aws

Guida per gli sviluppatori

Gateway Amazon API



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

Gateway Amazon API: Guida per gli sviluppatori

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

I marchi e il trade dress di Amazon non possono essere utilizzati in relazione ad alcun prodotto o servizio che non sia di Amazon, in alcun modo che possa causare confusione tra i clienti, né in alcun modo che possa denigrare o screditare Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà dei rispettivi proprietari, che possono o meno essere affiliati, collegati o sponsorizzati da Amazon.

Table of Contents

Che cosa è Amazon API Gateway? ... 1
Architettura di API Gateway ... 2
Caratteristiche di API Gateway ... 2
Casi d'uso di API Gateway ... 3
Utilizzo di API Gateway per creare API REST ... 4
Utilizzo di API Gateway per creare API HTTP ... 4
Usa API Gateway per creare WebSocket API ... 5
Chi utilizza le API Gateway? ... 6
Accesso ad API Gateway ... 7
Parte di AWS infrastruttura senza server ... 7
Informazioni su come iniziare a utilizzare Amazon API Gateway ... 7
Concetti di API Gateway ... 8
Scelta tra REST API e API HTTP ... 14
Tipo di endpoint ... 14
Sicurezza ... 14
Autorizzazione ... 15
Gestione API ... 15
Sviluppo ... 16
Monitoraggio ... 17
Integrazioni ... 17
Nozioni di base sulla console REST API ... 18
Fase 1: creazione di una funzione Lambda ... 19
Fase 2: Creazione di una REST API ... 20
Fase 3: Creazione di un'integrazione proxy Lambda ... 20
Fase 4: Implementazione dell'API ... 21
Fase 5: Richiamo dell'API ... 21
(Opzionale) Fase 6: pulizia ... 22
Prerequisiti ... 24
Iscriviti per un Account AWS ... 24
Inizia a usare ... 25
Fase 1: creazione di una funzione Lambda ... 26
Fase 2: creazione un'API HTTP ... 26
Fase 3: test dell'API ... 27
(Opzionale) Fase 4: pulizia ... 28

Fasi successive ... 29
Tutorial e workshop ... 31
Tutorial sull'API REST ... 32
Scelta di un tutorial per l'integrazione Lambda ... 32
Tutorial: creazione di un'API REST mediante l'importazione di un esempio ... 56
Scelta di un tutorial per l'integrazione HTTP ... 66
Tutorial: creazione di una REST API con un'integrazione privata ... 80
Tutorial: crea un'API REST con un' AWS integrazione ... 86
Tutorial: creazione di una REST API di calcolo con tre integrazioni ... 92
Tutorial: creazione di una REST API come un proxy Amazon S3 ... 119
Tutorial: creazione di un'API REST come un proxy Amazon Kinesis ... 164
Tutorial: creare un'API REST utilizzando AWS SDK o AWS CLI ... 209
Tutorial: creazione di una REST API privata ... 243
Tutorial sull'API HTTP ... 249
Tutorial: compilazione di un'API HTTP CRUD con Lambda e DynamoDB ... 250
Tutorial: creazione di un'API HTTP con un'integrazione privata con Amazon ECS ... 262
Tutorial sull'API WebSocket ... 269
Tutorial: crea un'app di chat WebSocket ... 269
Tutorial: crea un' WebSocket API con un' AWS integrazione ... 275
REST API di Gateway API ... 288
Sviluppare ... 288
Risorsa di esempio per una REST API ... 289
Funzionalità aggiuntive per lo sviluppo di REST API ... 292
Tipi di endpoint di Gateway API ... 294
Policy di sicurezza ... 298
Tipi di indirizzo IP per REST API in Gateway API ... 320
Metodi ... 322
Controllo accessi ... 342
Integrazioni ... 429
Convalida delle richieste ... 523
Trasformazioni dei dati ... 554
Risposte del gateway ... 637
NUCLEI ... 649
Tipi di supporti binari ... 663
Invoke ... 696
OpenAPI ... 728

Pubblica 743
Implementazione di REST API 743
Nomi di dominio personalizzati 790
Ottimizza 863
Impostazioni cache 864
Codifica dei contenuti 877
Distribuzione 883
Piani di utilizzo 884
Documentazione API 912
Generazione di SDK 979
Vendi il tuo APIs SaaS 1005
Protezione 1010
Richiedi certificati client 1011
Certificati client di backend 1018
AWS WAF 1067
Throttling 1070
REST API private 1073
Monitoraggio 1130
CloudWatch metriche 1131
CloudWatch registri 1141
X-Ray 1170
Portali API Gateway 1184
Considerazioni 1184
Prodotto del portale 1185
Esempio di prodotto del portale sull'adozione di animali domestici 1185
Fasi successive 1187
Crea un prodotto del portale 1188
Crea un endpoint REST del prodotto 1190
Crea una pagina di prodotto 1192
Aggiornare un prodotto del portale 1194
Aggiorna l'endpoint REST di un prodotto 1195
Aggiorna la pagina del prodotto 1196
Condividi i prodotti del portale 1196
Abilita l'endpoint REST di prova per un prodotto nel tuo portale 1201
Eliminare un prodotto del portale 1203
Crea un portale 1203

Scegli un nome di dominio 1204
Crea un portale 1204
Aggiornare un portale 1207
Aggiorna un portale 1207
Visualizza l'anteprima di un portale 1207
Considerazioni 1207
Visualizzare l'anteprima di un portale 1208
Pubblica un portale 1208
Considerazioni 1208
Pubblica un portale 1208
Usa un portale 1209
Usa un portale 1209
Risolvi i problemi del tuo portale 1210
Disabilita un portale 1210
Disabilitare un portale 1210
Eliminare un portale 1211
Eliminare un portale 1211
API HTTP di Gateway API 1212
Sviluppare 1212
Creazione di un'API HTTP 1213
Percorsi 1214
Tipi di indirizzo IP per API HTTP in Gateway API 1217
Controllo accessi 1219
Integrazioni 1239
CORS 1258
Mappatura dei parametri 1261
OpenAPI 1268
Pubblicare 1278
Stage 1279
Politica di sicurezza per HTTP APIs in API Gateway 1281
Nomi di dominio personalizzati 1283
Protezione 1292
Throttling 1293
Autenticazione TLS reciproca. 1294
Monitoraggio 1301
Metriche 1301

Registrazione dei log ... 1304
Risoluzione dei problemi ... 1315
Integrazioni Lambda ... 1315
Provider di autorizzazioni JWT ... 1318
API WebSocket di Gateway API ... 1320
Panoramica delle API WebSocket ... 1320
Usa i percorsi per un' WebSocket API ... 1321
Invio dei dati alle app client connesse ... 1321
WebSocket Codici di stato delle API ... 1322
Gestione di utenti connessi e app client ... 1323
Invocazione dell'integrazione backend ... 1326
WebSocket espressioni di selezione ... 1331
Sviluppare ... 1341
Creazione e configurazione ... 1341
Tipi di indirizzo IP per API WebSocket in Gateway API ... 1343
Percorsi ... 1345
Controllo accessi ... 1353
Integrazioni ... 1362
Convalida delle richieste ... 1372
Trasformazioni dei dati ... 1375
Tipi di supporti binari ... 1388
Invoke ... 1388
Pubblicare ... 1392
Stage ... 1392
Implementa un'API WebSocket ... 1395
Politica di sicurezza per WebSocket APIs API Gateway ... 1398
Nomi di dominio personalizzati ... 1399
Protezione ... 1407
Limitazione a livello di account per regione ... 1408
Throttling a livello di instradamento ... 1408
Monitoraggio ... 1409
Metriche ... 1409
Registrazione dei log ... 1411
Gateway API ARNs ... 1420
API HTTP e risorse WebSocket API ... 1420
Risorse API REST ... 1423

Risorse nomi dominio ... 1427
execute-api(HTTP APIs e WebSocket APIs REST APIs) ... 1428
Estensioni OpenAPI ... 1430
x-amazon-apigateway-any-method ... 1431
Esempio di x-amazon-apigateway-any-method ... 1432
x-amazon-apigateway-cors ... 1433
Esempio x-amazon-apigateway-cors example ... 1433
x-amazon-apigateway-api-key-source ... 1434
Esempio di x-amazon-apigateway-api-key-source ... 1435
x-amazon-apigateway-auth ... 1436
Esempio di x-amazon-apigateway-auth ... 1436
x-amazon-apigateway-authorizer ... 1437
Esempi di x-amazon-apigateway-authorizer per API REST ... 1440
Esempi di x-amazon-apigateway-authorizer per API HTTP ... 1444
x-amazon-apigateway-authtype ... 1446
Esempio di x-amazon-apigateway-authtype ... 1446
Consulta anche ... 1448
Proprietà x-amazon-apigateway-binary-media-type ... 1448
Esempio di x-amazon-apigateway-binary-media-types ... 1449
x-amazon-apigateway-documentation ... 1449
Esempio di x-amazon-apigateway-documentation ... 1449
modalità x-amazon-apigateway-endpoint-access- ... 1450
esempio di x-amazon-apigateway-endpoint-access-mode ... 1450
x-amazon-apigateway-endpoint-configuration ... 1450
Esempi di x-amazon-apigateway-endpoint-configuration ... 1451
x-amazon-apigateway-gateway-responses ... 1452
Esempio di x-amazon-apigateway-gateway-responses ... 1452
x-amazon-apigateway-gateway-responses.gatewayResponse ... 1453
Esempio di x-amazon-apigateway-gateway-responses.gatewayResponse ... 1454
x-amazon-apigateway-gateway-responses.responseParameters ... 1454
Esempio di x-amazon-apigateway-gateway-responses.responseParameters ... 1455
x-amazon-apigateway-gateway-responses.responseTemplates ... 1455
Esempio di x-amazon-apigateway-gateway-responses.responseTemplates ... 1455
x-amazon-apigateway-importexport-version ... 1456
esempio di x-amazon-apigateway-importexport-version ... 1456
x-amazon-apigateway-integration ... 1456

Esempio di x-amazon-apigateway-integration ... 1464
x-amazon-apigateway-integrations ... 1466
Esempio x-amazon-apigateway-integrations ... 1466
x-amazon-apigateway-integration.requestTemplates ... 1468
Esempio di x-amazon-apigateway-integration.requestTemplates ... 1468
x-amazon-apigateway-integration.requestParameters ... 1469
Esempio di x-amazon-apigateway-integration.requestParameters ... 1470
x-amazon-apigateway-integration.responses ... 1471
Esempio di x-amazon-apigateway-integration.response ... 1472
x-amazon-apigateway-integration.response ... 1473
esempio di x-amazon-apigateway-integration.response ... 1474
x-amazon-apigateway-integration.responseTemplates ... 1474
Esempio di x-amazon-apigateway-integration.responseTemplate ... 1475
x-amazon-apigateway-integration.responseParameters ... 1475
Esempio di X-Amazon-ApiGateway-Integration.ResponseParameters ... 1476
x-amazon-apigateway-integration.tlsConfig ... 1476
Esempi di x-amazon-apigateway-integration.tlsConfig ... 1478
x-amazon-apigateway-minimum-compression-size ... 1479
x-amazon-apigateway-minimum-compression-size example ... 1479
x-amazon-apigateway-policy ... 1480
esempio di x-amazon-apigateway-policy ... 1480
x-amazon-apigateway-request-validator ... 1481
Esempio di x-amazon-apigateway-request-validator ... 1481
x-amazon-apigateway-request-validators ... 1482
esempio di x-amazon-apigateway-request-validators ... 1482
x-amazon-apigateway-request-validators.requestValidator ... 1483
Esempio di x-amazon-apigateway-request-validators.requestValidator ... 1483
politica di sicurezza x-amazon-apigateway- ... 1484
esempio di x-amazon-apigateway-security-policy ... 1484
x-amazon-apigateway-tag-value ... 1484
Esempio di x-amazon-apigateway-tag-value ... 1484
Sicurezza ... 1486
Protezione dei dati ... 1487
Crittografia dei dati ... 1488
Riservatezza del traffico Internet ... 1490
Identity and Access Management ... 1490

Destinatari 1491
Autenticazione con identità 1491
Gestione dell'accesso tramite policy 1492
Come funziona Amazon API Gateway con IAM 1494
Esempi di policy basate su identità 1499
Esempi di policy basate su risorse 1512
Risoluzione dei problemi 1512
Utilizzo di ruoli collegati ai servizi 1514
Logging e monitoraggio 1519
Lavorare con CloudTrail 1520
Lavorare con AWS Config 1523
Convalida della conformità 1527
Resilienza 1527
Sicurezza dell'infrastruttura 1528
Analisi della configurazione e delle vulnerabilità 1529
Best practice 1529
Applicazione di tag 1531
Risorse API Gateway a cui è possibile applicare tag 1532
Eredità dei tag nell'API di Amazon API Gateway V1 1533
Limitazioni applicate ai tag e convenzioni di utilizzo 1534
Attribute-based controllo degli accessi 1535
Limitare le operazioni in base ai tag delle risorse 1535
Consentire operazioni in base ai tag delle risorse 1536
Negare operazioni di assegnazione di tag 1537
Consentire operazioni di assegnazione di tag 1538
riferimenti API 1540
Endpoint del servizio Gateway API 1540
IPv4 estremità 1540
Endpoint Dualstack (e) IPv4 IPv6 1541
Specificazione degli endpoint 1541
Utilizzo IPv6 degli indirizzi nelle politiche IAM 1542
Quote e note importanti 1544
Quote a livello di account API Gateway per ogni regione 1544
Quote di API Gateway per la creazione, la distribuzione e la gestione di un'API 1545
Quote di una REST API 1549
Best practice per ridurre le quote 1555

Quote API HTTP 1557
Best practice per ridurre le quote 1560
Quote del portale 1560
WebSocket Quote API 1561
Note importanti 1564
Note importanti su Amazon API Gateway per HTTP APIs 1565
Note importanti su Amazon API Gateway per HTTP e WebSocket APIs 1565
Note importanti per REST APIs e WebSocket APIs 1565
Note importanti per WebSocket APIs 1566
Note importanti per REST APIs 1566
Cronologia dei documenti 1573
Aggiornamenti precedenti 1588
..... mdxcix

Che cosa è Amazon API Gateway?

Amazon API Gateway è un AWS servizio per la creazione, la pubblicazione, la manutenzione, il monitoraggio e la protezione di REST, HTTP e WebSocket API su qualsiasi scala. Gli sviluppatori di API possono creare API che accedono AWS o ad altri servizi Web, oltre ai dati archiviati nel cloud.AWS In qualità di sviluppatore di API di API Gateway, puoi creare API da usare nelle tue applicazioni client, oppure puoi rendere le tue API disponibili per gli sviluppatori di applicazioni di terze parti. Per ulteriori informazioni, consulta the section called “Chi utilizza le API Gateway?”.

API Gateway crea API RESTful che:

- Sono HTTP-based.
- Consentono la comunicazione client-server stateless.
- Implementano metodi HTTP standard come GET, POST, PUT, PATCH e DELETE.

Per ulteriori informazioni sulle API REST e sulle API HTTP API Gateway, consulta the section called “Scelta tra REST API e API HTTP ”, API HTTP di Gateway API, the section called “Utilizzo di API Gateway per creare API REST”, e the section called “Sviluppare”.

API Gateway crea WebSocket API che:

- Aderisci al WebSocketprotocollo, che consente una comunicazione full-duplex basata sullo stato tra client e server.
- Instradano i messaggi in ingresso e sono basate sul contenuto dei messaggi.

Per ulteriori informazioni sulle API WebSocket API Gateway, consulta the section called “Usa API Gateway per creare WebSocket API” ethe section called “Panoramica delle API WebSocket ”.

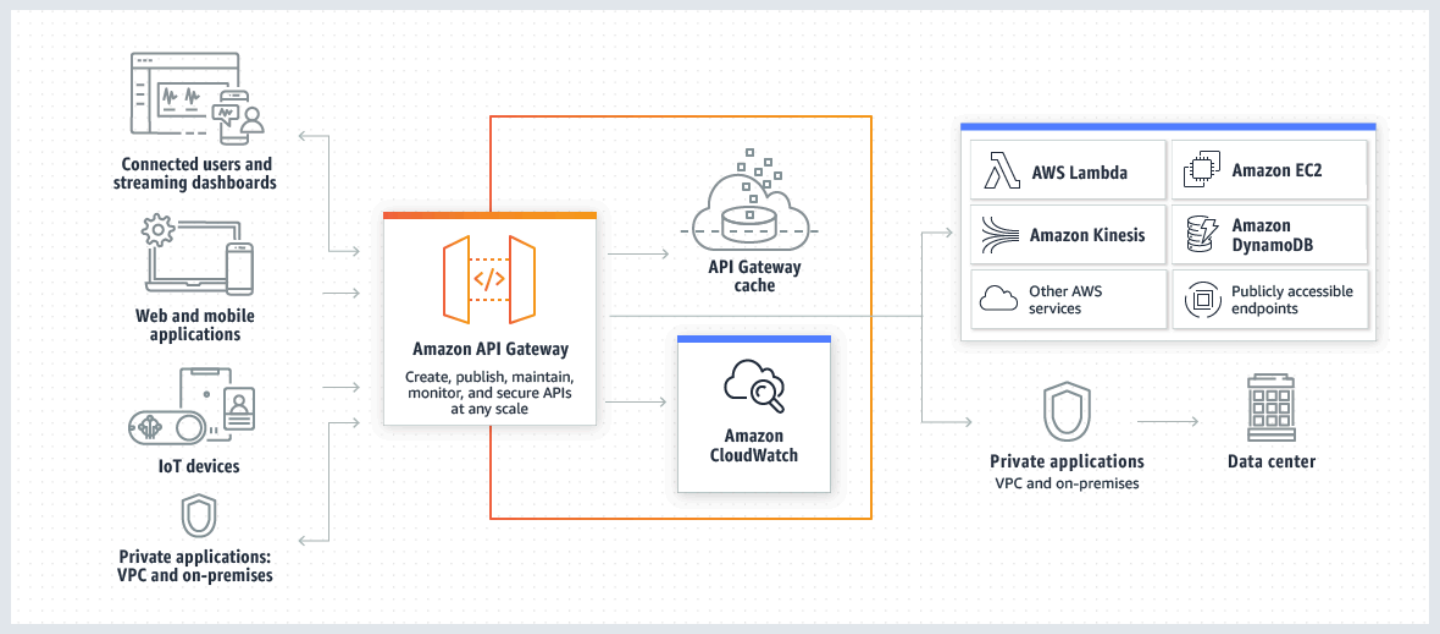
Argomenti

- Architettura di API Gateway
- Caratteristiche di API Gateway
- Casi d'uso di API Gateway
- Accesso ad API Gateway
- Parte di AWS infrastruttura senza server
- Informazioni su come iniziare a utilizzare Amazon API Gateway

- Concetti di Amazon API Gateway
- Scelta tra REST API e API HTTP
- Nozioni di base sulla console REST API

Architettura di API Gateway

Il diagramma seguente mostra l'architettura di API Gateway.



Questo diagramma illustra in che modo le API create in Amazon API Gateway offrono al cliente o agli sviluppatori un'esperienza di sviluppo integrata e coerente per la creazione di applicazioni AWS senza server. API Gateway gestisce tutte le attività di accettazione ed elaborazione relative a centinaia di migliaia di chiamate API simultanee. Queste attività includono la gestione del traffico, il controllo dell'autorizzazione e dell'accesso, il monitoraggio e la gestione delle versioni delle API.

API Gateway funge da «porta d'ingresso» per consentire alle applicazioni di accedere ai dati, alla logica di business o alle funzionalità dei servizi di backend, come i carichi di lavoro in esecuzione su Amazon Elastic Compute Cloud (Amazon EC2), il codice in esecuzione AWS Lambda su qualsiasi applicazione Web o applicazioni di comunicazione in tempo reale.

Caratteristiche di API Gateway

Amazon API Gateway offre caratteristiche come le seguenti:

- Support per API stateful (WebSocket) e stateless (HTTP e REST).
- Meccanismi di autenticazione potenti e flessibili, come AWS Identity and Access Management policy, funzioni di autorizzazione Lambda e pool di utenti di Amazon Cognito.
- Distribuzioni di versioni di Canary per l'implementazione sicura di modifiche.
- Logging e monitoraggio dell'utilizzo delle API e delle modifiche apportate alle API in CloudTrail.
- CloudWatch registrazione degli accessi e delle esecuzioni, inclusa la possibilità di impostare allarmi. Per ulteriori informazioni, consultare the section called “CloudWatch metriche” e the section called “Metriche”.
- Possibilità di utilizzare CloudFormation modelli per abilitare la creazione di API. Per ulteriori informazioni, consulta l'argomento relativo al riferimento ai tipi di risorse Amazon API Gateway e riferimento ai tipi di risorse Amazon API Gateway V2.
- Supporto per nomi di dominio personalizzati.
- Integrazione con AWS WAF per la protezione delle API contro gli exploit Web più comuni.
- Integrazione con AWS X-Ray per identificare e valutare le prestazioni in termini di latenza.

Per un elenco completo delle versioni delle caratteristiche di API Gateway, consulta Cronologia dei documenti.

Casi d'uso di API Gateway

La seguente sezione dedicata ai casi d’uso presenta una panoramica dei diversi tipi di API di Gateway API e delle diverse tipologie di sviluppatori che utilizzano Gateway API. Per informazioni più dettagliate sulla differenza tra REST API e API HTTP, consulta the section called “Scelta tra REST API e API HTTP ”.

Argomenti

- Utilizzo di API Gateway per creare API REST
- Utilizzo di API Gateway per creare API HTTP
- Usa API Gateway per creare WebSocket API
- Chi utilizza le API Gateway?

Utilizzo di API Gateway per creare API REST

Un'API REST API Gateway include risorse e metodi. Una risorsa è un'entità logica a cui un'app può accedere tramite un percorso della risorsa. Un metodo corrisponde a una richiesta API REST inviata dall'utente della tua API e alla risposta restituita all'utente.

Ad esempio, `/incomes` potrebbe essere il percorso di una risorsa che rappresenta il reddito dell'utente dell'app. Una risorsa può avere una o più operazioni definite dai verbi HTTP adeguati, come GET, POST, PUT, PATCH e DELETE. La combinazione tra percorso della risorsa e operazione identifica un metodo dell'API. Ad esempio, un metodo `POST /incomes` potrebbe aggiungere il reddito guadagnato dall'intermediario, mentre il metodo `GET /expenses` potrebbe eseguire una query relativa alle spese indicate sostenute dall'intermediario.

Non è necessario che l'app sappia dove i dati vengono archiviati e prelevati sul back-end. Nelle API REST API Gateway, il front-end è incapsulato dalle richieste di metodo e dalle risposte di metodo. L'API si interfaccia con il back-end tramite le richieste di integrazione e le risposte di integrazione.

Ad esempio, con DynamoDB come back-end, lo sviluppatore di API configura la richiesta di integrazione per inoltrare la richiesta di metodo in entrata al back-end scelto. La configurazione include le specifiche di un'operazione DynamoDB adeguata, i ruoli e le policy IAM richiesti e la trasformazione necessaria dei dati di input. Il back-end restituisce il risultato ad API Gateway come risposta di integrazione.

Per instradare la risposta di integrazione a una risposta di metodo appropriata (di un determinato codice di stato HTTP) al client, puoi configurare la risposta di integrazione per mappare i parametri di risposta richiesti dall'integrazione al metodo. Quindi puoi trasformare il formato dei dati di output del back-end nel formato del front-end, se necessario. API Gateway consente di definire uno schema o modello per il payload per agevolare la configurazione del modello di mappatura del corpo.

API Gateway fornisce le funzionalità di gestione delle API REST, ad esempio la seguente:

- Supporto per la generazione di SDK e la creazione di documentazione API mediante estensioni API Gateway per OpenAPI
- Throttling delle richieste HTTP

Utilizzo di API Gateway per creare API HTTP

Le API HTTP consentono di creare API RESTful con latenza minore e costi inferiori rispetto alle API REST.

È possibile utilizzare le API HTTP per inviare richieste a AWS Lambda funzioni o a qualsiasi endpoint HTTP instradabile pubblicamente.

Ad esempio, puoi creare un'API HTTP che si integra con una funzione Lambda sul back-end. Quando un client chiama l'API, API Gateway invia la richiesta alla funzione Lambda e restituisce la risposta della funzione al client.

API HTTP supportano l'autorizzazione OpenID Connect e OAuth 2.0. Sono dotate di supporto integrato per CORS (cross-origin resource sharing) e le distribuzioni automatiche.

Per ulteriori informazioni, consulta the section called “Scelta tra REST API e API HTTP ”.

Usa API Gateway per creare WebSocket API

In un' WebSocket API, il client e il server possono scambiarsi messaggi in qualsiasi momento. I server di back-end possono eseguire facilmente il push dei dati agli utenti e ai dispositivi connessi, evitando la necessità di implementare complessi meccanismi di polling.

Ad esempio, è possibile creare un'applicazione serverless utilizzando un'API WebSocket API Gateway e AWS Lambda inviare e ricevere messaggi da e verso singoli utenti o gruppi di utenti in una chat room. Oppure puoi richiamare servizi di backend come AWS Lambda Amazon Kinesis o un endpoint HTTP basato sul contenuto dei messaggi.

Puoi utilizzare le API WebSocket API Gateway per creare applicazioni di comunicazione sicure e in tempo reale senza dover fornire o gestire server per gestire connessioni o scambi di dati su larga scala. I casi d'uso di riferimento includono applicazioni in tempo reale come le seguenti:

- Applicazioni di chat
- Real-time dashboard come i ticker azionari
- Real-time avvisi e notifiche

API Gateway offre funzionalità di gestione delle WebSocket API come le seguenti:

- Monitoraggio e throttling di connessioni e messaggi
- Viene utilizzato AWS X-Ray per tracciare i messaggi mentre viaggiano attraverso le API verso i servizi di backend
- Facile integrazione con gli endpoint HTTP/HTTPS

Chi utilizza le API Gateway?

Esistono due tipi di sviluppatori che utilizzano API Gateway: gli sviluppatori di API e gli sviluppatori di app.

Uno sviluppatore di API crea e distribuisce un'API per abilitare la funzionalità richiesta in API Gateway. Lo sviluppatore dell'API deve essere un utente dell' AWS account proprietario dell'API.

Uno sviluppatore di app crea un'applicazione funzionante per chiamare AWS i servizi richiamando un' WebSocket API REST creata da uno sviluppatore di API in API Gateway.

Lo sviluppatore di app è il cliente dello sviluppatore di API. Lo sviluppatore dell'app non deve disporre di un AWS account, a condizione che l'API non richieda autorizzazioni IAM o supporti l'autorizzazione degli utenti tramite provider di identità federati di terze parti supportati dalla federazione delle identità dei pool di utenti di Amazon Cognito. Tra i provider di identità figurano Amazon, pool di utenti Amazon Cognito, Facebook e Google.

Creazione e gestione di un'API di API Gateway

Uno sviluppatore di API lavora con la componente del servizio API Gateway per la gestione delle API, denominata `apigateway`, per creare, configurare e distribuire un'API.

In qualità di sviluppatore di API, puoi creare e gestire un'API utilizzando la console API Gateway, descritta in Nozioni di base su Gateway API o chiamando il riferimenti API. Sono disponibili diversi modi per chiamare questa API. Includono l'utilizzo di AWS Command Line Interface (AWS CLI) o l'utilizzo di un SDK. AWS Inoltre, puoi abilitare la creazione di API con modelli AWS CloudFormation o (per le API REST e le API HTTP) Estensioni OpenAPI per Gateway API.

Per un elenco di regioni in cui API Gateway è disponibile, compresi gli endpoint del servizio di controllo associati, consulta Endpoint e quote Amazon API Gateway.

Chiamare un'API di API Gateway

Uno sviluppatore di app lavora con la componente del servizio API Gateway per l'esecuzione dell'API, denominata `execute-api`, per chiamare un'API creata o distribuita in API Gateway. Le entità di programmazione sottostanti vengono esposte dall'API creata. Sono disponibili diversi modi per chiamare questa API. Per ulteriori informazioni, consulta Invocazione di REST API in Gateway API e Invoca WebSocket APIs.

Accesso ad API Gateway

Puoi accedere ad Amazon API Gateway nei seguenti modi:

- Console di gestione AWS— Console di gestione AWS Fornisce un'interfaccia web per la creazione e la gestione delle API. Dopo aver completato le fasi descritte in Prerequisiti, è possibile accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
- AWS SDK: se utilizzi un linguaggio di programmazione che AWS fornisce un SDK per, puoi utilizzare un SDK per accedere all'API Gateway. Gli SDK semplificano l'autenticazione, si integrano facilmente nell'ambiente di sviluppo e ti offrono l'accesso ai comandi API Gateway. Per ulteriori informazioni, consulta Strumenti per Amazon Web Services.
- API V1 e V2 di API Gateway: se usi un linguaggio di programmazione per cui non è disponibile alcun SDK, consulta la documentazione di riferimento sulle API versione 1 di Amazon API Gateway e la documentazione di riferimento sulle API versione 2 di Amazon API Gateway.
- AWS Command Line Interface: per ulteriori informazioni, consulta Come configurare AWS Command Line Interface nella Guida per l'utente di AWS Command Line Interface .
- AWS Tools for Windows PowerShell: per ulteriori informazioni, consulta Come configurare AWS Tools for Windows PowerShell nella Guida per l'utente di AWS Strumenti per PowerShell .

Parte di AWS infrastruttura senza server

Oltre a AWS Lambda, API Gateway costituisce la parte rivolta alle app dell'infrastruttura AWS serverless. Per ulteriori informazioni su come iniziare a usare il serverless, consulta Serverless Developer Guide.

Affinché un'app richiami AWS servizi disponibili pubblicamente, puoi utilizzare Lambda per interagire con i servizi richiesti ed esporre le funzioni Lambda tramite metodi API in API Gateway. AWS Lambda esegue il codice su un'infrastruttura di elaborazione ad alta disponibilità. Esegue l'amministrazione e l'esecuzione necessarie delle risorse di calcolo. Per abilitare applicazioni serverless, API Gateway supporta integrazioni proxy semplificate con endpoint HTTP AWS Lambda .

Informazioni su come iniziare a utilizzare Amazon API Gateway

Per un'introduzione ad Amazon API Gateway, consulta gli argomenti seguenti:

- Inizia a usare, che fornisce una spiegazione passo per passo per la creazione di un'API HTTP.

- Terreno serverless, che fornisce video didattici.
- Happy Little API Shorts, una serie di brevi video didattici.

Concetti di Amazon API Gateway

La sezione seguente descrive i concetti introduttivi per l'utilizzo di Gateway API.

Gateway API

API Gateway è un AWS servizio che supporta quanto segue:

- Creazione, distribuzione e gestione di un'interfaccia di programmazione delle applicazioni (API) RESTful per esporre endpoint HTTP, AWS Lambda funzioni o altri servizi di backend. AWS
- Creazione, distribuzione e gestione di un'WebSocketAPI per esporre funzioni o altri servizi. AWS Lambda AWS
- Richiamo di metodi API esposti tramite il frontend HTTP e gli endpoint. WebSocket

API REST API Gateway

Una raccolta di risorse e metodi HTTP integrati con endpoint HTTP di backend, funzioni Lambda o altri servizi. AWS È possibile distribuire questa raccolta in una o più fasi. In genere, le risorse API sono organizzate in una struttura di risorse in base alla logica dell'applicazione. Ogni risorsa API può esporre uno o più metodi API che dispongono di verbi HTTP univoci supportati da API Gateway. Per ulteriori informazioni, consulta the section called “Scelta tra REST API e API HTTP ”.

API HTTP API Gateway

Una raccolta di route e metodi integrati con endpoint HTTP di back-end o funzioni Lambda. È possibile distribuire questa raccolta in una o più fasi. Ogni route può esporre uno o più metodi API che dispongono di verbi HTTP univoci supportati da API Gateway. Per ulteriori informazioni, consulta the section called “Scelta tra REST API e API HTTP ”.

API Gateway API WebSocket

Una raccolta di WebSocket route e chiavi di route integrate con endpoint HTTP di backend, funzioni Lambda o altri servizi. AWS È possibile distribuire questa raccolta in una o più fasi. I metodi API vengono richiamati tramite WebSocket connessioni frontend che è possibile associare a un nome di dominio personalizzato registrato.

Distribuzione API

Una snapshot point-in-time dell'API di API Gateway. Per essere utilizzabile dai client, la distribuzione deve essere associata a una o più fasi API.

Sviluppatore di API

L' AWS account proprietario di una distribuzione di API Gateway (ad esempio, un fornitore di servizi che supporta anche l'accesso programmatico).

Endpoint API

Un nome host per un'API in API Gateway che viene distribuita in una regione specifica. Il formato del nome host è `{api-id}.execute-api.{region}.amazonaws.com`. Sono supportati i tipi di endpoint API seguenti:

- Edge-optimized Endpoint API
- Endpoint API privato
- Endpoint API regionale

Chiave API

Una stringa alfanumerica utilizzata da API Gateway per identificare uno sviluppatore di app che utilizza il tuo REST o l'API. WebSocket API Gateway può generare chiavi API per tuo conto o puoi importarle da un file CSV. Puoi utilizzare le chiavi API insieme ai provider di autorizzazioni Lambda o ai piani di utilizzo per controllare l'accesso alle API.

Consulta la sezione relativa a endpoint API.

Proprietario di API

Vedi Sviluppatore di API.

Fase API

Un riferimento logico a uno stato del ciclo di vita dell'API (ad esempio, "svi", "prod", "beta", "v2"). Le fasi API sono identificate dall'ID API e dal nome della fase.

Sviluppatore di app

Un creatore di app che può avere o meno un AWS account e interagisce con l'API che tu, lo sviluppatore dell'API, hai implementato. Gli sviluppatori di app sono i tuoi clienti. Uno sviluppatore di app è in genere identificato da una chiave API.

URL di callback

Quando un nuovo client è connesso tramite una WebSocket connessione, puoi chiamare un'integrazione in API Gateway per memorizzare l'URL di callback del cliente. Puoi quindi utilizzare questo URL di callback per inviare messaggi al client dal sistema di back-end.

Portale per sviluppatori

Un'applicazione in cui i provider di API possono condividere le proprie API e la documentazione delle API con i consumatori di API. Le API sono raggruppate in prodotti, che sono una raccolta di endpoint API REST, documentazione API e documentazione supplementare del prodotto.

Vedi i portali API Gateway.

Edge-optimized Endpoint API

Il nome host predefinito di un'API Gateway che viene distribuita nella regione specificata utilizzando una CloudFront distribuzione per facilitare l'accesso dei client, in genere da più regioni. AWS Le richieste API vengono indirizzate al CloudFront Point of Presence (POP) più vicino, il che in genere migliora i tempi di connessione per client geograficamente diversi.

Consulta la sezione relativa a endpoint API.

Richiesta di integrazione

L'interfaccia interna di una route WebSocket API o di un metodo API REST in API Gateway, in cui si mappa il corpo di una richiesta di route o i parametri e il corpo di una richiesta di metodo nei formati richiesti dal backend.

Risposta di integrazione

L'interfaccia interna di una route WebSocket API o di un metodo API REST in API Gateway, in cui si mappano i codici di stato, le intestazioni e il payload ricevuti dal backend al formato di risposta che viene restituito a un'app client.

Modello di mappatura

Uno script in Velocity Template Language (VTL) che trasforma il corpo di una richiesta dal formato dati del front-end al formato dati del back-end o che trasforma il corpo di una risposta dal formato dati del back-end al formato dati del front-end. I modelli di mappatura possono essere specificati nella richiesta o nella risposta di integrazione. Possono fare riferimento ai dati disponibili al runtime come contesto e variabili di fase.

La mappatura può essere semplice come una trasformazione di identità che trasferisce le intestazioni o il corpo mediante l'integrazione senza alcuna modifica dal client al back-end per una richiesta. Lo stesso vale per una risposta, in cui il payload viene trasferito dal back-end al client.

Richiesta metodo

Interfaccia pubblica di un metodo API in API Gateway che definisce i parametri e il corpo che uno sviluppatore di app deve inviare nelle richieste per accedere al back-end tramite l'API.

Risposta di metodo

Interfaccia pubblica di un'API REST che definisce i codici di stato, le intestazioni e i modelli di corpo che uno sviluppatore di app dovrebbe aspettarsi in risposta dall'API.

Integrazione fittizia

In un'integrazione fittizia, le risposte API vengono generate direttamente da API Gateway, senza che sia richiesto un back-end di integrazione. Uno sviluppatore di API può decidere in che modo API Gateway risponde alle richieste di integrazione fittizia. A tale scopo, è necessario configurare la richiesta e la risposta di integrazione del metodo per associare una risposta a un codice di stato specifico.

Modello

Schema di dati che specifica la struttura di dati di una richiesta o di un payload di risposta. Un modello è necessario per generare un SDK fortemente tipizzato di un'API. Viene anche utilizzato per convalidare payload. Un modello è utile per generare un modello di mappatura di esempio per avviare la creazione di un modello di mappatura di produzione. Sebbene sia utile, il modello non è richiesto per la creazione di un modello di mappatura.

Portal

Vedi il portale per gli sviluppatori.

Prodotto Portal

Un servizio o una funzionalità che desideri condividere. Il prodotto del portale è una raccolta di endpoint REST e pagine di prodotto. Gli endpoint REST del prodotto sono i punti di accesso al prodotto del portale e sono costituiti dal percorso e dal metodo di un'API REST e dalla fase in cui viene distribuita. Le pagine dei prodotti sono la documentazione fornita per spiegare in che modo gli utenti delle API possono utilizzare gli endpoint del prodotto. Puoi condividere i prodotti tra più AWS account per aggiungerli ai portali.

Vedi Creare un prodotto.

Private API

Consulta la sezione relativa a endpoint API privato.

Endpoint API privato

Un endpoint API che viene esposto tramite endpoint VPC dell'interfaccia e che consente a un client di accedere in modo sicuro alle risorse API private all'interno di un VPC. Le API Private sono isolate dalla rete Internet pubblica e sono accessibili solo utilizzando endpoint VPC per API Gateway che hanno ricevuto l'accesso.

Integrazione privata

Un tipo di integrazione API Gateway per consentire a un client di accedere alle risorse all'interno del VPC di un cliente tramite un endpoint API REST privato senza esporre le risorse alla rete Internet pubblica.

Integrazione proxy

Configurazione di integrazione API Gateway semplificata. Puoi impostare un'integrazione proxy come un'integrazione proxy HTTP o un'integrazione proxy Lambda.

Per l'integrazione proxy HTTP, API Gateway trasferisce richiesta e risposta complete tra il front-end e il back-end HTTP. Per l'integrazione proxy Lambda, API Gateway invia la richiesta completa come input alla funzione Lambda di back-end. API Gateway trasforma quindi l'output della funzione Lambda in una risposta HTTP per il front-end.

Nelle API REST, l'integrazione proxy viene più comunemente usata con una risorsa proxy, rappresentata da una variabile di percorso greedy (ad esempio, `{proxy+}`) abbinata a un metodo catch-all `ANY`.

Creazione rapida

È possibile utilizzare la creazione rapida per semplificare la creazione di un'API HTTP. Creazione rapida consente di creare un'API con un'integrazione Lambda o HTTP, una route catch-all predefinita e una fase predefinita configurata per distribuire automaticamente le modifiche. Per ulteriori informazioni, consulta the section called “Crea un'API HTTP utilizzando la AWS CLI”.

Endpoint API regionale

Il nome host di un'API distribuita nella regione specificata e destinata a servire i client, come le istanze EC2, nella stessa regione. AWS Le richieste API sono indirizzate direttamente all' Region-specific API API Gateway senza passare attraverso alcuna distribuzione. CloudFront Per le

richieste all'interno della regione, un endpoint regionale aggira l'inutile andata e ritorno verso una distribuzione. CloudFront

Inoltre, puoi applicare l'instradamento basato su latenza su endpoint regionali per distribuire un'API in più regioni utilizzando la stessa configurazione dell'endpoint API regionale, impostare lo stesso nome di dominio personalizzato per ogni API distribuita e configurare i record DNS basati sulla latenza in Route 53 per instradare le richieste client alla regione con latenza minima.

Consulta la sezione relativa a endpoint API.

Route

Un WebSocket percorso in API Gateway viene utilizzato per indirizzare i messaggi in arrivo a un'integrazione specifica, ad esempio una AWS Lambda funzione, in base al contenuto del messaggio. Quando definisci la tua WebSocket API, specifichi una chiave di route e un backend di integrazione. La chiave route è un attributo nel corpo del messaggio. Quando viene trovata una corrispondenza per la chiave route in un messaggio in entrata, viene richiamato il back-end di integrazione.

È anche possibile impostare una route predefinita per chiavi route non corrispondenti o per specificare un modello di proxy che trasferisce il messaggio senza alcuna modifica ai componenti di back-end che eseguono l'instradamento ed elaborano la richiesta.

Richiesta di instradamento

L'interfaccia pubblica di un metodo WebSocket API in API Gateway che definisce il corpo a cui uno sviluppatore di app deve inviare le richieste per accedere al backend tramite l'API.

Risposta di instradamento

L'interfaccia pubblica di un' WebSocket API che definisce i codici di stato, le intestazioni e i modelli corporei che uno sviluppatore di app dovrebbe aspettarsi da API Gateway.

Piano di utilizzo

Un piano di utilizzo fornisce ai client API selezionati l'accesso a uno o più REST o WebSocket API distribuiti. Puoi usare un piano di utilizzo per configurare i limiti di throttling e di quote, applicati a singole chiavi API dei client.

WebSocket connessione

API Gateway mantiene una connessione permanente tra i client e l'API Gateway stesso. Non esiste alcuna connessione permanente tra API Gateway e le integrazioni di back-end, ad esempio

funzioni Lambda. I servizi di back-end vengono richiamati in base alle esigenze, a seconda del contenuto dei messaggi ricevuti dai client.

Scelta tra REST API e API HTTP

REST API e API HTTP sono entrambe prodotti dell'API RESTful. Le REST API supportano più funzionalità rispetto alle API HTTP, mentre le API HTTP sono progettate con funzionalità minime in modo che possano essere offerte a un prezzo inferiore. Scegliere le REST API se si ha bisogno di funzionalità come chiavi API, limitazione (della larghezza di banda della rete) per client, convalida delle richieste, integrazione AWS WAF o endpoint API privati. Scegliere le API HTTP se non si ha bisogno delle funzionalità incluse nelle REST API.

Nelle sezioni seguenti vengono riepilogate le funzionalità principali disponibili nelle REST API e nelle API HTTP. Se necessario, vengono forniti collegamenti aggiuntivi per navigare tra le sezioni REST API e API HTTP della Guida per gli sviluppatori di Gateway API.

Tipo di endpoint

Il tipo di endpoint si riferisce all'endpoint creato da API Gateway per la propria API. Per ulteriori informazioni, consulta the section called “Tipi di endpoint di Gateway API”.

Tipi di endpoint	REST API	API HTTP
Edge-optimized	Sì	No
Regionale	Sì	Sì
Privata	Sì	No

Sicurezza

API Gateway fornisce diversi modi per proteggere l'API da determinate minacce, ad esempio utenti malintenzionati o picchi di traffico. Per ulteriori informazioni, consultare the section called “Protezione” e the section called “Protezione”.

Funzionalità di sicurezza	REST API	API HTTP
Autenticazione TLS reciproca	Sì	Sì
Certificati per l'autenticazione di back-end	Sì	No
AWS WAF	Sì	No

Autorizzazione

API Gateway supporta più meccanismi per controllare e gestire l'accesso all'API. Per ulteriori informazioni, consulta the section called “Controllo accessi” e the section called “Controllo accessi”.

Opzioni di autorizzazione	REST API	API HTTP
IAM	Sì	Sì
Policy delle risorse	Sì	No
Amazon Cognito	Sì	Sì [1]
Autorizzazione personalizzata con una AWS Lambda funzione	Sì	Sì
Token Web JSON (JWT) [2]	No	Sì

[1] È possibile utilizzare Amazon Cognito con un'autorizzazione JWT.

[2] È possibile utilizzare un'autorizzazione Lambda per convalidare i JWT per le REST API.

Gestione API

Scegliere le REST API se si ha bisogno di funzionalità di gestione API come chiavi API e limitazione della frequenza per client. Per ulteriori informazioni, consultare the section called “Distribuzione”, the section called “Nomi di dominio personalizzati” e the section called “Nomi di dominio personalizzati”.

Funzionalità	REST API	API HTTP
Domini personalizzati	Sì	Sì
Chiavi API	Sì	No
Per-client limitazione della velocità	Sì	No
Per-client limitazione dell'utilizzo	Sì	No
Portale per sviluppatori	Sì	No

Sviluppo

Durante lo sviluppo delle API di API Gateway è possibile impostare una serie di caratteristiche dell'API. Queste caratteristiche dipendono dal caso d'uso dell'API. Per ulteriori informazioni, consulta the section called “Sviluppare” e the section called “Sviluppare”.

Funzionalità	REST API	API HTTP
Configurazione CORS	Sì	Sì
Chiamate di test	Sì	No
Caching	Sì	No
User-controlled implementazioni	Sì	Sì
Implementazioni automatiche	No	Sì
Risposte gateway personalizzate	Sì	No
Implementazione di una release Canary	Sì	No

Funzionalità	REST API	API HTTP
Convalida delle richieste	Sì	No
Trasformazione parametro della richiesta	Sì	Sì
Trasformazione corpo della richiesta	Sì	No

Monitoraggio

API Gateway supporta diverse opzioni per registrare le richieste API e monitorare le API. Per ulteriori informazioni, consulta the section called “Monitoraggio” e the section called “Monitoraggio”.

Funzionalità	REST API	API HTTP
metriche Amazon CloudWatch	Sì	Sì
Log di accesso ai CloudWatch log	Sì	Sì
Log di accesso ad Amazon Data Firehose	Sì	No
Registri di esecuzione	Sì	No
AWS X-Ray tracciamento	Sì	No

Integrazioni

Le integrazioni collegano l'API Gateway API alle risorse di back-end. Per ulteriori informazioni, consulta the section called “Integrazioni” e the section called “Integrazioni”.

Funzionalità	REST API	API HTTP
Endpoint HTTP pubblici	Sì	Sì

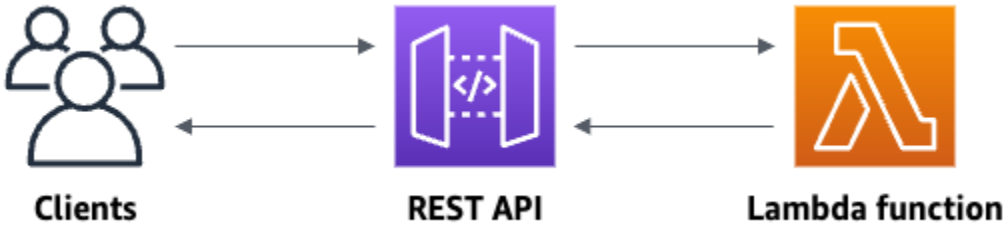
Funzionalità	REST API	API HTTP
AWS servizi	Sì	Sì
AWS Lambda funzioni	Sì	Sì
Integrazioni private con Network Load Balancer	Sì	Sì
Integrazioni private con il sistema di bilanciamento del carico dell'applicazione	Sì	Sì
Integrazioni private con AWS Cloud Map	No	Sì
Integrazioni fittizie	Sì	No
Streaming delle risposte	Sì	No

Nozioni di base sulla console REST API

In questa esercitazione introduttiva, creerai una REST API serverless utilizzando la console REST API Gateway API. La tecnologia serverless APIs consente di concentrarsi sulle applicazioni anziché dedicare tempo al provisioning e alla gestione dei server. Il completamento di questa esercitazione richiede meno di 20 minuti e rientra nel piano gratuito AWS.

Innanzitutto, crei una funzione Lambda utilizzando la console Lambda. Successivamente, crei una REST API utilizzando la console REST API Gateway API. Quindi, crei un metodo API e lo integri con una funzione Lambda utilizzando un'integrazione proxy Lambda. Infine, implementa e richiama la tua API.

Quando richiami la REST API, Gateway API instrada la richiesta alla funzione Lambda. Lambda esegue la funzione e restituisce una risposta a Gateway API. Gateway API restituisce quindi una risposta all'utente.



Per completare questo esercizio, sono necessari un utente Account AWS e un utente AWS Identity and Access Management (IAM) con accesso alla console. Per ulteriori informazioni, consulta Configurazione per l'utilizzo di Gateway API.

Argomenti

- Fase 1: creazione di una funzione Lambda
- Fase 2: Creazione di una REST API
- Fase 3: Creazione di un'integrazione proxy Lambda
- Fase 4: Implementazione dell'API
- Fase 5: Richiamo dell'API
- (Opzionale) Fase 6: pulizia

Fase 1: creazione di una funzione Lambda

Viene utilizzata una funzione Lambda per il backend della tua API. Lambda esegue il codice solo quando è necessario e si dimensiona automaticamente, da poche richieste al giorno a migliaia al secondo.

Per questa esercitazione, utilizzi una funzione Node.js predefinita nella console Lambda.

Come creare una funzione Lambda

1. Accedi alla console Lambda all'indirizzo https://console.aws.amazon.com /lambda.
2. Scegli Crea funzione.
3. In Basic information (Informazioni di base) , per Function name (Nome funzione) , inserisci **`my-function`**.
4. Utilizza le impostazioni predefinite per tutte le altre opzioni.
5. Scegli Crea funzione.

Il codice predefinito della funzione Lambda dovrebbe essere simile al seguente:

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

In questa esercitazione puoi modificare la funzione Lambda, purché la risposta della funzione sia conforme al formato richiesto da Gateway API.

Sostituisci il corpo della risposta predefinito (`Hello from Lambda!`) con `The API Gateway REST API console is great!`. Quando richiami la funzione di esempio, restituisce una risposta `200` ai client, insieme alla risposta aggiornata.

Fase 2: Creazione di una REST API

Successivamente, crei una REST API con una risorsa root (`/`).

Creazione di una REST API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Esegui una delle seguenti operazioni:
   - Per creare la tua prima API, in REST API, scegli Crea.
   - Se hai già creato un'API, scegli Crea API, quindi scegliere Crea per API REST.
3. Per API name (Nome API), immettere **`my-rest-api`**.
4. (Facoltativo) In Descrizione, immetti una descrizione.
5. Lasciare l'opzione Tipo di endpoint API impostata su Regionale.
6. Per il tipo di indirizzo IP, selezionare. IPv4
7. Seleziona Create API (Crea API).

Fase 3: Creazione di un'integrazione proxy Lambda

Successivamente, crei un metodo API per la REST API nella risorsa root (`/`) e integri tale il metodo con la funzione Lambda mediante un'integrazione proxy. In un'integrazione proxy Lambda, Gateway API passa la richiesta in entrata dal client direttamente alla funzione Lambda.

Creazione di un'integrazione proxy Lambda

1. Seleziona la risorsa `/`, quindi scegli Crea metodo.
2. In Tipo di metodo, seleziona `ANY`.
3. In Tipo di integrazione, seleziona Lambda.
4. Attiva l'opzione Integrazione proxy Lambda.
5. In Funzione Lambda, immetti **`my-function`**, quindi seleziona la funzione Lambda.
6. Scegli Crea metodo.

Fase 4: Implementazione dell'API

Successivamente, crei un'implementazione API e la associa a una fase.

Per distribuire l'API

1. Seleziona Deploy API (Distribuisci API).
2. In Fase, seleziona Nuova fase.
3. In Stage name (Nome fase) immettere **`Prod`**.
4. (Facoltativo) In Descrizione, immetti una descrizione.
5. Seleziona Implementa.

Ora i client possono chiamare la tua API. Per testare la tua API prima di implementarla, puoi facoltativamente scegliere il metodo Qualsiasi, accedere alla scheda Test e quindi scegliere Test.

Fase 5: Richiamo dell'API

Richiamo dell'API

1. Nel riquadro di navigazione principale, seleziona Log.
2. In Dettagli fase, scegli l'icona Copia per copiare l'URL di richiamo dell'API.



3. Immetti l'URL di richiamo in un browser Web.

   L'URL completo dovrebbe essere del tipo `https://abcd123.execute-api.us-east-2.amazonaws.com/Prod`.

   Il tuo browser invia una richiesta GET all'API.

4. Verifica la risposta dell'API. Il testo `"The API Gateway REST API console is great!"` dovrebbe essere visualizzato nel browser in uso.

(Opzionale) Fase 6: pulizia

Per evitare costi inutili Account AWS, elimina le risorse che hai creato come parte di questo esercizio. La procedura seguente elimina la REST API, la funzione Lambda e le risorse associate.

Eliminazione di una REST API

1. Nel riquadro Risorse, scegli Azioni API, Elimina API.
2. Nella finestra di dialogo Elimina immetti Conferma e quindi scegli Elimina.

Eliminazione della funzione Lambda

1. Accedi alla console Lambda all'indirizzo https://console.aws.amazon.com /lambda.
2. Nella pagina Funzioni, seleziona la tua funzione. Scegli Operazioni > Elimina.

3. Nella finestra di dialogo Elimina 1 funzione/i, immetti **`delete`** e quindi scegli Elimina.

Eliminazione del gruppo di log della funzione Lambda

1. Apri la pagina dei gruppi di log della CloudWatch console Amazon.
2. Nella pagina Gruppi di log, seleziona il gruppo di log della funzione (`/aws/lambda/my-function`). In Operazioni, scegli Elimina gruppo di log.
3. Nella finestra di dialogo Delete log group(s) (Elimina gruppo/i di log) scegli Delete (Elimina).

Eliminazione del ruolo di esecuzione della funzione Lambda

1. Aprire la pagina Ruoli della console IAM.
2. (Facoltativo) Nella pagina Ruoli, nella casella di ricerca, immetti **`my-function`**.
3. Seleziona il ruolo della tua funzione (ad esempio, `my-function-31exxmpl`), quindi scegli Elimina.
4. Nella finestra di dialogo Eliminare **`my-function-31exxmpl`**?, immetti il nome del ruolo, quindi scegli Elimina.

> **Tip**
>
> Puoi automatizzare la creazione e la pulizia delle AWS risorse utilizzando AWS CloudFormation o AWS Serverless Application Model ()AWS SAM. Per alcuni CloudFormation modelli di esempio, consulta i modelli di esempio per API Gateway nel repository awsdocs GitHub .

Configurazione per l'utilizzo di Gateway API

Prima di usare il Gateway Amazon API per la prima volta, è necessario completare le seguenti attività.

Iscriviti per un Account AWS

Per iniziare AWS, hai bisogno di un Account AWS. Esistono due modi per iscriversi AWS. Se desideri il controllo completo sulla configurazione dell'account e sulla posizione in cui sono archiviati i tuoi contenuti, o se hai carichi di lavoro regolamentati come HIPAA o FedRAMP, usa Sign up for (avanzato). AWS Se sei un nuovo utente e desideri utilizzare un nome di accesso che già possiedi per accedere AWS e creare un AWS ambiente con impostazioni preconfigurate preconfigurate in select, usa Sign up for. Regioni AWSAWS

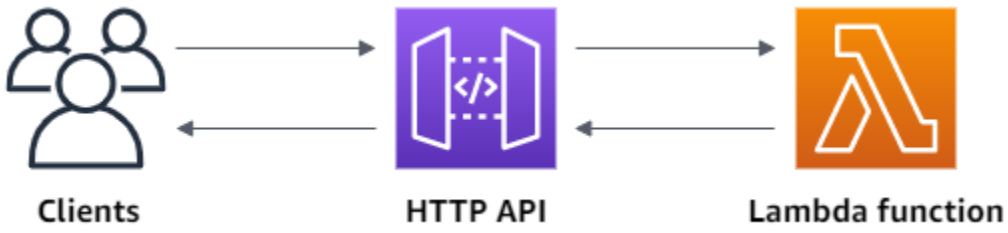
Nozioni di base su Gateway API

In questo esercizio di nozioni di base, viene creata un'API serverless. Le API serverless consentono di concentrarsi sulle applicazioni, invece di dedicare tempo al provisioning e alla gestione dei server. Il completamento di questo esercizio richiede meno di 20 minuti e può essere effettuato all'interno del piano gratuito AWS.

Innanzitutto, viene creata una funzione Lambda utilizzando la console AWS Lambda. Successivamente, è possibile creare un'API HTTP utilizzando la console API Gateway. Quindi, viene richiamata l'API.

> **Note**
>
> In questo esercizio si utilizza un'API HTTP. API Gateway supporta anche le API REST, che includono più funzionalità. Per un tutorial sull'utilizzo di una REST API, consulta the section called “Nozioni di base sulla console REST API”.
> Per ulteriori informazioni sulla differenza tra API HTTP e REST API, consulta the section called “Scelta tra REST API e API HTTP ”.

Quando si richiama l'API HTTP, API Gateway instrada la richiesta alla funzione Lambda. Lambda esegue la funzione Lambda e restituisce una risposta alla console API Gateway. La console API Gateway restituisce quindi una risposta all'utente.



Per completare questo esercizio, sono necessari un account AWS e un utente AWS Identity and Access Management con accesso alla console. Per ulteriori informazioni, consulta Prerequisiti.

Argomenti

- Fase 1: creazione di una funzione Lambda
- Fase 2: creazione un'API HTTP
- Fase 3: test dell'API

- (Opzionale) Fase 4: pulizia
- Fasi successive

Fase 1: creazione di una funzione Lambda

Viene utilizzata una funzione Lambda per il backend della tua API. Lambda esegue il codice solo quando è necessario e si dimensiona automaticamente, da poche richieste al giorno a migliaia al secondo.

Per questo esempio, viene utilizzata la funzione Node.js predefinita dalla console Lambda.

Per creare una funzione Lambda

1. Accedere alla console Lambda all'indirizzo https://console.aws.amazon.com/lambda/.
2. Selezionare Create function (Crea funzione).
3. Nel campo Function name (Nome funzione), immettere **`my-function`**.
4. Utilizza le impostazioni predefinite per tutte le altre opzioni.
5. Selezionare Create function (Crea funzione).

La funzione di esempio restituisce una risposta `200` ai client e il testo `Hello from Lambda!`.

È possibile modificare la funzione Lambda, purché la risposta della funzione si allinei al formato richiesto da API Gateway.

Il codice predefinito della funzione Lambda dovrebbe essere simile al seguente:

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

Fase 2: creazione un'API HTTP

Successivamente, viene creata un'API HTTP. La console API Gateway supporta anche API REST e API WebSocket, ma un'API HTTP è la scelta migliore per questo esercizio. Le API REST supportano

più funzionalità rispetto alle API HTTP, ma non sono necessarie per questo esercizio. Le API HTTP sono progettate con funzionalità minime in modo che possano essere offerte a un prezzo inferiore. Le API WebSocket mantengono connessioni persistenti con i client per la comunicazione full-duplex, che non è necessaria per questo esempio.

L'API HTTP fornisce un endpoint HTTP per la funzione Lambda. API Gateway instrada le richieste alla funzione Lambda e quindi restituisce la risposta della funzione ai client.

Per creare un'API HTTP

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere una delle seguenti operazioni:
   - Per creare la prima API, per l'API HTTP, scegliere Crea.
   - Se hai già creato un'API, scegliere Crea API, quindi scegliere Crea per API HTTP.
3. Per Integrazioni, scegliere Aggiungi integrazione.
4. Selezionare Lambda.
5. Per Lambda function (Funzione Lambda), immetti **`my-function`**.
6. Per API name (Nome API), immettere **`my-http-api`**.
7. Scegli Next (Successivo).
8. Esaminare la route creata da API Gateway e quindi scegliere Avanti.
9. Esaminare la fase creata da Gateway API e quindi scegliere Avanti.
10. Seleziona Crea.

Ora hai creato un'API HTTP con un'integrazione Lambda che è pronta a ricevere richieste dai client.

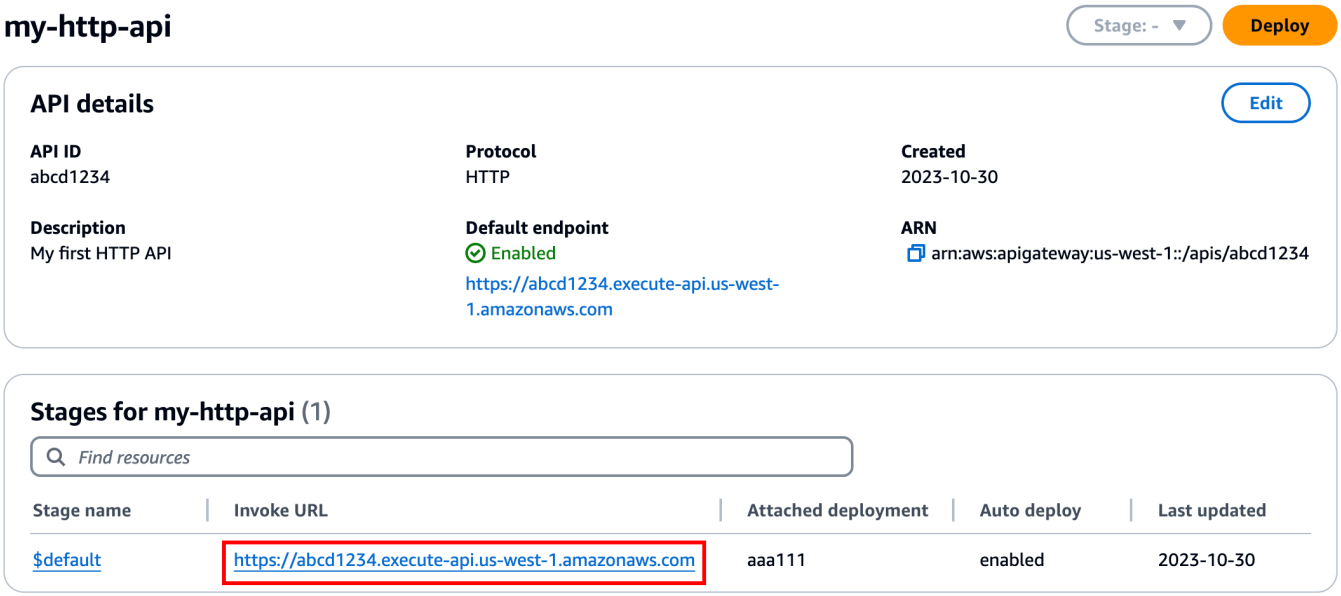
Fase 3: test dell'API

Successivamente, viene verificata l'API per assicurarsi che funzioni. Per semplicità, usa un browser Web per richiamare l'API.

Per testare l'API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere l'API.

3. Prendere nota del valore URL di chiamata per l'API.



4. Copiare il valore URL di chiamata per l'API e inserirlo in un browser Web. Aggiungere il nome della tua funzione Lambda al valore URL di chiamata per chiamare la tua funzione Lambda. Per impostazione predefinita, la console Gateway API crea un percorso con lo stesso nome della funzione Lambda, `my-function`.

   L'URL completo dovrebbe essere del tipo `https://`*`abcdef123`*`.execute-api.`*`us-east-2`*`.amazonaws.com/`*`my-function`*.

   Il tuo browser invia una richiesta `GET` all'API.

5. Verifica la risposta dell'API. Il testo `"Hello from Lambda!"` dovrebbe essere visualizzato nel browser in uso.

(Opzionale) Fase 4: pulizia

Per evitare costi non necessari, eliminare le risorse create nell'ambito di questo esercizio di nozioni di base. La procedura seguente elimina l'API HTTP, la funzione Lambda e le risorse associate.

Per eliminare un'API HTTP

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nella pagina API, selezionare un'API. Scegli Azioni, quindi Elimina.
3. Scegliere Delete (Elimina).

Per eliminare una funzione Lambda

1. Accedere alla console Lambda all'indirizzo https://console.aws.amazon.com/lambda/.
2. Nella pagina Funzioni, selezionare una funzione. Scegli Azioni, quindi Elimina.
3. Scegliere Delete (Elimina).

Per eliminare il gruppo di log di una funzione Lambda

1. Nella console Amazon CloudWatch, apri la pagina Log Groups (Gruppi di log).
2. Nella pagina Log Groups (Gruppi di log), seleziona il gruppo di log della funzione (`/aws/lambda/my-function`). Scegliere Actions (Operazioni), quindi selezionare Delete log group (Elimina gruppo di log).
3. Scegliere Delete (Elimina).

Per eliminare il ruolo di esecuzione di una funzione Lambda

1. Apri la pagina dei ruoli nella console AWS Identity and Access Management.
2. Seleziona il ruolo della funzione, ad esempi, `my-function-31exxmpl`.
3. Scegliere Delete role (Elimina ruolo).
4. Scegliere Yes, delete (Sì, elimina).

È possibile automatizzare la creazione e la pulizia delle risorse AWS utilizzando CloudFormation o AWS SAM. Per un esempio dei modelli CloudFormation, consulta i modelli di esempio CloudFormation.

Fasi successive

Per questo esempio, è stato utilizzato Console di gestione AWS per creare una semplice API HTTP. L'API HTTP richiama una funzione Lambda e restituisce una risposta ai client.

Di seguito, sono riportate le fasi successive quando si continua a lavorare con API Gateway.

- Configurare altri tipi di integrazioni API, tra cui:
  - Endpoint HTTP
  - Risorse private in un VPC, come i servizi Amazon ECS

  - AWS Servizi, come Amazon Simple Queue Service, AWS Step Functions e Kinesis Data Streams
- Controllare l'accesso alle API
- Abilitare la registrazione per le API
- Configurare la limitazione per le API
- Configurare domini personalizzati per le API

Per ricevere assistenza su Amazon API Gateway dalla community, consulta il forum di discussione di API Gateway. Per entrare nel forum, è possibile che AWS richieda l'accesso.

Per ottenere assistenza su API Gateway direttamente da AWS, consulta le opzioni di supporto nella pagina di supporto AWS.

Consulta anche le nostre domande frequenti (FAQ) oppure contattaci direttamente.

Tutorial e workshop di Amazon API Gateway

I seguenti tutorial e workshop forniscono esercizi pratici per aiutarti a usare API Gateway. Per iniziare, è consigliabile completare la sezione the section called “Tutorial: creazione di una REST API con un'integrazione proxy Lambda”.

Tutorial sull'API REST

- Scegli un tutorial di integrazione AWS Lambda
- Tutorial: creazione di un'API REST mediante l'importazione di un esempio
- Scelta di un tutorial per l'integrazione HTTP
- Tutorial: creazione di una REST API con un'integrazione privata
- Tutorial: creare un'API REST con un' AWS integrazione
- Tutorial: crea un'API REST per calcolatrice con due AWS integrazioni di servizi e un'integrazione non proxy Lambda
- Tutorial: creazione di una REST API come un proxy Amazon S3
- Tutorial: creazione di un'API REST come un proxy Amazon Kinesis
- Tutorial: creare un'API REST utilizzando AWS SDK o AWS CLI
- Tutorial: creazione di una REST API privata

Tutorial sull'API HTTP

- Tutorial: creazione di un'API HTTP CRUD con Lambda e DynamoDB
- Tutorial: creazione di un'API HTTP con un'integrazione privata con un servizio Amazon ECS

Tutorial sull'API WebSocket

- Tutorial: crea un'app di WebSocket chat con WebSocket API, Lambda e DynamoDB

Workshop

- Creazione di un'applicazione Web serverless
- CI/CD per applicazioni serverless
- Workshop sulla sicurezza serverless

- Gestione, autenticazione e autorizzazione delle identità serverless
- Workshop su Gateway Amazon API

Tutorial sull'API REST di Amazon API Gateway

I seguenti tutorial forniscono esercizi pratici per aiutarti a conoscere API Gateway REST. APIs

Argomenti

- Scegli un tutorial di integrazione AWS Lambda
- Tutorial: creazione di un'API REST mediante l'importazione di un esempio
- Scelta di un tutorial per l'integrazione HTTP
- Tutorial: creazione di una REST API con un'integrazione privata
- Tutorial: creare un'API REST con un' AWS integrazione
- Tutorial: crea un'API REST per calcolatrice con due AWS integrazioni di servizi e un'integrazione non proxy Lambda
- Tutorial: creazione di una REST API come un proxy Amazon S3
- Tutorial: creazione di un'API REST come un proxy Amazon Kinesis
- Tutorial: creare un'API REST utilizzando AWS SDK o AWS CLI
- Tutorial: creazione di una REST API privata

Scegli un tutorial di integrazione AWS Lambda

Per creare un'API tramite integrazioni Lambda, puoi usare l'integrazione proxy Lambda o l'integrazione non proxy Lambda.

Nell'integrazione proxy Lambda, l'input nella funzione Lambda può essere espresso con qualsiasi combinazione di intestazioni di richiesta, variabili di percorso, parametri di stringa di query, corpo e dati di configurazione dell'API. È necessario solo che tu scelga una funzione Lambda. API Gateway configura automaticamente la richiesta e la risposta di integrazione. Una volta configurate, il metodo API può evolvere senza modificare le impostazioni esistenti. Ciò è possibile in quanto la funzione Lambda backend analizza i dati della richiesta in entrata e risponde al client.

Con l'integrazione non proxy Lambda, devi accertarti che l'input della funzione Lambda venga fornito come payload della richiesta di integrazione. È necessario mappare i dati di input forniti dal client

come parametri di richiesta nel corpo della richiesta di integrazione appropriato. Potrebbe anche essere necessario convertire il corpo della richiesta fornito dal client in un formato riconosciuto dalla funzione Lambda.

In un proxy Lambda o in un'integrazione non proxy Lambda, è possibile utilizzare una funzione Lambda in un account diverso da quello in cui è stata creata l'API.

Argomenti

- Tutorial: creazione di una REST API con un'integrazione proxy Lambda
- Tutorial: creazione di una REST API con un'integrazione non proxy Lambda
- Tutorial: creazione di una REST API con un'integrazione proxy Lambda multi-account

Tutorial: creazione di una REST API con un'integrazione proxy Lambda

L'integrazione proxy Lambda è un tipo di integrazione API di API Gateway leggero e flessibile che permette di integrare un metodo API, o un'intera API, con una funzione Lambda. La funzione Lambda può essere scritta in qualsiasi linguaggio supportato da Lambda. Poiché si tratta di un'integrazione proxy, è possibile modificare l'implementazione della funzione Lambda in qualsiasi momento senza necessità di ridistribuire l'API.

In questo tutorial, esegui quanto indicato di seguito:

- Creazione di una funzione Lambda "Hello, World!" come back-end per l'API.
- Creazione e test di un'API "Hello World!" API con integrazione proxy Lambda.

Argomenti

- Creazione di una funzione Lambda "Hello, World!" valida e completa
- Creazione di una funzione Lambda "Hello, World!" API
- Distribuzione e test dell'API

Creazione di una funzione Lambda "Hello, World!" valida e completa

Per creare una funzione Lambda "Hello, World!" Funzione Lambda nella console Lambda

1. Accedi alla console Lambda all'indirizzo https://console.aws.amazon.com /lambda.
2. Nella barra di AWS navigazione, scegli un. Regione AWS

> **Note**
>
> Prendere nota della regione in cui si crea la funzione Lambda. Questa informazione sarà necessaria in fase di creazione dell'API.

3. Nel riquadro di navigazione scegliere Functions (Funzioni).
4. Selezionare Create function (Crea funzione).
5. Scegli Author from scratch (Crea da zero).
6. In Basic information (Informazioni di base) eseguire queste operazioni:
   a. In Function name (Nome funzione) immettere **`GetStartedLambdaProxyIntegration`**.
   b. Per Runtime, scegli l'ultimo runtime supportato di Node.js o di Python.
   c. Mantieni l'impostazione predefinita per Architettura.
   d. In Autorizzazioni espandere Modifica ruolo di esecuzione predefinito. Nell'elenco a discesa Ruolo di esecuzione, scegli Crea nuovo ruolo dai modelli di policy AWS .
   e. In Role name (Nome ruolo) immettere **`GetStartedLambdaBasicExecutionRole`**.
   f. Lasciare il campo Policy templates (Modelli di policy) vuoto.
   g. Selezionare Create function (Crea funzione).
7. In Codice funzione, nell'editor di codice in linea, copy/paste il codice seguente:

   Node.js

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

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

Python

```
import json


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

    greeter = 'World'

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

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

    try:
```

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

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

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

    return res
```

8. Selezionare Deploy (Distribuisci).

Creazione di una funzione Lambda "Hello, World!" API

Viene ora creata un'API per la funzione Lambda "Hello, World!" utilizzando la console API Gateway.

Per creare una funzione Lambda "Hello, World!" "Hello, World!"

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Se si utilizza API Gateway per la prima volta, verrà visualizzata una pagina che presenta le caratteristiche del servizio. In API REST, scegliere Crea. Quando appare il popup Create Example API (Crea API di esempio), scegliere OK.

   Se non è la prima volta che si utilizza API Gateway, scegliere Create API (Crea API). In API REST, scegliere Crea.
3. Per API name (Nome API), immettere **`LambdaProxyAPI`**.
4. (Facoltativo) In Descrizione, immetti una descrizione.

5. Lasciare l'opzione Tipo di endpoint API impostata su Regionale.
6. Per il tipo di indirizzo IP, selezionare. IPv4
7. Seleziona Create API (Crea API).

Dopo l'API, è necessario creare una risorsa. In genere, le risorse API sono organizzate in una struttura di risorse in base alla logica dell'applicazione. In questo esempio crei una risorsa /helloworld.

Come creare una risorsa

1. Scegli Crea risorsa.
2. Mantieni l'opzione Risorsa proxy disattivata.
3. Mantieni Percorso risorsa impostato su `/`.
4. Per Resource Name (Nome risorsa) immetti **`helloworld`**.
5. Mantieni CORS (Cross Origin Resource Sharing) disattivato.
6. Scegli Crea risorsa.

In un'integrazione proxy, l'intera richiesta viene inviata alla funzione Lambda di back-end così com'è, tramite un metodo `ANY` catch-all che rappresenta qualsiasi metodo HTTP. Il metodo HTTP effettivo viene specificato dal client in fase di runtime. Il metodo `ANY` consente di utilizzare la configurazione di un solo metodo API per tutti i metodi HTTP supportati: `DELETE`, `GET`, `HEAD`, `OPTIONS`, `PATCH`, `POST` e `PUT`.

Per creare un metodo **`ANY`**

1. Seleziona la risorsa /helloworld, quindi scegli Crea metodo.
2. Per Tipo di metodo seleziona ANY.
3. Per Tipo di integrazione seleziona Funzione Lambda.
4. Attiva l'opzione Integrazione proxy Lambda.
5. Per la funzione Lambda, seleziona il Regione AWS luogo in cui hai creato la funzione Lambda, quindi inserisci il nome della funzione.
6. Per utilizzare il valore di timeout predefinito di 29 secondi, mantieni attiva l'opzione Timeout predefinito. Per impostare un timeout personalizzato, scegli Timeout predefinito e immetti un valore di timeout compreso tra `50` e `29000` millisecondi.
7. Scegli Crea metodo.

Distribuzione e test dell'API

Per distribuire l'API

1. Seleziona Deploy API (Distribuisci API).
2. In Fase, seleziona Nuova fase.
3. In Stage name (Nome fase) immettere **`test`**.
4. (Facoltativo) In Descrizione, immetti una descrizione.
5. Seleziona Implementa.
6. In Dettagli fase, scegli l'icona Copia per copiare l'URL di richiamo dell'API.

Uso del browser e di cURL per testare un'API con integrazione proxy Lambda

Puoi utilizzare un browser oppure cURL per testare l'API.

Per testare le richieste `GET` utilizzando solo parametri della stringa di query, è possibile immettere l'URL per la risorsa `helloworld` dell'API nella barra degli indirizzi di un browser.

Per creare l'URL per la risorsa `helloworld` dell'API, aggiungi la risorsa `helloworld` e il parametro della stringa di query `?greeter=John` all'URL di invocazione. L'URL sarà simile al seguente.

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

Per gli altri metodi, occorre utilizzare utilità di test dell'API REST più avanzate, come POSTMAN o cURL. In questo tutorial viene usato cURL. Gli esempi di comando cURL seguenti presuppongono che cURL sia installato nel computer in uso.

Per testare l'API implementata tramite cURL:

1. Apri una finestra del terminale.
2. Copia il comando cURL seguente e incollalo nella finestra del terminale, quindi sostituisci l'URL di richiamata con quello copiato nella fase precedente e aggiungi **`/helloworld`** alla fine dell'URL.

> **Note**
>
> Se si esegue il comando in Windows, utilizzare questa sintassi:

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

a. Per chiamare l'API con il parametro della stringa di query `?greeter=John`:

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

b. Per chiamare l'API con un parametro di intestazione `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. Per chiamare l'API con un corpo `{"greeter":"John"}`:

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

In tutti i casi, l'output è una risposta 200 con il corpo seguente:

```
Hello, John!
```

Tutorial: creazione di una REST API con un'integrazione non proxy Lambda

In questa procedura guidata viene utilizzata la console API Gateway per creare un'API che permette a un client di chiamare le funzioni Lambda tramite l'integrazione non proxy Lambda (detta anche integrazione personalizzata). Per ulteriori informazioni su AWS Lambda e sulle funzioni Lambda, consulta la Guida per gli sviluppatori di AWS Lambda.

Per semplificare l'apprendimento, scegliamo una funzione Lambda semplice con una configurazione API minima per illustrare le fasi di creazione di un'API di API Gateway con l'integrazione

personalizzata Lambda. Laddove necessario, descriveremo la logica. Per un esempio più dettagliato dell'integrazione personalizzata Lambda, consulta Tutorial: crea un'API REST per calcolatrice con due AWS integrazioni di servizi e un'integrazione non proxy Lambda.

Prima di creare l'API, configura il back-end Lambda creando una funzione Lambda in AWS Lambda, come descritto di seguito.

Argomenti

- Creazione di una funzione Lambda per l'integrazione non proxy Lambda
- Creazione di un'API con l'integrazione non proxy Lambda
- Test della chiamata del metodo API
- Distribuzione dell'API
- Test dell'API in una fase di distribuzione
- Elimini

Creazione di una funzione Lambda per l'integrazione non proxy Lambda

> **Note**
>
> La creazione di funzioni Lambda può comportare addebiti sul tuo AWS account.

In questa fase, viene creata una funzione Lambda "Hello, World!" per l'integrazione Lambda personalizzata. In tutta la procedura guidata la funzione è denominata `GetStartedLambdaIntegration`.

L'implementazione di questa funzione Lambda `GetStartedLambdaIntegration` è la seguente:

Node.js

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

export const handler = async(event) => {
  console.log(event);
  // Parse the input for the name, city, time and day property values
```

```
  let name = event.name === null || event.name === undefined || event.name === "" ?
 'you' : event.name;
  let city = event.city === undefined ? 'World' : event.city;
  let time = times.indexOf(event.time)<0 ? 'day' : event.time;
  let day = days.indexOf(event.day)<0 ? null : event.day;

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

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

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

Python

```
import json

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


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

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

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

Per l'integrazione Lambda personalizzata, API Gateway passa l'input alla funzione Lambda dal client come corpo della richiesta di integrazione. L'oggetto `event` del gestore della funzione Lambda è l'input.

La nostra funzione Lambda è semplice. Analizza l'oggetto `event` di input per le proprietà `name`, `city`, `time` e `day`. Quindi restituisce un saluto, come un oggetto JSON di `{"message":greeting}`, all'intermediario. Il messaggio è nel modello `"Good [morning|afternoon|day], [name|you] in [city|World]. Happy day!"`. Si presume che l'input per la funzione Lambda sia dell'oggetto JSON seguente:

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

Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS Lambda.

Inoltre, la funzione registra la sua esecuzione su Amazon CloudWatch `console.log(...)` chiamando. Questo risulta utile per tracciare le chiamate durante il debug della funzione. Per consentire alla `GetStartedLambdaIntegration` funzione di registrare la chiamata, imposta un ruolo IAM con politiche appropriate per la funzione Lambda per creare gli CloudWatch stream e

aggiungere voci di registro agli stream. La console Lambda permette di creare le policy e i ruoli IAM necessari.

Se configuri l'API senza usare la console API Gateway, come nel caso in cui si esegue l'importazione di un'API da un file OpenAPI, dovrai creare esplicitamente, se necessario, e configurare una policy e un ruolo di invocazione affinché API Gateway possa invocare le funzioni Lambda. Per ulteriori informazioni su come configurare i ruoli di esecuzione e di invocazione Lambda per un'API di API Gateway, consulta Controllo degli accessi a una REST API con le autorizzazioni IAM.

A differenza di `GetStartedLambdaProxyIntegration`, la funzione Lambda per l'integrazione proxy Lambda, la funzione Lambda `GetStartedLambdaIntegration` per l'integrazione personalizzata Lambda riceve l'input solo dal corpo della richiesta di integrazione API di API Gateway. La funzione può restituire un output di qualsiasi oggetto JSON, una stringa, un numero, un valore booleano o un BLOB binario. La funzione Lambda per l'integrazione proxy Lambda, al contrario, può ricevere l'input da tutti i dati di richiesta ma deve restituire un output di un oggetto JSON specifico. La funzione `GetStartedLambdaIntegration` per l'integrazione personalizzata Lambda può ricevere come input i parametri di richiesta API, a condizione che API Gateway mappi i parametri di richiesta API necessari al corpo della richiesta di integrazione prima di inoltrare la richiesta client al back-end. Perché ciò accada, lo sviluppatore dell'API deve creare un modello di mappatura e configurarlo nel metodo API durante la creazione dell'API.

Viene ora creata la funzione Lambda `GetStartedLambdaIntegration`.

Per creare la funzione Lambda **`GetStartedLambdaIntegration`** per l'integrazione personalizzata Lambda

1. Apri la console all' AWS Lambda indirizzo. https://console.aws.amazon.com/lambda/
2. Esegui una delle seguenti operazioni:
   - Se viene visualizzata la pagina di benvenuto, scegliere Get Started Now (Inizia subito), quindi Create a function (Crea una funzione).
   - Se viene visualizzata la pagina di elenco Lambda > Funzioni, scegliere Create a function (Crea una funzione).
3. Scegli Author from scratch (Crea da zero).
4. Nel riquadro Author from scratch (Crea da zero) procedere nel seguente modo:
   a. In Nome (Name), immettere **`GetStartedLambdaIntegration`** come nome della funzione Lambda.

b. Per Runtime, scegli l'ultimo runtime supportato di Node.js o di Python.

c. Mantieni l'impostazione predefinita per Architettura.

d. In Autorizzazioni espandere Modifica ruolo di esecuzione predefinito. Nell'elenco a discesa Ruolo di esecuzione, scegli Crea nuovo ruolo dai modelli di policy AWS .

e. Per Role name (Nome ruolo), digitare un nome per il ruolo, ad esempio **`GetStartedLambdaIntegrationRole`**.

f. Per Modelli di policy, scegliere Autorizzazioni microservizi semplici.

g. Selezionare Create function (Crea funzione).

5. Nel riquadro Configure function (Configura funzione) effettua quanto segue in Function code (Codice funzione):

   a. Copiare il codice di funzione Lambda elencato all'inizio di questa sezione e incollarlo nell'editor di codice inline.

   b. Lascia i valori predefiniti negli altri campi di questa sezione.

   c. Selezionare Deploy (Distribuisci).

6. Per testare la funzione appena creata, scegli la scheda Test.

   a. Per Event name (Nome evento) immettere **`HelloWorldTest`**.

   b. Per JSON dell'evento, sostituisci il codice predefinito con il seguente.

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

   c. Scegliere Test (Testa) per invocare la funzione. Viene visualizzata la sezione Execution result: succeeded (Risultato esecuzione: riuscito). Espandi Dettaglio. Verrà visualizzato l'output seguente.

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

   L'output viene anche scritto in CloudWatch Logs.

Come esercizio complementare, puoi usare la console IAM per visualizzare il ruolo IAM (`GetStartedLambdaIntegrationRole`) creato insieme alla funzione Lambda. A questo ruolo IAM sono collegate due policy inline. Una stipula le autorizzazioni di base per l'esecuzione Lambda Consente di richiamare qualsiasi CloudWatch risorsa del tuo account nella regione in cui viene creata la funzione Lambda. CloudWatch `CreateLogGroup` Questa politica consente inoltre di creare CloudWatch flussi e registrare gli eventi per la funzione Lambda`GetStartedLambdaIntegration`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "logs:CreateLogGroup",
            "Resource": "arn:aws:logs:us-east-1:111111111111:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:111111111111:log-group:/aws/lambda/
GetStartedLambdaIntegration:*"
            ]
        }
    ]
}
```

L'altro documento di policy si applica all'invocazione di un altro AWS servizio che non viene utilizzato in questo esempio. Puoi ignorarlo in questa fase.

Al ruolo IAM è associata un'entità attendibile, ovvero `lambda.amazonaws.com`. La relazione di fiducia è la seguente:

JSON

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

La combinazione di questa relazione di fiducia e della politica in linea consente alla funzione Lambda di richiamare `console.log()` una funzione per registrare gli eventi nei registri. CloudWatch

Creazione di un'API con l'integrazione non proxy Lambda

Con la funzione Lambda (`GetStartedLambdaIntegration`) creata e testata, è possibile esporre la funzione mediante un'API di API Gateway. A scopo illustrativo, viene esposta la funzione Lambda con un metodo HTTP generico. Vengono utilizzati il corpo della richiesta, una variabile di percorso URL, una stringa di query e un'intestazione per ricevere i dati di input necessari dal client. Attiviamo il validatore di richieste API Gateway per l'API per garantire che tutti i dati necessari vengano definiti e specificati in modo appropriato. Configuriamo un modello di mappatura per API Gateway per trasformare i dati della richiesta forniti dal client nel formato valido, come richiesto dalla funzione Lambda di back-end.

Per creare un'API con un'integrazione non proxy Lambda

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Se si utilizza API Gateway per la prima volta, verrà visualizzata una pagina che presenta le caratteristiche del servizio. In API REST, scegliere Crea. Quando appare il popup Create Example API (Crea API di esempio), scegliere OK.

   Se non è la prima volta che si utilizza API Gateway, scegliere Create API (Crea API). In API REST, scegliere Crea.
3. Per API name (Nome API), immettere **`LambdaNonProxyAPI`**.

4. (Facoltativo) In Descrizione, immetti una descrizione.
5. Lasciare l'opzione Tipo di endpoint API impostata su Regionale.
6. Per il tipo di indirizzo IP, selezionare. IPv4
7. Seleziona Create API (Crea API).

Dopo l'API, è necessario creare una risorsa /{city}. Questo è l'esempio di una risorsa con una variabile di percorso che riceve un input dal client. Più avanti mapperai questa variabile nell'input della funzione Lambda utilizzando un modello di mappatura.

Come creare una risorsa

1. Scegli Crea risorsa.
2. Mantieni l'opzione Risorsa proxy disattivata.
3. Mantieni Percorso risorsa impostato su `/`.
4. Per Resource Name (Nome risorsa) immetti **`{city}`**.
5. Mantieni CORS (Cross Origin Resource Sharing) disattivato.
6. Scegli Crea risorsa.

Dopo la risorsa /{city}, è necessario creare un metodo `ANY`. Il metodo HTTP `ANY` funge da segnaposto per un metodo HTTP valido inviato da un client al runtime. Questo esempio mostra che il metodo `ANY` può essere utilizzato sia per l'integrazione Lambda personalizzata sia per quella proxy.

Per creare un metodo **`ANY`**

1. Seleziona la risorsa /{city}, quindi scegli Crea metodo.
2. Per Tipo di metodo seleziona ANY.
3. Per Tipo di integrazione seleziona Funzione Lambda.
4. Mantieni l'opzione Integrazione proxy Lambda disattivata.
5. Per la funzione Lambda, seleziona il Regione AWS luogo in cui hai creato la funzione Lambda, quindi inserisci il nome della funzione.
6. Scegli Impostazioni della richiesta del metodo.

   Ora, attivare un validatore richiesta per una variabile di percorso URL, un parametro di stringa di query e un'intestazione per assicurarsi che tutti i dati necessari siano definiti. Per questo esempio, crea un parametro della stringa di query `time` e un'intestazione `day`.

7. Per Validatore richiesta seleziona Convalida parametri di stringa query e intestazioni.
8. Scegli Parametri della stringa di query URL ed effettua le seguenti operazioni:
    a. Scegliere Add query string (Aggiungi stringa di query).
    b. In Nome, inserisci **`time`**.
    c. Attiva Campo obbligatorio.
    d. Mantieni disattivata l'opzione Caching.
9. Scegli Intestazioni di richiesta HTTP e procedi come segue:
    a. Seleziona Add header (Aggiungi intestazione).
    b. In Nome, inserisci **`day`**.
    c. Attiva Campo obbligatorio.
    d. Mantieni disattivata l'opzione Caching.
10. Scegli Crea metodo.

Dopo aver attivato un validatore della richiesta, configura la richiesta di integrazione per il metodo ANY aggiungendo un modello di mappatura del corpo per trasformare la richiesta in ingresso in un payload JSON, come richiesto dalla funzione Lambda di back-end.

Per configurare la richiesta di integrazione

1. Nella scheda Richiesta di integrazione scegli Modifica in Impostazioni della richiesta di integrazione.
2. Per Richiesta corpo passthrough scegli Quando non ci sono modelli definiti (consigliato).
3. Scegli Modelli di mappatura.
4. Scegliere Add mapping template (Aggiungi modello di mappatura).
5. Per Tipo di contenuto inserisci **`application/json`**.
6. Per Corpo del modello inserisci il seguente codice:

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

```
}
```

7. Scegli Save (Salva).

Test della chiamata del metodo API

La console API Gateway offre l'opportunità di testare la chiamata dell'API prima della distribuzione. La caratteristica Test della console consente di testare l'API inviando la richiesta seguente:

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

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

In questa richiesta di test imposterai `ANY` su `POST` e `{city}` su `Seattle` e assegnerai `Wednesday` come valore di intestazione di `day` e `"John"` come valore di `callerName`.

Test del metodo **ANY**

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. In Tipo di metodo, seleziona `POST`.
3. In Percorso immetti **`Seattle`** per city.
4. In Stringhe di query immetti **`time=morning`**.
5. In Intestazioni immetti **`day:Wednesday`**.
6. In Corpo della richiesta immetti **`{ "callerName": "John" }`**.
7. Scegli Test (Esegui test).

Verifica che il payload della risposta restituita sia come segue:

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

Puoi anche visualizzare i log per analizzare il modo in cui API Gateway elabora la richiesta e la risposta.

```
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
```

I log mostrano la richiesta in ingresso prima della mappatura e la richiesta di integrazione dopo la mappatura. I log risultano utili per valutare, in caso di esito negativo di un test, se l'input originale è corretto o il modello di mappatura funziona correttamente.

Distribuzione dell'API

La chiamata di test è una simulazione e presenta alcune limitazioni. Ad esempio ignora qualsiasi sistema di autorizzazione attuato per l'API. Per testare l'esecuzione dell'API in tempo reale, è necessario prima distribuire l'API. Per distribuire un'API, crea una fase per creare una snapshot dell'API in quel momento. Il nome della fase definisce inoltre il percorso di base in base al nome host predefinito dell'API. La risorsa radice dell'API viene aggiunta dopo il nome della fase. Quando modifichi l'API, devi ridistribuirla in una fase nuova o esistente per rendere effettive le modifiche.

Per distribuire l'API in una fase

1. Seleziona Deploy API (Distribuisci API).
2. In Fase, seleziona Nuova fase.
3. In Stage name (Nome fase) immettere **`test`**.

   > **Note**
   >
   > L'input deve essere un testo con codifica UTF-8, ovvero non localizzato.

4. (Facoltativo) In Descrizione, immetti una descrizione.
5. Seleziona Implementa.

In Dettagli fase, scegli l'icona Copia per copiare l'URL di richiamo dell'API. Il modello generale dell'URL di base dell'API è `https://`*`api-id`*`.`*`region`*`.amazonaws.com/`*`stageName`*. Ad esempio, l'URL di base dell'API (`beags1mnid`) creato nella regione `us-west-2` e distribuito nella fase `test` è `https://beags1mnid.execute-api.us-west-2.amazonaws.com/test`.

Test dell'API in una fase di distribuzione

Sono disponibili diversi modi per testare un'API distribuita. Per le richieste GET che utilizzano solo variabili di percorso URL o parametri di stringa di query, puoi immettere l'URL di risorsa dell'API nel browser. Per gli altri metodi, occorre utilizzare utilità di test dell'API REST più avanzate, come POSTMAN o cURL.

Per testare l'API tramite cURL

1. Apri una finestra terminale sul tuo computer locale connesso a Internet.
2. Per testare `POST /Seattle?time=evening`:

   Copia il comando cURL seguente e incollalo nella finestra terminale.

   ```
   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"
   }'
   ```

   Dovresti visualizzare una risposta di operazione riuscita con il payload seguente:

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

   Se modifichi `POST` in `PUT` in questa richiesta di metodo, otterrai la stessa risposta.

Elimini

Se non hai più bisogno delle funzioni Lambda create per questa procedura dettagliata, puoi eliminarle in questa fase. Puoi anche eliminare le risorse IAM associate.

> **Warning**
>
> Se intendi completare le altre procedure guidate di questa serie, non eliminare il ruolo di esecuzione o di invocazione Lambda. Se elimini una funzione Lambda su APIs cui fai

affidamento, quella non APIs funzionerà più. L'eliminazione di una funzione Lambda non può essere annullata. Se desideri utilizzare di nuovo la funzione Lambda, dovrai ricrearla.
Se elimini una risorsa IAM su cui si basa una funzione Lambda, quella funzione Lambda non funzionerà più e tutte le risorse APIs che si basano su quella funzione non funzioneranno più. L'eliminazione di una risorsa IAM non può essere annullata. Se desideri utilizzare di nuovo la risorsa IAM, dovrai ricrearla.

Per eliminare la funzione Lambda

1. Accedi a Console di gestione AWS e apri la console all' AWS Lambda indirizzo. https://console.aws.amazon.com/lambda/
2. Nell'elenco delle funzioni scegliere GetStartedLambdaIntegration, Actions (Operazioni), quindi Delete function (Elimina funzione). Quando viene richiesto, scegliere nuovamente Delete (Elimina).

Per eliminare le risorse IAM associate

1. Aprire la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. In Details (Dettagli) scegliere Roles (Ruoli).
3. Nell'elenco dei ruoli scegliere GetStartedLambdaIntegrationRole, Role Actions (Operazioni ruolo), quindi Delete role (Elimina ruolo). Per eliminare il ruolo utilizzando la console, attieniti alla procedura seguente.

Tutorial: creazione di una REST API con un'integrazione proxy Lambda multi-account

Ora puoi utilizzare una AWS Lambda funzione di un altro AWS account come backend di integrazione delle API. Ogni account può trovarsi in qualsiasi regione in cui è disponibile Amazon API Gateway. Ciò semplifica la gestione e la condivisione centralizzata delle funzioni di backend Lambda tra più utenti. APIs

In questa sezione, viene illustrato come configurare l'integrazione proxy Lambda tra più account utilizzando la console Amazon API Gateway.

Creazione di un'API per l'integrazione Lambda tra più account API Gateway

Come creare un'API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Se si utilizza API Gateway per la prima volta, verrà visualizzata una pagina che presenta le caratteristiche del servizio. In API REST, scegliere Crea. Quando appare il popup Create Example API (Crea API di esempio), scegliere OK.

   Se non è la prima volta che si utilizza API Gateway, scegliere Create API (Crea API). In API REST, scegliere Crea.
3. Per API name (Nome API), immettere **`CrossAccountLambdaAPI`**.
4. (Facoltativo) In Descrizione, immetti una descrizione.
5. Lasciare l'opzione Tipo di endpoint API impostata su Regionale.
6. Per il tipo di indirizzo IP, selezionare. IPv4
7. Seleziona Create API (Crea API).

Creazione di una funzione di integrazione Lambda in un altro account

Viene ora creata una funzione Lambda in un account diverso da quello in cui hai creato l'API di esempio.

Creazione di una funzione Lambda in un altro account

1. Accedere alla console Lambda in un account diverso da quello in cui è stata creata l'API di API Gateway.
2. Selezionare Create function (Crea funzione).
3. Scegli Author from scratch (Crea da zero).
4. In Author from scratch (Crea da zero), effettuare le seguenti operazioni:

   a. Per Function name (Nome funzione) immettere un nome.
   b. Dall'elenco a discesa Runtime, scegliere un runtime Node.js supportato.
   c. Mantieni l'impostazione predefinita per Architettura.
   d. In Autorizzazioni, espandere Scegli o crea un ruolo di esecuzione. È possibile creare un ruolo o scegliere un ruolo esistente.
   e. Selezionare Create function (Crea funzione) per continuare.

5. Scorrere verso il basso fino al riquadro Function code (Codice funzione).
6. Immetti l'implementazione della funzione Node.js da the section called “Tutorial: creazione di una REST API con un'integrazione proxy Lambda”.
7. Selezionare Deploy (Distribuisci).
8. Prendere nota dell'ARN completo della funzione (nell'angolo in alto a destra del riquadro della funzione Lambda), in quanto sarà necessario per la creazione dell'integrazione Lambda tra più account.

Configurazione dell'integrazione Lambda tra più account

Dopo la creazione di una funzione di integrazione Lambda in un altro account, puoi utilizzare la console API Gateway per aggiungerla all'API nel tuo primo account.

> **Note**
>
> Se si sta configurando una regione, un provider di autorizzazioni tra più account, `sourceArn` che viene aggiunto alla funzione di destinazione deve utilizzare la regione della funzione, non la regione dell'API.

Dopo l'API, è necessario creare una risorsa. In genere, le risorse API sono organizzate in una struttura di risorse in base alla logica dell'applicazione. In questo esempio crei una risorsa /helloworld.

Come creare una risorsa

1. Scegli Crea risorsa.
2. Mantieni l'opzione Risorsa proxy disattivata.
3. Mantieni Percorso risorsa impostato su `/`.
4. Per Resource Name (Nome risorsa) immetti **`helloworld`**.
5. Mantieni CORS (Cross Origin Resource Sharing) disattivato.
6. Scegli Crea risorsa.

Dopo la risorsa, è necessario creare un metodo `GET`. Integra il metodo `GET` con una funzione Lambda in un altro account.

Creazione di un metodo **GET**

1. Seleziona la risorsa /helloworld, quindi scegli Crea metodo.
2. Per Tipo di metodo seleziona GET.
3. Per Tipo di integrazione seleziona Funzione Lambda.
4. Attiva l'opzione Integrazione proxy Lambda.
5. Per Funzione Lambda immetti l'ARN completo della funzione Lambda dalla fase 1.

   Nella console Lambda, puoi trovare l'ARN della funzione nell'angolo in alto a destra.
6. Quando immetti l'ARN, viene visualizzata una stringa di comando `aws lambda add-permission`. Questa policy fornisce al tuo primo account l'accesso alla funzione Lambda del secondo account. Copia e incolla la stringa di `aws lambda add-permission` comando in una AWS CLI finestra configurata per il secondo account.
7. Scegli Crea metodo.

Puoi visualizzare la policy aggiornata per la funzione nella console Lambda.

(Facoltativo) Per visualizzare la policy aggiornata

1. Accedi a Console di gestione AWS e apri la AWS Lambda console all'indirizzo https://console.aws.amazon.com/lambda/.
2. Seleziona la funzione Lambda.
3. Seleziona Autorizzazioni.

   Viene visualizzata una policy `Allow` con una clausola `Condition` in cui `AWS:SourceArn` è l'ARN per il metodo `GET` dell'API.

Tutorial: creazione di un'API REST mediante l'importazione di un esempio

Puoi utilizzare la console Amazon API Gateway per creare e testare una semplice API REST con l'integrazione HTTP per un PetStore sito Web. La definizione dell'API è preconfigurata come file OpenAPI 2.0. Dopo aver caricato la definizione dell'API in API Gateway, puoi usare la console API Gateway per esaminare la struttura di base dell'API o semplicemente distribuire e testare l'API.

L'API di PetStore esempio supporta i seguenti metodi per consentire a un client di accedere al sito Web di backend HTTP dihttp://petstore-demo-endpoint.execute-api.com/petstore/pets.

> **Note**
>
> Questo tutorial utilizza un endpoint HTTP come esempio. Quando crei le tue API, consigliamo di utilizzare gli endpoint HTTPS per le integrazioni HTTP.

- `GET /`: per l'accesso in lettura della risorsa radice dell'API non integrata con un endpoint di back-end. API Gateway risponde con una panoramica del PetStore sito Web. Questo è un esempio del tipo di integrazione `MOCK`.
- `GET /pets`: per l'accesso in lettura alla risorsa `/pets` dell'API integrata con la risorsa `/pets` di back-end omonima. Il backend restituisce una pagina di animali domestici disponibili in. PetStore Questo è un esempio del tipo di integrazione `HTTP`. L'URL dell'endpoint di integrazione è `http://petstore-demo-endpoint.execute-api.com/petstore/pets`.
- `POST /pets`: per l'accesso in scrittura alla risorsa `/pets` dell'API integrata con la risorsa `/petstore/pets` di back-end. Dopo aver ricevuto una richiesta corretta, il backend aggiunge l'animale domestico specificato a PetStore e restituisce il risultato al chiamante. L'integrazione è anche `HTTP`.
- `GET /pets/{petId}`: per l'accesso in lettura a un animale domestico identificato da un valore `petId` specificato come variabile di percorso dell'URL di richiesta in entrata. Questo metodo ha anche il tipo di integrazione `HTTP`. Il backend restituisce l'animale domestico specificato trovato in. PetStore L'URL dell'endpoint HTTP di back-end è `http://petstore-demo-endpoint.execute-api.com/petstore/pets/`*n*, dove n è un valore intero come l'identificativo dell'animale domestico su cui è stata eseguita la query.

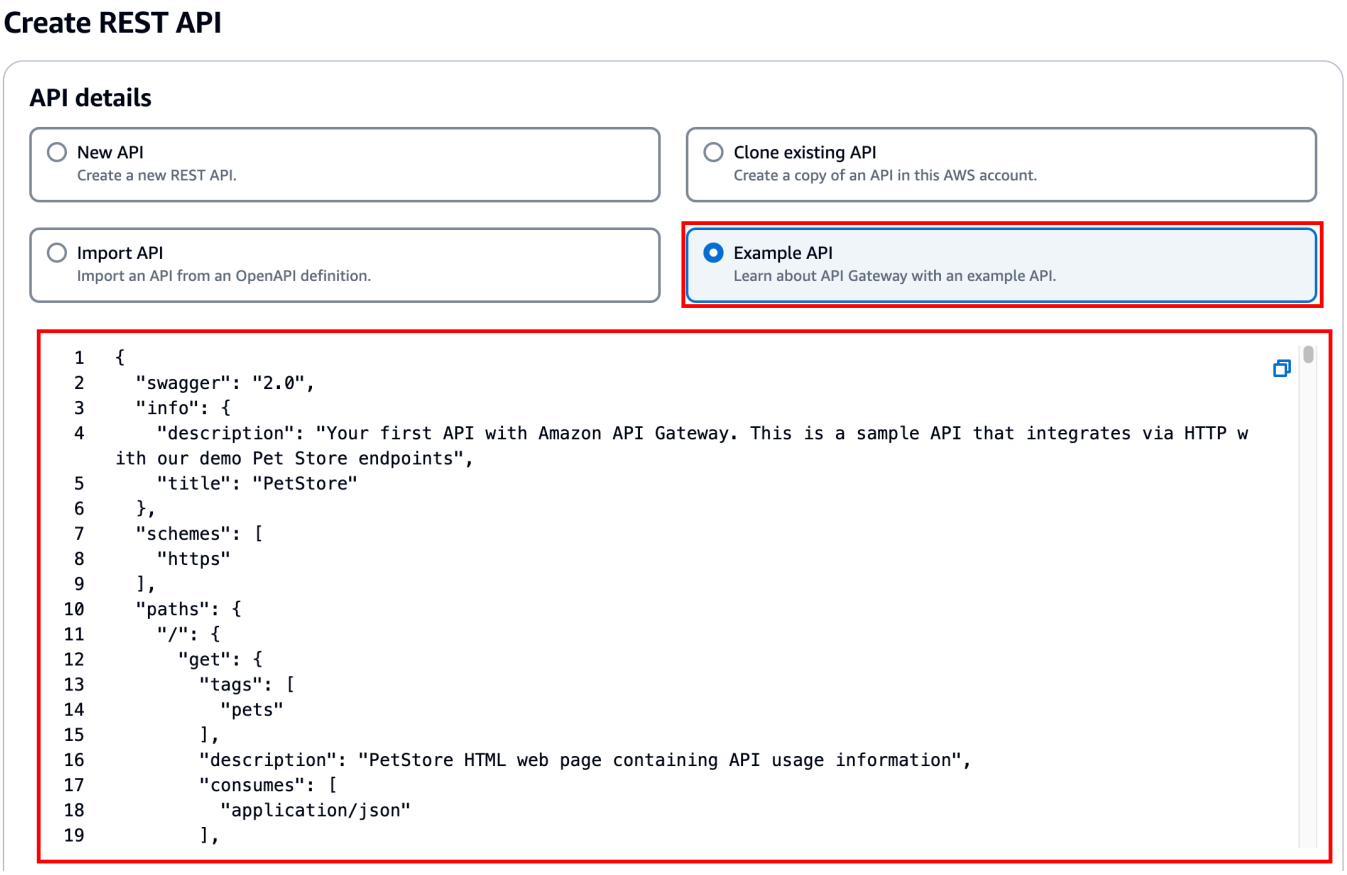
L'API supporta l'accesso CORS mediante i metodi `OPTIONS` del tipo di integrazione `MOCK`. API Gateway restituisce le intestazioni richieste che supportano l'accesso a CORS.

La procedura seguente descrive le diverse fasi per creare e testare un'API da un esempio mediante la console API Gateway.

Per importare, creare ed eseguire il test dell'API di esempio

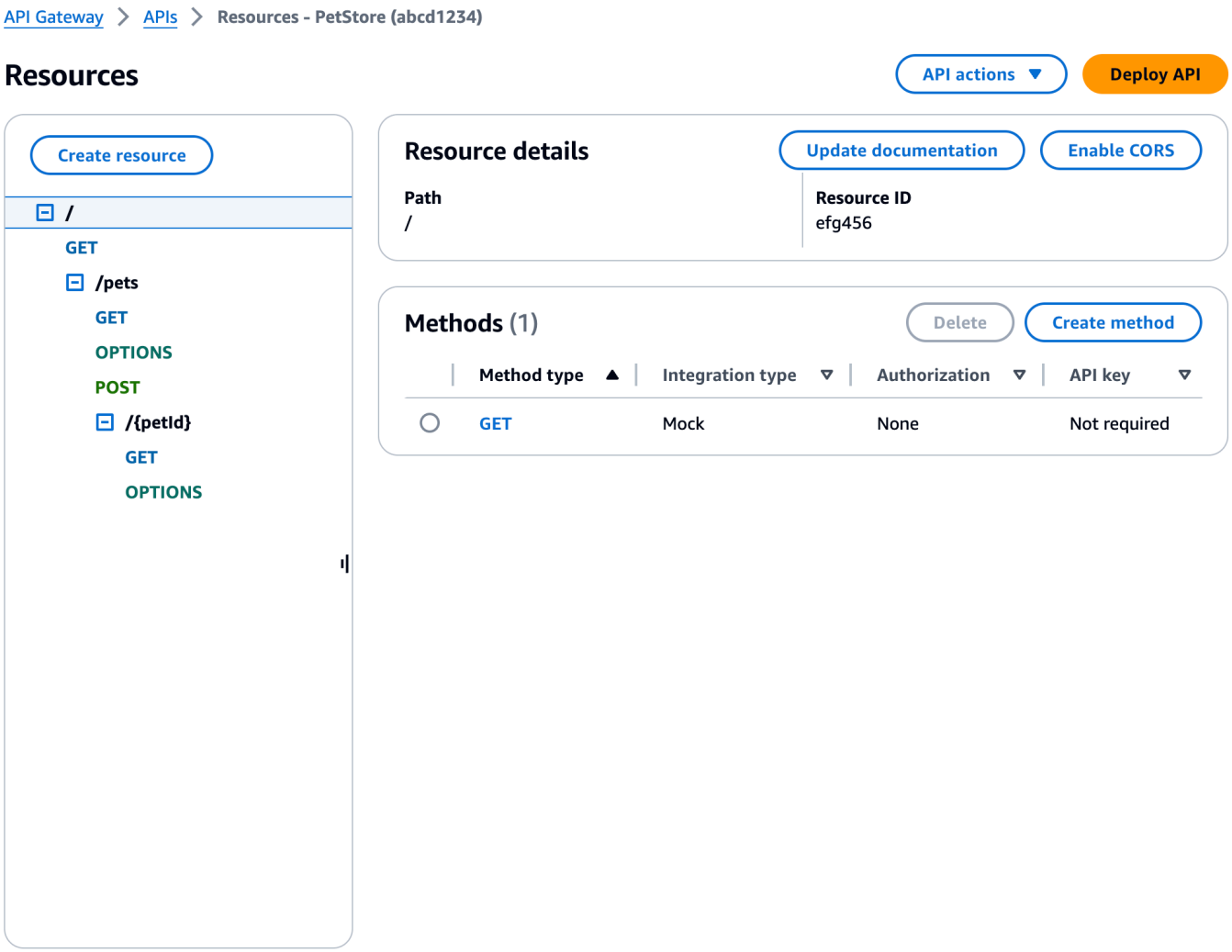
1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.

2. Esegui una delle seguenti operazioni:
   - Per creare la tua prima API, in REST API, scegli Crea.
   - Se hai già creato un'API, scegli Crea API, quindi scegliere Crea per API REST.
3. In Crea REST API scegli API di esempio, quindi seleziona Crea API per creare l'API di esempio.



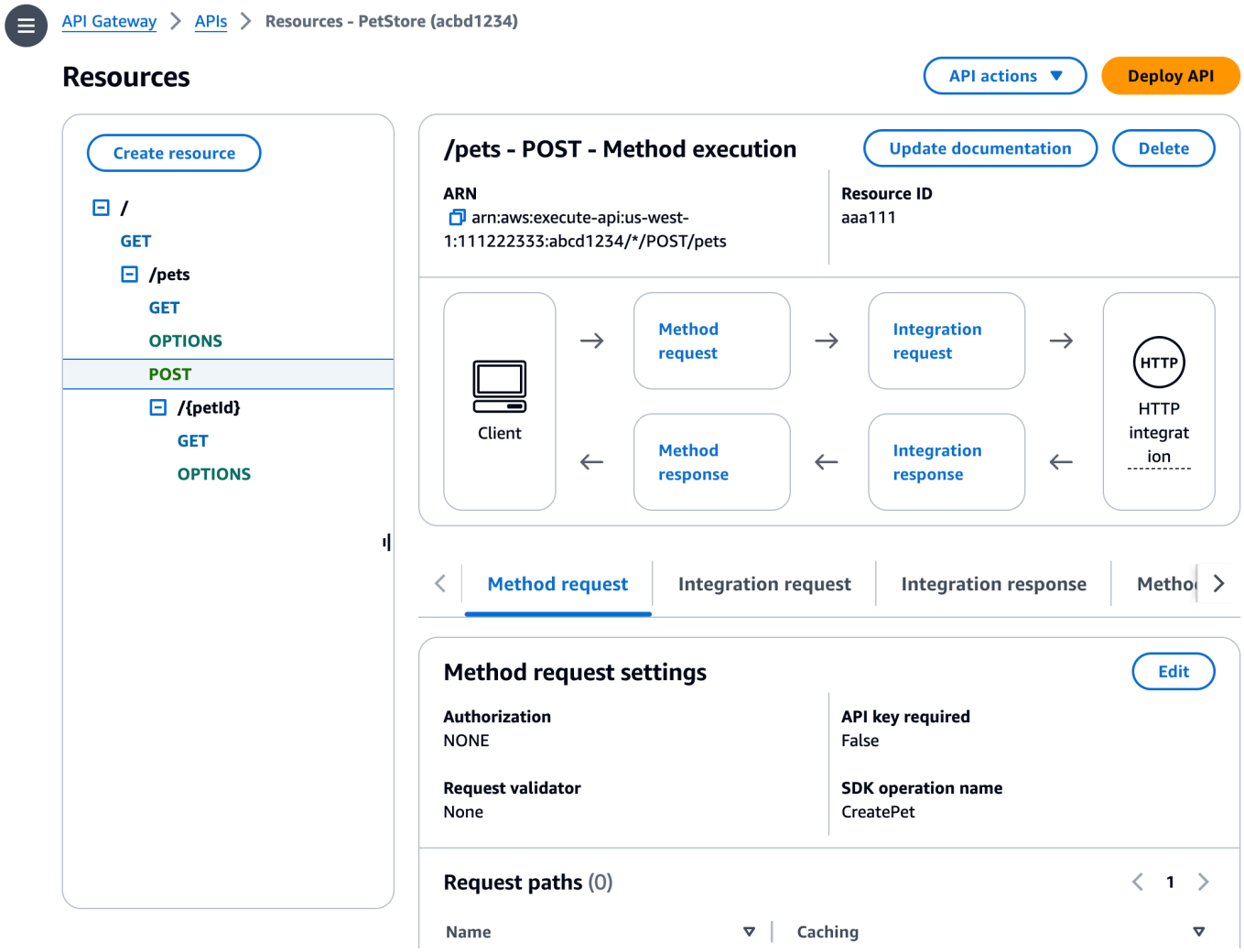
È possibile scorrere la definizione OpenAPI per i dettagli relativi a questa API di esempio prima di scegliere Crea API.

4. Nel riquadro di navigazione principale scegli Risorse. L'API appena creata viene mostrata come segue:



Il riquadro Resources (Risorse) mostra la struttura dell'API creata come albero di nodi. I metodi API definiti in ogni risorsa sono le estremità dell'albero. Quando una risorsa viene selezionata, tutti i relativi metodi vengono elencati nella tabella Metodi a destra. Con ogni metodo vengono visualizzati il tipo di metodo, il tipo di integrazione, il tipo di autorizzazione e il requisito della chiave API.

5. Per visualizzare i dettagli di un metodo, modificarne la configurazione o testarne la chiamata, scegli il nome del metodo nell'elenco o nella struttura di risorse. In questo caso scegliamo il metodo `POST /pets` a scopo illustrativo:



Il riquadro Esecuzione metodo risultante presenta una vista logica della struttura e dei comportamenti del metodo scelto (`POST /pets`).

La Richiesta metodo e la Risposta metodo rappresentano l'interfaccia dell'API con il front-end, mentre la Richiesta di integrazione e la Risposta di integrazione rappresentano l'interfaccia dell'API con il back-end.

Un client usa l'API per accedere a una funzionalità di back-end tramite la Richiesta metodo. Gateway API converte la richiesta del client, se necessario, nel formato accettabile per il back-end nella Richiesta di integrazione, prima di inoltrare la richiesta in entrata al back-end. La richiesta trasformata è nota come richiesta di integrazione. Analogamente, il back-end restituisce la risposta a Gateway API nella Risposta di integrazione. API Gateway quindi la instrada a Method Response (Risposta metodo) prima di inviarla al client. Anche in questo caso, se

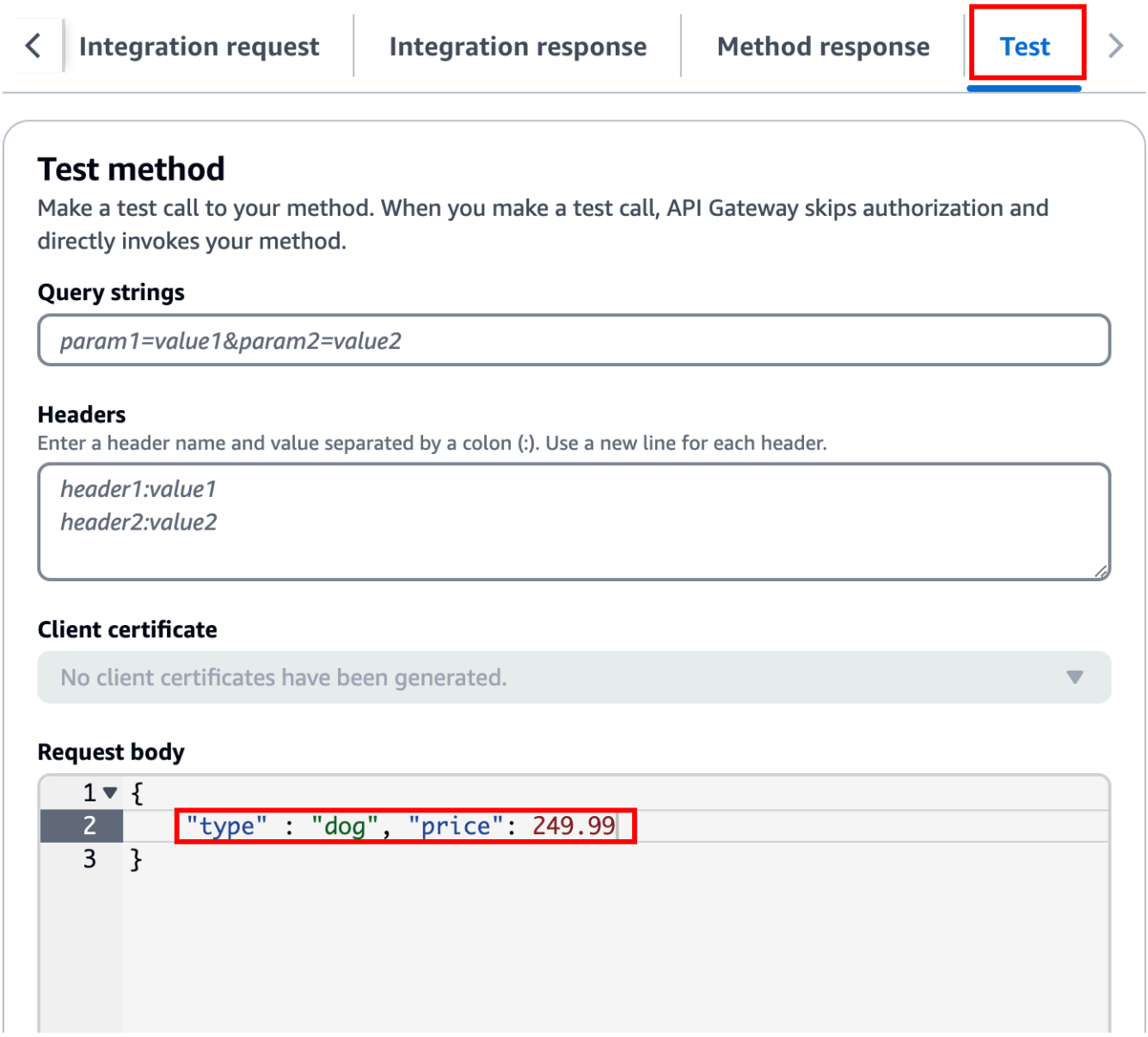
necessario, API Gateway può mappare i dati della risposta di back-end a un formato previsto dal client.

Per il metodo `POST` su una risorsa API, il payload della richiesta del metodo può essere passato attraverso la richiesta di integrazione senza alcuna modifica, se il formato del payload della richiesta del metodo è uguale a quello del payload della richiesta di integrazione.

La richiesta del metodo `GET /` usa il tipo di integrazione `MOCK` e non è legata a un endpoint di back-end reale. La Risposta di integrazione corrispondente è configurata per restituire una pagina HTML statica. Quando viene chiamato il metodo, Gateway API accetta la richiesta e restituisce immediatamente al client la risposta di integrazione configurata mediante la Risposta metodo. Puoi usare l'integrazione fittizia per testare un'API senza richiedere un endpoint di back-end. Puoi usarla anche per fornire una risposta locale, generata da un modello di mappatura del corpo della risposta.

In qualità di sviluppatore dell'API, puoi controllare i comportamenti delle interazioni front-end dell'API mediante la configurazione della richiesta e della risposta di metodo. Puoi inoltre controllare i comportamenti delle interazioni back-end dell'API mediante la configurazione della richiesta e della risposta di integrazione. Ciò prevede le mappature dei dati tra un metodo e l'integrazione corrispondente. Ora ci focalizziamo sul test dell'API per offrire un'esperienza utente end-to-end.

6. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
7. Ad esempio, per testare il metodo `POST /pets`, immetti il payload **`{"type": "dog","price": 249.99}`** seguente nel Corpo della richiesta prima di selezionare Test.



L'input specifica gli attributi dell'animale da aggiungere all'elenco degli animali domestici sul PetStore sito web.

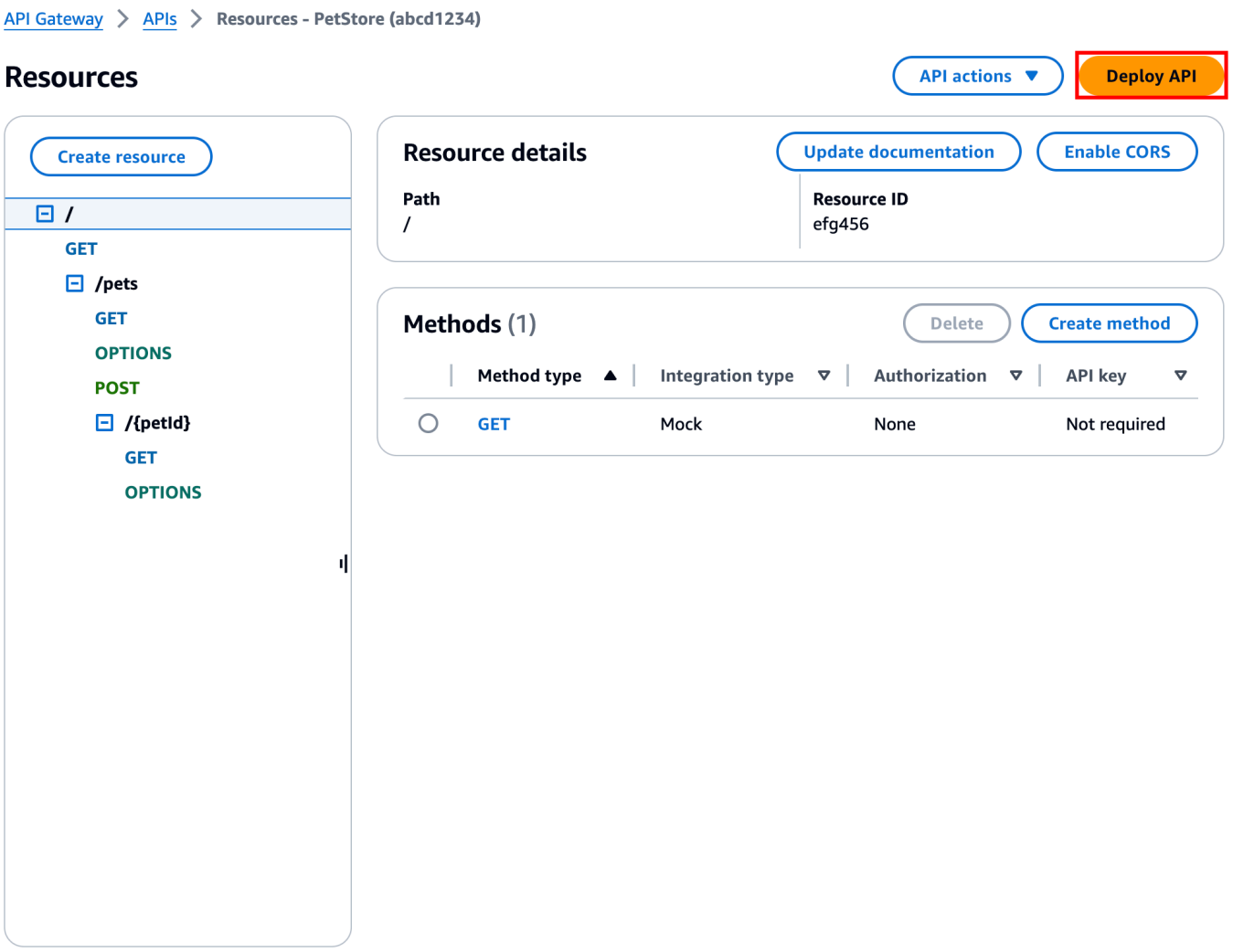
8. I risultati visualizzati sono simili ai seguenti:



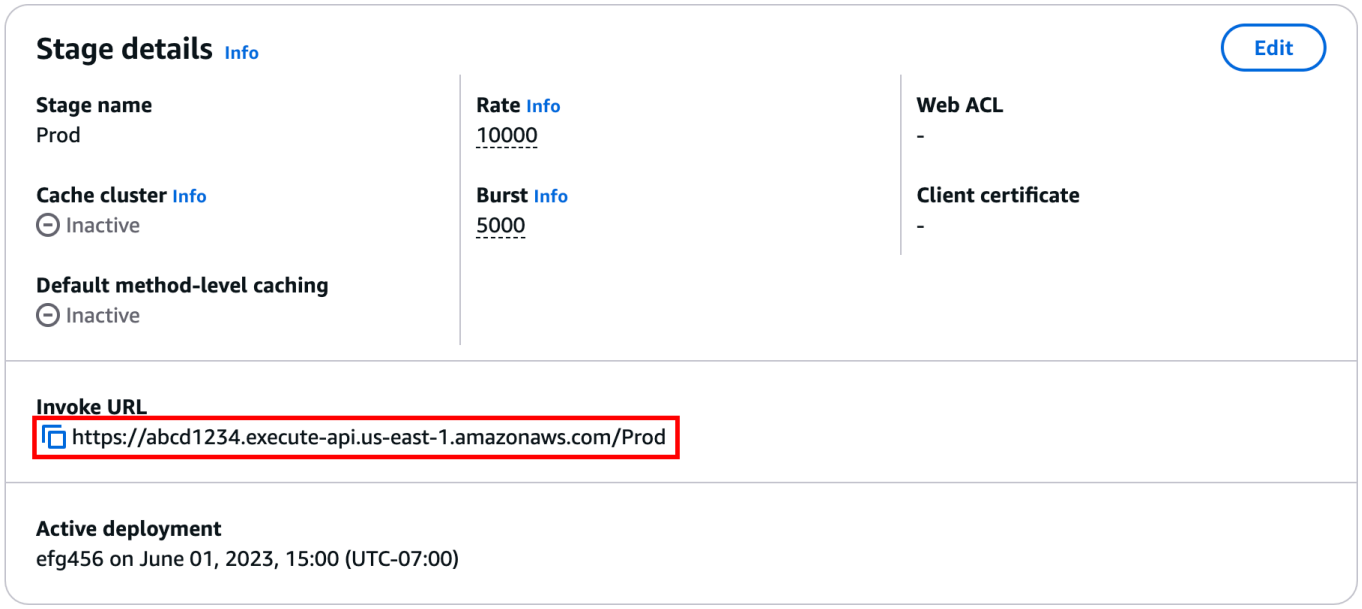
La voce Log dell'output mostra le modifiche dello stato dalla richiesta di metodo alla richiesta di integrazione e dalla risposta di integrazione alla risposta di metodo. Può risultare utile per la risoluzione dei problemi legati agli errori di mappatura che impediscono la riuscita della richiesta. In questo esempio non viene applicata alcuna mappatura: il payload della richiesta di metodo viene passato al back-end attraverso la richiesta di integrazione e, analogamente, la risposta del back-end viene passata alla risposta del metodo attraverso la risposta di integrazione.

Per testare l'API utilizzando un client diverso dalla caratteristica test-invoke-request di API Gateway, devi prima distribuire l'API in una fase.

9. Seleziona Implementa API per implementare l'API di esempio.



10. Per Fase seleziona Nuova fase, quindi immetti **`test`**.
11. (Facoltativo) In Descrizione, immetti una descrizione.
12. Seleziona Implementa.
13. Nel riquadro Fasi risultante, il campo Richiama URL in Dettagli fase visualizza l'URL per richiamare la richiesta del metodo `GET /` dell'API.



14. Scegli l'icona Copia per copiare l'URL di richiamo dell'API, quindi immetti l'URL di richiamo dell'API in un browser Web. La risposta con esito positivo restituisce il risultato, generato dal modello di mappatura nella risposta di integrazione.
15. Nel pannello di navigazione Stages (Fasi), espandere la fase test, selezionare GET per `/pets/{petId}`, quindi copiare il valore Invoke URL (URL chiamata) di `https://`*`api-id`*`.execute-api.`*`region`*`.amazonaws.com/test/pets/{petId}`. `{petId}` indica una variabile di percorso.

    Incollare il valore Invoke URL (URL chiamata) ottenuto nella fase precedente nella barra degli indirizzi di un browser, sostituendo `{petId}` con, ad esempio, `1` e premere Invio per inviare la richiesta. Dovrebbe essere restituita una risposta 200 OK con il payload JSON seguente:

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

    L'invocazione del metodo API nel modo indicato è possibile in quanto il tipo di Authorization (Autorizzazione) è impostato su `NONE`. Se venisse utilizzata l'autorizzazione `AWS_IAM`, la richiesta verrebbe firmata con i protocolli Signature Version 4 (SigV4) o Signature Version 4a (SigV4a). Per un esempio di questo tipo di richiesta, consulta the section called "Tutorial: creazione di una REST API con un'integrazione non proxy HTTP".

Scelta di un tutorial per l'integrazione HTTP

Per creare un'API con l'integrazione HTTP, è possibile utilizzare l'integrazione proxy HTTP o l'integrazione personalizzata HTTP.

Per l'integrazione proxy HTTP, è necessario solo impostare il metodo HTTP e l'URI dell'endpoint HTTP, in base ai requisiti del backend. Si consiglia, se possibile, di utilizzare l'integrazione proxy HTTP per semplificare la configurazione dell'API.

L'integrazione personalizzata HTTP può risultare utile se è necessario trasformare i dati delle richieste client per il backend o i dati delle risposte del backend per il client.

Argomenti

- Tutorial: creazione di una REST API con l'integrazione proxy HTTP
- Tutorial: creazione di una REST API con un'integrazione non proxy HTTP

Tutorial: creazione di una REST API con l'integrazione proxy HTTP

L'integrazione proxy HTTP è un meccanismo semplice, potente e versatile per creare un'API che permetta a un'applicazione Web di accedere a più caratteristiche o risorse dell'endpoint HTTP integrato, ad esempio l'intero sito Web, con una configurazione semplificata di un solo metodo API. In un'integrazione proxy HTTP, API Gateway passa la richiesta di metodo inviata dal client al back-end. I dati di richiesta che vengono passati includono intestazioni di richiesta, parametri delle stringhe di query, variabili di percorso URL e payload. L'endpoint HTTP di back-end o il server Web analizza i dati delle richieste in ingresso per stabilire la risposta da restituire. L'integrazione proxy HTTP consente l'interazione diretta tra client e back-end senza alcun intervento da parte di API Gateway dopo la configurazione del metodo API, tranne nel caso di problemi noti elencati in , ad esempio caratteri non supportat the section called “Note importanti”.

Con la risorsa proxy globale `{proxy+}` e il verbo catch-all `ANY` per il metodo HTTP, puoi usare un'integrazione proxy HTTP per creare un'API di un metodo API singolo. Il metodo espone l'intero set di operazioni e risorse HTTP accessibili pubblicamente di un sito Web. Quando il server Web di back-end apre più risorse per l'accesso pubblico, il client può usare le nuove risorse con la stessa configurazione API. Per consentirlo, lo sviluppatore del sito Web deve comunicare chiaramente con lo sviluppatore del client in merito alla natura delle nuove risorse e alle operazioni applicabili per ciascuna di esse.

Come breve introduzione, il tutorial che segue fornisce un esempio di integrazione proxy HTTP. Nel tutorial, creiamo un'API utilizzando la console API Gateway per l'integrazione con il PetStore sito Web tramite una risorsa `{proxy+}` proxy generica e creiamo il segnaposto del metodo HTTP di. `ANY`

Argomenti

- Creazione di un'API con l'integrazione proxy HTTP mediante la console API Gateway
- Verifica di un'API con l'integrazione proxy HTTP

Creazione di un'API con l'integrazione proxy HTTP mediante la console API Gateway

Nella procedura seguente vengono descritte le fasi necessarie per creare e testare un'API con una risorsa proxy per un back-end HTTP mediante la console API Gateway. Il back-end HTTP è il sito Web `PetStore` (`http://petstore-demo-endpoint.execute-api.com/petstore/pets`) da Tutorial: creazione di una REST API con un'integrazione non proxy HTTP, in cui vengono usati screenshot come supporto visivo per illustrare gli elementi dell'interfaccia utente API Gateway. Se non hai esperienza di utilizzo della console API Gateway per la creazione di un'API, può essere di aiuto seguire prima tale sezione.

Come creare un'API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Se si utilizza API Gateway per la prima volta, verrà visualizzata una pagina che presenta le caratteristiche del servizio. In API REST, scegliere Crea. Quando appare il popup Create Example API (Crea API di esempio), scegliere OK.

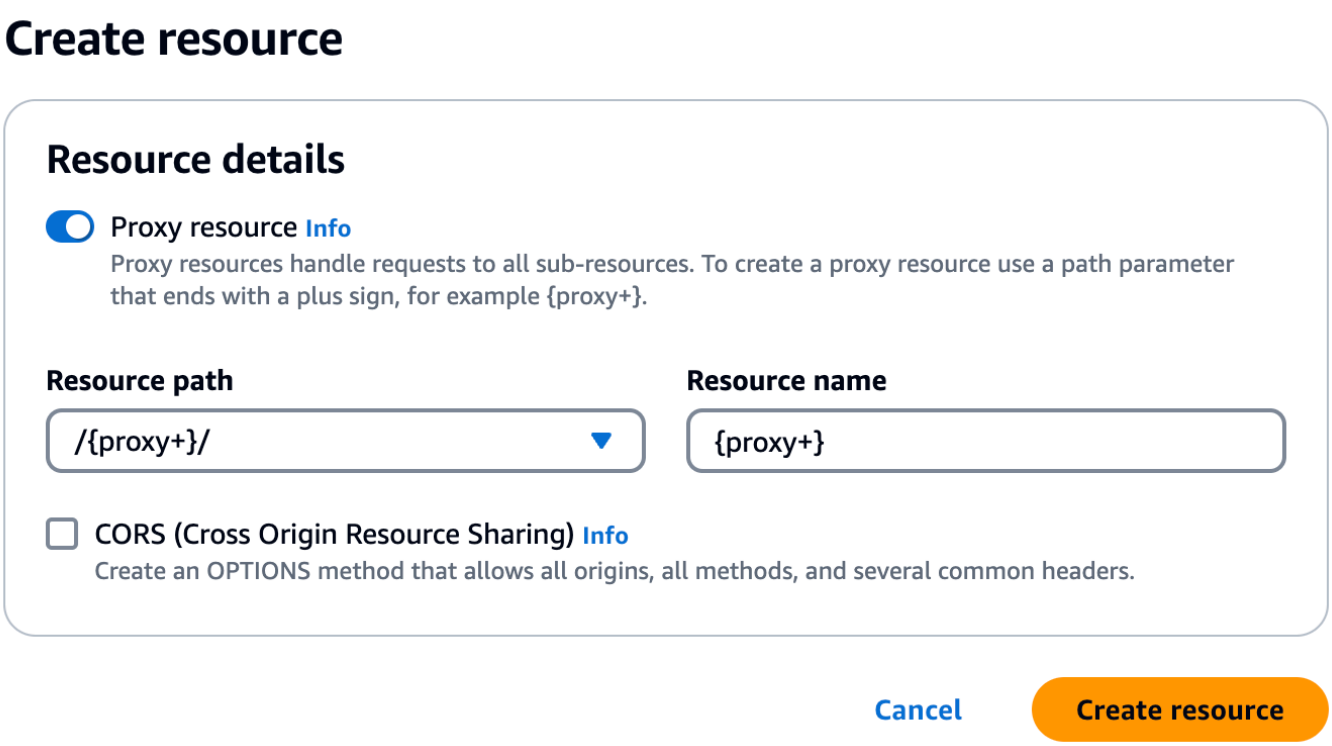
   Se non è la prima volta che si utilizza API Gateway, scegliere Create API (Crea API). In API REST, scegliere Crea.
3. Per API name (Nome API), immettere **`HTTPProxyAPI`**.
4. (Facoltativo) In Descrizione, immetti una descrizione.
5. Lasciare l'opzione Tipo di endpoint API impostata su Regionale.
6. Per Tipo di indirizzo IP, selezionare IPv4.
7. Seleziona Create API (Crea API).

In questa fase crei un percorso della risorsa proxy di `{proxy+}`. Questo è il segnaposto di un endpoint di back-end in `http://petstore-demo-endpoint.execute-api.com/`. Ad esempio, può essere `petstore`, `petstore/pets` e `petstore/pets/{petId}`. Quando crei la risorsa

`{proxy+}`, Gateway API crea il metodo ANY che funge da segnaposto per qualsiasi verbo HTTP supportato al runtime.

Per creare un /{proxy+} risorsa

1. Scegliere l'API.
2. Nel riquadro di navigazione principale scegli Risorse.
3. Scegli Crea risorsa.
4. Attiva Risorsa proxy.
5. Mantieni Percorso risorsa impostato su `/`.
6. Per Resource Name (Nome risorsa) immetti **`{proxy+}`**.
7. Mantieni CORS (Cross Origin Resource Sharing) disattivato.
8. Scegli Crea risorsa.



In questa fase integri il metodo ANY con un endpoint HTTP di back-end, utilizzando un'integrazione proxy. In un'integrazione proxy, Gateway API passa la richiesta del metodo inviata dal client al back-end senza intervento di Gateway API.

Per creare un metodo ANY

1. Scegli la risorsa /{proxy+}.
2. Scegli il metodo ANY.
3. Sotto il simbolo di avviso, scegli Modifica integrazione. Non puoi implementare un'API con un metodo senza un'integrazione.
4. Per Tipo di integrazione seleziona HTTP.
5. Attiva l'opzione Integrazione proxy HTTP.
6. Per Metodo HTTP seleziona ANY.
7. Per URL dell'endpoint immetti **http://petstore-demo-endpoint.execute-api.com/{proxy}**.
8. Scegli Save (Salva).

Verifica di un'API con l'integrazione proxy HTTP

L'esito di una richiesta client specifica dipende dagli elementi seguenti:

- Se il back-end ha reso disponibile l'endpoint di back-end corrispondente e, in caso affermativo, se ha concesso le autorizzazioni di accesso richieste.
- Se il client fornisce l'input corretto.

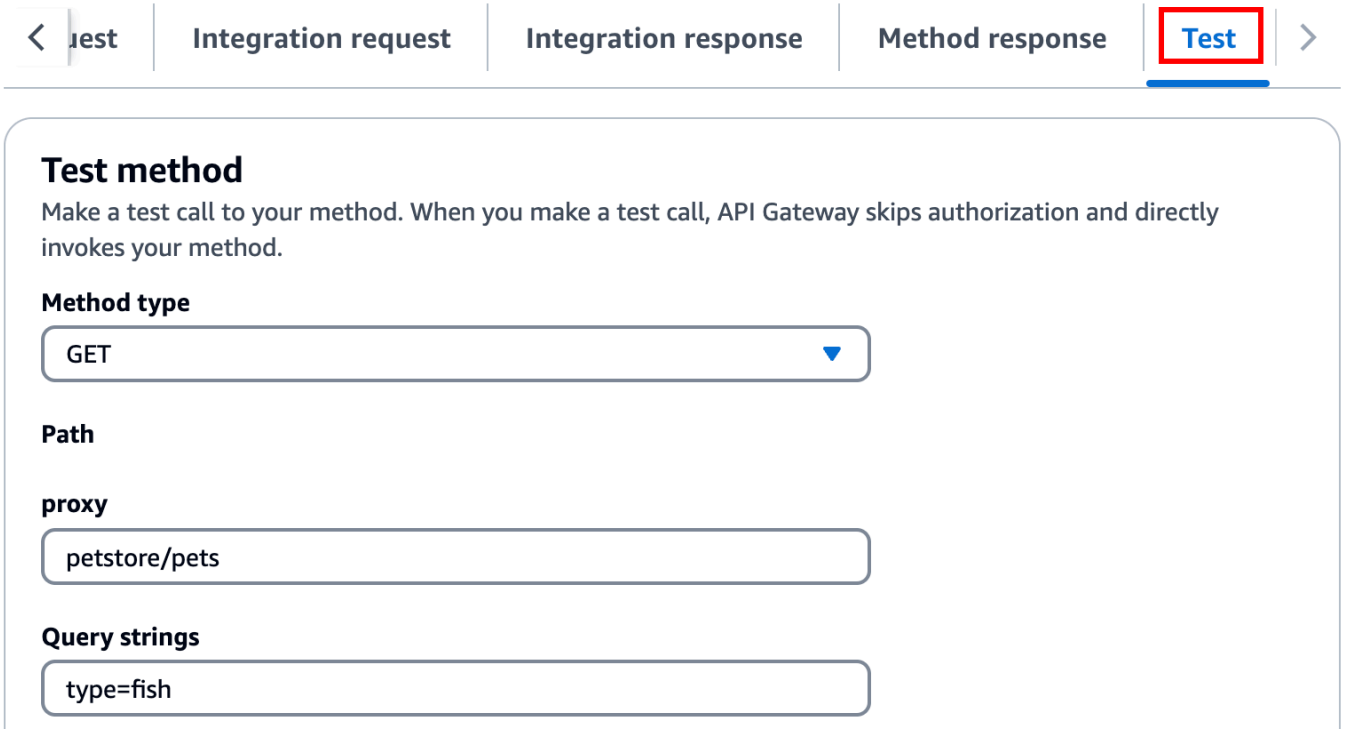
Ad esempio, l' PetStore API utilizzata qui non espone la `/petstore` risorsa. In questo modo, ottieni una risposta `404 Resource Not Found` contenente il messaggio di errore `Cannot GET /petstore`.

Inoltre per analizzare correttamente il risultato, il client deve essere in grado di gestire il formato di output del back-end. API Gateway non svolge alcuna mediazione per semplificare le interazioni tra il client e il back-end.

Per testare un'API integrata con il PetStore sito Web utilizzando l'integrazione del proxy HTTP tramite la risorsa proxy

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. In Tipo di metodo, seleziona `GET`.
3. In Percorso immetti **petstore/pets** per proxy.

4. In Stringhe di query immetti **`type=fish`**.
5. Scegli Test (Esegui test).



Poiché supporta la richiesta `GET /petstore/pets?type=fish`, il sito Web di back-end restituisce una risposta con esito positivo simile alla seguente:

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

