi langkah 4-7 untuk`GET /items`,`DELETE /items/{id}`, dan`PUT /items`.



Langkah 5: Buat Integrasi

Anda membuat integrasi untuk menghubungkan rute ke sumber daya backend. Untuk contoh API ini, Anda membuat satu integrasi Lambda yang Anda gunakan untuk semua rute.

Untuk membuat integrasi

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API Anda.
3. Pilih Integrasi.
4. Pilih Kelola integrasi, lalu pilih Buat.
5. Lewati Lampirkan integrasi ini ke rute. Anda menyelesaikannya di langkah berikutnya.
6. Untuk tipe Integrasi, pilih fungsi Lambda.
7. Untuk fungsi Lambda, masukkan**`http-crud-tutorial-function`**.
8. Pilih Create (Buat).

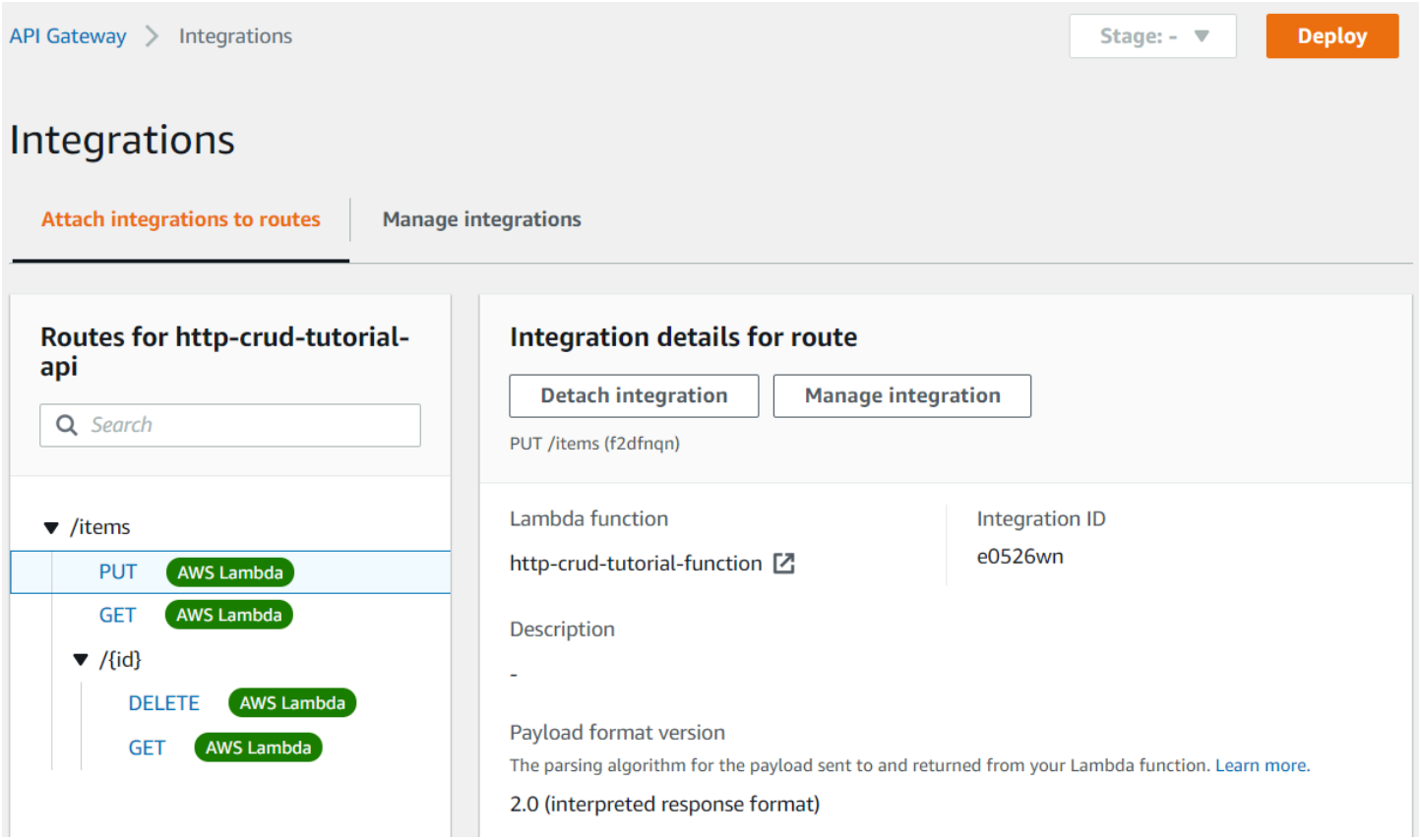
Langkah 6: Lampirkan integrasi Anda ke rute

Untuk contoh API ini, Anda menggunakan integrasi Lambda yang sama untuk semua rute. Setelah Anda melampirkan integrasi ke semua rute API, fungsi Lambda Anda dipanggil ketika klien memanggil salah satu rute Anda.

Untuk melampirkan integrasi ke rute

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API Anda.
3. Pilih Integrasi.
4. Pilih rute.
5. Di bawah Pilih integrasi yang ada, pilih**`http-crud-tutorial-function`**.
6. Pilih Lampirkan integrasi.
7. Ulangi langkah 4-6 untuk semua rute.

Semua rute menunjukkan bahwaAWS Lambda integrasi terpasang.



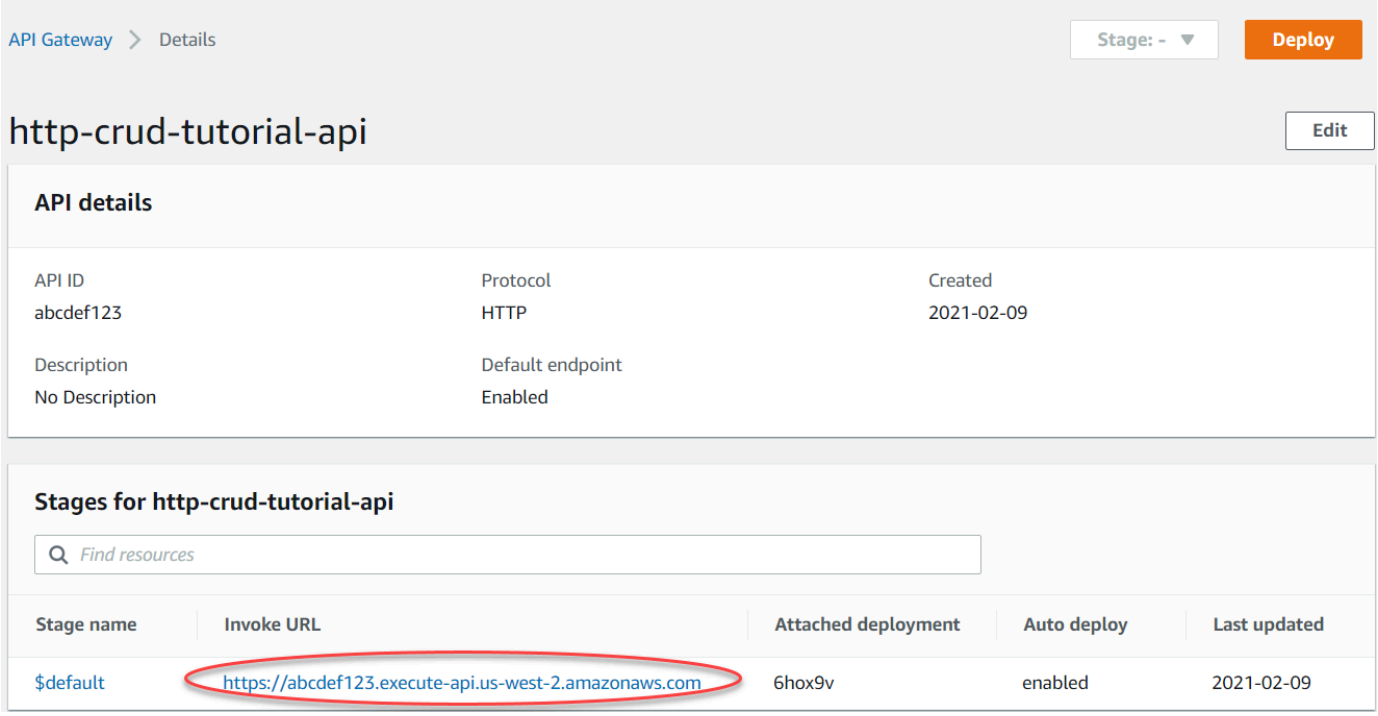
Setelah Anda memiliki API HTTP dengan rute dan integrasi, Anda dapat menguji API Anda.

Langkah 7: Uji API Anda

Untuk memastikan bahwa API Anda bekerja, Anda menggunakan curl.

Untuk mendapatkan URL untuk memanggil API Anda

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API Anda.
3. Perhatikan URL panggilan API Anda. Itu muncul di bawah URL Invoke pada halaman Detail.



4. Salin URL panggilan API Anda.

   URL lengkap terlihat seperti`https://abcdef123.execute-api.us-west-2.amazonaws.com`.

Untuk membuat atau memperbarui item

- Gunakan perintah berikut untuk membuat atau memperbarui item. Perintah ini mencakup badan permintaan dengan ID, harga, dan nama item.

```
curl -X "PUT" -H "Content-Type: application/json" -d "{\"id\": \"123\",
 \"price\": 12345, \"name\": \"myitem\"}" https://abcdef123.execute-api.us-
west-2.amazonaws.com/items
```

Untuk mendapatkan semua item

- Gunakan perintah berikut untuk mencantumkan semua item.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/items
```

Untuk mendapatkan item

- Gunakan perintah berikut untuk mendapatkan item dengan ID-nya.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/items/123
```

Untuk menghapus item

1. Gunakan perintah berikut untuk menghapus item.

```
curl -X "DELETE" https://abcdef123.execute-api.us-west-2.amazonaws.com/items/123
```

2. Dapatkan semua item untuk memverifikasi bahwa item telah dihapus.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/items
```

Langkah 8: Membersihkan

Untuk mencegah biaya yang tidak perlu, hapus sumber daya yang Anda buat sebagai bagian dari latihan memulai ini. Langkah-langkah berikut menghapus API HTTP Anda, fungsi Lambda Anda, dan sumber daya terkait.

Untuk menghapus tabel DynamoDB

1. Buka konsol DynamoDB di https://console.aws.amazon.com/dynamodb/.
2. Pilih tabel Anda.

3. Pilih Hapus tabel.
4. Konfirmasikan pilihan Anda, lalu pilih Hapus.

Untuk menghapus API HTTP

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pada halaman API, pilih API. Pilih Actions (Tindakan), lalu pilih Delete (Hapus).
3. Pilih Delete (Hapus).

Untuk menghapus fungsi Lambda

1. Masuk ke konsol Lambda di https://console.aws.amazon.com/lambda.
2. Pada halaman Functions, pilih fungsi. Pilih Actions (Tindakan), lalu pilih Delete (Hapus).
3. Pilih Delete (Hapus).

Untuk menghapus grup log fungsi Lambda

1. Di CloudWatch konsol Amazon, buka halaman Grup Log.
2. Pada halaman Grup log, pilih grup log fungsi (`/aws/lambda/http-crud-tutorial-function`). Pilih Tindakan, lalu pilih Hapus grup log.
3. Pilih Delete (Hapus).

Untuk menghapus peran eksekusi fungsi Lambda

1. DiAWS Identity and Access Management konsol, buka halaman Peran.
2. Pilih peran fungsi, misalnya,`http-crud-tutorial-role`.
3. Pilih Hapus peran.
4. Pilih Ya, hapus.

Langkah selanjutnya: Otomatiskan denganAWS SAM atauAWS CloudFormation

Anda dapat mengotomatisasi pembuatan dan pembersihanAWS sumber daya dengan menggunakanAWS CloudFormation atauAWS SAM. Untuk contohAWS SAM template untuk tutorial ini, lihat `template.yaml`.

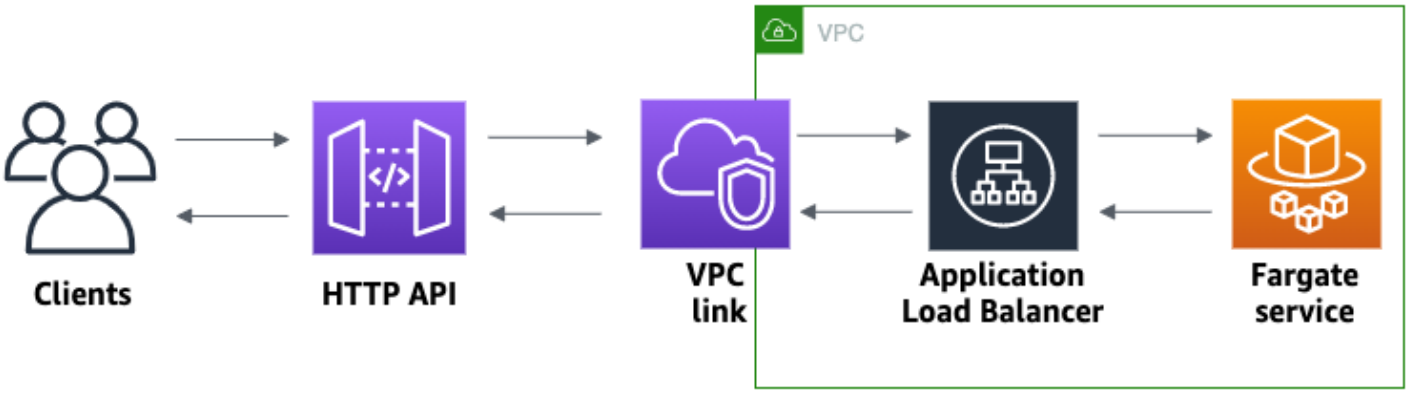
MisalnyaAWS CloudFormation template, lihat contohAWS CloudFormation template.

Tutorial: Membangun API HTTP dengan integrasi pribadi ke layanan Amazon ECS

Dalam tutorial ini, Anda membuat API tanpa server yang terhubung ke layanan Amazon ECS yang berjalan di VPC Amazon. Klien di luar VPC Amazon Anda dapat menggunakan API untuk mengakses layanan Amazon ECS Anda.

Tutorial ini membutuhkan waktu sekitar satu jam untuk menyelesaikannya. Pertama, Anda menggunakan AWS CloudFormation template untuk membuat Amazon VPC dan layanan Amazon ECS. Kemudian Anda menggunakan konsol API Gateway untuk membuat tautan VPC. Tautan VPC memungkinkan API Gateway untuk mengakses layanan Amazon ECS yang berjalan di VPC Amazon Anda. Selanjutnya, Anda membuat API HTTP yang menggunakan tautan VPC untuk terhubung ke layanan Amazon ECS Anda. Terakhir, Anda menguji API Anda.

Saat Anda menjalankan API HTTP, API Gateway merutekan permintaan ke layanan Amazon ECS Anda melalui tautan VPC Anda, lalu mengembalikan respons dari layanan.



Untuk menyelesaikan tutorial ini, Anda memerlukan AWS akun dan AWS Identity and Access Management pengguna dengan akses konsol. Untuk informasi selengkapnya, lihat Prasyarat.

Dalam tutorial ini, Anda menggunakan AWS Management Console. Untuk AWS CloudFormation template yang membuat API ini dan semua resource terkait, lihat template.yaml.

Topik

- Langkah 1: Buat layanan Amazon ECS
- Langkah 2: Buat tautan VPC

Langkah 1: Buat layanan Amazon ECS

Amazon ECS adalah layanan manajemen kontainer yang memudahkan untuk menjalankan, menghentikan, dan mengelola kontainer Docker di cluster. Dalam tutorial ini, Anda menjalankan cluster Anda pada infrastruktur tanpa server yang dikelola oleh Amazon ECS.

Unduh dan unzip AWS CloudFormation template ini, yang membuat semua dependensi untuk layanan, termasuk VPC Amazon. Anda menggunakan template untuk membuat layanan Amazon ECS yang menggunakan Application Load Balancer.

Untuk membuat AWS CloudFormation tumpukan

1. Buka AWS CloudFormation konsol di https://console.aws.amazon.com/cloudformation.
2. Pilih Buat tumpukan kemudian pilih Dengan sumber daya baru (standar).
3. Untuk Tentukan templat, pilih Unggah file templat.
4. Pilih template yang Anda unduh.
5. Pilih Berikutnya.
6. Untuk nama Stack, masukkan **`http-api-private-integrations-tutorial`** dan kemudian pilih Berikutnya.
7. Untuk opsi Konfigurasi tumpukan, pilih Berikutnya.
8. Untuk Kemampuan, akui bahwa AWS CloudFormation dapat membuat sumber daya IAM di akun Anda.
9. Pilih Buat tumpukan.

AWS CloudFormation menyediakan layanan ECS, yang dapat memakan waktu beberapa menit. Ketika status AWS CloudFormation tumpukan Anda adalah CREATE_COMPLETE, Anda siap untuk melanjutkan ke langkah berikutnya.

Langkah 2: Buat tautan VPC

Tautan VPC memungkinkan API Gateway untuk mengakses sumber daya pribadi di VPC Amazon. Anda menggunakan tautan VPC untuk memungkinkan klien mengakses layanan Amazon ECS Anda melalui HTTP API Anda.

Untuk membuat tautan VPC

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pada panel navigasi utama, pilih tautan VPC lalu pilih Buat.

   Anda mungkin perlu memilih ikon menu untuk membuka panel navigasi utama.
3. Untuk Pilih versi tautan VPC, pilih tautan VPC untuk API HTTP.
4. Untuk Nama, masukkan **`private-integrations-tutorial`**.
5. Untuk VPC, pilih VPC yang Anda buat di langkah 1. Nama harus dimulai dengan PrivateIntegrationsStack.
6. Untuk Subnet, pilih dua subnet pribadi di VPC Anda. Nama mereka diakhiri dengan `PrivateSubnet`
7. Pilih Buat.

Setelah Anda membuat tautan VPC, API Gateway menyediakan Antarmuka Jaringan Elastis untuk mengakses VPC Anda. Prosesnya bisa memakan waktu beberapa menit. Sementara itu, Anda dapat membuat API Anda.

Langkah 3: Buat API HTTP

HTTP API menyediakan endpoint HTTP untuk layanan Amazon ECS Anda. Pada langkah ini, Anda membuat API kosong. Pada Langkah 4 dan 5, Anda mengonfigurasi rute dan integrasi untuk menghubungkan API dan layanan Amazon ECS Anda.

Untuk membuat API HTTP

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih Buat API, lalu untuk HTTP API, pilih Build.
3. Untuk nama API, masukkan**`http-private-integrations-tutorial`**.
4. Pilih Berikutnya.

5. Untuk Mengonfigurasi rute, pilih Berikutnya untuk melewati pembuatan rute. Anda membuat rute nanti.
6. Tinjau tahapan yang dibuat API Gateway untuk Anda. API Gateway membuat `$default` panggung dengan penerapan otomatis diaktifkan, yang merupakan pilihan terbaik untuk tutorial ini. Pilih Berikutnya.
7. Pilih Buat.

Langkah 4: Buat rute

Rute adalah cara untuk mengirim permintaan API yang masuk ke sumber daya backend. Rute terdiri dari dua bagian: metode HTTP dan jalur sumber daya, misalnya,`GET /items`. Untuk contoh API ini, kita membuat satu rute.

Untuk membuat rute

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API Anda.
3. Pilih Rute.
4. Pilih Buat.
5. Untuk Metode, pilih**ANY**.
6. Untuk jalan, masuk**/{proxy+}**. `{proxy+}`Di ujung jalan adalah variabel jalur serakah. API Gateway mengirimkan semua permintaan ke API Anda ke rute ini.
7. Pilih Buat.

Langkah 5: Buat integrasi

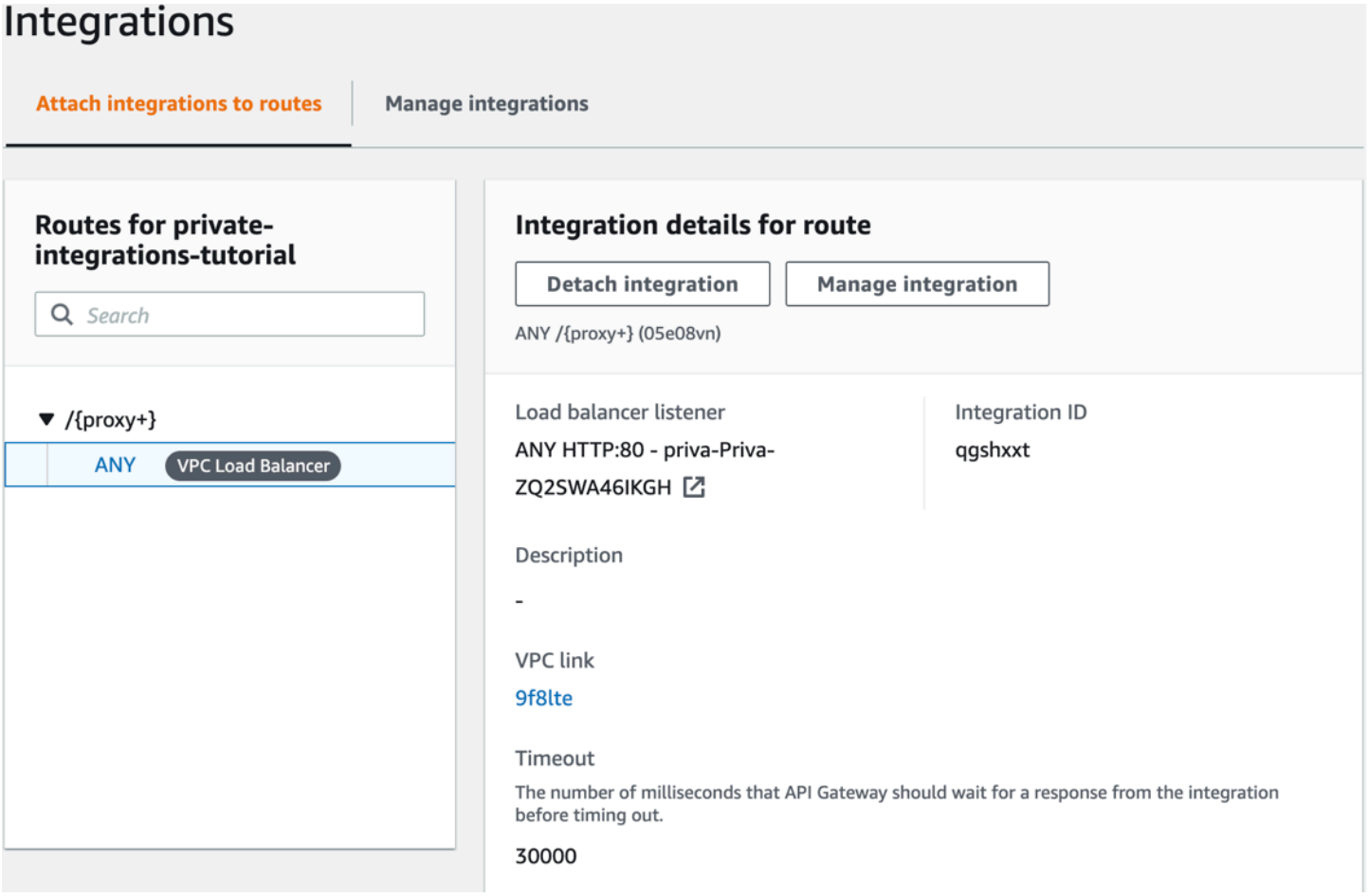
Anda membuat integrasi untuk menghubungkan rute ke sumber daya backend.

Untuk membuat integrasi

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API Anda.
3. Pilih Integrasi.
4. Pilih Kelola integrasi dan kemudian pilih Buat.
5. Untuk Lampirkan integrasi ini ke rute, pilih rute ANY/{proxy+} yang Anda buat sebelumnya.

6. Untuk jenis Integrasi, pilih Sumber daya pribadi.
7. Untuk detail Integrasi, pilih Pilih secara manual.
8. Untuk layanan Target, pilih ALB/NLB.
9. Untuk Load balancer, pilih load balancer yang Anda buat dengan AWS CloudFormation template di Langkah 1. Namanya harus dimulai dengan HTTP-Priva.
10. Untuk Listener, pilih**`HTTP 80`**.
11. Untuk tautan VPC, pilih tautan VPC yang Anda buat di Langkah 2. Seharusnya namanya`private-integrations-tutorial`.
12. Pilih Buat.

Untuk memverifikasi bahwa rute dan integrasi Anda telah diatur dengan benar, pilih Lampirkan integrasi ke rute. Konsol menunjukkan bahwa Anda memiliki `ANY /{proxy+}` rute dengan integrasi ke Load Balancer VPC.



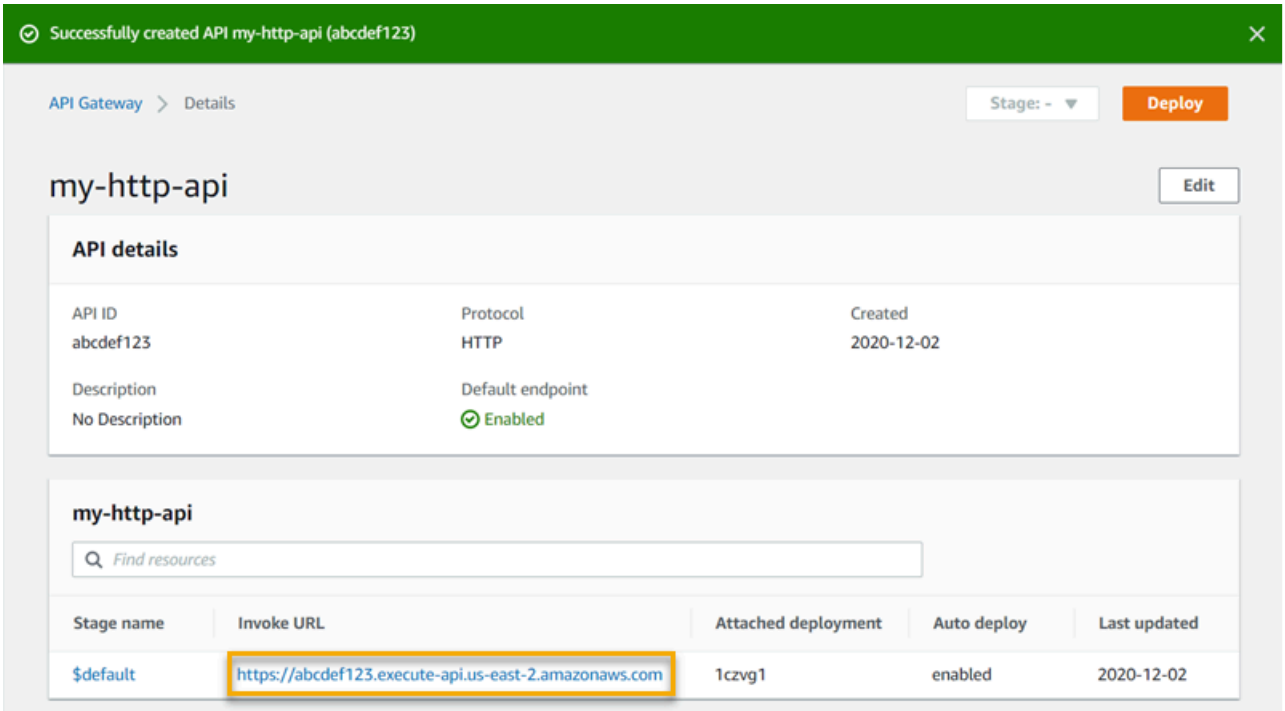
Sekarang Anda siap untuk menguji API Anda.

Langkah 6: Uji API Anda

Selanjutnya, Anda menguji API Anda untuk memastikan bahwa itu berfungsi. Untuk mempermudah, gunakan browser web untuk menjalankan API Anda.

Untuk menguji API Anda

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API Anda.
3. Perhatikan URL pemanggilan API Anda.



4. Di browser web, buka URL pemanggilan API Anda.

   URL lengkap akan terlihat seperti`https://abcdef123.execute-api.us-east-2.amazonaws.com`.

   Browser Anda mengirimkan `GET` permintaan ke API.

5. Verifikasi bahwa respons API Anda adalah pesan selamat datang yang memberi tahu Anda bahwa aplikasi Anda berjalan di Amazon ECS.

   Jika Anda melihat pesan selamat datang, Anda berhasil membuat layanan Amazon ECS yang berjalan di VPC Amazon, dan Anda menggunakan API API Gateway HTTP dengan tautan VPC untuk mengakses layanan Amazon ECS.

Langkah 7: Bersihkan

Untuk mencegah biaya yang tidak perlu, hapus sumber daya yang Anda buat sebagai bagian dari tutorial ini. Langkah-langkah berikut menghapus tautan VPC, AWS CloudFormation tumpukan, dan API HTTP Anda.

Untuk menghapus API HTTP

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pada halaman API, pilih API. Pilih Tindakan, pilih Hapus, lalu konfirmasikan pilihan Anda.

Untuk menghapus tautan VPC

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih tautan VPC.
3. Pilih tautan VPC Anda, pilih Hapus, lalu konfirmasikan pilihan Anda.

Untuk menghapus AWS CloudFormation tumpukan

1. Buka AWS CloudFormation konsol di https://console.aws.amazon.com/cloudformation.
2. Pilih AWS CloudFormation tumpukan Anda.
3. Pilih Hapus dan kemudian konfirmasikan pilihan Anda.

Langkah selanjutnya: Otomatisasi dengan AWS CloudFormation

Anda dapat mengotomatiskan pembuatan dan pembersihan semua AWS sumber daya yang terlibat dalam tutorial ini. Untuk contoh AWS CloudFormation template lengkap, lihat template.yaml.

Amazon API GatewayWebSocketTutorial API

Tutorial berikut memberikan latihan langsung untuk membantu Anda belajar tentang API GatewayWebSocketAPI.

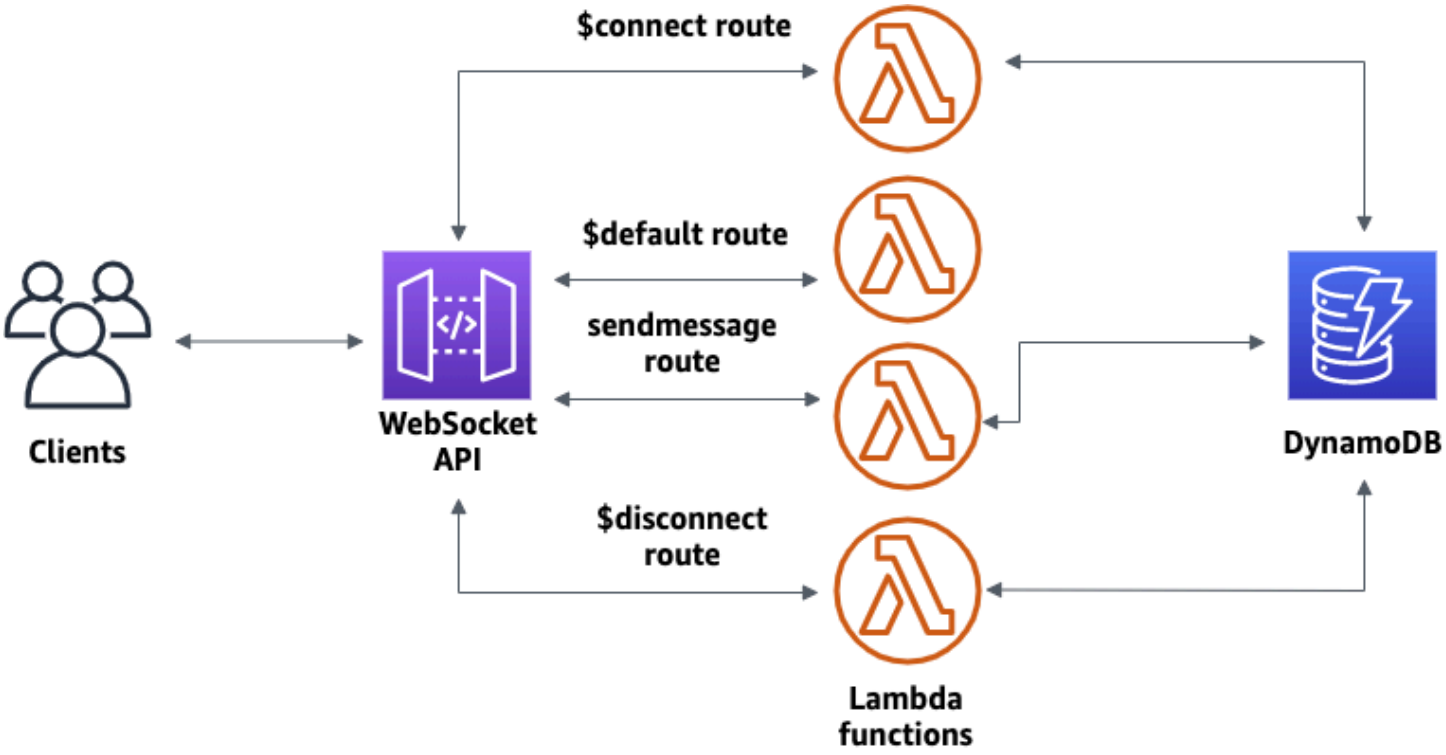
Topik

- Tutorial: Membangun aplikasi obrolan tanpa server dengan WebSocket API, Lambda, dan DynamoDB

Tutorial: Membangun aplikasi obrolan tanpa server dengan WebSocket API, Lambda, dan DynamoDB

Dalam tutorial ini, Anda akan membuat aplikasi obrolan tanpa server dengan WebSocket API. Dengan WebSocket API, Anda dapat mendukung komunikasi dua arah antar klien. Klien dapat menerima pesan tanpa harus melakukan polling untuk pembaruan.

Tutorial ini memakan waktu sekitar 30 menit untuk menyelesaikannya. Pertama, Anda akan menggunakan AWS CloudFormation template untuk membuat fungsi Lambda yang akan menangani permintaan API, serta tabel DynamoDB yang menyimpan ID klien Anda. Kemudian, Anda akan menggunakan konsol API Gateway untuk membuat WebSocket API yang terintegrasi dengan fungsi Lambda Anda. Terakhir, Anda akan menguji API Anda untuk memverifikasi bahwa pesan dikirim dan diterima.



Untuk menyelesaikan tutorial ini, Anda memerlukan AWS akun dan AWS Identity and Access Management pengguna dengan akses konsol. Untuk informasi selengkapnya, lihat Prasyarat.

Anda juga `wscat` perlu terhubung ke API Anda. Untuk informasi selengkapnya, lihat the section called “Gunakan `wscat` untuk terhubung ke WebSocket API dan mengirim pesan ke sana”.

Topik

- Langkah 1: Buat fungsi Lambda dan tabel DynamoDB

Langkah 1: Buat fungsi Lambda dan tabel DynamoDB

Unduh dan unzip template pembuatan aplikasi untuk AWS CloudFormation. Anda akan menggunakan template ini untuk membuat tabel Amazon DynamoDB untuk menyimpan ID klien aplikasi Anda. Setiap klien yang terhubung memiliki ID unik yang akan kita gunakan sebagai kunci partisi tabel. Template ini juga menciptakan fungsi Lambda yang memperbarui koneksi klien Anda di DynamoDB dan menangani pengiriman pesan ke klien yang terhubung.

Untuk membuat AWS CloudFormation tumpukan

1. Buka AWS CloudFormation konsol di https://console.aws.amazon.com/cloudformation.
2. Pilih Buat tumpukan kemudian pilih Dengan sumber daya baru (standar).
3. Untuk Tentukan templat, pilih Unggah file templat.
4. Pilih template yang Anda unduh.
5. Pilih Berikutnya.
6. Untuk nama Stack, masukkan **`websocket-api-chat-app-tutorial`** dan kemudian pilih Berikutnya.
7. Untuk opsi Konfigurasi tumpukan, pilih Berikutnya.
8. Untuk Kemampuan, akui bahwa AWS CloudFormation dapat membuat sumber daya IAM di akun Anda.
9. Pilih Buat tumpukan.

AWS CloudFormation ketentuan sumber daya yang ditentukan dalam template. Diperlukan beberapa menit untuk menyelesaikan penyediaan sumber daya Anda. Ketika status AWS CloudFormation tumpukan Anda adalah CREATE_COMPLETE, Anda siap untuk melanjutkan ke langkah berikutnya.

Langkah 2: Buat WebSocket API

Anda akan membuat WebSocket API untuk menangani koneksi klien dan merutekan permintaan ke fungsi Lambda yang Anda buat di Langkah 1.

Untuk membuat WebSocket API

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih Buat API. Kemudian untuk WebSocket API, pilih Build.
3. Untuk nama API, masukkan**`websocket-chat-app-tutorial`**.
4. Untuk ekspresi pemilihan Rute, masukkan**`request.body.action`**. Ekspresi pemilihan rute menentukan rute yang dipanggil API Gateway saat klien mengirim pesan.
5. Pilih Berikutnya.
6. Untuk rute Predefined, pilih Add $connect, Add $disconnect, dan Add $default. Rute $connect dan $disconnect adalah rute khusus yang dipanggil API Gateway secara otomatis saat klien terhubung atau terputus dari API. API Gateway memanggil `$default` rute ketika tidak ada rute lain yang cocok dengan permintaan.
7. Untuk rute kustom, pilih Tambahkan rute kustom. Untuk kunci Rute, masukkan**`sendmessage`**. Rute kustom ini menangani pesan yang dikirim ke klien yang terhubung.
8. Pilih Berikutnya.
9. Di bawah Lampirkan integrasi, untuk setiap rute dan jenis Integrasi, pilih Lambda.

   Untuk Lambda, pilih fungsi Lambda yang sesuai yang Anda buat di Langkah AWS CloudFormation 1. Setiap nama fungsi cocok dengan rute. Misalnya, untuk rute $connect, pilih fungsi bernama**`websocket-chat-app-tutorial-ConnectHandler`**.
10. Tinjau tahapan yang dibuat API Gateway untuk Anda. Secara default, API Gateway membuat nama panggung `production` dan secara otomatis menerapkan API Anda ke tahap itu. Pilih Berikutnya.
11. Pilih Buat dan terapkan.

Langkah 3: Uji API Anda

Selanjutnya, Anda akan menguji API Anda untuk memastikan bahwa itu berfungsi dengan benar. Gunakan `wscat` perintah untuk terhubung ke API.

Untuk mendapatkan URL pemanggilan untuk API Anda

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API Anda.
3. Pilih Tahapan, lalu pilih produksi.

4. Perhatikan WebSocket URL API Anda. URL akan terlihat seperti `wss://abcdef123.execute-api.us-east-2.amazonaws.com/production`.

Untuk terhubung ke API

1. Gunakan perintah berikut untuk terhubung ke API Anda. Saat Anda terhubung ke API, API Gateway akan memanggil `$connect` rute. Ketika rute ini dipanggil, ia memanggil fungsi Lambda yang menyimpan ID koneksi Anda di DynamoDB.

```
wscat -c wss://abcdef123.execute-api.us-west-2.amazonaws.com/production
```

```
Connected (press CTRL+C to quit)
```

2. Buka terminal baru dan jalankan wscat perintah lagi dengan parameter berikut.

```
wscat -c wss://abcdef123.execute-api.us-west-2.amazonaws.com/production
```

```
Connected (press CTRL+C to quit)
```

   Ini memberi Anda dua klien terhubung yang dapat bertukar pesan.

Untuk mengirim pesan

- API Gateway menentukan rute mana yang akan dipanggil berdasarkan ekspresi pemilihan rute API Anda. Ekspresi pemilihan rute API Anda adalah`$request.body.action`. Akibatnya, API Gateway memanggil `sendmessage` rute saat Anda mengirim pesan berikut:

```
{"action": "sendmessage", "message": "hello, everyone!"}
```

  Fungsi Lambda yang terkait dengan rute yang dipanggil mengumpulkan ID klien dari DynamoDB. Kemudian, fungsi tersebut memanggil API Gateway Management API dan mengirimkan pesan ke klien tersebut. Semua klien yang terhubung menerima pesan berikut:

```
< hello, everyone!
```

Untuk memanggil rute $default API Anda

- API Gateway memanggil rute default API Anda saat klien mengirim pesan yang tidak cocok dengan rute yang Anda tentukan. Fungsi Lambda yang terkait dengan `$default` rute menggunakan API Gateway Management API untuk mengirim informasi klien tentang koneksi mereka.

```
test
```

```
Use the sendmessage route to send a message. Your info:
 {"ConnectedAt":"2022-01-25T18:50:04.673Z","Identity":
{"SourceIp":"192.0.2.1","UserAgent":null},"LastActiveAt":"2022-01-25T18:50:07.642Z","connec
```

Untuk memutuskan sambungan dari API

- Tekan **CTRL+C** untuk memutuskan sambungan dari API Anda. Saat klien terputus dari API Anda, API Gateway akan memanggil rute API Anda. `$disconnect` Integrasi Lambda untuk `$disconnect` rute API Anda menghapus ID koneksi dari DynamoDB.

Langkah 4: Membersihkan

Untuk mencegah biaya yang tidak perlu, hapus sumber daya yang Anda buat sebagai bagian dari tutorial ini. Langkah-langkah berikut menghapus AWS CloudFormation tumpukan dan WebSocket API Anda.

Untuk menghapus WebSocket API

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pada halaman API, pilih `websocket-chat-app-tutorial` API Anda. Pilih Tindakan, pilih Hapus, lalu konfirmasikan pilihan Anda.

Untuk menghapus AWS CloudFormation tumpukan

1. Buka AWS CloudFormation konsol di https://console.aws.amazon.com/cloudformation.
2. Pilih AWS CloudFormation tumpukan Anda.
3. Pilih Hapus dan kemudian konfirmasikan pilihan Anda.

Langkah selanjutnya: Otomatisasi dengan AWS CloudFormation

Anda dapat mengotomatiskan pembuatan dan pembersihan semua sumber AWS daya yang terlibat dalam tutorial ini. Untuk AWS CloudFormation template yang membuat API ini dan semua resource terkait, lihat chat-app.yaml.

Bekerja dengan REST API

REST API di API Gateway adalah kumpulan sumber daya dan metode yang terintegrasi dengan backend HTTP endpoint, fungsi Lambda, atau lainnyaAWSlayanan. Anda dapat menggunakan fitur API Gateway untuk membantu Anda dengan semua aspek siklus hidup API, mulai dari pembuatan melalui pemantauan API produksi Anda.

API GatewaySISAAPI menggunakan model permintaan/respons di mana klien mengirimkan permintaan ke layanan dan layanan merespons kembali secara serentak. Model semacam ini cocok untuk berbagai jenis aplikasi yang bergantung pada komunikasi sinkron.

Topik

Mengembangkan REST API di API Gateway

Bagian ini memberikan detail tentang kemampuan API Gateway yang Anda butuhkan saat mengembangkan API Gateway API.

Saat Anda mengembangkan API Gateway API, Anda memutuskan sejumlah karakteristik API Anda. Karakteristik ini bergantung pada kasus penggunaan API Anda. Misalnya, Anda mungkin hanya ingin mengizinkan klien tertentu untuk memanggil API Anda, atau Anda mungkin ingin itu tersedia untuk semua orang. Anda mungkin ingin panggilan API untuk menjalankan fungsi Lambda, membuat kueri database, atau memanggil aplikasi.

Topik

- Menyiapkan transformasi data untuk REST API
- Tanggapan Gateway di API Gateway
- Mengaktifkan CORS untuk sumber daya REST API
- Bekerja dengan tipe media biner untuk REST API
- Memanggil REST API di Amazon API Gateway
- Mengkonfigurasi REST API menggunakan OpenAPI

Membuat REST API di Amazon API Gateway

Di Amazon API Gateway, Anda membuat REST API sebagai kumpulan entitas yang dapat diprogram yang dikenal sebagai sumber daya API Gateway. Misalnya, Anda menggunakan `RestApi`sumber daya untuk merepresentasikan API yang dapat berisi kumpulan `Resource`entitas. Setiap `Resource` entitas pada gilirannya dapat memiliki satu atau lebih `Method`sumber daya. Dinyatakan dalam parameter permintaan dan badan, a `Method` mendefinisikan antarmuka pemrograman aplikasi untuk klien untuk mengakses terbuka `Resource` dan mewakili permintaan masuk yang diajukan oleh klien. Anda kemudian membuat `Integration`sumber daya untuk mengintegrasikan `Method` dengan titik akhir backend, juga dikenal sebagai titik akhir integrasi, dengan meneruskan permintaan masuk ke URI titik akhir integrasi tertentu. Jika perlu, Anda mengubah parameter permintaan atau badan untuk memenuhi persyaratan backend. Untuk tanggapan, Anda dapat membuat `MethodResponse`sumber daya untuk mewakili respons permintaan yang diterima oleh klien dan Anda membuat `IntegrationResponse`sumber daya untuk mewakili respons permintaan yang dikembalikan oleh backend. Anda dapat mengonfigurasi respons integrasi untuk mengubah data respons backend sebelum mengembalikan data ke klien atau meneruskan respons backend apa adanya ke klien.

Untuk membantu pelanggan memahami API Anda, Anda juga dapat menyediakan dokumentasi untuk API, sebagai bagian dari pembuatan API atau setelah API dibuat. Untuk mengaktifkannya, tambahkan `DocumentationPart`sumber daya untuk entitas API yang didukung.

Untuk mengontrol cara klien memanggil API, gunakan izin IAM, otorisasi Lambda, atau kumpulan pengguna Amazon Cognito. Untuk mengukur penggunaan API Anda, siapkan rencana penggunaan untuk membatasi permintaan API. Anda dapat mengaktifkan ini saat membuat atau memperbarui API.

Anda dapat melakukan ini dan tugas lainnya dengan menggunakan konsol API Gateway, API Gateway REST API AWS CLI, atau salah satu AWS SDK. Kami membahas bagaimana melakukan tugas-tugas ini selanjutnya.

Topik

Pilih jenis titik akhir yang akan disiapkan untuk API Gateway API

Jenis titik akhir API mengacu pada nama host API. Jenis titik akhir API dapat dioptimalkan secara tepi, regional, atau pribadi, tergantung dari mana sebagian besar lalu lintas API Anda berasal.

Titik akhir API yang dioptimalkan tepi

Titik akhir API yang dioptimalkan tepi biasanya merutekan permintaan ke CloudFront Point of Presence (POP) terdekat, yang dapat membantu jika klien Anda didistribusikan secara geografis. Ini adalah tipe endpoint default untuk API Gateway REST API.

API yang dioptimalkan tepi menggunakan huruf besar pada nama header HTTP (misalnya,). `Cookie`

CloudFront mengurutkan cookie HTTP dalam urutan alami dengan nama cookie sebelum meneruskan permintaan ke asal Anda. Untuk informasi selengkapnya tentang cara CloudFront memproses cookie, lihat Caching Konten Berdasarkan Cookie.

Nama domain kustom apa pun yang Anda gunakan untuk API yang dioptimalkan tepi berlaku di semua wilayah.

Titik akhir API regional

Titik akhir API regional ditujukan untuk klien di wilayah yang sama. Ketika klien yang berjalan pada instans EC2 memanggil API di wilayah yang sama, atau ketika API dimaksudkan untuk melayani sejumlah kecil klien dengan permintaan tinggi, API regional mengurangi overhead koneksi.

Untuk API regional, nama domain kustom apa pun yang Anda gunakan khusus untuk wilayah tempat API digunakan. Jika Anda menerapkan API regional di beberapa wilayah, API tersebut dapat memiliki nama domain kustom yang sama di semua wilayah. Anda dapat menggunakan domain khusus bersama dengan Amazon Route 53 untuk melakukan tugas seperti perutean berbasis latensi. Lihat informasi yang lebih lengkap di the section called “Menyiapkan nama domain kustom regional” dan the section called “Membuat nama domain kustom yang dioptimalkan tepi”.

Titik akhir API regional meneruskan semua nama header melalui apa adanya.

Titik akhir API pribadi

Titik akhir API pribadi adalah titik akhir API yang hanya dapat diakses dari Amazon Virtual Private Cloud (VPC) menggunakan titik akhir VPC antarmuka, yang merupakan antarmuka jaringan titik akhir (ENI) yang Anda buat di VPC Anda. Untuk informasi selengkapnya, lihat the section called “API Privat”.

Titik akhir API pribadi meneruskan semua nama header melalui apa adanya.

Mengubah jenis titik akhir API publik atau pribadi di API Gateway

Mengubah tipe titik akhir API mengharuskan Anda memperbarui konfigurasi API. Anda dapat mengubah jenis API yang ada menggunakan konsol API Gateway, theAWS CLI, atau AWS SDK untuk API Gateway. Jenis titik akhir tidak dapat diubah lagi hingga perubahan saat ini selesai, tetapi API Anda akan tersedia.

Perubahan tipe endpoint berikut didukung:

- Dari yang dioptimalkan tepi ke regional atau pribadi
- Dari regional hingga yang dioptimalkan tepi atau pribadi
- Dari pribadi ke regional

Anda tidak dapat mengubah API pribadi menjadi API yang dioptimalkan tepi.

Jika Anda mengubah API publik dari yang dioptimalkan tepi ke regional atau sebaliknya, perhatikan bahwa API yang dioptimalkan tepi mungkin memiliki perilaku yang berbeda dari API regional. Misalnya, API yang dioptimalkan tepi menghapus header. `Content-MD5` Nilai hash MD5 apa pun yang diteruskan ke backend dapat dinyatakan dalam parameter string permintaan atau properti tubuh. Namun, API regional meneruskan header ini, meskipun mungkin memetakan ulang nama header ke beberapa nama lain. Memahami perbedaan akan membantu Anda memutuskan cara memperbarui API yang dioptimalkan tepi ke API regional atau dari API regional ke yang dioptimalkan tepi.

Topik

- Menggunakan konsol API Gateway untuk mengubah jenis titik akhir API
- Gunakan AWS CLI untuk mengubah tipe titik akhir API

Menggunakan konsol API Gateway untuk mengubah jenis titik akhir API

Untuk mengubah jenis titik akhir API API Anda, lakukan salah satu set langkah berikut:

Untuk mengonversi titik akhir publik dari Regional atau yang dioptimalkan tepi dan sebaliknya

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih REST API.
3. Pilih setelan API.
4. Di bagian detail API, pilih Edit.
5. Untuk jenis endpoint API, pilih Edge-optimized atau Regional.
6. Pilih Save changes (Simpan perubahan).
7. Menerapkan ulang API Anda sehingga perubahan akan berlaku.

Untuk mengonversi titik akhir pribadi ke titik akhir regional

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih REST API.
3. Edit kebijakan sumber daya untuk API Anda untuk menghapus penyebutan VPC atau titik akhir VPC sehingga panggilan API dari luar VPC Anda maupun di dalam VPC Anda akan berhasil.
4. Pilih setelan API.
5. Di bagian detail API, pilih Edit.
6. Untuk jenis titik akhir API, pilih Regional.
7. Pilih Simpan perubahan untuk memulai pembaruan.
8. Hapus kebijakan sumber daya dari API Anda.
9. Menerapkan ulang API Anda sehingga perubahan akan berlaku.

Gunakan AWS CLI untuk mengubah tipe titik akhir API

Untuk menggunakan AWS CLI untuk memperbarui API yang dioptimalkan tepi yang ID API-nya*{api-id}*, panggil update-rest-apisebagai berikut:

```
aws apigateway update-rest-api \
    --rest-api-id {api-id} \
```

```
    --patch-operations op=replace,path=/endpointConfiguration/types/EDGE,value=REGIONAL
```

Respons yang berhasil memiliki kode status `200 OK` dan muatan yang mirip dengan yang berikut ini:

```
{

    "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"
}
```

Sebaliknya, perbarui API regional ke API yang dioptimalkan tepi sebagai berikut:

```
aws apigateway update-rest-api \
    --rest-api-id {api-id} \
    --patch-operations op=replace,path=/endpointConfiguration/types/REGIONAL,value=EDGE
```

Karena put-rest-apiuntuk memperbarui definisi API, itu tidak berlaku untuk memperbarui jenis titik akhir API.

Inisialisasi penyiapan REST API di API Gateway

Anda dapat membuat REST API menggunakan konsol API Gateway, API Gateway REST API, AWS SDK untuk API Gateway, dan. AWS Command Line Interface

Saat Anda membuat REST API menggunakan API Gateway REST API, AWS SDK untuk API Gateway, atau konfigurasi defaultnya adalah API yang dioptimalkan tepi. AWS Command Line Interface Untuk informasi selengkapnya tentang jenis titik akhir API, lihatthe section called “Pilih jenis titik akhir API”.

Saat menerapkan API ke sebuah panggung, API Gateway akan membuat URL default untuk API Anda. Untuk informasi selengkapnya tentang URL default, lihatthe section called “Menerapkan REST API”. Anda dapat menetapkan nama domain kustom (misalnya,`apis.example.com`) sebagai nama host API dan memanggil API dengan URL dasar `https://apis.example.com/myApi` format. Untuk informasi selengkapnya tentang nama domain kustom, lihatthe section called “Nama domain kustom”.

Kami menyarankan Anda menggunakan salah satu contoh berikut untuk mempelajari cara membuat REST API.

Topik

- Menyiapkan API menggunakan konsol API Gateway
- Siapkan API yang dioptimalkan tepi menggunakan perintah AWS CLI
- Siapkan API yang dioptimalkan tepi menggunakan AWS SDK untuk Node.js
- Siapkan API yang dioptimalkan tepi dengan mengimpor definisi OpenAPI
- Menyiapkan API regional di API Gateway

Menyiapkan API menggunakan konsol API Gateway

Kami menyarankan Anda memilih dari tutorial berikut untuk mempelajari cara membuat REST API Gateway menggunakan konsol REST API Gateway.

Untuk membuat REST API yang meneruskan acara ke fungsi Lambda, pilih. the section called “Memulai dengan konsol REST API”

Untuk membuat REST API tempat Anda mengonfigurasi payload permintaan integrasi ke fungsi Lambda, pilih. the section called “Tutorial: Membangun API dengan integrasi non-proxy Lambda”

Untuk membuat REST API yang memiliki integrasi dengan titik akhir HTTP, pilihthe section called “Tutorial: Membangun REST API dengan integrasi proxy HTTP”.

Untuk membuat REST API tempat Anda mengonfigurasi permintaan integrasi ke titik akhir HTTP, pilihthe section called “Tutorial: Membangun API dengan integrasi non-proxy HTTP”.

Untuk mengimpor API contoh, pilihthe section called “Tutorial: Buat REST API dengan mengimpor contoh”.

Atau, Anda dapat menyiapkan API dengan menggunakan fitur API Gateway Import API untuk mengunggah definisi API eksternal, seperti yang dinyatakan dalam OpenAPI 2.0 dengan definisi API. Bekerja dengan ekstensi API Gateway ke OpenAPI Contoh yang diberikan Tutorial: Buat REST API dengan mengimpor contoh menggunakan fitur API Impor.

Siapkan API yang dioptimalkan tepi menggunakan perintah AWS CLI

Menyiapkan API menggunakan AWS CLI membutuhkan bekerja dengan `put-integration-response` perintah `create-rest-apiget-resources`, `create-resource` atau`put-method`,

put-method-responseput-integration,, dan. Prosedur berikut menunjukkan cara bekerja dengan AWS CLI perintah ini untuk membuat PetStore API sederhana dari jenis HTTP integrasi.

Untuk membuat PetStore API sederhana menggunakan AWS CLI

1. Panggil `create-rest-api` perintah untuk mengatur `RestApi` di wilayah tertentu (`us-west-2`).

   ```
   aws apigateway create-rest-api --name 'Simple PetStore (AWS CLI)' --region us-west-2
   ```

   Berikut ini adalah output dari perintah ini:

   ```
   {
       "id": "vaz7da96z6",
       "name": "Simple PetStore (AWS CLI)",
       "createdDate": "2022-12-15T08:07:04-08:00",
       "apiKeySource": "HEADER",
       "endpointConfiguration": {
           "types": [
               "EDGE"
           ]
       },
       "disableExecuteApiEndpoint": false
   }
   ```

   Perhatikan `id` kembalinya yang baru dibuat`RestApi`. Anda memerlukannya untuk menyiapkan bagian lain dari API.

2. Panggil `get-resources` perintah untuk mengambil pengenal sumber daya root dari file. `RestApi`

   ```
   aws apigateway get-resources --rest-api-id vaz7da96z6 --region us-west-2
   ```

   Berikut ini adalah output dari perintah ini:

   ```
   {
       "items": [
           {
               "id": "begaltmsm8",
               "path": "/"
   ```

```
        }
    ]
}
```

Perhatikan sumber daya rootId. Anda memerlukannya untuk mulai menyetel pohon sumber daya API dan mengonfigurasi metode dan integrasi.

3. Panggil `create-resource` perintah untuk menambahkan sumber daya anak (`pets`) di bawah sumber daya root (`begaltmsm8`):

```
aws apigateway create-resource --rest-api-id vaz7da96z6 \
      --region us-west-2 \
      --parent-id begaltmsm8 \
      --path-part pets
```

Berikut ini adalah output dari perintah ini:

```
{
    "id": "6sxz2j",
    "parentId": "begaltmsm8",
    "pathPart": "pets",
    "path": "/pets"
}
```

Untuk menambahkan sumber daya anak di bawah root, Anda menentukan sumber daya root `Id` sebagai nilai `parentId` properti. Demikian pula, untuk menambahkan sumber daya anak di bawah `pets` sumber daya, Anda mengulangi langkah sebelumnya sambil mengganti `parent-id` nilai dengan sumber daya: `pets` `id` `6sxz2j`

```
aws apigateway create-resource --rest-api-id vaz7da96z6 \
      --region us-west-2 \
      --parent-id 6sxz2j \
      --path-part '{petId}'
```

Untuk membuat bagian jalur menjadi parameter jalur, lampirkan dalam sepasang tanda kurung keriting. Jika berhasil, perintah ini mengembalikan respons berikut:

```
{
    "id": "rjkmth",
    "parentId": "6sxz2j",
```

```
    "path": "/pets/{petId}",
    "pathPart": "{petId}"
}
```

Sekarang setelah Anda membuat dua sumber daya: `/pets` (`6sxz2j`) dan `/pets/{petId}` (`rjkmth`), Anda dapat melanjutkan untuk mengatur metode pada mereka.

4. Panggil `put-method` perintah untuk menambahkan metode `GET` HTTP pada `/pets` sumber daya. Ini membuat API `Method GET /pets` dengan akses terbuka, mereferensikan `/pets` sumber daya dengan nilai ID-nya. `6sxz2j`

```
aws apigateway put-method --rest-api-id vaz7da96z6 \
       --resource-id 6sxz2j \
       --http-method GET \
       --authorization-type "NONE" \
       --region us-west-2
```

Berikut ini adalah output sukses dari perintah ini:

```
{
    "httpMethod": "GET",
    "authorizationType": "NONE",
    "apiKeyRequired": false
}
```

Metode ini untuk akses terbuka karena `authorization-type` diatur ke`NONE`. Untuk mengizinkan hanya pengguna yang diautentikasi untuk memanggil metode, Anda dapat menggunakan peran dan kebijakan IAM, otorisasi Lambda (sebelumnya dikenal sebagai otorisasi khusus), atau kumpulan pengguna Amazon Cognito. Untuk informasi selengkapnya, lihat the section called “Kontrol akses”.

Untuk mengaktifkan akses baca ke `/pets/{petId}` resource (`rjkmth`), tambahkan metode `GET` HTTP di atasnya untuk membuat API `Method GET /pets/{petId}` sebagai berikut.

```
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
```

Berikut ini adalah output sukses dari perintah ini:

```
{
    "httpMethod": "GET",
    "authorizationType": "NONE",
    "apiKeyRequired": false,
    "requestParameters": {
        "method.request.path.petId": true
    }
}
```

Perhatikan bahwa parameter jalur permintaan metode `petId` harus ditentukan sebagai parameter permintaan yang diperlukan untuk nilai yang disetel secara dinamis untuk dipetakan ke parameter permintaan integrasi yang sesuai dan diteruskan ke backend.

5. Panggil `put-method-response` perintah untuk mengatur respons 200 OK dari `GET /pets` metode, menentukan `/pets` sumber daya dengan nilai ID-nya. `6sxz2j`

```
aws apigateway put-method-response --rest-api-id vaz7da96z6 \
       --resource-id 6sxz2j --http-method GET \
       --status-code 200  --region us-west-2
```

Berikut ini adalah output dari perintah ini:

```
{
    "statusCode": "200"
}
```

Demikian pula, untuk mengatur respons 200 OK dari `GET /pets/{petId}` metode, lakukan hal berikut, tentukan `/pets/{petId}` sumber daya dengan nilai ID sumber dayanya: `rjkmth`

```
aws apigateway put-method-response --rest-api-id vaz7da96z6 \
       --resource-id rjkmth --http-method GET \
       --status-code 200  --region us-west-2
```

Setelah menyiapkan antarmuka klien sederhana untuk API, Anda dapat melanjutkan untuk mengatur integrasi metode API dengan backend.

6. Panggil `put-integration` perintah untuk mengatur `Integration` dengan titik akhir HTTP tertentu untuk `GET /pets` metode ini. Sumber `/pets` daya diidentifikasi oleh Id sumber dayanya`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
```

Berikut ini adalah output dari perintah ini:

```
{
    "type": "HTTP",
    "httpMethod": "GET",
    "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
    "connectionType": "INTERNET",
    "passthroughBehavior": "WHEN_NO_MATCH",
    "timeoutInMillis": 29000,
    "cacheNamespace": "6sxz2j",
    "cacheKeyParameters": []
}
```

Perhatikan bahwa integrasi `uri` `http://petstore-demo-endpoint.execute-api.com/petstore/pets` menentukan endpoint integrasi metode. `GET /pets`

Demikian pula, Anda membuat permintaan integrasi untuk `GET /pets/{petId}` metode sebagai berikut:

```
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
```

Di sini, titik akhir integrasi, `uri` dari`http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}`, juga menggunakan parameter jalur (`id`). Nilainya dipetakan dari parameter jalur permintaan metode yang sesuai dari. `{petId}` Pemetaan didefinisikan sebagai bagian dari. `request-parameters` Jika pemetaan ini tidak didefinisikan di sini, klien mendapatkan respons kesalahan saat mencoba memanggil metode.

Berikut ini adalah output dari perintah ini:

```
{
    "type": "HTTP",
    "httpMethod": "GET",
    "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}",
    "connectionType": "INTERNET",
    "requestParameters": {
        "integration.request.path.id": "method.request.path.petId"
    },
    "passthroughBehavior": "WHEN_NO_MATCH",
    "timeoutInMillis": 29000,
    "cacheNamespace": "rjkmth",
    "cacheKeyParameters": []
}
```

7. Panggil `put-integration-response` perintah untuk membuat `GET /pets` metode `IntegrationResponse` yang terintegrasi dengan backend HTTP.

```
aws apigateway put-integration-response --rest-api-id vaz7da96z6 \
       --resource-id 6sxz2j --http-method GET \
       --status-code 200 --selection-pattern ""  \
       --region us-west-2
```

Berikut ini adalah output dari perintah ini:

```
{
    "statusCode": "200",
    "selectionPattern": ""
}
```

Demikian pula, panggil `put-integration-response` perintah berikut untuk membuat `IntegrationResponse` `GET /pets/{petId}` metode:

```
aws apigateway put-integration-response --rest-api-id vaz7da96z6 \
      --resource-id rjkmth --http-method GET
      --status-code 200 --selection-pattern ""
      --region us-west-2
```

Dengan langkah-langkah sebelumnya, Anda selesai menyiapkan API sederhana yang memungkinkan pelanggan Anda menanyakan hewan peliharaan yang tersedia di PetStore situs web dan untuk melihat hewan peliharaan individu dari pengenal tertentu. Untuk membuatnya dapat dipanggil oleh pelanggan Anda, Anda harus menerapkan API.

8. Menerapkan API ke `stage` panggung, misalnya, dengan memanggil`create-deployment`:

```
aws apigateway create-deployment --rest-api-id vaz7da96z6 \
       --region us-west-2 \
       --stage-name test \
       --stage-description 'Test stage' \
       --description 'First deployment'
```

Berikut ini adalah output dari perintah ini:

```
{
    "id": "ab1c1d",
    "description": "First deployment",
    "createdDate": "2022-12-15T08:44:13-08:00"
}
```

Anda dapat menguji API ini dengan mengetikkan `https://vaz7da96z6.execute-api.us-west-2.amazonaws.com/test/pets` URL di browser, dan menggantinya `vaz7da96z6` dengan pengenal API Anda. Output yang diharapkan harus sebagai berikut:

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

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

Untuk menguji `GET /pets/{petId}` metode ini, `https://vaz7da96z6.execute-api.us-west-2.amazonaws.com/test/pets/3` ketik browser. Anda harus menerima tanggapan berikut:

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

Siapkan API yang dioptimalkan tepi menggunakan AWS SDK untuk Node.js

Sebagai ilustrasi, kami menggunakan AWS SDK untuk Node.js untuk menjelaskan bagaimana Anda dapat menggunakan AWS SDK untuk membuat API Gateway API. Untuk informasi selengkapnya menggunakan AWS SDK, termasuk cara mengatur lingkungan pengembangan, lihat AWS SDK.

Menyiapkan API menggunakan AWS SDK untuk Node.js melibatkan pemanggilan `putIntegrationResponse` fungsi `createRestApigetResources`, `createResource` atau`putMethod`,`putMethodResponse`,`putIntegration`, dan.

Prosedur berikut memandu Anda melalui langkah-langkah penting untuk menggunakan perintah SDK ini untuk menyiapkan PetStore API sederhana yang mendukung `GET /pets` dan `GET /pets/{petId}` metode.

Untuk menyiapkan PetStore API sederhana menggunakan AWS SDK untuk Node.js

1. Buat instance SDK:

```
var AWS = require('aws-sdk');

AWS.config.region = 'us-west-2';
var apig = new AWS.APIGateway({apiVersion: '2015/07/09'});
```

2. Panggil `createRestApi` fungsi untuk mengatur `RestApi` entitas.

```
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);
 }
});
```

Fungsi mengembalikan output yang mirip dengan hasil sebagai berikut:

```
{
  id: 'iuo308uaq7',
  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: [ '*' ]
}
```

Pengidentifikasi API yang dihasilkan adalah`iuo308uaq7`. Anda perlu menyediakan ini untuk melanjutkan penyiapan API.

3. Panggil `getResources` fungsi untuk mengambil pengenal sumber daya root dari file. `RestApi`

```
apig.getResources({
 restApiId: 'iuo308uaq7'
}, function(err, data){
 if (!err) {
  console.log(data);
 } else {
  console.log('Get the root resource failed:\n', err);
 }
})
```

Fungsi ini mengembalikan output yang mirip dengan hasil sebagai berikut:

```
{
    "items": [
        {
            "path": "/",
            "id": "s4fb0trnk0"
        }
    ]
}
```

Pengidentifikasi sumber daya root adalah`s4fb0trnk0`. Ini adalah titik awal bagi Anda untuk membangun pohon sumber daya API, yang Anda lakukan selanjutnya.

4. Panggil `createResource` fungsi untuk menyiapkan `/pets` sumber daya untuk API, dengan menentukan pengenal sumber daya root (`s4fb0trnk0`) pada `parentId` properti.

```
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);
 }
})
```

Hasil yang berhasil adalah sebagai berikut:

```
{
    "path": "/pets",
    "pathPart": "pets",
    "id": "8sxa2j",
    "parentId": "s4fb0trnk0'"
}
```

Untuk mengatur `/pets/{petId}` sumber daya, panggil `createResource` fungsi berikut, tentukan `/pets` resource (`8sxa2j`) yang baru dibuat di `parentId` properti.

```
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);
 }
})
```

Hasil yang berhasil mengembalikan `id` nilai sumber daya yang baru dibuat:

```
{
    "path": "/pets/{petId}",
    "pathPart": "{petId}",
    "id": "au5df2",
    "parentId": "8sxa2j"
}
```

Sepanjang prosedur ini, Anda merujuk ke `/pets` sumber daya dengan menentukan ID sumber dayanya`8sxa2j`, dan `/pets/{petId}` sumber daya dengan menentukan ID sumber dayanya dari. `au5df2`

5. Panggil `putMethod` fungsi untuk menambahkan metode `GET` HTTP pada `/pets` resource (`8sxa2j`). Ini mengatur `GET /pets Method` dengan akses terbuka.

```
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);
 }
})
```

Fungsi ini mengembalikan output yang mirip dengan hasil sebagai berikut:

```
{
    "apiKeyRequired": false,
    "httpMethod": "GET",
    "authorizationType": "NONE"
}
```

Untuk menambahkan metode GET HTTP pada `/pets/{petId}` resource (`au5df2`), yang mengatur metode API `GET /pets/{petId}` dengan akses terbuka, panggil `putMethod` fungsi sebagai berikut.

```
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);
 }
})
```

Fungsi ini mengembalikan output yang mirip dengan hasil sebagai berikut:

```
{
    "apiKeyRequired": false,
    "httpMethod": "GET",
    "authorizationType": "NONE",
    "requestParameters": {
       "method.request.path.petId": true
    }
}
```

Anda perlu mengatur `requestParameters` properti seperti yang ditunjukkan pada contoh sebelumnya untuk memetakan dan meneruskan `petId` nilai yang disediakan klien ke backend.

6. Panggil `putMethodResponse` fungsi untuk mengatur respons metode untuk `GET /pets` metode tersebut.

```
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);
 }

})
```

Fungsi ini mengembalikan output yang mirip dengan hasil sebagai berikut:

```
{
    "statusCode": "200"
}
```

Untuk mengatur respons 200 OK dari `GET /pets/{petId}` metode, panggil `putMethodResponse` fungsi, tentukan `/pets/{petId}` resource identifier (`au5df2`) pada `resourceId` properti.

```
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. Panggil `putIntegration` fungsi untuk mengatur `Integration` dengan titik akhir HTTP tertentu untuk `GET /pets` metode, menyediakan `/pets` resource identifier (`8sxa2j`) pada properti. `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);
 }

})
```

Fungsi ini mengembalikan output serupa sebagai berikut:

```
{
    "httpMethod": "GET",
    "passthroughBehavior": "WHEN_NO_MATCH",
    "cacheKeyParameters": [],
    "type": "HTTP",
    "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
    "cacheNamespace": "8sxa2j"
}
```

Untuk mengatur integrasi `GET /pets/{petId}` metode dengan titik akhir HTTP `http://perstore-demo-endpoint.execute-api.com/pets/{id}` dari backend, panggil

`putIntegration` fungsi berikut, dengan menyediakan `/pets/{petId}` resource identifier () `au5df2` API pada properti. `parentId`

```
apig.putIntegration({
 restApiId: 'iuo308uaq7',
 resourceId: 'au5df2',
 httpMethod: 'GET',
 type: 'HTTP',
 integrationHttpMethod: 'GET',
 uri: 'http://perstore-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);
 }

})
```

Fungsi ini mengembalikan output yang berhasil mirip dengan berikut ini:

```
{
    "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. Panggil `putIntegrationResponse` fungsi untuk menyiapkan respons integrasi 200 OK untuk `GET /pets` metode tersebut, dengan menentukan `/pets` resource identifier (`8sxa2j`) API pada `resourceId` properti.

```
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);

})
```

Fungsi ini akan mengembalikan output yang mirip dengan hasil berikut:

```
{
    "selectionPattern": "",
    "statusCode": "200"
}
```

Untuk menyiapkan respons integrasi 200 OK dari `GET /pets/{petId}` metode, panggil `putIntegrationResponse` fungsi, tentukan `/pets/{petId}` resource identifier (`au5df2`) API pada `resourceId` properti.

```
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. Sebagai praktik yang baik, uji pemanggilan API sebelum menerapkannya. Untuk menguji pemanggilan `GET /pets` metode, panggil`testInvokeMethod`, dengan menentukan `/pets` resource identifier (`8sxa2j`) pada properti: `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);
 }
})
```

Untuk menguji pemanggilan `GET /pets/{petId}` metode, panggil`testInvokeMethod`, dengan menentukan `/pets/{petId}` resource identifier (`au5df2`) pada properti: `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. Terakhir, Anda dapat menerapkan API agar pelanggan Anda dapat menelepon.

```
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);
```

```
 }
})
```

Siapkan API yang dioptimalkan tepi dengan mengimpor definisi OpenAPI

Anda dapat menyiapkan API di API Gateway dengan menentukan definisi OpenAPI dari entitas API Gateway API yang sesuai dan mengimpor definisi OpenAPI ke dalam API Gateway.

Definisi OpenAPI berikut menjelaskan API sederhana, hanya mengekspos `GET /` metode yang terintegrasi dengan titik akhir HTTP PetStore situs web di backend, dan mengembalikan respons. `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",
            "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"
        }
      }
    }
  }
}
```

Prosedur berikut menjelaskan cara mengimpor definisi OpenAPI ini ke API Gateway menggunakan konsol API Gateway.

Untuk mengimpor definisi OpenAPI sederhana menggunakan konsol API Gateway

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih Buat API, lalu untuk REST API, pilih Impor.

3. Jika Anda menyimpan definisi OpenAPI sebelumnya dalam sebuah file, pilih Pilih file. Anda juga dapat menyalin definisi OpenAPI dan menempelkannya ke editor teks impor.
4. Untuk jenis titik akhir API, pilih Optimalisasi tepi.
5. Pilih Buat API untuk mengimpor definisi OpenAPI.

Untuk mengimpor definisi OpenAPI menggunakan AWS CLI, simpan definisi OpenAPI ke dalam file dan kemudian jalankan perintah berikut, dengan asumsi bahwa Anda menggunakan wilayah `us-west-2` dan jalur file OpenAPI absolut adalah: `file:///path/to/API_OpenAPI_template.json`

```
aws apigateway import-rest-api --body 'file:///path/to/API_OpenAPI_template.json' --
region us-west-2
```

Menyiapkan API regional di API Gateway

Ketika permintaan API sebagian besar berasal dari instans atau layanan EC2 dalam wilayah yang sama dengan API yang digunakan, titik akhir API regional biasanya akan menurunkan latensi koneksi dan direkomendasikan untuk skenario tersebut.

> **Note**
>
> Dalam kasus di mana klien API tersebar secara geografis, mungkin masih masuk akal untuk menggunakan titik akhir API regional, bersama dengan CloudFront distribusi Amazon Anda sendiri untuk memastikan bahwa API Gateway tidak mengaitkan API dengan distribusi yang dikendalikan layanan. CloudFront Untuk informasi selengkapnya tentang kasus penggunaan ini, lihat Bagaimana cara menyiapkan API Gateway dengan CloudFront distribusi saya sendiri? .

Untuk membuat API regional, Anda mengikuti langkah-langkah dalam membuat API yang dioptimalkan tepi, tetapi harus secara eksplisit menetapkan `REGIONAL` type sebagai satu-satunya opsi EndpointConfiguration API.

Berikut ini, kami menunjukkan cara membuat API regional menggunakan konsol API GatewayAWS CLI, dan AWS SDK untuk Javascript untuk Node.js.

Topik

- Membuat API regional menggunakan konsol API Gateway

- Buat API regional menggunakan AWS CLI
- Membuat API regional menggunakan AWS SDK untuk JavaScript
- Uji API regional

Membuat API regional menggunakan konsol API Gateway

Untuk membuat API regional menggunakan konsol API Gateway

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Lakukan salah satu dari berikut:
   - Untuk membuat API pertama Anda, untuk REST API, pilih Build.
   - Jika Anda pernah membuat API sebelumnya, pilih Create API, lalu pilih Build for REST API.
3. Untuk Nama, masukkan nama.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Tetap tetapkan jenis endpoint API ke Regional.
6. Pilih Buat API.

Buat API regional menggunakan AWS CLI

Untuk membuat API regional menggunakanAWS CLI, panggil `create-rest-api` perintah:

```
aws apigateway create-rest-api \
        --name 'Simple PetStore (AWS CLI, Regional)' \
        --description 'Simple regional PetStore API' \
        --region us-west-2 \
        --endpoint-configuration '{ "types": ["REGIONAL"] }'
```

Respons yang berhasil mengembalikan payload yang mirip dengan berikut ini:

```
{
    "createdDate": "2017-10-13T18:41:39Z",
    "description": "Simple regional PetStore API",
    "endpointConfiguration": {
        "types": "REGIONAL"
    },
    "id": "0qzs2sy7bh",
    "name": "Simple PetStore (AWS CLI, Regional)"
```

```
}
```

Mulai sekarang, Anda dapat mengikuti instruksi yang sama yang diberikan the section called "Siapkan API yang dioptimalkan tepi menggunakan perintah AWS CLI" untuk menyiapkan metode dan integrasi untuk API ini.

Membuat API regional menggunakan AWS SDK untuk JavaScript

Untuk membuat API regional, gunakan AWS SDK untuk 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);
 }
});
```

Respons yang berhasil mengembalikan payload yang mirip dengan berikut ini:

```
{
    "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)"
}
```

Setelah menyelesaikan langkah-langkah sebelumnya, Anda dapat mengikuti petunjuk the section called "Siapkan API yang dioptimalkan tepi menggunakan AWS SDK untuk Node.js" untuk menyiapkan metode dan integrasi untuk API ini.

Uji API regional

Setelah di-deploy, nama host URL default API regional adalah dengan format berikut:

```
{restapi-id}.execute-api.{region}.amazonaws.com
```

URL dasar untuk menjalankan API adalah seperti berikut:

```
https://{restapi-id}.execute-api.{region}.amazonaws.com/{stage}
```

Dengan asumsi Anda mengatur `GET /pets` dan `GET /pets/{petId}` metode dalam contoh ini, Anda dapat menguji API dengan mengetikkan URL berikut di browser:

```
https://0qzs2sy7bh.execute-api.us-west-2.amazonaws.com/test/pets
```

and

```
https://0qzs2sy7bh.execute-api.us-west-2.amazonaws.com/test/pets/1
```

Atau, Anda dapat menggunakan perintah cURL:

```
curl -X GET https://0qzs2sy7bh.execute-api.us-west-2.amazonaws.com/test/pets
```

and

```
curl -X GET https://0qzs2sy7bh.execute-api.us-west-2.amazonaws.com/test/pets/2
```

Siapkan metode REST API di API Gateway

Di API Gateway, metode API mewujudkan permintaan metode dan respons metode. Anda menyiapkan metode API untuk menentukan apa yang harus atau harus dilakukan klien untuk mengirimkan permintaan untuk mengakses layanan di backend dan untuk menentukan tanggapan yang diterima klien sebagai imbalannya. Untuk masukan, Anda dapat memilih parameter permintaan metode, atau muatan yang berlaku, agar klien dapat menyediakan data yang diperlukan atau opsional pada waktu berjalan. Untuk output, Anda menentukan kode status respons metode, header, dan badan yang berlaku sebagai target untuk memetakan data respons backend, sebelum dikembalikan ke klien. Untuk membantu pengembang klien memahami perilaku dan format input dan

output API Anda, Anda dapat mendokumentasikan API Anda dan memberikan pesan kesalahan yang tepat untuk permintaan yang tidak valid.

Permintaan metode API adalah permintaan HTTP. Untuk menyiapkan permintaan metode, Anda mengonfigurasi metode HTTP (atau kata kerja), jalur ke sumber daya API, header, parameter string kueri yang berlaku. Anda juga mengonfigurasi payload ketika metode HTTP adalah`POST`,`PUT`, atau`PATCH`. Misalnya, untuk mengambil hewan peliharaan menggunakan API PetStore sampel, Anda menentukan permintaan metode API`GET /pets/{petId}`, di mana `{petId}` merupakan parameter jalur yang dapat mengambil nomor pada waktu proses.

```
GET /pets/1
Host: apigateway.us-east-1.amazonaws.com
...
```

Jika klien menentukan jalur yang salah, misalnya, `/pet/1` atau `/pets/one` bukan`/pets/1`, pengecualian dilemparkan.

Respons metode API adalah respons HTTP dengan kode status yang diberikan. Untuk integrasi non-proxy, Anda harus menyiapkan respons metode untuk menentukan target pemetaan yang diperlukan atau opsional. Ini mengubah header atau badan respons integrasi ke header atau badan respons metode terkait. Pemetaan dapat sesederhana transformasi identitas yang melewati header atau badan melalui integrasi apa adanya. Misalnya, respons `200` metode berikut menunjukkan contoh passthrough dari respons integrasi yang berhasil apa adanya.

```
200 OK
Content-Type: application/json
...

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

Pada prinsipnya, Anda dapat menentukan respons metode yang sesuai dengan respons tertentu dari backend. Biasanya, ini melibatkan respons 2XX, 4XX, dan 5XX. Namun, ini mungkin tidak praktis, karena seringkali Anda mungkin tidak tahu sebelumnya semua tanggapan yang mungkin dikembalikan oleh backend. Dalam praktiknya, Anda dapat menetapkan satu respons metode sebagai default untuk menangani respons yang tidak diketahui atau tidak dipetakan dari backend. Ini adalah praktik yang baik untuk menetapkan respons 500 sebagai default. Bagaimanapun, Anda

harus menyiapkan setidaknya satu respons metode untuk integrasi non-proxy. Jika tidak, API Gateway mengembalikan respons kesalahan 500 ke klien bahkan ketika permintaan berhasil di backend.

Untuk mendukung SDK yang diketik dengan kuat, seperti Java SDK, untuk API Anda, Anda harus menentukan model data untuk input untuk permintaan metode, dan menentukan model data untuk output dari respons metode.

Topik

- Siapkan permintaan metode di API Gateway
- Siapkan respons metode di API Gateway
- Siapkan metode menggunakan konsol API Gateway

Siapkan permintaan metode di API Gateway

Menyiapkan permintaan metode melibatkan melakukan tugas-tugas berikut, setelah membuat RestApisumber daya:

1. Membuat API baru atau memilih entitas Sumber Daya API yang ada.
2. Membuat sumber daya Metode API yang merupakan kata kerja HTTP spesifik pada API `Resource` baru atau yang dipilih. Tugas ini dapat dibagi lagi menjadi sub tugas berikut:
   - Menambahkan metode HTTP ke permintaan metode
   - Mengkonfigurasi parameter permintaan
   - Mendefinisikan model untuk badan permintaan
   - Memberlakukan skema otorisasi
   - Mengaktifkan validasi permintaan

Anda dapat melakukan tugas-tugas ini menggunakan metode berikut:

- Konsol API Gateway
- AWS CLI perintah (create-resource dan put-method)
- AWS Fungsi SDK (misalnya, di Node.js, createResource dan putMethod)
- API Gateway REST API (sumber daya: buat dan metode:put).

Untuk contoh menggunakan alat ini, lihat Inisialisasi penyiapan REST API di API Gateway.

Topik

- Siapkan sumber daya API
- Siapkan metode HTTP
- Siapkan parameter permintaan metode
- Siapkan model permintaan metode
- Siapkan otorisasi permintaan metode
- Siapkan validasi permintaan metode

Siapkan sumber daya API

Di API Gateway API, Anda mengekspos sumber daya yang dapat dialamatkan sebagai pohon entitas Sumber Daya API, dengan sumber daya root (`/`) di bagian atas hierarki. Sumber daya root relatif terhadap URL dasar API, yang terdiri dari titik akhir API dan nama panggung. Di konsol API Gateway, URI dasar ini disebut sebagai URI Invoke dan ditampilkan di editor tahap API setelah API diterapkan.

Titik akhir API dapat berupa nama host default atau nama domain khusus. Nama host default adalah dari format berikut:

```
{api-id}.execute-api.{region}.amazonaws.com
```

Dalam format ini, *`{api-id}`* mewakili pengenal API yang dihasilkan oleh API Gateway. *`{region}`*Variabel mewakili AWS Wilayah (misalnya,`us-east-1`) yang Anda pilih saat membuat API. Nama domain kustom adalah nama yang ramah pengguna di bawah domain internet yang valid. Misalnya, jika Anda telah mendaftarkan domain internet`example.com`, salah satu dari `*.example.com` adalah nama domain kustom yang valid. Untuk informasi selengkapnya, lihat membuat nama domain kustom.

Untuk API PetStore sampel, sumber daya root (`/`) mengekspos toko hewan peliharaan. Sumber `/pets` daya mewakili koleksi hewan peliharaan yang tersedia di toko hewan peliharaan. Ini `/pets/{petId}` mengekspos hewan peliharaan individu dari pengenal yang diberikan ()`petId`. Parameter jalur `{petId}` adalah bagian dari parameter permintaan.

Untuk menyiapkan sumber daya API, Anda memilih sumber daya yang ada sebagai induknya, lalu membuat sumber daya turunan di bawah sumber daya induk ini. Anda mulai dengan sumber daya root sebagai induk, menambahkan sumber daya ke induk ini, menambahkan sumber daya lain ke sumber daya turunan ini sebagai induk baru, dan seterusnya, ke pengenal induknya. Kemudian Anda menambahkan sumber daya bernama ke induk.

Dengan AWS CLI, Anda dapat memanggil `get-resources` perintah untuk mengetahui sumber daya API mana yang tersedia:

```
aws apigateway get-resources --rest-api-id <apiId> \
                             --region <region>
```

Hasilnya adalah daftar sumber daya API yang tersedia saat ini. Untuk API PetStore sampel kami, daftar ini terlihat seperti berikut:

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

Setiap item mencantumkan pengidentifikasi sumber daya (`id`) dan, kecuali sumber daya root, induk langsungnya (`parentId`), serta nama sumber daya (`pathPart`). Sumber daya root istimewa karena tidak memiliki induk. Setelah memilih sumber daya sebagai induk, panggil perintah berikut untuk menambahkan sumber daya anak.

```
aws apigateway create-resource --rest-api-id <apiId> \
```

```
                                --region <region> \
                                --parent-id <parentId> \
                                --path-part <resourceName>
```

Misalnya, untuk menambahkan makanan hewan untuk dijual di PetStore situs web, tambahkan `food` sumber daya ke root (/) dengan menyetel `path-part` ke `food` dan `parent-id` kesvzr2028x8. Hasilnya terlihat seperti berikut:

```
{
    "path": "/food",
    "pathPart": "food",
    "id": "xdsvhp",
    "parentId": "svzr2028x8"
}
```

Menggunakan sumber daya proxy untuk merampingkan penyiapan API

Seiring pertumbuhan bisnis, PetStore pemilik dapat memutuskan untuk menambahkan makanan, mainan, dan barang-barang terkait hewan peliharaan lainnya untuk dijual. Untuk mendukung ini, Anda dapat menambahkan`/food`,`/toys`, dan sumber daya lainnya di bawah sumber daya root. Di bawah setiap kategori penjualan, Anda mungkin juga ingin menambahkan lebih banyak sumber daya, seperti`/food/{type}/{item}`,`/toys/{type}/{item}`, dll. Ini bisa membosankan. Jika Anda memutuskan untuk menambahkan lapisan tengah `{subtype}` ke jalur sumber daya untuk mengubah hierarki jalur menjadi`/food/{type}/{subtype}/{item}`,`/toys/{type}/{subtype}/{item}`, dll., Perubahan akan merusak pengaturan API yang ada. Untuk menghindari hal ini, Anda dapat menggunakan sumber daya proxy API Gateway untuk mengekspos sekumpulan sumber daya API sekaligus.

API Gateway mendefinisikan sumber daya proxy sebagai placeholder untuk sumber daya yang akan ditentukan saat permintaan dikirimkan. Sumber daya proxy diekspresikan oleh parameter jalur khusus`{proxy+}`, sering disebut sebagai parameter jalur serakah. +Tanda tersebut menunjukkan sumber daya anak mana pun yang ditambahkan padanya. `/parent/{proxy+}`Placeholder adalah singkatan dari sumber daya apa pun yang cocok dengan pola jalur. `/parent/*` Nama parameter jalur serakah,`proxy`, dapat diganti dengan string lain dengan cara yang sama seperti Anda memperlakukan nama parameter jalur biasa.

Dengan menggunakan AWS CLI, Anda memanggil perintah berikut untuk menyiapkan sumber daya proxy di bawah root (`/{proxy+}`):

```
aws apigateway create-resource --rest-api-id <apiId> \
```

```
                                --region <region> \
                                --parent-id <rootResourceId> \
                                --path-part {proxy+}
```

Hasilnya mirip dengan yang berikut:

```
{
    "path": "/{proxy+}",
    "pathPart": "{proxy+}",
    "id": "234jdr",
    "parentId": "svzr2028x8"
}
```

Untuk contoh `PetStore` API, Anda dapat menggunakan `/{proxy+}` untuk mewakili `/pets` dan/`pets/{petId}`. Sumber daya proxy ini juga dapat mereferensikan sumber daya lain (yang ada atau to-be-added)`/food/{type}/{item}`, seperti`/toys/{type}/{item}`,, dll., Atau`/food/{type}/{subtype}/{item}`,`/toys/{type}/{subtype}/{item}`, dll. Pengembang backend menentukan hierarki sumber daya dan pengembang klien bertanggung jawab untuk memahaminya. API Gateway hanya meneruskan apa pun yang dikirimkan klien ke backend.

API dapat memiliki lebih dari satu sumber daya proxy. Misalnya, sumber daya proxy berikut diizinkan dalam API.

```
/{proxy+}
/parent/{proxy+}
/parent/{child}/{proxy+}
```

Ketika sumber daya proxy memiliki saudara kandung non-proxy, sumber daya saudara dikecualikan dari representasi sumber daya proxy. Untuk contoh sebelumnya, `/{proxy+}` mengacu pada sumber daya apa pun di bawah sumber daya root kecuali sumber daya. `/parent[/*]` Dengan kata lain, permintaan metode terhadap sumber daya tertentu lebih diutamakan daripada permintaan metode terhadap sumber daya generik pada tingkat hierarki sumber daya yang sama.

Sumber daya proxy tidak dapat memiliki sumber daya anak. Sumber daya API apa pun `{proxy+}` setelahnya berlebihan dan ambigu. Sumber daya proxy berikut tidak diizinkan dalam API.

```
/{proxy+}/child
/parent/{proxy+}/{child}
/parent/{child}/{proxy+}/{grandchild+}
```

Siapkan metode HTTP

Permintaan metode API dienkapsulasi oleh sumber daya Metode API Gateway. Untuk mengatur permintaan metode, Anda harus terlebih dahulu membuat instance `Method` sumber daya, menyetel setidaknya metode HTTP dan jenis otorisasi pada metode.

Terkait erat dengan sumber daya proxy, API Gateway mendukung metode HTTP`ANY`. `ANY`Metode ini mewakili setiap metode HTTP yang akan disediakan pada waktu berjalan. Ini memungkinkan Anda untuk menggunakan penyiapan metode API tunggal untuk semua metode HTTP yang didukung`DELETE`,`GET`,`HEAD`,`OPTIONS`,`PATCH`,`POST`, dan`PUT`.

Anda dapat mengatur `ANY` metode pada sumber daya non-proxy juga. Menggabungkan `ANY` metode dengan sumber daya proxy, Anda mendapatkan penyiapan metode API tunggal untuk semua metode HTTP yang didukung terhadap sumber daya API apa pun. Selain itu, backend dapat berkembang tanpa merusak pengaturan API yang ada.

Sebelum menyiapkan metode API, pertimbangkan siapa yang dapat memanggil metode tersebut. Atur jenis otorisasi sesuai dengan rencana Anda. Untuk akses terbuka, atur ke`NONE`. Untuk menggunakan izin IAM, atur jenis otorisasi ke. `AWS_IAM` Untuk menggunakan fungsi otorisasi Lambda, setel properti ini ke. `CUSTOM` Untuk menggunakan kumpulan pengguna Amazon Cognito, setel jenis otorisasi ke. `COGNITO_USER_POOLS`

AWS CLI Perintah berikut menunjukkan cara membuat permintaan metode `ANY` kata kerja terhadap sumber daya tertentu (`6sxz2j`), menggunakan izin IAM untuk mengontrol aksesnya.

```
aws apigateway put-method --rest-api-id vaz7da96z6 \
       --resource-id 6sxz2j \
       --http-method ANY \
       --authorization-type AWS_IAM \
       --region us-west-2
```

Untuk membuat permintaan metode API dengan jenis otorisasi yang berbeda, lihatthe section called “Siapkan otorisasi permintaan metode”.

Siapkan parameter permintaan metode

Parameter permintaan metode adalah cara bagi klien untuk menyediakan data input atau konteks eksekusi yang diperlukan untuk menyelesaikan permintaan metode. Parameter metode dapat berupa parameter jalur, header, atau parameter string kueri. Sebagai bagian dari pengaturan permintaan metode, Anda harus mendeklarasikan parameter permintaan yang diperlukan agar tersedia untuk

klien. Untuk integrasi non-proxy, Anda dapat menerjemahkan parameter permintaan ini ke formulir yang kompatibel dengan persyaratan backend.

Misalnya, untuk permintaan `GET /pets/{petId}` metode, variabel `{petId}` jalur adalah parameter permintaan yang diperlukan. Anda dapat mendeklarasikan parameter jalur ini saat memanggil `put-method` perintah. AWS CLI Ini diilustrasikan sebagai berikut:

```
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
```

Jika parameter tidak diperlukan, Anda dapat mengaturnya ke `false` dalam`request-parameters`. Misalnya, jika `GET /pets` metode menggunakan parameter string kueri opsional`type`, dan parameter header opsional`breed`, Anda dapat mendeklarasikannya menggunakan perintah CLI berikut, dengan asumsi bahwa sumber daya adalah: `/pets` `id` `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
```

Alih-alih formulir singkat ini, Anda dapat menggunakan string JSON untuk mengatur nilai`request-parameters`:

```
'{"method.request.querystring.type":false,"method.request.header.breed":false}'
```

Dengan pengaturan ini, klien dapat menanyakan hewan peliharaan berdasarkan jenis:

```
GET /pets?type=dog
```

Dan klien dapat menanyakan anjing-anjingnya dari jenis pudel sebagai berikut:

```
GET /pets?type=dog
breed:poodle
```

Untuk informasi tentang cara memetakan parameter permintaan metode ke parameter permintaan integrasi, lihatthe section called “Integrasi”.

Siapkan model permintaan metode

Untuk metode API yang dapat mengambil data input dalam payload, Anda dapat menggunakan model. Sebuah model dinyatakan dalam skema JSON draft 4 dan menjelaskan struktur data dari badan permintaan. Dengan model, klien dapat menentukan bagaimana membangun payload permintaan metode sebagai input. Lebih penting lagi, API Gateway menggunakan model untuk memvalidasi permintaan, menghasilkan SDK, dan menginisialisasi template pemetaan untuk menyiapkan integrasi di konsol API Gateway. Untuk informasi tentang cara membuat model, lihat Memahami model data.

Tergantung pada jenis konten, payload metode dapat memiliki format yang berbeda. Sebuah model diindeks terhadap jenis media dari muatan yang diterapkan. API Gateway menggunakan header `Content-Type` permintaan untuk menentukan jenis konten. Untuk mengatur model permintaan metode, tambahkan pasangan kunci-nilai "*<media-type>*":"*<model-name>*" format ke `requestModels` peta saat memanggil perintah. AWS CLI `put-method`

Untuk menggunakan model yang sama terlepas dari jenis konten, tentukan `$default` sebagai kunci.

Misalnya, untuk menyetel model pada payload JSON dari permintaan `POST /pets` metode API PetStore contoh, Anda dapat memanggil perintah berikut: AWS CLI

```
aws apigateway put-method \
       --rest-api-id vaz7da96z6 \
       --resource-id 6sxz2j \
       --http-method POST \
       --authorization-type "NONE" \
       --region us-west-2 \
       --request-models '{"application/json":"petModel"}'
```

Di sini, `petModel` adalah nilai `name` properti dari `Model`sumber daya yang menggambarkan hewan peliharaan. Definisi skema yang sebenarnya dinyatakan sebagai nilai string JSON dari `schema`properti sumber daya. `Model`

Di Java, atau SDK lain yang diketik kuat, dari API, data input dilemparkan sebagai `petModel` kelas yang berasal dari definisi skema. Dengan model permintaan, data input dalam SDK yang dihasilkan dilemparkan ke `Empty` kelas, yang berasal dari `Empty` model default. Dalam hal ini, klien tidak dapat membuat instance kelas data yang benar untuk memberikan input yang diperlukan.

Siapkan otorisasi permintaan metode

Untuk mengontrol siapa yang dapat memanggil metode API, Anda dapat mengonfigurasi jenis otorisasi pada metode. Anda dapat menggunakan jenis ini untuk memberlakukan salah satu otorisasi yang didukung, termasuk peran dan kebijakan IAM (`AWS_IAM`), kumpulan pengguna Amazon Cognito ()`COGNITO_USER_POOLS`, atau pemberi otorisasi Lambda (). `CUSTOM`

Untuk menggunakan izin IAM untuk mengotorisasi akses ke metode API, setel properti `authorization-type` input ke. **`AWS_IAM`** Saat Anda menyetel opsi ini, API Gateway memverifikasi tanda tangan pemanggil berdasarkan kredensional pemanggil. Jika pengguna terverifikasi memiliki izin untuk memanggil metode, ia menerima permintaan. Jika tidak, ia menolak permintaan dan pemanggil menerima respons kesalahan yang tidak sah. Panggilan ke metode tidak berhasil kecuali pemanggil memiliki izin untuk memanggil metode API. Kebijakan IAM berikut memberikan izin kepada pemanggil untuk memanggil metode API apa pun yang dibuat dalam metode yang sama: Akun AWS

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "execute-api:Invoke"
            ],
            "Resource": "arn:aws:execute-api:*:*:*"
        }
    ]
}
```

Untuk informasi selengkapnya, lihat the section called “Gunakan izin IAM”.

Saat ini, Anda hanya dapat memberikan kebijakan ini kepada pengguna, grup, dan peran dalam pemilik API Akun AWS. Pengguna dari yang berbeda Akun AWS dapat memanggil metode API hanya jika diizinkan untuk mengambil peran dalam pemilik API Akun AWS dengan izin yang diperlukan untuk memanggil `execute-api:Invoke` tindakan. Untuk informasi tentang izin lintas akun, lihat Menggunakan Peran IAM.

Anda dapat menggunakan AWS CLI, AWS SDK, atau klien REST API, seperti Postman, yang mengimplementasikan Signature Version 4 Signature Signature.

Untuk menggunakan otorisasi Lambda untuk mengotorisasi akses ke metode API, setel properti `authorization-type` input ke `CUSTOM` dan setel properti `authorizer-id`input ke nilai properti dari otorisasi Lambda yang sudah ada. `id` Authorizer Lambda yang direferensikan dapat dari `TOKEN` tipe atau. `REQUEST` Untuk informasi tentang membuat otorisasi Lambda, lihat. the section called "Gunakan otorisasi Lambda"

Untuk menggunakan kumpulan pengguna Amazon Cognito untuk mengotorisasi akses ke metode API, setel properti `authorization-type` input ke `COGNITO_USER_POOLS` dan setel properti `authorizer-id`input ke nilai `id`properti `COGNITO_USER_POOLS` otorisasi yang telah dibuat. Untuk informasi tentang membuat otorisasi kumpulan pengguna Amazon Cognito, lihat. the section called "Gunakan kumpulan pengguna Amazon Cognito sebagai otorisasi untuk REST API"

Siapkan validasi permintaan metode

Anda dapat mengaktifkan validasi permintaan saat menyiapkan permintaan metode API. Anda harus terlebih dahulu membuat validator permintaan:

```
aws apigateway create-request-validator \
    --rest-api-id 7zw9uyk9kl \
    --name bodyOnlyValidator \
    --validate-request-body  \
    --no-validate-request-parameters
```

Perintah CLI ini menciptakan validator permintaan body-only. Contoh output adalah sebagai berikut:

```
{
    "validateRequestParameters": false,
    "validateRequestBody": true,
    "id": "jgpyy6",
    "name": "bodyOnlyValidator"
}
```

Dengan validator permintaan ini, Anda dapat mengaktifkan validasi permintaan sebagai bagian dari pengaturan permintaan metode:

```
aws apigateway put-method \
    --rest-api-id 7zw9uyk9kl
    --region us-west-2
    --resource-id xdsvhp
    --http-method PUT
    --authorization-type "NONE"
```

```
    --request-parameters '{"method.request.querystring.type": false,
"method.request.querystring.page":false}'
    --request-models '{"application/json":"petModel"}'
    --request-validator-id jgpyy6
```

Untuk dimasukkan dalam validasi permintaan, parameter permintaan harus dinyatakan sebagai wajib. Jika parameter string kueri untuk halaman digunakan dalam validasi permintaan, `request-parameters` peta contoh sebelumnya harus ditentukan sebagai. `'{"method.request.querystring.type": false, "method.request.querystring.page":true}'`

Siapkan respons metode di API Gateway

Respons metode API merangkum output dari permintaan metode API yang akan diterima klien. Data output mencakup kode status HTTP, beberapa header, dan mungkin badan.

Dengan integrasi non-proxy, parameter dan badan respons yang ditentukan dapat dipetakan dari data respons integrasi terkait atau dapat diberikan nilai statis tertentu sesuai dengan pemetaan. Pemetaan ini ditentukan dalam respons integrasi. Pemetaan dapat menjadi transformasi identik yang melewati respons integrasi melalui apa adanya.

Dengan integrasi proxy, API Gateway meneruskan respons backend ke respons metode secara otomatis. Anda tidak perlu menyiapkan respons metode API. Namun, dengan integrasi proxy Lambda, fungsi Lambda harus mengembalikan hasil format keluaran ini agar API Gateway berhasil memetakan respons integrasi ke respons metode.

Secara terprogram, pengaturan respons metode sama dengan membuat MethodResponsesumber daya API Gateway dan menyetel properti StatusCode, ResponseParameters, dan ResponseModels.

Saat menyetel kode status untuk metode API, Anda harus memilih salah satu sebagai default untuk menangani respons integrasi kode status yang tidak terduga. Masuk akal untuk ditetapkan `500` sebagai default karena ini sama dengan mentransmisikan respons yang tidak dipetakan sebagai kesalahan sisi server. Untuk alasan instruksional, konsol API Gateway menetapkan `200` respons sebagai default. Tetapi Anda dapat mengatur ulang ke `500` respons.

Untuk mengatur respons metode, Anda harus telah membuat permintaan metode.

Mengatur kode status respons metode

Kode status dari respon metode mendefinisikan jenis respon. Misalnya, tanggapan 200, 400, dan 500 menunjukkan keberhasilan, kesalahan sisi klien dan respons kesalahan sisi server, masing-masing.

Untuk menyiapkan kode status respons metode, setel `statusCode`properti ke kode status HTTP. AWS CLI Perintah berikut menciptakan respon metode`200`.

```
aws apigateway put-method-response \
       --region us-west-2  \
       --rest-api-id vaz7da96z6 \
       --resource-id 6sxz2j \
       --http-method GET \
       --status-code 200
```

Siapkan parameter respons metode

Parameter respons metode menentukan header mana yang diterima klien sebagai respons terhadap permintaan metode terkait. Parameter respons juga menentukan target yang API Gateway memetakan parameter respons integrasi, sesuai dengan pemetaan yang ditentukan dalam respons integrasi metode API.

Untuk mengatur parameter respons metode, tambahkan ke `responseParameters`peta pasangan `MethodResponse` kunci-nilai format. `"{parameter-name}":"{boolean}"` Perintah CLI berikut menunjukkan contoh pengaturan header. `my-header`

```
aws apigateway put-method-response \
        --region us-west-2 \
        --rest-api-id vaz7da96z6 \
        --resource-id 6sxz2j \
        --http-method GET \
        --status-code 200  \
        --response-parameters method.response.header.my-header=false
```

Siapkan model respons metode

Model respons metode mendefinisikan format badan respons metode. Sebelum menyiapkan model respons, Anda harus terlebih dahulu membuat model di API Gateway. Untuk melakukannya, Anda dapat memanggil `create-model` perintah. Contoh berikut menunjukkan cara membuat `PetStorePet` model untuk menggambarkan tubuh respons terhadap permintaan `GET /pets/{petId}` metode.

```
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" } \
                } \
            }'
```

Hasilnya dibuat sebagai `Model`sumber daya API Gateway.

Untuk mengatur model respons metode untuk menentukan format payload, tambahkan pasangan nilai kunci “application/json”:”PetStorePet” ke peta sumber daya. `requestModelsMethodResponse` AWS CLI Perintah berikut `put-method-response` menunjukkan bagaimana hal ini dilakukan:

```
aws apigateway put-method-response \
        --region us-west-2 \
        --rest-api-id vaz7da96z6 \
        --resource-id 6sxz2j \
        --http-method GET \
        --status-code 200  \
        --response-parameters method.response.header.my-header=false \
        --response-models '{"application/json":"PetStorePet"}'
```

Menyiapkan model respons metode diperlukan saat Anda membuat SDK yang diketik kuat untuk API. Ini memastikan bahwa output dilemparkan ke kelas yang sesuai di Java atau Objective-C. Dalam kasus lain, pengaturan model adalah opsional.

Siapkan metode menggunakan konsol API Gateway

Sebelum menyiapkan metode API, verifikasi hal berikut:

- Anda harus memiliki metode yang tersedia di API Gateway. Ikuti petunjuk dalam Tutorial: Membangun REST API dengan integrasi non-proxy HTTP.
- Jika Anda ingin metode berkomunikasi dengan fungsi Lambda, Anda harus sudah membuat peran pemanggilan Lambda dan peran eksekusi Lambda di IAM. Anda juga harus membuat fungsi

Lambda yang dengannya metode Anda akan berkomunikasi. AWS Lambda Untuk membuat peran dan fungsi, gunakan instruksi dalam Buat fungsi Lambda untuk integrasi non-proxy LambdaBangun API REST API Gateway dengan integrasi Lambda.

- Jika Anda ingin metode berkomunikasi dengan integrasi proxy HTTP atau HTTP, Anda harus sudah membuat, dan memiliki akses ke, URL titik akhir HTTP yang dengannya metode Anda akan berkomunikasi.
- Verifikasi bahwa sertifikat Anda untuk titik akhir proxy HTTP dan HTTP didukung oleh API Gateway. Untuk detailnya lihatOtoritas sertifikat yang didukung API Gateway untuk integrasi proxy HTTP dan HTTP.

Topik

- Menyiapkan permintaan metode API Gateway di konsol API Gateway
- Menyiapkan respons metode API Gateway menggunakan konsol API Gateway

Menyiapkan permintaan metode API Gateway di konsol API Gateway

Untuk menggunakan konsol API Gateway untuk menentukan permintaan/respons metode API, dan untuk mengonfigurasi bagaimana metode akan mengotorisasi permintaan, ikuti petunjuk ini.

> **Note**
>
> Instruksi ini mengasumsikan Anda telah menyelesaikan langkah-langkahnyaMenyiapkan permintaan integrasi API menggunakan konsol API Gateway. Mereka paling baik digunakan untuk melengkapi diskusi yang diberikanBangun API REST API Gateway dengan integrasi Lambda.

1. Di panel Resources, pilih metode Anda, lalu pilih tab Permintaan metode.
2. Di bagian Pengaturan permintaan metode, pilih Edit.
3. Untuk Otorisasi, pilih otorisasi yang tersedia.

    a. Untuk mengaktifkan akses terbuka ke metode untuk pengguna mana pun, pilih Tidak Ada. Langkah ini dapat dilewati jika pengaturan default belum diubah.

    b. Untuk menggunakan izin IAM untuk mengontrol akses klien ke metode, pilih. `AWS_IAM` Dengan pilihan ini, hanya pengguna peran IAM dengan kebijakan IAM yang benar dilampirkan yang diizinkan untuk memanggil metode ini.

Untuk membuat peran IAM, tentukan kebijakan akses dengan format seperti berikut:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "resource-statement"
      ]
    }
  ]
}
```

Dalam kebijakan akses ini, ***pernyataan sumber daya adalah nilai*** bidang ARN di bagian Pengaturan Otorisasi. Untuk informasi selengkapnya tentang menyetel izin IAM, lihat. Kontrol akses ke API dengan izin IAM

Untuk membuat peran IAM, Anda dapat menyesuaikan instruksi di “Untuk membuat peran pemanggilan Lambda dan kebijakannya” dan “Untuk membuat peran eksekusi Lambda dan kebijakannya” di bagian Buat Fungsi Lambda pada. Bangun API REST API Gateway dengan integrasi Lambda

c. Untuk menggunakan otorisasi Lambda, pilih token atau otorisasi permintaan. Buat otorisasi Lambda agar pilihan ini ditampilkan di menu tarik-turun. Untuk informasi tentang cara membuat otorisasi Lambda, lihat. Gunakan otorisasi API Gateway Lambda

d. Untuk menggunakan kumpulan pengguna Amazon Cognito, pilih kumpulan pengguna yang tersedia di bawah otorisasi kumpulan pengguna Cognito. Buat kumpulan pengguna di Amazon Cognito dan otorisasi kumpulan pengguna Amazon Cognito di API Gateway agar pilihan ini ditampilkan di menu tarik-turun. Untuk informasi tentang cara membuat otorisasi kumpulan pengguna Amazon Cognito, lihat. Kontrol akses ke REST API menggunakan kumpulan pengguna Amazon Cognito sebagai otorisasi

4. Untuk menentukan validasi permintaan, pilih nilai dari menu tarik-turun Permintaan Validator. Untuk menonaktifkan validasi permintaan, pilih Tidak Ada. Untuk informasi selengkapnya tentang setiap opsi, lihatGunakan validasi permintaan di API Gateway.

5. Pilih kunci API yang diperlukan untuk meminta kunci API. Saat diaktifkan, kunci API digunakan dalam rencana penggunaan untuk membatasi lalu lintas klien.
6. (Opsional) Untuk menetapkan nama operasi di Java SDK API ini, yang dihasilkan oleh API Gateway, untuk nama Operasi, masukkan nama. Misalnya, untuk permintaan metode`GET /pets/{petId}`, nama operasi Java SDK yang sesuai adalah, secara default,`GetPetsPetId`. Nama ini dibangun dari kata kerja HTTP metode (`GET`) dan nama variabel jalur sumber daya (`Pets`dan`PetId`). Jika Anda menetapkan nama operasi sebagai`getPetById`, nama operasi SDK menjadi`GetPetById`.
7. Untuk menambahkan parameter string kueri ke metode, lakukan hal berikut:

   a. Pilih parameter string Kueri URL, lalu pilih Tambahkan string kueri.

   b. Untuk Nama, masukkan nama parameter string kueri.

   c. Pilih Diperlukan jika parameter string kueri yang baru dibuat akan digunakan untuk validasi permintaan. Untuk informasi selengkapnya tentang validasi permintaan, lihatGunakan validasi permintaan di API Gateway.

   d. Pilih Caching jika parameter string kueri yang baru dibuat akan digunakan sebagai bagian dari kunci caching. Untuk informasi lebih lanjut tentang caching, lihatGunakan metode atau parameter integrasi sebagai kunci cache untuk mengindeks respons yang di-cache.

   Untuk menghapus parameter string kueri, pilih Hapus.
8. Untuk menambahkan parameter header ke metode, lakukan hal berikut:

   a. Pilih header permintaan HTTP, lalu pilih Tambah header.

   b. Untuk Nama, masukkan nama header.

   c. Pilih Diperlukan jika header yang baru dibuat akan digunakan untuk validasi permintaan. Untuk informasi selengkapnya tentang validasi permintaan, lihatGunakan validasi permintaan di API Gateway.

   d. Pilih Caching jika header yang baru dibuat akan digunakan sebagai bagian dari kunci caching. Untuk informasi lebih lanjut tentang caching, lihatGunakan metode atau parameter integrasi sebagai kunci cache untuk mengindeks respons yang di-cache.

   Untuk menghapus header, pilih Hapus.
9. Untuk mendeklarasikan format payload permintaan metode dengan,, atau kata kerja `PATCH` HTTP `POSTPUT`, pilih Request body, dan lakukan hal berikut:

a. Pilih Tambah model.

b. Untuk Content-type, masukkan tipe MIME (misalnya,). `application/json`

c. Untuk Model, pilih model dari menu tarik-turun. Model yang tersedia saat ini untuk API mencakup default `Empty` dan `Error` model serta model apa pun yang telah Anda buat dan tambahkan ke koleksi Model API. Untuk informasi selengkapnya tentang membuat model, lihatMemahami model data.

> **Note**
>
> Model ini berguna untuk memberi tahu klien tentang format data yang diharapkan dari muatan. Sangat membantu untuk menghasilkan template pemetaan kerangka. Penting untuk menghasilkan SDK API yang diketik dengan kuat dalam bahasa seperti Java, C #, Objective-C, dan Swift. Ini hanya diperlukan jika validasi permintaan diaktifkan terhadap muatan.

10. Pilih Simpan.

Menyiapkan respons metode API Gateway menggunakan konsol API Gateway

Metode API dapat memiliki satu atau beberapa tanggapan. Setiap respons diindeks oleh kode status HTTP-nya. Secara default, konsol API Gateway menambahkan `200` respons ke respons metode. Anda dapat memodifikasinya, misalnya, agar metode dikembalikan `201` sebagai gantinya. Anda dapat menambahkan tanggapan lain, misalnya, untuk penolakan akses dan `409` `500` untuk variabel tahap yang tidak diinisialisasi yang digunakan.

Untuk menggunakan konsol API Gateway untuk memodifikasi, menghapus, atau menambahkan respons ke metode API, ikuti petunjuk berikut.

1. Di panel Resources, pilih metode Anda, lalu pilih tab Respons metode. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Di bagian Pengaturan respons metode, pilih Buat respons.
3. Untuk kode status HTTP, masukkan kode status HTTP seperti`200`,`400`, atau`500`.

   Ketika respons yang dikembalikan ke backend tidak memiliki respons metode yang sesuai yang ditentukan, API Gateway gagal mengembalikan respons ke klien. Sebaliknya, ia mengembalikan respons `500 Internal server error` kesalahan.

4. Pilih Tambahkan header.
5. Untuk nama Header, masukkan nama.

   Untuk mengembalikan header dari backend ke klien, tambahkan header dalam respons metode.
6. Pilih Tambahkan model untuk menentukan format badan respons metode.

   Masukkan jenis media payload respons untuk jenis Konten dan pilih model dari menu tarik-turun Model.
7. Pilih Simpan.

Untuk mengubah respons yang ada, navigasikan ke respons metode Anda, lalu pilih Edit. Untuk mengubah kode status HTTP, pilih Hapus dan buat respons metode baru.

Untuk setiap respons yang dikembalikan dari backend, Anda harus memiliki respons yang kompatibel yang dikonfigurasi sebagai respons metode. Namun, header respons metode konfigurasi dan model payload bersifat opsional kecuali Anda memetakan hasil dari backend ke respons metode sebelum kembali ke klien. Selain itu, model payload respons metode penting jika Anda membuat SDK yang diketik kuat untuk API Anda.

Mengontrol dan mengelola akses ke REST API di API Gateway

API Gateway mendukung beberapa mekanisme untuk mengendalikan dan mengelola akses ke API Anda.

Anda dapat menggunakan mekanisme autentikasi dan autentikasi berikut:

- Kebijakan sumber daya memungkinkan Anda membuat kebijakan berbasis sumber daya untuk mengizinkan atau menolak akses ke API dan metode Anda dari alamat IP sumber tertentu atau titik akhir VPC. Untuk informasi selengkapnya, lihat the section called “Menggunakan kebijakan sumber daya API Gateway”.
- Peran dan kebijakanAWS IAM standar menawarkan kontrol akses yang fleksibel dan kuat yang dapat diterapkan ke seluruh API atau metode individual. Peran dan kebijakan IAM dapat digunakan untuk mengendalikan siapa yang dapat membuat dan mengelola API Anda, serta siapa yang dapat memanggilnya. Untuk informasi selengkapnya, lihat the section called “Gunakan izin IAM”.
- Tag IAM dapat digunakan bersama dengan kebijakan IAM untuk mengontrol akses. Untuk informasi selengkapnya, lihat the section called “Kontrol akses berbasis atribut”.

- Kebijakan endpoint untuk endpoint VPC antarmuka memungkinkan Anda untuk melampirkan kebijakan sumber daya IAM untuk antarmuka endpoint VPC untuk meningkatkan keamanan API pribadi Anda. Untuk informasi selengkapnya, lihat the section called “Menggunakan kebijakan titik akhir VPC untuk API pribadi”.
- Pengotorisasi Lambda adalah fungsi Lambda yang mengontrol akses ke metode REST API menggunakan otentikasi token pembawa — serta informasi yang dijelaskan oleh header, jalur, string kueri, variabel tahap, atau parameter permintaan variabel konteks. Pengotorisasi Lambda digunakan untuk mengontrol siapa yang dapat memanggil metode REST API. Untuk informasi selengkapnya, lihat the section called “Gunakan otorisasi Lambda”.
- Kumpulan pengguna Amazon Cognito memungkinkan Anda membuat solusi autentikasi dan otorisasi yang dapat disesuaikan untuk REST API Anda. Pangkalan pengguna Amazon Cognito digunakan untuk mengontrol siapa yang dapat memanggil metode REST API. Untuk informasi selengkapnya, lihat the section called “Gunakan kumpulan pengguna Amazon Cognito sebagai otorisasi untuk REST API”.

Anda dapat menggunakan mekanisme berikut untuk melakukan tugas lain yang terkait dengan kontrol akses:

- Berbagi sumber daya lintas-asal (CORS) memungkinkan Anda mengontrol cara REST API pada permintaan sumber daya lintas-asal. Untuk informasi selengkapnya, lihat the section called “CORS”.
- Sertifikat SSL sisi klien dapat digunakan untuk memverifikasi bahwa permintaan HTTP ke sistem backend Anda berasal dari API Gateway. Untuk informasi selengkapnya, lihat the section called “Sertifikat Klien”.
- AWS WAFdapat digunakan untuk melindungi API Gateway API Anda dari eksploitasi web umum. Untuk informasi selengkapnya, lihat the section called “AWS WAF”.

Anda dapat menggunakan mekanisme berikut untuk melacak dan membatasi akses yang telah Anda berikan kepada klien resmi:

- Paket penggunaan memungkinkan Anda memberikan kunci API kepada pelanggan Anda—lalu melacak dan membatasi penggunaan tahapan dan metode API Anda untuk setiap kunci API. Untuk informasi selengkapnya, lihat the section called “Paket penggunaan”.

Mengontrol akses ke API dengan kebijakan sumber daya API Gateway

Kebijakan sumber daya Amazon API Gateway adalah dokumen kebijakan JSON yang Anda lampirkan ke API untuk mengontrol apakah prinsipal tertentu (biasanya peran atau grup IAM) dapat memanggil API. Anda dapat menggunakan kebijakan sumber daya API Gateway untuk memungkinkan API Anda dipanggil dengan aman oleh:

- Pengguna dari AWS akun tertentu.
- Rentang alamat IP sumber tertentu atau blok CIDR.
- Cloud pribadi virtual tertentu (VPC) atau titik akhir VPC (di akun apa pun).

Anda dapat menggunakan kebijakan sumber daya untuk semua jenis titik akhir API di API Gateway: pribadi, dioptimalkan tepi, dan Regional.

Untuk API pribadi, Anda dapat menggunakan kebijakan sumber daya bersama dengan kebijakan titik akhir VPC untuk mengontrol prinsipal mana yang memiliki akses ke sumber daya dan tindakan mana. Untuk informasi selengkapnya, lihat the section called “Menggunakan kebijakan titik akhir VPC untuk API pribadi”.

Anda dapat melampirkan kebijakan sumber daya ke API dengan menggunakan AWS Management Console, AWS CLI, atau AWS SDK.

Kebijakan sumber daya API Gateway berbeda dari kebijakan berbasis identitas IAM. Kebijakan berbasis identitas IAM dilampirkan pada pengguna, grup, atau peran IAM dan menentukan tindakan apa yang dapat dilakukan identitas tersebut pada sumber daya mana. Kebijakan sumber daya API Gateway dilampirkan ke sumber daya. Untuk diskusi lebih rinci tentang perbedaan antara kebijakan berbasis identitas dan kebijakan sumber daya, lihat Kebijakan Berbasis Identitas dan Kebijakan Berbasis Sumber Daya.

Anda dapat menggunakan kebijakan sumber daya API Gateway bersama dengan kebijakan IAM.

Topik

Mengakses ikhtisar bahasa kebijakan untuk Amazon API Gateway

Halaman ini menjelaskan elemen dasar yang digunakan dalam kebijakan sumber daya Amazon API Gateway.

Kebijakan sumber daya ditentukan menggunakan sintaks yang sama dengan kebijakan IAM. Untuk informasi bahasa kebijakan selengkapnya, lihat Gambaran Umum Kebijakan IAM dan Referensi AWS Identity and Access Management Kebijakan di Panduan Pengguna IAM.

Untuk informasi tentang cara AWS layanan memutuskan apakah permintaan yang diberikan harus diizinkan atau ditolak, lihat Menentukan Apakah Permintaan Diizinkan atau Ditolak.

Elemen umum dalam kebijakan akses

Dalam arti yang paling mendasar, kebijakan sumber daya berisi elemen-elemen berikut:

- Sumber Daya — API adalah sumber daya Amazon API Gateway yang dapat Anda izinkan atau tolak izinnya. Dalam sebuah kebijakan, Anda menggunakan Nama Sumber Daya Amazon (ARN) untuk mengidentifikasi sumber daya. Anda juga dapat menggunakan sintaks singkat, yang API Gateway secara otomatis diperluas ke ARN penuh saat Anda menyimpan kebijakan sumber daya. Untuk mempelajari selengkapnya, lihat Contoh kebijakan sumber daya API Gateway.

  Untuk format `Resource` elemen lengkap, lihatFormat sumber daya izin untuk menjalankan API di API Gateway.
- Tindakan - Untuk setiap sumber daya, Amazon API Gateway mendukung serangkaian operasi. Anda mengidentifikasi operasi sumber daya yang Anda izinkan (atau tolak) dengan menggunakan kata kunci tindakan.

  Misalnya, `execute-api:Invoke` izin akan memungkinkan izin pengguna untuk memanggil API atas permintaan klien.

  Untuk format `Action` elemen, lihatFormat tindakan izin untuk menjalankan API di API Gateway.
- Efek — Apa efeknya ketika pengguna meminta tindakan tertentu—ini bisa berupa salah satu atau. `Allow` `Deny` Anda juga dapat secara eksplisit menolak akses ke sumber daya, yang mungkin Anda lakukan untuk memastikan bahwa pengguna tidak dapat mengaksesnya, bahkan jika kebijakan lain memberikan akses.

> **Note**
>
> “Penyangkalan implisit” adalah hal yang sama dengan “tolak secara default”.

> “Penyangkalan implisit” berbeda dari “penolakan eksplisit”. Untuk informasi selengkapnya, lihat Perbedaan Antara Menyangkal Secara Default dan Penolakan Eksplisit.

- Principal — Akun atau pengguna mengizinkan akses ke tindakan dan sumber daya dalam pernyataan. Dalam kebijakan sumber daya, prinsipal adalah pengguna atau akun yang menerima izin ini.

Contoh kebijakan sumber daya berikut menunjukkan elemen kebijakan umum sebelumnya. *Kebijakan memberikan akses ke API di bawah id akun yang ditentukan di wilayah tertentu kepada pengguna yang alamat IP sumbernya ada di blok alamat 123.4.5.6/24.* Kebijakan menolak semua akses ke API jika IP sumber pengguna tidak berada dalam jangkauan.

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

Bagaimana kebijakan sumber daya API Gateway memengaruhi alur kerja otorisasi

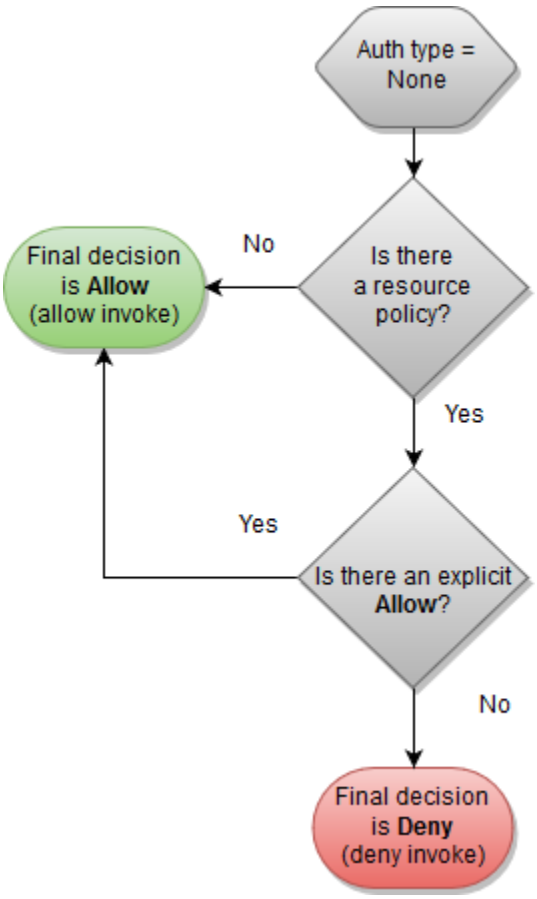
Saat API Gateway mengevaluasi kebijakan sumber daya yang dilampirkan ke API Anda, hasilnya dipengaruhi oleh jenis otentikasi yang telah Anda tetapkan untuk API, seperti yang diilustrasikan dalam diagram alur di bagian berikut.

Topik

Kebijakan sumber daya API Gateway saja

Dalam alur kerja ini, kebijakan sumber daya API Gateway dilampirkan ke API, tetapi tidak ada jenis otentikasi yang ditentukan untuk API. Evaluasi kebijakan melibatkan mencari izin eksplisit berdasarkan kriteria masuk penelepon. Penolakan implisit atau penolakan eksplisit menyebabkan penolakan penelepon.

Berikut adalah contoh kebijakan sumber daya semacam itu.

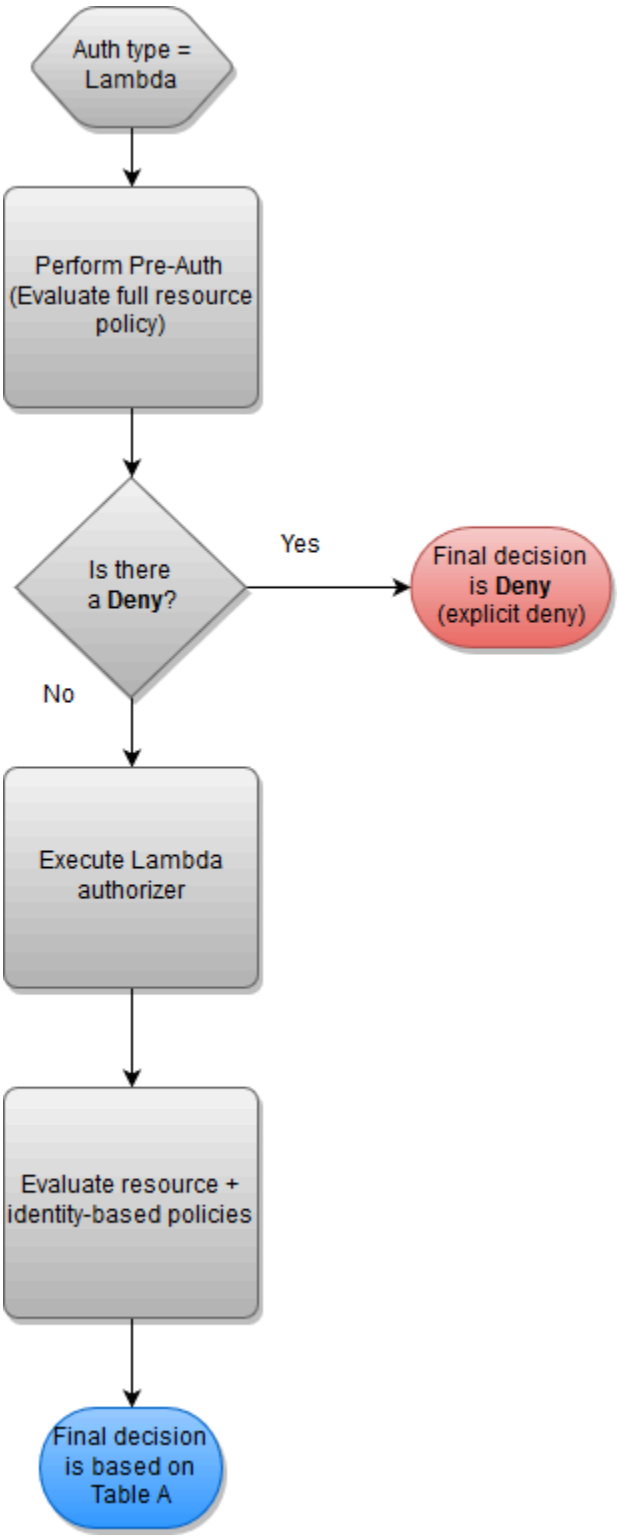



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "arn:aws:execute-api:region:account-id:api-id/",
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": ["192.0.2.0/24", "198.51.100.0/24" ]
                }
            }
        }
    ]
}
```

Kebijakan otorisasi dan sumber daya Lambda

Dalam alur kerja ini, otorisasi Lambda dikonfigurasi untuk API selain kebijakan sumber daya. Kebijakan sumber daya dievaluasi dalam dua fase. Sebelum memanggil otorisasi Lambda, API Gateway pertama-tama mengevaluasi kebijakan dan memeriksa penolakan eksplisit apa pun. Jika ditemukan, penelepon ditolak akses segera. Jika tidak, otorisasi Lambda dipanggil, dan akan mengembalikan dokumen kebijakan, yang dievaluasi bersamaan dengan kebijakan sumber daya. Hasilnya ditentukan berdasarkan Tabel A (menjelang akhir topik ini).

Contoh kebijakan sumber daya berikut memungkinkan panggilan hanya dari titik akhir VPC yang ID titik akhir VPC`vpce-1a2b3c4d`. Selama evaluasi “pra-auth”, hanya panggilan yang berasal dari titik akhir VPC yang ditunjukkan dalam contoh yang diizinkan untuk bergerak maju dan mengevaluasi otorisasi Lambda. Semua panggilan yang tersisa diblokir.



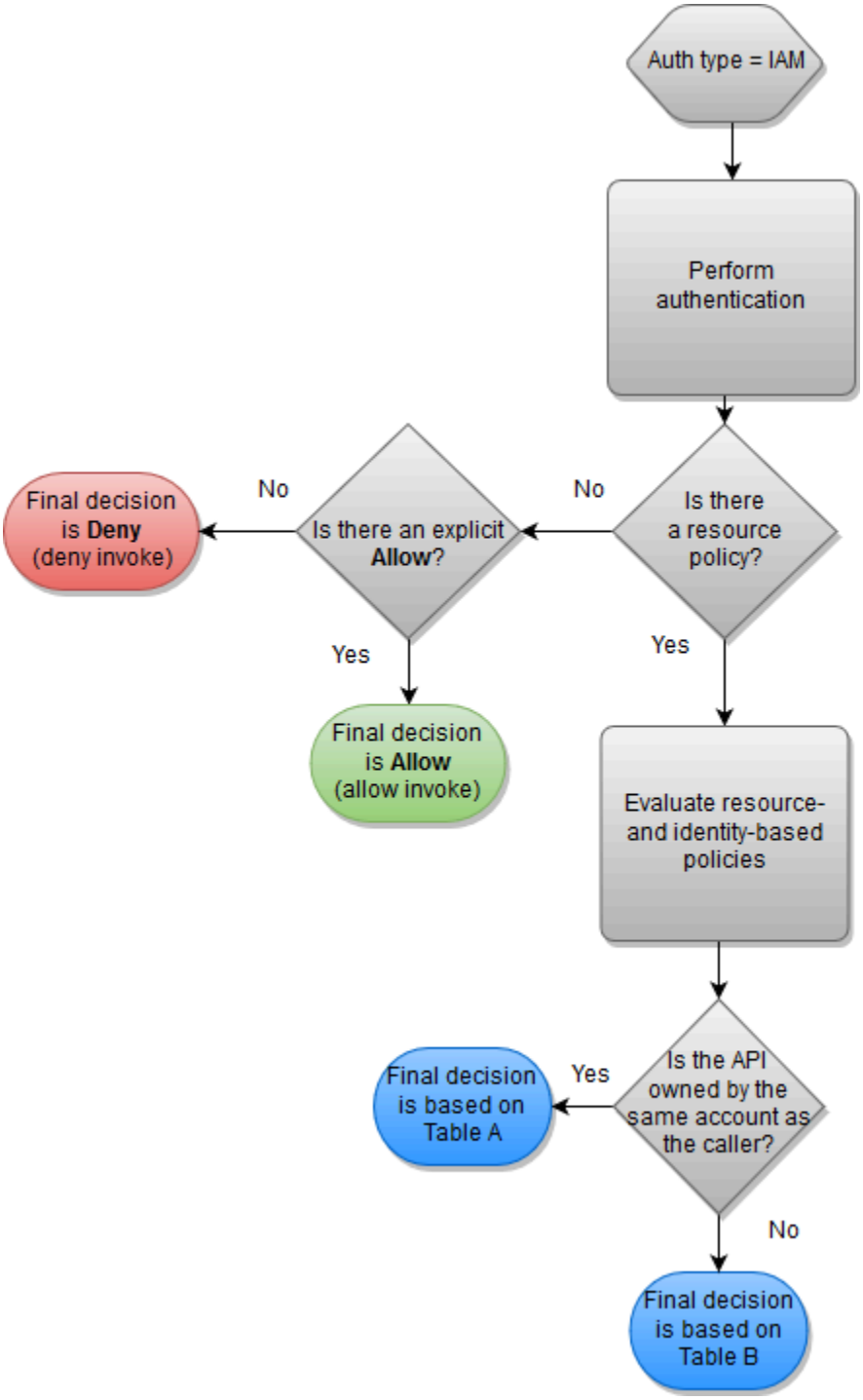

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "arn:aws:execute-api:region:account-id:api-id/"
            ],
            "Condition" : {
                "StringNotEquals": {
                    "aws:SourceVpce": "vpce-1a2b3c4d"
                }
            }
        }
    ]
}
```

Otentikasi IAM dan kebijakan sumber daya

Dalam alur kerja ini, Anda mengkonfigurasi otentikasi IAM untuk API selain kebijakan sumber daya. Setelah Anda mengautentikasi pengguna dengan layanan IAM, API mengevaluasi kebijakan yang dilampirkan ke pengguna dan kebijakan sumber daya. Hasilnya bervariasi berdasarkan apakah pemanggil samaAkun AWS atau terpisahAkun AWS, dari pemilik API.

Jika pemanggil dan pemilik API berasal dari akun terpisah, kebijakan IAM dan kebijakan sumber daya secara eksplisit mengizinkan pemanggil untuk melanjutkan. (Lihat Tabel B di akhir topik ini.) Namun, jika pemanggil dan pemilik API samaAkun AWS, maka kebijakan pengguna IAM atau kebijakan sumber daya harus secara eksplisit mengizinkan pemanggil untuk melanjutkan. (Lihat Tabel A di bawah ini.)



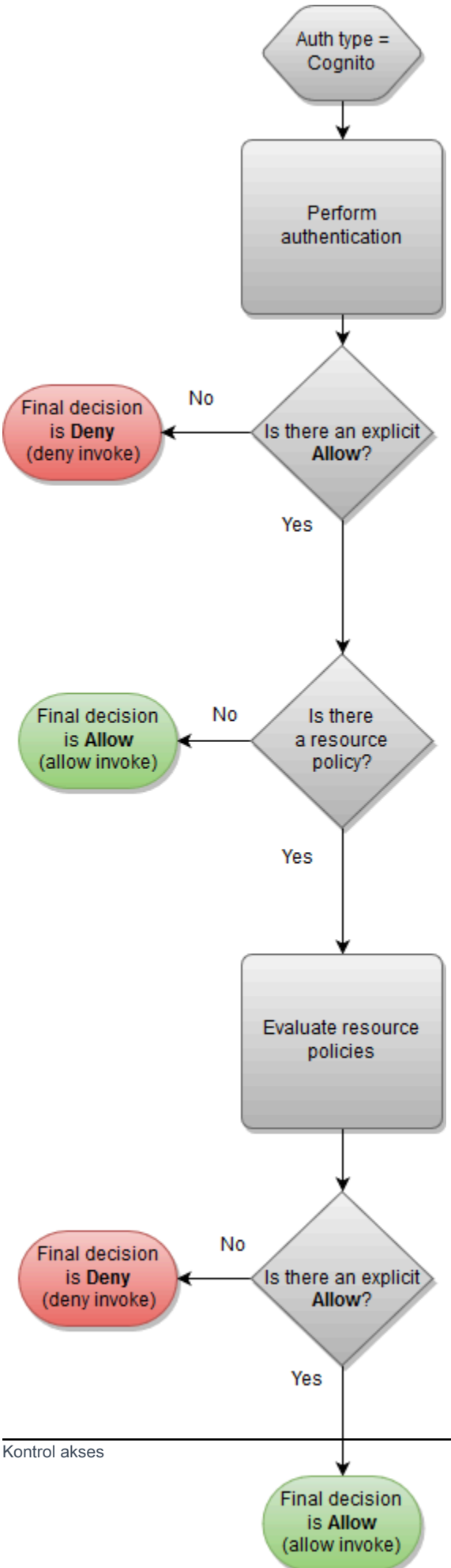
Berikut adalah contoh kebijakan sumber daya lintas akun. Dengan asumsi kebijakan IAM berisi efek allow, kebijakan sumber daya ini hanya mengizinkan panggilan dari VPC yang ID VPC-nya*vpc-2f09a348*. (Lihat Tabel B di akhir topik ini.)

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

Autentikasi Amazon Cognito dan kebijakan sumber daya

Dalam alur kerja ini, kumpulan pengguna Amazon Cognito dikonfigurasi untuk API selain kebijakan sumber daya. API Gateway pertama kali mencoba mengautentikasi pemanggil melalui Amazon Cognito. Ini biasanya dilakukan melalui token JWT yang disediakan oleh penelepon. Jika otentikasi berhasil, kebijakan sumber daya dievaluasi secara independen, dan izin eksplisit diperlukan. Penyangkalan atau “tidak mengizinkan atau menyangkal” menghasilkan penolakan. Berikut adalah contoh kebijakan sumber daya yang mungkin digunakan bersama dengan kolam pengguna Amazon Cognito.


Auth type = Cognito
Perform authentication
Is there an explicit Allow?
No
Final decision is Deny (deny invoke)
Yes
Is there a resource policy?
No
Final decision is Allow (allow invoke)
Yes
Evaluate resource policies
Is there an explicit Allow?
No
Final decision is Deny (deny invoke)
Yes
Final decision is Allow (allow invoke)

Berikut ini adalah contoh kebijakan sumber daya yang memungkinkan panggilan hanya dari IP sumber tertentu, dengan asumsi bahwa token autentikasi Amazon Cognito berisi izin. (Lihat Tabel A di dekat akhir topik ini.)

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

Tabel hasil evaluasi kebijakan

Tabel A mencantumkan perilaku yang dihasilkan saat akses ke API Gateway API dikendalikan oleh kebijakan IAM (atau otorisasi kumpulan pengguna Lambda atau Amazon Cognito) dan kebijakan sumber daya API Gateway, keduanya samaAkun AWS.

Tabel A: Akun A Panggilan API yang Dimiliki oleh Akun A

Kebijakan IAM (atau otorisasi kumpulan pengguna Lambda atau Amazon Cognito)	Kebijakan sumber daya API Gateway	Perilaku yang dihasilkan
Izinkan	Izinkan	Izinkan
Izinkan	Baik Izinkan atau Tolak	Izinkan
Izinkan	Menyangkal	Penolakan jelas
Baik Izinkan atau Tolak	Izinkan	Izinkan
Baik Izinkan atau Tolak	Baik Izinkan atau Tolak	Tolak implisit

Kebijakan IAM (atau otorisasi kumpulan pengguna Lambda atau Amazon Cognito)	Kebijakan sumber daya API Gateway	Perilaku yang dihasilkan
Baik Izinkan atau Tolak	Menyangkal	Penolakan jelas
Menyangkal	Izinkan	Penolakan jelas
Menyangkal	Baik Izinkan atau Tolak	Penolakan jelas
Menyangkal	Menyangkal	Penolakan jelas

Tabel B mencantumkan perilaku yang dihasilkan saat akses ke API Gateway API dikendalikan oleh kebijakan IAM (atau otorisasi kumpulan pengguna Lambda atau Amazon Cognito) dan kebijakan sumber daya API Gateway, yang berbedaAkun AWS. Jika salah satu diam (tidak mengizinkan atau menolak), akses lintas akun ditolak. Ini karena akses lintas akun mengharuskan kebijakan sumber daya dan kebijakan IAM (atau otorisasi kumpulan pengguna Lambda atau Amazon Cognito) secara eksplisit memberikan akses.

Tabel B: Akun B Panggilan API Dimiliki oleh Akun A

Kebijakan IAM (atau otorisasi kumpulan pengguna Lambda atau Amazon Cognito)	Kebijakan sumber daya API Gateway	Perilaku yang dihasilkan
Izinkan	Izinkan	Izinkan
Izinkan	Baik Izinkan atau Tolak	Tolak implisit
Izinkan	Menyangkal	Penolakan jelas
Baik Izinkan atau Tolak	Izinkan	Tolak implisit
Baik Izinkan atau Tolak	Baik Izinkan atau Tolak	Tolak implisit
Baik Izinkan atau Tolak	Menyangkal	Penolakan jelas
Menyangkal	Izinkan	Penolakan jelas
Menyangkal	Baik Izinkan atau Tolak	Penolakan jelas

Kebijakan IAM (atau otorisasi kumpulan pengguna Lambda atau Amazon Cognito)	Kebijakan sumber daya API Gateway	Perilaku yang dihasilkan
Menyangkal	Menyangkal	Penolakan jelas

Contoh kebijakan sumber daya API Gateway

Halaman ini menyajikan beberapa contoh kasus penggunaan umum untuk kebijakan sumber daya API Gateway.

Contoh kebijakan berikut menggunakan sintaks yang disederhanakan untuk menentukan sumber daya API. Sintaks yang disederhanakan ini adalah cara singkat yang dapat Anda rujuk ke sumber daya API, alih-alih menentukan Nama Sumber Daya Amazon (ARN) lengkap. API Gateway mengonversi sintaks yang disingkat menjadi ARN lengkap saat Anda menyimpan kebijakan. Misalnya, Anda dapat menentukan sumber daya `execute-api:/`*`stage-name`*`/`*`GET`*`/`*`pets`* dalam kebijakan sumber daya. API Gateway mengonversi sumber daya menjadi `arn:aws:execute-api:us-east-2:123456789012:aabbccddee/stage-name/GET/pets` saat Anda menyimpan kebijakan sumber daya. API Gateway membangun ARN lengkap dengan menggunakan Wilayah saat ini, ID akun AWS Anda, dan ID REST API yang terkait dengan kebijakan sumber daya. Anda dapat menggunakan `execute-api:/*` untuk mewakili semua tahapan, metode, dan jalur di API saat ini. Untuk informasi tentang bahasa kebijakan akses, lihat Mengakses ikhtisar bahasa kebijakan untuk Amazon API Gateway.

Topik

- Contoh: Izinkan peran di AWS akun lain untuk menggunakan API
- Contoh: Tolak lalu lintas API berdasarkan alamat atau rentang IP sumber
- Contoh: Tolak lalu lintas API berdasarkan alamat atau rentang IP sumber saat menggunakan API pribadi
- Contoh: Izinkan lalu lintas API pribadi berdasarkan titik akhir VPC atau VPC sumber

Contoh: Izinkan peran di AWS akun lain untuk menggunakan API

Contoh kebijakan sumber daya berikut memberikan akses API dalam satu AWS akun ke dua peran di akun yang berbeda AWS melalui protokol Signature Version 4 (SigV4). Secara khusus, peran pengembang dan administrator untuk AWS akun yang diidentifikasi oleh *`account-id-2`* diberikan

`execute-api:Invoke` tindakan untuk menjalankan `GET` tindakan pada `pets` sumber daya (API) di AWS akun Anda.

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::account-id-2:role/developer",
                    "arn:aws:iam::account-id-2:role/Admin"
                ]
            },
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/stage/GET/pets"
            ]
        }
    ]
}
```

Contoh: Tolak lalu lintas API berdasarkan alamat atau rentang IP sumber

Contoh kebijakan sumber daya berikut menyangkal (memblokir) lalu lintas masuk ke API dari dua blok alamat IP sumber yang ditentukan.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
```

```
                "execute-api:/*"
            ],
            "Condition" : {
                "IpAddress": {
                    "aws:SourceIp": ["192.0.2.0/24", "198.51.100.0/24" ]
                }
            }
        }
    ]
}
```

Contoh: Tolak lalu lintas API berdasarkan alamat atau rentang IP sumber saat menggunakan API pribadi

Contoh kebijakan sumber daya berikut menyangkal (memblokir) lalu lintas masuk ke API pribadi dari dua blok alamat IP sumber yang ditentukan. Saat menggunakan API pribadi, titik akhir VPC untuk `execute-api` menulis ulang alamat IP sumber asli. `aws:VpcSourceIp`Kondisi memfilter permintaan terhadap alamat IP pemohon asli.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ],
            "Condition" : {
                "IpAddress": {
                    "aws:VpcSourceIp": ["192.0.2.0/24", "198.51.100.0/24"]
                }
            }
        }
```

```
    ]
}
```

Contoh: Izinkan lalu lintas API pribadi berdasarkan titik akhir VPC atau VPC sumber

Contoh kebijakan sumber daya berikut mengizinkan lalu lintas masuk ke API pribadi hanya dari virtual private cloud (VPC) atau titik akhir VPC tertentu.

Contoh kebijakan sumber daya ini menentukan VPC sumber:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ],
            "Condition" : {
                "StringNotEquals": {
                   "aws:SourceVpc": "vpc-1a2b3c4d"
                }
            }
        }
    ]
}
```

Contoh kebijakan sumber daya ini menentukan titik akhir VPC sumber:

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

```
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ],
            "Condition" : {
                "StringNotEquals": {
                    "aws:SourceVpce": "vpce-1a2b3c4d"
                }
            }
        }
    ]
}
```

Membuat dan melampirkan kebijakan sumber daya API Gateway ke API

Untuk mengizinkan pengguna mengakses API Anda dengan memanggil layanan eksekusi API, Anda harus membuat kebijakan sumber daya API Gateway, yang mengontrol akses ke resource API Gateway, dan melampirkan kebijakan tersebut ke API.

> **Important**
>
> Untuk memperbarui kebijakan sumber daya API Gateway, Anda harus memiliki `apigateway:UpdateRestApiPolicy` izin selain `apigateway:PATCH` izin.

Kebijakan sumber daya dapat dilampirkan ke API saat API sedang dibuat, atau dapat dilampirkan setelahnya. Untuk API pribadi, perhatikan bahwa hingga Anda melampirkan kebijakan sumber daya ke API pribadi, semua panggilan ke API akan gagal.

> **Important**
>
> Jika memperbarui kebijakan sumber daya setelah API dibuat, Anda harus menerapkan API untuk menyebarkan perubahan setelah Anda melampirkan kebijakan yang diperbarui. Memperbarui atau menyimpan kebijakan saja tidak akan mengubah perilaku runtime API. Untuk informasi selengkapnya tentang penerapan API Anda, lihatMenerapkan REST API di Amazon API Gateway.

Anda dapat mengontrol akses dengan elemen kondisi IAM, termasuk kondisi pada, VPC sumber Akun AWS, titik akhir VPC sumber, atau rentang IP. Jika Anda menyetel kebijakan`"*"`, Anda dapat menggunakan jenis otorisasi lain di samping kebijakan sumber daya. `Principal` Namun, jika Anda mengatur `Principal` ke`"AWS"`, otorisasi gagal untuk semua sumber daya yang tidak diamankan dengan `AWS_IAM` otorisasi, termasuk sumber daya yang tidak aman.

Bagian berikut menjelaskan cara membuat kebijakan sumber daya API Gateway Anda sendiri dan melampirkannya ke API Anda. Melampirkan kebijakan akan menerapkan izin dalam kebijakan ke metode di API.

> **Important**
>
> Jika Anda menggunakan konsol API Gateway untuk melampirkan kebijakan sumber daya ke API yang diterapkan, atau jika Anda memperbarui kebijakan sumber daya yang ada, Anda harus menerapkan ulang API di konsol agar perubahan diterapkan.

Topik

- Melampirkan kebijakan sumber daya API Gateway (konsol)
- Melampirkan kebijakan sumber daya API Gateway (AWS CLI)
- Melampirkan kebijakan sumber daya API Gateway ()AWS CloudFormation

Melampirkan kebijakan sumber daya API Gateway (konsol)

Anda dapat menggunakan konsol AWS Manajemen untuk melampirkan kebijakan sumber daya ke API Gateway API.

Untuk melampirkan kebijakan sumber daya ke API Gateway API

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih REST API.
3. Di panel navigasi utama, pilih Kebijakan sumber daya.
4. Pilih Buat kebijakan.
5. (Opsional) Pilih templat untuk menghasilkan contoh kebijakan.

   Dalam contoh kebijakan, placeholder dilampirkan dalam double curly braces (). `"{{placeholder}}"` Ganti masing-masing placeholder, termasuk kawat gigi keriting, dengan informasi yang diperlukan.
6. Jika Anda tidak menggunakan salah satu contoh templat, masukkan kebijakan sumber daya Anda.
7. Pilih Simpan perubahan.

Jika API telah digunakan sebelumnya di konsol API Gateway, Anda harus menerapkannya kembali agar kebijakan sumber daya diterapkan.

Melampirkan kebijakan sumber daya API Gateway (AWS CLI)

Untuk menggunakan AWS CLI untuk membuat API baru dan melampirkan kebijakan sumber daya ke dalamnya, panggil `create-rest-api`perintah sebagai berikut:

```
aws apigateway create-rest-api \
    --name "api-name" \
    --policy "{\"jsonEscapedPolicyDocument\"}"
```

Untuk menggunakan kebijakan AWS CLI untuk melampirkan resource ke API yang ada, panggil `update-rest-api`perintah sebagai berikut:

```
aws apigateway update-rest-api \
    --rest-api-id api-id \
    --patch-operations op=replace,path=/
policy,value='"{\"jsonEscapedPolicyDocument\"}"'
```

Melampirkan kebijakan sumber daya API Gateway ()AWS CloudFormation

Anda dapat menggunakan AWS CloudFormation untuk membuat API dengan kebijakan sumber daya. Contoh berikut membuat REST API dengan contoh kebijakan sumber daya,the section called “Contoh: Tolak lalu lintas API berdasarkan alamat atau rentang IP sumber”.

Contoh AWS CloudFormation template

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Name: testapi
      Policy:
        Statement:
          - Action: 'execute-api:Invoke'
            Effect: Allow
            Principal: '*'
            Resource: 'execute-api/*'
          - Action: 'execute-api:Invoke'
            Effect: Deny
            Principal: '*'
            Resource: 'execute-api/*'
            Condition:
              IpAddress:
                'aws:SourceIp': ["192.0.2.0/24", "198.51.100.0/24" ]
        Version: 2012-10-17
  Resource:
    Type: 'AWS::ApiGateway::Resource'
    Properties:
      RestApiId: !Ref Api
      ParentId: !GetAtt Api.RootResourceId
      PathPart: 'helloworld'
  MethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref Resource
      HttpMethod: GET
      ApiKeyRequired: false
      AuthorizationType: NONE
      Integration:
```

```
        Type: MOCK
  ApiDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn:
      - MethodGet
    Properties:
      RestApiId: !Ref Api
      StageName: test
```

AWS kunci kondisi yang dapat digunakan dalam kebijakan sumber daya API Gateway

Tabel berikut berisi kunci AWS kondisi yang dapat digunakan dalam kebijakan sumber daya untuk API di API Gateway untuk setiap jenis otorisasi.

Untuk informasi selengkapnya tentang kunci AWS kondisi, lihat Kunci Konteks Kondisi AWS Global.

Tabel kunci kondisi

Kunci syarat	Kriteria	Kebutuhan**AuthN**?	Jenis otorisasi
`aws:CurrentTime`	Tidak ada	Tidak	Semua
`aws:EpochTime`	Tidak ada	Tidak	Semua
`aws:TokenIssueTime`	Kunci hanya ada dalam permintaan yang ditandatangani menggunakan kredenal keamanan sementara.	Ya	IAM
`aws:MultiFactorAuthPresent`	Kunci hanya ada dalam permintaan yang ditandatangani menggunakan kredenal keamanan sementara.	Ya	IAM

Kunci syarat	Kriteria	Kebutuhan**AuthN**?	Jenis otorisasi
`aws:MultiFactorAuthAge`	Kunci hadir hanya jika MFA hadir dalam permintaan.	Ya	IAM
`aws:PrincipalAccount`	Tidak ada	Ya	IAM
`aws:PrincipalArn`	Tidak ada	Ya	IAM
`aws:PrincipalOrgID`	Kunci ini disertakan dalam konteks permintaan hanya jika kepala sekolah adalah anggota organisasi.	Ya	IAM
`aws:PrincipalOrgPaths`	Kunci ini disertakan dalam konteks permintaan hanya jika kepala sekolah adalah anggota organisasi.	Ya	IAM
`aws:PrincipalTag`	Kunci ini disertakan dalam konteks permintaan jika prinsipal menggunakan pengguna IAM dengan tag terlampir. Ini disertakan untuk prinsipal yang menggunakan peran IAM dengan tag yang disematkan atau tag sesi.	Ya	IAM

Kunci syarat	Kriteria	Kebutuhan**AuthN**?	Jenis otorisasi
`aws:PrincipalType`	Tidak ada	Ya	IAM
`aws:Referer`	Kunci hadir hanya jika nilai disediakan oleh pemanggil di header HTTP.	Tidak	Semua
`aws:SecureTransport`	Tidak ada	Tidak	Semua
`aws:SourceArn`	Tidak ada	Tidak	Semua
`aws:SourceIp`	Tidak ada	Tidak	Semua
`aws:SourceVpc`	Kunci ini hanya dapat digunakan untuk API pribadi.	Tidak	Semua
`aws:SourceVpce`	Kunci ini hanya dapat digunakan untuk API pribadi.	Tidak	Semua
`aws:UserAgent`	Kunci hadir hanya jika nilai disediakan oleh pemanggil di header HTTP.	Tidak	Semua
`aws:userid`	Tidak ada	Ya	IAM
`aws:username`	Tidak ada	Ya	IAM

Kontrol akses ke API dengan izin IAM

Anda mengontrol akses ke Amazon API Gateway API dengan izin IAM dengan mengontrol akses ke dua proses komponen API Gateway berikut:

- Untuk membuat, menerapkan, dan mengelola API di API Gateway, Anda harus memberikan izin pengembang API untuk melakukan tindakan yang diperlukan yang didukung oleh komponen manajemen API API Gateway.
- Untuk memanggil API yang diterapkan atau menyegarkan cache API, Anda harus memberikan izin pemanggil API untuk melakukan tindakan IAM yang diperlukan yang didukung oleh komponen eksekusi API API Gateway.

Kontrol akses untuk dua proses melibatkan model izin yang berbeda, dijelaskan selanjutnya.

Model izin API Gateway untuk membuat dan mengelola API

Untuk mengizinkan pengembang API membuat dan mengelola API di API Gateway, Anda harus membuat kebijakan izin IAM yang memungkinkan pengembang API tertentu untuk membuat, memperbarui, menerapkan, melihat, atau menghapus entitas API yang diperlukan. Anda melampirkan kebijakan izin ke pengguna, peran, atau grup.

Untuk memberikan akses, tambahkan izin ke pengguna, grup, atau peran Anda:

- Pengguna dan grup diAWS IAM Identity Center:

  Buat set izin. Ikuti petunjuk di Buat set izin di Panduan AWS IAM Identity Center Pengguna.
- Pengguna yang dikelola di IAM melalui penyedia identitas:

  Buat peran untuk federasi identitas. Ikuti petunjuk dalam Membuat peran untuk penyedia identitas pihak ketiga (federasi) di Panduan Pengguna IAM.
- Pengguna IAM:
  - Buat peran yang dapat diasumsikan oleh pengguna Anda. Ikuti petunjuk dalam Membuat peran untuk pengguna IAM di Panduan Pengguna IAM.
  - (Tidak disarankan) Lampirkan kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti petunjuk di Menambahkan izin ke pengguna (konsol) di Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang cara menggunakan model izin ini, lihatthe section called “Kebijakan berbasis identitas API Gateway”.

Model izin API Gateway untuk menjalankan API

Untuk mengizinkan pemanggil API menjalankan API atau menyegarkan caching, Anda harus membuat kebijakan IAM yang mengizinkan pemanggil API tertentu untuk menjalankan metode API yang memungkinkan otentikasi pengguna diaktifkan. Pengembang API menetapkan `authorizationType` properti metode `AWS_IAM` agar pemanggil mengirimkan kredensyal pengguna untuk diautentikasi. Kemudian, Anda melampirkan kebijakan ke pengguna, peran, atau grup.

Dalam pernyataan kebijakan izin IAM ini, `Resource` elemen IAM berisi daftar metode API yang diterapkan yang diidentifikasi oleh kata kerja HTTP dan jalur sumber daya API Gateway yang diberikan. `Action`Elemen IAM berisi tindakan eksekusi API Gateway API yang diperlukan. Tindakan ini mencakup `execute-api:Invoke` atau`execute-api:InvalidateCache`, di mana `execute-api` menunjuk komponen eksekusi API yang mendasari API Gateway.

Untuk informasi selengkapnya tentang cara menggunakan model izin ini, lihat Kontrol akses untuk menjalankan API.

Ketika API terintegrasi dengan AWS layanan (misalnya,AWS Lambda) di bagian belakang, API Gateway juga harus memiliki izin untuk mengakses AWS sumber daya terintegrasi (misalnya, menjalankan fungsi Lambda) atas nama pemanggil API. Untuk memberikan izin ini, buat peran IAM dari AWSlayanan untuk jenis API Gateway. Saat Anda membuat peran ini di konsol Manajemen IAM, peran yang dihasilkan ini berisi kebijakan kepercayaan IAM berikut yang mendeklarasikan API Gateway sebagai entitas tepercaya yang diizinkan untuk mengambil peran:

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

Jika Anda membuat peran IAM dengan memanggil perintah create-role CLI atau metode SDK yang sesuai, Anda harus memberikan kebijakan kepercayaan di atas sebagai parameter input. `assume-`

`role-policy-document` Jangan mencoba membuat kebijakan semacam itu secara langsung di konsol Manajemen IAM atau memanggil perintah AWS CLI create-policy atau metode SDK yang sesuai.

Agar API Gateway memanggil AWS layanan terintegrasi, Anda juga harus melampirkan kebijakan izin IAM yang sesuai peran ini untuk memanggil layanan terintegrasiAWS. Misalnya, untuk memanggil fungsi Lambda, Anda harus menyertakan kebijakan izin IAM berikut dalam peran IAM:

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

Perhatikan bahwa Lambda mendukung kebijakan akses berbasis sumber daya, yang menggabungkan kebijakan kepercayaan dan izin. Saat mengintegrasikan API dengan fungsi Lambda menggunakan konsol API Gateway, Anda tidak diminta untuk menyetel peran IAM ini secara eksplisit, karena konsol menetapkan izin berbasis sumber daya pada fungsi Lambda untuk Anda, dengan persetujuan Anda.

> **Note**
>
> Untuk memberlakukan kontrol akses ke AWS layanan, Anda dapat menggunakan model izin berbasis pemanggil, di mana kebijakan izin langsung dilampirkan ke pengguna atau grup pemanggil, atau model izin berbasis peran, di mana kebijakan izin dilampirkan ke peran IAM yang dapat diasumsikan oleh API Gateway. Kebijakan izin mungkin berbeda dalam kedua model. Misalnya, kebijakan berbasis pemanggil memblokir akses sementara kebijakan berbasis peran mengizinkannya. Anda dapat memanfaatkan ini untuk meminta pengguna mengakses AWS layanan melalui API Gateway API saja.

Kontrol akses untuk menjalankan API

Di bagian ini Anda akan mempelajari cara menulis pernyataan kebijakan IAM untuk mengontrol siapa yang dapat memanggil API yang diterapkan di API Gateway. Di sini, Anda juga akan menemukan

referensi pernyataan kebijakan, termasuk format `Action` dan `Resource` bidang yang terkait dengan layanan eksekusi API. Anda juga harus mempelajari bagian IAM dithe section called “Bagaimana kebijakan sumber daya memengaruhi alur kerja otorisasi”.

Untuk API pribadi, Anda harus menggunakan kombinasi kebijakan sumber daya API Gateway dan kebijakan titik akhir VPC. Untuk informasi lain, lihat topik berikut:

- the section called “Menggunakan kebijakan sumber daya API Gateway”
- the section called “Menggunakan kebijakan titik akhir VPC untuk API pribadi”

Kontrol siapa yang dapat memanggil metode API Gateway API dengan kebijakan IAM

Untuk mengontrol siapa yang dapat atau tidak dapat memanggil API yang diterapkan dengan izin IAM, buat dokumen kebijakan IAM dengan izin yang diperlukan. Template untuk dokumen kebijakan semacam itu ditampilkan sebagai berikut.

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

Di sini, *`Permission`* akan diganti oleh `Allow` atau `Deny` tergantung pada apakah Anda ingin memberikan atau mencabut izin yang disertakan. *`Execution-operation`*akan digantikan oleh operasi yang didukung oleh layanan eksekusi API. *`METHOD_HTTP_VERB`*singkatan dari kata kerja HTTP yang didukung oleh sumber daya yang ditentukan. *`Resource-path`*adalah placeholder untuk jalur URL dari `Resource` instance API yang diterapkan yang mendukung hal tersebut. *`METHOD_HTTP_VERB`* Untuk informasi selengkapnya, lihat Referensi pernyataan kebijakan IAM untuk menjalankan API di API Gateway.

> **Note**
>
> Agar kebijakan IAM efektif, Anda harus mengaktifkan autentikasi IAM pada metode API dengan menyetel properti `AWS_IAM` metode. `authorizationType` Gagal melakukannya akan membuat metode API ini dapat diakses publik.

Misalnya, untuk memberikan izin kepada pengguna untuk melihat daftar hewan peliharaan yang diekspos oleh API tertentu, tetapi untuk menolak izin pengguna untuk menambahkan hewan peliharaan ke daftar, Anda dapat menyertakan pernyataan berikut dalam kebijakan 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/*/POST/pets"
      ]
    }
  ]
}
```

Untuk memberikan izin kepada pengguna untuk melihat hewan peliharaan tertentu yang diekspos oleh API yang dikonfigurasi sebagai`GET /pets/{petId}`, Anda dapat menyertakan pernyataan berikut dalam kebijakan 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"
      ]
    }
  ]
}
```

Referensi pernyataan kebijakan IAM untuk menjalankan API di API Gateway

Informasi berikut menjelaskan format Tindakan dan Sumber Daya pernyataan kebijakan IAM tentang izin akses untuk menjalankan API.

Format tindakan izin untuk menjalankan API di API Gateway

`Action`Ekspresi pelaksana API memiliki format umum berikut:

```
execute-api:action
```

di mana *`action adalah tindakan`* pelaksana API yang tersedia:

- *, yang mewakili semua tindakan berikut.
- Invoke, digunakan untuk memanggil API atas permintaan klien.
- InvalidateCache, digunakan untuk membatalkan cache API atas permintaan klien.

Format sumber daya izin untuk menjalankan API di API Gateway

`Resource`Ekspresi pelaksana API memiliki format umum berikut:

```
arn:aws:execute-api:region:account-id:api-id/stage-name/HTTP-VERB/resource-path-
specifier
```

di mana:

- *`wilayah`* adalah AWS wilayah (seperti **`us-east-1`** atau **`*`** untuk semua AWS wilayah) yang sesuai dengan API yang diterapkan untuk metode tersebut.

- *account-id* adalah 12 digit Id AWS akun dari pemilik REST API.
- *api-id* adalah pengidentifikasi API Gateway yang telah ditetapkan ke API untuk metode ini.
- nama *panggung adalah nama* panggung yang terkait dengan metode ini.
- *HTTP-VERB* adalah kata kerja HTTP untuk metode ini. Ini bisa menjadi salah satu dari yang berikut: GET, POST, PUT, DELETE, PATCH.
- *resource-path-specifier*adalah jalan menuju metode yang diinginkan.

> **Note**
>
> Jika Anda menentukan wildcard (*), `Resource` ekspresi akan menerapkan wildcard ke ekspresi lainnya.

Beberapa contoh ekspresi sumber daya meliputi:

- **arn:aws:execute-api:*:*:***untuk jalur sumber daya apa pun di tahap apa pun, untuk API apa pun di AWS wilayah mana pun.
- **arn:aws:execute-api:us-east-1:*:***untuk jalur sumber daya apa pun di tahap apa pun, untuk API apa pun di AWS wilayah`us-east-1`.
- **arn:aws:execute-api:us-east-1:*:*api-id*/***untuk jalur sumber daya apa pun di tahap apa pun, untuk API dengan pengidentifikasi *api-id* di AWS wilayah us-east-1.
- **arn:aws:execute-api:us-east-1:*:*api-id*/test/***untuk jalur sumber daya pada tahap`test`, untuk API dengan pengidentifikasi *api-id* di AWS wilayah us-east-1.

Untuk mempelajari selengkapnya, lihat Referensi API Gateway Amazon Resource Name (ARN).

Contoh kebijakan IAM untuk izin eksekusi API

Untuk model izin dan informasi latar belakang lainnya, lihat Kontrol akses untuk menjalankan API.

Pernyataan kebijakan berikut memberikan izin kepada pengguna untuk memanggil metode POST apa pun di sepanjang jalur`mydemoresource`, pada tahap`test`, untuk API dengan pengenal`a123456789`, dengan asumsi API yang sesuai telah diterapkan ke AWS wilayah 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/*"
      ]
    }
  ]
}
```

Contoh pernyataan kebijakan berikut memberikan izin kepada pengguna untuk memanggil metode apa pun di jalur sumber dayapetstorewalkthrough/pets, dalam tahap apa pun, untuk API dengan pengenala123456789, di AWS wilayah mana pun di mana API terkait telah digunakan:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:*:*:a123456789/*/*/petstorewalkthrough/pets"
      ]
    }
  ]
}
```

Membuat dan melampirkan kebijakan ke pengguna

Agar pengguna dapat memanggil layanan pengelolaan API atau layanan eksekusi API, Anda harus membuat kebijakan IAM yang mengontrol akses ke entitas API Gateway.

Untuk menggunakan editor kebijakan JSON untuk membuat kebijakan

1. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.

2. Di panel navigasi di sebelah kiri, pilih Kebijakan.

   Jika ini pertama kalinya Anda memilih Kebijakan, akan muncul laman Selamat Datang di Kebijakan Terkelola. Pilih Memulai.

3. Di bagian atas halaman, pilih Buat kebijakan.

4. Di bagian Editor kebijakan, pilih opsi JSON.

5. Masukkan dokumen kebijakan JSON berikut:

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [
        "action-statement"
      ],
      "Resource" : [
        "resource-statement"
      ]
    },
    {
      "Effect" : "Allow",
      "Action" : [
        "action-statement"
      ],
      "Resource" : [
        "resource-statement"
      ]
    }
  ]
}
```

6. Pilih Selanjutnya.

> **Note**
>
> Anda dapat beralih antara opsi editor Visual dan JSON kapan saja. Namun, jika Anda membuat perubahan atau memilih Berikutnya di editor Visual, IAM mungkin merestrukturisasi kebijakan Anda untuk mengoptimalkannya untuk editor visual. Untuk informasi selengkapnya, lihat Restrukturisasi kebijakan dalam Panduan Pengguna IAM.

7. Pada halaman Tinjau dan buat, masukkan nama Kebijakan dan Deskripsi (opsional) untuk kebijakan yang Anda buat. Tinjau Izin yang ditentukan dalam kebijakan ini untuk melihat izin yang diberikan oleh kebijakan Anda.
8. Pilih Buat kebijakan untuk menyimpan kebijakan baru Anda.

Dalam pernyataan ini, gantikan *`pernyataan tindakan dan pernyataan sumber daya sesuai kebutuhan, dan tambahkan pernyataan`* lain untuk menentukan entitas API Gateway yang ingin Anda izinkan pengguna kelola, metode API yang dapat dipanggil pengguna, atau keduanya. Secara default, pengguna tidak memiliki izin kecuali ada pernyataan eksplisit yang sesuai`Allow`.

Anda baru saja membuat kebijakan IAM. Itu tidak akan berpengaruh sampai Anda melampirkannya.

Untuk memberikan akses, tambahkan izin ke pengguna, grup, atau peran Anda:

- Pengguna dan grup diAWS IAM Identity Center:

  Buat set izin. Ikuti petunjuk di Buat set izin di Panduan AWS IAM Identity Center Pengguna.
- Pengguna yang dikelola di IAM melalui penyedia identitas:

  Buat peran untuk federasi identitas. Ikuti petunjuk dalam Membuat peran untuk penyedia identitas pihak ketiga (federasi) di Panduan Pengguna IAM.
- Pengguna IAM:
  - Buat peran yang dapat diasumsikan oleh pengguna Anda. Ikuti petunjuk dalam Membuat peran untuk pengguna IAM di Panduan Pengguna IAM.
  - (Tidak disarankan) Lampirkan kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti petunjuk di Menambahkan izin ke pengguna (konsol) di Panduan Pengguna IAM.

Untuk melampirkan dokumen kebijakan IAM ke grup IAM

1. Pilih Grup dari panel navigasi utama.
2. Pilih tab Izin di bawah grup yang dipilih.
3. Pilih Lampirkan kebijakan.
4. Pilih dokumen kebijakan yang Anda buat sebelumnya, lalu pilih Lampirkan kebijakan.

Agar API Gateway memanggil AWS layanan lain atas nama Anda, buat peran IAM dari jenis Amazon API Gateway.

Untuk membuat jenis peran Amazon API Gateway

1. Pilih Peran dari panel navigasi utama.
2. Pilih Buat Peran Baru.
3. Ketik nama untuk nama Peran dan kemudian pilih Langkah Berikutnya.
4. Di bawah Pilih Jenis Peran, di Peran AWS Layanan, pilih Pilih di samping Amazon API Gateway.
5. Pilih kebijakan izin IAM terkelola yang tersedia, misalnya, AmazonAPI GatewayPushToCloudWatchLog jika Anda ingin API Gateway mencatat metrik CloudWatch, di bawah Lampirkan Kebijakan, lalu pilih Langkah Berikutnya.
6. Di bawah Entitas Tepercaya, verifikasi bahwa apigateway.amazonaws.com terdaftar sebagai entri, lalu pilih Buat Peran.
7. Dalam peran yang baru dibuat, pilih tab Izin dan kemudian pilih Lampirkan Kebijakan.
8. Pilih dokumen kebijakan IAM kustom yang dibuat sebelumnya, lalu pilih Lampirkan Kebijakan.

Menggunakan kebijakan titik akhir VPC untuk API pribadi di API Gateway

Anda dapat meningkatkan keamanan API pribadi Anda dengan mengonfigurasi API Gateway untuk menggunakan titik akhir VPC antarmuka. Endpoint antarmuka didukung olehAWS PrivateLink, teknologi yang memungkinkan Anda mengakses AWS layanan secara pribadi dengan menggunakan alamat IP pribadi. Untuk informasi selengkapnya tentang membuat titik akhir VPC, lihat Membuat Endpoint Antarmuka.

Kebijakan titik akhir VPC adalah kebijakan sumber daya IAM yang dapat Anda lampirkan ke titik akhir VPC antarmuka untuk mengontrol akses ke titik akhir. Untuk informasi selengkapnya, lihat Mengontrol Akses ke Layanan dengan VPC Endpoints. Anda dapat menggunakan kebijakan endpoint untuk membatasi lalu lintas dari jaringan internal Anda untuk mengakses API pribadi Anda. Anda dapat memilih untuk mengizinkan atau melarang akses ke API pribadi tertentu yang dapat diakses melalui titik akhir VPC.

Kebijakan titik akhir VPC dapat digunakan bersama dengan kebijakan sumber daya API Gateway. Kebijakan sumber daya digunakan untuk menentukan prinsipal mana yang dapat mengakses API. Kebijakan endpoint menentukan API pribadi mana yang dapat dipanggil melalui titik akhir VPC. Untuk informasi selengkapnya tentang kebijakan sumber daya, lihatthe section called “Menggunakan kebijakan sumber daya API Gateway”.

Pertimbangan kebijakan titik akhir VPC

- Jika kebijakan membatasi prinsip IAM, Anda harus menyetel metode ke atauauthorizationType. AWS_IAM NONE
- Identitas invoker dievaluasi berdasarkan nilai Authorization header. Tergantung pada AndaauthorizationType, ini dapat menyebabkan kesalahan 403 IncompleteSignatureException atau 403 InvalidSignatureException kesalahan. Tabel berikut menunjukkan nilai Authorization header untuk masing-masingauthorizationType.

authorizationType	**Authorization** header dievaluasi?	Nilai **Authorization** header yang diizinkan
NONEdengan kebijakan akses penuh default	Tidak	Tidak lulus
NONEdengan kebijakan akses khusus	Ya	Harus berupa nilai SiGv4 yang valid
IAM	Ya	Harus berupa nilai SiGv4 yang valid
CUSTOM atau COGNITO_USER_POOLS	Tidak	Tidak lulus

Contoh kebijakan titik akhir VPC

Anda dapat membuat kebijakan untuk titik akhir Amazon Virtual Private Cloud untuk Amazon API Gateway yang dapat Anda tentukan:

- Principal yang dapat melakukan tindakan.
- Tindakan yang dapat dilakukan.
- Sumber daya yang dapat memiliki tindakan yang dilakukan pada mereka.

Untuk melampirkan kebijakan ke titik akhir VPC, Anda harus menggunakan konsol VPC. Untuk informasi selengkapnya, lihat Mengontrol Akses ke Layanan dengan VPC Endpoints.

Contoh 1: Kebijakan titik akhir VPC yang memberikan akses ke dua API

Contoh kebijakan berikut hanya memberikan akses ke dua API tertentu melalui titik akhir VPC tempat kebijakan dilampirkan.

```
{
    "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/*"
            ]
        }
    ]
}
```

Contoh 2: Kebijakan titik akhir VPC yang memberikan akses ke metode GET

Contoh kebijakan berikut memberi pengguna akses ke GET metode untuk API tertentu melalui titik akhir VPC tempat kebijakan dilampirkan.

```
{
    "Statement": [
        {
            "Principal": "*",
            "Action": [
                "execute-api:Invoke"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:execute-api:us-east-1:123412341234:a1b2c3d4e5/stageName/GET/*"
            ]
        }
    ]
}
```

Contoh 3: Kebijakan titik akhir VPC yang memberikan akses pengguna tertentu ke API tertentu

Contoh kebijakan berikut memberikan akses pengguna tertentu ke API tertentu melalui titik akhir VPC tempat kebijakan dilampirkan.

```
{
    "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/*"
            ]
        }
    ]
}
```

Menggunakan tag untuk mengontrol akses ke REST API di API Gateway

Izin untuk mengakses REST API dapat disesuaikan dengan menggunakan kontrol akses berbasis atribut dalam kebijakan IAM.

Untuk informasi selengkapnya, lihat the section called “Kontrol akses berbasis atribut”.

Gunakan otorisasi API Gateway Lambda

Authorizer Lambda (sebelumnya dikenal sebagai otorisasi khusus) adalah fitur API Gateway yang menggunakan fungsi Lambda untuk mengontrol akses ke API Anda.

Authorizer Lambda berguna jika Anda ingin menerapkan skema otorisasi khusus yang menggunakan strategi otentikasi token pembawa seperti OAuth atau SALL, atau yang menggunakan parameter permintaan untuk menentukan identitas pemanggil.

Saat klien membuat permintaan ke salah satu metode API Anda, API Gateway memanggil otorisasi Lambda Anda, yang mengambil identitas pemanggil sebagai input dan mengembalikan kebijakan IAM sebagai output.

Ada dua jenis otorisasi Lambda:

- Lambda authorizer berbasis token (juga disebut `TOKEN` authorizer) menerima identitas penelepon dalam token pembawa, seperti JSON Web Token (JWT) atau token OAuth. Untuk contoh aplikasi, lihat Open Banking Brazil - Sampel Otorisasi di GitHub.
- Authorizer Lambda berbasis parameter permintaan (juga disebut `REQUEST` authorizer) menerima identitas pemanggil dalam kombinasi header, parameter string kueri, dan variabel. `stageVariables$context`

  Untuk WebSocket API, hanya otorisasi berbasis parameter permintaan yang didukung.

Dimungkinkan untuk menggunakan AWS Lambda fungsi dari AWS akun yang berbeda dari yang Anda buat API. Untuk informasi selengkapnya, lihat the section called “Konfigurasikan otorisasi Lambda lintas akun”.

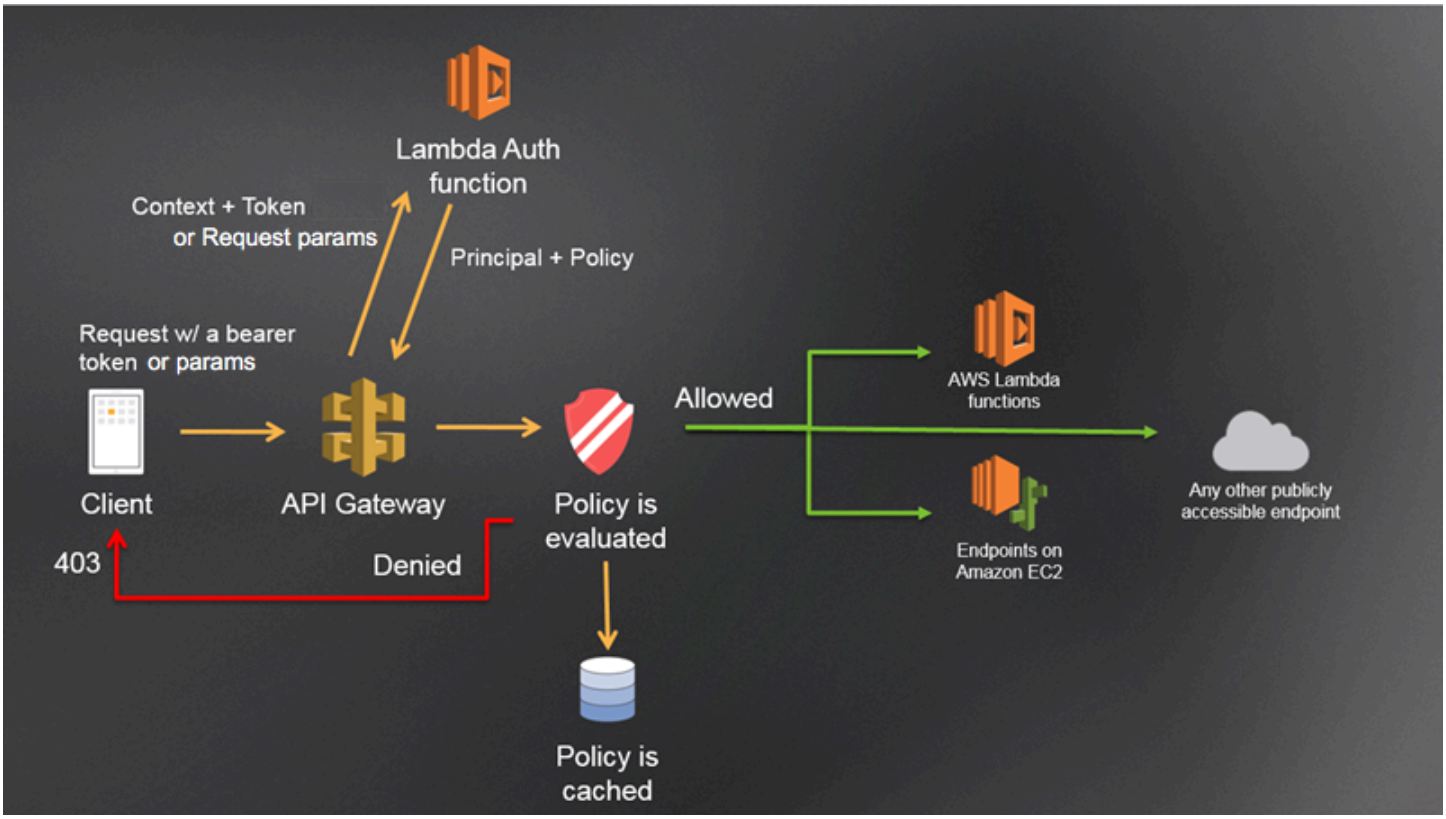
Misalnya fungsi Lambda, lihat aws-apigateway-lambda-authorizer-blueprints on. GitHub

Topik

- Alur kerja Auth otorisasi Lambda
- Langkah-langkah untuk membuat otorisasi API Gateway Lambda
- Buat fungsi otorisasi API Gateway Lambda di konsol Lambda
- Konfigurasikan otorisasi Lambda menggunakan konsol API Gateway
- Masukan ke otorisasi Lambda Amazon API Gateway
- Keluaran dari otorisasi Lambda Amazon API Gateway
- Panggil API dengan otorisasi API Gateway Lambda
- Konfigurasikan otorisasi Lambda lintas akun

Alur kerja Auth otorisasi Lambda

Diagram berikut menggambarkan alur kerja otorisasi untuk otorisasi Lambda.



Alur kerja otorisasi API Gateway Lambda

1. Klien memanggil metode pada metode API Gateway API, meneruskan token pembawa atau parameter permintaan.
2. API Gateway memeriksa apakah otorisasi Lambda dikonfigurasi untuk metode tersebut. Jika ya, API Gateway memanggil fungsi Lambda.
3. Fungsi Lambda mengautentikasi pemanggil dengan cara seperti berikut:
   - Memanggil penyedia OAuth untuk mendapatkan token akses OAuth.
   - Memanggil penyedia SAFL untuk mendapatkan pernyataan SAFL.
   - Membuat kebijakan IAM berdasarkan nilai parameter permintaan.
   - Mengambil kredensi dari database.
4. Jika panggilan berhasil, fungsi Lambda memberikan akses dengan mengembalikan objek keluaran yang berisi setidaknya kebijakan IAM dan pengidentifikasi utama.
5. API Gateway mengevaluasi kebijakan.
   - Jika akses ditolak, API Gateway mengembalikan kode status HTTP yang sesuai, seperti`403 ACCESS_DENIED`.

- Jika akses diizinkan, API Gateway akan memanggil metode. Jika caching diaktifkan di pengaturan otorisasi, API Gateway juga menyimpan kebijakan agar fungsi otorisasi Lambda tidak perlu dipanggil lagi.

6. Panggilan bisa gagal jika fungsi Lambda mengembalikan respons. `401 Unauthorized` Anda dapat menyesuaikan respons `401 Unauthorized` gateway. Untuk mempelajari selengkapnya, lihat the section called “Tanggapan Gateway”.

Langkah-langkah untuk membuat otorisasi API Gateway Lambda

Untuk membuat otorisasi Lambda, Anda perlu melakukan tugas-tugas berikut:

1. Buat fungsi otorisasi Lambda di konsol Lambda seperti yang dijelaskan dalam. the section called “Buat fungsi otorisasi Lambda di konsol Lambda” Anda dapat menggunakan salah satu contoh cetak biru sebagai titik awal dan menyesuaikan input dan output sesuai keinginan.
2. Konfigurasikan fungsi Lambda sebagai otorisasi API Gateway dan konfigurasikan metode API untuk memerlukannya, seperti yang dijelaskan dalam. the section called “Konfigurasikan otorisasi Lambda menggunakan konsol” Atau, jika Anda memerlukan otorisasi Lambda lintas akun, lihat. the section called “Konfigurasikan otorisasi Lambda lintas akun”

   > **Note**
   >
   > Anda juga dapat mengonfigurasi otorisasi dengan menggunakan AWS CLI atau AWS SDK.

3. Uji otorisasi Anda dengan menggunakan Tukang Pos seperti yang dijelaskan dalam. the section called “Panggil API dengan otorisasi Lambda”

Buat fungsi otorisasi API Gateway Lambda di konsol Lambda

Sebelum mengonfigurasi otorisasi Lambda, Anda harus terlebih dahulu membuat fungsi Lambda yang mengimplementasikan logika untuk mengotorisasi dan, jika perlu, untuk mengautentikasi pemanggil. Konsol Lambda menyediakan cetak biru Python, yang dapat Anda gunakan dengan memilih Gunakan cetak biru dan memilih cetak biru. api-gateway-authorizer-python Jika tidak, Anda akan ingin menggunakan salah satu cetak biru di GitHub repositori awslabs sebagai titik awal.

Misalnya fungsi otorisasi Lambda di bagian ini, yang tidak memanggil layanan lain, Anda dapat menggunakan built-in. `AWSLambdaBasicExecutionRole` Saat membuat fungsi Lambda untuk

otorisasi Lambda API Gateway Anda sendiri, Anda harus menetapkan peran eksekusi IAM ke fungsi Lambda jika memanggil layanan lain. AWS Untuk membuat peran, ikuti instruksi dalam Peran AWS Lambda Eksekusi.

Untuk contoh fungsi Lambda lainnya, lihat aws-apigateway-lambda-authorizer-blueprints on. GitHub Untuk contoh aplikasi, lihat Open Banking Brazil - Sampel Otorisasi di GitHub.

CONTOH: Buat fungsi otorisasi Lambda berbasis token

Untuk membuat fungsi otorisasi Lambda berbasis token, masukkan kode Node.js berikut untuk runtime terbaru di konsol Lambda. Kemudian, Anda menguji otorisasi di konsol API Gateway.

Untuk membuat fungsi otorisasi Lambda berbasis token

1. Di konsol Lambda, pilih Buat fungsi.
2. Pilih Tulis dari awal.
3. Masukkan nama untuk fungsi tersebut.
4. Untuk Runtime, pilih runtime Node.js atau Python terbaru yang didukung.
5. Pilih Buat fungsi.
6. Salin/tempel kode berikut ke editor kode.

   Node.js

```
// A simple token-based authorizer example to demonstrate how to use an
 authorization token
// to allow or deny a request. In this example, the caller named 'user' is
 allowed to invoke
// a request if the client-supplied token value is 'allow'. The caller is not
 allowed to invoke
// the request if the token value is 'deny'. If the token value is
 'unauthorized' or an empty
// string, the authorizer function returns an HTTP 401 status code. For any
 other token value,
// the authorizer returns an HTTP 500 status code.
// Note that token values are case-sensitive.

export const handler =  function(event, context, callback) {
    var token = event.authorizationToken;
    switch (token) {
        case 'allow':
            callback(null, generatePolicy('user', 'Allow', event.methodArn));
```

```
            break;
        case 'deny':
            callback(null, generatePolicy('user', 'Deny', event.methodArn));
            break;
        case 'unauthorized':
            callback("Unauthorized");   // Return a 401 Unauthorized response
            break;
        default:
            callback("Error: Invalid token"); // Return a 500 Invalid token
 response
    }
};

// Help function to generate an IAM policy
var generatePolicy = function(principalId, effect, resource) {
    var authResponse = {};

    authResponse.principalId = principalId;
    if (effect && resource) {
        var policyDocument = {};
        policyDocument.Version = '2012-10-17';
        policyDocument.Statement = [];
        var statementOne = {};
        statementOne.Action = 'execute-api:Invoke';
        statementOne.Effect = effect;
        statementOne.Resource = resource;
        policyDocument.Statement[0] = statementOne;
        authResponse.policyDocument = policyDocument;
    }

    // Optional output with custom properties of the String, Number or Boolean
 type.
    authResponse.context = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": true
    };
    return authResponse;
}
```

Python

```
# A simple token-based authorizer example to demonstrate how to use an
 authorization token
# to allow or deny a request. In this example, the caller named 'user' is
 allowed to invoke
# a request if the client-supplied token value is 'allow'. The caller is not
 allowed to invoke
# the request if the token value is 'deny'. If the token value is 'unauthorized'
 or an empty
# string, the authorizer function returns an HTTP 401 status code. For any other
 token value,
# the authorizer returns an HTTP 500 status code.
# Note that token values are case-sensitive.

import json


def lambda_handler(event, context):
    token = event['authorizationToken']
    if token == 'allow':
        print('authorized')
        response = generatePolicy('user', 'Allow', event['methodArn'])
    elif token == 'deny':
        print('unauthorized')
        response = generatePolicy('user', 'Deny', event['methodArn'])
    elif token == 'unauthorized':
        print('unauthorized')
        raise Exception('Unauthorized')  # Return a 401 Unauthorized response
        return 'unauthorized'
    try:
        return json.loads(response)
    except BaseException:
        print('unauthorized')
        return 'unauthorized'  # Return a 500 error


def generatePolicy(principalId, effect, resource):
    authResponse = {}
    authResponse['principalId'] = principalId
    if (effect and resource):
        policyDocument = {}
        policyDocument['Version'] = '2012-10-17'
```

```
        policyDocument['Statement'] = []
        statementOne = {}
        statementOne['Action'] = 'execute-api:Invoke'
        statementOne['Effect'] = effect
        statementOne['Resource'] = resource
        policyDocument['Statement'] = [statementOne]
        authResponse['policyDocument'] = policyDocument
    authResponse['context'] = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": True
    }
    authResponse_JSON = json.dumps(authResponse)
    return authResponse_JSON
```

7. Pilih Deploy.

Setelah membuat fungsi Lambda, Anda membuat dan menguji otorisasi Lambda berbasis token di konsol API Gateway.

Untuk membuat otorisasi Lambda berbasis token

1. Di konsol API Gateway, buat API sederhana jika Anda belum memilikinya.
2. Pilih API Anda dari daftar API.
3. Pilih Authorizer.
4. Pilih Buat Authorizer.
5. Untuk nama Authorizer, masukkan nama.
6. Untuk jenis Authorizer, pilih Lambda.
7. Untuk fungsi Lambda, pilih Wilayah AWS tempat Anda membuat fungsi otorisasi Lambda Anda, lalu masukkan nama fungsi.
8. Biarkan peran panggilan Lambda kosong.
9. Untuk payload acara Lambda, pilih Token.
10. Untuk sumber Token, masukkan**`authorizationToken`**.
11. Pilih Buat Authorizer.

Untuk menguji otorisasi Anda

1. Pilih nama otorisasi.
2. Di bawah Test authorizer, untuk nilai AuthorizationToken, masukkan. **`allow`**
3. Pilih Test Authorizer.

Dalam contoh ini, ketika API menerima permintaan metode, API Gateway meneruskan token sumber ke fungsi otorisasi Lambda ini di atribut. `event.authorizationToken` Fungsi otorisasi Lambda membaca token dan bertindak sebagai berikut:

- Jika nilai token`'allow'`, pengujian fungsi authorizer mengembalikan respons `200 OK` HTTP dan kebijakan IAM yang terlihat seperti berikut, dan permintaan metode berhasil:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "execute-api:Invoke",
      "Effect": "Allow",
      "Resource": "arn:aws:execute-api:us-east-1:123456789012:ivdtdhp7b5/
ESTestInvoke-stage/GET/"
    }
  ]
}
```

- Jika nilai token`'deny'`, pengujian fungsi authorizer mengembalikan respons `200 OK` HTTP dan kebijakan `Deny` IAM yang terlihat seperti berikut, dan permintaan metode gagal:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "execute-api:Invoke",
      "Effect": "Deny",
      "Resource": "arn:aws:execute-api:us-east-1:123456789012:ivdtdhp7b5/
ESTestInvoke-stage/GET/"
    }
  ]
}
```

> **Note**
>
> Di luar lingkungan pengujian, fungsi authorizer mengembalikan respons `403 Forbidden` HTTP dan permintaan metode gagal.

- Jika nilai token `'unauthorized'` atau string kosong, pengujian fungsi authorizer mengembalikan respons `401 Unauthorized` HTTP, dan panggilan metode gagal.
- Jika token adalah hal lain, klien menerima `500 Invalid token` respons, dan panggilan metode gagal.

> **Note**
>
> Dalam kode produksi, Anda mungkin perlu mengautentikasi pengguna sebelum memberikan otorisasi. Jika demikian, Anda dapat menambahkan logika otentikasi dalam fungsi Lambda juga dengan memanggil penyedia otentikasi seperti yang diarahkan dalam dokumentasi untuk penyedia tersebut.

Selain mengembalikan kebijakan IAM, fungsi otorisasi Lambda juga harus mengembalikan pengenal utama pemanggil. Ini juga dapat secara opsional mengembalikan `context` objek yang berisi informasi tambahan yang dapat diteruskan ke backend integrasi. Untuk informasi selengkapnya, lihat Keluaran dari otorisasi Lambda Amazon API Gateway.

CONTOH: Buat fungsi otorisasi Lambda berbasis permintaan

Untuk membuat fungsi otorisasi Lambda berbasis permintaan, masukkan kode Node.js berikut untuk runtime terbaru di konsol Lambda. Kemudian, Anda menguji otorisasi di konsol API Gateway.

1. Di konsol Lambda, pilih Buat fungsi.
2. Pilih Tulis dari awal.
3. Masukkan nama untuk fungsi tersebut.
4. Untuk Runtime, pilih runtime Node.js atau Python terbaru yang didukung.
5. Pilih Buat fungsi.
6. Salin/tempel kode berikut ke editor kode.

Node.js

```
   // A simple request-based authorizer example to demonstrate how to use
 request
    // parameters to allow or deny a request. In this example, a request is
    // authorized if the client-supplied headerauth1 header, QueryString1
    // query parameter, and stage variable of StageVar1 all match
    // specified values of 'headerValue1', 'queryValue1', and 'stageValue1',
    // respectively.

export const handler = function(event, context, callback) {
    console.log('Received event:', JSON.stringify(event, null, 2));

    // Retrieve request parameters from the Lambda function input:
    var headers = event.headers;
    var queryStringParameters = event.queryStringParameters;
    var pathParameters = event.pathParameters;
    var stageVariables = event.stageVariables;

    // Parse the input for the parameter values
    var tmp = event.methodArn.split(':');
    var apiGatewayArnTmp = tmp[5].split('/');
    var awsAccountId = tmp[4];
    var region = tmp[3];
    var restApiId = apiGatewayArnTmp[0];
    var stage = apiGatewayArnTmp[1];
    var method = apiGatewayArnTmp[2];
    var resource = '/'; // root resource
    if (apiGatewayArnTmp[3]) {
        resource += apiGatewayArnTmp[3];
    }

    // Perform authorization to return the Allow policy for correct parameters
 and
    // the 'Unauthorized' error, otherwise.
    var authResponse = {};
    var condition = {};
    condition.IpAddress = {};

    if (headers.headerauth1 === "headerValue1"
        && queryStringParameters.QueryString1 === "queryValue1"
        && stageVariables.StageVar1 === "stageValue1") {
        callback(null, generateAllow('me', event.methodArn));
```

```
    }  else {
        callback("Unauthorized");
    }
}

// Help function to generate an IAM policy
var generatePolicy = function(principalId, effect, resource) {
    // Required output:
    var authResponse = {};
    authResponse.principalId = principalId;
    if (effect && resource) {
        var policyDocument = {};
        policyDocument.Version = '2012-10-17'; // default version
        policyDocument.Statement = [];
        var statementOne = {};
        statementOne.Action = 'execute-api:Invoke'; // default action
        statementOne.Effect = effect;
        statementOne.Resource = resource;
        policyDocument.Statement[0] = statementOne;
        authResponse.policyDocument = policyDocument;
    }
    // Optional output with custom properties of the String, Number or Boolean
 type.
    authResponse.context = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": true
    };
    return authResponse;
}

var generateAllow = function(principalId, resource) {
    return generatePolicy(principalId, 'Allow', resource);
}

var generateDeny = function(principalId, resource) {
    return generatePolicy(principalId, 'Deny', resource);
}
```

Python

```
# A simple request-based authorizer example to demonstrate how to use request
# parameters to allow or deny a request. In this example, a request is
```

```
# authorized if the client-supplied headerauth1 header, QueryString1
# query parameter, and stage variable of StageVar1 all match
# specified values of 'headerValue1', 'queryValue1', and 'stageValue1',
# respectively.

import json


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

    # Retrieve request parameters from the Lambda function input:
    headers = event['headers']
    queryStringParameters = event['queryStringParameters']
    pathParameters = event['pathParameters']
    stageVariables = event['stageVariables']

    # Parse the input for the parameter values
    tmp = event['methodArn'].split(':')
    apiGatewayArnTmp = tmp[5].split('/')
    awsAccountId = tmp[4]
    region = tmp[3]
    restApiId = apiGatewayArnTmp[0]
    stage = apiGatewayArnTmp[1]
    method = apiGatewayArnTmp[2]
    resource = '/'

    if (apiGatewayArnTmp[3]):
        resource += apiGatewayArnTmp[3]

    # Perform authorization to return the Allow policy for correct parameters
    # and the 'Unauthorized' error, otherwise.

    authResponse = {}
    condition = {}
    condition['IpAddress'] = {}

    if (headers['headerauth1'] == "headerValue1" and
 queryStringParameters["QueryString1"]
            == "queryValue1" and stageVariables["StageVal1"] == "stageValue1"):
        response = generateAllow('me', event['methodArn'])
        print('authorized')
        return json.loads(response)
    else:
```

```
        print('unauthorized')
        raise Exception('Unauthorized') # Return a 401 Unauthorized response
        return 'unauthorized'

    # Help function to generate IAM policy


def generatePolicy(principalId, effect, resource):
    authResponse = {}
    authResponse['principalId'] = principalId
    if (effect and resource):
        policyDocument = {}
        policyDocument['Version'] = '2012-10-17'
        policyDocument['Statement'] = []
        statementOne = {}
        statementOne['Action'] = 'execute-api:Invoke'
        statementOne['Effect'] = effect
        statementOne['Resource'] = resource
        policyDocument['Statement'] = [statementOne]
        authResponse['policyDocument'] = policyDocument

    authResponse['context'] = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": True
    }

    authResponse_JSON = json.dumps(authResponse)

    return authResponse_JSON


def generateAllow(principalId, resource):
    return generatePolicy(principalId, 'Allow', resource)


def generateDeny(principalId, resource):
    return generatePolicy(principalId, 'Deny', resource)
```

7. Pilih Deploy.

Setelah membuat fungsi Lambda, Anda membuat dan menguji otorisasi Lambda berbasis permintaan di konsol API Gateway.

Untuk membuat otorisasi Lambda berbasis permintaan

1. Di konsol API Gateway, buat API sederhana jika Anda belum memilikinya.
2. Pilih API Anda dari daftar API.
3. Pilih Authorizer.
4. Pilih Buat Authorizer.
5. Untuk nama Authorizer, masukkan nama.
6. Untuk jenis Authorizer, pilih Lambda.
7. Untuk fungsi Lambda, pilih Wilayah AWS tempat Anda membuat fungsi otorisasi Lambda Anda, lalu masukkan nama fungsi.
8. Biarkan peran panggilan Lambda kosong.
9. Untuk payload acara Lambda, pilih Minta.
10. Di bawah Jenis sumber Identitas, masukkan yang berikut ini:
    a. Pilih Header dan enter**`headerauth1`**, lalu pilih Tambah parameter.
    b. Di bawah Jenis sumber identitas, pilih String kueri dan masukkan**`QueryString1`**, lalu pilih Tambah parameter.
    c. Di bawah Identity source type, pilih Stage variable dan enter**`StageVar1`**.
11. Pilih Buat Authorizer.

Untuk menguji otorisasi Anda

1. Pilih nama otorisasi.
2. Di bawah Pengotorisasi uji, masukkan yang berikut ini:
    a. Pilih Header dan enter**`headerValue1`**, lalu pilih Tambah parameter.
    b. Di bawah Jenis sumber identitas, pilih String kueri dan masukkan**`queryValue1`**, lalu pilih Tambah parameter.
    c. Di bawah Identity source type, pilih Stage variable dan enter**`stageValue1`**.
3. Pilih Test Authorizer.

Dalam contoh ini, fungsi Lambda Authorizer memeriksa parameter input dan bertindak sebagai berikut:

- Jika semua nilai parameter yang diperlukan cocok dengan nilai yang diharapkan, fungsi authorizer mengembalikan respons `200 OK` HTTP dan kebijakan IAM yang terlihat seperti berikut, dan permintaan metode berhasil:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "execute-api:Invoke",
      "Effect": "Allow",
      "Resource": "arn:aws:execute-api:us-east-1:123456789012:ivdtdhp7b5/
ESTestInvoke-stage/GET/"
    }
  ]
}
```

- Jika tidak, fungsi authorizer mengembalikan respons `401 Unauthorized` HTTP, dan panggilan metode gagal.

> **Note**
>
> Dalam kode produksi, Anda mungkin perlu mengautentikasi pengguna sebelum memberikan otorisasi. Jika demikian, Anda dapat menambahkan logika otentikasi dalam fungsi Lambda juga dengan memanggil penyedia otentikasi seperti yang diarahkan dalam dokumentasi untuk penyedia tersebut.

Selain mengembalikan kebijakan IAM, fungsi otorisasi Lambda juga harus mengembalikan pengenal utama pemanggil. Ini juga dapat secara opsional mengembalikan `context` objek yang berisi informasi tambahan yang dapat diteruskan ke backend integrasi. Untuk informasi selengkapnya, lihat Keluaran dari otorisasi Lambda Amazon API Gateway.

Konfigurasikan otorisasi Lambda menggunakan konsol API Gateway

Setelah Anda membuat fungsi Lambda dan memverifikasi apakah fungsi tersebut berfungsi, gunakan langkah-langkah berikut untuk mengonfigurasi otorisasi API Gateway Lambda (sebelumnya dikenal sebagai otorisasi khusus) di konsol API Gateway.

Untuk mengonfigurasi otorisasi Lambda menggunakan konsol API Gateway

1. Masuk ke konsol API Gateway.
2. Buat API baru atau pilih API yang sudah ada, lalu pilih Authorizers.
3. Pilih Buat Authorizer.
4. Untuk nama Authorizer, masukkan nama untuk otorisasi.
5. Untuk jenis Authorizer, pilih Lambda.
6. Untuk fungsi Lambda, pilih Wilayah AWS tempat Anda membuat fungsi otorisasi Lambda Anda, lalu masukkan nama fungsi.
7. Biarkan peran panggilan Lambda kosong untuk membiarkan konsol API Gateway menyetel kebijakan berbasis sumber daya. Kebijakan ini memberikan izin API Gateway untuk menjalankan fungsi Lambda otorisasi. Anda juga dapat memilih untuk memasukkan nama peran IAM untuk mengizinkan API Gateway menjalankan fungsi Lambda otorisasi. Untuk contoh peran seperti itu, lihatBuat peran IAM yang dapat diasumsikan.
8. Untuk muatan acara Lambda, pilih Token untuk otorisasi atau Permintaan **`TOKEN`** otorisasi. `REQUEST` (Ini sama dengan menyetel properti tipe ke `TOKEN` atau`REQUEST`.)
9. Bergantung pada pilihan langkah sebelumnya, lakukan salah satu hal berikut:

   a. Untuk opsi Token, lakukan hal berikut:

      - Untuk sumber Token, masukkan nama header yang berisi token otorisasi. Klien API harus menyertakan header dari nama ini untuk mengirim token otorisasi ke otorisasi Lambda.
      - Secara opsional, untuk validasi Token, masukkan pernyataan RegEx . API Gateway melakukan validasi awal token input terhadap ekspresi ini dan memanggil authorizer setelah validasi berhasil. Ini membantu mengurangi panggilan ke API Anda.
      - Untuk men-cache kebijakan otorisasi yang dihasilkan oleh otorisasi, biarkan caching Otorisasi tetap aktif. Saat caching kebijakan diaktifkan, Anda dapat memilih untuk mengubah nilai TTL. Menyetel TTL ke nol menonaktifkan caching kebijakan. Saat caching kebijakan diaktifkan, nama header yang ditentukan dalam sumber Token menjadi kunci cache. Jika beberapa nilai diteruskan ke header ini dalam permintaan, semua nilai akan menjadi kunci cache, dengan urutan dipertahankan.

> **Note**
>
> Nilai TTL default adalah 300 detik. Nilai maksimum adalah 3600 detik, batas ini tidak dapat ditingkatkan.

b. Untuk opsi Permintaan, lakukan hal berikut:

- Untuk tipe sumber Identity, pilih tipe parameter. Jenis parameter yang didukung adalah`Header`,`Query string`,`Stage variable`, dan`Context`. Untuk menambahkan lebih banyak sumber identitas, pilih Tambah parameter.
- Untuk men-cache kebijakan otorisasi yang dihasilkan oleh otorisasi, biarkan caching Otorisasi tetap aktif. Saat caching kebijakan diaktifkan, Anda dapat memilih untuk mengubah nilai TTL. Menyetel TTL ke nol menonaktifkan caching kebijakan.

  API Gateway menggunakan sumber identitas yang ditentukan sebagai kunci caching otorisasi permintaan. Saat caching diaktifkan, API Gateway memanggil fungsi Lambda otorisasi hanya setelah berhasil memverifikasi bahwa semua sumber identitas yang ditentukan ada saat runtime. Jika sumber identifikasi tertentu hilang, null, atau kosong, API Gateway mengembalikan `401 Unauthorized` respons tanpa memanggil fungsi Lambda otorisasi.

  Ketika beberapa sumber identitas didefinisikan, mereka semua digunakan untuk mendapatkan kunci cache otorisasi. Mengubah salah satu bagian kunci cache menyebabkan otorisasi membuang dokumen kebijakan yang di-cache dan menghasilkan yang baru. Jika header dengan beberapa nilai diteruskan dalam permintaan, maka semua nilai akan menjadi bagian dari kunci cache, dengan urutan dipertahankan.
- Ketika caching dimatikan, tidak perlu menentukan sumber identitas. API Gateway langsung meneruskan permintaan ke fungsi Lambda otorisasi.

> **Note**
>
> Untuk mengaktifkan caching, otorisasi Anda harus mengembalikan kebijakan yang berlaku untuk semua metode di seluruh API. Untuk menerapkan kebijakan khusus metode, Anda dapat menonaktifkan caching Otorisasi.

10. Pilih Buat Authorizer.

11. Setelah authorizer dibuat untuk API, Anda dapat menguji authorizer sebelum dikonfigurasi pada metode. Untuk menguji otorisasi, pilih nama otorisasi.
12. a. Untuk `TOKEN` otorisasi, di bawah nilai Token, masukkan token yang valid. Pilih Test Authorizer. Token akan diteruskan ke fungsi Lambda sebagai header yang Anda tentukan dalam pengaturan sumber Token dari otorisasi.

    b. Untuk `REQUEST` authorizer, di bawah Identity source type, pilih tipe parameter dan masukkan nilai. Untuk menambahkan lebih banyak parameter, pilih Tambah parameter. Pilih Test Authorizer.

Selain menggunakan konsol API Gateway, Anda dapat menggunakan AWS CLI atau AWS SDK untuk API Gateway untuk menguji pemanggilan otorisasi. Untuk melakukannya menggunakan AWS CLI, lihat test-invoke-authorizer.

> **Note**
>
> Test-invoke untuk eksekusi metode test-invoke untuk authorizer adalah proses independen.
> Untuk menguji pemanggilan metode menggunakan konsol API Gateway, lihatGunakan konsol untuk menguji metode REST API. Untuk menguji pemanggilan metode menggunakan AWS CLI, lihat test-invoke-method.
> Untuk menguji pemanggilan metode dan otorisasi yang dikonfigurasi, gunakan API, lalu gunakan cURL atau Postman untuk memanggil metode, dengan menyediakan parameter token atau permintaan yang diperlukan.

Prosedur selanjutnya menunjukkan cara mengonfigurasi metode API untuk menggunakan otorisasi Lambda.

Untuk mengonfigurasi metode API untuk menggunakan otorisasi Lambda

1. Pilih Sumber daya. Pilih metode baru atau pilih metode yang ada. Jika perlu, buat sumber daya baru.
2. Pada tab Permintaan metode, di bawah Pengaturan permintaan metode, pilih Edit.
3. Untuk Authorizer, dari menu dropdown, pilih Lambda Authorizer yang baru saja Anda buat.
4. (Opsional) Jika Anda ingin meneruskan token otorisasi ke backend, pilih header permintaan HTTP. Pilih Tambahkan header, lalu tambahkan nama header otorisasi. Di Nama, masukkan

nama header yang cocok dengan nama sumber Token yang Anda tentukan saat Anda membuat otorisasi Lambda untuk API. Langkah ini tidak berlaku untuk `REQUEST` otorisasi.

5. Pilih Simpan.
6. Pilih Deploy API untuk menerapkan API ke panggung. Perhatikan nilai Invoke URL. Anda membutuhkannya saat memanggil API. Untuk `REQUEST` otorisasi yang menggunakan variabel tahap, Anda juga harus menentukan variabel tahap yang diperlukan dan menentukan nilainya saat berada di halaman Tahapan.

Masukan ke otorisasi Lambda Amazon API Gateway

`TOKEN`format masukan

Untuk Authorizer Lambda (sebelumnya dikenal sebagai otorisasi kustom) dari `TOKEN` jenisnya, Anda harus menentukan header kustom sebagai Sumber Token saat Anda mengonfigurasi otorisasi untuk API Anda. Klien API harus meneruskan token otorisasi yang diperlukan di header tersebut dalam permintaan yang masuk. Setelah menerima permintaan metode masuk, API Gateway mengekstrak token dari header kustom. Kemudian melewati token sebagai `authorizationToken` properti dari `event` objek fungsi Lambda, selain metode ARN sebagai properti: `methodArn`

```
{
    "type":"TOKEN",
    "authorizationToken":"{caller-supplied-token}",
    "methodArn":"arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/{httpVerb}/
[{resource}/[{child-resources}]]"
}
```

Dalam contoh ini, `type` properti menentukan jenis authorizer, yang merupakan `TOKEN` authorizer. *`{caller-supplied-token}`*Berasal dari header otorisasi dalam permintaan klien, dan dapat berupa nilai string apa pun. `methodArn`Ini adalah ARN dari permintaan metode yang masuk dan diisi oleh API Gateway sesuai dengan konfigurasi otorisasi Lambda.

`REQUEST`format masukan

Untuk `REQUEST` jenis pengotorisasi Lambda, API Gateway meneruskan parameter permintaan ke fungsi Lambda otorisasi sebagai bagian dari objek. `event` Parameter permintaan termasuk header, parameter jalur, parameter string kueri, variabel tahap, dan beberapa variabel konteks permintaan. Pemanggil API dapat mengatur parameter jalur, header, dan parameter string kueri. Pengembang API harus menyetel variabel stage selama penerapan API dan API Gateway menyediakan konteks permintaan pada waktu berjalan.

> **Note**
>
> Parameter jalur dapat diteruskan sebagai parameter permintaan ke fungsi otorisasi Lambda, tetapi tidak dapat digunakan sebagai sumber identitas.

Contoh berikut menunjukkan input ke `REQUEST` authorizer untuk metode API (`GET /request`) dengan integrasi proxy:

```
{
  "type": "REQUEST",
  "methodArn": "arn:aws:execute-api:us-east-1:123456789012:abcdef123/test/GET/request",
  "resource": "/request",
  "path": "/request",
  "httpMethod": "GET",
  "headers": {
    "X-AMZ-Date": "20170718T062915Z",
    "Accept": "*/*",
    "HeaderAuth1": "headerValue1",
    "CloudFront-Viewer-Country": "US",
    "CloudFront-Forwarded-Proto": "https",
    "CloudFront-Is-Tablet-Viewer": "false",
    "CloudFront-Is-Mobile-Viewer": "false",
    "User-Agent": "..."
  },
  "queryStringParameters": {
    "QueryString1": "queryValue1"
  },
  "pathParameters": {},
  "stageVariables": {
    "StageVar1": "stageValue1"
  },
  "requestContext": {
    "path": "/request",
    "accountId": "123456789012",
    "resourceId": "05c7jb",
    "stage": "test",
    "requestId": "...",
    "identity": {
      "apiKey": "...",
      "sourceIp": "...",
      "clientCert": {
```

```
        "clientCertPem": "CERT_CONTENT",
        "subjectDN": "www.example.com",
        "issuerDN": "Example issuer",
        "serialNumber": "a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1",
        "validity": {
          "notBefore": "May 28 12:30:02 2019 GMT",
          "notAfter": "Aug  5 09:36:04 2021 GMT"
        }
      }
    },
    "resourcePath": "/request",
    "httpMethod": "GET",
    "apiId": "abcdef123"
  }
}
```

`requestContext`Ini adalah peta pasangan kunci-nilai dan sesuai dengan variabel $context. Hasilnya bergantung pada API. API Gateway dapat menambahkan kunci baru ke peta. Untuk informasi selengkapnya tentang input fungsi Lambda dalam integrasi proxy Lambda, lihat. Format input fungsi Lambda untuk integrasi proxy

Keluaran dari otorisasi Lambda Amazon API Gateway

Output fungsi otorisasi Lambda adalah objek mirip kamus, yang harus menyertakan pengidentifikasi utama (`principalId`) dan dokumen kebijakan () yang berisi daftar pernyataan kebijakan. `policyDocument` Outputnya juga dapat menyertakan `context` peta yang berisi pasangan kunci-nilai. Jika API menggunakan paket penggunaan (`apiKeySource`disetel ke`AUTHORIZER`), fungsi otorisasi Lambda harus mengembalikan salah satu kunci API paket penggunaan sebagai nilai properti. `usageIdentifierKey`

Berikut ini menunjukkan contoh output ini.

```
{
  "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}"
}
```

Di sini, pernyataan kebijakan menentukan apakah akan mengizinkan atau menolak (`Effect`) layanan eksekusi API Gateway untuk memanggil (`Action`) metode API yang ditentukan (`Resource`). Anda dapat menggunakan wild card (`*`) untuk menentukan jenis sumber daya (metode). Untuk informasi tentang menyetel kebijakan yang valid untuk memanggil API, lihatReferensi pernyataan kebijakan IAM untuk menjalankan API di API Gateway.

Untuk metode ARN yang diaktifkan otorisasi, misalnya`aarn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/{httpVerb}/[{resource}/[{child-resources}]]`, panjang maksimum adalah 1600 byte. Nilai parameter jalur, ukuran yang ditentukan pada waktu berjalan, dapat menyebabkan panjang ARN melebihi batas. Ketika ini terjadi, klien API akan menerima `414 Request URI too long` respons.

Selain itu, ARN Sumber Daya, seperti yang ditunjukkan dalam keluaran pernyataan kebijakan oleh otorisasi, saat ini dibatasi hingga 512 karakter. Untuk alasan ini, Anda tidak boleh menggunakan URI dengan token JWT dengan panjang yang signifikan dalam URI permintaan. Anda dapat dengan aman meneruskan token JWT di header permintaan, sebagai gantinya.

Anda dapat mengakses `principalId` nilai dalam template pemetaan menggunakan `$context.authorizer.principalId` variabel. Ini berguna jika Anda ingin meneruskan nilai ke backend. Untuk informasi selengkapnya, lihat `$context`Variabel untuk model data, otorisasi, templat pemetaan, dan CloudWatch pencatatan akses.

Anda dapat mengaksesstringKey,numberKey, atau booleanKey nilai (misalnya,, `"value"`"1", atau`"true"`) `context` peta dalam templat pemetaan dengan memanggil`$context.authorizer.stringKey`,, atau `$context.authorizer.numberKey`$context.authorizer.booleanKey, masing-masing.

Nilai yang dikembalikan semuanya dirangkai. Perhatikan bahwa Anda tidak dapat mengatur objek atau array JSON sebagai nilai yang valid dari kunci apa pun di `context` peta.

Anda dapat menggunakan `context` peta untuk mengembalikan kredensi cache dari otorisasi ke backend, menggunakan templat pemetaan permintaan integrasi. Hal ini memungkinkan backend untuk memberikan pengalaman pengguna yang lebih baik dengan menggunakan kredensi cache untuk mengurangi kebutuhan untuk mengakses kunci rahasia dan membuka token otorisasi untuk setiap permintaan.

Untuk integrasi proxy Lambda, API Gateway meneruskan `context` objek dari otorisasi Lambda langsung ke fungsi Lambda backend sebagai bagian dari input. `event` Anda dapat mengambil pasangan `context` kunci-nilai dalam fungsi Lambda dengan memanggil. `$event.requestContext.authorizer.`*`key`*

`{api-key}`singkatan dari kunci API dalam rencana penggunaan tahap API. Untuk informasi selengkapnya, lihat the section called “Paket penggunaan”.

Berikut ini menunjukkan contoh output dari contoh Lambda authorizer. Output contoh berisi pernyataan kebijakan untuk memblokir (`Deny`) panggilan ke `GET` metode untuk `dev` tahap API (`ymy8tbxw7b`) dari AWS account (`123456789012`).

```
{
  "principalId": "user",
  "policyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Deny",
        "Resource": "arn:aws:execute-api:us-west-2:123456789012:ymy8tbxw7b/dev/GET/"
      }
    ]
  }
}
```

Panggil API dengan otorisasi API Gateway Lambda

Setelah mengonfigurasi otorisasi Lambda (sebelumnya dikenal sebagai otorisasi khusus) dan menerapkan API, Anda harus menguji API dengan otorisasi Lambda diaktifkan. Untuk ini, Anda memerlukan klien REST, seperti cURL atau Postman. Untuk contoh berikut, kami menggunakan Postman.

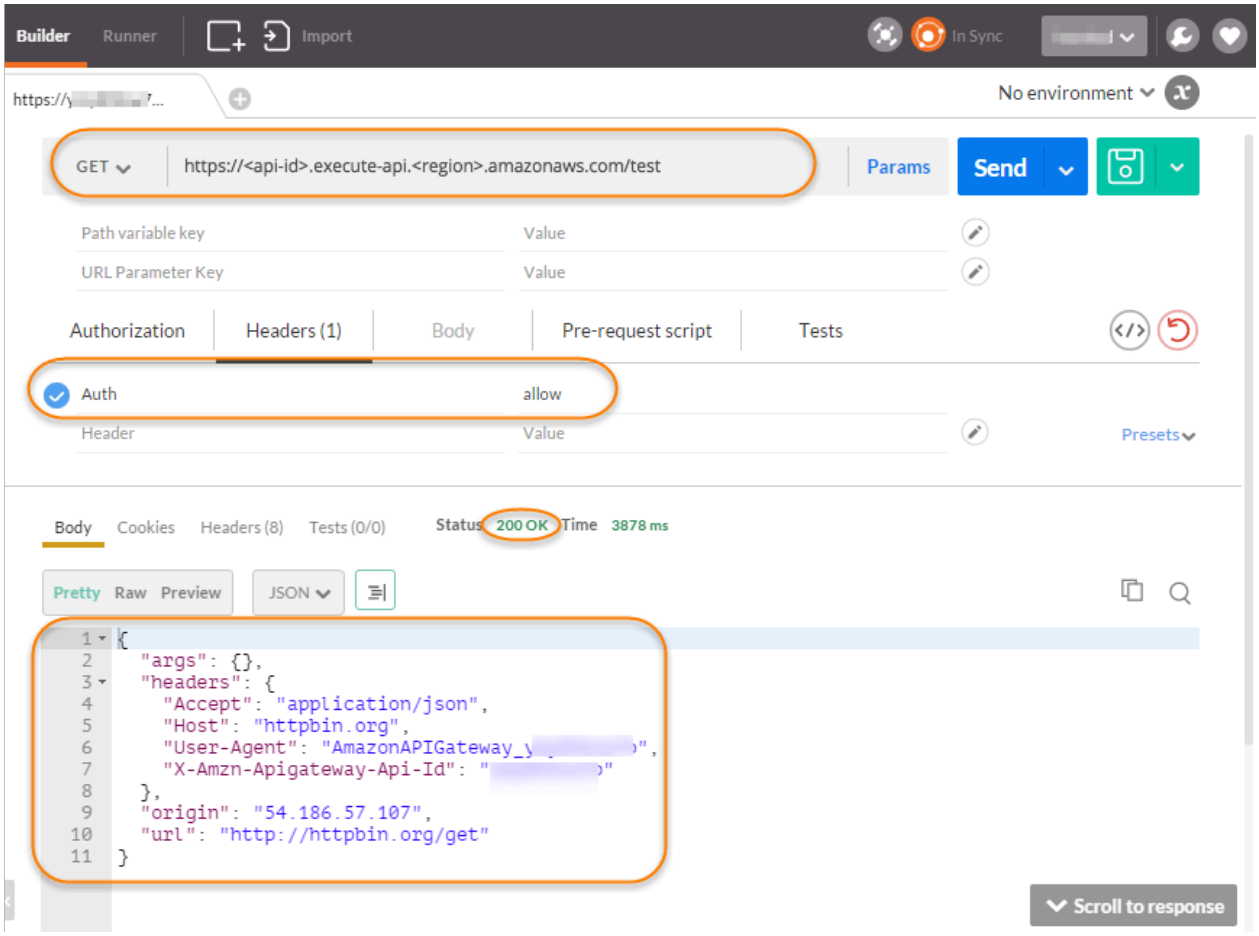
> **Note**
>
> Saat memanggil metode berkemampuan pengotorisasi, API Gateway tidak mencatat panggilan CloudWatch jika token yang diperlukan untuk **`TOKEN`** otorisasi tidak disetel, nol, atau tidak valid oleh ekspresi validasi Token yang ditentukan. Demikian pula, API Gateway tidak mencatat panggilan CloudWatch jika salah satu sumber identitas yang diperlukan untuk `REQUEST` otorisasi tidak disetel, nol, atau kosong.

Berikut ini, kami menunjukkan cara menggunakan Postman untuk memanggil atau menguji API dengan otorisasi `TOKEN` Lambda. Metode ini dapat diterapkan untuk memanggil API dengan `REQUEST` otorisasi Lambda, jika Anda menentukan parameter path, header, atau string kueri yang diperlukan secara eksplisit.