```
]
```

Se provi a chiamare `GET /petstore`, ottieni una risposta `404` con il messaggio di errore `Cannot GET /petstore`. Questo perché il back-end non supporta l'operazione specificata. Se chiami`GET /petstore/pets/1`, riceverai una `200 OK` risposta con il seguente payload, poiché la richiesta è supportata dal PetStore sito Web.

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

Puoi anche utilizzare un browser per testare l'API. Implementa l'API e associala a una fase per creare l'URL di richiamata dell'API.

Per distribuire l'API

1. Seleziona Deploy API (Distribuisci API).
2. In Fase, seleziona Nuova fase.
3. In Stage name (Nome fase) immettere **`test`**.
4. (Facoltativo) In Descrizione, immetti una descrizione.
5. Seleziona Implementa.

Ora i client possono chiamare la tua API.

Richiamo dell'API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere l'API.
3. Nel pannello di navigazione principale scegli Fase.
4. In Dettagli fase, scegli l'icona Copia per copiare l'URL di richiamo dell'API.

   Immetti l'URL di richiamata dell'API in un browser Web.

   L'URL completo dovrebbe essere del tipo `https://`*`abcdef123`*`.execute-api.`*`us-east-2`*`.amazonaws.com/`*`test`*`/petstore/pets?type=fish`.

Il tuo browser invia una richiesta `GET` all'API.

5. Il risultato sarà uguale a quello che viene restituito quando si usa Test nella console Gateway API.

Tutorial: creazione di una REST API con un'integrazione non proxy HTTP

In questo tutorial viene creata un'API da zero utilizzando la console Amazon API Gateway. Puoi pensare alla console come a uno studio di progettazione delle API e utilizzarla per analizzare le caratteristiche delle API, sperimentare i loro comportamenti, crearle e poi distribuirle per fasi.

Argomenti

- Creazione di un'API con l'integrazione personalizzata HTTP
- (Facoltativo) Mappatura dei parametri di richiesta

Creazione di un'API con l'integrazione personalizzata HTTP

Questa sezione illustra per fasi come creare le risorse, esporre i metodi su una risorsa, configurare un metodo per ottenere i comportamenti dell'API desiderati, testar e distribuire le API.

In questa fase si crea un'API vuota. Nelle seguenti fasi crei risorse e metodi per connettere l'API all'endpoint `http://petstore-demo-endpoint.execute-api.com/petstore/pets`, utilizzando un'integrazione HTTP non proxy.

Come creare un'API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Se si utilizza API Gateway per la prima volta, verrà visualizzata una pagina che presenta le caratteristiche del servizio. In API REST, scegliere Crea. Quando appare il popup Create Example API (Crea API di esempio), scegliere OK.

   Se non è la prima volta che si utilizza API Gateway, scegliere Create API (Crea API). In API REST, scegliere Crea.
3. Per API name (Nome API), immettere **`HTTPNonProxyAPI`**.
4. (Facoltativo) In Descrizione, immetti una descrizione.
5. Lasciare l'opzione Tipo di endpoint API impostata su Regionale.
6. Per Tipo di indirizzo IP, selezionare IPv4.

7. Seleziona Create API (Crea API).

L'albero Resources (Risorse) mostra la risorsa root (`/`) senza metodi. In questo esercizio, creeremo l'API con l'integrazione personalizzata HTTP del PetStore sito Web (http://petstore-demo-endpoint.execute-api.com/petstore/pets.) A scopo illustrativo, creeremo una `/pets` risorsa come elemento secondario della radice ed esporremo un metodo GET su questa risorsa affinché un client possa recuperare un elenco di elementi Pets disponibili dal sito Web. PetStore

Per creare una risorsa /pets

1. Scegli Crea risorsa.
2. Mantieni l'opzione Risorsa proxy disattivata.
3. Mantieni Percorso risorsa impostato su `/`.
4. Per Resource Name (Nome risorsa) immetti **`pets`**.
5. Mantieni CORS (Cross Origin Resource Sharing) disattivato.
6. Scegli Crea risorsa.

In questa fase crei un metodo `GET` per la risorsa /pets. Il metodo `GET` è integrato con il sito Web `http://petstore-demo-endpoint.execute-api.com/petstore/pets`. Altre opzioni per il metodo API sono:

- POST, usato principalmente per creare risorse figlio.
- PUT, utilizzato principalmente per aggiornare risorse esistenti (e, sebbene non consigliato, può essere usato per creare risorse figlio).
- DELETE, utilizzato per eliminare le risorse.
- PATCH, utilizzato per aggiornare le risorse.
- HEAD, utilizzato principalmente negli scenari di test. È lo stesso di GET ma non restituisce la rappresentazione della risorsa.
- OPTIONS, che può essere utilizzato dagli intermediari per ricevere informazioni sulle opzioni di comunicazione disponibili per il servizio di destinazione.

Per HTTP method (Metodo HTTP) della richiesta di integrazione, occorre selezionarne uno supportato dal back-end. Per `HTTP` o `Mock integration`, è opportuno che la richiesta di metodo e la richiesta di integrazione utilizzino lo stesso verbo HTTP. Per altri tipi di integrazione, la richiesta

di metodo utilizzerà probabilmente un verbo HTTP diverso da quello della richiesta di integrazione. Ad esempio, per chiamare una funzione Lambda, la richiesta di integrazione deve utilizzare `POST` per invocare la funzione, mentre la richiesta di metodo potrebbe utilizzare qualsiasi verbo HTTP in base alla logica della funzione Lambda.

Per creare un metodo **`GET` su** /animali risorsa

1. Seleziona la risorsa /pets.
2. Scegli Crea metodo.
3. Per Tipo di metodo seleziona GET.
4. Per Tipo di integrazione seleziona Integrazione HTTP.
5. Mantieni l'opzione Integrazione proxy HTTP disattivata.
6. Per Metodo HTTP seleziona GET.
7. Per URL dell'endpoint immetti **`http://petstore-demo-endpoint.execute-api.com/petstore/pets`**.

   Il PetStore sito Web consente di recuperare un elenco di `Pet` elementi in base al tipo di animale domestico, ad esempio «Cane» o «Gatto», in una determinata pagina.
8. Per Gestione contenuti seleziona Transito.
9. Scegli Parametri della stringa di query URL.

   Il PetStore sito Web utilizza `type` i parametri della stringa di `page` interrogazione per accettare un input. Aggiungi i parametri della stringa di query alla richiesta del metodo e mappali nei parametri della stringa di query corrispondenti della richiesta di integrazione.
10. Per aggiungere i parametri della stringa di query, procedi come indicato di seguito:

    a. Scegliere Add query string (Aggiungi stringa di query).

    b. Per Nome immetti **`type`**.

    c. Mantieni Obbligatorio e Caching disattivati.

    Ripeti le fasi precedenti per creare una stringa di query aggiuntiva denominata **`page`**.
11. Scegli Crea metodo.

Se invii una richiesta, ora il client può fornire un tipo di animale e un numero di pagina come parametri della stringa di query. Questi parametri di input devono essere mappati nei parametri della stringa di query dell'integrazione per inoltrare i valori di input al nostro PetStore sito Web nel backend.

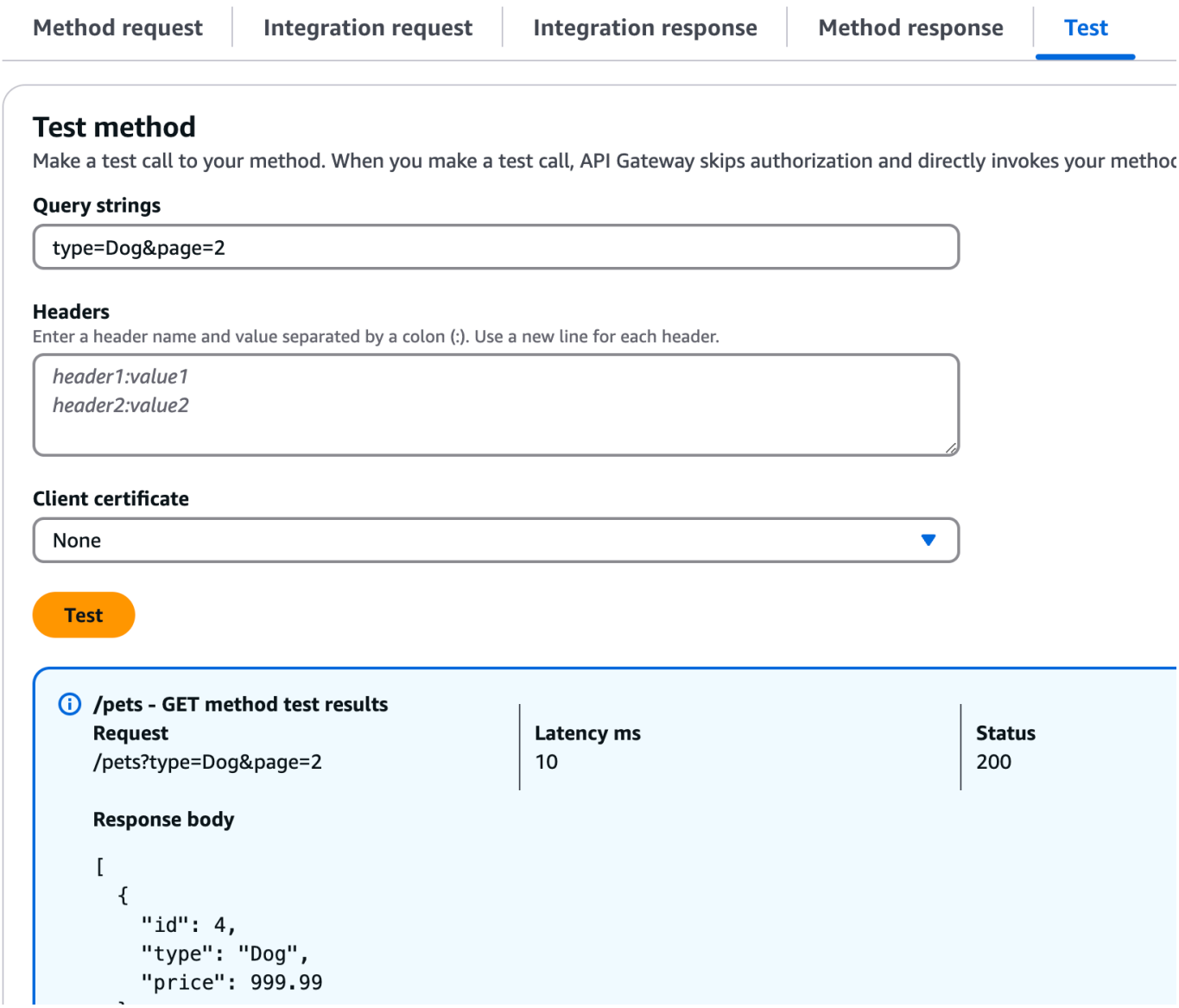
Per mappare i parametri di input alla richiesta di integrazione

1. Nella scheda Richiesta di integrazione scegli Modifica in Impostazioni della richiesta di integrazione.
2. Scegli Parametri della stringa di query URL ed effettua le seguenti operazioni:
   a. Scegli Aggiungi parametro della stringa di query.
   b. In Nome, inserisci **`type`**.
   c. Per Mappato da immetti **`method.request.querystring.type`**.
   d. Mantieni disattivata l'opzione Caching.
   e. Scegli Aggiungi parametro della stringa di query.
   f. In Nome, inserisci **`page`**.
   g. Per Mappato da immetti **`method.request.querystring.page`**.
   h. Mantieni disattivata l'opzione Caching.
3. Scegli Save (Salva).

Per testare l'API

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. In Stringhe di query immetti **`type=Dog&page=2`**.
3. Scegli Test (Esegui test).

   Il risultato è simile al seguente:



Ora che il test è andato a buon fine, puoi distribuire l'API per renderla pubblicamente disponibile.

4. Seleziona Deploy API (Distribuisci API).
5. In Fase, seleziona Nuova fase.
6. In Stage name (Nome fase) immettere **`Prod`**.
7. (Facoltativo) In Descrizione, immetti una descrizione.
8. Seleziona Implementa.
9. (Facoltativo) In Dettagli fase puoi scegliere l'icona Copia per copiare l'URL di richiamata dell'API in Richiama URL. È possibile usare questo valore con strumenti come Postman e cURL per testare l'API.

Se utilizzi un SDK per creare un client, puoi chiamare i metodi esposti dall'SDK per firmare la richiesta. Per maggiori dettagli sull'implementazione, consulta il kit SDK AWS scelto.

> **Note**
>
> Se apporti modifiche alla tua API, devi ridistribuirla per fare in modo che le caratteristiche nuove o aggiornate siano disponibili prima di chiamare di nuovo l'URL di richiesta.

(Facoltativo) Mappatura dei parametri di richiesta

Mappatura di parametri di richieste per un'API di API Gateway

Questo tutorial illustra come creare il parametro di percorso `{petId}` nella richiesta del metodo dell'API in modo da specificare un ID elemento, mapparlo al parametro di percorso `{id}` nell'URL della richiesta di integrazione e inviare la richiesta all'endpoint HTTP.

> **Note**
>
> Se immetti una lettera maiuscola o minuscola errata, ad esempio una lettera minuscola anziché maiuscola, verranno restituiti errori più avanti nella procedura dettagliata.

Fase 1: Creazione delle risorse

In questa fase crei una risorsa con un parametro di percorso {petId}.

Per creare la risorsa {petId}

1. Seleziona la risorsa /pets, quindi scegli Crea risorsa.
2. Mantieni l'opzione Risorsa proxy disattivata.
3. Per Percorso risorsa seleziona /pets.
4. Per Resource Name (Nome risorsa) immetti **`{petId}`**.

   Utilizza le parentesi graffe (`{ }`) per racchiudere `petId`, in modo da visualizzare /pets/{petId}.
5. Mantieni CORS (Cross Origin Resource Sharing) disattivato.
6. Scegli Crea risorsa.

Fase 2: Creazione e test dei metodi

In questa fase crei un metodo `GET` con un parametro di percorso `{petId}`.

Per configurare il metodo GET

1. Seleziona la risorsa /{petId}, quindi scegli Crea metodo.
2. Per Tipo di metodo seleziona GET.
3. Per Tipo di integrazione seleziona Integrazione HTTP.
4. Mantieni l'opzione Integrazione proxy HTTP disattivata.
5. Per Metodo HTTP seleziona GET.
6. Per URL dell'endpoint immetti **`http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}`**.
7. Per Gestione contenuti seleziona Transito.
8. Mantieni attivata l'opzione Timeout predefinito.
9. Scegli Crea metodo.

A questo punto mappare il parametro di percorso `{petId}` appena creato al parametro di percorso `{id}` nell'URL dell'endpoint HTTP della richiesta di integrazione. L'URL dell'endpoint HTTP è **`http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}`**.

Per mappare il **`parametro {petId} path`**

1. Nella scheda Richiesta di integrazione scegli Modifica in Impostazioni della richiesta di integrazione.
2. Scegli Parametri di percorso URL.
3. Gateway API crea un parametro di percorso per la richiesta di integrazione denominato petId che tuttavia non è valido per l'URL dell'endpoint HTTP impostato come integrazione backend. L'endpoint HTTP utilizza `{id}` come parametro di percorso. Per Nome, elimina petId e inserisci **`id`**.

   In questo modo, verrà mappato il parametro di percorso `petId` della richiesta del metodo al parametro di percorso `id` della richiesta di integrazione.
4. Scegli Save (Salva).

Quindi testa il metodo.

Test del metodo

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. In Percorso immetti **4** per petId.
3. Scegli Test (Esegui test).

   Se il test ha esito positivo, in Corpo della risposta viene visualizzato quanto segue:

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

Fase 3: Distribuzione dell'API

In questa fase, viene distribuita l'API per poter iniziare a richiamarla esternamente alla console API Gateway.

Per distribuire l'API

1. Seleziona Deploy API (Distribuisci API).
2. Per Fase seleziona Produzione.
3. (Facoltativo) In Descrizione, immetti una descrizione.
4. Selezionare Deploy (Distribuisci).

Fase 4: Test dell'API

In questa fase verrà chiusa la console API Gateway e utilizzata l'API per accedere all'endpoint HTTP.

1. Nel pannello di navigazione principale scegli Fase.
2. In Dettagli fase, scegli l'icona Copia per copiare l'URL di richiamo dell'API.

   Dovrebbe essere simile a quanto segue:

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

3. Inserisci questo URL nella casella dell'indirizzo di una nuova scheda del browser e aggiungi `/pets/4` all'URL prima di inviare la richiesta.
4. Il browser restituirà quanto segue:

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

Fasi successive

Puoi personalizzare ulteriormente la tua API attivando la convalida delle richieste, trasformando i dati o creando risposte del gateway personalizzate.

Per scoprire altri modi per personalizzare l'API, consulta i seguenti tutorial:

- Per ulteriori informazioni sulla convalida della richiesta, consulta Configurazione della convalida di base delle richieste in API Gateway.
- Per informazioni su come trasformare i payload di richiesta e risposta, consulta Tutorial: modifica la richiesta di integrazione e la risposta per le integrazioni a AWS services.
- Per informazioni su come creare risposte del gateway personalizzate, consulta Impostare una risposta del gateway per un'API REST utilizzando la console API Gateway.

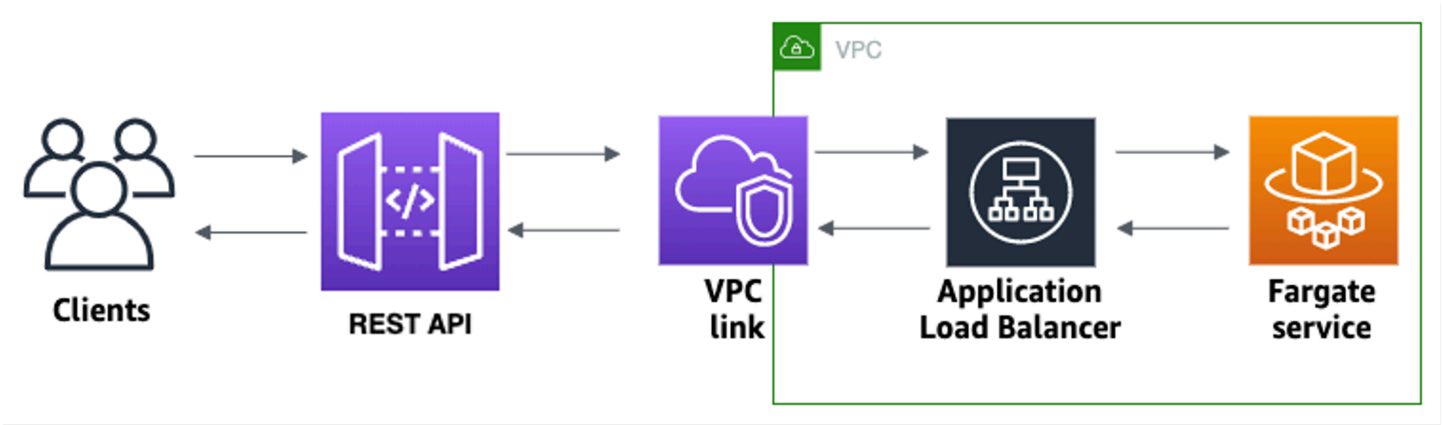
Tutorial: creazione di una REST API con un'integrazione privata

In questo tutorial, crei un'API REST che si connette a un servizio Amazon ECS eseguito in un Amazon VPC. I clienti al di fuori del tuo Amazon VPC possono utilizzare l'API per accedere al tuo servizio Amazon ECS.

Questo tutorial dura circa un'ora. Innanzitutto, utilizzi un CloudFormation modello per creare un servizio Amazon VPC e Amazon ECS. Quindi utilizzi la console API Gateway per creare un collegamento VPC V2. Il link VPC consente ad API Gateway di accedere al servizio Amazon ECS eseguito nel tuo VPC Amazon. Successivamente, crei un'API REST che utilizza il collegamento VPC V2 per connettersi al tuo servizio Amazon ECS. Infine, si esegue il dell'API.

Quando richiami la tua API REST, API Gateway indirizza la richiesta al tuo servizio Amazon ECS tramite il tuo link VPC V2, quindi restituisce la risposta dal servizio.

> **Note**
>
> Questo tutorial era precedentemente supportato per le API HTTP e ora è supportato per le API REST che utilizzano VPC link V2.



Per completare questo tutorial, sono necessari un AWS account e un AWS Identity and Access Management utente con accesso alla console. Per ulteriori informazioni, consulta Prerequisiti.

Argomenti

- Fase 1: Creazione di un servizio Amazon ECS
- Fase 2: Creazione di un link VPC
- Fase 3: Creare un'API REST
- Fase 4: Testa la tua API
- Fase 5: distribuzione dell'API
- Fase 6: Chiama la tua API
- Fase 7: pulire

Fase 1: Creazione di un servizio Amazon ECS

Amazon ECS è un servizio di gestione dei container che facilita l'esecuzione, l'arresto e la gestione di container Docker in un cluster. In questo tutorial eseguirai il cluster su un'infrastruttura serverless gestita da Amazon ECS.

Scarica e decomprimi questo CloudFormation modello, che crea tutte le dipendenze per il servizio, incluso un Amazon VPC. Si utilizza il modello per creare un servizio Amazon ECS che utilizza un Application Load Balancer.

Per creare un CloudFormation stack

1. Apri la CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation.
2. Scegliere Create stack (Crea stack), quindi With new resources (standard) (Con nuove risorse (standard)).
3. In Specificare modello, scegliere Carica un file modello.
4. Selezionare il modello scaricato.
5. Scegli Next (Successivo).
6. Per Stack name (Nome stack), inserire **`rest-api-private-integrations-tutorial`**, quindi scegliere Next (Avanti).
7. Per Configure stack options (Configura opzioni di stack), scegliere Next (Successivo).
8. Per quanto riguarda le funzionalità, riconosci che CloudFormation puoi creare risorse IAM nel tuo account.
9. Scegli Avanti, quindi scegli Invia.

CloudFormation esegue il provisioning del servizio ECS, operazione che può richiedere alcuni minuti. Quando lo stato del tuo CloudFormation stack è CREATE_COMPLETE, sei pronto per passare alla fase successiva.

Fase 2: Creazione di un link VPC

Un link VPC consente ad API Gateway di accedere alle risorse private in un VPC Amazon. Utilizzi un link VPC per consentire ai clienti di accedere al tuo servizio Amazon ECS tramite la tua API REST.

Per creare un link VPC

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel pannello di navigazione principale scegli Collegamenti VPC e quindi Crea.

   È possibile che per aprire il pannello di navigazione principale sia necessario scegliere l'icona del menu.
3. Per Scegli una versione di collegamento VPC, seleziona VPC link V2.

4. In Name (Nome), immettere **`private-integrations-tutorial`**.
5. Per VPC scegliere il VPC creato nella fase 1. Il nome deve iniziare con RestApiStack.
6. Per Sottoreti, selezionare le due sottoreti private nel VPC. I nomi finiscono con `PrivateSubnet`.
7. Per Gruppi di sicurezza seleziona l'ID gruppo che inizia con `private-integrations-tutorial` e ha la descrizione `RestApiStack/RestApiTutorialService/Service/SecurityGroup`.
8. Scegli Create (Crea).

Dopo aver creato il tuo VPC link V2, API Gateway fornisce interfacce di rete elastiche per accedere al tuo VPC. Il processo può richiedere alcuni minuti. Nel frattempo, puoi creare la tua API.

Fase 3: Creare un'API REST

L'API REST fornisce un endpoint HTTP per il tuo servizio Amazon ECS.

Creazione di una REST API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli Create API (Crea API), quindi per API REST, scegli Build (Crea).
3. In Nome, inserisci **`private-integration-api`**.
4. Per Tipo di indirizzo IP, selezionare IPv4.
5. Seleziona Create API (Crea API).

   Dopo aver creato l'API, crei un metodo.
6. Scegli Crea metodo e procedi come descritto di seguito:

   a. In Tipo di metodo, seleziona `GET`.
   b. In Tipo di integrazione seleziona Collegamento VPC.
   c. Attiva l'opzione Integrazione proxy VPC.
   d. Per Metodo HTTP seleziona `GET`.
   e. Per il collegamento VPC, scegli il collegamento VPC V2 che hai creato nel passaggio precedente.
   f. Per Integration Target, inserisci il load balancer che hai creato con il CloudFormation modello nel passaggio 1. Il nome dovrebbe iniziare con rest-.

g. Per URL dell'endpoint immetti `http://private-integrations-tutorial.com`.

   L'URL viene utilizzato per impostare l'`Host`intestazione della richiesta di integrazione. In questo caso, l'intestazione dell'host è. **`private-integrations-tutorial`**

h. Scegli Crea metodo.

   Con l'integrazione del proxy, l'API è pronta per essere testata.

Fase 4: Testa la tua API

Successivamente, provi a richiamare il metodo API.

Per testare l'API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere l'API.
3. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
4. Scegli Test

   Verifica che la risposta della tua API sia un messaggio di benvenuto che indica che la tua app è in esecuzione su Amazon ECS.

Fase 5: distribuzione dell'API

Successivamente, distribuisci la tua API.

Per distribuire l'API

1. Seleziona Deploy API (Distribuisci API).
2. In Fase, seleziona Nuova fase.
3. In Stage name (Nome fase) immettere **`Prod`**.
4. (Facoltativo) In Descrizione, immetti una descrizione.
5. Seleziona Implementa.

Fase 6: Chiama la tua API

Dopo aver distribuito l'API, puoi chiamarla.

Per chiamare la tua API

1. Immetti l'URL di richiamo in un browser Web.

   L'URL completo dovrebbe essere del tipo `https://abcd123.execute-api.us-east-2.amazonaws.com/Prod`.

   Il tuo browser invia una richiesta `GET` all'API.
2. Verifica che la risposta della tua API sia un messaggio di benvenuto che indica che la tua app è in esecuzione su Amazon ECS.

   Se vedi il messaggio di benvenuto, significa che hai creato con successo un servizio Amazon ECS che viene eseguito in un Amazon VPC e hai utilizzato un'API API Gateway REST con un collegamento VPC V2 per accedere al servizio Amazon ECS.

Fase 7: pulire

Per evitare costi non necessari, eliminare le risorse create nell'ambito di questo tutorial. I passaggi seguenti eliminano il collegamento VPC V2, lo CloudFormation stack e l'API REST.

Per eliminare un'API REST

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nella pagina API, selezionare un'API. Scegliere Azioni, scegliere Elimina, quindi confermare la scelta.

Per eliminare un link VPC

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere link VPC.
3. Selezionare il link VPC, scegliere Elimina, quindi confermare la scelta.

Per eliminare un CloudFormation stack

1. Apri la CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation.
2. Seleziona il tuo CloudFormation stack.
3. Scegli Elimina e conferma la tua scelta.

Tutorial: creare un'API REST con un' AWS integrazione

Gli argomenti Tutorial: creazione di una REST API con un'integrazione proxy Lambda e Tutorial: creazione di una REST API con un'integrazione non proxy Lambda descrivono entrambi come creare un'API di API Gateway per esporre la funzione Lambda integrata. Inoltre, puoi creare un'API API Gateway per esporre altri AWS servizi, come Amazon SNS, Amazon S3, Amazon Kinesis e persino. AWS Lambda Questo è possibile grazie all'integrazione di `AWS`. L'integrazione Lambda o l'integrazione proxy Lambda è un caso speciale, in cui la chiamata della funzione Lambda viene esposta tramite l'API di API Gateway.

Tutti i AWS servizi supportano un supporto dedicato per esporre APIs le proprie funzionalità. Tuttavia, è probabile che i protocolli applicativi o le interfacce di programmazione differiscano da servizio a servizio. Un'API API Gateway con `AWS` integrazione ha il vantaggio di fornire un protocollo applicativo coerente per consentire al cliente di accedere a diversi AWS servizi.

In questa procedura guidata creeremo un'API per esporre Amazon SNS. Per altri esempi di integrazione di un'API con altri AWS servizi, consultaTutorial e workshop di Amazon API Gateway.

Diversamente dall'integrazione proxy Lambda, non esiste un'integrazione proxy corrispondente per altri servizi AWS . Pertanto, un metodo API è integrato con una singola AWS azione. Per ottenere maggiore flessibilità, analogamente all'integrazione proxy, puoi configurare l'integrazione proxy Lambda. La funzione Lambda quindi analizza ed elabora le richieste di altre azioni. AWS

API Gateway non esegue un nuovo tentativo quando si verifica il timeout dell'endpoint. La chiamata API deve implementare una logica di ripetizione per gestire i timeout degli endpoint.

Questa procedura dettagliata si basa sulle istruzioni e sui concetti riportati in Tutorial: creazione di una REST API con un'integrazione non proxy Lambda. Se la procedura dettagliata non è ancora stata completata, è consigliabile eseguirla ora prima di proseguire.

Argomenti

- Prerequisiti
- Fase 1: creare il ruolo di esecuzione del proxy del AWS servizio
- Fase 2: creazione della risorsa
- Fase 3: creazione del metodo GET
- Fase 4: specifica delle impostazioni del metodo e test del metodo
- Fase 5: distribuzione dell'API
- Fase 6: test dell'API
- Fase 7: pulire

Prerequisiti

Prima di iniziare questa procedura guidata, esegui queste operazioni:

1. Completa le fasi descritte in Configurazione per l'utilizzo di Gateway API.
2. Crea una nuova API denominata `MyDemoAPI`. Per ulteriori informazioni, consulta Tutorial: creazione di una REST API con un'integrazione non proxy HTTP.
3. Distribuisci l'API almeno una volta in una fase denominata `test`. Per ulteriori informazioni, consulta la pagina relativa alla distribuzione dell'API in Scegli un tutorial di integrazione AWS Lambda.
4. Completa le altre fasi descritte in Scegli un tutorial di integrazione AWS Lambda.
5. Crea almeno un argomento in Amazon Simple Notification Service (Amazon SNS). Utilizzerai l'API distribuita per ottenere un elenco di argomenti in Amazon SNS associati AWS al tuo account. Per informazioni su come creare un argomento in Amazon SNS, consulta Creazione di un argomento. Non devi copiare l'ARN dell'argomento citato nella fase 5.

Fase 1: creare il ruolo di esecuzione del proxy del AWS servizio

Per permettere all'API di richiamare le operazioni Amazon SNS necessarie, devi aver collegato le policy IAM appropriate a un ruolo IAM. In questo passaggio si crea un nuovo ruolo IAM.

Per creare il ruolo di esecuzione del proxy di AWS servizio

1. Accedi Console di gestione AWS e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Scegliere Roles (Ruoli).

3. Scegli Crea ruolo.
4. Scegli il AWS servizio in Seleziona il tipo di entità affidabile, quindi seleziona API Gateway e seleziona Consenti a API Gateway di inviare i log ai CloudWatch log.
5. Scegli Successivo e di nuovo Successivo.
6. In Role name (Nome ruolo) immettere **`APIGatewaySNSProxyPolicy`** e quindi selezionare Create role (Crea ruolo).
7. Nell'elenco Roles (Ruoli) scegliere il ruolo appena creato. Potrebbe essere necessario scorrere o utilizzare la barra di ricerca per trovare il ruolo.
8. Per il ruolo selezionato, seleziona la scheda Aggiungi autorizzazioni.
9. Dall'elenco a discesa scegli Collega policy.
10. Nella barra di ricerca inserisci **`AmazonSNSReadOnlyAccess`** e scegli Aggiungi autorizzazioni.

> **Note**
>
> Per semplicità, questo tutorial utilizza una policy gestita. Come best practice, dovresti creare la tua policy IAM per concedere le autorizzazioni minime richieste.

11. Annota l'ARN del ruolo appena creato, lo utilizzerai in seguito.

Fase 2: creazione della risorsa

In questo passaggio, crei una risorsa che consente al proxy del AWS servizio di interagire con il AWS servizio.

Per creare la risorsa

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere l'API.
3. Seleziona la risorsa root /, rappresentata da una singola barra (/), quindi scegli Crea risorsa.
4. Mantieni l'opzione Risorsa proxy disattivata.
5. Mantieni Percorso risorsa impostato su `/`.
6. Per Resource Name (Nome risorsa) immetti **`mydemoawsproxy`**.
7. Mantieni CORS (Cross Origin Resource Sharing) disattivato.
8. Scegli Crea risorsa.

Fase 3: creazione del metodo GET

In questo passaggio, si crea un metodo GET che consente al proxy del AWS servizio di interagire con il servizio. AWS

Per creare il metodo **GET**

1. Seleziona la risorsa /mydemoawsproxy, quindi scegli Crea metodo.
2. Per Tipo di metodo seleziona GET.
3. Per Tipo di integrazione seleziona Servizio AWS.
4. Per Regione AWS, seleziona l'argomento Regione AWS in cui hai creato il tuo Amazon SNS.
5. Per Servizio AWS seleziona Amazon SNS.
6. Lascia vuoto Sottodominio AWS .
7. Per Metodo HTTP seleziona GET.
8. Per Tipo di operazione scegli Usa nome operazione.
9. Per Nome azione immetti **`ListTopics`**.
10. Per Ruolo di esecuzione immetti l'ARN del ruolo per **`APIGatewaySNSProxyPolicy`**.
11. Scegli Crea metodo.

Fase 4: specifica delle impostazioni del metodo e test del metodo

A questo punto puoi testare il metodo `GET` per verificare che sia stato configurato correttamente per elencare gli argomenti Amazon SNS.

Test del metodo **GET**

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. Scegli Test (Esegui test).

   Il risultato mostra una risposta simile a quella riportata di seguito:

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

Fase 5: distribuzione dell'API

In questa fase, viene distribuita l'API per poterla chiamare esternamente alla console API Gateway.

Per distribuire l'API

1. Seleziona Deploy API (Distribuisci API).
2. In Fase, seleziona Nuova fase.
3. In Stage name (Nome fase) immettere **`test`**.
4. (Facoltativo) In Descrizione, immetti una descrizione.
5. Selezionare Deploy (Distribuisci).

Fase 6: test dell'API

In questa fase, esci dalla console API Gateway e utilizzi il proxy del AWS servizio per interagire con il servizio Amazon SNS.

1. Nel pannello di navigazione principale scegli Fase.
2. In Dettagli fase, scegli l'icona Copia per copiare l'URL di richiamo dell'API.

   L'URL dovrebbe essere simile a questo:

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

3. Immetti l'URL nella barra degli indirizzi di una nuova scheda del browser.
4. Aggiungi `/mydemoawsproxy` in modo che l'URL sia simile al seguente:

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

   Accedi all'URL. Verranno visualizzate informazioni simili alle seguenti:

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

Fase 7: pulire

Puoi eliminare le risorse IAM necessarie al proxy del AWS servizio per funzionare.

> **Warning**
>
> Se elimini una risorsa IAM su cui si basa un proxy di AWS servizio, quel proxy di AWS servizio e tutti quelli APIs che si basano su di essa non funzioneranno più. L'eliminazione di una risorsa IAM non può essere annullata. Per usare di nuovo la risorsa IAM, è necessario ricrearla.

Per eliminare le risorse IAM associate

1. Aprire la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Nell'area Details (Dettagli) scegliere Roles (Ruoli).
3. Selezionare APIGatewayAWSProxyExecRole e quindi scegliere Role Actions (Operazioni ruolo), Delete Role (Elimina ruolo). Quando viene richiesto, scegliere Yes, Delete (Sì, elimina).
4. Nell'area Details (Dettagli) scegliere Policies (Policy).
5. Selezionare APIGatewayAWSProxyExecPolicy e quindi scegliere Policy actions (Operazioni policy), Delete (Elimina). Quando viene richiesto, scegliere Delete (Elimina).

Hai completato questa procedura guidata. Per discussioni più dettagliate sulla creazione di API come proxy di AWS servizio, consulta Tutorial: creazione di una REST API come un proxy Amazon S3Tutorial: crea un'API REST per calcolatrice con due AWS integrazioni di servizi e un'integrazione non proxy Lambda, oTutorial: creazione di un'API REST come un proxy Amazon Kinesis.

Tutorial: crea un'API REST per calcolatrice con due AWS integrazioni di servizi e un'integrazione non proxy Lambda

the section called “Tutorial: creazione di una REST API con un'integrazione non proxy Lambda” utilizza esclusivamente l'integrazione `Lambda Function`. L'integrazione `Lambda Function` è un caso speciale del tipo di integrazione `AWS Service` in cui gran parte delle operazioni di configurazione dell'integrazione vengono eseguite automaticamente, ad esempio l'aggiunta automatica delle autorizzazioni basate su risorse necessarie per invocare la funzione Lambda. In questo caso, due integrazioni su tre utilizzano l'integrazione `AWS Service`. Questo tipo di integrazione offre un controllo maggiore, ma richiede l'esecuzione manuale di attività come la creazione e la definizione di un ruolo IAM contenente le autorizzazioni appropriate.

In questo tutorial, creerai una funzione `Calc` Lambda che implementa operazioni aritmetiche di base, accettando e restituendo input e output. JSON-formatted Verrà quindi creata un'API REST, che verrà integrata con la funzione Lambda nei modi seguenti:

1. Esponendo un metodo `GET` nella risorsa `/calc` per richiamare la funzione Lambda, fornendo l'input sotto forma di parametri delle stringhe di query (integrazione `AWS Service`).
2. Esponendo un metodo `POST` nella risorsa `/calc` per richiamare la funzione Lambda, fornendo l'input nel payload della richiesta del metodo (integrazione `AWS Service`).
3. Esponendo un metodo `GET` nelle risorse `/calc/{operand1}/{operand2}/{operator}` nidificate per richiamare la funzione Lambda, fornendo l'input sotto forma di parametri di percorso (integrazione `Lambda Function`).

Oltre a provare questo tutorial, puoi esaminare il file di definizione OpenAPI per l'API `Calc`, che è possibile importare in API Gateway seguendo le istruzioni in the section called “OpenAPI”.

Argomenti

- Creazione di un ruolo IAM prevedibile
- Creare una funzione Calc Lambda
- Prova la funzione Calc Lambda

- Crea un'API Calc
- Integrazione 1: creazione di un metodo GET con parametri di query per chiamare la funzione Lambda
- Integrazione 2: creazione di un metodo POST con un payload JSON per chiamare la funzione Lambda
- Integrazione 3: creazione di un metodo GET con parametri di percorso per chiamare la funzione Lambda
- Definizioni OpenAPI di un'API di esempio integrata con una funzione Lambda

Creazione di un ruolo IAM prevedibile

Affinché l'API possa richiamare la funzione Lambda `Calc`, è necessario un ruolo IAM API Gateway prevedibile, ovvero un ruolo IAM con la relazione di trust seguente:

JSON

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

Il ruolo che crei dovrà avere l'autorizzazione Lambda InvokeFunction. In caso contrario, il chiamante dell'API riceverà una risposta `500 Internal Server Error`. Per assegnare al ruolo questa autorizzazione, è necessario collegare al ruolo la policy IAM seguente:

JSON

```
{
```

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

Ecco come procedere:

Creazione di un ruolo IAM API Gateway prevedibile

1. Accedi alla console IAM.
2. Scegliere Roles (Ruoli).
3. Seleziona Create Role (Crea ruolo).
4. In Select type of trusted entity (Seleziona tipo di entità attendibile), scegli AWS Service.
5. In Choose the service that will use this role (Scegli il servizio che utilizzerà questo ruolo), seleziona Lambda.
6. Scegli Next: Permissions (Successivo: Autorizzazioni).
7. Selezionare Create Policy (Crea policy).

   Si aprirà una nuova finestra Create Policy (Crea policy) della console. Nella finestra procedi nel seguente modo:

   a. Nella scheda JSON, sostituisci la policy esistente con la seguente:

      JSON

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

```
        }
    ]
}
```

b. Scegliere Review policy (Esamina policy).

c. InReview Policy (Rivedi policy), effettua le operazioni seguenti:

i. In Name (Nome), digita un nome, ad esempio **`lambda_execute`**.

ii. Selezionare Create Policy (Crea policy).

8. Nella finestra Create Role (Crea ruolo) originaria della console, procedi nel seguente modo:

a. In Attach permissions policies (Collega policy delle autorizzazioni), scegli la tua policy **`lambda_execute`** dall'elenco a discesa.

Se la policy non è visualizzata nell'elenco, scegliere il pulsante di aggiornamento in alto. Non aggiornare la pagina del browser.

b. Scegli Next:Tags.

c. Scegli Next:Review.

d. Per Role name (Nome ruolo), digita un nome, ad esempio **`lambda_invoke_function_assume_apigw_role`**.

e. Seleziona Create role (Crea ruolo).

9. Seleziona il tuo **`lambda_invoke_function_assume_apigw_role`** nell'elenco dei ruoli.

10. Seleziona la scheda Trust relationships (Relazioni di trust).

11. Seleziona Edit trust relationship (Modifica relazione di trust).

12. Sostituisci la policy esistente con la seguente:

JSON

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

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

13. Selezionare Update Trust Policy (Aggiorna policy di trust).
14. Prendi nota dell'ARN per il ruolo appena creato. Sarà necessario in seguito.

Creare una funzione **`Calc Lambda`**

Verrà quindi creata una funzione Lambda utilizzando la console Lambda.

1. Nella console Lambda, scegliere Create function (Crea funzione).
2. Scegli Author from Scratch (Crea da zero).
3. In Name (Nome), immettere **`Calc`**.
4. Per Runtime, scegli l'ultimo runtime supportato Node.jso Python.
5. Utilizza le impostazioni predefinite per tutte le altre opzioni.
6. Scegli Crea funzione.
7. Copia la seguente funzione Lambda e incollala nel runtime preferito e nell'editor di codice nella console Lambda.

   Node.js

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

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

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

  return res;
};
```

Python

```
import json


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

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

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

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

    return res
```

8. In Execution role (Ruolo di esecuzione) scegliere Choose an existing role (Scegli un ruolo esistente).
9. Immettere l'ARN per il ruolo **`lambda_invoke_function_assume_apigw_role`** creato in precedenza.
10. Selezionare Deploy (Distribuisci).

Questa funzione richiede due operandi (`a` e `b`) e un operatore (`op`) del parametro di input `event`. L'input è un oggetto JSON con il formato seguente:

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

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

Questa funzione restituisce il risultato calcolato (`c`) e l'input. Per un input non valido, la funzione restituisce il valore null o la stringa "Invalid op" come risultato. L'output ha il formato JSON seguente:

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

Prima dell'integrazione con l'API nella fase successiva, è necessario testare la funzione nella console Lambda.

Prova la funzione **`Calc Lambda`**

Di seguito viene descritto come testare la funzione `Calc` nella console Lambda:

1. Seleziona la scheda Test.
2. Come nome dell'evento di test, immettere **`calc2plus5`**.
3. Sostituire la definizione dell'evento di test con quanto segue:

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

4. Scegli Save (Salva).
5. Scegli Test (Esegui test).
6. Espandere Execution result: succeeded (Risultato esecuzione: riuscita). Verrà visualizzato un codice analogo al seguente:

   ```
   {
   ```

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

Crea un'API Calc

Nella procedura seguente viene illustrato come creare un'API per la funzione Lambda `Calc` appena creata. Nelle sezioni successive verranno aggiunti metodi e risorse.

Come creare un'API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Se si utilizza API Gateway per la prima volta, verrà visualizzata una pagina che presenta le caratteristiche del servizio. In API REST, scegliere Crea. Quando appare il popup Create Example API (Crea API di esempio), scegliere OK.

   Se non è la prima volta che si utilizza API Gateway, scegliere Create API (Crea API). In API REST, scegliere Crea.
3. Per API name (Nome API), immettere **`LambdaCalc`**.
4. (Facoltativo) In Descrizione, immetti una descrizione.
5. Lasciare l'opzione Tipo di endpoint API impostata su Regionale.
6. Per Tipo di indirizzo IP, selezionare IPv4.
7. Seleziona Create API (Crea API).

Integrazione 1: creazione di un metodo **`GET`** con parametri di query per chiamare la funzione Lambda

Creando un metodo `GET` che passa i parametri delle stringhe di query alla funzione Lambda, si permette all'API di essere richiamata da un browser. Questo approccio può essere utile, soprattutto per le API che permettono l'accesso aperto.

Dopo l'API, è necessario creare una risorsa. In genere, le risorse API sono organizzate in una struttura di risorse in base alla logica dell'applicazione. In questa fase crei una risorsa /calc.

Per creare un /calc risorsa

1. Scegli Crea risorsa.
2. Mantieni l'opzione Risorsa proxy disattivata.
3. Mantieni Percorso risorsa impostato su `/`.
4. Per Resource Name (Nome risorsa) immetti **`calc`**.
5. Mantieni CORS (Cross Origin Resource Sharing) disattivato.
6. Scegli Crea risorsa.

Creando un metodo `GET` che passa i parametri delle stringhe di query alla funzione Lambda, si permette all'API di essere richiamata da un browser. Questo approccio può essere utile, soprattutto per le API che permettono l'accesso aperto.

Con questo metodo, Lambda richiede che venga usata la richiesta `POST` per richiamare qualsiasi funzione Lambda. Questo esempio mostra che il metodo HTTP in una richiesta del metodo di front-end può essere diverso dalla richiesta di integrazione nel back-end.

`Per creare un metodo GET`

1. Seleziona la risorsa /calc quindi scegli Crea metodo.
2. Per Tipo di metodo seleziona GET.
3. Per Tipo di integrazione seleziona Servizio AWS.
4. Per Regione AWS, seleziona il Regione AWS luogo in cui hai creato la tua funzione Lambda.
5. Per Servizio AWS seleziona Lambda.
6. Lascia vuoto Sottodominio AWS .
7. Per Metodo HTTP seleziona POST.
8. In Tipo di operazione scegli Utilizza sostituzione percorso. Questa opzione permette di specificare l'ARN dell'operazione Invoke (Richiama) per eseguire la funzione `Calc`.
9. Per Sostituzione percorso immetti **`2015-03-31/functions/arn:aws:lambda:`*`us-east-2`*`:`*`account-id`*`:function:Calc/invocations`**. Per *`account-id`*, inserisci il tuo Account AWS ID. Per*`us-east-2`*, inserisci Regione AWS dove hai creato la tua funzione Lambda.
10. Per Ruolo di esecuzione immetti l'ARN del ruolo per **`lambda_invoke_function_assume_apigw_role`**.
11. Non modificare le impostazioni di Cache delle credenziali e Timeout predefinito.

12. Scegli Impostazioni della richiesta del metodo.
13. Per Validatore richiesta seleziona Convalida parametri di stringa query e intestazioni.

    Questa impostazione restituisce un messaggio di errore se il client non specifica i parametri obbligatori.
14. Scegli Parametri della stringa di query URL.

    A questo punto si configurano i parametri della stringa di query per il metodo GET sulla risorsa / calc in modo da poter ricevere l'input per conto della funzione Lambda di backend.

    Per creare i parametri della stringa di query, procedi come indicato di seguito:

    a. Scegliere Add query string (Aggiungi stringa di query).
    b. In Nome, inserisci **`operand1`**.
    c. Attiva Campo obbligatorio.
    d. Mantieni disattivata l'opzione Caching.

    Ripeti le stesse fasi e crea una stringa di query denominata **`operand2`** e una stringa di query denominata **`operator`**.
15. Scegli Crea metodo.

A questo punto crea un modello di mappatura per tradurre le stringhe di query fornite dal client in payload di richiesta di integrazione, come richiesto dalla funzione `Calc`. Questo modello mappa i tre parametri della stringa di query dichiarati in Richiesta metodo in valori di proprietà designati dell'oggetto JSON come input per la funzione Lambda di back-end. L'oggetto JSON trasformato verrà incluso come payload di richiesta di integrazione.

Per mappare i parametri di input alla richiesta di integrazione

1. Nella scheda Richiesta di integrazione scegli Modifica in Impostazioni della richiesta di integrazione.
2. Per Richiesta corpo passthrough scegli Quando non ci sono modelli definiti (consigliato).
3. Scegli Modelli di mappatura.
4. Scegliere Add mapping template (Aggiungi modello di mappatura).
5. Per Tipo di contenuto inserisci **`application/json`**.
6. Per Corpo del modello inserisci il seguente codice:

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

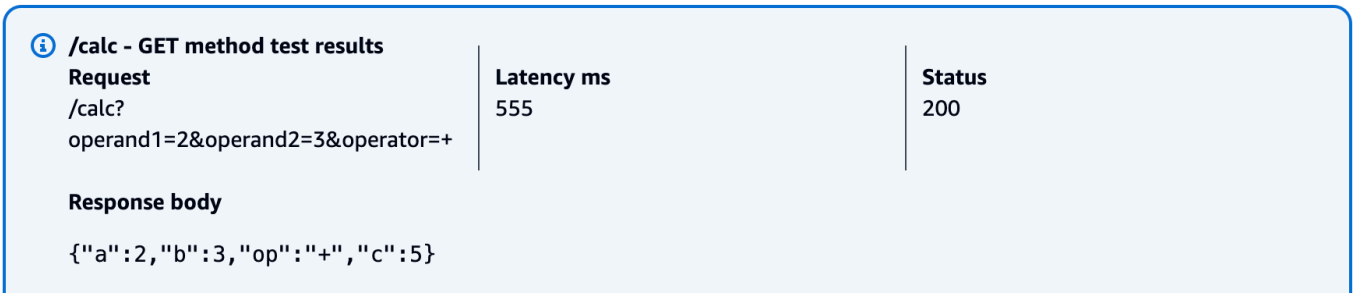
7. Scegli Save (Salva).

È ora possibile testare il metodo GET per verificare che sia stato configurato correttamente per richiamare la funzione Lambda.

Per testare il metodo **GET**

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. In Stringhe di query immetti **operand1=2&operand2=3&operator=+**.
3. Scegli Test (Esegui test).

   I risultati dovrebbero essere simili a quanto segue:



Integrazione 2: creazione di un metodo **POST** con un payload JSON per chiamare la funzione Lambda

Se si crea un metodo POST con un payload JSON per chiamare la funzione Lambda, si richiede al client di fornire l'input necessario alla funzione di back-end nel corpo della richiesta. Per garantire che il client carichi i dati di input corretti, verrà abilitata la convalida delle richieste nel payload.

Per creare un metodo **POST con un payload** JSON

1. Seleziona la risorsa /calc quindi scegli Crea metodo.
2. Per Tipo di metodo seleziona POST.

3. Per Tipo di integrazione seleziona Servizio AWS.
4. Per Regione AWS, seleziona il Regione AWS luogo in cui hai creato la tua funzione Lambda.
5. Per Servizio AWS seleziona Lambda.
6. Lascia vuoto Sottodominio AWS .
7. Per Metodo HTTP seleziona POST.
8. In Tipo di operazione scegli Utilizza sostituzione percorso. Questa opzione permette di specificare l'ARN dell'operazione Invoke (Richiama) per eseguire la funzione `Calc`.
9. Per Sostituzione percorso immetti **`2015-03-31/functions/arn:aws:lambda:`*`us-east-2`*`:`*`account-id`*`:function:Calc/invocations`**. Per *`account-id`*, inserisci il tuo Account AWS ID. Per*`us-east-2`*, inserisci Regione AWS dove hai creato la tua funzione Lambda.
10. Per Ruolo di esecuzione immetti l'ARN del ruolo per **`lambda_invoke_function_assume_apigw_role`**.
11. Non modificare le impostazioni di Cache delle credenziali e Timeout predefinito.
12. Scegli Crea metodo.

A questo punto crea un modello di input per descrivere la struttura dei dati di input e convalidare il corpo della richiesta in ingresso.

Per creare un modello di input

1. Nel riquadro di navigazione principale seleziona Modelli.
2. Scegli Crea modello.
3. In Nome, inserisci **`input`**.
4. Per Tipo di contenuto inserisci **`application/json`**.

   Se non viene trovato alcun tipo di contenuto corrispondente, la convalida della richiesta non viene eseguita. Per utilizzare lo stesso modello indipendentemente dal tipo di contenuti, inserisci **`$default`**.
5. Per Schema modello immetti il seguente modello:

```
{
    "type":"object",
    "properties":{
        "a":{"type":"number"},
```

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

6. Scegli Crea modello.

A questo punto crea un modello di output. Questo modello descrive la struttura di dati dell'output calcolato del back-end. Può essere usato per mappare i dati della risposta di integrazione a un modello diverso. Questo tutorial è basato sul comportamento di passthrough e non usa questo modello.

Per creare un modello di output

1. Scegli Crea modello.
2. In Nome, inserisci **`output`**.
3. Per Tipo di contenuto inserisci **`application/json`**.

   Se non viene trovato alcun tipo di contenuto corrispondente, la convalida della richiesta non viene eseguita. Per utilizzare lo stesso modello indipendentemente dal tipo di contenuti, inserisci **`$default`**.
4. Per Schema modello immetti il seguente modello:

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

5. Scegli Crea modello.

A questo punto crea un modello di risultato. Questo modello descrive la struttura dei dati della risposta restituiti. Fa riferimento agli schemi di input e output definiti nell'API.

Per creare un modello di risultato

1. Scegli Crea modello.

2. In Nome, inserisci **`result`**.
3. Per Tipo di contenuto inserisci **`application/json`**.

   Se non viene trovato alcun tipo di contenuto corrispondente, la convalida della richiesta non viene eseguita. Per utilizzare lo stesso modello indipendentemente dal tipo di contenuti, inserisci **`$default`**.
4. Per lo schema del modello, inserisci il seguente modello con il tuo*`restapi-id`*. Il tuo *`restapi-id`* è elencato tra parentesi nella parte superiore della console nel seguente flusso: `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. Scegli Crea modello.

A questo punto configura la richiesta del metodo POST per abilitare la convalida della richiesta nel corpo della richiesta in ingresso.

Abilitazione della convalida della richiesta del metodo POST

1. Nel pannello di navigazione scegli Risorse, quindi seleziona il metodo `POST` nella struttura di risorse.
2. Nella scheda Richiesta metodo, in Impostazioni richiesta metodo, scegli Modifica.
3. Per Validatore richiesta seleziona Convalida corpo.
4. Scegli Corpo della richiesta, quindi seleziona Aggiungi modello.
5. Per Tipo di contenuto inserisci **`application/json`**.

Se non viene trovato alcun tipo di contenuto corrispondente, la convalida della richiesta non viene eseguita. Per utilizzare lo stesso modello indipendentemente dal tipo di contenuti, inserisci **`$default`**.

6. Per Modello seleziona input.
7. Scegli Save (Salva).

È ora possibile testare il metodo `POST` per verificare che sia stato configurato correttamente per richiamare la funzione Lambda.

Per testare il metodo POST

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. Per Corpo della richiesta immetti il payload JSON seguente.

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

3. Scegli Test (Esegui test).

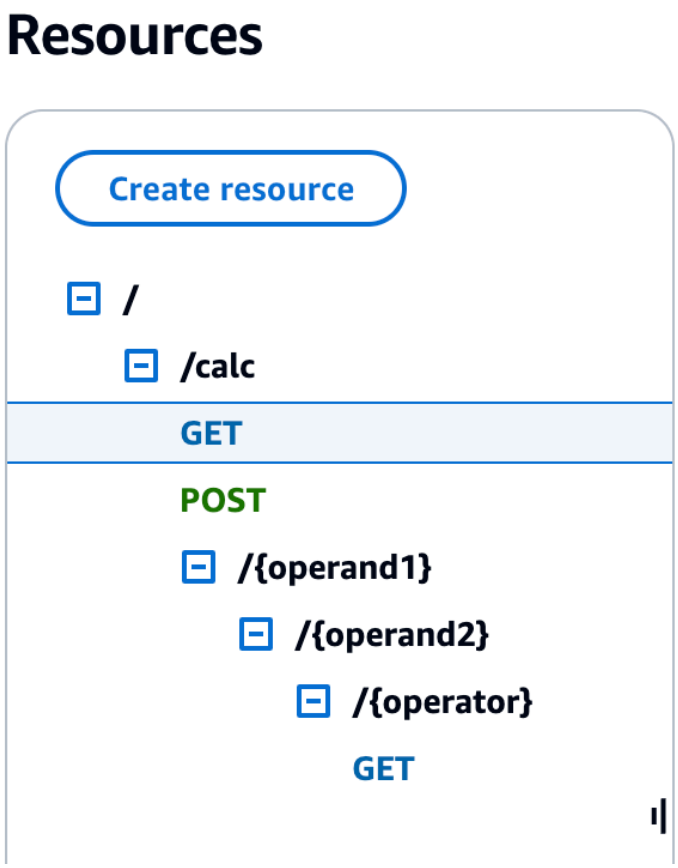
   Verrà visualizzato l'output seguente:

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

Integrazione 3: creazione di un metodo **GET** con parametri di percorso per chiamare la funzione Lambda

Verrà ora creato un metodo GET in una risorsa specificata da una sequenza di parametri di percorso per chiamare la funzione Lambda di back-end. I valori dei parametri di percorso specificano i dati di input per la funzione Lambda. Verrà usato un modello di mappatura per mappare i valori dei parametri di percorso in ingresso al payload di richiesta di integrazione necessario.

La struttura della risorsa API risultante sarà la seguente:



Per creare un /{operand1}/{operand2}/{operatore} risorsa

1. Scegli Crea risorsa.
2. Per Percorso risorsa seleziona `/calc`.
3. Per Resource Name (Nome risorsa) immetti **`{operand1}`**.
4. Mantieni CORS (Cross Origin Resource Sharing) disattivato.
5. Scegli Crea risorsa.
6. Per Percorso risorsa seleziona `/calc/{operand1}/`.

7. Per Resource Name (Nome risorsa) immetti **`{operand2}`**.
8. Mantieni CORS (Cross Origin Resource Sharing) disattivato.
9. Scegli Crea risorsa.
10. Per Percorso risorsa seleziona `/calc/{operand1}/{operand2}/`.
11. Per Resource Name (Nome risorsa) immetti **`{operator}`**.
12. Mantieni CORS (Cross Origin Resource Sharing) disattivato.
13. Scegli Crea risorsa.

Questa volta verrà utilizzata l'integrazione Lambda predefinita nella console Gateway API per configurare l'integrazione del metodo.

Per configurare l'integrazione di un metodo

1. Seleziona la risorsa /{operand1}/{operand2}/{operator}, quindi scegli Crea metodo.
2. Per Tipo di metodo seleziona GET.
3. In Tipo di integrazione, seleziona Lambda.
4. Mantieni l'opzione Integrazione proxy Lambda disattivata.
5. Per la funzione Lambda, seleziona il Regione AWS luogo in cui hai creato la funzione Lambda e inserisci. **`Calc`**
6. Mantieni attivata l'opzione Timeout predefinito.
7. Scegli Crea metodo.

A questo punto crea il modello per mappare i tre parametri del percorso URL, dichiarati quando è stata creata la risorsa /calc/{operand1}/{operand2}/{operator}, ai valori di proprietà designati nell'oggetto JSON. Poiché i percorsi URL devono essere URL-encoded, l'operatore di divisione deve essere specificato come `%2F` anziché. `/` Questo modello traduce `%2F` in `'/'` prima di passarlo alla funzione Lambda.

Per creare un modello di mappatura

1. Nella scheda Richiesta di integrazione scegli Modifica in Impostazioni della richiesta di integrazione.
2. Per Richiesta corpo passthrough scegli Quando non ci sono modelli definiti (consigliato).
3. Scegli Modelli di mappatura.

4. Per Tipo di contenuto inserisci **`application/json`**.
5. Per Corpo del modello inserisci il seguente codice:

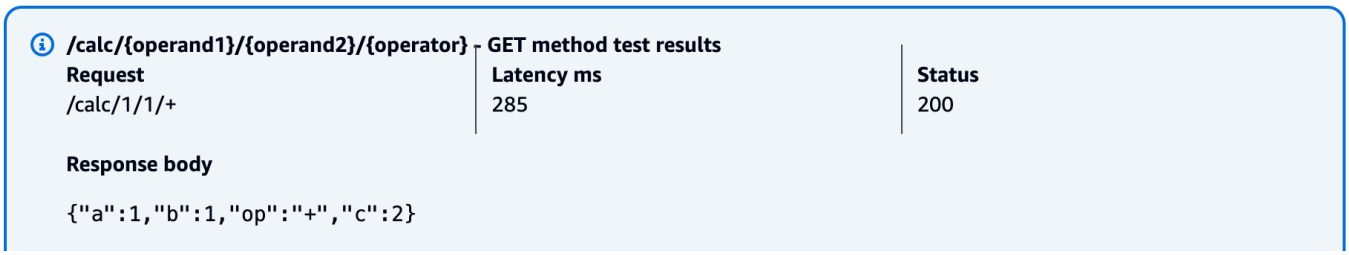
```
{
   "a": "$input.params('operand1')",
   "b": "$input.params('operand2')",
   "op":
 #if($input.params('operator')=='%2F')"/"#{else}"$input.params('operator')"#end
}
```

6. Scegli Save (Salva).

A questo punto testa il metodo `GET` per verificare che sia stato configurato correttamente per richiamare la funzione Lambda e passare l'output originale con la risposta di integrazione senza mappatura.

Per testare il metodo **`GET`**

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. Per Percorso procedi come segue:
   a. Per operand1 immetti **1**.
   b. Per operand2 immetti **1**.
   c. Per operator immetti **+**.
3. Scegli Test (Esegui test).
4. Il risultato deve essere simile al seguente:



A questo punto modella la struttura di dati del payload di risposta del metodo dopo lo schema `result`.

Per impostazione predefinita, al corpo della risposta del metodo viene assegnato un modello vuoto. Di conseguenza, il corpo della risposta di integrazione viene passato senza mappatura. Tuttavia, quando generi un SDK per uno dei linguaggi di tipo forte, come Java o Objective-C, gli utenti del tuo SDK riceveranno come risultato un oggetto vuoto. Affinché sia il client REST che i client SDK ricevano il risultato desiderato, è necessario modellare i dati della risposta utilizzando uno schema predefinito. Verrà illustrato come definire un modello per il corpo della risposta del metodo e creare un modello di mappatura per convertire il corpo della risposta di integrazione nel corpo della risposta del metodo.

Per creare una risposta del metodo

1. Nella scheda Risposta metodo scegli Modifica in Risposta 200.
2. In Corpo della risposta scegli Aggiungi modello.
3. Per Tipo di contenuto inserisci **`application/json`**.
4. Per Modello seleziona risultato.
5. Scegli Save (Salva).

L'impostazione del modello per il corpo della risposta del metodo assicura che i dati della risposta verranno trasmessi nell'oggetto `result` di un determinato SDK. Per fare in modo che i dati della risposta di integrazione siano mappati di conseguenza, è necessario un modello di mappatura.

Per creare un modello di mappatura

1. Nella scheda Risposta di integrazione scegli Modifica in Predefinito - Risposta.
2. Scegli Modelli di mappatura.
3. Per Tipo di contenuto inserisci **`application/json`**.
4. Per Corpo del modello inserisci il seguente codice:

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

```
}
```

5. Scegli Save (Salva).

Per testare il modello di mappatura

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. Per Percorso procedi come segue:
   a. Per operand1 immetti **1**.
   b. Per operand2 immetti **2**.
   c. Per operator immetti **+**.
3. Scegli Test (Esegui test).
4. Il risultato sarà simile al seguente:

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

A questo punto puoi chiamare l'API solo tramite la funzionalità Test nella console Gateway API. Per rendere l'API disponibile per i client, è necessario distribuirla. Assicurarsi sempre di ridistribuire l'API ogni volta che si aggiunge, si modifica o si elimina una risorsa o un metodo, si aggiorna una mappatura dei dati o si aggiornano le impostazioni della fase. In caso contrario, le nuove funzionalità o gli aggiornamenti non saranno disponibili per i client dell'API, come indicato di seguito:

Per distribuire l'API

1. Seleziona Deploy API (Distribuisci API).
2. In Fase, seleziona Nuova fase.

3. In Stage name (Nome fase) immettere **`Prod`**.
4. (Facoltativo) In Descrizione, immetti una descrizione.
5. Seleziona Implementa.
6. (Facoltativo) In Dettagli fase puoi scegliere l'icona Copia per copiare l'URL di richiamata dell'API in Richiama URL. È possibile usare questo valore con strumenti come Postman e cURL per testare l'API.

> **Note**
>
> Implementa nuovamente l'API ogni volta che aggiungi, modifichi o elemini una risorsa o un metodo e aggiorni una mappatura dei dati oppure le impostazioni della fase. In caso contrario, le nuove caratteristiche o gli aggiornamenti non saranno disponibili per i client dell'API.

Definizioni OpenAPI di un'API di esempio integrata con una funzione 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: creazione di una REST API come un proxy Amazon S3

Per illustrare l'uso di un'API REST in API Gateway come proxy Amazon S3, in questa sezione viene descritto come creare e configurare un'API REST per esporre le operazioni Amazon S3 seguenti:

- Esponi GET sulla risorsa radice dell'API per elencare tutti i bucket Amazon S3 di un intermediario.
- Esponi GET su una risorsa cartella per visualizzare un elenco di tutti gli oggetti di un bucket Amazon S3.
- Esporre GET su una Folder/Item risorsa per visualizzare o scaricare un oggetto da un bucket Amazon S3.

Puoi importare l'API di esempio come proxy Amazon S3, come illustrato in Definizioni OpenAPI dell'API di esempio come un proxy Amazon S3. Questo esempio include più metodi esposti. Per istruzioni su come importare l'API tramite la definizione OpenAPI, consulta Sviluppo di REST API utilizzando OpenAPI in Gateway API.

> **Note**
>
> Per integrare l'API di API Gateway con Amazon S3, devi scegliere una regione in cui i servizi API Gateway e Amazon S3 siano entrambi disponibili. Per la disponibilità delle regioni, consulta Endpoint e quote Amazon API Gateway.

Argomenti

- Configurazione delle autorizzazioni IAM per consentire all'API di invocare operazioni Amazon S3
- Creazione di risorse API per rappresentare risorse Amazon S3
- Esposizione di un metodo API per elencare i bucket Amazon S3 dell'intermediario
- Esposizione dei metodi API per accedere a un bucket Amazon S3
- Esposizione dei metodi API per accedere a un oggetto Amazon S3 in un bucket
- Definizioni OpenAPI dell'API di esempio come un proxy Amazon S3
- Chiamata dell'API mediante un client API REST

Configurazione delle autorizzazioni IAM per consentire all'API di invocare operazioni Amazon S3

Per permettere all'API di richiamare le operazioni Amazon S3, devi aver collegato le policy IAM appropriate a un ruolo IAM. In questo passaggio si crea un nuovo ruolo IAM.

Per creare AWS ruolo di esecuzione del proxy di servizio

1. Accedi Console di gestione AWS e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Scegliere Roles (Ruoli).
3. Scegliere Crea ruolo.
4. Scegli il AWS servizio in Seleziona il tipo di entità attendibile, quindi seleziona API Gateway e seleziona Consenti a API Gateway di inviare i log ai CloudWatch log.
5. Scegli Successivo e di nuovo Successivo.
6. In Role name (Nome ruolo) immettere **`APIGatewayS3ProxyPolicy`** e quindi selezionare Create role (Crea ruolo).

7. Nell'elenco Roles (Ruoli) scegliere il ruolo appena creato. Potrebbe essere necessario scorrere o utilizzare la barra di ricerca per trovare il ruolo.
8. Per il ruolo selezionato, seleziona la scheda Aggiungi autorizzazioni.
9. Dall'elenco a discesa scegli Collega policy.
10. Nella barra di ricerca inserisci **`AmazonS3FullAccess`** e scegli Aggiungi autorizzazioni.

> **Note**
>
> Per semplicità, questo tutorial utilizza una policy gestita. Come best practice, dovresti creare la tua policy IAM per concedere le autorizzazioni minime richieste.

11. Annota l'ARN del ruolo appena creato, lo utilizzerai in seguito.

Creazione di risorse API per rappresentare risorse Amazon S3

Si utilizza la risorsa radice dell'API (`/`) come container dei bucket Amazon S3 di un chiamante autenticato. Inoltre, si dovranno creare le risorse `Folder` e `Item` per rappresentare rispettivamente un bucket Amazon S3 e un oggetto Amazon S3 specifico. Il nome della cartella e la chiave dell'oggetto verranno specificati dall'intermediario nell'ambito di un URL di richiesta nel formato dei parametri di percorso.

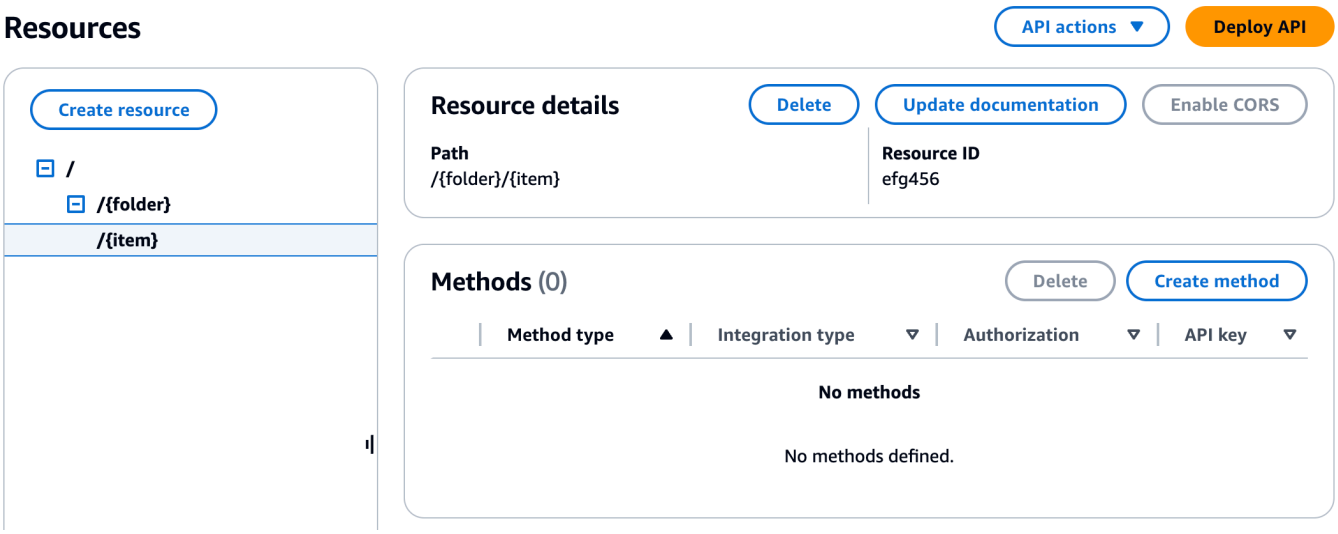
> **Note**
>
> Quando accedi a oggetti la cui chiave include `/` o qualsiasi altro carattere speciale, il carattere deve avere la codifica URL. Ad esempio, `test/test.txt` dovrebbe avere la codifica `test%2Ftest.txt`.

Per creare una risorsa API che espone le caratteristiche del servizio Amazon S3

1. Nello stesso momento in Regione AWS cui hai creato il bucket Amazon S3, crea un'API denominata mys3. Questa risorsa della radice dell'API (/) rappresenta il servizio Amazon S3. In questa fase crei le due risorse aggiuntive /{folder} e /{item}.
2. Scegli Crea risorsa.
3. Mantieni l'opzione Risorsa proxy disattivata.
4. Per Percorso risorsa seleziona `/`.

5. Per Resource Name (Nome risorsa) immetti **`{folder}`**.
6. Mantieni CORS (Cross Origin Resource Sharing) deselezionato.
7. Scegli Crea risorsa.
8. Seleziona la risorsa /{folder}, quindi scegli Crea risorsa.
9. Ripeti le fasi precedenti per creare una risorsa figlio /{folder} denominata **`{item}`**.

   L'API finale è simile a quanto riportato di seguito:



Esposizione di un metodo API per elencare i bucket Amazon S3 dell'intermediario

Per ottenere l'elenco dei bucket Amazon S3 dell'intermediario, è necessario richiamare l'operazione GET Service in Amazon S3. Nella risorsa radice dell'API, (/), crea il metodo GET. Configura il metodo GET per l'integrazione con Amazon S3, come indicato di seguito.

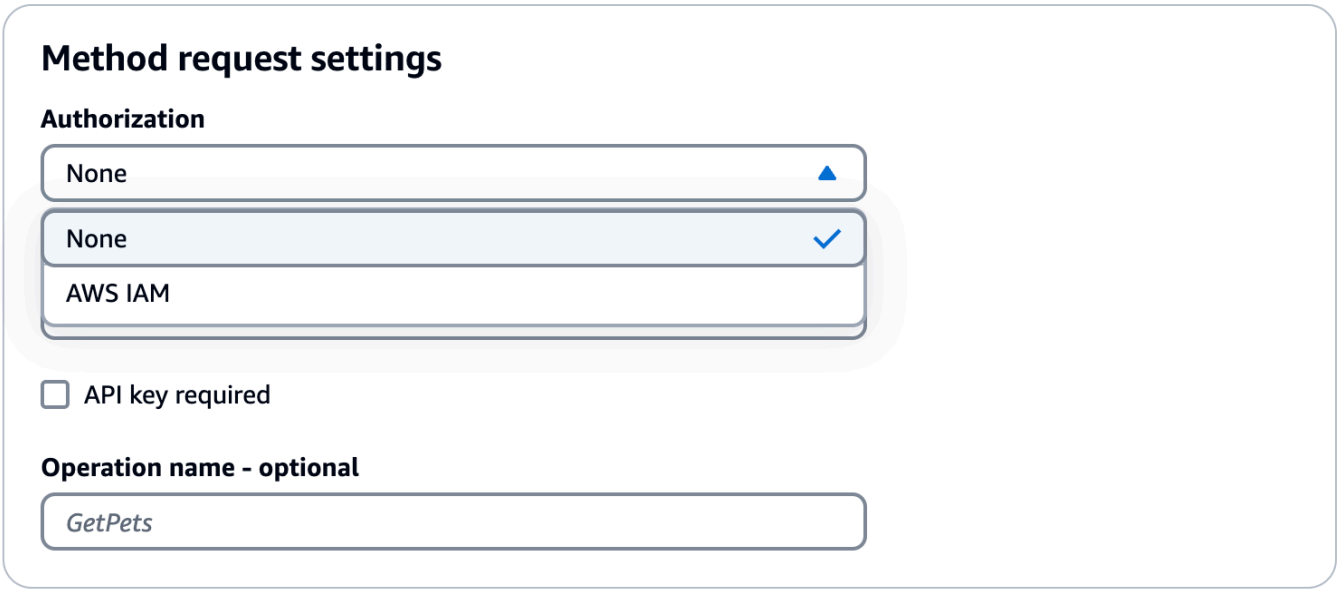
Per creare e inizializzare il metodo GET/dell'API

1. Seleziona la risorsa /, quindi scegli Crea metodo.
2. Per Tipo di metodo seleziona GET.
3. Per Tipo di integrazione seleziona Servizio AWS.
4. Per Regione AWS, seleziona il Regione AWS luogo in cui hai creato il bucket Amazon S3.
5. Per Servizio AWS seleziona Amazon Simple Storage Service.
6. Lascia vuoto Sottodominio AWS .
7. Per Metodo HTTP seleziona GET.

8. In Tipo di operazione scegli Utilizza sostituzione percorso.

   Con la sostituzione del percorso, API Gateway invia la richiesta client ad Amazon S3 come richiesta in stile percorso dell'API REST Amazon S3, in cui la risorsa Amazon S3 viene espressa dal percorso della risorsa del modello `s3-host-name/bucket/key`. API Gateway imposta il valore `s3-host-name` e passa i valori `bucket` e `key` specificati dal client ad Amazon S3.
9. In Sostituzione percorso immetti /.
10. Per Ruolo di esecuzione immetti l'ARN del ruolo per **`APIGatewayS3ProxyPolicy`**.
11. Scegli Impostazioni della richiesta del metodo.

    Si utilizzano le impostazioni della richiesta del metodo per controllare chi può chiamare questo metodo dell'API.
12. Per Autorizzazioni seleziona `AWS_IAM` nel menu a discesa.



13. Scegli Crea metodo.

Questa configurazione integra la richiesta `GET https://`*`your-api-host`*`/`*`stage`*`/` front-end con il back-end `GET https://`*`your-s3-host`*`/`.

Per consentire all'API di restituire correttamente risposte di esito positivo ed eccezioni al chiamante, si dichiarano le risposte 200, 400 e 500 in Risposta metodo. Si utilizza la mappatura predefinita per le risposte 200 in modo che le risposte di backend del codice di stato non dichiarate in questa sede vengano restituite al chiamante come risposte 200.

Per dichiarare i tipi di risposta per il **`metodo GET/`**

1. Nella scheda Risposta metodo scegli Modifica in Risposta 200.
2. Scegli Aggiungi intestazione e procedi come descritto di seguito:

   a. Per Nome intestazione immetti **`Content-Type`**.
   b. Seleziona Add header (Aggiungi intestazione).

   Ripeti queste fasi per creare un'intestazione **`Timestamp`** e un'intestazione **`Content-Length`**.
3. Selezionare Salva.
4. Nella scheda Risposta metodo scegli Crea risposta in Risposte del metodo.
5. Per Codice di stato HTTP immetti 400.

   Non impostare alcuna intestazione per questa risposta.
6. Selezionare Salva.
7. Ripeti le seguenti fasi per creare la risposta 500.

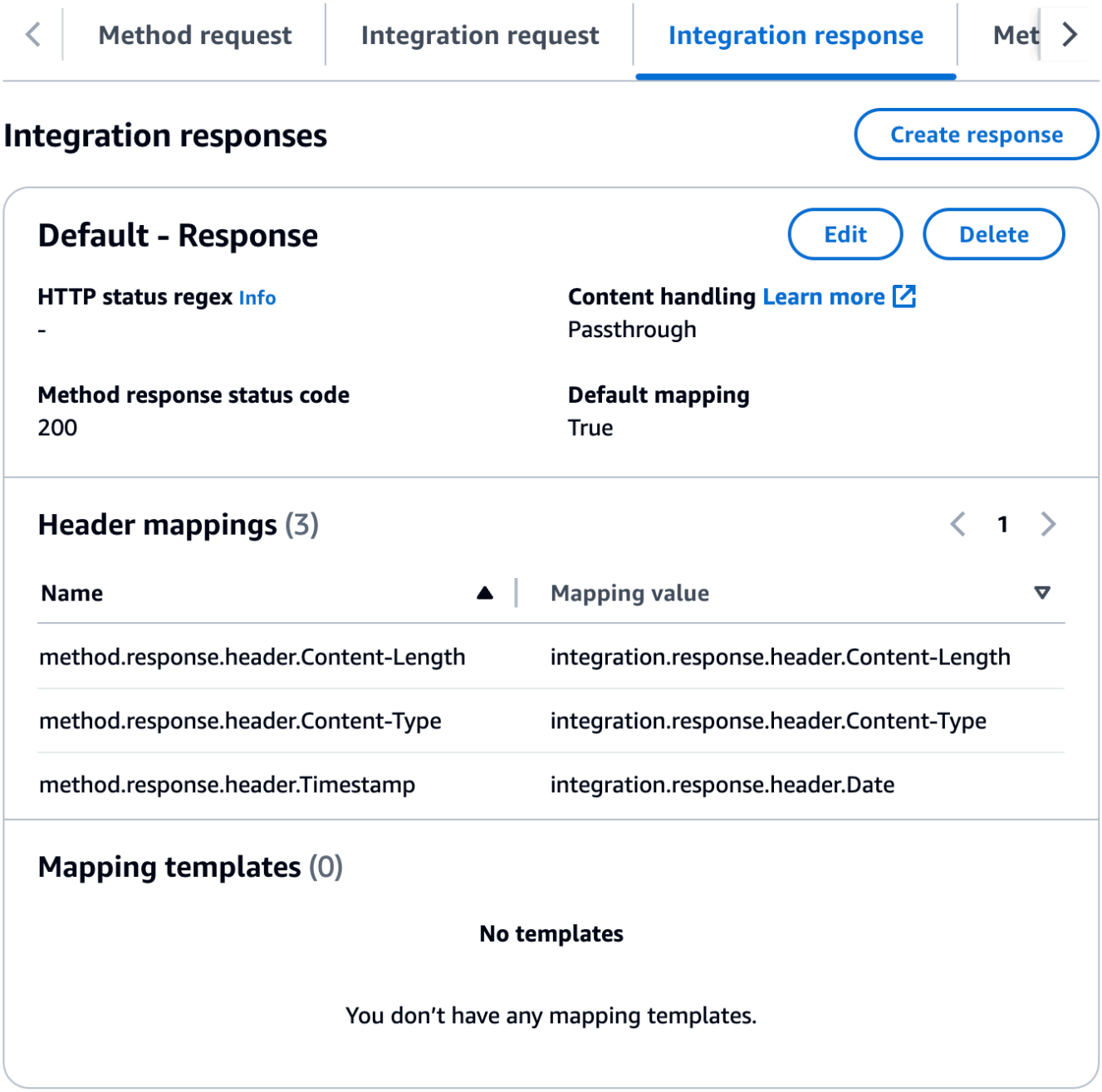
   Non impostare alcuna intestazione per questa risposta.

Poiché la risposta di integrazione corretta di Amazon S3 restituisce la bucket list come payload XML e la risposta del metodo predefinito di API Gateway restituisce un payload JSON, è necessario mappare il valore del parametro dell'intestazione del backend alla Content-Type controparte frontend. In caso contrario, il client riceverà `application/json` come tipo di contenuto quando il corpo della risposta è effettivamente una stringa XML. La procedura seguente descrive come eseguire questa configurazione. Inoltre, vuoi mostrare al cliente anche altri parametri di intestazione, come Date e. Content-Length

Per configurare le mappature delle intestazioni di risposta per il metodo GET /.

1. Nella scheda Risposta di integrazione scegli Modifica in Predefinito - Risposta.
2. Per l'Content-Lengthintestazione, immettete il valore **`integration.response.header.Content-Length`** di mappatura.
3. Per l'Content-Typeintestazione, immettete il valore di **`integration.response.header.Content-Type`** mappatura.
4. Per l'intestazione Timestamp immetti **`integration.response.header.Date`** come valore di mappatura.

5. Selezionare Salva. Il risultato sarà essere simile al seguente:



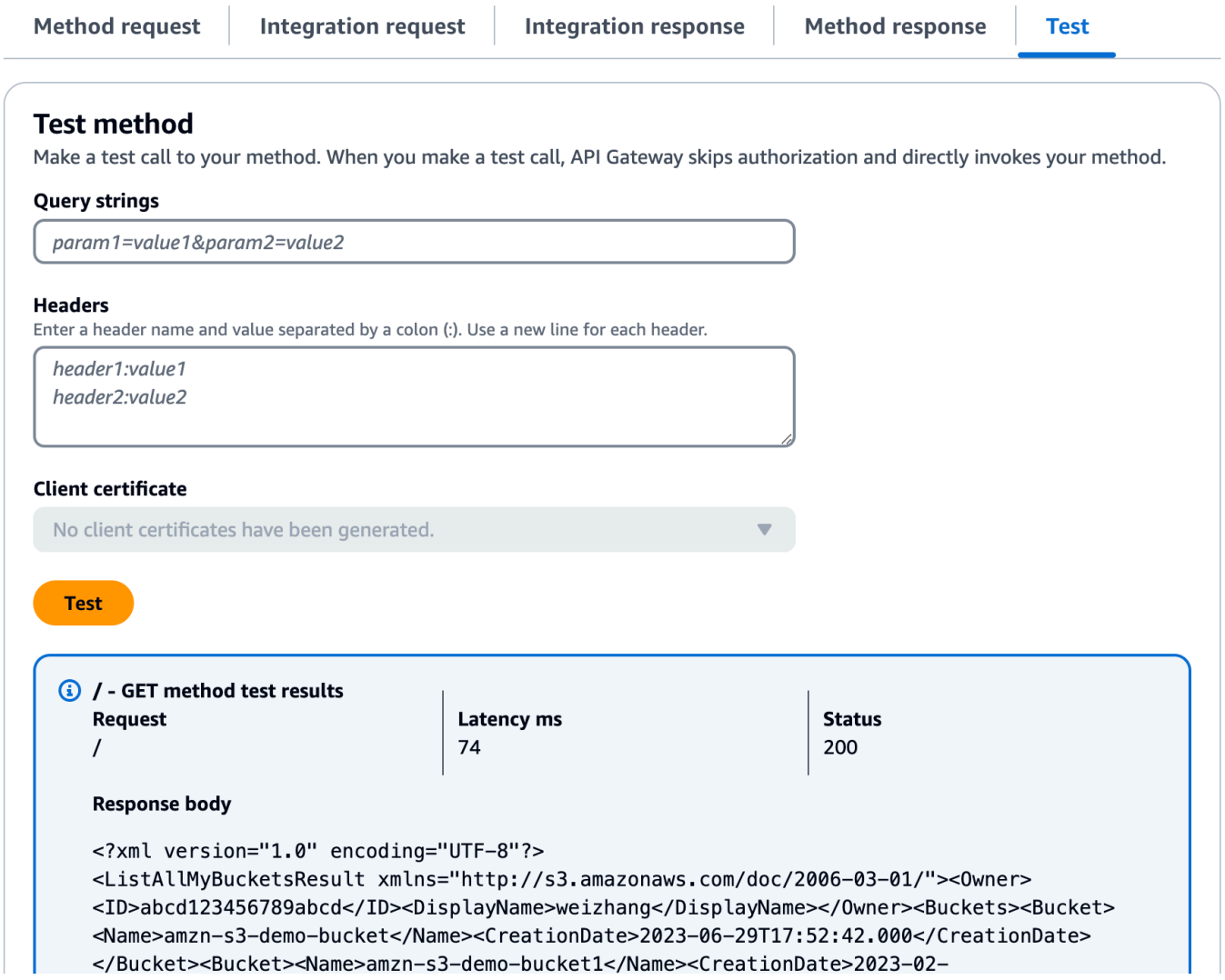
6. Nella scheda Risposta di integrazione scegli Crea risposta in Risposte di integrazione.
7. Per HTTP status regex (Regex stato HTTP), immettere **`4\d{2}`**. Tutti i codici di stato della risposta HTTP 4xx sono così mappati alla risposta del metodo.
8. Per Codice di stato risposta metodo seleziona **`400`**.
9. Scegli Crea.

10. Ripeti le seguenti fasi per creare una risposta di integrazione per la risposta del metodo 500. Per HTTP status regex (Regex stato HTTP), immettere **`5\d{2}`**.

Come suggerisce la buona norma, è opportuno testare l'API configurata fino a questo momento.

Per testare il metodo **`GET/`**

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. Scegli Test (Esegui test). Il risultato sarà simile all'immagine seguente:



Esposizione dei metodi API per accedere a un bucket Amazon S3

Per utilizzare un bucket Amazon S3, esponi il metodo `GET` nella risorsa /{folder} per elencare gli oggetti del bucket. Le istruzioni sono simili a quelle descritte in Esposizione di un metodo API per elencare i bucket Amazon S3 dell'intermediario. Per altri metodi, puoi importare l'API di esempio come descritto in Definizioni OpenAPI dell'API di esempio come un proxy Amazon S3.

Per esporre il metodo GET in una risorsa folder

1. Seleziona la risorsa /{folder}, quindi scegli Crea metodo.
2. Per Tipo di metodo seleziona GET.
3. Per Tipo di integrazione seleziona Servizio AWS.
4. Per Regione AWS, seleziona il Regione AWS luogo in cui hai creato il bucket Amazon S3.
5. Per Servizio AWS seleziona Amazon Simple Storage Service.
6. Lascia vuoto Sottodominio AWS .
7. Per Metodo HTTP seleziona GET.
8. In Tipo di operazione scegli Utilizza sostituzione percorso.
9. Per Sostituzione percorso immetti **`{bucket}`**.
10. Per Ruolo di esecuzione immetti l'ARN del ruolo per **`APIGatewayS3ProxyPolicy`**.
11. Scegli Crea metodo.

Imposta il parametro di percorso `{folder}` nell'URL dell'endpoint Amazon S3. Dovrai mappare il parametro di percorso `{folder}` della richiesta di metodo al parametro di percorso `{bucket}` della richiesta di integrazione.

Per mappare **`{folder}`** a **`{bucket}`**

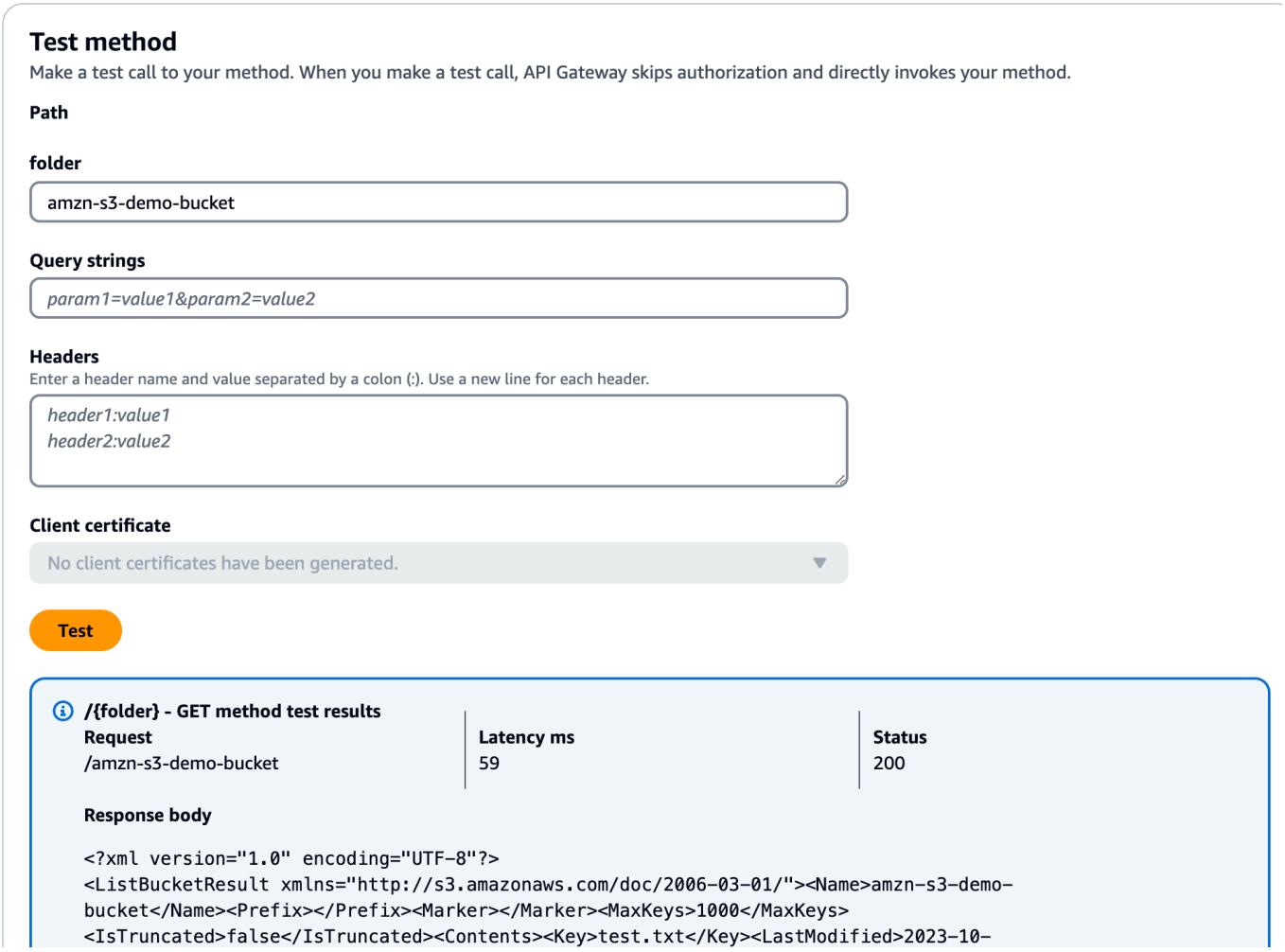
1. Nella scheda Richiesta di integrazione scegli Modifica in Impostazioni della richiesta di integrazione.
2. Seleziona Parametri di percorso URL, quindi scegli Aggiungi parametro di percorso.
3. In Nome, inserisci **`bucket`**.
4. In Mappato da, inserire **`method.request.path.folder`**.
5. Selezionare Salva.

A questo punto esegui il test dell'API.

Per testare il **`metodo`**/{folder} GET.

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. In Percorso immetti il nome del tuo bucket per folder.
3. Scegli Test (Esegui test).

   Il risultato del test conterrà l'elenco degli oggetti presenti nel bucket.



Esposizione dei metodi API per accedere a un oggetto Amazon S3 in un bucket

Amazon S3 supporta le operazioni GET, DELETE, HEAD, OPTIONS, POST e PUT per accedere agli oggetti e gestirli in un bucket specifico. In questo tutorial, esponi un metodo GET nella risorsa `{folder}/{item}` per ottenere un'immagine da un bucket. Per ulteriori applicazioni della risorsa

`{folder}/{item}`, consulta l'API di esempio in Definizioni OpenAPI dell'API di esempio come un proxy Amazon S3.

Per esporre il metodo GET in un elemento

1. Seleziona la risorsa /{item}, quindi scegli Crea metodo.
2. Per Tipo di metodo seleziona GET.
3. Per Tipo di integrazione seleziona Servizio AWS.
4. Per Regione AWS, seleziona il Regione AWS luogo in cui hai creato il bucket Amazon S3.
5. Per Servizio AWS seleziona Amazon Simple Storage Service.
6. Lascia vuoto Sottodominio AWS .
7. Per Metodo HTTP seleziona GET.
8. In Tipo di operazione scegli Utilizza sostituzione percorso.
9. Per Sostituzione percorso immetti {bucket}/{object}.
10. Per Ruolo di esecuzione immetti l'ARN del ruolo per **`APIGatewayS3ProxyPolicy`**.
11. Scegli Crea metodo.

Imposta i parametri di percorso `{folder}` e `{item}` nell'URL dell'endpoint Amazon S3. Dovrai mappare il parametro di percorso della richiesta di metodo al parametro di percorso della richiesta di integrazione.

In questa fase si effettuano le operazioni seguenti:

- Mappa il parametro di percorso `{folder}` della richiesta di metodo al parametro di percorso `{bucket}` della richiesta di integrazione.
- Mappa il parametro di percorso `{item}` della richiesta di metodo al parametro di percorso `{object}` della richiesta di integrazione.

Per mappare **`{folder}`** a **`{bucket}`** e **`{item}`** a **`{object}`**

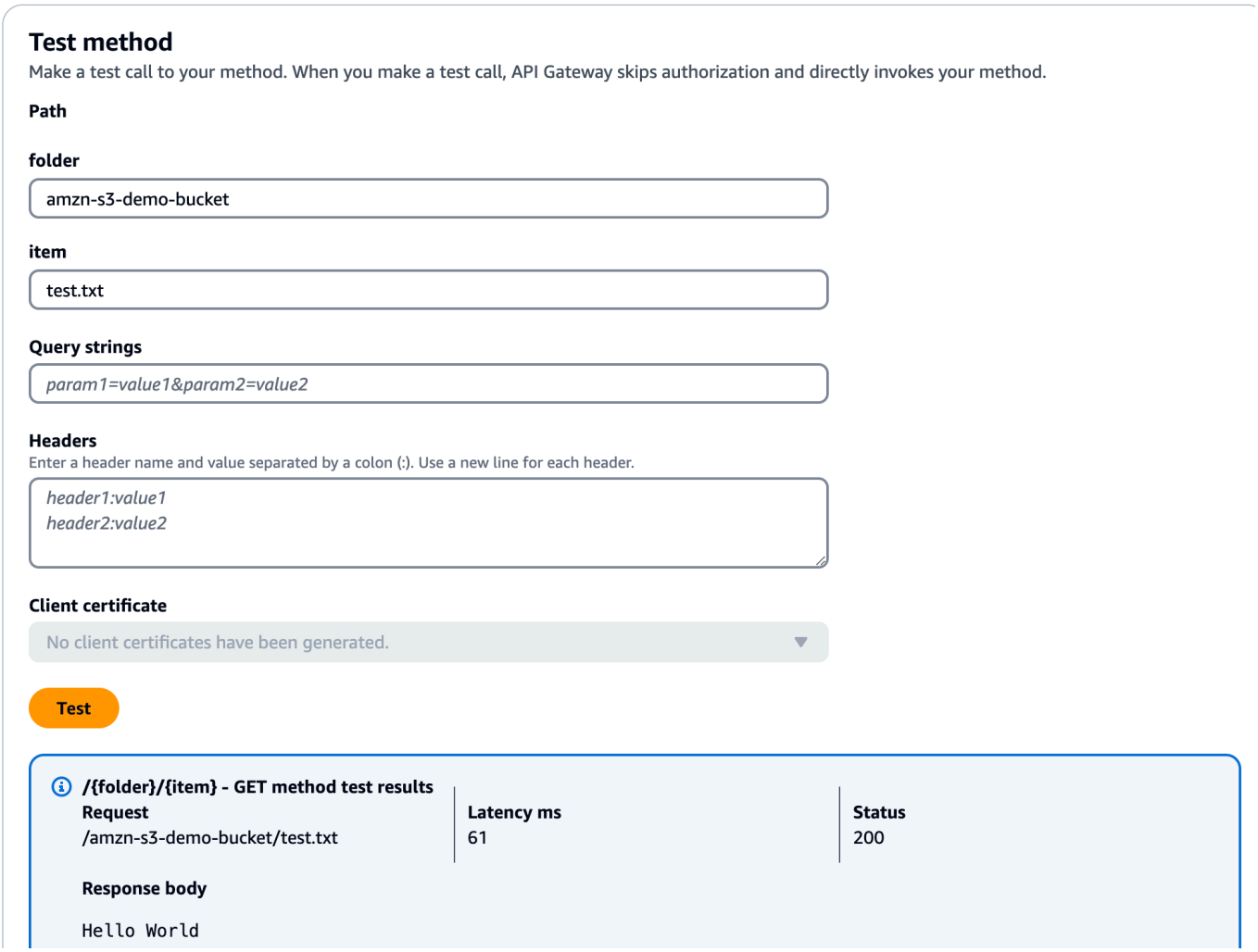
1. Nella scheda Richiesta di integrazione scegli Modifica in Impostazioni della richiesta di integrazione.
2. Scegli Parametri di percorso URL.
3. Scegli Aggiungi parametro di percorso.
4. In Nome, inserisci **`bucket`**.

5. In Mappato da, inserire **`method.request.path.folder`**.
6. Scegli Aggiungi parametro di percorso.
7. In Nome, inserisci **`object`**.
8. In Mappato da, inserire **`method.request.path.item`**.
9. Selezionare Salva.

Per testare il **`metodo`**/{folder}/{item} GET.

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. In Percorso immetti il nome del tuo bucket per folder.
3. In Percorso immetti il nome di un elemento per item.
4. Scegli Test (Esegui test).

   Il corpo della risposta conterrà il contenuto dell'elemento.



La richiesta restituisce correttamente il testo normale ("Hello world") come contenuto del file specificato (test.txt) nel bucket Amazon S3 fornito (amzn-s3-demo-bucket).

Per scaricare o caricare file binari ovvero, in API Gateway, tutti i contenuti JSON che hanno una codifica diversa da UTF-8, sono necessarie impostazioni API aggiuntive, come mostrato di seguito:

Per scaricare o caricare file binari da S3

1. Registra i tipi di file multimediali del file interessato nel file binario dell'APIMediaTypes. Questa operazione può essere eseguita nella console:

   a. Scegli Impostazioni API per l'API.

   b. In Tipi di media binari scegli Gestisci tipi di supporti.

c. Scegli Aggiungi tipo di supporto binario, quindi immetti il tipo di supporto richiesto, ad esempio `image/png`.

d. Scegli Salva modifiche per salvare l'impostazione.

2. Aggiungete l'`Content-Type`intestazione and/or `Accept` (per il caricamento) (per il download) alla richiesta del metodo per richiedere al client di specificare il tipo di supporto binario richiesto e mapparlo alla richiesta di integrazione.

3. Imposta Content Handling (Gestione contenuto) su `Passthrough` nella richiesta di integrazione (per il caricamento) e in una risposta di integrazione (per il download). Assicurati che per il tipo di contenuto interessato non sia definito un modello di mappatura. Per ulteriori informazioni, consulta the section called “Trasformazioni dei dati”.

Il limite della dimensione del payload è 10 MB. Consulta Quote per la configurazione e l’esecuzione di una REST API in Gateway API.

Assicurati che ai file in Amazon S3 vengano aggiunti tipi di contenuto corretti come metadati dei file. Per il contenuto multimediale riproducibile in streaming, potrebbe essere necessario aggiungere anche `Content-Disposition:inline` ai metadati.

Per ulteriori informazioni sul supporto binario in API Gateway, consulta Conversioni dei tipi di contenuto in API Gateway.

Definizioni OpenAPI dell'API di esempio come un proxy Amazon S3

Le definizioni OpenAPI seguenti descrivono un'API che funge da proxy Amazon S3. Questa API contiene più operazioni Amazon S3 rispetto all'API creata nel tutorial. Nelle definizioni OpenAPI sono esposti i seguenti metodi:

- Esponi GET sulla risorsa radice dell'API per elencare tutti i bucket Amazon S3 di un intermediario.
- Esponi GET su una risorsa cartella per visualizzare un elenco di tutti gli oggetti di un bucket Amazon S3.
- Esponi PUT su una risorsa cartella per aggiungere un bucket ad Amazon S3.
- Esponi DELETE su una risorsa cartella per rimuovere un bucket da Amazon S3.
- Esponi GET su una Folder/Item risorsa per visualizzare o scaricare un oggetto da un bucket Amazon S3.
- Esponi PUT su una Folder/Item risorsa per caricare un oggetto in un bucket Amazon S3.

- Esponi HEAD su una Folder/Item risorsa per ottenere i metadati degli oggetti in un bucket Amazon S3.
- Esponi DELETE su una Folder/Item risorsa per rimuovere un oggetto da un bucket Amazon S3.

Per istruzioni su come importare l'API tramite la definizione OpenAPI, consulta Sviluppo di REST API utilizzando OpenAPI in Gateway API.

Per istruzioni su come creare un'API simile, consulta Tutorial: creazione di una REST API come un proxy Amazon S3.

Per scoprire come richiamare questa API utilizzando Postman, che supporta l'autorizzazione IAM, consulta AWS . Chiamata dell'API mediante un client API REST

OpenAPI 2.0

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

OpenAPI 3.0

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
            }
          },
          "5\\d{2}" : {
            "statusCode" : "500"
          }
        },
        "passthroughBehavior" : "when_no_match",
        "type" : "aws"
      }
    }
  }
},
"components" : {
  "schemas" : {
    "Empty" : {
      "title" : "Empty Schema",
      "type" : "object"
    }
  }
}
}
```

Chiamata dell'API mediante un client API REST

Per fornire un tutorial completo, mostriamo ora come chiamare l'API utilizzando Postman, che supporta l'autorizzazione IAM. AWS

Per chiamare l'API proxy Amazon S3 utilizzando Postman

1. Distribuisci o ridistribuisci l'API. Annota l'URL di base dell'API visualizzato accanto a Invoke URL (URL chiamata) in alto in Stage Editor (Editor fasi).
2. Avvia Postman.
3. Seleziona Authorization (Autorizzazione), quindi scegli `AWS Signature`. Inserisci l'ID della chiave di accesso e la chiave di accesso segreta del tuo utente IAM rispettivamente nei campi SecretKey di immissione AccessKey e. Inserisci l'API Regione AWS a cui è distribuita la tua API nella casella di testo AWS Regione. Immetti `execute-api` nel campo di input Nome servizio.

   Puoi creare una coppia di chiavi dalla scheda Security Credentials (Credenziali di sicurezza) con il tuo account utente IAM nella console di gestione IAM.

4. Per aggiungere un bucket denominato `amzn-s3-demo-bucket` all'account Amazon S3 nella regione *`{region}`*:

   a. Seleziona PUT nell'elenco a discesa dei metodi e digita l'URL del metodo (`https://`*`api-id`*`.execute-api.`*`aws-region`*`.amazonaws.com/`*`stage`*`/`*`folder-name`*)

   b. Imposta il valore dell'intestazione `Content-Type` come `application/xml`. Potrebbe essere necessario eliminare le intestazioni esistenti prima di impostare il tipo di contenuto.

   c. Seleziona la voce di menu Body (Corpo) e digita il seguente frammento XML come corpo della richiesta:

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

   d. Seleziona Send (Invia) per inviare la richiesta. Se l'operazione viene eseguita correttamente, dovresti ricevere una risposta `200 OK` con un payload vuoto.

5. Per aggiungere un file di testo a un bucket, segui le istruzioni riportate sopra. Se specifichi il nome bucket **`amzn-s3-demo-bucket`** per `{folder}` e il nome file **`Readme.txt`** per `{item}` nell'URL e fornisci la stringa di testo **`Hello, World!`** come contenuti del file (rendendola così il payload di richiesta), la richiesta diventa

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

   Hello, World!
   ```

   Se l'operazione viene eseguita correttamente, dovresti ricevere una risposta `200 OK` con un payload vuoto.

6. Per ottenere il contenuto del file `Readme.txt` appena aggiunto al bucket `amzn-s3-demo-bucket`, esegui una richiesta GET come la seguente:

   ```
   GET /S3/amzn-s3-demo-bucket/Readme.txt HTTP/1.1
   ```

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

Se l'operazione viene eseguita correttamente, dovresti ricevere una risposta `200 OK` con la stringa di testo `Hello, World!` come payload.

7. Per elencare le voci nel bucket `amzn-s3-demo-bucket`, invia la richiesta seguente:

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

Se l'operazione riesce, dovresti ricevere una risposta `200 OK` con un payload XML che mostra una voce singola nel bucket specificato, a meno che tu non abbia aggiunto altri file al bucket prima di inviare questa richiesta.

```
<?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**
>
> Per caricare o scaricare un'immagine, è necessario impostare la gestione del contenuto su CONVERT_TO_BINARY.

Tutorial: creazione di un'API REST come un proxy Amazon Kinesis

In questa pagina viene descritto come creare e configurare un'API REST con un'integrazione di tipo `AWS` per accedere a Kinesis.

> **Note**
>
> Per integrare l'API di API Gateway con Kinesis, devi scegliere una regione in cui i servizi API Gateway e Kinesis siano entrambi disponibili. Per la disponibilità della regione, vedere Endpoint e quote del servizio.

Per scopi illustrativi, creeremo un'API di esempio per permettere a un client di eseguire le operazioni seguenti:

1. Elencare i flussi disponibili dell'utente in Kinesis
2. Creare, descrivere o eliminare un flusso specificato.
3. Leggere o scrivere record di dati nel flusso specificato

Per eseguire le attività precedenti, l'API espone metodi in diverse risorse per richiamare, rispettivamente, gli elementi seguenti:

1. L'operazione `ListStreams` in Kinesis
2. Operazione `CreateStream`, `DescribeStream` o `DeleteStream`
3. Operazione `GetRecords` o `PutRecords` (inclusa `PutRecord`) in Kinesis

In particolare, compileremo l'API in questo modo:

- Esponi un metodo HTTP GET sulla `/streams` risorsa dell'API e integra il metodo con l'ListStreamsazione in Kinesis per elencare gli stream nell'account del chiamante.
- Esponi un metodo HTTP POST sulla `/streams/{stream-name}` risorsa dell'API e integra il metodo con l'CreateStreamazione in Kinesis per creare uno stream denominato nell'account del chiamante.
- Esponi un metodo HTTP GET sulla `/streams/{stream-name}` risorsa dell'API e integra il metodo con l'DescribeStreamazione in Kinesis per descrivere uno stream denominato nell'account del chiamante.
- Esponi un metodo HTTP DELETE sulla `/streams/{stream-name}` risorsa dell'API e integra il metodo con l'DeleteStreamazione in Kinesis per eliminare uno stream nell'account del chiamante.
- Esponi un metodo HTTP PUT sulla `/streams/{stream-name}/record` risorsa dell'API e integra il metodo con l'PutRecordazione in Kinesis. In questo modo, il client può aggiungere un singolo record di dati al flusso denominato.
- Esponi un metodo HTTP PUT sulla `/streams/{stream-name}/records` risorsa dell'API e integra il metodo con l'PutRecordsazione in Kinesis. In questo modo, il client può aggiungere un elenco di record di dati al flusso denominato.
- Esponi un metodo HTTP GET sulla `/streams/{stream-name}/records` risorsa dell'API e integra il metodo con l'GetRecordsazione in Kinesis. In questo modo, il client può aggiungere un elenco di record di dati nel flusso denominato, con un'iterazione shard specificata. Un'iterazione shard specifica la posizione dello shard da cui iniziare a leggere i record di dati in sequenza.
- Esponi un metodo HTTP GET sulla `/streams/{stream-name}/sharditerator` risorsa dell'API e integra il metodo con l'GetShardIteratorazione in Kinesis. Questo metodo helper deve essere fornito all'operazione `ListStreams` in Kinesis.

Puoi applicare le istruzioni presentate qui ad altre operazioni di Kinesis. Per l'elenco completo delle operazioni di Kinesis, consulta la documentazione di riferimento delle API di Amazon Kinesis.

Invece di usare la console API Gateway per creare l'API di esempio, puoi importare l'API di esempio in API Gateway, usando l'API Gateway di Importa API. Per informazioni su come usare l'API di importazione, consulta Sviluppo di REST API utilizzando OpenAPI in Gateway API.

Creazione di un ruolo e una policy IAM per l'API per accedere a Kinesis

Per permettere all'API di richiamare le operazioni di Kinesis, devi aver collegato le policy IAM appropriate a un ruolo IAM. In questo passaggio si crea un nuovo ruolo IAM.

Per creare il AWS ruolo di esecuzione del servizio proxy

1. Accedi Console di gestione AWS e apri la console IAM all'indirizzo https://console.aws.amazon.com/iam/.
2. Scegliere Roles (Ruoli).
3. Scegli Crea ruolo.
4. Scegli il AWS servizio in Seleziona il tipo di entità affidabile, quindi seleziona API Gateway e seleziona Consenti a API Gateway di inviare i log ai CloudWatch log.
5. Scegli Successivo e di nuovo Successivo.
6. In Role name (Nome ruolo) immettere **`APIGatewayKinesisProxyPolicy`** e quindi selezionare Create role (Crea ruolo).
7. Nell'elenco Roles (Ruoli) scegliere il ruolo appena creato. Potrebbe essere necessario scorrere o utilizzare la barra di ricerca per trovare il ruolo.
8. Per il ruolo selezionato, seleziona la scheda Aggiungi autorizzazioni.
9. Dall'elenco a discesa scegli Collega policy.
10. Nella barra di ricerca inserisci **`AmazonKinesisFullAccess`** e scegli Aggiungi autorizzazioni.

> **Note**
>
> Per semplicità, questo tutorial utilizza una policy gestita. Come best practice, dovresti creare la tua policy IAM per concedere le autorizzazioni minime richieste.

11. Annota l'ARN del ruolo appena creato, lo utilizzerai in seguito.

Creazione di un'API come un proxy Kinesis

Usa la procedura seguente per creare l'API nella console API Gateway.

Per creare un'API come AWS proxy di servizio per Kinesis

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.

2. Se si utilizza API Gateway per la prima volta, verrà visualizzata una pagina che presenta le caratteristiche del servizio. In API REST, scegliere Crea. Quando appare il popup Create Example API (Crea API di esempio), scegliere OK.

   Se non è la prima volta che si utilizza API Gateway, scegliere Create API (Crea API). In API REST, scegliere Crea.
3. Selezionare New API (Nuova API).
4. Per API name (Nome API) immettere **`KinesisProxy`**. Per tutti gli altri campi mantieni i valori predefiniti.
5. (Facoltativo) In Descrizione, immetti una descrizione.
6. Per Tipo di indirizzo IP, selezionare IPv4.
7. Seleziona Create API (Crea API).

Dopo la creazione dell'API, la console API Gateway visualizza la pagina Resources (Risorse), che contiene solo la risorsa root (`/`) dell'API.

Visualizzazione dell'elenco di flussi in Kinesis

Kinesis supporta l'operazione `ListStreams` con la chiamata API REST seguente:

```
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>

{
   ...
}
```

Nella richiesta API REST precedente l'operazione è specificata nel parametro di query `Action`. In alternativa, puoi specificare l'operazione in un'intestazione `X-Amz-Target`:

```
POST / HTTP/1.1
Host: kinesis.<region>.<domain>
Content-Length: <PayloadSizeBytes>
```

```
User-Agent: <UserAgentString>
Content-Type: application/x-amz-json-1.1
Authorization: <AuthParams>
X-Amz-Date: <Date>
X-Amz-Target: Kinesis_20131202.ListStreams
{
   ...
}
```

In questo tutorial useremo il parametro di query per specificare l'operazione.

Per esporre un'operazione di Kinesis nell'API, aggiungi una risorsa `/streams` alla root dell'API. Imposta quindi un metodo `GET` nella risorsa e integra il metodo con l'operazione `ListStreams` di Kinesis.

La procedura seguente descrive come elencare flussi Kinesis tramite la console API Gateway.

Per elencare i flussi Kinesis utilizzando la console API Gateway

1. Seleziona la risorsa `/`, quindi scegli Crea risorsa.
2. Per Resource Name (Nome risorsa) immetti **`streams`**.
3. Mantieni CORS (Cross Origin Resource Sharing) disattivato.
4. Scegli Crea risorsa.
5. Scegli la risorsa `/streams` e seleziona Crea metodo, quindi procedi come segue:

   a. Per Tipo di metodo seleziona GET.

   > **Note**
   >
   > Il verbo HTTP per un metodo richiamato da un client può differire dal verbo HTTP per un'integrazione richiesta dal back-end. Qui selezioniamo `GET` perché la visualizzazione dell'elenco di flussi è intuitivamente un'operazione READ.

   b. Per Tipo di integrazione seleziona Servizio AWS .

   c. Per Regione AWS, seleziona il Regione AWS luogo in cui hai creato lo stream Kinesis.

   d. Per Servizio AWS seleziona Kinesis.

   e. Lascia vuoto Sottodominio AWS .

   f. Per HTTP method (Metodo HTTP) scegli POST.

> **Note**
>
> Qui scegliamo `POST` perché Kinesis richiede che con il metodo venga richiamata l'operazione `ListStreams`.

g. Per Tipo di operazione scegli Usa nome operazione.

h. Per Nome azione immetti **`ListStreams`**.

i. Per Ruolo di esecuzione immetti l'ARN del ruolo di esecuzione.

j. Lascia il valore predefinito Transito per Gestione contenuti.

k. Scegli Crea metodo.

6. Nella scheda Richiesta di integrazione scegli Modifica in Impostazioni della richiesta di integrazione.
7. Per Richiesta corpo passthrough scegli Quando non ci sono modelli definiti (consigliato).
8. Scegli Parametri delle intestazioni delle richieste URL ed effettua le seguenti operazioni:

   a. Scegli Aggiungi parametro delle intestazioni delle richieste.

   b. In Nome, inserisci **`Content-Type`**.

   c. In Mappato da, inserire **`'application/x-amz-json-1.1'`**.

   Usiamo la mappatura dei parametri delle richieste per impostare l'intestazione `Content-Type` sul valore statico `'application/x-amz-json-1.1'` per indicare a Kinesis che l'input è una versione specifica di JSON.

9. Scegli Modelli di mappatura, quindi seleziona Aggiungi modello di mappatura ed effettua le seguenti operazioni:

   a. In Content-Type, immettere **`application/json`**.

   b. Per Corpo del modello immetti **`{}`**.

   c. Scegli Save (Salva).

   La ListStreamsrichiesta richiede un payload nel seguente formato JSON:

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

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

Tuttavia, le proprietà sono facoltative. Per usare i valori predefiniti, qui abbiamo optato per un payload JSON vuoto.

10. Testa il metodo GET nella risorsa /streams per richiamare l'azione `ListStreams` in Kinesis:

    Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.

    Scegli Esegui test per testare il metodo.

    Se hai già creato due flussi denominati "myStream" e "yourStream" in Kinesis, il test con esito positivo restituisce una risposta 200 OK che contiene il payload seguente:

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

Creazione, descrizione ed eliminazione di un flusso in Kinesis

Per creare, descrivere ed eliminare un flusso in Kinesis è necessario effettuare, rispettivamente, le richieste API REST di Kinesis seguenti:

```
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"
}
```

Puoi compilare l'API in modo da accettare l'input richiesto come payload JSON della richiesta del metodo e passare il payload alla richiesta di integrazione. Tuttavia, per fornire più esempi di mappatura dei dati tra richieste di metodi e di integrazione e le rispettive risposte, creeremo l'API in un modo diverso.

Esporremo i metodi HTTP `GET`, `POST` e `Delete` in una risorsa `Stream` da denominare. Useremo la variabile di percorso `{stream-name}` come segnaposto della risorsa flusso e integreremo questi metodi API rispettivamente con le operazioni `DescribeStream`, `CreateStream` e `DeleteStream` di Kinesis. Richiederemo che il client passi altri dati di input come intestazioni, parametri di query o payload della richiesta di un metodo. Specificheremo modelli di mappatura per trasformare i dati nel payload della richiesta di integrazione necessario.

Per creare la risorsa {stream-name}

1. Scegli la risorsa /streams, quindi seleziona Crea risorsa.
2. Mantieni l'opzione Risorsa proxy disattivata.
3. Per Percorso risorsa seleziona `/streams`.
4. Per Resource Name (Nome risorsa) immetti **`{stream-name}`**.
5. Mantieni CORS (Cross Origin Resource Sharing) disattivato.
6. Scegli Crea risorsa.

Per configurare e testare il metodo GET in una risorsa flusso

1. Scegli la risorsa /{stream-name}, quindi seleziona Crea metodo.
2. Per Tipo di metodo seleziona GET.
3. Per Tipo di integrazione seleziona Servizio AWS .
4. Per Regione AWS, seleziona il Regione AWS luogo in cui hai creato lo stream Kinesis.
5. Per Servizio AWS seleziona Kinesis.
6. Lascia vuoto Sottodominio AWS .
7. Per HTTP method (Metodo HTTP) scegli POST.
8. Per Tipo di operazione scegli Usa nome operazione.
9. Per Nome azione immetti **`DescribeStream`**.
10. Per Ruolo di esecuzione immetti l'ARN del ruolo di esecuzione.
11. Lascia il valore predefinito Transito per Gestione contenuti.
12. Scegli Crea metodo.
13. Nella sezione Richiesta di integrazione aggiungi i seguenti parametri delle intestazioni delle richieste URL:

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

    L'attività segue la stessa procedura usata per configurare la mappatura dei parametri di richiesta per il metodo `GET /streams`.
14. Aggiungi il modello di mappatura del corpo seguente per mappare i dati dalla richiesta del metodo `GET /streams/{stream-name}` alla richiesta di integrazione `POST /?Action=DescribeStream`:

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

Questo modello di mappatura genera il payload della richiesta di integrazione necessario per l'operazione `DescribeStream` di Kinesis dal valore del parametro di percorso `stream-name` della richiesta del metodo.

15. Per testare il metodo `GET /stream/{stream-name}` per richiamare l'azione `DescribeStream` in Kinesis, scegli la scheda Test.
16. Per Percorso immetti il nome di un flusso Kinesis esistente in stream-name.
17. Scegli Test (Esegui test). Se il test ha esito positivo, viene restituita una risposta 200 OK con un payload simile al seguente:

```
{
  "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"
  }
}
```

Dopo aver distribuito l'API, puoi effettuare una richiesta REST su questo metodo API:

```
GET https://your-api-id.execute-api.region.amazonaws.com/stage/streams/myStream
 HTTP/1.1
Host: your-api-id.execute-api.region.amazonaws.com
Content-Type: application/json
Authorization: ...
X-Amz-Date: 20160323T194451Z
```

Per configurare e testare il metodo POST in una risorsa flusso

1. Scegli la risorsa /{stream-name}, quindi seleziona Crea metodo.
2. Per Tipo di metodo seleziona POST.
3. Per Tipo di integrazione seleziona Servizio AWS .
4. Per Regione AWS, seleziona il Regione AWS luogo in cui hai creato lo stream Kinesis.
5. Per Servizio AWS seleziona Kinesis.
6. Lascia vuoto Sottodominio AWS .
7. Per HTTP method (Metodo HTTP) scegli POST.
8. Per Tipo di operazione scegli Usa nome operazione.
9. Per Nome azione immetti **`CreateStream`**.
10. Per Ruolo di esecuzione immetti l'ARN del ruolo di esecuzione.
11. Lascia il valore predefinito Transito per Gestione contenuti.
12. Scegli Crea metodo.
13. Nella sezione Richiesta di integrazione aggiungi i seguenti parametri delle intestazioni delle richieste URL:

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

L'attività segue la stessa procedura usata per configurare la mappatura dei parametri di richiesta per il metodo `GET /streams`.

14. Aggiungi il modello di mappatura del corpo seguente per mappare i dati dalla richiesta del metodo `POST /streams/{stream-name}` alla richiesta di integrazione `POST /?Action=CreateStream`:

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

Nel modello di mappatura precedente impostiamo `ShardCount` sul valore fisso 5 se il client non specifica alcun valore nel payload della richiesta del metodo.

15. Per testare il metodo `POST /stream/{stream-name}` per richiamare l'azione `CreateStream` in Kinesis, scegli la scheda Test.
16. Per Percorso immetti il nome di un nuovo flusso Kinesis in stream-name.
17. Scegli Test (Esegui test). Se il test ha esito positivo, viene restituita una risposta 200 OK senza dati.

Dopo aver distribuito l'API, puoi anche effettuare una richiesta API REST sul metodo POST in una risorsa flusso per richiamare l'operazione `CreateStream` in 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
}
```

Configurazione e test del metodo DELETE in una risorsa flusso

1. Scegli la risorsa /{stream-name}, quindi seleziona Crea metodo.
2. Per Tipo di metodo seleziona DELETE.
3. Per Tipo di integrazione seleziona Servizio AWS .
4. Per Regione AWS, seleziona il Regione AWS luogo in cui hai creato lo stream Kinesis.
5. Per Servizio AWS seleziona Kinesis.
6. Lascia vuoto Sottodominio AWS .
7. Per HTTP method (Metodo HTTP) scegli POST.
8. Per Tipo di operazione scegli Usa nome operazione.
9. Per Nome azione immetti **`DeleteStream`**.
10. Per Ruolo di esecuzione immetti l'ARN del ruolo di esecuzione.
11. Lascia il valore predefinito Transito per Gestione contenuti.
12. Scegli Crea metodo.
13. Nella sezione Richiesta di integrazione aggiungi i seguenti parametri delle intestazioni delle richieste URL:

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

    L'attività segue la stessa procedura usata per configurare la mappatura dei parametri di richiesta per il metodo `GET /streams`.
14. Aggiungi il modello di mappatura del corpo seguente per mappare i dati dalla richiesta del metodo `DELETE /streams/{stream-name}` alla richiesta di integrazione corrispondente di `POST /?Action=DeleteStream`:

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

    Questo modello di mappatura genera l'input richiesto per l'operazione `DELETE /streams/{stream-name}` dal nome di percorso URL fornito dal client `stream-name`.
15. Per testare il metodo `DELETE /stream/{stream-name}` per richiamare l'azione `DeleteStream` in Kinesis, scegli la scheda Test.
16. Per Percorso immetti il nome di un flusso Kinesis esistente in stream-name.

17. Scegli Test (Esegui test). Se il test ha esito positivo, viene restituita una risposta 200 OK senza dati.

    Dopo aver distribuito l'API, puoi anche effettuare la richiesta API REST seguente sul metodo DELETE nella risorsa flusso per chiamare l'operazione `DeleteStream` in 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

{}
```

Recupero di record e aggiunta di record in un flusso in Kinesis

Dopo aver creato un flusso in Kinesis, puoi aggiungere record di dati al flusso e leggere i dati dal flusso. L'aggiunta di record di dati implica la chiamata all'PutRecordazione PutRecordso in Kinesis. La prima operazione aggiunge più record, la seconda aggiunge un singolo record al flusso.

```
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"
}
```

oppure

```
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"
}
```

Qui `StreamName` identifica il flusso di destinazione per aggiungere record. `StreamName`, `Data`, e `PartitionKey` sono dati di input obbligatori. Nel nostro esempio useremo i valori predefiniti per tutti i dati di input facoltativi e non ne specificheremo in modo esplicito i valori nell'input per la richiesta del metodo.

Leggere i dati in Kinesis equivale a richiamare l'azione 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
}
```

Qui il flusso di origine da cui vogliamo ottenere record è specificato nel valore `ShardIterator` obbligatorio, come mostrato nell'operazione di Kinesis seguente per ottenere un'iterazione shard:

```
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

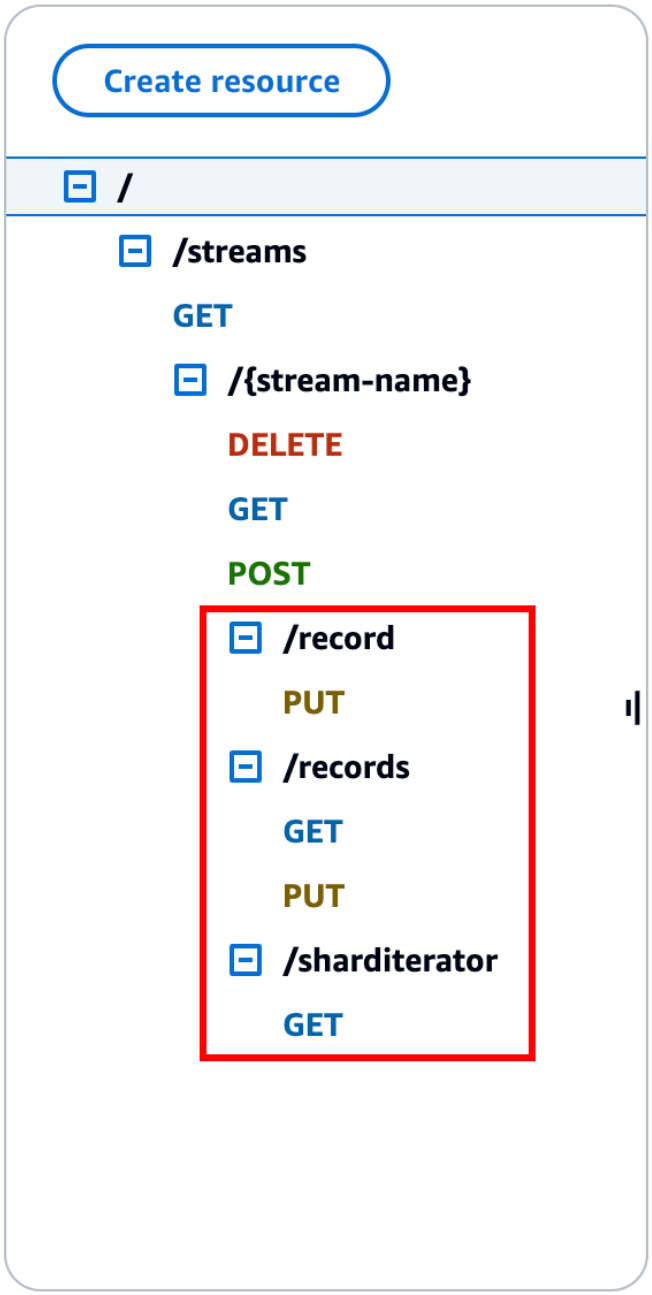
{
    "ShardId": "string",
    "ShardIteratorType": "string",
    "StartingSequenceNumber": "string",
    "StreamName": "string"
}
```

Per le operazioni `GetRecords` e `PutRecords`, esponiamo rispettivamente i metodi `GET` e `PUT` in una risorsa `/records` che viene aggiunta a una risorsa flusso denominata (`/{stream-name}`). Analogamente, esponiamo l'operazione `PutRecord` come metodo `PUT` in una risorsa `/record`.

Poiché l'operazione `GetRecords` accetta come input un valore `ShardIterator`, ottenuto chiamando l'operazione helper `GetShardIterator`, esponiamo un metodo helper `GET` in una risorsa `ShardIterator` (`/sharditerator`).

Per creare le risorse /record, /records e /sharditerator

1. Scegli la risorsa /{stream-name}, quindi seleziona Crea risorsa.
2. Mantieni l'opzione Risorsa proxy disattivata.
3. Per Percorso risorsa seleziona `/{stream-name}`.
4. Per Resource Name (Nome risorsa) immetti **`record`**.
5. Mantieni CORS (Cross Origin Resource Sharing) disattivato.
6. Scegli Crea risorsa.
7. Ripeti le fasi precedenti per creare una risorsa /records e una risorsa /sharditerator. L'API finale sarà simile alla seguente:



Le quattro procedure seguenti descrivono come configurare ognuno dei metodi, come mappare i dati dalle richieste dei metodi alle richieste di integrazione e come testare i metodi.

Per configurare e testare il **`metodo PUT /streams/ {stream-name} /record`** da invocare in Kinesis: **`PutRecord`**

1. Scegli la risorsa /record, quindi seleziona Crea metodo.

2. Per Tipo di metodo seleziona PUT.
3. Per Tipo di integrazione seleziona Servizio AWS .
4. Per Regione AWS, seleziona il Regione AWS luogo in cui hai creato lo stream Kinesis.
5. Per Servizio AWS seleziona Kinesis.
6. Lascia vuoto Sottodominio AWS .
7. Per HTTP method (Metodo HTTP) scegli POST.
8. Per Tipo di operazione scegli Usa nome operazione.
9. Per Nome azione immetti **`PutRecord`**.
10. Per Ruolo di esecuzione immetti l'ARN del ruolo di esecuzione.
11. Lascia il valore predefinito Transito per Gestione contenuti.
12. Scegli Crea metodo.
13. Nella sezione Richiesta di integrazione aggiungi i seguenti parametri delle intestazioni delle richieste URL:

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

    L'attività segue la stessa procedura usata per configurare la mappatura dei parametri di richiesta per il metodo `GET /streams`.
14. Aggiungi il modello di mappatura del corpo seguente per mappare i dati dalla richiesta del metodo `PUT /streams/{stream-name}/record` alla richiesta di integrazione corrispondente di `POST /?Action=PutRecord`:

    ```
    {
        "StreamName": "$input.params('stream-name')",
        "Data": "$util.base64Encode($input.json('$.Data'))",
        "PartitionKey": "$input.path('$.PartitionKey')"
    }
    ```

    Questo modello di mappatura presuppone che il payload della richiesta del metodo abbia il formato seguente:

    ```
    {
       "Data": "some data",
       "PartitionKey": "some key"
    }
    ```

Questi dati possono essere modellati tramite lo schema JSON seguente:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "PutRecord proxy single-record payload",
  "type": "object",
  "properties": {
      "Data": { "type": "string" },
      "PartitionKey": { "type": "string" }
  }
}
```

Puoi creare un modello per includere questo schema e usare il modello per semplificare la generazione del modello di mappatura. Tuttavia, puoi generare un modello di mappatura senza usare alcun modello.

15. Per testare il metodo PUT `/streams/{stream-name}/record`, imposta la variabile di percorso `stream-name` sul nome di un flusso esistente, fornisci un payload con il formato richiesto e quindi invia la richiesta del metodo. Il risultato con esito positivo è una risposta `200 OK` con un payload nel formato seguente:

```
{
  "SequenceNumber": "49559409944537880850133345460169886593573102115167928386",
  "ShardId": "shardId-000000000004"
}
```

Per configurare e testare il metodo **`PUT /streams/ {stream-name} /records`** da invocare in Kinesis **`PutRecords`**

1. Scegli la risorsa /records, quindi seleziona Crea metodo.
2. Per Tipo di metodo seleziona PUT.
3. Per Tipo di integrazione seleziona Servizio AWS .
4. Per Regione AWS, seleziona il Regione AWS luogo in cui hai creato lo stream Kinesis.
5. Per Servizio AWS seleziona Kinesis.
6. Lascia vuoto Sottodominio AWS .

7. Per HTTP method (Metodo HTTP) scegli POST.
8. Per Tipo di operazione scegli Usa nome operazione.
9. Per Nome azione immetti **`PutRecords`**.
10. Per Ruolo di esecuzione immetti l'ARN del ruolo di esecuzione.
11. Lascia il valore predefinito Transito per Gestione contenuti.
12. Scegli Crea metodo.
13. Nella sezione Richiesta di integrazione aggiungi i seguenti parametri delle intestazioni delle richieste URL:

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

L'attività segue la stessa procedura usata per configurare la mappatura dei parametri di richiesta per il metodo `GET /streams`.

14. Aggiungi il modello di mappatura seguente per mappare i dati dalla richiesta del metodo `PUT /streams/{stream-name}/records` alla richiesta di integrazione corrispondente di `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
    ]
}
```

Questo modello di mappatura presuppone che il payload della richiesta del metodo possa essere modellato tramite lo schema JSON seguente:

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

Puoi creare un modello per includere questo schema e usare il modello per semplificare la generazione del modello di mappatura. Tuttavia, puoi generare un modello di mappatura senza usare alcun modello.

In questo tutorial abbiamo usato due formati di payload leggermente diversi per mostrare come uno sviluppatore di API possa scegliere di esporre il formato di dati di back-end al client o nasconderlo dal client. Un formato è per il metodo PUT `/streams/{stream-name}/records` (sopra). Un altro formato viene usato per il metodo PUT `/streams/{stream-name}/record` (nella procedura precedente). Nell'ambiente di produzione devi mantenere i due formati coerenti.

15. Per testare il metodo PUT `/streams/{stream-name}/records`, imposta la variabile di percorso `stream-name` su un flusso esistente, fornisci il payload seguente e invia la richiesta del metodo.

```
{
    "records": [
        {
            "data": "some data",
            "partition-key": "some key"
        },
        {
            "data": "some other data",
            "partition-key": "some key"
        }
    ]
}
```

Il risultato con esito positivo è una risposta 200 OK con un payload simile all'output seguente:

```
{
  "FailedRecordCount": 0,
  "Records": [
    {
      "SequenceNumber": "49559409944537880850133345460167468741933742152373764162",
      "ShardId": "shardId-000000000004"
    },
    {
      "SequenceNumber": "49559409944537880850133345460168677667753356781548470338",
      "ShardId": "shardId-000000000004"
    }
  ]
}
```

Per configurare e testare il metodo **`GET /streams/ {stream-name} /sharditerator`**, invoca in Kinesis **`GetShardIterator`**

Il metodo `GET /streams/{stream-name}/sharditerator` è un metodo helper per acquisire un'iterazione shard necessaria prima di chiamare il metodo `GET /streams/{stream-name}/records`.

1. Scegli la risorsa /sharditerator, quindi seleziona Crea metodo.
2. Per Tipo di metodo seleziona GET.
3. Per Tipo di integrazione seleziona Servizio AWS .
4. Per Regione AWS, seleziona il Regione AWS luogo in cui hai creato lo stream Kinesis.
5. Per Servizio AWS seleziona Kinesis.
6. Lascia vuoto Sottodominio AWS .
7. Per HTTP method (Metodo HTTP) scegli POST.
8. Per Tipo di operazione scegli Usa nome operazione.
9. Per Nome azione immetti **`GetShardIterator`**.
10. Per Ruolo di esecuzione immetti l'ARN del ruolo di esecuzione.
11. Lascia il valore predefinito Transito per Gestione contenuti.
12. Scegli Parametri della stringa di query URL.

L'operazione `GetShardIterator` richiede l'immissione di un valore ShardId. Per passare un valore `ShardId` fornito dal client, aggiungiamo un parametro di query `shard-id` alla richiesta del metodo, come mostrato nella seguente fase.

13. Scegliere Add query string (Aggiungi stringa di query).
14. In Nome, inserisci **`shard-id`**.
15. Mantieni Obbligatorio e Caching disattivati.
16. Scegli Crea metodo.
17. Nella sezione Richiesta di integrazione aggiungi il seguente modello di mappatura per generare l'input richiesto (`ShardId` e `StreamName`) per l'azione `GetShardIterator` dei parametri `shard-id` e `stream-name` della richiesta del metodo. Inoltre, il modello di mappatura imposta anche `ShardIteratorType` su `TRIM_HORIZON` come comportamento predefinito.

```
{
    "ShardId": "$input.params('shard-id')",
    "ShardIteratorType": "TRIM_HORIZON",
    "StreamName": "$input.params('stream-name')"
}
```

18. Usando l'opzione Test nella console API Gateway, immetti il nome di un flusso esistente come valore della variabile `stream-name`Path (Percorso), imposta `shard-id`Query string (Stringa di query) su un valore `ShardId` esistente, ad esempio `shard-000000000004`, e seleziona Test.

    Il payload di risposta di esito positivo è simile all'output seguente:

```
{
  "ShardIterator": "AAAAAAAAAAFYVN3VlFy..."
}
```

    Annota il valore di `ShardIterator`. Ti servirà per ottenere record da un flusso.

Per configurare e testare il metodo **`GET /streams/ {stream-name} /records`** per richiamare l'azione in Kinesis **`GetRecords`**

1. Scegli la risorsa /records, quindi seleziona Crea metodo.
2. Per Tipo di metodo seleziona GET.
3. Per Tipo di integrazione seleziona Servizio AWS .

4. Per Regione AWS, seleziona il Regione AWS luogo in cui hai creato lo stream Kinesis.
5. Per Servizio AWS seleziona Kinesis.
6. Lascia vuoto Sottodominio AWS .
7. Per HTTP method (Metodo HTTP) scegli POST.
8. Per Tipo di operazione scegli Usa nome operazione.
9. Per Nome azione immetti **`GetRecords`**.
10. Per Ruolo di esecuzione immetti l'ARN del ruolo di esecuzione.
11. Lascia il valore predefinito Transito per Gestione contenuti.
12. Scegli Intestazioni di richiesta HTTP.

    L'operazione `GetRecords` richiede l'immissione di un valore `ShardIterator`. Per passare un valore `ShardIterator` fornito dal client, si aggiunge un parametro di intestazione `Shard-Iterator` alla richiesta del metodo.
13. Seleziona Add header (Aggiungi intestazione).
14. In Nome, inserisci **`Shard-Iterator`**.
15. Mantieni Obbligatorio e Caching disattivati.
16. Scegli Crea metodo.
17. Nella sezione Richiesta di integrazione aggiungi il seguente modello di mappatura del corpo per mappare il valore del parametro di intestazione `Shard-Iterator` al valore della proprietà `ShardIterator` del payload JSON per l'azione `GetRecords` in Kinesis.

    ```
    {
        "ShardIterator": "$input.params('Shard-Iterator')"
    }
    ```
18. Usando l'opzione Test nella console Gateway API, immetti il nome di un flusso esistente come valore della variabile Percorso `stream-name`, imposta Intestazione `Shard-Iterator` sul valore `ShardIterator` ottenuto dall'esecuzione del test del metodo `GET /streams/{stream-name}/sharditerator` (precedente) e scegli Test.

    Il payload di risposta di esito positivo è simile all'output seguente:

    ```
    {
      "MillisBehindLatest": 0,
      "NextShardIterator": "AAAAAAAAAAF...",
      "Records": [ ... ]
    ```

```
}
```

Definizioni OpenAPI di un'API di esempio come un proxy Kinesis

Di seguito sono riportate le definizioni OpenAPI per l'API di esempio come proxy Kinesis utilizzate in questo tutorial.

OpenAPI 3.0

```
{
  "openapi": "3.0.0",
  "info": {
    "title": "KinesisProxy",
    "version": "2016-03-31T18:25:32Z"
  },
  "paths": {
    "/streams/{stream-name}/sharditerator": {
      "get": {
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          },
          {
            "name": "shard-id",
            "in": "query",
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
```

```
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/GetShardIterator",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"ShardId\": \"$input.params('shard-
id')\",\n    \"ShardIteratorType\": \"TRIM_HORIZON\",\n    \"StreamName\":
 \"$input.params('stream-name')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      }
    },
    "/streams/{stream-name}/records": {
      "get": {
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          },
          {
            "name": "Shard-Iterator",
            "in": "header",
            "schema": {
              "type": "string"
```

```
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "aws",
          "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
          "uri": "arn:aws:apigateway:us-east-1:kinesis:action/GetRecords",
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"ShardIterator\": \"$input.params('Shard-
Iterator')\"\n}"
          },
          "passthroughBehavior": "when_no_match",
          "httpMethod": "POST"
        }
      },
      "put": {
        "parameters": [
          {
            "name": "Content-Type",
            "in": "header",
            "schema": {
              "type": "string"
            }
```

```
      },
      {
        "name": "stream-name",
        "in": "path",
        "required": true,
        "schema": {
          "type": "string"
        }
      }
    ],
    "requestBody": {
      "content": {
        "application/json": {
          "schema": {
            "$ref": "#/components/schemas/PutRecordsMethodRequestPayload"
          }
        },
        "application/x-amz-json-1.1": {
          "schema": {
            "$ref": "#/components/schemas/PutRecordsMethodRequestPayload"
          }
        }
      },
      "required": true
    },
    "responses": {
      "200": {
        "description": "200 response",
        "content": {
          "application/json": {
            "schema": {
              "$ref": "#/components/schemas/Empty"
            }
          }
        }
      }
    },
    "x-amazon-apigateway-integration": {
      "type": "aws",
      "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
      "uri": "arn:aws:apigateway:us-east-1:kinesis:action/PutRecords",
      "responses": {
        "default": {
          "statusCode": "200"
```

```
            }
          },
          "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
          },
          "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\",\n    \"Records\": [\n          {\n            \"Data\":
 \"$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: creare un'API REST utilizzando AWS SDK o AWS CLI

Usa il seguente tutorial per creare un' PetStore API che supporti i metodi and. `GET /pets` `GET /pets/{petId}` I metodi sono integrati con un endpoint HTTP. Puoi seguire questo tutorial usando l' AWS SDK per JavaScript, il AWS SDK per Python (Boto3), o il AWS CLI. Per configurare l'API vengono utilizzati i seguenti comandi o funzioni:

JavaScript v3

- CreateRestApiCommand
- CreateResourceCommand
- PutMethodCommand
- PutMethodResponseCommand
- PutIntegrationCommand
- PutIntegrationResponseCommand
- CreateDeploymentCommand

Python

- create_rest_api
- create_resource
- put_method
- put_method_response
- put_integration
- put_integration_response
- create_deployment

AWS CLI

- create-rest-api
- create-resource
- put-method
- put-method-response
- put-integration
- put-integration-response
- create-deployment

Per ulteriori informazioni sull' AWS SDK per la JavaScript v3, vedi A cosa serve l' AWS SDK? JavaScript . Per ulteriori informazioni sull'SDK per Python (Boto3), consulta AWS SDK per Python (Boto3). Per ulteriori informazioni su AWS CLI, vedere Cos'è il AWS CLI? .

Configura un'API ottimizzata per l'edge PetStore

In questo tutorial, i comandi di esempio utilizzano valori segnaposto per gli ID valore come ID API e ID risorsa. Una volta completato il tutorial, sostituisci questi valori con valori personalizzati.

Per configurare un'API ottimizzata per l'edge utilizzando PetStore AWS SDK

1. Utilizza il seguente esempio per creare un'entità `RestApi`:

   JavaScript v3

   ```
   import {APIGatewayClient, CreateRestApiCommand} from "@aws-sdk/client-api-
   gateway";
   (async function (){
   const apig = new APIGatewayClient({region:"us-east-1"});
   const command = new CreateRestApiCommand({
       name: "Simple PetStore (JavaScript v3 SDK)",
       description: "Demo API created using the AWS SDK for JavaScript v3",
       version: "0.00.001",
       binaryMediaTypes: [
       '*']
   });
   try {
       const results = await apig.send(command)
       console.log(results)
   } catch (err) {
       console.error(Couldn't create API:\n", err)
   }
   })();
   ```

   Una chiamata riuscita restituisce l'ID API e l'ID risorsa root dell'API in un output simile al seguente:

   ```
   {
     id: 'abc1234',
     name: 'PetStore (JavaScript v3 SDK)',
     description: 'Demo API created using the AWS SDK for node.js',
     createdDate: 2017-09-05T19:32:35.000Z,
     version: '0.00.001',
     rootResourceId: 'efg567'
     binaryMediaTypes: [ '*' ]
   }
   ```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.create_rest_api(
        name='Simple PetStore (Python SDK)',
        description='Demo API created using the AWS SDK for Python',
        version='0.00.001',
        binaryMediaTypes=[
            '*'
        ]
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Couldn't create REST API %s.", error)
    raise
attribute=["id","name","description","createdDate","version","binaryMediaTypes","apiKeyS
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

Una chiamata riuscita restituisce l'ID API e l'ID risorsa root dell'API in un output simile al seguente:

```
{'id': 'abc1234', 'name': 'Simple PetStore (Python SDK)', 'description':
 'Demo API created using the AWS SDK for Python', 'createdDate':
 datetime.datetime(2024, 4, 3, 14, 31, 39, tzinfo=tzlocal()), 'version':
 '0.00.001', 'binaryMediaTypes': ['*'], 'apiKeySource': 'HEADER',
 'endpointConfiguration': {'types': ['EDGE']}, 'disableExecuteApiEndpoint':
 False, 'rootResourceId': 'efg567'}
```

AWS CLI

```
aws apigateway create-rest-api --name 'Simple PetStore (AWS CLI)' --region us-
west-2
```

L'output di questo comando è il seguente:

```
{
    "id": "abcd1234",
    "name": "Simple PetStore (AWS CLI)",
    "createdDate": "2022-12-15T08:07:04-08:00",
    "apiKeySource": "HEADER",
    "endpointConfiguration": {
        "types": [
            "EDGE"
        ]
    },
    "disableExecuteApiEndpoint": false,
    "rootResourceId": "efg567"
}
```

L'API creata ha un ID API che corrisponde a `abcd1234` e un ID risorsa root che corrisponde a `efg567`. Questi valori vengono utilizzati nella configurazione dell'API.

2. Quindi, viene aggiunta una risorsa figlio sotto la risorsa root e specificato il valore `RootResourceId` come valore della proprietà `parentId`. Nel seguente esempio si crea una risorsa `/pets` per l'API:

   JavaScript v3

```
import {APIGatewayClient,  CreateResourceCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new CreateResourceCommand({
    restApiId: 'abcd1234',
    parentId: 'efg567',
    pathPart: 'pets'
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("The '/pets' resource setup failed:\n", err)
}
})();
```

Una chiamata riuscita restituisce informazioni sulla risorsa in un output simile al seguente:

```
{
    "path": "/pets",
    "pathPart": "pets",
    "id": "aaa111",
    "parentId": "efg567'"
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.create_resource(
        restApiId='abcd1234',
        parentId='efg567',
        pathPart='pets'
    )
except botocore.exceptions.ClientError as error:
    logger.exception("The '/pets' resource setup failed: %s.", error)
    raise
attribute=["id","parentId", "pathPart", "path",]
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

Una chiamata riuscita restituisce informazioni sulla risorsa in un output simile al seguente:

```
{'id': 'aaa111', 'parentId': 'efg567', 'pathPart': 'pets', 'path': '/pets'}
```

AWS CLI

```
aws apigateway create-resource --rest-api-id abcd1234 \
  --region us-west-2 \
  --parent-id efg567 \
```

```
  --path-part pets
```

L'output di questo comando è il seguente:

```
{
    "id": "aaa111",
    "parentId": "efg567",
    "pathPart": "pets",
    "path": "/pets"
}
```

La risorsa `/pets` creata ha un ID risorsa che corrisponde a `aaa111`. Questo valore viene utilizzato nella configurazione dell'API.

3. Quindi, viene aggiunta una risorsa figlio sotto la risorsa `/pets`. Questa risorsa `/{petId}` ha un parametro di percorso per `{petId}`. Per fare di una parte di percorso un parametro di percorso, racchiudila tra parentesi graffe, `{ }`. Nel seguente esempio si crea una risorsa `/pets/{petId}` per l'API:

   JavaScript v3

```
import {APIGatewayClient,  CreateResourceCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new CreateResourceCommand({
    restApiId: 'abcd1234',
    parentId: 'aaa111',
    pathPart: '{petId}'
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("The '/pets/{petId}' resource setup failed:\n", err)
}
})();
```

Una chiamata riuscita restituisce informazioni sulla risorsa in un output simile al seguente:

```
{
    "path": "/pets/{petId}",
    "pathPart": "{petId}",
    "id": "bbb222",
    "parentId": "aaa111'"
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.create_resource(
        restApiId='abcd1234',
        parentId='aaa111',
        pathPart='{petId}'
    )
except botocore.exceptions.ClientError as error:
    logger.exception("The '/pets/{petId}' resource setup failed: %s.", error)
    raise
attribute=["id","parentId", "pathPart", "path",]
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

Una chiamata riuscita restituisce informazioni sulla risorsa in un output simile al seguente:

```
{'id': 'bbb222', 'parentId': 'aaa111', 'pathPart': '{petId}', 'path': '/pets/
{petId}'}
```

AWS CLI

```
aws apigateway create-resource --rest-api-id abcd1234 \
  --region us-west-2 \
  --parent-id aaa111 \
  --path-part '{petId}'
```

L'output di questo comando è il seguente:

```
{
    "id": "bbb222",
    "parentId": "aaa111",
    "path": "/pets/{petId}",
    "pathPart": "{petId}"
}
```

La risorsa `/pets/{petId}` creata ha un ID risorsa che corrisponde a `bbb222`. Questo valore viene utilizzato nella configurazione dell'API.

4. Nei due passaggi seguenti, vengono aggiunti metodi HTTP alle risorse. In questo tutorial, vengono impostati i metodi per avere accesso aperto impostando `authorization-type` su `NONE`. Per consentire solo agli utenti autenticati di chiamare il metodo, puoi utilizzare i ruoli e le policy IAM, un'autorizzazione Lambda (nota in precedenza come autorizzazioni ad hoc) o un pool di utenti di Amazon Cognito. Per ulteriori informazioni, consulta the section called “Controllo accessi”.

   Nel seguente esempio si aggiunge il metodo HTTP `GET` per la risorsa `/pets`:

   JavaScript v3

```
import {APIGatewayClient,  PutMethodCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new PutMethodCommand({
    restApiId: 'abcd1234',
    resourceId: 'aaa111',
    httpMethod: 'GET',
    authorizationType: 'NONE'
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("The 'GET /pets' method setup failed:\n", err)
}
})();
```

In caso di esito positivo, la chiamata restituisce il seguente output:

```
{
    "apiKeyRequired": false,
    "httpMethod": "GET",
    "authorizationType": "NONE"
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_method(
        restApiId='abcd1234',
        resourceId='aaa111',
        httpMethod='GET',
        authorizationType='NONE'
    )
except botocore.exceptions.ClientError as error:
    logger.exception("The 'GET /pets' method setup failed: %s", error)
    raise
attribute=["httpMethod","authorizationType","apiKeyRequired"]
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

In caso di esito positivo, la chiamata restituisce il seguente output:

```
{'httpMethod': 'GET', 'authorizationType': 'NONE', 'apiKeyRequired': False}
```

AWS CLI

```
aws apigateway put-method --rest-api-id abcd1234 \
  --resource-id aaa111 \
  --http-method GET \
  --authorization-type "NONE" \
```

```
  --region us-west-2
```

L'output di questo comando è il seguente:

```
{
    "httpMethod": "GET",
    "authorizationType": "NONE",
    "apiKeyRequired": false
}
```

5. Nel seguente esempio si aggiunge il metodo HTTP `GET` alla risorsa `/pets/{petId}` e si imposta la proprietà `requestParameters` per passare il valore `petId` fornito dal client al backend:

   JavaScript v3

```
import {APIGatewayClient,  PutMethodCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new PutMethodCommand({
    restApiId: 'abcd1234',
    resourceId: 'bbb222',
    httpMethod: 'GET',
    authorizationType: 'NONE'
    requestParameters: {
        "method.request.path.petId" : true
    }
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("The 'GET /pets/{petId}' method setup failed:\n", err)
}
})();
```

   In caso di esito positivo, la chiamata restituisce il seguente output:

```
{
    "apiKeyRequired": false,
    "httpMethod": "GET",
```

```
    "authorizationType": "NONE",
    "requestParameters": {
       "method.request.path.petId": true
    }
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_method(
        restApiId='abcd1234',
        resourceId='bbb222',
        httpMethod='GET',
        authorizationType='NONE',
        requestParameters={
            "method.request.path.petId": True
        }
    )
except botocore.exceptions.ClientError as error:
    logger.exception("The 'GET /pets/{petId}' method setup failed: %s", error)
    raise
attribute=["httpMethod","authorizationType","apiKeyRequired",
 "requestParameters" ]
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

In caso di esito positivo, la chiamata restituisce il seguente output:

```
{'httpMethod': 'GET', 'authorizationType': 'NONE', 'apiKeyRequired': False,
 'requestParameters': {'method.request.path.petId': True}}
```

AWS CLI

```
aws apigateway put-method --rest-api-id abcd1234 \
  --resource-id bbb222 --http-method GET \
```

```
  --authorization-type "NONE" \
  --region us-west-2 \
  --request-parameters method.request.path.petId=true
```

L'output di questo comando è il seguente:

```
{
    "httpMethod": "GET",
    "authorizationType": "NONE",
    "apiKeyRequired": false,
    "requestParameters": {
        "method.request.path.petId": true
    }
}
```

6. Utilizza l'esempio seguente per aggiungere la risposta del metodo 200 OK per il metodo `GET /pets`:

   JavaScript v3

```
import {APIGatewayClient,  PutMethodResponseCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new PutMethodResponseCommand({
    restApiId: 'abcd1234',
    resourceId: 'aaa111',
    httpMethod: 'GET',
    statusCode: '200'
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("Set up the 200 OK response for the 'GET /pets' method failed:
\n", err)
}
})();
```

   In caso di esito positivo, la chiamata restituisce il seguente output:

```
{
```

```
    "statusCode": "200"
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_method_response(
        restApiId='abcd1234',
        resourceId='aaa111',
        httpMethod='GET',
        statusCode='200'
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Set up the 200 OK response for the 'GET /pets' method
 failed %s.", error)
    raise
attribute=["statusCode"]
filtered_result ={key:result[key] for key in attribute}
logger.info(filtered_result)
```

In caso di esito positivo, la chiamata restituisce il seguente output:

```
{'statusCode': '200'}
```

AWS CLI

```
aws apigateway put-method-response --rest-api-id abcd1234 \
  --resource-id aaa111 --http-method GET \
  --status-code 200  --region us-west-2
```

L'output di questo comando è il seguente:

```
{
    "statusCode": "200"
```

```
}
```

7. Utilizza l'esempio seguente per aggiungere la risposta del metodo 200 OK per il metodo `GET /pets/{petId}`:

   JavaScript v3

   ```
   import {APIGatewayClient,  PutMethodResponseCommand } from "@aws-sdk/client-api-
   gateway";
   (async function (){
   const apig = new APIGatewayClient({region:"us-east-1"});
   const command = new PutMethodResponseCommand({
       restApiId: 'abcd1234',
       resourceId: 'bbb222',
       httpMethod: 'GET',
       statusCode: '200'
   });
   try {
       const results = await apig.send(command)
       console.log(results)
   } catch (err) {
       console.log("Set up the 200 OK response for the 'GET /pets/{petId}' method
    failed:\n", err)
   }
   })();
   ```

   In caso di esito positivo, la chiamata restituisce il seguente output:

   ```
   {
       "statusCode": "200"
   }
   ```

   Python

   ```
   import botocore
   import boto3
   import logging

   logger = logging.getLogger()
   apig = boto3.client('apigateway')

   try:
   ```

```
    result = apig.put_method_response(
        restApiId='abcd1234',
        resourceId='bbb222',
        httpMethod='GET',
        statusCode='200'
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Set up the 200 OK response for the 'GET /pets/{petId}'
 method failed %s.", error)
    raise
attribute=["statusCode"]
filtered_result ={key:result[key] for key in attribute}
logger.info(filtered_result)
```

In caso di esito positivo, la chiamata restituisce il seguente output:

```
{'statusCode': '200'}
```

AWS CLI

```
aws apigateway put-method-response --rest-api-id abcd1234 \
  --resource-id bbb222 --http-method GET \
  --status-code 200  --region us-west-2
```

L'output di questo comando è il seguente:

```
{
    "statusCode": "200"
}
```

8. Nel seguente esempio si configura un'integrazione per il metodo `GET /pets` con un endpoint HTTP. L'endpoint HTTP è `http://petstore-demo-endpoint.execute-api.com/petstore/pets`.

   JavaScript v3

```
import {APIGatewayClient,  PutIntegrationCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new PutIntegrationCommand({
```

```
    restApiId: 'abcd1234',
    resourceId: 'aaa111',
    httpMethod: 'GET',
    type: 'HTTP',
    integrationHttpMethod: 'GET',
    uri: 'http://petstore-demo-endpoint.execute-api.com/petstore/pets'
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("Set up the integration of the 'GET /pets' method of the API
 failed:\n", err)
}
})();
```

In caso di esito positivo, la chiamata restituisce il seguente output:

```
{
    "httpMethod": "GET",
    "passthroughBehavior": "WHEN_NO_MATCH",
    "cacheKeyParameters": [],
    "type": "HTTP",
    "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
    "cacheNamespace": "ccc333"
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_integration(
        restApiId='abcd1234',
        resourceId='aaa111',
        httpMethod='GET',
        type='HTTP',
```

```
        integrationHttpMethod='GET',
        uri='http://petstore-demo-endpoint.execute-api.com/petstore/pets'
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Set up the integration of the 'GET /' method of the API
 failed %s.", error)
    raise
attribute=["httpMethod","passthroughBehavior","cacheKeyParameters", "type",
 "uri", "cacheNamespace"]
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

In caso di esito positivo, la chiamata restituisce il seguente output:

```
{'httpMethod': 'GET', 'passthroughBehavior': 'WHEN_NO_MATCH',
 'cacheKeyParameters': [], 'type': 'HTTP', 'uri': 'http://petstore-demo-
endpoint.execute-api.com/petstore/pets', 'cacheNamespace': 'ccc333'}
```

AWS CLI

```
aws apigateway put-integration --rest-api-id abcd1234 \
  --resource-id aaa111 --http-method GET --type HTTP \
  --integration-http-method GET \
  --uri 'http://petstore-demo-endpoint.execute-api.com/petstore/pets' \
  --region us-west-2
```

L'output di questo comando è il seguente:

```
{
    "type": "HTTP",
    "httpMethod": "GET",
    "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
    "connectionType": "INTERNET",
    "passthroughBehavior": "WHEN_NO_MATCH",
    "timeoutInMillis": 29000,
    "cacheNamespace": "6sxz2j",
    "cacheKeyParameters": []
}
```

9. Nel seguente esempio si configura un'integrazione per il metodo `GET /pets/{petId}` con un endpoint HTTP. L'endpoint HTTP è `http://petstore-demo-endpoint.execute-`

`api.com/petstore/pets/{id}`. In questa fase viene mappato il parametro di percorso `petId` al parametro di percorso dell'endpoint di integrazione `id`.