Untuk memanggil API dengan **`TOKEN`** otorisasi khusus

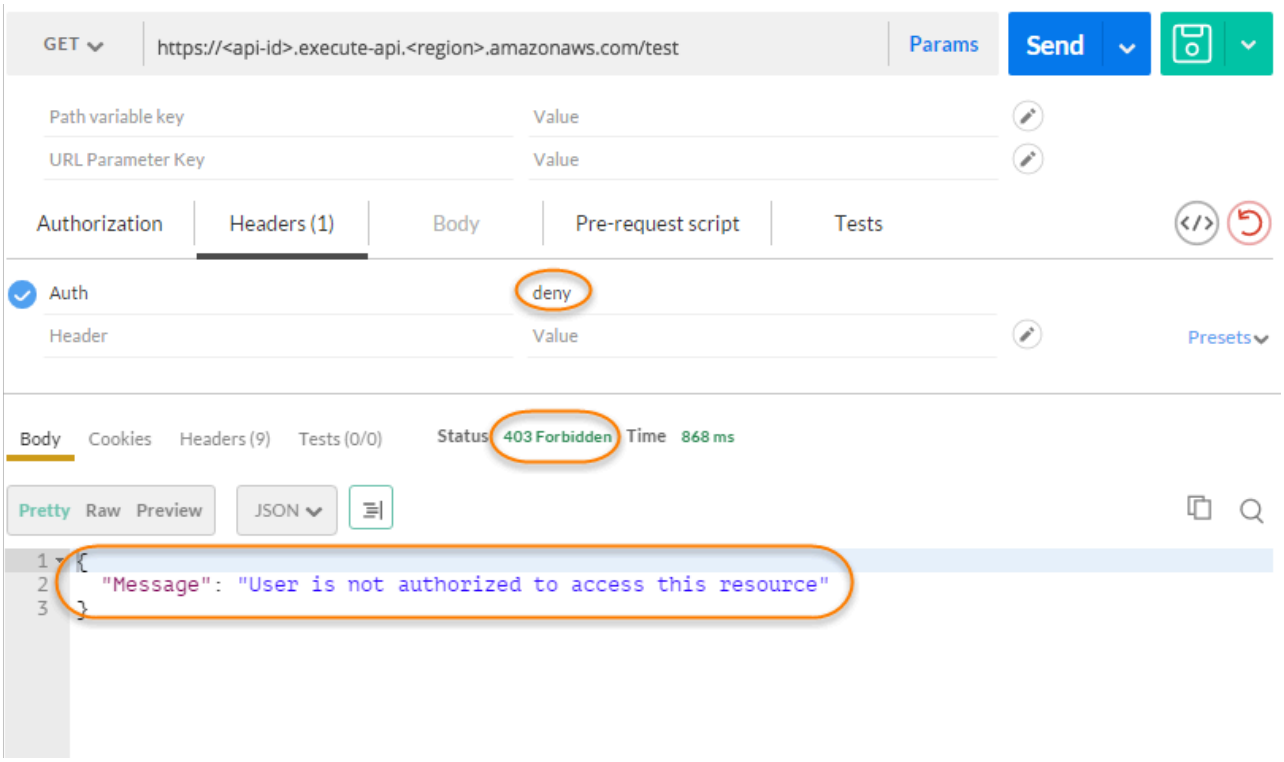
1. Buka Postman, pilih metode GET, dan tempel URL Invoke API ke bidang URL yang berdekatan.

   Tambahkan header token otorisasi Lambda dan atur nilainya. `allow` Pilih Kirim.



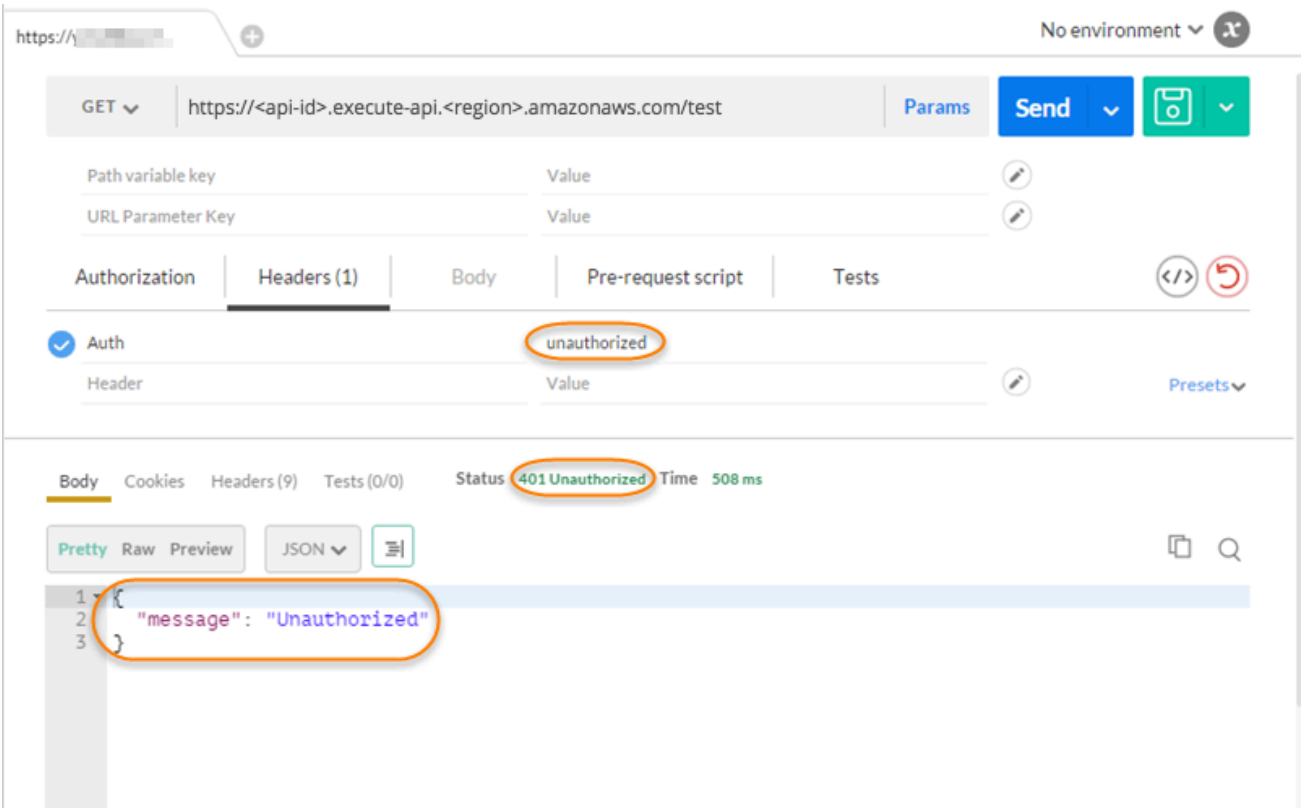
Tanggapan menunjukkan bahwa otorisasi API Gateway Lambda mengembalikan respons 200 OK dan berhasil mengotorisasi panggilan untuk mengakses titik akhir HTTP (http://httpbin.org/get) yang terintegrasi dengan metode.

2. Masih di Postman, ubah nilai header token otorisasi Lambda menjadi. deny Pilih Kirim.



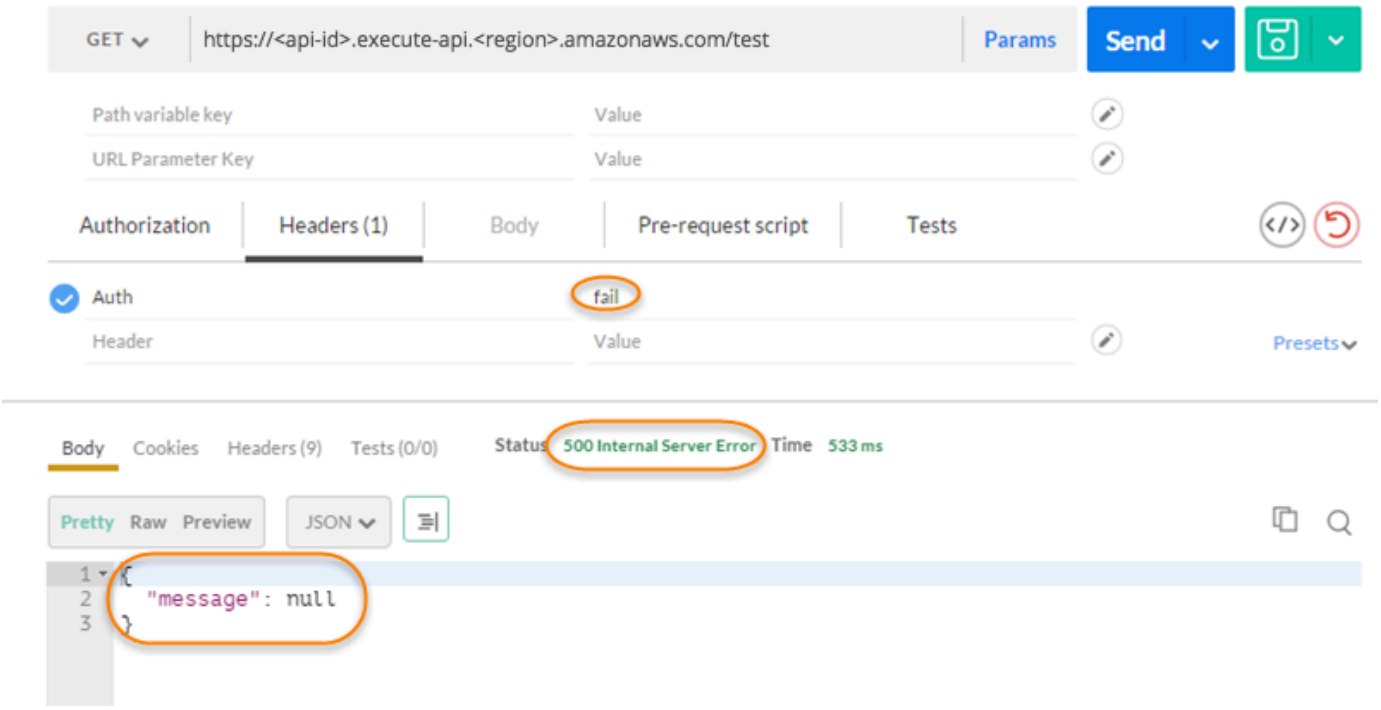
Tanggapan menunjukkan bahwa otorisasi API Gateway Lambda mengembalikan respons 403 Forbidden tanpa mengotorisasi panggilan untuk mengakses titik akhir HTTP.

3. Di Postman, ubah nilai **`unauthorized`** header token otorisasi Lambda menjadi dan pilih Kirim.



Respons menunjukkan bahwa API Gateway mengembalikan respons 401 Tidak Sah tanpa mengotorisasi panggilan untuk mengakses titik akhir HTTP.

4. Sekarang, ubah nilai header token otorisasi Lambda menjadi. `fail` Pilih Kirim.



Respons menunjukkan bahwa API Gateway mengembalikan respon 500 Internal Server Error tanpa mengotorisasi panggilan untuk mengakses endpoint HTTP.

Konfigurasikan otorisasi Lambda lintas akun

Anda sekarang juga dapat menggunakan AWS Lambda fungsi dari AWS akun yang berbeda sebagai fungsi otorisasi API Anda. Setiap akun dapat berada di wilayah mana pun di mana Amazon API Gateway tersedia. Fungsi otorisasi Lambda dapat menggunakan strategi otentikasi token pembawa seperti OAuth atau SALL. Ini memudahkan untuk mengelola dan berbagi fungsi otorisasi Lambda pusat secara terpusat di beberapa API Gateway API.

Di bagian ini, kami menunjukkan cara mengonfigurasi fungsi otorisasi Lambda lintas akun menggunakan konsol Amazon API Gateway.

Petunjuk ini mengasumsikan bahwa Anda sudah memiliki API Gateway API di satu AWS akun dan fungsi otorisasi Lambda di akun lain.

Konfigurasikan otorisasi Lambda lintas akun menggunakan konsol API Gateway

Masuk ke konsol Amazon API Gateway di akun yang memiliki API Anda di dalamnya, lalu lakukan hal berikut:

1. Pilih API Anda, lalu di panel navigasi utama, pilih Authorizers.
2. Pilih Buat Authorizer.
3. Untuk nama Authorizer, masukkan nama untuk otorisasi.
4. Untuk jenis Authorizer, pilih Lambda.
5. Untuk Fungsi Lambda, masukkan ARN lengkap untuk fungsi otorisasi Lambda yang Anda miliki di akun kedua Anda.

   > **Note**
   >
   > Di konsol Lambda, Anda dapat menemukan ARN untuk fungsi Anda di sudut kanan atas jendela konsol.

6. Peringatan dengan string `aws lambda add-permission` perintah akan muncul. Kebijakan ini memberikan izin API Gateway untuk menjalankan fungsi Lambda otorisasi. Salin perintah dan simpan untuk nanti. Anda menjalankan perintah setelah Anda membuat authorizer.

7. Biarkan peran panggilan Lambda kosong untuk membiarkan konsol API Gateway menyetel kebijakan berbasis sumber daya. Kebijakan ini memberikan izin API Gateway untuk menjalankan fungsi Lambda otorisasi. Anda juga dapat memilih untuk memasukkan peran IAM untuk mengizinkan API Gateway menjalankan fungsi Lambda otorisasi. Untuk contoh peran seperti itu, lihatBuat peran IAM yang dapat diasumsikan.
8. Untuk muatan acara Lambda, pilih Token untuk otorisasi atau Permintaan **`TOKEN`** otorisasi. `REQUEST`
9. Bergantung pada pilihan langkah sebelumnya, lakukan salah satu hal berikut:

   a. Untuk opsi Token, lakukan hal berikut:

   - Untuk sumber Token, masukkan nama header yang berisi token otorisasi. Klien API harus menyertakan header dari nama ini untuk mengirim token otorisasi ke otorisasi Lambda.
   - Secara opsional, untuk validasi Token, masukkan pernyataan RegEx . API Gateway melakukan validasi awal token input terhadap ekspresi ini dan memanggil authorizer setelah validasi berhasil. Ini membantu mengurangi panggilan ke API Anda.
   - Untuk men-cache kebijakan otorisasi yang dihasilkan oleh otorisasi, biarkan caching Otorisasi tetap aktif. Saat caching kebijakan diaktifkan, Anda dapat memilih untuk mengubah nilai TTL. Menyetel TTL ke nol menonaktifkan caching kebijakan. Saat caching kebijakan diaktifkan, nama header yang ditentukan dalam sumber Token menjadi kunci cache. Jika beberapa nilai diteruskan ke header ini dalam permintaan, semua nilai akan menjadi kunci cache, dengan urutan dipertahankan.

     > **Note**
     >
     > Nilai TTL default adalah 300 detik. Nilai maksimum adalah 3600 detik, batas ini tidak dapat ditingkatkan.

   b. Untuk opsi Permintaan, lakukan hal berikut:

   - Untuk tipe sumber Identity, pilih tipe parameter. Jenis parameter yang didukung adalah`Header`,`Query string`,`Stage variable`, dan`Context`. Untuk menambahkan lebih banyak sumber identitas, pilih Tambah parameter.
   - Untuk men-cache kebijakan otorisasi yang dihasilkan oleh otorisasi, biarkan caching Otorisasi tetap aktif. Saat caching kebijakan diaktifkan, Anda dapat memilih untuk mengubah nilai TTL. Menyetel TTL ke nol menonaktifkan caching kebijakan.

API Gateway menggunakan sumber identitas yang ditentukan sebagai kunci caching otorisasi permintaan. Saat caching diaktifkan, API Gateway memanggil fungsi Lambda otorisasi hanya setelah berhasil memverifikasi bahwa semua sumber identitas yang ditentukan ada saat runtime. Jika sumber identifikasi tertentu hilang, null, atau kosong, API Gateway mengembalikan `401 Unauthorized` respons tanpa memanggil fungsi Lambda otorisasi.

Ketika beberapa sumber identitas didefinisikan, mereka semua digunakan untuk mendapatkan kunci cache otorisasi. Mengubah salah satu bagian kunci cache menyebabkan otorisasi membuang dokumen kebijakan yang di-cache dan menghasilkan yang baru. Jika header dengan beberapa nilai diteruskan dalam permintaan, maka semua nilai akan menjadi bagian dari kunci cache, dengan urutan dipertahankan.

- Ketika caching dimatikan, tidak perlu menentukan sumber identitas.

> **Note**
>
> Untuk mengaktifkan caching, otorisasi Anda harus mengembalikan kebijakan yang berlaku untuk semua metode di seluruh API. Untuk menerapkan kebijakan khusus metode, Anda dapat menonaktifkan caching Otorisasi.

10. Pilih Buat Authorizer.
11. Tempel string `aws lambda add-permission` perintah yang Anda salin pada langkah sebelumnya ke AWS CLI jendela yang dikonfigurasi untuk akun kedua Anda. Ganti `AUTHORIZER_ID` dengan ID otorisasi Anda. Ini akan memberikan akses akun pertama Anda ke fungsi otorisasi Lambda akun kedua Anda.

Kontrol akses ke REST API menggunakan kumpulan pengguna Amazon Cognito sebagai otorisasi

Sebagai alternatif untuk menggunakan peran dan kebijakan IAM atau otorisasi Lambda (sebelumnya dikenal sebagai otorisasi khusus), Anda dapat menggunakan kumpulan pengguna Amazon Cognito untuk mengontrol siapa yang dapat mengakses API Anda di Amazon API Gateway.

Untuk menggunakan kumpulan pengguna Amazon Cognito dengan API Anda, Anda harus terlebih dahulu membuat otorisasi `COGNITO_USER_POOLS` jenis tersebut dan kemudian mengonfigurasi metode API untuk menggunakan otorisasi tersebut. Setelah API diterapkan, klien harus terlebih

dahulu memasukkan pengguna ke kumpulan pengguna, mendapatkan identitas atau token akses untuk pengguna, dan kemudian memanggil metode API dengan salah satu token, yang biasanya diatur ke `Authorization` header permintaan. Panggilan API hanya berhasil jika token yang diperlukan disediakan dan token yang disediakan valid, jika tidak, klien tidak berwenang untuk melakukan panggilan karena klien tidak memiliki kredensi yang dapat diotorisasi.

Token identitas digunakan untuk mengotorisasi panggilan API berdasarkan klaim identitas pengguna yang masuk. Token akses digunakan untuk mengotorisasi panggilan API berdasarkan cakupan kustom sumber daya yang dilindungi akses tertentu. Untuk informasi selengkapnya, lihat Menggunakan Token dengan Kumpulan Pengguna dan Server Sumber Daya dan Cakupan Khusus.

Untuk membuat dan mengonfigurasi kumpulan pengguna Amazon Cognito untuk API, Anda melakukan tugas berikut:

- Gunakan konsol Amazon Cognito, CLI/SDK, atau API untuk membuat kumpulan pengguna—atau gunakan yang dimiliki oleh akun lain. AWS
- Gunakan konsol API Gateway, CLI/SDK, atau API untuk membuat otorisasi API Gateway dengan kumpulan pengguna yang dipilih.
- Gunakan konsol API Gateway, CLI/SDK, atau API untuk mengaktifkan otorisasi pada metode API yang dipilih.

Untuk memanggil metode API apa pun dengan kumpulan pengguna diaktifkan, klien API Anda melakukan tugas berikut:

- Gunakan Amazon Cognito CLI/SDK atau API untuk menandatangani pengguna ke kumpulan pengguna yang dipilih, dan mendapatkan token identitas atau token akses. Untuk mempelajari lebih lanjut tentang menggunakan SDK, lihat Contoh kode untuk Amazon Cognito AWS menggunakan SDK.
- Gunakan kerangka kerja khusus klien untuk memanggil API Gateway API yang diterapkan dan menyediakan token yang sesuai di header. `Authorization`

Sebagai pengembang API, Anda harus memberikan ID kumpulan pengguna, ID klien, dan mungkin rahasia klien terkait yang didefinisikan sebagai bagian dari kumpulan pengguna.

> **Note**
>
> Untuk mengizinkan pengguna masuk menggunakan kredensi Amazon Cognito dan juga mendapatkan kredensi sementara untuk digunakan dengan izin peran IAM, gunakan Identitas Federasi Amazon Cognito. Untuk setiap metode HTTP titik akhir sumber daya API, tetapkan jenis otorisasi, kategori`Method Execution`, ke. `AWS_IAM`

Di bagian ini, kami menjelaskan cara membuat kumpulan pengguna, cara mengintegrasikan API Gateway API dengan kumpulan pengguna, dan cara menjalankan API yang terintegrasi dengan kumpulan pengguna.

Topik

- Dapatkan izin untuk membuat otorisasi kumpulan pengguna Amazon Cognito untuk REST API
- Buat kumpulan pengguna Amazon Cognito untuk REST API
- Integrasikan REST API dengan kumpulan pengguna Amazon Cognito
- Panggil REST API yang terintegrasi dengan kumpulan pengguna Amazon Cognito
- Konfigurasikan otorisasi Amazon Cognito lintas akun untuk REST API menggunakan konsol API Gateway
- Buat otorisasi Amazon Cognito untuk REST API menggunakan AWS CloudFormation

Dapatkan izin untuk membuat otorisasi kumpulan pengguna Amazon Cognito untuk REST API

Untuk membuat otorisasi dengan kumpulan pengguna Amazon Cognito, Anda harus `Allow` memiliki izin untuk membuat atau memperbarui otorisasi dengan kumpulan pengguna Amazon Cognito yang dipilih. Dokumen kebijakan IAM berikut menunjukkan contoh izin tersebut:

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

Pastikan kebijakan tersebut dilampirkan ke grup IAM tempat Anda berada atau peran IAM yang Anda tetapkan.

Dalam dokumen kebijakan sebelumnya, `apigateway:POST` tindakannya adalah untuk membuat otorisasi baru, dan `apigateway:PATCH` tindakannya adalah untuk memperbarui otorisasi yang ada. Anda dapat membatasi kebijakan ke wilayah tertentu atau API tertentu dengan mengganti dua karakter wildcard (*) pertama dari `Resource` nilai tersebut.

`Condition`Klausul yang digunakan di sini adalah untuk membatasi `Allowed` izin ke kumpulan pengguna yang ditentukan. Ketika `Condition` klausa ada, akses ke kumpulan pengguna mana pun yang tidak cocok dengan kondisi akan ditolak. Jika izin tidak memiliki `Condition` klausa, akses ke kumpulan pengguna apa pun diizinkan.

Anda memiliki opsi berikut untuk mengatur `Condition` klausa:

- Anda dapat mengatur ekspresi `ArnLike` atau `ArnEquals` kondisional untuk mengizinkan pembuatan atau pembaruan `COGNITO_USER_POOLS` otorisasi hanya dengan kumpulan pengguna yang ditentukan.
- Anda dapat menyetel ekspresi `ArnNotLike` atau `ArnNotEquals` kondisional untuk mengizinkan pembuatan atau pembaruan `COGNITO_USER_POOLS` otorisasi dengan kumpulan pengguna apa pun yang tidak ditentukan dalam ekspresi.
- Anda dapat menghilangkan `Condition` klausul untuk mengizinkan pembuatan atau pembaruan `COGNITO_USER_POOLS` otorisasi dengan kumpulan pengguna apa pun, AWS akun apa pun, dan di wilayah mana pun.

Untuk informasi selengkapnya tentang ekspresi bersyarat Amazon Resource Name (ARN), lihat Operator Kondisi Nama Sumber Daya Amazon. Seperti yang ditunjukkan dalam contoh, `apigateway:CognitoUserPoolProviderArn` adalah daftar ARN dari kumpulan `COGNITO_USER_POOLS` pengguna yang dapat atau tidak dapat digunakan dengan jenis otorisasi API Gateway. `COGNITO_USER_POOLS`

Buat kumpulan pengguna Amazon Cognito untuk REST API

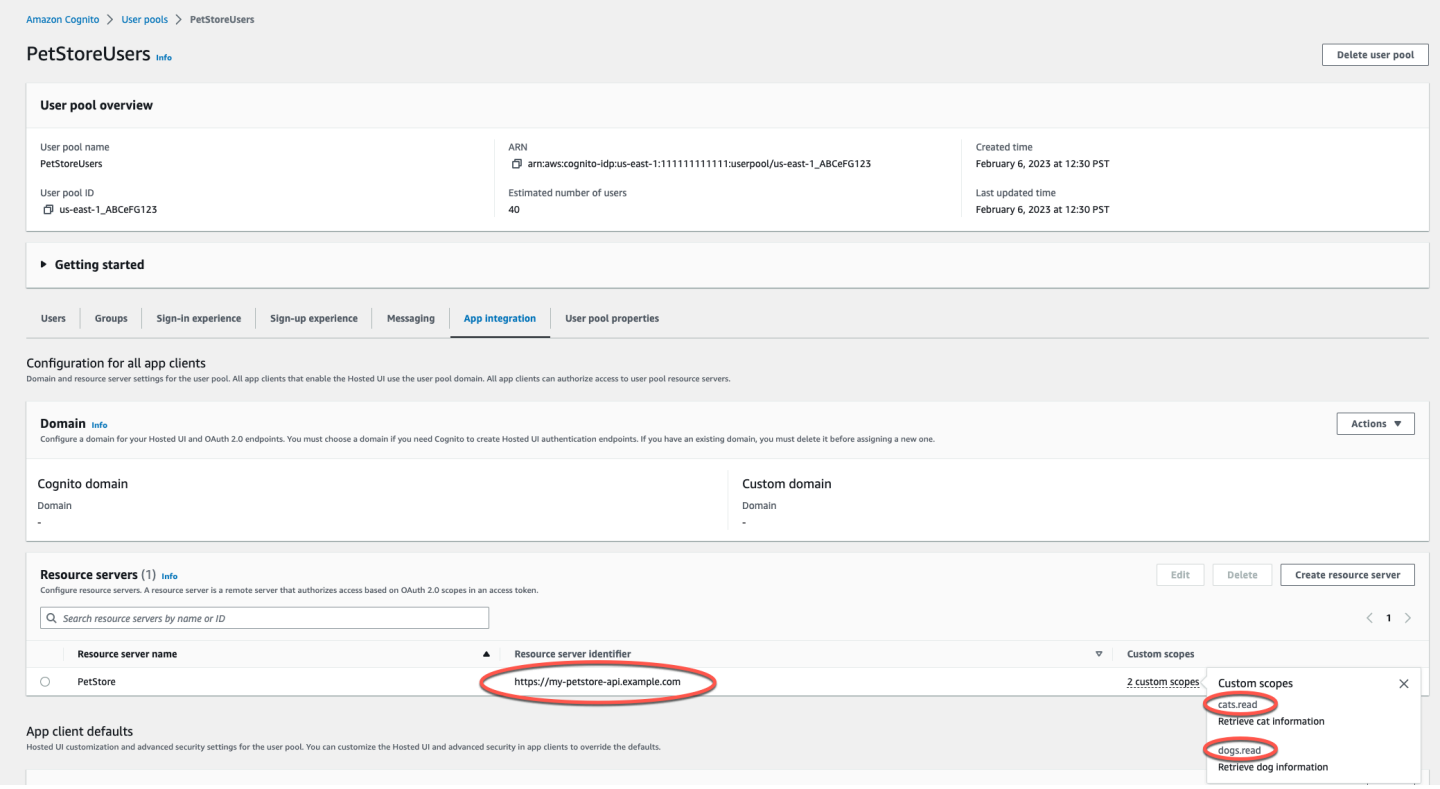
Sebelum mengintegrasikan API Anda dengan kumpulan pengguna, Anda harus membuat kumpulan pengguna di Amazon Cognito. Konfigurasi kumpulan pengguna Anda harus mengikuti semua kuota sumber daya untuk Amazon Cognito. Semua variabel Amazon Cognito yang ditentukan pengguna seperti grup, pengguna, dan peran hanya boleh menggunakan karakter alfanumerik. Untuk petunjuk tentang cara membuat kumpulan pengguna, lihat Tutorial: Membuat kumpulan pengguna di Panduan Pengembang Amazon Cognito.

Perhatikan ID kumpulan pengguna, ID klien, dan rahasia klien apa pun. Klien harus menyediakannya ke Amazon Cognito agar pengguna dapat mendaftar dengan kumpulan pengguna, masuk ke kumpulan pengguna, dan untuk mendapatkan identitas atau token akses untuk disertakan dalam permintaan memanggil metode API yang dikonfigurasi dengan kumpulan pengguna. Selain itu, Anda harus menentukan nama kumpulan pengguna saat mengonfigurasi kumpulan pengguna sebagai otorisasi di API Gateway, seperti yang dijelaskan selanjutnya.

Jika Anda menggunakan token akses untuk mengotorisasi panggilan metode API, pastikan untuk mengonfigurasi integrasi aplikasi dengan kumpulan pengguna untuk menyiapkan cakupan kustom yang Anda inginkan di server sumber daya tertentu. Untuk informasi selengkapnya tentang penggunaan token dengan kumpulan pengguna Amazon Cognito, lihat Menggunakan Token

dengan Kumpulan Pengguna. Untuk informasi selengkapnya tentang server sumber daya, lihat Mendefinisikan Server Sumber Daya untuk Kumpulan Pengguna Anda.

Perhatikan pengidentifikasi server sumber daya yang dikonfigurasi dan nama cakupan kustom. Anda membutuhkannya untuk membuat nama lengkap cakupan akses untuk OAuth Scopes, yang digunakan oleh otorisasi. `COGNITO_USER_POOLS`



Integrasikan REST API dengan kumpulan pengguna Amazon Cognito

Setelah membuat kumpulan pengguna Amazon Cognito, di API Gateway, Anda harus membuat `COGNITO_USER_POOLS` otorisasi yang menggunakan kumpulan pengguna. Prosedur berikut menunjukkan cara melakukannya menggunakan konsol API Gateway.

> **Note**
>
> Anda dapat menggunakan `CreateAuthorizer`tindakan untuk membuat `COGNITO_USER_POOLS` otorisasi yang menggunakan beberapa kumpulan pengguna. Anda dapat menggunakan hingga 1.000 kumpulan pengguna untuk satu `COGNITO_USER_POOLS` otorisasi. Batas ini tidak dapat dinaikkan.

> **Important**
>
> Setelah melakukan salah satu prosedur di bawah ini, Anda harus menerapkan atau menerapkan ulang API Anda untuk menyebarkan perubahan. Untuk informasi selengkapnya tentang penerapan API Anda, lihatMenerapkan REST API di Amazon API Gateway.

Untuk membuat **`COGNITO_USER_POOLS`** otorisasi menggunakan konsol API Gateway

1. Buat API baru, atau pilih API yang ada di API Gateway.
2. Di panel navigasi utama, pilih Authorizers.
3. Pilih Buat Authorizer.
4. Untuk mengonfigurasi otorisasi baru untuk menggunakan kumpulan pengguna, lakukan hal berikut:
   a. Untuk nama Authorizer, masukkan nama.
   b. Untuk jenis Authorizer, pilih Cognito.
   c. Untuk kumpulan pengguna Cognito, pilih Wilayah AWS tempat Anda membuat Amazon Cognito dan pilih kumpulan pengguna yang tersedia.
   d. Untuk sumber Token, masukkan **`Authorization`** sebagai nama header untuk meneruskan identitas atau token akses yang dikembalikan oleh Amazon Cognito saat pengguna berhasil masuk.
   e. (Opsional) Masukkan ekspresi reguler di bidang validasi Token untuk memvalidasi bidang `aud` (audiens) token identitas sebelum permintaan diotorisasi dengan Amazon Cognito. Perhatikan bahwa saat menggunakan token akses validasi ini menolak permintaan karena token akses tidak berisi bidang. `aud`
   f. Pilih Buat Authorizer.
5. Setelah membuat `COGNITO_USER_POOLS` otorisasi, Anda dapat menguji panggilannya secara opsional dengan menyediakan token identitas yang disediakan dari kumpulan pengguna. Anda dapat memperoleh token identitas ini dengan memanggil Amazon Cognito Identity SDK untuk melakukan login pengguna. Anda juga dapat menggunakan `InitiateAuth`aksinya. Jika Anda tidak mengonfigurasi cakupan Otorisasi apa pun, API Gateway memperlakukan token yang disediakan sebagai token identitas.

Prosedur sebelumnya membuat `COGNITO_USER_POOLS` otorisasi yang menggunakan kumpulan pengguna Amazon Cognito yang baru dibuat. Bergantung pada cara Anda mengaktifkan otorisasi pada metode API, Anda dapat menggunakan token identitas atau token akses yang disediakan dari kumpulan pengguna terintegrasi.

Untuk mengkonfigurasi **`COGNITO_USER_POOLS`** otorisasi pada metode

1. Pilih Sumber Daya. Pilih metode baru atau pilih metode yang ada. Jika perlu, buat sumber daya.
2. Pada tab Permintaan metode, di bawah Pengaturan permintaan metode, pilih Edit.
3. Untuk Authorizer, dari menu tarik-turun, pilih otorisasi kumpulan pengguna Amazon Cognito yang baru saja Anda buat.
4. Untuk menggunakan token identitas, lakukan hal berikut:

   a. Jaga Cakupan Otorisasi kosong.

   b. Jika diperlukan, dalam permintaan Integrasi, tambahkan `$context.authorizer.claims.`*`property-name`* ekspresi `$context.authorizer.claims['`*`property-name`*`']` atau dalam templat pemetaan tubuh untuk meneruskan properti klaim identitas yang ditentukan dari kumpulan pengguna ke backend. Untuk nama properti sederhana, seperti `sub` atau`custom-sub`, dua notasi identik. Untuk nama properti kompleks, seperti`custom:role`, Anda tidak dapat menggunakan notasi titik. Misalnya, ekspresi pemetaan berikut meneruskan bidang standar klaim `sub` dan `email` ke backend:

```
{
 "context" : {
  "sub" : "$context.authorizer.claims.sub",
  "email" : "$context.authorizer.claims.email"
 }
}
```

   Jika Anda mendeklarasikan bidang klaim kustom saat mengonfigurasi kumpulan pengguna, Anda dapat mengikuti pola yang sama untuk mengakses bidang kustom. Contoh berikut mendapatkan `role` bidang khusus klaim:

```
{
 "context" : {
  "role" : "$context.authorizer.claims.role"
    }
```

```
}
```

Jika bidang klaim kustom dideklarasikan sebagai`custom:role`, gunakan contoh berikut untuk mendapatkan properti klaim:

```
{
 "context" : {
  "role" : "$context.authorizer.claims['custom:role']"
    }
}
```

5. Untuk menggunakan token akses, lakukan hal berikut:

   a. Untuk Cakupan Otorisasi, masukkan satu atau beberapa nama lengkap cakupan yang telah dikonfigurasi saat kumpulan pengguna Amazon Cognito dibuat. Misalnya, mengikuti contoh yang diberikanBuat kumpulan pengguna Amazon Cognito untuk REST API, salah satu cakupannya adalah`https://my-petstore-api.example.com/cats.read`.

   Saat runtime, pemanggilan metode berhasil jika cakupan apa pun yang ditentukan pada metode dalam langkah ini cocok dengan cakupan yang diklaim dalam token yang masuk. Jika tidak, panggilan gagal dengan `401 Unauthorized` respons.

   b. Pilih Save (Simpan).

6. Ulangi langkah-langkah ini untuk metode lain yang Anda pilih.

Dengan `COGNITO_USER_POOLS` otorisasi, jika opsi OAuth Scopes tidak ditentukan, API Gateway memperlakukan token yang disediakan sebagai token identitas dan memverifikasi identitas yang diklaim terhadap yang dari kumpulan pengguna. Jika tidak, API Gateway memperlakukan token yang disediakan sebagai token akses dan memverifikasi cakupan akses yang diklaim dalam token terhadap cakupan otorisasi yang dideklarasikan pada metode.

Alih-alih menggunakan konsol API Gateway, Anda juga dapat mengaktifkan kumpulan pengguna Amazon Cognito pada metode dengan menentukan file definisi OpenAPI dan mengimpor definisi API ke API Gateway.

Untuk mengimpor otorisasi COGNITO_USER_POOLS dengan file definisi OpenAPI

1. Buat (atau ekspor) file definisi OpenAPI untuk API Anda.

2. Tentukan definisi JSON `COGNITO_USER_POOLS` authorizer (`MyUserPool`) sebagai bagian dari `securitySchemes` bagian di OpenAPI 3.0 atau `securityDefinitions` bagian di Open API 2.0 sebagai berikut:

   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. Untuk menggunakan token identitas untuk otorisasi metode, tambahkan `{ "MyUserPool": [] }` ke `security` definisi metode, seperti yang ditunjukkan dalam metode GET berikut pada sumber daya root.

   ```
   "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. Untuk menggunakan token akses untuk otorisasi metode, ubah definisi keamanan di atas menjadi`{ "MyUserPool": [resource-server/scope, ...] }`:

```
 "paths": {
   "/": {
     "get": {
       "consumes": [
         "application/json"
       ],
       "produces": [
         "text/html"
       ],
       "responses": {
         "200": {
           "description": "200 response",
           "headers": {
             "Content-Type": {
               "type": "string"
             }
           }
         }
       },
       "security": [
         {
           "MyUserPool": ["https://my-petstore-api.example.com/cats.read",
"http://my.resource.com/file.read"]
         }
       ],
       "x-amazon-apigateway-integration": {
         "type": "mock",
         "responses": {
           "default": {
             "statusCode": "200",
             "responseParameters": {
               "method.response.header.Content-Type": "'text/html'"
             },
           }
         },
         "requestTemplates": {
           "application/json": "{\"statusCode\": 200}"
         },
         "passthroughBehavior": "when_no_match"
       }
     },
     ...
```

```
    }
```

5. Jika diperlukan, Anda dapat mengatur pengaturan konfigurasi API lainnya dengan menggunakan definisi atau ekstensi OpenAPI yang sesuai. Untuk informasi selengkapnya, lihat Bekerja dengan ekstensi API Gateway ke OpenAPI.

Panggil REST API yang terintegrasi dengan kumpulan pengguna Amazon Cognito

Untuk memanggil metode dengan otorisasi kumpulan pengguna yang dikonfigurasi, klien harus melakukan hal berikut:

- Aktifkan pengguna untuk mendaftar dengan kumpulan pengguna.
- Aktifkan pengguna untuk masuk ke kumpulan pengguna.
- Dapatkan identitas atau token akses pengguna yang masuk dari kumpulan pengguna.
- Sertakan token di `Authorization` header (atau header lain yang Anda tentukan saat Anda membuat otorisasi).

Anda dapat menggunakan AWS Amplifyuntuk melakukan tugas-tugas ini. Lihat Mengintegrasikan Amazon Cognito Dengan Web dan Aplikasi Seluler untuk informasi selengkapnya.

- Untuk Android, lihat Memulai dengan Amplify untuk Android.
- Untuk menggunakan iOS, lihat Memulai Amplify untuk iOS.
- Untuk menggunakannya JavaScript, lihat Memulai dengan Amplify for Javascript.

Konfigurasikan otorisasi Amazon Cognito lintas akun untuk REST API menggunakan konsol API Gateway

Anda sekarang juga dapat menggunakan kumpulan pengguna Amazon Cognito dari AWS akun lain sebagai otorisasi API Anda. Setiap akun dapat berada di wilayah mana pun di mana Amazon API Gateway tersedia. Kumpulan pengguna Amazon Cognito dapat menggunakan strategi otentikasi token pembawa seperti OAuth atau SALL. Hal ini memudahkan pengelolaan dan berbagi otorisasi kumpulan pengguna Amazon Cognito pusat secara terpusat di beberapa API Gateway API.

Di bagian ini, kami menunjukkan cara mengonfigurasi kumpulan pengguna Amazon Cognito lintas akun menggunakan konsol Amazon API Gateway.

Petunjuk ini mengasumsikan bahwa Anda sudah memiliki API Gateway API di satu AWS akun dan kumpulan pengguna Amazon Cognito di akun lain.

Konfigurasikan otorisasi Amazon Cognito lintas akun menggunakan konsol API Gateway

Masuk ke konsol Amazon API Gateway di akun yang memiliki API Anda di dalamnya, lalu lakukan hal berikut:

1. Buat API baru, atau pilih API yang ada di API Gateway.
2. Di panel navigasi utama, pilih Authorizers.
3. Pilih Buat Authorizer.
4. Untuk mengonfigurasi otorisasi baru untuk menggunakan kumpulan pengguna, lakukan hal berikut:
   a. Untuk nama Authorizer, masukkan nama.
   b. Untuk jenis Authorizer, pilih Cognito.
   c. Untuk kumpulan pengguna Cognito, masukkan ARN lengkap untuk kumpulan pengguna yang Anda miliki di akun kedua Anda.

      > **Note**
      >
      > Di konsol Amazon Cognito, Anda dapat menemukan ARN untuk kumpulan pengguna Anda di bidang ARN Kolam pada panel Pengaturan Umum.

   d. Untuk sumber Token, masukkan **`Authorization`** sebagai nama header untuk meneruskan identitas atau token akses yang dikembalikan oleh Amazon Cognito saat pengguna berhasil masuk.
   e. (Opsional) Masukkan ekspresi reguler di bidang validasi Token untuk memvalidasi bidang `aud` (audiens) token identitas sebelum permintaan diotorisasi dengan Amazon Cognito. Perhatikan bahwa saat menggunakan token akses validasi ini menolak permintaan karena token akses tidak berisi bidang. `aud`
   f. Pilih Buat Authorizer.

Buat otorisasi Amazon Cognito untuk REST API menggunakan AWS CloudFormation

Anda dapat menggunakannya AWS CloudFormation untuk membuat kumpulan pengguna Amazon Cognito dan otorisasi Amazon Cognito. Contoh AWS CloudFormation template melakukan hal berikut:

- Buat kumpulan pengguna Amazon Cognito. Klien harus terlebih dahulu menandatangani pengguna ke kumpulan pengguna dan mendapatkan identitas atau token akses. Jika Anda menggunakan token akses untuk mengotorisasi panggilan metode API, pastikan untuk mengonfigurasi integrasi aplikasi dengan kumpulan pengguna untuk menyiapkan cakupan kustom yang Anda inginkan di server sumber daya tertentu.
- Membuat API Gateway API dengan `GET` metode.
- Membuat otorisasi Amazon Cognito yang menggunakan `Authorization` header sebagai sumber token.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  UserPool:
    Type: AWS::Cognito::UserPool
    Properties:
      AccountRecoverySetting:
        RecoveryMechanisms:
          - Name: verified_phone_number
            Priority: 1
          - Name: verified_email
            Priority: 2
      AdminCreateUserConfig:
        AllowAdminCreateUserOnly: true
      EmailVerificationMessage: The verification code to your new account is {####}
      EmailVerificationSubject: Verify your new account
      SmsVerificationMessage: The verification code to your new account is {####}
      VerificationMessageTemplate:
        DefaultEmailOption: CONFIRM_WITH_CODE
        EmailMessage: The verification code to your new account is {####}
        EmailSubject: Verify your new account
        SmsMessage: The verification code to your new account is {####}
    UpdateReplacePolicy: Retain
    DeletionPolicy: Retain
  CogAuthorizer:
    Type: AWS::ApiGateway::Authorizer
    Properties:
      Name: CognitoAuthorizer
      RestApiId:
        Ref: Api
      Type: COGNITO_USER_POOLS
      IdentitySource: method.request.header.Authorization
      ProviderARNs:
```

```
        - Fn::GetAtt:
            - UserPool
            - Arn
  Api:
    Type: AWS::ApiGateway::RestApi
    Properties:
      Name: MyCogAuthApi
  ApiDeployment:
    Type: AWS::ApiGateway::Deployment
    Properties:
      RestApiId:
        Ref: Api
    DependsOn:
      - CogAuthorizer
      - ApiGET
  ApiDeploymentStageprod:
    Type: AWS::ApiGateway::Stage
    Properties:
      RestApiId:
        Ref: Api
      DeploymentId:
        Ref: ApiDeployment
      StageName: prod
  ApiGET:
    Type: AWS::ApiGateway::Method
    Properties:
      HttpMethod: GET
      ResourceId:
        Fn::GetAtt:
          - Api
          - RootResourceId
      RestApiId:
        Ref: Api
      AuthorizationType: COGNITO_USER_POOLS
      AuthorizerId:
        Ref: CogAuthorizer
      Integration:
        IntegrationHttpMethod: GET
        Type: HTTP_PROXY
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets
Outputs:
  ApiEndpoint:
    Value:
      Fn::Join:
```

```
        - ""
        - - https://
          - Ref: Api
          - .execute-api.
          - Ref: AWS::Region
          - "."
          - Ref: AWS::URLSuffix
          - /
          - Ref: ApiDeploymentStageprod
          - /
```

Menyiapkan integrasi REST API

Setelah menyiapkan metode API, Anda harus mengintegrasikannya dengan titik akhir di backend. Endpoint backend juga disebut sebagai titik akhir integrasi dan dapat berupa fungsi Lambda, halaman web HTTP, atau tindakan layanan. AWS

Seperti halnya metode API, integrasi API memiliki permintaan integrasi dan respons integrasi. Permintaan integrasi merangkum permintaan HTTP yang diterima oleh backend. Ini mungkin atau mungkin tidak berbeda dari permintaan metode yang diajukan oleh klien. Respons integrasi adalah respons HTTP yang merangkum output yang dikembalikan oleh backend.

Menyiapkan permintaan integrasi melibatkan hal-hal berikut: mengonfigurasi cara meneruskan permintaan metode yang dikirimkan klien ke backend; mengonfigurasi cara mengubah data permintaan, jika perlu, ke data permintaan integrasi; dan menentukan fungsi Lambda mana yang akan dipanggil, menentukan server HTTP mana yang akan meneruskan permintaan masuk, atau menentukan tindakan layanan yang akan dipanggil. AWS

Menyiapkan respons integrasi (hanya berlaku untuk integrasi non-proxy) melibatkan hal berikut: mengonfigurasi cara meneruskan hasil yang dikembalikan ke respons metode dari kode status tertentu, mengonfigurasi cara mengubah parameter respons integrasi yang ditentukan ke parameter respons metode yang telah dikonfigurasi sebelumnya, dan mengonfigurasi cara memetakan badan respons integrasi ke badan respons metode sesuai dengan templat pemetaan tubuh yang ditentukan.

Secara terprogram, permintaan integrasi dienkapsulasi oleh `Integration`sumber daya dan respons integrasi oleh sumber daya API Gateway. `IntegrationResponse`

Untuk menyiapkan permintaan integrasi, Anda membuat `Integration`sumber daya dan menggunakannya untuk mengonfigurasi URL titik akhir integrasi. Anda kemudian mengatur izin IAM untuk mengakses backend, dan menentukan pemetaan untuk mengubah data permintaan yang

masuk sebelum meneruskannya ke backend. Untuk menyiapkan respons integrasi untuk integrasi non-proxy, Anda membuat `IntegrationResponse`sumber daya dan menggunakannya untuk menetapkan respons metode targetnya. Anda kemudian mengonfigurasi cara memetakan output backend ke respons metode.

Topik

- Menyiapkan permintaan integrasi di API Gateway
- Siapkan respons integrasi di API Gateway
- Siapkan integrasi Lambda di API Gateway
- Menyiapkan integrasi HTTP di API Gateway
- Siapkan integrasi pribadi API Gateway
- Siapkan integrasi tiruan di API Gateway

Menyiapkan permintaan integrasi di API Gateway

Untuk menyiapkan permintaan integrasi, Anda melakukan tugas wajib dan opsional berikut:

1. Pilih jenis integrasi yang menentukan bagaimana data permintaan metode diteruskan ke backend.
2. Untuk integrasi non-tiruan, tentukan metode HTTP dan URI dari titik akhir integrasi yang ditargetkan, kecuali untuk integrasi. `MOCK`
3. Untuk integrasi dengan fungsi Lambda dan tindakan layanan AWS lainnya, tetapkan peran IAM dengan izin yang diperlukan untuk API Gateway untuk memanggil backend atas nama Anda.
4. Untuk integrasi non-proxy, atur pemetaan parameter yang diperlukan untuk memetakan parameter permintaan metode yang telah ditentukan sebelumnya ke parameter permintaan integrasi yang sesuai.
5. Untuk integrasi non-proxy, setel pemetaan badan yang diperlukan untuk memetakan badan permintaan metode masuk dari jenis konten tertentu sesuai dengan templat pemetaan yang ditentukan.
6. Untuk integrasi non-proxy, tentukan kondisi di mana data permintaan metode masuk diteruskan ke backend apa adanya.
7. Secara opsional, tentukan cara menangani konversi tipe untuk muatan biner.
8. Secara opsional, deklarasikan nama namespace cache dan parameter kunci cache untuk mengaktifkan caching API.

Melakukan tugas ini melibatkan pembuatan sumber daya Integrasi API Gateway dan menetapkan nilai properti yang sesuai. Anda dapat melakukannya menggunakan konsol API Gateway, AWS CLI perintah, AWS SDK, atau API Gateway REST API.

Topik

- Tugas dasar permintaan integrasi API
- Pilih jenis integrasi API Gateway API
- Menyiapkan integrasi proxy dengan sumber daya proxy
- Menyiapkan permintaan integrasi API menggunakan konsol API Gateway

Tugas dasar permintaan integrasi API

Permintaan integrasi adalah permintaan HTTP yang API Gateway kirimkan ke backend, meneruskan data permintaan yang dikirimkan klien, dan mengubah data, jika perlu. Metode HTTP (atau kata kerja) dan URI dari permintaan integrasi ditentukan oleh backend (yaitu, titik akhir integrasi). Mereka dapat sama dengan atau berbeda dari metode HTTP permintaan metode dan URI, masing-masing.

Misalnya, ketika fungsi Lambda mengembalikan file yang diambil dari Amazon S3, Anda dapat mengekspos operasi ini secara intuitif sebagai permintaan `GET` metode ke klien meskipun permintaan integrasi yang sesuai mengharuskan permintaan `POST` digunakan untuk menjalankan fungsi Lambda. Untuk titik akhir HTTP, kemungkinan permintaan metode dan permintaan integrasi yang sesuai keduanya menggunakan kata kerja HTTP yang sama. Namun, ini tidak diperlukan. Anda dapat mengintegrasikan permintaan metode berikut:

```
GET /{var}?query=value
Host: api.domain.net
```

Dengan permintaan integrasi berikut:

```
POST /
Host: service.domain.com
Content-Type: application/json
Content-Length: ...

{
   path: "{var}'s value",
   type: "value"
}
```

Sebagai pengembang API, Anda dapat menggunakan kata kerja HTTP dan URI apa pun untuk permintaan metode yang sesuai dengan kebutuhan Anda. Tetapi Anda harus mengikuti persyaratan titik akhir integrasi. Ketika data permintaan metode berbeda dari data permintaan integrasi, Anda dapat merekonsiliasi perbedaan dengan menyediakan pemetaan dari data permintaan metode ke data permintaan integrasi.

Dalam contoh sebelumnya, pemetaan menerjemahkan nilai variabel jalur (`{var}`) dan parameter kueri (`query`) dari permintaan `GET` metode ke nilai properti payload permintaan integrasi dan. `path type` Data permintaan lain yang dapat dipetakan mencakup header dan badan permintaan. Ini dijelaskan dalamMengatur pemetaan data permintaan dan respons menggunakan konsol API Gateway.

Saat menyiapkan permintaan integrasi proxy HTTP atau HTTP, Anda menetapkan URL endpoint HTTP backend sebagai nilai URI permintaan integrasi. Misalnya, di PetStore API, permintaan metode untuk mendapatkan halaman hewan peliharaan memiliki URI permintaan integrasi berikut:

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

Saat menyiapkan integrasi proxy Lambda atau Lambda, Anda menetapkan Nama Sumber Daya Amazon (ARN) untuk menjalankan fungsi Lambda sebagai nilai URI permintaan integrasi. ARN ini memiliki format sebagai berikut:

```
arn:aws:apigateway:api-region:lambda:path//2015-03-31/functions/arn:aws:lambda:lambda-region:account-id:function:lambda-function-name/invocations
```

Bagian setelahnya`arn:aws:apigateway:api-region:lambda:path/`, yaitu`/2015-03-31/functions/arn:aws:lambda:lambda-region:account-id:function:lambda-function-name/invocations`, adalah jalur REST API URI dari tindakan Lambda Invoke. Jika Anda menggunakan konsol API Gateway untuk menyiapkan integrasi Lambda, API Gateway akan membuat ARN dan menetapkannya ke URI integrasi setelah meminta Anda untuk memilih dari suatu wilayah. *`lambda-function-name`*

Saat menyiapkan permintaan integrasi dengan tindakan AWS layanan lain, URI permintaan integrasi juga merupakan ARN, mirip dengan integrasi dengan tindakan Lambda. `Invoke` Misalnya, untuk integrasi dengan GetBucketaksi Amazon S3, URI permintaan integrasi adalah ARN dengan format berikut:

```
arn:aws:apigateway:api-region:s3:path/{bucket}
```

URI permintaan integrasi adalah konvensi jalur untuk menentukan tindakan, di *{bucket}* mana placeholder nama bucket. Atau, tindakan AWS layanan dapat direferensikan dengan namanya. Menggunakan nama tindakan, URI permintaan integrasi untuk `GetBucket` tindakan Amazon S3 menjadi sebagai berikut:

```
arn:aws:apigateway:api-region:s3:action/GetBucket
```

Dengan URI permintaan integrasi berbasis tindakan, nama bucket (*{bucket}*) harus ditentukan dalam isi permintaan integrasi (`{ Bucket: "{bucket}" }`), mengikuti format input tindakan`GetBucket`.

Untuk AWS integrasi, Anda juga harus mengonfigurasi kredensional agar API Gateway dapat memanggil tindakan terintegrasi. Anda dapat membuat peran IAM baru atau memilih peran IAM yang sudah ada untuk API Gateway untuk memanggil tindakan dan kemudian menentukan peran menggunakan ARN-nya. Berikut ini menunjukkan contoh ARN ini:

```
arn:aws:iam::account-id:role/iam-role-name
```

Peran IAM ini harus berisi kebijakan untuk memungkinkan tindakan dieksekusi. Itu juga harus memiliki API Gateway yang dideklarasikan (dalam hubungan kepercayaan peran) sebagai entitas tepercaya untuk mengambil peran. Izin tersebut dapat diberikan pada tindakan itu sendiri. Mereka dikenal sebagai izin berbasis sumber daya. Untuk integrasi Lambda, Anda dapat memanggil tindakan AddPermission Lambda untuk menyetel izin berbasis sumber daya dan kemudian menyetel ke null dalam permintaan integrasi API Gateway. `credentials`

Kami membahas pengaturan integrasi dasar. Pengaturan lanjutan melibatkan pemetaan data permintaan metode ke data permintaan integrasi. Setelah membahas pengaturan dasar untuk respons integrasi, kami membahas topik lanjutan diMengatur pemetaan data permintaan dan respons menggunakan konsol API Gateway, di mana kami juga membahas pengiriman muatan dan penanganan pengkodean konten.

Pilih jenis integrasi API Gateway API

Anda memilih jenis integrasi API sesuai dengan jenis titik akhir integrasi yang bekerja dengan Anda dan bagaimana Anda ingin data diteruskan ke dan dari titik akhir integrasi. Untuk fungsi Lambda, Anda dapat memiliki integrasi proxy Lambda, atau integrasi kustom Lambda. Untuk titik akhir HTTP,

Anda dapat memiliki integrasi proxy HTTP atau integrasi kustom HTTP. Untuk tindakan AWS layanan, Anda memiliki AWS integrasi tipe non-proxy saja. API Gateway juga mendukung integrasi tiruan, di mana API Gateway berfungsi sebagai titik akhir integrasi untuk menanggapi permintaan metode.

Integrasi kustom Lambda adalah kasus khusus AWS integrasi, di mana titik akhir integrasi sesuai dengan tindakan pemanggilan fungsi layanan Lambda.

Secara terprogram, Anda memilih jenis integrasi dengan menyetel `type`properti pada sumber daya. `Integration` Untuk integrasi proxy Lambda, nilainya adalah. `AWS_PROXY` Untuk integrasi kustom Lambda dan semua AWS integrasi lainnya, memang demikian. `AWS` Untuk integrasi proxy HTTP dan integrasi HTTP, nilainya adalah `HTTP_PROXY` dan`HTTP`, masing-masing. Untuk integrasi tiruan, `type` nilainya adalah`MOCK`.

Integrasi proxy Lambda mendukung pengaturan integrasi yang efisien dengan satu fungsi Lambda. Pengaturannya sederhana dan dapat berkembang dengan backend tanpa harus meruntuhkan pengaturan yang ada. Untuk alasan ini, sangat disarankan untuk integrasi dengan fungsi Lambda.

Sebaliknya, integrasi kustom Lambda memungkinkan penggunaan kembali template pemetaan yang dikonfigurasi untuk berbagai titik akhir integrasi yang memiliki persyaratan serupa dari format data input dan output. Pengaturan lebih terlibat dan direkomendasikan untuk skenario aplikasi yang lebih maju.

Demikian pula, integrasi proxy HTTP memiliki pengaturan integrasi yang efisien dan dapat berkembang dengan backend tanpa harus meruntuhkan pengaturan yang ada. Integrasi kustom HTTP lebih terlibat untuk disiapkan, tetapi memungkinkan penggunaan kembali template pemetaan yang dikonfigurasi untuk titik akhir integrasi lainnya.

Daftar berikut merangkum jenis integrasi yang didukung:

- `AWS`: Jenis integrasi ini memungkinkan API mengekspos tindakan AWS layanan. Dalam `AWS` integrasi, Anda harus mengonfigurasi permintaan integrasi dan respons integrasi dan menyiapkan pemetaan data yang diperlukan dari permintaan metode ke permintaan integrasi, dan dari respons integrasi ke respons metode.
- `AWS_PROXY`: Jenis integrasi ini memungkinkan metode API diintegrasikan dengan tindakan pemanggilan fungsi Lambda dengan pengaturan integrasi yang fleksibel, serbaguna, dan efisien. Integrasi ini bergantung pada interaksi langsung antara klien dan fungsi Lambda terintegrasi.

  Dengan jenis integrasi ini, juga dikenal sebagai integrasi proxy Lambda, Anda tidak mengatur permintaan integrasi atau respons integrasi. API Gateway meneruskan permintaan masuk dari

klien sebagai input ke fungsi Lambda backend. Fungsi Lambda terintegrasi mengambil input format ini dan mem-parsing input dari semua sumber yang tersedia, termasuk header permintaan, variabel jalur URL, parameter string kueri, dan isi yang berlaku. Fungsi mengembalikan hasil mengikuti format output ini.

Ini adalah jenis integrasi yang disukai untuk memanggil fungsi Lambda melalui API Gateway dan tidak berlaku untuk tindakan AWS layanan lainnya, termasuk tindakan Lambda selain tindakan pemanggilan fungsi.

- `HTTP`: Jenis integrasi ini memungkinkan API mengekspos titik akhir HTTP di backend. Dengan `HTTP` integrasi, juga dikenal sebagai integrasi kustom HTTP, Anda harus mengonfigurasi permintaan integrasi dan respons integrasi. Anda harus menyiapkan pemetaan data yang diperlukan dari permintaan metode ke permintaan integrasi, dan dari respons integrasi ke respons metode.
- `HTTP_PROXY`: Integrasi proxy HTTP memungkinkan klien untuk mengakses titik akhir HTTP backend dengan pengaturan integrasi yang disederhanakan pada metode API tunggal. Anda tidak mengatur permintaan integrasi atau respons integrasi. API Gateway meneruskan permintaan masuk dari klien ke titik akhir HTTP dan meneruskan respons keluar dari titik akhir HTTP ke klien.
- `MOCK`: Jenis integrasi ini memungkinkan API Gateway mengembalikan respons tanpa mengirim permintaan lebih lanjut ke backend. Ini berguna untuk pengujian API karena dapat digunakan untuk menguji pengaturan integrasi tanpa menimbulkan biaya untuk menggunakan backend dan untuk mengaktifkan pengembangan kolaboratif API.

  Dalam pengembangan kolaboratif, tim dapat mengisolasi upaya pengembangan mereka dengan menyiapkan simulasi komponen API yang dimiliki oleh tim lain dengan menggunakan integrasi. `MOCK` Ini juga digunakan untuk mengembalikan header terkait CORS untuk memastikan bahwa metode API mengizinkan akses CORS. Faktanya, konsol API Gateway mengintegrasikan `OPTIONS` metode untuk mendukung CORS dengan integrasi tiruan. Respons gateway adalah contoh lain dari integrasi tiruan.

Menyiapkan integrasi proxy dengan sumber daya proxy

Untuk menyiapkan integrasi proxy di API Gateway API dengansumber daya Proksi, Anda harus melakukan tugas berikut:

- Buat sumber daya proxy dengan variabel jalur serakah`{proxy+}`.
- MengaturANYmetode di sumber daya proxy.

- Integrasikan sumber daya dan metode dengan backend menggunakan tipe integrasi HTTP atau Lambda.

> **Note**
>
> Variabel jalur serakah,`ANY`metode, dan jenis integrasi proxy adalah fitur independen, meskipun mereka umumnya digunakan bersama-sama. Anda dapat mengonfigurasi metode HTTP tertentu pada sumber daya serakah atau menerapkan jenis integrasi non-proxy ke sumber daya proxy.

API Gateway memberlakukan pembatasan dan batasan tertentu saat menangani metode dengan integrasi proxy Lambda atau integrasi proxy HTTP. Untuk detailnya, lihat the section called “Catatan penting”.

> **Note**
>
> Saat menggunakan integrasi proxy dengan passthrough, API Gateway mengembalikan default`Content-Type:application/json`header jika jenis konten payload tidak ditentukan.

Sumber daya proxy paling kuat ketika terintegrasi dengan backend menggunakan integrasi proxy HTTP atau proxy Lambdaintegrasi.

Integrasi proxy HTTP dengan sumber daya proxy

Integrasi proxy HTTP, yang ditunjuk oleh`HTTP_PROXY`di API Gateway REST API, adalah untuk mengintegrasikan permintaan metode dengan endpoint HTTP backend. Dengan jenis integrasi ini, API Gateway hanya melewati seluruh permintaan dan respons antara frontend dan backend, tunduk pada tertentupembatasan dan batasan.

> **Note**
>
> Integrasi proxy HTTP mendukung header multi-nilai dan string kueri.

Saat menerapkan integrasi proxy HTTP ke sumber daya proxy, Anda dapat mengatur API Anda untuk mengekspos sebagian atau keseluruhan hirarki titik akhir backend HTTP dengan pengaturan integrasi tunggal. Misalnya, backend situs web diatur ke dalam beberapa cabang node pohon dari simpul akar (`/site`) sebagai:`/site/`a_0`/`a_1`/.../`a_N,`/site/`b_0`/`b_1`/.../`b_M, dll. Jika Anda mengintegrasikan`ANY`metode pada sumber daya proxy`/api/{proxy+}`dengan endpoint backend dengan jalur URL`/site/{proxy}`, permintaan integrasi tunggal dapat mendukung operasi HTTP (GET, POST, dll) pada salah satu`[`a_0`, `a_1`, ..., `a_N`, `b_0`, `b_1`, ...`b_M`, ...]`. Jika Anda menerapkan integrasi proxy ke metode HTTP tertentu, misalnya,`GET`, sebagai gantinya, permintaan integrasi yang dihasilkan bekerja dengan yang ditentukan (yaitu,`GET`) operasi pada salah satu node backend tersebut.

Integrasi proxy Lambda dengan sumber daya proxy

Integrasi proxy Lambda, yang ditunjuk oleh`AWS_PROXY`di API Gateway REST API, adalah untuk mengintegrasikan permintaan metode dengan fungsi Lambda di backend. Dengan tipe integrasi ini, API Gateway menerapkan templat pemetaan default untuk mengirim seluruh permintaan ke fungsi Lambda dan mengubah output dari fungsi Lambda menjadi respons HTTP.

Demikian pula, Anda dapat menerapkan integrasi proxy Lambda ke sumber daya proxy`/api/{proxy+}`untuk mengatur integrasi tunggal agar fungsi Lambda backend bereaksi secara individual terhadap perubahan dalam salah satu sumber daya API di bawah`/api`.

Menyiapkan permintaan integrasi API menggunakan konsol API Gateway

Penyiapan metode API mendefinisikan metode dan menjelaskan perilakunya. Untuk mengatur metode, Anda harus menentukan sumber daya, termasuk root (“/”), di mana metode diekspos, metode HTTP (,, dll.) `GETPOST`, Dan bagaimana itu akan diintegrasikan dengan backend yang ditargetkan. Permintaan dan respons metode menentukan kontrak dengan aplikasi panggilan, menetapkan parameter mana yang dapat diterima API dan seperti apa responsnya.

Prosedur berikut menjelaskan cara menggunakan konsol API Gateway untuk menentukan pengaturan metode.

1. Di panel Resources, pilih Create method.
2. Untuk jenis Metode, pilih metode HTTP.
3. Untuk jenis Integrasi, pilih dari berikut ini:

- Pilih fungsi Lambda jika API Anda akan terintegrasi dengan fungsi Lambda. Pada API level, ini adalah tipe `AWS` integrasi jika Anda membuat integrasi non-proxy, atau tipe `AWS_PROXY` integrasi jika Anda membuat integrasi proxy.
- Pilih HTTP jika API Anda akan terintegrasi dengan titik akhir HTTP. Pada level API, ini adalah tipe `HTTP` integrasi.
- Pilih AWSlayanan jika API Anda akan terintegrasi langsung dengan AWS layanan. Pada level API, ini adalah tipe `AWS` integrasi. Opsi fungsi Lambda adalah kasus khusus `AWS` integrasi untuk menjalankan fungsi Lambda.

  Untuk menyiapkan API Gateway API untuk melakukan salah satu hal berikut:

  - Buat fungsi Lambda baru.
  - Tetapkan izin sumber daya pada fungsi Lambda.
  - Lakukan tindakan layanan Lambda lainnya.

  Anda harus memilih AWSlayanan.
- Pilih Mock jika Anda ingin API Gateway bertindak sebagai backend Anda untuk mengembalikan respons statis. Pada level API, ini adalah tipe `MOCK` integrasi. Biasanya, Anda dapat menggunakan `MOCK` integrasi saat API Anda belum final, tetapi Anda ingin menghasilkan respons API untuk membuka blokir tim dependen untuk pengujian. Untuk `OPTION` metode ini, API Gateway menetapkan `MOCK` integrasi sebagai default untuk mengembalikan header yang mengaktifkan CORS untuk sumber daya API yang diterapkan. Jika Anda memilih opsi ini, lewati instruksi lainnya dalam topik ini dan lihatSiapkan integrasi tiruan di API Gateway.

4. Jika Anda memilih fungsi Lambda, lakukan hal berikut:

   a. Untuk menggunakan integrasi proxy Lambda, aktifkan integrasi proxy Lambda. Untuk mempelajari lebih lanjut tentang integrasi proxy Lambda, lihat. the section called “ Memahami integrasi proxy Lambda ”

   b. Untuk fungsi Lambda, masukkan nama fungsi Lambda. Jika Anda menggunakan fungsi Lambda di Wilayah yang berbeda dari API Anda, pilih Wilayah dari menu tarik-turun dan masukkan nama fungsi Lambda. Jika Anda menggunakan fungsi Lambda lintas akun, masukkan fungsi ARN.

      Untuk daftar nama dan pengenal Wilayah, lihat AWS Lambdadi Referensi Umum Amazon Web Services.

c. Untuk menggunakan nilai batas waktu default 29 detik, tetap aktifkan batas waktu default. Untuk menetapkan batas waktu kustom, pilih Batas waktu default dan masukkan nilai batas waktu antara `50` dan milidetik. `29000`

d. Pilih metode Buat.

5. Jika Anda memilih HTTP, lakukan hal berikut:

a. Untuk menggunakan integrasi proxy HTTP, aktifkan integrasi proxy HTTP. Untuk mempelajari lebih lanjut tentang integrasi proxy HTTP, lihatthe section called “Menyiapkan integrasi proxy HTTP di API Gateway”.

b. Untuk metode HTTP, pilih jenis metode HTTP yang paling cocok dengan metode di backend HTTP.

c. Untuk URL Endpoint, masukkan URL backend HTTP yang ingin digunakan metode ini.

d. Untuk penanganan Konten, pilih perilaku penanganan konten.

e. Untuk menggunakan nilai batas waktu default 29 detik, tetap aktifkan batas waktu default. Untuk menetapkan batas waktu kustom, pilih Batas waktu default dan masukkan nilai batas waktu antara `50` dan milidetik. `29000`

f. Pilih metode Buat.

6. Jika Anda memilih Mock, lakukan hal berikut:

- Pilih metode Buat.

7. Jika Anda memilih AWSlayanan, lakukan hal berikut:

a. Untuk AWSWilayah, pilih AWS Wilayah yang ingin digunakan metode ini untuk memanggil tindakan.

b. Untuk AWSlayanan, pilih AWS layanan yang Anda inginkan untuk memanggil metode ini.

c. Untuk AWSsubdomain, masukkan subdomain yang digunakan oleh layanan. AWS Biasanya, Anda akan membiarkan ini kosong. Beberapa AWS layanan dapat mendukung subdomain sebagai bagian dari host. Konsultasikan dokumentasi layanan untuk ketersediaan dan, jika tersedia, detailnya.

d. Untuk metode HTTP, pilih jenis metode HTTP yang sesuai dengan tindakan. Untuk jenis metode HTTP, lihat dokumentasi referensi API untuk AWS layanan yang Anda pilih untuk AWSlayanan.

e. Untuk tipe Tindakan, pilih Gunakan nama tindakan untuk menggunakan tindakan API atau Use path override untuk menggunakan jalur sumber daya kustom. Untuk tindakan yang

tersedia dan jalur sumber daya khusus, lihat dokumentasi referensi API untuk AWS layanan yang Anda pilih untuk AWSlayanan.

f. Masukkan nama Action atau Path override.

g. Untuk peran Eksekusi, masukkan ARN dari peran IAM yang akan digunakan metode untuk memanggil tindakan.

Untuk membuat peran IAM, Anda dapat menyesuaikan instruksi dithe section called "Langkah 1: Buat peran eksekusi proxy AWS layanan". Tentukan kebijakan akses format berikut, dengan jumlah tindakan dan pernyataan sumber daya yang diinginkan:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "action-statement"
      ],
      "Resource": [
        "resource-statement"
      ]
    },
    ...
  ]
}
```

Untuk sintaks pernyataan tindakan dan sumber daya, lihat dokumentasi untuk AWS layanan yang Anda pilih untuk AWSlayanan.

Untuk hubungan kepercayaan peran IAM, tentukan hal berikut, yang memungkinkan API Gateway mengambil tindakan atas nama AWS akun Anda:

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

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

h. Untuk menggunakan nilai batas waktu default 29 detik, tetap aktifkan batas waktu default. Untuk menetapkan batas waktu kustom, pilih Batas waktu default dan masukkan nilai batas waktu antara `50` dan milidetik. `29000`

i. Pilih metode Buat.

Siapkan respons integrasi di API Gateway

Untuk integrasi non-proxy, Anda harus menyiapkan setidaknya satu respons integrasi, dan menjadikannya respons default, untuk meneruskan hasil yang dikembalikan dari backend ke klien. Anda dapat memilih untuk melewati hasil apa adanya atau mengubah data respons integrasi ke data respons metode jika keduanya memiliki format yang berbeda.

Untuk integrasi proxy, API Gateway secara otomatis meneruskan output backend ke klien sebagai respons HTTP. Anda tidak menetapkan respons integrasi atau respons metode.

Untuk menyiapkan respons integrasi, Anda melakukan tugas wajib dan opsional berikut ini:

1. Tentukan kode status HTTP dari respons metode yang data respons integrasi dipetakan. Ini wajib diisi.
2. Tentukan ekspresi reguler untuk memilih output backend yang akan diwakili oleh respons integrasi ini. Jika Anda membiarkan ini kosong, responsnya adalah respons default yang digunakan untuk menangkap respons apa pun yang belum dikonfigurasi.
3. Jika diperlukan, deklarasikan pemetaan yang terdiri dari pasangan nilai kunci untuk memetakan parameter respons integrasi yang ditentukan ke parameter respons metode yang diberikan.
4. Jika perlu, tambahkan templat pemetaan tubuh untuk mengubah muatan respons integrasi yang diberikan menjadi muatan respons metode yang ditentukan.
5. Jika diperlukan, tentukan cara menangani konversi tipe untuk muatan biner.

Respons integrasi adalah respons HTTP yang merangkum respons backend. Untuk titik akhir HTTP, respons backend adalah respons HTTP. Kode status respons integrasi dapat mengambil kode status yang dikembalikan ke backend, dan badan respons integrasi adalah payload yang dikembalikan ke backend. Untuk titik akhir Lambda, respons backend adalah output yang dikembalikan dari

fungsi Lambda. Dengan integrasi Lambda, output fungsi Lambda dikembalikan sebagai respons. `200 OK` Payload dapat berisi hasil sebagai data JSON, termasuk string JSON atau objek JSON, atau pesan kesalahan sebagai objek JSON. Anda dapat menetapkan ekspresi reguler ke properti SelectionPattern untuk memetakan respons kesalahan terhadap respons kesalahan HTTP yang sesuai. Untuk informasi selengkapnya tentang respons kesalahan fungsi Lambda, lihat. Menangani kesalahan Lambda di API Gateway Dengan integrasi proxy Lambda, fungsi Lambda harus mengembalikan output dari format berikut:

```
{
    statusCode: "...",            // a valid HTTP status code
    headers: {
        custom-header: "..."      // any API-specific custom header
    },
    body: "...",                  // a JSON string.
    isBase64Encoded:  true|false  // for binary support
}
```

Tidak perlu memetakan respons fungsi Lambda ke respons HTTP yang tepat.

Untuk mengembalikan hasil ke klien, atur respons integrasi untuk meneruskan respons titik akhir melalui apa adanya ke respons metode yang sesuai. Atau Anda dapat memetakan data respons titik akhir ke data respons metode. Data respons yang dapat dipetakan mencakup kode status respons, parameter header respons, dan badan respons. Jika tidak ada respons metode yang ditentukan untuk kode status yang dikembalikan, API Gateway mengembalikan kesalahan 500. Untuk informasi selengkapnya, lihat Menggunakan template pemetaan untuk mengganti parameter permintaan dan respons API serta kode status.

Siapkan integrasi Lambda di API Gateway

Anda dapat mengintegrasikan metode API dengan fungsi Lambda menggunakan integrasi proxy Lambda atau integrasi Lambda non-proxy (kustom).

Dalam integrasi proxy Lambda, pengaturan yang diperlukan sederhana. Setel metode HTTP integrasi ke POST, URI endpoint integrasi ke ARN dari tindakan pemanggilan fungsi Lambda dari fungsi Lambda tertentu, dan berikan izin API Gateway untuk memanggil fungsi Lambda atas nama Anda.

Di integrasi non-proxy Lambda, selain langkah penyiapan integrasi proxy, Anda juga menentukan cara data permintaan masuk dipetakan ke permintaan integrasi dan bagaimana data respons integrasi yang dihasilkan dipetakan ke respons metode.

Topik

- Siapkan integrasi proxy Lambda di API Gateway
- Siapkan integrasi kustom Lambda di API Gateway
- Siapkan pemanggilan asinkron dari fungsi Lambda backend
- Menangani kesalahan Lambda di API Gateway

Siapkan integrasi proxy Lambda di API Gateway

Topik

- Memahami integrasi proxy API Gateway Lambda
- Support untuk header multi-nilai dan parameter string kueri
- Siapkan sumber daya proxy dengan integrasi proxy Lambda
- Siapkan integrasi proxy Lambda menggunakan AWS CLI
- Format input fungsi Lambda untuk integrasi proxy
- Format output dari fungsi Lambda untuk integrasi proxy

Memahami integrasi proxy API Gateway Lambda

Integrasi proxy Amazon API Gateway Lambda adalah mekanisme yang sederhana, kuat, dan gesit untuk membangun API dengan penyiapan metode API tunggal. Integrasi proxy Lambda memungkinkan klien untuk memanggil satu fungsi Lambda di backend. Fungsi ini mengakses banyak sumber daya atau fitur AWS layanan lain, termasuk memanggil fungsi Lambda lainnya.

Dalam integrasi proxy Lambda, ketika klien mengirimkan permintaan API, API Gateway meneruskan ke fungsi Lambda terintegrasi objek peristiwa, kecuali urutan parameter permintaan tidak dipertahankan. Data permintaan ini mencakup header permintaan, parameter string kueri, variabel jalur URL, payload, dan data konfigurasi API. Data konfigurasi dapat mencakup nama tahap penyebaran saat ini, variabel tahap, identitas pengguna, atau konteks otorisasi (jika ada). Fungsi backend Lambda mem-parsing data permintaan yang masuk untuk menentukan respons yang dikembalikan. Agar API Gateway meneruskan output Lambda sebagai respons API ke klien, fungsi Lambda harus mengembalikan hasilnya dalam format ini.

Karena API Gateway tidak terlalu banyak campur tangan antara klien dan fungsi Lambda backend untuk integrasi proxy Lambda, klien dan fungsi Lambda terintegrasi dapat beradaptasi dengan perubahan satu sama lain tanpa merusak pengaturan integrasi API yang ada. Untuk mengaktifkan ini, klien harus mengikuti protokol aplikasi yang diberlakukan oleh fungsi Lambda backend.

Anda dapat menyiapkan integrasi proxy Lambda untuk metode API apa pun. Tetapi integrasi proxy Lambda lebih kuat ketika dikonfigurasi untuk metode API yang melibatkan sumber daya proxy generik. Sumber daya proxy generik dapat dilambangkan dengan variabel jalur template khusus, placeholder metode catch-all `{proxy+}`ANY, atau keduanya. Klien dapat meneruskan input ke fungsi Lambda backend dalam permintaan masuk sebagai parameter permintaan atau muatan yang berlaku. Parameter permintaan termasuk header, variabel jalur URL, parameter string kueri, dan payload yang berlaku. Fungsi Lambda terintegrasi memverifikasi semua sumber input sebelum memproses permintaan dan menanggapi klien dengan pesan kesalahan yang berarti jika ada input yang diperlukan yang hilang.

Saat memanggil metode API yang terintegrasi dengan metode HTTP generik `ANY` dan sumber daya generik`{proxy+}`, klien mengirimkan permintaan dengan metode HTTP tertentu sebagai pengganti. `ANY` Klien juga menentukan jalur URL tertentu, bukan`{proxy+}`, dan menyertakan header yang diperlukan, parameter string kueri, atau payload yang berlaku.

Daftar berikut merangkum perilaku runtime dari berbagai metode API dengan integrasi proxy Lambda:

- `ANY /{proxy+}`: Klien harus memilih metode HTTP tertentu, harus menetapkan hierarki jalur sumber daya tertentu, dan dapat mengatur header, parameter string kueri, dan muatan yang berlaku untuk meneruskan data sebagai input ke fungsi Lambda terintegrasi.
- `ANY /res`: Klien harus memilih metode HTTP tertentu dan dapat mengatur header, parameter string kueri, dan payload yang berlaku untuk meneruskan data sebagai input ke fungsi Lambda terintegrasi.
- `GET|POST|PUT|... /{proxy+}`: Klien dapat mengatur hierarki jalur sumber daya tertentu, header apa pun, parameter string kueri, dan muatan yang berlaku untuk meneruskan data sebagai input ke fungsi Lambda terintegrasi.
- `GET|POST|PUT|... /res/{path}/...`: Klien harus memilih segmen jalur tertentu (untuk `{path}` variabel) dan dapat mengatur header permintaan, parameter string kueri, dan payload yang berlaku untuk meneruskan data input ke fungsi Lambda terintegrasi.
- `GET|POST|PUT|... /res`: Klien dapat memilih header permintaan, parameter string kueri, dan payload yang berlaku untuk meneruskan data input ke fungsi Lambda terintegrasi.

Baik sumber daya proxy `{proxy+}` dan sumber daya kustom `{custom}` dinyatakan sebagai variabel jalur templat. Namun `{proxy+}` dapat merujuk ke sumber daya apa pun di sepanjang hierarki jalur, sementara `{custom}` mengacu pada segmen jalur tertentu saja. Misalnya, toko

kelontong mungkin mengatur inventaris produk online-nya berdasarkan nama departemen, kategori produksi, dan jenis produk. Situs web toko kelontong kemudian dapat mewakili produk yang tersedia dengan variabel jalur template berikut dari sumber daya khusus:. `/{department}/{produce-category}/{product-type}` Misalnya, apel diwakili oleh `/produce/fruit/apple` dan wortel oleh`/produce/vegetables/carrot`. Ini juga dapat digunakan `/{proxy+}` untuk mewakili departemen apa pun, kategori produk apa pun, atau jenis produk apa pun yang dapat dicari pelanggan saat berbelanja di toko online. Misalnya, `/{proxy+}` dapat merujuk ke salah satu item berikut:

- `/produce`
- `/produce/fruit`
- `/produce/vegetables/carrot`

Untuk memungkinkan pelanggan mencari produk apa pun yang tersedia, kategori produksinya, dan departemen toko terkait, Anda dapat mengekspos satu metode `GET /{proxy+}` dengan izin hanya-baca. Demikian pula, untuk memungkinkan supervisor memperbarui inventaris `produce` departemen, Anda dapat mengatur metode tunggal lain `PUT /produce/{proxy+}` dengan izin baca/tulis. Untuk memungkinkan kasir memperbarui total sayuran yang sedang berjalan, Anda dapat mengatur `POST /produce/vegetables/{proxy+}` metode dengan izin baca/tulis. Untuk membiarkan manajer toko melakukan tindakan apa pun yang mungkin pada produk apa pun yang tersedia, pengembang toko online dapat mengekspos `ANY /{proxy+}` metode dengan izin baca/tulis. Dalam kasus apa pun, pada waktu berjalan, pelanggan atau karyawan harus memilih produk tertentu dari jenis tertentu di departemen yang dipilih, kategori produk tertentu di departemen yang dipilih, atau departemen tertentu.

Untuk informasi selengkapnya tentang menyiapkan integrasi proxy API Gateway, lihatMenyiapkan integrasi proxy dengan sumber daya proxy.

Integrasi proxy mengharuskan klien memiliki pengetahuan yang lebih rinci tentang persyaratan backend. Oleh karena itu, untuk memastikan kinerja aplikasi dan pengalaman pengguna yang optimal, pengembang backend harus berkomunikasi dengan jelas kepada pengembang klien persyaratan backend, dan memberikan mekanisme umpan balik kesalahan yang kuat ketika persyaratan tidak terpenuhi.

Support untuk header multi-nilai dan parameter string kueri

API Gateway mendukung beberapa header dan parameter string kueri yang memiliki nama yang sama. Header multi-nilai serta header dan parameter nilai tunggal dapat digabungkan dalam permintaan dan tanggapan yang sama. Lihat informasi yang lebih lengkap di Format input fungsi Lambda untuk integrasi proxy dan Format output dari fungsi Lambda untuk integrasi proxy.

Siapkan sumber daya proxy dengan integrasi proxy Lambda

Untuk menyiapkan sumber daya proxy dengan tipe integrasi proxy Lambda, buat sumber daya API dengan parameter jalur serakah (misalnya,`/parent/{proxy+}`) dan integrasikan sumber daya ini dengan backend fungsi Lambda (misalnya,) pada metode. `arn:aws:lambda:us-west-2:123456789012:function:SimpleLambda4ProxyResource` ANY Parameter jalur serakah harus berada di akhir jalur sumber daya API. Seperti sumber daya non-proxy, Anda dapat mengatur sumber daya proxy dengan menggunakan konsol API Gateway, mengimpor file definisi OpenAPI, atau memanggil API REST API Gateway API secara langsung.

File definisi API OpenAPI berikut menunjukkan contoh API dengan sumber daya proxy yang terintegrasi dengan fungsi Lambda bernama. `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"
        }
      }
    }
  }
}
```

Dalam integrasi proxy Lambda, pada waktu berjalan, API Gateway memetakan permintaan masuk ke `event` parameter input fungsi Lambda. Input mencakup metode permintaan, jalur, header, parameter string kueri apa pun, payload apa pun, konteks terkait, dan variabel tahap yang ditentukan. Format

input dijelaskan dalamFormat input fungsi Lambda untuk integrasi proxy. Agar API Gateway berhasil memetakan output Lambda ke respons HTTP, fungsi Lambda harus menampilkan hasil dalam format yang dijelaskan dalam. Format output dari fungsi Lambda untuk integrasi proxy

Dalam integrasi proxy Lambda dari sumber daya proxy melalui ANY metode, fungsi Lambda backend tunggal berfungsi sebagai event handler untuk semua permintaan melalui sumber daya proxy. Misalnya, untuk mencatat pola lalu lintas, Anda dapat meminta perangkat seluler mengirim informasi lokasi negara bagian, kota, jalan, dan bangunannya dengan mengirimkan permintaan `/state/city/street/house` di jalur URL untuk sumber daya proxy. Fungsi backend Lambda kemudian dapat mengurai jalur URL dan menyisipkan tupel lokasi ke dalam tabel DynamoDB.

Siapkan integrasi proxy Lambda menggunakan AWS CLI

Di bagian ini, kami menunjukkan cara menggunakan AWS CLI untuk menyiapkan API dengan integrasi proxy Lambda.

> **Note**
>
> Untuk petunjuk mendetail tentang penggunaan konsol API Gateway guna mengonfigurasi sumber daya proxy dengan integrasi proxy Lambda, lihat. Tutorial: Bangun API REST Hello World dengan integrasi proxy Lambda

Sebagai contoh, kami menggunakan contoh fungsi Lambda berikut sebagai backend API:

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

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

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

Membandingkan ini dengan pengaturan integrasi kustom Lambda, input ke fungsi Lambda ini dapat dinyatakan dalam parameter permintaan dan isi. Anda memiliki lebih banyak garis lintang untuk memungkinkan klien meneruskan data input yang sama. Di sini, klien dapat meneruskan nama penyambut sebagai parameter string kueri, header, atau properti tubuh. Fungsi ini juga dapat mendukung integrasi kustom Lambda. Penyiapan API lebih sederhana. Anda tidak mengonfigurasi respons metode atau respons integrasi sama sekali.

Untuk mengatur integrasi proxy Lambda menggunakan AWS CLI

1. Panggil `create-rest-api` perintah untuk membuat API:

   ```
   aws apigateway create-rest-api --name 'HelloWorld (AWS CLI)' --region us-west-2
   ```

   Perhatikan `id` nilai (`te6si5ach7`) API yang dihasilkan dalam respons:

   ```
   {
       "name": "HelloWorldProxy (AWS CLI)",
       "id": "te6si5ach7",
       "createdDate": 1508461860
   }
   ```

   Anda memerlukan API di `id` seluruh bagian ini.
2. Panggil `get-resources` perintah untuk mendapatkan sumber daya root`id`:

   ```
   aws apigateway get-resources --rest-api-id te6si5ach7 --region us-west-2
   ```

Respon yang berhasil ditunjukkan sebagai berikut:

```
{
    "items": [
        {
            "path": "/",
            "id": "krznpq9xpg"
        }
    ]
}
```

Perhatikan `id` nilai sumber daya root (`krznpq9xpg`). Anda membutuhkannya di langkah berikutnya dan nanti.

3. Panggilan `create-resource` untuk membuat Sumber Daya API Gateway dari`/greeting`:

```
aws apigateway create-resource --rest-api-id te6si5ach7 \
      --region us-west-2 \
      --parent-id krznpq9xpg \
      --path-part {proxy+}
```

Respons yang berhasil mirip dengan yang berikut:

```
{
    "path": "/{proxy+}",
    "pathPart": "{proxy+}",
    "id": "2jf6xt",
    "parentId": "krznpq9xpg"
}
```

Perhatikan `id` nilai `{proxy+}` sumber daya yang dihasilkan (`2jf6xt`). Anda membutuhkannya untuk membuat metode pada `/{proxy+}` sumber daya di langkah berikutnya.

4. Panggilan `put-method` untuk membuat permintaan `ANY` metode`ANY /{proxy+}`:

```
aws apigateway put-method --rest-api-id te6si5ach7 \
       --region us-west-2 \
       --resource-id 2jf6xt \
       --http-method ANY \
       --authorization-type "NONE"
```

Respons yang berhasil mirip dengan yang berikut:

```
{
    "apiKeyRequired": false,
    "httpMethod": "ANY",
    "authorizationType": "NONE"
}
```

Metode API ini memungkinkan klien untuk menerima atau mengirim salam dari fungsi Lambda di backend.

5. Panggilan `put-integration` untuk mengatur integrasi `ANY /{proxy+}` metode dengan fungsi Lambda, bernama. `HelloWorld` Fungsi ini menanggapi permintaan dengan pesan`"Hello, {name}!"`, jika `greeter` parameter disediakan, atau`"Hello, World!"`, jika parameter string kueri tidak diatur.

```
aws apigateway put-integration \
        --region us-west-2 \
        --rest-api-id te6si5ach7 \
        --resource-id 2jf6xt \
        --http-method ANY \
        --type AWS_PROXY \
        --integration-http-method POST \
        --uri arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:HelloWorld/invocations \
        --credentials arn:aws:iam::123456789012:role/apigAwsProxyRole
```

> **Important**
>
> Untuk integrasi Lambda, Anda harus menggunakan metode HTTP `POST` untuk permintaan integrasi, sesuai dengan spesifikasi tindakan layanan Lambda untuk pemanggilan fungsi. Peran IAM `apigAwsProxyRole` harus memiliki kebijakan yang memungkinkan `apigateway` layanan untuk menjalankan fungsi Lambda. Untuk informasi selengkapnya tentang izin IAM, lihat. the section called “ Model izin API Gateway untuk menjalankan API”

Output yang berhasil mirip dengan yang berikut:

```
{
    "passthroughBehavior": "WHEN_NO_MATCH",
    "cacheKeyParameters": [],
    "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:1234567890:function:HelloWorld/invocations",
    "httpMethod": "POST",
    "cacheNamespace": "vvom7n",
    "credentials": "arn:aws:iam::1234567890:role/apigAwsProxyRole",
    "type": "AWS_PROXY"
}
```

Alih-alih menyediakan peran IAM`credentials`, Anda dapat memanggil perintah add-permission untuk menambahkan izin berbasis sumber daya. Inilah yang dilakukan konsol API Gateway.

6. Panggilan `create-deployment` untuk menerapkan API ke `test` panggung:

```
aws apigateway create-deployment --rest-api-id te6si5ach7 --stage-name test --
region us-west-2
```

7. Uji API menggunakan perintah cURL berikut di terminal.

   Memanggil API dengan parameter string kueri`?greeter=jane`:

```
curl -X GET 'https://te6si5ach7.execute-api.us-west-2.amazonaws.com/test/greeting?
greeter=jane'
```

   Memanggil API dengan parameter header`greeter:jane`:

```
curl -X GET https://te6si5ach7.execute-api.us-west-2.amazonaws.com/test/hi \
  -H 'content-type: application/json' \
  -H 'greeter: jane'
```

   Memanggil API dengan badan`{"greeter":"jane"}`:

```
curl -X POST https://te6si5ach7.execute-api.us-west-2.amazonaws.com/test/hi \
  -H 'content-type: application/json' \
  -d '{ "greeter": "jane" }'
```

   Dalam semua kasus, outputnya adalah respons 200 dengan badan respons berikut:

```
Hello, jane!
```

Format input fungsi Lambda untuk integrasi proxy

Dalam integrasi proxy Lambda, API Gateway memetakan seluruh permintaan klien ke `event` parameter input fungsi Lambda backend. Contoh berikut menunjukkan struktur peristiwa yang dikirimkan API Gateway ke integrasi proxy Lambda.

```
{
  "resource": "/my/path",
  "path": "/my/path",
  "httpMethod": "GET",
  "headers": {
    "header1": "value1",
    "header2": "value1,value2"
  },
  "multiValueHeaders": {
    "header1": [
      "value1"
    ],
    "header2": [
      "value1",
      "value2"
    ]
  },
  "queryStringParameters": {
    "parameter1": "value1,value2",
    "parameter2": "value"
  },
  "multiValueQueryStringParameters": {
    "parameter1": [
      "value1",
      "value2"
    ],
    "parameter2": [
      "value"
    ]
  },
  "requestContext": {
    "accountId": "123456789012",
    "apiId": "id",
```

```
    "authorizer": {
      "claims": null,
      "scopes": null
    },
    "domainName": "id.execute-api.us-east-1.amazonaws.com",
    "domainPrefix": "id",
    "extendedRequestId": "request-id",
    "httpMethod": "GET",
    "identity": {
      "accessKey": null,
      "accountId": null,
      "caller": null,
      "cognitoAuthenticationProvider": null,
      "cognitoAuthenticationType": null,
      "cognitoIdentityId": null,
      "cognitoIdentityPoolId": null,
      "principalOrgId": null,
      "sourceIp": "IP",
      "user": null,
      "userAgent": "user-agent",
      "userArn": null,
      "clientCert": {
        "clientCertPem": "CERT_CONTENT",
        "subjectDN": "www.example.com",
        "issuerDN": "Example issuer",
        "serialNumber": "a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1",
        "validity": {
          "notBefore": "May 28 12:30:02 2019 GMT",
          "notAfter": "Aug  5 09:36:04 2021 GMT"
        }
      }
    },
    "path": "/my/path",
    "protocol": "HTTP/1.1",
    "requestId": "id=",
    "requestTime": "04/Mar/2020:19:15:17 +0000",
    "requestTimeEpoch": 1583349317135,
    "resourceId": null,
    "resourcePath": "/my/path",
    "stage": "$default"
  },
  "pathParameters": null,
  "stageVariables": null,
  "body": "Hello from Lambda!",
```

```
  "isBase64Encoded": false
}
```

> **Note**
>
> Dalam masukan:
>
> - `headers`Kuncinya hanya dapat berisi header nilai tunggal.
> - `multiValueHeaders`Kuncinya dapat berisi header multi-nilai serta header nilai tunggal.
> - Jika Anda menentukan nilai untuk keduanya `headers` dan`multiValueHeaders`, API Gateway menggabungkannya ke dalam satu daftar. Jika pasangan kunci-nilai yang sama ditentukan di keduanya, hanya nilai dari yang `multiValueHeaders` akan muncul dalam daftar gabungan.

Dalam input ke fungsi Lambda backend, `requestContext` objek adalah peta pasangan kunci-nilai. Di setiap pasangan, kuncinya adalah nama properti variabel $context, dan nilainya adalah nilai properti itu. API Gateway dapat menambahkan kunci baru ke peta.

Bergantung pada fitur yang diaktifkan, `requestContext` peta dapat bervariasi dari API ke API. Misalnya, dalam contoh sebelumnya, tidak ada jenis otorisasi yang ditentukan, jadi tidak ada `$context.authorizer.*` atau `$context.identity.*` properti yang ada. Ketika jenis otorisasi ditentukan, ini menyebabkan API Gateway meneruskan informasi pengguna yang diotorisasi ke titik akhir integrasi dalam `requestContext.identity` objek sebagai berikut:

- Ketika jenis otorisasi`AWS_IAM`, informasi pengguna yang berwenang mencakup `$context.identity.*` properti.
- Ketika jenis otorisasi adalah `COGNITO_USER_POOLS` (Amazon Cognito Authorizer), informasi `$context.identity.cognito*` pengguna yang berwenang termasuk dan properti. `$context.authorizer.claims.*`
- Ketika jenis otorisasi adalah `CUSTOM` (Lambda Authorizer), informasi pengguna yang berwenang `$context.authorizer.principalId` termasuk dan properti lain yang berlaku. `$context.authorizer.*`

Format output dari fungsi Lambda untuk integrasi proxy

Dalam integrasi proxy Lambda, API Gateway memerlukan fungsi Lambda backend untuk mengembalikan output sesuai dengan format JSON berikut:

```
{
    "isBase64Encoded": true|false,
    "statusCode": httpStatusCode,
    "headers": { "headerName": "headerValue", ... },
    "multiValueHeaders": { "headerName": ["headerValue", "headerValue2", ...], ... },
    "body": "..."
}
```

Dalam output:

- `multiValueHeaders`Tombol `headers` dan dapat tidak ditentukan jika tidak ada header respons tambahan yang akan dikembalikan.
- `headers`Kuncinya hanya dapat berisi header nilai tunggal.
- `multiValueHeaders`Kuncinya dapat berisi header multi-nilai serta header nilai tunggal. Anda dapat menggunakan `multiValueHeaders` kunci untuk menentukan semua header tambahan Anda, termasuk yang bernilai tunggal.
- Jika Anda menentukan nilai untuk keduanya `headers` dan`multiValueHeaders`, API Gateway menggabungkannya ke dalam satu daftar. Jika pasangan kunci-nilai yang sama ditentukan di keduanya, hanya nilai dari yang `multiValueHeaders` akan muncul dalam daftar gabungan.

Untuk mengaktifkan CORS untuk integrasi proxy Lambda, Anda harus `Access-Control-Allow-Origin:`*`domain-name`* menambahkan ke output. `headers` *`domain-name`*bisa * untuk nama domain apa pun. Output body disusun ke frontend sebagai payload respons metode. Jika **body** adalah gumpalan biner, Anda dapat menyandikannya sebagai string yang dikodekan Base64 dengan menyetel **`true`** dan **`isBase64Encoded`** mengonfigurasi **`*/*`** sebagai Tipe Media Biner. Jika tidak, Anda dapat mengaturnya ke `false` atau membiarkannya tidak ditentukan.

> **Note**
>
> Untuk informasi selengkapnya tentang mengaktifkan dukungan biner, lihatMengaktifkan dukungan biner menggunakan konsol API Gateway. Untuk contoh fungsi Lambda, lihat. Kembalikan media biner dari integrasi proxy Lambda

Jika output fungsi dari format yang berbeda, API Gateway mengembalikan respon `502 Bad Gateway` kesalahan.

Untuk mengembalikan respons dalam fungsi Lambda di Node.js, Anda dapat menggunakan perintah seperti berikut:

- Untuk mengembalikan hasil yang sukses, hubungi`callback(null, {"statusCode": 200, "body": "results"})`.
- Untuk melempar pengecualian, panggil`callback(new Error('internal server error'))`.
- Untuk kesalahan sisi klien (jika, misalnya, parameter yang diperlukan tidak ada), Anda dapat memanggil `callback(null, {"statusCode": 400, "body": "Missing parameters of ..."})` untuk mengembalikan kesalahan tanpa melempar pengecualian.

Dalam `async` fungsi Lambda di Node.js, sintaks yang setara adalah:

- Untuk mengembalikan hasil yang sukses, hubungi`return {"statusCode": 200, "body": "results"}`.
- Untuk melempar pengecualian, panggil`throw new Error("internal server error")`.
- Untuk kesalahan sisi klien (jika, misalnya, parameter yang diperlukan tidak ada), Anda dapat memanggil `return {"statusCode": 400, "body": "Missing parameters of ..."}` untuk mengembalikan kesalahan tanpa melempar pengecualian.

Siapkan integrasi kustom Lambda di API Gateway

Untuk menunjukkan cara mengatur integrasi kustom Lambda, kami membuat API Gateway API untuk mengekspos `GET /greeting?greeter={name}` metode untuk menjalankan fungsi Lambda. Gunakan salah satu contoh fungsi Lambda berikut untuk API Anda.

Gunakan salah satu contoh fungsi Lambda berikut:

Node.js

```
export const handler = function(event, context, callback) {
    var res ={
        "statusCode": 200,
        "headers": {
            "Content-Type": "*/*"
```

```
        }
    };
    if (event.greeter==null) {
        callback(new Error('Missing the required greeter parameter.'));
    } else if (event.greeter === "") {
        res.body = "Hello, World";
        callback(null, res);
    } else {
        res.body = "Hello, " + event.greeter +"!";
        callback(null, res);
    }
};
```

Python

```
import json


def lambda_handler(event, context):
    print(event)
    res = {
        "statusCode": 200,
        "headers": {
            "Content-Type": "*/*"
        }
    }

    if event['greeter'] == "":
        res['body'] = "Hello, World"
    elif (event['greeter']):
        res['body'] = "Hello, " + event['greeter'] + "!"
    else:
        raise Exception('Missing the required greeter parameter.')

    return res
```

Fungsi merespons dengan pesan `"Hello, {name}!"` jika nilai `greeter` parameter adalah string yang tidak kosong. Ia mengembalikan pesan `"Hello, World!"` jika `greeter` nilai adalah string kosong. Fungsi mengembalikan pesan kesalahan `"Missing the required greeter parameter."` jika parameter penyambut tidak diatur dalam permintaan masuk. Kami menamai fungsinya`HelloWorld`.

Anda dapat membuatnya di konsol Lambda atau dengan menggunakan. AWS CLI Pada bagian ini, kami mereferensikan fungsi ini menggunakan ARN berikut:

```
arn:aws:lambda:us-east-1:123456789012:function:HelloWorld
```

Dengan fungsi Lambda diatur di backend, lanjutkan untuk mengatur API.

Untuk mengatur integrasi kustom Lambda menggunakan AWS CLI

1. Panggil `create-rest-api` perintah untuk membuat API:

   ```
   aws apigateway create-rest-api --name 'HelloWorld (AWS CLI)' --region us-west-2
   ```

   Perhatikan `id` nilai (`te6si5ach7`) API yang dihasilkan dalam respons:

   ```
   {
       "name": "HelloWorld (AWS CLI)",
       "id": "te6si5ach7",
       "createdDate": 1508461860
   }
   ```

   Anda memerlukan API di `id` seluruh bagian ini.
2. Panggil `get-resources` perintah untuk mendapatkan sumber daya root`id`:

   ```
   aws apigateway get-resources --rest-api-id te6si5ach7 --region us-west-2
   ```

   Respon yang berhasil adalah sebagai berikut:

   ```
   {
       "items": [
           {
               "path": "/",
               "id": "krznpq9xpg"
           }
       ]
   }
   ```

   Perhatikan `id` nilai sumber daya root (`krznpq9xpg`). Anda membutuhkannya di langkah berikutnya dan nanti.

3. Panggilan `create-resource` untuk membuat Sumber Daya API Gateway dari`/greeting`:

```
aws apigateway create-resource --rest-api-id te6si5ach7 \
      --region us-west-2 \
      --parent-id krznpq9xpg \
      --path-part greeting
```

Respons yang berhasil mirip dengan yang berikut:

```
{
    "path": "/greeting",
    "pathPart": "greeting",
    "id": "2jf6xt",
    "parentId": "krznpq9xpg"
}
```

Perhatikan `id` nilai `greeting` sumber daya yang dihasilkan (`2jf6xt`). Anda membutuhkannya untuk membuat metode pada `/greeting` sumber daya di langkah berikutnya.

4. Panggilan `put-method` untuk membuat permintaan metode API dari`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
```

Respons yang berhasil mirip dengan yang berikut:

```
{
    "apiKeyRequired": false,
    "httpMethod": "GET",
    "authorizationType": "NONE",
    "requestParameters": {
        "method.request.querystring.greeter": false
    }
}
```

Metode API ini memungkinkan klien untuk menerima salam dari fungsi Lambda di backend. `greeter`Parameternya opsional karena backend harus menangani penelepon anonim atau penelepon yang diidentifikasi sendiri.

5. Panggilan `put-method-response` untuk mengatur `200 OK` respons terhadap permintaan metode`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. Panggilan `put-integration` untuk mengatur integrasi `GET /greeting?greeter={name}` metode dengan fungsi Lambda, bernama. `HelloWorld` Fungsi merespons permintaan dengan pesan`"Hello, {name}!"`, jika `greeter` parameter disediakan, atau`"Hello, World!"`, jika parameter string query tidak diatur.

```
aws apigateway put-integration \
        --region us-west-2 \
        --rest-api-id te6si5ach7 \
        --resource-id 2jf6xt \
        --http-method GET \
        --type AWS \
        --integration-http-method POST \
        --uri arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:HelloWorld/invocations \
        --request-templates '{"application/json":"{\"greeter\":
\"$input.params('greeter')\"}"}' \
        --credentials arn:aws:iam::123456789012:role/apigAwsProxyRole
```

Template pemetaan yang disediakan di sini menerjemahkan parameter string `greeter` kueri ke `greeter` properti payload JSON. Ini diperlukan karena input ke fungsi Lambda harus diekspresikan dalam tubuh.

> **Important**
>
> Untuk integrasi Lambda, Anda harus menggunakan metode HTTP POST untuk permintaan integrasi, sesuai dengan spesifikasi tindakan layanan Lambda untuk pemanggilan fungsi. `uri`Parameternya adalah ARN dari tindakan pemanggilan fungsi. Output yang berhasil mirip dengan yang berikut ini:

```
{
    "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"
}
```

Peran IAM `apigAwsProxyRole` harus memiliki kebijakan yang memungkinkan `apigateway` layanan untuk menjalankan fungsi Lambda. Alih-alih menyediakan peran IAM`credentials`, Anda dapat memanggil perintah add-permission untuk menambahkan izin berbasis sumber daya. Beginilah cara konsol API Gateway menambahkan izin ini.

7. Panggilan `put-integration-response` untuk mengatur respons integrasi untuk meneruskan output fungsi Lambda ke klien sebagai respons `200 OK` metode.

```
aws apigateway put-integration-response \
       --region us-west-2 \
       --rest-api-id te6si5ach7 \
       --resource-id 2jf6xt \
       --http-method GET \
       --status-code 200 \
       --selection-pattern ""
```

Dengan mengatur pola pilihan ke string kosong, `200 OK` responsnya adalah default.

Respons yang berhasil harus serupa dengan yang berikut:

```
 {
    "selectionPattern": "",
    "statusCode": "200"
}
```

8. Panggilan `create-deployment` untuk menerapkan API ke `test` panggung:

```
aws apigateway create-deployment --rest-api-id te6si5ach7 --stage-name test --
region us-west-2
```

9. Uji API menggunakan perintah cURL berikut di 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'
```

Siapkan pemanggilan asinkron dari fungsi Lambda backend

Dalam integrasi Lambda non-proxy (kustom), fungsi Lambda backend dipanggil secara sinkron secara default. Ini adalah perilaku yang diinginkan untuk sebagian besar operasi REST API. Beberapa aplikasi, bagaimanapun, memerlukan pekerjaan yang harus dilakukan secara asinkron (sebagai operasi batch atau operasi latensi panjang), biasanya oleh komponen backend terpisah. Dalam hal ini, fungsi Lambda backend dipanggil secara asinkron, dan metode REST API front-end tidak mengembalikan hasilnya.

Anda dapat mengonfigurasi fungsi Lambda agar integrasi non-proxy Lambda dipanggil secara asinkron dengan menentukan sebagai jenis pemanggilan Lambda. `'Event'` Ini dilakukan sebagai berikut:

Konfigurasikan pemanggilan asinkron Lambda di konsol API Gateway

Agar semua pemanggilan menjadi asinkron:

- Dalam permintaan Integrasi, tambahkan `X-Amz-Invocation-Type` header dengan nilai statis `'Event'`.

Bagi klien untuk memutuskan apakah pemanggilan asinkron atau sinkron:

1. Dalam permintaan Metode, tambahkan `InvocationType` header.
2. Dalam permintaan Integrasi tambahkan `X-Amz-Invocation-Type` header dengan ekspresi pemetaan. `method.request.header.InvocationType`
3. Klien dapat menyertakan `InvocationType: Event` header dalam permintaan API untuk pemanggilan asinkron atau untuk pemanggilan sinkron. `InvocationType: RequestResponse`

Konfigurasikan pemanggilan asinkron Lambda menggunakan OpenAPI

Agar semua pemanggilan menjadi asinkron:

- Tambahkan `X-Amz-Invocation-Type` header ke x-amazon-apigateway-integrationbagian.

```
"x-amazon-apigateway-integration" : {
          "type" : "aws",
          "httpMethod" : "POST",
          "uri" : "arn:aws:apigateway:us-east-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-2:123456789012:function:my-function/invocations",
          "responses" : {
            "default" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.header.X-Amz-Invocation-Type" : "'Event'"
          },
          "passthroughBehavior" : "when_no_match",
          "contentHandling" : "CONVERT_TO_TEXT"
        }
```

Bagi klien untuk memutuskan apakah pemanggilan asinkron atau sinkron:

1. Tambahkan header berikut pada OpenAPI Path Item Object.

```
"parameters" : [ {
"name" : "InvocationType",
"in" : "header",
"schema" : {
  "type" : "string"
}
} ]
```

2. Tambahkan `X-Amz-Invocation-Type` header ke x-amazon-apigateway-integrationbagian.

```
"x-amazon-apigateway-integration" : {
          "type" : "aws",
          "httpMethod" : "POST",
          "uri" : "arn:aws:apigateway:us-east-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-2:123456789012:function:my-function/invocations",
          "responses" : {
            "default" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.header.X-Amz-Invocation-Type" :
 "method.request.header.InvocationType"
          },
          "passthroughBehavior" : "when_no_match",
          "contentHandling" : "CONVERT_TO_TEXT"
        }
```

3. Klien dapat menyertakan `InvocationType: Event` header dalam permintaan API untuk pemanggilan asinkron atau untuk pemanggilan sinkron. `InvocationType: RequestResponse`

Menangani kesalahan Lambda di API Gateway

Untuk integrasi kustom Lambda, Anda harus memetakan kesalahan yang dikembalikan oleh Lambda dalam respons integrasi terhadap respons kesalahan HTTP standar untuk klien Anda. Jika tidak, kesalahan Lambda dikembalikan sebagai `200 OK` respons secara default dan hasilnya tidak intuitif untuk pengguna API Anda.

Ada dua jenis kesalahan yang dapat dikembalikan Lambda: kesalahan standar dan kesalahan khusus. Di API Anda, Anda harus menangani ini secara berbeda.

Dengan integrasi proxy Lambda, Lambda diperlukan untuk mengembalikan output dari format berikut:

```
{
  "isBase64Encoded" : "boolean",
  "statusCode": "number",
  "headers": { ... },
  "body": "JSON string"
}
```

Dalam output ini, `statusCode` biasanya `4XX` untuk kesalahan klien dan `5XX` untuk kesalahan server. API Gateway menangani kesalahan ini dengan memetakan kesalahan Lambda ke respons kesalahan HTTP, sesuai dengan yang ditentukan. `statusCode` Agar API Gateway dapat meneruskan jenis kesalahan (misalnya,`InvalidParameterException`), sebagai bagian dari respons terhadap klien, fungsi Lambda harus menyertakan header (misalnya,`"X-Amzn-ErrorType":"InvalidParameterException"`) di `headers` properti.

Topik

- Menangani kesalahan Lambda standar di API Gateway
- Menangani kesalahan Lambda khusus di API Gateway

Menangani kesalahan Lambda standar di API Gateway

AWS LambdaKesalahan standar memiliki format berikut:

```
{
  "errorMessage": "<replaceable>string</replaceable>",
  "errorType": "<replaceable>string</replaceable>",
  "stackTrace": [
    "<replaceable>string</replaceable>",
    ...
  ]
}
```

Di sini, `errorMessage` adalah ekspresi string dari kesalahan. `errorType`Ini adalah kesalahan atau tipe pengecualian yang bergantung pada bahasa. `stackTrace`Ini adalah daftar ekspresi string yang menunjukkan jejak tumpukan yang mengarah ke terjadinya kesalahan.

Misalnya, perhatikan fungsi Lambda berikut JavaScript (Node.js).

```
export const handler = function(event, context, callback) {
```

```
    callback(new Error("Malformed input ..."));
};
```

Fungsi ini mengembalikan kesalahan Lambda standar berikut, yang berisi `Malformed input ...` sebagai pesan kesalahan:

```
{
  "errorMessage": "Malformed input ...",
  "errorType": "Error",
  "stackTrace": [
    "export const handler (/var/task/index.js:3:14)"
  ]
}
```

Demikian pula, pertimbangkan fungsi Lambda Python berikut, yang memunculkan pesan kesalahan `Exception` yang sama. `Malformed input ...`

```
def lambda_handler(event, context):
    raise Exception('Malformed input ...')
```

Fungsi ini mengembalikan kesalahan Lambda standar berikut:

```
{
  "stackTrace": [
    [
      "/var/task/lambda_function.py",
      3,
      "lambda_handler",
      "raise Exception('Malformed input ...')"
    ]
  ],
  "errorType": "Exception",
  "errorMessage": "Malformed input ..."
}
```

Perhatikan bahwa nilai `errorType` dan `stackTrace` properti bergantung pada bahasa. Kesalahan standar juga berlaku untuk objek kesalahan apa pun yang merupakan perpanjangan dari `Error` objek atau subkelas `Exception` kelas.

Untuk memetakan kesalahan Lambda standar ke respons metode, Anda harus terlebih dahulu memutuskan kode status HTTP untuk kesalahan Lambda yang diberikan. Anda

kemudian menetapkan pola ekspresi reguler pada `selectionPattern` properti yang IntegrationResponseterkait dengan kode status HTTP yang diberikan. Di konsol API Gateway, ini `selectionPattern` dilambangkan sebagai regex kesalahan Lambda di bagian Respons Integrasi, di bawah setiap respons integrasi.

> **Note**
>
> API Gateway menggunakan regex gaya pola Java untuk pemetaan respons. Untuk informasi selengkapnya, lihat Pola dalam Oracle dokumentasi.

Misalnya, untuk mengatur `selectionPattern` ekspresi baru, menggunakanAWS CLI, panggil put-integration-responseperintah berikut:

```
aws apigateway put-integration-response --rest-api-id z0vprf0mdh --resource-id x3o5ih
 --http-method GET --status-code 400 --selection-pattern "Malformed.*" --region us-
west-2
```

Pastikan bahwa Anda juga mengatur kode kesalahan yang sesuai (`400`) pada respon metode. Jika tidak, API Gateway akan menampilkan respons kesalahan konfigurasi yang tidak valid saat runtime.

> **Note**
>
> Saat runtime, API Gateway mencocokkan kesalahan `errorMessage` Lambda dengan pola ekspresi reguler pada `selectionPattern` properti. Jika ada kecocokan, API Gateway mengembalikan kesalahan Lambda sebagai respons HTTP dari kode status HTTP yang sesuai. Jika tidak ada kecocokan, API Gateway mengembalikan kesalahan sebagai respons default atau melempar pengecualian konfigurasi yang tidak valid jika tidak ada respons default yang dikonfigurasi.
>
> Menyetel `selectionPattern` nilai `.*` untuk respons yang diberikan berarti mengatur ulang respons ini sebagai respons default. Ini karena pola pemilihan seperti itu akan cocok dengan semua pesan kesalahan, termasuk null, yaitu, pesan kesalahan yang tidak ditentukan. Pemetaan yang dihasilkan mengesampingkan pemetaan default.

Untuk memperbarui `selectionPattern` nilai yang ada menggunakanAWS CLI, panggil update-integration-responseoperasi untuk mengganti nilai `/selectionPattern` jalur dengan ekspresi regex yang ditentukan dari pola. `Malformed*`

Untuk mengatur `selectionPattern` ekspresi menggunakan konsol API Gateway, masukkan ekspresi di kotak teks regex kesalahan Lambda saat menyiapkan atau memperbarui respons integrasi kode status HTTP yang ditentukan.

Menangani kesalahan Lambda khusus di API Gateway

Alih-alih kesalahan standar yang dijelaskan di bagian sebelumnya, AWS Lambda memungkinkan Anda mengembalikan objek kesalahan kustom sebagai string JSON. Kesalahan dapat berupa objek JSON yang valid. Misalnya, fungsi Lambda berikut JavaScript (Node.js) mengembalikan kesalahan kustom:

```
export const handler = (event, context, callback) => {
    ...
    // Error caught here:
    var myErrorObj = {
        errorType : "InternalServerError",
        httpStatus : 500,
        requestId : context.awsRequestId,
        trace : {
            "function": "abc()",
            "line": 123,
            "file": "abc.js"
        }
    }
    callback(JSON.stringify(myErrorObj));
};
```

Anda harus mengubah `myErrorObj` objek menjadi string JSON sebelum memanggil `callback` untuk keluar dari fungsi. Jika tidak, `myErrorObj` dikembalikan sebagai string dari`"[object Object]"`. Saat metode API Anda terintegrasi dengan fungsi Lambda sebelumnya, API Gateway menerima respons integrasi dengan muatan berikut:

```
{
    "errorMessage": "{\"errorType\":\"InternalServerError\",\"httpStatus\":500,
\"requestId\":\"e5849002-39a0-11e7-a419-5bb5807c9fb2\",\"trace\":{\"function\":
\"abc()\",\"line\":123,\"file\":\"abc.js\"}}"
}
```

Seperti halnya respons integrasi apa pun, Anda dapat melewati respons kesalahan ini apa adanya terhadap respons metode. Atau Anda dapat memiliki template pemetaan untuk mengubah payload

menjadi format yang berbeda. Misalnya, pertimbangkan template pemetaan tubuh berikut untuk respons metode kode status`500`:

```
{
    errorMessage: $input.path('$.errorMessage');
}
```

Template ini menerjemahkan badan respons integrasi yang berisi string JSON kesalahan kustom ke badan respons metode berikut. Badan respons metode ini berisi objek JSON kesalahan kustom:

```
{
    "errorMessage" : {
        errorType : "InternalServerError",
        httpStatus : 500,
        requestId : context.awsRequestId,
        trace : {
            "function": "abc()",
            "line": 123,
            "file": "abc.js"
        }
    }
};
```

Bergantung pada persyaratan API Anda, Anda mungkin perlu meneruskan beberapa atau semua properti kesalahan kustom sebagai parameter header respons metode. Anda dapat mencapai ini dengan menerapkan pemetaan kesalahan kustom dari badan respons integrasi ke header respons metode.

Misalnya, ekstensi OpenAPI berikut mendefinisikan pemetaan dari`errorMessage.errorType`,,, dan `errorMessage.trace` properti ke `errorMessage.httpStatuserrorMessage.trace.function`,,, dan `error_type` header `error_statuserror_trace_function`, masing-masing. `error_trace`

```
"x-amazon-apigateway-integration": {
    "responses": {
        "default": {
          "statusCode": "200",
          "responseParameters": {
            "method.response.header.error_trace_function":
 "integration.response.body.errorMessage.trace.function",
```

```
            "method.response.header.error_status":
 "integration.response.body.errorMessage.httpStatus",
            "method.response.header.error_type":
 "integration.response.body.errorMessage.errorType",
            "method.response.header.error_trace":
 "integration.response.body.errorMessage.trace"
          },
          ...
        }
    }
}
```

Saat runtime, API Gateway melakukan deserialisasi `integration.response.body` parameter saat melakukan pemetaan header. Namun, deserialisasi ini hanya berlaku untuk body-to-header pemetaan untuk respons kesalahan khusus Lambda dan tidak berlaku untuk pemetaan yang menggunakan. body-to-body `$input.body` Dengan pemetaan custom-error-body-to -header ini, klien menerima header berikut sebagai bagian dari respons metode, asalkan,, `error_status` `error_traceerror_trace_function`, dan `error_type` header dideklarasikan dalam permintaan metode.

```
"error_status":"500",
"error_trace":"{\"function\":\"abc()\",\"line\":123,\"file\":\"abc.js\"}",
"error_trace_function":"abc()",
"error_type":"InternalServerError"
```

`errorMessage.trace`Properti badan respons integrasi adalah properti yang kompleks. Hal ini dipetakan ke `error_trace` header sebagai string JSON.

Menyiapkan integrasi HTTP di API Gateway

Anda dapat mengintegrasikan metode API dengan endpoint HTTP menggunakan integrasi proxy HTTP atau integrasi kustom HTTP.

API Gateway mendukung port endpoint berikut: 80, 443 dan 1024-65535.

Dengan integrasi proxy, penyiapannya sederhana. Anda hanya perlu mengatur metode HTTP dan URI endpoint HTTP, sesuai dengan persyaratan backend, jika Anda tidak peduli dengan pengkodean konten atau caching.

Dengan integrasi kustom, pengaturan lebih terlibat. Selain langkah-langkah pengaturan integrasi proxy, Anda perlu menentukan bagaimana data permintaan masuk dipetakan ke permintaan integrasi dan bagaimana data respons integrasi yang dihasilkan dipetakan ke respons metode.

Topik

Menyiapkan integrasi proxy HTTP di API Gateway

Untuk menyiapkan sumber daya proxy dengan tipe integrasi proxy HTTP, buat sumber daya API dengan parameter jalur serakah (misalnya,`/parent/{proxy+}`) dan mengintegrasikan sumber daya ini dengan endpoint backend HTTP (misalnya,`https://petstore-demo-endpoint.execute-api.com/petstore/{proxy}`) padaANYmetode. Parameter jalur serakah harus berada di ujung jalur sumber daya.

Seperti halnya sumber daya non-proxy, Anda dapat mengatur sumber daya proxy dengan integrasi proxy HTTP dengan menggunakan konsol API Gateway, mengimpor file definisi OpenAPI, atau memanggil API Gateway REST API secara langsung. Untuk petunjuk mendetail tentang penggunaan konsol API Gateway untuk mengonfigurasi sumber daya proxy dengan integrasi HTTP, lihatTutorial: Membangun REST API dengan integrasi proxy HTTP.

File definisi OpenAPI berikut menunjukkan contoh API dengan sumber daya proxy yang terintegrasi denganPetStoresitus web.

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"
        }
      }
    }
  }
}
```

Dalam contoh ini, kunci cache dideklarasikan pada`method.request.path.proxy`parameter path dari sumber daya proxy. Ini adalah pengaturan default saat Anda membuat API menggunakan konsol API Gateway. Jalur dasar API (`/test`, sesuai dengan tahap) dipetakan ke situs web PetStore halaman (`/petstore`). Permintaan integrasi tunggal mencerminkan keseluruhan PetStore situs web menggunakan variabel path serakah API dan catch-all`ANY`metode. Contoh berikut menggambarkan pencerminan ini.

- set**`ANY`**sebagai**`GET`**dan**`{proxy+}`**sebagai**`pets`**

  Permintaan metode dimulai dari frontend:

  ```
  GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets HTTP/1.1
  ```

  Permintaan integrasi dikirim ke backend:

  ```
  GET http://petstore-demo-endpoint.execute-api.com/petstore/pets HTTP/1.1
  ```

  Contoh run-time dari`ANY`metode dan sumber daya proxy keduanya valid. Panggilan mengembalikan`200 OK`respon dengan muatan yang berisi batch pertama hewan peliharaan, seperti yang dikembalikan dari backend.

- set**`ANY`**sebagai**`GET`**dan**`{proxy+}`**sebagai**`pets?type=dog`**

  ```
  GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets?type=dog
   HTTP/1.1
  ```

  Permintaan integrasi dikirim ke backend:

  ```
  GET http://petstore-demo-endpoint.execute-api.com/petstore/pets?type=dog HTTP/1.1
  ```

Contoh run-time dariANYmetode dan sumber daya proxy keduanya valid. Panggilan mengembalikan`200 OK`respon dengan muatan yang berisi batch pertama anjing-anjing tertentu, seperti yang dikembalikan dari backend.

- set**`ANY`**sebagai**`GET`**dan**`{proxy+}`**sebagai**`pets/{petId}`**

  Permintaan metode dimulai dari frontend:

```
GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets/1 HTTP/1.1
```

  Permintaan integrasi dikirim ke backend:

```
GET http://petstore-demo-endpoint.execute-api.com/petstore/pets/1 HTTP/1.1
```

  Contoh run-time dariANYmetode dan sumber daya proxy keduanya valid. Panggilan mengembalikan`200 OK`respon dengan muatan yang berisi hewan peliharaan yang ditentukan, seperti yang dikembalikan dari backend.

- set**`ANY`**sebagai**`POST`**dan**`{proxy+}`**sebagai**`pets`**

  Permintaan metode dimulai dari frontend:

```
POST https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets HTTP/1.1
Content-Type: application/json
Content-Length: ...

{
  "type" : "dog",
  "price" : 1001.00
}
```

  Permintaan integrasi dikirim ke backend:

```
POST http://petstore-demo-endpoint.execute-api.com/petstore/pets HTTP/1.1
Content-Type: application/json
Content-Length: ...

{
  "type" : "dog",
  "price" : 1001.00
```

```
}
```

Contoh run-time dari`ANY`metode dan sumber daya proxy keduanya valid. Panggilan mengembalikan`200 OK`respon dengan muatan yang berisi hewan peliharaan yang baru dibuat, seperti yang dikembalikan dari backend.

- set**`ANY`**sebagai**`GET`**dan**`{proxy+}`**sebagai**`pets/cat`**

  Permintaan metode dimulai dari frontend:

  ```
  GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets/cat
  ```

  Permintaan integrasi dikirim ke backend:

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

  Instance run-time dari jalur sumber daya proxy tidak sesuai dengan endpoint backend dan permintaan yang dihasilkan tidak valid. Hasilnya, a`400 Bad Request`tanggapan dikembalikan dengan pesan kesalahan berikut.

  ```
  {
    "errors": [
      {
        "key": "Pet2.type",
        "message": "Missing required field"
      },
      {
        "key": "Pet2.price",
        "message": "Missing required field"
      }
    ]
  }
  ```

- set**`ANY`**sebagai**`GET`**dan**`{proxy+}`**sebagai**`null`**

  Permintaan metode dimulai dari frontend:

  ```
  GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test
  ```

  Permintaan integrasi dikirim ke backend:

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

Sumber daya yang ditargetkan adalah induk dari sumber daya proxy, tetapi contoh run-time dariANYmetode tidak didefinisikan dalam API pada sumber daya itu. Hasilnya, iniGETpermintaan mengembalikan`403 Forbidden`tanggapan dengan`Missing Authentication Token`pesan kesalahan seperti yang dikembalikan oleh API Gateway. Jika API mengeksposANYatauGETmetode di sumber daya induk (`/`), panggilan mengembalikan`404 Not Found`tanggapan dengan`Cannot GET /petstore`pesan yang dikembalikan dari backend.

Untuk setiap permintaan klien, jika URL endpoint yang ditargetkan tidak valid atau kata kerja HTTP valid tetapi tidak didukung, backend mengembalikan`404 Not Found`tanggapan. Untuk metode HTTP yang tidak didukung, a`403 Forbidden`tanggapan dikembalikan.

Menyiapkan integrasi kustom HTTP di API Gateway

Dengan integrasi kustom HTTP, Anda memiliki lebih banyak kontrol atas data mana yang akan dilewatkan antara metode API dan integrasi API dan cara meneruskan data. Anda melakukan ini menggunakan pemetaan data.

Sebagai bagian dari pengaturan permintaan metode, Anda mengaturrequestParametersatribut diMetodesumber daya. Ini menyatakan parameter permintaan metode mana, yang disediakan dari klien, harus dipetakan ke parameter permintaan integrasi atau properti tubuh yang berlaku sebelum dikirim ke backend. Kemudian, sebagai bagian dari pengaturan permintaan integrasi, Anda mengaturrequestParametersproperti pada yang sesuaiIntegrasisumber daya untuk menentukan parameter-to-parameter pemetaan. Anda juga mengaturrequestTemplatesproperti untuk menentukan template pemetaan, satu untuk setiap jenis konten yang didukung. Pemetaan template peta metode permintaan parameter, atau tubuh, untuk badan permintaan integrasi.

Demikian pula, sebagai bagian dari pengaturan respons metode, Anda mengaturresponseParametersproperti diMethodResponsesumber daya. Ini menyatakan yang parameter metode respon, yang akan dikirim ke klien, yang akan dipetakan dari parameter respon integrasi atau properti tubuh tertentu yang berlaku yang dikembalikan dari backend. Kemudian, sebagai bagian dari pengaturan respons integrasi, Anda mengaturresponseParametersproperti pada yang sesuaiIntegrationResponsesumber daya untuk menentukan parameter-to-parameter pemetaan. Anda juga mengaturResponseTemplatespeta untuk menentukan template pemetaan, satu untuk setiap jenis konten yang didukung. Template pemetaan memetakan parameter respons integrasi, atau sifat tubuh respons integrasi, ke badan respons metode.

Untuk informasi lebih lanjut tentang penyiapan template pemetaan, lihatMenyiapkan transformasi data untuk REST API.

Siapkan integrasi pribadi API Gateway

Integrasi pribadi API Gateway memudahkan Anda mengekspos sumber daya HTTP/HTTPS Anda dalam VPC Amazon untuk diakses oleh klien di luar VPC. Untuk memperluas akses ke sumber daya VPC pribadi Anda di luar batas VPC, Anda dapat membuat API dengan integrasi pribadi. Anda dapat mengontrol akses ke API Anda dengan menggunakan salah satu metode otorisasi yang didukung API Gateway.

Untuk membuat integrasi pribadi, Anda harus terlebih dahulu membuat Network Load Balancer. Network Load Balancer Anda harus memiliki listener yang merutekan permintaan ke sumber daya di VPC Anda. Untuk meningkatkan ketersediaan API Anda, pastikan Network Load Balancer merutekan lalu lintas ke sumber daya di lebih dari satu Availability Zone di. Wilayah AWS Kemudian, Anda membuat tautan VPC yang Anda gunakan untuk menghubungkan API dan Network Load Balancer Anda. Setelah membuat tautan VPC, Anda membuat integrasi pribadi untuk merutekan lalu lintas dari API ke sumber daya di VPC melalui tautan VPC dan Network Load Balancer.

> **Note**
>
> Network Load Balancer dan API harus dimiliki oleh akun yang samaAWS.

Dengan integrasi pribadi API Gateway, Anda dapat mengaktifkan akses ke sumber daya HTTP/HTTPS dalam VPC tanpa pengetahuan rinci tentang konfigurasi jaringan pribadi atau peralatan khusus teknologi.

Topik

- Menyiapkan Network Load Balancer untuk integrasi pribadi API Gateway
- Berikan izin untuk membuat tautan VPC
- Siapkan API Gateway API dengan integrasi pribadi menggunakan konsol API Gateway
- Siapkan API Gateway API dengan integrasi pribadi menggunakan AWS CLI
- Siapkan API dengan integrasi pribadi menggunakan OpenAPI
- Akun API Gateway yang digunakan untuk integrasi pribadi

Menyiapkan Network Load Balancer untuk integrasi pribadi API Gateway

Prosedur berikut menguraikan langkah-langkah untuk menyiapkan Network Load Balancer (NLB) untuk integrasi pribadi API Gateway menggunakan konsol Amazon EC2 dan memberikan referensi untuk petunjuk terperinci untuk setiap langkah.

Untuk setiap VPC tempat Anda memiliki sumber daya, Anda hanya perlu mengkonfigurasi satu NLB dan satu VPCLink. NLB mendukung beberapa pendengar dan grup target per NLB. Anda dapat mengonfigurasi setiap layanan sebagai pendengar tertentu di NLB dan menggunakan satu VPCLink untuk terhubung ke NLB. Saat membuat integrasi pribadi di API Gateway, Anda kemudian menentukan setiap layanan menggunakan port spesifik yang ditetapkan untuk setiap layanan. Untuk informasi selengkapnya, lihat the section called “Tutorial: Membangun API dengan integrasi pribadi”.

> **Note**
>
> Network Load Balancer dan API harus dimiliki oleh akun yang samaAWS.

Untuk membuat Network Load Balancer untuk integrasi pribadi menggunakan konsol API Gateway

1. Masuk ke AWS Management Console dan buka konsol Amazon EC2 di https://console.aws.amazon.com/ec2/.
2. Siapkan server web pada instans Amazon EC2. Untuk contoh penyiapan, lihat Menginstal Server Web LAMP di Amazon Linux 2.
3. Buat Network Load Balancer, daftarkan instans EC2 dengan grup target, dan tambahkan grup target ke pendengar Network Load Balancer. Untuk detailnya, ikuti petunjuk di Memulai dengan Network Load Balancers.
4. Setelah Network Load Balancer dibuat, lakukan hal berikut:

    a. Perhatikan ARN dari Network Load Balancer. Anda akan membutuhkannya untuk membuat tautan VPC di API Gateway untuk mengintegrasikan API dengan sumber daya VPC di belakang Network Load Balancer

    b. Matikan evaluasi grup keamanan untuk PrivateLink. Gunakan perintah berikut untuk mematikan aturan masuk pada PrivateLink lalu lintas.

```
aws elbv2 set-security-groups --load-balancer-
arn arn:aws:elasticloadbalancing:us-east-2:111122223333:loadbalancer/net/my-
loadbalancer/abc12345 \
```

```
  --security-groups sg-123345a --enforce-security-group-inbound-rules-on-
private-link-traffic off
```

> **Note**
>
> Jangan menambahkan dependensi apa pun ke CIDR API Gateway karena mereka pasti akan berubah tanpa pemberitahuan.

Berikan izin untuk membuat tautan VPC

Agar Anda atau pengguna di akun Anda dapat membuat dan memelihara tautan VPC, Anda atau pengguna harus memiliki izin untuk membuat, menghapus, dan melihat konfigurasi layanan titik akhir VPC, mengubah izin layanan titik akhir VPC, dan memeriksa penyeimbang beban. Untuk memberikan izin tersebut, gunakan langkah-langkah berikut.

Untuk memberikan izin untuk membuat, memperbarui, dan menghapus tautan VPC

1. Buat kebijakan IAM yang mirip dengan berikut ini:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "apigateway:POST",
                "apigateway:GET",
                "apigateway:PATCH",
                "apigateway:DELETE"
            ],
            "Resource": [
                "arn:aws:apigateway:us-east-1::/vpclinks",
                "arn:aws:apigateway:us-east-1::/vpclinks/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:DescribeLoadBalancers"
            ],
```

```
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpointServiceConfiguration",
                "ec2:DeleteVpcEndpointServiceConfigurations",
                "ec2:DescribeVpcEndpointServiceConfigurations",
                "ec2:ModifyVpcEndpointServicePermissions"
            ],
            "Resource": "*"
        }
    ]
}
```

2. Buat atau pilih peran IAM dan lampirkan kebijakan sebelumnya ke peran tersebut.
3. Tetapkan peran IAM kepada Anda atau pengguna di akun Anda yang membuat tautan VPC.

Siapkan API Gateway API dengan integrasi pribadi menggunakan konsol API Gateway

Untuk petunjuk menggunakan API Gateway Console guna menyiapkan API dengan integrasi pribadi, lihatTutorial: Membangun REST API dengan integrasi pribadi API Gateway.

Siapkan API Gateway API dengan integrasi pribadi menggunakan AWS CLI

Sebelum membuat API dengan integrasi pribadi, Anda harus menyiapkan sumber daya VPC dan Network Load Balancer dibuat dan dikonfigurasi dengan sumber VPC Anda sebagai target. Jika persyaratan tidak terpenuhi, ikuti Menyiapkan Network Load Balancer untuk integrasi pribadi API Gateway untuk menginstal sumber daya VPC, buat Network Load Balancer, dan atur sumber daya VPC sebagai target Network Load Balancer.

> **Note**
>
> Network Load Balancer dan API harus dimiliki oleh akun yang samaAWS.

Agar Anda dapat membuat dan mengelola a`VpcLink`, Anda juga harus memiliki izin yang sesuai yang dikonfigurasi. Untuk informasi selengkapnya, lihat Berikan izin untuk membuat tautan VPC.

> **Note**
>
> Anda hanya perlu izin untuk membuat `VpcLink` di API Anda. Anda tidak memerlukan izin untuk menggunakan file. `VpcLink`

Setelah Network Load Balancer dibuat, perhatikan ARN-nya. Anda membutuhkannya untuk membuat tautan VPC untuk integrasi pribadi.

Untuk menyiapkan API dengan integrasi pribadi menggunakan AWS CLI

1. Buat `VpcLink` penargetan Network Load Balancer yang ditentukan.

```
aws apigateway create-vpc-link \
    --name my-test-vpc-link \
    --target-arns arn:aws:elasticloadbalancing:us-east-2:123456789012:loadbalancer/
net/my-vpclink-test-nlb/1234567890abcdef
```

   Output dari perintah ini mengakui penerimaan permintaan dan menunjukkan `PENDING` status untuk yang `VpcLink` sedang dibuat.

```
{
    "status": "PENDING",
    "targetArns": [
        "arn:aws:elasticloadbalancing:us-east-2:123456789012:loadbalancer/net/my-
vpclink-test-nlb/1234567890abcdef"
    ],
    "id": "gim7c3",
    "name": "my-test-vpc-link"
}
```

   Dibutuhkan 2-4 menit bagi API Gateway untuk menyelesaikan pembuatan`VpcLink`. Ketika operasi selesai dengan sukses, `status` adalah`AVAILABLE`. Anda dapat memverifikasi ini dengan memanggil perintah CLI berikut:

```
aws apigateway get-vpc-link --vpc-link-id gim7c3
```

   Jika operasi gagal, Anda mendapatkan `FAILED` status, dengan `statusMessage` berisi pesan kesalahan. Misalnya, jika Anda mencoba membuat `VpcLink` dengan Network Load Balancer

yang sudah dikaitkan dengan titik akhir VPC, Anda mendapatkan yang berikut di properti: `statusMessage`

```
"NLB is already associated with another VPC Endpoint Service"
```

Setelah berhasil `VpcLink` dibuat, Anda dapat membuat API dan mengintegrasikannya dengan sumber daya VPC melalui file. `VpcLink`

Perhatikan `id` nilai yang baru dibuat `VpcLink` (*`gim7c3`*dalam output sebelumnya). Anda membutuhkannya untuk mengatur integrasi pribadi.

2. Siapkan API dengan membuat `RestApi`resource API Gateway:

```
aws apigateway create-rest-api --name 'My VPC Link Test'
```

Perhatikan `id` nilai `RestApi` dalam hasil yang dikembalikan. Anda memerlukan nilai ini untuk melakukan operasi lebih lanjut pada API.

Untuk tujuan ilustrasi, kita akan membuat API dengan hanya `GET` metode pada sumber daya root (`/`) dan mengintegrasikan metode dengan. `VpcLink`

3. Siapkan `GET /` metode. Pertama dapatkan pengenal sumber daya root (`/`):

```
aws apigateway get-resources --rest-api-id  abcdef123
```

Dalam output, perhatikan `id` nilai `/` jalur. Dalam contoh ini, kami menganggapnya demikian*`skpp60rab7`*.

Siapkan permintaan metode untuk metode API`GET /`:

```
aws apigateway put-method \
       --rest-api-id  abcdef123 \
       --resource-id skpp60rab7 \
       --http-method GET \
       --authorization-type "NONE"
```

Jika Anda tidak menggunakan integrasi proxy dengan`VpcLink`, Anda juga harus mengatur setidaknya respons metode kode `200` status. Kami akan menggunakan integrasi proxy di sini.

4. Siapkan integrasi pribadi `HTTP_PROXY` tipe dan panggil `put-integration` perintah sebagai berikut:

```
aws apigateway put-integration \
    --rest-api-id abcdef123 \
    --resource-id skpp60rab7 \
    --uri 'http://my-vpclink-test-nlb-1234567890abcdef.us-east-2.amazonaws.com' \
    --http-method GET \
    --type HTTP_PROXY \
    --integration-http-method GET \
    --connection-type VPC_LINK \
    --connection-id gim7c3
```

Untuk integrasi pribadi, atur `connection-type` ke `VPC_LINK` dan setel `connection-id` ke pengenal Anda `VpcLink` atau variabel tahap yang mereferensikan ID Anda`VpcLink`. `uri`Parameter ini tidak digunakan untuk merutekan permintaan ke titik akhir Anda, tetapi digunakan untuk mengatur `Host` header dan untuk validasi sertifikat.

Perintah mengembalikan output berikut:

```
{
    "passthroughBehavior": "WHEN_NO_MATCH",
    "timeoutInMillis": 29000,
    "connectionId": "gim7c3",
    "uri": "http://my-vpclink-test-nlb-1234567890abcdef.us-east-2.amazonaws.com",
    "connectionType": "VPC_LINK",
    "httpMethod": "GET",
    "cacheNamespace": "skpp60rab7",
    "type": "HTTP_PROXY",
    "cacheKeyParameters": []
}
```

Menggunakan variabel tahap, Anda mengatur `connectionId` properti saat membuat integrasi:

```
aws apigateway put-integration \
    --rest-api-id abcdef123 \
    --resource-id skpp60rab7 \
    --uri 'http://my-vpclink-test-nlb-1234567890abcdef.us-east-2.amazonaws.com' \
    --http-method GET \
    --type HTTP_PROXY \
    --integration-http-method GET \
    --connection-type VPC_LINK \
    --connection-id "\${stageVariables.vpcLinkId}"
```

Pastikan untuk mengutip dua kali ekspresi variabel tahap (`${stageVariables.vpcLinkId}`) dan melarikan diri dari $ karakter.

Atau, Anda dapat memperbarui integrasi untuk mengatur ulang `connectionId` nilai dengan variabel tahap:

```
 aws apigateway update-integration \
    --rest-api-id abcdef123 \
    --resource-id skpp60rab7 \
    --http-method GET \
    --patch-operations '[{"op":"replace","path":"/
connectionId","value":"${stageVariables.vpcLinkId}"}]'
```

Pastikan untuk menggunakan daftar JSON stringified sebagai nilai parameter. `patch-operations`

Anda dapat menggunakan variabel stage untuk mengintegrasikan API Anda dengan VPC atau Network Load Balancer yang berbeda dengan mengatur ulang `VpcLink` nilai variabel tahap s.

Karena kami menggunakan integrasi proxy pribadi, API sekarang siap untuk penerapan dan untuk uji coba. Dengan integrasi non-proxy, Anda juga harus mengatur respons metode dan respons integrasi, seperti yang Anda lakukan saat menyiapkan API dengan integrasi kustom HTTP.

5. Untuk menguji API, terapkan API. Hal ini diperlukan jika Anda telah menggunakan variabel stage sebagai placeholder dari ID. `VpcLink` Untuk menerapkan API dengan variabel stage, panggil `create-deployment` perintah sebagai berikut:

```
aws apigateway create-deployment \
    --rest-api-id abcdef123 \
    --stage-name test \
    --variables vpcLinkId=gim7c3
```

Untuk memperbarui variabel stage dengan `VpcLink` ID yang berbeda (misalnya,*asf9d7*), panggil `update-stage` perintah:

```
aws apigateway update-stage \
    --rest-api-id abcdef123 \
    --stage-name test \
```

```
    --patch-operations op=replace,path='/variables/vpcLinkId',value='asf9d7'
```

Gunakan perintah berikut untuk menjalankan API Anda:

```
curl -X GET https://abcdef123.execute-api.us-east-2.amazonaws.com/test
```

Atau, Anda dapat mengetikkan URL Invoke-API di browser web untuk melihat hasilnya.

Saat Anda membuat hardcode `connection-id` properti dengan `VpcLink` ID literal, Anda juga dapat memanggil `test-invoke-method` untuk menguji pemanggilan API sebelum diterapkan.

Siapkan API dengan integrasi pribadi menggunakan OpenAPI

Anda dapat menyiapkan API dengan integrasi pribadi dengan mengimpor file OpenAPI API. Pengaturannya mirip dengan definisi OpenAPI API dengan integrasi HTTP, dengan pengecualian berikut:

- Anda harus secara eksplisit mengatur `connectionType` ke. `VPC_LINK`
- Anda harus secara eksplisit mengatur `connectionId` ke ID dari `VpcLink` atau ke variabel tahap yang merujuk ID dari sebuah. `VpcLink`
- `uri`Parameter dalam integrasi pribadi menunjuk ke titik akhir HTTP/HTTPS di VPC, tetapi digunakan sebagai gantinya untuk mengatur header permintaan integrasi. `Host`
- `uri`Parameter dalam integrasi pribadi dengan titik akhir HTTPS di VPC digunakan untuk memverifikasi nama domain yang dinyatakan terhadap yang ada di sertifikat yang diinstal pada titik akhir VPC.

Anda dapat menggunakan variabel tahap untuk mereferensikan `VpcLink` ID. Atau Anda dapat menetapkan nilai ID langsung ke`connectionId`.

File OpenAPI berformat JSON berikut menunjukkan contoh API dengan tautan VPC seperti yang direferensikan oleh variabel tahap (): `${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": {
              "$ref": "#/definitions/Empty"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "uri": "http://my-vpclink-test-nlb-1234567890abcdef.us-
east-2.amazonaws.com",
          "passthroughBehavior": "when_no_match",
          "connectionType": "VPC_LINK",
          "connectionId": "${stageVariables.vpcLinkId}",
          "httpMethod": "GET",
          "type": "http_proxy"
        }
      }
    }
  },
  "definitions": {
    "Empty": {
      "type": "object",
      "title": "Empty Schema"
    }
  }
```

```
}
```

Akun API Gateway yang digunakan untuk integrasi pribadi

ID akun API Gateway khusus wilayah berikut secara otomatis ditambahkan ke layanan titik akhir VPC Anda seperti `AllowedPrincipals` saat Anda membuat file. `VpcLink`

Wilayah	ID Akun
us-east-1	392220576650
us–east-2	718770453195
us-west-1	968246515281
us-west-2	109351309407
ca-central-1	796887884028
eu-west-1	631144002099
eu-west-2	544388816663
eu-west-3	061510835048
eu-central-1	474240146802
eu-central-2	166639821150
eu-north-1	394634713161
eu-south-1	753362059629
eu-south-2	359345898052
ap-northeast-1	969236854626
ap-northeast-2	020402002396
ap-northeast-3	360671645888

Wilayah	ID Akun
ap-southeast-1	195145609632
ap-southeast-2	798376113853
ap-southeast-3	652364314486
ap-southeast-4	849137399833
ap-south-1	507069717855
ap-south-2	644042651268
ap-east-1	174803364771
sa-east-1	287228555773
me-south-1	855739686837
me-central-1	614065512851

Siapkan integrasi tiruan di API Gateway

Amazon API Gateway mendukung integrasi tiruan untuk metode API. Fitur ini memungkinkan pengembang API untuk menghasilkan respons API dari API Gateway secara langsung, tanpa perlu backend integrasi. Sebagai pengembang API, Anda dapat menggunakan fitur ini untuk membuka blokir tim dependen yang perlu bekerja dengan API sebelum pengembangan proyek selesai. Anda juga dapat menggunakan fitur ini untuk menyediakan halaman landing untuk API Anda, yang dapat memberikan ikhtisar dan navigasi ke API Anda. Untuk contoh halaman arahan seperti itu, lihat permintaan integrasi dan respons metode GET pada sumber daya root dari contoh API yang dibahasTutorial: Buat REST API dengan mengimpor contoh.

Sebagai pengembang API, Anda memutuskan bagaimana API Gateway merespons permintaan integrasi tiruan. Untuk ini, Anda mengonfigurasi permintaan integrasi metode dan respons integrasi untuk mengaitkan respons dengan kode status tertentu. Untuk metode dengan integrasi tiruan untuk mengembalikan `200` respons, konfigurasikan template pemetaan badan permintaan integrasi untuk mengembalikan yang berikut ini.

```
{"statusCode": 200}
```

Konfigurasikan respons `200` integrasi untuk memiliki template pemetaan tubuh berikut, misalnya:

```
{
    "statusCode": 200,
    "message": "Go ahead without me."
}
```

Demikian pula, agar metode mengembalikan, misalnya, respons `500` kesalahan, siapkan templat pemetaan badan permintaan integrasi untuk mengembalikan yang berikut ini.

```
{"statusCode": 500}
```

Siapkan respons `500` integrasi dengan, misalnya, templat pemetaan berikut:

```
{
    "statusCode": 500,
    "message": "The invoked method is not supported on the API resource."
}
```

Atau, Anda dapat meminta metode integrasi tiruan mengembalikan respons integrasi default tanpa menentukan templat pemetaan permintaan integrasi. Respons integrasi default adalah respons dengan regex status HTTP yang tidak ditentukan. Pastikan perilaku passthrough yang sesuai ditetapkan.

> **Note**
>
> Integrasi tiruan tidak dimaksudkan untuk mendukung template respons besar. Jika Anda membutuhkannya untuk kasus penggunaan Anda, Anda harus mempertimbangkan untuk menggunakan integrasi Lambda sebagai gantinya.

Dengan menggunakan templat pemetaan permintaan integrasi, Anda dapat menyuntikkan logika aplikasi untuk memutuskan respons integrasi tiruan mana yang akan dikembalikan berdasarkan kondisi tertentu. Misalnya, Anda dapat menggunakan parameter `scope` kueri pada permintaan yang masuk untuk menentukan apakah akan mengembalikan respons yang berhasil atau respons kesalahan:

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

Dengan cara ini, metode integrasi tiruan memungkinkan panggilan internal dilakukan sambil menolak jenis panggilan lain dengan respons kesalahan.

Di bagian ini, kami menjelaskan cara menggunakan konsol API Gateway untuk mengaktifkan integrasi tiruan untuk metode API.

Topik

- Aktifkan integrasi tiruan menggunakan konsol API Gateway

Aktifkan integrasi tiruan menggunakan konsol API Gateway

Anda harus memiliki metode yang tersedia di API Gateway. Ikuti instruksi di Tutorial: Membangun REST API dengan integrasi non-proxy HTTP.

1. Pilih sumber daya API dan pilih metode Buat.

   Untuk membuat metode, lakukan hal berikut:

   a. Untuk jenis Metode, pilih metode.
   b. Untuk jenis Integrasi, pilih Mock.
   c. Pilih metode Buat.
   d. Pada tab Permintaan metode, untuk pengaturan permintaan Metode, pilih Edit.
   e. Pilih parameter string kueri URL. Pilih Tambahkan string kueri dan untuk Nama, masukkan**`scope`**. Parameter kueri ini menentukan apakah pemanggil internal atau sebaliknya.
   f. Pilih Save (Simpan).
2. Pada tab respons Metode, pilih Buat respons, lalu lakukan hal berikut:

   a. Untuk Status HTTP, masukkan**`500`**.

b. Pilih Save (Simpan).

3. Pada tab Permintaan integrasi, untuk pengaturan permintaan Integrasi, pilih Edit.
4. Pilih template Pemetaan, lalu lakukan hal berikut:

   a. Pilih Tambahkan templat pemetaan.

   b. Untuk jenis Konten, masukkan**`application/json`**.

   c. Untuk badan Template, masukkan yang berikut ini:

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

   d. Pilih Save (Simpan).

5. Pada tab Respons Integrasi, untuk Default - Respons pilih Edit.
6. Pilih template Pemetaan, lalu lakukan hal berikut:

   a. Untuk jenis Konten, masukkan**`application/json`**.

   b. Untuk badan Template, masukkan yang berikut ini:

```
{
    "statusCode": 200,
    "message": "Go ahead without me"
}
```

   c. Pilih Save (Simpan).

7. Pilih Buat respons.

   Untuk membuat respons 500, lakukan hal berikut:

   a. Untuk regex status HTTP, masukkan. **`5\d{2}`**

   b. Untuk status respons Metode, pilih**`500`**.

   c. Pilih Save (Simpan).

   d. Untuk 5\ d {2} - Respons, pilih Edit.

   e. Pilih Templat pemetaan, lalu pilih Tambahkan templat pemetaan.

f. Untuk jenis Konten, masukkan**`application/json`**.

g. Untuk badan Template, masukkan yang berikut ini:

```
{
    "statusCode": 500,
    "message": "The invoked method is not supported on the API resource."
}
```

h. Pilih Save (Simpan).

8. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab. Untuk menguji integrasi tiruan Anda, lakukan hal berikut:

a. Masukkan `scope=internal` di bawah String kueri. Pilih Uji. Hasil tes menunjukkan:

```
Request: /?scope=internal
Status: 200
Latency: 26 ms
Response Body

{
  "statusCode": 200,
  "message": "Go ahead without me"
}

Response Headers

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

b. Masukkan `scope=public` di bawah `Query strings` atau biarkan kosong. Pilih Uji. Hasil tes menunjukkan:

```
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"}
```

Anda juga dapat mengembalikan header dalam respons integrasi tiruan dengan terlebih dahulu menambahkan header ke respons metode dan kemudian menyiapkan pemetaan header dalam respons integrasi. Faktanya, beginilah cara konsol API Gateway mengaktifkan dukungan CORS dengan mengembalikan header yang diperlukan CORS.

Gunakan validasi permintaan di API Gateway

Anda dapat mengonfigurasi API Gateway untuk melakukan validasi dasar permintaan API sebelum melanjutkan dengan permintaan integrasi. Ketika validasi gagal, API Gateway segera gagal permintaan, mengembalikan respons kesalahan 400 ke pemanggil, dan menerbitkan hasil validasi di Log. CloudWatch Ini mengurangi panggilan yang tidak perlu ke backend. Lebih penting lagi, ini memungkinkan Anda fokus pada upaya validasi khusus untuk aplikasi Anda. Anda dapat memvalidasi badan permintaan dengan memverifikasi bahwa parameter permintaan yang diperlukan valid dan non-null atau dengan menentukan skema model untuk validasi data yang lebih rumit.

Topik

- Ikhtisar validasi permintaan dasar di API Gateway
- Memahami model data
- Siapkan validasi permintaan dasar di API Gateway
- Definisi OpenAPI dari API sampel dengan validasi permintaan dasar
- AWS CloudFormation template API sampel dengan validasi permintaan dasar

Ikhtisar validasi permintaan dasar di API Gateway

API Gateway dapat melakukan validasi permintaan dasar, sehingga Anda dapat fokus pada validasi khusus aplikasi di backend. Untuk validasi, API Gateway memverifikasi salah satu atau kedua kondisi berikut:

- Parameter permintaan yang diperlukan dalam URI, string kueri, dan header permintaan masuk disertakan dan tidak kosong.
- Payload permintaan yang berlaku mematuhi permintaan skema JSON yang dikonfigurasi dari metode ini.

Untuk mengaktifkan validasi, Anda menentukan aturan validasi dalam validator permintaan, menambahkan validator ke peta API validator permintaan, dan menetapkan validator ke metode API individual.

> **Note**
>
> Minta validasi badan dan Perilaku passthrough integrasi merupakan dua topik terpisah. Jika payload permintaan tidak memiliki skema model yang cocok, Anda dapat memilih untuk melewati atau memblokir muatan asli. Untuk informasi selengkapnya, lihat Perilaku passthrough integrasi.

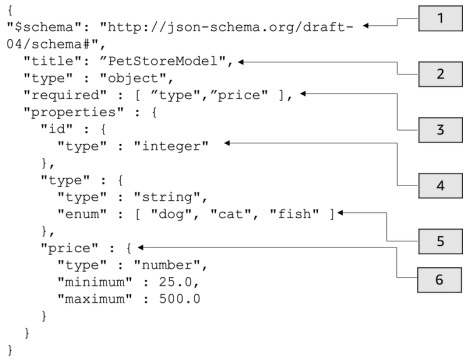
Memahami model data

Di API Gateway, model mendefinisikan struktur data payload. Di API Gateway, model didefinisikan menggunakan draf skema JSON 4. Objek JSON berikut adalah data sampel dalam contoh Pet Store.

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

Data berisi`id`,`type`, dan `price` hewan peliharaan. Model data ini memungkinkan Anda untuk:

- Gunakan validasi permintaan dasar.
- Buat template pemetaan untuk transformasi data.
- Buat tipe data yang ditentukan pengguna (UDT) saat Anda membuat SDK.



Dalam model ini:

1. `$schema`Objek mewakili pengidentifikasi versi Skema JSON yang valid. Skema ini adalah draf Skema JSON v4.
2. `title`Objek adalah pengidentifikasi yang dapat dibaca manusia untuk model tersebut. Judul ini adalah`PetStoreModel`.
3. Kata kunci `required` validasi membutuhkan`type`, dan `price` untuk validasi permintaan dasar.
4. Model `properties` tersebut adalah`id`,`type`, dan`price`. Setiap objek memiliki properti yang dijelaskan dalam model.
5. Objek hanya `type` dapat memiliki nilai`dog`,`cat`, atau`fish`.
6. Objek `price` adalah angka dan dibatasi dengan `minimum` 25 dan 500`maximum`.

PetStore model

```
1 {
2 "$schema": "http://json-schema.org/draft-04/schema#",
3  "title": "PetStoreModel",
4  "type" : "object",
5  "required" : [ "price", "type" ],
6  "properties" : {
7    "id" : {
8      "type" : "integer"
9    },
10    "type" : {
11      "type" : "string",
12      "enum" : [ "dog", "cat", "fish" ]
13    },
14    "price" : {
15      "type" : "number",
```

```
16      "minimum" : 25.0,
17      "maximum" : 500.0
18    }
19  }
20 }
```

Dalam model ini:

1. Pada baris 2, `$schema` objek mewakili pengenal versi Skema JSON yang valid. Skema ini adalah draf Skema JSON v4.
2. Pada baris 3, `title` objek adalah pengidentifikasi yang dapat dibaca manusia untuk model tersebut. Judul ini adalah`PetStoreModel`.
3. Pada baris 5, kata kunci `required` validasi membutuhkan`type`, dan `price` untuk validasi permintaan dasar.
4. Pada baris 6 - 17, modelnya adalah`id`,`type`, dan`price`. `properties` Setiap objek memiliki properti yang dijelaskan dalam model.
5. Pada baris 12, objek hanya `type` dapat memiliki nilai`dog`,`cat`, atau`fish`.
6. Pada baris 14 - 17, objek `price` adalah angka dan dibatasi dengan `minimum` 25 dan 500`maximum`.

Membuat model yang lebih kompleks

Anda dapat menggunakan `$ref` primitif untuk membuat definisi yang dapat digunakan kembali untuk model yang lebih panjang. Misalnya, Anda dapat membuat definisi yang disebut `Price` di `definitions` bagian yang menjelaskan `price` objek. Nilai `$ref` adalah `Price` definisi.

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "PetStoreModelReUsableRef",
  "required" : ["price", "type" ],
  "type" : "object",
  "properties" : {
    "id" : {
      "type" : "integer"
    },
    "type" : {
      "type" : "string",
      "enum" : [ "dog", "cat", "fish" ]
    },
```

```
    "price" : {
        "$ref": "#/definitions/Price"
    }
  },
  "definitions" : {
      "Price": {
        "type" : "number",
        "minimum" : 25.0,
        "maximum" : 500.0
            }
      }
}
```

Anda juga dapat mereferensikan skema model lain yang ditentukan dalam file model eksternal. Tetapkan nilai `$ref` properti ke lokasi model. Dalam contoh berikut, `Price` model didefinisikan dalam `PetStorePrice` model di API`a1234`.

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "PetStorePrice",
  "type": "number",
  "minimum": 25,
  "maximum": 500
}
```

Model yang lebih panjang dapat merujuk `PetStorePrice` model.

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "PetStoreModelReusableRefAPI",
  "required" : [ "price", "type" ],
  "type" : "object",
  "properties" : {
    "id" : {
      "type" : "integer"
    },
    "type" : {
      "type" : "string",
      "enum" : [ "dog", "cat", "fish" ]
    },
    "price" : {
        "$ref": "https://apigateway.amazonaws.com/restapis/a1234/models/PetStorePrice"
```

```
    }
  }
}
```

Menggunakan model data keluaran

Jika Anda mengubah data, Anda dapat menentukan model payload dalam respons integrasi. Model payload dapat digunakan saat Anda membuat SDK. Untuk bahasa yang diketik dengan kuat, seperti Java, Objective-C, atau Swift, objek sesuai dengan tipe data yang ditentukan pengguna (UDT). API Gateway membuat UDT jika Anda menyediakannya dengan model data saat Anda membuat SDK. Untuk informasi selengkapnya tentang transformasi data, lihatMemahami template pemetaan.

Data keluaran

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

Model keluaran

```
{
"$schema": "http://json-schema.org/
draft-04/schema#",
  "title": ”PetStoreOutputModel",
  "type" : "object",
  "required" : [ "description",
 "askingPrice" ],
  "properties" : {
    "description" : {
```

```
      "type" : "string"
    },
    "askingPrice" : {
      "type" : "number",
      "minimum" : 25.0,
      "maximum" : 500.0
    }
  }
}
```

Dengan model ini, Anda dapat memanggil SDK untuk mengambil nilai `description` dan `askingPrice` properti dengan membaca properti `PetStoreOutputModel[i].description` dan`PetStoreOutputModel[i].askingPrice`. Jika tidak ada model yang disediakan, API Gateway menggunakan model kosong untuk membuat UDT default.

Langkah selanjutnya

- Bagian ini menyediakan sumber daya yang dapat Anda gunakan untuk mendapatkan lebih banyak pengetahuan tentang konsep yang disajikan dalam topik ini.

  Anda dapat mengikuti tutorial validasi permintaan:

  - Siapkan validasi permintaan menggunakan konsol API Gateway
  - Siapkan validasi permintaan dasar menggunakan AWS CLI
  - Siapkan validasi permintaan dasar menggunakan definisi OpenAPI
- Anda bisa mendapatkan informasi lebih lanjut tentang transformasi data dan template pemetaan,Memahami template pemetaan.
- Anda juga dapat melihat model contoh album foto yang lebih canggih. Lihat Contoh foto.

Siapkan validasi permintaan dasar di API Gateway

Bagian ini menunjukkan cara menyiapkan validasi permintaan untuk API Gateway menggunakan konsol AWS CLI, dan definisi OpenAPI.

Topik

- Siapkan validasi permintaan menggunakan konsol API Gateway
- Siapkan validasi permintaan dasar menggunakan AWS CLI
- Siapkan validasi permintaan dasar menggunakan definisi OpenAPI

Siapkan validasi permintaan menggunakan konsol API Gateway

Anda dapat menggunakan konsol API Gateway untuk memvalidasi permintaan dengan memilih salah satu dari tiga validator untuk permintaan API:

- Validasi tubuh.
- Validasi parameter string kueri dan header.
- Validasi isi, parameter string kueri, dan header.

Saat Anda menerapkan salah satu validator pada metode API, konsol API Gateway menambahkan validator ke peta API. RequestValidators

Untuk mengikuti tutorial ini, Anda akan menggunakan AWS CloudFormation template untuk membuat API Gateway API yang tidak lengkap. API ini memiliki `/validator` sumber daya dengan `GET` dan `POST` metode. Kedua metode terintegrasi dengan titik akhir `http://petstore-demo-endpoint.execute-api.com/petstore/pets` HTTP. Anda akan mengkonfigurasi dua jenis validasi permintaan:

- Dalam `GET` metode ini, Anda akan mengkonfigurasi validasi permintaan untuk parameter string kueri URL.
- Dalam `POST` metode ini, Anda akan mengonfigurasi validasi permintaan untuk badan permintaan.

Ini akan memungkinkan hanya panggilan API tertentu untuk melewati API.

Unduh dan unzip template pembuatan aplikasi untuk AWS CloudFormation. Anda akan menggunakan template ini untuk membuat API yang tidak lengkap. Anda akan menyelesaikan langkah-langkah lainnya di konsol API Gateway.

Untuk membuat AWS CloudFormation tumpukan

1. Buka AWS CloudFormation konsol di https://console.aws.amazon.com/cloudformation.
2. Pilih Buat tumpukan kemudian pilih Dengan sumber daya baru (standar).
3. Untuk Tentukan templat, pilih Unggah file templat.
4. Pilih template yang Anda unduh.
5. Pilih Berikutnya.
6. Untuk nama Stack, masukkan **`request-validation-tutorial-console`** dan kemudian pilih Berikutnya.

7. Untuk opsi Konfigurasi tumpukan, pilih Berikutnya.
8. Untuk Kemampuan, akui bahwa AWS CloudFormation dapat membuat sumber daya IAM di akun Anda.
9. Pilih Buat tumpukan.

AWS CloudFormation ketentuan sumber daya yang ditentukan dalam template. Diperlukan beberapa menit untuk menyelesaikan penyediaan sumber daya Anda. Ketika status AWS CloudFormation tumpukan Anda adalah CREATE_COMPLETE, Anda siap untuk melanjutkan ke langkah berikutnya.

Untuk memilih API yang baru dibuat

1. Pilih **`request-validation-tutorial-console`** tumpukan yang baru dibuat.
2. Pilih Sumber daya.
3. Di bawah Physical ID, pilih API Anda. Tautan ini akan mengarahkan Anda ke konsol API Gateway.

Sebelum Anda memodifikasi `GET` dan `POST` metode, Anda harus membuat model.

Untuk membuat model

1. Model diperlukan untuk menggunakan validasi permintaan pada badan permintaan yang masuk. Untuk membuat model, di panel navigasi utama, pilih Model.
2. Pilih Buat model.
3. Untuk Nama, masukkan **`PetStoreModel`**.
4. Untuk Jenis Konten, masukkan**`application/json`**. Jika tidak ada jenis konten yang cocok ditemukan, validasi permintaan tidak dilakukan. Untuk menggunakan model yang sama terlepas dari jenis konten, masukkan**`$default`**.
5. Untuk Deskripsi, masukkan **`My PetStore Model`** sebagai deskripsi model.
6. Untuk skema Model, tempelkan model berikut ke editor kode, dan pilih Buat.

```
{
  "type" : "object",
  "required" : [ "name", "price", "type" ],
  "properties" : {
    "id" : {
      "type" : "integer"
```

```
    },
    "type" : {
      "type" : "string",
      "enum" : [ "dog", "cat", "fish" ]
    },
    "name" : {
      "type" : "string"
    },
    "price" : {
      "type" : "number",
      "minimum" : 25.0,
      "maximum" : 500.0
    }
  }
}
```

Untuk informasi lebih lanjut tentang model, lihatMemahami model data.

Untuk mengkonfigurasi validasi permintaan untuk metode **GET**

1. Di panel navigasi utama, pilih Resources, lalu pilih metode GET.
2. Pada tab Permintaan metode, di bawah Pengaturan permintaan metode, pilih Edit.
3. Untuk validator Permintaan, pilih Validasi parameter string kueri dan header.
4. Di bawah parameter string kueri URL, lakukan hal berikut:
   a. Pilih Tambahkan string kueri.
   b. Untuk Nama, masukkan **petType**.
   c. Aktifkan Diperlukan.
   d. Tetap caching dimatikan.
5. Pilih Simpan.
6. Pada tab Permintaan integrasi, di bawah Pengaturan permintaan integrasi, pilih Edit.
7. Di bawah parameter string kueri URL, lakukan hal berikut:
   a. Pilih Tambahkan string kueri.
   b. Untuk Nama, masukkan **petType**.
   c. Untuk Dipetakan dari, masukkan**method.request.querystring.petType**. Ini memetakan **petType** ke jenis hewan peliharaan.

Untuk informasi selengkapnya tentang pemetaan data, lihat tutorial pemetaan data.

d. Tetap caching dimatikan.

8. Pilih Simpan.

Untuk menguji validasi permintaan untuk metode ini **GET**

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Untuk string Kueri, masukkan**petType=dog**, lalu pilih Uji.
3. Tes metode akan kembali `200 OK` dan daftar anjing-anjingnya.

   Untuk informasi tentang cara mengubah data keluaran ini, lihat tutorial pemetaan data.
4. Hapus **petType=dog** dan pilih Uji.
5. Tes metode akan mengembalikan `400` kesalahan dengan pesan kesalahan berikut:

```
{
  "message": "Missing required request parameters: [petType]"
}
```

Untuk mengkonfigurasi validasi permintaan untuk metode **POST**

1. Di panel navigasi utama, pilih Resources, lalu pilih metode POST.
2. Pada tab Permintaan metode, di bawah Pengaturan permintaan metode, pilih Edit.
3. Untuk validator Permintaan, pilih Validasi isi.
4. Di bawah badan Permintaan, pilih Tambah model.
5. Untuk jenis Konten, masukkan**application/json**, dan kemudian untuk Model, pilih PetStoreModel.
6. Pilih Simpan.

Untuk menguji validasi permintaan untuk suatu metode **POST**

1. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
2. Untuk Request body paste berikut ini ke editor kode:

```
{
```

```
  "id": 2,
  "name": "Bella",
  "type": "dog",
  "price": 400
}
```

Pilih Uji.

3. Tes metode akan kembali `200 OK` dan pesan sukses.
4. Untuk Request body paste berikut ini ke editor kode:

```
{
  "id": 2,
  "name": "Bella",
  "type": "dog",
  "price": 4000
}
```

Pilih Uji.

5. Tes metode akan mengembalikan `400` kesalahan dengan pesan kesalahan berikut:

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

Di bagian bawah log pengujian, alasan untuk badan permintaan yang tidak valid dikembalikan. Dalam hal ini, harga hewan peliharaan berada di luar maksimum yang ditentukan dalam model.

Untuk menghapus AWS CloudFormation tumpukan

1. Buka AWS CloudFormation konsol di https://console.aws.amazon.com/cloudformation.
2. Pilih AWS CloudFormation tumpukan Anda.
3. Pilih Hapus dan kemudian konfirmasikan pilihan Anda.

Langkah selanjutnya

- Untuk informasi tentang cara mengubah data keluaran dan melakukan lebih banyak pemetaan data, lihat tutorial pemetaan data.

- Ikuti Mengatur validasi permintaan dasar menggunakan AWS CLI tutorial, untuk melakukan langkah-langkah serupa menggunakan. AWS CLI

Siapkan validasi permintaan dasar menggunakan AWS CLI

Anda dapat membuat validator untuk menyiapkan validasi permintaan menggunakan. AWS CLI Untuk mengikuti tutorial ini, Anda akan menggunakan AWS CloudFormation template untuk membuat API Gateway API yang tidak lengkap.

> **Note**
>
> Ini bukan AWS CloudFormation template yang sama dengan tutorial konsol.

Menggunakan `/validator` sumber daya pra-ekspos, Anda akan membuat `GET` dan `POST` metode. Kedua metode akan terintegrasi dengan titik akhir `http://petstore-demo-endpoint.execute-api.com/petstore/pets` HTTP. Anda akan mengkonfigurasi dua validasi permintaan berikut:

- Pada `GET` metode ini, Anda akan membuat `params-only` validator untuk memvalidasi parameter string kueri URL.
- Pada `POST` metode ini, Anda akan membuat `body-only` validator untuk memvalidasi badan permintaan.

Ini akan memungkinkan hanya panggilan API tertentu untuk melewati API.

Untuk membuat AWS CloudFormation tumpukan

Unduh dan unzip template pembuatan aplikasi untuk AWS CloudFormation.

Untuk menyelesaikan tutorial berikut, Anda memerlukan AWS Command Line Interface (AWS CLI) versi 2.

Untuk perintah panjang, karakter escape (\) digunakan untuk memisahkan perintah menjadi beberapa baris.

> **Note**
>
> Di Windows, beberapa perintah Bash CLI yang biasa Anda gunakan (`zip`seperti) tidak didukung oleh terminal bawaan sistem operasi. Untuk mendapatkan versi terintegrasi Windows dari Ubuntu dan Bash, instal Windows Subsystem untuk Linux. Contoh perintah CLI dalam panduan ini menggunakan pemformatan Linux. Perintah yang menyertakan dokumen JSON sebaris harus diformat ulang jika Anda menggunakan CLI Windows.

1. Gunakan perintah berikut untuk membuat AWS CloudFormation tumpukan.

```
aws cloudformation create-stack --stack-name request-validation-tutorial-cli
 --template-body file://request-validation-tutorial-cli.zip --capabilities
 CAPABILITY_NAMED_IAM
```

2. AWS CloudFormation ketentuan sumber daya yang ditentukan dalam template. Diperlukan beberapa menit untuk menyelesaikan penyediaan sumber daya Anda. Gunakan perintah berikut untuk melihat status AWS CloudFormation tumpukan Anda.

```
aws cloudformation describe-stacks --stack-name request-validation-tutorial-cli
```

3. Ketika status AWS CloudFormation tumpukan Anda`StackStatus: "CREATE_COMPLETE"`, gunakan perintah berikut untuk mengambil nilai output yang relevan untuk langkah-langkah masa depan.

```
aws cloudformation describe-stacks --stack-name request-validation-tutorial-cli
--query "Stacks[*].Outputs[*].{OutputKey: OutputKey, OutputValue: OutputValue,
Description: Description}"
```

   Nilai output adalah sebagai berikut:

   - ApiId, yang merupakan ID untuk API. Untuk tutorial ini, ID API adalah`abc123`.
   - ResourceId, yang merupakan ID untuk sumber daya validator tempat `GET` dan `POST` metode diekspos. Untuk tutorial ini, Resource ID adalah `efg456`

Untuk membuat validator permintaan dan mengimpor model

1. Validator diperlukan untuk menggunakan validasi permintaan dengan file. AWS CLI Gunakan perintah berikut untuk membuat validator yang hanya memvalidasi parameter permintaan.

```
aws apigateway create-request-validator --rest-api-id abc123 \
      --no-validate-request-body \
      --validate-request-parameters \
      --name params-only
```

   Perhatikan ID `params-only` validator.

2. Gunakan perintah berikut untuk membuat validator yang hanya memvalidasi badan permintaan.

```
aws apigateway create-request-validator --rest-api-id abc123 \
      --validate-request-body \
      --no-validate-request-parameters \
      --name body-only
```

   Perhatikan ID `body-only` validator.

3. Model diperlukan untuk menggunakan validasi permintaan pada badan permintaan yang masuk. Gunakan perintah berikut untuk mengimpor model.

```
aws apigateway create-model --rest-api-id abc123 --name PetStoreModel --description
 'My PetStore Model' --content-type 'application/json' --schema '{"type":
 "object", "required" : [ "name", "price", "type" ], "properties" : { "id" :
 {"type" : "integer"},"type" : {"type" : "string", "enum" : [ "dog", "cat",
 "fish" ]},"name" : { "type" : "string"},"price" : {"type" : "number","minimum" :
 25.0, "maximum" : 500.0}}}}'
```

   Jika tidak ada jenis konten yang cocok ditemukan, validasi permintaan tidak dilakukan. Untuk menggunakan model yang sama terlepas dari jenis konten, tentukan `$default` sebagai kunci.

Untuk membuat **`GET`** dan **`POST`** metode

1. Gunakan perintah berikut untuk menambahkan metode `GET` HTTP pada `/validate` sumber daya. Perintah ini menciptakan `GET` metode, menambahkan `params-only` validator, dan menetapkan string query `petType` sesuai kebutuhan.

```
aws apigateway put-method --rest-api-id abc123 \
       --resource-id efg456 \
       --http-method GET \
       --authorization-type "NONE" \
       --request-validator-id aaa111 \
       --request-parameters "method.request.querystring.petType=true"
```

Gunakan perintah berikut untuk menambahkan metode POST HTTP pada `/validate` sumber daya. Perintah ini membuat POST metode, menambahkan `body-only` validator, dan melampirkan model ke validator body-only.

```
aws apigateway put-method --rest-api-id abc123 \
       --resource-id efg456 \
       --http-method POST \
       --authorization-type "NONE" \
       --request-validator-id bbb222 \
       --request-models 'application/json'=PetStoreModel
```

2. Gunakan perintah berikut untuk mengatur `200 OK` respons `GET /validate` metode.

```
aws apigateway put-method-response --rest-api-id abc123  \
            --resource-id efg456 \
            --http-method GET \
            --status-code 200
```

Gunakan perintah berikut untuk mengatur `200 OK` respons `POST /validate` metode.

```
aws apigateway put-method-response --rest-api-id abc123  \
            --resource-id efg456 \
            --http-method POST \
            --status-code 200
```

3. Gunakan perintah berikut untuk mengatur `Integration` dengan titik akhir HTTP tertentu untuk `GET /validation` metode ini.

```
aws apigateway put-integration --rest-api-id abc123  \
            --resource-id efg456 \
            --http-method GET \
            --type HTTP \
            --integration-http-method GET \
```

```
            --request-parameters '{"integration.request.querystring.type" :
 "method.request.querystring.petType"}' \
            --uri 'http://petstore-demo-endpoint.execute-api.com/petstore/pets'
```

Gunakan perintah berikut untuk mengatur `Integration` dengan titik akhir HTTP tertentu untuk `POST /validation` metode ini.

```
aws apigateway put-integration --rest-api-id abc123  \
              --resource-id efg456 \
              --http-method POST \
              --type HTTP \
              --integration-http-method GET \
              --uri 'http://petstore-demo-endpoint.execute-api.com/petstore/pets'
```

4. Gunakan perintah berikut untuk menyiapkan respons integrasi untuk `GET /validation` metode ini.

```
aws apigateway put-integration-response --rest-api-id abc123 \
              --resource-id efg456\
              --http-method GET \
              --status-code 200 \
              --selection-pattern ""
```

Gunakan perintah berikut untuk menyiapkan respons integrasi untuk `POST /validation` metode ini.

```
aws apigateway put-integration-response --rest-api-id abc123 \
            --resource-id efg456 \
            --http-method POST \
            --status-code 200 \
            --selection-pattern ""
```

Untuk menguji API

1. Untuk menguji `GET` metode, yang akan melakukan validasi permintaan untuk string query, gunakan perintah berikut:

```
aws apigateway test-invoke-method --rest-api-id abc123 \
            --resource-id efg456 \
            --http-method GET \
```

```
            --path-with-query-string '/validate?petType=dog'
```

Hasilnya akan mengembalikan daftar `200 OK` dan anjing-anjingnya.

2. Gunakan perintah berikut untuk menguji tanpa menyertakan string kueri `petType`

```
aws apigateway test-invoke-method --rest-api-id abc123 \
            --resource-id efg456 \
            --http-method GET
```

Hasilnya akan mengembalikan `400` kesalahan.

3. Untuk menguji `POST` metode, yang akan melakukan validasi permintaan untuk badan permintaan, gunakan perintah berikut:

```
aws apigateway test-invoke-method --rest-api-id abc123 \
            --resource-id efg456 \
            --http-method POST \
            --body '{"id": 1, "name": "bella", "type": "dog", "price" : 400 }'
```

Hasilnya akan mengembalikan pesan `200 OK` dan sukses.

4. Gunakan perintah berikut untuk menguji menggunakan badan yang tidak valid.

```
aws apigateway test-invoke-method --rest-api-id abc123 \
             --resource-id efg456 \
             --http-method POST \
             --body '{"id": 1, "name": "bella", "type": "dog", "price" : 1000 }'
```

Hasilnya akan mengembalikan `400` kesalahan, karena harga anjingnya melebihi harga maksimum yang ditentukan oleh model.

Untuk menghapus AWS CloudFormation tumpukan

- Gunakan perintah berikut untuk menghapus AWS CloudFormation sumber daya Anda.

```
aws cloudformation delete-stack  --stack-name request-validation-tutorial-cli
```

Siapkan validasi permintaan dasar menggunakan definisi OpenAPI

Anda dapat mendeklarasikan validator permintaan di API level dengan menentukan satu set x-amazon-apigateway-request-Validators.requestValidator objek objek di x-amazon-apigateway-request-validator objek peta untuk memilih bagian permintaan mana yang akan divalidasi. Dalam definisi OpenAPI contoh, ada dua validator:

- `all`validator yang memvalidasi tubuh, menggunakan model `RequestBodyModel` data, dan parameter.
- `param-only`yang hanya memvalidasi parameter.

Untuk mengaktifkan validator permintaan pada semua metode API, tentukan x-amazon-apigateway-requestproperti -validator properti di tingkat API definisi OpenAPI. Dalam definisi OpenAPI contoh, `all` validator digunakan pada semua metode API, kecuali jika diganti. Saat menggunakan model untuk memvalidasi isi, jika tidak ada jenis konten yang cocok ditemukan, validasi permintaan tidak dilakukan. Untuk menggunakan model yang sama terlepas dari jenis konten, tentukan `$default` sebagai kunci.

Untuk mengaktifkan validator permintaan pada metode individual, tentukan `x-amazon-apigateway-request-validator` properti di tingkat metode. Dalam contoh, definisi OpenAPI, `param-only` validator menimpa validator pada metode. `all` `GET`

Untuk mengimpor contoh OpenAPI ke API Gateway, lihat petunjuk berikut ke Impor API regional ke API Gateway atau ke. Impor API yang dioptimalkan tepi ke API Gateway

OpenAPI 3.0

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "ReqValidators Sample",
    "version" : "1.0.0"
  },
  "servers" : [ {
    "url" : "/{basePath}",
    "variables" : {
      "basePath" : {
        "default" : "/v1"
      }
    }
```

```
 } ],
 "paths" : {
   "/validation" : {
     "get" : {
       "parameters" : [ {
         "name" : "q1",
         "in" : "query",
         "required" : true,
         "schema" : {
           "type" : "string"
         }
       } ],
       "responses" : {
         "200" : {
           "description" : "200 response",
           "headers" : {
             "test-method-response-header" : {
               "schema" : {
                 "type" : "string"
               }
             }
           },
           "content" : {
             "application/json" : {
               "schema" : {
                 "$ref" : "#/components/schemas/ArrayOfError"
               }
             }
           }
         }
       },
       "x-amazon-apigateway-request-validator" : "params-only",
       "x-amazon-apigateway-integration" : {
         "httpMethod" : "GET",
         "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
         "responses" : {
           "default" : {
             "statusCode" : "400",
             "responseParameters" : {
               "method.response.header.test-method-response-header" : "'static
value'"
             },
             "responseTemplates" : {
               "application/xml" : "xml 400 response template",
```

```
              "application/json" : "json 400 response template"
            }
          },
          "2\\d{2}" : {
            "statusCode" : "200"
          }
        },
        "requestParameters" : {
          "integration.request.querystring.type" : "method.request.querystring.q1"
        },
        "passthroughBehavior" : "when_no_match",
        "type" : "http"
      }
    },
    "post" : {
      "parameters" : [ {
        "name" : "h1",
        "in" : "header",
        "required" : true,
        "schema" : {
          "type" : "string"
        }
      } ],
      "requestBody" : {
        "content" : {
          "application/json" : {
            "schema" : {
              "$ref" : "#/components/schemas/RequestBodyModel"
            }
          }
        },
        "required" : true
      },
      "responses" : {
        "200" : {
          "description" : "200 response",
          "headers" : {
            "test-method-response-header" : {
              "schema" : {
                "type" : "string"
              }
            }
          },
          "content" : {
```

```
            "application/json" : {
              "schema" : {
                "$ref" : "#/components/schemas/ArrayOfError"
              }
            }
          }
        }
      },
      "x-amazon-apigateway-request-validator" : "all",
      "x-amazon-apigateway-integration" : {
        "httpMethod" : "POST",
        "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
        "responses" : {
          "default" : {
            "statusCode" : "400",
            "responseParameters" : {
              "method.response.header.test-method-response-header" : "'static
value'"
            },
            "responseTemplates" : {
              "application/xml" : "xml 400 response template",
              "application/json" : "json 400 response template"
            }
          },
          "2\\d{2}" : {
            "statusCode" : "200"
          }
        },
        "requestParameters" : {
          "integration.request.header.custom_h1" : "method.request.header.h1"
        },
        "passthroughBehavior" : "when_no_match",
        "type" : "http"
      }
    }
  }
},
"components" : {
  "schemas" : {
    "RequestBodyModel" : {
      "required" : [ "name", "price", "type" ],
      "type" : "object",
      "properties" : {
        "id" : {
```

```
            "type" : "integer"
          },
          "type" : {
            "type" : "string",
            "enum" : [ "dog", "cat", "fish" ]
          },
          "name" : {
            "type" : "string"
          },
          "price" : {
            "maximum" : 500.0,
            "minimum" : 25.0,
            "type" : "number"
          }
        }
      },
      "ArrayOfError" : {
        "type" : "array",
        "items" : {
          "$ref" : "#/components/schemas/Error"
        }
      },
      "Error" : {
        "type" : "object"
      }
    }
  },
  "x-amazon-apigateway-request-validators" : {
    "all" : {
      "validateRequestParameters" : true,
      "validateRequestBody" : true
    },
    "params-only" : {
      "validateRequestParameters" : true,
      "validateRequestBody" : false
    }
  }
}
```

OpenAPI 2.0

```
{
  "swagger" : "2.0",
```

```
 "info" : {
   "version" : "1.0.0",
   "title" : "ReqValidators Sample"
 },
 "basePath" : "/v1",
 "schemes" : [ "https" ],
 "paths" : {
   "/validation" : {
     "get" : {
       "produces" : [ "application/json", "application/xml" ],
       "parameters" : [ {
         "name" : "q1",
         "in" : "query",
         "required" : true,
         "type" : "string"
       } ],
       "responses" : {
         "200" : {
           "description" : "200 response",
           "schema" : {
             "$ref" : "#/definitions/ArrayOfError"
           },
           "headers" : {
             "test-method-response-header" : {
               "type" : "string"
             }
           }
         }
       },
       "x-amazon-apigateway-request-validator" : "params-only",
       "x-amazon-apigateway-integration" : {
         "httpMethod" : "GET",
         "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
         "responses" : {
           "default" : {
             "statusCode" : "400",
             "responseParameters" : {
               "method.response.header.test-method-response-header" : "'static
value'"
             },
             "responseTemplates" : {
               "application/xml" : "xml 400 response template",
               "application/json" : "json 400 response template"
             }
```

```
            },
            "2\\d{2}" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.querystring.type" : "method.request.querystring.q1"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "http"
        }
      },
      "post" : {
        "consumes" : [ "application/json" ],
        "produces" : [ "application/json", "application/xml" ],
        "parameters" : [ {
          "name" : "h1",
          "in" : "header",
          "required" : true,
          "type" : "string"
        }, {
          "in" : "body",
          "name" : "RequestBodyModel",
          "required" : true,
          "schema" : {
            "$ref" : "#/definitions/RequestBodyModel"
          }
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "schema" : {
              "$ref" : "#/definitions/ArrayOfError"
            },
            "headers" : {
              "test-method-response-header" : {
                "type" : "string"
              }
            }
          }
        },
        "x-amazon-apigateway-request-validator" : "all",
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "POST",
```

```
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
          "responses" : {
            "default" : {
              "statusCode" : "400",
              "responseParameters" : {
                "method.response.header.test-method-response-header" : "'static
value'"
              },
              "responseTemplates" : {
                "application/xml" : "xml 400 response template",
                "application/json" : "json 400 response template"
              }
            },
            "2\\d{2}" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.header.custom_h1" : "method.request.header.h1"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "http"
        }
      }
    }
  },
  "definitions" : {
    "RequestBodyModel" : {
      "type" : "object",
      "required" : [ "name", "price", "type" ],
      "properties" : {
        "id" : {
          "type" : "integer"
        },
        "type" : {
          "type" : "string",
          "enum" : [ "dog", "cat", "fish" ]
        },
        "name" : {
          "type" : "string"
        },
        "price" : {
          "type" : "number",
          "minimum" : 25.0,
```

```
          "maximum" : 500.0
        }
      }
    },
    "ArrayOfError" : {
      "type" : "array",
      "items" : {
        "$ref" : "#/definitions/Error"
      }
    },
    "Error" : {
      "type" : "object"
    }
  },
  "x-amazon-apigateway-request-validators" : {
    "all" : {
      "validateRequestParameters" : true,
      "validateRequestBody" : true
    },
    "params-only" : {
      "validateRequestParameters" : true,
      "validateRequestBody" : false
    }
  }
}
```

Definisi OpenAPI dari API sampel dengan validasi permintaan dasar

Definisi OpenAPI berikut mendefinisikan API sampel dengan validasi permintaan diaktifkan. API adalah bagian dari PetStoreAPI. Ini mengekspos `POST` metode untuk menambahkan hewan peliharaan ke `pets` koleksi dan `GET` metode untuk menanyakan hewan peliharaan dengan jenis tertentu.

Ada dua validator permintaan yang dideklarasikan dalam `x-amazon-apigateway-request-validators` peta di API level. `params-only`Validator diaktifkan pada API dan diwarisi oleh metode. `GET` Validator ini memungkinkan API Gateway untuk memverifikasi bahwa parameter kueri yang diperlukan (`q1`) disertakan dan tidak kosong dalam permintaan masuk. `all`Validator diaktifkan pada metode. `POST` Validator ini memverifikasi bahwa parameter header yang diperlukan (`h1`) disetel dan tidak kosong. Ini juga memverifikasi bahwa format payload mematuhi yang ditentukan `RequestBodyModel` Jika tidak ada jenis konten yang cocok ditemukan, validasi permintaan tidak dilakukan. Saat menggunakan model untuk memvalidasi isi, jika tidak ada jenis konten yang cocok

ditemukan, validasi permintaan tidak dilakukan. Untuk menggunakan model yang sama terlepas dari jenis konten, tentukan `$default` sebagai kunci.

Model ini mensyaratkan bahwa objek input JSON berisi`name`,`type`, dan `price` properti. `name`Properti dapat berupa string apa pun, `type` harus menjadi salah satu bidang enumerasi yang ditentukan (`["dog", "cat", "fish"]`), dan `price` harus berkisar antara 25 dan 500. `id`Parameter tidak diperlukan.

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "title": "ReqValidators Sample",
    "version": "1.0.0"
  },
  "schemes": [
    "https"
  ],
  "basePath": "/v1",
  "produces": [
    "application/json"
  ],
  "x-amazon-apigateway-request-validators" : {
    "all" : {
      "validateRequestBody" : true,
      "validateRequestParameters" : true
    },
    "params-only" : {
      "validateRequestBody" : false,
      "validateRequestParameters" : true
    }
  },
  "x-amazon-apigateway-request-validator" : "params-only",
  "paths": {
    "/validation": {
      "post": {
        "x-amazon-apigateway-request-validator" : "all",
        "parameters": [
          {
            "in": "header",
            "name": "h1",
            "required": true
```

```
      },
      {
        "in": "body",
        "name": "RequestBodyModel",
        "required": true,
        "schema": {
          "$ref": "#/definitions/RequestBodyModel"
        }
      }
    ],
    "responses": {
      "200": {
        "schema": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/Error"
          }
        },
        "headers" : {
          "test-method-response-header" : {
            "type" : "string"
          }
        }
      }
    },
    "security" : [{
      "api_key" : []
    }],
    "x-amazon-apigateway-auth" : {
      "type" : "none"
    },
    "x-amazon-apigateway-integration" : {
      "type" : "http",
      "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
      "httpMethod" : "POST",
      "requestParameters": {
        "integration.request.header.custom_h1": "method.request.header.h1"
      },
      "responses" : {
        "2\\d{2}" : {
          "statusCode" : "200"
        },
        "default" : {
          "statusCode" : "400",
```

```
            "responseParameters" : {
              "method.response.header.test-method-response-header" : "'static
value'"
            },
            "responseTemplates" : {
              "application/json" : "json 400 response template",
              "application/xml" : "xml 400 response template"
            }
          }
        }
      },
      "get": {
        "parameters": [
          {
            "name": "q1",
            "in": "query",
            "required": true
          }
        ],
        "responses": {
          "200": {
            "schema": {
              "type": "array",
              "items": {
                "$ref": "#/definitions/Error"
              }
            },
            "headers" : {
              "test-method-response-header" : {
                "type" : "string"
              }
            }
          }
        },
        "security" : [{
          "api_key" : []
        }],
        "x-amazon-apigateway-auth" : {
          "type" : "none"
        },
        "x-amazon-apigateway-integration" : {
          "type" : "http",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
```

```
          "httpMethod" : "GET",
          "requestParameters": {
            "integration.request.querystring.type": "method.request.querystring.q1"
          },
          "responses" : {
            "2\\d{2}" : {
              "statusCode" : "200"
            },
            "default" : {
              "statusCode" : "400",
              "responseParameters" : {
                "method.response.header.test-method-response-header" : "'static
 value'"
              },
              "responseTemplates" : {
                "application/json" : "json 400 response template",
                "application/xml" : "xml 400 response template"
              }
            }
          }
        }
      }
    }
  },
  "definitions": {
    "RequestBodyModel": {
      "type": "object",
      "properties": {
        "id":    { "type": "integer" },
        "type":  { "type": "string", "enum": ["dog", "cat", "fish"] },
        "name":  { "type": "string" },
        "price": { "type": "number", "minimum": 25, "maximum": 500 }
      },
      "required": ["type", "name", "price"]
    },
    "Error": {
      "type": "object",
      "properties": {

      }
    }
  }
}
```

AWS CloudFormation template API sampel dengan validasi permintaan dasar

AWS CloudFormation Contoh definisi template berikut mendefinisikan API sampel dengan validasi permintaan diaktifkan. API adalah bagian dari PetStoreAPI. Ini mengekspos `POST` metode untuk menambahkan hewan peliharaan ke `pets` koleksi dan `GET` metode untuk menanyakan hewan peliharaan dengan jenis tertentu.

Ada dua validator permintaan yang dideklarasikan:

`GETValidator`

Validator ini diaktifkan pada metode. `GET` Ini memungkinkan API Gateway untuk memverifikasi bahwa parameter kueri yang diperlukan (`q1`) disertakan dan tidak kosong dalam permintaan yang masuk.

`POSTValidator`

Validator ini diaktifkan pada metode. `POST` Ini memungkinkan API Gateway untuk memverifikasi bahwa format permintaan payload mematuhi yang ditentukan `RequestBodyModel` ketika jenis konten `application/json` jika tidak ada jenis konten yang cocok ditemukan, validasi permintaan tidak dilakukan. Untuk menggunakan model yang sama terlepas dari jenis konten, tentukan`$default`. `RequestBodyModel`berisi model tambahan,`RequestBodyModelId`, untuk menentukan ID hewan peliharaan.

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  StageName:
    Type: String
    Default: v1
    Description: Name of API stage.
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Name: ReqValidatorsSample
  RequestBodyModelId:
    Type: 'AWS::ApiGateway::Model'
    Properties:
      RestApiId: !Ref Api
      ContentType: application/json
      Description: Request body model for Pet ID.
```

```
      Schema:
        $schema: 'http://json-schema.org/draft-04/schema#'
        title: RequestBodyModelId
        properties:
            id:
              type: integer
  RequestBodyModel:
    Type: 'AWS::ApiGateway::Model'
    Properties:
      RestApiId: !Ref Api
      ContentType: application/json
      Description: Request body model for Pet type, name, price, and ID.
      Schema:
        $schema: 'http://json-schema.org/draft-04/schema#'
        title: RequestBodyModel
        required:
          - price
          - name
          - type
        type: object
        properties:
            id:
              "$ref": !Sub
                - 'https://apigateway.amazonaws.com/restapis/${Api}/models/
${RequestBodyModelId}'
                - Api: !Ref Api
                  RequestBodyModelId: !Ref RequestBodyModelId
            price:
              type: number
              minimum: 25
              maximum: 500
            name:
              type: string
            type:
              type: string
              enum:
                - "dog"
                - "cat"
                - "fish"
  GETValidator:
    Type: AWS::ApiGateway::RequestValidator
    Properties:
      Name: params-only
      RestApiId: !Ref Api
```

```
      ValidateRequestBody: False
      ValidateRequestParameters: True
  POSTValidator:
    Type: AWS::ApiGateway::RequestValidator
    Properties:
      Name: body-only
      RestApiId: !Ref Api
      ValidateRequestBody: True
      ValidateRequestParameters: False
  ValidationResource:
    Type: 'AWS::ApiGateway::Resource'
    Properties:
      RestApiId: !Ref Api
      ParentId: !GetAtt Api.RootResourceId
      PathPart: 'validation'
  ValidationMethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref ValidationResource
      HttpMethod: GET
      AuthorizationType: NONE
      RequestValidatorId: !Ref GETValidator
      RequestParameters:
        method.request.querystring.q1: true
      Integration:
        Type: HTTP_PROXY
        IntegrationHttpMethod: GET
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
  ValidationMethodPost:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref ValidationResource
      HttpMethod: POST
      AuthorizationType: NONE
      RequestValidatorId: !Ref POSTValidator
      RequestModels:
        application/json : !Ref RequestBodyModel
      Integration:
        Type: HTTP_PROXY
        IntegrationHttpMethod: POST
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
  ApiDeployment:
```

```
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn:
      - ValidationMethodGet
      - RequestBodyModel
    Properties:
      RestApiId: !Ref Api
      StageName: !Sub '${StageName}'
Outputs:
  ApiRootUrl:
    Description: Root Url of the API
    Value: !Sub 'https://${Api}.execute-api.${AWS::Region}.amazonaws.com/${StageName}'
```

Menyiapkan transformasi data untuk REST API

Di API Gateway, permintaan metode API dapat mengambil payload dalam format yang berbeda dari payload permintaan integrasi. Demikian pula, backend dapat mengembalikan payload respons integrasi yang berbeda dari payload respons metode. Anda dapat memetakan parameter jalur URL, parameter string kueri URL, header HTTP, dan badan permintaan di seluruh API Gateway menggunakan templat pemetaan.

Template pemetaan adalah skrip yang dinyatakan dalam Velocity Template Language (VTL) dan diterapkan ke payload menggunakan ekspresi JsonPath.

Muatan dapat memiliki model data sesuai dengan rancangan skema JSON 4. Untuk mempelajari lebih lanjut tentang model, lihatMemahami model data.

> **Note**
>
> Anda tidak perlu mendefinisikan model apa pun untuk membuat template pemetaan, tetapi Anda harus menentukan model agar memiliki API Gateway untuk menghasilkan SDK atau mengaktifkan validasi badan permintaan untuk API Anda.

Topik

- Memahami template pemetaan
- Mengatur transformasi data di API Gateway
- Menggunakan template pemetaan untuk mengganti parameter permintaan dan respons API serta kode status

- Mengatur pemetaan data permintaan dan respons menggunakan konsol API Gateway
- Model dan contoh template pemetaan
- Referensi pemetaan data permintaan dan respons API Amazon API Gateway
- Template pemetaan API Gateway dan referensi variabel pencatatan akses

Memahami template pemetaan

Di API Gateway, permintaan atau respons metode API dapat mengambil muatan dalam format yang berbeda dari permintaan atau respons integrasi.

Anda dapat mengubah data Anda menjadi:

- Cocokkan payload dengan format yang ditentukan API.
- Ganti parameter permintaan dan respons API serta kode status.
- Kembalikan header respons yang dipilih klien.
- Parameter jalur asosiasi, parameter string kueri, atau parameter header dalam permintaan metode proxy atau Layanan AWS proxy HTTP.
- Pilih data mana yang akan dikirim menggunakan integrasi Layanan AWS, seperti fungsi Amazon DynamoDB atau Lambda, atau titik akhir HTTP.

Anda dapat menggunakan template pemetaan untuk mengubah data Anda. Template pemetaan adalah skrip yang dinyatakan dalam Velocity Template Language (VTL) dan diterapkan ke payload menggunakan JsonPath.

Contoh berikut menunjukkan data input, template pemetaan, dan data output untuk transformasi PetStore data.

Masuka data

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

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

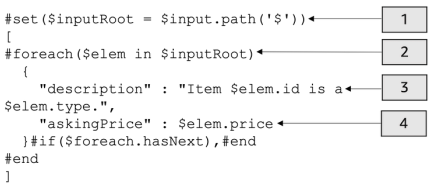
Templat pemeta

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

#end
]
```

Data keluara

```
[
  {
    "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
  }
]
```

Diagram berikut menunjukkan rincian template pemetaan ini.



1. `$inputRoot`Variabel mewakili objek root dalam data JSON asli dari bagian sebelumnya. Arahan dimulai dengan # simbol.
2. Sebuah `foreach` loop iterasi meskipun setiap objek dalam data JSON asli.
3. Deskripsi adalah gabungan dari Pet `id` dan `type` dari data JSON asli.
4. `askingPriceprice`adalah harga dari data JSON asli.

PetStore template pemetaan

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

Dalam template pemetaan ini:

1. Pada baris 1, `$inputRoot` variabel mewakili objek root dalam data JSON asli dari bagian sebelumnya. Arahan dimulai dengan # simbol.
2. Pada baris 3, `foreach` loop iterasi melalui setiap objek dalam data JSON asli.
3. Pada baris 5, `description` ini adalah gabungan dari Pet `id` dan `type` dari data JSON asli.
4. Pada baris 6, `askingPrice` `price` adalah harga dari data JSON asli.

Untuk informasi lebih lanjut tentang Bahasa Template Velocity, lihat Apache Velocity - Referensi VTL. Untuk informasi selengkapnya tentang JsonPath, lihat JSONPath - XPath untuk JSON.

Template pemetaan mengasumsikan bahwa data yang mendasarinya adalah objek JSON. Ini tidak mengharuskan model didefinisikan untuk data. Namun, model untuk data keluaran memungkinkan data sebelumnya dikembalikan sebagai objek khusus bahasa. Untuk informasi selengkapnya, lihat Memahami model data.

Template pemetaan yang kompleks

Anda juga dapat membuat template pemetaan yang lebih rumit. Contoh berikut menunjukkan rangkaian referensi dan batas 100 untuk menentukan apakah hewan peliharaan terjangkau.

Masuka data

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

Templat pemeta

```
#set($inputRoot = $input.path('$'))
#set($cheap = 100)
[
#foreach($elem in $inputRoot)
  {
#set($name = "${elem.type}number$elem.id")
    "name" : $name,
    "description" : "Item $elem.id is a $elem.type.",
    #if($elem.price > $cheap )#set ($afford = 'too much!') #{else}#set
 ($afford = $elem.price)#end
"askingPrice" : $afford
  }#if($foreach.hasNext),#end

#end
]
```

Data keluara

```
[
  {
    "name" : dognumber1,
    "description" : "Item 1 is a dog.",
    "askingPrice" : too much!
  },
  {
```

```
    "name" : catnumber2,
    "description" : "Item 2 is a cat.",
    "askingPrice" : too much!
  },
  {
    "name" : fishnumber3,
    "description" : "Item 3 is a fish.",
    "askingPrice" : 0.99
  }
]
```

Lihat contoh album foto Contoh foto untuk model yang lebih rumit.

Mengatur transformasi data di API Gateway

Bagian ini menunjukkan cara mengatur template pemetaan untuk mengubah permintaan dan tanggapan integrasi menggunakan konsol dan AWS CLI.

Topik

- Mengatur transformasi data menggunakan konsol API Gateway
- Mengatur transformasi data menggunakan AWS CLI
- AWS CloudFormation Templat transformasi data yang lengkap
- Langkah selanjutnya

Mengatur transformasi data menggunakan konsol API Gateway

Dalam tutorial ini, Anda akan membuat tabel API dan DynamoDB yang tidak lengkap menggunakan file.zip berikut.zip. data-transformation-tutorial-console API yang tidak lengkap ini memiliki `/pets` sumber daya dengan `GET` dan `POST` metode.

- `GET`Metode ini akan mendapatkan data dari titik akhir `http://petstore-demo-endpoint.execute-api.com/petstore/pets` HTTP. Data keluaran akan diubah sesuai dengan template pemetaan diPetStore template pemetaan.
- `POST`Metode ini akan memungkinkan pengguna untuk menyimpan informasi `POST` ke tabel Amazon DynamoDB menggunakan template pemetaan.

Unduh dan unzip template pembuatan aplikasi untuk AWS CloudFormation. Anda akan menggunakan template ini untuk membuat tabel DynamoDB untuk memposting informasi hewan peliharaan dan API yang tidak lengkap. Anda akan menyelesaikan langkah-langkah lainnya di konsol API Gateway.

Untuk membuat AWS CloudFormation tumpukan

1. Buka AWS CloudFormation konsol di https://console.aws.amazon.com/cloudformation.
2. Pilih Buat tumpukan kemudian pilih Dengan sumber daya baru (standar).
3. Untuk Tentukan templat, pilih Unggah file templat.
4. Pilih template yang Anda unduh.
5. Pilih Berikutnya.
6. Untuk nama Stack, masukkan **`data-transformation-tutorial-console`** dan kemudian pilih Berikutnya.
7. Untuk opsi Konfigurasi tumpukan, pilih Berikutnya.
8. Untuk Kemampuan, akui bahwa AWS CloudFormation dapat membuat sumber daya IAM di akun Anda.
9. Pilih Buat tumpukan.

AWS CloudFormation ketentuan sumber daya yang ditentukan dalam template. Diperlukan beberapa menit untuk menyelesaikan penyediaan sumber daya Anda. Ketika status AWS CloudFormation tumpukan Anda adalah CREATE_COMPLETE, Anda siap untuk melanjutkan ke langkah berikutnya.

Untuk menguji respon **`GET`** integrasi

1. Pada tab Sumber Daya AWS CloudFormation tumpukan untuk**`data-transformation-tutorial-console`**, pilih ID fisik API Anda.
2. Di panel navigasi utama, pilih Resources, lalu pilih metode GET.
3. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.

   Output dari tes akan menunjukkan yang berikut:

```
[
  {
    "id": 1,
    "type": "dog",
```

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

Anda akan mengubah output ini sesuai dengan template pemetaan diPetStore template pemetaan.

Untuk mengubah respons **GET** integrasi

1. Pilih tab Respons integrasi.

   Saat ini, tidak ada template pemetaan yang ditentukan, sehingga respons integrasi tidak akan diubah.
2. Untuk Default - Respons, pilih Edit.
3. Pilih template Pemetaan, lalu lakukan hal berikut:

   a. Pilih Tambahkan templat pemetaan.

   b. Untuk jenis Konten, masukkan**application/json**.

   c. Untuk badan Template, masukkan yang berikut ini:

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

#end
]
```

Pilih Simpan.

Untuk menguji respon **GET** integrasi

- Pilih tab Test, lalu pilih Test.

  Output dari tes akan menunjukkan respons yang diubah.

```
[
  {
    "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
  }
]
```

Untuk mengubah data input dari **POST** metode

1. Pilih metode POST.
2. Pilih tab Permintaan integrasi, dan kemudian untuk pengaturan permintaan Integrasi, pilih Edit.

   AWS CloudFormation Template telah mengisi beberapa bidang permintaan integrasi.

   - Jenis integrasi adalah Layanan AWS.
   - Layanan AWS Ini adalah DynamoDB.
   - Metode HTTP adalah`POST`.
   - Tindakan adalah`PutItem`.
   - Peran Eksekusi yang memungkinkan API Gateway untuk menempatkan item ke dalam tabel DynamoDB adalah. `data-transformation-tutorial-console-APIGatewayRole`

AWS CloudFormation membuat peran ini untuk memungkinkan API Gateway memiliki izin minimal untuk berinteraksi dengan DynamoDB.

Nama tabel DynamoDB belum ditentukan. Anda akan menentukan nama dalam langkah-langkah berikut.

3. Untuk Request body passthrough, pilih Never.

   Ini berarti bahwa API akan menolak data dengan Content-Types yang tidak memiliki template pemetaan.

4. Pilih template Pemetaan.

5. Jenis konten diatur ke`application/json`. Ini berarti jenis konten yang bukan aplikasi/json akan ditolak oleh API. Untuk informasi selengkapnya tentang perilaku passthrough integrasi, lihat Perilaku passthrough integrasi

6. Masukkan kode berikut ke dalam editor teks.

```
{
    "TableName":"data-transformation-tutorial-console-ddb",
    "Item": {
        "id": {
            "N": $input.json("$.id")
        },
        "type": {
            "S": $input.json("$.type")
        },
        "price": {
            "N": $input.json("$.price")
        }
    }
}
```

   Template ini menentukan tabel sebagai `data-transformation-tutorial-console-ddb` dan menetapkan item sebagai`id`,`type`, dan`price`. Item akan berasal dari tubuh `POST` metode. Anda juga dapat menggunakan model data untuk membantu membuat template pemetaan. Untuk informasi selengkapnya, lihat Gunakan validasi permintaan di API Gateway.

7. Pilih Simpan untuk menyimpan template pemetaan Anda.

Untuk menambahkan metode dan respons integrasi dari **`POST`** metode

AWS CloudFormation Membuat metode kosong dan respons integrasi. Anda akan mengedit tanggapan ini untuk memberikan informasi lebih lanjut. Untuk informasi selengkapnya tentang cara mengedit tanggapan, lihatReferensi pemetaan data permintaan dan respons API Amazon API Gateway.

1. Pada tab Respons Integrasi, untuk Default - Respons, pilih Edit.
2. Pilih Templat pemetaan, lalu pilih Tambahkan templat pemetaan.
3. Untuk tipe Konten, masukkan. **`application/json`**
4. Di editor kode, masukkan template pemetaan keluaran berikut untuk mengirim pesan output:

   ```
   { "message" : "Your response was recorded at $context.requestTime" }
   ```

   Untuk informasi lebih lanjut tentang variabel konteks, lihat`$context`Variabel untuk model data, otorisasi, templat pemetaan, dan CloudWatch pencatatan akses.
5. Pilih Simpan untuk menyimpan template pemetaan Anda.

Uji **`POST`** metodenya

Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.

1. Di badan permintaan, masukkan contoh berikut.

   ```
   {
             "id": "4",
             "type" : "dog",
             "price": "321"
   }
   ```
2. Pilih Uji.

   Output harus menunjukkan pesan sukses Anda.

   Anda dapat membuka konsol DynamoDB di https://console.aws.amazon.com/dynamodb/ untuk memverifikasi bahwa item contoh ada di tabel Anda.

Untuk menghapus AWS CloudFormation tumpukan

1. Buka AWS CloudFormation konsol di https://console.aws.amazon.com/cloudformation.
2. Pilih AWS CloudFormation tumpukan Anda.
3. Pilih Hapus dan kemudian konfirmasikan pilihan Anda.

Mengatur transformasi data menggunakan AWS CLI

Dalam tutorial ini, Anda akan membuat tabel API dan DynamoDB yang tidak lengkap menggunakan file.zip berikut.zip. data-transformation-tutorial-cli API yang tidak lengkap ini memiliki `/pets` sumber daya dengan `GET` metode yang terintegrasi dengan titik akhir `http://petstore-demo-endpoint.execute-api.com/petstore/pets` HTTP. Anda akan membuat `POST` metode untuk terhubung ke tabel DynamoDB dan menggunakan template pemetaan untuk memasukkan data ke dalam tabel DynamoDB.

- Anda akan mengubah data keluaran sesuai dengan template pemetaan diPetStore template pemetaan.
- Anda akan membuat `POST` metode untuk memungkinkan pengguna untuk menyimpan informasi ke `POST` tabel Amazon DynamoDB menggunakan template pemetaan.

Untuk membuat AWS CloudFormation tumpukan

Unduh dan unzip template pembuatan aplikasi untuk AWS CloudFormation.

Untuk menyelesaikan tutorial berikut, Anda memerlukan AWS Command Line Interface (AWS CLI) versi 2.

Untuk perintah panjang, karakter escape (\) digunakan untuk memisahkan perintah menjadi beberapa baris.

> **Note**
>
> Di Windows, beberapa perintah Bash CLI yang biasa Anda gunakan (`zip`seperti) tidak didukung oleh terminal bawaan sistem operasi. Untuk mendapatkan versi terintegrasi Windows dari Ubuntu dan Bash, instal Windows Subsystem untuk Linux. Contoh perintah CLI dalam panduan ini menggunakan pemformatan Linux. Perintah yang menyertakan dokumen JSON sebaris harus diformat ulang jika Anda menggunakan CLI Windows.

1. Gunakan perintah berikut untuk membuat AWS CloudFormation tumpukan.

```
aws cloudformation create-stack --stack-name data-transformation-tutorial-cli
 --template-body file://data-transformation-tutorial-cli.zip --capabilities
 CAPABILITY_NAMED_IAM
```

2. AWS CloudFormation ketentuan sumber daya yang ditentukan dalam template. Diperlukan beberapa menit untuk menyelesaikan penyediaan sumber daya Anda. Gunakan perintah berikut untuk melihat status AWS CloudFormation tumpukan Anda.

```
aws cloudformation describe-stacks --stack-name data-transformation-tutorial-cli
```

3. Ketika status AWS CloudFormation tumpukan Anda`StackStatus: "CREATE_COMPLETE"`, gunakan perintah berikut untuk mengambil nilai output yang relevan untuk langkah-langkah masa depan.

```
aws cloudformation describe-stacks --stack-name data-transformation-tutorial-cli
--query "Stacks[*].Outputs[*].{OutputKey: OutputKey, OutputValue: OutputValue,
Description: Description}"
```

   Nilai output adalah sebagai berikut:

   - ApiRole, yang merupakan nama peran yang memungkinkan API Gateway untuk menempatkan item dalam tabel DynamoDB. Untuk tutorial ini, nama perannya adalah`data-transformation-tutorial-cli-APIGatewayRole-ABCDEFG`.
   - DDBTableName, yang merupakan nama dari tabel DynamoDB. Untuk tutorial ini, nama tabelnya adalah `data-transformation-tutorial-cli-ddb`
   - ResourceId, yang merupakan ID untuk sumber daya hewan peliharaan tempat `POST` metode `GET` dan diekspos. Untuk tutorial ini, ID Sumber Daya adalah `efg456`
   - ApiId, yang merupakan ID untuk API. Untuk tutorial ini, ID API adalah`abc123`.

Untuk menguji **`GET`** metode sebelum transformasi data

- Gunakan perintah berikut untuk menguji `GET` metode.

```
aws apigateway test-invoke-method --rest-api-id abc123 \
          --resource-id efg456 \
          --http-method GET
```

Output dari tes akan menunjukkan yang berikut ini.

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

Anda akan mengubah output ini sesuai dengan template pemetaan diPetStore template pemetaan.

Untuk mengubah respons **GET** integrasi

- Gunakan perintah berikut untuk memperbarui respons integrasi untuk GET metode ini. Ganti *rest-api-id*dan *resource-id* dengan nilai Anda.

  Gunakan perintah berikut untuk membuat respons integrasi.

```
aws apigateway put-integration-response --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method GET \
  --status-code 200 \
  --selection-pattern "" \
  --response-templates '{"application/json": "#set($inputRoot = $input.path(\"$
\"))\n[\n#foreach($elem in $inputRoot)\n {\n  \"description\": \"Item $elem.id is a
 $elem.type\",\n  \"askingPrice\": \"$elem.price\"\n }#if($foreach.hasNext),#end\n
\n#end\n]"}'
```

Untuk menguji **GET** metode

- Gunakan perintah berikut untuk menguji GET metode.

```
aws apigateway test-invoke-method --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method GET \
```

Output dari tes akan menunjukkan respons yang diubah.

```
[
  {
    "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
  }
]
```

Untuk membuat **POST** metode

1. Gunakan perintah berikut untuk membuat metode baru pada `/pets` sumber daya.

```
aws apigateway put-method --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method POST \
  --authorization-type "NONE" \
```

Metode ini akan memungkinkan Anda untuk mengirim informasi hewan peliharaan ke tabel DynamoDB yang Anda buat di tumpukan. AWS CloudFormation

2. Gunakan perintah berikut untuk membuat Layanan AWS integrasi pada POST metode.

```
aws apigateway put-integration --rest-api-id abc123 \
  --resource-id efg456 \
```

```
  --http-method POST \
  --type AWS \
  --integration-http-method POST \
  --uri "arn:aws:apigateway:us-east-2:dynamodb:action/PutItem" \
  --credentials arn:aws:iam::111122223333:role/data-transformation-tutorial-cli-
APIGatewayRole-ABCDEFG \
  --request-templates '{"application/json":"{\"TableName\":\"data-transformation-
tutorial-cli-ddb\",\"Item\":{\"id\":{\"N\":$input.json(\"$.id\")},\"type\":{\"S\":
$input.json(\"$.type\")},\"price\":{\"N\":$input.json(\"$.price\")} }}"}'
```

3. Gunakan perintah berikut untuk membuat respons metode untuk panggilan POST metode yang berhasil.

```
aws apigateway put-method-response --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method POST \
  --status-code 200
```

4. Gunakan perintah berikut untuk membuat respons integrasi untuk panggilan POST metode yang berhasil.

```
aws apigateway put-integration-response --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method POST \
  --status-code 200 \
  --selection-pattern "" \
  --response-templates '{"application/json": "{\"message\": \"Your response was
 recorded at $context.requestTime\"}"}'
```

Untuk menguji **POST** metode

- Gunakan perintah berikut untuk menguji POST metode.

```
aws apigateway test-invoke-method --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method POST \
  --body '{\"id\": \"4\", \"type\": \"dog\", \"price\": \"321\"}'
```

Output akan menampilkan pesan yang berhasil.

Untuk menghapus AWS CloudFormation tumpukan

- Gunakan perintah berikut untuk menghapus AWS CloudFormation sumber daya Anda.

```
aws cloudformation delete-stack  --stack-name data-transformation-tutorial-cli
```

AWS CloudFormation Templat transformasi data yang lengkap

Contoh berikut adalah AWS CloudFormation template lengkap, yang membuat API dan tabel DynamoDB dengan sumber daya `/pets` `GET` dengan dan metode. `POST`

- `GET`Metode ini akan mendapatkan data dari titik akhir `http://petstore-demo-endpoint.execute-api.com/petstore/pets` HTTP. Data keluaran akan diubah sesuai dengan template pemetaan diPetStore template pemetaan.
- `POST`Metode ini akan memungkinkan pengguna untuk menyimpan informasi `POST` ke tabel DynamoDB menggunakan template pemetaan.

```
AWSTemplateFormatVersion: 2010-09-09
Description: A completed Amazon API Gateway REST API that uses non-proxy integration
 to POST to an Amazon DynamoDB table and non-proxy integration to GET transformed pets
 data.
Parameters:
  StageName:
    Type: String
    Default: v1
    Description: Name of API stage.
Resources:
  DynamoDBTable:
    Type: 'AWS::DynamoDB::Table'
    Properties:
      TableName: !Sub data-transformation-tutorial-complete
      AttributeDefinitions:
        - AttributeName: id
          AttributeType: N
      KeySchema:
        - AttributeName: id
          KeyType: HASH
      ProvisionedThroughput:
        ReadCapacityUnits: 5
        WriteCapacityUnits: 5
```

```
  APIGatewayRole:
    Type: 'AWS::IAM::Role'
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Action:
              - 'sts:AssumeRole'
            Effect: Allow
            Principal:
              Service:
                - apigateway.amazonaws.com
      Policies:
        - PolicyName: APIGatewayDynamoDBPolicy
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                  - 'dynamodb:PutItem'
                Resource: !GetAtt DynamoDBTable.Arn
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Name: data-transformation-complete-api
      ApiKeySourceType: HEADER
  PetsResource:
    Type: 'AWS::ApiGateway::Resource'
    Properties:
      RestApiId: !Ref Api
      ParentId: !GetAtt Api.RootResourceId
      PathPart: 'pets'
  PetsMethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref PetsResource
      HttpMethod: GET
      ApiKeyRequired: false
      AuthorizationType: NONE
      Integration:
        Type: HTTP
        Credentials: !GetAtt APIGatewayRole.Arn
        IntegrationHttpMethod: GET
```

```
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
        PassthroughBehavior: WHEN_NO_TEMPLATES
        IntegrationResponses:
          - StatusCode: '200'
            ResponseTemplates:
              application/json: "#set($inputRoot = $input.path(\"$
\"))\n[\n#foreach($elem in $inputRoot)\n {\n  \"description\": \"Item $elem.id is a
 $elem.type\",\n  \"askingPrice\": \"$elem.price\"\n }#if($foreach.hasNext),#end\n
\n#end\n]"
      MethodResponses:
        - StatusCode: '200'
  PetsMethodPost:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref PetsResource
      HttpMethod: POST
      ApiKeyRequired: false
      AuthorizationType: NONE
      Integration:
        Type: AWS
        Credentials: !GetAtt APIGatewayRole.Arn
        IntegrationHttpMethod: POST
        Uri: arn:aws:apigateway:us-west-1:dynamodb:action/PutItem
        PassthroughBehavior: NEVER
        RequestTemplates:
          application/json: "{\"TableName\":\"data-transformation-tutorial-complete
\",\"Item\":{\"id\":{\"N\":$input.json(\"$.id\")},\"type\":{\"S\":$input.json(\"$.type
\")},\"price\":{\"N\":$input.json(\"$.price\")} }}"
        IntegrationResponses:
          - StatusCode: 200
            ResponseTemplates:
              application/json: "{\"message\": \"Your response was recorded at
 $context.requestTime\"}"
      MethodResponses:
        - StatusCode: '200'

  ApiDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn:
      - PetsMethodGet
    Properties:
      RestApiId: !Ref Api
      StageName: !Sub '${StageName}'
```

```
Outputs:
  ApiId:
    Description: API ID for CLI commands
    Value: !Ref Api
  ResourceId:
    Description: /pets resource ID for CLI commands
    Value: !Ref PetsResource
  ApiRole:
    Description: Role ID to allow API Gateway to put and scan items in DynamoDB table
    Value: !Ref APIGatewayRole
  DDBTableName:
    Description: DynamoDB table name
    Value: !Ref DynamoDBTable
```

Langkah selanjutnya

Untuk menjelajahi template pemetaan yang lebih kompleks, lihat contoh berikut:

- Lihat model yang lebih kompleks dan templat pemetaan dengan contoh album Contoh foto foto.
- Untuk informasi lebih lanjut tentang model, lihatMemahami model data.
- Untuk informasi tentang cara memetakan output kode respons yang berbeda,Mengatur pemetaan data permintaan dan respons menggunakan konsol API Gateway.
- Untuk informasi tentang cara mengatur pemetaan data dari data permintaan metode API,. Template pemetaan API Gateway dan referensi variabel pencatatan akses

Menggunakan template pemetaan untuk mengganti parameter permintaan dan respons API serta kode status

Parameter API Gateway standar dan templat pemetaan kode respons memungkinkan Anda memetakan parameter one-to-one dan memetakan keluarga kode status respons integrasi (dicocokkan dengan ekspresi reguler) ke kode status respons tunggal. Penggantian template pemetaan memberi Anda fleksibilitas untuk melakukan pemetaan many-to-one parameter; mengganti parameter setelah pemetaan API Gateway standar diterapkan; memetakan parameter secara kondisional berdasarkan konten isi atau nilai parameter lainnya; membuat parameter baru secara terprogram dengan cepat; dan ganti kode status yang dikembalikan oleh titik akhir integrasi Anda. Semua jenis parameter permintaan, header respons, atau kode status respons dapat diganti.

Berikut ini adalah contoh penggunaan untuk penggantian template pemetaan:

- Untuk membuat header baru (atau menimpa header yang ada) sebagai gabungan dari dua parameter
- Untuk mengganti kode respons ke kode sukses atau gagal berdasarkan isi tubuh
- Untuk memetakan ulang parameter secara kondisional berdasarkan isinya atau isi dari beberapa parameter lainnya
- Untuk mengulangi konten badan json dan memetakan ulang pasangan nilai kunci ke header atau string kueri

Untuk membuat penggantian template pemetaan, gunakan satu atau beberapa $contextvariabel berikut dalam templat pemetaan:

Minta templat pemetaan tubuh	Templat pemetaan tubuh respons
`$context.requestOverride.header.` *`header_name`*	`$context.responseOverride.header.` *`header_name`*
`$context.requestOverride.path.` *`path_name`*	`$context.responseOverride.status`
`$context.requestOverride.querystring.` *`querystring_name`*	

Note

Penggantian template pemetaan tidak dapat digunakan dengan titik akhir integrasi proxy, yang tidak memiliki pemetaan data. Untuk informasi selengkapnya tentang jenis integrasi, lihatPilih jenis integrasi API Gateway API.