JavaScript v3

```
import {APIGatewayClient,  PutIntegrationCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new PutIntegrationCommand({
    restApiId: 'abcd1234',
    resourceId: 'bbb222',
    httpMethod: 'GET',
    type: 'HTTP',
    integrationHttpMethod: 'GET',
    uri: 'http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}'
    requestParameters: {
        "integration.request.path.id": "method.request.path.petId"
     }
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("Set up the integration of the 'GET /pets/{petId}' method of the
 API failed:\n", err)
}
})();
```

In caso di esito positivo, la chiamata restituisce il seguente output:

```
{
    "httpMethod": "GET",
    "passthroughBehavior": "WHEN_NO_MATCH",
    "cacheKeyParameters": [],
    "type": "HTTP",
    "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}",
    "cacheNamespace": "ddd444",
    "requestParameters": {
       "integration.request.path.id": "method.request.path.petId"
    }
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_integration(
        restApiId='ieps9b05sf',
        resourceId='t8zeb4',
        httpMethod='GET',
        type='HTTP',
        integrationHttpMethod='GET',
        uri='http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}',
        requestParameters={
            "integration.request.path.id": "method.request.path.petId"
        }
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Set up the integration of the 'GET /pets/{petId}' method
 of the API failed %s.", error)
    raise
attribute=["httpMethod","passthroughBehavior","cacheKeyParameters", "type",
 "uri", "cacheNamespace", "requestParameters"]
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

In caso di esito positivo, la chiamata restituisce il seguente output:

```
{'httpMethod': 'GET', 'passthroughBehavior': 'WHEN_NO_MATCH',
 'cacheKeyParameters': [], 'type': 'HTTP', 'uri': 'http://petstore-
demo-endpoint.execute-api.com/petstore/pets/{id}', 'cacheNamespace':
 'ddd444', 'requestParameters': {'integration.request.path.id':
 'method.request.path.petId'}}}
```

AWS CLI

```
aws apigateway put-integration --rest-api-id abcd1234 \
```

```
  --resource-id bbb222 --http-method GET --type HTTP \
  --integration-http-method GET \
  --uri 'http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}' \
  --request-parameters
 '{"integration.request.path.id":"method.request.path.petId"}' \
  --region us-west-2
```

L'output di questo comando è il seguente:

```
{
    "type": "HTTP",
    "httpMethod": "GET",
    "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}",
    "connectionType": "INTERNET",
    "requestParameters": {
        "integration.request.path.id": "method.request.path.petId"
    },
    "passthroughBehavior": "WHEN_NO_MATCH",
    "timeoutInMillis": 29000,
    "cacheNamespace": "rjkmth",
    "cacheKeyParameters": []
}
```

10. Nel seguente esempio si aggiunge la risposta di integrazione per l'integrazione `GET /pets`:

    JavaScript v3

```
import {APIGatewayClient,  PutIntegrationResponseCommand } from "@aws-sdk/
client-api-gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new PutIntegrationResponseCommand({
    restApiId: 'abcd1234',
    resourceId: 'aaa111',
    httpMethod: 'GET',
    statusCode: '200',
    selectionPattern: ''
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
```

```
    console.log("The 'GET /pets' method integration response setup failed:\n",
 err)
}
})();
```

In caso di esito positivo, la chiamata restituisce il seguente output:

```
{
    "selectionPattern": "",
    "statusCode": "200"
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_integration_response(
        restApiId='abcd1234',
        resourceId='aaa111',
        httpMethod='GET',
        statusCode='200',
        selectionPattern='',
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Set up the integration response of the 'GET /pets' method
 of the API failed: %s", error)
    raise
attribute=["selectionPattern","statusCode"]
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

In caso di esito positivo, la chiamata restituisce il seguente output:

```
{'selectionPattern': "", 'statusCode': '200'}
```

AWS CLI

```
aws apigateway put-integration-response --rest-api-id abcd1234 \
  --resource-id aaa111 --http-method GET \
  --status-code 200 --selection-pattern ""  \
  --region us-west-2
```

L'output di questo comando è il seguente:

```
{
    "statusCode": "200",
    "selectionPattern": ""
}
```

11. Nel seguente esempio si aggiunge la risposta di integrazione per l'integrazione `GET /pets/{petId}`:

    JavaScript v3

    ```
    import {APIGatewayClient,  PutIntegrationResponseCommand } from "@aws-sdk/
    client-api-gateway";
    (async function (){
    const apig = new APIGatewayClient({region:"us-east-1"});
    const command = new PutIntegrationResponseCommand({
        restApiId: 'abcd1234',
        resourceId: 'bbb222',
        httpMethod: 'GET',
        statusCode: '200',
        selectionPattern: ''
    });
    try {
        const results = await apig.send(command)
        console.log(results)
    } catch (err) {
        console.log("The 'GET /pets/{petId}' method integration response setup
     failed:\n", err)
    }
    })();
    ```

    In caso di esito positivo, la chiamata restituisce il seguente output:

```
{
    "selectionPattern": "",
    "statusCode": "200"
}
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_integration_response(
        restApiId='abcd1234',
        resourceId='bbb222',
        httpMethod='GET',
        statusCode='200',
        selectionPattern='',
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Set up the integration response of the 'GET /pets/{petId}'
 method of the API failed: %s", error)
    raise
attribute=["selectionPattern","statusCode"]
filtered_result ={key:result[key] for key in attribute}
print(filtered_result)
```

In caso di esito positivo, la chiamata restituisce il seguente output:

```
{'selectionPattern': "", 'statusCode': '200'}
```

AWS CLI

```
aws apigateway put-integration-response --rest-api-id abcd1234 \
  --resource-id bbb222 --http-method GET
  --status-code 200 --selection-pattern ""
  --region us-west-2
```

L'output di questo comando è il seguente:

```
{
    "statusCode": "200",
    "selectionPattern": ""
}
```

Dopo aver creato la risposta di integrazione, l'API può interrogare gli animali domestici disponibili sul PetStore sito Web e visualizzare un singolo animale domestico con un identificatore specificato. Affinché l'API possa essere chiamata dai clienti, devi implementarla. Ti consigliamo di testare l'API prima di implementarla.

12. Nel seguente esempio si testa il metodo `GET /pets`:

JavaScript v3

```
import {APIGatewayClient,  TestInvokeMethodCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new TestInvokeMethodCommand({
    restApiId: 'abcd1234',
    resourceId: 'aaa111',
    httpMethod: 'GET',
    pathWithQueryString: '/',
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("The test on 'GET /pets' method failed:\n", err)
}
})();
```

Python

```
import botocore
import boto3
import logging
```

```
logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.test_invoke_method(
        restApiId='abcd1234',
        resourceId='aaa111',
        httpMethod='GET',
        pathWithQueryString='/',
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Test invoke method on 'GET /pets' failed: %s", error)
    raise
print(result)
```

AWS CLI

```
aws apigateway test-invoke-method --rest-api-id abcd1234 /
  --resource-id aaa111 /
  --http-method GET /
  --path-with-query-string '/'
```

13. Nel seguente esempio si testa il metodo `GET /pets/{petId}` con un valore `petId` che corrisponde a 3:

    JavaScript v3

```
import {APIGatewayClient,  TestInvokeMethodCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new TestInvokeMethodCommand({
    restApiId: 'abcd1234',
    resourceId: 'bbb222',
    httpMethod: 'GET',
    pathWithQueryString: '/pets/3',
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("The test on 'GET /pets/{petId}' method failed:\n", err)
}
```

```
})();
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.test_invoke_method(
        restApiId='abcd1234',
        resourceId='bbb222',
        httpMethod='GET',
        pathWithQueryString='/pets/3',
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Test invoke method on 'GET /pets/{petId}' failed: %s",
 error)
    raise
print(result)
```

AWS CLI

```
aws apigateway test-invoke-method --rest-api-id abcd1234 /
  --resource-id bbb222 /
  --http-method GET /
  --path-with-query-string '/pets/3'
```

Dopo aver testato l'API correttamente, puoi implementarla in una fase.

14. Nel seguente esempio si implementa l'API in una fase denominata `test`. Quando implementi l'API in una fase, i chiamanti dell'API possono invocare l'API.

    JavaScript v3

```
import {APIGatewayClient,  CreateDeploymentCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
```

```
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new CreateDeploymentCommand({
    restApiId: 'abcd1234',
    stageName: 'test',
    stageDescription: 'test deployment'
});
try {
    const results = await apig.send(command)
    console.log("Deploying API succeeded\n", results)
} catch (err) {
    console.log("Deploying API failed:\n", err)
}
})();
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.create_deployment(
        restApiId='ieps9b05sf',
        stageName='test',
        stageDescription='my test stage',
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Error deploying stage  %s.", error)
    raise
print('Deploying API succeeded')
print(result)
```

AWS CLI

```
aws apigateway create-deployment --rest-api-id abcd1234 \
  --region us-west-2 \
  --stage-name test \
  --stage-description 'Test stage' \
  --description 'First deployment'
```

L'output di questo comando è il seguente:

```
{
    "id": "ab1c1d",
    "description": "First deployment",
    "createdDate": "2022-12-15T08:44:13-08:00"
}
```

L'API può ora essere chiamata dai clienti. Puoi testare l'API inserendo l'URL `https://abcd1234.execute-api.us-west-2.amazonaws.com/test/pets` in un browser e sostituendo `abcd1234` con l'identificativo dell'API.

Per altri esempi su come creare o aggiornare un'API utilizzando gli AWS SDK o il AWS CLI, consulta Azioni per API Gateway utilizzando AWS gli SDK.

Automatizzazione della configurazione di un'API

Invece di creare la tua API passo dopo passo, puoi automatizzare la creazione e la pulizia delle AWS risorse utilizzando OpenAPI o Terraform per creare la tua API. CloudFormation

Definizione di OpenAPI 3.0

Puoi importare una definizione OpenAPI in Gateway API. Per ulteriori informazioni, consulta the section called “OpenAPI”.

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "Simple PetStore (OpenAPI)",
    "description" : "Demo API created using OpenAPI",
    "version" : "2024-05-24T20:39:34Z"
  },
  "servers" : [ {
    "url" : "{basePath}",
    "variables" : {
      "basePath" : {
        "default" : "Prod"
      }
    }
  } ],
```

```
  "paths" : {
    "/pets" : {
      "get" : {
        "responses" : {
          "200" : {
            "description" : "200 response",
            "content" : { }
          }
        },
        "x-amazon-apigateway-integration" : {
          "type" : "http",
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
          "responses" : {
            "default" : {
              "statusCode" : "200"
            }
          },
          "passthroughBehavior" : "when_no_match",
          "timeoutInMillis" : 29000
        }
      }
    },
    "/pets/{petId}" : {
      "get" : {
        "parameters" : [ {
          "name" : "petId",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "content" : { }
          }
        },
        "x-amazon-apigateway-integration" : {
          "type" : "http",
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}",
          "responses" : {
```

```
            "default" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.path.id" : "method.request.path.petId"
          },
          "passthroughBehavior" : "when_no_match",
          "timeoutInMillis" : 29000
        }
      }
    }
  },
  "components" : { }
}
```

AWS CloudFormation modello

Per distribuire il CloudFormation modello, consulta Creazione di uno stack sulla console. AWS CloudFormation

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Name: Simple PetStore (AWS CloudFormation)
  PetsResource:
    Type: 'AWS::ApiGateway::Resource'
    Properties:
      RestApiId: !Ref Api
      ParentId: !GetAtt Api.RootResourceId
      PathPart: 'pets'
  PetIdResource:
    Type: 'AWS::ApiGateway::Resource'
    Properties:
      RestApiId: !Ref Api
      ParentId: !Ref PetsResource
      PathPart: '{petId}'
  PetsMethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
```

```
      ResourceId: !Ref PetsResource
      HttpMethod: GET
      AuthorizationType: NONE
      Integration:
        Type: HTTP
        IntegrationHttpMethod: GET
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
        IntegrationResponses:
          - StatusCode: '200'
      MethodResponses:
        - StatusCode: '200'
  PetIdMethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref PetIdResource
      HttpMethod: GET
      AuthorizationType: NONE
      RequestParameters:
        method.request.path.petId: true
      Integration:
        Type: HTTP
        IntegrationHttpMethod: GET
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}
        RequestParameters:
          integration.request.path.id: method.request.path.petId
        IntegrationResponses:
          - StatusCode: '200'
      MethodResponses:
        - StatusCode: '200'
  ApiDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn:
      - PetsMethodGet
    Properties:
      RestApiId: !Ref Api
      StageName: Prod
Outputs:
  ApiRootUrl:
    Description: Root Url of the API
    Value: !Sub 'https://${Api}.execute-api.${AWS::Region}.amazonaws.com/Prod'
```

Configurazione Terraform

Per ulteriori informazioni su Terraform, consulta Terraform.

```
provider "aws" {
  region = "us-east-1" # Update with your desired region
}
resource "aws_api_gateway_rest_api" "Api" {
  name        = "Simple PetStore (Terraform)"
  description = "Demo API created using Terraform"
}
resource "aws_api_gateway_resource" "petsResource"{
    rest_api_id = aws_api_gateway_rest_api.Api.id
    parent_id = aws_api_gateway_rest_api.Api.root_resource_id
    path_part = "pets"
}
resource "aws_api_gateway_resource" "petIdResource"{
    rest_api_id = aws_api_gateway_rest_api.Api.id
    parent_id = aws_api_gateway_resource.petsResource.id
    path_part = "{petId}"
}
resource "aws_api_gateway_method" "petsMethodGet" {
  rest_api_id   = aws_api_gateway_rest_api.Api.id
  resource_id   = aws_api_gateway_resource.petsResource.id
  http_method   = "GET"
  authorization = "NONE"
}


resource "aws_api_gateway_method_response" "petsMethodResponseGet" {
    rest_api_id = aws_api_gateway_rest_api.Api.id
    resource_id = aws_api_gateway_resource.petsResource.id
    http_method = aws_api_gateway_method.petsMethodGet.http_method
    status_code ="200"
}

resource "aws_api_gateway_integration" "petsIntegration" {
  rest_api_id = aws_api_gateway_rest_api.Api.id
  resource_id = aws_api_gateway_resource.petsResource.id
  http_method = aws_api_gateway_method.petsMethodGet.http_method
  type        = "HTTP"

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

```
  integration_http_method = "GET"
  depends_on              = [aws_api_gateway_method.petsMethodGet]
}

resource "aws_api_gateway_integration_response" "petsIntegrationResponse" {
    rest_api_id = aws_api_gateway_rest_api.Api.id
    resource_id = aws_api_gateway_resource.petsResource.id
    http_method = aws_api_gateway_method.petsMethodGet.http_method
    status_code = aws_api_gateway_method_response.petsMethodResponseGet.status_code
}

resource "aws_api_gateway_method" "petIdMethodGet" {
    rest_api_id   = aws_api_gateway_rest_api.Api.id
    resource_id   = aws_api_gateway_resource.petIdResource.id
    http_method   = "GET"
    authorization = "NONE"
    request_parameters = {"method.request.path.petId" = true}
}

resource "aws_api_gateway_method_response" "petIdMethodResponseGet" {
    rest_api_id = aws_api_gateway_rest_api.Api.id
    resource_id = aws_api_gateway_resource.petIdResource.id
    http_method = aws_api_gateway_method.petIdMethodGet.http_method
    status_code ="200"
}


resource "aws_api_gateway_integration" "petIdIntegration" {
    rest_api_id = aws_api_gateway_rest_api.Api.id
    resource_id = aws_api_gateway_resource.petIdResource.id
    http_method = aws_api_gateway_method.petIdMethodGet.http_method
    type        = "HTTP"
    uri                     = "http://petstore-demo-endpoint.execute-api.com/petstore/
pets/{id}"
    integration_http_method = "GET"
    request_parameters = {"integration.request.path.id" = "method.request.path.petId"}
    depends_on              = [aws_api_gateway_method.petIdMethodGet]
}

resource "aws_api_gateway_integration_response" "petIdIntegrationResponse" {
    rest_api_id = aws_api_gateway_rest_api.Api.id
    resource_id = aws_api_gateway_resource.petIdResource.id
    http_method = aws_api_gateway_method.petIdMethodGet.http_method
    status_code = aws_api_gateway_method_response.petIdMethodResponseGet.status_code
```

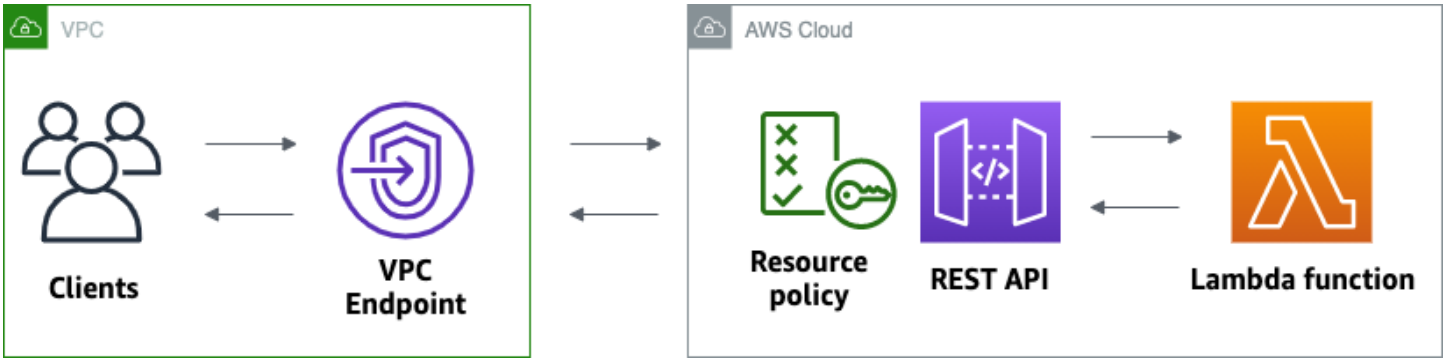
```
}


resource "aws_api_gateway_deployment" "Deployment" {
  rest_api_id = aws_api_gateway_rest_api.Api.id
  depends_on  =
 [aws_api_gateway_integration.petsIntegration,aws_api_gateway_integration.petIdIntegration ]
}
resource "aws_api_gateway_stage" "Stage" {
  stage_name    = "Prod"
  rest_api_id   = aws_api_gateway_rest_api.Api.id
  deployment_id = aws_api_gateway_deployment.Deployment.id
}
```

Tutorial: creazione di una REST API privata

In questo tutorial è possibile creare un'API REST privata. I client possono accedere all'API solo dall'interno del tuo Amazon VPC. L'API è isolata dall'internet pubblico: si tratta di un requisito di sicurezza comune.

Il completamento di questa esercitazione richiede circa 30 minuti. Innanzitutto, utilizzi un CloudFormation modello per creare un Amazon VPC, un endpoint VPC, una AWS Lambda funzione e avvia un'istanza Amazon EC2 che utilizzerai per testare la tua API. Successivamente, si utilizza Console di gestione AWS per creare un'API privata e allegare una politica delle risorse che consenta l'accesso solo dall'endpoint VPC. Infine, si esegue il dell'API.



Per completare questo tutorial, sono necessari un AWS account e un AWS Identity and Access Management utente con accesso alla console. Per ulteriori informazioni, consulta Prerequisiti.

In questo tutorial utilizzerai Console di gestione AWS. Per un CloudFormation modello che crea questa API e tutte le risorse correlate, vedi template.yaml.

Argomenti

- Fase 1: Creare dipendenze
- Fase 2: creare un'API privata
- Fase 3: Creare un metodo e un'integrazione
- Fase 4: Allegare una policy sulle risorse
- Fase 5: distribuzione dell'API
- Fase 6: verificare che l'API non sia accessibile pubblicamente
- Fase 7: connettersi a un'istanza nel VPC e richiamare l'API
- Fase 8: Pulizia
- Passaggi successivi: automatizza con CloudFormation

Fase 1: Creare dipendenze

Scarica e decomprimi questo modello. CloudFormation Utilizza il modello per creare tutte le dipendenze per la tua API privata, tra cui un Amazon VPC, un endpoint VPC e una funzione Lambda utilizzata come back-end della tua API. È possibile creare l'API privata in un secondo momento.

Per creare un CloudFormation stack

1. Apri la CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation.
2. Scegliere Create stack (Crea stack), quindi With new resources (standard) (Con nuove risorse (standard)).
3. In Specificare modello, scegliere Carica un file modello.
4. Selezionare il modello scaricato.
5. Scegli Next (Successivo).
6. Per Stack name (Nome stack), inserire **`private-api-tutorial`**, quindi scegliere Next (Avanti).
7. Per Configure stack options (Configura opzioni di stack), scegliere Next (Successivo).
8. Per quanto riguarda le funzionalità, riconosci che CloudFormation puoi creare risorse IAM nel tuo account.
9. Scegli Avanti, quindi scegli Invia.

CloudFormation esegue il provisioning delle dipendenze per l'API, operazione che può richiedere alcuni minuti. Quando lo stato del tuo CloudFormation stack è CREATE_COMPLETE, scegli Outputs. Annotare l'ID endpoint VPC. È necessario per le fasi successive di questo tutorial.

Fase 2: creare un'API privata

Si crea un'API privata per consentire solo ai client all'interno del VPC di accedervi.

Per creare un'API privata

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli Create API (Crea API), quindi per API REST, scegli Build (Crea).
3. Per API name (Nome API), immettere **`private-api-tutorial`**.
4. Per Tipo di endpoint API scegli Privato.
5. Per gli ID degli endpoint VPC, inserisci l'ID dell'endpoint VPC dagli output del tuo stack. CloudFormation
6. Per Tipo di indirizzo IP, scegli Dualstack.
7. Seleziona Create API (Crea API).

Fase 3: Creare un metodo e un'integrazione

Si crea un metodo `GET` e un'integrazione Lambda per gestire le richieste `GET` all'API. Quando un client chiama l'API, API Gateway invia la richiesta alla funzione Lambda creata nella fase 1 e restituisce una risposta al client.

Per creare un metodo e un'integrazione

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere l'API.
3. Scegli Crea metodo.
4. Per Tipo di metodo seleziona `GET`.
5. Per Tipo di integrazione seleziona Funzione Lambda.
6. Attiva l'opzione Integrazione proxy Lambda. Con un'integrazione proxy Lambda, API Gateway invia un evento a Lambda con una struttura definita e trasforma la risposta dalla funzione Lambda in una risposta HTTP.

7. Per la funzione Lambda, scegli la funzione che hai creato con il CloudFormation modello nel passaggio 1. Il nome della funzione inizia con **`private-api-tutorial`**.
8. Scegli Crea metodo.

Fase 4: Allegare una policy sulle risorse

Si allega una policy delle risorse all'API che consente ai client di richiamare l'API solo tramite l'endpoint VPC. Per limitare ulteriormente l'accesso all'API, è inoltre possibile configurare una policy degli endpoint VPC per l'endpoint VPC, ma per questo tutorial non è necessario.

Per allegare una policy sulle risorse

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere l'API.
3. Scegli Policy delle risorse, quindi seleziona Crea policy.
4. Immetti la seguente policy. Sostituiscilo *`vpceID`* con l'ID dell'endpoint VPC dagli output del tuo stack. CloudFormation

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "execute-api:/*",
            "Condition": {
                "StringNotEquals": {
                    "aws:sourceVpce": "vpce-abcd1234"
                }
            }
        },
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "execute-api:/*"
```

```
        }
    ]
}
```

5. Scegli Save changes (Salva modifiche).

Fase 5: distribuzione dell'API

Successivamente, distribuisci l'API per renderla disponibile ai client nel tuo Amazon VPC.

Per distribuire un'API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere l'API.
3. Seleziona Deploy API (Distribuisci API).
4. In Fase, seleziona Nuova fase.
5. In Stage name (Nome fase) immettere **`test`**.
6. (Facoltativo) In Descrizione, immetti una descrizione.
7. Selezionare Deploy (Distribuisci).

Ora siamo pronti per testare l'API.

Fase 6: verificare che l'API non sia accessibile pubblicamente

Utilizzare `curl` per verificare che non sia possibile richiamare l'API dall'esterno dell'Amazon VPC.

Per testare l'API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere l'API.
3. Nel pannello di navigazione principale scegli Fasi, quindi seleziona la fase test.
4. In Dettagli fase, scegli l'icona Copia per copiare l'URL di richiamo dell'API. L'URL è simile a `https://abcdef123.execute-api.us-west-2.amazonaws.com/test`. L'endpoint VPC creato nella fase 1 ha il DNS privato abilitato, quindi è possibile utilizzare l'URL fornito per richiamare l'API.
5. Usa curl per tentare di richiamare l'API dall'esterno del VPC.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/test
```

Curl indica che l'endpoint dell'API non può essere risolto. Se ricevi una risposta diversa, torna alla fase 2 e assicurati di scegliere Private (Privato) per il tipo di endpoint dell'API.

```
curl: (6) Could not resolve host: abcdef123.execute-api.us-west-2.amazonaws.com/
test
```

Successivamente, connettiti a un'istanza Amazon EC2 nel VPC per richiamare l'API.

Fase 7: connettersi a un'istanza nel VPC e richiamare l'API

Successivamente, testa l'API dall'interno del VPC Amazon. Per accedere alla tua API privata, connettiti a un'istanza Amazon EC2 nel tuo VPC e poi usa curl per richiamare l'API. Per connettersi all'istanza nel browser, si utilizza Systems Manager Session Manger.

Per testare l'API

1. Apri la console Amazon EC2 all'indirizzo https://console.aws.amazon.com/ec2/.
2. Seleziona Instances (Istanze).
3. Scegliere l'istanza denominata private-api-tutorial creata con il modello CloudFormation nella fase 1.
4. Scegliere Connect (Connetti), quindi Session Manager (Gestore di sessione).
5. Scegliere Connect per avviare una sessione basata su browser nell'istanza.
6. Nella sessione di Session Manager, usa curl per richiamare l'API. Puoi richiamare l'API perché stai utilizzando un'istanza nel tuo Amazon VPC.

   ```
   curl https://abcdef123.execute-api.us-west-2.amazonaws.com/test
   ```

   Verificare di ottenere la risposta `Hello from Lambda!`.



Hai creato un'API accessibile solo dall'interno del tuo VPC Amazon e hai verificato che funzioni.

Fase 8: Pulizia

Per evitare costi non necessari, eliminare le risorse create nell'ambito di questo tutorial. I seguenti passaggi eliminano la tua API REST e il tuo CloudFormation stack.

Per eliminare un'API REST

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nella pagina API, selezionare un'API. Scegli Azioni API, seleziona Elimina e quindi conferma la scelta.

Per eliminare un CloudFormation stack

1. Apri la CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation.
2. Seleziona il tuo CloudFormation stack.
3. Scegli Elimina e conferma la tua scelta.

Passaggi successivi: automatizza con CloudFormation

Puoi automatizzare la creazione e la pulizia di tutte le AWS risorse coinvolte in questo tutorial. Per un modello CloudFormation di esempio completo, consulta template.yaml.

Tutorial sull'API HTTP di Amazon API Gateway

I seguenti tutorial forniscono esercizi pratici per aiutarti a usare le API HTTP di API Gateway.

Argomenti

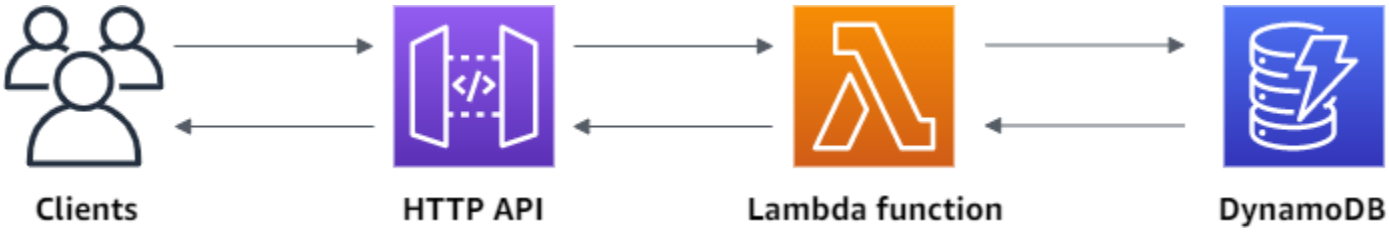
- Tutorial: creazione di un'API HTTP CRUD con Lambda e DynamoDB
- Tutorial: creazione di un'API HTTP con un'integrazione privata con un servizio Amazon ECS

Tutorial: creazione di un'API HTTP CRUD con Lambda e DynamoDB

In questo tutorial crei un'API serverless che crea, legge, aggiorna ed elimina voci da una tabella DynamoDB. DynamoDB è un servizio di database NoSQL interamente gestito che combina prestazioni elevate e prevedibili con una scalabilità continua. Questo tutorial richiede circa 30 minuti e può essere eseguito all'interno del piano gratuito di AWS.

Innanzitutto, si crea una tabella DynamoDB utilizzando la console DynamoDB. Quindi crei una funzione Lambda utilizzando la AWS Lambda console. Successivamente, è possibile creare un'API HTTP utilizzando la console API Gateway. Infine, si esegue il dell'API.

Quando si richiama l'API HTTP, API Gateway instrada la richiesta alla funzione Lambda. La funzione Lambda interagisce con DynamoDB e restituisce una risposta ad API Gateway. La console API Gateway restituisce quindi una risposta all'utente.



Per completare questo esercizio, sono necessari un AWS account e un AWS Identity and Access Management utente con accesso alla console. Per ulteriori informazioni, consulta Prerequisiti.

In questo tutorial utilizzerai Console di gestione AWS. Per un AWS SAM modello che crea questa API e tutte le risorse correlate, vedi `template.yaml`.

Argomenti

- Fase 1: creazione di una tabella DynamoDB
- Fase 2: creazione di una funzione Lambda
- Fase 3: creazione un'API HTTP
- Fase 4: creazione di route
- Fase 5: creazione di un'integrazione

- Fase 6: collega la tua integrazione alle route
- Fase 7: test dell'API
- Fase 8: Pulizia
- Passaggi successivi: automatizza con AWS SAM or CloudFormation

Fase 1: creazione di una tabella DynamoDB

Si utilizza una tabella DynamoDB per memorizzare i dati per l'API.

Ogni voce ha un ID univoco, che usiamo come chiave di partizione per la tabella.

Per creare una tabella DynamoDB

1. Apri la console DynamoDB all'indirizzo. https://console.aws.amazon.com/dynamodb/
2. Seleziona Create table (Crea tabella).
3. Nel campo Table name (Nome tabella) immetti **`http-crud-tutorial-items`**.
4. In Partition key (Chiave di partizione), inserisci **`id`**.
5. Scegliere Create table (Crea tabella).

Fase 2: creazione di una funzione Lambda

Crei una funzione Lambda per il back-end della tua API. Questa funzione Lambda crea, legge, aggiorna ed elimina voci da DynamoDB. La funzione utilizza gli eventi da API Gateway per determinare come interagire con DynamoDB. Per semplicità questo tutorial utilizza una singola funzione Lambda. Come best practice, dovresti creare funzioni separate per ogni route. Per ulteriori informazioni, consulta Il monolite Lambda.

Come creare una funzione Lambda

1. Accedi alla console Lambda all'indirizzo. https://console.aws.amazon.com/lambda
2. Selezionare Create function (Crea funzione).
3. Nel campo Function name (Nome funzione), immettere **`http-crud-tutorial-function`**.
4. Per Runtime, scegli l'ultimo runtime supportato Node.jso Python.
5. In Permissions (Autorizzazioni) scegli Change default execution role (Cambia ruolo di esecuzione predefinito).
6. Seleziona Crea un nuovo ruolo dai modelli di AWS policy.

7. In Role name (Nome ruolo), immettere **`http-crud-tutorial-role`**.
8. In Policy templates (Modelli di policy) scegli **`Simple microservice permissions`**. Questa policy concede alla funzione Lambda l'autorizzazione per interagire con DynamoDB.

> **Note**
>
> Per semplicità, questo tutorial utilizza una policy gestita. Come best practice, dovresti creare la tua policy IAM per concedere le autorizzazioni minime richieste.

9. Selezionare Create function (Crea funzione).
10. Aprire la funzione Lambda nell'editor di codice della console e sostituirne il contenuto con il codice seguente. Scegli Deploy (Distribuisci) per aggiornare la funzione.

Node.js

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  DynamoDBDocumentClient,
  ScanCommand,
  PutCommand,
  GetCommand,
  DeleteCommand,
} from "@aws-sdk/lib-dynamodb";

const client = new DynamoDBClient({});

const dynamo = DynamoDBDocumentClient.from(client);

const tableName = "http-crud-tutorial-items";

export const handler = async (event, context) => {
  let body;
  let statusCode = 200;
  const headers = {
    "Content-Type": "application/json",
  };

  try {
    switch (event.routeKey) {
      case "DELETE /items/{id}":
        await dynamo.send(
```

```
          new DeleteCommand({
            TableName: tableName,
            Key: {
              id: event.pathParameters.id,
            },
          })
        );
        body = `Deleted item ${event.pathParameters.id}`;
        break;
      case "GET /items/{id}":
        body = await dynamo.send(
          new GetCommand({
            TableName: tableName,
            Key: {
              id: event.pathParameters.id,
            },
          })
        );
        body = body.Item;
        break;
      case "GET /items":
        body = await dynamo.send(
          new ScanCommand({ TableName: tableName })
        );
        body = body.Items;
        break;
      case "PUT /items":
        let requestJSON = JSON.parse(event.body);
        await dynamo.send(
          new PutCommand({
            TableName: tableName,
            Item: {
              id: requestJSON.id,
              price: requestJSON.price,
              name: requestJSON.name,
            },
          })
        );
        body = `Put item ${requestJSON.id}`;
        break;
      default:
        throw new Error(`Unsupported route: "${event.routeKey}"`);
    }
  } catch (err) {
```

```
    statusCode = 400;
    body = err.message;
  } finally {
    body = JSON.stringify(body);
  }

  return {
    statusCode,
    body,
    headers,
  };
};
```

Python

```
import json
import boto3
from decimal import Decimal

client = boto3.client('dynamodb')
dynamodb = boto3.resource("dynamodb")
table = dynamodb.Table('http-crud-tutorial-items')
tableName = 'http-crud-tutorial-items'


def lambda_handler(event, context):
    print(event)
    body = {}
    statusCode = 200
    headers = {
        "Content-Type": "application/json"
    }

    try:
        if event['routeKey'] == "DELETE /items/{id}":
            table.delete_item(
                Key={'id': event['pathParameters']['id']})
            body = 'Deleted item ' + event['pathParameters']['id']
        elif event['routeKey'] == "GET /items/{id}":
            body = table.get_item(
                Key={'id': event['pathParameters']['id']})
            body = body["Item"]
            responseBody = [
```

```
                {'price': float(body['price']), 'id': body['id'], 'name':
body['name']}]
            body = responseBody
        elif event['routeKey'] == "GET /items":
            body = table.scan()
            body = body["Items"]
            print("ITEMS----")
            print(body)
            responseBody = []
            for items in body:
                responseItems = [
                    {'price': float(items['price']), 'id': items['id'], 'name':
items['name']}]
                responseBody.append(responseItems)
            body = responseBody
        elif event['routeKey'] == "PUT /items":
            requestJSON = json.loads(event['body'])
            table.put_item(
                Item={
                    'id': requestJSON['id'],
                    'price': Decimal(str(requestJSON['price'])),
                    'name': requestJSON['name']
                })
            body = 'Put item ' + requestJSON['id']
    except KeyError:
        statusCode = 400
        body = 'Unsupported route: ' + event['routeKey']
    body = json.dumps(body)
    res = {
        "statusCode": statusCode,
        "headers": {
            "Content-Type": "application/json"
        },
        "body": body
    }
    return res
```

Fase 3: creazione un'API HTTP

L'API HTTP fornisce un endpoint HTTP per la funzione Lambda. In questa fase si crea un'API vuota. Nelle fasi seguenti si configurano route e integrazioni per connettere l'API e la funzione Lambda.

Per creare un'API HTTP

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli Create API (Crea API), quindi per API HTTP, scegli Build (Crea).
3. Per API name (Nome API), immettere **`http-crud-tutorial-api`**.
4. Per Tipo di indirizzo IP, selezionare IPv4.
5. Scegli Next (Successivo).
6. Per Configura route, scegli Avanti per ignorare la creazione della route. È possibile creare route in un secondo momento.
7. Esamina la fase creata da API Gateway e quindi scegli Next (Avanti).
8. Seleziona Crea.

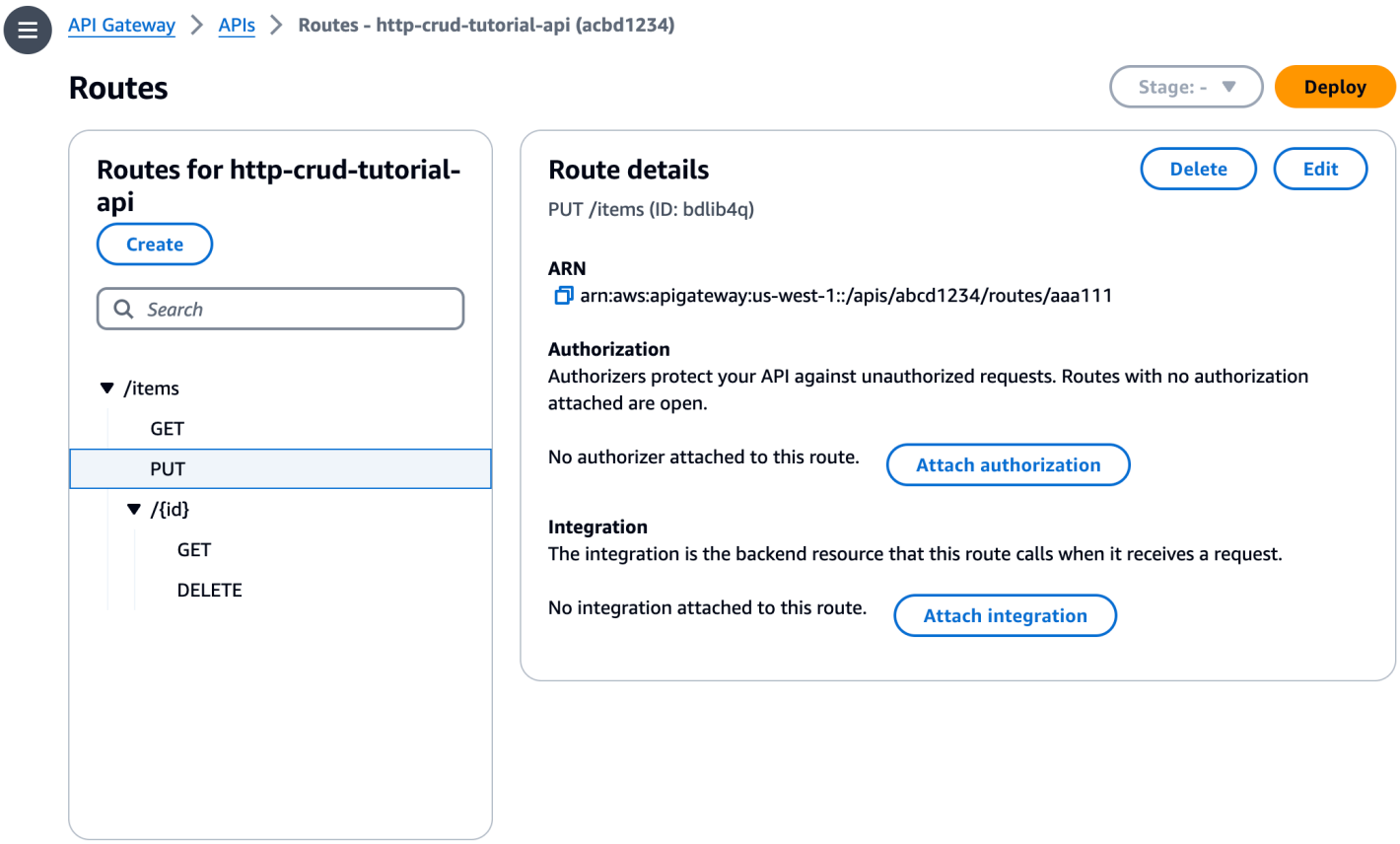
Fase 4: creazione di route

Le route inviano le richieste API in entrata alle risorse di back-end. Le route sono costituite da due parti: un metodo HTTP e un percorso della risorsa, ad esempi, `GET /items`. Per questa API di esempio, creiamo quattro route:

- `GET /items/{id}`
- `GET /items`
- `PUT /items`
- `DELETE /items/{id}`

Per creare route

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere l'API.
3. Scegliere Percorsi.
4. Seleziona Crea.
5. Per Method (Metodo) seleziona **`GET`**.
6. Per il percorso, immetti **`/items/{id}`**. Il `{id}` alla fine del percorso è un parametro che API Gateway recupera dal percorso della richiesta quando essa viene effettuata da un client.
7. Seleziona Crea.
8. Ripeti le fasi da 4 a 7 per `GET /items`, `DELETE /items/{id}` e `PUT /items`.



Fase 5: creazione di un'integrazione

Si crea un'integrazione per connettere una route alle risorse di back-end. Per questa API di esempio, si crea un'integrazione Lambda che si utilizza per tutte le route.

Per creare un'integrazione

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere l'API.
3. Scegli Integrations (Integrazioni).
4. Scegli Manage integrations (Gestisci integrazioni), quindi scegli Create (Crea).
5. Salta il passaggio Collega questa integrazione a una route. Lo completerai in una fase successiva.
6. Per Integration type (Tipo di integrazione), scegli Lambda Function (Funzione Lambda).
7. Per Lambda function (Funzione Lambda), immetti **`http-crud-tutorial-function`**.
8. Seleziona Crea.

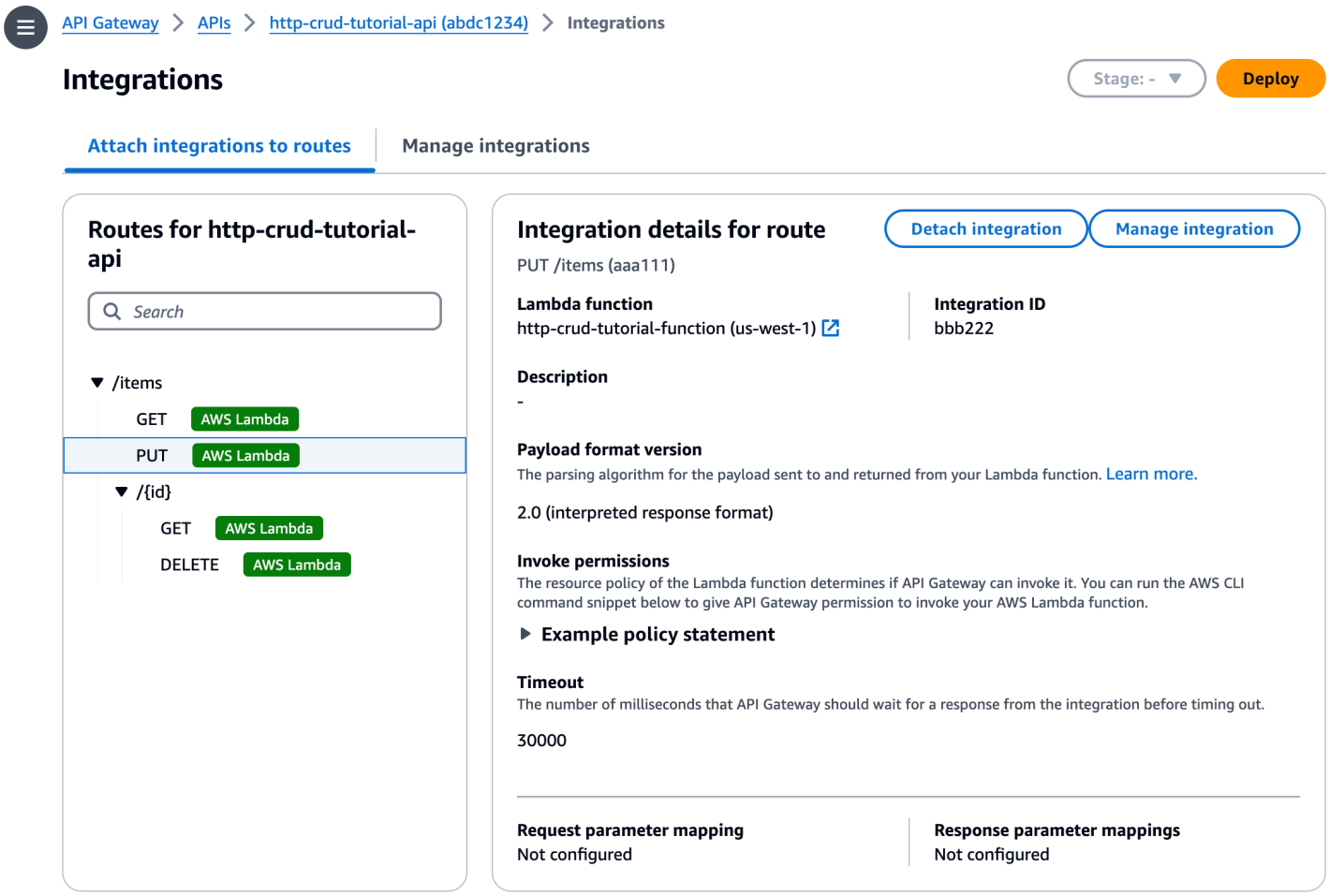
Fase 6: collega la tua integrazione alle route

Per questa API di esempio, si utilizza la stessa integrazione Lambda per tutte le route. Dopo aver collegato l'integrazione a tutte le route dell'API, la funzione Lambda viene richiamata quando un client chiama una delle route.

Per collegare integrazioni alle route

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere l'API.
3. Scegli Integrations (Integrazioni).
4. Scegli una route.
5. Nel campo Choose an existing integration (Scegli un'integrazione esistente), scegli **`http-crud-tutorial-function`**.
6. Scegli Collega integrazione.
7. Ripeti le fasi 4-6 per tutte le route.

Tutti i percorsi mostrano che è associata un' AWS Lambda integrazione.



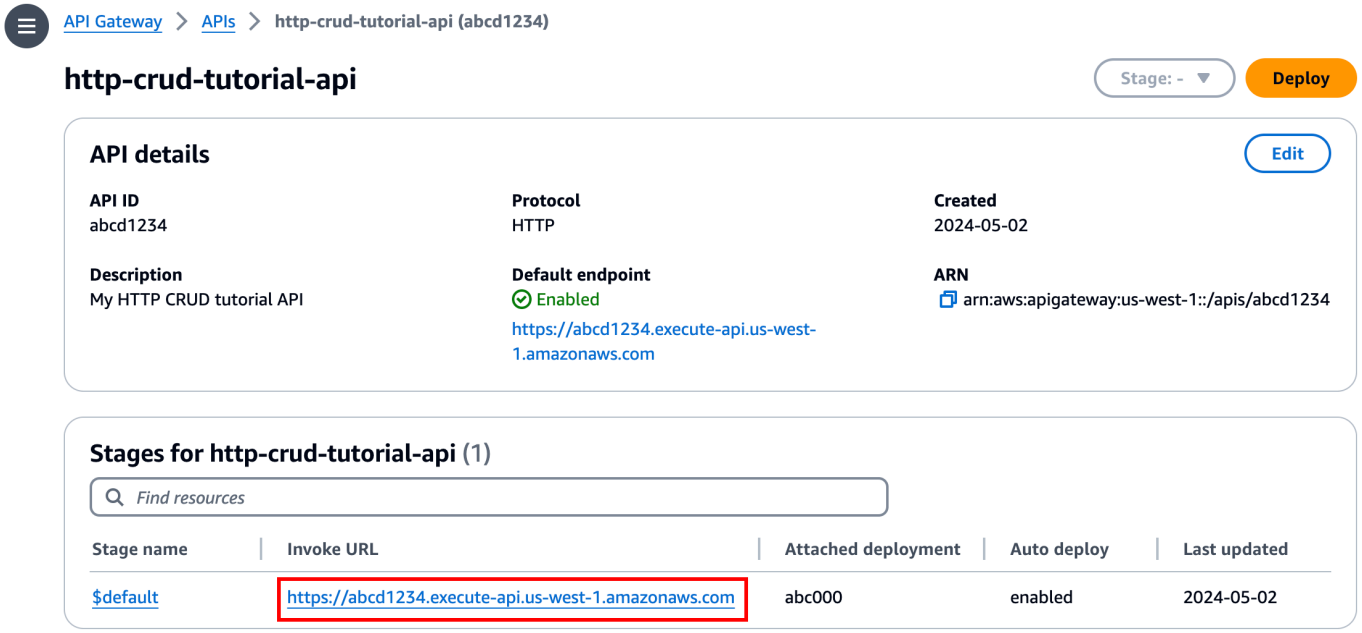
Ora che hai un'API HTTP con route e integrazioni, puoi testare la tua API.

Fase 7: test dell'API

Per assicurarti che la tua API funzioni, usa curl.

Per ottenere l'URL per richiamare la tua API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere l'API.
3. Prendere nota del valore URL di chiamata per l'API. Viene visualizzato in Invoke URL (Richiama URL) nella pagina Details (Dettagli).



4. Copia l'URL di chiamata per l'API.

   L'URL completo è simile a `https://abcdef123.execute-api.us-west-2.amazonaws.com`.

Per creare o aggiornare una voce

- Utilizza il comando seguente per creare o aggiornare una voce. Il comando include un corpo della richiesta con l'ID, il prezzo e il nome della voce.

```
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
```

Per ottenere tutte le voci

- Utilizza il seguente comando per elencare tutte le voci.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/items
```

Per ottenere una voce

- Utilizza il seguente comando per ottenere una voce in base al relativo ID.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/items/123
```

Per eliminare una voce

1. Utilizza il comando seguente per eliminare una voce.

```
curl -X "DELETE" https://abcdef123.execute-api.us-west-2.amazonaws.com/items/123
```

2. Ottieni tutte le voci per verificare che la voce sia stata eliminata.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/items
```

Fase 8: Pulizia

Per evitare costi non necessari, eliminare le risorse create nell'ambito di questo esercizio di nozioni di base. La procedura seguente elimina l'API HTTP, la funzione Lambda e le risorse associate.

Per eliminare una tabella DynamoDB

1. Apri la console DynamoDB all'indirizzo. https://console.aws.amazon.com/dynamodb/
2. Seleziona la tabella.
3. Seleziona Delete Table (Elimina tabella).
4. Conferma la scelta e seleziona Delete (Elimina).

Per eliminare un'API HTTP

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nella pagina API, selezionare un'API. Scegli Azioni, quindi Elimina.
3. Scegliere Delete (Elimina).

Per eliminare una funzione Lambda

1. Accedi alla console Lambda all'indirizzo. https://console.aws.amazon.com/lambda
2. Nella pagina Funzioni, selezionare una funzione. Scegli Azioni, quindi Elimina.
3. Scegliere Delete (Elimina).

Per eliminare il gruppo di log di una funzione Lambda

1. Nella CloudWatch console Amazon, apri la pagina dei gruppi di log.
2. Nella pagina Log Groups (Gruppi di log), seleziona il gruppo di log della funzione (`/aws/lambda/http-crud-tutorial-function`). Scegliere Actions (Operazioni), quindi selezionare Delete log group (Elimina gruppo di log).
3. Scegliere Delete (Elimina).

Per eliminare il ruolo di esecuzione di una funzione Lambda

1. Nella AWS Identity and Access Management console, apri la pagina Ruoli.
2. Seleziona il ruolo della funzione, ad esempi, `http-crud-tutorial-role`.
3. Scegliere Delete role (Elimina ruolo).
4. Scegliere Yes, delete (Sì, elimina).

Passaggi successivi: automatizza con AWS SAM or CloudFormation

È possibile automatizzare la creazione e la pulizia delle AWS risorse utilizzando o. CloudFormation AWS SAM Per un esempio di modello AWS SAM per questo tutorial, consulta `template.yaml`.

Per esempio CloudFormation modelli, vedi modelli di esempio CloudFormation.

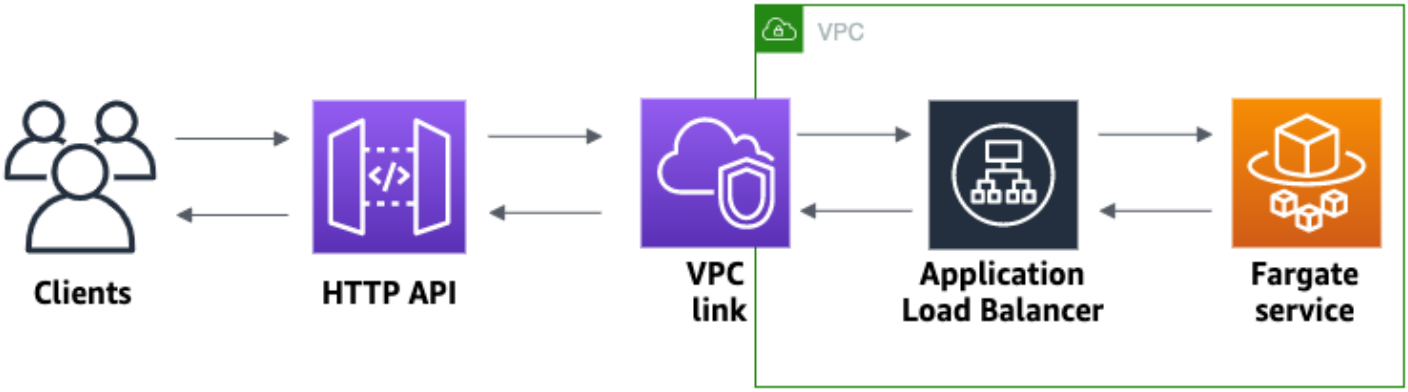
Tutorial: creazione di un'API HTTP con un'integrazione privata con un servizio Amazon ECS

In questo tutorial, si crea un'API serverless che si connette a un servizio Amazon ECS eseguito in un VPC Amazon. I clienti al di fuori del tuo Amazon VPC possono utilizzare l'API per accedere al tuo servizio Amazon ECS.

Questo tutorial dura circa un'ora. Innanzitutto, utilizzi un CloudFormation modello per creare un servizio Amazon VPC e Amazon ECS. Usa la console API Gateway per creare un link VPC. Il link

VPC consente ad API Gateway di accedere al servizio Amazon ECS eseguito nel tuo VPC Amazon. Successivamente, si crea un'API HTTP che utilizza il link VPC per connettersi al servizio Amazon ECS. Infine, si esegue il dell'API.

Quando si richiama l'API HTTP, API Gateway invia la richiesta al tuo servizio Amazon ECS tramite il link VPC e poi restituisce la risposta dal servizio.



Per completare questo tutorial, sono necessari un AWS account e un AWS Identity and Access Management utente con accesso alla console. Per ulteriori informazioni, consulta Prerequisiti.

In questo tutorial utilizzerai Console di gestione AWS. Per un CloudFormation modello che crea questa API e tutte le risorse correlate, vedi template.yaml.

Argomenti

- Fase 1: Creazione di un servizio Amazon ECS
- Fase 2: Creazione di un link VPC
- Fase 3: creazione un'API HTTP
- Fase 4: Creazione di una route
- Fase 5: creazione di un'integrazione
- Fase 6: Test dell'API
- Fase 7: pulizia
- Passaggi successivi: automatizza con CloudFormation

Fase 1: Creazione di un servizio Amazon ECS

Amazon ECS è un servizio di gestione dei container che facilita l'esecuzione, l'arresto e la gestione di container Docker in un cluster. In questo tutorial eseguirai il cluster su un'infrastruttura serverless gestita da Amazon ECS.

Scarica e decomprimi questo CloudFormation modello, che crea tutte le dipendenze per il servizio, incluso un Amazon VPC. Si utilizza il modello per creare un servizio Amazon ECS che utilizza un Application Load Balancer.

Per creare un CloudFormation stack

1. Apri la CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation.
2. Scegliere Create stack (Crea stack), quindi With new resources (standard) (Con nuove risorse (standard)).
3. In Specificare modello, scegliere Carica un file modello.
4. Selezionare il modello scaricato.
5. Scegli Next (Successivo).
6. Per Stack name (Nome stack), inserire **`http-api-private-integrations-tutorial`**, quindi scegliere Next (Avanti).
7. Per Configure stack options (Configura opzioni di stack), scegliere Next (Successivo).
8. Per quanto riguarda le funzionalità, riconosci che CloudFormation puoi creare risorse IAM nel tuo account.
9. Scegli Avanti, quindi scegli Invia.

CloudFormation esegue il provisioning del servizio ECS, operazione che può richiedere alcuni minuti. Quando lo stato del tuo CloudFormation stack è CREATE_COMPLETE, sei pronto per passare alla fase successiva.

Fase 2: Creazione di un link VPC

Un link VPC consente ad API Gateway di accedere alle risorse private in un VPC Amazon. Si utilizza un link VPC per consentire ai client di accedere al tuo servizio Amazon ECS tramite l'API HTTP.

Per creare un link VPC

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.

2. Nel pannello di navigazione principale scegli Collegamenti VPC e quindi Crea.

   È possibile che per aprire il pannello di navigazione principale sia necessario scegliere l'icona del menu.
3. Per Scegliere una versione del link VPC, selezionare Link VPC per API HTTP.
4. In Name (Nome), immettere **`private-integrations-tutorial`**.
5. Per VPC scegliere il VPC creato nella fase 1. Il nome deve iniziare con PrivateIntegrationsStack.
6. Per Sottoreti, selezionare le due sottoreti private nel VPC. I nomi finiscono con `PrivateSubnet`.
7. Per Gruppi di sicurezza seleziona l'ID gruppo che inizia con `private-integrations-tutorial` e ha la descrizione `PrivateIntegrationsStack/PrivateIntegrationsTutorialService/Service/SecurityGroup`.
8. Scegli Create (Crea).

Dopo aver creato il link VPC, API Gateway consente alle interfacce di rete elastiche di accedere al VPC. Il processo può richiedere alcuni minuti. Nel frattempo, puoi creare la tua API.

Fase 3: creazione un'API HTTP

L'API HTTP fornisce un endpoint HTTP per il Servizio Amazon ECS. In questa fase si crea un'API vuota. Nelle fasi 4 e 5, si configura un percorso e un'integrazione per connettere l'API e il servizio Amazon ECS.

Come creare un'API HTTP

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli Create API (Crea API), quindi per API HTTP, scegli Build (Crea).
3. Per API name (Nome API), immettere **`http-private-integrations-tutorial`**.
4. Per Tipo di indirizzo IP, selezionare IPv4.
5. Scegli Next (Successivo).
6. Per Configura route, scegli Avanti per ignorare la creazione della route. È possibile creare route in un secondo momento.
7. Esamina la fase creata da API Gateway. API Gateway crea una `$default` fase con distribuzioni automatiche abilitate, che è la scelta migliore per questo tutorial. Scegli Next (Successivo).
8. Seleziona Crea.

Fase 4: Creazione di una route

Le route inviano le richieste API in entrata alle risorse di back-end. Le route sono costituite da due parti: un metodo HTTP e un percorso della risorsa, ad esempi, `GET /items`. Per questa API di esempio, creiamo una route.

Per creare una route

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere l'API.
3. Scegliere Percorsi.
4. Seleziona Crea.
5. Per Method (Metodo) seleziona **`ANY`**.
6. Per il percorso, immetti **`/{proxy+}`**. Il`{proxy+}` alla fine del percorso è una variabile di percorso greedy. API Gateway invia tutte le richieste per l'API a questa route.
7. Seleziona Crea.

Fase 5: creazione di un'integrazione

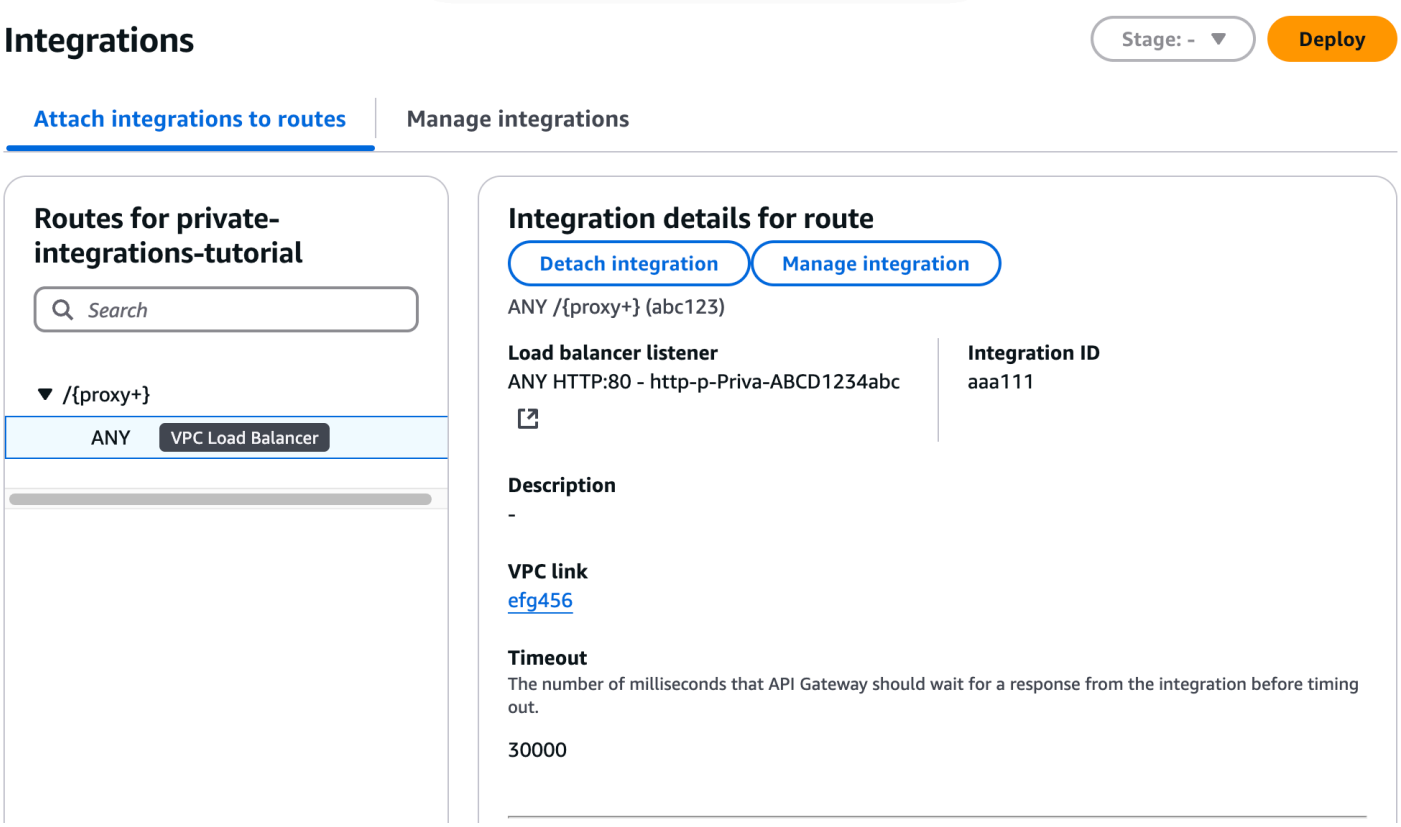
Si crea un'integrazione per connettere una route alle risorse di back-end.

Per creare un'integrazione

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere l'API.
3. Scegli Integrations (Integrazioni).
4. Scegli Manage integrations (Gestisci integrazioni), quindi scegli Create (Crea).
5. Per Allega questa integrazione a una route, selezionare la route ANY/{proxy+} creata in precedenza.
6. Per Tipo di integrazionescegliere Risorsa privata.
7. Per Dettagli sull'integrazione, scegliere Seleziona manualmente.
8. Per il servizio Target, scegli ALB/NLB.
9. Per load balancer, scegliere il load balancer creato con il modello CloudFormation nella fase 1. Il suo nome dovrebbe iniziare con HTTP-Priva.

10. Per Listener, scegliere **`HTTP 80`**.
11. Per Link VPC, scegliere il link VPC creato nella fase 2. Dovrebbe chiamarsi `private-integrations-tutorial`.
12. Seleziona Crea.

Per verificare che il percorso e l'integrazione siano configurati correttamente, selezionare Allega integrazioni alle route. La console mostra che si dispone di una route `ANY /{proxy+}` con un'integrazione a un load balancer VPC.



Ora siamo pronti per testare l'API.

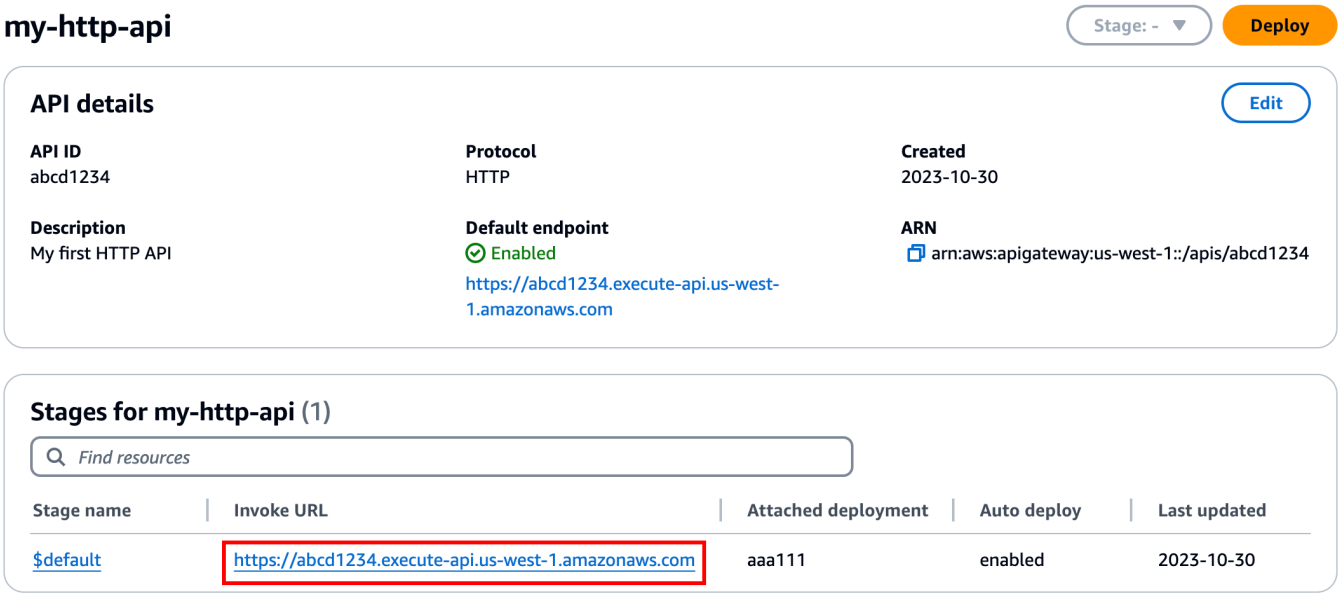
Fase 6: Test dell'API

Successivamente, viene verificata l'API per assicurarsi che funzioni. Per semplicità, usa un browser Web per richiamare l'API.

Per testare l'API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere l'API.

3. Prendere nota del valore URL di chiamata per l'API.



4. In un browser web, vai all'URL di chiamata dell'API.

   L'URL completo dovrebbe essere del tipo `https://abcdef123.execute-api.us-east-2.amazonaws.com`.

   Il tuo browser invia una richiesta `GET` all'API.

5. Verifica che la risposta della tua API sia un messaggio di benvenuto che indica che la tua app è in esecuzione su Amazon ECS.

   Se visualizzi il messaggio di benvenuto, hai creato correttamente un servizio Amazon ECS eseguito in un VPC Amazon e hai utilizzato un'API HTTP di API Gateway con un link VPC per accedere al servizio Amazon ECS.

Fase 7: pulizia

Per evitare costi non necessari, eliminare le risorse create nell'ambito di questo tutorial. I seguenti passaggi eliminano il link VPC, lo CloudFormation stack e l'API HTTP.

Per eliminare un'API HTTP

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nella pagina API, selezionare un'API. Scegliere Azioni, scegliere Elimina, quindi confermare la scelta.

Per eliminare un link VPC

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere link VPC.
3. Selezionare il link VPC, scegliere Elimina, quindi confermare la scelta.

Per eliminare un CloudFormation stack

1. Apri la CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation.
2. Seleziona il tuo CloudFormation stack.
3. Scegli Elimina e conferma la tua scelta.

Passaggi successivi: automatizza con CloudFormation

Puoi automatizzare la creazione e la pulizia di tutte le AWS risorse coinvolte in questo tutorial. Per un modello CloudFormation di esempio completo, consulta template.yaml.

Tutorial sull'API WebSocket di Amazon API Gateway

I seguenti tutorial offrono esercizi pratici per aiutare a utilizzare le API WebSocket di Gateway API.

Argomenti

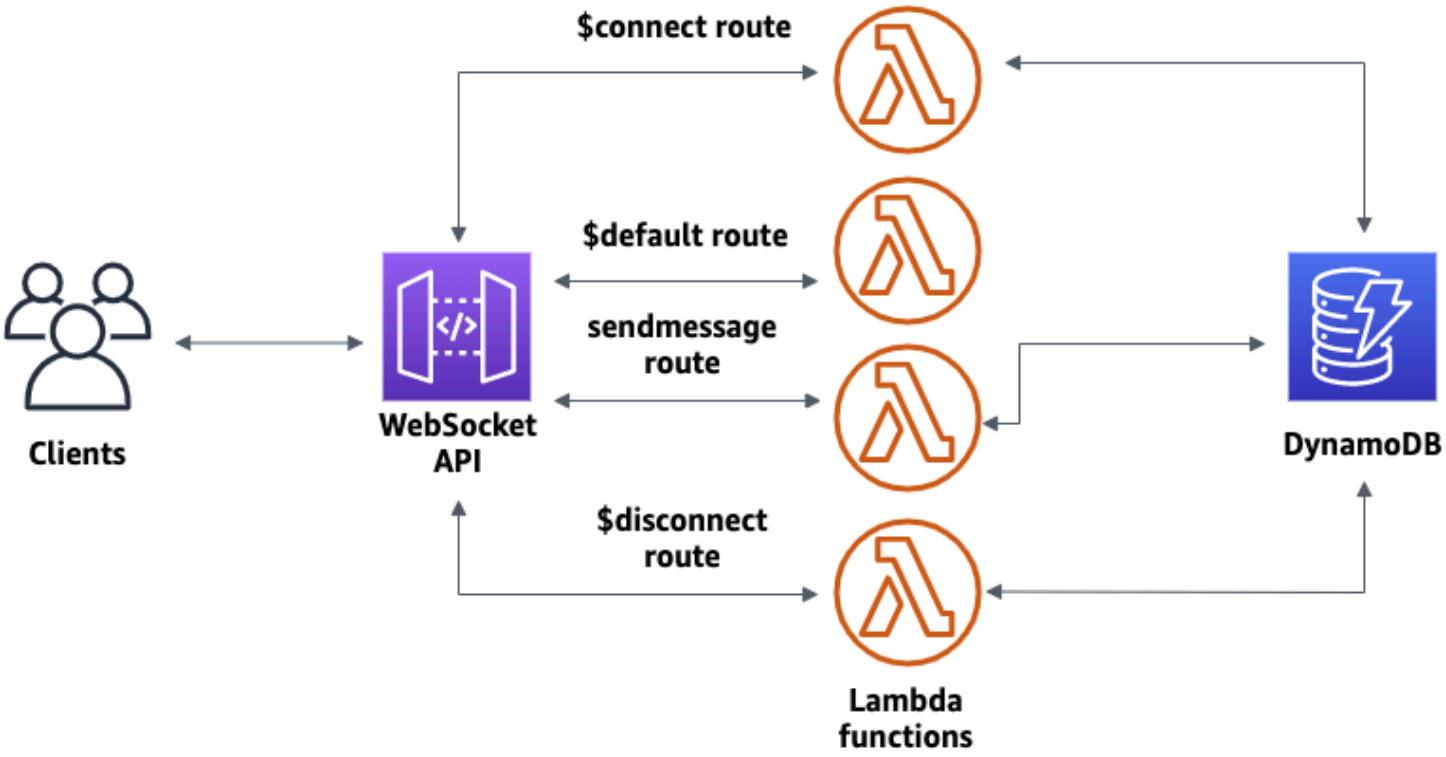
- Tutorial: crea un'app di WebSocket chat con WebSocket API, Lambda e DynamoDB
- Tutorial: creare un' WebSocket API con un AWS integrazione

Tutorial: crea un'app di WebSocket chat con WebSocket API, Lambda e DynamoDB

In questo tutorial, creerai un'applicazione di chat senza server con un' WebSocket API. Con un' WebSocket API, puoi supportare la comunicazione bidirezionale tra i client. I client possono ricevere messaggi senza dover eseguire il polling per gli aggiornamenti.

Il completamento di questa esercitazione richiede circa 30 minuti. Innanzitutto, utilizzerai un CloudFormation modello per creare funzioni Lambda che gestiranno le richieste API, oltre a una tabella DynamoDB che memorizza gli ID client. Quindi, utilizzerai la console API Gateway per creare

un' WebSocket API che si integri con le tue funzioni Lambda. Infine, eseguirai il test dell'API per verificare che i messaggi siano inviati e ricevuti.



Per completare questo tutorial, sono necessari un AWS account e un AWS Identity and Access Management utente con accesso alla console. Per ulteriori informazioni, consulta Prerequisiti.

Per la connessione all'API è inoltre necessario `wscat`. Per ulteriori informazioni, consulta the section called “Utilizzalo `wscat` per connetterti a un' WebSocket API e inviarle messaggi”.

Argomenti

- Fase 1: creazione di funzioni Lambda e di una tabella DynamoDB
- Fase 2: Creare un'API WebSocket
- Fase 3: test dell'API
- Fase 4: pulizia
- Passaggi successivi: automatizza con CloudFormation

Fase 1: creazione di funzioni Lambda e di una tabella DynamoDB

Scarica e decomprimi il modello di creazione dell'app per CloudFormation. Utilizzerai questo modello per creare una tabella Amazon DynamoDB per archiviare gli ID client dell'applicazione. Ogni client

connesso ha un ID univoco che utilizzeremo come chiave di partizione della tabella. Questo modello crea anche funzioni Lambda che aggiornano le connessioni client in DynamoDB e gestiscono l'invio di messaggi ai client connessi.

Per creare un CloudFormation stack

1. Apri la CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation.
2. Scegliere Create stack (Crea stack), quindi With new resources (standard) (Con nuove risorse (standard)).
3. In Specificare modello, scegliere Carica un file modello.
4. Selezionare il modello scaricato.
5. Scegli Next (Successivo).
6. Per Stack name (Nome stack), inserire **`websocket-api-chat-app-tutorial`**, quindi scegliere Next (Avanti).
7. Per Configure stack options (Configura opzioni di stack), scegliere Next (Successivo).
8. Per quanto riguarda le funzionalità, riconosci che CloudFormation puoi creare risorse IAM nel tuo account.
9. Scegli Avanti, quindi scegli Invia.

CloudFormation fornisce le risorse specificate nel modello. Per completare il provisioning delle risorse, potrebbero essere necessari alcuni minuti. Quando lo stato del tuo CloudFormation stack è CREATE_COMPLETE, sei pronto per passare alla fase successiva.

Fase 2: Creare un'API WebSocket

Creerai un' WebSocket API per gestire le connessioni dei client e indirizzare le richieste alle funzioni Lambda che hai creato nel passaggio 1.

Per creare un'API WebSocket

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Seleziona Create API (Crea API). Quindi, per WebSocket API, scegli Build.
3. Per API name (Nome API), immettere **`websocket-chat-app-tutorial`**.
4. Per Tipo di indirizzo IP, selezionare IPv4.

5. Per Route selection expression (Espressione di selezione routing), inserire **`request.body.action`**. L'espressione di selezione del routing determina quale routing viene richiamato da API Gateway quando un client invia un messaggio.
6. Scegli Next (Successivo).
7. Per Predefined routes (Routing predefiniti), scegliere Add $connect (Aggiungi $connect), Add $disconnect (Aggiungi $disconnect) e Add $default (Aggiungi $default). I routing $connect e $disconnect sono routing speciali che API Gateway richiama automaticamente quando un client si connette o si disconnette da un'API. API Gateway richiama il routing `$default` quando nessun altro routing corrisponde a una richiesta.
8. Per Custom routes (Routing personalizzati), scegli Add custom route (Aggiungi routing personalizzato). Per Route key (Chiave routing), inserire **`sendmessage`**. Questo routing personalizzato gestisce i messaggi inviati ai client connessi.
9. Scegli Next (Successivo).
10. In Attach integrations (Collega integrazioni), per ogni routing eIntegration type (Tipo di integrazione), scegliere Lambda.

    Per Lambda, scegli la funzione Lambda corrispondente che hai creato nel passaggio 1. CloudFormation Il nome di ciascuna funzione corrisponde a un routing. Ad esempio, per il routing $connect, scegliere la funzione denominata **`websocket-chat-app-tutorial-ConnectHandler`**.
11. Esamina la fase creata da API Gateway. Di default, API Gateway crea un nome di fase `production` e implementa automaticamente l'API in quella fase. Scegli Next (Successivo).
12. Scegliere Create and deploy (Crea e implementa).

Fase 3: test dell'API

Successivamente, eseguirai il test dell'API per assicurarti che funzioni correttamente. Per connetterti all'API, utilizza il comando `wscat`.

Per ottenere l'URL di richiamo dell'API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere l'API.
3. Scegli Stages (Fasi), quindi scegli production (produzione).

4. Annota l'WebSocket URL della tua API. L'URL dovrebbe essere del tipo `wss://abcdef123.execute-api.us-east-2.amazonaws.com/production`.

Per connetterti all'API

1. Per connetterti all'API, utilizza il seguente comando. Quando ti connetti all'API, API Gateway richiama il routing `$connect`. Quando si richiama il routing, esso richiama una funzione Lambda che archivia l'ID di connessione in DynamoDB.

```
wscat -c wss://abcdef123.execute-api.us-west-2.amazonaws.com/production
```

```
Connected (press CTRL+C to quit)
```

2. Apri un nuovo terminale ed esegui nuovamente il comando wscat con i parametri seguenti.

```
wscat -c wss://abcdef123.execute-api.us-west-2.amazonaws.com/production
```

```
Connected (press CTRL+C to quit)
```

   Ciò fornisce due client connessi in grado di scambiare messaggi.

Come inviare un messaggio

- API Gateway determina quale routing richiamare in base all'espressione di selezione del routing dell'API. L'espressione di selezione del routing dell'API è `$request.body.action`. Di conseguenza, API Gateway richiama il routing `sendmessage` quando si invia il seguente messaggio:

```
{"action": "sendmessage", "message": "hello, everyone!"}
```

  La funzione Lambda associata al routing richiamato raccoglie gli ID client da DynamoDB. Quindi, la funzione chiama l'API Gateway Management API e invia il messaggio a tali client. Tutti i client connessi ricevono il seguente messaggio:

```
< hello, everyone!
```

Per richiamare il routing $default dell'API

- API Gateway richiama il routing di default dell'API quando un client invia un messaggio che non corrisponde ai routing definiti. La funzione Lambda associata al routing `$default` utilizza l'API di API Gateway Management per inviare al client informazioni relative alla connessione.

```
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
```

Per disconnetterti dall'API

- Per disconnetterti dall'API, premi **`CTRL+C`**. Quando un client si disconnette dall'API, API Gateway richiama il routing `$disconnect` dell'API. L'integrazione Lambda per il routing `$disconnect` dell'API rimuove l'ID di connessione da DynamoDB.

Fase 4: pulizia

Per evitare costi non necessari, eliminare le risorse create nell'ambito di questo tutorial. I passaggi seguenti eliminano lo CloudFormation stack e WebSocket l'API.

Per eliminare un'API WebSocket

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nella pagina API, selezionare l'API `websocket-chat-app-tutorial`. Scegliere Azioni, scegliere Elimina, quindi confermare la scelta.

Per eliminare un CloudFormation stack

1. Apri la CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation.
2. Seleziona il tuo CloudFormation stack.
3. Scegli Elimina e conferma la tua scelta.

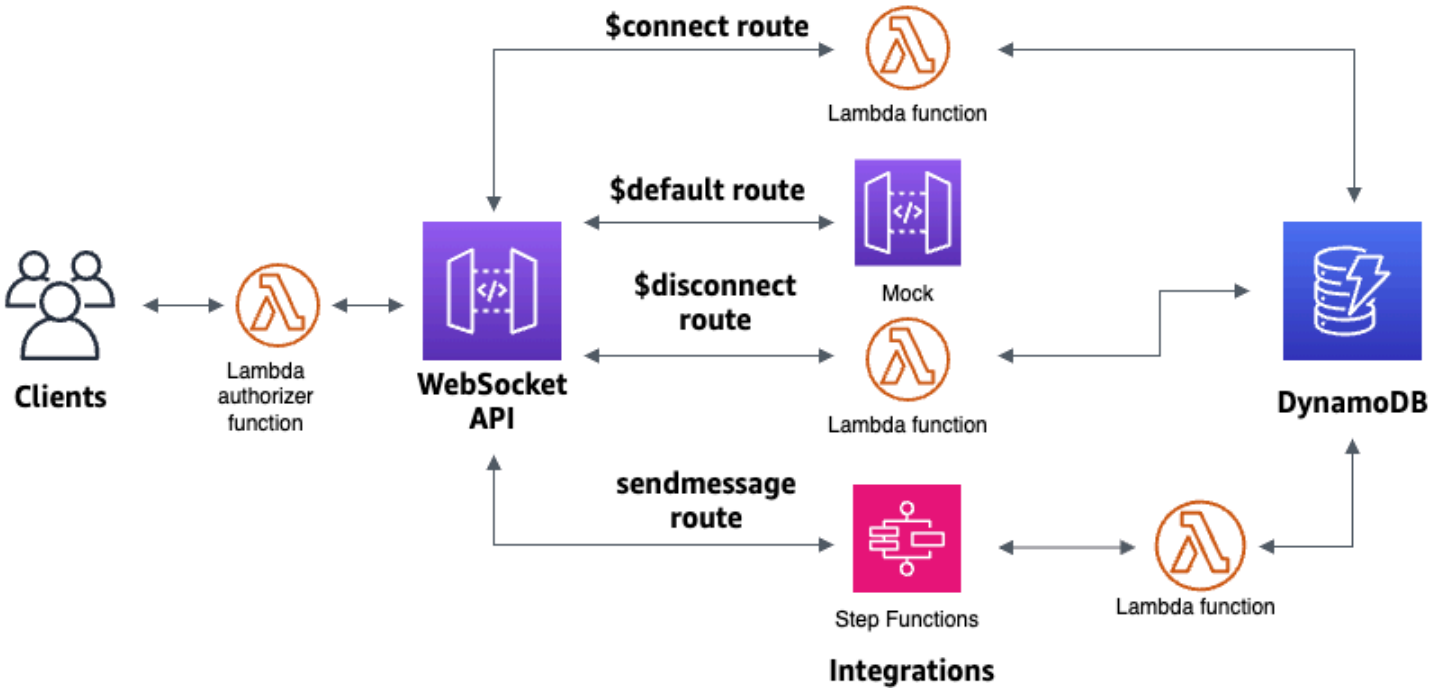
Passaggi successivi: automatizza con CloudFormation

Puoi automatizzare la creazione e la pulizia di tutte le AWS risorse coinvolte in questo tutorial. Per un CloudFormation modello che crea questa API e tutte le risorse correlate, vedi ws-chat-app.yaml.

Tutorial: creare un' WebSocket API con un AWS integrazione

In questo tutorial, crei un'applicazione di trasmissione serverless con un' WebSocket API. I client possono ricevere messaggi senza dover eseguire il polling per gli aggiornamenti.

Questo tutorial mostra come trasmettere messaggi ai client connessi e include un esempio di sistema di autorizzazione Lambda, un'integrazione fittizia e un'integrazione non proxy con Step Functions.



Dopo aver creato le risorse utilizzando un CloudFormation modello, utilizzerai la console API Gateway per creare un' WebSocket API che si integri con AWS le tue risorse. Allegherai un autorizzatore Lambda alla tua API e creerai un'integrazione di AWS servizi con Step Functions per avviare l'esecuzione di una macchina a stati. La macchina a stati Step Functions invocherà una funzione Lambda che invia un messaggio a tutti i client connessi.

Dopo aver creato l'API, testerai la connessione all'API e verificherai che i messaggi vengano inviati e ricevuti. Il completamento di questo tutorial richiede circa 45 minuti.

Argomenti

- Prerequisiti

- Fase 1: Creazione delle risorse
- Passaggio 2: creare un'API WebSocket
- Passaggio 3: creare un sistema di autorizzazione Lambda
- Passaggio 4: creare un'integrazione bidirezionale fittizia
- Passaggio 5: creare un'integrazione non proxy con Step Functions
- Fase 6: Test dell'API
- Fase 7: pulire
- Fasi successive

Prerequisiti

Sono necessari i seguenti prerequisiti:

- Un AWS account e un AWS Identity and Access Management utente con accesso alla console. Per ulteriori informazioni, consulta Prerequisiti.
- `wscat` per la connessione all'API. Per ulteriori informazioni, consulta the section called “Utilizzalo `wscat` per connetterti a un' WebSocket API e inviarle messaggi”.

Ti consigliamo di completare il tutorial dell'app di WebSocket chat prima di iniziare questo tutorial. Per completare il tutorial sull'app di WebSocket chat, consultathe section called “Tutorial: crea un'app di chat WebSocket ”.

Fase 1: Creazione delle risorse

Scarica e decomprimi il modello di creazione dell'app per CloudFormation. Userai questo modello per creare quanto segue:

- Funzioni Lambda che gestiscono le richieste API e autorizzano l'accesso alla tua API.
- Una tabella DynamoDB per memorizzare gli ID dei client e l'identificazione dell'utente principale restituita dal sistema di autorizzazione Lambda.
- Una macchina a stati Step Functions per inviare messaggi ai client connessi.

Per creare un CloudFormation stack

1. Apri la CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation.

2. Scegliere Create stack (Crea stack), quindi With new resources (standard) (Con nuove risorse (standard)).
3. In Specificare modello, scegliere Carica un file modello.
4. Selezionare il modello scaricato.
5. Scegli Next (Successivo).
6. Per Stack name (Nome stack), inserire **`websocket-step-functions-tutorial`**, quindi scegliere Next (Avanti).
7. Per Configure stack options (Configura opzioni di stack), scegliere Next (Successivo).
8. Per quanto riguarda le funzionalità, riconosci che CloudFormation puoi creare risorse IAM nel tuo account.
9. Scegli Avanti, quindi scegli Invia.

CloudFormation fornisce le risorse specificate nel modello. Per completare il provisioning delle risorse, potrebbero essere necessari alcuni minuti. Scegli la scheda Output per vedere le risorse create e i relativi ARN. Quando lo stato del tuo CloudFormation stack è CREATE_COMPLETE, sei pronto per passare alla fase successiva.

Passaggio 2: creare un'API WebSocket

Creerai un' WebSocket API per gestire le connessioni dei client e indirizzare le richieste verso le risorse che hai creato nel passaggio 1.

Per creare un' WebSocket API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Seleziona Create API (Crea API). Quindi, per WebSocket API, scegli Build.
3. Per API name (Nome API), immettere **`websocket-step-functions-tutorial`**.
4. Per Tipo di indirizzo IP, selezionare IPv4.
5. Per Route selection expression (Espressione di selezione routing), inserire **`request.body.action`**.

   L'espressione di selezione del routing determina quale routing viene richiamato da API Gateway quando un client invia un messaggio.
6. Scegli Next (Successivo).

7. Per Instradamenti predefiniti, scegli Aggiungi $connect, Aggiungi $disconnect e Aggiungi $default.

   Gli instradamenti $connect e $disconnect sono instradamenti speciali che Gateway API richiama automaticamente quando un client si connette o si disconnette da un'API. Gateway API invoca l'instradamento $default quando non ci sono altri instradamenti corrispondenti a una richiesta. Creerai un percorso personalizzato per connetterti a Step Functions dopo aver creato l'API.
8. Seleziona Successivo.
9. Per Integrazione per $connect, procedi come segue:

   a. Per Tipo di integrazione, scegli Lambda.
   b. Per la funzione Lambda, scegli la funzione $connect Lambda corrispondente che hai creato CloudFormation nel passaggio 1. Il nome della funzione Lambda deve iniziare con **`websocket-step`**.
10. Per Integrazione per $disconnect, procedi come segue:

    a. Per Tipo di integrazione, scegli Lambda.
    b. Per la funzione Lambda, scegli la funzione Lambda $disconnect corrispondente che hai creato nel passaggio 1. CloudFormation Il nome della funzione Lambda deve iniziare con **`websocket-step`**.
11. Per Integrazione per $default, scegli mock.

    In un'integrazione fittizia, Gateway API gestisce la risposta di instradamento senza un backend di integrazione.
12. Scegli Next (Successivo).
13. Esamina la fase creata da API Gateway. Di default, Gateway API crea una fase denominata production e implementa automaticamente l'API in quella fase. Scegli Next (Successivo).
14. Scegliere Create and deploy (Crea e implementa).

Passaggio 3: creare un sistema di autorizzazione Lambda

Per controllare l'accesso alla tua WebSocket API, crei un autorizzatore Lambda. Il CloudFormation modello ha creato la funzione di autorizzazione Lambda per te. La funzione Lambda è visibile nella console Lambda. Il nome deve iniziare con **`websocket-step-functions-tutorial-AuthorizerHandler`**. Questa funzione Lambda nega tutte le chiamate all' WebSocket API a meno che l'intestazione non lo `Authorization` sia. `Allow` La funzione Lambda passa anche la variabile

`$context.authorizer.principalId` all'API, che viene utilizzata successivamente nella tabella DynamoDB per identificare i chiamanti dell'API.

In questo passaggio configurerai l'instradamento $connect per utilizzare il sistema di autorizzazione Lambda.

Per creare un sistema di autorizzazione Lambda

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel riquadro di navigazione principale, scegli Autorizzazioni.
3. Scegli Crea provider di autorizzazioni.
4. Per Nome del provider di autorizzazioni, inserisci **`LambdaAuthorizer`**.
5. Per Authorizer ARN, inserire il nome dell'autorizzatore creato dal modello. CloudFormation Il nome deve iniziare con **`websocket-step-functions-tutorial-AuthorizerHandler`**.

   > **Note**
   >
   > È consigliabile non utilizzare questo sistema di autorizzazione di esempio per le API di produzione.

6. Per Tipo di origine identità, scegli Intestazione. In Chiave, inserire **`Authorization`**.
7. Scegli Crea autorizzazioni.

Dopo aver creato il sistema di autorizzazione, collegalo all'instradamento $connect della tua API.

Per collegare un sistema di autorizzazione all'instradamento $connect

1. Nel pannello di navigazione principale scegli Instradamenti.
2. Scegli l'instradamento $connect.
3. Nella sezione Impostazioni della richiesta di instradamento scegli Modifica.
4. Per Autorizzazione, scegli il menu a discesa, quindi seleziona il sistema di autorizzazione della richiesta.
5. Scegli Save changes (Salva modifiche).

Passaggio 4: creare un'integrazione bidirezionale fittizia

A questo punto creerai l'integrazione bidirezionale fittizia per l'instradamento $default. Un'integrazione fittizia consente di inviare una risposta al client senza utilizzare un backend. Quando crei un'integrazione per l'instradamento $default, puoi mostrare ai client come interagire con la tua API.

Configura l'instradamento $default per indicare ai client di utilizzare l'instradamento sendmessage.

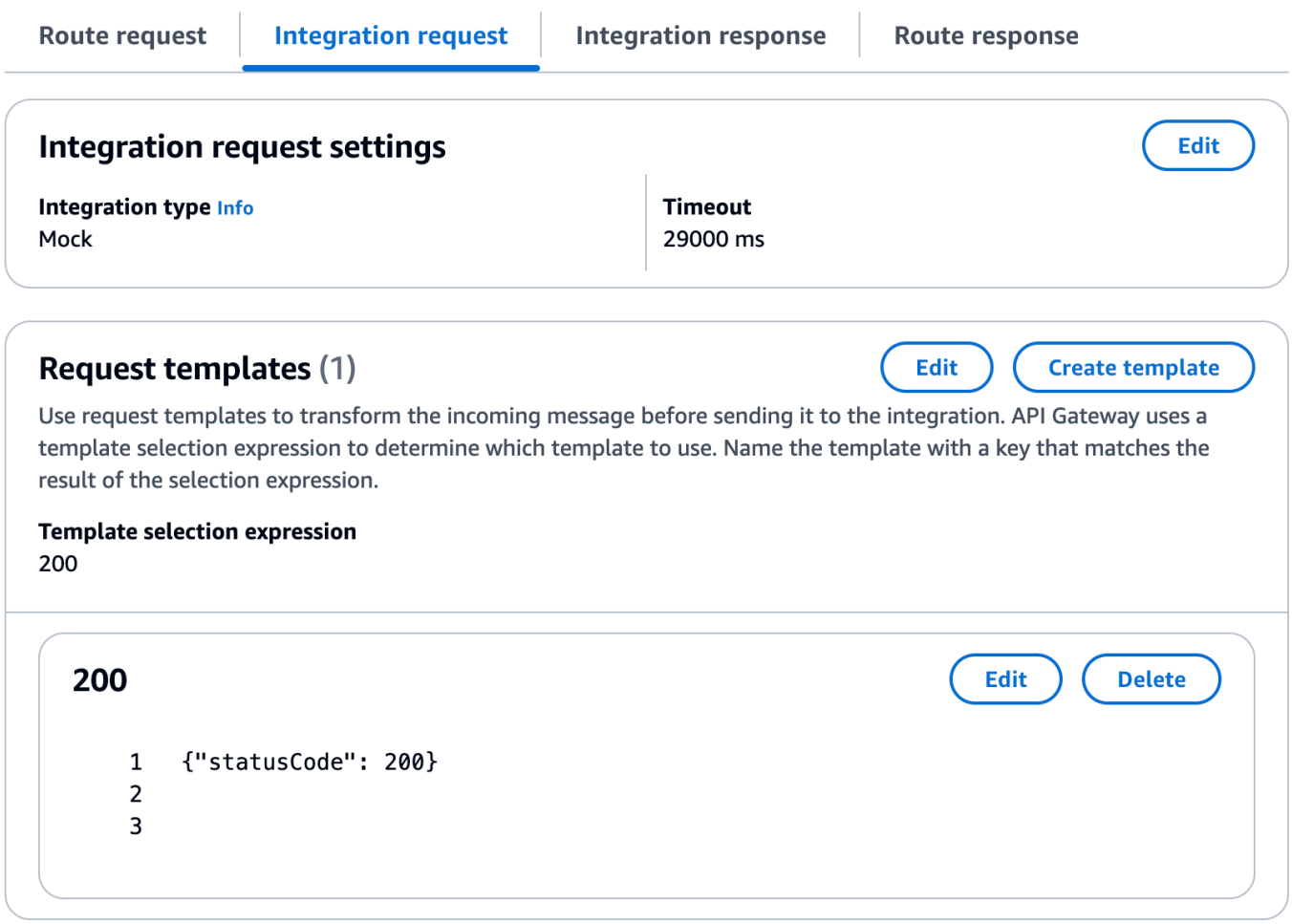
Per creare un'integrazione fittizia

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli l'instradamento $default, quindi scegli la scheda Richiesta di integrazione.
3. Per Modelli di richiesta, scegli Modifica.
4. Per Espressione di selezione del modello, inserisci **`200`**, quindi scegli Modifica.
5. Nella scheda Richiesta di integrazione, in Modelli di richiesta, scegli Crea modello.
6. Per Chiave del modello, inserisci **`200`**.
7. Per Genera modello, inserisci il seguente modello di mappatura:

   ```
   {"statusCode": 200}
   ```

   Scegli Crea modello.

   Il risultato sarà simile al seguente:

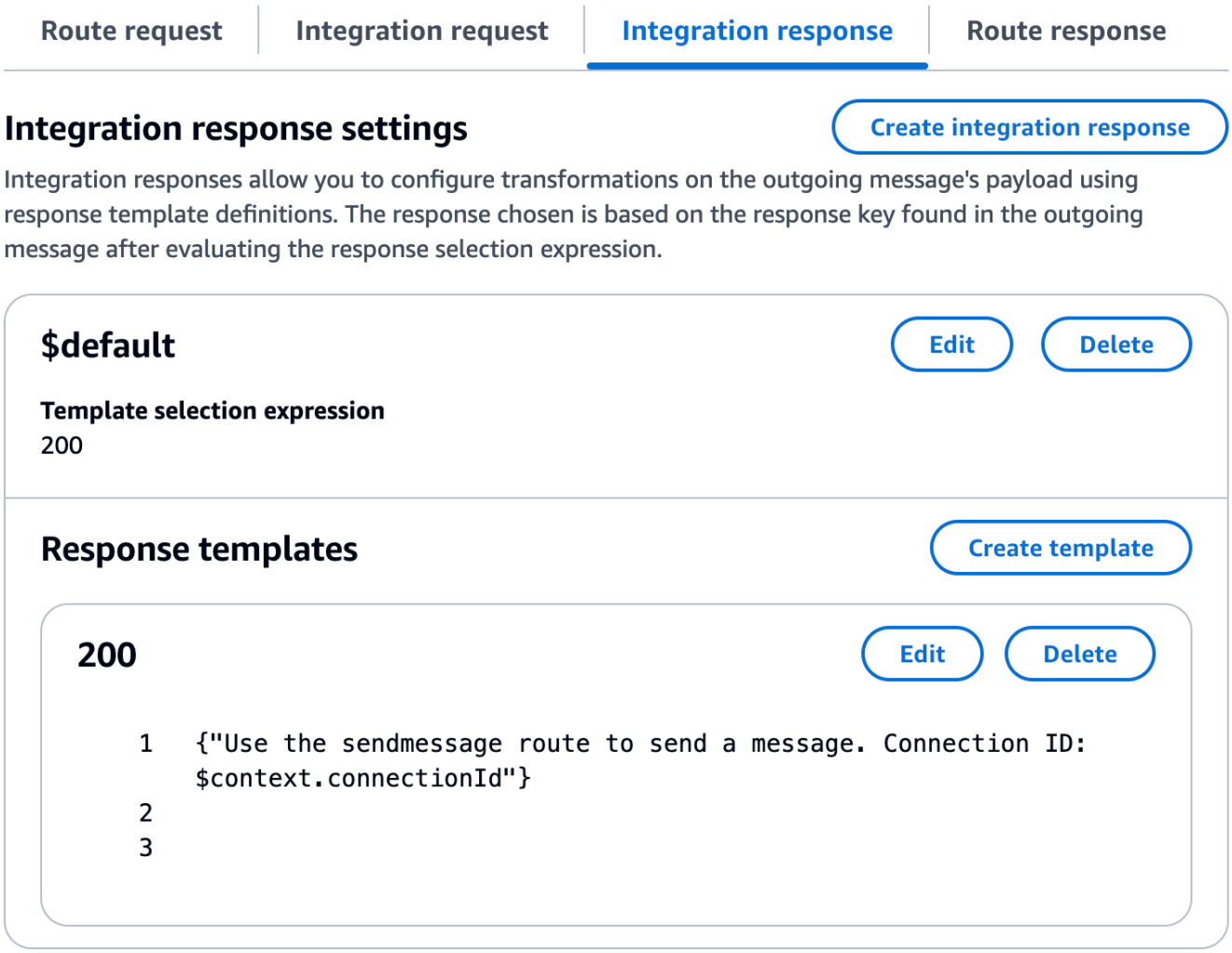


8. Nel riquadro Instradamento $default, scegli Abilita la comunicazione bidirezionale.
9. Scegli la scheda Risposta di integrazione, quindi scegli Crea risposta di integrazione.
10. Per Chiave della risposta, inserisci **`$default`**.
11. Per Espressione di selezione del modello, inserisci **`200`**.
12. Scegli Crea risposta.
13. Per Modelli di risposta, scegli Crea modello.
14. Per Chiave del modello, inserisci **`200`**.
15. Per Modello di risposta, inserisci il seguente modello di mappatura:

```
{"Use the sendmessage route to send a message. Connection ID:
 $context.connectionId"}
```

16. Scegli Crea modello.

Il risultato sarà simile al seguente:



Passaggio 5: creare un'integrazione non proxy con Step Functions

A questo punto creerai un instradamento sendmessage. I client possono invocare l'instradamento sendmessage per trasmettere un messaggio a tutti i client connessi. Il percorso sendmessage ha un'integrazione del AWS servizio non proxy con. AWS Step Functions L'integrazione richiama il StartExecutioncomando per la macchina a stati Step Functions che il CloudFormation modello ha creato per te.

Per creare un'integrazione non proxy

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.

2. Selezionare Create Route (Crea route).
3. Per Chiave instradamento, inserisci **`sendmessage`**.
4. Per Tipo di integrazione scegli Servizio AWS .
5. Per AWS Regione, inserisci la regione in cui hai distribuito il CloudFormation modello.
6. Per Servizio AWS , scegli Step Functions.
7. Per Metodo HTTP, scegli POST.
8. Per Nome azione immetti **`StartExecution`**.
9. Per Ruolo di esecuzione, inserisci il ruolo di esecuzione creato dal CloudFormation modello. Il nome deve essere WebsocketTutorialApiRole.
10. Selezionare Create Route (Crea route).

A questo punto creerai un modello di mappatura per inviare i parametri di richiesta alla macchina a stati Step Functions.

Per creare un modello di mappatura

1. Scegli l'instradamento sendmessage, quindi scegli la scheda Richiesta di integrazione.
2. Nella sezione Modelli di richiesta, scegli Modifica.
3. Per Espressione di selezione del modello, inserisci **`\$default`**.
4. Scegli Modifica.
5. Nella sezione Modelli di richiesta, scegli Crea modello.
6. Per Chiave del modello, inserisci **`\$default`**.
7. Per Genera modello, inserisci il seguente modello di mappatura:

```
#set($domain = "$context.domainName")
#set($stage = "$context.stage")
#set($body = $input.json('$'))
#set($getMessage = $util.parseJson($body))
#set($mymessage = $getMessage.message)
{
"input": "{\"domain\": \"$domain\", \"stage\": \"$stage\", \"message\":
 \"$mymessage\"}",
"stateMachineArn": "arn:aws:states:us-east-2:123456789012:stateMachine:WebSocket-
Tutorial-StateMachine"
}
```

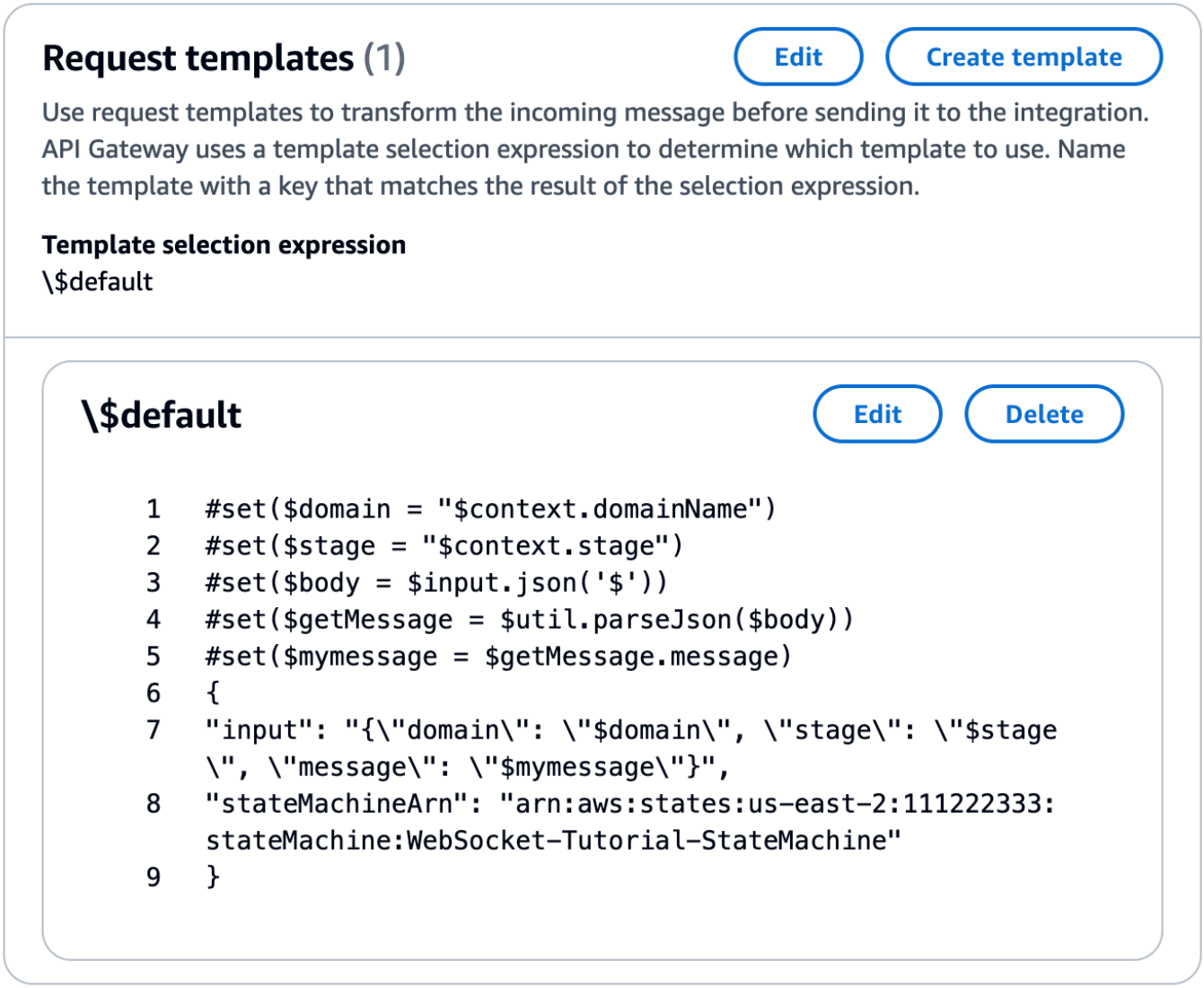
Sostituire *stateMachineArn* con l'ARN della macchina a stati creata da. CloudFormation

Questo modello di mappatura esegue queste operazioni:

- Crea la variabile `$domain` utilizzando la variabile di contesto `domainName`.
- Crea la variabile `$stage` utilizzando la variabile di contesto `stage`.

  Le variabili `$domain` e `$stage` sono necessarie per creare un URL di callback.
- Riceve il messaggio JSON `sendmessage` in arrivo ed estrae la proprietà `message`.
- Crea l'input per la macchina a stati. L'input è il dominio e lo stadio dell' WebSocket API e il messaggio proveniente dalla `sendmessage` route.

8. Scegli Crea modello.



È possibile creare un'integrazione non proxy sugli instradamenti $connect o $disconnect per aggiungere o rimuovere direttamente un ID di connessione dalla tabella DynamoDB, senza invocare una funzione Lambda.

Fase 6: Test dell'API

A questo punto distribuirai e testerai l'API per assicurarti che funzioni correttamente. Utilizzerai il `wscat` comando per connetterti all'API e poi utilizzerai un comando slash per inviare un frame ping per verificare la connessione all' WebSocket API.

Per distribuire l'API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel pannello di navigazione principale scegli Instradamenti.
3. Seleziona Deploy API (Distribuisci API).
4. Per Fase, scegli produzione.
5. (Facoltativo) Inserisci una descrizione in Descrizione distribuzione.
6. Seleziona Implementa.

Dopo aver distribuito l'API, puoi invocarla. Utilizza l'URL di richiamo per chiamare la tua API.

Per ottenere l'URL di richiamo dell'API

1. Scegliere l'API.
2. Scegli Stages (Fasi), quindi scegli production (produzione).
3. Annota l'WebSocket URL della tua API. L'URL dovrebbe essere del tipo `wss://abcdef123.execute-api.us-east-2.amazonaws.com/production`.

Ora che hai l'URL di invoke, puoi testare la connessione alla tua WebSocket API.

Per testare la connessione all'API

1. Per connetterti all'API, utilizza il seguente comando. Innanzitutto, testa la connessione invocando il percorso `/ping`.

```
wscat -c wss://abcdef123.execute-api.us-east-2.amazonaws.com/production -H
 "Authorization: Allow" --slash -P
```

```
Connected (press CTRL+C to quit)
```

2. Immetti i seguenti comandi per eseguire il ping del frame di controllo. Puoi utilizzare un frame di controllo per il keepalive dal lato client.

   ```
   /ping
   ```

   Il risultato sarà simile al seguente:

   ```
   < Received pong (data: "")
   ```

Ora che hai testato la connessione, puoi verificare che la tua API funzioni correttamente. In questo passaggio, apri una nuova finestra di terminale in modo che l' WebSocket API possa inviare un messaggio a tutti i client connessi.

Per testare l'API

1. Apri un nuovo terminale ed esegui nuovamente il comando `wscat` con i parametri seguenti.

   ```
   wscat -c wss://abcdef123.execute-api.us-east-2.amazonaws.com/production -H
    "Authorization: Allow"
   ```

   ```
   Connected (press CTRL+C to quit)
   ```

2. Gateway API determina quale instradamento invocare in base all'espressione di selezione della richiesta di instradamento dell'API. L'espressione di selezione dell'instradamento dell'API è `$request.body.action`. Di conseguenza, API Gateway richiama il routing `sendmessage` quando si invia il seguente messaggio:

   ```
   {"action": "sendmessage", "message": "hello, from Step Functions!"}
   ```

   La macchina a stati Step Functions associata all'instradamento invoca una funzione Lambda con il messaggio e l'URL di callback. La funzione Lambda chiama l'API di gestione di Gateway API e invia il messaggio a tutti i client connessi. Tutti i client connessi ricevono il seguente messaggio:

   ```
   < hello, from Step Functions!
   ```

Ora che hai testato la tua WebSocket API, puoi disconnetterti dalla tua API.

Per disconnetterti dall'API

- Per disconnetterti dall'API, premi `CTRL+C`.

  Quando un client si disconnette dall'API, Gateway API invoca l'instradamento $disconnect dell'API. L'integrazione Lambda per l'instradamento $disconnect dell'API rimuove l'ID di connessione da DynamoDB.

Fase 7: pulire

Per evitare costi non necessari, eliminare le risorse create nell'ambito di questo tutorial. I passaggi seguenti eliminano lo CloudFormation stack e WebSocket l'API.

Per eliminare un'API WebSocket

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nella pagina API, seleziona la tua websocket-api.
3. Scegliere Azioni, scegliere Elimina, quindi confermare la scelta.

Per eliminare un CloudFormation stack

1. Apri la CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation.
2. Seleziona il tuo CloudFormation stack.
3. Scegli Elimina e conferma la tua scelta.

Fasi successive

Puoi automatizzare la creazione e la pulizia di tutte le AWS risorse coinvolte in questo tutorial. Per un esempio di CloudFormation modello che automatizza queste azioni per questo tutorial, vedi ws-sfn.zip.

REST API di Gateway API

Un'API REST in API Gateway è una raccolta di risorse e metodi integrati con endpoint HTTP di back-end, funzioni Lambda o altri AWS servizi. Puoi utilizzare le caratteristiche API Gateway per semplificare tutti gli aspetti del ciclo di vita dell'API, dalla creazione al monitoraggio delle API di produzione.

Le API REST di API Gateway utilizzano un modello di richiesta/risposta in cui un client invia una richiesta a un servizio e il servizio risponde in modo sincrono. Questo tipo di modello è adatto per molti diversi tipi di applicazioni che dipendono dalla comunicazione sincrona.

Argomenti

- Sviluppo di REST API in Gateway API
- Pubblica REST APIs per i clienti da richiamare
- Ottimizzazione delle prestazioni di REST APIs
- Distribuisci il tuo REST APIs ai clienti in API Gateway
- Protezione delle REST API in Gateway API
- Monitoraggio delle REST API in Gateway API

Sviluppo di REST API in Gateway API

In Amazon API Gateway è possibile creare un'API REST come una raccolta di entità programmabili note come risorse di API Gateway. Ad esempio, si utilizza una RestApirisorsa per rappresentare un'API che può contenere una raccolta di entità Resource.

Ciascuna entità `Resource` può presentare una o più risorse Method. `Method` è una richiesta in entrata inviata dal client che può contenere un parametro di percorso, un'intestazione o un parametro della stringa di query. Inoltre, a seconda del metodo HTTP, la richiesta può contenere un corpo. Il metodo definisce in che modo il client accede alla `Resource` esposta. Per integrare `Method` con un endpoint di backend, noto anche come endpoint di integrazione, devi creare una risorsa Integrazione. In questo modo la richiesta in arrivo viene inoltrata all'URI dell'endpoint di integrazione specificato. Se necessario, è possibile trasformare i parametri o il corpo della richiesta per soddisfare i requisiti del backend oppure creare un'integrazione proxy, in cui Gateway API invia l'intera richiesta in un formato standardizzato all'URI dell'endpoint di integrazione e quindi invia la risposta direttamente al client.

Per le risposte, è possibile creare una MethodResponserisorsa per rappresentare una risposta ricevuta dal client e creare una IntegrationResponserisorsa per rappresentare la risposta restituita dal backend. Si utilizza la risposta di integrazione per trasformare i dati della risposta del backend prima di restituirli al client oppure si passa la risposta del backend al client senza modificarla.

Risorsa di esempio per una REST API

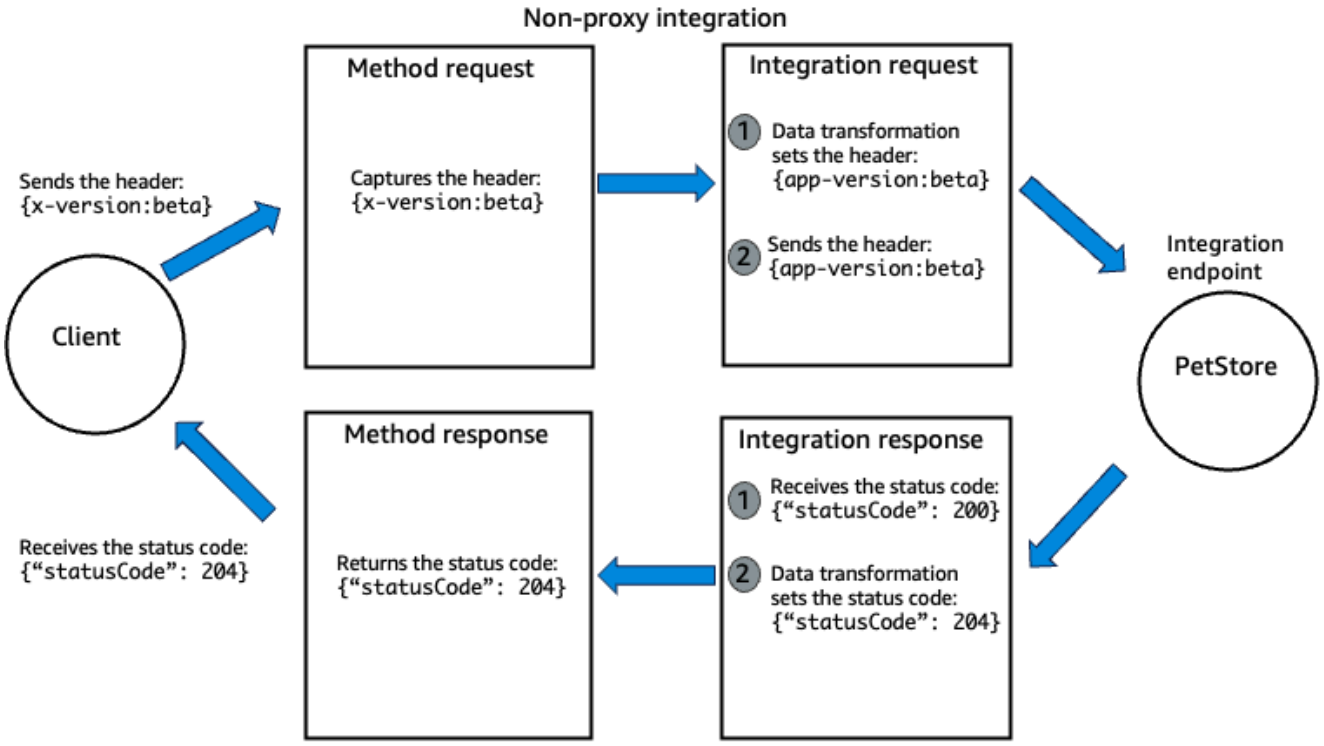
Il diagramma seguente mostra come API Gateway implementa questo request/response modello per un proxy HTTP e un'integrazione HTTP non proxy per la risorsa. `GET /pets` Il client invia l'intestazione `x-version:beta` a Gateway API che quindi invia il codice di stato `204` al client.

Nell'integrazione non proxy, Gateway API esegue le trasformazioni di dati per soddisfare i requisiti del backend, modificando la richiesta e la risposta di integrazione. In un'integrazione non proxy, è possibile accedere al corpo nella richiesta di metodo ma è necessario trasformarlo nella richiesta di integrazione. Quando l'endpoint di integrazione restituisce una risposta con un corpo, è necessario accedere e trasformarlo nella risposta di integrazione. Non è possibile modificare il corpo nella risposta di metodo.

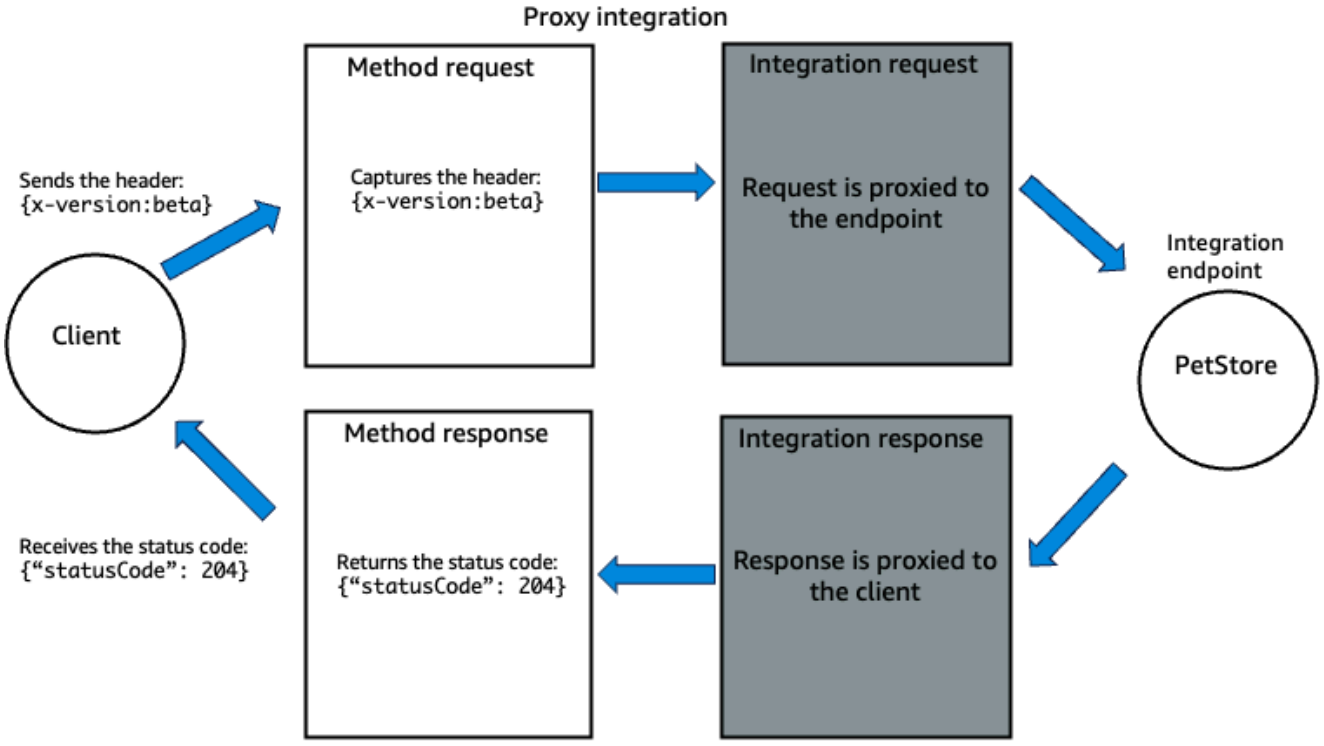
Nell'integrazione proxy, l'endpoint di integrazione modifica la richiesta e la risposta. Gateway API non modifica la richiesta o la risposta di integrazione e invia la richiesta in entrata al backend così com'è.

Indipendentemente dal tipo di integrazione, il client ha inviato una richiesta a Gateway API che ha risposto in modo sincrono.

Non-proxy integration



Proxy integration



I log di esecuzione seguenti mostrano cosa dell'esempio precedente Gateway API registra nei log. Per maggiore chiarezza, alcuni valori e log iniziali sono stati rimossi:

Non-proxy integration

```
Wed Feb 12 23:56:44 UTC 2025 : Starting execution for request: abcd-1234-5678
Wed Feb 12 23:56:44 UTC 2025 : HTTP Method: GET, Resource Path: /pets
Wed Feb 12 23:56:44 UTC 2025 : Method request path: {}
Wed Feb 12 23:56:44 UTC 2025 : Method request query string: {}
Wed Feb 12 23:56:44 UTC 2025 : Method request headers: {x-version=beta}
Wed Feb 12 23:56:44 UTC 2025 : Method request body before transformations:
Wed Feb 12 23:56:44 UTC 2025 : Endpoint request URI: http://petstore-demo-
endpoint.execute-api.com/petstore/pets
Wed Feb 12 23:56:44 UTC 2025 : Endpoint request headers: {app-version=beta}
Wed Feb 12 23:56:44 UTC 2025 : Endpoint request body after transformations:
Wed Feb 12 23:56:44 UTC 2025 : Sending request to http://petstore-demo-
endpoint.execute-api.com/petstore/pets
Wed Feb 12 23:56:45 UTC 2025 : Received response. Status: 200, Integration latency:
 123 ms
Wed Feb 12 23:56:45 UTC 2025 : Endpoint response headers: {Date=Wed, 12 Feb 2025
 23:56:45 GMT}
Wed Feb 12 23:56:45 UTC 2025 : Endpoint response body before transformations:
Wed Feb 12 23:56:45 UTC 2025 : Method response body after transformations: (null)
Wed Feb 12 23:56:45 UTC 2025 : Method response headers: {X-Amzn-Trace-Id=Root=1-
abcd-12345}
Wed Feb 12 23:56:45 UTC 2025 : Successfully completed execution
Wed Feb 12 23:56:45 UTC 2025 : Method completed with status: 204
```

Proxy integration

```
Wed Feb 12 23:59:42 UTC 2025 : Starting execution for request: abcd-1234-5678
Wed Feb 12 23:59:42 UTC 2025 : HTTP Method: GET, Resource Path: /pets
Wed Feb 12 23:59:42 UTC 2025 : Method request path: {}
Wed Feb 12 23:59:42 UTC 2025 : Method request query string: {}
Wed Feb 12 23:59:42 UTC 2025 : Method request headers: {x-version=beta}
Wed Feb 12 23:59:42 UTC 2025 : Method request body before transformations:
Wed Feb 12 23:59:42 UTC 2025 : Endpoint request URI: http://petstore-demo-
endpoint.execute-api.com/petstore/pets
Wed Feb 12 23:59:42 UTC 2025 : Endpoint request headers: { x-version=beta}
Wed Feb 12 23:59:42 UTC 2025 : Endpoint request body after transformations:
Wed Feb 12 23:59:42 UTC 2025 : Sending request to http://petstore-demo-
endpoint.execute-api.com/petstore/pets
```

```
Wed Feb 12 23:59:43 UTC 2025 : Received response. Status: 204, Integration latency:
 123 ms
Wed Feb 12 23:59:43 UTC 2025 : Endpoint response headers: {Date=Wed, 12 Feb 2025
 23:59:43 GMT}
Wed Feb 12 23:59:43 UTC 2025 : Endpoint response body before transformations:
Wed Feb 12 23:59:43 UTC 2025 : Method response body after transformations:
Wed Feb 12 23:59:43 UTC 2025 : Method response headers: {Date=Wed, 12 Feb 2025
 23:59:43 GMT}
Wed Feb 12 23:59:43 UTC 2025 : Successfully completed execution
Wed Feb 12 23:59:43 UTC 2025 : Method completed with status: 204
```

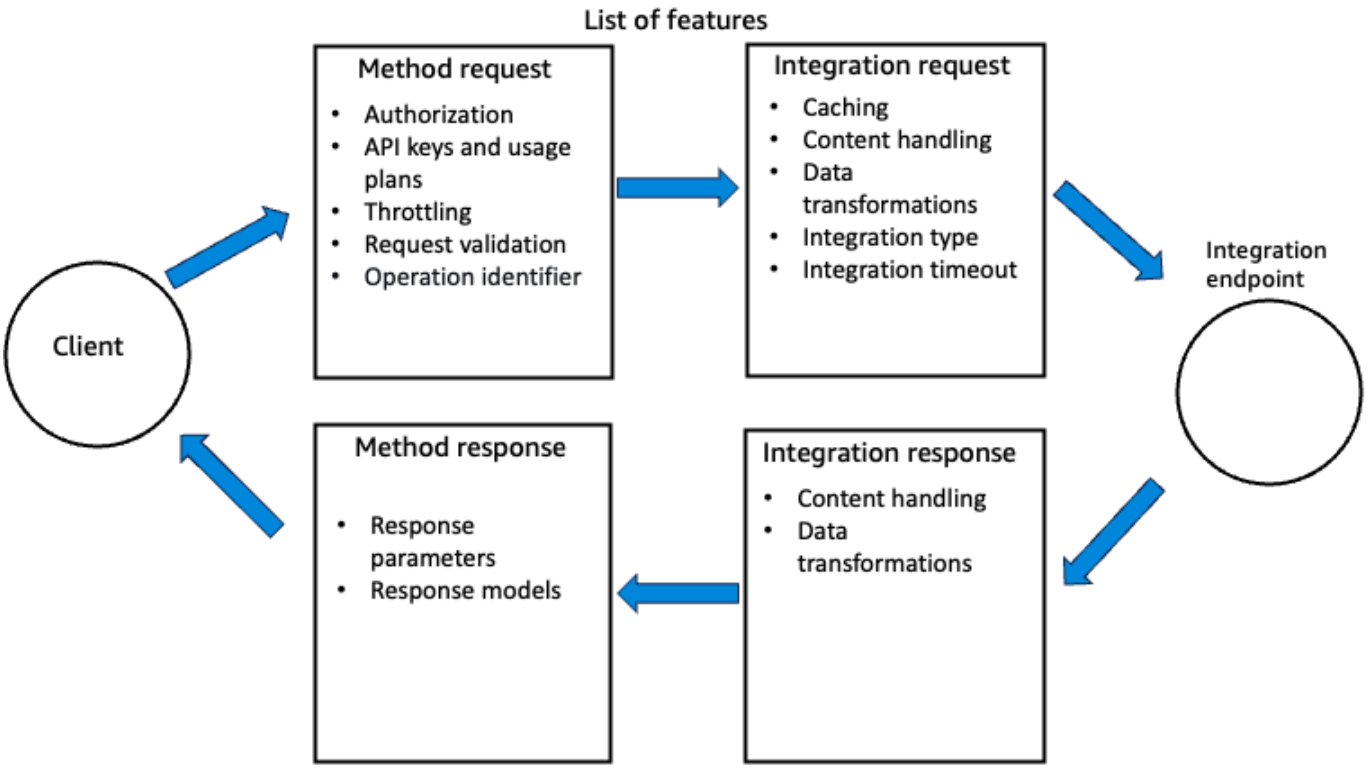
Per importare un'API simile e testarla in Console di gestione AWS, vedi l'API di esempio.

Funzionalità aggiuntive per lo sviluppo di REST API

Gateway API supporta funzionalità aggiuntive per lo sviluppo della REST API. Ad esempio, per aiutare i clienti a comprendere l'API, è possibile fornire la documentazione relativa all'API. A questo scopo, aggiungi una risorsa DocumentationPart per un'entità API supportata.

Per controllare il modo in cui i client chiamano un'API, usa le autorizzazioni IAM, un'autorizzazione Lambda o un pool di utenti di Amazon Cognito. Per misurare l'uso dell'API, configura piani di utilizzo per eseguire il throttling delle richieste API. Puoi abilitare queste impostazioni durante la creazione o l'aggiornamento dell'API.

Il diagramma seguente mostra le funzionalità disponibili per lo sviluppo di API REST e la posizione del request/response modello in cui tali funzionalità sono configurate.



Per un'introduzione su come creare un'API, consulta the section called “Tutorial: creazione di una REST API con un'integrazione proxy Lambda”. Per ulteriori informazioni sulle funzionalità di Gateway API che potresti utilizzare durante lo sviluppo di una REST API, consulta i seguenti argomenti. Questi argomenti contengono informazioni e procedure concettuali che è possibile eseguire utilizzando la console API Gateway, l'API REST API Gateway o uno degli AWS SDK. AWS CLI

Argomenti

- Tipi di endpoint API per REST APIs in API Gateway
- Politiche di sicurezza per le API REST in API Gateway
- Tipi di indirizzo IP per REST API in Gateway API
- Metodi per REST APIs in API Gateway
- Controlla e gestisci l'accesso a REST APIs in API Gateway
- Integrazioni per REST APIs in API Gateway
- Convalida delle richieste per REST API in Gateway API
- Trasformazioni dei dati per REST API in Gateway API
- Risposte gateway per REST APIs in API Gateway
- CORS per REST API in Gateway API

- Tipi di supporti binari per REST API in Gateway API
- Invocazione di REST API in Gateway API
- Sviluppo di REST API utilizzando OpenAPI in Gateway API

Tipi di endpoint API per REST APIs in API Gateway

Per tipo di endpoint API si intende il nome host dell'API. Il tipo di endpoint dell'API può essere ottimizzato per l'edge, regionale o privato, a seconda della provenienza della maggior parte del traffico dell'API.

Endpoint API ottimizzati per edge

Un endpoint API ottimizzato per l'edge in genere indirizza le richieste al CloudFront Point of Presence (POP) più vicino, il che può essere utile nei casi in cui i clienti sono distribuiti geograficamente. Questo è il tipo di endpoint predefinito per API Gateway APIs REST.

Ottimizzato per Edge, usa APIs le maiuscole per i nomi delle intestazioni HTTP (ad esempio,). `Cookie`

CloudFront ordina i cookie HTTP in ordine naturale in base al nome del cookie prima di inoltrare la richiesta all'origine. Per ulteriori informazioni sul modo in cui CloudFront elabora i cookie, consulta Memorizzazione nella cache dei contenuti basati sui cookie.

Qualsiasi nome di dominio utilizzato per un'API con edge ottimizzato si applica in tutte le regioni.

Endpoint API regionali

Un endpoint API regionale è destinato a client inclusi nella stessa Regione. L'API regionale riduce il sovraccarico delle connessioni nei casi in cui un client in esecuzione su un'istanza EC2 chiama un'API nella stessa Regione o un'API è destinata all'utilizzo da parte di un numero limitato di client con domanda elevata.

Per un'API regionale, il nome di dominio personalizzato è specifico della Regione in cui viene implementata l'API. Se si distribuisce un'API regionale in più Regioni, questa può avere lo stesso nome di dominio personalizzato in tutte le Regioni. È possibile utilizzare domini personalizzati insieme ad Amazon Route 53 per eseguire attività come l'instradamento basato su latenza. Per ulteriori informazioni, consulta the section called "Configurazione di un nome di dominio personalizzato regionale" e the section called "Configurazione di un nome di dominio personalizzato ottimizzato per l'edge in Gateway API".

Gli endpoint di API regionali passano tutti i nomi di intestazione senza alcuna modifica.

> **Note**
>
> Nei casi in cui i client API siano distribuiti geograficamente, può comunque essere opportuno utilizzare un endpoint API regionale, insieme alla propria CloudFront distribuzione Amazon per garantire che API Gateway non associ l'API a distribuzioni controllate dal servizio. CloudFront Per ulteriori informazioni su questo caso d'uso, vedi Come posso configurare API Gateway con la mia CloudFront distribuzione? .

Endpoint API privati

Un endpoint di API privato è un endpoint di API al quale è possibile accedere solo da Amazon Virtual Private Cloud (VPC) utilizzando un endpoint VPC di interfaccia, ovvero un'interfaccia di rete dell'endpoint creato nel VPC. Per ulteriori informazioni, consulta the section called “REST API private”.

Gli endpoint API privati passano tutti i nomi delle intestazioni senza alcuna modifica.

Modifica di un tipo di endpoint API pubblico o privato in API Gateway

La modifica di un tipo di endpoint API richiede l'aggiornamento della configurazione dell'API. Puoi modificare un tipo di API esistente utilizzando la console API Gateway AWS CLI, o un AWS SDK per API Gateway. Il tipo di endpoint non potrà essere modificato fino a quando non viene completata la modifica corrente ma, durante tale periodo, l’API sarà disponibile.

Sono supportate le seguenti modifiche ai tipi di endpoint:

- Da ottimizzato per l'edge a regionale o privato
- Da regionale a ottimizzato per l'edge o privato
- Da privato a regionale

Non è possibile modificare un'API privata in un'API ottimizzata per i confini.

Se si intende modificare un'API pubblica da ottimizzata per l'edge a regionale o viceversa, è importante notare che un'API ottimizzata per l'edge può avere comportamenti diversi rispetto a un'API regionale. Ad esempio, un'API ottimizzata per i confini rimuove l'intestazione `Content-MD5`. Qualsiasi valore MD5 hash passato al backend può essere espresso in un parametro della stringa

di richiesta o in una proprietà body. Tuttavia, l'API regionale passa questa intestazione, ma può rimappare il nome di intestazione a un altro nome. La comprensione delle differenze può aiutare a scegliere come aggiornare un'API ottimizzata per l'edge in una regionale o un'API regionale in una ottimizzata per l'edge.

Argomenti

- Uso della console API Gateway per modificare un tipo di endpoint API
- Utilizza il per modificare il tipo AWS CLI di endpoint dell'API

Uso della console API Gateway per modificare un tipo di endpoint API

Per modificare il tipo di endpoint API della tua API, esegui uno dei seguenti insiemi di passaggi:

Conversione di un endpoint pubblico da regionale o ottimizzato per l'edge e viceversa

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere una REST API.
3. Scegli Impostazioni API.
4. Nella sezione Dettagli API, scegli Modifica.
5. In Tipo di endpoint API, seleziona Ottimizzato per l'edge o Regionale.
6. Scegli Save changes (Salva modifiche).
7. Ridistribuisci la tua API in modo che le modifiche diventino effettive.

Per convertire un endpoint privato in un endpoint regionale

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere una REST API.
3. Modifica la politica delle risorse per la tua API per rimuovere qualsiasi menzione VPCs o endpoint VPC in modo che le chiamate API dall'esterno del tuo VPC e dall'interno del tuo VPC abbiano esito positivo.
4. Scegli Impostazioni API.
5. Nella sezione Dettagli API, scegli Modifica.
6. In Tipo di endpoint, scegliRegionale.
7. Scegli Save changes (Salva modifiche).
8. Rimuovi la policy delle risorse dall'API.

9. Ridistribuisci la tua API in modo che le modifiche diventino effettive.

   Poiché stai migrando il tipo di endpoint da privato a regionale, API Gateway cambia il tipo di indirizzo IP in. IPv4 Per ulteriori informazioni, consulta the section called “Tipi di indirizzo IP per REST API in Gateway API”.

Per convertire un endpoint regionale in un endpoint privato

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere una REST API.
3. Crea una policy delle risorse che fornisca l'accesso al VPC o all'endpoint VPC. Per ulteriori informazioni, consulta ???.
4. Scegli Impostazioni API.
5. Nella sezione Dettagli API, scegli Modifica.
6. Per Tipo di endpoint API scegli Privato.
7. (Facoltativo) Per l'endpoint VPC IDs, seleziona l'endpoint VPC IDs che desideri associare alla tua API privata.
8. Scegli Save changes (Salva modifiche).
9. Ridistribuisci la tua API in modo che le modifiche diventino effettive.

   Dal momento che si sta migrando il tipo di endpoint da regionale a privato, Gateway API modifica il tipo di indirizzo IP in dualstack. Per ulteriori informazioni, consulta the section called “Tipi di indirizzo IP per REST API in Gateway API”.

Utilizza il per modificare il tipo AWS CLI di endpoint dell'API

Il update-rest-apicomando seguente aggiorna un'API ottimizzata per l'edge in un'API regionale:

```
aws apigateway update-rest-api \
    --rest-api-id a1b2c3 \
    --patch-operations op=replace,path=/endpointConfiguration/types/EDGE,value=REGIONAL
```

La risposta di esito positivo ha il codice di stato `200 OK` e un payload simile al seguente:

```
{
    "createdDate": "2017-10-16T04:09:31Z",
```

```
    "description": "Your first API with Amazon API Gateway. This is a sample API that
 integrates via HTTP with our demo Pet Store endpoints",
    "endpointConfiguration": {
        "types": "REGIONAL"
    },
    "id": "a1b2c3",
    "name": "PetStore imported as edge-optimized"
}
```

Il update-rest-apicomando seguente aggiorna un'API regionale a un'API ottimizzata per l'edge:

```
aws apigateway update-rest-api \
    --rest-api-id a1b2c3 \
    --patch-operations op=replace,path=/endpointConfiguration/types/REGIONAL,value=EDGE
```

Poiché put-rest-apiserve per aggiornare le definizioni delle API, non è applicabile all'aggiornamento di un tipo di endpoint API.

Politiche di sicurezza per le API REST in API Gateway

Una policy di sicurezza è una combinazione predefinita di versione TLS minima e suite di crittografia offerta da Gateway API. Quando i client stabiliscono un handshake TLS sull'API o sul nome di dominio personalizzato, la policy di sicurezza applica la versione di TLS e il pacchetto di crittografia accettato da API Gateway. Le policy di sicurezza proteggono le API e i nomi di dominio personalizzati da problemi di sicurezza della rete, ad esempio manomissioni e eavesdropping tra un client e un server.

API Gateway supporta le politiche di sicurezza precedenti e le politiche di sicurezza avanzate. `TLS_1_0`e `TLS_1_2` sono politiche di sicurezza obsolete. Utilizza queste politiche di sicurezza per la compatibilità con le versioni precedenti. Qualsiasi politica che inizia con `SecurityPolicy_` è una politica di sicurezza avanzata. Utilizza queste policy per carichi di lavoro regolamentati, governance avanzata o per utilizzare la crittografia post-quantistica. Quando utilizzi una policy di sicurezza avanzata, devi anche impostare la modalità di accesso agli endpoint per una governance aggiuntiva. Per ulteriori informazioni, consulta the section called "Modalità di accesso agli endpoint".

In che modo API Gateway applica le politiche di sicurezza

L'esempio seguente mostra come API Gateway applica le politiche di `SecurityPolicy_TLS13_1_3_2025_09` sicurezza utilizzando la politica di sicurezza come esempio.

La politica `SecurityPolicy_TLS13_1_3_2025_09` di sicurezza accetta il traffico TLS 1.3 e rifiuta il traffico TLS 1.2 e TLS 1.0. Per il traffico TLS 1.3, la politica di sicurezza accetta le seguenti suite di crittografia:

- `TLS_AES_128_GCM_SHA256`
- `TLS_AES_256_GCM_SHA384`
- `TLS_CHACHA20_POLY1305_SHA256`

API Gateway non accetta altre suite di crittografia. Ad esempio, la politica di sicurezza rifiuterebbe qualsiasi traffico TLS 1.3 che utilizza la suite di crittografia. `AES128-SHA` Per ulteriori informazioni sulle versioni e le crittografie TLS supportate, vedere. the section called “Policy di sicurezza supportate”

Per monitorare il protocollo TLS e i codici utilizzati dai client per accedere all'API Gateway, puoi utilizzare le variabili `$context.tlsVersion` e di `$context.cipherSuite` contesto nei log di accesso. Per ulteriori informazioni, consulta the section called “Monitoraggio”.

Modalità di accesso agli endpoint

La modalità di accesso agli endpoint è un parametro aggiuntivo che è necessario specificare per qualsiasi REST API o nome di dominio personalizzato che utilizza una politica di sicurezza avanzata che inizia con `SecurityPolicy_`. Lo fai quando crei la tua risorsa o se modifichi la politica di sicurezza da una politica precedente a una politica avanzata.

Quando la modalità di accesso all'endpoint è impostata su`STRICT`, qualsiasi richiesta all'API REST o al nome di dominio personalizzato deve superare i seguenti controlli:

- La richiesta deve provenire dallo stesso tipo di endpoint API Gateway della risorsa. Potrebbe provenire da un endpoint regionale, ottimizzato per l’edge o privato.
- Se utilizzi un endpoint regionale o privato, API Gateway utilizza la corrispondenza degli host SNI. Se utilizzi un endpoint ottimizzato per l'edge, API Gateway è conforme alla protezione del fronting del dominio. CloudFront Per ulteriori informazioni, consulta Domain fronting. https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/CNAMEs.html#alternate-domain-names-restrictions

Se una di queste condizioni non è soddisfatta, API Gateway rifiuta la richiesta. Ti consigliamo di utilizzare la modalità di accesso agli `STRICT` endpoint quando possibile.

Per migrare un'API o un nome di dominio esistente per utilizzare la modalità di accesso rigorosa agli endpoint, aggiorna innanzitutto la tua politica di sicurezza con una politica di sicurezza avanzata e mantieni la modalità di accesso agli endpoint impostata su. `BASIC` Dopo aver convalidato il traffico e i log di accesso, imposta la modalità di accesso agli endpoint su. `STRICT` Quando esegui la migrazione della modalità di accesso all'endpoint da `STRICT` a`BASIC`, l'endpoint non sarà disponibile per circa 15 minuti man mano che le modifiche si propagano.

Non è necessario impostare la modalità di accesso agli endpoint su `STRICT` per determinate architetture applicative, ma impostare invece la modalità di accesso agli endpoint su. `BASIC` La tabella seguente mostra alcune architetture applicative e una raccomandazione per consentire all'API REST o al nome di dominio personalizzato di utilizzare la modalità di accesso agli endpoint. `STRICT`

Architecture	Migrazione consigliata
Utilizzo di un endpoint VPC per accedere a un nome di dominio pubblico personalizzato.	Questa architettura utilizza traffico di tipo cross-endpoint. Ti consigliamo di eseguire la migrazione a. the section called “Nomi di dominio personalizzati per API private”
Utilizzo di qualsiasi metodo per richiamare un'API privata che non utilizza un nome di dominio personalizzato o nomi DNS privati.	Questa architettura crea una discrepanza tra l'intestazione dell'host e l'SNI utilizzato nell'handshake TLS e non passa le restrizioni di fronting del dominio. CloudFront Ti consiglia mo di migrare il tuo VPC per utilizzare un DNS privato.
Utilizzo dello sharding dei domini per distribuire i contenuti su più domini o sottodomini.	Questa architettura crea una discrepanza tra l'intestazione dell'host e l'SNI utilizzato nell'handshake TLS e non supera le restrizioni di fronting del dominio. CloudFront Ti consiglia mo di utilizzare `HTTP/2` e abbandonare questo anti-pattern.

Di seguito sono riportate le considerazioni per l'utilizzo della modalità di accesso agli endpoint:

- Se la modalità di accesso all'endpoint di un'API o di un nome di dominio è`STRICT`, non puoi modificare il tipo di endpoint. Per modificare il tipo di endpoint, modifica prima la modalità di accesso all'endpoint in. `BASIC`
- Dopo aver modificato la modalità di accesso agli endpoint da `BASIC` a`STRICT`, è previsto un ritardo di 15 minuti prima che API Gateway applichi la modalità di accesso rigorosa agli endpoint.
- Quando si modifica una policy di sicurezza da una policy che inizia con `SecurityPolicy_` a una policy precedente, è necessario disattivare la modalità di accesso agli endpoint su. `""`

Considerazioni

Di seguito sono riportate le considerazioni relative alle politiche di sicurezza per le API REST in API Gateway:

- È possibile importare la politica di sicurezza in un file di definizione OpenAPI. Per ulteriori informazioni, consulta x-amazon-apigateway-security-policy.
- La tua API può essere mappata su un nome di dominio personalizzato con una politica di sicurezza diversa dalla tua API. Quando richiami quel nome di dominio personalizzato, API Gateway utilizza la politica di sicurezza del dominio personalizzato per negoziare l'handshake TLS. Se si disabilita l'endpoint API predefinito, si potrebbe influire sul modo in cui i chiamanti possono invocare l'API.
- Se modifichi la politica di sicurezza, sono necessari circa 15 minuti per completare l'aggiornamento. Puoi monitorare `apiStatus` la tua API. Man mano che l'API si aggiorna, lo `apiStatus` è `UPDATING` e quando sarà completata, lo sarà`AVAILABLE`. Se lo stato della tua API è`UPDATING`, puoi comunque richiamarla.
- API Gateway supporta le politiche di sicurezza su tutte le API. Tuttavia, puoi scegliere solo una politica di sicurezza per le API REST. API Gateway supporta solo la politica `TLS_1_2` di sicurezza per HTTP o WebSocket API.
- Non puoi aggiornare la politica di sicurezza per un'API da `TLS_1_0` a`TLS_1_2`.
- Alcune politiche di sicurezza supportano le suite di crittografia ECDSA e RSA. Se utilizzi questo tipo di policy con un nome di dominio personalizzato, le suite di crittografia corrispondono al tipo di chiave di certificato fornito dal cliente, RSA o ECDSA. Se utilizzi questo tipo di policy con un'API REST, le suite di crittografia corrispondono alle suite di crittografia compatibili con i tipi di certificato RSA.

Policy di sicurezza supportate

Le tabelle seguenti descrivono le politiche di sicurezza che possono essere specificate per ogni tipo di endpoint dell'API REST e per ogni tipo di nome di dominio personalizzato. Queste politiche consentono di controllare le connessioni in entrata. API Gateway supporta TLS 1.2 solo in uscita. Puoi aggiornare la politica di sicurezza per la tua API o il nome di dominio personalizzato in qualsiasi momento.

Le politiche contenute `FIPS` nel titolo sono compatibili con il Federal Information Processing Standard (FIPS), uno standard governativo statunitense e canadese che specifica i requisiti di sicurezza per i moduli crittografici che proteggono le informazioni sensibili. Per saperne di più, consulta Federal Information Processing Standard (FIPS) 140 nella pagina AWS Cloud Security Compliance.

Tutte le policy FIPS sfruttano il modulo crittografico convalidato AWS-LC FIPS. Per saperne di più, consulta la pagina del modulo AWS-LC crittografico sul sito del NIST Cryptographic Module Validation Program.

Le policy contenute `PQ` nel titolo utilizzano la Post-Quantum crittografia (PQC) per implementare algoritmi ibridi di scambio di chiavi per TLS al fine di garantire la riservatezza del traffico contro future minacce informatiche quantistiche.

Le policy contenute `PFS` nel titolo utilizzano Perfect Forward Secrecy (PFS) per assicurarsi che le chiavi di sessione non vengano compromesse.

Le policy che le contengono entrambe `FIPS` e `PQ` nel loro titolo supportano entrambe queste funzionalità.

Politiche di sicurezza predefinite

Quando si crea una nuova API REST o un dominio personalizzato, alla risorsa viene assegnata una politica di sicurezza predefinita. La tabella seguente mostra la politica di sicurezza predefinita per queste risorse.

Risorsa	Nome predefinito della politica di sicurezza		
API regionali	TLS_1_0		

Risorsa	Nome predefinito della politica di sicurezza
Edge-optimized API	TLS_1_0
API private	TLS_1_2
Dominio regionale	TLS_1_2
Edge-optimized dominio	TLS_1_2
Dominio privato	TLS_1_2

Politiche di sicurezza supportate per API regionali e private e nomi di dominio personalizzati

La tabella seguente descrive le politiche di sicurezza che possono essere specificate per le API regionali e private e i nomi di dominio personalizzati:

Policy di sicurezza	Versioni TLS supportate	Cifrari supportati
SecurityPolicy_TLS13_1_3_2025_09	TLS1.3	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256
SecurityPolicy_TLS13_1_3_FIPS_2025_09	TLS1.3	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384

Policy di sicurezza	Versioni TLS supportate	Cifrari supportati
SecurityPolicy_TLS13_1_2_FIPS_PFS_PQ_2025_09	TLS1.3 TLS1.2	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384

Policy di sicurezza	Versioni TLS supportate	Cifrari supportati
SecurityPolicy_TLS13_1_2_PFS_PQ_2025_09	TLS1.3 TLS1.2	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384

Policy di sicurezza	Versioni TLS supportate	Cifrari supportati
SecurityPolicy_TLS 13_1_2_PQ_2025_09	TLS1.3 TLS1.2	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • AES128-GCM-SHA256 • AES128-SHA256 • AES256-GCM-SHA384 • AES256-SHA256

Policy di sicurezza	Versioni TLS supportate	Cifrari supportati
SecurityPolicy_TLS13_1_2_2021_06	TLS1.3 TLS1.2	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • AES128-GCM-SHA256 • AES128-SHA256 • AES256-GCM-SHA384 • AES256-SHA256

Policy di sicurezza	Versioni TLS supportate	Cifrari supportati
TLS_1_2	TLS1.3 TLS1.2	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • AES128-GCM-SHA256 • AES128-SHA256 • AES256-GCM-SHA384 • AES256-SHA256

Policy di sicurezza	Versioni TLS supportate	Cifrari supportati
TLS_1_0	TLS1.3 TLS1.2 TLS1.1 TLS1.0	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 TLS1.0-TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES128-SHA • ECDHE-RSA-AES128-SHA • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • ECDHE-ECDSA-AES256-SHA

Policy di sicurezza	Versioni TLS supportate	Cifrari supportati
		• ECDHE-RSA-AES256-SHA • AES128-GCM-SHA256 • AES128-SHA256 • AES128-SHA • AES256-GCM-SHA384 • AES256-SHA256 • AES256-SHA

Policy di sicurezza supportate per API ottimizzate per l'edge e nomi di dominio personalizzati

La tabella seguente descrive le politiche di sicurezza che possono essere specificate per le API ottimizzate per l'edge e i nomi di dominio personalizzati ottimizzati per l'edge:

Nome della politica di sicurezza	Versioni TLS supportate	Cifrari supportati
SecurityPolicy_TLS13_2025_EDGE	TLS1.3	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256
SecurityPolicy_TLS12_PFS_2025_EDGE	TLS1.3 TLS1.2	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256

Nome della politica di sicurezza	Versioni TLS supportate	Cifrari supportati
		TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-ECDSA-CHACHA20-POLY1305 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-RSA-CHACHA20-POLY1305

Nome della politica di sicurezza	Versioni TLS supportate	Cifrari supportati
SecurityPolicy_TLS12_2018_EDGE	TLS1.3 TLS1.2	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-ECDSA-CHACHA20-POLY1305 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-RSA-CHACHA20-POLY1305 • ECDHE-RSA-AES256-SHA384

Nome della politica di sicurezza	Versioni TLS supportate	Cifrari supportati
TLS_1_2 (solo nomi di dominio personalizzati ottimizzati per l'edge)	TLS1.3 TLS1.2	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-ECDSA-CHACHA20-POLY1305 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-RSA-CHACHA20-POLY1305 • ECDHE-RSA-AES256-SHA384

Nome della politica di sicurezza	Versioni TLS supportate	Cifrari supportati
		• AES128-GCM-SHA256 • AES256-GCM-SHA384 • AES128-SHA256

Nome della politica di sicurezza	Versioni TLS supportate	Cifrari supportati
TLS_1_0	TLS1.3 TLS1.2 TLS1.1 TLS1.0	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 TLS1.0-TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES128-SHA • ECDHE-RSA-AES128-SHA • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384

Nome della politica di sicurezza	Versioni TLS supportate	Cifrari supportati
		• ECDHE-ECDSA-AES256-SHA • ECDHE-RSA-AES256-SHA • AES256-SHA256 • AES256-SHA

OpenSSL e nomi crittografia RFC

OpenSSL e IETF RFC 5246 utilizzano nomi diversi per le stesse crittografie. La tabella seguente mappa il nome OpenSSL al nome RFC per ogni crittografia. Per ulteriori informazioni, consultare ciphers nella documentazione OpenSSL.

Nome crittografia OpenSSL	Nome crittografia RFC
TLS_AES_128_GCM_SHA256	TLS_AES_128_GCM_SHA256
TLS_AES_256_GCM_SHA384	TLS_AES_256_GCM_SHA384
TLS_CHACHA20_POLY1305_SHA256	TLS_CHACHA20_POLY1305_SHA256
ECDHE-RSA-AES128-GCM-SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
ECDHE-RSA-AES128-SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256

Nome crittografia OpenSSL	Nome crittografia RFC
ECDHE-RSA-AES128-SHA	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
ECDHE-RSA-AES256-GCM-SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
ECDHE-RSA-AES256-SHA384	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
ECDHE-RSA-AES256-SHA	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
AES128-GCM-SHA256	TLS_RSA_WITH_AES_128_GCM_SHA256
AES256-GCM-SHA384	TLS_RSA_WITH_AES_256_GCM_SHA384
AES128-SHA256	TLS_RSA_WITH_AES_128_CBC_SHA256
AES256-SHA	TLS_RSA_WITH_AES_256_CBC_SHA
AES128-SHA	TLS_RSA_WITH_AES_128_CBC_SHA
DES-CBC3-SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA

Come modificare una politica di sicurezza

Puoi modificare la politica di sicurezza per la tua API. Se invii traffico alle tue API tramite il tuo nome di dominio personalizzato, non è necessario che l'API e il nome di dominio personalizzato abbiano la stessa politica di sicurezza. Quando richiami quel nome di dominio personalizzato, API Gateway utilizza la politica di sicurezza del dominio personalizzato per negoziare l'handshake TLS. Tuttavia, per motivi di coerenza, ti consigliamo di utilizzare la stessa politica di sicurezza per il nome di dominio e l'API personalizzati.

Se modifichi la politica di sicurezza, sono necessari circa 15 minuti per completare l'aggiornamento. Puoi monitorare `apiStatus` la tua API. Man mano che l'API si aggiorna, lo `apiStatus` è `UPDATING` e quando sarà completata, lo sarà`AVAILABLE`. Quando l'API si aggiorna, puoi comunque richiamarla.

Console di gestione AWS

Per modificare la politica di sicurezza di un'API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere una REST API.
3. Scegliere Impostazioni API, quindi scegliere Modifica.
4. Per Politica di sicurezza, seleziona una nuova politica che inizi con`SecurityPolicy_`.
5. Per la modalità di accesso agli endpoint, scegli Strict.
6. Scegli Save changes (Salva modifiche).

   Implementa nuovamente l'API per rendere effettive le modifiche. Poiché hai modificato la modalità di accesso agli endpoint in strict, ci vorranno circa 15 minuti perché le modifiche si propaghino completamente.

AWS CLI

Il seguente comando update-rest-api aggiorna un'API per utilizzare la politica di sicurezza: `SecurityPolicy_TLS13_1_3_2025_09`

```
aws apigateway update-rest-api \
    --rest-api-id abcd1234 \
    --patch-operations '[
        {
            "op": "replace",
            "path": "/securityPolicy",
            "value": "SecurityPolicy_TLS13_1_3_2025_09"
        },
        {
            "op": "replace",
            "path": "/endpointAccessMode",
            "value": "STRICT"
        }
    ]'
```

L'output sarà simile al seguente:

```
{
    "id": "abcd1234",
    "name": "MyAPI",
    "description": "My API with a new security policy",
    "createdDate": "2025-02-04T11:47:06-08:00",
    "apiKeySource": "HEADER",
    "endpointConfiguration": {
        "types": [
            "REGIONAL"
        ],
        "ipAddressType": "dualstack"
    },
    "tags": {},
    "disableExecuteApiEndpoint": false,
    "securityPolicy": "SecurityPolicy_TLS13_1_3_2025_09",
    "endpointAccessMode": "STRICT"
    "rootResourceId": "efg456"
}
```

Il seguente comando update-rest-api aggiorna un'API che utilizzava una politica di sicurezza avanzata per utilizzare la politica di sicurezza. `TLS_1_0`

```
aws apigateway update-rest-api \
    --rest-api-id abcd1234 \
    --patch-operations '[
        {
            "op": "replace",
            "path": "/securityPolicy",
            "value": "TLS_1_0"
        },
        {
            "op": "replace",
            "path": "/endpointAccessMode",
            "value": ""
        }
    ]'
```

L'output sarà simile al seguente:

```
{
```

```
    "id": "abcd1234",
    "name": "MyAPI",
    "description": "My API with a new security policy",
    "createdDate": "2025-02-04T11:47:06-08:00",
    "apiKeySource": "HEADER",
    "endpointConfiguration": {
        "types": [
            "REGIONAL"
        ],
        "ipAddressType": "dualstack"
    },
    "tags": {},
    "disableExecuteApiEndpoint": false,
    "securityPolicy": "TLS_1_0",
    "rootResourceId": "efg456"
}
```

Tipi di indirizzo IP per REST API in Gateway API

Quando si crea un'API, è possibile specificare il tipo di indirizzo IP che può invocarla. È possibile scegliere IPv4 per risolvere gli indirizzi IPv4 per invocare l'API oppure dualstack per consentire a entrambi gli indirizzi IPv4 e IPv6 di invocare l'API. È consigliabile impostare il tipo di indirizzo IP su dualstack per ridurre l'esaurimento dello spazio IP o per il livello di sicurezza. Per ulteriori informazioni sui vantaggi del tipo di indirizzo IP dualstack, consultare IPv6 on AWS.

Per limitare l'API al solo traffico IPv6, è possibile creare una policy di risorse e limitare gli indirizzi IP di origine solo agli intervalli IPv6. È possibile modificare il tipo di indirizzo IP aggiornando la configurazione dell'API. Questa modifica ha effetto immediato e non è necessario implementare nuovamente l'API. Per ulteriori informazioni, consulta the section called "Esempio: negare il traffico API in base all'indirizzo IP di origine o a un intervallo di indirizzi IP"

Considerazioni sui tipi di indirizzo IP

Le seguenti considerazioni potrebbero influire sull'utilizzo dei tipi di indirizzo IP:

- Il tipo di indirizzo IP predefinito per tutte le API regionali e ottimizzate l'edge è IPv4.
- Le API private possono avere solo il tipo di indirizzo IP dualstack.
- Se per un'API esistente si modifica il tipo di indirizzo IP da IPv4 a dualstack, è necessario verificare che tutte le policy che controllano l'accesso alle API siano aggiornate in modo da tenere conto

delle chiamate IPv6. Quando si modifica il tipo di indirizzo IP, la modifica diventa immediatamente effettiva.

- Se si esegue la migrazione del tipo di endpoint di un'API da ottimizzato per l'edge o regionale a privato, Gateway API modifica il tipo di indirizzo IP in dualstack. Per ulteriori informazioni, consulta the section called "Cambia un tipo di endpoint API pubblico o privato".
- Se si esegue la migrazione del tipo di endpoint di un'API da privato a regionale, è necessario impostare il tipo di indirizzo IP su dualstack. Una volta completata la migrazione dell'endpoint, è possibile modificare il tipo di indirizzo IP in IPv4. Per ulteriori informazioni, consulta the section called "Cambia un tipo di endpoint API pubblico o privato".
- L'API può essere mappata su un nome di dominio personalizzato con un tipo di indirizzo IP diverso da quello dell'API. Se si disabilita l'endpoint API predefinito, si potrebbe influire sul modo in cui i chiamanti possono invocare l'API.
- Non è possibile utilizzare un file di definizione esterno per configurare il tipo di indirizzo IP dell'API.

Modifica del tipo di indirizzo IP di una REST API

È possibile modificare il tipo di indirizzo IP aggiornando la configurazione dell'API. È possibile aggiornare la configurazione dell'API utilizzando la Console di gestione AWS, AWS CLI, CloudFormation o un AWS SDK. Se si modifica il tipo di indirizzo IP dell'API, non è necessario eseguire di nuovo l'implementazione dell'API per rendere effettive le modifiche. Prima di modificare il tipo di indirizzo IP, è necessario verificare che tutte le policy che controllano l'accesso alle API siano aggiornate in modo da tenere conto delle chiamate IPv6.

Console di gestione AWS

Per modificare il tipo di indirizzo IP di una REST API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere una REST API.
3. Scegliere Impostazioni API, quindi scegliere Modifica.
4. Per Tipo di indirizzo IP, selezionare IPv4 o Dualstack.
5. Scegli Save changes (Salva modifiche).

   La modifica alla configurazione dell'API ha effetto immediato.

AWS CLI

Il comando update-rest-api seguente aggiorna un'API in modo che abbia un tipo di indirizzo IP dualstack:

```
aws apigateway update-rest-api \
    --rest-api-id abcd1234 \
    --patch-operations "op='replace',path='/endpointConfiguration/
ipAddressType',value='dualstack'"
```

L'output sarà simile al seguente:

```
{
    "id": "abcd1234",
    "name": "MyAPI",
    "description": "My API with a dualstack IP address type",
    "createdDate": "2025-02-04T11:47:06-08:00",
    "apiKeySource": "HEADER",
    "endpointConfiguration": {
        "types": [
            "REGIONAL"
        ],
        "ipAddressType": "dualstack"
    },
    "tags": {},
    "disableExecuteApiEndpoint": false,
    "rootResourceId": "efg456"
}
```

Metodi per REST APIs in API Gateway

In API Gateway un metodo API include una richiesta del metodo e una risposta del metodo. È possibile configurare un metodo API per specificare le operazioni che un client deve eseguire per inviare una richiesta di accesso al servizio nel back-end e le risposte che dovrà ricevere. Per l'input puoi scegliere i parametri di richiesta del metodo o un payload applicabile per consentire al client di fornire i dati obbligatori o facoltativi al runtime. Per l'output si specifica il codice di stato della risposta del metodo, le intestazioni e il corpo applicabile come destinazioni a cui mappare i dati delle risposte di back-end prima che vengano restituite al client. Per aiutare lo sviluppatore del client a capire i comportamenti e i formati di input e output dell'API, puoi documentare l'API e fornire messaggi di errore appropriati per le richieste non valide.

Una richiesta del metodo API è una richiesta HTTP. Per configurare la richiesta del metodo, si configura un metodo (o verbo) HTTP, il percorso di una risorsa API, intestazioni e parametri di stringa di query applicabili. Se il metodo HTTP è `POST`, `PUT` o `PATCH`, devi configurare anche un payload. Ad esempio, per recuperare un animale domestico utilizzando l'API di PetStore esempio, è necessario definire il metodo API request of`GET /pets/{petId}`, where `{petId}` è un parametro di percorso che può richiedere un numero in fase di esecuzione.