Important

Override adalah final. Override hanya dapat diterapkan ke setiap parameter satu kali. Mencoba mengganti parameter yang sama beberapa kali akan menghasilkan 5XX respons dari Amazon API Gateway. Jika Anda harus mengganti parameter yang sama beberapa kali di seluruh template, kami sarankan membuat variabel dan menerapkan override di akhir

template. Perhatikan bahwa template diterapkan hanya setelah seluruh template diuraikan. Lihat Tutorial: Ganti parameter permintaan API dan header dengan konsol API Gateway.

Tutorial berikut menunjukkan cara membuat dan menguji penggantian template pemetaan di konsol API Gateway. Tutorial ini menggunakan PetStore contoh API sebagai titik awal. Kedua tutorial berasumsi bahwa Anda telah membuat API PetStore sampel.

Topik

- Tutorial: Ganti kode status respons API dengan konsol API Gateway
- Tutorial: Ganti parameter permintaan API dan header dengan konsol API Gateway
- Contoh: Ganti parameter permintaan API dan header dengan API Gateway CLI
- Contoh: Ganti parameter permintaan API dan header menggunakan SDK for JavaScript

Tutorial: Ganti kode status respons API dengan konsol API Gateway

Untuk mengambil hewan peliharaan menggunakan API PetStore sampel, Anda menggunakan permintaan metode API`GET /pets/{petId}`, di mana `{petId}` merupakan parameter jalur yang dapat mengambil nomor pada waktu proses.

Dalam tutorial ini, Anda akan mengganti kode respons `GET` metode ini dengan membuat template pemetaan yang `$context.responseOverride.status` memetakan `400` ketika kondisi kesalahan terdeteksi.

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Di bawah API, pilih PetStore API, lalu pilih Resources.
3. Di pohon Resources, pilih `GET` metode di bawah`/{petId}`.
4. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
5. Untuk PeTiD, masukkan**`-1`**, lalu pilih Test.

   Dalam hasilnya, Anda akan melihat dua hal:

   Pertama, badan Respons menunjukkan out-of-range kesalahan:

```
{
  "errors": [
    {
```

```
      "key": "GetPetRequest.petId",
      "message": "The value is out of range."
    }
  ]
}
```

Kedua, baris terakhir di bawah kotak Log diakhiri dengan:`Method completed with status: 200`.

6. Pada tab Respons Integrasi, untuk Default - Respons, pilih Edit.
7. Pilih template Pemetaan.
8. Pilih Tambahkan templat pemetaan.
9. Untuk jenis Konten, masukkan**`application/json`**.
10. Untuk badan Template, masukkan yang berikut ini:

```
#set($inputRoot = $input.path('$'))
$input.json("$")
#if($inputRoot.toString().contains("error"))
#set($context.responseOverride.status = 400)
#end
```

11. Pilih Simpan.
12. Pilih tab Uji.
13. Untuk PeTiD, masukkan. **`-1`**
14. Dalam hasilnya, Response Body menunjukkan out-of-range kesalahan:

```
{
  "errors": [
    {
      "key": "GetPetRequest.petId",
      "message": "The value is out of range."
    }
  ]
}
```

Namun, baris terakhir di bawah kotak Log sekarang berakhir dengan:`Method completed with status: 400`.

Tutorial: Ganti parameter permintaan API dan header dengan konsol API Gateway

Dalam tutorial ini, Anda akan mengganti kode header permintaan `GET` metode dengan membuat template pemetaan yang memetakan `$context.requestOverride.header.`*`header_name`* ke header baru yang menggabungkan dua header lainnya.

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Di bawah API, pilih PetStore API.
3. Di pohon Resources, pilih `GET` metode di bawah`/pet`.
4. Pada tab Permintaan metode, untuk pengaturan permintaan Metode, pilih Edit.
5. Pilih header permintaan HTTP, lalu pilih Tambah header.
6. Untuk Nama, masukkan **`header1`**.
7. Pilih Tambahkan header, lalu buat header kedua yang disebut**`header2`**.
8. Pilih Simpan.
9. Pada tab Permintaan integrasi, untuk pengaturan permintaan Integrasi, pilih Edit.
10. Untuk Request body passthrough, pilih Bila tidak ada templat yang ditentukan (disarankan).
11. Pilih template Pemetaan, lalu lakukan hal berikut:
    a. Pilih Tambahkan templat pemetaan.
    b. Untuk jenis Konten, masukkan**`application/json`**.
    c. Untuk badan Template, masukkan yang berikut ini:

```
#set($header1Override = "foo")
#set($header3Value = "$input.params('header1')$input.params('header2')")
$input.json("$")
#set($context.requestOverride.header.header3 = $header3Value)
#set($context.requestOverride.header.header1 = $header1Override)
#set($context.requestOverride.header.multivalueheader=[$header1Override,
 $header3Value])
```

12. Pilih Simpan.
13. Pilih tab Uji.
14. Di bawah Header untuk {pets}, salin kode berikut:

```
header1:header1Val
```

```
header2:header2Val
```

15. Pilih Uji.

    Di Log, Anda akan melihat entri yang menyertakan teks ini:

```
Endpoint request headers: {header3=header1Valheader2Val,
header2=header2Val, header1=foo, x-amzn-apigateway-api-id=<api-id>,
Accept=application/json, multivalueheader=foo,header1Valheader2Val}
```

Contoh: Ganti parameter permintaan API dan header dengan API Gateway CLI

Contoh CLI berikut menunjukkan cara menggunakan `put-integration` perintah untuk mengganti kode respons:

```
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>
```

di mana <REQUEST_TEMPLATE_MAP>adalah peta dari jenis konten ke string template untuk diterapkan. Struktur peta tersebut adalah sebagai berikut:

```
Content_type1=template_string,Content_type2=template_string
```

atau, dalam sintaks JSON:

```
{"content_type1": "template_string"
...}
```

Contoh berikut menunjukkan cara menggunakan `put-integration-response` perintah untuk mengganti kode respons 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>
```

di mana <RESPONSE_TEMPLATE_MAP>memiliki format yang sama seperti <REQUEST_TEMPLATE_MAP>di atas.

Contoh: Ganti parameter permintaan API dan header menggunakan SDK for JavaScript

Contoh berikut menunjukkan cara menggunakan `put-integration` perintah untuk mengganti kode respons:

Permintaan:

```
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
});
```

Tanggapan:

```
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
});
```

Mengatur pemetaan data permintaan dan respons menggunakan konsol API Gateway

Untuk menggunakan konsol API Gateway untuk menentukan permintaan/respons integrasi API, ikuti petunjuk berikut.

> **Note**
>
> Instruksi ini mengasumsikan Anda telah menyelesaikan langkah-langkahnyaMenyiapkan permintaan integrasi API menggunakan konsol API Gateway.

1. Di panel Resources, pilih metode Anda.
2. Pada tab Permintaan integrasi, di bawah Pengaturan permintaan integrasi, pilih Edit.
3. Pilih opsi untuk Request body passthrough untuk mengonfigurasi bagaimana badan permintaan metode dari jenis konten yang tidak dipetakan akan diteruskan melalui permintaan integrasi tanpa transformasi ke fungsi Lambda, proxy HTTP, atau proxy layanan. AWS Ada tiga opsi:

   - Pilih Bila tidak ada templat yang cocok dengan header tipe konten permintaan jika Anda ingin badan permintaan metode melewati permintaan integrasi ke backend tanpa transformasi saat jenis konten permintaan metode tidak cocok dengan jenis konten apa pun yang terkait dengan templat pemetaan, seperti yang ditentukan pada langkah berikutnya.

     > **Note**
     >
     > Saat memanggil API Gateway API, Anda memilih opsi ini dengan menetapkan `WHEN_NO_MATCH` sebagai nilai `passthroughBehavior` properti pada sumber daya Integrasi.

   - Pilih Ketika tidak ada templat yang ditentukan (disarankan) jika Anda ingin badan permintaan metode melewati permintaan integrasi ke backend tanpa transformasi ketika tidak ada templat pemetaan yang ditentukan dalam permintaan integrasi. Jika templat ditentukan saat opsi ini dipilih, permintaan metode dari jenis konten yang tidak dipetakan akan ditolak dengan respons Jenis Media Tidak Didukung HTTP 415.

     > **Note**
     >
     > Saat memanggil API Gateway API, Anda memilih opsi ini dengan menetapkan `WHEN_NO_TEMPLATE` sebagai nilai `passthroughBehavior` properti pada sumber daya Integrasi.

   - Pilih Jangan Pernah jika Anda tidak ingin permintaan metode terlewati saat jenis konten permintaan metode tidak cocok dengan jenis konten apa pun yang terkait dengan templat

pemetaan yang ditentukan dalam permintaan integrasi atau tidak ada templat pemetaan yang ditentukan dalam permintaan integrasi. Permintaan metode dari jenis konten yang tidak dipetakan akan ditolak dengan respons Jenis Media Tidak Didukung HTTP 415.

> **Note**
>
> Saat memanggil API Gateway API, Anda memilih opsi ini dengan menetapkan `NEVER` sebagai nilai `passthroughBehavior` properti pada sumber daya Integrasi.

Untuk informasi selengkapnya tentang perilaku passthrough integrasi, lihatPerilaku passthrough integrasi.

4. Untuk proxy HTTP atau proxy AWS layanan, untuk mengaitkan parameter jalur, parameter string kueri, atau parameter header yang ditentukan dalam permintaan integrasi dengan parameter jalur yang sesuai, parameter string kueri, atau parameter header dalam permintaan metode proxy HTTP atau proxy AWS layanan, lakukan hal berikut:

   a. Pilih parameter jalur URL, parameter string kueri URL, atau header permintaan HTTP masing-masing, lalu pilih Tambah jalur, Tambahkan string kueri, atau Tambahkan header, masing-masing.

   b. Untuk Nama, ketik nama parameter jalur, parameter string kueri, atau parameter header di proxy HTTP atau proxy AWS layanan.

   c. Untuk Dipetakan dari, masukkan nilai pemetaan untuk parameter jalur, parameter string kueri, atau parameter header. Gunakan salah satu format berikut:

      - **`method.request.path.`*`parameter-name`*** untuk parameter jalur bernama *`parameter-name`* seperti yang didefinisikan dalam halaman permintaan Metode.
      - **`method.request.querystring.`*`parameter-name`*** untuk parameter string kueri bernama *`parameter-name`* seperti yang didefinisikan dalam halaman permintaan Metode.
      - **`method.request.multivaluequerystring.`*`parameter-name`*** untuk parameter string kueri multi-nilai bernama ***`parameter-name`*** seperti yang didefinisikan dalam halaman permintaan Metode.
      - **`method.request.header.`*`parameter-name`*** untuk parameter header bernama *`parameter-name`* seperti yang didefinisikan dalam halaman permintaan Metode.

Atau, Anda dapat mengatur nilai string literal (tertutup oleh sepasang tanda kutip tunggal) ke header integrasi.

- **`method.request.multivalueheader.`*`parameter-name`*** untuk parameter header multi-nilai bernama ***`parameter-name`*** seperti yang didefinisikan dalam halaman permintaan Metode.

d. Untuk menambahkan parameter lain, pilih tombol Tambah.

5. Untuk menambahkan templat pemetaan, pilih Template pemetaan.
6. Untuk menentukan template pemetaan untuk permintaan masuk, pilih Tambahkan templat pemetaan. Untuk jenis Konten, masukkan jenis konten (mis.,**`application/json`**). Kemudian, masukkan template pemetaan secara manual atau pilih Buat template untuk membuatnya dari template model. Untuk informasi selengkapnya, lihat Memahami template pemetaan.
7. Pilih Simpan.
8. Anda dapat memetakan respons integrasi dari backend ke respons metode API yang dikembalikan ke aplikasi pemanggilan. Ini termasuk mengembalikan header respons yang dipilih klien dari yang tersedia dari backend, mengubah format data payload respons backend ke format yang ditentukan API. Anda dapat menentukan pemetaan tersebut dengan mengonfigurasi respons Metode dan tanggapan Integrasi.

   Agar metode menerima format data respons khusus berdasarkan kode status HTTP yang dikembalikan oleh fungsi Lambda, proxy HTTP, atau proxy AWS layanan, lakukan hal berikut:

   a. Pilih tanggapan Integrasi. Pilih salah satu Edit pada Default - Respons, untuk menentukan pengaturan untuk 200 kode respons HTTP dari metode, atau pilih Buat respons untuk menentukan pengaturan untuk kode status respons HTTP lainnya dari metode.

   b. Untuk regex kesalahan Lambda (untuk fungsi Lambda) atau regex status HTTP (untuk proxy HTTP atau proxy AWS layanan), masukkan ekspresi reguler untuk menentukan string kesalahan fungsi Lambda mana (untuk fungsi Lambda) atau kode status respons HTTP (untuk proxy HTTP atau proxy layanan) peta ke pemetaan keluaran ini. AWS Misalnya, untuk memetakan semua kode status respons HTTP 2xx dari proxy HTTP ke pemetaan keluaran ini, ketik "**`2\d{2}`**" untuk regex status HTTP. Untuk mengembalikan pesan kesalahan yang berisi “Permintaan Tidak Valid” dari fungsi Lambda ke **`400 Bad Request`** respons, masukkan "" **`.*Invalid request.*`** sebagai ekspresi regex kesalahan Lambda. Di sisi lain, **`400 Bad Request`** untuk mengembalikan semua pesan kesalahan yang belum dipetakan dari Lambda, masukkan "" **`(\n|.)+`** di Lambda error regex. Ekspresi reguler terakhir ini dapat digunakan untuk respons kesalahan default API.

> **Note**
>
> API Gateway menggunakan regex gaya pola Java untuk pemetaan respons. Untuk informasi selengkapnya, lihat Pola dalam Oracle dokumentasi.
> Pola kesalahan dicocokkan dengan seluruh string `errorMessage` properti dalam respons Lambda, yang diisi `callback(errorMessage)` oleh di Node.js atau `throw new MyException(errorMessage)` oleh di Java. Selain itu, karakter yang lolos tidak dapat diloloskan sebelum ekspresi reguler diterapkan.
> Jika Anda menggunakan '.+' sebagai pola pemilihan untuk memfilter respons, ketahuilah bahwa itu mungkin tidak cocok dengan respons yang berisi karakter baris baru ('\n').

c. Jika diaktifkan, untuk status respons Metode, pilih kode status respons HTTP yang Anda tentukan di halaman respons Metode.

d. Untuk pemetaan Header, untuk setiap header yang Anda tentukan untuk kode status respons HTTP pada halaman respons Metode, tentukan nilai pemetaan. Untuk nilai Pemetaan, gunakan salah satu format berikut:

- **`integration.response.multivalueheaders.`*`header-name`*** di mana *`header-name`* adalah nama header respons multi-nilai dari backend.

  Misalnya, untuk mengembalikan header respons backend sebagai **`Date`** header respons metode API, kolom header Response akan berisi entri Timestamp, dan nilai Pemetaan terkait harus disetel ke Integration.response.MultivalueHeaders.date. **`Timestamp`**

- **`integration.response.header.`*`header-name`*** di mana *`header-name`* adalah nama header respons bernilai tunggal dari backend.

  Misalnya, untuk mengembalikan header respons backend sebagai **`Date`** header respons metode API, kolom header Response akan berisi entri Timestamp, dan nilai Pemetaan terkait harus disetel ke Integration.response.Header.date. **`Timestamp`**

e. Pilih Templat pemetaan, lalu pilih Tambahkan templat pemetaan. Di kotak Jenis konten, masukkan tipe konten data yang akan diteruskan dari fungsi Lambda, proxy HTTP, atau proxy AWS layanan ke metode. Kemudian, masukkan template pemetaan secara manual atau pilih Buat template untuk membuatnya dari template model. Untuk informasi selengkapnya, lihat Memahami template pemetaan.

f. Pilih Simpan.

Model dan contoh template pemetaan

Bagian berikut memberikan contoh model dan templat pemetaan yang dapat digunakan sebagai titik awal untuk API Anda sendiri di API Gateway. Untuk informasi selengkapnya tentang model dan templat pemetaan di API Gateway, lihatPetStore template pemetaan.

Topik

- Contoh foto (model API Gateway dan templat pemetaan)
- Contoh artikel berita (model API Gateway dan template pemetaan)
- Contoh faktur penjualan (model API Gateway dan templat pemetaan)
- Contoh catatan karyawan (model API Gateway dan templat pemetaan)

Contoh foto (model API Gateway dan templat pemetaan)

Contoh berikut menunjukkan API album foto di API Gateway. Kami menyediakan contoh transformasi data, model tambahan, dan templat pemetaan. Untuk informasi selengkapnya tentang transformasi data, lihat Memahami template pemetaan Untuk informasi selengkapnya tentang model dan templat pemetaan di API Gateway, lihat. PetStore template pemetaan

Topik

- Contoh transformasi data
- Model masukan untuk data foto
- Model keluaran untuk data foto
- Template pemetaan masukan untuk data foto

Contoh transformasi data

Contoh berikut menunjukkan bagaimana Anda dapat mengubah data input tentang foto dengan menggunakan template pemetaan Velocity Template Language (VTL). Untuk informasi lebih lanjut tentang Bahasa Template Velocity, lihat Apache Velocity - Referensi VTL.

Masuka data

```
{
  "photos": {
    "page": 1,
    "pages": "1234",
    "perpage": 100,
```

```
    "total": "123398",
    "photo": [
      {
        "id": "12345678901",
        "owner": "23456789@A12",
        "photographer_first_name" : "Saanvi",
        "photographer_last_name" : "Sarkar",
        "secret": "abc123d456",
        "server": "1234",
        "farm": 1,
        "title": "Sample photo 1",
        "ispublic": true,
        "isfriend": false,
        "isfamily": false
      },
      {
        "id": "23456789012",
        "owner": "34567890@B23",
        "photographer_first_name" : "Richard",
        "photographer_last_name" : "Roe",
        "secret": "bcd234e567",
        "server": "2345",
        "farm": 2,
        "title": "Sample photo 2",
        "ispublic": true,
        "isfriend": false,
        "isfamily": false
      }
    ]
  }
}
```

Templat pemeta keluara

```
#set($inputRoot = $input.path('$'))
{
  "photos": [
#foreach($elem in $inputRoot.photos.photo)
    {
      "id": "$elem.id",
      "photographedBy": "$elem.photographer_first_name $elem.pho
tographer_last_name",
      "title": "$elem.title",
      "ispublic": $elem.ispublic,
      "isfriend": $elem.isfriend,
      "isfamily": $elem.isfamily
    }#if($foreach.hasNext),#end

#end
  ]
}
```

Data keluara

```
{
  "photos": [
    {
      "id": "12345678901",
      "photographedBy": "Saanvi Sarkar",
      "title": "Sample photo 1",
      "ispublic": true,
      "isfriend": false,
      "isfamily": false
    },
    {
      "id": "23456789012",
      "photographedBy": "Richard Roe",
      "title": "Sample photo 2",
      "ispublic": true,
      "isfriend": false,
      "isfamily": false
    }
  ]
}
```

Model masukan untuk data foto

Anda dapat menentukan model untuk data input Anda. Model input ini mengharuskan Anda mengunggah satu foto, dan menentukan minimal 10 foto per halaman. Anda dapat menggunakan model input ini untuk menghasilkan SDK atau mengaktifkan validasi permintaan untuk API Anda.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "PhotosInputModel",
  "type": "object",
  "properties": {
    "photos": {
      "type": "object",
      "required" : [
      "photo"
      ],
      "properties": {
        "page": { "type": "integer" },
        "pages": { "type": "string" },
        "perpage": { "type": "integer", "minimum" : 10 },
        "total": { "type": "string" },
        "photo": {
          "type": "array",
          "items": {
            "type": "object",
            "properties": {
              "id": { "type": "string" },
              "owner": { "type": "string" },
              "photographer_first_name" : {"type" : "string"},
              "photographer_last_name" : {"type" : "string"},
              "secret": { "type": "string" },
              "server": { "type": "string" },
              "farm": { "type": "integer" },
              "title": { "type": "string" },
              "ispublic": { "type": "boolean" },
              "isfriend": { "type": "boolean" },
              "isfamily": { "type": "boolean" }
            }
          }
        }
      }
    }
  }
```

```
}
```

Model keluaran untuk data foto

Anda dapat menentukan model untuk data keluaran Anda. Anda dapat menggunakan model ini untuk model respons metode, yang diperlukan saat Anda membuat SDK yang diketik kuat untuk API. Hal ini menyebabkan output dilemparkan ke kelas yang sesuai di Java atau Objective-C.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "PhotosOutputModel",
  "type": "object",
  "properties": {
    "photos": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "id": { "type": "string" },
          "photographedBy": { "type": "string" },
          "title": { "type": "string" },
          "ispublic": { "type": "boolean" },
          "isfriend": { "type": "boolean" },
          "isfamily": { "type": "boolean" }
        }
      }
    }
  }
}
```

Template pemetaan masukan untuk data foto

Anda dapat menentukan template pemetaan untuk memodifikasi data input. Anda dapat memodifikasi data input untuk integrasi fungsi atau respons integrasi lebih lanjut.

```
#set($inputRoot = $input.path('$'))
{
  "photos": {
    "page": $inputRoot.photos.page,
    "pages": "$inputRoot.photos.pages",
    "perpage": $inputRoot.photos.perpage,
    "total": "$inputRoot.photos.total",
    "photo": [
```

```
#foreach($elem in $inputRoot.photos.photo)
      {
        "id": "$elem.id",
        "owner": "$elem.owner",
        "photographer_first_name" : "$elem.photographer_first_name",
        "photographer_last_name" : "$elem.photographer_last_name",
        "secret": "$elem.secret",
        "server": "$elem.server",
        "farm": $elem.farm,
        "title": "$elem.title",
        "ispublic": $elem.ispublic,
        "isfriend": $elem.isfriend,
        "isfamily": $elem.isfamily
      }#if($foreach.hasNext),#end

#end
    ]
  }
}
```

Contoh artikel berita (model API Gateway dan template pemetaan)

Bagian berikut memberikan contoh model dan template pemetaan yang dapat digunakan untuk contoh artikel berita API di API Gateway. Untuk informasi selengkapnya tentang model dan templat pemetaan di API Gateway, lihatPetStore template pemetaan.

Topik

- Data asli (contoh artikel berita)
- Model masukan (contoh artikel berita)
- Template pemetaan masukan (contoh artikel berita)
- Data yang diubah (contoh artikel berita)
- Model keluaran (contoh artikel berita)
- Template pemetaan keluaran (contoh artikel berita)

Data asli (contoh artikel berita)

Berikut ini adalah data JSON asli untuk contoh artikel berita:

```
{
  "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
}
```

Model masukan (contoh artikel berita)

Berikut ini adalah model input yang sesuai dengan data JSON asli untuk contoh artikel berita:

```
{
  "$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" },
          "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" }
  }
}
```

Template pemetaan masukan (contoh artikel berita)

Berikut ini adalah template pemetaan masukan yang sesuai dengan data JSON asli untuk contoh artikel berita:

```
#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
}
```

Data yang diubah (contoh artikel berita)

Berikut ini adalah salah satu contoh bagaimana artikel berita asli contoh data JSON dapat diubah untuk output:

```
{
  "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
}
```

Model keluaran (contoh artikel berita)

Berikut ini adalah model output yang sesuai dengan format data JSON yang diubah:

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

Template pemetaan keluaran (contoh artikel berita)

Berikut ini adalah template pemetaan output yang sesuai dengan format data JSON yang diubah. Variabel template di sini didasarkan pada format data JSON asli, tidak diubah:

```
#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
}
```

Contoh faktur penjualan (model API Gateway dan templat pemetaan)

Bagian berikut memberikan contoh model dan templat pemetaan yang dapat digunakan untuk contoh API faktur penjualan di API Gateway. Untuk informasi selengkapnya tentang model dan templat pemetaan di API Gateway, lihatPetStore template pemetaan.

Topik

- Data asli (contoh faktur penjualan)
- Model input (contoh faktur penjualan)
- Template pemetaan masukan (contoh faktur penjualan)
- Data yang diubah (contoh faktur penjualan)
- Model keluaran (contoh faktur penjualan)
- Template pemetaan keluaran (contoh faktur penjualan)

Data asli (contoh faktur penjualan)

Berikut ini adalah data JSON asli untuk contoh faktur penjualan:

```
{
  "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
}
```

Model input (contoh faktur penjualan)

Berikut ini adalah model input yang sesuai dengan data JSON asli untuk contoh faktur penjualan:

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

Template pemetaan masukan (contoh faktur penjualan)

Berikut ini adalah template pemetaan masukan yang sesuai dengan data JSON asli untuk contoh faktur penjualan:

```
#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

#end
  ],
  "Vendor": {
    "value": "$inputRoot.Vendor.value",
    "name": "$inputRoot.Vendor.name"
  },
  "APRef": {
    "value": "$inputRoot.APRef.value",
    "name": "$inputRoot.APRef.name"
  },
```

```
  "TotalAmt": $inputRoot.TotalAmt
}
```

Data yang diubah (contoh faktur penjualan)

Berikut ini adalah salah satu contoh bagaimana data JSON contoh faktur penjualan asli dapat diubah untuk output:

```
{
  "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
}
```

Model keluaran (contoh faktur penjualan)

Berikut ini adalah model output yang sesuai dengan format data JSON yang diubah:

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

Template pemetaan keluaran (contoh faktur penjualan)

Berikut ini adalah template pemetaan output yang sesuai dengan format data JSON yang diubah. Variabel template di sini didasarkan pada format data JSON asli, tidak diubah:

```
#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

#end
  ],
```

```
  "TotalAmt": $inputRoot.TotalAmt
}
```

Contoh catatan karyawan (model API Gateway dan templat pemetaan)

Bagian berikut memberikan contoh model dan templat pemetaan yang dapat digunakan untuk contoh API rekaman karyawan di API Gateway. Untuk informasi selengkapnya tentang model dan templat pemetaan di API Gateway, lihatPetStore template pemetaan.

Topik

- Data asli (contoh catatan karyawan)
- Model masukan (contoh catatan karyawan)
- Template pemetaan masukan (contoh catatan karyawan)
- Data yang diubah (contoh catatan karyawan)
- Model keluaran (contoh catatan karyawan)
- Template pemetaan keluaran (contoh catatan karyawan)

Data asli (contoh catatan karyawan)

Berikut ini adalah data JSON asli untuk contoh catatan karyawan:

```
{
  "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"
}
```

Model masukan (contoh catatan karyawan)

Berikut ini adalah model input yang sesuai dengan data JSON asli untuk contoh catatan karyawan:

```
{
  "$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" }
              }
            },
            "PrimaryAddr": {
              "type": "object",
              "properties": {
                "Line1": { "type": "string" },
                "City": { "type": "string" },
                "CountrySubDivisionCode": { "type": "string" },
                "PostalCode": { "type": "string" }
              }
            }
          }
        }
      }
    },
    "time": { "type": "string" }
  }
}
```

Template pemetaan masukan (contoh catatan karyawan)

Berikut ini adalah template pemetaan masukan yang sesuai dengan data JSON asli untuk contoh catatan karyawan:

```
#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"
}
```

Data yang diubah (contoh catatan karyawan)

Berikut ini adalah salah satu contoh bagaimana data JSON catatan karyawan asli dapat diubah untuk output:

```
{
  "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"
}
```

Model keluaran (contoh catatan karyawan)

Berikut ini adalah model output yang sesuai dengan format data JSON yang diubah:

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

Template pemetaan keluaran (contoh catatan karyawan)

Berikut ini adalah template pemetaan output yang sesuai dengan format data JSON yang diubah. Variabel template di sini didasarkan pada format data JSON asli, tidak diubah:

```
#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"
}
```

Referensi pemetaan data permintaan dan respons API Amazon API Gateway

Bagian ini menjelaskan cara mengatur pemetaan data dari data permintaan metode API, termasuk data lain yang disimpan dalam `context`, `stage`, atau `util`variabel, ke parameter permintaan integrasi yang sesuai dan dari data respons integrasi, termasuk data lainnya, ke parameter respons metode. Data permintaan metode mencakup parameter permintaan (jalur, string kueri, header) dan isi. Data respons integrasi mencakup parameter respons (header) dan badan. Untuk informasi lebih lanjut tentang menggunakan variabel tahap, lihatReferensi variabel tahap Amazon API Gateway.

Topik

Metode peta meminta data ke parameter permintaan integrasi

Parameter permintaan integrasi, dalam bentuk variabel jalur, string kueri atau header, dapat dipetakan dari parameter permintaan metode yang ditentukan dan payload.

Dalam tabel berikut, *PARAM_NAME*adalah nama parameter permintaan metode dari jenis parameter yang diberikan. Itu harus cocok dengan ekspresi reguler`'^[a-zA-Z0-9._$-]+$]'`. Itu harus didefinisikan sebelum dapat direferensikan. *JSONPath_EXPRESSION*adalah ekspresi JSONPath untuk bidang JSON dari badan permintaan atau respons.

> **Note**
>
> "$"Awalan dihilangkan dalam sintaks ini.

Ekspresi pemetaan data permintaan integrasi

Sumber data yang dipetakan	Ekspresi pemetaan
Jalur permintaan metode	`method.request.path.` *PARAM_NAME*
String kueri permintaan metode	`method.request.querystring.` *PARAM_NAME*

Sumber data yang dipetakan	Ekspresi pemetaan
String kueri permintaan metode multi-nilai	`method.request.multivaluequerystring.` *`PARAM_NAME`*
Header permintaan metode	`method.request.header.` *`PARAM_NAME`*
Header permintaan metode multi-nilai	`method.request.multivalueheader.` *`PARAM_NAME`*
Badan permintaan metode	`method.request.body`
Metode permintaan badan (JsonPath)	`method.request.body.` *`JSONPath_EXPRESSION`* .
Variabel tahap	`stageVariables.` *`VARIABLE_NAME`*
Variabel konteks	`context.`*`VARIABLE_NAME`* yang harus menjadi salah satu variabel konteks yang didukung.
Nilai statis	*`'STATIC_VALUE'`* . *`STATIC_VALUE`* adalah string literal dan harus diapit dalam sepasang tanda kutip tunggal.

Example Pemetaan dari parameter permintaan metode di OpenAPI

Contoh berikut menunjukkan cuplikan OpenAPI yang memetakan:

- header permintaan metode, bernama`methodRequestHeaderParam`, ke dalam parameter jalur permintaan integrasi, bernama `integrationPathParam`
- string kueri permintaan metode multi-nilai, bernama`methodRequestQueryParam`, ke dalam string kueri permintaan integrasi, bernama `integrationQueryParam`

```
...
"requestParameters" : {
```

```
    "integration.request.path.integrationPathParam" :
 "method.request.header.methodRequestHeaderParam",
    "integration.request.querystring.integrationQueryParam" :
 "method.request.multivaluequerystring.methodRequestQueryParam"

}
...
```

Parameter permintaan integrasi juga dapat dipetakan dari bidang di badan permintaan JSON menggunakan ekspresi JSONPath. Tabel berikut menunjukkan ekspresi pemetaan untuk badan permintaan metode dan bidang JSON nya.

Example Pemetaan dari badan permintaan metode di OpenAPI

Contoh berikut menunjukkan cuplikan OpenAPI yang memetakan 1) badan permintaan metode ke header permintaan integrasi, bernama`body-header`, dan 2) bidang JSON dari tubuh, seperti yang diungkapkan oleh ekspresi JSON (`petstore.pets[0].name`, tanpa awalan). `$.`

```
...
"requestParameters" : {

    "integration.request.header.body-header" : "method.request.body",
    "integration.request.path.pet-name" : "method.request.body.petstore.pets[0].name",

}
...
```

Memetakan data respons integrasi ke header respons metode

Parameter header respons metode dapat dipetakan dari header respons integrasi atau badan respons integrasi, `$context` variabel, atau nilai statis.

Ekspresi pemetaan header respons metode

Sumber data yang dipetakan	Ekspresi pemetaan
Header respons integrasi	`integration.response.header.` *PARAM_NAME*
Header respons integrasi	`integration.response.multivalueheader.` *PARAM_NAME*
Badan respons integrasi	`integration.response.body`
Integrasi respon body (JsonPath)	`integration.response.body.` *JSONPath_EXPRESSION*
Variabel tahap	`stageVariables.` *VARIABLE_NAME*
Variabel konteks	`context.`*VARIABLE_NAME* yang harus menjadi salah satu variabel konteks yang didukung.
Nilai statis	*'STATIC_VALUE'* . *STATIC_VALUE* adalah string literal dan harus diapit dalam sepasang tanda kutip tunggal.

Example Pemetaan data dari respons integrasi di OpenAPI

Contoh berikut menunjukkan cuplikan OpenAPI yang memetakan 1) respons integrasi`redirect.url`, bidang JsonPath ke header respons permintaan; dan 2) `location` header respons integrasi ke `x-app-id` header respons metode. `id`

```
...
"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",

}
```

```
...
```

Permintaan peta dan muatan respons antara metode dan integrasi

API Gateway menggunakan mesin Velocity Template Language (VTL) untuk memproses template pemetaan tubuh untuk permintaan integrasi dan respons integrasi. Templat pemetaan menerjemahkan payload permintaan metode ke muatan permintaan integrasi yang sesuai dan menerjemahkan badan respons integrasi ke badan respons metode.

Template VTL menggunakan ekspresi JSONPath, parameter lain seperti memanggil konteks dan variabel tahap, dan fungsi utilitas untuk memproses data JSON.

Jika model didefinisikan untuk menggambarkan struktur data muatan, API Gateway dapat menggunakan model untuk menghasilkan template pemetaan kerangka untuk permintaan integrasi atau respons integrasi. Anda dapat menggunakan template kerangka sebagai bantuan untuk menyesuaikan dan memperluas skrip VTL pemetaan. Namun, Anda dapat membuat template pemetaan dari awal tanpa menentukan model untuk struktur data payload.

Pilih templat pemetaan VTL

API Gateway menggunakan logika berikut untuk memilih template pemetaan, di Velocity Template Language (VTL), untuk memetakan payload dari permintaan metode ke permintaan integrasi yang sesuai atau untuk memetakan payload dari respons integrasi ke respons metode yang sesuai.

Untuk payload permintaan, API Gateway menggunakan nilai `Content-Type` header permintaan sebagai kunci untuk memilih template pemetaan untuk payload permintaan. Untuk payload respons, API Gateway menggunakan nilai `Accept` header permintaan masuk sebagai kunci untuk memilih template pemetaan.

Ketika `Content-Type` header tidak ada dalam permintaan, API Gateway mengasumsikan bahwa nilai defaultnya adalah`application/json`. Untuk permintaan seperti itu, API Gateway digunakan `application/json` sebagai kunci default untuk memilih template pemetaan, jika sudah ditentukan. Jika tidak ada template yang cocok dengan kunci ini, API Gateway meneruskan payload melalui unmapped jika properti PassThroughBehavior disetel ke or. `WHEN_NO_MATCH` `WHEN_NO_TEMPLATES`

Ketika `Accept` header tidak ditentukan dalam permintaan, API Gateway mengasumsikan bahwa nilai defaultnya adalah`application/json`. Dalam kasus ini, API Gateway memilih template pemetaan yang ada `application/json` untuk memetakan payload respons. Jika tidak ada template yang ditentukan`application/json`, API Gateway memilih template pertama yang ada

dan menggunakannya sebagai default untuk memetakan payload respons. Demikian pula, API Gateway menggunakan template pertama yang ada saat nilai `Accept` header yang ditentukan tidak cocok dengan kunci template yang ada. Jika tidak ada template yang ditentukan, API Gateway hanya meneruskan payload respons melalui unmapped.

Misalnya, API memiliki `application/json` template yang ditentukan untuk payload permintaan dan memiliki `application/xml` template yang ditentukan untuk payload respons. Jika klien menetapkan`"Content-Type : application/json"`, dan `"Accept : application/xml"` header dalam permintaan, muatan permintaan dan respons akan diproses dengan templat pemetaan yang sesuai. Jika `Accept:application/xml` header tidak ada, template `application/xml` pemetaan akan digunakan untuk memetakan payload respons. Untuk mengembalikan payload respons yang tidak dipetakan, Anda harus menyiapkan templat kosong untuk. `application/json`

Hanya tipe MIME yang digunakan dari `Content-Type` header `Accept` dan saat memilih template pemetaan. Misalnya, header `"Content-Type: application/json; charset=UTF-8"` akan memiliki template permintaan dengan `application/json` kunci yang dipilih.

Perilaku passthrough integrasi

Dengan integrasi non-proxy, ketika permintaan metode membawa muatan dan `Content-Type` header tidak cocok dengan templat pemetaan tertentu atau tidak ada templat pemetaan yang ditentukan, Anda dapat memilih untuk meneruskan payload permintaan yang disediakan klien melalui permintaan integrasi ke backend tanpa transformasi. Proses ini dikenal sebagai integrasi passthrough.

Untuk integrasi proxy, API Gateway meneruskan seluruh permintaan ke backend Anda, dan Anda tidak memiliki opsi untuk mengubah perilaku passthrough.

Perilaku passthrough aktual dari permintaan masuk ditentukan oleh opsi yang Anda pilih untuk templat pemetaan tertentu, selama pengaturan permintaan integrasi, dan header Jenis Konten yang ditetapkan klien dalam permintaan masuk. Ada tiga opsi:

Bila tidak ada template yang cocok dengan header Content-Type permintaan

Pilih opsi ini jika Anda ingin badan permintaan metode melewati permintaan integrasi ke backend tanpa transformasi ketika jenis konten permintaan metode tidak cocok dengan jenis konten apa pun yang terkait dengan templat pemetaan.

Saat memanggil API Gateway API, Anda memilih opsi ini dengan menetapkan `WHEN_NO_MATCH` sebagai nilai `passthroughBehavior` properti pada Integrasi.

Ketika tidak ada templat yang ditentukan (disarankan)

Pilih opsi ini jika Anda ingin badan permintaan metode melewati permintaan integrasi ke backend tanpa transformasi ketika tidak ada templat pemetaan yang ditentukan dalam permintaan integrasi. Jika templat ditentukan saat opsi ini dipilih, permintaan metode dari jenis konten yang tidak dipetakan akan ditolak dengan respons Jenis Media Tidak Didukung HTTP 415.

Saat memanggil API Gateway API, Anda memilih opsi ini dengan menetapkan `WHEN_NO_TEMPLATES` sebagai nilai `passthroughBehavior` properti pada Integrasi.

Tidak pernah

Pilih opsi ini jika Anda tidak ingin badan permintaan metode melewati permintaan integrasi ke backend tanpa transformasi ketika tidak ada templat pemetaan yang ditentukan dalam permintaan integrasi. Jika templat ditentukan saat opsi ini dipilih, permintaan metode dari jenis konten yang tidak dipetakan akan ditolak dengan respons Jenis Media Tidak Didukung HTTP 415.

Saat memanggil API Gateway API, Anda memilih opsi ini dengan menetapkan `NEVER` sebagai nilai `passthroughBehavior` properti pada Integrasi.

Contoh-contoh berikut menggambarkan kemungkinan perilaku passthrough.

Contoh 1: Satu template pemetaan didefinisikan dalam permintaan integrasi untuk jenis `application/json` konten.

Header tipe konten \ Opsi passthrough yang dipilih	**WHEN_NO_MATCH**	**WHEN_NO_TEMPLATES**	**NEVER**
Tidak ada (default keapplication/json)	Muatan permintaan diubah menggunakan templat.	Muatan permintaan diubah menggunakan templat.	Muatan permintaan diubah menggunakan templat.
`application/json`	Muatan permintaan diubah menggunakan templat.	Muatan permintaan diubah menggunakan templat.	Muatan permintaan diubah menggunakan templat.
`application/xml`	Payload permintaan tidak diubah dan	Permintaan ditolak dengan 415	Permintaan ditolak dengan 415

Header tipe konten \ Opsi passthrough yang dipilih	**WHEN_NO_MATCH**	**WHEN_NO_TEMPLATES**	**NEVER**
	dikirim ke backend apa adanya.	`Unsupported Media Type` respons HTTP.	`Unsupported Media Type` respons HTTP.

Contoh 2: Satu template pemetaan didefinisikan dalam permintaan integrasi untuk jenis `application/xml` konten.

Header tipe konten \ Opsi passthrough yang dipilih	**WHEN_NO_MATCH**	**WHEN_NO_TEMPLATES**	**NEVER**
Tidak ada (default keapplication/json)	Payload permintaan tidak diubah dan dikirim ke backend apa adanya.	Permintaan ditolak dengan `415 Unsupported Media Type` respons HTTP.	Permintaan ditolak dengan `415 Unsupported Media Type` respons HTTP.
`application/json`	Payload permintaan tidak diubah dan dikirim ke backend apa adanya.	Permintaan ditolak dengan `415 Unsupported Media Type` respons HTTP.	Permintaan ditolak dengan `415 Unsupported Media Type` respons HTTP.
`application/xml`	Muatan permintaan diubah menggunakan templat.	Muatan permintaan diubah menggunakan templat.	Muatan permintaan diubah menggunakan templat.

Template pemetaan API Gateway dan referensi variabel pencatatan akses

Bagian ini menyediakan informasi referensi untuk variabel dan fungsi yang didefinisikan Amazon API Gateway untuk digunakan dengan model data, otorisasi, templat pemetaan, dan CloudWatch pencatatan akses. Untuk informasi rinci tentang cara menggunakan variabel dan fungsi ini,

lihatMemahami template pemetaan. Untuk informasi lebih lanjut tentang Velocity Template Language (VTL), lihat Referensi VTL.

Topik

- $contextVariabel untuk model data, otorisasi, templat pemetaan, dan CloudWatch pencatatan akses
- $contextContoh template variabel
- $contextVariabel hanya untuk pencatatan akses
- $inputVariabel
- $inputContoh template variabel
- $stageVariables
- $utilVariabel

> **Note**
>
> Untuk `$method` dan `$integration` variabel, lihatthe section called “Referensi pemetaan data permintaan dan respons”.

`$context`Variabel untuk model data, otorisasi, templat pemetaan, dan CloudWatch pencatatan akses

`$context`Variabel berikut dapat digunakan dalam model data, otorisasi, templat pemetaan, dan logging CloudWatch akses.

Untuk `$context` variabel yang hanya dapat digunakan dalam pencatatan CloudWatch akses, lihatthe section called “`$context`Variabel hanya untuk pencatatan akses”.

Parameter	Deskripsi
`$context.accountId`	ID AWS akun pemilik API.
`$context.apiId`	API Gateway identifier ditetapkan ke API Anda.
`$context.authorizer.claims.`*`property`*	Properti klaim yang dikembalikan dari kumpulan pengguna Amazon Cognito setelah pemanggil metode berhasil diautentikasi.

Parameter	Deskripsi
	Untuk informasi selengkapnya, lihat the section called “Gunakan kumpulan pengguna Amazon Cognito sebagai otorisasi untuk REST API”. **Note** Memanggil `$context.authorizer.claims` mengembalikan null.
`$context.authorizer.principalId`	Identifikasi pengguna utama yang terkait dengan token yang dikirim oleh klien dan dikembalikan dari otorisasi API Gateway Lambda (sebelumnya dikenal sebagai otorisasi khusus). Untuk informasi selengkapnya, lihat the section called “Gunakan otorisasi Lambda”.

Parameter	Deskripsi
`$context.authorizer.` *`property`*	Nilai stringifikasi dari pasangan nilai kunci yang ditentukan dari `context` peta dikembalikan dari fungsi otorisasi API Gateway Lambda. Misalnya, jika otorisasi mengembalikan `context` peta berikut: `"context" : { "key": "value", "numKey": 1, "boolKey": true }` memanggil `$context.authorizer.key` mengembalikan `"value"` string, memanggil `$context.authorizer.numKey` mengembalikan `"1"` string, dan memanggil `$context.authorizer.boolKey` mengembalikan `"true"` string. Untuk informasi selengkapnya, lihat the section called “Gunakan otorisasi Lambda”.
`$context.awsEndpointRequestId`	ID permintaan AWS titik akhir.
`$context.domainName`	Nama domain lengkap yang digunakan untuk memanggil API. Ini harus sama dengan `Host` header yang masuk.
`$context.domainPrefix`	Label pertama dari`$context.domainNam e` .

Parameter	Deskripsi
`$context.error.message`	String yang berisi pesan kesalahan API Gateway. Variabel ini hanya dapat digunakan untuk substitusi variabel sederhana dalam template GatewayResponsepemetaan tubuh, yang tidak diproses oleh mesin Velocity Template Language, dan dalam logging akses. Lihat informasi yang lebih lengkap di the section called “Metrik” dan the section called “Menyiapkan respons gateway untuk menyesuaikan respons kesalahan”.
`$context.error.messageString`	Nilai yang dikutip dari`$context.error.message` , yaitu`"$context.error.message"` .
`$context.error.responseType`	Jenis GatewayResponse. Variabel ini hanya dapat digunakan untuk substitusi variabel sederhana dalam template GatewayResponsepemetaan tubuh, yang tidak diproses oleh mesin Velocity Template Language, dan dalam logging akses. Lihat informasi yang lebih lengkap di the section called “Metrik” dan the section called “Menyiapkan respons gateway untuk menyesuaikan respons kesalahan”.
`$context.error.validationErrorString`	Sebuah string yang berisi pesan kesalahan validasi rinci.
`$context.extendedRequestId`	ID tambahan yang dibuat dan ditetapkan API Gateway ke permintaan API. ID permintaan yang diperluas berisi informasi yang berguna untuk debugging dan pemecahan masalah.

Parameter	Deskripsi
`$context.httpMethod`	Metode HTTP yang digunakan . Nilai yang valid meliputi: `DELETEGET`,,`HEAD`,`OPTIONS`,`PATCH`,`POST`, dan`PUT`.
`$context.identity.accountId`	ID AWS akun yang terkait dengan permintaan.
`$context.identity.apiKey`	Untuk metode API yang memerlukan kunci API, variabel ini adalah kunci API yang terkait dengan permintaan metode. Untuk metode yang tidak memerlukan kunci API, variabel ini adalah null. Untuk informasi selengkapnya, lihat the section called “Paket penggunaan”.
`$context.identity.apiKeyId`	ID kunci API yang terkait dengan permintaan API yang memerlukan kunci API.
`$context.identity.caller`	Pengenal utama penelepon yang menandata ngani permintaan. Didukung untuk sumber daya yang menggunakan otorisasi IAM.

Parameter	Deskripsi
`$context.identity.cognitoAuthenticationProvider`	Daftar penyedia otentikasi Amazon Cognito yang dipisahkan koma yang digunakan oleh penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredenal Amazon Cognito. Misalnya, untuk identitas dari kumpulan pengguna Amazon Cognito, `cognito-idp.` *`region`*`.amazonaws.com/` *`user_pool_id`* `,cognito-idp.` *`region`*`.amazonaws.com/` *`user_pool_id`* `:CognitoSignIn:` *`token subject claim`* Untuk selengkapnya, lihat Menggunakan Identitas Federasi di Panduan Pengembang Amazon Cognito.
`$context.identity.cognitoAuthenticationType`	Jenis otentikasi Amazon Cognito dari penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredenal Amazon Cognito. Nilai yang mungkin termasuk `authenticated` untuk identitas yang diautentikasi dan `unauthenticated` untuk identitas yang tidak diautentikasi.
`$context.identity.cognitoIdentityId`	ID identitas Amazon Cognito dari penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredenal Amazon Cognito.
`$context.identity.cognitoIdentityPoolId`	ID kumpulan identitas Amazon Cognito dari penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredenal Amazon Cognito.
`$context.identity.principalOrgId`	ID AWS organisasi.

Parameter	Deskripsi
`$context.identity.sourceIp`	Alamat IP sumber dari koneksi TCP langsung membuat permintaan ke titik akhir API Gateway.
`$context.identity.clientCert.clientCertPem`	Sertifikat klien yang dikodekan PEM yang disajikan klien selama otentikasi TLS timbal balik. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik. Hadir hanya di log akses jika otentikasi TLS timbal balik gagal.
`$context.identity.clientCert.subjectDN`	Nama yang dibedakan dari subjek sertifikat yang disajikan klien. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik. Hadir hanya di log akses jika otentikasi TLS timbal balik gagal.
`$context.identity.clientCert.issuerDN`	Nama terhormat dari penerbit sertifikat yang disajikan klien. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik. Hadir hanya di log akses jika otentikasi TLS timbal balik gagal.
`$context.identity.clientCert.serialNumber`	Nomor seri sertifikat. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik. Hadir hanya di log akses jika otentikasi TLS timbal balik gagal.
`$context.identity.clientCert.validity.notBefore`	Tanggal sebelum sertifikat tidak valid. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik. Hadir hanya di log akses jika otentikasi TLS timbal balik gagal.

Parameter	Deskripsi
`$context.identity.clientCert.validity.notAfter`	Tanggal setelah sertifikat tidak valid. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik. Hadir hanya di log akses jika otentikasi TLS timbal balik gagal.
`$context.identity.user`	Pengidentifikasi utama pengguna yang akan diotorisasi terhadap akses sumber daya. Didukung untuk sumber daya yang menggunakan otorisasi IAM.
`$context.identity.userAgent`	`User-Agent` Header pemanggil API.
`$context.identity.userArn`	Nama Sumber Daya Amazon (ARN) dari pengguna efektif yang diidentifikasi setelah otentikasi. Untuk informasi selengkapnya, lihat https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html.
`$context.path`	Jalur permintaan. Misalnya, untuk URL permintaan non-proxy dari`https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/root/child` , `$context.path` nilainya adalah`/{stage}/root/child` .

Parameter	Deskripsi
`$context.protocol`	Protokol permintaan, misalnya,`HTTP/1.1`. **Note** API Gateway API dapat menerima permintaan HTTP/2, tetapi API Gateway mengirimkan permintaan ke integrasi backend menggunakan HTTP/1.1. Akibatnya, protokol permintaan dicatat sebagai HTTP/1.1 bahkan jika klien mengirim permintaan yang menggunakan HTTP/2.
`$context.requestId`	ID untuk permintaan. Klien dapat mengganti ID permintaan ini. Gunakan `$context.extendedRequestId` untuk ID permintaan unik yang dihasilkan API Gateway.
`$context.requestOverride.header.` *`header_name`*	Header permintaan menimpa. Jika parameter ini didefinisikan, ini berisi header yang akan digunakan alih-alih Header HTTP yang didefinisikan di panel Permintaan Integrasi. Untuk informasi selengkapnya, lihat Menggunakan template pemetaan untuk mengganti parameter permintaan dan respons API serta kode status.
`$context.requestOverride.path.` *`path_name`*	Jalur permintaan menimpa. Jika parameter ini ditentukan, parameter ini berisi jalur permintaan yang akan digunakan, bukan Parameter Jalur URL yang ditentukan di panel Permintaan Integrasi. Untuk informasi selengkapnya, lihat Menggunakan template pemetaan untuk mengganti parameter permintaan dan respons API serta kode status.

Parameter	Deskripsi
`$context.requestOverride.querystring.` *querystring_name*	Permintaan query string override. Jika parameter ini didefinisikan, parameter ini berisi string permintaan permintaan yang akan digunakan, bukan Parameter String Kueri URL yang didefinisikan di panel Permintaan Integrasi. Untuk informasi selengkapnya, lihat Menggunakan template pemetaan untuk mengganti parameter permintaan dan respons API serta kode status.
`$context.responseOverride.header.` *header_name*	Response header override. Jika parameter ini didefinisikan, ini berisi header yang akan dikembalikan, bukan header Response yang didefinisikan sebagai pemetaan Default di panel Integration Response. Untuk informasi selengkapnya, lihat Menggunakan template pemetaan untuk mengganti parameter permintaan dan respons API serta kode status.
`$context.responseOverride.status`	Kode status respons menimpa. Jika parameter ini didefinisikan, ini berisi kode status yang akan dikembalikan, bukan status respons Metode yang didefinisikan sebagai pemetaan Default di panel Respons Integrasi. Untuk informasi selengkapnya, lihat Menggunakan template pemetaan untuk mengganti parameter permintaan dan respons API serta kode status.
`$context.requestTime`	Waktu permintaan yang diformat CLF (). `dd/MMM/yyyy:HH:mm:ss +-hhmm`
`$context.requestTimeEpoch`	Waktu permintaan yang diformat Epoch, dalam milidetik.
`$context.resourceId`	Pengidentifikasi yang ditetapkan API Gateway ke sumber daya Anda.

Parameter	Deskripsi
`$context.resourcePath`	Jalan menuju sumber daya Anda. Misalnya, untuk URI permintaan non-proxy dari`https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/root/child` , `$context.resourcePath` Nilainya adalah`/root/child` . Untuk informasi selengkapnya, lihat Tutorial: Membangun REST API dengan integrasi non-proxy HTTP.
`$context.stage`	Tahap penerapan permintaan API (misalnya, `Beta` atau`Prod`).
`$context.wafResponseCode`	Tanggapan yang diterima dari AWS WAF: `WAF_ALLOW` atau`WAF_BLOCK` . Tidak akan diatur jika tahap tidak terkait dengan ACL web. Untuk informasi selengkapnya, lihat the section called “AWS WAF”.
`$context.webaclArn`	ARN lengkap dari ACL web yang digunakan untuk memutuskan apakah akan mengizinkan atau memblokir permintaan. Tidak akan diatur jika tahap tidak terkait dengan ACL web. Untuk informasi selengkapnya, lihat the section called “AWS WAF”.

`$context`Contoh template variabel

Anda mungkin ingin menggunakan `$context` variabel dalam template pemetaan jika metode API Anda meneruskan data terstruktur ke backend yang mengharuskan data berada dalam format tertentu.

Contoh berikut menunjukkan template pemetaan yang memetakan `$context` variabel masuk ke variabel backend dengan nama yang sedikit berbeda dalam payload permintaan integrasi:

> **Note**
>
> Perhatikan bahwa salah satu variabel adalah kunci API. Contoh ini mengasumsikan bahwa metode telah mengaktifkan “require API key”.

```
{
    "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"
}
```

`$context`Variabel hanya untuk pencatatan akses

`$context`Variabel berikut hanya tersedia untuk logging akses. Untuk informasi selengkapnya, lihat the section called “CloudWatch log”. (Untuk WebSocket API, lihatthe section called “Metrik”.)

Parameter	Deskripsi
`$context.authorize.error`	Pesan kesalahan otorisasi.
`$context.authorize.latency`	Latensi otorisasi di ms.
`$context.authorize.status`	Kode status dikembalikan dari upaya otorisasi.
`$context.authorizer.error`	Pesan kesalahan dikembalikan dari otorisasi.
`$context.authorizer.integrationLatency`	Latensi otorisasi di ms.

Parameter	Deskripsi
`$context.authorizer.integrationStatus`	Kode status dikembalikan dari otorisasi Lambda.
`$context.authorizer.latency`	Latensi otorisasi di ms.
`$context.authorizer.requestId`	ID permintaan AWS titik akhir.
`$context.authorizer.status`	Kode status dikembalikan dari otorisasi.
`$context.authenticate.error`	Pesan kesalahan dikembalikan dari upaya otentikasi.
`$context.authenticate.latency`	Latensi otentikasi di ms.
`$context.authenticate.status`	Kode status dikembalikan dari upaya otentikasi.
`$context.customDomain.basePathMatched`	Jalur untuk pemetaan API yang cocok dengan permintaan masuk. Berlaku ketika klien menggunakan nama domain khusus untuk mengakses API. Misalnya jika klien mengirim permintaan kehttps://api.example.com/v1/orders/1234 , dan permintaan tersebut cocok dengan pemetaan API dengan jalurv1/orders , nilainya adalahv1/orders . Untuk mempelajari selengkapnya, lihat the section called “Pemetaan API”.
`$context.integration.error`	Pesan kesalahan dikembalikan dari integrasi.
`$context.integration.integrationStatus`	Untuk integrasi proxy Lambda, kode status dikembalikan dari AWS Lambda, bukan dari kode fungsi Lambda backend.
`$context.integration.latency`	Latensi integrasi dalam ms. Setara dengan$context.integrationLatency .

Parameter	Deskripsi
`$context.integration.requestId`	ID permintaan AWS titik akhir. Setara dengan`$context.awsEndpointRequestId` .
`$context.integration.status`	Kode status dikembalikan dari integrasi. Untuk integrasi proxy Lambda, ini adalah kode status yang dikembalikan oleh kode fungsi Lambda Anda.
`$context.integrationLatency`	Latensi integrasi dalam ms.
`$context.integrationStatus`	Untuk integrasi proxy Lambda, parameter ini mewakili kode status yang dikembalikan dari AWS Lambda, bukan dari kode fungsi Lambda backend.
`$context.responseLatency`	Latensi respons dalam ms.
`$context.responseLength`	Panjang payload respon dalam byte.
`$context.status`	Status respons metode.
`$context.waf.error`	Pesan kesalahan dikembalikan dari AWS WAF.
`$context.waf.latency`	AWS WAF Latensi dalam ms.
`$context.waf.status`	Kode status dikembalikan dari AWS WAF.
`$context.xrayTraceId`	ID jejak untuk jejak X-Ray. Untuk informasi selengkapnya, lihat the section called “Menyiapkan AWS X-Ray”.

`$input`Variabel

`$input`Variabel mewakili payload permintaan metode dan parameter yang akan diproses oleh template pemetaan. Ini menyediakan empat fungsi:

Variabel dan fungsi	Deskripsi
`$input.body`	Mengembalikan payload permintaan mentah sebagai string.
`$input.json(x)`	Fungsi ini mengevaluasi ekspresi JSONPath dan mengembalikan hasil sebagai string JSON. Misalnya, `$input.json('$.pets')` mengembalikan string JSON yang mewakili `pets` struktur. Untuk informasi selengkapnya tentang JsonPath, lihat JsonPath atau JsonPath for Java.
`$input.params()`	Mengembalikan peta dari semua parameter permintaan. Kami menyarankan Anda menggunakan `$util.escapeJavaScript` untuk membersihkan hasilnya untuk menghindari potensi serangan injeksi. Untuk kontrol penuh sanitasi permintaan, gunakan integrasi proxy tanpa templat dan tangani sanitasi permintaan dalam integrasi Anda.
`$input.params(x)`	Mengembalikan nilai parameter permintaan metode dari path, query string, atau nilai header (dicari dalam urutan itu), diberikan string `x` nama parameter. Kami menyarankan Anda menggunakan `$util.escapeJavaScript` untuk membersihkan parameter untuk menghindari potensi serangan injeksi. Untuk kontrol penuh sanitasi parameter, gunakan integrasi proxy tanpa templat dan tangani sanitasi permintaan dalam integrasi Anda.
`$input.path(x)`	Mengambil ekspresi string JSONPath (`x`) dan mengembalikan representasi objek JSON

Variabel dan fungsi	Deskripsi
	dari hasil. Ini memungkinkan Anda untuk mengakses dan memanipulasi elemen payload secara native di Apache Velocity Template Language (VTL). Misalnya, jika ekspresi `$input.path('$.pets')` mengembalikan objek seperti ini: `[ { "id": 1, "type": "dog", "price": 249.99 }, { "id": 2, "type": "cat", "price": 124.99 }, { "id": 3, "type": "fish", "price": 0.99 } ]` `$input.path('$.pets').count()` akan kembali`"3"`. Untuk informasi selengkapnya tentang JsonPath, lihat JsonPath atau JsonPath for Java.

`$input`Contoh template variabel

Contoh template pemetaan parameter

Contoh pemetaan parameter berikut meneruskan semua parameter, termasuk,, dan `path` `querystring`header, hingga ke titik akhir integrasi melalui payload 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
  }
}
```

Akibatnya, template pemetaan ini mengeluarkan semua parameter permintaan dalam muatan seperti yang diuraikan sebagai berikut:

```
{
  "params" : {
     "path" : {
       "path_name" : "path_value",
       ...
     }
     "header" : {
       "header_name" : "header_value",
       ...
     }
     "querystring" : {
       "querystring_name" : "querystring_value",
       ...
     }
   }
}
```

Anda mungkin ingin menggunakan `$input` variabel untuk mendapatkan string kueri dan badan permintaan dengan atau tanpa menggunakan model. Anda mungkin juga ingin memasukkan parameter dan payload, atau subbagian payload, ke dalam fungsi Lambda Anda. Contoh berikut menunjukkan cara melakukan hal ini.

Contoh template pemetaan JSON menggunakan **`$input`**

Contoh berikut menunjukkan bagaimana menggunakan pemetaan untuk membaca nama dari string query dan kemudian menyertakan seluruh badan POST dalam elemen:

```
{
    "name" : "$input.params('name')",
    "body" : $input.json('$')
}
```

Jika input JSON berisi karakter unescaped yang tidak dapat diuraikan oleh JavaScript, respons 400 dapat dikembalikan. Menerapkan `$util.escapeJavaScript($input.json('$'))` di atas akan memastikan bahwa input JSON dapat diurai dengan benar.

Contoh template pemetaan menggunakan **`$input`**

Contoh berikut menunjukkan bagaimana untuk meneruskan ekspresi JsonPath ke metode. `json()` Anda juga dapat membaca properti tertentu dari objek badan permintaan Anda dengan menggunakan titik`(.)`, diikuti dengan nama properti Anda:

```
{
    "name" : "$input.params('name')",
    "body" : $input.json('$.mykey')
}
```

Jika payload permintaan metode berisi karakter unescaped yang tidak dapat diuraikan JavaScript, Anda mungkin mendapatkan respons. `400` Dalam hal ini, Anda perlu memanggil `$util.escapeJavaScript()` fungsi dalam template pemetaan, seperti yang ditunjukkan sebagai berikut:

```
{
    "name" : "$input.params('name')",
    "body" : "$util.escapeJavaScript($input.json('$.mykey'))"
}
```

Contoh permintaan dan respons menggunakan **`$input`**

Berikut adalah contoh yang menggunakan ketiga fungsi:

Templat permintaan:

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

Tanggapan:

```
{
  "id": "abc",
  "count": "3",
  "things": {
    "1": {},
    "2": {},
    "3": {}
  }
}
```

Untuk contoh pemetaan lainnya, lihatMemahami template pemetaan.

`$stageVariables`

Variabel tahap dapat digunakan dalam template pemetaan dan pemetaan parameter dan sebagai placeholder di ARN dan URL yang digunakan dalam integrasi metode. Untuk informasi selengkapnya, lihat the section called “Mengatur variabel tahap”.

Sintaks	Deskripsi
`$stageVariables. <variable_name>`	<variable_name>merupakan nama variabel tahap.
`$stageVariables[' <variable _name> ']`	<variable_name>mewakili setiap nama variabel tahap.
`${stageVariables['  <variable _name> ']}`	<variable_name>mewakili setiap nama variabel tahap.

`$util`Variabel

`$util`Variabel berisi fungsi utilitas untuk digunakan dalam template pemetaan.

> **Note**
>
> Kecuali ditentukan lain, set karakter default adalah UTF-8.

Fungsi	Deskripsi
`$util.escapeJavaScript()`	Melarikan diri dari karakter dalam string menggunakan aturan JavaScript string. **Note** Fungsi ini akan mengubah tanda kutip tunggal biasa (`'`) menjadi yang keluar (`\'`). Namun, tanda kutip tunggal yang lolos tidak valid di JSON. Jadi, ketika output dari fungsi ini digunakan dalam properti JSON, Anda harus mengubah tanda kutip tunggal yang diloloskan (`\'`) kembali ke tanda kutip tunggal biasa (`'`). Ini ditunjukkan dalam contoh berikut:

Fungsi	Deskripsi
	<pre>"input" : "$util.escapeJavaScript(data).replaceAll("\\'",'''')"</pre>

`$util.parseJson()`	Mengambil “stringified” JSON dan mengembalikan representasi objek dari hasilnya. Anda dapat menggunakan hasil dari fungsi ini untuk mengakses dan memanipulasi elemen payload secara native di Apache Velocity Template Language (VTL). Misalnya, jika Anda memiliki muatan berikut:
	<pre>{"errorMessage":"{\"key1\":\"var1\",\"key2\":{\"arr\":[1,2,3]}}"}</pre>dan gunakan template pemetaan berikut
	<pre>#set ($errorMessageObj = $util.parseJson($input.path('$.errorMessage')))
	{
	   "errorMessageObjKey2ArrVal" : $errorMessageObj.key2.arr[0]
	}</pre>Anda akan mendapatkan output sebagai berikut:
	<pre>{
	   "errorMessageObjKey2ArrVal" : 1
	}</pre>

`$util.urlEncode()`	Mengkonversi string ke dalam format “aplikasi/x-www-form-urlencoded”.

Fungsi	Deskripsi
`$util.urlDecode()`	Mendekode string “aplikasi/x-www-form-urlencoded”.
`$util.base64Encode()`	Mengkodekan data ke dalam string yang dikodekan base64.
`$util.base64Decode()`	Mendekode data dari string yang dikodekan base64.

Tanggapan Gateway di API Gateway

Respons gateway diidentifikasi oleh tipe respons yang ditentukan oleh API Gateway. Respons terdiri dari kode status HTTP, satu set header tambahan yang ditentukan oleh pemetaan parameter, dan payload yang dihasilkan oleh template pemetaan non-VTL.

Di API Gateway REST API, respons gateway diwakili oleh GatewayResponse. Di OpenAPI, `GatewayResponse` instance dijelaskan oleh ekstensi x-amazon-apigateway-gateway-responses.gatewayResponse.

Untuk mengaktifkan respons gateway, Anda menyiapkan respons gateway untuk jenis respons yang didukung di API level. Setiap kali API Gateway menampilkan respons jenis ini, templat pemetaan header dan pemetaan muatan yang ditentukan dalam respons gateway diterapkan untuk mengembalikan hasil yang dipetakan ke pemanggil API.

Di bagian berikut, kami menunjukkan cara mengatur respons gateway dengan menggunakan konsol API Gateway dan API Gateway REST API.

Menyiapkan respons gateway untuk menyesuaikan respons kesalahan

Jika API Gateway gagal memproses permintaan yang masuk, ia mengembalikan respons kesalahan ke klien tanpa meneruskan permintaan ke backend integrasi. Secara default, respons kesalahan berisi pesan kesalahan deskriptif singkat. Misalnya, jika Anda mencoba memanggil operasi pada sumber daya API yang tidak ditentukan, Anda menerima respons kesalahan dengan `{ "message": "Missing Authentication Token" }` pesan tersebut. Jika Anda baru mengenal API Gateway, Anda mungkin merasa sulit untuk memahami apa yang sebenarnya salah.

Untuk beberapa respons kesalahan, API Gateway memungkinkan penyesuaian oleh pengembang API untuk mengembalikan respons dalam format yang berbeda. `Missing Authentication Token`Misalnya, Anda dapat menambahkan petunjuk ke payload respons asli dengan kemungkinan penyebabnya, seperti dalam contoh ini:. `{"message":"Missing Authentication Token", "hint":"The HTTP method or resources may not be supported."}`

Saat API Anda memediasi antara pertukaran eksternal dan AWS Cloud, Anda menggunakan templat pemetaan VTL untuk permintaan integrasi atau respons integrasi untuk memetakan payload dari satu format ke format lainnya. Namun, template pemetaan VTL hanya berfungsi untuk permintaan yang valid dengan tanggapan yang berhasil.

Untuk permintaan yang tidak valid, API Gateway melewati integrasi sama sekali dan mengembalikan respons kesalahan. Anda harus menggunakan kustomisasi untuk membuat respons kesalahan dalam format yang sesuai dengan pertukaran. Di sini, kustomisasi diberikan dalam template pemetaan non-VTL yang hanya mendukung substitusi variabel sederhana.

Menggeneralisasi respons kesalahan yang dihasilkan API Gateway terhadap respons apa pun yang dihasilkan oleh API Gateway, kami menyebutnya sebagai respons gateway. Ini membedakan respons yang dihasilkan API Gateway dari respons integrasi. Template pemetaan respons gateway dapat mengakses nilai `$context` variabel dan nilai `$stageVariables` properti, serta parameter permintaan metode, dalam bentuk. `method.request.`*`param-position`*`.`*`param-name`*

Untuk informasi lebih lanjut tentang `$context` variabel, lihat$contextVariabel untuk model data, otorisasi, templat pemetaan, dan CloudWatch pencatatan akses. Untuk informasi selengkapnya tentang `$stageVariables`, lihat `$stageVariables`. Untuk informasi selengkapnya tentang parameter permintaan metode, lihatthe section called “$inputVariabel”.

Topik

- Menyiapkan respons gateway untuk REST API menggunakan konsol API Gateway
- Siapkan respons gateway menggunakan API Gateway REST API
- Siapkan kustomisasi respons gateway di OpenAPI
- Jenis respons gateway

Menyiapkan respons gateway untuk REST API menggunakan konsol API Gateway

Untuk menyesuaikan respons gateway menggunakan konsol API Gateway

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.

2. Pilih REST API.
3. Di panel navigasi utama, pilih Respons Gateway.
4. Pilih jenis respons, lalu pilih Edit. Dalam panduan ini, kami menggunakan token otentikasi Hilang sebagai contoh.
5. Anda dapat mengubah kode Status yang dihasilkan API Gateway untuk menampilkan kode status berbeda yang memenuhi persyaratan API Anda. Dalam contoh ini, kustomisasi mengubah kode status dari default (`403`) menjadi `404` karena pesan kesalahan ini terjadi ketika klien memanggil sumber daya yang tidak didukung atau tidak valid yang dapat dianggap sebagai tidak ditemukan.
6. Untuk mengembalikan header khusus, pilih Tambahkan header respons di bawah Header respons. Untuk tujuan ilustrasi, kami menambahkan header khusus berikut:

```
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
```

   Dalam pemetaan header sebelumnya, nama domain statis (`'a.b.c'`) dipetakan ke `Allow-Control-Allow-Origin` header untuk memungkinkan akses CORS ke API; header permintaan input `x-amzn-RequestId` dipetakan ke `request-id` dalam respons; variabel `petId` jalur permintaan yang masuk dipetakan ke `request-path` header dalam respons; dan parameter q kueri dari permintaan asli dipetakan ke header respons. `request-query`
7. Di bawah Template Response, simpan `application/json` untuk Jenis Konten dan masukkan template pemetaan tubuh berikut di editor badan Template:

```
{
     "message":"$context.error.messageString",
     "type": "$context.error.responseType",
     "statusCode": "'404'",
     "stage": "$context.stage",
     "resourcePath": "$context.resourcePath",
     "stageVariables.a": "$stageVariables.a"
}
```

   Contoh ini menunjukkan cara memetakan `$context` dan `$stageVariables` properti ke properti badan respons gateway.
8. Pilih Save changes (Simpan perubahan).

9. Menerapkan API ke tahap baru atau yang sudah ada.

Uji respons gateway Anda dengan memanggil perintah CURL berikut, dengan asumsi URL pemanggilan metode API yang sesuai adalah: `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
```

Karena parameter string kueri tambahan q=1 tidak kompatibel dengan API, kesalahan dikembalikan untuk memicu respons gateway yang ditentukan. Anda harus mendapatkan respons gateway yang mirip dengan yang berikut ini:

```
> GET /custErr/pets/5?q=1 HTTP/1.1
Host: o81lxisefl.execute-api.us-east-1.amazonaws.com
User-Agent: curl/7.51.0
Accept: */*

HTTP/1.1 404 Not Found
Content-Type: application/json
Content-Length: 334
Connection: keep-alive
Date: Tue, 02 May 2017 03:15:47 GMT
x-amzn-RequestId: 123344566
Access-Control-Allow-Origin: a.b.c
x-amzn-ErrorType: MissingAuthenticationTokenException
header-1: static
x-request-query: 1
x-request-path: 5
X-Cache: Error from cloudfront
Via: 1.1 441811a054e8d055b893175754efd0c3.cloudfront.net (CloudFront)
X-Amz-Cf-Id: nNDR-fX4csbRoAgtQJ16u0rTDz9FZWT-Mk93KgoxnfzDlTUh3flmzA==

{
     "message":"Missing Authentication Token",
     "type": MISSING_AUTHENTICATION_TOKEN,
     "statusCode": '404',
     "stage": custErr,
     "resourcePath": /pets/{petId},
     "stageVariables.a": a
}
```

Contoh sebelumnya mengasumsikan bahwa backend API adalah Pet Store dan API memiliki variabel stage,, didefinisikan. a

Siapkan respons gateway menggunakan API Gateway REST API

Sebelum menyesuaikan respons gateway menggunakan API Gateway REST API, Anda harus sudah membuat API dan telah memperoleh pengenalnya. Untuk mengambil pengenal API, Anda dapat mengikuti relasi tautan restapi:gateway-response dan memeriksa hasilnya.

Untuk menyesuaikan respons gateway menggunakan API Gateway REST API

1. Untuk menimpa seluruh GatewayResponseinstance, panggil tindakan gatewayresponse:put. Tentukan responseType yang diinginkan dalam parameter jalur URL, dan berikan dalam permintaan payload pemetaan StatusCode, ResponseParameters, dan ResponseTemplates.
2. Untuk memperbarui bagian dari `GatewayResponse` instance, panggil tindakan gatewayresponse:update. Tentukan parameter jalur URL yang diinginkan`responseType`, dan berikan dalam permintaan payload `GatewayResponse` properti individual yang Anda inginkan—misalnya, atau `responseParameters` pemetaan. `responseTemplates`

Siapkan kustomisasi respons gateway di OpenAPI

Anda dapat menggunakan `x-amazon-apigateway-gateway-responses` ekstensi di tingkat root API untuk menyesuaikan respons gateway di OpenAPI. Definisi OpenAPI berikut menunjukkan contoh untuk menyesuaikan GatewayResponsetipe. `MISSING_AUTHENTICATION_TOKEN`

```
  "x-amazon-apigateway-gateway-responses": {
    "MISSING_AUTHENTICATION_TOKEN": {
      "statusCode": 404,
      "responseParameters": {
        "gatewayresponse.header.x-request-path": "method.input.params.petId",
        "gatewayresponse.header.x-request-query": "method.input.params.q",
        "gatewayresponse.header.Access-Control-Allow-Origin": "'a.b.c'",
        "gatewayresponse.header.x-request-header": "method.input.params.Accept"
      },
      "responseTemplates": {
        "application/json": "{\n     \"message\": $context.error.messageString,\n
    \"type\":  \"$context.error.responseType\",\n     \"stage\":  \"$context.stage
\",\n     \"resourcePath\":  \"$context.resourcePath\",\n     \"stageVariables.a\":
 \"$stageVariables.a\",\n     \"statusCode\": \"'404'\"\n}"
      }
```

```
    }
```

Dalam contoh ini, kustomisasi mengubah kode status dari default (`403`) menjadi`404`. Ini juga menambah respons gateway empat parameter header dan satu template pemetaan tubuh untuk jenis `application/json` media.

Jenis respons gateway

API Gateway memaparkan respons gateway berikut untuk penyesuaian oleh pengembang API.

Jenis respons gateway	Kode status default	Deskripsi
`ACCESS_DENIED`	`403`	Respons gateway untuk kegagalan otorisasi—misalnya , saat akses ditolak oleh otorisasi khusus atau Amazon Cognito. Jika tipe respons tidak ditentukan, respons ini default ke tipe. `DEFAULT_4XX`
`API_CONFIGURATION_ERROR`	`500`	Respons gateway untuk konfigurasi API yang tidak valid—termasuk saat alamat titik akhir yang tidak valid dikirimkan, saat decoding base64 gagal pada data biner saat dukungan biner diberlakukan, atau saat pemetaan respons integrasi tidak dapat cocok dengan templat apa pun dan tidak ada templat default yang dikonfigurasi. Jika tipe respons tidak ditentukan, respons ini default ke tipe. `DEFAULT_5XX`

Jenis respons gateway	Kode status default	Deskripsi
AUTHORIZER_CONFIGURATION_ERROR	500	Respons gateway karena gagal terhubung ke otorisasi khusus atau Amazon Cognito. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_5XX
AUTHORIZER_FAILURE	500	Respons gateway saat otorisasi khusus atau Amazon Cognito gagal mengauten tikasi pemanggil. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_5XX
BAD_REQUEST_PARAMETERS	400	Respons gateway ketika parameter permintaan tidak dapat divalidasi sesuai dengan validator permintaan yang diaktifkan. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_4XX
BAD_REQUEST_BODY	400	Respons gateway saat badan permintaan tidak dapat divalidasi sesuai dengan validator permintaan yang diaktifkan. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_4XX

Jenis respons gateway	Kode status default	Deskripsi
DEFAULT_4XX	Nol	Respons gateway default untuk jenis respons yang tidak ditentukan dengan kode status. 4XX Mengubah kode status respons gateway fallback ini mengubah kode status semua 4XX respons lain ke nilai baru. Menyetel ulang kode status ini ke null mengembalikan kode status dari semua 4XX respons lain ke nilai aslinya. **Note** AWS WAFtanggapan khusus lebih diutamakan daripada respons gateway khusus.
DEFAULT_5XX	Nol	Respons gateway default untuk jenis respons yang tidak ditentukan dengan kode status. 5XX Mengubah kode status respons gateway fallback ini mengubah kode status semua 5XX respons lain ke nilai baru. Menyetel ulang kode status ini ke null mengembalikan kode status dari semua 5XX respons lain ke nilai aslinya.

Jenis respons gateway	Kode status default	Deskripsi
EXPIRED_TOKEN	403	Respons gateway untuk kesalahan kedaluwarsa token AWS otentikasi. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_4XX
INTEGRATION_FAILURE	504	Respons gateway untuk kesalahan integrasi gagal. Jika tipe respons tidak ditentuka n, respons ini default ke tipe. DEFAULT_5XX
INTEGRATION_TIMEOUT	504	Respons gateway untuk kesalahan waktu integrasi habis. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_5XX
INVALID_API_KEY	403	Respons gateway untuk kunci API tidak valid yang dikirimkan untuk metode yang memerlukan kunci API. Jika tipe respons tidak ditentuka n, respons ini default ke tipe. DEFAULT_4XX
INVALID_SIGNATURE	403	Respons gateway untuk kesalahan AWS tanda tangan yang tidak valid. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_4XX

Jenis respons gateway	Kode status default	Deskripsi
MISSING_AUTHENTICATION_TOKEN	403	Respons gateway untuk kesalahan token otentikasi yang hilang, termasuk kasus ketika klien mencoba memanggil metode atau sumber daya API yang tidak didukung. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_4XX
QUOTA_EXCEEDED	429	Respons gateway untuk kuota paket penggunaan melebihi kesalahan. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_4XX
REQUEST_TOO_LARGE	413	Respons gateway untuk permintaan kesalahan terlalu besar. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_4XX
RESOURCE_NOT_FOUND	404	Respons gateway ketika API Gateway tidak dapat menemukan sumber daya yang ditentukan setelah permintaan API melewati otentikasi dan otorisasi, kecuali untuk autentikasi dan otorisasi kunci API. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_4XX

Jenis respons gateway	Kode status default	Deskripsi
THROTTLED	429	Respons gateway saat batas pelambatan tingkat rencana, metode, tahap, atau akun terlampaui. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_4XX
UNAUTHORIZED	401	Respons gateway saat otorisasi khusus atau Amazon Cognito gagal mengauten tikasi pemanggil.
UNSUPPORTED_MEDIA_TYPE	415	Respons gateway saat payload adalah tipe media yang tidak didukung, jika perilaku passthrough yang ketat diaktifkan. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_4XX
WAF_FILTERED	403	Respons gateway saat permintaan diblokir olehAWS WAF. Jika tipe respons tidak ditentukan, respons ini default ke tipe. DEFAULT_4XX **Note** AWS WAFtanggapan khusus lebih diutamakan daripada respons gateway khusus.

Mengaktifkan CORS untuk sumber daya REST API

Cross-origin resource sharing (CORS) adalah fitur keamanan browser yang membatasi permintaan HTTP lintas asal yang dimulai dari skrip yang berjalan di browser.

Menentukan apakah akan mengaktifkan dukungan CORS

Permintaan HTTP lintas asal adalah permintaan yang dibuat untuk:

- Domain yang berbeda (misalnya, dari `example.com` ke `amazondomains.com`)
- Subdomain yang berbeda (misalnya, dari `example.com` ke `petstore.example.com`)
- Port yang berbeda (misalnya, dari `example.com` ke `example.com:10777`)
- Protokol yang berbeda (misalnya, dari `https://example.com` ke `http://example.com`)

Jika Anda tidak dapat mengakses API dan menerima pesan kesalahan yang berisi `Cross-Origin Request Blocked`, Anda mungkin perlu mengaktifkan CORS.

Permintaan HTTP cross-origin dapat dibagi menjadi dua jenis: permintaan sederhana dan permintaan non-sederhana.

Mengaktifkan CORS untuk permintaan sederhana

Permintaan HTTP sederhana jika semua kondisi berikut benar:

- Ini dikeluarkan terhadap sumber daya API yang hanya mengizinkan `GET`, `HEAD`, dan `POST` permintaan.
- Jika itu adalah permintaan `POST` metode, itu harus menyertakan `Origin` header.
- Jenis konten payload permintaan adalah `text/plain`, `multipart/form-data`, atau `application/x-www-form-urlencoded`.
- Permintaan tidak berisi header khusus.
- Persyaratan tambahan apa pun yang tercantum dalam dokumentasi Mozilla CORS untuk permintaan sederhana.

Untuk permintaan `POST` metode lintas asal sederhana, respons dari sumber daya Anda harus menyertakan header `Access-Control-Allow-Origin: '*'` atau `Access-Control-Allow-Origin:'origin'`.

Semua permintaan HTTP lintas asal lainnya adalah permintaan non-sederhana.

Mengaktifkan CORS untuk permintaan yang tidak sederhana

Jika sumber daya API Anda menerima permintaan yang tidak sederhana, Anda harus mengaktifkan dukungan CORS tambahan tergantung pada jenis integrasi Anda.

Mengaktifkan CORS untuk integrasi non-proxy

Untuk integrasi ini, protokol CORS mengharuskan browser untuk mengirim permintaan preflight ke server dan menunggu persetujuan (atau permintaan kredensional) dari server sebelum mengirim permintaan yang sebenarnya. Anda harus mengonfigurasi API Anda untuk mengirim respons yang sesuai ke permintaan preflight.

Untuk membuat respons preflight:

1. Buat `OPTIONS` metode dengan integrasi tiruan.
2. Tambahkan header respons berikut ke respons metode 200:
   - `Access-Control-Allow-Headers`
   - `Access-Control-Allow-Methods`
   - `Access-Control-Allow-Origin`
3. Masukkan nilai untuk header respons. Untuk mengizinkan semua asal, semua metode, dan header umum, gunakan nilai header berikut:
   - `Access-Control-Allow-Headers: 'Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token'`
   - `Access-Control-Allow-Methods: '*'`
   - `Access-Control-Allow-Origin: '*'`

Setelah membuat permintaan preflight, Anda harus mengembalikan `Access-Control-Allow-Origin:` *`'origin'`* header `Access-Control-Allow-Origin: '*'` or untuk semua metode yang mendukung CORS untuk setidaknya semua 200 tanggapan.

Mengaktifkan CORS untuk integrasi non-proxy menggunakan AWS Management Console

Anda dapat menggunakan AWS Management Console untuk mengaktifkan CORS. API Gateway membuat `OPTIONS` metode dan menambahkan `Access-Control-Allow-Origin` header ke

respons integrasi metode yang ada. Ini tidak selalu berhasil, dan terkadang Anda perlu memodifikasi respons integrasi secara manual untuk mengembalikan `Access-Control-Allow-Origin` header untuk semua metode berkemampuan CORS untuk setidaknya semua 200 respons.

Mengaktifkan dukungan CORS untuk integrasi proxy

Untuk integrasi proxy Lambda atau integrasi proxy HTTP, backend Anda bertanggung jawab untuk mengembalikan`Access-Control-Allow-Origin`,`Access-Control-Allow-Methods`, dan `Access-Control-Allow-Headers` header, karena integrasi proxy tidak mengembalikan respons integrasi.

Contoh berikut fungsi Lambda mengembalikan header CORS yang diperlukan:

Node.js

```
export const handler = async (event) => {
    const response = {
        statusCode: 200,
        headers: {
            "Access-Control-Allow-Headers" : "Content-Type",
            "Access-Control-Allow-Origin": "https://www.example.com",
            "Access-Control-Allow-Methods": "OPTIONS,POST,GET"
        },
        body: JSON.stringify('Hello from Lambda!'),
    };
    return response;
};
```

Python 3

```
import json

def lambda_handler(event, context):
    return {
        'statusCode': 200,
        'headers': {
            'Access-Control-Allow-Headers': 'Content-Type',
            'Access-Control-Allow-Origin': 'https://www.example.com',
            'Access-Control-Allow-Methods': 'OPTIONS,POST,GET'
        },
        'body': json.dumps('Hello from Lambda!')
    }
```

Topik

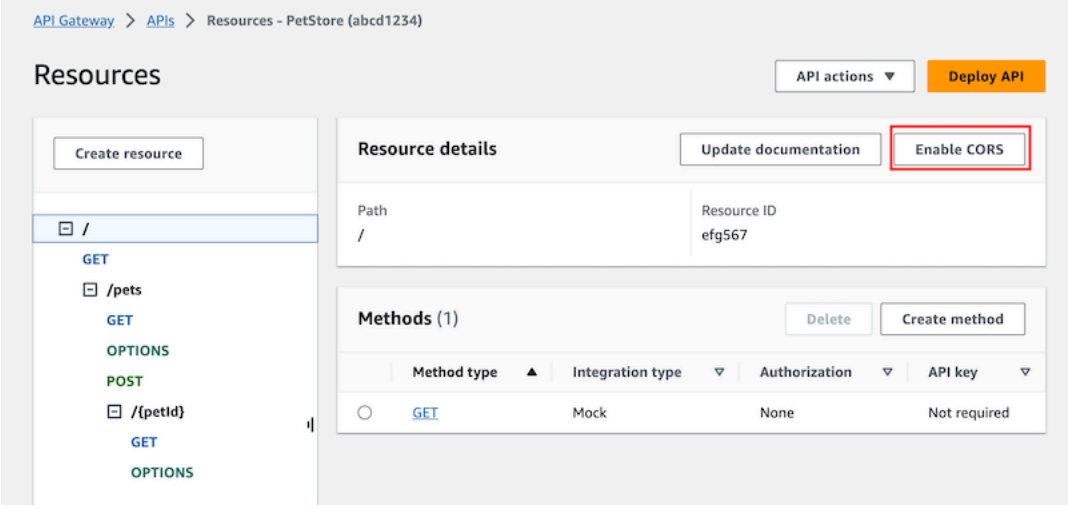
Aktifkan CORS pada sumber daya menggunakan konsol API Gateway

Anda dapat menggunakan konsol API Gateway untuk mengaktifkan dukungan CORS untuk satu atau semua metode pada sumber daya REST API yang telah Anda buat.

> **Important**
>
> Sumber daya dapat berisi sumber daya anak. Mengaktifkan dukungan CORS untuk sumber daya dan metodenya tidak secara rekursif mengaktifkannya untuk sumber daya anak dan metodenya.

Untuk mengaktifkan dukungan CORS pada sumber daya REST API

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API.
3. Pilih sumber daya di bawah Sumber Daya.
4. Di bagian Rincian sumber daya, pilih Aktifkan CORS.



5. Di kotak Aktifkan CORS, lakukan hal berikut:

a. (Opsional) Jika Anda membuat respons gateway khusus dan ingin mengaktifkan dukungan CORS untuk respons, pilih respons gateway.

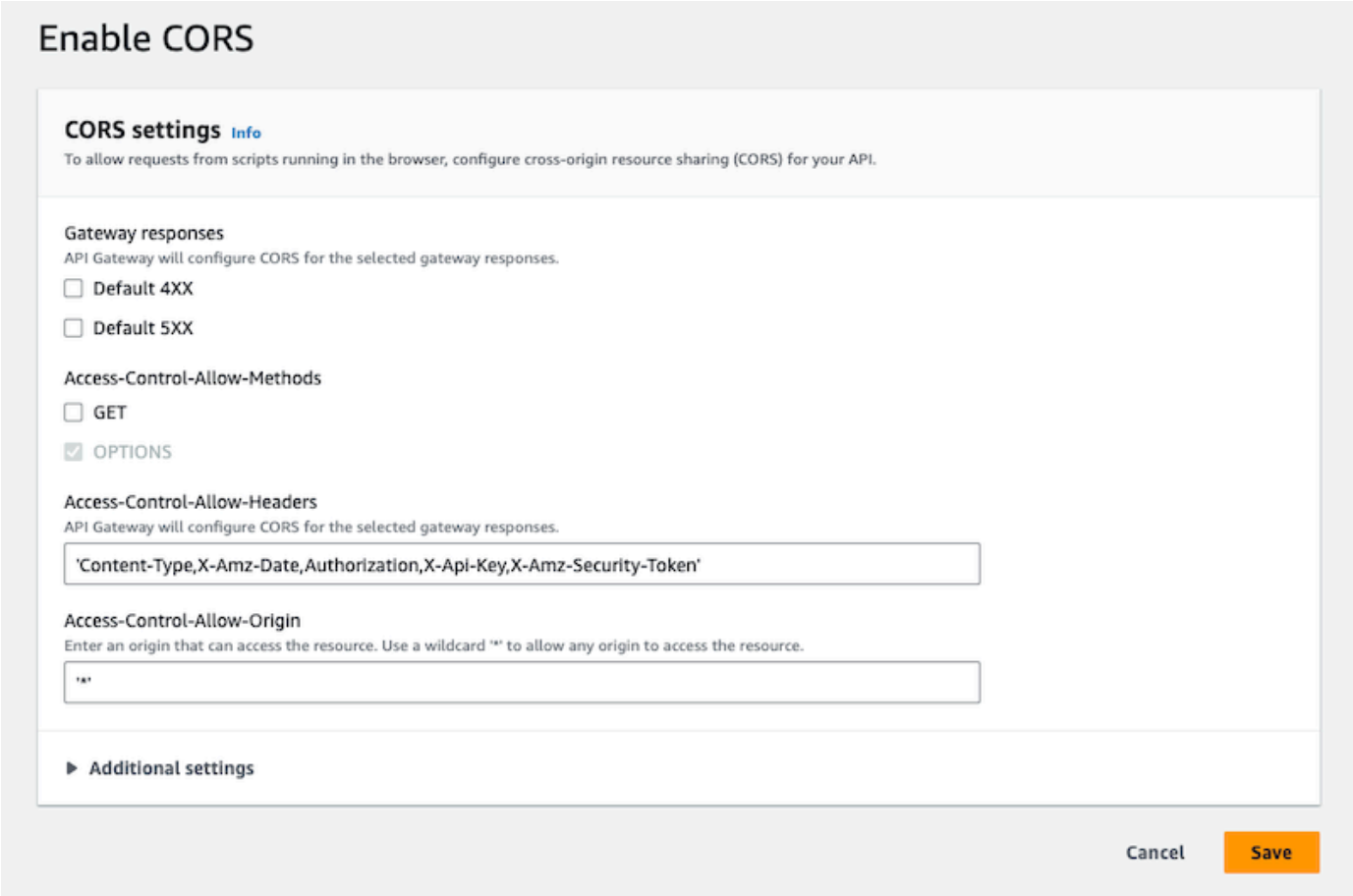
b. Pilih setiap metode untuk mengaktifkan dukungan CORS. `OPTIONS`Metode ini harus mengaktifkan CORS.

   Jika Anda mengaktifkan dukungan CORS untuk suatu `ANY` metode, CORS diaktifkan untuk semua metode.

c. Di bidang input Access-Control-Allow-Headers, masukkan string statis dari daftar header yang dipisahkan koma yang harus dikirimkan klien dalam permintaan sumber daya yang sebenarnya. Gunakan daftar header yang disediakan konsol `'Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token'` atau tentukan header Anda sendiri.

d. Gunakan nilai yang disediakan konsol `'*'` sebagai nilai header Access-Control-Allow-Origin untuk mengizinkan permintaan akses dari semua asal, atau tentukan asal yang akan diizinkan mengakses sumber daya.

e. Pilih Simpan.



> **Important**
>
> Saat menerapkan instruksi di atas ke `ANY` metode dalam integrasi proxy, header CORS apa pun yang berlaku tidak akan disetel. Sebagai gantinya, backend Anda harus mengembalikan header CORS yang berlaku, seperti. `Access-Control-Allow-Origin`

Setelah CORS diaktifkan pada `GET` metode, `OPTIONS` metode ditambahkan ke sumber daya, jika belum ada. `200`Respons `OPTIONS` metode ini secara otomatis dikonfigurasi untuk mengembalikan tiga `Access-Control-Allow-*` header untuk memenuhi jabat tangan preflight. Selain itu, metode aktual (`GET`) juga dikonfigurasi secara default untuk mengembalikan `Access-Control-Allow-Origin` header dalam respons 200 juga. Untuk jenis tanggapan lain, Anda perlu mengonfigurasinya secara manual untuk mengembalikan `Access-Control-Allow-Origin'` header dengan '*' atau asal tertentu, jika Anda tidak ingin mengembalikan `Cross-origin access` kesalahan.

Setelah Anda mengaktifkan dukungan CORS pada sumber daya Anda, Anda harus menerapkan atau menerapkan ulang API agar pengaturan baru diterapkan. Untuk informasi selengkapnya, lihat the section called “Menerapkan REST API (konsol)”.

> **Note**
>
> Jika Anda tidak dapat mengaktifkan dukungan CORS pada sumber daya Anda setelah mengikuti prosedur, kami sarankan Anda membandingkan konfigurasi CORS Anda dengan sumber daya API `/pets` contoh. Untuk mempelajari cara membuat contoh API, lihatthe section called “Tutorial: Buat REST API dengan mengimpor contoh”.

Aktifkan CORS pada sumber daya menggunakan API impor API Gateway

Jika Anda menggunakan API Gateway Import API, Anda dapat mengatur dukungan CORS menggunakan file OpenAPI. Anda harus terlebih dahulu menentukan `OPTIONS` metode dalam sumber daya Anda yang mengembalikan header yang diperlukan.

> **Note**
>
> Browser web mengharapkan header Access-Control-Allow-Headers, dan Access-Control-Allow-Origin disiapkan di setiap metode API yang menerima permintaan CORS. Selain itu, beberapa browser pertama-tama membuat permintaan HTTP ke `OPTIONS` metode di sumber daya yang sama, dan kemudian berharap untuk menerima header yang sama.

`Options`Metode contoh

Contoh berikut menciptakan `OPTIONS` metode untuk integrasi tiruan.

OpenAPI 3.0

```
/users:
  options:
    summary: CORS support
    description: |
      Enable CORS by returning correct headers
    tags:
    - CORS
    responses:
```

```
      200:
        description: Default response for CORS method
        headers:
          Access-Control-Allow-Origin:
            schema:
              type: "string"
          Access-Control-Allow-Methods:
            schema:
              type: "string"
          Access-Control-Allow-Headers:
            schema:
              type: "string"
        content: {}
    x-amazon-apigateway-integration:
      type: mock
      requestTemplates:
        application/json: "{\"statusCode\": 200}"
      responses:
        default:
          statusCode: "200"
          responseParameters:
            method.response.header.Access-Control-Allow-Headers: "'Content-Type,X-
Amz-Date,Authorization,X-Api-Key'"
            method.response.header.Access-Control-Allow-Methods: "'*'"
            method.response.header.Access-Control-Allow-Origin: "'*'"
```

OpenAPI 2.0

```
/users:
   options:
      summary: CORS support
      description: |
        Enable CORS by returning correct headers
      consumes:
        - "application/json"
      produces:
        - "application/json"
      tags:
        - CORS
      x-amazon-apigateway-integration:
        type: mock
        requestTemplates: "{\"statusCode\": 200}"
```

```
        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"
```

Setelah Anda mengonfigurasi `OPTIONS` metode untuk sumber daya Anda, Anda dapat menambahkan header yang diperlukan ke metode lain di sumber daya yang sama yang perlu menerima permintaan CORS.

1. Deklarasikan Access-Control-Allow-Origin dan Header ke tipe respons.

   OpenAPI 3.0

   ```
     responses:
       200:
         description: Default response for CORS method
         headers:
           Access-Control-Allow-Origin:
             schema:
               type: "string"
           Access-Control-Allow-Methods:
             schema:
               type: "string"
           Access-Control-Allow-Headers:
             schema:
               type: "string"
         content: {}
   ```

OpenAPI 2.0

```
      responses:
        200:
          description: Default response for CORS method
          headers:
            Access-Control-Allow-Headers:
              type: "string"
            Access-Control-Allow-Methods:
              type: "string"
            Access-Control-Allow-Origin:
              type: "string"
```

2. Di `x-amazon-apigateway-integration` tag, atur pemetaan untuk header tersebut ke nilai statis Anda:

OpenAPI 3.0

```
    responses:
        default:
          statusCode: "200"
          responseParameters:
            method.response.header.Access-Control-Allow-Headers: "'Content-
Type,X-Amz-Date,Authorization,X-Api-Key'"
            method.response.header.Access-Control-Allow-Methods: "'*'"
            method.response.header.Access-Control-Allow-Origin: "'*'"
          responseTemplates:
            application/json: |
              {}
```

OpenAPI 2.0

```
        responses:
          "default":
            statusCode: "200"
            responseParameters:
              method.response.header.Access-Control-Allow-Headers : "'Content-
Type,X-Amz-Date,Authorization,X-Api-Key'"
              method.response.header.Access-Control-Allow-Methods : "'*'"
              method.response.header.Access-Control-Allow-Origin : "'*'"
```

Contoh API

Contoh berikut membuat API lengkap dengan `OPTIONS` metode dan `GET` metode dengan `HTTP` integrasi.

OpenAPI 3.0

```
openapi: "3.0.1"
info:
  title: "cors-api"
  description: "cors-api"
  version: "2024-01-16T18:36:01Z"
servers:
- url: "/{basePath}"
  variables:
    basePath:
      default: "/test"
paths:
  /:
    get:
      operationId: "GetPet"
      responses:
        "200":
          description: "200 response"
          headers:
            Access-Control-Allow-Origin:
              schema:
                type: "string"
          content: {}
      x-amazon-apigateway-integration:
        httpMethod: "GET"
        uri: "http://petstore.execute-api.us-east-1.amazonaws.com/petstore/pets"
        responses:
          default:
            statusCode: "200"
            responseParameters:
              method.response.header.Access-Control-Allow-Origin: "'*'"
        passthroughBehavior: "when_no_match"
        type: "http"
    options:
      responses:
        "200":
          description: "200 response"
          headers:
```

```
            Access-Control-Allow-Origin:
              schema:
                type: "string"
            Access-Control-Allow-Methods:
              schema:
                type: "string"
            Access-Control-Allow-Headers:
              schema:
                type: "string"
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/Empty"
      x-amazon-apigateway-integration:
        responses:
          default:
            statusCode: "200"
            responseParameters:
              method.response.header.Access-Control-Allow-Methods: "'GET,OPTIONS'"
              method.response.header.Access-Control-Allow-Headers: "'Content-Type,X-
Amz-Date,Authorization,X-Api-Key'"
              method.response.header.Access-Control-Allow-Origin: "'*'"
        requestTemplates:
          application/json: "{\"statusCode\": 200}"
        passthroughBehavior: "when_no_match"
        type: "mock"
components:
  schemas:
    Empty:
      type: "object"
```

OpenAPI 2.0

```
swagger: "2.0"
info:
  description: "cors-api"
  version: "2024-01-16T18:36:01Z"
  title: "cors-api"
basePath: "/test"
schemes:
- "https"
paths:
  /:
```

```
    get:
      operationId: "GetPet"
      produces:
      - "application/json"
      responses:
        "200":
          description: "200 response"
          headers:
            Access-Control-Allow-Origin:
              type: "string"
      x-amazon-apigateway-integration:
        httpMethod: "GET"
        uri: "http://petstore.execute-api.us-east-1.amazonaws.com/petstore/pets"
        responses:
          default:
            statusCode: "200"
            responseParameters:
              method.response.header.Access-Control-Allow-Origin: "'*'"
        passthroughBehavior: "when_no_match"
        type: "http"
    options:
      consumes:
      - "application/json"
      produces:
      - "application/json"
      responses:
        "200":
          description: "200 response"
          schema:
            $ref: "#/definitions/Empty"
          headers:
            Access-Control-Allow-Origin:
              type: "string"
            Access-Control-Allow-Methods:
              type: "string"
            Access-Control-Allow-Headers:
              type: "string"
      x-amazon-apigateway-integration:
        responses:
          default:
            statusCode: "200"
            responseParameters:
              method.response.header.Access-Control-Allow-Methods: "'GET,OPTIONS'"
```

```
              method.response.header.Access-Control-Allow-Headers: "'Content-Type,X-
Amz-Date,Authorization,X-Api-Key'"
              method.response.header.Access-Control-Allow-Origin: "'*'"
        requestTemplates:
          application/json: "{\"statusCode\": 200}"
        passthroughBehavior: "when_no_match"
        type: "mock"
definitions:
  Empty:
    type: "object"
```

Menguji CORS

Anda dapat menguji konfigurasi CORS API Anda dengan menjalankan API Anda, dan memeriksa header CORS dalam respons. `curl`Perintah berikut mengirimkan permintaan OPTIONS ke API yang diterapkan.

```
curl -v -X OPTIONS https://{restapi_id}.execute-api.{region}.amazonaws.com/{stage_name}
```

```
< HTTP/1.1 200 OK
< Date: Tue, 19 May 2020 00:55:22 GMT
< Content-Type: application/json
< Content-Length: 0
< Connection: keep-alive
< x-amzn-RequestId: a1b2c3d4-5678-90ab-cdef-abc123
< Access-Control-Allow-Origin: *
< Access-Control-Allow-Headers: Content-Type,Authorization,X-Amz-Date,X-Api-Key,X-Amz-
Security-Token
< x-amz-apigw-id: Abcd=
< Access-Control-Allow-Methods: DELETE,GET,HEAD,OPTIONS,PATCH,POST,PUT
```

`Access-Control-Allow-Methods`Header `Access-Control-Allow-Origin`Access-`Control-Allow-Headers`,, dan dalam respons menunjukkan bahwa API mendukung CORS. Lihat informasi yang lebih lengkap di Mengaktifkan CORS untuk sumber daya REST API.

Bekerja dengan tipe media biner untuk REST API

Di API Gateway, permintaan dan respons API memiliki muatan teks atau biner. Payload teks adalah string JSON `UTF-8` -encoded. Payload biner adalah apa pun selain payload teks. Payload biner

dapat berupa, misalnya, file JPEG, file GZip, atau file XHTML. Konfigurasi API yang diperlukan untuk mendukung media biner bergantung pada apakah API Anda menggunakan integrasi proxy atau non-proxy.

AWS Lambdaintegrasi proxy

Untuk menangani payload biner untuk integrasi AWS Lambda proxy, Anda harus base64-menyandikan respons fungsi Anda. Anda juga harus mengonfigurasi binaryMediaTypesuntuk API Anda. `binaryMediaTypes`Konfigurasi API Anda adalah daftar tipe konten yang API Anda perlakukan sebagai data biner. Contoh jenis media biner termasuk `image/png` atau`application/octet-stream`. Anda dapat menggunakan karakter wildcard (`*`) untuk mencakup beberapa jenis media. Misalnya, `*/*` termasuk semua jenis konten.

Untuk kode sampel, lihat the section called “Kembalikan media biner dari integrasi proxy Lambda”.

Integrasi non-proxy

Untuk menangani muatan biner untuk integrasi non-proxy, Anda menambahkan jenis media ke binaryMediaTypesdaftar sumber daya. `RestApi` `binaryMediaTypes`Konfigurasi API Anda adalah daftar tipe konten yang API Anda perlakukan sebagai data biner. Atau, Anda dapat mengatur properti ContentHandling pada Integrasi dan sumber daya. IntegrationResponse `contentHandling`Nilainya bisa`CONVERT_TO_BINARY`,`CONVERT_TO_TEXT`, atau tidak terdefinisi.

Bergantung pada `contentHandling` nilainya, dan apakah header respons atau `Content-Type` header permintaan yang masuk cocok dengan entri dalam `binaryMediaTypes` daftar, API Gateway dapat menyandikan byte biner mentah sebagai string yang disandikan base64, memecahkan kode string yang dikodekan base64 kembali ke byte mentahnya, atau meneruskan isi tanpa modifikasi. `Accept`

Anda harus mengonfigurasi API sebagai berikut untuk mendukung payload biner untuk API Anda di API Gateway:

- Tambahkan jenis media biner yang diinginkan ke `binaryMediaTypes` daftar pada RestApisumber daya. Jika properti ini dan properti tidak ditentukan, muatan ditangani sebagai string JSON yang dikodekan UTF-8. `contentHandling`
- Alamat `contentHandling` properti sumber daya Integrasi.
  - Agar payload permintaan dikonversi dari string yang dikodekan base64 ke gumpalan binernya, setel properti ke. `CONVERT_TO_BINARY`

- Agar payload permintaan dikonversi dari gumpalan biner ke string yang dikodekan base64, setel properti ke. `CONVERT_TO_TEXT`
- Untuk meneruskan muatan tanpa modifikasi, biarkan properti tidak terdefinisi. Untuk meneruskan payload biner tanpa modifikasi, Anda juga harus memastikan bahwa entri tersebut `Content-Type` cocok dengan salah satu `binaryMediaTypes` entri, dan perilaku passthrough diaktifkan untuk API.

- Atur `contentHandling` properti sumber IntegrationResponsedaya. `contentHandling`Properti, `Accept` header dalam permintaan klien, dan `binaryMediaTypes` gabungan API Anda menentukan cara API Gateway menangani konversi tipe konten. Untuk detailnya, lihat the section called “Konversi jenis konten di API Gateway”.

> **Important**
>
> Jika permintaan berisi beberapa jenis media di `Accept` header, API Gateway hanya menghormati jenis `Accept` media pertama. Jika Anda tidak dapat mengontrol urutan jenis `Accept` media dan jenis media konten biner Anda bukan yang pertama dalam daftar, tambahkan jenis `Accept` media pertama dalam `binaryMediaTypes` daftar API Anda. API Gateway menangani semua jenis konten dalam daftar ini sebagai biner.
> Misalnya, untuk mengirim file JPEG menggunakan `<img>` elemen di browser, browser mungkin mengirim `Accept:image/webp,image/*,*/*;q=0.8` permintaan. Dengan menambahkan `image/webp` ke `binaryMediaTypes` daftar, titik akhir menerima file JPEG sebagai biner.

Untuk informasi terperinci tentang cara API Gateway menangani muatan teks dan biner, lihatKonversi jenis konten di API Gateway.

Konversi jenis konten di API Gateway

Kombinasi API Anda`binaryMediaTypes`, header dalam permintaan klien, dan `contentHandling` properti integrasi menentukan cara API Gateway menyandikan payload.

Tabel berikut menunjukkan cara API Gateway mengonversi payload permintaan untuk konfigurasi spesifik `Content-Type` header permintaan, `binaryMediaTypes` daftar RestApisumber daya, dan nilai `contentHandling` properti sumber daya Integrasi.

API meminta konversi jenis konten di API Gateway

Metode permintaan payload	Permintaan **`Content-Type`** header	**`binaryMediaTypes`**	**`contentHandling`**	Muatan permintaan integrasi
Data teks	Tipe data apa pun	Tidak terdefinisi	Tidak terdefinisi	String yang dikodekan UTF8
Data teks	Tipe data apa pun	Tidak terdefinisi	`CONVERT_TO_BINARY`	Gumpalan biner yang didekode Base64
Data teks	Tipe data apa pun	Tidak terdefinisi	`CONVERT_TO_TEXT`	String yang dikodekan UTF8
Data teks	Tipe data teks	Set dengan jenis media yang cocok	Tidak terdefinisi	Data teks
Data teks	Tipe data teks	Set dengan jenis media yang cocok	`CONVERT_TO_BINARY`	Gumpalan biner yang didekode Base64
Data teks	Tipe data teks	Set dengan jenis media yang cocok	`CONVERT_TO_TEXT`	Data teks
Binary data	Tipe data biner	Set dengan jenis media yang cocok	Tidak terdefinisi	Binary data
Binary data	Tipe data biner	Set dengan jenis media yang cocok	`CONVERT_TO_BINARY`	Binary data
Binary data	Tipe data biner	Set dengan jenis media yang cocok	`CONVERT_TO_TEXT`	String yang dikodekan Base64

Tabel berikut menunjukkan cara API Gateway mengonversi payload respons untuk konfigurasi spesifik `Accept` header permintaan, `binaryMediaTypes` daftar RestApisumber daya, dan nilai `contentHandling` properti sumber daya. IntegrationResponse

> **Important**
>
> Jika permintaan berisi beberapa jenis media di `Accept` header, API Gateway hanya menghormati jenis `Accept` media pertama. Jika Anda tidak dapat mengontrol urutan jenis `Accept` media dan jenis media konten biner Anda bukan yang pertama dalam daftar, tambahkan jenis `Accept` media pertama dalam `binaryMediaTypes` daftar API Anda. API Gateway menangani semua jenis konten dalam daftar ini sebagai biner.
> Misalnya, untuk mengirim file JPEG menggunakan `<img>` elemen di browser, browser mungkin mengirim `Accept:image/webp,image/*,*/*;q=0.8` permintaan. Dengan menambahkan `image/webp` ke `binaryMediaTypes` daftar, titik akhir menerima file JPEG sebagai biner.

Konversi jenis konten respons API Gateway

Payload respons integrasi	Permintaan **`Accept`** header	**`binaryMediaTypes`**	**`contentHandling`**	Metode respon payload
Teks atau data biner	Tipe teks	Tidak terdefinisi	Tidak terdefinisi	String yang dikodekan UTF8
Teks atau data biner	Tipe teks	Tidak terdefinisi	`CONVERT_TO_BINARY`	Gumpalan yang didekodekan Base64
Teks atau data biner	Tipe teks	Tidak terdefinisi	`CONVERT_TO_TEXT`	String yang dikodekan UTF8
Data teks	Tipe teks	Set dengan jenis media yang cocok	Tidak terdefinisi	Data teks
Data teks	Tipe teks	Set dengan jenis media yang cocok	`CONVERT_TO_BINARY`	Gumpalan yang didekodekan Base64

Payload respons integrasi	Permintaan **`Accept`** header	**`binaryMediaTypes`**	**`contentHandling`**	Metode respon payload
Data teks	Tipe teks	Set dengan jenis media yang cocok	`CONVERT_TO_TEXT`	String yang dikodekan UTF8
Data teks	Tipe biner	Set dengan jenis media yang cocok	Tidak terdefinisi	Gumpalan yang didekodekan Base64
Data teks	Tipe biner	Set dengan jenis media yang cocok	`CONVERT_TO_BINARY`	Gumpalan yang didekodekan Base64
Data teks	Tipe biner	Set dengan jenis media yang cocok	`CONVERT_TO_TEXT`	String yang dikodekan UTF8
Binary data	Tipe teks	Set dengan jenis media yang cocok	Tidak terdefinisi	String yang dikodekan Base64
Binary data	Tipe teks	Set dengan jenis media yang cocok	`CONVERT_TO_BINARY`	Binary data
Binary data	Tipe teks	Set dengan jenis media yang cocok	`CONVERT_TO_TEXT`	String yang dikodekan Base64
Binary data	Tipe biner	Set dengan jenis media yang cocok	Tidak terdefinisi	Binary data
Binary data	Tipe biner	Set dengan jenis media yang cocok	`CONVERT_TO_BINARY`	Binary data

Payload respons integrasi	Permintaan **`Accept`** header	**`binaryMediaTypes`**	**`contentHandling`**	Metode respon payload
Binary data	Tipe biner	Set dengan jenis media yang cocok	`CONVERT_TO_TEXT`	String yang dikodekan Base64

Saat mengonversi payload teks menjadi gumpalan biner, API Gateway mengasumsikan bahwa data teks adalah string yang dikodekan base64 dan mengeluarkan data biner sebagai gumpalan yang diterjemahkan base64. Jika konversi gagal, ia mengembalikan `500` respons, yang menunjukkan kesalahan konfigurasi API. Anda tidak menyediakan template pemetaan untuk konversi semacam itu, meskipun Anda harus mengaktifkan perilaku passthrough di API.

Saat mengonversi payload biner menjadi string teks, API Gateway selalu menerapkan pengkodean base64 pada data biner. Anda dapat menentukan templat pemetaan untuk muatan semacam itu, tetapi hanya dapat mengakses string yang dikodekan base64 dalam templat pemetaan`$input.body`, seperti yang ditunjukkan dalam kutipan contoh templat pemetaan berikut.

```
{
    "data": "$input.body"
}
```

Agar payload biner melewati tanpa modifikasi, Anda harus mengaktifkan perilaku passthrough di API.

Mengaktifkan dukungan biner menggunakan konsol API Gateway

Bagian ini menjelaskan cara mengaktifkan dukungan biner menggunakan konsol API Gateway. Sebagai contoh, kami menggunakan API yang terintegrasi dengan Amazon S3. Kami fokus pada tugas untuk mengatur jenis media yang didukung dan untuk menentukan bagaimana payload harus ditangani. Untuk informasi terperinci tentang cara membuat API yang terintegrasi dengan Amazon S3, lihat. Tutorial: Membuat REST API sebagai proxy Amazon S3 di API Gateway

Untuk mengaktifkan dukungan biner dengan menggunakan konsol API Gateway

1. Tetapkan tipe media biner untuk API:

    a. Buat API baru atau pilih API yang sudah ada. Untuk contoh ini, kami memberi nama API`FileMan`.

b. Di bawah API yang dipilih di panel navigasi utama, pilih pengaturan API.

c. Di panel pengaturan API, pilih Kelola jenis media di bagian Jenis Media Biner.

d. Pilih Tambahkan tipe media biner.

e. Masukkan jenis media yang diperlukan, misalnya**`image/png`**, di bidang teks input. Jika perlu, ulangi langkah ini untuk menambahkan lebih banyak jenis media. Untuk mendukung semua jenis media biner, tentukan`*/*`.

f. Pilih Simpan perubahan.

2. Mengatur cara payload pesan ditangani untuk metode API:

a. Buat yang baru atau pilih sumber daya yang ada di API. Untuk contoh ini, kami menggunakan `/{folder}/{item}` sumber daya.

b. Buat yang baru atau pilih metode yang ada pada sumber daya. Sebagai contoh, kami menggunakan `GET /{folder}/{item}` metode yang terintegrasi dengan `Object GET` tindakan di Amazon S3.

c. Untuk penanganan Konten, pilih opsi.

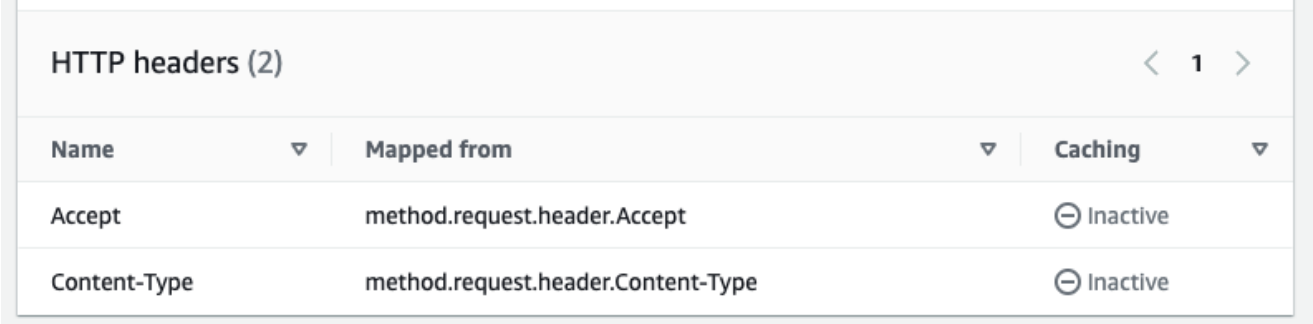
Pilih Passthrough jika Anda tidak ingin mengonversi isi ketika klien dan backend menerima format biner yang sama. Pilih Konversi ke teks untuk mengonversi badan biner menjadi string yang dikodekan base64 ketika, misalnya, backend mengharuskan payload permintaan biner diteruskan sebagai properti JSON. Dan pilih Konversi ke biner ketika klien

mengirimkan string yang dikodekan base64 dan backend memerlukan format biner asli, atau ketika titik akhir mengembalikan string yang dikodekan base64 dan klien hanya menerima output biner.

d. Untuk passthrough badan Permintaan, pilih Bila tidak ada templat yang ditentukan (disarankan) untuk mengaktifkan perilaku passthrough pada badan permintaan.

   Anda juga bisa memilih Never. Ini berarti bahwa API akan menolak data dengan tipe konten yang tidak memiliki template pemetaan.

e. Pertahankan `Accept` header permintaan masuk dalam permintaan integrasi. Anda harus melakukan ini jika Anda telah mengatur `passthrough` dan `contentHandling` ingin mengganti pengaturan itu saat runtime.



f. Untuk konversi ke teks, tentukan template pemetaan untuk menempatkan data biner yang dikodekan base64 ke dalam format yang diperlukan.

   Contoh template pemetaan untuk dikonversi ke teks adalah sebagai berikut:

```
{
  "operation": "thumbnail",
  "base64Image": "$input.body"
}
```

   Format template pemetaan ini tergantung pada persyaratan titik akhir input.

g. Pilih Simpan.

Mengaktifkan dukungan biner menggunakan API Gateway REST API

Tugas berikut menunjukkan cara mengaktifkan dukungan biner menggunakan panggilan API Gateway REST API.

Topik

Menambahkan dan memperbarui tipe media biner yang didukung ke API

Untuk mengaktifkan API Gateway untuk mendukung jenis media biner baru, Anda harus menambahkan jenis media biner ke `binaryMediaTypes` daftar `RestApi` sumber daya. Misalnya, agar API Gateway menangani gambar JPEG, kirimkan `PATCH` permintaan ke `RestApi` sumber daya:

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/image~1jpeg"
  }
 ]
}
```

Spesifikasi tipe MIME `image/jpeg` yang merupakan bagian dari nilai `path` properti diloloskan sebagai. `image~1jpeg`

Untuk memperbarui jenis media biner yang didukung, ganti atau hapus jenis media dari `binaryMediaTypes` daftar `RestApi` sumber daya. Misalnya, untuk mengubah dukungan biner dari file JPEG ke byte mentah, kirimkan `PATCH` permintaan ke `RestApi` sumber daya, sebagai berikut:

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [{
    "op" : "replace",
    "path" : "/binaryMediaTypes/image~1jpeg",
```

```
    "value" : "application/octet-stream"
  },
  {
    "op" : "remove",
    "path" : "/binaryMediaTypes/image~1jpeg"
  }]
}
```

Konfigurasikan konversi payload permintaan

Jika titik akhir membutuhkan input biner, atur `contentHandling` properti `Integration` sumber daya ke`CONVERT_TO_BINARY`. Untuk melakukannya, kirimkan `PATCH` permintaan, sebagai berikut:

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/integration

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "/contentHandling",
    "value" : "CONVERT_TO_BINARY"
  }]
}
```

Konfigurasikan konversi payload respons

Jika klien menerima hasilnya sebagai gumpalan biner alih-alih muatan yang dikodekan base64 yang dikembalikan dari titik akhir, setel properti sumber daya ke. `contentHandling` `IntegrationResponse` `CONVERT_TO_BINARY` Untuk melakukan ini, kirimkan `PATCH` permintaan, sebagai berikut:

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/integration/
responses/<status_code>

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "/contentHandling",
    "value" : "CONVERT_TO_BINARY"
  }]
}
```

Ubah data biner menjadi data teks

Untuk mengirim data biner sebagai properti JSON dari input ke AWS Lambda atau Kinesis melalui API Gateway, lakukan hal berikut:

1. Aktifkan dukungan payload biner API dengan menambahkan jenis media biner baru `application/octet-stream` ke `binaryMediaTypes` daftar API.

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/application~1octet-stream"
  }
 ]
}
```

2. Tetapkan `CONVERT_TO_TEXT` pada `contentHandling` properti `Integration` sumber daya dan sediakan template pemetaan untuk menetapkan string data biner yang dikodekan base64 ke properti JSON. Dalam contoh berikut, properti JSON adalah `body` dan `$input.body` memegang string yang dikodekan base64.

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/
integration

{
  "patchOperations" : [
    {
      "op" : "replace",
      "path" : "/contentHandling",
      "value" : "CONVERT_TO_TEXT"
    },
    {
      "op" : "add",
      "path" : "/requestTemplates/application~1octet-stream",
      "value" : "{\"body\": \"$input.body\"}"
    }
  ]
}
```

Mengkonversi data teks ke payload biner

Misalkan fungsi Lambda mengembalikan file gambar sebagai string yang dikodekan base64. Untuk meneruskan output biner ini ke klien melalui API Gateway, lakukan hal berikut:

1. Perbarui `binaryMediaTypes` daftar API dengan menambahkan jenis media biner`application/octet-stream`, jika belum ada dalam daftar.

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/application~1octet-stream",
  }]
}
```

2. Setel `contentHandling` properti pada `Integration` sumber daya ke`CONVERT_TO_BINARY`. Jangan mendefinisikan template pemetaan. Jika Anda tidak mendefinisikan template pemetaan, API Gateway memanggil template passthrough untuk mengembalikan blob biner yang diterjemahkan base64 sebagai file gambar ke klien.

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/
integration/responses/<status_code>

{
  "patchOperations" : [
    {
      "op" : "replace",
      "path" : "/contentHandling",
      "value" : "CONVERT_TO_BINARY"
    }
  ]
}
```

Melewati muatan biner

Untuk menyimpan gambar di bucket Amazon S3 menggunakan API Gateway, lakukan hal berikut:

1. Perbarui `binaryMediaTypes` daftar API dengan menambahkan jenis media biner`application/octet-stream`, jika belum ada dalam daftar.

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/application~1octet-stream"
  }
 ]
}
```

2. Pada `contentHandling` properti sumber `Integration` daya, atur`CONVERT_TO_BINARY`. Tetapkan `WHEN_NO_MATCH` sebagai nilai `passthroughBehavior` properti tanpa mendefinisikan template pemetaan. Hal ini memungkinkan API Gateway untuk memanggil template passthrough.

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/
integration

{
  "patchOperations" : [
    {
      "op" : "replace",
      "path" : "/contentHandling",
      "value" : "CONVERT_TO_BINARY"
    },
    {
      "op" : "replace",
      "path" : "/passthroughBehaviors",
      "value" : "WHEN_NO_MATCH"
    }
  ]
}
```

Impor dan ekspor pengkodean konten

Untuk mengimpor `binaryMediaTypes` daftar di a RestApi, gunakan ekstensi API Gateway berikut ke file definisi OpenAPI API. Ekstensi ini juga digunakan untuk mengekspor pengaturan API.

- x-amazon-apigateway-binaryproperti -media-tipe

Untuk mengimpor dan mengekspor nilai `contentHandling` properti pada `IntegrationResponse` sumber daya `Integration` atau, gunakan ekstensi API Gateway berikut ke definisi OpenAPI:

- x-amazon-apigateway-integration objek
- x-amazon-apigateway-integration.response objek

Contoh dukungan biner

Contoh berikut menunjukkan cara mengakses file biner di Amazon S3 atau AWS Lambda melalui API Gateway API.

Topik

- Kembalikan media biner dari integrasi proxy Lambda
- Akses file biner di Amazon S3 melalui API Gateway API
- Akses file biner di Lambda menggunakan API Gateway API

Kembalikan media biner dari integrasi proxy Lambda

Untuk mengembalikan media biner dari integrasi AWS Lambda proxy, base64 menyandikan respons dari fungsi Lambda Anda. Anda juga harus mengonfigurasi tipe media biner API Anda. Batas ukuran muatan adalah 10 MB.

> **Note**
>
> Untuk menggunakan browser web untuk menjalankan API dengan contoh integrasi ini, setel tipe media biner API Anda ke`*/*`. API Gateway menggunakan `Accept` header pertama dari klien untuk menentukan apakah respons harus mengembalikan media biner. Untuk mengembalikan media biner saat Anda tidak dapat mengontrol urutan nilai `Accept` header, seperti permintaan dari browser, setel tipe media biner API Anda ke `*/*` (untuk semua jenis konten).

Contoh berikut fungsi Lambda dapat mengembalikan gambar biner dari Amazon S3 atau teks ke klien. Respons fungsi mencakup `Content-Type` header untuk menunjukkan kepada klien jenis data yang dikembalikan. Fungsi secara kondisional menetapkan `isBase64Encoded` properti dalam responsnya, tergantung pada jenis data yang dikembalikan.

Node.js

```
import { S3Client, GetObjectCommand } from "@aws-sdk/client-s3"

const client = new S3Client({region: 'us-east-2'});

export const handler = async (event) => {

  var randomint = function(max) {
    return Math.floor(Math.random() * max);
  }
  var number = randomint(2);
  if (number == 1){
    const input = {
      "Bucket" : "bucket-name",
      "Key" : "image.png"
      }
    try {
      const command = new GetObjectCommand(input)
      const response = await client.send(command);
      var str = await response.Body.transformToByteArray();
    } catch (err) {
      console.error(err);
    }
    const base64body = Buffer.from(str).toString('base64');
    return {
      'headers': { "Content-Type": "image/png" },
      'statusCode': 200,
      'body': base64body,
      'isBase64Encoded': true
      }
    } else {
        return {
        'headers': { "Content-Type": "text/html" },
        'statusCode': 200,
        'body': "<h1>This is text</h1>",
        }
    }
}
```

Python

```
import base64
```

```
import boto3
import json
import random

s3 = boto3.client('s3')

def lambda_handler(event, context):
    number = random.randint(0,1)
    if number == 1:
        response = s3.get_object(
            Bucket='bucket-name',
            Key='image.png',
        )
        image = response['Body'].read()
        return {
            'headers': { "Content-Type": "image/png" },
            'statusCode': 200,
            'body': base64.b64encode(image).decode('utf-8'),
            'isBase64Encoded': True
        }
    else:
        return {
            'headers': { "Content-type": "text/html" },
            'statusCode': 200,
            'body': "<h1>This is text</h1>",
        }
```

Untuk mempelajari lebih lanjut tentang jenis media biner, lihatBekerja dengan tipe media biner untuk REST API.

Akses file biner di Amazon S3 melalui API Gateway API

Contoh berikut menunjukkan file OpenAPI yang digunakan untuk mengakses gambar di Amazon S3, cara mengunduh gambar dari Amazon S3, dan cara mengunggah gambar ke Amazon S3.

Topik

- File OpenAPI dari API sampel untuk mengakses gambar di Amazon S3
- Unduh gambar dari Amazon S3
- Unggah gambar ke Amazon S3

File OpenAPI dari API sampel untuk mengakses gambar di Amazon S3

File OpenAPI berikut menunjukkan contoh API yang menggambarkan mengunduh file gambar dari Amazon S3 dan mengunggah file gambar ke Amazon S3. API ini mengekspos `GET /s3?key={file-name}` dan `PUT /s3?key={file-name}` metode untuk mengunduh dan mengunggah file gambar tertentu. `GET`Metode mengembalikan file gambar sebagai string yang dikodekan base64 sebagai bagian dari output JSON, mengikuti template pemetaan yang disediakan, dalam respons 200 OK. `PUT`Metode ini mengambil gumpalan biner mentah sebagai input dan mengembalikan respons 200 OK dengan muatan kosong.

OpenAPI 3.0

```
{
   "openapi": "3.0.0",
   "info": {
      "version": "2016-10-21T17:26:28Z",
      "title": "ApiName"
   },
   "paths": {
      "/s3": {
         "get": {
            "parameters": [
               {
                  "name": "key",
                  "in": "query",
                  "required": false,
                  "schema": {
                     "type": "string"
                  }
               }
            ],
            "responses": {
               "200": {
                  "description": "200 response",
                  "content": {
                     "application/json": {
                        "schema": {
                           "$ref": "#/components/schemas/Empty"
                        }
                     }
                  }
               },
```

```
            "500": {
               "description": "500 response"
            }
         },
         "x-amazon-apigateway-integration": {
            "credentials": "arn:aws:iam::123456789012:role/binarySupportRole",
            "responses": {
               "default": {
                  "statusCode": "500"
               },
               "2\\d{2}": {
                  "statusCode": "200"
               }
            },
            "requestParameters": {
               "integration.request.path.key": "method.request.querystring.key"
            },
            "uri": "arn:aws:apigateway:us-west-2:s3:path/{key}",
            "passthroughBehavior": "when_no_match",
            "httpMethod": "GET",
            "type": "aws"
         }
      },
      "put": {
         "parameters": [
            {
               "name": "key",
               "in": "query",
               "required": false,
               "schema": {
                  "type": "string"
               }
            }
         ],
         "responses": {
            "200": {
               "description": "200 response",
               "content": {
                  "application/json": {
                     "schema": {
                        "$ref": "#/components/schemas/Empty"
                     }
                  },
                  "application/octet-stream": {
```

```
                        "schema": {
                           "$ref": "#/components/schemas/Empty"
                        }
                     }
                  }
               },
               "500": {
                  "description": "500 response"
               }
            },
            "x-amazon-apigateway-integration": {
               "credentials": "arn:aws:iam::123456789012:role/binarySupportRole",
               "responses": {
                  "default": {
                     "statusCode": "500"
                  },
                  "2\\d{2}": {
                     "statusCode": "200"
                  }
               },
               "requestParameters": {
                  "integration.request.path.key": "method.request.querystring.key"
               },
               "uri": "arn:aws:apigateway:us-west-2:s3:path/{key}",
               "passthroughBehavior": "when_no_match",
               "httpMethod": "PUT",
               "type": "aws",
               "contentHandling": "CONVERT_TO_BINARY"
            }
         }
      }
   },
   "x-amazon-apigateway-binary-media-types": [
      "application/octet-stream",
      "image/jpeg"
   ],
   "servers": [
      {
         "url": "https://abcdefghi.execute-api.us-east-1.amazonaws.com/{basePath}",
         "variables": {
            "basePath": {
              "default": "/v1"
            }
         }
```

```
      }
   ],
   "components": {
      "schemas": {
         "Empty": {
            "type": "object",
            "title": "Empty Schema"
         }
      }
   }
}
```

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2016-10-21T17:26:28Z",
    "title": "ApiName"
  },
  "host": "abcdefghi.execute-api.us-east-1.amazonaws.com",
  "basePath": "/v1",
  "schemes": [
    "https"
  ],
  "paths": {
    "/s3": {
      "get": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "key",
            "in": "query",
            "required": false,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
```

```
            "$ref": "#/definitions/Empty"
          }
        },
        "500": {
          "description": "500 response"
        }
      },
      "x-amazon-apigateway-integration": {
        "credentials": "arn:aws:iam::123456789012:role/binarySupportRole",
        "responses": {
          "default": {
            "statusCode": "500"
          },
          "2\\d{2}": {
            "statusCode": "200"            }
        },
        "requestParameters": {
          "integration.request.path.key": "method.request.querystring.key"
        },
        "uri": "arn:aws:apigateway:us-west-2:s3:path/{key}",
        "passthroughBehavior": "when_no_match",
        "httpMethod": "GET",
        "type": "aws"
      }
    },
    "put": {
      "produces": [
        "application/json", "application/octet-stream"
      ],
      "parameters": [
        {
          "name": "key",
          "in": "query",
          "required": false,
          "type": "string"
        }
      ],
      "responses": {
        "200": {
          "description": "200 response",
          "schema": {
            "$ref": "#/definitions/Empty"
          }
        },
```

```
          "500": {
            "description": "500 response"
          }
        },
        "x-amazon-apigateway-integration": {
          "credentials": "arn:aws:iam::123456789012:role/binarySupportRole",
          "responses": {
            "default": {
              "statusCode": "500"
            },
            "2\\d{2}": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.path.key": "method.request.querystring.key"
          },
          "uri": "arn:aws:apigateway:us-west-2:s3:path/{key}",
          "passthroughBehavior": "when_no_match",
          "httpMethod": "PUT",
          "type": "aws",
          "contentHandling" : "CONVERT_TO_BINARY"
        }
      }
    }
  },
  "x-amazon-apigateway-binary-media-types" : ["application/octet-stream", "image/
jpeg"],
  "definitions": {
    "Empty": {
      "type": "object",
      "title": "Empty Schema"
    }
  }
}
```

Unduh gambar dari Amazon S3

Untuk mengunduh file gambar (`image.jpg`) sebagai gumpalan biner dari 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
```

Respons yang berhasil terlihat seperti ini:

```
200 OK HTTP/1.1

[raw bytes]
```

Byte mentah dikembalikan karena `Accept` header diatur ke jenis media biner `application/octet-stream` dan dukungan biner diaktifkan untuk API.

Atau, untuk mengunduh file gambar (`image.jpg`) sebagai string yang disandikan base64 (diformat sebagai properti JSON) dari Amazon S3, tambahkan template respons ke respons integrasi 200, seperti yang ditunjukkan pada blok definisi OpenAPI berwajah tebal berikut:

```
        "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}"
              }
            }
          },
```

Permintaan untuk mengunduh file gambar terlihat seperti berikut:

```
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
```

Respons yang berhasil terlihat seperti berikut:

```
200 OK HTTP/1.1

{
```

```
  "image": "W3JhdyBieXRlc10="
}
```

Unggah gambar ke Amazon S3

Untuk mengunggah file gambar (`image.jpg`) sebagai gumpalan biner ke 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]
```

Respons yang berhasil terlihat seperti berikut:

```
200 OK HTTP/1.1
```

Untuk mengunggah file gambar (`image.jpg`) sebagai string yang dikodekan base64 ke 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=
```

Muatan input harus berupa string yang dikodekan base64 karena nilai `Content-Type` header diatur ke. `application/json` Respons yang berhasil terlihat seperti berikut:

```
200 OK HTTP/1.1
```

Akses file biner di Lambda menggunakan API Gateway API

Contoh berikut menunjukkan cara mengakses file biner AWS Lambda melalui API Gateway API. API sampel disajikan dalam file OpenAPI. Contoh kode menggunakan panggilan API API Gateway REST.

Topik

- File OpenAPI dari API sampel untuk mengakses gambar di Lambda
- Unduh gambar dari Lambda

- Unggah gambar ke Lambda

File OpenAPI dari API sampel untuk mengakses gambar di Lambda

File OpenAPI berikut menunjukkan contoh API yang menggambarkan mengunduh file gambar dari Lambda dan mengunggah file gambar ke Lambda.

OpenAPI 3.0

```
{
   "openapi": "3.0.0",
   "info": {
      "version": "2016-10-21T17:26:28Z",
      "title": "ApiName"
   },
   "paths": {
      "/lambda": {
         "get": {
            "parameters": [
               {
                  "name": "key",
                  "in": "query",
                  "required": false,
                  "schema": {
                     "type": "string"
                  }
               }
            ],
            "responses": {
               "200": {
                  "description": "200 response",
                  "content": {
                     "application/json": {
                        "schema": {
                           "$ref": "#/components/schemas/Empty"
                        }
                     }
                  }
               },
               "500": {
                  "description": "500 response"
               }
            },
```

```
            "x-amazon-apigateway-integration": {
               "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:123456789012:function:image/invocations",
               "type": "AWS",
               "credentials": "arn:aws:iam::123456789012:role/Lambda",
               "httpMethod": "POST",
               "requestTemplates": {
                  "application/json": "{\n   \"imageKey\":
 \"$input.params('key')\"\n}"
               },
               "responses": {
                  "default": {
                     "statusCode": "500"
                  },
                  "2\\d{2}": {
                     "statusCode": "200",
                     "responseTemplates": {
                        "application/json": "{\n   \"image\": \"$input.body\"\n}"
                     }
                  }
               }
            }
         },
         "put": {
            "parameters": [
               {
                  "name": "key",
                  "in": "query",
                  "required": false,
                  "schema": {
                     "type": "string"
                  }
               }
            ],
            "responses": {
               "200": {
                  "description": "200 response",
                  "content": {
                     "application/json": {
                        "schema": {
                           "$ref": "#/components/schemas/Empty"
                        }
                     },
                     "application/octet-stream": {
```

```
                        "schema": {
                           "$ref": "#/components/schemas/Empty"
                        }
                     }
                  }
               },
               "500": {
                  "description": "500 response"
               }
            },
            "x-amazon-apigateway-integration": {
               "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:123456789012:function:image/invocations",
               "type": "AWS",
               "credentials": "arn:aws:iam::123456789012:role/Lambda",
               "httpMethod": "POST",
               "contentHandling": "CONVERT_TO_TEXT",
               "requestTemplates": {
                  "application/json": "{\n   \"imageKey\": \"$input.params('key')\",
 \"image\": \"$input.body\"\n}"
               },
               "responses": {
                  "default": {
                     "statusCode": "500"
                  },
                  "2\\d{2}": {
                     "statusCode": "200"
                  }
               }
            }
         }
      }
   },
   "x-amazon-apigateway-binary-media-types": [
      "application/octet-stream",
      "image/jpeg"
   ],
   "servers": [
      {
         "url": "https://abcdefghi.execute-api.us-east-1.amazonaws.com/{basePath}",
         "variables": {
            "basePath": {
              "default": "/v1"
            }
```

```
         }
      }
   ],
   "components": {
      "schemas": {
         "Empty": {
            "type": "object",
            "title": "Empty Schema"
         }
      }
   }
}
```

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2016-10-21T17:26:28Z",
    "title": "ApiName"
  },
  "host": "abcdefghi.execute-api.us-east-1.amazonaws.com",
  "basePath": "/v1",
  "schemes": [
    "https"
  ],
  "paths": {
    "/lambda": {
      "get": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "key",
            "in": "query",
            "required": false,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
```

```
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          },
          "500": {
            "description": "500 response"
          }
        },
        "x-amazon-apigateway-integration": {
          "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:image/invocations",
          "type": "AWS",
          "credentials": "arn:aws:iam::123456789012:role/Lambda",
          "httpMethod": "POST",
          "requestTemplates": {
            "application/json": "{\n   \"imageKey\": \"$input.params('key')\"\n}"
          },
          "responses": {
            "default": {
              "statusCode": "500"
            },
            "2\\d{2}": {
              "statusCode": "200",
              "responseTemplates": {
                "application/json": "{\n   \"image\": \"$input.body\"\n}"
              }
            }
          }
        }
      },
      "put": {
        "produces": [
          "application/json", "application/octet-stream"
        ],
        "parameters": [
          {
            "name": "key",
            "in": "query",
            "required": false,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
```

```
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          },
          "500": {
            "description": "500 response"
          }
        },
        "x-amazon-apigateway-integration": {
          "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:image/invocations",
          "type": "AWS",
          "credentials": "arn:aws:iam::123456789012:role/Lambda",
          "httpMethod": "POST",
          "contentHandling" : "CONVERT_TO_TEXT",
          "requestTemplates": {
            "application/json": "{\n   \"imageKey\": \"$input.params('key')\",
 \"image\": \"$input.body\"\n}"
          },
          "responses": {
            "default": {
              "statusCode": "500"
            },
            "2\\d{2}": {
              "statusCode": "200"
            }
          }
        }
      }
    }
  },
  "x-amazon-apigateway-binary-media-types" : ["application/octet-stream", "image/
jpeg"],
  "definitions": {
    "Empty": {
      "type": "object",
      "title": "Empty Schema"
    }
  }
}
```

Unduh gambar dari Lambda

Untuk mengunduh file gambar (`image.jpg`) sebagai gumpalan biner dari 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
```

Respons yang berhasil terlihat seperti berikut:

```
200 OK HTTP/1.1

[raw bytes]
```

Untuk mengunduh file gambar (`image.jpg`) sebagai string yang dikodekan base64 (diformat sebagai properti JSON) dari 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
```

Respons yang berhasil terlihat seperti berikut:

```
200 OK HTTP/1.1

{
  "image": "W3JhdyBieXRlc10="
}
```

Unggah gambar ke Lambda

Untuk mengunggah file gambar (`image.jpg`) sebagai gumpalan biner ke 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]
```

Respons yang berhasil terlihat seperti berikut:

```
200 OK
```

Untuk mengunggah file gambar (`image.jpg`) sebagai string yang dikodekan base64 ke 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=
```

Respons yang berhasil terlihat seperti berikut:

```
200 OK
```

Memanggil REST API di Amazon API Gateway

Untuk memanggil API yang diterapkan, klien mengirimkan permintaan ke URL untuk layanan komponen API Gateway untuk eksekusi API, yang dikenal sebagai`execute-api`.

URL dasar untuk REST API dalam format berikut:

```
https://restapi_id.execute-api.region.amazonaws.com/stage_name/
```

di mana *`restapi_id`* adalah pengidentifikasi API, *`wilayah adalah AWS Wilayah`*, dan *`stage_name adalah nama`* panggung penerapan API.

> **Important**
>
> Sebelum Anda dapat menjalankan API, Anda harus menerapkannya di API Gateway. Untuk petunjuk tentang penerapan API, lihatMenerapkan REST API di Amazon API Gateway.

Topik

- Mendapatkan URL pemanggilan API
- Memanggil API
- Menggunakan konsol API Gateway untuk menguji metode REST API
- Panggil REST API melalui SDK yang dihasilkan
- Cara memanggil API pribadi

Mendapatkan URL pemanggilan API

Anda dapat menggunakan konsol, definisi OpenAPI AWS CLI, atau yang diekspor untuk mendapatkan URL pemanggilan API.

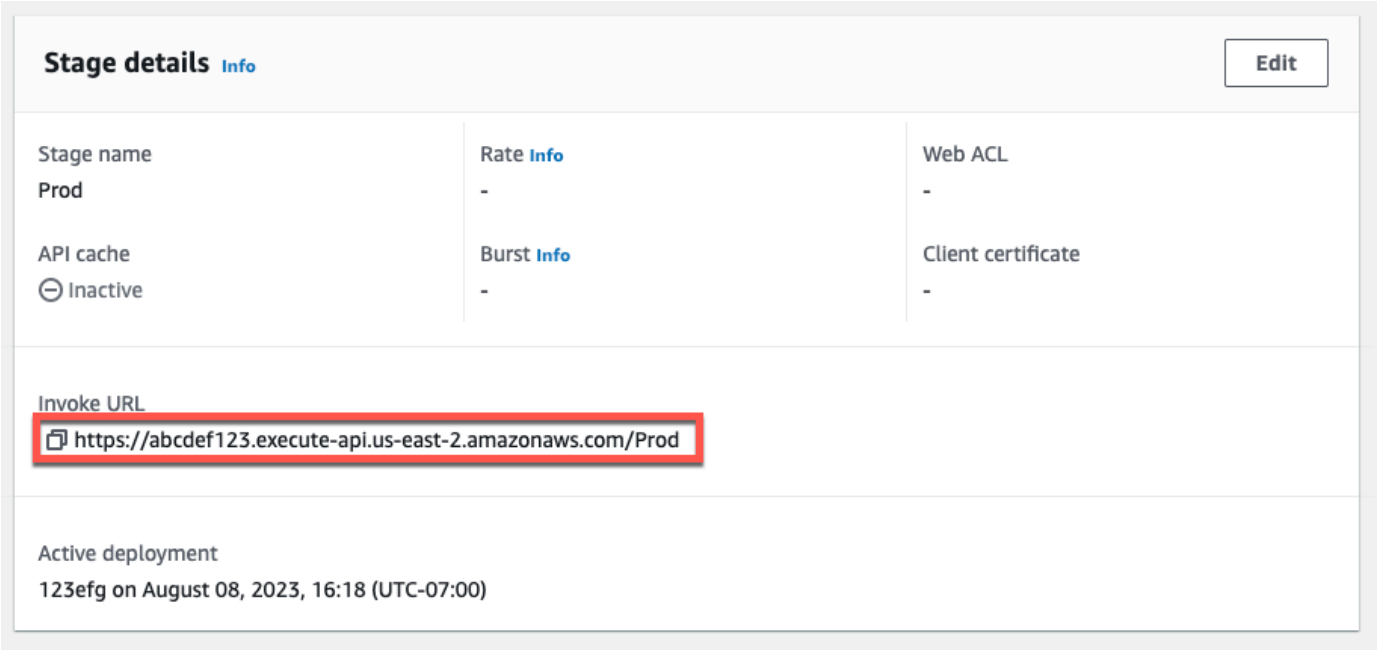
Mendapatkan URL pemanggilan API menggunakan konsol

Prosedur berikut menunjukkan cara mendapatkan URL pemanggilan API di konsol REST API.

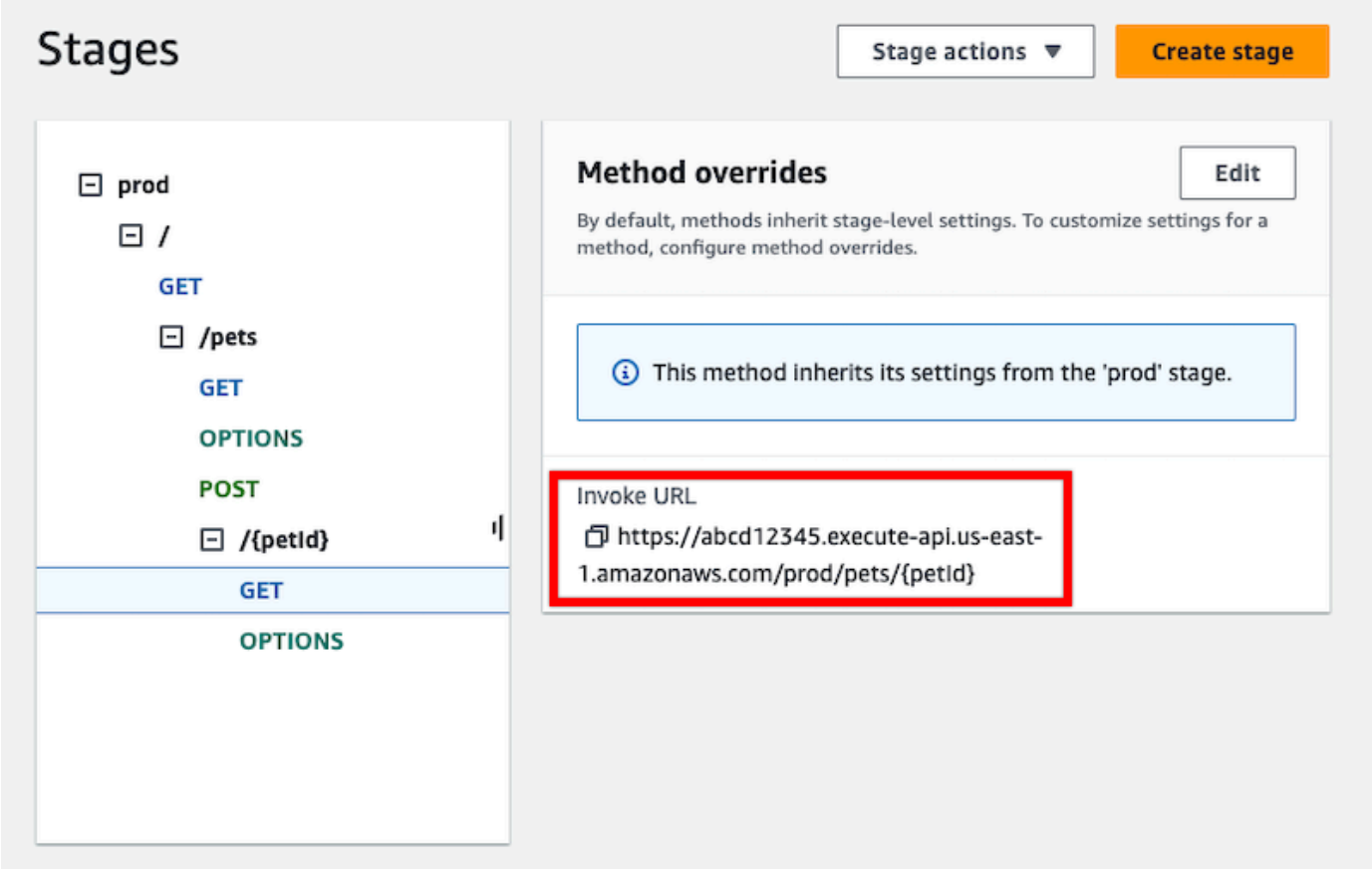
Untuk mendapatkan URL pemanggilan API menggunakan konsol REST API

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API yang diterapkan.
3. Dari panel navigasi utama, pilih Stage.
4. Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda.

   URL ini untuk sumber daya root API Anda.



5. Untuk mendapatkan URL pemanggilan API untuk sumber daya lain di API Anda, perluas tahapan di bawah panel navigasi sekunder, lalu pilih metode.
6. Pilih ikon salin untuk menyalin URL pemanggilan tingkat sumber daya API Anda.



Mendapatkan URL pemanggilan API menggunakan AWS CLI

Prosedur berikut menunjukkan cara mendapatkan URL pemanggilan API menggunakan. AWS CLI

Untuk mendapatkan URL pemanggilan API menggunakan AWS CLI

1. Gunakan perintah berikut untuk mendapatkan`rest-api-id`. Perintah ini mengembalikan semua `rest-api-id` nilai di Wilayah Anda. Untuk informasi lebih lanjut, lihat get-rest-apis.

```
aws apigateway get-rest-apis
```

2. Ganti contoh `rest-api-id` dengan contoh Anda`rest-api-id`, ganti contoh *`{stage-name}`* dengan *`{stage-name} Anda, dan ganti {region}, dengan Region`* Anda.

```
https://{restapi_id}.execute-api.{region}.amazonaws.com/{stage_name}/
```

Memperoleh URL pemanggilan API menggunakan file definisi OpenAPI yang diekspor dari API

Anda juga dapat membuat URL root dengan menggabungkan `host` dan `basePath` bidang file definisi OpenAPI yang diekspor dari API. Untuk petunjuk tentang cara mengekspor API Anda, lihatthe section called “Ekspor REST API”.

Memanggil API

Anda dapat memanggil API yang digunakan menggunakan browser, curl, atau aplikasi lain, seperti Postman.

Selain itu, Anda dapat menggunakan konsol API Gateway untuk menguji panggilan API. Pengujian menggunakan `TestInvoke` fitur API Gateway, yang memungkinkan pengujian API sebelum API diterapkan. Untuk informasi selengkapnya, lihat the section called “Gunakan konsol untuk menguji metode REST API”.

> **Note**
>
> Nilai parameter string kueri dalam URL pemanggilan tidak dapat berisi. `%%`

Memanggil API menggunakan browser web

Jika API Anda mengizinkan akses anonim, Anda dapat menggunakan browser web apa pun untuk menjalankan metode apa pun`GET`. Masukkan URL pemanggilan lengkap di bilah alamat browser.

Untuk metode lain atau panggilan yang diperlukan otentikasi, Anda harus menentukan payload atau menandatangani permintaan. Anda dapat menangani ini dalam skrip di belakang halaman HTML atau dalam aplikasi klien menggunakan salah satu AWS SDK.

Memanggil API menggunakan curl

Anda dapat menggunakan alat seperti curl di terminal Anda untuk memanggil API Anda. Contoh perintah curl berikut memanggil metode GET pada `getUsers` sumber daya `prod` tahap API.

Linux or Macintosh

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

Windows

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

Menggunakan konsol API Gateway untuk menguji metode REST API

Gunakan konsol API Gateway untuk menguji metode REST API.

Topik

- Prasyarat
- Uji metode dengan konsol API Gateway

Prasyarat

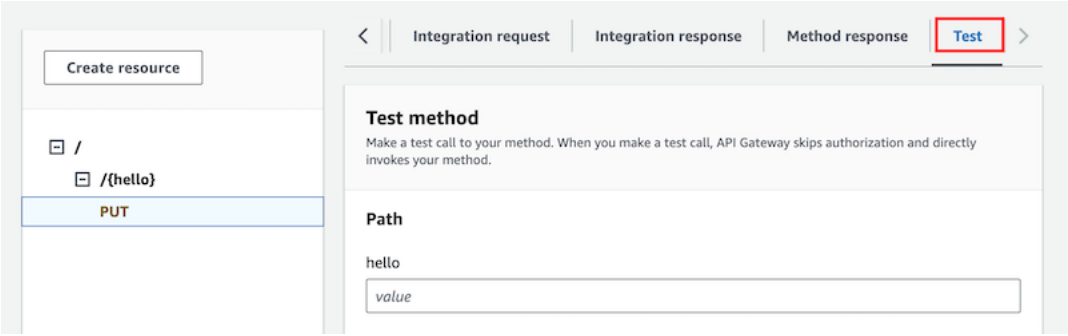
- Anda harus menentukan pengaturan untuk metode yang ingin Anda uji. Ikuti instruksi di Siapkan metode REST API di API Gateway.

Uji metode dengan konsol API Gateway

> **Important**
>
> Metode pengujian dengan konsol API Gateway dapat mengakibatkan perubahan pada sumber daya yang tidak dapat dibatalkan. Menguji metode dengan konsol API Gateway sama dengan memanggil metode di luar konsol API Gateway. Misalnya, jika Anda menggunakan konsol API Gateway untuk memanggil metode yang menghapus sumber daya API, jika pemanggilan metode berhasil, sumber daya API akan dihapus.

Untuk menguji suatu metode

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih REST API.
3. Di panel Resources, pilih metode yang ingin Anda uji.
4. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.



Masukkan nilai di salah satu kotak yang ditampilkan (seperti string Kueri, Header, dan badan Permintaan). Konsol menyertakan nilai-nilai ini dalam permintaan metode dalam bentuk aplikasi/json default.

Untuk opsi tambahan yang mungkin perlu Anda tentukan, hubungi pemilik API.

5. Pilih Uji. Informasi berikut akan ditampilkan:

- Permintaan adalah jalur sumber daya yang dipanggil untuk metode.
- Status adalah kode status HTTP respon.
- Latensi adalah waktu antara penerimaan permintaan dari penelepon dan respons yang dikembalikan.
- Response body adalah badan respon HTTP.
- Response header adalah header respon HTTP.

> **Tip**
>
> Bergantung pada pemetaan, kode status HTTP, badan respons, dan header respons mungkin berbeda dari yang dikirim dari fungsi Lambda, proxy HTTP, atau proxy layanan. AWS

- Log menunjukkan simulasi entri Amazon CloudWatch Logs yang akan ditulis jika metode ini dipanggil di luar konsol API Gateway.

> **Note**
>
> Meskipun entri CloudWatch Log disimulasikan, hasil pemanggilan metode adalah nyata.

Selain menggunakan konsol API Gateway, Anda dapat menggunakan AWS CLI atau AWS SDK untuk API Gateway untuk menguji pemanggilan metode. Untuk melakukannya dengan menggunakanAWS CLI, lihat test-invoke-method.

Panggil REST API melalui SDK yang dihasilkan

Bagian ini menunjukkan cara memanggil API melalui SDK yang dihasilkan di aplikasi klien yang ditulis dalam Java, Java untuk Android, Ruby, JavaScript Objective-C, dan Swift.

Topik

- Menggunakan Java SDK yang dihasilkan oleh API Gateway untuk REST API
- Menggunakan Android SDK yang dihasilkan oleh API Gateway untuk REST API
- Menggunakan JavaScript SDK yang dihasilkan oleh API Gateway untuk REST API
- Menggunakan Ruby SDK yang dihasilkan oleh API Gateway untuk REST API
- Menggunakan iOS SDK yang dihasilkan oleh API Gateway untuk REST API di Objective-C atau Swift

Menggunakan Java SDK yang dihasilkan oleh API Gateway untuk REST API

Pada bagian ini, kami menguraikan langkah-langkah untuk menggunakan Java SDK yang dihasilkan oleh API Gateway untuk REST API, dengan menggunakanKalkulator SederhanaAPI sebagai contoh. Sebelum melanjutkan, Anda harus menyelesaikan langkahMenghasilkan SDK untuk API menggunakan konsol API Gateway.

Untuk menginstal dan menggunakan Java SDK yang dihasilkan oleh API Gateway

1. Ekstrak isi file.zip yang dihasilkan API Gateway-yang Anda unduh sebelumnya.
2. Unduh dan instalApache Maven(harus versi 3.5 atau yang lebih baru).
3. Unduh dan instalJDK 8.
4. MengaturJAVA_HOMEvariabel lingkungan.
5. Pergi ke folder SDK unzipped di mana file pom.xml berada. Folder inigenerated-codesecara default. Jalankanmvn installperintah untuk menginstal file artefak dikompilasi ke repositori Maven lokal Anda. Ini menciptakantargetfolder yang berisi pustaka SDK yang dikompilasi.
6. Ketik perintah berikut dalam direktori kosong untuk membuat rintisan proyek klien untuk memanggil API menggunakan perpustakaan SDK diinstal.

```
mvn -B archetype:generate \
    -DarchetypeGroupdId=org.apache.maven.archetypes \
    -DgroupId=examples.aws.apig.simpleCalc.sdk.app \
    -DartifactId=SimpleCalc-sdkClient
```

> **Note**
>
> Pemisah\dalam perintah sebelumnya disertakan untuk dibaca. Seluruh perintah harus dalam satu baris tanpa pemisah.

Perintah ini membuat rintisan aplikasi. Aplikasi rintisan berisipom.xmlfile dansrcfolder di bawah direktori root proyek (*SimpleCalc-SDKClient*dalam perintah sebelumnya). Awalnya, ada dua file sumber:src/main/java/*{package-path}*/App.javadansrc/test/java/*{package-path}*/AppTest.java. Dalam contoh ini,*{paket-path}*adalahexamples/aws/apig/simpleCalc/sdk/app. Path paket ini berasal dariDarchetypeGroupdIdnilai. Anda dapat menggunakanApp.javafile sebagai template untuk aplikasi klien Anda, dan Anda dapat menambahkan orang lain dalam folder yang sama jika diperlukan. Anda dapat menggunakanAppTest.javafile sebagai template pengujian unit untuk aplikasi Anda, dan Anda dapat menambahkan file kode tes lainnya ke folder tes yang sama seperti yang diperlukan.

7. Memperbarui dependensi paket dalam yang dihasilkanpom.xmlfile ke berikut, menggantikan proyek AndagroupId,artifactId,version, dannameproperti, jika perlu:

```
<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**
>
> Ketika versi yang lebih baru dari artefak tergantung`aws-java-sdk-core`tidak kompatibel dengan versi yang ditentukan di atas (`1.11.94`), Anda harus memperbarui`<version>`tag ke versi baru.

8. Selanjutnya, kami menunjukkan cara memanggil API menggunakan SDK dengan memanggilgetABOp(GetABOpRequest req),getApiRoot(GetApiRootRequest req), danpostApiRoot(PostApiRootRequest req)metode SDK. Metode ini sesuai denganGET /{a}/{b}/{op},GET /?a={x}&b={y}&op={operator}, danPOST /metode, dengan muatan{"a": x, "b": y, "op": "operator"}Permintaan API, masing-masing.

   MemperbaruiApp.javafile sebagai berikut:

```
package examples.aws.apig.simpleCalc.sdk.app;

import java.io.IOException;

import com.amazonaws.opensdk.config.ConnectionConfiguration;
import com.amazonaws.opensdk.config.TimeoutConfiguration;

import examples.aws.apig.simpleCalc.sdk.*;
import examples.aws.apig.simpleCalc.sdk.model.*;
import examples.aws.apig.simpleCalc.sdk.SimpleCalcSdk.*;

public class App
{
    SimpleCalcSdk sdkClient;

    public App() {
        initSdk();
    }

    // The configuration settings are for illustration purposes and may not be a
 recommended best practice.
    private void initSdk() {
        sdkClient = SimpleCalcSdk.builder()
              .connectionConfiguration(
                  new ConnectionConfiguration()
                        .maxConnections(100)
                        .connectionMaxIdleMillis(1000))
              .timeoutConfiguration(
                  new TimeoutConfiguration()
                        .httpRequestTimeout(3000)
                        .totalExecutionTimeout(10000)
                        .socketTimeout(2000))
        .build();

    }
```

```
   // Calling shutdown is not necessary unless you want to exert explicit control
of this resource.
   public void shutdown() {
       sdkClient.shutdown();
   }

   // GetABOpResult getABOp(GetABOpRequest getABOpRequest)
   public Output getResultWithPathParameters(String x, String y, String operator)
{
    operator = operator.equals("+") ? "add" : operator;
    operator = operator.equals("/") ? "div" : operator;

       GetABOpResult abopResult = sdkClient.getABOp(new
GetABOpRequest().a(x).b(y).op(operator));
       return abopResult.getResult().getOutput();
   }

   public Output getResultWithQueryParameters(String a, String b, String op) {
       GetApiRootResult rootResult = sdkClient.getApiRoot(new
GetApiRootRequest().a(a).b(b).op(op));
       return rootResult.getResult().getOutput();
   }

   public Output getResultByPostInputBody(Double x, Double y, String o) {
    PostApiRootResult postResult = sdkClient.postApiRoot(
     new PostApiRootRequest().input(new Input().a(x).b(y).op(o)));
    return postResult.getResult().getOutput();
   }

   public static void main( String[] args )
   {
       System.out.println( "Simple calc" );
       // to begin
       App calc = new App();

       // call the SimpleCalc API
       Output res = calc.getResultWithPathParameters("1", "2", "-");
       System.out.printf("GET /1/2/-: %s\n", res.getC());

       // Use the type query parameter
       res = calc.getResultWithQueryParameters("1", "2", "+");
       System.out.printf("GET /?a=1&b=2&op=+: %s\n", res.getC());

       // Call POST with an Input body.
```

```
        res = calc.getResultByPostInputBody(1.0, 2.0, "*");
        System.out.printf("PUT /\n\n{\"a\":1, \"b\":2,\"op\":\"*\"}\n %s\n",
 res.getC());


    }
}
```

Dalam contoh sebelumnya, pengaturan konfigurasi yang digunakan untuk instantiate klien SDK adalah untuk tujuan ilustrasi dan tidak selalu direkomendasikan praktik terbaik. Juga, meneleponsdkClient.shutdown()bersifat opsional, terutama jikaAnda memerlukan kontrol yang tepat kapan harus membebaskan sumber daya.

Kami telah menunjukkan pola penting untuk memanggil API menggunakan Java SDK. Anda dapat memperluas instruksi untuk memanggil metode API lainnya.

Menggunakan Android SDK yang dihasilkan oleh API Gateway untuk REST API

Pada bagian ini, kami akan menguraikan langkah-langkah untuk menggunakan Android SDK yang dihasilkan oleh API Gateway untuk REST API. Sebelum melanjutkan lebih jauh, Anda harus sudah menyelesaikan langkah-langkah diMenghasilkan SDK untuk API menggunakan konsol API Gateway.

> **Note**
>
> SDK yang dihasilkan tidak kompatibel dengan Android 4.4 dan versi sebelumnya. Untuk informasi selengkapnya, lihat the section called “Catatan penting”.

Untuk menginstal dan menggunakan SDK Android yang dihasilkan oleh API Gateway

1. Ekstrak isi file.zip yang dihasilkan API Gateway-yang Anda unduh sebelumnya.
2. Unduh dan instalApache Maven(sebaiknya versi 3.x).
3. Unduh dan instalJDK.
4. MengaturJAVA_HOMEVariabel Lingkungan.
5. Jalankanmvn installperintah untuk menginstal file artefak dikompilasi ke repositori Maven lokal Anda. Ini menciptakantargetfolder yang berisi pustaka SDK dikompilasi.

6. Salin file SDK (nama yang berasal dariId ArtifactdanVersi ArtifactAnda tentukan saat menghasilkan SDK, misalnya,`simple-calcsdk-1.0.0.jar`) dari`target`folder, bersama dengan semua perpustakaan lain dari`target/lib`folder, ke proyek Anda`lib`folder.

   Jika Anda menggunakan Android Studio, buat`libs`folder di bawah modul aplikasi klien Anda dan salin file.jar yang diperlukan ke dalam folder ini. Pastikan bagian dependensi dalam file gradle modul berisi yang berikut ini.

   ```
       compile fileTree(include: ['*.jar'], dir: 'libs')
       compile fileTree(include: ['*.jar'], dir: 'app/libs')
   ```

   Pastikan tidak ada duplikasi file.jar dideklarasikan.

7. Menggunakan`ApiClientFactory`kelas untuk menginisialisasi SDK API Gateway-dihasilkan. Misalnya:

   ```
   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. Untuk menggunakan penyedia kredensi Amazon Cognito untuk mengotorisasi panggilan ke API Anda, gunakan`ApiClientFactory`kelas untuk meneruskan satu set`AWS`kredensi dengan menggunakan SDK yang dihasilkan oleh API Gateway, seperti yang ditunjukkan dalam contoh berikut.

```
// 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. Untuk mengatur kunci API dengan menggunakan API Gateway- dihasilkan SDK, gunakan kode yang mirip dengan berikut ini.

```
ApiClientFactory factory = new ApiClientFactory()
  .apiKey("YOUR_API_KEY");
```

Menggunakan JavaScript SDK yang dihasilkan oleh API Gateway untuk REST API

> **Note**
>
> Instruksi ini menganggap Anda telah menyelesaikan instruksi diMenghasilkan SDK untuk API menggunakan konsol API Gateway.

> **Important**
>
> Jika API Anda hanya memiliki metode APAPUN didefinisikan, paket SDK yang dihasilkan tidak akan berisi`apigClient.js`file, dan Anda akan perlu untuk menentukan metode APAPUN sendiri.

Untuk menginstal, memulai dan memanggil JavaScript SDK yang dihasilkan oleh API Gateway untuk REST API

1. Ekstrak isi file.zip yang dihasilkan API Gateway-yang Anda unduh sebelumnya.
2. Aktifkan cross-origin resource sharing (CORS) untuk semua metode SDK yang dihasilkan oleh API Gateway akan memanggil. Untuk petunjuk, lihat Mengaktifkan CORS untuk sumber daya REST API.
3. Di halaman web Anda, sertakan referensi ke skrip berikut.

```
<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. Dalam kode Anda, inisialisasi SDK yang dihasilkan oleh API Gateway dengan menggunakan kode yang mirip dengan berikut ini.

   ```
   var apigClient = apigClientFactory.newClient();
   ```

   Untuk menginisialisasi SDK yang dihasilkan oleh API Gateway denganAWSkredensial, menggunakan kode serupa dengan yang berikut ini. Jika Anda menggunakanAWSkredensia, semua permintaan ke API akan ditandatangani.

   ```
   var apigClient = apigClientFactory.newClient({
     accessKey: 'ACCESS_KEY',
     secretKey: 'SECRET_KEY',
   });
   ```

   Untuk menggunakan kunci API dengan SDK yang dihasilkan oleh API Gateway, teruskan kunci API sebagai parameter ke`Factory`objek dengan menggunakan kode serupa dengan berikut ini. Jika Anda menggunakan kunci API, itu ditentukan sebagai bagian dari`x-api-key`header dan semua permintaan ke API akan ditandatangani. Ini berarti Anda harus mengatur header CORS Accept yang sesuai untuk setiap permintaan.

   ```
   var apigClient = apigClientFactory.newClient({
     apiKey: 'API_KEY'
   });
   ```

5. Panggil metode API di API Gateway dengan menggunakan kode yang mirip dengan berikut ini. Setiap panggilan mengembalikan janji dengan callback sukses dan kegagalan.

   ```
   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.
    });
```

Di sini,*methodName*dibangun dari jalur sumber daya permintaan metode dan kata kerja HTTP. Untuk SimpleCalc API, metode SDK untuk metode API

```
1. GET /?a=...&b=...&op=...
2. POST /

   { "a": ..., "b": ..., "op": ...}
3. GET /{a}/{b}/{op}
```

metode SDK yang sesuai adalah sebagai berikut:

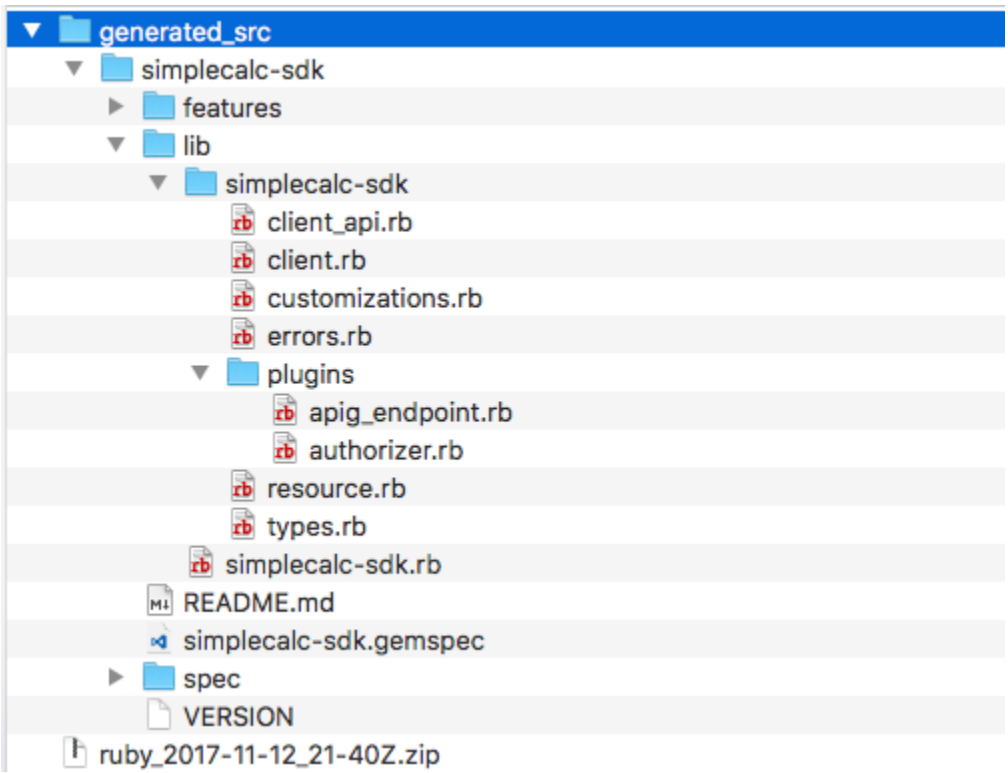
```
1. rootGet(params);      // where params={"a": ..., "b": ..., "op": ...} is
 resolved to the query parameters
2. rootPost(null, body); // where body={"a": ..., "b": ..., "op": ...}
3. aBOpGet(params);      // where params={"a": ..., "b": ..., "op": ...} is
 resolved to the path parameters
```

Menggunakan Ruby SDK yang dihasilkan oleh API Gateway untuk REST API

> **Note**
>
> Instruksi ini menganggap Anda sudah menyelesaikan instruksi diMenghasilkan SDK untuk API menggunakan konsol API Gateway.

Untuk menginstal, instantiate, dan memanggil Ruby SDK yang dihasilkan oleh API Gateway untuk REST API

1. Unzip file Ruby SDK yang diunduh. Sumber SDK yang dihasilkan ditunjukkan sebagai berikut.



2. Buat Ruby Gem dari sumber SDK yang dihasilkan, menggunakan perintah shell berikut di jendela terminal:

```
# change to /simplecalc-sdk directory
cd simplecalc-sdk

# build the generated gem
gem build simplecalc-sdk.gemspec
```

Setelah ini,simplecalc-sdk-1.0.0.gemmenjadi tersedia.

3. Instal permata:

```
gem install simplecalc-sdk-1.0.0.gem
```

4. Membuat aplikasi klien. Instantiate dan inisialisasi klien Ruby SDK di aplikasi:

```
require 'simplecalc-sdk'
client = SimpleCalc::Client.new(
    http_wire_trace: true,
    retry_limit: 5,
    http_read_timeout: 50
)
```

Jika API memiliki otorisasi`AWS_IAM`jenis dikonfigurasi, Anda dapat menyertakan pemanggilAWSkredensi dengan memasok`accessKey`dan`secretKey`selama inisialisasi:

```
require 'pet-sdk'
client = Pet::Client.new(
    http_wire_trace: true,
    retry_limit: 5,
    http_read_timeout: 50,
    access_key_id: 'ACCESS_KEY',
    secret_access_key: 'SECRET_KEY'
)
```

5. Lakukan panggilan API melalui SDK di aplikasi.

> **Tip**
>
> Jika Anda tidak terbiasa dengan konvensi panggilan metode SDK, Anda dapat meninjau`client.rb`file dalam SDK yang dihasilkan`lib`folder Folder berisi dokumentasi setiap panggilan metode API yang didukung.

Untuk menemukan operasi yang didukung:

```
# to show supported operations:
puts client.operation_names
```

Hal ini menghasilkan tampilan berikut, sesuai dengan metode API`GET /?a={.}&b={.}&op={.}`,`GET /{a}/{b}/{op}`, dan`POST /`, ditambah muatan`{a:"…", b:"…", op:"…"}`format, masing-masing:

```
[:get_api_root, :get_ab_op, :post_api_root]
```

Untuk memanggil`GET /?a=1&b=2&op=+`Metode API, memanggil berikut metode Ruby SDK:

```
resp = client.get_api_root({a:"1", b:"2", op:"+"})
```

Untuk memanggil`POST /`Metode API dengan muatan`{a: "1", b: "2", "op": "+"}`, hubungi metode Ruby SDK berikut:

```
resp = client.post_api_root(input: {a:"1", b:"2", op:"+"})
```

Untuk memanggil`GET /1/2/+`Metode API, panggil metode Ruby SDK berikut:

```
resp = client.get_ab_op({a:"1", b:"2", op:"+"})
```

Panggilan metode SDK yang sukses mengembalikan respons berikut:

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

Menggunakan iOS SDK yang dihasilkan oleh API Gateway untuk REST API di Objective-C atau Swift

Dalam tutorial ini, kita akan menunjukkan bagaimana menggunakan SDK iOS yang dihasilkan oleh API Gateway untuk REST API di aplikasi Objective-C atau Swift untuk memanggil API yang

mendasarinya. Kami akan menggunakanAPI SimpleCalcsebagai contoh untuk menggambarkan topik berikut:

- Bagaimana menginstalAWSKomponen SDK seluler ke dalam proyek Xcode Anda
- Cara membuat objek klien API sebelum memanggil metode API
- Cara memanggil metode API melalui metode SDK yang sesuai pada objek klien API
- Bagaimana mempersiapkan input metode dan mengurai hasilnya menggunakan kelas model yang sesuai dari SDK

Topik

- Gunakan iOS SDK (Objective-C) yang dihasilkan untuk memanggil API
- Gunakan iOS SDK (Swift) yang dihasilkan untuk memanggil API

Gunakan iOS SDK (Objective-C) yang dihasilkan untuk memanggil API

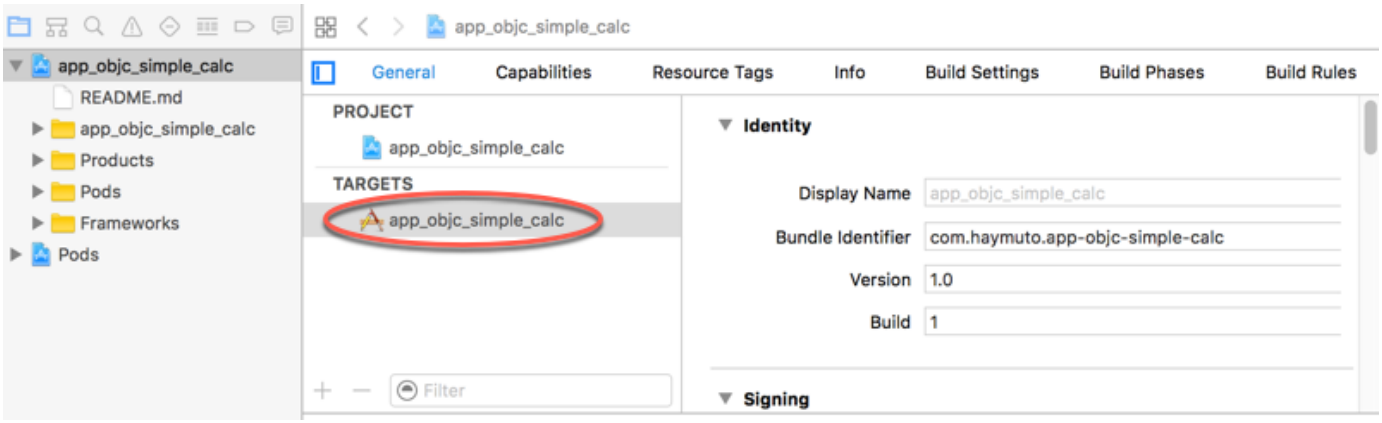
Sebelum memulai prosedur berikut, Anda harus menyelesaikan langkah-langkah diMenghasilkan SDK untuk API menggunakan konsol API Gatewayuntuk iOS di Objective-C dan download file.zip dari SDK yang dihasilkan.

InstalAWSSDK seluler dan SDK iOS yang dihasilkan oleh API Gateway dalam proyek Objective-C

Prosedur berikut menjelaskan cara menginstal SDK.

Untuk menginstal dan menggunakan SDK iOS yang dihasilkan oleh API Gateway di Objective-C

1. Ekstrak isi file.zip yang dihasilkan API Gateway-yang Anda unduh sebelumnya. MenggunakanSimpleCalc API, Anda mungkin ingin mengubah nama folder SDK unzipped menjadi sesuatu seperti**`sdk_objc_simple_calc`**. Dalam folder SDK ini ada`README.md`file dan a`Podfile`berkas. Parameter`README.md`berisi petunjuk untuk menginstal dan menggunakan SDK. Tutorial ini memberikan rincian tentang petunjuk ini. Instalasi memanfaatkanCocoaPodsuntuk mengimpor pustaka API Gateway yang diperlukan dan dependen lainnyaAWSKomponen Mobile SDK. Anda harus memperbaruui`Podfile`untuk mengimpor SDK ke proyek Xcode aplikasi Anda. Folder SDK yang belum diarsipkan juga berisi`generated-src`folder yang berisi kode sumber SDK API Anda.
2. Luncurkan Xcode dan buat proyek iOS Objective-C baru. Perhatikan target proyek. Anda perlu mengaturnya di`Podfile`.



3. Untuk mengimporAWS Mobile SDK for iOSke proyek Xcode dengan menggunakan CocoaPods, lakukan hal berikut:

   a. Instal CocoaPods dengan menjalankan perintah berikut di jendela terminal:

   ```
   sudo gem install cocoapods
   pod setup
   ```

   b. SalinPodfilefile dari folder SDK yang diekstrak ke dalam direktori yang sama yang berisi file proyek Xcode Anda. Ganti blok berikut:

   ```
   target '<YourXcodeTarget>' do
       pod 'AWSAPIGateway', '~> 2.4.7'
   end
   ```

   dengan nama target proyek Anda:

   ```
   target 'app_objc_simple_calc' do
       pod 'AWSAPIGateway', '~> 2.4.7'
   end
   ```

   Jika proyek Xcode Anda sudah berisi file bernamaPodfile, tambahkan baris kode berikut:

   ```
   pod 'AWSAPIGateway', '~> 2.4.7'
   ```

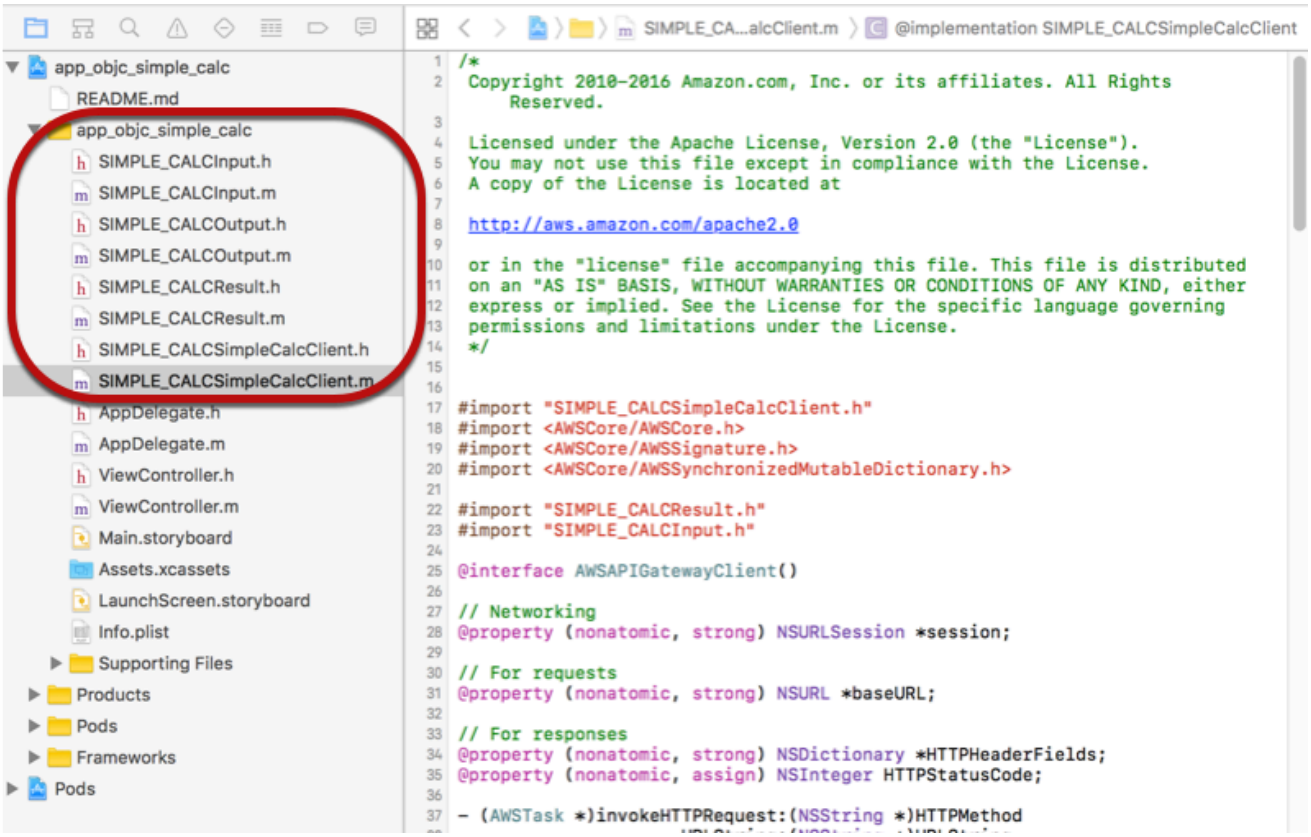
   c. Buka jendela terminal dan jalankan perintah berikut:

   ```
   pod install
   ```

Ini menginstal komponen API Gateway dan dependen lainnyaAWSKomponen Mobile SDK.

d. Tutup proyek Xcode dan kemudian buka `.xcworkspace`file untuk meluncurkan kembali Xcode.

e. Tambahkan semua `.h`dan `.m`file dari SDK yang diekstrak`generated-src`direktori ke proyek Xcode Anda.



Untuk mengimporAWS Mobile SDK for iOSObjective-C ke proyek Anda dengan mengunduh secara eksplisitAWSMobile SDK atau menggunakanKartago, ikuti petunjuk diReadme.mdberkas. Pastikan untuk hanya menggunakan salah satu dari opsi ini untuk mengimporAWSMobile SDK.

Memanggil metode API menggunakan iOS SDK yang dihasilkan oleh API Gateway dalam proyek Objective-C

Ketika Anda membuat SDK dengan awalan`SIMPLE_CALC`untuk iniSimpleCalc APIdengan dua model untuk input (`Input`) dan output (`Result`) dari metode, di SDK, kelas klien API yang dihasilkan menjadi`SIMPLE_CALCSimpleCalcClient`dan kelas data yang sesuai

adalah`SIMPLE_CALCInput`dan`SIMPLE_CALCResult`, masing-masing. Permintaan dan tanggapan API dipetakan ke metode SDK sebagai berikut:

- Permintaan API

```
GET /?a=...&b=...&op=...
```

  menjadi metode SDK

```
(AWSTask *)rootGet:(NSString *)op a:(NSString *)a b:(NSString *)b
```

  Parameter`AWSTask.result`properti adalah`SIMPLE_CALCResult`ketik jika`ResultModel` ditambahkan ke respon metode. Jika tidak, properti adalah dari`NSDictionary`ketik.

- Permintaan API

```
POST /

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

  menjadi metode SDK

```
(AWSTask *)rootPost:(SIMPLE_CALCInput *)body
```

- Permintaan API

```
GET /{a}/{b}/{op}
```

  menjadi metode SDK

```
(AWSTask *)aBOpGet:(NSString *)a b:(NSString *)b op:(NSString *)op
```

Prosedur berikut menjelaskan cara memanggil metode API dalam kode sumber aplikasi Objective-C; misalnya, sebagai bagian dari`viewDidLoad`delegasi dalam`ViewController.m`berkas.

Untuk memanggil API melalui iOS SDK yang dihasilkan oleh API Gateway

1. Impor file header kelas klien API untuk membuat API client class callable di aplikasi:

```
#import "SIMPLE_CALCSimpleCalc.h"
```

Parameter`#import`pernyataan juga impor`SIMPLE_CALCInput.h`dan`SIMPLE_CALCResult.h`untuk dua kelas model.

2. Instantiate kelas klien API:

```
SIMPLE_CALCSimpleCalcClient *apiInstance = [SIMPLE_CALCSimpleCalcClient
 defaultClient];
```

Untuk menggunakan Amazon Cognito dengan API, atur`defaultServiceConfiguration`properti pada default`AWSServiceManager`objek, seperti yang ditunjukkan pada berikut, sebelum memanggil`defaultClient`metode untuk membuat objek klien API (ditunjukkan dalam contoh sebelumnya):

```
AWSCognitoCredentialsProvider *creds = [[AWSCognitoCredentialsProvider alloc]
 initWithRegionType:AWSRegionUSEast1 identityPoolId:your_cognito_pool_id];
AWSServiceConfiguration *configuration = [[AWSServiceConfiguration alloc]
 initWithRegion:AWSRegionUSEast1 credentialsProvider:creds];
AWSServiceManager.defaultServiceManager.defaultServiceConfiguration =
 configuration;
```

3. Memanggil`GET /?a=1&b=2&op=+`Metode untuk melakukan`1+2`:

```
[[apiInstance rootGet: @"+" a:@"1" b:@"2"] continueWithBlock:^id _Nullable(AWSTask
 * _Nonnull task) {
    _textField1.text = [self handleApiResponse:task];
    return nil;
}];
```

dimana fungsi pembantu`handleApiResponse:task`memformat hasilnya sebagai string yang akan ditampilkan dalam bidang teks (`_textField1`).

```
- (NSString *)handleApiResponse:(AWSTask *)task {
    if (task.error != nil) {
        return [NSString stringWithFormat: @"Error: %@", task.error.description];
```

```
    } else if (task.result != nil && [task.result isKindOfClass:[SIMPLE_CALCResult
 class]]) {
        return [NSString stringWithFormat:@"%@ %@ %@ = %@\n",task.result.input.a,
 task.result.input.op, task.result.input.b, task.result.output.c];
    }
    return nil;
}
```

Tampilan yang dihasilkan adalah`1 + 2 = 3`.

4. Memanggil`POST /`dengan muatan untuk melakukan`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;
    }];
```

Tampilan yang dihasilkan adalah`1 - 2 = -1`.

5. Memanggil`GET /{a}/{b}/{op}`melakukan`1/2`:

```
[[apiInstance aBOpGet:@"1" b:@"2" op:@"div"] continueWithBlock:^id
 _Nullable(AWSTask * _Nonnull task) {
    _textField3.text = [self handleApiResponse:task];
    return nil;
}];
```

Tampilan yang dihasilkan adalah`1 div 2 = 0.5`. Di sini,`div`digunakan sebagai pengganti`/`karenaFungsi Lambda sederhanadi backend tidak menangani URL dikodekan variabel path.

Gunakan iOS SDK (Swift) yang dihasilkan untuk memanggil API

Sebelum memulai prosedur berikut, Anda harus menyelesaikan langkah-langkahMenghasilkan SDK untuk API menggunakan konsol API Gateway untuk iOS di Swift dan mengunduh file.zip dari SDK yang dihasilkan.

Topik

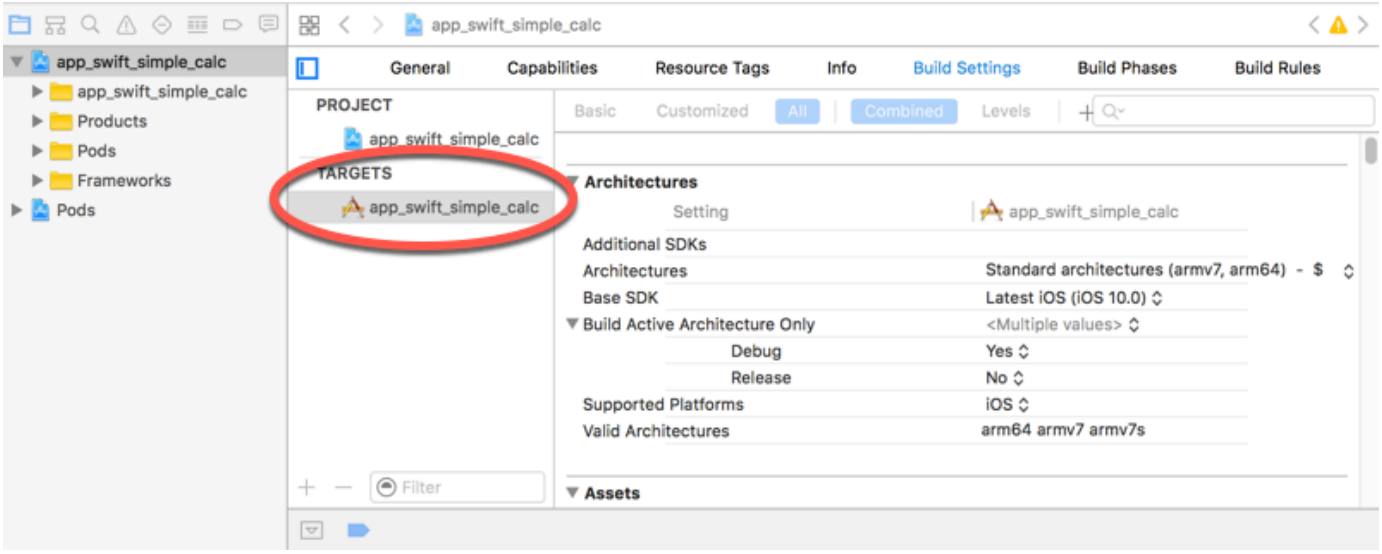
- Instal SDKAWS seluler dan SDK yang dibuat API Gateway dalam proyek Swift
- Memanggil metode API melalui SDK iOS yang dihasilkan oleh API Gateway dalam proyek Swift

Instal SDKAWS seluler dan SDK yang dibuat API Gateway dalam proyek Swift

Prosedur berikut menjelaskan cara menginstal SDK.

Untuk menginstal dan menggunakan SDK iOS yang dihasilkan oleh API Gateway di Swift

1. Ekstrak isi file.zip yang dibuat API Gateway yang Anda unduh sebelumnya. Menggunakan SimpleCalc API, Anda mungkin ingin mengubah nama folder SDK unzip untuk sesuatu seperti**`sdk_swift_simple_calc`**. Dalam folder SDK ini ada`README.md` file dan`Podfile` file. `README.md`File berisi petunjuk untuk menginstal dan menggunakan SDK. Tutorial ini memberikan rincian tentang petunjuk ini. Instalasi memanfaatkan CocoaPodsuntuk mengimpor komponen SDKAWS Seluler yang diperlukan. Anda harus memperbaruui`Podfile` untuk mengimpor SDK ke proyek Xcode aplikasi Swift Anda. Folder SDK yang tidak diarsipkan juga berisi`generated-src` folder yang berisi kode sumber SDK API yang dihasilkan.
2. Buka Xcode dan buat proyek Swift ft baru. Perhatikan target proyek. Anda akan perlu untuk mengaturnya di`Podfile`.



3. Untuk mengimpor komponenAWS Mobile SDK yang diperlukan ke dalam proyek Xcode dengan menggunakan CocoaPods, lakukan hal berikut:

a. Jika tidak diinstal, instal CocoaPods dengan menjalankan perintah berikut di jendela terminal:

```
sudo gem install cocoapods
pod setup
```

b. Salin`Podfile` file dari folder SDK yang diekstrak ke direktori yang sama yang berisi file proyek Xcode Anda. Ganti blok berikut:

```
target '<YourXcodeTarget>' do
    pod 'AWSAPIGateway', '~> 2.4.7'
end
```

dengan nama target proyek Anda seperti yang ditunjukkan:

```
target 'app_swift_simple_calc' do
    pod 'AWSAPIGateway', '~> 2.4.7'
end
```

Jika proyek Xcode Anda sudah berisi target yang benar, Anda cukup menambahkan baris kode berikut ke`do ... end` loop:`Podfile`

```
pod 'AWSAPIGateway', '~> 2.4.7'
```

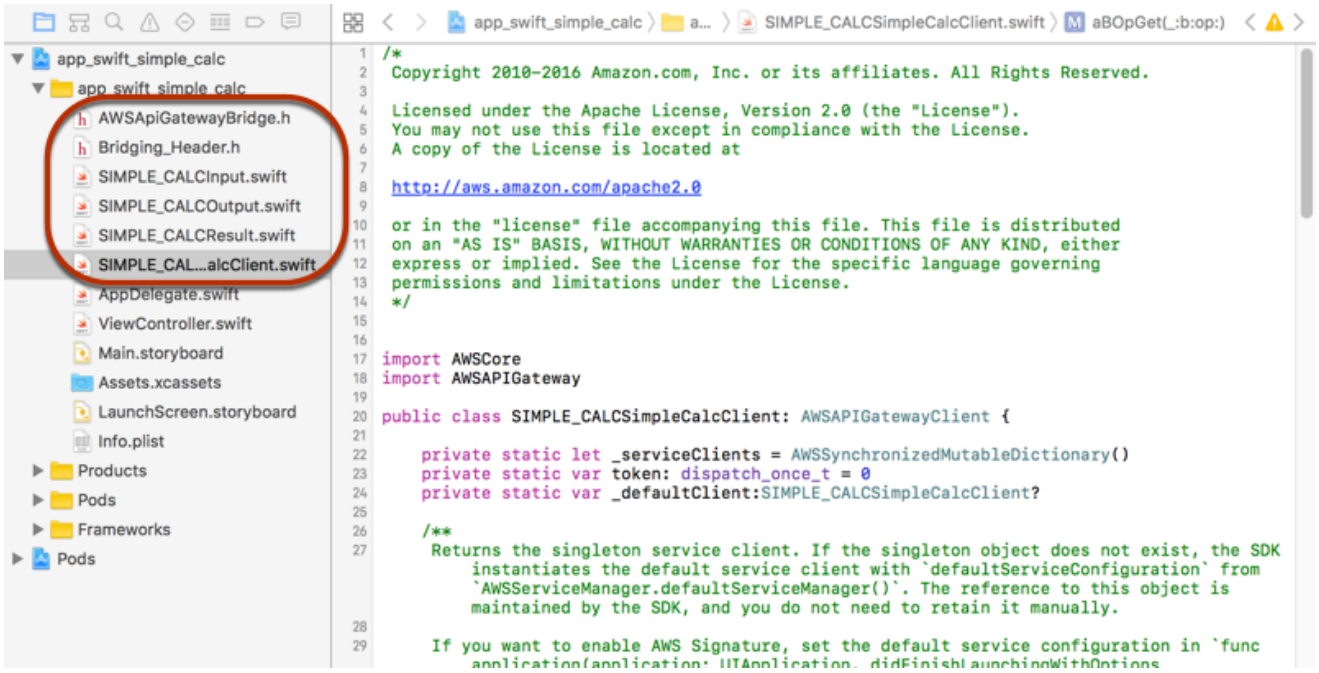
c. Buka jendela terminal dan jalankan perintah berikut di direktori app:

```
pod install
```

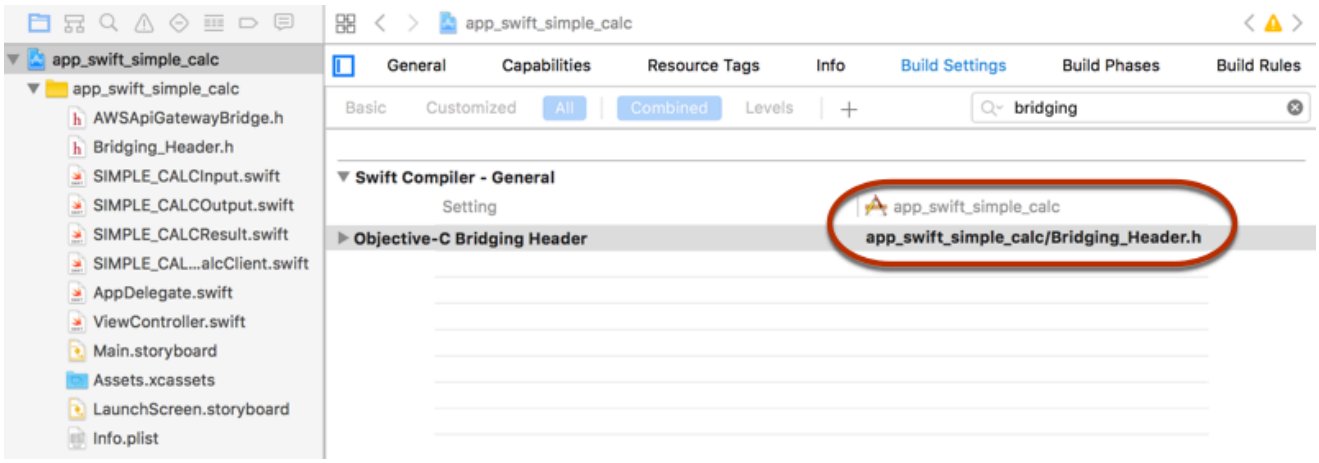
Ini menginstal komponen API Gateway dan komponenAWS Mobile SDK yang bergantung ke dalam proyek aplikasi.

d. Tutup proyek Xcode dan kemudian buka`*.xcworkspace` file untuk meluncurkan kembali Xcode.

e. Tambahkan semua file header SDK (`.h`) dan file kode sumber Swift (`.swift`) dari`generated-src` direktori yang diekstrak ke proyek Xcode Anda.



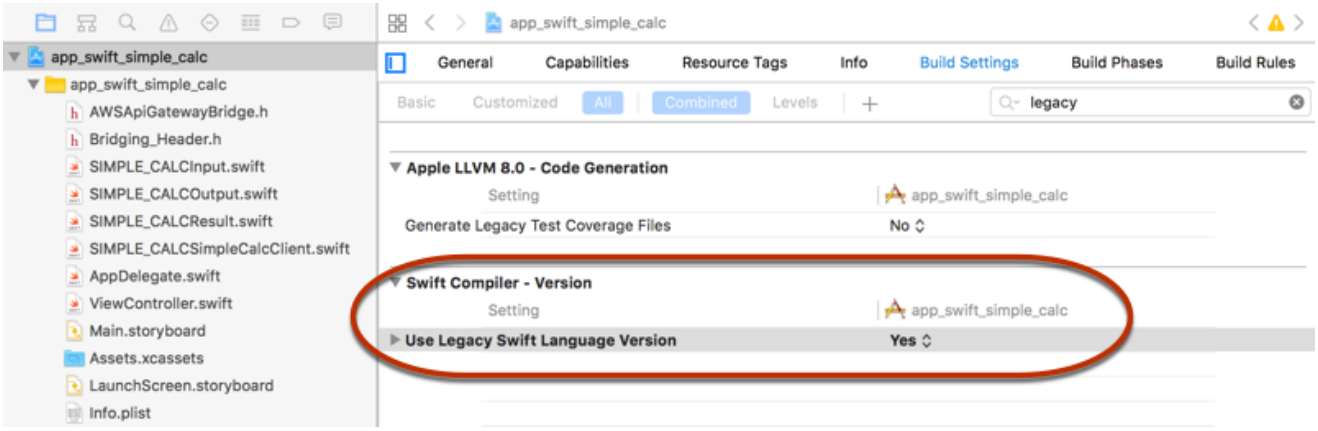
f. Untuk mengaktifkan panggilan pustaka Objective-C dariAWS Mobile SDK dari proyek kode Swift Anda, atur jalur`Bridging_Header.h` file pada properti Objective-C Bridging Header di bawah Swift Compiler - Pengaturan umum konfigurasi proyek Xcode Anda:



> **Tip**
>
> Anda dapat mengetik**`bridging`** di kotak pencarian Xcode untuk menemukan properti Objective-C Bridging Header.

g. Bangun proyek Xcode untuk memverifikasi bahwa proyek tersebut telah dikonfigurasi dengan benar sebelum melanjutkan lebih jauh. Jika Xcode Anda menggunakan versi Swift yang lebih baru daripada versi yang didukung untukAWS Mobile SDK, Anda akan

mendapatkan error compiler Swift. Dalam kasus ini, atur properti Use Legacy Swift Language Version ke Yes di bawah pengaturan Swift Compiler - Version:



Untuk mengimporAWS Mobile SDK for iOS di Swift ke proyek Anda dengan mengunduhAWS Mobile SDK atau menggunakan Carthage secara eksplisit, ikuti petunjuk dalam`README.md` file yang disertakan dengan paket SDK. Pastikan untuk hanya menggunakan salah satu opsi ini untuk mengimpor SDKAWS Seluler.

Memanggil metode API melalui SDK iOS yang dihasilkan oleh API Gateway dalam proyek Swift

Ketika Anda membuat SDK dengan awalan`SIMPLE_CALC` untuk SimpleCalc API ini dengan dua model untuk menggambarkan input (`Input`) dan output (`Result`) dari permintaan dan respons API, dalam SDK, kelas klien API yang dihasilkan menjadi`SIMPLE_CALCSimpleCalcClient` dan kelas data yang sesuai masing-masing.`SIMPLE_CALCInput``SIMPLE_CALCResult` Permintaan dan tanggapan API dipetakan ke metode SDK sebagai berikut:

- Kuota ota ling ling

```
GET /?a=...&b=...&op=...
```

menjadi metode SDK

```
public func rootGet(op: String?, a: String?, b: String?) -> AWSTask
```

`AWSTask.result`Properti ini dari`SIMPLE_CALCResult` jenis jika`Result` model ditambahkan ke respon metode. Jika tidak, itu adalah`NSDictionary` jenis.

- Kuota ota ling ling

```
POST /

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

menjadi metode SDK

```
public func rootPost(body: SIMPLE_CALCInput) -> AWSTask
```

- Kuota ota ling ling

```
GET /{a}/{b}/{op}
```

menjadi metode SDK

```
public func aBOpGet(a: String, b: String, op: String) -> AWSTask
```

Prosedur berikut menjelaskan cara memanggil metode API di kode sumber aplikasi Swift; misalnya, sebagai bagian dari`viewDidLoad()` delegasi dalam`ViewController.m` file.

Memanggil API melalui SDK iOS yang dihasilkan oleh API Gateway

1. Instantiate kelas klien API:

   ```
   let client = SIMPLE_CALCSimpleCalcClient.default()
   ```

   Untuk menggunakan Amazon Cognito dengan API, tetapkan konfigurasiAWS layanan default (ditampilkan berikut) sebelum mendapatkan`default` metode (ditampilkan sebelumnya):

   ```
   let credentialsProvider =
    AWSCognitoCredentialsProvider(regionType: AWSRegionType.USEast1, identityPoolId:
    "my_pool_id")
   let configuration = AWSServiceConfiguration(region: AWSRegionType.USEast1,
    credentialsProvider: credentialsProvider)
   ```

```
AWSServiceManager.defaultServiceManager().defaultServiceConfiguration =
 configuration
```

2. PanggilGET /?a=1&b=2&op=+ metode untuk melakukan1+2:

```
client.rootGet("+", a: "1", b:"2").continueWithBlock {(task: AWSTask) -> AnyObject?
 in
    self.showResult(task)
    return nil
}
```

dimana fungsi pembantuself.showResult(task) mencetak hasil atau kesalahan ke konsol; misalnya:

```
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)")
        }
    }
}
```

Di aplikasi produksi, Anda dapat menampilkan hasil atau kesalahan di bidang teks. Tampilan yang dihasilkan adalah1 + 2 = 3.

3. PanggilPOST / dengan muatan untuk melakukan1-2:

```
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
}
```

Tampilan yang dihasilkan adalah`1 - 2 = -1`.

4. Panggil`GET /{a}/{b}/{op}` untuk melakukan`1/2`:

```
client.aBOpGet("1", b:"2", op:"div").continueWithBlock {(task: AWSTask) ->
 AnyObject? in
    self.showResult(task)
    return nil
}
```

Tampilan yang dihasilkan adalah`1 div 2 = 0.5`. Di sini,`div` digunakan sebagai pengganti`/` karena fungsi Lambda sederhana di backend tidak menangani variabel jalur yang dikodekan URL.

Cara memanggil API pribadi

API pribadi hanya dapat diakses dari dalam VPC Anda, dan kebijakan sumber daya harus mengizinkan akses dari VPC dan titik akhir VPC yang telah Anda konfigurasi. Cara Anda mengakses API pribadi Anda akan tergantung pada apakah Anda telah mengaktifkan DNS pribadi di titik akhir VPC atau tidak. Misalnya, saat mengakses API pribadi dari jaringan lokal melalui Direct AWS Connect, DNS pribadi Anda akan diaktifkan di titik akhir VPC. Dalam kasus seperti itu, ikuti langkah-langkah yang diuraikan dalam Memanggil API Pribadi Anda Menggunakan Nama Host DNS Publik Khusus Titik Akhir.

Setelah Anda menerapkan API pribadi, Anda dapat mengaksesnya melalui DNS pribadi (jika Anda telah mengaktifkan penamaan DNS pribadi) dan melalui DNS publik.

Untuk mendapatkan nama DNS untuk API pribadi Anda, lakukan hal berikut:

1. Masuk ke AWS Management Console dan buka konsol VPC Amazon di https://console.aws.amazon.com/vpc/.
2. Di panel navigasi kiri, pilih Endpoints lalu pilih endpoint VPC antarmuka Anda untuk API Gateway.
3. Di panel Detail, Anda akan melihat 5 nilai di bidang nama DNS. 3 yang pertama adalah nama DNS publik untuk API Anda. Dua lainnya adalah nama DNS pribadi untuk itu.

Memanggil API pribadi Anda menggunakan nama DNS pribadi

> **Warning**
>
> Saat Anda memilih opsi Aktifkan Nama DNS Pribadi saat membuat titik akhir VPC antarmuka untuk API Gateway, VPC tempat titik akhir VPC hadir tidak akan dapat mengakses API publik (dioptimalkan tepi dan regional). Untuk informasi selengkapnya, lihat Mengapa saya tidak dapat terhubung ke API publik saya dari titik akhir VPC API Gateway? .

Jika Anda telah mengaktifkan DNS pribadi, Anda dapat mengakses API pribadi menggunakan nama DNS pribadi sebagai berikut:

```
{restapi-id}.execute-api.{region}.amazonaws.com
```

URL dasar untuk menjalankan API adalah dalam format berikut:

```
https://{restapi-id}.execute-api.{region}.amazonaws.com/{stage}
```

Misalnya, dengan asumsi Anda menyiapkan `GET /pets/{petId}` metode `GET /pets` dan dalam contoh ini, dan dengan asumsi bahwa ID API rest Anda adalah `01234567ab` dan wilayah Anda`us-west-2`, Anda dapat menguji API Anda dengan mengetikkan URL berikut di browser:

```
https://01234567ab.execute-api.us-west-2.amazonaws.com/test/pets
```

and

```
https://01234567ab.execute-api.us-west-2.amazonaws.com/test/pets/1
```

Atau, Anda dapat menggunakan perintah cURL berikut:

```
curl -X GET https://01234567ab.execute-api.us-west-2.amazonaws.com/test/pets
```

and

```
curl -X GET https://01234567ab.execute-api.us-west-2.amazonaws.com/test/pets/2
```

Mengakses API pribadi Anda menggunakan AWS Direct Connect

Anda juga dapat menggunakan AWS Direct Connect untuk membuat koneksi pribadi khusus dari jaringan lokal ke Amazon VPC dan mengakses titik akhir API pribadi Anda melalui koneksi tersebut dengan menggunakan nama DNS publik.

Anda juga dapat menggunakan nama DNS pribadi untuk mengakses API pribadi Anda dari jaringan lokal dengan menyiapkan titik akhir Amazon Route 53 Resolver masuk dan meneruskan semua kueri DNS DNS pribadi dari jaringan jarak jauh Anda. Untuk informasi selengkapnya, lihat Meneruskan kueri DNS masuk ke VPC Anda di Panduan Pengembang Amazon Route 53.

Mengakses API pribadi Anda menggunakan alias Route53

Anda dapat mengaitkan atau memisahkan titik akhir VPC dengan API pribadi Anda dengan menggunakan prosedur yang diuraikan dalam Associate atau Disassociate VPC Endpoint dengan Private REST API.

Setelah Anda mengaitkan ID API REST API pribadi Anda dengan titik akhir VPC, Anda akan memanggil REST API, Anda dapat menggunakan URL dasar format berikut untuk menjalankan API menggunakan alias Route53.

URL dasar yang dihasilkan dalam format berikut:

```
https://{rest-api-id}-{vpce-id}.execute-api.{region}.amazonaws.com/{stage}
```

Misalnya, dengan asumsi Anda menyiapkan metode GET /pets dan GET /pets/ *{PeTiD*} dalam contoh ini, dan dengan asumsi bahwa ID API Anda adalah 01234567ab, ID Titik Akhir VPC adalah vpce-01234567abcdef012, dan wilayah Anda adalah us-barat-2, Anda dapat memanggil API Anda sebagai:

```
curl -v https://01234567ab-vpce-01234567abcdef012.execute-api.us-west-2.amazonaws.com/
test/pets/
```

Memanggil API pribadi Anda menggunakan nama host DNS publik khusus titik akhir

Anda dapat mengakses API pribadi menggunakan nama host DNS khusus titik akhir. Ini adalah nama host DNS publik yang berisi ID titik akhir VPC atau ID API untuk API pribadi Anda.

URL dasar yang dihasilkan dalam format berikut:

```
https://{public-dns-hostname}.execute-api.{region}.vpce.amazonaws.com/{stage}
```

Misalnya, dengan asumsi Anda menyiapkan `GET /pets/{petId}` metode `GET /pets` dan dalam contoh ini, dan dengan asumsi bahwa ID API Anda adalah`01234567ab`, nama host DNS publiknya adalah`vpce-01234567abcdef012-01234567`, dan wilayah Anda, Anda dapat menguji API Anda melalui ID VPCE-nya dengan menggunakan header `Host` dalam perintah cURL, seperti pada contoh berikut: `us-west-2`

```
curl -v https://vpce-01234567abcdef012-01234567.execute-api.us-
west-2.vpce.amazonaws.com/test/pets -H 'Host: 01234567ab.execute-api.us-
west-2.amazonaws.com'
```

Atau, Anda dapat mengakses API pribadi Anda melalui ID API-nya dengan menggunakan `x-apigw-api-id` header dalam perintah cURL dalam format berikut:

```
curl -v https://{public-dns-hostname}.execute-api.{region}.vpce.amazonaws.com/test -
H'x-apigw-api-id:{api-id}'
```

Mengkonfigurasi REST API menggunakan OpenAPI

Anda dapat menggunakan API Gateway untuk mengimpor REST API dari file definisi eksternal ke API Gateway. Saat ini, API Gateway mendukung file definisi OpenAPIOpenAPIv2.0 dan v3.0, dengan pengecualian yang tercantum dalamCatatan penting Amazon API Gateway untuk REST API. Anda dapat memperbarui API dengan menimpanya dengan definisi baru, atau Anda dapat menggabungkan definisi dengan API yang ada. Anda menentukan opsi dengan menggunakan parametermode kueri di URL permintaan.

Untuk tutorial tentang penggunaan fitur Import API dari konsol API Gateway, lihatTutorial: Buat REST API dengan mengimpor contoh.

Topik

- Impor API yang dioptimalkan tepi ke API Gateway
- Impor API regional ke API Gateway
- Mengimpor file OpenAPI untuk memperbarui definisi API yang ada
- MengaturbasePath properti OpenAPI
- AWSvariabel untuk impor OpenAPI
- Kesalahan dan peringatan selama impor
- Ekspor REST API dari API Gateway

Impor API yang dioptimalkan tepi ke API Gateway

Anda dapat mengimpor file definisi OpenAPI API untuk membuat API baru yang dioptimalkan tepi dengan menentukan tipe `EDGE` titik akhir sebagai input tambahan, selain file OpenAPI, ke operasi impor. Anda dapat melakukannya menggunakan konsol API GatewayAWS CLI, atau AWS SDK.

Untuk tutorial tentang penggunaan fitur Impor API dari konsol API Gateway, lihatTutorial: Buat REST API dengan mengimpor contoh.

Topik

- Impor API yang dioptimalkan tepi menggunakan konsol API Gateway
- Impor API yang dioptimalkan tepi menggunakan AWS CLI

Impor API yang dioptimalkan tepi menggunakan konsol API Gateway

Untuk mengimpor API yang dioptimalkan tepi menggunakan konsol API Gateway, lakukan hal berikut:

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih Buat API.
3. Di bawah REST API, pilih Impor.
4. Salin definisi OpenAPI API dan tempelkan ke editor kode, atau pilih Pilih file untuk memuat file OpenAPI dari drive lokal.
5. Untuk jenis endpoint API, pilih Edge-optimized.
6. Pilih Buat API untuk mulai mengimpor definisi OpenAPI.

Impor API yang dioptimalkan tepi menggunakan AWS CLI

Untuk mengimpor API dari file definisi OpenAPI untuk membuat API baru yang dioptimalkan tepi menggunakanAWS CLI, gunakan perintah sebagai berikut`import-rest-api`:

```
aws apigateway import-rest-api \
    --fail-on-warnings \
    --body 'file://path/to/API_OpenAPI_template.json'
```

atau dengan spesifikasi eksplisit dari parameter string `endpointConfigurationTypes` kueri untuk`EDGE`:

```
aws apigateway import-rest-api \
    --parameters endpointConfigurationTypes=EDGE \
    --fail-on-warnings \
    --body 'file://path/to/API_OpenAPI_template.json'
```

Impor API regional ke API Gateway

Saat mengimpor API, Anda dapat memilih konfigurasi titik akhir regional untuk API. Anda dapat menggunakan konsol API Gateway, theAWS CLI, atau AWS SDK.

Saat Anda mengekspor API, konfigurasi titik akhir API tidak disertakan dalam definisi API yang diekspor.

Untuk tutorial tentang penggunaan fitur Impor API dari konsol API Gateway, lihatTutorial: Buat REST API dengan mengimpor contoh.

Topik

- Mengimpor API regional menggunakan konsol API Gateway
- Impor API regional menggunakan AWS CLI

Mengimpor API regional menggunakan konsol API Gateway

Untuk mengimpor API titik akhir regional menggunakan konsol API Gateway, lakukan hal berikut:

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih Buat API.
3. Di bawah REST API, pilih Impor.
4. Salin definisi OpenAPI API dan tempelkan ke editor kode, atau pilih Pilih file untuk memuat file OpenAPI dari drive lokal.
5. Untuk jenis titik akhir API, pilih Regional.
6. Pilih Buat API untuk mulai mengimpor definisi OpenAPI.

Impor API regional menggunakan AWS CLI

Untuk mengimpor API dari file definisi OpenAPI menggunakanAWS CLI, gunakan perintah: `import-rest-api`

```
aws apigateway import-rest-api \
    --parameters endpointConfigurationTypes=REGIONAL \
    --fail-on-warnings \
    --body 'file://path/to/API_OpenAPI_template.json'
```

Mengimpor file OpenAPI untuk memperbarui definisi API yang ada

Anda dapat mengimpor definisi API hanya untuk memperbarui API yang ada, tanpa mengubah konfigurasi endpoint, serta variabel stage dan stage, atau referensi ke kunci API.

import-to-update Operasi dapat terjadi dalam dua mode: menggabungkan atau menimpa.

Ketika API (`A`) digabungkan ke (`B`) lainnya, API yang dihasilkan mempertahankan definisi keduanya`A` dan`B` jika kedua API tidak berbagi definisi yang saling bertentangan. Ketika konflik muncul, definisi metode API penggabungan (`A`) menimpa definisi metode yang sesuai dari API gabungan (`B`). Misalnya, misalkan`B` telah menyatakan metode berikut untuk kembali`200` dan`206` tanggapan:

```
GET /a
POST /a
```

dan`A` menyatakan metode berikut untuk kembali`200` dan`400` tanggapan:

```
GET /a
```

Ketika`A` digabungkan menjadi`B`, API yang dihasilkan menghasilkan metode berikut:

```
GET /a
```

yang kembali`200` dan`400` tanggapan, dan

```
POST /a
```

yang kembali`200` dan`206` tanggapan.

Penggabungan API berguna ketika Anda telah menguraikan definisi API eksternal Anda menjadi beberapa bagian yang lebih kecil dan hanya ingin menerapkan perubahan dari salah satu bagian tersebut sekaligus. Misalnya, ini mungkin terjadi jika beberapa tim bertanggung jawab atas berbagai bagian API dan memiliki perubahan yang tersedia pada tarif yang berbeda. Dalam mode ini, item dari API yang ada yang tidak ditentukan secara khusus dalam definisi yang diimpor dibiarkan sendiri.

Ketika API (A) menimpa API lain (B), API yang dihasilkan mengambil definisi API Timpa (A). Timpa API berguna ketika definisi API eksternal berisi definisi lengkap API. Dalam mode ini, item dari API yang ada yang tidak ditentukan secara khusus dalam definisi yang diimpor akan dihapus.

Untuk menggabungkan API, kirimkanPUT permintaan kehttps://apigateway.*<region>*.amazonaws.com/restapis/*<restapi_id>*?mode=merge. Nilai parameterrestapi_id path menentukan API tempat definisi API yang disediakan akan digabungkan.

Cuplikan kode berikut menunjukkan contohPUT permintaan untuk menggabungkan definisiOpenAPI API di JSON, sebagai payload, dengan API yang ditentukan sudah ada di 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
```

Operasi pembaruan penggabungan mengambil dua definisi API lengkap dan menggabungkannya bersama-sama. Untuk perubahan kecil dan bertahap, Anda dapat menggunakan operasi pembaruan sumber daya.

Untuk menimpa API, kirimkanPUT permintaan kehttps://apigateway.*<region>*.amazonaws.com/restapis/*<restapi_id>*?mode=overwrite. Parameterrestapi_id path menentukan API yang akan ditimpa dengan definisi API yang disediakan.

Cuplikan kode berikut menunjukkan contoh permintaan timpa dengan payloadOpenAPI definisi berformat JSON:

```
PUT /restapis/<restapi_id>?mode=overwrite
Host:apigateway.<region>.amazonaws.com
Content-Type: application/json
Content-Length: ...

An OpenAPI API definition in JSON
```

Bila parametermode kueri tidak ditentukan, penggabungan diasumsikan.

> **Note**
>
> PUTOperasi idempoten, tapi tidak atom. Itu berarti jika kesalahan sistem terjadi sebagian cara melalui pemrosesan, API dapat berakhir dalam keadaan buruk. Namun, mengulangi operasi berhasil menempatkan API ke keadaan akhir yang sama seolah-olah operasi pertama telah berhasil.

Mengatur**basePath** properti OpenAPI

Di OpenAPI 2.0, Anda dapat menggunakan`basePath` properti untuk menyediakan satu atau lebih bagian jalur yang mendahului setiap jalur yang didefinisikan dalam`paths` properti. Karena API Gateway memiliki beberapa cara untuk mengekspresikan jalur sumber daya, fitur Import API menyediakan opsi berikut untuk menafsirkan`basePath` properti selama impor: ignore, prepend, dan split.

Di OpenAPI 3.0,`basePath` tidak lagi properti tingkat atas. Sebaliknya, API Gateway menggunakan variabel server sebagai konvensi. Fitur Import API menyediakan opsi yang sama untuk menafsirkan jalur dasar selama impor. Jalur dasar diidentifikasi sebagai berikut:

- Jika API tidak mengandung`basePath` variabel apa pun, fitur Import API memeriksa`server.url` string untuk melihat apakah itu berisi jalur di luar`"/"`. Jika ya, jalan yang digunakan sebagai jalur dasar.
- Jika API hanya berisi satu`basePath` variabel, fitur Import API menggunakannya sebagai jalur dasar, bahkan jika itu tidak direferensikan di`server.url`.
- Jika API berisi beberapa`basePath` variabel, fitur Import API hanya menggunakan yang pertama sebagai jalur dasar.

Abaikan

Jika file OpenAPI memiliki`basePath` nilai`/a/b/c` dan`paths` properti berisi`/e` dan`/f`, berikut`POST` atau`PUT` permintaan:

```
POST /restapis?mode=import&basepath=ignore
```

```
PUT /restapis/api_id?basepath=ignore
```

menghasilkan berikut ini:

- `/`
- `/e`
- `/f`

Efeknya adalah untuk memperlakukan`basePath` seolah-olah itu tidak ada, dan semua sumber daya API dideklarasikan disajikan relatif terhadap host. Ini dapat digunakan, misalnya, jika Anda memiliki nama domain khusus dengan pemetaan API yang tidak menyertakan Jalur Dasar dan nilai Stage yang mengacu pada tahap produksi Anda.

> **Note**
>
> API Gateway secara otomatis membuat sumber daya root untuk Anda, meskipun tidak dideklarasikan secara eksplisit dalam file definisi Anda.

Ketika tidak ditentukan,`basePath` mengambil secara`ignore` default.

Tambahkan

JikaOpenAPI file memiliki`basePath` nilai`/a/b/c` dan`paths` properti berisi`/e` dan`/f`, berikut`POST` atau`PUT` permintaan:

```
POST /restapis?mode=import&basepath=prepend
```

```
PUT /restapis/api_id?basepath=prepend
```

menghasilkan berikut ini:

- `/`
- `/a`
- `/a/b`
- `/a/b/c`

- `/a/b/c/e`
- `/a/b/c/f`

Efeknya adalah untuk memperlakukan`basePath` sebagai menentukan sumber daya tambahan (tanpa metode) dan menambahkannya ke set sumber daya dinyatakan. Ini dapat digunakan, misalnya, ketika tim yang berbeda bertanggung jawab atas berbagai bagian API dan`basePath` dapat merujuk lokasi jalur untuk setiap bagian API tim.

> **Note**
>
> API Gateway secara otomatis membuat sumber daya perantara untuk Anda, meskipun tidak dideklarasikan secara eksplisit dalam definisi Anda.

Split

JikaOpenAPI file memiliki`basePath` nilai`/a/b/c` dan`paths` properti berisi`/e` dan`/f`, berikut`POST` atau`PUT` permintaan:

```
POST /restapis?mode=import&basepath=split
```

```
PUT /restapis/api_id?basepath=split
```

menghasilkan berikut ini:

- `/`
- `/b`
- `/b/c`
- `/b/c/e`
- `/b/c/f`

Efeknya adalah memperlakukan bagian jalur paling atas`/a`,, sebagai awal dari setiap jalur sumber daya, dan untuk membuat sumber daya tambahan (tanpa metode) dalam API itu sendiri. Ini bisa, misalnya, digunakan ketika`a` adalah nama tahap yang ingin Anda mengekspos sebagai bagian dari API Anda.

AWSvariabel untuk impor OpenAPI

Anda dapat menggunakanAWS variabel berikut dalam definisi OpenAPI. API Gateway menyelesaikan variabel saat API diimpor. Untuk menentukan variabel, gunakan${*variable-name*}.

AWSvariabel

Nama variabel	Deskripsi
`AWS::AccountId`	IDAWS akun yang mengimpor API — misalnya, 123456789012.
`AWS::Partition`	AWSPartisi tempat API diimpor. Untuk Wilayah AWS standar, partisinya adalah `aws`.
`AWS::Region`	AWSWilayah di mana API diimpor — misalnya,`us-east-2` .

AWSvariabel contoh

Contoh berikut menggunakanAWS variabel untuk menentukanAWS Lambda fungsi untuk integrasi.

OpenAPI 3.0

```
openapi: "3.0.1"
info:
  title: "tasks-api"
  version: "v1.0"
paths:
  /:
    get:
      summary: List tasks
      description: Returns a list of tasks
      responses:
        200:
```

```
          description: "OK"
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: "#/components/schemas/Task"
        500:
          description: "Internal Server Error"
          content: {}
      x-amazon-apigateway-integration:
        uri:
          arn:${AWS::Partition}:apigateway:${AWS::Region}:lambda:path/2015-03-31/
functions/arn:${AWS::Partition}:lambda:${AWS::Region}:
${AWS::AccountId}:function:LambdaFunctionName/invocations
        responses:
          default:
            statusCode: "200"
        passthroughBehavior: "when_no_match"
        httpMethod: "POST"
        contentHandling: "CONVERT_TO_TEXT"
        type: "aws_proxy"
components:
  schemas:
    Task:
      type: object
      properties:
        id:
          type: integer
        name:
          type: string
        description:
          type: string
```

Kesalahan dan peringatan selama impor

Kesalahan selama impor

Selama impor, kesalahan dapat dihasilkan untuk masalah besar sepertiOpenAPI dokumen yang tidak valid. Kesalahan dikembalikan sebagai pengecualian (misalnya,`BadRequestException`) dalam respons yang tidak berhasil. Saat terjadi kesalahan, definisi API baru akan dibuang dan tidak ada perubahan yang dilakukan pada API yang ada.

Peringatan selama impor

Selama impor, peringatan dapat dihasilkan untuk masalah kecil seperti referensi model yang hilang. Jika peringatan terjadi, operasi akan berlanjut jika ekspresi`failonwarnings=false` kueri ditambahkan ke URL permintaan. Jika tidak, pembaruan akan digulung kembali. Secara default, `failonwarnings` diatur ke `false`. Dalam kasus seperti itu, peringatan dikembalikan sebagai bidang dalam RestApisumber daya yang dihasilkan. Jika tidak, peringatan dikembalikan sebagai pesan dalam pengecualian.

Ekspor REST API dari API Gateway

Setelah Anda membuat dan mengonfigurasi REST API di API Gateway, menggunakan konsol API Gateway atau sebaliknya, Anda dapat mengekspornya ke file OpenAPI menggunakan API Gateway Export API, yang merupakan bagian dari Amazon API Gateway Control Service. Untuk menggunakan API Gateway Export API, Anda harus menandatangani permintaan API Anda. Untuk informasi selengkapnya tentang permintaan penandatanganan, lihat Menandatangani permintaan AWS API di Panduan Pengguna IAM. Anda memiliki opsi untuk menyertakan ekstensi integrasi API Gateway, serta ekstensi Postman, dalam file definisi OpenAPI yang diekspor.

> **Note**
>
> Saat mengekspor API menggunakanAWS CLI, pastikan untuk menyertakan parameter ekstensi seperti yang ditunjukkan pada contoh berikut, untuk memastikan bahwa `x-amazon-apigateway-request-validator` ekstensi disertakan:
>
> ```
> aws apigateway get-export --parameters extensions='apigateway' --rest-api-id
>  abcdefg123 --stage-name dev --export-type swagger latestswagger2.json
> ```

Anda tidak dapat mengekspor API jika payloadnya bukan dari `application/json` jenisnya. Jika Anda mencoba, Anda akan mendapatkan respons kesalahan yang menyatakan bahwa model tubuh JSON tidak ditemukan.

Permintaan untuk mengekspor REST API

Dengan Export API, Anda mengekspor REST API yang ada dengan mengirimkan permintaan GET, menetapkan to-be-exported API sebagai bagian dari jalur URL. URL permintaan adalah dari format berikut:

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
```

Anda dapat menambahkan string `extensions` kueri untuk menentukan apakah akan menyertakan ekstensi API Gateway (dengan `integration` nilai) atau ekstensi Postman (dengan `postman` nilai).

Selain itu, Anda dapat mengatur `Accept` header ke `application/json` atau `application/yaml` untuk menerima output definisi API masing-masing dalam format JSON atau YAMAL.

Untuk informasi selengkapnya tentang mengirimkan permintaan GET menggunakan API Gateway Export API, lihat. GetExport

> **Note**
>
> Jika Anda mendefinisikan model di API Anda, model tersebut harus untuk jenis konten “application/json” agar API Gateway dapat mengekspor model. Jika tidak, API Gateway melempar pengecualian dengan pesan kesalahan “Hanya menemukan model tubuh non-JSON untuk...”.
> Model harus berisi properti atau didefinisikan sebagai jenis JsonSchema tertentu.

Unduh definisi REST API OpenAPI dalam JSON

Untuk mengekspor dan mengunduh REST API dalam definisi OpenAPI dalam format 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
```

Di sini, *<region>* bisa jadi, misalnya,`us-east-1`. Untuk semua wilayah di mana API Gateway tersedia, lihat Wilayah dan Titik Akhir

Unduh definisi REST API OpenAPI dalam YAMAL

Untuk mengekspor dan mengunduh REST API dalam definisi OpenAPI dalam format YAMAL:

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
```

Unduh definisi OpenAPI REST API dengan ekstensi Postman di JSON

Untuk mengekspor dan mengunduh REST API dalam definisi OpenAPI dengan format Postman 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
```

Unduh definisi OpenAPI REST API dengan integrasi API Gateway di YAMAL

Untuk mengekspor dan mengunduh REST API dalam definisi OpenAPI dengan integrasi API Gateway dalam format YAMAL:

OpenAPI 3.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/oas30?extensions=integrations
Host: apigateway.<region>.amazonaws.com
Accept: application/yaml
```

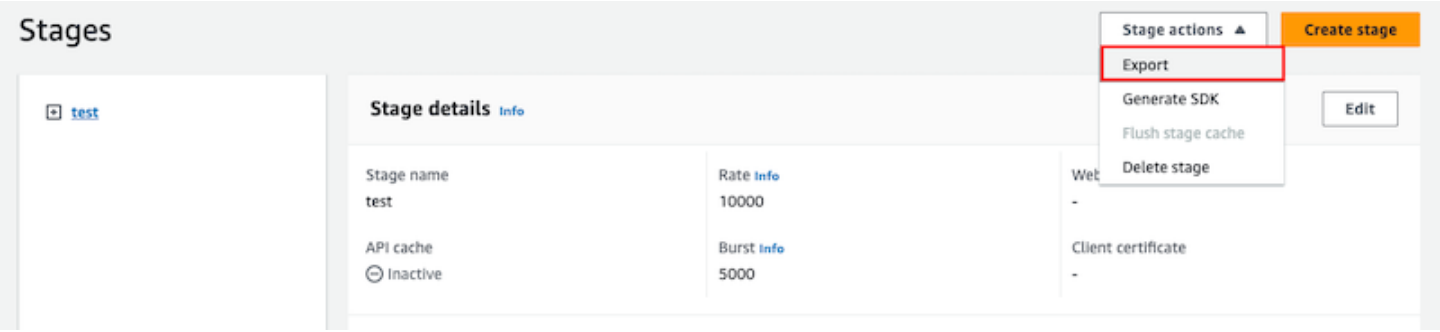
OpenAPI 2.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/swagger?
extensions=integrations
Host: apigateway.<region>.amazonaws.com
Accept: application/yaml
```

Ekspor REST API menggunakan konsol API Gateway

Setelah menerapkan REST API ke tahap, Anda dapat melanjutkan untuk mengekspor API di tahap ke file OpenAPI menggunakan konsol API Gateway.

Di panel Tahapan di konsol API Gateway, pilih Tindakan tahap, Ekspor.



Tentukan jenis spesifikasi API, Format, dan Ekstensi untuk mengunduh definisi OpenAPI API Anda.

Menerbitkan REST API bagi pelanggan untuk dipanggil

Cukup membuat dan mengembangkan API Gateway API tidak secara otomatis membuatnya dapat dipanggil oleh pengguna Anda. Untuk membuatnya dapat dipanggil, Anda harus menerapkan API Anda ke sebuah panggung. Selain itu, Anda mungkin ingin menyesuaikan URL yang akan digunakan pengguna Anda untuk mengakses API Anda. Anda dapat memberikan domain yang konsisten dengan merek Anda atau lebih mudah diingat daripada URL default untuk API Anda.

Di bagian ini, Anda dapat mempelajari cara menerapkan API dan menyesuaikan URL yang Anda berikan kepada pengguna untuk mengaksesnya.

> **Note**
>
> Untuk meningkatkan keamanan API Gateway API Anda, `execute-api.{region}.amazonaws.com` domain tersebut terdaftar di Daftar Akhiran Publik (PSL). Untuk keamanan lebih lanjut, kami menyarankan Anda menggunakan cookie dengan `__Host-`

awalan jika Anda perlu mengatur cookie sensitif di nama domain default untuk API Gateway API Anda. Praktik ini akan membantu mempertahankan domain Anda dari upaya pemalsuan permintaan lintas situs (CSRF). Untuk informasi selengkapnya, lihat halaman Set-Cookie di Jaringan Pengembang Mozilla.

Topik

- Menerapkan REST API di Amazon API Gateway
- Menyiapkan nama domain kustom untuk REST API

Menerapkan REST API di Amazon API Gateway

Setelah membuat API, Anda harus menerapkannya agar dapat dipanggil oleh pengguna Anda.

Untuk menerapkan API, Anda membuat penerapan API dan mengaitkannya dengan stage. Tahap adalah referensi logis ke status siklus hidup API Anda (misalnya,`dev`,`prod`,`beta`,`v2`). Tahapan API diidentifikasi oleh ID API dan nama tahap. Mereka termasuk dalam URL yang Anda gunakan untuk memanggil API. Setiap tahap adalah referensi bernama untuk deployment API dan dibuat tersedia bagi aplikasi klien untuk dipanggil.

Important

Setiap kali Anda memperbarui API, Anda harus men-deploy API ke tahap yang ada atau ke tahap baru. Memperbarui API termasuk memodifikasi rute, metode, integrasi, otorisasi, dan apa pun selain pengaturan panggung.

Saat API Anda berkembang, Anda dapat terus menerapkannya ke berbagai tahap sebagai versi API yang berbeda. Anda juga dapat menerapkan pembaruan API Anda sebagaipenyebaran canary. Ini memungkinkan klien API Anda untuk mengakses, pada tahap yang sama, versi produksi melalui rilis produksi, dan versi terbaru melalui rilis kenari.

Untuk memanggil API yang di-deploy, klien mengirimkan permintaan terhadap URL API. URL ditentukan oleh protokol API (HTTP (S) atau (WSS)), nama host, nama panggung, dan (untuk REST API) jalur sumber daya. Nama host dan nama panggung menentukan URL dasar API.

Menggunakan nama domain default API, URL dasar REST API (misalnya) dalam tahap tertentu (*`{stageName}`*) Dalam format berikut:

```
https://{restapi-id}.execute-api.{region}.amazonaws.com/{stageName}
```

Untuk membuat URL dasar default API lebih ramah pengguna, Anda dapat membuat nama domain kustom (misalnya,`api.example.com`) untuk mengganti nama host default API. Untuk mendukung beberapa API di bawah nama domain kustom, Anda harus memetakan tahap API ke jalur dasar.

Dengan nama domain khusus*{api.example.com}*dan tahap API dipetakan ke jalur dasar (*{basePath}*) di bawah nama domain kustom, URL dasar REST API menjadi berikut:

```
https://{api.example.com}/{basePath}
```

Untuk setiap tahap, Anda dapat mengoptimalkan kinerja API dengan menyesuaikan batas pembatasan permintaan tingkat akun default dan mengaktifkan cache API. Anda juga dapat mengaktifkan logging untuk panggilan API CloudTrail atau CloudWatch, dan dapat memilih sertifikat klien untuk backend untuk mengotentikasi permintaan API. Selain itu, Anda dapat mengganti pengaturan tingkat tahap untuk metode individual dan menentukan variabel tahap untuk meneruskan konteks lingkungan spesifik tahap ke integrasi API saat runtime.

Tahapan memungkinkan kontrol versi yang kuat dari API Anda. Misalnya, Anda dapat menerapkan API ke`test`tahap dan tahap`prod`panggung, dan gunakan`test`tahap sebagai tes membangun dan menggunakan`prod`panggung sebagai membangun stabil. Setelah pembaruan lulus tes, Anda dapat mempromosikan`test`tahap`prod`tahap Promosi dapat dilakukan dengan menyebarkan kembali API ke`prod`tahap atau memperbaruiVariabel Tahapnilai dari nama tahap`test`untuk itu`prod`.

Pada bagian ini, kita membahas bagaimana menerapkan API dengan menggunakanKonsol API Gatewayatau memanggilAPI REST API Gateway. Untuk menggunakan alat lain, lihat dokumentasiAWSCLIatauAWSSDK.

Topik

- Menerapkan REST API di API Gateway
- Menyiapkan panggung untuk REST API
- Siapkan penerapan rilis kenari API Gateway
- Pembaruan ke REST API yang memerlukan penyebaran ulang

Menerapkan REST API di API Gateway

Di API Gateway, penerapan REST API diwakili oleh sumber daya Deployment. Ini mirip dengan executable API yang diwakili oleh sumber daya. RestApi

Agar klien dapat memanggil API Anda, Anda harus membuat penerapan dan mengaitkan tahap dengannya. Sebuah panggung diwakili oleh sumber daya Panggung. Ini mewakili snapshot API, termasuk metode, integrasi, model, template pemetaan, dan otorisasi Lambda (sebelumnya dikenal sebagai otorisasi khusus). Saat memperbarui API, Anda dapat menerapkan ulang API dengan mengaitkan tahap baru dengan penerapan yang ada. Kami membahas membuat panggung dithe section called “Siapkan panggung”.

Topik

- Buat penerapan menggunakan AWS CLI
- Menerapkan REST API dari konsol API Gateway

Buat penerapan menggunakan AWS CLI

Saat membuat penerapan, Anda membuat instance sumber daya Deployment. Anda dapat menggunakan konsol API Gateway, AWS SDK, atau API Gateway REST API untuk membuat penerapan. AWS CLI

Untuk menggunakan CLI untuk membuat penerapan, gunakan perintah create-deployment:

```
aws apigateway create-deployment --rest-api-id <rest-api-id> --region <region>
```

API tidak dapat dipanggil sampai Anda mengaitkan penerapan ini dengan sebuah panggung. Dengan tahap yang ada, Anda dapat melakukan ini dengan memperbarui `deploymentId`properti stage dengan ID deployment (`<deployment-id>`) yang baru dibuat.

```
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>'
```

Saat menerapkan API pertama kali, Anda dapat menggabungkan pembuatan panggung dan pembuatan penerapan secara bersamaan:

```
aws apigateway create-deployment --region <region> \
```

```
    --rest-api-id <rest-api-id> \
    --stage-name <stage-name>
```

Inilah yang dilakukan di balik layar di konsol API Gateway saat Anda menerapkan API pertama kali, atau saat Anda menerapkan ulang API ke tahap baru.

Menerapkan REST API dari konsol API Gateway

Anda harus telah membuat REST API sebelum menerapkannya untuk pertama kalinya. Untuk informasi selengkapnya, lihat Membuat REST API di Amazon API Gateway.

Topik

- Menerapkan REST API ke panggung
- Menerapkan ulang REST API ke panggung
- Perbarui konfigurasi tahap penerapan REST API
- Menetapkan variabel panggung untuk penerapan REST API
- Kaitkan tahap dengan penerapan REST API yang berbeda

Menerapkan REST API ke panggung

Konsol API Gateway memungkinkan Anda men-deploy API dengan membuat deployment dan mengaitkannya dengan tahap baru atau yang sudah ada.

> **Note**
>
> Untuk mengaitkan tahap di API Gateway dengan penerapan yang berbeda, lihat Kaitkan tahap dengan penerapan REST API yang berbeda sebagai gantinya.

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Di panel navigasi API, pilih API yang ingin Anda terapkan.
3. Di panel Resources, pilih Deploy API.
4. Untuk Stage, pilih dari berikut ini:

    a. Untuk membuat tahap baru, pilih Tahap baru, lalu masukkan nama di Nama panggung. Secara opsional, Anda dapat memberikan deskripsi untuk penyebaran dalam deskripsi Deployment.

b. Untuk memilih tahap yang ada, pilih nama panggung dari menu tarik-turun. Anda mungkin ingin memberikan deskripsi penerapan baru dalam deskripsi Deployment.

c. Untuk membuat penyebaran yang tidak terkait dengan tahap, pilih Tidak ada tahap. Nanti, Anda dapat mengaitkan penerapan ini dengan sebuah panggung.

5. Pilih Deploy.

Menerapkan ulang REST API ke panggung

Untuk menerapkan ulang API, lakukan langkah yang sama seperti di. the section called “Menerapkan REST API ke panggung” Anda dapat menggunakan kembali tahap yang sama sebanyak yang diinginkan.

Perbarui konfigurasi tahap penerapan REST API

Setelah API diterapkan, Anda dapat memodifikasi pengaturan panggung untuk mengaktifkan atau menonaktifkan cache API, logging, atau meminta pembatasan. Anda juga dapat memilih sertifikat klien untuk backend untuk mengautentikasi API Gateway dan mengatur variabel tahap untuk meneruskan konteks penerapan ke integrasi API saat runtime. Untuk informasi selengkapnya, lihat Perbarui pengaturan panggung.

> **Important**
>
> Setelah memodifikasi pengaturan tahap, Anda harus menerapkan ulang API agar perubahan diterapkan.

> **Note**
>
> Jika pengaturan yang diperbarui, seperti mengaktifkan logging, memerlukan peran IAM baru, Anda dapat menambahkan peran IAM yang diperlukan tanpa menerapkan ulang API. Namun, perlu beberapa menit sebelum peran IAM baru berlaku. Sebelum itu terjadi, jejak panggilan API Anda tidak dicatat meskipun Anda telah mengaktifkan opsi logging.

Menetapkan variabel panggung untuk penerapan REST API

Untuk penerapan, Anda dapat menyetel atau memodifikasi variabel tahap untuk meneruskan data khusus penerapan ke integrasi API saat runtime. Anda dapat melakukan ini pada tab Stage Variables

di Stage Editor. Untuk informasi lebih lanjut, lihat petunjuk diMenyiapkan variabel tahap untuk penerapan REST API.

Kaitkan tahap dengan penerapan REST API yang berbeda

Karena penerapan merepresentasikan snapshot API dan tahap menentukan jalur ke dalam snapshot, Anda dapat memilih kombinasi tahap penerapan yang berbeda untuk mengontrol cara pengguna memanggil ke versi API yang berbeda. Ini berguna, misalnya, ketika Anda ingin memutar kembali status API ke penerapan sebelumnya atau menggabungkan 'cabang pribadi' API ke publik.

Prosedur berikut menunjukkan cara melakukannya menggunakan Stage Editor di konsol API Gateway. Diasumsikan bahwa Anda harus telah menerapkan API lebih dari sekali.

1. Jika Anda belum berada di panel Tahapan, di panel navigasi utama, pilih Tahapan.
2. Pilih tahap yang ingin Anda perbarui.
3. Pada tab Riwayat Deployment, pilih penyebaran yang ingin Anda gunakan untuk tahap.
4. Pilih Ubah penerapan aktif.
5. Konfirmasikan bahwa Anda ingin mengubah penerapan aktif dan pilih Ubah penerapan aktif di kotak dialog Make active deployment.

Menyiapkan panggung untuk REST API

Tahap adalah referensi bernama untuk penerapan, yang merupakan snapshot dari API. Anda menggunakan Stage untuk mengelola dan mengoptimalkan penerapan tertentu. Misalnya, Anda dapat mengonfigurasi pengaturan tahap untuk mengaktifkan caching, menyesuaikan pembatasan permintaan, mengonfigurasi logging, menentukan variabel tahap, atau melampirkan rilis kenari untuk pengujian.

Topik

- Menyiapkan panggung menggunakan konsol API Gateway
- Menyiapkan tag untuk tahap API di API Gateway
- Menyiapkan variabel tahap untuk penerapan REST API

Menyiapkan panggung menggunakan konsol API Gateway

Topik

- Buat panggung baru

- Perbarui pengaturan panggung
- Ganti pengaturan tingkat panggung
- Hapus panggung

Buat panggung baru

Setelah penerapan awal, Anda dapat menambahkan lebih banyak tahapan dan mengaitkannya dengan penerapan yang ada. Anda dapat menggunakan konsol API Gateway untuk membuat tahap baru, atau Anda dapat memilih tahap yang ada saat menerapkan API. Secara umum, Anda dapat menambahkan tahap baru ke penerapan API sebelum menerapkan ulang API. Untuk membuat tahap baru menggunakan konsol API Gateway, ikuti langkah-langkah berikut:

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih REST API.
3. Di panel navigasi utama, pilih Tahapan di bawah API.
4. Dari panel navigasi Tahapan, pilih Buat tahap.
5. Untuk nama Panggung, masukkan nama, misalnya,**`prod`**.

   > **Note**
   >
   > Nama panggung hanya dapat berisi karakter alfanumerik, tanda hubung, dan garis bawah. Panjang maksimum adalah 128 karakter.

6. (Opsional). Untuk Deskripsi, masukkan deskripsi panggung.
7. Untuk Deployment, pilih tanggal dan waktu penerapan API yang ada yang ingin Anda kaitkan dengan tahap ini.
8. Di bawah Pengaturan tambahan, Anda dapat menentukan pengaturan tambahan untuk panggung Anda.
9. Pilih Buat panggung.

Perbarui pengaturan panggung

Setelah penerapan API berhasil, tahap diisi dengan pengaturan default. Anda dapat menggunakan konsol atau API Gateway REST API untuk mengubah pengaturan panggung, termasuk caching API dan logging. Langkah-langkah berikut menunjukkan cara melakukannya menggunakan editor Stage dari konsol API Gateway.

Perbarui pengaturan panggung menggunakan konsol API Gateway

Langkah-langkah ini mengasumsikan bahwa Anda telah menerapkan API ke sebuah panggung.

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih REST API.
3. Di panel navigasi utama, pilih Tahapan di bawah API.
4. Di panel Tahapan, pilih nama panggung.
5. Di bagian Detail tahap, pilih Edit.
6. (Opsional) Untuk deskripsi Tahap, edit deskripsi.
7. Untuk pengaturan tambahan, Anda mengubah pengaturan berikut:

   Pengaturan cache

   Untuk mengaktifkan caching API untuk stage, aktifkan cache API Provision. Kemudian konfigurasikan caching tingkat metode Default, Kapasitas cache, Enkripsi data cache, Cache time-to-live (TTL), serta persyaratan apa pun untuk pembatalan cache per kunci.

   Caching tidak aktif sampai Anda mengaktifkan caching tingkat metode default atau mengaktifkan cache tingkat metode untuk metode tertentu.

   Untuk informasi selengkapnya tentang pengaturan cache, lihatMengaktifkan caching API untuk meningkatkan daya tanggap.

   > **Note**
   >
   > Jika Anda mengaktifkan caching API untuk tahap API, AWS akun Anda mungkin dikenakan biaya untuk caching API. Caching tidak memenuhi syarat untuk Tingkat AWS Gratis.

   Pengaturan pelambatan

   Untuk menetapkan target pelambatan tingkat tahap untuk semua metode yang terkait dengan API ini, aktifkan Throttling.

   Untuk Rate, masukkan target rate. Ini adalah tarif, dalam permintaan per detik, token ditambahkan ke ember token. Tingkat tingkat tahap tidak boleh lebih dari tingkat tingkat akun

seperti yang ditentukan dalam. Kuota API Gateway untuk mengonfigurasi dan menjalankan REST API

Untuk Burst, masukkan target burst rate. Tingkat ledakan, adalah kapasitas ember token. Ini memungkinkan lebih banyak permintaan melalui untuk jangka waktu tertentu daripada tingkat target. Tingkat ledakan tingkat tahap ini tidak boleh lebih dari tingkat burst rate akun seperti yang ditentukan dalam. Kuota API Gateway untuk mengonfigurasi dan menjalankan REST API

> **Note**
>
> Tarif pelambatan bukanlah batas yang sulit, dan diterapkan atas dasar upaya terbaik. Dalam beberapa kasus, klien dapat melebihi target yang Anda tetapkan. Jangan mengandalkan pembatasan untuk mengontrol biaya atau memblokir akses ke API. Pertimbangkan AWS Budgetsuntuk menggunakan untuk memantau biaya dan AWS WAFmengelola permintaan API.

Pengaturan firewall dan sertifikat

Untuk mengaitkan ACL AWS WAF web dengan panggung, pilih ACL web dari daftar dropdown Web ACL. Jika diinginkan, pilih Blokir Permintaan API jika WebACL tidak dapat dievaluasi (Gagal- Tutup).

Untuk memilih sertifikat klien untuk tahap Anda, pilih sertifikat dari menu tarik-turun sertifikat Klien.

8. Pilih Simpan.
9. Untuk mengaktifkan CloudWatch Log Amazon untuk semua metode yang terkait dengan tahap API Gateway API ini, di bagian Log dan penelusuran, pilih Edit.

> **Note**
>
> Untuk mengaktifkan CloudWatch Log, Anda juga harus menentukan ARN dari peran IAM yang memungkinkan API Gateway untuk menulis informasi ke CloudWatch Log atas nama pengguna Anda. Untuk melakukannya, pilih Pengaturan dari panel navigasi utama API. Kemudian, untuk peran CloudWatch log, masukkan ARN dari peran IAM.

Untuk skenario aplikasi umum, peran IAM dapat melampirkan kebijakan terkelolaAmazonAPIGatewayPushToCloudWatchLogs, yang berisi pernyataan kebijakan akses berikut:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents",
                "logs:GetLogEvents",
                "logs:FilterLogEvents"
            ],
            "Resource": "*"
        }
    ]
}
```

Peran IAM juga harus berisi pernyataan hubungan kepercayaan berikut:

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

Untuk informasi selengkapnya CloudWatch, lihat Panduan CloudWatch Pengguna Amazon.

10. Untuk tingkat log, pilih dari yang berikut ini:
    - Untuk hanya menulis entri tingkat kesalahan ke CloudWatch Log, pilih Error only.
    - Untuk menulis hanya entri tingkat kesalahan ke CloudWatch Log, atau pilih Kesalahan dan log info untuk menyertakan semua peristiwa ERROR, serta peristiwa informasi tambahan.
    - Untuk mencatat permintaan panggilan API lengkap dan informasi respons, pilih Log permintaan dan respons lengkap. Tidak ada data sensitif yang dicatat kecuali opsi Log permintaan dan respons lengkap dipilih.
11. Agar API Gateway melaporkan ke CloudWatch metrik APIAPI `calls`,,, `Latency`Integration `latency`, dan `400 errors`500 `errors`, pilih Metrik terperinci. Untuk informasi selengkapnya CloudWatch, lihat Panduan CloudWatch Pengguna Amazon.

    > **Important**
    >
    > Akun Anda dikenakan biaya untuk mengakses metrik tingkat metode, tetapi bukan CloudWatch metrik tingkat API atau tingkat tahap.

12. Untuk mengaktifkan pencatatan akses ke tujuan, aktifkan Pencatatan akses khusus.
13. Untuk ARN tujuan log Access, masukkan ARN grup log atau aliran Firehose.

    Format ARN untuk Firehose adalah. `arn:aws:firehose:`*`{region}`*`:`*`{account-id}`*`:deliverystream/amazon-apigateway-`*`{your-stream-name}`* Nama aliran Firehose Anda harus. `amazon-apigateway-`*`{your-stream-name}`*
14. Dalam format Log, masukkan format log. Untuk mempelajari lebih lanjut tentang contoh format log, lihatthe section called “CloudWatch format log untuk API Gateway”.
15. Untuk mengaktifkan AWS X-Raypenelusuran untuk tahap API, pilih penelusuran X-Ray. Untuk informasi selengkapnya, lihat Menelusuri permintaan pengguna ke REST API menggunakan X-Ray.
16. Pilih Simpan perubahan. Menerapkan ulang API Anda agar pengaturan baru diterapkan.

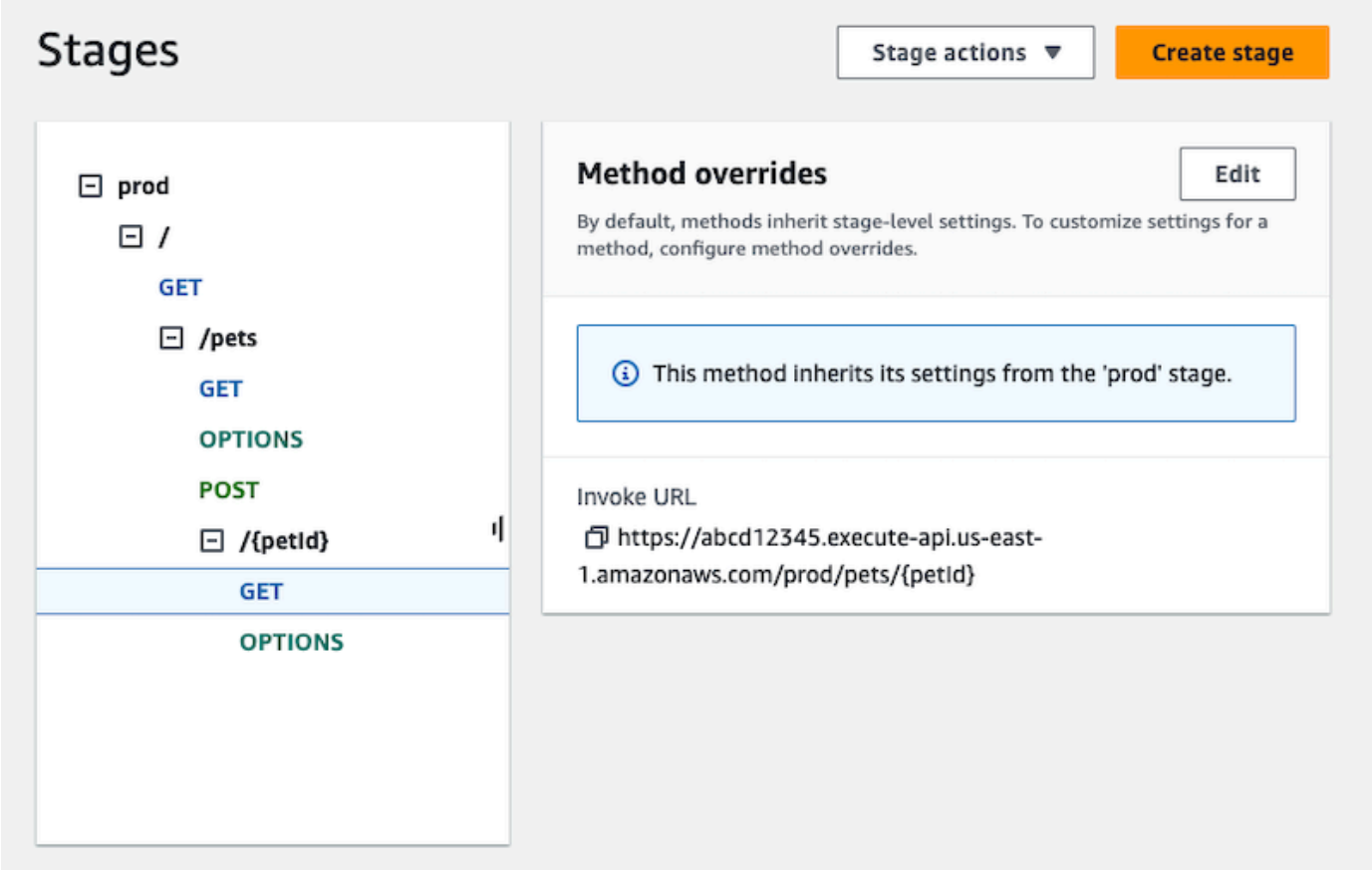
Ganti pengaturan tingkat panggung

Anda dapat mengganti pengaturan tingkat tahap yang diaktifkan berikut ini. Beberapa opsi ini dapat mengakibatkan biaya tambahan untuk Anda Akun AWS.

Ganti pengaturan level tahap menggunakan konsol API Gateway

Untuk mengganti setelan tingkat tahap menggunakan konsol API Gateway

1. Untuk mengonfigurasi penggantian metode, perluas panggung di bawah panel navigasi sekunder, lalu pilih metode.



2. Untuk penggantian Metode, pilih Edit.
3. Untuk mengaktifkan CloudWatch pengaturan tingkat metode, untuk CloudWatch Log, pilih level logging.
4. Untuk mengaktifkan metrik detail tingkat metode, pilih Metrik terperinci. Akun Anda dikenakan biaya untuk mengakses metrik tingkat metode, tetapi bukan CloudWatch metrik tingkat API atau tingkat tahap.
5. Untuk mengaktifkan pelambatan tingkat metode, pilih Throttling. Masukkan opsi tingkat metode yang sesuai. Untuk mempelajari selengkapnya tentang throttling, lihat. the section called “Throttling”
6. Untuk mengkonfigurasi cache tingkat metode, pilih Aktifkan cache metode. Jika Anda mengubah pengaturan caching tingkat metode default di detail Tahap, itu tidak memengaruhi pengaturan ini.

7. Pilih Simpan.

Hapus panggung

Ketika Anda tidak lagi membutuhkan panggung, Anda dapat menghapusnya untuk menghindari membayar sumber daya yang tidak digunakan. Langkah-langkah berikut menunjukkan cara menggunakan konsol API Gateway untuk menghapus panggung.

> **Warning**
>
> Menghapus tahap dapat menyebabkan sebagian atau semua API terkait tidak dapat digunakan oleh pemanggil API. Menghapus tahap tidak dapat dibatalkan, tetapi Anda dapat membuat ulang panggung dan mengaitkannya dengan penerapan yang sama.

Menghapus panggung menggunakan konsol API Gateway

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih REST API.
3. Di panel navigasi utama, pilih Tahapan.
4. Di panel Tahapan, pilih tahap yang ingin Anda hapus, lalu pilih Tindakan tahap, Hapus tahap.
5. Saat Anda diminta, masukkan**`confirm`**, lalu pilih Hapus.

Menyiapkan tag untuk tahap API di API Gateway

Di API Gateway, Anda dapat menambahkan tag ke tahap API, menghapus tag dari tahap, atau melihat tag. Untuk melakukannya, Anda dapat menggunakan konsol API Gateway, AWS CLI/SDK, atau API Gateway REST API.

Sebuah panggung juga dapat mewarisi tag dari REST API induknya. Untuk informasi selengkapnya, lihat the section called “Warisan tag di API Amazon API Gateway V1”.

Untuk informasi selengkapnya tentang menandai resource API Gateway, lihatPenandaan.

Topik

- Menyiapkan tag untuk tahap API menggunakan konsol API Gateway
- Menyiapkan tag untuk tahap API menggunakan AWS CLI

- Menyiapkan tag untuk tahap API menggunakan API Gateway REST API

Menyiapkan tag untuk tahap API menggunakan konsol API Gateway

Prosedur berikut menjelaskan cara menyiapkan tag untuk tahap API.

Untuk menyiapkan tag untuk tahap API dengan menggunakan konsol API Gateway

1. Masuk ke konsol API Gateway.
2. Pilih API yang sudah ada, atau buat API baru yang menyertakan sumber daya, metode, dan integrasi terkait.
3. Pilih tahap atau terapkan API ke tahap baru.
4. Di panel navigasi utama, pilih Tahapan.
5. Pilih tab Tanda. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.
6. Pilih Kelola tanda.
7. Di Editor Tag, pilih Tambahkan tag. Masukkan kunci tag (misalnya,`Department`) di bidang Kunci, dan masukkan nilai tag (misalnya,`Sales`) di bidang Nilai. Pilih Simpan untuk menyimpan tag.
8. Jika perlu, ulangi langkah 5 untuk menambahkan lebih banyak tag ke tahap API. Jumlah maksimum tag per tahap adalah 50.
9. Untuk menghapus tag yang ada dari panggung, pilih Hapus.
10. Jika API telah digunakan sebelumnya di konsol API Gateway, Anda perlu menerapkannya kembali agar perubahan diterapkan.

Menyiapkan tag untuk tahap API menggunakan AWS CLI

Anda dapat mengatur tag untuk tahap API menggunakan perintah menggunakan perintah create-stage atau perintah tag-resource. AWS CLI Anda dapat menghapus satu atau beberapa tag dari tahap API menggunakan perintah untag-resource.

Contoh berikut menambahkan tag saat membuat `test` panggung:

```
aws apigateway create-stage --rest-api-id abc1234 --stage-name test --description
 'Testing stage' --deployment-id efg456 --tag Department=Sales
```

Contoh berikut menambahkan tag ke `prod` panggung:

```
aws apigateway tag-resource --resource-arn arn:aws:apigateway:us-east-2::/
restapis/abc123/stages/prod --tags Department=Sales
```

Contoh berikut menghapus `Department=Sales` tag dari `test` panggung:

```
aws apigateway untag-resource --resource-arn arn:aws:apigateway:us-east-2::/
restapis/abc123/stages/test --tag-keys Department
```

Menyiapkan tag untuk tahap API menggunakan API Gateway REST API

Anda dapat menyiapkan tag untuk tahap API menggunakan API Gateway REST API dengan melakukan salah satu hal berikut:

- Panggilan `tags:tag`untuk menandai tahap API.
- Panggilan `tags:untag`untuk menghapus satu atau beberapa tag dari tahap API.
- Panggil `stage:create`untuk menambahkan satu atau beberapa tag ke tahap API yang Anda buat.

Anda juga dapat menelepon `tags:get`untuk mendeskripsikan tag dalam tahap API.

Menandai tahap API

Setelah Anda menerapkan API (`m5zr3vnks7`) ke stage (`test`), beri tag stage dengan memanggil `tags:tag`. Tahap yang diperlukan Nama Sumber Daya Amazon (ARN) (`arn:aws:apigateway:us-east-1::/restapis/m5zr3vnks7/stages/test`) harus dikodekan URL (). `arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis%2Fm5zr3vnks7%2Fstages%2Ftest`

```
PUT /tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis%2Fm5zr3vnks7%2Fstages
%2Ftest

{
  "tags" : {
    "Department" : "Sales"
  }
}
```

Anda juga dapat menggunakan permintaan sebelumnya untuk memperbarui tag yang ada ke nilai baru.

Anda dapat menambahkan tag ke panggung saat memanggil `stage:create`untuk membuat panggung:

```
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"
  }
}
```

Membatalkan tag tahap API

Untuk menghapus `Department` tag dari panggung, hubungi `tags:untag`:

```
DELETE /tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis%2Fm5zr3vnks7%2Fstages
%2Ftest?tagKeys=Department
Host: apigateway.us-east-1.amazonaws.com
Authorization: ...
```

Untuk menghapus lebih dari satu tag, gunakan daftar kunci tag yang dipisahkan koma dalam ekspresi kueri—misalnya,. `?tagKeys=Department,Division,…`

Jelaskan tag untuk tahap API

Untuk mendeskripsikan tag yang ada pada tahap tertentu, panggil `tags:get`:

```
GET /tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis%2Fm5zr3vnks7%2Fstages
%2Ftags
Host: apigateway.us-east-1.amazonaws.com
```

```
Authorization: ...
```

Respons yang berhasil mirip dengan yang berikut:

```
200 OK

{
    "_links": {
        "curies": {
            "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/
restapi-tags-{rel}.html",
            "name": "tags",
            "templated": true
        },
        "tags:tag": {
            "href": "/tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis
%2Fm5zr3vnks7%2Fstages%2Ftags"
        },
        "tags:untag": {
            "href": "/tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis
%2Fm5zr3vnks7%2Fstages%2Ftags{?tagKeys}",
            "templated": true
        }
    },
    "tags": {
        "Department": "Sales"
    }
}
```

Menyiapkan variabel tahap untuk penerapan REST API

Variabel tahap adalah pasangan nama-nilai yang dapat Anda definisikan sebagai atribut konfigurasi yang terkait dengan tahap penerapan REST API. Mereka bertindak seperti variabel lingkungan dan dapat digunakan dalam template penyiapan dan pemetaan API Anda.

Misalnya, Anda dapat menentukan variabel tahap dalam konfigurasi tahap, dan kemudian menetapkan nilainya sebagai string URL dari integrasi HTTP untuk metode di REST API Anda. Kemudian, Anda dapat mereferensikan string URL dengan menggunakan nama variabel tahap terkait dari penyiapan API. Dengan melakukan ini, Anda dapat menggunakan penyiapan API yang sama dengan titik akhir yang berbeda di setiap tahap dengan mengatur ulang nilai variabel tahap ke URL yang sesuai.

Anda juga dapat mengakses variabel tahap dalam template pemetaan, atau meneruskan parameter konfigurasi ke backend AWS Lambda atau HTTP Anda.

Untuk informasi selengkapnya tentang templat pemetaan, lihatTemplate pemetaan API Gateway dan referensi variabel pencatatan akses.

> **Note**
>
> Variabel tahap tidak dimaksudkan untuk digunakan untuk data sensitif, seperti kredensial. Untuk meneruskan data sensitif ke integrasi, gunakan AWS Lambda otorisasi. Anda dapat meneruskan data sensitif ke integrasi dalam output otorisasi Lambda. Untuk mempelajari selengkapnya, lihat the section called “Keluaran dari otorisasi Lambda Amazon API Gateway”.

Kasus penggunaan

Dengan tahapan penerapan di API Gateway, Anda dapat mengelola beberapa tahapan rilis untuk setiap API, seperti alfa, beta, dan produksi. Dengan menggunakan variabel tahap, Anda dapat mengonfigurasi tahap penerapan API untuk berinteraksi dengan titik akhir backend yang berbeda.

Misalnya, API Anda dapat meneruskan permintaan GET sebagai proxy HTTP ke host web backend (misalnya,`http://example.com`). Dalam hal ini, host web backend dikonfigurasi dalam variabel tahap sehingga ketika pengembang memanggil titik akhir produksi Anda, API Gateway memanggil example.com. Saat Anda memanggil titik akhir beta, API Gateway menggunakan nilai yang dikonfigurasi dalam variabel tahap untuk tahap beta, dan memanggil host web yang berbeda (misalnya,`beta.example.com`). Demikian pula, variabel tahap dapat digunakan untuk menentukan nama AWS Lambda fungsi yang berbeda untuk setiap tahap di API Anda.

Anda juga dapat menggunakan variabel tahap untuk meneruskan parameter konfigurasi ke fungsi Lambda melalui templat pemetaan Anda. Misalnya, Anda mungkin ingin menggunakan kembali fungsi Lambda yang sama untuk beberapa tahap di API Anda, tetapi fungsi tersebut harus membaca data dari tabel Amazon DynamoDB yang berbeda tergantung pada tahap mana yang dipanggil. Dalam template pemetaan yang menghasilkan permintaan untuk fungsi Lambda, Anda dapat menggunakan variabel tahap untuk meneruskan nama tabel ke Lambda.

Contoh-contoh

Untuk menggunakan variabel tahap untuk menyesuaikan titik akhir integrasi HTTP, Anda harus terlebih dahulu mengkonfigurasi variabel tahap dari nama tertentu (misalnya,**`url`**), dan kemudian

menetapkannya nilai, (misalnya,**`example.com`**). Selanjutnya, dari konfigurasi metode Anda, siapkan integrasi proxy HTTP. Alih-alih memasukkan URL titik akhir, Anda dapat memberi tahu API Gateway untuk menggunakan nilai variabel stage,**`http://${stageVariables.url}`**. Nilai ini memberi tahu API Gateway untuk mengganti variabel stage Anda `${}` saat runtime, tergantung pada tahap mana API Anda berjalan.

Anda dapat mereferensikan variabel tahap dengan cara yang sama untuk menentukan nama fungsi Lambda, jalur Proxy AWS Layanan, atau AWS ARN peran di bidang kredensyal.

Saat menentukan nama fungsi Lambda sebagai nilai variabel tahap, Anda harus mengonfigurasi izin pada fungsi Lambda secara manual. Saat Anda menentukan fungsi Lambda di konsol API Gateway, sebuah AWS CLI perintah akan muncul untuk mengonfigurasi izin yang tepat. Anda juga dapat menggunakan AWS Command Line Interface (AWS CLI) untuk melakukan ini.

```
aws lambda add-permission --function-name "arn:aws:lambda:us-
east-2:123456789012:function:my-function" --source-arn "arn:aws:execute-api:us-
east-2:123456789012:api_id/*/HTTP_METHOD/resource" --principal apigateway.amazonaws.com
 --statement-id apigateway-access --action lambda:InvokeFunction
```

Menyetel variabel tahapan menggunakan konsol Amazon API Gateway

Dalam tutorial ini, Anda mempelajari cara mengatur variabel tahap untuk dua tahap penerapan API sampel dengan menggunakan konsol Amazon API Gateway. Sebelum Anda mulai, pastikan prasyarat berikut terpenuhi:

- Anda harus memiliki API yang tersedia di API Gateway. Ikuti petunjuk dalam Membuat REST API di Amazon API Gateway.
- Anda harus telah menerapkan API setidaknya sekali. Ikuti petunjuk dalam Menerapkan REST API di Amazon API Gateway.
- Anda harus telah membuat tahap pertama untuk API yang diterapkan. Ikuti petunjuk dalam Buat panggung baru.

Untuk mendeklarasikan variabel stage menggunakan konsol API Gateway

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Buat API, lalu buat `GET` metode pada sumber daya root API. Atur tipe integrasi ke HTTP dan atur URL Endpoint ke**`http://${stageVariables.url}`**.

3. Menerapkan API ke tahap baru bernama**`beta`**.
4. Di panel navigasi utama, pilih Tahapan, lalu pilih tahap beta.
5. Pada tab variabel Stage, pilih Edit.
6. Pilih Tambahkan variabel tahap.
7. Untuk Nama, masukkan **`url`**. Untuk nilai, masukkan**`httpbin.org/get`**.
8. Pilih Tambahkan variabel tahap, dan kemudian lakukan hal berikut:

   Untuk Nama, masukkan **`stageName`**. Untuk nilai, masukkan**`beta`**.
9. Pilih Tambahkan variabel tahap, dan kemudian lakukan hal berikut:

   Untuk Nama, masukkan **`function`**. Untuk nilai, masukkan**`HelloWorld`**.

   > **Note**
   >
   > Saat menyetel fungsi Lambda sebagai nilai variabel tahap, gunakan nama lokal fungsi, mungkin termasuk alias atau spesifikasi versinya, seperti dalam**`HelloWorld`**, atau. **`HelloWorld:1`** **`HelloWorld:alpha`** Jangan gunakan ARN fungsi (misalnya,**`arn:aws:lambda:us-east-1:123456789012:function:HelloWorld`**). Konsol API Gateway mengasumsikan nilai variabel stage untuk fungsi Lambda sebagai nama fungsi yang tidak memenuhi syarat dan memperluas variabel tahap yang diberikan menjadi ARN.
10. Pilih Simpan.
11. Sekarang buat tahap kedua. Dari panel navigasi Tahapan, pilih Buat tahap. Untuk nama Panggung, masukkan**`prod`**. Pilih penerapan terbaru dari Deployment, lalu pilih Create stage.
12. Seperti tahap beta, atur variabel tiga tahap yang sama (url, stageName, dan fungsi) ke nilai yang berbeda (**`petstore-demo-endpoint.execute-api.com/petstore/pets`**,, dan**`HelloEveryone`**)**`prod`**, masing-masing.

    Untuk mempelajari cara menggunakan variabel tahap, lihatthe section called “Gunakan variabel tahap ”.

Menggunakan variabel tahap Amazon API Gateway

Anda dapat menggunakan variabel tahap API Gateway untuk mengakses backend HTTP dan Lambda untuk tahapan penerapan API yang berbeda. Anda juga dapat menggunakan variabel tahap

untuk meneruskan metadata konfigurasi spesifik tahap ke backend HTTP sebagai parameter kueri dan ke dalam fungsi Lambda sebagai muatan yang dihasilkan dalam template pemetaan input.

Prasyarat

Anda harus membuat dua tahap dengan variabel tahap url yang disetel ke dua titik akhir HTTP yang berbeda: variabel tahap fungsi yang ditetapkan ke dua fungsi Lambda yang berbeda, dan variabel tahap StageName yang berisi metadata spesifik tahap.

Mengakses titik akhir HTTP melalui API dengan variabel tahap

1. Di panel navigasi Tahapan, pilih beta. Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda, lalu masukkan URL pemanggilan API Anda di browser web. Ini memulai `GET` permintaan tahap beta pada sumber daya root API.

   > **Note**
   >
   > Tautan URL Invoke menunjuk ke sumber daya root API dalam tahap beta. Memasukkan URL di browser web memanggil `GET` metode tahap beta pada sumber daya root. Jika metode didefinisikan pada sumber daya anak dan bukan pada sumber daya root itu sendiri, memasukkan URL di browser web mengembalikan respons `{"message":"Missing Authentication Token"}` kesalahan. Dalam hal ini, Anda harus menambahkan nama sumber daya anak tertentu ke tautan URL Panggilan.

2. Respons yang Anda dapatkan dari `GET` permintaan tahap beta ditampilkan berikutnya. Anda juga dapat memverifikasi hasilnya dengan menggunakan browser untuk menavigasi ke http://httpbin.org/get. Nilai ini ditetapkan ke `url` variabel dalam tahap beta. Kedua tanggapan itu identik.
3. Di panel navigasi Tahapan, pilih tahap prod. Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda, lalu masukkan URL pemanggilan API Anda di browser web. Ini memulai `GET` permintaan tahap prod pada sumber daya root API.
4. Respons yang Anda dapatkan dari `GET` permintaan tahap prod ditampilkan berikutnya. Anda dapat memverifikasi hasilnya dengan menggunakan browser untuk menavigasi ke http://petstore-demo-endpoint.execute-api.com/petstore/pets. Nilai ini ditetapkan ke `url` variabel dalam tahap prod. Kedua tanggapan itu identik.

Lulus metadata spesifik tahap ke backend HTTP melalui variabel tahap dalam ekspresi parameter kueri

Prosedur ini menjelaskan cara menggunakan nilai variabel tahap dalam ekspresi parameter kueri untuk meneruskan metadata spesifik tahap ke backend HTTP. Kita akan menggunakan variabel `stageName` stage yang dideklarasikan di[Menyetel variabel tahapan menggunakan konsol Amazon API Gateway](#).

1. Di panel navigasi Resource, pilih metode GET.

   Untuk menambahkan parameter string kueri ke URL metode, pilih tab Permintaan metode, lalu di bagian Pengaturan permintaan metode, pilih Edit.
2. Pilih parameter string kueri URL dan lakukan hal berikut:

   a. Pilih Tambahkan string kueri.

   b. Untuk Nama, masukkan **`stageName`**.

   c. Tetap Diperlukan dan Caching dimatikan.
3. Pilih Simpan.
4. Pilih tab Permintaan integrasi, dan kemudian di bagian Pengaturan permintaan integrasi, pilih Edit.
5. Untuk URL Endpoint, tambahkan **`?stageName=${stageVariables.stageName}`** ke nilai URL yang ditentukan sebelumnya, sehingga seluruh URL Endpoint adalah. **`http://${stageVariables.url}?stageName=${stageVariables.stageName}`**
6. Pilih Deploy API dan pilih tahap beta.
7. Di panel navigasi utama, pilih Tahapan. Di panel navigasi Tahapan, pilih beta. Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda, lalu masukkan URL pemanggilan API Anda di browser web.

   > **Note**
   >
   > Kami menggunakan tahap beta di sini karena titik akhir HTTP (seperti yang ditentukan oleh `url` variabel "http://httpbin.org/get “) menerima ekspresi parameter kueri dan mengembalikannya sebagai `args` objek dalam responsnya.
8. Anda mendapatkan tanggapan berikut. Perhatikan bahwa`beta`, ditugaskan ke variabel `stageName` tahap, diteruskan di backend sebagai argumen. `stageName`

Panggil fungsi Lambda melalui API dengan variabel panggung

Prosedur ini menjelaskan cara menggunakan variabel stage untuk memanggil fungsi Lambda sebagai backend API Anda. Kita akan menggunakan variabel `function` stage yang dideklarasikan sebelumnya. Untuk informasi selengkapnya, lihat Menyetel variabel tahapan menggunakan konsol Amazon API Gateway.

1. Buat fungsi Lambda bernama **`HelloWorld`** menggunakan runtime Node.js default. Kode harus berisi yang berikut:

   ```
   export const handler = function(event, context, callback) {
       if (event.stageName)
           callback(null, 'Hello, World! I\'m calling from the ' + event.stageName + '
    stage.');
       else
           callback(null, 'Hello, World! I\'m not sure where I\'m calling from...');
   };
   ```

   Untuk informasi selengkapnya tentang cara membuat fungsi Lambda, lihat Memulai konsol REST API.
2. Di panel Resources, pilih Buat sumber daya, lalu lakukan hal berikut:

   a. Untuk jalur Sumber Daya, pilih/.

   b. Untuk Nama sumber daya, masukkan **`lambdav1`**.

   c. Pilih Buat sumber daya.

3. Pilih sumber daya /lambdav1, lalu pilih Create method.

   Kemudian, lakukan hal berikut:

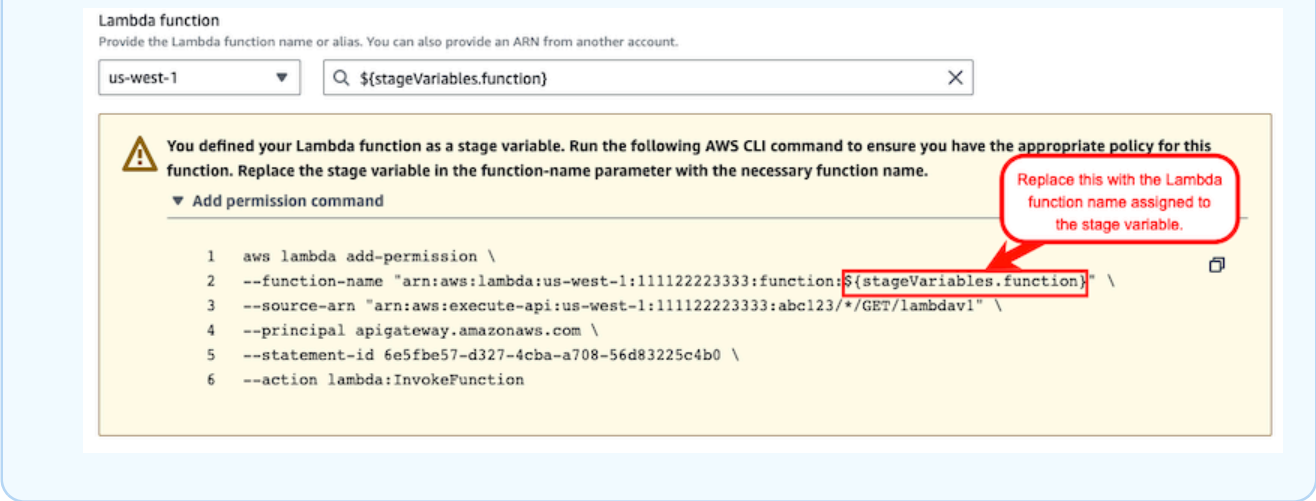
   a. Untuk tipe Metode, pilih GET.

   b. Untuk jenis Integrasi, pilih fungsi Lambda.

   c. Matikan integrasi proxy Lambda.

   d. Untuk fungsi Lambda, masukkan. `${stageVariables.function}`

> **Tip**
>
> Saat diminta dengan perintah Tambah izin, salin perintah AWS CLI. Jalankan perintah pada setiap fungsi Lambda yang akan ditugaskan ke variabel `function` panggung. Misalnya, jika `$stageVariables.function` nilainya`HelloWorld`, jalankan AWS CLI perintah berikut:
>
> ```
> aws lambda add-permission --function-name arn:aws:lambda:us-
> east-1:account-id:function:HelloWorld --source-arn arn:aws:execute-
> api:us-east-1:account-id:api-id/*/GET/lambdav1 --principal
>  apigateway.amazonaws.com --statement-id statement-id-guid --action
>  lambda:InvokeFunction
> ```
>
> Gagal melakukannya menghasilkan `500 Internal Server Error` respons saat menjalankan metode. Ganti `${stageVariables.function}` dengan nama fungsi Lambda yang ditetapkan ke variabel stage.



e. Pilih metode Buat.

4. Menerapkan API ke kedua tahapan **`prod`** dan **`beta`** tahap.
5. Di panel navigasi utama, pilih Tahapan. Di panel navigasi Tahapan, pilih beta. Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda, lalu masukkan URL pemanggilan API Anda di browser web. Tambahkan **`/lambdav1`** ke URL sebelum Anda menekan enter.

   Anda mendapatkan tanggapan berikut.

```
"Hello, World! I'm not sure where I'm calling from..."
```

Lewati metadata spesifik tahap ke fungsi Lambda melalui variabel panggung

Prosedur ini menjelaskan cara menggunakan variabel tahap untuk meneruskan metadata konfigurasi spesifik tahap ke dalam fungsi Lambda. Anda membuat `POST` metode dan template pemetaan masukan untuk menghasilkan payload menggunakan variabel `stageName` stage yang Anda deklarasikan sebelumnya.

1. Pilih sumber daya /lambdav1, lalu pilih Create method.

   Kemudian, lakukan hal berikut:

   a. Untuk jenis Metode, pilih POST.

   b. Untuk jenis Integrasi, pilih fungsi Lambda.

   c. Matikan integrasi proxy Lambda.

   d. Untuk fungsi Lambda, masukkan. `${stageVariables.function}`

e. Saat diminta dengan perintah Tambah izin, salin perintah AWS CLI. Jalankan perintah pada setiap fungsi Lambda yang akan ditugaskan ke variabel `function` panggung.

f. Pilih metode Buat.

2. Pilih tab Permintaan integrasi, dan kemudian di bagian Pengaturan permintaan integrasi, pilih Edit.
3. Pilih Templat pemetaan, lalu pilih Tambahkan templat pemetaan.
4. Untuk jenis Konten, masukkan**`application/json`**.
5. Untuk badan Template, masukkan template berikut:

```
#set($inputRoot = $input.path('$'))
{
    "stageName" : "$stageVariables.stageName"
}
```

> **Note**
>
> Dalam template pemetaan, variabel tahap harus direferensikan dalam tanda kutip (seperti dalam `"$stageVariables.stageName"` atau`"${stageVariables.stageName}"`). Di tempat lain, itu harus direferensikan tanpa tanda kutip (seperti dalam`${stageVariables.function}`).

6. Pilih Simpan.
7. Menerapkan API ke kedua tahapan **`beta`** dan **`prod`** tahap.
8. Untuk menggunakan klien REST API untuk meneruskan metadata khusus tahap, lakukan hal berikut:

a. Di panel navigasi Tahapan, pilih beta. Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda, lalu masukkan URL pemanggilan API Anda di bidang input klien REST API. Tambahkan **`/lambdav1`** sebelum Anda mengirimkan permintaan Anda.

Anda mendapatkan tanggapan berikut.

```
"Hello, World! I'm calling from the beta stage."
```

b. Di panel navigasi Tahapan, pilih prod. Di bawah Detail tahap, pilih ikon salin untuk menyalin URL pemanggilan API Anda, lalu masukkan URL pemanggilan API Anda di bidang input klien REST API. Tambahkan **`/lambdav1`** sebelum Anda mengirimkan permintaan Anda.

Anda mendapatkan tanggapan berikut.

```
"Hello, World! I'm calling from the prod stage."
```

9. Untuk menggunakan fitur Uji untuk meneruskan metadata khusus tahapan, lakukan hal berikut:

   a. Di panel navigasi Sumber daya, pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab.

   b. Untuk fungsi, masukkan**`HelloWorld`**.

   c. Untuk StageName, masukkan. **`beta`**

   d. Pilih Uji. Anda tidak perlu menambahkan badan ke `POST` permintaan Anda.

      Anda mendapatkan tanggapan berikut.

      ```
      "Hello, World! I'm calling from the beta stage."
      ```

   e. Anda dapat mengulangi langkah-langkah sebelumnya untuk menguji tahap Prod. Untuk StageName, masukkan. **`Prod`**

      Anda mendapatkan tanggapan berikut.

      ```
      "Hello, World! I'm calling from the prod stage."
      ```

Referensi variabel tahap Amazon API Gateway

Anda dapat menggunakan variabel tahap API Gateway dalam kasus berikut.

Ekspresi pemetaan parameter

Variabel tahap dapat digunakan dalam ekspresi pemetaan parameter untuk permintaan metode API atau parameter header respons, tanpa substitusi sebagian. Dalam contoh berikut, variabel tahap direferensikan tanpa `$` dan `{...}` terlampir.

- `stageVariables.<variable_name>`

Templat pemetaan

Variabel tahap dapat digunakan di mana saja dalam template pemetaan, seperti yang ditunjukkan pada contoh berikut.

- `{ "name" : "$stageVariables.<variable_name>"}`
- `{ "name" : "${stageVariables.<variable_name>}"}`

URI integrasi HTTP

Variabel tahap dapat digunakan sebagai bagian dari URL integrasi HTTP, seperti yang ditunjukkan pada contoh berikut:

- URI lengkap tanpa protokol — `http://${stageVariables.<variable_name>}`
- Domain lengkap — `http://${stageVariables.<variable_name>}/resource/operation`
- Sebuah subdomain — `http://${stageVariables.<variable_name>}.example.com/resource/operation`
- Sebuah jalan — `http://example.com/${stageVariables.<variable_name>}/bar`
- Sebuah string kueri - `http://example.com/foo?q=${stageVariables.<variable_name>}`

AWS URI integrasi

Variabel tahap dapat digunakan sebagai bagian dari tindakan AWS URI atau komponen jalur, seperti yang ditunjukkan pada contoh berikut.

- `arn:aws:apigateway:<region>:<service>:${stageVariables.<variable_name>}`

AWS URI integrasi (fungsi Lambda)

Variabel tahap dapat digunakan sebagai pengganti nama fungsi Lambda, atau versi/alias, seperti yang ditunjukkan pada contoh berikut.

- `arn:aws:apigateway:<region>:lambda:path/2015-03-31/functions/arn:aws:lambda:<region>:<account_id>:function:${stageVariables.<function_variable_name>}/invocations`

- `arn:aws:apigateway:<region>:lambda:path/2015-03-31/functions/arn:aws:lambda:<region>:<account_id>:function:<function_name>:${stageVariables.<version_variable_name>}/invocations`

> **Note**
>
> Untuk menggunakan variabel stage untuk fungsi Lambda, fungsi tersebut harus berada di akun yang sama dengan API. Variabel tahap tidak mendukung fungsi Lambda lintas akun.

AWS kredensi integrasi

Variabel tahap dapat digunakan sebagai bagian dari ARN kredensi AWS pengguna/peran, seperti yang ditunjukkan pada contoh berikut.

- `arn:aws:iam::<account_id>:${stageVariables.<variable_name>}`

Siapkan penerapan rilis kenari API Gateway

Rilis Canary adalah strategi pengembangan perangkat lunak di mana versi baru API (serta perangkat lunak lainnya) digunakan untuk tujuan pengujian, dan versi dasar tetap digunakan sebagai rilis produksi untuk operasi normal pada tahap yang sama. Untuk tujuan diskusi, kami merujuk ke versi dasar sebagai rilis produksi dalam dokumentasi ini. Meskipun ini masuk akal, Anda bebas menerapkan rilis kenari pada versi non-produksi apa pun untuk pengujian.

Dalam penerapan rilis kenari, total lalu lintas API dipisahkan secara acak menjadi rilis produksi dan rilis kenari dengan rasio yang telah dikonfigurasi sebelumnya. Biasanya, rilis kenari menerima sebagian kecil lalu lintas API dan rilis produksi memakan sisanya. Fitur API yang diperbarui hanya dapat dilihat oleh lalu lintas API melalui canary. Anda dapat menyesuaikan persentase lalu lintas kenari untuk mengoptimalkan cakupan atau kinerja pengujian.

Dengan menjaga lalu lintas kenari kecil dan pemilihan acak, sebagian besar pengguna tidak terpengaruh setiap saat oleh potensi bug di versi baru, dan tidak ada satu pengguna pun yang terpengaruh sepanjang waktu.

Setelah metrik pengujian memenuhi persyaratan Anda, Anda dapat mempromosikan rilis kenari ke rilis produksi dan menonaktifkan kenari dari penerapan. Ini membuat fitur-fitur baru tersedia dalam tahap produksi.

Topik

Penerapan rilis Canary di API Gateway

Di API Gateway, penerapan rilis kenari menggunakan tahap penerapan untuk rilis produksi versi dasar API, dan melampirkan ke tahap rilis kenari untuk versi baru, relatif terhadap versi dasar, API. Tahap ini dikaitkan dengan penyebaran awal dan kenari dengan penyebaran berikutnya. Pada awalnya, baik stage maupun canary menunjuk ke versi API yang sama. Kami menggunakan rilis panggung dan produksi secara bergantian dan menggunakan pelepasan kenari dan kenari secara bergantian di seluruh bagian ini.

Untuk menerapkan API dengan rilis kenari, Anda membuat penerapan rilis kenari dengan menambahkan setelan kenari ke tahap penerapan reguler. Pengaturan kenari menjelaskan rilis kenari yang mendasarinya dan tahapannya mewakili rilis produksi API dalam penerapan ini. Untuk menambahkan pengaturan kenari, atur `canarySettings` pada tahap penerapan dan tentukan yang berikut ini:

- ID penerapan, awalnya identik dengan ID penerapan versi dasar yang ditetapkan di panggung.
- Persentase lalu lintas API, antara 0,0 dan 100,0 inklusif, untuk rilis kenari.
- Variabel tahap untuk rilis kenari yang dapat mengganti variabel tahap rilis produksi.
- Penggunaan cache panggung untuk permintaan kenari, jika useStageCachedisetel dan caching API diaktifkan di atas panggung.

Setelah rilis kenari diaktifkan, tahap penerapan tidak dapat dikaitkan dengan penerapan rilis non-kenari lainnya hingga rilis kenari dinonaktifkan dan pengaturan kenari dihapus dari panggung.

Saat Anda mengaktifkan logging eksekusi API, rilis canary memiliki log dan metriknya sendiri yang dihasilkan untuk semua permintaan canary. Mereka dilaporkan ke grup CloudWatch log Log tahap produksi serta grup log CloudWatch Log khusus kenari. Hal yang sama berlaku untuk mengakses logging. Log khusus kenari yang terpisah sangat membantu untuk memvalidasi perubahan API baru

dan memutuskan apakah akan menerima perubahan dan mempromosikan rilis kenari ke tahap produksi, atau untuk membuang perubahan dan mengembalikan rilis kenari dari tahap produksi.

Grup log eksekusi tahap produksi diberi nama `API-Gateway-Execution-Logs/{rest-api-id}/{stage-name}` dan grup log eksekusi rilis kenari diberi nama`API-Gateway-Execution-Logs/{rest-api-id}/{stage-name}/Canary`. Untuk akses logging, Anda harus membuat grup log baru atau memilih yang sudah ada. Nama grup log akses rilis kenari memiliki `/Canary` akhiran yang ditambahkan ke nama grup log yang dipilih.

Rilis kenari dapat menggunakan cache panggung, jika diaktifkan, untuk menyimpan respons dan menggunakan entri yang di-cache untuk mengembalikan hasil ke permintaan kenari berikutnya, dalam periode pra-konfigurasi time-to-live (TTL).

Dalam penerapan rilis kenari, rilis produksi dan rilis canary API dapat dikaitkan dengan versi yang sama atau dengan versi yang berbeda. Ketika mereka dikaitkan dengan versi yang berbeda, respons untuk permintaan produksi dan kenari di-cache secara terpisah dan cache tahap mengembalikan hasil yang sesuai untuk permintaan produksi dan kenari. Ketika rilis produksi dan rilis kenari dikaitkan dengan penerapan yang sama, cache tahap menggunakan kunci cache tunggal untuk kedua jenis permintaan dan mengembalikan respons yang sama untuk permintaan yang sama dari rilis produksi dan rilis kenari.

Buat penerapan rilis kenari

Anda membuat penerapan rilis kenari saat menerapkan API dengan setelan canary sebagai input tambahan untuk operasi pembuatan penerapan.

Anda juga dapat membuat penerapan rilis kenari dari penerapan non-canary yang ada dengan membuat `stage:update`permintaan untuk menambahkan pengaturan kenari di panggung.

Saat membuat penyebaran rilis non-canary, Anda dapat menentukan nama panggung yang tidak ada. API Gateway membuat satu jika tahap yang ditentukan tidak ada. Namun, Anda tidak dapat menentukan nama panggung yang tidak ada saat membuat penerapan rilis kenari. Anda akan mendapatkan kesalahan dan API Gateway tidak akan membuat penerapan rilis kenari apa pun.

Anda dapat membuat penerapan rilis canary di API Gateway menggunakan konsol API Gateway, theAWS CLI, atau SDK. AWS

Topik

- Buat penerapan canary menggunakan konsol API Gateway
- Buat penerapan kenari menggunakan AWS CLI

Buat penerapan canary menggunakan konsol API Gateway

Untuk menggunakan konsol API Gateway untuk membuat penerapan rilis canary, ikuti petunjuk di bawah ini:

Untuk membuat penerapan rilis kenari awal

1. Masuk ke konsol API Gateway.
2. Pilih REST API yang ada atau buat REST API baru.
3. Di panel navigasi utama, pilih Resources, lalu pilih Deploy API. Ikuti petunjuk di layar di Deploy API untuk menerapkan API ke tahap baru.

   Sejauh ini, Anda telah menerapkan API ke tahap rilis produksi. Selanjutnya, Anda mengonfigurasi pengaturan kenari di panggung dan, jika perlu, juga mengaktifkan caching, mengatur variabel tahap, atau mengonfigurasi eksekusi API atau log akses.
4. Untuk mengaktifkan caching API atau mengaitkan ACL AWS WAF web dengan stage, di bagian Detail tahap, pilih Edit. Untuk informasi selengkapnya, lihat the section called “Pengaturan cache” atau the section called “Untuk mengaitkan ACL Web AWS WAF regional dengan tahap API Gateway API menggunakan konsol API Gateway”.
5. Untuk mengonfigurasi eksekusi atau mengakses logging, di bagian Log dan penelusuran, pilih Edit dan ikuti petunjuk di layar. Untuk informasi selengkapnya, lihat Menyiapkan CloudWatch logging untuk REST API di API Gateway.
6. Untuk mengatur variabel tahap, pilih tab variabel Tahap dan ikuti petunjuk di layar untuk menambah atau memodifikasi variabel tahap. Untuk informasi selengkapnya, lihat the section called “Mengatur variabel tahap”.
7. Pilih tab Canary, lalu pilih Create canary. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab Canary.
8. Di bawah pengaturan Canary, untuk Canary, masukkan persentase permintaan yang akan dialihkan ke kenari.
9. Jika diinginkan, pilih Cache tahap untuk mengaktifkan caching untuk rilis kenari. Cache tidak tersedia untuk rilis canary sampai caching API diaktifkan.
10. Untuk mengganti variabel tahap yang ada, untuk penggantian Canary, masukkan nilai variabel tahap baru.

Setelah rilis canary diinisialisasi pada tahap penerapan, Anda mengubah API dan ingin menguji perubahannya. Anda dapat menerapkan ulang API ke tahap yang sama sehingga versi yang

diperbarui dan versi dasar dapat diakses melalui tahap yang sama. Langkah-langkah berikut menjelaskan cara melakukannya.

Untuk menerapkan versi API terbaru ke kenari

1. Dengan setiap pembaruan API, pilih Deploy API.
2. Di Deploy API, pilih tahap yang berisi kenari dari daftar dropdown tahap Deployment.
3. (Opsional) Masukkan deskripsi untuk deskripsi Deployment.
4. Pilih Deploy untuk mendorong versi API terbaru ke rilis canary.
5. Jika diinginkan, konfigurasikan ulang pengaturan panggung, log, atau pengaturan kenari, seperti yang dijelaskan dalam. Untuk membuat penerapan rilis kenari awal

Akibatnya, rilis kenari menunjuk ke versi terbaru sementara rilis produksi masih menunjuk ke versi awal API. CanarySettings sekarang memiliki nilai deploymentID baru, sedangkan stage masih memiliki nilai DeployMentID awal. Di belakang layar, konsol memanggil stage:update.

Buat penerapan kenari menggunakan AWS CLI

Pertama buat penerapan dasar dengan dua variabel tahap, tetapi tanpa kenari apa pun:

```
aws apigateway create-deployment \
    --variables sv0=val0,sv1=val1 \
    --rest-api-id abcd1234 \
    --stage-name 'prod' \
```

Perintah mengembalikan representasi yang dihasilkan `Deployment`, mirip dengan yang berikut:

```
{
    "id": "du4ot1",
    "createdDate": 1511379050
}
```

Penerapan yang dihasilkan `id` mengidentifikasi snapshot (atau versi) API.

Sekarang buat penyebaran kenari di atas panggung: `prod`

```
aws apigateway create-deployment  --rest-api-id abcd1234 \
    --canary-settings \
    '{
```

```
        "percentTraffic":10.5,
        "useStageCache":false,
        "stageVariableOverrides":{
            "sv1":"val2",
            "sv2":"val3"
        }
    }' \
    --stage-name 'prod'
```

Jika tahap tertentu (`prod`) tidak ada, perintah sebelumnya mengembalikan kesalahan. Jika tidak, ia mengembalikan representasi sumber daya penerapan yang baru dibuat mirip dengan yang berikut ini:

```
{
    "id": "a6rox0",
    "createdDate": 1511379433
}
```

Penerapan yang dihasilkan `id` mengidentifikasi versi uji API untuk rilis canary. Akibatnya, tahap terkait diaktifkan kenari. Anda dapat melihat representasi tahap ini dengan memanggil `get-stage` perintah, mirip dengan yang berikut ini:

```
aws apigateway get-stage --rest-api-id acbd1234 --stage-name prod
```

Berikut ini menunjukkan representasi dari `Stage` sebagai output dari perintah:

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

Dalam contoh ini, versi dasar API akan menggunakan variabel tahap`{"sv0":val0", "sv1":val1"}`, sedangkan versi pengujian menggunakan variabel tahap`{"sv1":val2", "sv2":val3"}`. Baik rilis produksi dan rilis kenari menggunakan variabel tahap yang sama`sv1`, tetapi dengan nilai yang berbeda, `val1` dan`val2`, masing-masing. Variabel tahap `sv0` digunakan hanya dalam rilis produksi dan variabel tahap `sv2` digunakan hanya dalam pelepasan kenari.

Anda dapat membuat penerapan rilis kenari dari penerapan reguler yang ada dengan memperbarui tahapan untuk mengaktifkan kenari. Untuk mendemonstrasikannya, buat penerapan reguler terlebih dahulu:

```
aws apigateway create-deployment \
    --variables sv0=val0,sv1=val1 \
    --rest-api-id abcd1234 \
    --stage-name 'beta'
```

Perintah mengembalikan representasi penerapan versi dasar:

```
{
    "id": "cifeiw",
    "createdDate": 1511380879
}
```

Tahap beta terkait tidak memiliki pengaturan kenari:

```
{
    "stageName": "beta",
    "variables": {
        "sv0": "val0",
        "sv1": "val1"
    },
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": "cifeiw",
```

```
    "lastUpdatedDate": 1511380879,
    "createdDate": 1511380879,
    "methodSettings": {}
}
```

Sekarang, buat penyebaran rilis kenari baru dengan melampirkan kenari di atas panggung:

```
aws apigateway update-stage \
    --rest-api-id abcd1234 \
    --stage-name 'beta' \
    --patch-operations '[{
            "op": "replace",
            "value": "0.0",
            "path": "/canarySettings/percentTraffic"
        }, {
            "op": "copy",
            "from": "/canarySettings/stageVariableOverrides",
            "path": "/variables"
        }, {
            "op": "copy",
            "from": "/canarySettings/deploymentId",
            "path": "/deploymentId"
        }]'
```

Representasi dari tahap yang diperbarui terlihat seperti ini:

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

Karena kami baru saja mengaktifkan kenari pada versi API yang ada, baik production release (`Stage`) dan canary release (`canarySettings`) menunjuk ke penerapan yang sama, yaitu versi (`deploymentId`) API yang sama. Setelah Anda mengubah API dan menerapkannya ke tahap ini lagi, versi baru akan berada di rilis canary, sementara versi dasar tetap dalam rilis produksi. Ini dimanifestasikan dalam evolusi tahap ketika rilis `deploymentId` dalam kenari diperbarui ke penerapan baru `id` dan rilis produksi tetap tidak berubah. `deploymentId`

Perbarui rilis kenari

Setelah rilis kenari diterapkan, Anda mungkin ingin menyesuaikan persentase lalu lintas kenari atau mengaktifkan atau menonaktifkan penggunaan cache panggung untuk mengoptimalkan kinerja pengujian. Anda juga dapat memodifikasi variabel tahap yang digunakan dalam rilis kenari saat konteks eksekusi diperbarui. Untuk melakukan pembaruan seperti itu, panggil operasi stage:update dengan nilai baru di CanarySettings.

Anda dapat memperbarui rilis canary menggunakan konsol API Gateway, perintah AWS CLI tahap pembaruan, atau SDK. AWS

Topik

- Memperbarui rilis kenari menggunakan konsol API Gateway
- Perbarui rilis kenari menggunakan AWS CLI

Memperbarui rilis kenari menggunakan konsol API Gateway

Untuk menggunakan konsol API Gateway untuk memperbarui setelan canary yang ada di atas panggung, lakukan hal berikut:

Untuk memperbarui pengaturan kenari yang ada

1. Masuk ke konsol API Gateway dan pilih REST API yang ada.
2. Di panel navigasi utama, pilih Tahapan, lalu pilih tahap yang ada.
3. Pilih tab Canary, lalu pilih Edit. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab Canary.

4. Perbarui distribusi Permintaan dengan menambah atau mengurangi jumlah persentase antara 0,0 dan 100,0, inklusif.
5. Pilih atau hapus Stage cache kotak centang.
6. Menambahkan, menghapus, atau memodifikasi variabel tahap Canary.
7. Pilih Simpan.

Perbarui rilis kenari menggunakan AWS CLI

Untuk menggunakan AWS CLI untuk memperbarui kenari, panggil `update-stage`perintah.

Untuk mengaktifkan atau menonaktifkan penggunaan cache panggung untuk kenari, panggil `update-stage`perintah sebagai berikut:

```
aws apigateway update-stage \
    --rest-api-id {rest-api-id} \
    --stage-name '{stage-name}'  \
    --patch-operations op=replace,path=/canarySettings/useStageCache,value=true
```

Untuk menyesuaikan persentase lalu lintas kenari, panggil `update-stage` untuk mengganti `/canarySettings/percentTraffic` nilai di atas panggung.

```
aws apigateway update-stage \
    --rest-api-id {rest-api-id} \
    --stage-name '{stage-name}' \
    --patch-operations op=replace,path=/canarySettings/percentTraffic,value=25.0
```

Untuk memperbarui variabel tahap kenari, termasuk menambahkan, mengganti, atau menghapus variabel tahap kenari:

```
aws apigateway update-stage  \
    --rest-api-id {rest-api-id} \
    --stage-name '{stage-name}'  \
    --patch-operations '[{
        "op": "replace",
        "path": "/canarySettings/stageVariableOverrides/newVar",
        "value": "newVal"
      }, {
        "op": "replace",
        "path": "/canarySettings/stageVariableOverrides/var2",
        "value": "val4"
```

```
      }, {
        "op": "remove",
        "path": "/canarySettings/stageVariableOverrides/var1"
      }]'
```

Anda dapat memperbarui semua hal di atas dengan menggabungkan operasi menjadi satu `patch-operations` nilai:

```
aws apigateway update-stage  \
    --rest-api-id {rest-api-id} \
    --stage-name '{stage-name}' \
    --patch-operations '[{
        "op": "replace",
        "path": "/canarySettings/percentTraffic",
        "value": "20.0"
    }, {
        "op": "replace",
        "path": "/canarySettings/useStageCache",
        "value": "true"
    }, {
        "op": "remove",
        "path": "/canarySettings/stageVariableOverrides/var1"
    }, {
        "op": "replace",
        "path": "/canarySettings/stageVariableOverrides/newVar",
        "value": "newVal"
    }, {
        "op": "replace",
        "path": "/canarySettings/stageVariableOverrides/val2",
        "value": "val4"
      }]'
```

Promosikan pelepasan kenari

Untuk mempromosikan rilis kenari membuatnya tersedia dalam tahap produksi versi API yang sedang diuji. Operasi ini melibatkan tugas-tugas berikut:

- Setel ulang ID penyebaran panggung dengan pengaturan ID penerapan kenari. Ini memperbarui snapshot API panggung dengan snapshot kenari, menjadikan versi uji sebagai rilis produksi juga.
- Perbarui variabel tahap dengan variabel tahap kenari, jika ada. Ini memperbarui konteks eksekusi API panggung dengan konteks kenari. Tanpa pembaruan ini, versi API baru dapat menghasilkan

hasil yang tidak terduga jika versi pengujian menggunakan variabel tahap yang berbeda atau nilai berbeda dari variabel tahap yang ada.

- Atur persentase lalu lintas kenari menjadi 0,0%.

Mempromosikan pelepasan kenari tidak menonaktifkan kenari di atas panggung. Untuk menonaktifkan kenari, Anda harus menghapus pengaturan kenari di atas panggung.

Topik

- Promosikan rilis kenari menggunakan konsol API Gateway
- Promosikan pelepasan kenari menggunakan AWS CLI

Promosikan rilis kenari menggunakan konsol API Gateway

Untuk menggunakan konsol API Gateway untuk mempromosikan penerapan rilis canary, lakukan hal berikut:

Untuk mempromosikan penyebaran rilis kenari

1. Masuk ke konsol API Gateway dan pilih API yang ada di panel navigasi utama.
2. Di panel navigasi utama, pilih Tahapan, lalu pilih tahap yang ada.
3. Pilih tab Canary.
4. Pilih Promosikan kenari.
5. Konfirmasikan perubahan yang akan dilakukan dan pilih Promosikan kenari.

Setelah promosi, rilis produksi mereferensikan versi API yang sama (deploymentID) dengan rilis canary. Anda dapat memverifikasi ini menggunakanAWS CLI. Sebagai contoh, lihat the section called “Promosikan pelepasan kenari menggunakan AWS CLI”.

Promosikan pelepasan kenari menggunakan AWS CLI

Untuk mempromosikan rilis kenari ke rilis produksi menggunakan AWS CLI perintah, panggil `update-stage` perintah untuk menyalin kenari yang terkait dengan tahap terkait, `deploymentId` untuk mengatur ulang persentase lalu lintas kenari ke nol (`0.0`)`deploymentId`, dan, untuk menyalin variabel tahap terikat kenari ke yang terikat tahap yang sesuai.

Misalkan kita memiliki penyebaran rilis kenari, dijelaskan oleh tahap yang mirip dengan yang berikut ini:

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

Kami menyebut `update-stage` permintaan berikut untuk mempromosikannya:

```
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"
      }]'
```

Setelah promosi, panggung sekarang terlihat seperti ini:

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

Seperti yang Anda lihat, mempromosikan rilis kenari ke panggung tidak menonaktifkan kenari dan penerapan tetap menjadi penerapan rilis kenari. Untuk menjadikannya penyebaran rilis produksi

reguler, Anda harus menonaktifkan pengaturan kenari. Untuk informasi selengkapnya tentang cara menonaktifkan penerapan rilis kenari, lihat. the section called “Matikan pelepasan kenari”

Matikan pelepasan kenari

Untuk mematikan penerapan rilis kenari berarti menyetel ke null `canarySettings`untuk menghapusnya dari panggung.

Anda dapat menonaktifkan penerapan rilis canary menggunakan konsol API Gateway, theAWS CLI, atau SDK. AWS

Topik

- Matikan rilis kenari menggunakan konsol API Gateway
- Matikan pelepasan kenari menggunakan AWS CLI

Matikan rilis kenari menggunakan konsol API Gateway

Untuk menggunakan konsol API Gateway untuk menonaktifkan penerapan rilis canary, gunakan langkah-langkah berikut:

Untuk mematikan penerapan rilis kenari

1. Masuk ke konsol API Gateway dan pilih API yang ada di panel navigasi utama.
2. Di panel navigasi utama, pilih Tahapan, lalu pilih tahap yang ada.
3. Pilih tab Canary.
4. Pilih Hapus.
5. Konfirmasikan bahwa Anda ingin menghapus kenari dengan memilih Hapus.

Akibatnya, `canarySettings`properti menjadi `null` dan dihapus dari tahap penyebaran. Anda dapat memverifikasi ini menggunakanAWS CLI. Sebagai contoh, lihat the section called “Matikan pelepasan kenari menggunakan AWS CLI”.

Matikan pelepasan kenari menggunakan AWS CLI

Untuk menggunakan AWS CLI untuk mematikan penyebaran rilis kenari, panggil `update-stage` perintah sebagai berikut:

```
aws apigateway update-stage \
    --rest-api-id abcd1234 \
```

```
    --stage-name canary \
    --patch-operations '[{"op":"remove", "path":"/canarySettings"}]'
```

Respons yang berhasil mengembalikan payload yang mirip dengan berikut ini:

```
{
    "stageName": "prod",
    "accessLogSettings": {
        ...
    },
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": "nfcn0x",
    "lastUpdatedDate": 1511309280,
    "createdDate": 1511152939,
    "methodSettings": {
        ...
    }
}
```

Seperti yang ditunjukkan dalam output, `canarySettings`properti tidak lagi ada dalam tahap penyebaran yang dinonaktifkan kenari.

Pembaruan ke REST API yang memerlukan penyebaran ulang

Mempertahankan jumlah API untuk melihat, memperbarui, dan menghapus pengaturan API yang ada. Anda dapat mempertahankan API menggunakan konsol API Gateway,AWS CLI, SDK atau REST API Gateway. Memperbarui API melibatkan memodifikasi properti sumber daya tertentu atau pengaturan konfigurasi API. Pembaruan sumber daya memerlukan penggelaran ulang API, sedangkan pembaruan konfigurasi tidak.

Sumber daya API yang dapat diperbarui dijelaskan dalam tabel berikut.

Pembaruan sumber daya API yang memerlukan penyebaran ulang API

Resource	Keterangan
ApiKey	Untuk properti yang berlaku dan operasi yang didukung, lihatapikey: perbarui. Pembaruan memerlukan penggelaran ulang API.
Otorisasi	Untuk properti yang berlaku dan operasi yang didukung, lihatotorisasi: update. Pembaruan memerlukan penggelaran ulang API.

Resource	Keterangan
DocumentationPart	Untuk properti yang berlaku dan operasi yang didukung, lihatdocumentationpart: update. Pembaruan memerlukan penggelaran ulang API.
DocumentationVersion	Untuk properti yang berlaku dan operasi yang didukung, lihatdocumentationversion:update. Pembaruan memerlukan penggelaran ulang API.
GatewayResponse	Untuk properti yang berlaku dan operasi yang didukung, lihatgateway respons: pembaruan. Pembaruan memerlukan penggelaran ulang API.
Integrasi	Untuk properti yang berlaku dan operasi yang didukung, lihatintegrasi: update. Pembaruan memerlukan penggelaran ulang API.
IntegrationResponse	Untuk properti yang berlaku dan operasi yang didukung, lihatintegrasirespons: pembaruan. Pembaruan memerlukan penggelaran ulang API.
Metode	Untuk properti yang berlaku dan operasi yang didukung, lihatmetode: update. Pembaruan memerlukan penggelaran ulang API.
MethodResponse	Untuk properti yang berlaku dan operasi yang didukung, lihatmetodesponse: memperbarui. Pembaruan memerlukan penggelaran ulang API.
Model	Untuk properti yang berlaku dan operasi yang didukung, lihatModel: update. Pembaruan memerlukan penggelaran ulang API.
RequestValidator	Untuk properti yang berlaku dan operasi yang didukung, lihatrequestvalidator:update. Pembaruan memerlukan penggelaran ulang API.
Sumber Daya	Untuk properti yang berlaku dan operasi yang didukung, lihatsumber daya: update. Pembaruan memerlukan penggelaran ulang API.
RestApi	Untuk properti yang berlaku dan operasi yang didukung, lihatrestapi:perbarui. Pembaruan memerlukan penggelaran ulang API.
VpcLink	Untuk properti yang berlaku dan operasi yang didukung, lihatvpclink: perbarui. Pembaruan memerlukan penggelaran ulang API.

Konfigurasi API yang dapat diperbarui dijelaskan dalam tabel berikut.

Pembaruan konfigurasi API tanpa memerlukan pengalihan ulang API

Konfigurasi	Keterangan
Akun	Untuk properti yang berlaku dan operasi yang didukung, lihatakun: update. Pembaruan tidak memerlukan penggelaran ulang API.
Penerapan	Untuk properti yang berlaku dan operasi yang didukung, lihatpenyebaran: update.
DomainName	Untuk properti yang berlaku dan operasi yang didukung, lihatnama domain: update. Pembaruan tidak memerlukan penggelaran ulang API.
BasePathMapping	Untuk properti yang berlaku dan operasi yang didukung, lihatbasepathmapping: perbarui. Pembaruan tidak memerlukan penggelaran ulang API.
Tahap	Untuk properti yang berlaku dan operasi yang didukung, lihattahap: update. Pembaruan tidak memerlukan penggelaran ulang API.
Penggunaan	Untuk properti yang berlaku dan operasi yang didukung, lihatpenggunaan: update. Pembaruan tidak memerlukan penggelaran ulang API.
UsagePlan	Untuk properti yang berlaku dan operasi yang didukung, lihatUsageplan: update. Pembaruan tidak memerlukan penggelaran ulang API.

Menyiapkan nama domain kustom untuk REST API

Nama domain khusus adalah URL yang lebih sederhana dan lebih intuitif yang dapat Anda berikan kepada pengguna API Anda.

Setelah menerapkan API Anda, Anda (dan pelanggan Anda) dapat memanggil API menggunakan URL dasar default dari format berikut:

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

dimanaapi-id dihasilkan oleh API Gateway,region (AWSRegion) ditentukan oleh Anda saat membuat API, danstage ditentukan oleh Anda saat menerapkan API.

Bagian hostname dari URL (yaitu,*api-id*.execute-api.*region*.amazonaws.com) mengacu pada endpoint API. Endpoint API default bisa sulit untuk diingat dan tidak ramah pengguna.

Dengan nama domain kustom, Anda dapat mengatur nama host API Anda, dan memilih jalur dasar (misalnya,`myservice`) untuk memetakan URL alternatif ke API Anda. Misalnya, URL dasar API yang lebih ramah pengguna dapat menjadi:

```
https://api.example.com/myservice
```

> **Note**
>
> Domain kustom Regional dapat dikaitkan dengan REST API dan API HTTP. Anda dapat menggunakan API API Gateway Versi 2 untuk membuat dan mengelola nama domain kustom Regional untuk REST API.
> Nama domain kustom tidak didukung untuk API pribadi.
> Anda dapat memilih versi TLS minimum yang didukung REST API Anda. Untuk API REST, Anda dapat memilih TLS 1.2 atau TLS 1.0.

Mendaftarkan nama domain

Anda harus memiliki nama domain internet terdaftar untuk mengatur nama domain kustom untuk API Anda. Jika diperlukan, Anda dapat mendaftarkan domain internet menggunakan Amazon Route 53 atau menggunakan registrar domain pihak ketiga pilihan Anda. Nama domain kustom API dapat berupa nama subdomain atau domain root (juga dikenal sebagai “zona apex”) dari domain internet terdaftar.

Setelah nama domain kustom dibuat di API Gateway, Anda harus membuat atau memperbarui data sumber daya penyedia DNS Anda untuk memetakan ke titik akhir API Anda. Tanpa pemetaan seperti itu, permintaan API yang terikat untuk nama domain kustom tidak dapat mencapai API Gateway.

> **Note**
>
> Nama domain kustom yang dioptimalkan tepi dibuat di Wilayah tertentu dan dimiliki olehAWS akun tertentu. Memindahkan nama domain khusus antara Wilayah atauAWS akun melibatkan penghapusanCloudFront distribusi yang ada dan membuat yang baru. Prosesnya mungkin memakan waktu sekitar 30 menit sebelum nama domain kustom baru tersedia. Untuk informasi lebih lanjut, lihat MemperbaruiCloudFront Distribusi.

Nama domain kustom yang dioptimalkan tepi

Saat Anda menerapkan API yang dioptimalkan tepi, API Gateway menyiapkanCloudFront distribusi Amazon dan catatan DNS untuk memetakan nama domain API ke nama domainCloudFront distribusi. Permintaan API kemudian dialihkan ke API Gateway melaluiCloudFront distribusi yang dipetakan.

Saat Anda membuat nama domain khusus untuk API yang dioptimalkan tepi, API Gateway menyiapkanCloudFront distribusi. Tetapi Anda harus menyiapkan data DNS untuk memetakan nama domain kustom ke nama domainCloudFront distribusi. Pemetaan ini untuk permintaan API yang terikat untuk nama domain kustom yang akan dialihkan ke API Gateway melaluiCloudFront distribusi yang dipetakan. Anda juga harus memberikan sertifikat untuk nama domain kustom.

> **Note**
>
> CloudFrontDistribusi yang dibuat oleh API Gateway dimiliki oleh akun khusus Wilayah yang berafiliasi dengan API Gateway. Saat melacak operasi untuk membuat dan memperbaruiCloudFront distribusi tersebut diCloudWatch Log, Anda harus menggunakan ID akun API Gateway ini. Untuk informasi selengkapnya, lihat Log pembuatan nama domain kustomCloudTrail.

Untuk mengatur nama domain kustom yang dioptimalkan atau untuk memperbarui sertifikatnya, Anda harus memiliki izin untuk memperbaruiCloudFront distribusi.

Untuk menyediakan akses, tambahkan izin ke pengguna, grup, atau peran Anda:

- Pengguna dan grup diAWS IAM Identity Center:

  Buat set izin. Ikuti petunjuk di Buat set izin di PanduanAWS IAM Identity Center Pengguna.
- Pengguna yang dikelola dalam IAM melalui penyedia identitas:

  Buat peran untuk federasi identitas. Ikuti petunjuk dalam Membuat peran untuk penyedia identitas pihak ketiga (federasi) di Panduan Pengguna IAM.
- Pengguna IAM:
  - Buat peran yang dapat diasumsikan pengguna Anda. Ikuti petunjuk dalam Membuat peran untuk pengguna IAM di Panduan Pengguna IAM.

- (Tidak disarankan) Lampirkan kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti petunjuk dalam Menambahkan izin ke pengguna (konsol) di Panduan Pengguna IAM.

Izin berikut diperlukan untuk memperbaruiCloudFront distribusi.

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Sid": "AllowCloudFrontUpdateDistribution",
            "Effect": "Allow",
            "Action": [
                "cloudfront:updateDistribution"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

API Gateway mendukung nama domain kustom yang dioptimalkan tepi dengan memanfaatkan Server Name Indication (SNI) padaCloudFront distribusi. Untuk informasi selengkapnya tentang penggunaan nama domain kustom padaCloudFront distribusi, termasuk format sertifikat yang diperlukan dan ukuran maksimum panjang kunci sertifikat, lihat Menggunakan Nama Domain Alternatif dan HTTPS di PanduanCloudFront Pengembang Amazon.

Untuk menyiapkan nama domain kustom sebagai nama host API, Anda, sebagai pemilik API, harus memberikan sertifikat SSL/TLS untuk nama domain kustom.

Untuk memberikan sertifikat untuk nama domain kustom yang dioptimalkan tepi, Anda dapat meminta AWS Certificate Manager(ACM) untuk membuat sertifikat baru di ACM atau mengimpor ke ACM yang dikeluarkan oleh otoritas sertifikat pihak ketiga di`us-east-1` Wilayah (AS Timur (Virginia Utara)).

Nama domain kustom regional

Saat Anda membuat nama domain khusus untuk API Regional, API Gateway membuat nama domain Regional untuk API. Anda harus menyiapkan data DNS untuk memetakan nama domain kustom ke nama domain Regional. Anda juga harus memberikan sertifikat untuk nama domain kustom.

Nama domain kustom wildcard

Dengan nama domain kustom wildcard, Anda dapat mendukung jumlah nama domain yang hampir tak terbatas tanpa melebihi kuota default. Misalnya, Anda bisa memberi masing-masing pelanggan nama domain mereka sendiri,*customername*.api.example.com.

Untuk membuat nama domain kustom wildcard, tentukan wildcard (*) sebagai subdomain pertama dari domain kustom yang mewakili semua kemungkinan subdomain dari domain root.

Misalnya, nama domain kustom wildcard*.example.com menghasilkan subdomain sepertia.example.com,, danb.example.comc.example.com, yang semuanya dirutekan ke domain yang sama.

Nama domain kustom wildcard mendukung konfigurasi berbeda dari nama domain kustom standar API Gateway. Misalnya, dalam satuAWS akun, Anda dapat mengkonfigurasi*.example.com dana.example.com berperilaku berbeda.

Anda dapat menggunakan variabel$context.domainName dan$context.domainPrefix konteks untuk menentukan nama domain yang digunakan klien untuk memanggil API Anda. Untuk mempelajari tentang variabel konteks, lihatTemplate pemetaan API Gateway dan referensi variabel pencatatan akses.

Untuk membuat nama domain kustom wildcard, Anda harus memberikan sertifikat yang dikeluarkan oleh ACM yang telah divalidasi menggunakan DNS atau metode validasi email.

> **Note**
>
> Anda tidak dapat membuat nama domain kustom wildcard jikaAWS akun lain telah membuat nama domain kustom yang bertentangan dengan nama domain kustom wildcard. Misalnya, jika akun A telah dibuata.example.com, maka akun B tidak dapat membuat nama domain kustom wildcard*.example.com.
> Jika akun A dan akun B berbagi pemilik, Anda dapat menghubungi PusatAWS Support untuk meminta pengecualian.

Sertifikat untuk nama domain kustom

> **Important**
>
> Anda menentukan sertifikat untuk nama domain kustom Anda. Jika aplikasi Anda menggunakan penyematan sertifikat, yang terkadang dikenal sebagai penyematan SSL, untuk menyematkan sertifikat ACM, aplikasi mungkin tidak dapat terhubung ke domain Anda setelahAWS memperbarui sertifikat. Untuk informasi selengkapnya, lihat Masalah penyematan sertifikat di PanduanAWS Certificate Manager Pengguna.

Untuk memberikan sertifikat untuk nama domain kustom di Wilayah tempat ACM didukung, Anda harus meminta sertifikat dari ACM. Untuk memberikan sertifikat untuk nama domain kustom Regional di Wilayah yang tidak didukung ACM, Anda harus mengimpor sertifikat ke API Gateway di Wilayah tersebut.

Untuk mengimpor sertifikat SSL/TLS, Anda harus menyediakan badan sertifikat SSL/TLS berformat PEM, kunci pribadinya, dan rantai sertifikat untuk nama domain kustom. Setiap sertifikat yang disimpan dalam ACM diidentifikasi oleh ARN-nya. Untuk menggunakan sertifikatAWS terkelola untuk nama domain, Anda cukup mereferensikan ARN-nya.

ACM membuatnya mudah untuk mengatur dan menggunakan nama domain kustom untuk API. Anda membuat sertifikat untuk nama domain yang diberikan (atau mengimpor sertifikat), menyiapkan nama domain di API Gateway dengan ARN sertifikat yang disediakan oleh ACM, dan memetakan jalur dasar di bawah nama domain kustom ke tahap API yang diterapkan. Dengan sertifikat yang dikeluarkan oleh ACM, Anda tidak perlu khawatir tentang mengekspos rincian sertifikat sensitif, seperti kunci pribadi.

Topik

- Mendapatkan sertifikat siap diAWS Certificate Manager
- Memilih kebijakan keamanan untuk domain kustom Anda di API Gateway
- Membuat nama domain kustom yang dioptimalkan tepi
- Menyiapkan nama domain kustom regional di API Gateway
- Migrasi nama domain kustom ke endpoint API yang berbeda
- Bekerja dengan pemetaan API untuk REST API
- Menonaktifkan titik akhir default untuk REST API

- Konfigurasikan pemeriksaan kesehatan khusus untuk failover DNS

Mendapatkan sertifikat siap diAWS Certificate Manager

Sebelum menyiapkan nama domain kustom untuk API, Anda harus memiliki sertifikat SSL/TLS yang siapAWS Certificate Manager. Langkah-langkah berikut menjelaskan cara menyelesaikannya. Untuk informasi selengkapnya, lihat Panduan Pengguna AWS Certificate Manager.

> **Note**
>
> Untuk menggunakan sertifikat ACM dengan nama domain kustom yang dioptimalkan tepi API Gateway, Anda harus meminta atau mengimpor sertifikat di Wilayah AS Timur (Virginia Utara) (`us-east-1`). Untuk nama domain kustom API Gateway Regional, Anda harus meminta atau mengimpor sertifikat di Wilayah yang sama dengan API Anda. Sertifikat harus ditandatangani oleh Otoritas Sertifikat yang dipercaya publik dan mencakup nama domain kustom.

Pertama, daftarkan domain internet Anda, misalnya,*`example`*`.com`. Anda dapat menggunakan Amazon Route 53 atau pencatat domain terakreditasi pihak ketiga. Untuk daftar pencatat tersebut, lihat Direktori Pencatat Terakreditasi di situs web ICANN.

Untuk membuat atau mengimpor sertifikat SSL/TLS ke ACM untuk nama domain, lakukan salah satu hal berikut:

Untuk meminta sertifikat yang disediakan oleh ACM untuk nama domain

1. Masuk ke konsol AWS Certificate Manager tersebut.
2. Pilih Minta sertifikat.
3. Masukkan nama domain kustom untuk API Anda, misalnya`api.example.com`, dalam Nama Domain.
4. Secara opsional, pilih Tambahkan nama lain ke sertifikat ini.
5. Pilih Tinjau dan minta.
6. Pilih Konfirmasi dan minta.
7. Untuk permintaan yang valid, pemilik domain internet yang terdaftar harus menyetujui permintaan tersebut sebelum ACM mengeluarkan sertifikat.

Untuk mengimpor ke ACM sertifikat untuk nama domain

1. Dapatkan sertifikat SSL/TLS yang dikodekan PEM untuk nama domain kustom Anda dari otoritas sertifikat. Untuk daftar sebagian CA tersebut, lihat Daftar CA Termasuk Mozilla

   a. Buat kunci pribadi untuk sertifikat dan simpan output ke file, menggunakan toolkit OpenSSL di situs web OpenSSL:

   ```
   openssl genrsa -out private-key-file 2048
   ```

   > **Note**
   >
   > Amazon API Gateway memanfaatkan AmazonCloudFront untuk mendukung sertifikat untuk nama domain khusus. Dengan demikian, persyaratan dan batasan sertifikat SSL/TLS nama domain kustom ditentukan oleh CloudFront. Misalnya, ukuran maksimum kunci publik adalah 2048 dan ukuran kunci pribadi dapat 1024, 2048, dan 4096. Ukuran kunci publik ditentukan oleh otoritas sertifikat yang Anda gunakan. Minta otoritas sertifikat Anda untuk mengembalikan kunci dengan ukuran yang berbeda dari panjang default. Untuk informasi selengkapnya, lihat Mengamankan akses ke objek Anda dan Membuat URL yang ditandatangani dan cookie yang ditandatangani.

   b. Membuat permintaan penandatanganan sertifikat (CSR) dengan kunci privat yang dibuat sebelumnya, menggunakan OpenSSL:

   ```
   openssl req -new -sha256 -key private-key-file -out CSR-file
   ```

   c. Kirimkan CSR ke otoritas sertifikat dan simpan sertifikat yang dihasilkan.

   d. Unduh rantai sertifikat dari otoritas sertifikat.

   > **Note**
   >
   > Jika Anda mendapatkan kunci pribadi dengan cara lain dan kunci dienkripsi, Anda dapat menggunakan perintah berikut untuk mendekripsi kunci sebelum mengirimkannya ke API Gateway untuk menyiapkan nama domain kustom.

```
openssl pkcs8 -topk8 -inform pem -in MyEncryptedKey.pem -outform pem -
nocrypt -out MyDecryptedKey.pem
```

2. Unggah sertifikat keAWS Certificate Manager:

   a. Masuk ke konsol AWS Certificate Manager tersebut.

   b. Pilih Impor sertifikat.

   c. Untuk badan Sertifikat, masukkan atau tempel badan sertifikat server berformat PEM dari otoritas sertifikat Anda. Berikut ini adalah contoh singkat dari sertifikat semacam itu.

```
-----BEGIN CERTIFICATE-----
EXAMPLECA+KgAwIBAgIQJ1XxJ8Pl++gOfQtj0IBoqDANBgkqhkiG9w0BAQUFADBB
...
az8Cg1aicxLBQ7EaWIhhgEXAMPLE
-----END CERTIFICATE-----
```

   d. Untuk kunci pribadi Sertifikat, masukkan atau tempel kunci pribadi sertifikat yang diformat PEM. Berikut ini adalah contoh singkat dari kunci tersebut.

```
-----BEGIN RSA PRIVATE KEY-----
EXAMPLEBAAKCAQEA2Qb3LDHD7StY7Wj6U2/opV6Xu37qUCCkeDWhwpZMYJ9/nETO
...
1qGvJ3u04vdnzaYN5WoyN5LFckrlA71+CszD1CGSqbVDWEXAMPLE
-----END RSA PRIVATE KEY-----
```

   e. Untuk rantai Sertifikat, masukkan atau tempel sertifikat perantara berformat PEM dan, secara opsional, sertifikat root, satu demi satu tanpa baris kosong. Jika Anda menyertakan sertifikat root, rantai sertifikat Anda harus dimulai dengan sertifikat perantara dan diakhiri dengan sertifikat root. Gunakan sertifikat perantara yang disediakan oleh otoritas sertifikat Anda. Jangan sertakan perantara yang tidak berada dalam rantai jalur kepercayaan. Berikut ini adalah contoh singkat.

```
-----BEGIN CERTIFICATE-----
EXAMPLECA4ugAwIBAgIQWrYdrB5NogYUx1U9Pamy3DANBgkqhkiG9w0BAQUFADCB
...
8/ifBlIK3se2e4/hEfcEejX/arxbx1BJCHBvlEPNnsdw8EXAMPLE
-----END CERTIFICATE-----
```

Inilah contoh lainnya.

```
-----BEGIN CERTIFICATE-----
Intermediate certificate 2
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Intermediate certificate 1
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Optional: Root certificate
-----END CERTIFICATE-----
```

f. Pilih Tinjau dan impor.

Setelah sertifikat berhasil dibuat atau diimpor, catat sertifikat ARN. Anda membutuhkannya saat menyiapkan nama domain khusus.

Memilih kebijakan keamanan untuk domain kustom Anda di API Gateway

Untuk keamanan domain kustom Amazon API Gateway yang lebih baik, Anda dapat memilih kebijakan keamanan di konsol API Gateway AWS CLI, atau AWS SDK.

Kebijakan keamanan adalah kombinasi standar dari versi TLS minimum dan cipher suite yang ditawarkan oleh API Gateway. Anda dapat memilih kebijakan keamanan TLS versi 1.2 atau TLS versi 1.0. Protokol TLS mengatasi masalah keamanan jaringan seperti gangguan dan penyadapan antara klien dan server. Ketika klien Anda membuat jabat tangan TLS ke API Anda melalui domain kustom, kebijakan keamanan memberlakukan versi TLS dan pilihan cipher suite yang dapat dipilih klien Anda untuk digunakan.

Dalam pengaturan domain kustom, kebijakan keamanan menentukan dua pengaturan:

- Versi TLS minimum yang digunakan API Gateway untuk berkomunikasi dengan klien API
- Cipher yang digunakan API Gateway untuk mengenkripsi konten yang dikembalikan ke klien API

Jika Anda memilih kebijakan keamanan TLS 1.0, kebijakan keamanan menerima lalu lintas TLS 1.0, TLS 1.2, dan TLS 1.3. Jika Anda memilih kebijakan keamanan TLS 1.2, kebijakan keamanan menerima lalu lintas TLS 1.2 dan TLS 1.3 dan menolak lalu lintas TLS 1.0.

> **Note**
>
> Anda hanya dapat menentukan kebijakan keamanan untuk domain kustom. Untuk API yang menggunakan titik akhir default, API Gateway menggunakan kebijakan keamanan berikut:
>
> - Untuk API yang dioptimalkan tepi: `TLS-1-0`
> - Untuk API Regional: `TLS-1-0`
> - Untuk API pribadi: `TLS-1-2`

Topik

Cara menentukan kebijakan keamanan untuk domain kustom

Saat Anda membuat nama domain kustom, Anda menentukan kebijakan keamanan untuk itu. Untuk mempelajari cara membuat domain kustom, lihat the section called “Membuat nama domain kustom yang dioptimalkan tepi” atauthe section called “Menyiapkan nama domain kustom regional”.

Untuk mengubah kebijakan keamanan nama domain kustom Anda, perbarui pengaturan domain kustom. Anda dapat memperbarui setelan nama domain kustom menggunakan AWS CLI,, atau AWS SDK. AWS Management Console

Bila Anda menggunakan API Gateway REST API atau AWS CLI, tentukan versi TLS baru, `TLS_1_0` atau `TLS_1_2` dalam `securityPolicy` parameter. Untuk informasi selengkapnya, lihat domainname:update di Referensi API REST Amazon API Gateway atau di Referensi. update-domain-nameAWS CLI

Operasi pembaruan mungkin memakan waktu beberapa menit untuk diselesaikan.

Kebijakan keamanan yang didukung, versi protokol TLS, dan cipher untuk domain kustom yang dioptimalkan tepi

Tabel berikut menjelaskan kebijakan keamanan yang dapat ditentukan untuk nama domain kustom yang dioptimalkan tepi.

Kebijakan keamanan	TLS_1_0	TLS_1_2
Protokol TLS		
TLSv1.3	♦	♦
TLSv1.2	♦	♦
TLSv1.1	♦	
TLSv1	♦	
Cipher TLS		
TLS_AES_128_GCM_SHA256	♦	♦
TLS_AES_256_GCM_SHA384	♦	♦
TLS_CHACHA20_POLY1305_SHA256	♦	♦
ECDHE-ECDSA-AES128-GCM-SHA256	♦	♦
ECDHE-ECDSA-AES128-SHA256	♦	♦
ECDHE-ECDSA-AES128-SHA	♦	
ECDHE-ECDSA-AES256-GCM-SHA384	♦	♦
ECDHE-ECDSA-CHACHA20-POLY1305	♦	♦

Kebijakan keamanan	TLS_1_0	TLS_1_2
ECDHE-ECDSA-AES256-SHA384	♦	♦
ECDHE-ECDSA-AES256-SHA	♦	
ECDHE-RSA-AES128-GCM-SHA256	♦	♦
ECDHE-RSA-AES128-SHA256	♦	♦
ECDHE-RSA-AES128-SHA	♦	
ECDHE-RSA-AES256-GCM-SHA384	♦	♦
ECDHE-RSA-CHACHA20-POLY1305	♦	♦
ECDHE-RSA-AES256-SHA384	♦	♦
ECDHE-RSA-AES256-SHA	♦	
AES128-GCM-SHA256	♦	
AES256-GCM-SHA384	♦	♦
AES128-SHA256	♦	♦
AES256-SHA	♦	
AES128-SHA	♦	
DES-CBC3-SHA	♦	

Kebijakan keamanan yang didukung, versi protokol TLS, dan cipher untuk domain kustom Regional

Tabel berikut menjelaskan kebijakan keamanan yang dapat ditentukan untuk nama domain kustom Regional.

Kebijakan keamanan	TLS_1_0	TLS_1_2
Protokol TLS		
TLSv1.3	♦	♦
TLSV1.2	♦	♦
TLSV1.1	♦	
TLSv1	♦	
Cipher TLS		
TLS_AES_128_GCM_SHA256	♦	♦
TLS_AES_256_GCM_SHA384	♦	♦
TLS_CHACHA20_POLY1305_SHA256	♦	♦
ECDHE-ECDSA-AES128-GCM-SHA256	♦	♦
ECDHE-RSA-AES128-GCM-SHA256	♦	♦
ECDHE-ECDSA-AES128-SHA256	♦	♦
ECDHE-RSA-AES128-SHA256	♦	♦
ECDHE-ECDSA-AES128-SHA	♦	
ECDHE-RSA-AES128-SHA	♦	

Kebijakan keamanan	TLS_1_0	TLS_1_2
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	♦	

Versi protokol TLS yang didukung dan cipher untuk API pribadi

Tabel berikut menjelaskan protokol TLS yang didukung dan cipher untuk API pribadi. Menentukan kebijakan keamanan untuk API pribadi tidak didukung.

Kebijakan keamanan	TLS_1_2
Protokol TLS	
TLSv1.2	♦

Kebijakan keamanan	TLS_1_2
Cipher TLS	
ECDHE-ECDSA-AES128-GCM-SHA256	♦
ECDHE-RSA-AES128-GCM-SHA256	♦
ECDHE-ECDSA-AES128-SHA256	♦
ECDHE-RSA-AES128-SHA256	♦
ECDHE-ECDSA-AES256-GCM-SHA384	♦
ECDHE-RSA-AES256-GCM-SHA384	♦
ECDHE-ECDSA-AES256-SHA384	♦
ECDHE-RSA-AES256-SHA384	♦
AES128-GCM-SHA256	♦
AES128-SHA256	♦
AES256-GCM-SHA384	♦
AES256-SHA256	♦

Nama sandi OpenSSL dan RFC

OpenSSL dan IETF RFC 5246 menggunakan nama yang berbeda untuk cipher yang sama. Tabel berikut memetakan nama OpenSSL ke nama RFC untuk setiap cipher.

Nama sandi OpenSSL	Nama cipher RFC
TLS_AES_128_GCM_SHA256	TLS_AES_128_GCM_SHA256

Nama sandi OpenSSL	Nama cipher RFC
TLS_AES_256_GCM_SHA384	TLS_AES_256_GCM_SHA384
TLS_CHACHA20_POLY1305_SHA256	TLS_CHACHA20_POLY1305_SHA256
ECDHE-RSA-AES128-GCM-SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
ECDHE-RSA-AES128-SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
ECDHE-RSA-AES128-SHA	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
ECDHE-RSA-AES256-GCM-SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
ECDHE-RSA-AES256-SHA384	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
ECDHE-RSA-AES256-SHA	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
AES128-GCM-SHA256	TLS_RSA_WITH_AES_128_GCM_SHA256
AES256-GCM-SHA384	TLS_RSA_WITH_AES_256_GCM_SHA384
AES128-SHA256	TLS_RSA_WITH_AES_128_CBC_SHA256

Nama sandi OpenSSL	Nama cipher RFC
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

Informasi tentang HTTP API dan WebSocket API

Untuk informasi selengkapnya tentang API dan WebSocket API HTTP, lihat the section called "Kebijakan keamanan untuk HTTP API" danthe section called "Kebijakan keamanan untuk WebSocket API".

Membuat nama domain kustom yang dioptimalkan tepi

Topik

- Menyiapkan nama domain kustom yang dioptimalkan tepi untuk API Gateway API
- Log pembuatan nama domain kustomCloudTrail
- Konfigurasikan pemetaan jalur dasar API dengan nama domain khusus sebagai nama hostnya
- Memutar sertifikat yang diimpor ke ACM
- Panggil API Anda dengan nama domain khusus

Menyiapkan nama domain kustom yang dioptimalkan tepi untuk API Gateway API

Prosedur berikut menjelaskan cara membuat nama domain kustom untuk API menggunakan konsol API Gateway.

Untuk membuat nama domain kustom menggunakan konsol API Gateway

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih Nama Domain kustom dari panel navigasi utama.
3. Pilih Create (Buat).
4. Untuk Nama Domain, masukkan nama domain.
5. Di bawah Konfigurasi, pilih Ujung-dioptimalkan.

6. Pilih versi TLS minimum.
7. Pilih sertifikat ACM.

> **Note**
>
> Untuk menggunakan sertifikat ACM dengan nama domain kustom yang dioptimalkan tepi API Gateway, Anda harus meminta atau mengimpor sertifikat di`us-east-1` Wilayah (AS Timur (Virginia Utara)).

8. Pilih Buat nama domain.
9. Setelah nama domain kustom dibuat, konsol menampilkan nama domainCloudFront distribusi terkait, dalam bentuk*`distribution-id`*`.cloudfront.net`, bersama dengan ARN sertifikat. Perhatikan nama domainCloudFront distribusi yang ditunjukkan pada output. Anda memerlukannya di langkah berikutnya untuk menetapkan nilai CNAME domain kustom atau target alias A-record di DNS Anda.

> **Note**
>
> Nama domain kustom yang baru dibuat membutuhkan waktu sekitar 40 menit untuk siap. Sementara itu, Anda dapat mengkonfigurasi alias data DNS untuk memetakan nama domain kustom ke nama domainCloudFront distribusi terkait dan untuk mengatur pemetaan jalur dasar untuk nama domain kustom saat nama domain kustom sedang diinisialisasi.

10. Selanjutnya, Anda mengonfigurasi data DNS dengan penyedia DNS Anda untuk memetakan nama domain kustom keCloudFront distribusi terkait. Untuk petunjuk tentang Amazon Route 53, lihat Perutekan lalu lintas ke Amazon API Gateway dengan menggunakan nama domain Anda dalam Panduan Pengembang Amazon Route 53.

    Untuk sebagian besar penyedia DNS, nama domain kustom ditambahkan ke zona host sebagai kumpulan catatan sumber daya CNAME. Nama data CNAME menentukan nama domain kustom yang Anda masukkan sebelumnya di Nama Domain (misalnya,`api.example.com`). Nilai rekaman CNAME menentukan nama domain untukCloudFront distribusi. Namun, penggunaan data CNAME tidak akan berfungsi jika domain kustom Anda adalah puncak zona (yaitu,`example.com` bukan`api.example.com`). Apex zona juga dikenal sebagai domain root organisasi Anda. Untuk puncak zona Anda perlu menggunakan alias A-record, asalkan didukung oleh penyedia DNS Anda.

Dengan Route 53 Anda dapat membuat alias rekaman A untuk nama domain kustom Anda dan menentukan nama domainCloudFront distribusi sebagai target alias. Ini berarti bahwa Route 53 dapat merutekan nama domain kustom Anda meskipun itu adalah puncak zona. Untuk informasi lebih lanjut, lihat Memilih antara set catatan sumber daya alias dan non-alias dalam Panduan Pengembang Amazon Route 53.

Penggunaan alias A-record juga menghilangkan paparan nama domainCloudFront distribusi yang mendasarinya karena pemetaan nama domain dilakukan hanya dalam Route 53. Untuk alasan ini, kami menyarankan Anda menggunakan Route 53 A-record alias bila memungkinkan.

Selain menggunakan konsol API Gateway, Anda dapat menggunakan API Gateway REST API,AWS CLI, atau salah satuAWS SDK untuk mengatur nama domain khusus untuk API Anda. Sebagai ilustrasi, prosedur berikut menguraikan langkah-langkah untuk melakukannya menggunakan panggilan REST API.

Untuk menyiapkan nama domain kustom menggunakan API REST API Gateway

1. Panggil nama domain:buat, tentukan nama domain kustom dan ARN sertifikat yang disimpan di dalamnyaAWS Certificate Manager.

   Panggilan API yang berhasil mengembalikan`201 Created` respons yang berisi ARN sertifikat serta namaCloudFront distribusi terkait dalam payloadnya.
2. Perhatikan nama domainCloudFront distribusi yang ditunjukkan pada output. Anda memerlukannya di langkah berikutnya untuk menetapkan nilai CNAME domain kustom atau target alias A-record di DNS Anda.
3. Ikuti prosedur sebelumnya untuk menyiapkan alias A-record untuk memetakan nama domain khusus ke namaCloudFront distribusinya.

Untuk contoh kode panggilan REST API ini, lihat domainname:create.

Log pembuatan nama domain kustomCloudTrail

SaatCloudTrail diaktifkan untuk mencatat panggilan API Gateway yang dilakukan oleh akun Anda, API Gateway mencatat pembaruanCloudFront distribusi terkait saat nama domain khusus dibuat atau diperbarui untuk API. KarenaCloudFront distribusi ini dimiliki oleh API Gateway, masing-masingCloudFront distribusi yang dilaporkan ini diidentifikasi oleh salah satu ID akun API Gateway khusus Wilayah berikut, bukan ID akun pemilik API.

Wilayah	ID Akun
us-east-1	392220576650
us–east-2	718770453195
us-west-1	968246515281
us-west-2	109351309407
ca-central-1	796887884028
eu-west-1	631144002099
eu-west-2	544388816663
eu-west-3	061510835048
eu-central-1	474240146802
eu-central-2	166639821150
eu-north-1	394634713161
eu-south-1	753362059629
eu-south-2	359345898052
ap-northeast-1	969236854626
ap-northeast-2	020402002396
ap-northeast-3	360671645888
ap-southeast-1	195145609632
ap-southeast-2	798376113853
ap-southeast-3	652364314486
ap-southeast-4	849137399833

Wilayah	ID Akun
ap-south-1	507069717855
ap-south-2	644042651268
ap-east-1	174803364771
sa-east-1	287228555773
me-south-1	855739686837
me-central-1	614065512851

Konfigurasikan pemetaan jalur dasar API dengan nama domain khusus sebagai nama hostnya

Anda dapat menggunakan nama domain kustom tunggal sebagai nama host dari beberapa API. Anda mencapai ini dengan mengkonfigurasi pemetaan jalur dasar pada nama domain kustom. Dengan pemetaan jalur dasar, API di bawah domain kustom dapat diakses melalui kombinasi nama domain kustom dan jalur dasar terkait.

Misalnya, jika Anda membuat API bernama`PetStore` dan API lain bernama`PetShop` dan menyiapkan nama domain kustom`api.example.com` di API Gateway, Anda dapat mengatur URL`PetStore` API sebagai`https://api.example.com` atau`https://api.example.com/myPetStore`. `PetStore`API dikaitkan dengan jalur dasar string kosong atau`myPetStore` di bawah nama domain kustom`api.example.com`. Demikian pula, Anda dapat menetapkan jalur dasar`yourPetShop` untuk`PetShop` API. URL dari`https://api.example.com/yourPetShop` kemudian URL root`PetShop` API.

Sebelum mengatur jalur dasar untuk API, selesaikan langkah-langkahnyaMenyiapkan nama domain kustom yang dioptimalkan tepi untuk API Gateway API.

Prosedur berikut mengatur pemetaan API untuk memetakan jalur dari nama domain kustom Anda ke tahap API Anda.

Untuk membuat nama pemetaan API menggunakan konsol API Gateway

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih nama domain kustom.

3. Pilih Konfigurasi pemetaan API.
4. Pilih Tambahkan pemetaan baru.
5. Tentukan API, Stage, dan Path (opsional) untuk pemetaan.
6. Pilih Save (Simpan).

Selain itu, Anda dapat memanggil API Gateway REST API,AWS CLI, atau salah satuAWS SDK untuk mengatur pemetaan jalur dasar API dengan nama domain khusus sebagai hostname-nya. Sebagai ilustrasi, prosedur berikut menguraikan langkah-langkah untuk melakukannya menggunakan panggilan REST API.

Untuk menyiapkan pemetaan jalur dasar API menggunakan API REST API Gateway

- Panggil basepathmapping:create pada nama domain kustom tertentu`basePath`, menentukan`restApiId`,, dan`stage` properti deployment dalam payload permintaan.

  Panggilan API yang berhasil mengembalikan`201 Created` respons.

Untuk contoh kode panggilan REST API, lihat basepathmapping:create.

Memutar sertifikat yang diimpor ke ACM

ACM secara otomatis menangani perpanjangan sertifikat yang dikeluarkannya. Anda tidak perlu memutar sertifikat yang dikeluarkan ACM untuk nama domain kustom Anda. CloudFrontmenanganinya atas nama Anda.

Namun, jika Anda mengimpor sertifikat ke ACM dan menggunakannya untuk nama domain kustom, Anda harus memutar sertifikat sebelum kedaluwarsa. Ini melibatkan mengimpor sertifikat pihak ketiga baru untuk nama domain dan memutar sertifikat yang ada ke yang baru. Anda perlu mengulangi proses saat sertifikat yang baru diimpor berakhir. Atau, Anda dapat meminta ACM untuk mengeluarkan sertifikat baru untuk nama domain dan memutar yang sudah ada ke sertifikat baru yang dikeluarkan ACM. Setelah itu, Anda dapat meninggalkan ACM danCloudFront menangani rotasi sertifikat untuk Anda secara otomatis. Untuk membuat atau mengimpor sertifikat ACM baru, ikuti langkah-langkah untuk meminta atau mengimpor sertifikat ACM baru untuk nama domain yang ditentukan.

Untuk memutar sertifikat nama domain, Anda dapat menggunakan konsol API Gateway, API REST API Gateway API,AWS CLI, atau salah satuAWS SDK.

Untuk memutar sertifikat kedaluwarsa yang diimpor ke ACM menggunakan konsol API Gateway

1. Meminta atau mengimpor sertifikat di ACM.
2. Kembali ke konsol API Gateway.
3. Pilih Nama domain khusus dari panel navigasi utama konsol API Gateway.
4. Pilih nama domain kustom.
5. Pilih Edit.
6. Pilih sertifikat yang diinginkan dari daftar tarik-turun sertifikat ACM.
7. Pilih Simpan untuk mulai memutar sertifikat untuk nama domain kustom.

> **Note**
>
> Dibutuhkan sekitar 40 menit untuk proses selesai. Setelah rotasi selesai, Anda dapat memilih ikon panah dua arah di sebelah Sertifikat ACM untuk memutar kembali ke sertifikat asli.

Untuk mengilustrasikan cara memutar sertifikat yang diimpor secara terprogram untuk nama domain kustom, kami menguraikan langkah-langkah menggunakan API Gateway REST API.

Putar sertifikat yang diimpor menggunakan API REST API Gateway

- Panggil nama domain:update tindakan, menentukan ARN sertifikat ACM baru untuk nama domain yang ditentukan.

Panggil API Anda dengan nama domain khusus

Memanggil API dengan nama domain kustom sama dengan memanggil API dengan nama domain defaultnya, asalkan URL yang benar digunakan.

Contoh berikut membandingkan dan membedakan satu set URL default dan URL kustom yang sesuai dari dua API (`udxjefdanqf3duz`) dalam Region (`us-east-1`) tertentu, dan nama domain kustom yang diberikan (`api.example.com`).

URL root API dengan nama domain default dan kustom

ID API	Tahap	URL default	Jalur dasar	URL Kustom
udxjef	pro	https://udxjef.execute-api.us-east-1.amazonaws.com/pro	/toko hewan peliharaan	https://api.example.com/petstore
udxjef	tst	https://udxjef.execute-api.us-east-1.amazonaws.com/tst	/Petdepot	https://api.example.com/petdepot
qf3duz	dev	https://qf3duz.execute-api.us-east-1.amazonaws.com/dev	/toko buku	https://api.example.com/bookstore
qf3duz	tst	https://qf3duz.execute-api.us-east-1.amazonaws.com/tst	/rak buku	https://api.example.com/bookstand

API Gateway mendukung nama domain kustom untuk API dengan menggunakan Server Name Indication (SNI). Anda dapat memanggil API dengan nama domain kustom menggunakan browser atau pustaka klien yang mendukung SNI.

API Gateway memberlakukan SNI padaCloudFront distribusi. Untuk informasi tentang caraCloudFront menggunakan nama domain kustom, lihat AmazonCloudFront Custom SSL.

Menyiapkan nama domain kustom regional di API Gateway

Anda dapat membuat nama domain khusus untuk endpoint API Regional (untukAWSWilayah). Untuk membuat nama domain khusus, Anda harus memberikan sertifikat ACM khusus Wilayah.

Untuk informasi selengkapnya tentang membuat atau mengunggah sertifikat nama domain khusus, lihatMendapatkan sertifikat siap diAWS Certificate Manager.

> **Important**
>
> Untuk nama domain kustom API Gateway Regional, Anda harus meminta atau mengimpor sertifikat di Wilayah yang sama dengan API Anda.

Saat Anda membuat nama domain kustom Regional (atau memigrasikannya) dengan sertifikat ACM, API Gateway akan membuat peran tertaut layanan di akun Anda jika perannya belum ada. Peran terkait layanan diperlukan untuk melampirkan sertifikat ACM Anda ke titik akhir Regional Anda. Peran ini diberi namaAWSServiceroleForaPigatewaydan akan memilikiAPIGatewayServiceRolePolicykebijakan yang dikelola melekat padanya. Untuk informasi selengkapnya tentang penggunaan peran tertaut layanan, lihatMenggunakan Peran Terkait Layanan.

> **Important**
>
> Anda harus membuat data DNS untuk mengarahkan nama domain kustom ke nama domain Regional. Hal ini memungkinkan lalu lintas yang terikat pada nama domain khusus untuk diarahkan ke nama host Regional API. Catatan DNS dapat berupa tipe CNAME atau “A”.

Topik

- Menyiapkan nama domain kustom regional dengan sertifikat ACM menggunakan konsol API Gateway
- Menyiapkan nama domain kustom regional dengan sertifikat ACM menggunakanAWS CLI

Menyiapkan nama domain kustom regional dengan sertifikat ACM menggunakan konsol API Gateway

Untuk menggunakan konsol API Gateway untuk mengatur nama domain kustom Regional, gunakan prosedur berikut.

Untuk menyiapkan nama domain kustom regional menggunakan konsol API Gateway

1. Masuk ke konsol API Gatewayhttps://console.aws.amazon.com/apigateway.
2. PilihNama domain khususdari panel navigasi utama.

3. Pilih Create (Buat).
4. UntukNama domain, masukkan nama domain.
5. Di bawah Konfigurasi, pilihRegional.
6. Pilih versi TLS minimum.
7. Pilih sertifikat ACM. Sertifikat harus berada di Wilayah yang sama dengan API.
8. Pilih Create (Buat).
9. Ikuti dokumentasi Route 53mengonfigurasi Route 53 untuk merutekan lalu lintas ke API Gateway.

Prosedur berikut mengatur pemetaan API untuk memetakan jalur dari nama domain kustom Anda ke tahapan API Anda.

Untuk membuat nama pemetaan API menggunakan konsol API Gateway

1. Masuk ke konsol API Gatewayhttps://console.aws.amazon.com/apigateway.
2. Pilih nama domain khusus.
3. PilihMengonfigurasi pemetaan API.
4. PilihTambahkan pemetaan baru.
5. TentukanAPI,Tahap, danJaluruntuk pemetaan.
6. Pilih Save (Simpan).

Untuk mempelajari tentang menyetel pemetaan basepath untuk domain kustom, lihatKonfigurasikan pemetaan jalur dasar API dengan nama domain khusus sebagai nama hostnya.

Menyiapkan nama domain kustom regional dengan sertifikat ACM menggunakanAWS CLI

Untuk menggunakanAWS CLIUntuk menyiapkan nama domain khusus untuk API Regional, gunakan prosedur berikut.

1. Memanggil`create-domain-name`, menentukan nama domain kustom dan ARN sertifikat Regional.

```
aws apigatewayv2 create-domain-name \
    --domain-name 'regional.example.com' \
    --domain-name-configurations CertificateArn=arn:aws:acm:us-
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678
```

Perhatikan bahwa sertifikat yang ditentukan berasal dari`us-west-2`Region dan untuk contoh ini, kita berasumsi bahwa API yang mendasarinya berasal dari Wilayah yang sama.

Jika berhasil, panggilan mengembalikan hasil yang mirip dengan berikut ini:

```
{
    "ApiMappingSelectionExpression": "$request.basepath",
    "DomainName": "regional.example.com",
    "DomainNameConfigurations": [
        {
            "ApiGatewayDomainName": "d-id.execute-api.us-west-2.amazonaws.com",
            "CertificateArn": "arn:aws:acm:us-west-2:123456789012:certificate/id",
            "DomainNameStatus": "AVAILABLE",
            "EndpointType": "REGIONAL",
            "HostedZoneId": "id",
            "SecurityPolicy": "TLS_1_2"
        }
    ]
}
```

Parameter`DomainNameConfigurations`nilai properti mengembalikan hostname API Regional. Anda harus membuat data DNS untuk mengarahkan nama domain kustom Anda ke nama domain Regional ini. Hal ini memungkinkan lalu lintas yang terikat pada nama domain khusus untuk diarahkan ke nama host API Regional ini.

2. Buat data DNS untuk mengaitkan nama domain khusus dan nama domain Regional. Hal ini memungkinkan permintaan yang terikat pada nama domain kustom untuk dialihkan ke nama host Regional API.
3. Tambahkan pemetaan jalur dasar untuk mengekspos API yang ditentukan (misalnya,`0qzs2sy7bh`) dalam tahap penyebaran (misalnya,`test`) di bawah nama domain kustom yang ditentukan (misalnya,`regional.example.com`).

```
aws apigatewayv2 create-api-mapping \
    --domain-name 'regional.example.com' \
    --api-mapping-key 'myApi' \
    --api-id 0qzs2sy7bh \
    --stage 'test'
```

Akibatnya, URL dasar menggunakan nama domain khusus untuk API yang digunakan dalam tahap menjadi`https://regional.example.com/myAPI`.

4. Konfigurasikan data DNS Anda untuk memetakan nama domain kustom Regional ke hostname dari ID zona host yang diberikan. Pertama buat file JSON yang berisi konfigurasi untuk menyiapkan data DNS untuk nama domain Regional. Contoh berikut menunjukkan cara membuat DNSAmerekam untuk memetakan nama domain kustom Regional (`regional.example.com`) ke hostname Regional (`d-numh1z56v6.execute-api.us-west-2.amazonaws.com`) disediakan sebagai bagian dari pembuatan nama domain kustom. Parameter`DNSName`dan`HostedZoneId`sifat`AliasTarget`dapat mengambil`regionalDomainName`dan`regionalHostedZoneId`nilai-nilai, masing-masing, dari nama domain kustom. Anda juga bisa mendapatkan Regional Route 53 Hosted Zone ID di[Endpoint dan Kuota Amazon API Gateway](#).

```
{
  "Changes": [
    {
      "Action": "CREATE",
      "ResourceRecordSet": {
        "Name": "regional.example.com",
        "Type": "A",
        "AliasTarget": {
          "DNSName": "d-numh1z56v6.execute-api.us-west-2.amazonaws.com",
          "HostedZoneId": "Z2OJLYMUO9EFXC",
          "EvaluateTargetHealth": false
        }
      }
    }
  ]
}
```

5. Jalankan perintah CLI berikut:

```
aws route53 change-resource-record-sets \
    --hosted-zone-id {your-hosted-zone-id} \
    --change-batch file://path/to/your/setup-dns-record.json
```

di mana*`{Anda-hosted-zone-id}`*adalah Route 53 Hosted Zone ID dari catatan DNS yang ditetapkan di akun Anda. Parameter`change-batch`nilai parameter poin ke file JSON (*`pengaturan-dns-record.json`*) dalam folder (*`jalur/ke/Anda`*).

Migrasi nama domain kustom ke endpoint API yang berbeda

Anda dapat memigrasi nama domain kustom Anda antara titik akhir yang dioptimalkan tepi dan Regional. Anda pertama kali menambahkan tipe konfigurasi endpoint baru ke yang ada`endpointConfiguration.types`daftar untuk nama domain kustom. Selanjutnya, Anda menyiapkan data DNS untuk mengarahkan nama domain khusus ke titik akhir yang baru disediakan. Langkah terakhir opsional adalah menghapus data konfigurasi nama domain kustom usang.

Saat merencanakan migrasi, ingatlah bahwa untuk nama domain kustom API yang dioptimalkan dengan tepi, sertifikat yang diperlukan yang disediakan oleh ACM harus berasal dari Wilayah AS Timur (Virginia Utara) (Virginia N.)`us-east-1`). Sertifikat ini didistribusikan ke semua lokasi geografis. Namun, untuk API Regional, sertifikat ACM untuk nama domain Regional harus dari Wilayah yang sama dengan hosting API. Anda dapat memigrasi nama domain kustom yang dioptimalkan tepi yang tidak ada di`us-east-1`Wilayah ke nama domain kustom Regional dengan terlebih dahulu meminta sertifikat ACM baru dari Wilayah yang lokal ke API.

Diperlukan waktu hingga 60 detik untuk menyelesaikan migrasi antara nama domain kustom yang dioptimalkan dengan tepi dan nama domain kustom Regional di API Gateway. Agar titik akhir yang baru dibuat agar siap menerima lalu lintas, waktu migrasi juga tergantung pada kapan Anda memperbarui data DNS Anda.

Topik

- Migrasi nama domain kustom menggunakanAWS CLI

Migrasi nama domain kustom menggunakanAWS CLI

Untuk menggunakanAWS CLIuntuk memigrasikan nama domain kustom dari titik akhir yang dioptimalkan tepi ke titik akhir Regional atau sebaliknya, hubungi`update-domain-name`perintah untuk menambahkan jenis endpoint baru dan, opsional, memanggil`update-domain-name`perintah untuk menghapus tipe endpoint lama.

Topik

- Migrasikan nama domain kustom yang dioptimalkan dengan tepi ke regional
- Migrasikan nama domain kustom regional ke tepi-dioptimalkan

Migrasikan nama domain kustom yang dioptimalkan dengan tepi ke regional

Untuk memigrasikan nama domain kustom yang dioptimalkan dengan tepi ke nama domain kustom Regional, hubungi`update-domain-name`Perintah CLI, sebagai berikut:

```
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' } \
      ]
```

Sertifikat Wilayah harus memiliki Wilayah yang sama dengan API Wilayah.

Respon keberhasilan memiliki`200 OK`kode status dan body yang serupa dengan yang berikut ini:

```
{
    "certificateArn": "arn:aws:acm:us-
east-1:123456789012:certificate/34a95aa1-77fa-427c-aa07-3a88bd9f3c0a",
    "certificateName": "edge-cert",
    "certificateUploadDate": "2017-10-16T23:22:57Z",
    "distributionDomainName": "d1frvgze7vy1bf.cloudfront.net",
    "domainName": "api.example.com",
    "endpointConfiguration": {
        "types": [
            "EDGE",
            "REGIONAL"
        ]
    },
    "regionalCertificateArn": "arn:aws:acm:us-west-2:123456789012:certificate/
cd833b28-58d2-407e-83e9-dce3fd852149",
    "regionalDomainName": "d-fdisjghyn6.execute-api.us-west-2.amazonaws.com"
}
```

Untuk nama domain kustom Regional yang dimigrasi, hasil`regionalDomainName`properti mengembalikan nama host API Regional. Anda harus menyiapkan data DNS untuk mengarahkan nama domain kustom Regional ke nama host Regional ini. Hal ini memungkinkan lalu lintas yang terikat dengan nama domain kustom untuk diarahkan ke host Regional.

Setelah data DNS diatur, Anda dapat menghapus nama domain kustom yang dioptimalkan tepi dengan memanggil[update-domain-name](#)perintahAWS 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'} \
        ]
```

Migrasikan nama domain kustom regional ke tepi-dioptimalkan

Untuk memigrasikan nama domain kustom Regional ke nama domain kustom yang dioptimalkan dengan tepi, hubungi`update-domain-name`perintahAWS CLI, sebagai berikut:

```
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' } \
        ]
```

Sertifikat domain edge-dioptimalkan harus dibuat dalam`us-east-1`Wilayah.

Respon keberhasilan memiliki`200 OK`kode status dan body yang serupa dengan yang berikut ini:

```
{
    "certificateArn": "arn:aws:acm:us-
east-1:738575810317:certificate/34a95aa1-77fa-427c-aa07-3a88bd9f3c0a",
    "certificateName": "edge-cert",
    "certificateUploadDate": "2017-10-16T23:22:57Z",
    "distributionDomainName": "d1frvgze7vy1bf.cloudfront.net",
    "domainName": "api.example.com",
    "endpointConfiguration": {
        "types": [
            "EDGE",
            "REGIONAL"
        ]
    },
    "regionalCertificateArn": "arn:aws:acm:us-
east-1:123456789012:certificate/3d881b54-851a-478a-a887-f6502760461d",
    "regionalDomainName": "d-cgkq2qwgzf.execute-api.us-east-1.amazonaws.com"
```

```
}
```

Untuk nama domain kustom yang ditentukan, API Gateway mengembalikan nama host API yang dioptimalkan tepi sebagai`distributionDomainName`nilai properti. Anda harus menetapkan data DNS untuk mengarahkan nama domain kustom yang dioptimalkan tepi ke nama domain distribusi ini. Hal ini memungkinkan lalu lintas yang terikat dengan nama domain kustom yang dioptimalkan edge untuk dirutekan ke nama host API yang dioptimalkan edge.

Setelah catatan DNS diatur, Anda dapat menghapus`REGIONAL`jenis endpoint dari nama domain kustom:

```
aws apigateway update-domain-name \
    --domain-name api.example.com \
    --patch-operations [ \
        {op:'remove', path:'/endpointConfiguration/types', value:'REGIONAL'}, \
        {op:'remove', path:'regionalCertificateArn'} \
      ]
```

Hasil dari perintah ini mirip dengan output berikut, dengan hanya data konfigurasi nama domain edge-dioptimalkan:

```
{
    "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"
    }
}
```

Bekerja dengan pemetaan API untuk REST API

Anda menggunakan pemetaan API untuk menghubungkan tahapan API ke nama domain kustom. Setelah membuat nama domain dan mengonfigurasi catatan DNS, Anda menggunakan pemetaan API untuk mengirim lalu lintas ke API melalui nama domain kustom Anda.

Pemetaan API menentukan API, tahap, dan jalur opsional yang akan digunakan untuk pemetaan. Misalnya, Anda dapat memetakan`production`tahap API ke`https://api.example.com/orders`.

Anda dapat memetakan tahap HTTP dan REST API ke nama domain kustom yang sama.

Sebelum membuat pemetaan API, Anda harus memiliki API, panggung, dan nama domain khusus. Untuk mempelajari selengkapnya tentang membuat nama domain kustom, lihatthe section called “Menyiapkan nama domain kustom regional”.

Permintaan API perutean

Anda dapat mengonfigurasi pemetaan API dengan beberapa level, misalnya`orders/v1/items`dan`orders/v2/items`.

> **Note**
>
> Untuk mengonfigurasi pemetaan API dengan beberapa level, nama domain kustom Anda harus bersifat regional dan menggunakan kebijakan keamanan TLS 1.2.

Untuk pemetaan API dengan beberapa level, API Gateway merutekan permintaan ke pemetaan API yang memiliki jalur pencocokan terpanjang. API Gateway hanya mempertimbangkan jalur yang dikonfigurasi untuk pemetaan API, dan bukan rute API, untuk memilih API yang akan dipanggil. Jika tidak ada jalur yang cocok dengan permintaan, API Gateway mengirimkan permintaan ke API yang telah Anda petakan ke jalur kosong(`none`).

Untuk nama domain kustom yang menggunakan pemetaan API dengan beberapa level, API Gateway merutekan permintaan ke pemetaan API yang memiliki awalan pencocokan terpanjang.

Misalnya, pertimbangkan nama domain khusus`https://api.example.com`dengan pemetaan API berikut:

1. (`none`)dipetakan ke API 1.
2. `orders`dipetakan ke API 2.
3. `orders/v1/items`dipetakan ke API 3.
4. `orders/v2/items`dipetakan ke API 4.
5. `orders/v2/items/categories`dipetakan ke API 5.

Permintaan	API yang dipilih	Penjelasan
`https://api.example.com/orders`	`API 2`	Permintaan sama persis dengan pemetaan API ini.
`https://api.example.com/orders/v1/items`	`API 3`	Permintaan sama persis dengan pemetaan API ini.
`https://api.example.com/orders/v2/items`	`API 4`	Permintaan sama persis dengan pemetaan API ini.
`https://api.example.com/orders/v1/items/123`	`API 3`	API Gateway memilih pemetaan yang memiliki jalur pencocokan terpanjang. The`123`di akhir permintaan tidak mempengaruhi pemilihan .
`https://api.example.com/orders/v2/items/categories/5`	`API 5`	API Gateway memilih pemetaan yang memiliki jalur pencocokan terpanjang.
`https://api.example.com/customers`	`API 1`	API Gateway menggunakan pemetaan kosong sebagai tangkapan semua.
`https://api.example.com/ordersandmore`	`API 2`	API Gateway memilih pemetaan yang memiliki awalan pencocokan terpanjang. Untuk nama domain kustom yang dikonfigurasi dengan pemetaan satu tingkat, seperti saja`https://api.example.com/orders` dan`https://api.example.com/` , API

Permintaan	API yang dipilih	Penjelasan
		Gateway akan memilih`API 1`, karena tidak ada jalur yang cocok dengan`ordersand more` .

Pembatasan

- Dalam pemetaan API, nama domain khusus dan API yang dipetakan harus samaAWSakun.
- Pemetaan API harus hanya berisi huruf, angka, dan karakter berikut:`$-_.+!*'()/`.
- Panjang maksimum jalur dalam pemetaan API adalah 300 karakter.
- Anda dapat memiliki 200 pemetaan API dengan beberapa level untuk setiap nama domain.
- Anda hanya dapat memetakan API HTTP ke nama domain kustom regional dengan kebijakan keamanan TLS 1.2.
- Anda tidak dapat memetakanWebSocketAPI ke nama domain kustom yang sama dengan API HTTP atau REST API.

Buat pemetaan API

Untuk membuat pemetaan API, Anda harus terlebih dahulu membuat nama domain kustom, API, dan stage. Untuk informasi tentang membuat nama domain kustom, lihatthe section called “Menyiapkan nama domain kustom regional”.

Sebagai contohAWS Serverless Application Modeltemplate yang membuat semua sumber daya, lihatSesi Dengan SAMdi atasGitHub.

AWS Management Console

Untuk membuat pemetaan API

1. Masuk ke konsol API Gateway dihttps://console.aws.amazon.com/apigateway.
2. PilihNama domain kustom.
3. Pilih nama domain khusus yang sudah Anda buat.
4. PilihPemetaan API.
5. PilihKonfigurasikan pemetaan API.

6. PilihTambahkan pemetaan baru.
7. Masukkan sebuahAPI, sebuahPanggung, dan secara opsional aJalan.
8. Pilih Save (Simpan).

AWS CLI

Berikut iniAWS CLIperintah membuat pemetaan API. Dalam contoh ini, API Gateway mengirimkan permintaan ke`api.example.com/v1/orders`ke API dan panggung yang ditentukan.

> **Note**
>
> Untuk membuat pemetaan API dengan beberapa level, Anda harus menggunakan`apigatewayv2`.

```
aws apigatewayv2 create-api-mapping \
    --domain-name api.example.com \
    --api-mapping-key v1/orders \
    --api-id a1b2c3d4 \
    --stage test
```

AWS CloudFormation

Berikut iniAWS CloudFormationcontoh membuat pemetaan API.

> **Note**
>
> Untuk membuat pemetaan API dengan beberapa level, Anda harus menggunakan`AWS::ApiGatewayV2`.

```
MyApiMapping:
  Type: 'AWS::ApiGatewayV2::ApiMapping'
  Properties:
    DomainName: api.example.com
    ApiMappingKey: 'orders/v2/items'
    ApiId: !Ref MyApi
    Stage: !Ref MyStage
```

Menonaktifkan titik akhir default untuk REST API

Secara default, klien dapat memanggil API Anda dengan menggunakan `execute-api` titik akhir yang dihasilkan API Gateway untuk API Anda. Untuk memastikan bahwa klien dapat mengakses API Anda hanya dengan menggunakan nama domain khusus, nonaktifkan `execute-api` titik akhir default. Klien masih dapat terhubung ke titik akhir default Anda, tetapi mereka akan menerima kode `403 Forbidden` status.

> **Note**
>
> Saat Anda menonaktifkan titik akhir default, itu memengaruhi semua tahapan API.

AWS CLIPerintah berikut menonaktifkan titik akhir default untuk REST API.

```
aws apigateway update-rest-api \
    --rest-api-id abcdef123 \
    --patch-operations op=replace,path=/disableExecuteApiEndpoint,value='True'
```

Setelah menonaktifkan titik akhir default, Anda harus menerapkan API agar perubahan diterapkan.

AWS CLIPerintah berikut membuat penyebaran.

```
aws apigateway create-deployment \
    --rest-api-id abcdef123 \
    --stage-name dev
```

Konfigurasikan pemeriksaan kesehatan khusus untuk failover DNS

Anda dapat menggunakan pemeriksaan kesehatan Amazon Route 53 untuk mengontrol failover DNS dari API Gateway API di primerWilayah AWSke satu di wilayah sekunder. Ini dapat membantu mengurangi dampak jika terjadi masalah Regional. Jika Anda menggunakan domain kustom, Anda dapat melakukan failover tanpa mengharuskan klien untuk mengubah titik akhir API.

Ketika Anda memilihEvaluasi Kesehatan Targetuntuk catatan alias, catatan tersebut gagal hanya jika layanan API Gateway tidak tersedia di Wilayah. Dalam beberapa kasus, API Gateway API Anda sendiri dapat mengalami gangguan sebelum waktu tersebut. Untuk mengontrol failover DNS

secara langsung, konfigurasikan pemeriksaan kesehatan Route 53 kustom untuk API Gateway API Anda. Untuk contoh ini, Anda menggunakanCloudWatchalarm yang membantu operator mengontrol failover DNS. Untuk contoh selengkapnya dan pertimbangan lain saat Anda mengonfigurasi failover, lihatMembuat Mekanisme Pemulihan Bencana Menggunakan Rute 53danMelakukan pemeriksaan kesehatan Route 53 pada sumber daya pribadi di VPC denganAWS LambdadanCloudWatch.

Topik

- Prasyarat
- Langkah 1: Siapkan sumber daya
- Langkah 2: Memulai failover ke Wilayah sekunder
- Langkah 3: Uji failover
- Langkah 4: Kembali ke wilayah utama
- Langkah selanjutnya: Sesuaikan dan uji secara teratur

Prasyarat

Untuk menyelesaikan prosedur ini, Anda harus membuat dan mengonfigurasi sumber daya berikut:

- Nama domain yang Anda miliki.
- Sertifikat ACM untuk nama domain tersebut menjadi duaWilayah AWS. Untuk info lebih lanjut, lihatthe section called “Mendapatkan sertifikat siap diAWS Certificate Manager”.
- Zona yang dihosting Route 53 untuk nama domain Anda. Untuk informasi lebih lanjut, lihatBekerja dengan zona yang dihostingdi Panduan Pengembang Amazon Route 53.

Untuk informasi selengkapnya tentang cara membuat catatan DNS failover Route 53 untuk nama domain, lihatPilih kebijakan peruteandi Panduan Pengembang Amazon Route 53. Untuk informasi lebih lanjut tentang cara memantauCloudWatchalarm, lihatPemantauan aCloudWatchalarmdi Panduan Pengembang Amazon Route 53.

Langkah 1: Siapkan sumber daya

Dalam contoh ini, Anda membuat sumber daya berikut untuk mengonfigurasi failover DNS untuk nama domain Anda:

- API Gateway API menjadi duaWilayah AWS
- API Gateway nama domain kustom dengan nama yang sama dalam duaWilayah AWS

- Pemetaan API Gateway API yang menghubungkan API Gateway API Anda ke nama domain kustom
- Route 53 catatan DNS failover untuk nama domain
- SEBUAHCloudWatchalarm di Wilayah sekunder
- Pemeriksaan kesehatan Route 53 berdasarkanCloudWatchalarm di Wilayah sekunder

Pertama, pastikan Anda memiliki semua sumber daya yang diperlukan di Wilayah primer dan sekunder. Wilayah sekunder harus berisi alarm dan pemeriksaan kesehatan. Dengan cara ini, Anda tidak bergantung pada Wilayah utama untuk melakukan failover. Sebagai contohAWS CloudFormationtemplate yang membuat sumber daya ini, lihat`primary.yaml`dan`secondary.yaml`.

> **Important**
>
> Sebelum failover ke Wilayah sekunder, pastikan bahwa semua sumber daya yang diperlukan tersedia. Jika tidak, API Anda tidak akan siap untuk lalu lintas di Wilayah sekunder.

Langkah 2: Memulai failover ke Wilayah sekunder

Dalam contoh berikut, Wilayah siaga menerima aCloudWatchmetrik dan memulai failover. Kami menggunakan metrik khusus yang memerlukan intervensi operator untuk memulai failover.

```
aws cloudwatch put-metric-data \
    --metric-name Failover \
    --namespace HealthCheck \
    --unit Count \
    --value 1 \
    --region us-west-1
```

Ganti data metrik dengan data yang sesuai untukCloudWatchalarm yang Anda konfigurasikan.

Langkah 3: Uji failover

Panggil API Anda dan verifikasi bahwa Anda mendapatkan respons dari Wilayah sekunder. Jika Anda menggunakan contoh templat pada langkah 1, respons berubah dari`{"message": "Hello from the primary Region!"}`kepada`{"message": "Hello from the secondary Region!"}`setelah failover.

```
curl https://my-api.example.com

{"message": "Hello from the secondary Region!"}
```

Langkah 4: Kembali ke wilayah utama

Untuk kembali ke Wilayah utama, kirimCloudWatchmetrik yang menyebabkan pemeriksaan kesehatan lulus.

```
aws cloudwatch put-metric-data \
    --metric-name Failover \
    --namespace HealthCheck \
    --unit Count \
    --value 0 \
    --region us-west-1
```

Ganti data metrik dengan data yang sesuai untukCloudWatchalarm yang Anda konfigurasikan.

Panggil API Anda dan verifikasi bahwa Anda mendapatkan respons dari Wilayah utama. Jika Anda menggunakan contoh templat pada langkah 1, respons berubah dari`{"message": "Hello from the secondary Region!"}`kepada`{"message": "Hello from the primary Region!"}`.

```
curl https://my-api.example.com

{"message": "Hello from the primary Region!"}
```

Langkah selanjutnya: Sesuaikan dan uji secara teratur

Contoh ini menunjukkan salah satu cara untuk mengkonfigurasi failover DNS. Anda dapat menggunakan berbagaiCloudWatchmetrik atau titik akhir HTTP untuk pemeriksaan kesehatan yang mengelola failover. Uji mekanisme failover Anda secara teratur untuk memastikan bahwa mereka bekerja seperti yang diharapkan, dan operator terbiasa dengan prosedur failover Anda.

Mengoptimalkan kinerja REST API

Setelah Anda membuat API tersedia untuk dipanggil, Anda mungkin menyadari bahwa itu perlu dioptimalkan untuk meningkatkan daya tanggap. API Gateway menyediakan beberapa strategi untuk mengoptimalkan API Anda, seperti cache respons dan kompresi payload. Di bagian ini, Anda dapat mempelajari cara mengaktifkan kemampuan ini.

Topik

Mengaktifkan caching API untuk meningkatkan daya tanggap

Anda dapat mengaktifkan caching API di Amazon API Gateway untuk men-cache respons titik akhir Anda. Dengan caching, Anda dapat mengurangi jumlah panggilan yang dilakukan ke titik akhir Anda dan juga meningkatkan latensi permintaan ke API Anda.

Saat Anda mengaktifkan caching untuk suatu tahap, API Gateway menyimpan respons dari titik akhir Anda untuk periode tertentu time-to-live (TTL), dalam hitungan detik. API Gateway kemudian merespons permintaan dengan mencari respons titik akhir dari cache alih-alih membuat permintaan ke titik akhir Anda. Nilai TTL default untuk caching API adalah 300 detik. Nilai TTL maksimum adalah 3600 detik. TTL = 0 berarti caching dinonaktifkan.

> **Note**
>
> Caching adalah upaya terbaik. Anda dapat menggunakan `CacheMissCount` metrik `CacheHitCount` dan di Amazon CloudWatch untuk memantau permintaan yang disajikan API Gateway dari cache API.

Ukuran maksimum respons yang dapat di-cache adalah 1048576 byte. Enkripsi data cache dapat meningkatkan ukuran respons saat sedang di-cache.

Ini adalah Layanan yang Memenuhi Syarat HIPAA. Untuk informasi lebih lanjut tentang AWS, Undang-Undang Portabilitas dan Akuntabilitas Asuransi Kesehatan AS tahun 1996 (HIPAA), dan menggunakan AWS layanan untuk memproses, menyimpan, dan mengirimkan informasi kesehatan yang dilindungi (PHI), lihat Ikhtisar HIPAA.

> **Important**
>
> Saat Anda mengaktifkan caching untuk suatu tahap, hanya `GET` metode yang mengaktifkan caching secara default. Ini membantu memastikan keamanan dan ketersediaan API Anda. Anda dapat mengaktifkan caching untuk metode lain dengan mengganti pengaturan metode.

> **Important**
>
> Caching dibebankan per jam berdasarkan ukuran cache yang Anda pilih. Caching tidak memenuhi syarat untuk Tingkat AWS Gratis. Untuk informasi selengkapnya, lihat Harga API Gateway.

Aktifkan caching Amazon API Gateway

Di API Gateway, Anda dapat mengaktifkan caching untuk tahap tertentu.

Ketika Anda mengaktifkan caching, Anda harus memilih kapasitas cache. Secara umum, kapasitas yang lebih besar memberikan kinerja yang lebih baik, tetapi juga lebih mahal. Untuk ukuran cache yang didukung, lihat cacheClusterSizedi Referensi API Gateway API.

API Gateway memungkinkan caching dengan membuat instance cache khusus. Proses ini bisa memakan waktu hingga 4 menit.

API Gateway mengubah kapasitas caching dengan menghapus instance cache yang ada dan membuat yang baru dengan kapasitas yang dimodifikasi. Semua data cache yang ada dihapus.

> **Note**
>
> Kapasitas cache mempengaruhi CPU, memori, dan bandwidth jaringan dari instance cache. Akibatnya, kapasitas cache dapat memengaruhi kinerja cache Anda.
> API Gateway merekomendasikan agar Anda menjalankan uji pemuatan 10 menit untuk memverifikasi bahwa kapasitas cache sesuai dengan beban kerja Anda. Pastikan bahwa lalu lintas selama uji beban mencerminkan lalu lintas produksi. Misalnya, sertakan ramp up, lalu lintas konstan, dan lonjakan lalu lintas. Tes beban harus mencakup respons yang dapat disajikan dari cache, serta respons unik yang menambahkan item ke cache. Pantau metrik latensi, 4xx, 5xx, cache hit, dan cache miss selama uji beban. Sesuaikan kapasitas cache sesuai kebutuhan berdasarkan metrik ini. Untuk informasi selengkapnya tentang pengujian beban, lihat Bagaimana cara memilih kapasitas cache API Gateway terbaik agar tidak mencapai batas laju? .

Di konsol API Gateway, Anda mengonfigurasi caching di halaman Tahapan. Anda menyediakan cache tahap dan menentukan pengaturan cache tingkat metode default. Jika Anda mengaktifkan cache tingkat metode default, caching tingkat metode diaktifkan untuk semua metode di panggung

Anda, kecuali metode tersebut memiliki penggantian metode. GETMetode tambahan apa pun yang Anda terapkan ke tahap Anda akan memiliki cache tingkat metode. Untuk mengonfigurasi pengaturan caching tingkat metode untuk metode spesifik tahap Anda, Anda dapat menggunakan penggantian metode. Untuk informasi selengkapnya tentang penggantian metode, lihat. the section called “Ganti caching tahap untuk caching metode”

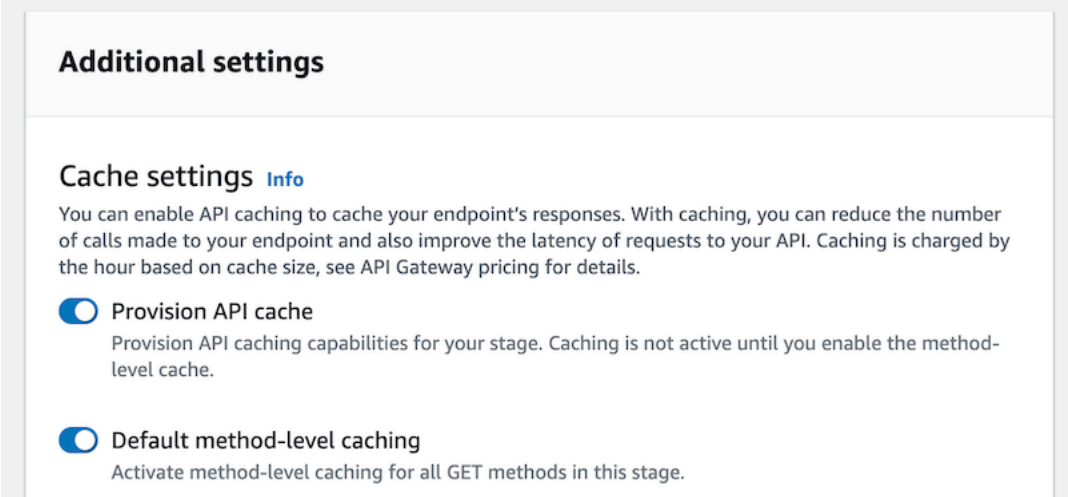
Untuk mengonfigurasi caching API untuk tahap tertentu:

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Memilih Tahapan.
3. Dalam daftar Tahapan untuk API, pilih stage.
4. Di bagian Detail tahap, pilih Edit.
5. Di bawah Pengaturan tambahan, untuk pengaturan Cache, aktifkan cache API Penyediaan.

   Ini menyediakan cluster cache untuk tahap Anda.
6. Untuk mengaktifkan caching untuk tahap Anda, aktifkan caching tingkat metode Default.

   Ini mengaktifkan caching tingkat metode untuk semua metode di panggung Anda. GETMetode tambahan apa pun yang Anda terapkan ke tahap ini akan memiliki cache tingkat metode.

> **Note**
>
> Jika Anda memiliki setelan yang ada untuk cache tingkat metode, mengubah pengaturan caching tingkat metode default tidak memengaruhi pengaturan yang ada.



7. Pilih Simpan perubahan.

> **Note**
>
> Membuat atau menghapus cache membutuhkan waktu sekitar 4 menit agar API Gateway selesai.
> Ketika cache dibuat, nilai cluster Cache berubah dari `Create in progress` ke`Active`.
> Ketika penghapusan cache selesai, nilai cluster Cache berubah dari `Delete in progress` ke. `Inactive`
> Saat Anda mengaktifkan caching tingkat metode untuk semua metode di panggung Anda, nilai caching tingkat metode Default berubah menjadi. `Active` Jika Anda menonaktifkan caching tingkat metode untuk semua metode di panggung Anda, nilai caching tingkat metode Default akan berubah menjadi. `Inactive` Jika Anda memiliki setelan yang ada untuk cache tingkat metode, mengubah status cache tidak memengaruhi pengaturan tersebut.

Saat Anda mengaktifkan caching dalam pengaturan Cache tahap, hanya `GET` metode yang di-cache. Untuk memastikan keamanan dan ketersediaan API Anda, sebaiknya jangan mengubah setelan ini. Namun, Anda dapat mengaktifkan caching untuk metode lain dengan mengganti pengaturan metode.

Jika Anda ingin memverifikasi apakah caching berfungsi seperti yang diharapkan, Anda memiliki dua opsi umum:

- Periksa CloudWatch metrik CacheHitCountdan CacheMissCountuntuk API dan panggung Anda.
- Masukkan stempel waktu dalam respons.

> **Note**
>
> Anda tidak boleh menggunakan `X-Cache` header dari CloudFront respons untuk menentukan apakah API Anda sedang dilayani dari instance cache API Gateway Anda.

Ganti caching tingkat tahap API Gateway untuk caching tingkat metode

Anda dapat mengganti pengaturan cache tingkat tahap dengan mengaktifkan atau mematikan caching untuk metode tertentu. Anda juga dapat memodifikasi periode TTL atau mengaktifkan atau menonaktifkan enkripsi untuk respons yang di-cache.

Jika Anda mengubah pengaturan caching tingkat metode default dalam detail Tahap, itu tidak memengaruhi pengaturan cache tingkat metode yang memiliki penggantian.

Jika Anda mengantisipasi bahwa metode yang Anda caching akan menerima data sensitif dalam tanggapannya, di Pengaturan Cache, pilih Enkripsi data cache.

Untuk mengonfigurasi caching API untuk metode individual menggunakan konsol:

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API.
3. Memilih Tahapan.
4. Dalam daftar Tahapan untuk API, perluas tahap dan pilih metode di API.
5. Di bagian Penggantian Metode, pilih Edit.
6. Di bagian Pengaturan metode, aktifkan atau matikan Aktifkan cache metode atau sesuaikan opsi lain yang diinginkan.

   > **Note**
   >
   > Caching tidak aktif sampai Anda menyediakan cluster cache untuk tahap Anda.

7. Pilih Simpan.

Gunakan metode atau parameter integrasi sebagai kunci cache untuk mengindeks respons yang di-cache

Ketika metode cache atau integrasi memiliki parameter, yang dapat mengambil bentuk header kustom, jalur URL, atau string kueri, Anda dapat menggunakan beberapa atau semua parameter untuk membentuk kunci cache. API Gateway dapat men-cache respons metode, tergantung pada nilai parameter yang digunakan.

> **Note**
>
> Kunci cache diperlukan saat mengatur caching pada sumber daya.

Misalnya, Anda memiliki permintaan dalam format berikut:

```
GET /users?type=... HTTP/1.1
host: example.com
...
```

Dalam permintaan ini, `type` dapat mengambil nilai `admin` atau`regular`. Jika Anda menyertakan `type` parameter sebagai bagian dari kunci cache, respons dari di-cache `GET /users?type=admin` secara terpisah dari yang dari`GET /users?type=regular`.

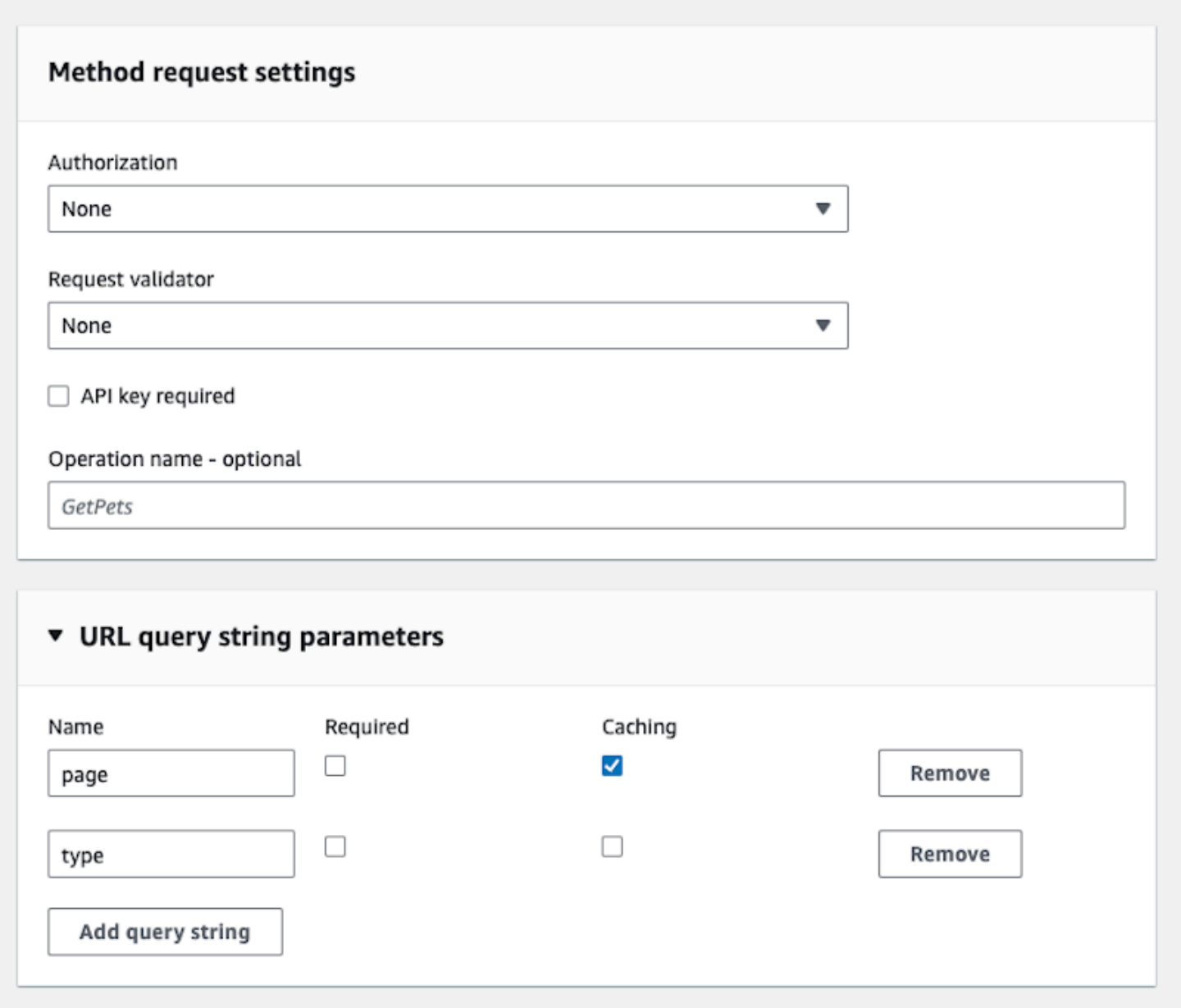
Ketika sebuah metode atau permintaan integrasi mengambil lebih dari satu parameter, Anda dapat memilih untuk menyertakan beberapa atau semua parameter untuk membuat kunci cache. Misalnya, Anda hanya dapat menyertakan `type` parameter dalam kunci cache untuk permintaan berikut, dibuat dalam urutan yang tercantum dalam periode TTL:

```
GET /users?type=admin&department=A HTTP/1.1
host: example.com
...
```

Respons dari permintaan ini di-cache dan digunakan untuk melayani permintaan berikut:

```
GET /users?type=admin&department=B HTTP/1.1
host: example.com
...
```

Untuk menyertakan metode atau parameter permintaan integrasi sebagai bagian dari kunci cache di konsol API Gateway, pilih Caching setelah Anda menambahkan parameter.



Siram cache tahap API di API Gateway

Saat caching API diaktifkan, Anda dapat membersihkan cache tahap API Anda untuk memastikan bahwa klien API Anda mendapatkan respons terbaru dari titik akhir integrasi Anda.

Untuk menyiram cache tahap API, pilih menu Tindakan tahap, lalu pilih Cache tahap Flush.

> **Note**
>
> Setelah cache dimatikan, respons dilayani dari titik akhir integrasi hingga cache dibangun kembali. Selama periode ini, jumlah permintaan yang dikirim ke titik akhir integrasi dapat meningkat. Ini dapat meningkatkan latensi keseluruhan API untuk sementara waktu.

Membatalkan entri cache API Gateway

Klien API Anda dapat membatalkan entri cache yang ada dan memuatnya kembali dari titik akhir integrasi untuk permintaan individual. Klien harus mengirim permintaan yang berisi `Cache-Control: max-age=0` header. Klien menerima respons langsung dari titik akhir integrasi alih-alih cache, asalkan klien berwenang untuk melakukannya. Ini menggantikan entri cache yang ada dengan respons baru, yang diambil dari titik akhir integrasi.

Untuk memberikan izin kepada klien, lampirkan kebijakan format berikut ke peran eksekusi IAM bagi pengguna.

> **Note**
>
> Pembatalan cache lintas akun tidak didukung.

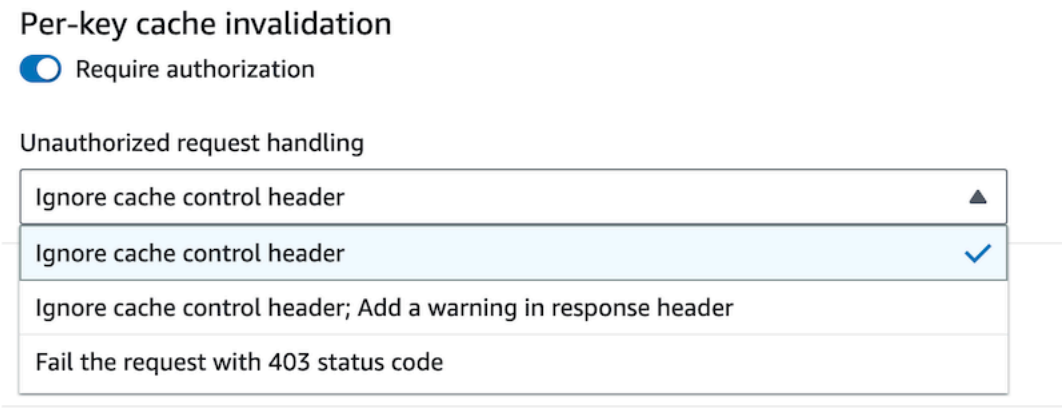
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:InvalidateCache"
      ],
      "Resource": [
        "arn:aws:execute-api:region:account-id:api-id/stage-name/GET/resource-path-specifier"
      ]
    }
  ]
}
```

Kebijakan ini memungkinkan layanan eksekusi API Gateway membatalkan cache untuk permintaan pada sumber daya tertentu (atau sumber daya). Untuk menentukan sekelompok sumber daya yang ditargetkan, gunakan karakter wildcard (*) untuk `account-idapi-id`, dan entri lain dalam nilai ARN. `Resource` Untuk informasi selengkapnya tentang cara menyetel izin untuk layanan eksekusi API Gateway, lihatKontrol akses ke API dengan izin IAM.

Jika Anda tidak memaksakan `InvalidateCache` kebijakan (atau memilih kotak centang Memerlukan otorisasi di konsol), klien mana pun dapat membatalkan cache API. Jika sebagian besar atau semua klien membatalkan cache API, ini dapat meningkatkan latensi API Anda secara signifikan.

Saat kebijakan diberlakukan, caching diaktifkan dan otorisasi diperlukan.

Anda dapat mengontrol cara penanganan permintaan yang tidak sah dengan memilih opsi dari penanganan permintaan Tidak Sah di konsol API Gateway.



Tiga opsi menghasilkan perilaku berikut:

- Gagal permintaan dengan kode status 403: mengembalikan respons 403 Tidak Sah.

  Untuk mengatur opsi ini menggunakan API, gunakan`FAIL_WITH_403`.
- Abaikan header kontrol cache; Tambahkan peringatan di header respons: proses permintaan dan tambahkan header peringatan dalam respons.

  Untuk mengatur opsi ini menggunakan API, gunakan`SUCCEED_WITH_RESPONSE_HEADER`.
- Abaikan header kontrol cache: proses permintaan dan jangan tambahkan header peringatan dalam respons.

  Untuk mengatur opsi ini menggunakan API, gunakan`SUCCEED_WITHOUT_RESPONSE_HEADER`.

Mengaktifkan kompresi payload untuk API

API Gateway memungkinkan klien Anda memanggil API Anda dengan muatan terkompresi dengan menggunakan salah satu pengkodean konten yang didukung. Secara default, API Gateway mendukung dekompresi payload permintaan metode. Namun, Anda harus mengonfigurasi API Anda untuk mengaktifkan kompresi payload respons metode.

Untuk mengaktifkan kompresi pada `API`, setel `minimumCompressionsSize`properti ke bilangan bulat non-negatif antara 0 dan 10485760 (10M byte) saat Anda membuat API atau setelah Anda membuat API. Untuk menonaktifkan kompresi pada API, atur `minimumCompressionSize` ke null atau hapus sama sekali. Anda dapat mengaktifkan atau menonaktifkan kompresi untuk API dengan menggunakan konsol API Gateway, APIAWS CLI, atau API Gateway REST API.

Jika Anda ingin kompresi diterapkan pada muatan dari berbagai ukuran, atur `minimumCompressionSize` nilainya ke nol. Namun, mengompresi data dengan ukuran kecil sebenarnya dapat meningkatkan ukuran data akhir. Selain itu, kompresi di API Gateway dan dekompresi di klien dapat meningkatkan latensi keseluruhan dan membutuhkan lebih banyak waktu komputasi. Anda harus menjalankan kasus pengujian terhadap API Anda untuk menentukan nilai optimal.

Klien dapat mengirimkan permintaan API dengan muatan terkompresi dan `Content-Encoding` header yang sesuai untuk API Gateway untuk mendekompresi dan menerapkan templat pemetaan yang berlaku, sebelum meneruskan permintaan ke titik akhir integrasi. Setelah kompresi diaktifkan dan API diterapkan, klien dapat menerima respons API dengan muatan terkompresi jika menentukan `Accept-Encoding` header yang sesuai dalam permintaan metode.

Saat titik akhir integrasi mengharapkan dan mengembalikan muatan JSON yang tidak terkompresi, templat pemetaan apa pun yang dikonfigurasi untuk muatan JSON yang tidak terkompresi berlaku untuk muatan terkompresi. Untuk payload permintaan metode terkompresi, API Gateway mendekompresi payload, menerapkan template pemetaan, dan meneruskan permintaan yang dipetakan ke titik akhir integrasi. Untuk payload respons integrasi yang tidak terkompresi, API Gateway menerapkan template pemetaan, memampatkan payload yang dipetakan, dan mengembalikan payload terkompresi ke klien.

Topik

- Aktifkan kompresi payload untuk API
- Panggil metode API dengan muatan terkompresi
- Menerima respons API dengan muatan terkompresi

Aktifkan kompresi payload untuk API

Anda dapat mengaktifkan kompresi untuk API menggunakan konsol API Gateway, theAWS CLI, atau AWS SDK.

Untuk API yang ada, Anda harus menerapkan API setelah mengaktifkan kompresi agar perubahan diterapkan. Untuk API baru, Anda dapat menerapkan API setelah penyiapan API selesai.

> **Note**
>
> Pengkodean konten dengan prioritas tertinggi harus didukung oleh API Gateway. Jika tidak, kompresi tidak diterapkan pada muatan respons.

Topik

Mengaktifkan kompresi payload untuk API menggunakan konsol API Gateway

Prosedur berikut menjelaskan cara mengaktifkan kompresi payload untuk API.

Untuk mengaktifkan kompresi payload dengan menggunakan konsol API Gateway

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API yang sudah ada atau buat yang baru.
3. Di panel navigasi utama, pilih pengaturan API.
4. Di bagian detail API, pilih Edit.
5. Aktifkan pengkodean konten untuk mengaktifkan kompresi muatan. Untuk ukuran tubuh Minimum, masukkan angka untuk ukuran kompresi minimum (dalam byte). Untuk mematikan kompresi, matikan opsi Pengkodean konten.
6. Pilih Save changes (Simpan perubahan).

Aktifkan kompresi payload untuk API menggunakan AWS CLI

Untuk menggunakan AWS CLI untuk membuat API baru dan mengaktifkan kompresi, panggil `create-rest-api`perintah sebagai berikut:

```
aws apigateway create-rest-api \
    --name "My test API" \
    --minimum-compression-size 0
```

Untuk menggunakan AWS CLI untuk mengaktifkan kompresi pada API yang ada, panggil `update-rest-api`perintah sebagai berikut:

```
aws apigateway update-rest-api \
    --rest-api-id 1234567890 \
    --patch-operations op=replace,path=/minimumCompressionSize,value=0
```

`minimumCompressionSize`Properti memiliki nilai integer non-negatif antara 0 dan 10485760 (10M byte). Ini mengukur ambang kompresi. Jika ukuran muatan lebih kecil dari nilai ini, kompresi atau dekompresi tidak diterapkan pada muatan. Pengaturan ke nol memungkinkan kompresi untuk ukuran muatan apa pun.

Untuk menggunakan AWS CLI untuk menonaktifkan kompresi, panggil `update-rest-api`perintah sebagai berikut:

```
aws apigateway update-rest-api \
    --rest-api-id 1234567890 \
    --patch-operations op=replace,path=/minimumCompressionSize,value=
```

Anda juga dapat mengatur `value` ke string kosong `""` atau menghilangkan `value` properti sama sekali dalam panggilan sebelumnya.

Pengkodean konten yang didukung oleh API Gateway

API Gateway mendukung pengkodean konten berikut:

- `deflate`
- `gzip`
- `identity`

API Gateway juga mendukung format `Accept-Encoding` header berikut, sesuai dengan spesifikasi RFC 7231:

- `Accept-Encoding:deflate,gzip`
- `Accept-Encoding:`
- `Accept-Encoding:*`
- `Accept-Encoding:deflate;q=0.5,gzip;q=1.0`
- `Accept-Encoding:gzip;q=1.0,identity;q=0.5,*;q=0`

Panggil metode API dengan muatan terkompresi

Untuk membuat permintaan API dengan muatan terkompresi, klien harus menyetel `Content-Encoding` header dengan salah satu pengkodean konten yang didukung.

Misalkan Anda adalah klien API dan ingin memanggil metode PetStore API (`POST /pets`). Jangan panggil metode dengan menggunakan output JSON berikut:

```
POST /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Content-Length: ...

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

Sebagai gantinya, Anda dapat memanggil metode dengan muatan yang sama dikompresi dengan menggunakan pengkodean 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
```

Ketika API Gateway menerima permintaan, itu memverifikasi apakah pengkodean konten yang ditentukan didukung. Kemudian, ia mencoba untuk mendekompresi muatan dengan pengkodean konten yang ditentukan. Jika dekompresi berhasil, ia mengirimkan permintaan ke titik akhir integrasi. Jika pengkodean yang ditentukan tidak didukung atau muatan yang disediakan tidak dikompresi dengan pengkodean tertentu, API Gateway mengembalikan respons `415 Unsupported Media`

Type kesalahan. Kesalahan tidak dicatat ke CloudWatch Log, jika terjadi pada fase awal dekompresi sebelum API dan tahap Anda diidentifikasi.

Menerima respons API dengan muatan terkompresi

Saat membuat permintaan pada API yang mendukung kompresi, klien dapat memilih untuk menerima muatan respons terkompresi dari format tertentu dengan menentukan `Accept-Encoding` header dengan pengkodean konten yang didukung.

API Gateway hanya memampatkan payload respons jika kondisi berikut terpenuhi:

- Permintaan masuk memiliki `Accept-Encoding` header dengan pengkodean dan format konten yang didukung.

  > **Note**
  >
  > Jika header tidak disetel, nilai default adalah * seperti yang didefinisikan dalam RFC 7231. Dalam kasus seperti itu, API Gateway tidak memampatkan muatan. Beberapa browser atau klien dapat menambahkan `Accept-Encoding` (misalnya,`Accept-Encoding:gzip, deflate, br`) secara otomatis ke permintaan yang diaktifkan kompresi. Ini dapat memicu kompresi payload di API Gateway. Tanpa spesifikasi eksplisit dari nilai `Accept-Encoding` header yang didukung, API Gateway tidak memampatkan payload.

- `minimumCompressionSize`Ini diatur pada API untuk mengaktifkan kompresi.
- Respons integrasi tidak memiliki `Content-Encoding` header.
- Ukuran muatan respons integrasi, setelah templat pemetaan yang berlaku diterapkan, lebih besar dari atau sama dengan nilai yang ditentukan`minimumCompressionSize`.

API Gateway menerapkan template pemetaan apa pun yang dikonfigurasi untuk respons integrasi sebelum mengompresi payload. Jika respons integrasi berisi `Content-Encoding` header, API Gateway mengasumsikan bahwa payload respons integrasi sudah dikompresi dan melewatkan pemrosesan kompresi.

Contohnya adalah contoh PetStore API dan permintaan berikut:

```
GET /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
```

```
Accept: application/json
```

Backend merespons permintaan dengan muatan JSON yang tidak terkompresi yang mirip dengan yang berikut ini:

```
200 OK

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

Untuk menerima output ini sebagai payload terkompresi, klien API Anda dapat mengirimkan permintaan sebagai berikut:

```
GET /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Accept-Encoding:gzip
```

Klien menerima respons dengan `Content-Encoding` header dan muatan yang dikodekan GZIP yang mirip dengan berikut ini:

```
200 OK
Content-Encoding:gzip
...

���RP�
```

```
J◆)JV
◆:P^IeA*◆◆◆◆◆◆+(◆L ◆X◆YZ◆ku0L0B7!9◆◆C#◆&◆◆◆◆Y◆◆a◆◆◆^◆X
```

Ketika payload respons dikompresi, hanya ukuran data terkompresi yang ditagih untuk transfer data.

Mendistribusikan REST API Anda ke klien

Bagian ini memberikan rincian tentang mendistribusikan API API Gateway Anda kepada pelanggan Anda. Mendistribusikan API Anda termasuk menghasilkan SDK bagi pelanggan Anda untuk mengunduh dan mengintegrasikan dengan aplikasi klien mereka, mendokumentasikan API Anda sehingga pelanggan tahu cara memanggilnya dari aplikasi klien mereka, dan membuat API Anda tersedia sebagai bagian dari penawaran produk.

Topik

- Membuat dan menggunakan paket penggunaan dengan kunci API
- Mendokumentasikan REST API
- Membuat SDK untuk REST API di API Gateway
- Jual API Gateway API Anda melaluiAWS Marketplace

Membuat dan menggunakan paket penggunaan dengan kunci API

Setelah membuat, menguji, dan menerapkan API, Anda dapat menggunakan paket penggunaan API Gateway agar tersedia sebagai penawaran produk bagi pelanggan Anda. Anda dapat mengonfigurasi paket penggunaan dan kunci API untuk memungkinkan pelanggan mengakses API yang dipilih, dan mulai membatasi permintaan ke API tersebut berdasarkan batas dan kuota yang ditentukan. Ini dapat diatur pada API, atau tingkat metode API.

Apa itu paket penggunaan dan kunci API?

Paket penggunaan menentukan siapa yang dapat mengakses satu atau beberapa tahapan dan metode API yang disebarkan—dan secara opsional menetapkan tingkat permintaan target untuk memulai pembatasan permintaan. Paket menggunakan kunci API untuk mengidentifikasi klien API dan siapa yang dapat mengakses tahapan API terkait untuk setiap kunci.

Kunci API adalah nilai string alfanumerik yang Anda distribusikan ke pelanggan pengembang aplikasi untuk memberikan akses ke API Anda. Anda dapat menggunakan kunci API bersama dengan

otorisasi Lambda, peran IAM, atau Amazon Cognito untuk mengontrol akses ke API Anda. API Gateway dapat menghasilkan kunci API atas nama Anda, atau Anda dapat mengimpornya dari file CSV. Anda dapat membuat kunci API di API Gateway, atau mengimpornya ke API Gateway dari sumber eksternal. Untuk informasi selengkapnya, lihat the section called “Mengatur kunci API menggunakan konsol API Gateway”.

Kunci API memiliki nama dan nilai. (Istilah “kunci API” dan “nilai kunci API” sering digunakan secara bergantian.) Nama tidak boleh melebihi 1024 karakter. Nilainya adalah string alfanumerik antara 20 dan 128 karakter, misalnya,. `apikey1234abcdefghij0123456789`

> **Important**
>
> Nilai kunci API harus unik. Jika Anda mencoba membuat dua kunci API dengan nama berbeda dan nilai yang sama, API Gateway menganggapnya sebagai kunci API yang sama. Kunci API dapat dikaitkan dengan lebih dari satu paket penggunaan. Rencana penggunaan dapat dikaitkan dengan lebih dari satu tahap. Namun, kunci API tertentu hanya dapat dikaitkan dengan satu paket penggunaan untuk setiap tahap API Anda.

Batas pelambatan menetapkan titik target di mana pelambatan permintaan harus dimulai. Ini dapat diatur pada tingkat metode API atau API.

Batas kuota menetapkan jumlah maksimum target permintaan dengan kunci API tertentu yang dapat dikirimkan dalam interval waktu tertentu. Anda dapat mengonfigurasi metode API individual agar memerlukan otorisasi kunci API berdasarkan konfigurasi paket penggunaan.

Batas pembatasan dan kuota berlaku untuk permintaan kunci API individual yang digabungkan di semua tahapan API dalam paket penggunaan.

> **Note**
>
> Pelambatan rencana penggunaan dan kuota bukanlah batas yang sulit, dan diterapkan atas dasar upaya terbaik. Dalam beberapa kasus, klien dapat melebihi kuota yang Anda tetapkan. Jangan mengandalkan kuota paket penggunaan atau pembatasan untuk mengontrol biaya atau memblokir akses ke API. Pertimbangkan AWS Budgetsuntuk menggunakan untuk memantau biaya dan AWS WAFmengelola permintaan API.

Praktik terbaik untuk kunci API dan paket penggunaan

Berikut ini adalah praktik terbaik yang disarankan untuk diikuti saat menggunakan kunci API dan rencana penggunaan.

> **Important**
>
> - Jangan gunakan kunci API untuk autentikasi atau otorisasi guna mengontrol akses ke API Anda. Jika Anda memiliki beberapa API dalam paket penggunaan, pengguna dengan kunci API yang valid untuk satu API dalam paket penggunaan tersebut dapat mengakses semua API dalam paket penggunaan tersebut. Sebagai gantinya, untuk mengontrol akses ke API Anda, gunakan peran IAM, otorisasi Lambda, atau kumpulan pengguna Amazon Cognito.
> - Gunakan kunci API yang dihasilkan API Gateway. Kunci API tidak boleh menyertakan informasi rahasia; klien biasanya mengirimkannya dalam header yang dapat dicatat.

- Jika Anda menggunakan portal pengembang untuk mempublikasikan API Anda, perhatikan bahwa semua API dalam paket penggunaan tertentu dapat dilanggan, meskipun Anda belum membuatnya terlihat oleh pelanggan Anda.
- Dalam beberapa kasus, klien dapat melebihi kuota yang Anda tetapkan. Jangan mengandalkan rencana penggunaan untuk mengontrol biaya. Pertimbangkan AWS Budgetsuntuk menggunakan untuk memantau biaya dan AWS WAFmengelola permintaan API.
- Setelah Anda menambahkan kunci API ke paket penggunaan, operasi pembaruan mungkin memerlukan beberapa menit untuk menyelesaikannya.

Langkah-langkah untuk mengonfigurasi paket penggunaan

Langkah-langkah berikut menguraikan bagaimana Anda, sebagai pemilik API, membuat dan mengonfigurasi paket penggunaan untuk pelanggan Anda.

Untuk mengonfigurasi paket penggunaan

1. Buat satu atau beberapa API, konfigurasikan metode untuk memerlukan kunci API, dan terapkan API ke tahapan.
2. Buat atau impor kunci API untuk didistribusikan ke pengembang aplikasi (pelanggan Anda) yang akan menggunakan API Anda.
3. Buat paket penggunaan dengan batas throttle dan kuota yang diinginkan.

4. Kaitkan tahapan API dan kunci API dengan paket penggunaan.

Penelepon API harus menyediakan kunci API yang ditetapkan di `x-api-key` header dalam permintaan ke API.

> **Note**
>
> Untuk menyertakan metode API dalam paket penggunaan, Anda harus mengonfigurasi metode API individual agar memerlukan kunci API. Untuk praktik terbaik untuk dipertimbangkan, lihatthe section called “Praktik terbaik untuk kunci API dan paket penggunaan”.

Pilih sumber kunci API

Saat Anda mengaitkan rencana penggunaan dengan API dan mengaktifkan kunci API pada metode API, setiap permintaan masuk ke API harus berisi kunci API. API Gateway membaca kunci dan membandingkannya dengan kunci dalam paket penggunaan. Jika ada kecocokan, API Gateway membatasi permintaan berdasarkan batas permintaan dan kuota paket. Kalau tidak, itu melempar `InvalidKeyParameter` pengecualian. Akibatnya, penelepon menerima `403 Forbidden` respons.

API Gateway API Anda dapat menerima kunci API dari salah satu dari dua sumber:

`HEADER`

Anda mendistribusikan kunci API ke pelanggan Anda dan meminta mereka untuk meneruskan kunci API sebagai `X-API-Key` header dari setiap permintaan yang masuk.

`AUTHORIZER`

Anda memiliki otorisasi Lambda yang mengembalikan kunci API sebagai bagian dari respons otorisasi. Untuk informasi selengkapnya tentang respons otorisasi, lihatthe section called “Keluaran dari otorisasi Lambda Amazon API Gateway”.

> **Note**
>
> Untuk praktik terbaik untuk dipertimbangkan, lihatthe section called “Praktik terbaik untuk kunci API dan paket penggunaan”.

Untuk memilih sumber kunci API untuk API dengan menggunakan konsol API Gateway

1. Masuk ke konsol API Gateway.
2. Pilih API yang sudah ada atau buat yang baru.
3. Di panel navigasi utama, pilih pengaturan API.
4. Di bagian detail API, pilih Edit.
5. Di bawah sumber kunci API, pilih `Header` atau `Authorizer` dari daftar tarik-turun.
6. Pilih Simpan perubahan.

Untuk memilih sumber kunci API untuk API dengan menggunakan AWS CLI, panggil `update-rest-api`perintah sebagai berikut:

```
aws apigateway update-rest-api --rest-api-id 1234123412 --patch-operations
 op=replace,path=/apiKeySource,value=AUTHORIZER
```

Untuk meminta klien mengirimkan kunci API, atur `value` ke `HEADER` dalam perintah CLI sebelumnya.

Untuk memilih sumber kunci API untuk API dengan menggunakan API Gateway REST API, panggil `restapi:update`sebagai berikut:

```
PATCH /restapis/fugvjdxtri/ HTTP/1.1
Content-Type: application/json
Host: apigateway.us-east-1.amazonaws.com
X-Amz-Date: 20160603T205348Z
Authorization: AWS4-HMAC-SHA256 Credential={access_key_ID}/20160603/us-east-1/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature={sig4_hash}

{
  "patchOperations" : [
    {
        "op" : "replace",
        "path" : "/apiKeySource",
        "value" : "HEADER"
    }
  ]
}
```

Agar otorisasi mengembalikan kunci API, setel `value` ke `AUTHORIZER` dalam `patchOperations` input sebelumnya.

Bergantung pada jenis sumber kunci API yang Anda pilih, gunakan salah satu prosedur berikut untuk menggunakan kunci API bersumber header atau kunci API yang dikembalikan oleh otorisasi dalam pemanggilan metode:

Untuk menggunakan kunci API bersumber header:

1. Buat API dengan metode API yang diinginkan, lalu terapkan API ke panggung.
2. Buat paket penggunaan baru atau pilih yang sudah ada. Tambahkan tahap API yang diterapkan ke paket penggunaan. Lampirkan kunci API ke paket penggunaan atau pilih kunci API yang ada dalam paket. Perhatikan nilai kunci API yang dipilih.
3. Siapkan metode API untuk memerlukan kunci API.
4. Menerapkan ulang API ke tahap yang sama. Jika Anda menerapkan API ke tahap baru, pastikan untuk memperbarui paket penggunaan untuk melampirkan tahap API baru.

Klien sekarang dapat memanggil metode API sambil memasok `x-api-key` header dengan kunci API yang dipilih sebagai nilai header.

Untuk menggunakan kunci API yang bersumber dari otorisasi:

1. Buat API dengan metode API yang diinginkan, lalu terapkan API ke panggung.
2. Buat paket penggunaan baru atau pilih yang sudah ada. Tambahkan tahap API yang diterapkan ke paket penggunaan. Lampirkan kunci API ke paket penggunaan atau pilih kunci API yang ada dalam paket. Perhatikan nilai kunci API yang dipilih.
3. Buat otorisasi Lambda berbasis token. Sertakan, `usageIdentifierKey:`*`{api-key}`* sebagai properti tingkat root dari respons otorisasi.
4. Siapkan metode API untuk memerlukan kunci API dan aktifkan otorisasi Lambda pada metode juga.
5. Menerapkan ulang API ke tahap yang sama. Jika Anda menerapkan API ke tahap baru, pastikan untuk memperbarui paket penggunaan untuk melampirkan tahap API baru.

Klien sekarang dapat memanggil metode yang diperlukan kunci API tanpa secara eksplisit memasok kunci API apa pun. Kunci API yang dikembalikan authorizer digunakan secara otomatis.

Mengatur kunci API menggunakan konsol API Gateway

Untuk menyiapkan kunci API, lakukan hal berikut:

- Konfigurasikan metode API untuk memerlukan kunci API.
- Membuat atau mengimpor kunci API untuk API di suatu wilayah.

Sebelum menyiapkan kunci API, Anda harus membuat API dan menerapkannya ke panggung. Setelah Anda membuat nilai kunci API, nilai tersebut tidak dapat diubah.

Untuk petunjuk tentang cara membuat dan menerapkan API menggunakan konsol API Gateway, lihat Membuat REST API di Amazon API Gateway danMenerapkan REST API di Amazon API Gateway, masing-masing.

Setelah membuat kunci API, Anda harus mengaitkannya dengan paket penggunaan. Untuk informasi selengkapnya, lihat Membuat, mengonfigurasi, dan menguji paket penggunaan dengan konsol API Gateway.

> **Note**
>
> Untuk praktik terbaik untuk dipertimbangkan, lihatthe section called “Praktik terbaik untuk kunci API dan paket penggunaan”.

Topik

- Memerlukan kunci API pada suatu metode
- Buat kunci API
- Impor kunci API

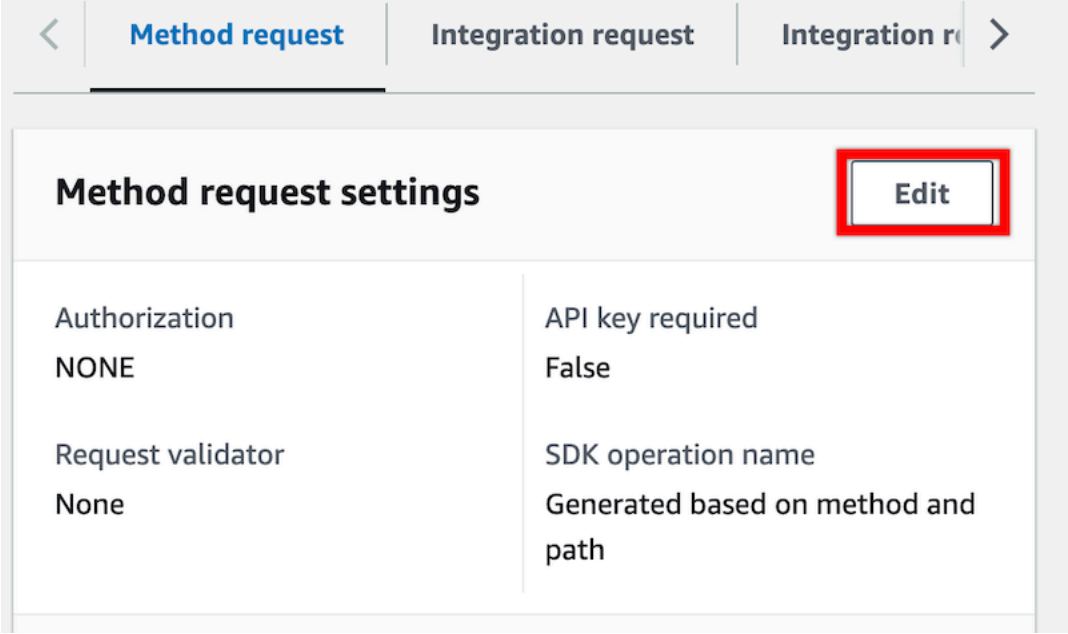
Memerlukan kunci API pada suatu metode

Prosedur berikut menjelaskan cara mengonfigurasi metode API agar memerlukan kunci API.

Untuk mengonfigurasi metode API agar memerlukan kunci API

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih REST API.

3. Di panel navigasi utama API Gateway, pilih Resources.
4. Di bawah Sumber Daya, buat metode baru atau pilih yang sudah ada.
5. Pada tab Permintaan metode, di bawah Pengaturan permintaan metode, pilih Edit.



6. Pilih kunci API yang diperlukan.
7. Pilih Simpan.
8. Terapkan atau terapkan ulang API agar persyaratan diterapkan.

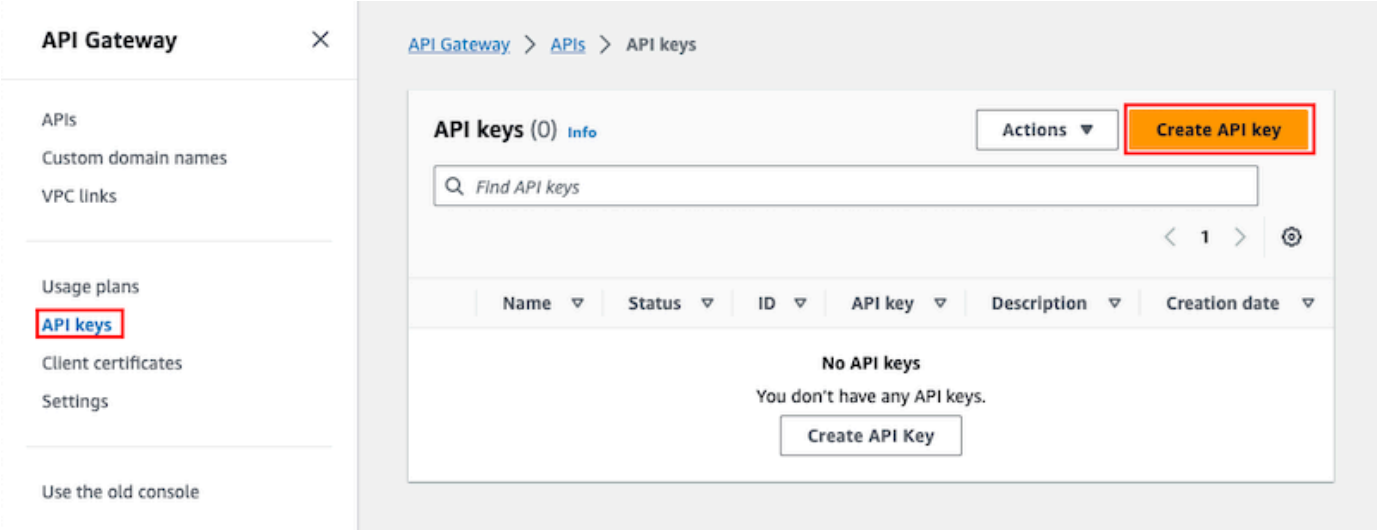
Jika opsi yang diperlukan kunci API disetel ke `false` dan Anda tidak menjalankan langkah sebelumnya, kunci API apa pun yang terkait dengan tahap API tidak akan digunakan untuk metode tersebut.

Buat kunci API

Jika Anda telah membuat atau mengimpor kunci API untuk digunakan dengan paket penggunaan, Anda dapat melewati ini dan prosedur berikutnya.

Untuk membuat kunci API

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih REST API.
3. Di panel navigasi utama API Gateway, pilih kunci API.
4. Pilih Buat kunci API.



5. Untuk Nama, masukkan nama.
6. (Opsional) Untuk Deskripsi, masukkan deskripsi.
7. Untuk kunci API, pilih Auto generate agar API Gateway menghasilkan nilai kunci, atau pilih Custom untuk membuat nilai kunci Anda sendiri.
8. Pilih Simpan.

Impor kunci API

Prosedur berikut menjelaskan cara mengimpor kunci API untuk digunakan dengan rencana penggunaan.

Untuk mengimpor kunci API

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih REST API.
3. Di panel navigasi utama, pilih kunci API.
4. Pilih menu tarik-turun Tindakan, lalu pilih Impor kunci API.
5. Untuk memuat file kunci yang dipisahkan koma, pilih Pilih file. Anda juga dapat memasukkan tombol di editor teks. Untuk informasi tentang format file, lihatthe section called “Format file kunci API Gateway API”.
6. Pilih Gagal pada peringatan untuk menghentikan impor ketika ada kesalahan, atau pilih Abaikan peringatan untuk terus mengimpor entri kunci yang valid saat ada peringatan.
7. Pilih Impor untuk mengimpor kunci API Anda.

Membuat, mengonfigurasi, dan menguji paket penggunaan dengan konsol API Gateway

Sebelum membuat rencana penggunaan, pastikan Anda telah menyiapkan kunci API yang diinginkan. Untuk informasi selengkapnya, lihat Mengatur kunci API menggunakan konsol API Gateway.

Bagian ini menjelaskan cara membuat dan menggunakan paket penggunaan menggunakan konsol API Gateway.

Topik

- Migrasi API Anda ke paket penggunaan default (jika diperlukan)
- Buat rencana penggunaan
- Uji rencana penggunaan
- Pertahankan rencana penggunaan

Migrasi API Anda ke paket penggunaan default (jika diperlukan)

Jika Anda mulai menggunakan API Gateway setelah fitur paket penggunaan diluncurkan pada 11 Agustus 2016, Anda akan secara otomatis mengaktifkan paket penggunaan untuk Anda di semua Wilayah yang didukung.

Jika Anda mulai menggunakan API Gateway sebelum tanggal tersebut, Anda mungkin perlu bermigrasi ke paket penggunaan default. Anda akan diminta dengan opsi Aktifkan Paket Penggunaan sebelum menggunakan paket penggunaan untuk pertama kalinya di Wilayah yang dipilih. Saat mengaktifkan opsi ini, Anda memiliki paket penggunaan default yang dibuat untuk setiap tahap API unik yang terkait dengan kunci API yang ada. Dalam paket penggunaan default, tidak ada batas throttle atau kuota yang ditetapkan pada awalnya, dan asosiasi antara kunci API dan tahapan API disalin ke paket penggunaan. API berperilaku sama seperti sebelumnya. Namun, Anda harus menggunakan `UsagePlan`apiStagesproperti untuk mengaitkan nilai tahap API tertentu (`apiId`dan`stage`) dengan kunci API yang disertakan (via `UsagePlanKey`), alih-alih menggunakan `ApiKey`stageKeysproperti.

Untuk memeriksa apakah Anda sudah bermigrasi ke paket penggunaan default, gunakan perintah `get-account`CLI. Dalam output perintah, `features` daftar menyertakan entri `"UsagePlans"` kapan rencana penggunaan diaktifkan.

Anda juga dapat memigrasikan API ke paket penggunaan default dengan menggunakan AWS CLI sebagai berikut:

Untuk bermigrasi ke paket penggunaan default menggunakan AWS CLI

1. Panggil perintah CLI ini:. `update-account`
2. Untuk `cli-input-json` parameter, gunakan JSON berikut:

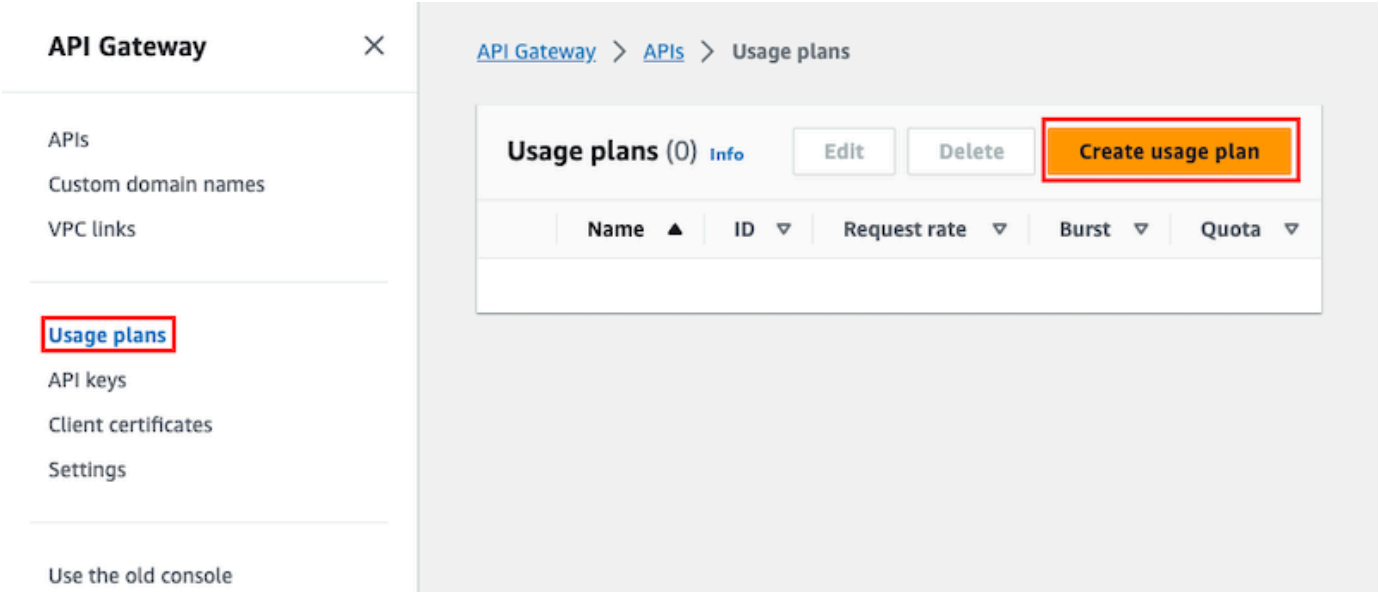
```
[
    {
        "op": "add",
        "path": "/features",
        "value": "UsagePlans"
    }
]
```

Buat rencana penggunaan

Prosedur berikut menjelaskan cara membuat rencana penggunaan.

Untuk membuat rencana penggunaan

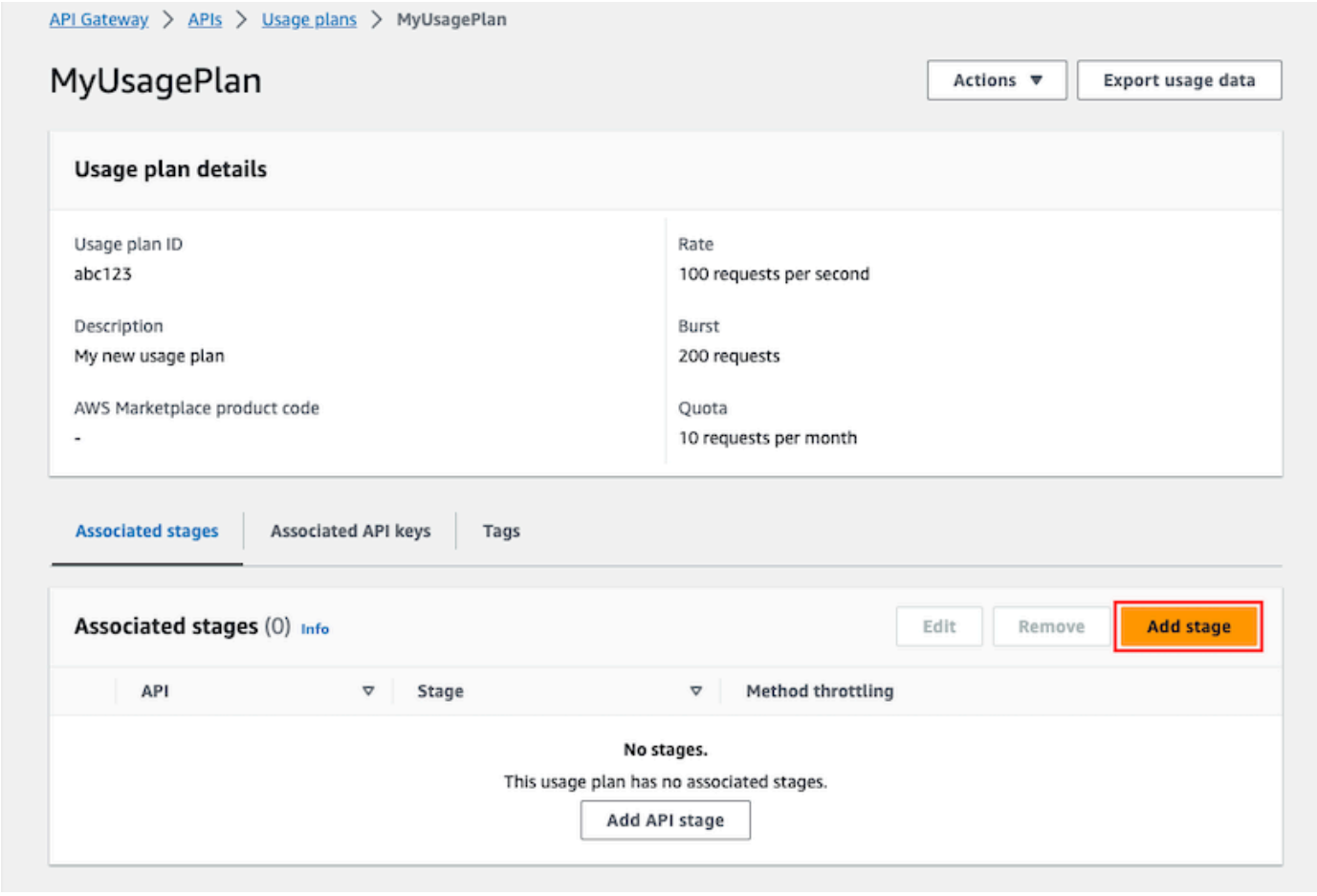
1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Di panel navigasi utama API Gateway, pilih Paket penggunaan, lalu pilih Buat paket penggunaan.



3. Untuk Nama, masukkan nama.
4. (Opsional) Untuk Deskripsi, masukkan deskripsi.
5. Secara default, paket penggunaan mengaktifkan pelambatan. Masukkan Rate dan Burst untuk paket penggunaan Anda. Pilih Throttling untuk mematikan throttling.
6. Secara default, paket penggunaan mengaktifkan kuota untuk jangka waktu tertentu. Untuk Permintaan, masukkan jumlah total permintaan yang dapat dibuat pengguna dalam periode waktu paket penggunaan Anda. Pilih Kuota untuk mematikan kuota.
7. Pilih Buat paket penggunaan.

Untuk menambahkan tahapan ke rencana penggunaan

1. Pilih paket penggunaan Anda.
2. Di bawah tab Tahapan terkait, pilih Tambahkan tahap.



3. Untuk API, pilih API.
4. Untuk Stage, pilih panggung.

5. (Opsional) Untuk mengaktifkan pelambatan tingkat metode, lakukan hal berikut:

   a. Pilih pelambatan tingkat metode, lalu pilih Tambahkan metode.

   b. Untuk Sumber Daya, pilih sumber daya dari API Anda.

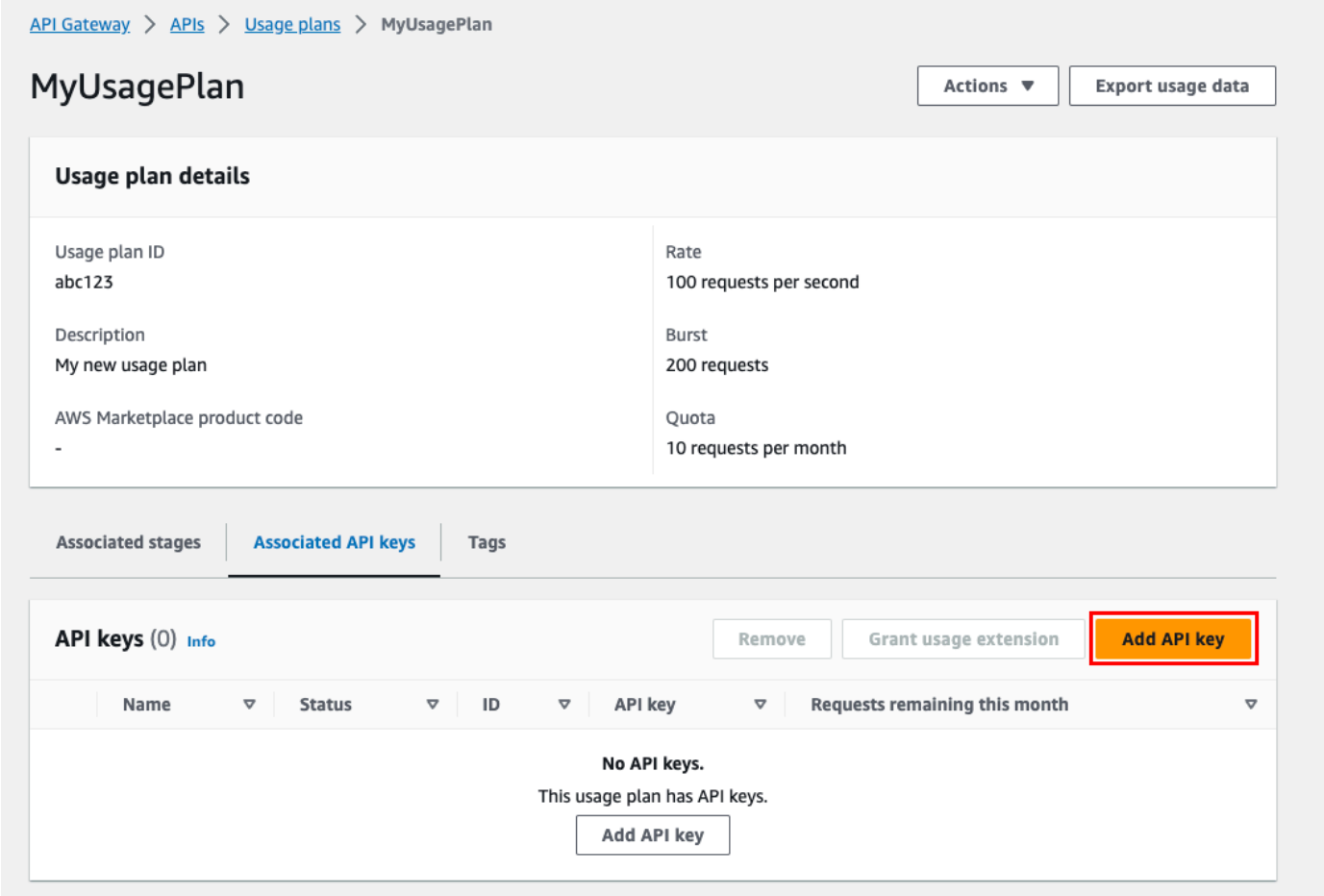
   c. Untuk Metode, pilih metode dari API Anda.

   d. Masukkan Rate dan Burst untuk paket penggunaan Anda.

6. Pilih Tambahkan ke paket penggunaan.

Untuk menambahkan kunci ke paket penggunaan

1. Di bawah tab Kunci API Terkait, pilih Tambahkan kunci API.



2. a. Untuk mengaitkan kunci yang ada ke paket penggunaan Anda, pilih Tambahkan kunci yang ada, lalu pilih kunci yang ada dari menu tarik-turun.

   b. Untuk membuat kunci API baru, pilih Buat dan tambahkan kunci baru, lalu buat kunci baru. Untuk informasi selengkapnya tentang cara membuat kunci baru, lihatBuat kunci API.

3. Pilih Tambahkan kunci API.

Uji rencana penggunaan

Untuk menguji paket penggunaan, Anda dapat menggunakan AWS SDK, AWS CLI, atau klien REST API seperti Postman. Untuk contoh menggunakan Postman untuk menguji rencana penggunaan, lihatUji rencana penggunaan.

Pertahankan rencana penggunaan

Mempertahankan rencana penggunaan melibatkan pemantauan kuota yang digunakan dan yang tersisa selama periode waktu tertentu dan, jika diperlukan, memperpanjang kuota yang tersisa dengan jumlah tertentu. Prosedur berikut menjelaskan cara memantau kuota.

Untuk memantau kuota yang digunakan dan yang tersisa

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Di panel navigasi utama API Gateway, pilih Paket penggunaan.
3. Pilih paket penggunaan.
4. Pilih tab Kunci API Terkait untuk melihat jumlah permintaan yang tersisa untuk periode waktu untuk setiap kunci.
5. (Opsional) Pilih Ekspor data penggunaan, lalu pilih Tanggal Dari dan Tanggal Sampai saat ini. Kemudian pilih JSON atau CSV untuk format data yang diekspor, lalu pilih Ekspor.