```
GET /pets/1
Host: apigateway.us-east-1.amazonaws.com
...
```

Se il client specifica un percorso non corretto, ad esempio `/pet/1` o `/pets/one` anziché `/pets/1`, viene generata un'eccezione.

Una risposta di metodo API è una risposta HTTP con un codice di stato specifico. Per un'integrazione non proxy, è necessario configurare le risposte di metodo per specificare le destinazioni obbligatorie o facoltative delle mappature. Queste trasformano il corpo o le intestazioni delle risposte di integrazione nel corpo o nelle intestazioni delle risposte del metodo associato. La mappatura può essere semplice come la trasformazione di un'identità che trasferisce le intestazioni o il corpo tramite l'integrazione senza alcuna modifica. Ad esempio, la risposta del metodo `200` seguente mostra l'esempio di una risposta di integrazione riuscita che viene passata senza essere modificata.

```
200 OK
Content-Type: application/json
...

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

Inizialmente puoi definire una risposta di metodo che corrisponde a una risposta specifica dal back-end. Generalmente vengono utilizzate le risposte 2XX, 4XX e 5XX. Tuttavia questo approccio potrebbe essere poco pratico perché non sempre si conoscono in anticipo le risposte che un back-end potrebbe restituire. In pratica, è possibile designare una sola risposta del metodo come predefinita per gestire le risposte non note o non mappate dal back-end. È buona norma designare la risposta 500 come predefinita. In ogni caso, devi configurare almeno una risposta del metodo per le

integrazioni non proxy. In caso contrario API Gateway restituisce una risposta di errore 500 al client, anche quando la richiesta al back-end ha esito positivo.

Per fare in modo che l'API supporti un SDK tipizzato in modo sicuro, come un SDK Java, è necessario definire il modello di dati per l'input per le richieste del metodo e il modello di dati per l'output della risposta del metodo.

Prerequisiti

Prima di configurare un metodo API, verifica quanto segue:

- Il metodo deve essere disponibile in API Gateway. Segui le istruzioni in Tutorial: creazione di una REST API con un'integrazione non proxy HTTP.
- Se desideri che il metodo comunichi con una funzione Lambda, devi avere già creato il ruolo di invocazione e il ruolo di esecuzione Lambda in IAM. Devi aver creato anche la funzione Lambda con cui il metodo comunicherà in AWS Lambda. Per creare ruoli e funzioni, usa le istruzioni disponibili nella sezione Creazione di una funzione Lambda per l'integrazione non proxy Lambda dell'argomento Scegli un tutorial di integrazione AWS Lambda.
- Se desideri che il metodo comunichi con un'integrazione HTTP o proxy HTTP, devi avere già creato l'URL dell'endpoint HTTP con cui il metodo comunicherà e disporre del relativo accesso.
- Verifica che i certificati per gli endpoint HTTP e proxy HTTP siano supportati da API Gateway. Per dettagli, consulta Autorità di certificazione supportate da Gateway API per le integrazioni HTTP e proxy HTTP in Gateway API.

Argomenti

- Configurazione di una richiesta del metodo in API Gateway
- Configurazione di una risposta di metodo in Gateway API
- Configurazione di un metodo mediante la console API Gateway

Configurazione di una richiesta del metodo in API Gateway

La configurazione di una richiesta di metodo implica l'esecuzione delle seguenti attività, dopo aver creato una RestApirisorsa:

1. Creazione di una nuova API o scelta di un'entità Resource (Risorsa) di un'API esistente.

2. Creazione di una risorsa Method (Metodo) dell'API rappresentata da un verbo HTTP specifico per l'entità `Resource` dell'API nuova o esistente. Puoi suddividere ulteriormente questa attività svolgendo le operazioni secondarie seguenti:
   - Aggiungi un metodo HTTP alla richiesta del metodo
   - Configura i parametri di richiesta
   - Definisci un modello per il corpo della richiesta
   - Attua uno schema di autorizzazione
   - Abilita la convalida delle richieste

Puoi eseguire queste attività utilizzando i metodi seguenti:

- Console API Gateway
- AWS CLI comandi (create-resource e put-method)
- AWS Funzioni SDK (ad esempio, in Node.js, CreateResource e putMethod)
- API REST API Gateway (resource:create e method:put).

Argomenti

- Impostazione delle risorse API
- Impostazione di un metodo HTTP
- Configurazione dei parametri di richiesta del metodo
- Configurazione di un modello di richiesta del metodo
- Configurazione dell'autorizzazione della richiesta del metodo
- Configurazione della convalida della richiesta del metodo

Impostazione delle risorse API

In un'API di API Gateway le risorse indirizzabili vengono esposte come una struttura di entità Resources (Risorse) dell'API, con la risorsa root (`/`) nella parte più alta della gerarchia. La risorsa radice è relativa rispetto all'URL di base dell'API, costituito dall'endpoint dell'API e da un nome di fase. Nella console API Gateway questo URI di base, visualizzato nell'editor delle fasi dell'API dopo che l'API è stata distribuita, è Invoke URI (URL chiamata).

L'endpoint dell'API può essere un nome host predefinito o un nome di dominio personalizzato. Il formato del nome host predefinito è il seguente:

```
{api-id}.execute-api.{region}.amazonaws.com
```

In questo formato, *{api-id}* rappresenta l'identificatore API generato da API Gateway. La variabile *{region}* rappresenta la regione AWS (ad esempio, `us-east-1`) scelta durante la creazione dell'API. Un nome di dominio personalizzato è un nome intuitivo in un dominio Internet valido. Ad esempio se hai registrato un dominio Internet di `example.com`, qualsiasi `*.example.com` è un nome di dominio personalizzato valido. Per ulteriori informazioni, consulta l'argomento relativo alla creazione di un nome di dominio personalizzato.

Per l'API PetStore di esempio, la risorsa root (`/`) espone il pet store. La risorsa `/pets` rappresenta l'insieme di animali domestici disponibili nel negozio. `/pets/{petId}` espone un singolo animale domestico di un identificatore specificato (`petId`). Il parametro di percorso di `{petId}` fa parte dei parametri di richiesta.

Per configurare una risorsa API, scegli una risorsa esistente come padre e crea la rispettiva risorsa figlio. Inizia con la risorsa radice come padre, aggiungi una risorsa alla risorsa padre, aggiungi un'alta risorsa a questa risorsa figlio come nuovo padre e così via fino all'identificatore del padre. Quindi aggiungi la risorsa con nome alla risorsa padre.

Il comando get-resources seguente recupera tutte le risorse di un'API:

```
aws apigateway get-resources --rest-api-id apiId
```

Per l'API PetStore di esempio, l'output è simile al seguente:

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

Ogni voce elenca gli identificatori della risorsa (`id`) e, ad eccezione della risorsa radice, la rispettiva risorsa padre più prossima (`parentId`), nonché il nome della risorsa (`pathPart`). La risorsa radice ha la particolarità di non avere una risorsa padre. Dopo avere scelto una risorsa come padre, si utilizza il comando seguente per aggiungere una risorsa figlio:

```
aws apigateway create-resource --rest-api-id apiId \
    --parent-id parentId \
    --path-part resourceName
```

Ad esempio, per aggiungere cibo per animali domestici in vendita sul PetStore sito Web, utilizza il seguente comando:

```
aws apigateway create-resource --rest-api-id a1b2c3 \
    --parent-id svzr2028x8 \
    --path-part food
```

L'output sarà simile al seguente:

```
{
    "path": "/food",
    "pathPart": "food",
    "id": "xdsvhp",
    "parentId": "svzr2028x8"
}
```

Uso di una risorsa proxy per semplificare l'impostazione dell'API

Man mano che l'attività cresce, il PetStore proprietario può decidere di mettere in vendita cibo, giocattoli e altri articoli relativi agli animali domestici. A questo scopo, puoi aggiungere `/food`, `/toys` e altre risorse sotto la risorsa radice. Puoi anche aggiungere altre risorse sotto ogni categoria

di vendita, ad esempio `/food/{type}/{item}`, `/toys/{type}/{item}` e così via. Questa operazione può risultare tediosa. Se decidi di aggiungere un livello intermedio `{subtype}` ai percorsi delle risorse per modificare la gerarchia delle risorse in `/food/{type}/{subtype}/{item}`, `/toys/{type}/{subtype}/{item}` e così via, la configurazione dell'API esistente verrà alterata. Per evitare che ciò accada, puoi usare una risorsa proxy di API Gateway per esporre un set di risorse API tutte insieme.

API Gateway definisce una risorsa proxy come segnaposto per consentire di specificare una risorsa quando viene inviata una richiesta. È possibile esprimere una risorsa proxy mediante uno speciale parametro di percorso `{proxy+}`, spesso definito parametro di percorso greedy. Il segno + indica le risorse figlio aggiunte. Il segnaposto `/parent/{proxy+}` indica qualsiasi risorsa che corrisponda al modello di percorso di `/parent/*`. È possibile utilizzare qualsiasi stringa per il nome del parametro di percorso greedy.

Il comando create-resource seguente crea una risorsa proxy sotto la root (`/{proxy+}`):

```
aws apigateway create-resource --rest-api-id apiId \
    --parent-id rootResourceId \
    --path-part {proxy+}
```

L'output sarà simile al seguente:

```
{
    "path": "/{proxy+}",
    "pathPart": "{proxy+}",
    "id": "234jdr",
    "parentId": "svzr2028x8"
}
```

Per l'esempio API `PetStore`, puoi usare `/{proxy+}` per rappresentare sia `/pets`, sia `/pets/{petId}`. Questa risorsa proxy può anche fare riferimento a qualsiasi altra risorsa (esistente o to-be-added)`/food/{type}/{item}`, ad esempio`/toys/{type}/{item}`, ecc., oppure `/food/{type}/{subtype}/{item}/toys/{type}/{subtype}/{item}`, ecc. Lo sviluppatore del back-end stabilisce la gerarchia delle risorse, mentre allo sviluppatore del client spetta il compito di comprenderla. API Gateway si limita a passare al back-end tutto ciò che il client ha inviato.

Un'API può avere più di una risorsa proxy. Ad esempio, le seguenti risorse proxy sono consentite all'interno di un'API, presupponendo che `/parent/{proxy+}` non sia lo stesso elemento padre di `/parent/{child}/{proxy+}`.

```
/{proxy+}
/parent/{proxy+}
/parent/{child}/{proxy+}
```

Quando una risorsa proxy presenta elementi di pari livello non proxy, le risorse di tali elementi vengono escluse dalla rappresentazione della risorsa proxy. Per gli esempi precedenti, `/{proxy+}` si riferisce a qualsiasi risorsa al di sotto della risorsa radice, ad eccezione delle risorse `/parent[/*]`. In altri termini, una richiesta di metodo a una risorsa specifica ha la precedenza rispetto a una richiesta di metodo a una risorsa generica allo stesso livello della gerarchia di risorse.

La tabella seguente mostra come Gateway API instrada le richieste alle seguenti risorse per la fase `prod` di un'API.

```
ANY /{proxy+}
GET /pets/{proxy+}
GET /pets/dog
```

Richiesta	Percorso selezionato	Spiegazione
`GET https://api-id.execute-api. region.amazonaws.com/prod/pets/dog`	`GET /pets/dog`	La richiesta corrisponde completamente alla risorsa.
`GET https://api-id.execute-api. region.amazonaws.com/prod/pets/cats`	`GET /pets/{proxy+}`	La variabile di percorso greedy `/pets/{proxy+}` rileva questa richiesta.
`GET https://api-id.execute-api. region.amazonaws.com/prod/animals`	`GET /{proxy+}`	La variabile di percorso greedy `/{proxy+}` rileva questa richiesta.

Una risorsa proxy non può avere risorse figlio. Qualsiasi risorsa API dopo `{proxy+}` è ridondante e ambigua. Di seguito sono riportate risorse poxy non consentite in un'API.

```
/{proxy+}/child
/parent/{proxy+}/{child}
/parent/{child}/{proxy+}/{grandchild+}
```

Impostazione di un metodo HTTP

Una richiesta del metodo API viene incapsulata dalla risorsa Method (Metodo) di API Gateway. Per configurare la richiesta del metodo, è necessario prima creare un'istanza della risorsa `Method` impostando almeno un metodo HTTP e un tipo di autorizzazione per il metodo.

Strettamente associato alla risorsa proxy, API Gateway supporta un metodo HTTP di `ANY`. Questo metodo `ANY` rappresenta qualsiasi metodo HTTP da fornire al runtime. Consente di utilizzare la configurazione di un solo metodo API per tutti i metodi HTTP supportati di `DELETE`, `GET`, `HEAD`, `OPTIONS`, `PATCH`, `POST` e `PUT`.

Puoi configurare il metodo `ANY` anche per una risorsa non proxy. Combinando il metodo `ANY` con una risorsa proxy, ottieni un'unica configurazione di un metodo API per tutti i metodi HTTP supportati con tutte le risorse di un'API. Inoltre il back-end può evolvere senza alterare la configurazione API esistente.

Prima di configurare un metodo API, valuta chi può chiamare il metodo. Imposta il tipo di autorizzazione in base al tuo piano. Per l'accesso aperto, impostalo su `NONE`. Per usare le autorizzazioni IAM, imposta il tipo di autorizzazione su `AWS_IAM`. Per usare una funzione di autorizzazione Lambda, imposta questa proprietà su `CUSTOM`. Per utilizzare un pool di utenti di Amazon Cognito, imposta il tipo di autorizzazione su `COGNITO_USER_POOLS`.

Il comando put-method seguente crea una richiesta di metodo per il verbo `ANY` utilizzando le autorizzazioni IAM per controllare l'accesso.

```
aws apigateway put-method --rest-api-id vaz7da96z6 \
    --resource-id 6sxz2j \
    --http-method ANY \
    --authorization-type AWS_IAM
```

Per creare una richiesta del metodo API con un tipo di autorizzazione diverso, consulta the section called "Configurazione dell'autorizzazione della richiesta del metodo".

Configurazione dei parametri di richiesta del metodo

I parametri di richiesta del metodo consentono a un client di fornire i dati di input o il contesto di esecuzione necessari per completare la richiesta del metodo. Un parametro di metodo può essere un parametro di percorso, un'intestazione o un parametro stringa di query. Nell'ambito della configurazione della richiesta del metodo, è necessario dichiarare i parametri di richiesta obbligatori per renderli disponibili per il client. Per l'integrazione non proxy, puoi convertire questi parametri di richiesta in un formato compatibile con i requisiti di back-end.

Ad esempio, per la richiesta del metodo `GET /pets/{petId}`, la variabile di percorso `{petId}` è un parametro di richiesta obbligatorio. Puoi dichiarare questo parametro di percorso quando chiami il comando `put-method` di AWS CLI, Il comando put-method seguente crea un metodo con un parametro di percorso obbligatorio:

```
aws apigateway put-method --rest-api-id vaz7da96z6 \
    --resource-id rjkmth \
    --http-method GET \
    --authorization-type "NONE" \
    --request-parameters method.request.path.petId=true
```

Se un parametro non è obbligatorio, puoi impostarlo su `false` in `request-parameters`. Ad esempio, se il metodo `GET /pets` usa un parametro della stringa di query `type` e un parametro dell'intestazione `age` facoltativi, è possibile dichiararli utilizzando il comando put-method seguente:

```
aws apigateway put-method --rest-api-id vaz7da96z6 \
    --resource-id 6sxz2j \
    --http-method GET \
    --authorization-type "NONE" \
    --request-parameters
 method.request.querystring.type=false,method.request.header.age=false
```

Per impostare il valore `request-parameters`, anziché questo formato abbreviato, puoi usare una stringa JSON:

```
'{"method.request.querystring.type":false,"method.request.header.age":false}'
```

Con questa configurazione, il client può eseguire query sugli animali domestici in base al tipo:

```
GET /pets?type=dog
```

Il client può eseguire la query sui cani cuccioli come segue:

```
GET /pets?type=dog
age:puppy
```

Per informazioni su come mappare i parametri di richiesta dei metodi ai parametri di richiesta di integrazione, consulta the section called “Integrazioni”.

Configurazione di un modello di richiesta del metodo

Per un metodo API che accetta dati di input in un payload, puoi usare un modello. Un modello viene espresso in una bozza 4 dello schema JSON e descrive la struttura dati del corpo della richiesta. Utilizzando un modello, un client può stabilire come costruire un payload di richiesta del metodo come input. Cosa ancora più importante è che API Gateway usa il modello per convalidare una richiesta, generare un SDK e inizializzare un modello di mappatura per la configurazione dell'integrazione nella console API Gateway. Per informazioni su come creare un modello, consultare Informazioni sui modelli di dati.

A seconda del tipo di contenuto, un payload del metodo può avere formati diversi. Un modello viene indicizzato in base al tipo di supporto del payload applicato. API Gateway utilizza l'intestazione di richiesta `Content-Type` per determinare il tipo di contenuto. Per impostare i modelli di richiesta del metodo, aggiungi coppie chiave-valore del `"`*`media-type`*`":"`*`model-name`*`"` formato alla `requestModels` mappa quando chiami il AWS CLI `put-method` comando.

Per utilizzare lo stesso modello indipendentemente dal tipo di contenuto, specifica `$default` come chiave.

Ad esempio, per impostare un modello sul payload JSON della richiesta del `POST /pets` metodo dell'API di PetStore esempio, puoi utilizzare il seguente comando put-method:

```
aws apigateway put-method \
    --rest-api-id vaz7da96z6 \
    --resource-id 6sxz2j \
    --http-method POST \
    --authorization-type "NONE" \
    --request-models '{"application/json":"petModel"}'
```

In questo esempio `petModel` è il valore della proprietà `name` di una risorsa `Model` che descrive un animale domestico. La definizione effettiva dello schema viene espressa come valore di stringa JSON della proprietà `schema` della risorsa `Model`.

In un SDK Java o in un altro SDK tipizzato in modo sicuro dell'API i dati di input vengono trasmessi come classe `petModel` derivata dalla definizione dello schema. Con il modello della richiesta i dati di input nell'SDK generato vengono trasmessi nella classe `Empty`, che è derivata dal modello `Empty` predefinito. In questo caso il client non può creare istanze della classe di dati corretta per fornire l'input richiesto.

Configurazione dell'autorizzazione della richiesta del metodo

Per controllare chi può chiamare il metodo API, puoi configurare il tipo di autorizzazione per il metodo. Puoi utilizzare questo tipo per implementare una delle autorizzazioni supportate, inclusi i ruoli e le policy IAM (`AWS_IAM`), un pool di utenti di Amazon Cognito (`COGNITO_USER_POOLS`) o un'autorizzazione Lambda (`CUSTOM`).

Per utilizzare le autorizzazioni IAM per consentire l'accesso al metodo API, imposta la proprietà di input `authorization-type` su **`AWS_IAM`**. Se si imposta questa opzione, API Gateway verifica la firma del chiamante sulla richiesta in base alle credenziali del chiamante. Se l'utente verificato dispone dell'autorizzazione per chiamare il metodo, la richiesta viene accettata. In caso contrario, la richiesta viene rifiutata e il chiamante riceve una risposta di errore. La chiamata al metodo ha esito positivo solo se il chiamante dispone del permesso di richiamare il metodo API. La seguente policy IAM concede al chiamante l'autorizzazione a chiamare qualsiasi metodo API creato all'interno dello stesso Account AWS:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "execute-api:Invoke"
            ],
            "Resource": "arn:aws:execute-api:*:*:*"
        }
    ]
}
```

Per ulteriori informazioni, consulta the section called “Uso delle autorizzazioni IAM”.

Attualmente, è possibile concedere questa policy solo agli utenti, ai gruppi e ai ruoli all'interno dell' Account AWS del proprietario dell'API. Gli utenti di un altro soggetto Account AWS possono chiamare i metodi dell'API solo se autorizzati ad assumere un ruolo all'interno del proprietario dell'API Account AWS con le autorizzazioni necessarie per richiamare l'azione. `execute-api:Invoke` Per informazioni sulle autorizzazioni tra account, consulta l'argomento relativo all'uso dei ruoli IAM.

Puoi utilizzare AWS CLI un AWS SDK o un client API REST, come Postman, che implementa la firma Signature Version 4 (SigV4).

Per utilizzare un autorizzatore Lambda allo scopo di autorizzare l'accesso al metodo API, imposta la proprietà di input `authorization-type` su `CUSTOM` e la proprietà di input `authorizer-id` sul valore della proprietà `id` di un autorizzatore Lambda esistente. L'autorizzazione Lambda di riferimento può essere di tipo `TOKEN` o `REQUEST`. Per informazioni su come creare un'autorizzazione Lambda, consulta the section called “Uso di autorizzazioni Lambda”.

Per usare un bacino d'utenza di Amazon Cognito per autorizzare l'accesso al metodo API, imposta la proprietà di input `authorization-type` su `COGNITO_USER_POOLS` e la proprietà di input `authorizer-id` sul valore della proprietà `id` dell'autorizzatore `COGNITO_USER_POOLS` già creato. Per informazioni sulla creazione di un'autorizzazione per il bacino d’utenza di Amazon Cognito, consulta the section called “Utilizza un pool di utenti di Amazon Cognito come autorizzazione per un'API REST”.

Configurazione della convalida della richiesta del metodo

Puoi abilitare la convalida della richiesta durante la configurazione di una richiesta del metodo API. È necessario creare prima un validatore richiesta: Il create-request-validatorcomando seguente crea un validatore di richieste solo per il corpo.

```
aws apigateway create-request-validator \
    --rest-api-id 7zw9uyk9kl \
    --name bodyOnlyValidator \
    --validate-request-body  \
    --no-validate-request-parameters
```

L'output sarà simile al seguente:

```
{
    "validateRequestParameters": false,
    "validateRequestBody": true,
```

```
    "id": "jgpyy6",
    "name": "bodyOnlyValidator"
}
```

Con questo validatore richiesta è possibile utilizzare la convalida della richiesta nell'ambito della configurazione della richiesta di metodo: Il comando put-method seguente crea una richiesta di metodo che richiede che il corpo della richiesta in entrata corrisponda a `PetModel` e include due parametri di richiesta non obbligatori:

```
aws apigateway put-method \
    --rest-api-id 7zw9uyk9kl \
    --resource-id xdsvhp \
    --http-method PUT \
    --authorization-type "NONE" \
    --request-parameters '{"method.request.querystring.type": false,
 "method.request.querystring.page":false}' \
    --request-models '{"application/json":"petModel"}' \
    --request-validator-id jgpyy6
```

Per includere un parametro di richiesta nella convalida della richiesta, è necessario impostare `validateRequestParameters` su `true` per il validatore richiesta e impostare il parametro di richiesta specifico su `true` nel comando `put-method`.

Configurazione di una risposta di metodo in Gateway API

Una risposta del metodo API incapsula l'output di una richiesta del metodo API che verrà ricevuta dal client. I dati di output includono un codice di stato HTTP, alcune intestazioni ed eventualmente un corpo.

Con le integrazioni non proxy, i parametri di risposta specificati e il corpo possono essere mappati dai dati della risposta di integrazione associati oppure possono essere assegnati determinati valori statici in base alle mappature. Queste mappature vengono specificate nella risposta di integrazione. La mappatura può essere una trasformazione identica che passa attraverso l'integrazione senza alcuna modifica.

Con un'integrazione, API Gateway passa automaticamente la risposta di back-end alla risposta del metodo. Non è necessario configurare la risposta del metodo API. Tuttavia, con l'integrazione proxy Lambda, la funzione Lambda deve restituire un risultato in questo formato di output per consentire ad API Gateway di mappare la risposta di integrazione a una risposta del metodo.

A livello di codice, la configurazione della risposta del metodo equivale alla creazione di una MethodResponserisorsa di API Gateway e all'impostazione delle proprietà di StatusCode, ResponseParameters e ResponseModels.

Quando si impostano i codici di stato per un metodo API, è necessario sceglierne uno come predefinito per gestire le risposte di integrazione di un codice di stato non anticipato. È accettabile impostare `500` come valore predefinito perché equivale a trasmettere risposte non mappate come errore del server. Per fini dimostrativi, la console API Gateway imposta come predefinita la risposta `200`. Tuttavia puoi reimpostare il valore su `500`.

Per configurare una risposta di metodo, devi aver creato la richiesta.

Configurazione dello stato del codice di una risposta di metodo

Lo stato del codice di una risposta di metodo definisce un tipo di risposta. Ad esempio, le risposte 200, 400 e 500 indicano rispettivamente riposte completate, errore lato client ed errore lato server.

Per configurare un codice di stato della risposta del metodo, imposta la proprietà `statusCode` su un codice di stato HTTP. put-method-responseIl comando seguente crea la risposta del metodo`200`.

```
aws apigateway put-method-response \
    --rest-api-id vaz7da96z6 \
    --resource-id 6sxz2j \
    --http-method GET \
    --status-code 200
```

Configurazione dei parametri di risposta del metodo

I parametri di risposta del metodo definiscono le intestazioni che il client deve ricevere in risposta alla richiesta di metodo associata. Specificano inoltre una destinazione a cui API Gateway mappa un parametro di risposta di integrazione in base alle mappature prescritte nella risposta di integrazione del metodo API.

Per configurare i parametri di risposta del metodo, aggiungi alla mappa `responseParameters` di `MethodResponse` coppie chiave-valore nel formato `"{parameter-name}":"{boolean}"`. Il put-method-responsecomando seguente imposta l'`my-header`intestazione.

```
aws apigateway put-method-response \
        --rest-api-id vaz7da96z6 \
        --resource-id 6sxz2j \
        --http-method GET \
```

```
        --status-code 200  \
        --response-parameters method.response.header.my-header=false
```

Configurazione dei modelli di risposta del metodo

Un modello di risposta del metodo definisce un formato del corpo della risposta. La configurazione di un modello di risposta del metodo è necessaria quando si genera un SDK tipizzato in modo sicuro per l'API. Garantisce che l'output venga trasmesso in una classe appropriata in Java o Objective-C. In altri casi l'impostazione di un modello è facoltativa.

Prima di configurare il modello di risposta, è necessario creare il modello in API Gateway. A questo scopo, chiama il comando `create-model`. Il comando create-model seguente mostra come creare un modello `PetStorePet` per descrivere il corpo della risposta nella richiesta di metodo `GET /pets/{petId}`.

```
aws apigateway create-model \
    --rest-api-id vaz7da96z6 \
    --content-type application/json \
    --name PetStorePet \
    --schema '{ \
                "$schema": "http://json-schema.org/draft-04/schema#", \
                "title": "PetStorePet", \
                "type": "object", \
                "properties": { \
                  "id": { "type": "number" }, \
                  "type": { "type": "string" }, \
                  "price": { "type": "number" } \
                } \
             }'
```

Il risultato viene creato come una risorsa `Model` di API Gateway.

Per configurare i modelli di risposta del metodo per definire il formato del payload, aggiungi la coppia chiave-valore «application/json»:» PetStorePet "alla mappa della risorsa. `requestModelsMethodResponse` Il put-method-responsecomando seguente crea un metodo di risposta che utilizza un modello di risposta per definire il formato del payload:

```
aws apigateway put-method-response \
    --rest-api-id vaz7da96z6 \
    --resource-id 6sxz2j \
    --http-method GET \
```

```
    --status-code 200  \
    --response-parameters method.response.header.my-header=false \
    --response-models '{"application/json":"PetStorePet"}'
```

Configurazione di un metodo mediante la console API Gateway

Quando si crea un metodo utilizzando la console REST API, si configura sia la richiesta di integrazione sia la richiesta del metodo. Per impostazione predefinita, Gateway API crea la risposta `200` per il metodo.

Le seguenti istruzioni mostrano come modificare le impostazioni di richiesta del metodo e come creare risposte aggiuntive per il metodo.

Argomenti

- Modifica di una richiesta del metodo di Gateway API nella console Gateway API
- Configurazione di una risposta del metodo di API Gateway tramite console API Gateway

Modifica di una richiesta del metodo di Gateway API nella console Gateway API

Queste istruzioni presuppongono che la richiesta del metodo sia già stata creata. Per ulteriori informazioni su come creare un metodo, consulta the section called “ Configurazione di una richiesta di integrazione tramite la console”.

1. Seleziona il metodo nel riquadro Risorse, quindi scegli la scheda Richiesta metodo.
2. Nella sezione Impostazioni della richiesta del metodo scegli Modifica.
3. Per Autorizzazione seleziona un sistema di autorizzazione disponibile.

    a. Per abilitare l'accesso aperto al metodo per qualsiasi utente, scegli Nessuno. Questa fase può essere ignorata se l'impostazione predefinita non è stata modificata.

    b. Per utilizzare le autorizzazioni IAM per controllare l'accesso del client al metodo, scegli `AWS_IAM`. In questo modo il metodo potrà essere chiamato solo dagli utenti dei ruoli IAM a cui è collegata la policy IAM corretta.

    Per creare il ruolo IAM, specifica una policy di accesso con un formato simile al seguente:

    JSON

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:111111111111:aaabbb/*/GET/"
      ]
    }
  ]
}
```

In questa policy di accesso, *arn:aws:execute-api:us-east-1:111111111111:aaabbb/*/GET/* è l'ARN del metodo. Per trovare l'ARN del metodo, seleziona il metodo nella pagina Risorse. Per ulteriori informazioni sull'impostazione delle autorizzazioni IAM, consulta Controllo degli accessi a una REST API con le autorizzazioni IAM.

Per creare il ruolo IAM è possibile adattare le istruzioni disponibili nel tutorial ???.

c. Per utilizzare un sistema di autorizzazione Lambda, seleziona un token o un sistema di autorizzazione di richiesta. Affinché questa scelta sia visualizzata nel menu a discesa, è necessario creare un sistema di autorizzazione Lambda. Per informazioni su come creare un'autorizzazione Lambda, consulta Uso di autorizzazioni Lambda di API Gateway.

d. Per utilizzare un pool di utenti di Amazon Cognito, sceglierne uno disponibile in Cognito user pool authorizers (Autorizzazioni pool di utenti Cognito). Affinché questa scelta sia visualizzata nel menu a discesa, è necessario creare un pool di utenti in Amazon Cognito e un'autorizzazione del pool di utenti di Amazon Cognito in API Gateway. Per informazioni su come creare un'autorizzazione del pool di utenti di Amazon Cognito, consulta Controllo degli accessi alle REST API utilizzando pool di utenti di Amazon Cognito come sistema di autorizzazione.

4. Per specificare la convalida della richiesta, seleziona un valore dal menu a discesa Validatore richiesta. Per disattivare la convalida della richiesta, seleziona Nessuno. Per ulteriori informazioni su ciascuna opzione, consulta Convalida delle richieste per REST API in Gateway API.

5. Seleziona Chiave API necessaria per richiedere una chiave API. Quando sono abilitate, le chiavi API vengono utilizzate nei piani di utilizzo per limitare il traffico client.

6. (Facoltativo) Per assegnare un nome di operazione in un SDK Java di questa API generata da Gateway API, immetti un nome in Nome operazione. Ad esempio, per la richiesta del metodo `GET /pets/{petId}`, il nome di operazione dell'SDK Java corrispondente è `GetPetsPetId` per impostazione predefinita. Questo nome viene costruito dal verbo HTTP del metodo (`GET`) e dai nomi delle variabili di percorso della risorsa (`Pets` e `PetId`). Se imposti il nome dell'operazione come `getPetById`, il nome di operazione dell'SDK diventa `GetPetById`.
7. Per aggiungere un parametro stringa di query al metodo, procedi come indicato di seguito:

   a. Scegli Parametri della stringa di query URL, quindi seleziona Aggiungi stringa di query.

   b. In Nome digita il nome del parametro della stringa di query.

   c. Se il parametro della stringa di query appena creato viene usato per la convalida della richiesta, scegli l'opzione Obbligatorio. Per ulteriori informazioni sulla convalida della richiesta, consulta Convalida delle richieste per REST API in Gateway API.

   d. Se il parametro della stringa di query appena creato viene usato come parte di una chiave di caching, seleziona l'opzione Caching. Per ulteriori informazioni sul caching, consulta Uso dei parametri di metodo o di integrazione come chiavi di cache per indicizzare le risposte memorizzate nella cache.

   Per rimuovere il parametro della stringa di query, scegli Rimuovi.
8. Per aggiungere un parametro di intestazione al metodo, procedi come indicato di seguito:

   a. Scegli Intestazioni di richiesta HTTP, quindi seleziona Aggiungi intestazione.

   b. Per Nome immetti il nome dell'intestazione.

   c. Se l'intestazione appena creata viene usata per la convalida della richiesta, scegli l'opzione Obbligatorio. Per ulteriori informazioni sulla convalida della richiesta, consulta Convalida delle richieste per REST API in Gateway API.

   d. Se l'intestazione appena creata viene usata come parte di una chiave di caching, scegli l'opzione Caching. Per ulteriori informazioni sul caching, consulta Uso dei parametri di metodo o di integrazione come chiavi di cache per indicizzare le risposte memorizzate nella cache.

   Per rimuovere l'intestazione, scegli Rimuovi.
9. Per dichiarare il formato del payload di una richiesta di metodo con il verbo HTTP `POST`, `PUT` o `PATCH`, espandi Corpo della richiesta e procedi come segue:

a. Scegliere Add model (Aggiungi modello).

b. Per Content-Type immetti un tipo MIME (ad esempio `application/json`).

c. Per Modello seleziona un modello dal menu a discesa. I modelli attualmente disponibili per l'API includono i modelli `Empty` ed `Error` predefiniti e tutti i modelli creati e aggiunti alla raccolta Models (Modelli) dell'API. Per ulteriori informazioni sulla creazione di un modello, consulta Modelli di dati per REST API.

> **Note**
>
> Il modello è utile per informare il client del formato dati previsto di un payload, nonché per generare un modello di mappatura. È importante generare un SDK tipizzato in modo sicuro dell'API in linguaggi come Java, C#, Objective-C e Swift. È obbligatorio solo se la convalida di richiesta è abilitata per il payload.

10. Scegli Save (Salva).

Configurazione di una risposta del metodo di API Gateway tramite console API Gateway

Un metodo API può avere una o più risposte. Ogni risposta è indicizzata dal rispettivo codice di stato HTTP. Per impostazione predefinita, la console API Gateway aggiunge la risposta `200` alle risposte di metodo. Puoi modificarla in modo che il metodo restituisca `201`. Puoi aggiungere altre risposte, ad esempio `409` per negare l'accesso e `500` per le variabili di fase non inizializzate utilizzate.

Per utilizzare la console API Gateway per modificare, eliminare o aggiungere una risposta a un metodo API, segui queste istruzioni.

1. Seleziona il metodo nel riquadro Risorse, quindi scegli la scheda Risposta metodo. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. Nella sezione Impostazioni risposta metodo scegli Crea risposta.
3. Per Codice di stato HTTP inserisci un codice di stato HTTP come `200`, `400` o `500`.

   Se per una risposta restituita dal back-end non è definita una risposta del metodo corrispondente, API Gateway non è in grado di restituire la risposta al client. Restituisce invece una risposta di errore `500 Internal server error`.
4. Seleziona Add header (Aggiungi intestazione).
5. Per Nome intestazione immetti un nome.

Per restituire un'intestazione dal back-end al client, aggiungi l'intestazione nella risposta del metodo.

6. Scegli Aggiungi modello per definire un formato del corpo della risposta del metodo.

   Immetti il tipo di supporto del payload della risposta per Tipo di contenuto e seleziona un modello dal menu a discesa Modelli.

7. Scegli Save (Salva).

Per modificare una risposta esistente, passa alla risposta del metodo e scegli Modifica. Per modificare il codice di stato HTTP, scegli Elimina e crea una nuova risposta del metodo.

Per ogni risposta restituita dal back-end, deve essere configurata una risposta compatibile come risposta del metodo. Tuttavia le intestazioni delle risposte del metodo di configurazione e il modello di payload sono facoltativi, a meno che non si mappi il risultato dal back-end alla risposta del metodo prima della restituzione al client. Inoltre un modello di payload della risposta è importante quando si genera un SDK tipizzato in modo sicuro per l'API.

Controlla e gestisci l'accesso a REST APIs in API Gateway

API Gateway supporta più meccanismi per controllare e gestire l'accesso all'API.

Puoi usare i seguenti meccanismi per l'autenticazione e l'autorizzazione:

- Le policy relative alle risorse ti consentono di creare policy basate sulle risorse per consentire o negare l'accesso ai tuoi metodi APIs e ai tuoi metodi da indirizzi IP di origine o endpoint VPC specifici. Per ulteriori informazioni, consulta the section called “Uso delle policy delle risorse API Gateway”.
- I ruoli e le policy AWS IAM standard offrono controlli di accesso flessibili e robusti che possono essere applicati a un'intera API o a singoli metodi. I ruoli e le policy IAM possono essere utilizzati per controllare chi può creare e gestire APIs i tuoi e chi può richiamarli. Per ulteriori informazioni, consulta the section called “Uso delle autorizzazioni IAM”.
- I tag IAM possono essere utilizzati insieme alle policy IAM per controllare l'accesso. Per ulteriori informazioni, consulta the section called “Attribute-based controllo degli accessi”.
- Le policy degli endpoint per gli endpoint VPC di interfaccia ti consentono di collegare le policy delle risorse IAM agli endpoint VPC di interfaccia per migliorare la sicurezza del tuo ambiente privato. APIs Per ulteriori informazioni, consulta the section called “Utilizzo di policy di endpoint VPC per API private”.

- Le autorizzazioni Lambda sono funzioni Lambda che controllano l'accesso ai metodi API REST usando l'autenticazione con token di connessione, nonché le informazioni descritte da intestazioni, percorsi, stringhe di query, variabili di fase o parametri di richiesta di variabili di contesto. Le autorizzazioni Lambda vengono utilizzate per controllare chi può richiamare i metodi API REST. Per ulteriori informazioni, consulta the section called “Uso di autorizzazioni Lambda”.
- I pool di utenti di Amazon Cognito ti consentono di creare soluzioni di autenticazione e autorizzazione personalizzabili per il tuo REST. APIs I pool di utenti di Amazon Cognito vengono utilizzati per controllare chi può richiamare i metodi delle API REST. Per ulteriori informazioni, consulta the section called “Utilizza un pool di utenti di Amazon Cognito come autorizzazione per un'API REST”.

Puoi utilizzare i seguenti meccanismi per l'esecuzione di altre attività correlate al controllo degli accessi:

- La funzionalità Cross-origin resource sharing (CORS) consente di controllare il modo in cui l'API REST risponde alle richieste di risorse multidominio. Per ulteriori informazioni, consulta the section called “NUCLEI”.
- I certificati SSL lato client possono essere utilizzati per verificare che le richieste HTTP al sistema back-end provengano da API Gateway. Per ulteriori informazioni, consulta the section called “Certificati API Gateway-generated ”.
- AWS WAF può essere utilizzato per proteggere l'API di API Gateway da exploit di Web comuni. Per ulteriori informazioni, consulta the section called “AWS WAF”.

Puoi utilizzare i seguenti meccanismi per monitorare e limitare l'accesso che è stato consentito ai client autorizzati:

- I piani di utilizzo consentono di fornire chiavi API ai clienti e quindi di monitorare e limitare l'utilizzo delle fasi e dei metodi API per ciascuna chiave API. Per ulteriori informazioni, consulta the section called “Piani di utilizzo”.

Controllo degli accessi a una REST API con le policy delle risorse Gateway API

Le policy delle risorse di Gateway Amazon API sono documenti di policy JSON collegati a un'API per controllare se un principale specificato (in genere un gruppo o un ruolo IAM) può richiamare l'API. È possibile usare le policy di risorse API Gateway per consentire alle API di essere chiamate in modo sicuro da:

- Utenti di un AWS account specificato.
- Intervalli di indirizzi IP sorgente specificati o blocchi CIDR.
- Cloud privati virtuali (VPCs) o endpoint VPC specificati (in qualsiasi account).

È possibile allegare una policy di risorse per qualsiasi tipo di endpoint API in API Gateway utilizzando Console di gestione AWS la AWS CLI o. AWS SDKs In ambito privato APIs, puoi utilizzare le policy delle risorse insieme alle policy degli endpoint VPC per controllare quali principali hanno accesso a quali risorse e azioni. Per ulteriori informazioni, consulta the section called “Utilizzo di policy di endpoint VPC per API private”.

Le policy delle risorse API Gateway sono diverse dalle policy IAM basate su identità. Le policy IAM basate su identità sono collegate a utenti, gruppi o ruoli IAM e definiscono le operazioni che tali identità sono in grado di eseguire e su quali risorse. Le policy delle risorse API Gateway sono collegate alle risorse. L'utilizzo combinato delle policy delle risorse API Gateway e delle policy IAM è consentito. Per ulteriori informazioni, consultare Identity-Based Policies and Resource-Based Policies.

Argomenti

- Panoramica della sintassi delle policy di accesso per Amazon API Gateway
- Come le policy delle risorse API Gateway influiscono sul flusso di lavoro delle autorizzazioni
- Esempi di policy delle risorse API Gateway
- Creazione e collegamento di una policy delle risorse API Gateway a un'API
- AWS chiavi di condizione che possono essere utilizzate nelle politiche delle risorse di API Gateway

Panoramica della sintassi delle policy di accesso per Amazon API Gateway

Questa pagina descrive gli elementi di base usati nelle policy per le risorse Amazon API Gateway.

Le policy delle risorse vengono specificate utilizzando la stessa sintassi delle policy IAM. Per informazioni complete sulla sintassi delle policy, consulta Panoramica delle policy IAM e Riferimento alle policy AWS Identity and Access Management nella Guida per l'utente di IAM.

Per informazioni su come un AWS servizio decide se una determinata richiesta deve essere consentita o rifiutata, vedere Determinare se una richiesta è consentita o rifiutata.

Elementi comuni in una policy di accesso

In termini basilari, una policy delle risorse contiene i seguenti elementi:

- Risorse: APIs sono le risorse di Amazon API Gateway per le quali puoi consentire o negare le autorizzazioni. In una policy, utilizzare l'Amazon Resource Name (ARN) per identificare la risorsa. È inoltre possibile utilizzare la sintassi abbreviata, che API Gateway espande automaticamente fino all'ARN completo quando si salva una policy delle risorse. Per ulteriori informazioni, consulta Esempi di policy delle risorse API Gateway.

  Per il formato dell'elemento `Resource` completo, consulta Formato dell'elemento Resource delle autorizzazioni per l'esecuzione dell'API in API Gateway.

- Azioni: per ogni risorsa, Amazon API Gateway supporta un insieme di operazioni. Vengono identificate le operazioni delle risorse che verranno consentite (o rifiutate) utilizzando le parole chiave dell'operazione.

  Ad esempio, l'autorizzazione `execute-api:Invoke` consentirà all'utente di richiamare un'API in seguito alla richiesta di un client.

  Per il formato dell'elemento `Action`, consulta Formato dell'elemento Action delle autorizzazioni per l'esecuzione dell'API in API Gateway.

- Effetto: l'effetto prodotto quando l'utente richiede una determinata operazione; può essere `Allow` o `Deny`. È anche possibile rifiutare esplicitamente l'accesso a una risorsa per essere certi che un utente non sia in grado di accedervi, anche quando tale accesso è assegnato da un'altra policy.

  > **Note**
  >
  > "Rifiuto implicito" ha lo stesso significato di "rifiuto per default".
  > Un "rifiuto implicito" è diverso da un "rifiuto esplicito". Per ulteriori informazioni, consulta la sezione sulla differenza tra rifiuto per default e rifiuto esplicito.

- Principale: account o utente autorizzato ad accedere alle operazioni e alle risorse nell'istruzione. In una policy delle risorse, il principale è l'utente o l'account che riceve questa autorizzazione.

L'esempio di policy delle risorse seguente mostra gli elementi di policy comuni descritti in precedenza. La policy concede l'accesso all'API in base a quanto specificato *`account-id`* nel campo specificato *`region`* a qualsiasi utente il cui indirizzo IP di origine si trova nel blocco di indirizzi. *`123.4.5.6/24`* La policy nega l'accesso all'API se l'IP di origine dell'utente non rientra nell'intervallo.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "arn:aws:execute-api:us-east-1:111111111111:*"
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "arn:aws:execute-api:us-east-1:111111111111:*",
            "Condition": {
                "NotIpAddress": {
                    "aws:SourceIp": "123.4.5.6/24"
                }
            }
        }
    ]
}
```

Come le policy delle risorse API Gateway influiscono sul flusso di lavoro delle autorizzazioni

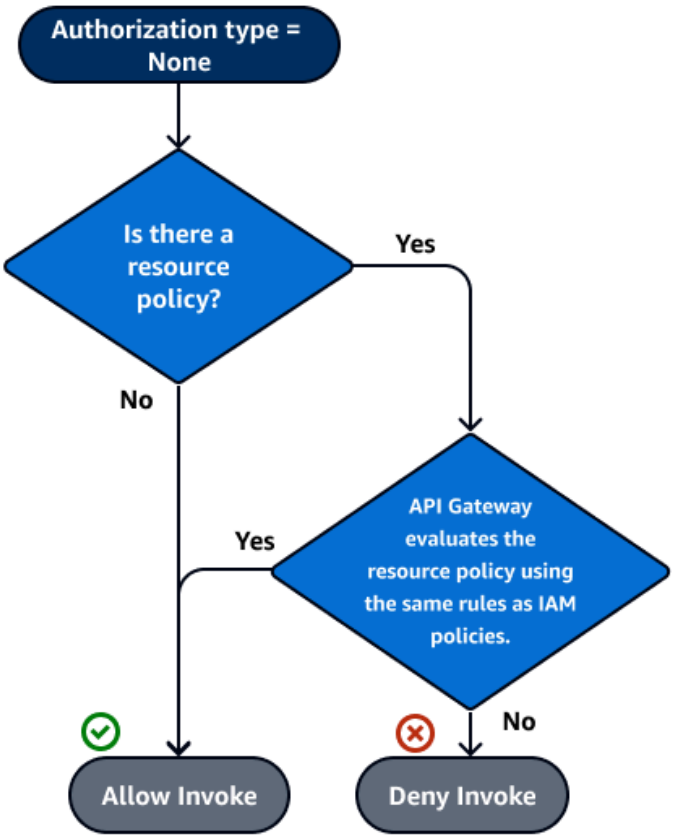
Quando API Gateway valuta la policy delle risorse collegata all'API, il risultato è influenzato dal tipo di autenticazione definito per l'API, come illustrato nei diagrammi nelle seguenti sezioni.

Argomenti

- Solo policy delle risorse API Gateway
- Autorizzazione Lambda e policy delle risorse
- Autenticazione IAM e policy delle risorse
- Autenticazione Amazon Cognito e policy delle risorse
- Tabelle dei risultati della valutazione delle policy

Solo policy delle risorse API Gateway

In questo flusso di lavoro, una policy delle risorse API Gateway è collegata all'API, ma non viene definito alcun tipo di autenticazione per l'API. La valutazione della policy comporta la ricerca di un'autorizzazione esplicita basata sui criteri in entrata dell'intermediario. Un rifiuto implicito o esplicito comporta il rifiuto dell'intermediario.



Di seguito viene riportato un esempio di tale policy delle risorse.

JSON

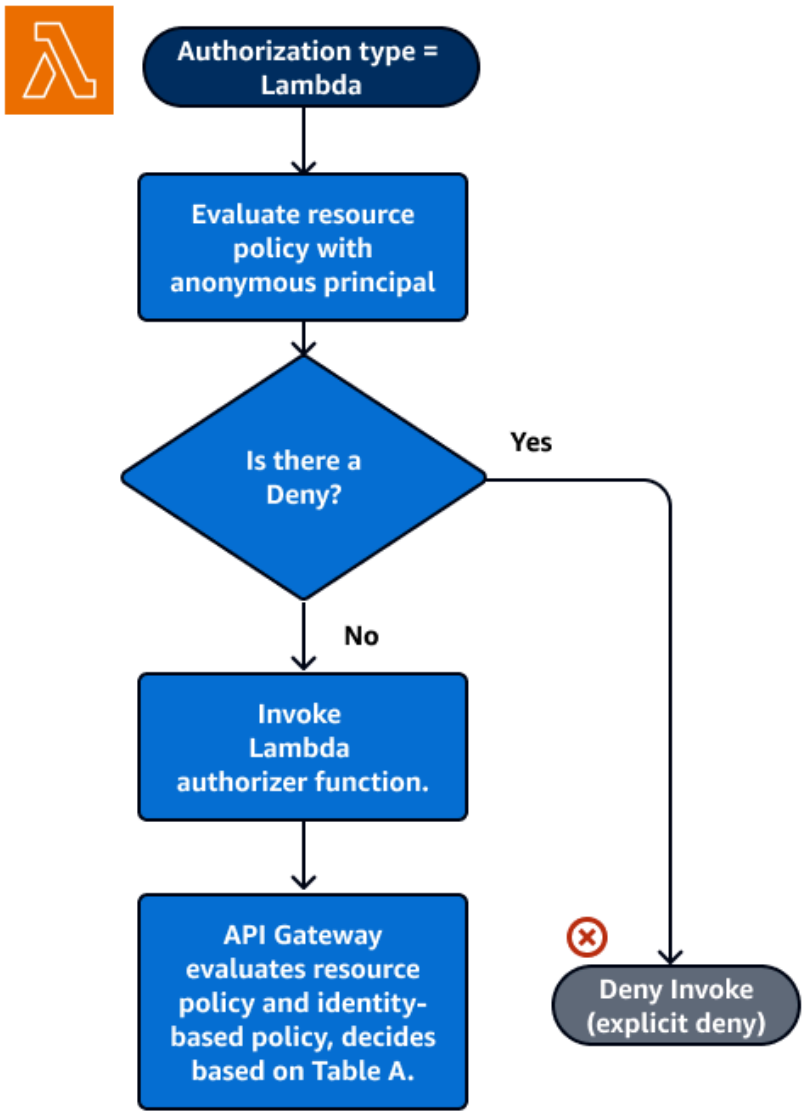
```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "arn:aws:execute-api:us-east-1:111111111111:api-id/",
            "Condition": {
```

```
                "IpAddress": {
                    "aws:SourceIp": ["192.0.2.0/24", "198.51.100.0/24" ]
                }
            }
        }
    ]
}
```

Autorizzazione Lambda e policy delle risorse

In questo flusso di lavoro, un'autorizzazione Lambda è configurata per l'API in aggiunta a una policy delle risorse. La policy delle risorse viene valutata in due fasi. Prima di chiamare l'autorizzazione Lambda, API Gateway valuta innanzitutto la policy e controlla la presenza di eventuali negazioni esplicite. Se riscontrati, all'intermediario viene negato subito l'accesso. In caso contrario viene chiamata l'autorizzazione Lambda, che restituisce un documento della policy che viene valutato insieme alla policy delle risorse. Se il sistema di autorizzazione utilizza il caching, Gateway API potrebbe restituire il documento di policy memorizzato nella cache. Il risultato è determinato in base alla tabella A.

La seguente policy delle risorse di esempio autorizza le chiamate solo dall'endpoint VPC il cui ID è *vpce-1a2b3c4d*. Nella fase di valutazione "pre-autorizzazione", solo le chiamate provenienti dall'endpoint VPC indicato nell'esempio sono autorizzate a proseguire e a valutare l'autorizzazione Lambda. Tutte le chiamate rimanenti vengono bloccate. Questo flusso di lavoro di autorizzazione è lo stesso se si utilizza un nome di dominio personalizzato per un'API privata.



JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "arn:aws:execute-api:us-east-1:111111111111:api-id/"
            ],
            "Condition" : {
```

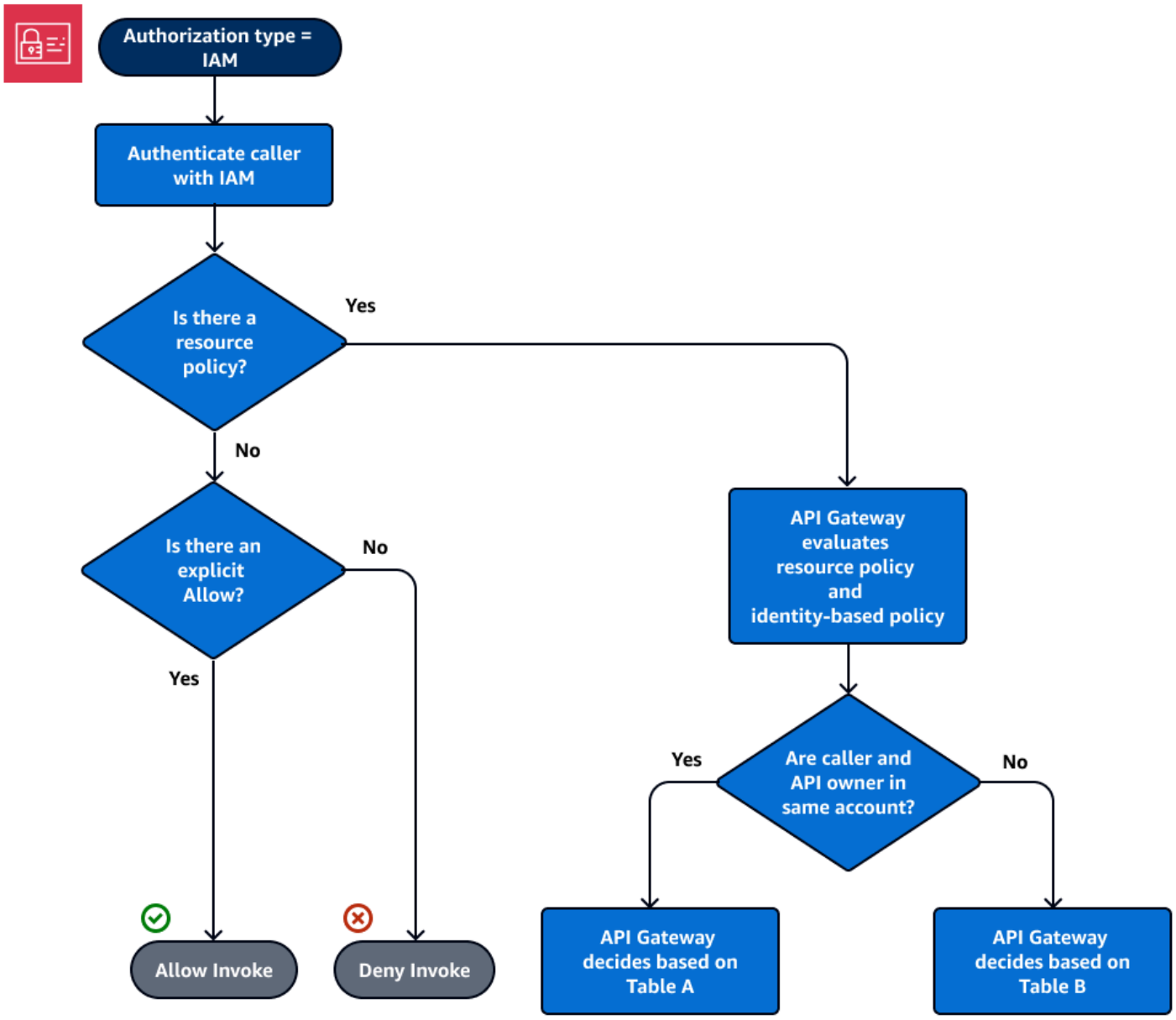
```
                "StringNotEquals": {
                    "aws:SourceVpce": "vpce-1a2b3c4d"
                }
            }
        }
    ]
}
```

Autenticazione IAM e policy delle risorse

In questo flusso di lavoro, viene configurata un'autenticazione IAM per l'API in aggiunta a una policy delle risorse. Dopo l'autenticazione dell'utente con il servizio IAM, l'API valuta sia le policy collegate all'utente sia la policy delle risorse. Il risultato varia a seconda che il chiamante si trovi nello stesso account Account AWS o in un altro Account AWS, rispetto al proprietario dell'API.

Se il chiamante e il proprietario dell'API sono in account diversi, il chiamante sarà autorizzato esplicitamente a procedere sia dalle policy dell'utente IAM sia dalla policy delle risorse. Per ulteriori informazioni, consulta la tabella B.

Tuttavia, se il chiamante e il proprietario dell'API si trovano nello stesso Account AWS, le policy dell'utente IAM o la policy delle risorse devono esplicitamente autorizzare il chiamante a proseguire. Per ulteriori informazioni, consulta la tabella A.



Di seguito viene riportato un esempio di policy delle risorse multiaccount. Ipotizzando che la policy IAM contenga un'autorizzazione di questo tipo, questa policy delle risorse permette chiamate solo dal VPC il cui ID VPC è *vpc-2f09a348*. Per ulteriori informazioni, consulta la tabella B.

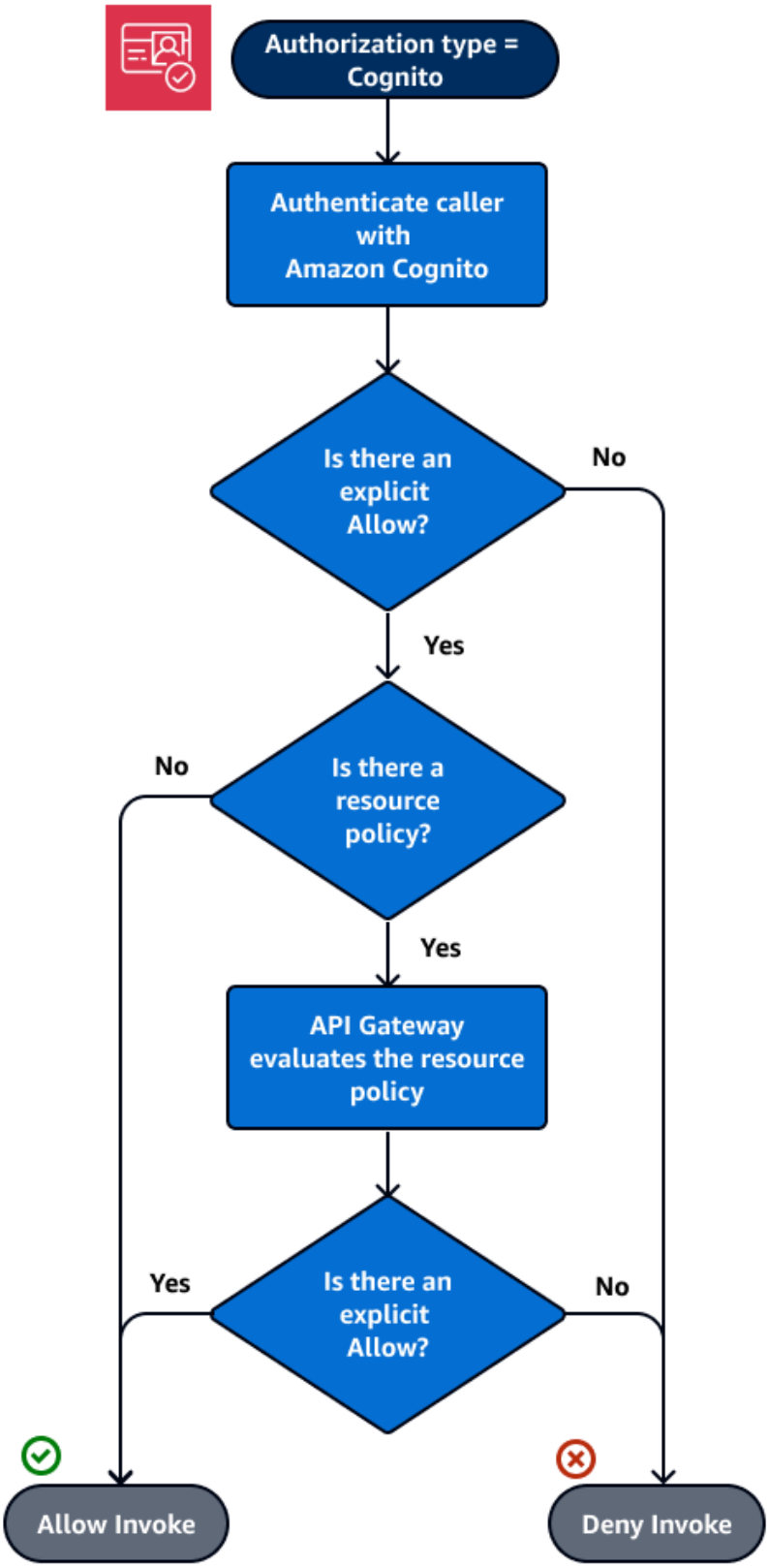
JSON

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

```
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "arn:aws:execute-api:us-east-1:111111111111:api-id/"
            ],
            "Condition" : {
                "StringEquals": {
                    "aws:SourceVpc": "vpc-2f09a348"
                    }
            }
        }
    ]
}
```

Autenticazione Amazon Cognito e policy delle risorse

In questo flusso di lavoro, un pool di utenti di Amazon Cognito è configurato per l'API in aggiunta a una policy delle risorse. API Gateway tenta innanzitutto di autenticare l'intermediario tramite Amazon Cognito. Questa operazione viene in genere eseguita tramite un token JWT fornito dall'intermediario. Se l'autenticazione va a buon fine, la policy delle risorse viene valutata in modo indipendente ed è richiesta un'autorizzazione esplicita. Un rifiuto o "non consentire né rifiutare" si traduce in un rifiuto. Di seguito è riportato un esempio di una policy delle risorse che potrebbe essere utilizzata insieme a pool di utenti di Amazon Cognito.


Authorization type = Cognito
Authenticate caller with Amazon Cognito
Is there an explicit Allow?
No
Yes
Is there a resource policy?
No
Yes
API Gateway evaluates the resource policy
Is there an explicit Allow?
Yes
No
Allow Invoke
Deny Invoke

Di seguito è riportato un esempio di una policy delle risorse che consente chiamate solo da determinati IP di origine, ipotizzando che il token di autenticazione di Amazon Cognito contenga un'autorizzazione. Per ulteriori informazioni, consulta la tabella B.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "arn:aws:execute-api:us-east-1:111111111111:api-id/",
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": ["192.0.2.0/24", "198.51.100.0/24" ]
                }
            }
        }
    ]
}
```

Tabelle dei risultati della valutazione delle policy

La tabella A riporta il comportamento risultante quando l'accesso a un'API di Gateway API è controllato da una policy IAM o da un sistema di autorizzazione Lambda e da una policy delle risorse Gateway API, entrambe nello stesso Account AWS.

Policy IAM (o sistema di autorizzazione Lambda)	Policy delle risorse Gateway API	Comportamento risultante
Abilita	Abilita	Abilita
Abilita	Né Abilita né Rifiuta	Abilita
Abilita	Rifiuta	Explicit Deny (rifiuto esplicito)
Né Abilita né Rifiuta	Abilita	Abilita

Policy IAM (o sistema di autorizzazione Lambda)	Policy delle risorse Gateway API	Comportamento risultante
Né Abilita né Rifiuta	Né Abilita né Rifiuta	Rifiuto implicito
Né Abilita né Rifiuta	Rifiuta	Explicit Deny (rifiuto esplicito)
Rifiuta	Abilita	Explicit Deny (rifiuto esplicito)
Rifiuta	Né Abilita né Rifiuta	Explicit Deny (rifiuto esplicito)
Rifiuta	Rifiuta	Explicit Deny (rifiuto esplicito)

La Tabella B elenca il comportamento risultante quando l'accesso a un'API API Gateway è controllato da una policy IAM o da un autorizzatore di pool di utenti di Amazon Cognito e da una policy di risorse API Gateway, che sono diversi. Account AWS Se una delle due è silente (non permette né rifiuta). l'accesso multiaccount viene negato. Questo perché l'accesso multi-account richiede l'autorizzazione esplicita all'accesso sia dalla policy delle risorse sia dalla policy IAM o dal sistema di autorizzazione del pool di utenti di Amazon Cognito.

Policy IAM o sistema di autorizzazione del pool di utenti di Amazon Cognito	Policy delle risorse Gateway API	Comportamento risultante
Abilita	Abilita	Abilita
Abilita	Né Abilita né Rifiuta	Rifiuto implicito
Abilita	Rifiuta	Explicit Deny (rifiuto esplicito)
Né Abilita né Rifiuta	Abilita	Rifiuto implicito
Né Abilita né Rifiuta	Né Abilita né Rifiuta	Rifiuto implicito
Né Abilita né Rifiuta	Rifiuta	Explicit Deny (rifiuto esplicito)
Rifiuta	Abilita	Explicit Deny (rifiuto esplicito)
Rifiuta	Né Abilita né Rifiuta	Explicit Deny (rifiuto esplicito)

Policy IAM o sistema di autorizzazione del pool di utenti di Amazon Cognito	Policy delle risorse Gateway API	Comportamento risultante
Rifiuta	Rifiuta	Explicit Deny (rifiuto esplicito)

Esempi di policy delle risorse API Gateway

Questa pagina include alcuni esempi di casi d'uso tipici per le policy delle risorse API Gateway.

Le policy di esempio seguenti utilizzano una sintassi semplificata per specificare la risorsa API. Questa sintassi semplificata è un modo abbreviato per fare riferimento a una risorsa API, invece di specificare l'Amazon Resource Name (ARN) completo. API Gateway converte la sintassi abbreviata nell'ARN completo quando si salva la policy. Ad esempio, è possibile specificare la risorsa `execute-api:/stage-name/GET/pets` in una policy delle risorse. API Gateway converte la risorsa in `arn:aws:execute-api:us-east-2:123456789012:aabbccddee/stage-name/GET/pets` quando si salva la policy delle risorse. API Gateway crea l'ARN completo utilizzando la regione corrente, l'ID dell' AWS account e l'ID dell'API REST a cui è associata la politica delle risorse. È possibile utilizzare `execute-api:/*` per rappresentare tutti gli stadi, i metodi e i percorsi nell'API corrente. Per informazioni sulla sintassi della/e policy di accesso, consulta Panoramica della sintassi delle policy di accesso per Amazon API Gateway.

Argomenti

- Esempio: consentire ai ruoli di un altro AWS account di utilizzare un'API
- Esempio: negare il traffico API in base all'indirizzo IP di origine o a un intervallo di indirizzi IP
- Esempio: rifiutare il traffico API in base all'indirizzo IP di origine o all'intervallo quando si utilizza un'API privata
- Esempio: consenti il traffico di API private in base all'endpoint VPC o al VPC sorgente

Esempio: consentire ai ruoli di un altro AWS account di utilizzare un'API

L'esempio seguente di policy in materia di risorse concede l'accesso dell'API in un AWS account a due ruoli in un AWS account diverso tramite i protocolli Signature Version 4 (SigV4) o Signature Version 4a (SigV4a). In particolare, allo sviluppatore e al ruolo di amministratore dell' AWS account identificato da *account-id-2* viene concessa l'azione per eseguire l'`execute-api:InvokeGET`azione sulla `pets` risorsa (API) dell'account. AWS

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:role/developer",
                    "arn:aws:iam::111122223333:role/Admin"
                ]
            },
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/stage/GET/pets"
            ]
        }
    ]
}
```

Esempio: negare il traffico API in base all'indirizzo IP di origine o a un intervallo di indirizzi IP

L'esempio di policy delle risorse seguente rifiuta (blocca) il traffico in ingresso verso un'API da due blocchi di indirizzi IP di origine specificati.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
```

```
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
               "execute-api:/*"
            ],
            "Condition" : {
                "IpAddress": {
                    "aws:SourceIp": ["192.0.2.0/24", "198.51.100.0/24" ]
                }
            }
        }
    ]
}
```

Se utilizzi policy utente IAM o policy di risorse API Gateway per controllare l'accesso ad API Gateway o a qualsiasi API Gateway APIs, conferma che le policy siano aggiornate per includere intervalli di IPv6 indirizzi. Le policy che non sono aggiornate per gestire IPv6 gli indirizzi potrebbero influire sull'accesso del cliente ad API Gateway quando inizia a utilizzare l'endpoint dualstack. Per ulteriori informazioni, consulta the section called “Utilizzo IPv6 degli indirizzi nelle politiche IAM”.

Esempio: rifiutare il traffico API in base all'indirizzo IP di origine o all'intervallo quando si utilizza un'API privata

L'esempio di policy delle risorse seguente rifiuta (blocca) il traffico in ingresso verso un'API da due blocchi di indirizzi IP di origine specificati. Quando si utilizza private APIs, l'endpoint VPC `execute-api` riscrive l'indirizzo IP di origine originale. La condizione `aws:VpcSourceIp` filtra la richiesta rispetto all'indirizzo IP del richiedente originale.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
```

```
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
               "execute-api:/*"
            ],
            "Condition" : {
                "IpAddress": {
                    "aws:VpcSourceIp": ["192.0.2.0/24", "198.51.100.0/24"]
                }
            }
        }
    ]
}
```

Esempio: consenti il traffico di API private in base all'endpoint VPC o al VPC sorgente

Le seguenti policy delle risorse di esempio consentono il traffico in entrata per un'API privata solo da un VPC o da un endpoint VPC specifici.

Questa policy della risorsa di esempio specifica il VPC di origine:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
```

```
                "execute-api:/*"
            ],
            "Condition" : {
                "StringNotEquals": {
                   "aws:SourceVpc": "vpc-1a2b3c4d"
                }
            }
        }
    ]
}
```

Questa policy della risorsa di esempio specifica l'endpoint VPC di origine:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ],
            "Condition" : {
                "StringNotEquals": {
                    "aws:SourceVpce": "vpce-1a2b3c4d"
                }
            }
        }
    ]
}
```

Creazione e collegamento di una policy delle risorse API Gateway a un'API

Per consentire a un utente di accedere all'API chiamando il servizio di esecuzione dell'API, è necessario creare una policy delle risorse Gateway API e collegare la policy all'API. Il collegamento di una policy all'API comporta l'applicazione delle autorizzazioni della policy ai metodi dell'API. Se si aggiorna la policy delle risorse, è necessario implementare l'API.

Argomenti

- Prerequisiti
- Collegamento di una policy delle risorse a un'API di Gateway API
- Risoluzione dei problemi relativi alla policy delle risorse

Prerequisiti

Per aggiornare una policy delle risorse Gateway API, è necessario disporre delle autorizzazioni `apigateway:UpdateRestApiPolicy` e `apigateway:PATCH`.

Per un'API ottimizzata per l'edge o regionale, è possibile collegare la policy delle risorse all'API al momento della creazione o dopo che è stata implementata. Un'API privata non può essere implementata senza una policy delle risorse. Per ulteriori informazioni, consulta the section called “REST API private”.

Collegamento di una policy delle risorse a un'API di Gateway API

La procedura seguente mostra come collegare una policy delle risorse a un'API di Gateway API.

Console di gestione AWS

Per collegare una policy delle risorse a un'API di API Gateway

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere una REST API.
3. Nel riquadro di navigazione principale, scegli Policy delle risorse.
4. Scegli Crea policy.
5. (Facoltativo) Scegli Seleziona un modello per generare una policy di esempio.

   Negli esempi di policy, i segnaposto sono racchiusi tra parentesi graffe doppie (`"{{placeholder}}"`). Sostituisci ogni segnaposto, incluse le parentesi graffe, con le informazioni necessarie.

6. Se non utilizzi uno dei modelli di esmepio, immetti la tua policy delle risorse.
7. Scegli Save changes (Salva modifiche).

Se l'API è stata distribuita in precedenza nella console API Gateway, sarà necessario ridistribuirla affinché la policy delle risorse diventi effettiva.

AWS CLI

Per utilizzare il AWS CLI per creare una nuova API e allegare ad essa una politica delle risorse, usa il seguente comando: create-rest-api

```
aws apigateway create-rest-api \
    --name "api-name" \
    --policy "{\"jsonEscapedPolicyDocument\"}"
```

Per utilizzare AWS CLI per allegare una politica delle risorse a un'API esistente, utilizzare il update-rest-apicomando seguente:

```
aws apigateway update-rest-api \
    --rest-api-id api-id \
    --patch-operations op=replace,path=/
policy,value='"{\"jsonEscapedPolicyDocument\"}"'
```

È inoltre possibile allegare la politica delle risorse come `policy.json` file separato e includerla nel create-rest-apicomando. Il create-rest-apicomando seguente crea una nuova API con una politica delle risorse:

```
aws apigateway create-rest-api \
    --name "api-name" \
    --policy file://policy.json
```

Il file `policy.json` è una policy delle risorse Gateway API, come descritto in the section called “Esempio: negare il traffico API in base all'indirizzo IP di origine o a un intervallo di indirizzi IP”.

AWS CloudFormation

È possibile utilizzare CloudFormation per creare un'API con una politica delle risorse. L'esempio seguente crea una REST API con la policy delle risorse di esempio, come descritto in the section called “Esempio: negare il traffico API in base all'indirizzo IP di origine o a un intervallo di indirizzi IP”.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Name: testapi
      Policy:
        Statement:
          - Action: 'execute-api:Invoke'
            Effect: Allow
            Principal: '*'
            Resource: 'execute-api:/*'
          - Action: 'execute-api:Invoke'
            Effect: Deny
            Principal: '*'
            Resource: 'execute-api:/*'
            Condition:
              IpAddress:
                'aws:SourceIp': ["192.0.2.0/24", "198.51.100.0/24" ]
        Version: 2012-10-17
  Resource:
    Type: 'AWS::ApiGateway::Resource'
    Properties:
      RestApiId: !Ref Api
      ParentId: !GetAtt Api.RootResourceId
      PathPart: 'helloworld'
  MethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref Resource
      HttpMethod: GET
      ApiKeyRequired: false
      AuthorizationType: NONE
      Integration:
        Type: MOCK
        RequestTemplates:
          application/json: '{"statusCode": 200}'
        IntegrationResponses:
          - StatusCode: 200
            ResponseTemplates:
              application/json: '{}'
      MethodResponses:
```

```
      - StatusCode: 200
        ResponseModels:
          application/json: 'Empty'
ApiDeployment:
  Type: 'AWS::ApiGateway::Deployment'
  DependsOn:
    - MethodGet
  Properties:
    RestApiId: !Ref Api
    StageName: test
```

Risoluzione dei problemi relativi alla policy delle risorse

La seguente guida può aiutare a risolvere i problemi relativi alla policy delle risorse.

L'API restituisce {"Message":"User: anonymous is not authorized to perform: execute-api:Invoke on resource: arn:aws:execute-api:us-east-1:********/****/****/"}

Nella politica delle risorse, se imposti il AWS Principal su un principale, ad esempio:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111111111111:role/developer",
                    "arn:aws:iam::111111111111:role/Admin"
                ]
            },
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/stage/GET/pets"
            ]
        }
    ]
}
```

Devi utilizzare l'autorizzazione `AWS_IAM` per ogni metodo dell'API, altrimenti l'API restituirà il messaggio di errore precedente. Per ulteriori indicazioni su come attivare l'autorizzazione `AWS_IAM` per un metodo, consulta the section called “Metodi”.

La policy delle risorse non viene aggiornata

Se aggiorni la policy delle risorse dopo la creazione dell'API, dovrai distribuire l'API per propagare le modifiche dopo il collegamento della policy aggiornata. L'aggiornamento o il salvataggio della policy non modificherà il comportamento di runtime dell'API. Per ulteriori informazioni sulla distribuzione della tua API, vedi the section called “Implementazione di REST API”.

La policy delle risorse restituisce il seguente errore: Invalid policy document. Please check the policy syntax and ensure that Principals are valid.

Per correggere questo errore, è consigliabile innanzitutto controllare la sintassi della policy. Per ulteriori informazioni, consulta the section called “Panoramica della sintassi delle policy di accesso per Amazon API Gateway”. Si suggerisce anche di verificare che tutti i principali specificati siano validi e non siano stati eliminati.

Inoltre, se l'API si trova in una Regione con consenso esplicito, verificare che la Regione sia abilitata per tutti gli account della policy delle risorse.

AWS chiavi di condizione che possono essere utilizzate nelle politiche delle risorse di API Gateway

La tabella seguente contiene le chiavi di AWS condizione che possono essere utilizzate nelle politiche delle risorse per APIs API Gateway per ogni tipo di autorizzazione.

Per ulteriori informazioni sulle chiavi AWS condizionali, consulta AWS Global Condition Context Keys.

Chiavi di condizione	Criteri	Necessita di **`AuthN`**?	Tipo di autorizzazione
`aws:CurrentTime`	Nessuna	No	Tutti
`aws:EpochTime`	Nessuna	No	Tutti
`aws:TokenIssueTime`	La chiave è presente solo nelle richieste firmate utilizzando credenziali di sicurezza temporanee.	Sì	IAM

Chiavi di condizione	Criteri	Necessita di **AuthN**?	Tipo di autorizzazione
`aws:MultiFactorAuthPresent`	La chiave è presente solo nelle richieste firmate utilizzando credenziali di sicurezza temporanee.	Sì	IAM
`aws:MultiFactorAuthAge`	La chiave è presente solo se MFA è presente nelle richieste.	Sì	IAM
`aws:PrincipalAccount`	Nessuna	Sì	IAM
`aws:PrincipalArn`	Nessuna	Sì	IAM
`aws:PrincipalOrgID`	Questa chiave viene inclusa nel contesto della richiesta solo se l'entità è membro di un'organizzazione.	Sì	IAM
`aws:PrincipalOrgPaths`	Questa chiave viene inclusa nel contesto della richiesta solo se l'entità è membro di un'organizzazione.	Sì	IAM

Chiavi di condizione	Criteri	Necessita di **AuthN**?	Tipo di autorizzazione
`aws:PrincipalTag`	Questa chiave è inclusa nel contesto della richiesta se l'entità utilizza un utente IAM con tag collegati. È inclusa per un'entità che utilizza un ruolo IAM con tag collegati o tag di sessione.	Sì	IAM
`aws:PrincipalType`	Nessuna	Sì	IAM
`aws:Referer`	La chiave è presente solo se il valore è fornito dall'intermediario nell'interstazione HTTP.	No	Tutti
`aws:SecureTransport`	Nessuna	No	Tutti
`aws:SourceArn`	Nessuna	No	Tutti
`aws:SourceIp`	Nessuna	No	Tutti
`aws:SourceVpc`	Questa chiave può essere utilizzata solo per uso privato APIs.	No	Tutti
`aws:SourceVpce`	Questa chiave può essere usata solo per uso privato APIs.	No	Tutti

Chiavi di condizione	Criteri	Necessita di **AuthN**?	Tipo di autorizzazione
`aws:VpcSourceIp`	Questa chiave può essere usata solo per uso privato APIs.	No	Tutti
`aws:UserAgent`	La chiave è presente solo se il valore è fornito dall'inte rmediario nell'inte stazione HTTP.	No	Tutti
`aws:userid`	Nessuna	Sì	IAM
`aws:username`	Nessuna	Sì	IAM

Controllo degli accessi a una REST API con le autorizzazioni IAM

Puoi controllare l'accesso all'API di Amazon API Gateway con le autorizzazioni IAM controllando l'accesso ai seguenti due processi componenti di API Gateway:

- Per creare, distribuire e gestire un'API in API Gateway, è necessario concedere allo sviluppatore dell'API le autorizzazioni per eseguire le operazioni richieste supportate dal componente di gestione dell'API di API Gateway.
- Per chiamare un'API distribuita o per aggiornare la memorizzazione nella cache dell'API, è necessario concedere all'intermediario dell'API le autorizzazioni per eseguire le operazioni IAM supportate dal componente di esecuzione dell'API di API Gateway.

Il controllo degli accessi per i due processi prevede l'uso dei due diversi modelli di autorizzazione illustrati più avanti.

Modello di autorizzazione API Gateway per la creazione e la gestione di un'API

Per consentire a uno sviluppatore di API di creare e gestire un'API in API Gateway, è necessario creare le policy di autorizzazione IAM che consentono a uno sviluppatore di API specifico di creare, aggiornare, distribuire, visualizzare o eliminare le entità API richieste. È possibile collegare la policy delle autorizzazioni a un utente, un ruolo o un gruppo.

Per fornire l'accesso, aggiungi autorizzazioni agli utenti, gruppi o ruoli:

- Utenti e gruppi in Centro identità AWS IAM:

  Crea un set di autorizzazioni. Segui le istruzioni riportate nella pagina Create a permission set (Creazione di un set di autorizzazioni) nella Guida per l'utente di Centro identità AWS IAM .

- Utenti gestiti in IAM tramite un provider di identità:

  Crea un ruolo per la federazione delle identità. Segui le istruzioni riportate nella pagina Create a role for a third-party identity provider (federation) della Guida per l'utente IAM.

- Utenti IAM:
  - Crea un ruolo che l'utente possa assumere. Segui le istruzioni riportate nella pagina Create a role for an IAM user della Guida per l'utente IAM.
  - (Non consigliato) Collega una policy direttamente a un utente o aggiungi un utente a un gruppo di utenti. Segui le istruzioni riportate nella pagina Aggiunta di autorizzazioni a un utente (console) nella Guida per l'utente IAM.

Per ulteriori informazioni su come usare il modello di autorizzazione, consulta the section called "Policy basate su identità dei API Gateway".

Modello di autorizzazione API Gateway per invocare un'API

Per consentire a un chiamante dell'API di richiamare l'API o aggiornare la relativa memorizzazione nella cache, è necessario creare policy IAM che permettano al chiamante dell'API specificato di richiamare il metodo API per il quale è abilitata l'autenticazione dell'utente. Lo sviluppatore dell'API imposta la proprietà `authorizationType` del metodo su `AWS_IAM` per richiedere al chiamante di inviare le chiavi di accesso dell'utente per l'autenticazione. Gateway API supporta Signature Version 4a (SigV4a) e Signature Version 4 (SigV4) per autenticare le credenziali dell'utente. Per ulteriori informazioni, consultare AWS Signature Version 4. Sarà quindi possibile collegare la policy all'utente, al gruppo o al ruolo.

In questa istruzione di policy di autorizzazione IAM l'elemento IAM `Resource` contiene un elenco di metodi API distribuiti identificati da verbi HTTP e percorsi di risorsa API Gateway specifici. L'elemento IAM `Action` contiene le operazioni di esecuzione API di API Gateway richieste. Tali operazioni includono `execute-api:Invoke` o `execute-api:InvalidateCache`, dove `execute-api` designa il componente di esecuzione dell'API sottostante di API Gateway.

Per ulteriori informazioni su come usare il modello di autorizzazione, consulta Controllo degli accessi per invocare un'API.

Quando un'API è integrata con un AWS servizio (ad esempio AWS Lambda) nel back-end, API Gateway deve inoltre disporre delle autorizzazioni per accedere alle AWS risorse integrate (ad esempio, richiamando una funzione Lambda) per conto del chiamante dell'API. Per concedere queste autorizzazioni, crea un ruolo IAM del tipo AWS service for API Gateway (Servizio AWS per API Gateway). Quando crei questo ruolo nella console di gestione IAM, tale ruolo contiene la policy di attendibilità IAM seguente, in base alla quale API Gateway viene dichiarato un'entità attendibile che può assumere quel ruolo:

JSON

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

Se crei il ruolo IAM chiamando il comando create-role della CLI o un metodo SDK corrispondente, devi fornire la policy di attendibilità precedente come parametro di input di `assume-role-policy-document`. Non tentate di creare una policy di questo tipo direttamente nella console di gestione IAM o di chiamare il comando AWS CLI create-policy o un metodo SDK corrispondente.

Affinché API Gateway chiami il AWS servizio integrato, devi anche allegare a questo ruolo le politiche di autorizzazione IAM appropriate per chiamare i servizi integrati. AWS Ad esempio, per chiamare una funzione Lambda, devi includere la seguente policy di autorizzazione IAM nel ruolo IAM:

JSON

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

Lambda supporta le policy di accesso basate su risorse, che combinano policy di autorizzazione e di attendibilità. Quando integri un'API con una funzione Lambda utilizzando la console API Gateway, non ti viene richiesto di impostare questo ruolo IAM esplicitamente, poiché la console imposta automaticamente con il tuo consenso le autorizzazioni basate su risorse sulla funzione Lambda.

> **Note**
>
> Per attuare il controllo degli accessi a un AWS servizio, puoi utilizzare il modello di autorizzazioni basato sul chiamante, in cui una policy di autorizzazioni è collegata direttamente all'utente o al gruppo del chiamante, oppure il modello di autorizzazione basato sui ruoli, in cui una policy di autorizzazioni è associata a un ruolo IAM che API Gateway può assumere. Le policy di autorizzazione possono differire nei due modelli. Ad esempio la policy basata su intermediario blocca l'accesso, mentre quella basata su ruolo lo consente. Puoi trarne vantaggio per richiedere che un utente acceda a un servizio solo tramite un'API API Gateway. AWS

Controllo degli accessi per invocare un'API

In questa sezione viene illustrato il modello di autorizzazioni per controllare l'accesso all'API utilizzando le autorizzazioni IAM. Quando l'autorizzazione IAM è abilitata, i client devono utilizzare Signature Version 4a (Sigv4a) e Signature Version 4 (Sigv4) per firmare le proprie richieste con credenziali. AWS Per ulteriori informazioni, consultare AWS Signature Version 4.

Questa sezione illustra un modello di istruzione di policy IAM e il riferimento all'istruzione di policy. Il riferimento all'istruzione di policy include i formati dei campi `Action` e `Resource` correlati al servizio di esecuzione dell'API. Si utilizzano questi riferimenti per creare l'istruzione di policy IAM. Quando si crea l'istruzione di policy IAM, è opportuno a volte considerare il modo in cui le policy delle risorse Gateway API influiscono sul flusso di lavoro di autorizzazione. Per ulteriori informazioni, consulta the section called "Come le policy delle risorse influiscono sul flusso di lavoro delle autorizzazioni".

Per le API private, dovresti utilizzare una combinazione di una policy di risorse API Gateway e una policy di endpoint VPC. Per ulteriori informazioni, consultare i seguenti argomenti:

- the section called "Uso delle policy delle risorse API Gateway"
- the section called "Utilizzo di policy di endpoint VPC per API private"

Controllo di chi può chiamare un metodo API di API Gateway con le policy IAM

Per controllare chi può o non può chiamare un'API distribuita con le autorizzazioni IAM, crea un documento di policy IAM con le autorizzazioni richieste. Di seguito viene mostrato un modello per questo tipo di documento di policy.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Permission",
      "Action": [
        "execute-api:Execution-operation"
      ],
      "Resource": [
        "arn:aws:execute-api:region:123456789012:api-
id/stage/METHOD_HTTP_VERB/Resource-path"
      ]
    }
  ]
}
```

Nell'esempio riportato *`Permission`* deve essere sostituito da `Allow` o da `Deny` a seconda che si desideri concedere o revocare le autorizzazioni incluse. *`Execution-operation`* deve essere

sostituito dalle operazioni supportate dal servizio di esecuzione dell'API. *METHOD_HTTP_VERB* indica un verbo HTTP supportato dalle risorse specificate. *Resource-path* è il segnaposto per il percorso URL dell'istanza `Resource` di un'API distribuita che supporta l'elemento *METHOD_HTTP_VERB* citato. Per ulteriori informazioni, consulta Riferimento delle istruzioni delle policy IAM per l'esecuzione dell'API in API Gateway.

> **Note**
>
> Affinché le policy IAM diventino effettive, devi avere abilitato l'autenticazione IAM sui metodi API impostando `AWS_IAM` per la proprietà `authorizationType` del metodo. In caso contrario, questi metodi API saranno accessibili pubblicamente.

Ad esempio, per concedere a un utente l'autorizzazione per visualizzare un elenco di animali domestici esposti tramite un'API specificata, rifiutandogli tuttavia l'autorizzazione per aggiungere un animale domestico all'elenco, è possibile includere nella policy IAM l'istruzione seguente:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:111111111111:api-id/*/GET/pets"
      ]
    },
    {
      "Effect": "Deny",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:111111111111:api-id/*/POST/pets"
      ]
    }
```

```
  ]
}
```

Per concedere a un utente l'autorizzazione per visualizzare un animale domestico specifico esposto da un'API configurata come `GET /pets/{petId}`, è possibile includere nella policy IAM l'istruzione seguente:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "execute-api:Invoke"
            ],
            "Resource": [
                "arn:aws:execute-api:us-east-1:111122223333:api-id/*/
GET/pets/a1b2"
            ]
        }
    ]
}
```

Riferimento delle istruzioni delle policy IAM per l'esecuzione dell'API in API Gateway

Le informazioni seguenti descrivono il formato Action e Resource delle istruzioni di policy IAM per le autorizzazioni di accesso per l'esecuzione di un'API.

Formato dell'elemento Action delle autorizzazioni per l'esecuzione dell'API in API Gateway

L' API-executing `Action`espressione ha il seguente formato generale:

```
execute-api:action
```

dove *`action`* è disponibile un' API-executingazione:

- *, che rappresenta tutte le operazioni che seguono.

- Invoke, utilizzato per invocare un'API su richiesta del client.
- InvalidateCache, utilizzato per invalidare la cache delle API su richiesta del client.

Formato dell'elemento Resource delle autorizzazioni per l'esecuzione dell'API in API Gateway

L' API-executing `Resource`espressione ha il seguente formato generale:

```
arn:aws:execute-api:region:account-id:api-id/stage-name/HTTP-VERB/resource-path-specifier
```

dove:

- *`region`*è l' AWS area (ad esempio **`us-east-1`** o **`*`** per tutte le AWS regioni) che corrisponde all'API distribuita per il metodo.
- *`account-id`*è l'ID di AWS account a 12 cifre del proprietario dell'API REST.
- *`api-id`*è l'identificatore che API Gateway ha assegnato all'API per il metodo.
- *`stage-name`*è il nome della fase associata al metodo.
- *`HTTP-VERB`*è il verbo HTTP per il metodo. Può essere uno dei seguenti: GET, POST, PUT, DELETE, PATCH.
- *`resource-path-specifier`*è il percorso del metodo desiderato.

> **Note**
>
> Se specifichi un carattere jolly (`*`), l'espressione `Resource` applica il carattere jolly al resto dell'espressione.

Alcuni esempi dell'espressione Resource includono:

- **`arn:aws:execute-api:*:*:*`**per qualsiasi percorso di risorse in qualsiasi fase, per qualsiasi API in qualsiasi AWS regione.
- **`arn:aws:execute-api:us-east-1:*:*`**per qualsiasi percorso di risorse in qualsiasi fase, per qualsiasi API nella AWS regione di`us-east-1`.
- **`arn:aws:execute-api:us-east-1:*:`*`api-id`*`/*`**per qualsiasi percorso di risorsa in qualsiasi fase, per l'API con l'identificatore di *`api-id`* nella AWS regione us-east-1.

- **`arn:aws:execute-api:us-east-1:*:`*`api-id`*`/test/*`**per qualsiasi percorso di risorsa nella fase di`test`, per l'API con l'identificatore di *`api-id`* nella regione us-east-1. AWS

Per ulteriori informazioni, consulta Documentazione di riferimento Amazon Resource Name (ARN) API Gateway.

Esempi di policy IAM per le autorizzazioni di esecuzione API

Per informazioni sui modelli di autorizzazioni e per altre informazioni di base, consulta Controllo degli accessi per invocare un'API.

L'istruzione di policy seguente concede all'utente l'autorizzazione di chiamare qualsiasi metodo POST nel percorso `mydemoresource`, nella fase `test`, per l'API con l'identificatore `a123456789`, presupponendo che l'API corrispondente sia stata distribuita nella regione AWS us-east-1:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:*:a123456789/test/POST/my-demo-resource-
path/*"
      ]
    }
  ]
}
```

L'istruzione di policy di esempio seguente concede all'utente l'autorizzazione di chiamare qualsiasi metodo nel percorso di risorsa `petstorewalkthrough/pets`, in qualsiasi fase, per l'API con l'identificativo `a123456789`, in qualsiasi regione AWS in cui è stata distribuita l'API corrispondente:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:*:*:a123456789/*/*/petstorewalkthrough/pets"
      ]
    }
  ]
}
```

Utilizzo di policy di endpoint VPC per API private in API Gateway

È possibile creare una policy di endpoint VPC per migliorare la sicurezza dell'API privata. Una policy di endpoint VPC è una policy delle risorse IAM che è possibile allegare all'endpoint VPC. Per ulteriori informazioni, consulta Controllo dell'accesso ai servizi con endpoint VPC.

È possibile creare una policy di endpoint VPC per eseguire le seguenti attività.

- Consentire solo a determinate organizzazioni o risorse di accedere all'endpoint VPC e invocare l'API.
- Utilizzare un'unica policy ed evitare le policy basate sulla sessione o sul ruolo per controllare il traffico verso l'API.
- Rafforza il perimetro di sicurezza della tua applicazione durante la migrazione da on-premise a. AWS

Considerazioni sulla policy degli endpoint VPC

Di seguito sono riportate alcune considerazioni sulla policy di endpoint VPC:

- L'identità dell'invoker viene valutata in base al valore dell'intestazione `Authorization`. La policy di endpoint VPC viene valutata per prima, quindi Gateway API valuta la richiesta, in

base al tipo di autorizzazione configurato nella richiesta di metodo. La tabella seguente mostra come viene valutata la policy di endpoint VPC in base al contenuto del valore dell'intestazione `Authorization`.

Contenuto del valore dell'intestazione **`Authorization`**	Come viene valutata la policy di endpoint VPC
Nessun contenuto	L'invoker viene valutato come utente anonimo
Firma SigV4 o SigV4a valida	L'invoker viene valutato come identità IAM autenticata dalla firma
Firma SigV4 o SigV4a non valida	Gateway API nega l'accesso
Non-SigV4 informazioni di autorizzazione, ad esempio un token al portatore	L'invoker viene valutato come utente anonimo

- Se il controllo degli accessi dipende dall'utilizzo di un token di connessione, come un sistema di autorizzazione Lambda o Amazon Cognito, è possibile controllare il perimetro di sicurezza utilizzando le proprietà della risorsa.
- Se i controlli di autorizzazione utilizzano l'autorizzazione IAM, è possibile controllare il perimetro di sicurezza utilizzando le proprietà della risorsa e le proprietà del principale.
- Le policy di endpoint VPC possono essere utilizzate insieme alle policy di risorse API Gateway. La policy delle risorse Gateway API specifica quali principali possono accedere all'API. La policy di endpoint specifica chi può accedere al VPC e quali API possono essere chiamate dall'endpoint VPC. L'API privata richiede una policy delle risorse, ma non è necessario creare una policy di endpoint VPC personalizzata.

Esempi di policy di endpoint VPC

È possibile creare le policy degli endpoint Amazon Virtual Private Cloud per Gateway Amazon API in cui specificare quanto segue.

- Il principale che può eseguire azioni.
- Le operazioni che possono essere eseguite.
- Le risorse su cui è possibile eseguire le operazioni.

Ciò potrebbe dipendere dal contenuto dell'intestazione dell'autorizzazione. Per ulteriori informazioni, consulta the section called "Considerazioni sulla policy degli endpoint VPC". Per ulteriori politiche di esempio, consulta Esempi di politiche perimetrali dei dati sul sito Web. GitHub

Per collegare la policy all'endpoint VPC, è necessario utilizzare la console VPC. Per ulteriori informazioni, consulta Controllo dell'accesso ai servizi con endpoint VPC.

Esempio 1: policy di endpoint VPC che concede l'accesso a due API

L'esempio di policy seguente concede l'accesso solo a due API specifiche tramite l'endpoint VPC a cui è collegata la policy.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Principal": "*",
            "Action": [
                "execute-api:Invoke"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:execute-api:us-east-1:123412341234:a1b2c3d4e5/*",
                "arn:aws:execute-api:us-east-1:123412341234:aaaaa11111/*"
            ]
        }
    ]
}
```

Esempio 2: policy di endpoint VPC che concede l'accesso a metodi GET

L'esempio di policy seguente concede agli utenti l'accesso ai metodi `GET` per un'API specifica tramite l'endpoint VPC a cui è collegata l'API.

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Principal": "*",
            "Action": [
                "execute-api:Invoke"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:execute-api:us-east-1:123412341234:a1b2c3d4e5/stageName/
GET/*"
            ]
        }
    ]
}
```

Esempio 3: policy di endpoint VPC che concede a un utente specifico l'accesso a un'API specifica

L'esempio di policy seguente concede a un utente specifico l'accesso a un'API specifica tramite l'endpoint VPC a cui è collegata la policy.

In questo caso, poiché la policy limita l'accesso a specifici principali IAM, è necessario impostare il valore di `authorizationType` del metodo su `AWS_IAM` o `NONE`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123412341234:user/MyUser"
                ]
            },
            "Action": [
                "execute-api:Invoke"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:execute-api:us-east-1:123412341234:a1b2c3d4e5/*"
            ]
```

```
        }
    ]
}
```

Esempio 4: policy di endpoint VPC che fornisce agli utenti l'accesso a uno specifico nome di dominio personalizzato e a tutte le API mappate al dominio

La seguente policy di esempio fornisce agli utenti l'accesso a uno specifico nome di dominio personalizzato per API private tramite l'endpoint VPC a cui è collegata la policy. Con questa policy un utente può invocare qualsiasi API mappata al nome di dominio personalizzato, purché abbia creato un'associazione di accesso al nome di dominio tra l'endpoint VPC e il nome di dominio personalizzato e disponga dell'accesso per invocare il nome di dominio personalizzato e qualsiasi API privata mappata al nome di dominio personalizzato. Per ulteriori informazioni, consulta the section called “Nomi di dominio personalizzati per API private”.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "execute-api:Invoke",
      "Resource": [
        "*"
      ],
       "Condition": {
        "ArnEquals": {
          "execute-api:viaDomainArn": "arn:aws:execute-api:us-
west-2:111122223333:/domainnames/private.test.com+f4g5h6"
        }
      }
    }
  ]
}
```

Esempio 5: policy di endpoint VPC che fornisce o nega l'accesso ad API e risorse di dominio specifiche

La seguente policy di esempio fornisce agli utenti l'accesso ad API e risorse di dominio specifiche. Con questa policy un utente può invocare le API private e le risorse di dominio consentite, purché abbia creato un'associazione di accesso al nome di dominio tra l'endpoint VPC e il nome di dominio personalizzato e disponga dell'accesso per invocare il nome di dominio personalizzato e qualsiasi API privata mappata al nome di dominio personalizzato. Per ulteriori informazioni, consulta the section called “Nomi di dominio personalizzati per API private”.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "execute-api:Invoke",
      "Resource": [
        "arn:aws:execute-api:us-west-2:111122223333:/domainnames/private.test.com
+f4g5h6",
        "arn:aws:execute-api:us-west-2:111122223333:a1b2c3d4e5/*"
      ]
    },
    {
      "Effect": "Deny",
      "Principal": {
        "AWS": "*"
      },
      "Action": "execute-api:Invoke",
      "Resource": [
        "arn:aws:execute-api:us-west-2:111122223333:a1b2c3d4e5/admin/*",
        "arn:aws:execute-api:us-west-2:111122223333:bcd123455/*"
      ]
    }
  ]
}
```

Esempio 6: policy di endpoint VPC che fornisce o nega l'accesso a principali e risorse di un'organizzazione

La seguente policy di esempio fornisce l'accesso ai principali e alle risorse di un'organizzazione.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Condition": {
                "StringEquals": {
                    "aws:ResourceOrgID": "o-abcd1234",
                    "aws:PrincipalOrgID": "o-abcd1234"
                }
            },
            "Action": "*",
            "Resource": "*",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Sid": "AllowRequestsByOrgsIdentitiesToOrgsResources"
        }
    ]
}
```

Usa i tag per controllare l'accesso a REST APIs in API Gateway

L'autorizzazione ad accedere a REST APIs può essere ottimizzata utilizzando il controllo degli accessi basato sugli attributi nelle policy IAM.

Per ulteriori informazioni, consulta the section called "Attribute-based controllo degli accessi".

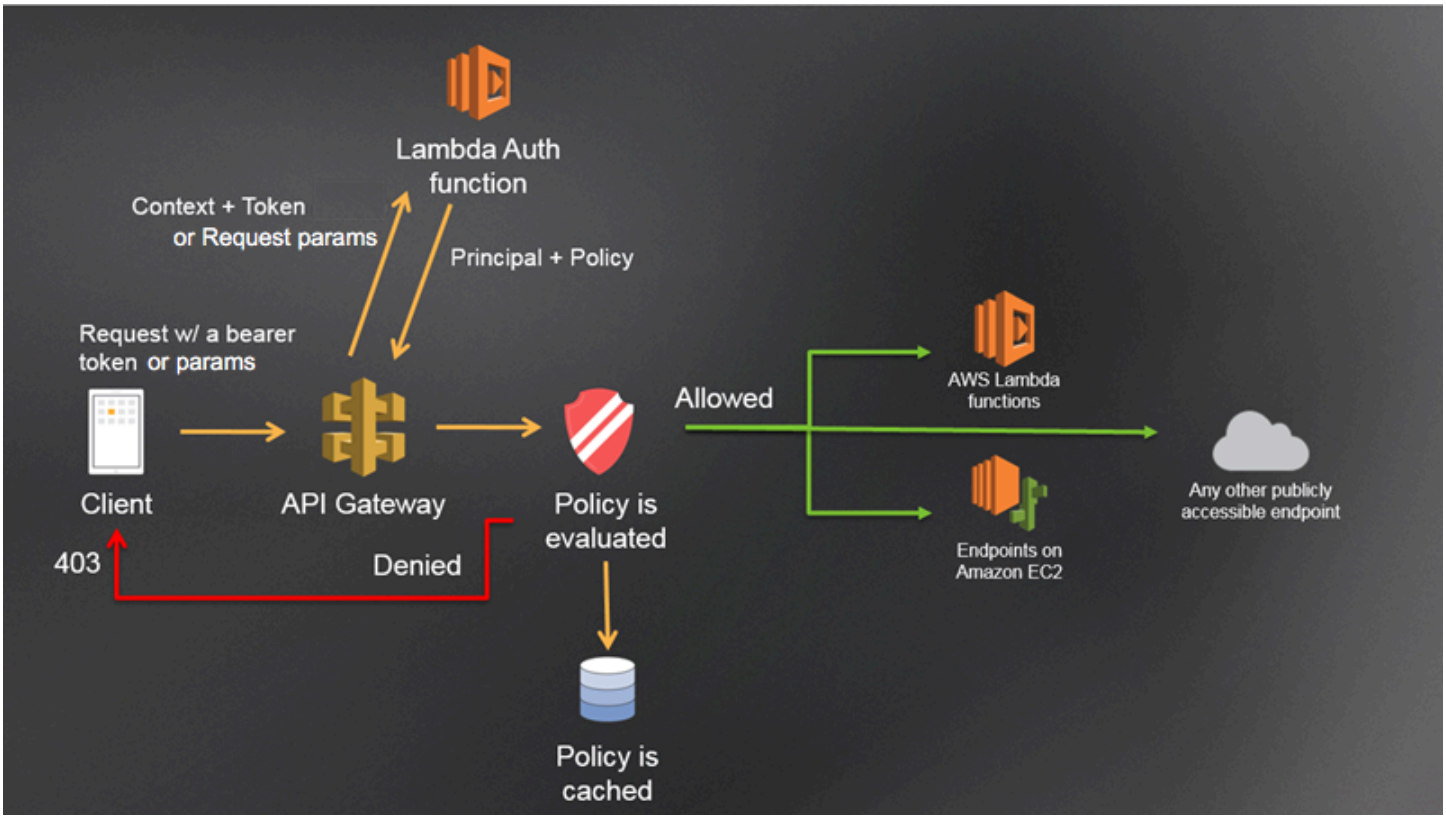
Uso di autorizzazioni Lambda di API Gateway

Usa un sistema di autorizzazione Lambda (noto in precedenza come sistema di autorizzazione ad hoc) per controllare l'accesso all'API. Quando un client effettua una richiesta al metodo dell'API, Gateway API chiama il sistema di autorizzazione Lambda. Il sistema di autorizzazione Lambda accetta l'identità del chiamante come input e restituisce una policy IAM come output.

Usa un sistema di autorizzazione Lambda per implementare uno schema di autorizzazione personalizzato. Lo schema può utilizzare i parametri della richiesta per determinare l'identità del chiamante o utilizzare una strategia di autenticazione tramite token di connessione, ad esempio OAuth o SAML. Crea un autorizzatore Lambda nella console API REST di API Gateway, utilizzando o un AWS CLI SDK. AWS

Flusso di lavoro di autorizzazione per il sistema di autorizzazione Lambda

Il diagramma seguente mostra il flusso di lavoro di autorizzazione per un sistema di autorizzazione Lambda.



Flusso di lavoro per le autorizzazioni Lambda di API Gateway

1. Il client chiama un metodo in un'API di Gateway API, passando un token di connessione o i parametri della richiesta.
2. Gateway API verifica se la richiesta di metodo è configurata con un sistema di autorizzazione Lambda. In tal caso, API Gateway chiama la funzione Lambda.
3. La funzione Lambda autentica il chiamante. La funzione può eseguire l'autenticazione nei modi seguenti:

   - Chiamando un provider OAuth per ottenere un token di accesso OAuth.

- Chiamando un provider SAML per ottenere un'asserzione SAML.
- Generando una policy IAM in base ai valori dei parametri della richiesta.
- Recuperando le credenziali da un database.

4. La funzione Lambda restituisce una policy IAM e un identificatore principale. Se la funzione Lambda non restituisce tali informazioni, la chiamata ha esito negativo.
5. Gateway API valuta la policy IAM.

   - Se l'accesso viene negato, Gateway API restituisce un codice di stato HTTP appropriato, ad esempio `403 ACCESS_DENIED`.
   - Se l'accesso viene concesso, Gateway API invoca il metodo.

     Se si abilita il caching delle autorizzazioni, Gateway API memorizza la policy nella cache in modo che non venga invocata nuovamente la funzione del sistema di autorizzazione Lambda. È necessario assicurarsi che la policy sia applicabile a tutte le risorse e ai metodi dell'API.

È possibile personalizzare le risposte `403 ACCESS_DENIED` o `401 UNAUTHORIZED` del gateway. Per ulteriori informazioni, consulta the section called "Risposte del gateway".

Scelta di un tipo di sistema di autorizzazione Lambda

Esistono due tipi di autorizzazioni Lambda:

Sistema di autorizzazione Lambda basato sui parametri della richiesta (sistema di autorizzazione **`REQUEST`**)

Un sistema di autorizzazione `REQUEST` riceve l'identità del chiamante in una combinazione di intestazioni, parametri della stringa di query, `stageVariables` e variabili `$context`. È possibile utilizzare un sistema di autorizzazione `REQUEST` per creare policy dettagliate in base a informazioni provenienti da più origini di identità, ad esempio le variabili di contesto `$context.path` e `$context.httpMethod`.

Se attivi il caching delle autorizzazioni per un sistema di autorizzazione `REQUEST`, Gateway API verifica che nella richiesta siano presenti tutte le origini di identità specificate. Se un'origine di identità specificata non è presente, è null o è vuota, Gateway API restituisce una risposta HTTP `401 Unauthorized` senza chiamare la funzione del sistema di autorizzazione Lambda. Quando sono definite più origini di identità, vengono utilizzate tutte per derivare la chiave cache del sistema di autorizzazione, mantenendo l'ordine. È possibile definire una chiave di cache dettagliata utilizzando più origini di identità.

Se modifichi qualsiasi parte della chiave di cache, il sistema di autorizzazione rimuove il documento di policy memorizzato nella cache e ne generano uno nuovo.

Se disattivi il caching delle autorizzazioni per un sistema di autorizzazione `REQUEST`, Gateway API passa direttamente la richiesta alla funzione Lambda.

Token-based Autorizzatore Lambda (autorizzatore) **`TOKEN`**

Un sistema di autorizzazione `TOKEN` riceve l'identità del chiamante in un token di connessione, ad esempio un JSON Web Token (JWT) o un token OAuth.

Se attivi il caching delle autorizzazioni per un sistema di autorizzazione `TOKEN`, il nome dell'intestazione specificato nell'origine token diventa la chiave di cache.

Inoltre, puoi utilizzare la convalida dei token per inserire una dichiarazione. RegEx Gateway API esegue la convalida iniziale del token di input per l'espressione e, se la convalida ha esito positivo, invoca la funzione del sistema di autorizzazione Lambda. Ciò aiuta a ridurre le chiamate all'API.

La proprietà `IdentityValidationExpression` è supportata solo per i sistemi di autorizzazione `TOKEN`. Per ulteriori informazioni, consulta the section called “`x-amazon-apigateway-authorizer`”.

> **Note**
>
> È consigliabile utilizzare un sistema di autorizzazione `REQUEST` per controllare l'accesso all'API. Puoi controllare l'accesso all'API in base a più origini di identità quando usi un sistema di autorizzazione `REQUEST` e in base a una singola origine di identità quando usi un sistema di autorizzazione `TOKEN`. Inoltre, puoi separare le chiavi di cache utilizzando più origini di identità per un sistema di autorizzazione `REQUEST`.

Esempio di autorizzazione **`REQUEST`** (funzione Lambda)

Il codice di esempio seguente crea una funzione del sistema di autorizzazione Lambda che consente una richiesta se l'intestazione `HeaderAuth1` fornita dal client, il parametro di query `QueryString1` e la variabile di fase di `StageVar1` corrispondono, rispettivamente, ai valori specificati di `headerValue1`, `queryValue1` e `stageValue1`.

Node.js

```
// A simple request-based authorizer example to demonstrate how to use request
// parameters to allow or deny a request. In this example, a request is
// authorized if the client-supplied HeaderAuth1 header, QueryString1
// query parameter, and stage variable of StageVar1 all match
// specified values of 'headerValue1', 'queryValue1', and 'stageValue1',
// respectively.

export const handler = function(event, context, callback) {
    console.log('Received event:', JSON.stringify(event, null, 2));

    // Retrieve request parameters from the Lambda function input:
    var headers = event.headers;
    var queryStringParameters = event.queryStringParameters;
    var pathParameters = event.pathParameters;
    var stageVariables = event.stageVariables;

    // Parse the input for the parameter values
    var tmp = event.methodArn.split(':');
    var apiGatewayArnTmp = tmp[5].split('/');
    var awsAccountId = tmp[4];
    var region = tmp[3];
    var restApiId = apiGatewayArnTmp[0];
    var stage = apiGatewayArnTmp[1];
    var method = apiGatewayArnTmp[2];
    var resource = '/'; // root resource
    if (apiGatewayArnTmp[3]) {
        resource += apiGatewayArnTmp[3];
    }

    // Perform authorization to return the Allow policy for correct parameters and
    // the 'Unauthorized' error, otherwise.


    if (headers.HeaderAuth1 === "headerValue1"
        && queryStringParameters.QueryString1 === "queryValue1"
        && stageVariables.StageVar1 === "stageValue1") {
        callback(null, generateAllow('me', event.methodArn));
    }  else {
        callback(null, generateDeny('me', event.methodArn));
    }
}
```

```
// Help function to generate an IAM policy
var generatePolicy = function(principalId, effect, resource) {
    // Required output:
    var authResponse = {};
    authResponse.principalId = principalId;
    if (effect && resource) {
        var policyDocument = {};
        policyDocument.Version = '2012-10-17'; // default version
        policyDocument.Statement = [];
        var statementOne = {};
        statementOne.Action = 'execute-api:Invoke'; // default action
        statementOne.Effect = effect;
        statementOne.Resource = resource;
        policyDocument.Statement[0] = statementOne;
        authResponse.policyDocument = policyDocument;
    }
    // Optional output with custom properties of the String, Number or Boolean type.
    authResponse.context = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": true
    };
    return authResponse;
}

var generateAllow = function(principalId, resource) {
    return generatePolicy(principalId, 'Allow', resource);
}

var generateDeny = function(principalId, resource) {
    return generatePolicy(principalId, 'Deny', resource);
}
```

Python

```
# A simple request-based authorizer example to demonstrate how to use request
# parameters to allow or deny a request. In this example, a request is
# authorized if the client-supplied headerauth1 header, QueryString1
# query parameter, and stage variable of StageVar1 all match
# specified values of 'headerValue1', 'queryValue1', and 'stageValue1',
# respectively.

def lambda_handler(event, context):
```

```
    print(event)

    # Retrieve request parameters from the Lambda function input:
    headers = event['headers']
    queryStringParameters = event['queryStringParameters']
    pathParameters = event['pathParameters']
    stageVariables = event['stageVariables']

    # Parse the input for the parameter values
    tmp = event['methodArn'].split(':')
    apiGatewayArnTmp = tmp[5].split('/')
    awsAccountId = tmp[4]
    region = tmp[3]
    restApiId = apiGatewayArnTmp[0]
    stage = apiGatewayArnTmp[1]
    method = apiGatewayArnTmp[2]
    resource = '/'

    if (apiGatewayArnTmp[3]):
        resource += apiGatewayArnTmp[3]

    # Perform authorization to return the Allow policy for correct parameters
    # and the 'Unauthorized' error, otherwise.

    if (headers['HeaderAuth1'] == "headerValue1" and
 queryStringParameters['QueryString1'] == "queryValue1" and
 stageVariables['StageVar1'] == "stageValue1"):
        response = generateAllow('me', event['methodArn'])
        print('authorized')
        return response
    else:
        print('unauthorized')
        response = generateDeny('me', event['methodArn'])
        return response
    # Help function to generate IAM policy


def generatePolicy(principalId, effect, resource):
    authResponse = {}
    authResponse['principalId'] = principalId
    if (effect and resource):
        policyDocument = {}
        policyDocument['Version'] = '2012-10-17'
        policyDocument['Statement'] = []
```

```
        statementOne = {}
        statementOne['Action'] = 'execute-api:Invoke'
        statementOne['Effect'] = effect
        statementOne['Resource'] = resource
        policyDocument['Statement'] = [statementOne]
        authResponse['policyDocument'] = policyDocument

    authResponse['context'] = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": True
    }

    return authResponse


def generateAllow(principalId, resource):
    return generatePolicy(principalId, 'Allow', resource)


def generateDeny(principalId, resource):
    return generatePolicy(principalId, 'Deny', resource)
```

In questo esempio, la funzione di autorizzazione Lambda controlla i parametri di input e si comporta nel modo seguente:

- Se tutti i valori dei parametri richiesti corrispondono ai valori previsti, la funzione di autorizzazione restituisce una risposta HTTP `200 OK` e una policy IAM simile alla seguente e la richiesta del metodo ha esito positivo:

  JSON

  ```
  {
    "Version":"2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Allow",
        "Resource": "arn:aws:execute-api:us-east-1:123456789012:ivdtdhp7b5/
  ESTestInvoke-stage/GET/"
      }
    ]
  ```

```
}
```

- In caso contrario, la funzione del sistema di autorizzazione restituisce una risposta HTTP `401 Unauthorized` e la richiesta di metodo ha esito negativo.

Oltre a restituire una policy IAM, la funzione di autorizzazione Lambda deve restituire anche l'identificatore dell'entità principale dell'intermediario. Facoltativamente, può restituire un oggetto `context` contenente informazioni aggiuntive che possono essere passate al backend di integrazione. Per ulteriori informazioni, consulta Output da un sistema di autorizzazione Lambda di Gateway API.

Nel codice di produzione può essere necessario autenticare l'utente prima di concedere l'autorizzazione. Puoi aggiungere una logica di autenticazione nella funzione Lambda chiamando un provider di autenticazione come indicato nella documentazione del provider.

Esempio di funzione Lambda dell'autorizzatore **`TOKEN`**

Il codice di esempio seguente crea una funzione di autorizzazione `TOKEN` Lambda che consente a un chiamante di invocare un metodo se il valore del token fornito dal client è `allow`. Al chiamante non è consentito invocare la richiesta se il valore del token è `deny`. Se il valore del token è `unauthorized` o una stringa vuota, la funzione del sistema di autorizzazione restituisce una risposta `401 UNAUTHORIZED`.

Node.js

```
// A simple token-based authorizer example to demonstrate how to use an
 authorization token
// to allow or deny a request. In this example, the caller named 'user' is allowed
 to invoke
// a request if the client-supplied token value is 'allow'. The caller is not
 allowed to invoke
// the request if the token value is 'deny'. If the token value is 'unauthorized' or
 an empty
// string, the authorizer function returns an HTTP 401 status code. For any other
 token value,
// the authorizer returns an HTTP 500 status code.
// Note that token values are case-sensitive.

export const handler =  function(event, context, callback) {
    var token = event.authorizationToken;
    switch (token) {
```

```
        case 'allow':
            callback(null, generatePolicy('user', 'Allow', event.methodArn));
            break;
        case 'deny':
            callback(null, generatePolicy('user', 'Deny', event.methodArn));
            break;
        case 'unauthorized':
            callback("Unauthorized");   // Return a 401 Unauthorized response
            break;
        default:
            callback("Error: Invalid token"); // Return a 500 Invalid token response
    }
};

// Help function to generate an IAM policy
var generatePolicy = function(principalId, effect, resource) {
    var authResponse = {};

    authResponse.principalId = principalId;
    if (effect && resource) {
        var policyDocument = {};
        policyDocument.Version = '2012-10-17';
        policyDocument.Statement = [];
        var statementOne = {};
        statementOne.Action = 'execute-api:Invoke';
        statementOne.Effect = effect;
        statementOne.Resource = resource;
        policyDocument.Statement[0] = statementOne;
        authResponse.policyDocument = policyDocument;
    }

    // Optional output with custom properties of the String, Number or Boolean type.
    authResponse.context = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": true
    };
    return authResponse;
}
```

Python

```
# A simple token-based authorizer example to demonstrate how to use an authorization
 token
# to allow or deny a request. In this example, the caller named 'user' is allowed to
 invoke
# a request if the client-supplied token value is 'allow'. The caller is not allowed
 to invoke
# the request if the token value is 'deny'. If the token value is 'unauthorized' or
 an empty
# string, the authorizer function returns an HTTP 401 status code. For any other
 token value,
# the authorizer returns an HTTP 500 status code.
# Note that token values are case-sensitive.

import json


def lambda_handler(event, context):
    token = event['authorizationToken']
    if token == 'allow':
        print('authorized')
        response = generatePolicy('user', 'Allow', event['methodArn'])
    elif token == 'deny':
        print('unauthorized')
        response = generatePolicy('user', 'Deny', event['methodArn'])
    elif token == 'unauthorized':
        print('unauthorized')
        raise Exception('Unauthorized')  # Return a 401 Unauthorized response
        return 'unauthorized'
    try:
        return json.loads(response)
    except BaseException:
        print('unauthorized')
        return 'unauthorized'  # Return a 500 error


def generatePolicy(principalId, effect, resource):
    authResponse = {}
    authResponse['principalId'] = principalId
    if (effect and resource):
        policyDocument = {}
        policyDocument['Version'] = '2012-10-17'
        policyDocument['Statement'] = []
```

```
        statementOne = {}
        statementOne['Action'] = 'execute-api:Invoke'
        statementOne['Effect'] = effect
        statementOne['Resource'] = resource
        policyDocument['Statement'] = [statementOne]
        authResponse['policyDocument'] = policyDocument
    authResponse['context'] = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": True
    }
    authResponse_JSON = json.dumps(authResponse)
    return authResponse_JSON
```

In questo esempio, quando l'API riceve una richiesta del metodo, API Gateway passa il token di origine alla funzione di autorizzazione Lambda nell'attributo `event.authorizationToken`. La funzione di autorizzazione Lambda legge il token e si comporta nel modo seguente:

- Se il valore del token è `allow`, la funzione di autorizzazione restituisce una risposta HTTP `200 OK` e una policy IAM simile alla seguente e la richiesta del metodo ha esito positivo:

  JSON

  ```
  {
    "Version":"2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Allow",
        "Resource": "arn:aws:execute-api:us-east-1:123456789012:ivdtdhp7b5/
  ESTestInvoke-stage/GET/"
      }
    ]
  }
  ```

- Se il valore del token è `deny`, la funzione di autorizzazione restituisce una risposta HTTP `200 OK` e una policy IAM `Deny` simile alla seguente e la richiesta del metodo ha esito negativo:

  JSON

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

```
  "Statement": [
    {
      "Action": "execute-api:Invoke",
      "Effect": "Deny",
      "Resource": "arn:aws:execute-api:us-east-1:123456789012:ivdtdhp7b5/
ESTestInvoke-stage/GET/"
    }
  ]
}
```

> **Note**
>
> Esternamente all'ambiente di test, Gateway API restituisce una risposta HTTP `403 Forbidden` e la richiesta di metodo ha esito negativo.

- Se il valore del token è `unauthorized` o una stringa vuota, la funzione del provider di autorizzazione restituisce una risposta HTTP `401 Unauthorized` e la chiamata al metodo ha esito negativo.
- Se il token ha un valore diverso dai precedenti, il client riceve una risposta `500 Invalid token` e la chiamata al metodo ha esito negativo.

Oltre a restituire una policy IAM, la funzione di autorizzazione Lambda deve restituire anche l'identificatore dell'entità principale dell'intermediario. Facoltativamente, può restituire un oggetto `context` contenente informazioni aggiuntive che possono essere passate al backend di integrazione. Per ulteriori informazioni, consulta Output da un sistema di autorizzazione Lambda di Gateway API.

Nel codice di produzione può essere necessario autenticare l'utente prima di concedere l'autorizzazione. Puoi aggiungere una logica di autenticazione nella funzione Lambda chiamando un provider di autenticazione come indicato nella documentazione del provider.

Esempi aggiuntivi di funzioni del sistema di autorizzazione Lambda

L'elenco seguente mostra altri esempi di funzioni del sistema di autorizzazione Lambda. Puoi creare una funzione Lambda nello stesso account o in un account diverso da quello da cui hai creato l'API.

Per l'esempio precedente di funzioni Lambda, puoi utilizzare le funzioni integrate AWSLambdaBasicExecutionRole, poiché queste funzioni non chiamano altri AWS servizi. Se la

tua funzione Lambda chiama altri AWS servizi, dovrai assegnare un ruolo di esecuzione IAM alla funzione Lambda. Per creare il ruolo, segui le istruzioni in AWS Lambda Ruolo di esecuzione.

Esempi aggiuntivi di funzioni del sistema di autorizzazione Lambda

- Per un esempio di applicazione, vedere Open Banking Brazile - Authorization Samples GitHub on.
- Per altri esempi di funzioni Lambda, consulta aws-apigateway-lambda-authorizer-blueprints on. GitHub
- Puoi creare un sistema di autorizzazione Lambda che autentica gli utenti utilizzando i pool di utenti di Amazon Cognito e autorizza i chiamanti in base a un archivio di policy utilizzando Autorizzazioni verificate. Per ulteriori informazioni, consulta the section called “Controllo dell'accesso in base agli attributi di un'identità con Autorizzazioni verificate”.
- La console Lambda offre un progetto Python, che è possibile utilizzare scegliendo Use a blueprint (Usa un piano) e selezionando il progetto api-gateway-authorizer-python.

Configurazione di un sistema di autorizzazione Lambda di Gateway API

Dopo aver creato una funzione Lambda, configuri la funzione Lambda come sistema di autorizzazione per la tua API. Quindi configuri il metodo per invocare il sistema di autorizzazione Lambda per stabilire se un chiamante può invocare il tuo metodo. Puoi creare una funzione Lambda nello stesso account o in un account diverso da quello da cui hai creato l'API.

Puoi testare il sistema di autorizzazione Lambda utilizzando gli strumenti integrati nella console Gateway API o utilizzando Postman. Per istruzioni su come utilizzare Postman per testare la funzione del sistema di autorizzazione Lambda, consulta the section called “Chiama un'API con le autorizzazioni Lambda”.

Configurazione di un sistema di autorizzazione Lambda (console)

La procedura seguente illustra come creare un sistema di autorizzazione Lambda nella console REST API di Gateway API. Per ulteriori informazioni sui diversi tipi di sistemi di autorizzazione Lambda, consulta the section called “Scelta di un tipo di sistema di autorizzazione Lambda”.

REQUEST authorizer

Per configurare un autorizzatore **`REQUEST`** Lambda

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Seleziona un'API, quindi scegli Autorizzazioni.

3. Scegli Crea autorizzazioni.
4. In Nome del provider di autorizzazioni, immetti il nome di un provider di autorizzazioni.
5. In Tipo di autorizzazione, seleziona Lambda.
6. Per la funzione Lambda, seleziona il Regione AWS luogo in cui hai creato la funzione di autorizzazione Lambda, quindi inserisci il nome della funzione.
7. Non specificare nulla in Ruolo di richiamo Lambda per consentire alla console REST API di Gateway API di impostare una policy basata sulle risorse. La policy concede a Gateway API le autorizzazioni per invocare la funzione del sistema di autorizzazione Lambda. Puoi anche scegliere di immettere il nome di un ruolo IAM per consentire a Gateway API di invocare la funzione del sistema di autorizzazione Lambda. Per un ruolo di esempio, consulta Creazione di un ruolo IAM prevedibile.
8. In Payload evento Lambda, scegli Richiesta.
9. In Tipo di origine identità, seleziona un tipo di parametro. I tipi di parametri supportati sono `Header`, `Query string`, `Stage variable` e `Context`. Per aggiungere altre origini di identità, scegli Aggiungi parametro.
10. Per memorizzare nella cache la policy di autorizzazione generata dalla funzione di autorizzazione, attiva l'opzione Autorizzazione caching. Quando il caching delle policy è abilitato, puoi modificare il valore TTL. L'impostazione dell'opzione TTL su zero disabilita la memorizzazione nella cache delle policy.

    Per abilitare il caching, il sistema di autorizzazione deve restituire una policy applicabile a tutti i metodi di un'API. Per applicare una policy specifica del metodo, utilizza le variabili di contesto `$context.path` e `$context.httpMethod`.
11. Scegli Crea autorizzazioni.

TOKEN authorizer

Per configurare un autorizzatore **`TOKEN`** Lambda

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Seleziona un'API, quindi scegli Autorizzazioni.
3. Scegli Crea autorizzazioni.
4. In Nome del provider di autorizzazioni, immetti il nome di un provider di autorizzazioni.
5. In Tipo di autorizzazione, seleziona Lambda.

6. Per la funzione Lambda, seleziona il Regione AWS luogo in cui hai creato la funzione di autorizzazione Lambda, quindi inserisci il nome della funzione.
7. Non specificare nulla in Ruolo di richiamo Lambda per consentire alla console REST API di Gateway API di impostare una policy basata sulle risorse. La policy concede a Gateway API le autorizzazioni per invocare la funzione del sistema di autorizzazione Lambda. Puoi anche scegliere di immettere il nome di un ruolo IAM per consentire a Gateway API di invocare la funzione del sistema di autorizzazione Lambda. Per un ruolo di esempio, consulta Creazione di un ruolo IAM prevedibile.
8. In Payload evento Lambda, scegli Token.
9. In Origine token, immetti il nome dell'intestazione contenente il token di autorizzazione. Il client API deve includere un'intestazione con questo nome per inviare il token di autorizzazione al sistema di autorizzazione Lambda.
10. (Facoltativo) Per la convalida del token, inserisci una dichiarazione. RegEx API Gateway esegue la convalida iniziale del token di input per l'espressione e, se la convalida ha esito positivo, richiama l'autorizzazione.
11. Per memorizzare nella cache la policy di autorizzazione generata dalla funzione di autorizzazione, attiva l'opzione Autorizzazione caching. Quando la memorizzazione della policy nella cache è abilitata, il nome dell'intestazione specificato in Origine token diventa la chiave della cache. Quando il caching delle policy è abilitato, puoi modificare il valore TTL. L'impostazione dell'opzione TTL su zero disabilita la memorizzazione nella cache delle policy.

    Per abilitare il caching, il sistema di autorizzazione deve restituire una policy applicabile a tutti i metodi di un'API. Per applicare policy specifiche del metodo, puoi disattivare l'opzione Memorizzazione nella cache dell'autorizzazione.
12. Scegli Crea autorizzazioni.

Dopo aver creato il sistema di autorizzazione Lambda, puoi testarlo. La procedura seguente illustra come testare il sistema di autorizzazione Lambda.

REQUEST authorizer

Per testare un autorizzatore **`REQUEST`** Lambda

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Seleziona il nome del sistema di autorizzazione.
3. In Testa autorizzazioni, inserisci un valore per la tua origine di identità.

Se utilizzi the section called “Esempio di autorizzazione REQUEST (funzione Lambda)”, procedi nel modo seguente:

a. Seleziona Intestazione e immetti **`headerValue1`**, quindi scegli Aggiungi parametro.

b. In Tipo di origine identità, seleziona Stringa di query e immetti **`queryValue1`**, quindi scegli Aggiungi parametro.

c. In Tipo di origine identità, seleziona Variabile di fase e immetti **`stageValue1`**.

Non puoi modificare le variabili di contesto per l'invocazione di test, ma puoi modificare il modello di evento di test Sistema di autorizzazione di Gateway API per la tua funzione Lambda. Quindi, puoi testare la funzione del sistema di autorizzazione Lambda con variabili di contesto modificate. Per ulteriori informazioni, consulta Testing Lambda functions in the console nella Guida per gli sviluppatori di AWS Lambda .

4. Scegli Testa autorizzazioni.

TOKEN authorizer

Per testare un autorizzatore **`TOKEN`** Lambda

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Seleziona il nome del sistema di autorizzazione.
3. In Testa autorizzazioni, inserisci un valore per il token.

   Se utilizzi the section called “Esempio di funzione Lambda dell'autorizzatore TOKEN”, procedi nel modo seguente:

   - Per authorizationToken, inserisci **`allow`**.
4. Scegli Testa autorizzazioni.

   Se il sistema di autorizzazione Lambda nega correttamente una richiesta nell'ambiente di test, il test restituisce una risposta HTTP `200 OK`. Esternamente all'ambiente di test, invece, Gateway API restituisce una risposta HTTP `403 Forbidden` e la richiesta di metodo ha esito negativo.

Configurare un autorizzatore Lambda (AWS CLI)

Il seguente comando create-authorizer mostra come creare un sistema di autorizzazione Lambda tramite AWS CLI.

REQUEST authorizer

Il comando create-authorizer seguente crea un sistema di autorizzazione `REQUEST` e utilizza l'intestazione `Authorizer` e la variabile di contesto `accountId` come origini di identità:

```
aws apigateway create-authorizer \
    --rest-api-id 1234123412 \
    --name 'First_Request_Custom_Authorizer' \
    --type REQUEST \
    --authorizer-uri 'arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123412341234:function:customAuthFunction/invocations' \
    --identity-source 'method.request.header.Authorization,context.accountId' \
    --authorizer-result-ttl-in-seconds 300
```

TOKEN authorizer

Il comando create-authorizer seguente crea un sistema di autorizzazione `TOKEN` e utilizza l'intestazione `Authorization` come origine di identità:

```
aws apigateway create-authorizer \
    --rest-api-id 1234123412 \
    --name 'First_Token_Custom_Authorizer' \
    --type TOKEN \
    --authorizer-uri 'arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123412341234:function:customAuthFunction/invocations' \
    --identity-source 'method.request.header.Authorization' \
    --authorizer-result-ttl-in-seconds 300
```

Dopo aver creato il sistema di autorizzazione Lambda, puoi testarlo. Il comando test-invoke-authorizer seguente testa un sistema di autorizzazione Lambda:

```
aws apigateway test-invoke-authorizer --rest-api-id 1234123412 \
   --authorizer-id efg1234 \
   --headers Authorization='Value'
```

Configurazione di un metodo per utilizzare un sistema di autorizzazione Lambda (console)

Dopo aver configurato il sistema di autorizzazione Lambda, devi collegarlo a un metodo per l'API. Se il sistema di autorizzazione utilizza il caching delle autorizzazioni, è necessario aggiornare la policy per controllare l'accesso del metodo aggiuntivo.

Configurazione di un metodo API per utilizzare le autorizzazioni Lambda

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Seleziona un'API.
3. Seleziona Risorse, quindi scegli un nuovo metodo o scegline uno esistente.
4. Nella scheda Richiesta metodo, in Impostazioni richiesta metodo, scegli Modifica.
5. In Autorizzazioni, nel menu a discesa, seleziona la funzione di autorizzazione Lambda appena creata.
6. (Facoltativo) Se desideri passare il token di autorizzazione al back-end, scegli Intestazioni di richiesta HTTP. Scegli Aggiungi intestazione, quindi aggiungi il nome dell'intestazione di autorizzazione. In Nome immetti il nome dell'intestazione corrispondente al nome specificato in Origine token al momento della creazione della funzione di autorizzazione Lambda per l'API. Questo passaggio non si applica alle autorizzazioni `REQUEST`.
7. Scegli Save (Salva).
8. Seleziona Deploy API (Distribuisci API) per distribuire l'API in una fase. Per una funzione di autorizzazione basata su `REQUEST` che usa variabili di fase, devi anche definire le variabili di fase richieste e specificare i relativi valori in Fasi.

Configurare un metodo per utilizzare un autorizzatore Lambda (AWS CLI)

Dopo aver configurato il sistema di autorizzazione Lambda, devi collegarlo a un metodo per l'API. Puoi creare un nuovo metodo o utilizzare un'operazione di patch per collegare un sistema di autorizzazione a un metodo esistente. Se il sistema di autorizzazione utilizza il caching delle autorizzazioni, è necessario aggiornare la policy per controllare l'accesso del metodo aggiuntivo.

Il comando put-method seguente crea un nuovo metodo che utilizza un sistema di autorizzazione Lambda:

```
aws apigateway put-method --rest-api-id 1234123412 \
  --resource-id a1b2c3 \
  --http-method PUT \
```

```
  --authorization-type CUSTOM \
  --authorizer-id efg1234
```

Il comando update-method seguente aggiorna un metodo esistente in modo da utilizzare un sistema di autorizzazione Lambda:

```
aws apigateway update-method \
    --rest-api-id 1234123412 \
    --resource-id a1b2c3 \
    --http-method PUT \
    --patch-operations op="replace",path="/authorizationType",value="CUSTOM"
 op="replace",path="/authorizerId",value="efg1234"
```

Input per un sistema di autorizzazione Lambda di Gateway API

La sezione seguente illustra il formato di input da Gateway API a un sistema di autorizzazione Lambda.

Formato di input TOKEN

Per l'autorizzazione Lambda (nota in precedenza come autorizzazione ad hoc) di tipo `TOKEN`, devi specificare un'intestazione personalizzata in Token Source (Origine token) al momento della configurazione dell'autorizzazione per l'API. Il client API deve passare il token di autorizzazione richiesto nell'intestazione nella richiesta in ingresso. Quando riceve la richiesta del metodo in entrata, API Gateway estrae il token dall'intestazione personalizzata. Passa quindi il token come proprietà `authorizationToken` dell'oggetto `event` della funzione Lambda, in aggiunta all'ARN del metodo come proprietà `methodArn`:

```
{
    "type":"TOKEN",
    "authorizationToken":"{caller-supplied-token}",
    "methodArn":"arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/{httpVerb}/
[{resource}/[{child-resources}]]"
}
```

In questo esempio la proprietà `type` specifica il tipo di autorizzazioni, ovvero `TOKEN`. L'oggetto *`{caller-supplied-token}`* deriva dall'intestazione di autorizzazioni in una richiesta client e può essere qualsiasi valore di stringa. `methodArn` è l'ARN della richiesta del metodo in entrata e viene popolato da API Gateway in base alla configurazione dell'autorizzazione Lambda.

Formato **`di input REQUEST`**

Per un'autorizzazione per Lambda di tipo `REQUEST`, API Gateway passa i parametri della richiesta alla funzione di autorizzazione Lambda come parte dell'oggetto `event`. I parametri della richiesta includono intestazioni, parametri di percorso, parametri della stringa di query, variabili di fase e alcune variabili di contesto della richiesta. L'autore della chiamata API può impostare parametri di percorso, intestazioni e parametri della stringa di query. Lo sviluppatore dell'API deve impostare le variabili di fase durante la distribuzione dell'API e API Gateway fornisce il contesto della richiesta in fase di runtime.

> **Note**
>
> I parametri di percorso possono essere passati come parametri della richiesta alla funzione di autorizzazione Lambda, ma non possono essere utilizzati come origini di identità.

L'esempio seguente mostra un input per le autorizzazioni `REQUEST` per un metodo API (`GET /request`) con integrazione 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` è una mappa di coppie chiave/valore e corrisponde alla variabile $context. Il suo risultato è API-dependent.

Gateway API potrebbe aggiungere nuove chiavi alla mappa. Per ulteriori informazioni sull'input della funzione Lambda nell'integrazione proxy Lambda, consulta Formato di input di una funzione Lambda per l'integrazione proxy.

Output da un sistema di autorizzazione Lambda di Gateway API

L'output di una funzione di autorizzazione Lambda è un oggetto simile a un dizionario che deve includere l'identificatore dell'entità principale (`principalId`) e un documento di policy (`policyDocument`) contenente un elenco di dichiarazioni di policy. L'output può includere anche una mappa `context` contenente coppie chiave-valore. Se l'API utilizza un piano di utilizzo (`apiKeySource` è impostato su `AUTHORIZER`), la funzione di autorizzazione Lambda deve restituire una delle chiavi API del piano di utilizzo come il valore della proprietà `usageIdentifierKey`.

Di seguito è illustrato un esempio di output.

JSON

```
{
  "principalId": "yyyyyyyy",
  "policyDocument": {
    "Version":"2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Allow|Deny",
        "Resource": "arn:aws:execute-
api:{regionId}:{accountId}:{apiId}/{stage}/{httpVerb}/[{resource}/[{child-
resources}]]"
      }
    ]
  },
  "context": {
    "stringKey": "value",
    "numberKey": "1",
    "booleanKey": "true"
  },
  "usageIdentifierKey": "{api-key}"
}
```

In questo caso, una dichiarazione di policy specifica se permettere o rifiutare (`Effect`) al servizio di esecuzione API Gateway di richiamare (`Action`) il metodo API specificato (`Resource`). Potrebbe essere necessario controllare l'accesso a più risorse in base al sistema di autorizzazione. Per specificare un tipo di risorsa (metodo), è possibile usare un carattere jolly (`*`). Per informazioni sull'impostazione di policy valide per chiamare un'API, consulta Riferimento delle istruzioni delle policy IAM per l'esecuzione dell'API in API Gateway.

Per un ARN di metodo abilitato per le autorizzazioni, ad esempio `arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/{httpVerb}/[{resource}/[{child-resources}]]`, la lunghezza massima è 1600 byte. A causa dei valori dei parametri del percorso, la cui dimensione viene determinata al momento dell'esecuzione, l'ARN può superare il limite di lunghezza impostato. In questo caso, il client API riceverà una risposta `414 Request URI too long`.

Per l'ARN di risorsa, inoltre, come illustrato nell'output della dichiarazione di policy da parte delle autorizzazioni, è attualmente previsto un limite di 512 caratteri. Per questo motivo, non devi usare un URI con un token JWT di lunghezza significativa in un URI di richiesta. Puoi invece passare senza problemi il token JWT in un'intestazione di richiesta.

Puoi accedere al valore di `principalId` in un modello di mappatura usando la variabile `$context.authorizer.principalId`. Ciò è utile se desideri passare il valore al back-end. Per ulteriori informazioni, consulta Variabili di contesto per le trasformazioni dei dati.

Puoi accedere al valore di `stringKey`, `numberKey` o `booleanKey` (ad esempio `"value"`, `"1"` o `"true"`) della mappa `context` in un modello di mappatura chiamando, rispettivamente, `$context.authorizer.stringKey`, `$context.authorizer.numberKey` o `$context.authorizer.booleanKey`. I valori restituiti sono tutti in formato stringa. Non è possibile impostare una matrice o un oggetto JSON come valore valido di una chiave nella mappa `context`.

È possibile usare la mappa `context` per restituire le credenziali memorizzate nella cache dalle autorizzazioni al back-end, usando un modello di mappatura di richiesta di integrazione. In questo modo, il back-end può offrire un'esperienza utente migliore usando le credenziali memorizzate nella cache per ridurre la necessità di accesso alle chiavi segrete e di apertura dei token di autorizzazione per ogni richiesta.

Per l'integrazione proxy Lambda, API Gateway passa l'oggetto `context` da un'autorizzazione Lambda direttamente alla funzione Lambda back-end come parte dell'input `event`. È possibile recuperare la coppia chiave/valore `context` nella funzione Lambda chiamando `$event.requestContext.authorizer.`*`key`*.

`{api-key}` indica una chiave API nel piano di utilizzo della fase API. Per ulteriori informazioni, consulta the section called “Piani di utilizzo”.

Di seguito è illustrato l'output dell'autorizzazione Lambda di esempio. L'output di esempio contiene una dichiarazione politica per bloccare (`Deny`) le chiamate al `GET` metodo per la `dev` fase di un'API (`ymy8tbxw7b`) di un AWS account (`123456789012`).

JSON

```
{
  "principalId": "user",
  "policyDocument": {
    "Version":"2012-10-17",
```

```
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Deny",
        "Resource": "arn:aws:execute-api:us-west-2:123456789012:ymy8tbxw7b/dev/
GET/"
      }
    ]
  }
}
```

Chiamata di un'API con il sistema di autorizzazione Lambda di Gateway API

Dopo aver configurato l'autorizzazione Lambda (nota in precedenza come autorizzazione ad hoc) e aver distribuito l'API, devi testare l'API con l'autorizzazione Lambda abilitata. A tale scopo, è necessario un client REST, ad esempio cURL o Postman. Per gli esempi seguenti usiamo Postman.

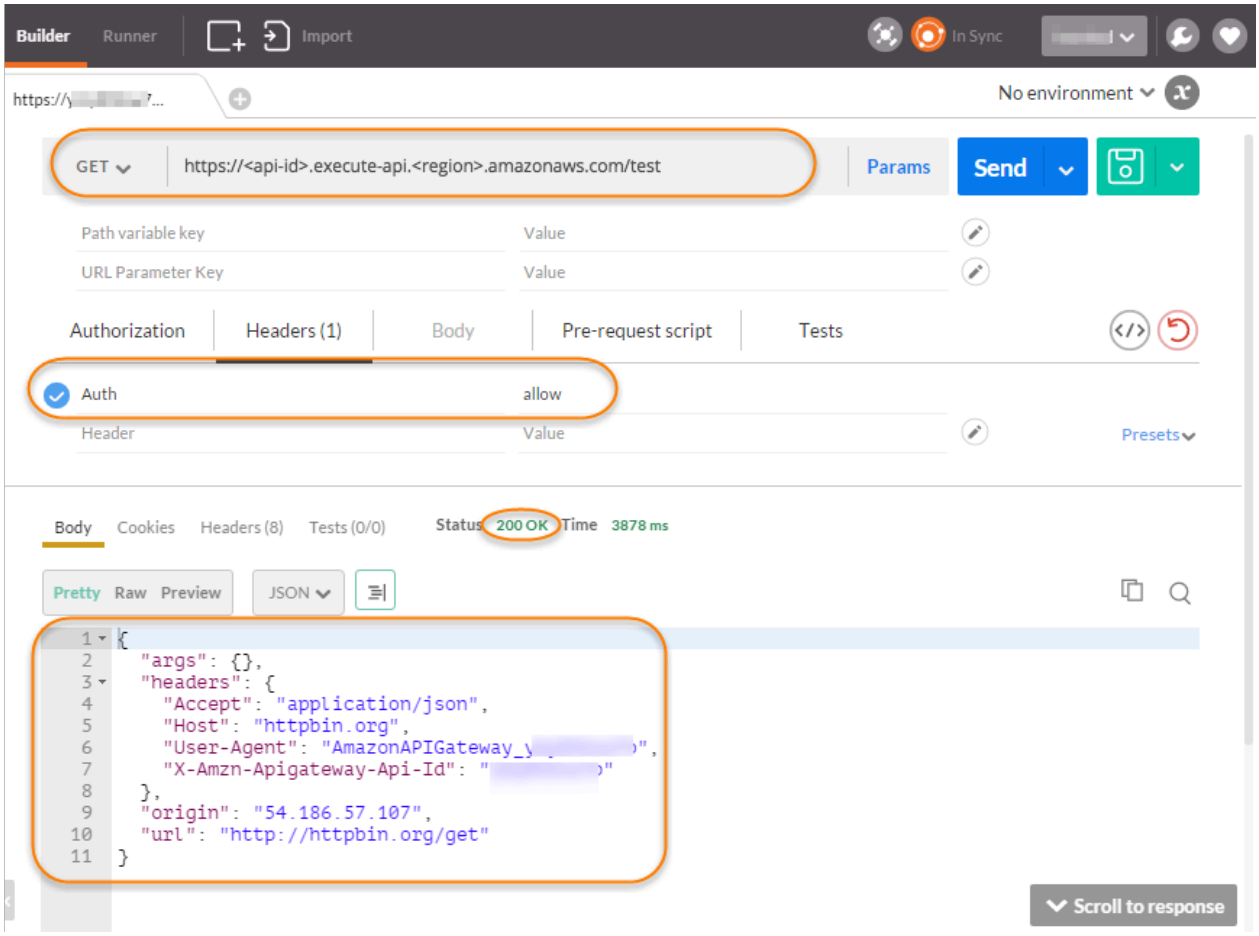
> **Note**
>
> Quando si chiama un metodo abilitato all'autorizzazione, API Gateway non registra la chiamata CloudWatch se il token richiesto per l'**`TOKEN`**autorizzatore non è impostato, è nullo o è invalidato dall'espressione di convalida Token specificata. Allo stesso modo, API Gateway non registra la chiamata CloudWatch se una delle fonti di identità richieste per l'`REQUEST`autorizzatore non è impostata, è nulla o è vuota.

Di seguito viene illustrato come usare Postman per chiamare o testare un'API con l'autorizzazione Lambda `TOKEN`. Il metodo può essere applicato alla chiamata a un'API con l'autorizzazione Lambda `REQUEST` specificando in modo esplicito i parametri di stringa di query, percorso o intestazione richiesti.

`Per chiamare un'API con l'autorizzatore TOKEN personalizzato`

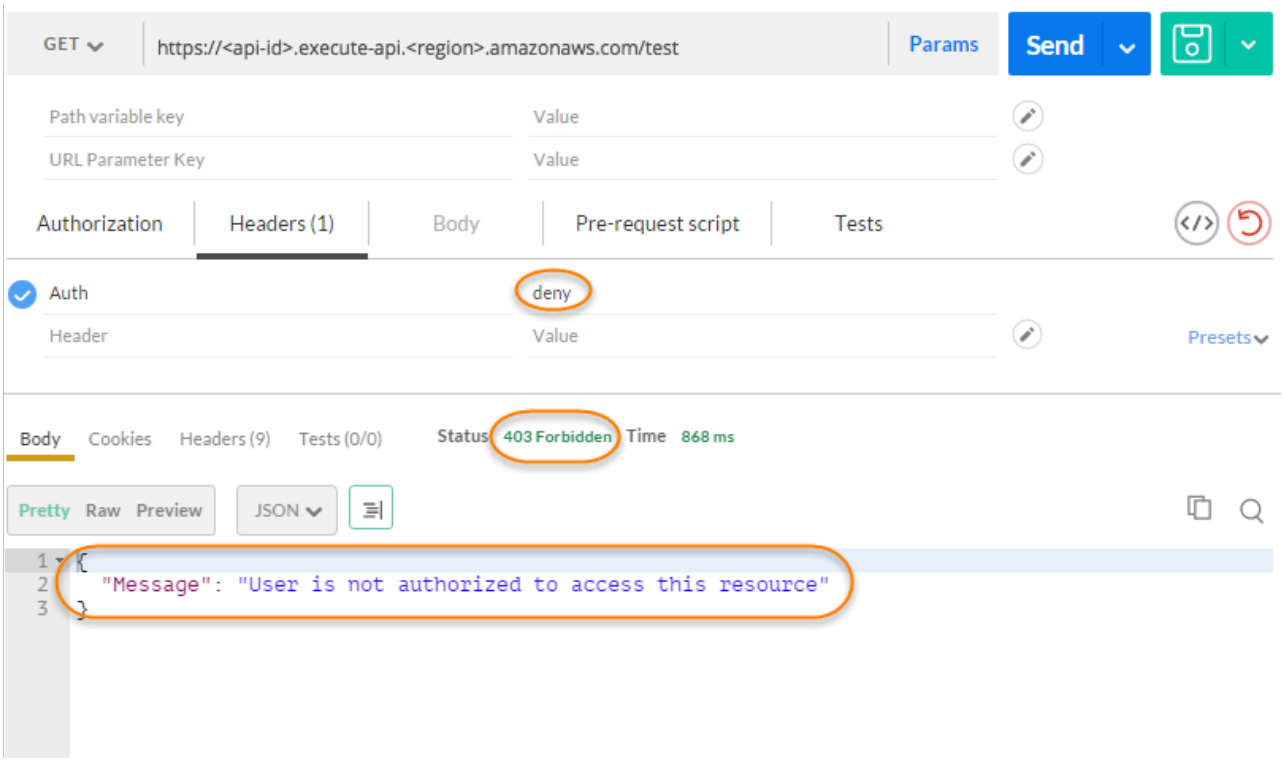
1. Apri Postman, scegli il metodo GET e incolla il valore di Invoke URL (URL chiamata) dell'API nel campo URL adiacente.

   Aggiungi l'intestazione del token di autorizzazione Lambda e imposta il valore su `allow`. Scegliere Send (Invia).



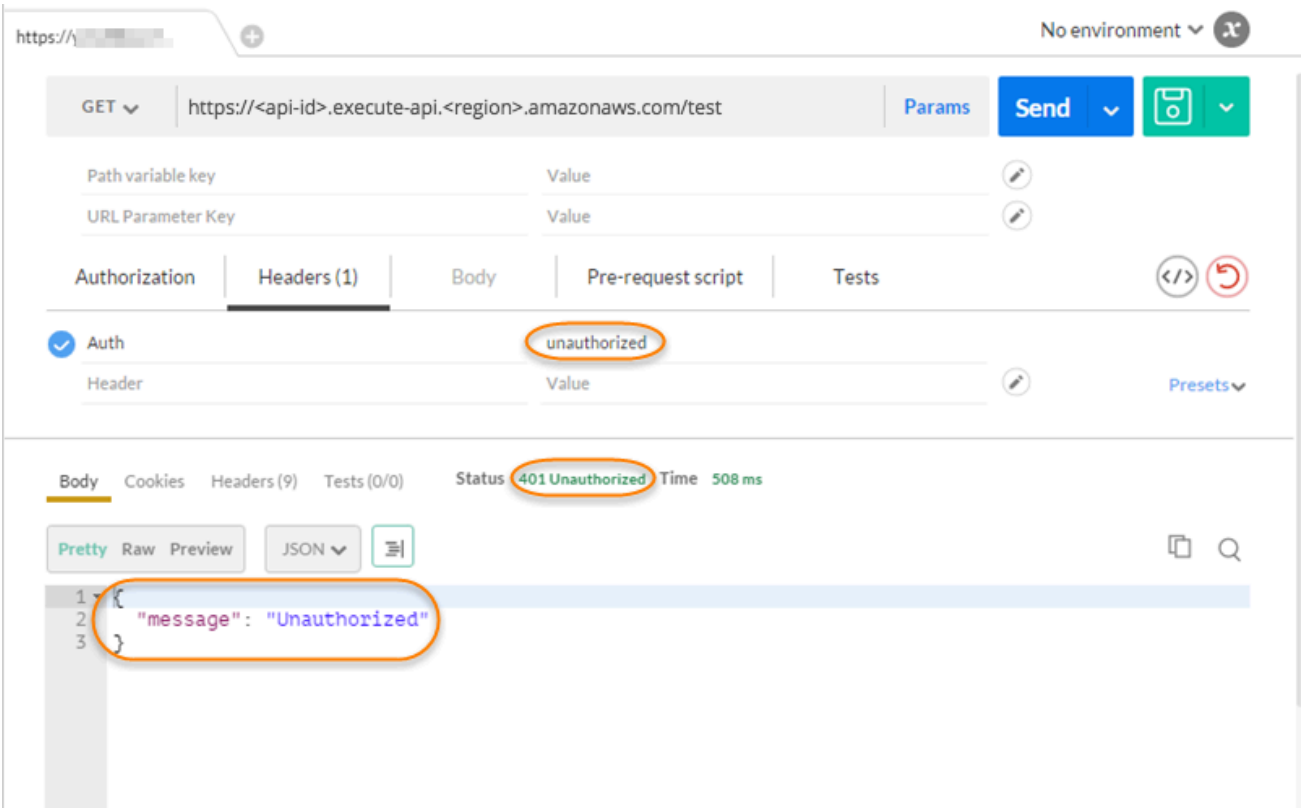
La risposta mostra che l'autorizzatore API Gateway Lambda restituisce una risposta di 200 OK e autorizza correttamente la chiamata per accedere all'endpoint HTTP (http://httpbin.org/get) integrato con il metodo.

2. Sempre in Postman, modifica il valore dell'intestazione del token di autorizzazione Lambda impostandolo su `deny`. Scegliere Send (Invia).



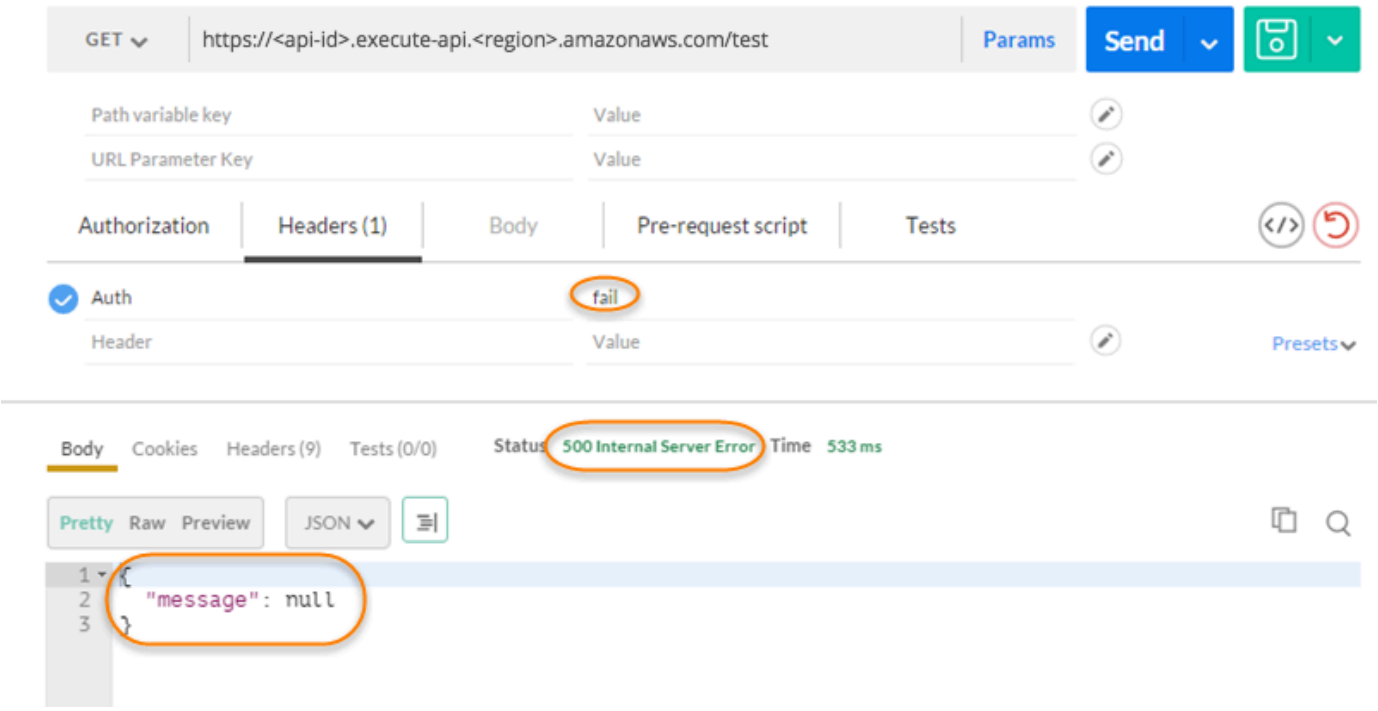
La risposta mostra che l'autorizzazione Lambda di API Gateway restituisce una risposta 403 Forbidden (403 Non consentito) senza autorizzare la chiamata per l'accesso all'endpoint HTTP.

3. In Postman, modifica il valore dell'intestazione del token di autorizzazione Lambda impostandolo su `unauthorized` e seleziona Send (Invia).



La risposta indica che API Gateway restituisce una risposta 401 Unauthorized (401 Non autorizzato) senza autorizzare la chiamata per l'accesso all'endpoint HTTP.

4. Modifica quindi il valore dell'intestazione del token di autorizzazione Lambda impostandolo su `fail`. Scegliere Send (Invia).



La risposta indica che API Gateway restituisce una risposta 500 Internal Server Error (500 Errore interno del server) senza autorizzare la chiamata per l'accesso all'endpoint HTTP.

Configurazione di un sistema di autorizzazione Lambda di Gateway API multi-account

Ora puoi anche utilizzare una AWS Lambda funzione di un altro AWS account come funzione di autorizzazione dell'API. Ogni account può trovarsi in qualsiasi regione in cui è disponibile Amazon API Gateway. La funzione di autorizzazione Lambda può utilizzare strategie di autenticazione con token di connessione come OAuth o SAML. In questo modo, è più semplice gestire e condividere a livello centralizzato le autorizzazioni Lambda tra più API di API Gateway.

In questa sezione, viene illustrato come configurare un'autorizzazione Lambda tra account tramite la console Amazon API Gateway.

Queste istruzioni presuppongono che tu abbia già un'API API Gateway in un AWS account e una funzione di autorizzazione Lambda in un altro account.

Configurazione di un'autorizzazione Lambda tra account tramite la console API Gateway

Accedi alla console Gateway Amazon API con l'account contenente la tua API e quindi procedi come descritto di seguito:

1. Scegli la tua API, quindi nel pannello di navigazione principale, scegli Autorizzazioni.

2. Scegli Crea autorizzazioni.
3. In Nome del provider di autorizzazioni, immetti il nome di un provider di autorizzazioni.
4. In Tipo di autorizzazione, seleziona Lambda.
5. In Funzione Lambda, copia e incolla l'ARN completo della funzione di autorizzazione Lambda nel secondo account.

> **Note**
>
> Nella console Lambda, puoi trovare l'ARN della funzione nell'angolo in alto a destra.

6. Verrà visualizzato un avviso con una stringa del comando `aws lambda add-permission`. La policy concede a Gateway API l'autorizzazione per richiamare la funzione di autorizzazione Lambda. Copiare il comando e salvarlo per un secondo momento. È possibile eseguire il comando dopo aver creato l'autorizzatore.
7. In Payload evento Lambda, scegli Token per una funzione di autorizzazione basata su `TOKEN` o Richiesta per una funzione di autorizzazione basata su `REQUEST`.
8. A seconda dell'opzione scelta in precedenza, esegui una di queste operazioni:

   a. Per l'opzione Token esegui le operazioni indicate di seguito:

   - In Origine token, immetti il nome dell'intestazione contenente il token di autorizzazione. Il client API deve includere un'intestazione con questo nome per inviare il token di autorizzazione all'autorizzazione Lambda.
   - Facoltativamente, per la convalida del token, inserisci una dichiarazione. RegEx API Gateway esegue la convalida iniziale del token di input per l'espressione e, se la convalida ha esito positivo, richiama l'autorizzazione. Ciò aiuta a ridurre le chiamate all'API.
   - Per memorizzare nella cache la policy di autorizzazione generata dalla funzione di autorizzazione, attiva l'opzione Autorizzazione caching. Quando la memorizzazione nella cache delle policy è abilitata, è possibile scegliere di modificare il valore TTL. L'impostazione dell'opzione TTL su zero disabilita la memorizzazione nella cache delle policy. Quando la memorizzazione della policy nella cache è abilitata, il nome dell'intestazione specificato in Origine token diventa la chiave della cache. Se nella richiesta vengono passati più valori a questa intestazione, tutti i valori diventeranno la chiave della cache, con l'ordine mantenuto.

> **Note**
>
> Il valore TTL predefinito è 300 secondi. Il valore massimo è 3600 secondi e non è possibile aumentare questo limite.

b. Per l'opzione Request (Richiesta), esegui queste operazioni:

- In Tipo di origine identità, seleziona un tipo di parametro. I tipi di parametri supportati sono `Header`, `Query string`, `Stage variable` e `Context`. Per aggiungere altre origini di identità, scegli Aggiungi parametro.
- Per memorizzare nella cache la policy di autorizzazione generata dalla funzione di autorizzazione, attiva l'opzione Autorizzazione caching. Quando la memorizzazione nella cache delle policy è abilitata, è possibile scegliere di modificare il valore TTL. L'impostazione dell'opzione TTL su zero disabilita la memorizzazione nella cache delle policy.

  API Gateway usa le origini di identità specificate come chiave di caching delle autorizzazioni della richiesta. Quando il caching è abilitato, API Gateway chiama la funzione di autorizzazione Lambda solo dopo aver verificato che tutte le origini di identità specificate siano presenti in fase di runtime. Se un'origine di identità specificata non è presente, è null o è vuota, Gateway API restituisce una risposta `401 Unauthorized` senza chiamare la funzione Lambda del sistema di autorizzazione.

  Quando sono definite più origini di identità, vengono utilizzate tutte per derivare la chiave cache delle autorizzazioni. Se vengono modificate parti della chiave cache, le autorizzazioni rimuovono il documento di policy memorizzato nella cache e ne generano uno nuovo. Se nella richiesta viene passata un'intestazione con più valori, tutti i valori diventeranno la chiave della cache, con l'ordine mantenuto.
- Quando la memorizzazione nella cache è disattivata, non è necessario specificare un'origine di identità.

> **Note**
>
> Per abilitare il caching, le autorizzazioni devono restituire una policy applicabile a tutti i metodi di un'API. Per applicare policy specifiche del metodo, puoi disattivare l'opzione Autorizzazione caching.

9. Scegli Crea autorizzazioni.
10. Incolla la stringa di `aws lambda add-permission` comando che hai copiato nel passaggio precedente in una AWS CLI finestra configurata per il tuo secondo account. Sostituire `AUTHORIZER_ID` con l'ID del proprio autorizzatore. In questo modo, al primo account verrà concesso l'accesso alla funzione di autorizzazione Lambda del secondo account.

Controllo dell'accesso in base agli attributi di un'identità con Autorizzazioni verificate

Utilizza Autorizzazioni verificate da Amazon per controllare l'accesso alla tua API di Gateway API. Quando utilizzi Gateway API con Autorizzazioni verificate, Autorizzazioni verificate crea un sistema di autorizzazione Lambda che utilizza decisioni di autorizzazione granulari per controllare l'accesso alla tua API. Autorizzazioni verificate autorizza i chiamanti sulla base di uno schema di archivio di policy e di policy che utilizzano il linguaggio di policy Cedar per definire autorizzazioni granulari per gli utenti dell'applicazione. Per ulteriori informazioni, consulta Creare un archivio di policy con un'API e un provider di identità connessi nella Guida per l'utente di Autorizzazioni verificate da Amazon.

Autorizzazioni verificate supporta i pool di utenti di Amazon Cognito o i provider di identità OpenID Connect (OIDC) come origini di identità. Autorizzazioni verificate presuppone che il principale sia stato precedentemente identificato e autenticato. Autorizzazione verificate è supportato solo per le REST API regionali e ottimizzate per l'edge.

Creazione di un sistema di autorizzazione Lambda con Autorizzazioni verificate

Autorizzazioni verificate crea un sistema di autorizzazione Lambda per determinare se un principale è autorizzato a eseguire un'operazione sulla tua API. La policy Cedar utilizzata da Autorizzazioni verificate per eseguire le sue attività di autorizzazione viene creata da te.

Di seguito è riportato un esempio di policy Cedar che consente l'accesso per invocare un'API in base al pool di utenti di Amazon Cognito`us-east-1_ABC1234`, per il gruppo `developer` sulla risorsa `GET /users` di un'API. Autorizzazioni verificate determina l'appartenenza al gruppo analizzando l'identità del chiamante nel token di connessione.

```
permit(
  principal in MyAPI::UserGroup::"us-east-1_ABC1234|developer",
  action in [ MyAPI::Action::"get /users" ],
  resource
  );
```

Facoltativamente, Autorizzazioni verificate può collegare il sistema di autorizzazione ai metodi dell'API. Nelle fasi di produzione dell'API, è consigliabile non consentire ad Autorizzazioni verificate di collegare automaticamente il sistema di autorizzazione.

L'elenco seguente mostra come configurare Autorizzazioni verificate per collegare o non collegare il sistema di autorizzazione Lambda alla richiesta di metodo dei metodi dell'API.

Sistema di autorizzazione collegato automaticamente (Console di gestione AWS)

Quando scegli Crea archivio di policy nella console Autorizzazioni verificate, nella pagina Implementa integrazione app scegli Ora.

Sistema di autorizzazione non collegato automaticamente (Console di gestione AWS)

Quando scegli Crea archivio di policy nella console Autorizzazioni verificate, nella pagina Implementa integrazione app scegli Più tardi.

Autorizzazioni verificate crea comunque automaticamente un sistema di autorizzazione Lambda. Il sistema di autorizzazione Lambda inizia con `AVPAuthorizerLambda-`. Per ulteriori indicazioni su come collegare il sistema di autorizzazione per un metodo, consulta the section called “Configurazione di un metodo per utilizzare un sistema di autorizzazione Lambda (console)”.

Sistema di autorizzazione collegato automaticamente (CloudFormation)

Nel Permissions-generated CloudFormation modello verificato, nella `Conditions` sezione, imposta su`"Ref": "shouldAttachAuthorizer"`. `true`

Sistema di autorizzazione non collegato automaticamente (CloudFormation)

Nel Permissions-generated CloudFormation modello verificato, nella `Conditions` sezione, imposta `"Ref": "shouldAttachAuthorizer"` su`false`.

Autorizzazioni verificate crea comunque automaticamente un sistema di autorizzazione Lambda. Il sistema di autorizzazione Lambda inizia con `AVPAuthorizerLambda-`. Per ulteriori indicazioni su come collegare il sistema di autorizzazione per un metodo, consulta the section called “Configurare un metodo per utilizzare un autorizzatore Lambda (AWS CLI)”.

Chiamata di un sistema di autorizzazione Lambda con Autorizzazioni verificate

Puoi chiamare il tuo sistema di autorizzazione Lambda specificando un'identità o un token di accesso nell'intestazione `Authorization`. Per ulteriori informazioni, consulta the section called “Chiama un'API con le autorizzazioni Lambda”.

Gateway API memorizza nella cache la policy restituita dal sistema di autorizzazione Lambda per 120 secondi. È possibile modificare il TTL nella console Gateway API o tramite AWS CLI.

Controllo degli accessi alle REST API utilizzando pool di utenti di Amazon Cognito come sistema di autorizzazione

In alternativa all'utilizzo di ruoli e policy IAM o autorizzazioni Lambda (note in precedenza come autorizzazioni ad hoc), puoi utilizzare un pool di utenti di Amazon Cognito per controllare chi può accedere all'API in Amazon API Gateway.

Per usare un pool di utenti di Amazon Cognito con l'API, devi prima creare l'autorizzazione di tipo `COGNITO_USER_POOLS` e quindi configurare un metodo API per usare tale autorizzazione. Una volta distribuita l'API, il client deve innanzitutto registrare l'utente nel pool di utenti, ottenere un token di identità o di accesso per l'utente e quindi chiamare il metodo API con uno dei token, che sono in genere impostati sull'intestazione `Authorization` della richiesta. La chiamata API riesce solo se viene fornito il token necessario e se questo è valido. In caso contrario, il client non è autorizzato ad effettuare la chiamata perché non ha credenziali che è stato possibile autorizzare.

Il token di identità viene usato per autorizzare chiamate API in base alle richieste di identità dell'utente connesso. Il token di accesso viene usato per autorizzare chiamate API in base agli ambiti personalizzati di risorse con accesso protetto specificate. Per ulteriori informazioni, consulta Utilizzo di token con pool di utenti e la pagina relativa a server di risorse e ambiti personalizzati.

Per creare e configurare un pool di utenti di Amazon Cognito per l'API, esegui le attività seguenti:

- Usa la console o l'API di Amazon Cognito per creare un pool di utenti o usane uno di proprietà di un altro account. CLI/SDK AWS
- Utilizza la console API Gateway o CLI/SDK l'API per creare un autorizzatore API Gateway con il pool di utenti scelto.
- Utilizza la console API Gateway o l'API per abilitare l'autorizzatore su metodi API selezionati. CLI/SDK

Per chiamare qualsiasi metodo API con un pool di utenti abilitato, i client API eseguono le attività seguenti:

- Usa Amazon Cognito CLI/SDK o l'API per far accedere un utente al pool di utenti scelto e ottenere un token di identità o un token di accesso. Per ulteriori informazioni sull'utilizzo degli SDK, consulta Esempi di codice per Amazon Cognito AWS utilizzando gli SDK.

- Utilizzano un framework specifico del client per chiamare l'API di API Gateway distribuita e specificare il token appropriato nell'intestazione `Authorization`.

In quanto sviluppatore dell'API, devi fornire agli sviluppatori client l'ID pool di utenti, un ID client e possibilmente i segreti client definiti come parte del pool di utenti.

> **Note**
>
> Per permettere a un utente di accedere usando credenziali Amazon Cognito e anche di ottenere credenziali temporanee da usare con le autorizzazioni di un ruolo IAM, usa identità federate di Amazon Cognito. Per ogni metodo HTTP dell'endpoint della risorsa API, imposta il tipo di autorizzazione, la categoria `Method Execution`, su `AWS_IAM`.

In questa sezione descriveremo come creare un pool di utenti, come integrare un'API di API Gateway con il pool di utenti e come richiamare un'API integrata con il pool di utenti.

Argomenti

- Creazione di un pool di utenti di Amazon Cognito per un'API REST
- Integrazione di un'API REST con un pool di utenti di Amazon Cognito
- Chiamata di un'API REST integrata con un pool di utenti di Amazon Cognito
- Configurazione dell'autorizzazione Amazon Cognito tra account per un'API REST tramite la console API Gateway
- Crea un autorizzatore Amazon Cognito per un'API REST utilizzando CloudFormation

Creazione di un pool di utenti di Amazon Cognito per un'API REST

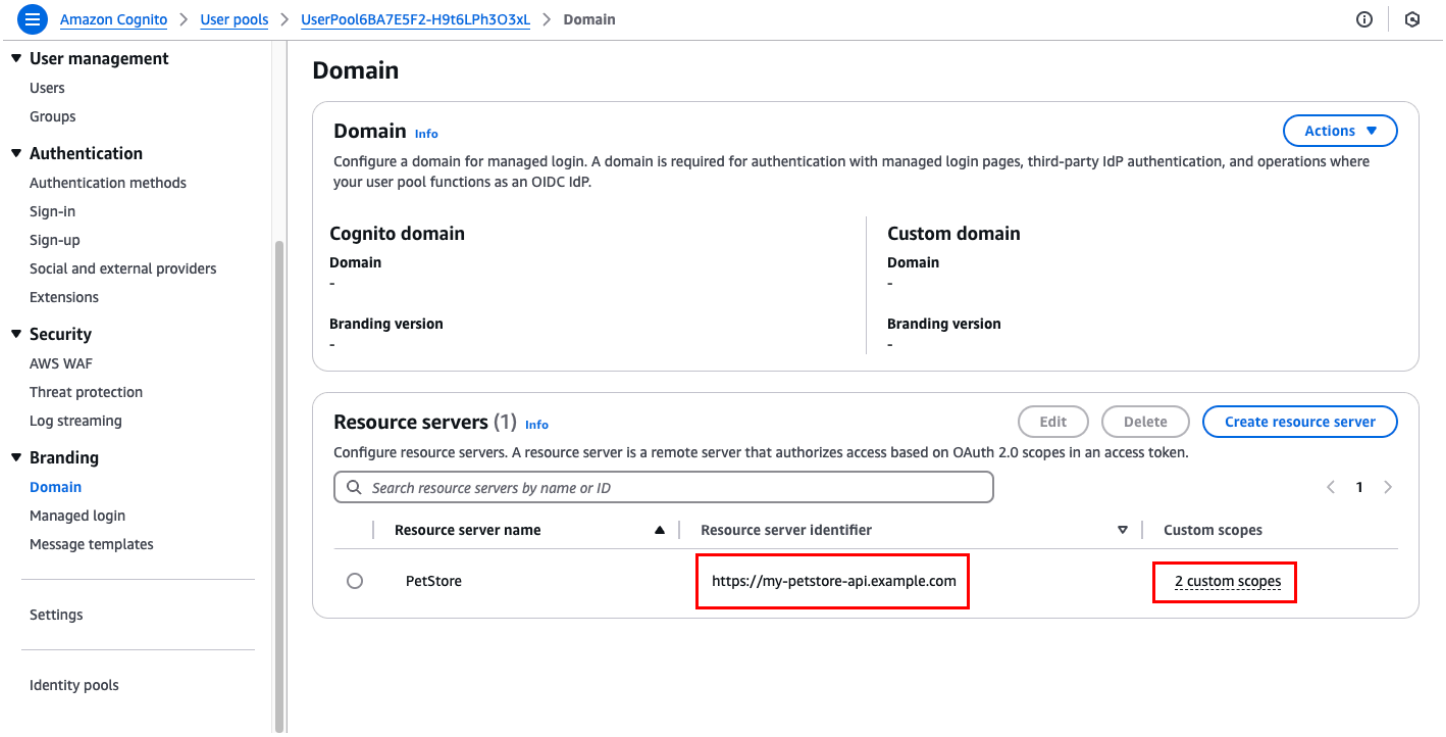
Prima di integrare l'API con un pool di utenti, devi creare il pool di utenti in Amazon Cognito. La configurazione del pool di utenti deve rispettare tutte le quote di risorse per Amazon Cognito. Tutte le variabili Amazon Cognito definite dall'utente, come gruppi, utenti e ruoli, devono utilizzare solo caratteri alfanumerici. Per istruzioni su come creare un pool di utenti, consulta Tutorial: Creazione di un bacino d'utenza nella Guida per gli sviluppatori di Amazon Cognito.

Prendi nota dell'ID pool di utenti, dell'ID client e di qualsiasi segreto client. Il client dovrà fornirli ad Amazon Cognito perché l'utente possa registrarsi con il pool di utenti, accedere al pool di utenti e ottenere un token di identità o di accesso da includere nelle richieste per chiamare metodi API

configurati con il pool di utenti. Inoltre, devi specificare il nome del pool di utenti quando configuri il pool di utenti come autorizzazioni in API Gateway, come descritto nella prossima sezione.

Se stai usando token di accesso per autorizzare le chiamate di metodi API, assicurati di configurare l'integrazione dell'app con il pool di utenti per configurare gli ambiti personalizzati che desideri in un determinato server di risorse. Per ulteriori informazioni sull'utilizzo di token con pool di utenti di Amazon Cognito, consulta Utilizzo di token con pool di utenti. Per ulteriori informazioni sui server di risorse, consulta Definizione dei server di risorse per il pool di utenti.

Annota gli identificatori dei server di risorse configurati e i nomi degli ambiti personalizzati. Ti serviranno per creare i nomi completi degli ambiti di accesso per l'opzione OAuth Scopes (Ambiti OAuth), che viene utilizzata dal provider di autorizzazioni `COGNITO_USER_POOLS`.



Integrazione di un'API REST con un pool di utenti di Amazon Cognito

Dopo aver creato un pool di utenti di Amazon Cognito in API Gateway, devi creare un'autorizzazione `COGNITO_USER_POOLS` che usa il pool di utenti. La procedura seguente illustra come eseguire questa operazione tramite la console API Gateway.

> **Note**
>
> Puoi utilizzare l'operazione `CreateAuthorizer` per creare un'autorizzazione `COGNITO_USER_POOLS` che utilizzi più pool di utenti. Puoi utilizzare fino a 1.000 pool di

utenti per un unica autorizzazione `COGNITO_USER_POOLS`. Questo limite non può essere aumentato.

Important

Dopo aver eseguito una delle procedure di seguito, è necessario distribuire o ridistribuire l'API per la propagazione delle modifiche. Per ulteriori informazioni sulla distribuzione della tua API, vedi Implementazione di REST API in Gateway API.

Per creare un autorizzatore **`COGNITO_USER_POOLS`** utilizzando la console API Gateway

1. Crea una nuova API oppure selezionane una esistente in API Gateway.
2. Nel riquadro di navigazione principale, scegli Autorizzazioni.
3. Scegli Crea autorizzazioni.
4. Per configurare la nuova autorizzazione per usare un pool di utenti, esegui queste operazioni:
   a. In Nome del provider di autorizzazioni, immetti un nome.
   b. In Tipo di autorizzazione, seleziona Cognito.
   c. Per il pool di utenti Cognito, scegli Regione AWS dove hai creato Amazon Cognito e seleziona un pool di utenti disponibile.

      Per definire il pool di utenti è possibile usare una variabile di fase. Utilizza il seguente formato per il pool di utenti: `arn:aws:cognito-idp:`*`us-east-2`*`:`*`111122223333`*`:userpool/${stageVariables.`*`MyUserPool`*`}`.
   d. In Origine token, immetti **`Authorization`** come nome di intestazione da passare al token di identità o di accesso restituito da Amazon Cognito quando un utente accede correttamente.
   e. (Facoltativo) Immetti un'espressione regolare nel campo Convalida del token per convalidare il campo `aud` (Audience, Destinatari) del token di identità prima che la richiesta venga autorizzata con Amazon Cognito. Si noti che quando si utilizza un token di accesso questa convalida rifiuta la richiesta poiché il token di accesso non contiene il campo `aud`.
   f. Scegli Crea autorizzazioni.

5. Dopo aver creato il sistema di autorizzazione `COGNITO_USER_POOLS`, è possibile testarne l'invocazione specificando un token di identità allocato dal pool di utenti. Non è possibile utilizzare un token di accesso per testare l'invocazione del sistema di autorizzazione.

   Puoi ottenere questo token di identità chiamando l'SDK Amazon Cognito Identity per eseguire l'accesso dell'utente. È anche possibile usare l'operazione `InitiateAuth`. Se in Ambiti di autorizzazione non configuri alcun valore, Gateway API considera il token fornito come un token di identità.

La procedura precedente crea un'autorizzazione `COGNITO_USER_POOLS` che utilizza il nuovo pool di utenti di Amazon Cognito appena creato. A seconda del modo in cui abiliti l'autorizzazione per un metodo API, puoi usare un token di identità o un token di accesso assegnato dal pool di utenti integrato.

Per configurare un autorizzatore **`COGNITO_USER_POOLS`** sui metodi

1. Scegliere Resources (Risorse). Seleziona un nuovo metodo o scegline uno esistente. Se necessario, crea una risorsa.
2. Nella scheda Richiesta metodo, in Impostazioni richiesta metodo, scegli Modifica.
3. In Autorizzazioni, nel menu a discesa, seleziona Autorizzazioni del gruppo di utenti di Cognito.
4. Per usare un token di identità, esegui queste operazioni:

   a. Non specificare alcun valore in Ambiti di autorizzazione.

   b. Se necessario, in Richiesta di integrazione, aggiungi le espressioni `$context.authorizer.claims['`*`property-name`*`']` o `$context.authorizer.claims.`*`property-name`* in un modello di mappatura del corpo per passare la proprietà delle richieste di identità specificata dal pool di utenti al back-end. Per i nomi di proprietà semplici, come `sub` o `custom-sub`, le due notazioni sono identiche. Per i nomi di proprietà complessi, come `custom:role`, non puoi usare la notazione punto. Ad esempio, le espressioni di mappatura seguenti passano i campi standard `sub` e `email` della richiesta al back-end:

```
{
 "context" : {
  "sub" : "$context.authorizer.claims.sub",
  "email" : "$context.authorizer.claims.email"
 }
```

```
}
```

Se hai dichiarato un campo di richiesta personalizzato quando hai configurato un pool di utenti, puoi seguire lo stesso modello per accedere ai campi personalizzati. L'esempio seguente ottiene un campo `role` personalizzato di una richiesta:

```
{
 "context" : {
  "role" : "$context.authorizer.claims.role"
    }
}
```

Se il campo personalizzato della richiesta viene dichiarato come `custom:role`, usa l'esempio seguente per ottenere le proprietà della richiesta:

```
{
 "context" : {
  "role" : "$context.authorizer.claims['custom:role']"
    }
}
```

5. Per usare un token di accesso, esegui queste operazioni:

   a. In Ambiti di autorizzazione, immetti uno o più nomi completi di un ambito configurato quando è stato creato il pool di utenti di Amazon Cognito. Ad esempio, seguendo l'esempio fornito in Creazione di un pool di utenti di Amazon Cognito per un'API REST, uno degli ambiti è `https://my-petstore-api.example.com/cats.read`.

      In fase di runtime, la chiamata del metodo riesce se qualsiasi ambito specificato nel metodo in questa fase corrisponde a un ambito richiesto nel token in ingresso. Altrimenti, la chiamata non riesce e restituisce una risposta `401 Unauthorized`.

   b. Scegli Save (Salva).

6. Ripeti queste fasi per gli altri metodi scelti.

Se con l'autorizzazione `COGNITO_USER_POOLS` l'opzione OAuth Scopes (Ambiti OAuth) non è specificata, API Gateway considera il token specificato un token di identità e verifica l'identità richiesta rispetto a quella del pool di utenti. Altrimenti, API Gateway considera il token specificato

un token di accesso e verifica gli ambiti di accesso richiesti nel token rispetto agli ambiti di autorizzazione dichiarati nel metodo.

Invece di usare la console API Gateway, puoi anche abilitare un pool di utenti di Amazon Cognito in un metodo specificando un file di definizione OpenAPI e importando la definizione API in API Gateway.

Per importare un'autorizzazione COGNITO_USER_POOLS con un file di definizione OpenAPI

1. Crea (o esporta) un file di definizione OpenAPI per l'API.
2. Specificare la definizione JSON dell'autorizzazione `COGNITO_USER_POOLS` (`MyUserPool`) come parte della sezione `securitySchemes` OpenAPI 3.0 o della sezione `securityDefinitions` in OpenAPI 2.0, nel seguente modo as follows:

   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. Per usare il token di identità per l'autorizzazione del metodo, aggiungi `{ "MyUserPool": [] }` alla definizione `security` del metodo, come mostrato nel metodo GET seguente nella risorsa 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. Per usare il token di accesso per l'autorizzazione del metodo, modifica la definizione di sicurezza precedente in `{ "MyUserPool": [resource-server/scope, ...] }`:

```
 "paths": {
   "/": {
     "get": {
       "consumes": [
         "application/json"
       ],
       "produces": [
         "text/html"
       ],
       "responses": {
         "200": {
           "description": "200 response",
           "headers": {
             "Content-Type": {
               "type": "string"
             }
           }
         }
       },
       "security": [
         {
           "MyUserPool": ["https://my-petstore-api.example.com/cats.read",
"http://my.resource.com/file.read"]
         }
       ],
       "x-amazon-apigateway-integration": {
         "type": "mock",
         "responses": {
           "default": {
             "statusCode": "200",
             "responseParameters": {
               "method.response.header.Content-Type": "'text/html'"
```

```
            },
          }
        },
        "requestTemplates": {
          "application/json": "{\"statusCode\": 200}"
        },
        "passthroughBehavior": "when_no_match"
      }
    },
    ...
}
```

5. Se necessario, è possibile definire altre impostazioni di configurazione dell'API utilizzando le definizioni o estensioni OpenAPI appropriate. Per ulteriori informazioni, consulta Estensioni OpenAPI per Gateway API.

Chiamata di un'API REST integrata con un pool di utenti di Amazon Cognito

Per chiamare un metodo con un'autorizzazione del pool di utenti configurata, il client deve eseguire queste operazioni:

- Permettere all'utente di iscriversi al pool di utenti.
- Permettere all'utente di accedere al pool di utenti.
- Ottenere un token di identità o di accesso per l'utente che ha effettuato l'accesso dal pool di utenti.
- Includere il token nell'intestazione `Authorization` (o in un'altra intestazione specificata durante la creazione dell'autorizzazione).

È possibile utilizzare AWS Amplify per eseguire queste attività. Per maggiori informazioni, consulta Integrazione di Amazon Cognito con le app Web e per dispositivi mobili.

- Per Android, consulta Nozioni di base su Amplify per Android.
- Per usare iOS, consulta Nozioni di base su Amplify per iOS.
- Per utilizzarlo JavaScript, consulta Getting Started with Amplify for Javascript.

Configurazione dell'autorizzazione Amazon Cognito tra account per un'API REST tramite la console API Gateway

Ora puoi anche utilizzare un pool di utenti Amazon Cognito di un altro AWS account come autorizzatore API. Il pool di utenti di Amazon Cognito può utilizzare strategie di autenticazione con token di connessione come OAuth o SAML. In questo modo, è più semplice gestire e condividere a livello centralizzato l'autorizzazione di un pool di utenti di Amazon Cognito centrale tra più API di API Gateway.

In questa sezione, viene illustrato come configurare un pool di utenti di Amazon Cognito tra account tramite la console Amazon API Gateway.

Queste istruzioni presuppongono che tu abbia già un'API API Gateway in un AWS account e un pool di utenti Amazon Cognito in un altro account.

Creazione di un sistema di autorizzazione di Amazon Cognito multi-account per una REST API

Accedi alla console Gateway Amazon API con l'account contenente la tua API e quindi procedi come descritto di seguito:

1. Crea una nuova API oppure selezionane una esistente in API Gateway.
2. Nel riquadro di navigazione principale, scegli Autorizzazioni.
3. Scegli Crea autorizzazioni.
4. Per configurare la nuova autorizzazione per usare un pool di utenti, esegui queste operazioni:

    a. In Nome del provider di autorizzazioni, immetti un nome.

    b. In Tipo di autorizzazione, seleziona Cognito.

    c. Per Gruppo di utenti di Cognito, copia e incolla l'ARN completo del pool di utenti nel secondo account.

    > **Note**
    >
    > Nella console Amazon Cognito, puoi trovare l'ARN del pool di utenti nel campo Pool ARN (ARN pool) del riquadro General Settings (Impostazioni generali).

    d. In Origine token, immetti **`Authorization`** come nome di intestazione da passare al token di identità o di accesso restituito da Amazon Cognito quando un utente accede correttamente.

e. (Facoltativo) Immetti un'espressione regolare nel campo Convalida del token per convalidare il campo `aud` (Audience, Destinatari) del token di identità prima che la richiesta venga autorizzata con Amazon Cognito. Si noti che quando si utilizza un token di accesso questa convalida rifiuta la richiesta poiché il token di accesso non contiene il campo `aud`.

f. Scegli Crea autorizzazioni.

Crea un autorizzatore Amazon Cognito per un'API REST utilizzando CloudFormation

Puoi utilizzarlo CloudFormation per creare un pool di utenti Amazon Cognito e un autorizzatore Amazon Cognito. Il CloudFormation modello di esempio esegue le seguenti operazioni:

- Crea un pool di utenti di Amazon Cognito. Il client deve prima far accedere l'utente al pool di utenti e ottenere un'identità o un token di accesso. Se stai usando token di accesso per autorizzare le chiamate di metodi API, assicurati di configurare l'integrazione dell'app con il pool di utenti per configurare gli ambiti personalizzati che desideri in un determinato server di risorse.
- Crea un'API Gateway API con un metodo `GET`.
- Crea un'autorizzazione Amazon Cognito che utilizza l'intestazione `Authorization` come origine del token.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  UserPool:
    Type: AWS::Cognito::UserPool
    Properties:
      AccountRecoverySetting:
        RecoveryMechanisms:
          - Name: verified_phone_number
            Priority: 1
          - Name: verified_email
            Priority: 2
      AdminCreateUserConfig:
        AllowAdminCreateUserOnly: true
      EmailVerificationMessage: The verification code to your new account is {####}
      EmailVerificationSubject: Verify your new account
      SmsVerificationMessage: The verification code to your new account is {####}
      VerificationMessageTemplate:
        DefaultEmailOption: CONFIRM_WITH_CODE
        EmailMessage: The verification code to your new account is {####}
        EmailSubject: Verify your new account
```

```
        SmsMessage: The verification code to your new account is {####}
    UpdateReplacePolicy: Retain
    DeletionPolicy: Retain
  CogAuthorizer:
    Type: AWS::ApiGateway::Authorizer
    Properties:
      Name: CognitoAuthorizer
      RestApiId:
        Ref: Api
      Type: COGNITO_USER_POOLS
      IdentitySource: method.request.header.Authorization
      ProviderARNs:
        - Fn::GetAtt:
            - UserPool
            - Arn
  Api:
    Type: AWS::ApiGateway::RestApi
    Properties:
      Name: MyCogAuthApi
  ApiDeployment:
    Type: AWS::ApiGateway::Deployment
    Properties:
      RestApiId:
        Ref: Api
    DependsOn:
      - CogAuthorizer
      - ApiGET
  ApiDeploymentStageprod:
    Type: AWS::ApiGateway::Stage
    Properties:
      RestApiId:
        Ref: Api
      DeploymentId:
        Ref: ApiDeployment
      StageName: prod
  ApiGET:
    Type: AWS::ApiGateway::Method
    Properties:
      HttpMethod: GET
      ResourceId:
        Fn::GetAtt:
          - Api
          - RootResourceId
      RestApiId:
```

```
        Ref: Api
      AuthorizationType: COGNITO_USER_POOLS
      AuthorizerId:
        Ref: CogAuthorizer
      Integration:
        IntegrationHttpMethod: GET
        Type: HTTP_PROXY
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets
Outputs:
  ApiEndpoint:
    Value:
      Fn::Join:
        - ""
        - - https://
          - Ref: Api
          - .execute-api.
          - Ref: AWS::Region
          - "."
          - Ref: AWS::URLSuffix
          - /
          - Ref: ApiDeploymentStageprod
          - /
```

Integrazioni per REST APIs in API Gateway

Dopo aver impostato un metodo API, devi integrarlo con un endpoint nel back-end. Un endpoint di backend viene anche definito endpoint di integrazione e può essere una funzione Lambda, una pagina Web HTTP o un'azione di servizio. AWS

Come nel caso del metodo API, l'integrazione dell'API presenta una richiesta e una risposta di integrazione. Una richiesta di integrazione comprende una richiesta HTTP ricevuta dal back-end. Potrebbe o non potrebbe essere diversa dalla richiesta di metodo inviata dal cliente. Una risposta di integrazione è una risposta HTTP contenente l'output restituito dal back-end.

La configurazione di una richiesta di integrazione comporta le seguenti operazioni: configurare come passare le richieste di metodo inviate dal client al back-end; configurare come trasformare i dati della richiesta, se necessario, in dati della richiesta di integrazione; specificare quale funzione Lambda chiamare; specificare a quale server HTTP inoltrare la richiesta in arrivo o specificare l'azione del servizio AWS da chiamare.

La configurazione di una risposta di integrazione (applicabile solo alle integrazioni non proxy) implica le seguenti operazioni: come passare il risultato restituito dal back-end alla risposta di un metodo di

un dato codice di stato, definire come trasformare parametri di risposta di integrazione specifici in base a parametri di risposta dei metodi pre-configurati e configurare come mappare il corpo della risposta di integrazione al corpo della risposta del metodo in base a modelli specifici di mappatura del corpo.

A livello di programmazione, una richiesta di integrazione viene incapsulata dalla risorsa `Integration`, mentre la risposta di integrazione dalla risorsa `IntegrationResponse` di API Gateway.

Per configurare una richiesta di integrazione, è necessario creare una risorsa `Integration` e usarla per configurare l'URL dell'endpoint di integrazione. Successivamente, devi impostare le autorizzazioni IAM per accedere al back-end e specificare le mappature per trasformare i dati di richiesta in entrata prima di passarli al back-end. Per configurare una risposta di integrazione per un'integrazione non proxy, è necessario creare una risorsa `IntegrationResponse` e usarla per configurare la relativa risposta del metodo target. Successivamente, devi configurare la mappatura dell'output di back-end alla risposta del metodo.

Argomenti

- Configurazione di una richiesta di integrazione in API Gateway
- Configurazione di una risposta di integrazione in API Gateway
- Integrazioni Lambda per REST API in Gateway API
- Integrazioni HTTP per REST APIs in API Gateway
- Trasmetti in streaming la risposta di integrazione per le integrazioni proxy in API Gateway
- Integrazioni private per REST APIs in API Gateway
- Integrazioni fittizie per REST APIs in API Gateway

Configurazione di una richiesta di integrazione in API Gateway

Per configurare una richiesta di integrazione, è necessario completare i seguenti task obbligatori e opzionali:

1. Seleziona un tipo di integrazione che stabilisca in che modo i dati di richiesta del metodo vengano passati al back-end.
2. Per le integrazioni non fittizie, specifica un metodo HTTP e l'URI dell'endpoint di integrazione target, eccetto per l'integrazione `MOCK`.

3. Per le integrazioni con le funzioni Lambda e AWS altre azioni di servizio, imposta un ruolo IAM con le autorizzazioni necessarie affinché API Gateway chiami il backend per tuo conto.
4. Per integrazioni non-proxy, imposta le mappature dei parametri necessari per mappare i parametri delle richieste di metodo predefinite ai parametri di richiesta di integrazione più appropriati.
5. Per integrazioni non-proxy, imposta le mappature dei corpi necessari per mappare il corpo delle richieste di metodo in entrata di un tipo di contenuto specifico in base al modello di mappatura designato.
6. Per le integrazioni non-proxy, specifica la condizione in base alla quale i dati della richiesta di metodo in entrata vengano passati attraverso il back-end inalterati.
7. A scelta, specifica come gestire una conversione dei tipi per un payload binario.
8. A scelta, dichiara il nome del namespace cache e i parametri della chiave cache per abilitare il caching delle API.

L'esecuzione di queste attività implica la creazione di una risorsa Integration di API Gateway e la configurazione di valori di proprietà adeguati. Puoi farlo utilizzando la console API Gateway, AWS CLI i comandi, un AWS SDK o l'API REST di API Gateway.

Argomenti

- Task di base di una richiesta di integrazione API
- Scegliere un tipo di integrazione API Gateway API
- Configurazione di un'integrazione proxy mediante una risorsa proxy
- Configurazione di una richiesta di integrazione API tramite la console API Gateway

Task di base di una richiesta di integrazione API

Una richiesta di integrazione è una richiesta HTTP che API Gateway invia al back-end, passando i dati della richiesta inviata dal client e trasformandoli, se necessario. Il metodo HTTP (o verbo) e l'URI della richiesta di integrazione sono determinati dal back-end (l'endpoint dell'integrazione). Possono essere diversi o uguali, rispettivamente, al metodo HTTP e all'URI della richiesta di metodo.

Ad esempio, quando una funzione Lambda restituisce un file proveniente da Amazon S3, puoi esporre intuitivamente questa operazione al client come una richiesta del metodo `GET`, anche se la corrispondente richiesta di integrazione richiede l'uso di una richiesta `POST` per chiamare la funzione Lambda. Per un endpoint HTTP, è probabile che la richiesta di metodo e la richiesta di integrazione

corrispondente utilizzino entrambi lo stesso verbo HTTP. Tuttavia, non è obbligatorio. Puoi integrare la seguente richiesta di metodo:

```
GET /{var}?query=value
Host: api.domain.net
```

Con la seguente richiesta di integrazione:

```
POST /
Host: service.domain.com
Content-Type: application/json
Content-Length: ...

{
   path: "{var}'s value",
   type: "value"
}
```

Come sviluppatore di API, puoi utilizzare qualunque verbo HTTP e URI per una richiesta di metodo che soddisfino i tuoi requisiti. Tuttavia, devi soddisfare i requisiti dell'endpoint di integrazione. Quando i dati della richiesta di metodo sono diversi dai dati della richiesta di integrazione, puoi eliminare la differenza tramite mappature dei dati della richiesta di metodo ai dati della richiesta di integrazione.

Negli esempi precedenti, la mappatura traduce i valori della variabile di percorso (`{var}`) e dei parametri di query (`query`) della richiesta di metodo `GET` nei valori delle proprietà del payload della richiesta di integrazione di `path` e `type`. Tra gli altri dati di richiesta mappabili figurano il corpo e le intestazioni della richiesta. Tali dati sono descritti in the section called “Mappatura dei parametri”.

Quando configuri la richiesta HTTP o di integrazione proxy HTTP, assegni l'URL dell'endpoint HTTP del back-end come valore URI della richiesta di integrazione. Ad esempio, nell' PetStoreAPI, la richiesta del metodo per ottenere una pagina di animali domestici ha il seguente URI di richiesta di integrazione:

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

Quando configuri Lambda o l'integrazione proxy Lambda, assegni l'Amazon Resource Name (ARN) per chiamare la funzione Lambda come valore URI della richiesta di integrazione. Questo ARN ha il seguente formato:

```
arn:aws:apigateway:api-region:lambda:path//2015-03-31/functions/arn:aws:lambda:lambda-region:account-id:function:lambda-function-name/invocations
```

La parte dopo `arn:aws:apigateway:`*`api-region`*`:lambda:path/`, ovvero `/2015-03-31/functions/arn:aws:lambda:`*`lambda-region`*`:`*`account-id`*`:function:`*`lambda-function-name`*`/invocations`, è il percorso URI dell'API REST dell'azione Lambda Invoke. Se utilizzi la console API Gateway per configurare l'integrazione Lambda, API Gateway crea l'ARN e lo assegna all'URI di integrazione dopo avere chiesto di selezionare il *`lambda-function-name`* da una regione.

Quando si imposta la richiesta di integrazione con un'altra azione di AWS servizio, anche l'URI della richiesta di integrazione è un ARN, simile all'integrazione con l'azione Lambda. `Invoke` Ad esempio, per l'integrazione con l'GetBucketazione di Amazon S3, l'URI della richiesta di integrazione è un ARN del seguente formato:

```
arn:aws:apigateway:api-region:s3:path/{bucket}
```

L'URI della richiesta di integrazione è la convenzione del percorso che specifica l'operazione, dove *`{bucket}`* è il segnaposto del nome di un bucket. In alternativa, è possibile fare riferimento a un'azione di AWS servizio tramite il relativo nome. Utilizzando il nome dell'operazione, l'URI della richiesta di integrazione per l'operazione `GetBucket` di Amazon S3; diventa il seguente:

```
arn:aws:apigateway:api-region:s3:action/GetBucket
```

Con l'URI della richiesta di integrazione basata sull'operazione, il nome del bucket (*`{bucket}`*) deve essere specificato nel corpo della richiesta di integrazione (`{ Bucket: "`*`{bucket}`*`" }`), seguito dal formato di input dell'operazione `GetBucket`.

Per AWS le integrazioni, devi anche configurare le credenziali per consentire ad API Gateway di richiamare le azioni integrate. Puoi creare un ruolo IAM o sceglierne uno esistente per consentire ad API Gateway di chiamare l'operazione, quindi specificare il ruolo utilizzando il proprio ARN. Di seguito è illustrato un esempio di questo ARN:

```
arn:aws:iam::account-id:role/iam-role-name
```

Tale ruolo IAM deve contenere una policy che consenta di eseguire l'operazione. API Gateway deve inoltre essere dichiarata (nella relazione di fiducia del ruolo) come entità affidabile per l'assunzione del ruolo. Tali autorizzazioni possono essere assegnate per l'operazione stessa. Si tratta di

autorizzazioni basate sulle risorse. Per l'integrazione Lambda, puoi chiamare l'operazione Lambda addPermission per impostare le autorizzazioni basate sulle risorse, quindi configurare `credentials` su null nella richiesta di integrazione di API Gateway.

Abbiamo illustrato la configurazione di integrazione di base. Le impostazioni avanzate prevedono la mappatura dei dati della richiesta di metodo ai dati della richiesta di integrazione. Per ulteriori informazioni, consulta the section called “Trasformazioni dei dati”.

Scegliere un tipo di integrazione API Gateway API

Seleziona un tipo di integrazione API in base ai tipi di endpoint di integrazione con cui lavori e alla quantità di dati che transitano nell'endpoint di integrazione. Per una funzione Lambda, puoi avere l'integrazione proxy di Lambda o l'integrazione personalizzata di Lambda. Per un endpoint HTTP, puoi avere l'integrazione proxy di HTTP o l'integrazione personalizzata di HTTP. Per un'azione AWS di servizio, è disponibile l' AWS integrazione solo del tipo non proxy. API Gateway supporta anche l'integrazione fittizia, in cui API Gateway serve come endpoint di integrazione per rispondere a una richiesta del metodo.

L'integrazione personalizzata Lambda è un caso speciale di integrazione, in cui AWS l'endpoint di integrazione corrisponde all'azione di richiamo della funzione del servizio Lambda.

A livello di programmazione, è possibile scegliere un tipo di integrazione configurando la proprietà `type` della risorsa `Integration`. Per l'integrazione proxy Lambda, il valore è `AWS_PROXY`. Per l'integrazione personalizzata Lambda e per tutte le altre integrazioni AWS , è `AWS`. Per l'integrazione proxy HTTP e l'integrazione HTTP, il valore è, rispettivamente, `HTTP_PROXY` e `HTTP`. Per l'integrazione fittizia, il valore `type` è `MOCK`.

L'integrazione proxy Lambda supporta una configurazione di integrazione semplificata con una singola funzione Lambda. La configurazione è semplice e può evolvere con il back-end senza annullare le impostazioni esistenti. Per questi motivi, è altamente consigliata per l'integrazione con una funzione Lambda.

Al contrario, l'integrazione personalizzata Lambda consente il riutilizzo di modelli di mappatura configurati per vari endpoint di integrazione che presentano requisiti simili in termini di formato di dati di input e di output. La configurazione è più complessa ed è consigliata per scenari applicativi più avanzati.

Analogamente, l'integrazione proxy HTTP ha una configurazione di integrazione più semplice e può evolvere con il back-end senza annullare le impostazioni esistenti. L'integrazione personalizzata

HTTP è più complessa da configurare, ma consente il riutilizzo di modelli di mappatura configurati per altri endpoint di integrazione.

L'elenco che segue riepiloga i tipi di integrazione supportati:

- `AWS`: questo tipo di integrazione consente a un'API di esporre le operazioni dei servizi AWS . Nell'integrazione `AWS`, devi configurare la richiesta e la risposta di integrazione, oltre a impostare le mappature di dati necessarie dalla richiesta di metodo alla richiesta di integrazione e dalla risposta di integrazione alla risposta di metodo.
- `AWS_PROXY`: questo tipo di integrazione consente l'integrazione di un metodo API con un'operazione di chiamata della funzione Lambda tramite una configurazione di integrazione semplificata, flessibile e versatile. Questa integrazione si basa su interazioni dirette tra il client e la funzione Lambda integrata.

  Con questo tipo di integrazione, noto anche come integrazione proxy Lambda, non devi configurare la richiesta o la risposta di integrazione. API Gateway trasferisce la richiesta in entrata dal client come input alla funzione Lambda back-end. La funzione Lambda integrata prende l'input di questo formato e analizza l'input di tutte le fonti disponibili, incluse le intestazioni delle richieste, le variabili di percorso dell'URL, i parametri delle stringhe di query e il corpo applicabile. La funzione restituisce il risultato seguendo questo formato di output.

  Questo è il tipo di integrazione preferito per chiamare una funzione Lambda tramite API Gateway e non è applicabile ad altre azioni di AWS servizio, incluse le azioni Lambda diverse dall'azione di richiamo della funzione.
- `HTTP`: questo tipo di integrazione consente a un'API di esporre gli endpoint HTTP nel back-end. Con l'integrazione `HTTP`, nota anche come integrazione HTTP personalizzata, devi configurare l richiesta e la risposta di integrazione. Devi configurare le mappature di dati necessarie dalla richiesta di metodo alla richiesta di integrazione e dalla risposta di integrazione alla risposta di metodo.
- `HTTP_PROXY`: l'integrazione proxy HTTP consente a un cliente di accedere agli endpoint HTTP di back-end con una configurazione di integrazione semplificata e un singolo metodo API. Non devi configurare la richiesta o la risposta di integrazione. API Gateway trasferisce la richiesta in entrata dal client all'endpoint HTTP e la risposta in uscita dall'endpoint HTTP al client.
- `MOCK`: questo tipo di integrazione consente ad API Gateway di restituire una risposta senza inviare la richiesta fino al back-end. Si tratta di un'opzione utile per il test di API, poiché può essere utilizzata per testare la configurazione di integrazione senza sostenere costi per l'utilizzo del back-end e per favorire uno sviluppo collaborativo di un'API.

In uno sviluppo collaborativo, un team può isolare la propria attività di sviluppo impostando simulazioni di componenti API di proprietà di altri team utilizzando le integrazioni `MOCK`. Questa opzione viene anche utilizzata per restituire intestazioni relative alla configurazione CORS per verificare che il metodo API ne consenta l'accesso. Di fatto, la console API Gateway integra il metodo `OPTIONS` per supportare la configurazione CORS con un'integrazione fittizia. Le risposte del gateway sono altri esempi di integrazioni fittizie.

Configurazione di un'integrazione proxy mediante una risorsa proxy

Per configurare un'integrazione proxy in un'API di API Gateway mediante una risorsa proxy, completa le attività seguenti:

- Crea una risorsa proxy con una variabile di percorso greedy `{proxy+}`.
- Imposta il metodo `ANY` sulla risorsa proxy.
- Integra la risorsa e il metodo con un back-end utilizzando il tipo di integrazione HTTP o Lambda.

> **Note**
>
> Anche se normalmente vengono utilizzati insieme, i metodi `ANY`, i tipi di integrazione proxy e le variabili di percorso greedy sono caratteristiche indipendenti. È possibile configurare un metodo HTTP specifico per una risorsa greedy o applicare tipi di integrazione non proxy a una risorsa proxy.

In API Gateway la gestione dei metodi prevede restrizioni e limitazioni specifiche per entrambi tipi di integrazione proxy, Lambda e HTTP. Per informazioni dettagliate, vedi the section called “Note importanti”.

> **Note**
>
> Quando si utilizza l'integrazione proxy con un passthrough, API Gateway restituisce l'intestazione `Content-Type:application/json` di default, se non è specificato il tipo di contenuto di un payload.

Una risorsa proxy è più efficace se integrata con un back-end tramite l'integrazione proxy HTTP o l'integrazione proxy Lambda.

Integrazione proxy HTTP mediante una risorsa proxy

L'integrazione proxy HTTP, designata da `HTTP_PROXY` nell'API REST API Gateway, consente di integrare una richiesta del metodo con un endpoint HTTP di back-end. Con questo tipo di integrazione, API Gateway si limita a trasferire richiesta e risposta complete tra il front-end e il back-end, seguendo restrizioni e limitazioni specifiche.

> **Note**
>
> L'integrazione proxy HTTP supporta le intestazioni multi-valore e le stringhe di query.

Quando si applica l'integrazione proxy HTTP a una risorsa proxy, è possibile configurare l'API in modo che esponga una parte di un'intera gerarchia di endpoint del back-end HTTP con un'unica configurazione di integrazione. Supponi ad esempio che il back-end del sito Web sia organizzato in più rami dei nodi di struttura del nodo radice (`/site`) come: `/site/`a_0`/`a_1`/.../`a_N, `/site/` b_0`/`b_1`/.../`b_M e così via. Se integri il metodo `ANY` di una risorsa proxy `/api/{proxy+}` con gli endpoint del back-end con percorsi URL `/site/{proxy}`, una singola richiesta di integrazione può supportare qualsiasi operazione HTTP (GET, POST e così via) ovunque in `[`a_0`, `a_1`, ..., `a_N`, `b_0`, `b_1`, ...`b_M`, ...]`. Se invece applichi un'integrazione proxy a un metodo HTTP specifico, ad esempio `GET`, la richiesta di integrazione risultante supporta le operazioni specificate (`GET`) su qualsiasi nodo di back-end.

Integrazione proxy Lambda mediante una risorsa proxy

L'integrazione proxy Lambda, designata da `AWS_PROXY` nell'API REST API Gateway, consente di integrare una richiesta del metodo con una funzione Lambda nel back-end. Con questo tipo di integrazione, API Gateway applica un modello di mappatura predefinito per inviare l'intera richiesta alla funzione Lambda e trasforma l'output di tale funzione in risposte HTTP.

Analogamente, puoi applicare l'integrazione proxy Lambda a una risorsa proxy di `/api/{proxy+}` per configurare una sola integrazione e fare in modo che una funzione Lambda back-end reagisca individualmente alle modifiche apportate in una delle risorse API in `/api`.

Configurazione di una richiesta di integrazione API tramite la console API Gateway

La configurazione di un metodo API definisce il metodo e descrive i suoi comportamenti. Per configurare un metodo, è necessario specificare una risorsa su cui il metodo è esposto, inclusa la root ("/"), un metodo HTTP (`GET`, `POST` e così via) e il modo in cui sarà integrato con il relativo back-end. La richiesta e la risposta di metodo specificano il contratto con l'app di chiamata e stipulano i parametri che l'API può ricevere e l'aspetto della risposta.

Le seguenti procedure descrivono come utilizzare la console Gateway API per creare una richiesta di integrazione.

Argomenti

- Configurazione di un'integrazione Lambda
- Configurazione di un'integrazione HTTP
- Configura un'integrazione AWS di servizi
- Configurazione di un'integrazione fittizia

Configurazione di un'integrazione Lambda

L'integrazione della funzione Lambda si utilizza per integrare l'API con una funzione Lambda. A livello di API, sarà un tipo di integrazione `AWS` se si crea un'integrazione non proxy o un tipo di integrazione `AWS_PROXY` se si crea un'integrazione proxy.

Per configurare l'integrazione Lambda

1. Nel riquadro Risorse scegli Crea metodo.
2. Per Tipo di metodo seleziona un metodo HTTP.
3. Per Integration type (Tipo di integrazione), scegli Lambda Function (Funzione Lambda).
4. Per utilizzare un'integrazione proxy Lambda, attiva Integrazione proxy Lambda. Per ulteriori informazioni sulle integrazioni proxy Lambda, consulta the section called “ Informazioni sull'integrazione proxy Lambda ”.
5. Per Funzione Lambda immetti il nome della funzione Lambda.

   Se utilizzi una funzione Lambda in una regione diversa da quella dell'API, seleziona la regione dal menu a discesa e inserisci il nome della funzione Lambda. Se usi una funzione Lambda tra diversi account immetti il nome della risorsa Amazon (ARN) della funzione.

6. Per utilizzare il valore di timeout predefinito di 29 secondi, mantieni attiva l'opzione Timeout predefinito. Per impostare un timeout personalizzato, scegli Timeout predefinito e immetti un valore di timeout compreso tra `50` e `29000` millisecondi.
7. (Facoltativo) È possibile configurare le impostazioni di richiesta del metodo utilizzando i seguenti menu a discesa. Scegli Impostazioni della richiesta del metodo e configura la richiesta del metodo. Per ulteriori informazioni, consulta il passaggio 3 di the section called “Modifica di una richiesta del metodo nella console”.

   È possibile configurare le impostazioni della richiesta del metodo anche dopo aver creato il metodo.
8. Scegli Crea metodo.

Configurazione di un'integrazione HTTP

L'integrazione HTTP si utilizza per integrare l'API con un endpoint HTTP. A livello di API, questa è un'integrazione di tipo `HTTP`.

Per configurare un'integrazione HTTP

1. Nel riquadro Risorse scegli Crea metodo.
2. Per Tipo di metodo seleziona un metodo HTTP.
3. Per Tipo di integrazione scegli Fittizio.
4. Per utilizzare un'integrazione proxy HTTP, attiva Integrazione proxy HTTP. Per ulteriori informazioni sulle integrazioni proxy HTTP, consulta the section called “Configurazione di integrazioni proxy HTTP in API Gateway”.
5. Per HTTP method (Metodo HTTP) scegliere il tipo di metodo HTTP che corrisponde maggiormente al metodo nel back-end HTTP.
6. Per URL endpoint immetti l'URL del back-end HTTP che desideri venga utilizzato dal metodo.
7. Per Gestione contenuti seleziona un comportamento di gestione dei contenuti.
8. Per utilizzare il valore di timeout predefinito di 29 secondi, mantieni attiva l'opzione Timeout predefinito. Per impostare un timeout personalizzato, scegli Timeout predefinito e immetti un valore di timeout compreso tra `50` e `29000` millisecondi.
9. (Facoltativo) È possibile configurare le impostazioni di richiesta del metodo utilizzando i seguenti menu a discesa. Scegli Impostazioni della richiesta del metodo e configura la richiesta del metodo. Per ulteriori informazioni, consulta il passaggio 3 di the section called “Modifica di una richiesta del metodo nella console”.

È possibile configurare le impostazioni della richiesta del metodo anche dopo aver creato il metodo.

10. Scegli Crea metodo.

Configura un'integrazione AWS di servizi

Utilizza un'integrazione AWS di servizi per integrare la tua API direttamente con un AWS servizio. A livello di API, questa è un'integrazione di tipo `AWS`.

Per configurare un'API Gateway API procedi come segue:

- Crea una nuova funzione Lambda.
- Imposta l'autorizzazione per le risorse alla funzione Lambda.
- Esegui qualsiasi altra azione del servizio Lambda.

È necessario scegliere Servizio AWS .

Per configurare un'integrazione AWS di servizi

1. Nel riquadro Risorse scegli Crea metodo.
2. Per Tipo di metodo seleziona un metodo HTTP.
3. Per Tipo di integrazione scegli Servizio AWS .
4. Per AWS Regione, scegli la AWS regione che desideri utilizzare questo metodo per avviare l'azione.
5. Per l'AWS assistenza, scegli il AWS servizio che desideri chiamare con questo metodo.
6. Per AWS sottodominio, inserisci il sottodominio utilizzato dal AWS servizio. In genere, questo spazio rimane vuoto. Alcuni servizi AWS supportano i sottodomini come parte degli host. Consulta la documentazione relativa al servizio per la disponibilità e, se disponibili, ulteriori dettagli.
7. In HTTP method (Metodo HTTP) scegliere il tipo di metodo HTTP che corrisponde all'operazione. Per il tipo di metodo HTTP, consulta la documentazione di riferimento dell'API per il AWS servizio che hai scelto per AWS il servizio.
8. Per Tipo di operazione seleziona Usa nome operazione per utilizzare un'operazione API o Utilizza sostituzione percorso per usare un percorso personalizzato della risorsa. Per le azioni

disponibili e i percorsi di risorse personalizzati, consulta la documentazione di riferimento dell'API per il AWS servizio che hai scelto per il AWS servizio.

9. Inserisci il Nome azione o la Sostituzione percorso.
10. In Ruolo di esecuzione immetti il nome della risorsa Amazon (ARN) del ruolo IAM utilizzato dal metodo per chiamare l'operazione.

    Per creare il ruolo IAM puoi adattare le istruzioni disponibili in the section called “Fase 1: creare il ruolo di esecuzione del proxy del AWS servizio”. Specifica una policy di accesso con il numero desiderato di dichiarazioni di risorse e azioni: Per ulteriori informazioni, consulta the section called “Come funziona Amazon API Gateway con IAM”.

    Per la sintassi dell'istruzione relativa alle azioni e alle risorse, consulta la documentazione del AWS servizio che hai scelto per il AWS servizio.

    Per la relazione di fiducia del ruolo IAM, specificare quanto segue, in modo da consentire ad API Gateway di intervenire per conto dell'account AWS :

    JSON

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

11. Per utilizzare il valore di timeout predefinito di 29 secondi, mantieni attiva l'opzione Timeout predefinito. Per impostare un timeout personalizzato, scegli Timeout predefinito e immetti un valore di timeout compreso tra `50` e `29000` millisecondi.
12. (Facoltativo) È possibile configurare le impostazioni di richiesta del metodo utilizzando i seguenti menu a discesa. Scegli Impostazioni della richiesta del metodo e configura la richiesta del

metodo. Per ulteriori informazioni, consulta il passaggio 3 di the section called “Modifica di una richiesta del metodo nella console”.

È possibile configurare le impostazioni della richiesta del metodo anche dopo aver creato il metodo.

13. Scegli Crea metodo.

Configurazione di un'integrazione fittizia

L'integrazione fittizia si utilizza affinché Gateway API agisca come backend e restituisca risposte statiche. A livello di API, questa è un'integrazione di tipo `MOCK`. In genere, puoi utilizzare l'integrazione `MOCK` quando la tua API non è ancora quella definitiva, ma vuoi generare risposte API per liberare i team relativi e dedicarli al test. Per il metodo `OPTION`, API Gateway imposta l'integrazione `MOCK` come predefinita per restituire intestazioni abilitate a CORS per la risorsa API applicata.

Per configurare un'integrazione fittizia

1. Nel riquadro Risorse scegli Crea metodo.
2. Per Tipo di metodo seleziona un metodo HTTP.
3. Per Tipo di integrazione scegli Fittizio.
4. (Facoltativo) È possibile configurare le impostazioni di richiesta del metodo utilizzando i seguenti menu a discesa. Scegli Impostazioni della richiesta del metodo e configura la richiesta del metodo. Per ulteriori informazioni, consulta il passaggio 3 di the section called “Modifica di una richiesta del metodo nella console”.

   È possibile configurare le impostazioni della richiesta del metodo anche dopo aver creato il metodo.
5. Scegli Crea metodo.

Configurazione di una risposta di integrazione in API Gateway

Per un'integrazione non-proxy, devi impostare almeno una risposta di integrazione come predefinita, per trasmettere il risultato restituito dal back-end al client. Puoi scegliere di trasmettere il risultato invariato o di trasformare i dati della risposta di integrazione in dati della risposta di metodo se i formati sono diversi.

Per un'integrazione proxy, API Gateway passa automaticamente l'output del back-end al client come risposta HTTP. Non devi configurare la risposta del metodo o la risposta di integrazione.

Per configurare una risposta di integrazione, è necessario completare le seguenti attività obbligatorie e facoltative:

1. Specifica un codice di stato HTTP di una risposta di metodo su cui sono mappati i dati della risposta di integrazione. Questo dato è obbligatorio.
2. Definisci un'espressione regolare per scegliere che l'output del back-end sia rappresentato da questa risposta di integrazione. Se questo spazio rimane vuoto, la risposta corrisponde a quella predefinita utilizzata per cogliere qualsiasi risposta non ancora configurata.
3. Se necessario, dichiara le mappature che consistono di coppie chiave-valore per mappare parametri di risposta di integrazione specifici su parametri di risposta di metodo.
4. Se necessario, aggiungi modelli di mappatura del corpo per trasformare i payload di risposte di integrazione specifiche in payload di risposta di metodo.
5. Se necessario, specifica come gestire una conversione dei tipi per un payload binario.

Una risposta di integrazione è una risposta HTTP contenente la risposta del back-end. Per un endpoint HTTP, la risposta del back-end è una risposta HTTP. Il codice di stato della risposta di integrazione può riferirsi al codice di stato restituito dal back-end, mentre il corpo della risposta di integrazione coincide con il payload restituito dal back-end. Per un endpoint Lambda, la risposta del back-end è l'output restituito dalla funzione Lambda. Con l'integrazione Lambda, l'output della funzione Lambda viene restituito come risposta `200 OK`. Il payload può contenere il risultato come dati JSON, incluso una stringa o un oggetto JSON, o un messaggio di errore come oggetto JSON. Puoi assegnare un'espressione regolare alla proprietà selectionPattern per mappare una risposta di errore a una risposta di errore HTTP adeguata. Per ulteriori informazioni sulle risposte di errore della funzione Lambda, consulta Gestione degli errori Lambda in API Gateway. Con l'integrazione proxy di Lambda, la funzione Lambda deve restituire l'output nel formato seguente:

```
{
    statusCode: "...",            // a valid HTTP status code
    headers: {
        custom-header: "..."      // any API-specific custom header
    },
    body: "...",                  // a JSON string.
    isBase64Encoded:  true|false  // for binary support
}
```

Non è necessario mappare la risposta della funzione Lambda alla risposta HTTP appropriata.

Per restituire il risultato al client, configura la risposta di integrazione per trasmettere la risposta invariata dell'endpoint alla risposta di metodo corrispondente. Oppure puoi mappare i dati di risposta dell'endpoint ai dati di risposta del metodo. Tra i dati di risposta che possono essere mappati figurano il codice di stato della risposta, i parametri di intestazione della risposta e il corpo della risposta. Se per il codice di stato restituito non è stata definita una risposta del metodo, API Gateway restituisce l'errore 500. Per ulteriori informazioni, consulta Sostituzione dei parametri della richiesta e della risposta e dei codici di stato per REST API in Gateway API.

Integrazioni Lambda per REST API in Gateway API

Puoi integrare un metodo API con una funzione Lambda tramite l'integrazione proxy Lambda o l'integrazione non proxy (personalizzata) Lambda.

Nell'integrazione proxy Lambda la configurazione richiesta è semplice. Impostare il metodo HTTP dell'integrazione su POST, l'URI dell'endpoint di integrazione sull'ARN dell'operazione di chiamata di una specifica funzione Lambda e la credenziale su un ruolo IAM con autorizzazioni per permettere ad API Gateway di chiamare la funzione Lambda per conto tuo.

Nell'integrazione non proxy Lambda, oltre alla procedura di configurazione dell'integrazione proxy, devi specificare anche il modo in cui i dati della richiesta in entrata vengono mappati alla richiesta di integrazione e il modo in cui i dati della risposta di integrazione risultante vengono mappati alla risposta del metodo.

Argomenti

- Integrazioni proxy Lambda in Gateway API
- Configurazione di integrazioni personalizzate Lambda in API Gateway
- Configurazione della chiamata asincrona della funzione Lambda back-end
- Gestione degli errori Lambda in API Gateway

Integrazioni proxy Lambda in Gateway API

La sezione seguente illustra come utilizzare un'integrazione proxy Lambda.

Argomenti

- Informazioni sull'integrazione proxy Lambda di API Gateway

- Supporto per le intestazioni multi-valore e i parametri di stringa di query
- Formato di input di una funzione Lambda per l'integrazione proxy
- Formato di output di una funzione Lambda per l'integrazione proxy
- Configura l'integrazione del proxy Lambda per API Gateway utilizzando AWS CLI
- Configurazione di una risorsa proxy con l'integrazione proxy Lambda con una definizione OpenAPI

Informazioni sull'integrazione proxy Lambda di API Gateway

L'integrazione proxy Lambda di Amazon API Gateway è un sistema semplice, potente e agile per creare un'API con una configurazione di un singolo metodo API. Permette al client di chiamare una singola funzione Lambda nel back-end. La funzione accede a molte risorse o funzionalità di altri AWS servizi, inclusa la chiamata ad altre funzioni Lambda.

Nell'integrazione proxy Lambda, quando un client invia una richiesta API, API Gateway passa alla funzione Lambda integrata un oggetto evento, ad eccezione del fatto che non viene mantenuto l'ordine dei parametri della richiesta. Questi dati di richiesta includono intestazioni di richiesta, parametri delle stringhe di query, variabili di percorso URL, payload e dati di configurazione API. I dati di configurazione possono includere il nome della fase di distribuzione corrente, variabili di fasi, identità utente o contesto di autorizzazione (se presente). La funzione Lambda back-end analizza i dati delle richieste in entrata per stabilire la risposta da restituire. Perché API Gateway possa passare l'output Lambda come risposta API al client, la funzione Lambda deve restituire il risultato in questo formato.

Poiché API Gateway non interviene in modo significativo tra il client e la funzione Lambda back-end per l'integrazione proxy Lambda, il client e la funzione Lambda integrata possono adattarsi alle modifiche reciproche senza alterare la configurazione di integrazione esistente dell'API. Per consentirlo, il client deve seguire i protocolli di applicazione attuati dalla funzione Lambda back-end.

Puoi configurare un'integrazione proxy Lambda per qualsiasi metodo API. Tuttavia un'integrazione proxy Lambda è più potente quando viene configurata per un metodo API che coinvolge una risorsa proxy generica. La risorsa proxy generica può essere indicata dalla speciale variabile di percorso preconfigurata `{proxy+}`, dal segnaposto del metodo catch-all `ANY` o da entrambi. Il client può passare l'input alla funzione Lambda back-end nella richiesta in entrata come parametri di richiesta o payload applicabile. I parametri di richiesta includono intestazioni, variabili di percorso URL, parametri delle stringhe di query e il payload applicabile. La funzione Lambda integrata verifica tutte le sorgenti di input prima di elaborare la richiesta e rispondere al client con messaggi di errore significativi, se manca qualcuno degli input richiesti.

Quando si chiama un metodo API integrato con il metodo HTTP generico ANY e la risorsa generica {proxy+}, il client invia una richiesta con un metodo HTTP specifico al posto di ANY. Il client inoltre specifica un particolare percorso URL al posto di {proxy+} e include eventuali intestazioni, parametri delle stringhe di query o payload applicabile necessari.

L'elenco di seguito riepiloga i comportamenti di runtime di diversi metodi API con l'integrazione proxy Lambda:

- ANY /{proxy+}: per passare i dati come input alla funzione Lambda integrata, il client deve scegliere un metodo HTTP specifico, deve impostare una determinata gerarchia di percorso di risorsa e può impostare qualsiasi intestazione, parametro di stringa di query e payload applicabile.
- ANY /res: per passare i dati come input alla funzione Lambda integrata, il client deve scegliere un metodo HTTP specifico e può impostare qualsiasi intestazione, parametro di stringa di query e payload applicabile.
- GET|POST|PUT|... /{proxy+}: per passare i dati come input alla funzione Lambda integrata, il client può impostare una specifica gerarchia di percorso di risorsa, qualsiasi intestazione, parametro di stringa di query e payload applicabile.
- GET|POST|PUT|... /res/{path}/...: per passare i dati di input alla funzione Lambda integrata, il client deve scegliere un segmento di percorso specifico (per la variabile {path}) e può impostare qualsiasi intestazione di richiesta, parametro di stringa di query e payload applicabile.
- GET|POST|PUT|... /res: per passare i dati di input alla funzione Lambda integrata, il client può scegliere qualsiasi intestazione di richiesta, parametro di stringa di query e payload applicabile.

La risorsa proxy {proxy+} e quella personalizzata {custom} sono entrambe espresse come variabili di percorso preconfigurate. Tuttavia {proxy+} può fare riferimento a qualsiasi risorsa in una gerarchia di percorso, mentre {custom} fa riferimento solo a un segmento di percorso specifico. Ad esempio, un negozio di generi alimentari potrebbe organizzare il proprio inventario di prodotti online in base ai nomi dei reparti, alle categorie di produzione e ai tipi di prodotto. Il sito Web del negozio potrebbe quindi rappresentare i prodotti disponibili in base alle seguenti variabili di percorso preconfigurate di risorse personalizzate: /{department}/{produce-category}/{product-type}. Ad esempio le mele sono rappresentate da /produce/fruit/apple e le carote da /produce/vegetables/carrot. Può anche usare /{proxy+} per rappresentare qualsiasi reparto, categoria di produzione o tipo di prodotto che un cliente può cercare quando fa acquisti nel negozio online. Ad esempio, /{proxy+} può fare riferimento a uno qualsiasi degli articoli seguenti:

- /produce

- `/produce/fruit`
- `/produce/vegetables/carrot`

Per permettere ai clienti di cercare i prodotti disponibili, la categoria di produzione e il reparto del negozio associato, puoi esporre un singolo metodo `GET /{proxy+}` con autorizzazioni di sola lettura. Allo stesso modo, per consentire a un supervisore di aggiornare l'inventario del `produce` reparto, puoi impostare un altro metodo unico di concessione delle `PUT /produce/{proxy+}` autorizzazioni. read/write Per consentire a un cassiere di aggiornare il totale corrente di un ortaggio, puoi impostare un `POST /produce/vegetables/{proxy+}` metodo con autorizzazioni. read/write Per consentire a un responsabile del negozio di eseguire qualsiasi azione possibile su qualsiasi prodotto disponibile, lo sviluppatore del negozio online può esporre il metodo con le `ANY /{proxy+}` autorizzazioni. read/write In ogni caso, al runtime il cliente o il dipendente deve selezionare un prodotto specifico di un determinato tipo in un reparto scelto, una particolare categoria di produzione in un reparto scelto o un reparto specifico.

Per ulteriori informazioni sull'impostazione delle integrazioni proxy di API Gateway, consulta Configurazione di un'integrazione proxy mediante una risorsa proxy.

Per l'integrazione proxy è necessario che il client abbia una conoscenza più dettagliata dei requisiti del back-end. Pertanto, per garantire prestazioni dell'app e un'esperienza utente ottimali, lo sviluppatore di back-end deve comunicare chiaramente allo sviluppatore del client i requisiti del back-end e fornire un sistema solido di notifica degli errori quando i requisiti non vengono soddisfatti.

Supporto per le intestazioni multi-valore e i parametri di stringa di query

API Gateway supporta più intestazioni e parametri della stringa di query con lo stesso nome. Multi-value le intestazioni, le intestazioni e i parametri a valore singolo possono essere combinati nelle stesse richieste e risposte. Per ulteriori informazioni, consulta Formato di input di una funzione Lambda per l'integrazione proxy e Formato di output di una funzione Lambda per l'integrazione proxy.

Formato di input di una funzione Lambda per l'integrazione proxy

Con l'integrazione proxy Lambda, API Gateway mappa l'intera richiesta client al parametro di input `event` della funzione Lambda di back-end. Nell'esempio seguente viene illustrata la struttura di un evento che API Gateway invia a un'integrazione proxy Lambda.

In questo esempio si presuppone che l'invocazione a Gateway API sia la seguente:

```
curl 'https://a1b2c3.execute-api.us-east-1.amazonaws.com/my/path?
parameter1=value1&parameter2=value1&parameter2=value2&parameter3=value1,value2'
 -H 'header1: value1' -H 'header2: value1' -H 'header2: value2' -H 'header3:
 value1,value2'
```

L'output sarà simile al seguente:

```
{
  "resource": "/my/path",
  "path": "/my/path",
  "httpMethod": "GET",
  "headers": {
      "header1": "value1",
      "header2": "value2",
      "header3": "value1,value2"
  },
  "multiValueHeaders": {
    "header1": ["value1"],
    "header2": ["value1","value2"],
    "header3": ["value1,value2"]
  },
  "queryStringParameters": {
      "parameter1": "value1",
      "parameter2": "value2",
      "parameter3": "value1,value2"
  },
  "multiValueQueryStringParameters": {
    "parameter1": ["value1"],
    "parameter2": ["value1","value2"],
    "parameter3": ["value1,value2"]
  },
  "requestContext": {
    "accountId": "123456789012",
    "apiId": "id",
    "authorizer": {
      "claims": null,
      "scopes": null
    },
    "domainName": "id.execute-api.us-east-1.amazonaws.com",
    "domainPrefix": "id",
    "extendedRequestId": "request-id",
    "httpMethod": "GET",
    "identity": {
```

```
      "accessKey": null,
      "accountId": null,
      "caller": null,
      "cognitoAuthenticationProvider": null,
      "cognitoAuthenticationType": null,
      "cognitoIdentityId": null,
      "cognitoIdentityPoolId": null,
      "principalOrgId": null,
      "sourceIp": "IP",
      "user": null,
      "userAgent": "user-agent",
      "userArn": null,
      "clientCert": {
        "clientCertPem": "CERT_CONTENT",
        "subjectDN": "www.example.com",
        "issuerDN": "Example issuer",
        "serialNumber": "a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1",
        "validity": {
          "notBefore": "May 28 12:30:02 2019 GMT",
          "notAfter": "Aug  5 09:36:04 2021 GMT"
        }
      }
    },
    "path": "/my/path",
    "protocol": "HTTP/1.1",
    "requestId": "id=",
    "requestTime": "04/Mar/2020:19:15:17 +0000",
    "requestTimeEpoch": 1583349317135,
    "resourceId": null,
    "resourcePath": "/my/path",
    "stage": "$default"
  },
  "pathParameters": null,
  "stageVariables": null,
  "body": "Hello from Lambda!",
  "isBase64Encoded": false
}
```

> **Note**
>
> Nell'input:
>
> - La chiave `headers` può contenere solo una decina di intestazioni di valore.

- La chiave `multiValueHeaders` può contenere intestazioni multi-valore e il valore di una decina di intestazioni.
- Se si specificano valori sia per `headers` che per `multiValueHeaders`, API Gateway li unisce in un unico elenco. Se la stessa coppia chiave-valore viene specificata in entrambi, solo i valori di `multiValueHeaders` verranno visualizzati nell'elenco risultante.

Nell'input per la funzione Lambda back-end l'oggetto `requestContext` è una mappa di coppie chiave/valore. In ogni coppia la chiave è il nome di una proprietà della variabile $context e il valore è il valore della proprietà. Gateway API potrebbe aggiungere nuove chiavi alla mappa.

A seconda delle funzionalità abilitate, la mappa `requestContext` può variare in base all'API. Ad esempio, nell'esempio precedente non è specificato alcun tipo di autorizzazione e di conseguenza non è presente alcuna proprietà `$context.authorizer.*` o `$context.identity.*`. Quando è specificato un tipo di autorizzazione, API Gateway passa informazioni sugli utenti autorizzati all'endpoint di integrazione in un oggetto `requestContext.identity` nel modo seguente:

- Quando il tipo di autorizzazione è `AWS_IAM`, le informazioni sugli utenti autorizzati includono proprietà `$context.identity.*`.
- Quando il tipo di autorizzazione è `COGNITO_USER_POOLS` (autorizzazione di Amazon Cognito), le informazioni sugli utenti autorizzati includono le proprietà `$context.identity.cognito*` e `$context.authorizer.claims.*`.
- Quando il tipo di autorizzazione è `CUSTOM` (autorizzazione Lambda), le informazioni sugli utenti autorizzati includono `$context.authorizer.principalId` e altre proprietà `$context.authorizer.*` applicabili.

Formato di output di una funzione Lambda per l'integrazione proxy

Nell'integrazione proxy Lambda, API Gateway richiede la funzione Lambda back-end per restituire l'output in base al seguente formato JSON:

```
{
    "isBase64Encoded": true|false,
    "statusCode": httpStatusCode,
    "headers": { "headerName": "headerValue", ... },
    "multiValueHeaders": { "headerName": ["headerValue", "headerValue2", ...], ... },
    "body": "..."
```

```
}
```

Nell'output:

- Le chiavi `headers` e `multiValueHeaders` possono non essere specificate se non devono essere restituite intestazioni di risposta aggiuntive.
- La chiave `headers` può contenere solo una decina di intestazioni di valore.
- La chiave `multiValueHeaders` può contenere intestazioni multi-valore e il valore di una decina di intestazioni. È possibile utilizzare la chiave `multiValueHeaders` per specificare tutte le intestazioni aggiuntive, incluse quelle con valore singolo.
- Se si specificano valori sia per `headers` che per `multiValueHeaders`, API Gateway li unisce in un unico elenco. Se la stessa coppia chiave-valore viene specificata in entrambi, solo i valori di `multiValueHeaders` verranno visualizzati nell'elenco risultante.

Per abilitare CORS per l'integrazione proxy Lambda, devi aggiungere `Access-Control-Allow-Origin:`*`domain-name`* all'output `headers`. *`domain-name`* può essere `*` per qualsiasi nome di dominio. Viene eseguito il marshalling del parametro `body` di output al front-end come payload di risposta del metodo. Se `body` si tratta di un blob binario, è possibile codificarlo come Base64-encoded stringa `isBase64Encoded` impostandolo `true` e `*/*` configurandolo come Binary Media Type. In caso contrario, puoi impostarlo su `false` o lasciarlo non specificato.

> **Note**
>
> Per ulteriori informazioni sull'abilitazione del supporto binario, consulta Abilitazione del supporto binario tramite la console API Gateway. Per un esempio di funzione Lambda, consulta Restituzione di supporti binari da un'integrazione proxy Lambda in Gateway API.

Se l'output della funzione ha un formato diverso, API Gateway restituisce una risposta di errore `502 Bad Gateway`.

Per restituire una risposta in una funzione Lambda in Node.js, puoi usare comandi come i seguenti:

- Per ottenere un buon risultato, effettuare la chiamata a `callback(null, {"statusCode": 200, "body": "results"})`.
- Per generare un'eccezione, chiama `callback(new Error('internal server error'))`.

- Per un errore lato client (se, ad esempio, un parametro obbligatorio è mancante), puoi chiamare `callback(null, {"statusCode": 400, "body": "Missing parameters of ..."})` per restituire l'errore senza generare un'eccezione.

In una `async` funzione Lambda in Node.js, la sintassi equivalente sarebbe:

- Per ottenere un buon risultato, effettuare la chiamata a `return {"statusCode": 200, "body": "results"}`.
- Per generare un'eccezione, chiama `throw new Error("internal server error")`.
- Per un errore lato client (se, ad esempio, un parametro obbligatorio è mancante), puoi chiamare `return {"statusCode": 400, "body": "Missing parameters of ..."}` per restituire l'errore senza generare un'eccezione.

Configura l'integrazione del proxy Lambda per API Gateway utilizzando AWS CLI

In questa sezione viene illustrato come usare AWS CLI per configurare un'API con l'integrazione proxy Lambda. Per istruzioni dettagliate sull'uso della console API Gateway per configurare una risorsa proxy con l'integrazione proxy Lambda, consulta Tutorial: creazione di una REST API con un'integrazione proxy Lambda.

Per questo scenario useremo la funzione Lambda di esempio seguente come back-end dell'API:

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

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

Rispetto alla configurazione dell'integrazione personalizzata Lambda riportata in the section called " Configurazione di integrazioni personalizzate Lambda ", l'input per questa funzione Lambda può essere espresso nei parametri e nel corpo della richiesta. Avrai maggiore libertà per permettere al client di passare gli stessi dati di input. Qui il client può passare il nome del mittente della formula di saluto come parametro della stringa di query, intestazione o proprietà del corpo. La funzione può anche supportare l'integrazione personalizzata Lambda. La configurazione dell'API è più semplice. Non devi configurare la risposta del metodo o la risposta di integrazione.

Per configurare un'integrazione proxy Lambda utilizzando AWS CLI

1. Utilizza il seguente comando create-rest-api per creare un'API:

   ```
   aws apigateway create-rest-api --name 'HelloWorld (AWS CLI)'
   ```

   L'output sarà simile al seguente:

   ```
   {
       "name": "HelloWorldProxy (AWS CLI)",
       "id": "te6si5ach7",
       "rootResourceId" : "krznpq9xpg",
       "createdDate": 1508461860
   }
   ```

   In questo esempio si utilizza `id` dell'API (`te6si5ach7`) e `rootResourceId` (`krznpq9xpg`).

2. Utilizza il seguente comando create-resource per creare una risorsa `/greeting` di Gateway API:

   ```
   aws apigateway create-resource \
         --rest-api-id te6si5ach7 \
   ```

```
      --parent-id krznpq9xpg \
      --path-part {proxy+}
```

L'output sarà simile al seguente:

```
{
    "path": "/{proxy+}",
    "pathPart": "{proxy+}",
    "id": "2jf6xt",
    "parentId": "krznpq9xpg"
}
```

Nel passaggio successivo si utilizza il valore `id` (`2jf6xt`) della risorsa `{proxy+}` per creare un metodo sulla risorsa `/{proxy+}`.

3. Utilizza il seguente comando put-method per creare una richiesta di metodo `ANY` come `ANY /{proxy+}`:

```
aws apigateway put-method --rest-api-id te6si5ach7 \
       --resource-id 2jf6xt \
       --http-method ANY \
       --authorization-type "NONE"
```

L'output sarà simile al seguente:

```
{
    "apiKeyRequired": false,
    "httpMethod": "ANY",
    "authorizationType": "NONE"
}
```

Questo metodo API permette al client di ricevere o inviare formule di saluto dalla funzione Lambda nel back-end.

4. Utilizza il seguente comando put-integration per configurare l'integrazione del metodo `ANY /{proxy+}` con una funzione Lambda denominata `HelloWorld`. Questa funzione risponde alla richiesta con un messaggio `"Hello, {name}!"` se è specificato il parametro `greeter`, altrimenti con un messaggio `"Hello, World!"` se il parametro della stringa di query non è impostato.

```
aws apigateway put-integration \
      --rest-api-id te6si5ach7 \
      --resource-id 2jf6xt \
      --http-method ANY \
      --type AWS_PROXY \
      --integration-http-method POST \
      --uri arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:HelloWorld/invocations \
      --credentials arn:aws:iam::123456789012:role/apigAwsProxyRole
```

> **Important**
>
> Per le integrazioni Lambda, devi usare il metodo HTTP `POST` per la richiesta di integrazione, secondo la specifica dell'operazione del servizio Lambda per le chiamate della funzione. Il ruolo IAM `apigAwsProxyRole` deve includere policy che permettono al servizio `apigateway` di richiamare funzioni Lambda. Per ulteriori informazioni sulle autorizzazioni IAM, consultare the section called “Modello di autorizzazione API Gateway per invocare un'API”.

L'output sarà simile al seguente:

```
{
    "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"
}
```

Invece di specificare un ruolo IAM per `credentials`, è possibile utilizzare il comando add-permission per aggiungere autorizzazioni basate su risorse. Questo è ciò che fa la console API Gateway.

5. Utilizza il seguente comando create-deployment per distribuire l’API in una fase `test`.

```
aws apigateway create-deployment  \
      --rest-api-id te6si5ach7 \
      --stage-name test
```

6. Testa l'API usando i comandi cURL seguenti in un terminale.

   Chiamata dell'API con il parametro della stringa di query `?greeter=jane`:

   ```
   curl -X GET 'https://te6si5ach7.execute-api.us-west-2.amazonaws.com/test/greeting?
   greeter=jane'
   ```

   Chiamata dell'API con un parametro di intestazione `greeter:jane`:

   ```
   curl -X GET https://te6si5ach7.execute-api.us-west-2.amazonaws.com/test/hi \
     -H 'content-type: application/json' \
     -H 'greeter: jane'
   ```

   Chiamata dell'API con un corpo `{"greeter":"jane"}`:

   ```
   curl -X POST https://te6si5ach7.execute-api.us-west-2.amazonaws.com/test/hi \
     -H 'content-type: application/json' \
     -d '{ "greeter": "jane" }'
   ```

   In tutti i casi, l'output è una risposta 200 con il corpo seguente:

   ```
   Hello, jane!
   ```

Configurazione di una risorsa proxy con l'integrazione proxy Lambda con una definizione OpenAPI

Per configurare una risorsa proxy con il tipo di integrazione proxy Lambda, crea una risorsa API con un parametro di percorso greedy, ad esempio `/parent/{proxy+}`, e integra la risorsa con il back-end di una funzione Lambda , ad esempio `arn:aws:lambda:us-west-2:123456789012:function:SimpleLambda4ProxyResource`, nel metodo `ANY`. Il parametro di percorso greedy deve trovarsi alla fine del percorso della risorsa API. Come per una risorsa non proxy, puoi configurare la risorsa proxy usando la console API Gateway, importando un file di definizione OpenAPI o chiamando direttamente l'API REST di API Gateway.

Il file di definizione API OpenAPI mostra un esempio di API con una risorsa proxy integrata con una funzione Lambda denominata `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"
        }
      }
    }
  }
}
```

Nell'integrazione proxy Lambda, in fase di esecuzione, API Gateway mappa una richiesta in entrata nel parametro di input `event` della funzione Lambda. L'input include il metodo della richiesta, il percorso, le intestazioni, qualsiasi parametro della stringa di query, qualsiasi payload, il contesto associato e qualsiasi variabile di fase definita. Il formato di input viene descritto in Formato di input di una funzione Lambda per l'integrazione proxy. Perché API Gateway possa mappare correttamente l'output di Lambda a risposte HTTP, la funzione Lambda deve restituire il risultato nel formato descritto in Formato di output di una funzione Lambda per l'integrazione proxy.

Con l'integrazione proxy Lambda di una risorsa proxy tramite il metodo `ANY`, la singola funzione Lambda back-end funge da gestore eventi per tutte le richieste attraverso la risorsa proxy. Ad esempio, per registrare i modelli di traffico, puoi fare in modo che un dispositivo mobile invii le informazioni di posizione relative a stato, città, indirizzo ed edificio inviando una richiesta con `/state/city/street/house` nel percorso URL della risorsa proxy. La funzione Lambda back-end può quindi analizzare il percorso URL e inserire le tuple di posizione in una tabella DynamoDB.

Configurazione di integrazioni personalizzate Lambda in API Gateway

Per mostrare come configurare l'integrazione Lambda personalizzata o non proxy, creiamo un'API Gateway API per esporre il `GET /greeting?greeter={name}` metodo per richiamare una funzione Lambda. Usa uno dei seguenti esempi di funzioni Lambda per l'API.

Usa uno dei seguenti esempi di funzioni Lambda:

Node.js

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

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

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

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

  // Return a greeting to the caller
  return greeting;
};
```

Python

```
import json


def lambda_handler(event, context):
    print(event)
    res = {
```

```
        "statusCode": 200,
        "headers": {
            "Content-Type": "*/*"
        }
    }

    if event['greeter'] == "":
        res['body'] = "Hello, World"
    elif (event['greeter']):
        res['body'] = "Hello, " + event['greeter'] + "!"
    else:
        raise Exception('Missing the required greeter parameter.')

    return res
```

Questa funzione risponde con un messaggio `"Hello, {name}!"` se il valore del parametro `greeter` è una stringa non vuota. La funzione restituisce un messaggio `"Hello, World!"` se il valore di `greeter` è una stringa vuota. La funzione restituisce un messaggio di errore `"Missing the required greeter parameter."` se il parametro greeter non è impostato nella richiesta in ingresso. Denominiamo la funzione `HelloWorld`.

Puoi crearla nella console Lambda o tramite la AWS CLI. In questa sezione faremo riferimento alla funzione usando l'ARN seguente:

```
arn:aws:lambda:us-east-1:123456789012:function:HelloWorld
```

Con la funzione Lambda impostata nel back-end, procedi a configurare l'API.

Per configurare l'integrazione personalizzata Lambda utilizzando AWS CLI

1. Utilizza il seguente comando create-rest-api per creare un'API:

   ```
   aws apigateway create-rest-api --name 'HelloWorld (AWS CLI)'
   ```

   L'output sarà simile al seguente:

   ```
   {
       "name": "HelloWorld (AWS CLI)",
       "id": "te6si5ach7",
       "rootResourceId" : "krznpq9xpg",
   ```

```
    "createdDate": 1508461860
}
```

In questo esempio si utilizza `id` dell'API (`te6si5ach7`) e `rootResourceId` (`krznpq9xpg`).

2. Utilizza il seguente comando create-resource per creare una risorsa `/greeting` di Gateway API:

```
aws apigateway create-resource \
      --rest-api-id te6si5ach7 \
      --parent-id krznpq9xpg \
      --path-part greeting
```

L'output sarà simile al seguente:

```
{
    "path": "/greeting",
    "pathPart": "greeting",
    "id": "2jf6xt",
    "parentId": "krznpq9xpg"
}
```

Nel passaggio successivo si utilizza il valore `id` (`2jf6xt`) della risorsa `greeting` per creare un metodo sulla risorsa `/greeting`.

3. Utilizza il seguente comando put-method per creare una richiesta di metodo API `GET /greeting?greeter={name}`:

```
aws apigateway put-method --rest-api-id te6si5ach7 \
       --resource-id 2jf6xt \
       --http-method GET \
       --authorization-type "NONE" \
       --request-parameters method.request.querystring.greeter=false
```

L'output sarà simile al seguente:

```
{
    "apiKeyRequired": false,
    "httpMethod": "GET",
    "authorizationType": "NONE",
    "requestParameters": {
```

```
        "method.request.querystring.greeter": false
    }
}
```

Questo metodo API permette al client di ricevere una formula di saluto dalla funzione Lambda nel back-end. Il parametro `greeter` è facoltativo perché il back-end deve gestire un chiamante anonimo o un chiamante autoidentificato.

4. Utilizza il seguente comando put-method-response per configurare la risposta `200 OK` sulla richiesta di metodo `GET /greeting?greeter={name}`:

```
aws apigateway put-method-response \
        --rest-api-id te6si5ach7 \
        --resource-id 2jf6xt \
        --http-method GET \
        --status-code 200
```

5. Utilizza il seguente comando put-integration per configurare l'integrazione del metodo `GET /greeting?greeter={name}` con una funzione Lambda denominata `HelloWorld`. La funzione risponde alla richiesta con un messaggio `"Hello, {name}!"` se è specificato il parametro `greeter`, altrimenti con `"Hello, World!"` se il parametro della stringa di query non è impostato.

```
aws apigateway put-integration \
        --rest-api-id te6si5ach7 \
        --resource-id 2jf6xt \
        --http-method GET \
        --type AWS \
        --integration-http-method POST \
        --uri arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:HelloWorld/invocations \
        --request-templates '{"application/json":"{\"greeter\":
\"$input.params('greeter')\"}"}' \
        --credentials arn:aws:iam::123456789012:role/apigAwsProxyRole
```

Il modello di mappatura fornito qui traduce il parametro della stringa di query `greeter` nella proprietà `greeter` del payload JSON. Questo è necessario perché l'input di una funzione Lambda deve essere espresso nel corpo.

> **Important**
>
> Per le integrazioni Lambda, devi usare il metodo HTTP `POST` per la richiesta di integrazione, secondo la specifica dell'operazione del servizio Lambda per le chiamate della funzione. Il parametro `uri` è l'ARN dell'operazione di chiamata della funzione. L'output sarà simile al seguente:

```
{
    "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"
}
```

Il ruolo IAM `apigAwsProxyRole` deve includere policy che permettono al servizio `apigateway` di richiamare funzioni Lambda. Invece di specificare un ruolo IAM per `credentials`, puoi chiamare il comando add-permission per aggiungere autorizzazioni basate su risorse. Questo è il modo in cui la console API Gateway aggiunge queste autorizzazioni.

6. Utilizza il seguente comando put-integration-response per configurare la risposta di integrazione in modo da passare l'output della funzione Lambda al client come risposta di metodo `200 OK`:

```
aws apigateway put-integration-response \
       --rest-api-id te6si5ach7 \
       --resource-id 2jf6xt \
       --http-method GET \
       --status-code 200 \
       --selection-pattern ""
```

Impostando il modello di selezione su una stringa vuota, la risposta predefinita è `200 OK`.

L'output sarà simile al seguente:

```
 {
    "selectionPattern": "",
    "statusCode": "200"
}
```

7. Utilizza il seguente comando create-deployment per distribuire l'API in una fase `test`.

```
aws apigateway create-deployment \
        --rest-api-id te6si5ach7 \
        --stage-name test
```

8. Testa l'API usando il comando cURL seguente in un terminale:

```
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'
```

Configurazione della chiamata asincrona della funzione Lambda back-end

Nell'integrazione non proxy (personalizzata) Lambda la funzione Lambda back-end viene richiamata in modo sincrono per impostazione predefinita. Questo è il comportamento desiderato per la maggior parte delle operazioni API REST. Alcune applicazioni, tuttavia, richiedono che la chiamata venga effettuata in modo asincrono, ad esempio un'operazione batch o una con latenza elevata, in genere da un componente di back-end separato. In questo caso, la funzione Lambda back-end viene richiamata in modo asincrono e il metodo API REST front-end non restituisce il risultato.

Puoi configurare la funzione Lambda affinché un'integrazione non proxy Lambda venga richiamata in modo asincrono specificando `'Event'` come tipo di chiamata Lambda. Questo avviene così:

Configurazione della chiamata asincrona Lambda nella console API Gateway

Affinché tutte le invocazioni siano asincrone:

- In Richiesta di integrazione aggiungi un'intestazione `X-Amz-Invocation-Type` con un valore statico `'Event'`.

Affinché i client decidano se le invocazioni sono asincrone o sincrone:

1. In Richiesta metodo aggiungi un'intestazione `InvocationType`.
2. In Richiesta di integrazione aggiungi un'intestazione `X-Amz-Invocation-Type` con un'espressione di mappatura `method.request.header.InvocationType`.
3. I client possono includere l'intestazione `InvocationType: Event` nelle richieste API per le invocazioni asincrone o `InvocationType: RequestResponse` per le invocazioni sincrone.

Configurazione della chiamata asincrona Lambda utilizzando OpenAPI

Affinché tutte le invocazioni siano asincrone:

- Aggiungi l'intestazione `X-Amz-Invocation-Type` alla sezione x-amazon-apigateway-integration.

```
"x-amazon-apigateway-integration" : {
          "type" : "aws",
          "httpMethod" : "POST",
          "uri" : "arn:aws:apigateway:us-east-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-2:123456789012:function:my-function/invocations",
          "responses" : {
            "default" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.header.X-Amz-Invocation-Type" : "'Event'"
          },
          "passthroughBehavior" : "when_no_match",
          "contentHandling" : "CONVERT_TO_TEXT"
        }
```

Affinché i client decidano se le invocazioni sono asincrone o sincrone:

1. Aggiungi la seguente intestazione su qualsiasi oggetto OpenAPI Path Item.

```
"parameters" : [ {
"name" : "InvocationType",
"in" : "header",
"schema" : {
```

```
  "type" : "string"
}
} ]
```

2. Aggiungi l'intestazione `X-Amz-Invocation-Type` alla sezione x-amazon-apigateway-integration.

```
"x-amazon-apigateway-integration" : {
          "type" : "aws",
          "httpMethod" : "POST",
          "uri" : "arn:aws:apigateway:us-east-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-2:123456789012:function:my-function/invocations",
          "responses" : {
            "default" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.header.X-Amz-Invocation-Type" :
 "method.request.header.InvocationType"
          },
          "passthroughBehavior" : "when_no_match",
          "contentHandling" : "CONVERT_TO_TEXT"
        }
```

3. I client possono includere l'intestazione `InvocationType: Event` nelle richieste API per le invocazioni asincrone o `InvocationType: RequestResponse` per le invocazioni sincrone.

Configurare l'invocazione asincrona Lambda utilizzando CloudFormation

I seguenti CloudFormation modelli mostrano come configurare le invocazioni asincrone. `AWS::ApiGateway::Method`

Affinché tutte le invocazioni siano asincrone:

```
AsyncMethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref AsyncResource
      HttpMethod: GET
      ApiKeyRequired: false
```

```
      AuthorizationType: NONE
      Integration:
        Type: AWS
        RequestParameters:
          integration.request.header.X-Amz-Invocation-Type: "'Event'"
        IntegrationResponses:
            - StatusCode: '200'
        IntegrationHttpMethod: POST
        Uri: !Sub arn:aws:apigateway:${AWS::Region}:lambda:path/2015-03-31/functions/
${myfunction.Arn}$/invocations
      MethodResponses:
        - StatusCode: '200'
```

Affinché i client decidano se le invocazioni sono asincrone o sincrone:

```
AsyncMethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref AsyncResource
      HttpMethod: GET
      ApiKeyRequired: false
      AuthorizationType: NONE
      RequestParameters:
        method.request.header.InvocationType: false
      Integration:
        Type: AWS
        RequestParameters:
          integration.request.header.X-Amz-Invocation-Type:
 method.request.header.InvocationType
        IntegrationResponses:
            - StatusCode: '200'
        IntegrationHttpMethod: POST
        Uri: !Sub arn:aws:apigateway:${AWS::Region}:lambda:path/2015-03-31/functions/
${myfunction.Arn}$/invocations
      MethodResponses:
        - StatusCode: '200'
```

I client possono includere l'intestazione `InvocationType: Event` nelle richieste API per le invocazioni asincrone o `InvocationType: RequestResponse` per le invocazioni sincrone.

Gestione degli errori Lambda in API Gateway

Per le integrazioni personalizzate Lambda, devi mappare gli errori restituiti da Lambda nella risposta di integrazione alle risposte di errore HTTP standard per i client. Altrimenti, gli errori Lambda vengono restituiti come risposte `200 OK` per impostazione predefinita e il risultato non è intuitivo per gli utenti dell'API.

Esistono due tipi di errore che Lambda può restituire: errori standard ed errori personalizzati. Nella tua API, devi gestirli in modo diverso.

Con l'integrazione proxy Lambda, Lambda deve restituire un output nel formato seguente:

```
{
  "isBase64Encoded" : "boolean",
  "statusCode": "number",
  "headers": { ... },
  "body": "JSON string"
}
```

In questo output, `statusCode` è in genere `4XX` per un errore del client e `5XX` per un errore del server. API Gateway gestisce questi errori mappando l'errore Lambda a una risposta di errore HTTP, in base al `statusCode` specificato. Affinché API Gateway passi il tipo di errore (ad esempio, `InvalidParameterException`), come parte della risposta al client, la funzione Lambda deve includere un'intestazione (ad esempio, `"X-Amzn-ErrorType":"InvalidParameterException"`) nella proprietà `headers`.

Argomenti

- Gestione degli errori Lambda standard in API Gateway
- Gestione degli errori Lambda personalizzati in API Gateway

Gestione degli errori Lambda standard in API Gateway

Un AWS Lambda errore standard ha il seguente formato:

```
{
  "errorMessage": "<replaceable>string</replaceable>",
  "errorType": "<replaceable>string</replaceable>",
  "stackTrace": [
    "<replaceable>string</replaceable>",
```

```
    ...
  ]
}
```

Qui, `errorMessage` è un'espressione della stringa dell'errore. `errorType` è un tipo di eccezione o di errore dipendente dal linguaggio. `stackTrace` è un elenco di espressioni delle stringhe che mostrano la traccia dello stack che porta al verificarsi dell'errore.

Ad esempio, si consideri la seguente JavaScript funzione Lambda (Node.js).

```
export const handler = function(event, context, callback) {
    callback(new Error("Malformed input ..."));
};
```

Questa funzione restituisce il seguente errore Lambda standard, contenente `Malformed input ...` come messaggio di errore:

```
{
  "errorMessage": "Malformed input ...",
  "errorType": "Error",
  "stackTrace": [
    "export const handler (/var/task/index.js:3:14)"
  ]
}
```

Allo stesso modo, considera la seguente funzione Python Lambda che genera un'eccezione `Exception` con lo stesso messaggio di errore `Malformed input ...`.

```
def lambda_handler(event, context):
    raise Exception('Malformed input ...')
```

La funzione restituisce il seguente errore Lambda standard:

```
{
  "stackTrace": [
    [
      "/var/task/lambda_function.py",
      3,
      "lambda_handler",
      "raise Exception('Malformed input ...')"
```

```
    ]
  ],
  "errorType": "Exception",
  "errorMessage": "Malformed input ..."
}
```

Da notare che i valori delle proprietà `errorType` e `stackTrace` dipendono dal linguaggio. L'errore standard si applica anche a qualsiasi oggetto di errore che è un'estensione dell'oggetto `Error` o una sottoclasse della classe `Exception`.

Per mappare l'errore Lambda standard a una risposta del metodo, devi prima stabilire il codice di stato HTTP per un dato errore Lambda. Quindi impostate un modello di espressione regolare sulla `selectionPattern` proprietà del codice di stato HTTP IntegrationResponseassociato al dato codice di stato HTTP. Nella console Gateway API, questo `selectionPattern` è indicato come Espressione regolare errore Lambda sotto ogni risposta di integrazione nella sezione Risposta di integrazione.

> **Note**
>
> API Gateway usa le espressioni regolari basate sullo stile del modello Java per la mappatura della risposta. Per ulteriori informazioni, consulta Pattern nella documentazione Oracle.

Ad esempio, utilizzate quanto segue put-integration-responseper impostare una nuova `selectionPattern` espressione:

```
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
```

Accertati di configurare anche il codice di errore corrispondente (`400`) sulla risposta del metodo. Altrimenti, API Gateway restituisce una risposta di errore di configurazione non valida al runtime.

> **Note**
>
> Al runtime, API Gateway abbina l'elemento `errorMessage` dell'errore Lambda al modello dell'espressione regolare per la proprietà `selectionPattern`. Se c'è una corrispondenza, API Gateway restituisce l'errore Lambda come risposta HTTP del codice di stato HTTP corrispondente. Se non c'è corrispondenza, API Gateway restituisce l'errore come risposta

predefinita oppure genera un'eccezione di configurazione non valida se non è stata configurata una risposta predefinita.
Impostare il valore `selectionPattern` su `.*` per una data risposta significa reimpostare questa risposta come risposta predefinita. Questo succede perché tale modello di selezione corrisponderà a tutti i messaggi di errore, incluso null, come, ad esempio, messaggi di errore non specificati. La mappatura che ne deriva sovrascrive la mappatura predefinita. Se si usa `.+` come modello di selezione per filtrare le risposte, potrebbe non corrispondere a una risposta contenente un carattere di nuova riga (’\n).

Per aggiornare un `selectionPattern` valore esistente utilizzando AWS CLI, chiamate l'update-integration-responseoperazione per sostituire il valore del `/selectionPattern` percorso con l'espressione regex specificata del `Malformed*` pattern.

Per impostare l'espressione `selectionPattern` tramite la console Gateway API, immetti l'espressione nella casella di testo Espressione regolare errore Lambda quando configuri o aggiorni una risposta di integrazione di un codice di stato HTTP specifico.

Gestione degli errori Lambda personalizzati in API Gateway

Invece dell'errore standard descritto nella sezione precedente, AWS Lambda consente di restituire un oggetto di errore personalizzato come stringa JSON. L'errore può essere qualsiasi oggetto JSON valido. Ad esempio, la seguente funzione Lambda JavaScript (Node.js) restituisce un errore personalizzato:

```
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));
```

```
};
```

Devi trasformare l'oggetto `myErrorObj` in una stringa JSON prima di chiamare `callback` per uscire dalla funzione. In caso contrario, viene restituito `myErrorObj` come stringa di `"[object Object]"`. Quando un metodo dell'API è integrato con la precedente funzione Lambda, API Gateway riceve una risposta di integrazione con il seguente payload:

```
{
    "errorMessage": "{\"errorType\":\"InternalServerError\",\"httpStatus\":500,
\"requestId\":\"e5849002-39a0-11e7-a419-5bb5807c9fb2\",\"trace\":{\"function\":
\"abc()\",\"line\":123,\"file\":\"abc.js\"}}"
}
```

Come nel caso delle risposte di integrazione, puoi passare questa risposta di errore invariata alla risposta di metodo. Oppure puoi avere un modello di mappatura per trasformare il payload in un formato diverso. Ad esempio, considera il seguente modello di mappatura del corpo per la risposta di un metodo del codice di stato `500`:

```
{
    errorMessage: $input.path('$.errorMessage');
}
```

Questo modello traduce il corpo della risposta di integrazione che contiene la stringa JSON di errore personalizzato nel corpo di risposta del metodo seguente. Questo corpo di risposta del metodo contiene un oggetto JSON di errore personalizzato:

```
{
    "errorMessage" : {
        errorType : "InternalServerError",
        httpStatus : 500,
        requestId : context.awsRequestId,
        trace : {
            "function": "abc()",
            "line": 123,
            "file": "abc.js"
        }
    }
};
```

In base ai tuoi requisiti API, potresti aver bisogno di passare alcune o tutte le proprietà degli errori personalizzati come parametri dell'intestazione della risposta di metodo. Puoi farlo applicando le mappature degli errori personalizzati dal corpo della risposta di integrazione alle intestazioni della risposta di metodo.

Ad esempio, la seguente estensione OpenAPI definisce una mappatura dalle proprietà `errorMessage.errorType`, `errorMessage.httpStatus`, `errorMessage.trace.function` ed `errorMessage.trace` alle intestazioni `error_type`, `error_status`, `error_trace_function` ed `error_trace`, rispettivamente.

```
"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"
          },
          ...
        }
    }
}
```

Al runtime, API Gateway deserializza il parametro `integration.response.body` nel corso delle mappature delle intestazioni. Tuttavia, questa deserializzazione si applica solo alle body-to-header mappature per le risposte di errore personalizzate Lambda e non si applica alle mappature che utilizzano. body-to-body `$input.body` Con queste mappature custom-error-body-to -header, il client riceve le seguenti intestazioni come parte della risposta del metodo, a condizione che,, e le intestazioni siano dichiarate nella richiesta del `error_status` metodo. `error_trace` `error_trace_function` `error_type`

```
"error_status":"500",
"error_trace":"{\"function\":\"abc()\",\"line\":123,\"file\":\"abc.js\"}",
"error_trace_function":"abc()",
```

```
"error_type":"InternalServerError"
```

La proprietà `errorMessage.trace` del corpo della risposta di integrazione è una proprietà complessa. Viene mappata all'intestazione `error_trace` come una stringa JSON.

Integrazioni HTTP per REST APIs in API Gateway

Puoi integrare un metodo API con un endpoint HTTP tramite l'integrazione proxy di HTTP o l'integrazione personalizzata di HTTP.

API Gateway supporta le seguenti porte dell'endpoint: 80, 443 e 1024-65535.

Con l'integrazione proxy la configurazione è semplice. Devi solo impostare il metodo HTTP e l'URI dell'endpoint HTTP, in base ai requisiti del back-end, se non sei interessato alla codifica o al caching dei contenuti.

Con l'integrazione personalizzata la configurazione è più complessa. Oltre alla procedura di configurazione dell'integrazione proxy, devi specificare come verranno mappati i dati della richiesta in ingresso alla richiesta di integrazione e i dati della risposta di integrazione risultante alla risposta del metodo.

Argomenti

- Configurazione di integrazioni proxy HTTP in API Gateway
- Configurazione di integrazioni personalizzate HTTP in API Gateway

Configurazione di integrazioni proxy HTTP in API Gateway

Per configurare una risorsa proxy con il tipo di integrazione proxy HTTP, crea una risorsa API con un parametro di percorso greedy, ad esempio `/parent/{proxy+}`, e integra la risorsa con un endpoint di back-end HTTP, ad esempio `https://petstore-demo-endpoint.execute-api.com/petstore/{proxy}`, nel metodo `ANY`. Il parametro di percorso greedy deve trovarsi alla fine del percorso della risorsa.

Come per una risorsa non proxy, puoi configurare una risorsa proxy con l'integrazione proxy HTTP usando la console API Gateway, importando un file di definizione OpenAPI o chiamando direttamente l'API REST di API Gateway. Per istruzioni dettagliate sull'uso della console API Gateway per configurare una risorsa proxy con l'integrazione HTTP, consulta Tutorial: creazione di una REST API con l'integrazione proxy HTTP.

Il seguente file di definizione OpenAPI mostra un esempio di API con una risorsa proxy integrata con il PetStoresito 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"
        }
      }
    }
  }
}
```

In questo esempio, viene dichiarata una chiave cache nel parametro del percorso `method.request.path.proxy` della risorsa proxy. Questa è l'impostazione predefinita quando crei l'API tramite la console API Gateway. Il percorso di base dell'API (`/test`corrispondente a una fase) è mappato alla PetStore pagina del sito Web (`/petstore`). La singola richiesta di integrazione rispecchia l'intero PetStore sito Web utilizzando la variabile greedy path dell'API e il metodo `ANY` catch-all. Negli esempi seguenti viene illustrato questo mirroring.

- Imposta **`ANY`** come **`GET`** e **`{proxy+}`** come **`pets`**

  Richiesta di metodo iniziata dal front-end:

  ```
  GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets HTTP/1.1
  ```

  Richiesta di integrazione inviata al back-end:

  ```
  GET http://petstore-demo-endpoint.execute-api.com/petstore/pets HTTP/1.1
  ```

Le istanze run-time del metodo ANY e le risorse proxy sono entrambe valide. La chiamata restituisce una risposta `200 OK` con il payload contenente il primo batch di animali, nel modo in cui viene restituito dal back-end.

- Imposta **`ANY`** come **`GET`** e **`{proxy+}`** come **`pets?type=dog`**

```
GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets?type=dog
 HTTP/1.1
```

Richiesta di integrazione inviata al back-end:

```
GET http://petstore-demo-endpoint.execute-api.com/petstore/pets?type=dog HTTP/1.1
```

Le istanze run-time del metodo ANY e le risorse proxy sono entrambe valide. La chiamata restituisce una risposta `200 OK` con il payload contenente il primo batch di cani specifici, nel modo in cui viene restituito dal back-end.

- Imposta **`ANY`** come **`GET`** e **`{proxy+}`** come **`pets/{petId}`**

Richiesta di metodo iniziata dal front-end:

```
GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets/1 HTTP/1.1
```

Richiesta di integrazione inviata al back-end:

```
GET http://petstore-demo-endpoint.execute-api.com/petstore/pets/1 HTTP/1.1
```

Le istanze run-time del metodo ANY e le risorse proxy sono entrambe valide. La chiamata restituisce una risposta `200 OK` con il payload contenente l'animale specificato, nel modo in cui viene restituito dal back-end.

- Imposta **`ANY`** come **`POST`** e **`{proxy+}`** come **`pets`**

Richiesta di metodo iniziata dal front-end:

```
POST https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets HTTP/1.1
Content-Type: application/json
Content-Length: ...

{
```

```
  "type" : "dog",
  "price" : 1001.00
}
```

Richiesta di integrazione inviata al back-end:

```
POST http://petstore-demo-endpoint.execute-api.com/petstore/pets HTTP/1.1
Content-Type: application/json
Content-Length: ...

{
  "type" : "dog",
  "price" : 1001.00
}
```

Le istanze run-time del metodo ANY e le risorse proxy sono entrambe valide. La chiamata restituisce una risposta `200 OK` con il payload contenente il nuovo animale creato, nel modo in cui viene restituito dal back-end.

- Imposta **`ANY`** come **`GET`** e **`{proxy+}`** come **`pets/cat`**

  Richiesta di metodo iniziata dal front-end:

  ```
  GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets/cat
  ```

  Richiesta di integrazione inviata al back-end:

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

  L'istanza run-time del percorso della risorsa del proxy non corrisponde all'endpoint di un back-end e la relativa richiesta non è valida. Di conseguenza, viene restituita una risposta `400 Bad Request` con il seguente messaggio di errore.

  ```
  {
    "errors": [
      {
        "key": "Pet2.type",
        "message": "Missing required field"
      },
      {
        "key": "Pet2.price",
  ```

```
      "message": "Missing required field"
    }
  ]
}
```

- Imposta **`ANY`** come **`GET`** e **`{proxy+}`** come **`null`**

  Richiesta di metodo iniziata dal front-end:

  ```
  GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test
  ```

  Richiesta di integrazione inviata al back-end:

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

  La risorsa di destinazione è il padre della risorsa del proxy, ma l'istanza run-time del metodo `ANY` non è definita nell'API di quella risorsa. Di conseguenza, la richiesta `GET` restituisce una risposta `403 Forbidden` con il messaggio di errore `Missing Authentication Token` restituito da API Gateway. Se l'API espone il metodo `ANY` o `GET` alla risorsa padre (`/`), la chiamata restituisce una risposta `404 Not Found` con il messaggio `Cannot GET /petstore` restituito dal back-end.

Per qualsiasi richiesta del client, se l'URL dell'endpoint di destinazione non è valido o il verbo HTTP è valido ma non è supportato, il back-end restituisce una risposta `404 Not Found`. Per un metodo HTTP non supportato, viene restituita una risposta `403 Forbidden`.

Configurazione di integrazioni personalizzate HTTP in API Gateway

Con l'integrazione HTTP personalizzata, conosciuta anche come integrazione non proxy, si ha più controllo su quali dati passare tra un metodo API e un'integrazione API e su come passarli. Puoi fare questo utilizzando le mappature dei dati.

Come parte della configurazione della richiesta del metodo, imposti la proprietà requestParameters su una risorsa Method. Questo dichiara quali parametri della richiesta di metodo, forniti dal client, devono essere mappati ai parametri di richiesta di integrazione o alle proprietà del corpo applicabili prima di essere inviati al back-end. Quindi, come parte della configurazione della richiesta di integrazione, impostate la proprietà RequestParameters sulla risorsa di integrazione corrispondente per specificare le parameter-to-parameter mappature. Imposti inoltre la proprietà requestTemplates per specificare i modelli di mappatura, uno per ogni tipo di contenuto supportato. I modelli di

mappatura mappano i parametri della richiesta di metodo, o il corpo, al corpo della richiesta di integrazione.

Analogamente, come parte della configurazione della risposta del metodo, impostate la proprietà ResponseParameters sulla risorsa. MethodResponse Questo dichiara quali parametri della risposta di metodo, da inviare al client, devono essere mappati dai parametri della risposta di integrazione o da alcune proprietà del corpo applicabili restituite dal back-end. Quindi si configura selectionPattern per scegliere una risposta di integrazione basata sulla risposta del backend. Per un'integrazione HTTP non proxy, si tratta di un'espressione regolare. Ad esempio, per mappare tutti i codici di stato delle risposte HTTP 2xx da un endpoint HTTP a questa mappatura di output, si usa `2\d{2}`.

> **Note**
>
> API Gateway usa le espressioni regolari basate sullo stile del modello Java per la mappatura della risposta. Per ulteriori informazioni, consulta Pattern nella documentazione Oracle.

Quindi, come parte della configurazione della risposta di integrazione, impostate la proprietà ResponseParameters sulla risorsa IntegrationResponsecorrispondente per specificare le mappature. parameter-to-parameter Imposti inoltre la mappa responseTemplates per specificare i modelli di mappatura, uno per ogni tipo di contenuto supportato. I modelli di mappatura mappano i parametri della risposta di integrazione, o le proprietà del corpo della risposta di integrazione, al corpo della risposta del metodo.

Per ulteriori informazioni sulla configurazione dei modelli di mappatura, consulta Trasformazioni dei dati per REST API in Gateway API.

Trasmetti in streaming la risposta di integrazione per le integrazioni proxy in API Gateway

Puoi configurare l'integrazione del proxy per controllare in che modo API Gateway restituisce la tua risposta di integrazione. Per impostazione predefinita, API Gateway attende di ricevere la risposta completa prima di iniziare la trasmissione. Tuttavia, se imposti la modalità di trasferimento delle risposte dell'integrazione su`STREAM`, API Gateway non attende che una risposta sia completamente calcolata prima di inviarla al client. Lo streaming di risposte funziona per tutti i tipi di endpoint di REST API.

Utilizza lo streaming di risposte per i seguenti casi d'uso:

- Riduci il time-to-first-byte (TTFB) per le applicazioni di intelligenza artificiale generativa come i chatbot.
- Trasmetti in streaming file di immagini, video o musica di grandi dimensioni senza utilizzare un URL prefirmato S3.
- Esegui operazioni di lunga durata segnalando progressi incrementali come gli eventi inviati dal server (SSE).
- Supera il limite di payload di risposta di 10 MB di API Gateway.
- Supera il limite di timeout di 29 secondi di API Gateway senza richiedere un aumento del limite di timeout di integrazione.
- Ricevi un payload binario senza configurare i tipi di supporti binari.

Considerazioni sullo streaming del payload di risposta

Le seguenti considerazioni potrebbero influire sull'utilizzo dello streaming di payload di risposta:

- È possibile utilizzare lo streaming di payload di risposta solo per i `HTTP_PROXY` tipi di integrazione. `AWS_PROXY` Ciò include le integrazioni proxy Lambda e le integrazioni private che utilizzano integrazioni. `HTTP_PROXY`
- L'impostazione predefinita della modalità di trasferimento è. `BUFFERED` Per utilizzare lo streaming di risposta, è necessario modificare la modalità di trasferimento della risposta in`STREAM`.
- Lo streaming di risposte è supportato solo per le API REST.
- Lo streaming delle richieste non è supportato.
- Puoi trasmettere in streaming la tua risposta per un massimo di 15 minuti.

- I tuoi stream sono soggetti a timeout di inattività. Per gli endpoint regionali o privati, il timeout è di 5 minuti. Per gli endpoint ottimizzati per l'edge, il timeout è di 30 secondi.
- Se utilizzi lo streaming di risposta per un'API REST regionale con la tua CloudFront distribuzione, puoi ottenere un timeout di inattività superiore a 30 secondi aumentando il timeout di risposta della tua distribuzione. CloudFront Per ulteriori informazioni, consulta Timeout di risposta.
- Quando la modalità di trasferimento della risposta è impostata su`STREAM`, API Gateway non può supportare funzionalità che richiedono il buffering dell'intera risposta di integrazione. Per questo motivo, le seguenti funzionalità non sono supportate con lo streaming di risposta:
  - Memorizzazione nella cache degli endpoint
  - Codifica dei contenuti. Se desideri comprimere la tua risposta di integrazione, esegui questa operazione nella tua integrazione.
  - Trasformazione della risposta con VTL
- All'interno di ogni risposta in streaming, i primi 10 MB di payload di risposta non sono soggetti ad alcuna restrizione di larghezza di banda. I dati di payload di risposta superiori a 10 MB sono limitati a. 2MB/s
- Quando la connessione tra il client e API Gateway o tra API Gateway e Lambda viene chiusa a causa di un timeout, la funzione Lambda potrebbe continuare a essere eseguita. Per ulteriori informazioni, consulta Configurare il timeout della funzione Lambda.
- Lo streaming delle risposte comporta un costo. Per ulteriori informazioni, consulta Prezzi di API Gateway.

Configura un'integrazione proxy HTTP con lo streaming di risposta del payload in API Gateway

Quando si configura lo streaming del payload di risposta, si specifica la modalità di trasferimento della risposta nella richiesta di integrazione del metodo. Queste impostazioni vengono configurate nella richiesta di integrazione per controllare il comportamento di API Gateway prima e durante la risposta di integrazione. Quando utilizzi lo streaming di risposte, puoi configurare il timeout di integrazione fino a 15 minuti.

Quando utilizzi lo streaming di risposta del payload con un'`HTTP_PROXY`integrazione, API Gateway non invia il codice di stato della risposta HTTP o alcuna intestazione di risposta HTTP finché non riceve completamente tutte le intestazioni.

Crea un'integrazione proxy HTTP con lo streaming di risposta del payload

La procedura seguente mostra come importare una nuova API con l'`responseTransferMode`impostazione su. `STREAM` Se disponi di un'API di integrazione esistente e desideri modificarla`responseTransferMode`, consultathe section called “Aggiorna la modalità di trasferimento delle risposte per un'integrazione con proxy HTTP”.

Console di gestione AWS

Per creare un'integrazione proxy HTTP con lo streaming di risposta del payload

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere una REST API.
3. Scegli Crea risorsa.
4. Per Resource Name (Nome risorsa) immetti **`streaming`**.
5. Scegli Crea risorsa.
6. Con la risorsa /streaming selezionata, scegli il metodo Create.
7. Per Tipo di metodo, scegliete QUALSIASI.
8. Per Tipo di integrazione scegli Fittizio.
9. Scegli l'integrazione del proxy HTTP.
10. Per la modalità di trasferimento Response, scegli Stream.
11. Per il metodo HTTP, scegliete un metodo.
12. Per Endpoint URL, inserisci un endpoint di integrazione. Assicurati di scegliere un endpoint che produca un payload di grandi dimensioni da trasmettere in streaming.
13. Scegli Crea metodo.

Dopo aver creato il metodo, implementa l'API.

Per distribuire l'API

1. Seleziona Deploy API (Distribuisci API).
2. In Fase, seleziona Nuova fase.
3. In Stage name (Nome fase) immettere **`prod`**.
4. (Facoltativo) In Descrizione, immetti una descrizione.
5. Seleziona Implementa.

AWS CLI

Per creare una nuova API con streaming di risposta al payload

1. Copia il seguente file Open API, quindi salvalo con nome. `ResponseStreamDemoSwagger.yaml` In questo file, `responseTransferMode` è impostato su`STREAM`. L'endpoint di integrazione è impostato su`https://example.com`, ma ti consigliamo di modificarlo su un endpoint che produca un payload di grandi dimensioni da trasmettere in streaming.

```
openapi: "3.0.1"
info:
  title: "ResponseStreamingDemo"
  version: "2025-04-28T17:28:25Z"
servers:
- url: "{basePath}"
  variables:
    basePath:
      default: "prod"
paths:
  /streaming:
    get:
      x-amazon-apigateway-integration:
        httpMethod: "GET"
        uri: "https://example.com"
        type: "http_proxy"
        timeoutInMillis: 900000
        responseTransferMode: "STREAM"
```

2. Usa il seguente `import-rest-api` comando per importare la tua definizione OpenAPI:

```
aws apigateway import-rest-api \
  --body 'fileb://~/ResponseStreamDemoSwagger.yaml' \
  --parameters endpointConfigurationTypes=REGIONAL \
  --region us-west-1
```

3. Usa il seguente `create-deployment` comando per distribuire la tua nuova API in una fase:

```
aws apigateway create-deployment \
  --rest-api-id a1b2c3 \
  --stage-name prod \
  --region us-west-1
```

Aggiorna la modalità di trasferimento delle risposte per un'integrazione con proxy HTTP

La procedura seguente mostra come aggiornare la modalità di trasferimento della risposta per un'integrazione con proxy HTTP.

Console di gestione AWS

Per aggiornare la modalità di trasferimento delle risposte per un'integrazione con proxy HTTP

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere una REST API.
3. Scegliere un metodo.
4. Nella scheda Richiesta di integrazione scegli Modifica in Impostazioni della richiesta di integrazione.
5. Per la modalità di trasferimento Response, scegli Stream.
6. Scegli Save (Salva).

Dopo aver aggiornato il metodo, distribuisci l'API.

Per distribuire l'API

1. Seleziona Deploy API (Distribuisci API).
2. In Fase, seleziona Nuova fase.
3. In Stage name (Nome fase) immettere **`prod`**.
4. (Facoltativo) In Descrizione, immetti una descrizione.
5. Seleziona Implementa.

AWS CLI

Il `update-integration` comando seguente aggiorna la modalità di trasferimento di un'integrazione da `BUFFERED` a`STREAM`. Per tutte le API esistenti, la modalità di trasferimento della risposta per tutte le integrazioni è impostata su. `BUFFERED`

```
aws apigateway update-integration \
 --rest-api-id a1b2c3 \
 --resource-id aaa111 \
```

```
 --http-method GET \
 --patch-operations "op='replace',path='/responseTransferMode',value=STREAM" \
 --region us-west-1
```

È necessario ridistribuire l'API per rendere effettive le modifiche. Se hai personalizzato il timeout di integrazione, questo valore di timeout viene rimosso, poiché API Gateway trasmette la tua risposta per un massimo di 5 minuti.

Il `update-integration` comando seguente aggiorna la modalità di trasferimento di un'integrazione da a: `STREAM` `BUFFERED`

```
aws apigateway update-integration \
 --rest-api-id a1b2c3 \
 --resource-id aaa111 \
 --http-method GET \
 --patch-operations "op='replace',path='/responseTransferMode',value=BUFFERED" \
 --region us-west-1
```

È necessario ridistribuire l'API per rendere effettive le modifiche.

Configura l'integrazione di un proxy Lambda con lo streaming della risposta del payload in API Gateway

Puoi trasmettere in streaming la risposta di una funzione Lambda per migliorare le prestazioni del time-to-first byte (TTFB) e inviare risposte parziali al client non appena diventano disponibili. API Gateway richiede l'utilizzo dell'API InvokeWithResponseStreamLambda per richiamare la funzione Lambda. API Gateway passa un oggetto evento alla funzione Lambda. La funzione Lambda back-end analizza i dati delle richieste in entrata per stabilire la risposta da restituire. Affinché API Gateway possa trasmettere l'output Lambda, la funzione Lambda deve restituire il formato richiesto da API Gateway.

Differenze nelle integrazioni del proxy Lambda tra la modalità di trasferimento di risposta in streaming e bufferizzata

L'elenco seguente descrive le differenze tra un'integrazione con proxy Lambda e un'integrazione con proxy Lambda per lo streaming di risposte:

- API Gateway utilizza l'InvokeWithResponseStreamAPI per richiamare l'integrazione del proxy Lambda per lo streaming delle risposte. Ciò si traduce in un URI diverso, che è il seguente:

```
arn:aws:apigateway:us-west-1:lambda:path/2021-11-15/functions/arn:aws:lambda:us-
west-1:111122223333:function:my-function-name/response-streaming-invocations
```

Questo ARN utilizza una data diversa per la versione dell'API e un'azione di servizio diversa rispetto all'integrazione del proxy Lambda.

Se utilizzi la console API Gateway per lo streaming delle risposte, la console utilizza l'URI corretto per te.

- In un'integrazione con proxy Lambda, API Gateway invia la risposta al client solo dopo aver ricevuto la risposta completa da Lambda. In un'integrazione proxy Lambda per lo streaming di risposte, API Gateway avvia il flusso di payload dopo aver ricevuto i metadati e il delimitatore validi da Lambda.
- L'integrazione del proxy Lambda per lo streaming di risposte utilizza lo stesso formato di input dell'integrazione proxy, ma richiede un formato di output diverso.

Formato di integrazione proxy Lambda per lo streaming delle risposte

Quando API Gateway richiama una funzione Lambda con streaming di risposte, il formato di input è lo stesso di quello di una funzione Lambda per l'integrazione del proxy. Per ulteriori informazioni, consulta the section called “Formato di input di una funzione Lambda per l'integrazione proxy”.

Quando Lambda invia una risposta a API Gateway, la risposta deve rispettare il seguente formato. Questo formato utilizza un delimitatore per separare i metadati JSON dal payload non elaborato. In questo caso, i dati del payload vengono trasmessi in streaming man mano che vengono trasmessi dalla funzione di streaming Lambda:

```
{
  "headers": {"headerName": "headerValue", ...},
  "multiValueHeaders": { "headerName": ["headerValue", "headerValue2", ...], ... },
  "cookies" : ["cookie1", "cookie2"],
  "statusCode": httpStatusCode
}<DELIMITER>PAYLOAD1 | PAYLOAD2 | PAYLOAD3
```

Nell'output:

- Le `statusCode` chiaviheaders, `multiValueHeaders`cookies, e possono non essere specificate se non devono essere restituite intestazioni di risposta aggiuntive.

- La chiave `headers` può contenere solo una decina di intestazioni di valore.
- L'output prevede che le intestazioni contengano o. `Transfer-Encoding: chunked` `Content-length: number` Se la funzione non restituisce nessuna di queste intestazioni, API Gateway viene aggiunta `Transfer-Encoding: chunked` all'intestazione della risposta.
- La chiave `multiValueHeaders` può contenere intestazioni multi-valore e il valore di una decina di intestazioni. È possibile utilizzare la chiave `multiValueHeaders` per specificare tutte le intestazioni aggiuntive, incluse quelle con valore singolo.
- Se si specificano valori sia per `headers` che per `multiValueHeaders`, API Gateway li unisce in un unico elenco. Se la stessa coppia chiave-valore viene specificata in entrambi, solo i valori di `multiValueHeaders` verranno visualizzati nell'elenco risultante.
- I metadati devono essere in formato JSON valido. Solo `headersmultiValueHeaders`, `cookies` e le `statusCode` chiavi sono supportate.
- È necessario fornire un delimitatore dopo i metadati JSON. Il delimitatore deve essere composto da 8 byte nulli e deve apparire all'interno dei primi 16 KB di dati di flusso.
- API Gateway non richiede un formato specifico per il payload di risposta del metodo.

Se utilizzi l'URL di una funzione per lo streaming della tua funzione Lambda, devi modificare l'input e l'output della funzione Lambda per soddisfare questi requisiti.

Se l'output della funzione Lambda non soddisfa i requisiti di questo formato, API Gateway potrebbe comunque richiamare la funzione Lambda. La tabella seguente mostra le combinazioni delle impostazioni della richiesta di integrazione API e del codice della funzione Lambda supportato da API Gateway. Ciò include le combinazioni supportate per la modalità di trasferimento della risposta in modalità buffered.

Modalità di trasferimento della risposta	Il codice della funzione è conforme al formato richiesto	API di richiamo Lambda	Supportato da API Gateway
Flusso	Sì	InvokeWithResponseStream	Sì. API Gateway trasmette la tua risposta.
Flusso	No	InvokeWithResponseStream	No. API Gateway richiama la funzione Lambda e restituisce

Modalità di trasferimento della risposta	Il codice della funzione è conforme al formato richiesto	API di richiamo Lambda	Supportato da API Gateway
			una risposta di errore 500.
Flusso	Sì	Invoke	No. API Gateway non supporta questa configurazione di integrazione.
Flusso	No	Invoke	No. API Gateway non supporta questa configurazione di integrazione.
Bufferato	Sì	InvokeWithResponseStream	No. API Gateway non supporta questa configurazione di integrazione.
Bufferato	No	InvokeWithResponseStream	No. API Gateway non supporta questa configurazione di integrazione.
Bufferato	Sì	Invoke	API Gateway restituisce le intestazioni HTTP e il codice di stato ma non il corpo della risposta.

Modalità di trasferimento della risposta	Il codice della funzione è conforme al formato richiesto	API di richiamo Lambda	Supportato da API Gateway
Bufferato	No	Invoke	Sì. Si tratta di un'integrazione proxy Lambda. Per ulteriori informazioni, consulta Integrazione del proxy Lambda.

Configura l'integrazione di un proxy Lambda con lo streaming della risposta del payload in API Gateway

Quando configuri lo streaming del payload di risposta, specifichi la modalità di trasferimento nella richiesta di integrazione della risorsa. Queste impostazioni vengono configurate nella richiesta di integrazione per controllare il comportamento di API Gateway prima e durante la risposta di integrazione.

Esempi di funzioni Lambda per lo streaming delle risposte

La tua funzione Lambda deve rispettare il. the section called “Formato di integrazione proxy Lambda per lo streaming delle risposte” Ti consigliamo di utilizzare una delle tre funzioni Lambda di esempio per testare lo streaming delle risposte. Quando crei la tua funzione Lambda, assicurati di fare quanto segue:

- Fornisci un timeout adeguato per la tua funzione. Ti consigliamo di configurare un timeout di almeno 1 minuto per conoscere lo streaming di risposte. Quando crei le tue risorse di produzione, assicurati che il timeout della funzione Lambda copra l'intero ciclo di richiesta. Per ulteriori informazioni, consulta Configurare il timeout della funzione Lambda.
- Usa il runtime Node.js più recente.
- Usa una regione in cui è disponibile lo streaming di risposta Lambda.

Using HttpResponseStream.from

Il seguente esempio di codice trasmette gli oggetti di metadati e i payload JSON al client utilizzando il `awslambda.HttpResponseStream()` metodo senza utilizzare il metodo pipeline.

Non è necessario creare il delimitatore. Per ulteriori informazioni, consulta Scrittura di funzioni Lambda abilitate allo streaming di risposte.