   Contoh berikut menunjukkan file yang diekspor.

```
{
  "thisPeriod": {
    "px1KW6...qBazOJH": [
      [
        0,
        5000
      ],
      [
        0,
        5000
      ],
      [
        0,
        10
```

```
      ]
    ]
  },
  "startDate": "2016-08-01",
  "endDate": "2016-08-03"
}
```

Data penggunaan dalam contoh menunjukkan data penggunaan harian untuk klien API, seperti yang diidentifikasi oleh kunci API (`px1KW6...qBaz0JH`), antara 1 Agustus 2016 dan 3 Agustus 2016. Setiap data penggunaan harian menunjukkan kuota yang digunakan dan yang tersisa. Dalam contoh ini, pelanggan belum menggunakan kuota yang dialokasikan, dan pemilik atau administrator API telah mengurangi sisa kuota dari 5000 menjadi 10 pada hari ketiga.

Prosedur berikut menjelaskan cara memodifikasi kuota.

Untuk memperpanjang kuota yang tersisa

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Di panel navigasi utama API Gateway, pilih Paket penggunaan.
3. Pilih paket penggunaan.
4. Pilih tab Kunci API Terkait untuk melihat jumlah permintaan yang tersisa untuk periode waktu untuk setiap kunci.
5. Pilih kunci API, lalu pilih Grant usage extension.
6. Masukkan nomor untuk kuota Permintaan yang tersisa. Anda dapat meningkatkan permintaan penggantian nama atau mengurangi permintaan yang tersisa untuk jangka waktu paket penggunaan Anda.
7. Pilih Perbarui kuota.

Mengatur kunci API menggunakan API Gateway REST API

Untuk menyiapkan kunci API, lakukan hal berikut:

- Konfigurasikan metode API untuk memerlukan kunci API.
- Membuat atau mengimpor kunci API untuk API di suatu wilayah.

Sebelum menyiapkan kunci API, Anda harus membuat API dan menerapkannya ke panggung. Setelah Anda membuat nilai kunci API, nilai tersebut tidak dapat diubah.

Untuk panggilan REST API untuk membuat dan menerapkan API, lihat `restapi:create`dan `deployment:create`, masing-masing.

> **Note**
>
> Untuk praktik terbaik untuk dipertimbangkan, lihatthe section called “Praktik terbaik untuk kunci API dan paket penggunaan”.

Topik

- Memerlukan kunci API pada suatu metode
- Membuat atau mengimpor kunci API

Memerlukan kunci API pada suatu metode

Untuk mewajibkan kunci API pada suatu metode, lakukan salah satu hal berikut:

- Panggilan `method:put`untuk membuat metode. Setel `apiKeyRequired` ke `true` dalam payload permintaan.
- Panggilan `method:update`untuk mengatur `apiKeyRequired` ke`true`.

Membuat atau mengimpor kunci API

Untuk membuat atau mengimpor kunci API, lakukan salah satu hal berikut:

- Panggilan `apikey:create`untuk membuat kunci API.
- Panggilan `apikey:import`untuk mengimpor kunci API dari file. Untuk format file, lihatFormat file kunci API Gateway API.

Anda tidak dapat mengubah nilai kunci API baru. Untuk mempelajari cara mengonfigurasi paket penggunaan, lihatMembuat, mengonfigurasi, dan menguji rencana penggunaan menggunakan API Gateway CLI dan REST API.

Membuat, mengonfigurasi, dan menguji rencana penggunaan menggunakan API Gateway CLI dan REST API

Sebelum mengonfigurasi paket penggunaan, Anda harus sudah melakukan hal berikut: menyiapkan metode API yang dipilih untuk meminta kunci API, menerapkan atau menerapkan ulang API ke tahap, dan membuat atau mengimpor satu atau beberapa kunci API. Untuk informasi selengkapnya, lihat Mengatur kunci API menggunakan API Gateway REST API.

Untuk mengonfigurasi paket penggunaan menggunakan API Gateway REST API, gunakan petunjuk berikut, dengan asumsi bahwa Anda telah membuat API yang akan ditambahkan ke paket penggunaan.

Topik

- Migrasi ke paket penggunaan default
- Buat rencana penggunaan
- Mengelola rencana penggunaan dengan menggunakan AWS CLI
- Uji rencana penggunaan

Migrasi ke paket penggunaan default

Saat membuat paket penggunaan pertama kali, Anda dapat memigrasikan tahapan API yang ada yang terkait dengan kunci API yang dipilih ke paket penggunaan dengan memanggil `account:update`dengan isi berikut:

```
{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/features",
    "value" : "UsagePlans"
  } ]
}
```

Untuk informasi selengkapnya tentang memigrasi tahapan API yang terkait dengan kunci API, lihat Memigrasi ke Paket Penggunaan Default di Konsol API Gateway.

Buat rencana penggunaan

Prosedur berikut menjelaskan cara membuat rencana penggunaan.

Untuk membuat rencana penggunaan dengan REST API

1. Hubungi `usageplan:create`untuk membuat paket penggunaan. Di payload, tentukan nama dan deskripsi paket, tahapan API terkait, batas tarif, dan kuota.

   Catat pengenal rencana penggunaan yang dihasilkan. Anda membutuhkannya di langkah berikutnya.

2. Lakukan salah satu dari cara berikut:

   a. Panggil `usageplankey:create`untuk menambahkan kunci API ke paket penggunaan. Tentukan `keyId` dan `keyType` di payload.

      Untuk menambahkan lebih banyak kunci API ke paket penggunaan, ulangi panggilan sebelumnya, satu kunci API pada satu waktu.

   b. Panggil `apikey:import`untuk menambahkan satu atau beberapa kunci API langsung ke paket penggunaan yang ditentukan. Payload permintaan harus berisi nilai kunci API, pengenal paket penggunaan terkait, flag Boolean untuk menunjukkan bahwa kunci diaktifkan untuk paket penggunaan, dan, mungkin, nama dan deskripsi kunci API.

      Contoh `apikey:import` permintaan berikut ini menambahkan tiga kunci API (seperti yang diidentifikasi oleh`key`,`name`, dan`description`) ke satu paket penggunaan (seperti yang diidentifikasi oleh`usageplanIds`):

      ```
      POST /apikeys?mode=import&format=csv&failonwarnings=fase HTTP/1.1
      Host: apigateway.us-east-1.amazonaws.com
      Content-Type: text/csv
      Authorization: ...

      key,name, description, enabled, usageplanIds
      abcdef1234ghijklmnop8901234567, importedKey_1, firstone,  tRuE, n371pt
      abcdef1234ghijklmnop0123456789, importedKey_2, secondone, TRUE, n371pt
      abcdef1234ghijklmnop9012345678, importedKey_3,          , true, n371pt
      ```

      Akibatnya, tiga `UsagePlanKey` sumber daya dibuat dan ditambahkan ke`UsagePlan`.

      Anda juga dapat menambahkan kunci API ke lebih dari satu paket penggunaan dengan cara ini. Untuk melakukan ini, ubah setiap nilai `usageplanIds` kolom menjadi string yang dipisahkan koma yang berisi pengidentifikasi rencana penggunaan yang dipilih, dan diapit dalam sepasang tanda kutip (atau). `"n371pt,m282qs"` `'n371pt,m282qs'`

> **Note**
>
> Kunci API dapat dikaitkan dengan lebih dari satu paket penggunaan. Rencana penggunaan dapat dikaitkan dengan lebih dari satu tahap. Namun, kunci API tertentu hanya dapat dikaitkan dengan satu paket penggunaan untuk setiap tahap API Anda.

Mengelola rencana penggunaan dengan menggunakan AWS CLI

Contoh kode berikut menunjukkan cara menambahkan, menghapus, atau memodifikasi pengaturan pelambatan tingkat metode dalam paket penggunaan dengan memanggil perintah. `update-usage-plan`

> **Note**
>
> Pastikan untuk mengubah `us-east-1` ke nilai Region yang sesuai untuk API Anda.

Untuk menambah atau mengganti batas tarif untuk membatasi sumber daya dan metode individual:

```
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"
```

Untuk menambah atau mengganti batas burst untuk membatasi sumber daya dan metode individual:

```
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"
```

Untuk menghapus pengaturan pelambatan tingkat metode untuk sumber daya dan metode individual:

```
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=""
```

Untuk menghapus semua setelan pelambatan tingkat metode untuk API:

```
aws apigateway --region us-east-1 update-usage-plan --usage-plan-id <planId> --patch-
operations op="remove",path="/apiStages/<apiId>:<stage>/throttle ",value=""
```

Berikut adalah contoh menggunakan contoh API Pet Store:

```
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}}"'
```

Uji rencana penggunaan

Sebagai contoh, mari kita gunakan PetStore API, yang dibuat di[Tutorial: Buat REST API dengan mengimpor contoh](). Asumsikan bahwa API dikonfigurasi untuk menggunakan kunci API dari`Hiorr45VR...c4GJc`. Langkah-langkah berikut menjelaskan cara menguji rencana penggunaan.

Untuk menguji paket penggunaan Anda

- Buat `GET` permintaan pada resource Pets (`/pets`), dengan parameter `?type=...&page=...` kueri, API (misalnya,`xbvxlpijch`) dalam paket penggunaan:

  ```
  GET /testStage/pets?type=dog&page=1 HTTP/1.1
  x-api-key: Hiorr45VR...c4GJc
  Content-Type: application/x-www-form-urlencoded
  Host: xbvxlpijch.execute-api.ap-southeast-1.amazonaws.com
  X-Amz-Date: 20160803T001845Z
  Authorization: AWS4-HMAC-SHA256 Credential={access_key_ID}/20160803/ap-southeast-1/
  execute-api/aws4_request, SignedHeaders=content-type;host;x-amz-date;x-api-key,
   Signature={sigv4_hash}
  ```

  > **Note**
  >
  > Anda harus mengirimkan permintaan ini ke `execute-api` komponen API Gateway dan memberikan kunci API yang diperlukan (misalnya,`Hiorr45VR...c4GJc`) di `x-api-key` header yang diperlukan.

Respons yang berhasil mengembalikan kode `200 OK` status dan payload yang berisi hasil yang diminta dari backend. Jika Anda lupa mengatur `x-api-key` header atau mengaturnya dengan kunci yang salah, Anda mendapatkan `403 Forbidden` respons. Namun, jika Anda tidak mengonfigurasi metode untuk memerlukan kunci API, kemungkinan besar Anda akan mendapatkan `200 OK` respons apakah Anda menyetel `x-api-key` header dengan benar atau tidak, dan batas throttle dan kuota paket penggunaan dilewati.

Kadang-kadang, ketika terjadi kesalahan internal di mana API Gateway tidak dapat menerapkan batas pembatasan rencana penggunaan atau kuota untuk permintaan, API Gateway melayani permintaan tanpa menerapkan batas pembatasan atau kuota seperti yang ditentukan dalam paket penggunaan. Tapi, itu mencatat pesan kesalahan `Usage Plan check failed due to an internal error` in CloudWatch. Anda dapat mengabaikan kesalahan sesekali seperti itu.

Membuat dan mengonfigurasi kunci API dan rencana penggunaan dengan AWS CloudFormation

Anda dapat menggunakan AWS CloudFormation untuk mewajibkan kunci API pada metode API dan membuat rencana penggunaan untuk API. Contoh AWS CloudFormation template melakukan hal berikut:

- Membuat API Gateway API dengan `GET` dan `POST` metode.
- Memerlukan kunci API untuk `GET` dan `POST` metode. API ini menerima kunci dari `X-API-KEY` header setiap permintaan yang masuk.
- Membuat kunci API.
- Membuat paket penggunaan untuk menentukan kuota bulanan 1.000 permintaan setiap bulan, batas laju pelambatan 100 permintaan setiap detik, dan batas burst throttling 200 permintaan setiap detik.
- Menentukan batas laju pelambatan tingkat metode 50 permintaan setiap detik dan batas burst throttling tingkat metode 100 permintaan per detik untuk metode tersebut. `GET`
- Mengaitkan tahap API dan kunci API dengan paket penggunaan.

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  StageName:
```

```
    Type: String
    Default: v1
    Description: Name of API stage.
  KeyName:
    Type: String
    Default: MyKeyName
    Description: Name of an API key
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Name: keys-api
      ApiKeySourceType: HEADER
  PetsResource:
    Type: 'AWS::ApiGateway::Resource'
    Properties:
      RestApiId: !Ref Api
      ParentId: !GetAtt Api.RootResourceId
      PathPart: 'pets'
  PetsMethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref PetsResource
      HttpMethod: GET
      ApiKeyRequired: true
      AuthorizationType: NONE
      Integration:
        Type: HTTP_PROXY
        IntegrationHttpMethod: GET
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
  PetsMethodPost:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref PetsResource
      HttpMethod: POST
      ApiKeyRequired: true
      AuthorizationType: NONE
      Integration:
        Type: HTTP_PROXY
        IntegrationHttpMethod: GET
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
  ApiDeployment:
```

```
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn:
      - PetsMethodGet
    Properties:
      RestApiId: !Ref Api
      StageName: !Sub '${StageName}'
  UsagePlan:
    Type: AWS::ApiGateway::UsagePlan
    DependsOn:
      - ApiDeployment
    Properties:
      Description: Example usage plan with a monthly quota of 1000 calls and method-
level throttling for /pets GET
      ApiStages:
        - ApiId: !Ref Api
          Stage: !Sub '${StageName}'
          Throttle:
            "/pets/GET":
              RateLimit: 50.0
              BurstLimit: 100
      Quota:
        Limit: 1000
        Period: MONTH
      Throttle:
        RateLimit: 100.0
        BurstLimit: 200
      UsagePlanName: "My Usage Plan"
  ApiKey:
    Type: AWS::ApiGateway::ApiKey
    Properties:
      Description: API Key
      Name: !Sub '${KeyName}'
      Enabled: True
  UsagePlanKey:
    Type: AWS::ApiGateway::UsagePlanKey
    Properties:
      KeyId: !Ref ApiKey
      KeyType: API_KEY
      UsagePlanId: !Ref UsagePlan
Outputs:
  ApiRootUrl:
    Description: Root Url of the API
    Value: !Sub 'https://${Api}.execute-api.${AWS::Region}.amazonaws.com/${StageName}'
```

Format file kunci API Gateway API

API Gateway dapat mengimpor kunci API dari file eksternal dengan format nilai dipisahkan koma (CSV), lalu mengaitkan kunci yang diimpor dengan satu atau beberapa paket penggunaan. File yang diimpor harus berisi `Key` kolom `Name` dan. Nama header kolom tidak peka huruf besar/kecil, dan kolom dapat dalam urutan apa pun, seperti yang ditunjukkan pada contoh berikut:

```
Key,name
apikey1234abcdefghij0123456789,MyFirstApiKey
```

`Key`Nilai harus antara 20 dan 128 karakter. `Name`Nilai tidak boleh melebihi 1024 karakter.

File kunci API juga dapat memiliki`Description`,`Enabled`, atau `UsagePlanIds` kolom, seperti yang ditunjukkan pada contoh berikut:

```
Name,key,description,Enabled,usageplanIds
MyFirstApiKey,apikey1234abcdefghij0123456789,An imported key,TRUE,c7y23b
```

Ketika kunci dikaitkan dengan lebih dari satu paket penggunaan, `UsagePlanIds` nilainya adalah string yang dipisahkan koma dari ID paket penggunaan, diapit dengan sepasang tanda kutip ganda atau tunggal, seperti yang ditunjukkan pada contoh berikut:

```
Enabled,Name,key,UsageplanIds
true,MyFirstApiKey,apikey1234abcdefghij0123456789,"c7y23b,glvrsr"
```

Kolom yang tidak dikenal diizinkan, tetapi diabaikan. Nilai default adalah string kosong atau nilai `true` Boolean.

Kunci API yang sama dapat diimpor beberapa kali, dengan versi terbaru menimpa yang sebelumnya. Dua kunci API identik jika memiliki `key` nilai yang sama.

> **Note**
>
> Untuk praktik terbaik untuk dipertimbangkan, lihatthe section called “Praktik terbaik untuk kunci API dan paket penggunaan”.

Mendokumentasikan REST API

Untuk membantu pelanggan memahami dan menggunakan API Anda, Anda harus mendokumentasikan API. Untuk membantu Anda mendokumentasikan API, API Gateway memungkinkan Anda menambahkan dan memperbarui konten bantuan untuk entitas API individual sebagai bagian integral dari proses pengembangan API Anda. API Gateway menyimpan konten sumber dan memungkinkan Anda mengarsipkan berbagai versi dokumentasi. Anda dapat mengaitkan versi dokumentasi dengan tahap API, mengekspor snapshot dokumentasi khusus tahap ke file OpenAPI eksternal, dan mendistribusikan file sebagai publikasi dokumentasi.

Untuk mendokumentasikan API Anda, Anda dapat memanggil API Gateway REST API, menggunakan salah satu AWSSDK atau AWS CLIs untuk API Gateway, atau menggunakan konsol API Gateway. Selain itu, Anda dapat mengimpor atau mengekspor bagian dokumentasi yang didefinisikan dalam file OpenAPI eksternal.

Untuk berbagi dokumentasi API dengan pengembang, Anda dapat menggunakan portal pengembang. Sebagai contoh, lihat Mengintegrasikan ReadMe dengan API Gateway untuk Menjaga Hub Pengembang Anda Tetap Terbaru di blog Jaringan AWS Mitra (APN).

Topik

- Representasi dokumentasi API di API Gateway
- Dokumentasikan API menggunakan konsol API Gateway
- Publikasikan dokumentasi API menggunakan konsol API Gateway
- Dokumentasikan API menggunakan API Gateway REST API
- Publikasikan dokumentasi API menggunakan API Gateway REST API
- Impor dokumentasi API
- Kontrol akses ke dokumentasi API

Representasi dokumentasi API di API Gateway

Dokumentasi API Gateway API terdiri dari bagian-bagian dokumentasi individual yang terkait dengan entitas API tertentu yang mencakup API, sumber daya, metode, permintaan, respons, parameter pesan (yaitu, jalur, kueri, header), serta otorisasi dan model.

Di API Gateway, bagian dokumentasi diwakili oleh DocumentationPartsumber daya. Dokumentasi API secara keseluruhan diwakili oleh DocumentationPartskoleksi.

Mendokumentasikan API melibatkan pembuatan `DocumentationPart` instance, menambahkannya ke `DocumentationParts` koleksi, dan memelihara versi bagian dokumentasi saat API Anda berkembang.

Topik

- Bagian dokumentasi
- Versi dokumentasi

Bagian dokumentasi

DocumentationPartSumber daya adalah objek JSON yang menyimpan konten dokumentasi yang berlaku untuk entitas API individual. `properties`Bidangnya berisi konten dokumentasi sebagai peta pasangan kunci-nilai. `location`Properti mengidentifikasi entitas API terkait.

Bentuk peta konten ditentukan oleh Anda, pengembang API. Nilai pasangan kunci-nilai dapat berupa string, angka, boolean, objek, atau array. Bentuk `location` objek tergantung pada jenis entitas yang ditargetkan.

`DocumentationPart`Sumber daya mendukung pewarisan konten: konten dokumentasi entitas API berlaku untuk turunan dari entitas API tersebut. Untuk informasi selengkapnya tentang definisi entitas turunan dan pewarisan konten, lihat Mewarisi Konten dari Entitas API dengan Spesifikasi Lebih Umum.

Lokasi bagian dokumentasi

Properti lokasi DocumentationPartinstance mengidentifikasi entitas API yang menerapkan konten terkait. Entitas API dapat berupa sumber API API Gateway REST API, seperti, Resource RestApi, Method, MethodResponse, Authorizer, atau Model. Entitas juga dapat berupa parameter pesan, seperti parameter jalur URL, parameter string kueri, parameter header permintaan atau respons, badan permintaan atau respons, atau kode status respons.

Untuk menentukan entitas API, tetapkan atribut type `location` objek menjadi salah satu dari`API`,`AUTHORIZER`,`MODEL`,`RESOURCE`,`METHOD`,`PATH_PARAMETER`,`QUERY_PARAMETER`,`REQUEST_HEADE` `REQUEST_BODY``RESPONSE`,`RESPONSE_HEADER`, atau`RESPONSE_BODY`.

Bergantung pada `type` entitas API, Anda dapat menentukan `location` atribut lain, termasuk metode, nama, jalur, dan StatusCode. Tidak semua atribut ini valid untuk entitas API tertentu. Misalnya,,`type`, `pathname`, dan `statusCode` merupakan atribut valid dari `RESPONSE` entitas; hanya `type` dan `path` merupakan atribut lokasi yang valid dari `RESOURCE` entitas. Merupakan kesalahan

untuk menyertakan bidang yang tidak valid di `location` a `DocumentationPart` untuk entitas API tertentu.

Tidak semua `location` bidang yang valid diperlukan. Misalnya, `type` adalah `location` bidang yang valid dan wajib dari semua entitas API. Namun,`method`,`path`, dan `statusCode` merupakan atribut yang valid tetapi tidak diperlukan untuk `RESPONSE` entitas. Ketika tidak ditentukan secara eksplisit, `location` bidang yang valid mengasumsikan nilai defaultnya. `path`Nilai defaultnya adalah`/`, yaitu sumber daya root dari API. Nilai default`method`, atau `statusCode` is`*`, yang berarti metode apa pun, atau nilai kode status, masing-masing.

Isi bagian dokumentasi

`properties`Nilai dikodekan sebagai string JSON. `properties`Nilai berisi informasi apa pun yang Anda pilih untuk memenuhi persyaratan dokumentasi Anda. Misalnya, berikut ini adalah peta konten yang valid:

```
{
  "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."
}
```

Meskipun API Gateway menerima string JSON yang valid sebagai peta konten, atribut konten diperlakukan sebagai dua kategori: atribut yang dapat dikenali oleh OpenAPI dan yang tidak bisa. Dalam contoh sebelumnya,, `infodescription`, dan `x-custom-info` diakui oleh OpenAPI sebagai objek OpenAPI standar, properti, atau ekstensi. Sebaliknya, tidak `my-info` sesuai dengan spesifikasi OpenAPI. API Gateway menyebarkan atribut konten yang sesuai dengan OpenAPI ke dalam definisi entitas API dari instance terkait. `DocumentationPart` API Gateway tidak menyebarkan atribut konten yang tidak sesuai ke dalam definisi entitas API.

Sebagai contoh lain, di sini `DocumentationPart` ditargetkan untuk `Resource` entitas:

```
{
    "location" : {
        "type" : "RESOURCE",
        "path": "/pets"
    },
    "properties" : {
        "summary" : "The /pets resource represents a collection of pets in PetStore.",
```

```
        "description": "... a child resource under the root...",
    }
}
```

Di sini, `path` keduanya `type` dan merupakan bidang yang valid untuk mengidentifikasi target `RESOURCE` jenis. Untuk sumber daya root (`/`), Anda dapat menghilangkan `path` bidang.

```
{
    "location" : {
        "type" : "RESOURCE"
    },
    "properties" : {
        "description" : "The root resource with the default path specification."
    }
}
```

Ini sama dengan `DocumentationPart` contoh berikut:

```
{
    "location" : {
        "type" : "RESOURCE",
        "path": "/"
    },
    "properties" : {
        "description" : "The root resource with an explicit path specification"
    }
}
```

Mewarisi konten dari entitas API dengan spesifikasi yang lebih umum

Nilai default `location` bidang opsional memberikan deskripsi berpola entitas API. Menggunakan nilai default dalam `location` objek, Anda dapat menambahkan deskripsi umum di `properties` peta ke `DocumentationPart` instance dengan jenis `location` pola ini. API Gateway mengekstrak atribut dokumentasi OpenAPI yang berlaku dari entitas API generik dan menyuntikkannya ke entitas API tertentu dengan `location` bidang yang cocok dengan `location` pola umum, atau mencocokkan nilai persisnya, kecuali entitas tertentu sudah memiliki `DocumentationPart` instance yang terkait dengannya. `DocumentationPart` Perilaku ini juga dikenal sebagai pewarisan konten dari entitas API dengan spesifikasi yang lebih umum.

Warisan konten tidak berlaku untuk tipe entitas API tertentu. Lihat tabel di bawah ini untuk detailnya.

Jika entitas API cocok dengan lebih dari pola lokasi `DocumentationPart` seseorang, entitas akan mewarisi bagian dokumentasi dengan bidang lokasi dengan prioritas dan kekhususan tertinggi. Urutan prioritas adalah `path` >. `statusCode` Untuk mencocokkan dengan `path` bidang, API Gateway memilih entitas dengan nilai jalur paling spesifik. Tabel berikut menunjukkan ini dengan beberapa contoh.

Kasu	**path**	**statusCo e**	**name**	Ketera n
1	`/pets`	`*`	`id`	Dokumenta si yang terkait dengan pola lokasi ini akan diwarisi oleh entitas yang cocok dengan pola lokasi.
2	`/pets`	`200`	`id`	Dokumenta si yang terkait dengan pola lokasi ini akan diwarisi

Kasu	path	statusCode	name	Keterangan
				oleh entitas yang cocok dengan pola lokasi ketika Kasus 1 dan Kasus 2 dicocokkan, karena Kasus 2 lebih spesifik daripada Kasus 1.

Kasu	path	statusCo e	name	Ketera n
3	/pets/ petId	*	id	Dokumenta si yang terkait dengan pola lokasi ini akan diwarisi oleh entitas yang cocok dengan pola lokasi ketika Kasus 1, 2, dan 3 dicocokka n, karena Kasus 3 memiliki prioritas yang lebih tinggi daripada

Kasu	path	statusCo e	name	Ketera n	
				Kasus 2 dan lebih spesifik daripada Kasus 1.	

Berikut adalah contoh lain untuk membandingkan `DocumentationPart` contoh yang lebih umum dengan yang lebih spesifik. Pesan kesalahan umum berikut `"Invalid request error"` ini disuntikkan ke dalam definisi OpenAPI dari respons `400` kesalahan, kecuali diganti.

```
{
    "location" : {
        "type" : "RESPONSE",
        "statusCode": "400"
    },
    "properties" : {
        "description" : "Invalid request error."
    }"
}
```

Dengan penimpaan berikut, `400` tanggapan terhadap metode apa pun pada `/pets` sumber daya memiliki deskripsi sebagai `"Invalid petId specified"` gantinya.

```
{
    "location" : {
        "type" : "RESPONSE",
        "path": "/pets",
        "statusCode": "400"
    },
    "properties" : "{
        "description" : "Invalid petId specified."
    }"
}
```

Bidang lokasi yang valid dari **`DocumentationPart`**

Tabel berikut menunjukkan bidang yang valid dan wajib serta nilai default yang berlaku dari DocumentationPartsumber daya yang dikaitkan dengan jenis entitas API tertentu.

Entitas API	Bidang lokasi yang valid	Bidang lokasi yang diperlukan	Nilai bidang default	Konten yang dapat diwariskan
API	`{ "location": { "type": "API" }, ... }`	`type`	T/A	Tidak
Sumber	`{ "location": { "type": "RESOURCE", "path": "resource_path " }, ... }`	`type`	Nilai default `path` adalah `/`.	Tidak
Metode	`{ "location": { "type": "METHOD", "path": "resource_path ", "method": "http_verb " }, ... }`	`type`	Nilai default `path` dan `method` adalah `/` dan`*`, masing-masing.	Ya, cocok `path` dengan awalan dan pencocokan `method` nilai apa pun.
Parameter kueri	`{`	`type`	Nilai default `path` dan `method`	Ya, mencocokk

Entitas API	Bidang lokasi yang valid	Bidang lokasi yang diperlukan	Nilai bidang default	Konten yang dapat diwariskan

	<pre>{
	"location": {
	"type": "QUERY_PARAMETER",
	"path": "*resource_path* ",
	"method": "*HTTP_verb* ",
	"name": "*query_parameter_name* "
	},
	...
	}</pre>
			adalah / dan*, masing-masing.	an path dengan awalan dan mencocokkan method dengan nilai yang tepat.

Isi permintaan	<pre>{
	"location": {
	"type": "REQUEST_BODY",
	"path": "*resource_path* ",
	"method": "*http_verb* "
	},
	...
	}</pre>
		type	Nilai default daripath, dan method are / dan*, masing-masing.	Ya, mencocokkan path dengan awalan, dan mencocokkan method dengan nilai yang tepat.

Entitas API	Bidang lokasi yang valid	Bidang lokasi yang diperlukan	Nilai bidang default	Konten yang dapat diwariskan
Permintaan parameter header	`{ "location": { "type": "REQUEST_HEADER", "path": "resource_path ", "method": "HTTP_verb ", "name": "header_name " }, ... }`	`type`, `name`	Nilai default `path` dan `method` adalah `/` dan`*`, masing-masing.	Ya, mencocokkan `path` dengan awalan dan mencocokkan `method` dengan nilai yang tepat.
Parameter jalur permintaan	`{ "location": { "type": "PATH_PARAMETER", "path": "resource/{path_parameter_name }", "method": "HTTP_verb ", "name": "path_parameter_name " }, ... }`	`type`, `name`	Nilai default `path` dan `method` adalah `/` dan`*`, masing-masing.	Ya, mencocokkan `path` dengan awalan dan mencocokkan `method` dengan nilai yang tepat.

Entitas API	Bidang lokasi yang valid	Bidang lokasi yang diperlukan	Nilai bidang default	Konten yang dapat diwariskan
Response	`{ "location": { "type": "RESPONSE", "path": "resource_path ", "method": "http_verb ", "statusCode": "status_code " }, ... }`	`type`	Nilai default dari`path`,`method`, dan `statusCode` adalah`/`, `*` dan`*`, masing-masing.	Ya, cocok `path` dengan awalan dan pencocokan `method` dan `statusCode` dengan nilai yang tepat.
Header respons	`{ "location": { "type": "RESPONSE_HEADER", "path": "resource_path ", "method": "http_verb ", "statusCode": "status_code ", "name": "header_name " }, ... }`	`type`, `name`	Nilai default dari`path`, `method` dan `statusCode` are`/`, `*` dan`*`, masing-masing.	Ya, cocok `path` dengan awalan dan pencocokan`method`, dan `statusCode` dengan nilai yang tepat.

Entitas API	Bidang lokasi yang valid	Bidang lokasi yang diperlukan	Nilai bidang default	Konten yang dapat diwariskan
Isi respons	`{ "location": { "type": "RESPONSE_BODY", "path": "resource_path ", "method": "http_verb ", "statusCode": "status_code " }, ... }`	`type`	Nilai default dari`path`, `method` dan `statusCode` are`/`, `*` dan`*`, masing-masing.	Ya, cocok `path` dengan awalan dan pencocokan`method`, dan `statusCode` dengan nilai yang tepat.
Pengotorisasi	`{ "location": { "type": "AUTHORIZER", "name": "authorizer_name " }, ... }`	`type`	T/A	Tidak
Model	`{ "location": { "type": "MODEL", "name": "model_name " }, ... }`	`type`	T/A	Tidak

Versi dokumentasi

Versi dokumentasi adalah snapshot dari DocumentationPartskoleksi API dan ditandai dengan pengenal versi. Menerbitkan dokumentasi API melibatkan pembuatan versi dokumentasi, mengaitkannya dengan tahap API, dan mengekspor versi spesifik tahap dokumentasi API ke file OpenAPI eksternal. Di API Gateway, snapshot dokumentasi direpresentasikan sebagai DocumentationVersionsumber daya.

Saat memperbarui API, Anda membuat versi API yang baru. Di API Gateway, Anda memelihara semua versi dokumentasi menggunakan DocumentationVersionskoleksi.

Dokumentasikan API menggunakan konsol API Gateway

Di bagian ini, kami menjelaskan cara membuat dan memelihara bagian dokumentasi API menggunakan konsol API Gateway.

Prasyarat untuk membuat dan mengedit dokumentasi API adalah Anda harus sudah membuat API. Pada bagian ini, kita menggunakan PetStoreAPI sebagai contoh. Untuk membuat API menggunakan konsol API Gateway, ikuti petunjuk diTutorial: Buat REST API dengan mengimpor contoh.

Topik

- Dokumentasikan API entitas
- Dokumentasikan RESOURCE entitas
- Dokumentasikan METHOD entitas
- Dokumentasikan QUERY_PARAMETER entitas
- Dokumentasikan PATH_PARAMETER entitas
- Dokumentasikan REQUEST_HEADER entitas
- Dokumentasikan REQUEST_BODY entitas
- Dokumentasikan RESPONSE entitas
- Dokumentasikan RESPONSE_HEADER entitas
- Dokumentasikan RESPONSE_BODY entitas
- Dokumentasikan MODEL entitas
- Dokumentasikan AUTHORIZER entitas

Dokumentasikan **`API`** entitas

Untuk menambahkan bagian dokumentasi baru untuk `API` entitas, lakukan hal berikut:

1. Di panel navigasi utama, pilih Dokumentasi, lalu pilih Buat bagian dokumentasi.
2. Untuk jenis Dokumentasi, pilih API.

   Jika bagian dokumentasi tidak dibuat untukAPI, Anda mendapatkan editor `properties` peta bagian dokumentasi. Masukkan `properties` peta berikut di editor teks.

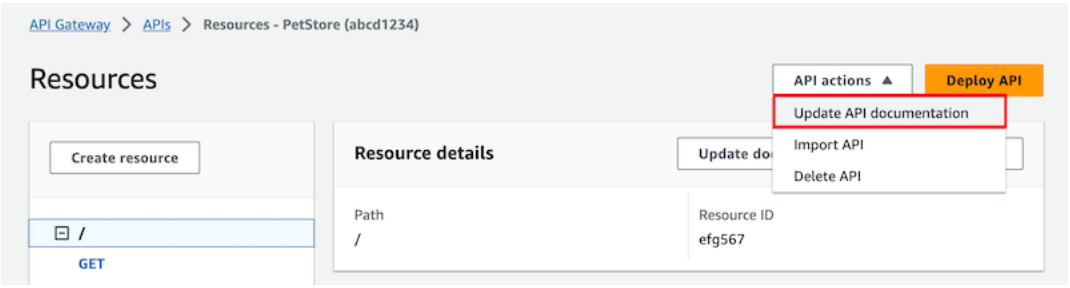
```
{
  "info": {
    "description": "Your first API Gateway API.",
    "contact": {
        "name": "John Doe",
        "email": "john.doe@api.com"
    }
  }
}
```

> **Note**
>
> Anda tidak perlu menyandikan `properties` peta menjadi string JSON. Konsol API Gateway membuat stringifikasi objek JSON untuk Anda.

3. Pilih Buat bagian dokumentasi.

Untuk menambahkan bagian dokumentasi baru untuk `API` entitas di panel Resources, lakukan hal berikut:

1. Di panel navigasi utama, pilih Resources.
2. Pilih menu tindakan API, lalu pilih Perbarui dokumentasi API.



Untuk mengedit bagian dokumentasi yang ada, lakukan hal berikut:

1. Di panel Dokumentasi, pilih tab Sumber dan metode.
2. Pilih nama API Anda, lalu pada kartu API, pilih Edit.

Dokumentasikan **RESOURCE** entitas

Untuk menambahkan bagian dokumentasi baru untuk RESOURCE entitas, lakukan hal berikut:

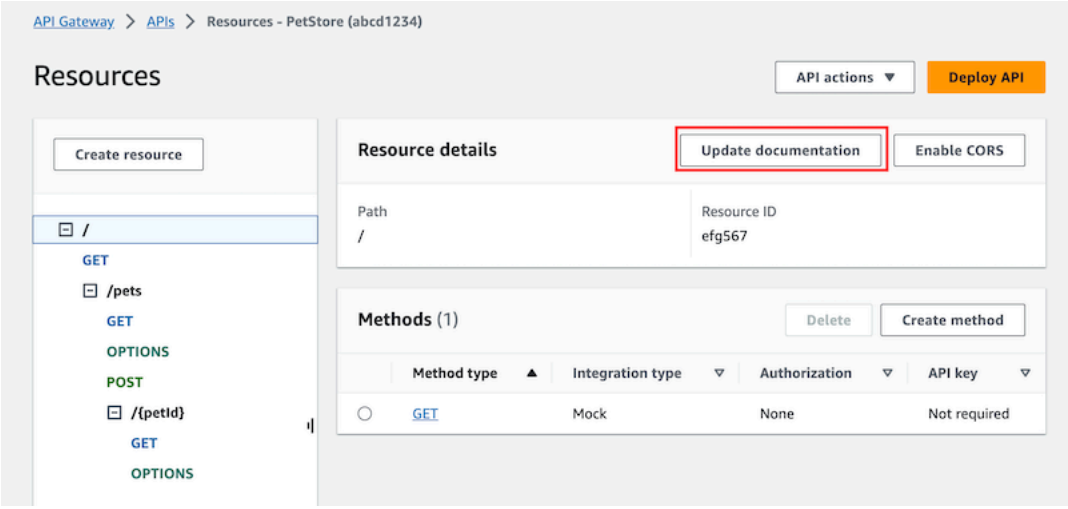
1. Di panel navigasi utama, pilih Dokumentasi, lalu pilih Buat bagian dokumentasi.
2. Untuk jenis Dokumentasi, pilih Sumber Daya.
3. Untuk Path, masukkan jalan.
4. Masukkan deskripsi di editor teks, misalnya:

```
{
    "description": "The PetStore's root resource."
}
```

5. Pilih Buat bagian dokumentasi. Anda dapat membuat dokumentasi untuk sumber daya yang tidak terdaftar.
6. Jika diperlukan, ulangi langkah-langkah ini untuk menambah atau mengedit bagian dokumentasi lain.

Untuk menambahkan bagian dokumentasi baru untuk RESOURCE entitas di panel Resources, lakukan hal berikut:

1. Di panel navigasi utama, pilih Resources.
2. Pilih sumber daya, lalu pilih Perbarui dokumentasi.



Untuk mengedit bagian dokumentasi yang ada, lakukan hal berikut:

1. Di panel Dokumentasi, pilih tab Sumber dan metode.
2. Pilih sumber daya yang berisi bagian dokumentasi Anda, lalu pilih Edit.

Dokumentasikan **`METHOD`** entitas

Untuk menambahkan bagian dokumentasi baru untuk `METHOD` entitas, lakukan hal berikut:

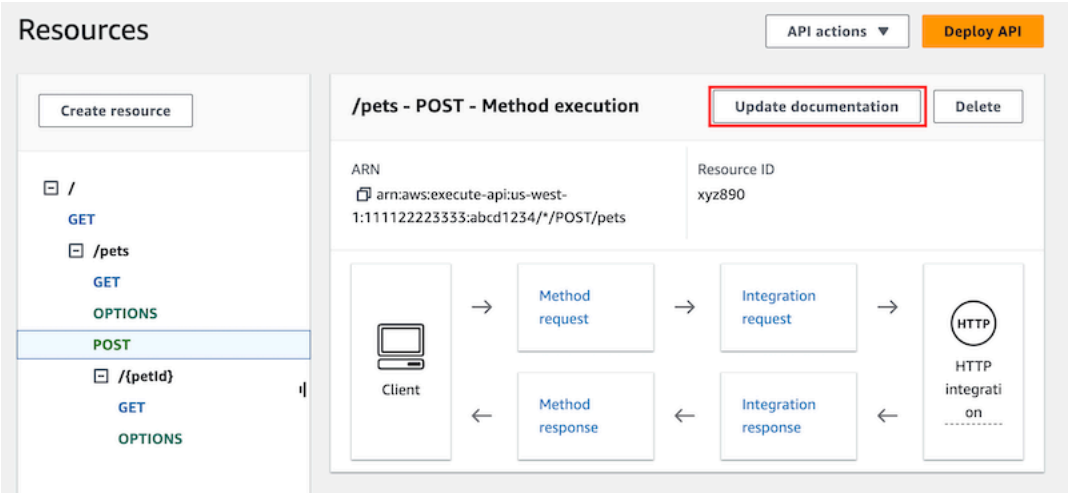
1. Di panel navigasi utama, pilih Dokumentasi, lalu pilih Buat bagian dokumentasi.
2. Untuk jenis Dokumentasi, pilih Metode.
3. Untuk Path, masukkan jalan.
4. Untuk Metode, pilih kata kerja HTTP.
5. Masukkan deskripsi di editor teks, misalnya:

```
{
  "tags" : [ "pets" ],
  "summary" : "List all pets"
}
```

6. Pilih Buat bagian dokumentasi. Anda dapat membuat dokumentasi untuk metode yang tidak terdaftar.
7. Jika diperlukan, ulangi langkah-langkah ini untuk menambah atau mengedit bagian dokumentasi lain.

Untuk menambahkan bagian dokumentasi baru untuk `METHOD` entitas di panel Resources, lakukan hal berikut:

1. Di panel navigasi utama, pilih Resources.
2. Pilih metode, dan kemudian pilih Perbarui dokumentasi.



Untuk mengedit bagian dokumentasi yang ada, lakukan hal berikut:

1. Di panel Dokumentasi, pilih tab Sumber dan metode.
2. Anda dapat memilih metode atau memilih sumber daya yang berisi metode, dan kemudian menggunakan bilah pencarian untuk menemukan dan memilih bagian dokumentasi Anda.
3. Pilih Edit.

Dokumentasikan **`QUERY_PARAMETER`** entitas

Untuk menambahkan bagian dokumentasi baru untuk `QUERY_PARAMETER` entitas, lakukan hal berikut:

1. Di panel navigasi utama, pilih Dokumentasi, lalu pilih Buat bagian dokumentasi.
2. Untuk jenis Dokumentasi, pilih Parameter kueri.
3. Untuk Path, masukkan jalan.
4. Untuk Metode, pilih kata kerja HTTP.
5. Untuk Nama, masukkan nama.
6. Masukkan deskripsi di editor teks.

7. Pilih Buat bagian dokumentasi. Anda dapat membuat dokumentasi untuk parameter kueri yang tidak terdaftar.
8. Jika diperlukan, ulangi langkah-langkah ini untuk menambah atau mengedit bagian dokumentasi lain.

Untuk mengedit bagian dokumentasi yang ada, lakukan hal berikut:

1. Di panel Dokumentasi, pilih tab Sumber dan metode.
2. Anda dapat memilih parameter kueri atau memilih sumber daya yang berisi parameter kueri, dan kemudian menggunakan bilah pencarian untuk menemukan dan memilih bagian dokumentasi Anda.
3. Pilih Edit.

Dokumentasikan **`PATH_PARAMETER`** entitas

Untuk menambahkan bagian dokumentasi baru untuk `PATH_PARAMETER` entitas, lakukan hal berikut:

1. Di panel navigasi utama, pilih Dokumentasi, lalu pilih Buat bagian dokumentasi.
2. Untuk jenis Dokumentasi, pilih parameter Path.
3. Untuk Path, masukkan jalan.
4. Untuk Metode, pilih kata kerja HTTP.
5. Untuk Nama, masukkan nama.
6. Masukkan deskripsi di editor teks.
7. Pilih Buat bagian dokumentasi. Anda dapat membuat dokumentasi untuk parameter jalur yang tidak terdaftar.
8. Jika diperlukan, ulangi langkah-langkah ini untuk menambah atau mengedit bagian dokumentasi lain.

Untuk mengedit bagian dokumentasi yang ada, lakukan hal berikut:

1. Di panel Dokumentasi, pilih tab Sumber dan metode.
2. Anda dapat memilih parameter jalur atau memilih sumber daya yang berisi parameter jalur, lalu menggunakan bilah pencarian untuk menemukan dan memilih bagian dokumentasi Anda.
3. Pilih Edit.

Dokumentasikan **REQUEST_HEADER** entitas

Untuk menambahkan bagian dokumentasi baru untuk REQUEST_HEADER entitas, lakukan hal berikut:

1. Di panel navigasi utama, pilih Dokumentasi, lalu pilih Buat bagian dokumentasi.
2. Untuk jenis Dokumentasi, pilih Request header.
3. Untuk Path, masukkan jalur untuk header permintaan.
4. Untuk Metode, pilih kata kerja HTTP.
5. Untuk Nama, masukkan nama.
6. Masukkan deskripsi di editor teks.
7. Pilih Buat bagian dokumentasi. Anda dapat membuat dokumentasi untuk header permintaan yang tidak terdaftar.
8. Jika diperlukan, ulangi langkah-langkah ini untuk menambah atau mengedit bagian dokumentasi lain.

Untuk mengedit bagian dokumentasi yang ada, lakukan hal berikut:

1. Di panel Dokumentasi, pilih tab Sumber dan metode.
2. Anda dapat memilih header permintaan atau memilih sumber daya yang berisi header permintaan, dan kemudian menggunakan bilah pencarian untuk menemukan dan memilih bagian dokumentasi Anda.
3. Pilih Edit.

Dokumentasikan **REQUEST_BODY** entitas

Untuk menambahkan bagian dokumentasi baru untuk REQUEST_BODY entitas, lakukan hal berikut:

1. Di panel navigasi utama, pilih Dokumentasi, lalu pilih Buat bagian dokumentasi.
2. Untuk jenis Dokumentasi, pilih Request body.
3. Untuk Path, masukkan jalur untuk badan permintaan.
4. Untuk Metode, pilih kata kerja HTTP.
5. Masukkan deskripsi di editor teks.
6. Pilih Buat bagian dokumentasi. Anda dapat membuat dokumentasi untuk badan permintaan yang tidak terdaftar.

7. Jika diperlukan, ulangi langkah-langkah ini untuk menambah atau mengedit bagian dokumentasi lain.

Untuk mengedit bagian dokumentasi yang ada, lakukan hal berikut:

1. Di panel Dokumentasi, pilih tab Sumber dan metode.
2. Anda dapat memilih badan permintaan atau memilih sumber daya yang berisi badan permintaan, lalu menggunakan bilah pencarian untuk menemukan dan memilih bagian dokumentasi Anda.
3. Pilih Edit.

Dokumentasikan **`RESPONSE`** entitas

Untuk menambahkan bagian dokumentasi baru untuk `RESPONSE` entitas, lakukan hal berikut:

1. Di panel navigasi utama, pilih Dokumentasi, lalu pilih Buat bagian dokumentasi.
2. Untuk jenis Dokumentasi, pilih Respons (kode status).
3. Untuk Path, masukkan jalur untuk respons.
4. Untuk Metode, pilih kata kerja HTTP.
5. Untuk kode Status, masukkan kode status HTTP.
6. Masukkan deskripsi di editor teks.
7. Pilih Buat bagian dokumentasi. Anda dapat membuat dokumentasi untuk kode status respons yang tidak terdaftar.
8. Jika diperlukan, ulangi langkah-langkah ini untuk menambah atau mengedit bagian dokumentasi lain.

Untuk mengedit bagian dokumentasi yang ada, lakukan hal berikut:

1. Di panel Dokumentasi, pilih tab Sumber dan metode.
2. Anda dapat memilih kode status respons atau memilih sumber daya yang berisi kode status respons, lalu gunakan bilah pencarian untuk menemukan dan memilih bagian dokumentasi Anda.
3. Pilih Edit.

Dokumentasikan **RESPONSE_HEADER** entitas

Untuk menambahkan bagian dokumentasi baru untuk RESPONSE_HEADER entitas, lakukan hal berikut:

1. Di panel navigasi utama, pilih Dokumentasi, lalu pilih Buat bagian dokumentasi.
2. Untuk jenis Dokumentasi, pilih header Response.
3. Untuk Path, masukkan jalur untuk header respons.
4. Untuk Metode, pilih kata kerja HTTP.
5. Untuk kode Status, masukkan kode status HTTP.
6. Masukkan deskripsi di editor teks.
7. Pilih Buat bagian dokumentasi. Anda dapat membuat dokumentasi untuk header respons yang tidak terdaftar.
8. Jika diperlukan, ulangi langkah-langkah ini untuk menambah atau mengedit bagian dokumentasi lain.

Untuk mengedit bagian dokumentasi yang ada, lakukan hal berikut:

1. Di panel Dokumentasi, pilih tab Sumber dan metode.
2. Anda dapat memilih header respons atau memilih sumber daya yang berisi header respons, lalu gunakan bilah pencarian untuk menemukan dan memilih bagian dokumentasi Anda.
3. Pilih Edit.

Dokumentasikan **RESPONSE_BODY** entitas

Untuk menambahkan bagian dokumentasi baru untuk RESPONSE_BODY entitas, lakukan hal berikut:

1. Di panel navigasi utama, pilih Dokumentasi, lalu pilih Buat bagian dokumentasi.
2. Untuk jenis Dokumentasi, pilih Badan respons.
3. Untuk Path, masukkan jalur untuk badan respons.
4. Untuk Metode, pilih kata kerja HTTP.
5. Untuk kode Status, masukkan kode status HTTP.
6. Masukkan deskripsi di editor teks.
7. Pilih Buat bagian dokumentasi. Anda dapat membuat dokumentasi untuk badan respons yang tidak terdaftar.

8. Jika diperlukan, ulangi langkah-langkah ini untuk menambah atau mengedit bagian dokumentasi lain.

Untuk mengedit bagian dokumentasi yang ada, lakukan hal berikut:

1. Di panel Dokumentasi, pilih tab Sumber dan metode.
2. Anda dapat memilih badan respons atau memilih sumber daya yang berisi badan respons, lalu menggunakan bilah pencarian untuk menemukan dan memilih bagian dokumentasi Anda.
3. Pilih Edit.

Dokumentasikan **`MODEL`** entitas

Mendokumentasikan `MODEL` entitas melibatkan pembuatan dan pengelolaan `DocumentPart` instance untuk model dan masing-masing model '. `properties` Misalnya, untuk `Error` model yang disertakan dengan setiap API secara default memiliki definisi skema berikut,

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

dan membutuhkan dua `DocumentationPart` contoh, satu untuk `Model` dan yang lainnya untuk `message` propertinya:

```
{
  "location": {
    "type": "MODEL",
    "name": "Error"
  },
  "properties": {
    "title": "Error Schema",
    "description": "A description of the Error model"
  }
}
```

and

```
{
  "location": {
    "type": "MODEL",
    "name": "Error.message"
  },
  "properties": {
    "description": "An error message."
  }
}
```

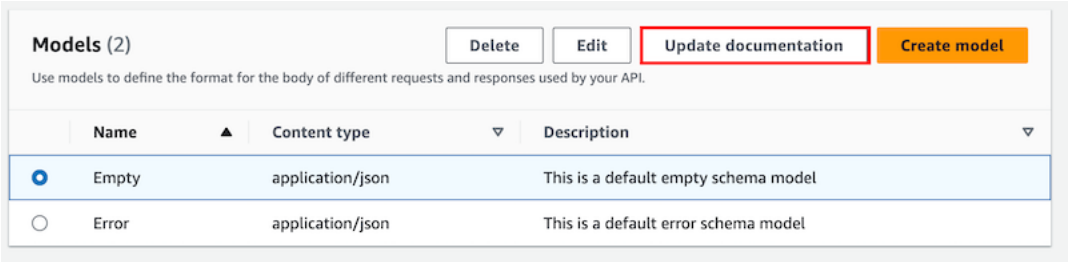
Saat API diekspor, properti akan mengganti nilai dalam skema asli. `DocumentationPart`

Untuk menambahkan bagian dokumentasi baru untuk `MODEL` entitas, lakukan hal berikut:

1. Di panel navigasi utama, pilih Dokumentasi, lalu pilih Buat bagian dokumentasi.
2. Untuk jenis Dokumentasi, pilih Model.
3. Untuk Nama, masukkan nama untuk model.
4. Masukkan deskripsi di editor teks.
5. Pilih Buat bagian dokumentasi. Anda dapat membuat dokumentasi untuk model yang tidak terdaftar.
6. Jika diperlukan, ulangi langkah-langkah ini untuk menambah atau mengedit bagian dokumentasi ke model lain.

Untuk menambahkan bagian dokumentasi baru untuk `MODEL` entitas di panel Model, lakukan hal berikut:

1. Di panel navigasi utama, pilih Model.
2. Pilih model, lalu pilih Perbarui dokumentasi.



Untuk mengedit bagian dokumentasi yang ada, lakukan hal berikut:

1. Di panel Dokumentasi, pilih tab Model.
2. Gunakan bilah pencarian atau pilih model, lalu pilih Edit.

Dokumentasikan **`AUTHORIZER`** entitas

Untuk menambahkan bagian dokumentasi baru untuk `AUTHORIZER` entitas, lakukan hal berikut:

1. Di panel navigasi utama, pilih Dokumentasi, lalu pilih Buat bagian dokumentasi.
2. Untuk jenis Dokumentasi, pilih Authorizer.
3. Untuk Nama, masukkan nama otorisasi Anda.
4. Masukkan deskripsi di editor teks. Tentukan nilai untuk `location` bidang yang valid untuk otorisasi.
5. Pilih Buat bagian dokumentasi. Anda dapat membuat dokumentasi untuk otorisasi yang tidak terdaftar.
6. Jika diperlukan, ulangi langkah-langkah ini untuk menambah atau mengedit bagian dokumentasi ke otorisasi lain.

Untuk mengedit bagian dokumentasi yang ada, lakukan hal berikut:

1. Di panel Dokumentasi, pilih tab Authorizers.
2. Gunakan bilah pencarian atau pilih otorisasi, lalu pilih Edit.

Publikasikan dokumentasi API menggunakan konsol API Gateway

Prosedur berikut menjelaskan cara mempublikasikan versi dokumentasi.

Untuk memublikasikan versi dokumentasi menggunakan konsol API Gateway

1. Di panel navigasi utama, pilih Dokumentasi.
2. Pilih Publikasikan dokumentasi.
3. Siapkan publikasi:
    a. Untuk Stage, pilih panggung.
    b. Untuk Versi, masukkan pengenal versi, misalnya,`1.0.0`.

c. (Opsional) Untuk Deskripsi, masukkan deskripsi.

4. Pilih Terbitkan.

Anda sekarang dapat melanjutkan untuk mengunduh dokumentasi yang diterbitkan dengan mengekspor dokumentasi ke file OpenAPI eksternal. Untuk mempelajari selengkapnya, lihat the section called “Ekspor REST API”.

Dokumentasikan API menggunakan API Gateway REST API

Di bagian ini, kami menjelaskan cara membuat dan memelihara bagian dokumentasi API menggunakan API Gateway REST API.

Sebelum membuat dan mengedit dokumentasi API, buat dulu API. Pada bagian ini, kita menggunakan PetStoreAPI sebagai contoh. Untuk membuat API menggunakan konsol API Gateway, ikuti petunjuk diTutorial: Buat REST API dengan mengimpor contoh.

Topik

- Dokumentasikan API entitas
- Dokumentasikan RESOURCE entitas
- Dokumentasikan METHOD entitas
- Dokumentasikan QUERY_PARAMETER entitas
- Dokumentasikan PATH_PARAMETER entitas
- Dokumentasikan REQUEST_BODY entitas
- Dokumentasikan REQUEST_HEADER entitas
- Dokumentasikan RESPONSE entitas
- Dokumentasikan RESPONSE_HEADER entitas
- Dokumentasikan AUTHORIZER entitas
- Dokumentasikan MODEL entitas
- Perbarui bagian dokumentasi
- Daftar bagian dokumentasi

Dokumentasikan **`API`** entitas

Untuk menambahkan dokumentasi untuk API, tambahkan DocumentationPartresource untuk entitas API:

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "API"
    },
    "properties": "{\n\t\"info\": {\n\t\t\"description\" : \"Your first API with Amazon
 API Gateway.\"\n\t}\n}"
}
```

Jika berhasil, operasi mengembalikan `201 Created` respons yang berisi `DocumentationPart` instance yang baru dibuat di payload. Misalnya:

```
{
  ...
  "id": "s2e5xf",
  "location": {
    "path": null,
    "method": null,
    "name": null,
    "statusCode": null,
    "type": "API"
  },
  "properties": "{\n\t\"info\": {\n\t\t\"description\" : \"Your first API with Amazon
 API Gateway.\"\n\t}\n}"
}
```

Jika bagian dokumentasi telah ditambahkan, `409 Conflict` respons kembali, berisi pesan kesalahan Dalam hal ini, Anda harus memanggil operasi documentationpart:update. `Documentation part already exists for the specified location: type 'API'."`

```
PATCH /restapis/4wk1k4onj3/documentation/parts/part_id HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
```

```
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "/properties",
    "value" : "{\n\t\"info\": {\n\t\t\"description\" : \"Your first API with Amazon API
 Gateway.\"\n\t}\n}"
  } ]
}
```

Respons yang berhasil mengembalikan kode `200 OK` status dengan payload yang berisi `DocumentationPart` instance yang diperbarui dalam payload.

Dokumentasikan **`RESOURCE`** entitas

Untuk menambahkan dokumentasi sumber daya root API, tambahkan sumber DocumentationPartdaya yang ditargetkan untuk sumber daya Sumber Daya terkait:

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
       "type" : "RESOURCE",
    },
    "properties" : "{\n\t\"description\" : \"The PetStore root resource.\"\n}"
}
```

Jika berhasil, operasi mengembalikan `201 Created` respons yang berisi `DocumentationPart` instance yang baru dibuat di payload. Misalnya:

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

Ketika jalur sumber daya tidak ditentukan, sumber daya diasumsikan sebagai sumber daya root. Anda dapat menambahkan `"path": "/"` `properties` untuk membuat spesifikasi eksplisit.

Untuk membuat dokumentasi sumber daya turunan API, tambahkan sumber DocumentationPartdaya yang ditargetkan untuk sumber daya Sumber Daya terkait:

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "RESOURCE",
```

```
        "path" : "/pets"
    },
    "properties": "{\n\t\"description\" : \"A child resource under the root of
 PetStore.\"\n}"
}
```

Jika berhasil, operasi mengembalikan `201 Created` respons yang berisi `DocumentationPart` instance yang baru dibuat di payload. Misalnya:

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

Untuk menambahkan dokumentasi sumber daya turunan yang ditentukan oleh parameter jalur, tambahkan DocumentationPartsumber daya yang ditargetkan untuk sumber daya Sumber Daya:

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
```

```
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "RESOURCE",
         "path" : "/pets/{petId}"
    },
    "properties": "{\n\t\"description\" : \"A child resource specified by the petId
 path parameter.\"\n}"
}
```

Jika berhasil, operasi mengembalikan `201 Created` respons yang berisi `DocumentationPart` instance yang baru dibuat di payload. Misalnya:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/k6fpwb"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/k6fpwb"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/k6fpwb"
    }
  },
  "id": "k6fpwb",
  "location": {
    "path": "/pets/{petId}",
    "method": null,
    "name": null,
    "statusCode": null,
```

```
    "type": "RESOURCE"
  },
  "properties": "{\n\t\"description\" : \"A child resource specified by the petId path
 parameter.\"\n}"
}
```

> **Note**
>
> DocumentationPartInstance suatu RESOURCE entitas tidak dapat diwarisi oleh sumber daya anak mana pun.

Dokumentasikan **`METHOD`** entitas

Untuk menambahkan dokumentasi untuk metode API, tambahkan DocumentationPartsumber daya yang ditargetkan untuk sumber daya Metode yang sesuai:

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "METHOD",
         "path" : "/pets",
         "method" : "GET"
    },
    "properties": "{\n\t\"summary\" : \"List all pets.\"\n}"
}
```

Jika berhasil, operasi mengembalikan `201 Created` respons yang berisi `DocumentationPart` instance yang baru dibuat di payload. Misalnya:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
```

```
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    }
  },
  "id": "o64jbj",
  "location": {
    "path": "/pets",
    "method": "GET",
    "name": null,
    "statusCode": null,
    "type": "METHOD"
  },
  "properties": "{\n\t\"summary\" : \"List all pets.\"\n}"
}
```

Jika berhasil, operasi mengembalikan `201 Created` respons yang berisi `DocumentationPart` instance yang baru dibuat di payload. Misalnya:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
```

```
    }
  },
  "id": "o64jbj",
  "location": {
    "path": "/pets",
    "method": "GET",
    "name": null,
    "statusCode": null,
    "type": "METHOD"
  },
  "properties": "{\n\t\"summary\" : \"List all pets.\"\n}"
}
```

Jika `location.method` bidang tidak ditentukan dalam permintaan sebelumnya, diasumsikan sebagai `ANY` metode yang diwakili oleh karakter kartu `*` liar.

Untuk memperbarui konten dokumentasi `METHOD` entitas, panggil operasi documentationpart:update, menyediakan peta baru: `properties`

```
PATCH /restapis/4wk1k4onj3/documentation/parts/part_id HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "/properties",
    "value" : "{\n\t\"tags\" : [ \"pets\" ], \n\t\"summary\" : \"List all pets.\"\n}"
  } ]
}
```

Respons yang berhasil mengembalikan kode `200 OK` status dengan payload yang berisi `DocumentationPart` instance yang diperbarui dalam payload. Misalnya:

```
{
  "_links": {
    "curies": {
```

```
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    }
  },
  "id": "o64jbj",
  "location": {
    "path": "/pets",
    "method": "GET",
    "name": null,
    "statusCode": null,
    "type": "METHOD"
  },
  "properties": "{\n\t\"tags\" : [ \"pets\" ], \n\t\"summary\" : \"List all pets.\"\n}"
}
```

Dokumentasikan **QUERY_PARAMETER** entitas

Untuk menambahkan dokumentasi untuk parameter kueri permintaan, tambahkan DocumentationPartsumber daya yang ditargetkan untuk QUERY_PARAMETER jenis, dengan bidang yang valid dari path dannname.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "QUERY_PARAMETER",
```

```
        "path" : "/pets",
        "method" : "GET",
        "name" : "page"
    },
    "properties": "{\n\t\"description\" : \"Page number of results to return.\"\n}"
}
```

Jika berhasil, operasi mengembalikan `201 Created` respons yang berisi `DocumentationPart` instance yang baru dibuat di payload. Misalnya:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/h9ht5w"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/h9ht5w"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/h9ht5w"
    }
  },
  "id": "h9ht5w",
  "location": {
    "path": "/pets",
    "method": "GET",
    "name": "page",
    "statusCode": null,
    "type": "QUERY_PARAMETER"
  },
  "properties": "{\n\t\"description\" : \"Page number of results to return.\"\n}"
}
```

`properties`Peta bagian dokumentasi `QUERY_PARAMETER` entitas dapat diwarisi oleh salah satu `QUERY_PARAMETER` entitas turunannya. Misalnya, jika Anda menambahkan `treats` sumber daya setelahnya`/pets/{petId}`, mengaktifkan `GET` metode`/pets/{petId}/treats`, dan

mengekspos parameter `page` kueri, parameter kueri anak ini mewarisi `properties` peta dari parameter kueri yang diberi nama sama dari `GET /pets` metode tersebut, kecuali jika Anda secara eksplisit menambahkan `DocumentationPart` sumber daya ke parameter kueri metode tersebut`page`. `DocumentationPart` `GET /pets/{petId}/treats`

Dokumentasikan **`PATH_PARAMETER`** entitas

Untuk menambahkan dokumentasi untuk parameter jalur, tambahkan DocumentationPartsumber daya untuk `PATH_PARAMETER` entitas.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "PATH_PARAMETER",
         "path" : "/pets/{petId}",
         "method" : "*",
         "name" : "petId"
    },
    "properties": "{\n\t\"description\" : \"The id of the pet to retrieve.\"\n}"
}
```

Jika berhasil, operasi mengembalikan `201 Created` respons yang berisi `DocumentationPart` instance yang baru dibuat di payload. Misalnya:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/ckpgog"
    },
```

```
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/ckpgog"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/ckpgog"
    }
  },
  "id": "ckpgog",
  "location": {
    "path": "/pets/{petId}",
    "method": "*",
    "name": "petId",
    "statusCode": null,
    "type": "PATH_PARAMETER"
  },
  "properties": "{\n  \"description\" : \"The id of the pet to retrieve\"\n}"
}
```

Dokumentasikan **`REQUEST_BODY`** entitas

Untuk menambahkan dokumentasi untuk badan permintaan, tambahkan DocumentationPartsumber daya untuk badan permintaan.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "REQUEST_BODY",
         "path" : "/pets",
         "method" : "POST"
    },
    "properties": "{\n\t\"description\" : \"A Pet object to be added to PetStore.\"\n}"
}
```

Jika berhasil, operasi mengembalikan `201 Created` respons yang berisi `DocumentationPart` instance yang baru dibuat di payload. Misalnya:

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

Dokumentasikan **REQUEST_HEADER** entitas

Untuk menambahkan dokumentasi untuk header permintaan, tambahkan DocumentationPartsumber daya untuk header permintaan.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
```

```
    "location" : {
         "type" : "REQUEST_HEADER",
         "path" : "/pets",
         "method" : "GET",
         "name" : "x-my-token"
    },
    "properties": "{\n\t\"description\" : \"A custom token used to authorization the
 method invocation.\"\n}"
}
```

Jika berhasil, operasi mengembalikan `201 Created` respons yang berisi `DocumentationPart` instance yang baru dibuat di payload. Misalnya:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/h0m3uf"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/h0m3uf"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/h0m3uf"
    }
  },
  "id": "h0m3uf",
  "location": {
    "path": "/pets",
    "method": "GET",
    "name": "x-my-token",
    "statusCode": null,
    "type": "REQUEST_HEADER"
  },
  "properties": "{\n\t\"description\" : \"A custom token used to authorization the
 method invocation.\"\n}"
}
```

Dokumentasikan **RESPONSE** entitas

Untuk menambahkan dokumentasi untuk respons kode status, tambahkan DocumentationPartsumber daya yang ditargetkan untuk sumber MethodResponsedaya yang sesuai.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location": {
      "path": "/",
      "method": "*",
      "name": null,
      "statusCode": "200",
      "type": "RESPONSE"
    },
    "properties": "{\n  \"description\" : \"Successful operation.\"\n}"
}
```

Jika berhasil, operasi mengembalikan `201 Created` respons yang berisi `DocumentationPart` instance yang baru dibuat di payload. Misalnya:

```
{
    "_links": {
        "self": {
            "href": "/restapis/4wk1k4onj3/documentation/parts/lattew"
        },
        "documentationpart:delete": {
            "href": "/restapis/4wk1k4onj3/documentation/parts/lattew"
        },
        "documentationpart:update": {
            "href": "/restapis/4wk1k4onj3/documentation/parts/lattew"
        }
    },
    "id": "lattew",
    "location": {
        "path": "/",
        "method": "*",
```

```
        "name": null,
        "statusCode": "200",
        "type": "RESPONSE"
    },
    "properties": "{\n  \"description\" : \"Successful operation.\"\n}"
}
```

Dokumentasikan **`RESPONSE_HEADER`** entitas

Untuk menambahkan dokumentasi untuk header respons, tambahkan DocumentationPartsumber daya untuk header respons.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

  "location": {
    "path": "/",
    "method": "GET",
    "name": "Content-Type",
    "statusCode": "200",
    "type": "RESPONSE_HEADER"
  },
  "properties": "{\n  \"description\" : \"Media type of request\"\n}"
```

Jika berhasil, operasi mengembalikan `201 Created` respons yang berisi `DocumentationPart` instance yang baru dibuat di payload. Misalnya:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/fev7j7"
    },
```

```
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/fev7j7"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/fev7j7"
    }
  },
  "id": "fev7j7",
  "location": {
    "path": "/",
    "method": "GET",
    "name": "Content-Type",
    "statusCode": "200",
    "type": "RESPONSE_HEADER"
  },
  "properties": "{\n  \"description\" : \"Media type of request\"\n}"
}
```

Dokumentasi header `Content-Type` respons ini adalah dokumentasi default untuk `Content-Type` header dari setiap respons API.

Dokumentasikan **`AUTHORIZER`** entitas

Untuk menambahkan dokumentasi bagi otorisasi API, tambahkan DocumentationPartsumber daya yang ditargetkan untuk otorisasi yang ditentukan.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "AUTHORIZER",
         "name" : "myAuthorizer"
    },
    "properties": "{\n\t\"description\" : \"Authorizes invocations of configured
 methods.\"\n}"
}
```

Jika berhasil, operasi mengembalikan `201 Created` respons yang berisi `DocumentationPart` instance yang baru dibuat di payload. Misalnya:


```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/pw3qw3"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/pw3qw3"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/pw3qw3"
    }
  },
  "id": "pw3qw3",
  "location": {
    "path": null,
    "method": null,
    "name": "myAuthorizer",
    "statusCode": null,
    "type": "AUTHORIZER"
  },
  "properties": "{\n\t\"description\" : \"Authorizes invocations of configured methods.
\"\n}"
}
```


> **Note**
>
> DocumentationPartInstance suatu `AUTHORIZER` entitas tidak dapat diwarisi oleh sumber daya anak mana pun.

Dokumentasikan **`MODEL`** entitas

Mendokumentasikan `MODEL` entitas melibatkan pembuatan dan pengelolaan `DocumentPart` instance untuk model dan masing-masing model '. `properties` Misalnya, untuk `Error` model yang disertakan dengan setiap API secara default memiliki definisi skema berikut,

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

dan membutuhkan dua `DocumentationPart` contoh, satu untuk `Model` dan yang lainnya untuk `message` propertinya:

```
{
  "location": {
    "type": "MODEL",
    "name": "Error"
  },
  "properties": {
    "title": "Error Schema",
    "description": "A description of the Error model"
  }
}
```

and

```
{
  "location": {
    "type": "MODEL",
    "name": "Error.message"
  },
  "properties": {
    "description": "An error message."
  }
}
```

Saat API diekspor, properti akan mengganti nilai dalam skema asli. `DocumentationPart`

Untuk menambahkan dokumentasi untuk model API, tambahkan DocumentationPartsumber daya yang ditargetkan untuk model yang ditentukan.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "MODEL",
         "name" : "Pet"
    },
    "properties": "{\n\t\"description\" : \"Data structure of a Pet object.\"\n}"
}
```

Jika berhasil, operasi mengembalikan `201 Created` respons yang berisi `DocumentationPart` instance yang baru dibuat di payload. Misalnya:

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

Ulangi langkah yang sama untuk membuat DocumentationPart instance untuk salah satu properti model.

> **Note**
>
> DocumentationPartInstance suatu `MODEL` entitas tidak dapat diwarisi oleh sumber daya anak mana pun.

Perbarui bagian dokumentasi

Untuk memperbarui bagian dokumentasi dari semua jenis entitas API, kirimkan permintaan PATCH pada DocumentationPartinstance pengidentifikasi bagian tertentu untuk mengganti `properties` peta yang ada dengan yang baru.

```
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"
  } ]
}
```

Respons yang berhasil mengembalikan kode `200 OK` status dengan payload yang berisi `DocumentationPart` instance yang diperbarui dalam payload.

Anda dapat memperbarui beberapa bagian dokumentasi dalam satu `PATCH` permintaan.

Daftar bagian dokumentasi

Untuk mencantumkan bagian dokumentasi dari semua jenis entitas API, kirimkan permintaan GET pada DocumentationPartskoleksi.

```
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
```

Respons yang berhasil mengembalikan kode `200 OK` status dengan payload yang berisi `DocumentationPart` instance yang tersedia di payload.

Publikasikan dokumentasi API menggunakan API Gateway REST API

Untuk memublikasikan dokumentasi untuk API, buat, perbarui, atau dapatkan snapshot dokumentasi, lalu kaitkan snapshot dokumentasi dengan tahap API. Saat membuat snapshot dokumentasi, Anda juga dapat mengaitkannya dengan tahap API secara bersamaan.

Topik

- Buat snapshot dokumentasi dan kaitkan dengan tahap API
- Buat snapshot dokumentasi
- Perbarui snapshot dokumentasi
- Dapatkan snapshot dokumentasi
- Kaitkan snapshot dokumentasi dengan tahap API
- Unduh snapshot dokumentasi yang terkait dengan panggung

Buat snapshot dokumentasi dan kaitkan dengan tahap API

Untuk membuat snapshot bagian dokumentasi API dan mengaitkannya dengan tahap API secara bersamaan, kirimkan `POST` permintaan berikut:

```
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"
}
```

Jika berhasil, operasi mengembalikan `200 OK` respons, yang berisi `DocumentationVersion` instance yang baru dibuat sebagai muatan.

Atau, Anda dapat membuat snapshot dokumentasi tanpa mengaitkannya dengan tahap API terlebih dahulu dan kemudian memanggil restapi:update untuk mengaitkan snapshot dengan tahap API tertentu. Anda juga dapat memperbarui atau menanyakan snapshot dokumentasi yang ada dan kemudian memperbarui asosiasi tahapannya. Kami menunjukkan langkah-langkah di empat bagian berikutnya.

Buat snapshot dokumentasi

Untuk membuat snapshot bagian dokumentasi API, buat DocumentationVersionresource baru dan tambahkan ke DocumentationVersionskoleksi 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"
}
```

Jika berhasil, operasi mengembalikan `200 OK` respons, yang berisi `DocumentationVersion` instance yang baru dibuat sebagai muatan.

Perbarui snapshot dokumentasi

Anda hanya dapat memperbarui snapshot dokumentasi dengan memodifikasi `description` properti sumber daya yang sesuai DocumentationVersion. Contoh berikut menunjukkan cara memperbarui deskripsi snapshot dokumentasi seperti yang diidentifikasi oleh pengenal versinya, misalnya*version*,. `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."
    }]
}
```

Jika berhasil, operasi mengembalikan `200 OK` respons, yang berisi `DocumentationVersion` instance yang diperbarui sebagai muatan.

Dapatkan snapshot dokumentasi

Untuk mendapatkan snapshot dokumentasi, kirimkan `GET` permintaan terhadap DocumentationVersionsumber daya yang ditentukan. Contoh berikut menunjukkan cara mendapatkan snapshot dokumentasi dari pengenal versi tertentu, 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
```

Kaitkan snapshot dokumentasi dengan tahap API

Untuk mempublikasikan dokumentasi API, kaitkan snapshot dokumentasi dengan tahap API. Anda harus sudah membuat tahap API sebelum mengaitkan versi dokumentasi dengan stage.

Untuk mengaitkan snapshot dokumentasi dengan tahap API menggunakan API Gateway REST API, panggil operasi stage:update untuk menyetel versi dokumentasi yang diinginkan di properti: `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"
    }]
}
```

Unduh snapshot dokumentasi yang terkait dengan panggung

Setelah versi bagian dokumentasi dikaitkan dengan tahapan, Anda dapat mengekspor bagian dokumentasi bersama dengan definisi entitas API, ke file eksternal, menggunakan konsol API Gateway, API Gateway REST API, salah satu SDK, atau AWS CLI untuk API Gateway. Prosesnya sama dengan mengekspor API. Format file yang diekspor dapat berupa JSON atau YAMB.

Menggunakan API Gateway REST API, Anda juga dapat secara eksplisit menyetel parameter `extension=documentation,integrations,authorizers` kueri untuk menyertakan bagian dokumentasi API, integrasi API, dan otorisasi dalam ekspor API. Secara default, bagian dokumentasi disertakan, tetapi integrasi dan otorisasi dikecualikan, saat Anda mengekspor API. Output default dari ekspor API cocok untuk distribusi dokumentasi.

Untuk mengekspor dokumentasi API dalam file OpenAPI JSON eksternal menggunakan API Gateway REST API, kirimkan permintaan berikut`GET`:

```
GET /restapis/restapi_id/stages/stage_name/exports/swagger?extensions=documentation
 HTTP/1.1
Accept: application/json
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret
```

Di sini, `x-amazon-apigateway-documentation` objek berisi bagian dokumentasi dan definisi entitas API berisi properti dokumentasi yang didukung oleh OpenAPI. Output tidak termasuk rincian integrasi atau otorisasi Lambda (sebelumnya dikenal sebagai otorisasi khusus). Untuk memasukkan kedua detail, atur`extensions=integrations,authorizers,documentation`. Untuk menyertakan detail integrasi tetapi bukan otorisasi, atur. `extensions=integrations,documentation`

Anda harus mengatur `Accept:application/json` header dalam permintaan untuk menampilkan hasil dalam file JSON. Untuk menghasilkan output YAMAL, ubah header permintaan menjadi`Accept:application/yaml`.

Sebagai contoh, kita akan melihat API yang mengekspos `GET` metode sederhana pada sumber daya root (`/`). API ini memiliki empat entitas API yang didefinisikan dalam file definisi OpenAPI, satu untuk masing-masing`API`,, `MODEL`, `METHOD`, dan `RESPONSE` tipe. Bagian dokumentasi telah ditambahkan ke masing-masing`API`,`METHOD`, dan `RESPONSE` entitas. Memanggil perintah documentation-exporting sebelumnya, kita mendapatkan output berikut, dengan bagian-bagian dokumentasi yang tercantum dalam `x-amazon-apigateway-documentation` objek sebagai ekstensi ke file OpenAPI standar.

OpenAPI 3.0

```
{
   "openapi": "3.0.0",
   "info": {
      "description": "API info description",
      "version": "2016-11-22T22:39:14Z",
      "title": "doc",
      "x-bar": "API info x-bar"
   },
   "paths": {
      "/": {
         "get": {
```

```
            "description": "Method description.",
            "responses": {
               "200": {
                  "description": "200 response",
                  "content": {
                     "application/json": {
                        "schema": {
                           "$ref": "#/components/schemas/Empty"
                        }
                     }
                  }
               }
            },
            "x-example": "x- Method example"
         },
         "x-bar": "resource x-bar"
      }
   },
   "x-amazon-apigateway-documentation": {
      "version": "1.0.0",
      "createdDate": "2016-11-22T22:41:40Z",
      "documentationParts": [
         {
            "location": {
               "type": "API"
            },
            "properties": {
               "description": "API description",
               "foo": "API foo",
               "x-bar": "API x-bar",
               "info": {
                  "description": "API info description",
                  "version": "API info version",
                  "foo": "API info foo",
                  "x-bar": "API info x-bar"
               }
            }
         },
         {
            "location": {
               "type": "METHOD",
               "method": "GET"
            },
            "properties": {
```

```
               "description": "Method description.",
               "x-example": "x- Method example",
               "foo": "Method foo",
               "info": {
                  "version": "method info version",
                  "description": "method info description",
                  "foo": "method info foo"
               }
            }
         },
         {
            "location": {
               "type": "RESOURCE"
            },
            "properties": {
               "description": "resource description",
               "foo": "resource foo",
               "x-bar": "resource x-bar",
               "info": {
                  "description": "resource info description",
                  "version": "resource info version",
                  "foo": "resource info foo",
                  "x-bar": "resource info x-bar"
               }
            }
         }
      ]
   },
   "x-bar": "API x-bar",
   "servers": [
      {
         "url": "https://rznaap68yi.execute-api.ap-southeast-1.amazonaws.com/
{basePath}",
         "variables": {
            "basePath": {
              "default": "/test"
            }
         }
      }
   ],
   "components": {
      "schemas": {
         "Empty": {
            "type": "object",
```

```
            "title": "Empty Schema"
         }
      }
   }
}
```

OpenAPI 2.0

```
{
  "swagger" : "2.0",
  "info" : {
    "description" : "API info description",
    "version" : "2016-11-22T22:39:14Z",
    "title" : "doc",
    "x-bar" : "API info x-bar"
  },
  "host" : "rznaap68yi.execute-api.ap-southeast-1.amazonaws.com",
  "basePath" : "/test",
  "schemes" : [ "https" ],
  "paths" : {
    "/" : {
      "get" : {
        "description" : "Method description.",
        "produces" : [ "application/json" ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "schema" : {
              "$ref" : "#/definitions/Empty"
            }
          }
        },
        "x-example" : "x- Method example"
      },
      "x-bar" : "resource x-bar"
    }
  },
  "definitions" : {
    "Empty" : {
      "type" : "object",
      "title" : "Empty Schema"
    }
  },
```

```
  "x-amazon-apigateway-documentation" : {
    "version" : "1.0.0",
    "createdDate" : "2016-11-22T22:41:40Z",
    "documentationParts" : [ {
      "location" : {
        "type" : "API"
      },
      "properties" : {
        "description" : "API description",
        "foo" : "API foo",
        "x-bar" : "API x-bar",
        "info" : {
          "description" : "API info description",
          "version" : "API info version",
          "foo" : "API info foo",
          "x-bar" : "API info x-bar"
        }
      }
    }, {
      "location" : {
        "type" : "METHOD",
        "method" : "GET"
      },
      "properties" : {
        "description" : "Method description.",
        "x-example" : "x- Method example",
        "foo" : "Method foo",
        "info" : {
          "version" : "method info version",
          "description" : "method info description",
          "foo" : "method info foo"
        }
      }
    }, {
      "location" : {
        "type" : "RESOURCE"
      },
      "properties" : {
        "description" : "resource description",
        "foo" : "resource foo",
        "x-bar" : "resource x-bar",
        "info" : {
          "description" : "resource info description",
          "version" : "resource info version",
```

```
          "foo" : "resource info foo",
          "x-bar" : "resource info x-bar"
        }
      }
    } ]
  },
  "x-bar" : "API x-bar"
}
```

Untuk atribut yang sesuai dengan OpenAPI yang ditentukan dalam `properties` peta bagian dokumentasi, API Gateway menyisipkan atribut ke dalam definisi entitas API terkait. Atribut `x-`*`something`* adalah ekstensi OpenAPI standar. Ekstensi ini disebarkan ke dalam definisi entitas API. Misalnya, lihat `x-example` atribut untuk `GET` metode. Atribut seperti `foo` bukan bagian dari spesifikasi OpenAPI dan tidak disuntikkan ke dalam definisi entitas API terkait.

Jika alat rendering dokumentasi (misalnya, OpenAPI UI) mem-parsing definisi entitas API untuk mengekstrak atribut dokumentasi, atribut 'instance' yang tidak sesuai dengan OpenAPI `properties` tidak tersedia untuk alat tersebut. `DocumentationPart` Namun, jika alat rendering dokumentasi mem-parsing `x-amazon-apigateway-documentation` objek untuk mendapatkan konten, atau jika alat memanggil restapi:documentation-parts dan documenationpart:by-id untuk mengambil bagian dokumentasi dari API Gateway, semua atribut dokumentasi tersedia untuk ditampilkan oleh alat tersebut.

Untuk mengekspor dokumentasi dengan definisi entitas API yang berisi detail integrasi ke file OpenAPI JSON, kirimkan permintaan berikut`GET`:

```
GET /restapis/restapi_id/stages/stage_name/exports/swagger?
extensions=integrations,documentation HTTP/1.1
Accept: application/json
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret
```

Untuk mengekspor dokumentasi dengan definisi entitas API yang berisi detail integrasi dan otorisasi ke file OpenAPI YAMAL, kirimkan permintaan berikut: `GET`

```
GET /restapis/restapi_id/stages/stage_name/exports/swagger?
extensions=integrations,authorizers,documentation HTTP/1.1
Accept: application/yaml
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret
```

Untuk menggunakan konsol API Gateway untuk mengekspor dan mengunduh dokumentasi API yang dipublikasikan, ikuti petunjuk diEkspor REST API menggunakan konsol API Gateway.

Impor dokumentasi API

Seperti halnya mengimpor definisi entitas API, Anda dapat mengimpor bagian dokumentasi dari file OpenAPI eksternal ke API di API Gateway. Anda menentukan bagian to-be-imported dokumentasi dalam x-amazon-apigateway-documentation objek ekstensi dalam file definisi OpenAPI yang valid. Mengimpor dokumentasi tidak mengubah definisi entitas API yang ada.

Anda memiliki opsi untuk menggabungkan bagian dokumentasi yang baru ditentukan ke dalam bagian dokumentasi yang ada di API Gateway atau menimpa bagian dokumentasi yang ada. Dalam `MERGE` mode, bagian dokumentasi baru yang ditentukan dalam file OpenAPI ditambahkan ke `DocumentationParts` koleksi API. Jika impor `DocumentationPart` sudah ada, atribut yang diimpor menggantikan yang sudah ada jika keduanya berbeda. Atribut dokumentasi lain yang ada tetap tidak terpengaruh. Dalam `OVERWRITE` mode, seluruh `DocumentationParts` koleksi diganti sesuai dengan file definisi OpenAPI yang diimpor.

Mengimpor bagian dokumentasi menggunakan API Gateway REST API

Untuk mengimpor dokumentasi API menggunakan API Gateway REST API, panggil operasi documentationpart:import. Contoh berikut menunjukkan cara menimpa bagian dokumentasi API yang ada dengan satu `GET /` metode, mengembalikan `200 OK` respons saat berhasil.

OpenAPI 3.0

```
PUT /restapis/<restapi_id>/documentation/parts&mode=overwrite&failonwarnings=true
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
```

```
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
   "openapi": "3.0.0",
   "info": {
      "description": "description",
      "version": "1",
      "title": "doc"
   },
   "paths": {
      "/": {
         "get": {
            "description": "Method description.",
            "responses": {
               "200": {
                  "description": "200 response",
                  "content": {
                     "application/json": {
                        "schema": {
                           "$ref": "#/components/schemas/Empty"
                        }
                     }
                  }
               }
            }
         }
      }
   },
   "x-amazon-apigateway-documentation": {
      "version": "1.0.3",
      "documentationParts": [
         {
            "location": {
               "type": "API"
            },
            "properties": {
               "description": "API description",
               "info": {
                  "description": "API info description 4",
                  "version": "API info version 3"
               }
            }
```

```
        },
        {
           "location": {
              "type": "METHOD",
              "method": "GET"
           },
           "properties": {
              "description": "Method description."
           }
        },
        {
           "location": {
              "type": "MODEL",
              "name": "Empty"
           },
           "properties": {
              "title": "Empty Schema"
           }
        },
        {
           "location": {
              "type": "RESPONSE",
              "method": "GET",
              "statusCode": "200"
           },
           "properties": {
              "description": "200 response"
           }
        }
     ]
  },
  "servers": [
     {
        "url": "/"
     }
  ],
  "components": {
     "schemas": {
        "Empty": {
           "type": "object",
           "title": "Empty Schema"
        }
     }
  }
```

```
}
```

OpenAPI 2.0

```
PUT /restapis/<restapi_id>/documentation/parts&mode=overwrite&failonwarnings=true
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
  "swagger": "2.0",
  "info": {
    "description": "description",
    "version": "1",
    "title": "doc"
  },
  "host": "",
  "basePath": "/",
  "schemes": [
    "https"
  ],
  "paths": {
    "/": {
      "get": {
        "description": "Method description.",
        "produces": [
          "application/json"
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        }
      }
    }
  },
  "definitions": {
```

```
    "Empty": {
      "type": "object",
      "title": "Empty Schema"
    }
  },
  "x-amazon-apigateway-documentation": {
    "version": "1.0.3",
    "documentationParts": [
      {
        "location": {
          "type": "API"
        },
        "properties": {
          "description": "API description",
          "info": {
            "description": "API info description 4",
            "version": "API info version 3"
          }
        }
      },
      {
        "location": {
          "type": "METHOD",
          "method": "GET"
        },
        "properties": {
          "description": "Method description."
        }
      },
      {
        "location": {
          "type": "MODEL",
          "name": "Empty"
        },
        "properties": {
          "title": "Empty Schema"
        }
      },
      {
        "location": {
          "type": "RESPONSE",
          "method": "GET",
          "statusCode": "200"
        },
```

```
        "properties": {
          "description": "200 response"
        }
      }
    ]
  }
}
```

Jika berhasil, permintaan ini mengembalikan respons 200 OK yang berisi impor `DocumentationPartId` dalam muatan.

```
{
  "ids": [
    "kg3mth",
    "796rtf",
    "zhek4p",
    "5ukm9s"
  ]
}
```

Selain itu, Anda juga dapat memanggil restapi:import atau restapi:put, memasok bagian dokumentasi dalam objek `x-amazon-apigateway-documentation` sebagai bagian dari file OpenAPI input dari definisi API. Untuk mengecualikan bagian dokumentasi dari impor API, atur `ignore=documentation` parameter kueri permintaan.

Mengimpor bagian dokumentasi menggunakan konsol API Gateway

Petunjuk berikut menjelaskan cara mengimpor bagian dokumentasi.

Untuk menggunakan konsol untuk mengimpor bagian dokumentasi API dari file eksternal

1. Di panel navigasi utama, pilih Dokumentasi.
2. Pilih Import (Impor).
3. Jika Anda memiliki dokumentasi yang sudah ada, pilih untuk Menimpa atau Menggabungkan dokumentasi baru Anda.
4. Pilih Pilih file untuk memuat file dari drive, atau masukkan konten file ke dalam tampilan file. Sebagai contoh, lihat payload permintaan contoh diMengimpor bagian dokumentasi menggunakan API Gateway REST API.

5. Pilih cara menangani peringatan saat impor. Pilih salah satu Gagal pada peringatan atau Abaikan peringatan. Untuk informasi selengkapnya, lihat the section called “Kesalahan dan peringatan selama impor”.
6. Pilih Import (Impor).

Kontrol akses ke dokumentasi API

Jika Anda memiliki tim dokumentasi khusus untuk menulis dan mengedit dokumentasi API Anda, Anda dapat mengonfigurasi izin akses terpisah untuk pengembang Anda (untuk pengembangan API) dan untuk penulis atau editor Anda (untuk pengembangan konten). Ini sangat tepat ketika vendor pihak ketiga terlibat dalam membuat dokumentasi untuk Anda.

Untuk memberi tim dokumentasi Anda akses untuk membuat, memperbarui, dan memublikasikan dokumentasi API Anda, Anda dapat menetapkan peran IAM kepada tim dokumentasi dengan kebijakan IAM berikut, di mana *account_id adalah ID* AWS akun tim dokumentasi Anda.

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

    {
      "Sid": "StmtDocPartsAddEditViewDelete",
      "Effect": "Allow",
      "Action": [
        "apigateway:GET",
        "apigateway:PUT",
        "apigateway:POST",
        "apigateway:PATCH",
        "apigateway:DELETE"
      ],
      "Resource": [
        "arn:aws:apigateway::account_id:/restapis/*/documentation/*"
      ]
    }
  ]
}
```

Untuk informasi tentang menyetel izin untuk mengakses sumber daya API Gateway, lihatthe section called “Cara kerja Amazon API Gateway dengan IAM”.

Membuat SDK untuk REST API di API Gateway

Untuk memanggil REST API Anda dengan cara khusus platform atau bahasa, Anda harus membuat SDK API khusus platform atau bahasa. Saat ini, API Gateway mendukung pembuatan SDK untuk API di Java, Java untuk Android JavaScript, Objective-C atau Swift untuk iOS, dan Ruby.

Bagian ini menjelaskan cara membuat SDK API Gateway API. Ini juga menunjukkan cara menggunakan SDK yang dihasilkan di aplikasi Java, aplikasi Java untuk Android, Objective-C dan Swift untuk aplikasi iOS, dan aplikasi. JavaScript

Untuk memfasilitasi diskusi, kami menggunakan API Gateway API ini, yang memperlihatkan fungsi Lambda Kalkulator Sederhana ini.

Sebelum melanjutkan, buat atau impor API dan terapkan setidaknya sekali di API Gateway. Untuk petunjuk, lihat Menerapkan REST API di Amazon API Gateway.

Topik

- Menghasilkan SDK untuk API menggunakan konsol API Gateway
- Menghasilkan SDK untuk API menggunakan perintah AWS CLI
- Fungsi Lambda Kalkulator Sederhana
- API kalkulator sederhana di API Gateway
- Definisi OpenAPI API kalkulator sederhana

Menghasilkan SDK untuk API menggunakan konsol API Gateway

Untuk membuat SDK khusus platform atau bahasa untuk API di API Gateway, Anda harus terlebih dahulu membuat, menguji, dan menerapkan API dalam satu tahap. Untuk tujuan ilustrasi, kami menggunakan Simple Calculator API sebagai contoh untuk menghasilkan SDK khusus bahasa atau platform di seluruh bagian ini. Untuk petunjuk tentang cara membuat, menguji, dan menerapkan API ini, lihat Membuat API Kalkulator Sederhana.

Topik

- Menghasilkan Java SDK dari API
- Menghasilkan SDK Android dari API
- Menghasilkan SDK iOS dari API
- Menghasilkan JavaScript SDK dari REST API
- Menghasilkan Ruby SDK dari API

Menghasilkan Java SDK dari API

Untuk menghasilkan Java SDK API di API Gateway

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih REST API.
3. Memilih Tahapan.
4. Di panel Tahapan, pilih nama panggung.
5. Buka menu Stage Actions, lalu pilih Generate SDK.
6. Untuk Platform, pilih platform Java dan lakukan hal berikut:
   a. Untuk Nama Layanan, tentukan nama SDK Anda. Sebagai contoh, **`SimpleCalcSdk`**. Ini menjadi nama kelas klien SDK Anda. Nama sesuai dengan `<name>` tag di bawah `<project>` dalam file pom.xml, yang ada di folder proyek SDK. Jangan sertakan tanda hubung.
   b. Untuk Java Package Name, tentukan nama paket untuk SDK Anda. Sebagai contoh, **`examples.aws.apig.simpleCalc.sdk`**. Nama paket ini digunakan sebagai namespace pustaka SDK Anda. Jangan sertakan tanda hubung.
   c. Untuk Java Build System, masukkan **`maven`** atau **`gradle`** tentukan sistem build.
   d. Untuk Id Grup Java, masukkan pengenal grup untuk proyek SDK Anda. Misalnya, masukkan **`my-apig-api-examples`**. Pengidentifikasi ini sesuai dengan `<groupId>` tag di bawah `<project>` dalam `pom.xml` file, yang ada di folder proyek SDK.
   e. Untuk Id Artifact Java, masukkan pengenal artefak untuk proyek SDK Anda. Misalnya, masukkan **`simple-calc-sdk`**. Pengidentifikasi ini sesuai dengan `<artifactId>` tag di bawah `<project>` dalam `pom.xml` file, yang ada di folder proyek SDK.
   f. Untuk Versi Artifact Java, masukkan string pengenal versi. Sebagai contoh, **`1.0.0`**. Pengidentifikasi versi ini sesuai dengan `<version>` tag di bawah `<project>` dalam `pom.xml` file, yang ada di folder proyek SDK.
   g. Untuk Teks Lisensi Kode Sumber, masukkan teks lisensi kode sumber Anda, jika ada.
7. Pilih Hasilkan SDK, lalu ikuti petunjuk di layar untuk mengunduh SDK yang dihasilkan oleh API Gateway.

Ikuti petunjuk Menggunakan Java SDK yang dihasilkan oleh API Gateway untuk REST API untuk menggunakan SDK yang dihasilkan.

Setiap kali memperbarui API, Anda harus menerapkan ulang API dan membuat ulang SDK agar pembaruan disertakan.

Menghasilkan SDK Android dari API

Untuk menghasilkan SDK Android API di API Gateway

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih REST API.
3. Memilih Tahapan.
4. Di panel Tahapan, pilih nama panggung.
5. Buka menu Stage Actions, lalu pilih Generate SDK.
6. Untuk Platform, pilih platform Android dan lakukan hal berikut:
   a. Untuk ID Grup, masukkan pengenal unik untuk proyek terkait. Ini digunakan dalam `pom.xml` file (misalnya,**`com.mycompany`**).
   b. Untuk paket Invoker, masukkan namespace untuk kelas klien yang dihasilkan (misalnya,). **`com.mycompany.clientsdk`**
   c. Untuk Artifact ID, masukkan nama file.jar yang dikompilasi tanpa versi. Ini digunakan dalam `pom.xml` file (misalnya,**`aws-apigateway-api-sdk`**).
   d. Untuk versi Artifact, masukkan nomor versi artefak untuk klien yang dihasilkan. Ini digunakan dalam `pom.xml` file dan harus mengikuti *`jurusan`*. *`kecil`*. pola *`tambalan`* (misalnya,**`1.0.0`**).
7. Pilih Hasilkan SDK, lalu ikuti petunjuk di layar untuk mengunduh SDK yang dihasilkan oleh API Gateway.

Ikuti petunjuk Menggunakan Android SDK yang dihasilkan oleh API Gateway untuk REST API untuk menggunakan SDK yang dihasilkan.

Setiap kali memperbarui API, Anda harus menerapkan ulang API dan membuat ulang SDK agar pembaruan disertakan.

Menghasilkan SDK iOS dari API

Untuk menghasilkan SDK iOS API di API Gateway

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.

2. Pilih REST API.
3. Memilih Tahapan.
4. Di panel Tahapan, pilih nama panggung.
5. Buka menu Stage Actions, lalu pilih Generate SDK.
6. Untuk Platform, pilih platform iOS (Objective-C) atau iOS (Swift) dan lakukan hal berikut:

    - Ketik awalan unik di kotak Awalan.

      Efek awalan adalah sebagai berikut: jika Anda menetapkan, misalnya, **`SIMPLE_CALC`** sebagai awalan untuk SDK SimpleCalcAPI dengan, dan `result` model `inputoutput`, SDK yang dihasilkan akan berisi `SIMPLE_CALCSimpleCalcClient` kelas yang merangkum API, termasuk permintaan/tanggapan metode. Selain itu, SDK yang dihasilkan akan berisi`SIMPLE_CALCinput`,`SIMPLE_CALCoutput`, dan `SIMPLE_CALCresult` kelas untuk mewakili input, output, dan hasil, masing-masing, untuk mewakili input permintaan dan output respons. Untuk informasi selengkapnya, lihat Menggunakan iOS SDK yang dihasilkan oleh API Gateway untuk REST API di Objective-C atau Swift.
7. Pilih Hasilkan SDK, lalu ikuti petunjuk di layar untuk mengunduh SDK yang dihasilkan oleh API Gateway.

Ikuti petunjuk Menggunakan iOS SDK yang dihasilkan oleh API Gateway untuk REST API di Objective-C atau Swift untuk menggunakan SDK yang dihasilkan.

Setiap kali memperbarui API, Anda harus menerapkan ulang API dan membuat ulang SDK agar pembaruan disertakan.

Menghasilkan JavaScript SDK dari REST API

Untuk menghasilkan JavaScript SDK API di API Gateway

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih REST API.
3. Memilih Tahapan.
4. Di panel Tahapan, pilih nama panggung.
5. Buka menu Stage Actions, lalu pilih Generate SDK.
6. Untuk Platform, pilih JavaScriptplatformnya.

7. Pilih Hasilkan SDK, lalu ikuti petunjuk di layar untuk mengunduh SDK yang dihasilkan oleh API Gateway.

Ikuti petunjuk Menggunakan JavaScript SDK yang dihasilkan oleh API Gateway untuk REST API untuk menggunakan SDK yang dihasilkan.

Setiap kali memperbarui API, Anda harus menerapkan ulang API dan membuat ulang SDK agar pembaruan disertakan.

Menghasilkan Ruby SDK dari API

Untuk menghasilkan Ruby SDK API di API Gateway

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih REST API.
3. Memilih Tahapan.
4. Di panel Tahapan, pilih nama panggung.
5. Buka menu Stage Actions, lalu pilih Generate SDK.
6. Untuk Platform, pilih platform Ruby dan lakukan hal berikut:
   a. Untuk Nama Layanan, tentukan nama SDK Anda. Sebagai contoh, **`SimpleCalc`**. Ini digunakan untuk menghasilkan namespace Ruby Gem dari API Anda. Nama harus semua huruf, (`a-zA-Z`), tanpa karakter atau angka khusus lainnya.
   b. Untuk Nama Permata Ruby, tentukan nama Ruby Gem yang berisi kode sumber SDK yang dihasilkan untuk API Anda. Secara default, ini adalah nama layanan dengan cache yang lebih rendah ditambah `-sdk` sufiks—misalnya,. **`simplecalc-sdk`**
   c. Untuk Ruby Gem Version, tentukan nomor versi untuk Ruby Gem yang dihasilkan. Secara default, nilainya diatur ke `1.0.0`.
7. Pilih Hasilkan SDK, lalu ikuti petunjuk di layar untuk mengunduh SDK yang dihasilkan oleh API Gateway.

Ikuti petunjuk Menggunakan Ruby SDK yang dihasilkan oleh API Gateway untuk REST API untuk menggunakan SDK yang dihasilkan.

Setiap kali memperbarui API, Anda harus menerapkan ulang API dan membuat ulang SDK agar pembaruan disertakan.

Menghasilkan SDK untuk API menggunakan perintah AWS CLI

Anda dapat menggunakan AWS CLI untuk membuat dan mengunduh SDK API untuk platform yang didukung dengan memanggil perintah get-sdk. Kami mendemonstrasikan ini untuk beberapa platform yang didukung berikut ini.

Topik

Buat dan unduh Java untuk Android SDK menggunakan AWS CLI

Untuk membuat dan mengunduh Java untuk Android SDK yang dihasilkan oleh API Gateway dari API (`udpuvvzbkc`) pada tahap tertentu (`test`), panggil perintah sebagai berikut:

```
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
```

Input terakhir dari `~/apps/myApi/myApi-android-sdk.zip` adalah path ke file SDK yang diunduh bernama`myApi-android-sdk.zip`.

Buat dan unduh JavaScript SDK menggunakan AWS CLI

Untuk membuat dan mengunduh JavaScript SDK yang dihasilkan oleh API Gateway dari API (`udpuvvzbkc`) pada tahap tertentu (`test`), panggil perintah sebagai berikut:

```
aws apigateway get-sdk \
            --rest-api-id udpuvvzbkc \
            --stage-name test \
            --sdk-type javascript \
            ~/apps/myApi/myApi-js-sdk.zip
```

Input terakhir dari `~/apps/myApi/myApi-js-sdk.zip` adalah path ke file SDK yang diunduh bernama`myApi-js-sdk.zip`.

Buat dan unduh Ruby SDK menggunakan AWS CLI

Untuk membuat dan mengunduh Ruby SDK dari API (`udpuvvzbkc`) pada tahap tertentu (`test`), panggil perintah sebagai berikut:

```
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
```

Input terakhir dari `~/apps/myApi/myApi-ruby-sdk.zip` adalah path ke file SDK yang diunduh bernama`myApi-ruby-sdk.zip`.

Selanjutnya, kami menunjukkan cara menggunakan SDK yang dihasilkan untuk memanggil API yang mendasarinya. Untuk informasi selengkapnya, lihat Panggil REST API melalui SDK yang dihasilkan.

Fungsi Lambda Kalkulator Sederhana

Sebagai ilustrasi, kita akan menggunakan fungsi Node.js Lambda yang melakukan operasi biner penambahan, pengurangan, perkalian dan pembagian.

Topik

- Kalkulator sederhana Format input fungsi Lambda
- Kalkulator sederhana Format output fungsi Lambda
- Implementasi fungsi Lambda kalkulator sederhana

Kalkulator sederhana Format input fungsi Lambda

Fungsi ini mengambil masukan dari format berikut:

```
{ "a": "Number", "b": "Number", "op": "string"}
```

di manaop bisa salah satu`(+, -, *, /, add, sub, mul, div)`.

Kalkulator sederhana Format output fungsi Lambda

Ketika operasi berhasil, ia mengembalikan hasil dari format berikut:

```
{ "a": "Number", "b": "Number", "op": "string", "c": "Number"}
```

dimanac memegang hasil perhitungan.

Implementasi fungsi Lambda kalkulator sederhana

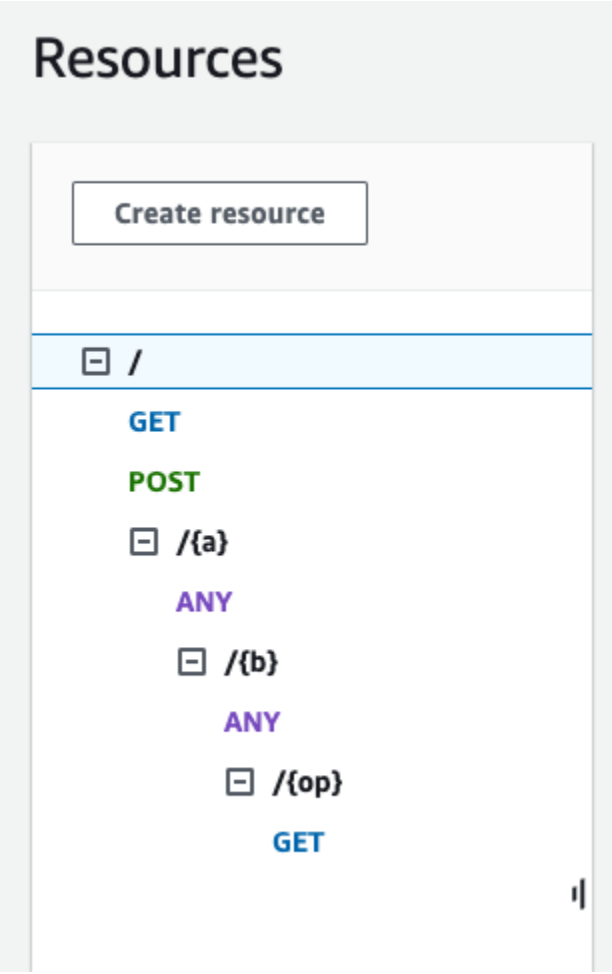
Implementasi fungsi Lambda adalah sebagai berikut:

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

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

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

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

  return res;
};
```

API kalkulator sederhana di API Gateway

API kalkulator sederhana kami mengekspos tiga metode (GET, POST, GET) untuk memanggil. the section called “Fungsi Lambda Kalkulator Sederhana” Representasi grafis dari API ini ditampilkan sebagai berikut:



Ketiga metode ini menunjukkan cara berbeda untuk memasok input untuk fungsi Lambda backend untuk melakukan operasi yang sama:

- `GET /?a=...&b=...&op=...`Metode ini menggunakan parameter kueri untuk menentukan input.
- `POST /`Metode ini menggunakan payload JSON `{"a":"Number", "b":"Number", "op":"string"}` untuk menentukan input.
- `GET /{a}/{b}/{op}`Metode ini menggunakan parameter jalur untuk menentukan input.

Jika tidak ditentukan, API Gateway menghasilkan nama metode SDK yang sesuai dengan menggabungkan metode HTTP dan bagian jalur. Bagian jalur root (`/`) disebut sebagai`Api Root`. Misalnya, nama metode Java SDK default untuk metode API `GET /?a=...&b=...&op=...` adalah`getAB0p`, nama metode SDK default untuk `POST /` adalah`postApiRoot`, dan nama metode SDK default untuk adalah. `GET /{a}/{b}/{op}` `getAB0p` SDK individu dapat menyesuaikan konvensi. Lihat dokumentasi di sumber SDK yang dihasilkan untuk nama metode khusus SDK.

Anda dapat, dan harus, mengganti nama metode SDK default dengan menentukan properti operationName pada setiap metode API. Anda dapat melakukannya saat membuat metode API atau memperbarui metode API menggunakan API Gateway REST API. Dalam definisi API Swagger, Anda dapat mengatur `operationId` untuk mencapai hasil yang sama.

Sebelum menunjukkan cara memanggil metode ini menggunakan SDK yang dihasilkan oleh API Gateway untuk API ini, mari kita ingat secara singkat cara mengaturnya. Untuk instruksi detail, lihat Membuat REST API di Amazon API Gateway. Jika Anda baru mengenal API Gateway, lihat Bangun API REST API Gateway dengan integrasi Lambda terlebih dahulu.

Buat model untuk input dan output

Untuk menentukan input yang diketik dengan kuat di SDK, kami membuat `Input` model untuk API. Untuk menggambarkan tipe data badan respons, kami membuat `Output` model dan `Result` model.

Untuk membuat model untuk input, output, dan hasil

1. Di panel navigasi utama, pilih Model.
2. Pilih Buat model.
3. Untuk Nama, masukkan **`input`**.
4. Untuk jenis Konten, masukkan**`application/json`**.

   Jika tidak ada jenis konten yang cocok ditemukan, validasi permintaan tidak dilakukan. Untuk menggunakan model yang sama terlepas dari jenis konten, masukkan**`$default`**.
5. Untuk skema Model, masukkan model berikut:

```
{
    "$schema" : "$schema": "http://json-schema.org/draft-04/schema#",
    "type":"object",
    "properties":{
        "a":{"type":"number"},
        "b":{"type":"number"},
        "op":{"type":"string"}
    },
    "title":"Input"
}
```

6. Pilih Buat model.
7. Ulangi langkah-langkah berikut untuk membuat `Output` model dan `Result` model.

Untuk `Output` model, masukkan yang berikut ini untuk skema Model:

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "properties": {
        "c": {"type":"number"}
    },
    "title": "Output"
}
```

Untuk `Result` model, masukkan yang berikut ini untuk skema Model. Ganti ID API `abc123` dengan ID API Anda.

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type":"object",
    "properties":{
        "input":{
            "$ref":"https://apigateway.amazonaws.com/restapis/abc123/models/Input"
        },
        "output":{
            "$ref":"https://apigateway.amazonaws.com/restapis/abc123/models/Output"
        }
    },
    "title":"Result"
}
```

Siapkan GET/parameter kueri metode

Untuk `GET /?a=..&b=..&op=..` metode ini, parameter kueri dideklarasikan dalam Permintaan Metode:

Untuk mengatur parameter string kueri GET/URL

1. Di bagian Permintaan metode untuk `GET` metode pada sumber daya root (`/`), pilih Edit.
2. Pilih parameter string kueri URL dan lakukan hal berikut:

    a. Pilih Tambahkan string kueri.

b. Untuk Nama, masukkan **a**.

c. Tetap Diperlukan dan Caching dimatikan.

d. Tetap caching dimatikan.

Ulangi langkah yang sama dan buat string kueri bernama **b** dan string kueri bernama**op**.

3. Pilih Save (Simpan).

Siapkan model data untuk payload sebagai masukan ke backend

Untuk `POST /` metode ini, kita membuat `Input` model dan menambahkannya ke permintaan metode untuk menentukan bentuk data input.

Untuk mengatur model data untuk payload sebagai masukan ke backend

1. Di bagian Permintaan metode, untuk `POST` metode pada sumber daya root (`/`) pilih Edit.
2. Pilih badan Permintaan.
3. Pilih Tambah model.
4. Untuk jenis Konten, masukkan**`application/json`**.
5. Untuk Model, pilih Input.
6. Pilih Save (Simpan).

Dengan model ini, pelanggan API Anda dapat memanggil SDK untuk menentukan input dengan membuat instance objek. `Input` Tanpa model ini, pelanggan Anda akan diminta untuk membuat objek kamus untuk mewakili input JSON ke fungsi Lambda.

Mengatur model data untuk output hasil dari backend

Untuk ketiga metode, kita membuat `Result` model dan menambahkannya ke metode `Method Response` untuk menentukan bentuk output yang dikembalikan oleh fungsi Lambda.

Untuk mengatur model data untuk output hasil dari backend

1. Pilih sumber daya/{a}/{b}/{op}, lalu pilih metode GET.
2. Pada tab Respons Metode, di bawah Respons 200, pilih Edit.
3. Di bawah Badan respons, pilih Tambahkan model.
4. Untuk jenis Konten, masukkan**`application/json`**.

5. Untuk Model, pilih Hasil.
6. Pilih Save (Simpan).

Dengan model ini, pelanggan API Anda dapat mengurai output yang berhasil dengan membaca properti `Result` objek. Tanpa model ini, pelanggan akan diminta untuk membuat objek kamus untuk mewakili output JSON.

Definisi OpenAPI API kalkulator sederhana

Berikut ini adalah definisi OpenAPI dari API kalkulator sederhana. Anda dapat mengimpornya ke akun Anda. Namun, Anda perlu mengatur ulang izin berbasis sumber daya pada fungsi Lambda setelah impor. Untuk melakukannya, pilih kembali fungsi Lambda yang Anda buat di akun dari Permintaan Integrasi di konsol API Gateway. Ini akan menyebabkan konsol API Gateway mengatur ulang izin yang diperlukan. Atau, Anda dapat menggunakan AWS Command Line Interface untuk Lambda perintah 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{\n
  \"a\" : $input.params('a'),\n  \"b\" : $input.params('b'),\n  \"op\" :
 \"$input.params('op')\"\n}"
          },
          "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:Calc/invocations",
          "passthroughBehavior": "when_no_templates",
          "httpMethod": "POST",
          "responses": {
            "default": {
              "statusCode": "200",
              "responseTemplates": {
                "application/json": "#set($inputRoot = $input.path('$'))\n{\n
 \"input\" : {\n    \"a\" : $inputRoot.a,\n    \"b\" : $inputRoot.b,\n    \"op\" :
 \"$inputRoot.op\"\n  },\n  \"output\" : {\n    \"c\" : $inputRoot.c\n  }\n}"
              }
            }
```

```
          },
          "type": "aws"
        }
      },
      "post": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "in": "body",
            "name": "Input",
            "required": true,
            "schema": {
              "$ref": "#/definitions/Input"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Result"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:Calc/invocations",
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST",
          "responses": {
            "default": {
              "statusCode": "200",
              "responseTemplates": {
                "application/json": "#set($inputRoot = $input.path('$'))\n{\n
 \"input\" : {\n    \"a\" : $inputRoot.a,\n    \"b\" : $inputRoot.b,\n    \"op\" :
 \"$inputRoot.op\"\n  },\n  \"output\" : {\n    \"c\" : $inputRoot.c\n  }\n}"
              }
            }
          },
```

```
          "type": "aws"
        }
      }
    },
    "/{a}": {
      "x-amazon-apigateway-any-method": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "a",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "404": {
            "description": "404 response"
          }
        },
        "x-amazon-apigateway-integration": {
          "requestTemplates": {
            "application/json": "{\"statusCode\": 200}"
          },
          "passthroughBehavior": "when_no_match",
          "responses": {
            "default": {
              "statusCode": "404",
              "responseTemplates": {
                "application/json": "{ \"Message\" : \"Can't $context.httpMethod
$context.resourcePath\" }"
              }
            }
          },
          "type": "mock"
        }
      }
    },
    "/{a}/{b}": {
```

```
      "x-amazon-apigateway-any-method": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "a",
            "in": "path",
            "required": true,
            "type": "string"
          },
          {
            "name": "b",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "404": {
            "description": "404 response"
          }
        },
        "x-amazon-apigateway-integration": {
          "requestTemplates": {
            "application/json": "{\"statusCode\": 200}"
          },
          "passthroughBehavior": "when_no_match",
          "responses": {
            "default": {
              "statusCode": "404",
              "responseTemplates": {
                "application/json": "{ \"Message\" : \"Can't $context.httpMethod
$context.resourcePath\" }"
              }
            }
          },
          "type": "mock"
        }
      }
    },
```

```
    "/{a}/{b}/{op}": {
      "get": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "a",
            "in": "path",
            "required": true,
            "type": "string"
          },
          {
            "name": "b",
            "in": "path",
            "required": true,
            "type": "string"
          },
          {
            "name": "op",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Result"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "requestTemplates": {
            "application/json": "#set($inputRoot = $input.path('$'))\n{\n
  \"a\" : $input.params('a'),\n  \"b\" : $input.params('b'),\n  \"op\" :
 \"$input.params('op')\"\n}"
          },
          "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:Calc/invocations",
```

```
          "passthroughBehavior": "when_no_templates",
          "httpMethod": "POST",
          "responses": {
            "default": {
              "statusCode": "200",
              "responseTemplates": {
                "application/json": "#set($inputRoot = $input.path('$'))\n{\n
\"input\" : {\n    \"a\" : $inputRoot.a,\n    \"b\" : $inputRoot.b,\n    \"op\" :
\"$inputRoot.op\"\n  },\n  \"output\" : {\n    \"c\" : $inputRoot.c\n  }\n}"
              }
            }
          },
          "type": "aws"
        }
      }
    }
  },
  "definitions": {
    "Input": {
      "type": "object",
      "properties": {
        "a": {
          "type": "number"
        },
        "b": {
          "type": "number"
        },
        "op": {
          "type": "string"
        }
      },
      "title": "Input"
    },
    "Output": {
      "type": "object",
      "properties": {
        "c": {
          "type": "number"
        }
      },
      "title": "Output"
    },
    "Result": {
      "type": "object",
```

```
      "properties": {
        "input": {
          "$ref": "#/definitions/Input"
        },
        "output": {
          "$ref": "#/definitions/Output"
        }
      },
      "title": "Result"
    }
  }
}
```

OpenAPI 3.0

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "SimpleCalc",
    "version" : "2016-09-29T20:27:30Z"
  },
  "servers" : [ {
    "url" : "https://t6dve4zn25.execute-api.us-west-2.amazonaws.com/{basePath}",
    "variables" : {
      "basePath" : {
        "default" : "demo"
      }
    }
  } ],
  "paths" : {
    "/{a}/{b}" : {
      "x-amazon-apigateway-any-method" : {
        "parameters" : [ {
          "name" : "a",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "b",
          "in" : "path",
          "required" : true,
```

```
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "404" : {
            "description" : "404 response",
            "content" : { }
          }
        },
        "x-amazon-apigateway-integration" : {
          "type" : "mock",
          "responses" : {
            "default" : {
              "statusCode" : "404",
              "responseTemplates" : {
                "application/json" : "{ \"Message\" : \"Can't $context.httpMethod
$context.resourcePath\" }"
              }
            }
          },
          "requestTemplates" : {
            "application/json" : "{\"statusCode\": 200}"
          },
          "passthroughBehavior" : "when_no_match"
        }
      }
    },
    "/{a}/{b}/{op}" : {
      "get" : {
        "parameters" : [ {
          "name" : "a",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "b",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
```

```
        }, {
          "name" : "op",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/Result"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : {
          "type" : "aws",
          "httpMethod" : "POST",
          "uri" : "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:111122223333:function:Calc/invocations",
          "responses" : {
            "default" : {
              "statusCode" : "200",
              "responseTemplates" : {
                "application/json" : "#set($inputRoot = $input.path('$'))\n{\n
 \"input\" : {\n    \"a\" : $inputRoot.a,\n    \"b\" : $inputRoot.b,\n    \"op\" :
 \"$inputRoot.op\"\n  },\n  \"output\" : {\n    \"c\" : $inputRoot.c\n  }\n}"
              }
            }
          },
          "requestTemplates" : {
            "application/json" : "#set($inputRoot = $input.path('$'))\n{\n
  \"a\" : $input.params('a'),\n  \"b\" : $input.params('b'),\n  \"op\" :
 \"$input.params('op')\"\n}"
          },
          "passthroughBehavior" : "when_no_templates"
        }
      }
    },
```

```
    "/" : {
      "get" : {
        "parameters" : [ {
          "name" : "op",
          "in" : "query",
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "a",
          "in" : "query",
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "b",
          "in" : "query",
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/Result"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : {
          "type" : "aws",
          "httpMethod" : "POST",
          "uri" : "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:111122223333:function:Calc/invocations",
          "responses" : {
            "default" : {
              "statusCode" : "200",
              "responseTemplates" : {
```

```
                "application/json" : "#set($inputRoot = $input.path('$'))\n{\n
 \"input\" : {\n    \"a\" : $inputRoot.a,\n    \"b\" : $inputRoot.b,\n    \"op\" :
 \"$inputRoot.op\"\n  },\n  \"output\" : {\n    \"c\" : $inputRoot.c\n  }\n}"
              }
            }
          },
          "requestTemplates" : {
            "application/json" : "#set($inputRoot = $input.path('$'))\n{\n
  \"a\" : $input.params('a'),\n  \"b\" : $input.params('b'),\n  \"op\" :
 \"$input.params('op')\"\n}"
          },
          "passthroughBehavior" : "when_no_templates"
        }
      },
      "post" : {
        "requestBody" : {
          "content" : {
            "application/json" : {
              "schema" : {
                "$ref" : "#/components/schemas/Input"
              }
            }
          },
          "required" : true
        },
        "responses" : {
          "200" : {
            "description" : "200 response",
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/Result"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : {
          "type" : "aws",
          "httpMethod" : "POST",
          "uri" : "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:111122223333:function:Calc/invocations",
          "responses" : {
            "default" : {
```

```
            "statusCode" : "200",
            "responseTemplates" : {
              "application/json" : "#set($inputRoot = $input.path('$'))\n{\n
\"input\" : {\n    \"a\" : $inputRoot.a,\n    \"b\" : $inputRoot.b,\n    \"op\" :
\"$inputRoot.op\"\n  },\n  \"output\" : {\n    \"c\" : $inputRoot.c\n  }\n}"
            }
          }
        },
        "passthroughBehavior" : "when_no_match"
      }
    }
  },
  "/{a}" : {
    "x-amazon-apigateway-any-method" : {
      "parameters" : [ {
        "name" : "a",
        "in" : "path",
        "required" : true,
        "schema" : {
          "type" : "string"
        }
      } ],
      "responses" : {
        "404" : {
          "description" : "404 response",
          "content" : { }
        }
      },
      "x-amazon-apigateway-integration" : {
        "type" : "mock",
        "responses" : {
          "default" : {
            "statusCode" : "404",
            "responseTemplates" : {
              "application/json" : "{ \"Message\" : \"Can't $context.httpMethod
$context.resourcePath\" }"
            }
          }
        },
        "requestTemplates" : {
          "application/json" : "{\"statusCode\": 200}"
        },
        "passthroughBehavior" : "when_no_match"
      }
```

```
      }
    }
  },
  "components" : {
    "schemas" : {
      "Input" : {
        "title" : "Input",
        "type" : "object",
        "properties" : {
          "a" : {
            "type" : "number"
          },
          "b" : {
            "type" : "number"
          },
          "op" : {
            "type" : "string"
          }
        }
      },
      "Output" : {
        "title" : "Output",
        "type" : "object",
        "properties" : {
          "c" : {
            "type" : "number"
          }
        }
      },
      "Result" : {
        "title" : "Result",
        "type" : "object",
        "properties" : {
          "input" : {
            "$ref" : "#/components/schemas/Input"
          },
          "output" : {
            "$ref" : "#/components/schemas/Output"
          }
        }
      }
    }
  }
```

```
}
```

Jual API Gateway API Anda melaluiAWS Marketplace

Setelah membangun, dan menguji, dan men-deploy API Gatewayrencana penggunaandan menjual paket sebagai produk Software as a Service (SaaS) melaluiAWS Marketplace. Pembeli API yang berlangganan penawaran produk Anda ditagih olehAWS Marketplaceberdasarkan jumlah permintaan yang dibuat untuk rencana penggunaan.

Untuk menjual API AndaAWS Marketplace, Anda harus menyiapkan saluran penjualan untuk diintegrasikanAWS Marketplacedengan API Gateway. Secara umum, ini melibatkan daftar produk AndaAWS Marketplace, menyiapkan peran IAM dengan kebijakan yang sesuai untuk memungkinkan API Gateway mengirim metrik penggunaanAWS Marketplace, mengasosiasikanAWS Marketplaceproduk dengan paket penggunaan API Gateway, dan mengaitkanAWS Marketplacepembeli dengan kunci API Gateway. Rincian dibahas di bagian berikut.

Untuk informasi lebih lanjut tentang menjual API Anda sebagai produk SaaSAWS Marketplace, lihatAWS MarketplacePanduan Pengguna.

Topik

- InisialisasiAWS Marketplaceintegrasi dengan API Gateway
- Menangani langganan pelanggan ke paket penggunaan

InisialisasiAWS Marketplaceintegrasi dengan API Gateway

Tugas berikut adalah untuk inisialisasi satu kaliAWS Marketplaceintegrasi dengan API Gateway, yang memungkinkan Anda menjual API Anda sebagai produk SaaS.

Daftar produk padaAWS Marketplace

Untuk mencantumkan paket penggunaan Anda sebagai produk SaaS, kirimkan formulir pemuatan produk melaluiAWS Marketplace. Produk harus berisi dimensi bernama`apigateway`dari`requests`jenis. Dimensi ini mendefinisikan price-per-request dan digunakan oleh API Gateway untuk meteran permintaan untuk API Anda.

Buat peran pengukuran

Buat IAM role bernama`ApiGatewayMarketplaceMeteringRole`dengan kebijakan eksekusi dan kebijakan kepercayaan berikut. Peran ini memungkinkan API Gateway untuk mengirim metrik penggunaan keAWS Marketplaceatas nama Anda.

Kebijakan eksekusi peran metering

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "aws-marketplace:BatchMeterUsage",
        "aws-marketplace:ResolveCustomer"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

Kebijakan hubungan tepercaya dari peran pengukuran

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

Mengaitkan rencana penggunaan denganAWS Marketplaceproduk

Saat Anda mencantumkan produk diAWS Marketplace, Anda menerimaAWS Marketplacekode produk. Untuk mengintegrasikan API Gateway denganAWS Marketplace, kaitkan rencana penggunaan Anda denganAWS Marketplacekode produk. Anda mengaktifkan asosiasi dengan

menetapkan API GatewayUsagePlan'sproductCodebidang untuk AndaAWS Marketplacekode produk, menggunakan konsol API Gateway, API Gateway REST API,AWS CLIuntuk API Gateway, atauAWSSDK untuk API Gateway. Contoh kode berikut menggunakan API Gateway:

```
PATCH /usageplans/USAGE_PLAN_ID
Host: apigateway.region.amazonaws.com
Authorization: ...

{
    "patchOperations" : [{
        "path" : "/productCode",
        "value" : "MARKETPLACE_PRODUCT_CODE",
        "op" : "replace"
    }]
}
```

Menangani langganan pelanggan ke paket penggunaan

Tugas berikut ditangani oleh aplikasi portal pengembang Anda.

Ketika pelanggan berlangganan produk Anda melaluiAWS Marketplace,AWS Marketplaceke depanPOSTmeminta ke URL langganan SaaS yang Anda daftarkan saat mendaftarkan produk AndaAWS Marketplace. YangPOSTpermintaan dilengkapi dengan`x-amzn-marketplace-token`parameter yang berisi informasi pembeli. Ikuti instruksi diOrientasi pelanggan SaaSuntuk menangani pengalihan ini di aplikasi portal pengembang Anda.

Menanggapi permintaan berlangganan pelanggan,AWS Marketplacemengirimkan`subscribe-success`pemberitahuan untuk topik Amazon SNS yang dapat Anda berlangganan. (LihatOrientasi pelanggan SaaS). Untuk menerima permintaan berlangganan pelanggan, Anda menangani`subscribe-success`pemberitahuan dengan membuat atau mengambil kunci API Gateway API untuk pelanggan, mengaitkan pelangganAWS Marketplace-provisioned`customerId`dengan kunci API, dan kemudian menambahkan kunci API ke paket penggunaan Anda.

Ketika permintaan berlangganan pelanggan selesai, aplikasi portal pengembang harus menyajikan pelanggan dengan kunci API terkait dan memberi tahu pelanggan bahwa kunci API harus disertakan dalam`x-api-key`header dalam permintaan ke API.

Ketika pelanggan membatalkan langganan ke paket penggunaan,AWS Marketplacemengirimkan`unsubscribe-success`pemberitahuan untuk topik SNS. Untuk

menyelesaikan proses berhenti berlangganan pelanggan, Anda menanganiunsubscribe-successnotifikasi dengan menghapus kunci API pelanggan dari paket penggunaan.

Mengizinkan pelanggan untuk mengakses paket penggunaan

Untuk mengotorisasi akses ke paket penggunaan Anda untuk pelanggan tertentu, gunakan API Gateway API untuk mengambil atau membuat kunci API untuk pelanggan dan menambahkan kunci API ke paket penggunaan.

Contoh berikut menunjukkan cara memanggil API Gateway REST API untuk membuat kunci API baru dengan spesifikAWS Marketplace customerIdvalue (*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"
}
```

Contoh berikut menunjukkan cara mendapatkan kunci API dengan spesifikAWS Marketplace customerIdvalue (*MARKETPLACE_CUSTOMER_ID*).

```
GET apikeys?customerId=MARKETPLACE_CUSTOMER_ID HTTP/1.1
Host: apigateway.region.amazonaws.com
Authorization: ...
```

Untuk menambahkan kunci API ke paket penggunaan, buatUsagePlanKeydengan kunci API untuk rencana penggunaan yang relevan. Contoh berikut menunjukkan cara melakukannya menggunakan API Gatewayn371ptadalah ID paket penggunaan danq5ugs7qjjhadalah contoh APIkeyIdkembali dari contoh sebelumnya.

```
POST /usageplans/n371pt/keys HTTP/1.1
Host: apigateway.region.amazonaws.com
```

```
Authorization: ...

{
    "keyId": "q5ugs7qjjh",
    "keyType": "API_KEY"
}
```

Mengaitkan pelanggan dengan kunci API

Anda harus memperbaruiApiKey'scustomerIdkeAWS MarketplaceID pelanggan pelanggan. Ini mengaitkan kunci API denganAWS Marketplacepelanggan, yang memungkinkan metering dan penagihan untuk pembeli. Contoh kode berikut memanggil API Gateway REST API untuk melakukannya.

```
PATCH /apikeys/q5ugs7qjjh
Host: apigateway.region.amazonaws.com
Authorization: ...

{
    "patchOperations" : [{
        "path" : "/customerId",
        "value" : "MARKETPLACE_CUSTOMER_ID",
        "op" : "replace"
    }]
}
```

Melindungi REST API

API Gateway menyediakan sejumlah cara untuk melindungi API Anda dari ancaman tertentu, seperti pengguna berbahaya atau lonjakan lalu lintas. Anda dapat melindungi API Anda menggunakan strategi seperti menghasilkan sertifikat SSL, mengkonfigurasi firewall aplikasi web, menetapkan target throttling, dan hanya memungkinkan akses ke API Anda dari Virtual Private Cloud (VPC). Pada bagian ini Anda dapat mempelajari cara mengaktifkan kemampuan ini menggunakan API Gateway.

Topik

- Mengonfigurasi otentikasi TLS timbal balik untuk REST API
- Menghasilkan dan mengkonfigurasi sertifikat SSL untuk otentikasi backend
- Menggunakan AWS WAF untuk melindungi API Anda
- Permintaan Throttle API untuk throughput yang lebih baik

- Membuat API pribadi di Amazon API Gateway

Mengonfigurasi otentikasi TLS timbal balik untuk REST API

Mutual TLS otentikasi membutuhkan otentikasi dua arah antara klien dan server. Dengan TLS bersama, klien harus menunjukkan sertifikat X.509 untuk memverifikasi identitas mereka untuk mengakses API Anda. Mutual TLS adalah persyaratan umum untuk Internet of Things (IoT) business-to-business dan aplikasi.

Anda dapat menggunakan TLS bersama dengan operasi otorisasi dan otentikasi lain yang didukung API Gateway. API Gateway meneruskan sertifikat yang disediakan klien kepada otorisasi Lambda dan integrasi backend.

> **Important**
>
> Secara default, klien dapat memanggil API Anda dengan menggunakan `execute-api` titik akhir yang dihasilkan API Gateway untuk API Anda. Untuk memastikan bahwa klien dapat mengakses API Anda hanya dengan menggunakan nama domain khusus dengan TLS timbal balik, nonaktifkan titik `execute-api` akhir default. Untuk mempelajari selengkapnya, lihat the section called “Nonaktifkan titik akhir default”.

Topik

- Prasyarat untuk TLS timbal balik
- Mengkonfigurasi TLS timbal balik untuk nama domain khusus
- Memanggil API dengan menggunakan nama domain khusus yang membutuhkan TLS timbal balik
- Memperbarui truststore Anda
- Nonaktifkan TLS timbal balik
- Memecahkan masalah peringatan sertifikat
- Memecahkan masalah konflik nama domain
- Memecahkan masalah pesan status nama domain

Prasyarat untuk TLS timbal balik

Untuk mengkonfigurasi TLS timbal balik yang Anda butuhkan:

- Sebuah nama domain kustom
- Setidaknya satu sertifikat yang dikonfigurasi AWS Certificate Manager untuk nama domain kustom Anda
- Truststore yang dikonfigurasi dan diunggah ke Amazon S3

Nama domain kustom

Untuk mengaktifkan TLS bersama untuk REST API, Anda harus mengonfigurasi nama domain khusus untuk API Anda. Anda dapat mengaktifkan TLS timbal balik untuk nama domain khusus, dan kemudian memberikan nama domain khusus kepada klien. Untuk mengakses API dengan menggunakan nama domain kustom yang mengaktifkan TLS bersama, klien harus menunjukkan sertifikat yang Anda percayai dalam permintaan API. Anda dapat menemukan informasi lebih lanjut dithe section called “Nama domain kustom”.

Menggunakan sertifikat AWS Certificate Manager yang dikeluarkan

Anda dapat meminta sertifikat tepercaya publik langsung dari ACM atau mengimpor sertifikat publik atau yang ditandatangani sendiri. Untuk menyiapkan sertifikat di ACM, buka ACM. Jika Anda ingin mengimpor sertifikat, lanjutkan membaca di bagian berikut.

Menggunakan impor atau AWS Private Certificate Authority sertifikat

Untuk menggunakan sertifikat yang diimpor ke ACM atau sertifikat dari TLS AWS Private Certificate Authority bersama, API Gateway memerlukan yang `ownershipVerificationCertificate` dikeluarkan oleh ACM. Sertifikat kepemilikan ini hanya digunakan untuk memverifikasi bahwa Anda memiliki izin untuk menggunakan nama domain. Ini tidak digunakan untuk jabat tangan TLS. Jika Anda belum memiliki`ownershipVerificationCertificate`, buka https://console.aws.amazon.com/acm/ untuk mengaturnya.

Anda harus menjaga sertifikat ini tetap berlaku selama masa pakai nama domain Anda. Jika sertifikat kedaluwarsa dan perpanjangan otomatis gagal, semua pembaruan pada nama domain akan dikunci. Anda perlu memperbarui `ownershipVerificationCertificateArn` dengan valid `ownershipVerificationCertificate` sebelum Anda dapat membuat perubahan lainnya. Ini `ownershipVerificationCertificate` tidak dapat digunakan sebagai sertifikat server untuk domain TLS timbal balik lainnya di API Gateway. Jika sertifikat langsung diimpor kembali ke ACM, penerbit harus tetap sama.

Mengkonfigurasi truststore Anda

Truststores adalah file teks dengan ekstensi `.pem` file. Mereka adalah daftar sertifikat tepercaya dari Otoritas Sertifikat. Untuk menggunakan TLS timbal balik, buat truststore sertifikat X.509 yang Anda percayai untuk mengakses API Anda.

Anda harus menyertakan rantai kepercayaan lengkap, mulai dari sertifikat CA penerbitan, hingga sertifikat CA root, di truststore Anda. API Gateway menerima sertifikat klien yang dikeluarkan oleh CA mana pun yang ada dalam rantai kepercayaan. Sertifikat dapat dari otoritas sertifikat publik atau swasta. Sertifikat dapat memiliki panjang rantai maksimum empat. Anda juga dapat memberikan sertifikat yang ditandatangani sendiri. Algoritma hashing berikut didukung di truststore:

- SHA-256 atau lebih kuat
- RSA-2048 atau lebih kuat
- ECDSA-256 atau lebih kuat

API Gateway memvalidasi sejumlah properti sertifikat. Anda dapat menggunakan otorisasi Lambda untuk melakukan pemeriksaan tambahan saat klien memanggil API, termasuk memeriksa apakah sertifikat telah dicabut. API Gateway memvalidasi properti berikut:

Validasi	Deskripsi
Sintaks X.509	Sertifikat harus memenuhi persyaratan sintaks X.509.
Integritas	Konten sertifikat tidak boleh diubah dari yang ditandatangani oleh otoritas sertifikat dari truststore.
Validitas	Masa berlaku sertifikat harus terkini.
Nama rantai/rantai kunci	Nama dan subjek sertifikat harus membentuk rantai yang tidak terputus. Sertifikat dapat memiliki panjang rantai maksimum empat.

Unggah truststore ke bucket Amazon S3 dalam satu file

Berikut ini adalah contoh dari apa file.pem mungkin terlihat seperti.

Example sertifikat.pem

```
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
...
```

AWS CLI Perintah berikut akan diunggah `certificates.pem` ke bucket Amazon S3 Anda.

```
aws s3 cp certificates.pem s3://bucket-name
```

Bucket Amazon S3 Anda harus memiliki izin baca untuk API Gateway agar API Gateway dapat mengakses truststore Anda.

Mengkonfigurasi TLS timbal balik untuk nama domain khusus

Untuk mengonfigurasi TLS bersama untuk REST API, Anda harus menggunakan nama domain kustom Regional untuk API Anda, dengan versi TLS minimal 1.2. Untuk mempelajari selengkapnya tentang membuat dan mengonfigurasi nama domain kustom, lihatthe section called “Menyiapkan nama domain kustom regional”.

> **Note**
>
> Mutual TLS tidak didukung untuk API pribadi.

Setelah Anda mengunggah truststore Anda ke Amazon S3, Anda dapat mengonfigurasi nama domain khusus Anda untuk menggunakan TLS bersama. Tempelkan yang berikut (termasuk garis miring) ke terminal:

```
aws apigateway create-domain-name --region us-east-2 \
    --domain-name api.example.com \
    --regional-certificate-arn arn:aws:acm:us-
east-2:123456789012:certificate/123456789012-1234-1234-1234-12345678 \
    --endpoint-configuration types=REGIONAL \
```

```
    --security-policy TLS_1_2 \
    --mutual-tls-authentication truststoreUri=s3://bucket-name/key-name
```

Setelah membuat nama domain, Anda harus mengonfigurasi catatan DNS dan pemetaan basepath untuk operasi API. Untuk mempelajari selengkapnya, lihat Menyiapkan nama domain kustom regional di API Gateway.

Memanggil API dengan menggunakan nama domain khusus yang membutuhkan TLS timbal balik

Untuk menjalankan API dengan TLS timbal balik diaktifkan, klien harus menunjukkan sertifikat tepercaya dalam permintaan API. Saat klien mencoba menjalankan API Anda, API Gateway mencari penerbit sertifikat klien di truststore Anda. Agar API Gateway dapat melanjutkan permintaan, penerbit sertifikat dan rantai kepercayaan lengkap hingga sertifikat CA root harus ada di truststore Anda.

Contoh `curl` perintah berikut mengirimkan permintaan ke `api.example.com`, yang termasuk `my-cert.pem` dalam permintaan. `my-key.key`adalah kunci pribadi untuk sertifikat.

```
curl -v --key ./my-key.key --cert ./my-cert.pem api.example.com
```

API Anda dipanggil hanya jika truststore Anda mempercayai sertifikat. Kondisi berikut akan menyebabkan API Gateway gagal dalam jabat tangan TLS dan menolak permintaan dengan kode `403` status. Jika sertifikat Anda:

- tidak dipercaya
- kedaluwarsa
- tidak menggunakan algoritme yang didukung

> **Note**
>
> API Gateway tidak memverifikasi apakah sertifikat telah dicabut.

Memperbarui truststore Anda

Untuk memperbarui sertifikat di truststore Anda, unggah bundel sertifikat baru ke Amazon S3. Kemudian, Anda dapat memperbarui nama domain kustom Anda untuk menggunakan sertifikat yang diperbarui.

Gunakan versi Amazon S3 untuk mempertahankan beberapa versi truststore Anda. Saat memperbarui nama domain khusus untuk menggunakan versi truststore baru, API Gateway mengembalikan peringatan jika sertifikat tidak valid.

API Gateway menghasilkan peringatan sertifikat hanya ketika Anda memperbarui nama domain Anda. API Gateway tidak memberi tahu Anda jika sertifikat yang diunggah sebelumnya kedaluwarsa.

AWS CLI Perintah berikut memperbarui nama domain khusus untuk menggunakan versi truststore baru.

```
aws apigateway update-domain-name \
    --domain-name api.example.com \
    --patch-operations op='replace',path='/mutualTlsAuthentication/
truststoreVersion',value='abcdef123'
```

Nonaktifkan TLS timbal balik

Untuk menonaktifkan TLS timbal balik untuk nama domain kustom, hapus truststore dari nama domain kustom Anda, seperti yang ditunjukkan pada perintah berikut.

```
aws apigateway update-domain-name \
    --domain-name api.example.com \
    --patch-operations op='replace',path='/mutualTlsAuthentication/
truststoreUri',value=''
```

Memecahkan masalah peringatan sertifikat

Saat membuat nama domain khusus dengan TLS timbal balik, API Gateway mengembalikan peringatan jika sertifikat di truststore tidak valid. Ini juga dapat terjadi saat memperbarui nama domain khusus untuk menggunakan truststore baru. Peringatan menunjukkan masalah dengan sertifikat dan subjek sertifikat yang menghasilkan peringatan. Mutual TLS masih diaktifkan untuk API Anda, tetapi beberapa klien mungkin tidak dapat mengakses API Anda.

Anda harus memecahkan kode sertifikat di truststore Anda untuk mengidentifikasi sertifikat mana yang menghasilkan peringatan. Anda dapat menggunakan alat seperti `openssl` untuk memecahkan kode sertifikat dan mengidentifikasi subjek mereka.

Perintah berikut menampilkan isi sertifikat, termasuk subjeknya:

```
openssl x509 -in certificate.crt -text -noout
```

Perbarui atau hapus sertifikat yang menghasilkan peringatan, lalu unggah truststore baru ke Amazon S3. Setelah mengunggah truststore baru, perbarui nama domain kustom Anda untuk menggunakan truststore baru.

Memecahkan masalah konflik nama domain

Kesalahan `"The certificate subject <certSubject> conflicts with an existing certificate from a different issuer."` berarti beberapa Otoritas Sertifikat telah mengeluarkan sertifikat untuk domain ini. Untuk setiap subjek dalam sertifikat, hanya ada satu penerbit di API Gateway untuk domain TLS bersama. Anda harus mendapatkan semua sertifikat Anda untuk subjek itu melalui satu penerbit. Jika masalahnya adalah dengan sertifikat yang tidak Anda kendalikan tetapi Anda dapat membuktikan kepemilikan nama domain, hubungi AWS Support untuk membuka tiket.

Memecahkan masalah pesan status nama domain

`PENDING_CERTIFICATE_REIMPORT`: Ini berarti Anda mengimpor ulang sertifikat ke ACM dan validasi gagal karena sertifikat baru memiliki SAN (nama alternatif subjek) yang tidak tercakup oleh `ownershipVerificationCertificate` atau subjek atau SAN dalam sertifikat tidak mencakup nama domain. Sesuatu mungkin dikonfigurasi dengan tidak benar atau sertifikat yang tidak valid diimpor. Anda perlu mengimpor ulang sertifikat yang valid ke ACM. Untuk informasi selengkapnya tentang validasi, lihat Memvalidasi kepemilikan domain.

`PENDING_OWNERSHIP_VERIFICATION`: Ini berarti sertifikat Anda yang telah diverifikasi sebelumnya telah kedaluwarsa dan ACM tidak dapat memperbaruinya secara otomatis. Anda perlu memperbarui sertifikat atau meminta sertifikat baru. Informasi lebih lanjut tentang perpanjangan sertifikat dapat ditemukan di panduan perpanjangan sertifikat terkelola pemecahan masalah ACM.

Menghasilkan dan mengkonfigurasi sertifikat SSL untuk otentikasi backend

Anda dapat menggunakan API Gateway untuk menghasilkan sertifikat SSL dan kemudian menggunakan kunci publiknya di backend untuk memverifikasi bahwa permintaan HTTP ke sistem backend Anda berasal dari API Gateway. Hal ini memungkinkan backend HTTP Anda untuk mengontrol dan hanya menerima permintaan yang berasal dari Amazon API Gateway, bahkan jika backend dapat diakses publik.

> **Note**
>
> Beberapa server backend mungkin tidak mendukung otentikasi klien SSL seperti yang dilakukan API Gateway dan dapat mengembalikan kesalahan sertifikat SSL. Untuk daftar server backend yang tidak kompatibel, lihat. the section called “Catatan penting”

Sertifikat SSL yang dihasilkan oleh API Gateway ditandatangani sendiri, dan hanya kunci publik sertifikat yang terlihat di konsol API Gateway atau melalui API.

Topik

- Buat sertifikat klien menggunakan konsol API Gateway
- Konfigurasikan API untuk menggunakan sertifikat SSL
- Uji pemanggilan untuk memverifikasi konfigurasi sertifikat klien
- Konfigurasikan server HTTPS backend untuk memverifikasi sertifikat klien
- Putar sertifikat klien yang kedaluwarsa
- Otoritas sertifikat yang didukung API Gateway untuk integrasi proxy HTTP dan HTTP

Buat sertifikat klien menggunakan konsol API Gateway

1. Buka konsol API Gateway di https://console.aws.amazon.com/apigateway/.
2. Pilih REST API.
3. Di panel navigasi utama, pilih Sertifikat klien.
4. Dari halaman Sertifikat klien, pilih Hasilkan sertifikat.
5. (Opsional) Untuk Deskripsi, masukkan deskripsi.
6. Pilih Hasilkan sertifikat untuk menghasilkan sertifikat. API Gateway menghasilkan sertifikat baru dan mengembalikan GUID sertifikat baru, bersama dengan kunci publik yang dikodekan PEM.

Anda sekarang siap untuk mengkonfigurasi API untuk menggunakan sertifikat.

Konfigurasikan API untuk menggunakan sertifikat SSL

Instruksi ini mengasumsikan bahwa Anda sudah selesaiBuat sertifikat klien menggunakan konsol API Gateway.

1. Di konsol API Gateway, buat atau buka API yang ingin Anda gunakan sertifikat kliennya. Pastikan bahwa API telah diterapkan ke sebuah panggung.
2. Di panel navigasi utama, pilih Tahapan.
3. Di bagian Detail tahap, pilih Edit.
4. Untuk sertifikat Klien, pilih sertifikat.
5. Pilih Save changes (Simpan perubahan).

   Jika API telah digunakan sebelumnya di konsol API Gateway, Anda harus menerapkannya kembali agar perubahan diterapkan. Untuk informasi selengkapnya, lihat the section called "Menerapkan ulang REST API ke panggung".

Setelah sertifikat dipilih untuk API dan disimpan, API Gateway menggunakan sertifikat untuk semua panggilan ke integrasi HTTP di API Anda.

Uji pemanggilan untuk memverifikasi konfigurasi sertifikat klien

1. Pilih metode API. Pilih tab Uji. Anda mungkin perlu memilih tombol panah kanan untuk menampilkan tab Uji.
2. Untuk sertifikat Klien, pilih sertifikat.
3. Pilih Uji.

API Gateway menyajikan sertifikat SSL yang dipilih untuk backend HTTP untuk mengautentikasi API.

Konfigurasikan server HTTPS backend untuk memverifikasi sertifikat klien

Instruksi ini mengasumsikan bahwa Anda sudah menyelesaikan Buat sertifikat klien menggunakan konsol API Gateway dan mengunduh salinan sertifikat klien. Anda dapat mengunduh sertifikat klien dengan memanggil `clientcertificate:by-id`API Gateway REST API atau `get-client-certificate`dariAWS CLI.

Sebelum mengonfigurasi server HTTPS backend untuk memverifikasi sertifikat SSL klien API Gateway, Anda harus telah memperoleh kunci pribadi yang dikodekan PEM dan sertifikat sisi server yang disediakan oleh otoritas sertifikat tepercaya.

Jika nama domain server`myserver.mydomain.com`, nilai CNAME sertifikat server harus `myserver.mydomain.com` atau`*.mydomain.com`.

Otoritas sertifikat yang didukung termasuk Let's Encrypt atau salah satu dari. the section called "Otoritas sertifikat yang didukung untuk integrasi proxy HTTP dan HTTP"

Sebagai contoh, misalkan file sertifikat klien adalah `apig-cert.pem` dan kunci pribadi server dan file sertifikat adalah `server-key.pem` dan`server-cert.pem`, masing-masing. Untuk server Node.js di backend, Anda dapat mengonfigurasi server yang mirip dengan yang berikut ini:

```
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);
```

Untuk aplikasi node- express, Anda dapat menggunakan client-certificate-authmodul untuk mengautentikasi permintaan klien dengan sertifikat yang dikodekan PEM.

Untuk server HTTPS lainnya, lihat dokumentasi untuk server.

Putar sertifikat klien yang kedaluwarsa

Sertifikat klien yang dihasilkan oleh API Gateway berlaku selama 365 hari. Anda harus memutar sertifikat sebelum sertifikat klien pada tahap API kedaluwarsa untuk menghindari waktu henti API. Anda dapat memeriksa tanggal kedaluwarsa sertifikat dengan memanggil clientCertificate:by-ID API Gateway REST API AWS CLI atau perintah dan memeriksa properti ExpirationDate get-client-certificateyang dikembalikan.

Untuk memutar sertifikat klien, lakukan hal berikut:

1. Buat sertifikat klien baru dengan memanggil clientcertificate:generate API Gateway REST API atau perintah dari. AWS CLI generate-client-certificate Dalam tutorial ini, kita berasumsi bahwa ID sertifikat klien baru adalah`ndiqef`.

2. Perbarui server backend untuk menyertakan sertifikat klien baru. Jangan menghapus sertifikat klien yang ada.

   Beberapa server mungkin memerlukan restart untuk menyelesaikan pembaruan. Konsultasikan dokumentasi server untuk melihat apakah Anda harus me-restart server selama pembaruan.

3. Perbarui tahap API untuk menggunakan sertifikat klien baru dengan memanggil stage:update API Gateway REST API, dengan ID sertifikat klien baru (): `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"
       }
     ]
   }
   ```

   atau dengan memanggil perintah CLI dari tahap pembaruan.

4. Perbarui server backend untuk menghapus sertifikat lama.

5. Hapus sertifikat lama dari API Gateway dengan memanggil clientcertificate:delete API Gateway REST API, dengan menentukan clientCertificateId () `a1b2c3` sertifikat lama:

   ```
   DELETE /clientcertificates/a1b2c3
   ```

   atau dengan memanggil perintah CLI dari: delete-client-certificate

   ```
   aws apigateway delete-client-certificate --client-certificate-id a1b2c3
   ```

Untuk memutar sertifikat klien di konsol untuk API yang digunakan sebelumnya, lakukan hal berikut:

1. Di panel navigasi utama, pilih Sertifikat klien.

2. Dari panel Sertifikat klien, pilih Menghasilkan sertifikat.
3. Buka API yang ingin Anda gunakan sertifikat klien.
4. Pilih Tahapan di bawah API yang dipilih dan kemudian pilih panggung.
5. Di bagian Detail tahap, pilih Edit.
6. Untuk sertifikat Klien, pilih sertifikat baru.
7. Untuk menyimpan pengaturan, pilih Simpan perubahan.

   Anda perlu menerapkan ulang API agar perubahan diterapkan. Untuk informasi selengkapnya, lihat the section called “Menerapkan ulang REST API ke panggung”.

Otoritas sertifikat yang didukung API Gateway untuk integrasi proxy HTTP dan HTTP

Daftar berikut menunjukkan otoritas sertifikat yang didukung oleh API Gateway untuk HTTP, HTTP proxy, dan integrasi pribadi.

```
Alias name: accvraiz1
  SHA1: 93:05:7A:88:15:C6:4F:CE:88:2F:FA:91:16:52:28:78:BC:53:64:17
  SHA256:
 9A:6E:C0:12:E1:A7:DA:9D:BE:34:19:4D:47:8A:D7:C0:DB:18:22:FB:07:1D:F1:29:81:49:6E:D1:04:38:41:1
Alias name: acraizfnmtrcm
  SHA1: EC:50:35:07:B2:15:C4:95:62:19:E2:A8:9A:5B:42:99:2C:4C:2C:20
  SHA256:
 EB:C5:57:0C:29:01:8C:4D:67:B1:AA:12:7B:AF:12:F7:03:B4:61:1E:BC:17:B7:DA:B5:57:38:94:17:9B:93:F
Alias name: actalis
  SHA1: F3:73:B3:87:06:5A:28:84:8A:F2:F3:4A:CE:19:2B:DD:C7:8E:9C:AC
  SHA256:
 55:92:60:84:EC:96:3A:64:B9:6E:2A:BE:01:CE:0B:A8:6A:64:FB:FE:BC:C7:AA:B5:AF:C1:55:B3:7F:D7:60:6
Alias name: actalisauthenticationrootca
  SHA1: F3:73:B3:87:06:5A:28:84:8A:F2:F3:4A:CE:19:2B:DD:C7:8E:9C:AC
  SHA256:
 55:92:60:84:EC:96:3A:64:B9:6E:2A:BE:01:CE:0B:A8:6A:64:FB:FE:BC:C7:AA:B5:AF:C1:55:B3:7F:D7:60:6
Alias name: addtrustclass1ca
  SHA1: CC:AB:0E:A0:4C:23:01:D6:69:7B:DD:37:9F:CD:12:EB:24:E3:94:9D
  SHA256:
 8C:72:09:27:9A:C0:4E:27:5E:16:D0:7F:D3:B7:75:E8:01:54:B5:96:80:46:E3:1F:52:DD:25:76:63:24:E9:A
Alias name: addtrustexternalca
  SHA1: 02:FA:F3:E2:91:43:54:68:60:78:57:69:4D:F5:E4:5B:68:85:18:68
  SHA256:
 68:7F:A4:51:38:22:78:FF:F0:C8:B1:1F:8D:43:D5:76:67:1C:6E:B2:BC:EA:B4:13:FB:83:D9:65:D0:6D:2F:F
Alias name: addtrustqualifiedca
```

```
  SHA1: 4D:23:78:EC:91:95:39:B5:00:7F:75:8F:03:3B:21:1E:C5:4D:8B:CF
  SHA256:
 80:95:21:08:05:DB:4B:BC:35:5E:44:28:D8:FD:6E:C2:CD:E3:AB:5F:B9:7A:99:42:98:8E:B8:F4:DC:D0:60:1
Alias name: affirmtrustcommercial
  SHA1: F9:B5:B6:32:45:5F:9C:BE:EC:57:5F:80:DC:E9:6E:2C:C7:B2:78:B7
  SHA256:
 03:76:AB:1D:54:C5:F9:80:3C:E4:B2:E2:01:A0:EE:7E:EF:7B:57:B6:36:E8:A9:3C:9B:8D:48:60:C9:6F:5F:A
Alias name: affirmtrustcommercialca
  SHA1: F9:B5:B6:32:45:5F:9C:BE:EC:57:5F:80:DC:E9:6E:2C:C7:B2:78:B7
  SHA256:
 03:76:AB:1D:54:C5:F9:80:3C:E4:B2:E2:01:A0:EE:7E:EF:7B:57:B6:36:E8:A9:3C:9B:8D:48:60:C9:6F:5F:A
Alias name: affirmtrustnetworking
  SHA1: 29:36:21:02:8B:20:ED:02:F5:66:C5:32:D1:D6:ED:90:9F:45:00:2F
  SHA256:
 0A:81:EC:5A:92:97:77:F1:45:90:4A:F3:8D:5D:50:9F:66:B5:E2:C5:8F:CD:B5:31:05:8B:0E:17:F3:F0:B4:1
Alias name: affirmtrustnetworkingca
  SHA1: 29:36:21:02:8B:20:ED:02:F5:66:C5:32:D1:D6:ED:90:9F:45:00:2F
  SHA256:
 0A:81:EC:5A:92:97:77:F1:45:90:4A:F3:8D:5D:50:9F:66:B5:E2:C5:8F:CD:B5:31:05:8B:0E:17:F3:F0:B4:1
Alias name: affirmtrustpremium
  SHA1: D8:A6:33:2C:E0:03:6F:B1:85:F6:63:4F:7D:6A:06:65:26:32:28:27
  SHA256:
 70:A7:3F:7F:37:6B:60:07:42:48:90:45:34:B1:14:82:D5:BF:0E:69:8E:CC:49:8D:F5:25:77:EB:F2:E9:3B:9
Alias name: affirmtrustpremiumca
  SHA1: D8:A6:33:2C:E0:03:6F:B1:85:F6:63:4F:7D:6A:06:65:26:32:28:27
  SHA256:
 70:A7:3F:7F:37:6B:60:07:42:48:90:45:34:B1:14:82:D5:BF:0E:69:8E:CC:49:8D:F5:25:77:EB:F2:E9:3B:9
Alias name: affirmtrustpremiumecc
  SHA1: B8:23:6B:00:2F:1D:16:86:53:01:55:6C:11:A4:37:CA:EB:FF:C3:BB
  SHA256:
 BD:71:FD:F6:DA:97:E4:CF:62:D1:64:7A:DD:25:81:B0:7D:79:AD:F8:39:7E:B4:EC:BA:9C:5E:84:88:82:14:2
Alias name: affirmtrustpremiumeccca
  SHA1: B8:23:6B:00:2F:1D:16:86:53:01:55:6C:11:A4:37:CA:EB:FF:C3:BB
  SHA256:
 BD:71:FD:F6:DA:97:E4:CF:62:D1:64:7A:DD:25:81:B0:7D:79:AD:F8:39:7E:B4:EC:BA:9C:5E:84:88:82:14:2
Alias name: amazon-ca-g4-acm1
  SHA1: F2:0D:28:B6:29:C2:2C:5E:84:05:E6:02:4D:97:FE:8F:A0:84:93:A0
  SHA256:
 B0:11:A4:F7:29:6C:74:D8:2B:F5:62:DF:87:D7:28:C7:1F:B5:8C:F4:E6:73:F2:78:FC:DA:F3:FF:83:A6:8C:8
Alias name: amazon-ca-g4-acm2
  SHA1: A7:E6:45:32:1F:7A:B7:AD:C0:70:EA:73:5F:AB:ED:C3:DA:B4:D0:C8
  SHA256:
 D7:A8:7C:69:95:D0:E2:04:2A:32:70:A7:E2:87:FE:A7:E8:F4:C1:70:62:F7:90:C3:EB:BB:53:F2:AC:39:26:B
Alias name: amazon-ca-g4-acm3
```

```
  SHA1: 7A:DB:56:57:5F:D6:EE:67:85:0A:64:BB:1C:E9:E4:B0:9A:DB:9D:07
  SHA256:
 6B:EB:9D:20:2E:C2:00:70:BD:D2:5E:D3:C0:C8:33:2C:B4:78:07:C5:82:94:4E:7E:23:28:22:71:A4:8E:0E:C
Alias name: amazon-ca-g4-legacy
  SHA1: EA:E7:DE:F9:0A:BE:9F:0B:68:CE:B7:24:0D:80:74:03:BF:6E:B1:6E
  SHA256:
 CD:72:C4:7F:B4:AD:28:A4:67:2B:E1:86:47:D4:40:E9:3B:16:2D:95:DB:3C:2F:94:BB:81:D9:09:F7:91:24:5
Alias name: amazon-root-ca-ecc-384-1
  SHA1: F9:5E:4A:AB:9C:2D:57:61:63:3D:B2:57:B4:0F:24:9E:7B:E2:23:7D
  SHA256:
 C6:BD:E5:66:C2:72:2A:0E:96:E9:C1:2C:BF:38:92:D9:55:4D:29:03:57:30:72:40:7F:4E:70:17:3B:3C:9B:6
Alias name: amazon-root-ca-rsa-2k-1
  SHA1: 8A:9A:AC:27:FC:86:D4:50:23:AD:D5:63:F9:1E:AE:2C:AF:63:08:6C
  SHA256:
 0F:8F:33:83:FB:70:02:89:49:24:E1:AA:B0:D7:FB:5A:BF:98:DF:75:8E:0F:FE:61:86:92:BC:F0:75:35:CC:8
Alias name: amazon-root-ca-rsa-4k-1
  SHA1: EC:BD:09:61:F5:7A:B6:A8:76:BB:20:8F:14:05:ED:7E:70:ED:39:45
  SHA256:
 36:AE:AD:C2:6A:60:07:90:6B:83:A3:73:2D:D1:2B:D4:00:5E:C7:F2:76:11:99:A9:D4:DA:63:2F:59:B2:8B:C
Alias name: amazon1
  SHA1: 8D:A7:F9:65:EC:5E:FC:37:91:0F:1C:6E:59:FD:C1:CC:6A:6E:DE:16
  SHA256:
 8E:CD:E6:88:4F:3D:87:B1:12:5B:A3:1A:C3:FC:B1:3D:70:16:DE:7F:57:CC:90:4F:E1:CB:97:C6:AE:98:19:6
Alias name: amazon2
  SHA1: 5A:8C:EF:45:D7:A6:98:59:76:7A:8C:8B:44:96:B5:78:CF:47:4B:1A
  SHA256:
 1B:A5:B2:AA:8C:65:40:1A:82:96:01:18:F8:0B:EC:4F:62:30:4D:83:CE:C4:71:3A:19:C3:9C:01:1E:A4:6D:B
Alias name: amazon3
  SHA1: 0D:44:DD:8C:3C:8C:1A:1A:58:75:64:81:E9:0F:2E:2A:FF:B3:D2:6E
  SHA256:
 18:CE:6C:FE:7B:F1:4E:60:B2:E3:47:B8:DF:E8:68:CB:31:D0:2E:BB:3A:DA:27:15:69:F5:03:43:B4:6D:B3:A
Alias name: amazon4
  SHA1: F6:10:84:07:D6:F8:BB:67:98:0C:C2:E2:44:C2:EB:AE:1C:EF:63:BE
  SHA256:
 E3:5D:28:41:9E:D0:20:25:CF:A6:90:38:CD:62:39:62:45:8D:A5:C6:95:FB:DE:A3:C2:2B:0B:FB:25:89:70:9
Alias name: amazonrootca1
  SHA1: 8D:A7:F9:65:EC:5E:FC:37:91:0F:1C:6E:59:FD:C1:CC:6A:6E:DE:16
  SHA256:
 8E:CD:E6:88:4F:3D:87:B1:12:5B:A3:1A:C3:FC:B1:3D:70:16:DE:7F:57:CC:90:4F:E1:CB:97:C6:AE:98:19:6
Alias name: amazonrootca2
  SHA1: 5A:8C:EF:45:D7:A6:98:59:76:7A:8C:8B:44:96:B5:78:CF:47:4B:1A
  SHA256:
 1B:A5:B2:AA:8C:65:40:1A:82:96:01:18:F8:0B:EC:4F:62:30:4D:83:CE:C4:71:3A:19:C3:9C:01:1E:A4:6D:B
Alias name: amazonrootca3
```

```
  SHA1: 0D:44:DD:8C:3C:8C:1A:1A:58:75:64:81:E9:0F:2E:2A:FF:B3:D2:6E
  SHA256:
 18:CE:6C:FE:7B:F1:4E:60:B2:E3:47:B8:DF:E8:68:CB:31:D0:2E:BB:3A:DA:27:15:69:F5:03:43:B4:6D:B3:A
Alias name: amazonrootca4
  SHA1: F6:10:84:07:D6:F8:BB:67:98:0C:C2:E2:44:C2:EB:AE:1C:EF:63:BE
  SHA256:
 E3:5D:28:41:9E:D0:20:25:CF:A6:90:38:CD:62:39:62:45:8D:A5:C6:95:FB:DE:A3:C2:2B:0B:FB:25:89:70:9
Alias name: amzninternalinfoseccag3
  SHA1: B9:B1:CA:38:F7:BF:9C:D2:D4:95:E7:B6:5E:75:32:9B:A8:78:2E:F6
  SHA256:
 81:03:0B:C7:E2:54:DA:7B:F8:B7:45:DB:DD:41:15:89:B5:A3:81:86:FB:4B:29:77:1F:84:0A:18:D9:67:6D:6
Alias name: amzninternalrootca
  SHA1: A7:B7:F6:15:8A:FF:1E:C8:85:13:38:BC:93:EB:A2:AB:A4:09:EF:06
  SHA256:
 0E:DE:63:C1:DC:7A:8E:11:F1:AB:BC:05:4F:59:EE:49:9D:62:9A:2F:DE:9C:A7:16:32:A2:64:29:3E:8B:66:A
Alias name: aolrootca1
  SHA1: 39:21:C1:15:C1:5D:0E:CA:5C:CB:5B:C4:F0:7D:21:D8:05:0B:56:6A
  SHA256:
 77:40:73:12:C6:3A:15:3D:5B:C0:0B:4E:51:75:9C:DF:DA:C2:37:DC:2A:33:B6:79:46:E9:8E:9B:FA:68:0A:E
Alias name: aolrootca2
  SHA1: 85:B5:FF:67:9B:0C:79:96:1F:C8:6E:44:22:00:46:13:DB:17:92:84
  SHA256:
 7D:3B:46:5A:60:14:E5:26:C0:AF:FC:EE:21:27:D2:31:17:27:AD:81:1C:26:84:2D:00:6A:F3:73:06:CC:80:B
Alias name: atostrustedroot2011
  SHA1: 2B:B1:F5:3E:55:0C:1D:C5:F1:D4:E6:B7:6A:46:4B:55:06:02:AC:21
  SHA256:
 F3:56:BE:A2:44:B7:A9:1E:B3:5D:53:CA:9A:D7:86:4A:CE:01:8E:2D:35:D5:F8:F9:6D:DF:68:A6:F4:1A:A4:7
Alias name: autoridaddecertificacionfirmaprofesionalcifa62634068
  SHA1: AE:C5:FB:3F:C8:E1:BF:C4:E5:4F:03:07:5A:9A:E8:00:B7:F7:B6:FA
  SHA256:
 04:04:80:28:BF:1F:28:64:D4:8F:9A:D4:D8:32:94:36:6A:82:88:56:55:3F:3B:14:30:3F:90:14:7F:5D:40:E
Alias name: baltimorecodesigningca
  SHA1: 30:46:D8:C8:88:FF:69:30:C3:4A:FC:CD:49:27:08:7C:60:56:7B:0D
  SHA256:
 A9:15:45:DB:D2:E1:9C:4C:CD:F9:09:AA:71:90:0D:18:C7:35:1C:89:B3:15:F0:F1:3D:05:C1:3A:8F:FB:46:8
Alias name: baltimorecybertrustca
  SHA1: D4:DE:20:D0:5E:66:FC:53:FE:1A:50:88:2C:78:DB:28:52:CA:E4:74
  SHA256:
 16:AF:57:A9:F6:76:B0:AB:12:60:95:AA:5E:BA:DE:F2:2A:B3:11:19:D6:44:AC:95:CD:4B:93:DB:F3:F2:6A:E
Alias name: baltimorecybertrustroot
  SHA1: D4:DE:20:D0:5E:66:FC:53:FE:1A:50:88:2C:78:DB:28:52:CA:E4:74
  SHA256:
 16:AF:57:A9:F6:76:B0:AB:12:60:95:AA:5E:BA:DE:F2:2A:B3:11:19:D6:44:AC:95:CD:4B:93:DB:F3:F2:6A:E
Alias name: buypassclass2ca
```

```
  SHA1: 49:0A:75:74:DE:87:0A:47:FE:58:EE:F6:C7:6B:EB:C6:0B:12:40:99
  SHA256:
 9A:11:40:25:19:7C:5B:B9:5D:94:E6:3D:55:CD:43:79:08:47:B6:46:B2:3C:DF:11:AD:A4:A0:0E:FF:15:FB:4
Alias name: buypassclass2rootca
  SHA1: 49:0A:75:74:DE:87:0A:47:FE:58:EE:F6:C7:6B:EB:C6:0B:12:40:99
  SHA256:
 9A:11:40:25:19:7C:5B:B9:5D:94:E6:3D:55:CD:43:79:08:47:B6:46:B2:3C:DF:11:AD:A4:A0:0E:FF:15:FB:4
Alias name: buypassclass3ca
  SHA1: DA:FA:F7:FA:66:84:EC:06:8F:14:50:BD:C7:C2:81:A5:BC:A9:64:57
  SHA256:
 ED:F7:EB:BC:A2:7A:2A:38:4D:38:7B:7D:40:10:C6:66:E2:ED:B4:84:3E:4C:29:B4:AE:1D:5B:93:32:E6:B2:4
Alias name: buypassclass3rootca
  SHA1: DA:FA:F7:FA:66:84:EC:06:8F:14:50:BD:C7:C2:81:A5:BC:A9:64:57
  SHA256:
 ED:F7:EB:BC:A2:7A:2A:38:4D:38:7B:7D:40:10:C6:66:E2:ED:B4:84:3E:4C:29:B4:AE:1D:5B:93:32:E6:B2:4
Alias name: cadisigrootr2
  SHA1: B5:61:EB:EA:A4:DE:E4:25:4B:69:1A:98:A5:57:47:C2:34:C7:D9:71
  SHA256:
 E2:3D:4A:03:6D:7B:70:E9:F5:95:B1:42:20:79:D2:B9:1E:DF:BB:1F:B6:51:A0:63:3E:AA:8A:9D:C5:F8:07:0
Alias name: camerfirmachambersca
  SHA1: 78:6A:74:AC:76:AB:14:7F:9C:6A:30:50:BA:9E:A8:7E:FE:9A:CE:3C
  SHA256:
 06:3E:4A:FA:C4:91:DF:D3:32:F3:08:9B:85:42:E9:46:17:D8:93:D7:FE:94:4E:10:A7:93:7E:E2:9D:96:93:C
Alias name: camerfirmachamberscommerceca
  SHA1: 6E:3A:55:A4:19:0C:19:5C:93:84:3C:C0:DB:72:2E:31:30:61:F0:B1
  SHA256:
 0C:25:8A:12:A5:67:4A:EF:25:F2:8B:A7:DC:FA:EC:EE:A3:48:E5:41:E6:F5:CC:4E:E6:3B:71:B3:61:60:6A:C
Alias name: camerfirmachambersignca
  SHA1: 4A:BD:EE:EC:95:0D:35:9C:89:AE:C7:52:A1:2C:5B:29:F6:D6:AA:0C
  SHA256:
 13:63:35:43:93:34:A7:69:80:16:A0:D3:24:DE:72:28:4E:07:9D:7B:52:20:BB:8F:BD:74:78:16:EE:BE:BA:C
Alias name: certigna
  SHA1: B1:2E:13:63:45:86:A4:6F:1A:B2:60:68:37:58:2D:C4:AC:FD:94:97
  SHA256:
 E3:B6:A2:DB:2E:D7:CE:48:84:2F:7A:C5:32:41:C7:B7:1D:54:14:4B:FB:40:C1:1F:3F:1D:0B:42:F5:EE:A1:2
Alias name: certignarootca
  SHA1: 2D:0D:52:14:FF:9E:AD:99:24:01:74:20:47:6E:6C:85:27:27:F5:43
  SHA256:
 D4:8D:3D:23:EE:DB:50:A4:59:E5:51:97:60:1C:27:77:4B:9D:7B:18:C9:4D:5A:05:95:11:A1:02:50:B9:31:6
Alias name: certplusclass2primaryca
  SHA1: 74:20:74:41:72:9C:DD:92:EC:79:31:D8:23:10:8D:C2:81:92:E2:BB
  SHA256:
 0F:99:3C:8A:EF:97:BA:AF:56:87:14:0E:D5:9A:D1:82:1B:B4:AF:AC:F0:AA:9A:58:B5:D5:7A:33:8A:3A:FB:C
Alias name: certplusclass3pprimaryca
```

```
  SHA1: 21:6B:2A:29:E6:2A:00:CE:82:01:46:D8:24:41:41:B9:25:11:B2:79
  SHA256:
 CC:C8:94:89:37:1B:AD:11:1C:90:61:9B:EA:24:0A:2E:6D:AD:D9:9F:9F:6E:1D:4D:41:E5:8E:D6:DE:3D:02:8
Alias name: certsignrootca
  SHA1: FA:B7:EE:36:97:26:62:FB:2D:B0:2A:F6:BF:03:FD:E8:7C:4B:2F:9B
  SHA256:
 EA:A9:62:C4:FA:4A:6B:AF:EB:E4:15:19:6D:35:1C:CD:88:8D:4F:53:F3:FA:8A:E6:D7:C4:66:A9:4E:60:42:B
Alias name: certsignrootcag2
  SHA1: 26:F9:93:B4:ED:3D:28:27:B0:B9:4B:A7:E9:15:1D:A3:8D:92:E5:32
  SHA256:
 65:7C:FE:2F:A7:3F:AA:38:46:25:71:F3:32:A2:36:3A:46:FC:E7:02:09:51:71:07:02:CD:FB:B6:EE:DA:33:0
Alias name: certum2
  SHA1: D3:DD:48:3E:2B:BF:4C:05:E8:AF:10:F5:FA:76:26:CF:D3:DC:30:92
  SHA256:
 B6:76:F2:ED:DA:E8:77:5C:D3:6C:B0:F6:3C:D1:D4:60:39:61:F4:9E:62:65:BA:01:3A:2F:03:07:B6:D0:B8:0
Alias name: certumca
  SHA1: 62:52:DC:40:F7:11:43:A2:2F:DE:9E:F7:34:8E:06:42:51:B1:81:18
  SHA256:
 D8:E0:FE:BC:1D:B2:E3:8D:00:94:0F:37:D2:7D:41:34:4D:99:3E:73:4B:99:D5:65:6D:97:78:D4:D8:14:36:2
Alias name: certumtrustednetworkca
  SHA1: 07:E0:32:E0:20:B7:2C:3F:19:2F:06:28:A2:59:3A:19:A7:0F:06:9E
  SHA256:
 5C:58:46:8D:55:F5:8E:49:7E:74:39:82:D2:B5:00:10:B6:D1:65:37:4A:CF:83:A7:D4:A3:2D:B7:68:C4:40:8
Alias name: certumtrustednetworkca2
  SHA1: D3:DD:48:3E:2B:BF:4C:05:E8:AF:10:F5:FA:76:26:CF:D3:DC:30:92
  SHA256:
 B6:76:F2:ED:DA:E8:77:5C:D3:6C:B0:F6:3C:D1:D4:60:39:61:F4:9E:62:65:BA:01:3A:2F:03:07:B6:D0:B8:0
Alias name: cfcaevroot
  SHA1: E2:B8:29:4B:55:84:AB:6B:58:C2:90:46:6C:AC:3F:B8:39:8F:84:83
  SHA256:
 5C:C3:D7:8E:4E:1D:5E:45:54:7A:04:E6:87:3E:64:F9:0C:F9:53:6D:1C:CC:2E:F8:00:F3:55:C4:C5:FD:70:F
Alias name: chambersofcommerceroot2008
  SHA1: 78:6A:74:AC:76:AB:14:7F:9C:6A:30:50:BA:9E:A8:7E:FE:9A:CE:3C
  SHA256:
 06:3E:4A:FA:C4:91:DF:D3:32:F3:08:9B:85:42:E9:46:17:D8:93:D7:FE:94:4E:10:A7:93:7E:E2:9D:96:93:C
Alias name: chunghwaepkirootca
  SHA1: 67:65:0D:F1:7E:8E:7E:5B:82:40:A4:F4:56:4B:CF:E2:3D:69:C6:F0
  SHA256:
 C0:A6:F4:DC:63:A2:4B:FD:CF:54:EF:2A:6A:08:2A:0A:72:DE:35:80:3E:2F:F5:FF:52:7A:E5:D8:72:06:DF:D
Alias name: cia-crt-g3-01-ca
  SHA1: 2B:EE:2C:BA:A3:1D:B5:FE:60:40:41:95:08:ED:46:82:39:4D:ED:E2
  SHA256:
 20:48:AD:4C:EC:90:7F:FA:4A:15:D4:CE:45:E3:C8:E4:2C:EA:78:33:DC:C7:D3:40:48:FC:60:47:27:42:99:E
Alias name: cia-crt-g3-02-ca
```

```
  SHA1: 96:4A:BB:A7:BD:DA:FC:97:34:C0:0A:2D:F0:05:98:F7:E6:C6:6F:09
  SHA256:
 93:F1:72:FB:BA:43:31:5C:06:EE:0F:9F:04:89:B8:F6:88:BC:75:15:3C:BE:B4:80:AC:A7:14:3A:F6:FC:4A:C
Alias name: comodo-ca
  SHA1: AF:E5:D2:44:A8:D1:19:42:30:FF:47:9F:E2:F8:97:BB:CD:7A:8C:B4
  SHA256:
 52:F0:E1:C4:E5:8E:C6:29:29:1B:60:31:7F:07:46:71:B8:5D:7E:A8:0D:5B:07:27:34:63:53:4B:32:B4:02:3
Alias name: comodoaaaca
  SHA1: D1:EB:23:A4:6D:17:D6:8F:D9:25:64:C2:F1:F1:60:17:64:D8:E3:49
  SHA256:
 D7:A7:A0:FB:5D:7E:27:31:D7:71:E9:48:4E:BC:DE:F7:1D:5F:0C:3E:0A:29:48:78:2B:C8:3E:E0:EA:69:9E:F
Alias name: comodoaaaservicesroot
  SHA1: D1:EB:23:A4:6D:17:D6:8F:D9:25:64:C2:F1:F1:60:17:64:D8:E3:49
  SHA256:
 D7:A7:A0:FB:5D:7E:27:31:D7:71:E9:48:4E:BC:DE:F7:1D:5F:0C:3E:0A:29:48:78:2B:C8:3E:E0:EA:69:9E:F
Alias name: comodocertificationauthority
  SHA1: 66:31:BF:9E:F7:4F:9E:B6:C9:D5:A6:0C:BA:6A:BE:D1:F7:BD:EF:7B
  SHA256:
 0C:2C:D6:3D:F7:80:6F:A3:99:ED:E8:09:11:6B:57:5B:F8:79:89:F0:65:18:F9:80:8C:86:05:03:17:8B:AF:6
Alias name: comodoecccertificationauthority
  SHA1: 9F:74:4E:9F:2B:4D:BA:EC:0F:31:2C:50:B6:56:3B:8E:2D:93:C3:11
  SHA256:
 17:93:92:7A:06:14:54:97:89:AD:CE:2F:8F:34:F7:F0:B6:6D:0F:3A:E3:A3:B8:4D:21:EC:15:DB:BA:4F:AD:C
Alias name: comodorsacertificationauthority
  SHA1: AF:E5:D2:44:A8:D1:19:42:30:FF:47:9F:E2:F8:97:BB:CD:7A:8C:B4
  SHA256:
 52:F0:E1:C4:E5:8E:C6:29:29:1B:60:31:7F:07:46:71:B8:5D:7E:A8:0D:5B:07:27:34:63:53:4B:32:B4:02:3
Alias name: cybertrustglobalroot
  SHA1: 5F:43:E5:B1:BF:F8:78:8C:AC:1C:C7:CA:4A:9A:C6:22:2B:CC:34:C6
  SHA256:
 96:0A:DF:00:63:E9:63:56:75:0C:29:65:DD:0A:08:67:DA:0B:9C:BD:6E:77:71:4A:EA:FB:23:49:AB:39:3D:A
Alias name: deprecateditsecca
  SHA1: 12:12:0B:03:0E:15:14:54:F4:DD:B3:F5:DE:13:6E:83:5A:29:72:9D
  SHA256:
 9A:59:DA:86:24:1A:FD:BA:A3:39:FA:9C:FD:21:6A:0B:06:69:4D:E3:7E:37:52:6B:BE:63:C8:BC:83:74:2E:C
Alias name: deutschetelekomrootca2
  SHA1: 85:A4:08:C0:9C:19:3E:5D:51:58:7D:CD:D6:13:30:FD:8C:DE:37:BF
  SHA256:
 B6:19:1A:50:D0:C3:97:7F:7D:A9:9B:CD:AA:C8:6A:22:7D:AE:B9:67:9E:C7:0B:A3:B0:C9:D9:22:71:C1:70:D
Alias name: digicertassuredidrootca
  SHA1: 05:63:B8:63:0D:62:D7:5A:BB:C8:AB:1E:4B:DF:B5:A8:99:B2:4D:43
  SHA256:
 3E:90:99:B5:01:5E:8F:48:6C:00:BC:EA:9D:11:1E:E7:21:FA:BA:35:5A:89:BC:F1:DF:69:56:1E:3D:C6:32:5
Alias name: digicertassuredidrootg2
```

```
  SHA1: A1:4B:48:D9:43:EE:0A:0E:40:90:4F:3C:E0:A4:C0:91:93:51:5D:3F
  SHA256:
 7D:05:EB:B6:82:33:9F:8C:94:51:EE:09:4E:EB:FE:FA:79:53:A1:14:ED:B2:F4:49:49:45:2F:AB:7D:2F:C1:8
Alias name: digicertassuredidrootg3
  SHA1: F5:17:A2:4F:9A:48:C6:C9:F8:A2:00:26:9F:DC:0F:48:2C:AB:30:89
  SHA256:
 7E:37:CB:8B:4C:47:09:0C:AB:36:55:1B:A6:F4:5D:B8:40:68:0F:BA:16:6A:95:2D:B1:00:71:7F:43:05:3F:C
Alias name: digicertglobalrootca
  SHA1: A8:98:5D:3A:65:E5:E5:C4:B2:D7:D6:6D:40:C6:DD:2F:B1:9C:54:36
  SHA256:
 43:48:A0:E9:44:4C:78:CB:26:5E:05:8D:5E:89:44:B4:D8:4F:96:62:BD:26:DB:25:7F:89:34:A4:43:C7:01:6
Alias name: digicertglobalrootg2
  SHA1: DF:3C:24:F9:BF:D6:66:76:1B:26:80:73:FE:06:D1:CC:8D:4F:82:A4
  SHA256:
 CB:3C:CB:B7:60:31:E5:E0:13:8F:8D:D3:9A:23:F9:DE:47:FF:C3:5E:43:C1:14:4C:EA:27:D4:6A:5A:B1:CB:5
Alias name: digicertglobalrootg3
  SHA1: 7E:04:DE:89:6A:3E:66:6D:00:E6:87:D3:3F:FA:D9:3B:E8:3D:34:9E
  SHA256:
 31:AD:66:48:F8:10:41:38:C7:38:F3:9E:A4:32:01:33:39:3E:3A:18:CC:02:29:6E:F9:7C:2A:C9:EF:67:31:D
Alias name: digicerthighassuranceevrootca
  SHA1: 5F:B7:EE:06:33:E2:59:DB:AD:0C:4C:9A:E6:D3:8F:1A:61:C7:DC:25
  SHA256:
 74:31:E5:F4:C3:C1:CE:46:90:77:4F:0B:61:E0:54:40:88:3B:A9:A0:1E:D0:0B:A6:AB:D7:80:6E:D3:B1:18:C
Alias name: digicerttrustedrootg4
  SHA1: DD:FB:16:CD:49:31:C9:73:A2:03:7D:3F:C8:3A:4D:7D:77:5D:05:E4
  SHA256:
 55:2F:7B:DC:F1:A7:AF:9E:6C:E6:72:01:7F:4F:12:AB:F7:72:40:C7:8E:76:1A:C2:03:D1:D9:D2:0A:C8:99:8
Alias name: dstrootcax3
  SHA1: DA:C9:02:4F:54:D8:F6:DF:94:93:5F:B1:73:26:38:CA:6A:D7:7C:13
  SHA256:
 06:87:26:03:31:A7:24:03:D9:09:F1:05:E6:9B:CF:0D:32:E1:BD:24:93:FF:C6:D9:20:6D:11:BC:D6:77:07:3
Alias name: dtrustrootclass3ca22009
  SHA1: 58:E8:AB:B0:36:15:33:FB:80:F7:9B:1B:6D:29:D3:FF:8D:5F:00:F0
  SHA256:
 49:E7:A4:42:AC:F0:EA:62:87:05:00:54:B5:25:64:B6:50:E4:F4:9E:42:E3:48:D6:AA:38:E0:39:E9:57:B1:C
Alias name: dtrustrootclass3ca2ev2009
  SHA1: 96:C9:1B:0B:95:B4:10:98:42:FA:D0:D8:22:79:FE:60:FA:B9:16:83
  SHA256:
 EE:C5:49:6B:98:8C:E9:86:25:B9:34:09:2E:EC:29:08:BE:D0:B0:F3:16:C2:D4:73:0C:84:EA:F1:F3:D3:48:8
Alias name: ecacc
  SHA1: 28:90:3A:63:5B:52:80:FA:E6:77:4C:0B:6D:A7:D6:BA:A6:4A:F2:E8
  SHA256:
 88:49:7F:01:60:2F:31:54:24:6A:E2:8C:4D:5A:EF:10:F1:D8:7E:BB:76:62:6F:4A:E0:B7:F9:5B:A7:96:87:9
Alias name: emsigneccrootcac3
```

```
  SHA1: B6:AF:43:C2:9B:81:53:7D:F6:EF:6B:C3:1F:1F:60:15:0C:EE:48:66
  SHA256:
 BC:4D:80:9B:15:18:9D:78:DB:3E:1D:8C:F4:F9:72:6A:79:5D:A1:64:3C:A5:F1:35:8E:1D:DB:0E:DC:0D:7E:B
Alias name: emsigneccrootcag3
  SHA1: 30:43:FA:4F:F2:57:DC:A0:C3:80:EE:2E:58:EA:78:B2:3F:E6:BB:C1
  SHA256:
 86:A1:EC:BA:08:9C:4A:8D:3B:BE:27:34:C6:12:BA:34:1D:81:3E:04:3C:F9:E8:A8:62:CD:5C:57:A3:6B:BE:6
Alias name: emsignrootcac1
  SHA1: E7:2E:F1:DF:FC:B2:09:28:CF:5D:D4:D5:67:37:B1:51:CB:86:4F:01
  SHA256:
 12:56:09:AA:30:1D:A0:A2:49:B9:7A:82:39:CB:6A:34:21:6F:44:DC:AC:9F:39:54:B1:42:92:F2:E8:C8:60:8
Alias name: emsignrootcag1
  SHA1: 8A:C7:AD:8F:73:AC:4E:C1:B5:75:4D:A5:40:F4:FC:CF:7C:B5:8E:8C
  SHA256:
 40:F6:AF:03:46:A9:9A:A1:CD:1D:55:5A:4E:9C:CE:62:C7:F9:63:46:03:EE:40:66:15:83:3D:C8:C8:D0:03:6
Alias name: entrust2048ca
  SHA1: 50:30:06:09:1D:97:D4:F5:AE:39:F7:CB:E7:92:7D:7D:65:2D:34:31
  SHA256:
 6D:C4:71:72:E0:1C:BC:B0:BF:62:58:0D:89:5F:E2:B8:AC:9A:D4:F8:73:80:1E:0C:10:B9:C8:37:D2:1E:B1:7
Alias name: entrustevca
  SHA1: B3:1E:B1:B7:40:E3:6C:84:02:DA:DC:37:D4:4D:F5:D4:67:49:52:F9
  SHA256:
 73:C1:76:43:4F:1B:C6:D5:AD:F4:5B:0E:76:E7:27:28:7C:8D:E5:76:16:C1:E6:E6:14:1A:2B:2C:BC:7D:8E:4
Alias name: entrustnetpremium2048secureserverca
  SHA1: 50:30:06:09:1D:97:D4:F5:AE:39:F7:CB:E7:92:7D:7D:65:2D:34:31
  SHA256:
 6D:C4:71:72:E0:1C:BC:B0:BF:62:58:0D:89:5F:E2:B8:AC:9A:D4:F8:73:80:1E:0C:10:B9:C8:37:D2:1E:B1:7
Alias name: entrustrootcag2
  SHA1: 8C:F4:27:FD:79:0C:3A:D1:66:06:8D:E8:1E:57:EF:BB:93:22:72:D4
  SHA256:
 43:DF:57:74:B0:3E:7F:EF:5F:E4:0D:93:1A:7B:ED:F1:BB:2E:6B:42:73:8C:4E:6D:38:41:10:3D:3A:A7:F3:3
Alias name: entrustrootcertificationauthority
  SHA1: B3:1E:B1:B7:40:E3:6C:84:02:DA:DC:37:D4:4D:F5:D4:67:49:52:F9
  SHA256:
 73:C1:76:43:4F:1B:C6:D5:AD:F4:5B:0E:76:E7:27:28:7C:8D:E5:76:16:C1:E6:E6:14:1A:2B:2C:BC:7D:8E:4
Alias name: entrustrootcertificationauthorityec1
  SHA1: 20:D8:06:40:DF:9B:25:F5:12:25:3A:11:EA:F7:59:8A:EB:14:B5:47
  SHA256:
 02:ED:0E:B2:8C:14:DA:45:16:5C:56:67:91:70:0D:64:51:D7:FB:56:F0:B2:AB:1D:3B:8E:B0:70:E5:6E:DF:F
Alias name: entrustrootcertificationauthorityg2
  SHA1: 8C:F4:27:FD:79:0C:3A:D1:66:06:8D:E8:1E:57:EF:BB:93:22:72:D4
  SHA256:
 43:DF:57:74:B0:3E:7F:EF:5F:E4:0D:93:1A:7B:ED:F1:BB:2E:6B:42:73:8C:4E:6D:38:41:10:3D:3A:A7:F3:3
Alias name: entrustrootcertificationauthorityg4
```

```
  SHA1: 14:88:4E:86:26:37:B0:26:AF:59:62:5C:40:77:EC:35:29:BA:96:01
  SHA256:
 DB:35:17:D1:F6:73:2A:2D:5A:B9:7C:53:3E:C7:07:79:EE:32:70:A6:2F:B4:AC:42:38:37:24:60:E6:F0:1E:8
Alias name: epkirootcertificationauthority
  SHA1: 67:65:0D:F1:7E:8E:7E:5B:82:40:A4:F4:56:4B:CF:E2:3D:69:C6:F0
  SHA256:
 C0:A6:F4:DC:63:A2:4B:FD:CF:54:EF:2A:6A:08:2A:0A:72:DE:35:80:3E:2F:F5:FF:52:7A:E5:D8:72:06:DF:D
Alias name: equifaxsecureebusinessca1
  SHA1: AE:E6:3D:70:E3:76:FB:C7:3A:EB:B0:A1:C1:D4:C4:7A:A7:40:B3:F4
  SHA256:
 2E:3A:2B:B5:11:25:05:83:6C:A8:96:8B:E2:CB:37:27:CE:9B:56:84:5C:6E:E9:8E:91:85:10:4A:FB:9A:F5:9
Alias name: equifaxsecureglobalebusinessca1
  SHA1: 3A:74:CB:7A:47:DB:70:DE:89:1F:24:35:98:64:B8:2D:82:BD:1A:36
  SHA256:
 86:AB:5A:65:71:D3:32:9A:BC:D2:E4:E6:37:66:8B:A8:9C:73:1E:C2:93:B6:CB:A6:0F:71:63:40:A0:91:CE:A
Alias name: eszignorootca2017
  SHA1: 89:D4:83:03:4F:9E:9A:48:80:5F:72:37:D4:A9:A6:EF:CB:7C:1F:D1
  SHA256:
 BE:B0:0B:30:83:9B:9B:C3:2C:32:E4:44:79:05:95:06:41:F2:64:21:B1:5E:D0:89:19:8B:51:8A:E2:EA:1B:9
Alias name: etugracertificationauthority
  SHA1: 51:C6:E7:08:49:06:6E:F3:92:D4:5C:A0:0D:6D:A3:62:8F:C3:52:39
  SHA256:
 B0:BF:D5:2B:B0:D7:D9:BD:92:BF:5D:4D:C1:3D:A2:55:C0:2C:54:2F:37:83:65:EA:89:39:11:F5:5E:55:F2:3
Alias name: gd-class2-root.pem
  SHA1: 27:96:BA:E6:3F:18:01:E2:77:26:1B:A0:D7:77:70:02:8F:20:EE:E4
  SHA256:
 C3:84:6B:F2:4B:9E:93:CA:64:27:4C:0E:C6:7C:1E:CC:5E:02:4F:FC:AC:D2:D7:40:19:35:0E:81:FE:54:6A:E
Alias name: gd_bundle-g2.pem
  SHA1: 27:AC:93:69:FA:F2:52:07:BB:26:27:CE:FA:CC:BE:4E:F9:C3:19:B8
  SHA256:
 97:3A:41:27:6F:FD:01:E0:27:A2:AA:D4:9E:34:C3:78:46:D3:E9:76:FF:6A:62:0B:67:12:E3:38:32:04:1A:A
Alias name: gdcatrustauthr5root
  SHA1: 0F:36:38:5B:81:1A:25:C3:9B:31:4E:83:CA:E9:34:66:70:CC:74:B4
  SHA256:
 BF:FF:8F:D0:44:33:48:7D:6A:8A:A6:0C:1A:29:76:7A:9F:C2:BB:B0:5E:42:0F:71:3A:13:B9:92:89:1D:38:9
Alias name: gdroot-g2.pem
  SHA1: 47:BE:AB:C9:22:EA:E8:0E:78:78:34:62:A7:9F:45:C2:54:FD:E6:8B
  SHA256:
 45:14:0B:32:47:EB:9C:C8:C5:B4:F0:D7:B5:30:91:F7:32:92:08:9E:6E:5A:63:E2:74:9D:D3:AC:A9:19:8E:D
Alias name: geotrustglobalca
  SHA1: DE:28:F4:A4:FF:E5:B9:2F:A3:C5:03:D1:A3:49:A7:F9:96:2A:82:12
  SHA256:
 FF:85:6A:2D:25:1D:CD:88:D3:66:56:F4:50:12:67:98:CF:AB:AA:DE:40:79:9C:72:2D:E4:D2:B5:DB:36:A7:3
Alias name: geotrustprimaryca
```

```
  SHA1: 32:3C:11:8E:1B:F7:B8:B6:52:54:E2:E2:10:0D:D6:02:90:37:F0:96
  SHA256:
 37:D5:10:06:C5:12:EA:AB:62:64:21:F1:EC:8C:92:01:3F:C5:F8:2A:E9:8E:E5:33:EB:46:19:B8:DE:B4:D0:6
Alias name: geotrustprimarycag2
  SHA1: 8D:17:84:D5:37:F3:03:7D:EC:70:FE:57:8B:51:9A:99:E6:10:D7:B0
  SHA256:
 5E:DB:7A:C4:3B:82:A0:6A:87:61:E8:D7:BE:49:79:EB:F2:61:1F:7D:D7:9B:F9:1C:1C:6B:56:6A:21:9E:D7:6
Alias name: geotrustprimarycag3
  SHA1: 03:9E:ED:B8:0B:E7:A0:3C:69:53:89:3B:20:D2:D9:32:3A:4C:2A:FD
  SHA256:
 B4:78:B8:12:25:0D:F8:78:63:5C:2A:A7:EC:7D:15:5E:AA:62:5E:E8:29:16:E2:CD:29:43:61:88:6C:D1:FB:D
Alias name: geotrustprimarycertificationauthority
  SHA1: 32:3C:11:8E:1B:F7:B8:B6:52:54:E2:E2:10:0D:D6:02:90:37:F0:96
  SHA256:
 37:D5:10:06:C5:12:EA:AB:62:64:21:F1:EC:8C:92:01:3F:C5:F8:2A:E9:8E:E5:33:EB:46:19:B8:DE:B4:D0:6
Alias name: geotrustprimarycertificationauthorityg2
  SHA1: 8D:17:84:D5:37:F3:03:7D:EC:70:FE:57:8B:51:9A:99:E6:10:D7:B0
  SHA256:
 5E:DB:7A:C4:3B:82:A0:6A:87:61:E8:D7:BE:49:79:EB:F2:61:1F:7D:D7:9B:F9:1C:1C:6B:56:6A:21:9E:D7:6
Alias name: geotrustprimarycertificationauthorityg3
  SHA1: 03:9E:ED:B8:0B:E7:A0:3C:69:53:89:3B:20:D2:D9:32:3A:4C:2A:FD
  SHA256:
 B4:78:B8:12:25:0D:F8:78:63:5C:2A:A7:EC:7D:15:5E:AA:62:5E:E8:29:16:E2:CD:29:43:61:88:6C:D1:FB:D
Alias name: geotrustuniversalca
  SHA1: E6:21:F3:35:43:79:05:9A:4B:68:30:9D:8A:2F:74:22:15:87:EC:79
  SHA256:
 A0:45:9B:9F:63:B2:25:59:F5:FA:5D:4C:6D:B3:F9:F7:2F:F1:93:42:03:35:78:F0:73:BF:1D:1B:46:CB:B9:1
Alias name: geotrustuniversalca2
  SHA1: 37:9A:19:7B:41:85:45:35:0C:A6:03:69:F3:3C:2E:AF:47:4F:20:79
  SHA256:
 A0:23:4F:3B:C8:52:7C:A5:62:8E:EC:81:AD:5D:69:89:5D:A5:68:0D:C9:1D:1C:B8:47:7F:33:F8:78:B9:5B:0
Alias name: globalchambersignroot2008
  SHA1: 4A:BD:EE:EC:95:0D:35:9C:89:AE:C7:52:A1:2C:5B:29:F6:D6:AA:0C
  SHA256:
 13:63:35:43:93:34:A7:69:80:16:A0:D3:24:DE:72:28:4E:07:9D:7B:52:20:BB:8F:BD:74:78:16:EE:BE:BA:C
Alias name: globalsignca
  SHA1: B1:BC:96:8B:D4:F4:9D:62:2A:A8:9A:81:F2:15:01:52:A4:1D:82:9C
  SHA256:
 EB:D4:10:40:E4:BB:3E:C7:42:C9:E3:81:D3:1E:F2:A4:1A:48:B6:68:5C:96:E7:CE:F3:C1:DF:6C:D4:33:1C:9
Alias name: globalsigneccrootcar4
  SHA1: 69:69:56:2E:40:80:F4:24:A1:E7:19:9F:14:BA:F3:EE:58:AB:6A:BB
  SHA256:
 BE:C9:49:11:C2:95:56:76:DB:6C:0A:55:09:86:D7:6E:3B:A0:05:66:7C:44:2C:97:62:B4:FB:B7:73:DE:22:8
Alias name: globalsigneccrootcar5
```

```
  SHA1: 1F:24:C6:30:CD:A4:18:EF:20:69:FF:AD:4F:DD:5F:46:3A:1B:69:AA
  SHA256:
 17:9F:BC:14:8A:3D:D0:0F:D2:4E:A1:34:58:CC:43:BF:A7:F5:9C:81:82:D7:83:A5:13:F6:EB:EC:10:0C:89:2
Alias name: globalsignr2ca
  SHA1: 75:E0:AB:B6:13:85:12:27:1C:04:F8:5F:DD:DE:38:E4:B7:24:2E:FE
  SHA256:
 CA:42:DD:41:74:5F:D0:B8:1E:B9:02:36:2C:F9:D8:BF:71:9D:A1:BD:1B:1E:FC:94:6F:5B:4C:99:F4:2C:1B:9
Alias name: globalsignr3ca
  SHA1: D6:9B:56:11:48:F0:1C:77:C5:45:78:C1:09:26:DF:5B:85:69:76:AD
  SHA256:
 CB:B5:22:D7:B7:F1:27:AD:6A:01:13:86:5B:DF:1C:D4:10:2E:7D:07:59:AF:63:5A:7C:F4:72:0D:C9:63:C5:3
Alias name: globalsignrootca
  SHA1: B1:BC:96:8B:D4:F4:9D:62:2A:A8:9A:81:F2:15:01:52:A4:1D:82:9C
  SHA256:
 EB:D4:10:40:E4:BB:3E:C7:42:C9:E3:81:D3:1E:F2:A4:1A:48:B6:68:5C:96:E7:CE:F3:C1:DF:6C:D4:33:1C:9
Alias name: globalsignrootcar2
  SHA1: 75:E0:AB:B6:13:85:12:27:1C:04:F8:5F:DD:DE:38:E4:B7:24:2E:FE
  SHA256:
 CA:42:DD:41:74:5F:D0:B8:1E:B9:02:36:2C:F9:D8:BF:71:9D:A1:BD:1B:1E:FC:94:6F:5B:4C:99:F4:2C:1B:9
Alias name: globalsignrootcar3
  SHA1: D6:9B:56:11:48:F0:1C:77:C5:45:78:C1:09:26:DF:5B:85:69:76:AD
  SHA256:
 CB:B5:22:D7:B7:F1:27:AD:6A:01:13:86:5B:DF:1C:D4:10:2E:7D:07:59:AF:63:5A:7C:F4:72:0D:C9:63:C5:3
Alias name: globalsignrootcar6
  SHA1: 80:94:64:0E:B5:A7:A1:CA:11:9C:1F:DD:D5:9F:81:02:63:A7:FB:D1
  SHA256:
 2C:AB:EA:FE:37:D0:6C:A2:2A:BA:73:91:C0:03:3D:25:98:29:52:C4:53:64:73:49:76:3A:3A:B5:AD:6C:CF:6
Alias name: godaddyclass2ca
  SHA1: 27:96:BA:E6:3F:18:01:E2:77:26:1B:A0:D7:77:70:02:8F:20:EE:E4
  SHA256:
 C3:84:6B:F2:4B:9E:93:CA:64:27:4C:0E:C6:7C:1E:CC:5E:02:4F:FC:AC:D2:D7:40:19:35:0E:81:FE:54:6A:E
Alias name: godaddyrootcertificateauthorityg2
  SHA1: 47:BE:AB:C9:22:EA:E8:0E:78:78:34:62:A7:9F:45:C2:54:FD:E6:8B
  SHA256:
 45:14:0B:32:47:EB:9C:C8:C5:B4:F0:D7:B5:30:91:F7:32:92:08:9E:6E:5A:63:E2:74:9D:D3:AC:A9:19:8E:D
Alias name: godaddyrootg2ca
  SHA1: 47:BE:AB:C9:22:EA:E8:0E:78:78:34:62:A7:9F:45:C2:54:FD:E6:8B
  SHA256:
 45:14:0B:32:47:EB:9C:C8:C5:B4:F0:D7:B5:30:91:F7:32:92:08:9E:6E:5A:63:E2:74:9D:D3:AC:A9:19:8E:D
Alias name: gtsrootr1
  SHA1: E1:C9:50:E6:EF:22:F8:4C:56:45:72:8B:92:20:60:D7:D5:A7:A3:E8
  SHA256:
 2A:57:54:71:E3:13:40:BC:21:58:1C:BD:2C:F1:3E:15:84:63:20:3E:CE:94:BC:F9:D3:CC:19:6B:F0:9A:54:7
Alias name: gtsrootr2
```

```
  SHA1: D2:73:96:2A:2A:5E:39:9F:73:3F:E1:C7:1E:64:3F:03:38:34:FC:4D
  SHA256:
 C4:5D:7B:B0:8E:6D:67:E6:2E:42:35:11:0B:56:4E:5F:78:FD:92:EF:05:8C:84:0A:EA:4E:64:55:D7:58:5C:6
Alias name: gtsrootr3
  SHA1: 30:D4:24:6F:07:FF:DB:91:89:8A:0B:E9:49:66:11:EB:8C:5E:46:E5
  SHA256:
 15:D5:B8:77:46:19:EA:7D:54:CE:1C:A6:D0:B0:C4:03:E0:37:A9:17:F1:31:E8:A0:4E:1E:6B:7A:71:BA:BC:E
Alias name: gtsrootr4
  SHA1: 2A:1D:60:27:D9:4A:B1:0A:1C:4D:91:5C:CD:33:A0:CB:3E:2D:54:CB
  SHA256:
 71:CC:A5:39:1F:9E:79:4B:04:80:25:30:B3:63:E1:21:DA:8A:30:43:BB:26:66:2F:EA:4D:CA:7F:C9:51:A4:B
Alias name: hellenicacademicandresearchinstitutionseccrootca2015
  SHA1: 9F:F1:71:8D:92:D5:9A:F3:7D:74:97:B4:BC:6F:84:68:0B:BA:B6:66
  SHA256:
 44:B5:45:AA:8A:25:E6:5A:73:CA:15:DC:27:FC:36:D2:4C:1C:B9:95:3A:06:65:39:B1:15:82:DC:48:7B:48:3
Alias name: hellenicacademicandresearchinstitutionsrootca2011
  SHA1: FE:45:65:9B:79:03:5B:98:A1:61:B5:51:2E:AC:DA:58:09:48:22:4D
  SHA256:
 BC:10:4F:15:A4:8B:E7:09:DC:A5:42:A7:E1:D4:B9:DF:6F:05:45:27:E8:02:EA:A9:2D:59:54:44:25:8A:FE:7
Alias name: hellenicacademicandresearchinstitutionsrootca2015
  SHA1: 01:0C:06:95:A6:98:19:14:FF:BF:5F:C6:B0:B6:95:EA:29:E9:12:A6
  SHA256:
 A0:40:92:9A:02:CE:53:B4:AC:F4:F2:FF:C6:98:1C:E4:49:6F:75:5E:6D:45:FE:0B:2A:69:2B:CD:52:52:3F:3
Alias name: hongkongpostrootca1
  SHA1: D6:DA:A8:20:8D:09:D2:15:4D:24:B5:2F:CB:34:6E:B2:58:B2:8A:58
  SHA256:
 F9:E6:7D:33:6C:51:00:2A:C0:54:C6:32:02:2D:66:DD:A2:E7:E3:FF:F1:0A:D0:61:ED:31:D8:BB:B4:10:CF:B
Alias name: hongkongpostrootca3
  SHA1: 58:A2:D0:EC:20:52:81:5B:C1:F3:F8:64:02:24:4E:C2:8E:02:4B:02
  SHA256:
 5A:2F:C0:3F:0C:83:B0:90:BB:FA:40:60:4B:09:88:44:6C:76:36:18:3D:F9:84:6E:17:10:1A:44:7F:B8:EF:D
Alias name: identrustcommercialrootca1
  SHA1: DF:71:7E:AA:4A:D9:4E:C9:55:84:99:60:2D:48:DE:5F:BC:F0:3A:25
  SHA256:
 5D:56:49:9B:E4:D2:E0:8B:CF:CA:D0:8A:3E:38:72:3D:50:50:3B:DE:70:69:48:E4:2F:55:60:30:19:E5:28:A
Alias name: identrustpublicsectorrootca1
  SHA1: BA:29:41:60:77:98:3F:F4:F3:EF:F2:31:05:3B:2E:EA:6D:4D:45:FD
  SHA256:
 30:D0:89:5A:9A:44:8A:26:20:91:63:55:22:D1:F5:20:10:B5:86:7A:CA:E1:2C:78:EF:95:8F:D4:F4:38:9F:2
Alias name: isrgrootx1
  SHA1: CA:BD:2A:79:A1:07:6A:31:F2:1D:25:36:35:CB:03:9D:43:29:A5:E8
  SHA256:
 96:BC:EC:06:26:49:76:F3:74:60:77:9A:CF:28:C5:A7:CF:E8:A3:C0:AA:E1:1A:8F:FC:EE:05:C0:BD:DF:08:C
Alias name: izenpecom
```

```
  SHA1: 2F:78:3D:25:52:18:A7:4A:65:39:71:B5:2C:A2:9C:45:15:6F:E9:19
  SHA256:
 25:30:CC:8E:98:32:15:02:BA:D9:6F:9B:1F:BA:1B:09:9E:2D:29:9E:0F:45:48:BB:91:4F:36:3B:C0:D4:53:1
Alias name: keynectisrootca
  SHA1: 9C:61:5C:4D:4D:85:10:3A:53:26:C2:4D:BA:EA:E4:A2:D2:D5:CC:97
  SHA256:
 42:10:F1:99:49:9A:9A:C3:3C:8D:E0:2B:A6:DB:AA:14:40:8B:DD:8A:6E:32:46:89:C1:92:2D:06:97:15:A3:3
Alias name: microseceszignorootca2009
  SHA1: 89:DF:74:FE:5C:F4:0F:4A:80:F9:E3:37:7D:54:DA:91:E1:01:31:8E
  SHA256:
 3C:5F:81:FE:A5:FA:B8:2C:64:BF:A2:EA:EC:AF:CD:E8:E0:77:FC:86:20:A7:CA:E5:37:16:3D:F3:6E:DB:F3:7
Alias name: mozillacert0.pem
  SHA1: 97:81:79:50:D8:1C:96:70:CC:34:D8:09:CF:79:44:31:36:7E:F4:74
  SHA256:
 A5:31:25:18:8D:21:10:AA:96:4B:02:C7:B7:C6:DA:32:03:17:08:94:E5:FB:71:FF:FB:66:67:D5:E6:81:0A:3
Alias name: mozillacert1.pem
  SHA1: 23:E5:94:94:51:95:F2:41:48:03:B4:D5:64:D2:A3:A3:F5:D8:8B:8C
  SHA256:
 B4:41:0B:73:E2:E6:EA:CA:47:FB:C4:2F:8F:A4:01:8A:F4:38:1D:C5:4C:FA:A8:44:50:46:1E:ED:09:45:4D:E
Alias name: mozillacert10.pem
  SHA1: 5F:3A:FC:0A:8B:64:F6:86:67:34:74:DF:7E:A9:A2:FE:F9:FA:7A:51
  SHA256:
 21:DB:20:12:36:60:BB:2E:D4:18:20:5D:A1:1E:E7:A8:5A:65:E2:BC:6E:55:B5:AF:7E:78:99:C8:A2:66:D9:2
Alias name: mozillacert100.pem
  SHA1: 58:E8:AB:B0:36:15:33:FB:80:F7:9B:1B:6D:29:D3:FF:8D:5F:00:F0
  SHA256:
 49:E7:A4:42:AC:F0:EA:62:87:05:00:54:B5:25:64:B6:50:E4:F4:9E:42:E3:48:D6:AA:38:E0:39:E9:57:B1:C
Alias name: mozillacert101.pem
  SHA1: 99:A6:9B:E6:1A:FE:88:6B:4D:2B:82:00:7C:B8:54:FC:31:7E:15:39
  SHA256:
 62:F2:40:27:8C:56:4C:4D:D8:BF:7D:9D:4F:6F:36:6E:A8:94:D2:2F:5F:34:D9:89:A9:83:AC:EC:2F:FF:ED:5
Alias name: mozillacert102.pem
  SHA1: 96:C9:1B:0B:95:B4:10:98:42:FA:D0:D8:22:79:FE:60:FA:B9:16:83
  SHA256:
 EE:C5:49:6B:98:8C:E9:86:25:B9:34:09:2E:EC:29:08:BE:D0:B0:F3:16:C2:D4:73:0C:84:EA:F1:F3:D3:48:8
Alias name: mozillacert103.pem
  SHA1: 70:C1:8D:74:B4:28:81:0A:E4:FD:A5:75:D7:01:9F:99:B0:3D:50:74
  SHA256:
 3C:FC:3C:14:D1:F6:84:FF:17:E3:8C:43:CA:44:0C:00:B9:67:EC:93:3E:8B:FE:06:4C:A1:D7:2C:90:F2:AD:B
Alias name: mozillacert104.pem
  SHA1: 4F:99:AA:93:FB:2B:D1:37:26:A1:99:4A:CE:7F:F0:05:F2:93:5D:1E
  SHA256:
 1C:01:C6:F4:DB:B2:FE:FC:22:55:8B:2B:CA:32:56:3F:49:84:4A:CF:C3:2B:7B:E4:B0:FF:59:9F:9E:8C:7A:F
Alias name: mozillacert105.pem
```

```
  SHA1: 77:47:4F:C6:30:E4:0F:4C:47:64:3F:84:BA:B8:C6:95:4A:8A:41:EC
  SHA256:
 F0:9B:12:2C:71:14:F4:A0:9B:D4:EA:4F:4A:99:D5:58:B4:6E:4C:25:CD:81:14:0D:29:C0:56:13:91:4C:38:4
Alias name: mozillacert106.pem
  SHA1: E7:A1:90:29:D3:D5:52:DC:0D:0F:C6:92:D3:EA:88:0D:15:2E:1A:6B
  SHA256:
 D9:5F:EA:3C:A4:EE:DC:E7:4C:D7:6E:75:FC:6D:1F:F6:2C:44:1F:0F:A8:BC:77:F0:34:B1:9E:5D:B2:58:01:5
Alias name: mozillacert107.pem
  SHA1: 8E:1C:74:F8:A6:20:B9:E5:8A:F4:61:FA:EC:2B:47:56:51:1A:52:C6
  SHA256:
 F9:6F:23:F4:C3:E7:9C:07:7A:46:98:8D:5A:F5:90:06:76:A0:F0:39:CB:64:5D:D1:75:49:B2:16:C8:24:40:C
Alias name: mozillacert108.pem
  SHA1: B1:BC:96:8B:D4:F4:9D:62:2A:A8:9A:81:F2:15:01:52:A4:1D:82:9C
  SHA256:
 EB:D4:10:40:E4:BB:3E:C7:42:C9:E3:81:D3:1E:F2:A4:1A:48:B6:68:5C:96:E7:CE:F3:C1:DF:6C:D4:33:1C:9
Alias name: mozillacert109.pem
  SHA1: B5:61:EB:EA:A4:DE:E4:25:4B:69:1A:98:A5:57:47:C2:34:C7:D9:71
  SHA256:
 E2:3D:4A:03:6D:7B:70:E9:F5:95:B1:42:20:79:D2:B9:1E:DF:BB:1F:B6:51:A0:63:3E:AA:8A:9D:C5:F8:07:0
Alias name: mozillacert11.pem
  SHA1: 05:63:B8:63:0D:62:D7:5A:BB:C8:AB:1E:4B:DF:B5:A8:99:B2:4D:43
  SHA256:
 3E:90:99:B5:01:5E:8F:48:6C:00:BC:EA:9D:11:1E:E7:21:FA:BA:35:5A:89:BC:F1:DF:69:56:1E:3D:C6:32:5
Alias name: mozillacert110.pem
  SHA1: 93:05:7A:88:15:C6:4F:CE:88:2F:FA:91:16:52:28:78:BC:53:64:17
  SHA256:
 9A:6E:C0:12:E1:A7:DA:9D:BE:34:19:4D:47:8A:D7:C0:DB:18:22:FB:07:1D:F1:29:81:49:6E:D1:04:38:41:1
Alias name: mozillacert111.pem
  SHA1: 9C:BB:48:53:F6:A4:F6:D3:52:A4:E8:32:52:55:60:13:F5:AD:AF:65
  SHA256:
 59:76:90:07:F7:68:5D:0F:CD:50:87:2F:9F:95:D5:75:5A:5B:2B:45:7D:81:F3:69:2B:61:0A:98:67:2F:0E:1
Alias name: mozillacert112.pem
  SHA1: 43:13:BB:96:F1:D5:86:9B:C1:4E:6A:92:F6:CF:F6:34:69:87:82:37
  SHA256:
 DD:69:36:FE:21:F8:F0:77:C1:23:A1:A5:21:C1:22:24:F7:22:55:B7:3E:03:A7:26:06:93:E8:A2:4B:0F:A3:8
Alias name: mozillacert113.pem
  SHA1: 50:30:06:09:1D:97:D4:F5:AE:39:F7:CB:E7:92:7D:7D:65:2D:34:31
  SHA256:
 6D:C4:71:72:E0:1C:BC:B0:BF:62:58:0D:89:5F:E2:B8:AC:9A:D4:F8:73:80:1E:0C:10:B9:C8:37:D2:1E:B1:7
Alias name: mozillacert114.pem
  SHA1: 51:C6:E7:08:49:06:6E:F3:92:D4:5C:A0:0D:6D:A3:62:8F:C3:52:39
  SHA256:
 B0:BF:D5:2B:B0:D7:D9:BD:92:BF:5D:4D:C1:3D:A2:55:C0:2C:54:2F:37:83:65:EA:89:39:11:F5:5E:55:F2:3
Alias name: mozillacert115.pem
```

```
  SHA1: 59:0D:2D:7D:88:4F:40:2E:61:7E:A5:62:32:17:65:CF:17:D8:94:E9
  SHA256:
 91:E2:F5:78:8D:58:10:EB:A7:BA:58:73:7D:E1:54:8A:8E:CA:CD:01:45:98:BC:0B:14:3E:04:1B:17:05:25:5
Alias name: mozillacert116.pem
  SHA1: 2B:B1:F5:3E:55:0C:1D:C5:F1:D4:E6:B7:6A:46:4B:55:06:02:AC:21
  SHA256:
 F3:56:BE:A2:44:B7:A9:1E:B3:5D:53:CA:9A:D7:86:4A:CE:01:8E:2D:35:D5:F8:F9:6D:DF:68:A6:F4:1A:A4:7
Alias name: mozillacert117.pem
  SHA1: D4:DE:20:D0:5E:66:FC:53:FE:1A:50:88:2C:78:DB:28:52:CA:E4:74
  SHA256:
 16:AF:57:A9:F6:76:B0:AB:12:60:95:AA:5E:BA:DE:F2:2A:B3:11:19:D6:44:AC:95:CD:4B:93:DB:F3:F2:6A:E
Alias name: mozillacert118.pem
  SHA1: 7E:78:4A:10:1C:82:65:CC:2D:E1:F1:6D:47:B4:40:CA:D9:0A:19:45
  SHA256:
 5F:0B:62:EA:B5:E3:53:EA:65:21:65:16:58:FB:B6:53:59:F4:43:28:0A:4A:FB:D1:04:D7:7D:10:F9:F0:4C:0
Alias name: mozillacert119.pem
  SHA1: 75:E0:AB:B6:13:85:12:27:1C:04:F8:5F:DD:DE:38:E4:B7:24:2E:FE
  SHA256:
 CA:42:DD:41:74:5F:D0:B8:1E:B9:02:36:2C:F9:D8:BF:71:9D:A1:BD:1B:1E:FC:94:6F:5B:4C:99:F4:2C:1B:9
Alias name: mozillacert12.pem
  SHA1: A8:98:5D:3A:65:E5:E5:C4:B2:D7:D6:6D:40:C6:DD:2F:B1:9C:54:36
  SHA256:
 43:48:A0:E9:44:4C:78:CB:26:5E:05:8D:5E:89:44:B4:D8:4F:96:62:BD:26:DB:25:7F:89:34:A4:43:C7:01:6
Alias name: mozillacert120.pem
  SHA1: DA:40:18:8B:91:89:A3:ED:EE:AE:DA:97:FE:2F:9D:F5:B7:D1:8A:41
  SHA256:
 CF:56:FF:46:A4:A1:86:10:9D:D9:65:84:B5:EE:B5:8A:51:0C:42:75:B0:E5:F9:4F:40:BB:AE:86:5E:19:F6:7
Alias name: mozillacert121.pem
  SHA1: CC:AB:0E:A0:4C:23:01:D6:69:7B:DD:37:9F:CD:12:EB:24:E3:94:9D
  SHA256:
 8C:72:09:27:9A:C0:4E:27:5E:16:D0:7F:D3:B7:75:E8:01:54:B5:96:80:46:E3:1F:52:DD:25:76:63:24:E9:A
Alias name: mozillacert122.pem
  SHA1: 02:FA:F3:E2:91:43:54:68:60:78:57:69:4D:F5:E4:5B:68:85:18:68
  SHA256:
 68:7F:A4:51:38:22:78:FF:F0:C8:B1:1F:8D:43:D5:76:67:1C:6E:B2:BC:EA:B4:13:FB:83:D9:65:D0:6D:2F:F
Alias name: mozillacert123.pem
  SHA1: 2A:B6:28:48:5E:78:FB:F3:AD:9E:79:10:DD:6B:DF:99:72:2C:96:E5
  SHA256:
 07:91:CA:07:49:B2:07:82:AA:D3:C7:D7:BD:0C:DF:C9:48:58:35:84:3E:B2:D7:99:60:09:CE:43:AB:6C:69:2
Alias name: mozillacert124.pem
  SHA1: 4D:23:78:EC:91:95:39:B5:00:7F:75:8F:03:3B:21:1E:C5:4D:8B:CF
  SHA256:
 80:95:21:08:05:DB:4B:BC:35:5E:44:28:D8:FD:6E:C2:CD:E3:AB:5F:B9:7A:99:42:98:8E:B8:F4:DC:D0:60:1
Alias name: mozillacert125.pem
```

```
  SHA1: B3:1E:B1:B7:40:E3:6C:84:02:DA:DC:37:D4:4D:F5:D4:67:49:52:F9
  SHA256:
 73:C1:76:43:4F:1B:C6:D5:AD:F4:5B:0E:76:E7:27:28:7C:8D:E5:76:16:C1:E6:E6:14:1A:2B:2C:BC:7D:8E:4
Alias name: mozillacert126.pem
  SHA1: 25:01:90:19:CF:FB:D9:99:1C:B7:68:25:74:8D:94:5F:30:93:95:42
  SHA256:
 AF:8B:67:62:A1:E5:28:22:81:61:A9:5D:5C:55:9E:E2:66:27:8F:75:D7:9E:83:01:89:A5:03:50:6A:BD:6B:4
Alias name: mozillacert127.pem
  SHA1: DE:28:F4:A4:FF:E5:B9:2F:A3:C5:03:D1:A3:49:A7:F9:96:2A:82:12
  SHA256:
 FF:85:6A:2D:25:1D:CD:88:D3:66:56:F4:50:12:67:98:CF:AB:AA:DE:40:79:9C:72:2D:E4:D2:B5:DB:36:A7:3
Alias name: mozillacert128.pem
  SHA1: A9:E9:78:08:14:37:58:88:F2:05:19:B0:6D:2B:0D:2B:60:16:90:7D
  SHA256:
 CA:2D:82:A0:86:77:07:2F:8A:B6:76:4F:F0:35:67:6C:FE:3E:5E:32:5E:01:21:72:DF:3F:92:09:6D:B7:9B:8
Alias name: mozillacert129.pem
  SHA1: E6:21:F3:35:43:79:05:9A:4B:68:30:9D:8A:2F:74:22:15:87:EC:79
  SHA256:
 A0:45:9B:9F:63:B2:25:59:F5:FA:5D:4C:6D:B3:F9:F7:2F:F1:93:42:03:35:78:F0:73:BF:1D:1B:46:CB:B9:1
Alias name: mozillacert13.pem
  SHA1: 06:08:3F:59:3F:15:A1:04:A0:69:A4:6B:A9:03:D0:06:B7:97:09:91
  SHA256:
 6C:61:DA:C3:A2:DE:F0:31:50:6B:E0:36:D2:A6:FE:40:19:94:FB:D1:3D:F9:C8:D4:66:59:92:74:C4:46:EC:9
Alias name: mozillacert130.pem
  SHA1: E5:DF:74:3C:B6:01:C4:9B:98:43:DC:AB:8C:E8:6A:81:10:9F:E4:8E
  SHA256:
 F4:C1:49:55:1A:30:13:A3:5B:C7:BF:FE:17:A7:F3:44:9B:C1:AB:5B:5A:0A:E7:4B:06:C2:3B:90:00:4C:01:0
Alias name: mozillacert131.pem
  SHA1: 37:9A:19:7B:41:85:45:35:0C:A6:03:69:F3:3C:2E:AF:47:4F:20:79
  SHA256:
 A0:23:4F:3B:C8:52:7C:A5:62:8E:EC:81:AD:5D:69:89:5D:A5:68:0D:C9:1D:1C:B8:47:7F:33:F8:78:B9:5B:0
Alias name: mozillacert132.pem
  SHA1: 39:21:C1:15:C1:5D:0E:CA:5C:CB:5B:C4:F0:7D:21:D8:05:0B:56:6A
  SHA256:
 77:40:73:12:C6:3A:15:3D:5B:C0:0B:4E:51:75:9C:DF:DA:C2:37:DC:2A:33:B6:79:46:E9:8E:9B:FA:68:0A:E
Alias name: mozillacert133.pem
  SHA1: 85:B5:FF:67:9B:0C:79:96:1F:C8:6E:44:22:00:46:13:DB:17:92:84
  SHA256:
 7D:3B:46:5A:60:14:E5:26:C0:AF:FC:EE:21:27:D2:31:17:27:AD:81:1C:26:84:2D:00:6A:F3:73:06:CC:80:B
Alias name: mozillacert134.pem
  SHA1: 70:17:9B:86:8C:00:A4:FA:60:91:52:22:3F:9F:3E:32:BD:E0:05:62
  SHA256:
 69:FA:C9:BD:55:FB:0A:C7:8D:53:BB:EE:5C:F1:D5:97:98:9F:D0:AA:AB:20:A2:51:51:BD:F1:73:3E:E7:D1:2
Alias name: mozillacert135.pem
```