```
export const handler = awslambda.streamifyResponse(
  async (event, responseStream, context) => {
    const httpResponseMetadata = {
      "statusCode": 200,
      "headers": {
        "x-foo": "bar"
      },
      "multiValueHeaders": {
        "x-mv1": ["hello", "world"],
        "Set-Cookie": ["c1=blue", "c2=red"]
      }
    };

    responseStream = awslambda.HttpResponseStream.from(responseStream,
 httpResponseMetadata);
    await new Promise(r => setTimeout(r, 1000)); // synthetic delay

    responseStream.write("First payload ");
    await new Promise(r => setTimeout(r, 1000)); // synthetic delay

    responseStream.write("Final payload");
    responseStream.end();
});
```

Using the pipeline method

Lambda consiglia, quando si scrivono funzioni abilitate allo streaming di risposte, di utilizzare il `awslambda.streamifyResponse()` decoratore fornito dai runtime nativi di Node.js e il metodo. `pipeline()` Quando usi il metodo pipeline, non è necessario creare il delimitatore, Lambda lo fa per te. Per ulteriori informazioni, consulta Scrittura di funzioni Lambda abilitate allo streaming di risposte.

Il seguente esempio di codice invia gli oggetti di metadati JSON e tre payload al client.

```
import { pipeline } from 'node:stream/promises';
import { Readable } from 'node:stream';

export const handler = awslambda.streamifyResponse(
  async (event, responseStream, context) => {
```

```
    const httpResponseMetadata = {
      statusCode: 200,
      headers: {
        "Content-Type": "text/plain",
        "X-Custom-Header": "Example-Custom-Header"
      }
    };

    responseStream = awslambda.HttpResponseStream.from(responseStream,
 httpResponseMetadata);

    const dataStream = Readable.from(async function* () {
      yield "FIRST payload\n";
      await new Promise(r => setTimeout(r, 1000));
      yield "SECOND payload\n";
      await new Promise(r => setTimeout(r, 1000));
      yield "THIRD payload\n";
      await new Promise(r => setTimeout(r, 1000));
    }());

    await pipeline(dataStream, responseStream);
  }
);
```

Without using the pipeline method

Il seguente esempio di codice invia gli oggetti di metadati JSON e tre payload al client senza utilizzare il metodo. `awslambda.HttpResponseStream()` Senza il `awslambda.HttpResponseStream()` metodo, è necessario includere un delimitatore di 8 byte nulli tra i metadati e il payload.

```
export const handler = awslambda.streamifyResponse(async (event, response, ctx) => {
  response.write('{"statusCode": 200, "headers": {"hdr-x": "val-x"}}');
  response.write("\x00".repeat(8)); // DELIMITER
  await new Promise(r => setTimeout(r, 1000));

  response.write("FIRST payload");
  await new Promise(r => setTimeout(r, 1000));

  response.write("SECOND payload");
  await new Promise(r => setTimeout(r, 1000));

  response.write("FINAL payload");
```

```
  response.end();
});
```

Crea un'integrazione proxy Lambda con lo streaming di risposta del payload

La procedura seguente mostra come creare un'integrazione proxy Lambda con lo streaming di risposta del payload. Usa la funzione Lambda di esempio o creane una tua.

Console di gestione AWS

Per creare un'integrazione proxy Lambda con lo streaming di risposta del payload

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere una REST API.
3. Scegli Crea risorsa.
4. Per Resource Name (Nome risorsa) immetti **`streaming`**.
5. Scegli Crea risorsa.
6. Con la risorsa /streaming selezionata, scegli il metodo Create.
7. Per Tipo di metodo, scegliete QUALSIASI.
8. Per Tipo di integrazione, scegli Lambda.
9. Scegli l'integrazione con proxy Lambda.
10. Per la modalità di trasferimento Response, scegli Stream.
11. Per la funzione Lambda, scegli il nome della tua funzione Lambda.

    La console API Gateway utilizza automaticamente InvokeWithResponseStreaml'API per richiamare la funzione Lambda. Sei responsabile della scrittura di una funzione Lambda abilitata per lo streaming di risposte. Per vedere un esempio, consulta the section called "Esempi di funzioni Lambda per lo streaming delle risposte".
12. Scegli Crea metodo.

Dopo aver creato il metodo, distribuisci l'API.

Per distribuire l'API

1. Seleziona Deploy API (Distribuisci API).
2. In Fase, seleziona Nuova fase.

3. In Stage name (Nome fase) immettere **`prod`**.
4. (Facoltativo) In Descrizione, immetti una descrizione.
5. Seleziona Implementa.

AWS CLI

La procedura seguente mostra come importare una nuova API con l'`responseTransferMode`impostazione su. `STREAM` Se disponi di un'API di integrazione esistente e desideri modificarla`responseTransferMode`, consultathe section called “Aggiorna la modalità di trasferimento delle risposte per un'integrazione con proxy Lambda”.

Per creare una nuova API con streaming di risposta al payload

1. Copia il seguente file Open API, quindi salvalo con nome. `ResponseStreamDemoSwagger.yaml` In questo file, `responseTransferMode` è impostato `STREAM` su e l'URI di integrazione è impostato su`arn:aws:apigateway:us-west-1:lambda:path/2021-11-15/functions/arn:aws:lambda:us-west-1:111122223333:function:my-function-name/response-streaming-invocations`.

   Sostituisci il nome della funzione `my-function` con una funzione abilitata allo streaming e sostituisci le credenziali con un ruolo IAM dotato di policy che consentono al `apigateway` servizio di richiamare le funzioni Lambda.

```
openapi: "3.0.1"
info:
  title: "ResponseStreamingDemo"
  version: "2025-04-28T17:28:25Z"
servers:
- url: "{basePath}"
  variables:
    basePath:
      default: "prod"
paths:
  /lambda:
    get:
      x-amazon-apigateway-integration:
        httpMethod: "POST"
```

```
        uri: "arn:aws:apigateway:us-west-1:lambda:path/2021-11-15/functions/
arn:aws:lambda:us-west-1:111122223333:function:my-function-name/response-
streaming-invocations"
        type: "aws_proxy"
        timeoutInMillis: 90000
        responseTransferMode: "STREAM"
        credentials: "arn:aws:iam::111122223333:role/apigateway-lambda-role"
```

Invece di fornire un ruolo IAM per le credenziali, puoi utilizzare il `add-permission` comando per Lambda per aggiungere autorizzazioni basate sulle risorse.

2. Usa il seguente `import-rest-api` comando per importare la tua definizione OpenAPI:

```
aws apigateway import-rest-api \
  --body 'fileb://~/ResponseStreamDemoSwagger.yaml' \
  --parameters endpointConfigurationTypes=REGIONAL \
  --region us-west-1
```

3. Usa il seguente `create-deployment` comando per distribuire la tua nuova API in una fase:

```
aws apigateway create-deployment \
  --rest-api-id a1b2c2 \
  --stage-name prod \
  --region us-west-1
```

Aggiorna la modalità di trasferimento delle risposte per un'integrazione con proxy Lambda

La procedura seguente mostra come aggiornare la modalità di trasferimento delle risposte per un'integrazione proxy Lambda. Quando modifichi la modalità di trasferimento della risposta in streaming, aggiorna la funzione Lambda in modo che soddisfi i requisiti per lo streaming di risposta. Usa la funzione Lambda di esempio o creane una tua.

Console di gestione AWS

Per aggiornare la modalità di trasferimento delle risposte per l'integrazione di un proxy Lambda

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere una REST API.
3. Scegliere un metodo.

4. Nella scheda Richiesta di integrazione scegli Modifica in Impostazioni della richiesta di integrazione.
5. Per la modalità di trasferimento Response, scegli Stream.
6. Per la funzione Lambda, scegli il nome della tua funzione Lambda.
7. Scegli Save (Salva).

Dopo aver aggiornato il metodo, distribuisci l'API.

Per distribuire l'API

1. Seleziona Deploy API (Distribuisci API).
2. In Fase, seleziona Nuova fase.
3. In Stage name (Nome fase) immettere **`prod`**.
4. (Facoltativo) In Descrizione, immetti una descrizione.
5. Seleziona Implementa.

AWS CLI

1. Aggiorna la tua funzione Lambda in modo che sia abilitata allo streaming.
2. Usa il seguente AWS CLI comando per aggiornare l'URI di integrazione e la modalità di trasferimento delle risposte della tua integrazione:

```
aws apigateway update-integration \
 --rest-api-id a1b2c3 \
 --resource-id aaa111 \
 --http-method ANY \
 --patch-operations "[{\"op\":\"replace\",\"path\":\"/uri\",\"value
\":\"arn:aws:apigateway:us-west-1:lambda:path/2021-11-15/functions/
arn:aws:lambda:us-west-1:111122223333:function:my-function-name/response-
streaming-invocations\"}, {\"op\":\"replace\",\"path\":\"/responseTransferMode
\",\"value\":\"STREAM\"}]" \
 --region us-west-1
```

3. Implementa nuovamente l'API per rendere effettive le modifiche.

Risolvi i problemi relativi allo streaming delle risposte in API Gateway

La seguente guida alla risoluzione dei problemi potrebbe aiutarti a risolvere i problemi con le API che utilizzano lo streaming di risposte.

Risoluzione dei problemi generali

Puoi utilizzare la scheda test TestInvokeMethodo la scheda test della console per testare la risposta dello stream. Le seguenti considerazioni potrebbero influire sull'utilizzo di test invoke per lo streaming di risposte:

- Quando testate il vostro metodo, API Gateway memorizza nel buffer il payload di risposta in streaming. Una volta soddisfatta una delle seguenti condizioni, API Gateway restituisce una risposta unica contenente il payload bufferizzato:
  - La richiesta è completa
  - Sono trascorsi 35 secondi
  - È stato memorizzato nel buffer più di 1 MB di payload di risposta
- Se trascorrono più di 35 secondi prima che il metodo restituisca lo stato di risposta HTTP e tutte le intestazioni, lo stato della risposta restituito è 0. TestInvokeMethod
- API Gateway non produce alcun log di esecuzione.

Dopo aver distribuito l'API, puoi testare la risposta dello stream utilizzando un comando curl. Ti consigliamo di utilizzare l'`-i`opzione per includere le intestazioni di risposta del protocollo nell'output. Per vedere i dati di risposta non appena arrivano, usa l'opzione curl `--no-buffer`

Risolvi gli errori cURL

Se stai testando un'integrazione e ricevi l'errore`curl: (18) transfer closed with outstanding read data remaining`, assicurati che il timeout dell'integrazione sia sufficientemente lungo. Se utilizzi una funzione Lambda, devi aggiornare il timeout di risposta della funzione Lambda. Per ulteriori informazioni, consulta Configurare il timeout della funzione Lambda.

Risolvi i problemi relativi all'utilizzo della registrazione degli accessi

Puoi utilizzare i log di accesso per la fase dell'API REST per registrare e risolvere i problemi del flusso di risposta. Oltre alle variabili esistenti, puoi utilizzare le seguenti variabili di registro degli accessi:

`$context.integration.responseTransferMode`

La modalità di trasferimento delle risposte della tua integrazione. Ciò può essere `BUFFERED` o `STREAMED`.

`$context.integration.timeToAllHeaders`

Il tempo che intercorre tra il momento in cui API Gateway stabilisce la connessione di integrazione e il momento in cui riceve tutte le intestazioni di risposta all'integrazione dal client.

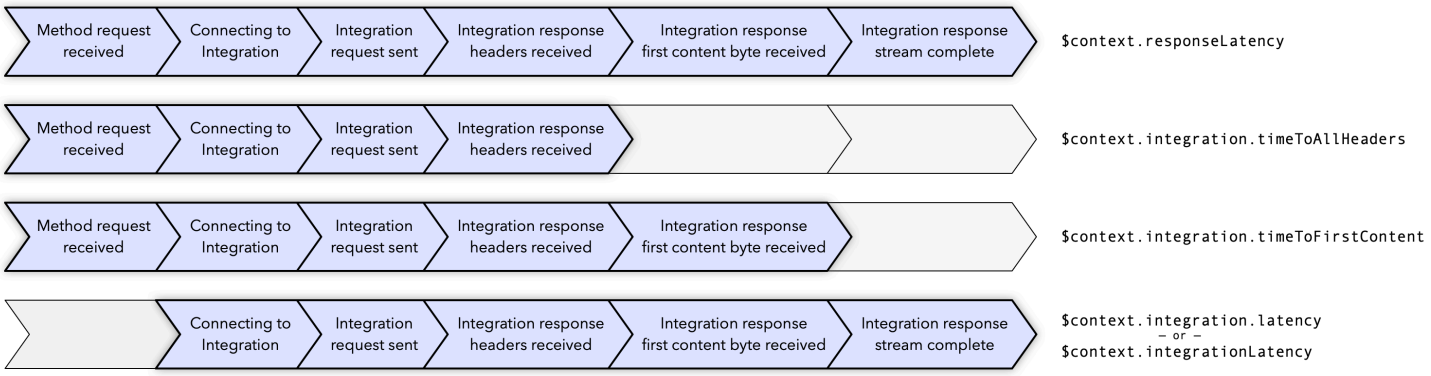
`$context.integration.timeToFirstContent`

Il tempo che intercorre tra il momento in cui API Gateway stabilisce la connessione di integrazione e il momento in cui riceve i primi byte di contenuto.

`$context.integration.latency` o `$context.integrationLatency`

Il momento in cui API Gateway stabilisce la connessione di integrazione e il momento in cui il flusso di risposta dell'integrazione è completo.

La figura seguente mostra come queste variabili di log di accesso rappresentino diversi componenti di un flusso di risposta.



Per ulteriori informazioni sui log degli accessi al, consultare the section called “CloudWatch registri”. Puoi anche utilizzarle X-Ray per monitorare il flusso di risposta. Per ulteriori informazioni, consulta the section called “X-Ray”.

Integrazioni private per REST APIs in API Gateway

Utilizza un'integrazione privata per esporre HTTP/HTTPS le tue risorse all'interno di un Amazon VPC per consentire l'accesso da parte di client esterni al VPC. Ciò estende l'accesso alle tue risorse VPC private oltre i confini del VPC. È possibile controllare l'accesso all'API utilizzando uno qualsiasi dei metodi di autorizzazione supportati da API Gateway.

Per creare un'integrazione privata, devi innanzitutto creare un collegamento VPC. API Gateway supporta i collegamenti VPC V2 per REST. APIs I collegamenti VPC V2 consentono di creare integrazioni private che collegano l'API REST agli Application Load Balancer senza utilizzare un Network Load Balancer. L'utilizzo di un Application Load Balancer ti consente di connetterti alle applicazioni ECSs basate su container Amazon e a molti altri backend. I link VPC V1 sono considerati un tipo di integrazione legacy. Sebbene siano supportati da API Gateway, ti consigliamo di non creare nuovi collegamenti VPC V1.

Considerazioni

Le seguenti considerazioni potrebbero influire sull'utilizzo delle integrazioni private:

- Tutte le risorse devono essere di proprietà dello stesso. Account AWS Ciò include il sistema di bilanciamento del carico, il collegamento VPC e l'API REST.
- Per impostazione predefinita, il traffico di integrazione privata utilizza il protocollo HTTP. Per utilizzare HTTPS, specifica un file `uri`che contenga un nome di server sicuro, ad esempio. `https://example.com:443/test`
- In un'integrazione privata, API Gateway include la parte di fase dell'endpoint API nella richiesta alle risorse di backend. Ad esempio, se richiedi la `test` fase di un'API, API Gateway include `test/path` nella richiesta alla tua integrazione privata. Per rimuovere il nome della fase dalla richiesta alle risorse di backend, utilizzate la mappatura dei parametri per creare un override per la variabile. `$context.requestOverride.path`
- Le integrazioni private con AWS Cloud Map non sono supportate.

Argomenti

- Configura i collegamenti VPC V2 in API Gateway
- Configura un'integrazione privata
- Integrazione privata tramite collegamenti VPC V1 (legacy)

Configura i collegamenti VPC V2 in API Gateway

I link VPC ti consentono di creare integrazioni private che collegano i tuoi percorsi API a risorse private in un VPC, come Application Load Balancers o applicazioni basate su container Amazon ECS. Un'integrazione privata utilizza un collegamento VPC V2 per incapsulare le connessioni tra API Gateway e risorse VPC mirate. Puoi riutilizzare i link VPC su diverse risorse e API.

Quando crei un collegamento VPC, API Gateway crea e gestisce interfacce di rete elastiche per il collegamento VPC V2 nel tuo account. Questo processo può richiedere alcuni minuti. Quando un collegamento VPC V2 è pronto per l'uso, il suo stato passa da a. `PENDING` `AVAILABLE`

> **Note**
>
> Se non viene inviato alcun traffico tramite il collegamento VPC per 60 giorni, lo stato cambia in `INACTIVE`. Quando lo stato di un collegamento VPC è `INACTIVE`, API Gateway elimina tutte le interfacce di rete del collegamento VPC. In questo caso, le richieste API che dipendono dal collegamento VPC non vanno a buon fine. Se le richieste API riprendono, API Gateway effettua nuovamente il provisioning delle interfacce di rete. Potrebbero essere necessari alcuni minuti per creare le interfacce di rete e riattivare il collegamento VPC. Puoi utilizzare lo stato del collegamento VPC per monitorare lo stato del tuo collegamento VPC.

Creare un collegamento VPC V2 utilizzando AWS CLI

Per creare un collegamento VPC V2, tutte le risorse coinvolte devono essere di proprietà dello stesso account. AWS Il comando create-vpc-link seguente crea un collegamento VPC:

```
aws apigatewayv2 create-vpc-link --name MyVpcLink \
    --subnet-ids subnet-aaaa subnet-bbbb \
    --security-group-ids sg1234 sg5678
```

> **Note**
>
> I link VPC V2 sono immutabili. Dopo aver creato un collegamento VPC V2, non puoi modificarne le sottoreti o i gruppi di sicurezza.

Eliminare un collegamento VPC V2 utilizzando AWS CLI

Il comando delete-vpc-link seguente elimina un collegamento VPC.

```
aws apigatewayv2 delete-vpc-link --vpc-link-id abcd123
```

Disponibilità in base alla Regione

I collegamenti VPC V2 sono supportati nelle seguenti regioni e zone di disponibilità:

Nome Regione	Regione	Zone di disponibilità supportate
Stati Uniti orientali (Ohio)	us-east-2	use2-az1, use2-az2, use2-az3
Stati Uniti orientali (Virginia settentrionale)	us-east-1	use1-az1, use1-az2, use1-az4, use1-az5, use1-az6
Stati Uniti occidentali (California settentrionale)	us-west-1	usw1-az1, usw1-az3
Stati Uniti occidentali (Oregon)	us-west-2	usw2-az1, usw2-az2, usw2-az3, usw2-az4
Asia Pacific (Hong Kong)	ap-east-1	ape1-az2, ape1-az3
Asia Pacifico (Mumbai)	ap-south-1	aps1-az1, aps1-az2, aps1-az3
Asia Pacifico (Osaka-Locale)	ap-northeast-3	apne3-az1, apne3-az2, apne3-az3
Asia Pacific (Seoul)	ap-northeast-2	apne2-az1, apne2-az2, apne2-az3
Asia Pacifico (Singapore)	ap-southeast-1	apse1-az1, apse1-az2, apse1-az3

Nome Regione	Regione	Zone di disponibilità supportate
Asia Pacifico (Sydney)	ap-southeast-2	apse2-az1, apse2-az2, apse2-az3
Asia Pacifico (Tokyo)	ap-northeast-1	apne1-az1, apne1-az2, apne1-az4
Canada (Centrale)	ca-central-1	cac1-az1, cac1-az2
Europa (Francoforte)	eu-central-1	euc1-az1, euc1-az2, euc1-az3
Europa (Irlanda)	eu-west-1	euw1-az1, euw1-az2, euw1-az3
Europe (London)	eu-west-2	euw2-az1, euw2-az2, euw2-az3
Europa (Parigi)	eu-west-3	euw3-az1, euw3-az3
Europa (Spagna)	eu-south-2	eus2-az1, eus2-az2, eus2-az3
Europa (Stoccolma)	eu-north-1	eun1-az1, eun1-az2, eun1-az3
Medio Oriente (Bahrein)	me-south-1	mes1-az1, mes1-az2, mes1-az3
Sud America (San Paolo)	sa-east-1	sae1-az1, sae1-az2, sae1-az3

Nome Regione	Regione	Zone di disponibilità supportate
AWS GovCloud (US-West)	us-gov-west-1	usgw1-az1, usgw1-az2, usgw1-az3

Configura un'integrazione privata

Per creare un'integrazione privata con un Application Load Balancer o Network Load Balancer, devi creare un'integrazione proxy HTTP, specificare il link VPC V2 da utilizzare e fornire l'ARN di un Network Load Balancer o di un Application Load Balancer. Per impostazione predefinita, il traffico di integrazione privata utilizza il protocollo HTTP. Per utilizzare HTTPS, specificane uno `uri`che contenga un nome di server sicuro, ad esempio. `https://example.com:443/test` Per un tutorial completo su come creare un'API REST con un'integrazione privata, consultathe section called “Tutorial: creazione di una REST API con un'integrazione privata”.

Creazione di un'integrazione privata

La procedura seguente mostra come creare un'integrazione privata che si connette a un sistema di bilanciamento del carico utilizzando un collegamento VPC V2.

Console di gestione AWS

Per un tutorial su come creare un'integrazione privata, vedi,. the section called “Tutorial: creazione di una REST API con un'integrazione privata”

AWS CLI

Il seguente comando put-integration crea un'integrazione privata che si connette a un sistema di bilanciamento del carico utilizzando un collegamento VPC V2:

```
aws apigateway put-integration \
    --rest-api-id abcdef123 \
    --resource-id aaa000 \
    --integration-target 'arn:aws:elasticloadbalancing:us-
east-2:111122223333:loadbalancer/app/myLoadBalancerName/1234567891011' \
    --uri 'https://example.com:443/path' \
    --http-method GET \
    --type HTTP_PROXY \
    --integration-http-method GET \
```

```
    --connection-type VPC_LINK \
    --connection-id bbb111
```

Invece di fornire direttamente l'ID di connessione, puoi utilizzare una variabile di fase. Quando distribuisci l'API su una fase, imposti l'ID VPC link V2. Il seguente comando put-integration crea un'integrazione privata utilizzando una variabile stage per l'ID VPC link V2:

```
aws apigateway put-integration \
    --rest-api-id abcdef123 \
    --resource-id aaa000 \
    --integration-target 'arn:aws:elasticloadbalancing:us-
east-2:111122223333:loadbalancer/app/myLoadBalancerName/1234567891011' \
    --uri 'https://example.com:443/path' \
    --http-method GET \
    --type HTTP_PROXY \
    --integration-http-method GET \
    --connection-type VPC_LINK \
    --connection-id "\${stageVariables.vpcLinkV2Id}"
```

Assicurati di inserire due virgolette nell'espressione della variabile stage ($ {stageVariables.vpclinkV2ID}) e di evitare il carattere $.

OpenAPI

È possibile configurare un'API con integrazione privata importando il file OpenAPI dell'API. Le impostazioni sono simili alle definizioni di OpenAPI di un'API con integrazioni HTTP, con le eccezioni seguenti:

- Devi impostare esplicitamente `connectionType` su `VPC_LINK`.
- Devi impostare esplicitamente `connectionId` sull'ID di un oggetto `VpcLinkV2` o su una variabile di fase che fa riferimento all'ID di un oggetto `VpcLinkV2`.
- Il `uri` parametro nell'integrazione privata punta a un HTTP/HTTPS endpoint nel VPC, ma viene invece utilizzato per configurare l'intestazione della richiesta `Host` di integrazione.
- Il parametro `uri` nell'integrazione privata con un endpoint HTTPS in VPC viene usato per verificare il nome di dominio indicato confrontandolo con quello nel certificato installato nell'endpoint VPC.

Per fare riferimento all'ID di `VpcLinkV2`, è possibile usare una variabile di fase. In alternativa, è possibile assegnare il valore ID direttamente a `connectionId`.

Il file OpenAPI dell'API in formato JSON seguente mostra un esempio di un'API con un collegamento VPC in base al riferimento nella variabile di fase (`${stageVariables.vpcLinkIdV2}`):

```
{
  "swagger": "2.0",
  "info": {
    "version": "2017-11-17T04:40:23Z",
    "title": "MyApiWithVpcLinkV2"
  },
  "host": "abcdef123.execute-api.us-west-2.amazonaws.com",
  "basePath": "/test",
  "schemes": [
    "https"
  ],
  "paths": {
    "/": {
      "get": {
        "produces": [
          "application/json"
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "uri": "https://example.com:443/path",
          "passthroughBehavior": "when_no_match",
          "connectionType": "VPC_LINK",
          "connectionId": "${stageVariables.vpcLinkV2Id}",
          "integration-target": "arn:aws:elasticloadbalancing:us-
east-2:111122223333:loadbalancer/app/myLoadBalancerName/1234567891011",
          "httpMethod": "GET",
          "type": "http_proxy"
```

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

Aggiorna un'integrazione privata

L'esempio seguente aggiorna il collegamento VPC V2 per un'integrazione privata.

Console di gestione AWS

Per aggiornare un'integrazione privata

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli un'API REST con un'integrazione privata.
3. Scegli la risorsa e il metodo che utilizzano un'integrazione privata.
4. Nella scheda Richiesta di integrazione scegli Modifica in Impostazioni della richiesta di integrazione.
5. Puoi modificare l'impostazione della tua integrazione privata. Se attualmente utilizzi un collegamento VPC V1, puoi modificare il tuo collegamento VPC in un collegamento VPC V2.
6. Scegli Save (Salva).
7. Implementa nuovamente l'API per rendere effettive le modifiche.

AWS CLI

Il seguente comando update-integration aggiorna un'integrazione privata per utilizzare un collegamento VPC V2:

```
aws apigateway update-integration \
    --rest-api-id a1b2c3d4e5 \
    --resource-id a1b2c3 \
```

```
    --http-method GET \
    --patch-operations "[{\"op\":\"replace\",\"path\":\"/connectionId\",\"value
\":\"pk0000\"}, {\"op\":\"replace\",\"path\":\"/uri\",\"value\":\"http://
example.com\"}, {\"op\":\"replace\",\"path\":\"/integrationTarget\",\"value
\":\"arn:aws:elasticloadbalancing:us-east-2:111122223333:loadbalancer/app/
myLoadBalancerName/1234567891011\"}]"
```

Integrazione privata tramite collegamenti VPC V1 (legacy)

> **Note**
>
> La seguente implementazione di integrazioni private utilizza i collegamenti VPC V1. I link VPC V1 sono risorse legacy. Ti consigliamo di utilizzare i collegamenti VPC V2 per REST. APIs

Per creare un'integrazione privata, è necessario innanzitutto creare un collegamento Network Load Balancer. Il Network Load Balancer deve disporre di un listener che instrada le richieste alle risorse del VPC. Per migliorare la disponibilità della tua API, assicurati che il sistema di Network Load Balancer stia indirizzando il traffico alle risorse in più zone di disponibilità nella Regione AWS. Quindi, si crea un collegamento VPC che viene utilizzato per connettere l'API e il Network Load Balancer. Dopo avere creato un collegamento VPC, è possibile creare integrazioni private per instradare il traffico dall'API alle risorse del VPC tramite il collegamento VPC e il Network Load Balancer. Il Network Load Balancer e l'API devono essere di proprietà dello stesso. Account AWS

Argomenti

- Configurazione di un Network Load Balancer per le integrazioni private di API Gateway (legacy)
- Concedi le autorizzazioni per API Gateway per creare un collegamento VPC (legacy)
- Configura un'API API Gateway con integrazioni private utilizzando AWS CLI (legacy)
- Account API Gateway utilizzati per integrazioni private (legacy)

Configurazione di un Network Load Balancer per le integrazioni private di API Gateway (legacy)

> **Note**
>
> La seguente implementazione di integrazioni private utilizza i collegamenti VPC V1. I link VPC V1 sono risorse legacy. Ti consigliamo di utilizzare i collegamenti VPC V2 per REST. APIs

Nella procedura seguente vengono illustrate le fasi necessarie per configurare un sistema Network Load Balancer (NLB) per le integrazioni private di API Gateway usando la console Amazon EC2 e vengono forniti i riferimenti a istruzioni dettagliate per ogni fase.

Per ogni VPC in cui sono presenti risorse, è sufficiente configurare un NLB e uno. VPCLink Il NLB supporta più listener e gruppi target per NLB. È possibile configurare ogni servizio come listener specifico sull'NLB e utilizzarne uno per connettersi all'NLB. VPCLink Al momento della creazione dell'integrazione privata in API Gateway è possibile quindi definire ciascun servizio utilizzando la porta specifica assegnata per il servizio. Per ulteriori informazioni, consulta the section called “Tutorial: creazione di una REST API con un'integrazione privata”. Il Network Load Balancer e l'API devono essere di proprietà dello stesso. Account AWS

Per creare un sistema Network Load Balancer per l'integrazione privata tramite la console API Gateway

1. Accedi Console di gestione AWS e apri la console Amazon EC2 all'indirizzo. https://console.aws.amazon.com/ec2/
2. Configura un server Web in un'istanza di Amazon EC2. Per una configurazione di esempio, consulta Installazione di un server Web LAMP su Amazon Linux 2.
3. Crea un sistema Network Load Balancer, registra l'istanza EC2 con un gruppo target e aggiungi il gruppo target a un listener del sistema Network Load Balancer. Per ulteriori informazioni, segui le istruzioni contenute in Nozioni di base sui sistemi Network Load Balancer.
4. Dopo la creazione del Network Load Balancer, procedi come segue:

    a. Prendi nota dell'ARN del Network Load Balancer. Sarà necessario per creare un collegamento VPC in API Gateway per l'integrazione dell'API con le risorse VPC nel sistema Network Load Balancer.

    b. Disattiva la valutazione dei gruppi di sicurezza per PrivateLink.

- Per disattivare la valutazione dei gruppi di sicurezza per il PrivateLink traffico che utilizza la console, puoi scegliere la scheda Sicurezza, quindi Modifica. Nelle impostazioni di sicurezza, deseleziona Applica le regole in entrata al PrivateLink traffico.
- Usa il seguente set-security-groupscomando per disattivare la valutazione dei gruppi di sicurezza per il PrivateLink traffico:

```
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**
>
> Non aggiungete alcuna dipendenza ad API Gateway CIDRs poiché sono destinate a cambiare senza preavviso.

Concedi le autorizzazioni per API Gateway per creare un collegamento VPC (legacy)

> **Note**
>
> La seguente implementazione di integrazioni private utilizza i collegamenti VPC V1. I link VPC V1 sono risorse legacy. Ti consigliamo di utilizzare i collegamenti VPC V2 per REST. APIs

Per poter creare e gestire un collegamento VPC, è necessario disporre delle autorizzazioni per creare, eliminare e visualizzare le configurazioni del servizio endpoint VPC, modificare le autorizzazioni del servizio endpoint VPC ed esaminare i sistemi di bilanciamento del carico. Per concedere tali autorizzazioni, esegui la procedura illustrata di seguito.

Per concedere le autorizzazioni per creare, aggiornare o eliminare un link VPC

1. Crea una policy IAM simile alla seguente:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "apigateway:POST",
                "apigateway:GET",
                "apigateway:PATCH",
                "apigateway:DELETE"
            ],
            "Resource": [
                "arn:aws:apigateway:us-east-1::/vpclinks",
                "arn:aws:apigateway:us-east-1::/vpclinks/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:DescribeLoadBalancers"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpointServiceConfiguration",
                "ec2:DeleteVpcEndpointServiceConfigurations",
                "ec2:DescribeVpcEndpointServiceConfigurations",
                "ec2:ModifyVpcEndpointServicePermissions"
            ],
            "Resource": "*"
        }
    ]
}
```

Se desideri abilitare il tagging per il collegamento VPC, assicurati di consentire le operazioni di tagging. Per ulteriori informazioni, consulta the section called “Consentire operazioni di assegnazione di tag”.

2. Crea o scegli un ruolo IAM e collega la policy precedente al ruolo.
3. Assegna il ruolo IAM a te stesso o a un utente nell'account che sta creando collegamenti VPC.

Configura un'API API Gateway con integrazioni private utilizzando AWS CLI (legacy)

> **Note**
>
> La seguente implementazione di integrazioni private utilizza i collegamenti VPC V1. I link VPC V1 sono risorse legacy. Ti consigliamo di utilizzare i collegamenti VPC V2 per REST. APIs

Il seguente tutorial mostra come utilizzare per AWS CLI creare un collegamento VPC e un'integrazione privata. Devono essere soddisfatti i seguenti prerequisiti:

- È necessario creare e configurare un Network Load Balancer con l'origine VPC come destinazione. Per ulteriori informazioni, consulta Configurazione di un Network Load Balancer per le integrazioni private di API Gateway (legacy). Deve essere Account AWS uguale alla tua API. È necessario l'ARN del Network Load Balancer per creare il collegamento VPC.
- Per creare e gestire un `VpcLink`, sono necessarie le autorizzazioni per creare un `VpcLink` nell'API. Non sono necessarie autorizzazioni per utilizzare il `VpcLink`. Per ulteriori informazioni, consulta Concedi le autorizzazioni per API Gateway per creare un collegamento VPC (legacy).

Per configurare un'API con l'integrazione privata utilizzando AWS CLI

1. Usa il create-vpc-linkcomando seguente per creare un `VpcLink` target per il Network Load Balancer specificato:

```
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
```

L'output di questo comando riconosce la ricezione della richiesta e mostra lo stato `PENDING` per il `VpcLink` in fase di creazione.

```
{
    "status": "PENDING",
    "targetArns": [
        "arn:aws:elasticloadbalancing:us-east-2:123456789012:loadbalancer/net/my-
vpclink-test-nlb/1234567890abcdef"
    ],
    "id": "gim7c3",
    "name": "my-test-vpc-link"
}
```

Per la creazione dell'oggetto `VpcLink` in API Gateway sono necessari da 2 a 4 minuti. Dopo il completamento dell'operazione, il valore di `status` è `AVAILABLE`. È possibile verificarlo utilizzando il seguente comando: get-vpc-link

```
aws apigateway get-vpc-link --vpc-link-id gim7c3
```

Se l'operazione non riesce, lo stato è `FAILED` e `statusMessage` contiene il messaggio di errore. Se, ad esempio, tenti di creare un oggetto `VpcLink` con un sistema Network Load Balancer già associato a un endpoint VPC, la proprietà `statusMessage` contiene quanto segue:

```
"NLB is already associated with another VPC Endpoint Service"
```

Dopo la corretta creazione di `VpcLink` è possibile creare un'API e integrarla con la risorsa VPC tramite la `VpcLink`.

Prendi nota del valore `id` del `VpcLink` appena creato. In questo output di esempio, è `gim7c3`. Questo valore è necessario per configurare l'integrazione privata.

2. Utilizzate il seguente create-rest-apicomando per creare una `RestApi`risorsa API Gateway:

```
aws apigateway create-rest-api --name 'My VPC Link Test'
```

Prendi nota del valore `id` di `RestApi` e del valore `rootResourceId` di `RestApi` nel risultato restituito. Questo valore è necessario per eseguire ulteriori operazioni sull'API.

Quindi, crea un'API con solo un metodo `GET` nella risorsa root (`/`) e integra il metodo con `VpcLink`.

3. Utilizza il comando put-method seguente per creare il metodo `GET /`:

```
aws apigateway put-method \
       --rest-api-id  abcdef123 \
       --resource-id skpp60rab7 \
       --http-method GET \
       --authorization-type "NONE"
```

Se non usi l'integrazione proxy con il `VpcLink`, devi configurare anche almeno una risposta del metodo per il codice di stato `200`. In questo caso si usa l'integrazione proxy.

4. Dopo aver creato il metodo `GET /`, viene configurata l'integrazione. Per un'integrazione privata, si utilizza il parametro `connection-id` per fornire l'ID del `VpcLink`. Puoi utilizzare una variabile di fase o inserire direttamente l'ID del `VpcLink`. Il parametro `uri` non viene usato per il routing delle richieste all'endpoint, ma viene usato per impostare l'intestazione `Host` per la convalida del certificato.

   Use the VPC link ID

   Utilizza il comando put-integration seguente per usare l'ID del `VpcLink` direttamente nell'integrazione:

```
aws apigateway put-integration \
    --rest-api-id abcdef123 \
    --resource-id skpp60rab7 \
    --uri 'http://my-vpclink-test-nlb-1234567890abcdef.us-east-2.amazonaws.com'
 \
    --http-method GET \
    --type HTTP_PROXY \
    --integration-http-method GET \
    --connection-type VPC_LINK \
    --connection-id gim7c3
```

   Use a stage variable

   Utilizza il comando put-integration seguente per usare una variabile di fase per fare riferimento all'ID del collegamento VPC. Quando implementi l'API in una fase, viene impostato l'ID del collegamento VPC.

```
aws apigateway put-integration \
    --rest-api-id abcdef123 \
    --resource-id skpp60rab7 \
    --uri 'http://my-vpclink-test-nlb-1234567890abcdef.us-east-2.amazonaws.com'
 \
    --http-method GET \
    --type HTTP_PROXY \
    --integration-http-method GET \
    --connection-type VPC_LINK \
    --connection-id "\${stageVariables.vpcLinkId}"
```

Assicurati di racchiudere tra virgolette doppie l'espressione della variabile di fase (`${stageVariables.vpcLinkId}`) e di aggiungere il carattere di escape davanti a `$`.

In qualsiasi momento, puoi anche aggiornare l'integrazione per modificare il valore `connection-id`. Utilizza il comando update-integration seguente per aggiornare l'integrazione:

```
 aws apigateway update-integration \
    --rest-api-id abcdef123 \
    --resource-id skpp60rab7 \
    --http-method GET \
    --patch-operations '[{"op":"replace","path":"/
connectionId","value":"${stageVariables.vpcLinkId}"}]'
```

Assicurati di usare un elenco JSON in formato stringa come valore del parametro `patch-operations`.

Poiché è stata utilizzata l'integrazione proxy privata, l'API è ora pronta per l'implementazione e i test.

5. Se hai utilizzato la variabile di fase per definire il valore `connection-id`, devi implementare l'API per testarla. Utilizza il comando create-deployment seguente per implementare l'API con una variabile di fase:

```
aws apigateway create-deployment \
    --rest-api-id abcdef123 \
    --stage-name test \
    --variables vpcLinkId=gim7c3
```

Per aggiornare la variabile di fase con un ID del `VpcLink` diverso, ad esempio *`asf9d7`*, utilizza il comando update-stage seguente:

```
aws apigateway update-stage \
    --rest-api-id abcdef123 \
    --stage-name test \
    --patch-operations op=replace,path='/variables/vpcLinkId',value='asf9d7'
```

Quando imposti come hardcoded la proprietà `connection-id` con il valore letterale ID di `VpcLink`, non è necessario implementare l'API per testarla. Usa il test-invoke-methodcomando per testare l'API prima che venga distribuita.

6. Utilizza il seguente comando per invocare l'API:

```
curl -X GET https://abcdef123.execute-api.us-east-2.amazonaws.com/test
```

In alternativa, puoi inserire l'URL di invocazione dell'API in un browser web per visualizzare il risultato.

Account API Gateway utilizzati per integrazioni private (legacy)

Il seguente account API Gateway specifico della regione IDs viene aggiunto automaticamente al tuo servizio endpoint VPC come quando `AllowedPrincipals` crei un. `VpcLink`

Region	ID account
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

Region	ID account
eu-west-3	061510835048
eu-central-1	474240146802
eu-central-2	166639821150
eu-north-1	394634713161
eu-south-1	753362059629
eu-south-2	359345898052
ap-northeast-1	969236854626
ap-northeast-2	020402002396
ap-northeast-3	360671645888
ap-southeast-1	195145609632
ap-southeast-2	798376113853
ap-southeast-3	652364314486
ap-southeast-4	849137399833
ap-south-1	507069717855
ap-south-2	644042651268
ap-east-1	174803364771
sa-east-1	287228555773
me-south-1	855739686837
me-central-1	614065512851

Integrazioni fittizie per REST APIs in API Gateway

Amazon API Gateway supporta le integrazioni fittizie per i metodi API. Questa caratteristica permette agli sviluppatori di API di generare risposte API da API Gateway direttamente, senza che sia necessario un back-end di integrazione. Uno sviluppatore di API può usare questa caratteristica per sbloccare team dipendenti che devono lavorare a un'API prima che lo sviluppo del progetto sia completato. Puoi anche usare questa caratteristica per allestire una pagina di destinazione per l'API, con una panoramica e comandi di navigazione per l'API. Per un esempio di pagina di destinazione di questo tipo, consulta la richiesta e la risposta di integrazione del metodo GET nella risorsa root dell'API di esempio illustrata in Tutorial: creazione di un'API REST mediante l'importazione di un esempio.

Uno sviluppatore di API può decidere in che modo API Gateway risponde alle richieste di integrazione fittizia. A tale scopo, è necessario configurare la richiesta e la risposta di integrazione del metodo per associare una risposta a un codice di stato specifico. Affinché un metodo con l'integrazione fittizia restituisca una risposta `200`, configura il modello di mappatura del corpo della richiesta di integrazione per restituire quanto segue.

```
{"statusCode": 200}
```

Configura una risposta di integrazione `200` con il modello di mappatura del corpo seguente, ad esempio:

```
{
    "statusCode": 200,
    "message": "Go ahead without me."
}
```

Analogamente, affinché il metodo restituisca, ad esempio, una risposta di errore `500`, configura il modello di mappatura del corpo della richiesta di integrazione per restituire quanto segue.

```
{"statusCode": 500}
```

Configura, ad esempio, una risposta di integrazione `500` con il modello di mappatura seguente:

```
{
    "statusCode": 500,
    "message": "The invoked method is not supported on the API resource."
}
```

In alternativa, puoi fare in modo che un metodo dell'integrazione fittizia restituisca la risposta di integrazione predefinita senza definire il modello di mappatura della richiesta di integrazione. La risposta di integrazione predefinita è quella con valore HTTP status regex (Regex stato HTTP) non definito. Assicurati che siano impostati i comportamenti di passthrough appropriati.

> **Note**
>
> Le integrazioni fittizie non sono concepite per supportare modelli di risposta di grandi dimensioni. Se ne hai bisogno per il caso d'uso specifico, dovresti valutare invece l'utilizzo di un'integrazione Lambda.

Usando un modello di mappatura della richiesta di integrazione puoi inserire la logica dell'applicazione per stabilire quale risposta di integrazione fittizia restituire in base a determinate condizioni. Puoi ad esempio usare un parametro di query `scope` nella richiesta in ingresso per determinare se restituire una risposta di esito positivo o di errore:

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

In questo modo, il metodo dell'integrazione fittizia consente alle chiamate interne di passare, mentre gli altri tipi di chiamate vengono rifiutati con una risposta di errore.

Questa sezione descrive come utilizzare la console API Gateway per abilitare l'integrazione fittizia per un metodo API.

Argomenti

- Abilitazione dell'integrazione fittizia tramite la console API Gateway

Abilitazione dell'integrazione fittizia tramite la console API Gateway

In Gateway API deve essere disponibile un metodo. Segui le istruzioni in Tutorial: creazione di una REST API con un'integrazione non proxy HTTP.

1. Seleziona una risorsa API e scegli Crea metodo.

   Per creare il metodo, procedi nel seguente modo:

   a. Per Tipo di metodo seleziona un metodo.
   b. Per Tipo di integrazione seleziona Fittizio.
   c. Scegli Crea metodo.
   d. Nella scheda Richiesta metodo scegli Modifica in Impostazioni della richiesta del metodo.
   e. Scegli Parametri della stringa di query URL. Scegli Aggiungi stringa di query e per Nome immetti **`scope`**. Questo parametro di query determina se il chiamante è interno o meno.
   f. Scegli Save (Salva).
2. Nella scheda Risposta metodo scegli Crea risposta, quindi procedi come indicato di seguito:

   a. Per Stato HTTP immetti **`500`**.
   b. Scegli Save (Salva).
3. Nella scheda Richiesta di integrazione seleziona Modifica per Impostazioni della richiesta di integrazione.
4. Scegli Modelli di mappatura e procedi come indicato di seguito:

   a. Scegliere Add mapping template (Aggiungi modello di mappatura).
   b. Per Tipo di contenuto inserisci **`application/json`**.
   c. Per Corpo del modello inserisci quanto segue:

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

   d. Scegli Save (Salva).
5. Nella scheda Risposta di integrazione scegli Modifica per Predefinito - Risposta.
6. Scegli Modelli di mappatura e procedi come indicato di seguito:

   a. Per Tipo di contenuto inserisci **`application/json`**.
   b. Per Corpo del modello inserisci quanto segue:

```
{
    "statusCode": 200,
    "message": "Go ahead without me"
}
```

c. Scegli Save (Salva).

7. Scegli Crea risposta.

   Per creare una risposta 500, procedi come descritto qui di seguito:

   a. Per HTTP status regex (Regex stato HTTP), immettere **`5\d{2}`**.

   b. Per Stato metodo risposta seleziona **`500`**.

   c. Scegli Save (Salva).

   d. Per 5\d{2} - Risposta scegli Modifica.

   e. Scegli Modelli di mappatura, quindi seleziona Aggiungi modello di mappatura.

   f. Per Tipo di contenuto inserisci **`application/json`**.

   g. Per Corpo del modello inserisci quanto segue:

```
{
    "statusCode": 500,
    "message": "The invoked method is not supported on the API resource."
}
```

   h. Scegli Save (Salva).

8. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda. Per testare l'integrazione fittizia, procedi come indicato di seguito:

   a. Immetti `scope=internal` in Stringhe di query. Scegli Test (Esegui test). Viene visualizzato il risultato del test:

```
Request: /?scope=internal
Status: 200
Latency: 26 ms
Response Body

{
  "statusCode": 200,
```

```
  "message": "Go ahead without me"
}

Response Headers

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

b. Immettere `scope=public` in `Query strings` oppure lasciare vuoto il campo. Scegli Test (Esegui test). Viene visualizzato il risultato del test:

```
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"}
```

Puoi anche restituire le intestazioni in una risposta di integrazione fittizia aggiungendo prima di tutto un'intestazione a una risposta del metodo e quindi configurando una mappatura dell'intestazione nella risposta di integrazione. È in questo modo che la console API Gateway abilita il supporto CORS restituendo le intestazioni richieste da CORS.

Convalida delle richieste per REST API in Gateway API

Puoi configurare API Gateway per l'esecuzione di una convalida di base di una richiesta API prima di procedere con la richiesta di integrazione. Quando la convalida fallisce, API Gateway fallisce immediatamente la richiesta, restituisce una risposta di errore 400 al chiamante e pubblica i risultati della convalida in Logs. CloudWatch Questo comportamento riduce le chiamate non necessarie al back-end. Aspetto ancora più importante, ti permette di concentrarti sulle attività di convalida specifiche dell'applicazione. È possibile convalidare il corpo di una richiesta verificando che i

parametri obbligatori della richiesta siano validi e diversi da null oppure specificando uno schema di modello per una convalida dei dati più complessa.

Argomenti

- Panoramica della convalida di base delle richieste in API Gateway
- Modelli di dati per REST API
- Configurazione della convalida di base delle richieste in API Gateway
- AWS CloudFormation modello di un'API di esempio con convalida delle richieste di base

Panoramica della convalida di base delle richieste in API Gateway

Gateway Amazon API può eseguire la convalida di base delle richieste, in modo che sia possibile concentrarsi sulla convalida specifica dell'app nel back-end. Per la convalida, Gateway Amazon API verifica una o entrambe le condizioni seguenti:

- I parametri della richiesta obbligatori nell'URI, nella stringa di query e nelle intestazioni di una richiesta in entrata sono inclusi e non sono vuoti. Gateway API verifica solo l'esistenza di un parametro e non controlla il tipo o il formato.
- Il payload della richiesta applicabile soddisfa la richiesta configurata dello schema JSON del metodo per un tipo di contenuto specifico. Se non viene trovato alcun tipo di contenuto corrispondente, la convalida della richiesta non viene eseguita. Per utilizzare lo stesso modello indipendentemente dal tipo di contenuto, impostare il tipo di contenuto per il modello di dati su `$default`.

Per attivare la convalida di base, è necessario specificare regole di convalida in un validatore di richieste, aggiungere il validatore alla mappa di validatori di richieste dell'API e assegnare il validatore a singoli metodi API.

> **Note**
>
> La convalida del corpo della richiesta e Comportamento della richiesta di metodo per i payload senza modelli di mappatura per REST API in Gateway API sono due argomenti separati. Quando il payload di una richiesta non dispone di uno schema di modello corrispondente, puoi scegliere di eseguire il transito o il blocco del payload originale. Per ulteriori informazioni, consulta Comportamento della richiesta di metodo per i payload senza modelli di mappatura per REST API in Gateway API.

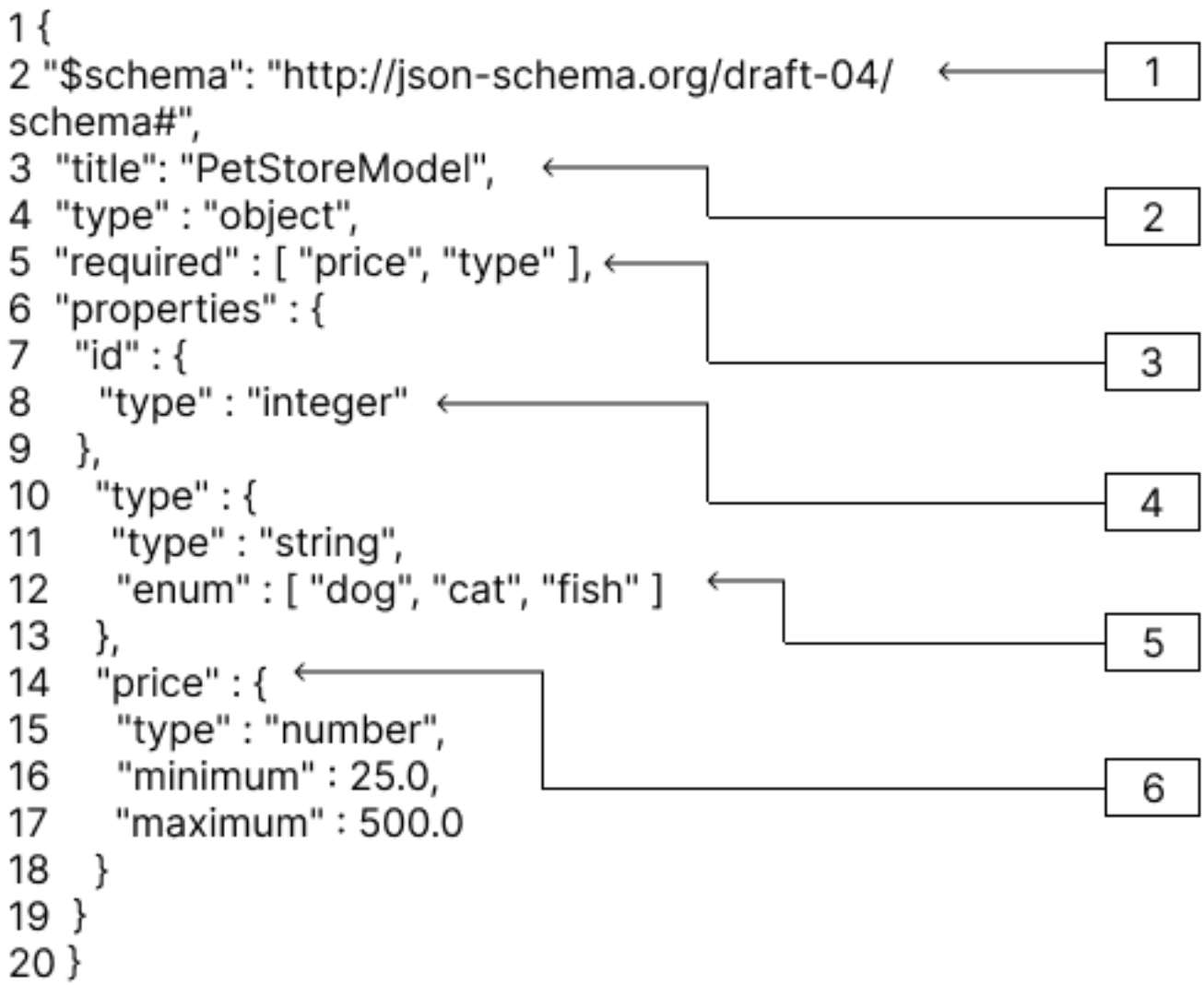
Modelli di dati per REST API

In API Gateway, un modello definisce la struttura dei dati di un payload. In Gateway Amazon API i modelli sono definiti utilizzando lo schema JSON bozza 4. Il seguente oggetto JSON è un esempio di dati nell'esempio di Pet Store.

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

I dati contengono i valori `id`, `type` e `price` dell'animale domestico. Un modello di questi dati consente di:

- Usare la convalida di base delle richieste.
- Creare modelli di mappatura per la trasformazione dei dati.
- Creare un tipo di dati definito dall'utente (UDT) quando viene generato un SDK.



In questo modello:

1. L'oggetto `$schema` rappresenta un identificatore valido della versione dello schema JSON. Questo schema è lo schema JSON bozza v4.
2. L'oggetto `title` è un identificatore in formato leggibile del modello. Questo titolo è `PetStoreModel`.
3. La parola chiave di convalida `required` richiede `type` e `price` per la convalida di base della richiesta.
4. La variabile `properties` del modello è `id`, `type` e`price`. Ogni oggetto ha proprietà che vengono descritte nel modello.

5. L'oggetto `type` può avere solo i valori `dog`, `cat` o `fish`.
6. L'oggetto `price` è un numero ed è vincolato con un valore `minimum` di 25 e un valore `maximum` di 500.

PetStore modello

```
1 {
2 "$schema": "http://json-schema.org/draft-04/schema#",
3  "title": "PetStoreModel",
4  "type" : "object",
5  "required" : [ "price", "type" ],
6  "properties" : {
7    "id" : {
8      "type" : "integer"
9    },
10    "type" : {
11      "type" : "string",
12      "enum" : [ "dog", "cat", "fish" ]
13    },
14    "price" : {
15      "type" : "number",
16      "minimum" : 25.0,
17      "maximum" : 500.0
18    }
19  }
20 }
```

In questo modello:

1. Alla riga 2, l'oggetto `$schema` rappresenta un identificatore valido della versione dello schema JSON. Questo schema è lo schema JSON bozza v4.
2. Alla riga 3, l'oggetto `title` è un identificatore in formato leggibile del modello. Questo titolo è `PetStoreModel`.
3. Alla riga 5, la parola chiave di convalida `required` richiede `type` e `price` per la convalida di base della richiesta.
4. Alle righe da 6 a 17, la variabile `properties` del modello è `id`. `type` e `price`. Ogni oggetto ha proprietà che vengono descritte nel modello.
5. Alla riga 12, l'oggetto `type` può avere solo i valori `dog`, `cat` o `fish`.

6. Alle righe da 14 a 17, l'oggetto `price` è un numero ed è vincolato con un valore `minimum` di 25 e un valore `maximum` di 500.

Creazione di modelli più complessi

È possibile utilizzare la primitiva `$ref` per creare definizioni riutilizzabili per modelli più lunghi. Ad esempio, è possibile creare una definizione chiamata `Price` nella sezione `definitions` che descrive l'oggetto `price`. Il valore di `$ref` è la definizione `Price`.

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "PetStoreModelReUsableRef",
  "required" : ["price", "type" ],
  "type" : "object",
  "properties" : {
    "id" : {
      "type" : "integer"
    },
    "type" : {
      "type" : "string",
      "enum" : [ "dog", "cat", "fish" ]
    },
    "price" : {
        "$ref": "#/definitions/Price"
    }
  },
  "definitions" : {
      "Price": {
        "type" : "number",
        "minimum" : 25.0,
        "maximum" : 500.0
            }
      }
}
```

È inoltre possibile fare riferimento a un altro schema del modello definito in un file di modello esterno. Impostare il valore della proprietà `$ref` sulla posizione del modello. Nell'esempio seguente, il modello `Price` è definito nel modello `PetStorePrice` nell'API `a1234`.

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
```

```
  "title" : "PetStorePrice",
  "type": "number",
  "minimum": 25,
  "maximum": 500
}
```

Il modello più lungo può fare riferimento al modello `PetStorePrice`.

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "PetStoreModelReusableRefAPI",
  "required" : [ "price", "type" ],
  "type" : "object",
  "properties" : {
    "id" : {
      "type" : "integer"
    },
    "type" : {
      "type" : "string",
      "enum" : [ "dog", "cat", "fish" ]
    },
    "price" : {
        "$ref": "https://apigateway.amazonaws.com/restapis/a1234/models/PetStorePrice"
    }
  }
}
```

Utilizzo di modelli di dati di output

In caso di trasformazione dei dati, è possibile definire un modello di payload nella risposta dell'integrazione. È possibile utilizzare un modello di payload quando si genera un SDK. Per linguaggi fortemente tipizzati, come Java o Swift Objective-C, l'oggetto corrisponde a un tipo di dati definito dall'utente (UDT). Gateway Amazon API crea un tipo di dati definito dall'utente se viene specificato con un modello di dati durante la generazione di un SDK. Per ulteriori informazioni sulle trasformazioni dei dati, consultare Trasformazioni dei modelli di mappatura per REST API in Gateway API.

L'esempio seguente indica i dati di output di una risposta di integrazione.

```
{
[
```

```
  {
    "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
  }
]
}
```

L'esempio seguente indica un modello di payload che descrive i dati di output.

```
{
"$schema": "http://json-schema.org/draft-04/schema#",
  "title": "PetStoreOutputModel",
  "type" : "object",
  "required" : [ "description", "askingPrice" ],
  "properties" : {
    "description" : {
      "type" : "string"
    },
    "askingPrice" : {
      "type" : "number",
      "minimum" : 25.0,
      "maximum" : 500.0
    }
  }
}
```

Con questo modello, è possibile eseguire una chiamata a un SDK per recuperare i valori delle proprietà `description`, `askingPrice` leggendo le proprietà `PetStoreOutputModel[i].description` e `PetStoreOutputModel[i].askingPrice`. Se non viene fornito alcun modello, API Gateway utilizzerà il modello vuoto per creare un UDT predefinito.

Fasi successive

- Questa sezione fornisce risorse che è possibile utilizzare per acquisire maggiori conoscenze sui concetti trattati in questo argomento.

  È possibile seguire i tutorial relativi alla convalida delle richieste:

  - Configurazione della convalida delle richieste tramite la console Gateway Amazon API
  - Imposta la convalida delle richieste di base utilizzando il AWS CLI
  - Configurazione della convalida di base delle richieste utilizzando una definizione OpenAPI
- Per ulteriori informazioni sulla trasformazione dei dati e sui modelli di mappatura, consultare Trasformazioni dei modelli di mappatura per REST API in Gateway API.

Configurazione della convalida di base delle richieste in API Gateway

Questa sezione mostra come configurare la convalida delle richieste per API Gateway utilizzando la console e una definizione OpenAPI. AWS CLI

Argomenti

- Configurazione della convalida delle richieste tramite la console Gateway Amazon API
- Imposta la convalida delle richieste di base utilizzando il AWS CLI
- Configurazione della convalida di base delle richieste utilizzando una definizione OpenAPI

Configurazione della convalida delle richieste tramite la console Gateway Amazon API

È possibile utilizzare la console Gateway Amazon API per convalidare una richiesta selezionando uno dei tre validatori per una richiesta API:

- Convalida del corpo.
- Convalida dei parametri e delle intestazioni delle stringhe di query.
- Convalida del corpo, dei parametri delle stringhe di query e delle intestazioni.

Quando applichi uno dei validatori a un metodo API, la console API Gateway aggiunge il validatore alla mappa dell'RequestValidatorsAPI.

Per seguire questo tutorial, utilizzerai un CloudFormation modello per creare un'API API Gateway API incompleta. Questa API ha una risorsa `/validator` con i metodi `GET` e `POST`. Entrambi i metodi

sono integrati con l'endpoint HTTP `http://petstore-demo-endpoint.execute-api.com/petstore/pets`. Configurare due tipi di convalida delle richieste:

- Nel metodo `GET`, verrà configurata la convalida delle richieste per i parametri della stringa di query URL.
- Nel metodo `POST`, verrà configurata la convalida delle richieste per il corpo della richiesta.

Ciò consentirà solo a determinate chiamate API di eseguire il transito all'API.

Scarica e decomprimi il modello di creazione dell'app per CloudFormation. Verrà utilizzato questo modello per creare un'API incompleta. Il resto della procedura verrà completato nella console Gateway Amazon API.

Per creare un CloudFormation stack

1. Apri la CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation.
2. Scegliere Create stack (Crea stack), quindi With new resources (standard) (Con nuove risorse (standard)).
3. In Specificare modello, scegliere Carica un file modello.
4. Selezionare il modello scaricato.
5. Scegli Next (Successivo).
6. Per Stack name (Nome stack), inserire **`request-validation-tutorial-console`**, quindi scegliere Next (Avanti).
7. Per Configure stack options (Configura opzioni di stack), scegliere Next (Successivo).
8. Per quanto riguarda le funzionalità, riconosci che CloudFormation puoi creare risorse IAM nel tuo account.
9. Scegli Avanti, quindi scegli Invia.

CloudFormation fornisce le risorse specificate nel modello. Per completare il provisioning delle risorse, potrebbero essere necessari alcuni minuti. Quando lo stato del tuo CloudFormation stack è CREATE_COMPLETE, sei pronto per passare alla fase successiva.

Selezione dell'API appena creata

1. Selezionare lo stack **`request-validation-tutorial-console`** appena creato.
2. Scegliere Resources (Risorse).

3. In ID fisico, scegliere l'API. Questo link consente di venire reindirizzati alla console di Gateway Amazon API.

Prima di modificare i metodi `GET` e `POST`, è necessario creare un modello.

Creazione di un modello

1. È necessario che un modello utilizzi la convalida della richiesta nel corpo di una richiesta in arrivo. Per creare un modello seleziona Modelli nel pannello di navigazione principale.
2. Scegli Crea modello.
3. In Nome, inserisci **`PetStoreModel`**.
4. In Tipo di contenuto, inserire **`application/json`**. Se non viene trovato alcun tipo di contenuto corrispondente, la convalida della richiesta non viene eseguita. Per utilizzare lo stesso modello indipendentemente dal tipo di contenuti, inserisci **`$default`**.
5. Per Descrizione immetti **`My PetStore Model`** come descrizione del modello.
6. Per Schema modello incolla il seguente modello nell'editor di codice e scegli Crea.

```
{
  "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
    }
  }
}
```

Per ulteriori informazioni sul modello, consulta Modelli di dati per REST API.

Per configurare la convalida delle richieste per un metodo GET

1. Nel pannello di navigazione principale scegli Risorse, quindi seleziona il metodo GET.
2. Nella scheda Richiesta metodo, in Impostazioni richiesta metodo, scegli Modifica.
3. Per Validatore richiesta seleziona Convalida parametri di stringa query e intestazioni.
4. In Parametri della stringa di query URL ed effettua le seguenti operazioni:
   a. Scegliere Add query string (Aggiungi stringa di query).
   b. In Nome, inserisci **`petType`**.
   c. Attiva Campo obbligatorio.
   d. Mantieni disattivata l'opzione Caching.
5. Scegli Save (Salva).
6. Nella scheda Richiesta di integrazione scegli Modifica in Impostazioni della richiesta di integrazione.
7. In Parametri della stringa di query URL ed effettua le seguenti operazioni:
   a. Scegliere Add query string (Aggiungi stringa di query).
   b. In Nome, inserisci **`petType`**.
   c. In Mappato da, inserire **`method.request.querystring.petType`**. Questa operazione associa **`petType`** al tipo di animale domestico.

      Per ulteriori informazioni sulla mappatura dei dati, consultare il tutorial relativo alla mappatura dei dati.
   d. Mantieni disattivata l'opzione Caching.
8. Scegli Save (Salva).

Per testare la convalida delle richieste per il metodo GET

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. In Stringhe di query immetti **`petType=dog`** e scegli Test.
3. Il test del metodo restituirà `200 OK` e un elenco di cani.

Per informazioni su come trasformare questi dati di output, consultare il tutorial sulla mappatura dei dati.

4. Rimuovere **petType=dog** e scegliere Test.
5. Il test del metodo restituirà un errore `400` con il seguente messaggio di errore:

```
{
  "message": "Missing required request parameters: [petType]"
}
```

Per configurare la convalida delle richieste per il metodo POST

1. Nel pannello di navigazione principale scegli Risorse, quindi seleziona il metodo POST.
2. Nella scheda Richiesta metodo, in Impostazioni richiesta metodo, scegli Modifica.
3. Per Validatore richiesta seleziona Convalida corpo.
4. In Corpo della richiesta scegli Aggiungi modello.
5. Per Tipo di contenuto inserisci **application/json**. Se non viene trovato alcun tipo di contenuto corrispondente, la convalida della richiesta non viene eseguita. Per utilizzare lo stesso modello indipendentemente dal tipo di contenuti, inserisci `$default`.

   Per Modello, selezionare PetStoreModel.
6. Scegli Save (Salva).

Per testare la convalida della richiesta per un metodo **POST**

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. In Corpo della richiesta incolla quanto segue nell'editor di codice:

```
{
  "id": 2,
  "name": "Bella",
  "type": "dog",
  "price": 400
}
```

Scegli Test (Esegui test).

3. Il test del metodo restituirà `200 OK` e un messaggio di operazione riuscita.
4. In Corpo della richiesta incolla quanto segue nell'editor di codice:

```
{
  "id": 2,
  "name": "Bella",
  "type": "dog",
  "price": 4000
}
```

Scegli Test (Esegui test).

5. Il test del metodo restituirà un errore `400` con il seguente messaggio di errore:

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

Nella parte inferiore dei log del test viene restituito il motivo del corpo della richiesta non valido. In questo caso, il prezzo dell'animale era maggiore del valore massimo specificato nel modello.

Per eliminare un CloudFormation stack

1. Apri la CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation.
2. Seleziona il tuo CloudFormation stack.
3. Scegli Elimina e conferma la tua scelta.

Fasi successive

- Per informazioni su come trasformare i dati di output ed eseguire ulteriori mappature dei dati, consultare il tutorial sulla mappatura dei dati.
- Eseguire il tutorial Configurazione della convalida di base delle richieste in AWS CLI per eseguire passaggi simili utilizzando la AWS CLI.

Imposta la convalida delle richieste di base utilizzando il AWS CLI

È possibile creare un validatore per configurare la convalida della richiesta utilizzando la AWS CLI. Per seguire questo tutorial, utilizzerai un CloudFormation modello per creare un'API API Gateway API incompleta.

> **Note**
>
> Questo non è lo stesso CloudFormation modello del tutorial della console.

Utilizzando una risorsa `/validator` precedentemente esposta, verranno creati i metodi `GET` e `POST`. Entrambi i metodi saranno integrati con l'endpoint HTTP `http://petstore-demo-endpoint.execute-api.com/petstore/pets`. Verranno configurate le due convalide delle richieste seguenti:

- Nel metodo `GET`, verrà creato un validatore `params-only` per convalidare i parametri della stringa di query URL.
- Nel metodo `POST`, verrà creato un validatore `body-only` per convalidare il corpo della richiesta.

Ciò consentirà solo a determinate chiamate API di eseguire il transito all'API.

Per creare un CloudFormation stack

Scarica e decomprimi il modello di creazione dell'app per CloudFormation.

Per completare il tutorial seguente, è necessario disporre della AWS Command Line Interface (AWS CLI) versione 2.

Per i comandi lunghi viene utilizzato un carattere di escape (\) per dividere un comando su più righe.

> **Note**
>
> In Windows, alcuni comandi della CLI Bash utilizzati comunemente (ad esempio, `zip`) non sono supportati dai terminali integrati del sistema operativo. Per ottenere una Windows-integrated versione di Ubuntu e Bash, installa il sottosistema Windows per Linux. I comandi della CLI di esempio in questa guida utilizzano la formattazione Linux. Se si utilizza la CLI di Windows, i comandi che includono documenti JSON in linea dovranno essere riformattati.

1. Usa il seguente comando per creare lo stack. CloudFormation

```
aws cloudformation create-stack --stack-name request-validation-tutorial-cli
 --template-body file://request-validation-tutorial-cli.zip --capabilities
 CAPABILITY_NAMED_IAM
```

2. CloudFormation fornisce le risorse specificate nel modello. Per completare il provisioning delle risorse, potrebbero essere necessari alcuni minuti. Usa il comando seguente per vedere lo stato del tuo CloudFormation stack.

```
aws cloudformation describe-stacks --stack-name request-validation-tutorial-cli
```

3. Quando lo stato dello CloudFormation stack è`StackStatus: "CREATE_COMPLETE"`, usa il seguente comando per recuperare i valori di output pertinenti per i passaggi futuri.

```
 aws cloudformation describe-stacks --stack-name request-validation-tutorial-cli
 --query "Stacks[*].Outputs[*].{OutputKey: OutputKey, OutputValue: OutputValue,
 Description: Description}"
```

   Di seguito sono riportati i valori di output:

   - ApiId, che è l'ID dell'API. Per questo tutorial, l'ID API è `abc123`.
   - ResourceId, che è l'ID della risorsa di validazione in cui sono esposti `POST` i metodi `GET` and. Per questo tutorial, l'ID risorsa è `efg456`.

Creazione dei validatori della richiesta e importazione di un modello

1. Per utilizzare la convalida della richiesta con la AWS CLI, è necessario un validatore. Usare il seguente comando per creare un validatore che convalidi solo i parametri della richiesta.

```
aws apigateway create-request-validator --rest-api-id abc123 \
      --no-validate-request-body \
      --validate-request-parameters \
      --name params-only
```

   Annotare l'ID del validatore `params-only`.

2. Usare il seguente comando per creare un validatore che convalidi solo il corpo della richiesta.

```
aws apigateway create-request-validator --rest-api-id abc123 \
```

```
      --validate-request-body \
      --no-validate-request-parameters \
      --name body-only
```

Annotare l'ID del validatore `body-only`.

3. È necessario che un modello utilizzi la convalida della richiesta nel corpo di una richiesta in arrivo. Utilizzare il comando seguente per importare un modello.

```
aws apigateway create-model --rest-api-id abc123 --name PetStoreModel --description
 'My PetStore Model' --content-type 'application/json' --schema '{"type":
 "object", "required" : [ "name", "price", "type" ], "properties" : { "id" :
 {"type" : "integer"},"type" : {"type" : "string", "enum" : [ "dog", "cat",
 "fish" ]},"name" : { "type" : "string"},"price" : {"type" : "number","minimum" :
 25.0, "maximum" : 500.0}}}}'
```

Se non viene trovato alcun tipo di contenuto corrispondente, la convalida della richiesta non viene eseguita. Per utilizzare lo stesso modello indipendentemente dal tipo di contenuto, specifica `$default` come chiave.

Per creare i metodi **`GET`** e **`POST`**

1. Utilizzare il seguente comando per aggiungere il metodo HTTP `GET` per la risorsa `/validate`. Questo comando crea il metodo `GET`, aggiunge il validatore `params-only` e imposta la stringa di query `petType` come richiesto.

```
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"
```

Utilizzare il seguente comando per aggiungere il metodo HTTP `POST` per la risorsa `/validate`. Questo comando crea il metodo `POST`, aggiunge il validatore `body-only` e collega il modello al validatore solo del corpo.

```
aws apigateway put-method --rest-api-id abc123 \
       --resource-id efg456 \
       --http-method POST \
```

```
        --authorization-type "NONE" \
        --request-validator-id bbb222 \
        --request-models 'application/json'=PetStoreModel
```

2. Utilizzare il comando seguente per configurare la risposta `200 OK` del metodo `GET /validate`.

```
aws apigateway put-method-response --rest-api-id abc123  \
            --resource-id efg456 \
            --http-method GET \
            --status-code 200
```

   Utilizzare il comando seguente per configurare la risposta `200 OK` del metodo `POST /validate`.

```
aws apigateway put-method-response --rest-api-id abc123  \
            --resource-id efg456 \
            --http-method POST \
            --status-code 200
```

3. Utilizzare il comando seguente per configurare una variabile `Integration` con un endpoint HTTP specificato per il metodo `GET /validation`.

```
aws apigateway put-integration --rest-api-id abc123  \
            --resource-id efg456 \
            --http-method GET \
            --type HTTP \
            --integration-http-method GET \
            --request-parameters '{"integration.request.querystring.type" :
 "method.request.querystring.petType"}' \
            --uri 'http://petstore-demo-endpoint.execute-api.com/petstore/pets'
```

   Utilizzare il comando seguente per configurare una variabile `Integration` con un endpoint HTTP specificato per il metodo `POST /validation`.

```
aws apigateway put-integration --rest-api-id abc123  \
              --resource-id efg456 \
              --http-method POST \
              --type HTTP \
              --integration-http-method GET \
              --uri 'http://petstore-demo-endpoint.execute-api.com/petstore/pets'
```

4. Utilizzare il comando seguente per configurare una risposta di integrazione per il metodo `GET /validation`.

```
aws apigateway put-integration-response --rest-api-id abc123 \
              --resource-id efg456\
              --http-method GET \
              --status-code 200 \
              --selection-pattern ""
```

   Utilizzare il comando seguente per configurare una risposta di integrazione per il metodo `POST /validation`.

```
aws apigateway put-integration-response --rest-api-id abc123 \
            --resource-id efg456 \
            --http-method POST \
            --status-code 200 \
            --selection-pattern ""
```

Per testare l'API

1. Per testare il metodo `GET`, che eseguirà la convalida della richiesta per le stringhe di query, usare il seguente comando:

```
aws apigateway test-invoke-method --rest-api-id abc123 \
            --resource-id efg456 \
            --http-method GET \
            --path-with-query-string '/validate?petType=dog'
```

   Il risultato restituirà `200 OK` e l'elenco dei cani.

2. Utilizzare il seguente comando per eseguire il test senza includere la stringa di query `petType`.

```
aws apigateway test-invoke-method --rest-api-id abc123 \
            --resource-id efg456 \
            --http-method GET
```

   Il risultato restituirà un errore `400`.

3. Per testare il metodo `POST`, che eseguirà la convalida della richiesta per il corpo della richiesta, usare il seguente comando:

```
aws apigateway test-invoke-method --rest-api-id abc123 \
           --resource-id efg456 \
           --http-method POST \
           --body '{"id": 1, "name": "bella", "type": "dog", "price" : 400 }'
```

Il risultato restituirà `200 OK` e un messaggio di operazione riuscita.

4. Usare il seguente comando per testare l'utilizzo di un corpo non valido.

```
aws apigateway test-invoke-method --rest-api-id abc123 \
            --resource-id efg456 \
            --http-method POST \
            --body '{"id": 1, "name": "bella", "type": "dog", "price" : 1000 }'
```

Il risultato restituirà un errore `400`, perché il prezzo del cane supera il prezzo massimo definito dal modello.

Per eliminare un CloudFormation stack

- Usa il seguente comando per eliminare CloudFormation le tue risorse.

```
aws cloudformation delete-stack  --stack-name request-validation-tutorial-cli
```

Configurazione della convalida di base delle richieste utilizzando una definizione OpenAPI

È possibile dichiarare un validatore di richiesta a livello di API specificando un set di oggetti Oggetto x-amazon-apigateway-request-validators.requestValidator nella mappa Oggetto x-amazon-apigateway-request-validators per selezionare la parte della richiesta che verrà convalidata. Nell'esempio di definizione OpenAPI, sono presenti due validatori:

- Il validatore `all`, che convalida sia il corpo, mediante il modello di dati `RequestBodyModel`, sia i parametri.

  In base al modello di dati `RequestBodyModel`, l'oggetto JSON di input deve contenere le proprietà `name`, `type` e `price`. La proprietà `name` può essere qualsiasi stringa, `type` deve essere uno dei campi dell'enumerazione specificati (`["dog", "cat", "fish"]`) e `price` deve essere compreso tra 25 e 500. Il parametro `id` non è obbligatorio.
- Il validatore `param-only`, che convalida solo i parametri.

Per attivare un validatore di richieste in tutti i metodi di un'API, specificare una proprietà Proprietà x-amazon-apigateway-request-validator a livello di API della definizione OpenAPI. Nell'esempio di definizione OpenAPI, il validatore `all` viene utilizzato su tutti i metodi API, salvo diversamente definito. Quando si utilizza un modello per convalidare il corpo, se non viene trovato alcun tipo di contenuto corrispondente, la convalida della richiesta non viene eseguita. Per utilizzare lo stesso modello indipendentemente dal tipo di contenuto, specifica `$default` come chiave.

Per attivare un validatore di richieste in un singolo metodo, specificare la proprietà `x-amazon-apigateway-request-validator` a livello di metodo. Nell'esempio di definizione OpenAPI, il validatore `param-only` sovrascrive il validatore `all` nel metodo `GET`.

Per importare l'esempio OpenAPI in Gateway Amazon API, consultare le seguenti istruzioni per eseguire una Importazione di un'API regionale in Gateway API o una Importazione di un'API ottimizzata per l'edge in API Gateway.

OpenAPI 3.0

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "ReqValidators Sample",
    "version" : "1.0.0"
  },
  "servers" : [ {
    "url" : "/{basePath}",
    "variables" : {
      "basePath" : {
        "default" : "/v1"
      }
    }
  } ],
  "paths" : {
    "/validation" : {
      "get" : {
        "parameters" : [ {
          "name" : "q1",
          "in" : "query",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
```

```
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "headers" : {
              "test-method-response-header" : {
                "schema" : {
                  "type" : "string"
                }
              }
            },
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/ArrayOfError"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-request-validator" : "params-only",
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
          "responses" : {
            "default" : {
              "statusCode" : "400",
              "responseParameters" : {
                "method.response.header.test-method-response-header" : "'static
value'"
              },
              "responseTemplates" : {
                "application/xml" : "xml 400 response template",
                "application/json" : "json 400 response template"
              }
            },
            "2\\d{2}" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.querystring.type" : "method.request.querystring.q1"
          },
          "passthroughBehavior" : "when_no_match",
```

```
        "type" : "http"
      }
    },
    "post" : {
      "parameters" : [ {
        "name" : "h1",
        "in" : "header",
        "required" : true,
        "schema" : {
          "type" : "string"
        }
      } ],
      "requestBody" : {
        "content" : {
          "application/json" : {
            "schema" : {
              "$ref" : "#/components/schemas/RequestBodyModel"
            }
          }
        },
        "required" : true
      },
      "responses" : {
        "200" : {
          "description" : "200 response",
          "headers" : {
            "test-method-response-header" : {
              "schema" : {
                "type" : "string"
              }
            }
          },
          "content" : {
            "application/json" : {
              "schema" : {
                "$ref" : "#/components/schemas/ArrayOfError"
              }
            }
          }
        }
      },
      "x-amazon-apigateway-request-validator" : "all",
      "x-amazon-apigateway-integration" : {
        "httpMethod" : "POST",
```

```
        "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
        "responses" : {
          "default" : {
            "statusCode" : "400",
            "responseParameters" : {
              "method.response.header.test-method-response-header" : "'static
value'"
            },
            "responseTemplates" : {
              "application/xml" : "xml 400 response template",
              "application/json" : "json 400 response template"
            }
          },
          "2\\d{2}" : {
            "statusCode" : "200"
          }
        },
        "requestParameters" : {
          "integration.request.header.custom_h1" : "method.request.header.h1"
        },
        "passthroughBehavior" : "when_no_match",
        "type" : "http"
      }
    }
  }
},
"components" : {
  "schemas" : {
    "RequestBodyModel" : {
      "required" : [ "name", "price", "type" ],
      "type" : "object",
      "properties" : {
        "id" : {
          "type" : "integer"
        },
        "type" : {
          "type" : "string",
          "enum" : [ "dog", "cat", "fish" ]
        },
        "name" : {
          "type" : "string"
        },
        "price" : {
          "maximum" : 500.0,
```

```
            "minimum" : 25.0,
            "type" : "number"
          }
        }
      },
      "ArrayOfError" : {
        "type" : "array",
        "items" : {
          "$ref" : "#/components/schemas/Error"
        }
      },
      "Error" : {
        "type" : "object"
      }
    }
  },
  "x-amazon-apigateway-request-validators" : {
    "all" : {
      "validateRequestParameters" : true,
      "validateRequestBody" : true
    },
    "params-only" : {
      "validateRequestParameters" : true,
      "validateRequestBody" : false
    }
  }
}
```

OpenAPI 2.0

```
{
  "swagger" : "2.0",
  "info" : {
    "version" : "1.0.0",
    "title" : "ReqValidators Sample"
  },
  "basePath" : "/v1",
  "schemes" : [ "https" ],
  "paths" : {
    "/validation" : {
      "get" : {
        "produces" : [ "application/json", "application/xml" ],
        "parameters" : [ {
```

```
          "name" : "q1",
          "in" : "query",
          "required" : true,
          "type" : "string"
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "schema" : {
              "$ref" : "#/definitions/ArrayOfError"
            },
            "headers" : {
              "test-method-response-header" : {
                "type" : "string"
              }
            }
          }
        },
        "x-amazon-apigateway-request-validator" : "params-only",
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
          "responses" : {
            "default" : {
              "statusCode" : "400",
              "responseParameters" : {
                "method.response.header.test-method-response-header" : "'static
value'"
              },
              "responseTemplates" : {
                "application/xml" : "xml 400 response template",
                "application/json" : "json 400 response template"
              }
            },
            "2\\d{2}" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.querystring.type" : "method.request.querystring.q1"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "http"
        }
```

```
    },
    "post" : {
      "consumes" : [ "application/json" ],
      "produces" : [ "application/json", "application/xml" ],
      "parameters" : [ {
        "name" : "h1",
        "in" : "header",
        "required" : true,
        "type" : "string"
      }, {
        "in" : "body",
        "name" : "RequestBodyModel",
        "required" : true,
        "schema" : {
          "$ref" : "#/definitions/RequestBodyModel"
        }
      } ],
      "responses" : {
        "200" : {
          "description" : "200 response",
          "schema" : {
            "$ref" : "#/definitions/ArrayOfError"
          },
          "headers" : {
            "test-method-response-header" : {
              "type" : "string"
            }
          }
        }
      },
      "x-amazon-apigateway-request-validator" : "all",
      "x-amazon-apigateway-integration" : {
        "httpMethod" : "POST",
        "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
        "responses" : {
          "default" : {
            "statusCode" : "400",
            "responseParameters" : {
              "method.response.header.test-method-response-header" : "'static
value'"
            },
            "responseTemplates" : {
              "application/xml" : "xml 400 response template",
              "application/json" : "json 400 response template"
```

```
            }
          },
          "2\\d{2}" : {
            "statusCode" : "200"
          }
        },
        "requestParameters" : {
          "integration.request.header.custom_h1" : "method.request.header.h1"
        },
        "passthroughBehavior" : "when_no_match",
        "type" : "http"
      }
    }
  }
},
"definitions" : {
  "RequestBodyModel" : {
    "type" : "object",
    "required" : [ "name", "price", "type" ],
    "properties" : {
      "id" : {
        "type" : "integer"
      },
      "type" : {
        "type" : "string",
        "enum" : [ "dog", "cat", "fish" ]
      },
      "name" : {
        "type" : "string"
      },
      "price" : {
        "type" : "number",
        "minimum" : 25.0,
        "maximum" : 500.0
      }
    }
  },
  "ArrayOfError" : {
    "type" : "array",
    "items" : {
      "$ref" : "#/definitions/Error"
    }
  },
  "Error" : {
```

```
      "type" : "object"
    }
  },
  "x-amazon-apigateway-request-validators" : {
    "all" : {
      "validateRequestParameters" : true,
      "validateRequestBody" : true
    },
    "params-only" : {
      "validateRequestParameters" : true,
      "validateRequestBody" : false
    }
  }
}
```

AWS CloudFormation modello di un'API di esempio con convalida delle richieste di base

La seguente definizione di modello di CloudFormation esempio definisce un'API di esempio con la convalida della richiesta abilitata. L'API PetStore è un subset dell'API . L'API espone un metodo `POST` per aggiungere un animale domestico alla raccolta `pets` e un metodo `GET` per l'esecuzione di query relative agli animali domestici in base a un tipo specificato.

Sono dichiarati due validatori di richiesta:

`GETValidator`

Questo validatore è abilitato nel metodo `GET`. Permette a Gateway API di verificare che il parametro di query obbligatorio (`q1`) sia incluso e non vuoto nella richiesta in entrata.

`POSTValidator`

Questo validatore è abilitato nel metodo `POST`. Consente a Gateway API di verificare che il formato di payload della richiesta sia conforme all'impostazione `RequestBodyModel` specificata quando il tipo di contenuto è `application/json`. Se non viene trovato alcun tipo di contenuto corrispondente, la convalida della richiesta non viene eseguita. Per utilizzare lo stesso modello indipendentemente dal tipo di contenuto, specifica `$default`. `RequestBodyModel` contiene un modello aggiuntivo, `RequestBodyModelId`, per definire l'ID dell'animale domestico.

```
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
```

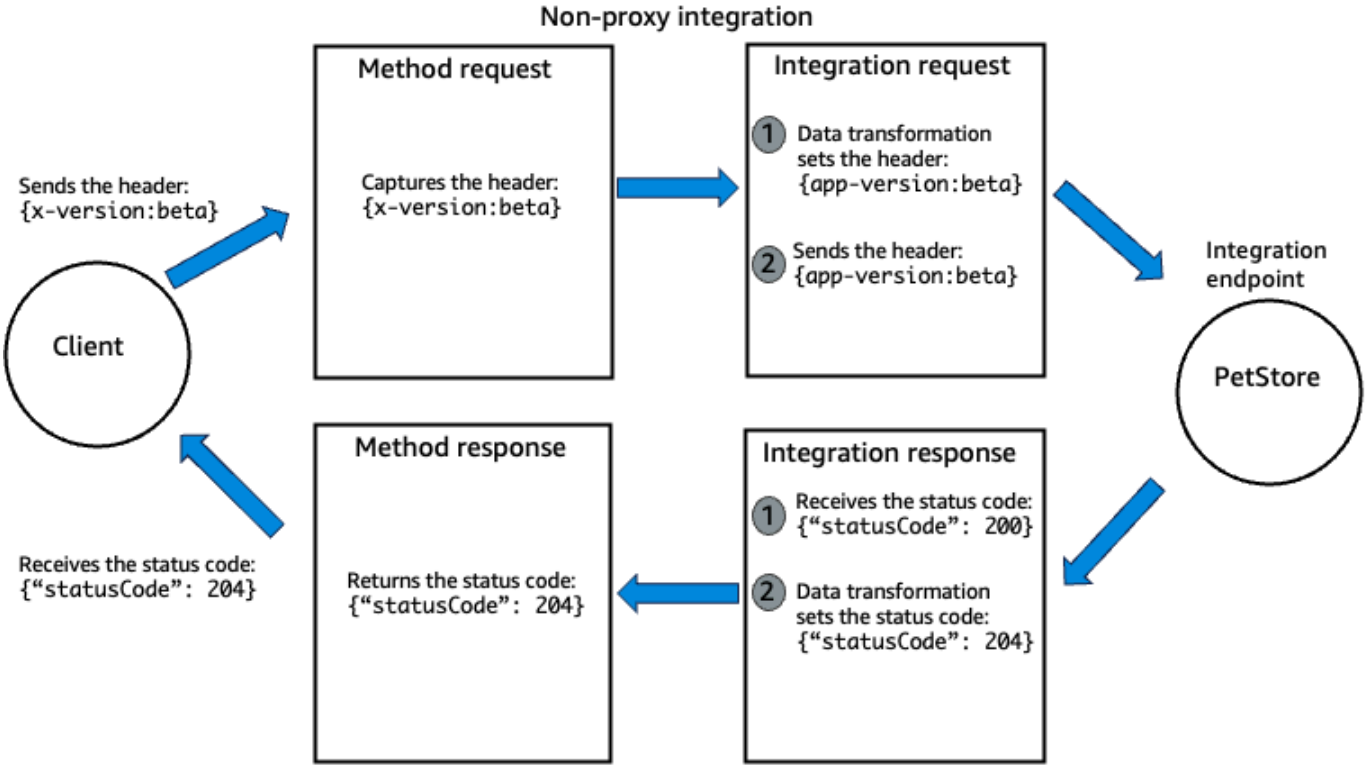
```
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
  ValidationMethodPost:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref ValidationResource
      HttpMethod: POST
      AuthorizationType: NONE
      RequestValidatorId: !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}'
```

Trasformazioni dei dati per REST API in Gateway API

> **Note**
>
> In questa sezione vengono illustrate le funzionalità da utilizzare con un'integrazione non proxy. Tuttavia, è consigliabile, ove possibile, usare un'integrazione proxy per la REST API. L'integrazione proxy ha una configurazione di integrazione più semplice e può evolvere con il backend senza annullare le impostazioni esistenti. Per ulteriori informazioni, consulta the section called "Selezione di un tipo di integrazione API".

Se si utilizza un'integrazione non proxy, è possibile usare due funzionalità di Gateway API per trasformare la richiesta di metodo e la risposta di integrazione. È possibile trasformare la richiesta di metodo se utilizza un formato di payload diverso rispetto al payload della richiesta di integrazione. È possibile trasformare la risposta di integrazione se restituisce un formato di payload diverso da quello che deve essere restituito nella risposta di metodo. Per ulteriori informazioni sul ciclo di vita della richiesta, consultare the section called "Risorsa di esempio per una REST API".

L'esempio seguente mostra una trasformazione di dati in cui per l'intestazione `"x-version:beta"`, il parametro di intestazione `x-version` viene trasformato nel parametro di intestazione `app-version`. La trasformazione di dati da `x-version` a `app-version` avviene nella richiesta di integrazione. In questo modo, l'endpoint di integrazione riceve il valore del parametro di intestazione trasformato. Quando l'endpoint di integrazione restituisce un codice di stato, il codice di stato viene trasformato da `200` a `204` prima della risposta di metodo.



Per creare una trasformazione di dati, è possibile utilizzare le seguenti funzionalità:

Mappatura dei parametri

Nella mappatura dei parametri, è possibile modificare i parametri del percorso URL della richiesta di integrazione, i parametri della stringa di query URL o i valori dell'intestazione HTTP, ma non è possibile modificare il payload della richiesta di integrazione. È inoltre possibile modificare i valori

dell'intestazione della risposta HTTP. La mappatura dei parametri si utilizza per creare valori di intestazione statici per CORS (Cross-Origin Resource Sharing).

È possibile utilizzare la mappatura dei parametri nella richiesta di integrazione per integrazioni proxy e non proxy, ma per usarla per una risposta di integrazione, è necessaria un'integrazione non proxy. La mappatura dei parametri non richiede script in Velocity Template Language (VTL). Per ulteriori informazioni, consulta the section called "Mappatura dei parametri".

Trasformazioni dei modelli di mappatura

Nelle trasformazioni dei modelli di mappatura, si usa un modello di mappatura per mappare i parametri del percorso URL, i parametri della stringa di query URL, le intestazioni HTTP e la richiesta di integrazione o il corpo della risposta di integrazione. Un modello di mappatura è uno script espresso in Velocity Template Language (VTL) utilizzando espressioni JSONPath e applicato al payload in base all'intestazione `Content-type`.

Con un modello di mappatura è possibile eseguire le seguenti operazioni:

- Seleziona i dati da inviare utilizzando l'integrazione Servizi AWS, ad esempio le funzioni Amazon DynamoDB o Lambda o gli endpoint HTTP. Per ulteriori informazioni, consulta the section called "Tutorial: modifica la richiesta di integrazione e la risposta per le integrazioni a AWS services".
- Sostituire in modo condizionale i parametri di richiesta e risposta di integrazione di un'API, creare nuovi valori di intestazione e sostituire i codici di stato. Per ulteriori informazioni, consulta the section called "Sostituzione dei parametri della richiesta e della risposta e dei codici di stato dell'API".

È anche possibile specificare il comportamento dell'API quando il corpo di una richiesta di integrazione ha un'intestazione `Content-type` senza modelli di mappatura corrispondenti. Questo processo è chiamato comportamento passthrough di integrazione. Per ulteriori informazioni, consulta the section called "Comportamento della richiesta di metodo per payload senza modelli di mappatura".

Scelta tra mappatura dei parametri e trasformazioni dei modelli di mappatura

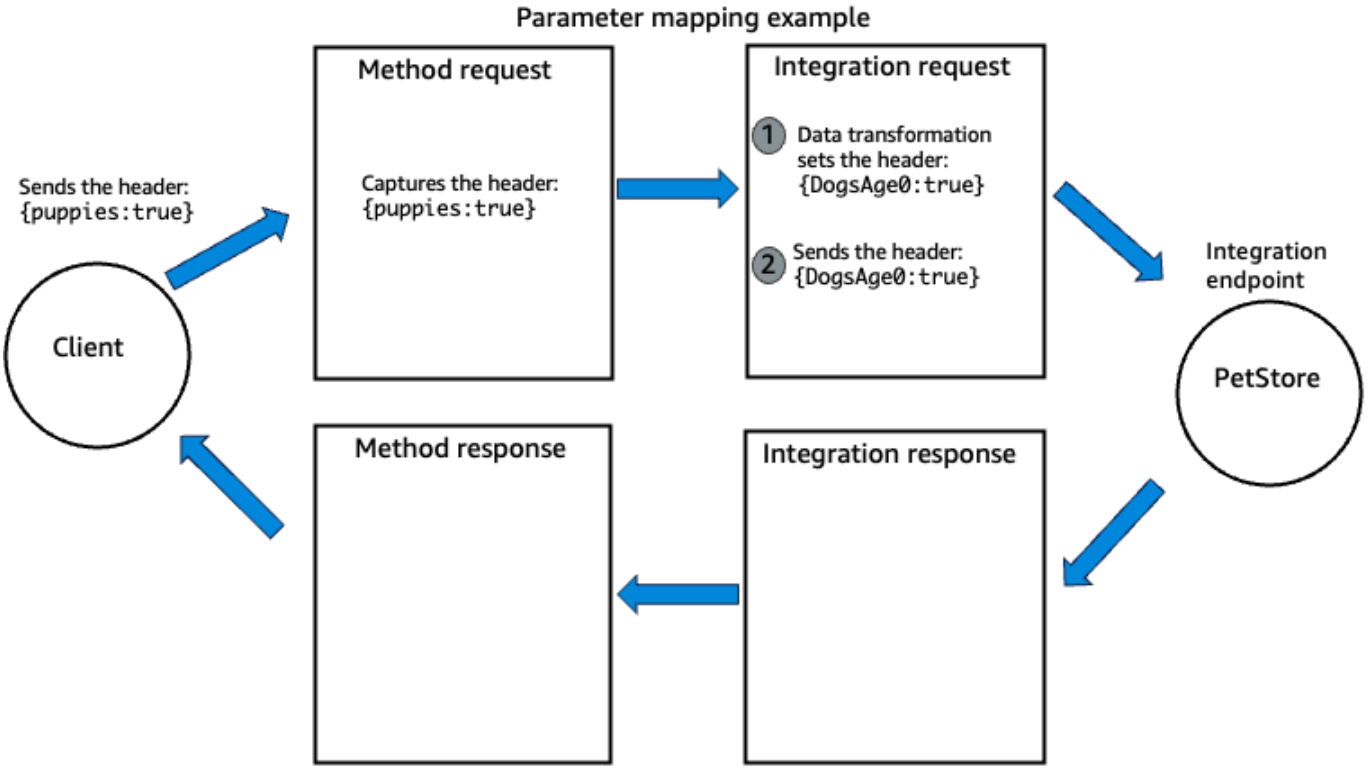
Quando possibile, è consigliabile utilizzare la mappatura dei parametri per trasformare i dati. Se l'API richiede la modifica del corpo o l'esecuzione di sostituzioni e modifiche condizionali in base alla richiesta o alla risposta di integrazione in entrata e non è possibile utilizzare un'integrazione proxy, si usano le trasformazioni dei modelli di mappatura.

Mappatura dei parametri per REST API in Gateway API

> **Note**
>
> Se si utilizza un'API HTTP, consultare the section called "Mappatura dei parametri".

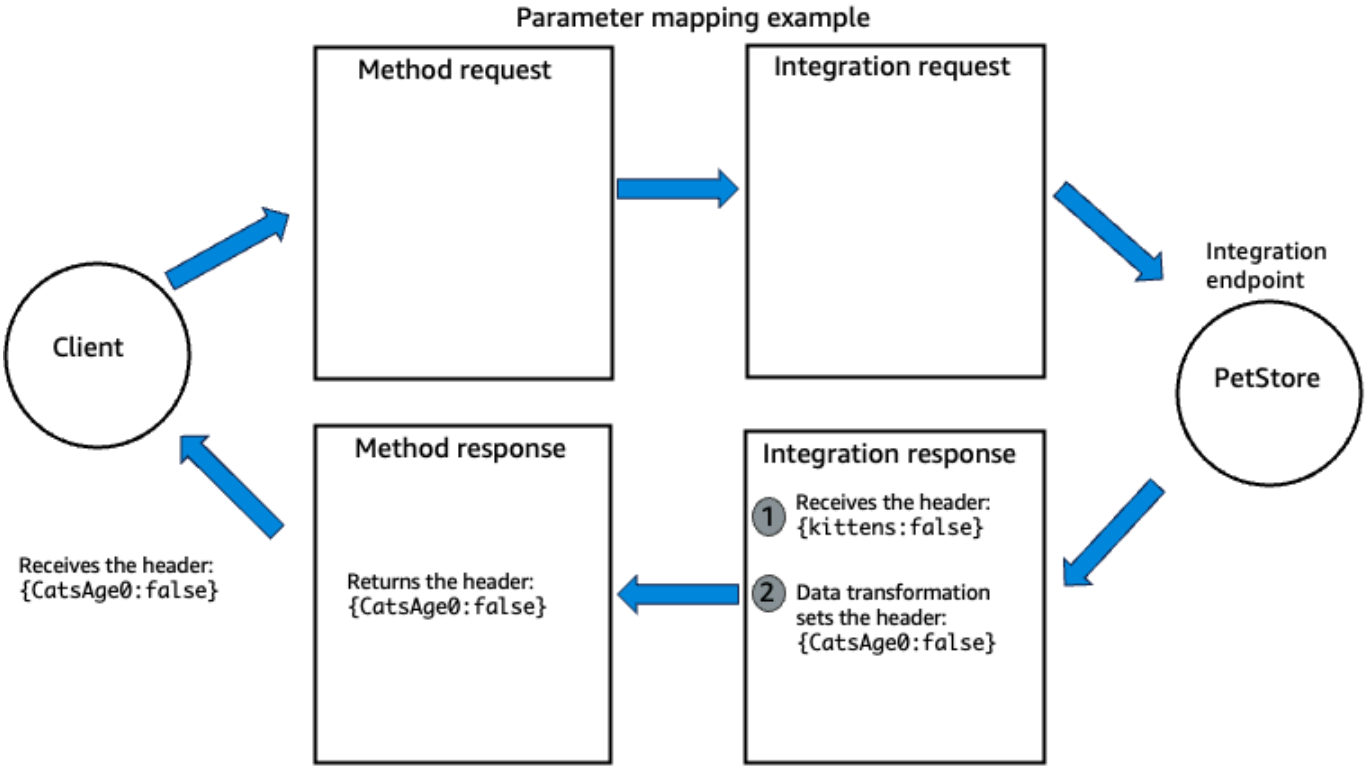
Nella mappatura dei parametri si mappano i parametri della richiesta o della risposta. È possibile mappare i parametri utilizzando valori statici o espressioni di mappatura dei parametri. Per l'elenco delle espressioni di mappatura, consultare the section called "Riferimento di origine per la mappatura dei parametri". È possibile utilizzare la mappatura dei parametri nella richiesta di integrazione per integrazioni proxy e non proxy, ma per usarla per una risposta di integrazione, è necessaria un'integrazione non proxy.

Ad esempio, è possibile mappare il parametro di intestazione della richiesta di metodo `puppies` al parametro di intestazione della richiesta di integrazione `DogsAge0`. Quindi, se un client invia l'intestazione `puppies:true` all'API, la richiesta di integrazione invia l'intestazione della richiesta `DogsAge0:true` all'endpoint di integrazione. Nel diagramma seguente viene illustrato il ciclo di vita della richiesta di questo esempio.



Per eseguire questo esempio utilizzando Gateway API, consultare the section called “Esempio 1: mappare un parametro della richiesta di metodo a un parametro della richiesta di integrazione”.

Inoltre, è possibile mappare il parametro di intestazione `kittens` della risposta di integrazione al parametro di intestazione `CatsAge0` della risposta di metodo. Quindi, se l'endpoint di integrazione restituisce `kittens:false`, il client riceve l'intestazione `CatsAge0:false`. Nel diagramma seguente viene illustrato il ciclo di vita della richiesta di questo esempio.



Argomenti

- Esempi di mappatura dei parametri per REST API in Gateway API
- Riferimento di origine per la mappatura dei parametri per REST API in Gateway API

Esempi di mappatura dei parametri per REST API in Gateway API

Gli esempi seguenti mostrano come creare espressioni di mappatura dei parametri utilizzando la console Gateway API, OpenAPI e modelli CloudFormation . Per un esempio di come utilizzare la mappatura dei parametri per creare le intestazioni CORS richieste, consultare the section called “NUCLEI”.

Esempio 1: mappare un parametro della richiesta di metodo a un parametro della richiesta di integrazione

L'esempio seguente mappa il parametro di intestazione `puppies` della richiesta di metodo al parametro di intestazione `DogsAge0`della richiesta di integrazione.

Console di gestione AWS

Per mappare il parametro della richiesta di metodo

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere una REST API.
3. Scegliere un metodo.

   Il metodo deve disporre di un'integrazione non proxy.
4. Per Impostazioni della richiesta del metodo scegliere Modifica.
5. Scegli Intestazioni di richiesta HTTP.
6. Seleziona Add header (Aggiungi intestazione).
7. In Nome, inserisci **`puppies`**.
8. Scegli Save (Salva).
9. Scegli la scheda Richiesta di integrazione, quindi seleziona Modifica per Impostazioni della richiesta di integrazione.

   Aggiunge Console di gestione AWS automaticamente una mappatura dei parametri da `method.request.header.puppies` a `puppies` per te, ma devi modificare il nome in modo che corrisponda al parametro di intestazione della richiesta previsto dall'endpoint di integrazione.
10. In Nome, inserisci **`DogsAge0`**.
11. Scegli Save (Salva).
12. Implementa nuovamente l'API per rendere effettive le modifiche.

I passaggi seguenti mostrano come verificare che la mappatura dei parametri sia stata completata correttamente.

(Facoltativo) Test della mappatura dei parametri

1. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
2. In Intestazioni, immetti **`puppies:true`**.
3. Scegli Test (Esegui test).
4. Nella sezione Log, il risultato sarà simile al seguente:

```
Tue Feb 04 00:28:36 UTC 2025 : Method request headers: {puppies=true}
Tue Feb 04 00:28:36 UTC 2025 : Method request body before transformations:
Tue Feb 04 00:28:36 UTC 2025 : Endpoint request URI: http://petstore-demo-
endpoint.execute-api.com/petstore/pets
Tue Feb 04 00:28:36 UTC 2025 : Endpoint request headers: {DogsAge0=true,
 x-amzn-apigateway-api-id=abcd1234, Accept=application/json, User-
Agent=AmazonAPIGateway_aaaaaaa, X-Amzn-Trace-Id=Root=1-abcd-12344}
```

Il parametro dell'intestazione della richiesta è cambiato da `puppies` a `DogsAge0`.

CloudFormation

In questo esempio, si utilizza la proprietà body per importare un file di definizione OpenAPI in Gateway API.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Body:
        openapi: 3.0.1
        info:
          title: ParameterMappingExample
          version: "2025-02-04T00:30:41Z"
        paths:
          /pets:
            get:
              parameters:
                - name: puppies
                  in: header
                  schema:
```

```
                    type: string
              responses:
                "200":
                  description: 200 response
              x-amazon-apigateway-integration:
                httpMethod: GET
                uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets
                responses:
                  default:
                    statusCode: "200"
                requestParameters:
                  integration.request.header.DogsAge0: method.request.header.puppies
                passthroughBehavior: when_no_match
                type: http
  ApiGatewayDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn: Api
    Properties:
      RestApiId: !Ref Api
  ApiGatewayDeployment20250219:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn: Api
    Properties:
      RestApiId: !Ref Api
  Stage:
    Type: 'AWS::ApiGateway::Stage'
    Properties:
       DeploymentId: !Ref ApiGatewayDeployment20250219
       RestApiId: !Ref Api
       StageName: prod
```

OpenAPI

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "ParameterMappingExample",
    "version" : "2025-02-04T00:30:41Z"
  },
  "paths" : {
    "/pets" : {
      "get" : {
        "parameters" : [ {
```

```
          "name" : "puppies",
          "in" : "header",
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response"
          }
        },
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
          "responses" : {
            "default" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.header.DogsAge0" : "method.request.header.puppies"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "http"
        }
      }
    }
  }
}
```

Esempio 2: mappare più parametri di richiesta di metodo a diversi parametri di richiesta di integrazione

L'esempio seguente mappa il parametro della stringa di query della richiesta di metodo con più valori `methodRequestQueryParam` al parametro della stringa di query della richiesta di integrazione `integrationQueryParam` e mappa il parametro dell'intestazione della richiesta di metodo `methodRequestHeaderParam` al parametro del percorso della richiesta di integrazione `integrationPathParam`.

Console di gestione AWS

Per mappare i parametri della richiesta di metodo

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere una REST API.
3. Scegliere un metodo.

   Il metodo deve disporre di un'integrazione non proxy.
4. Per Impostazioni della richiesta del metodo scegliere Modifica.
5. Scegli Parametri della stringa di query URL.
6. Scegliere Add query string (Aggiungi stringa di query).
7. In Nome, inserisci **`methodRequestQueryParam`**.
8. Scegli Intestazioni di richiesta HTTP.
9. Seleziona Add header (Aggiungi intestazione).
10. In Nome, inserisci **`methodRequestHeaderParam`**.
11. Scegli Save (Salva).
12. Scegli la scheda Richiesta di integrazione, quindi seleziona Modifica per Impostazioni della richiesta di integrazione.
13. Scegli Parametri di percorso URL.
14. Scegli Aggiungi parametro di percorso.
15. In Nome, inserisci **`integrationPathParam`**.
16. In Mappato da, inserire **`method.request.header.methodRequestHeaderParam`**.

    In questo modo l'intestazione della richiesta di metodo specificata nella richiesta di metodo viene mappata a un nuovo parametro del percorso della richiesta di integrazione.
17. Scegli Parametri della stringa di query URL.
18. Scegliere Add query string (Aggiungi stringa di query).
19. In Nome, inserisci **`integrationQueryParam`**.
20. In Mappato da, inserire **`method.request.multivaluequerystring.methodRequestQueryParam`**.

    In questo modo il parametro della stringa di query con più valori viene mappato a un nuovo parametro della stringa di query della richiesta di integrazione a valore singolo.

21. Scegli Save (Salva).
22. Implementa nuovamente l'API per rendere effettive le modifiche.

CloudFormation

In questo esempio, si utilizza la proprietà body per importare un file di definizione OpenAPI in Gateway API.

La seguente definizione OpenAPI crea le mappature dei parametri per un'integrazione HTTP:

- L'intestazione della richiesta di metodo `methodRequestHeaderParam` nel parametro di percorso della richiesta di integrazione `integrationPathParam`
- La stringa di query della richiesta di metodo con più valori `methodRequestQueryParam` nella stringa di query della richiesta di integrazione `integrationQueryParam`

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Body:
        openapi: 3.0.1
        info:
          title: Parameter mapping example 2
          version: "2025-01-15T19:12:31Z"
        paths:
          /:
            post:
              parameters:
                - name: methodRequestQueryParam
                  in: query
                  schema:
                    type: string
                - name: methodRequestHeaderParam
                  in: header
                  schema:
                    type: string
              responses:
                "200":
                  description: 200 response
              x-amazon-apigateway-integration:
```

```
                httpMethod: GET
                uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets
                responses:
                  default:
                    statusCode: "200"
                requestParameters:
                  integration.request.querystring.integrationQueryParam:
method.request.multivaluequerystring.methodRequestQueryParam
                  integration.request.path.integrationPathParam:
method.request.header.methodRequestHeaderParam
                requestTemplates:
                  application/json: '{"statusCode": 200}'
                passthroughBehavior: when_no_templates
                timeoutInMillis: 29000
                type: http
 ApiGatewayDeployment:
   Type: 'AWS::ApiGateway::Deployment'
   DependsOn: Api
   Properties:
     RestApiId: !Ref Api
 ApiGatewayDeployment20250219:
   Type: 'AWS::ApiGateway::Deployment'
   DependsOn: Api
   Properties:
     RestApiId: !Ref Api
 Stage:
   Type: 'AWS::ApiGateway::Stage'
   Properties:
      DeploymentId: !Ref ApiGatewayDeployment20250219
      RestApiId: !Ref Api
      StageName: prod
```

OpenAPI

La seguente definizione OpenAPI crea le mappature dei parametri per un'integrazione HTTP:

- L'intestazione della richiesta di metodo `methodRequestHeaderParam` nel parametro di percorso della richiesta di integrazione `integrationPathParam`
- La stringa di query della richiesta di metodo con più valori `methodRequestQueryParam` nella stringa di query della richiesta di integrazione `integrationQueryParam`

```
{
```

```
 "openapi" : "3.0.1",
 "info" : {
   "title" : "Parameter mapping example 2",
   "version" : "2025-01-15T19:12:31Z"
 },
 "paths" : {
   "/" : {
     "post" : {
       "parameters" : [ {
         "name" : "methodRequestQueryParam",
         "in" : "query",
         "schema" : {
           "type" : "string"
         }
       }, {
         "name" : "methodRequestHeaderParam",
         "in" : "header",
         "schema" : {
           "type" : "string"
         }
       } ],
       "responses" : {
         "200" : {
           "description" : "200 response"
         }
       },
       "x-amazon-apigateway-integration" : {
         "httpMethod" : "GET",
         "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
         "responses" : {
           "default" : {
             "statusCode" : "200"
           }
         },
         "requestParameters" : {
           "integration.request.querystring.integrationQueryParam" :
"method.request.multivaluequerystring.methodRequestQueryParam",
           "integration.request.path.integrationPathParam" :
"method.request.header.methodRequestHeaderParam"
         },
         "requestTemplates" : {
           "application/json" : "{\"statusCode\": 200}"
         },
         "passthroughBehavior" : "when_no_templates",
```

```
            "timeoutInMillis" : 29000,
            "type" : "http"
          }
        }
      }
    }
  }
```

Esempio 3: mappare i campi dal corpo della richiesta JSON ai parametri della richiesta di integrazione

È possibile mappare i parametri della richiesta di integrazione anche dai campi del corpo della richiesta JSON tramite un'espressione JSONPath. L'esempio seguente mappa il corpo della richiesta di metodo a un'intestazione della richiesta di integrazione denominata `body-header` e mappa parte del corpo della richiesta, come indicato da un'espressione JSON, a un'intestazione della richiesta di integrazione denominata `pet-price`.

Per testare questo esempio è necessario fornire un input che contenga una categoria di prezzo, come la seguente:

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

Console di gestione AWS

Per mappare i parametri della richiesta di metodo

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere una REST API.
3. Scegliere un metodo `POST`, `PUT`, `PATCH` o `ANY`.

   Il metodo deve disporre di un'integrazione non proxy.
4. Per Impostazioni della richiesta di integrazione, scegliere Modifica.
5. Scegliere Parametri delle intestazioni delle richieste URL.

6. Scegliere Aggiungi parametro dell'intestazione della richiesta.
7. In Nome, inserisci **`body-header`**.
8. In Mappato da, inserire **`method.request.body`**.

   In questo modo il corpo della richiesta di metodo viene mappato a un nuovo parametro di intestazione della richiesta di integrazione.
9. Scegliere Aggiungi parametro dell'intestazione della richiesta.
10. In Nome, inserisci **`pet-price`**.
11. In Mappato da, inserire **`method.request.body[0].price`**.

    In questo modo una parte del corpo della richiesta di metodo viene mappata a un nuovo parametro di intestazione della richiesta di integrazione.
12. Scegli Save (Salva).
13. Implementa nuovamente l'API per rendere effettive le modifiche.

CloudFormation

In questo esempio, si utilizza la proprietà body per importare un file di definizione OpenAPI in Gateway API.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Body:
        openapi: 3.0.1
        info:
          title: Parameter mapping example 3
          version: "2025-01-15T19:19:14Z"
        paths:
          /:
            post:
              responses:
                "200":
                  description: 200 response
              x-amazon-apigateway-integration:
                httpMethod: GET
                uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets
```

```
                responses:
                  default:
                    statusCode: "200"
                requestParameters:
                  integration.request.header.pet-price: method.request.body[0].price
                  integration.request.header.body-header: method.request.body
                requestTemplates:
                  application/json: '{"statusCode": 200}'
                passthroughBehavior: when_no_templates
                timeoutInMillis: 29000
                type: http
  ApiGatewayDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn: Api
    Properties:
      RestApiId: !Ref Api
  ApiGatewayDeployment20250219:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn: Api
    Properties:
      RestApiId: !Ref Api
  Stage:
    Type: 'AWS::ApiGateway::Stage'
    Properties:
       DeploymentId: !Ref ApiGatewayDeployment20250219
       RestApiId: !Ref Api
       StageName: prod
```

OpenAPI

La seguente definizione OpenAPI mappa i parametri della richiesta di integrazione dai campi del corpo della richiesta JSON.

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "Parameter mapping example 3",
    "version" : "2025-01-15T19:19:14Z"
  },
  "paths" : {
    "/" : {
      "post" : {
        "responses" : {
```

```
          "200" : {
            "description" : "200 response"
          }
        },
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
          "responses" : {
            "default" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.header.pet-price" : "method.request.body[0].price",
            "integration.request.header.body-header" : "method.request.body"
          },
          "requestTemplates" : {
            "application/json" : "{\"statusCode\": 200}"
          },
          "passthroughBehavior" : "when_no_templates",
          "timeoutInMillis" : 29000,
          "type" : "http"
        }
      }
    }
  }
}
```

Esempio 4: mappare la risposta di integrazione alla risposta di metodo

È anche possibile mappare la risposta di integrazione alla risposta di metodo. L'esempio seguente mappa il corpo della risposta di integrazione a un'intestazione della risposta di metodo denominata `location`, mappa l'intestazione della risposta di integrazione `x-app-id` all'intestazione della risposta di metodo e mappa l'intestazione `id` della risposta di integrazione con più valori `item` all'intestazione della risposta del metodo `items`.

Console di gestione AWS

Per mappare la risposta di integrazione

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.

2. Scegliere una REST API.
3. Scegliere un metodo.

   Il metodo deve disporre di un'integrazione non proxy.
4. Scegliere la scheda Metodo di risposta, quindi per Risposta 200 scegliere Modifica.
5. Per Nome intestazione, scegliere Aggiungi intestazione.
6. Creare tre intestazioni denominate **`id`**, **`item`** e **`location`**.
7. Scegli Save (Salva).
8. Nella scheda Risposta di integrazione scegliere Modifica per Predefinito - Risposta.
9. In Mappature intestazione immettere i seguenti valori.

   a. Per id, immettere **`integration.response.header.x-app-id`**
   b. Per elemento, immettere **`integration.response.multivalueheader.item`**
   c. Per percorso, immettere **`integration.response.body.redirect.url`**
10. Scegli Save (Salva).
11. Implementa nuovamente l'API per rendere effettive le modifiche.

CloudFormation

In questo esempio, si utilizza la proprietà body per importare un file di definizione OpenAPI in Gateway API.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Body:
        openapi: 3.0.1
        info:
          title: Parameter mapping example
          version: "2025-01-15T19:21:35Z"
        paths:
          /:
            post:
              responses:
                "200":
                  description: 200 response
```

```
                  headers:
                    item:
                      schema:
                        type: string
                    location:
                      schema:
                        type: string
                    id:
                      schema:
                        type: string
              x-amazon-apigateway-integration:
                type: http
                httpMethod: GET
                uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets
                responses:
                  default:
                    statusCode: "200"
                    responseParameters:
                      method.response.header.id: integration.response.header.x-app-
id
                      method.response.header.location:
 integration.response.body.redirect.url
                      method.response.header.item:
 integration.response.multivalueheader.item
                requestTemplates:
                  application/json: '{"statusCode": 200}'
                passthroughBehavior: when_no_templates
                timeoutInMillis: 29000
  ApiGatewayDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn: Api
    Properties:
      RestApiId: !Ref Api
  ApiGatewayDeployment20250219:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn: Api
    Properties:
      RestApiId: !Ref Api
  Stage:
    Type: 'AWS::ApiGateway::Stage'
    Properties:
       DeploymentId: !Ref ApiGatewayDeployment20250219
       RestApiId: !Ref Api
```

```
      StageName: prod
```

OpenAPI

La definizione OpenAPI seguente mappa la risposta di integrazione alla risposta di metodo.

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "Parameter mapping example",
    "version" : "2025-01-15T19:21:35Z"
  },
  "paths" : {
    "/" : {
      "post" : {
        "responses" : {
          "200" : {
            "description" : "200 response",
            "headers" : {
              "item" : {
                "schema" : {
                  "type" : "string"
                }
              },
              "location" : {
                "schema" : {
                  "type" : "string"
                }
              },
              "id" : {
                "schema" : {
                  "type" : "string"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : {
          "type" : "http",
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
          "responses" : {
            "default" : {
              "statusCode" : "200",
```

```
              "responseParameters" : {
                "method.response.header.id" : "integration.response.header.x-app-
id",
                "method.response.header.location" :
 "integration.response.body.redirect.url",
                "method.response.header.item" :
 "integration.response.multivalueheader.item"
              }
            }
          },
          "requestTemplates" : {
            "application/json" : "{\"statusCode\": 200}"
          },
          "passthroughBehavior" : "when_no_templates",
          "timeoutInMillis" : 29000
        }
      }
    }
  }
}
```

Riferimento di origine per la mappatura dei parametri per REST API in Gateway API

Quando si crea una mappatura dei parametri, è possibile specificare i parametri della richiesta di metodo o della risposta di integrazione da modificare e le modalità con cui si desidera modificare tali parametri.

La tabella seguente mostra i parametri della richiesta di metodo che è possibile mappare e l'espressione per creare la mappatura. In queste espressioni, *`name`* è il nome di un parametro di richiesta del metodo. Ad esempio, per mappare il parametro dell'intestazione della richiesta `puppies`, si utilizza l'espressione `method.request.header.puppies`. L'espressione deve corrispondere all'espressione regolare `'^[a-zA-Z0-9._$-]+$]'`. È possibile utilizzare la mappatura dei parametri nella richiesta di integrazione per le integrazioni proxy e non proxy.

Origine dati mappata	Espressione di mappatura
Percorso della richiesta di metodo	`method.request.path.` *`name`*
Stringa di query della richiesta di metodo	`method.request.querystring.` *`name`*

Origine dati mappata	Espressione di mappatura
Multi-value stringa di interrogazione della richiesta di metodo	`method.request.multivaluequerystring.` *`name`*
Intestazione della richiesta di metodo	`method.request.header.` *`name`*
Multi-value intestazione della richiesta del metodo	`method.request.multivalueheader.` *`name`*
Corpo della richiesta di metodo	`method.request.body`
corpo della richiesta del metodo () JsonPath	`method.request.body.` *`JSONPath_EXPRESSION`* . *`JSONPath_EXPRESSION`* è un'espressione JSONPath per un campo JSON del corpo di una richiesta. Per ulteriori informazioni, consultare JSONPath expression.
Variabili di fase	`stageVariables.` *`name`*
Variabili di contesto	`context.`*`name`* Il nome deve essere una delle variabili di contesto supportate.
Valore statico	*`'static_value'`* . *`static_value`* è una stringa letterale e deve essere racchiusa tra virgolette singole. Ad esempio, `'https://www.example.com'` .

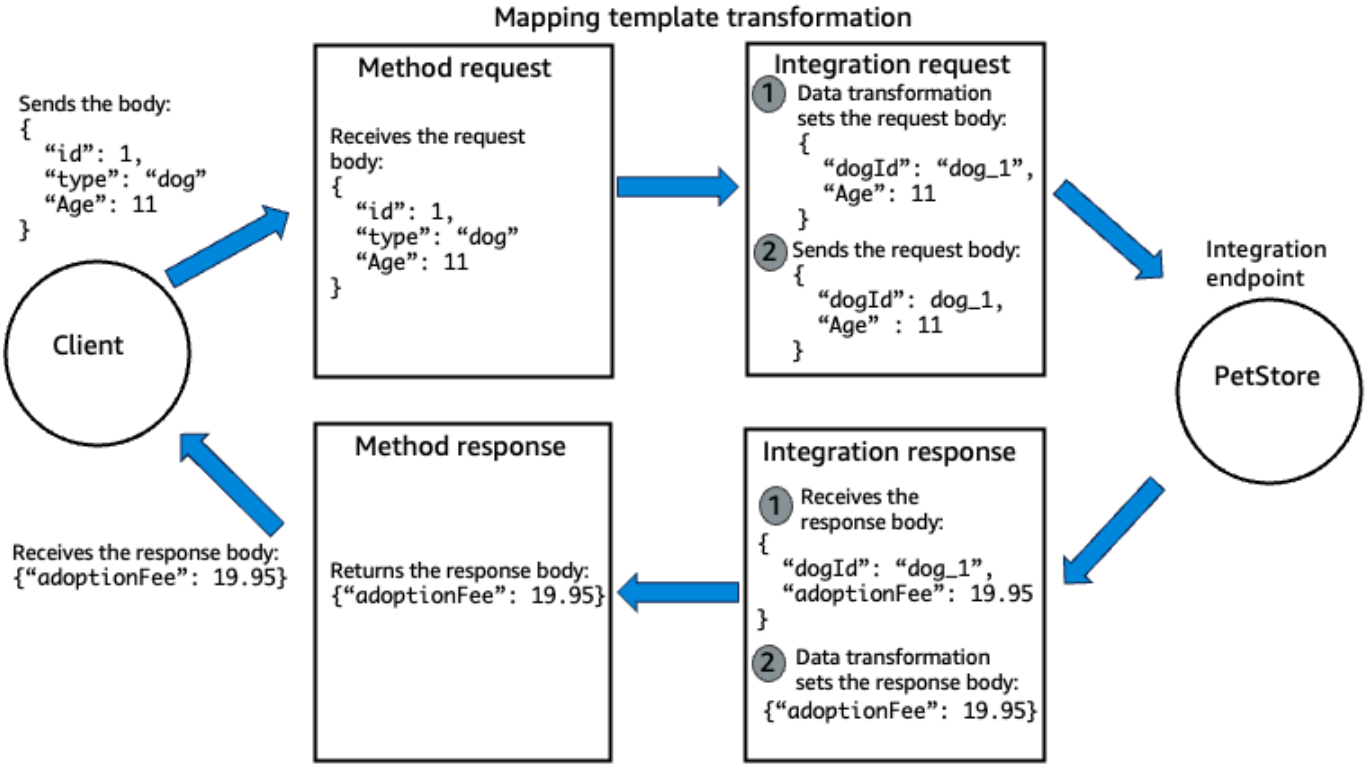
La tabella seguente mostra i parametri della risposta di integrazione che è possibile mappare e l'espressione per creare la mappatura. In queste espressioni, *`name`* è il nome di un parametro di risposta di integrazione. È possibile mappare le intestazioni della risposta di metodo da qualsiasi intestazione o corpo della risposta di integrazione, variabili $context o valori statici. Per utilizzare la mappatura dei parametri per una risposta di integrazione, è necessaria un'integrazione non proxy.

Origine dati mappata	Espressione di mappatura
Intestazione della risposta di integrazione	`integration.response.header.` *`name`*
Intestazione della risposta di integrazione	`integration.response.multivalueheader.` *`name`*
Corpo della risposta di integrazione	`integration.response.body`
Corpo della risposta di integrazione (JsonPath)	`integration.response.body.` *`JSONPath_EXPRESSION`* *`JSONPath_EXPRESSION`* è un'espressione JSONPath per un campo JSON del corpo di una risposta. Per ulteriori informazioni, consultare JSONPath expression.
Variabile di fase	`stageVariables.` *`name`*
Variabile di contesto	`context.`*`name`* Il nome deve essere una delle variabili di contesto supportate.
Valore statico	*`'static_value'`* *`static_value`* è una stringa letterale e deve essere racchiusa tra virgolette singole. Ad esempio, `'https://www.example.com'` .

Trasformazioni dei modelli di mappatura per REST API in Gateway API

Una trasformazione del modello di mappatura utilizza un modello di mappatura per modificare la richiesta o la risposta di integrazione. Un modello di mappatura è uno script espresso in Velocity Template Language (VTL) e applicato al payload tramite JSONPath in base all'intestazione `Content-type`. Si utilizzano i modelli di mappatura per le trasformazioni dei modelli di mappatura. Questa sezione descrive le informazioni concettuali relative ai modelli di mappatura.

Il diagramma seguente mostra il ciclo di vita della richiesta per una `POST /pets` risorsa che ha un'integrazione con un endpoint di integrazione. PetStore In questa API, un utente invia i dati su un animale domestico e l'endpoint di integrazione restituisce la tariffa di adozione associata all'animale domestico. In questo ciclo di vita della richiesta, le trasformazioni dei modelli di mappatura filtrano il corpo della richiesta all'endpoint di integrazione e il corpo della risposta dall'endpoint di integrazione.



Le sezioni seguenti illustrano il ciclo di vita della richiesta e della risposta.

Richiesta di metodo e richiesta di integrazione

Nell'esempio precedente, se il corpo della richiesta inviato alla richiesta di metodo è:

```
POST /pets
    HTTP/1.1
    Host:abcd1234.us-west-2.amazonaws.com
    Content-type: application/json

  {
    "id": 1,
    "type": "dog",
    "Age": 11,
  }
```

Questo corpo della richiesta non è nel formato corretto per essere utilizzato dall'endpoint di integrazione, pertanto Gateway API esegue una trasformazione del modello di mappatura. API Gateway esegue solo una trasformazione del modello di mappatura perché esiste un modello di mappatura definito per. Content-Type `application/json` Se non definisci un modello di mappatura per, per impostazione predefinita Content-Type, API Gateway trasmette il corpo attraverso la richiesta di integrazione all'endpoint di integrazione. Per modificare questo comportamento, consultare the section called "Comportamento della richiesta di metodo per payload senza modelli di mappatura".

Il seguente modello di mappatura trasforma i dati della richiesta di metodo nella richiesta di integrazione prima che vengano inviati all'endpoint di integrazione:

```
#set($inputRoot = $input.path('$'))
  {
    "dogId" : "dog_"$elem.id,
    "Age": $inputRoot.Age
  }
```

1. La variabile `$inputRoot` rappresenta l'oggetto radice nei dati JSON originali della sezione precedente. Le direttive iniziano con il simbolo #.
2. `dog` è una concatenazione di `id` dell'utente e di un valore di stringa.
3. `Age` proviene dal corpo della richiesta di metodo.

Quindi, il seguente output viene inoltrato all'endpoint di integrazione:

```
{
    "dogId" : "dog_1",
    "Age": 11
  }
```

Risposta di integrazione e risposta di metodo

Dopo l'esito positivo della richiesta all'endpoint di integrazione, l'endpoint invia una risposta alla risposta di integrazione di Gateway API. L'esempio seguente indica i dati di output di un endpoint di integrazione:

```
{
    "dogId" : "dog_1",
    "adoptionFee": 19.95,
```

```
}
```

La risposta di metodo prevede un payload diverso da quello restituito dalla risposta di integrazione. Gateway API esegue una trasformazione del modello di mappatura. API Gateway esegue solo una trasformazione del modello di mappatura perché esiste un modello di mappatura definito per. Content-Type `application/json` Se non definisci un modello di mappatura per, per impostazione predefinita Content-Type, API Gateway passa il corpo attraverso la risposta di integrazione alla risposta del metodo. Per modificare questo comportamento, consultare the section called “Comportamento della richiesta di metodo per payload senza modelli di mappatura”.

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

Il seguente output viene inviato alla risposta di metodo:

```
{"adoptionFee": 19.95}
```

L’esempio di trasformazione del modello di mappatura è completato. Se possibile, anziché utilizzare trasformazioni di modelli di mappatura, è consigliabile un’integrazione proxy per trasformare i dati. Per ulteriori informazioni, consulta the section called “Selezione di un tipo di integrazione API”.

Comportamento della richiesta di metodo per i payload senza modelli di mappatura per REST API in Gateway API

Se la richiesta di metodo ha un payload senza modello di mappatura definito per l’intestazione `Content-Type`, è possibile scegliere di passare il payload della richiesta fornito dal client tramite la richiesta di integrazione al backend senza trasformazione. Il processo è noto come passthrough di integrazione.

L’effettivo comportamento del passthrough di una richiesta in entrata è determinato da questa impostazione. Sono disponibili tre opzioni:

Quando nessun modello corrisponde all'intestazione della richiesta Content-Type

> Selezionare questa opzione se si desidera che venga eseguito il passthrough del corpo della richiesta di metodo nella richiesta di integrazione al back-end senza trasformazione quando il tipo di contenuto della richiesta di metodo non corrisponde ai tipi di contenuto associati ai modelli di mappatura.

Durante la chiamata all'API Gateway API, scegli questa opzione impostando `WHEN_NO_MATCH` come valore della proprietà `passthroughBehavior` in Integrazione.

Quando non ci sono modelli definiti (consigliato)

Selezionare questa opzione se si desidera che venga eseguito il passthrough del corpo della richiesta di metodo nella richiesta di integrazione al back-end senza trasformazione quando nella richiesta di integrazione non è stato definito un modello di mappatura. Se viene definito un modello con questa opzione selezionata, la richiesta di metodo con un payload e un tipo di contenuto che non corrispondono ad alcun modello di mappatura definito verrà rifiutata con una risposta HTTP 415 Tipo di supporto non compatibile.

Durante la chiamata all'API Gateway API, scegli questa opzione impostando `WHEN_NO_TEMPLATES` come valore della proprietà `passthroughBehavior` in Integrazione.

Mai

Selezionare questa opzione se non si desidera che venga eseguito il passthrough del corpo della richiesta di metodo nella richiesta di integrazione al back-end senza trasformazione quando nella richiesta di integrazione non è stato definito un modello di mappatura. Se viene definito un modello al momento della selezione di questa opzione, la richiesta di metodo di un tipo di contenuto non mappato sarà rifiutata con la risposta Tipo di supporto non supportato HTTP 415.

Durante la chiamata all'API Gateway API, scegli questa opzione impostando `NEVER` come valore della proprietà `passthroughBehavior` in Integrazione.

Gli esempi seguenti illustrano i possibili comportamenti del passthrough.

Esempio 1: viene definito un modello di mappatura nella richiesta di integrazione per il tipo di contenuto `application/json`.

Content-type	Opzione passthrough	Comportamento
Nessuno Gateway API utilizza l'impostazione predefinita `application/json`	`WHEN_NO_MATCH`	Il payload della richiesta viene trasformato utilizzando il modello.

Content-type	Opzione passthrough	Comportamento
Nessuno Gateway API utilizza l'impostazione predefinita `application/json`	`WHEN_NO_TEMPLATES`	Il payload della richiesta viene trasformato utilizzando il modello.
Nessuno Gateway API utilizza l'impostazione predefinita `application/json`	`NEVER`	Il payload della richiesta viene trasformato utilizzando il modello.
`application/json`	`WHEN_NO_MATCH`	Il payload della richiesta viene trasformato utilizzando il modello.
`application/json`	`WHEN_NO_TEMPLATES`	Il payload della richiesta viene trasformato utilizzando il modello.
`application/json`	`NEVER`	Il payload della richiesta viene trasformato utilizzando il modello.
`application/xml`	`WHEN_NO_MATCH`	Il payload della richiesta non viene trasformato e viene inviato al back-end inalterato.
`application/xml`	`WHEN_NO_TEMPLATES`	La richiesta viene respinta con una risposta HTTP `415 Unsupported Media Type`.
`application/xml`	`NEVER`	La richiesta viene respinta con una risposta HTTP `415 Unsupported Media Type`.

Esempio 2: viene definito un modello di mappatura nella richiesta di integrazione per il tipo di contenuto `application/xml`.

Content-type	Opzione passthrough	Comportamento
Nessuno Gateway API utilizza l'impostazione predefinita `application/json`	`WHEN_NO_MATCH`	Il payload della richiesta non viene trasformato e viene inviato al back-end inalterato.
Nessuno Gateway API utilizza l'impostazione predefinita `application/json`	`WHEN_NO_TEMPLATES`	La richiesta viene respinta con una risposta HTTP `415 Unsupported Media Type`.
Nessuno Gateway API utilizza l'impostazione predefinita `application/json`	`NEVER`	La richiesta viene respinta con una risposta HTTP `415 Unsupported Media Type`.
`application/json`	`WHEN_NO_MATCH`	Il payload della richiesta non viene trasformato e viene inviato al back-end inalterato.
`application/json`	`WHEN_NO_TEMPLATES`	La richiesta viene respinta con una risposta HTTP `415 Unsupported Media Type`.
`application/json`	`NEVER`	La richiesta viene respinta con una risposta HTTP `415 Unsupported Media Type`.

Content-type	Opzione passthrough	Comportamento
`application/xml`	`WHEN_NO_MATCH`	Il payload della richiesta viene trasformato utilizzando il modello.
`application/xml`	`WHEN_NO_TEMPLATES`	Il payload della richiesta viene trasformato utilizzando il modello.
`application/xml`	`NEVER`	Il payload della richiesta viene trasformato utilizzando il modello.

Esempio 3: nessun modello di mappatura viene definito nella richiesta di integrazione

Content-type	Opzione passthrough	Comportamento
Nessuno Gateway API utilizza l'impostazione predefinita `application/json`	`WHEN_NO_MATCH`	Il payload della richiesta non viene trasformato e viene inviato al back-end inalterato.
Nessuno Gateway API utilizza l'impostazione predefinita `application/json`	`WHEN_NO_TEMPLATES`	Il payload della richiesta non viene trasformato e viene inviato al back-end inalterato.
Nessuno Gateway API utilizza l'impostazione predefinita `application/json`	`NEVER`	La richiesta viene respinta con una risposta HTTP `415 Unsupported Media Type`.

Content-type	Opzione passthrough	Comportamento
`application/json`	`WHEN_NO_MATCH`	Il payload della richiesta non viene trasformato e viene inviato al back-end inalterato.
`application/json`	`WHEN_NO_TEMPLATES`	Il payload della richiesta non viene trasformato e viene inviato al back-end inalterato.
`application/json`	`NEVER`	La richiesta viene respinta con una risposta HTTP `415 Unsupported Media Type`.
`application/xml`	`WHEN_NO_MATCH`	Il payload della richiesta non viene trasformato e viene inviato al back-end inalterato.
`application/xml`	`WHEN_NO_TEMPLATES`	Il payload della richiesta non viene trasformato e viene inviato al back-end inalterato.
`application/xml`	`NEVER`	La richiesta viene respinta con una risposta HTTP `415 Unsupported Media Type`.

Esempio aggiuntivo di modello di mappatura per REST API in Gateway API

L'esempio seguente mostra un'API per album fotografico in Gateway API che utilizza modelli di mappatura per trasformare i dati delle richieste e delle risposte di integrazione. Utilizza anche modelli di dati per definire i payload della richiesta di metodo e della risposta di integrazione. Per ulteriori informazioni sui modelli di dati, consulta Modelli di dati per REST API.

Richiesta di metodo e richiesta di integrazione

Di seguito è riportato un modello che definisce il corpo della richiesta di metodo. Questo modello di input richiede che il chiamante carichi una pagina di foto con un minimo di 10 foto per ogni pagina.

È possibile utilizzare questo modello di input per generare un SDK o per usare la convalida della richiesta per l'API. Quando si utilizza la convalida della richiesta, se il corpo della richiesta di metodo non aderisce alla struttura dei dati del modello, Gateway API non esegue la richiesta.

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

Di seguito è riportato un esempio di corpo della richiesta di metodo che aderisce alla struttura dei dati del modello di dati precedente.

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

In questo esempio, quando il corpo della richiesta di metodo precedente viene inviato dal client, il modello di mappatura trasforma il payload in modo che corrisponda al formato richiesto dall'endpoint di integrazione.

```
#set($inputRoot = $input.path('$'))
{
  "photos": [
#foreach($elem in $inputRoot.photos.photo)
    {
      "id": "$elem.id",
      "photographedBy": "$elem.photographer_first_name $elem.photographer_last_name",
      "title": "$elem.title",
      "ispublic": $elem.ispublic,
      "isfriend": $elem.isfriend,
      "isfamily": $elem.isfamily
    }#if($foreach.hasNext),#end

#end
  ]
}
```

L'esempio seguente fornisce i dati di output della trasformazione:

```
{
  "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
    }
  ]
}
```

Questi dati vengono inviati alla richiesta di integrazione e quindi all'endpoint di integrazione.

Risposta di integrazione e risposta di metodo

Di seguito è riportato un esempio di modello di output per i dati delle foto dall'endpoint di integrazione. È possibile utilizzare questo metodo per un modello di risposta del metodo, che è necessario quando si genera un SDK tipizzato in modo sicuro per l'API. Ciò fa sì che l'output venga inserito in una classe appropriata in Java o 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" }
        }
      }
    }
  }
}
```

L'endpoint di integrazione potrebbe non generare una risposta che aderisce alla struttura dei dati di questo modello. Ad esempio, la risposta di integrazione potrebbe essere simile alla seguente:

```
  "photos": [
    {
      "id": "12345678901",
      "photographedBy": "Saanvi Sarkar",
      "title": "Sample photo 1",
      "description": "My sample photo 1",
      "public": true,
      "friend": false,
      "family": false
```

```
    },
    {
      "id": "23456789012",
      "photographedBy": "Richard Roe",
      "title": "Sample photo 2",
      "description": "My sample photo 1",
      "public": true,
      "friend": false,
      "family": false
    }
  ]
}
```

Il modello di mappatura di esempio seguente trasforma i dati della risposta di integrazione nel formato previsto dalla risposta di metodo:

```
#set($inputRoot = $input.path('$'))
{
  "photos": [
#foreach($elem in $inputRoot.photos.photo)
    {
      "id": "$elem.id",
      "photographedBy": "$elem.photographer_first_name $elem.photographer_last_name",
      "title": "$elem.title",
      "ispublic": $elem.public,
      "isfriend": $elem.friend,
      "isfamily": $elem.family
    }#if($foreach.hasNext),#end

#end
  ]
}
```

L'esempio seguente fornisce i dati di output della trasformazione:

```
{
  "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
    }
  ]
}
```

Questi dati vengono inviati alla risposta di metodo e quindi nuovamente al client.

Sostituzione dei parametri della richiesta e della risposta e dei codici di stato per REST API in Gateway API

È possibile utilizzare le trasformazioni dei modelli di mappatura per sostituire qualsiasi tipo di parametro della richiesta, intestazione della risposta o codice di stato della risposta. Per utilizzare il modello di mappatura:

- Eseguire mappature di parametri da molti a uno
- Sostituire i parametri dopo l'applicazione delle mappature standard di Gateway API
- Mappare i parametri in modo condizionale in base al contenuto del corpo o ad altri valori dei parametri
- Creare nuovi parametri in modo programmatico
- Sostituire i codici di stato restituiti dall'endpoint di integrazione

Le sovrascritture sono finali. Una sovrascrittura può essere applicata a ciascun parametro una sola volta. Se si prova a sovrascrivere lo stesso parametro più volte, Gateway API restituisce una risposta 5XX. Se occorre sovrascrivere lo stesso parametro più volte in tutto il modello, ti consigliamo di creare una variabile e applicare la sovrascrittura alla fine del modello. Il modello viene applicato solo dopo che l'intero modello è stato analizzato.

Esempio 1: sostituire il codice di stato in base al corpo di integrazione

L'esempio seguente utilizza l'API di esempio per sostituire il codice di stato in base al corpo della risposta di integrazione.

Console di gestione AWS

Per sostituire un codice di stato in base al corpo della risposta di integrazione

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Seleziona Create API (Crea API).
3. Per API REST, scegliere Crea.
4. Per Dettagli API, scegliere API di esempio.
5. Seleziona Create API (Crea API).

   Gateway API crea un'API PetStore di esempio. Per recuperare le informazioni su un animale domestico, si utilizza la richiesta di metodo dell'API `GET /pets/{petId}`, dove `{petId}` è un parametro di percorso corrispondente al numero ID di un animale domestico.

   In questo esempio, si sostituisce il codice di risposta del metodo `GET` con `400` quando viene rilevata una condizione di errore.
6. Nella struttura Risorse, scegli il metodo `GET` in `/{petId}`.
7. Innanzitutto, esegui il test dell'attuale implementazione dell'API.

   Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
8. Per petId immetti **`-1`**, quindi seleziona Test.

   Il Corpo risposta indica un errore non compreso nell'intervallo:

```
{
  "errors": [
    {
      "key": "GetPetRequest.petId",
      "message": "The value is out of range."
    }
  ]
}
```

   Inoltre, l'ultima riga nella sezione Log termina con: `Method completed with status: 200`.

   L'integrazione è stata completata, ma si è verificato un errore. Ora si sostituisce il codice di stato in base alla risposta di integrazione.

9. Nella scheda Risposta di integrazione scegli Modifica per Predefinito - Risposta.
10. Scegli Modelli di mappatura.
11. Scegliere Add mapping template (Aggiungi modello di mappatura).
12. Per Tipo di contenuto inserisci **`application/json`**.
13. Per Corpo del modello inserisci quanto segue:

```
#set($inputRoot = $input.path('$'))
$input.json("$")
#if($inputRoot.toString().contains("error"))
#set($context.responseOverride.status = 400)
#end
```

    Questo modello di mappatura utilizza la variabile `$context.responseOverride.status` per sostituire il codice di stato con `400` se la risposta di integrazione contiene la stringa `error`.
14. Scegli Save (Salva).
15. Seleziona la scheda Test.
16. Per petId immetti **`-1`**.
17. Nei risultati, il Response Body (Corpo della risposta) indica un errore non compreso nell'intervallo:

```
{
  "errors": [
    {
      "key": "GetPetRequest.petId",
      "message": "The value is out of range."
    }
  ]
}
```

    Tuttavia, l'ultima riga nella sezione Log termina ora con: `Method completed with status: 400`.

CloudFormation

In questo esempio, si utilizza la proprietà body per importare un file di definizione OpenAPI in Gateway API.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Body:
        openapi: 3.0.1
        info:
          title: PetStore Example 1
          description: Example pet store API.
          version: "2025-01-14T00:13:18Z"
        paths:
          /pets/{petId}:
            get:
              parameters:
                - name: petId
                  in: path
                  required: true
                  schema:
                    type: string
              responses:
                "200":
                  description: 200 response
              x-amazon-apigateway-integration:
                httpMethod: GET
                uri: http://petstore.execute-api.us-east-1.amazonaws.com/petstore/
pets/{petId}
                responses:
                  default:
                    statusCode: "200"
                    responseTemplates:
                      application/json: |-
                        #set($inputRoot = $input.path('$'))
                        $input.json("$")
                        #if($inputRoot.toString().contains("error"))
                        #set($context.responseOverride.status = 400)
                        #end
                requestParameters:
                  integration.request.path.petId: method.request.path.petId
                passthroughBehavior: when_no_match
                type: http
        components:
          schemas:
```

```
            Pet:
              type: object
              properties:
                id:
                  type: integer
                type:
                  type: string
                price:
                  type: number
  ApiGatewayDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn: Api
    Properties:
      RestApiId: !Ref Api
  ApiGatewayDeployment20250219:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn: Api
    Properties:
      RestApiId: !Ref Api
  Stage:
    Type: 'AWS::ApiGateway::Stage'
    Properties:
       DeploymentId: !Ref ApiGatewayDeployment20250219
       RestApiId: !Ref Api
       StageName: prod
```

OpenAPI

La seguente definizione OpenAPI crea la risorsa `GET pets/{petId}` e sostituisce il codice di stato in base al corpo di integrazione.

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "PetStore Example 1",
    "description" : "Example pet store API.",
    "version" : "2025-01-14T00:13:18Z"
  },
  "paths" : {
    "/pets/{petId}" : {
      "get" : {
        "parameters" : [ {
          "name" : "petId",
```

```
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response"
          }
        },
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "GET",
          "uri" : "http://petstore.execute-api.us-east-1.amazonaws.com/petstore/
pets/{petId}",
          "responses" : {
            "default" : {
              "statusCode" : "200",
              "responseTemplates" : {
                "application/json" : "#set($inputRoot = $input.path('$'))\n
$input.json(\"$\")\n#if($inputRoot.toString().contains(\"error
\"))\n#set($context.responseOverride.status = 400)\n#end"
              }
            }
          },
          "requestParameters" : {
            "integration.request.path.petId" : "method.request.path.petId"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "http"
        }
      }
    }
  },
  "components" : {
    "schemas" : {
      "Pet" : {
        "type" : "object",
        "properties" : {
          "id" : {
            "type" : "integer"
          },
          "type" : {
            "type" : "string"
```

```
          },
          "price" : {
            "type" : "number"
          }
        }
      }
    }
  }
}
```

Esempio 2: sostituire l'intestazione della richiesta e creare nuove intestazioni

L'esempio seguente utilizza l'API di esempio per sostituire l'intestazione della richiesta e creare nuove intestazioni.

Console di gestione AWS

Per sostituire un'intestazione della richiesta di metodo creando una nuova intestazione

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli l'API di esempio creata nel tutorial precedente. Il nome dell'API dovrebbe essere PetStore.
3. Nella struttura Risorse, scegli il metodo `GET` in `/pet`.
4. Nella scheda Richiesta metodo scegli Modifica in Impostazioni della richiesta del metodo.
5. Scegli Intestazioni di richiesta HTTP, quindi seleziona Aggiungi intestazione.
6. In Nome, inserisci **`header1`**.
7. Scegli Aggiungi intestazione, quindi crea una seconda intestazione chiamata **`header2`**.
8. Scegli Save (Salva).

   Ora, si combinano queste intestazioni in un unico valore di intestazione utilizzando un modello di mappatura.
9. Nella scheda Richiesta di integrazione seleziona Modifica per Impostazioni della richiesta di integrazione.
10. Per Richiesta corpo passthrough scegli Quando non ci sono modelli definiti (consigliato).
11. Scegli Modelli di mappatura e procedi come indicato di seguito:

    a. Scegliere Add mapping template (Aggiungi modello di mappatura).

b. Per Tipo di contenuto inserisci **`application/json`**.

c. Per Corpo del modello inserisci quanto segue:

```
#set($header1Override = "pets")
#set($header3Value = "$input.params('header1')$input.params('header2')")
$input.json("$")
#set($context.requestOverride.header.header3 = $header3Value)
#set($context.requestOverride.header.header1 = $header1Override)
#set($context.requestOverride.header.multivalueheader=[$header1Override,
 $header3Value])
```

Questo modello di mappatura sostituisce `header1` con la stringa `pets` e crea un'intestazione con più valori chiamata `$header3Value` che combina `header1` e `header2`.

12. Scegli Save (Salva).
13. Seleziona la scheda Test.
14. In Intestazioni, copiare il codice seguente:

```
header1:header1Val
header2:header2Val
```

15. Scegli Test (Esegui test).

Nella sezione Log è presente una voce che include questo testo:

```
Endpoint request headers: {header3=header1Valheader2Val,
header2=header2Val, header1=pets, x-amzn-apigateway-api-id=api-id,
Accept=application/json, multivalueheader=pets,header1Valheader2Val}
```

CloudFormation

In questo esempio, si utilizza la proprietà body per importare un file di definizione OpenAPI in Gateway API.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
```

```
      Body:
        openapi: 3.0.1
        info:
          title: PetStore Example 2
          description: Example pet store API.
          version: "2025-01-14T00:36:18Z"
        paths:
          /pets:
            get:
              parameters:
                - name: header2
                  in: header
                  schema:
                    type: string
                - name: page
                  in: query
                  schema:
                    type: string
                - name: type
                  in: query
                  schema:
                    type: string
                - name: header1
                  in: header
                  schema:
                    type: string
              responses:
                "200":
                  description: 200 response
              x-amazon-apigateway-integration:
                httpMethod: GET
                uri: http://petstore.execute-api.us-east-1.amazonaws.com/petstore/
pets
                responses:
                  default:
                    statusCode: "200"
                requestParameters:
                  integration.request.header.header1: method.request.header.header1
                  integration.request.header.header2: method.request.header.header2
                  integration.request.querystring.page:
 method.request.querystring.page
                  integration.request.querystring.type:
 method.request.querystring.type
                requestTemplates:
```

```
                  application/json: |-
                    #set($header1Override = "pets")
                    #set($header3Value =
"$input.params('header1')$input.params('header2')")
                    $input.json("$")
                    #set($context.requestOverride.header.header3 = $header3Value)
                    #set($context.requestOverride.header.header1 = $header1Override)

#set($context.requestOverride.header.multivalueheader=[$header1Override,
$header3Value])
                passthroughBehavior: when_no_match
                type: http
        components:
          schemas:
            Pet:
              type: object
              properties:
                id:
                  type: integer
                type:
                  type: string
                price:
                  type: number
 ApiGatewayDeployment:
   Type: 'AWS::ApiGateway::Deployment'
   DependsOn: Api
   Properties:
     RestApiId: !Ref Api
 ApiGatewayDeployment20250219:
   Type: 'AWS::ApiGateway::Deployment'
   DependsOn: Api
   Properties:
     RestApiId: !Ref Api
 Stage:
   Type: 'AWS::ApiGateway::Stage'
   Properties:
      DeploymentId: !Ref ApiGatewayDeployment20250219
      RestApiId: !Ref Api
      StageName: prod
```

OpenAPI

La seguente definizione OpenAPI crea la risorsa `GET pets`, sostituisce l'intestazione della richiesta e crea nuove intestazioni.

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "PetStore Example 2",
    "description" : "Example pet store API.",
    "version" : "2025-01-14T00:36:18Z"
  },
  "paths" : {
    "/pets" : {
      "get" : {
        "parameters" : [ {
          "name" : "header2",
          "in" : "header",
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "page",
          "in" : "query",
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "type",
          "in" : "query",
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "header1",
          "in" : "header",
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response"
          }
        },
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "GET",
```

```
          "uri" : "http://petstore.execute-api.us-east-1.amazonaws.com/petstore/
pets",
          "responses" : {
            "default" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.header.header1" : "method.request.header.header1",
            "integration.request.header.header2" : "method.request.header.header2",
            "integration.request.querystring.page" :
 "method.request.querystring.page",
            "integration.request.querystring.type" :
 "method.request.querystring.type"
          },
          "requestTemplates" : {
            "application/json" : "#set($header1Override = \"pets
\")\n#set($header3Value = \"$input.params('header1')$input.params('header2')\")\n
$input.json(\"$\")\n#set($context.requestOverride.header.header3
 = $header3Value)\n#set($context.requestOverride.header.header1 =
 $header1Override)\n#set($context.requestOverride.header.multivalueheader=[$header1Override,
 $header3Value])"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "http"
        }
      }
    }
  }
}
```

Per utilizzare la sostituzione di un modello di mappatura, si aggiunge una o più delle seguenti variabili `$context`. Per l'elenco delle variabili `$context`, consultare the section called "Variabili di contesto per le trasformazioni dei dati".

Tutorial: modifica la richiesta di integrazione e la risposta per le integrazioni a AWS services

Il seguente tutorial mostra come utilizzare le trasformazioni dei modelli di mappatura per configurare i modelli di mappatura per trasformare le richieste e le risposte di integrazione utilizzando la console e la CLI. AWS

Argomenti

- Configurare la trasformazione dei dati utilizzando la console Gateway Amazon API
- Imposta la trasformazione dei dati utilizzando AWS CLI
- Trasformazione dei dati completata CloudFormation modello

Configurare la trasformazione dei dati utilizzando la console Gateway Amazon API

In questo tutorial, verranno create un'API e una tabella DynamoDB incomplete utilizzando il seguente file.zip data-transformation-tutorial-console.zip. Questa API incompleta dispone di una risorsa `/pets` con i metodi `GET` e `POST`.

- Il metodo `GET` otterrà i dati dall'endpoint HTTP `http://petstore-demo-endpoint.execute-api.com/petstore/pets`. I dati di output verranno trasformati in base al modello di mappatura in the section called “Trasformazioni dei modelli di mappatura”.
- Il metodo `POST` consentirà all'utente di pubblicare (`POST`) le informazioni sugli animali domestici in una tabella Amazon DynamoDB utilizzando un modello di mappatura.

Scarica e decomprimi il modello di creazione dell'app per. CloudFormation Verrà utilizzato questo modello per creare una tabella DynamoDB per pubblicare informazioni sugli animali domestici e un'API incompleta. Il resto della procedura verrà completato nella console Gateway Amazon API.

Per creare un CloudFormation stack

1. Apri la CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation.
2. Scegliere Create stack (Crea stack), quindi With new resources (standard) (Con nuove risorse (standard)).
3. In Specificare modello, scegliere Carica un file modello.
4. Selezionare il modello scaricato.
5. Scegli Next (Successivo).
6. Per Stack name (Nome stack), inserire **`data-transformation-tutorial-console`**, quindi scegliere Next (Avanti).
7. Per Configure stack options (Configura opzioni di stack), scegliere Next (Successivo).
8. Per quanto riguarda le funzionalità, riconosci che CloudFormation puoi creare risorse IAM nel tuo account.
9. Scegli Avanti, quindi scegli Invia.

CloudFormation fornisce le risorse specificate nel modello. Per completare il provisioning delle risorse, potrebbero essere necessari alcuni minuti. Quando lo stato del tuo CloudFormation stack è CREATE_COMPLETE, sei pronto per passare alla fase successiva.

Per testare la risposta di integrazione GET

1. Nella scheda Risorse dello CloudFormation stack di**data-transformation-tutorial-console**, seleziona l'ID fisico della tua API.
2. Nel pannello di navigazione principale scegli Risorse, quindi seleziona il metodo GET.
3. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.

   L'output del test mostrerà quanto segue:

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

   Questo output verrà trasformato in base al modello di mappatura in the section called “Trasformazioni dei modelli di mappatura”.

Per trasformare la risposta di integrazione **GET**

1. Scegli la scheda Risposta di integrazione.

Attualmente non sono definiti modelli di mappatura, quindi la risposta di integrazione non verrà trasformata.

2. Per Predefinito - Risposta scegli Modifica.
3. Scegli Modelli di mappatura e procedi come indicato di seguito:

   a. Scegliere Add mapping template (Aggiungi modello di mappatura).

   b. Per Tipo di contenuto inserisci **`application/json`**.

   c. Per Corpo del modello inserisci quanto segue:

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

#end
]
```

   Scegli Save (Salva).

Per testare la risposta di integrazione **`GET`**

- Scegli la scheda Test, quindi seleziona Test.

  L'output del test mostrerà la risposta trasformata.

```
[
  {
    "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
  }
]
```

Per trasformare i dati di input dal metodo **POST**

1. Scegli il metodo POST.
2. Scegli la scheda Richiesta di integrazione, quindi seleziona Modifica per Impostazioni della richiesta di integrazione.

   Il CloudFormation modello ha compilato alcuni campi di richiesta di integrazione.

   - Il tipo di integrazione è Servizio AWS.
   - Servizio AWS Si tratta di DynamoDB.
   - Il metodo HTTP è `POST`.
   - L'operazione è `PutItem`.
   - Il ruolo di esecuzione che consente ad API Gateway di inserire un elemento nella tabella DynamoDB è. `data-transformation-tutorial-console-APIGatewayRole` CloudFormation ha creato questo ruolo per consentire ad API Gateway di disporre delle autorizzazioni minime per interagire con DynamoDB.

   Il nome della tabella DynamoDB non è stato specificato. Il nome verrà specificato nei passaggi seguenti.
3. Per Richiesta corpo passthrough seleziona Mai.

   Ciò significa che l'API rifiuterà i dati Content-Types che non dispongono di un modello di mappatura.
4. Scegli Modelli di mappatura.
5. Tipo di contenuto è impostato su `application/json`. Ciò significa che i tipi di contenuto che non lo sono application/json verranno rifiutati dall'API. Per ulteriori informazioni sui comportamenti passthrough di integrazione, consultare Comportamento della richiesta di metodo per i payload senza modelli di mappatura per REST API in Gateway API.
6. Inserire il codice seguente nell'editor di testo.

```
{
```

```
    "TableName":"data-transformation-tutorial-console-ddb",
    "Item": {
        "id": {
            "N": $input.json("$.id")
        },
        "type": {
            "S": $input.json("$.type")
        },
        "price": {
            "N": $input.json("$.price")
        }
    }
}
```

Questo modello specifica la tabella come `data-transformation-tutorial-console-ddb` e imposta gli elementi come `id`, `type` e `price`. Gli elementi proverranno dal corpo del metodo `POST`. È anche possibile utilizzare un modello di dati per creare un modello di mappatura. Per ulteriori informazioni, consulta Convalida delle richieste per REST API in Gateway API.

7. Scegliere il pulsante Salva per salvare il modello di mappatura.

Per aggiungere un metodo e una risposta di integrazione dal metodo **`POST`**

Hanno CloudFormation creato un metodo vuoto e una risposta di integrazione. Questa risposta verrà modificata per fornire ulteriori informazioni. Per ulteriori informazioni su come modificare le risposte, consultare Esempi di mappatura dei parametri per REST API in Gateway API.

1. Nella scheda Risposta di integrazione scegli Modifica per Predefinito - Risposta.
2. Scegli Modelli di mappatura, quindi seleziona Aggiungi modello di mappatura.
3. In Content-type, immettere **`application/json`**.
4. Nell'editor di codice, inserire il seguente modello di mappatura di output per inviare un messaggio di output:

```
{ "message" : "Your response was recorded at $context.requestTime" }
```

Per ulteriori informazioni sulle variabili di contesto, consultare Variabili di contesto per le trasformazioni dei dati.

5. Scegliere il pulsante Salva per salvare il modello di mappatura.

Prova il metodo **POST**

Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.

1. Nel corpo della richiesta, inserire il seguente esempio.

```
{
          "id": "4",
          "type" : "dog",
          "price": "321"
}
```

2. Scegli Test (Esegui test).

   L'output dovrebbe mostrare il messaggio di operazione riuscita.

   Puoi aprire la console DynamoDB https://console.aws.amazon.com/dynamodb/all'indirizzo per verificare che l'elemento di esempio sia nella tua tabella.

Per eliminare un CloudFormation stack

1. Apri la CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformation.
2. Seleziona il tuo CloudFormation stack.
3. Scegli Elimina e conferma la tua scelta.

Imposta la trasformazione dei dati utilizzando AWS CLI

In questo tutorial, verranno create un'API e una tabella DynamoDB incomplete utilizzando il seguente file.zip data-transformation-tutorial-cli.zip. Questa API incompleta dispone di una risorsa `/pets` con un metodo `GET` integrato con l'endpoint HTTP `http://petstore-demo-endpoint.execute-api.com/petstore/pets`. Verrà creato un metodo `POST` per connettersi a una tabella DynamoDB e si useranno modelli di mappatura per inserire dati in una tabella DynamoDB.

- I dati di output verranno quindi trasformati in base al modello di mappatura in the section called “Trasformazioni dei modelli di mappatura”.
- Verrà creato un metodo `POST` per consentire all'utente di pubblicare (`POST`) le informazioni sugli animali domestici in una tabella Amazon DynamoDB utilizzando un modello di mappatura.

Per creare un CloudFormation stack

Scarica e decomprimi il modello di creazione dell'app per CloudFormation.

Per completare il tutorial seguente, è necessario disporre della AWS Command Line Interface (AWS CLI) versione 2.

Per i comandi lunghi viene utilizzato un carattere di escape (\) per dividere un comando su più righe.

> **Note**
>
> In Windows, alcuni comandi della CLI Bash utilizzati comunemente (ad esempio, `zip`) non sono supportati dai terminali integrati del sistema operativo. Per ottenere una Windows-integrated versione di Ubuntu e Bash, installa il sottosistema Windows per Linux. I comandi della CLI di esempio in questa guida utilizzano la formattazione Linux. Se si utilizza la CLI di Windows, i comandi che includono documenti JSON in linea dovranno essere riformattati.

1. Usa il seguente comando per creare lo stack. CloudFormation

```
aws cloudformation create-stack --stack-name data-transformation-tutorial-cli
 --template-body file://data-transformation-tutorial-cli.zip --capabilities
 CAPABILITY_NAMED_IAM
```

2. CloudFormation fornisce le risorse specificate nel modello. Per completare il provisioning delle risorse, potrebbero essere necessari alcuni minuti. Usa il comando seguente per vedere lo stato del tuo CloudFormation stack.

```
aws cloudformation describe-stacks --stack-name data-transformation-tutorial-cli
```

3. Quando lo stato dello CloudFormation stack è`StackStatus: "CREATE_COMPLETE"`, usa il seguente comando per recuperare i valori di output pertinenti per i passaggi futuri.

```
aws cloudformation describe-stacks --stack-name data-transformation-tutorial-cli
--query "Stacks[*].Outputs[*].{OutputKey: OutputKey, OutputValue: OutputValue,
Description: Description}"
```

   Di seguito sono riportati i valori di output:

- ApiRole, che è il nome del ruolo che consente ad API Gateway di inserire elementi nella tabella DynamoDB. Per questo tutorial, il nome del ruolo è `data-transformation-tutorial-cli-APIGatewayRole-ABCDEFG`.
- DDBTableName, che è il nome della tabella DynamoDB. Per questo tutorial il nome della tabella è `data-transformation-tutorial-cli-ddb`.
- ResourceId, che è l'ID della risorsa pets in cui sono esposti i `POST` metodi `GET` and. Per questo tutorial, l'ID risorsa è `efg456`.
- ApiId, che è l'ID dell'API. Per questo tutorial, l'ID API è `abc123`.

Per testare il metodo **`GET`** prima della trasformazione dei dati

- Utilizzare il seguente comando per eseguire il test del metodo `GET`.

```
aws apigateway test-invoke-method --rest-api-id abc123 \
          --resource-id efg456 \
          --http-method GET
```

L'output del test riporterà le seguenti informazioni.

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

Questo output verrà trasformato in base al modello di mappatura in the section called "Trasformazioni dei modelli di mappatura".

Per trasformare la risposta di integrazione **GET**

- Utilizzare il comando seguente per aggiornare una risposta di integrazione per il metodo GET. Sostituisci *rest-api-id* e *resource-id* con i tuoi valori.

  Per creare una risposta di integrazione, utilizzare il comando seguente.

```
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]"}'
```

Per testare il metodo **GET**

- Utilizzare il seguente comando per eseguire il test del metodo GET.

```
aws apigateway test-invoke-method --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method GET \
```

  L'output del test mostrerà la risposta trasformata.

```
[
  {
    "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
  }
]
```

Per creare un metodo **POST**

1. Utilizzare il seguente comando per creare un nuovo metodo nella risorsa `/pets`.

```
aws apigateway put-method --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method POST \
  --authorization-type "NONE" \
```

   Questo metodo ti consentirà di inviare informazioni sugli animali domestici alla tabella DynamoDB che hai creato nello stack. CloudFormation

2. Utilizzate il seguente comando per creare un' Servizio AWS integrazione sul metodo. `POST`

```
aws apigateway put-integration --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method POST \
  --type AWS \
  --integration-http-method POST \
  --uri "arn:aws:apigateway:us-east-2:dynamodb:action/PutItem" \
  --credentials arn:aws:iam::111122223333:role/data-transformation-tutorial-cli-APIGatewayRole-ABCDEFG \
  --request-templates '{"application/json":"{\"TableName\":\"data-transformation-tutorial-cli-ddb\",\"Item\":{\"id\":{\"N\":$input.json(\"$.id\")},\"type\":{\"S\":$input.json(\"$.type\")},\"price\":{\"N\":$input.json(\"$.price\")} }}"}'
```

3. Utilizzare il comando seguente per creare una risposta del metodo per una corretta chiamata del metodo `POST`.

```
aws apigateway put-method-response --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method POST \
  --status-code 200
```

4. Utilizzare il comando seguente per creare una risposta dell'integrazione per una corretta chiamata del metodo POST.

```
aws apigateway put-integration-response --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method POST \
  --status-code 200 \
  --selection-pattern "" \
  --response-templates '{"application/json": "{\"message\": \"Your response was
 recorded at $context.requestTime\"}"}'
```

Per testare il metodo **POST**

- Utilizzare il seguente comando per eseguire il test del metodo POST.

```
aws apigateway test-invoke-method --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method POST \
  --body '{\"id\": \"4\", \"type\": \"dog\", \"price\": \"321\"}'
```

L'output mostrerà il messaggio di operazione riuscita.

Per eliminare un CloudFormation stack

- Usa il seguente comando per eliminare CloudFormation le tue risorse.

```
aws cloudformation delete-stack  --stack-name data-transformation-tutorial-cli
```

Trasformazione dei dati completata CloudFormation modello

L'esempio seguente è un CloudFormation modello completo, che crea un'API e una tabella DynamoDB con `/pets` una risorsa `GET` con metodi and. `POST`

- Il metodo `GET` recupererà i dati dall'endpoint HTTP `http://petstore-demo-endpoint.execute-api.com/petstore/pets`. I dati di output verranno trasformati in base al modello di mappatura in the section called “Trasformazioni dei modelli di mappatura”.
- Il metodo `POST` consentirà all'utente di pubblicare (`POST`) le informazioni sugli animali domestici in una tabella DynamoDB utilizzando un modello di mappatura.

Esempio CloudFormation modello

```
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
```

Esempi di utilizzo di variabili per trasformazioni dei modelli di mappatura per Gateway API

L'esempio seguente mostra come utilizzare le variabili `$context`, `input` e `util` nei modelli di mappatura. È possibile utilizzare un'integrazione fittizia o un'integrazione non proxy Lambda

che restituisce l'evento di input a Gateway API. Per l'elenco di tutte le variabili supportate per le trasformazioni dei dati, consultare the section called "Variabili per le trasformazioni dei dati".

Esempio 1: Passare più variabili **`$context all'endpoint`** di integrazione

L'esempio seguente mostra un modello di mappatura che mappa variabili `$context` in ingresso a variabili back-end con nomi leggermente diversi nel payload di una richiesta di integrazione:

```
{
    "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"
}
```

L'output di questo modello di mappatura deve essere analogo al seguente:

```
{
  stage: 'prod',
  request_id: 'abcdefg-000-000-0000-abcdefg',
  api_id: 'abcd1234',
  resource_path: '/',
  resource_id: 'efg567',
  http_method: 'GET',
  source_ip: '192.0.2.1',
  user-agent: 'curl/7.84.0',
  account_id: '111122223333',
  api_key: 'MyTestKey',
  caller: 'ABCD-0000-12345',
  user: 'ABCD-0000-12345',
  user_arn: 'arn:aws:sts::111122223333:assumed-role/Admin/carlos-salazar'
}
```

Una delle variabili è una chiave API. L'esempio presuppone che il metodo richieda una chiave API.

Esempio 2: passare tutti i parametri della richiesta all'endpoint di integrazione tramite un payload JSON

L'esempio seguente passa tutti i parametri della richiesta, inclusi `path`, `querystring` e `header`, all'endpoint di integrazione tramite un 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
  }
}
```

Se una richiesta ha i seguenti parametri di input:

- Un parametro di percorso denominato `myparam`
- Parametri della stringa di query `querystring1=value1,value2`
- Intestazioni `"header1" : "value1"`.

L'output di questo modello di mappatura deve essere analogo al seguente:

```
{"params":{"path":{"example2":"myparamm"},"querystring":
{"querystring1":"value1,value2"},"header":{"header1":"value1"}}}
```

Esempio 3: passare una sezione secondaria di una richiesta di metodo all'endpoint di integrazione

L'esempio seguente utilizza il parametro di input `name` per recuperare solo il parametro `name` e il parametro di input `input.json('$')` per recuperare l'intero corpo della richiesta di metodo:

```
{
    "name" : "$input.params('name')",
```

```
    "body" : $input.json('$')
}
```

Per una richiesta che include i parametri della stringa di query `name=Bella&type=dog` e il corpo seguente:

```
{
    "Price" : "249.99",
    "Age": "6"
}
```

L'output di questo modello di mappatura deve essere analogo al seguente:

```
{
    "name" : "Bella",
    "body" : {"Price":"249.99","Age":"6"}
}
```

Questo modello di mappatura rimuove il parametro della stringa di query `type=dog`.

Se l'input JSON contiene caratteri senza escape che non possono essere analizzati, API JavaScript Gateway potrebbe restituire una risposta 400. Applica `$util.escapeJavaScript($input.json('$'))` per assicurare che l'input JSON possa essere analizzato correttamente.

L'esempio precedente con `$util.escapeJavaScript($input.json('$'))` applicato risulta come segue:

```
{
    "name" : "$input.params('name')",
    "body" : "$util.escapeJavaScript($input.json('$'))"
}
```

In questo caso, l'output di questo modello di mappatura deve essere analogo al seguente:

```
{
    "name" : "Bella",
    "body": {"Price":"249.99","Age":"6"}
}
```

Esempio 4: utilizzare l'espressione JSONPath per passare una sezione secondaria di una richiesta di metodo all'endpoint di integrazione

L'esempio seguente utilizza le espressioni JSONPath per recuperare solo il parametro di input `name` e `Age` dal corpo della richiesta:

```
{
    "name" : "$input.params('name')",
    "body" : $input.json('$.Age')
}
```

Per una richiesta che include i parametri della stringa di query `name=Bella&type=dog` e il corpo seguente:

```
{
    "Price" : "249.99",
    "Age": "6"
}
```

L'output di questo modello di mappatura deve essere analogo al seguente:

```
{
    "name" : "Bella",
    "body" : "6"
}
```

Questo modello di mappatura rimuove il parametro della stringa di query `type=dog` e il campo `Price` dal corpo.

Se il payload di una richiesta di metodo contiene caratteri senza escape che non possono essere analizzati, API JavaScript Gateway potrebbe restituire una risposta. `400` Applica `$util.escapeJavaScript()` per assicurare che l'input JSON possa essere analizzato correttamente.

L'esempio precedente con `$util.escapeJavaScript($input.json('$.Age'))` applicato risulta come segue:

```
{
    "name" : "$input.params('name')",
    "body" : "$util.escapeJavaScript($input.json('$.Age'))"
```

```
}
```

In questo caso, l'output di questo modello di mappatura deve essere analogo al seguente:

```
{
    "name" : "Bella",
    "body": "\"6\""
}
```

Esempio 5: utilizzare un'espressione JSONPath per passare informazioni su una richiesta di metodo all'endpoint di integrazione

L'esempio seguente utilizza `$input.params()`, `$input.path()` e `$input.json()` per inviare informazioni su una richiesta di metodo all'endpoint di integrazione. Questo modello di mappatura utilizza il metodo `size()` per fornire il numero di elementi di un elenco.

```
{
    "id" : "$input.params('id')",
    "count" : "$input.path('$.things').size()",
    "things" : $input.json('$.things')
}
```

Per una richiesta che include il parametro di percorso `123` e il corpo seguente:

```
{
      "things": {
            "1": {},
            "2": {},
            "3": {}
      }
}
```

L'output di questo modello di mappatura deve essere analogo al seguente:

```
{"id":"123","count":"3","things":{"1":{},"2":{},"3":{}}}
```

Se il payload di una richiesta di metodo contiene caratteri senza escape che non possono essere analizzati, API JavaScript Gateway potrebbe restituire una risposta. `400` Applica `$util.escapeJavaScript()` per assicurare che l'input JSON possa essere analizzato correttamente.

L'esempio precedente con `$util.escapeJavaScript($input.json('$.things'))` applicato risulta come segue:

```
{
     "id" : "$input.params('id')",
     "count" : "$input.path('$.things').size()",
     "things" : "$util.escapeJavaScript($input.json('$.things'))"
}
```

L'output di questo modello di mappatura deve essere analogo al seguente:

```
{"id":"123","count":"3","things":"{\"1\":{},\"2\":{},\"3\":{}}"}
```

Variabili per le trasformazioni dei dati per Gateway API

Quando si crea una mappatura dei parametri, è possibile utilizzare le variabili di contesto come origine dati. Per creare le trasformazioni di modelli di mappatura, è possibile utilizzare variabili di contesto, input e util negli script scritti in Velocity Template Language (VTL). Per modelli di mappatura di esempio che utilizzano queste variabili di riferimento, consultare the section called “Esempi di utilizzo di variabili per trasformazioni dei modelli di mappatura”.

Per l’elenco delle variabili di riferimento per la registrazione degli accessi, consultare the section called “Variabili per la registrazione dei log degli accessi per Gateway API”.

Variabili di contesto per le trasformazioni dei dati

È possibile utilizzare le seguenti variabili `$context` con distinzione tra maiuscole e minuscole per le trasformazioni dei dati.

Parametro	Description
`$context.accountId`	L'ID dell'account del proprietario dell'API. AWS
`$context.apiId`	Identificatore assegnato da API Gateway all'API.
`$context.authorizer.claims.`*`property`*	Proprietà delle richieste restituite dal pool di utenti di Amazon Cognito dopo che l'interme diario del metodo viene autenticato correttam ente. Per ulteriori informazioni, consulta the

Parametro	Description
	section called “Utilizza un pool di utenti di Amazon Cognito come autorizzazione per un'API REST”. **Note** La chiamata di `$context.authorizer.claims` restituisce null.
`$context.authorizer.principalId`	Identificazione dell'utente dell'entità principale associata al token inviata dal client e restituita da un'autorizzazione Lambda in API Gateway (precedentemente noto come autorizzazione ad hoc). Per ulteriori informazioni, consulta the section called “Uso di autorizzazioni Lambda”.

Parametro	Description
`$context.authorizer.` *`property`*	Valore in formato stringa della coppia chiave/valore specificata della mappa `context` restituita da una funzione delle autorizzazioni Lambda di API Gateway. Ad esempio, se le autorizzazioni restituiscono la mappa `context` seguente: `"context" : { "key": "value", "numKey": 1, "boolKey": true }` La chiamata di `$context.authorizer.key` restituisce la stringa `"value"`, la chiamata di `$context.authorizer.numKey` restituisce la stringa `"1"` e la chiamata di `$context.authorizer.boolKey` restituisce la stringa `"true"`. Infatti*`property`*, l'unico carattere speciale supportato è il carattere di sottolineatura(`_`). Per ulteriori informazioni, consulta the section called “Uso di autorizzazioni Lambda”.
`$context.awsEndpointRequestId`	L'ID della richiesta dell' AWS endpoint.
`$context.deploymentId`	ID dell'implementazione API.
`$context.domainName`	Nome di dominio completo usato per richiamare l'API. Deve essere lo stesso dell'intestazione `Host` in ingresso.
`$context.domainPrefix`	Prima etichetta di `$context.domainName` .

Parametro	Description
`$context.error.message`	Stringa contenente un messaggio di errore di API Gateway. Questa variabile può essere utilizzata solo per la semplice sostituzione di variabili in un modello di GatewayResponsebody mapping, che non viene elaborato dal motore Velocity Template Language, e nella registrazione degli accessi. Per ulteriori informazioni, consultare the section called “Metriche” e the section called “Configurazione delle risposte del gateway per la personalizzazione delle risposte agli errori”.
`$context.error.messageString`	Valore `$context.error.message` tra virgolette, ovvero `"$context.error.message"`.
`$context.error.responseType`	Un tipo di. GatewayResponse Questa variabile può essere utilizzata solo per la semplice sostituzione di variabili in un modello di GatewayResponsebody mapping, che non viene elaborato dal motore Velocity Template Language, e nella registrazione degli accessi. Per ulteriori informazioni, consultare the section called “Metriche” e the section called “Configurazione delle risposte del gateway per la personalizzazione delle risposte agli errori”.
`$context.error.validationErrorString`	Stringa contenente un messaggio dettagliato di errore di convalida.
`$context.extendedRequestId`	ID esteso generato da API Gateway e assegnato alla richiesta API. L’ID della richiesta esteso contiene ulteriori informazioni utili per il debug e la risoluzione dei problemi.

Parametro	Description
`$context.httpMethod`	Metodo HTTP usato. I valori validi sono: `DELETE`, `GET`, `HEAD`, `OPTIONS`, `PATCH`, `POST` e `PUT`.
`$context.identity.accountId`	L'ID dell'account associato alla richiesta. AWS
`$context.identity.apiKey`	Per i metodi API che richiedono una chiave API, questa variabile è la chiave API associata alla richiesta del metodo. Per i metodi che non richiedono una chiave API, questa variabile è null. Per ulteriori informazioni, consulta the section called “Piani di utilizzo”.
`$context.identity.apiKeyId`	ID chiave API associato a una richiesta API che richiede una chiave API.
`$context.identity.caller`	Identificatore dell'entità principale dell'inte rmediario che ha firmato la richiesta. Supportat o per risorse che utilizzano l'autorizzazione IAM.

Parametro	Description
`$context.identity.cognitoAuthenticationProvider`	Un elenco separato da virgole con tutti i provider di autenticazione Amazon Cognito utilizzati dal chiamante che effettua la richiesta . Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito. Ad esempio, per un'identità di un pool di utenti Amazon Cognito, `cognito-idp.` *`region`*`.amazonaws.com/` *`user_pool_id`* `,cognito-idp.` *`region`*`.amazonaws.com/` *`user_pool_id`* `:CognitoSignIn:` *`token subject claim`* Per informazioni sui provider di autenticazione Amazon Cognito disponibili, consulta Using Federated Identities nella Guida per gli sviluppatori di Amazon Cognito.
`$context.identity.cognitoAuthenticationType`	Tipo di autenticazione Amazon Cognito dell'intermediario da cui proviene la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito. I valori possibili includono `authenticated` per le identità autenticate e `unauthenticated` per le identità non autenticate.
`$context.identity.cognitoIdentityId`	ID identità di Amazon Cognito dell'intermediario da cui proviene la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito.
`$context.identity.cognitoIdentityPoolId`	ID pool di identità di Amazon Cognito dell'intermediario da cui proviene la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito.

Parametro	Description
`$context.identity.principalOrgId`	L'ID organizzazione AWS.
`$context.identity.sourceIp`	L'indirizzo IP di origine della connessione TCP immediata da cui proviene la richiesta all'endpoint di Gateway API.
`$context.identity.clientCert.clientCertPem`	Il certificato PEM-encoded client presentato dal client durante l'autenticazione TLS reciproca. Presente quando un client accede a un'API utilizzando un nome di dominio personalizzato che ha attivato l'autenticazione TLS reciproca. Presente solo nei log di accesso se l'autenticazione TLS reciproca non riesce.
`$context.identity.clientCert.subjectDN`	Nome distinto dell'oggetto del certificato presentato da un client. Presente quando un client accede a un'API utilizzando un nome di dominio personalizzato che ha attivato l'autenticazione TLS reciproca. Presente solo nei log di accesso se l'autenticazione TLS reciproca non riesce.
`$context.identity.clientCert.issuerDN`	Nome distinto dell'approvatore del certificato presentato da un cliente. Presente quando un client accede a un'API utilizzando un nome di dominio personalizzato che ha attivato l'autenticazione TLS reciproca. Presente solo nei log di accesso se l'autenticazione TLS reciproca non riesce.
`$context.identity.clientCert.serialNumber`	Il numero di serie del certificato. Presente quando un client accede a un'API utilizzando un nome di dominio personalizzato che ha attivato l'autenticazione TLS reciproca. Presente solo nei log di accesso se l'autenticazione TLS reciproca non riesce.

Parametro	Description
`$context.identity.clientCert.validity.notBefore`	La data prima della quale il certificato non è valido. Presente quando un client accede a un'API utilizzando un nome di dominio personalizzato che ha attivato l'autenticazione TLS reciproca. Presente solo nei log di accesso se l'autenticazione TLS reciproca non riesce.
`$context.identity.clientCert.validity.notAfter`	La data dopo la quale il certificato non è valido. Presente quando un client accede a un'API utilizzando un nome di dominio personalizzato che ha attivato l'autenticazione TLS reciproca . Presente solo nei log di accesso se l'autenticazione TLS reciproca non riesce.
`$context.identity.vpcId`	L'ID VPC del VPC da cui proviene la richiesta all'endpoint Gateway API.
`$context.identity.vpceId`	L'ID endpoint VPC dell'endpoint VPC da cui proviene la richiesta all'endpoint Gateway API. Presente solo quando si dispone di un'API privata.
`$context.identity.user`	Identificatore dell'entità principale dell'utente che sarà autorizzato per l'accesso alle risorse. Supportato per risorse che utilizzano l'autorizzazione IAM.
`$context.identity.userAgent`	Intestazione `User-Agent` del chiamante API.
`$context.identity.userArn`	Amazon Resource Name (ARN) dell'utente valido identificato dopo l'autenticazione. Per ulteriori informazioni, consulta https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html.

Parametro	Description
`$context.isCanaryRequest`	Restituisce `true` se la richiesta era indirizzata al canary e `false` se la richiesta non era indirizzata al canary. Presente solo quando è abilitato un canary.
`$context.path`	Percorso della richiesta. Ad esempio, per un URL di richiesta non proxy `https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/root/child`, il valore di `$context.path` è `/{stage}/root/child`.
`$context.protocol`	Protocollo della richiesta, ad esempi, `HTTP/1.1`. **Note** Le API API Gateway possono accettare HTTP/2 richieste, ma API Gateway invia le richieste alle integrazioni di backend utilizzando. HTTP/1.1 Di conseguenza, il protocollo di richiesta viene registrato HTTP/1.1 anche se un client invia una richiesta che utilizza. HTTP/2
`$context.requestId`	L'ID della richiesta. I client possono sovrascrivere questo ID di richiesta. Utilizza `$context.extendedRequestId` per un ID di richiesta univoco generato da API Gateway.

Parametro	Description
`$context.requestOverride.header. header_name`	Sovrascrittura intestazione della richiesta. Se definito, questo parametro contiene le intestazioni da utilizzare al posto delle intestazioni HTTP che sono definite nel riquadro Integration Request (Richiesta di integrazione). Per ulteriori informazioni, consulta Sostituzione dei parametri della richiesta e della risposta e dei codici di stato per REST API in Gateway API.
`$context.requestOverride.path. path_name`	Sovrascrittura percorso della richiesta. Se definito, questo parametro contiene il percorso della richiesta da utilizzare al posto dei parametri di percorso URL che sono definiti nel riquadro Integration Request (Richiesta di integrazione). Per ulteriori informazioni, consulta Sostituzione dei parametri della richiesta e della risposta e dei codici di stato per REST API in Gateway API.
`$context.requestOverride.querystring. querystring_name`	Sovrascrittura stringa di query della richiesta. Se definito, questo parametro contiene la stringa di query della richiesta da utilizzare al posto dei URL Query String Parameters (Parametri stringa di query URL) che sono definiti nel riquadro Integration Request (Richiesta di integrazione). Per ulteriori informazioni, consulta Sostituzione dei parametri della richiesta e della risposta e dei codici di stato per REST API in Gateway API.

Parametro	Description
`$context.responseOverride.header. header_name`	Sovrascrittura intestazione della risposta. Se definito, questo parametro contiene l'intestazione da restituire al posto della Response header (Intestazione di risposta) che è definita come la Default mapping (mappatura predefinita) nel riquadro Integration Response (Risposta integrazione). Per ulteriori informazioni, consulta Sostituzione dei parametri della richiesta e della risposta e dei codici di stato per REST API in Gateway API.
`$context.responseOverride.status`	Sovrascrittura codice di stato della risposta. Se definito, questo parametro contiene il codice di stato da restituire al posto di Method response status (Stato risposta metodo) che è definito come la Default mapping (Mappatura predefinita) nel riquadro Integration Response (Risposta integrazione). Per ulteriori informazioni, consulta Sostituzione dei parametri della richiesta e della risposta e dei codici di stato per REST API in Gateway API.
`$context.requestTime`	Ora della richiesta in formato CLF (`dd/MMM/yyyy:HH:mm:ss +-hhmm`).
`$context.requestTimeEpoch`	L'ora della richiesta in formato epoca (Unix epoch) in millisecondi.
`$context.resourceId`	Identificatore assegnato da API Gateway alla risorsa.

Parametro	Description
`$context.resourcePath`	Percorso della risorsa. Ad esempio, per l'URI della richiesta non proxy di `https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/root/child`, il valore di `$context.resourcePath` è `/root/child`. Per ulteriori informazioni, consulta Tutorial: creazione di una REST API con un'integrazione non proxy HTTP.
`$context.stage`	La fase di distribuzione della richiesta API (ad esempio `Beta` o `Prod`).
`$context.wafResponseCode`	La risposta ricevuta da AWS WAF: `WAF_ALLOW` o `WAF_BLOCK`. Non verrà impostata se la fase non è associata a un ACL Web. Per ulteriori informazioni, consulta the section called “AWS WAF”.
`$context.webaclArn`	ARN completo della lista di controllo accessi Web usata per stabilire se consentire o bloccare la richiesta. Non verrà impostata se la fase non è associata a un ACL Web. Per ulteriori informazioni, consulta the section called “AWS WAF”.

Variabili input

È possibile utilizzare le seguenti variabili `$input` con distinzione tra maiuscole e minuscole per fare riferimento ai parametri e al payload della richiesta di metodo. Sono disponibili le funzioni seguenti:

Variabile e funzione	Descrizione
`$input.body`	Restituisce il payload della richiesta non elaborata come stringa. È possibile utilizzare

Variabile e funzione	Descrizione
	`$input.body` per conservare interi numeri in virgola mobile, ad esempio `10.00`.
`$input.json(x)`	Questa funzione valuta un'espressione JSONPath e restituisce i risultati come stringa JSON. Ad esempio, `$input.json('$.pets')` restituisce una stringa JSON che rappresenta la struttura `pets`. Per ulteriori informazioni su JSONPath, consulta la pagina relativa a JSONPath o JSONPath per Java.
`$input.params()`	Restituisce una mappa di tutti i parametri della richiesta. Si consiglia di utilizzare `$util.escapeJavaScript` per sanificare il risultato ed evitare un potenziale attacco di iniezione. Per il pieno controllo della sanificazione delle richieste, utilizzare un'integrazione proxy senza un modello e gestire la sanificazione delle richieste nell'integrazione.
`$input.params(x)`	Restituisce il valore di un parametro della richiesta del metodo dal percorso, dalla stringa di query o dal valore dell'intestazione (cercati in questo ordine) a partire da una stringa del nome del parametro `x`. Si consiglia di utilizzare `$util.escapeJavaScript` per sanificare il parametro ed evitare un potenziale attacco di iniezione. Per il controllo completo della sanificazione dei parametri, utilizzare un'integrazione proxy senza un modello e gestire la sanificazione delle richieste nell'integrazione.

Variabile e funzione	Descrizione
`$input.path(x)`	Da una stringa di espressione JSONPath (x) restituisce una rappresentazione di oggetto JSON del risultato. In questo modo, puoi accedere agli elementi del payload e modificarli in modo nativo in Apache Velocity Template Language (VTL). Ad esempio, se l'espressione `$input.path('$.pets')` restituisce un oggetto in questo modo: `[ { "id": 1, "type": "dog", "price": 249.99 }, { "id": 2, "type": "cat", "price": 124.99 }, { "id": 3, "type": "fish", "price": 0.99 } ]` `$input.path('$.pets').size()` restituisce `"3"`. Per ulteriori informazioni su JSONPath, consulta la pagina relativa a JSONPath o JSONPath per Java.

Variabili di fase

È possibile utilizzare le seguenti variabili di fase come segnaposto per ARN e URL nelle integrazioni di metodo. Per ulteriori informazioni, consulta the section called “Utilizzo delle variabili di fase”.

Sintassi	Description
`$stageVariables.` *`variable_name`* , `$stageVariables['` *`variable_name`* `']` o `${stageVariables['` *`variable_name`* `']}`	*`variable_name`* rappresenta il nome di una variabile di fase.

Variabili util

È possibile utilizzare le seguenti variabili `$util` con distinzione tra maiuscole e minuscole per usare le funzioni di utilità per i modelli di mappatura. Se non diversamente specificato, il set di caratteri predefinito è UTF-8.

Funzione	Description
`$util.escapeJavaScript()`	Consente di sfuggire ai caratteri di una stringa utilizzando le regole delle JavaScript stringhe. **Note** Questa funzione trasforma qualsiasi virgoletta singola (`'`) in virgoletta preceduta da un carattere escape (`\'`). Tuttavia, le virgolette singole con escape non sono valide in JSON. Di conseguenza, quando l'output di questa funzione viene usato in una proprietà JSON, devi modificare di nuovo qualsiasi virgoletta singola con carattere escape (`\'`) in virgoletta singola normale (`'`). Questo viene mostrato nell'esempio seguente:

Funzione	Description
	<pre>"input" : "$util.escapeJavaScript(data).replaceAll("\\'",'''')"</pre>

`$util.parseJson()`	Da una stringa JSON restituisce una rappresentazione oggetto del risultato. Puoi usare il risultato di questa funzione per accedere agli elementi del payload e modificarli in modo nativo in Apache Velocity Template Language (VTL). Ad esempio, in presenza del payload seguente:
	<pre>{"errorMessage":"{\"key1\":\"var1\",\"key2\":{\"arr\":[1,2,3]}}"}</pre>E se usi il modello di mappatura seguente:
	<pre>#set ($errorMessageObj = $util.parseJson($input.path('$.errorMessage')))
	{
	   "errorMessageObjKey2ArrVal" : $errorMessageObj.key2.arr[0]
	}</pre>Otterrai l'output seguente:
	<pre>{
	   "errorMessageObjKey2ArrVal" : 1
	}</pre>

`$util.urlEncode()`	Converte una stringa nel formato "application/x-www-form-urlencoded».

Funzione	Description
`$util.urlDecode()`	Decodifica una stringa application/x "-www-form-urlencoded».
`$util.base64Encode()`	Codifica i dati in una stringa con codifica base64.
`$util.base64Decode()`	Decodifica i dati da una stringa con codifica base64.

Risposte gateway per REST APIs in API Gateway

Una risposta del gateway è identificata da un tipo di risposta definito da API Gateway. La risposta consiste in un codice di stato HTTP, un insieme di intestazioni aggiuntive specificate dalle mappature dei parametri e un payload generato da un modello di mappatura non VTL.

Nell'API REST di API Gateway, una risposta del gateway è rappresentata da GatewayResponse. In OpenAPI, un'`GatewayResponse`istanza è descritta dall'estensione x-amazon-apigateway-gateway-Response.gatewayResponse.

Per abilitare una risposta del gateway, dei impostare una risposta del gateway per un tipo di risposta supportato a livello di API. Ogni volta che API Gateway restituisce una risposta di questo tipo, vengono applicate le mappature delle intestazioni e i modelli di mappatura del payload definiti nel gateway per restituire i risultati mappati all'intermediario dell'API.

Nella sezione seguente viene illustrato come impostare le risposte del gateway tramite la console API Gateway e l'API REST API Gateway.

Configurazione delle risposte del gateway per la personalizzazione delle risposte agli errori

Se API Gateway non riesce a elaborare una richiesta in entrata, restituisce al client una risposta di errore senza inoltrare la richiesta al back-end di integrazione. Per impostazione predefinita, la risposta di errore contiene un breve messaggio di errore descrittivo. Ad esempio, se tenti di chiamare un'operazione da una risorsa API non definita, riceverai una risposta di errore con il messaggio `{ "message": "Missing Authentication Token" }`. Se sei nuovo su API Gateway, potrebbe essere difficile capire che cosa esattamente non sia andato a buon fine.

Per alcune delle risposte di errore, API Gateway consente agli sviluppatori API di personalizzare le risposte in formati differenti. Per l'esempio `Missing Authentication Token`, puoi aggiungere un suggerimento al payload originale della risposta con la possibile causa, come in questo esempio: `{"message":"Missing Authentication Token", "hint":"The HTTP method or resources may not be supported."}`.

Quando l'API fa da intermediario tra uno scambio esterno e il AWS Cloud, utilizzi modelli di mappatura VTL per la richiesta di integrazione o la risposta di integrazione per mappare il payload da un formato all'altro. Tuttavia, i modelli di mappatura VTL funzionano solo per le richieste valide con risposte andate a buon fine.

Per le richieste non valide, API Gateway ignora completamente l'integrazione e restituisce una risposta di errore. Devi utilizzare la personalizzazione per rendere le risposte di errore in un formato conforme allo scambio. Qui, la personalizzazione viene realizzata in un modello di mappatura non-VTL che supporta solo le sostituzioni semplici delle variabili.

Per generalizzare la risposta di errore generata da API Gateway e qualsiasi risposta generata da API Gateway, usiamo il termine risposte del gateway. In questo modo facciamo distinzione tra le risposte generate da API Gateway e le risposte di integrazione. Un modello di mappatura di una risposta del gateway può accedere ai valori della variabile `$context` e ai valori della proprietà `$stageVariables`, nonché ai parametri della richiesta di metodo, sotto forma di `method.request.`*`param-position`*`.`*`param-name`*.

Per ulteriori informazioni sulle variabili `$context`, consulta Variabili di contesto per le trasformazioni dei dati. Per ulteriori informazioni su `$stageVariables`, consulta Variabili di fase. Per ulteriori informazioni sui parametri di richiesta del metodo, consulta the section called “Variabili input”.

Argomenti

- Impostare una risposta del gateway per un'API REST utilizzando la console API Gateway
- Configurazione di una risposta del gateway mediante l'API REST API Gateway
- Configurazione della personalizzazione di una risposta del gateway in OpenAPI
- Tipi di risposta del gateway in Gateway API

Impostare una risposta del gateway per un'API REST utilizzando la console API Gateway

Il seguente esempio mostra come impostare una risposta del gateway per una REST API utilizzando la console Gateway API.

Per configurare una risposta del gateway mediante la console API Gateway

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere una REST API.
3. Nel riquadro di navigazione principale di Gateway API, scegli Risposte di Gateway.
4. Scegli un tipo di risposta, quindi scegli Modifica. In questa procedura guidata, utilizziamo Token di autenticazione mancante come esempio.
5. Ora puoi cambiare il valore dell'opzione Codice di stato generato da Gateway API per restituire un codice di stato diverso che soddisfi i requisiti dell'API. In questo esempio, la personalizzazione cambia il codice di stato da quello predefinito (`403`) al codice `404`, poiché questo messaggio di errore viene ricevuto quando un client chiama una risorsa non valida o non supportata che può essere considerata come non trovata.
6. Per tornare alle intestazioni personalizzate, seleziona Aggiungi intestazione della risposta in Intestazioni di risposta. A scopo illustrativo, aggiungiamo le seguenti intestazioni personalizzate:

   ```
   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
   ```

   Nelle precedenti mappature delle intestazioni, viene mappato un nome di dominio statico (`'a.b.c'`) all'intestazione `Allow-Control-Allow-Origin` per consentire a CORS di accedere all'API; l'intestazione della richiesta di input di `x-amzn-RequestId` viene mappata a `request-id` nella risposta; la variabile di percorso `petId` della richiesta in entrata viene mappata all'intestazione `request-path` nella risposta, mentre il parametro di query `q` della richiesta originale viene mappato all'intestazione `request-query` della risposta.
7. In Modelli di risposta, non modificare `application/json` in Tipo di contenuto e immetti il seguente modello di mappatura del corpo nell'editor Corpo modello:

   ```
   {
        "message":"$context.error.messageString",
        "type": "$context.error.responseType",
        "statusCode": "'404'",
        "stage": "$context.stage",
        "resourcePath": "$context.resourcePath",
        "stageVariables.a": "$stageVariables.a"
   }
   ```

Questo modello mostra come mappare le proprietà `$context` e `$stageVariables` alle proprietà del corpo di risposta del gateway.

8. Scegli Save changes (Salva modifiche).
9. Distribuzione dell'API in una fase nuova o esistente.

Esegui un test della risposta del gateway chiamando il seguente comando CURL, supponendo che l'URL di richiamo del metodo API corrispondente sia `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
```

Considerato che il parametro aggiuntivo della stringa di query q=1 non è compatibile con l'API, viene restituito un errore dalla risposta del gateway specificata. Dovresti visualizzare una risposta del gateway simile alla seguente:

```
> 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
}
```

Il precedente esempio presuppone che il back-end dell'API sia Pet Store e l'API abbia una variabile di fase, a, definita.

Configurazione di una risposta del gateway mediante l'API REST API Gateway

Prima di personalizzare una risposta del gateway tramite l'API REST API Gateway, devi aver già creato un'API e aver ottenuto il relativo identificatore. Per recuperare l'identificatore dell'API puoi seguire la relazione di collegamento restapi:gateway-responses ed esaminare il risultato.

Per configurare una risposta del gateway mediante l'API REST API Gateway

1. Per sovrascrivere un'intera GatewayResponseistanza, chiama il gatewayresponse:put action. Specificare un valore desiderato per responseType nel parametro di percorso URL e fornire nel payload della richiesta le mappature statusCode, responseParameters e responseTemplates.
2. Per aggiornare parte di un'istanza `GatewayResponse`, chiamare l'operazione gatewayresponse:update. Specificare un valore desiderato per `responseType` nel parametro di percorso URL e fornire nel payload della richiesta le singole proprietà `GatewayResponse` desiderate, ad esempio la mappatura `responseParameters` o `responseTemplates`.

Configurazione della personalizzazione di una risposta del gateway in OpenAPI

Puoi usare l'estensione `x-amazon-apigateway-gateway-responses` a livello di radice dell'API per personalizzare le risposte del gateway in OpenAPI. La seguente definizione di OpenAPI mostra un esempio per personalizzare il GatewayResponsetipo. `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}"
      }
    }
```

In questo esempio, la personalizzazione cambia il codice di stato dal predefinito (`403`) a `404`. Aggiunge anche alla risposta del gateway quattro parametri di intestazione e un modello di mappatura del corpo per il tipo di supporto `application/json`.

Tipi di risposta del gateway in Gateway API

API Gateway espone le seguenti risposte del gateway alla personalizzazione da parte degli sviluppatori di API.

Tipo di risposta del gateway	Codice di stato predefinito	Descrizione
`ACCESS_DENIED`	`403`	Risposta del gateway per un'autorizzazione non riuscita, ad esempio quando l'accesso viene negato da un'autorizzazione ad hoc o di Amazon Cognito. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito sul tipo `DEFAULT_4XX` .
`API_CONFIGURATION_ERROR`	`500`	La risposta del gateway per configurazioni API non valide, incluso l'invio dell'indirizzo dell'endpoint errato, la codifica base64 non riuscita sui dati binari quando il supporto binario è abilitato o la mappatura della risposta di integrazione non corrispon

Tipo di risposta del gateway	Codice di stato predefinito	Descrizione
		dente a nessun modello e nessuna configurazione di un modello predefinito. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito sul tipo `DEFAULT_5XX` .
`AUTHORIZER_CONFIGURATION_ERROR`	`500`	Risposta del gateway per la mancata connessione ad autorizzazioni ad hoc o di Amazon Cognito. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito sul tipo `DEFAULT_5XX` .
`AUTHORIZER_FAILURE`	`500`	Risposta del gateway quando autorizzazioni ad hoc o di Amazon Cognito non riescono ad autenticare l'intermediario. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito sul tipo `DEFAULT_5XX` .
`BAD_REQUEST_PARAMETERS`	`400`	Riposta del gateway quando il parametro di richiesta non può essere convalidato in base a un validatore abilitato delle richieste. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito sul tipo `DEFAULT_4XX` .

Tipo di risposta del gateway	Codice di stato predefinito	Descrizione
BAD_REQUEST_BODY	400	Riposta del gateway quando il corpo della richiesta non può essere convalidato in base a un validatore abilitato delle richieste. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito sul tipo DEFAULT_4XX .
DEFAULT_4XX	Null	La risposta predefinita del gateway per un tipo di risposta non specificato con codice di stato 4XX. Cambiando il codice di stato di questa risposta di fallback del gateway significa cambiare i codici di stato di tutte le altre risposte 4XX al nuovo valore. Reimpostando il codice di stato su null, i codici di stato di tutte le altre risposte 4XX ritornano ai loro valori originali. **Note** AWS WAF le risposte personalizzate hanno la precedenza sulle risposte gateway personalizzate.

Tipo di risposta del gateway	Codice di stato predefinito	Descrizione
DEFAULT_5XX	Null	La risposta predefinita del gateway per un tipo di risposta non specificato con codice di stato 5XX. Cambiando il codice di stato di questa risposta di fallback del gateway significa cambiare i codici di stato di tutte le altre risposte 5XX al nuovo valore. Reimpostando il codice di stato su null, i codici di stato di tutte le altre risposte 5XX ritornano ai loro valori originali.
EXPIRED_TOKEN	403	La risposta del gateway per un errore del token AWS di autenticazione è scaduta. Se il tipo di risposta non è specifica to, la risposta viene impostata in modo predefinito sul tipo DEFAULT_4XX .
INTEGRATION_FAILURE	504	Risposta del gateway per un errore di integrazione non riuscita. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito sul tipo DEFAULT_5XX .

Tipo di risposta del gateway	Codice di stato predefinito	Descrizione
INTEGRATION_TIMEOUT	504	Risposta del gateway per un errore di integrazione scaduta. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito sul tipo DEFAULT_5XX .
INVALID_API_KEY	403	Risposta del gateway per una chiave API non valida inviata per un metodo che la richiedev a. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito sul tipo DEFAULT_4XX .
INVALID_SIGNATURE	403	La risposta del gateway per un errore di AWS firma non valido. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito sul tipo DEFAULT_4XX .
MISSING_AUTHENTICATION_TOKEN	403	Risposta del gateway per un errore di token autenticazione mancante, inclusi i casi in cui il cliente tenta di chiamare un metodo o una risorsa API non supportati. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito sul tipo DEFAULT_4XX .

Tipo di risposta del gateway	Codice di stato predefinito	Descrizione
QUOTA_EXCEEDED	429	Risposta del gateway per un errore di superamento della quota di utilizzo del piano. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito sul tipo DEFAULT_4XX .
REQUEST_TOO_LARGE	413	Risposta del gateway per un errore di dimensioni eccessive della richiesta. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito su HTTP content length exceeded 10485760 bytes.
RESOURCE_NOT_FOUND	404	Risposta del gateway quando API Gateway non riesce a trovare l risorsa specifica ta dopo che una richiesta API passa autenticazione e autorizzazione, ad eccezione dell'autenticazione e dell'auto rizzazione della chiave API. Se il tipo di risposta non è specificato, la risposta viene impostata in modo predefinito sul tipo DEFAULT_4XX .

Tipo di risposta del gateway	Codice di stato predefinito	Descrizione
THROTTLED	429	Risposta del gateway quando vengono superati i limiti di throttling a livello di account, fase, metodo o piano. Se il tipo di risposta non è specifica to, la risposta viene impostata in modo predefinito sul tipo DEFAULT_4XX .
UNAUTHORIZED	401	Risposta del gateway quando l'autorizzazione ad hoc o di Amazon Cognito non riesce ad autenticare l'intermediario.
UNSUPPORTED_MEDIA_ TYPE	415	Risposta del gateway quando il payload presenta un tipo di supporto non supportato, se è stato abilitato un comportam ento passthrough severo. Se il tipo di risposta non è specifica to, la risposta viene impostata in modo predefinito sul tipo DEFAULT_4XX .

Tipo di risposta del gateway	Codice di stato predefinito	Descrizione
`WAF_FILTERED`	`403`	La risposta del gateway quando una richiesta è bloccato da AWS WAF. Se il tipo di risposta non è specifica to, la risposta viene impostata in modo predefinito sul tipo `DEFAULT_4XX` . **Note** AWS WAF le risposte personalizzate hanno la precedenza sulle risposte gateway personalizzate.

CORS per REST API in Gateway API

Cross-origin la condivisione delle risorse (CORS) è una funzionalità di sicurezza del browser che limita le richieste HTTP tra origini che vengono avviate da script in esecuzione nel browser. Per ulteriori informazioni, consulta Che cos'è CORS?

Determinazione della necessità di abilitare il supporto di CORS

Una richiesta HTTP multiorigine è una richiesta effettuata a:

- Un dominio diverso (ad esempio da `example.com` a `amazondomains.com`)
- Un sottodominio diverso (ad esempio da `example.com` a `petstore.example.com`)
- Una porta diversa (ad esempio da `example.com` a `example.com:10777`)
- Un protocollo diverso (ad esempio da `https://example.com` a `http://example.com`)

Se non riesci ad accedere all'API e ricevi un messaggio di errore che contiene `Cross-Origin Request Blocked`, potresti dover abilitare CORS.

Cross-origin Le richieste HTTP possono essere suddivise in due tipi: richieste semplici e richieste non semplici.

Abilitazione di CORS per una richiesta semplice

Una richiesta HTTP è semplice se si verificano tutte le condizioni seguenti:

- Viene inviata a una risorsa API che permette solo richieste `GET`, `HEAD` e `POST`.
- Se si tratta di una richiesta di metodo `POST`, deve includere un'intestazione `Origin`.
- Il tipo di contenuto del payload della richiesta è `text/plain`, `multipart/form-data` o `application/x-www-form-urlencoded`.
- La richiesta non contiene intestazioni personalizzate.
- Eventuali requisiti aggiuntivi elencati nella documentazione di Mozilla CORS per le richieste semplici.

Per richieste di metodo `POST` multiorigine semplici, la risposta della risorsa deve includere l'intestazione `Access-Control-Allow-Origin: '*'` o `Access-Control-Allow-Origin:'origin'`.

Tutte le altre richieste HTTP multiorigine sono richieste non semplici.

Abilitazione di CORS per una richiesta non semplice

Se le risorse dell'API ricevono richieste non semplici, è necessario abilitare il supporto CORS aggiuntivo a seconda del tipo di integrazione.

Abilitazione di CORS per integrazioni non proxy

Per questo tipo di integrazioni, il protocollo CORS richiede al browser di inviare una richiesta preliminare al server e di attendere l'approvazione (o una richiesta di credenziali) del server prima di inviare la richiesta effettiva. È necessario configurare l'API per inviare una risposta appropriata alla richiesta di verifica preliminare.

Creazione di una risposta di verifica preliminare

1. Creare un metodo `OPTIONS` per l'integrazione fittizia.
2. Aggiungere le seguenti intestazioni di risposta alla risposta del metodo 200:
    - `Access-Control-Allow-Headers`

- `Access-Control-Allow-Methods`
- `Access-Control-Allow-Origin`

3. Impostare il comportamento passthrough di integrazione su `NEVER`. In questo caso, la richiesta di metodo di un tipo di contenuto non mappato sarà rifiutata con la risposta Tipo di supporto non compatibile HTTP 415. Per ulteriori informazioni, consulta Comportamento della richiesta di metodo per i payload senza modelli di mappatura per REST API in Gateway API.
4. Immettere i valori per le intestazioni delle risposte. Per consentire tutte le origini, tutti i metodi e le intestazioni comuni, usare i seguenti valori di intestazione:

   - `Access-Control-Allow-Headers: 'Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token'`
   - `Access-Control-Allow-Methods: 'DELETE,GET,HEAD,OPTIONS,PUT,POST,PATCH'`
   - `Access-Control-Allow-Origin: '*'`

Dopo aver creato la richiesta di preflight, è necessario restituire l'`Access-Control-Allow-Origin:`*`'origin'`*intestazione `Access-Control-Allow-Origin: '*'` o per tutti i CORS-enabled metodi per almeno tutte le 200 risposte.

Abilitazione di CORS per integrazioni non proxy utilizzando il Console di gestione AWS

È possibile utilizzare il per abilitare CORS Console di gestione AWS . Gateway Amazon API crea un metodo `OPTIONS` e aggiunge l'intestazione `Access-Control-Allow-Origin` alle risposte di integrazione del metodo esistente. Questo non sempre funziona e talvolta è necessario modificare manualmente la risposta di integrazione per restituire l'`Access-Control-Allow-Origin`intestazione per tutti i CORS-enabled metodi per almeno tutte le 200 risposte.

Se i tipi di supporti binari sono impostati su `*/*` per l'API, quando Gateway API crea un metodo `OPTIONS`, modificare `contentHandling` in `CONVERT_TO_TEXT`.

Il comando update-integration seguente modifica `contentHandling` in `CONVERT_TO_TEXT` per una richiesta di integrazione:

```
aws apigateway update-integration \
  --rest-api-id abc123 \
  --resource-id aaa111 \
  --http-method OPTIONS \
  --patch-operations op='replace',path='/contentHandling',value='CONVERT_TO_TEXT'
```

Il comando update-integration-response seguente modifica `contentHandling` in `CONVERT_TO_TEXT` per una risposta di integrazione:

```
aws apigateway update-integration-response \
  --rest-api-id abc123 \
  --resource-id aaa111 \
  --http-method OPTIONS \
  --status-code 200 \
  --patch-operations op='replace',path='/contentHandling',value='CONVERT_TO_TEXT'
```

Abilitazione del supporto di CORS per le integrazioni proxy

Per un'integrazione proxy Lambda o un'integrazione con proxy HTTP, il back-end è responsabile della restituzione delle intestazioni `Access-Control-Allow-Origin`, `Access-Control-Allow-Methods` e `Access-Control-Allow-Headers` , perché un'integrazione proxy non restituisce una risposta di integrazione.

Le seguenti funzioni Lambda di esempio restituiscono le intestazioni CORS richieste:

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!')
    }
```

Argomenti

- Abilitazione CORS su una risorsa tramite la console API Gateway
- Abilitazione di CORS per una risorsa tramite l'API di importazione di API Gateway
- Test di CORS per un'API di Gateway API

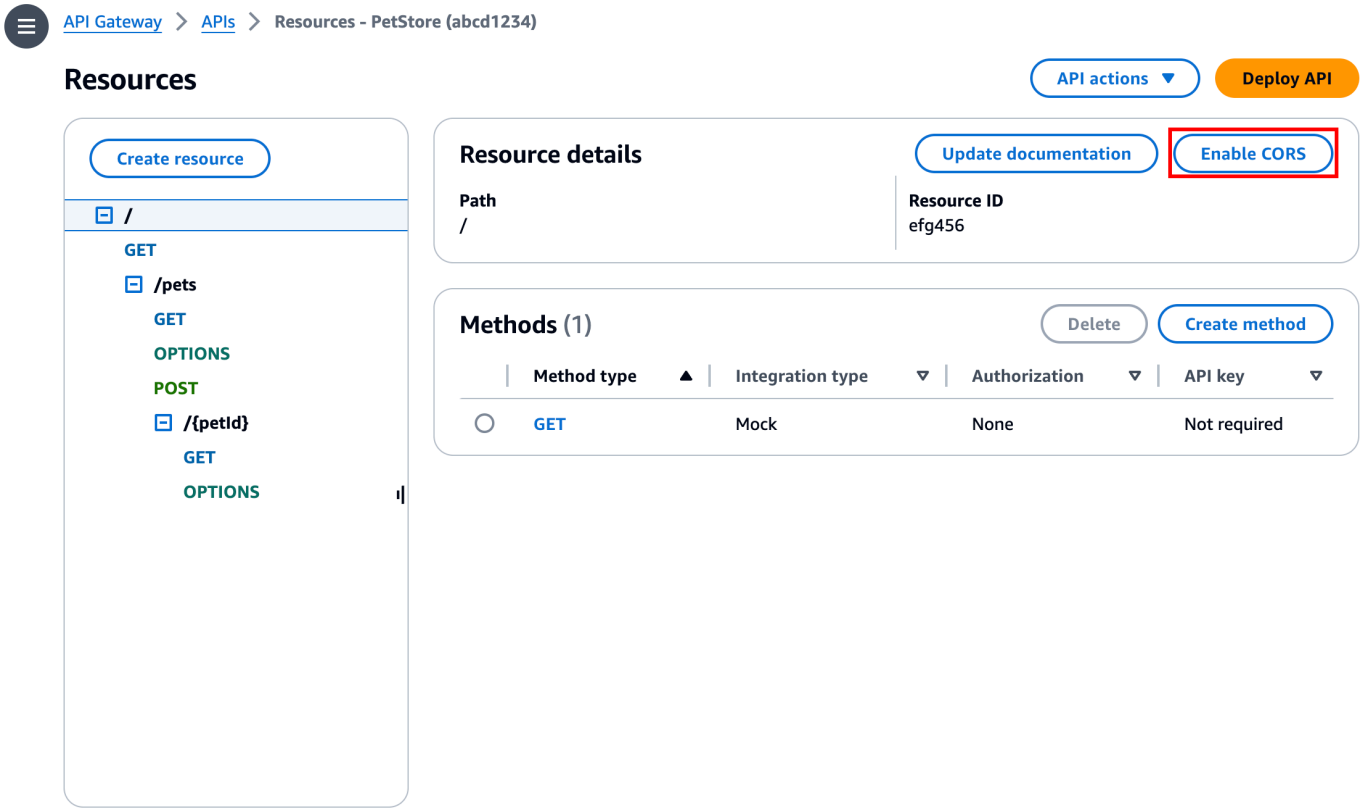
Abilitazione CORS su una risorsa tramite la console API Gateway

È possibile utilizzare la console API Gateway per abilitare il supporto di CORS per uno o per tutti i metodi in una risorsa API REST creata. Dopo aver abilitato il supporto CORS, impostare il comportamento passthrough di integrazione su `NEVER`. In questo caso, la richiesta di metodo di un tipo di contenuto non mappato sarà rifiutata con la risposta Tipo di supporto non compatibile HTTP 415. Per ulteriori informazioni, consulta Comportamento della richiesta di metodo per i payload senza modelli di mappatura per REST API in Gateway API

> **Important**
>
> Le risorse possono contenere risorse figlio. L'abilitazione del supporto di CORS per una risorsa e per i relativi metodi non comporta l'abilitazione ricorsiva per le risorse figlio e i relativi metodi.

Abilitazione del supporto di CORS in una risorsa REST API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere un'API.
3. Scegliere una risorsa in Resources (Risorse).
4. Nella sezione Dettagli delle risorse, scegli Abilita CORS.



5. Nella casella Abilita CORS esegui quanto segue:

   a. (Facoltativo) Se hai creato una risposta del gateway del cliente e desideri abilitare il supporto di CORS per una risposta, seleziona una risposta gateway.

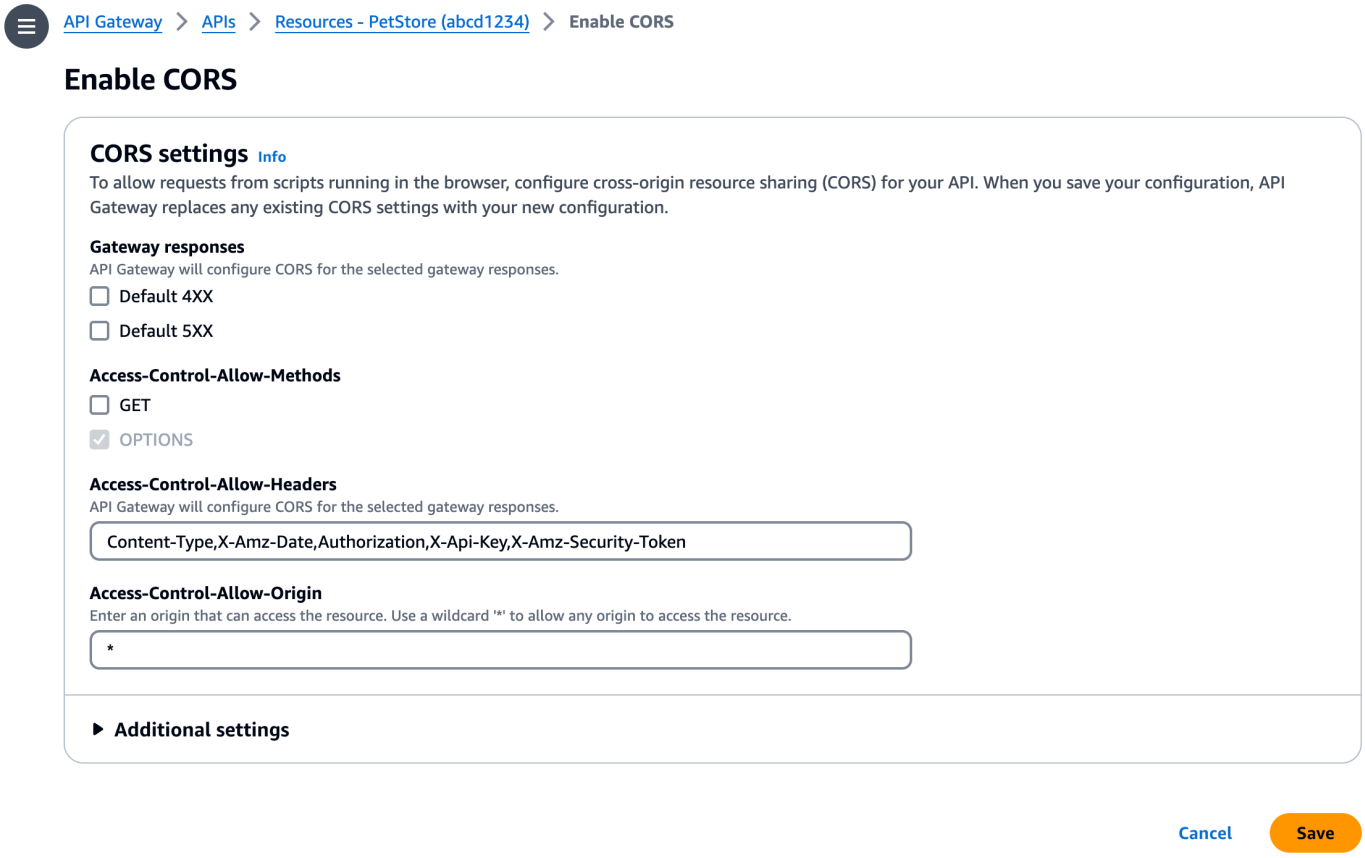
   b. Seleziona ciascun metodo per abilitare il supporto di CORS. Per il metodo `OPTION` è necessario abilitare il supporto di CORS.

      Se abiliti il supporto di CORS per un metodo `ANY`, CORS viene abilitato per tutti i metodi.

   c. Nel campo Access-Control-Allow-Headersdi immissione, inserisci una stringa statica di un elenco di intestazioni separate da virgole che il client deve inviare nella richiesta effettiva della risorsa. Usa l'elenco di intestazioni `'Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token'` fornito dalla console o specifica intestazioni personalizzate.

   d. Utilizza il valore fornito dalla console di `'*'` come valore di Access-Control-Allow-Originintestazione per consentire le richieste di accesso da tutte le origini o specifica un'origine a cui consentire l'accesso alla risorsa.

   e. Scegli Save (Salva).



> **Important**
>
> Quando si applicano le istruzioni sopra riportate al metodo ANY in un'integrazione proxy, non verrà impostata nessuna intestazione CORS applicabile. Il back-end deve invece restituire le intestazioni CORS applicabili, ad esempio `Access-Control-Allow-Origin`.

Dopo l'abilitazione di CORS per il metodo `GET`, alla risorsa viene aggiunto un metodo `OPTIONS`, se non è già presente. La risposta `200` del metodo `OPTIONS` viene configurata automaticamente per restituire le tre intestazioni `Access-Control-Allow-*` per soddisfare gli handshake preliminari. Inoltre, il metodo effettivo (`GET`) è configurato per impostazione predefinita per restituire l'intestazione `Access-Control-Allow-Origin` anche nella risposta 200. Gli altri tipi di risposta devono essere configurati manualmente in modo da restituire l'intestazione `Access-Control-Allow-Origin'` con '*' oppure origini specifiche, se non si vuole che venga restituito l'errore `Cross-origin access`.

Dopo aver abilitato il supporto di CORS nella risorsa, è necessario distribuire o ridistribuire l'API per rendere effettive le nuove impostazioni. Per ulteriori informazioni, consulta the section called “Crea distribuzione”.

> **Note**
>
> Se non è possibile abilitare il supporto CORS sulla risorsa con questa procedura, si consiglia di confrontare la configurazione CORS con la risorsa `/pets` dell'API di esempio. Per informazioni su come creare l'API di esempio, consulta the section called “Tutorial: creazione di un'API REST mediante l'importazione di un esempio”.

Abilitazione di CORS per una risorsa tramite l'API di importazione di API Gateway

Se utilizzi l'API di importazione di API Gateway, puoi configurare il supporto di CORS utilizzando un file OpenAPI. È prima necessario definire un metodo `OPTIONS` nella risorsa che restituisce le intestazioni necessarie.

> **Note**
>
> I browser Web Access-Control-Allow-Headers si aspettano che le Access-Control-Allow-Origin intestazioni siano configurate in ogni metodo API che accetta richieste CORS. Inoltre, alcuni browser inviano prima una richiesta HTTP a un metodo `OPTIONS` nella stessa risorsa e quindi aspettano di ricevere le stesse intestazioni.

Esempio di metodo Options

L'esempio seguente crea un metodo `OPTIONS` per un'integrazione fittizia.

OpenAPI 3.0

```
/users:
  options:
    summary: CORS support
    description: |
      Enable CORS by returning correct headers
    tags:
    - CORS
    responses:
```

```
      200:
        description: Default response for CORS method
        headers:
          Access-Control-Allow-Origin:
            schema:
              type: "string"
          Access-Control-Allow-Methods:
            schema:
              type: "string"
          Access-Control-Allow-Headers:
            schema:
              type: "string"
        content: {}
    x-amazon-apigateway-integration:
      type: mock
      requestTemplates:
        application/json: "{\"statusCode\": 200}"
      passthroughBehavior: "never"
      responses:
        default:
          statusCode: "200"
          responseParameters:
            method.response.header.Access-Control-Allow-Headers: "'Content-Type,X-
Amz-Date,Authorization,X-Api-Key'"
            method.response.header.Access-Control-Allow-Methods: "'*'"
            method.response.header.Access-Control-Allow-Origin: "'*'"
```

OpenAPI 2.0

```
/users:
   options:
      summary: CORS support
      description: |
        Enable CORS by returning correct headers
      consumes:
        - "application/json"
      produces:
        - "application/json"
      tags:
        - CORS
      x-amazon-apigateway-integration:
        type: mock
```

```
        requestTemplates: "{\"statusCode\": 200}"
        passthroughBehavior: "never"
        responses:
          "default":
            statusCode: "200"
            responseParameters:
              method.response.header.Access-Control-Allow-Headers : "'Content-
Type,X-Amz-Date,Authorization,X-Api-Key'"
              method.response.header.Access-Control-Allow-Methods : "'*'"
              method.response.header.Access-Control-Allow-Origin : "'*'"
      responses:
        200:
          description: Default response for CORS method
          headers:
            Access-Control-Allow-Headers:
              type: "string"
            Access-Control-Allow-Methods:
              type: "string"
            Access-Control-Allow-Origin:
              type: "string"
```

Una volta configurato il metodo `OPTIONS` per la risorsa, puoi aggiungere le intestazioni richieste agli altri metodi nella stessa risorsa che deve accettare le richieste CORS.

1. Dichiara le intestazioni Access-Control-Allow-Originand ai tipi di risposta.

   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. Nel tag `x-amazon-apigateway-integration` tag configura la mappatura per quelle intestazioni nei valori statici:

OpenAPI 3.0

```
    responses:
        default:
          statusCode: "200"
          responseParameters:
            method.response.header.Access-Control-Allow-Headers: "'Content-
Type,X-Amz-Date,Authorization,X-Api-Key'"
            method.response.header.Access-Control-Allow-Methods: "'*'"
            method.response.header.Access-Control-Allow-Origin: "'*'"
          responseTemplates:
            application/json: |
              {}
```

OpenAPI 2.0

```
    responses:
          "default":
            statusCode: "200"
            responseParameters:
              method.response.header.Access-Control-Allow-Headers : "'Content-
Type,X-Amz-Date,Authorization,X-Api-Key'"
              method.response.header.Access-Control-Allow-Methods : "'*'"
              method.response.header.Access-Control-Allow-Origin : "'*'"
```

API di esempio

L'esempio seguente crea un'API completa con un metodo `OPTIONS` e un metodo `GET` con un'integrazione `HTTP`.

OpenAPI 3.0

```
openapi: "3.0.1"
info:
  title: "cors-api"
  description: "cors-api"
  version: "2024-01-16T18:36:01Z"
servers:
- url: "/{basePath}"
  variables:
    basePath:
      default: "/test"
paths:
  /:
    get:
      operationId: "GetPet"
      responses:
        "200":
          description: "200 response"
          headers:
            Access-Control-Allow-Origin:
              schema:
                type: "string"
          content: {}
      x-amazon-apigateway-integration:
        httpMethod: "GET"
        uri: "http://petstore.execute-api.us-east-1.amazonaws.com/petstore/pets"
        responses:
          default:
            statusCode: "200"
            responseParameters:
              method.response.header.Access-Control-Allow-Origin: "'*'"
        passthroughBehavior: "never"
        type: "http"
    options:
      responses:
        "200":
          description: "200 response"
          headers:
```

```
            Access-Control-Allow-Origin:
              schema:
                type: "string"
            Access-Control-Allow-Methods:
              schema:
                type: "string"
            Access-Control-Allow-Headers:
              schema:
                type: "string"
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/Empty"
      x-amazon-apigateway-integration:
        responses:
          default:
            statusCode: "200"
            responseParameters:
              method.response.header.Access-Control-Allow-Methods: "'GET,OPTIONS'"
              method.response.header.Access-Control-Allow-Headers: "'Content-Type,X-
Amz-Date,Authorization,X-Api-Key'"
              method.response.header.Access-Control-Allow-Origin: "'*'"
        requestTemplates:
          application/json: "{\"statusCode\": 200}"
        passthroughBehavior: "never"
        type: "mock"
components:
  schemas:
    Empty:
      type: "object"
```

OpenAPI 2.0

```
swagger: "2.0"
info:
  description: "cors-api"
  version: "2024-01-16T18:36:01Z"
  title: "cors-api"
basePath: "/test"
schemes:
- "https"
paths:
  /:
```

```
    get:
      operationId: "GetPet"
      produces:
      - "application/json"
      responses:
        "200":
          description: "200 response"
          headers:
            Access-Control-Allow-Origin:
              type: "string"
      x-amazon-apigateway-integration:
        httpMethod: "GET"
        uri: "http://petstore.execute-api.us-east-1.amazonaws.com/petstore/pets"
        responses:
          default:
            statusCode: "200"
            responseParameters:
              method.response.header.Access-Control-Allow-Origin: "'*'"
        passthroughBehavior: "never"
        type: "http"
    options:
      consumes:
      - "application/json"
      produces:
      - "application/json"
      responses:
        "200":
          description: "200 response"
          schema:
            $ref: "#/definitions/Empty"
          headers:
            Access-Control-Allow-Origin:
              type: "string"
            Access-Control-Allow-Methods:
              type: "string"
            Access-Control-Allow-Headers:
              type: "string"
      x-amazon-apigateway-integration:
        responses:
          default:
            statusCode: "200"
            responseParameters:
              method.response.header.Access-Control-Allow-Methods: "'GET,OPTIONS'"
```

```
              method.response.header.Access-Control-Allow-Headers: "'Content-Type,X-
Amz-Date,Authorization,X-Api-Key'"
              method.response.header.Access-Control-Allow-Origin: "'*'"
        requestTemplates:
          application/json: "{\"statusCode\": 200}"
        passthroughBehavior: "never"
        type: "mock"
definitions:
  Empty:
    type: "object"
```

Test di CORS per un'API di Gateway API

È possibile testare la configurazione CORS dell'API richiamando l'API e controllando le intestazioni CORS nella risposta. Il comando `curl` seguente invia una richiesta OPTIONS a un'API distribuita.

```
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
```

Le intestazioni `Access-Control-Allow-Origin`, `Access-Control-Allow-Headers` e `Access-Control-Allow-Methods` nella risposta mostrano che l'API supporta CORS. Per ulteriori informazioni, consulta CORS per REST API in Gateway API.

Tipi di supporti binari per REST API in Gateway API

In API Gateway, la richiesta e la risposta delle API possono avere un payload binario o di testo. Il payload di testo è una stringa JSON in codifica `UTF-8`, mentre un payload binario è rappresentato da tutto ciò che non è un payload di testo. Il payload binario può essere, ad esempio, un file JPEG, GZip

o XML. La configurazione API necessaria per supportare i supporti binari dipende dal fatto che l'API utilizzi integrazioni proxy o non proxy.

Se utilizzi un'integrazione proxy con lo streaming di risposta del payload, non è necessario configurare i tipi di media binari. Per ulteriori informazioni, consulta the section called “Trasmetti in streaming la risposta di integrazione per le tue integrazioni proxy”.

AWS Lambda integrazioni proxy

Per gestire i payload binari per le integrazioni AWS Lambda proxy, è necessario codificare in base 64 la risposta della funzione. È inoltre necessario configurare il file binario per l'API. MediaTypes La configurazione `binaryMediaTypes` dell'API è un elenco di tipi di contenuti che l'API considera come dati binari. I tipi di supporti binari di esempio includono `image/png` o `application/octet-stream`. È possibile utilizzare il carattere jolly (`*`) per includere più tipi di file multimediali.

API Gateway utilizza la prima intestazione `Accept` dai client per determinare se una risposta deve restituire supporti binari. Per restituire un supporto binario quando non è possibile controllare l’ordine dei valori delle intestazioni `Accept`, ad esempio le richieste da un browser, impostare i tipi di supporti binari dell’API su `*/*`.

Per il codice di esempio, consulta the section called “Restituzione di supporti binari da un'integrazione proxy Lambda in Gateway API”.

Se utilizzi un'integrazione proxy Lambda con lo streaming di risposta del payload, non è necessario configurare i tipi di media binari. Per ulteriori informazioni, consulta the section called “Configura l’integrazione proxy Lambda con lo streaming di risposta al payload”.

Non-proxy integrazioni

Per gestire i payload binari per le integrazioni non proxy, aggiungete i tipi di file multimediali all'MediaTypeselenco binario della risorsa. `RestApi` La configurazione `binaryMediaTypes` dell'API è un elenco di tipi di contenuti che l'API considera come dati binari. In alternativa, puoi impostare le proprietà ContentHandling sull'integrazione e sulle risorse. IntegrationResponse Il valore `contentHandling` può essere `CONVERT_TO_BINARY`, `CONVERT_TO_TEXT` o non definito.

> **Note**
>
> Per le integrazioni `MOCK` o private, l’impostazione delle proprietà `contentHandling` non è supportata nella Console di gestione AWS. È necessario utilizzare AWS CLI CloudFormation, o un SDK per impostare le proprietà. `contentHandling`

A seconda del valore `contentHandling` e se l'intestazione `Content-Type` della risposta o l'intestazione `Accept` della richiesta in entrata corrisponde a una voce dell'elenco `binaryMediaTypes`, API Gateway può codificare i byte binari grezzi come stringa con codifica base64, decodificare una stringa con codifica base64 di nuovo in byte grezzi oppure passare il corpo senza alcuna modifica.

Devi configurare l'API come segue per supportare i payload binari per l'API in API Gateway:

- Aggiungete i tipi di file multimediali binari desiderati all'`binaryMediaTypes`elenco della RestApirisorsa. Se questa proprietà e la `contentHandling` proprietà non sono definite, i payload vengono gestiti come stringhe JSON UTF-8 codificate.
- Indirizzare la proprietà `contentHandling` della risorsa Integration.
  - Per convertire il payload della richiesta da una stringa con codifica base64 nel relativo BLOB binario, impostare la proprietà su `CONVERT_TO_BINARY`.
  - Per convertire il payload della richiesta da un BLOB binario a una stringa con codifica base64, impostare la proprietà su `CONVERT_TO_TEXT`.
  - Per passare il payload senza modifiche, lasciare la proprietà indefinita. Per passare un payload binario senza modifiche, è inoltre necessario assicurarsi che `Content-Type` corrisponda a una delle voci `binaryMediaTypes` e che per l'API siano abilitati i comportamenti passthrough.
- Imposta la proprietà della risorsa. `contentHandling` IntegrationResponse La proprietà `contentHandling`, l'intestazione `Accept` nelle richieste del client e i tipi `binaryMediaTypes` combinati dell'API determinano il modo in cui API Gateway gestisce le conversioni dei tipi di contenuto. Per informazioni dettagliate, vedi the section called “Conversioni dei tipi di contenuto in API Gateway”.

> **Important**
>
> Quando una richiesta contiene più tipi di supporto nell'intestazione `Accept`, API Gateway mantiene la conformità solo con il primo tipo di supporto `Accept`. Se non è possibile controllare l'ordine dei tipi di supporto `Accept` e il tipo di supporto del contenuto binario non è il primo dell'elenco, aggiungere il primo tipo di supporto `Accept` nell'elenco `binaryMediaTypes` dell'API. API Gateway gestisce tutti i tipi di contenuto in questo elenco come binari.
>
> Ad esempio, per inviare un file JPEG utilizzando un elemento `<img>` in un browser, il browser potrebbe inviare `Accept:image/webp,image/*,*/*;q=0.8` in una richiesta.

Aggiungendo `image/webp` all'elenco `binaryMediaTypes`, l'endpoint riceve il file JPEG come binario.

Per informazioni dettagliate su come API Gateway gestisce i payload di testo e binari, consulta Conversioni dei tipi di contenuto in API Gateway.

Conversioni dei tipi di contenuto in API Gateway

La combinazione di `binaryMediaTypes` dell'API, le intestazioni nelle richieste del client e la proprietà `contentHandling` di integrazione determinano il modo in cui API Gateway codifica i payload.

La tabella seguente mostra come API Gateway converte il payload della richiesta per configurazioni specifiche dell'`Content-Type`intestazione di una richiesta, l'`binaryMediaTypes`elenco di una RestApirisorsa e il valore della `contentHandling` proprietà della risorsa di integrazione.

Payload della richiesta di metodo	Intestazione **`Content-Type`** della richiesta	**`binaryMediaTypes`**	**`contentHandling`**	Payload della richiesta di integrazione
Dati di testo	Qualsiasi tipo di dato	Undefined	Undefined	UTF8-encoded stringa
Dati di testo	Qualsiasi tipo di dato	Undefined	`CONVERT_TO_BINARY`	Base64-decoded blob binario
Dati di testo	Qualsiasi tipo di dato	Undefined	`CONVERT_TO_TEXT`	UTF8-encoded corda
Dati di testo	Un tipo di dati di testo	Imposta con tipi di supporto corrispondenti	Undefined	Dati di testo
Dati di testo	Un tipo di dati di testo	Imposta con tipi di supporto corrispondenti	`CONVERT_TO_BINARY`	Base64-decoded blob binario

Payload della richiesta di metodo	Intestazione **`Content-Type`** della richiesta	**`binaryMediaTypes`**	**`contentHandling`**	Payload della richiesta di integrazione
Dati di testo	Un tipo di dati di testo	Imposta con tipi di supporto corrispondenti	`CONVERT_TO_TEXT`	Dati di testo
Dati binari	Un tipo di dati binari	Imposta con tipi di supporto corrispondenti	Undefined	Dati binari
Dati binari	Un tipo di dati binari	Imposta con tipi di supporto corrispondenti	`CONVERT_TO_BINARY`	Dati binari
Dati binari	Un tipo di dati binari	Imposta con tipi di supporto corrispondenti	`CONVERT_TO_TEXT`	Base64-encoded corda

La tabella seguente mostra come API Gateway converte il payload di risposta per configurazioni specifiche dell'`Accept`intestazione di una richiesta, l'`binaryMediaTypes`elenco di una RestApirisorsa e il valore della `contentHandling` proprietà della risorsa. IntegrationResponse

> **Important**
>
> Quando una richiesta contiene più tipi di supporto nell'intestazione `Accept`, API Gateway mantiene la conformità solo con il primo tipo di supporto `Accept`. Se non è possibile controllare l'ordine dei tipi di supporto `Accept` e il tipo di supporto del contenuto binario non è il primo dell'elenco, aggiungere il primo tipo di supporto `Accept` nell'elenco `binaryMediaTypes` dell'API. API Gateway gestisce tutti i tipi di contenuto in questo elenco come binari.
>
> Ad esempio, per inviare un file JPEG utilizzando un elemento `<img>` in un browser, il browser potrebbe inviare `Accept:image/webp,image/*,*/*;q=0.8` in una richiesta.

Aggiungendo `image/webp` all'elenco `binaryMediaTypes`, l'endpoint riceve il file JPEG come binario.

Payload della risposta di integrazione	Intestazione **`Accept`** della richiesta	**`binaryMediaTypes`**	**`contentHandling`**	Payload della risposta del metodo
Dati di testo o binari	Un tipo di testo	Undefined	Undefined	UTF8-encoded stringa
Dati di testo o binari	Un tipo di testo	Undefined	`CONVERT_TO_BINARY`	Base64-decoded macchia
Dati di testo o binari	Un tipo di testo	Undefined	`CONVERT_TO_TEXT`	UTF8-encoded corda
Dati di testo	Un tipo di testo	Imposta con tipi di supporto corrispondenti	Undefined	Dati di testo
Dati di testo	Un tipo di testo	Imposta con tipi di supporto corrispondenti	`CONVERT_TO_BINARY`	Base64-decoded macchia
Dati di testo	Un tipo di testo	Imposta con tipi di supporto corrispondenti	`CONVERT_TO_TEXT`	UTF8-encoded corda
Dati di testo	Un tipo binario	Imposta con tipi di supporto corrispondenti	Undefined	Base64-decoded macchia
Dati di testo	Un tipo binario	Imposta con tipi di supporto corrispondenti	`CONVERT_TO_BINARY`	Base64-decoded macchia

Payload della risposta di integrazione	Intestazione **Accept** della richiesta	**binaryMediaTypes**	**contentHandling**	Payload della risposta del metodo
Dati di testo	Un tipo binario	Imposta con tipi di supporto corrispondenti	`CONVERT_TO_TEXT`	UTF8-encoded corda
Dati binari	Un tipo di testo	Imposta con tipi di supporto corrispondenti	Undefined	Base64-encoded corda
Dati binari	Un tipo di testo	Imposta con tipi di supporto corrispondenti	`CONVERT_TO_BINARY`	Dati binari
Dati binari	Un tipo di testo	Imposta con tipi di supporto corrispondenti	`CONVERT_TO_TEXT`	Base64-encoded corda
Dati binari	Un tipo binario	Imposta con tipi di supporto corrispondenti	Undefined	Dati binari
Dati binari	Un tipo binario	Imposta con tipi di supporto corrispondenti	`CONVERT_TO_BINARY`	Dati binari
Dati binari	Un tipo binario	Imposta con tipi di supporto corrispondenti	`CONVERT_TO_TEXT`	Base64-encoded corda

Quando converti un payload di testo in un BLOB binario, API Gateway assume che i dati di testo siano una stringa con codifica base64 e genera un output dei dati binari come BLOB con codifica base64. Se la conversione non va a buon fine, viene restituita una risposta `500` che indica un errore di configurazione dell'API. Non fornisci un modello di mappatura per una conversione di questo tipo, sebbene tu debba abilitare i comportamenti passthrough sull'API.

Quando converti un payload binario in una stringa di testo, API Gateway applica sempre una codifica base64 ai dati binari. Per tale payload puoi definire un modello di mappatura, ma puoi accedere alla stringa con codifica base64 nel modello di mappatura solo attraverso `$input.body`, come viene mostrato nel seguente estratto di un modello di mappatura di esempio.

```
{
    "data": "$input.body"
}
```

Per fare in modo che il payload binario venga passato senza alcuna modifica, devi abilitare i comportamenti passthrough sull'API.

Abilitazione del supporto binario tramite la console API Gateway

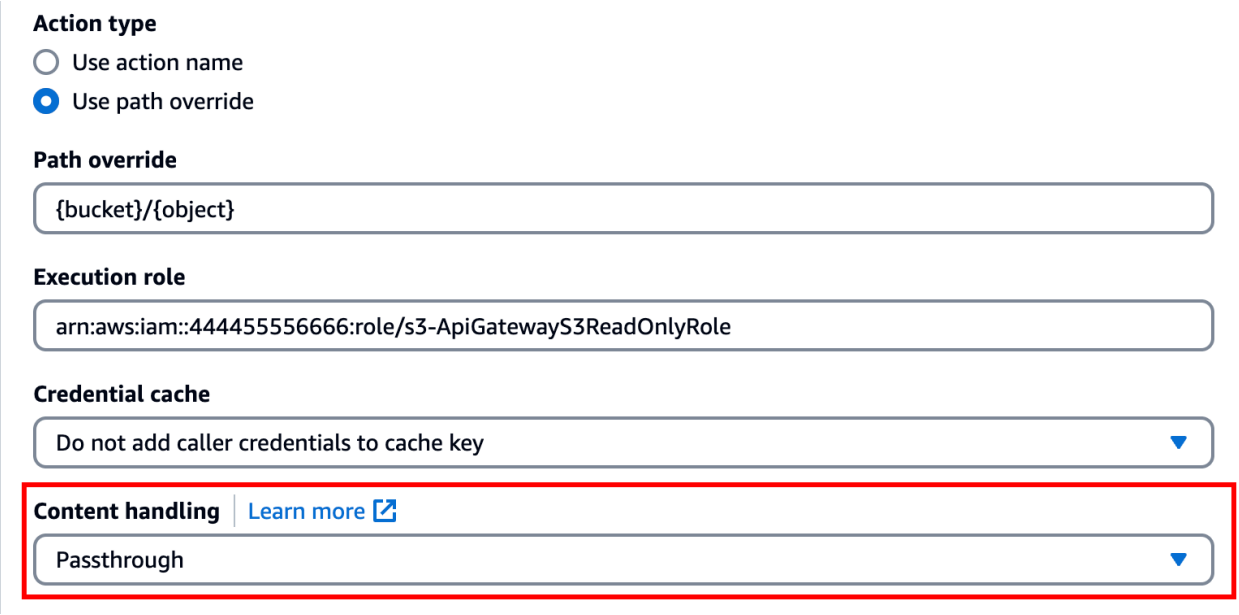
In questa sezione viene descritto come abilitare il supporto binario tramite la console API Gateway. Come esempio, utilizziamo un'API integrata con Amazon S3. Ci focalizziamo sulle attività per impostare i tipi di supporti ammessi e per specificare come dovrebbe essere gestito il payload. Per informazioni dettagliate su come creare un'API integrata con Amazon S3, consulta Tutorial: creazione di una REST API come un proxy Amazon S3.

Per abilitare il supporto binario mediante la console API Gateway

1. Imposta i tipi di supporto binario per l'API:
    a. Crea una nuova API o scegline una esistente. Ad esempio, noi utilizziamo l'API `FileMan`.
    b. Nell'API selezionata nel pannello di navigazione principale scegli Impostazioni API.
    c. Nel riquadro Impostazioni API scegli Gestisci tipi di supporti nella sezione Tipi di media binari.
    d. Scegli Aggiungi tipo di supporto binario.
    e. Immetti il tipo di supporto richiesto, ad esempio **`image/png`**, nel campo di input. Se necessario, ripeti questa fase per aggiungere altri tipi di supporto. Per supportare tutti i tipi di file multimediali binari, specifica `*/*`.
    f. Scegli Save changes (Salva modifiche).
2. Imposta il modo in cui i payload dei messaggi vengono gestiti per il metodo API:
    a. Crea una nuova risorsa o scegli una risorsa esistente dell'API. Ad esempio, noi utilizziamo la risorsa `/{folder}/{item}`.

b. Crea un nuovo metodo o scegli un metodo esistente della risorsa. Come esempio, utilizziamo il metodo `GET /{folder}/{item}` integrato nell'azione `Object GET` in Amazon S3.

c. Per Gestione contenuti scegli un'opzione.

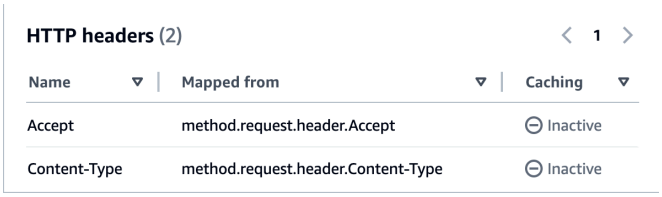


Scegliere Passthrough se non si vuole convertire il corpo quando il client e il back-end accettano lo stesso formato binario. Scegli Converti in testo per convertire il corpo binario in una stringa con codifica base64 quando, ad esempio, il back-end richiede che il payload di una richiesta binaria venga passato come proprietà JSON. Scegli quindi Converti in binario quando il client invia una stringa con codifica base64 e il back-end richiede il formato binario originale o quando l'endpoint restituisce una stringa con codifica base64 e il client accetta solo l'output binario.

d. Per Richiesta corpo passthrough scegli Quando non ci sono modelli definiti (consigliato).

Puoi anche scegliere Mai. Ciò significa che l'API rifiuterà i dati con content-types che non dispongono di un modello di mappatura.

e. Mantieni l'intestazione `Accept` della richiesta in entrata nella richiesta di integrazione. Procedi in questo modo se hai impostato `contentHandling` su `passthrough` e vuoi sovrascrivere questa impostazione al runtime.



f. Per la conversione in testo, definisci un modello di mappatura per mettere i dati binari con codifica base64 nel formato richiesto.

   Un esempio di modello di mappatura per la conversione in testo è il seguente:

```
{
  "operation": "thumbnail",
  "base64Image": "$input.body"
}
```

   Il formato di questo modello di mappatura dipende dai requisiti dell'endpoint dell'input.

g. Scegli Save (Salva).

Abilitazione del supporto binario tramite l'API REST API Gateway

Le seguenti attività mostrano come abilitare il supporto binario tramite le chiamate dell'API REST API Gateway.

Argomenti

- Aggiungi e aggiorna i tipi di supporti binari esistenti per un'API
- Configurazione delle conversioni dei payload delle richieste
- Configurazione delle conversioni dei payload delle risposte
- Conversione dei dati binari in dati di testo
- Conversione dei dati di testo in un payload binario
- Passaggio attraverso un payload binario

Aggiungi e aggiorna i tipi di supporti binari esistenti per un'API

Per fare in modo che API Gateway utilizzi un nuovo tipo di supporto binario, devi aggiungerlo all'elenco `binaryMediaTypes` della risorsa `RestApi`. Ad esempio, per fare in modo che API Gateway gestisca le immagini JPEG, invia una richiesta `PATCH` alla risorsa `RestApi`:

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/image~1jpeg"
  }
 ]
}
```

La specifica di tipo MIME di `image/jpeg`, parte del valore della proprietà `path`, viene ignorata come `image~1jpeg`.

Per aggiornare i tipi di supporto binario utilizzati, sostituisci o elimina i tipi di supporto dall'elenco `binaryMediaTypes` della risorsa `RestApi`. Ad esempio, per cambiare il supporto binario dai file JPEG ai byte non elaborati, invia una richiesta `PATCH` alla risorsa `RestApi`, come segue.

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [{
    "op" : "replace",
    "path" : "/binaryMediaTypes/image~1jpeg",
    "value" : "application/octet-stream"
  },
  {
    "op" : "remove",
    "path" : "/binaryMediaTypes/image~1jpeg"
  }]
}
```

Configurazione delle conversioni dei payload delle richieste

Se l'endpoint richiede un input binario, imposta la proprietà `contentHandling` della risorsa `Integration` su `CONVERT_TO_BINARY`. A tale scopo, invia una richiesta `PATCH`, come segue:

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/integration

{
  "patchOperations" : [ {
    "op" : "replace",
```

```
    "path" : "/contentHandling",
    "value" : "CONVERT_TO_BINARY"
  }]
}
```

Configurazione delle conversioni dei payload delle risposte

Se il client accetta il risultato come BLOB binario invece del payload con codifica base64 restituito dall'endpoint, imposta la proprietà `contentHandling` della risorsa `IntegrationResponse` su `CONVERT_TO_BINARY`. A questo scopo, inviare una richiesta `PATCH`, come segue:

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/integration/
responses/<status_code>

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "/contentHandling",
    "value" : "CONVERT_TO_BINARY"
  }]
}
```

Conversione dei dati binari in dati di testo

Per inviare dati binari come proprietà JSON dell'input a AWS Lambda o Kinesis tramite API Gateway, procedi come segue:

1. Abilita il supporto del payload binario dell'API aggiungendo il nuovo tipo di supporto binario di `application/octet-stream` all'elenco `binaryMediaTypes` dell'API.

   ```
   PATCH /restapis/<restapi_id>

   {
     "patchOperations" : [ {
       "op" : "add",
       "path" : "/binaryMediaTypes/application~1octet-stream"
     }
    ]
   }
   ```

2. Imposta `CONVERT_TO_TEXT` per la proprietà `contentHandling` della risorsa `Integration` e fornisci un modello di mappatura per assegnare la stringa con codifica base64 dei dati binari alla

proprietà JSON. Nell'esempio che segue, la proprietà JSON è `body` e `$input.body` detiene la stringa con codifica 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\"}"
    }
  ]
}
```

Conversione dei dati di testo in un payload binario

Supponiamo che una funzione Lambda restituisca un file di immagine come stringa con codifica base64. Per passare questo output binario al client tramite API Gateway, procedi come segue:

1. Aggiorna l'elenco `binaryMediaTypes` dell'API aggiungendo il tipo di supporto binario di `application/octet-stream`, se non è già presente nell'elenco.

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/application~1octet-stream",
  }]
}
```

2. Imposta la proprietà `contentHandling` della risorsa `Integration` su `CONVERT_TO_BINARY`. Non definire un modello di mappatura. Se non definisci un modello di mappatura, API Gateway

chiede al modello passthrough di restituire il BLOB binario con codifica base64 come file immagine al client.

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/
integration/responses/<status_code>

{
  "patchOperations" : [
    {
      "op" : "replace",
      "path" : "/contentHandling",
      "value" : "CONVERT_TO_BINARY"
    }
  ]
}
```

Passaggio attraverso un payload binario

Per memorizzare un'immagine in un bucket Amazon S3 tramite API Gateway, procedi come segue:

1. Aggiorna l'elenco `binaryMediaTypes` dell'API aggiungendo il tipo di supporto binario di `application/octet-stream`, se non è già presente nell'elenco.

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/application~1octet-stream"
  }
 ]
}
```

2. Imposta la proprietà `contentHandling` della risorsa `Integration` su `CONVERT_TO_BINARY`. Imposta `WHEN_NO_MATCH` come il valore della proprietà `passthroughBehavior` senza definire un modello di mappatura. Ciò consente ad API Gateway di richiamare il modello 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"
    }
  ]
}
```

Importazione ed esportazione di codifiche di contenuti per Gateway API

Per importare l'`binaryMediaTypes`elenco su un RestApi, utilizzate la seguente estensione API Gateway nel file di definizione OpenAPI dell'API. L'estensione viene anche utilizzata per esportare le impostazioni dell'API.

- Proprietà x-amazon-apigateway-binary-media-types

Per importare ed esportare il valore della proprietà `contentHandling` su una risorsa `Integration` o `IntegrationResponse`, utilizza le seguenti estensioni API Gateway per le definizioni OpenAPI:

- Oggetto di x-amazon-apigateway-integration
- Oggetto x-amazon-apigateway-integration.response

Restituzione di supporti binari da un'integrazione proxy Lambda in Gateway API

Per restituire un supporto binario da un'integrazione proxy AWS Lambda, codificare in base64 la risposta dalla funzione Lambda. È inoltre necessario configurare i tipi di supporti binari dell'API. Quando si configurano i tipi di supporti binari dell'API, l'API tratta quel tipo di contenuto come dati binari. Il limite della dimensione del payload è 10 MB.

> **Note**
>
> Per utilizzare un browser Web per richiamare un'API con questo esempio di integrazione, imposta i tipi di supporti binari dell'API su `*/*`. API Gateway utilizza la prima intestazione

`Accept` dai client per determinare se una risposta deve restituire supporti binari. Per restituire un supporto binario quando non è possibile controllare l'ordine dei valori delle intestazioni `Accept`, ad esempio le richieste da un browser, impostare i tipi di supporti binari dell'API su `*/*` (per tutti i tipi di contenuto).

Il seguente esempio di funzione Lambda può restituire ai client un'immagine binaria da Amazon S3 o del testo. La risposta della funzione include un'intestazione `Content-Type` per indicare al client il tipo di dati che restituisce. La funzione imposta in modo condizionale la proprietà `isBase64Encoded` nella risposta, a seconda del tipo di dati che restituisce.

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>",
        }
```

Per ulteriori informazioni sui tipi di supporti binari, consulta Tipi di supporti binari per REST API in Gateway API.

Accesso a file binari in Amazon S3 tramite l'API di API Gateway

Gli esempi seguenti mostrano il file OpenAPI utilizzato per accedere alle immagini in Amazon S3, come eseguire il download di un'immagine da Amazon S3 e come caricare un'immagine su Amazon S3.

Argomenti

- File OpenAPI di un'API di esempio per accedere alle immagini in Amazon S3
- Download di un'immagine da Amazon S3
- Caricamento di un'immagine su Amazon S3

File OpenAPI di un'API di esempio per accedere alle immagini in Amazon S3

Il seguente file OpenAPI mostra un'API di esempio che illustra il download di un file di immagine da Amazon S3 e il caricamento di un file di immagine su Amazon S3. Questa API espone i metodi `GET /s3?key={file-name}` e `PUT /s3?key={file-name}` per il download e il caricamento di un file di immagine specifico. Il metodo `GET` restituisce il file immagine come stringa con codifica base64, parte di un output JSON, seguita dal modello di mappatura fornito, in una risposta 200 OK. Il metodo `PUT` prende un BLOB binario non elaborato come input e restituisce una risposta 200 OK con un payload vuoto.

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"
    }
  }
}
```

Download di un'immagine da Amazon S3

Per eseguire il download di un file immagine (`image.jpg`) come BLOB binario da 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
```

La risposta andata a buon fine avrebbe questa struttura:

```
200 OK HTTP/1.1

[raw bytes]
```

I byte non elaborati vengono restituiti perché l'intestazione `Accept` è impostata sul tipo di supporto binario `application/octet-stream` e il supporto binario è abilitato per l'API.

In alternativa, per eseguire il download di un file di immagine (`image.jpg`) come stringa con codifica base64 (formattata come una proprietà JSON) da Amazon S3, aggiungi un modello di risposta alla risposta 200 di integrazione, come mostrato nel blocco di definizione OpenAPI in grassetto:

```
      "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}"
            }
          }
        },
```

Di seguito è riportato l'aspetto della richiesta di download di un file di immagine:

```
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
```

L'aspetto di una risposta andata a buon fine è il seguente:

```
200 OK HTTP/1.1

{
  "image": "W3JhdyBieXRlc10="
}
```

Caricamento di un'immagine su Amazon S3

Per caricare un file immagine (`image.jpg`) come BLOB binario su 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]
```

L'aspetto di una risposta andata a buon fine è il seguente:

```
200 OK HTTP/1.1
```

Per caricare un file immagine (`image.jpg`) come stringa con codifica base64 su 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=
```

Il payload di input deve essere una stringa con codifica base64, poiché il valore dell'intestazione `Content-Type` è impostato su `application/json`. L'aspetto di una risposta andata a buon fine è il seguente:

```
200 OK HTTP/1.1
```

Accesso a file binari in Lambda tramite l'API di API Gateway

Il seguente esempio di OpenAPI dimostra come accedere a un file binario AWS Lambda tramite un'API API Gateway. Questa API espone i metodi `GET /lambda?key={file-name}` e `PUT /lambda?key={file-name}` per il download e il caricamento di un file di immagine specifico. Il metodo `GET` restituisce il file immagine come stringa con codifica base64, parte di un output JSON, seguita dal modello di mappatura fornito, in una risposta 200 OK. Il metodo `PUT` prende un BLOB binario non elaborato come input e restituisce una risposta 200 OK con un payload vuoto.

Si crea la funzione Lambda chiamata dall'API che restituirà una stringa con codifica base64 con l'intestazione `Content-Type` di `application/json`.

Argomenti

- File OpenAPI di un'API di esempio per accedere alle immagini in Lambda
- Download di un'immagine da Lambda
- Caricamento di un'immagine su Lambda

File OpenAPI di un'API di esempio per accedere alle immagini in Lambda

Il seguente file OpenAPI mostra un'API di esempio che illustra il download di un file di immagine da Lambda e il caricamento di un file di immagine in 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"
    }
  }
}
```

Download di un'immagine da Lambda

Per eseguire il download di un file immagine (`image.jpg`) come BLOB binario da 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
```

L'aspetto di una risposta andata a buon fine è il seguente:

```
200 OK HTTP/1.1

[raw bytes]
```

Per eseguire il download di un file immagine (`image.jpg`) come stringa con codifica base64, formattato come proprietà JSON, da 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
```

L'aspetto di una risposta andata a buon fine è il seguente:

```
200 OK HTTP/1.1

{
  "image": "W3JhdyBieXRlc10="
}
```

Caricamento di un'immagine su Lambda

Per caricare un file immagine (`image.jpg`) come BLOB binario su 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]
```

L'aspetto di una risposta andata a buon fine è il seguente:

```
200 OK
```

Per caricare un file immagine (`image.jpg`) come stringa con codifica base64 su 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=
```

L'aspetto di una risposta andata a buon fine è il seguente:

```
200 OK
```

Invocazione di REST API in Gateway API

Per chiamare un'API distribuita, i client inviano richieste all'URL del servizio del componente API Gateway per l'esecuzione dell'API, ovvero `execute-api`.

L'URL di base per le API REST si presenta nel formato seguente:

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

dove *`api-id`* è l'identificativo dell'API, *`region`* è la Regione AWS e *`stage`* è il nome della fase dell'implementazione dell'API.

> **Important**
>
> Prima di invocare un'API, devi distribuirla in API Gateway. Per istruzioni sull'implementazione di un'API, consulta Implementazione di REST API in Gateway API.

Argomenti

- Ottenimento dell'URL di invocazione di un'API
- Invocazione di un'API
- Utilizzo della console API Gateway per il test di un metodo API REST
- Utilizzo di un SDK Java generato da API Gateway per un'API REST
- Utilizzo di un SDK Android generato da API Gateway per un'API REST
- Usa un JavaScript SDK generato da API Gateway per un'API REST
- Utilizzo di un SDK Ruby generato da API Gateway per un'API REST
- Usa l'SDK iOS generato da API Gateway per un'API REST in Objective-C o Swift

Ottenimento dell'URL di invocazione di un'API

Per ottenere l'URL di invocazione di un'API è possibile utilizzare la console, AWS CLI o una definizione OpenAPI esportata.

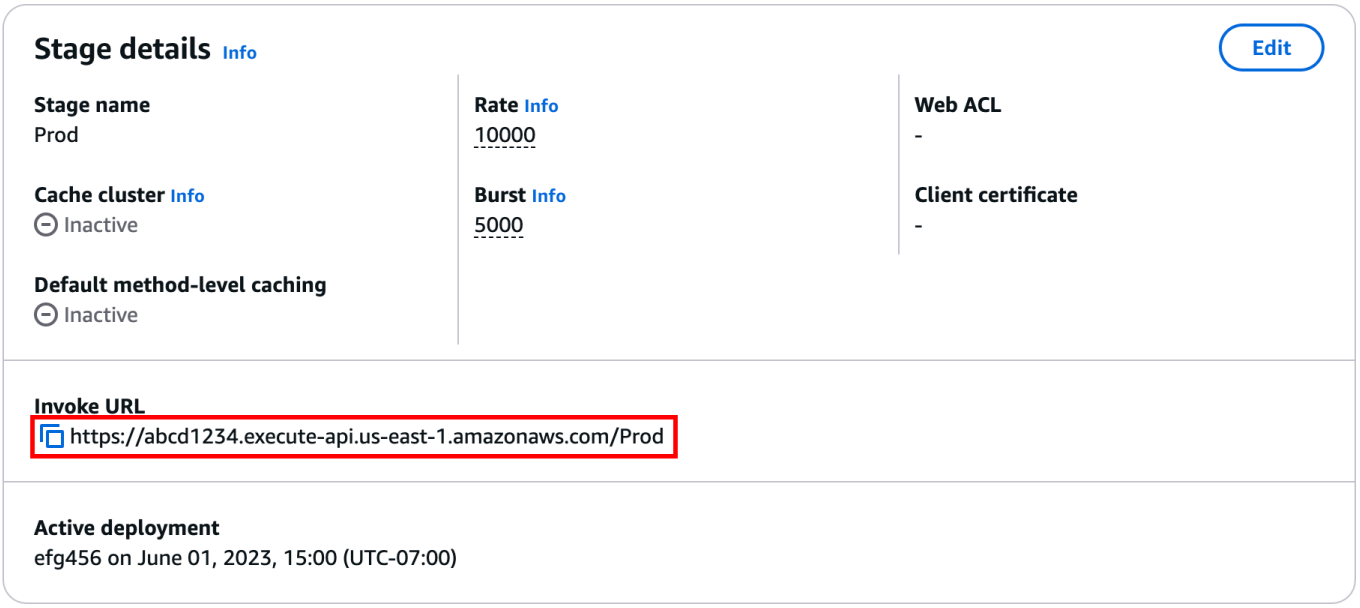
Ottenimento dell'URL di invocazione di un'API tramite la console

La procedura seguente mostra come ottenere l'URL di invocazione di un'API tramite la console REST API.

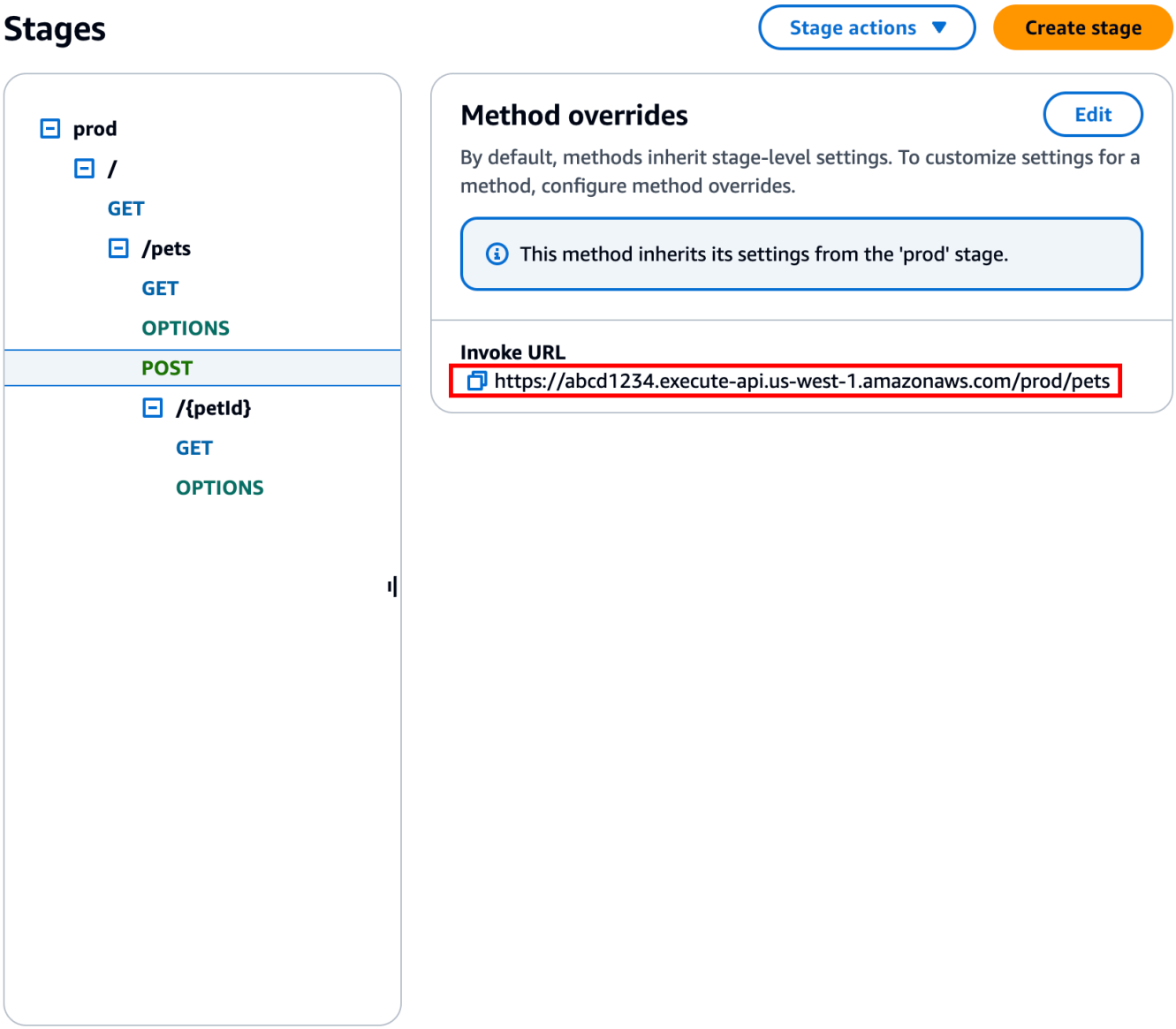
Per ottenere l'URL di invocazione di un'API tramite la console REST API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli un'API implementata.
3. Nel riquadro di navigazione principale, seleziona Log.
4. In Dettagli fase, scegli l'icona Copia per copiare l'URL di richiamo dell'API.

   Questo è l'URL per la risorsa root dell'API.



5. Per ottenere l'URL di invocazione per un'altra risorsa dell'API, espandi la fase nel pannello di navigazione secondario, quindi seleziona un metodo.
6. Scegli l'icona Copia per copiare l'URL di invocazione a livello di risorsa dell'API.



Ottenimento dell'URL di invocazione di un'API tramite AWS CLI

La procedura seguente mostra come ottenere l'URL di invocazione di un'API tramite AWS CLI.

Per ottenere l'URL di invocazione di un'API tramite AWS CLI

1. Esegui il comando seguente per ottenere il valore `rest-api-id`. Questo comando restituisce tutti i valori `rest-api-id` presenti nella Regione in uso. Per ulteriori informazioni, consulta get-rest-apis.

```
aws apigateway get-rest-apis
```

2. Sostituisci il valore `rest-api-id` di esempio con il valore `rest-api-id` da utilizzare, il valore *`{stage-name}`* di esempio con il valore *`{stage-name}`* da utilizzare, quindi sostituisci *`{region}`* con la Regione in uso.

```
https://{restapi_id}.execute-api.{region}.amazonaws.com/{stage_name}/
```

Ottenimento dell'URL di invocazione di un'API tramite il file di definizione OpenAPI esportato dell'API

È possibile anche creare l'URL root combinando i campi `host` e `basePath` di un file di definizione OpenAPI esportato dell'API. Per istruzioni su come esportare l'API, consulta the section called “Esportazione di un'API REST”.

Invocazione di un'API

È possibile chiamare l'API implementata utilizzando il browser, curl o altre applicazioni, come Postman.

Inoltre, è possibile utilizzare la console Gateway API per testare una chiamata API. Il test sfrutta la funzionalità `TestInvoke` di Gateway API, che consente di testare l'API prima che sia implementata. Per ulteriori informazioni, consulta the section called “Utilizzo della console per il test di un metodo API REST.”.

> **Note**
>
> I valori dei parametri delle stringe di query di un URL di chiamata non possono includere `%%`.

Invocazione di un'API tramite un browser web

Se l'API consente l'accesso anonimo, è possibile utilizzare qualsiasi browser web per invocare un metodo `GET`. Inserisci l'URL di invocazione completo nella barra degli indirizzi del browser.

Per altri metodi o chiamate che richiedono l'autenticazione, è necessario specificare un payload oppure firmare le richieste. Puoi gestire queste operazioni in uno script di una pagina HTML o in un'applicazione client utilizzando uno degli SDK AWS.

Invocazione di un'API tramite curl

Per chiamare l'API è possibile utilizzare uno strumento come curl nel terminale. Il seguente comando curl di esempio invoca il metodo GET sulla risorsa `getUsers` della fase `prod` di un'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"
```

Utilizzo della console API Gateway per il test di un metodo API REST

Utilizzo della console API Gateway per il test di un metodo API REST.

Argomenti

- Prerequisiti
- Test di un metodo tramite la console API Gateway

Prerequisiti

- È necessario specificare le impostazioni per i metodi che si intende testare. Segui le istruzioni in Metodi per REST APIs in API Gateway.

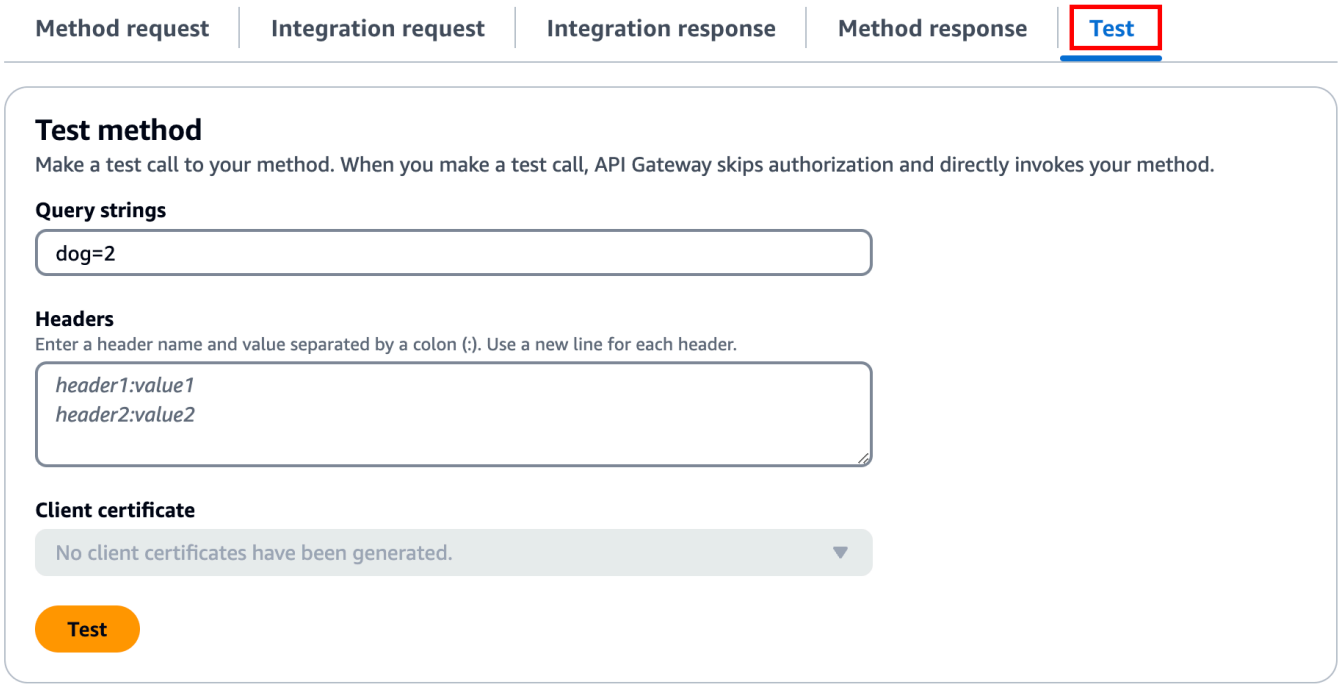
Test di un metodo tramite la console API Gateway

> **Important**
>
> Quando si testano i metodi con la console Gateway API è possibile che alle risorse vengano apportate modifiche non annullabili. Eseguire il test di un metodo tramite la console API Gateway equivale a chiamare il metodo dall'esterno della console. Ad esempio, se si utilizza la console API Gateway per chiamare un metodo che elimina le risorse di un'API e la chiamata del metodo riesce, le risorse vengono eliminate.

Test del metodo

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere una REST API.

3. Nel riquadro Resources (Risorse) scegliere il metodo che si desidera testare.
4. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.



   Immetti i valori nelle caselle visualizzate, ad esempio Stringhe di query, Intestazioni e Corpo della richiesta. La console include questi valori nella richiesta del metodo in application/json formato predefinito.

   Per specificare eventuali opzioni aggiuntive, contatta il proprietario dell'API.
5. Scegli Test (Esegui test). Verranno visualizzate le seguenti informazioni:
   - Request (Richiesta) è il percorso della risorsa chiamato per il metodo.
   - Status (Stato) è il codice dello stato HTTP della risposta.
   - Latenza in ms è l'intervallo di tempo tra la ricezione della richiesta dal chiamante e la risposta restituita.
   - Corpo della risposta è il corpo della risposta HTTP.
   - Intestazioni delle risposte sono le intestazioni di risposta HTTP.

> **Tip**
>
> A seconda della mappatura, il codice di stato HTTP, il corpo della risposta e le intestazioni di risposta potrebbero essere diversi da quelli inviati dalla funzione Lambda, dal proxy HTTP o dal proxy di servizio. AWS

- I log sono le voci simulate di Amazon CloudWatch Logs che sarebbero state scritte se questo metodo fosse stato chiamato al di fuori della console API Gateway.

> **Note**
>
> Sebbene le voci di CloudWatch Logs siano simulate, i risultati della chiamata al metodo sono reali.

Oltre a utilizzare la console API Gateway, puoi utilizzare AWS CLI o un AWS SDK per API Gateway per testare l'invocazione di un metodo. Per farlo utilizzando AWS CLI, consulta test-invoke-method.

Utilizzo di un SDK Java generato da API Gateway per un'API REST

In questa sezione verranno illustrate, a titolo di esempio, le fasi necessarie per utilizzare un SDK Java generato da API Gateway per un'API REST mediante l'API del calcolatore semplice. Per poter procedere, è necessario avere completato le fasi descritte in Genera SDKs per REST APIs in API Gateway.

Per installare e utilizzare l'SDK Java generato da API Gateway

1. Estrai il contenuto del file .zip generato da API Gateway scaricato precedentemente.
2. Scaricare e installare Apache Maven (versione 3.5 o successiva).
3. Scaricare e installare JDK 8.
4. Imposta la variabile di ambiente `JAVA_HOME`.
5. Passa alla cartella dell'SDK decompressa in cui si trova il file pom.xml. Per impostazione predefinita, questa cartella è `generated-code`. Eseguire il comando mvn install per installare i file di artefatto compilati nel repository Maven locale. Viene creata una cartella `target` contenente la libreria SDK compilata.
6. Digita il comando seguente in una directory vuota per creare uno stub di progetto del client e chiamare l'API mediante la libreria SDK installata.

```
mvn -B archetype:generate \
    -DarchetypeGroupdId=org.apache.maven.archetypes \
    -DgroupId=examples.aws.apig.simpleCalc.sdk.app \
    -DartifactId=SimpleCalc-sdkClient
```

> **Note**
>
> Nel comando precedente è stato inserito il separatore \ per maggiore leggibilità. Il comando deve trovarsi su un'unica riga senza il separatore.

Questo comando crea uno stub di applicazione. Lo stub di applicazione contiene un file `pom.xml` e una cartella `src` nella directory root del progetto (*`SimpleCalc-sdkClient`* nel comando precedente). Inizialmente sono presenti due file di origine: `src/main/java/`*`{package-path}`*`/App.java` e `src/test/java/`*`{package-path}`*`/AppTest.java`. In questo esempio *`{package-path}`* è `examples/aws/apig/simpleCalc/sdk/app`. Il precorso del pacchetto viene derivato dal valore `DarchetypeGroupdId`. Puoi usare il file `App.java` come modello per l'applicazione client e, se necessario, aggiungerne altri nella stessa cartella. Puoi usare il file `AppTest.java` come modello di test di unità per l'applicazione e, se necessario, aggiungere altri file di codice di test nella stessa cartella.

7. Aggiorna le dipendenze di pacchetto nel file `pom.xml` generato come segue sostituendo, se necessario, le proprietà `groupId`, `artifactId`, `version` e `name` del progetto:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/
POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>examples.aws.apig.simpleCalc.sdk.app</groupId>
  <artifactId>SimpleCalc-sdkClient</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>SimpleCalc-sdkClient</name>
  <url>http://maven.apache.org</url>

   <dependencies>
      <dependency>
          <groupId>com.amazonaws</groupId>
          <artifactId>aws-java-sdk-core</artifactId>
```

```
            <version>1.11.94</version>
        </dependency>
        <dependency>
            <groupId>my-apig-api-examples</groupId>
            <artifactId>simple-calc-sdk</artifactId>
            <version>1.0.0</version>
        </dependency>

      <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
      </dependency>

      <dependency>
          <groupId>commons-io</groupId>
          <artifactId>commons-io</artifactId>
          <version>2.5</version>
      </dependency>
    </dependencies>

    <build>
      <plugins>
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-compiler-plugin</artifactId>
          <version>3.5.1</version>
          <configuration>
            <source>1.8</source>
            <target>1.8</target>
          </configuration>
        </plugin>
      </plugins>
    </build>
</project>
```

> **Note**
>
> Quando una versione più recente dell'artefatto dipendente di `aws-java-sdk-core` è incompatibile con la versione specificata sopra (`1.11.94`), è necessario aggiornare il tag `<version>` alla nuova versione.

8. Quindi mostreremo come chiamare l'API mediante l'SDK chiamando i metodi `getABOp(GetABOpRequest req)`, `getApiRoot(GetApiRootRequest req)` e `postApiRoot(PostApiRootRequest req)` dell'SDK. Questi metodi corrispondono ai metodi `GET /{a}/{b}/{op}`, `GET /?a={x}&b={y}&op={operator}` e `POST /`, rispettivamente con un payload di richieste API `{"a": x, "b": y, "op": "operator"}`.

   Aggiorna il file `App.java` come segue:

```
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());


    }
}
```

Nel'esempio precedente le impostazioni di configurazione utilizzate per creare istanze del client SDK hanno esclusivamente scopo illustrativo e non sono necessariamente best practice consigliate. Anche la chiamata a `sdkClient.shutdown()` è facoltativa, soprattutto se si ha necessità di esercitare un controllo accurato su quando liberare risorse.

Abbiamo mostrato i modelli fondamentali per chiamare un'API utilizzando un SDK Java. Le istruzioni possono essere applicate per chiamare altri metodi API.

Utilizzo di un SDK Android generato da API Gateway per un'API REST

In questa sezione verranno illustrate le fasi necessarie per utilizzare un SDK Android generato da API Gateway per un'API REST. Per poter procedere, è necessario avere già completato le fasi descritte in Genera SDKs per REST APIs in API Gateway.

> **Note**
>
> L'SDK generato non è compatibile con Android 4.4 e versioni precedenti. Per ulteriori informazioni, consulta the section called “Note importanti”.

Per installare e utilizzare l'SDK Android generato da API Gateway

1. Estrai il contenuto del file .zip generato da API Gateway scaricato precedentemente.
2. Scaricare e installare Apache Maven (preferibilmente versione 3.x).
3. Scaricare e installare JDK 8.
4. Imposta la variabile di ambiente `JAVA_HOME`.
5. Eseguire il comando mvn install per installare i file di artefatto compilati nel repository Maven locale. Viene creata una cartella `target` contenente la libreria SDK compilata.
6. Copiare il file SDK (il cui nome viene derivato dai valori specificati per Artifact Id (Id artefatto) e Artifact Version (Versione artefatto) durante la generazione dell'SDK, ad esempio `simple-`

`calcsdk-1.0.0.jar`) dalla cartella `target`, insieme a tutte le altre librerie della cartella `target/lib`, nella cartella `lib` del progetto.

Se usi Android Studio, crea una cartella `libs` nel modulo della tua app client e copia il file .jar richiesto in questa cartella. Verifica che la sezione delle dipendenze nel file gradle del modulo contenga il codice seguente.

```
    compile fileTree(include: ['*.jar'], dir: 'libs')
    compile fileTree(include: ['*.jar'], dir: 'app/libs')
```

Assicurati che non siano dichiarati file .jar duplicati.

7. Usa la classe `ApiClientFactory` per inizializzare l'SDK generato da API Gateway. Ad esempio:

```
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. Per usare un fornitore di credenziali Amazon Cognito per autorizzare le chiamate all'API, usa la classe `ApiClientFactory` per passare un insieme di credenziali AWS utilizzando l'SDK generato da API Gateway, come mostrato nell'esempio seguente.

```
// 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. Per impostare una chiave API utilizzando l'SDK generato da API Gateway, usa un codice simile al seguente.

```
ApiClientFactory factory = new ApiClientFactory()
```

```
  .apiKey("YOUR_API_KEY");
```

Usa un JavaScript SDK generato da API Gateway per un'API REST

La procedura seguente mostra come utilizzare un JavaScript SDK generato da API Gateway.

> **Note**
>
> Queste istruzioni presuppongono che tu abbia già completato la procedura descritta in Genera SDKs per REST APIs in API Gateway.

> **Important**
>
> Se l'API ha solo metodi ANY definiti, il pacchetto SDK generato non conterrà un file `apigClient.js` e sarà necessario definire i metodi ANY manualmente.

Per installare, avviare e chiamare un JavaScript SDK generato da API Gateway per un'API REST

1. Estrai il contenuto del file .zip generato da API Gateway scaricato precedentemente.
2. Abilita la condivisione delle risorse multi-origine (CORS) per tutti i metodi che l'SDK generato da API Gateway chiamerà. Per istruzioni, consulta CORS per REST API in Gateway API.
3. Nella tua pagina Web, includi i riferimenti ai seguenti script.

```
<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. Nel codice inizializza l'SDK generato da API Gateway tramite un codice simile al seguente.

```
var apigClient = apigClientFactory.newClient();
```

Per inizializzare l'SDK generato da API Gateway con AWS credenziali, utilizzate un codice simile al seguente. Se utilizzi AWS le credenziali, tutte le richieste all'API verranno firmate.

```
var apigClient = apigClientFactory.newClient({
  accessKey: 'ACCESS_KEY',
  secretKey: 'SECRET_KEY',
});
```

Per utilizzare una chiave API con l'SDK generato da API Gateway, passa la chiave API come parametro all'oggetto `Factory`, usando un codice simile al seguente. Se utilizzi una chiave API, viene specificato come parte dell'intestazione `x-api-key` e tutte le richieste all'API verranno firmate. Questo significa che devi impostare le intestazioni Accept corrette di CORS per ogni richiesta.

```
var apigClient = apigClientFactory.newClient({
  apiKey: 'API_KEY'
});
```

5. Chiama i metodi API in API Gateway tramite un codice simile al seguente. Ogni chiamata restituisce una promessa con callback di successo e di fallimento.

```
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.
    });
```

Qui, *methodName* è costruito dal percorso di risorsa della richiesta del metodo e dal verbo HTTP. Per l' SimpleCalc API, i metodi SDK per i metodi API di

```
1. GET /?a=...&b=...&op=...
2. POST /

   { "a": ..., "b": ..., "op": ...}
3. GET /{a}/{b}/{op}
```

metodi SDK corrispondenti sono i seguenti:

```
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
```

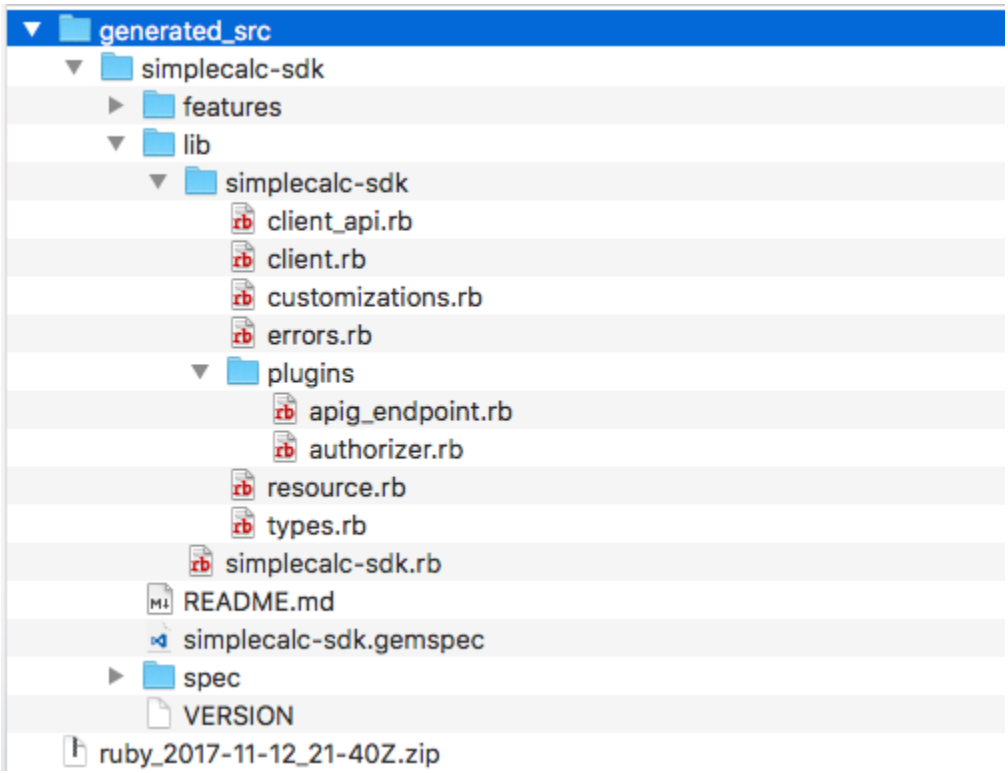
Utilizzo di un SDK Ruby generato da API Gateway per un'API REST

La seguente procedura mostra come utilizzare un SDK Ruby generato da Gateway API.

> **Note**
>
> Queste istruzioni presuppongono che sia già stata completata la procedura in Genera SDKs per REST APIs in API Gateway.

Per installare, creare istanze e chiamare un SDK Ruby generato da API Gateway per un'API REST

1. Decomprimi il file SDK Ruby scaricato. L'origine dell'SDK generato viene mostrata come segue.



2. Crea una gem Ruby dall'origine dell'SDK generato utilizzando i comandi della shell seguenti in una finestra del terminale:

```
# change to /simplecalc-sdk directory
cd simplecalc-sdk

# build the generated gem
gem build simplecalc-sdk.gemspec
```

   Dopo questa operazione, simplecalc-sdk-1.0.0.gem diventa disponibile.

3. Installa la gem:

```
gem install simplecalc-sdk-1.0.0.gem
```

4. Crea un'applicazione client. Crea un'istanza e inizializza il client SDK Ruby nell'app:

```
require 'simplecalc-sdk'
client = SimpleCalc::Client.new(
    http_wire_trace: true,
    retry_limit: 5,
    http_read_timeout: 50
)
```

Se l'API dispone di un'autorizzazione del `AWS_IAM` tipo configurato, è possibile includere le AWS credenziali del chiamante fornendo `accessKey` e `secretKey` durante l'inizializzazione:

```
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. Fai chiamate API mediante l'SDK nell'app.

> **Tip**
>
> Se non hai familiarità con le convenzioni delle chiamate ai metodi SDK, esamina il file `client.rb` nella cartella `lib` dell'SDK generato. La cartella contiene la documentazione relativa a ciascuna chiamata ai metodi API supportati.

Per conoscere le operazioni supportate:

```
# to show supported operations:
puts client.operation_names
```

I risultati sono visualizzati di seguito e corrispondono ai metodi API rispettivamente di `GET /?a={.}&b={.}&op={.}`, `GET /{a}/{b}/{op}` e `POST /`, oltre a un payload del formato `{a:"…", b:"…", op:"…"}`:

```
[:get_api_root, :get_ab_op, :post_api_root]
```

Per invocare il metodo API `GET /?a=1&b=2&op=+`, chiama il metodo SDK Ruby seguente:

```
resp = client.get_api_root({a:"1", b:"2", op:"+"})
```

Per invocare il metodo API `POST /` con un payload di `{a: "1", b: "2", "op": "+"}`, chiama il metodo SDK Ruby seguente:

```
resp = client.post_api_root(input: {a:"1", b:"2", op:"+"})
```

Per invocare il metodo API `GET /1/2/+`, chiama il metodo SDK Ruby seguente:

```
resp = client.get_ab_op({a:"1", b:"2", op:"+"})
```

Le chiamate ai metodi SDK che riescono restituiscono la risposta seguente:

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

Usa l'SDK iOS generato da API Gateway per un'API REST in Objective-C o Swift

In questo tutorial, mostreremo come utilizzare un SDK iOS generato da API Gateway per un'API REST in un'app Objective-C o Swift per chiamare l'API sottostante. Useremo l'SimpleCalc API come esempio per illustrare i seguenti argomenti:

- Come installare i componenti AWS Mobile SDK richiesti nel tuo progetto Xcode
- Come creare l'oggetto client dell'API prima di chiamare i metodi dell'API
- Come chiamare i metodi API attraverso i metodi SDK corrispondenti nell'oggetto client dell'API
- Come preparare un input di metodo e analizzare il risultato utilizzando le classi di modello corrispondenti dell'SDK

Argomenti

- Usa l'SDK iOS generato (Objective-C) per chiamare l'API
- Utilizzo di un SDK iOS (Swift) generato per chiamare l'API

Usa l'SDK iOS generato (Objective-C) per chiamare l'API

Prima di iniziare la procedura seguente, è necessario completare i passaggi in Genera SDKs per REST APIs in API Gateway per iOS in Objective-C e scaricare il file.zip dell'SDK generato.

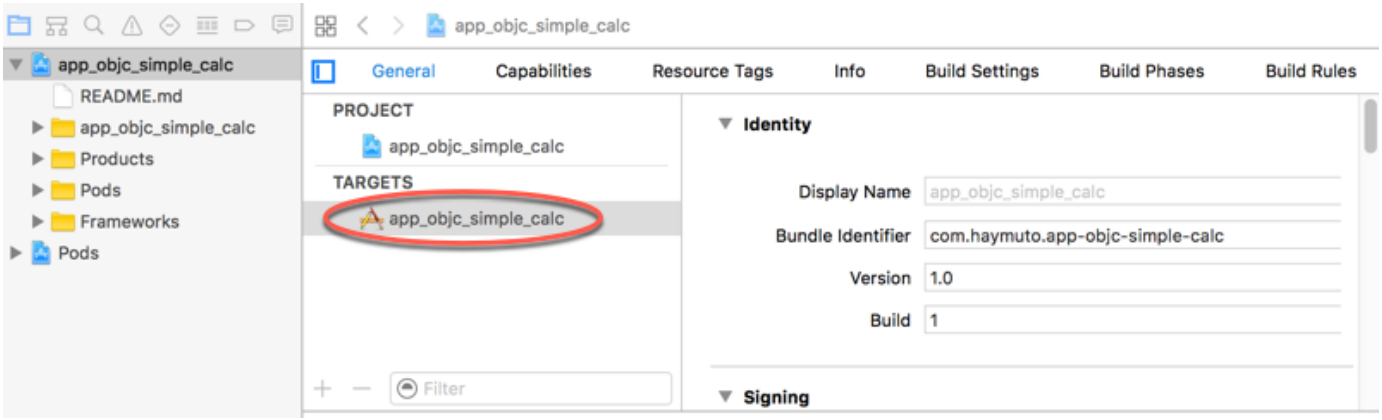
Installa il AWS SDK mobile e un SDK iOS generato da API Gateway in un progetto Objective-C

La procedura seguente descrive come installare l'SDK.

Per installare e utilizzare un SDK iOS generato da API Gateway in Objective-C

1. Estrai il contenuto del Gateway-generated file.zip dell'API che hai scaricato in precedenza. Utilizzando l'SimpleCalc API, potresti voler rinominare la cartella SDK decompressa con qualcosa del genere. **`sdk_objc_simple_calc`** In questa cartella SDK sono presenti un file `README.md` e un file `Podfile`. Il file `README.md` contiene le istruzioni per installare e usare l'SDK. Questo tutorial fornisce i dettagli relativi alle istruzioni. L'installazione consente di CocoaPodsimportare le librerie API Gateway richieste e altri componenti AWS Mobile SDK dipendenti. Per importare gli SDK nel progetto Xcode dell'app, devi aggiornare il `Podfile`. La cartella SDK non archiviata contiene anche una cartella `generated-src` con il codice sorgente dell'SDK generato dell'API.

2. Avvia Xcode e crea un nuovo Objective-C progetto iOS. Prendi nota della destinazione del progetto. Dovrai specificare questa impostazione nel `Podfile`.