```
  SHA1: 62:52:DC:40:F7:11:43:A2:2F:DE:9E:F7:34:8E:06:42:51:B1:81:18
  SHA256:
 D8:E0:FE:BC:1D:B2:E3:8D:00:94:0F:37:D2:7D:41:34:4D:99:3E:73:4B:99:D5:65:6D:97:78:D4:D8:14:36:2
Alias name: mozillacert136.pem
  SHA1: D1:EB:23:A4:6D:17:D6:8F:D9:25:64:C2:F1:F1:60:17:64:D8:E3:49
  SHA256:
 D7:A7:A0:FB:5D:7E:27:31:D7:71:E9:48:4E:BC:DE:F7:1D:5F:0C:3E:0A:29:48:78:2B:C8:3E:E0:EA:69:9E:F
Alias name: mozillacert137.pem
  SHA1: 4A:65:D5:F4:1D:EF:39:B8:B8:90:4A:4A:D3:64:81:33:CF:C7:A1:D1
  SHA256:
 BD:81:CE:3B:4F:65:91:D1:1A:67:B5:FC:7A:47:FD:EF:25:52:1B:F9:AA:4E:18:B9:E3:DF:2E:34:A7:80:3B:E
Alias name: mozillacert138.pem
  SHA1: E1:9F:E3:0E:8B:84:60:9E:80:9B:17:0D:72:A8:C5:BA:6E:14:09:BD
  SHA256:
 3F:06:E5:56:81:D4:96:F5:BE:16:9E:B5:38:9F:9F:2B:8F:F6:1E:17:08:DF:68:81:72:48:49:CD:5D:27:CB:6
Alias name: mozillacert139.pem
  SHA1: DE:3F:40:BD:50:93:D3:9B:6C:60:F6:DA:BC:07:62:01:00:89:76:C9
  SHA256:
 A4:5E:DE:3B:BB:F0:9C:8A:E1:5C:72:EF:C0:72:68:D6:93:A2:1C:99:6F:D5:1E:67:CA:07:94:60:FD:6D:88:7
Alias name: mozillacert14.pem
  SHA1: 5F:B7:EE:06:33:E2:59:DB:AD:0C:4C:9A:E6:D3:8F:1A:61:C7:DC:25
  SHA256:
 74:31:E5:F4:C3:C1:CE:46:90:77:4F:0B:61:E0:54:40:88:3B:A9:A0:1E:D0:0B:A6:AB:D7:80:6E:D3:B1:18:C
Alias name: mozillacert140.pem
  SHA1: CA:3A:FB:CF:12:40:36:4B:44:B2:16:20:88:80:48:39:19:93:7C:F7
  SHA256:
 85:A0:DD:7D:D7:20:AD:B7:FF:05:F8:3D:54:2B:20:9D:C7:FF:45:28:F7:D6:77:B1:83:89:FE:A5:E5:C4:9E:8
Alias name: mozillacert141.pem
  SHA1: 31:7A:2A:D0:7F:2B:33:5E:F5:A1:C3:4E:4B:57:E8:B7:D8:F1:FC:A6
  SHA256:
 58:D0:17:27:9C:D4:DC:63:AB:DD:B1:96:A6:C9:90:6C:30:C4:E0:87:83:EA:E8:C1:60:99:54:D6:93:55:59:6
Alias name: mozillacert142.pem
  SHA1: 1F:49:14:F7:D8:74:95:1D:DD:AE:02:C0:BE:FD:3A:2D:82:75:51:85
  SHA256:
 18:F1:FC:7F:20:5D:F8:AD:DD:EB:7F:E0:07:DD:57:E3:AF:37:5A:9C:4D:8D:73:54:6B:F4:F1:FE:D1:E1:8D:3
Alias name: mozillacert143.pem
  SHA1: 36:B1:2B:49:F9:81:9E:D7:4C:9E:BC:38:0F:C6:56:8F:5D:AC:B2:F7
  SHA256:
 E7:5E:72:ED:9F:56:0E:EC:6E:B4:80:00:73:A4:3F:C3:AD:19:19:5A:39:22:82:01:78:95:97:4A:99:02:6B:6
Alias name: mozillacert144.pem
  SHA1: 37:F7:6D:E6:07:7C:90:C5:B1:3E:93:1A:B7:41:10:B4:F2:E4:9A:27
  SHA256:
 79:08:B4:03:14:C1:38:10:0B:51:8D:07:35:80:7F:FB:FC:F8:51:8A:00:95:33:71:05:BA:38:6B:15:3D:D9:2
Alias name: mozillacert145.pem
```

```
  SHA1: 10:1D:FA:3F:D5:0B:CB:BB:9B:B5:60:0C:19:55:A4:1A:F4:73:3A:04
  SHA256:
 D4:1D:82:9E:8C:16:59:82:2A:F9:3F:CE:62:BF:FC:DE:26:4F:C8:4E:8B:95:0C:5F:F2:75:D0:52:35:46:95:A
Alias name: mozillacert146.pem
  SHA1: 21:FC:BD:8E:7F:6C:AF:05:1B:D1:B3:43:EC:A8:E7:61:47:F2:0F:8A
  SHA256:
 48:98:C6:88:8C:0C:FF:B0:D3:E3:1A:CA:8A:37:D4:E3:51:5F:F7:46:D0:26:35:D8:66:46:CF:A0:A3:18:5A:E
Alias name: mozillacert147.pem
  SHA1: 58:11:9F:0E:12:82:87:EA:50:FD:D9:87:45:6F:4F:78:DC:FA:D6:D4
  SHA256:
 85:FB:2F:91:DD:12:27:5A:01:45:B6:36:53:4F:84:02:4A:D6:8B:69:B8:EE:88:68:4F:F7:11:37:58:05:B3:4
Alias name: mozillacert148.pem
  SHA1: 04:83:ED:33:99:AC:36:08:05:87:22:ED:BC:5E:46:00:E3:BE:F9:D7
  SHA256:
 6E:A5:47:41:D0:04:66:7E:ED:1B:48:16:63:4A:A3:A7:9E:6E:4B:96:95:0F:82:79:DA:FC:8D:9B:D8:81:21:3
Alias name: mozillacert149.pem
  SHA1: 6E:3A:55:A4:19:0C:19:5C:93:84:3C:C0:DB:72:2E:31:30:61:F0:B1
  SHA256:
 0C:25:8A:12:A5:67:4A:EF:25:F2:8B:A7:DC:FA:EC:EE:A3:48:E5:41:E6:F5:CC:4E:E6:3B:71:B3:61:60:6A:C
Alias name: mozillacert15.pem
  SHA1: 74:20:74:41:72:9C:DD:92:EC:79:31:D8:23:10:8D:C2:81:92:E2:BB
  SHA256:
 0F:99:3C:8A:EF:97:BA:AF:56:87:14:0E:D5:9A:D1:82:1B:B4:AF:AC:F0:AA:9A:58:B5:D5:7A:33:8A:3A:FB:C
Alias name: mozillacert150.pem
  SHA1: 33:9B:6B:14:50:24:9B:55:7A:01:87:72:84:D9:E0:2F:C3:D2:D8:E9
  SHA256:
 EF:3C:B4:17:FC:8E:BF:6F:97:87:6C:9E:4E:CE:39:DE:1E:A5:FE:64:91:41:D1:02:8B:7D:11:C0:B2:29:8C:E
Alias name: mozillacert151.pem
  SHA1: AC:ED:5F:65:53:FD:25:CE:01:5F:1F:7A:48:3B:6A:74:9F:61:78:C6
  SHA256:
 7F:12:CD:5F:7E:5E:29:0E:C7:D8:51:79:D5:B7:2C:20:A5:BE:75:08:FF:DB:5B:F8:1A:B9:68:4A:7F:C9:F6:6
Alias name: mozillacert16.pem
  SHA1: DA:C9:02:4F:54:D8:F6:DF:94:93:5F:B1:73:26:38:CA:6A:D7:7C:13
  SHA256:
 06:87:26:03:31:A7:24:03:D9:09:F1:05:E6:9B:CF:0D:32:E1:BD:24:93:FF:C6:D9:20:6D:11:BC:D6:77:07:3
Alias name: mozillacert17.pem
  SHA1: 40:54:DA:6F:1C:3F:40:74:AC:ED:0F:EC:CD:DB:79:D1:53:FB:90:1D
  SHA256:
 76:7C:95:5A:76:41:2C:89:AF:68:8E:90:A1:C7:0F:55:6C:FD:6B:60:25:DB:EA:10:41:6D:7E:B6:83:1F:8C:4
Alias name: mozillacert18.pem
  SHA1: 79:98:A3:08:E1:4D:65:85:E6:C2:1E:15:3A:71:9F:BA:5A:D3:4A:D9
  SHA256:
 44:04:E3:3B:5E:14:0D:CF:99:80:51:FD:FC:80:28:C7:C8:16:15:C5:EE:73:7B:11:1B:58:82:33:A9:B5:35:A
Alias name: mozillacert19.pem
```

```
  SHA1: B4:35:D4:E1:11:9D:1C:66:90:A7:49:EB:B3:94:BD:63:7B:A7:82:B7
  SHA256:
 C4:70:CF:54:7E:23:02:B9:77:FB:29:DD:71:A8:9A:7B:6C:1F:60:77:7B:03:29:F5:60:17:F3:28:BF:4F:6B:E
Alias name: mozillacert2.pem
  SHA1: 22:D5:D8:DF:8F:02:31:D1:8D:F7:9D:B7:CF:8A:2D:64:C9:3F:6C:3A
  SHA256:
 69:DD:D7:EA:90:BB:57:C9:3E:13:5D:C8:5E:A6:FC:D5:48:0B:60:32:39:BD:C4:54:FC:75:8B:2A:26:CF:7F:7
Alias name: mozillacert20.pem
  SHA1: D8:C5:38:8A:B7:30:1B:1B:6E:D4:7A:E6:45:25:3A:6F:9F:1A:27:61
  SHA256:
 62:DD:0B:E9:B9:F5:0A:16:3E:A0:F8:E7:5C:05:3B:1E:CA:57:EA:55:C8:68:8F:64:7C:68:81:F2:C8:35:7B:9
Alias name: mozillacert21.pem
  SHA1: 9B:AA:E5:9F:56:EE:21:CB:43:5A:BE:25:93:DF:A7:F0:40:D1:1D:CB
  SHA256:
 BE:6C:4D:A2:BB:B9:BA:59:B6:F3:93:97:68:37:42:46:C3:C0:05:99:3F:A9:8F:02:0D:1D:ED:BE:D4:8A:81:D
Alias name: mozillacert22.pem
  SHA1: 32:3C:11:8E:1B:F7:B8:B6:52:54:E2:E2:10:0D:D6:02:90:37:F0:96
  SHA256:
 37:D5:10:06:C5:12:EA:AB:62:64:21:F1:EC:8C:92:01:3F:C5:F8:2A:E9:8E:E5:33:EB:46:19:B8:DE:B4:D0:6
Alias name: mozillacert23.pem
  SHA1: 91:C6:D6:EE:3E:8A:C8:63:84:E5:48:C2:99:29:5C:75:6C:81:7B:81
  SHA256:
 8D:72:2F:81:A9:C1:13:C0:79:1D:F1:36:A2:96:6D:B2:6C:95:0A:97:1D:B4:6B:41:99:F4:EA:54:B7:8B:FB:9
Alias name: mozillacert24.pem
  SHA1: 59:AF:82:79:91:86:C7:B4:75:07:CB:CF:03:57:46:EB:04:DD:B7:16
  SHA256:
 66:8C:83:94:7D:A6:3B:72:4B:EC:E1:74:3C:31:A0:E6:AE:D0:DB:8E:C5:B3:1B:E3:77:BB:78:4F:91:B6:71:6
Alias name: mozillacert25.pem
  SHA1: 4E:B6:D5:78:49:9B:1C:CF:5F:58:1E:AD:56:BE:3D:9B:67:44:A5:E5
  SHA256:
 9A:CF:AB:7E:43:C8:D8:80:D0:6B:26:2A:94:DE:EE:E4:B4:65:99:89:C3:D0:CA:F1:9B:AF:64:05:E4:1A:B7:D
Alias name: mozillacert26.pem
  SHA1: 87:82:C6:C3:04:35:3B:CF:D2:96:92:D2:59:3E:7D:44:D9:34:FF:11
  SHA256:
 F1:C1:B5:0A:E5:A2:0D:D8:03:0E:C9:F6:BC:24:82:3D:D3:67:B5:25:57:59:B4:E7:1B:61:FC:E9:F7:37:5D:7
Alias name: mozillacert27.pem
  SHA1: 3A:44:73:5A:E5:81:90:1F:24:86:61:46:1E:3B:9C:C4:5F:F5:3A:1B
  SHA256:
 42:00:F5:04:3A:C8:59:0E:BB:52:7D:20:9E:D1:50:30:29:FB:CB:D4:1C:A1:B5:06:EC:27:F1:5A:DE:7D:AC:6
Alias name: mozillacert28.pem
  SHA1: 66:31:BF:9E:F7:4F:9E:B6:C9:D5:A6:0C:BA:6A:BE:D1:F7:BD:EF:7B
  SHA256:
 0C:2C:D6:3D:F7:80:6F:A3:99:ED:E8:09:11:6B:57:5B:F8:79:89:F0:65:18:F9:80:8C:86:05:03:17:8B:AF:6
Alias name: mozillacert29.pem
```

```
  SHA1: 74:F8:A3:C3:EF:E7:B3:90:06:4B:83:90:3C:21:64:60:20:E5:DF:CE
  SHA256:
 15:F0:BA:00:A3:AC:7A:F3:AC:88:4C:07:2B:10:11:A0:77:BD:77:C0:97:F4:01:64:B2:F8:59:8A:BD:83:86:0
Alias name: mozillacert3.pem
  SHA1: 87:9F:4B:EE:05:DF:98:58:3B:E3:60:D6:33:E7:0D:3F:FE:98:71:AF
  SHA256:
 39:DF:7B:68:2B:7B:93:8F:84:71:54:81:CC:DE:8D:60:D8:F2:2E:C5:98:87:7D:0A:AA:C1:2B:59:18:2B:03:1
Alias name: mozillacert30.pem
  SHA1: E7:B4:F6:9D:61:EC:90:69:DB:7E:90:A7:40:1A:3C:F4:7D:4F:E8:EE
  SHA256:
 A7:12:72:AE:AA:A3:CF:E8:72:7F:7F:B3:9F:0F:B3:D1:E5:42:6E:90:60:B0:6E:E6:F1:3E:9A:3C:58:33:CD:4
Alias name: mozillacert31.pem
  SHA1: 9F:74:4E:9F:2B:4D:BA:EC:0F:31:2C:50:B6:56:3B:8E:2D:93:C3:11
  SHA256:
 17:93:92:7A:06:14:54:97:89:AD:CE:2F:8F:34:F7:F0:B6:6D:0F:3A:E3:A3:B8:4D:21:EC:15:DB:BA:4F:AD:C
Alias name: mozillacert32.pem
  SHA1: 60:D6:89:74:B5:C2:65:9E:8A:0F:C1:88:7C:88:D2:46:69:1B:18:2C
  SHA256:
 B9:BE:A7:86:0A:96:2E:A3:61:1D:AB:97:AB:6D:A3:E2:1C:10:68:B9:7D:55:57:5E:D0:E1:12:79:C1:1C:89:3
Alias name: mozillacert33.pem
  SHA1: FE:B8:C4:32:DC:F9:76:9A:CE:AE:3D:D8:90:8F:FD:28:86:65:64:7D
  SHA256:
 A2:2D:BA:68:1E:97:37:6E:2D:39:7D:72:8A:AE:3A:9B:62:96:B9:FD:BA:60:BC:2E:11:F6:47:F2:C6:75:FB:3
Alias name: mozillacert34.pem
  SHA1: 59:22:A1:E1:5A:EA:16:35:21:F8:98:39:6A:46:46:B0:44:1B:0F:A9
  SHA256:
 41:C9:23:86:6A:B4:CA:D6:B7:AD:57:80:81:58:2E:02:07:97:A6:CB:DF:4F:FF:78:CE:83:96:B3:89:37:D7:F
Alias name: mozillacert35.pem
  SHA1: 2A:C8:D5:8B:57:CE:BF:2F:49:AF:F2:FC:76:8F:51:14:62:90:7A:41
  SHA256:
 92:BF:51:19:AB:EC:CA:D0:B1:33:2D:C4:E1:D0:5F:BA:75:B5:67:90:44:EE:0C:A2:6E:93:1F:74:4F:2F:33:C
Alias name: mozillacert36.pem
  SHA1: 23:88:C9:D3:71:CC:9E:96:3D:FF:7D:3C:A7:CE:FC:D6:25:EC:19:0D
  SHA256:
 32:7A:3D:76:1A:BA:DE:A0:34:EB:99:84:06:27:5C:B1:A4:77:6E:FD:AE:2F:DF:6D:01:68:EA:1C:4F:55:67:D
Alias name: mozillacert37.pem
  SHA1: B1:2E:13:63:45:86:A4:6F:1A:B2:60:68:37:58:2D:C4:AC:FD:94:97
  SHA256:
 E3:B6:A2:DB:2E:D7:CE:48:84:2F:7A:C5:32:41:C7:B7:1D:54:14:4B:FB:40:C1:1F:3F:1D:0B:42:F5:EE:A1:2
Alias name: mozillacert38.pem
  SHA1: CB:A1:C5:F8:B0:E3:5E:B8:B9:45:12:D3:F9:34:A2:E9:06:10:D3:36
  SHA256:
 A6:C5:1E:0D:A5:CA:0A:93:09:D2:E4:C0:E4:0C:2A:F9:10:7A:AE:82:03:85:7F:E1:98:E3:E7:69:E3:43:08:5
Alias name: mozillacert39.pem
```

```
  SHA1: AE:50:83:ED:7C:F4:5C:BC:8F:61:C6:21:FE:68:5D:79:42:21:15:6E
  SHA256:
 E6:B8:F8:76:64:85:F8:07:AE:7F:8D:AC:16:70:46:1F:07:C0:A1:3E:EF:3A:1F:F7:17:53:8D:7A:BA:D3:91:B
Alias name: mozillacert4.pem
  SHA1: E3:92:51:2F:0A:CF:F5:05:DF:F6:DE:06:7F:75:37:E1:65:EA:57:4B
  SHA256:
 0B:5E:ED:4E:84:64:03:CF:55:E0:65:84:84:40:ED:2A:82:75:8B:F5:B9:AA:1F:25:3D:46:13:CF:A0:80:FF:3
Alias name: mozillacert40.pem
  SHA1: 80:25:EF:F4:6E:70:C8:D4:72:24:65:84:FE:40:3B:8A:8D:6A:DB:F5
  SHA256:
 8D:A0:84:FC:F9:9C:E0:77:22:F8:9B:32:05:93:98:06:FA:5C:B8:11:E1:C8:13:F6:A1:08:C7:D3:36:B3:40:8
Alias name: mozillacert41.pem
  SHA1: 6B:2F:34:AD:89:58:BE:62:FD:B0:6B:5C:CE:BB:9D:D9:4F:4E:39:F3
  SHA256:
 EB:F3:C0:2A:87:89:B1:FB:7D:51:19:95:D6:63:B7:29:06:D9:13:CE:0D:5E:10:56:8A:8A:77:E2:58:61:67:E
Alias name: mozillacert42.pem
  SHA1: 85:A4:08:C0:9C:19:3E:5D:51:58:7D:CD:D6:13:30:FD:8C:DE:37:BF
  SHA256:
 B6:19:1A:50:D0:C3:97:7F:7D:A9:9B:CD:AA:C8:6A:22:7D:AE:B9:67:9E:C7:0B:A3:B0:C9:D9:22:71:C1:70:D
Alias name: mozillacert43.pem
  SHA1: F9:CD:0E:2C:DA:76:24:C1:8F:BD:F0:F0:AB:B6:45:B8:F7:FE:D5:7A
  SHA256:
 50:79:41:C7:44:60:A0:B4:70:86:22:0D:4E:99:32:57:2A:B5:D1:B5:BB:CB:89:80:AB:1C:B1:76:51:A8:44:D
Alias name: mozillacert44.pem
  SHA1: 5F:43:E5:B1:BF:F8:78:8C:AC:1C:C7:CA:4A:9A:C6:22:2B:CC:34:C6
  SHA256:
 96:0A:DF:00:63:E9:63:56:75:0C:29:65:DD:0A:08:67:DA:0B:9C:BD:6E:77:71:4A:EA:FB:23:49:AB:39:3D:A
Alias name: mozillacert45.pem
  SHA1: 67:65:0D:F1:7E:8E:7E:5B:82:40:A4:F4:56:4B:CF:E2:3D:69:C6:F0
  SHA256:
 C0:A6:F4:DC:63:A2:4B:FD:CF:54:EF:2A:6A:08:2A:0A:72:DE:35:80:3E:2F:F5:FF:52:7A:E5:D8:72:06:DF:D
Alias name: mozillacert46.pem
  SHA1: 40:9D:4B:D9:17:B5:5C:27:B6:9B:64:CB:98:22:44:0D:CD:09:B8:89
  SHA256:
 EC:C3:E9:C3:40:75:03:BE:E0:91:AA:95:2F:41:34:8F:F8:8B:AA:86:3B:22:64:BE:FA:C8:07:90:15:74:E9:3
Alias name: mozillacert47.pem
  SHA1: 1B:4B:39:61:26:27:6B:64:91:A2:68:6D:D7:02:43:21:2D:1F:1D:96
  SHA256:
 E4:C7:34:30:D7:A5:B5:09:25:DF:43:37:0A:0D:21:6E:9A:79:B9:D6:DB:83:73:A0:C6:9E:B1:CC:31:C7:C5:2
Alias name: mozillacert48.pem
  SHA1: A0:A1:AB:90:C9:FC:84:7B:3B:12:61:E8:97:7D:5F:D3:22:61:D3:CC
  SHA256:
 0F:4E:9C:DD:26:4B:02:55:50:D1:70:80:63:40:21:4F:E9:44:34:C9:B0:2F:69:7E:C7:10:FC:5F:EA:FB:5E:3
Alias name: mozillacert49.pem
```

```
  SHA1: 61:57:3A:11:DF:0E:D8:7E:D5:92:65:22:EA:D0:56:D7:44:B3:23:71
  SHA256:
 B7:B1:2B:17:1F:82:1D:AA:99:0C:D0:FE:50:87:B1:28:44:8B:A8:E5:18:4F:84:C5:1E:02:B5:C8:FB:96:2B:2
Alias name: mozillacert5.pem
  SHA1: B8:01:86:D1:EB:9C:86:A5:41:04:CF:30:54:F3:4C:52:B7:E5:58:C6
  SHA256:
 CE:CD:DC:90:50:99:D8:DA:DF:C5:B1:D2:09:B7:37:CB:E2:C1:8C:FB:2C:10:C0:FF:0B:CF:0D:32:86:FC:1A:A
Alias name: mozillacert50.pem
  SHA1: 8C:96:BA:EB:DD:2B:07:07:48:EE:30:32:66:A0:F3:98:6E:7C:AE:58
  SHA256:
 35:AE:5B:DD:D8:F7:AE:63:5C:FF:BA:56:82:A8:F0:0B:95:F4:84:62:C7:10:8E:E9:A0:E5:29:2B:07:4A:AF:B
Alias name: mozillacert51.pem
  SHA1: FA:B7:EE:36:97:26:62:FB:2D:B0:2A:F6:BF:03:FD:E8:7C:4B:2F:9B
  SHA256:
 EA:A9:62:C4:FA:4A:6B:AF:EB:E4:15:19:6D:35:1C:CD:88:8D:4F:53:F3:FA:8A:E6:D7:C4:66:A9:4E:60:42:B
Alias name: mozillacert52.pem
  SHA1: 8B:AF:4C:9B:1D:F0:2A:92:F7:DA:12:8E:B9:1B:AC:F4:98:60:4B:6F
  SHA256:
 E2:83:93:77:3D:A8:45:A6:79:F2:08:0C:C7:FB:44:A3:B7:A1:C3:79:2C:B7:EB:77:29:FD:CB:6A:8D:99:AE:A
Alias name: mozillacert53.pem
  SHA1: 7F:8A:B0:CF:D0:51:87:6A:66:F3:36:0F:47:C8:8D:8C:D3:35:FC:74
  SHA256:
 2D:47:43:7D:E1:79:51:21:5A:12:F3:C5:8E:51:C7:29:A5:80:26:EF:1F:CC:0A:5F:B3:D9:DC:01:2F:60:0D:1
Alias name: mozillacert54.pem
  SHA1: 03:9E:ED:B8:0B:E7:A0:3C:69:53:89:3B:20:D2:D9:32:3A:4C:2A:FD
  SHA256:
 B4:78:B8:12:25:0D:F8:78:63:5C:2A:A7:EC:7D:15:5E:AA:62:5E:E8:29:16:E2:CD:29:43:61:88:6C:D1:FB:D
Alias name: mozillacert55.pem
  SHA1: AA:DB:BC:22:23:8F:C4:01:A1:27:BB:38:DD:F4:1D:DB:08:9E:F0:12
  SHA256:
 A4:31:0D:50:AF:18:A6:44:71:90:37:2A:86:AF:AF:8B:95:1F:FB:43:1D:83:7F:1E:56:88:B4:59:71:ED:15:5
Alias name: mozillacert56.pem
  SHA1: F1:8B:53:8D:1B:E9:03:B6:A6:F0:56:43:5B:17:15:89:CA:F3:6B:F2
  SHA256:
 4B:03:F4:58:07:AD:70:F2:1B:FC:2C:AE:71:C9:FD:E4:60:4C:06:4C:F5:FF:B6:86:BA:E5:DB:AA:D7:FD:D3:4
Alias name: mozillacert57.pem
  SHA1: D6:DA:A8:20:8D:09:D2:15:4D:24:B5:2F:CB:34:6E:B2:58:B2:8A:58
  SHA256:
 F9:E6:7D:33:6C:51:00:2A:C0:54:C6:32:02:2D:66:DD:A2:E7:E3:FF:F1:0A:D0:61:ED:31:D8:BB:B4:10:CF:B
Alias name: mozillacert58.pem
  SHA1: 8D:17:84:D5:37:F3:03:7D:EC:70:FE:57:8B:51:9A:99:E6:10:D7:B0
  SHA256:
 5E:DB:7A:C4:3B:82:A0:6A:87:61:E8:D7:BE:49:79:EB:F2:61:1F:7D:D7:9B:F9:1C:1C:6B:56:6A:21:9E:D7:6
Alias name: mozillacert59.pem
```

```
  SHA1: 36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
  SHA256:
 23:99:56:11:27:A5:71:25:DE:8C:EF:EA:61:0D:DF:2F:A0:78:B5:C8:06:7F:4E:82:82:90:BF:B8:60:E8:4B:3
Alias name: mozillacert6.pem
  SHA1: 27:96:BA:E6:3F:18:01:E2:77:26:1B:A0:D7:77:70:02:8F:20:EE:E4
  SHA256:
 C3:84:6B:F2:4B:9E:93:CA:64:27:4C:0E:C6:7C:1E:CC:5E:02:4F:FC:AC:D2:D7:40:19:35:0E:81:FE:54:6A:E
Alias name: mozillacert60.pem
  SHA1: 3B:C4:9F:48:F8:F3:73:A0:9C:1E:BD:F8:5B:B1:C3:65:C7:D8:11:B3
  SHA256:
 BF:0F:EE:FB:9E:3A:58:1A:D5:F9:E9:DB:75:89:98:57:43:D2:61:08:5C:4D:31:4F:6F:5D:72:59:AA:42:16:1
Alias name: mozillacert61.pem
  SHA1: E0:B4:32:2E:B2:F6:A5:68:B6:54:53:84:48:18:4A:50:36:87:43:84
  SHA256:
 03:95:0F:B4:9A:53:1F:3E:19:91:94:23:98:DF:A9:E0:EA:32:D7:BA:1C:DD:9B:C8:5D:B5:7E:D9:40:0B:43:4
Alias name: mozillacert62.pem
  SHA1: A1:DB:63:93:91:6F:17:E4:18:55:09:40:04:15:C7:02:40:B0:AE:6B
  SHA256:
 A4:B6:B3:99:6F:C2:F3:06:B3:FD:86:81:BD:63:41:3D:8C:50:09:CC:4F:A3:29:C2:CC:F0:E2:FA:1B:14:03:0
Alias name: mozillacert63.pem
  SHA1: 89:DF:74:FE:5C:F4:0F:4A:80:F9:E3:37:7D:54:DA:91:E1:01:31:8E
  SHA256:
 3C:5F:81:FE:A5:FA:B8:2C:64:BF:A2:EA:EC:AF:CD:E8:E0:77:FC:86:20:A7:CA:E5:37:16:3D:F3:6E:DB:F3:7
Alias name: mozillacert64.pem
  SHA1: 62:7F:8D:78:27:65:63:99:D2:7D:7F:90:44:C9:FE:B3:F3:3E:FA:9A
  SHA256:
 AB:70:36:36:5C:71:54:AA:29:C2:C2:9F:5D:41:91:16:3B:16:2A:22:25:01:13:57:D5:6D:07:FF:A7:BC:1F:7
Alias name: mozillacert65.pem
  SHA1: 69:BD:8C:F4:9C:D3:00:FB:59:2E:17:93:CA:55:6A:F3:EC:AA:35:FB
  SHA256:
 BC:23:F9:8A:31:3C:B9:2D:E3:BB:FC:3A:5A:9F:44:61:AC:39:49:4C:4A:E1:5A:9E:9D:F1:31:E9:9B:73:01:9
Alias name: mozillacert66.pem
  SHA1: DD:E1:D2:A9:01:80:2E:1D:87:5E:84:B3:80:7E:4B:B1:FD:99:41:34
  SHA256:
 E6:09:07:84:65:A4:19:78:0C:B6:AC:4C:1C:0B:FB:46:53:D9:D9:CC:6E:B3:94:6E:B7:F3:D6:99:97:BA:D5:9
Alias name: mozillacert67.pem
  SHA1: D6:9B:56:11:48:F0:1C:77:C5:45:78:C1:09:26:DF:5B:85:69:76:AD
  SHA256:
 CB:B5:22:D7:B7:F1:27:AD:6A:01:13:86:5B:DF:1C:D4:10:2E:7D:07:59:AF:63:5A:7C:F4:72:0D:C9:63:C5:3
Alias name: mozillacert68.pem
  SHA1: AE:C5:FB:3F:C8:E1:BF:C4:E5:4F:03:07:5A:9A:E8:00:B7:F7:B6:FA
  SHA256:
 04:04:80:28:BF:1F:28:64:D4:8F:9A:D4:D8:32:94:36:6A:82:88:56:55:3F:3B:14:30:3F:90:14:7F:5D:40:E
Alias name: mozillacert69.pem
```

```
  SHA1: 2F:78:3D:25:52:18:A7:4A:65:39:71:B5:2C:A2:9C:45:15:6F:E9:19
  SHA256:
 25:30:CC:8E:98:32:15:02:BA:D9:6F:9B:1F:BA:1B:09:9E:2D:29:9E:0F:45:48:BB:91:4F:36:3B:C0:D4:53:1
Alias name: mozillacert7.pem
  SHA1: AD:7E:1C:28:B0:64:EF:8F:60:03:40:20:14:C3:D0:E3:37:0E:B5:8A
  SHA256:
 14:65:FA:20:53:97:B8:76:FA:A6:F0:A9:95:8E:55:90:E4:0F:CC:7F:AA:4F:B7:C2:C8:67:75:21:FB:5F:B6:5
Alias name: mozillacert70.pem
  SHA1: 78:6A:74:AC:76:AB:14:7F:9C:6A:30:50:BA:9E:A8:7E:FE:9A:CE:3C
  SHA256:
 06:3E:4A:FA:C4:91:DF:D3:32:F3:08:9B:85:42:E9:46:17:D8:93:D7:FE:94:4E:10:A7:93:7E:E2:9D:96:93:C
Alias name: mozillacert71.pem
  SHA1: 4A:BD:EE:EC:95:0D:35:9C:89:AE:C7:52:A1:2C:5B:29:F6:D6:AA:0C
  SHA256:
 13:63:35:43:93:34:A7:69:80:16:A0:D3:24:DE:72:28:4E:07:9D:7B:52:20:BB:8F:BD:74:78:16:EE:BE:BA:C
Alias name: mozillacert72.pem
  SHA1: 47:BE:AB:C9:22:EA:E8:0E:78:78:34:62:A7:9F:45:C2:54:FD:E6:8B
  SHA256:
 45:14:0B:32:47:EB:9C:C8:C5:B4:F0:D7:B5:30:91:F7:32:92:08:9E:6E:5A:63:E2:74:9D:D3:AC:A9:19:8E:D
Alias name: mozillacert73.pem
  SHA1: B5:1C:06:7C:EE:2B:0C:3D:F8:55:AB:2D:92:F4:FE:39:D4:E7:0F:0E
  SHA256:
 2C:E1:CB:0B:F9:D2:F9:E1:02:99:3F:BE:21:51:52:C3:B2:DD:0C:AB:DE:1C:68:E5:31:9B:83:91:54:DB:B7:F
Alias name: mozillacert74.pem
  SHA1: 92:5A:8F:8D:2C:6D:04:E0:66:5F:59:6A:FF:22:D8:63:E8:25:6F:3F
  SHA256:
 56:8D:69:05:A2:C8:87:08:A4:B3:02:51:90:ED:CF:ED:B1:97:4A:60:6A:13:C6:E5:29:0F:CB:2A:E6:3E:DA:B
Alias name: mozillacert75.pem
  SHA1: D2:32:09:AD:23:D3:14:23:21:74:E4:0D:7F:9D:62:13:97:86:63:3A
  SHA256:
 08:29:7A:40:47:DB:A2:36:80:C7:31:DB:6E:31:76:53:CA:78:48:E1:BE:BD:3A:0B:01:79:A7:07:F9:2C:F1:7
Alias name: mozillacert76.pem
  SHA1: F9:B5:B6:32:45:5F:9C:BE:EC:57:5F:80:DC:E9:6E:2C:C7:B2:78:B7
  SHA256:
 03:76:AB:1D:54:C5:F9:80:3C:E4:B2:E2:01:A0:EE:7E:EF:7B:57:B6:36:E8:A9:3C:9B:8D:48:60:C9:6F:5F:A
Alias name: mozillacert77.pem
  SHA1: 13:2D:0D:45:53:4B:69:97:CD:B2:D5:C3:39:E2:55:76:60:9B:5C:C6
  SHA256:
 EB:04:CF:5E:B1:F3:9A:FA:76:2F:2B:B1:20:F2:96:CB:A5:20:C1:B9:7D:B1:58:95:65:B8:1C:B9:A1:7B:72:4
Alias name: mozillacert78.pem
  SHA1: 29:36:21:02:8B:20:ED:02:F5:66:C5:32:D1:D6:ED:90:9F:45:00:2F
  SHA256:
 0A:81:EC:5A:92:97:77:F1:45:90:4A:F3:8D:5D:50:9F:66:B5:E2:C5:8F:CD:B5:31:05:8B:0E:17:F3:F0:B4:1
Alias name: mozillacert79.pem
```

```
  SHA1: D8:A6:33:2C:E0:03:6F:B1:85:F6:63:4F:7D:6A:06:65:26:32:28:27
  SHA256:
 70:A7:3F:7F:37:6B:60:07:42:48:90:45:34:B1:14:82:D5:BF:0E:69:8E:CC:49:8D:F5:25:77:EB:F2:E9:3B:9
Alias name: mozillacert8.pem
  SHA1: 3E:2B:F7:F2:03:1B:96:F3:8C:E6:C4:D8:A8:5D:3E:2D:58:47:6A:0F
  SHA256:
 C7:66:A9:BE:F2:D4:07:1C:86:3A:31:AA:49:20:E8:13:B2:D1:98:60:8C:B7:B7:CF:E2:11:43:B8:36:DF:09:E
Alias name: mozillacert80.pem
  SHA1: B8:23:6B:00:2F:1D:16:86:53:01:55:6C:11:A4:37:CA:EB:FF:C3:BB
  SHA256:
 BD:71:FD:F6:DA:97:E4:CF:62:D1:64:7A:DD:25:81:B0:7D:79:AD:F8:39:7E:B4:EC:BA:9C:5E:84:88:82:14:2
Alias name: mozillacert81.pem
  SHA1: 07:E0:32:E0:20:B7:2C:3F:19:2F:06:28:A2:59:3A:19:A7:0F:06:9E
  SHA256:
 5C:58:46:8D:55:F5:8E:49:7E:74:39:82:D2:B5:00:10:B6:D1:65:37:4A:CF:83:A7:D4:A3:2D:B7:68:C4:40:8
Alias name: mozillacert82.pem
  SHA1: 2E:14:DA:EC:28:F0:FA:1E:8E:38:9A:4E:AB:EB:26:C0:0A:D3:83:C3
  SHA256:
 FC:BF:E2:88:62:06:F7:2B:27:59:3C:8B:07:02:97:E1:2D:76:9E:D1:0E:D7:93:07:05:A8:09:8E:FF:C1:4D:1
Alias name: mozillacert83.pem
  SHA1: A0:73:E5:C5:BD:43:61:0D:86:4C:21:13:0A:85:58:57:CC:9C:EA:46
  SHA256:
 8C:4E:DF:D0:43:48:F3:22:96:9E:7E:29:A4:CD:4D:CA:00:46:55:06:1C:16:E1:B0:76:42:2E:F3:42:AD:63:0
Alias name: mozillacert84.pem
  SHA1: D3:C0:63:F2:19:ED:07:3E:34:AD:5D:75:0B:32:76:29:FF:D5:9A:F2
  SHA256:
 79:3C:BF:45:59:B9:FD:E3:8A:B2:2D:F1:68:69:F6:98:81:AE:14:C4:B0:13:9A:C7:88:A7:8A:1A:FC:CA:02:F
Alias name: mozillacert85.pem
  SHA1: CF:9E:87:6D:D3:EB:FC:42:26:97:A3:B5:A3:7A:A0:76:A9:06:23:48
  SHA256:
 BF:D8:8F:E1:10:1C:41:AE:3E:80:1B:F8:BE:56:35:0E:E9:BA:D1:A6:B9:BD:51:5E:DC:5C:6D:5B:87:11:AC:4
Alias name: mozillacert86.pem
  SHA1: 74:2C:31:92:E6:07:E4:24:EB:45:49:54:2B:E1:BB:C5:3E:61:74:E2
  SHA256:
 E7:68:56:34:EF:AC:F6:9A:CE:93:9A:6B:25:5B:7B:4F:AB:EF:42:93:5B:50:A2:65:AC:B5:CB:60:27:E4:4E:7
Alias name: mozillacert87.pem
  SHA1: 5F:3B:8C:F2:F8:10:B3:7D:78:B4:CE:EC:19:19:C3:73:34:B9:C7:74
  SHA256:
 51:3B:2C:EC:B8:10:D4:CD:E5:DD:85:39:1A:DF:C6:C2:DD:60:D8:7B:B7:36:D2:B5:21:48:4A:A4:7A:0E:BE:F
Alias name: mozillacert88.pem
  SHA1: FE:45:65:9B:79:03:5B:98:A1:61:B5:51:2E:AC:DA:58:09:48:22:4D
  SHA256:
 BC:10:4F:15:A4:8B:E7:09:DC:A5:42:A7:E1:D4:B9:DF:6F:05:45:27:E8:02:EA:A9:2D:59:54:44:25:8A:FE:7
Alias name: mozillacert89.pem
```

```
  SHA1: C8:EC:8C:87:92:69:CB:4B:AB:39:E9:8D:7E:57:67:F3:14:95:73:9D
  SHA256:
 E3:89:36:0D:0F:DB:AE:B3:D2:50:58:4B:47:30:31:4E:22:2F:39:C1:56:A0:20:14:4E:8D:96:05:61:79:15:0
Alias name: mozillacert9.pem
  SHA1: F4:8B:11:BF:DE:AB:BE:94:54:20:71:E6:41:DE:6B:BE:88:2B:40:B9
  SHA256:
 76:00:29:5E:EF:E8:5B:9E:1F:D6:24:DB:76:06:2A:AA:AE:59:81:8A:54:D2:77:4C:D4:C0:B2:C0:11:31:E1:B
Alias name: mozillacert90.pem
  SHA1: F3:73:B3:87:06:5A:28:84:8A:F2:F3:4A:CE:19:2B:DD:C7:8E:9C:AC
  SHA256:
 55:92:60:84:EC:96:3A:64:B9:6E:2A:BE:01:CE:0B:A8:6A:64:FB:FE:BC:C7:AA:B5:AF:C1:55:B3:7F:D7:60:6
Alias name: mozillacert91.pem
  SHA1: 3B:C0:38:0B:33:C3:F6:A6:0C:86:15:22:93:D9:DF:F5:4B:81:C0:04
  SHA256:
 C1:B4:82:99:AB:A5:20:8F:E9:63:0A:CE:55:CA:68:A0:3E:DA:5A:51:9C:88:02:A0:D3:A6:73:BE:8F:8E:55:7
Alias name: mozillacert92.pem
  SHA1: A3:F1:33:3F:E2:42:BF:CF:C5:D1:4E:8F:39:42:98:40:68:10:D1:A0
  SHA256:
 E1:78:90:EE:09:A3:FB:F4:F4:8B:9C:41:4A:17:D6:37:B7:A5:06:47:E9:BC:75:23:22:72:7F:CC:17:42:A9:1
Alias name: mozillacert93.pem
  SHA1: 31:F1:FD:68:22:63:20:EE:C6:3B:3F:9D:EA:4A:3E:53:7C:7C:39:17
  SHA256:
 C7:BA:65:67:DE:93:A7:98:AE:1F:AA:79:1E:71:2D:37:8F:AE:1F:93:C4:39:7F:EA:44:1B:B7:CB:E6:FD:59:9
Alias name: mozillacert94.pem
  SHA1: 49:0A:75:74:DE:87:0A:47:FE:58:EE:F6:C7:6B:EB:C6:0B:12:40:99
  SHA256:
 9A:11:40:25:19:7C:5B:B9:5D:94:E6:3D:55:CD:43:79:08:47:B6:46:B2:3C:DF:11:AD:A4:A0:0E:FF:15:FB:4
Alias name: mozillacert95.pem
  SHA1: DA:FA:F7:FA:66:84:EC:06:8F:14:50:BD:C7:C2:81:A5:BC:A9:64:57
  SHA256:
 ED:F7:EB:BC:A2:7A:2A:38:4D:38:7B:7D:40:10:C6:66:E2:ED:B4:84:3E:4C:29:B4:AE:1D:5B:93:32:E6:B2:4
Alias name: mozillacert96.pem
  SHA1: 55:A6:72:3E:CB:F2:EC:CD:C3:23:74:70:19:9D:2A:BE:11:E3:81:D1
  SHA256:
 FD:73:DA:D3:1C:64:4F:F1:B4:3B:EF:0C:CD:DA:96:71:0B:9C:D9:87:5E:CA:7E:31:70:7A:F3:E9:6D:52:2B:B
Alias name: mozillacert97.pem
  SHA1: 85:37:1C:A6:E5:50:14:3D:CE:28:03:47:1B:DE:3A:09:E8:F8:77:0F
  SHA256:
 83:CE:3C:12:29:68:8A:59:3D:48:5F:81:97:3C:0F:91:95:43:1E:DA:37:CC:5E:36:43:0E:79:C7:A8:88:63:8
Alias name: mozillacert98.pem
  SHA1: C9:A8:B9:E7:55:80:5E:58:E3:53:77:A7:25:EB:AF:C3:7B:27:CC:D7
  SHA256:
 3E:84:BA:43:42:90:85:16:E7:75:73:C0:99:2F:09:79:CA:08:4E:46:85:68:1F:F1:95:CC:BA:8A:22:9B:8A:7
Alias name: mozillacert99.pem
```

```
  SHA1: F1:7F:6F:B6:31:DC:99:E3:A3:C8:7F:FE:1C:F1:81:10:88:D9:60:33
  SHA256:
 97:8C:D9:66:F2:FA:A0:7B:A7:AA:95:00:D9:C0:2E:9D:77:F2:CD:AD:A6:AD:6B:A7:4A:F4:B9:1C:66:59:3C:5
Alias name: netlockaranyclassgoldfotanusitvany
  SHA1: 06:08:3F:59:3F:15:A1:04:A0:69:A4:6B:A9:03:D0:06:B7:97:09:91
  SHA256:
 6C:61:DA:C3:A2:DE:F0:31:50:6B:E0:36:D2:A6:FE:40:19:94:FB:D1:3D:F9:C8:D4:66:59:92:74:C4:46:EC:9
Alias name: networksolutionscertificateauthority
  SHA1: 74:F8:A3:C3:EF:E7:B3:90:06:4B:83:90:3C:21:64:60:20:E5:DF:CE
  SHA256:
 15:F0:BA:00:A3:AC:7A:F3:AC:88:4C:07:2B:10:11:A0:77:BD:77:C0:97:F4:01:64:B2:F8:59:8A:BD:83:86:0
Alias name: oistewisekeyglobalrootgaca
  SHA1: 59:22:A1:E1:5A:EA:16:35:21:F8:98:39:6A:46:46:B0:44:1B:0F:A9
  SHA256:
 41:C9:23:86:6A:B4:CA:D6:B7:AD:57:80:81:58:2E:02:07:97:A6:CB:DF:4F:FF:78:CE:83:96:B3:89:37:D7:F
Alias name: oistewisekeyglobalrootgbca
  SHA1: 0F:F9:40:76:18:D3:D7:6A:4B:98:F0:A8:35:9E:0C:FD:27:AC:CC:ED
  SHA256:
 6B:9C:08:E8:6E:B0:F7:67:CF:AD:65:CD:98:B6:21:49:E5:49:4A:67:F5:84:5E:7B:D1:ED:01:9F:27:B8:6B:D
Alias name: oistewisekeyglobalrootgcca
  SHA1: E0:11:84:5E:34:DE:BE:88:81:B9:9C:F6:16:26:D1:96:1F:C3:B9:31
  SHA256:
 85:60:F9:1C:36:24:DA:BA:95:70:B5:FE:A0:DB:E3:6F:F1:1A:83:23:BE:94:86:85:4F:B3:F3:4A:55:71:19:8
Alias name: quovadisrootca
  SHA1: DE:3F:40:BD:50:93:D3:9B:6C:60:F6:DA:BC:07:62:01:00:89:76:C9
  SHA256:
 A4:5E:DE:3B:BB:F0:9C:8A:E1:5C:72:EF:C0:72:68:D6:93:A2:1C:99:6F:D5:1E:67:CA:07:94:60:FD:6D:88:7
Alias name: quovadisrootca1g3
  SHA1: 1B:8E:EA:57:96:29:1A:C9:39:EA:B8:0A:81:1A:73:73:C0:93:79:67
  SHA256:
 8A:86:6F:D1:B2:76:B5:7E:57:8E:92:1C:65:82:8A:2B:ED:58:E9:F2:F2:88:05:41:34:B7:F1:F4:BF:C9:CC:7
Alias name: quovadisrootca2
  SHA1: CA:3A:FB:CF:12:40:36:4B:44:B2:16:20:88:80:48:39:19:93:7C:F7
  SHA256:
 85:A0:DD:7D:D7:20:AD:B7:FF:05:F8:3D:54:2B:20:9D:C7:FF:45:28:F7:D6:77:B1:83:89:FE:A5:E5:C4:9E:8
Alias name: quovadisrootca2g3
  SHA1: 09:3C:61:F3:8B:8B:DC:7D:55:DF:75:38:02:05:00:E1:25:F5:C8:36
  SHA256:
 8F:E4:FB:0A:F9:3A:4D:0D:67:DB:0B:EB:B2:3E:37:C7:1B:F3:25:DC:BC:DD:24:0E:A0:4D:AF:58:B4:7E:18:4
Alias name: quovadisrootca3
  SHA1: 1F:49:14:F7:D8:74:95:1D:DD:AE:02:C0:BE:FD:3A:2D:82:75:51:85
  SHA256:
 18:F1:FC:7F:20:5D:F8:AD:DD:EB:7F:E0:07:DD:57:E3:AF:37:5A:9C:4D:8D:73:54:6B:F4:F1:FE:D1:E1:8D:3
Alias name: quovadisrootca3g3
```

```
  SHA1: 48:12:BD:92:3C:A8:C4:39:06:E7:30:6D:27:96:E6:A4:CF:22:2E:7D
  SHA256:
 88:EF:81:DE:20:2E:B0:18:45:2E:43:F8:64:72:5C:EA:5F:BD:1F:C2:D9:D2:05:73:07:09:C5:D8:B8:69:0F:4
Alias name: secomevrootca1
  SHA1: FE:B8:C4:32:DC:F9:76:9A:CE:AE:3D:D8:90:8F:FD:28:86:65:64:7D
  SHA256:
 A2:2D:BA:68:1E:97:37:6E:2D:39:7D:72:8A:AE:3A:9B:62:96:B9:FD:BA:60:BC:2E:11:F6:47:F2:C6:75:FB:3
Alias name: secomscrootca1
  SHA1: 36:B1:2B:49:F9:81:9E:D7:4C:9E:BC:38:0F:C6:56:8F:5D:AC:B2:F7
  SHA256:
 E7:5E:72:ED:9F:56:0E:EC:6E:B4:80:00:73:A4:3F:C3:AD:19:19:5A:39:22:82:01:78:95:97:4A:99:02:6B:6
Alias name: secomscrootca2
  SHA1: 5F:3B:8C:F2:F8:10:B3:7D:78:B4:CE:EC:19:19:C3:73:34:B9:C7:74
  SHA256:
 51:3B:2C:EC:B8:10:D4:CD:E5:DD:85:39:1A:DF:C6:C2:DD:60:D8:7B:B7:36:D2:B5:21:48:4A:A4:7A:0E:BE:F
Alias name: secomvalicertclass1ca
  SHA1: E5:DF:74:3C:B6:01:C4:9B:98:43:DC:AB:8C:E8:6A:81:10:9F:E4:8E
  SHA256:
 F4:C1:49:55:1A:30:13:A3:5B:C7:BF:FE:17:A7:F3:44:9B:C1:AB:5B:5A:0A:E7:4B:06:C2:3B:90:00:4C:01:0
Alias name: secureglobalca
  SHA1: 3A:44:73:5A:E5:81:90:1F:24:86:61:46:1E:3B:9C:C4:5F:F5:3A:1B
  SHA256:
 42:00:F5:04:3A:C8:59:0E:BB:52:7D:20:9E:D1:50:30:29:FB:CB:D4:1C:A1:B5:06:EC:27:F1:5A:DE:7D:AC:6
Alias name: securesignrootca11
  SHA1: 3B:C4:9F:48:F8:F3:73:A0:9C:1E:BD:F8:5B:B1:C3:65:C7:D8:11:B3
  SHA256:
 BF:0F:EE:FB:9E:3A:58:1A:D5:F9:E9:DB:75:89:98:57:43:D2:61:08:5C:4D:31:4F:6F:5D:72:59:AA:42:16:1
Alias name: securetrustca
  SHA1: 87:82:C6:C3:04:35:3B:CF:D2:96:92:D2:59:3E:7D:44:D9:34:FF:11
  SHA256:
 F1:C1:B5:0A:E5:A2:0D:D8:03:0E:C9:F6:BC:24:82:3D:D3:67:B5:25:57:59:B4:E7:1B:61:FC:E9:F7:37:5D:7
Alias name: securitycommunicationrootca
  SHA1: 36:B1:2B:49:F9:81:9E:D7:4C:9E:BC:38:0F:C6:56:8F:5D:AC:B2:F7
  SHA256:
 E7:5E:72:ED:9F:56:0E:EC:6E:B4:80:00:73:A4:3F:C3:AD:19:19:5A:39:22:82:01:78:95:97:4A:99:02:6B:6
Alias name: securitycommunicationrootca2
  SHA1: 5F:3B:8C:F2:F8:10:B3:7D:78:B4:CE:EC:19:19:C3:73:34:B9:C7:74
  SHA256:
 51:3B:2C:EC:B8:10:D4:CD:E5:DD:85:39:1A:DF:C6:C2:DD:60:D8:7B:B7:36:D2:B5:21:48:4A:A4:7A:0E:BE:F
Alias name: soneraclass1ca
  SHA1: 07:47:22:01:99:CE:74:B9:7C:B0:3D:79:B2:64:A2:C8:55:E9:33:FF
  SHA256:
 CD:80:82:84:CF:74:6F:F2:FD:6E:B5:8A:A1:D5:9C:4A:D4:B3:CA:56:FD:C6:27:4A:89:26:A7:83:5F:32:31:3
Alias name: soneraclass2ca
```

```
  SHA1: 37:F7:6D:E6:07:7C:90:C5:B1:3E:93:1A:B7:41:10:B4:F2:E4:9A:27
  SHA256:
 79:08:B4:03:14:C1:38:10:0B:51:8D:07:35:80:7F:FB:FC:F8:51:8A:00:95:33:71:05:BA:38:6B:15:3D:D9:2
Alias name: soneraclass2rootca
  SHA1: 37:F7:6D:E6:07:7C:90:C5:B1:3E:93:1A:B7:41:10:B4:F2:E4:9A:27
  SHA256:
 79:08:B4:03:14:C1:38:10:0B:51:8D:07:35:80:7F:FB:FC:F8:51:8A:00:95:33:71:05:BA:38:6B:15:3D:D9:2
Alias name: sslcomevrootcertificationauthorityecc
  SHA1: 4C:DD:51:A3:D1:F5:20:32:14:B0:C6:C5:32:23:03:91:C7:46:42:6D
  SHA256:
 22:A2:C1:F7:BD:ED:70:4C:C1:E7:01:B5:F4:08:C3:10:88:0F:E9:56:B5:DE:2A:4A:44:F9:9C:87:3A:25:A7:C
Alias name: sslcomevrootcertificationauthorityrsar2
  SHA1: 74:3A:F0:52:9B:D0:32:A0:F4:4A:83:CD:D4:BA:A9:7B:7C:2E:C4:9A
  SHA256:
 2E:7B:F1:6C:C2:24:85:A7:BB:E2:AA:86:96:75:07:61:B0:AE:39:BE:3B:2F:E9:D0:CC:6D:4E:F7:34:91:42:5
Alias name: sslcomrootcertificationauthorityecc
  SHA1: C3:19:7C:39:24:E6:54:AF:1B:C4:AB:20:95:7A:E2:C3:0E:13:02:6A
  SHA256:
 34:17:BB:06:CC:60:07:DA:1B:96:1C:92:0B:8A:B4:CE:3F:AD:82:0E:4A:A3:0B:9A:CB:C4:A7:4E:BD:CE:BC:6
Alias name: sslcomrootcertificationauthorityrsa
  SHA1: B7:AB:33:08:D1:EA:44:77:BA:14:80:12:5A:6F:BD:A9:36:49:0C:BB
  SHA256:
 85:66:6A:56:2E:E0:BE:5C:E9:25:C1:D8:89:0A:6F:76:A8:7E:C1:6D:4D:7D:5F:29:EA:74:19:CF:20:12:3B:6
Alias name: staatdernederlandenevrootca
  SHA1: 76:E2:7E:C1:4F:DB:82:C1:C0:A6:75:B5:05:BE:3D:29:B4:ED:DB:BB
  SHA256:
 4D:24:91:41:4C:FE:95:67:46:EC:4C:EF:A6:CF:6F:72:E2:8A:13:29:43:2F:9D:8A:90:7A:C4:CB:5D:AD:C1:5
Alias name: staatdernederlandenrootcag3
  SHA1: D8:EB:6B:41:51:92:59:E0:F3:E7:85:00:C0:3D:B6:88:97:C9:EE:FC
  SHA256:
 3C:4F:B0:B9:5A:B8:B3:00:32:F4:32:B8:6F:53:5F:E1:72:C1:85:D0:FD:39:86:58:37:CF:36:18:7F:A6:F4:2
Alias name: starfieldclass2ca
  SHA1: AD:7E:1C:28:B0:64:EF:8F:60:03:40:20:14:C3:D0:E3:37:0E:B5:8A
  SHA256:
 14:65:FA:20:53:97:B8:76:FA:A6:F0:A9:95:8E:55:90:E4:0F:CC:7F:AA:4F:B7:C2:C8:67:75:21:FB:5F:B6:5
Alias name: starfieldrootcertificateauthorityg2
  SHA1: B5:1C:06:7C:EE:2B:0C:3D:F8:55:AB:2D:92:F4:FE:39:D4:E7:0F:0E
  SHA256:
 2C:E1:CB:0B:F9:D2:F9:E1:02:99:3F:BE:21:51:52:C3:B2:DD:0C:AB:DE:1C:68:E5:31:9B:83:91:54:DB:B7:F
Alias name: starfieldrootg2ca
  SHA1: B5:1C:06:7C:EE:2B:0C:3D:F8:55:AB:2D:92:F4:FE:39:D4:E7:0F:0E
  SHA256:
 2C:E1:CB:0B:F9:D2:F9:E1:02:99:3F:BE:21:51:52:C3:B2:DD:0C:AB:DE:1C:68:E5:31:9B:83:91:54:DB:B7:F
Alias name: starfieldservicesrootcertificateauthorityg2
```

```
  SHA1: 92:5A:8F:8D:2C:6D:04:E0:66:5F:59:6A:FF:22:D8:63:E8:25:6F:3F
  SHA256:
 56:8D:69:05:A2:C8:87:08:A4:B3:02:51:90:ED:CF:ED:B1:97:4A:60:6A:13:C6:E5:29:0F:CB:2A:E6:3E:DA:B
Alias name: starfieldservicesrootg2ca
  SHA1: 92:5A:8F:8D:2C:6D:04:E0:66:5F:59:6A:FF:22:D8:63:E8:25:6F:3F
  SHA256:
 56:8D:69:05:A2:C8:87:08:A4:B3:02:51:90:ED:CF:ED:B1:97:4A:60:6A:13:C6:E5:29:0F:CB:2A:E6:3E:DA:B
Alias name: swisssigngoldcag2
  SHA1: D8:C5:38:8A:B7:30:1B:1B:6E:D4:7A:E6:45:25:3A:6F:9F:1A:27:61
  SHA256:
 62:DD:0B:E9:B9:F5:0A:16:3E:A0:F8:E7:5C:05:3B:1E:CA:57:EA:55:C8:68:8F:64:7C:68:81:F2:C8:35:7B:9
Alias name: swisssigngoldg2ca
  SHA1: D8:C5:38:8A:B7:30:1B:1B:6E:D4:7A:E6:45:25:3A:6F:9F:1A:27:61
  SHA256:
 62:DD:0B:E9:B9:F5:0A:16:3E:A0:F8:E7:5C:05:3B:1E:CA:57:EA:55:C8:68:8F:64:7C:68:81:F2:C8:35:7B:9
Alias name: swisssignplatinumg2ca
  SHA1: 56:E0:FA:C0:3B:8F:18:23:55:18:E5:D3:11:CA:E8:C2:43:31:AB:66
  SHA256:
 3B:22:2E:56:67:11:E9:92:30:0D:C0:B1:5A:B9:47:3D:AF:DE:F8:C8:4D:0C:EF:7D:33:17:B4:C1:82:1D:14:3
Alias name: swisssignsilvercag2
  SHA1: 9B:AA:E5:9F:56:EE:21:CB:43:5A:BE:25:93:DF:A7:F0:40:D1:1D:CB
  SHA256:
 BE:6C:4D:A2:BB:B9:BA:59:B6:F3:93:97:68:37:42:46:C3:C0:05:99:3F:A9:8F:02:0D:1D:ED:BE:D4:8A:81:D
Alias name: swisssignsilverg2ca
  SHA1: 9B:AA:E5:9F:56:EE:21:CB:43:5A:BE:25:93:DF:A7:F0:40:D1:1D:CB
  SHA256:
 BE:6C:4D:A2:BB:B9:BA:59:B6:F3:93:97:68:37:42:46:C3:C0:05:99:3F:A9:8F:02:0D:1D:ED:BE:D4:8A:81:D
Alias name: szafirrootca2
  SHA1: E2:52:FA:95:3F:ED:DB:24:60:BD:6E:28:F3:9C:CC:CF:5E:B3:3F:DE
  SHA256:
 A1:33:9D:33:28:1A:0B:56:E5:57:D3:D3:2B:1C:E7:F9:36:7E:B0:94:BD:5F:A7:2A:7E:50:04:C8:DE:D7:CA:F
Alias name: teliasonerarootcav1
  SHA1: 43:13:BB:96:F1:D5:86:9B:C1:4E:6A:92:F6:CF:F6:34:69:87:82:37
  SHA256:
 DD:69:36:FE:21:F8:F0:77:C1:23:A1:A5:21:C1:22:24:F7:22:55:B7:3E:03:A7:26:06:93:E8:A2:4B:0F:A3:8
Alias name: thawtepersonalfreemailca
  SHA1: E6:18:83:AE:84:CA:C1:C1:CD:52:AD:E8:E9:25:2B:45:A6:4F:B7:E2
  SHA256:
 5B:38:BD:12:9E:83:D5:A0:CA:D2:39:21:08:94:90:D5:0D:4A:AE:37:04:28:F8:DD:FF:FF:FA:4C:15:64:E1:8
Alias name: thawtepremiumserverca
  SHA1: E0:AB:05:94:20:72:54:93:05:60:62:02:36:70:F7:CD:2E:FC:66:66
  SHA256:
 3F:9F:27:D5:83:20:4B:9E:09:C8:A3:D2:06:6C:4B:57:D3:A2:47:9C:36:93:65:08:80:50:56:98:10:5D:BC:E
Alias name: thawteprimaryrootca
```

```
  SHA1: 91:C6:D6:EE:3E:8A:C8:63:84:E5:48:C2:99:29:5C:75:6C:81:7B:81
  SHA256:
 8D:72:2F:81:A9:C1:13:C0:79:1D:F1:36:A2:96:6D:B2:6C:95:0A:97:1D:B4:6B:41:99:F4:EA:54:B7:8B:FB:9
Alias name: thawteprimaryrootcag2
  SHA1: AA:DB:BC:22:23:8F:C4:01:A1:27:BB:38:DD:F4:1D:DB:08:9E:F0:12
  SHA256:
 A4:31:0D:50:AF:18:A6:44:71:90:37:2A:86:AF:AF:8B:95:1F:FB:43:1D:83:7F:1E:56:88:B4:59:71:ED:15:5
Alias name: thawteprimaryrootcag3
  SHA1: F1:8B:53:8D:1B:E9:03:B6:A6:F0:56:43:5B:17:15:89:CA:F3:6B:F2
  SHA256:
 4B:03:F4:58:07:AD:70:F2:1B:FC:2C:AE:71:C9:FD:E4:60:4C:06:4C:F5:FF:B6:86:BA:E5:DB:AA:D7:FD:D3:4
Alias name: thawteserverca
  SHA1: 9F:AD:91:A6:CE:6A:C6:C5:00:47:C4:4E:C9:D4:A5:0D:92:D8:49:79
  SHA256:
 87:C6:78:BF:B8:B2:5F:38:F7:E9:7B:33:69:56:BB:CF:14:4B:BA:CA:A5:36:47:E6:1A:23:25:BC:10:55:31:6
Alias name: trustcenterclass2caii
  SHA1: AE:50:83:ED:7C:F4:5C:BC:8F:61:C6:21:FE:68:5D:79:42:21:15:6E
  SHA256:
 E6:B8:F8:76:64:85:F8:07:AE:7F:8D:AC:16:70:46:1F:07:C0:A1:3E:EF:3A:1F:F7:17:53:8D:7A:BA:D3:91:B
Alias name: trustcenterclass4caii
  SHA1: A6:9A:91:FD:05:7F:13:6A:42:63:0B:B1:76:0D:2D:51:12:0C:16:50
  SHA256:
 32:66:96:7E:59:CD:68:00:8D:9D:D3:20:81:11:85:C7:04:20:5E:8D:95:FD:D8:4F:1C:7B:31:1E:67:04:FC:3
Alias name: trustcenteruniversalcai
  SHA1: 6B:2F:34:AD:89:58:BE:62:FD:B0:6B:5C:CE:BB:9D:D9:4F:4E:39:F3
  SHA256:
 EB:F3:C0:2A:87:89:B1:FB:7D:51:19:95:D6:63:B7:29:06:D9:13:CE:0D:5E:10:56:8A:8A:77:E2:58:61:67:E
Alias name: trustcoreca1
  SHA1: 58:D1:DF:95:95:67:6B:63:C0:F0:5B:1C:17:4D:8B:84:0B:C8:78:BD
  SHA256:
 5A:88:5D:B1:9C:01:D9:12:C5:75:93:88:93:8C:AF:BB:DF:03:1A:B2:D4:8E:91:EE:15:58:9B:42:97:1D:03:9
Alias name: trustcorrootcertca1
  SHA1: FF:BD:CD:E7:82:C8:43:5E:3C:6F:26:86:5C:CA:A8:3A:45:5B:C3:0A
  SHA256:
 D4:0E:9C:86:CD:8F:E4:68:C1:77:69:59:F4:9E:A7:74:FA:54:86:84:B6:C4:06:F3:90:92:61:F4:DC:E2:57:5
Alias name: trustcorrootcertca2
  SHA1: B8:BE:6D:CB:56:F1:55:B9:63:D4:12:CA:4E:06:34:C7:94:B2:1C:C0
  SHA256:
 07:53:E9:40:37:8C:1B:D5:E3:83:6E:39:5D:AE:A5:CB:83:9E:50:46:F1:BD:0E:AE:19:51:CF:10:FE:C7:C9:6
Alias name: trustisfpsrootca
  SHA1: 3B:C0:38:0B:33:C3:F6:A6:0C:86:15:22:93:D9:DF:F5:4B:81:C0:04
  SHA256:
 C1:B4:82:99:AB:A5:20:8F:E9:63:0A:CE:55:CA:68:A0:3E:DA:5A:51:9C:88:02:A0:D3:A6:73:BE:8F:8E:55:7
Alias name: ttelesecglobalrootclass2
```

```
  SHA1: 59:0D:2D:7D:88:4F:40:2E:61:7E:A5:62:32:17:65:CF:17:D8:94:E9
  SHA256:
 91:E2:F5:78:8D:58:10:EB:A7:BA:58:73:7D:E1:54:8A:8E:CA:CD:01:45:98:BC:0B:14:3E:04:1B:17:05:25:5
Alias name: ttelesecglobalrootclass2ca
  SHA1: 59:0D:2D:7D:88:4F:40:2E:61:7E:A5:62:32:17:65:CF:17:D8:94:E9
  SHA256:
 91:E2:F5:78:8D:58:10:EB:A7:BA:58:73:7D:E1:54:8A:8E:CA:CD:01:45:98:BC:0B:14:3E:04:1B:17:05:25:5
Alias name: ttelesecglobalrootclass3
  SHA1: 55:A6:72:3E:CB:F2:EC:CD:C3:23:74:70:19:9D:2A:BE:11:E3:81:D1
  SHA256:
 FD:73:DA:D3:1C:64:4F:F1:B4:3B:EF:0C:CD:DA:96:71:0B:9C:D9:87:5E:CA:7E:31:70:7A:F3:E9:6D:52:2B:B
Alias name: ttelesecglobalrootclass3ca
  SHA1: 55:A6:72:3E:CB:F2:EC:CD:C3:23:74:70:19:9D:2A:BE:11:E3:81:D1
  SHA256:
 FD:73:DA:D3:1C:64:4F:F1:B4:3B:EF:0C:CD:DA:96:71:0B:9C:D9:87:5E:CA:7E:31:70:7A:F3:E9:6D:52:2B:B
Alias name: tubitakkamusmsslkoksertifikasisurum1
  SHA1: 31:43:64:9B:EC:CE:27:EC:ED:3A:3F:0B:8F:0D:E4:E8:91:DD:EE:CA
  SHA256:
 46:ED:C3:68:90:46:D5:3A:45:3F:B3:10:4A:B8:0D:CA:EC:65:8B:26:60:EA:16:29:DD:7E:86:79:90:64:87:1
Alias name: twcaglobalrootca
  SHA1: 9C:BB:48:53:F6:A4:F6:D3:52:A4:E8:32:52:55:60:13:F5:AD:AF:65
  SHA256:
 59:76:90:07:F7:68:5D:0F:CD:50:87:2F:9F:95:D5:75:5A:5B:2B:45:7D:81:F3:69:2B:61:0A:98:67:2F:0E:1
Alias name: twcarootcertificationauthority
  SHA1: CF:9E:87:6D:D3:EB:FC:42:26:97:A3:B5:A3:7A:A0:76:A9:06:23:48
  SHA256:
 BF:D8:8F:E1:10:1C:41:AE:3E:80:1B:F8:BE:56:35:0E:E9:BA:D1:A6:B9:BD:51:5E:DC:5C:6D:5B:87:11:AC:4
Alias name: ucaextendedvalidationroot
  SHA1: A3:A1:B0:6F:24:61:23:4A:E3:36:A5:C2:37:FC:A6:FF:DD:F0:D7:3A
  SHA256:
 D4:3A:F9:B3:54:73:75:5C:96:84:FC:06:D7:D8:CB:70:EE:5C:28:E7:73:FB:29:4E:B4:1E:E7:17:22:92:4D:2
Alias name: ucaglobalg2root
  SHA1: 28:F9:78:16:19:7A:FF:18:25:18:AA:44:FE:C1:A0:CE:5C:B6:4C:8A
  SHA256:
 9B:EA:11:C9:76:FE:01:47:64:C1:BE:56:A6:F9:14:B5:A5:60:31:7A:BD:99:88:39:33:82:E5:16:1A:A0:49:3
Alias name: usertrustecc
  SHA1: D1:CB:CA:5D:B2:D5:2A:7F:69:3B:67:4D:E5:F0:5A:1D:0C:95:7D:F0
  SHA256:
 4F:F4:60:D5:4B:9C:86:DA:BF:BC:FC:57:12:E0:40:0D:2B:ED:3F:BC:4D:4F:BD:AA:86:E0:6A:DC:D2:A9:AD:7
Alias name: usertrustecccertificationauthority
  SHA1: D1:CB:CA:5D:B2:D5:2A:7F:69:3B:67:4D:E5:F0:5A:1D:0C:95:7D:F0
  SHA256:
 4F:F4:60:D5:4B:9C:86:DA:BF:BC:FC:57:12:E0:40:0D:2B:ED:3F:BC:4D:4F:BD:AA:86:E0:6A:DC:D2:A9:AD:7
Alias name: usertrustrsa
```

```
  SHA1: 2B:8F:1B:57:33:0D:BB:A2:D0:7A:6C:51:F7:0E:E9:0D:DA:B9:AD:8E
  SHA256:
 E7:93:C9:B0:2F:D8:AA:13:E2:1C:31:22:8A:CC:B0:81:19:64:3B:74:9C:89:89:64:B1:74:6D:46:C3:D4:CB:D
Alias name: usertrustrsacertificationauthority
  SHA1: 2B:8F:1B:57:33:0D:BB:A2:D0:7A:6C:51:F7:0E:E9:0D:DA:B9:AD:8E
  SHA256:
 E7:93:C9:B0:2F:D8:AA:13:E2:1C:31:22:8A:CC:B0:81:19:64:3B:74:9C:89:89:64:B1:74:6D:46:C3:D4:CB:D
Alias name: utndatacorpsgcca
  SHA1: 58:11:9F:0E:12:82:87:EA:50:FD:D9:87:45:6F:4F:78:DC:FA:D6:D4
  SHA256:
 85:FB:2F:91:DD:12:27:5A:01:45:B6:36:53:4F:84:02:4A:D6:8B:69:B8:EE:88:68:4F:F7:11:37:58:05:B3:4
Alias name: utnuserfirstclientauthemailca
  SHA1: B1:72:B1:A5:6D:95:F9:1F:E5:02:87:E1:4D:37:EA:6A:44:63:76:8A
  SHA256:
 43:F2:57:41:2D:44:0D:62:74:76:97:4F:87:7D:A8:F1:FC:24:44:56:5A:36:7A:E6:0E:DD:C2:7A:41:25:31:A
Alias name: utnuserfirsthardwareca
  SHA1: 04:83:ED:33:99:AC:36:08:05:87:22:ED:BC:5E:46:00:E3:BE:F9:D7
  SHA256:
 6E:A5:47:41:D0:04:66:7E:ED:1B:48:16:63:4A:A3:A7:9E:6E:4B:96:95:0F:82:79:DA:FC:8D:9B:D8:81:21:3
Alias name: utnuserfirstobjectca
  SHA1: E1:2D:FB:4B:41:D7:D9:C3:2B:30:51:4B:AC:1D:81:D8:38:5E:2D:46
  SHA256:
 6F:FF:78:E4:00:A7:0C:11:01:1C:D8:59:77:C4:59:FB:5A:F9:6A:3D:F0:54:08:20:D0:F4:B8:60:78:75:E5:8
Alias name: valicertclass2ca
  SHA1: 31:7A:2A:D0:7F:2B:33:5E:F5:A1:C3:4E:4B:57:E8:B7:D8:F1:FC:A6
  SHA256:
 58:D0:17:27:9C:D4:DC:63:AB:DD:B1:96:A6:C9:90:6C:30:C4:E0:87:83:EA:E8:C1:60:99:54:D6:93:55:59:6
Alias name: verisignc1g1.pem
  SHA1: 90:AE:A2:69:85:FF:14:80:4C:43:49:52:EC:E9:60:84:77:AF:55:6F
  SHA256:
 D1:7C:D8:EC:D5:86:B7:12:23:8A:48:2C:E4:6F:A5:29:39:70:74:2F:27:6D:8A:B6:A9:E4:6E:E0:28:8F:33:5
Alias name: verisignc1g2.pem
  SHA1: 27:3E:E1:24:57:FD:C4:F9:0C:55:E8:2B:56:16:7F:62:F5:32:E5:47
  SHA256:
 34:1D:E9:8B:13:92:AB:F7:F4:AB:90:A9:60:CF:25:D4:BD:6E:C6:5B:9A:51:CE:6E:D0:67:D0:0E:C7:CE:9B:7
Alias name: verisignc1g3.pem
  SHA1: 20:42:85:DC:F7:EB:76:41:95:57:8E:13:6B:D4:B7:D1:E9:8E:46:A5
  SHA256:
 CB:B5:AF:18:5E:94:2A:24:02:F9:EA:CB:C0:ED:5B:B8:76:EE:A3:C1:22:36:23:D0:04:47:E4:F3:BA:55:4B:6
Alias name: verisignc1g6.pem
  SHA1: 51:7F:61:1E:29:91:6B:53:82:FB:72:E7:44:D9:8D:C3:CC:53:6D:64
  SHA256:
 9D:19:0B:2E:31:45:66:68:5B:E8:A8:89:E2:7A:A8:C7:D7:AE:1D:8A:AD:DB:A3:C1:EC:F9:D2:48:63:CD:34:B
Alias name: verisignc2g1.pem
```

```
  SHA1: 67:82:AA:E0:ED:EE:E2:1A:58:39:D3:C0:CD:14:68:0A:4F:60:14:2A
  SHA256:
 BD:46:9F:F4:5F:AA:E7:C5:4C:CB:D6:9D:3F:3B:00:22:55:D9:B0:6B:10:B1:D0:FA:38:8B:F9:6B:91:8B:2C:E
Alias name: verisignc2g2.pem
  SHA1: B3:EA:C4:47:76:C9:C8:1C:EA:F2:9D:95:B6:CC:A0:08:1B:67:EC:9D
  SHA256:
 3A:43:E2:20:FE:7F:3E:A9:65:3D:1E:21:74:2E:AC:2B:75:C2:0F:D8:98:03:05:BC:50:2C:AF:8C:2D:9B:41:A
Alias name: verisignc2g3.pem
  SHA1: 61:EF:43:D7:7F:CA:D4:61:51:BC:98:E0:C3:59:12:AF:9F:EB:63:11
  SHA256:
 92:A9:D9:83:3F:E1:94:4D:B3:66:E8:BF:AE:7A:95:B6:48:0C:2D:6C:6C:2A:1B:E6:5D:42:36:B6:08:FC:A1:B
Alias name: verisignc2g6.pem
  SHA1: 40:B3:31:A0:E9:BF:E8:55:BC:39:93:CA:70:4F:4E:C2:51:D4:1D:8F
  SHA256:
 CB:62:7D:18:B5:8A:D5:6D:DE:33:1A:30:45:6B:C6:5C:60:1A:4E:9B:18:DE:DC:EA:08:E7:DA:AA:07:81:5F:F
Alias name: verisignc3g1.pem
  SHA1: A1:DB:63:93:91:6F:17:E4:18:55:09:40:04:15:C7:02:40:B0:AE:6B
  SHA256:
 A4:B6:B3:99:6F:C2:F3:06:B3:FD:86:81:BD:63:41:3D:8C:50:09:CC:4F:A3:29:C2:CC:F0:E2:FA:1B:14:03:0
Alias name: verisignc3g2.pem
  SHA1: 85:37:1C:A6:E5:50:14:3D:CE:28:03:47:1B:DE:3A:09:E8:F8:77:0F
  SHA256:
 83:CE:3C:12:29:68:8A:59:3D:48:5F:81:97:3C:0F:91:95:43:1E:DA:37:CC:5E:36:43:0E:79:C7:A8:88:63:8
Alias name: verisignc3g3.pem
  SHA1: 13:2D:0D:45:53:4B:69:97:CD:B2:D5:C3:39:E2:55:76:60:9B:5C:C6
  SHA256:
 EB:04:CF:5E:B1:F3:9A:FA:76:2F:2B:B1:20:F2:96:CB:A5:20:C1:B9:7D:B1:58:95:65:B8:1C:B9:A1:7B:72:4
Alias name: verisignc3g4.pem
  SHA1: 22:D5:D8:DF:8F:02:31:D1:8D:F7:9D:B7:CF:8A:2D:64:C9:3F:6C:3A
  SHA256:
 69:DD:D7:EA:90:BB:57:C9:3E:13:5D:C8:5E:A6:FC:D5:48:0B:60:32:39:BD:C4:54:FC:75:8B:2A:26:CF:7F:7
Alias name: verisignc3g5.pem
  SHA1: 4E:B6:D5:78:49:9B:1C:CF:5F:58:1E:AD:56:BE:3D:9B:67:44:A5:E5
  SHA256:
 9A:CF:AB:7E:43:C8:D8:80:D0:6B:26:2A:94:DE:EE:E4:B4:65:99:89:C3:D0:CA:F1:9B:AF:64:05:E4:1A:B7:D
Alias name: verisignc4g2.pem
  SHA1: 0B:77:BE:BB:CB:7A:A2:47:05:DE:CC:0F:BD:6A:02:FC:7A:BD:9B:52
  SHA256:
 44:64:0A:0A:0E:4D:00:0F:BD:57:4D:2B:8A:07:BD:B4:D1:DF:ED:3B:45:BA:AB:A7:6F:78:57:78:C7:01:19:6
Alias name: verisignc4g3.pem
  SHA1: C8:EC:8C:87:92:69:CB:4B:AB:39:E9:8D:7E:57:67:F3:14:95:73:9D
  SHA256:
 E3:89:36:0D:0F:DB:AE:B3:D2:50:58:4B:47:30:31:4E:22:2F:39:C1:56:A0:20:14:4E:8D:96:05:61:79:15:0
Alias name: verisignclass1ca
```

```
  SHA1: CE:6A:64:A3:09:E4:2F:BB:D9:85:1C:45:3E:64:09:EA:E8:7D:60:F1
  SHA256:
 51:84:7C:8C:BD:2E:9A:72:C9:1E:29:2D:2A:E2:47:D7:DE:1E:3F:D2:70:54:7A:20:EF:7D:61:0F:38:B8:84:2
Alias name: verisignclass1g2ca
  SHA1: 27:3E:E1:24:57:FD:C4:F9:0C:55:E8:2B:56:16:7F:62:F5:32:E5:47
  SHA256:
 34:1D:E9:8B:13:92:AB:F7:F4:AB:90:A9:60:CF:25:D4:BD:6E:C6:5B:9A:51:CE:6E:D0:67:D0:0E:C7:CE:9B:7
Alias name: verisignclass1g3ca
  SHA1: 20:42:85:DC:F7:EB:76:41:95:57:8E:13:6B:D4:B7:D1:E9:8E:46:A5
  SHA256:
 CB:B5:AF:18:5E:94:2A:24:02:F9:EA:CB:C0:ED:5B:B8:76:EE:A3:C1:22:36:23:D0:04:47:E4:F3:BA:55:4B:6
Alias name: verisignclass2g2ca
  SHA1: B3:EA:C4:47:76:C9:C8:1C:EA:F2:9D:95:B6:CC:A0:08:1B:67:EC:9D
  SHA256:
 3A:43:E2:20:FE:7F:3E:A9:65:3D:1E:21:74:2E:AC:2B:75:C2:0F:D8:98:03:05:BC:50:2C:AF:8C:2D:9B:41:A
Alias name: verisignclass2g3ca
  SHA1: 61:EF:43:D7:7F:CA:D4:61:51:BC:98:E0:C3:59:12:AF:9F:EB:63:11
  SHA256:
 92:A9:D9:83:3F:E1:94:4D:B3:66:E8:BF:AE:7A:95:B6:48:0C:2D:6C:6C:2A:1B:E6:5D:42:36:B6:08:FC:A1:B
Alias name: verisignclass3ca
  SHA1: A1:DB:63:93:91:6F:17:E4:18:55:09:40:04:15:C7:02:40:B0:AE:6B
  SHA256:
 A4:B6:B3:99:6F:C2:F3:06:B3:FD:86:81:BD:63:41:3D:8C:50:09:CC:4F:A3:29:C2:CC:F0:E2:FA:1B:14:03:0
Alias name: verisignclass3g2ca
  SHA1: 85:37:1C:A6:E5:50:14:3D:CE:28:03:47:1B:DE:3A:09:E8:F8:77:0F
  SHA256:
 83:CE:3C:12:29:68:8A:59:3D:48:5F:81:97:3C:0F:91:95:43:1E:DA:37:CC:5E:36:43:0E:79:C7:A8:88:63:8
Alias name: verisignclass3g3ca
  SHA1: 13:2D:0D:45:53:4B:69:97:CD:B2:D5:C3:39:E2:55:76:60:9B:5C:C6
  SHA256:
 EB:04:CF:5E:B1:F3:9A:FA:76:2F:2B:B1:20:F2:96:CB:A5:20:C1:B9:7D:B1:58:95:65:B8:1C:B9:A1:7B:72:4
Alias name: verisignclass3g4ca
  SHA1: 22:D5:D8:DF:8F:02:31:D1:8D:F7:9D:B7:CF:8A:2D:64:C9:3F:6C:3A
  SHA256:
 69:DD:D7:EA:90:BB:57:C9:3E:13:5D:C8:5E:A6:FC:D5:48:0B:60:32:39:BD:C4:54:FC:75:8B:2A:26:CF:7F:7
Alias name: verisignclass3g5ca
  SHA1: 4E:B6:D5:78:49:9B:1C:CF:5F:58:1E:AD:56:BE:3D:9B:67:44:A5:E5
  SHA256:
 9A:CF:AB:7E:43:C8:D8:80:D0:6B:26:2A:94:DE:EE:E4:B4:65:99:89:C3:D0:CA:F1:9B:AF:64:05:E4:1A:B7:D
Alias name: verisignclass3publicprimarycertificationauthorityg4
  SHA1: 22:D5:D8:DF:8F:02:31:D1:8D:F7:9D:B7:CF:8A:2D:64:C9:3F:6C:3A
  SHA256:
 69:DD:D7:EA:90:BB:57:C9:3E:13:5D:C8:5E:A6:FC:D5:48:0B:60:32:39:BD:C4:54:FC:75:8B:2A:26:CF:7F:7
Alias name: verisignclass3publicprimarycertificationauthorityg5
```

```
  SHA1: 4E:B6:D5:78:49:9B:1C:CF:5F:58:1E:AD:56:BE:3D:9B:67:44:A5:E5
  SHA256:
 9A:CF:AB:7E:43:C8:D8:80:D0:6B:26:2A:94:DE:EE:E4:B4:65:99:89:C3:D0:CA:F1:9B:AF:64:05:E4:1A:B7:D
Alias name: verisignroot.pem
  SHA1: 36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
  SHA256:
 23:99:56:11:27:A5:71:25:DE:8C:EF:EA:61:0D:DF:2F:A0:78:B5:C8:06:7F:4E:82:82:90:BF:B8:60:E8:4B:3
Alias name: verisigntsaca
  SHA1: 20:CE:B1:F0:F5:1C:0E:19:A9:F3:8D:B1:AA:8E:03:8C:AA:7A:C7:01
  SHA256:
 CB:6B:05:D9:E8:E5:7C:D8:82:B1:0B:4D:B7:0D:E4:BB:1D:E4:2B:A4:8A:7B:D0:31:8B:63:5B:F6:E7:78:1A:9
Alias name: verisignuniversalrootca
  SHA1: 36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
  SHA256:
 23:99:56:11:27:A5:71:25:DE:8C:EF:EA:61:0D:DF:2F:A0:78:B5:C8:06:7F:4E:82:82:90:BF:B8:60:E8:4B:3
Alias name: verisignuniversalrootcertificationauthority
  SHA1: 36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
  SHA256:
 23:99:56:11:27:A5:71:25:DE:8C:EF:EA:61:0D:DF:2F:A0:78:B5:C8:06:7F:4E:82:82:90:BF:B8:60:E8:4B:3
Alias name: xrampglobalca
  SHA1: B8:01:86:D1:EB:9C:86:A5:41:04:CF:30:54:F3:4C:52:B7:E5:58:C6
  SHA256:
 CE:CD:DC:90:50:99:D8:DA:DF:C5:B1:D2:09:B7:37:CB:E2:C1:8C:FB:2C:10:C0:FF:0B:CF:0D:32:86:FC:1A:A
Alias name: xrampglobalcaroot
  SHA1: B8:01:86:D1:EB:9C:86:A5:41:04:CF:30:54:F3:4C:52:B7:E5:58:C6
  SHA256:
 CE:CD:DC:90:50:99:D8:DA:DF:C5:B1:D2:09:B7:37:CB:E2:C1:8C:FB:2C:10:C0:FF:0B:CF:0D:32:86:FC:1A:A
```

Menggunakan AWS WAF untuk melindungi API Anda

AWS WAFadalah firewall aplikasi web yang membantu melindungi aplikasi web dan API dari serangan. Ini memungkinkan Anda untuk mengonfigurasi seperangkat aturan yang disebut daftar kontrol akses web (web ACL) yang memungkinkan, memblokir, atau menghitung permintaan web berdasarkan aturan dan kondisi keamanan web yang dapat disesuaikan yang Anda tentukan. Untuk informasi selengkapnya, lihat Cara AWS WAF Kerja.

Anda dapat menggunakan AWS WAF untuk melindungi API Gateway REST API Anda dari eksploitasi web umum, seperti injeksi SQL dan serangan cross-site scripting (XSS). Ini dapat memengaruhi ketersediaan dan kinerja API, membahayakan keamanan, atau mengkonsumsi sumber daya yang berlebihan. Misalnya, Anda dapat membuat aturan untuk mengizinkan atau memblokir permintaan dari rentang alamat IP tertentu, permintaan dari blok CIDR, permintaan yang berasal dari

negara atau wilayah tertentu, permintaan yang berisi kode SQL berbahaya, atau permintaan yang berisi skrip berbahaya.

Anda juga dapat membuat aturan yang cocok dengan string tertentu atau pola ekspresi reguler di header HTTP, metode, string kueri, URI, dan badan permintaan (terbatas pada 8 KB pertama). Selain itu, Anda dapat membuat aturan untuk memblokir serangan dari agen pengguna tertentu, bot buruk, dan pencakar konten. Misalnya, Anda dapat menggunakan aturan berbasis tarif untuk menentukan jumlah permintaan web yang diizinkan oleh setiap IP klien dalam periode 5 menit yang terus diperbarui.

> **Important**
>
> AWS WAFadalah garis pertahanan pertama Anda terhadap eksploitasi web. Bila AWS WAF diaktifkan pada API, AWS WAF aturan dievaluasi sebelum fitur kontrol akses lainnya, seperti kebijakan sumber daya, kebijakanIAM, otorisasi Lambda, dan otorisasi Amazon Cognito. Misalnya, jika AWS WAF memblokir akses dari blok CIDR yang diizinkan oleh kebijakan sumber daya, AWS WAF diutamakan dan kebijakan sumber daya tidak dievaluasi.

AWS WAFUntuk mengaktifkan API Anda, Anda perlu melakukan hal berikut:

1. Gunakan AWS WAF konsol, AWS SDK, atau CLI untuk membuat ACL web Regional yang berisi kombinasi aturan terkelola yang diinginkan dan aturan AWS WAF kustom Anda sendiri. Untuk informasi selengkapnya, lihat Memulai AWS WAF dan Membuat dan Mengkonfigurasi Daftar Kontrol Akses Web (Web ACL).

   > **Important**
   >
   > API Gateway membutuhkan ACL web Regional.

2. Kaitkan ACL web AWS WAF Regional dengan tahap API. Anda dapat melakukannya dengan menggunakan AWS WAF konsol, AWS SDK, CLI, atau dengan menggunakan konsol API Gateway.

Untuk mengaitkan ACL Web AWS WAF regional dengan tahap API Gateway API menggunakan konsol API Gateway

Untuk menggunakan konsol API Gateway untuk mengaitkan ACL web AWS WAF Regional dengan tahap API Gateway API yang ada, gunakan langkah-langkah berikut:

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API yang sudah ada atau buat yang baru.
3. Di panel navigasi utama, pilih Tahapan, lalu pilih panggung.
4. Di bagian Detail tahap, pilih Edit.
5. Untuk mengaitkan ACL web Regional dengan tahap API:
   - Di bawah Web application firewall (AWSWAF), pilih ACL web Regional yang ingin Anda kaitkan dengan tahap ini.
6. Pilih Save changes (Simpan perubahan).

Kaitkan ACL Web AWS WAF regional dengan tahap API Gateway API menggunakan AWS CLI

Untuk menggunakan AWS CLI untuk mengaitkan ACL web AWS WAF Regional dengan tahap API Gateway API yang ada, panggil `associate-web-acl`perintah tersebut, seperti pada contoh berikut:

```
aws waf-regional associate-web-acl \
--web-acl-id 'aabc123a-fb4f-4fc6-becb-2b00831cadcf' \
--resource-arn 'arn:aws:apigateway:{region}::/restapis/4wk1k4onj3/stages/prod'
```

Kaitkan ACL web AWS WAF regional dengan tahap API menggunakan AWS WAF REST API

Untuk menggunakan AWS WAF REST API untuk mengaitkan ACL web AWS WAF Regional dengan tahap API Gateway API yang ada, panggil `AssociateWebACL`perintah tersebut, seperti pada contoh berikut:

```
import boto3

waf = boto3.client('waf-regional')
```

```
waf.associate_web_acl(
    WebACLId='aabc123a-fb4f-4fc6-becb-2b00831cadcf',
    ResourceArn='arn:aws:apigateway:{region}::/restapis/4wk1k4onj3/stages/prod'
)
```

Permintaan Throttle API untuk throughput yang lebih baik

Anda dapat mengonfigurasi pembatasan dan kuota untuk API Anda untuk membantu melindungi mereka dari kewalahan oleh terlalu banyak permintaan. Baik throttle dan kuota diterapkan atas dasar upaya terbaik dan harus dianggap sebagai target daripada plafon permintaan yang dijamin.

API Gateway membatasi permintaan ke API Anda menggunakan algoritme token bucket, tempat token diperhitungkan untuk permintaan. Secara khusus, API Gateway memeriksa tingkat dan ledakan pengiriman permintaan terhadap semua API di akun Anda, per Wilayah. Dalam algoritma token bucket, burst dapat memungkinkan overrun yang telah ditentukan sebelumnya dari batas-batas tersebut, tetapi faktor lain juga dapat menyebabkan batas dikuasai dalam beberapa kasus.

Jika pengiriman permintaan melebihi tingkat permintaan kondisi tunak dan batas burst, API Gateway mulai membatasi permintaan. Klien mungkin menerima tanggapan `429 Too Many Requests` kesalahan pada saat ini. Setelah menangkap pengecualian tersebut, klien dapat mengirimkan kembali permintaan yang gagal dengan cara yang membatasi tarif.

Sebagai pengembang API, Anda dapat menetapkan batas target untuk setiap tahapan atau metode API untuk meningkatkan kinerja keseluruhan di semua API di akun Anda. Atau, Anda dapat mengaktifkan paket penggunaan untuk mengatur pembatasan pada pengiriman permintaan klien berdasarkan tarif permintaan dan kuota yang ditentukan.

Topik

- Bagaimana pengaturan batas pelambatan diterapkan di API Gateway
- Pelambatan tingkat akun per Wilayah
- Mengonfigurasi target pelambatan level API dan level tahap dalam rencana penggunaan
- Mengkonfigurasi target pelambatan tingkat tahap
- Mengonfigurasi target pelambatan tingkat metode dalam rencana penggunaan

Bagaimana pengaturan batas pelambatan diterapkan di API Gateway

Sebelum Anda mengonfigurasi setelan throttle dan kuota untuk API Anda, penting untuk memahami cara penerapannya oleh Amazon API Gateway.

Amazon API Gateway menyediakan empat tipe dasar pengaturan terkait pelambatan:

- AWSbatas pembatasan diterapkan di semua akun dan klien di suatu wilayah. Pengaturan batas ini ada untuk mencegah API Anda—dan akun Anda—kewalahan oleh terlalu banyak permintaan. Batasan ini ditetapkan oleh AWS dan tidak dapat diubah oleh pelanggan.
- Batas per akun diterapkan ke semua API di akun di Wilayah tertentu. Batas tingkat tingkat akun dapat ditingkatkan berdasarkan permintaan - batas yang lebih tinggi dimungkinkan dengan API yang memiliki batas waktu lebih pendek dan muatan yang lebih kecil. Untuk meminta peningkatan batas pembatasan tingkat akun per Wilayah, hubungi Pusat Dukungan. AWS Untuk informasi selengkapnya, lihat Kuota dan catatan penting. Perhatikan bahwa batas ini tidak boleh lebih tinggi dari batas AWS pelambatan.
- Batas pelambatan per API, per tahap diterapkan pada tingkat metode API untuk suatu tahap. Anda dapat mengonfigurasi pengaturan yang sama untuk semua metode, atau mengonfigurasi pengaturan throttle yang berbeda untuk setiap metode. Perhatikan bahwa batas ini tidak boleh lebih tinggi dari batas AWS pelambatan.
- Batas pembatasan per klien diterapkan pada klien yang menggunakan kunci API yang terkait dengan paket penggunaan Anda sebagai pengenal klien. Perhatikan bahwa batas ini tidak boleh lebih tinggi dari batas per akun.

Setelan terkait pelambatan API Gateway diterapkan dalam urutan berikut:

1. Batas pembatasan per klien atau per-metode yang Anda tetapkan untuk tahap API dalam paket penggunaan
2. Batas pembatasan per metode yang Anda tetapkan untuk tahap API
3. Pelambatan tingkat akun per Wilayah
4. AWSPelambatan regional

Pelambatan tingkat akun per Wilayah

Secara default, API Gateway membatasi permintaan steady-state per detik (RPS) di semua API dalam AWS akun, per Wilayah. Ini juga membatasi burst (yaitu, ukuran bucket maksimum) di semua API dalam AWS akun, per Wilayah. Di API Gateway, batas burst mewakili jumlah maksimum target pengiriman permintaan bersamaan yang akan dipenuhi oleh API Gateway sebelum mengembalikan respons `429 Too Many Requests` kesalahan. Untuk informasi lebih lanjut tentang pembatasan kuota, lihat. Kuota dan catatan penting

Mengonfigurasi target pelambatan level API dan level tahap dalam rencana penggunaan

Dalam paket penggunaan, Anda dapat menetapkan target pelambatan per metode untuk semua metode di API atau level tahap. Anda dapat menentukan laju pelambatan, yang merupakan tingkat, dalam permintaan per detik, token tersebut ditambahkan ke keranjang token. Anda juga dapat menentukan burst throttling, yang merupakan kapasitas bucket token.

Mengkonfigurasi target pelambatan tingkat tahap

Anda dapat menggunakan AWS CLI, SDK, dan AWS Management Console untuk membuat target pelambatan tingkat tahap. Untuk informasi selengkapnya, lihat ???.

Mengonfigurasi target pelambatan tingkat metode dalam rencana penggunaan

Anda dapat menetapkan target pelambatan tambahan pada tingkat metode dalam Rencana Penggunaan seperti yang ditunjukkan pada. Buat rencana penggunaan Di konsol API Gateway, ini diatur dengan menentukan`Resource=<resource>`, `Method=<method>` dalam pengaturan Configure Method Throttling. Misalnya, PetStoremisalnya, Anda dapat menentukan`Resource=/pets`,`Method=GET`.

Membuat API pribadi di Amazon API Gateway

Menggunakan Amazon API Gateway, Anda dapat membuat API REST pribadi yang hanya dapat diakses dari cloud pribadi virtual Anda di Amazon VPC dengan menggunakan titik akhir VPC antarmuka. Ini adalah antarmuka jaringan endpoint yang Anda buat di VPC Anda.

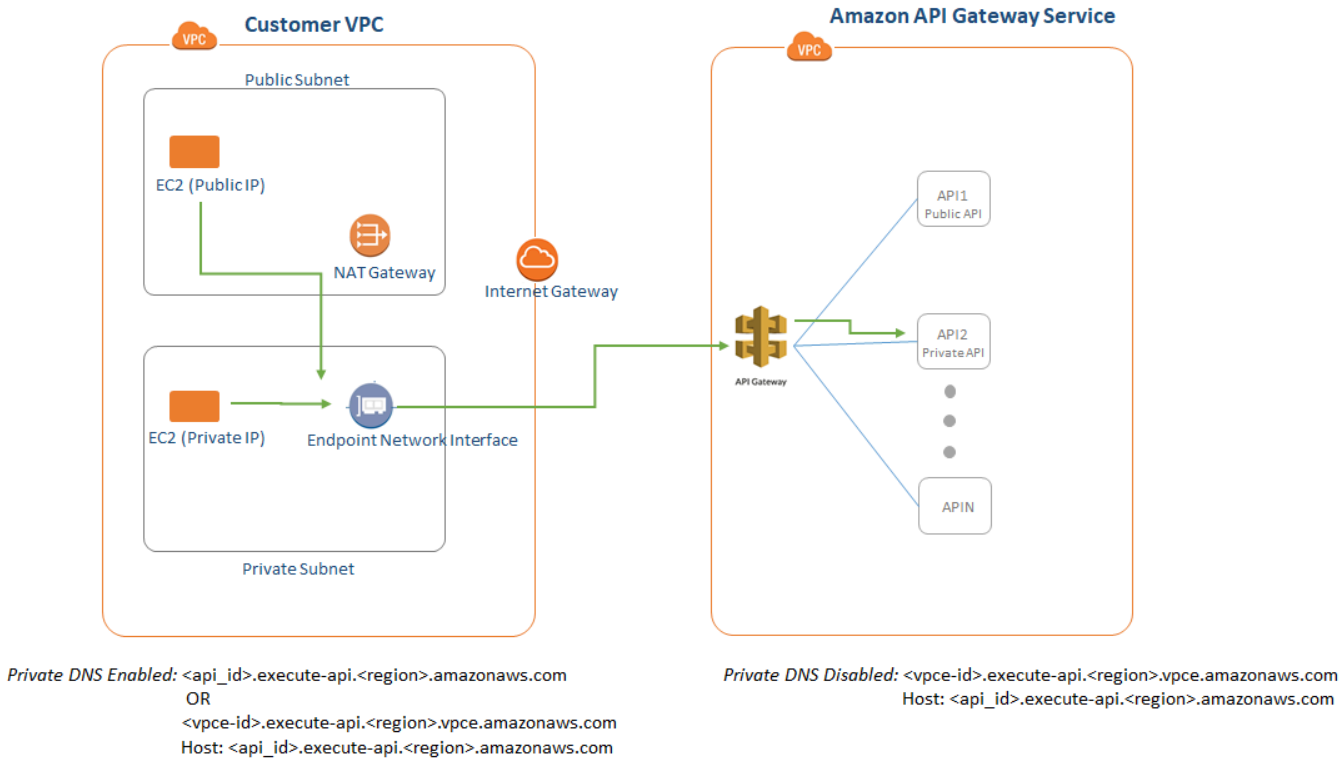
Dengan menggunakan kebijakan sumber daya, Anda dapat mengizinkan atau menolak akses ke API dari VPC dan titik akhir VPC yang dipilih, termasuk di seluruh akun. AWS Setiap titik akhir dapat digunakan untuk mengakses beberapa API pribadi. Anda juga dapat menggunakan AWS Direct Connect untuk membuat sambungan dari jaringan lokal ke Amazon VPC dan mengakses API pribadi Anda melalui koneksi tersebut.

> **Important**
>
> Untuk membatasi akses ke API pribadi Anda ke VPC atau titik akhir VPC tertentu, tambahkan `aws:SourceVpc` atau `aws:SourceVpce` kondisi ke kebijakan sumber daya API Anda. Untuk kebijakan-kebijakan contoh, lihat the section called “Contoh: Izinkan lalu lintas API pribadi berdasarkan titik akhir VPC atau VPC sumber”.

Dalam semua kasus, lalu lintas ke API pribadi Anda menggunakan koneksi aman dan tidak meninggalkan jaringan Amazon — itu terisolasi dari internet publik.

Anda dapat mengakses API pribadi Anda melalui titik akhir VPC antarmuka untuk API Gateway seperti yang ditunjukkan pada diagram berikut. Jika DNS pribadi diaktifkan, Anda dapat menggunakan nama DNS pribadi atau publik untuk mengakses API Anda. Jika DNS pribadi dinonaktifkan, Anda hanya dapat menggunakan nama DNS publik.

> **Note**
>
> API pribadi API Gateway hanya mendukung TLS 1.2. Versi TLS sebelumnya tidak didukung.



Pada tingkat tinggi, langkah-langkah untuk membuat API pribadi adalah sebagai berikut:

1. Pertama, buat endpoint VPC antarmuka untuk layanan komponen API Gateway untuk eksekusi API, yang dikenal sebagai`execute-api`, di VPC Anda.
2. Buat dan uji API pribadi Anda.

    a. Gunakan salah satu prosedur berikut untuk membuat API Anda:

- Konsol API Gateway
- API Gateway CLI
- AWS SDK untuk JavaScript

b. Untuk memberikan akses ke titik akhir VPC Anda, buat kebijakan sumber daya dan lampirkan ke API Anda.

c. Uji API Anda.

> **Note**
>
> Prosedur di bawah ini mengasumsikan Anda sudah memiliki VPC yang sepenuhnya dikonfigurasi. Untuk informasi selengkapnya, dan untuk memulai membuat VPC, lihat Memulai Dengan Amazon VPC di Panduan Pengguna Amazon VPC.

Pertimbangan pengembangan API pribadi

- Anda dapat mengonversi API publik yang ada (Regional atau dioptimalkan tepi) ke API pribadi, dan Anda dapat mengonversi API pribadi ke API Regional. Anda tidak dapat mengonversi API pribadi ke API yang dioptimalkan tepi. Untuk informasi selengkapnya, lihat ???.
- Untuk memberikan akses ke API pribadi Anda ke VPC dan titik akhir VPC, Anda perlu membuat kebijakan sumber daya dan melampirkannya ke API yang baru dibuat (atau dikonversi). Sampai Anda melakukannya, semua panggilan ke API akan gagal. Untuk informasi selengkapnya, lihat ???.
- Nama domain khusus tidak didukung untuk API pribadi.
- Anda dapat menggunakan satu titik akhir VPC untuk mengakses beberapa API pribadi.
- Anda dapat mengaitkan atau memisahkan titik akhir VPC ke REST API, yang memberikan catatan DNS alias Route 53 dan menyederhanakan menjalankan API pribadi Anda. Untuk informasi selengkapnya, lihat Mengaitkan atau Memutuskan Titik Akhir VPC dengan REST API Pribadi.

> **Note**
>
> Titik akhir VPC untuk API pribadi tunduk pada batasan yang sama dengan titik akhir VPC antarmuka lainnya. Untuk informasi selengkapnya, lihat Properti dan Batasan Titik Akhir Antarmuka dalam AWS PrivateLink Panduan. Untuk informasi selengkapnya tentang

penggunaan API Gateway dengan VPC bersama dan subnet bersama, lihat Subnet bersama dalam Panduan.AWS PrivateLink

Topik

- Buat titik akhir VPC antarmuka untuk API Gateway execute-api
- Membuat API pribadi menggunakan konsol API Gateway
- Buat API pribadi menggunakan AWS CLI
- Membuat API pribadi menggunakan AWS SDK untuk JavaScript
- Menyiapkan kebijakan sumber daya untuk API pribadi
- Menerapkan API pribadi menggunakan konsol API Gateway
- Kaitkan atau pisahkan titik akhir VPC dengan REST API pribadi

Buat titik akhir VPC antarmuka untuk API Gateway **`execute-api`**

Layanan komponen API Gateway untuk eksekusi API dipanggil`execute-api`. Untuk mengakses API pribadi Anda setelah digunakan, Anda perlu membuat titik akhir VPC antarmuka untuknya di VPC Anda.

Setelah membuat titik akhir VPC, Anda dapat menggunakannya untuk mengakses beberapa API pribadi.

Untuk membuat antarmuka VPC endpoint untuk API Gateway **`execute-api`**

1. Masuk ke AWS Management Console dan buka konsol VPC Amazon di https://console.aws.amazon.com/vpc/.
2. Di panel navigasi, pilih Endpoints, Create Endpoint.
3. Untuk kategori Layanan, pastikan bahwa AWS layanan dipilih.
4. Untuk Nama Layanan, pilih titik akhir layanan API Gateway, termasuk AWS Wilayah yang ingin Anda sambungkan. Ini dalam bentuk `com.amazonaws.`*`region`*`.execute-api` — misalnya,`com.amazonaws.us-east-1.execute-api`.

   Untuk Tipe, pastikan itu menunjukkan Antarmuka.
5. Lengkapi informasi berikut:

   - Untuk VPC, pilih VPC tempat Anda ingin membuat endpoint.

- Untuk Subnet, pilih subnet (Availability Zones) untuk membuat antarmuka jaringan endpoint. Untuk meningkatkan ketersediaan API Anda, pilih beberapa subnet.

  > **Note**
  >
  > Tidak semua Availability Zone dapat didukung untuk semua AWS layanan.

- Untuk Aktifkan Nama DNS Pribadi, biarkan kotak centang dipilih. DNS pribadi diaktifkan secara default.

  Ketika DNS pribadi diaktifkan, Anda dapat mengakses API Anda melalui DNS pribadi atau publik. (Pengaturan ini tidak memengaruhi siapa yang dapat mengakses API Anda, hanya alamat DNS mana yang dapat mereka gunakan.) Namun, Anda tidak dapat mengakses API publik dari VPC dengan menggunakan titik akhir VPC API Gateway dengan DNS pribadi diaktifkan. Perhatikan bahwa pengaturan DNS ini tidak memengaruhi kemampuan untuk memanggil API publik ini dari VPC jika Anda menggunakan nama domain kustom yang dioptimalkan tepi untuk mengakses API publik. Menggunakan nama domain kustom yang dioptimalkan untuk mengakses API publik Anda (saat menggunakan DNS pribadi untuk mengakses API pribadi Anda) adalah salah satu cara untuk mengakses API publik dan pribadi dari VPC di mana titik akhir telah dibuat dengan DNS pribadi diaktifkan.

  > **Note**
  >
  > Membiarkan DNS pribadi diaktifkan adalah pilihan yang disarankan. Jika Anda memilih untuk tidak mengaktifkan DNS pribadi, Anda hanya dapat mengakses API Anda melalui DNS publik. Untuk mempelajari selengkapnya, lihat Cara memanggil API pribadi.

  Untuk menggunakan opsi DNS pribadi, `enableDnsHostnames` atribut `enableDnsSupport` dan VPC Anda harus disetel ke. `true` Untuk informasi selengkapnya, lihat Dukungan DNS di VPC Anda dan Memperbarui Dukungan DNS untuk VPC Anda di Panduan Pengguna Amazon VPC.

- Untuk grup Keamanan, pilih grup keamanan untuk diasosiasikan dengan antarmuka jaringan titik akhir VPC.

  Grup keamanan yang Anda pilih harus diatur untuk mengizinkan lalu lintas HTTPS masuk TCP Port 443 dari rentang IP di VPC Anda atau grup keamanan lain di VPC Anda.

6. Pilih Buat titik akhir.

Membuat API pribadi menggunakan konsol API Gateway

Untuk membuat API pribadi menggunakan konsol API Gateway

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih Buat API.
3. Di bawah REST API, pilih Build.
4. Untuk Nama, masukkan nama.
5. (Opsional) Untuk Deskripsi, masukkan deskripsi.
6. Untuk jenis endpoint API, pilih Private.
7. Pilih Buat API.

Mulai sekarang, Anda dapat menyiapkan metode API dan integrasi terkaitnya seperti yang dijelaskan dalam langkah 1-6 dari. ???

> **Note**
>
> Hingga API Anda memiliki kebijakan sumber daya yang memberikan akses ke titik akhir VPC atau VPC Anda, semua panggilan API akan gagal. Sebelum menguji dan menerapkan API, Anda perlu membuat kebijakan sumber daya dan melampirkannya ke API seperti yang dijelaskan dalam???.

Buat API pribadi menggunakan AWS CLI

Untuk membuat API pribadi menggunakan AWS CLI, panggil `create-rest-api` perintah:

```
aws apigateway create-rest-api \
        --name 'Simple PetStore (AWS CLI, Private)' \
        --description 'Simple private PetStore API' \
        --region us-west-2 \
        --endpoint-configuration '{ "types": ["PRIVATE"] }'
```

Panggilan yang berhasil mengembalikan output yang mirip dengan berikut ini:

```
{
    "createdDate": "2017-10-13T18:41:39Z",
    "description": "Simple private PetStore API",
    "endpointConfiguration": {
        "types": "PRIVATE"
    },
    "id": "0qzs2sy7bh",
    "name": "Simple PetStore (AWS CLI, Private)"
}
```

Mulai sekarang, Anda dapat mengikuti instruksi yang sama yang diberikan the section called "Siapkan API yang dioptimalkan tepi menggunakan perintah AWS CLI" untuk menyiapkan metode dan integrasi untuk API ini.

Ketika Anda siap untuk menguji API Anda, pastikan untuk membuat kebijakan sumber daya dan melampirkannya ke API seperti yang dijelaskan dalam???.

Membuat API pribadi menggunakan AWS SDK untuk JavaScript

Untuk membuat API pribadi dengan menggunakan AWS SDK untuk 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);
 }
});
```

Panggilan yang berhasil mengembalikan output yang mirip dengan berikut ini:

```
{
    "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)"
}
```

Setelah menyelesaikan langkah-langkah sebelumnya, Anda dapat mengikuti petunjuk the section called “Siapkan API yang dioptimalkan tepi menggunakan AWS SDK untuk Node.js” untuk menyiapkan metode dan integrasi untuk API ini.

Ketika Anda siap untuk menguji API Anda, pastikan untuk membuat kebijakan sumber daya dan melampirkannya ke API seperti yang dijelaskan dalam???.

Menyiapkan kebijakan sumber daya untuk API pribadi

Sebelum API pribadi Anda dapat diakses, Anda perlu membuat kebijakan sumber daya dan melampirkannya ke API. Ini memberikan akses ke API dari VPC dan titik akhir VPC Anda atau dari VPC dan titik akhir VPC di akun lain yang Anda berikan akses secara eksplisit. AWS

Untuk melakukan ini, ikuti instruksi dithe section called “Membuat dan melampirkan kebijakan sumber daya API Gateway ke API”. Pada langkah 5, pilih contoh Sumber VPC. Ganti `{{vpceID}}` (termasuk kurawal kurawal) dengan ID titik akhir VPC Anda, lalu pilih Simpan untuk menyimpan kebijakan sumber daya Anda.

Anda juga harus mempertimbangkan untuk melampirkan kebijakan titik akhir ke titik akhir VPC untuk menentukan akses yang diberikan. Untuk informasi selengkapnya, lihat the section called “Menggunakan kebijakan titik akhir VPC untuk API pribadi”.

Menerapkan API pribadi menggunakan konsol API Gateway

Untuk menerapkan API pribadi Anda, lakukan hal berikut di konsol API Gateway:

1. Pilih API Anda.
2. Pilih Deploy API.
3. Untuk Stage, pilih New stage.
4. Untuk nama Panggung, masukkan nama panggung.
5. (Opsional) Untuk Deskripsi, masukkan deskripsi.
6. Pilih Deploy.

Kaitkan atau pisahkan titik akhir VPC dengan REST API pribadi

Saat Anda mengaitkan titik akhir VPC dengan API pribadi Anda, API Gateway menghasilkan catatan DNS Route 53 ALIAS baru. Anda dapat menggunakan data ini untuk menjalankan API pribadi seperti halnya Anda melakukan API yang dioptimalkan tepi atau Regional tanpa mengganti `Host` header atau meneruskan header. `x-apigw-api-id`

URL dasar yang dihasilkan dalam format berikut:

```
https://{rest-api-id}-{vpce-id}.execute-api.{region}.amazonaws.com/{stage}
```

Mengaitkan atau memisahkan titik akhir VPC dengan REST API pribadi mengharuskan Anda memperbarui konfigurasi API. Anda dapat melakukan perubahan ini menggunakan konsol API Gateway, AWS CLI, atau AWS SDK untuk API Gateway. Operasi pembaruan mungkin memakan waktu beberapa menit untuk diselesaikan karena propagasi DNS. Selama waktu ini, API Anda tersedia, tetapi propagasi DNS untuk URL DNS yang baru dibuat mungkin masih dalam proses. Anda dapat mencoba membuat penerapan baru untuk API Anda, jika bahkan setelah beberapa menit URL baru Anda tidak diselesaikan dalam DNS.

Gunakan konsol API Gateway untuk mengaitkan titik akhir VPC dengan REST API pribadi

Untuk mengaitkan titik akhir VPC tambahan dengan API pribadi

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API pribadi Anda.
3. Di panel navigasi utama, pilih Kebijakan sumber daya.
4. Edit kebijakan sumber daya Anda untuk mengizinkan panggilan dari titik akhir VPC tambahan Anda.
5. Di panel navigasi utama, pilih pengaturan API.
6. Di bagian detail API, pilih Edit.
7. Untuk ID titik akhir VPC, pilih ID titik akhir VPC tambahan.
8. Pilih Simpan.
9. Menerapkan ulang API Anda agar perubahan diterapkan.

Gunakan AWS CLI untuk mengaitkan titik akhir VPC dengan REST API pribadi

Untuk mengaitkan titik akhir VPC pada saat pembuatan API, gunakan perintah berikut:

```
aws apigateway create-rest-api \
    --name Petstore \
    --endpoint-configuration '{ "types": ["PRIVATE"], "vpcEndpointIds" :
 ["vpce-0212a4ababd5b8c3e", "vpce-0393a628149c867ee"] }' \
    --region us-west-2
```

Outputnya akan terlihat seperti berikut:

```
{
    "apiKeySource": "HEADER",
    "endpointConfiguration": {
        "types": [
            "PRIVATE"
        ],
        "vpcEndpointIds": [
            "vpce-0212a4ababd5b8c3e",
            "vpce-0393a628149c867ee"
        ]
    },
    "id": "u67n3ov968",
    "createdDate": 1565718256,
    "name": "Petstore"
}
```

Untuk mengaitkan titik akhir VPC ke API pribadi yang sudah dibuat, gunakan perintah CLI berikut:

```
aws apigateway update-rest-api \
    --rest-api-id u67n3ov968 \
    --patch-operations "op='add',path='/endpointConfiguration/
vpcEndpointIds',value='vpce-01d622316a7df47f9'" \
    --region us-west-2
```

Outputnya akan terlihat seperti berikut:

```
{
    "name": "Petstore",
    "apiKeySource": "1565718256",
    "tags": {},
    "createdDate": 1565718256,
    "endpointConfiguration": {
        "vpcEndpointIds": [
```

```
            "vpce-0212a4ababd5b8c3e",
            "vpce-0393a628149c867ee",
            "vpce-01d622316a7df47f9"
        ],
        "types": [
            "PRIVATE"
        ]
    },
    "id": "u67n3ov968"
}
```

Menggunakan konsol API Gateway untuk memisahkan titik akhir VPC dari REST API pribadi

Untuk memisahkan titik akhir VPC dari REST API pribadi

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API pribadi Anda.
3. Di panel navigasi utama, pilih Kebijakan sumber daya.
4. Edit kebijakan sumber daya Anda untuk menghapus penyebutan titik akhir VPC yang ingin Anda pisahkan dari API pribadi Anda.
5. Di panel navigasi utama, pilih pengaturan API.
6. Di bagian detail API, pilih Edit.
7. Untuk ID titik akhir VPC, pilih X untuk memisahkan titik akhir VPC.
8. Pilih Simpan.
9. Menerapkan ulang API Anda agar perubahan diterapkan.

Gunakan AWS CLI untuk memisahkan titik akhir VPC dari REST API pribadi

Untuk memisahkan titik akhir VPC dari API pribadi, gunakan perintah CLI berikut:

```
aws apigateway update-rest-api \
    --rest-api-id u67n3ov968 \
    --patch-operations "op='remove',path='/endpointConfiguration/
vpcEndpointIds',value='vpce-0393a628149c867ee'" \
    --region us-west-2
```

Outputnya akan terlihat seperti berikut:

```
{
```

```
    "name": "Petstore",
    "apiKeySource": "1565718256",
    "tags": {},
    "createdDate": 1565718256,
    "endpointConfiguration": {
        "vpcEndpointIds": [
            "vpce-0212a4ababd5b8c3e",
            "vpce-01d622316a7df47f9"
        ],
        "types": [
            "PRIVATE"
        ]
    },
    "id": "u67n3ov968"
}
```

Memantau REST API

Di bagian ini, Anda dapat mempelajari cara memantau API menggunakan CloudWatch metrik, CloudWatch Log, Firehose, dan. AWS X-Ray Dengan menggabungkan log CloudWatch eksekusi dan CloudWatch metrik, Anda dapat mencatat kesalahan dan jejak eksekusi, serta memantau kinerja API Anda. Anda mungkin juga ingin mencatat panggilan API ke Firehose. Anda juga dapat menggunakan AWS X-Ray untuk melacak panggilan melalui layanan hilir yang membentuk API Anda.

> **Note**
>
> API Gateway mungkin tidak menghasilkan log dan metrik dalam kasus berikut:
>
> - 413 Kesalahan Permintaan Entitas Terlalu Besar
> - Berlebihan 429 Terlalu Banyak Kesalahan Permintaan
> - 400 error seri dari permintaan yang dikirim ke domain kustom yang tidak memiliki pemetaan API
> - 500 kesalahan seri yang disebabkan oleh kegagalan internal
>
> API Gateway tidak akan menghasilkan log dan metrik saat menguji metode REST API. CloudWatch Entri disimulasikan. Lihat informasi yang lebih lengkap di the section called “Gunakan konsol untuk menguji metode REST API”.

Topik

- Memantau eksekusi REST API dengan CloudWatch metrik Amazon
- Menyiapkan CloudWatch logging untuk REST API di API Gateway
- Pencatatan panggilan API ke Amazon Data Firehose
- Menelusuri permintaan pengguna ke REST API menggunakan X-Ray

Memantau eksekusi REST API dengan CloudWatch metrik Amazon

Anda dapat memantau eksekusi API dengan menggunakanCloudWatch, yang mengumpulkan dan memproses data mentah dari API Gateway menjadi metrik yang dapat dibaca. near-real-time Statistik ini dicatat untuk jangka waktu 15 bulan sehingga Anda dapat mengakses informasi historis dan mendapatkan perspektif yang lebih baik tentang bagaimana kinerja aplikasi atau layanan web Anda. Secara default, data metrik API Gateway secara otomatis dikirim CloudWatch dalam periode satu menit. Untuk informasi lebih lanjut, lihat Apa Itu Amazon CloudWatch? dalam Panduan Pengguna Amazon CloudWatch.

Metrik yang dilaporkan oleh API Gateway memberikan informasi yang dapat Anda analisis dengan cara yang berbeda. Daftar berikut menunjukkan beberapa kegunaan umum untuk metrik yang merupakan saran untuk membantu Anda memulai:

- Pantau IntegrationLatencymetrik untuk mengukur respons backend.
- Pantau metrik Latensi untuk mengukur respons keseluruhan panggilan API Anda.
- Pantau CacheHitCountdan CacheMissCountmetrik untuk mengoptimalkan kapasitas cache untuk mencapai kinerja yang diinginkan.

Topik

- Dimensi dan metrik Amazon API Gateway
- Lihat CloudWatch metrik dengan dasbor API di API Gateway
- Melihat metrik API Gateway di konsol CloudWatch
- Melihat peristiwa log API Gateway di CloudWatch konsol
- Alat pemantauan di AWS

Dimensi dan metrik Amazon API Gateway

Metrik dan dimensi yang dikirimkan API Gateway ke Amazon CloudWatch tercantum di bawah ini. Untuk informasi selengkapnya, lihat Memantau eksekusi REST API dengan CloudWatch metrik Amazon.

Metrik API Gateway

Amazon API Gateway mengirimkan data metrik ke CloudWatch setiap menit.

Namespace `AWS/ApiGateway` mencakup metrik berikut.

Metrik	Deskripsi
4XXError	Jumlah kesalahan sisi klien yang ditangkap dalam periode tertentu. API Gateway menghitung kode status respons gateway yang dimodifikasi sebagai kesalahan 4xxError. `Sum`Statistik mewakili metrik ini, yaitu, jumlah total 4XXError kesalahan dalam periode tertentu. `Average`Statistik mewakili tingkat 4XXError kesalahan, yaitu, jumlah total 4XXError kesalahan dibagi dengan jumlah total permintaan selama periode tersebut. Penyebut sesuai dengan Count metrik (di bawah). Unit: Count
5XXError	Jumlah kesalahan sisi server yang ditangkap dalam periode tertentu. `Sum`Statistik mewakili metrik ini, yaitu, jumlah total 5XXError kesalahan dalam periode tertentu. `Average`Statistik mewakili tingkat 5XXError kesalahan, yaitu, jumlah total 5XXError kesalahan dibagi dengan jumlah total permintaan selama

Metrik	Deskripsi
	periode tersebut. Penyebut sesuai dengan Count metrik (di bawah). Unit: Count
CacheHitCount	Jumlah permintaan yang disajikan dari cache API dalam periode tertentu. SumStatistik mewakili metrik ini, yaitu, jumlah total hit cache pada periode tertentu. AverageStatistik mewakili tingkat hit cache, yaitu, jumlah total klik cache dibagi dengan jumlah total permintaan selama periode tersebut. Penyebut sesuai dengan Count metrik (di bawah). Unit: Count
CacheMissCount	Jumlah permintaan yang disajikan dari backend dalam periode tertentu, saat caching API diaktifkan. SumStatistik mewakili metrik ini, yaitu, jumlah total cache yang hilang dalam periode tertentu. AverageStatistik mewakili tingkat kehilangan cache, yaitu, jumlah total cache yang hilang dibagi dengan jumlah total permintaan selama periode tersebut. Penyebut sesuai dengan Count metrik (di bawah). Unit: Count
Count	Jumlah total permintaan API dalam periode tertentu. SampleCount Statistik mewakili metrik ini. Unit: Count

Metrik	Deskripsi
IntegrationLatency	Waktu antara saat API Gateway menyampaikan permintaan ke backend dan saat menerima respons dari backend. Unit: Millisecond
Latency	Waktu antara saat API Gateway menerima permintaan dari klien dan saat mengembalikan respons ke klien. Latensi mencakup latensi integrasi dan overhead API Gateway lainnya. Unit: Millisecond

Dimensi untuk metrik

Anda dapat menggunakan dimensi dalam tabel berikut untuk memfilter metrik API Gateway.

> **Note**
>
> API Gateway menghapus karakter non-ASCII dari ApiName dimensi sebelum mengirim metrik ke. CloudWatch Jika tidak APIName mengandung karakter ASCII, API ID digunakan sebagai. ApiName

Dimensi	Deskripsi
ApiName	Memfilter metrik API Gateway untuk REST API dengan nama API yang ditentukan.
ApiName, Method, Resource, Stage	Memfilter metrik API Gateway untuk metode API dengan nama API, tahapan, sumber daya, dan metode yang ditentukan. API Gateway tidak akan mengirimkan metrik ini kecuali Anda telah mengaktifkan metrik terperinci secara eksplisit. CloudWatch Di konsol, pilih

Dimensi	Deskripsi
	panggung, lalu untuk Log dan penelusuran, pilih Edit. Pilih Metrik terperinci, lalu pilih Simpan perubahan . Atau, Anda dapat memanggil AWS CLI perin tah update-stage untuk memperbarui properti ke. `metricsEnabled true` Mengaktifkan metrik ini akan dikenakan biaya tambahan ke akun Anda. Untuk informasi harga, lihat CloudWatchHarga Amazon.
ApiName, Stage	Memfilter metrik API Gateway untuk sumber daya tahap API dengan nama dan tahapan API yang ditentukan.

Lihat CloudWatch metrik dengan dasbor API di API Gateway

Anda dapat menggunakan dasbor API di API Gateway Console untuk menampilkan CloudWatch metrik API yang Anda gunakan di API Gateway. Ini ditampilkan sebagai ringkasan aktivitas API dari waktu ke waktu.

Topik

- Prasyarat
- Periksa aktivitas API di dasbor

Prasyarat

1. Anda harus memiliki API yang dibuat di API Gateway. Ikuti instruksi di Membuat REST API di Amazon API Gateway.
2. Anda harus memiliki API yang diterapkan setidaknya sekali. Ikuti instruksi di Menerapkan REST API di Amazon API Gateway.

Periksa aktivitas API di dasbor

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API.

3. Di panel navigasi utama, pilih Dasbor.
4. Untuk Panggung, pilih tahap yang diinginkan.
5. Pilih Rentang tanggal untuk menentukan rentang tanggal.
6. Segarkan, jika diperlukan, dan lihat metrik individual yang ditampilkan dalam grafik terpisah berjudul Panggilan API, Latensi, Latensi integrasi, Latensi, kesalahan 4xx, dan kesalahan 5xx.

> **Tip**
>
> Untuk memeriksa CloudWatch metrik tingkat metode, pastikan Anda telah mengaktifkan CloudWatch Log pada tingkat metode. Untuk informasi selengkapnya tentang cara mengatur logging tingkat metode, lihat. Perbarui pengaturan panggung menggunakan konsol API Gateway

Melihat metrik API Gateway di konsol CloudWatch

Metrik dikelompokkan terlebih dahulu berdasarkan namespace layanan, kemudian berdasarkan berbagai kombinasi dimensi dalam setiap namespace.

Untuk melihat metrik API Gateway menggunakan konsol CloudWatch

1. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
2. Jika perlu, ubahWilayah AWS. Dari bilah navigasi, pilih Wilayah tempat AWS sumber daya Anda berada. Untuk informasi lebih lanjut, lihat Wilayah dan Titik Akhir.
3. Di panel navigasi, pilih Metrik.
4. Di tab Semua metrik, pilih API Gateway.
5. Untuk melihat metrik berdasarkan tahap, pilih panel Berdasarkan Panggung. Dan kemudian pilih API yang diinginkan dan nama metrik.
6. Untuk melihat metrik berdasarkan API tertentu, pilih panel By Api Name. Dan kemudian pilih API yang diinginkan dan nama metrik.

Untuk melihat metrik menggunakan CLI AWS

1. Pada prompt perintah, gunakan perintah berikut untuk mencantumkan metrik:

```
aws cloudwatch list-metrics --namespace "AWS/ApiGateway"
```

2. Untuk melihat statistik tertentu (misalnya,`Average`) selama periode waktu interval 5 menit, panggil perintah berikut:

```
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
```

Melihat peristiwa log API Gateway di CloudWatch konsol

Prasyarat

1. Anda harus memiliki API yang dibuat di API Gateway. Ikuti instruksi di Membuat REST API di Amazon API Gateway.
2. Anda harus memiliki API yang diterapkan dan dipanggil setidaknya sekali. Ikuti petunjuk di Menerapkan REST API di Amazon API Gateway danMemanggil REST API di Amazon API Gateway.
3. Anda harus memiliki CloudWatch Log diaktifkan untuk tahap. Ikuti instruksi di Menyiapkan CloudWatch logging untuk REST API di API Gateway.

Untuk melihat permintaan dan respons API yang dicatat menggunakan konsol CloudWatch

1. Buka konsol CloudWatch di https://console.aws.amazon.com/cloudwatch/.
2. Jika perlu, ubahWilayah AWS. Dari bilah navigasi, pilih Wilayah tempat AWS sumber daya Anda berada. Untuk informasi lebih lanjut, lihat Wilayah dan Titik Akhir.
3. Di panel navigasi, pilih Log, Log grup.
4. Di bawah tabel Grup Log, pilih grup log nama API-gateway-Execution-logs_ {}/{stage-name}. rest-api-id
5. Di bawah tabel Log Streams, pilih aliran log. Anda dapat menggunakan stempel waktu untuk membantu menemukan aliran log yang Anda minati.
6. Pilih Teks untuk melihat teks mentah atau pilih Baris untuk melihat baris peristiwa demi baris.

> **Important**
>
> CloudWatchmemungkinkan Anda menghapus grup log atau aliran. Jangan menghapus grup atau aliran log API Gateway API secara manual; biarkan API Gateway mengelola sumber daya ini. Menghapus grup log atau aliran secara manual dapat menyebabkan permintaan dan tanggapan API tidak dicatat. Jika itu terjadi, Anda dapat menghapus seluruh grup log untuk API dan men-deploy ulang API. Ini karena API Gateway membuat grup log atau aliran log untuk tahap API pada saat diterapkan.

Alat pemantauan di AWS

AWSmenyediakan berbagai alat yang dapat Anda gunakan untuk memantau API Gateway. Anda dapat mengonfigurasi beberapa alat ini untuk melakukan pemantauan untuk Anda secara otomatis, sementara alat lain memerlukan intervensi manual. Kami menyarankan agar Anda mengautomasi tugas pemantauan sebanyak mungkin.

Alat pemantauan otomatis di AWS

Anda dapat menggunakan alat pemantauan otomatis berikut untuk menonton API Gateway dan melaporkan ketika ada sesuatu yang salah:

- Amazon CloudWatch Alarm – Melihat metrik tunggal selama periode waktu yang Anda tentukan, dan lakukan satu atau beberapa tindakan berdasarkan nilai metrik relatif terhadap ambang batas tertentu selama sejumlah periode waktu. Tindakan ini adalah pengiriman notifikasi ke topik Amazon Simple Notification Service (Amazon SNS) atau kebijakan Amazon EC2 Auto Scaling. CloudWatchalarm tidak memanggil tindakan hanya karena mereka berada dalam keadaan tertentu; negara harus telah berubah dan dipertahankan untuk sejumlah periode tertentu. Untuk informasi selengkapnya, lihat Memantau eksekusi REST API dengan CloudWatch metrik Amazon.
- Amazon CloudWatch Logs — Pantau, simpan, dan akses file log Anda dari AWS CloudTrail atau sumber lain. Untuk informasi selengkapnya, lihat Memantau File Log di Panduan CloudWatch Pengguna Amazon.
- Amazon CloudWatch Events — Cocokkan peristiwa dan rutekan ke satu atau beberapa fungsi atau aliran target untuk membuat perubahan, menangkap informasi status, dan mengambil tindakan korektif. Untuk informasi selengkapnya, lihat Apa itu CloudWatch Acara Amazon di Panduan CloudWatch Pengguna Amazon.
- AWS CloudTrailPemantauan Log - Bagikan file log antar akun, pantau file CloudTrail log secara real time dengan mengirimkannya ke CloudWatch Log, tulis aplikasi pemrosesan log di Jawa,

dan validasi bahwa file log Anda tidak berubah setelah pengiriman. CloudTrail Untuk informasi selengkapnya, lihat Bekerja dengan File CloudTrail Log di Panduan AWS CloudTrail Pengguna.

Alat pemantauan manual

Bagian penting lainnya dari pemantauan API Gateway melibatkan pemantauan item yang tidak dicakup oleh CloudWatch alarm secara manual. API GatewayCloudWatch,, dan dasbor AWS konsol lainnya memberikan at-a-glance tampilan status AWS lingkungan Anda. Kami menyarankan Anda juga memeriksa file log pada eksekusi API.

- Dasbor API Gateway menunjukkan statistik berikut untuk tahap API tertentu selama periode waktu tertentu:
  - Panggilan API
  - Cache Hit, hanya ketika caching API diaktifkan.
  - Cache Miss, hanya ketika caching API diaktifkan.
  - Latensi
  - Latensi Integrasi
  - Kesalahan 4XX
  - Kesalahan 5XX
- Halaman CloudWatch rumah menunjukkan:
  - Alarm dan status saat ini
  - Grafik alarm dan sumber daya
  - Status kesehatan layanan

  Selain itu, Anda dapat menggunakan CloudWatch untuk melakukan hal berikut:
  - Membuat dasbor yang disesuaikan untuk memantau layanan yang ingin Anda ketahui
  - Grafik data metrik untuk memecahkan masalah dan menemukan tren
  - Cari dan telusuri semua metrik sumber daya AWS Anda
  - Membuat dan mengedit alarm untuk menerima notifikasi tentang masalah

Membuat CloudWatch alarm untuk memantau API Gateway

Anda dapat membuat alarm CloudWatch yang mengirimkan pesan ke Amazon SNS ketika status alarm berubah. Sebuah alarm mengawasi metrik tunggal selama periode waktu yang Anda tentukan,

dan melakukan satu atau beberapa tindakan berdasarkan nilai metrik relatif terhadap ambang batas tertentu selama sejumlah periode waktu. Tindakan ini adalah notitifkasi yang dikirim ke topik Amazon SNS atau kebijakan Auto Scaling. Alarm memicu tindakan hanya untuk status dengan perubahan berkelanjutan saja. CloudWatchalarm tidak memanggil tindakan hanya karena mereka berada dalam keadaan tertentu; negara harus telah berubah dan dipertahankan untuk sejumlah periode tertentu.

Menyiapkan CloudWatch logging untuk REST API di API Gateway

Untuk membantu men-debug masalah yang terkait dengan eksekusi permintaan atau akses klien ke API Anda, Anda dapat mengaktifkan Amazon CloudWatch Logs untuk mencatat panggilan API. Untuk informasi lebih lanjut tentang CloudWatch, lihatthe section called “Metrik CloudWatch”.

CloudWatch format log untuk API Gateway

Ada dua jenis log masuk API CloudWatch: logging eksekusi dan logging akses. Dalam pencatatan eksekusi, API Gateway mengelola CloudWatch Log. Prosesnya mencakup pembuatan grup log dan aliran log, dan pelaporan ke aliran log permintaan dan tanggapan pemanggil apa pun.

Data yang dicatat mencakup kesalahan atau jejak eksekusi (seperti nilai parameter permintaan atau respons atau muatan), data yang digunakan oleh otorisasi Lambda (sebelumnya dikenal sebagai otorisasi khusus), apakah kunci API diperlukan, apakah paket penggunaan diaktifkan, dan sebagainya.

Saat Anda menerapkan API, API Gateway membuat grup log dan aliran log di bawah grup log. Grup log diberi nama mengikuti `API-Gateway-Execution-Logs_{rest-api-id}/{stage_name}` format. Dalam setiap grup log, log dibagi lagi menjadi aliran log, yang diurutkan berdasarkan Waktu Peristiwa Terakhir saat data yang dicatat dilaporkan.

Dalam pencatatan akses, Anda, sebagai pengembang API, ingin mencatat siapa yang telah mengakses API Anda dan bagaimana pemanggil mengakses API. Anda dapat membuat grup log Anda sendiri atau memilih grup log yang sudah ada yang dapat dikelola oleh API Gateway. Untuk menentukan rincian akses, Anda memilih `$context`variabel (dinyatakan dalam format pilihan Anda) dan memilih grup log sebagai tujuan. Format log akses harus menyertakan setidaknya `$context.requestId` atau`$context.extendedRequestId`. Sebagai praktik terbaik, sertakan `$context.requestId` dan `$context.extendedRequestId` dalam format log Anda. `$context.requestId`mencatat nilai di `x-amzn-RequestId` header. Klien dapat mengganti nilai di `x-amzn-RequestId` header dengan nilai dalam format pengenal unik universal (UUID). API Gateway mengembalikan ID permintaan ini di header `x-amzn-RequestId` respons. API Gateway menggantikan ID permintaan yang diganti yang tidak dalam format UUID dengan log akses Anda.

*UUID*_REPLACED_INVALID_REQUEST_ID `$context.extendedRequestId`adalah ID unik yang dihasilkan API Gateway. API Gateway mengembalikan ID permintaan ini di header `x-amz-apigw-id` respons. Pemanggil API tidak dapat memberikan atau mengganti ID permintaan ini. Untuk informasi selengkapnya, lihat the section called "`$context`Variabel untuk model data, otorisasi, templat pemetaan, dan CloudWatch pencatatan akses".

> **Note**
>
> Hanya `$context` variabel yang didukung (tidak`$input`, dan seterusnya).

Pilih format log yang juga diadopsi oleh backend analitik Anda, seperti Common Log Format (CLF), JSON, XHTML, atau CSV. Anda kemudian dapat memasukkan log akses ke sana secara langsung agar metrik Anda dihitung dan dirender. Untuk menentukan format log, atur grup log ARN pada properti accessLogSettings/destinationArn di atas panggung. Anda dapat memperoleh grup log ARN di CloudWatch konsol, asalkan kolom ARN dipilih untuk ditampilkan. Untuk menentukan format log akses, atur format yang dipilih pada properti accessLogSetting/format di panggung.

Contoh beberapa format log akses yang umum digunakan ditampilkan di konsol API Gateway dan dicantumkan sebagai berikut.

- `CLF`(Format Log Umum):

```
$context.identity.sourceIp $context.identity.caller  \
$context.identity.user [$context.requestTime] \
"$context.httpMethod $context.resourcePath $context.protocol" \
$context.status $context.responseLength $context.requestId $context.extendedRequestId
```

  Karakter kelanjutan (\) dimaksudkan sebagai alat bantu visual. Format log harus satu baris. Anda dapat menambahkan karakter baris baru (\n) di akhir format log untuk menyertakan baris baru di akhir setiap entri log.

- `JSON`:

```
{ "requestId":"$context.requestId", \
  "extendedRequestId":"$context.extendedRequestId", \
  "ip": "$context.identity.sourceIp", \
  "caller":"$context.identity.caller", \
  "user":"$context.identity.user", \
  "requestTime":"$context.requestTime", \
```

```
  "httpMethod":"$context.httpMethod", \
  "resourcePath":"$context.resourcePath", \
  "status":"$context.status", \
  "protocol":"$context.protocol", \
  "responseLength":"$context.responseLength" \
}
```

Karakter kelanjutan (\) dimaksudkan sebagai alat bantu visual. Format log harus satu baris. Anda dapat menambahkan karakter baris baru (\n) di akhir format log untuk menyertakan baris baru di akhir setiap entri log.

- XML:

```
<request id="$context.requestId"> \
    <extendedRequestId>$context.extendedRequestId</extendedRequestId>
    <ip>$context.identity.sourceIp</ip> \
    <caller>$context.identity.caller</caller> \
    <user>$context.identity.user</user> \
    <requestTime>$context.requestTime</requestTime> \
    <httpMethod>$context.httpMethod</httpMethod> \
    <resourcePath>$context.resourcePath</resourcePath> \
    <status>$context.status</status> \
    <protocol>$context.protocol</protocol> \
    <responseLength>$context.responseLength</responseLength> \
</request>
```

Karakter kelanjutan (\) dimaksudkan sebagai alat bantu visual. Format log harus satu baris. Anda dapat menambahkan karakter baris baru (\n) di akhir format log untuk menyertakan baris baru di akhir setiap entri log.

- CSV(nilai yang dipisahkan koma):

```
$context.identity.sourceIp,$context.identity.caller,\
$context.identity.user,$context.requestTime,$context.httpMethod,\
$context.resourcePath,$context.protocol,$context.status,\
$context.responseLength,$context.requestId,$context.extendedRequestId
```

Karakter kelanjutan (\) dimaksudkan sebagai alat bantu visual. Format log harus satu baris. Anda dapat menambahkan karakter baris baru (\n) di akhir format log untuk menyertakan baris baru di akhir setiap entri log.

Izin untuk CloudWatch pencatatan

Untuk mengaktifkan CloudWatch Log, Anda harus memberikan izin API Gateway untuk membaca dan menulis log CloudWatch untuk akun Anda. Kebijakan `AmazonAPIGatewayPushToCloudWatchLogs` terkelola (dengan ARN`arn:aws:iam::aws:policy/service-role/AmazonAPIGatewayPushToCloudWatchLogs`) memiliki semua izin yang diperlukan:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents",
                "logs:GetLogEvents",
                "logs:FilterLogEvents"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> API Gateway memanggil AWS Security Token Service untuk mengambil peran IAM, jadi pastikan itu AWS STS diaktifkan untuk Wilayah. Untuk informasi selengkapnya, lihat Mengelola AWS STS di suatu AWS Wilayah.

Untuk memberikan izin ini ke akun Anda, buat peran IAM `apigateway.amazonaws.com` sebagai entitas tepercaya, lampirkan kebijakan sebelumnya ke peran IAM, dan tetapkan peran IAM ARN di properti Arn di Akun Anda. cloudWatchRole Anda harus mengatur properti cloudWatchRoleArn secara terpisah untuk setiap AWS Wilayah di mana Anda ingin mengaktifkan CloudWatch Log.

Jika Anda menerima kesalahan saat menyetel ARN peran IAM, periksa pengaturan akun AWS Security Token Service Anda untuk memastikan AWS STS bahwa diaktifkan di Wilayah yang Anda gunakan. Untuk informasi selengkapnya tentang mengaktifkanAWS STS, lihat Mengelola AWS STS di AWS Wilayah di Panduan Pengguna IAM.

Siapkan pencatatan CloudWatch API menggunakan konsol API Gateway

Untuk menyiapkan pencatatan CloudWatch API, Anda harus menerapkan API ke sebuah panggung. Anda juga harus mengonfigurasi ARN peran CloudWatch Log yang sesuai untuk akun Anda.

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pada panel navigasi utama, pilih Pengaturan, lalu di bawah Logging, pilih Edit.
3. Untuk ARN peran CloudWatch log, masukkan ARN dari peran IAM dengan izin yang sesuai. Anda perlu melakukan ini sekali untuk setiap Akun AWS yang membuat API menggunakan API Gateway.
4. Di panel navigasi utama, pilih API, lalu lakukan salah satu hal berikut:
   a. Pilih API yang ada, lalu pilih panggung.
   b. Buat API, lalu terapkan ke panggung.
5. Di panel navigasi utama, pilih Tahapan.
6. Di bagian Log dan penelusuran, pilih Edit.
7. Untuk mengaktifkan pencatatan eksekusi:
   a. Pilih level logging dari menu dropdown CloudWatch Log.

      > **Warning**
      >
      > Log permintaan dan respons lengkap dapat berguna untuk memecahkan masalah API, tetapi dapat mengakibatkan pencatatan data sensitif. Kami menyarankan Anda untuk tidak menggunakan log permintaan dan respons lengkap untuk API produksi.

   b. Jika diinginkan, pilih Metrik terperinci untuk mengaktifkan CloudWatch metrik terperinci.

   Untuk informasi selengkapnya tentang CloudWatch metrik, lihatthe section called “Metrik CloudWatch”.
8. Untuk mengaktifkan pencatatan akses:

a. Aktifkan Pencatatan akses khusus.

b. Untuk akses log tujuan ARN, masukkan ARN dari grup log. Format ARN adalah. `arn:aws:logs:{region}:{account-id}:log-group:log-group-name`

c. Untuk Format Log, masukkan format log. Anda dapat memilih CLF, JSON, XHTML, atau CSV. Untuk mempelajari lebih lanjut tentang contoh format log, lihatthe section called "CloudWatch format log untuk API Gateway".

9. Pilih Save changes (Simpan perubahan).

> **Note**
>
> Anda dapat mengaktifkan pencatatan eksekusi dan mengakses logging secara independen satu sama lain.

API Gateway sekarang siap untuk mencatat permintaan ke API Anda. Anda tidak perlu menerapkan ulang API saat memperbarui pengaturan panggung, log, atau variabel tahap.

Siapkan pencatatan CloudWatch API menggunakan AWS CloudFormation

Gunakan contoh AWS CloudFormation template berikut untuk membuat grup CloudWatch log Amazon Logs dan mengonfigurasi eksekusi dan mengakses logging untuk sebuah panggung. Untuk mengaktifkan CloudWatch Log, Anda harus memberikan izin API Gateway untuk membaca dan menulis log CloudWatch untuk akun Anda. Untuk mempelajari selengkapnya, lihat Mengaitkan akun dengan peran IAM di Panduan AWS CloudFormation Pengguna.

```
TestStage:
  Type: AWS::ApiGateway::Stage
  Properties:
    StageName: test
    RestApiId: !Ref MyAPI
    DeploymentId: !Ref Deployment
    Description: "test stage description"
    MethodSettings:
      - ResourcePath: "/*"
        HttpMethod: "*"
        LoggingLevel: INFO
    AccessLogSetting:
      DestinationArn: !GetAtt MyLogGroup.Arn
```

```
        Format: $context.extendedRequestId $context.identity.sourceIp
$context.identity.caller $context.identity.user [$context.requestTime]
"$context.httpMethod $context.resourcePath $context.protocol" $context.status
$context.responseLength $context.requestId
 MyLogGroup:
   Type: AWS::Logs::LogGroup
   Properties:
     LogGroupName: !Join
       - '-'
       - - !Ref MyAPI
         - access-logs
```

Pencatatan panggilan API ke Amazon Data Firehose

Untuk membantu masalah debug yang terkait dengan akses klien ke API Anda, Anda dapat mencatat panggilan API ke Amazon Data Firehose. Untuk informasi selengkapnya tentang Firehose, lihat Apa itu Amazon Data Firehose? .

Untuk pencatatan akses, Anda hanya dapat mengaktifkan CloudWatch atau Firehose—Anda tidak dapat mengaktifkan keduanya. Namun, Anda dapat mengaktifkan CloudWatch pencatatan eksekusi dan Firehose untuk pencatatan akses.

Topik

- Format log Firehose untuk API Gateway
- Izin untuk pencatatan Firehose
- Mengatur pencatatan akses Firehose dengan menggunakan konsol API Gateway

Format log Firehose untuk API Gateway

Firehose logging menggunakan format yang sama dengan logging. CloudWatch

Izin untuk pencatatan Firehose

Saat pencatatan akses Firehose diaktifkan di panggung, API Gateway akan membuat peran terkait layanan di akun Anda jika peran tersebut belum ada. Peran tersebut dinamai `AWSServiceRoleForAPIGateway` dan memiliki kebijakan `APIGatewayServiceRolePolicy` terkelola yang melekat padanya. Untuk informasi selengkapnya tentang peran terkait layanan, lihat Menggunakan Peran Tertaut Layanan.

> **Note**
>
> Nama aliran Firehose Anda harus. `amazon-apigateway-{your-stream-name}`

Mengatur pencatatan akses Firehose dengan menggunakan konsol API Gateway

Untuk menyiapkan pencatatan API, Anda harus menerapkan API ke sebuah panggung. Anda juga harus membuat aliran Firehose.

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Lakukan salah satu dari cara berikut:
   a. Pilih API yang ada, lalu pilih panggung.
   b. Buat API dan terapkan ke panggung.
3. Di panel navigasi utama, pilih Tahapan.
4. Di bagian Log dan penelusuran, pilih Edit.
5. Untuk mengaktifkan pencatatan akses ke aliran Firehose:
   a. Aktifkan Pencatatan akses khusus.
   b. Untuk akses log tujuan ARN, masukkan ARN dari aliran Firehose. Format ARN adalah. `arn:aws:firehose:{region}:{account-id}:deliverystream/amazon-apigateway-{your-stream-name}`

      > **Note**
      >
      > Nama aliran Firehose Anda harus. `amazon-apigateway-{your-stream-name}`

   c. Untuk format Log, masukkan format log. Anda dapat memilih CLF, JSON, XHTML, atau CSV. Untuk mempelajari lebih lanjut tentang contoh format log, lihatthe section called "CloudWatch format log untuk API Gateway".
6. Pilih Simpan perubahan.

API Gateway sekarang siap untuk mencatat permintaan ke API Anda ke Firehose. Anda tidak perlu menerapkan ulang API saat memperbarui pengaturan panggung, log, atau variabel tahap.

Menelusuri permintaan pengguna ke REST API menggunakan X-Ray

Anda dapat menggunakanAWS X-Rayuntuk menelusuri dan menganalisis permintaan pengguna saat mereka melakukan perjalanan melalui API Amazon API Gateway ke layanan yang mendasarinya. API Gateway mendukung penelusuran X-Ray untuk semua tipe titik akhir API Gateway REST: Regional, tepi-dioptimalkan, dan pribadi. Anda dapat menggunakan X-Ray dengan Amazon API Gateway di semua Wilayah AWS yang menyediakan X-Ray.

Karena X-Ray memberi Anda tampilan end-to-end atas seluruh permintaan, Anda dapat menganalisis latensi di API dan layanan backend mereka. Anda dapat menggunakan peta layanan X-Ray untuk melihat latensi seluruh permintaan dan layanan hilir yang terintegrasi dengan X-Ray. Anda juga dapat mengonfigurasi aturan pengambilan sampel untuk memberi tahu X-Ray mengenai permintaan yang dicatat dan tingkat pengambilan sampel, sesuai kriteria yang Anda tentukan.

Jika Anda memanggil API Gateway API dari layanan yang sudah dilacak, API Gateway meneruskan pelacakan, meskipun pelacakan X-Ray tidak diaktifkan di API.

Anda dapat mengaktifkan X-Ray untuk tahap API dengan menggunakan konsol API Gateway, atau dengan menggunakan API Gateway API atau CLI.

Topik

- Menyiapkan AWS X-Ray dengan API API Gateway REST
- MenggunakanAWS X-Raypeta layanan dan tampilan pelacakan dengan API Gateway
- MengonfigurasiaturanAWS X-Ray pengambilan sampel untuk API Gateway API
- MemahamiAWS X-Raypelacakan untuk Amazon API Gateway API

Menyiapkan AWS X-Ray dengan API API Gateway REST

Di bagian ini Anda dapat menemukan informasi terperinci tentang cara mengatur AWS X-Raydengan API API Gateway REST API.

Topik

- Mode penelusuran X-Ray untuk API Gateway
- Izin untuk penelusuran X-Ray
- Mengaktifkan penelusuran X-Ray di konsol API Gateway
- Mengaktifkan AWS X-Ray penelusuran menggunakan API Gateway CLI

Mode penelusuran X-Ray untuk API Gateway

Jalur permintaan melalui aplikasi Anda dilacak dengan ID jejak. Jejak mengumpulkan semua segmen yang dihasilkan oleh satu permintaan, biasanya permintaan HTTP `GET` atau `POST` permintaan.

Ada dua mode penelusuran untuk API Gateway API:

- Pasif: Ini adalah pengaturan default jika Anda belum mengaktifkan penelusuran X-Ray pada tahap API. Pendekatan ini berarti API Gateway API hanya dilacak jika X-Ray telah diaktifkan pada layanan upstream.
- Aktif: Saat tahap API Gateway API memiliki pengaturan ini, API Gateway secara otomatis mengambil sampel permintaan pemanggilan API, berdasarkan algoritma pengambilan sampel yang ditentukan oleh X-Ray.

  Saat penelusuran aktif diaktifkan di panggung, API Gateway akan membuat peran terkait layanan di akun Anda, jika peran tersebut belum ada. Peran tersebut diberi nama `AWSServiceRoleForAPIGateway` dan akan memiliki kebijakan `APIGatewayServiceRolePolicy` terkelola yang melekat padanya. Untuk informasi selengkapnya tentang peran terkait layanan, lihat Menggunakan Peran Tertaut Layanan.

  > **Note**
  >
  > X-Ray menerapkan algoritma sampling untuk memastikan bahwa penelusuran efisien, sambil tetap memberikan sampel representatif dari permintaan yang diterima API Anda. Algoritma pengambilan sampel default adalah 1 permintaan per detik, dengan 5 persen permintaan sampel melewati batas itu.

Anda dapat mengubah mode penelusuran untuk API Anda dengan menggunakan konsol manajemen API Gateway, API Gateway CLI, atau AWS SDK.

Izin untuk penelusuran X-Ray

Saat Anda mengaktifkan penelusuran X-Ray di panggung, API Gateway akan membuat peran terkait layanan di akun Anda, jika peran tersebut belum ada. Peran tersebut diberi nama `AWSServiceRoleForAPIGateway` dan akan memiliki kebijakan `APIGatewayServiceRolePolicy` terkelola yang melekat padanya. Untuk informasi selengkapnya tentang peran terkait layanan, lihat Menggunakan Peran Tertaut Layanan.

Mengaktifkan penelusuran X-Ray di konsol API Gateway

Anda dapat menggunakan konsol Amazon API Gateway untuk mengaktifkan penelusuran aktif pada tahap API.

Langkah-langkah ini mengasumsikan bahwa Anda telah menerapkan API ke suatu panggung.

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API Anda, lalu di panel navigasi utama, pilih Tahapan.
3. Di panel Tahapan, pilih panggung.
4. Di bagian Log dan penelusuran, pilih Edit.
5. Untuk mengaktifkan penelusuran X-Ray aktif, pilih penelusuran X-Ray untuk mengaktifkan penelusuran X-Ray.
6. Pilih Save changes (Simpan perubahan).

Setelah mengaktifkan X-Ray untuk tahap API, Anda dapat menggunakan konsol manajemen X-Ray untuk melihat jejak dan peta layanan.

Mengaktifkan AWS X-Ray penelusuran menggunakan API Gateway CLI

Untuk menggunakan AWS CLI cara mengaktifkan penelusuran X-Ray aktif untuk tahap API saat Anda membuat stage, panggil `create-stage`perintah seperti pada contoh berikut:

```
aws apigateway create-stage \
    --rest-api-id {rest-api-id} \
    --stage-name {stage-name} \
    --deployment-id {deployment-id} \
    --region {region} \
    --tracing-enabled=true
```

Berikut ini adalah contoh output untuk pemanggilan yang berhasil:

```
{
    "tracingEnabled": true,
    "stageName": {stage-name},
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": {deployment-id},
```

```
    "lastUpdatedDate": 1533849811,
    "createdDate": 1533849811,
    "methodSettings": {}
}
```

Untuk menggunakan AWS CLI cara menonaktifkan penelusuran X-Ray aktif untuk tahap API saat Anda membuat stage, panggil `create-stage`perintah seperti pada contoh berikut:

```
aws apigateway create-stage \
    --rest-api-id {rest-api-id} \
    --stage-name {stage-name} \
    --deployment-id {deployment-id} \
    --region {region} \
    --tracing-enabled=false
```

Berikut ini adalah contoh output untuk pemanggilan yang berhasil:

```
{
    "tracingEnabled": false,
    "stageName": {stage-name},
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": {deployment-id},
    "lastUpdatedDate": 1533849811,
    "createdDate": 1533849811,
    "methodSettings": {}
}
```

Untuk menggunakan fitur AWS CLI untuk mengaktifkan penelusuran X-Ray aktif untuk API yang sudah di-deploy, panggil `update-stage`perintah sebagai berikut:

```
aws apigateway update-stage \
    --rest-api-id {rest-api-id} \
    --stage-name {stage-name} \
    --patch-operations op=replace,path=/tracingEnabled,value=true
```

Untuk menggunakan AWS CLI untuk menonaktifkan penelusuran X-Ray aktif untuk API yang sudah digunakan, panggil `update-stage`perintah seperti pada contoh berikut:

```
aws apigateway update-stage \
```

```
    --rest-api-id {rest-api-id} \
    --stage-name {stage-name} \
    --region {region} \
    --patch-operations op=replace,path=/tracingEnabled,value=false
```

Berikut ini adalah contoh output untuk pemanggilan yang berhasil:

```
{
    "tracingEnabled": false,
    "stageName": {stage-name},
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": {deployment-id},
    "lastUpdatedDate": 1533850033,
    "createdDate": 1533849811,
    "methodSettings": {}
}
```

Setelah Anda mengaktifkan X-Ray untuk tahap API Anda, gunakan X-Ray CLI untuk mengambil informasi jejak. Untuk informasi selengkapnya, lihat Menggunakan AWS X-Ray API dengan AWS CLI.

MenggunakanAWS X-Raypeta layanan dan tampilan pelacakan dengan API Gateway

Pada bagian ini Anda dapat menemukan informasi detail tentang bagaimana menggunakanAWS X-Raypeta layanan dan melacak tampilan dengan API Gateway.

Untuk informasi rinci tentang peta layanan dan tampilan pelacakan, dan cara menafsirkannya, lihatAWS X-RayKonsol.

Topik

- Peta layanan X-Ray
- Tampilan pelacakan X-Ray

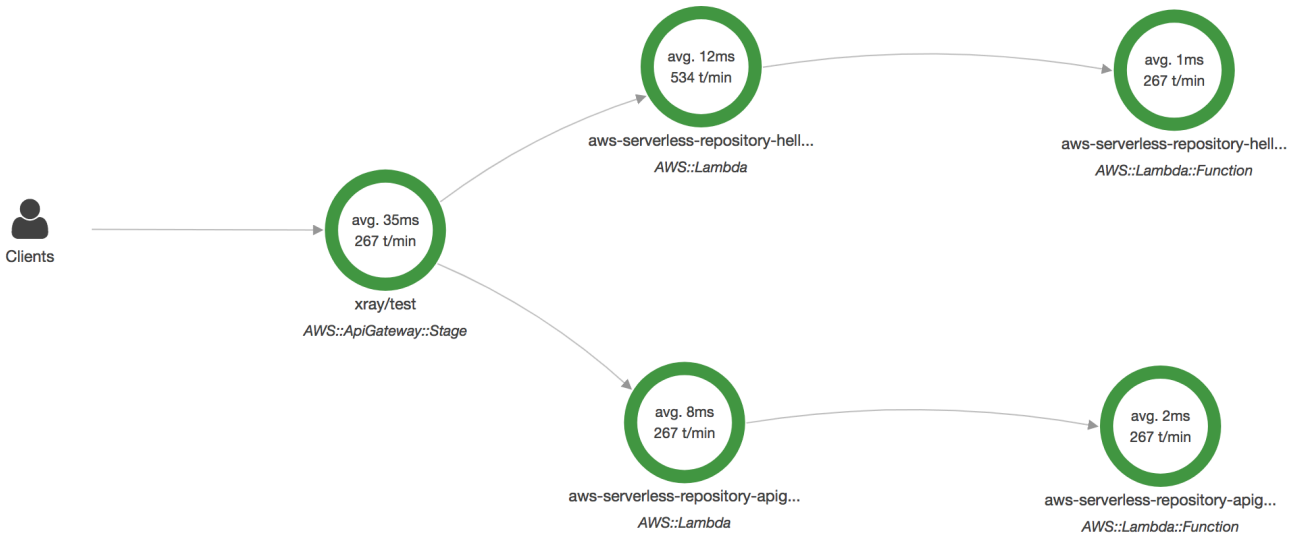
Peta layanan X-Ray

AWS X-Raypeta layanan menunjukkan informasi tentang API Anda dan semua layanan hilir. Ketika X-Ray diaktifkan untuk tahap API di API Gateway, Anda akan melihat node di peta layanan yang berisi informasi tentang keseluruhan waktu yang dihabiskan di layanan API Gateway. Anda bisa

mendapatkan informasi rinci tentang status respons dan histogram waktu respons API untuk jangka waktu yang dipilih. Untuk API yang terintegrasi denganAWSlayanan sepertiAWS Lambdadan Amazon DynamoDB, Anda akan melihat lebih banyak node yang menyediakan metrik kinerja yang terkait dengan layanan tersebut. Akan ada peta layanan untuk setiap tahap API.

Contoh berikut menunjukkan Peta layanan untuk`test`tahap API yang disebut`xray`. API ini memiliki integrasi Lambda dengan fungsi otorisasi Lambda dan fungsi backend Lambda. Node mewakili layanan API Gateway, layanan Lambda, dan dua fungsi Lambda.

Untuk penjelasan detail tentang struktur Peta layanan, lihatMelihat Peta Layanan.



Dari peta layanan, Anda dapat memperbesar untuk melihat tampilan jejak tahap API Anda. Pelacakan akan menampilkan informasi mendalam mengenai API Anda, direpresentasikan sebagai segmen dan subsegmen. Misalnya, pelacakan untuk peta layanan yang ditunjukkan di atas akan mencakup segmen untuk layanan Lambda dan fungsi Lambda. Untuk informasi selengkapnya, lihatLambda sebagaiAWS X-RayJejak.

Jika Anda memilih simpul atau edge pada Peta layanan X-Ray menunjukkan histogram distribusi latensi. Anda dapat menggunakan histogram latensi untuk melihat berapa lama waktu yang dibutuhkan layanan untuk menyelesaikan permintaannya. Berikut ini adalah histogram dari tahap API Gateway bernama`xray/test`di peta layanan sebelumnya. Untuk penjelasan detail tentang histogram distribusi, lihatMenggunakan Histogram Latency diAWS X-RayKonsol.

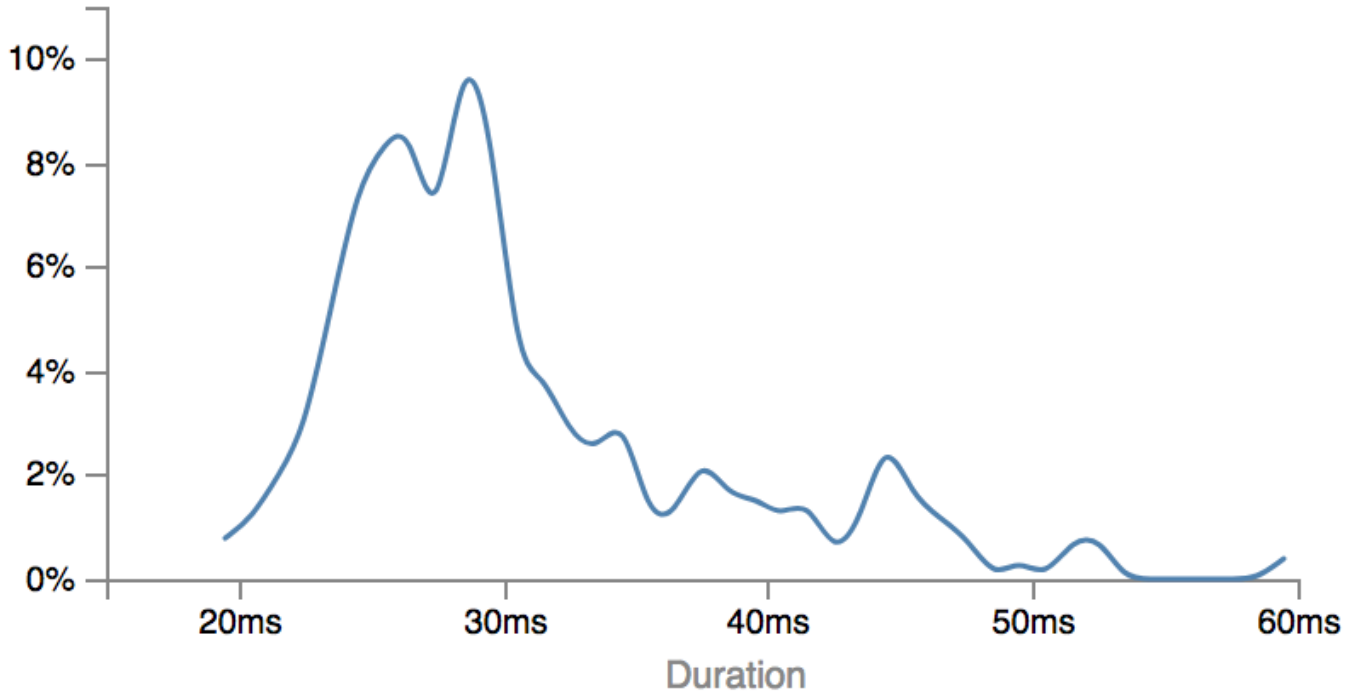
Service details

Name: xray/test

Type: AWS::ApiGateway::Stage

Response distribution

Click and drag to select an area to zoom in on or use as a latency filter when viewing traces.



Response status

Choose response statuses to add to the filter when viewing traces.

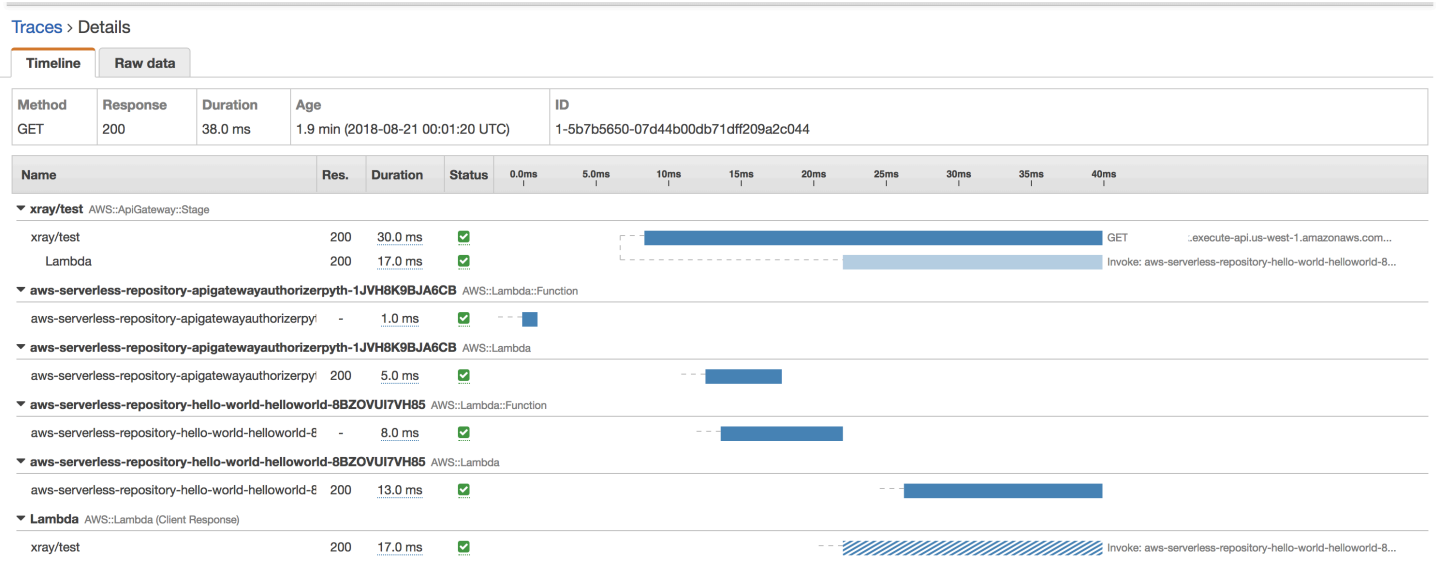
OK: 100% Error: 0%

Fault: 0% Throttle: 0%

Tampilan pelacakan X-Ray

Diagram berikut menunjukkan tampilan jejak yang dihasilkan untuk contoh API yang dijelaskan di atas, dengan fungsi backend Lambda dan fungsi otorisasi Lambda. Permintaan metode API yang berhasil ditampilkan dengan kode respons 200.

Untuk penjelasan detail tentang tampilan pelacakan, lihatMelihat Jejak.



Mengonfigurasi aturanAWS X-Ray pengambilan sampel untuk API Gateway API

Anda dapat menggunakanAWS X-Ray konsol atau SDK untuk mengonfigurasi aturan pengambilan sampel untuk Amazon API Gateway. Aturan sampling menentukan permintaan mana yang harus direkam X-Ray untuk API Anda. Dengan menyesuaikan aturan pengambilan sampel, Anda dapat mengontrol jumlah data yang Anda catat, dan mengubah perilaku pengambilan sampel dengan cepat tanpa mengubah atau men-deploy ulang kode Anda.

Sebelum menentukan aturan pengambilan sampel X-Ray, baca topik berikut dalam Panduan Developer X-Ray:

- Mengonfigurasi Aturan Sampling diAWS X-Ray Konsol
- Penggunaan aturan pengambilan dengan API X-Ray

Topik

- Nilai opsi aturan sampling X-Ray untuk API Gateway API

- X-Ray

Nilai opsi aturan sampling X-Ray untuk API Gateway API

Opsi pengambilan sampel X-Ray berikut relevan untuk API Gateway. Nilai string dapat menggunakan wildcard untuk mencocokkan satu karakter (?) atau nol atau lebih karakter (*). Untuk detail selengkapnya, termasuk penjelasan mendetail tentang cara pengaturan Reservoir dan Rate digunakan, Mengonfigurasi Aturan Sampling diAWS X-Ray Konsol.

- Nama aturan (string) — Nama unik untuk aturan.
- Prioritas (bilangan bulat antara 1 hingga 9999) — Prioritas aturan pengambilan sampel. Layanan mengevaluasi aturan dalam urutan prioritas naik, dan membuat keputusan pengambilan sampel dengan aturan pertama yang cocok.
- Reservoir (bilangan bulat bukan negatif) — Jumlah tetap permintaan yang cocok dengan instrumen per detik, sebelum menerapkan tingkat tetap. Reservoir tidak digunakan secara langsung oleh layanan, tetapi berlaku untuk semua layanan yang menggunakan aturan secara kolektif.
- Tingkat (angka antara 0 hingga 100) — Persentase permintaan yang cocok dengan instrumen, setelah reservoir habis.
- Nama layanan (string) - Nama tahap API, dalam formulir***{api-name}*****/*****{stage-name}***. Misalnya, jika Anda menerapkan API PetStoresampel ke tahap bernama`test`, nilai nama Layanan yang akan ditentukan dalam aturan pengambilan sampel akan menjadi**`pets/test`**.
- Jenis layanan (string) - Untuk API Gateway API, baik**`AWS::ApiGateway::Stage`** atau**`AWS::ApiGateway::*`** dapat ditentukan.
- Host (string) — Nama host dari header host HTTP. Atur ini**`*`** agar sesuai dengan semua nama host. Atau Anda dapat menentukan nama host penuh atau sebagian untuk dicocokkan, misalnya,**`api.example.com`** atau**`*.example.com`**.
- Sumber daya ARN (string) - ARN tahap API, misalnya,**`arn:aws:apigateway:`*`region`*`::/restapis/`*`api-id`*`/stages/`*`stage-name`**.

  Nama panggung dapat diperoleh dari konsol atau API Gateway CLI atau API. Untuk informasi selengkapnya tentang format ARN, lihat Referensi Umum Amazon Web Services.
- Metode HTTP (string) - Metode yang akan diambil sampel, misalnya,**`GET`**.
- Jalur URL (string) - Jalur URL permintaan.

- (opsional) Atribut (kunci dan nilai) - Header dari permintaan HTTP asli, misalnya,**`Connection`**,**`Content-Length`**, atau**`Content-Type`**. Setiap nilai atribut dapat memiliki panjang hingga 32 karakter.

X-Ray

Contoh aturan sampling #1

Aturan ini mengambil sampel semua`GET` permintaan untuk`testxray` API di`test` panggung.

- Nama aturan —**`test-sampling`**
- Prioritas —**`17`**
- Ukuran waduk -**`10`**
- Tingkat tetap -**`10`**
- Nama layanan —**`testxray/test`**
- Jenis layanan -**`AWS::ApiGateway::Stage`**
- Metode HTTP -**`GET`**
- Sumber daya ARN —**`*`**
- Tuan rumah —**`*`**

Contoh aturan sampling #2

Aturan ini mengambil sampel semua permintaan untuk`testxray` API di`prod` panggung.

- Nama aturan —**`prod-sampling`**
- Prioritas —**`478`**
- Ukuran waduk -**`1`**
- Tingkat tetap -**`60`**
- Nama layanan —**`testxray/prod`**
- Jenis layanan -**`AWS::ApiGateway::Stage`**
- Metode HTTP -**`*`**
- Sumber daya ARN —**`*`**
- Tuan rumah —**`*`**
- Atribut -**`{}`**

MemahamiAWS X-Raypelacakan untuk Amazon API Gateway API

Bagian ini membahasAWS X-Raymelacak segmen, subsegmen, dan bidang pelacakan lainnya untuk API Amazon API Gateway.

Sebelum membaca bagian ini, tinjau topik berikut dalam Panduan Developer X-Ray:

- AWS X-RayKonsol
- AWS X-RayDokumen Segment
- Konsep X-Ray

Topik

- Contoh pelacakan objek untuk API Gateway
- Memahami pelacakan

Contoh pelacakan objek untuk API Gateway

Bagian ini membahas beberapa objek yang mungkin Anda lihat dalam pelacakan API Gateway API.

Anotasi

Anotasi dapat muncul di segmen dan subsegmen. Mereka digunakan sebagai ekspresi penyaringan dalam aturan sampling untuk menyaring jejak. Untuk informasi selengkapnya, lihatMengkonfigurasi Aturan Sampling diAWS X-RayKonsol.

Berikut ini adalah contoh`annotations`objek, di mana tahap API diidentifikasi oleh ID API dan nama tahap API:

```
"annotations": {
    "aws:api_id": "a1b2c3d4e5",
    "aws:api_stage": "dev"
}
```

AWSData sumber daya

Parameter`aws`objek hanya muncul di segmen. Berikut ini adalah contoh`aws`objek yang cocok dengan aturan default sampling. Untuk penjelasan mendalam tentang aturan pengambilan sampel, lihatMengkonfigurasi Aturan Sampling diAWS X-RayKonsol.

```
"aws": {
    "xray": {
        "sampling_rule_name": "Default"
    },
    "api_gateway": {
        "account_id": "123412341234",
        "rest_api_id": "a1b2c3d4e5",
        "stage": "dev",
        "request_id": "a1b2c3d4-a1b2-a1b2-a1b2-a1b2c3d4e5f6"
    }
}
```

Memahami pelacakan

Berikut ini adalah segmen jejak untuk tahap API Gateway. Untuk penjelasan rinci tentang bidang yang membentuk segmen jejak, lihatAWS X-RayDokumen SegmentdiAWS X-RayPanduan Developer.

```
        {
            "Document": {
                "id": "a1b2c3d4a1b2c3d4",
                "name": "testxray/dev",
                "start_time": 1533928226.229,
                "end_time": 1533928226.614,
                "metadata": {
                    "default": {
                        "extended_request_id": "abcde12345abcde=",
                        "request_id": "a1b2c3d4-a1b2-a1b2-a1b2-a1b2c3d4e5f6"
                    }
                },
                "http": {
                    "request": {
                        "url": "https://example.com/dev?
username=demo&message=hellofromdemo/",
                        "method": "GET",
                        "client_ip": "192.0.2.0",
                        "x_forwarded_for": true
                    },
                    "response": {
                        "status": 200,
                        "content_length": 0
                    }
                },
```

```
                "aws": {
                    "xray": {
                        "sampling_rule_name": "Default"
                    },
                    "api_gateway": {
                        "account_id": "123412341234",
                        "rest_api_id": "a1b2c3d4e5",
                        "stage": "dev",
                        "request_id": "a1b2c3d4-a1b2-a1b2-a1b2-a1b2c3d4e5f6"
                    }
                },
                "annotations": {
                    "aws:api_id": "a1b2c3d4e5",
                    "aws:api_stage": "dev"
                },
                "trace_id": "1-a1b2c3d4-a1b2c3d4a1b2c3d4a1b2c3d4",
                "origin": "AWS::ApiGateway::Stage",
                "resource_arn": "arn:aws:apigateway:us-east-1::/restapis/a1b2c3d4e5/
stages/dev",
                "subsegments": [
                    {
                        "id": "abcdefgh12345678",
                        "name": "Lambda",
                        "start_time": 1533928226.233,
                        "end_time": 1533928226.6130002,
                        "http": {
                            "request": {
                                "url": "https://example.com/2015-03-31/functions/
arn:aws:lambda:us-east-1:123412341234:function:xray123/invocations",
                                "method": "GET"
                            },
                            "response": {
                                "status": 200,
                                "content_length": 62
                            }
                        },
                        "aws": {
                            "function_name": "xray123",
                            "region": "us-east-1",
                            "operation": "Invoke",
                            "resource_names": [
                                "xray123"
                            ]
                        },
```

```
                    "namespace": "aws"
                }
            ]
        },
        "Id": "a1b2c3d4a1b2c3d4"
    }
```

Bekerja dengan HTTP API

REST API dan HTTP API keduanya merupakan produk RESTful API. REST API mendukung lebih banyak fitur daripada API HTTP, sedangkan API HTTP dirancang dengan fitur minimal sehingga dapat ditawarkan dengan harga lebih murah. Untuk informasi selengkapnya, lihat the section called “Memilih antara REST API dan HTTP API ”.

Anda dapat menggunakan API HTTP untuk mengirim permintaan ke AWS Lambda fungsi atau ke titik akhir HTTP yang dapat dirutekan. Misalnya, Anda dapat membuat API HTTP yang terintegrasi dengan fungsi Lambda di backend. Saat klien memanggil API Anda, API Gateway mengirimkan permintaan ke fungsi Lambda dan mengembalikan respons fungsi ke klien.

HTTP API mendukung otorisasi OpenID Connect dan OAuth 2.0. Mereka datang dengan dukungan bawaan untuk berbagi sumber daya lintas asal (CORS) dan penerapan otomatis.

Anda dapat membuat API HTTP menggunakan AWS Management ConsoleAWS CLI, APIAWS CloudFormation, atau SDK.

Topik

- Mengembangkan API HTTP di API Gateway
- Menerbitkan API HTTP untuk dipanggil pelanggan
- Melindungi API HTTP Anda
- Pemantauan HTTP API
- Pemecahan masalah dengan HTTP

Mengembangkan API HTTP di API Gateway

Bagian ini memberikan rincian tentang kemampuan API Gateway yang Anda butuhkan saat Anda mengembangkan API Gateway API Anda.

Saat Anda mengembangkan API Gateway API, Anda memutuskan sejumlah karakteristik API Anda. Karakteristik ini tergantung pada kasus penggunaan API Anda. Misalnya, Anda mungkin hanya mengizinkan klien tertentu untuk memanggil API Anda, atau mungkin Anda ingin tersedia untuk semua orang. Anda mungkin ingin panggilan API untuk menjalankan fungsi Lambda, membuat kueri database, atau memanggil aplikasi.

Topik

Membuat API HTTP

Untuk membuat API fungsional, Anda harus memiliki setidaknya satu rute, integrasi, tahap, dan penyebaran.

Contoh berikut ini menunjukkan cara membuat API denganAWS LambdaIntegrasi HTTP, rute, dan tahap default yang dikonfigurasi untuk men-deploy perubahan secara otomatis.

Panduan ini mengasumsikan bahwa Anda sudah terbiasa dengan API Gateway dan Lambda. Untuk panduan yang lebih rinci, lihatMulai.

Topik

Buat HTTP API dengan menggunakanAWS Management Console

1. BukaKonsol API Gateway.
2. Pilih Buat API.
3. Di bawahAPI HTTP, pilihMembangun.
4. MemiilihMenambahkan integrasi, dan kemudian memilihAWS Lambdafungsi atau masukkan endpoint HTTP.
5. UntukNama, masukkan nama untuk API Anda.
6. Pilih Periksa dan buat.

7. Pilih Create (Buat).

Sekarang API Anda siap untuk dipanggil. Anda dapat menguji API Anda dengan memasukkan URL pemanggilan di browser, atau dengan menggunakan Curl.

```
curl https://api-id.execute-api.us-east-2.amazonaws.com
```

Buat HTTP API dengan menggunakanAWSCLI

Anda dapat menggunakan pembuatan cepat untuk membuat API dengan integrasi Lambda atau HTTP, rute default catch-all, dan tahap default yang dikonfigurasi untuk men-deploy perubahan secara otomatis. Perintah berikut menggunakan quick create untuk membuat API yang terintegrasi dengan fungsi Lambda pada backend.

> **Note**
>
> Untuk memanggil integrasi Lambda, API Gateway harus memiliki izin yang diperlukan. Anda dapat menggunakan kebijakan berbasis sumber daya atau IAM role untuk memberikan izin API Gateway guna menjalankan fungsi Lambda. Untuk mempelajari lebih lanjut, lihatAWS Lambdalzindi AWS LambdaPanduan Pengembang.

Example

```
aws apigatewayv2 create-api --name my-api --protocol-type HTTP --target
 arn:aws:lambda:us-east-2:123456789012:function:function-name
```

Sekarang API Anda siap untuk dipanggil. Anda dapat menguji API Anda dengan memasukkan URL pemanggilan di browser, atau dengan menggunakan Curl.

```
curl https://api-id.execute-api.us-east-2.amazonaws.com
```

Bekerja dengan rute untuk HTTP API

Merutekan permintaan API yang masuk langsung ke sumber daya backend. Rute terdiri dari dua bagian: metode HTTP dan jalur sumber daya—misalnya,. `GET /pets` Anda dapat menentukan metode HTTP spesifik untuk rute Anda. Atau, Anda dapat menggunakan `ANY` metode untuk mencocokkan semua metode yang belum Anda tetapkan untuk sumber daya. Anda dapat membuat

`$default` rute yang bertindak sebagai tangkapan semua untuk permintaan yang tidak cocok dengan rute lain.

> **Note**
>
> API Gateway menerjemahkan parameter permintaan yang disandikan URL sebelum meneruskannya ke integrasi backend Anda.

Bekerja dengan variabel jalur

Anda dapat menggunakan variabel jalur di rute API HTTP.

Misalnya, `GET /pets/{petID}` rute menangkap `GET` permintaan yang dikirimkan klien. `https://api-id.execute-api.us-east-2.amazonaws.com/pets/6`

Variabel jalur serakah menangkap semua sumber daya anak dari suatu rute. Untuk membuat variabel jalur serakah, tambahkan + ke nama variabel—misalnya,. `{proxy+}` Variabel jalur serakah harus berada di ujung jalur sumber daya.

Bekerja dengan parameter string kueri

Secara default, API Gateway mengirimkan parameter string kueri ke integrasi backend Anda jika mereka disertakan dalam permintaan ke API HTTP.

Misalnya, ketika klien mengirim permintaan kehttps://*api-id*.execute-api.*us-east-2*.amazonaws.com/pets*?id=4&type=dog*, parameter string kueri `?id=4&type=dog` dikirim ke integrasi Anda.

Bekerja dengan **`$default`** rute

`$default`Rute menangkap permintaan yang tidak secara eksplisit cocok dengan rute lain di API Anda.

Saat `$default` rute menerima permintaan, API Gateway mengirimkan jalur permintaan lengkap ke integrasi. Misalnya, Anda dapat membuat API hanya dengan `$default` rute dan mengintegrasikannya pada `ANY` metode dengan titik akhir `https://petstore-demo-endpoint.execute-api.com` HTTP. Saat Anda mengirim permintaan kehttps://*api-id*.execute-api.*us-east-2*.amazonaws.com/store/checkout, API Gateway mengirimkan permintaan kehttps://petstore-demo-endpoint.execute-api.com/store/checkout.

Untuk mempelajari lebih lanjut tentang integrasi HTTP, lihatBekerja dengan integrasi proxy HTTP untuk API HTTP.

Permintaan API perutean

Saat klien mengirim permintaan API, API Gateway terlebih dahulu menentukan tahap mana yang akan merutekan permintaan tersebut. Jika permintaan secara eksplisit cocok dengan tahapan, API Gateway mengirimkan permintaan ke tahap tersebut. Jika tidak ada tahap yang sepenuhnya cocok dengan permintaan, API Gateway mengirimkan permintaan ke `$default` panggung. Jika tidak ada `$default` tahap, maka API kembali `{"message":"Not Found"}` dan tidak menghasilkan CloudWatch log.

Setelah memilih tahapan, API Gateway memilih rute. API Gateway memilih rute dengan kecocokan paling spesifik, menggunakan prioritas berikut:

1. Pertandingan penuh untuk rute dan metode.
2. Cocokkan rute dan metode dengan variabel jalur serakah (`{proxy+}`).
3. `$default`Rute.

Jika tidak ada rute yang cocok dengan permintaan, API Gateway `{"message":"Not Found"}` akan kembali ke klien.

Misalnya, pertimbangkan API dengan `$default` tahapan dan contoh rute berikut:

1. `GET /pets/dog/1`
2. `GET /pets/dog/{id}`
3. `GET /pets/{proxy+}`
4. `ANY /{proxy+}`
5. `$default`

Tabel berikut merangkum cara API Gateway merutekan permintaan ke rute contoh.

Permintaan	Rute yang dipilih	Penjelasan
`GET https://api-id.execute-`	`GET /pets/dog/1`	Permintaan sepenuhnya cocok dengan rute statis ini.

Permintaan	Rute yang dipilih	Penjelasan
`api. region.amazonaws.com/pets/dog/1`		
`GET https://api-id.execute-api. region.amazonaws.com/pets/dog/2`	`GET /pets/dog/{id}`	Permintaan sepenuhnya cocok dengan rute ini.
`GET https://api-id.execute-api. region.amazonaws.com/pets/cat/1`	`GET /pets/{proxy+}`	Permintaan tidak sepenuhnya cocok dengan rute. Rute dengan `GET` metode dan variabel jalur serakah menangkap permintaan ini.
`POST https://api-id.execute-api. region.amazonaws.com/test/5`	`ANY /{proxy+}`	`ANY`Metode ini cocok dengan semua metode yang belum Anda tetapkan untuk rute. Rute dengan variabel jalur serakah memiliki prioritas lebih tinggi daripada `$default` rute.

Mengontrol dan mengelola akses ke API HTTP di API Gateway

API Gateway mendukung beberapa mekanisme untuk mengendalikan dan mengelola akses ke HTTP API Anda:

- Otorisasi Lambdagunakan fungsi Lambda untuk mengontrol akses ke API. Untuk informasi selengkapnya, lihat Bekerja dengan AWS Lambda otorisasi untuk API HTTP.
- Otorisasi JWTgunakan token web JSON untuk mengontrol akses ke API. Untuk informasi selengkapnya, lihat Mengontrol akses ke API HTTP dengan otorisasi JWT.
- StandardAWSIAM role dan kebijakanmenawarkan kontrol akses yang fleksibel dan kuat. Anda dapat menggunakan peran dan kebijakan IAM untuk mengontrol siapa yang dapat membuat dan mengelola API Anda, serta siapa yang dapat memanggilnya. Untuk informasi selengkapnya, lihat Menggunakan otorisasi IAM.

Bekerja dengan AWS Lambda otorisasi untuk API HTTP

Anda menggunakan otorisasi Lambda untuk menggunakan fungsi Lambda untuk mengontrol akses ke HTTP API Anda. Kemudian, ketika klien memanggil API Anda, API Gateway memanggil fungsi Lambda Anda. API Gateway menggunakan respons dari fungsi Lambda Anda untuk menentukan apakah klien dapat mengakses API Anda.

Versi format payload

Versi format payload authorizer menentukan format data yang dikirimkan API Gateway ke pengotorisasi Lambda, dan bagaimana API Gateway menafsirkan respons dari Lambda. Jika Anda tidak menentukan versi format payload, akan AWS Management Console menggunakan versi terbaru secara default. Jika Anda membuat otorisasi Lambda menggunakan AWS CLI,, atau SDK AWS CloudFormation, Anda harus menentukan. `authorizerPayloadFormatVersion` Nilai yang di-support adalah `1.0` dan `2.0`.

Jika Anda membutuhkan kompatibilitas dengan REST API, gunakan versi`1.0`.

Contoh berikut menunjukkan struktur setiap versi format payload.

2.0

```
{
  "version": "2.0",
  "type": "REQUEST",
  "routeArn": "arn:aws:execute-api:us-east-1:123456789012:abcdef123/test/GET/
request",
  "identitySource": ["user1", "123"],
  "routeKey": "$default",
  "rawPath": "/my/path",
  "rawQueryString": "parameter1=value1&parameter1=value2&parameter2=value",
  "cookies": ["cookie1", "cookie2"],
  "headers": {
    "header1": "value1",
    "header2": "value2"
  },
  "queryStringParameters": {
    "parameter1": "value1,value2",
    "parameter2": "value"
  },
  "requestContext": {
    "accountId": "123456789012",
```

```
    "apiId": "api-id",
    "authentication": {
      "clientCert": {
        "clientCertPem": "CERT_CONTENT",
        "subjectDN": "www.example.com",
        "issuerDN": "Example issuer",
        "serialNumber": "1",
        "validity": {
          "notBefore": "May 28 12:30:02 2019 GMT",
          "notAfter": "Aug  5 09:36:04 2021 GMT"
        }
      }
    },
    "domainName": "id.execute-api.us-east-1.amazonaws.com",
    "domainPrefix": "id",
    "http": {
      "method": "POST",
      "path": "/my/path",
      "protocol": "HTTP/1.1",
      "sourceIp": "IP",
      "userAgent": "agent"
    },
    "requestId": "id",
    "routeKey": "$default",
    "stage": "$default",
    "time": "12/Mar/2020:19:03:58 +0000",
    "timeEpoch": 1583348638390
  },
  "pathParameters": { "parameter1": "value1" },
  "stageVariables": { "stageVariable1": "value1", "stageVariable2": "value2" }
}
```

1.0

```
{
  "version": "1.0",
  "type": "REQUEST",
  "methodArn": "arn:aws:execute-api:us-east-1:123456789012:abcdef123/test/GET/
request",
  "identitySource": "user1,123",
  "authorizationToken": "user1,123",
  "resource": "/request",
  "path": "/request",
```

```
  "httpMethod": "GET",
  "headers": {
    "X-AMZ-Date": "20170718T062915Z",
    "Accept": "*/*",
    "HeaderAuth1": "headerValue1",
    "CloudFront-Viewer-Country": "US",
    "CloudFront-Forwarded-Proto": "https",
    "CloudFront-Is-Tablet-Viewer": "false",
    "CloudFront-Is-Mobile-Viewer": "false",
    "User-Agent": "..."
  },
  "queryStringParameters": {
    "QueryString1": "queryValue1"
  },
  "pathParameters": {},
  "stageVariables": {
    "StageVar1": "stageValue1"
  },
  "requestContext": {
    "path": "/request",
    "accountId": "123456789012",
    "resourceId": "05c7jb",
    "stage": "test",
    "requestId": "...",
    "identity": {
      "apiKey": "...",
      "sourceIp": "...",
      "clientCert": {
        "clientCertPem": "CERT_CONTENT",
        "subjectDN": "www.example.com",
        "issuerDN": "Example issuer",
        "serialNumber": "a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1",
        "validity": {
          "notBefore": "May 28 12:30:02 2019 GMT",
          "notAfter": "Aug  5 09:36:04 2021 GMT"
        }
      }
    },
    "resourcePath": "/request",
    "httpMethod": "GET",
    "apiId": "abcdef123"
  }
}
```

Format respons otorisasi Lambda

Versi format payload juga menentukan struktur respons yang harus Anda kembalikan dari fungsi Lambda Anda.

Respons fungsi Lambda untuk format 1.0

Jika Anda memilih versi `1.0` format, otorisasi Lambda harus menampilkan kebijakan IAM yang mengizinkan atau menolak akses ke rute API Anda. Anda dapat menggunakan sintaks kebijakan IAM standar dalam kebijakan. Untuk contoh kebijakan IAM, lihatthe section called " Kontrol akses untuk menjalankan API". Anda dapat meneruskan properti konteks ke integrasi Lambda atau mengakses log dengan menggunakan. `$context.authorizer.property` `context`Objek adalah opsional dan `claims` merupakan placeholder yang dicadangkan dan tidak dapat digunakan sebagai objek konteks. Untuk mempelajari selengkapnya, lihat the section called "Variabel logging".

Example

```
{
  "principalId": "abcdef", // The principal user identification associated with the
 token sent by the client.
  "policyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Allow|Deny",
        "Resource": "arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/
{httpVerb}/[{resource}/[{child-resources}]]"
      }
    ]
  },
  "context": {
    "exampleKey": "exampleValue"
  }
}
```

Respons fungsi Lambda untuk format 2.0

Jika Anda memilih versi `2.0` format, Anda dapat mengembalikan nilai Boolean atau kebijakan IAM yang menggunakan sintaks kebijakan IAM standar dari fungsi Lambda Anda. Untuk mengembalikan nilai Boolean, aktifkan tanggapan sederhana untuk otorisasi. Contoh berikut menunjukkan format

yang harus Anda kodekan fungsi Lambda Anda untuk kembali. `context`Objeknya opsional. Anda dapat meneruskan properti konteks ke integrasi Lambda atau mengakses log dengan menggunakan. `$context.authorizer.`*`property`* Untuk mempelajari selengkapnya, lihat the section called "Variabel logging".

Simple response

```
{
  "isAuthorized": true/false,
  "context": {
    "exampleKey": "exampleValue"
  }
}
```

IAM policy

```
{
  "principalId": "abcdef", // The principal user identification associated with the
 token sent by the client.
  "policyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Allow|Deny",
        "Resource": "arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/
{httpVerb}/[{resource}/[{child-resources}]]"
      }
    ]
  },
  "context": {
    "exampleKey": "exampleValue"
  }
}
```

Contoh fungsi otorisasi Lambda

Contoh berikut fungsi Lambda Node.js menunjukkan format respons yang diperlukan yang perlu Anda kembalikan dari fungsi Lambda Anda untuk versi format payload. `2.0`

Simple response - Node.js

```
export const handler = async(event) => {
    let response = {
        "isAuthorized": false,
        "context": {
            "stringKey": "value",
            "numberKey": 1,
            "booleanKey": true,
            "arrayKey": ["value1", "value2"],
            "mapKey": {"value1": "value2"}
        }
    };

    if (event.headers.authorization === "secretToken") {
        console.log("allowed");
        response = {
            "isAuthorized": true,
            "context": {
                "stringKey": "value",
                "numberKey": 1,
                "booleanKey": true,
                "arrayKey": ["value1", "value2"],
                "mapKey": {"value1": "value2"}
            }
        };
    }

    return response;

};
```

Simple response - Python

```
import json


def lambda_handler(event, context):
    response = {
        "isAuthorized": False,
        "context": {
            "stringKey": "value",
            "numberKey": 1,
```

```
            "booleanKey": True,
            "arrayKey": ["value1", "value2"],
            "mapKey": {"value1": "value2"}
        }
    }

    try:
        if (event["headers"]["authorization"] == "secretToken"):
            response = {
                "isAuthorized": True,
                "context": {
                    "stringKey": "value",
                    "numberKey": 1,
                    "booleanKey": True,
                    "arrayKey": ["value1", "value2"],
                    "mapKey": {"value1": "value2"}
                }
            }
            print('allowed')
            return response
        else:
            print('denied')
            return response
    except BaseException:
        print('denied')
        return response
```

IAM policy - Node.js

```
export const handler = async(event) => {
  if (event.headers.authorization == "secretToken") {
    console.log("allowed");
    return {
      "principalId": "abcdef", // The principal user identification associated with
 the token sent by the client.
      "policyDocument": {
        "Version": "2012-10-17",
        "Statement": [{
          "Action": "execute-api:Invoke",
          "Effect": "Allow",
          "Resource": event.routeArn
        }]
      },
```

```
      "context": {
        "stringKey": "value",
        "numberKey": 1,
        "booleanKey": true,
        "arrayKey": ["value1", "value2"],
        "mapKey": { "value1": "value2" }
      }
    };
  }
  else {
    console.log("denied");
    return {
      "principalId": "abcdef", // The principal user identification associated with
 the token sent by the client.
      "policyDocument": {
        "Version": "2012-10-17",
        "Statement": [{
          "Action": "execute-api:Invoke",
          "Effect": "Deny",
          "Resource": event.routeArn
        }]
      },
      "context": {
        "stringKey": "value",
        "numberKey": 1,
        "booleanKey": true,
        "arrayKey": ["value1", "value2"],
        "mapKey": { "value1": "value2" }
      }
    };
  }
};
```

IAM policy - Python

```
import json


def lambda_handler(event, context):
    response = {
        # The principal user identification associated with the token sent by
        # the client.
        "principalId": "abcdef",
```

```
        "policyDocument": {
            "Version": "2012-10-17",
            "Statement": [{
                "Action": "execute-api:Invoke",
                "Effect": "Deny",
                "Resource": event["routeArn"]
            }]
        },
        "context": {
            "stringKey": "value",
            "numberKey": 1,
            "booleanKey": True,
            "arrayKey": ["value1", "value2"],
            "mapKey": {"value1": "value2"}
        }
    }

    try:
        if (event["headers"]["authorization"] == "secretToken"):
            response = {
                # The principal user identification associated with the token
                # sent by the client.
                "principalId": "abcdef",
                "policyDocument": {
                    "Version": "2012-10-17",
                    "Statement": [{
                        "Action": "execute-api:Invoke",
                        "Effect": "Allow",
                        "Resource": event["routeArn"]
                    }]
                },
                "context": {
                    "stringKey": "value",
                    "numberKey": 1,
                    "booleanKey": True,
                    "arrayKey": ["value1", "value2"],
                    "mapKey": {"value1": "value2"}
                }
            }
            print('allowed')
            return response
        else:
            print('denied')
            return response
```

```
    except BaseException:
        print('denied')
        return response
```

Sumber identitas

Anda dapat secara opsional menentukan sumber identitas untuk otorisasi Lambda. Sumber identitas menentukan lokasi data yang diperlukan untuk mengotorisasi permintaan. Misalnya, Anda dapat menentukan nilai string header atau kueri sebagai sumber identitas. Jika Anda menentukan sumber identitas, klien harus memasukkannya dalam permintaan. Jika permintaan klien tidak menyertakan sumber identitas, API Gateway tidak memanggil otorisasi Lambda Anda, dan klien menerima kesalahan. `401` Sumber identitas berikut didukung:

Ekspresi seleksi

Tipe	Contoh	Catatan
Nilai header	$ request.header. *`nama`*	Nama header tidak peka huruf besar/kecil.
Nilai string kueri	$request.querystring. *`nama`*	Nama string kueri peka huruf besar/kecil.
Variabel konteks	$ konteks. *`VariableName`*	Nilai variabel konteks yang didukung.
Variabel tahap	$ StageVariables. *`VariableName`*	Nilai variabel tahap.

Tanggapan otorisasi cache

Anda dapat mengaktifkan caching untuk otorisasi Lambda dengan menentukan file. authorizerResultTtlInSeconds Saat caching diaktifkan untuk otorisasi, API Gateway menggunakan sumber identitas otorisasi sebagai kunci cache. Jika klien menentukan parameter yang sama dalam sumber identitas dalam TTL yang dikonfigurasi, API Gateway menggunakan hasil otorisasi yang di-cache, daripada menjalankan fungsi Lambda Anda.

Untuk mengaktifkan caching, otorisasi Anda harus memiliki setidaknya satu sumber identitas.

Jika Anda mengaktifkan respons sederhana untuk otorisasi, respons otorisasi sepenuhnya mengizinkan atau menolak semua permintaan API yang cocok dengan nilai sumber identitas cache. Untuk izin yang lebih terperinci, nonaktifkan tanggapan sederhana dan kembalikan kebijakan IAM.

Secara default, API Gateway menggunakan respons otorisasi yang di-cache untuk semua rute API yang menggunakan otorisasi. Untuk menyimpan respons per rute, tambahkan `$context.routeKey` ke sumber identitas otorisasi Anda.

Buat Authorizer Lambda

Saat membuat otorisasi Lambda, Anda menentukan fungsi Lambda untuk API Gateway yang akan digunakan. Anda harus memberikan izin API Gateway untuk menjalankan fungsi Lambda dengan menggunakan kebijakan sumber daya fungsi atau peran IAM. Untuk contoh ini, kami memperbarui kebijakan sumber daya untuk fungsi tersebut sehingga memberikan izin API Gateway untuk menjalankan fungsi Lambda kami.

```
aws apigatewayv2 create-authorizer \
    --api-id abcdef123 \
    --authorizer-type REQUEST \
    --identity-source '$request.header.Authorization' \
    --name lambda-authorizer \
    --authorizer-uri 'arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-west-2:123456789012:function:my-function/invocations' \
    --authorizer-payload-format-version '2.0' \
    --enable-simple-responses
```

Perintah berikut memberikan izin API Gateway untuk menjalankan fungsi Lambda Anda. Jika API Gateway tidak memiliki izin untuk menjalankan fungsi Anda, klien akan menerima file. `500 Internal Server Error`

```
aws lambda add-permission \
    --function-name my-authorizer-function \
    --statement-id apigateway-invoke-permissions-abc123 \
    --action lambda:InvokeFunction \
    --principal apigateway.amazonaws.com \
    --source-arn "arn:aws:execute-api:us-west-2:123456789012:api-
id/authorizers/authorizer-id"
```

Setelah Anda membuat otorisasi dan memberikan izin API Gateway untuk memanggilnya, perbarui rute Anda untuk menggunakan otorisasi.

```
aws apigatewayv2 update-route \
    --api-id abcdef123 \
    --route-id acd123 \
    --authorization-type CUSTOM \
    --authorizer-id def123
```

Memecahkan masalah otorisasi Lambda

Jika API Gateway tidak dapat memanggil otorisasi Lambda Anda, atau otorisasi Lambda Anda mengembalikan respons dalam format yang tidak valid, klien akan menerima file. `500 Internal Server Error`

Untuk memecahkan masalah kesalahan, aktifkan pencatatan akses untuk tahap API Anda. Sertakan variabel `$context.authorizer.error` logging dalam format log Anda.

Jika log menunjukkan bahwa API Gateway tidak memiliki izin untuk menjalankan fungsi Anda, perbarui kebijakan sumber daya fungsi Anda atau berikan peran IAM untuk memberikan izin API Gateway untuk memanggil otorisasi Anda.

Jika log menunjukkan bahwa fungsi Lambda Anda mengembalikan respons yang tidak valid, verifikasi bahwa fungsi Lambda Anda mengembalikan respons dalam format yang diperlukan.

Mengontrol akses ke API HTTP dengan otorisasi JWT

Anda dapat menggunakan JSON Web Tokens (JWTs) sebagai bagian dari kerangka kerja OpenID Connect (OIDC) dan OAuth 2.0 untuk membatasi akses klien ke API Anda.

Jika Anda mengonfigurasi otorisasi JWT untuk rute API Anda, API Gateway memvalidasi JWT yang dikirimkan klien dengan permintaan API. API Gateway memungkinkan atau menolak permintaan berdasarkan validasi token, dan secara opsional, cakupan dalam token. Jika Anda mengonfigurasi cakupan untuk rute, token harus menyertakan setidaknya satu cakupan rute.

Anda dapat mengonfigurasi otorisasi yang berbeda untuk setiap rute API, atau menggunakan otorisasi yang sama untuk beberapa rute.

> **Note**
>
> Tidak ada mekanisme standar untuk membedakan token akses JWT dari jenis JWT lainnya, seperti token ID OpenID Connect. Kecuali Anda memerlukan token ID untuk otorisasi API, kami sarankan Anda mengonfigurasi rute Anda agar memerlukan cakupan otorisasi. Anda

juga dapat mengonfigurasi otorisasi JWT Anda untuk meminta penerbit atau audiens yang hanya digunakan penyedia identitas Anda saat mengeluarkan token akses JWT.

Mengotorisasi permintaan API dengan otorisasi JWT

API Gateway menggunakan alur kerja umum berikut untuk mengotorisasi permintaan ke rute yang dikonfigurasi untuk menggunakan otorisasi JWT.

1. `identitySource`Periksa token. `identitySource`Dapat hanya menyertakan token, atau token yang diawali dengan`Bearer`.
2. Mendekode token.
3. Periksa algoritma dan tanda tangan token dengan menggunakan kunci publik yang diambil dari penerbit. `jwks_uri` Saat ini, hanya algoritma berbasis RSA yang didukung. API Gateway dapat menyimpan kunci publik selama dua jam. Sebagai praktik terbaik, saat Anda memutar tombol, izinkan masa tenggang di mana kunci lama dan baru valid.
4. Validasi klaim. API Gateway mengevaluasi klaim token berikut:
   - `kid`— Token harus memiliki klaim header yang cocok dengan kunci `jwks_uri` yang menandatangani token.
   - `iss`— Harus cocok dengan `issuer`yang dikonfigurasi untuk otorisasi.
   - `aud`atau `client_id` — Harus cocok dengan salah satu `audience`entri yang dikonfigurasi untuk otorisasi. API Gateway `client_id` hanya memvalidasi jika tidak `aud` ada. Saat keduanya `aud` dan `client_id` ada, API Gateway mengevaluasi. `aud`
   - `exp`— Harus setelah waktu saat ini di UTC.
   - `nbf`— Harus sebelum waktu saat ini di UTC.
   - `iat`— Harus sebelum waktu saat ini di UTC.
   - `scope`atau `scp` — Token harus menyertakan setidaknya satu cakupan dalam rute. `authorizationScopes`

Jika salah satu langkah ini gagal, API Gateway menolak permintaan API.

Setelah memvalidasi JWT, API Gateway meneruskan klaim dalam token ke integrasi rute API. Sumber daya backend, seperti fungsi Lambda, dapat mengakses klaim JWT. Misalnya, jika JWT menyertakan klaim identitas`emailID`, itu tersedia untuk integrasi Lambda di. `$event.requestContext.authorizer.jwt.claims.emailID` Untuk informasi selengkapnya

tentang payload yang dikirimkan API Gateway ke integrasi Lambda, lihat. the section called “AWS Lambdaintegrasi”

Buat otorisasi JWT

Sebelum Anda membuat otorisasi JWT, Anda harus mendaftarkan aplikasi klien dengan penyedia identitas. Anda juga harus membuat API HTTP. Untuk contoh pembuatan API HTTP, lihatMembuat API HTTP.

Buat otorisasi JWT menggunakan konsol

Langkah-langkah berikut menunjukkan cara membuat otorisasi JWT menggunakan konsol.

Untuk membuat otorisasi JWT menggunakan konsol

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API HTTP.
3. Di panel navigasi utama, pilih Otorisasi.
4. Pilih tab Kelola otorisasi.
5. Pilih Buat.
6. Untuk jenis Authorizer, pilih JWT.
7. Konfigurasikan otorisasi JWT Anda, dan tentukan sumber Identitas yang menentukan sumber token.
8. Pilih Buat.

Buat otorisasi JWT menggunakan AWS CLI

AWS CLI Perintah berikut membuat otorisasi JWT. Untuk`jwt-configuration`, tentukan `Audience` dan `Issuer` untuk penyedia identitas Anda. Jika Anda menggunakan Amazon Cognito sebagai penyedia identitas, itu adalah. `IssuerUrl` `https://cognito-idp.`*`us-east-2`*`.amazonaws.com/`*`userPoolID`*

```
aws apigatewayv2 create-authorizer \
    --name authorizer-name \
    --api-id api-id \
    --authorizer-type JWT \
    --identity-source '$request.header.Authorization' \
    --jwt-configuration Audience=audience,Issuer=IssuerUrl
```

Buat otorisasi JWT menggunakan AWS CloudFormation

AWS CloudFormation Template berikut membuat API HTTP dengan otorisasi JWT yang menggunakan Amazon Cognito sebagai penyedia identitas.

Output dari AWS CloudFormation template adalah URL untuk UI yang dihosting Amazon Cognito tempat klien dapat mendaftar dan masuk untuk menerima JWT. Setelah klien masuk, klien dialihkan ke HTTP API Anda dengan token akses di URL. Untuk menjalankan API dengan token akses, ubah URL ke a ? untuk menggunakan token sebagai parameter string kueri. #

Contoh AWS CloudFormation template

```
AWSTemplateFormatVersion: '2010-09-09'
Description: |
  Example HTTP API with a JWT authorizer. This template includes an Amazon Cognito user
 pool as the issuer for the JWT authorizer
  and an Amazon Cognito app client as the audience for the authorizer. The outputs
 include a URL for an Amazon Cognito hosted UI where clients can
  sign up and sign in to receive a JWT. After a client signs in, the client is
 redirected to your HTTP API with an access token
  in the URL. To invoke the API with the access token, change the '#' in the URL to a
 '?' to use the token as a query string parameter.

Resources:
  MyAPI:
    Type: AWS::ApiGatewayV2::Api
    Properties:
      Description: Example HTTP API
      Name: api-with-auth
      ProtocolType: HTTP
      Target: !GetAtt MyLambdaFunction.Arn
  DefaultRouteOverrides:
    Type: AWS::ApiGatewayV2::ApiGatewayManagedOverrides
    Properties:
      ApiId: !Ref MyAPI
      Route:
        AuthorizationType: JWT
        AuthorizerId: !Ref JWTAuthorizer
  JWTAuthorizer:
    Type: AWS::ApiGatewayV2::Authorizer
    Properties:
      ApiId: !Ref MyAPI
      AuthorizerType: JWT
```

```
      IdentitySource:
        - '$request.querystring.access_token'
      JwtConfiguration:
        Audience:
        - !Ref AppClient
        Issuer: !Sub https://cognito-idp.${AWS::Region}.amazonaws.com/${UserPool}
      Name: test-jwt-authorizer
  MyLambdaFunction:
    Type: AWS::Lambda::Function
    Properties:
      Runtime: nodejs18.x
      Role: !GetAtt FunctionExecutionRole.Arn
      Handler: index.handler
      Code:
        ZipFile: |
          exports.handler = async (event) => {
              const response = {
                  statusCode: 200,
                  body: JSON.stringify('Hello from the ' + event.routeKey + ' route!'),
              };
              return response;
          };
  APIInvokeLambdaPermission:
    Type: AWS::Lambda::Permission
    Properties:
      FunctionName: !Ref MyLambdaFunction
      Action: lambda:InvokeFunction
      Principal: apigateway.amazonaws.com
      SourceArn: !Sub arn:${AWS::Partition}:execute-api:${AWS::Region}:
${AWS::AccountId}:${MyAPI}/$default/$default
  FunctionExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service:
              - lambda.amazonaws.com
            Action:
              - 'sts:AssumeRole'
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
```

```
  UserPool:
    Type: AWS::Cognito::UserPool
    Properties:
      UserPoolName: http-api-user-pool
      AutoVerifiedAttributes:
        - email
      Schema:
        - Name: name
          AttributeDataType: String
          Mutable: true
          Required: true
        - Name: email
          AttributeDataType: String
          Mutable: false
          Required: true
  AppClient:
    Type: AWS::Cognito::UserPoolClient
    Properties:
      AllowedOAuthFlows:
        - implicit
      AllowedOAuthScopes:
        - aws.cognito.signin.user.admin
        - email
        - openid
        - profile
      AllowedOAuthFlowsUserPoolClient: true
      ClientName: api-app-client
      CallbackURLs:
        - !Sub https://${MyAPI}.execute-api.${AWS::Region}.amazonaws.com
      ExplicitAuthFlows:
        - ALLOW_USER_PASSWORD_AUTH
        - ALLOW_REFRESH_TOKEN_AUTH
      UserPoolId: !Ref UserPool
      SupportedIdentityProviders:
        - COGNITO
  HostedUI:
    Type: AWS::Cognito::UserPoolDomain
    Properties:
      Domain: !Join
        - '-'
        - - !Ref MyAPI
          - !Ref AppClient
      UserPoolId: !Ref UserPool
Outputs:
```

```
  SignupURL:
    Value: !Sub https://${HostedUI}.auth.${AWS::Region}.amazoncognito.com/login?
client_id=${AppClient}&response_type=token&scope=email+profile&redirect_uri=https://
${MyAPI}.execute-api.${AWS::Region}.amazonaws.com
```

Perbarui rute untuk menggunakan otorisasi JWT

Anda dapat menggunakan konsol, SDK AWS CLI, atau AWS SDK untuk memperbarui rute untuk menggunakan otorisasi JWT.

Perbarui rute untuk menggunakan otorisasi JWT dengan menggunakan konsol

Langkah-langkah berikut menunjukkan cara memperbarui rute untuk menggunakan otorisasi JWT menggunakan konsol.

Untuk membuat otorisasi JWT menggunakan konsol

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API HTTP.
3. Di panel navigasi utama, pilih Otorisasi.
4. Pilih metode, lalu pilih otorisasi Anda dari menu tarik-turun, dan pilih Lampirkan otorisasi.

Perbarui rute untuk menggunakan otorisasi JWT dengan menggunakan AWS CLI

Perintah berikut memperbarui rute untuk menggunakan otorisasi JWT menggunakan file. AWS CLI

```
aws apigatewayv2 update-route \
   --api-id api-id  \
   --route-id route-id  \
   --authorization-type JWT \
   --authorizer-id authorizer-id \
   --authorization-scopes user.email
```

Menggunakan otorisasi IAM

Anda dapat mengaktifkan otorisasi IAM untuk rute API HTTP. Ketika otorisasi IAM diaktifkan, klien harus menggunakanTanda Tangan Versi 4untuk menandatangani permintaan mereka denganAWSkredensi. API Gateway memanggil rute API Anda hanya jika klien memilikiexecute-apiizin untuk rute.

IAM otorisasi untuk HTTP API mirip dengan yang untukAPI REST.

> **Note**
>
> Kebijakan sumber daya saat ini tidak didukung untuk API HTTP.

Untuk contoh kebijakan IAM yang memberikan izin kepada klien untuk memanggil API, lihatthe section called “ Kontrol akses untuk menjalankan API”.

Aktifkan otorisasi IAM untuk rute

BerikutAWS CLIperintah memungkinkan otorisasi IAM untuk rute API HTTP.

```
aws apigatewayv2 update-route \
    --api-id abc123 \
    --route-id abcdef \
    --authorization-type AWS_IAM
```

Mengkonfigurasi integrasi untuk HTTP API

Integrasimenghubungkan rute ke sumber daya backend. API HTTP mendukung proxy Lambda,AWSlayanan, dan integrasi proxy HTTP. Misalnya, Anda dapat mengkonfigurasiP0STrequest/signuprute API Anda untuk diintegrasikan dengan fungsi Lambda yang menangani pelanggan yang mendaftar.

Topik

- Bekerja dengan integrasiAWS Lambda proxy untuk API HTTP
- Bekerja dengan integrasi proxy HTTP untuk API HTTP
- Bekerja dengan integrasi AWS layanan untuk API HTTP
- Bekerja dengan integrasi pribadi untuk API HTTP

Bekerja dengan integrasiAWS Lambda proxy untuk API HTTP

Integrasi proxy Lambda memungkinkan Anda mengintegrasikan rute API dengan fungsi Lambda. Ketika klien memanggil API Anda, API Gateway mengirimkan permintaan ke fungsi Lambda dan mengembalikan respons fungsi ke klien. Untuk contoh membuat API HTTP, lihatMembuat API HTTP.

Versi format muatan

Versi format payload menentukan format data yang dikirim API Gateway ke integrasi Lambda, dan bagaimana API Gateway menafsirkan respons dari Lambda. Jika Anda tidak menentukan versinya format, penggunaAWS Management Console menggunakan versi terbaru secara default. Jika Anda membuat integrasi Lambda dengan menggunakanAWS CLI,AWS CloudFormation, atau SDK, Anda harus menentukan`payloadFormatVersion`. Nilai yang di-support adalah `1.0` dan `2.0`.

Contoh berikut menunjukkan struktur dari setiap versi format payload.

> **Note**
>
> Nama header lowercased.
> Format`2.0` tidak memiliki`multiValueHeaders` atau`multiValueQueryStringParameters` bidang. Header duplikat digabungkan dengan koma dan disertakan dalam`headers` bidang. String kueri duplikat digabungkan dengan koma dan disertakan dalam`queryStringParameters` bidang.
> Format`2.0` termasuk`cookies` bidang baru. Semua header cookie dalam permintaan digabungkan dengan koma dan ditambahkan ke`cookies` bidang. Dalam menanggapi klien, setiap cookie menjadi`set-cookie` header.

2.0

```
{
  "version": "2.0",
  "routeKey": "$default",
  "rawPath": "/my/path",
  "rawQueryString": "parameter1=value1&parameter1=value2&parameter2=value",
  "cookies": [
    "cookie1",
    "cookie2"
  ],
  "headers": {
    "header1": "value1",
    "header2": "value1,value2"
  },
  "queryStringParameters": {
    "parameter1": "value1,value2",
    "parameter2": "value"
  },
```

```
  "requestContext": {
    "accountId": "123456789012",
    "apiId": "api-id",
    "authentication": {
      "clientCert": {
        "clientCertPem": "CERT_CONTENT",
        "subjectDN": "www.example.com",
        "issuerDN": "Example issuer",
        "serialNumber": "a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1",
        "validity": {
          "notBefore": "May 28 12:30:02 2019 GMT",
          "notAfter": "Aug  5 09:36:04 2021 GMT"
        }
      }
    },
    "authorizer": {
      "jwt": {
        "claims": {
          "claim1": "value1",
          "claim2": "value2"
        },
        "scopes": [
          "scope1",
          "scope2"
        ]
      }
    },
    "domainName": "id.execute-api.us-east-1.amazonaws.com",
    "domainPrefix": "id",
    "http": {
      "method": "POST",
      "path": "/my/path",
      "protocol": "HTTP/1.1",
      "sourceIp": "192.0.2.1",
      "userAgent": "agent"
    },
    "requestId": "id",
    "routeKey": "$default",
    "stage": "$default",
    "time": "12/Mar/2020:19:03:58 +0000",
    "timeEpoch": 1583348638390
  },
  "body": "Hello from Lambda",
  "pathParameters": {
```

```
    "parameter1": "value1"
  },
  "isBase64Encoded": false,
  "stageVariables": {
    "stageVariable1": "value1",
    "stageVariable2": "value2"
  }
}
```

1.0

```
{
  "version": "1.0",
  "resource": "/my/path",
  "path": "/my/path",
  "httpMethod": "GET",
  "headers": {
    "header1": "value1",
    "header2": "value2"
  },
  "multiValueHeaders": {
    "header1": [
      "value1"
    ],
    "header2": [
      "value1",
      "value2"
    ]
  },
  "queryStringParameters": {
    "parameter1": "value1",
    "parameter2": "value"
  },
  "multiValueQueryStringParameters": {
    "parameter1": [
      "value1",
      "value2"
    ],
    "parameter2": [
      "value"
    ]
  },
  "requestContext": {
```

```
    "accountId": "123456789012",
    "apiId": "id",
    "authorizer": {
      "claims": null,
      "scopes": null
    },
    "domainName": "id.execute-api.us-east-1.amazonaws.com",
    "domainPrefix": "id",
    "extendedRequestId": "request-id",
    "httpMethod": "GET",
    "identity": {
      "accessKey": null,
      "accountId": null,
      "caller": null,
      "cognitoAuthenticationProvider": null,
      "cognitoAuthenticationType": null,
      "cognitoIdentityId": null,
      "cognitoIdentityPoolId": null,
      "principalOrgId": null,
      "sourceIp": "192.0.2.1",
      "user": null,
      "userAgent": "user-agent",
      "userArn": null,
      "clientCert": {
        "clientCertPem": "CERT_CONTENT",
        "subjectDN": "www.example.com",
        "issuerDN": "Example issuer",
        "serialNumber": "a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1",
        "validity": {
          "notBefore": "May 28 12:30:02 2019 GMT",
          "notAfter": "Aug  5 09:36:04 2021 GMT"
        }
      }
    },
    "path": "/my/path",
    "protocol": "HTTP/1.1",
    "requestId": "id=",
    "requestTime": "04/Mar/2020:19:15:17 +0000",
    "requestTimeEpoch": 1583349317135,
    "resourceId": null,
    "resourcePath": "/my/path",
    "stage": "$default"
  },
  "pathParameters": null,
```

```
  "stageVariables": null,
  "body": "Hello from Lambda!",
  "isBase64Encoded": false
}
```

Format respons fungsi Lambda

Versi format payload menentukan struktur respons yang harus dikembalikan fungsi Lambda Anda.

Respon fungsi Lambda untuk format 1.0

Dengan versi`1.0` format, integrasi Lambda harus mengembalikan respon dalam format berikut.

Example

```
{
    "isBase64Encoded": true|false,
    "statusCode": httpStatusCode,
    "headers": { "headername": "headervalue", ... },
    "multiValueHeaders": { "headername": ["headervalue", "headervalue2", ...], ... },
    "body": "..."
}
```

Respons fungsi Lambda untuk format 2.0

Dengan versi`2.0` format, API Gateway dapat menyimpulkan format respons untuk Anda. API Gateway membuat asumsi berikut jika fungsi Lambda Anda mengembalikan JSON yang valid dan tidak mengembalikan`statusCode`:

- `isBase64Encoded` adalah `false`.
- `statusCode` adalah `200`.
- `content-type` adalah `application/json`.
- `body`adalah respon fungsi.

Contoh berikut menunjukkan output dari fungsi Lambda dan interpretasi API Gateway.

Output fungsi Lambda	Interpretasi API Gateway
`"Hello from Lambda!"`	`{`

Output fungsi Lambda	Interpretasi API Gateway

	<pre>"isBase64Encoded": false,
	"statusCode": 200,
	"body": "Hello from Lambda!",
	"headers": {
	"content-type": "application/
	json"
	}
	}</pre>

<pre>{ "message": "Hello from Lambda!" }</pre>	<pre>{
	"isBase64Encoded": false,
	"statusCode": 200,
	"body": "{ \"message\": \"Hello from
	Lambda!\" }",
	"headers": {
	"content-type": "application/
	json"
	}
	}</pre>

Untuk menyesuaikan respons, fungsi Lambda Anda harus mengembalikan respons dengan format berikut.

```
{
    "cookies" : ["cookie1", "cookie2"],
    "isBase64Encoded": true|false,
    "statusCode": httpStatusCode,
    "headers": { "headername": "headervalue", ... },
    "body": "Hello from Lambda!"
}
```

Bekerja dengan integrasi proxy HTTP untuk API HTTP

Integrasi proxy HTTP memungkinkan Anda menghubungkan rute API ke titik akhir HTTP yang dapat dirutekan secara publik. Dengan tipe integrasi ini, API Gateway meneruskan seluruh permintaan dan respons antara frontend dan backend.

Untuk membuat integrasi proxy HTTP, berikan URL titik akhir HTTP yang dapat dirutekan secara publik.

Integrasi proxy HTTP dengan variabel jalur

Anda dapat menggunakan variabel jalur di rute API HTTP.

Misalnya, rute `/pets/{petID}` menangkap permintaan ke`/pets/6`. Anda dapat mereferensikan variabel jalur dalam URI integrasi untuk mengirim konten variabel ke integrasi. Contohnya adalah `/pets/extendedpath/{petID}`.

Anda dapat menggunakan variabel jalur serakah untuk menangkap semua sumber daya anak dari suatu rute. Untuk membuat variabel jalur serakah, tambahkan + ke nama variabel—misalnya,. `{proxy+}`

Untuk menyiapkan rute dengan integrasi proxy HTTP yang menangkap semua permintaan, buat rute API dengan variabel jalur serakah (misalnya,`/parent/{proxy+}`). Integrasikan rute dengan titik akhir HTTP (misalnya,`https://petstore-demo-endpoint.execute-api.com/petstore/{proxy}`) pada `ANY` metode. Variabel jalur serakah harus berada di ujung jalur sumber daya.

Bekerja dengan integrasi AWS layanan untuk API HTTP

Anda dapat mengintegrasikan HTTP API Anda dengan AWS layanan dengan menggunakan integrasi kelas satu. Integrasi kelas satu menghubungkan rute API HTTP ke API AWS layanan. Saat klien memanggil rute yang didukung oleh integrasi kelas satu, API Gateway akan memanggil API AWS layanan untuk Anda. Misalnya, Anda dapat menggunakan integrasi kelas satu untuk mengirim pesan ke antrean Amazon Simple Queue Service, atau untuk memulai mesin status. AWS Step Functions Untuk tindakan layanan yang didukung, lihatthe section called “AWSReferensi integrasi layanan”.

Parameter permintaan pemetaan

Integrasi kelas satu memiliki parameter yang diperlukan dan opsional. Anda harus mengkonfigurasi semua parameter yang diperlukan untuk membuat integrasi. Anda dapat menggunakan nilai statis atau parameter peta yang dievaluasi secara dinamis saat runtime. Untuk daftar lengkap integrasi dan parameter yang didukung, lihatthe section called “AWSReferensi integrasi layanan”.

Pemetaan parameter

Tipe	Contoh	Catatan
Nilai header	$ request.header. *nama*	Nama header tidak peka huruf besar/kecil. API Gateway

Tipe	Contoh	Catatan
		menggabungkan beberapa nilai header dengan koma, misalnya`"header1": "value1,value2"` .
Nilai string kueri	$request.querystring. *`nama`*	Nama string kueri peka huruf besar/kecil. API Gateway menggabungkan beberapa nilai dengan koma, misalnya`"querystring1": "Value1,Value2"` .
Parameter jalur	$request.path. *`nama`*	Nilai parameter jalur dalam permintaan. Misalnya jika rutenya`/pets/{petId}` , Anda dapat memetakan `petId` parameter dari permintaan dengan *`$request.path.petid`* .
Minta passthrough tubuh	$ request.body	API Gateway melewati seluruh isi permintaan.

Tipe	Contoh	Catatan
Isi permintaan	$ request.body. *nama*	Ekspresi jalur JSON. Descent rekursif (`$request.body..` *name*) dan ekspresi filter (`?(`*expression*` )`) tidak didukung. **Note** Saat Anda menentukan jalur JSON, API Gateway memotong isi permintaan pada 100 KB dan kemudian menerapkan ekspresi seleksi. Untuk mengirim muatan yang lebih besar dari 100 KB, tentukan`$request.body` .
Variabel konteks	$ konteks. *VariableName*	Nilai variabel konteks yang didukung.
Variabel tahap	$ StageVariables. *VariableName*	Nilai variabel tahap.
Nilai statis	*tali*	Nilai konstan.

Buat integrasi kelas satu

Sebelum membuat integrasi kelas satu, Anda harus membuat peran IAM yang memberikan izin API Gateway untuk menjalankan tindakan AWS layanan yang Anda integrasikan. Untuk mempelajari lebih lanjut, lihat Membuat peran untuk AWS layanan.

Untuk membuat integrasi kelas satu, pilih tindakan AWS layanan yang didukung, seperti`SQS-SendMessage`, mengonfigurasi parameter permintaan, dan berikan peran yang memberikan izin API Gateway untuk menjalankan API layanan terintegrasi. AWS Tergantung pada subtipe integrasi, parameter permintaan yang berbeda diperlukan. Untuk mempelajari selengkapnya, lihat the section called “AWSReferensi integrasi layanan”.

AWS CLIPerintah berikut membuat integrasi yang mengirimkan pesan Amazon SQS.

```
aws apigatewayv2 create-integration \
    --api-id abcdef123 \
    --integration-subtype SQS-SendMessage \
    --integration-type AWS_PROXY \
    --payload-format-version 1.0 \
    --credentials-arn arn:aws:iam::123456789012:role/apigateway-sqs \
    --request-parameters '{"QueueUrl": "$request.header.queueUrl", "MessageBody":
 "$request.body.message"}'
```

Buat integrasi kelas satu menggunakan AWS CloudFormation

Contoh berikut menunjukkan AWS CloudFormation cuplikan yang membuat `/{source}/{detailType}` rute dengan integrasi kelas satu dengan Amazon. EventBridge

`Source`Parameter dipetakan ke parameter `{source}` jalur, `DetailType` dipetakan ke parameter `{DetailType}` jalur, dan `Detail` parameter dipetakan ke badan permintaan.

Cuplikan tidak menampilkan bus acara atau peran IAM yang memberikan izin API Gateway untuk menjalankan tindakan. `PutEvents`

```
Route:
    Type: AWS::ApiGatewayV2::Route
    Properties:
      ApiId: !Ref HttpApi
      AuthorizationType: None
      RouteKey: 'POST /{source}/{detailType}'
      Target: !Join
        - /
        - - integrations
          - !Ref Integration
  Integration:
    Type: AWS::ApiGatewayV2::Integration
    Properties:
      ApiId: !Ref HttpApi
```

```
      IntegrationType: AWS_PROXY
      IntegrationSubtype: EventBridge-PutEvents
      CredentialsArn: !GetAtt EventBridgeRole.Arn
      RequestParameters:
        Source: $request.path.source
        DetailType: $request.path.detailType
        Detail: $request.body
        EventBusName: !GetAtt EventBus.Arn
      PayloadFormatVersion: "1.0"
```

Referensi subtipe integrasi

Berikutsubtipe integrasididukung untuk API HTTP.

Subtipe integrasi

- Peristiwa Bridge-putEvents
- Pesan SQS
- Penerima SQS
- Penghapus SQS
- Antrean SQS
- AppConfig-getConfiguration
- Rekam bahasa Indonesia
- Langkah-StartExecution
- Langkah-StartSyncExecution
- Langkah-stopExecution

Peristiwa Bridge-putEvents

Mengirim peristiwa khusus ke Amazon EventBridge sehingga mereka dapat dicocokkan dengan aturan.

EventBridge-putEvents 1.0

Parameter	Diperlukan
Detail	Benar
DetailType	Benar

Parameter	Diperlukan
Source	Benar
Waktu	Salah
EventBusName	Salah
Sumber daya	Salah
Wilayah	Salah
TraceHeader	Salah

Untuk mempelajari lebih lanjut, lihatPutEventsdiReferensi API Amazon EventBridge.

Pesan SQS

Mengirimkan pesan ke antrian yang ditentukan.

SQS-SendMessage 1.0

Parameter	Diperlukan
QueueUrl	Benar
MessageBody	Benar
DelaySeconds	Salah
MessageAttributes	Salah
MessageDeduplicationId	Salah
MessageGroupID	Salah
MessageSystemAttributes	Salah
Wilayah	Salah

Untuk mempelajari lebih lanjut, lihatSendMessagediReferensi Amazon Simple Queue Service.

Penerima SQS

Mengambil satu atau lebih pesan (hingga 10), dari antrian yang ditentukan.

Penerima SQS 1.0

Parameter	Diperlukan
QueueUrl	Benar
AttributeName	Salah
MaxNumberOfMessages	Salah
MessageAttributeNames	Salah
ReceiveRequestAttemptId	Salah
VisibilityTimeout	Salah
WaitTimeSeconds	Salah
Wilayah	Salah

Untuk mempelajari lebih lanjut, lihatReceiveMessagediReferensi Amazon Simple Queue Service.

Penghapus SQS

Menghapus pesan yang ditentukan dari antrian yang ditentukan.

Penghapus SQS 1.0

Parameter	Diperlukan
ReceiptHandle	Benar
QueueUrl	Benar
Wilayah	Salah

Untuk mempelajari lebih lanjut, lihatDeleteMessagediReferensi Amazon Simple Queue Service.

Antrean SQS

Menghapus semua pesan dalam antrian yang ditentukan.

SQS-Purgeue 1.0

Parameter	Diperlukan
QueueUrl	Benar
Wilayah	Salah

Untuk mempelajari lebih lanjut, lihatPurgeQueuediReferensi Amazon Simple Queue Service.

AppConfig-getConfiguration

Menerima informasi tentang konfigurasi.

AppConfig-getConfiguration 1.0

Parameter	Diperlukan
Aplikasi	Benar
Environment	Benar
Konfigurasi	Benar
ClientId	Benar
ClientConfigurationVersion	Salah
Wilayah	Salah

Untuk mempelajari lebih lanjut, lihatGetConfigurationdiAWSReferensi AppConfig API.

Rekam bahasa Indonesia

Menulis catatan data tunggal ke Amazon Kinesis data stream.

Minesis-putRecord 1.0

Parameter	Diperlukan
StreamName	Benar
Data	Benar
PartitionKey	Benar
SequenceNumberForOrdering	Salah
ExplicitHashKey	Salah
Wilayah	Salah

Untuk mempelajari lebih lanjut, lihatPutRecorddiReferensi Amazon Kinesis Data Streams.

Langkah-StartExecution

Memulai eksekusi mesin status.

Langkah-StartExecution 1.0

Parameter	Diperlukan
StateMachineArn	Benar
Nama	Salah
Input	Salah
Wilayah	Salah

Untuk mempelajari lebih lanjut, lihatStartExecutiondiAWS Step FunctionsReferensi API.

Langkah-StartSyncExecution

Memulai eksekusi mesin negara sinkron.

Langkah-StartSyncExecution 1.0

Parameter	Diperlukan
StateMachineArn	Benar
Nama	Salah
Input	Salah
Wilayah	Salah
TraceHeader	Salah

Untuk mempelajari lebih lanjut, lihatStartSyncEksekusidiAWS Step FunctionsReferensi API.

Langkah-stopExecution

Menghentikan eksekusi.

Langkah-stopExecution 1.0

Parameter	Diperlukan
ExecutionARN	Benar
Penyebab	Salah
Kesalahan	Salah
Wilayah	Salah

Untuk mempelajari lebih lanjut, lihatStopExecutiondiAWS Step FunctionsReferensi API.

Bekerja dengan integrasi pribadi untuk API HTTP

Integrasi pribadi memungkinkan Anda membuat integrasi API dengan sumber daya pribadi di VPC, seperti Application Load Balancers atau aplikasi berbasis kontainer Amazon ECS.

Anda dapat mengekspos sumber daya Anda dalam VPC untuk akses oleh klien di luar VPC dengan menggunakan integrasi pribadi. Anda dapat mengontrol akses ke API Anda dengan menggunakan salah satumetode otorisasiyang didukung API Gateway.

Untuk membuat integrasi pribadi, Anda harus terlebih dahulu membuat tautan VPC. Untuk mempelajari lebih lanjut tentang tautan VPC, lihatBekerja dengan link VPC untuk HTTP API.

Setelah membuat tautan VPC, Anda dapat mengatur integrasi pribadi yang terhubung ke Application Load Balancer, Network Load Balancer, atau sumber daya yang terdaftar denganAWS Cloud Maplayanan.

Untuk membuat integrasi pribadi, semua sumber daya harus dimiliki oleh yang samaAWSakun (termasuk penyeimbang beban atauAWS Cloud Maplayanan, tautan VPC dan API HTTP).

Secara default, lalu lintas integrasi pribadi menggunakan protokol HTTP. Anda dapat menentukan`tlsConfig`jika Anda memerlukan lalu lintas integrasi pribadi untuk menggunakan HTTPS.

> **Note**
>
> Untuk integrasi pribadi, API Gateway menyertakantahapporsi endpoint API dalam permintaan ke sumber daya backend Anda. Misalnya, permintaan `ketest`tahap API termasuk`test/`*`route-path`*dalam permintaan untuk integrasi pribadi Anda. Untuk menghapus nama tahap dari permintaan ke sumber daya backend Anda, gunakanpemetaan parameteruntuk menimpa jalur permintaan ke`$request.path`.

Buat integrasi pribadi menggunakan Application Load Balancer atau Network Load Balancer

Sebelum membuat integrasi pribadi, Anda harus membuat tautan VPC. Untuk mempelajari lebih lanjut tentang tautan VPC, lihatBekerja dengan link VPC untuk HTTP API.

Untuk membuat integrasi pribadi dengan Application Load Balancer atau Network Load Balancer, buat integrasi proxy HTTP, tentukan tautan VPC yang akan digunakan, dan berikan ARN listener dari load balancer.

Gunakan perintah berikut untuk membuat integrasi pribadi yang terhubung ke penyeimbang muatan dengan menggunakan tautan VPC.

```
aws apigatewayv2 create-integration --api-id api-id --integration-type HTTP_PROXY \
```

```
    --integration-method GET --connection-type VPC_LINK \
    --connection-id VPC-link-ID \
    --integration-uri arn:aws:elasticloadbalancing:us-east-2:123456789012:listener/app/
my-load-balancer/50dc6c495c0c9188/0467ef3c8400ae65
    --payload-format-version 1.0
```

Buat integrasi pribadi menggunakanAWS Cloud Mappenemuan layanan

Sebelum membuat integrasi pribadi, Anda harus membuat tautan VPC. Untuk mempelajari lebih lanjut tentang tautan VPC, lihatBekerja dengan link VPC untuk HTTP API.

Untuk integrasi denganAWS Cloud Map, API Gateway menggunakan`DiscoverInstances`untuk mengidentifikasi sumber daya. Anda dapat menggunakan parameter kueri untuk menargetkan sumber daya tertentu. Atribut sumber daya terdaftar harus menyertakan alamat IP dan port. API Gateway mendistribusikan permintaan ke seluruh sumber daya sehat yang dikembalikan`DiscoverInstances`. Untuk mempelajari lebih lanjut, lihatDiscoverInstancesdi dalamAWS Cloud MapReferensi API.

> **Note**
>
> Jika Anda menggunakan Amazon ECS untuk mengisi entriAWS Cloud Map, Anda harus mengonfigurasi tugas Amazon ECS Anda untuk menggunakan data SRV dengan Amazon ECS Service Discovery atau mengaktifkan Amazon ECS Service Connect. Untuk informasi lebih lanjut, lihatLayanan interkoneksidalam Panduan Pengembang Layanan Kontainer Amazon Elastic.

Untuk membuat integrasi pribadi denganAWS Cloud Map, buat integrasi proxy HTTP, tentukan tautan VPC yang akan digunakan, dan berikan ARN dariAWS Cloud Maplayanan.

Gunakan perintah berikut untuk membuat integrasi pribadi yang menggunakanAWS Cloud Mappenemuan layanan untuk mengidentifikasi sumber daya.

```
aws apigatewayv2 create-integration --api-id api-id --integration-type HTTP_PROXY  \
    --integration-method GET --connection-type VPC_LINK \
    --connection-id VPC-link-ID \
    --integration-uri arn:aws:servicediscovery:us-east-2:123456789012:service/srv-id?
stage=prod&deployment=green_deployment
    --payload-format-version 1.0
```

Bekerja dengan link VPC untuk HTTP API

Tautan VPC memungkinkan Anda membuat integrasi pribadi yang menghubungkan rute API HTTP Anda ke sumber daya pribadi di VPC, seperti Application Load Balancers atau aplikasi berbasis kontainer Amazon ECS. Untuk mempelajari selengkapnya tentang cara membuat integrasi privat, lihatBekerja dengan integrasi pribadi untuk API HTTP.

Integrasi pribadi menggunakan tautan VPC untuk merangkum koneksi antara API Gateway dan sumber daya VPC yang ditargetkan. Anda dapat menggunakan kembali tautan VPC di berbagai rute dan API.

Saat Anda membuat tautan VPC, API Gateway membuat dan mengelolaantarmuka jaringan elastisuntuk tautan VPC di akun Anda. Proses ini dapat menghabiskan waktu beberapa menit. Ketika link VPC siap digunakan, transisi statusnya dari`PENDING`kepada`AVAILABLE`.

> **Note**
>
> Jika tidak ada lalu lintas yang dikirim melalui tautan VPC selama 60 hari, itu menjadi`INACTIVE`. Ketika link VPC dalam sebuah`INACTIVE`negara, API Gateway menghapus semua antarmuka jaringan tautan VPC. Hal ini menyebabkan permintaan API yang bergantung pada link VPC gagal. Jika permintaan API dilanjutkan, API Gateway mengatur kembali antarmuka jaringan. Perlu waktu beberapa menit untuk membuat antarmuka jaringan dan mengaktifkan kembali tautan VPC. Anda dapat menggunakan status tautan VPC untuk memantau status tautan VPC Anda.

Buat tautan VPC dengan menggunakanAWS CLI

Gunakan perintah berikut untuk membuat tautan VPC. Untuk membuat tautan VPC, semua sumber daya yang terlibat harus dimiliki oleh yang samaAWSakun.

```
aws apigatewayv2 create-vpc-link --name MyVpcLink \
    --subnet-ids subnet-aaaa subnet-bbbb \
    --security-group-ids sg1234 sg5678
```

> **Note**
>
> Tautan VPC bersifat tetap. Setelah membuat tautan VPC, Anda tidak dapat mengubah subnet atau grup keamanan.

Menghapus tautan VPC dengan menggunakanAWS CLI

Gunakan perintah berikut untuk menghapus tautan VPC.

```
aws apigatewayv2 delete-vpc-link --vpc-link-id abcd123
```

Ketersediaan berdasarkan Wilayah

Tautan VPC untuk API HTTP didukung di Wilayah dan Availability Zone berikut:

Nama wilayah	Wilayah	Availability Zone
Timur AS (Ohio)	us–east-2	use2-az1, use2-az2, use2-az3
US East (Northern Virginia)	as-east-1	use1-az1, use1-az2, use1-az4, use1-az5, use1-az6
US West (Northern California)	us-west-1	usw1-az1, usw1-az3
US West (Oregon)	us-west-2	usw2-az1, usw2-az2, usw2-az3, usw2-az4
Asia Pasifik (Hong Kong)	ap-east-1	ape1-az2, ape1-az3
Asia Pasifik (Mumbai)	ap-south-1	aps1-az1, aps1-az2, aps1-az3
Asia Pacific (Seoul)	ap-northeast-2	apne2-az1, apne2-az2, apne2-az3
Asia Pacific (Singapore)	ap-southeast-1	apse1-az1, apse1-az2, apse1-az3

Nama wilayah	Wilayah	Availability Zone
Asia Pacific (Sydney)	ap-southeast-2	apse2-az1, apse2-az2, apse2-az3
Asia Pacific (Tokyo)	ap-northeast-1	apne1-az1, apne1-az2, apne1-az4
Canada (Central)	ca-central-1	cac1-az1, cac1-az2
Eropa (Frankfurt)	eu-central-1	euc1-az1, euc1-az2, euc1-az3
Eropa (Irlandia)	eu-west-1	euw1-az1, euw1-az2, euw1-az3
Eropa (London)	eu-west-2	euw2-az1, euw2-az2, euw2-az3
Europe (Paris)	eu-west-3	euw3-az1, euw3-az3
Eropa (Stockholm)	eu-north-1	eun1-az1, eun1-az2, eun1-az3
Timur Tengah (Bahrain)	me-south-1	mes1-az1, mes1-az2, mes1-az3
South America (São Paulo)	sa-east-1	sae1-az1, sae1-az2, sae1-az3
AWS GovCloud (US-West)	us-gov-west-1	usgw1-az1, usgw1-az2, usgw1-az3

Mengonfigurasi CORS untuk HTTP API

Cross-origin resource sharing (CORS) adalah fitur keamanan browser yang membatasi permintaan HTTP yang dimulai dari skrip yang berjalan di browser. Jika Anda tidak dapat mengakses API dan menerima pesan kesalahan yang berisi`Cross-Origin Request Blocked`, Anda mungkin perlu mengaktifkan CORS.

CORS biasanya diperlukan untuk membangun aplikasi web yang mengakses API yang dihosting pada domain atau asal yang berbeda. Anda dapat mengaktifkan CORS untuk mengizinkan permintaan ke API Anda dari aplikasi web yang dihosting di domain lain. Misalnya, jika API Anda di-host `https://{api_id}.execute-api.{region}.amazonaws.com/` dan Anda ingin memanggil API Anda dari aplikasi web yang di-host`example.com`, API Anda harus mendukung CORS.

Jika Anda mengonfigurasi CORS untuk API, API Gateway secara otomatis mengirimkan respons ke permintaan OPTIONS preflight, meskipun tidak ada rute OPTIONS yang dikonfigurasi untuk API Anda. Untuk permintaan CORS, API Gateway menambahkan header CORS yang dikonfigurasi ke respons dari integrasi.

> **Note**
>
> Jika Anda mengonfigurasi CORS untuk API, API Gateway mengabaikan header CORS yang dikembalikan dari integrasi backend Anda.

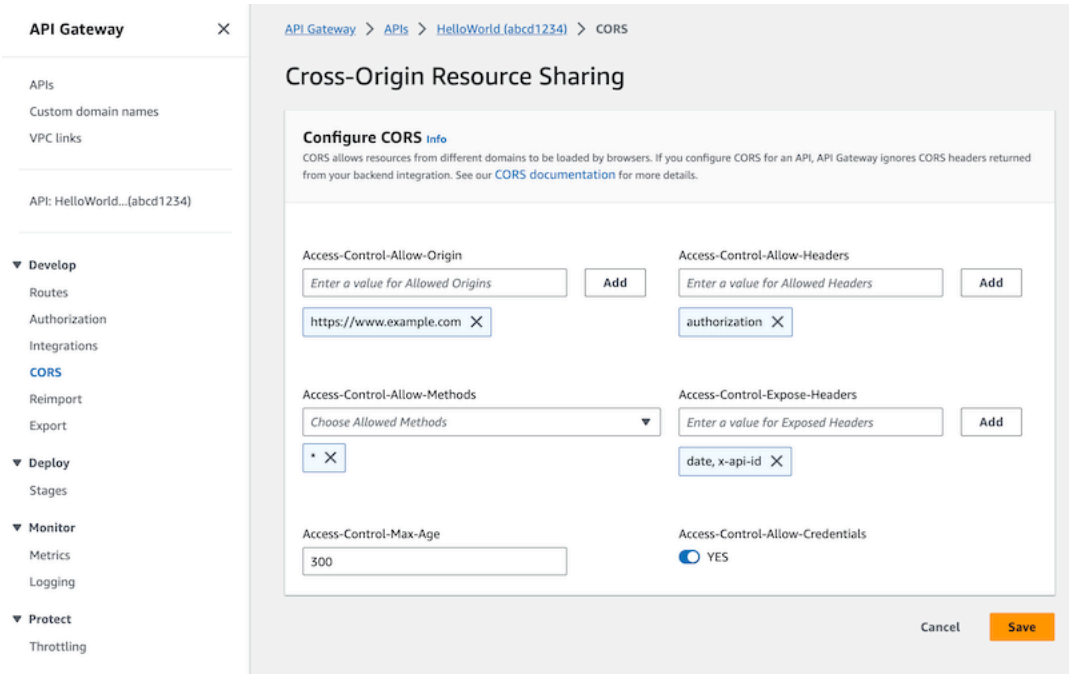
Anda dapat menentukan parameter berikut dalam konfigurasi CORS. Untuk menambahkan parameter ini menggunakan API Gateway HTTP API console, pilih Tambah setelah Anda memasukkan nilai Anda.

Header CORS	Properti konfigurasi CORS	Contoh nilai
Access-Control-Allow-Origin	allowOrigins	• `https://www.example.com` • `*`(izinkan semua asal) • `https://*` (izinkan asal apa pun yang dimulai dengan https://)

Header CORS	Properti konfigurasi CORS	Contoh nilai
		• `http://*`(izinkan asal apa pun yang dimulai dengan http://)
Access-Control-Allow-Credentials	allowCredentials	betul
Access-Control-Expose-Header	exposeHeaders	Tanggal, x-api-id
Akses-Kontrol-Max-Age	maxAge	300
Access-Control-Allow-Methods	allowMethods	DAPATKAN, POSTING, HAPUS, *
Access-Control-Allow-Header	allowHeaders	Otorisasi, *

Untuk mengembalikan header CORS, permintaan Anda harus berisi header. `origin`

Konfigurasi CORS Anda mungkin terlihat mirip dengan yang berikut ini:



Mengkonfigurasi CORS untuk API HTTP dengan **`$default`** rute dan otorisasi JWT

Anda dapat mengaktifkan CORS dan mengonfigurasi otorisasi untuk rute apa pun dari API HTTP. Ketika Anda mengaktifkan CORS dan otorisasi untuk `$defaultrute`, ada beberapa pertimbangan khusus. `$default`Rute menangkap permintaan untuk semua metode dan rute yang belum Anda tetapkan secara eksplisit, termasuk permintaan. `OPTIONS` Untuk mendukung `OPTIONS` permintaan yang tidak sah, tambahkan `OPTIONS /{proxy+}` rute ke API Anda yang tidak memerlukan otorisasi dan lampirkan integrasi ke rute. `OPTIONS /{proxy+}`Rute ini memiliki prioritas lebih tinggi daripada `$default` rute. Akibatnya, ini memungkinkan klien untuk mengirimkan `OPTIONS` permintaan ke API Anda tanpa otorisasi. Untuk informasi selengkapnya tentang prioritas perutean, lihat. Permintaan API perutean

Konfigurasikan CORS untuk API HTTP dengan menggunakan CLI AWS

Anda dapat menggunakan perintah berikut untuk mengaktifkan permintaan CORS dari`https://www.example.com`.

Example

```
aws apigatewayv2 update-api --api-id api-id --cors-configuration AllowOrigins="https://www.example.com"
```

Untuk informasi selengkapnya, lihat CORS di Referensi API Amazon API Gateway Versi 2.

Mengubah permintaan dan tanggapan API

Anda dapat memodifikasi permintaan API dari klien sebelum mereka mencapai integrasi backend Anda. Anda juga dapat mengubah respons dari integrasi sebelum API Gateway mengembalikan respons ke klien. Anda menggunakan pemetaan parameter untuk memodifikasi permintaan API dan respons untuk API HTTP. Untuk menggunakan pemetaan parameter, Anda menentukan permintaan API atau parameter respons untuk dimodifikasi, dan menentukan cara memodifikasi parameter tersebut.

Mengubah permintaan API

Anda menggunakan parameter permintaan untuk mengubah permintaan sebelum mencapai integrasi backend Anda. Anda dapat memodifikasi header, string kueri, atau jalur permintaan.

Parameter permintaan adalah peta nilai kunci. Kunci mengidentifikasi lokasi parameter permintaan untuk diubah, dan bagaimana mengubahnya. Nilai menentukan data baru untuk parameter.

Tabel berikut menunjukkan kunci yang didukung.

Tombol pemetaan parameter

Tipe	Sintaksis
Header	`append|overwrite|remove:header.` *`headername`*
String kueri	`append|overwrite|remove:querystring.` *`querystring-name`*
Jalur	`overwrite:path`

Tabel berikut menunjukkan nilai yang didukung yang dapat Anda petakan ke parameter.

Minta nilai pemetaan parameter

Tipe	Sintaksis	Catatan
Nilai header	$ request.header. *`nama`* atau $ {request.header. *`nama`*}	Nama header tidak peka huruf besar/kecil. API Gateway menggabungkan beberapa nilai header dengan koma, misalnya`"header1": "value1,value2"` . Beberapa header dicadangkan. Untuk mempelajari selengkapnya, lihat the section called “Header yang dipesan”.
Nilai string kueri	$request.querystring. *`nama`* atau $ {request.querystring. *`nama`*}	Nama string kueri peka huruf besar/kecil. API Gateway menggabungkan beberapa nilai dengan koma,

Tipe	Sintaksis	Catatan
		misalnya`"querystring1"` `"Value1,Value2"` .
Isi permintaan	$ request.body. *nama* atau $ {request.body. *nama*}	Ekspresi jalur JSON. Penurunan rekursif (`$request.body..name`) dan ekspresi filter (`?(expression)`) tidak didukung. **Note** Saat Anda menentukan jalur JSON, API Gateway memotong isi permintaan pada 100 KB dan kemudian menerapkan ekspresi seleksi. Untuk mengirim muatan yang lebih besar dari 100 KB, tentukan`$request.body` .
Jalur permintaan	$request.path atau $ {request.path}	Jalur permintaan, tanpa nama panggung.

Tipe	Sintaksis	Catatan
Parameter jalur	$request.path. *nama* atau $ {request.path. *nama*}	Nilai parameter jalur dalam permintaan. Misalnya jika rutenya/ `pets/{petId}` , Anda dapat memetakan `petId` parameter dari permintaa n dengan`$request. path.petId` .
Variabel konteks	$ konteks. *VariableName* atau $ {context. *VariableN ame}*	Nilai variabel konteks. **Note** Hanya karakter khusus `.` dan `_` didukung.
Variabel tahap	$ StageVariables. *VariableN ame* atau $ {stageVariables. *VariableName}*	Nilai variabel tahap.
Nilai statis	*tali*	Nilai konstan.

Note

Untuk menggunakan beberapa variabel dalam ekspresi seleksi, lampirkan variabel dalam tanda kurung. Sebagai contoh, `${request.path.name} ${request.path.id}`.

Mengubah respons API

Anda menggunakan parameter respons untuk mengubah respons HTTP dari integrasi backend sebelum mengembalikan respons ke klien. Anda dapat mengubah header atau kode status respons sebelum API Gateway mengembalikan respons ke klien.

Anda mengonfigurasi parameter respons untuk setiap kode status yang dikembalikan oleh integrasi Anda. Parameter respons adalah peta nilai kunci. Kunci mengidentifikasi lokasi parameter permintaan untuk diubah, dan bagaimana mengubahnya. Nilai menentukan data baru untuk parameter.

Tabel berikut menunjukkan kunci yang didukung.

Tombol pemetaan parameter respons

Tipe	Sintaksis
Header	`append|overwrite|remove:header.` *`headername`*
Kode status	`overwrite:statuscode`

Tabel berikut menunjukkan nilai yang didukung yang dapat Anda petakan ke parameter.

Nilai pemetaan parameter respons

Tipe	Sintaksis	Catatan
Nilai header	$ response.header. *`nama`* atau $ {response.header. *`nama`*}	Nama header tidak peka huruf besar/kecil. API Gateway menggabungkan beberapa nilai header dengan koma, misalnya`"header1": "value1,value2"` . Beberapa header dicadangkan. Untuk mempelajari selengkapnya, lihat the section called “Header yang dipesan”.
Isi respons	$response.body. *`nama`* atau $ {response.body. *`nama`*}	Ekspresi jalur JSON. Descent rekursif (`$response.body..name`) dan ekspresi filter (`?(expression)`) tidak didukung.

Tipe	Sintaksis	Catatan
		Note Saat Anda menentukan jalur JSON, API Gateway memotong badan respons pada 100 KB dan kemudian menerapkan ekspresi pemilihan. Untuk mengirim muatan yang lebih besar dari 100 KB, tentukan `$response.body`.
Variabel konteks	$ konteks. *VariableName* atau $ {context. *VariableName}*	Nilai variabel konteks yang didukung.
Variabel tahap	$ StageVariables. *VariableName* atau $ {stageVariables. *VariableName}*	Nilai variabel tahap.
Nilai statis	*tali*	Nilai konstan.

Note

Untuk menggunakan beberapa variabel dalam ekspresi seleksi, lampirkan variabel dalam tanda kurung. Sebagai contoh, `${request.path.name} ${request.path.id}`.

Header yang dipesan

Header berikut dicadangkan. Anda tidak dapat mengonfigurasi pemetaan permintaan atau respons untuk header ini.

- akses-kontrol-*
- apigw-*
- Otorisasi
- Koneksi
- Pengkodean Konten
- Content-Length
- Content-Location
- Diteruskan
- Jaga-Hidup
- Asal
- Proksi-Otentikasi
- Otorisasi Proksi
- TE
- Trailer
- Transfer-Encoding
- Meningkatkan
- x-amz-*
- x-amzn-*
- X-Diteruskan-Untuk
- X-Forwarded-Host
- X-Diteruskan-Proto
- Melalui

Contoh

AWS CLIContoh berikut mengkonfigurasi pemetaan parameter. Misalnya AWS CloudFormation template, lihat GitHub.

Menambahkan header ke permintaan API

Contoh berikut menambahkan header bernama `header1` ke permintaan API sebelum mencapai integrasi backend Anda. API Gateway mengisi header dengan ID permintaan.

```
aws apigatewayv2 create-integration \
    --api-id abcdef123 \
    --integration-type HTTP_PROXY \
    --payload-format-version 1.0 \
    --integration-uri 'https://api.example.com' \
    --integration-method ANY \
    --request-parameters '{ "append:header.header1": "$context.requestId" }'
```

Ganti nama header permintaan

Contoh berikut mengganti nama header permintaan dari `header1` ke. `header2`

```
aws apigatewayv2 create-integration \
    --api-id abcdef123 \
    --integration-type HTTP_PROXY \
    --payload-format-version 1.0 \
    --integration-uri 'https://api.example.com' \
    --integration-method ANY \
    --request-parameters '{ "append:header.header2": "$request.header.header1",
 "remove:header.header1": "''"}'
```

Mengubah respon dari integrasi

Contoh berikut mengkonfigurasi parameter respons untuk integrasi. Saat integrasi mengembalikan kode status 500, API Gateway mengubah kode status menjadi 403, dan menambahkan `header1` 1 ke respons. Saat integrasi mengembalikan kode status 404, API Gateway menambahkan `error` header ke respons.

```
aws apigatewayv2 create-integration \
    --api-id abcdef123 \
    --integration-type HTTP_PROXY \
    --payload-format-version 1.0 \
    --integration-uri 'https://api.example.com' \
    --integration-method ANY \
    --response-parameters '{"500" : {"append:header.header1": "$context.requestId",
 "overwrite:statuscode" : "403"}, "404" : {"append:header.error" :
 "$stageVariables.environmentId"}  }'
```

Hapus pemetaan parameter yang dikonfigurasi

Contoh perintah berikut menghapus parameter permintaan yang dikonfigurasi sebelumnya untuk`append:header.header1`. Ini juga menghapus parameter respons yang dikonfigurasi sebelumnya untuk kode status 200.

```
aws apigatewayv2 update-integration \
    --api-id abcdef123 \
    --integration-id hijk456 \
    --request-parameters '{"append:header.header1" : ""}' \
    --response-parameters '{"200" : {}}'
```

Bekerja dengan definisi OpenAPI untuk HTTP API

Anda dapat menentukan HTTP API Anda dengan menggunakan file definisi OpenAPI 3.0. Kemudian Anda dapat mengimpor definisi ke API Gateway untuk membuat API. Untuk mempelajari lebih lanjut tentang ekstensi API Gateway ke OpenAPI, lihatEkstensi OpenAPI.

Mengimpor API HTTP

Anda dapat membuat HTTP API dengan mengimpor file definisi OpenAPI 3.0.

Untuk bermigrasi dari REST API ke API HTTP, Anda dapat mengekspor REST API Anda sebagai file definisi OpenAPI 3.0. Kemudian impor definisi API sebagai HTTP API. Untuk mempelajari tentang mengekspor REST API, lihatEkspor REST API dari API Gateway.

> **Note**
>
> API HTTP mendukung hal yang samaAWSvariabel sebagai API REST. Untuk mempelajari selengkapnya, lihat AWSvariabel untuk impor OpenAPI.

Informasi validasi

Saat Anda mengimpor API, API Gateway menyediakan tiga kategori informasi validasi.

INFO

Properti berlaku sesuai dengan spesifikasi OpenAPI, tetapi properti itu tidak didukung untuk API HTTP.

Misalnya, cuplikan OpenAPI 3.0 berikut ini menghasilkan info tentang impor karena API HTTP tidak mendukung validasi permintaan. API Gateway mengabaikan`requestBody`dan`schema`bidang.

```
"paths": {
  "/": {
    "get": {
      "x-amazon-apigateway-integration": {
        "type": "AWS_PROXY",
        "httpMethod": "POST",
        "uri": "arn:aws:lambda:us-east-2:123456789012:function:HelloWorld",
        "payloadFormatVersion": "1.0"
      },
      "requestBody": {
        "content": {
          "application/json": {
            "schema": {
              "$ref": "#/components/schemas/Body"
            }
          }
        }
      }
    }
  }
  ...
},
"components": {
  "schemas": {
    "Body": {
      "type": "object",
      "properties": {
        "key": {
          "type": "string"
        }
      }
    }
    ...
  }
  ...
}
```

Peringatan

Properti atau struktur tidak valid sesuai dengan spesifikasi OpenAPI, tetapi tidak memblokir pembuatan API. Anda dapat menentukan apakah API Gateway harus mengabaikan peringatan ini dan terus membuat API, atau berhenti membuat API pada peringatan.

Dokumen OpenAPI 3.0 berikut menghasilkan peringatan tentang impor karena API HTTP hanya mendukung integrasi proxy Lambda dan proxy HTTP.

```
"x-amazon-apigateway-integration": {
  "type": "AWS",
  "httpMethod": "POST",
  "uri": "arn:aws:lambda:us-east-2:123456789012:function:HelloWorld",
  "payloadFormatVersion": "1.0"
}
```

Kesalahan

Spesifikasi OpenAPI tidak valid atau cacat. API Gateway tidak dapat membuat sumber daya apa pun dari dokumen cacat. Anda harus memperbaiki kesalahan, kemudian coba lagi.

Definisi API berikut menghasilkan kesalahan pada impor karena API HTTP hanya mendukung spesifikasi OpenAPI 3.0.

```
{
  "swagger": "2.0.0",
  "info": {
    "title": "My API",
    "description": "An Example OpenAPI definition for Errors/Warnings/ImportInfo",
    "version": "1.0"
  }
  ...
}
```

Sebagai contoh lain, sementara OpenAPI memungkinkan pengguna untuk mendefinisikan API dengan beberapa persyaratan keamanan yang melekat pada operasi tertentu, API Gateway tidak mendukung ini. Setiap operasi hanya dapat memiliki satu otorisasi IAM, otorisasi Lambda, atau otorisasi JWT. Mencoba memodelkan beberapa persyaratan keamanan menghasilkan kesalahan.

Mengimpor API dengan menggunakanAWS CLI

Perintah berikut mengimpor file definisi OpenAPI 3.0`api-definition.json`sebagai HTTP API.

Example

```
aws apigatewayv2 import-api --body file://api-definition.json
```

Example

Anda dapat mengimpor contoh berikut OpenAPI 3.0 definisi untuk membuat HTTP API.

```
{
  "openapi": "3.0.1",
  "info": {
    "title": "Example Pet Store",
    "description": "A Pet Store API.",
    "version": "1.0"
  },
  "paths": {
    "/pets": {
      "get": {
        "operationId": "GET HTTP",
        "parameters": [
          {
            "name": "type",
            "in": "query",
            "schema": {
              "type": "string"
            }
          },
          {
            "name": "page",
            "in": "query",
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "headers": {
              "Access-Control-Allow-Origin": {
```

```
              "schema": {
                "type": "string"
              }
            }
          },
          "content": {
            "application/json": {
              "schema": {
                "$ref": "#/components/schemas/Pets"
              }
            }
          }
        }
      },
      "x-amazon-apigateway-integration": {
        "type": "HTTP_PROXY",
        "httpMethod": "GET",
        "uri": "http://petstore.execute-api.us-west-1.amazonaws.com/petstore/pets",
        "payloadFormatVersion": 1.0
      }
    },
    "post": {
      "operationId": "Create Pet",
      "requestBody": {
        "content": {
          "application/json": {
            "schema": {
              "$ref": "#/components/schemas/NewPet"
            }
          }
        },
        "required": true
      },
      "responses": {
        "200": {
          "description": "200 response",
          "headers": {
            "Access-Control-Allow-Origin": {
              "schema": {
                "type": "string"
              }
            }
          },
          "content": {
```

```
            "application/json": {
              "schema": {
                "$ref": "#/components/schemas/NewPetResponse"
              }
            }
          }
        }
      },
      "x-amazon-apigateway-integration": {
        "type": "HTTP_PROXY",
        "httpMethod": "POST",
        "uri": "http://petstore.execute-api.us-west-1.amazonaws.com/petstore/pets",
        "payloadFormatVersion": 1.0
      }
    }
  },
  "/pets/{petId}": {
    "get": {
      "operationId": "Get Pet",
      "parameters": [
        {
          "name": "petId",
          "in": "path",
          "required": true,
          "schema": {
            "type": "string"
          }
        }
      ],
      "responses": {
        "200": {
          "description": "200 response",
          "headers": {
            "Access-Control-Allow-Origin": {
              "schema": {
                "type": "string"
              }
            }
          },
          "content": {
            "application/json": {
              "schema": {
                "$ref": "#/components/schemas/Pet"
              }
```

```
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "HTTP_PROXY",
          "httpMethod": "GET",
          "uri": "http://petstore.execute-api.us-west-1.amazonaws.com/petstore/pets/
{petId}",
          "payloadFormatVersion": 1.0
        }
      }
    }
  },
  "x-amazon-apigateway-cors": {
    "allowOrigins": [
      "*"
    ],
    "allowMethods": [
      "GET",
      "OPTIONS",
      "POST"
    ],
    "allowHeaders": [
      "x-amzm-header",
      "x-apigateway-header",
      "x-api-key",
      "authorization",
      "x-amz-date",
      "content-type"
    ]
  },
  "components": {
    "schemas": {
      "Pets": {
        "type": "array",
        "items": {
          "$ref": "#/components/schemas/Pet"
        }
      },
      "Empty": {
        "type": "object"
      },
      "NewPetResponse": {
```

```
        "type": "object",
        "properties": {
          "pet": {
            "$ref": "#/components/schemas/Pet"
          },
          "message": {
            "type": "string"
          }
        }
      },
      "Pet": {
        "type": "object",
        "properties": {
          "id": {
            "type": "string"
          },
          "type": {
            "type": "string"
          },
          "price": {
            "type": "number"
          }
        }
      },
      "NewPet": {
        "type": "object",
        "properties": {
          "type": {
            "$ref": "#/components/schemas/PetType"
          },
          "price": {
            "type": "number"
          }
        }
      },
      "PetType": {
        "type": "string",
        "enum": [
          "dog",
          "cat",
          "fish",
          "bird",
          "gecko"
        ]
```

```
      }
    }
  }
}
```

Mengekspor API HTTP dari API Gateway

Setelah membuat API HTTP, Anda dapat mengekspor definisi OpenAPI 3.0 API Anda dari API Gateway. Anda dapat memilih tahap untuk mengekspor, atau mengekspor konfigurasi terbaru API Anda. Anda juga dapat mengimpor definisi API yang diekspor ke API Gateway untuk membuat API lain yang identik. Untuk mempelajari lebih lanjut tentang mengimpor definisi API, lihatMengimpor API HTTP.

Ekspor definisi OpenAPI 3.0 dari sebuah tahap dengan menggunakan CLI AWS

Perintah berikut mengekspor definisi OpenAPI dari tahap API yang diberi nama ke file YAMAL `prod` bernama. `stage-definition.yaml` File definisi yang diekspor menyertakan ekstensi API Gateway secara default.

```
aws apigatewayv2 export-api \
    --api-id api-id  \
    --output-type YAML  \
    --specification OAS30 \
    --stage-name prod \
    stage-definition.yaml
```

Ekspor definisi OpenAPI 3.0 dari perubahan terbaru API Anda dengan menggunakan CLI AWS

Perintah berikut mengekspor definisi OpenAPI dari API HTTP ke file JSON bernama. `latest-api-definition.json` Karena perintah tidak menentukan tahapan, API Gateway mengekspor konfigurasi terbaru API Anda, apakah itu telah diterapkan ke panggung atau belum. File definisi yang diekspor tidak menyertakan ekstensi API Gateway.

```
aws apigatewayv2 export-api \
    --api-id api-id  \
    --output-type JSON  \
    --specification OAS30 \
    --no-include-extensions \
    latest-api-definition.json
```

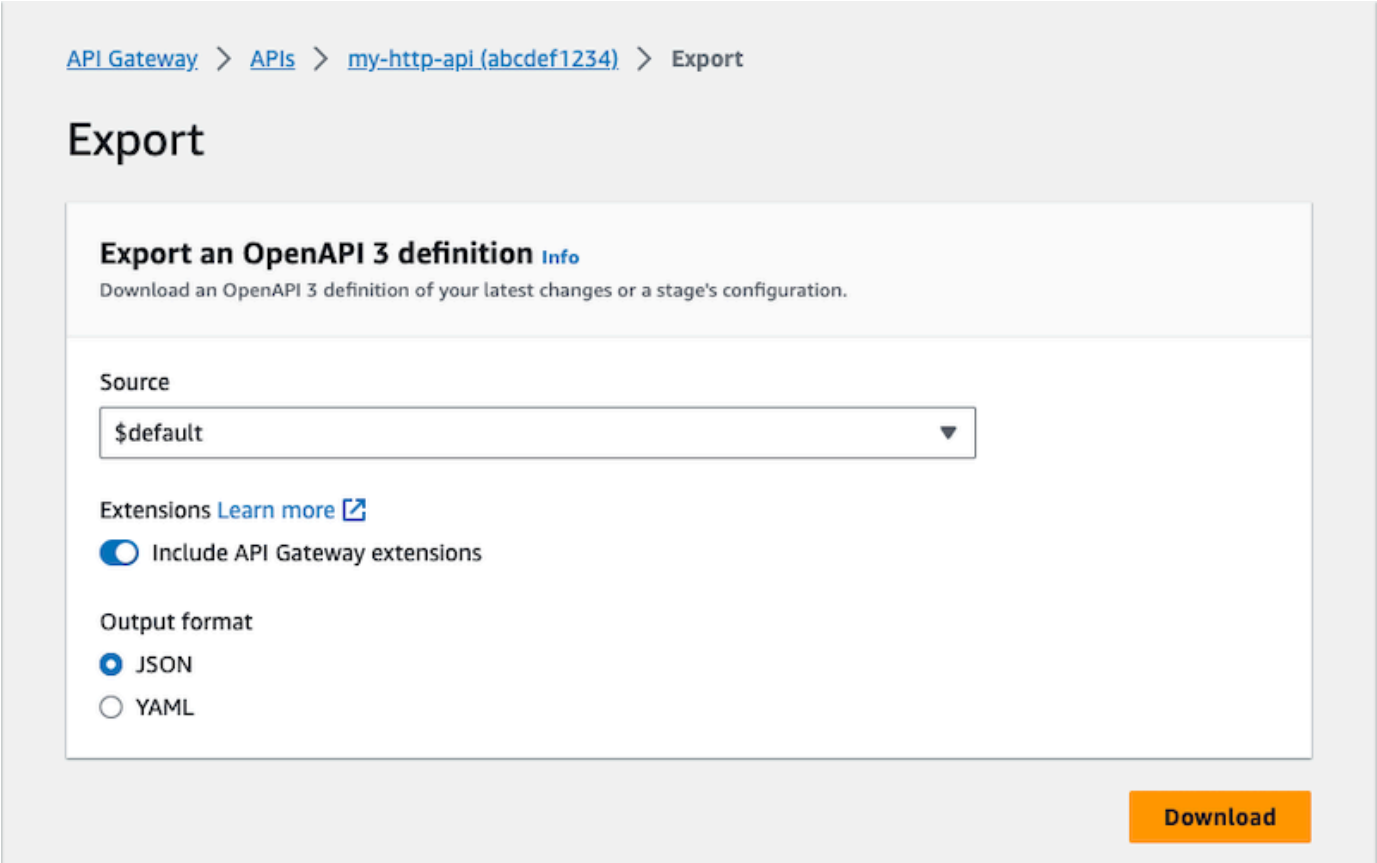
Untuk informasi selengkapnya, lihat ExportAPI di Referensi API Amazon API Gateway Versi 2.

Ekspor definisi OpenAPI 3.0 menggunakan konsol API Gateway

Prosedur berikut menunjukkan cara mengekspor definisi OpenAPI dari API HTTP.

Untuk mengekspor definisi OpenAPI 3.0 menggunakan konsol API Gateway

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih API HTTP.
3. Pada panel navigasi utama, di bawah Kembangkan, pilih Ekspor.
4. Pilih dari opsi berikut untuk mengekspor API Anda:



   a. Untuk Sumber, pilih sumber untuk definisi OpenAPI 3.0. Anda dapat memilih tahap untuk mengekspor, atau mengekspor konfigurasi terbaru API Anda.
   b. Aktifkan Sertakan ekstensi API Gateway untuk menyertakan ekstensi API Gateway.
   c. Untuk format Output, pilih format output.
5. Pilih Unduh.

Menerbitkan API HTTP untuk dipanggil pelanggan

Anda dapat menggunakan tahapan dan nama domain khusus untuk mempublikasikan API agar klien dapat dipanggil.

Tahap API adalah referensi logis ke status siklus hidup API Anda (misalnya,,, `dev` `prod`beta, atauv2). Setiap tahap adalah referensi bernama untuk deployment API dan dibuat tersedia bagi aplikasi klien untuk dipanggil. Anda dapat mengonfigurasi integrasi dan pengaturan yang berbeda untuk setiap tahap API.

Anda dapat menggunakan nama domain khusus untuk memberikan URL yang lebih sederhana dan lebih intuitif bagi klien untuk memanggil API Anda daripada URL default. `https://`*`api-id`*`.execute-api.`*`region`*`.amazonaws.com/`*`stage`*

> **Note**
>
> Untuk meningkatkan keamanan API Gateway API Anda, `execute-api.{`*`region`*`}.amazonaws.com` domain tersebut terdaftar di Daftar Akhiran Publik (PSL). Untuk keamanan lebih lanjut, kami menyarankan Anda menggunakan cookie dengan `__Host-` awalan jika Anda perlu mengatur cookie sensitif di nama domain default untuk API Gateway API Anda. Praktik ini akan membantu mempertahankan domain Anda dari upaya pemalsuan permintaan lintas situs (CSRF). Untuk informasi selengkapnya, lihat halaman Set-Cookie di Jaringan Pengembang Mozilla.

Topik

- Bekerja dengan tahap untuk API HTTP
- Kebijakan keamanan untuk HTTP API
- Menyiapkan nama domain khusus untuk HTTP API

Bekerja dengan tahap untuk API HTTP

Tahap API adalah referensi logis ke status siklus hidup API Anda (misalnya,,, `dev` `prod`beta, atauv2). Tahapan API diidentifikasi oleh ID API dan nama stage, dan mereka disertakan dalam URL yang Anda gunakan untuk memanggil API. Setiap tahap adalah referensi bernama untuk deployment API dan dibuat tersedia bagi aplikasi klien untuk dipanggil.

Anda dapat membuat `$default` stage yang disajikan dari dasar URL API Anda—misalnya,. `https://{api_id}.execute-api.{region}.amazonaws.com/` Anda menggunakan URL ini untuk memanggil tahap API.

Deployment adalah snapshot dari konfigurasi API Anda. Setelah Anda menerapkan API ke tahap, tersedia bagi klien untuk dipanggil. Anda harus menerapkan API agar perubahan diterapkan. Jika Anda mengaktifkan penerapan otomatis, perubahan pada API akan dirilis secara otomatis untuk Anda.

Variabel tahap

Variabel tahap kunci-nilai yang dapat Anda tentukan untuk tahap API HTTP. Mereka bertindak seperti variabel lingkungan dan dapat digunakan dalam pengaturan API Anda.

Misalnya, Anda dapat menentukan variabel stage, dan kemudian menetapkan nilainya sebagai endpoint HTTP untuk integrasi proxy HTTP. Kemudian, Anda dapat referensi endpoint dengan menggunakan nama variabel tahap terkait. Dengan melakukan ini, Anda dapat menggunakan pengaturan API yang sama dengan endpoint akhir yang berbeda pada setiap tahap. Demikian pula, Anda dapat menggunakan variabel tahap untuk menentukan integrasi AWS Lambda fungsi yang berbeda untuk setiap tahap API Anda.

> **Note**
>
> Variabel tahap tidak dimaksudkan untuk digunakan untuk data sensitif, seperti kredensial. Untuk meneruskan data sensitif ke integrasi, gunakan AWS Lambda otorisasi. Anda dapat meneruskan data sensitif ke integrasi dalam output otorisasi Lambda. Untuk mempelajari selengkapnya, lihat the section called “Format respons otorisasi Lambda”.

Contoh

Untuk menggunakan variabel stage untuk menyesuaikan endpoint integrasi HTTP, Anda harus terlebih dahulu menetapkan nama dan nilai variabel stage (misalnya,`url`) dengan nilai`example.com`. Selanjutnya, siapkan integrasi proxy HTTP. Alih-alih memasukkan URL endpoint, Anda dapat memberi tahu API Gateway untuk menggunakan nilai variabel stage,**`http://${stageVariables.url}`**. Nilai ini memberi tahu API Gateway untuk mengganti variabel stage Anda `${}` saat runtime, tergantung pada tahap API Anda.

Anda dapat mereferensikan variabel tahap dengan cara yang sama untuk menentukan nama fungsi Lambda atau ARN peranAWS.

Saat menentukan nama fungsi Lambda sebagai nilai variabel tahap, Anda harus mengonfigurasi izin pada fungsi Lambda secara manual. Anda dapat menggunakan AWS Command Line Interface (AWS CLI) untuk melakukan hal ini.

```
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
```

Referensi variabel tahap API Gateway

URI integrasi HTTP

Anda dapat menggunakan variabel tahapan sebagai bagian dari URI integrasi HTTP, seperti yang ditunjukkan dalam contoh berikut.

- URI lengkap tanpa protokol — `http://${stageVariables.<variable_name>}`
- Domain lengkap — `http://${stageVariables.<variable_name>}/resource/operation`
- Sebuah subdomain — `http://${stageVariables.<variable_name>}.example.com/resource/operation`
- Sebuah jalan — `http://example.com/${stageVariables.<variable_name>}/bar`
- Sebuah string kueri - `http://example.com/foo?q=${stageVariables.<variable_name>}`

Fungsi Lambda

Anda dapat menggunakan variabel stage sebagai pengganti nama integrasi fungsi Lambda atau alias, seperti yang ditunjukkan pada contoh berikut.

- `arn:aws:apigateway:<region>:lambda:path/2015-03-31/functions/arn:aws:lambda:<region>:<account_id>:function:${stageVariables.<function_variable_name>}/invocations`
- `arn:aws:apigateway:<region>:lambda:path/2015-03-31/functions/arn:aws:lambda:<region>:<account_id>:function:<function_name>:${stageVariables.<version_variable_name>}/invocations`

> **Note**
>
> Untuk menggunakan variabel tahapan untuk function Lambda, fungsinya harus berada dalam akun tahapan (API tahapan). Variabel tahap tidak support lintas akun Lambda.

AWSkredensi integrasi

Anda dapat menggunakan variabel tahapan sebagai bagian dari AWS pengguna atau role credential ARN, seperti yang ditunjukkan dalam contoh berikut.

- `arn:aws:iam::<account_id>:${stageVariables.<variable_name>}`

Kebijakan keamanan untuk HTTP API

API Gateway memberlakukan kebijakan keamanan `TLS_1_2` untuk semua titik akhir HTTP API.

Kebijakan keamanan adalah kombinasi standar dari versi TLS minimum dan cipher suite yang ditawarkan oleh Amazon API Gateway. Protokol TLS mengatasi masalah keamanan jaringan seperti gangguan dan penyadapan antara klien dan server. Ketika klien Anda membuat jabat tangan TLS ke API Anda melalui domain kustom, kebijakan keamanan memberlakukan versi TLS dan pilihan cipher suite yang dapat dipilih klien Anda untuk digunakan. Kebijakan keamanan ini menerima lalu lintas TLS 1.2 dan TLS 1.3 dan menolak lalu lintas TLS 1.0.

Protokol dan cipher TLS yang didukung untuk API HTTP

Tabel berikut menjelaskan protokol dan cipher TLS yang didukung untuk API HTTP.

Kebijakan keamanan	TLS_1_2
Protokol TLS	
TLSv1.3	♦
TLSv1.2	♦
Cipher TLS	
TLS-AES-128-GCM-SHA256	♦

Kebijakan keamanan	TLS_1_2
TLS-AES-256-GCM-SHA384	♦
TLS-CHACHA20-POLY1305-SHA256	♦
ECDHE-ECDSA-AES128-GCM-SHA256	♦
ECDHE-RSA-AES128-GCM-SHA256	♦
ECDHE-ECDSA-AES128-SHA256	♦
ECDHE-RSA-AES128-SHA256	♦
ECDHE-ECDSA-AES256-GCM-SHA384	♦
ECDHE-RSA-AES256-GCM-SHA384	♦
ECDHE-ECDSA-AES256-SHA384	♦
ECDHE-RSA-AES256-SHA384	♦
AES128-GCM-SHA256	♦
AES128-SHA256	♦
AES256-GCM-SHA384	♦
AES256-SHA256	♦

Nama sandi OpenSSL dan RFC

OpenSSL dan IETF RFC 5246 menggunakan nama yang berbeda untuk cipher yang sama. Untuk daftar nama sandi, lihat. the section called “Nama sandi OpenSSL dan RFC”

Informasi tentang REST API dan WebSocket API

Untuk informasi selengkapnya tentang REST WebSocket API dan API, lihat the section called “Memilih kebijakan keamanan” danthe section called “Kebijakan keamanan untuk WebSocket API”.

Menyiapkan nama domain khusus untuk HTTP API

Nama domain khusus adalah URL yang lebih sederhana dan lebih intuitif yang dapat Anda berikan kepada pengguna API Anda.

Setelah menerapkan API Anda, Anda (dan pelanggan Anda) dapat memanggil API menggunakan URL dasar default dari format berikut:

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

dimana `api-id` dihasilkan oleh API Gateway, `region` (AWS Wilayah) ditentukan oleh Anda saat membuat API, dan `stage` ditentukan oleh Anda saat menerapkan API.

Bagian nama host dari URL (yaitu,*`api-id`*`.execute-api.`*`region`*`.amazonaws.com`) mengacu pada titik akhir API. Titik akhir API default bisa sulit diingat dan tidak ramah pengguna.

Dengan nama domain khusus, Anda dapat mengatur nama host API Anda, dan memilih jalur dasar (misalnya,`myservice`) untuk memetakan URL alternatif ke API Anda. Misalnya, URL dasar API yang lebih ramah pengguna dapat menjadi:

```
https://api.example.com/myservice
```

> **Note**
>
> Domain kustom dapat dikaitkan dengan REST API dan HTTP API. Anda dapat menggunakan API Gateway Versi 2 API untuk membuat dan mengelola nama domain kustom Regional untuk REST API dan API HTTP.
> Untuk HTTP API, TLS 1.2 adalah satu-satunya versi TLS yang didukung.

Daftarkan nama domain

Anda harus memiliki nama domain internet terdaftar untuk menyiapkan nama domain khusus untuk API Anda. Jika diperlukan, Anda dapat mendaftarkan domain internet menggunakan Amazon Route 53 atau menggunakan registrar domain pihak ketiga pilihan Anda. Nama domain kustom API dapat berupa nama subdomain atau domain root (juga dikenal sebagai “zone apex”) dari domain internet terdaftar.

Setelah nama domain kustom dibuat di API Gateway, Anda harus membuat atau memperbarui catatan sumber daya penyedia DNS Anda untuk dipetakan ke titik akhir API Anda. Tanpa pemetaan seperti itu, permintaan API yang terikat untuk nama domain khusus tidak dapat mencapai API Gateway.

Nama domain kustom regional

Saat Anda membuat nama domain khusus untuk API Regional, API Gateway akan membuat nama domain Regional untuk API. Anda harus menyiapkan catatan DNS untuk memetakan nama domain kustom ke nama domain Regional. Anda juga harus memberikan sertifikat untuk nama domain kustom.

Nama domain kustom wildcard

Dengan nama domain khusus wildcard, Anda dapat mendukung jumlah nama domain yang hampir tak terbatas tanpa melebihi kuota default. Misalnya, Anda bisa memberi setiap pelanggan Anda nama domain mereka sendiri*customername*`.api.example.com`.

Untuk membuat nama domain kustom wildcard, tentukan wildcard (`*`) sebagai subdomain pertama dari domain kustom yang mewakili semua kemungkinan subdomain dari domain root.

Misalnya, nama domain kustom wildcard `*.example.com` menghasilkan subdomain sepertia`.example.com`,, dan `b.example.comc.example.com`, yang semuanya merutekan ke domain yang sama.

Nama domain kustom wildcard mendukung konfigurasi yang berbeda dari nama domain kustom standar API Gateway. Misalnya, dalam satu AWS akun, Anda dapat mengkonfigurasi `*.example.com` dan `a.example.com` berperilaku berbeda.

Untuk membuat nama domain kustom wildcard, Anda harus memberikan sertifikat yang dikeluarkan oleh ACM yang telah divalidasi menggunakan DNS atau metode validasi email.

> **Note**
>
> Anda tidak dapat membuat nama domain khusus wildcard jika AWS akun lain telah membuat nama domain kustom yang bertentangan dengan nama domain kustom wildcard. Misalnya, jika akun A telah dibuata`.example.com`, maka akun B tidak dapat membuat nama `*.example.com` domain khusus wildcard.
> Jika akun A dan akun B berbagi pemilik, Anda dapat menghubungi Pusat AWS Dukungan untuk meminta pengecualian.

Sertifikat untuk nama domain kustom

> **Important**
>
> Anda menentukan sertifikat untuk nama domain kustom Anda. Jika aplikasi Anda menggunakan pinning sertifikat, kadang-kadang dikenal sebagai penyematan SSL, untuk menyematkan sertifikat ACM, aplikasi mungkin tidak dapat terhubung ke domain Anda setelah AWS memperbarui sertifikat. Untuk informasi selengkapnya, lihat Masalah penyematan sertifikat di Panduan AWS Certificate Manager Pengguna.

Untuk memberikan sertifikat untuk nama domain kustom di Wilayah di mana ACM didukung, Anda harus meminta sertifikat dari ACM. Untuk memberikan sertifikat untuk nama domain kustom Regional di Wilayah di mana ACM tidak didukung, Anda harus mengimpor sertifikat ke API Gateway di Wilayah tersebut.

Untuk mengimpor sertifikat SSL/TLS, Anda harus menyediakan badan sertifikat SSL/TLS yang diformat PEM, kunci pribadinya, dan rantai sertifikat untuk nama domain kustom. Setiap sertifikat yang disimpan dalam ACM diidentifikasi oleh ARN-nya. Untuk menggunakan sertifikat AWS terkelola untuk nama domain, Anda cukup mereferensikan ARN-nya.

ACM membuatnya mudah untuk mengatur dan menggunakan nama domain khusus untuk API. Anda membuat sertifikat untuk nama domain yang diberikan (atau mengimpor sertifikat), menyiapkan nama domain di API Gateway dengan ARN sertifikat yang disediakan oleh ACM, dan memetakan jalur dasar di bawah nama domain kustom ke tahap API yang diterapkan. Dengan sertifikat yang dikeluarkan oleh ACM, Anda tidak perlu khawatir mengekspos detail sertifikat sensitif apa pun, seperti kunci pribadi.

Untuk detail tentang menyiapkan nama domain kustom, lihat Mendapatkan sertifikat siap diAWS Certificate Manager danMenyiapkan nama domain kustom regional di API Gateway.

Bekerja dengan pemetaan API untuk API HTTP

Anda menggunakan pemetaan API untuk menghubungkan tahapan API ke nama domain khusus. Setelah membuat nama domain dan mengonfigurasi catatan DNS, Anda menggunakan pemetaan API untuk mengirim lalu lintas ke API melalui nama domain kustom Anda.

Pemetaan API menentukan API, tahap, dan jalur opsional yang akan digunakan untuk pemetaan. Misalnya, Anda dapat memetakan`production`tahap API ke`https://api.example.com/orders`.

Anda dapat memetakan tahap HTTP dan REST API ke nama domain kustom yang sama.

Sebelum membuat pemetaan API, Anda harus memiliki API, panggung, dan nama domain khusus. Untuk mempelajari selengkapnya tentang membuat nama domain kustom, lihatthe section called "Menyiapkan nama domain kustom regional".

Permintaan API perutean

Anda dapat mengonfigurasi pemetaan API dengan beberapa level, misalnya`orders/v1/items`dan`orders/v2/items`.

Untuk pemetaan API dengan beberapa level, API Gateway merutekan permintaan ke pemetaan API yang memiliki jalur pencocokan terpanjang. API Gateway hanya mempertimbangkan jalur yang dikonfigurasi untuk pemetaan API, dan bukan rute API, untuk memilih API yang akan dipanggil. Jika tidak ada jalur yang cocok dengan permintaan, API Gateway mengirimkan permintaan ke API yang telah Anda petakan ke jalur kosong`(none)`.

Untuk nama domain kustom yang menggunakan pemetaan API dengan beberapa level, API Gateway merutekan permintaan ke pemetaan API yang memiliki awalan pencocokan terpanjang.

Misalnya, pertimbangkan nama domain khusus`https://api.example.com`dengan pemetaan API berikut:

1. `(none)`dipetakan ke API 1.
2. `orders`dipetakan ke API 2.
3. `orders/v1/items`dipetakan ke API 3.
4. `orders/v2/items`dipetakan ke API 4.
5. `orders/v2/items/categories`dipetakan ke API 5.

Permintaan	API yang dipilih	Penjelasan
`https://api.example.com/orders`	`API 2`	Permintaan sama persis dengan pemetaan API ini.
`https://api.example.com/orders/v1/items`	`API 3`	Permintaan sama persis dengan pemetaan API ini.

Permintaan	API yang dipilih	Penjelasan
`https://api.example.com/orders/v2/items`	`API 4`	Permintaan sama persis dengan pemetaan API ini.
`https://api.example.com/orders/v1/items/123`	`API 3`	API Gateway memilih pemetaan yang memiliki jalur pencocokan terpanjang. The`123`di akhir permintaan tidak mempengaruhi pemilihan .
`https://api.example.com/orders/v2/items/categories/5`	`API 5`	API Gateway memilih pemetaan yang memiliki jalur pencocokan terpanjang.
`https://api.example.com/customers`	`API 1`	API Gateway menggunakan pemetaan kosong sebagai catch-all.
`https://api.example.com/ordersandmore`	`API 2`	API Gateway memilih pemetaan yang memiliki awalan pencocokan terpanjang. Untuk nama domain kustom yang dikonfigurasi dengan pemetaan satu tingkat, seperti saja`https://api.example.com/orders` dan`https://api.example.com/` , API Gateway akan memilih`API 1`, karena tidak ada jalur yang cocok dengan`ordersandmore` .

Pembatasan

- Dalam pemetaan API, nama domain khusus dan API yang dipetakan harus samaAWSakun.
- Pemetaan API harus hanya berisi huruf, angka, dan karakter berikut:`$-_.+!*'()/`.
- Panjang maksimum jalur dalam pemetaan API adalah 300 karakter.
- Anda dapat memiliki 200 pemetaan API dengan beberapa level untuk setiap nama domain.
- Anda hanya dapat memetakan API HTTP ke nama domain kustom regional dengan kebijakan keamanan TLS 1.2.
- Anda tidak dapat memetakanWebSocketAPI ke nama domain kustom yang sama dengan API HTTP atau REST API.

Buat pemetaan API

Untuk membuat pemetaan API, Anda harus terlebih dahulu membuat nama domain kustom, API, dan stage. Untuk informasi tentang membuat nama domain kustom, lihatthe section called “Menyiapkan nama domain kustom regional”.

Sebagai contohAWS Serverless Application Modeltemplate yang membuat semua sumber daya, lihatSesi Dengan SAMdi atasGitHub.

AWS Management Console

Untuk membuat pemetaan API

1. Masuk ke konsol API Gateway dihttps://console.aws.amazon.com/apigateway.
2. PilihNama domain kustom.
3. Pilih nama domain khusus yang sudah Anda buat.
4. PilihPemetaan API.
5. PilihKonfigurasikan pemetaan API.
6. PilihTambahkan pemetaan baru.
7. Masukkan sebuahAPI, sebuahPanggung, dan secara opsional aJalan.
8. Pilih Save (Simpan).

AWS CLI

Berikut iniAWS CLIperintah membuat pemetaan API. Dalam contoh ini, API Gateway mengirimkan permintaan ke`api.example.com/v1/orders`ke API dan panggung yang ditentukan.

```
aws apigatewayv2 create-api-mapping \
    --domain-name api.example.com \
    --api-mapping-key v1/orders \
    --api-id a1b2c3d4 \
    --stage test
```

AWS CloudFormation

Berikut iniAWS CloudFormationcontoh membuat pemetaan API.

```
MyApiMapping:
  Type: 'AWS::ApiGatewayV2::ApiMapping'
  Properties:
    DomainName: api.example.com
    ApiMappingKey: 'orders/v2/items'
    ApiId: !Ref MyApi
    Stage: !Ref MyStage
```

Menonaktifkan titik akhir default untuk API HTTP

Secara default, klien dapat memanggil API dengan menggunakan`execute-api`endpoint yang dihasilkan API Gateway untuk API Anda. Untuk memastikan bahwa klien dapat mengakses API Anda hanya dengan menggunakan nama domain kustom, nonaktifkan default`execute-api`titik akhir.

> **Note**
>
> Ketika Anda menonaktifkan endpoint default, itu akan mempengaruhi semua tahapan API.

BerikutAWS CLIperintah menonaktifkan titik akhir default untuk API HTTP.

```
aws apigatewayv2 update-api \
    --api-id abcdef123 \
```

```
    --disable-execute-api-endpoint
```

Setelah menonaktifkan endpoint default, Anda harus menerapkan API agar perubahan berlaku, kecuali penerapan otomatis diaktifkan.

BerikutAWS CLIperintah menciptakan deployment.

```
aws apigatewayv2 create-deployment \
    --api-id abcdef123 \
    --stage-name dev
```

Melindungi API HTTP Anda

API Gateway menyediakan sejumlah cara untuk melindungi API Anda dari ancaman tertentu, seperti pengguna berbahaya atau lonjakan lalu lintas. Anda dapat melindungi API Anda menggunakan strategi seperti menetapkan target throttling, dan mengaktifkan TLS bersama. Pada bagian ini Anda dapat mempelajari cara mengaktifkan kemampuan ini menggunakan API Gateway.

Topik

- Pembatasan permintaan ke API HTTP Anda
- Mengkonfigurasi otentikasi TLS timbal balik untuk API HTTP

Pembatasan permintaan ke API HTTP Anda

Anda dapat mengkonfigurasi throttling untuk API Anda untuk membantu melindungi mereka dari kewalahan oleh terlalu banyak permintaan. Throttle diterapkan berdasarkan upaya terbaik dan harus dianggap sebagai target daripada langit-langit permintaan yang dijamin.

API Gateway membatasi permintaan ke API Anda menggunakan algoritma token bucket, di mana token dihitung untuk permintaan. Secara khusus, API Gateway memeriksa tarif dan ledakan pengiriman permintaan terhadap semua API di akun Anda, per Wilayah. Dalam algoritma token bucket, burst dapat memungkinkan overrun yang telah ditentukan sebelumnya dari batas-batas tersebut, tetapi faktor lain juga dapat menyebabkan batasan untuk dikuasai dalam beberapa kasus.

Ketika pengiriman permintaan melebihi tingkat permintaan steady-state dan batas burst, API Gateway mulai membatasi permintaan. Klien mungkin menerima respons`429 Too Many Requests` kesalahan pada saat ini. Setelah menangkap pengecualian tersebut, klien dapat mengirimkan kembali permintaan yang gagal dengan cara yang membatasi tarif.

Sebagai pengembang API, Anda dapat menetapkan batas target untuk setiap tahapan API atau rute untuk meningkatkan kinerja keseluruhan di semua API di akun Anda.

Pelambatan tingkat akun per Wilayah

Secara default, API Gateway membatasi permintaan steady-state per detik (RPS) di semua API dalamAWS akun, per Wilayah. Ini juga membatasi burst (yaitu, ukuran bucket maksimum) di semua API dalamAWS akun, per Wilayah. Di API Gateway, batas burst mewakili jumlah maksimum target kiriman permintaan bersamaan yang akan dipenuhi API Gateway sebelum mengembalikan respons`429 Too Many Requests` kesalahan. Untuk informasi selengkapnya tentang kuota pelambatan, lihatKuota dan catatan penting.

Batas per akun diterapkan ke semua API di akun di Wilayah tertentu. Batas tingkat tingkat akun dapat ditingkatkan berdasarkan permintaan - batas yang lebih tinggi dimungkinkan dengan API yang memiliki waktu tunggu lebih pendek dan muatan yang lebih kecil. Untuk meminta peningkatan batas pembatasan tingkat akun per Wilayah, hubungi PusatAWS Support. Untuk informasi selengkapnya, lihat Kuota dan catatan penting. Perhatikan bahwa batas ini tidak boleh lebih tinggi dari batasAWS pembatasan.

Pelambatan tingkat rute

Anda dapat mengatur pembatasan tingkat rute untuk mengesampingkan batas pembatasan permintaan tingkat akun untuk tahap tertentu atau untuk rute individual di API Anda. Batas pembatasan rute default tidak dapat melebihi batas tingkat akun.

Anda dapat mengonfigurasi pelambatan tingkat rute dengan menggunakanAWS CLI. Perintah berikut mengkonfigurasi pembatasan khusus untuk tahap dan rute API yang ditentukan.

```
aws apigatewayv2 update-stage \
    --api-id a1b2c3d4 \
    --stage-name dev \
    --route-settings '{"GET /pets":
{"ThrottlingBurstLimit":100,"ThrottlingRateLimit":2000}}'
```

Mengkonfigurasi otentikasi TLS timbal balik untuk API HTTP

Mutual TLS otentikasi membutuhkan otentikasi dua arah antara klien dan server. Dengan TLS bersama, klien harus menunjukkan sertifikat X.509 untuk memverifikasi identitas mereka untuk

mengakses API Anda. Mutual TLS adalah persyaratan umum untuk Internet of Things (IoT) business-to-business dan aplikasi.

Anda dapat menggunakan TLS bersama dengan operasi otorisasi dan otentikasi lain yang didukung API Gateway. API Gateway meneruskan sertifikat yang disediakan klien kepada otorisasi Lambda dan integrasi backend.

> **Important**
>
> Secara default, klien dapat memanggil API Anda dengan menggunakan `execute-api` titik akhir yang dihasilkan API Gateway untuk API Anda. Untuk memastikan bahwa klien dapat mengakses API Anda hanya dengan menggunakan nama domain khusus dengan TLS timbal balik, nonaktifkan titik `execute-api` akhir default. Untuk mempelajari selengkapnya, lihat the section called “Nonaktifkan endpoint default”.

Prasyarat untuk TLS timbal balik

Untuk mengkonfigurasi TLS timbal balik yang Anda butuhkan:

- Nama domain khusus
- Setidaknya satu sertifikat dikonfigurasi AWS Certificate Manager untuk nama domain kustom Anda
- Truststore dikonfigurasi dan diunggah ke Amazon S3

Nama domain khusus

Untuk mengaktifkan TLS bersama untuk HTTP API, Anda harus mengonfigurasi nama domain khusus untuk API Anda. Anda dapat mengaktifkan TLS timbal balik untuk nama domain khusus, dan kemudian memberikan nama domain khusus kepada klien. Untuk mengakses API dengan menggunakan nama domain kustom yang mengaktifkan TLS bersama, klien harus menunjukkan sertifikat yang Anda percayai dalam permintaan API. Anda dapat menemukan informasi lebih lanjut dithe section called “Nama domain kustom”.

Menggunakan sertifikat AWS Certificate Manager yang dikeluarkan

Anda dapat meminta sertifikat tepercaya publik langsung dari ACM atau mengimpor sertifikat publik atau yang ditandatangani sendiri. Untuk menyiapkan sertifikat di ACM, buka ACM. Jika Anda ingin mengimpor sertifikat, lanjutkan membaca di bagian berikut.

Menggunakan impor atau AWS Private Certificate Authority sertifikat

Untuk menggunakan sertifikat yang diimpor ke ACM atau sertifikat dari TLS AWS Private Certificate Authority bersama, API Gateway memerlukan yang `ownershipVerificationCertificate` dikeluarkan oleh ACM. Sertifikat kepemilikan ini hanya digunakan untuk memverifikasi bahwa Anda memiliki izin untuk menggunakan nama domain. Ini tidak digunakan untuk jabat tangan TLS. Jika Anda belum memilikiownershipVerificationCertificate, buka https://console.aws.amazon.com/acm/ untuk mengaturnya.

Anda harus menjaga sertifikat ini tetap berlaku selama masa pakai nama domain Anda. Jika sertifikat kedaluwarsa dan perpanjangan otomatis gagal, semua pembaruan pada nama domain akan dikunci. Anda perlu memperbarui `ownershipVerificationCertificateArn` dengan valid `ownershipVerificationCertificate` sebelum Anda dapat membuat perubahan lainnya. `ownershipVerificationCertificate`Tidak dapat digunakan sebagai sertifikat server untuk domain TLS timbal balik lainnya di API Gateway. Jika sertifikat langsung diimpor kembali ke ACM, penerbit harus tetap sama.

Mengkonfigurasi truststore Anda

Truststores adalah file teks dengan ekstensi `.pem` file. Mereka adalah daftar sertifikat tepercaya dari Otoritas Sertifikat. Untuk menggunakan TLS bersama, buat truststore sertifikat X.509 yang Anda percayai untuk mengakses API Anda.

Anda harus menyertakan rantai kepercayaan lengkap, mulai dari sertifikat CA penerbitan, hingga sertifikat CA root, di truststore Anda. API Gateway menerima sertifikat klien yang dikeluarkan oleh CA mana pun yang ada dalam rantai kepercayaan. Sertifikat dapat dari otoritas sertifikat publik atau swasta. Sertifikat dapat memiliki panjang rantai maksimum empat. Anda juga dapat memberikan sertifikat yang ditandatangani sendiri. Algoritma hashing berikut didukung di truststore:

- SHA-256 atau lebih kuat
- RSA-2048 atau lebih kuat
- ECDSA-256 atau lebih kuat

API Gateway memvalidasi sejumlah properti sertifikat. Anda dapat menggunakan otorisasi Lambda untuk melakukan pemeriksaan tambahan saat klien memanggil API, termasuk memeriksa apakah sertifikat telah dicabut. API Gateway memvalidasi properti berikut:

Validasi	Deskripsi
Sintaks X.509	Sertifikat harus memenuhi persyaratan sintaks X.509.
Integritas	Konten sertifikat tidak boleh diubah dari yang ditandatangani oleh otoritas sertifikat dari truststore.
Validitas	Masa berlaku sertifikat harus terkini.
Nama rantai/rantai kunci	Nama dan subjek sertifikat harus membentuk rantai yang tidak terputus. Sertifikat dapat memiliki panjang rantai maksimum empat.

Unggah truststore ke bucket Amazon S3 dalam satu file

Example sertifikat.pem

```
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
...
```

AWS CLIPerintah berikut diunggah `certificates.pem` ke bucket Amazon S3 Anda.

```
aws s3 cp certificates.pem s3://bucket-name
```

Mengkonfigurasi TLS timbal balik untuk nama domain khusus

Untuk mengonfigurasi TLS timbal balik untuk API HTTP, Anda harus menggunakan nama domain kustom Regional untuk API Anda, dengan versi TLS minimal 1.2. Untuk mempelajari lebih lanjut

tentang membuat dan mengonfigurasi nama domain kustom, lihatthe section called “Menyiapkan nama domain kustom regional”.

> **Note**
>
> Mutual TLS tidak didukung untuk API pribadi.

Setelah mengunggah truststore Anda ke Amazon S3, Anda dapat mengonfigurasi nama domain khusus Anda untuk menggunakan TLS bersama. Tempelkan yang berikut (termasuk garis miring) ke terminal:

```
aws apigatewayv2 create-domain-name \
    --domain-name api.example.com \
    --domain-name-configurations CertificateArn=arn:aws:acm:us-
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678 \
    --mutual-tls-authentication TruststoreUri=s3://bucket-name/key-name
```

Setelah membuat nama domain, Anda harus mengonfigurasi catatan DNS dan pemetaan basepath untuk operasi API. Untuk mempelajari selengkapnya, lihat Menyiapkan nama domain kustom regional di API Gateway.

Memanggil API dengan menggunakan nama domain khusus yang membutuhkan TLS timbal balik

Untuk menjalankan API dengan TLS timbal balik diaktifkan, klien harus menunjukkan sertifikat tepercaya dalam permintaan API. Saat klien mencoba menjalankan API Anda, API Gateway mencari penerbit sertifikat klien di truststore Anda. Agar API Gateway dapat melanjutkan permintaan, penerbit sertifikat dan rantai kepercayaan lengkap hingga sertifikat CA root harus ada di truststore Anda.

Contoh `curl` perintah berikut mengirimkan permintaan ke `api.example.com`, yang termasuk `my-cert.pem` dalam permintaan. `my-key.key`adalah kunci pribadi untuk sertifikat.

```
curl -v --key ./my-key.key --cert ./my-cert.pem api.example.com
```

API Anda dipanggil hanya jika truststore Anda mempercayai sertifikat. Kondisi berikut akan menyebabkan API Gateway gagal dalam jabat tangan TLS dan menolak permintaan dengan kode `403` status. Jika sertifikat Anda:

- tidak dipercaya
- kedaluwarsa
- tidak menggunakan algoritma yang didukung

> **Note**
>
> API Gateway tidak memverifikasi apakah sertifikat telah dicabut.

Memperbarui truststore Anda

Untuk memperbarui sertifikat di truststore Anda, unggah bundel sertifikat baru ke Amazon S3. Kemudian, Anda dapat memperbarui nama domain kustom Anda untuk menggunakan sertifikat yang diperbarui.

Gunakan versi Amazon S3 untuk mempertahankan beberapa versi truststore Anda. Saat memperbarui nama domain khusus untuk menggunakan versi truststore baru, API Gateway mengembalikan peringatan jika sertifikat tidak valid.

API Gateway menghasilkan peringatan sertifikat hanya ketika Anda memperbarui nama domain Anda. API Gateway tidak memberi tahu Anda jika sertifikat yang diunggah sebelumnya kedaluwarsa.

AWS CLIPerintah berikut memperbarui nama domain khusus untuk menggunakan versi truststore baru.

```
aws apigatewayv2 update-domain-name \
    --domain-name api.example.com \
    --domain-name-configurations CertificateArn=arn:aws:acm:us-
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678 \
    --mutual-tls-authentication TruststoreVersion='abcdef123'
```

Nonaktifkan TLS timbal balik

Untuk menonaktifkan TLS timbal balik untuk nama domain kustom, hapus truststore dari nama domain kustom Anda, seperti yang ditunjukkan pada perintah berikut.

```
aws apigatewayv2 update-domain-name \
    --domain-name api.example.com \
```

```
    --domain-name-configurations CertificateArn=arn:aws:acm:us-
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678 \
    --mutual-tls-authentication TruststoreUri=''
```

Memecahkan masalah peringatan sertifikat

Saat membuat nama domain khusus dengan TLS timbal balik, API Gateway mengembalikan peringatan jika sertifikat di truststore tidak valid. Ini juga dapat terjadi saat memperbarui nama domain khusus untuk menggunakan truststore baru. Peringatan menunjukkan masalah dengan sertifikat dan subjek sertifikat yang menghasilkan peringatan. Mutual TLS masih diaktifkan untuk API Anda, tetapi beberapa klien mungkin tidak dapat mengakses API Anda.

Anda harus memecahkan kode sertifikat di truststore Anda untuk mengidentifikasi sertifikat mana yang menghasilkan peringatan. Anda dapat menggunakan alat seperti `openssl` untuk memecahkan kode sertifikat dan mengidentifikasi subjek mereka.

Perintah berikut menampilkan isi sertifikat, termasuk subjeknya:

```
openssl x509 -in certificate.crt -text -noout
```

Perbarui atau hapus sertifikat yang menghasilkan peringatan, lalu unggah truststore baru ke Amazon S3. Setelah mengunggah truststore baru, perbarui nama domain kustom Anda untuk menggunakan truststore baru.

Memecahkan masalah konflik nama domain

Kesalahan `"The certificate subject <certSubject> conflicts with an existing certificate from a different issuer."` berarti beberapa Otoritas Sertifikat telah mengeluarkan sertifikat untuk domain ini. Untuk setiap subjek dalam sertifikat, hanya ada satu penerbit di API Gateway untuk domain TLS bersama. Anda harus mendapatkan semua sertifikat Anda untuk subjek itu melalui satu penerbit. Jika masalahnya adalah dengan sertifikat yang tidak Anda kendalikan tetapi Anda dapat membuktikan kepemilikan nama domain, hubungi AWS Support untuk membuka tiket.

Memecahkan masalah pesan status nama domain

`PENDING_CERTIFICATE_REIMPORT`: Ini berarti Anda mengimpor ulang sertifikat ke ACM dan validasi gagal karena sertifikat baru memiliki SAN (nama alternatif subjek) yang tidak tercakup oleh `ownershipVerificationCertificate` atau subjek atau SAN dalam sertifikat tidak mencakup nama domain. Sesuatu mungkin dikonfigurasi dengan tidak benar atau sertifikat yang tidak valid

diimpor. Anda perlu mengimpor ulang sertifikat yang valid ke ACM. Untuk informasi selengkapnya tentang validasi, lihat Memvalidasi kepemilikan domain.

`PENDING_OWNERSHIP_VERIFICATION`: Ini berarti sertifikat Anda yang telah diverifikasi sebelumnya telah kedaluwarsa dan ACM tidak dapat memperbaruinya secara otomatis. Anda perlu memperbarui sertifikat atau meminta sertifikat baru. Informasi lebih lanjut tentang perpanjangan sertifikat dapat ditemukan di panduan perpanjangan sertifikat terkelola pemecahan masalah ACM.

Pemantauan HTTP API

Anda dapat menggunakan metrik CloudWatch dan CloudWatch Logs untuk memantau API HTTP. Dengan menggabungkan log dan metrik, Anda dapat mencatat kesalahan dan memantau kinerja API Anda.

> **Note**
>
> API Gateway mungkin tidak menghasilkan log dan metrik dalam kasus berikut:
>
> - 413 Permintaan Entitas Kesalahan Terlalu Besar
> - Berlebihan 429 Terlalu Banyak Permintaan kesalahan
> - Kesalahan seri 400 dari permintaan yang dikirim ke domain khusus yang tidak memiliki pemetaan API
> - 500 kesalahan seri yang disebabkan oleh kegagalan internal

Topik

- Bekerja dengan metrik untuk HTTP API
- Mengkonfigurasi logging untuk HTTP API

Bekerja dengan metrik untuk HTTP API

Anda dapat memantau eksekusi API dengan menggunakan CloudWatch, yang mengumpulkan dan memproses data mentah dari API Gateway menjadi metrik yang dapat dibaca. near-real-time Statistik ini dicatat untuk jangka waktu 15 bulan sehingga Anda dapat mengakses informasi historis dan mendapatkan perspektif yang lebih baik tentang kinerja aplikasi atau layanan web Anda. Secara default, data metrik API Gateway dikirim secara otomatis CloudWatch dalam periode satu menit. Untuk memantau metrik Anda, buat CloudWatch dasbor untuk API Anda. Untuk informasi

selengkapnya tentang cara membuat CloudWatch dasbor, lihat Membuat CloudWatch dasbor di Panduan CloudWatch Pengguna Amazon. Untuk informasi selengkapnya, lihat Apa itu Amazon CloudWatch? di Panduan CloudWatch Pengguna Amazon.

Metrik berikut didukung untuk API HTTP. Anda juga dapat mengaktifkan metrik terperinci untuk menulis metrik tingkat rute ke Amazon. CloudWatch

Metrik	Deskripsi
4xx	Jumlah kesalahan sisi klien yang ditangkap dalam periode tertentu.
5xx	Jumlah kesalahan sisi server yang ditangkap dalam periode tertentu.
Count	Jumlah total permintaan API dalam periode tertentu.
IntegrationLatency	Waktu antara saat API Gateway menyampaikan permintaan ke backend dan saat menerima respons dari backend.
Latensi	Waktu antara saat API Gateway menerima permintaan dari klien dan saat mengembalikan respons ke klien. Latensi mencakup latensi integrasi dan overhead API Gateway lainnya.
DataProcessed	Jumlah data yang diproses dalam byte.

Anda dapat menggunakan dimensi dalam tabel berikut untuk memfilter metrik API Gateway.

Dimensi	Deskripsi
ApiId	Memfilter metrik API Gateway untuk API dengan ID API yang ditentukan.
ApiId, Panggung	Memfilter metrik API Gateway untuk tahap API dengan ID API dan ID tahap yang ditentukan.
ApiId, Panggung, Rute	Memfilter metrik API Gateway untuk metode API dengan ID API, ID tahap, dan ID rute yang ditentukan. API Gateway tidak akan mengirimkan metrik ini kecuali Anda telah mengaktifkan metrik terperinci secara eksplisit. CloudWatch Anda dapat melakukan ini dengan memanggil UpdateStageaksi API Gateway V2 REST API untuk memperbarui `detailedMetricsEnabled` properti ke`true`. Atau, Anda dapat memanggil AWS CLI perintah update-stage untuk memperbarui properti ke. `DetailedMetricsEnabled` `true` Mengaktifkan metrik tersebut akan dikenakan biaya tambahan ke akun Anda. Untuk informasi harga, lihat CloudWatchHarga Amazon.

Mengkonfigurasi logging untuk HTTP API

Anda dapat mengaktifkan logging untuk menulis log ke CloudWatch Log. Anda dapat menggunakan variabel logging untuk menyesuaikan konten log Anda.

Untuk mengaktifkan logging untuk API HTTP, Anda harus melakukan hal berikut.

1. Pastikan bahwa pengguna Anda memiliki izin yang diperlukan untuk mengaktifkan logging.
2. Buat grup CloudWatch log Log.
3. Berikan ARN dari grup CloudWatch log Log untuk tahap API Anda.

Izin untuk mengaktifkan logging

Untuk mengaktifkan logging untuk API, pengguna Anda harus memiliki izin berikut.

Example

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:GetLogEvents",
                "logs:FilterLogEvents"
            ],
            "Resource": "arn:aws:logs:us-east-2:123456789012:log-group:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:PutResourcePolicy",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:CreateLogGroup",
                "logs:DescribeResourcePolicies",
                "logs:GetLogDelivery",
                "logs:ListLogDeliveries"
```

```
            ],
            "Resource": "*"
        }
    ]
}
```

Buat grup log dan aktifkan logging untuk HTTP API

Anda dapat membuat grup log dan mengaktifkan log akses menggunakan AWS Management Console atauAWS CLI.

AWS Management Console

1. Buat grup log.

   Untuk mempelajari cara membuat grup log menggunakan konsol, lihat Membuat Grup Log di Panduan Pengguna Amazon CloudWatch Logs.
2. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
3. Pilih API HTTP.
4. Di bawah tab Monitor di panel navigasi utama, pilih Logging.
5. Pilih tahap untuk mengaktifkan logging dan pilih Pilih.
6. Pilih Edit untuk mengaktifkan pencatatan akses.
7. Aktifkan Pencatatan akses, masukkan CloudWatch Log, dan pilih format log.
8. Pilih Save (Simpan).

AWS CLI

AWS CLIPerintah berikut membuat grup log.

```
aws logs create-log-group --log-group-name my-log-group
```

Anda memerlukan Amazon Resource Name (ARN) untuk grup log Anda untuk mengaktifkan logging. *Format ARN adalah arn:aws:logs: region: account-id:log-group:. log-group-name*

AWS CLIPerintah berikut mengaktifkan logging untuk `$default` tahap HTTP API.

```
aws apigatewayv2 update-stage --api-id abcdef \
    --stage-name '$default' \
    --access-log-settings '{"DestinationArn": "arn:aws:logs:region:account-
id:log-group:log-group-name", "Format": "$context.identity.sourceIp - -
 [$context.requestTime] \"$context.httpMethod $context.routeKey $context.protocol\"
 $context.status $context.responseLength $context.requestId"}'
```

Contoh format log

Contoh beberapa format log akses umum tersedia di konsol API Gateway dan dicantumkan sebagai berikut.

- CLF(Format Log Umum):

```
$context.identity.sourceIp - - [$context.requestTime] "$context.httpMethod
 $context.routeKey $context.protocol" $context.status $context.responseLength
 $context.requestId $context.extendedRequestId
```

- JSON:

```
{ "requestId":"$context.requestId", "ip": "$context.identity.sourceIp",
 "requestTime":"$context.requestTime",
 "httpMethod":"$context.httpMethod","routeKey":"$context.routeKey",
 "status":"$context.status","protocol":"$context.protocol",
 "responseLength":"$context.responseLength", "extendedRequestId":
 "$context.extendedRequestId" }
```

- XML:

```
<request id="$context.requestId"> <ip>$context.identity.sourceIp</ip> <requestTime>
$context.requestTime</requestTime> <httpMethod>$context.httpMethod</httpMethod>
 <routeKey>$context.routeKey</routeKey> <status>$context.status</status> <protocol>
$context.protocol</protocol> <responseLength>$context.responseLength</responseLength>
 <extendedRequestId>$context.extendedRequestId</extendedRequestId> </request>
```

- CSV(nilai yang dipisahkan koma):

```
$context.identity.sourceIp,$context.requestTime,$context.httpMethod,
$context.routeKey,$context.protocol,$context.status,$context.responseLength,
$context.requestId,$context.extendedRequestId
```

Menyesuaikan log akses HTTP API

Anda dapat menggunakan variabel berikut untuk menyesuaikan log akses HTTP API. Untuk mempelajari lebih lanjut tentang log akses untuk API HTTP, lihatMengkonfigurasi logging untuk HTTP API.

Parameter	Deskripsi
`$context.accountId`	ID AWS akun pemilik API.
`$context.apiId`	API Gateway identifier ditetapkan ke API Anda.
`$context.authorizer.claims.` *`property`*	Properti klaim yang dikembalikan dari JSON Web Token (JWT) setelah pemanggil metode berhasil diautentikasi, seperti. `$context.authorizer.claims.username` Untuk informasi selengkapnya, lihat Mengontrol akses ke API HTTP dengan otorisasi JWT. **Note** Memanggil `$context.authorizer.claims` mengembalikan null.
`$context.authorizer.error`	Pesan kesalahan dikembalikan dari otorisasi.
`$context.authorizer.principalId`	Identifikasi pengguna utama yang dikembalikan oleh otorisasi Lambda.
`$context.authorizer.` *`property`*	Nilai pasangan nilai kunci yang ditentukan dari `context` peta dikembalikan dari fungsi otorisasi API Gateway Lambda. Misalnya, jika otorisasi mengembalikan `context` peta berikut: `"context" : { "key": "value", "numKey": 1, "boolKey": true`

Parameter	Deskripsi
	`}` memanggil `$context.authorizer.key` mengembalikan `"value"` string, memanggil `$context.authorizer.numKey` mengembalikan`1`, dan memanggil `$context.authorizer.boolKey` kembali`true`.
`$context.awsEndpointRequestId`	ID permintaan AWS titik akhir dari `x-amzn-requestId` header `x-amz-request-id` atau.
`$context.awsEndpointRequestId2`	ID permintaan AWS titik akhir dari `x-amz-id-2` header.
`$context.customDomain.basePathMatched`	Jalur untuk pemetaan API yang cocok dengan permintaan masuk. Berlaku ketika klien menggunakan nama domain khusus untuk mengakses API. Misalnya jika klien mengirim permintaan ke`https://api.example.com/v1/orders/1234` , dan permintaan tersebut cocok dengan pemetaan API dengan jalur`v1/orders` , nilainya adalah`v1/orders` . Untuk mempelajari selengkapnya, lihat the section called “Pemetaan API”.
`$context.dataProcessed`	Jumlah data yang diproses dalam byte.
`$context.domainName`	Nama domain lengkap yang digunakan untuk memanggil API. Ini harus sama dengan `Host` header yang masuk.
`$context.domainPrefix`	Label pertama dari`$context.domainName` .
`$context.error.message`	String yang berisi pesan kesalahan API Gateway.

Parameter	Deskripsi
`$context.error.messageString`	Nilai yang dikutip dari`$context.error.message` , yaitu`"$context.error.message"` .
`$context.error.responseType`	Sebuah jenis`GatewayResponse` . Lihat informasi yang lebih lengkap di the section called “Metrik” dan the section called “Menyiapkan respons gateway untuk menyesuaikan respons kesalahan”.
`$context.extendedRequestId`	Setara dengan`$context.requestId` .
`$context.httpMethod`	Metode HTTP yang digunakan . Nilai yang valid meliputi: `DELETE`,`GET`,`HEAD`,`OPTIONS`,`PATCH`,`POST`, dan`PUT`.
`$context.identity.accountId`	ID AWS akun yang terkait dengan permintaan. Didukung untuk rute yang menggunakan otorisasi IAM.
`$context.identity.caller`	Pengenal utama penelepon yang menandatangani permintaan. Didukung untuk rute yang menggunakan otorisasi IAM.

Parameter	Deskripsi
`$context.identity.cognitoAuthenticationProvider`	Daftar penyedia otentikasi Amazon Cognito yang dipisahkan koma yang digunakan oleh penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredenal Amazon Cognito. Misalnya, untuk identitas dari kumpulan pengguna Amazon Cognito, `cognito-idp.` *`region`*`.amazonaws.com/` *`user_pool_id`* `,cognito-idp.` *`region`*`.amazonaws.com/` *`user_pool_id`* `:CognitoSignIn:` *`token subject claim`* Untuk selengkapnya, lihat Menggunakan Identitas Federasi di Panduan Pengembang Amazon Cognito.
`$context.identity.cognitoAuthenticationType`	Jenis otentikasi Amazon Cognito dari penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredenal Amazon Cognito. Nilai yang mungkin termasuk `authenticated` untuk identitas yang diautentikasi dan `unauthenticated` untuk identitas yang tidak diautentikasi.
`$context.identity.cognitoIdentityId`	ID identitas Amazon Cognito dari penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredenal Amazon Cognito.
`$context.identity.cognitoIdentityPoolId`	ID kumpulan identitas Amazon Cognito dari penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredenal Amazon Cognito.

Parameter	Deskripsi
`$context.identity.principalOrgId`	ID AWS organisasi. Didukung untuk rute yang menggunakan otorisasi IAM.
`$context.identity.clientCert.clientCertPem`	Sertifikat klien yang dikodekan PEM yang disajikan klien selama otentikasi TLS timbal balik. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik.
`$context.identity.clientCert.subjectDN`	Nama yang dibedakan dari subjek sertifikat yang disajikan klien. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik.
`$context.identity.clientCert.issuerDN`	Nama terhormat dari penerbit sertifikat yang disajikan klien. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik.
`$context.identity.clientCert.serialNumber`	Nomor seri sertifikat. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik.
`$context.identity.clientCert.validity.notBefore`	Tanggal sebelum sertifikat tidak valid. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik.
`$context.identity.clientCert.validity.notAfter`	Tanggal setelah sertifikat tidak valid. Hadir saat klien mengakses API dengan menggunakan nama domain khusus yang mengaktifkan TLS timbal balik.

Parameter	Deskripsi
`$context.identity.sourceIp`	Alamat IP sumber dari koneksi TCP langsung membuat permintaan ke titik akhir API Gateway.
`$context.identity.user`	Pengidentifikasi utama pengguna yang akan diotorisasi terhadap akses sumber daya. Didukung untuk rute yang menggunakan otorisasi IAM.
`$context.identity.userAgent`	`User-Agent` Header pemanggil API.
`$context.identity.userArn`	Nama Sumber Daya Amazon (ARN) dari pengguna efektif yang diidentifikasi setelah otentikasi. Didukung untuk rute yang menggunakan otorisasi IAM. Untuk informasi selengkapnya, lihat https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html.
`$context.integration.error`	Pesan kesalahan dikembalikan dari integrasi . Setara dengan`$context.integrationErrorMessage` .
`$context.integration.integrationStatus`	Untuk integrasi proxy Lambda, kode status dikembalikan dari AWS Lambda, bukan dari kode fungsi Lambda backend.
`$context.integration.latency`	Latensi integrasi dalam ms. Setara dengan`$context.integrationLatency` .
`$context.integration.requestId`	ID permintaan AWS titik akhir. Setara dengan`$context.awsEndpointRequestId` .

Parameter	Deskripsi
`$context.integration.status`	Kode status dikembalikan dari integrasi. Untuk integrasi proxy Lambda, ini adalah kode status yang dikembalikan oleh kode fungsi Lambda Anda.
`$context.integrationErrorMessage`	String yang berisi pesan kesalahan integrasi.
`$context.integrationLatency`	Latensi integrasi dalam ms.
`$context.integrationStatus`	Untuk integrasi proxy Lambda, parameter ini mewakili kode status yang dikembalikan dari AWS Lambda, bukan dari fungsi Lambda backend.
`$context.path`	Jalur permintaan. Misalnya, `/{stage}/root/child` .
`$context.protocol`	Protokol permintaan, misalnya,`HTTP/1.1`. **Note** API Gateway API dapat menerima permintaan HTTP/2, tetapi API Gateway mengirimkan permintaa n ke integrasi backend menggunak an HTTP/1.1. Akibatnya, protokol permintaan dicatat sebagai HTTP/1.1 bahkan jika klien mengirim permintaan yang menggunakan HTTP/2.
`$context.requestId`	ID yang ditetapkan API Gateway ke permintaan API.
`$context.requestTime`	Waktu permintaan yang diformat CLF (). `dd/MMM/yyyy:HH:mm:ss +-hhmm`

Parameter	Deskripsi
`$context.requestTimeEpoch`	Waktu permintaan yang diformat Epoch.
`$context.responseLatency`	Latensi respons dalam ms.
`$context.responseLength`	Panjang payload respon dalam byte.
`$context.routeKey`	Kunci rute permintaan API, misalnya`/pets`.
`$context.stage`	Tahap penerapan permintaan API (misalnya, `beta` atau`prod`).
`$context.status`	Status respons metode.

Pemecahan masalah dengan HTTP

Topik berikut memberikan saran pemecahan masalah untuk kesalahan dan masalah yang mungkin Anda temui saat menggunakan API HTTP.

Topik

- Memecahkan masalah dengan integrasi Lambda API HTTP
- Memecahkan masalah dengan otorisasi HTTP API JWT

Memecahkan masalah dengan integrasi Lambda API HTTP

Berikut ini memberikan saran pemecahan masalah untuk kesalahan dan masalah yang mungkin Anda temui saat menggunakan AWS Lambdaintegrasi dengan API HTTP.

Masalah: API saya dengan integrasi Lambda kembali **`{"message":"Internal Server Error"}`**

Untuk memecahkan masalah kesalahan server internal, tambahkan variabel `$context.integrationErrorMessage` logging ke format log Anda, dan lihat log HTTP API Anda. Untuk mencapai ini, lakukan hal berikut:

Untuk membuat grup log dengan menggunakan AWS Management Console

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pilih Grup log.
3. Pilih Buat grup log.
4. Masukkan nama grup log, lalu pilih Buat.
5. Perhatikan Nama Sumber Daya Amazon (ARN) untuk grup log Anda. *Format ARN adalah arn:aws:logs: region: account-id:log-group:. log-group-name* Anda memerlukan grup log ARN untuk mengaktifkan pencatatan akses untuk HTTP API Anda.

Untuk menambahkan variabel **`$context.integrationErrorMessage`** logging

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih HTTP API Anda.
3. Di bawah Monitor, pilih Logging.
4. Pilih tahap API Anda.
5. Pilih Edit, lalu aktifkan pencatatan akses.
6. Untuk tujuan Log, masukkan ARN dari grup log yang Anda buat pada langkah sebelumnya.
7. Untuk format Log, pilih CLF. API Gateway membuat contoh format log.
8. Tambahkan `$context.integrationErrorMessage` ke akhir format log.
9. Pilih Simpan.

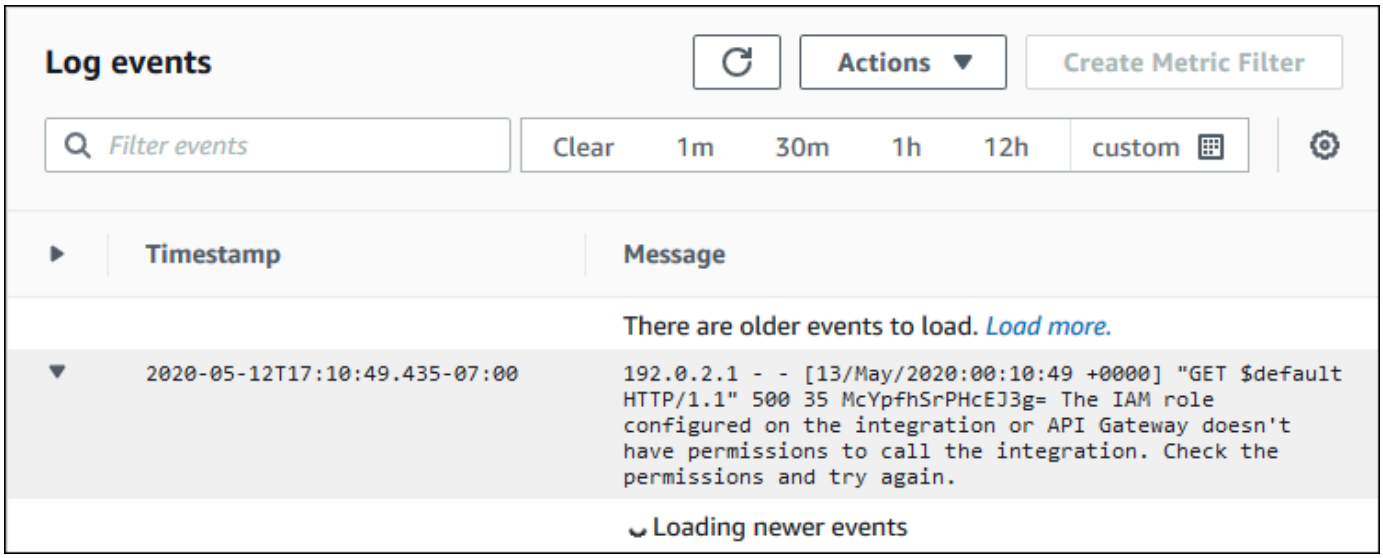
Untuk melihat log API Anda

1. Hasilkan log. Gunakan browser atau `curl` untuk menjalankan API Anda.

```
$curl https://api-id.execute-api.us-west-2.amazonaws.com/route
```

2. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
3. Pilih HTTP API Anda.
4. Di bawah Monitor, pilih Logging.
5. Pilih tahap API yang Anda aktifkan logging.
6. Pilih Lihat log masuk CloudWatch.
7. Pilih aliran log terbaru untuk melihat log HTTP API Anda.

8. Entri log Anda akan terlihat mirip dengan yang berikut ini:



Karena kami menambahkan `$context.integrationErrorMessage` ke format log, kami melihat pesan kesalahan di log kami yang merangkum masalah.

Log Anda mungkin menyertakan pesan kesalahan berbeda yang menunjukkan bahwa ada masalah dengan kode fungsi Lambda Anda. Dalam hal ini, periksa kode fungsi Lambda Anda, dan verifikasi bahwa fungsi Lambda Anda mengembalikan respons dalam format yang diperlukan. Jika log Anda tidak menyertakan pesan kesalahan, tambahkan `$context.error.message` dan `$context.error.responseType` ke format log Anda untuk informasi selengkapnya guna membantu memecahkan masalah.

Dalam kasus ini, log menunjukkan bahwa API Gateway tidak memiliki izin yang diperlukan untuk menjalankan fungsi Lambda.

Saat Anda membuat integrasi Lambda di konsol API Gateway, API Gateway secara otomatis mengonfigurasi izin untuk menjalankan fungsi Lambda. Saat membuat integrasi Lambda dengan menggunakan AWS CLI,, atau SDK AWS CloudFormation, Anda harus memberikan izin kepada API Gateway untuk menjalankan fungsi tersebut. Contoh AWS CLI perintah berikut memberikan izin untuk rute API HTTP yang berbeda untuk menjalankan fungsi Lambda.

Example Contoh - Untuk **`$default`** tahap dan **`$default`** rute API HTTP

```
aws lambda add-permission \
    --function-name my-function \
    --statement-id apigateway-invoke-permissions \
    --action lambda:InvokeFunction \
```

```
    --principal apigateway.amazonaws.com \
    --source-arn "arn:aws:execute-api:us-west-2:123456789012:api-id/\$default/\
$default"
```

Example Contoh - Untuk **`prod`** tahap dan **`test`** rute API HTTP

```
aws lambda add-permission \
    --function-name my-function \
    --statement-id apigateway-invoke-permissions \
    --action lambda:InvokeFunction \
    --principal apigateway.amazonaws.com \
    --source-arn "arn:aws:execute-api:us-west-2:123456789012:api-id/prod/*/test"
```

Konfirmasi kebijakan fungsi di tab Izin di konsol Lambda.

Coba jalankan API Anda lagi. Anda akan melihat respons fungsi Lambda Anda.

Memecahkan masalah dengan otorisasi HTTP API JWT

Berikut ini memberikan saran pemecahan masalah untuk kesalahan dan masalah yang mungkin Anda temui saat menggunakan otorisasi JSON Web Token (JWT) dengan API HTTP.

Masalah: API saya kembali **`401 {"message":"Unauthorized"}`**

Periksa `www-authenticate` header dalam respons dari API.

Perintah berikut digunakan `curl` untuk mengirim permintaan ke API dengan otorisasi JWT yang digunakan `$request.header.Authorization` sebagai sumber identitasnya.

```
$curl -v -H "Authorization: token" https://api-id.execute-api.us-
west-2.amazonaws.com/route
```

Respons dari API termasuk `www-authenticate` header.

```
...
< HTTP/1.1 401 Unauthorized
< Date: Wed, 13 May 2020 04:07:30 GMT
< Content-Length: 26
< Connection: keep-alive
< www-authenticate: Bearer scope="" error="invalid_token" error_description="the token
 does not have a valid audience"
```

```
< apigw-requestid: Mc7UVioPPHcEKPA=
<
* Connection #0 to host api-id.execute-api.us-west-2.amazonaws.com left intact
{"message":"Unauthorized"}}
```

Dalam hal ini, `www-authenticate` header menunjukkan bahwa token tidak dikeluarkan untuk audiens yang valid. Agar API Gateway dapat mengotorisasi permintaan, JWT `aud` atau `client_id` klaim harus cocok dengan salah satu entri audiens yang dikonfigurasi untuk otorisasi. API Gateway `client_id` hanya memvalidasi jika tidak `aud` ada. Saat keduanya `aud` dan `client_id` ada, API Gateway mengevaluasi. `aud`

Anda juga dapat memecahkan kode JWT dan memverifikasi bahwa JWT cocok dengan penerbit, audiens, dan cakupan yang dibutuhkan API Anda. Situs web jwt.io dapat men-debug JWT di browser. OpenID Foundation menyimpan daftar pustaka untuk bekerja dengan JWT.

Untuk mempelajari selengkapnya tentang otorisasi JWT, lihat. Mengontrol akses ke API HTTP dengan otorisasi JWT

Bekerja dengan WebSocket API

WebSocket API di API Gateway adalah kumpulan WebSocket rute yang terintegrasi dengan titik akhir HTTP backend, fungsi Lambda, atau layanan lainnya. AWS Anda dapat menggunakan fitur API Gateway untuk membantu Anda dengan semua aspek siklus hidup API, mulai dari pembuatan hingga pemantauan API produksi Anda.

API Gateway WebSocket API bersifat dua arah. Klien dapat mengirim pesan ke layanan, dan layanan dapat secara mandiri mengirim pesan ke klien. Perilaku dua arah ini memungkinkan interaksi klien/layanan yang lebih kaya karena layanan dapat mendorong data ke klien tanpa mengharuskan klien untuk membuat permintaan eksplisit. WebSocket API sering digunakan dalam aplikasi real-time seperti aplikasi obrolan, platform kolaborasi, game multipemain, dan platform perdagangan keuangan.

Untuk contoh aplikasi untuk memulai, lihatTutorial: Membangun aplikasi obrolan tanpa server dengan WebSocket API, Lambda, dan DynamoDB.

Di bagian ini, Anda dapat mempelajari cara mengembangkan, menerbitkan, melindungi, dan memantau WebSocket API Anda menggunakan API Gateway.

Topik

Tentang WebSocket API di API Gateway

Di API Gateway Anda dapat membuat WebSocket API sebagai frontend stateful untuk AWS layanan (seperti Lambda atau DynamoDB) atau untuk titik akhir HTTP. WebSocket API memanggil backend Anda berdasarkan konten pesan yang diterimanya dari aplikasi klien.

Tidak seperti REST API, yang menerima dan merespons permintaan, WebSocket API mendukung komunikasi dua arah antara aplikasi klien dan backend Anda. Backend dapat mengirim pesan callback ke klien yang terhubung.

Di WebSocket API Anda, pesan JSON yang masuk diarahkan ke integrasi backend berdasarkan rute yang Anda konfigurasikan. (Pesan non-JSON diarahkan ke `$default` rute yang Anda konfigurasikan.)

Rute menyertakan kunci rute, yang merupakan nilai yang diharapkan setelah ekspresi pemilihan rute dievaluasi. `routeSelectionExpression`Ini adalah atribut yang didefinisikan pada tingkat API. Ini menentukan properti JSON yang diharapkan hadir dalam payload pesan. Untuk informasi selengkapnya tentang ekspresi pemilihan rute, lihatthe section called “”.

Misalnya, jika pesan JSON Anda berisi `action` properti, dan Anda ingin melakukan tindakan berbeda berdasarkan properti ini, ekspresi pemilihan rute Anda mungkin`${request.body.action}`. Tabel routing Anda akan menentukan tindakan mana yang akan dilakukan dengan mencocokkan nilai `action` properti terhadap nilai kunci rute kustom yang telah Anda tentukan dalam tabel.

Ada tiga rute standar yang dapat digunakan:`$connect`,`$disconnect`, dan`$default`. Selain itu, Anda dapat membuat rute khusus.

- API Gateway memanggil `$connect` rute saat koneksi persisten antara klien dan WebSocket API sedang dimulai.
- API Gateway memanggil `$disconnect` rute saat klien atau server terputus dari API.
- API Gateway memanggil rute kustom setelah ekspresi pemilihan rute dievaluasi terhadap pesan jika rute yang cocok ditemukan; kecocokan menentukan integrasi mana yang dipanggil.
- API Gateway memanggil `$default` rute jika ekspresi pemilihan rute tidak dapat dievaluasi terhadap pesan atau jika tidak ditemukan rute yang cocok.

Untuk informasi selengkapnya tentang `$disconnect` rute `$connect` dan rute, lihatthe section called “Mengelola pengguna dan aplikasi klien yang terhubung”.

Untuk informasi selengkapnya tentang `$default` rute dan rute khusus, lihatthe section called “Memohon integrasi backend Anda”.

Layanan backend dapat mengirim data ke aplikasi klien yang terhubung. Untuk informasi selengkapnya, lihat the section called “Mengirim data dari layanan backend ke klien yang terhubung”.

Mengelola pengguna dan aplikasi klien yang terhubung: **$connect** dan **$disconnect** rute

Topik

$connect Rute

Aplikasi klien terhubung ke WebSocket API Anda dengan mengirimkan permintaan WebSocket pemutakhiran. Jika permintaan berhasil, `$connect` rute dijalankan saat koneksi sedang dibuat.

Karena WebSocket koneksi adalah koneksi stateful, Anda dapat mengonfigurasi otorisasi pada rute saja. `$connect` `AuthN`/`AuthZ`akan dilakukan hanya pada waktu koneksi.

Sampai eksekusi integrasi yang terkait dengan `$connect` rute selesai, permintaan upgrade tertunda dan koneksi aktual tidak akan dibuat. Jika `$connect` permintaan gagal (misalnya, karena `AuthN` `AuthZ` /kegagalan atau kegagalan integrasi), koneksi tidak akan dibuat.

> **Note**
>
> Jika otorisasi gagal`$connect`, koneksi tidak akan dibuat, dan klien akan menerima `401` atau `403` tanggapan.

Menyiapkan integrasi untuk `$connect` adalah opsional. Anda harus mempertimbangkan untuk menyiapkan `$connect` integrasi jika:

- Anda ingin mengaktifkan klien untuk menentukan subprotokol dengan menggunakan bidang. `Sec-WebSocket-Protocol` Untuk kode sampel, lihat Menyiapkan `$connect` rute yang membutuhkan WebSocket subprotokol.
- Anda ingin diberi tahu saat klien terhubung.
- Anda ingin membatasi koneksi atau mengontrol siapa yang terhubung.
- Anda ingin backend Anda mengirim pesan kembali ke klien menggunakan URL callback.

- Anda ingin menyimpan setiap ID koneksi dan informasi lainnya ke dalam database (misalnya, Amazon DynamoDB).

Melewati informasi koneksi dari **`$connect`** rute

Anda dapat menggunakan integrasi proxy dan non-proxy untuk meneruskan informasi dari `$connect` rute ke database atau lainnya. Layanan AWS

Untuk meneruskan informasi koneksi menggunakan integrasi proxy

Anda dapat mengakses informasi koneksi dari integrasi proxy Lambda dalam acara tersebut. Gunakan AWS Lambda fungsi lain Layanan AWS atau untuk memposting ke koneksi.

Fungsi Lambda berikut menunjukkan cara menggunakan `requestContext` objek untuk mencatat ID koneksi, nama domain, nama panggung, dan string kueri.

Node.js

```
 export const handler = async(event, context) => {
    const connectId = event["requestContext"]["connectionId"]
    const domainName = event["requestContext"]["domainName"]
    const stageName = event["requestContext"]["stage"]
    const qs = event['queryStringParameters']
    console.log('Connection ID: ', connectId, 'Domain Name: ', domainName, 'Stage
 Name: ', stageName, 'Query Strings: ', qs )
    return {"statusCode" : 200}
};
```

Python

```
import json
import logging
logger = logging.getLogger()
logger.setLevel("INFO")


def lambda_handler(event, context):
    connectId = event["requestContext"]["connectionId"]
    domainName = event["requestContext"]["domainName"]
    stageName = event["requestContext"]["stage"]
    qs = event['queryStringParameters']
    connectionInfo = {
```

```
        'Connection ID': connectId,
        'Domain Name': domainName,
        'Stage Name': stageName,
        'Query Strings': qs}
    logging.info(connectionInfo)
    return {"statusCode": 200}
```

Untuk meneruskan informasi koneksi menggunakan integrasi non-proxy

- Anda dapat mengakses informasi koneksi dengan integrasi non-proxy. Siapkan permintaan integrasi dan berikan template permintaan WebSocket API. Template pemetaan Velocity Template Language (VTL) berikut menyediakan permintaan integrasi. Permintaan ini mengirimkan rincian berikut ke integrasi non-proxy:

  - ID Koneksi
  - Nama domain
  - Nama panggung
  - Jalur
  - Header
  - String pertanyaan

  Permintaan ini mengirimkan ID koneksi, nama domain, nama panggung, jalur, header, dan string kueri ke integrasi non-proxy.

  ```
  {
      "connectionId": "$context.connectionId",
      "domain": "$context.domainName",
      "stage": "$context.stage",
      "params": "$input.params()"
  }
  ```

  Untuk informasi selengkapnya tentang pengaturan transformasi data, lihatthe section called “Transformasi data”.

  Untuk menyelesaikan permintaan integrasi, `StatusCode: 200` tetapkan respons integrasi. Untuk mempelajari lebih lanjut tentang menyiapkan respons integrasi, lihatMenyiapkan respons integrasi menggunakan konsol API Gateway.

`$disconnect`Rute

`$disconnect`Rute dijalankan setelah koneksi ditutup.

Koneksi dapat ditutup oleh server atau oleh klien. Karena koneksi sudah ditutup saat dijalankan, `$disconnect` adalah acara upaya terbaik. API Gateway akan mencoba yang terbaik untuk mengirimkan `$disconnect` acara ke integrasi Anda, tetapi tidak dapat menjamin pengiriman.

Backend dapat memulai pemutusan dengan menggunakan API. `@connections` Untuk informasi selengkapnya, lihat the section called “Gunakan `@connections` perintah di layanan backend Anda”.

Memanggil integrasi backend Anda: `$default` Rute dan rute khusus

Topik

Menggunakan rute untuk memproses pesan

Di API Gateway WebSocket API, pesan dapat dikirim dari klien ke layanan backend Anda dan sebaliknya. Tidak seperti model permintaan/respons HTTP, di WebSocket backend dapat mengirim pesan ke klien tanpa klien mengambil tindakan apa pun.

Pesan dapat berupa JSON atau non-JSON. Namun, hanya pesan JSON yang dapat dialihkan ke integrasi tertentu berdasarkan konten pesan. Pesan non-JSON diteruskan ke backend melalui rute. `$default`

> **Note**
>
> API Gateway mendukung muatan pesan hingga 128 KB dengan ukuran bingkai maksimum 32 KB. Jika pesan melebihi 32 KB, Anda harus membaginya menjadi beberapa frame, masing-masing 32 KB atau lebih kecil. Jika pesan yang lebih besar (atau bingkai) diterima, koneksi ditutup dengan kode 1009.

Saat ini payload biner tidak didukung. Jika bingkai biner diterima, koneksi ditutup dengan kode 1003. Namun, dimungkinkan untuk mengubah muatan biner menjadi teks. Lihat the section called “Tipe media biner”.

Dengan WebSocket API di API Gateway, pesan JSON dapat dirutekan untuk menjalankan layanan backend tertentu berdasarkan konten pesan. Ketika klien mengirim pesan melalui WebSocket koneksinya, ini menghasilkan permintaan rute ke WebSocket API. Permintaan akan dicocokkan dengan rute dengan kunci rute yang sesuai di API Gateway. Anda dapat menyiapkan permintaan rute untuk WebSocket API di konsol API Gateway, dengan menggunakan AWS CLI, atau dengan menggunakan AWS SDK.

Note

Di AWS CLI dan AWS SDK, Anda dapat membuat rute sebelum atau setelah Anda membuat integrasi. Saat ini konsol tidak mendukung penggunaan kembali integrasi, jadi Anda harus membuat rute terlebih dahulu dan kemudian membuat integrasi untuk rute itu.

Anda dapat mengonfigurasi API Gateway untuk melakukan validasi pada permintaan rute sebelum melanjutkan dengan permintaan integrasi. Jika validasi gagal, API Gateway gagal permintaan tanpa memanggil backend Anda, mengirimkan respons `"Bad request body"` gateway yang serupa dengan yang berikut ke klien, dan menerbitkan hasil validasi di Log: CloudWatch

```
{"message" : "Bad request body", "connectionId": "{connectionId}", "messageId":
 "{messageId}"}
```

Ini mengurangi panggilan yang tidak perlu ke backend Anda dan memungkinkan Anda fokus pada persyaratan lain dari API Anda.

Anda juga dapat menentukan respons rute untuk rute API Anda untuk mengaktifkan komunikasi dua arah. Respons rute menjelaskan data apa yang akan dikirim ke klien Anda setelah menyelesaikan integrasi rute tertentu. Tidak perlu menentukan respons untuk rute jika, misalnya, Anda ingin klien mengirim pesan ke backend Anda tanpa menerima respons (komunikasi satu arah). Namun, jika Anda tidak memberikan respons rute, API Gateway tidak akan mengirimkan informasi apa pun tentang hasil integrasi Anda ke klien Anda.

`$default`Rute

Setiap API Gateway WebSocket API dapat memiliki `$default` rute. Ini adalah nilai routing khusus yang dapat digunakan dengan cara-cara berikut:

- Anda dapat menggunakannya bersama dengan kunci rute yang ditentukan, untuk menentukan rute "fallback" (misalnya, integrasi tiruan generik yang mengembalikan pesan kesalahan tertentu) untuk pesan masuk yang tidak cocok dengan kunci rute yang ditentukan.
- Anda dapat menggunakannya tanpa kunci rute yang ditentukan, untuk menentukan model proxy yang mendelegasikan perutean ke komponen backend.
- Anda dapat menggunakannya untuk menentukan rute untuk muatan non-JSON.

Rute khusus

Jika Anda ingin menjalankan integrasi tertentu berdasarkan konten pesan, Anda dapat melakukannya dengan membuat rute khusus.

Rute khusus menggunakan kunci rute dan integrasi yang Anda tentukan. Saat pesan masuk berisi properti JSON, dan properti tersebut mengevaluasi nilai yang cocok dengan nilai kunci rute, API Gateway akan memanggil integrasi. (Untuk informasi selengkapnya, lihat the section called "Tentang WebSocket API".)

Misalnya, Anda ingin membuat aplikasi ruang obrolan. Anda dapat memulai dengan membuat WebSocket API yang ekspresi pemilihan rutenya`$request.body.action`. Anda kemudian dapat menentukan dua rute: `joinroom` dan`sendmessage`. Aplikasi klien mungkin memanggil `joinroom` rute dengan mengirim pesan seperti berikut:

```
{"action":"joinroom","roomname":"developers"}
```

Dan itu mungkin memanggil `sendmessage` rute dengan mengirim pesan seperti berikut:

```
{"action":"sendmessage","message":"Hello everyone"}
```

Menggunakan integrasi API Gateway WebSocket API untuk terhubung ke logika bisnis Anda

Setelah menyiapkan rute untuk API Gateway WebSocket API, Anda harus menentukan integrasi yang ingin Anda gunakan. Seperti halnya rute, yang dapat memiliki permintaan rute dan respons

rute, integrasi dapat memiliki permintaan integrasi dan respons integrasi. Permintaan integrasi berisi informasi yang diharapkan oleh backend Anda untuk memproses permintaan yang berasal dari klien Anda. Respons integrasi berisi data yang dikembalikan backend Anda ke API Gateway, dan yang dapat digunakan untuk membuat pesan yang akan dikirim ke klien (jika respons rute ditentukan).

Untuk informasi selengkapnya tentang menyiapkan integrasi, lihatthe section called “Integrasi”.

Perbedaan penting antara WebSocket API dan REST API

Integrasi untuk WebSocket API mirip dengan integrasi untuk REST API, kecuali untuk perbedaan berikut:

- Saat ini, di konsol API Gateway Anda harus membuat rute terlebih dahulu dan kemudian membuat integrasi sebagai target rute tersebut. Namun, di API dan CLI, Anda dapat membuat rute dan integrasi secara independen, dalam urutan apa pun.
- Anda dapat menggunakan integrasi tunggal untuk beberapa rute. Misalnya, jika Anda memiliki serangkaian tindakan yang terkait erat satu sama lain, Anda mungkin ingin semua rute tersebut menuju ke satu fungsi Lambda. Daripada mendefinisikan detail integrasi beberapa kali, Anda dapat menentukannya sekali dan menetapkannya ke setiap rute terkait.

  > **Note**
  >
  > Saat ini konsol tidak mendukung penggunaan kembali integrasi, jadi Anda harus membuat rute terlebih dahulu dan kemudian membuat integrasi untuk rute itu.
  > Di AWS CLI dan AWS SDK, Anda dapat menggunakan kembali integrasi dengan menyetel target rute ke nilai"`integrations/`*`{integration-id}`*", di *`{integration-id}`*" mana ID unik integrasi yang akan dikaitkan dengan rute.

- API Gateway menyediakan beberapa ekspresi pilihan yang dapat Anda gunakan dalam rute dan integrasi Anda. Anda tidak perlu bergantung pada jenis konten untuk memilih template input atau pemetaan output. Seperti ekspresi pemilihan rute, Anda dapat menentukan ekspresi seleksi yang akan dievaluasi oleh API Gateway untuk memilih item yang tepat. Semuanya akan kembali ke `$default` template jika template yang cocok tidak ditemukan.
  - Dalam permintaan integrasi, ekspresi pemilihan template mendukung `$request.body.`*`<json_path_expression>`* dan nilai statis.
  - Dalam tanggapan integrasi, ekspresi pemilihan template mendukung`$request.body.`*`<json_path_expression>`*,,

`$integration.response.statuscode$integration.response.header.<headerName>`, dan nilai statis.

Dalam protokol HTTP, di mana permintaan dan tanggapan dikirim secara serempak; komunikasi pada dasarnya satu arah. Dalam WebSocket protokol, komunikasi adalah dua arah. Respons bersifat asinkron dan tidak harus diterima oleh klien dalam urutan yang sama dengan pesan klien yang dikirim. Selain itu, backend dapat mengirim pesan ke klien.

> **Note**
>
> Untuk rute yang dikonfigurasi untuk digunakan `AWS_PROXY` atau `LAMBDA_PROXY` diintegrasikan, komunikasi dilakukan satu arah, dan API Gateway tidak akan meneruskan respons backend ke respons rute secara otomatis. Misalnya, dalam kasus `LAMBDA_PROXY` integrasi, badan yang dikembalikan fungsi Lambda tidak akan dikembalikan ke klien. Jika Anda ingin klien menerima tanggapan integrasi, Anda harus menentukan respons rute untuk memungkinkan komunikasi dua arah.

Mengirim data dari layanan backend ke klien yang terhubung

API Gateway WebSocket API menawarkan cara-cara berikut bagi Anda untuk mengirim data dari layanan backend ke klien yang terhubung:

- Integrasi dapat mengirim respons, yang dikembalikan ke klien dengan respons rute yang telah Anda tentukan.
- Anda dapat menggunakan `@connections` API untuk mengirim permintaan POST. Lihat informasi yang lebih lengkap di the section called “Gunakan `@connections` perintah di layanan backend Anda”.

Ekspresi pemilihan WebSocket di API Gateway

Topik

API Gateway menggunakan ekspresi seleksi sebagai cara untuk mengevaluasi konteks permintaan dan respons dan menghasilkan kunci. Kuncinya kemudian digunakan untuk memilih dari serangkaian nilai yang mungkin, biasanya disediakan oleh Anda, pengembang API. Set yang tepat dari variabel yang didukung akan bervariasi tergantung pada ekspresi tertentu. Setiap ekspresi dibahas secara lebih rinci di bawah ini.

Untuk semua ekspresi, bahasa mengikuti seperangkat aturan yang sama:

- Sebuah variabel diawali dengan`"$"`.
- kurung kurawal dapat digunakan untuk secara eksplisit mendefinisikan batas-batas variabel, misalnya,`"${request.body.version}-beta"`.
- Beberapa variabel didukung, tetapi evaluasi hanya terjadi sekali (tidak ada evaluasi rekursif).
- Tanda dolar ($) dapat melarikan diri dengan`"\"`. Hal ini paling berguna ketika mendefinisikan ekspresi yang memetakan ke reserved`$default`kunci, misalnya,`"\$default"`.
- Dalam beberapa kasus, format pola diperlukan. Dalam hal ini, ekspresi harus dibungkus dengan garis miring ke depan (`"/"`), mis.`"/2\d\d/"`untuk mencocokkan2*XX*kode status

Ekspresi pemilihan respons

SEBUAHrespon rutedigunakan untuk pemodelan respon dari backend ke klien. Untuk API WebSocket, respons rute bersifat opsional. Ketika didefinisikan, sinyal ke API Gateway bahwa itu harus mengembalikan respon ke klien setelah menerima pesan WebSocket.

Evaluasiekspresi pemilihan respon rutemenghasilkan kunci respon rute. Akhirnya, kunci ini akan digunakan untuk memilih dari salah satu`RouteResponses`terkait dengan API. Namun, saat ini hanya`$default`kunci didukung.

Ekspresi pemilihan kunci API

Ekspresi ini dievaluasi ketika layanan menentukan permintaan yang diberikan harus dilanjutkan hanya jika klien memberikan validKunci API.

Saat ini hanya dua nilai yang didukung`$request.header.x-api-key`dan`$context.authorizer.usageIdentifierKey`.

Ekspresi pemilihan pemetaan API

Ekspresi ini dievaluasi untuk menentukan tahap API mana yang dipilih saat permintaan dibuat menggunakan domain kustom.

Saat ini, satu-satunya nilai yang didukung adalah `$request.basepath`.

Ringkasan ekspresi seleksi WebSocket

Tabel berikut merangkum kasus penggunaan untuk ekspresi seleksi di WebSocket API:

Ekspresi seleksi	Mengevaluasi ke kunci untuk	Catatan	Contoh kasus penggunaan
`Api.RouteSelectionExpression`	`Route.RouteKey`	`$defaul` sebagai rute catch-all.	Route g pesan WebSocket berdasarkan konteks permintaan klien.
`Route.ModelSelectionExpression`	Kunci untuk`Route.RequestModels`	Tidak wajib. Jika disediakan untuk integrasi non-proxy, validasi model terjadi. `$defaul` sebagai catch-all.	Lakukan validasi permintaan secara dinamis dalam rute yang sama. g

Ekspresi seleksi	Mengevaluasi ke kunci untuk	Catatan	Contoh kasus penggunaan
`Integration.TemplateSelectionExpression`	Kunci untuk`Integration.RequestTemplates`	Tidak wajib. Dapat disediakan untuk integrasi non-proxy untuk memanipulasi muatan masuk. `${reque .body.j nPath}` nilai-nilai statis yang didukung `$defaul` sebagai catch-all.	Memanipulasi permintaan pemanggil berdasarkan properti dinamis permintaan. g

Ekspresi seleksi	Mengevaluasi ke kunci untuk	Catatan	Contoh kasus penggunaan
`Integration.IntegrationResponseSelectionExpression`	`IntegrationResponse.IntegrationResponseKey`	Tidak wajib. Dapat disediakan untuk integrasi non-proxy. Bertindak sebagai kecocokan pola untuk pesan kesalahan (dari Lambda) atau kode status (dari integrasi HTTP). `$default` untuk integrasi non-proxy	Memanipulasi respons dari backend. Pilih tindakan yang akan terjadi berdasarkan respons dinamis backend (misalnya, menangani kesalahan tertentu secara jelas).

Ekspresi seleksi	Mengevaluasi ke kunci untuk	Catatan	Contoh kasus penggunaan
		untuk bertindak sebagai menangk semua untuk respon yang berhasil.	

Ekspresi seleksi	Mengevaluasi ke kunci untuk	Catatan	Contoh kasus penggunaan
`IntegrationResponse.TemplateSelectionExpression`	Kunci untuk`IntegrationResponse.ResponseTemplates`	Tidak wajib. Dapat disediakan untuk integrasi non-proxy. $ default didukung	Dalam beberapa kasus, properti dinamis respon dapat mendikte transformasi yang berbeda dalam rute yang sama dan integrasi terkait. `${request.body.jsonPath} ,${integration.response.statuscode} ,${integration.response.header.headerName} ,${integration.response.mult`

Ekspresi seleksi	Mengevaluasi ke kunci untuk	Catatan	Contoh kasus penggunaan
			`ivalueheader.headerName}` , dan nilai-nilai statis yang didukung. `$defaultdidul` sebagai catch-all.

Ekspresi seleksi	Mengevaluasi ke kunci untuk	Catatan	Contoh kasus penggunaan
`Route.RouteResponseSelectionExpression`	`RouteResponse.RouteResponseKey`	Harus disediakan untuk memulai komunikasi dua arah untuk rute WebSocket. Saat ini, nilai ini dibatasi untuk$de	nya
`RouteResponse.ModelSelectionExpression`	Kunci untuk`RouteResponse.RequestModels`	Saat ini tidak didukung	

Mengembangkan WebSocket API di API Gateway

Bagian ini memberikan detail tentang kemampuan API Gateway yang Anda butuhkan saat mengembangkan API Gateway API.

Saat Anda mengembangkan API Gateway API, Anda memutuskan sejumlah karakteristik API Anda. Karakteristik ini bergantung pada kasus penggunaan API Anda. Misalnya, Anda mungkin hanya ingin

mengizinkan klien tertentu untuk memanggil API Anda, atau Anda mungkin ingin itu tersedia untuk semua orang. Anda mungkin ingin panggilan API untuk menjalankan fungsi Lambda, membuat kueri database, atau memanggil aplikasi.

Topik

Membuat WebSocket API di API Gateway

Anda dapat membuat WebSocket API di konsol API Gateway, dengan menggunakan AWS CLI `create-api`perintah, atau dengan menggunakan `CreateApi` perintah dalam AWS SDK. Prosedur berikut menunjukkan cara membuat WebSocket API baru.

> **Note**
>
> WebSocket API hanya mendukung TLS 1.2. Versi TLS sebelumnya tidak didukung.

Buat WebSocket API menggunakan AWS CLI perintah

Membuat WebSocket API menggunakan `create-api`perintah pemanggilan AWS CLI membutuhkan seperti yang ditunjukkan pada contoh berikut, yang membuat API dengan ekspresi pemilihan `$request.body.action` rute:

```
aws apigatewayv2 --region us-east-1 create-api --name "myWebSocketApi3" --protocol-type
 WEBSOCKET --route-selection-expression '$request.body.action'
```

Contoh keluaran:

```
{
```

```
    "ApiKeySelectionExpression": "$request.header.x-api-key",
    "Name": "myWebSocketApi3",
    "CreatedDate": "2018-11-15T06:23:51Z",
    "ProtocolType": "WEBSOCKET",
    "RouteSelectionExpression": "'$request.body.action'",
    "ApiId": "aabbccddee"
}
```

Membuat WebSocket API menggunakan konsol API Gateway

Anda dapat membuat WebSocket API di konsol dengan memilih WebSocket protokol dan memberi API nama.

> **Important**
>
> Setelah Anda membuat API, Anda tidak dapat mengubah protokol yang telah Anda pilih untuk itu. Tidak ada cara untuk mengubah WebSocket API menjadi REST API atau sebaliknya.

Untuk membuat WebSocket API menggunakan konsol API Gateway

1. Masuk ke konsol API Gateway dan pilih Buat API.
2. Di bawah WebSocket API, pilih Build. Hanya titik akhir Regional yang didukung.
3. Untuk nama API, masukkan nama API Anda.
4. Untuk ekspresi pemilihan Rute, masukkan nilai. Sebagai contoh, `$request.body.action`.

   Untuk informasi selengkapnya tentang ekspresi pemilihan rute, lihatthe section called “”.
5. Lakukan salah satu dari berikut:
   - Pilih Buat API kosong untuk membuat API tanpa rute.
   - Pilih Berikutnya untuk melampirkan rute ke API Anda.

   Anda dapat melampirkan rute setelah membuat API.

Bekerja dengan rute untuk WebSocket API

Di WebSocket API Anda, pesan JSON yang masuk diarahkan ke integrasi backend berdasarkan rute yang Anda konfigurasikan. (Pesan non-JSON diarahkan ke `$default` rute yang Anda konfigurasikan.)

Rute menyertakan kunci rute, yang merupakan nilai yang diharapkan setelah ekspresi pemilihan rute dievaluasi. `routeSelectionExpression`Ini adalah atribut yang didefinisikan pada tingkat API. Ini menentukan properti JSON yang diharapkan hadir dalam payload pesan. Untuk informasi selengkapnya tentang ekspresi pemilihan rute, lihatthe section called “”.

Misalnya, jika pesan JSON Anda berisi `action` properti dan Anda ingin melakukan tindakan yang berbeda berdasarkan properti ini, ekspresi pemilihan rute Anda mungkin`${request.body.action}`. Tabel routing Anda akan menentukan tindakan mana yang akan dilakukan dengan mencocokkan nilai `action` properti terhadap nilai kunci rute kustom yang telah Anda tentukan dalam tabel.

Ada tiga rute standar yang dapat digunakan:`$connect`,`$disconnect`, dan`$default`. Selain itu, Anda dapat membuat rute khusus.

- API Gateway memanggil `$connect` rute saat koneksi persisten antara klien dan WebSocket API sedang dimulai.
- API Gateway memanggil `$disconnect` rute saat klien atau server terputus dari API.
- API Gateway memanggil rute kustom setelah ekspresi pemilihan rute dievaluasi terhadap pesan jika rute yang cocok ditemukan; kecocokan menentukan integrasi mana yang dipanggil.
- API Gateway memanggil `$default` rute jika ekspresi pemilihan rute tidak dapat dievaluasi terhadap pesan atau jika tidak ditemukan rute yang cocok.

Ekspresi pemilihan rute

Ekspresi pemilihan rute dievaluasi saat layanan memilih rute yang akan diikuti untuk pesan masuk. Layanan menggunakan rute yang `routeKey` persis sama dengan nilai yang dievaluasi. Jika tidak ada yang cocok dan rute dengan kunci `$default` rute ada, rute itu dipilih. Jika tidak ada rute yang cocok dengan nilai yang dievaluasi dan tidak ada `$default` rute, layanan mengembalikan kesalahan. Untuk API WebSocket berbasis, ekspresi harus berupa`$request.body.`*`{path_to_body_element}`*.

Misalnya, Anda mengirim pesan JSON berikut:

```
{
    "service" : "chat",
    "action" : "join",
    "data" : {
        "room" : "room1234"
   }
```

```
}
```

Anda mungkin ingin memilih perilaku API berdasarkan `action` properti. Dalam hal ini, Anda dapat menentukan ekspresi pemilihan rute berikut:

```
$request.body.action
```

Dalam contoh ini, `request.body` mengacu pada payload JSON pesan Anda, dan `.action` merupakan ekspresi JSONPath. Anda dapat menggunakan ekspresi jalur JSON apa pun setelahnya`request.body`, tetapi perlu diingat bahwa hasilnya akan dirangkai. Misalnya, jika ekspresi JsonPath Anda mengembalikan array dari dua elemen, yang akan disajikan sebagai string. `"[item1, item2]"` Untuk alasan ini, ini adalah praktik yang baik untuk membuat ekspresi Anda mengevaluasi nilai dan bukan array atau objek.

Anda cukup menggunakan nilai statis, atau Anda dapat menggunakan beberapa variabel. Tabel berikut menunjukkan contoh dan hasil evaluasi mereka terhadap muatan sebelumnya.

Ekspresi	Hasil yang dievaluasi	Deskripsi
`$request.body.action`	`join`	Variabel yang tidak dibungkus
`${request.body.action}`	`join`	Variabel yang dibungkus
`${request.body.service}/${request.body.action}`	`chat/join`	Beberapa variabel dengan nilai statis
`${request.body.action}-${request.bod`	`join-`	Jika JsonPath tidak ditemukan

Ekspresi	Hasil yang dievaluasi	Deskripsi
`y.invalid Path}`		, variabel diselesaikan sebagai “”.
`action`	`action`	Nilai statis
`\$default`	`$default`	Nilai statis

Hasil evaluasi digunakan untuk menemukan rute. Jika ada rute dengan kunci rute yang cocok, rute dipilih untuk memproses pesan. Jika tidak ditemukan rute yang cocok, API Gateway mencoba menemukan `$default` rute jika tersedia. Jika `$default` rute tidak ditentukan, API Gateway mengembalikan kesalahan.

Menyiapkan rute untuk WebSocket API di API Gateway

Saat pertama kali membuat WebSocket API baru, ada tiga rute yang telah ditentukan:`$connect`,`$disconnect`, dan`$default`. Anda dapat membuatnya dengan menggunakan konsol, API, atauAWS CLI. Jika diinginkan, Anda dapat membuat rute khusus. Untuk informasi selengkapnya, lihat the section called “Tentang WebSocket API”.

> **Note**
>
> Di CLI, Anda dapat membuat rute sebelum atau setelah Anda membuat integrasi, dan Anda dapat menggunakan kembali integrasi yang sama untuk beberapa rute.

Membuat rute menggunakan konsol API Gateway

Untuk membuat rute menggunakan konsol API Gateway

1. Masuk ke konsol API Gateway, pilih API, dan pilih Rute.
2. Pilih Buat rute

3. Untuk tombol Rute, masukkan nama kunci rute. Anda dapat membuat rute yang telah ditentukan (`$connect`,`$disconnect`, dan`$default`), atau rute khusus.

   > **Note**
   >
   > Saat Anda membuat rute khusus, jangan gunakan $ awalan dalam nama kunci rute. Awalan ini dicadangkan untuk rute yang telah ditentukan.

4. Pilih dan konfigurasikan jenis integrasi untuk rute. Untuk informasi selengkapnya, lihat the section called “Menyiapkan permintaan integrasi WebSocket API menggunakan konsol API Gateway”.

Buat rute menggunakan AWS CLI

Untuk membuat rute menggunakanAWS CLI, panggil `create-route`seperti yang ditunjukkan pada contoh berikut:

```
aws apigatewayv2 --region us-east-1 create-route --api-id aabbccddee --route-key
 $default
```

Contoh keluaran:

```
{
    "ApiKeyRequired": false,
    "AuthorizationType": "NONE",
    "RouteKey": "$default",
    "RouteId": "1122334"
}
```

Tentukan pengaturan permintaan rute untuk **`$connect`**

Saat Anda menyiapkan `$connect` rute untuk API Anda, pengaturan opsional berikut tersedia untuk mengaktifkan otorisasi API Anda. Untuk informasi selengkapnya, lihat the section called “`$connectRute`”.

- Otorisasi: Jika tidak diperlukan otorisasi, Anda dapat menentukan. `NONE` Jika tidak, Anda dapat menentukan:
  - `AWS_IAM`untuk menggunakan kebijakan AWS IAM standar untuk mengontrol akses ke API Anda.

- `CUSTOM`untuk mengimplementasikan otorisasi untuk API dengan menentukan fungsi otorisasi Lambda yang telah Anda buat sebelumnya. Authorizer dapat berada di akun Anda sendiri atau AWS akun lainAWS. Untuk informasi selengkapnya tentang otorisasi Lambda, lihat. Gunakan otorisasi API Gateway Lambda

  > **Note**
  >
  > Di konsol API Gateway, `CUSTOM` pengaturan hanya terlihat setelah Anda menyiapkan fungsi otorisasi seperti yang dijelaskan dalamthe section called “Konfigurasikan otorisasi Lambda menggunakan konsol”.

  > **Important**
  >
  > Pengaturan Otorisasi diterapkan ke seluruh API, bukan hanya `$connect` rute. `$connect`Rute melindungi rute lain, karena dipanggil pada setiap koneksi.

- Kunci API Diperlukan: Anda dapat secara opsional meminta kunci API untuk `$connect` rute API. Anda dapat menggunakan kunci API bersama dengan rencana penggunaan untuk mengontrol dan melacak akses ke API Anda. Untuk informasi selengkapnya, lihat the section called “Paket penggunaan”.

Siapkan permintaan **`$connect`** rute menggunakan konsol API Gateway

Untuk menyiapkan permintaan `$connect` rute WebSocket API menggunakan konsol API Gateway:

1. Masuk ke konsol API Gateway, pilih API, dan pilih Rute.
2. Di bawah Rute`$connect`, pilih, atau buat `$connect` rute dengan mengikutithe section called “Membuat rute menggunakan konsol API Gateway”.
3. Di bagian Pengaturan permintaan rute, pilih Edit.
4. Untuk Otorisasi, pilih jenis otorisasi.
5. Untuk mewajibkan API untuk `$connect` rute tersebut, pilih Memerlukan kunci API.
6. Pilih Save changes (Simpan perubahan).

Menyiapkan respons rute untuk WebSocket API di API Gateway

WebSocket rute dapat dikonfigurasi untuk komunikasi dua arah atau satu arah. API Gateway tidak akan meneruskan respons backend ke respons rute, kecuali jika Anda menyiapkan respons rute.

> **Note**
>
> Anda hanya dapat menentukan respons `$default` rute untuk WebSocket API. Anda dapat menggunakan respons integrasi untuk memanipulasi respons dari layanan backend. Untuk informasi selengkapnya, lihat the section called “Ikhtisar tanggapan integrasi”.

Anda dapat mengonfigurasi respons rute dan ekspresi pemilihan respons menggunakan konsol API Gateway atau SDK AWS CLI atau AWS SDK.

Untuk informasi selengkapnya tentang ekspresi pemilihan respons rute, lihatthe section called “”.

Topik

- Menyiapkan respons rute menggunakan konsol API Gateway
- Siapkan respons rute menggunakan AWS CLI

Menyiapkan respons rute menggunakan konsol API Gateway

Setelah membuat WebSocket API dan melampirkan fungsi Lambda proxy ke rute default, Anda dapat mengatur respons rute menggunakan konsol API Gateway:

1. Masuk ke konsol API Gateway, pilih WebSocket API dengan integrasi fungsi Lambda proxy pada `$default` rute.
2. Di bawah Rute, pilih `$default` rute.
3. Pilih Aktifkan komunikasi dua arah.
4. Pilih Deploy API.
5. Menerapkan API Anda ke panggung.

Gunakan perintah wscat berikut untuk terhubung ke API Anda. Untuk informasi selengkapnya tentang `wscat`, lihat the section called “Gunakan `wscat` untuk terhubung ke WebSocket API dan mengirim pesan ke sana”.

```
wscat -c wss://api-id.execute-api.us-east-2.amazonaws.com/test
```

Tekan tombol enter untuk memanggil rute default. Tubuh fungsi Lambda Anda harus kembali.

Siapkan respons rute menggunakan AWS CLI

Untuk menyiapkan respons rute untuk WebSocket API menggunakanAWS CLI, panggil `create-route-response`perintah seperti yang ditunjukkan pada contoh berikut. Anda dapat mengidentifikasi ID API dan ID rute dengan memanggil `get-apis`dan `get-routes`.

```
aws apigatewayv2 create-route-response \
    --api-id aabbccddee \
    --route-id 1122334  \
    --route-response-key '$default'
```

Contoh keluaran:

```
{
    "RouteResponseId": "abcdef",
    "RouteResponseKey": "$default"
}
```

Menyiapkan **`$connect`** rute yang membutuhkan WebSocket subprotokol

Klien dapat menggunakan `Sec-WebSocket-Protocol` bidang untuk meminta WebSocket subprotokol selama koneksi ke WebSocket API Anda. Anda dapat menyiapkan integrasi untuk `$connect` rute untuk mengizinkan koneksi hanya jika klien meminta subprotokol yang didukung API Anda.

Contoh berikut fungsi Lambda mengembalikan `Sec-WebSocket-Protocol` header ke klien. Fungsi ini membuat koneksi ke API Anda hanya jika klien menentukan subprotokol. `myprotocol`

Untuk AWS CloudFormation template yang membuat contoh integrasi proxy API dan Lambda ini, lihat. `ws-subprotocol.yaml`

```
export const handler = async (event) => {
    if (event.headers != undefined) {
        const headers = toLowerCaseProperties(event.headers);
```

```
        if (headers['sec-websocket-protocol'] != undefined) {
            const subprotocolHeader = headers['sec-websocket-protocol'];
            const subprotocols = subprotocolHeader.split(',');

            if (subprotocols.indexOf('myprotocol') >= 0) {
                const response = {
                    statusCode: 200,
                    headers: {
                        "Sec-WebSocket-Protocol" : "myprotocol"
                    }
                };
                return response;
            }
        }
    }

    const response = {
        statusCode: 400
    };

    return response;
};

function toLowerCaseProperties(obj) {
    var wrapper = {};
    for (var key in obj) {
        wrapper[key.toLowerCase()] = obj[key];
    }
    return wrapper;
}
```

Anda dapat menggunakan [wscat](untuk menguji apakah API Anda mengizinkan koneksi hanya jika klien meminta subprotokol yang didukung API Anda. Perintah berikut menggunakan `-s` bendera untuk menentukan subprotokol selama koneksi.

Perintah berikut mencoba koneksi dengan subprotokol yang tidak didukung. Karena klien menentukan `chat1` subprotokol, integrasi Lambda mengembalikan kesalahan 400, dan koneksi tidak berhasil.

```
wscat -c wss://api-id.execute-api.region.amazonaws.com/beta -s chat1
error: Unexpected server response: 400
```

Perintah berikut mencakup subprotokol yang didukung dalam permintaan koneksi. Integrasi Lambda memungkinkan koneksi.

```
wscat -c wss://api-id.execute-api.region.amazonaws.com/beta -s chat1,myprotocol
connected (press CTRL+C to quit)
```

Untuk mempelajari lebih lanjut tentang menjalankan WebSocket API, lihatMemanggil API WebSocket .

Mengontrol dan mengelola akses ke WebSocket API di API Gateway

API Gateway mendukung beberapa mekanisme untuk mengontrol dan mengelola akses ke WebSocket API Anda.

Anda dapat menggunakan mekanisme berikut untuk otentikasi dan otorisasi:

- Peran dan kebijakan AWS IAM standar menawarkan kontrol akses yang fleksibel dan kuat. Anda dapat menggunakan peran dan kebijakan IAM untuk mengontrol siapa yang dapat membuat dan mengelola API Anda, serta siapa yang dapat memanggilnya. Untuk informasi selengkapnya, lihat Menggunakan otorisasi IAM.
- Tag IAM dapat digunakan bersama dengan kebijakan IAM untuk mengontrol akses. Untuk informasi selengkapnya, lihat Menggunakan tanda untuk mengontrol API Gateway sumber daya.
- Lambda Authorizer adalah fungsi Lambda yang mengontrol akses ke API. Untuk informasi selengkapnya, lihat Membuat fungsi otorisasi Lambda REQUEST.

Topik

- Menggunakan otorisasi IAM
- Membuat fungsi otorisasi Lambda REQUEST

Menggunakan otorisasi IAM

Otorisasi IAM di WebSocket API mirip dengan yang untuk REST API, dengan pengecualian berikut:

- `execute-api`Tindakan mendukung `ManageConnections` selain tindakan yang ada (`Invoke`,`InvalidateCache`). `ManageConnections`mengontrol akses ke @connections API.
- WebSocket rute menggunakan format ARN yang berbeda:

```
arn:aws:execute-api:region:account-id:api-id/stage-name/route-key
```

- `@connections`API menggunakan format ARN yang sama dengan REST API:

```
arn:aws:execute-api:region:account-id:api-id/stage-name/POST/@connections
```

> **Important**
>
> Ketika Anda menggunakan otorisasi IAM, Anda harus menandatangani permintaan dengan Signature Version 4 (SigV4).

Misalnya, Anda dapat menyiapkan kebijakan berikut ke klien. Contoh ini memungkinkan setiap orang untuk mengirim pesan (`Invoke`) untuk semua rute kecuali untuk rute rahasia di `prod` panggung dan mencegah semua orang mengirim pesan kembali ke klien yang terhubung (`ManageConnections`) untuk semua tahapan.

```
{
  "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/*"
      ]
    }
  ]
}
```

Membuat fungsi otorisasi Lambda **REQUEST**

Fungsi otorisasi Lambda di WebSocket API mirip dengan fungsi untuk REST API, dengan pengecualian berikut:

- Anda hanya dapat menggunakan fungsi otorisasi Lambda untuk rute tersebut. `$connect`
- Anda tidak dapat menggunakan variabel jalur (`event.pathParameters`), karena jalurnya sudah diperbaiki.
- `event.methodArn`berbeda dari setara REST API-nya, karena tidak memiliki metode HTTP. Dalam kasus`$connect`, `methodArn` diakhiri dengan`"$connect"`:

  ```
  arn:aws:execute-api:region:account-id:api-id/stage-name/$connect
  ```

- Variabel konteks `event.requestContext` berbeda dari variabel untuk REST API.

Contoh berikut menunjukkan input ke `REQUEST` authorizer untuk WebSocket API:

```
{
    "type": "REQUEST",
    "methodArn": "arn:aws:execute-api:us-east-1:123456789012:abcdef123/default/
$connect",
    "headers": {
        "Connection": "upgrade",
        "content-length": "0",
        "HeaderAuth1": "headerValue1",
        "Host": "abcdef123.execute-api.us-east-1.amazonaws.com",
        "Sec-WebSocket-Extensions": "permessage-deflate; client_max_window_bits",
        "Sec-WebSocket-Key": "...",
        "Sec-WebSocket-Version": "13",
        "Upgrade": "websocket",
```

```
        "X-Amzn-Trace-Id": "...",
        "X-Forwarded-For": "...",
        "X-Forwarded-Port": "443",
        "X-Forwarded-Proto": "https"
    },
    "multiValueHeaders": {
        "Connection": [
            "upgrade"
        ],
        "content-length": [
            "0"
        ],
        "HeaderAuth1": [
            "headerValue1"
        ],
        "Host": [
            "abcdef123.execute-api.us-east-1.amazonaws.com"
        ],
        "Sec-WebSocket-Extensions": [
            "permessage-deflate; client_max_window_bits"
        ],
        "Sec-WebSocket-Key": [
            "..."
        ],
        "Sec-WebSocket-Version": [
            "13"
        ],
        "Upgrade": [
            "websocket"
        ],
        "X-Amzn-Trace-Id": [
            "..."
        ],
        "X-Forwarded-For": [
            "..."
        ],
        "X-Forwarded-Port": [
            "443"
        ],
        "X-Forwarded-Proto": [
            "https"
        ]
    },
    "queryStringParameters": {
```

```
        "QueryString1": "queryValue1"
    },
    "multiValueQueryStringParameters": {
        "QueryString1": [
            "queryValue1"
        ]
    },
    "stageVariables": {},
    "requestContext": {
        "routeKey": "$connect",
        "eventType": "CONNECT",
        "extendedRequestId": "...",
        "requestTime": "19/Jan/2023:21:13:26 +0000",
        "messageDirection": "IN",
        "stage": "default",
        "connectedAt": 1674162806344,
        "requestTimeEpoch": 1674162806345,
        "identity": {
            "sourceIp": "..."
        },
        "requestId": "...",
        "domainName": "abcdef123.execute-api.us-east-1.amazonaws.com",
        "connectionId": "...",
        "apiId": "abcdef123"
    }
}
```

Contoh fungsi Lambda authorizer berikut adalah WebSocket versi fungsi otorisasi Lambda untuk REST API di: the section called “Buat fungsi otorisasi Lambda di konsol Lambda”

Node.js

```
  // A simple REQUEST authorizer example to demonstrate how to use request
  // parameters to allow or deny a request. In this example, a request is
  // authorized if the client-supplied HeaderAuth1 header and QueryString1 query
parameter
  // in the request context match the specified values of
  // of 'headerValue1' and 'queryValue1' respectively.
           export const handler = function(event, context, callback) {
   console.log('Received event:', JSON.stringify(event, null, 2));

  // Retrieve request parameters from the Lambda function input:
  var headers = event.headers;
```

```
    var queryStringParameters = event.queryStringParameters;
    var stageVariables = event.stageVariables;
    var requestContext = event.requestContext;

    // Parse the input for the parameter values
    var tmp = event.methodArn.split(':');
    var apiGatewayArnTmp = tmp[5].split('/');
    var awsAccountId = tmp[4];
    var region = tmp[3];
    var ApiId = apiGatewayArnTmp[0];
    var stage = apiGatewayArnTmp[1];
    var route = apiGatewayArnTmp[2];

    // Perform authorization to return the Allow policy for correct parameters and
    // the 'Unauthorized' error, otherwise.
    var authResponse = {};
    var condition = {};
     condition.IpAddress = {};

    if (headers.HeaderAuth1 === "headerValue1"
        && queryStringParameters.QueryString1 === "queryValue1") {
         callback(null, generateAllow('me', event.methodArn));
     }  else {
         callback("Unauthorized");
     }
}

// Helper function to generate an IAM policy
var generatePolicy = function(principalId, effect, resource) {
    // Required output:
    var authResponse = {};
     authResponse.principalId = principalId;
    if (effect && resource) {
        var policyDocument = {};
         policyDocument.Version = '2012-10-17'; // default version
        policyDocument.Statement = [];
        var statementOne = {};
         statementOne.Action = 'execute-api:Invoke'; // default action
        statementOne.Effect = effect;
         statementOne.Resource = resource;
         policyDocument.Statement[0] = statementOne;
         authResponse.policyDocument = policyDocument;
     }
    // Optional output with custom properties of the String, Number or Boolean type.
```

```
   authResponse.context = {
       "stringKey": "stringval",
       "numberKey": 123,
       "booleanKey": true
    };
   return authResponse;
}

var generateAllow = function(principalId, resource) {
   return generatePolicy(principalId, 'Allow', resource);
}

var generateDeny = function(principalId, resource) {
   return generatePolicy(principalId, 'Deny', resource);
}
```

Python

```
# A simple REQUEST authorizer example to demonstrate how to use request
# parameters to allow or deny a request. In this example, a request is
# authorized if the client-supplied HeaderAuth1 header and QueryString1 query
 parameter
# in the request context match the specified values of
# of 'headerValue1' and 'queryValue1' respectively.

import json


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

    # Retrieve request parameters from the Lambda function input:
    headers = event['headers']
    queryStringParameters = event['queryStringParameters']
    stageVariables = event['stageVariables']
    requestContext = event['requestContext']

    # Parse the input for the parameter values
    tmp = event['methodArn'].split(':')
    apiGatewayArnTmp = tmp[5].split('/')
    awsAccountId = tmp[4]
    region = tmp[3]
    ApiId = apiGatewayArnTmp[0]
```

```
    stage = apiGatewayArnTmp[1]
    route = apiGatewayArnTmp[2]

    # Perform authorization to return the Allow policy for correct parameters
    # and the 'Unauthorized' error, otherwise.

    authResponse = {}
    condition = {}
    condition['IpAddress'] = {}

    if (headers['HeaderAuth1'] ==
            "headerValue1" and queryStringParameters["QueryString1"] ==
 "queryValue1"):
        response = generateAllow('me', event['methodArn'])
        print('authorized')
        return json.loads(response)
    else:
        print('unauthorized')
        return 'unauthorized'

    # Help function to generate IAM policy


def generatePolicy(principalId, effect, resource):
    authResponse = {}
    authResponse['principalId'] = principalId
    if (effect and resource):
        policyDocument = {}
        policyDocument['Version'] = '2012-10-17'
        policyDocument['Statement'] = []
        statementOne = {}
        statementOne['Action'] = 'execute-api:Invoke'
        statementOne['Effect'] = effect
        statementOne['Resource'] = resource
        policyDocument['Statement'] = [statementOne]
        authResponse['policyDocument'] = policyDocument

    authResponse['context'] = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": True
    }

    authResponse_JSON = json.dumps(authResponse)
```

```
    return authResponse_JSON


def generateAllow(principalId, resource):
    return generatePolicy(principalId, 'Allow', resource)


def generateDeny(principalId, resource):
    return generatePolicy(principalId, 'Deny', resource)
```

Untuk mengonfigurasi fungsi Lambda sebelumnya sebagai fungsi REQUEST otorisasi untuk WebSocket API, ikuti prosedur yang sama seperti untuk REST API.

Untuk mengonfigurasi `$connect` rute untuk menggunakan otorisasi Lambda ini di konsol, pilih atau buat rute. `$connect` Di bagian Pengaturan permintaan rute, pilih Edit. Pilih otorisasi Anda di menu tarik-turun Otorisasi, lalu pilih Simpan perubahan.

Untuk menguji otorisasi, Anda perlu membuat koneksi baru. Mengubah otorisasi `$connect` tidak memengaruhi klien yang sudah terhubung. Saat Anda terhubung ke WebSocket API, Anda perlu memberikan nilai untuk sumber identitas yang dikonfigurasi. Misalnya, Anda dapat terhubung dengan mengirimkan string kueri dan header yang valid menggunakan `wscat` seperti pada contoh berikut:

```
wscat -c 'wss://myapi.execute-api.us-east-1.amazonaws.com/beta?
QueryString1=queryValue1' -H HeaderAuth1:headerValue1
```

Jika Anda mencoba untuk terhubung tanpa nilai identitas yang valid, Anda akan menerima `401` tanggapan:

```
wscat -c wss://myapi.execute-api.us-east-1.amazonaws.com/beta
error: Unexpected server response: 401
```

Menyiapkan integrasi WebSocket API

Setelah menyiapkan rute API, Anda harus mengintegrasikannya dengan titik akhir di backend. Endpoint backend juga disebut sebagai titik akhir integrasi dan dapat berupa fungsi Lambda, titik akhir HTTP, atau tindakan layanan. AWS Integrasi API memiliki permintaan integrasi dan respons integrasi.

Di bagian ini, Anda dapat mempelajari cara menyiapkan permintaan integrasi dan respons integrasi untuk WebSocket API Anda.

Topik

Menyiapkan permintaan integrasi WebSocket API di API Gateway

Menyiapkan permintaan integrasi melibatkan hal-hal berikut:

- Memilih kunci rute untuk diintegrasikan ke backend.
- Menentukan titik akhir backend untuk dipanggil. WebSocket API mendukung jenis integrasi berikut:
  - `AWS_PROXY`
  - `AWS`
  - `HTTP_PROXY`
  - `HTTP`
  - `MOCK`

  Untuk informasi selengkapnya tentang jenis integrasi, lihat IntegrationTypedi API REST API Gateway V2.
- Mengkonfigurasi cara mengubah data permintaan rute, jika perlu, menjadi data permintaan integrasi dengan menentukan satu atau beberapa templat permintaan.

Menyiapkan permintaan integrasi WebSocket API menggunakan konsol API Gateway

Untuk menambahkan permintaan integrasi ke rute di WebSocket API menggunakan konsol API Gateway

1. Masuk ke konsol API Gateway, pilih API, dan pilih Rute.
2. Di bawah Rute, pilih rute.
3. Pilih tab Permintaan integrasi, dan kemudian di bagian Pengaturan permintaan integrasi, pilih Edit.
4. Untuk jenis Integrasi, pilih salah satu dari berikut ini:

- Pilih fungsi Lambda hanya jika API Anda akan diintegrasikan dengan AWS Lambda fungsi yang telah Anda buat di akun ini atau di akun lain.

  Untuk membuat fungsi Lambda baru di AWS Lambda, untuk menetapkan izin sumber daya pada fungsi Lambda, atau untuk melakukan tindakan layanan Lambda lainnya, pilih Layanan sebagai gantinya.AWS
- Pilih HTTP jika API Anda akan terintegrasi dengan titik akhir HTTP yang ada. Untuk informasi selengkapnya, lihat Menyiapkan integrasi HTTP di API Gateway.
- Pilih Mock jika Anda ingin menghasilkan respons API dari API Gateway secara langsung, tanpa perlu backend integrasi. Untuk informasi selengkapnya, lihat Siapkan integrasi tiruan di API Gateway.
- Pilih AWS layanan jika API Anda akan terintegrasi dengan AWS layanan.
- Pilih tautan VPC jika API Anda akan menggunakan titik akhir integrasi `VpcLink` sebagai pribadi. Untuk informasi selengkapnya, lihat Siapkan integrasi pribadi API Gateway.

5. Jika Anda memilih fungsi Lambda, lakukan hal berikut:

   a. Untuk Menggunakan integrasi proxy Lambda, pilih kotak centang jika Anda ingin menggunakan integrasi proxy Lambda atau integrasi proxy Lambda lintas akun.

   b. Untuk fungsi Lambda, tentukan fungsi dengan salah satu cara berikut:

      - Jika fungsi Lambda Anda berada di akun yang sama, masukkan nama fungsi dan kemudian pilih fungsi dari daftar dropdown.

        > **Note**
        >
        > Nama fungsi secara opsional dapat menyertakan alias atau spesifikasi versinya, seperti dalam, `HelloWorldHelloWorld:1`, atau. `HelloWorld:alpha`

      - Jika fungsinya ada di akun yang berbeda, masukkan ARN untuk fungsi tersebut.

   c. Untuk menggunakan nilai batas waktu default 29 detik, tetap aktifkan batas waktu default. Untuk menetapkan batas waktu kustom, pilih Batas waktu default dan masukkan nilai batas waktu antara `50` dan milidetik. `29000`

6. Jika Anda memilih HTTP, ikuti petunjuk di langkah 4 darithe section called “ Siapkan permintaan integrasi menggunakan konsol”.

7. Jika Anda memilih Mock, lanjutkan ke langkah Permintaan Template.

8. Jika Anda memilih AWS layanan, ikuti instruksi pada langkah 6 darithe section called “ Siapkan permintaan integrasi menggunakan konsol”.
9. Jika Anda memilih tautan VPC, lakukan hal berikut:

   a. Untuk integrasi proxy VPC, pilih kotak centang jika Anda ingin permintaan Anda diproksi ke titik akhir Anda. `VPCLink`

   b. Untuk metode HTTP, pilih jenis metode HTTP yang paling cocok dengan metode di backend HTTP.

   c. Dari daftar dropdown tautan VPC, pilih tautan VPC. Anda dapat memilih `[Use Stage Variables]` dan memasukkan **`${stageVariables.vpcLinkId}`** dalam kotak teks di bawah daftar.

      Anda dapat menentukan variabel `vpcLinkId` stage setelah menerapkan API ke tahap dan menetapkan nilainya ke ID. `VpcLink`

   d. Untuk URL Endpoint, masukkan URL backend HTTP yang Anda inginkan untuk digunakan integrasi ini.

   e. Untuk menggunakan nilai batas waktu default 29 detik, tetap aktifkan batas waktu default. Untuk menetapkan batas waktu kustom, pilih Batas waktu default dan masukkan nilai batas waktu antara `50` dan milidetik. `29000`

10. Pilih Simpan perubahan.
11. Di bawah templat Permintaan, lakukan hal berikut:

    a. Untuk memasukkan ekspresi pemilihan Template, di bawah Permintaan template, pilih Edit.

    b. Masukkan ekspresi pemilihan Template. Gunakan ekspresi yang dicari API Gateway di payload pesan. Jika ditemukan, itu dievaluasi, dan hasilnya adalah nilai kunci template yang digunakan untuk memilih template pemetaan data yang akan diterapkan ke data dalam payload pesan. Anda membuat template pemetaan data di langkah berikutnya. Pilih Edit untuk menyimpan perubahan Anda.

    c. Pilih Buat template untuk membuat template pemetaan data. Untuk kunci Template, masukkan nilai kunci template yang digunakan untuk memilih template pemetaan data yang akan diterapkan ke data dalam payload pesan. Kemudian, masukkan template pemetaan. Pilih Buat templat.

       Untuk informasi tentang ekspresi pemilihan templat, lihatthe section called “Ekspresi pemilihan template”.

Siapkan permintaan integrasi menggunakan AWS CLI

Anda dapat menyiapkan permintaan integrasi untuk rute di WebSocket API dengan menggunakan AWS CLI seperti pada contoh berikut, yang membuat integrasi tiruan:

1. Buat file bernama`integration-params.json`, dengan konten berikut:

```
{"PassthroughBehavior": "WHEN_NO_MATCH", "TimeoutInMillis": 29000,
 "ConnectionType": "INTERNET", "RequestTemplates": {"application/json":
 "{\"statusCode\":200}"}, "IntegrationType": "MOCK"}
```

2. Jalankan `create-integration`perintah seperti yang ditunjukkan pada contoh berikut:

```
aws apigatewayv2 --region us-east-1 create-integration --api-id aabbccddee --cli-
input-json file://integration-params.json
```

Berikut ini adalah contoh output untuk contoh ini:

```
{
    "PassthroughBehavior": "WHEN_NO_MATCH",
    "TimeoutInMillis": 29000,
    "ConnectionType": "INTERNET",
    "IntegrationResponseSelectionExpression": "${response.statuscode}",
    "RequestTemplates": {
        "application/json": "{\"statusCode\":200}"
    },
    "IntegrationId": "0abcdef",
    "IntegrationType": "MOCK"
}
```

Atau, Anda dapat mengatur permintaan integrasi untuk integrasi proxy dengan menggunakan AWS CLI seperti pada contoh berikut:

1. Buat fungsi Lambda di konsol Lambda dan berikan peran eksekusi Lambda dasar.
2. Jalankan `create-integration`perintah seperti pada contoh berikut:

```
aws apigatewayv2 create-integration --api-id aabbccddee --integration-type
 AWS_PROXY --integration-method POST --integration-uri arn:aws:apigateway:us-
east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:us-
east-1:123412341234:function:simpleproxy-echo-e2e/invocations
```

Berikut ini adalah contoh output untuk contoh ini:

```
{
    "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"
}
```

Format input fungsi Lambda untuk integrasi proxy untuk API WebSocket

Dalam integrasi proxy Lambda, API Gateway memetakan seluruh permintaan klien ke `event` parameter input fungsi Lambda backend. Contoh berikut menunjukkan struktur peristiwa masukan dari `$connect` rute dan peristiwa masukan dari rute yang dikirimkan API Gateway ke integrasi proxy Lambda. `$disconnect`

Input from the $connect route

```
{
    headers: {
      Host: 'abcd123.execute-api.us-east-1.amazonaws.com',
      'Sec-WebSocket-Extensions': 'permessage-deflate; client_max_window_bits',
      'Sec-WebSocket-Key': '...',
      'Sec-WebSocket-Version': '13',
      'X-Amzn-Trace-Id': '...',
      'X-Forwarded-For': '192.0.2.1',
      'X-Forwarded-Port': '443',
      'X-Forwarded-Proto': 'https'
    },
    multiValueHeaders: {
      Host: [ 'abcd123.execute-api.us-east-1.amazonaws.com' ],
      'Sec-WebSocket-Extensions': [ 'permessage-deflate; client_max_window_bits' ],
      'Sec-WebSocket-Key': [ '...' ],
      'Sec-WebSocket-Version': [ '13' ],
      'X-Amzn-Trace-Id': [ '...' ],
      'X-Forwarded-For': [ '192.0.2.1' ],
      'X-Forwarded-Port': [ '443' ],
      'X-Forwarded-Proto': [ 'https' ]
    },
```

```
    requestContext: {
      routeKey: '$connect',
      eventType: 'CONNECT',
      extendedRequestId: 'ABCD1234=',
      requestTime: '09/Feb/2024:18:11:43 +0000',
      messageDirection: 'IN',
      stage: 'prod',
      connectedAt: 1707502303419,
      requestTimeEpoch: 1707502303420,
      identity: { sourceIp: '192.0.2.1' },
      requestId: 'ABCD1234=',
      domainName: 'abcd1234.execute-api.us-east-1.amazonaws.com',
      connectionId: 'AAAA1234=',
      apiId: 'abcd1234'
    },
    isBase64Encoded: false
  }
```

Input from the $disconnect route

```
{
    headers: {
      Host: 'abcd1234.execute-api.us-east-1.amazonaws.com',
      'x-api-key': '',
      'X-Forwarded-For': '',
      'x-restapi': ''
    },
    multiValueHeaders: {
      Host: [ 'abcd1234.execute-api.us-east-1.amazonaws.com' ],
      'x-api-key': [ '' ],
      'X-Forwarded-For': [ '' ],
      'x-restapi': [ '' ]
    },
    requestContext: {
      routeKey: '$disconnect',
      disconnectStatusCode: 1005,
      eventType: 'DISCONNECT',
      extendedRequestId: 'ABCD1234=',
      requestTime: '09/Feb/2024:18:23:28 +0000',
      messageDirection: 'IN',
      disconnectReason: 'Client-side close frame status not set',
      stage: 'prod',
```

```
    connectedAt: 1707503007396,
    requestTimeEpoch: 1707503008941,
    identity: { sourceIp: '192.0.2.1' },
    requestId: 'ABCD1234=',
    domainName: 'abcd1234.execute-api.us-east-1.amazonaws.com',
    connectionId: 'AAAA1234=',
    apiId: 'abcd1234'
  },
  isBase64Encoded: false
}
```

Menyiapkan respons integrasi WebSocket API di API Gateway

Topik

Ikhtisar tanggapan integrasi

Respons integrasi API Gateway adalah cara memodelkan dan memanipulasi respons dari layanan backend. Ada beberapa perbedaan dalam penyiapan REST API versus respons integrasi WebSocket API, tetapi secara konseptual perilakunya sama.

WebSocket rute dapat dikonfigurasi untuk komunikasi dua arah atau satu arah.

- Saat rute dikonfigurasi untuk komunikasi dua arah, respons integrasi memungkinkan Anda mengonfigurasi transformasi pada muatan pesan yang dikembalikan, mirip dengan respons integrasi untuk REST API.
- Jika rute dikonfigurasi untuk komunikasi satu arah, maka terlepas dari konfigurasi respons integrasi apa pun, tidak ada respons yang akan dikembalikan melalui WebSocket saluran setelah pesan diproses.

API Gateway tidak akan meneruskan respons backend ke respons rute, kecuali jika Anda menyiapkan respons rute. Untuk mempelajari cara menyiapkan respons rute, lihatthe section called "Siapkan respons rute WebSocket API".

Respons integrasi untuk komunikasi dua arah

Integrasi dapat dibagi menjadi integrasi proxy dan integrasi non-proxy.

> **Important**
>
> Untuk integrasi proxy, API Gateway secara otomatis meneruskan output backend ke pemanggil sebagai muatan lengkap. Tidak ada respon integrasi.

Untuk integrasi non-proxy, Anda harus menyiapkan setidaknya satu respons integrasi:

- Idealnya, salah satu tanggapan integrasi Anda harus bertindak sebagai tangkapan semua ketika tidak ada pilihan eksplisit yang dapat dibuat. Kasus default ini diwakili dengan menyetel kunci respons integrasi dari`$default`.
- Dalam semua kasus lain, kunci respons integrasi berfungsi sebagai ekspresi reguler. Ini harus mengikuti format`"/expression/"`.

Untuk integrasi HTTP non-proxy:

- API Gateway akan mencoba mencocokkan kode status HTTP dari respons backend. Tombol respons integrasi akan berfungsi sebagai ekspresi reguler dalam kasus ini. Jika kecocokan tidak dapat ditemukan, `$default` maka dipilih sebagai respons integrasi.
- Ekspresi pemilihan template, seperti dijelaskan di atas, berfungsi secara identik. Sebagai contoh:
  - `/2\d\d/`: Menerima dan mengubah tanggapan yang berhasil
  - `/4\d\d/`: Menerima dan mengubah kesalahan permintaan yang buruk
  - `$default`: Menerima dan mengubah semua tanggapan tak terduga

Untuk informasi selengkapnya tentang ekspresi pemilihan templat, lihatthe section called "Ekspresi pemilihan template".

Menyiapkan respons integrasi menggunakan konsol API Gateway

Untuk menyiapkan respons integrasi rute untuk WebSocket API menggunakan konsol API Gateway:

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih WebSocket API Anda dan pilih rute Anda.
3. Pilih tab Permintaan integrasi, dan kemudian di bagian Pengaturan respons integrasi, pilih Buat respons integrasi.
4. Untuk tombol Respons, masukkan nilai yang akan ditemukan di kunci respons dalam pesan keluar setelah mengevaluasi ekspresi pemilihan respons. Misalnya, Anda dapat memasukkan **`/4\d\d/`** untuk menerima dan mengubah kesalahan permintaan buruk atau masuk **`$default`** untuk menerima dan mengubah semua respons yang cocok dengan ekspresi pemilihan templat.
5. Untuk ekspresi pemilihan Template, masukkan ekspresi seleksi untuk mengevaluasi pesan keluar.
6. Pilih Buat respons.
7. Anda juga dapat menentukan template pemetaan untuk mengonfigurasi transformasi payload pesan yang dikembalikan. Pilih Buat templat.
8. Masukkan nama kunci. Jika Anda memilih ekspresi pemilihan template default, masukkan**`\$default`**.
9. Untuk template Response, masukkan template pemetaan Anda di editor kode.
10. Pilih Buat templat.
11. Pilih Deploy API untuk menerapkan API Anda.

Gunakan perintah wscat berikut untuk terhubung ke API Anda. Untuk informasi selengkapnya tentang `wscat`, lihat the section called “Gunakan `wscat` untuk terhubung ke WebSocket API dan mengirim pesan ke sana”.

```
wscat -c wss://api-id.execute-api.us-east-2.amazonaws.com/test
```

Saat Anda memanggil rute Anda, muatan pesan yang dikembalikan akan kembali.

Siapkan respons integrasi menggunakan AWS CLI

Untuk menyiapkan respons integrasi untuk WebSocket API menggunakan AWS CLI panggilan `create-integration-response`perintah. Perintah CLI berikut menunjukkan contoh membuat respons `$default` integrasi:

```
aws apigatewayv2 create-integration-response \
    --api-id vaz7da96z6 \
    --integration-id a1b2c3 \
```

```
    --integration-response-key '$default'
```

Minta validasi

Anda dapat mengonfigurasi API Gateway untuk melakukan validasi pada permintaan rute sebelum melanjutkan dengan permintaan integrasi. Jika validasi gagal, API Gateway gagal permintaan tanpa memanggil backend Anda, mengirimkan respons gateway "Badan permintaan buruk" ke klien, dan menerbitkan hasil validasi di Log. CloudWatch Menggunakan validasi dengan cara ini mengurangi panggilan yang tidak perlu ke backend API Anda.

Ekspresi pemilihan model

Anda dapat menggunakan ekspresi pemilihan model untuk memvalidasi permintaan secara dinamis dalam rute yang sama. Validasi model terjadi jika Anda memberikan ekspresi pemilihan model untuk integrasi proxy atau non-proxy. Anda mungkin perlu mendefinisikan `$default` model sebagai fallback ketika tidak ada model yang cocok ditemukan. Jika tidak ada model yang cocok dan `$default` tidak ditentukan, validasi gagal. Ekspresi seleksi terlihat seperti `Route.ModelSelectionExpression` dan mengevaluasi kunci untuk`Route.RequestModels`.

Saat menentukan rute untuk WebSocket API, Anda dapat menentukan ekspresi pemilihan model secara opsional. Ekspresi ini dievaluasi untuk memilih model yang akan digunakan untuk validasi tubuh ketika permintaan diterima. Ekspresi mengevaluasi ke salah satu entri dalam rute. `requestmodels`

Sebuah model dinyatakan sebagai skema JSON dan menggambarkan struktur data dari badan permintaan. Sifat ekspresi seleksi ini memungkinkan Anda memilih model secara dinamis untuk divalidasi saat runtime untuk rute tertentu. Untuk informasi tentang cara membuat model, lihatthe section called "Memahami model data".

Menyiapkan validasi permintaan menggunakan konsol API Gateway

Contoh berikut menunjukkan cara mengatur validasi permintaan pada rute.

Pertama, Anda membuat model, dan kemudian Anda membuat rute. Selanjutnya, Anda mengonfigurasi validasi permintaan pada rute yang baru saja Anda buat. Terakhir, Anda menerapkan dan menguji API Anda. Untuk menyelesaikan tutorial ini, Anda memerlukan WebSocket API dengan `$request.body.action` ekspresi pemilihan rute dan titik akhir integrasi untuk rute baru Anda.

Anda juga `wscat` perlu terhubung ke API Anda. Untuk informasi selengkapnya, lihat the section called "Gunakan `wscat` untuk terhubung ke WebSocket API dan mengirim pesan ke sana".

Untuk membuat model

1. Masuk ke konsol API Gateway di https://console.aws.amazon.com/apigateway.
2. Pilih WebSocket API.
3. Di panel navigasi utama, pilih Model.
4. Pilih Buat model.
5. Untuk Nama, masukkan **`emailModel`**.
6. Untuk jenis Konten, masukkan**`application/json`**.
7. Untuk skema Model, masukkan model berikut:

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type" : "object",
    "required" : [ "address"],
    "properties" : {
        "address": {
            "type": "string"
        }
    }
}
```

   Model ini mengharuskan permintaan berisi alamat email.
8. Pilih Simpan.

Pada langkah ini, Anda membuat rute untuk WebSocket API Anda.

Untuk membuat rute

1. Di panel navigasi utama, pilih Rute.
2. Pilih Buat rute.
3. Untuk kunci Rute, masukkan**`sendMessage`**.
4. Pilih jenis integrasi dan tentukan titik akhir integrasi. Untuk informasi selengkapnya, lihat the section called “Integrasi”.
5. Pilih Buat rute.

Pada langkah ini, Anda menyiapkan validasi permintaan untuk `sendMessage` rute tersebut.

Untuk mengatur validasi permintaan

1. Pada tab Permintaan rute, di bawah Pengaturan permintaan rute, pilih Edit.
2. Untuk ekspresi pemilihan Model, masukkan**`${request.body.messageType}`**.

   API Gateway menggunakan `messageType` properti untuk memvalidasi permintaan yang masuk.
3. Pilih Tambahkan model permintaan.
4. Untuk kunci Model, masukkan**`email`**.
5. Untuk Model, pilih EmailModel.

   API Gateway memvalidasi pesan masuk dengan `messageType` properti yang disetel ke `email` terhadap model ini.

   > **Note**
   >
   > Jika API Gateway tidak dapat mencocokkan ekspresi pemilihan model dengan kunci model, maka ia memilih `$default` model. Jika tidak ada `$default` model, maka validasi gagal. Untuk API produksi, kami menyarankan Anda membuat `$default` model.
6. Pilih Simpan perubahan.

Pada langkah ini, Anda menerapkan dan menguji API Anda.

Untuk menerapkan dan menguji API

1. Pilih Deploy API.
2. Pilih tahap yang diinginkan dari daftar dropdown atau masukkan nama tahap baru.
3. Pilih Deploy.
4. Di panel navigasi utama, pilih Tahapan.
5. Salin WebSocket URL API Anda. URL akan terlihat seperti `wss://abcdef123.execute-api.us-east-2.amazonaws.com/production`.
6. Buka terminal baru dan jalankan wscat perintah dengan parameter berikut.

```
wscat -c wss://abcdef123.execute-api.us-west-2.amazonaws.com/production
```

```
Connected (press CTRL+C to quit)
```

7. Gunakan perintah berikut untuk menguji API Anda.

```
{"action": "sendMessage", "messageType": "email"}
```

```
{"message": "Invalid request body", "connectionId":"ABCD1=234",
 "requestId":"EFGH="}
```

API Gateway akan gagal permintaan.

Gunakan perintah berikutnya untuk mengirim permintaan yang valid ke API Anda.

```
{"action": "sendMessage", "messageType": "email", "address":
 "mary_major@example.com"}
```

Menyiapkan transformasi data untuk API WebSocket

Di API Gateway, permintaan metode WebSocket API dapat mengambil payload dalam format yang berbeda dari payload permintaan integrasi yang sesuai, seperti yang diperlukan di backend. Demikian pula, backend dapat mengembalikan payload respons integrasi yang berbeda dari payload respons metode, seperti yang diharapkan oleh frontend.

API Gateway memungkinkan Anda menggunakan templat pemetaan untuk memetakan payload dari permintaan metode ke permintaan integrasi yang sesuai dan dari respons integrasi ke respons metode yang sesuai. Anda menentukan ekspresi pemilihan template untuk menentukan template mana yang akan digunakan untuk melakukan transformasi data yang diperlukan.

Anda dapat menggunakan pemetaan data untuk memetakan data dari permintaan rute ke integrasi backend. Untuk mempelajari selengkapnya, lihat the section called “Pemetaan data”.

Templat dan model pemetaan

Template pemetaan adalah skrip yang dinyatakan dalam Velocity Template Language (VTL) dan diterapkan ke payload menggunakan ekspresi JsonPath. Untuk informasi selengkapnya tentang template pemetaan API Gateway, lihatMemahami template pemetaan.

Muatan dapat memiliki model data sesuai dengan rancangan skema JSON 4. Anda tidak perlu mendefinisikan model untuk membuat template pemetaan. Namun, model dapat membantu Anda membuat template karena API Gateway menghasilkan cetak biru template berdasarkan model yang disediakan. Untuk informasi selengkapnya tentang model API Gateway, lihatMemahami model data.

Ekspresi pemilihan template

Untuk mengubah payload dengan template pemetaan, Anda menentukan ekspresi pemilihan template WebSocket API dalam permintaan integrasi atau respons integrasi. Ekspresi ini dievaluasi untuk menentukan template input atau output (jika ada) yang akan digunakan untuk mengubah badan permintaan menjadi badan permintaan integrasi (melalui template input) atau badan respons integrasi ke badan respons rute (melalui template keluaran).

`Integration.TemplateSelectionExpression`mendukung `${request.body.jsonPath}` dan nilai statis.

`IntegrationResponse.TemplateSelectionExpression`mendukung`${request.body.jsonPath}`, dan nilai-nilai statis.

Ekspresi pemilihan respons integrasi

Saat menyiapkan respons integrasi untuk WebSocket API, Anda dapat menentukan ekspresi pemilihan respons integrasi secara opsional. Ekspresi ini menentukan apa yang `IntegrationResponse` harus dipilih ketika integrasi kembali. Nilai ekspresi ini saat ini dibatasi oleh API Gateway, seperti yang didefinisikan di bawah ini. Sadarilah bahwa ekspresi ini hanya relevan untuk integrasi non-proxy; integrasi proxy hanya meneruskan payload respons kembali ke pemanggil tanpa pemodelan atau modifikasi.

Berbeda dengan ekspresi seleksi sebelumnya lainnya, ekspresi ini saat ini mendukung format pencocokan pola. Ekspresi harus dibungkus dengan garis miring ke depan.

Saat ini nilainya tetap tergantung pada`integrationType`:

- Untuk integrasi berbasis Lambda, memang demikian. `$integration.response.body.errorMessage`
- Untuk `HTTP` dan `MOCK` integrasi, itu`$integration.response.statuscode`.
- Untuk `HTTP_PROXY` dan`AWS_PROXY`, ungkapan tidak digunakan karena Anda meminta agar payload diteruskan ke penelepon.

Menyiapkan pemetaan data untuk API WebSocket

Pemetaan data memungkinkan Anda memetakan data dari permintaan rute ke integrasi backend.

> **Note**
>
> Pemetaan data untuk WebSocket API tidak didukung di file. AWS Management Console Anda harus menggunakanAWS CLI,AWS CloudFormation, atau SDK untuk mengkonfigurasi pemetaan data.

Topik

- Memetakan data permintaan rute ke parameter permintaan integrasi
- Contoh

Memetakan data permintaan rute ke parameter permintaan integrasi

Parameter permintaan integrasi dapat dipetakan dari parameter permintaan rute yang ditentukan, badan permintaan, `context`atau `stage`variabel, dan nilai statis.

Pada tabel berikut, *`PARAM_NAME`*adalah nama parameter permintaan rute dari jenis parameter yang diberikan. Itu harus sesuai dengan ekspresi reguler`'^[a-zA-Z0-9._$-]+$]'`. *`JSONPath_expression adalah ekspresi`* JSONPath untuk bidang JSON dari badan permintaan.

Ekspresi pemetaan data permintaan integrasi

Sumber data yang dipetakan	Ekspresi pemetaan
Permintaan query string (didukung hanya untuk `$connect` rute)	`route.request.quer ystring.` *`PARAM_NAME`*
Permintaan header (didukung hanya untuk `$connect` rute)	`route.request.header.` *`PARAM_NAME`*
String permintaan permintaan multi-nilai (didukung hanya untuk `$connect` rute)	`route.request.multivalueque rystring.` *`PARAM_NAME`*

Sumber data yang dipetakan	Ekspresi pemetaan
Header permintaan multi-nilai (didukung hanya untuk `$connect` rute)	`route.request.multivaluehea der.` *`PARAM_NAME`*
Isi permintaan	`route.request.body.` *`JSONPath_ EXPRESSION`*
Variabel tahap	`stageVariables.` *`VARIABLE_NAME`*
Variabel konteks	`context.`*`VARIABLE_NAME`* yang harus menjadi salah satu variabel konteks yang didukung.
Nilai statis	*`'STATIC_VALUE'`* . *`STATIC_VALUE`* adalah string literal dan harus tertutup dalam tanda kutip tunggal.

Contoh

AWS CLIContoh berikut mengkonfigurasi pemetaan data. Untuk contoh AWS CloudFormation template, lihat `websocket-data-mapping.yaml`.

Petakan connectionID klien ke header dalam permintaan integrasi

Contoh perintah berikut memetakan klien `connectionId` ke `connectionId` header dalam permintaan untuk integrasi backend.

```
aws apigatewayv2 update-integration \
    --integration-id abc123 \
    --api-id a1b2c3d4 \
    --request-parameters
 'integration.request.header.connectionId'='context.connectionId'
```

Memetakan parameter string kueri ke header dalam permintaan integrasi

Contoh berikut perintah memetakan parameter string `authToken` query ke `authToken` header dalam permintaan integrasi.

Pertama, tambahkan parameter string `authToken` kueri ke parameter permintaan rute.

```
aws apigatewayv2 update-route --route-id 0abcdef \
    --api-id a1b2c3d4 \
    --request-parameters '{"route.request.querystring.authToken": {"Required": false}}'
```

Selanjutnya, petakan parameter string kueri ke `authToken` header dalam permintaan ke integrasi backend.

```
aws apigatewayv2 update-integration \
    --integration-id abc123 \
    --api-id a1b2c3d4 \
    --request-parameters
 'integration.request.header.authToken'='route.request.querystring.authToken'
```

Jika perlu, hapus parameter string `authToken` kueri dari parameter permintaan rute.

```
aws apigatewayv2 delete-route-request-parameter \
    --route-id 0abcdef \
    --api-id a1b2c3d4 \
    --request-parameter-key 'route.request.querystring.authToken'
```

Referensi pemetaan API API API Gateway WebSocket

Bagian ini merangkum kumpulan variabel yang saat ini didukung untuk API WebSocket di API Gateway.

Parameter	Deskripsi
`$context.connectionId`	ID unik untuk koneksi yang dapat digunakan untuk membuat callback ke klien.
`$context.connectedAt`	Parameterjangka waktuwaktu koneksi -diformat .
`$context.domainName`	Nama domain untuk API WebSocket. Ini dapat digunakan untuk membuat callback ke klien (bukan nilai hard-code).
`$context.eventType`	Jenis peristiwa:`CONNECT`,`MESSAGE`, atau`DISCONNECT` .

Parameter	Deskripsi
`$context.messageId`	ID sisi server yang unik untuk pesan. Hanya tersedia bila`$context.eventType` adalah`MESSAGE`.
`$context.routeKey`	Kunci rute yang dipilih.
`$context.requestId`	Sama seperti`$context.extendedRequestId` .
`$context.extendedRequestId`	ID yang dihasilkan secara otomatis untuk panggilan API, yang berisi informasi yang lebih berguna untuk debugging/pemecahan masalah.
`$context.apiId`	Pengenal API Gateway memberikan API Anda.
`$context.authorizer.principalId`	Identifikasi pengguna utama yang terkait dengan token yang dikirim oleh klien dan dikembalikan dari otorisasi API Gateway Lambda (sebelumnya dikenal sebagai otorisasi khusus) fungsi Lambda.

Parameter	Deskripsi

`$context.authorizer.` *`property`*	Nilai stringified dari pasangan nilai kunci yang ditentukan dari`context`peta kembali dari fungsi API Gateway Lambda authorize r. Misalnya, jika authorizer mengembalikan berikut`context`peta:
	`"context" : {`
	`"key": "value",`
	`"numKey": 1,`
	`"boolKey": true`
	`}`
	panggilan`$context.authorize r.key` mengembalikan`"value"`string, panggilan`$context.authorize r.numKey` mengembalikan`"1"`string, dan panggilan`$context.authorize r.boolKey` mengembalikan`"true"`string.

`$context.error.messageString`	Nilai dikutip`$context.error.message` , yaitu`"$context.error.message"` .
`$context.error.validationEr rorString`	Sebuah string yang berisi pesan kesalahan validasi rinci.
`$context.identity.accountId`	ParameterAWSID akun yang terkait dengan permintaan.
`$context.identity.apiKey`	Kunci pemilik API yang terkait dengan permintaan API berkemampuan kunci.
`$context.identity.apiKeyId`	ID kunci API yang terkait dengan permintaan API berkemampuan kunci
`$context.identity.caller`	Pengenal utama pemanggil membuat permintaan.

Parameter	Deskripsi
`$context.identity.cognitoAuthenticationProvider`	Daftar penyedia autentikasi Amazon Cognito yang digunakan oleh pemanggil yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredensi Amazon Cognito. Misalnya, untuk identitas dari kumpulan pengguna Amazon Cognito,`cognito-idp.` *`region`*`.amazonaws.com/` *`user_pool_id`* `,cognito-idp.` *`region`*`.amazonaws.com/` *`user_pool_id`* `:CognitoSignIn:` *`token subject claim`* Untuk informasi, lihatMenggunakan Identitas GabungandiPanduan Developer Amazon Cognito.
`$context.identity.cognitoAuthenticationType`	Jenis otentikasi Amazon Cognito dari pemanggil yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredensi Amazon Cognito. Nilai yang mungkin termasuk`authenticated` untuk identitas yang diautentikasi dan`unauthenticated` untuk identitas yang tidak diautentikasi.
`$context.identity.cognitoIdentityId`	ID identitas Amazon Cognito pemanggil yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredensi Amazon Cognito.
`$context.identity.cognitoIdentityPoolId`	ID kumpulan identitas Amazon Cognito dari pemanggil yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredensi Amazon Cognito.

Parameter	Deskripsi
`$context.identity.sourceIp`	Alamat IP sumber dari koneksi TCP langsung membuat permintaan ke API Gateway endpoint.
`$context.identity.user`	Pengenal utama pengguna membuat permintaan.
`$context.identity.userAgent`	Agen Pengguna pemanggil API.
`$context.identity.userArn`	Amazon Resource Name (ARN) dari pengguna yang efektif diidentifikasi setelah autentikasi.
`$context.requestTime`	ParameterCLFwaktu permintaan -diformat (`dd/MMM/yyyy:HH:mm:ss +-hhmm`).
`$context.requestTimeEpoch`	Parameterjangka waktuwaktu permintaan -diformat, dalam milidetik.
`$context.stage`	Tahap penyebaran panggilan API (misalnya, Beta atau Prod).
`$context.status`	Status respon.
`$input.body`	Mengembalikan payload mentah sebagai string.
`$input.json(x)`	Fungsi ini mengevaluasi ekspresi JSONPath dan mengembalikan hasil sebagai string JSON. Misalnya,`$input.json('$.pets')` akan mengembalikan string JSON yang mewakili struktur hewan peliharaan. Untuk informasi lebih lanjut tentang JSONPath, lihatJSONPathatauJSONPath untuk Java.

Parameter	Deskripsi
`$input.path(x)`	Membawa string ekspresi JSONPath (x) dan mengembalikan representasi objek JSON hasil. Hal ini memungkinkan Anda untuk mengakses dan memanipulasi elemen muatan native di[Apache Kecepatan Template Bahasa (VTL)](). Misalnya, jika ekspresi`$input.path('$.pets')` mengembalikan sebuah objek seperti ini: `[ { "id": 1, "type": "dog", "price": 249.99 }, { "id": 2, "type": "cat", "price": 124.99 }, { "id": 3, "type": "fish", "price": 0.99 } ]` `$input.path('$.pets').count()` akan kembali`"3"`. Untuk informasi lebih lanjut tentang JSONPath, lihat[JSONPath]()atau[JSONPath untuk Java]().
`$stageVariables.` *`<variable_name>`*	*`<variable_name>`* merupakan nama variabel tahap.

Parameter	Deskripsi
`$stageVariables['` *`<variable_name>`* `']`	*`<variable_name>`* mewakili nama variabel tahap apa pun.
`${stageVariables['` *`<variable_name>`* `']}`	*`<variable_name>`* mewakili nama variabel tahap apa pun.
`$util.escapeJavaScript()`	Melarikan diri karakter dalam string menggunakan aturan string JavaScript. **Note** Fungsi ini akan mengubah setiap tanda kutip tunggal biasa (`'`) menjadi yang melarikan diri (`\'`). Namun, tanda kutip tunggal yang lolos tidak berlaku di JSON. Jadi, ketika output dari fungsi ini digunakan dalam properti JSON, Anda harus mengubah setiap tanda kutip tunggal melarikan diri (`\'`) kembali ke tanda kutip tunggal reguler (`'`). Ini seperti yang ditunjukkan dalam contoh berikut: `$util.escapeJavaScript( data).replaceAll("\\'",'')`

Parameter	Deskripsi

`$util.parseJson()`	Membawa “stringified” JSON dan mengembal ikan representasi objek dari hasilnya. Anda dapat menggunakan hasil dari fungsi ini untuk mengakses dan memanipulasi elemen payload native di Apache Velocity Template Language (VTL). Misalnya, jika Anda memiliki muatan berikut:
	`{"errorMessage":"{\"key1\":\"var1\", \"key2\":{\"arr\":[1,2,3]}}"}`
	dan gunakan template pemetaan berikut
	`#set ($errorMessageObj = $util.par seJson($input.path('$.error Message')))`
	`{`
	`"errorMessageObjKey2ArrVal" : $errorMessageObj.key2.arr[0]`
	`}`
	Anda akan mendapatkan output sebagai berikut:
	`{`
	`"errorMessageObjKey2ArrVal" : 1`
	`}`

`$util.urlEncode()`	Mengkonversi string menjadi format “applicat ion/x-www-form-urlencoded”.
`$util.urlDecode()`	Decode string “application/x-form-urlencoded”.
`$util.base64Encode()`	Mengkodekan data menjadi string base64-en coded.
`$util.base64Decode()`	Decode data dari string base64-encoded.

Bekerja dengan tipe media biner untuk API WebSocket

API Gateway WebSocket API saat ini tidak mendukung frame biner dalam muatan pesan masuk. Jika aplikasi klien mengirimkan bingkai biner, API Gateway menolaknya dan memutus klien dengan kode 1003.

Ada solusi untuk perilaku ini. Jika klien mengirimkan data biner yang dikodekan teks (misalnya, base64) sebagai bingkai teks, Anda dapat mengatur integrasi`contentHandlingStrategy`properti`CONVERT_TO_BINARY`untuk mengkonversi payload dari string base64-encoded untuk biner.

Untuk mengembalikan respons rute untuk payload biner dalam integrasi non-proxy, Anda dapat mengatur respons integrasi`contentHandlingStrategy`properti`CONVERT_TO_TEXT`untuk mengkonversi payload dari string biner untuk base64-encoded.

Memanggil API WebSocket

Setelah menerapkan WebSocket API, aplikasi klien dapat terhubung dan mengirim pesan ke API itu—dan layanan backend Anda dapat mengirim pesan ke aplikasi klien yang terhubung:

- Anda dapat menggunakan `wscat` untuk menyambung ke WebSocket API Anda dan mengirim pesan ke sana untuk mensimulasikan perilaku klien. Lihat the section called “Gunakan `wscat` untuk terhubung ke WebSocket API dan mengirim pesan ke sana”.
- Anda dapat menggunakan @connections API dari layanan backend Anda untuk mengirim pesan callback ke klien yang terhubung, mendapatkan informasi koneksi, atau memutuskan sambungan klien. Lihat the section called “Gunakan `@connections` perintah di layanan backend Anda”.
- Aplikasi klien dapat menggunakan WebSocket pustaka sendiri untuk menjalankan WebSocket API Anda.

Gunakan **`wscat`** untuk terhubung ke WebSocket API dan mengirim pesan ke sana

`wscat`Utilitas adalah alat yang nyaman untuk menguji WebSocket API yang telah Anda buat dan gunakan di API Gateway. Anda dapat menginstal dan menggunakan `wscat` sebagai berikut:

1. Unduh `wscat` dari https://www.npmjs.com/package/wscat.
2. Instal `wscat` dengan menjalankan perintah berikut:

```
npm install -g wscat
```

3. Untuk terhubung ke API Anda, jalankan `wscat` perintah seperti yang ditunjukkan pada contoh berikut. Perhatikan bahwa contoh ini mengasumsikan bahwa `Authorization` pengaturannya adalah`NONE`.

   ```
   wscat -c wss://aabbccddee.execute-api.us-east-1.amazonaws.com/test/
   ```

   Anda perlu mengganti *`aabbccddee`* dengan ID API yang sebenarnya, yang ditampilkan di konsol API Gateway atau dikembalikan oleh AWS CLI `create-api`perintah.

   Selain itu, jika API Anda berada di Wilayah selain`us-east-1`, Anda perlu mengganti Region yang benar.

4. Untuk menguji API Anda, masukkan pesan seperti berikut saat terhubung:

   ```
   {"{jsonpath-expression}":"{route-key}"}
   ```

   di mana *`{jsonpath-expression}`* adalah ekspresi JsonPath dan *`{route-key} adalah kunci`* rute untuk API. Sebagai contoh:

   ```
   {"action":"action1"}
   {"message":"test response body"}
   ```

   Untuk informasi selengkapnya tentang JsonPath, lihat JsonPath atau JsonPath for Java.

5. Untuk memutuskan sambungan dari API Anda, masukkan`ctrl-C`.

Gunakan **`@connections`** perintah di layanan backend Anda

Layanan backend Anda dapat menggunakan permintaan HTTP WebSocket koneksi berikut untuk mengirim pesan callback ke klien yang terhubung, mendapatkan informasi koneksi, atau memutuskan sambungan klien.

> **Important**
>
> Permintaan ini menggunakan otorisasi IAM, jadi Anda harus menandatanganinya dengan Sigv4 (SigV4). Untuk melakukan ini, Anda dapat menggunakan API Gateway Management API. Untuk informasi lebih lanjut, lihat ApiGatewayManagementApi.

Dalam perintah berikut, Anda perlu mengganti *{api-id}* dengan ID API yang sebenarnya, yang ditampilkan di konsol API Gateway atau dikembalikan oleh AWS CLI `create-api`perintah. Anda harus membuat koneksi sebelum menggunakan perintah ini.

Untuk mengirim pesan callback ke klien, gunakan:

```
POST https://{api-id}.execute-api.us-east-1.amazonaws.com/{stage}/
@connections/{connection_id}
```

Anda dapat menguji permintaan ini dengan menggunakan `Postman` atau dengan memanggil `awscurl` seperti pada contoh berikut:

```
awscurl --service execute-api -X POST -d "hello world" https://{prefix}.execute-api.us-
east-1.amazonaws.com/{stage}/@connections/{connection_id}
```

Anda perlu mengkodekan URL perintah seperti pada contoh berikut:

```
awscurl --service execute-api -X POST -d "hello world" https://aabbccddee.execute-
api.us-east-1.amazonaws.com/prod/%40connections/R0oXAdfD0kwCH6w%3D
```

Untuk mendapatkan status koneksi terbaru dari klien, gunakan:

```
GET https://{api-id}.execute-api.us-east-1.amazonaws.com/{stage}/
@connections/{connection_id}
```

Untuk memutuskan sambungan klien, gunakan:

```
DELETE https://{api-id}.execute-api.us-east-1.amazonaws.com/{stage}/
@connections/{connection_id}
```

Anda dapat membuat URL callback secara dinamis dengan menggunakan `$context` variabel dalam integrasi Anda. Misalnya, jika Anda menggunakan integrasi proxy Lambda dengan fungsi `Node.js` Lambda, Anda dapat membuat URL dan mengirim pesan ke klien yang terhubung sebagai berikut:

```
import {
  ApiGatewayManagementApiClient,
  PostToConnectionCommand,
} from "@aws-sdk/client-apigatewaymanagementapi";
```

```
export const handler = async (event) => {
  const domain = event.requestContext.domainName;
  const stage = event.requestContext.stage;
  const connectionId = event.requestContext.connectionId;
  const callbackUrl = `https://${domain}/${stage}`;
  const client = new ApiGatewayManagementApiClient({ endpoint: callbackUrl });

  const requestParams = {
    ConnectionId: connectionId,
    Data: "Hello!",
  };

  const command = new PostToConnectionCommand(requestParams);

  try {
    await client.send(command);
  } catch (error) {
    console.log(error);
  }

  return {
    statusCode: 200,
  };
};
```

Saat mengirim pesan panggilan balik, fungsi Lambda Anda harus memiliki izin untuk memanggil API Gateway Management API. Anda mungkin menerima kesalahan yang berisi `GoneException` jika Anda memposting pesan sebelum koneksi dibuat, atau setelah klien terputus.

Menerbitkan WebSocket API bagi pelanggan untuk dipanggil

Cukup membuat dan mengembangkan API Gateway API tidak secara otomatis membuatnya dapat dipanggil oleh pengguna Anda. Untuk membuatnya dapat dipanggil, Anda harus menerapkan API Anda ke sebuah panggung. Selain itu, Anda mungkin ingin menyesuaikan URL yang akan digunakan pengguna Anda untuk mengakses API Anda. Anda dapat memberikan domain yang konsisten dengan merek Anda atau lebih mudah diingat daripada URL default untuk API Anda.

Di bagian ini, Anda dapat mempelajari cara menerapkan API dan menyesuaikan URL yang Anda berikan kepada pengguna untuk mengaksesnya.

> **Note**
>
> Untuk meningkatkan keamanan API Gateway API Anda, `execute-api.{region}.amazonaws.com` domain tersebut terdaftar di Daftar Akhiran Publik (PSL). Untuk keamanan lebih lanjut, kami menyarankan Anda menggunakan cookie dengan `__Host-` awalan jika Anda perlu mengatur cookie sensitif di nama domain default untuk API Gateway API Anda. Praktik ini akan membantu mempertahankan domain Anda dari upaya pemalsuan permintaan lintas situs (CSRF). Untuk informasi selengkapnya, lihat halaman Set-Cookie di Jaringan Pengembang Mozilla.

Topik

Bekerja dengan tahapan API WebSocket

Tahap API adalah referensi logis ke status siklus hidup API Anda (misalnya,`dev`,`prod`,`beta`, atau`v2`). Tahapan API diidentifikasi oleh ID API dan nama stage, dan mereka disertakan dalam URL yang Anda gunakan untuk memanggil API. Setiap tahap adalah referensi bernama untuk deployment API dan dibuat tersedia bagi aplikasi klien untuk dipanggil.

Penyebaran adalah snapshot dari konfigurasi API Anda. Setelah Anda menerapkan API ke panggung, tersedia bagi klien untuk dipanggil. Anda harus menerapkan API agar perubahan diterapkan.

Variabel tahap

Variabel tahap adalah pasangan kunci-nilai yang dapat Anda tentukan untuk tahap API WebSocket. Mereka bertindak seperti variabel lingkungan dan dapat digunakan dalam pengaturan API Anda.

Misalnya, Anda dapat menentukan variabel stage, dan kemudian menetapkan nilainya sebagai endpoint HTTP untuk integrasi proxy HTTP. Kemudian, Anda dapat mereferensikan endpoint dengan menggunakan nama variabel tahap terkait. Dengan melakukan ini, Anda dapat menggunakan

pengaturan API yang sama dengan endpoint yang berbeda pada setiap tahap. Demikian pula, Anda dapat menggunakan variabel stage untuk menentukan yang berbedaAWS Lambdaintegrasi fungsi untuk setiap tahap API Anda.

> **Note**
>
> Variabel tahap tidak dimaksudkan untuk digunakan untuk data sensitif, seperti kredensi. Untuk meneruskan data sensitif ke integrasi, gunakanAWS Lambdapemberi kuasa. Anda dapat meneruskan data sensitif ke integrasi dalam output dari otorisasi Lambda. Untuk mempelajari selengkapnya, lihat the section called “Format respons otorisasi Lambda”.

Contoh

Untuk menggunakan variabel stage untuk menyesuaikan titik akhir integrasi HTTP, Anda harus terlebih dahulu mengatur nama dan nilai variabel stage (misalnya,`url`) dengan nilai`example.com`. Selanjutnya, siapkan integrasi proxy HTTP. Alih-alih memasukkan URL endpoint, Anda dapat memberi tahu API Gateway untuk menggunakan nilai variabel stage,**`http://${stageVariables.url}`**. Nilai ini memberi tahu API Gateway untuk menggantikan variabel stage Anda`${}`saat runtime, tergantung pada tahap API Anda.

Anda dapat mereferensikan variabel tahap dengan cara yang sama untuk menentukan nama fungsi Lambda atauAWSARN peran.

Ketika menentukan nama fungsi Lambda sebagai nilai variabel stage, Anda harus mengkonfigurasi izin pada fungsi Lambda secara manual. Anda dapat menggunakanAWS Command Line Interface(AWS CLI) untuk melakukan ini.

```
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
```

Referensi variabel tahap API Gateway

URI integrasi HTTP

Anda dapat menggunakan variabel tahap sebagai bagian dari URI integrasi HTTP, seperti yang ditampilkan dalam contoh berikut.

- URI lengkap tanpa protokol —`http://${stageVariables.<variable_name>}`
- Domain lengkap —`http://${stageVariables.<variable_name>}/resource/operation`
- Subdomain —`http://${stageVariables.<variable_name>}.example.com/resource/operation`
- Jalur —`http://example.com/${stageVariables.<variable_name>}/bar`
- String kueri —`http://example.com/foo?q=${stageVariables.<variable_name>}`

Fungsi Lambda

Anda dapat menggunakan variabel tahap di tempat nama fungsi Lambda atau alias, seperti yang ditampilkan dalam contoh berikut.

- `arn:aws:apigateway:<region>:lambda:path/2015-03-31/functions/arn:aws:lambda:<region>:<account_id>:function:${stageVariables.<function_variable_name>}/invocations`
- `arn:aws:apigateway:<region>:lambda:path/2015-03-31/functions/arn:aws:lambda:<region>:<account_id>:function:<function_name>:${stageVariables.<version_variable_name>}/invocations`

> **Note**
>
> Untuk menggunakan variabel tahap untuk fungsi Lambda, fungsi harus berada di akun yang sama dengan API. Variabel tahap tidak mendukung fungsi Lambda lintas akun.

AWSkredensi integrasi

Anda dapat menggunakan variabel stage sebagai bagian dariAWSARN kredensi pengguna atau peran, seperti yang ditampilkan dalam contoh berikut.

- `arn:aws:iam::<account_id>:${stageVariables.<variable_name>}`

Menerapkan WebSocket API di API Gateway

Setelah membuat WebSocket API Anda, Anda harus menerapkannya agar tersedia bagi pengguna Anda untuk dipanggil.

Untuk menerapkan API, Anda membuat penerapan API dan mengaitkannya dengan sebuah panggung. Setiap tahap adalah snapshot dari API dan tersedia untuk aplikasi klien untuk dipanggil.

> **Important**
>
> Setiap kali Anda memperbarui API, Anda harus menerapkannya kembali. Perubahan pada apa pun selain pengaturan tahap memerlukan pemindahan, termasuk modifikasi pada sumber daya berikut:
>
> - Rute
> - Integrasi
> - Pengotorisasi
>
> Secara default, Anda dibatasi hingga 10 tahap untuk setiap API. Kami menyarankan Anda menggunakan kembali tahapan untuk penerapan Anda.

Untuk memanggil WebSocket API yang diterapkan, klien mengirim pesan ke URL API. URL ditentukan oleh nama host dan nama panggung API.

> **Note**
>
> API Gateway akan mendukung payload hingga 128 KB dengan ukuran frame maksimal 32 KB. Jika pesan melebihi 32 KB, itu harus dibagi menjadi beberapa frame, masing-masing 32 KB atau lebih kecil.

Menggunakan nama domain default API, URL (misalnya) WebSocket API dalam tahap tertentu (*`{stageName}`*) adalah dalam format berikut:

```
wss://{api-id}.execute-api.{region}.amazonaws.com/{stageName}
```

Untuk membuat URL WebSocket API lebih user-friendly, Anda dapat membuat nama domain kustom (misalnya,`api.example.com`) untuk mengganti nama host default API. Proses konfigurasi sama dengan REST API. Untuk informasi selengkapnya, lihat the section called “Nama domain kustom”.

Tahapan memungkinkan kontrol versi yang kuat dari API Anda. Misalnya, Anda dapat menerapkan API ke `test` stage dan `prod` stage, dan menggunakan `test` stage sebagai test build dan

menggunakan `prod` stage sebagai build stabil. Setelah pembaruan lulus tes, Anda dapat mempromosikan `test` panggung ke `prod` panggung. Promosi dapat dilakukan dengan memindahkan API ke panggung. `prod` Untuk detail lebih lanjut tentang tahapan, lihatthe section called “Siapkan panggung”.

Topik

- Membuat penerapan WebSocket API menggunakan AWS CLI
- Membuat penerapan WebSocket API menggunakan konsol API Gateway

Membuat penerapan WebSocket API menggunakan AWS CLI

Untuk digunakan AWS CLI untuk membuat penyebaran, gunakan `create-deployment`perintah seperti yang ditunjukkan dalam contoh berikut:

```
aws apigatewayv2 --region us-east-1 create-deployment --api-id aabbccddee
```

Contoh keluaran:

```
{
    "DeploymentId": "fedcba",
    "DeploymentStatus": "DEPLOYED",
    "CreatedDate": "2018-11-15T06:49:09Z"
}
```

API yang diterapkan tidak dapat dipanggil sampai Anda mengaitkan penerapan dengan tahapan. Anda dapat membuat tahap baru atau menggunakan kembali tahap yang telah Anda buat sebelumnya.

Untuk membuat tahap baru dan mengaitkannya dengan penyebaran, gunakan `create-stage`perintah seperti yang ditunjukkan pada contoh berikut:

```
aws apigatewayv2 --region us-east-1 create-stage --api-id aabbccddee --deployment-id
 fedcba --stage-name test
```

Contoh keluaran:

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

Untuk menggunakan kembali tahap yang ada, perbarui `deploymentId` properti stage dengan ID penerapan (*`{deployment-id}`*) yang baru dibuat dengan menggunakan perintah. `update-stage`

```
aws apigatewayv2 update-stage --region {region} \
   --api-id {api-id} \
   --stage-name {stage-name} \
   --deployment-id {deployment-id}
```

Membuat penerapan WebSocket API menggunakan konsol API Gateway

Untuk menggunakan konsol API Gateway untuk membuat penerapan WebSocket API:

1. Masuk ke konsol API Gateway dan pilih API.
2. Pilih Deploy API.
3. Pilih tahap yang diinginkan dari daftar dropdown atau masukkan nama tahap baru.

Kebijakan keamanan untuk WebSocket API

API Gateway memberlakukan kebijakan keamanan `TLS_1_2` untuk semua titik akhir WebSocket API.

Kebijakan keamanan adalah kombinasi standar dari versi TLS minimum dan cipher suite yang ditawarkan oleh Amazon API Gateway. Protokol TLS mengatasi masalah keamanan jaringan seperti gangguan dan penyadapan antara klien dan server. Ketika klien Anda membuat jabat tangan TLS ke API Anda melalui domain kustom, kebijakan keamanan memberlakukan versi TLS dan pilihan cipher suite yang dapat dipilih klien Anda untuk digunakan. Kebijakan keamanan ini menerima lalu lintas TLS 1.2 dan TLS 1.3 dan menolak lalu lintas TLS 1.0.

Protokol dan cipher TLS yang didukung untuk API WebSocket

Tabel berikut menjelaskan protokol dan cipher TLS yang didukung untuk API. WebSocket

Kebijakan keamanan	TLS_1_2
Protokol TLS	
TLSv1.3	♦
TLSv1.2	♦
Cipher TLS	
TLS_AES_128_GCM_SHA256	♦
TLS_AES_256_GCM_SHA384	♦
TLS_CHACHA20_POLY1305_SHA256	♦
ECDHE-ECDSA-AES128-GCM-SHA256	♦
ECDHE-RSA-AES128-GCM-SHA256	♦
ECDHE-ECDSA-AES128-SHA256	♦
ECDHE-RSA-AES128-SHA256	♦
ECDHE-ECDSA-AES256-GCM-SHA384	♦
ECDHE-RSA-AES256-GCM-SHA384	♦
ECDHE-ECDSA-AES256-SHA384	♦
ECDHE-RSA-AES256-SHA384	♦
AES128-GCM-SHA256	♦
AES128-SHA256	♦
AES256-GCM-SHA384	♦
AES256-SHA256	♦

Nama sandi OpenSSL dan RFC

OpenSSL dan IETF RFC 5246, menggunakan nama yang berbeda untuk cipher yang sama. Untuk daftar nama sandi, lihat. the section called “Nama sandi OpenSSL dan RFC”

Informasi tentang REST API dan HTTP API

Untuk informasi selengkapnya tentang REST API dan HTTP API, lihat the section called “Memilih kebijakan keamanan” danthe section called “Kebijakan keamanan untuk HTTP API”.

Menyiapkan nama domain khusus untuk WebSocket API

Nama domain khusus adalah URL yang lebih sederhana dan lebih intuitif yang dapat Anda berikan kepada pengguna API Anda.

Setelah menerapkan API, Anda (dan pelanggan) dapat menjalankan API menggunakan URL dasar default dari format berikut:

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

dimana `api-id` dihasilkan oleh API Gateway, `region` (AWS Wilayah) ditentukan oleh Anda saat membuat API, dan `stage` ditentukan oleh Anda saat menerapkan API.

Bagian nama host dari URL (yaitu,*`api-id`*`.execute-api.`*`region`*`.amazonaws.com`) mengacu pada titik akhir API. Titik akhir API default bisa sulit diingat dan tidak ramah pengguna.

Dengan nama domain khusus, Anda dapat mengatur nama host API Anda, dan memilih jalur dasar (misalnya,`myservice`) untuk memetakan URL alternatif ke API Anda. Misalnya, URL dasar API yang lebih ramah pengguna dapat menjadi:

```
https://api.example.com/myservice
```

> **Note**
>
> Nama domain khusus untuk WebSocket API tidak dapat dipetakan ke REST API atau HTTP API.
> Untuk WebSocket API, nama domain kustom Regional didukung.
> Untuk WebSocket API, TLS 1.2 adalah satu-satunya versi TLS yang didukung.

Daftarkan nama domain

Anda harus memiliki nama domain internet terdaftar untuk menyiapkan nama domain khusus untuk API Anda. Jika diperlukan, Anda dapat mendaftarkan domain internet menggunakan Amazon Route 53 atau menggunakan registrar domain pihak ketiga pilihan Anda. Nama domain kustom API dapat berupa nama subdomain atau domain root (juga dikenal sebagai “zone apex”) dari domain internet terdaftar.

Setelah nama domain kustom dibuat di API Gateway, Anda harus membuat atau memperbarui catatan sumber daya penyedia DNS Anda untuk dipetakan ke titik akhir API Anda. Tanpa pemetaan seperti itu, permintaan API yang terikat untuk nama domain khusus tidak dapat mencapai API Gateway.

Nama domain kustom regional

Saat Anda membuat nama domain khusus untuk API Regional, API Gateway membuat nama domain Regional untuk API. Anda harus menyiapkan catatan DNS untuk memetakan nama domain kustom ke nama domain Regional. Anda juga harus memberikan sertifikat untuk nama domain kustom.

Nama domain kustom wildcard

Dengan nama domain khusus wildcard, Anda dapat mendukung jumlah nama domain yang hampir tak terbatas tanpa melebihi kuota default. Misalnya, Anda bisa memberi setiap pelanggan Anda nama domain mereka sendiri*customername*`.api.example.com`.

Untuk membuat nama domain kustom wildcard, tentukan wildcard (`*`) sebagai subdomain pertama dari domain kustom yang mewakili semua kemungkinan subdomain dari domain root.

Misalnya, nama domain kustom wildcard `*.example.com` menghasilkan subdomain sepertia`.example.com`,, dan `b.example.comc.example.com`, yang semuanya merutekan ke domain yang sama.

Nama domain kustom wildcard mendukung konfigurasi yang berbeda dari nama domain kustom standar API Gateway. Misalnya, dalam satu AWS akun, Anda dapat mengkonfigurasi `*.example.com` dan `a.example.com` berperilaku berbeda.

Anda dapat menggunakan variabel `$context.domainName` dan `$context.domainPrefix` konteks untuk menentukan nama domain yang digunakan klien untuk memanggil API Anda. Untuk mempelajari lebih lanjut tentang variabel konteks, lihatTemplate pemetaan API Gateway dan referensi variabel pencatatan akses.

Untuk membuat nama domain kustom wildcard, Anda harus memberikan sertifikat yang dikeluarkan oleh ACM yang telah divalidasi menggunakan DNS atau metode validasi email.

> **Note**
>
> Anda tidak dapat membuat nama domain khusus wildcard jika AWS akun lain telah membuat nama domain kustom yang bertentangan dengan nama domain kustom wildcard. Misalnya, jika akun A telah dibuata `.example.com`, maka akun B tidak dapat membuat nama `*.example.com` domain khusus wildcard.
> Jika akun A dan akun B berbagi pemilik, Anda dapat menghubungi Pusat AWS Dukungan untuk meminta pengecualian.

Sertifikat untuk nama domain kustom

> **Important**
>
> Anda menentukan sertifikat untuk nama domain kustom Anda. Jika aplikasi Anda menggunakan pinning sertifikat, kadang-kadang dikenal sebagai penyematan SSL, untuk menyematkan sertifikat ACM, aplikasi mungkin tidak dapat terhubung ke domain Anda setelah AWS memperbarui sertifikat. Untuk informasi selengkapnya, lihat Masalah penyematan sertifikat di Panduan AWS Certificate Manager Pengguna.

Untuk memberikan sertifikat untuk nama domain kustom di Wilayah di mana ACM didukung, Anda harus meminta sertifikat dari ACM. Untuk memberikan sertifikat untuk nama domain kustom Regional di Wilayah di mana ACM tidak didukung, Anda harus mengimpor sertifikat ke API Gateway di Wilayah tersebut.

Untuk mengimpor sertifikat SSL/TLS, Anda harus menyediakan badan sertifikat SSL/TLS yang diformat PEM, kunci pribadinya, dan rantai sertifikat untuk nama domain kustom. Setiap sertifikat yang disimpan dalam ACM diidentifikasi oleh ARN-nya. Untuk menggunakan sertifikat AWS terkelola untuk nama domain, Anda cukup mereferensikan ARN-nya.

ACM membuatnya mudah untuk mengatur dan menggunakan nama domain khusus untuk API. Anda membuat sertifikat untuk nama domain yang diberikan (atau mengimpor sertifikat), menyiapkan nama domain di API Gateway dengan ARN sertifikat yang disediakan oleh ACM, dan memetakan jalur

dasar di bawah nama domain kustom ke tahap penerapan API. Dengan sertifikat yang dikeluarkan oleh ACM, Anda tidak perlu khawatir mengekspos detail sertifikat sensitif, seperti kunci pribadi.

Siapkan nama domain khusus

Untuk detail tentang menyiapkan nama domain kustom, lihat Mendapatkan sertifikat siap diAWS Certificate Manager danMenyiapkan nama domain kustom regional di API Gateway.

Bekerja dengan pemetaan API untuk API WebSocket

Anda menggunakan pemetaan API untuk menghubungkan tahapan API ke nama domain khusus. Setelah membuat nama domain dan mengonfigurasi data DNS, Anda menggunakan pemetaan API untuk mengirim traffic ke API melalui nama domain khusus.

Pemetaan API menentukan API, stage, dan opsional path yang digunakan untuk pemetaan. Misalnya, Anda dapat memetakan`production`tahap API ke`wss://api.example.com/orders`.

Sebelum membuat pemetaan API, Anda harus memiliki API, tahap, dan nama domain khusus. Untuk mempelajari lebih lanjut tentang membuat nama domain kustom, lihatthe section called “Menyiapkan nama domain kustom regional”.

Pembatasan

- Dalam pemetaan API, nama domain khusus dan API yang dipetakan harus samaAWSakun.
- Pemetaan API hanya berisi huruf, angka, dan karakter berikut:`$-_.+!*'()`.
- Panjang maksimum jalur dalam pemetaan API adalah 300 karakter.
- Anda tidak dapat memetakan API WebSocket ke nama domain kustom yang sama dengan API HTTP atau REST API.

Membuat pemetaan API

Untuk membuat pemetaan API, Anda harus terlebih dahulu membuat nama domain kustom, API, dan stage. Untuk informasi tentang membuat nama domain kustom, lihatthe section called “Menyiapkan nama domain kustom regional”.

AWS Management Console

Untuk membuat pemetaan API

1. Masuk ke konsol API Gatewayhttps://console.aws.amazon.com/apigateway.

2. PilihNama domain kustom.
3. Pilih nama domain khusus yang telah Anda buat.
4. PilihPemetaan API.
5. PilihMengkonfigurasi pemetaan API.
6. PilihTambahkan pemetaan baru.
7. MasukkanAPI, aTahap, dan opsionalJalur.
8. Pilih Save (Simpan).

AWS CLI

BerikutAWS CLIperintah menciptakan pemetaan API. Dalam contoh ini, API Gateway mengirimkan permintaan ke`api.example.com/v1`ke API dan tahap yang ditentukan.

```
aws apigatewayv2 create-api-mapping \
    --domain-name api.example.com \
    --api-mapping-key v1 \
    --api-id a1b2c3d4 \
    --stage test
```

AWS CloudFormation

BerikutAWS CloudFormationcontoh menciptakan pemetaan API.

```
MyApiMapping:
  Type: 'AWS::ApiGatewayV2::ApiMapping'
  Properties:
    DomainName: api.example.com
    ApiMappingKey: 'v1'
    ApiId: !Ref MyApi
    Stage: !Ref MyStage
```

Menonaktifkan endpoint default untuk API WebSocket

Secara default, klien dapat memanggil API dengan menggunakan`execute-api`endpoint yang dihasilkan API Gateway untuk API Anda. Untuk memastikan bahwa klien dapat mengakses API Anda hanya dengan menggunakan nama domain kustom, nonaktifkan default`execute-api`titik akhir.

> **Note**
>
> Ketika Anda menonaktifkan endpoint default, itu mempengaruhi semua tahapan API.

BerikutAWS CLIperintah menonaktifkan titik akhir default untuk API WebSocket.

```
aws apigatewayv2 update-api \
    --api-id abcdef123 \
    --disable-execute-api-endpoint
```

Setelah titik akhir default, Anda harus menerapkan API agar perubahan dapat diterapkan.

BerikutAWS CLIperintah menciptakan deployment.

```
aws apigatewayv2 create-deployment \
    --api-id abcdef123 \
    --stage-name dev
```

Melindungi WebSocket API Anda

Anda dapat mengkonfigurasi throttling untuk API Anda untuk membantu melindungi mereka dari kewalahan oleh terlalu banyak permintaan. Throttle diterapkan berdasarkan upaya terbaik dan harus dianggap sebagai target daripada langit-langit permintaan yang dijamin.

API Gateway membatasi permintaan ke API Anda menggunakan algoritma token bucket, di mana token dihitung untuk permintaan. Secara khusus, API Gateway memeriksa tarif dan ledakan pengiriman permintaan terhadap semua API di akun Anda, per Wilayah. Dalam algoritma token bucket, burst dapat memungkinkan overrun yang telah ditentukan sebelumnya dari batas-batas tersebut, tetapi faktor lain juga dapat menyebabkan batasan untuk dikuasai dalam beberapa kasus.

Ketika pengiriman permintaan melebihi tingkat permintaan steady-state dan batas burst, API Gateway mulai membatasi permintaan. Klien mungkin menerima respons`429 Too Many Requests` kesalahan pada saat ini. Setelah menangkap pengecualian tersebut, klien dapat mengirimkan kembali permintaan yang gagal dengan cara yang membatasi tarif.

Sebagai pengembang API, Anda dapat menetapkan batas target untuk setiap tahapan API atau rute untuk meningkatkan kinerja keseluruhan di semua API di akun Anda.

Pelambatan tingkat akun per Wilayah

Secara default, API Gateway membatasi permintaan steady-state per detik (RPS) di semua API dalamAWS akun, per Wilayah. Ini juga membatasi burst (yaitu, ukuran bucket maksimum) di semua API dalamAWS akun, per Wilayah. Di API Gateway, batas burst mewakili jumlah maksimum target kiriman permintaan bersamaan yang akan dipenuhi API Gateway sebelum mengembalikan respons`429 Too Many Requests` kesalahan. Untuk informasi lebih lanjut tentang kuota pembatasan, lihatKuota dan catatan penting.

Batas per akun diterapkan ke semua API di akun di Wilayah tertentu. Batas tingkat tingkat akun dapat ditingkatkan berdasarkan permintaan - batas yang lebih tinggi dimungkinkan dengan API yang memiliki waktu tunggu lebih pendek dan muatan yang lebih kecil. Untuk meminta peningkatan batas pembatasan tingkat akun per Wilayah, hubungi PusatAWS Support. Untuk informasi selengkapnya, lihat Kuota dan catatan penting. Perhatikan bahwa batas ini tidak boleh lebih tinggi dari batasAWS pembatasan.

Pelambatan tingkat rute

Anda dapat mengatur pembatasan tingkat rute untuk mengesampingkan batas pembatasan permintaan tingkat akun untuk tahap tertentu atau untuk rute individual di API Anda. Batas pembatasan rute default tidak dapat melebihi batas tingkat akun.

Anda dapat mengonfigurasi pelambatan tingkat rute dengan menggunakanAWS CLI. Perintah berikut mengkonfigurasi pembatasan khusus untuk tahap dan rute API yang ditentukan.

```
aws apigatewayv2 update-stage \
    --api-id a1b2c3d4 \
    --stage-name dev \
    --route-settings '{"messages":
{"ThrottlingBurstLimit":100,"ThrottlingRateLimit":2000}}'
```

API WebSocket

Anda dapat menggunakan metrik CloudWatch dan CloudWatch Logs untuk memantau API WebSocket. Dengan menggabungkan log dan metrik, Anda dapat mencatat kesalahan dan memantau kinerja API Anda.

> **Note**
>
> API Gateway mungkin tidak menghasilkan log dan metrik dalam kasus berikut:
>
> - 413 Permintaan Entitas Kesalahan Terlalu Besar
> - Berlebihan 429 Terlalu Banyak Permintaan kesalahan
> - Kesalahan seri 400 dari permintaan yang dikirim ke domain khusus yang tidak memiliki pemetaan API
> - 500 kesalahan seri yang disebabkan oleh kegagalan internal

Topik

Memantau eksekusi WebSocket API dengan CloudWatch metrik

Anda dapat menggunakan CloudWatch metrik Amazon untuk memantau WebSocket API. Konfigurasinya mirip dengan yang digunakan untuk REST API. Untuk informasi selengkapnya, lihat Memantau eksekusi REST API dengan CloudWatch metrik Amazon.

Metrik berikut didukung untuk WebSocket API:

Metrik	Deskripsi
ConnectCount	Jumlah pesan yang dikirim ke integrasi `$connect` rute.
MessageCount	Jumlah pesan yang dikirim ke WebSocket API, baik dari atau ke klien.
IntegrationError	Jumlah permintaan yang mengembalikan respons 4XX/5XX dari integrasi.

Metrik	Deskripsi
ClientError	Jumlah permintaan yang memiliki respons 4XX yang dikembalikan oleh API Gateway sebelum integrasi dipanggil.
ExecutionError	Kesalahan yang terjadi saat memanggil integrasi.
IntegrationLatency	Perbedaan waktu antara API Gateway mengirimk an permintaan ke integrasi dan API Gateway menerima respons dari integrasi. Ditekan untuk panggilan balik dan integrasi tiruan.

Anda dapat menggunakan dimensi dalam tabel berikut untuk memfilter metrik API Gateway.

Dimensi	Deskripsi
ApiId	Memfilter metrik API Gateway untuk API dengan ID API yang ditentukan.
ApiId, Panggung	Memfilter metrik API Gateway untuk tahap API dengan ID API dan ID tahap yang ditentukan.
ApiId, Panggung, Rute	Memfilter metrik API Gateway untuk metode API dengan ID API, ID tahap, dan ID rute yang ditentukan.

Dimensi	Deskripsi
	API Gateway tidak akan mengirimkan metrik ini kecuali Anda telah mengaktifkan metrik terperinci secara eksplisit. CloudWatch Anda dapat melakukan ini dengan memanggil UpdateStageaksi API Gateway V2 REST API untuk memperbarui `detailedMetricsEnabled` properti ke`true`. Atau, Anda dapat memanggil AWS CLI perintah update-stage untuk memperbarui properti ke. `DetailedMetricsEnabled` `true` Mengaktifkan metrik tersebut akan dikenakan biaya tambahan ke akun Anda. Untuk informasi harga, lihat CloudWatchHarga Amazon.

Mengonfigurasi logging untuk API WebSocket

Anda dapat mengaktifkan logging untuk menulis log ke CloudWatch Log. Ada dua jenis log masuk API CloudWatch: logging eksekusi dan logging akses. Dalam pencatatan eksekusi, API Gateway mengelola CloudWatch Log. Prosesnya mencakup pembuatan grup log dan aliran log, dan pelaporan ke aliran log permintaan dan tanggapan pemanggil apa pun.

Dalam pencatatan akses, Anda, sebagai pengembang API, ingin mencatat siapa yang telah mengakses API Anda dan bagaimana pemanggil mengakses API. Anda dapat membuat grup log Anda sendiri atau memilih grup log yang sudah ada yang dapat dikelola oleh API Gateway. Untuk menentukan rincian akses, Anda memilih `$context` variabel (dinyatakan dalam format pilihan Anda) dan memilih grup log sebagai tujuan.

Untuk petunjuk tentang cara mengatur CloudWatch logging, lihatthe section called “Siapkan pencatatan CloudWatch API menggunakan konsol API Gateway”.

Saat Anda menentukan Format Log, Anda dapat memilih variabel konteks mana yang akan dicatat. Variabel berikut didukung.

Parameter	Deskripsi
`$context.apiId`	API Gateway identifier ditetapkan ke API Anda.
`$context.authorize.error`	Pesan kesalahan otorisasi.
`$context.authorize.latency`	Latensi otorisasi di ms.
`$context.authorize.status`	Kode status dikembalikan dari upaya otorisasi.
`$context.authorizer.error`	Pesan kesalahan dikembalikan dari otorisasi.
`$context.authorizer.integrationLatency`	Latensi otorisasi Lambda di ms.
`$context.authorizer.integrationStatus`	Kode status dikembalikan dari otorisasi Lambda.
`$context.authorizer.latency`	Latensi otorisasi di ms.
`$context.authorizer.requestId`	ID permintaan AWS titik akhir.
`$context.authorizer.status`	Kode status dikembalikan dari otorisasi.
`$context.authorizer.principalId`	Identifikasi pengguna utama yang dikaitkan dengan token yang dikirim oleh klien dan dikembalikan dari fungsi Lambda otorisasi API Gateway Lambda. (Authorizer Lambda sebelumnya dikenal sebagai otorisasi khusus.)
`$context.authorizer.` *`property`*	Nilai stringifikasi dari pasangan nilai kunci `context` peta yang ditentukan dikembalikan dari fungsi otorisasi API Gateway Lambda.

Parameter	Deskripsi
	Misalnya, jika otorisasi mengembalikan `context` peta berikut: `"context" : { "key": "value", "numKey": 1, "boolKey": true }` memanggil `$context.authorizer.key` mengembalikan `"value"` string, memanggil `$context.authorizer.numKey` mengembalikan `"1"` string, dan memanggil `$context.authorizer.boolKey` mengembalikan `"true"` string.
`$context.authenticate.error`	Pesan kesalahan dikembalikan dari upaya otentikasi.
`$context.authenticate.latency`	Latensi otentikasi di ms.
`$context.authenticate.status`	Kode status dikembalikan dari upaya otentikasi.
`$context.connectedAt`	Waktu koneksi yang diformat Epoch.
`$context.connectionId`	ID unik untuk koneksi yang dapat digunakan untuk membuat callback ke klien.
`$context.domainName`	Sebuah nama domain untuk WebSocket API. Ini dapat digunakan untuk membuat panggilan balik ke klien (bukan nilai hardcode).
`$context.error.message`	String yang berisi pesan kesalahan API Gateway.

Parameter	Deskripsi
`$context.error.messageString`	Nilai yang dikutip dari`$context.error.message` , yaitu`"$context.error.message"` .
`$context.error.responseType`	Jenis respons kesalahan.
`$context.error.validationErrorString`	Sebuah string yang berisi pesan kesalahan validasi rinci.
`$context.eventType`	Jenis acara:`CONNECT`,`MESSAGE`, atau`DISCONNECT` .
`$context.extendedRequestId`	Setara dengan`$context.requestId` .
`$context.identity.accountId`	ID AWS akun yang terkait dengan permintaan.
`$context.identity.apiKey`	Kunci pemilik API yang terkait dengan permintaan API berkemampuan kunci.
`$context.identity.apiKeyId`	ID kunci API yang terkait dengan permintaan API berkemampuan kunci
`$context.identity.caller`	Pengenal utama penelepon yang menandatangani permintaan. Didukung untuk rute yang menggunakan otorisasi IAM.

Parameter	Deskripsi
`$context.identity.cognitoAuthenticationProvider`	Daftar penyedia otentikasi Amazon Cognito yang dipisahkan koma yang digunakan oleh penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredenal Amazon Cognito. Misalnya, untuk identitas dari kumpulan pengguna Amazon Cognito, `cognito-idp.` *`region`*`.amazonaws.com/` *`user_pool_id`* `,cognito-idp.` *`region`*`.amazonaws.com/` *`user_pool_id`* `:CognitoSignIn:` *`token subject claim`* Untuk selengkapnya, lihat Menggunakan Identitas Federasi di Panduan Pengembang Amazon Cognito.
`$context.identity.cognitoAuthenticationType`	Jenis otentikasi Amazon Cognito dari penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredenal Amazon Cognito. Nilai yang mungkin termasuk `authenticated` untuk identitas yang diautentikasi dan `unauthenticated` untuk identitas yang tidak diautentikasi.
`$context.identity.cognitoIdentityId`	ID identitas Amazon Cognito dari penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredenal Amazon Cognito.
`$context.identity.cognitoIdentityPoolId`	ID kumpulan identitas Amazon Cognito dari penelepon yang membuat permintaan. Hanya tersedia jika permintaan ditandatangani dengan kredenal Amazon Cognito.

Parameter	Deskripsi
`$context.identity.principalOrgId`	ID AWS organisasi. Didukung untuk rute yang menggunakan otorisasi IAM.
`$context.identity.sourceIp`	Alamat IP sumber koneksi TCP membuat permintaan ke API Gateway.
`$context.identity.user`	Pengidentifikasi utama pengguna yang akan diotorisasi terhadap akses sumber daya. Didukung untuk rute yang menggunakan otorisasi IAM.
`$context.identity.userAgent`	Agen pengguna pemanggil API.
`$context.identity.userArn`	Nama Sumber Daya Amazon (ARN) dari pengguna efektif yang diidentifikasi setelah otentikasi.
`$context.integration.error`	Pesan kesalahan dikembalikan dari integrasi.
`$context.integration.integrationStatus`	Untuk integrasi proxy Lambda, kode status dikembalikan dariAWS Lambda, bukan dari kode fungsi Lambda backend.
`$context.integration.latency`	Latensi integrasi dalam ms. Setara dengan`$context.integrationLatency` .
`$context.integration.requestId`	ID permintaan AWS titik akhir. Setara dengan`$context.awsEndpointRequestId` .
`$context.integration.status`	Kode status dikembalikan dari integrasi. Untuk integrasi proxy Lambda, ini adalah kode status yang dikembalikan oleh kode fungsi Lambda Anda. Setara dengan`$context.integrationStatus` .

Parameter	Deskripsi
`$context.integrationLatency`	Latensi integrasi dalam ms, hanya tersedia untuk pencatatan akses.
`$context.messageId`	ID sisi server unik untuk pesan. Hanya tersedia ketika `$context.eventType` ada`MESSAGE`.
`$context.requestId`	Sama seperti`$context.extendedRequestId` .
`$context.requestTime`	Waktu permintaan yang diformat CLF (). `dd/MMM/yyyy:HH:mm:ss +-hhmm`
`$context.requestTimeEpoch`	Waktu permintaan yang diformat Epoch, dalam milidetik.
`$context.routeKey`	Kunci rute yang dipilih.
`$context.stage`	Tahap penerapan panggilan API (misalnya, beta atau prod).
`$context.status`	Status respon.
`$context.waf.error`	Pesan kesalahan dikembalikan dariAWS WAF.
`$context.waf.latency`	AWS WAFLatensi dalam ms.
`$context.waf.status`	Kode status dikembalikan dariAWS WAF.

Contoh beberapa format log akses yang umum digunakan ditampilkan di konsol API Gateway dan dicantumkan sebagai berikut.

- `CLF`(Format Log Umum):

```
$context.identity.sourceIp $context.identity.caller \
$context.identity.user [$context.requestTime] "$context.eventType $context.routeKey
 $context.connectionId" \
```

```
$context.status $context.requestId
```

Karakter kelanjutan (\) dimaksudkan sebagai alat bantu visual. Format log harus satu baris. Anda dapat menambahkan karakter baris baru (\n) di akhir format log untuk menyertakan baris baru di akhir setiap entri log.

- JSON:

```
{
"requestId":"$context.requestId", \
"ip": "$context.identity.sourceIp", \
"caller":"$context.identity.caller", \
"user":"$context.identity.user", \
"requestTime":"$context.requestTime", \
"eventType":"$context.eventType", \
"routeKey":"$context.routeKey", \
"status":"$context.status", \
"connectionId":"$context.connectionId"
}
```

Karakter kelanjutan (\) dimaksudkan sebagai alat bantu visual. Format log harus satu baris. Anda dapat menambahkan karakter baris baru (\n) di akhir format log untuk menyertakan baris baru di akhir setiap entri log.

- XML:

```
<request id="$context.requestId"> \
 <ip>$context.identity.sourceIp</ip> \
 <caller>$context.identity.caller</caller> \
 <user>$context.identity.user</user> \
 <requestTime>$context.requestTime</requestTime> \
 <eventType>$context.eventType</eventType> \
 <routeKey>$context.routeKey</routeKey> \
 <status>$context.status</status> \
 <connectionId>$context.connectionId</connectionId> \
</request>
```

Karakter kelanjutan (\) dimaksudkan sebagai alat bantu visual. Format log harus satu baris. Anda dapat menambahkan karakter baris baru (\n) di akhir format log untuk menyertakan baris baru di akhir setiap entri log.

- CSV(nilai yang dipisahkan koma):

```
$context.identity.sourceIp,$context.identity.caller, \
$context.identity.user,$context.requestTime,$context.eventType, \
$context.routeKey,$context.connectionId,$context.status, \
$context.requestId
```

Karakter kelanjutan (\) dimaksudkan sebagai alat bantu visual. Format log harus satu baris. Anda dapat menambahkan karakter baris baru (\n) di akhir format log untuk menyertakan baris baru di akhir setiap entri log.

Referensi API Gateway Amazon Resource Name (ARN)

Tabel berikut mencantumkan Nama Sumber Daya Amazon (ARN) untuk sumber daya API Gateway. Untuk mempelajari lebih lanjut tentang menggunakan ARN dalam AWS Identity and Access Management kebijakan, lihat Cara kerja Amazon API Gateway dengan IAM danKontrol akses ke API dengan izin IAM.

Sumber daya API dan WebSocket API HTTP

Resource	ARN
AccessLogSettings	`arn:partition :apigatew ay: region::/apis/api-id/ stages/stage-name /accesslo gsettings`
Api	`arn:partition :apigatew ay: region::/apis/api-id`
Apis	`arn:partition :apigatew ay: region::/apis`
ApiMapping	`arn:partition :apigatew ay: region::/domainnames/ domain-na me /apimappings/ id`
ApiMappings	`arn:partition :apigatew ay: region::/domainnames/ domain-na me /apimappings`
Pengotorisasi	`arn:partition :apigatew ay: region::/apis/api-id/authoriz ers/ id`
Pengotorisasi	`arn:partition :apigatew ay: region::/apis/api-id/authoriz ers`

Resource	ARN
Cors	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/cors
Deployment	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/deployme nts/ *id*
Deployment	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/deployme nts
DomainName	arn:*partition* :apigatew ay: *region*::/domainnames/ *domain-na me*
DomainNames	arn:*partition* :apigatew ay: *region*::/domainnames
ExportedAPI	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/exports/ *specification*
Integrasi	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/integrat ions/ *integration-id*
Integrasi	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/integrat ions
IntegrationResponse	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/integrat ionresponses/ *integration-respon se*

Resource	ARN
IntegrationResponses	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/integrat ionresponses
Model	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/models/*id*
Model	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/models
ModelTemplate	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/models/*id*/ template
Rute	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/routes/*id*
Rute	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/routes
RouteRequestParameter	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/routes/*id*/ requestparameters/ *key*
RouteResponse	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/routes/*id*/ routeresponses/ *id*
RouteResponses	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/routes/*id*/ routeresponses
RouteSettings	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/ stages/*stage-name* /routeset tings/ *route-key*

Resource	ARN
Tahap	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/ stages/*stage-name*
Tahapan	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/stages
VpcLink	arn:*partition* :apigatew ay: *region*::/vpclinks/ *vpclink-id*
VpcLinks	arn:*partition* :apigatew ay: *region*::/vpclinks

Sumber daya REST API

Resource	ARN
Akun	arn:*partition* :apigatew ay: *region*::/account
ApiKey	arn:*partition* :apigatew ay: *region*::/apikeys/ *id*
ApiKeys	arn:*partition* :apigatew ay: *region*::/apikeys
Pengotorisasi	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ authorizers/ *id*
Pengotorisasi	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ authorizers

Resource	ARN
BasePathMapping	arn:*partition* :apigatew ay: *region*::/domainnames/ *domain-na me* /basepathmappings/ *basepath*
BasePathMappings	arn:*partition* :apigatew ay: *region*::/domainnames/ *domain-na me* /basepathmappings
ClientCertificate	arn:*partition* :apigatew ay: *region*::/clientcertifica tes/ *id*
ClientCertificates	arn:*partition* :apigatew ay: *region*::/clientcertificates
Deployment	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ deployments/ *id*
Deployment	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ deployments
DocumentationPart	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ documentation/parts/ *id*
DocumentationParts	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ documentation/parts
DocumentationVersion	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ documentation/versions/ *version*

Resource	ARN
DocumentationVersions	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ documentation/versions
DomainName	arn:*partition* :apigatew ay: *region*::/domainnames/ *domain-na me*
DomainNames	arn:*partition* :apigatew ay: *region*::/domainnames
GatewayResponse	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ gatewayresponses/ *response-type*
GatewayResponses	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ gatewayresponses
Integrasi	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ resources/ *resource-id* /methods/ *http-method* /integration
IntegrationResponse	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ resources/ *resource-id* /methods/ *http-method* /integration/respo nses/ *status-code*
Metode	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ resources/ *resource-id* /methods/ *http-method*

Resource	ARN
MethodResponse	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ resources/ *resource-id* /methods/ *http-method* /responses/ *status-co de*
Model	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ models/*model-name*
Model	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ models
RequestValidator	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ requestvalidators/ *id*
RequestValidators	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ requestvalidators
Resource	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ resources/ *id*
Sumber daya	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ resources
RestApi	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*
RestApis	arn:*partition* :apigatew ay: *region*::/restapis

Resource	ARN
Tahap	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ stages/*stage-name*
Tahapan	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ stages
Tanda	arn:*partition* :apigatew ay: *region*::/tags/*url-encoded-resource-arn*
Templat	arn:*partition* :apigatew ay: *region*::/restapis/models / *model-name* /template
UsagePlan	arn:*partition* :apigatew ay: *region*::/usageplans/ *usageplan-id*
UsagePlans	arn:*partition* :apigatew ay: *region*::/usageplans
UsagePlanKey	arn:*partition* :apigatew ay: *region*::/usageplans/ *usageplan-id* /keys/*id*
UsagePlanKeys	arn:*partition* :apigatew ay: *region*::/usageplans/ *usageplan-id* /keys
VpcLink	arn:*partition* :apigatew ay: *region*::/vpclinks/ *vpclink-id*
VpcLinks	arn:*partition* :apigatew ay: *region*::/vpclinks

execute-api(API HTTP, WebSocket API, dan REST API)

Resource	ARN
WebSocket Titik akhir API	`arn:`*`partition`*` :execute-api: `*`region`*`:`*`account-id`*` :`*`api-id`*`/`*`stage`*`/`*`route-key`*
HTTP API dan titik akhir REST API*	`arn:`*`partition`*` :execute-api: `*`region`*`:`*`account-id`*` :`*`api-id`*`/`*`stage`*`/`*`http-method`*` /`*`resource-path`*
Pengotorisasi Lambda**	`arn:`*`partition`*` :execute-api: `*`region`*`:`*`account-id`*` :`*`api-id`*`/authorizers/ `*`authorizer-id`*

* ARN untuk titik akhir `$default` rute untuk API HTTP adalah. `arn:`*`partition`*`:execute-api:`*`region`*`:`*`account-id`*`:`*`api-id`*`/`*`*`/$default`

** ARN ini hanya berlaku saat menyetel `SourceArn` kondisi dalam kebijakan sumber daya untuk fungsi otorisasi Lambda. Sebagai contoh, lihat the section called “Buat Authorizer Lambda”.

Bekerja dengan ekstensi API Gateway ke OpenAPI

Ekstensi API Gateway mendukung otorisasi AWS khusus dan integrasi API khusus Gateway API untuk REST API dan API HTTP. Di bagian ini, kami menjelaskan ekstensi API Gateway ke spesifikasi OpenAPI.

> **Tip**
>
> Untuk memahami bagaimana ekstensi API Gateway digunakan dalam aplikasi, Anda dapat menggunakan konsol API Gateway untuk membuat REST API atau HTTP API dan mengekspornya ke file definisi OpenAPI. Untuk informasi selengkapnya tentang cara mengekspor API, lihat Ekspor REST API dari API Gateway danMengekspor API HTTP dari API Gateway.

Topik

- x-amazon-apigateway-integrationObjek. RequestParameters
- x-amazon-apigateway-integration.response objek
- x-amazon-apigateway-integration.response objek
- x-amazon-apigateway-integrationObjek. ResponseTemplates
- x-amazon-apigateway-integration.ResponseParameters objek
- x-amazon-apigateway-integrationObjek .tlsConfig
- x-amazon-apigateway-minimum-ukuran kompresi
- x-amazon-apigateway-policy
- x-amazon-apigateway-requestproperti -validator
- x-amazon-apigateway-request-validator objek
- x-amazon-apigateway-request-Validators.requestValidator objek
- x-amazon-apigateway-tag-nilai properti

x-amazon-apigateway-anyobjek -metode

Menentukan Objek Operasi OpenAPI untuk ANY metode catch-all API Gateway dalam Objek Item Jalur OpenAPI. Objek ini dapat eksis bersama objek Operasi lainnya dan akan menangkap metode HTTP apa pun yang tidak dideklarasikan secara eksplisit.

Tabel berikut mencantumkan properti yang diperluas oleh API Gateway. Untuk properti Operasi OpenAPI lainnya, lihat spesifikasi OpenAPI.

Properti

Nama properti	Tipe	Deskripsi
`isDefaultRoute`	`Boolean`	Menentukan apakah rute adalah `$default` rute. Haya didukung untuk API HTTP. Untuk mempelajari selengkap nya, lihat Bekerja dengan rute untuk HTTP API.
`x-amazon-apigateway-integration`	x-amazon-apigateway-integration objek	Menentukan integrasi metode dengan backend. Ini adalah properti yang

Nama properti	Tipe	Deskripsi
		diperluas dari objek Operasi OpenAPI. Integrasi dapat berupa tipeAWS,AWS_PROXY ,HTTP,HTTP_PROXY , atauMOCK.

x-amazon-apigateway-any-contoh metode

Contoh berikut mengintegrasikan ANY metode pada sumber daya proxy,{proxy+}, dengan fungsi Lambda,. TestSimpleProxy

```
    "/{proxy+}": {
      "x-amazon-apigateway-any-method": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "proxy",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {},
        "x-amazon-apigateway-integration": {
          "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:TestSimpleProxy/invocations",
          "httpMethod": "POST",
          "type": "aws_proxy"
        }
```

Contoh berikut membuat $default rute untuk HTTP API yang terintegrasi dengan fungsi Lambda,. HelloWorld

```
"/$default": {
    "x-amazon-apigateway-any-method": {
      "isDefaultRoute": true,
```

```
      "x-amazon-apigateway-integration": {
        "type": "AWS_PROXY",
        "httpMethod": "POST",
        "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:HelloWorld/invocations",
        "timeoutInMillis": 1000,
        "connectionType": "INTERNET",
        "payloadFormatVersion": 1.0
      }
   }
}
```

x-amazon-apigateway-cors objek

Menentukan konfigurasi cross-origin resource sharing (CORS) untuk HTTP API. Ekstensi ini berlaku untuk struktur OpenAPI tingkat root. Untuk mempelajari selengkapnya, lihat Mengonfigurasi CORS untuk HTTP API.

Properti

Nama properti	Tipe	Deskripsi
`allowOrigins`	`Array`	Menentukan asal-usul diizinkan.
`allowCredentials`	`Boolean`	Menentukan apakah kredensial termasuk dalam permintaan CORS.
`exposeHeaders`	`Array`	Menentukan header yang diekspos.
`maxAge`	`Integer`	Menentukan jumlah detik bahwa browser harus cache hasil permintaan preflight.
`allowMethods`	`Array`	Menentukan metode HTTP diperbolehkan.

Nama properti	Tipe	Deskripsi
`allowHeaders`	`Array`	Menentukan header diperbolehkan.

x-amazon-apigateway-cors contoh

Berikut ini adalah contoh konfigurasi CORS untuk HTTP API.

```
"x-amazon-apigateway-cors": {
    "allowOrigins": [
      "https://www.example.com"
    ],
    "allowCredentials": true,
    "exposeHeaders": [
      "x-apigateway-header",
      "x-amz-date",
      "content-type"
    ],
    "maxAge": 3600,
    "allowMethods": [
      "GET",
      "OPTIONS",
      "POST"
    ],
    "allowHeaders": [
      "x-apigateway-header",
      "x-amz-date",
      "content-type"
    ]
}
```

x-amazon-apigateway-apiproperti -key-source

Tentukan sumber untuk menerima kunci API untuk membatasi metode API yang memerlukan kunci. Properti API-level ini adalah tipe. `String`

Tentukan sumber kunci API untuk permintaan. Nilai yang valid adalah:

- `HEADER`untuk menerima kunci API dari `X-API-Key` header permintaan.

- `AUTHORIZER`Untuk menerima kunci API dari `UsageIdentifierKey` dari pengotorisasi Lambda (sebelumnya dikenal sebagai otorisasi khusus).

x-amazon-apigateway-api-contoh sumber kunci

Contoh berikut menetapkan `X-API-Key` header sebagai sumber kunci API.

OpenAPI 2.0

```
{
  "swagger" : "2.0",
  "info" : {
    "title" : "Test1"
   },
  "schemes" : [ "https" ],
  "basePath" : "/import",
  "x-amazon-apigateway-api-key-source" : "HEADER",
    .
    .
    .
}
```

OpenAPI 3.0.1

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "Test1"
  },
  "servers" : [ {
    "url" : "/{basePath}",
    "variables" : {
      "basePath" : {
        "default" : "import"
      }
    }
  } ],
  "x-amazon-apigateway-api-key-source" : "HEADER",
    .
    .
    .
```

```
}
```

x-amazon-apigateway-auth objek

Mendefinisikan jenis otorisasi yang akan diterapkan untuk otorisasi pemanggilan metode di API Gateway.

Properti

Nama properti	Tipe	Deskripsi
`type`	`string`	Menentukan jenis otorisasi . Tentukan `"NONE"` untuk akses terbuka. Tentukan `"AWS_IAM"` untuk menggunakan izin IAM. Nilai tidak peka huruf besar/kecil.

x-amazon-apigateway-auth contoh

Contoh berikut menetapkan jenis otorisasi untuk metode API.

OpenAPI 3.0.1

```
{
  "openapi": "3.0.1",
  "info": {
    "title": "openapi3",
    "version": "1.0"
  },
  "paths": {
    "/protected-by-iam": {
      "get": {
        "x-amazon-apigateway-auth": {
          "type": "AWS_IAM"
        }
      }
    }
  }
```

```
}
```

x-amazon-apigateway-authorizer objek

Mendefinisikan otorisasi Lambda, kumpulan pengguna Amazon Cognito, atau otorisasi JWT yang akan diterapkan untuk otorisasi pemanggilan metode di API Gateway. Ekstensi ini berlaku untuk definisi keamanan di OpenAPI 2 dan OpenAPI 3.

Properti

Nama properti	Tipe	Deskripsi
`type`	`string`	Jenis otorisasi. Ini adalah properti yang diperlukan. Untuk REST API, tentukan `token` otorisasi dengan identitas pemanggil yang disematkan dalam token otorisasi. Tentukan `request` otorisasi dengan identitas pemanggil yang terkandung dalam parameter permintaan. Tentukan `cognito_user_pools` otorisasi yang menggunakan kumpulan pengguna Amazon Cognito untuk mengontrol akses ke API Anda. Untuk API HTTP, tentukan `request` otorisasi Lambda dengan identitas pemanggil yang terdapat dalam parameter permintaan. Tentukan `jwt` untuk otorisasi JWT.

Nama properti	Tipe	Deskripsi
`authorizerUri`	`string`	Uniform Resource Identifier (URI) dari fungsi Lambda authorizer. Sintaksnya adalah sebagai berikut: `"arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:us-east-1:account-id:function:auth_function_name/invocations"`
`authorizerCredentials`	`string`	Kredensyal yang diperlukan untuk memanggil otorisasi, jika ada, dalam bentuk ARN dari peran eksekusi IAM. *Misalnya, "arn:aws:iam::account-id:iam_role".*
`authorizerPayloadFormatVersion`	`string`	Untuk API HTTP, tentukan format data yang dikirimkan API Gateway ke pengotorisasi Lambda, dan bagaimana API Gateway menafsirkan respons dari Lambda. Untuk mempelajari selengkapnya, lihat the section called "Versi format payload".

Nama properti	Tipe	Deskripsi
`enableSimpleResponses`	`Boolean`	Untuk API HTTP, menentukan apakah `request` otorisasi mengembalikan nilai Boolean atau kebijakan IAM. Didukung hanya untuk otorisasi dengan `authorizerPayloadFormatVersion` dari. `2.0` Jika diaktifkan, fungsi Lambda authorizer mengembalikan nilai Boolean. Untuk mempelajari selengkapnya, lihat the section called "Respons fungsi Lambda untuk format 2.0".
`identitySource`	`string`	Daftar ekspresi pemetaan parameter permintaan yang dipisahkan koma sebagai sumber identitas. Berlaku untuk otorisasi `request` dan `jwt` tipe saja.
`jwtConfiguration`	`Object`	Menentukan penerbit dan audiens untuk otorisasi JWT. Untuk mempelajari selengkapnya, lihat JWTConfiguration di Referensi API Gateway Versi 2 API. Haya didukung untuk API HTTP.
`identityValidationExpression`	`string`	Ekspresi reguler untuk memvalidasi token sebagai identitas yang masuk. Misalnya, "^x- [a-z] +". Hanya didukung untuk REST API.

Nama properti	Tipe	Deskripsi
`authorizerResultTtlInSeconds`	`string`	Jumlah detik selama hasil otorisasi di-cache.
`providerARNs`	Sebuah array dari `string`	Daftar ARN kumpulan pengguna Amazon Cognito untuk. `COGNITO_USER_POOLS`

x-amazon-apigateway-authorizer contoh untuk REST API

Contoh definisi keamanan OpenAPI berikut menentukan otorisasi Lambda dari jenis “token” dan bernama. `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
     }
   }
 }
```

Cuplikan objek operasi OpenAPI berikut menetapkan `GET /http` untuk menggunakan otorisasi Lambda sebelumnya.

```
  "/http" : {
     "get" : {
       "responses" : { },
       "security" : [ {
         "test-authorizer" : [ ]
       } ],
       "x-amazon-apigateway-integration" : {
         "type" : "http",
         "responses" : {
           "default" : {
             "statusCode" : "200"
           }
         },
         "httpMethod" : "GET",
         "uri" : "http://api.example.com"
       }
     }
   }
```

Contoh definisi keamanan OpenAPI berikut menentukan otorisasi Lambda dari jenis “permintaan”, dengan parameter header tunggal () `auth` sebagai sumber identitas. `securityDefinitions`Itu dinamai`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
      }
    }
}
```

Contoh definisi keamanan OpenAPI berikut menentukan otorisasi Lambda dari jenis “permintaan”, dengan satu header (`HeaderAuth1`) dan satu parameter string kueri sebagai sumber identitas. `QueryString1`

```
"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
      }
    }
}
```

Contoh definisi keamanan OpenAPI berikut menentukan otorisasi Lambda API Gateway dari jenis “permintaan”, dengan variabel tahap tunggal () `stage` sebagai sumber identitas.

```
"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
      }
    }
}
```

Contoh definisi keamanan OpenAPI berikut menentukan kumpulan pengguna Amazon Cognito sebagai otorisasi.

```
 "securityDefinitions": {
    "cognito-pool": {
      "type": "apiKey",
      "name": "Authorization",
      "in": "header",
      "x-amazon-apigateway-authtype": "cognito_user_pools",
      "x-amazon-apigateway-authorizer": {
        "type": "cognito_user_pools",
        "providerARNs": [
          "arn:aws:cognito-idp:us-east-1:123456789012:userpool/us-east-1_ABC123"
        ]
      }
    }
```

Cuplikan objek operasi OpenAPI berikut menyetel `GET /http` untuk menggunakan kumpulan pengguna Amazon Cognito sebelumnya sebagai otorisasi, tanpa cakupan khusus.

```
  "/http" : {
    "get" : {
```

```
        "responses" : { },
        "security" : [ {
          "cognito-pool" : [ ]
        } ],
        "x-amazon-apigateway-integration" : {
          "type" : "http",
          "responses" : {
            "default" : {
              "statusCode" : "200"
            }
          },
          "httpMethod" : "GET",
          "uri" : "http://api.example.com"
        }
      }
    }
```

x-amazon-apigateway-authorizer contoh untuk HTTP API

Contoh OpenAPI 3.0 berikut membuat otorisasi JWT untuk API HTTP yang menggunakan Amazon Cognito sebagai penyedia identitas, dengan header sebagai sumber identitas. `Authorization`

```
"securitySchemes": {
  "jwt-authorizer-oauth": {
    "type": "oauth2",
     "x-amazon-apigateway-authorizer": {
       "type": "jwt",
       "jwtConfiguration": {
          "issuer": "https://cognito-idp.region.amazonaws.com/userPoolId",
          "audience": [
            "audience1",
            "audience2"
          ]
        },
        "identitySource": "$request.header.Authorization"
    }
  }
}
```

Contoh OpenAPI 3.0 berikut menghasilkan otorisasi JWT yang sama dengan contoh sebelumnya. Namun, contoh ini menggunakan `openIdConnectUrl` properti OpenAPI untuk mendeteksi penerbit secara otomatis. `openIdConnectUrl`Harus sepenuhnya terbentuk.

```
"securitySchemes": {
  "jwt-authorizer-autofind": {
    "type": "openIdConnect",
    "openIdConnectUrl": "https://cognito-idp.region.amazonaws.com/userPoolId/.well-
known/openid-configuration",
    "x-amazon-apigateway-authorizer": {
      "type": "jwt",
      "jwtConfiguration": {
        "audience": [
          "audience1",
          "audience2"
        ]
      },
      "identitySource": "$request.header.Authorization"
    }
  }
}
```

Contoh berikut membuat authorizer Lambda untuk HTTP API. Authorizer contoh ini menggunakan `Authorization` header sebagai sumber identitasnya. Authorizer menggunakan versi format `2.0` payload, dan mengembalikan nilai Boolean, karena `enableSimpleResponses` diatur ke. `true`

```
"securitySchemes" : {
  "lambda-authorizer" : {
    "type" : "apiKey",
    "name" : "Authorization",
    "in" : "header",
    "x-amazon-apigateway-authorizer" : {
      "type" : "request",
      "identitySource" : "$request.header.Authorization",
      "authorizerUri" : "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:function-name/invocations",
      "authorizerPayloadFormatVersion" : "2.0",
      "authorizerResultTtlInSeconds" : 300,
      "enableSimpleResponses" : true
    }
  }
}
```

x-amazon-apigateway-authtype properti

Untuk REST API, ekstensi ini dapat digunakan untuk menentukan jenis khusus dari otorisasi Lambda. Dalam hal ini, nilainya adalah bentuk bebas. Misalnya, API mungkin memiliki beberapa otorisasi Lambda yang menggunakan skema internal yang berbeda. Anda dapat menggunakan ekstensi ini untuk mengidentifikasi skema internal otorisasi Lambda.

Lebih umum, di HTTP API dan REST API, ini juga dapat digunakan sebagai cara untuk mendefinisikan otorisasi IAM di beberapa operasi yang berbagi skema keamanan yang sama. Dalam hal ini, istilah tersebut `awsSigv4` adalah istilah yang dicadangkan, bersama dengan istilah apa pun yang diawali oleh`aws`.

Ekstensi ini berlaku untuk skema keamanan `apiKey` tipe di OpenAPI 2 dan OpenAPI 3.

x-amazon-apigateway-authtype contoh

Contoh OpenAPI 3 berikut mendefinisikan otorisasi IAM di beberapa sumber daya dalam REST API atau HTTP API:

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "openapi3",
    "version" : "1.0"
  },
  "paths" : {
    "/operation1" : {
      "get" : {
        "responses" : {
          "default" : {
            "description" : "Default response"
          }
        },
        "security" : [ {
          "sigv4Reference" : [ ]
        } ]
      }
    },
    "/operation2" : {
      "get" : {
        "responses" : {
```

```
          "default" : {
            "description" : "Default response"
          }
        },
        "security" : [ {
          "sigv4Reference" : [ ]
        } ]
      }
    }
  },
  "components" : {
    "securitySchemes" : {
      "sigv4Reference" : {
        "type" : "apiKey",
        "name" : "Authorization",
        "in" : "header",
        "x-amazon-apigateway-authtype": "awsSigv4"
      }
    }
  }
}
```

Contoh OpenAPI 3 berikut mendefinisikan otorisasi Lambda dengan skema khusus untuk REST API:

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "openapi3 for REST API",
    "version" : "1.0"
  },
  "paths" : {
    "/protected-by-lambda-authorizer" : {
      "get" : {
        "responses" : {
          "200" : {
            "description" : "Default response"
          }
        },
        "security" : [ {
          "myAuthorizer" : [ ]
        } ]
      }
    }
```

```
  },
  "components" : {
    "securitySchemes" : {
      "myAuthorizer" : {
        "type" : "apiKey",
        "name" : "Authorization",
        "in" : "header",
        "x-amazon-apigateway-authorizer" : {
          "identitySource" : "method.request.header.Authorization",
          "authorizerUri" : "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:account-id:function:function-name/invocations",
          "authorizerResultTtlInSeconds" : 300,
          "type" : "request",
          "enableSimpleResponses" : false
        },
        "x-amazon-apigateway-authtype": "Custom scheme with corporate claims"
      }
    }
  },
  "x-amazon-apigateway-importexport-version" : "1.0"
}
```

Lihat juga

Authorizer.authType

x-amazon-apigateway-binaryproperti -media-tipe

Menentukan daftar jenis media biner yang akan didukung oleh API Gateway, seperti `application/octet-stream` dan`image/jpeg`. Ekstensi ini adalah array JSON. Ini harus disertakan sebagai ekstensi vendor tingkat atas ke dokumen OpenAPI.

x-amazon-apigateway-binary-media-jenis contoh

Contoh berikut menunjukkan urutan pencarian pengkodean API.

```
"x-amazon-apigateway-binary-media-types": [ "application/octet", "image/jpeg" ]
```

x-amazon-apigateway-documentation objek

Mendefinisikan bagian dokumentasi yang akan diimpor ke API Gateway. Objek ini adalah objek JSON yang berisi array dari `DocumentationPart` instance.

Properti

Nama properti	Tipe	Deskripsi
`documentationParts`	`Array`	Array `DocumentationPart` instance yang diekspor atau diimpor.
`version`	`String`	Pengidentifikasi versi snapshot dari bagian dokumentasi yang diekspor.

x-amazon-apigateway-documentation contoh

Contoh ekstensi API Gateway ke OpenAPI berikut mendefinisikan `DocumentationParts` instance yang akan diimpor atau diekspor dari API di API Gateway.

```
{ ...
  "x-amazon-apigateway-documentation": {
    "version": "1.0.3",
    "documentationParts": [
       {
         "location": {
           "type": "API"
       },
        "properties": {
          "description": "API description",
          "info": {
            "description": "API info description 4",
            "version": "API info version 3"
          }
        }
      },
      {
         … // Another DocumentationPart instance
```

```
      }
    ]
  }
}
```

x-amazon-apigateway-endpoint-konfigurasi objek

Menentukan rincian konfigurasi endpoint untuk API. Ekstensi ini adalah properti yang diperluas dari objek Operasi OpenAPI. Objek ini harus ada di ekstensi vendor tingkat atas untuk Swagger 2.0. Untuk OpenAPI 3.0, itu harus ada di bawah ekstensi vendor dari objek Server.

Properti

Nama properti	Tipe	Deskripsi
`disableExecuteApiEndpoint`	Boolean	Menentukan apakah klien dapat memanggil API dengan menggunakan titik akhir `execute-api` default. Secara default, klien dapat memanggil API Anda dengan titik `https://{api_id}.execute-api.{region}.amazonaws.com` akhir default. Untuk mengharuskan klien menggunakan nama domain khusus untuk menjalankan API Anda, tentukan`true`.
`vpcEndpointIds`	Sebuah array dari `String`	Daftar VpcEndpoint pengidentifikasi yang digunakan untuk membuat catatan alias Route 53 untuk REST API. Ini hanya didukung untuk REST API tipe `PRIVATE` titik akhir.

x-amazon-apigateway-endpoint-contoh konfigurasi

Contoh berikut mengaitkan titik akhir VPC tertentu ke REST API.

```
"x-amazon-apigateway-endpoint-configuration": {
    "vpcEndpointIds": ["vpce-0212a4ababd5b8c3e", "vpce-01d622316a7df47f9"]
}
```

Contoh berikut menonaktifkan titik akhir default untuk API.

```
"x-amazon-apigateway-endpoint-configuration": {
    "disableExecuteApiEndpoint": true
}
```

x-amazon-apigateway-gateway-respon objek

Mendefinisikan respons gateway untuk API sebagai GatewayResponsestring-to-map dari pasangan nilai kunci. Ekstensi ini berlaku untuk struktur OpenAPI tingkat root.

Properti

Nama properti	Tipe	Deskripsi
responseType	x-amazon-apigateway-gateway-Responses.gateway Response	A `GatewayResponse` untuk *ResponseType* yang ditentukan.

x-amazon-apigateway-gateway-contoh tanggapan

Ekstensi API Gateway berikut ke OpenAPI contoh mendefinisikan GatewayResponsespeta yang berisi dua GatewayResponseinstance—satu untuk tipe dan satu lagi untuk tipe. `DEFAULT_4XX` `INVALID_API_KEY`

```
{
  "x-amazon-apigateway-gateway-responses": {
    "DEFAULT_4XX": {
      "responseParameters": {
```

```
        "gatewayresponse.header.Access-Control-Allow-Origin": "'domain.com'"
      },
      "responseTemplates": {
        "application/json": "{\"message\": test 4xx b }"
      }
    },
    "INVALID_API_KEY": {
      "statusCode": "429",
      "responseTemplates": {
        "application/json": "{\"message\": test forbidden }"
      }
    }
  }
}
```

x-amazon-apigateway-gateway-Responses.gatewayResponse objek

Mendefinisikan respons gateway dari jenis respons tertentu, termasuk kode status, parameter respons apa pun yang berlaku, atau templat respons.

Properti

Nama properti	Tipe	Deskripsi
responseParameters	x-amazon-apigateway-gateway-Responses.ResponseParameters	Menentukan GatewayResponseparameter, yaitu parameter header. Nilai parameter dapat mengambil nilai parameter permintaan masuk atau nilai kustom statis.
responseTemplates	x-amazon-apigateway-gateway-Responses.ResponseTemplates	Menentukan template pemetaan dari respon gateway. Template tidak diproses oleh mesin VTL.
statusCode	`string`	Kode status HTTP untuk respons gateway.

x-amazon-apigateway-gateway-Responses.gatewayResponse contoh

Contoh berikut dari ekstensi API Gateway ke OpenAPI mendefinisikan a GatewayResponseuntuk menyesuaikan `INVALID_API_KEY` respons untuk mengembalikan kode status`456`, nilai `api-key` header permintaan masuk, dan pesan. `"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\" }"
  }
}
```

x-amazon-apigateway-gateway-Responses.ResponseParameters objek

Mendefinisikan string-to-string peta pasangan kunci-nilai untuk menghasilkan parameter respons gateway dari parameter permintaan masuk atau menggunakan string literal. Hanya didukung untuk REST API.

Properti

Nama properti	Tipe	Deskripsi
`gatewayre sponse.` *`param-pos ition`* `.`*`param-name`*	`string`	*`param-pos ition`* bisa`header`,`path`, atau`querystring` . Untuk informasi selengkapnya, lihat Metode peta meminta data ke parameter permintaan integrasi.

x-amazon-apigateway-gateway-Responses.ResponseParameters contoh

Contoh ekstensi OpenAPI berikut menunjukkan ekspresi pemetaan parameter GatewayResponserespons untuk mengaktifkan dukungan CORS untuk sumber daya pada domain. `*.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
}
```

x-amazon-apigateway-gateway-Responses.responseTemplates objek

Mendefinisikan template GatewayResponsepemetaan, sebagai string-to-string peta pasangan kunci-nilai, untuk respons gateway yang diberikan. Untuk setiap pasangan kunci-nilai, kuncinya adalah tipe konten. Misalnya, “application/json” dan nilainya adalah template pemetaan stringified untuk substitusi variabel sederhana. Template `GatewayResponse` pemetaan tidak diproses oleh mesin Velocity Template Language (VTL).

Properti

Nama properti	Tipe	Deskripsi
`content-type`	`string`	Template pemetaan `GatewayResponse` tubuh yang hanya mendukung substitusi variabel sederhana untuk menyesuaikan badan respons gateway.

x-amazon-apigateway-gateway-Responses.responseTemplates contoh

Contoh ekstensi OpenAPI berikut menunjukkan template GatewayResponsepemetaan untuk menyesuaikan respons kesalahan yang dihasilkan oleh gateway API ke dalam format khusus aplikasi.

```
      "responseTemplates": {
        "application/json": "{ \"message\": $context.error.messageString, \"type\":
$context.error.responseType, \"statusCode\": '488' }"
      }
```

Contoh ekstensi OpenAPI berikut menunjukkan template GatewayResponsepemetaan untuk mengganti respons kesalahan yang dihasilkan oleh gateway API dengan pesan kesalahan statis.

```
      "responseTemplates": {
        "application/json": "{ \"message\": 'API-specific errors' }"
      }
```

x-amazon-apigateway-importexport-versi

Menentukan versi algoritma impor dan ekspor API Gateway untuk API HTTP. Saat ini, satu-satunya nilai yang didukung adalah `1.0`. Untuk mempelajari lebih lanjut, lihat ExportVersion di Referensi API Gateway Versi 2 API.

x-amazon-apigateway-importexport-versi contoh

Contoh berikut menetapkan versi impor dan ekspor ke`1.0`.

```
{
    "openapi": "3.0.1",
    "x-amazon-apigateway-importexport-version": "1.0",
    "info": { ...
```

x-amazon-apigateway-integration objek

Menentukan rincian integrasi backend yang digunakan untuk metode ini. Ekstensi ini adalah properti yang diperluas dari objek Operasi OpenAPI. Hasilnya adalah objek integrasi API Gateway.

Properti

Nama properti	Tipe	Deskripsi
`cacheKeyParameters`	Sebuah array dari `string`	Daftar parameter permintaan yang nilainya akan di-cache.
`cacheNamespace`	`string`	Grup tag spesifik API dari parameter yang di-cache terkait.
`connectionId`	`string`	ID a VpcLinkuntuk integrasi pribadi.
`connectionType`	`string`	Jenis koneksi integrasi. Nilai yang valid adalah `"VPC_LINK"` untuk integrasi pribadi atau`"INTERNET"` , sebaliknya.
`credentials`	`string`	Untuk kredensyal berbasis peran AWS IAM, tentukan ARN dari peran IAM yang sesuai. Jika tidak ditentukan, kredensyal default ke izin berbasis sumber daya yang harus ditambahkan secara manual agar API dapat mengakses sumber daya. Untuk informasi selengkapnya, lihat Memberikan Izin Menggunakan Kebijakan Sumber Daya.

Nama properti	Tipe	Deskripsi
		Catatan: Saat menggunak an kredensi IAM, pastikan titik akhir Regional AWS STS diaktifkan untuk Wilayah tempat API ini digunakan untuk kinerja terbaik.
`contentHandling`	`string`	Minta jenis konversi pengkodean muatan. Nilai yang valid adalah 1)`CONVERT_TO_TEXT` , untuk mengubah muatan biner menjadi string yang dikodekan base64 atau mengubah muatan teks menjadi string yang `utf-8` dikodekan atau melewati muatan teks secara asli tanpa modifikasi, dan 2)`CONVERT_TO_BINARY` , untuk mengubah muatan teks menjadi gumpalan yang diterjemahkan base64 atau melewati muatan biner secara asli tanpa modifikasi.
`httpMethod`	`string`	Metode HTTP yang digunakan dalam permintaan integrasi . Untuk pemanggilan fungsi Lambda, nilainya harus. `POST`

Nama properti	Tipe	Deskripsi
`integrationSubtype`	`string`	Menentukan subtipe integrasi untuk integrasi AWS layanan. Haya didukung untuk API HTTP. Untuk subtipe integrasi yang didukung, lihatthe section called “AWSReferensi integrasi layanan”.
`passthroughBehavior`	`string`	Menentukan bagaimana payload permintaan jenis konten yang tidak dipetakan diteruskan melalui permintaa n integrasi tanpa modifikasi. Nilai yang didukung adalah `when_no_templates` , `when_no_match` , dan `never`. Untuk informasi selengkapnya, lihat Integrati on.PassThroughBehavior.
`payloadFormatVersion`	`string`	Menentukan format muatan yang dikirim ke integrasi. Diperlukan untuk API HTTP. Untuk API HTTP, nilai-nilai yang didukung untuk integrasi proxy Lambda adalah `1.0` dan `2.0`. Untuk semua integrasi lainnya, `1.0` adalah satu-satu nya nilai yang didukung. Untuk mempelajari selengkapnya, lihat the section called “AWS Lambdaintegrasi” dan the section called “AWSReferensi integrasi layanan”.

Nama properti	Tipe	Deskripsi
`requestParameters`	x-amazon-apigateway-integrationObjek. RequestParameters	Untuk REST API, tentukan pemetaan dari parameter permintaan metode hingga parameter permintaan integrasi. Parameter permintaan yang didukung adalah `querystring` `path`,`header`,, dan`body`. Untuk API HTTP, parameter permintaan adalah peta nilai kunci yang menentukan parameter yang diteruskan ke `AWS_PROXY` integrasi dengan yang ditentukan. `integrationSubtype` Anda dapat memberikan nilai statis, atau data permintaan peta, variabel tahap, atau variabel konteks yang dievaluasi pada saat runtime. Untuk mempelajari selengkapnya, lihat the section called “AWSintegrasi layanan”.
`requestTemplates`	x-amazon-apigateway-integrationObjek. RequestTemplates	Memetakan template untuk payload permintaan tipe MIME tertentu.
`responses`	x-amazon-apigateway-integration.response objek	Mendefinisikan respons metode dan menentukan pemetaan parameter atau pemetaan muatan yang diinginkan dari respons integrasi hingga respons metode.

Nama properti	Tipe	Deskripsi
`timeoutInMillis`	`integer`	Batas waktu integrasi antara 50 ms dan 29.000 ms.
`type`	`string`	Jenis integrasi dengan backend yang ditentukan. Nilai yang valid adalah: • `http`atau`http_proxy` , untuk integrasi dengan backend HTTP. • `aws_proxy` , untuk integrasi dengan fungsi AWS Lambda. • `aws`, untuk integrasi dengan fungsi AWS Lambda atau AWS layanan lain, seperti Amazon DynamoDB, Amazon Simple Notificat ion Service, atau Amazon Simple Queue Service. • `mock`, untuk integrasi dengan API Gateway tanpa memanggil backend apa pun. Untuk informasi selengkap nya tentang tipe integrasi, lihat integration:type.
`tlsConfig`	the section called “x-amazon -apigateway-integration.tls Config”	Menentukan konfigurasi TLS untuk integrasi.

Nama properti	Tipe	Deskripsi
`uri`	`string`	URI endpoint dari backend. Untuk integrasi `aws` tipe, ini adalah nilai ARN. Untuk integrasi HTTP, ini adalah URL dari titik akhir HTTP termasuk `http` skema `https` or.

x-amazon-apigateway-integration contoh

Untuk API HTTP, Anda dapat menentukan integrasi di bagian komponen definisi OpenAPI Anda. Untuk mempelajari selengkapnya, lihat x-amazon-apigateway-integrations objek.

```
"x-amazon-apigateway-integration": {
    "$ref": "#/components/x-amazon-apigateway-integrations/integration1"
}
```

Contoh berikut membuat integrasi dengan fungsi Lambda. Untuk tujuan demonstrasi, contoh templat pemetaan yang ditunjukkan dalam `requestTemplates` dan `responseTemplates` contoh di bawah ini diasumsikan berlaku untuk muatan berformat JSON berikut: `{ "name":"value_1", "key":"value_2", "redirect": {"url" :"..."} }` untuk menghasilkan output JSON dari atau output XML. dari. `{ "stage":"value_1", "user-id":"value_2" }` `<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'"
            }
        }
    }
}
```

Perhatikan bahwa tanda kutip ganda (“) untuk string JSON dalam template pemetaan harus string-escaped (\”).

x-amazon-apigateway-integrations objek

Mendefinisikan kumpulan integrasi. Anda dapat menentukan integrasi di bagian komponen definisi OpenAPI Anda, dan menggunakan kembali integrasi untuk beberapa rute. Haya didukung untuk API HTTP.

Properti

Nama properti	Tipe	Deskripsi
integrasi	x-amazon-apigateway-integration objek	Kumpulan objek integrasi.

x-amazon-apigateway-integrations contoh

Contoh berikut membuat API HTTP yang mendefinisikan dua integrasi, dan referensi integrasi dengan menggunakan. `$ref": "#/components/x-amazon-apigateway-integrations/`*integration-name*

```
{
  "openapi": "3.0.1",
  "info":
    {
      "title": "Integrations",
      "description": "An API that reuses integrations",
      "version": "1.0"
    },
  "servers": [
  {
    "url": "https://example.com/{basePath}",
    "description": "The production API server",
    "variables":
      {
        "basePath":
          {
            "default": "example/path"
          }
      }
  }],
  "paths":
    {
```

```
      "/":
        {
          "get":
            {
              "x-amazon-apigateway-integration":
                {
                  "$ref": "#/components/x-amazon-apigateway-integrations/integration1"

                }
            }
        },
       "/pets":
        {
          "get":
            {
              "x-amazon-apigateway-integration":
                {
                  "$ref": "#/components/x-amazon-apigateway-integrations/integration1"

                }
            }
        },
       "/checkout":
        {
          "get":
            {
              "x-amazon-apigateway-integration":
                {
                  "$ref": "#/components/x-amazon-apigateway-integrations/integration2"
                }
            }
        }
    },
    "components": {
      "x-amazon-apigateway-integrations":
        {
          "integration1":
            {
              "type": "aws_proxy",
              "httpMethod": "POST",
              "uri": "arn:aws:apigateway:us-east-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-2:123456789012:function:my-function/invocations",
              "passthroughBehavior": "when_no_templates",
              "payloadFormatVersion": "1.0"
```

```
            },
          "integration2":
            {
              "type": "aws_proxy",
              "httpMethod": "POST",
              "uri": "arn:aws:apigateway:us-east-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-2:123456789012:function:example-function/invocations",
              "passthroughBehavior": "when_no_templates",
              "payloadFormatVersion" : "1.0"
            }
        }
    }
}
```

x-amazon-apigateway-integrationObjek. RequestTemplates

Menentukan template pemetaan untuk payload permintaan dari jenis MIME tertentu.

Properti

Nama properti	Tipe	Deskripsi
MIME type	`string`	Contoh tipe MIME adalah`application/json`. Untuk informasi tentang membuat template pemetaan, lihatPetStore template pemetaan.

x-amazon-apigateway-integrationContoh. RequestTemplates

Contoh berikut menetapkan template pemetaan untuk payload permintaan tipe `application/json` dan `application/xml` MIME.

```
"requestTemplates" : {
    "application/json" : "#set ($root=$input.path('$')) { \"stage\": \"$root.name\",
 \"user-id\": \"$root.key\" }",
```

```
    "application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</stage> "
}
```

x-amazon-apigateway-integrationObjek. RequestParameters

Untuk REST API, tentukan pemetaan dari parameter permintaan metode bernama ke parameter permintaan integrasi. Parameter permintaan metode harus didefinisikan sebelum direferensikan.

Untuk API HTTP, menentukan parameter yang diteruskan ke `AWS_PROXY` integrasi dengan yang ditentukan. `integrationSubtype`

Properti

Nama properti	Tipe	Deskripsi
`integration.request.` *<param-type>* `.` *<param-name>*	`string`	Untuk REST API, nilainya biasanya merupakan parameter permintaan metode yang telah ditentukan sebelumnya dari `method.request.` *<param-type>* `.` *<param-name>* format, di mana `<param-type>` bisa `querystring`, `pathheader`, atau `body`. Namun, `$context.` *VARIABLE_NAME*, `$stageVariables.` *VARIABLE_NAME*, dan *STATIC_VALUE* juga valid. Untuk body parameter, `<param-name>` adalah ekspresi jalur JSON tanpa `$.` awalan.
parameter	`string`	Untuk API HTTP, parameter permintaan adalah peta nilai

Nama properti	Tipe	Deskripsi
		kunci yang menentukan parameter yang diteruskan ke `AWS_PROXY` integrasi dengan yang ditentukan. `integrationSubtype` Anda dapat memberikan nilai statis, atau data permintaan peta, variabel tahap, atau variabel konteks yang dievaluasi pada saat runtime. Untuk mempelajari selengkapnya, lihat the section called “AWSintegrasi layanan”.

Contoh `x-amazon-apigateway-integration.requestParameters`

Contoh pemetaan parameter permintaan berikut menerjemahkan parameter query (`version`), header (), dan path (`x-user-idservice`) permintaan metode ke query permintaan integrasi (`stage`), header (), dan parameter jalur (`x-userid`), masing-masingop.

> **Note**
>
> Jika Anda membuat sumber daya melalui OpenAPI atau AWS CloudFormation, nilai statis harus disertakan dalam tanda kutip tunggal.
> Untuk menambahkan nilai ini dari konsol, masukkan `application/json` di dalam kotak, tanpa tanda kutip.

```
"requestParameters" : {
    "integration.request.querystring.stage" : "method.request.querystring.version",
    "integration.request.header.x-userid" : "method.request.header.x-user-id",
    "integration.request.path.op" : "method.request.path.service"
},
```

x-amazon-apigateway-integration.response objek

Mendefinisikan respons metode dan menentukan pemetaan parameter atau pemetaan muatan dari respons integrasi ke respons metode.

Properti

Nama properti	Tipe	Deskripsi
`Pola status respons`	x-amazon-apigateway-integration.response objek	Baik ekspresi reguler yang digunakan untuk mencocokkan respons integrasi dengan respons metode, atau `default` untuk menangkap respons apa pun yang belum Anda konfigurasikan. Untuk integrasi HTTP, regex berlaku untuk kode status respons integrasi. Untuk pemanggilan Lambda, regex berlaku untuk `errorMessage` bidang objek informasi kesalahan yang dikembalikan oleh AWS Lambda sebagai badan respons kegagalan saat eksekusi fungsi Lambda melempar pengecualian. **Note** Nama properti *`pola status respons`* mengacu pada kode status respons atau ekspresi reguler yang menjelaskan sekelompok kode status respons. Itu

Nama properti	Tipe	Deskripsi
		tidak sesuai dengan pengenal IntegrationResponse sumber daya apa pun di API Gateway REST API.

Contoh **x-amazon-apigateway-integration.responses**

Contoh berikut menunjukkan daftar tanggapan dari `2xx` dan `302` tanggapan. Untuk responsnya, `2xx` respons metode dipetakan dari muatan respons integrasi tipe MIME `application/json` atau `application/xml` MIME. Respons ini menggunakan template pemetaan yang disediakan. Untuk `302` respon, respon metode mengembalikan `Location` header yang nilainya berasal dari `redirect.url` properti pada payload respon integrasi.

```
"responses" : {
    "2\\d{2}" : {
        "statusCode" : "200",
        "responseTemplates" : {
            "application/json" : "#set ($root=$input.path('$')) { \"stage\":
 \"$root.name\", \"user-id\": \"$root.key\" }",
            "application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</
stage> "
        }
    },
    "302" : {
        "statusCode" : "302",
        "responseParameters" : {
            "method.response.header.Location": "integration.response.body.redirect.url"
        }
    }
}
```

x-amazon-apigateway-integration.response objek

Mendefinisikan respons dan menentukan pemetaan parameter atau pemetaan muatan dari respons integrasi ke respons metode.

Properti

Nama properti	Tipe	Deskripsi
`statusCode`	`string`	Kode status HTTP untuk respons metode; misalnya, `"200"`. Ini harus sesuai dengan respons yang cocok di bidang Operasi `responses` OpenAPI.
`responseTemplates`	x-amazon-apigateway-integrationObjek. ResponseTemplates	Menentukan templat pemetaan khusus tipe MIME untuk muatan respons.
`responseParameters`	x-amazon-apigateway-integration.ResponseParameters objek	Menentukan pemetaan parameter untuk respon. Hanya `body` parameter `header` dan respons integrasi yang dapat dipetakan ke `header` parameter metode.
`contentHandling`	`string`	Jenis konversi pengkodean muatan respons. Nilai yang valid adalah 1)`CONVERT_TO_TEXT` , untuk mengubah muatan biner menjadi string yang dikodekan base64 atau mengubah muatan teks menjadi string yang `utf-8` dikodekan atau melewati muatan teks secara asli tanpa modifikasi, dan

Nama properti	Tipe	Deskripsi
		2)CONVERT_TO_BINARY , untuk mengubah muatan teks menjadi gumpalan yang diterjemahkan base64 atau melewati muatan biner secara asli tanpa modifikasi.

Contoh **x-amazon-apigateway-integration.response**

Contoh berikut mendefinisikan `302` respons untuk metode yang memperoleh muatan tipe `application/json` atau `application/xml` MIME dari backend. Respons menggunakan templat pemetaan yang disediakan dan mengembalikan URL pengalihan dari respons integrasi di header metode. `Location`

```
{
    "statusCode" : "302",
    "responseTemplates" : {
         "application/json" : "#set ($root=$input.path('$')) { \"stage\": \"$root.name
\", \"user-id\": \"$root.key\" }",
         "application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</stage> "
    },
    "responseParameters" : {
        "method.response.header.Location": "integration.response.body.redirect.url"
    }
}
```

x-amazon-apigateway-integrationObjek. ResponseTemplates

Menentukan template pemetaan untuk payload respon dari jenis MIME tertentu.

Properti

Nama properti	Tipe	Deskripsi
MIME type	`string`	Menentukan template pemetaan untuk mengubah badan respons integrasi ke badan respons metode untuk tipe MIME tertentu. Untuk informasi tentang membuat template pemetaan, lihatPetStore template pemetaan. Contoh *tipe MIME* adalah`application/json`.

x-amazon-apigateway-integrationContoh .responseTemplate

Contoh berikut menetapkan template pemetaan untuk payload permintaan tipe `application/json` dan `application/xml` MIME.

```
"responseTemplates" : {
    "application/json" : "#set ($root=$input.path('$')) { \"stage\": \"$root.name\",
 \"user-id\": \"$root.key\" }",
    "application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</stage> "
}
```

x-amazon-apigateway-integration.ResponseParameters objek

Menentukan pemetaan dari parameter respon metode integrasi untuk parameter respon metode. Anda dapat memetakan `header`body,, atau nilai statis ke `header` jenis respons metode. Hanya didukung untuk REST API.

Properti

Nama properti	Tipe	Deskripsi
`method.response.header. <param-name>`	`string`	Nilai parameter bernama dapat diturunkan dari `header` dan `body` jenis parameter respons integrasi.

Contoh `x-amazon-apigateway-integration.responseParameters`

Berikut contoh peta `body` dan `header` parameter dari respon integrasi untuk dua `header` parameter dari respon metode.

```
"responseParameters" : {
    "method.response.header.Location" : "integration.response.body.redirect.url",
    "method.response.header.x-user-id" : "integration.response.header.x-userid"
}
```

x-amazon-apigateway-integrationObjek .tlsConfig

Menentukan konfigurasi TLS untuk integrasi.

Properti

Nama properti	Tipe	Deskripsi
`insecureSkipVerification`	`Boolean`	Hanya didukung untuk REST API. Menentukan apakah API Gateway melewatkan verifikasi bahwa sertifikat untuk titik akhir integrasi dikeluarkan oleh otoritas sertifikat yang didukung. Ini tidak disarankan, tetapi memungkinkan Anda untuk menggunakan sertifika

Nama properti	Tipe	Deskripsi
		t yang ditandatangani oleh otoritas sertifikat swasta, atau sertifikat yang ditandatangani sendiri. Jika diaktifkan, API Gateway masih melakukan validasi sertifikat dasar, termasuk memeriksa tanggal kedaluwarsa sertifikat, nama host, dan keberadaan otoritas sertifikat root. Sertifikat root milik otoritas swasta harus memenuhi kendala berikut: • ekstensi x509 `keyUsage` harus memiliki. `keyCertSign` • ekstensi x509 `basicConstraints` harus memiliki. `CA:TRUE` Didukung hanya untuk `HTTP` dan `HTTP_PROXY` integrasi. ⚠ Warning Pengaktifan `insecureSkipVerification` tidak disarankan, terutama untuk integrasi dengan titik akhir HTTPS publik. Jika Anda mengaktifkan `insecureSkipVerifi`

Nama properti	Tipe	Deskripsi
		`cation` , Anda meningkatkan risiko man-in-the-middle serangan.
`serverNameToVerify`	`string`	Hanya didukung untuk integrasi pribadi HTTP API. Jika Anda menentukan nama server, API Gateway menggunakannya untuk memverifikasi nama host pada sertifikat integrasi. Nama server juga termasuk dalam jabat tangan TLS untuk mendukung Indikasi Nama Server (SNI) atau hosting virtual.

x-amazon-apigateway-integrationContoh .tlsConfig

Contoh OpenAPI 3.0 berikut memungkinkan integrasi `insecureSkipVerification` proxy HTTP REST API.

```
"x-amazon-apigateway-integration": {
  "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
  "responses": {
     default": {
       "statusCode": "200"
      }
  },
  "passthroughBehavior": "when_no_match",
  "httpMethod": "ANY",
  "tlsConfig" : {
    "insecureSkipVerification" : true
  }
  "type": "http_proxy",
```

```
}
```

Contoh OpenAPI 3.0 berikut menentukan `serverNameToVerify` untuk integrasi pribadi HTTP API.

```
"x-amazon-apigateway-integration" : {
  "payloadFormatVersion" : "1.0",
  "connectionId" : "abc123",
  "type" : "http_proxy",
  "httpMethod" : "ANY",
  "uri" : "arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/app/my-load-
balancer/50dc6c495c0c9188/0467ef3c8400ae65",
  "connectionType" : "VPC_LINK",
  "tlsConfig" : {
     "serverNameToVerify" : "example.com"
  }
}
```

x-amazon-apigateway-minimum-ukuran kompresi

Menentukan ukuran kompresi minimum untuk REST API. Untuk mengaktifkan kompresi, tentukan bilangan bulat antara 0 dan 10485760. Untuk mempelajari selengkapnya, lihat Mengaktifkan kompresi payload untuk API.

x-amazon-apigateway-minimum-contoh ukuran kompresi

Contoh berikut menentukan ukuran kompresi minimum `5242880` byte untuk REST API.

```
"x-amazon-apigateway-minimum-compression-size": 5242880
```

x-amazon-apigateway-policy

Menentukan kebijakan sumber daya untuk REST API. Untuk mempelajari lebih lanjut tentang kebijakan sumber daya, lihatMengontrol akses ke API dengan kebijakan sumber daya API Gateway. Untuk contoh kebijakan sumber daya, lihatContoh kebijakan sumber daya API Gateway.

Contoh **`x-amazon-apigateway-policy`**

Contoh berikut menentukan kebijakan sumber daya untuk REST API. Kebijakan sumber daya menolak (memblokir) lalu lintas masuk ke API dari blok alamat IP sumber tertentu. Saat diimpor,

`"execute-api:/*"` dikonversi `kearn:aws:execute-api:`*region*`:`*account-id*`:`*api-id*`/*`, menggunakan Wilayah saat ini, ID AWS akun Anda, dan ID REST API saat ini.

```
"x-amazon-apigateway-policy": {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
               "execute-api:/*"
            ],
            "Condition" : {
                "IpAddress": {
                    "aws:SourceIp": "192.0.2.0/24"
                }
            }
        }
    ]
}
```

x-amazon-apigateway-requestproperti -validator

Menentukan validator permintaan, dengan mereferensikan x-amazon-apigateway-request-validator objek peta, untuk mengaktifkan validasi permintaan pada API yang mengandung atau metode. *request_validator_name* Nilai ekstensi ini adalah string JSON.

Ekstensi ini dapat ditentukan pada tingkat API atau di tingkat metode. Validator API-level berlaku untuk semua metode kecuali jika diganti oleh validator tingkat metode.

Contoh `x-amazon-apigateway-request-validator`

Contoh berikut menerapkan validator `basic` permintaan di API level saat menerapkan validator `parameter-only` permintaan pada permintaan. `POST /validation`

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "x-amazon-apigateway-request-validators" : {
    "basic" : {
      "validateRequestBody" : true,
      "validateRequestParameters" : true
    },
    "params-only" : {
      "validateRequestBody" : false,
      "validateRequestParameters" : true
    }
  },
  "x-amazon-apigateway-request-validator" : "basic",
  "paths": {
    "/validation": {
      "post": {
        "x-amazon-apigateway-request-validator" : "params-only",
       ...
     }
}
```

x-amazon-apigateway-request-validator objek

Mendefinisikan validator permintaan yang didukung untuk API yang berisi sebagai peta antara nama validator dan aturan validasi permintaan terkait. Ekstensi ini berlaku untuk REST API.

Properti

Nama properti	Tipe	Deskripsi
`request_validator_name`	x-amazon-apigateway-request-Validators.requestValidator objek	Menentukan aturan validasi yang terdiri dari validator bernama. Sebagai contoh:

Nama properti	Tipe	Deskripsi

		<pre>"basic" : {
		"validate
		RequestBody" : true,
		"validate
		RequestParameters" :
		true
		},</pre>
		Untuk menerapkan validator ini ke metode tertentu, referensi nama validator (`basic`) sebagai nilai properti. x-amazon-apigateway-request properti -validator

Contoh `x-amazon-apigateway-request-validators`

Contoh berikut menunjukkan sekumpulan validator permintaan untuk API sebagai peta antara nama validator dan aturan validasi permintaan terkait.

OpenAPI 2.0

```
{
  "swagger": "2.0",
  ...
  "x-amazon-apigateway-request-validators" : {
    "basic" : {
      "validateRequestBody" : true,
      "validateRequestParameters" : true
    },
    "params-only" : {
      "validateRequestBody" : false,
      "validateRequestParameters" : true
    }
  },
  ...
}
```

x-amazon-apigateway-request-Validators.requestValidator objek

Menentukan aturan validasi validator permintaan sebagai bagian dari definisi peta. x-amazon-apigateway-request-validator objek

Properti

Nama properti	Tipe	Deskripsi
`validateRequestBody`	`Boolean`	Menentukan apakah untuk memvalidasi badan permintaan (`true`) atau tidak (`false`).
`validateRequestParameters`	`Boolean`	Menentukan apakah untuk memvalidasi parameter permintaan yang diperlukan (`true`) atau tidak (`false`).

Contoh **`x-amazon-apigateway-request-validators.requestValidator`**

Contoh berikut menunjukkan validator permintaan parameter-only:

```
"params-only": {
    "validateRequestBody" : false,
    "validateRequestParameters" : true
}
```

x-amazon-apigateway-tag-nilai properti

Menentukan nilai AWS tag untuk HTTP API. Anda dapat menggunakan `x-amazon-apigateway-tag-value` properti sebagai bagian dari objek tag OpenAPI tingkat root untuk menentukan AWS tag untuk API HTTP. Jika Anda menentukan nama tag tanpa `x-amazon-apigateway-tag-value` properti, API Gateway akan membuat tag dengan string kosong untuk nilai.

Untuk mempelajari lebih lanjut tentang penandaan, lihatMemberi tag API Gateway daya.

Properti

Nama properti	Tipe	Deskripsi
`name`	`String`	Menentukan kunci tag.
`x-amazon-apigateway-tag-value`	`String`	Menentukan nilai tag.

Contoh **`x-amazon-apigateway-tag-value`**

Contoh berikut menentukan dua tag untuk HTTP API:

- “Pemilik”: “Admin”
- “Prod”: “”

```
"tags": [
    {
      "name": "Owner",
      "x-amazon-apigateway-tag-value": "Admin"
    },
    {
      "name": "Prod"
    }
]
```

Keamanan di Amazon API Gateway

Keamanan cloud di AWS merupakan prioritas tertinggi. Sebagai pelanggan AWS, Anda akan mendapatkan manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara AWS dan Anda. Model tanggung jawab bersama model menggambarkan ini sebagai keamanan cloud dan keamanan di cloud:

- Keamanan cloud – AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan layanan-layanan AWS di dalam AWS Cloud. AWS juga memberikan Anda layanan yang dapat digunakan dengan aman. Auditor pihak ketiga melakukan pengujian dan verifikasi secara berkala terhadap efektivitas keamanan kami sebagai bagian dari Program Kepatuhan AWS. Untuk mempelajari tentang program kepatuhan yang berlaku untuk Amazon API Gateway, lihat AWSlayanan dalam cakupan berdasarkan AWS layanan program kepatuhan .
- Keamanan di cloud – Tanggung jawab Anda ditentukan menurut layanan AWS yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain termasuk sensitivitas data Anda, persyaratan perusahaan Anda, serta hukum dan peraturan yang berlaku.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan API Gateway. Topik berikut menunjukkan cara mengonfigurasi API Gateway untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga mempelajari cara menggunakan AWS layanan lain yang membantu Anda memantau dan mengamankan sumber daya API Gateway Anda.

Untuk informasi selengkapnya, lihat Gambaran Umum Keamanan Amazon API Gateway.

Topik

- Praktik terbaik keamanan di Amazon API Gateway

Perlindungan data di Amazon API Gateway

Model tanggung jawab AWS bersama model tanggung berlaku untuk perlindungan data di Amazon API Gateway. Sebagaimana diuraikan dalam model ini, AWS bertanggung jawab untuk memberikan perlindungan terhadap infrastruktur global yang menjalankan semua AWS Cloud. Anda harus bertanggung jawab untuk memelihara kendali terhadap konten yang di-hosting pada infrastruktur ini. Anda juga bertanggung jawab atas konfigurasi keamanan dan tugas manajemen untuk Layanan AWS yang Anda gunakan. Untuk informasi lebih lanjut tentang privasi data, lihat FAQ tentang Privasi Data. Untuk informasi tentang perlindungan data di Eropa, lihat postingan blog Model Tanggung Jawab Bersama AWS dan GDPR di Blog Keamanan AWS.

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensyal dan mengatur pengguna individu dengan AWS IAM Identity Center atau AWS Identity and Access Management (IAM). Dengan cara tersebut, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tugas pekerjaan mereka. Kami juga merekomendasikan agar Anda mengamankan data Anda dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk melakukan komunikasi dengan sumber daya AWS. Kami membutuhkan TLS 1.2 dan merekomendasikan TLS 1.3.
- Siapkan API dan log aktivitas pengguna dengan AWS CloudTrail.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnyaLayanan AWS.
- Gunakan layanan keamanan terkelola tingkat lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-2 ketika mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Untuk informasi lebih lanjut tentang titik akhir FIPS yang tersedia, lihat Standar Pemrosesan Informasi Federal (FIPS) 140-2.

Kami sangat menyarankan agar Anda tidak pernah memasukkan informasi rahasia atau sensitif, seperti alamat email pelanggan Anda, ke dalam tag atau bidang teks bentuk bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan API Gateway atau lainnya Layanan AWS menggunakan konsol, APIAWS CLI, atau AWS SDK. Data apa pun yang Anda masukkan ke dalam tag atau bidang teks bentuk bebas yang digunakan untuk nama dapat digunakan untuk

penagihan atau log diagnostik. Saat Anda memberikan URL ke server eksternal, sebaiknya Anda tidak menyertakan informasi kredensial di URL untuk memvalidasi permintaan Anda ke server tersebut.

Enkripsi data di Amazon API Gateway

Perlindungan data mengacu pada melindungi data saat dalam perjalanan (saat bepergian ke dan dari API Gateway) dan saat istirahat (saat disimpan dalamAWS).

Enkripsi data saat istirahat di Amazon API Gateway

Jika Anda memilih untuk mengaktifkan caching untuk REST API, Anda dapat mengaktifkan enkripsi cache. Untuk mempelajari selengkapnya, lihat Mengaktifkan caching API untuk meningkatkan daya tanggap.

Untuk informasi selengkapnya tentang perlindungan data, lihat postingan blog Model Tanggung Jawab Bersama AWS dan GDPR di Blog Keamanan AWS.

Enkripsi data dalam perjalanan di Amazon API Gateway

API yang dibuat dengan Amazon API Gateway hanya mengekspos endpoint HTTPS. API Gateway tidak mendukung endpoint (HTTP) yang tidak terenkripsi.

API Gateway mengelola sertifikat untuk titik `execute-api` akhir default. Jika Anda mengkonfigurasi nama domain kustom, Anda menentukan sertifikat untuk nama domain. Sebagai praktik terbaik, jangan menyematkan sertifikat.

Untuk keamanan yang lebih baik, Anda dapat memilih versi protokol Transport Layer Security (TLS) minimum yang akan diberlakukan untuk domain kustom API Gateway Anda. WebSocketAPI dan API HTTP hanya mendukung TLS 1.2. Untuk mempelajari selengkapnya, lihat Memilih kebijakan keamanan untuk domain kustom Anda di API Gateway.

Anda juga dapat mengatur CloudFront distribusi Amazon dengan sertifikat SSL khusus di akun Anda dan menggunakannya dengan API Regional. Anda kemudian dapat mengonfigurasi kebijakan keamanan untuk CloudFront distribusi dengan TLS 1.1 atau yang lebih tinggi berdasarkan persyaratan keamanan dan kepatuhan Anda.

Untuk informasi selengkapnya tentang perlindungan data, lihat Melindungi REST API dan Model Tanggung Jawab AWS Bersama dan posting blog GDPR di Blog AWSKeamanan.

Privasi lalu lintas jaringan Internet

Dengan Amazon API Gateway, Anda dapat membuat API REST privat yang dapat diakses hanya dari Amazon Virtual Private Cloud (VPC) Anda. VPC menggunakanTitik akhir VPC antarmuka, yang merupakan antarmuka jaringan endpoint yang Anda buat di VPC Anda. Menggunakankebijakan sumber daya, Anda dapat mengizinkan atau menolak akses ke API Anda dari VPC yang dipilih dan endpoint VPC, termasuk di seluruhAWSakun. Setiap endpoint dapat digunakan untuk mengakses beberapa API pribadi. Anda juga dapat menggunakanAWS Direct Connectuntuk membuat koneksi dari jaringan lokal ke Amazon VPC dan mengakses API pribadi Anda melalui koneksi tersebut. Dalam semua kasus, lalu lintas ke API pribadi Anda menggunakan koneksi aman dan tidak meninggalkan jaringan Amazon; itu terisolasi dari internet publik. Untuk mempelajari informasi lebih lanjut, lihat the section called “API Privat”.

Manajemen identitas dan akses untuk Amazon API Gateway

AWS Identity and Access Management(IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. Administrator IAM mengontrol siapa yang dapat diautentikasi (masuk) dan diberi wewenang (memiliki izin) untuk menggunakan sumber daya API Gateway. IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

Penonton

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan di API Gateway.

Pengguna layanan — Jika Anda menggunakan layanan API Gateway untuk melakukan pekerjaan Anda, administrator Anda memberi Anda kredensi dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak fitur API Gateway untuk melakukan pekerjaan Anda, Anda mungkin memerlukan izin tambahan. Memahami cara akses dikelola dapat membantu Anda untuk meminta izin yang tepat dari administrator Anda. Jika Anda tidak dapat mengakses fitur di API Gateway, lihatMemecahkan masalah identitas dan akses Amazon API Gateway.

Administrator layanan - Jika Anda bertanggung jawab atas sumber daya API Gateway di perusahaan Anda, Anda mungkin memiliki akses penuh ke API Gateway. Tugas Anda adalah menentukan fitur dan sumber daya API Gateway mana yang harus diakses pengguna layanan Anda. Anda kemudian harus mengirimkan permintaan ke administrator IAM Anda untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep dasar IAM. Untuk mempelajari selengkapnya tentang cara perusahaan Anda dapat menggunakan IAM dengan API Gateway, lihatCara kerja Amazon API Gateway dengan IAM.

Administrator IAM - Jika Anda administrator IAM, Anda mungkin ingin mempelajari detail tentang cara menulis kebijakan untuk mengelola akses ke API Gateway. Untuk melihat contoh kebijakan berbasis identitas API Gateway yang dapat Anda gunakan di IAM, lihat. Contoh kebijakan berbasis identitas Amazon API Gateway

Mengautentikasi dengan identitas

Autentikasi adalah cara Anda untuk masuk ke AWS menggunakan kredensial identitas Anda. Anda harus diautentikasi (masuk keAWS) sebagaiPengguna root akun AWS, sebagai pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk AWS sebagai identitas federasi dengan menggunakan kredensil yang disediakan melalui sumber identitas. AWS IAM Identity Center Pengguna (IAM Identity Center), autentikasi masuk tunggal perusahaan Anda, dan kredensi Google atau Facebook Anda adalah contoh identitas federasi. Saat Anda masuk sebagai identitas federasi, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan peran IAM. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil peran.

Bergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal AWS akses. Untuk informasi selengkapnya tentang masukAWS, lihat Cara masuk ke Panduan AWS Sign-In Pengguna Anda Akun AWS.

Jika Anda mengakses AWS secara terprogram, AWS sediakan kit pengembangan perangkat lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara

kriptografis dengan menggunakan kredensil Anda. Jika Anda tidak menggunakan AWS alat, Anda harus menandatangani permintaan sendiri. Untuk informasi selengkapnya tentang penggunaan metode yang disarankan untuk menandatangani permintaan sendiri, lihat Menandatangani permintaan AWS API di Panduan Pengguna IAM.

Terlepas dari metode otentikasi yang Anda gunakan, Anda mungkin diminta untuk memberikan informasi keamanan tambahan. Misalnya, AWS menyarankan supaya Anda menggunakan autentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari lebih lanjut, lihat Autentikasi multi-faktor di Panduan AWS IAM Identity Center Pengguna dan Menggunakan otentikasi multi-faktor (MFA) AWS di Panduan Pengguna IAM.

Pengguna root Akun AWS

Saat Anda membuatAkun AWS, Anda mulai dengan satu identitas masuk yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya di akun. Identitas ini disebut pengguna root Akun AWS dan diakses dengan cara masuk menggunakan alamat email dan kata sandi yang Anda gunakan saat membuat akun. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari Anda. Lindungi kredensil pengguna root Anda dan gunakan untuk melakukan tugas-tugas yang hanya dapat dilakukan oleh pengguna root. Untuk daftar lengkap tugas yang mengharuskan Anda masuk sebagai pengguna root, lihat Tugas yang memerlukan kredensi pengguna root di Panduan Pengguna IAM.

Pengguna dan grup IAM

Pengguna IAM adalah identitas dalam akun Akun AWS Anda yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, sebaiknya mengandalkan kredensi sementara daripada membuat pengguna IAM yang memiliki kredensi jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan khusus yang memerlukan kredensil jangka panjang dengan pengguna IAM, kami sarankan Anda memutar kunci akses. Untuk informasi selengkapnya, lihat Memutar kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensil jangka panjang di Panduan Pengguna IAM.

Grup IAM adalah identitas yang menentukan kumpulan dari para pengguna IAM. Anda tidak dapat masuk sebagai kelompok. Anda dapat menggunakan grup untuk menentukan izin untuk beberapa pengguna sekaligus. Grup membuat izin lebih mudah dikelola untuk set besar pengguna. Misalnya, Anda dapat memiliki grup yang diberi nama IAMAdmins dan memberikan izin kepada grup tersebut untuk mengelola sumber daya IAM.

Para pengguna berbeda dari peran. Seorang pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran ini dimaksudkan untuk dapat digunakan oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari selengkapnya, lihat Kapan harus membuat pengguna IAM (bukan peran) dalam Panduan Pengguna IAM.

IAM role

IAM role adalah identitas dalam akun Akun AWS Anda yang memiliki izin khusus. Peran ini mirip dengan pengguna IAM, tetapi tidak terkait dengan orang tertentu. Anda dapat menggunakan IAM role untuk sementara di dalam AWS Management Console dengan cara berganti peran. Anda dapat mengambil peran dengan cara memanggil operasi API AWS CLI atau AWS atau menggunakan URL khusus. Untuk informasi selengkapnya tentang metode penggunaan peran, lihat Menggunakan IAM roles dalam Panduan Pengguna IAM.

IAM role dengan kredensial sementara berguna dalam situasi berikut:

- Akses pengguna terfederasi — Untuk menetapkan izin ke identitas federasi, Anda membuat peran dan menentukan izin untuk peran tersebut. Ketika identitas federasi mengautentikasi, identitas dikaitkan dengan peran dan diberikan izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat Membuat peran untuk Penyedia Identitas pihak ketiga di Panduan Pengguna IAM. Jika Anda menggunakan Pusat Identitas IAM, Anda mengonfigurasi set izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah diautentikasi, IAM Identity Center mengkorelasikan izin yang disetel ke peran dalam IAM. Untuk informasi tentang set izin, lihat Set izin di Panduan AWS IAM Identity Center Pengguna.
- Izin pengguna IAM sementara — Pengguna atau peran IAM dapat mengambil peran IAM untuk sementara mengambil izin yang berbeda untuk tugas tertentu.
- Akses lintas akun – Anda dapat menggunakan IAM role agar seseorang (principal tepercaya) di akun lain diizinkan untuk mengakses sumber daya di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, dengan beberapaLayanan AWS, Anda dapat melampirkan kebijakan secara langsung ke sumber daya (alih-alih menggunakan peran sebagai proxy). Untuk mempelajari perbedaan antara kebijakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Perbedaan antara IAM role dan kebijakan berbasis sumber daya dalam Panduan Pengguna IAM.
- Akses lintas layanan — Beberapa Layanan AWS menggunakan fitur lainLayanan AWS. Sebagai conoth, ketika Anda melakukan panggilan dalam suatu layanan, biasanya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Layanan mungkin

melakukan ini menggunakan izin panggilan principal, menggunakan peran layanan, atau peran tertaut layanan.

- Sesi akses teruskan (FAS) — Saat Anda menggunakan pengguna IAM atau peran untuk melakukan tindakanAWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan tindakan yang kemudian memulai tindakan lain di layanan yang berbeda. FAS menggunakan izin dari pemanggilan utamaLayanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain Layanan AWS atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan saat membuat permintaan FAS, lihat Meneruskan sesi akses.
- Peran layanan – Peran layanan adalah IAM role yang diambil oleh layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, memodifikasi, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan Pengguna IAM.
- Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke peran layanan. Layanan AWS Layanan dapat menggunakan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

- Aplikasi yang berjalan di Amazon EC2 – Anda dapat menggunakan IAM role untuk mengelola kredensial sementara untuk aplikasi yang berjalan pada instans EC2, dan membuat permintaan API AWS CLI atau AWS. Menyimpan access key di dalam instans EC2 lebih disarankan. Untuk menugaskan sebuah peran AWS ke instans EC2 dan membuatnya tersedia untuk semua aplikasinya, Anda dapat membuat sebuah profil instans yang dilampirkan ke instans. Profil instans berisi peran dan memungkinkan program yang berjalan di instans EC2 untuk mendapatkan kredensial sementara. Untuk informasi selengkapnya, lihat Menggunakan IAM role untuk memberikan izin ke aplikasi yang berjalan di instans Amazon EC2 dalam Panduan Pengguna IAM.

Untuk mempelajari kapan waktunya menggunakan IAM role atau pengguna IAM, lihat Kapan harus membuat IAM role (bukan pengguna) dalam Panduan Pengguna IAM.

Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan adalah objek AWS yang, ketika dikaitkan dengan identitas atau sumber daya, menentukan izinnya. AWSmengevaluasi kebijakan ini ketika prinsipal (pengguna, pengguna

root, atau sesi peran) membuat permintaan. Izin dalam kebijakan dapat menentukan permintaan yang diizinkan atau ditolak. Sebagian besar kebijakan disimpan di AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang struktur dan isi dokumen kebijakan JSON, lihat Gambaran Umum kebijakan JSON dalam Panduan Pengguna IAM.

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan siapa yang memiliki akses ke hal apa. Yaitu, principal mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam syarat apa.

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan pada sumber daya yang mereka butuhkan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat mengambil peran.

Kebijakan IAM mendefinisikan izin untuk suatu tindakan terlepas dari metode yang Anda gunakan untuk pengoperasiannya. Misalnya, Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan tersebut dapat memperoleh informasi peran dari API AWS Management Console, the AWS CLI, or the AWS.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke identitas, seperti pengguna IAM, grup pengguna, atau peran. Kebijakan ini mengontrol tipe tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan dalam syarat. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan terkelola. Kebijakan inline disematkan secara langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat Anda lampirkan ke beberapa pengguna, grup, dan peran dalam akun Akun AWS Anda. Kebijakan terkelola meliputi kebijakan yang dikelola AWS dan kebijakan yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan terkelola atau kebijakan inline, lihat Memilih antara kebijakan terkelola dan kebijakan inline dalam Panduan Pengguna IAM.

Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan tepercaya IAM role dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator

layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya yang dilampiri kebijakan tersebut, kebijakan menetapkan tindakan apa yang dapat dilakukan oleh principal tertentu di sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan principal dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan yang dikelola AWS dari IAM dalam kebijakan berbasis sumber daya.

Daftar kontrol akses (ACL)

Daftar kontrol akses (ACL) mengendalikan principal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL serupa dengan kebijakan berbasis sumber daya, meskipun tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung ACL. Untuk mempelajari ACL selengkapnya, lihat Gambaran umum daftar kontrol akses (ACL) dalam Panduan Developer Amazon Simple Storage Service.

Tipe kebijakan lainnya

AWS mendukung tipe kebijakan tambahan, yang kurang umum. Tipe kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda oleh tipe kebijakan yang lebih umum.

- Batasan izin – Batasan izin adalah fitur lanjutan tempat Anda mengatur izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM (pengguna atau IAM role). Anda dapat menetapkan batas izin untuk suatu entitas. Izin yang dihasilkan adalah persimpangan kebijakan berbasis identitas entitas dan batas izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan eksplisit dalam salah satu kebijakan ini dapat membatalkan izin. Untuk informasi selengkapnya tentang batasan izin, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.
- Kebijakan Kontrol Layanan (SCPs) – SCP adalah kebijakan JSON yang menentukan izin maksimum untuk sebuah organisasi atau unit organisasi (OU) di AWS Organizations. AWS Organizations adalah layanan untuk mengelompokkan dan mengelola beberapa akun Akun AWS secara terpusat yang dimiliki oleh bisnis Anda. Jika Anda mengaktifkan semua fitur di sebuah organisasi, maka Anda dapat menerapkan kebijakan kontrol layanan (SCP) ke salah satu atau ke semua akun Anda. SCP membatasi izin untuk entitas dalam akun anggota, termasuk setiap

Pengguna root akun AWS. Untuk informasi selengkapnya tentang Organizations dan SCP, lihat Cara kerja SCP dalam Panduan Pengguna AWS Organizations.

- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda berikan sebagai parameter saat Anda membuat sesi sementara secara terprogram bagi peran atau pengguna gabungan. Izin sesi yang dihasilkan adalah persimpangan kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga dapat berasal dari kebijakan berbasis sumber daya. Penolakan eksplisit dalam salah satu kebijakan ini membatalkan izin. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Berbagai tipe kebijakan

Ketika beberapa tipe kebijakan berlaku untuk sebuah permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan untuk mengizinkan permintaan ketika beberapa tipe kebijakan dilibatkan, lihat Logika evaluasi kebijakan dalam Panduan Pengguna IAM.

Cara kerja Amazon API Gateway dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke API Gateway, Anda harus memahami fitur IAM apa yang tersedia untuk digunakan dengan API Gateway. Untuk mendapatkan tampilan tingkat tinggi tentang cara kerja API Gateway dan AWS layanan lainnya dengan IAM, lihat AWSLayanan yang Bekerja dengan IAM di Panduan Pengguna IAM.

Topik

- Kebijakan berbasis identitas API Gateway
- Kebijakan berbasis sumber daya API Gateway
- Otorisasi berdasarkan tag API Gateway
- Peran IAM API Gateway

Kebijakan berbasis identitas API Gateway

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan tindakan dan sumber daya mana yang diizinkan atau ditolak serta kondisi di mana tindakan diizinkan atau ditolak. API Gateway mendukung tindakan, sumber daya, dan kunci kondisi tertentu. Untuk informasi selengkapnya tentang tindakan, sumber daya, dan kunci kondisi khusus Gateway API, lihat Kunci tindakan, sumber daya, dan kondisi untuk Pengelolaan dan Tindakan Amazon API Gateway, sumber daya, dan kunci kondisi untuk Amazon API Gateway Management V2. Untuk informasi tentang semua elemen yang

Anda gunakan dalam kebijakan JSON, lihat Referensi Elemen Kebijakan IAM JSON di Panduan Pengguna IAM.

Contoh berikut menunjukkan kebijakan berbasis identitas yang memungkinkan pengguna untuk membuat atau memperbarui hanya API REST pribadi. Untuk contoh lainnya, lihat the section called "Contoh kebijakan berbasis identitas".

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ScopeToPrivateApis",
      "Effect": "Allow",
      "Action": [
        "apigateway:PATCH",
        "apigateway:POST",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/restapis",
        "arn:aws:apigateway:us-east-1::/restapis/??????????"
      ],
      "Condition": {
        "ForAllValues:StringEqualsIfExists": {
          "apigateway:Request/EndpointType": "PRIVATE",
          "apigateway:Resource/EndpointType": "PRIVATE"
        }
      }
    },
    {
      "Sid": "AllowResourcePolicyUpdates",
      "Effect": "Allow",
      "Action": [
          "apigateway:UpdateRestApiPolicy"
      ],
      "Resource": [
          "arn:aws:apigateway:us-east-1::/restapis/*"
      ]
    }
  ]
}
```

Tindakan

Elemen `Action` dari kebijakan JSON menjelaskan tindakan-tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan.

Tindakan kebijakan di API Gateway menggunakan awalan berikut sebelum tindakan:`apigateway:`. Pernyataan kebijakan harus memuat elemen `Action` atau `NotAction`. API Gateway mendefinisikan serangkaian tindakannya sendiri yang menjelaskan tugas yang dapat Anda lakukan dengan layanan ini.

`Action`Ekspresi API-managing memiliki format`apigateway:`*`action`*, di mana *`action`* adalah salah satu tindakan API Gateway berikut: GET, POST, PUT, DELETE, PATCH (untuk memperbarui sumber daya), atau *, yang merupakan semua tindakan sebelumnya.

Beberapa contoh `Action` ekspresi meliputi:

- **`apigateway:*`**untuk semua tindakan API Gateway.
- **`apigateway:GET`**hanya untuk tindakan GET di API Gateway.

Untuk menetapkan beberapa tindakan dalam satu pernyataan, pisahkan tindakan-tindakan tersebut menggunakan koma seperti berikut:

```
"Action": [
      "apigateway:action1",
      "apigateway:action2"
```

Untuk informasi tentang kata kerja HTTP yang akan digunakan untuk operasi API Gateway tertentu, lihat Referensi API Amazon API Gateway Versi 1 (REST API) dan Referensi API Amazon API Gateway Versi 2 (WebSocket dan API HTTP).

Untuk informasi selengkapnya, lihat the section called “Contoh kebijakan berbasis identitas”.

Sumber daya

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan siapa yang memiliki akses ke hal apa. Yaitu, principal mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam syarat apa.

Elemen kebijakan JSON `Resource` menentukan objek atau objek-objek yang menjadi target penerapan tindakan. Pernyataan harus mencakup elemen `Resource` atau `NotResource`. Sebagai

praktik terbaik, tentukan sumber daya menggunakan Amazon Resource Name (ARN). Anda dapat melakukan ini untuk tindakan yang mendukung tipe sumber daya tertentu, yang dikenal sebagai izin tingkat sumber daya.

Untuk tindakan yang tidak mendukung izin tingkat sumber daya, misalnya operasi pencantuman, gunakan karakter wildcart (*) untuk menunjukkan bahwa pernyataan tersebut berlaku bagi semua sumber daya.

```
"Resource": "*"
```

Sumber daya API Gateway memiliki format ARN berikut:

```
arn:aws:apigateway:region::resource-path-specifier
```

Misalnya, untuk menentukan REST API dengan id *api-id*dan sub-sumber dayanya, seperti otorisasi dalam pernyataan Anda, gunakan ARN berikut:

```
"Resource": "arn:aws:apigateway:us-east-2::/restapis/api-id/*"
```

Untuk menentukan semua API REST dan sub-sumber daya milik akun tertentu, gunakan wildcard (*):

```
"Resource": "arn:aws:apigateway:us-east-2::/restapis/*"
```

Untuk daftar jenis sumber daya API Gateway dan ARNnya, lihatReferensi API Gateway Amazon Resource Name (ARN).

Kunci syarat

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan siapa yang memiliki akses ke hal apa. Yaitu, prinsipal mana yang dapat melakukan tindakan pada sumber daya apa, dan menurut persyaratan apa.

Elemen `Condition` (atau `Condition` blok) memungkinkan Anda menentukan syarat di mana suatu pernyataan berlaku. Elemen `Condition` bersifat opsional. Anda dapat membuat ekspresi bersyarat yang menggunakan operator syarat, seperti sama dengan atau kurang dari, untuk mencocokkan syarat dalam kebijakan dengan nilai dalam permintaan.

Jika Anda menentukan beberapa elemen `Condition` dalam pernyataan, atau beberapa kunci dalam satu elemen `Condition`, AWS akan mengevaluasinya denga menggunakan operasi logika `AND`.

Jika Anda menetapkan beberapa nilai untuk kunci syarat tunggal, AWS akan mengevaluasi syarat tersebut dengan menggunakan operasi logika OR. Semua persyaratan harus dipenuhi sebelum izin pernyataan diberikan.

Anda juga dapat menggunakan variabel placeholder saat menentukan syarat. Sebagai contoh, Anda dapat memberikan izin pengguna IAM untuk mengakses sumber daya hanya jika ditandai dengan nama pengguna IAM mereka. Untuk informasi lebih lanjut, lihat Elemen kebijakan IAM: variabel dan tag dalam Panduan Pengguna IAM.

AWS mendukung kunci syarat global dan kunci syarat khusus layanan. Untuk melihat semua kunci syarat global AWS, lihat Kunci konteks syarat global AWS dalam Panduan Pengguna IAM.

API Gateway mendefinisikan kumpulan kunci kondisinya sendiri dan juga mendukung penggunaan beberapa kunci kondisi global. Untuk daftar kunci kondisi API Gateway, lihat Kunci Kondisi untuk Amazon API Gateway di Panduan Pengguna IAM. Untuk informasi tentang tindakan dan sumber daya yang dapat Anda gunakan dengan kunci kondisi, lihat Tindakan yang Ditentukan oleh Amazon API Gateway.

Untuk informasi tentang penandaan, termasuk kontrol akses berbasis atribut, lihat. Penandaan

Contoh

Untuk contoh kebijakan berbasis identitas API Gateway, lihat. Contoh kebijakan berbasis identitas Amazon API Gateway

Kebijakan berbasis sumber daya API Gateway

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya API Gateway dan dalam kondisi apa. API Gateway mendukung kebijakan izin berbasis sumber daya untuk REST API. Anda menggunakan kebijakan sumber daya untuk mengontrol siapa yang dapat menjalankan REST API. Untuk informasi selengkapnya, lihat the section called “Menggunakan kebijakan sumber daya API Gateway”.

Contoh

Untuk contoh kebijakan berbasis sumber daya API Gateway, lihat. Contoh kebijakan sumber daya API Gateway

Otorisasi berdasarkan tag API Gateway

Anda dapat melampirkan tag ke resource API Gateway atau meneruskan tag dalam permintaan ke API Gateway. Untuk mengendalikan akses berdasarkan tanda, Anda dapat memberikan informasi tentang tanda di kebijakan elemen syarat menggunakan kunci syarat `apigateway:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`. Untuk informasi selengkapnya tentang menandai resource API Gateway, lihatthe section called "Kontrol akses berbasis atribut".

Untuk contoh kebijakan berbasis identitas untuk membatasi akses ke sumber daya berdasarkan tag pada sumber daya tersebut, lihat. Menggunakan tanda untuk mengontrol API Gateway sumber daya

Peran IAM API Gateway

IAM role adalah entitas di dalam akun AWS Anda yang memiliki izin tertentu.

Menggunakan kredensi sementara dengan API Gateway

Anda dapat menggunakan kredensial sementara untuk masuk dengan gabungan, menjalankan IAM role, atau menjalankan peran lintas akun. Anda memperoleh kredensil keamanan sementara dengan memanggil operasi AWS STS API seperti AssumeRoleatau. GetFederationToken

API Gateway mendukung penggunaan kredensi sementara.

Peran terkait layanan

Peran terkait layanan mengizinkan layanan AWS untuk mengakses sumber daya di layanan lain untuk menyelesaikan tindakan atas nama Anda. Peran terkait layanan muncul di akun IAM Anda dan dimiliki oleh layanan tersebut. Administrator IAM dapat melihat tetapi tidak dapat mengedit izin untuk peran terkait layanan.

API Gateway mendukung peran terkait layanan. Untuk informasi tentang membuat atau mengelola peran terkait layanan API Gateway, lihat. Menggunakan peran terkait layanan untuk API Gateway

Peran layanan

Layanan dapat mengambil peran layanan atas nama Anda. Peran layanan memungkinkan layanan mengakses sumber daya di layanan lain untuk menyelesaikan tindakan atas nama Anda. Peran layanan muncul di akun IAM Anda dan dimiliki oleh akun, sehingga administrator dapat mengubah izin untuk peran ini. Namun, melakukan hal itu dapat merusak fungsionalitas layanan.

API Gateway mendukung peran layanan.

Contoh kebijakan berbasis identitas Amazon API Gateway

Secara default, pengguna dan peran IAM tidak memiliki izin untuk membuat atau memodifikasi sumber daya API Gateway. Mereka juga tidak dapat melakukan tugas menggunakanAWS Management Console,AWS CLI, atau AWS SDK. Administrator IAM harus membuat kebijakan IAM yang memberikan izin kepada pengguna dan peran untuk melakukan operasi API tertentu pada sumber daya yang diperlukan. Administrator kemudian harus melampirkan kebijakan tersebut ke pengguna IAM atau grup yang memerlukan izin tersebut.

Untuk informasi tentang cara membuat kebijakan IAM, lihat Membuat Kebijakan pada Tab JSON di Panduan Pengguna IAM. Untuk informasi tentang tindakan, sumber daya, dan kondisi khusus untuk API Gateway, lihat Kunci tindakan, sumber daya, dan kondisi untuk Manajemen dan Tindakan Amazon API Gateway, sumber daya, dan kunci kondisi untuk Amazon API Gateway Management V2.

Topik

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya API Gateway di akun Anda. Tindakan ini membuat Akun AWS Anda terkena biaya. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulailah dengan kebijakan AWS terkelola dan beralih ke izin hak istimewa paling sedikit — Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan AWS terkelola yang memberikan izin untuk banyak kasus penggunaan umum. Mereka tersedia di AndaAkun AWS. Kami menyarankan Anda mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola AWS pelanggan yang khusus untuk kasus penggunaan Anda. Untuk informasi

selengkapnya, lihat kebijakan AWSAWS terkelola atau kebijakan terkelola untuk fungsi pekerjaan di Panduan Pengguna IAM.

- Menerapkan izin hak istimewa paling sedikit — Saat Anda menetapkan izin dengan kebijakan IAM, berikan hanya izin yang diperlukan untuk melakukan tugas. Anda melakukan ini dengan mendefinisikan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, juga dikenal sebagai izin hak istimewa paling sedikit. Untuk informasi selengkapnya tentang penggunaan IAM untuk menerapkan izin, lihat Kebijakan dan izin di IAM di Panduan Pengguna IAM.
- Gunakan ketentuan dalam kebijakan IAM untuk membatasi akses lebih lanjut — Anda dapat menambahkan kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Misalnya, Anda dapat menulis kondisi kebijakan untuk menentukan bahwa semua permintaan harus dikirim menggunakan SSL. Anda juga dapat menggunakan ketentuan untuk memberikan akses ke tindakan layanan jika digunakan melalui yang spesifikLayanan AWS, sepertiAWS CloudFormation. Untuk informasi selengkapnya, lihat elemen kebijakan IAM JSON: Kondisi dalam Panduan Pengguna IAM.
- Gunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda guna memastikan izin yang aman dan fungsional — IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan tersebut mematuhi bahasa kebijakan IAM (JSON) dan praktik terbaik IAM. IAM Access Analyzer menyediakan lebih dari 100 pemeriksaan kebijakan dan rekomendasi yang dapat ditindaklanjuti untuk membantu Anda membuat kebijakan yang aman dan fungsional. Untuk informasi selengkapnya, lihat validasi kebijakan IAM Access Analyzer di Panduan Pengguna IAM.
- Memerlukan otentikasi multi-faktor (MFA) - Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di Anda, Akun AWS aktifkan MFA untuk keamanan tambahan. Untuk mewajibkan MFA saat operasi API dipanggil, tambahkan kondisi MFA ke kebijakan Anda. Untuk informasi selengkapnya, lihat Mengonfigurasi akses API yang dilindungi MFA di Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik di IAM, lihat Praktik terbaik keamanan di IAM di Panduan Pengguna IAM.

Izinkan para pengguna untuk melihat izin mereka sendiri

Contoh ini menunjukkan cara Anda dapat membuat kebijakan yang mengizinkan para pengguna IAM untuk melihat kebijakan inline dan terkelola yang dilampirkan ke identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan pada konsol atau secara terprogram menggunakan API AWS CLI atau AWS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Izin baca sederhana

Kebijakan contoh ini memberikan izin kepada pengguna untuk mendapatkan informasi tentang semua sumber daya HTTP atau WebSocket API dengan pengenal `a123456789` di AWS Wilayah us-east-1. Sumber daya `arn:aws:apigateway:us-east-1::/apis/a123456789/*` mencakup semua sub-sumber daya API seperti otorisasi dan penerapan.

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

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:GET"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis/a123456789/*"
      ]
    }
  ]
}
```

Buat hanya REQUEST atau otorisasi JWT

Kebijakan contoh ini memungkinkan pengguna untuk membuat API hanya dengan `REQUEST` atau `JWT` otorisasi, termasuk melalui impor. Di `Resource` bagian kebijakan, `arn:aws:apigateway:us-east-1::/apis/??????????` sumber daya harus memiliki maksimal 10 karakter, yang mengecualikan sub-sumber daya API. Contoh ini digunakan `ForAllValues` di `Condition` bagian karena pengguna dapat membuat beberapa otorisasi sekaligus dengan mengimpor API.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "OnlyAllowSomeAuthorizerTypes",
      "Effect": "Allow",
      "Action": [
        "apigateway:PUT",
        "apigateway:POST",
        "apigateway:PATCH"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis",
        "arn:aws:apigateway:us-east-1::/apis/??????????",
        "arn:aws:apigateway:us-east-1::/apis/*/authorizers",
        "arn:aws:apigateway:us-east-1::/apis/*/authorizers/*"
      ],
      "Condition": {
        "ForAllValues:StringEqualsIfExists": {
          "apigateway:Request/AuthorizerType": [
            "REQUEST",
            "JWT"
```

```
          ]
        }
      }
    }
  ]
}
```

Mengharuskan **`execute-api`** titik akhir default dinonaktifkan

Kebijakan contoh ini memungkinkan pengguna untuk membuat, memperbarui, atau mengimpor API, dengan persyaratan yang `DisableExecuteApiEndpoint` ada`true`. `DisableExecuteApiEndpoint`Kapan`true`, klien tidak dapat menggunakan `execute-api` titik akhir default untuk menjalankan API.

Kami menggunakan `BoolIfExists` kondisi untuk menangani panggilan untuk memperbarui API yang tidak memiliki kunci `DisableExecuteApiEndpoint` kondisi terisi. Saat pengguna mencoba membuat atau mengimpor API, kunci `DisableExecuteApiEndpoint` kondisi selalu diisi.

Karena `apis/*` sumber daya juga menangkap sub sumber daya seperti otorisasi atau metode, kami secara eksplisit mencakupnya hanya ke API dengan pernyataan. `Deny`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DisableExecuteApiEndpoint",
      "Effect": "Allow",
      "Action": [
        "apigateway:PATCH",
        "apigateway:POST",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis",
        "arn:aws:apigateway:us-east-1::/apis/*"
      ],
      "Condition": {
        "BoolIfExists": {
          "apigateway:Request/DisableExecuteApiEndpoint": true
        }
      }
    },
```

```
    {
      "Sid": "ScopeDownToJustApis",
      "Effect": "Deny",
      "Action": [
        "apigateway:PATCH",
        "apigateway:POST",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis/*/*"
      ]
    }
  ]
}
```

Izinkan pengguna untuk membuat atau memperbarui hanya API REST pribadi

Kebijakan contoh ini menggunakan kunci kondisi untuk mengharuskan pengguna hanya membuat `PRIVATE` API, dan untuk mencegah pembaruan yang mungkin mengubah API dari `PRIVATE` tipe lain, sepertiREGIONAL.

Kami menggunakan `ForAllValues` untuk mewajibkan bahwa setiap `EndpointType` ditambahkan ke API adalahPRIVATE. Kami menggunakan kunci kondisi sumber daya untuk mengizinkan pembaruan apa pun ke API selama ituPRIVATE. `ForAllValues`hanya berlaku jika ada kunci kondisi.

Kami menggunakan non-greedy matcher (?) untuk secara eksplisit mencocokkan dengan ID API untuk mencegah mengizinkan sumber daya non-API seperti otorisasi.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ScopePutToPrivateApis",
            "Effect": "Allow",
            "Action": [
                "apigateway:PUT"
            ],
            "Resource": [
                "arn:aws:apigateway:us-east-1::/restapis",
                "arn:aws:apigateway:us-east-1::/restapis/??????????"
            ],
```

```
            "Condition": {
                "ForAllValues:StringEquals": {
                    "apigateway:Resource/EndpointType": "PRIVATE"
                }
            }
        },
        {
            "Sid": "ScopeToPrivateApis",
            "Effect": "Allow",
            "Action": [
                "apigateway:DELETE",
                "apigateway:PATCH",
                "apigateway:POST"
            ],
            "Resource": [
                "arn:aws:apigateway:us-east-1::/restapis",
                "arn:aws:apigateway:us-east-1::/restapis/??????????"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "apigateway:Request/EndpointType": "PRIVATE",
                    "apigateway:Resource/EndpointType": "PRIVATE"
                }
            }
        },
        {
            "Sid": "AllowResourcePolicyUpdates",
            "Effect": "Allow",
            "Action": [
                "apigateway:UpdateRestApiPolicy"
            ],
            "Resource": [
                "arn:aws:apigateway:us-east-1::/restapis/*"
            ]
        }
    ]
}
```

Mengharuskan rute API memiliki otorisasi

Kebijakan ini menyebabkan upaya untuk membuat atau memperbarui rute (termasuk melalui impor) gagal jika rute tidak memiliki otorisasi. `ForAnyValue`mengevaluasi ke false jika kunci tidak ada,

seperti ketika rute tidak sedang dibuat atau diperbarui. Kami menggunakan `ForAnyValue` karena beberapa rute dapat dibuat melalui impor.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowUpdatesOnApisAndRoutes",
      "Effect": "Allow",
      "Action": [
        "apigateway:POST",
        "apigateway:PATCH",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis",
        "arn:aws:apigateway:us-east-1::/apis/??????????",
        "arn:aws:apigateway:us-east-1::/apis/*/routes",
        "arn:aws:apigateway:us-east-1::/apis/*/routes/*"
      ]
    },
    {
      "Sid": "DenyUnauthorizedRoutes",
      "Effect": "Deny",
      "Action": [
        "apigateway:POST",
        "apigateway:PATCH",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis",
        "arn:aws:apigateway:us-east-1::/apis/*"
      ],
      "Condition": {
        "ForAnyValue:StringEqualsIgnoreCase": {
          "apigateway:Request/RouteAuthorizationType": "NONE"
        }
      }
    }
  ]
}
```

Mencegah pengguna membuat atau memperbarui tautan VPC

Kebijakan ini mencegah pengguna membuat atau memperbarui tautan VPC. Tautan VPC memungkinkan Anda mengekspos sumber daya dalam VPC Amazon ke klien di luar VPC.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyVPCLink",
      "Effect": "Deny",
      "Action": [
        "apigateway:POST",
        "apigateway:PUT",
        "apigateway:PATCH"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/vpclinks",
        "arn:aws:apigateway:us-east-1::/vpclinks/*"
      ]
    }
  ]
}
```

Contoh kebijakan berbasis sumber daya Amazon API Gateway

Untuk contoh kebijakan berbasis sumber daya, lihat. the section called “Contoh kebijakan sumber daya API Gateway”

Memecahkan masalah identitas dan akses Amazon API Gateway

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan API Gateway dan IAM.

Topik

- Saya tidak berwenang untuk melakukan tindakan di API Gateway
- Saya tidak berwenang untuk melakukan iam: PassRole
- Saya ingin mengizinkan orang di luar AWS akun saya untuk mengakses sumber daya API Gateway saya

Saya tidak berwenang untuk melakukan tindakan di API Gateway

Jika Anda menerima kesalahan bahwa Anda tidak berwenang untuk melakukan tindakan, kebijakan Anda harus diperbarui untuk memungkinkan Anda melakukan tindakan.

Contoh kesalahan berikut terjadi ketika pengguna `mateojackson` IAM mencoba menggunakan konsol untuk melihat detail tentang *`my-example-widget`* sumber daya fiksi tetapi tidak memiliki izin `apigateway:`*`GetWidget`* fiksi.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 apigateway:GetWidget on resource: my-example-widget
```

Dalam hal ini, kebijakan untuk `mateojackson` pengguna harus diperbarui untuk mengizinkan akses ke *`my-example-widget`* sumber daya dengan menggunakan `apigateway:`*`GetWidget`* tindakan.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensi masuk Anda.

Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan yang tidak diizinkan untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran ke API Gateway.

Beberapa Layanan AWS memungkinkan Anda untuk meneruskan peran yang ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait layanan. Untuk melakukannya, Anda harus memiliki izin untuk meneruskan peran ke layanan.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama `marymajor` mencoba menggunakan konsol untuk melakukan tindakan di API Gateway. Namun, tindakan tersebut mengharuskan layanan memiliki izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut ke layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dalam hal ini, kebijakan Mary harus diperbarui untuk memungkinkannya melakukan `iam:PassRole` tindakan.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensi masuk Anda.

Saya ingin mengizinkan orang di luar AWS akun saya untuk mengakses sumber daya API Gateway saya

Anda dapat membuat peran yang dapat digunakan para pengguna di akun lain atau orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACL), Anda dapat menggunakan kebijakan tersebut untuk memberi akses pada orang ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa hal berikut:

- Untuk mempelajari apakah API Gateway mendukung fitur-fitur ini, lihatCara kerja Amazon API Gateway dengan IAM.
- Untuk mempelajari cara memberikan akses ke sumber daya di seluruh Akun AWS yang Anda miliki, lihat Menyediakan akses ke pengguna IAM di akun Akun AWS lain yang Anda miliki dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses ke sumber daya Anda ke Akun AWS pihak ketiga, lihat Menyediakan akses ke akun Akun AWS yang dimiliki oleh pihak ketiga dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat Menyediakan akses ke pengguna terautentikasi eksternal (gabungan identitas) dalam Panduan Pengguna IAM .
- Untuk mempelajari perbedaan antara penggunaan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Perbedaan IAM role dan kebijakan berbasis sumber daya di Panduan Pengguna IAM.

Menggunakan peran terkait layanan untuk API Gateway

Amazon API Gateway menggunakan AWS Identity and Access Management peran terkait layanan (IAM). Peran terkait layanan adalah jenis peran IAM unik yang ditautkan langsung ke API Gateway. Peran terkait layanan telah ditentukan sebelumnya oleh API Gateway dan menyertakan semua izin yang diperlukan layanan untuk memanggil AWS layanan lain atas nama Anda.

Peran terkait layanan membuat pengaturan API Gateway lebih mudah karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. API Gateway mendefinisikan izin dari peran terkait layanan, dan kecuali ditentukan lain, hanya API Gateway yang dapat mengambil perannya. Izin yang ditentukan mencakup kebijakan kepercayaan dan kebijakan izin, serta bahwa kebijakan izin tidak dapat dilampirkan ke entitas IAM lainnya.

Anda dapat menghapus peran tertaut layanan hanya setelah menghapus sumber daya terkait terlebih dahulu. Ini melindungi sumber daya API Gateway Anda karena Anda tidak dapat secara tidak sengaja menghapus izin untuk mengakses sumber daya.

Untuk informasi tentang layanan lain yang mendukung peran terkait layanan, silakan lihat Layanan AWS yang bisa digunakan dengan IAM dan carilah layanan yang memiliki opsi Ya di kolom Peran Terkait Layanan. Pilih Ya dengan sebuah tautan untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Izin peran terkait layanan untuk API Gateway

API Gateway menggunakan peran terkait layanan bernama AWSServiceRoleForAPIGateway—Memungkinkan API Gateway mengakses Elastic Load Balancing, Amazon Data Firehose, dan sumber daya layanan lainnya atas nama Anda.

Peran AWSServiceRoleForAPIGateway terkait layanan mempercayai layanan berikut untuk mengambil peran:

- `ops.apigateway.amazonaws.com`

Kebijakan izin peran memungkinkan API Gateway menyelesaikan tindakan berikut pada sumber daya yang ditentukan:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:AddListenerCertificates",
                "elasticloadbalancing:RemoveListenerCertificates",
                "elasticloadbalancing:ModifyListener",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:DescribeLoadBalancers",
                "xray:PutTraceSegments",
                "xray:PutTelemetryRecords",
                "xray:GetSamplingTargets",
                "xray:GetSamplingRules",
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
```

```
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries",
                "servicediscovery:DiscoverInstances"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "firehose:DescribeDeliveryStream",
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Resource": "arn:aws:firehose:*:*:deliverystream/amazon-apigateway-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "acm:DescribeCertificate",
                "acm:GetCertificate"
            ],
            "Resource": "arn:aws:acm:*:*:certificate/*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateNetworkInterfacePermission",
            "Resource": "arn:aws:ec2:*:*:network-interface/*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:*:*:network-interface/*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": [
                        "Owner",
                        "VpcLinkId"
                    ]
                }
            }
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DeleteNetworkInterface",
                "ec2:AssignPrivateIpAddresses",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeNetworkInterfaceAttribute",
                "ec2:DescribeVpcs",
                "ec2:DescribeNetworkInterfacePermissions",
                "ec2:UnassignPrivateIpAddresses",
                "ec2:DescribeSubnets",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "servicediscovery:GetNamespace",
            "Resource": "arn:aws:servicediscovery:*:*:namespace/*"
        },
        {
            "Effect": "Allow",
            "Action": "servicediscovery:GetService",
            "Resource": "arn:aws:servicediscovery:*:*:service/*"
        }
    ]
}
```

Anda harus mengonfigurasi izin untuk mengizinkan entitas IAM (seperti pengguna, grup, atau peran) untuk membuat, mengedit, atau menghapus peran terkait layanan. Untuk informasi lebih lanjut, lihat Izin Peran Tertaut Layanan di Panduan Pengguna IAM.

Membuat peran terkait layanan untuk API Gateway

Anda tidak perlu membuat peran tertaut layanan secara manual. Saat Anda membuat API, nama domain kustom, atau tautan VPC di, API Gateway AWS Management Console AWS CLI, atau AWS API, API Gateway membuat peran terkait layanan untuk Anda.

Jika Anda menghapus peran terkait layanan ini, dan ingin membuatnya lagi, Anda dapat mengulangi proses yang sama untuk membuat kembali peran tersebut di akun Anda. Saat Anda membuat API, nama domain kustom, atau tautan VPC, API Gateway membuat peran terkait layanan untuk Anda lagi.

Mengedit peran terkait layanan untuk API Gateway

API Gateway tidak memungkinkan Anda mengedit peran AWSServiceRoleForAPIGateway terkait layanan. Setelah membuat peran terkait layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin mereferensikan peran tersebut. Namun, Anda dapat menyunting penjelasan peran menggunakan IAM. Untuk informasi selengkapnya, silakan lihat Mengedit peran tertaut layanan dalam Panduan Pengguna IAM.

Menghapus peran terkait layanan untuk API Gateway

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran tertaut layanan, sebaiknya Anda menghapus peran tersebut. Dengan demikian, Anda tidak memiliki entitas tidak terpakai yang tidak dipantau atau dipelihara secara aktif. Tetapi, Anda harus membersihkan sumber daya peran terkait layanan sebelum menghapusnya secara manual.

> **Note**
>
> Jika layanan API Gateway menggunakan peran saat Anda mencoba menghapus sumber daya, penghapusan mungkin gagal. Jika hal itu terjadi, tunggu beberapa menit dan coba mengoperasikannya lagi.

Untuk menghapus sumber daya API Gateway yang digunakan oleh AWSServiceRoleForAPIGateway

1. Buka konsol API Gateway di https://console.aws.amazon.com/apigateway/.
2. Arahkan ke API, nama domain kustom, atau tautan VPC yang menggunakan peran terkait layanan.
3. Gunakan konsol untuk menghapus sumber daya.
4. Ulangi prosedur untuk menghapus semua API, nama domain khusus, atau tautan VPC yang menggunakan peran terkait layanan.

Untuk menghapus peran terkait layanan secara manual menggunakan IAM

Gunakan konsol IAM, the AWS CLI, atau AWS API untuk menghapus peran AWSServiceRoleForAPIGateway terkait layanan. Untuk informasi selengkapnya, silakan lihat Menghapus Peran Terkait Layanan di Panduan Pengguna IAM.

Wilayah yang Didukung untuk peran terkait layanan API Gateway

API Gateway mendukung penggunaan peran terkait layanan di semua Wilayah tempat layanan tersedia. Untuk informasi selengkapnya, lihat Titik Akhir AWS Layanan.

Pembaruan API Gateway ke kebijakan AWS terkelola

Lihat detail tentang pembaruan kebijakan AWS terkelola untuk API Gateway sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan umpan RSS di halaman riwayat Dokumen API Gateway.

Perubahan	Deskripsi	Tanggal
Menambahkan `acm:GetCertificate` dukungan pada `AWSServiceRoleForAPIGateway` kebijakan.	`AWSServiceRoleForAPIGateway` Kebijakan ini sekarang menyertakan izin untuk memanggil tindakan ACM `GetCertificate` API.	12 Juli 2021
API Gateway mulai melacak perubahan	API Gateway mulai melacak perubahan untuk kebijakan AWS terkelolanya.	12 Juli 2021

Pencatatan dan pemantauan di Amazon API Gateway

Pemantauan adalah bagian penting dalam menjaga keandalan, ketersediaan, dan kinerja API Gateway dan AWS solusi Anda. Anda harus mengumpulkan data pemantauan dari semua bagian AWS solusi Anda sehingga Anda dapat lebih mudah men-debug kegagalan multi-titik jika terjadi. AWS menyediakan beberapa alat untuk memantau sumber daya API Gateway Anda dan menanggapi potensi insiden:

CloudWatch Log Amazon

Untuk membantu men-debug masalah yang terkait dengan eksekusi permintaan atau akses klien ke API Anda, Anda dapat mengaktifkan CloudWatch Log untuk mencatat panggilan API. Untuk informasi selengkapnya, lihat the section called “CloudWatch log”.

CloudWatch Alarm Amazon

Menggunakan CloudWatch alarm, Anda menonton satu metrik selama periode waktu yang Anda tentukan. Jika metrik melebihi ambang batas tertentu, pemberitahuan akan dikirim ke topik atau AWS Auto Scaling kebijakan Amazon Simple Notification Service. CloudWatch alarm tidak memanggil tindakan ketika metrik berada dalam keadaan tertentu. Sebaliknya, kondisi tersebut harus diubah dan dipertahankan selama periode tertentu. Untuk informasi selengkapnya, lihat the section called “Metrik CloudWatch”.

Akses Logging ke Firehose

Untuk membantu men-debug masalah yang terkait dengan akses klien ke API Anda, Anda dapat mengaktifkan Firehose untuk mencatat panggilan API. Untuk informasi selengkapnya, lihat the section called “Firehose”.

AWS CloudTrail

CloudTrail menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di API Gateway. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat ke API Gateway, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan. Untuk informasi selengkapnya, lihat the section called “Bekerja dengan AWS CloudTrail”.

AWS X-Ray

X-Ray adalah AWS layanan yang mengumpulkan data tentang permintaan yang dilayani aplikasi Anda, dan menggunakannya untuk membuat peta layanan yang dapat Anda gunakan untuk mengidentifikasi masalah dengan aplikasi Anda dan peluang untuk pengoptimalan. Untuk informasi selengkapnya, lihat the section called “Menyiapkan AWS X-Ray”.

AWS Config

AWS Config memberikan tampilan terperinci tentang konfigurasi AWS sumber daya di akun Anda. Anda dapat melihat bagaimana sumber daya terkait, mendapatkan riwayat perubahan konfigurasi, dan melihat bagaimana hubungan dan konfigurasi berubah seiring waktu. Anda dapat menggunakan AWS Config untuk menentukan aturan yang mengevaluasi konfigurasi sumber daya untuk kepatuhan data. AWS Config aturan mewakili setelan konfigurasi ideal untuk sumber

daya API Gateway Anda. Jika sumber daya melanggar aturan dan ditandai sebagai tidak sesuai, AWS Config Anda dapat memperingatkan Anda menggunakan topik Amazon Simple Notification Service (Amazon SNS). Lihat perinciannya di the section called “Bekerja dengan AWS Config”.

Menebus panggilan ke API Amazon API Gateway denganAWS CloudTrail

Amazon API Gateway terintegrasi denganAWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atauAWSlayanan di API Gateway. CloudTrail menangkap semua panggilan REST API untuk API Gateway sebagai peristiwa, termasuk panggilan dari konsol API Gateway dan panggilan kode ke API Gateway.

> **Note**
>
> TestInvokeAuthorizerdanTestInvokeMethodtidak login CloudTrail.

Jika membuat jejak, Anda dapat mengaktifkan pengiriman kejadian CloudTrail berkelanjutan ke bucket Amazon S3, termasuk peristiwa untuk API Gateway. Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru dalam konsol CloudTrail di Riwayat peristiwa.

Menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat ke API Gateway, alamat IP asal permintaan dibuat, siapa yang membuat permintaan, kapan permintaan dibuat, dan detail lainnya.

Untuk mempelajari selengkapnya tentang CloudTrail, lihat Panduan Pengguna AWS CloudTrail.

Informasi API Gateway di CloudTrail

CloudTrail diaktifkan pada akun AWS Anda saat Anda membuat akun tersebut. Saat aktivitas terjadi di Amazon API Gateway, aktivitas tersebut dicatat di kejadian CloudTrail bersama lainnyaAWSperistiwa layanan diRiwayat peristiwa. Anda dapat melihat, mencari, dan mengunduh peristiwa terbaru di akun AWS Anda. Untuk informasi selengkapnya, lihat Melihat Peristiwa dengan Riwayat Peristiwa CloudTrail.

Untuk catatan berkelanjutan tentang peristiwa diAWSakun, termasuk peristiwa untuk API Gateway, buat jejak. Jejak memungkinkan CloudTrail mengirim file log ke bucket Amazon S3. Secara default, saat Anda membuat jejak di dalam konsol tersebut, jejak diterapkan ke semua Wilayah AWS. Jejak mencatat peristiwa dari semua Wilayah di partisi AWS dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi layanan AWS lainnya

untuk menganalisis lebih lanjut dan bertindak berdasarkan data peristiwa yang dikumpulkan di log CloudTrail. Untuk informasi selengkapnya, lihat:

- Ikhtisar untuk Membuat Jejak
- Layanan yang Didukung dan Integrasi CloudTrail
- Mengonfigurasi Notifikasi Amazon SNS untuk CloudTrail
- Menerima Berkas Log CloudTrail dari Berbagai Wilayah dan Menerima Berkas Log CloudTrail dari Berbagai Akun

Semua tindakan Amazon API Gateway dicatat oleh CloudTrail dan didokumentasikan diReferensi API. Misalnya, panggilan untuk membuat API, resource, atau metode baru di API Gateway menghasilkan entri di berkas log CloudTrail.

Setiap entri peristiwa atau log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan hal berikut:

- Jika permintaan tersebut dibuat dengan kredensial pengguna root atau IAM.
- Jika permintaan tersebut dibuat dengan kredensial keamanan sementara untuk peran atau pengguna federasi.
- Apakah permintaan dibuat oleh layanan AWS lain.

Untuk informasi lebih lanjut, lihat Elemen userIdentity CloudTrail.

Memahami entri berkas log API Gateway

Jejak adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai berkas log ke bucket Amazon S3 yang telah Anda tentukan. File log CloudTrail berisi satu atau beberapa entri log. Sebuah peristiwa mewakili permintaan tunggal dari sumber apa pun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. Berkas log CloudTrail bukan merupakan jejak tumpukan yang diurutkan dari panggilan API publik, sehingga tidak muncul dalam urutan tertentu.

Contoh berikut menampilkan entri log CloudTrail yang menunjukkan API Gateway`GetResource`tindakan:

```
{
    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 ...
    ]
}
```

Memantau konfigurasi API Gateway API denganAWS Config

Anda dapat menggunakan AWS Configuntuk mencatat perubahan konfigurasi yang dibuat ke sumber daya API Gateway API Anda dan mengirim notifikasi berdasarkan perubahan sumber daya. Mempertahankan riwayat perubahan konfigurasi untuk sumber daya API Gateway berguna untuk pemecahan masalah operasional, audit, dan kasus penggunaan kepatuhan.

AWS Configdapat melacak perubahan ke:

- Konfigurasi tahap API, seperti:
  - pengaturan klaster cache
  - pengaturan throttle
  - pengaturan log akses
  - penyebaran aktif yang ditetapkan di atas panggung
- Konfigurasi API, seperti:
  - konfigurasi endpoint
  - versi
  - protokol
  - tag

Selain itu,Aturan AWS Config fitur ini memungkinkan Anda untuk menentukan aturan konfigurasi dan secara otomatis mendeteksi, melacak, dan memperingatkan pelanggaran terhadap aturan ini. Dengan melacak perubahan pada properti konfigurasi sumber daya ini, Anda juga dapat membuatAWS Config aturan yang dipicu perubahan untuk sumber daya API Gateway, dan menguji konfigurasi sumber daya Anda terhadap praktik terbaik.

Anda dapat mengaktifkanAWS Config di akun Anda dengan menggunakanAWS Config konsol atauAWS CLI. Pilih jenis sumber daya yang ingin Anda lacak perubahannya. Jika sebelumnya Anda mengonfigurasiAWS Config untuk merekam semua jenis sumber daya, maka sumber daya API Gateway ini akan direkam secara otomatis di akun Anda. Support untuk Amazon API GatewayAWS Config tersedia di semua wilayahAWS publik danAWS GovCloud (US). Untuk daftar lengkap Wilayah yang didukung, lihat Endpoint dan Kuota Amazon API Gateway diReferensi Umum AWS.

Topik

Jenis sumber daya yang mendukung

Jenis sumber daya API Gateway berikut terintegrasi denganAWS Config dan didokumentasikan dalam JenisAWS Sumber DayaAWS Config yang Didukung dan Hubungan Sumber Daya:

- `AWS::ApiGatewayV2::Api`(WebSocketdan HTTP API)
- `AWS::ApiGateway::RestApi`(API ISTIRAHAT)
- `AWS::ApiGatewayV2::Stage`(WebSocketdan tahap API HTTP)
- `AWS::ApiGateway::Stage`(Tahap API REST)

Untuk informasi tentang AWS Config, lihat Panduan Developer AWS Config. Untuk informasi harga, lihat halaman informasi harga AWS Config.

> **Important**
>
> Jika Anda mengubah salah satu properti API berikut setelah API diterapkan, Anda harus men-deploy ulang API untuk menyebarkan perubahan. Jika tidak, Anda akan melihat perubahan atribut diAWS Config konsol, tetapi pengaturan properti sebelumnya akan tetap berlaku; perilaku runtime API tidak akan berubah.
>
> - **`AWS::ApiGateway::RestApi`** – `binaryMediaTypes`, `minimumCompressionSize`, `apiKeySource`
> - **`AWS::ApiGatewayV2::Api`** – `apiKeySelectionExpression`

Menyiapkan AWS Config

Untuk menyiapkan AWS Config, lihat topik berikut dalam Panduan Developer AWS Config.

- MenyiapkanAWS Config dengan Konsol
- MenyiapkanAWS Config denganAWS CLI

MengkonfigurasiAWS Config untuk merekam sumber daya API Gateway

Secara default,AWS Config mencatat perubahan konfigurasi untuk semua jenis yang didukung sumber daya regional yang ditemukan di wilayah di mana lingkungan Anda berjalan. Anda dapat

menyesuaikanAWS Config untuk mencatat perubahan hanya untuk jenis sumber daya tertentu, atau perubahan ke sumber daya global.

Untuk mempelajari tentang sumber daya regional vs global dan mempelajari cara menyesuaikanAWS Config konfigurasi Anda, lihat MemilihAWS Config Rekaman Sumber Daya yang mana.

Melihat detail konfigurasi API Gateway diAWS Config konsol

Anda dapat menggunakanAWS Config konsol untuk mencari sumber daya API Gateway dan mendapatkan rincian saat ini dan sejarah tentang konfigurasi mereka. Prosedur berikut menunjukkan cara menemukan informasi tentang API Gateway API.

Untuk menemukan sumber daya API Gateway di konsolAWS konfigurasi

1. Buka konsol AWS Config.
2. Pilih Sumber Daya.
3. Pada halamanInventarisasi sumber daya, pilih Sumber Daya.
4. Buka menu Jenis sumber daya, gulir ke Apigateway atau Apigatewayv2, dan kemudian pilih satu atau lebih dari jenis sumber daya API Gateway.
5. Pilih Lihat.
6. Pilih ID sumber daya dalam daftar sumber daya yangAWS Config tampilkan. AWS Configmenampilkan rincian konfigurasi dan informasi lainnya tentang sumber daya yang Anda pilih.
7. Untuk melihat detail lengkap konfigurasi yang direkam, pilih Lihat Detail.

Untuk mempelajari lebih lanjut cara menemukan sumber daya dan melihat informasi di halaman ini, lihat Melihat KonfigurasiAWS Sumber Daya dan Sejarah dalam PanduanAWS Config Developer.

Mengevaluasi sumber daya API Gateway menggunakanAWS Config aturan

Anda dapat membuatAWS Config aturan, yang mewakili pengaturan konfigurasi yang ideal untuk sumber daya API Gateway Anda. Anda dapat menggunakan AturanAWS Config Terkelola yang telah ditetapkan, atau menentukan aturan khusus. AWS Configterus melacak perubahan pada konfigurasi sumber daya Anda untuk menentukan apakah perubahan tersebut melanggar salah satu syarat dalam aturan Anda. Konsol AWS Config menunjukkan status kepatuhan aturan dan sumber daya Anda.

Jika sumber daya melanggar aturan dan ditandai sebagai tidak patuh,AWS Config dapat mengingatkan Anda menggunakan topik Amazon Simple Notification Service (Amazon SNS). Untuk pemrograman mengonsumsi data dalamAWS Config peringatan, gunakan antrian Amazon Simple Queue Service (Amazon SQS) sebagai titik akhir pemberitahuan untuk topik Amazon SNS.

Untuk mempelajari lebih lanjut tentang menyiapkan dan menggunakan aturan, lihat Mengevaluasi Sumber daya dengan Aturan dalam PanduanAWS Config Developer.

Validasi kepatuhan untuk Amazon API Gateway

Untuk mempelajari apakah an Layanan AWS berada dalam lingkup program kepatuhan tertentu, lihat Layanan AWSdi Lingkup oleh Program Kepatuhan Layanan AWS dan pilih program kepatuhan yang Anda minati. Untuk informasi umum, lihat Program Kepatuhan AWS.

Anda bisa mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di AWS Artifact.

Tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, serta hukum dan peraturan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- Panduan Memulai Cepat Keamanan dan Kepatuhan — Panduan penerapan ini membahas pertimbangan arsitektur dan memberikan langkah-langkah untuk menerapkan lingkungan dasar AWS yang berfokus pada keamanan dan kepatuhan.
- Arsitektur untuk Keamanan dan Kepatuhan HIPAA di Amazon Web Services — Whitepaper ini menjelaskan bagaimana perusahaan dapat menggunakan AWS untuk membuat aplikasi yang memenuhi syarat HIPAA.

  > **Note**
  >
  > Tidak semua Layanan AWS memenuhi syarat HIPAA. Untuk informasi selengkapnya, lihat Referensi Layanan yang Memenuhi Syarat HIPAA.

- Sumber Daya Kepatuhan AWS – Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- AWSPanduan Kepatuhan Pelanggan - Memahami model tanggung jawab bersama melalui lensa kepatuhan. Panduan ini merangkum praktik terbaik untuk mengamankan Layanan AWS dan memetakan panduan untuk kontrol keamanan di berbagai kerangka kerja (termasuk Institut

Standar dan Teknologi Nasional (NIST), Dewan Standar Keamanan Industri Kartu Pembayaran (PCI), dan Organisasi Internasional untuk Standardisasi (ISO)).

- Mengevaluasi Sumber Daya dengan Aturan di Panduan Developer AWS Config – Layanan AWS Config menilai seberapa baik konfigurasi sumber daya Anda dalam mematuhi praktik-praktik internal, pedoman industri, dan regulasi internal.
- AWS Security Hub— Ini Layanan AWS memberikan pandangan komprehensif tentang keadaan keamanan Anda di dalamnyaAWS. Security Hub menggunakan kontrol keamanan untuk mengevaluasi AWS sumber daya Anda dan untuk memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik. Untuk daftar layanan dan kontrol yang didukung, lihat referensi kontrol Security Hub.
- AWS Audit Manager – Layanan AWS ini akan membantu Anda untuk terus-menerus mengaudit penggunaan AWS untuk menyederhanakan bagaimana Anda mengelola risiko dan kepatuhan terhadap regulasi dan standar industri.

ketahanan di Amazon API Gateway

Infrastruktur global AWS dibangun di sekitar Wilayah AWS dan Availability Zone. AWS Wilayah menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan jaringan berlatensi rendah, throughput yang tinggi, dan sangat redundan. Dengan Availability Zone, Anda dapat mendesain dan mengoperasikan aplikasi dan basis data yang secara otomatis mengalami kegagalan di antara zona tanpa gangguan. Availability Zone lebih tersedia, memiliki toleransi kesalahan, dan dapat diskalakan dibandingkan dengan satu atau beberapa infrastruktur pusat data tradisional.

Untuk informasi selengkapnya tentang Wilayah AWS dan Availability Zone, lihat AWS Infrastruktur Global.

Untuk mencegah API Anda kewalahan oleh terlalu banyak permintaan, API Gateway membatasi permintaan ke API Anda. Secara khusus, API Gateway menetapkan batas pada tingkat steady-state dan ledakan pengiriman permintaan terhadap semua API di akun Anda. Anda dapat mengonfigurasi pembatasan khusus untuk API Anda. Untuk mempelajari selengkapnya, lihat Permintaan Throttle API untuk throughput yang lebih baik.

Anda dapat menggunakan pemeriksaan kesehatan Route 53 untuk mengontrol failover DNS dari API Gateway API di wilayah utama ke API Gateway API di wilayah sekunder. Sebagai contoh, lihat the section called “failover DNS”.

Keamanan infrastruktur di Amazon API Gateway

Sebagai layanan terkelola, Amazon API Gateway dilindungi olehAWSkeamanan jaringan global. Untuk informasi tentangAWSlayanan keamanan dan bagaimanaAWSmelindungi infrastruktur, lihatAWSKeamanan Cloud. Untuk mendesainAWSlingkungan menggunakan praktik terbaik untuk keamanan infrastruktur, lihatPerlindungan InfrastrukturdiPilar KeamananAWSKerangka Kerja yang Diarsiteksikan dengan Baik.

Anda menggunakanAWSpanggilan API yang dipublikasikan untuk mengakses API Gateway melalui jaringan. Klien harus mendukung hal berikut:

- Transport Layer Security (TLS). Kami membutuhkan TLS 1.2 dan merekomendasikan TLS 1.3.
- Suite cipher dengan kerahasiaan maju sempurna (PFS) seperti DHE (Ephemeral Diffie-Hellman) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Sebagian besar sistem modern seperti Java 7 dan sistem yang lebih baru mendukung mode ini.

Selain itu, permintaan harus ditandatangani menggunakan access key ID dan secret access key yang terkait dengan principal IAM. Atau Anda bisa menggunakan AWS Security Token Service (AWS STS) untuk membuat kredensial keamanan sementara guna menandatangani permintaan.

Anda dapat memanggil operasi API ini dari lokasi jaringan mana pun, tetapi API Gateway mendukung kebijakan akses berbasis sumber daya, yang dapat mencakup pembatasan berdasarkan alamat IP sumber. Anda juga dapat menggunakan kebijakan berbasis sumber daya untuk mengontrol akses dari titik akhir Amazon Virtual Private Cloud (Amazon VPC) tertentu atau VPC tertentu. Secara efektif, ini mengisolasi akses jaringan ke sumber daya API Gateway yang diberikan hanya dari VPC tertentu di dalamAWSjaringan.

Analisis kerentanan di Amazon API Gateway

Konfigurasi dan kontrol IT merupakan tanggung jawab bersama antara AWS dan Anda, pelanggan kami. Untuk informasi selengkapnya, lihat model tanggung jawab bersama AWS.

Praktik terbaik keamanan di Amazon API Gateway

API Gateway menyediakan sejumlah fitur keamanan untuk dipertimbangkan saat Anda mengembangkan dan menerapkan kebijakan keamanan Anda sendiri. Praktik terbaik berikut adalah pedoman umum dan tidak mewakili solusi keamanan yang lengkap. Karena praktik terbaik

ini mungkin tidak sesuai atau tidak memadai untuk lingkungan Anda, perlakukan itu sebagai pertimbangan yang bermanfaat, bukan sebagai resep.

Terapkan akses hak akses paling rendah

Gunakan kebijakan IAM untuk menerapkan akses hak istimewa paling sedikit untuk membuat, membaca, memperbarui, atau menghapus API Gateway API. Untuk mempelajari selengkapnya, lihat Manajemen identitas dan akses untuk Amazon API Gateway. API Gateway menawarkan beberapa opsi untuk mengontrol akses ke API yang Anda buat. Untuk mempelajari lebih lanjut, lihatMengontrol dan mengelola akses ke REST API di API Gateway,Mengontrol dan mengelola akses ke WebSocket API di API Gateway, danMengontrol akses ke API HTTP dengan otorisasi JWT.

Menerapkan logging

Gunakan CloudWatch Log atau Amazon Data Firehose untuk mencatat permintaan ke API Anda. Untuk mempelajari lebih lanjut, lihatMemantau REST API,Mengonfigurasi logging untuk API WebSocket , danMengkonfigurasi logging untuk HTTP API.

Menerapkan CloudWatch alarm Amazon

Menggunakan CloudWatch alarm, Anda menonton satu metrik selama periode waktu yang Anda tentukan. Jika metrik melebihi ambang batas tertentu, pemberitahuan akan dikirim ke topik atau AWS Auto Scaling kebijakan Amazon Simple Notification Service. CloudWatch alarm tidak memanggil tindakan ketika metrik berada dalam keadaan tertentu. Sebaliknya, negara harus telah berubah dan dipertahankan untuk sejumlah periode tertentu. Untuk informasi selengkapnya, lihat the section called “Metrik CloudWatch”.

Aktifkan AWS CloudTrail

CloudTrail menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di API Gateway. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat ke API Gateway, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan. Untuk informasi selengkapnya, lihat the section called “Bekerja dengan AWS CloudTrail”.

Aktifkan AWS Config

AWS Config memberikan tampilan terperinci tentang konfigurasi AWS sumber daya di akun Anda. Anda dapat melihat bagaimana sumber daya terkait, mendapatkan riwayat perubahan konfigurasi, dan melihat bagaimana hubungan dan konfigurasi berubah seiring waktu. Anda dapat menggunakan AWS Config untuk menentukan aturan yang mengevaluasi konfigurasi sumber

daya untuk kepatuhan data. AWS Config aturan mewakili pengaturan konfigurasi ideal untuk sumber daya API Gateway Anda. Jika sumber daya melanggar aturan dan ditandai sebagai tidak sesuai, AWS Config Anda dapat memperingatkan Anda menggunakan topik Amazon Simple Notification Service (Amazon SNS). Untuk detailnya, lihat the section called “Bekerja dengan AWS Config”.

Gunakan AWS Security Hub

Pantau penggunaan API Gateway yang berkaitan dengan praktik terbaik keamanan dengan menggunakan AWS Security Hub. Hub Keamanan menggunakan kontrol keamanan untuk mengevaluasi konfigurasi sumber daya dan standar keamanan guna membantu Anda mematuhi berbagai kerangka kerja kepatuhan. Untuk informasi selengkapnya tentang menggunakan Security Hub guna mengevaluasi resource API Gateway, lihat kontrol Amazon API Gateway di Panduan AWS Security Hub Pengguna.

Memberi tag API Gateway daya

Tanda adalah label metadata yang Anda tetapkan atau AWS yang ditetapkan ke sumber daya AWS. Setiap tanda memiliki dua bagian:

- Sebuah kunci tag (misalnya, `CostCenter`, `Environment`, atau `Project`). Kunci tag peka terhadap huruf besar dan kecil.
- Bidang opsional yang dikenal sebagai nilai tanda (misalnya, `111122223333` atau `Production`). Mengabaikan nilai tag sama saja dengan menggunakan string kosong. Seperti kunci tag, nilai tag peka huruf besar.

Tanda membantu Anda melakukan hal berikut:

- Kontrol akses ke sumber daya Anda berdasarkan tanda yang ditetapkan untuk sumber daya tersebut. Anda mengontrol akses dengan menentukan kunci tanda dan nilai-nilai dalam syarat untuk (IAM) kebijakan AWS Identity and Access Management. Untuk informasi selengkapnya tentang kontrol akses berbasis tanda, lihat Mengontrol Akses Menggunakan Tanda dalam Panduan Pengguna IAM.
- Telusuri biaya AWS Anda. Anda mengaktifkan tag ini pada AWS Billing and Cost Management dasbor.AWS menggunakan tag untuk mengategorikan biaya Anda lalu mengirimkan laporan alokasi biaya bulanan kepada Anda. Untuk informasi selengkapnya, lihat Gunakan Tag Alokasi Biaya dalam AWS BillingPanduan Pengguna.
- Identifikasi dan organisir sumber daya AWS Anda. Banyak layanan AWS yang mendukung penandaan, sehingga Anda dapat menetapkan tanda yang sama ke sumber daya dari layanan yang berbeda untuk menunjukkan bahwa sumber daya tersebut terkait. Misalnya, Anda dapat menugaskan tanda yang sama ke tahap API Gateway yang Anda tetapkan ke aturan CloudWatch Events.

Untuk tips menggunakan tanda, lihat postingan AWS Strategi Penandaan di blog AWS Jawaban.

Bagian berikut menyediakan informasi selengkapnya tentang tag untuk Amazon API Gateway

Topik

- Sumber daya API Gateway yang dapat ditandai
- Menggunakan tanda untuk mengontrol API Gateway sumber daya

Sumber daya API Gateway yang dapat ditandai

Tag dapat diatur pada API HTTP atau sumber daya WebSocket API berikut di Amazon API Gateway V2 API:

- `Api`
- `DomainName`
- `Stage`
- `VpcLink`

Selain itu, tag dapat diatur pada sumber daya REST API berikut di Amazon API Gateway V1 API:

- `ApiKey`
- `ClientCertificate`
- `DomainName`
- `RestApi`
- `Stage`
- `UsagePlan`
- `VpcLink`

Tag tidak dapat disetel langsung pada sumber daya lain. Namun, dalam API Amazon API Gateway V1, sumber daya turunan mewarisi tag yang ditetapkan pada sumber daya induk. Misalnya:

- Jika tag disetel pada `RestApi` sumber daya, tag tersebut diwarisi oleh sumber daya turunan berikut `RestApi` untuk kontrol akses berbasis atribut:
  - `Authorizer`
  - `Deployment`
  - `Documentation`
  - `GatewayResponse`
  - `Integration`
  - `Method`
  - `Model`
  - `Resource`

- `ResourcePolicy`
- `Setting`
- `Stage`

- Jika tag diatur pada`DomainName`, tag tersebut diwarisi oleh `BasePathMapping` sumber daya apa pun di bawahnya.
- Jika tag diatur pada`UsagePlan`, tag tersebut diwarisi oleh `UsagePlanKey` sumber daya apa pun di bawahnya.

> **Note**
>
> Warisan tag hanya berlaku untuk kontrol akses berbasis atribut. Misalnya, Anda tidak dapat menggunakan tag yang diwariskan AWS Cost Explorer API Gateway tidak mengembalikan tag yang diwariskan saat Anda memanggil GetTagssumber daya.

Warisan tag di API Amazon API Gateway V1

Sebelumnya itu hanya mungkin untuk mengatur tag pada tahap. Sekarang bahwa Anda juga dapat mengatur mereka pada sumber daya lain, `Stage` dapat menerima tag dua cara:

- Tag dapat diatur langsung pada`Stage`.
- Panggung dapat mewarisi tag dari `RestApi` induknya.

Jika sebuah panggung menerima tag dua arah, tag yang ditetapkan langsung di atas panggung akan diutamakan. Misalnya, sebuah stage mewarisi tag berikut dari REST API induknya:

```
{
 'foo': 'bar',
 'x':'y'
}
```

Misalkan juga memiliki tag berikut yang ditetapkan di atasnya secara langsung:

```
{
 'foo': 'bar2',
 'hello': 'world'
```

```
}
```

Efek bersih akan untuk tahap untuk memiliki tag berikut, dengan nilai-nilai berikut:

```
{
 'foo': 'bar2',
 'hello': 'world'
 'x':'y'
}
```

Pembatasan dan penggunaan

Pembatasan dan penggunaan berikut berlaku untuk penggunaan tag dengan sumber API Gateway:

- Setiap sumber daya dapat memiliki maksimum tanda 50.
- Untuk setiap sumber daya, setiap tanda kunci harus unik, dan setiap tanda kunci hanya dapat memiliki satu nilai.
- Panjang tanda kunci maksimum adalah 128 karakter Unicode dalam UTF-8.
- Panjang tanda nilai maksimum adalah 256 karakter Unicode dalam UTF-8.
- Karakter yang diperbolehkan untuk kunci dan nilai adalah huruf, angka, spasi yang dapat diwakili dalam UTF-8, dan karakter berikut: .: + = @ _/- (tanda penghubung). Sumber daya Amazon EC2 memungkinkan karakter apa pun.
- Kunci dan nilai tanda peka huruf besar dan kecil. Sebagai praktik terbaik, putuskan strategi untuk memanfaatkan tag dan terapkan strategi tersebut secara konsisten di semua jenis sumber daya. Misalnya, putuskan apakah akan menggunakan `Costcenter`, `costcenter`, atau `CostCenter` dan menggunakan kesepakatan yang sama untuk semua tag. Hindari penggunaan tag yang serupa dengan perlakuan kasus yang tidak konsisten.
- `aws:`Awalan dilarang untuk tag; itu diperuntukkan untuk AWS penggunaan. Anda tidak dapat mengedit atau menghapus kunci atau nilai tanda dengan prefiks ini. Tanda dengan prefiks ini tidak dihitung, berlawanan dengan tanda milik Anda per batas sumber daya.

Menggunakan tanda untuk mengontrol API Gateway sumber daya

Syarat dalam AWS Identity and Access Management kebijakan adalah bagian dari sintaks yang Anda gunakan untuk menentukan izin ke sumber API Gateway. Untuk detail tentang menentukan

kebijakan IAM, lihat. the section called “Gunakan izin IAM” Di API Gateway, sumber daya dapat memiliki tag, dan beberapa tindakan dapat mencakup tanda. Saat membuat kebijakan IAM, Anda dapat menggunakan kunci kondisi tag untuk mengontrol:

- Manakah pengguna API Gateway dapat melakukan tindakan pada sumber daya
- Tanda apa yang dapat diteruskan dalam permintaan tindakan.
- Apakah kunci tanda tertentu dapat digunakan dalam permintaan.

Menggunakan tag untuk kontrol akses berbasis atribut dapat memungkinkan kontrol yang lebih baik daripada kontrol tingkat API, serta kontrol yang lebih dinamis daripada kontrol akses berbasis sumber daya. Kebijakan IAM dapat dibuat yang mengizinkan atau melarang operasi berdasarkan tag yang disediakan dalam permintaan (tag permintaan), atau tag pada sumber daya yang sedang dioperasikan (tag sumber daya). Secara umum, tag sumber daya adalah untuk sumber daya yang sudah ada. Tanda yang diperlukan adalah saat Anda membuat sumber daya.

Untuk sintaks dan semantik kunci syarat tanda yang lengkap, lihat Controlling Access Using Tags dalam Panduan Pengguna IAM.

Contoh berikut ini menunjukkan cara menentukan syarat tag dalam kebijakan untuk pengguna API Gateway.

Batasi tindakan berdasarkan tanda sumber daya

Kebijakan contoh berikut memberi pengguna izin untuk melakukan semua tindakan pada semua sumber daya, asalkan sumber daya tersebut tidak memiliki tag `Environment` dengan nilai`prod`.

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

```
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Environment": "prod"
        }
      }
    }
  ]
}
```

Izinkan tindakan berdasarkan tanda sumber daya

Contoh kebijakan berikut memungkinkan pengguna untuk melakukan semua tindakan pada sumber daya API Gateway, asalkan sumber daya memiliki tag `Environment` dengan nilai`Development`. `Deny`Pernyataan ini mencegah pengguna dari mengubah nilai `Environment` tag.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ConditionallyAllow",
      "Effect": "Allow",
      "Action": [
        "apigateway:*"
      ],
      "Resource": [
        "arn:aws:apigateway:*::*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Environment": "Development"
        }
      }
    },
    {
      "Sid": "AllowTagging",
      "Effect": "Allow",
      "Action": [
        "apigateway:*"
      ],
      "Resource": [
        "arn:aws:apigateway:*::/tags/*"
      ]
    },
```

```
    {
      "Sid": "DenyChangingTag",
      "Effect": "Deny",
      "Action": [
        "apigateway:*"
      ],
      "Resource": [
        "arn:aws:apigateway:*::/tags/*"
      ],
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:TagKeys": "Environment"
        }
      }
    }
  ]
}
```

Operasi pemberian tag ke sesi

Kebijakan contoh berikut memungkinkan pengguna untuk melakukan semua tindakan API Gateway, kecuali untuk mengubah tag.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:*"
      ],
      "Resource": [
        "*"
      ],
    },
    {
      "Effect": "Deny",
      "Action": [
        "apigateway:*"
      ],
      "Resource": "arn:aws:apigateway:*::/tags*",
    }
  ]
```

```
}
```

Izinkan operasi pemberian tag

Contoh kebijakan berikut memungkinkan pengguna untuk mendapatkan semua sumber daya API Gateway, dan mengubah tag untuk sumber daya tersebut. Untuk mendapatkan tag untuk sumber daya, pengguna harus memiliki `GET` izin untuk sumber daya tersebut. Untuk memperbarui tag sumber daya, pengguna harus memiliki `PATCH` izin untuk sumber daya tersebut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:GET",
        "apigateway:PUT",
        "apigateway:POST",
        "apigateway:DELETE"
      ],
      "Resource": [
        "arn:aws:apigateway:*::/tags/*",
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:GET",
        "apigateway:PATCH",
      ],
      "Resource": [
        "arn:aws:apigateway:*::*",
      ]
    }
  ]
}
```

Referensi API

Amazon API Gateway menyediakan API untuk membuat dan menerapkan HTTP Anda sendiri dan WebSocketAPI. Selain itu, API Gateway API tersedia dalam standarAWSSDK.

Jika Anda menggunakan bahasa yangAWSSDK ada, Anda mungkin lebih suka menggunakan SDK daripada menggunakan API REST API Gateway secara langsung. SDK membuat autentikasi menjadi lebih sederhana, berintegrasi dengan mudah dengan lingkungan pengembangan Anda, dan menyediakan akses mudah ke perintah API Gateway.

Di sinilah menemukanAWSDokumentasi referensi API REST SDK dan API Gateway

- Alat untuk Amazon Web Services
- Referensi API REST Amazon API Gateway
- Amazon API Gateway WebSocket Referensi API HTTP

Kuota Amazon API Gateway dan catatan penting

Topik

Kecuali disebutkan lain, kuota dapat ditingkatkan berdasarkan permintaan. Untuk meminta kenaikan kuota, Anda dapat menggunakan Service Quotas atau menghubungi AWSSupport Center.

Ketika otorisasi diaktifkan pada metode, panjang maksimum ARN metode (misalnya`arn:aws:execute-api:{region-id}:{account-id}:{api-id}/{stage-id}/{method}/{resource}/{path}`,) adalah 1600 byte. Nilai parameter jalur (yang ukurannya ditentukan saat runtime) dapat menyebabkan panjang ARN melebihi batas. Ketika ini terjadi, klien API menerima `414 Request URI too long` respons.

> **Note**
>
> Ini membatasi panjang URI saat kebijakan sumber daya digunakan. Dalam kasus API pribadi di mana kebijakan sumber daya diperlukan, ini membatasi panjang URI semua API pribadi.

Kuota tingkat akun API Gateway, per Wilayah

Kuota berikut berlaku per akun, per Wilayah di Amazon API Gateway.

Sumber daya atau operasi	Kuota default	Dapat ditingkatkan
Throttle kuota per akun, per Wilayah di seluruh API HTTP, REST API, API, dan WebSocket API callback WebSocket	10.000 permintaan per detik (RPS) dengan kapasitas burst tambahan yang disediakan oleh algoritma token bucket, menggunakan kapasitas bucket maksimum 5.000 permintaan. * **Note** Kuota burst ditentukan oleh tim layanan API Gateway berdasarkan keseluruhan kuota RPS untuk akun di Wilayah. Ini bukan kuota yang dapat dikendalikan atau diminta oleh pelanggan untuk melakukan perubahan.	Ya
API Regional	600	Tidak
API yang dioptimalkan untuk edge	120	Tidak

* Untuk Wilayah Afrika (Cape Town) dan Eropa (Milan), kuota throttle default adalah 2500 RPS dan kuota burst default adalah 1250 RPS.

Kuota HTTP API

Kuota berikut berlaku untuk mengonfigurasi dan menjalankan API HTTP di API Gateway.

Sumber daya atau operasi	Kuota default	Dapat ditingkatkan
Rute per API	300	Ya
Integrasi per API	300	Tidak

Sumber daya atau operasi	Kuota default	Dapat ditingkatkan
Batas waktu integrasi maksimum	30 detik	Tidak
Stage per API	10	Ya
Pemetaan API multi-level per domain	200	Tidak
Tag per tahap	50	Tidak
Total ukuran gabungan dari baris permintaan dan nilai header	10240 byte	Tidak
Ukuran muatan	10 MB	Tidak
Domain kustom per akun per Wilayah	120	Ya
Akses ukuran templat log	3 KB	Tidak
Entri CloudWatch log Amazon Logs	1 MB	Tidak
Pengotorisasi untuk API	10	Ya
Pemirsa per otorisasi	50	Tidak
Lingkup per rute	10	Tidak

Sumber daya atau operasi	Kuota default	Dapat ditingkatkan
Batas waktu untuk titik akhir Set Kunci Web JSON	1500 ms	Tidak
Ukuran respons dari titik akhir JSON Web Key Set	150000 byte	Tidak
Batas waktu untuk titik akhir penemuan OpenID Connect	1500 ms	Tidak
Batas waktu untuk respons otorisasi Lambda	10000 ms	Tidak
Tautan VPC per akun per Wilayah	10	Ya
Subnet untuk tautan VPC	10	Ya
Variabel stage per stage	100	Tidak
Panjang, dalam karakter, dari kunci dalam variabel tahap	64	Tidak

Sumber daya atau operasi	Kuota default	Dapat ditingkatkan
Panjang, dalam karakter, dari nilai dalam variabel tahap	512	Tidak

Kuota API Gateway untuk mengonfigurasi dan menjalankan API WebSocket

Kuota berikut berlaku untuk mengonfigurasi dan menjalankan WebSocket API di Amazon API Gateway.

Sumber daya atau operasi	Kuota default	Dapat ditingkatkan
Koneksi baru per detik per akun (di semua WebSocket API) per Wilayah	500	Ya
Koneksi bersamaan	Tidak berlaku*	Tidak berlaku
AWS Lambdaotorisasi per API	10	Ya
AWS Lambdaukuran hasil otorisasi	8 KB	Tidak
Rute per API	300	Ya
Integrasi per API	300	Ya

Sumber daya atau operasi	Kuota default	Dapat ditingkatkan
Batas waktu integrasi	50 milidetik - 29 detik untuk semua jenis integrasi, termasuk Lambda, Lambda proxy, HTTP, HTTP, dan integrasi. AWS	Tidak
Stage per API	10	Ya
WebSocket ukuran bingkai	32 KB	Tidak
Ukuran muatan pesan	128 KB **	Tidak
Durasi koneksi untuk WebSocket API	2 jam	Tidak
Batas Waktu Koneksi Idle	10 menit	Tidak
Panjang, dalam karakter, URL untuk WebSocket API	4096	Tidak

* API Gateway tidak memberlakukan kuota pada koneksi bersamaan. Jumlah maksimum koneksi bersamaan ditentukan oleh tingkat koneksi baru per detik dan durasi koneksi maksimum dua jam. Misalnya, dengan kuota default 500 koneksi baru per detik, jika klien terhubung dengan kecepatan maksimum selama dua jam, API Gateway dapat melayani hingga 3.600.000 koneksi bersamaan.

** Karena kuota WebSocket ukuran bingkai 32 KB, pesan yang lebih besar dari 32 KB harus dibagi menjadi beberapa frame, masing-masing 32 KB atau lebih kecil. Ini berlaku untuk `@connections` perintah. Jika pesan yang lebih besar (atau ukuran bingkai yang lebih besar) diterima, koneksi ditutup dengan kode 1009.

Kuota API Gateway untuk mengonfigurasi dan menjalankan REST API

Kuota berikut berlaku untuk mengonfigurasi dan menjalankan REST API di Amazon API Gateway.

Sumber daya atau operasi	Kuota default	Dapat ditingkatkan
Nama domain khusus per akun per Wilayah	120	Ya
Pemetaan API multi-level per domain	200	Tidak
Panjang, dalam karakter, URL untuk API yang dioptimalkan tepi	8192	Tidak
Panjang, dalam karakter, URL untuk API regional	10240	Tidak
API pribadi per akun per Wilayah	600	Tidak
Panjang, dalam karakter, kebijakan sumber daya API Gateway	8192	Ya
Kunci API per akun per Wilayah	10000	Tidak

Sumber daya atau operasi	Kuota default	Dapat ditingkatkan
Sertifikat klien per akun per Wilayah	60	Ya
Otorisasi per API (AWS Lambdadan Amazon Cognito)	10	Ya
Bagian dokumentasi per API	2000	Ya
Sumber daya per API	300	Ya
Stage per API	10	Ya
Variabel stage per stage	100	Tidak
Panjang, dalam karakter, dari kunci dalam variabel tahap	64	Tidak
Panjang, dalam karakter, dari nilai dalam variabel tahap	512	Tidak
Paket penggunaan per akun per Wilayah	300	Ya
Paket penggunaan per kunci API	10	Ya

Sumber daya atau operasi	Kuota default	Dapat ditingkatkan
Tautan VPC per akun per Wilayah	20	Ya
API caching TTL	300 detik secara default dan dapat dikonfigurasi antara 0 dan 3600 oleh pemilik API.	Bukan untuk batas atas (3600)
Ukuran respons cache	1048576 Byte. Enkripsi data cache dapat meningkatkan ukuran item yang sedang di-cache.	Tidak
Batas waktu integrasi	50 milidetik - 29 detik untuk semua jenis integrasi, termasuk Lambda, Lambda proxy, HTTP, HTTP, dan integrasi. AWS	Bukan untuk batas bawah atau atas.
Total ukuran gabungan dari semua nilai header	10240 Byte	Tidak
Total ukuran gabungan dari semua nilai header untuk API pribadi	8000 Byte	Tidak
Ukuran muatan	10 MB	Tidak
Tag per tahap	50	Tidak

Sumber daya atau operasi	Kuota default	Dapat ditingkatkan
Jumlah iterasi dalam satu `#foreach ... #end` loop dalam template pemetaan	1000	Tidak
Panjang ARN dari metode dengan otorisasi	1600 byte	Tidak
Pengaturan pelambatan tingkat metode untuk tahap dalam rencana penggunaan	20	Ya
Ukuran model per API	400 KB	Tidak

Untuk restapi:import atau restapi:put, ukuran maksimum file definisi API adalah 6 MB.

Semua kuota per API hanya dapat ditingkatkan pada API tertentu.

Kuota API Gateway untuk membuat, menerapkan, dan mengelola API

Kuota tetap berikut berlaku untuk membuat, menerapkan, dan mengelola API di API Gateway, menggunakan, konsol API GatewayAWS CLI, atau API Gateway REST API dan SDK-nya. Kuota ini tidak dapat ditingkatkan.

Action	Kuota default	Dapat ditingkatkan
CreateApiKey	5 permintaan per detik per akun	Tidak
CreateDeployment	1 permintaan setiap 5 detik per akun	Tidak
CreateDocumentationVersion	1 permintaan setiap 20 detik per akun	Tidak
CreateDomainName	1 permintaan setiap 30 detik per akun	Tidak
CreateResource	5 permintaan per detik per akun	Tidak
CreateRestApi	API regional atau pribadi • 1 permintaan setiap 3 detik per akun API yang dioptimalkan tepi • 1 permintaan setiap 30 detik per akun	Tidak
CreateVpcLink(V2)	1 permintaan setiap 15 detik per akun	Tidak
DeleteApiKey	5 permintaan per detik per akun	Tidak
DeleteDomainName	1 permintaan setiap 30 detik per akun	Tidak
DeleteResource	5 permintaan per detik per akun	Tidak

Action	Kuota default	Dapat ditingkatkan
DeleteRestApi	1 permintaan setiap 30 detik per akun	Tidak
GetResources	5 permintaan setiap 2 detik per akun	Tidak
DeleteVpcLink(V2)	1 permintaan setiap 30 detik per akun	Tidak
ImportDocumentationParts	1 permintaan setiap 30 detik per akun	Tidak
ImportRestApi	API regional atau pribadi • 1 permintaan setiap 3 detik per akun API yang dioptimalkan tepi • 1 permintaan setiap 30 detik per akun	Tidak
PutRestApi	1 permintaan per detik per akun	Tidak
UpdateAccount	1 permintaan setiap 20 detik per akun	Tidak
UpdateDomainName	1 permintaan setiap 30 detik per akun	Tidak
UpdateUsagePlan	1 permintaan setiap 20 detik per akun	Tidak
Operasi lainnya	Tidak ada kuota hingga total kuota akun.	Tidak

Action	Kuota default	Dapat ditingkatkan
Total operasi	10 permintaan per detik dengan kuota burst 40 permintaan per detik.	Tidak

Catatan penting Amazon API Gateway

Topik

- Catatan penting Amazon API Gateway untuk REST dan WebSocket API
- Catatan penting Amazon API Gateway untuk WebSocket API
- Catatan penting Amazon API Gateway untuk REST API

Catatan penting Amazon API Gateway untuk REST dan WebSocket API

- API Gateway tidak mendukung berbagi nama domain khusus di seluruh REST dan WebSocket API.
- Nama panggung hanya dapat berisi karakter alfanumerik, tanda hubung, dan garis bawah. Panjang maksimum adalah 128 karakter.
- `/sping`Jalur `/ping` dan disediakan untuk pemeriksaan kesehatan layanan. Penggunaan ini untuk sumber daya tingkat root API dengan domain khusus akan gagal menghasilkan hasil yang diharapkan.
- API Gateway saat ini membatasi peristiwa log hingga 1024 byte. Peristiwa log yang lebih besar dari 1024 byte, seperti badan permintaan dan respons, akan dipotong oleh API Gateway sebelum dikirim ke Log. CloudWatch
- CloudWatch Metrik saat ini membatasi nama dan nilai dimensi hingga 255 karakter XHTML yang valid. (Untuk informasi selengkapnya, lihat Panduan CloudWatch Pengguna.) Nilai dimensi adalah fungsi dari nama yang ditentukan pengguna, termasuk nama API, nama label (tahap), dan nama sumber daya. Saat memilih nama-nama ini, berhati-hatilah untuk tidak melebihi batas CloudWatch Metrik.
- Ukuran maksimum template pemetaan adalah 300 KB.

Catatan penting Amazon API Gateway untuk WebSocket API

- API Gateway mendukung muatan pesan hingga 128 KB dengan ukuran bingkai maksimum 32 KB. Jika pesan melebihi 32 KB, Anda harus membaginya menjadi beberapa frame, masing-masing 32 KB atau lebih kecil. Jika pesan yang lebih besar diterima, koneksi ditutup dengan kode 1009.

Catatan penting Amazon API Gateway untuk REST API

- Karakter pipa teks biasa (`|`) tidak didukung untuk string kueri URL permintaan apa pun dan harus dikodekan URL.
- Karakter titik koma (`;`) tidak didukung untuk string kueri URL permintaan apa pun dan menghasilkan data yang dibagi. Secara umum, REST API mendekode parameter permintaan yang disandikan URL sebelum meneruskannya ke integrasi backend.
- Saat menggunakan konsol API Gateway untuk menguji API, Anda mungkin mendapatkan respons “kesalahan titik akhir tidak dikenal” jika sertifikat yang ditandatangani sendiri ditampilkan ke backend, sertifikat perantara hilang dari rantai sertifikat, atau pengecualian terkait sertifikat lain yang tidak dapat dikenali yang dilemparkan oleh backend.
- Untuk API `Resource`atau `Method`entitas dengan integrasi pribadi, Anda harus menghapusnya setelah menghapus referensi kode keras dari file. `VpcLink` Jika tidak, Anda memiliki integrasi yang menggantung dan menerima kesalahan yang menyatakan bahwa tautan VPC masih digunakan bahkan ketika entitas `Resource` atau `Method` dihapus. Perilaku ini tidak berlaku ketika integrasi pribadi mereferensikan `VpcLink` melalui variabel tahap.
- Backend berikut mungkin tidak mendukung otentikasi klien SSL dengan cara yang kompatibel dengan API Gateway:
  - NGINX
  - Heroku
- API Gateway mendukung sebagian besar spesifikasi OpenAPI 2.0 dan spesifikasi OpenAPI 3.0, dengan pengecualian berikut:
  - Segmen jalur hanya dapat berisi karakter alfanumerik, garis bawah, tanda hubung, titik, koma, titik dua, dan kurawal kurawal. Parameter jalur harus segmen jalur terpisah. Misalnya, “resource/{path_parameter_name}” valid; “resource {path_parameter_name}” tidak.
  - Nama model hanya dapat berisi karakter alfanumerik.
  - Untuk parameter input, hanya atribut berikut yang didukung:`name`,`in`,`required`,`type`,`description`. Atribut lainnya diabaikan.

- `securitySchemes`Jenisnya, jika digunakan, harus`apiKey`. Namun, otentikasi OAuth 2 dan HTTP Basic didukung melalui otorisasi Lambda; konfigurasi OpenAPI dicapai melalui ekstensi vendor.
- `deprecated`Bidang tidak didukung dan dijatuhkan di API yang diekspor.
- Model API Gateway didefinisikan menggunakan skema JSON draft 4, bukan skema JSON yang digunakan oleh OpenAPI.
- `discriminator`Parameter tidak didukung dalam objek skema apa pun.
- `example`Tag tidak didukung.
- `exclusiveMinimum`tidak didukung oleh API Gateway.
- `minItems`Tag `maxItems` dan tidak termasuk dalam validasi permintaan sederhana. Untuk mengatasinya, perbarui model setelah impor sebelum melakukan validasi.
- `oneOf`tidak didukung untuk OpenAPI 2.0 atau pembuatan SDK.
- `readOnly`Bidang tidak didukung.
- `$ref`tidak dapat digunakan untuk referensi file lain.
- Definisi respons `"500": {"$ref": "#/responses/UnexpectedError"}` formulir tidak didukung di root dokumen OpenAPI. Untuk mengatasinya, ganti referensi dengan skema inline.
- Nomor `Int64` tipe `Int32` atau tidak didukung. Contoh ditunjukkan sebagai berikut:

```
"elementId": {
    "description": "Working Element Id",
    "format": "int32",
    "type": "number"
}
```

- Jenis format angka desimal (`"format": "decimal"`) tidak didukung dalam definisi skema.
- Dalam respons metode, definisi skema harus dari tipe objek dan tidak dapat tipe primitif. Misalnya, `"schema": { "type": "string"}` tidak didukung. Namun, Anda dapat mengatasinya menggunakan jenis objek berikut:

```
"schema": {
     "$ref": "#/definitions/StringResponse"
            }

 "definitions": {
    "StringResponse": {
      "type": "string"
```

```
    }
  }
```

- API Gateway tidak menggunakan keamanan tingkat root yang ditentukan dalam spesifikasi OpenAPI. Oleh karena itu keamanan perlu didefinisikan pada tingkat operasi agar diterapkan dengan tepat.
- Kata `default` kunci tidak didukung.

- API Gateway memberlakukan batasan dan batasan berikut saat menangani metode dengan integrasi Lambda atau integrasi HTTP.
  - Nama header dan parameter kueri diproses dengan cara yang peka huruf besar/kecil.
  - Tabel berikut mencantumkan header yang mungkin dihapus, dipetakan ulang, atau diubah saat dikirim ke titik akhir integrasi atau dikirim kembali oleh titik akhir integrasi Anda. Dalam tabel ini:
    - `Remapped`berarti bahwa nama header diubah dari *`<string>`* menjadi`X-Amzn-Remapped-`*`<string>`*.

      `Remapped Overwritten`berarti bahwa nama header diubah dari *`<string>`* ke `X-Amzn-Remapped-`*`<string>`* dan nilainya ditimpa.

Nama header	Permintaan (**`http`**/**`http_proxy`** /**`lambda`**)	Tanggapan (**`http`**/**`http_proxy`** /**`lambda`**)
`Age`	Passthrough	Passthrough
`Accept`	Passthrough	Jatuh/Passthrough/Passthrough
`Accept-Charset`	Passthrough	Passthrough
`Accept-Encoding`	Passthrough	Passthrough

Nama header	Permintaan (**http**/**http_proxy** /**lambda**)	Tanggapan (**http**/**http_pr y** /**lambda**)
`Authoriza tion`	Passthrough *	Dipetakan ulang
`Connection`	Passthrough/Passthrough/Drop	Dipetakan ulang
`Content-E ncoding`	Passthrough/Dropped/Passthrough	Passthrou gh
`Content-L ength`	Passthrough (dihasilkan berdasarkan tubuh)	Passthrou gh
`Content-MD5`	Jatuh	Dipetakan ulang
`Content-Type`	Passthrough	Passthrou gh
`Date`	Passthrough	Ditimpa Ulang
`Expect`	Jatuh	Jatuh
`Host`	Ditimpa ke titik akhir integrasi	Jatuh
`Max-Forwards`	Jatuh	Dipetakan ulang
`Pragma`	Passthrough	Passthrou gh
`Proxy-Aut henticate`	Jatuh	Jatuh
`Range`	Passthrough	Passthrou gh

Nama header	Permintaan (**`http`**/**`http_proxy`** /**`lambda`**)	Tanggapan (**`http`**/**`http_proxy`** /**`lambda`**)
`Referer`	Passthrough	Passthrough
`Server`	Jatuh	Ditimpa Ulang
`TE`	Jatuh	Jatuh
`Transfer-Encoding`	Jatuh/Jatuh/Pengecualian	Jatuh
`Trailer`	Jatuh	Jatuh
`Upgrade`	Jatuh	Jatuh
`User-Agent`	Passthrough	Dipetakan ulang
`Via`	Jatuh/Jatuh/Passthrough	Passthrough/ Dropped/Drop
`Warn`	Passthrough	Passthrough
`WWW-Authenticate`	Jatuh	Dipetakan ulang

* `Authorization` Header dijatuhkan jika berisi tanda tangan Versi Tanda Tangan 4 atau jika `AWS_IAM` otorisasi digunakan.

- Android SDK API yang dihasilkan oleh API Gateway menggunakan `java.net.HttpURLConnection` class tersebut. Kelas ini akan menampilkan pengecualian yang tidak tertangani, pada perangkat yang menjalankan Android 4.4 dan yang lebih lama, untuk

respons 401 yang dihasilkan dari pemetaan ulang header ke. `WWW-Authenticate` `X-Amzn-Remapped-WWW-Authenticate`

- Tidak seperti SDK API Java, Android, dan iOS yang dibuat oleh API Gateway, JavaScript SDK API yang dihasilkan oleh API Gateway tidak mendukung percobaan ulang untuk kesalahan tingkat 500.
- Pemanggilan pengujian metode menggunakan jenis konten default `application/json` dan mengabaikan spesifikasi jenis konten lainnya.
- Saat mengirim permintaan ke API dengan meneruskan `X-HTTP-Method-Override` header, API Gateway mengganti metode tersebut. Jadi untuk meneruskan header ke backend, header perlu ditambahkan ke permintaan integrasi.
- Jika permintaan berisi beberapa tipe media di `Accept` headernya, API Gateway hanya menghormati jenis `Accept` media pertama. Dalam situasi di mana Anda tidak dapat mengontrol urutan jenis `Accept` media dan jenis media konten biner Anda bukan yang pertama dalam daftar, Anda dapat menambahkan jenis `Accept` media pertama dalam `binaryMediaTypes` daftar API Anda, API Gateway akan mengembalikan konten Anda sebagai biner. Misalnya, untuk mengirim file JPEG menggunakan `<img>` elemen di browser, browser mungkin mengirim `Accept:image/webp,image/*,*/*;q=0.8` permintaan. Dengan menambahkan `image/webp` ke `binaryMediaTypes` daftar, titik akhir akan menerima file JPEG sebagai biner.
- Menyesuaikan respons gateway default untuk saat ini `413 REQUEST_TOO_LARGE` tidak didukung.
- API Gateway menyertakan `Content-Type` header untuk semua respons integrasi. Secara default, jenis kontennya adalah`application/json`.

Riwayat dokumen

Tabel berikut menjelaskan perubahan penting pada dokumentasi sejak rilis terakhir Amazon API Gateway. Untuk pemberitahuan tentang pembaruan dokumentasi ini, Anda dapat berlangganan umpan RSS dengan memilih tombol RSS di panel menu atas.

- Pembaruan dokumentasi terbaru: 10 Desember 2023

Perubahan	Deskripsi	Tanggal
Pembaruan API REST dan konsol WebSocket API	Informasi konsol yang diperbarui untuk REST API dan WebSocket API	Desember 10, 2023
Pembaruan dokumentasi	Memperbarui informasi konseptual dan membuat tutorial baru untuk transform asi data dan meminta topik validasi untuk API REST API Gateway API. Untuk informasi selengkapnya, lihat Menggunakan validasi permintaan di API Gateway dan Menyiapkan transformasi data untuk REST API.	22 Juni 2023
Konfigurasikan failover DNS untuk API Gateway Multi-wil ayah	Menambahkan dukungan untuk menggunakan pemeriksaan kesehatan Amazon Route 53 untuk mengontrol failover DNS dari API REST API Gateway API di primer Wilayah AWS ke satu di Wilayah sekunder. Untuk informasi selengkapnya, lihat Mengkonfigurasi pemeriksa	31 Oktober 2022

	an kesehatan khusus untuk failover DNS.	
Pembaruan dokumentasi	Ringkasan fitur inti yang diperbarui untuk REST API dan API HTTP API. Untuk informasi selengkapnya, lihat Memilih antara REST API dan API HTTP API.	31 Mei 2022
Pembaruan kebijakan terkelola	Menambahkan `acm:GetCertificate` dukungan pada `AWSServiceRoleForAPIGateway` kebijakan. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk API Gateway.	12 Juli 2021
Pemetaan parameter untuk HTTP API	Menambahkan dukungan untuk pemetaan parameter untuk API HTTP. Untuk informasi selengkapnya, lihat Mengubah permintaan dan respons API.	7 Januari 2021
Nonaktifkan titik akhir default untuk REST API	Menambahkan dukungan untuk menonaktifkan titik akhir default untuk REST API. Untuk informasi selengkapnya, lihat Menonaktifkan titik akhir default untuk REST API.	29 Oktober 2020

Otentikasi TLS timbal balik	Menambahkan dukungan untuk otentikasi TLS timbal balik untuk REST API dan HTTP API. Untuk informasi selengkapnya, lihat Mengonfigurasi autentikasi TLS timbal balik untuk REST API dan Mengonfigurasi otentikasi TLS timbal balik untuk API HTTP.	17 September 2020
AWS Lambda Otorisasi HTTP API	Menambahkan dukungan untuk AWS Lambda otorisasi untuk API HTTP. Untuk informasi selengkapnya, lihat Bekerja dengan AWS Lambda otorisasi untuk API HTTP.	9 September 2020
Integrasi AWS layanan HTTP API	Menambahkan dukungan untuk integrasi AWS layanan untuk API HTTP. Untuk informasi selengkapnya, lihat Bekerja dengan integrasi AWS layanan untuk API HTTP.	20 Agustus 2020
Domain kustom wildcard API HTTP	Menambahkan dukungan untuk nama domain khusus wildcard untuk API HTTP. Untuk informasi selengkapnya, lihat Nama domain kustom Wildcard.	10 Agustus 2020

Perbaikan portal pengembang tanpa server	Menambahkan manajemen pengguna ke panel administrator dan dukungan untuk mengekspor definisi API. Untuk informasi selengkapnya, lihat Menggunakan portal developer tanpa server untuk membuat katalog API Gateway API Anda.	25 Juni 2020
WebSocket `Sec-WebSocket-Protocol` Dukungan API	Menambahkan dukungan untuk `Sec-WebSocket-Protocol` lapangan. Untuk informasi selengkapnya, lihat Menyiapkan rute $connect yang memerlukan WebSocket subprotokol.	16 Juni 2020
Ekspor API HTTP	Menambahkan dukungan untuk mengekspor definisi OpenAPI 3.0 dari HTTP API. Untuk informasi selengkapnya, lihat Mengekspor API HTTP dari API Gateway.	20 April 2020
Dokumentasi keamanan	Menambahkan dokumentasi keamanan. Untuk informasi selengkapnya, lihat Keamanan di Amazon API Gateway.	31 Maret 2020
Dokumentasi direorganisasi	Menata ulang panduan pengembang.	12 Maret 2020
Ketersediaan umum HTTP API	Dirilis HTTP API dalam ketersediaan umum. Untuk informasi selengkapnya, lihat Bekerja dengan API HTTP.	12 Maret 2020

Pencatatan API HTTP	Menambahkan dukungan untuk `$context.integrationErrorMessage` di log API HTTP. Untuk informasi selengkapnya, lihat Variabel Pencatatan API HTTP.	26 Februari 2020
AWS variabel untuk impor OpenAPI	Menambahkan dukungan untuk AWS variabel dalam definisi OpenAPI. Untuk informasi selengkapnya, lihat AWS Variabel untuk Impor OpenAPI.	17 Februari 2020
HTTP API	Dirilis HTTP API dalam versi beta. Untuk informasi selengkapnya, lihat API HTTP.	4 Desember 2019
Nama domain kustom wildcard	Ditambahkan dukungan untuk nama domain kustom wildcard. Untuk informasi selengkapnya, lihat Nama Domain Kustom Wildcard.	Oktober 21, 2019
Pencatatan Amazon Data Firehose	Menambahkan dukungan untuk Amazon Data Firehose sebagai tujuan untuk mengakses data logging. Untuk informasi selengkapnya, lihat Menggunakan Amazon Data Firehose sebagai Tujuan untuk Pencatatan Akses Gateway API.	15 Oktober 2019

Alias Route53 untuk menjalankan API pribadi	Menambahkan dukungan untuk catatan DNS alias Route53 tambahan untuk menjalankan API pribadi. Untuk informasi selengkapnya, lihat Mengakses API Pribadi Anda Menggunakan Alias Route53.	18 September 2019
Kontrol akses berbasis tag untuk API WebSocket	Menambahkan dukungan untuk kontrol akses berbasis tag untuk WebSocket API. Untuk informasi selengkap nya, lihat Sumber Daya API Gateway yang Dapat Ditandai.	27 Juni 2019
Pilihan versi TLS untuk domain kustom	Menambahkan dukungan untuk pemilihan versi Transport Layer Security (TLS) untuk API yang diterapkan ke domain kustom. Lihat catatan di Pilih Versi TLS Minimum untuk Domain Kustom di API Gateway.	20 Juni 2019
Kebijakan titik akhir VPC untuk API pribadi	Menambahkan dukungan untuk meningkatkan keamanan API pribadi dengan melampirkan kebijakan titik akhir ke titik akhir VPC antarmuka. Untuk informasi selengkapnya, lihat Menggunakan Kebijakan Titik Akhir VPC untuk API Pribadi di API Gateway.	4 Juni 2019

Dokumentasi diperbarui	Menulis Ulang Memulai dengan Amazon API Gateway. Pindah tutorial ke Amazon API Gateway Tutorial.	29 Mei 2019
Kontrol akses berbasis tag untuk REST API	Menambahkan dukungan untuk kontrol akses berbasis tag untuk REST API. Untuk informasi selengkapnya, lihat Menggunakan Tag dengan Kebijakan IAM untuk Mengontrol Akses ke Sumber Daya API Gateway.	23 Mei 2019
Dokumentasi diperbarui	Menulis ulang topik 6: Apa itu Amazon API Gateway? , Tutorial: Membangun API dengan Integrasi Proxy HTTP, Tutorial: Membuat Calc REST API dengan Tiga Integrasi Non-Proxy, Template Pemetaan API Gateway dan Access Logging Variabel Referensi, Gunakan API Gateway Lambda Authorizers, dan Aktifkan CORS untuk API Gateway REST API Resource.	5 April 2019

Perbaikan portal pengembang tanpa server	Menambahkan panel administrator dan peningkatan lainnya untuk mempermudah penerbitan API di portal pengembang Amazon API Gateway. Untuk informasi selengkapnya, lihat Menggunakan Portal Pengembang untuk Katalog API Anda.	28 Maret 2019
Support untuk AWS Config	Menambahkan dukungan untuk AWS Config. Untuk informasi selengkapnya, lihat Memantau Konfigurasi API Gateway API dengan AWS Config.	20 Maret 2019
Support untuk AWS CloudFormation	Menambahkan API Gateway V2 API ke referensi AWS CloudFormation template. Untuk informasi selengkapnya, lihat Referensi Jenis Sumber Daya Amazon API Gateway V2.	7 Februari 2019
Support untuk WebSocket API	Menambahkan dukungan untuk WebSocket API. Untuk informasi selengkapnya, lihat Membuat WebSocket API di Amazon API Gateway.	18 Desember 2018

Portal pengembang tanpa server tersedia melalui AWS Serverless Application Repository	Aplikasi tanpa server portal pengembang Amazon API Gateway sekarang tersedia dari AWS Serverless Application Repository(selain GitHub). Untuk informasi selengkapnya, lihat Menggunakan Portal Pengembang untuk Katalog API Gateway API Anda.	16 Novbucket 2018
Support untuk AWS WAF	Ditambahkan dukungan untuk AWS WAF(Web Application Firewall). Untuk informasi selengkapnya, lihat Mengontrol Akses ke Menggunakan API AWS WAF.	5 November 2018
Portal pengembang tanpa server	Amazon API Gateway sekarang menyediakan portal pengembang yang sepenuhnya dapat disesuaikan sebagai aplikasi tanpa server yang dapat Anda terapkan untuk memublikasikan API Gateway API Anda. Untuk informasi selengkapnya, lihat Menggunakan Portal Pengembang untuk Katalog API Gateway API Anda.	29 Oktober 2018

Support untuk header multi-nilai dan parameter string kueri	Amazon API Gateway sekarang mendukung beberapa header dan parameter string kueri yang memiliki nama yang sama. Untuk informasi selengkapnya, lihat Support for Multi-Value Header dan Query String Parameters.	4 Oktober 2018
Dukungan OpenAPI	Amazon API Gateway sekarang mendukung OpenAPI 3.0 serta OpenAPI (Swagger) 2.0.	27 September 2018
Dokumentasi diperbarui	Menambahkan topik baru: Bagaimana Kebijakan Sumber Daya Amazon API Gateway Mempengaruhi Alur Kerja Otorisasi.	27 September 2018
AWS X-Ray Integrasi aktif	Sekarang Anda dapat menggunakan AWS X-Ray untuk melacak dan menganalisis latensi dalam permintaan pengguna saat mereka melakukan perjalanan melalui API Anda ke layanan yang mendasarinya. Untuk informasi selengkapnya, lihat Melacak Eksekusi API Gateway API dengan AWS X-Ray.	6 September 2018

Perbaikan caching	Hanya GET metode yang akan mengaktifkan caching secara default saat Anda mengaktifkan caching untuk tahap API. Ini membantu memastikan keamanan API Anda. Anda dapat mengaktif kan caching untuk metode lain dengan mengganti pengatura n metode. Untuk informasi selengkapnya, lihat Mengaktif kan Caching API untuk Meningkatkan Responsivitas.	20 Agustus 2018
Batas layanan direvisi	Beberapa batasan telah direvisi: Peningkatan jumlah API per akun. Peningkatan batas tarif API untuk Create/ Import/Deploy API. Mengoreks i beberapa tarif dari per menit ke per detik. Untuk informasi selengkapnya, lihat Batasan-b atasan.	13 Juli 2018
Mengganti parameter dan header permintaan dan respons API	Menambahkan dukungan untuk mengganti header permintaan, string kueri, dan jalur, serta header respons dan kode status. Untuk informasi selengkapnya, lihat Menggunakan Template Pemetaan untuk Mengganti Parameter dan Header Permintaan dan Respons API.	12 Juli 2018

Pelambatan tingkat metode untuk paket penggunaan	Menambahkan dukungan untuk menyetel batas pelambatan per metode default, serta menetapka n batas pelambatan untuk metode API individual dalam pengaturan paket penggunaan. Pengaturan ini merupakan tambahan dari pembatasan tingkat akun yang ada dan batas pelambata n tingkat metode default yang dapat Anda atur dalam pengaturan tahap. Untuk informasi selengkapnya, lihat Permintaan API Throttle untuk Throughput yang Lebih Baik.	11 Juli 2018
Pemberitahuan pembaruan Panduan Pengembang API Gateway sekarang tersedia melalui RSS	Versi HTML Panduan Pengembang API Gateway sekarang mendukung umpan RSS pembaruan yang didokumentasikan di halaman Riwayat Dokumen ini. Umpan RSS mencakup pembaruan yang dibuat 27 Juni 2018, dan yang lebih baru. Pembaruan yang diumumkan sebelumnya masih tersedia di halaman ini. Gunakan tombol RSS di panel menu atas untuk berlangga nan feed.	27 Juni 2018

Pembaruan sebelumnya

Tabel berikut menjelaskan perubahan penting dalam setiap rilis Panduan Pengembang API Gateway sebelum 27 Juni 2018.

Perubahan	Deskripsi	Tanggal diubah
API Privat	Menambahkan dukungan untuk API pribadi, yang Anda paparkan melalui titik akhir VPC antarmuka. Lalu lintas ke API pribadi Anda tidak meninggalkan jaringan Amazon; itu terisolasi dari internet publik.	14 Juni 2018
Otorisasi dan Integrasi Lambda Lintas Akun dan Otorisasi Kumpulan Pengguna Amazon Cognito Lintas Akun	Gunakan AWS Lambda fungsi dari AWS akun yang berbeda sebagai fungsi otorisasi Lambda atau sebagai backend integrasi API. Atau gunakan kumpulan pengguna Amazon Cognito sebagai otorisasi. Akun lain dapat berada di wilayah mana pun di mana Amazon API Gateway tersedia. Lihat informasi selengkapnya di the section called “Konfigurasikan otorisasi Lambda lintas akun”, the section called “Tutorial: Bangun API dengan integrasi proxy Lambda lintas akun”, dan the section called “Konfigur asikan otorisasi Amazon Cognito lintas akun untuk REST API”.	2 April 2018
Kebijakan Sumber Daya untuk API	Gunakan kebijakan sumber daya API Gateway untuk memungkinkan pengguna dari AWS akun lain mengakses API Anda dengan aman atau mengizinkan API dipanggil hanya dari rentang alamat IP sumber tertentu atau blok CIDR. Untuk informasi selengkapnya, lihat the section called “Menggunakan kebijakan sumber daya API Gateway”.	Selasa, 02 April 2018
Penandaan untuk sumber daya API Gateway	Tandai tahap API dengan hingga 50 tag untuk alokasi biaya permintaan API dan caching di API Gateway. Untuk informasi selengkapnya, lihat the section called “Siapkan tag”.	19 Desember 2017

Perubahan	Deskripsi	Tanggal diubah
Kompresi muatan dan dekompresi	Aktifkan panggilan API Anda dengan muatan terkompresi menggunakan salah satu pengkodean konten yang didukung. Muatan terkompresi tunduk pada pemetaan jika templat pemetaan tubuh ditentukan. Untuk informasi selengkapnya, lihat the section called “Pengkodean konten”.	19 Desember 2017
Kunci API bersumber dari otorisasi khusus	Kembalikan kunci API dari otorisasi khusus ke API Gateway untuk menerapkan rencana penggunaan metode API yang memerlukan kunci tersebut. Untuk informasi selengkapnya, lihat the section called “Pilih sumber kunci API”.	19 Desember 2017
Otorisasi dengan cakupan OAuth 2	Aktifkan otorisasi pemanggilan metode dengan menggunakan cakupan OAuth 2 dengan otorisasi. `COGNITO_USER_POOLS` Untuk informasi selengkapnya, lihat the section called “Gunakan kumpulan pengguna Amazon Cognito sebagai otorisasi untuk REST API”.	14 Desember 2017
Integrasi Pribadi dan Tautan VPC	Buat API dengan integrasi pribadi API Gateway untuk memberi klien akses ke sumber daya HTTP/HTTPS di VPC Amazon dari luar VPC melalui sumber daya. VpcLink Lihat informasi yang lebih lengkap di the section called “Tutorial: Membangun API dengan integrasi pribadi” dan the section called “Integrasi pribadi”.	30 November 2017
Menerapkan Canary untuk pengujian API	Tambahkan rilis canary ke penerapan API yang ada untuk menguji versi API yang lebih baru sambil menjaga versi saat ini beroperasi pada tahap yang sama. Anda dapat mengatur persentase lalu lintas tahap untuk rilis kenari dan mengaktifkan eksekusi khusus kenari dan akses yang dicatat di log Log terpisah. CloudWatch Untuk informasi selengkapnya, lihat the section called “Siapkan penerapan rilis kenari”.	Selasa, 28 Nopember 2017

Perubahan	Deskripsi	Tanggal diubah
Akses Pencatatan	Log akses klien ke API Anda dengan data yang berasal dari variabel $context dalam format yang Anda pilih. Untuk informasi selengkapnya, lihat the section called “CloudWatch log”.	21 November 2017
Ruby SDK dari sebuah API	Buat Ruby SDK untuk API Anda dan gunakan untuk menjalankan metode API Anda. Lihat informasi yang lebih lengkap di the section called “Menghasilkan Ruby SDK dari API” dan the section called “Menggunakan Ruby SDK yang dihasilkan oleh API Gateway untuk REST API”.	20 November 2017
Titik akhir API regional	Tentukan titik akhir API regional untuk membuat API bagi klien non-seluler. Klien non-seluler, seperti instans EC2, berjalan di AWS Wilayah yang sama tempat API diterapkan. Seperti halnya API yang dioptimalkan tepi, Anda dapat membuat nama domain khusus untuk API regional. Lihat informasi yang lebih lengkap di the section called “Menyiapkan API regional” dan the section called “Menyiapkan nama domain kustom regional”.	2 November 2017
Authorizer permintaan khusus	Gunakan otorisasi permintaan khusus untuk menyediakan informasi otentikasi pengguna dalam parameter permintaan untuk mengotorisasi panggilan metode API. Parameter permintaan mencakup header dan parameter string kueri serta variabel tahap dan konteks. Untuk informasi selengkapnya, lihat Gunakan otorisasi API Gateway Lambda.	15 September 2017

Perubahan	Deskripsi	Tanggal diubah
Menyesuaikan tanggapan gateway	Sesuaikan respons gateway yang dihasilkan API Gateway ke permintaan API yang gagal mencapai backend integrasi . Pesan gateway yang disesuaikan dapat memberi pemanggil pesan kesalahan khusus API, termasuk mengembalikan header CORS yang diperlukan, atau dapat mengubah data respons gateway ke format pertukaran eksternal. Untuk informasi selengkapnya, lihat Menyiapkan respons gateway untuk menyesuaikan respons kesalahan.	6 Juni 2017
Memetakan properti kesalahan kustom Lambda ke header respons metode	Petakan properti kesalahan kustom individual yang dikembalikan dari Lambda ke parameter header respons metode menggunakan `integration.respon se.body` parameter, mengandalkan API Gateway untuk deserialisasi objek kesalahan kustom yang dirangkai saat dijalankan. Untuk informasi selengkapnya, lihat Menangani kesalahan Lambda khusus di API Gateway.	6 Juni 2017
Batas pelambatan meningkat	Tingkatkan batas tingkat permintaan kondisi tunak tingkat akun menjadi 10.000 permintaan per detik (rps) dan batas bust menjadi 5000 permintaan bersamaan. Untuk informasi selengkapnya, lihat Permintaan Throttle API untuk throughput yang lebih baik.	6 Juni 2017
Memvalidasi permintaan metode	Konfigurasikan validator permintaan dasar pada level API atau level metode sehingga API Gateway dapat memvalida si permintaan yang masuk. API Gateway memverifikasi bahwa parameter yang diperlukan disetel dan tidak kosong, dan memverifikasi bahwa format muatan yang berlaku sesuai dengan model yang dikonfigurasi. Untuk informasi selengkapnya, lihat Gunakan validasi permintaan di API Gateway.	11 April 2017

Perubahan	Deskripsi	Tanggal diubah
Integrasi dengan ACM	Gunakan Sertifikat ACM untuk nama domain kustom API Anda. Anda dapat membuat sertifikat AWS Certificate Manager atau mengimpor sertifikat berformat PEM yang ada ke ACM. Anda kemudian merujuk ke ARN sertifikat saat menyetel nama domain khusus untuk API Anda. Untuk informasi selengkapnya, lihat Menyiapkan nama domain kustom untuk REST API.	Maret 9, 2017
Membuat dan memanggil Java SDK dari API	Biarkan API Gateway menghasilkan Java SDK untuk API Anda dan gunakan SDK untuk memanggil API di klien Java Anda. Untuk informasi selengkapnya, lihat Menggunakan Java SDK yang dihasilkan oleh API Gateway untuk REST API.	Januari 13, 2017
Integrasi dengan AWS Marketplace	Jual API Anda dalam paket penggunaan sebagai produk SaaS. AWS Marketplace Gunakan AWS Marketplace untuk memperluas jangkauan API Anda. AWS Marketplace Andalkan tagihan pelanggan atas nama Anda. Biarkan API Gateway menangani otorisasi pengguna dan pengukuran penggunaan. Untuk informasi selengkapnya, lihat Jual API Anda sebagai SaaS.	1 Desember 2016
Mengaktifkan Dukungan Dokumentasi untuk API Anda	Tambahkan dokumentasi untuk entitas API dalam DocumentationPartsumber daya di API Gateway. Kaitkan snapshot `DocumentationPart` instance koleksi dengan tahap API untuk membuat file. DocumentationVersion Publikasikan dokumentasi API dengan mengekspor versi dokumentasi ke file eksternal, seperti file Swagger. Untuk informasi selengkapnya, lihat Mendokumentasikan REST API.	1 Desember 2016

Perubahan	Deskripsi	Tanggal diubah
Authorizer kustom yang diperbarui	Fungsi Lambda otorisasi pelanggan sekarang mengembalikan pengenal utama pemanggil. Fungsi ini juga dapat mengembalikan informasi lain sebagai pasangan kunci-nilai `context` peta dan kebijakan IAM. Untuk informasi selengkapnya, lihat Keluaran dari otorisasi Lambda Amazon API Gateway.	1 Desember 2016
Mendukung muatan biner	Tetapkan binaryMediaTypespada API Anda untuk mendukung muatan biner permintaan atau respons. Tetapkan `contentHandling` properti pada Integrasi atau IntegrationResponseuntuk menentukan apakah akan menangani payload biner sebagai gumpalan biner asli, sebagai string Base64-enocded, atau sebagai passthrough tanpa modifikasi. Untuk informasi selengkapnya, lihat Bekerja dengan tipe media biner untuk REST API.	17 November 2016
Mengaktifkan integrasi proxy dengan backend HTTP atau Lambda melalui sumber daya proxy API.	Buat sumber daya proxy dengan parameter jalur serakah dari formulir `{proxy+}` dan metode catch-all`ANY`. Sumber daya proxy terintegrasi dengan backend HTTP atau Lambda masing-masing menggunakan integrasi proxy HTTP atau Lambda. Untuk informasi selengkapnya, lihat Menyiapkan integrasi proxy dengan sumber daya proxy.	20 September 2016
Memperluas API yang dipilih di API Gateway sebagai penawaran produk untuk pelanggan Anda dengan menyediakan satu atau beberapa paket penggunaan.	Buat paket penggunaan di API Gateway untuk memungkinkan klien API terpilih mengakses tahapan API tertentu dengan tarif dan kuota permintaan yang disepakati. Untuk informasi selengkapnya, lihat Membuat dan menggunakan paket penggunaan dengan kunci API.	11 Agustus, 2016

Perubahan	Deskripsi	Tanggal diubah
Mengaktifkan otorisasi tingkat metode dengan kumpulan pengguna di Amazon Cognito	Buat kumpulan pengguna di Amazon Cognito dan gunakan sebagai penyedia identitas Anda sendiri. Anda dapat mengonfigurasi kumpulan pengguna sebagai otorisasi tingkat metode untuk memberikan akses bagi pengguna yang terdaftar di kumpulan pengguna. Untuk informasi selengkapnya, lihat Kontrol akses ke REST API menggunakan kumpulan pengguna Amazon Cognito sebagai otorisasi.	28 Juli 2016
Mengaktifkan CloudWatch metrik dan dimensi Amazon di bawah namespace. `AWS/ApiGateway`	Metrik API Gateway sekarang distandarisasi di bawah namespace. CloudWatch `AWS/ApiGateway` Anda dapat melihatnya di konsol API Gateway dan CloudWatch konsol Amazon. Untuk informasi selengkapnya, lihat Dimensi dan metrik Amazon API Gateway.	28 Juli 2016
Mengaktifkan rotasi sertifikat untuk nama domain kustom	Rotasi sertifikat memungkinkan Anda untuk mengungga h dan memperbarui sertifikat kedaluwarsa untuk nama domain kustom. Untuk informasi selengkapnya, lihat Memutar sertifikat yang diimpor ke ACM.	27 April 2016
Mendokume ntasikan perubahan untuk konsol Amazon API Gateway yang diperbarui.	Pelajari cara membuat dan menyiapkan API menggunak an konsol API Gateway yang diperbarui. Lihat informasi yang lebih lengkap di Tutorial: Buat REST API dengan mengimpor contoh dan Tutorial: Membangun REST API dengan integrasi non-proxy HTTP.	5 April 2016

Perubahan	Deskripsi	Tanggal diubah
Mengaktifkan fitur Import API untuk membuat API baru atau memperbar ui API yang sudah ada dari definisi API eksternal.	Dengan fitur API Impor, Anda dapat membuat API baru atau memperbarui API yang sudah ada dengan mengunggah definisi API eksternal yang dinyatakan dalam Swagger 2.0 dengan ekstensi API Gateway. Untuk informasi selengkapnya tentang API Impor, lihatMengkonfigurasi REST API menggunakan OpenAPI.	5 April 2016
Mengekspos `$input.bo dy` variabel untuk mengakses muatan mentah sebagai string dan `$util.par seJson()` fungsi untuk mengubah string JSON menjadi objek JSON dalam template pemetaan.	Untuk informasi selengkapnya tentang `$input.body` dan `$util.parseJson()` , lihat Template pemetaan API Gateway dan referensi variabel pencatatan akses.	5 April 2016
Mengaktifkan permintaan klien dengan pembatala n cache tingkat metode, dan meningkatkan manajemen pembatasan permintaan.	Siram cache tingkat tahap API dan batalkan entri cache individual. Lihat informasi yang lebih lengkap di Siram cache tahap API di API Gateway dan Membatalkan entri cache API Gateway. Tingkatkan pengalaman konsol untuk mengelola pembatasan permintaan API. Untuk informasi selengkapnya, lihat Permintaan Throttle API untuk throughput yang lebih baik.	25 Maret 2016

Perubahan	Deskripsi	Tanggal diubah
Mengaktifkan dan memanggil API Gateway API menggunakan otorisasi khusus	Membuat dan mengkonfigurasi AWS Lambda fungsi untuk menerapkan otorisasi kustom. Fungsi ini mengembalikan dokumen kebijakan IAM yang memberikan izin Izinkan atau Tolak ke permintaan klien API Gateway API. Untuk informasi selengkapnya, lihat Gunakan otorisasi API Gateway Lambda.	Selasa, 11 Februari 2016
Mengimpor dan mengekspor API Gateway API menggunakan file definisi Swagger dan ekstensi	Buat dan perbarui API Gateway API Anda menggunak an spesifikasi Swagger dengan ekstensi API Gateway. Impor definisi Swagger menggunakan API Gateway Importer. Ekspor API Gateway API ke file definisi Swagger menggunakan konsol API Gateway atau API Gateway Export API. Lihat informasi yang lebih lengkap di Mengkonfi gurasi REST API menggunakan OpenAPI dan Ekspor REST API dari API Gateway.	18 Desember 2015
Memetakan permintaan atau badan respons atau bidang JSON badan untuk meminta atau menanggapi parameter.	Metode peta meminta badan atau bidang JSON ke dalam jalur permintaan integrasi, string kueri, atau header. Petakan badan respons integrasi atau bidang JSON-nya ke dalam header respons permintaan. Untuk informasi selengkapnya, lihat Referensi pemetaan data permintaan dan respons API Amazon API Gateway.	18 Desember 2015
Bekerja dengan Variabel Tahap di Amazon API Gateway	Pelajari cara mengaitkan atribut konfigurasi dengan tahap penerapan API di Amazon API Gateway. Untuk informasi selengkapnya, lihat Menyiapkan variabel tahap untuk penerapan REST API.	November 5, 2015
Cara: Aktifkan CORS untuk Metode	Sekarang lebih mudah untuk mengaktifkan berbagi sumber daya lintas asal (CORS) untuk metode di Amazon API Gateway. Untuk informasi selengkapnya, lihat Mengaktif kan CORS untuk sumber daya REST API.	November 3, 2015

Perubahan	Deskripsi	Tanggal diubah
Cara: Menggunakan Otentikasi SSL Sisi Klien	Gunakan Amazon API Gateway untuk menghasilkan sertifikat SSL yang dapat Anda gunakan untuk mengauten tikasi panggilan ke backend HTTP Anda. Untuk informasi selengkapnya, lihat Menghasilkan dan mengkonfigurasi sertifikat SSL untuk otentikasi backend.	September 22, 2015
Integrasi metode tiruan	Pelajari cara mengintegrasikan API tiruan dengan Amazon API Gateway. Fitur ini memungkinkan pengembang untuk menghasilkan respons API dari API Gateway secara langsung tanpa perlu backend integrasi akhir sebelumnya.	16 September 2015
Dukungan Identitas Amazon Cognito	Amazon API Gateway telah memperluas cakupan `$context` variabel sehingga sekarang mengembal ikan informasi tentang Identitas Amazon Cognito ketika permintaan ditandatangani dengan kredensyal Amazon Cognito. Selain itu, kami telah menambahkan `$util` variabel untuk melarikan diri karakter dan menyandikan URL JavaScript dan string. Untuk informasi selengkap nya, lihat Template pemetaan API Gateway dan referensi variabel pencatatan akses.	Agustus 28, 2015
Integrasi kesombongan	Gunakan alat impor Swagger GitHub untuk mengimpor definisi Swagger API ke Amazon API Gateway. Pelajari lebih lanjut Bekerja dengan ekstensi API Gateway ke OpenAPI cara membuat dan menerapkan API dan metode menggunakan alat impor. Dengan alat importir Swagger, Anda juga dapat memperbarui API yang ada.	21 Juli 2015
Referensi Template Pemetaan	Baca tentang `$input` parameter dan fungsinya diTemplate pemetaan API Gateway dan referensi variabel pencatatan akses.	18 Juli 2015
Rilis publik awal	Ini adalah rilis publik awal dari Panduan Pengembang API Gateway.	9 Juli 2015

AWSGlosarium

Untuk AWS terminologi terbaru, lihat AWSglosarium di Referensi. Glosarium AWS

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.