3. Per AWS Mobile SDK for iOS importarlo nel progetto Xcode utilizzando CocoaPods, procedi come segue:

   a. Installa CocoaPods eseguendo il seguente comando in una finestra di terminale:

   ```
   sudo gem install cocoapods
   pod setup
   ```

   b. Copia il file `Podfile` dalla cartella SDK estratta nella stessa directory in cui si trova il file di progetto Xcode. Sostituisci il blocco seguente:

   ```
   target '<YourXcodeTarget>' do
       pod 'AWSAPIGateway', '~> 2.4.7'
   end
   ```

   con il nome di destinazione del progetto:

   ```
   target 'app_objc_simple_calc' do
       pod 'AWSAPIGateway', '~> 2.4.7'
   end
   ```

   Se il progetto Xcode già contiene un file denominato `Podfile`, aggiungi la riga di codice seguente:

   ```
   pod 'AWSAPIGateway', '~> 2.4.7'
   ```

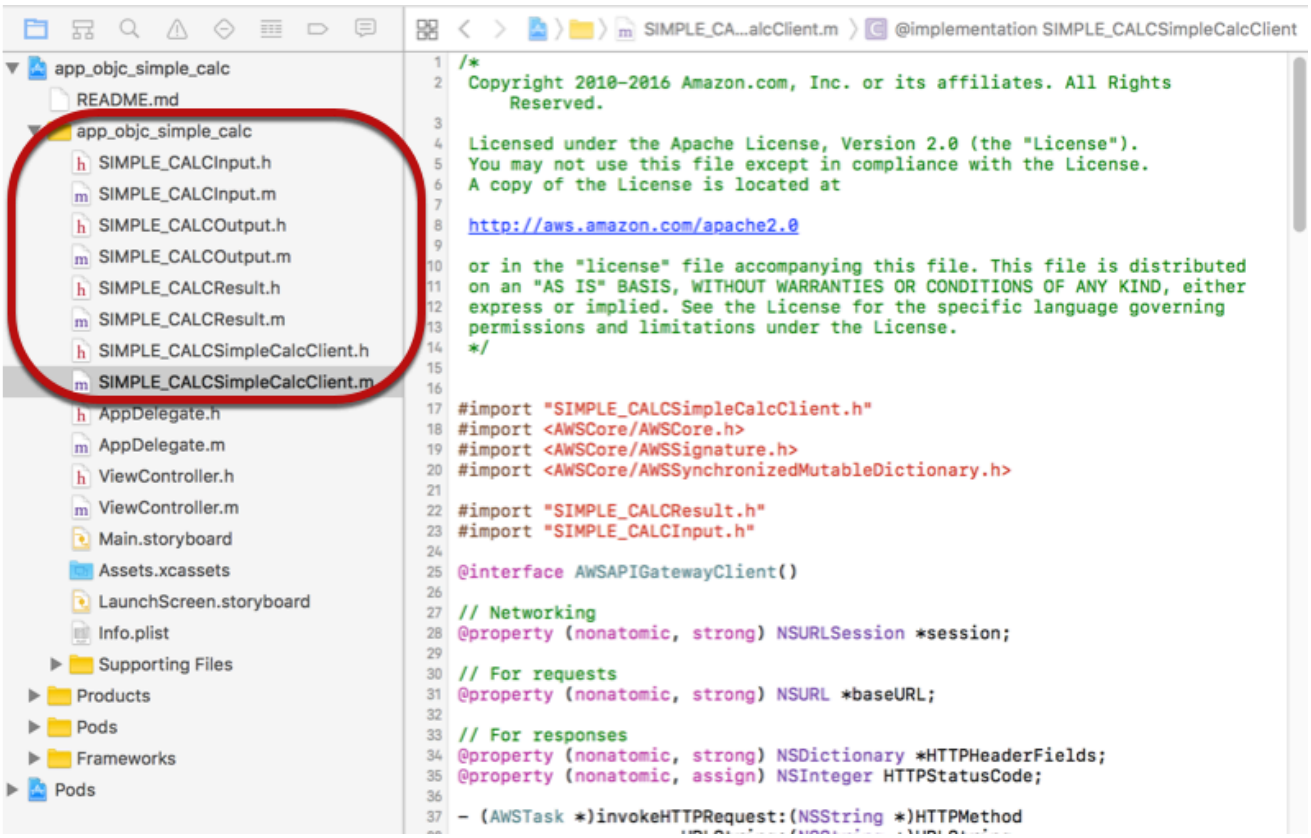
   c. Apri una finestra del terminale ed esegui il comando seguente:

```
pod install
```

Questo installa il componente API Gateway e altri componenti AWS Mobile SDK dipendenti.

d. Chiudi il progetto Xcode e apri il file `.xcworkspace` per riavviare Xcode.

e. Aggiungi tutti i file `.h` e `.m` dalla directory `generated-src` estratta dell'SDK nel progetto Xcode.



Per importarli AWS Mobile SDK for iOS Objective-C nel tuo progetto scaricando esplicitamente AWS Mobile SDK o utilizzando Carthage, segui le istruzioni contenute nel file. README.md Assicurati di utilizzare solo una di queste opzioni per importare Mobile SDK. AWS

Chiama i metodi API utilizzando l'SDK iOS generato da API Gateway in un progetto Objective-C

Quando avete generato l'SDK con il prefisso di `SIMPLE_CALC` for questa SimpleCalc API con due modelli per l'input (`Input`) e l'output (`Result`) dei metodi, nell'SDK, la classe client API risultante diventa `SIMPLE_CALCSimpleCalcClient` e le classi di dati corrispondenti sono

`SIMPLE_CALCInput` e, rispettivamente. `SIMPLE_CALCResult` Le richieste e le risposte API sono mappate ai metodi SDK come segue:

- La richiesta API di

  ```
  GET /?a=...&b=...&op=...
  ```

  diventa il metodo SDK di

  ```
  (AWSTask *)rootGet:(NSString *)op a:(NSString *)a b:(NSString *)b
  ```

  La proprietà `AWSTask.result` è del tipo `SIMPLE_CALCResult`, se il modello `Result` è stato aggiunto alla risposta del metodo. In caso contrario, è del tipo `NSDictionary`.

- Questa richiesta API di

  ```
  POST /

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

  diventa il metodo SDK di

  ```
  (AWSTask *)rootPost:(SIMPLE_CALCInput *)body
  ```

- La richiesta API di

  ```
  GET /{a}/{b}/{op}
  ```

  diventa il metodo SDK di

  ```
  (AWSTask *)aBOpGet:(NSString *)a b:(NSString *)b op:(NSString *)op
  ```

La procedura seguente descrive come chiamare i metodi API nel codice sorgente Objective-C dell'app, ad esempio come parte del `viewDidLoad` delegato in un file. `ViewController.m`

Per chiamare l'API mediante l'SDK iOS generato da API Gateway

1. Importa il file di intestazione della classe client dell'API per rendere tale classe chiamabile nell'app:

```
#import "SIMPLE_CALCSimpleCalc.h"
```

   L'istruzione `#import` inoltre importa `SIMPLE_CALCInput.h` e `SIMPLE_CALCResult.h` per le due classi di modelli.

2. Crea un'istanza della classe client dell'API:

```
SIMPLE_CALCSimpleCalcClient *apiInstance = [SIMPLE_CALCSimpleCalcClient
 defaultClient];
```

   Per utilizzare Amazon Cognito con l'API, impostare la proprietà `defaultServiceConfiguration` sull'oggetto `AWSServiceManager` predefinito, come mostrato di seguito, prima di chiamare il metodo `defaultClient` per creare l'oggetto client dell'API (mostrato nell'esempio precedente):

```
AWSCognitoCredentialsProvider *creds = [[AWSCognitoCredentialsProvider alloc]
 initWithRegionType:AWSRegionUSEast1 identityPoolId:your_cognito_pool_id];
AWSServiceConfiguration *configuration = [[AWSServiceConfiguration alloc]
 initWithRegion:AWSRegionUSEast1 credentialsProvider:creds];
AWSServiceManager.defaultServiceManager.defaultServiceConfiguration =
 configuration;
```

3. Chiama il metodo `GET /?a=1&b=2&op=+` per eseguire 1+2:

```
[[apiInstance rootGet: @"+" a:@"1" b:@"2"] continueWithBlock:^id _Nullable(AWSTask
 * _Nonnull task) {
    _textField1.text = [self handleApiResponse:task];
    return nil;
}];
```

   dove la funzione dell'helper `handleApiResponse:task` formatta il risultato come stringa da visualizzare in un campo di testo (`_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;
}
```

Il risultato visualizzato è `1 + 2 = 3`.

4. Chiama `POST /` con un payload per eseguire `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;
    }];
```

Il risultato visualizzato è `1 - 2 = -1`.

5. Chiama `GET /{a}/{b}/{op}` per eseguire `1/2`:

```
[[apiInstance aBOpGet:@"1" b:@"2" op:@"div"] continueWithBlock:^id
 _Nullable(AWSTask * _Nonnull task) {
    _textField3.text = [self handleApiResponse:task];
    return nil;
}];
```

Il risultato visualizzato è `1 div 2 = 0.5`. Nell'esempio, `div` viene usato al posto di `/` perché la funzione Lambda semplice nel back-end non gestisce le variabili di percorso con codifica URL.

Utilizzo di un SDK iOS (Swift) generato per chiamare l'API

Prima di iniziare la procedura seguente, devi completare le fasi descritte in Genera SDKs per REST APIs in API Gateway per iOS in Swift e scaricare il file .zip dell'SDK generato.

Argomenti

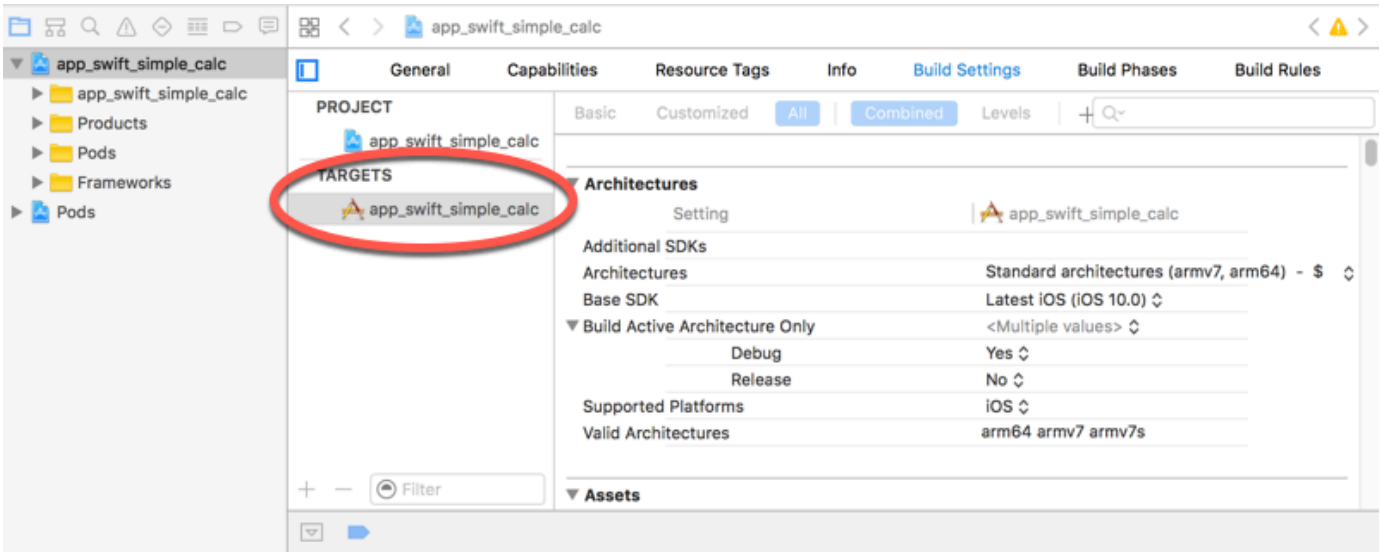
- Installa AWS SDK mobile e API Gateway-generated SDK in un progetto Swift
- Chiamata ai metodi API mediante l'SDK iOS generato da API Gateway in un progetto Swift

Installa AWS SDK mobile e API Gateway-generated SDK in un progetto Swift

La procedura seguente descrive come installare l'SDK.

Per installare e utilizzare un SDK iOS generato da API Gateway in Swift

1. Estrai il contenuto del Gateway-generated file.zip dell'API che hai scaricato in precedenza. Utilizzando l'SimpleCalc API, potresti voler rinominare la cartella SDK decompressa con qualcosa del genere. **`sdk_swift_simple_calc`** In questa cartella SDK sono presenti un file `README.md` e un file `Podfile`. Il file `README.md` contiene le istruzioni per installare e usare l'SDK. Questo tutorial fornisce i dettagli relativi alle istruzioni. L'installazione consente di importare i componenti Mobile CocoaPodsSDK richiesti. AWS Per importare gli SDK nel progetto Xcode dell'app Swift, devi aggiornare il `Podfile`. La cartella SDK non archiviata contiene anche una cartella `generated-src` con il codice sorgente dell'SDK generato dell'API.
2. Avvia Xcode e crea un nuovo progetto Swift per iOS. Prendi nota della destinazione del progetto. Dovrai specificare questa impostazione nel `Podfile`.



3. Per importare i componenti AWS Mobile SDK richiesti nel progetto Xcode utilizzando CocoaPods, procedi come segue:

   a. Se non è installato, installalo CocoaPods eseguendo il seguente comando in una finestra di terminale:

```
sudo gem install cocoapods
pod setup
```

b. Copia il file `Podfile` dalla cartella SDK estratta nella stessa directory in cui si trova il file di progetto Xcode. Sostituisci il blocco seguente:

```
target '<YourXcodeTarget>' do
    pod 'AWSAPIGateway', '~> 2.4.7'
end
```

con il nome di destinazione del progetto, come mostrato

```
target 'app_swift_simple_calc' do
    pod 'AWSAPIGateway', '~> 2.4.7'
end
```

Se il progetto Xcode già contiene un `Podfile` con la destinazione corretta, puoi semplicemente aggiungere la riga di codice seguente al loop `do ... end`:

```
pod 'AWSAPIGateway', '~> 2.4.7'
```

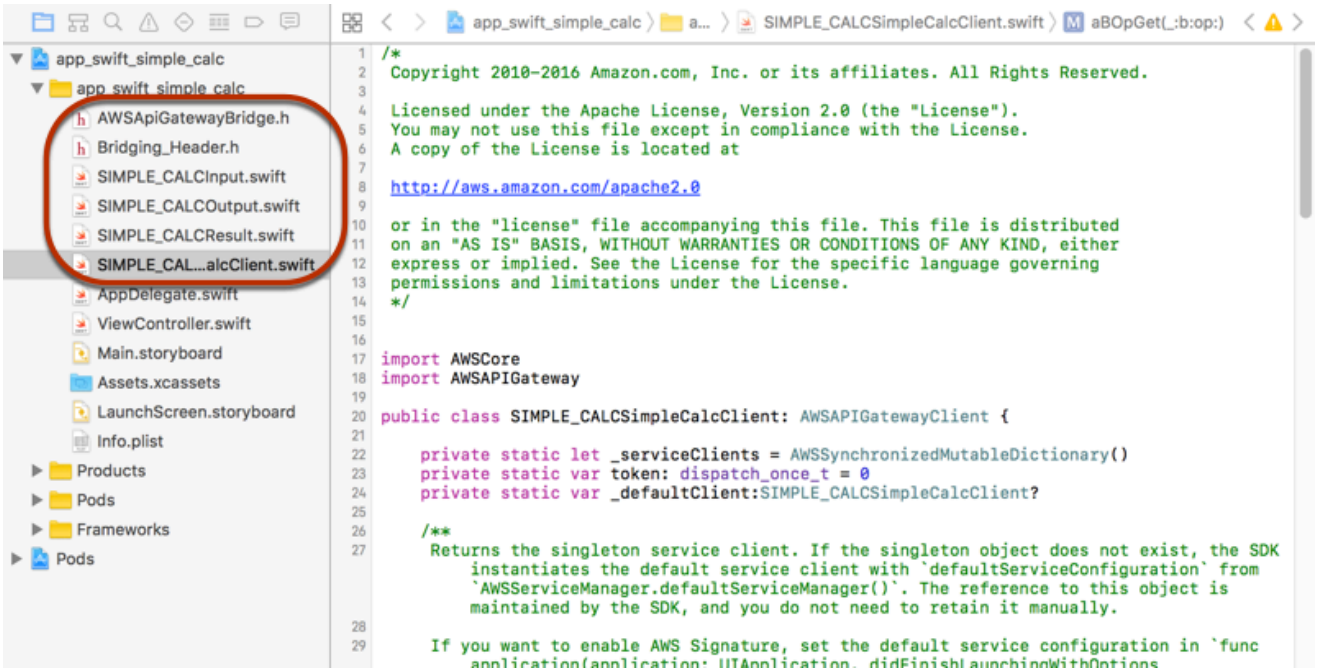
c. Apri una finestra del terminale ed esegui il comando seguente nella directory dell'app:

```
pod install
```

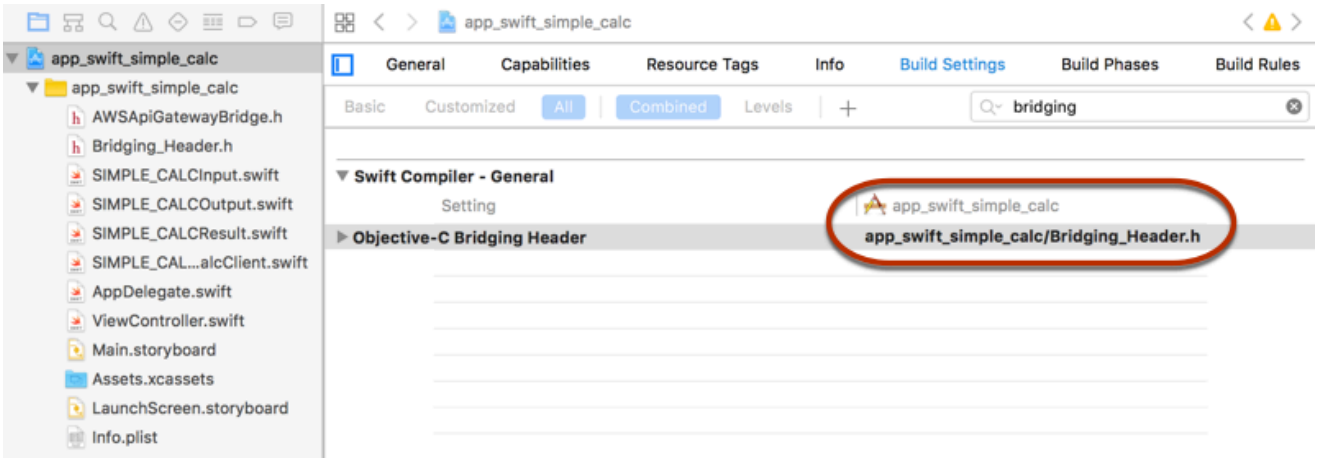
Questo installa il componente API Gateway e tutti i componenti AWS Mobile SDK dipendenti nel progetto dell'app.

d. Chiudi il progetto Xcode e apri il file `*.xcworkspace` per riavviare Xcode.

e. Aggiungi tutti i file di intestazione dell'SDK (`.h`) e i file del codice di origine Swift (`.swift`) dalla directory `generated-src` estratta nel progetto Xcode.



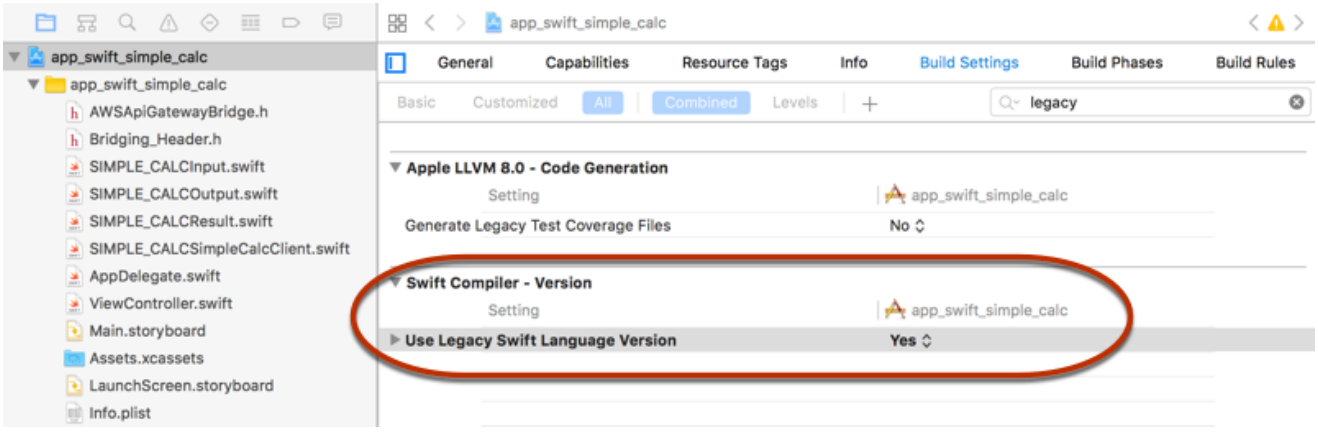
f. Per abilitare la chiamata alle Objective-C librerie di AWS Mobile SDK dal tuo progetto di codice Swift, imposta il percorso del `Bridging_Header.h` file nella proprietà Objective-C Bridging Header in Swift Compiler - Impostazione generale della configurazione del tuo progetto Xcode:



> **Tip**
>
> Puoi digitare **`bridging`** nella casella di ricerca di Xcode per individuare la proprietà Bridging Header. Objective-C

g. Prima di procedere, crea il progetto Xcode per verificare che sia configurato correttamente. Se il tuo Xcode utilizza una versione di Swift più recente di quella supportata per AWS

Mobile SDK, otterrai errori nel compilatore Swift. In questo caso, impostare la proprietà Use Legacy Swift Language Version (Usa versione linguaggio Swift legacy) su Yes (Sì) in Swift Compiler - Version (Compilatore Swift - Versione):



Per importare AWS Mobile SDK for iOS in Swift nel tuo progetto AWS scaricando esplicitamente Mobile SDK o utilizzando Carthage, segui le istruzioni nel `README.md` file fornito con il pacchetto SDK. Assicurati di utilizzare solo una di queste opzioni per importare Mobile SDK. AWS

Chiamata ai metodi API mediante l'SDK iOS generato da API Gateway in un progetto Swift

Quando avete generato l'SDK con il prefisso di `SIMPLE_CALC` for questa SimpleCalc API con due modelli per descrivere l'input (`Input`) e l'output (`Result`) delle richieste e delle risposte dell'API, nell'SDK, la classe client API risultante diventa `SIMPLE_CALCSimpleCalcClient` e le classi di dati corrispondenti sono `SIMPLE_CALCInput` e, rispettivamente. `SIMPLE_CALCResult` Le richieste e le risposte API sono mappate ai metodi SDK come segue:

- La richiesta API di

```
GET /?a=...&b=...&op=...
```

diventa il metodo SDK di

```
public func rootGet(op: String?, a: String?, b: String?) -> AWSTask
```

La proprietà `AWSTask.result` è del tipo `SIMPLE_CALCResult`, se il modello `Result` è stato aggiunto alla risposta del metodo. In caso contrario, è del tipo `NSDictionary`.

- Questa richiesta API di

```
POST /

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

  diventa il metodo SDK di

```
public func rootPost(body: SIMPLE_CALCInput) -> AWSTask
```

- La richiesta API di

```
GET /{a}/{b}/{op}
```

  diventa il metodo SDK di

```
public func aBOpGet(a: String, b: String, op: String) -> AWSTask
```

La procedura seguente descrive come chiamare i metodi API nel codice di origine delle app Swift, ad esempio nell'ambito del delegato `viewDidLoad()` in un file `ViewController.m`.

Per chiamare l'API mediante l'SDK iOS generato da API Gateway

1. Crea un'istanza della classe client dell'API:

   ```
   let client = SIMPLE_CALCSimpleCalcClient.default()
   ```

   Per utilizzare Amazon Cognito con l'API, imposta una configurazione di AWS servizio predefinita (mostrata di seguito) prima di ottenere il `default` metodo (mostrato in precedenza):

   ```
   let credentialsProvider =
    AWSCognitoCredentialsProvider(regionType: AWSRegionType.USEast1, identityPoolId:
    "my_pool_id")
   let configuration = AWSServiceConfiguration(region: AWSRegionType.USEast1,
    credentialsProvider: credentialsProvider)
   ```

```
AWSServiceManager.defaultServiceManager().defaultServiceConfiguration =
 configuration
```

2. Chiama il metodo `GET /?a=1&b=2&op=+` per eseguire 1+2:

```
client.rootGet("+", a: "1", b:"2").continueWithBlock {(task: AWSTask) -> AnyObject?
 in
    self.showResult(task)
    return nil
}
```

dove la funzione dell'helper `self.showResult(task)` stampa il risultato o l'errore nella console, ad esempio:

```
func showResult(task: AWSTask) {
    if let error = task.error {
        print("Error: \(error)")
    } else if let result = task.result {
        if result is SIMPLE_CALCResult {
            let res = result as! SIMPLE_CALCResult
            print(String(format:"%@ %@ %@ = %@", res.input!.a!, res.input!.op!,
 res.input!.b!, res.output!.c!))
        } else if result is NSDictionary {
            let res = result as! NSDictionary
            print("NSDictionary: \(res)")
        }
    }
}
```

In un'app di produzione puoi visualizzare il risultato o l'errore in un campo di testo. Il risultato visualizzato è `1 + 2 = 3`.

3. Chiama `POST /` con un payload per eseguire `1-2`:

```
let body = SIMPLE_CALCInput()
body.a=1
body.b=2
body.op="-"
client.rootPost(body).continueWithBlock {(task: AWSTask) -> AnyObject? in
    self.showResult(task)
    return nil
}
```

Il risultato visualizzato è `1 - 2 = -1`.

4. Chiama `GET /{a}/{b}/{op}` per eseguire `1/2`:

```
client.aBOpGet("1", b:"2", op:"div").continueWithBlock {(task: AWSTask) ->
 AnyObject? in
    self.showResult(task)
    return nil
}
```

Il risultato visualizzato è `1 div 2 = 0.5`. Nell'esempio, `div` viene usato al posto di `/` perché la funzione Lambda semplice nel back-end non gestisce le variabili di percorso con codifica URL.

Sviluppo di REST API utilizzando OpenAPI in Gateway API

Puoi utilizzare API Gateway per importare un'API REST da un file di definizione esterno in API Gateway. Attualmente, API Gateway supporta i file di definizione v2.0 e OpenAPI v3.0 con le eccezioni elencate in Note importanti su Amazon API Gateway per REST APIs. Puoi aggiornare un'API sovrascrivendola con una nuova definizione oppure puoi unire la definizione a quella di un'API esistente. Puoi specificare le opzioni utilizzando un parametro di query `mode` nell'URL di richiesta.

Per un tutorial sull'utilizzo della caratteristica Importa API dalla console API Gateway, consulta Tutorial: creazione di un'API REST mediante l'importazione di un esempio.

Argomenti

- Importazione di un'API ottimizzata per l'edge in API Gateway
- Importazione di un'API regionale in Gateway API
- Importazione di un file OpenAPI per aggiornare una definizione API esistente
- Impostare la proprietà OpenAPI basePath
- AWS variabili per l'importazione OpenAPI
- Errori e avvisi relativi all'importazione dell'API in Gateway API
- Esportazione di un'API REST da API Gateway

Importazione di un'API ottimizzata per l'edge in API Gateway

Puoi importare un file di definizione OpenAPI dell'API per creare una nuova API ottimizzata per gli edge specificando il tipo di endpoint `EDGE` come input aggiuntivo, oltre al file OpenAPI, nell'operazione di importazione. Puoi farlo utilizzando la console API Gateway o un AWS SDK. AWS CLI

Per un tutorial sull'utilizzo della caratteristica Importa API dalla console API Gateway, consulta Tutorial: creazione di un'API REST mediante l'importazione di un esempio.

Argomenti

- Importazione di un'API ottimizzata per l'edge mediante la console API Gateway
- Importa un'API ottimizzata per i dispositivi periferici utilizzando AWS CLI

Importazione di un'API ottimizzata per l'edge mediante la console API Gateway

Per importare un'API ottimizzata per l'edge tramite la console API Gateway, procedi nel seguente modo:

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Seleziona Create API (Crea API).
3. In API REST, scegliere Import (Importa).
4. Copia una definizione OpenAPI dell'API e incollala nell'editor di codice o seleziona Scegli il file per caricare un file OpenAPI da un'unità locale.
5. In Tipo di endpoint, scegli Ottimizzato per gli edge.
6. Scegli Crea API per iniziare a importare le definizioni OpenAPI.

Importa un'API ottimizzata per i dispositivi periferici utilizzando AWS CLI

Il import-rest-apicomando seguente importa un'API da un file di definizione OpenAPI per creare una nuova API ottimizzata per i bordi:

```
aws apigateway import-rest-api \
    --fail-on-warnings \
    --body 'file://path/to/API_OpenAPI_template.json'
```

oppure specificando esplicitamente il parametro della stringa di query `endpointConfigurationTypes` come `EDGE`:

```
aws apigateway import-rest-api \
    --parameters endpointConfigurationTypes=EDGE \
    --fail-on-warnings \
    --body 'file://path/to/API_OpenAPI_template.json'
```

Importazione di un'API regionale in Gateway API

Quando importi un'API, puoi scegliere la configurazione dell'endpoint regionale per l'API. Puoi utilizzare la console API Gateway AWS CLI, o un AWS SDK.

Quando esporti un'API, la configurazione dell'endpoint dell'API non è inclusa nelle definizioni dell'API esportate.

Per un tutorial sull'utilizzo della caratteristica Importa API dalla console API Gateway, consulta Tutorial: creazione di un'API REST mediante l'importazione di un esempio.

Argomenti

- Importazione di un'API regionale tramite la console API Gateway
- Importa un'API regionale utilizzando AWS CLI

Importazione di un'API regionale tramite la console API Gateway

Per importare un'API di un endpoint regionale mediante la console API Gateway, procedi come indicato di seguito:

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Seleziona Create API (Crea API).
3. In API REST, scegliere Import (Importa).
4. Copia una definizione OpenAPI dell'API e incollala nell'editor di codice o seleziona Scegli il file per caricare un file OpenAPI da un'unità locale.
5. In Tipo di endpoint, scegliRegionale.
6. Scegli Crea API per iniziare a importare le definizioni OpenAPI.

Importa un'API regionale utilizzando AWS CLI

Il import-rest-apicomando seguente importa un file di definizione OpenAPI e imposta il tipo di endpoint su Regional:

```
aws apigateway import-rest-api \
    --parameters endpointConfigurationTypes=REGIONAL \
    --fail-on-warnings \
    --body 'file://path/to/API_OpenAPI_template.json'
```

Importazione di un file OpenAPI per aggiornare una definizione API esistente

Puoi importare le definizioni API solo per aggiornare un'API esistente, senza cambiare la configurazione del suo endpoint, le fasi, le variabili di fase o i riferimenti alle chiavi API.

L'operazione "importare per aggiornare" può avvenire in due modi: per unione o per sovrascrittura.

Quando un'API (`A`) viene unita a un'altra (`B`), l'API risultante conserva le definizioni sia di `A` sia di `B` se le due API non condividono definizioni in conflitto. Quando si verificano dei conflitti, le definizioni di metodo dell'API di unione (`A`) sostituisce le definizioni di metodo corrispondenti dell'API unita (`B`). Ad esempio, supponiamo che `B` abbia dichiarato i seguenti metodi per restituire le risposte `200` e `206`:

```
GET /a
POST /a
```

e `A` dichiari il metodo seguente per restituire le risposte `200` e `400`:

```
GET /a
```

Quando `A` viene unita a `B`, l'API risultante produrrà i seguenti metodi:

```
GET /a
```

che restituisce le risposte `200` e `400` e

```
POST /a
```

che restituisce le risposte `200` e `206`.

Unire un'API è utile quando hai scomposto le definizioni API esterne in molteplici parti più piccole e vuoi applicare le modifiche solo a una parte per volta. Ad esempio, questo può succedere se più team

sono responsabili di parti diverse di un'API e hanno modifiche disponibili a tariffe diverse. In questo modo, gli item dell'API esistente che non vengono esplicitamente indicati nella definizione importata saranno tralasciati.

Quando un'API (A) sovrascrive un'altra API (B), l'API risultante prende le definizioni dell'API di sovrascrittura (A). Sovrascrivere un'API è utile quando una definizione API esterna ne contiene la definizione completa. In questo modo, gli item dell'API esistente che non vengono esplicitamente indicati nella definizione importata saranno eliminati.

> **Important**
>
> Quando si utilizza`mode=overwrite`, l'intera definizione dell'API viene sostituita. Tutte le configurazioni API non incluse esplicitamente nel corpo di importazione vengono ripristinate ai valori predefiniti. Ciò include impostazioni come politiche di sicurezza, configurazioni di autorizzazione e altre proprietà. API-level
> Per preservare le configurazioni esistenti, utilizzate`mode=merge`, che conserva le configurazioni esistenti che non sono specificate nel corpo di importazione. Se lo utilizzi`mode=overwrite`, assicurati che la tua definizione OpenAPI includa tutte le configurazioni API desiderate, come le estensioni API Gateway.

Per unire un'API, invia una richiesta PUT a `https://apigateway.<region>.amazonaws.com/restapis/<restapi_id>?mode=merge`. Il valore del parametro percorso `restapi_id` specifica l'API con cui la definizione API fornita si unirà.

Il seguente frammento di codice mostra un esempio della richiesta PUT di unione di una definizione API OpenAPI in JSON, come payload, con l'API specificata già presente in 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
```

L'operazione di aggiornamento dell'unione consiste nell'unione di due definizioni API complete. Per una modifica piccola e incrementale, puoi utilizzare l'operazione di aggiornamento risorsa.

Per sovrascrivere un'API, invia una richiesta PUT a `https://apigateway.<region>.amazonaws.com/restapis/<restapi_id>?mode=overwrite`. Il parametro percorso `restapi_id` specifica l'API che verrà sovrascritta dalle definizioni API fornite.

Il seguente frammento di codice mostra un esempio di richiesta di sovrascrittura con il payload di una definizione: JSON-formatted OpenAPI

```
PUT /restapis/<restapi_id>?mode=overwrite
Host:apigateway.<region>.amazonaws.com
Content-Type: application/json
Content-Length: ...

An OpenAPI API definition in JSON
```

Se il parametro di query `mode` non viene specificato, si procede con l'unione.

> **Note**
>
> Le operazioni PUT sono idempotenti, ma non atomiche. Ciò significa che se si verifica un errore prima della fine dell'elaborazione, l'API può risultare in uno stato non valido. Tuttavia, ripetendo l'operazione con successo, l'API potrà essere nello stesso stato finale che avrebbe avuto se la prima operazione fosse riuscita.

Impostare la proprietà OpenAPI basePath

In OpenAPI 2.0 è possibile usare la proprietà `basePath` per specificare una o più parti di percorso che precedono ogni percorso definito nella proprietà `paths`. Poiché API Gateway include diversi modi per esprimere il percorso di una risorsa, la caratteristica Importa API offre queste opzioni per interpretare la proprietà `basePath` durante l'importazione: ignore, prepend e split.

In OpenAPI 3.0 `basePath` non è più una proprietà di primo livello. Al contrario, API Gateway usa una variabile server come convenzione. La funzionalità Import API (Importa API) offre le stesse opzioni per interpretare il percorso di base durante l'importazione. Il percorso di base può essere identificato in questo modo:

- Se l'API non contiene variabili `basePath`, la funzionalità Import API (Importa API) controlla la stringa `server.url` per verificare se contiene un percorso dopo "`/`". Se sì, il percorso viene usato come percorso di base.
- Se l'API contiene solo una variabile `basePath`, la funzionalità Import API (Importa API) la usa come percorso di base, anche se alla variabile non viene fatto riferimento in `server.url`.
- Se l'API contiene più variabili `basePath`, la funzionalità Import API (Importa API) usa solo la prima come percorso di base.

Ignorare

Se nel file OpenAPI il valore di `basePath` è `/a/b/c` e la proprietà `paths` contiene `/e` e `/f`, la richiesta `POST` o `PUT` seguente:

```
POST /restapis?mode=import&basepath=ignore
```

```
PUT /restapis/api_id?basepath=ignore
```

porterà le seguenti risorse nell'API:

- `/`
- `/e`
- `/f`

Il punto è trattare il `basePath` come se non fosse presente e tutte le risorse API dichiarate vengono servite in relazione all'host. Questo può essere utile, ad esempio, quando hai un nome di dominio personalizzato con una mappatura API che non include un Percorso di base e un valore della Fase che si riferisce alla fase di produzione.

> **Note**
>
> API Gateway creerà automaticamente una risorsa root, anche se non è stato dichiarato in modo esplicito nel file di definizione.

In assenza di specifiche, `basePath` seleziona `ignore` per impostazione predefinita.

Metti come prefisso

Se nel file OpenAPI il valore di `basePath` è `/a/b/c` e la proprietà `paths` contiene `/e` e `/f`, la richiesta `POST` o `PUT` seguente:

```
POST /restapis?mode=import&basepath=prepend
```

```
PUT /restapis/api_id?basepath=prepend
```

porterà le seguenti risorse nell'API:

- `/`
- `/a`
- `/a/b`
- `/a/b/c`
- `/a/b/c/e`
- `/a/b/c/f`

L'effetto è trattare il `basePath` come se specificasse risorse aggiuntive (senza metodo) e aggiungerle al set di risorse dichiarato. Questo può essere utile, ad esempio, quando team diversi sono responsabili di parti diverse di un'API e il `basePath` potrebbe riferirsi al percorso della parte dell'API di ogni team.

> **Note**
>
> API Gateway creerà automaticamente le risorse intermedie, anche se non sono dichiarate in modo esplicito nella definizione.

Split

Se nel file OpenAPI il valore di `basePath` è `/a/b/c` e la proprietà `paths` contiene `/e` e `/f`, la richiesta `POST` o `PUT` seguente:

```
POST /restapis?mode=import&basepath=split
```

```
PUT /restapis/api_id?basepath=split
```

porterà le seguenti risorse nell'API:

- `/`
- `/b`
- `/b/c`
- `/b/c/e`
- `/b/c/f`

L'effetto è trattare la parte più alta del percorso, `/a`, come l'inizio del percorso di ogni risorsa e creare risorse aggiuntive (senza metodo) all'interno dell'API stessa. Questo potrebbe, ad esempio, essere utilizzato quando `a` è il nome di una fase che vuoi esporre come parte dell'API.

AWS variabili per l'importazione OpenAPI

È possibile utilizzare le seguenti AWS variabili nelle definizioni OpenAPI. API Gateway risolve le variabili quando l'API viene importata. Per specificare una variabile, utilizzare `${variable-name}`. La tabella seguente descrive le AWS variabili disponibili.

Nome della variabile	Description	
`AWS::AccountId`	L'ID AWS dell'account che importa l'API. Ad esempio, 123456789012.	
`AWS::Partition`	La AWS partizione in cui viene importata l'API. Per le AWS regioni standard, la partizione è. `aws`	
`AWS::Region`	La AWS regione in cui viene importata l'API. Ad esempio, `us-east-2` .	

AWS esempio di variabili

L'esempio seguente utilizza AWS le variabili per specificare una AWS Lambda funzione per un'integrazione.

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
```

Errori e avvisi relativi all'importazione dell'API in Gateway API

Quando si importa il file di definizione esterno in Gateway API è possibile che vengano generati avvisi ed errori. Nelle sezioni seguenti vengono descritti gli errori e gli avvisi che possono essere restituiti durante l'importazione.

Errori durante l'importazione

Nel corso dell'importazione, possono essere generati errori per problematiche rilevanti come nel caso di un documento OpenAPI non valido. Gli errori vengono restituiti come eccezioni (ad esempio, `BadRequestException`) in una risposta non andata a buon fine. Quando si verifica un errore, la nuova definizione dell'API viene scartata e all'API esistente non viene apportata alcuna modifica.

Avvisi durante l'importazione

Nel corso dell'importazione, gli avvisi possono essere generati per problematiche minori come nel caso di un riferimento mancante a un modello. In presenza di un avviso, l'operazione proseguirà se l'espressione di query `failonwarnings=false` viene aggiunta all'URL della richiesta. In caso contrario, gli aggiornamenti verranno ripristinati. Per impostazione predefinita, `failonwarnings` è impostato su `false`. In questi casi, gli avvisi vengono restituiti come campo nella RestApirisorsa risultante. In caso contrario, gli avvisi vengono restituiti come messaggio nell'eccezione.

Esportazione di un'API REST da API Gateway

Dopo aver creato e configurato un'API REST in API Gateway tramite la console API Gateway o in un altro modo, puoi esportarla in un file OpenAPI usando l'API di esportazione di API Gateway integrata nel servizio di controllo di Amazon API Gateway. Per utilizzare l'API di Gateway Amazon API, è necessario firmare le richieste API. Per ulteriori informazioni sulla firma delle richieste, consulta Signing AWS API requests nella IAM User Guide. Puoi scegliere di includere le estensioni di integrazione di API Gateway, nonché le estensioni Postman, nel file di definizione OpenAPI esportato.

> **Note**
>
> Quando esporti l'API utilizzando il AWS CLI, assicurati di includere il parametro extensions come mostrato nell'esempio seguente, per assicurarti che l'`x-amazon-apigateway-request-validator`estensione sia inclusa:
>
> ```
> aws apigateway get-export --parameters extensions='apigateway' --rest-api-id
>  abcdefg123 --stage-name dev --export-type swagger latestswagger2.json
> ```

Non puoi esportare un'API se i suoi payload non sono di tipo `application/json`. Se ci provi, verrà restituita una risposta di errore che indica che non è stato possibile trovare i modelli di corpo JSON.

Richiesta di esportazione di un'API REST

Con l'API di esportazione puoi esportare un'API REST esistente inviando una richiesta GET e specificando l'API da esportare come parte dei percorsi URL. L'URL della richiesta ha il formato seguente:

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
```

Puoi aggiungere la stringa di query `extensions` per specificare se includere le estensioni API Gateway (con il valore `integration`) o le estensioni Postman (con il valore `postman`).

Inoltre, puoi impostare l'intestazione `Accept` su `application/json` o `application/yaml` per ricevere l'output della definizione API rispettivamente in formato JSON o YAML.

Per ulteriori informazioni sull'invio di richieste GET utilizzando l'API API Gateway Export, vedere GetExport.

> **Note**
>
> Se definisci modelli nella tua API, devono essere per il tipo di contenuto "application/json" affinché API Gateway possa esportare il modello. In caso contrario, API Gateway genera un'eccezione con un messaggio di errore che indica che sono stati trovati solo modelli di corpo non JSON.
> I modelli devono contenere proprietà oppure possono essere definiti come tipo JSONSchema specifico.

Download della definizione OpenAPI dell'API REST in JSON

Per esportare e scaricare un'API REST nelle definizioni OpenAPI in formato JSON:

OpenAPI 3.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/oas30
Host: apigateway.<region>.amazonaws.com
Accept: application/json
```

OpenAPI 2.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/swagger
Host: apigateway.<region>.amazonaws.com
Accept: application/json
```

Qui *<region>* può essere, ad esempio, `us-east-1`. Per l'elenco di tutte le Regioni in cui Gateway API è disponibile, consulta Regions and Endpoints.

Download della definizione OpenAPI dell'API REST in YAML

Per esportare e scaricare un'API REST nelle definizioni OpenAPI in formato YAML:

OpenAPI 3.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/oas30
Host: apigateway.<region>.amazonaws.com
Accept: application/yaml
```

OpenAPI 2.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/swagger
Host: apigateway.<region>.amazonaws.com
Accept: application/yaml
```

Download della definizione OpenAPI dell'API REST con estensioni Postman in JSON

Per esportare e scaricare un'API REST nelle definizioni OpenAPI con Postman in formato JSON:

OpenAPI 3.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/oas30?extensions=postman
Host: apigateway.<region>.amazonaws.com
Accept: application/json
```

OpenAPI 2.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/swagger?extensions=postman
Host: apigateway.<region>.amazonaws.com
Accept: application/json
```

Download della definizione OpenAPI dell'API REST con l'integrazione di API Gateway in YAML

Per esportare e scaricare un'API REST nelle definizioni OpenAPI con l'integrazione di API Gateway in formato YAML:

OpenAPI 3.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/oas30?extensions=integrations
Host: apigateway.<region>.amazonaws.com
Accept: application/yaml
```

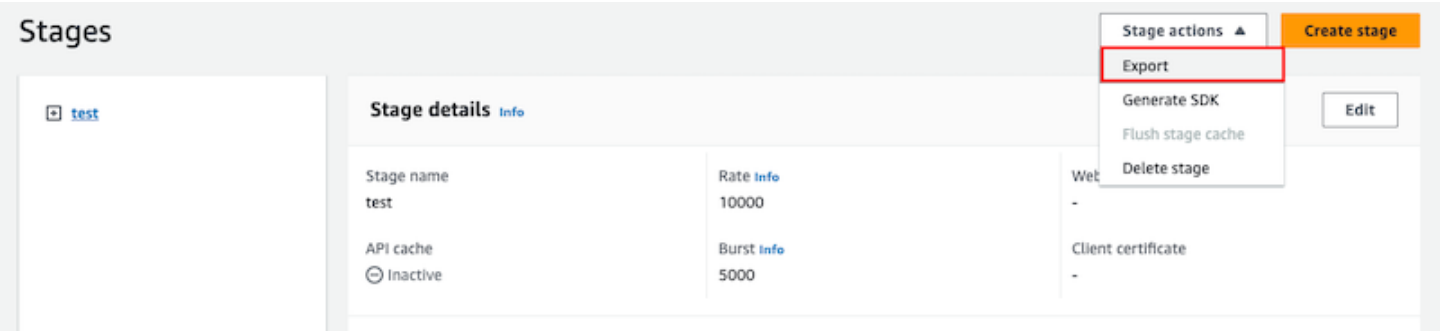
OpenAPI 2.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/swagger?
extensions=integrations
Host: apigateway.<region>.amazonaws.com
Accept: application/yaml
```

Esportazione di un'API REST tramite la console API Gateway

Dopo aver distribuito l'API REST in una fase, puoi procedere all'esportazione dell'API nella fase in un file OpenAPI usando la console API Gateway.

Nel riquadro Fasi della console Gateway API, scegli Operazioni fase, Esporta.



Specifica un valore in Tipo di specifica API, Formato ed Estensioni per scaricare la definizione OpenAPI dell'API.

Pubblica REST APIs per i clienti da richiamare

La semplice creazione e sviluppo di un'API di API Gateway non la rende automaticamente richiamabile dagli utenti. Per renderla richiamabile, è necessario distribuire l'API in una fase. Inoltre, potrebbe essere necessario personalizzare l'URL che verrà utilizzato dagli utenti per accedere all'API. Puoi assegnarli un dominio che sia coerente con il tuo marchio o che sia più facile da ricordare dell'URL predefinito dell'API.

In questa sezione viene descritto come distribuire l'API e personalizzare l'URL fornito agli utenti per accedervi.

> **Note**
>
> Per aumentare la sicurezza del tuo API Gateway APIs, il `execute-api.{region}.amazonaws.com` dominio è registrato nella Public Suffix List (PSL). Per una maggiore sicurezza, ti consigliamo di utilizzare i cookie con un `__Host-` prefisso se hai bisogno di impostare cookie sensibili nel nome di dominio predefinito per il tuo API Gateway APIs. Questa pratica ti aiuterà a difendere il tuo dominio dai tentativi CSRF (cross-site request forgery). Per ulteriori informazioni, consulta la pagina Impostazione cookie nella pagina Mozilla Developer Network.

Argomenti

- Implementazione di REST API in Gateway API
- Nome di dominio personalizzato per REST pubblico APIs in API Gateway

Implementazione di REST API in Gateway API

Dopo aver creato l'API, è necessario distribuirla per renderla chiamabile dagli utenti.

Per distribuire un'API, crea una distribuzione API e associala a una fase. Una fase è un riferimento logico a uno stato del ciclo di vita dell'API (ad esempio, `dev`, `prod`, `beta`, `v2`). Le fasi API sono identificate dall'ID API e dal nome della fase. Sono incluse nell'URL utilizzato per richiamare l'API. Ogni fase è un riferimento con nome a una distribuzione dell'API e viene resa disponibile per le applicazioni client da chiamare.

> **Important**
>
> Ogni volta che si aggiorna un'API, è necessario distribuire nuovamente l'API in una fase esistente o in una nuova fase. L'aggiornamento di un'API include la modifica di instradamenti, metodi, integrazioni, sistemi di autorizzazione, policy delle risorse e qualsiasi altra cosa diversa dalle impostazioni di fase.

Con l'evoluzione dell'API, puoi continuare a distribuirla in fasi diverse come versioni differenti. Puoi inoltre distribuire gli aggiornamenti API come una distribuzione di una release Canary. Ciò consente ai client API di accedere, sulla stessa fase, alla versione di produzione tramite la release e alla versione aggiornata tramite la release Canary.

Per chiamare un'API distribuita, il client invia una richiesta all'URL di un'API. L'URL è determinato dal protocollo (HTTP(S) o (WSS)), nome host, nome fase e (per le API REST) dal percorso delle risorse di un'API. Il nome host e il nome della fase determinano l'URL di base dell'API.

Se, ad esempio, si utilizza il nome di dominio predefinito dell'API, il formato dell'URL di base di un'API REST (ad esempio) in una fase specifica (*{stageName}*) è il seguente:

```
https://{restapi-id}.execute-api.{region}.amazonaws.com/{stageName}
```

Per rendere più intuitivo l'URL di base predefinito dell'API, puoi creare un nome di dominio personalizzato (ad esempio, `api.example.com`) per sostituire il nome host predefinito dell'API. Per supportare più API con il nome di dominio personalizzato, è necessario mappare una fase API a un percorso di base.

Con il nome di dominio personalizzato *{api.example.com}* e la fase API mappata al percorso di base (*{basePath}*) nel nome di dominio personalizzato, l'URL di base di un'API REST diventa:

```
https://{api.example.com}/{basePath}
```

Per ogni fase puoi ottimizzare le prestazioni dell'API impostando i limiti di throttling predefiniti delle richieste a livello di account e abilitando il caching dell'API. Puoi inoltre abilitare la registrazione delle chiamate API in CloudTrail o CloudWatch e selezionare un certificato client per permettere al back-end di autenticare le richieste API. Inoltre, puoi ignorare le impostazioni a livello di fase per i singoli metodi e definire le variabili di fase per il passaggio di contesti di ambiente specifici della fase all'integrazione API al runtime.

Le fasi permettono un efficace controllo delle versioni dell'API. Ad esempio, puoi distribuire un'API in una fase `test` e una fase `prod` e utilizzare la fase `test` come build di test e la fase `prod` come build stabile. Dopo che gli aggiornamenti hanno superato il test, puoi promuovere la fase `test` alla fase `prod`. La promozione può essere eseguita implementando nuovamente l'API nella fase `prod` o aggiornando il valore di una variabile di fase dal nome `test` al nome `prod`.

In questa sezione viene illustrato come distribuire un'API utilizzando la console API Gateway o chiamando l'API REST di API Gateway. Per utilizzare altri strumenti, consulta la documentazione della AWS CLI o di un SDK AWS.

Argomenti

- Creazione di un'implementazione per una REST API in Gateway API
- Configurazione di una fase per una REST API in Gateway API
- Configurare la distribuzione di una release Canary di API Gateway
- Aggiornamenti a REST APIs che richiedono una ridistribuzione

Creazione di un'implementazione per una REST API in Gateway API

In API Gateway una distribuzione di API REST è rappresentata da una risorsa Distribuzione. È simile a un eseguibile di un'API rappresentato da una risorsa RestApi.

Per consentire al client di chiamare l'API, è necessario creare una distribuzione e associarvi una fase. Una fase è rappresentata da una risorsa Fase. Rappresenta una snapshot dell'API, inclusi metodi, integrazioni, modelli, modelli di mappatura e autorizzazioni Lambda (in precedenza note come autorizzazioni ad hoc). Quando si aggiorna l'API, è possibile ridistribuirla associando una nuova fase alla distribuzione esistente. La procedura di creazione di una fase è illustrata in the section called “Configurare una fase”.

Argomenti

- Crea distribuzione
- Fasi successive per l'implementazione di un'API

Crea distribuzione

Le procedure seguenti mostrano come creare l'implementazione di una REST API.

Console di gestione AWS

Per poter distribuire un'API REST per la prima volta, è necessario crearla. Per ulteriori informazioni, consulta Sviluppo di REST API in Gateway API.

La console API Gateway consente di distribuire un'API creando una distribuzione e associandola a una fase nuova o esistente.

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel riquadro di navigazione APIs (API) scegliere l'API che si desidera distribuire.
3. Nel riquadro Resources (Risorse) scegliere Deploy API (Distribuisci API).
4. In Fase, procedi come segue:
   a. Per creare una nuova fase, seleziona Nuova fase, quindi immetti un nome in Nome fase. Facoltativamente, puoi immettere una descrizione dell'implementazione in Descrizione distribuzione.
   b. Per scegliere una fase esistente, seleziona il nome della fase nel menu a discesa. Se lo desideri, puoi specificare una descrizione della nuova implementazione in Descrizione distribuzione.
   c. Per creare un'implementazione non associata a una fase, seleziona Nessuna fase. Successivamente, puoi associare questa implementazione a una fase.
5. Seleziona Deploy (Implementa).

AWS CLI

Quando si crea una distribuzione, viene creata un'istanza della risorsa Distribuzione.

Il comando create-deployment seguente crea una nuova implementazione:

```
aws apigateway create-deployment --rest-api-id rest-api-id
```

Non è possibile chiamare l'API fino a quando non si associa questa implementazione a una fase. Con una fase esistente, è possibile eseguire questa operazione aggiornando la proprietà deploymentId della fase con l'ID di implementazione appena creato. Il comando update-stage seguente aggiorna la fase con una nuova implementazione. Nella console, questa azione è chiamata Implementazione attiva.

```
aws apigateway update-stage \
```

```
    --rest-api-id rest-api-id \
    --stage-name 'stage-name' \
    --patch-operations op='replace',path='/deploymentId',value='deployment-id'
```

Quando si crea l'implementazione, è possibile contemporaneamente associarla a una nuova fase. Il comando create-deployment seguente crea una nuova implementazione e la associa a una nuova fase chiamata `beta`:

```
aws apigateway create-deployment \
   --rest-api-id rest-api-id \
   --stage-name beta
```

Per implementare nuovamente un'API, esegui la stessa procedura. È possibile riutilizzare la stessa fase.

Fasi successive per l'implementazione di un'API

Di seguito sono riportati i passaggi successivi per l'implementazione dell'API.

Modifica delle impostazioni di fase

Dopo che un'API è stata distribuita, puoi modificare le impostazioni delle fasi per abilitare o disabilitare la cache, la registrazione o il throttling delle richieste dell'API. Puoi inoltre scegliere un certificato client per consentire al back-end di autenticare API Gateway e impostare le variabili delle fasi per passare il contesto della distribuzione all'integrazione dell'API al runtime. Per ulteriori informazioni, consulta Modifica delle impostazioni di fase

Dopo avere modificato le impostazioni della fase, per renderle effettive sarà necessario eseguire di nuovo la ridistribuzione dell'API.

> **Note**
>
> Se le impostazioni aggiornate, come l'abilitazione della registrazione, richiedono un nuovo ruolo IAM, puoi aggiungere il ruolo IAM richiesto senza ridistribuire l'API. Tuttavia potrebbero essere necessari alcuni minuti prima che il nuovo ruolo IAM diventi effettivo. Fino ad allora, le tracce delle chiamate API non vengono registrate, anche se hai abilitato l'opzione di registrazione.

Scelta di diverse combinazioni di fase-implementazione

Dal momento che una distribuzione rappresenta una snapshot dell'API e una fase definisce un percorso in una snapshot, puoi scegliere combinazioni diverse di distribuzione-fase per controllare il modo in cui gli utenti effettuano chiamate nelle diverse versioni dell'API. Ciò risulta utile se ad esempio desideri eseguire il rollback a uno stato precedente della distribuzione dell'API o unire una branca privata dell'API in quella pubblica.

La procedura seguente illustra come eseguire questa operazione tramite Stage Editor (Editor fasi) nella console API Gateway. Si presume che un'API sia stata distribuita più di una volta.

1. Se non sei già nel riquadro Fasi, nel pannello di navigazione principale, scegli Fasi.
2. Seleziona la fase da aggiornare.
3. Nella scheda Cronologia delle distribuzioni seleziona l'implementazione da utilizzare per la fase.
4. Scegli Cambia implementazione attiva.
5. Conferma di voler cambiare l'implementazione attiva e scegli Cambia implementazione attiva nella finestra di dialogo Rendi attiva l'implementazione.

Passare i dati specifici dell'implementazione all'API.

Per una distribuzione, puoi impostare o modificare le variabili delle fasi per passare i dati specifici della distribuzione all'integrazione dell'API in fase di runtime. Puoi eseguire questa operazione nella scheda Stage Variables (Variabili di fase) in Stage Editor (Editor fasi). Per ulteriori informazioni, consulta le istruzioni in Utilizzo delle variabili di fase per una REST API in Gateway API.

Configurazione di una fase per una REST API in Gateway API

Una fase è un riferimento con nome a un'implementazione, corrispondente a uno snapshot dell'API. Utilizzare una Stage (Fase) per gestire e ottimizzare una specifica distribuzione. Ad esempio, puoi configurare le impostazioni di fase per abilitare la memorizzazione nella cache, personalizzare il throttling della richiesta, configurare la registrazione, definire le variabili di fase o collegare una release Canary a scopi di test. La sezione seguente illustra come creare e configurare la fase.

Creazione di una nuova fase

Dopo la distribuzione iniziale, puoi aggiungere altre fasi e associarle alle distribuzioni esistenti. Puoi usare la console API Gateway per creare una nuova fase oppure puoi scegliere una fase esistente durante la distribuzione di un'API. In generale, puoi aggiungere una nuova fase a una distribuzione API prima di ridistribuirla. Per creare una nuova fase mediante la console Gateway API procedi come riportato di seguito:

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere una REST API.
3. Nel riquadro di navigazione principale scegli Fasi sotto un'API.
4. Dal riquadro di navigazione Fasi scegli Crea fase.
5. Per Nome fase immetti un nome, ad esempio **`prod`**.

   > **Note**
   >
   > I nomi di fasi possono contenere solo caratteri alfanumerici, trattini e caratteri di sottolineatura. La lunghezza massima è 128 caratteri.

6. (Facoltativo). In Descrizione inserisci una breve descrizione.
7. In Implementazione seleziona la data e l'ora dell'implementazione API esistente che intendi associare a questa fase.
8. In Impostazioni aggiuntive puoi specificare le impostazioni aggiuntive per la fase.
9. Scegli Crea fase.

Modifica delle impostazioni di fase

Dopo una distribuzione riuscita di un'API, la fase viene popolata di impostazioni predefinite. Per modificare le impostazioni di una fase, incluso caching e registrazione delle API, puoi utilizzare la console o l'API REST di API Gateway. Se è stato modificato l'endpoint predefinito della REST API, quando si aggiorna una fase, la modifica viene propagata a tutte le fasi dell'API. La procedura seguente illustra come effettuare questa operazione tramite l'editor della fase della console Gateway API.

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere una REST API.

3. Nel riquadro di navigazione principale scegli Fasi sotto un'API.
4. Nel riquadro Stages (Fasi), selezionare il nome della fase.
5. Nella sezione Dettagli fase scegli Modifica.
6. (Facoltativo) In Descrizione fase modifica la descrizione.
7. In Impostazioni aggiuntive modifica le seguenti impostazioni:

Impostazioni cache

Per abilitare il caching delle API per la fase, attiva Cache API con provisioning. Quindi configura la memorizzazione nella cache a livello di metodo predefinita, la capacità della cache, la crittografia dei dati della cache, la cache time-to-live (TTL) e tutti i requisiti per l'invalidazione della cache per chiave.

Il caching non è attivo finché non viene attivato il caching predefinito a livello di metodo o il caching a livello di metodo per un metodo specifico.

Per ulteriori informazioni sulle impostazioni della cache, consulta Impostazioni della cache per REST API in Gateway API.

> **Note**
>
> Se abiliti la memorizzazione nella cache delle API per una fase dell'API, al tuo account potrebbe essere addebitato un costo per la memorizzazione nella cache delle API. AWS Il caching non è idoneo per il AWS piano gratuito.

Impostazioni di limitazione (della larghezza di banda della rete)

Per impostare le destinazioni di limitazione (della larghezza di banda della rete) a livello di fase per tutti i metodi associati a questa API, attiva Throttling.

Per Rate(Tasso), inserire un tasso di destinazione. Questa è la velocità, espressa in richieste al secondo, con cui i token vengono aggiunti al bucket di token. La velocità a livello di fase non deve essere superiore alla velocità a livello di account come specificato in Quote per la configurazione e l'esecuzione di una REST API in Gateway API.

Per Burst (ottimizzazione), inserisci un tasso di destinazione. La frequenza di burst è la capacità del token bucket. Ciò consente di passare più richieste per un periodo di tempo

rispetto al tasso di destinazione. Questo tasso di ottimizzazione a livello di fase non deve essere superiore al tasso di ottimizzazione a livello di account come specificato in Quote per la configurazione e l'esecuzione di una REST API in Gateway API.

> **Note**
>
> I tassi di limitazione (della larghezza di banda della rete) non sono limiti rigidi e vengono applicati sulla base del miglior tentativo. In alcuni casi, i client possono superare gli obiettivi impostati. Non fare affidamento sulla limitazione (della larghezza di banda della rete) per controllare i costi o bloccare l'accesso a un'API. Considera l'utilizzo di Budget AWS per monitorare i costi e di AWS WAF per gestire le richieste API.

Impostazioni di firewall e certificati

Per associare un ACL AWS WAF Web allo stage, selezionate un ACL Web dall'elenco a discesa Web ACL. Se si desidera, scegliere Block API Request if WebACL cannot be evaluated (Blocca richiesta API se non è possibile valutare WebACL).

Per selezionare un certificato client per la fase, scegli un certificato dal menu a discesa Certificati client.

8. Scegli Continua.
9. Esamina le modifiche e scegli Salva modifiche.
10. Per abilitare Amazon CloudWatch Logs per tutti i metodi associati a questa fase di questa API API Gateway, nella sezione Logs and tracing, scegli Modifica.

> **Note**
>
> Per abilitare CloudWatch i log, devi anche specificare l'ARN di un ruolo IAM che consente ad API Gateway di scrivere informazioni nei log CloudWatch per conto del tuo utente. Per farlo, scegli Impostazioni dal pannello di navigazione APIsprincipale. Quindi, per il ruolo di CloudWatch registro, inserisci l'ARN di un ruolo IAM.
> Per scenari applicativi comuni, il ruolo IAM potrebbe allegare la policy gestita di Amazon APIGateway PushToCloudWatchLogs.
> Il ruolo IAM deve anche contenere la seguente dichiarazione di relazione di attendibilità:

JSON

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

Per ulteriori informazioni CloudWatch, consulta la Amazon CloudWatch User Guide.

11. Seleziona un livello di registrazione dal menu a discesa CloudWatch Logs. I livelli di registrazione dei log sono:

    - Inattivo: la registrazione dei log non è attivata per questa fase.
    - Solo errori: la registrazione dei log è abilitata solo per gli errori.
    - Errori e log informativi: la registrazione dei log è abilitata per tutti gli eventi.

12. Seleziona Data tracing per fare in modo che API Gateway riferisca alla registrazione CloudWatch della traccia dei dati per la tua fase. Questo può essere utile per la risoluzione dei problemi APIs, ma può comportare la registrazione di dati sensibili.

> **Note**
>
> Ti consigliamo di non utilizzare il tracciamento dei dati per la produzione. APIs

13. Seleziona Metriche dettagliate per fare in modo che API Gateway CloudWatch riferisca alle metriche API di`API calls`,`Latency`, `Integration latency`400 errors, e. `500 errors` Per ulteriori informazioni CloudWatch, consulta il monitoraggio di base e il monitoraggio dettagliato nella Amazon CloudWatch User Guide.

> **Important**
>
> Al tuo account viene addebitato l'accesso ai parametri a livello di metodo, ma non ai CloudWatch parametri a livello di API o a livello di fase.

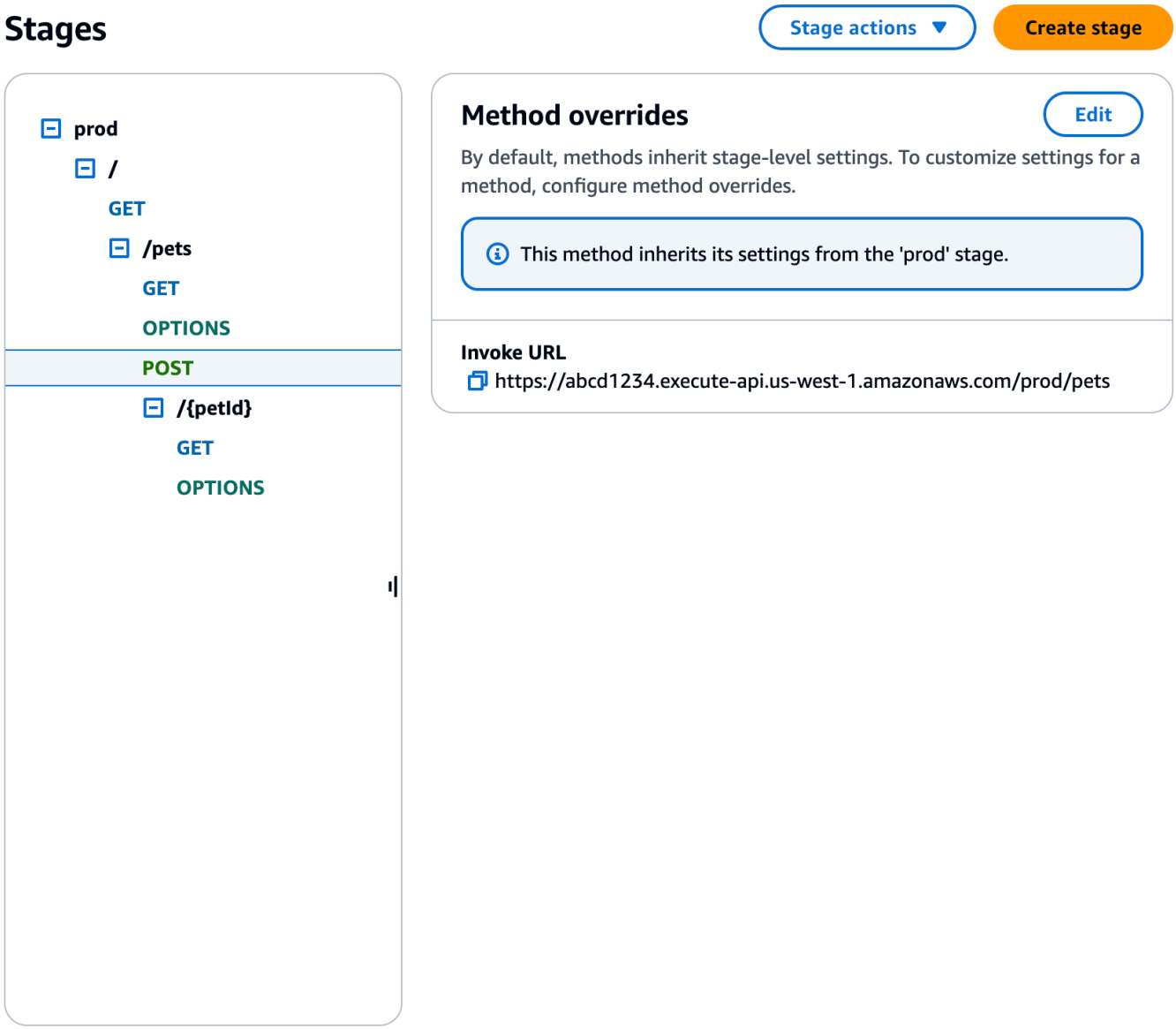
14. Per abilitare la registrazione degli accessi a una destinazione, attiva Registrazione accesso personalizzato.
15. In ARN di destinazione del log degli accessi inserisci l'ARN di un gruppo di log o un flusso Firehose.

    Il formato dell'ARN per Firehose è `arn:aws:firehose:{region}:{account-id}:deliverystream/amazon-apigateway-{your-stream-name}`. Il nome del flusso di Firehose deve essere `amazon-apigateway-{your-stream-name}`.
16. In Formato log immetti un formato di log. Per ulteriori informazioni sui formati di log di esempio, consulta the section called “Formati di log di accesso”.
17. Per abilitare il tracciamento AWS X-Ray per la fase API, seleziona Tracciamento X-Ray. Per ulteriori informazioni, consulta Traccia delle richieste degli utenti alle REST API utilizzando X-Ray in Gateway API.
18. Scegli Salva modifiche. Implementa nuovamente l'API per rendere effettive le nuove impostazioni.

Sostituzione delle impostazioni a livello di fase

Dopo aver personalizzato le impostazioni a livello di fase, è possibile sostituirle per ogni metodo API. Alcune di queste opzioni potrebbero comportare costi aggiuntivi per il tuo Account AWS.

1. Per configurare le sostituzioni dei metodi, espandi la fase nel pannello di navigazione secondario, quindi scegli un metodo.



2. Per Sostituzioni del metodo scegli Modifica.
3. Per attivare CloudWatch le impostazioni a livello di metodo, per CloudWatch Registri, seleziona un livello di registrazione.
4. Per attivare la registrazione dei log della traccia dei dati per il metodo, seleziona Tracciamento dei dati.

> **Note**
>
> Si consiglia di non utilizzare il tracciamento dei dati per la produzione. APIs

5. Per attivare i parametri dettagliati a livello di metodo, seleziona Parametri dettagliati. Al tuo account viene addebitato l'accesso alle metriche a livello di metodo, ma non alle CloudWatch metriche a livello di API o a livello di fase.
6. Per attivare la limitazione (della larghezza di banda della rete) a livello di metodo, seleziona Throttling. Immetti le opzioni appropriate a livello di metodo. Per ulteriori informazioni sulla limitazione, consulta the section called “Throttling”.
7. Per configurare la cache a livello di metodo, seleziona Abilita cache metodo. La modifica dell'impostazione del caching predefinito a livello di metodo nei dettagli della fase non influisce su questa impostazione.
8. Scegli Save (Salva).

Aggiungi un'API API Gateway REST come destinazione per Amazon Bedrock AgentCore Gateway

Un Amazon Bedrock AgentCore Gateway offre agli sviluppatori di agenti AI un modo sicuro per esporre il tuo API Gateway REST APIs come strumenti compatibili con il Model Context Protocol (MCP). AgentCore Gateway utilizza obiettivi per definire gli strumenti. Quando aggiungi la tua fase come destinazione, il tuo Gateway diventa un singolo URL MCP che consente l'accesso agli strumenti per un agente. Per ulteriori informazioni, consulta le fasi dell'API REST di API Gateway come obiettivi nella Amazon Bedrock AgentCore Gateway Developer Guide.

Gli obiettivi API Gateway collegano il AgentCore gateway alle fasi del REST APIs. Puoi includere l'intera fase come obiettivo o selezionare risorse. Dopo aver creato l'oggetto API Gateway, AgentCore Gateway traduce le richieste MCP in entrata in richieste HTTP e gestisce la formattazione della risposta. I client MCP possono recuperare la documentazione dell'API utilizzando il metodo e richiamarla utilizzando il `tools/list` metodo. APIs `tools/call`

Considerazioni

Le seguenti considerazioni potrebbero influire sull'utilizzo dell'aggiunta di una fase come destinazione a un Gateway: AgentCore

- È necessario disporre già di un AgentCore Gateway.
- Sono supportati solo APIs i REST pubblici.
- L'endpoint predefinito dell'API non può essere disabilitato.
- Per ogni metodo dell'API deve essere definito un nome di operazione oppure è necessario creare un nome sostitutivo quando si aggiunge lo stage come destinazione. Questo nome viene utilizzato come nome dello strumento utilizzato dagli agenti per interagire con il metodo.

- È possibile utilizzare `API_KEY` tipi di provider di `GATEWAY_IAM_ROLE` credenziali per Outbound Auth per consentire al gateway di accedere all'API. `NO_AUTH` Il provider di `API_KEY` credenziali è definito da Gateway. AgentCore Puoi utilizzare la tua chiave API Gateway esistente. Per ulteriori informazioni, consulta Configurazione dell'autenticazione in uscita.
- Se utilizzi un pool di utenti Amazon Cognito o un autorizzatore Lambda per controllare l'accesso alla tua API, i client MCP non possono accedervi.
- L'API deve trovarsi nello stesso account e nella stessa regione del gateway. AgentCore

Aggiungi una fase di un'API come destinazione per un AgentCore gateway

La procedura seguente mostra come aggiungere una fase di un'API come destinazione per un AgentCore Gateway.

Per aggiungere una fase di un'API come destinazione per un AgentCore Gateway

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli un'API REST distribuita in una fase.
3. Nel riquadro di navigazione principale scegli Fasi.
4. Scegli Stage actions, quindi scegli Create MCP target.
5. Per AgentCore Gateway, selezionate un AgentCore Gateway.
6. Per Nome destinazione, inserisci un nome di destinazione.
7. Per Descrizione dell'oggetto, inserisci una descrizione.
8. Conserva l'API e lo stage forniti.
9. Per le risorse Select API, seleziona le risorse della tua API a cui possono accedere gli agenti che utilizzano il tuo AgentCore Gateway.

   Se non selezioni una risorsa, un agente non può visualizzare la documentazione o richiamare l'endpoint.
10. La combinazione della risorsa e del metodo sono le operazioni dello strumento. Se l'operazione non ha un nome, crea un nome override.

    È inoltre possibile definire un nome di operazione per un metodo al momento della creazione.
11. Per la configurazione di autenticazione in uscita, scegli IAM Role, Nessuna autorizzazione o API key.
12. Seleziona Crea destinazione.

Per visualizzare tutti i AgentCore gateway che hanno accesso al tuo APIs, scegli la sezione MCP targets nel pannello di navigazione principale. In questa sezione, puoi creare un target MCP per qualsiasi API nella tua regione distribuita in una fase. Scegli Crea obiettivo MCP e segui i passaggi precedenti.

Puoi anche visualizzare gli strumenti disponibili per il tuo target e modificarlo nella console AgentCore Gateway. Per ulteriori informazioni, consulta Aggiungere obiettivi a un AgentCore gateway esistente.

Eliminazione di una fase

Quando una fase non è più necessaria, puoi eliminarla per evitare di pagare per risorse inutilizzate. Le fasi seguenti mostrano come utilizzare la console API Gateway per eliminare una fase.

> **Warning**
>
> L'eliminazione di una fase potrebbe rendere inutilizzabile parte o tutta l'API corrispondente per i chiamanti dell'API. L'eliminazione di una fase non può essere annullata, tuttavia è possibile creare nuovamente una fase e associarla alla stessa distribuzione.

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere una REST API.
3. Nel riquadro di navigazione principale scegli Fasi.
4. Nel riquadro Fasi scegli la fase che vuoi eliminare, quindi seleziona Operazioni fase, Elimina fase.
5. Quando richiesto, immetti **`confirm`**, quindi scegli Elimina.

Configurazione dei tag per una fase API in Gateway API

In API Gateway è possibile aggiungere un tag a una fase API, rimuoverlo o visualizzarlo. A tale scopo è possibile utilizzare la console API Gateway, la AWS CLI/l'SDK o l'API REST di API Gateway.

Una fase può anche ereditare i tag dalla sua API REST padre. Per ulteriori informazioni, consulta the section called “Eredità dei tag nell'API di Amazon API Gateway V1”.

Per ulteriori informazioni sull'assegnazione di tag alle risorse dell'API Gateway, consulta Applicazione di tag.

Argomenti

- Configurare i tag per una fase API utilizzando la console si API Gateway
- Configurazione dei tag per una fase API usando AWS CLI
- Impostare i tag per una fase API utilizzando l'API REST di API Gateway

Configurare i tag per una fase API utilizzando la console si API Gateway

La procedura seguente descrive come configurare i tag per una fase API.

Per configurare i tag per una fase API mediante la console API Gateway

1. Accedere alla console API Gateway.
2. Seleziona un'API esistente o crea una nuova API che includa risorse, metodi e le integrazioni corrispondenti.
3. Seleziona una fase o distribuisci l'API in una nuova fase.
4. Nel riquadro di navigazione principale scegli Fasi.
5. Seleziona la scheda Tags (Tag). Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
6. Scegliere Gestisci tag.
7. In Editor di tag scegli Aggiungi nuovo tag. Immetti una chiave di tag (ad esempio, `Department`) nella colonna Key (Chiave), quindi immetti un valore di tag (ad esempio, `Sales`) nella colonna Value (Valore). Per salvare il tag scegli Salva.
8. Se necessario, ripeti la fase 5 per aggiungere altri tag alla fase API. Il numero massimo di tag per ogni fase è 50.
9. Per rimuovere un tag esistente dalla fase scegli Rimuovi.
10. Se l'API è stata implementata in precedenza nella console API Gateway, sarà necessario ridistribuirla per rendere effettive le modifiche.

Configurazione dei tag per una fase API usando AWS CLI

È possibile impostare i tag per una fase API usando AWS CLI con il comando create-stage o il comando tag-resource. È possibile eliminare uno o più tag da una fase API eseguendo il comando untag-resource.

Il comando create-stage seguente aggiunge un tag durante la creazione di una fase `test`:

```
aws apigateway create-stage --rest-api-id abc1234 --stage-name test --description
 'Testing stage' --deployment-id efg456 --tag Department=Sales
```

Il comando tag-resource seguente aggiunge un tag a una fase `prod`:

```
aws apigateway tag-resource --resource-arn arn:aws:apigateway:us-east-2::/
restapis/abc123/stages/prod --tags Department=Sales
```

Il comando untag-resource seguente rimuove il tag `Department=Sales` dalla fase `test`:

```
aws apigateway untag-resource --resource-arn arn:aws:apigateway:us-east-2::/
restapis/abc123/stages/test --tag-keys Department
```

Impostare i tag per una fase API utilizzando l'API REST di API Gateway

È possibile configurare i tag per una fase API utilizzando l'API REST di API Gateway con una delle operazioni seguenti:

- Richiamare `tags:tag` per aggiungere tag a una fase API.
- Richiamare `tags:untag` per eliminare uno o più tag da una fase API.
- Richiamare `stage:create` per aggiungere uno o più tag a una fase API in fase di creazione.

È anche possibile richiamare `tags:get` per descrivere i tag in una fase API.

Aggiunta di tag a una fase API

Dopo avere distribuito un'API (`m5zr3vnks7`) in una fase (`test`), è possibile aggiungere tag alla fase richiamando `tags:tag`. L'Amazon Resource Name (ARN) richiesto della fase (`arn:aws:apigateway:us-east-1::/restapis/m5zr3vnks7/stages/test`) deve avere codifica 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"
  }
```

```
}
```

Puoi anche usare la richiesta precedente per aggiornare un tag esistente in un nuovo valore.

È possibile aggiungere tag a una fase richiamando `stage:create` per la relativa creazione:

```
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"
  }
}
```

Rimozione di tag da una fase API

Per rimuovere il tag `Department` dalla fase, richiamare `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: ...
```

Per rimuovere più di un tag, usare un elenco di chiavi di tag separate da virgole nell'espressione di query, ad esempio `?tagKeys=Department,Division,…`.

Descrizione dei tag per una fase API

Per descrivere i tag esistenti per una fase specifica, richiamare `tags:get`:

```
GET /tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis%2Fm5zr3vnks7%2Fstages
%2Ftags
```

```
Host: apigateway.us-east-1.amazonaws.com
Authorization: ...
```

La risposta con esito positivo è simile a quella riportata di seguito.

```
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"
    }
}
```

Utilizzo delle variabili di fase per una REST API in Gateway API

Le variabili di fase sono coppie chiave-valore che è possibile definire come attributi di configurazione associati a una fase di implementazione di una REST API. Fungono da variabili di ambiente e possono essere utilizzate nei modelli di mappatura e configurazione dell'API. Con le fasi di distribuzione in API Gateway, puoi gestire più fasi di rilascio per ciascuna API e utilizzare le variabili di fase per configurare una fase di distribuzione dell'API per interagire con diversi endpoint di backend.

Le variabili di fase non sono destinate ad essere utilizzate per i dati sensibili, come le credenziali. Per passare dati sensibili alle integrazioni, utilizza un autorizzatore. AWS Lambda È possibile passare dati sensibili alle integrazioni nell'output del provider di autorizzazioni Lambda. Per ulteriori

informazioni, consulta the section called “Output da un sistema di autorizzazione Lambda di Gateway API”.

Casi d'uso per le variabili di fase

Di seguito sono riportati alcuni casi d'uso per le variabili di fase.

Specificare un endpoint di backend diverso

L'API può passare una richiesta `GET` come proxy HTTP all'host web di backend. Puoi utilizzare una variabile di fase per far sì che, quando i chiamanti dell'API invocano l'endpoint di produzione, Gateway API chiami `example.com`. Quindi, quando i chiamanti dell'API invocano la fase beta, Gateway API chiama un host web diverso, ad esempio `beta.example.com`. Analogamente, le variabili di fase si possono utilizzare per specificare un nome di funzione AWS Lambda per ogni fase dell'API. Non puoi utilizzare una variabile di fase per impostare un endpoint di integrazione diverso, ad esempio per far sì che la richiesta `GET` punti a un'integrazione di proxy HTTP in una fase e a un'integrazione di proxy Lambda in un'altra fase.

Quando si specifica un nome di funzione Lambda come valore della variabile di fase, è necessario configurare manualmente le autorizzazioni per la funzione Lambda. Quando specifichi una funzione Lambda nella console API Gateway, verrà visualizzato un AWS CLI comando per configurare le autorizzazioni appropriate. Puoi anche usare il seguente AWS CLI comando per farlo.

```
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
```

Passare le informazioni utilizzando modelli di mappatura

Puoi accedere alle variabili di fase nei modelli di mappatura o passare i parametri di configurazione a AWS Lambda o al backend HTTP. Ad esempio, potrebbe essere necessario riutilizzare la stessa funzione Lambda per più fasi nell'API, ma la funzione deve leggere i dati da una tabella di Amazon DynamoDB diversa a seconda della fase. Nei modelli di mappatura che generano la richiesta per la funzione Lambda è possibile usare le variabili di fase per passare il nome della tabella a Lambda.

Per utilizzare una variabile di fase, è necessario prima configurare una variabile di fase e quindi assegnarle un valore. Ad esempio, per personalizzare l'endpoint di integrazione HTTP, crea prima la variabile di fase `url` e poi, nella richiesta di integrazione dell'API, inserisci il valore della variabile di fase **`http://${stageVariables.url}`**. Questo valore indica ad API Gateway di sostituire la variabile di fase `${}` al runtime, a seconda della fase di esecuzione dell'API. Per ulteriori informazioni, consulta the section called “Configurazione delle variabili di fase per le REST API in Gateway API”.

Configurazione delle variabili di fase per le REST API in Gateway API

Questa sezione mostra come configurare le variabili di fase per due fasi di implementazione di un'API di esempio utilizzando la console Gateway Amazon API. Per capire come utilizzare le variabili di fase in Gateway API, ti consigliamo di seguire tutte le procedure illustrate in questa sezione.

Prerequisiti

Prima di iniziare, verifica che siano soddisfatti i seguenti requisiti preliminari:

- Devi disporre di un'API in API Gateway. Segui le istruzioni in Sviluppo di REST API in Gateway API.
- Devi avere distribuito l'API almeno una volta. Segui le istruzioni in Implementazione di REST API in Gateway API.
- Devi aver creato la prima fase per un'API distribuita. Segui le istruzioni in Creazione di una nuova fase.

Invocazione di un endpoint HTTP mediante un'API con una variabile di fase

Questa procedura descrive come creare una variabile di fase per un endpoint HTTP e due fasi per l'API. Inoltre, vengono create le variabili di fase `url`, `stageName` e `function`, che vengono utilizzate nelle procedure seguenti di questa sezione.

Per invocare un endpoint HTTP mediante un'API con una variabile di fase

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Crea un'API, quindi crea un metodo `GET` nella risorsa radice dell'API. Imposta il tipo di integrazione su HTTP e imposta l'URL dell'endpoint su **`http://${stageVariables.url}`**.
3. Implementa l'API in una nuova fase denominata **`beta`**.
4. Nel riquadro di navigazione scegli Fasi, quindi seleziona la fase beta.

5. Nella scheda Variabili di fase scegli Modifica.
6. Scegli Aggiungi variabile di fase.
7. In Nome, inserisci **`url`**. In Valore, inserisci **`httpbin.org/get`**.
8. Scegli Aggiungi variabile di fase, quindi effettua le seguenti operazioni:

   In Nome, inserisci **`stageName`**. In Valore, inserisci **`beta`**.
9. Scegli Aggiungi variabile di fase, quindi effettua le seguenti operazioni:

   In Nome, inserisci **`function`**. In Valore, inserisci **`HelloWorld`**.
10. Selezionare Salva.
11. Ora crea una seconda fase. Dal riquadro di navigazione Fasi scegli Crea fase. In Stage name (Nome fase) immettere **`prod`**. Seleziona un'implementazione recente da Implementazione e scegli Crea.
12. Come per la fase beta, imposta le stesse tre variabili di fase (url, version e function) su valori diversi (rispettivamente **`petstore-demo-endpoint.execute-api.com/petstore/pets`**, **`prod`** e **`HelloEveryone`**).
13. Nel riquadro di navigazione Stages (Fasi) scegli beta. In Dettagli fase scegli l'icona Copia per copiare l'URL di richiamo dell'API, quindi immetti l'URL di richiamo dell'API in un browser Web. Viene avviata la richiesta `GET` della fase beta nella risorsa radice dell'API.

   > **Note**
   >
   > Il collegamento Invoke URL (URL chiamata) punta alla risorsa radice dell'API nella rispettiva fase beta. L'immissione dell'URL in un browser Web chiama il metodo `GET` della fase beta nella risorsa radice. Se i metodi vengono definiti nelle risorse figlio e non nella risorsa radice stessa, scegliendo l'URL in un browser Web viene restituita la risposta di errore `{"message":"Missing Authentication Token"}`. In questo caso, devi aggiungere al collegamento Invoke URL (URL chiamata) il nome di una risorsa figlio specifica.
14. La risposta che si ottiene dalla richiesta `GET` della fase beta è mostrata più avanti. Puoi anche verificare il risultato utilizzando un browser su cui accedere http://httpbin.org/get. Questo valore è stato assegnato alla variabile `url` nella fase beta. Le due risposte sono identiche.
15. Nel riquadro di navigazione Stages (Fasi) scegli la fase prod. In Dettagli fase scegli l'icona Copia per copiare l'URL di richiamo dell'API, quindi immetti l'URL di richiamo dell'API in un browser Web. Viene avviata la richiesta `GET` della fase prod nella risorsa radice dell'API.

16. La risposta che si ottiene dalla richiesta `GET` della fase prod è mostrata più avanti. È possibile verificare il risultato utilizzando un browser verso il quale accedere http://petstore-demo-endpoint.execute-api.com/petstore/pets. Questo valore è stato assegnato alla variabile `url` nella fase prod. Le due risposte sono identiche.

Passare i metadati specifici della fase in un backend HTTP

Questa procedura descrive come utilizzare un valore di variabile di fase in un'espressione di parametri di query per passare i metadati specifici delle fasi in un back-end HTTP. Utilizzeremo la variabile di fase `stageName` dichiarata nella procedura precedente.

Per passare metadati specifici della fase in un backend HTTP

1. Nel riquadro di navigazione Resource (Risorsa) scegli il metodo GET.

   Per aggiungere un parametro della stringa di query all'URL del metodo, seleziona la scheda Richiesta del metodo, quindi nella sezione Impostazioni della richiesta del metodo, scegli Modifica.
2. Scegli i parametri della stringa di query URL ed effettua le seguenti operazioni:

   a. Scegliere Add query string (Aggiungi stringa di query).

   b. In Nome, inserisci **`stageName`**.

   c. Mantieni Obbligatorio e Caching disattivati.
3. Selezionare Salva.
4. Scegli la scheda Richiesta di integrazione, quindi seleziona Modifica nella sezione Impostazioni della richiesta di integrazione.
5. Per URL dell'endpoint aggiungi **`?stageName=${stageVariables.stageName}`** al valore URL definito in precedenza, in modo che sia l'intero URL dell'endpoint sia **`http://${stageVariables.url}?stageName=${stageVariables.stageName}`**.
6. Scegli Implementa API e seleziona la fase beta.
7. Nel riquadro di navigazione principale scegli Fasi. Nel riquadro di navigazione Stages (Fasi) scegli beta. In Dettagli fase scegli l'icona Copia per copiare l'URL di richiamo dell'API, quindi immetti l'URL di richiamo dell'API in un browser Web.

> **Note**
>
> Qui utilizziamo la fase beta perché l'endpoint HTTP (come specificato dalla `url` variabile "http://httpbin.org/get«) accetta le espressioni dei parametri di query e le restituisce come `args` oggetto nella sua risposta.

8. Si ottiene la risposta seguente. `beta`, assegnato alla variabile di fase `stageName`, viene passata nel back-end come argomento `stageName`.

```
{
  "args": {
    "stageName": "beta"
  },
  "headers": {
    "Accept": "application/json",
    "Host": "httpbin.org",
    "User-Agent": "AmazonAPIGateway_abcd1234",
    "X-Amzn-Apigateway-Api-Id": "abcd1234",
    "X-Amzn-Trace-Id": "Self=1-abcd-11111111111111;Root=1-11111111-111111111111111"
  },
  "origin": "192.0.2.9",
  "url": "http://httpbin.org/get?stageName=beta"
}
```

Invocare una funzione Lambda attraverso un'API con una variabile di fase

Questa procedura descrive come utilizzare una variabile di fase per chiamare una funzione Lambda come back-end dell'API. Utilizzerai la variabile di fase `function` dichiarata in Invocazione di un endpoint HTTP mediante un'API con una variabile di fase.

Quando imposti una funzione Lambda come valore di una variabile di fase, usa il nome locale della funzione, se possibile includendo l'alias o la specifica della versione, come in **`HelloWorld`**, **`HelloWorld:1`** o **`HelloWorld:alpha`**. Non usare l'ARN della funzione (ad esempi, **`arn:aws:lambda:us-east-1:123456789012:function:HelloWorld`**). La console API Gateway assume il valore della variabile di fase per una funzione Lambda come nome di funzione non qualificato ed espande la variabile di fase specificata in un ARN.

Per invocare la funzione Lambda mediante un'API con una variabile di fase

1. Crea una funzione Lambda denominata **`HelloWorld`** utilizzando il runtime predefinito Node.js . Il codice deve contenere quanto segue:

```
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...');
};
```

Per ulteriori informazioni su come creare una funzione Lambda, consulta Nozioni di base sulla console REST API.

2. Nel riquadro Risorse seleziona Crea risorsa, quindi procedi come segue:

   a. Per Percorso risorsa, seleziona /.

   b. Per Resource Name (Nome risorsa) immetti **`lambdav1`**.

   c. Scegli Crea risorsa.

3. Scegli la risorsa /lambdav1 e poi scegli Crea metodo.

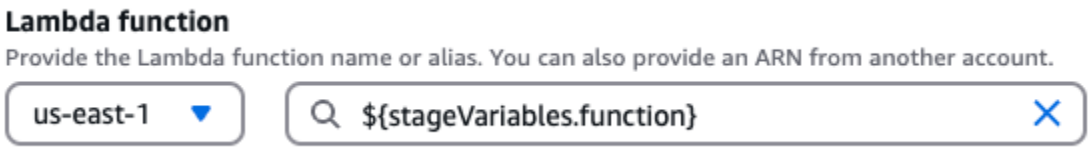
   Successivamente, esegui queste operazioni:

   a. Per Tipo di metodo seleziona GET.

   b. Per Tipo di integrazione seleziona Funzione Lambda.

   c. Mantieni l'opzione Integrazione proxy Lambda disattivata.

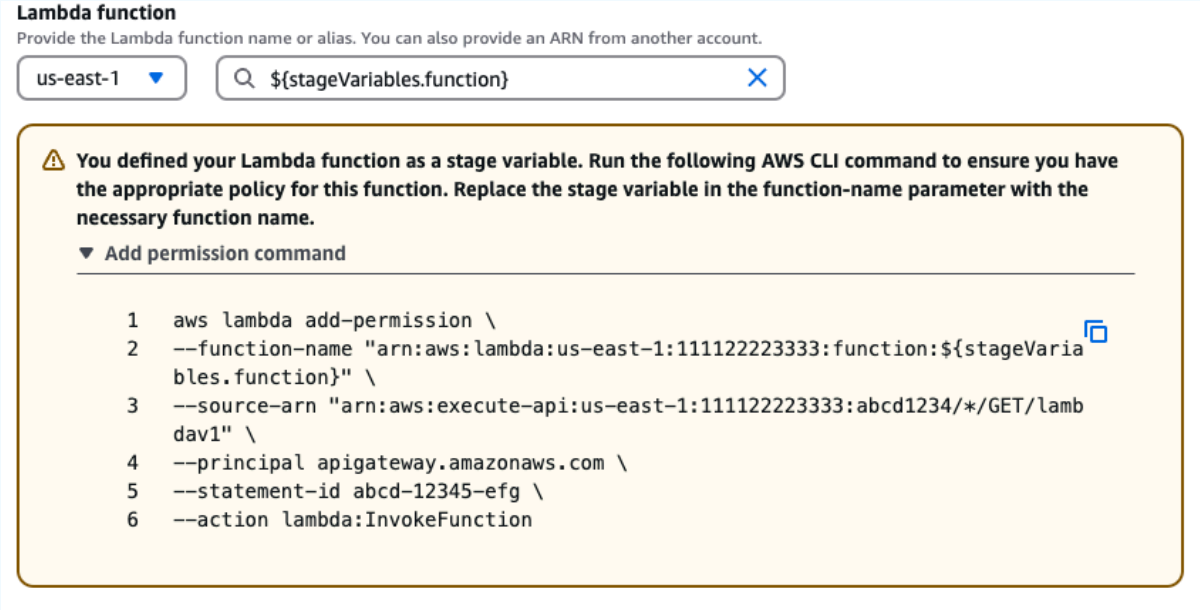
   d. Per Lambda function (Funzione Lambda), immetti `${stageVariables.function}`.



> **Tip**
>
> Quando richiesto da Aggiungi comando di autorizzazione, copia il comando add-permission. Esegui il comando per ciascuna funzione Lambda che sarà assegnata alla variabile di fase `function`. Ad esempio, se il valore di `$stageVariables.function` è `HelloWorld`, esegui il comando AWS CLI seguente:

```
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
```

In caso contrario, riceverai una risposta `500 Internal Server Error` quando verrà invocato il metodo. Sostituisci `${stageVariables.function}` con il nome della funzione Lambda assegnato alla variabile di fase.



e. Scegli Crea metodo.

4. Implementa l'API in entrambe le fasi prod e beta.
5. Nel riquadro di navigazione principale scegli Fasi. Nel riquadro di navigazione Stages (Fasi) scegli beta. In Dettagli fase scegli l'icona Copia per copiare l'URL di richiamo dell'API, quindi immetti l'URL di richiamo dell'API in un browser Web. Aggiungi **`/lambdav1`** all'URL prima di premere invio.

   Si ottiene la risposta seguente.

```
"Hello, World! I'm not sure where I'm calling from..."
```

Passaggio dei metadati specifici delle fasi a una funzione Lambda mediante una variabile di fase

Questa procedura descrive come utilizzare una variabile di fase per passare i metadata di configurazione specifici delle fasi in una funzione Lambda. Crei un metodo `POST` e un modello di mappatura di input per generare il payload utilizzando la variabile di fase `stageName` dichiarata precedentemente.

Per passare i metadati specifici delle fasi a una funzione Lambda mediante una variabile di fase

1. Scegli la risorsa /lambdav1 e poi scegli Crea metodo.

   Successivamente, esegui queste operazioni:

   a. Per Tipo di metodo seleziona POST.

   b. Per Tipo di integrazione seleziona Funzione Lambda.

   c. Mantieni l'opzione Integrazione proxy Lambda disattivata.

   d. Per Lambda function (Funzione Lambda), immetti `${stageVariables.function}`.

   e. Quando richiesto da Aggiungi comando di autorizzazione, copia il comando add-permission. Esegui il comando per ciascuna funzione Lambda che sarà assegnata alla variabile di fase `function`.

   f. Scegli Crea metodo.
2. Scegli la scheda Richiesta di integrazione, quindi seleziona Modifica nella sezione Impostazioni della richiesta di integrazione.
3. Scegli Modelli di mappatura, quindi seleziona Aggiungi modello di mappatura.
4. Per Tipo di contenuto inserisci **`application/json`**.
5. Per Corpo del modello inserisci il seguente modello:

```
#set($inputRoot = $input.path('$'))
{
    "stageName" : "$stageVariables.stageName"
}
```

> **Note**
>
> In un modello di mappatura il riferimento a una variabile di fase deve essere racchiuso tra apici (come in `"$stageVariables.stageName"` o

"${stageVariables.stageName}"), mentre altrove non si devono utilizzare gli apici (come in ${stageVariables.function}).

6. Selezionare Salva.
7. Implementa l'API in entrambe le fasi beta e prod.
8. Per utilizzare un client REST API per passare i metadati specifici della fase, procedi come segue:

   a. Nel riquadro di navigazione Stages (Fasi) scegli beta. In Dettagli fase scegli l'icona Copia per copiare l'URL di richiamo dell'API, quindi immetti l'URL di richiamo dell'API nel campo di input di un client REST API. Aggiungi **`/lambdav1`** prima di inviare la richiesta.

      Si ottiene la risposta seguente.

      ```
      "Hello, World! I'm calling from the beta stage."
      ```

   b. Nel riquadro di navigazione Fasi scegli prod. In Dettagli fase scegli l'icona Copia per copiare l'URL di richiamo dell'API, quindi immetti l'URL di richiamo dell'API nel campo di input di un client REST API. Aggiungi **`/lambdav1`** prima di inviare la richiesta.

      Si ottiene la risposta seguente.

      ```
      "Hello, World! I'm calling from the prod stage."
      ```

9. Per utilizzare la funzionalità Test per trasmettere i metadati specifici della fase, procedi come segue:

   a. Nel riquadro di navigazione Risorse scegli la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda.
   b. Per function immetti **`HelloWorld`**.
   c. Per stageName immetti **`beta`**.
   d. Scegli Test (Esegui test). Non è necessario aggiungere un corpo alla richiesta `POST`.

      Si ottiene la risposta seguente.

      ```
      "Hello, World! I'm calling from the beta stage."
      ```

   e. Puoi ripetere i passaggi precedenti per testare la fase Prod. Per stageName immetti **`Prod`**.

      Si ottiene la risposta seguente.

```
"Hello, World! I'm calling from the prod stage."
```

Riferimento alle variabili di fase di Gateway API per le REST API in Gateway API

È possibile usare le variabili di fase di API Gateway nei casi seguenti.

Espressioni di mappatura dei parametri

Una variabile di fase può essere utilizzata in un'espressione di mappatura dei parametri per un parametro di intestazione di risposta o di richiesta del metodo API, senza alcuna sostituzione parziale. Nell'esempio che segue si fa riferimento alla variabile di fase senza il simbolo $ e il simbolo `{...}` di chiusura.

- `stageVariables.<variable_name>`

Modelli di mappatura

Una variabile di fase può essere utilizzata ovunque in un modello di mappatura, come mostrato negli esempi seguenti.

- `{ "name" : "$stageVariables.<variable_name>"}`
- `{ "name" : "${stageVariables.<variable_name>}"}`

URI di integrazione HTTP

Una variabile di fase può essere utilizzata come parte di un URL di integrazione HTTP, come mostrato negli esempi seguenti:

- Un URI completo senza protocoll – `http://${stageVariables.<variable_name>}`
- Un dominio completo – `http://${stageVariables.<variable_name>}/resource/operation`
- Un sottodominio – `http://${stageVariables.<variable_name>}.example.com/resource/operation`
- Un percorso – `http://example.com/${stageVariables.<variable_name>}/bar`
- Una stringa di query – `http://example.com/foo?q=${stageVariables.<variable_name>}`

AWS URI di integrazione

Una variabile di fase può essere utilizzata come parte dell'azione o dei componenti del percorso dell' AWS URI, come mostrato nell'esempio seguente.

- `arn:aws:apigateway:<region>:<service>:${stageVariables.<variable_name>}`

AWS URI di integrazione (funzioni Lambda)

Una variabile di fase può essere utilizzata al posto del nome di una funzione Lambda o version/alias, come mostrato negli esempi seguenti.

- `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**
>
> Per utilizzare una variabile di fase per una funzione Lambda, la funzione deve essere nello stesso account dell'API. Le variabili di fase non supportano le funzioni Lambda tra più account.

Bacino d’utenza di Amazon Cognito

Una variabile di fase può essere utilizzata al posto di un pool di utenti di Amazon Cognito per un sistema di autorizzazione `COGNITO_USER_POOLS`.

- `arn:aws:cognito-idp:<region>:<account_id>:userpool/${stageVariables.<variable_name>}`

AWS credenziali di integrazione

Una variabile di fase può essere utilizzata come parte del AWS user/role credenziale ARN, come mostrato nell'esempio seguente.

- `arn:aws:iam::<account_id>:${stageVariables.<variable_name>}`

Configurare la distribuzione di una release Canary di API Gateway

Con release Canary si intende una strategia di sviluppo software in cui una nuova versione di un'API o di un altro software viene distribuita a scopo di test e contemporaneamente la versione di base rimane distribuita come release di produzione per le normali operazioni nella stessa fase. In questa documentazione la versione di base viene detta release di produzione. È possibile applicare una release Canary a qualsiasi versione non di produzione a scopo di test.

Nella distribuzione di una release Canary, il traffico API totale viene separato in modo casuale tra release di produzione e release Canary, in base a un rapporto preconfigurato. In genere, la release Canary riceve una piccola percentuale di traffico API e il resto viene assegnato alla release di produzione. Le caratteristiche API aggiornate sono visibili solo al traffico API nella release Canary. Puoi modificare la percentuale di traffico nella release Canary per ottimizzare le prestazioni o la copertura dei test.

Mantenendo limitato il traffico nella release Canary e la selezione casuale, la maggior parte degli utenti non riscontra conseguenze negative a causa di bug potenziali nella nuova versione e nessun utente riscontra conseguenze negative durature.

Se i parametri di test soddisfano i requisiti, puoi promuovere la release Canary a release di produzione e disabilitarne la distribuzione. Le nuove caratteristiche vengono così rese disponibili nella fase di produzione.

Argomenti

- Distribuzione di una release Canary in API Gateway
- Creazione di una distribuzione di una release Canary
- Aggiornamento di una release Canary
- Promozione di una release Canary
- Disabilitazione di una release Canary

Distribuzione di una release Canary in API Gateway

In API Gateway la distribuzione di una release Canary utilizza la fase di distribuzione per il rilascio di produzione di una versione di base di un'API e collega alla fase una release Canary per le versioni dell'API nuove rispetto alla versione di base. La fase è associata alla distribuzione iniziale e la

release Canary alle distribuzioni successive. All'inizio, sia la fase che la release Canary puntano alla stessa versione API. In questa sezione i termini fase e release di produzione vengono usati in modo intercambiabile, come pure i termini Canary e release Canary.

Per distribuire un'API con una release Canary devi creare una distribuzione della release Canary aggiungendo le impostazioni Canary alla fase di una distribuzione standard. Le impostazioni Canary descrivono la release Canary sottostante e la fase rappresenta la release di produzione dell'API nella distribuzione. Per aggiungere le impostazioni Canary, imposta `canarySettings` nella fase di distribuzione e specifica quanto segue:

- Un ID di distribuzione, inizialmente identico all'ID della distribuzione della versione di base impostato nella fase.
- Una percentuale di traffico API, con un valore compreso tra 0,0 e 100,0 inclusi, per la release Canary.
- Variabili di fase per la release Canary che possono sovrascrivere le variabili di fase per la release di produzione.
- L'uso della cache dello stage per le richieste Canary, se useStageCacheè impostata e la memorizzazione nella cache delle API è abilitata sullo stage.

Dopo l'abilitazione di una release Canary, la fase di distribuzione non può essere associata a un'altra distribuzione di una release non Canary fino a quando la release Canary non viene disabilitata e le impostazioni Canary non vengono rimosse dalla fase.

Quando si abilita il logging delle esecuzioni API, per la release Canary vengono generati log e parametri per tutte le richieste Canary. Vengono segnalate a un gruppo di log CloudWatch Logs in fase di produzione e a un gruppo di log Logs specifico per Canary CloudWatch . Lo stesso vale per il logging degli accessi. I log specifici della release Canary sono utili per convalidare le nuove modifiche dell'API e stabilire se accettare le modifiche e promuovere la release Canary alla fase di produzione o se eliminare le modifiche e annullare la release Canary nella fase di produzione.

Il gruppo di log delle esecuzioni della fase di produzione è denominato `API-Gateway-Execution-Logs/{rest-api-id}/{stage-name}` e il gruppo di log delle esecuzioni della release Canary è denominato `API-Gateway-Execution-Logs/{rest-api-id}/{stage-name}/Canary`. Per il logging degli accessi, è necessario creare un nuovo gruppo di log o sceglierne uno esistente. Al nome del gruppo di log degli accessi della release Canary scelto viene aggiunto il suffisso `/Canary`.

Una versione canary può utilizzare lo stage cache, se abilitato, per memorizzare le risposte e utilizzare le voci memorizzate nella cache per restituire i risultati alle successive richieste Canary, entro un periodo preconfigurato (TTL). time-to-live

Nella distribuzione di una release Canary, la release di produzione e la release Canary dell'API possono essere associate alla stessa versione o a versioni diverse. Quando sono associate a versioni diverse, le risposte per le richieste di produzione e Canary vengono memorizzate nella cache separatamente e la cache di fase restituisce i risultati corrispondenti per le richieste di produzione e Canary. Quando la release di produzione e la release Canary sono associate alla stessa distribuzione, la cache di fase usa una singola chiave cache per entrambi i tipi di richieste e restituisce la stessa risposta per le stesse richieste dalla release di produzione e dalla release Canary.

Creazione di una distribuzione di una release Canary

Puoi creare una distribuzione di una release Canary quando distribuisci l'API con impostazioni Canary come input aggiuntivo all'operazione di creazione della distribuzione.

Puoi anche creare una distribuzione di una release Canary da una distribuzione non Canary esistente effettuando una richiesta `stage:update` per aggiungere le impostazioni Canary nella fase.

Quando crei una distribuzione di una release non Canary, puoi specificare un nome di fase non esistente. Se la fase specificata non esiste, API Gateway la crea. Non puoi tuttavia specificare un nome di fase non esistente quando crei una distribuzione di una release Canary. In questo caso si verificherà un errore e API Gateway non creerà una distribuzione della release Canary.

Puoi creare una distribuzione Canary Release in API Gateway utilizzando la console API Gateway AWS CLI, o un AWS SDK.

Argomenti

- Creare una distribuzione Canary utilizzando la console API Gateway
- Crea una distribuzione Canary utilizzando il AWS CLI

Creare una distribuzione Canary utilizzando la console API Gateway

Per usare la console API Gateway per creare una distribuzione di una release Canary, segui queste istruzioni:

Per creare la distribuzione di una release Canary iniziale

1. Accedere alla console API Gateway.
2. Crea una nuova REST API o scegline una esistente.
3. Nel riquadro di navigazione principale scegli Risorse, quindi seleziona Distribuisci l'API. Seguire le istruzioni visualizzate sullo schermo in Deploy API (Distribuisci API) per distribuire l'API in una nuova fase.

   Al momento, l'API è stata distribuita in una fase della release di produzione. Configura quindi le impostazioni Canary nella fase e, se necessario, abilita il caching, imposta le variabili di fase o configura i log degli accessi o delle esecuzioni dell'API.
4. Per abilitare la memorizzazione nella cache delle API o associare un ACL AWS WAF web allo stage, nella sezione Dettagli dello stage, scegli Modifica. Per ulteriori informazioni, consulta the section called “Impostazioni cache” o the section called “Per associare un ACL AWS WAF Web a uno stadio API Gateway API utilizzando la console API Gateway”.
5. Per configurare la registrazione degli accessi o delle esecuzioni, scegli Modifica nella sezione Log e tracciamento e segui le istruzioni visualizzate sullo schermo. Per ulteriori informazioni, consulta Configurare la CloudWatch registrazione per le API REST in API Gateway.
6. Per impostare le variabili di fase, scegli la scheda Variabili di fase e segui le istruzioni visualizzate sullo schermo per aggiungere o modificare le variabili di fase. Per ulteriori informazioni, consulta the section called “Utilizzo delle variabili di fase”.
7. Scegli la scheda Canary, quindi seleziona Crea Canary. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda Canary.
8. In Impostazioni di Canary, inserisci in Canary la percentuale di richieste da deviare al Canary.
9. Se lo desideri, seleziona Cache della fase per attivare il caching per la release Canary. La cache non è disponibile per la release Canary fino a quando non viene abilitato il caching dell'API.
10. Per sovrascrivere le variabili di fase esistenti, inserisci in Sostituzione canary un nuovo valore per la variabile di fase.

Dopo l'inizializzazione della release Canary nella fase di distribuzione, puoi modificare l'API e testare le modifiche. Puoi ridistribuire l'API nella stessa fase in modo che sia la versione di base che quella aggiornata siano accessibili tramite la stessa fase. Di seguito viene descritto come fare.

Per distribuire la versione API più recente in una release Canary

1. Con ogni aggiornamento dell'API scegli Distribuisci l'API.

2. In Distribuisci l'API scegli la fase contenente un canary nell'elenco a discesa Fase della distribuzione.
3. (Facoltativo) Immetti una descrizione in Descrizione distribuzione.
4. Scegliere Deploy (Distribuisci) per inviare la versione API più recente alla release Canary.
5. Se lo desideri, riconfigura le impostazioni della fase, i log o le impostazioni Canary, come descritto in Per creare la distribuzione di una release Canary iniziale.

La release Canary punta ora alla versione più recente, mentre la release di produzione continua a puntare alla versione iniziale dell'API. Per canarySettings è ora presente un nuovo valore deploymentId, mentre la fase ha ancora il valore deploymentId iniziale. In background la console chiama stage:update.

Crea una distribuzione Canary utilizzando il AWS CLI

Per creare una distribuzione canary per una nuova fase

1. Utilizza il seguente comando create-deployment per creare un'implementazione con due variabili di fase, ma senza un canary:

```
aws apigateway create-deployment \
    --variables sv0=val0,sv1=val1 \
    --rest-api-id abcd1234 \
    --stage-name 'prod'
```

   L'output sarà simile al seguente:

```
{
    "id": "du4ot1",
    "createdDate": 1511379050
}
```

2. Utilizza il seguente comando create-deployment per creare una distribuzione canary per la fase `prod`:

```
aws apigateway create-deployment \
    --rest-api-id abcd1234 \
    --canary-settings
 percentTraffic=10.5,stageVariableOverrides={sv1='val2',sv2='val3'},useStageCache=false
 \
```

```
    --stage-name 'prod'
```

L'output sarà simile al seguente:

```
{
    "id": "a6rox0",
    "createdDate": 1511379433
}
```

Il valore `id` della distribuzione risultante identifica la versione di verifica dell'API per la release Canary. Di conseguenza, la fase associata è abilitata per la release Canary.

3. (Facoltativo) Utilizza il seguente comando get-stage per visualizzare la rappresentazione della fase:

```
aws apigateway get-stage --rest-api-id acbd1234 --stage-name prod
```

Di seguito è illustrata una rappresentazione dell'oggetto `Stage` come output del comando:

```
{
    "stageName": "prod",
    "variables": {
        "sv0": "val0",
        "sv1": "val1"
    },
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": "du4ot1",
    "lastUpdatedDate": 1511379433,
    "createdDate": 1511379050,
    "canarySettings": {
        "percentTraffic": 10.5,
        "deploymentId": "a6rox0",
        "useStageCache": false,
        "stageVariableOverrides": {
            "sv2": "val3",
            "sv1": "val2"
        }
    },
    "methodSettings": {}
}
```

In questo esempio la versione di base dell'API usa le variabili di fase `{"sv0":val0", "sv1":val1"}`, mentre la versione di test usa le variabili di fase `{"sv1":val2", "sv2":val3"}`. Sia la release di produzione che la release Canary usano la stessa variabile di fase `sv1`, ma con valori diversi, `val1` e `val2`, rispettivamente. La variabile di fase `sv0` viene usata unicamente nella release di produzione e la variabile di fase `sv2` viene usata unicamente nella release Canary.

È anche possibile creare una distribuzione di rilascio canary da un'implementazione standard esistente aggiornando la fase per abilitare un canary.

Per creare una distribuzione di rilascio canary da un'implementazione esistente

1. Utilizza il comando create-deployment per creare un'implementazione senza un canary:

```
aws apigateway create-deployment \
    --variables sv0=val0,sv1=val1 \
    --rest-api-id abcd1234 \
    --stage-name 'beta'
```

2. Utilizza il comando update-stage per aggiornare la fase e abilitare un canary:

```
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"
        }]'
```

L'output sarà simile al seguente:

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

Poiché è stato abilitato un canary in una versione esistente dell'API, sia il rilascio di produzione (`Stage`) sia il rilascio canary (`canarySettings`) puntano alla stessa implementazione. Dopo aver modificato l'API e averla distribuita di nuovo in questa fase, la nuova versione sarà nella release Canary, mentre la versione di base rimane nella release di produzione. Ciò si manifesta nell'evoluzione della fase quando il valore `deploymentId` nella release Canary viene aggiornato al valore `id` della nuova distribuzione e il valore `deploymentId` nella release di produzione rimane invariato.

Aggiornamento di una release Canary

Dopo la distribuzione di una release Canary, è possibile modificare la percentuale di traffico nella release Canary oppure abilitare o disabilitare l'uso di una cache di fase per ottimizzare le prestazioni di test. È anche possibile modificare le variabili di fase usate nella release Canary quando il contesto di esecuzione viene aggiornato. Per eseguire tali aggiornamenti, chiama l'operazione stage:update con i nuovi valori in canarySettings.

Puoi aggiornare una versione Canary utilizzando la console API Gateway, il comando AWS CLI update-stage o un SDK. AWS

Argomenti

- Aggiornare una release Canary utilizzando la console API Gateway
- Aggiorna una versione Canary utilizzando il AWS CLI

Aggiornare una release Canary utilizzando la console API Gateway

Per usare la console API Gateway per aggiornare le impostazioni Canary esistenti in una fase, esegui queste operazioni:

Per aggiornare le impostazioni di Canary esistenti

1. Accedi alla console Gateway API e scegli una REST API esistente.
2. Nel riquadro di navigazione principale scegli Fasi, quindi seleziona una fase esistente.
3. Seleziona la scheda Canary, quindi scegli Modifica. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda Canary.
4. Aggiorna il campo Distribuzione richiesta aumentando o diminuendo il valore percentuale scegliendo un numero compreso tra 0,0 e 100,0 inclusi.
5. Seleziona o deseleziona la casella di controllo Cache della fase.
6. Aggiungi, rimuovi o modifica le Variabili di fase di Canary.
7. Scegli Save (Salva).

Aggiorna una versione Canary utilizzando il AWS CLI

Per usare AWS CLI per aggiornare un canary, usa il `update-stage`comando e modifica l'operazione di patch per ogni parametro del canary.

Il comando update-stage seguente aggiorna se il canary utilizza la cache della fase:

```
aws apigateway update-stage \
    --rest-api-id {rest-api-id} \
    --stage-name '{stage-name}' \
    --patch-operations op=replace,path=/canarySettings/useStageCache,value=true
```

Il comando update-stage seguente aggiorna la percentuale di traffico del canary:

```
aws apigateway update-stage \
    --rest-api-id {rest-api-id} \
    --stage-name '{stage-name}' \
    --patch-operations op=replace,path=/canarySettings/percentTraffic,value=25.0
```

Il comando update-stage seguente aggiorna le variabili di fase. L'esempio mostra come creare una nuova variabile di fase denominata `newVar`, sostituire la variabile di fase `var2` e rimuovere la variabile di fase `var1`:

```
aws apigateway update-stage  \
    --rest-api-id {rest-api-id} \
    --stage-name '{stage-name}'  \
    --patch-operations '[{
        "op": "replace",
        "path": "/canarySettings/stageVariableOverrides/newVar",
        "value": "newVal"
      }, {
        "op": "replace",
        "path": "/canarySettings/stageVariableOverrides/var2",
        "value": "val4"
      }, {
        "op": "remove",
        "path": "/canarySettings/stageVariableOverrides/var1"
      }]'
```

È possibile aggiornare tutti gli elementi precedenti combinando le operazioni in un singolo valore `patch-operations`:

```
aws apigateway update-stage  \
    --rest-api-id {rest-api-id} \
    --stage-name '{stage-name}' \
    --patch-operations '[{
        "op": "replace",
        "path": "/canarySettings/percentTraffic",
        "value": "20.0"
    }, {
        "op": "replace",
        "path": "/canarySettings/useStageCache",
        "value": "true"
    }, {
        "op": "remove",
        "path": "/canarySettings/stageVariableOverrides/var1"
```

```
    }, {
        "op": "replace",
        "path": "/canarySettings/stageVariableOverrides/newVar",
        "value": "newVal"
    }, {
        "op": "replace",
        "path": "/canarySettings/stageVariableOverrides/val2",
        "value": "val4"
      }]'
```

Promozione di una release Canary

Quando si promuove una release canary, questa sostituisce le impostazioni correnti della fase. La promozione di una release Canary non comporta la disabilitazione della release Canary nella fase. Per disabilitare una release Canary, è necessario rimuovere le impostazioni Canary nella fase. Per promuovere una release canary, procedi come descritto di seguito.

- Reimpostazione dell'ID distribuzione della fase con le impostazioni dell'ID distribuzione della release Canary. Questa operazione comporta l'aggiornamento dello snapshot API della fase con lo snapshot della release Canary e l'impostazione della versione di test come release di produzione.
- Aggiornamento delle variabili di fase con le variabili di fase della release Canary, se presenti. Questa operazione comporta l'aggiornamento del contesto di esecuzione dell'API della fase con quello della release Canary. Senza questo aggiornamento, la nuova versione API può produrre risultati imprevisti se la versione di test usa variabili di fase diverse o valori diversi delle variabili di fase esistenti.
- Impostazione della percentuale del traffico della release Canary su 0,0%.

Argomenti

- Promuovere una release Canary utilizzando la console API Gateway
- Promuovi una versione canaria usando il AWS CLI

Promuovere una release Canary utilizzando la console API Gateway

Per usare la console API Gateway per promuovere una distribuzione di una release Canary, esegui queste operazioni:

Per promuovere l'implementazione di una release Canary

1. Accedi alla console API Gateway e seleziona un'API esistente nel riquadro di navigazione principale.
2. Nel riquadro di navigazione principale scegli Fasi, quindi seleziona una fase esistente.
3. Scegli la scheda Canary.
4. Scegli Promuovi Canary.
5. Verifica le modifiche da apportare e scegli Promuovi Canary.

Dopo la promozione, la release di produzione fa riferimento alla stessa versione API (deploymentId) della release Canary. È possibile verificarlo utilizzando il. AWS CLI Per un esempio, consulta the section called “Promuovi una versione canaria usando il AWS CLI”.

Promuovi una versione canaria usando il AWS CLI

Per promuovere una versione canary alla versione di produzione utilizzando AWS CLI i comandi, chiamate il `update-stage` comando per copiare il file canary-associato `deploymentId` a quello associato allo stage`deploymentId`, per reimpostare la percentuale di traffico canarino a zero (`0.0`) e per copiare qualsiasi variabile dello stage canary-bound su quelle corrispondenti.

Supponiamo di avere una distribuzione di rilascio di Canary, descritta da una fase simile alla seguente:

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

Utilizza il seguente comando update-stage per promuovere il canary:

```
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"
      }]'
```

L'output sarà simile al seguente:

```
{
    "_links": {
        ...
    },
    "accessLogSettings": {
        ...
    },
    "cacheClusterEnabled": false,
```

```
    "cacheClusterStatus": "NOT_AVAILABLE",
    "canarySettings": {
        "deploymentId": "eh1sby",
        "useStageCache": false,
        "stageVariableOverrides": {
            "sv2": "val3",
            "sv1": "val2"
        },
        "percentTraffic": 0
    },
    "createdDate": "2017-11-20T04:42:19Z",
    "deploymentId": "eh1sby",
    "lastUpdatedDate": "2017-11-22T05:29:47Z",
    "methodSettings": {
        ...
    },
    "stageName": "prod",
    "variables": {
        "sv2": "val3",
        "sv1": "val2"
    }
}
```

La promozione di una release canary nella fase non disabilita il canary e l'implementazione rimane un'implementazione di release canary. Per renderla una normale distribuzione di produzione, è necessario disabilitare le impostazioni Canary. Per ulteriori informazioni su come disabilitare una distribuzione di release Canary, consulta the section called “Disabilitazione di una release Canary”.

Disabilitazione di una release Canary

Per disabilitare l'implementazione di una release Canary, è necessario impostare `canarySettings` su null per rimuoverla dalla fase.

Puoi disabilitare una distribuzione Canary Release utilizzando la console API Gateway AWS CLI, o un AWS SDK.

Argomenti

- Disabilitazione di una release Canary utilizzando la console Gateway API
- Disattiva una versione canary usando il AWS CLI

Disabilitazione di una release Canary utilizzando la console Gateway API

Per usare la console Gateway API per disabilitare l'implementazione di una release Canary, procedi come segue:

Per disabilitare l'implementazione di una release Canary

1. Accedi alla console Gateway API e seleziona un'API esistente nel riquadro di navigazione principale.
2. Nel riquadro di navigazione principale scegli Fasi, quindi seleziona una fase esistente.
3. Scegli la scheda Canary.
4. Scegli Elimina.
5. Confermare che si desidera eliminare la release Canary scegliendo Delete (Elimina).

Di conseguenza, la proprietà `canarySettings` diventa `null` e viene rimossa dalla fase di distribuzione. Puoi verificarlo utilizzando. AWS CLI Per un esempio, consulta the section called “Disattiva una versione canary usando il AWS CLI”.

Disattiva una versione canary usando il AWS CLI

Il comando update-stage seguente disattiva la distribuzione di rilascio canary:

```
aws apigateway update-stage \
    --rest-api-id abcd1234 \
    --stage-name canary \
    --patch-operations '[{"op":"remove", "path":"/canarySettings"}]'
```

L'output sarà simile al seguente:

```
{
    "stageName": "prod",
    "accessLogSettings": {
        ...
    },
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": "nfcn0x",
    "lastUpdatedDate": 1511309280,
    "createdDate": 1511152939,
```

```
    "methodSettings": {
        ...
    }
}
```

Come illustrato nell'output, la proprietà canarySettings non è più presente nella fase della distribuzione di rilascio canary disabilitata.

Aggiornamenti a REST APIs che richiedono una ridistribuzione

Gestire un'API significa visualizzare, aggiornare ed eliminare le configurazioni API esistenti. Puoi gestire un'API utilizzando la console API Gateway, AWS CLI CloudFormation, un SDK o l'API REST API Gateway. L'aggiornamento di un'API implica la modifica di determinate proprietà delle risorse o delle impostazioni di configurazione dell'API. Gli aggiornamenti delle risorse richiedono una nuova implementazione dell'API, operazione non necessaria per gli aggiornamenti di configurazione.

La tabella seguente descrive le risorse API che richiedono una nuova implementazione dell'API quando vengono aggiornate.

Risorsa	Note
ApiKey	Per le proprietà applicabili e le operazioni supportate, consulta apikey:update. L'aggiornamento richiede la ridistribuzione dell'API.
Authorizer	Per le proprietà applicabili e le operazioni supportate, consulta authorizer:update. L'aggiornamento richiede la ridistribuzione dell'API.
disableExecuteApiEndpoint	L'aggiornamento richiede la modifica di una fase per l'API, ad esempio una nuova implementazione dell'API in una fase.
DocumentationPart	Per le proprietà applicabili e le operazioni supportate, consulta documentationpart:update. L'aggiornamento richiede la ridistribuzione dell'API.
DocumentationVersion	Per le proprietà applicabili e le operazioni supportate, consulta documentationversion:update. L'aggiornamento richiede la ridistribuzione dell'API.
GatewayResponse	Per le proprietà applicabili e le operazioni supportate, consulta gatewayresponse:update. L'aggiornamento richiede la ridistribuzione dell'API.

Risorsa	Note
Integration	Per le proprietà applicabili e le operazioni supportate, consulta integration:update. L'aggiornamento richiede la ridistribuzione dell'API.
IntegrationResponse	Per le proprietà applicabili e le operazioni supportate, consulta integrationresponse:update. L'aggiornamento richiede la ridistribuzione dell'API.
Metodo	Per le proprietà applicabili e le operazioni supportate, consulta method:update. L'aggiornamento richiede la ridistribuzione dell'API.
MethodResponse	Per le proprietà applicabili e le operazioni supportate, consulta methodresponse:update. L'aggiornamento richiede la ridistribuzione dell'API.
Modello	Per le proprietà applicabili e le operazioni supportate, consulta model:update. L'aggiornamento richiede la ridistribuzione dell'API.
RequestValidator	Per le proprietà applicabili e le operazioni supportate, consulta requestvalidator:update. L'aggiornamento richiede la ridistribuzione dell'API.
Risorsa	Per le proprietà applicabili e le operazioni supportate, consulta resource:update. L'aggiornamento richiede la ridistribuzione dell'API.
RestApi	Per le proprietà applicabili e le operazioni supportate, consulta restapi:update. L'aggiornamento richiede la ridistribuzione dell'API. È inclusa la modifica delle policy di risorse.
VpcLink	Per le proprietà applicabili e le operazioni supportate, consulta vpclink:update. L'aggiornamento richiede la ridistribuzione dell'API.

La tabella seguente descrive le configurazioni API che non richiedono una nuova implementazione dell'API quando vengono aggiornate.

Configurazione	Note
Account	Per le proprietà applicabili e le operazioni supportate, consulta account:update. L'aggiornamento non richiede la ridistribuzione dell'API.

Configurazione	Note
Distribuzione	Per le proprietà applicabili e le operazioni supportate, consulta deployment:update.
DomainName	Per le proprietà applicabili e le operazioni supportate, consulta domainname:update. L'aggiornamento non richiede la ridistribuzione dell'API.
BasePathMapping	Per le proprietà applicabili e le operazioni supportate, consulta basepathmapping:update. L'aggiornamento non richiede la ridistribuzione dell'API.
Tipo di indirizzo IP	L'aggiornamento non richiede la ridistribuzione dell'API.
Fase	Per le proprietà applicabili e le operazioni supportate, consulta stage:update. L'aggiornamento non richiede la ridistribuzione dell'API.
Utilizzo	Per le proprietà applicabili e le operazioni supportate, consulta usage:update. L'aggiornamento non richiede la ridistribuzione dell'API.
UsagePlan	Per le proprietà applicabili e le operazioni supportate, consulta usageplan:update. L'aggiornamento non richiede la ridistribuzione dell'API.

Nome di dominio personalizzato per REST pubblico APIs in API Gateway

I nomi di dominio personalizzati sono più semplici e intuitivi URLs che puoi fornire agli utenti delle tue API.

Dopo aver distribuito un'API, tu e i tuoi clienti potete richiamarla usando l'URL di base predefinito nel formato seguente:

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

dove *`api-id`* viene generato da API Gateway, *`region`* è la AWS regione e *`stage`* viene specificato dall'utente al momento della distribuzione dell'API.

La parte del nome host dell'URL, *`api-id`*`.execute-api.`*`region`*`.amazonaws.com`, fa riferimento a un endpoint API. Il nome dell'endpoint API predefinito viene generato casualmente, è complesso da richiamare e non è intuitivo.

Con i nomi di dominio personalizzati, è possibile impostare il nome host dell'API e scegliere un percorso base (ad esempio `myservice`) per mappare l'URL alternativo all'API. Ad esempio, un URL di base dell'API più intuitivo può diventare:

```
https://api.example.com/myservice
```

> **Note**
>
> Per informazioni sui nomi di dominio personalizzati per uso privato APIs, consultathe section called “Nomi di dominio personalizzati per API private”.

Considerazioni

Le seguenti considerazioni potrebbero influire sull’utilizzo di un nome di dominio personalizzato:

- È possibile disabilitare l'endpoint predefinito per un'API. I client possono comunque connettersi all'endpoint predefinito, ma riceveranno un codice di stato `403 Forbidden`.
- Un nome di dominio personalizzato regionale può essere associato a REST APIs e HTTP APIs. Puoi utilizzare API Gateway versione 2 APIs per creare e gestire nomi di dominio personalizzati regionali per REST APIs.
- Un nome di dominio personalizzato deve essere univoco all'interno di una regione per tutti gli AWS account.
- È possibile eseguire la migrazione di un nome di dominio personalizzato tra endpoint ottimizzati per l'edge e regionali, ma non la migrazione di un dominio personalizzato pubblico a un nome di dominio personalizzato privato.
- È necessario creare o aggiornare il record di risorse del provider DNS per eseguire la mappatura all'endpoint API. In assenza di questa mappatura, le richieste API destinate al nome di dominio personalizzato non possono raggiungere API Gateway.
- È possibile supportare un numero pressoché infinito di nomi di dominio senza superare la quota predefinita con un certificato jolly. Per ulteriori informazioni, consulta the section called “Nomi di dominio personalizzati con caratteri jolly”.
- Puoi scegliere una policy di sicurezza per il dominio personalizzato. Per ulteriori informazioni, consulta the section called “Scelta di una policy di sicurezza”.
- Per configurare le mappature API con più livelli, è necessario utilizzare un nome di dominio personalizzato regionale e la policy di sicurezza TLS 1.2.

Prerequisiti per i nomi di dominio personalizzati

Di seguito sono riportati i prerequisiti per la creazione di un nome di dominio personalizzato pubblico o privato. Per informazioni sui nomi di dominio personalizzati per uso privato APIs, consultathe section called “Nomi di dominio personalizzati per API private”.

Registrare un nome di dominio

È necessario disporre di un nome di dominio Internet registrato per configurare nomi di dominio personalizzati per il tuo APIs. Puoi registrare un nome di dominio Internet utilizzando Amazon Route 53 oppure un registrar di dominio di terze parti a tua scelta. Il nome di dominio personalizzato può essere il nome di un sottodominio o del dominio root (detto anche "apex di zona") di un dominio Internet registrato.

Il nome di dominio deve seguire la specifica RFC 1035 e può avere un massimo di 63 ottetti per etichetta e 255 ottetti in totale.

Impostazione del certificato per il nome di dominio personalizzato

Prima di configurare un nome di dominio personalizzato per un'API, devi disporre di un SSL/TLS certificato pronto in ACM. Se ACM non è disponibile nella AWS regione in cui stai creando il tuo nome di dominio personalizzato, devi importare un certificato in API Gateway in quella regione.

Per importare un SSL/TLS certificato, devi fornire l'ente del SSL/TLS certificato in formato PEM, la relativa chiave privata e la catena di certificati per il nome di dominio personalizzato.

Ogni certificato archiviato in ACM è identificato dal relativo ARN. Con i certificati emessi da ACM, non devi preoccuparti di esporre dettagli sensibili del certificato, come la chiave privata. Per utilizzare un certificato AWS gestito per un nome di dominio, è sufficiente fare riferimento al relativo ARN.

Se l'applicazione utilizza il blocco dei certificati, a volte noto come pinning SSL, per bloccare un certificato ACM, l'applicazione potrebbe non essere in grado di connettersi al dominio dopo il rinnovo del certificato. AWS Per ulteriori informazioni, consulta Probemi nel blocco dei certificati nella Guida per l'utente di AWS Certificate Manager .

Nomi di dominio personalizzati con caratteri jolly

Con i nomi di dominio personalizzati con caratteri jolly, è possibile supportare un numero quasi infinito di nomi di dominio senza superare la quota predefinita. Ad esempio, è possibile dare a ciascuno dei clienti il proprio nome di dominio, `customername.example.com`.

Per creare un nome di dominio personalizzato con caratteri jolly, specificare un carattere jolly (*) come primo sottodominio di un dominio personalizzato che rappresenta tutti i possibili sottodomini di un dominio radice.

Ad esempio, il nome di dominio personalizzato con caratteri jolly `*.example.com` restituisce sottodomini quali `a.example.com`, `b.example.com` e `c.example.com`. Quando si crea il nome di dominio personalizzato con caratteri jolly, tutti i relativi sottodomini vengono instradati dalla modalità di routing del nome di dominio con caratteri jolly. Per indirizzare sottodomini verso diversi APIs, puoi effettuare una delle seguenti operazioni:

- Utilizza le regole di routing per indirizzare le richieste in entrata verso destinazioni REST APIs diverse utilizzando `*.example.com` l'intestazione. `Host` Per ulteriori informazioni, consulta the section called “Esempio 4: regole di routing per i nomi di dominio con caratteri jolly”.
- Creare un nome di dominio per i sottodomini da instradare a un endpoint diverso. In un unico AWS account, puoi avere entrambi e. `*.example.com` `a.example.com`

È possibile utilizzare le variabili di contesto `$context.domainName` e `$context.domainPrefix` per determinare il nome di dominio utilizzato da un client per chiamare l'API. Per ulteriori informazioni sulle variabili di contesto, consulta Variabili per le trasformazioni dei dati per Gateway API.

Per creare un nome di dominio personalizzato con caratteri jolly, è necessario fornire un certificato emesso da ACM che sia stato convalidato utilizzando il DNS o il metodo di convalida della posta elettronica.

> **Note**
>
> Non puoi creare un nome di dominio personalizzato con caratteri jolly se un altro AWS account ha creato un nome di dominio personalizzato che è in conflitto con il nome di dominio personalizzato con caratteri jolly. Ad esempio, se l'account A ha creato `a.example.com`, l'account B non può creare il nome di dominio personalizzato con caratteri jolly `*.example.com`.
> Se l'account A e l'account B condividono un proprietario, è possibile contattare il Centro assistenza AWS per richiedere un'eccezione.

Passaggi successivi per nomi di dominio personalizzati

Di seguito sono riportati i passaggi successivi per i nomi di dominio personalizzati.

Fasi successive

- Per informazioni su come impostare il SSL/TLS certificato, consulta. the section called “Prepara i certificati in AWS Certificate Manager”
- Per informazioni su come creare un dominio personalizzato regionale, consulta the section called “Configurazione di un nome di dominio personalizzato regionale”.
- Per informazioni su come creare un dominio personalizzato ottimizzato per l'edge, consulta the section called “Configurazione di un nome di dominio personalizzato ottimizzato per l'edge in Gateway API”.
- Per informazioni su come eseguire la migrazione tra nomi di dominio personalizzati regionali e ottimizzati per l'edge, consulta the section called “Migrazione di nomi di dominio personalizzati”.
- Per informazioni su come connettere le fasi API a un nome di dominio personalizzato, consulta the section called “Invia traffico al tuo APIs tramite il tuo nome di dominio personalizzato in API Gateway”.
- Per informazioni su come scegliere una policy di sicurezza per il dominio personalizzato, consulta the section called “Scelta di una policy di sicurezza”.
- Per informazioni su come disattivare l'endpoint predefinito per il nome di dominio personalizzato, consulta the section called “Disabilitazione dell'endpoint predefinito”.
- Per informazioni su come utilizzare i controlli dell'integrità di Route 53 per controllare il failover DNS da un'API di Gateway API, consulta the section called “failover DNS”.

Se è la prima volta che crei un nome di dominio personalizzato, ti consigliamo di iniziare con the section called “Prepara i certificati in AWS Certificate Manager”, per specificare il certificato, quindi the section called “Configurazione di un nome di dominio personalizzato regionale” per creare un nome di dominio personalizzato regionale.

Prepara i certificati in AWS Certificate Manager

Prima di configurare un nome di dominio personalizzato per un'API, è necessario che sia già presente un certificato SSL/TLS in AWS Certificate Manager. Per ulteriori informazioni, consulta la Guida per l'utente AWS Certificate Manager.

Considerazioni

Di seguito sono riportate le considerazioni relative al tuo SSL/TLS certificato.

- Se crei un nome di dominio personalizzato ottimizzato per i dispositivi edge, API Gateway lo sfrutta per supportare i certificati per i nomi CloudFront di dominio personalizzati. Pertanto, i requisiti e i vincoli di un certificato di nome di dominio personalizzato sono dettati da. SSL/TLS CloudFront La dimensione massima della chiave pubblica è ad esempio di 2048, mentre la dimensione della chiave privata può essere di 1024, 2048 e 4096. La dimensione della chiave pubblica è determinata dall'autorità di certificazione usata. Chiedi all'autorità di certificazione di restituire chiavi di dimensioni diverse dalla lunghezza predefinita. Per ulteriori informazioni, consulta Accesso sicuro agli oggetti e Creazione di cookie firmati URLs e firmati.
- Se crei un nome di dominio personalizzato regionale, la dimensione massima della chiave pubblica è 2048.
- Per usare un certificato ACM con un nome di dominio personalizzato regionale, devi richiedere o importare il certificato nella stessa Regione dell'API. Il certificato deve coprire il nome di dominio personalizzato.
- Per usare un certificato ACM con un nome di dominio personalizzato ottimizzato per l'edge, devi richiedere o importare il certificato nella Regione Stati Uniti orientali (Virginia settentrionale) – `us-east-1`.
- È necessario disporre di un nome di dominio registrato, ad esempio *`example.com`*. È possibile usare Amazon Route 53 o un registrar di dominio di terze parti accreditato. Per un elenco dei registrar, consulta la pagina Accredited Registrar Directory nel sito Web ICANN.

Per creare o importare un SSL/TLS certificato in ACM

Le seguenti procedure mostrano come creare o importare un SSL/TLS certificato per un nome di dominio.

To request a certificate provided by ACM for a domain name

1. Accedere alla console AWS Certificate Manager.
2. Scegliere Request a certificate (Richiedi un certificato).
3. Per Tipo di certificato, scegli Richiedi un certificato pubblico.
4. Scegli Next (Successivo).
5. In Nome di dominio completo, immetti un nome di dominio personalizzato per la tua API, ad esempio `api.example.com`.
6. Facoltativamente, scegliere Add another name to this certificate (Aggiungi un altro nome a questo certificato).

7. Per Metodo di convalida, scegli un metodo per convalidare la proprietà del dominio.
8. Per Algoritmo chiave, scegli un algoritmo di crittografia.
9. Scegli Richiedi.
10. Per una richiesta valida, un proprietario registrato del dominio Internet deve accettare la richiesta prima che ACM emetta il certificato. Se utilizzi Route 53 per gestire i record DNS pubblici, puoi aggiornarli direttamente tramite la console ACM.

To import into ACM a certificate for a domain name

1. Richiedi un SSL/TLS certificato con codifica PEM per il tuo nome di dominio personalizzato da un'autorità di certificazione. Per un elenco parziale di questi CAs, consulta Mozilla Included CA List.

   a. Generare una chiave privata per il certificato e salvare l'output in un file usando il kit di strumenti OpenSSL disponibile nel sito Web OpenSSL:

   ```
   openssl genrsa -out private-key-file 2048
   ```

   b. Genera una richiesta di firma del certificato con la chiave privata generata in precedenza, usando OpenSSL:

   ```
   openssl req -new -sha256 -key private-key-file -out CSR-file
   ```

   c. Invia la richiesta di firma del certificato all'autorità di certificazione e salva il certificato risultante.

   d. Scarica la catena di certificati dall'autorità di certificazione.

   > **Note**
   >
   > Se ottieni la chiave privata in un altro modo e la chiave è crittografata, puoi usare il comando seguente per decrittarla prima di inviarla ad API Gateway per la configurazione di un nome di dominio personalizzato.
   >
   > ```
   > openssl pkcs8 -topk8 -inform pem -in MyEncryptedKey.pem -outform pem -
   > nocrypt -out MyDecryptedKey.pem
   > ```

2. Carica il certificato su AWS Certificate Manager:

a. Accedere alla console AWS Certificate Manager.

b. Seleziona Import a certificate (Importa un certificato).

c. Per Corpo certificato, inserisci il corpo del certificato server in formato PEM fornito dall'autorità di certificazione. Di seguito è illustrato un esempio abbreviato di tale certificato.

```
-----BEGIN CERTIFICATE-----
EXAMPLECA+KgAwIBAgIQJ1XxJ8Pl++gOfQtj0IBoqDANBgkqhkiG9w0BAQUFADBB
...
az8Cg1aicxLBQ7EaWIhhgEXAMPLE
-----END CERTIFICATE-----
```

d. Per Chiave privata del certificato, inserisci la chiave privata del certificato in formato PEM. Di seguito è illustrato un esempio abbreviato di tale chiave.

```
-----BEGIN RSA PRIVATE KEY-----
EXAMPLEBAAKCAQEA2Qb3LDHD7StY7Wj6U2/opV6Xu37qUCCkeDWhwpZMYJ9/nETO
...
1qGvJ3u04vdnzaYN5WoyN5LFckrlA71+CszD1CGSqbVDWEXAMPLE
-----END RSA PRIVATE KEY-----
```

e. Per Catena di certificati, inserisci i certificati intermedi in formato PEM e, facoltativamente, il certificato root, uno dopo l'altro senza righe vuote. Se includi il certificato root, la catena di certificati deve iniziare con i certificati intermedi e terminare con il certificato root. Usa i certificati intermedi forniti dall'autorità di certificazione. Non includere intermediari non presenti nella catena del percorso di trust. Di seguito è illustrato un esempio abbreviato.

```
-----BEGIN CERTIFICATE-----
EXAMPLECA4ugAwIBAgIQWrYdrB5NogYUx1U9Pamy3DANBgkqhkiG9w0BAQUFADCB
...
8/ifBlIK3se2e4/hEfcEejX/arxbx1BJCHBvlEPNnsdw8EXAMPLE
-----END CERTIFICATE-----
```

Ecco un altro esempio.

```
-----BEGIN CERTIFICATE-----
Intermediate certificate 2
-----END CERTIFICATE-----
```

```
-----BEGIN CERTIFICATE-----
Intermediate certificate 1
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Optional: Root certificate
-----END CERTIFICATE-----
```

f. Scegli Successivo e di nuovo Successivo.

Dopo la creazione o l'importazione del certificato, prendi nota del relativo ARN. Sarà necessario in seguito per la configurazione del nome di dominio personalizzato.

Configurazione di un nome di dominio personalizzato regionale in Gateway API

Usa un nome di dominio personalizzato regionale per creare un URL di base dell'API intuitivo. Con un nome di dominio personalizzato regionale, puoi mappare le fasi HTTP e REST API allo stesso nome di dominio personalizzato e utilizzare l'autenticazione TLS reciproca.

Considerazioni

Di seguito sono riportate alcune considerazioni relative al nome di dominio personalizzato regionale:

- Devi fornire un certificato ACM specifico della Regione. Il certificato deve trovarsi nella stessa Regione dell'API. Per ulteriori informazioni sulla creazione o il caricamento di un certificato di un nome di dominio personalizzato consulta the section called “Prepara i certificati in AWS Certificate Manager”.
- Quando si crea un nome di dominio personalizzato regionale (o si esegue la migrazione di uno di essi) con un certificato ACM, Gateway API crea un ruolo collegato ai servizi nel tuo account. Il ruolo collegato ai servizi è necessario per collegare il certificato ACM all'endpoint regionale. Il ruolo è denominato AWSServiceRoleForAPIGateway e sarà collegato alla policy gestita APIGatewayServiceRolePolicy. Per ulteriori informazioni sull'uso del ruolo collegato ai servizi, consulta Utilizzo dei ruoli collegati ai servizi.
- Dopo aver creato il nome di dominio personalizzato regionale, è necessario creare un record DNS per far sì che il nome di dominio personalizzato punti al dominio regionale. Questo consente al traffico destinato al nome di dominio personalizzato di essere reindirizzato al nome host regionale dell'API.

  Il record DNS può essere un record CNAME o un record alias A. Se si utilizza Route 53 come provider DNS, occorre creare un record di alias di tipo A. Se si utilizza un provider DNS di terze

parti, occorre usare un record CNAME. Se si utilizza un record CNAME e si crea un endpoint VPC dell'interfaccia Gateway API con DNS privato abilitato per un'API privata, non è possibile risolvere il nome di dominio personalizzato all'interno del VPC che ospita l'API privata.

Creazione di un nome di dominio personalizzato regionale

La procedura seguente mostra come creare un nome di dominio personalizzato regionale. Una volta completata questa procedura, si crea una regola di routing per instradare le fasi dell'API al nome di dominio personalizzato.

Console di gestione AWS

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere Custom Domain Names (Nomi di dominio personalizzati) nel riquadro di navigazione principale.
3. Seleziona Crea.
4. In Domain name (Nome di dominio), immettere un nome di dominio.
5. Per Modalità di routing, scegliere Solo regole di routing.

   In questa modalità di routing, puoi inviare traffico dal tuo nome di dominio personalizzato al tuo solo utilizzando le regole di routing. APIs Per ulteriori informazioni, consulta the section called "Invia traffico al tuo APIs tramite il tuo nome di dominio personalizzato in API Gateway".
6. Per Versione TLS minima, seleziona una versione.
7. Sotto Configurazione endpoint, per Tipo di endpoint API scegli Regionale.
8. Scegliere un certificato ACM. Il certificato deve trovarsi nella stessa regione dell'API.
9. Scegli Create (Crea).

AWS CLI

Il create-domain-namecomando seguente crea un nome di dominio personalizzato:

```
aws apigatewayv2 create-domain-name \
    --domain-name 'regional.example.com' \
    --domain-name-configurations CertificateArn=arn:aws:acm:us-
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678 \
    --routing-mode ROUTING_RULE_ONLY
```

L'output sarà simile al seguente:

```
{
    "ApiMappingSelectionExpression": "$request.basepath",
    "DomainName": "regional.example.com",
    "DomainNameConfigurations": [
        {
            "ApiGatewayDomainName": "d-numh1z56v6.execute-api.us-
west-2.amazonaws.com",
            "CertificateArn": "arn:aws:acm:us-
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678",
            "DomainNameStatus": "AVAILABLE",
            "EndpointType": "REGIONAL",
            "HostedZoneId": "Z2OJLYMUO9EFXC",
            "SecurityPolicy": "TLS_1_2"
        }
        "RoutingMode": "ROUTING_RULE_ONLY"
    ]
}
```

Il valore della proprietà `DomainNameConfigurations` restituisce il nome host dell'API regionale. Devi creare un record DNS per puntare il tuo nome di dominio personalizzato al nome del dominio regionale. Questo consente al traffico destinato al nome di dominio personalizzato di essere reindirizzato al nome host dell'API regionale.

Creazione di una regola di routing per il nome di dominio personalizzato regionale

Dopo aver creato il tuo nome di dominio personalizzato, configuri il modo in cui il traffico viene instradato dal tuo nome di dominio personalizzato al tuo APIs. Poiché hai impostato la modalità di routing su`ROUTING_RULE_ONLY`, utilizzi le regole di routing per indirizzare le richieste in entrata dal tuo nome di dominio personalizzato al tuo. APIs

In questo esempio, si crea una regola catch-all che instrada a una fase dell'API tutte le richieste in entrata per il nome di dominio personalizzato. È anche possibile configurare le regole di routing in base a diverse condizioni di intestazione e percorso. Per ulteriori informazioni, consulta the section called “Regole di routing”.

Console di gestione AWS

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.

2. Scegliere un nome di dominio personalizzato.
3. Nella scheda Dettagli di routing, scegliere Aggiungi una regola di routing.
4. Scegliere Aggiungi una nuova condizione per aggiungere una nuova condizione.
5. Mantenere questa regola senza condizioni. In tal modo si instradano all'API di destinazione e alla fase di destinazione tutte le richieste per il nome di dominio personalizzato.
6. Per Azione, selezionare nel menu a discesa l'API e la fase di destinazione.
7. Scegli Next (Successivo).
8. Nel campo Priorità, immettere **`100`**.

   Gateway API valuta le regole in ordine di priorità, dal valore più basso a quello più alto. Poiché si tratta di una regola catch-all, si utilizza una priorità elevata in modo che Gateway API possa soddisfare qualsiasi regola aggiuntiva creata.
9. Scegliere Crea una regola di routing.

AWS CLI

Il comando `create-routing-rule` seguente crea una regola di routing catch-all:

```
aws apigatewayv2 create-routing-rule \
  --domain-name 'regional.example.com' \
  --priority 100 \
  --conditions  \
  --actions '[{
    "InvokeApi": {
      "ApiId": "a1b2c3",
      "Stage": "prod"
    }
  }]'
```

È possibile modificare la modalità di routing e creare nuove regole in qualsiasi momento. Per ulteriori informazioni, consulta the section called "Invia traffico al tuo APIs tramite il tuo nome di dominio personalizzato in API Gateway".

Creazione di un record DNS per il nome di dominio personalizzato regionale

Dopo aver creato il nome di dominio personalizzato e le mappature dei percorsi di base, puoi creare un record DNS in modo che il nome di dominio personalizzato punti al nome di dominio regionale appena creato.

Console di gestione AWS

Per utilizzare il Console di gestione AWS, segui la documentazione di Route 53 sulla configurazione di Route 53 per instradare il traffico verso API Gateway.

AWS CLI

Per configurare i record DNS in modo da mappare il nome di dominio personalizzato regionale al relativo nome host dell'ID della zona ospitata, crea innanzitutto un file JSON contenente la configurazione per configurare un record DNS per il nome di dominio regionale.

Il seguente `setup-dns-record.json` mostra come creare un record DNS `A` per mappare un nome di dominio personalizzato regionale (`regional.example.com`) al relativo nome host regionale (`d-numh1z56v6.execute-api.us-west-2.amazonaws.com`) fornito durante la creazione del nome di dominio personalizzato. Le proprietà `DNSName` e `HostedZoneId` di `AliasTarget` possono prendere rispettivamente i valori `regionalDomainName` e `regionalHostedZoneId` del nome di dominio personalizzato. Puoi anche ottenere la Regional Route 53 Hosted Zone IDs in Amazon API Gateway Endpoints and Quotas.

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

Quanto segue change-resource-record-setscrea un record DNS per il tuo nome di dominio personalizzato regionale:

```
aws route53 change-resource-record-sets \
    --hosted-zone-id Z2OJLYMUO9EFXC \
    --change-batch file://path/to/your/setup-dns-record.json
```

Sostituisci `hosted-zone-id` con l'ID della zona ospitata di Route 53 del record DNS impostato nel tuo account. Il valore del parametro `change-batch` punta a un file JSON (*`setup-dns-record.json`*) in una cartella (*`path/to/your`*).

Configurazione di un nome di dominio personalizzato ottimizzato per l'edge in Gateway API

Quando crei un nome di dominio personalizzato per un'API ottimizzata per l'edge, API Gateway configura una CloudFront distribuzione e un record DNS per mappare il nome di dominio API al CloudFront nome di dominio di distribuzione. Le richieste per l'API vengono quindi indirizzate ad API Gateway tramite la distribuzione mappata CloudFront . Questa mappatura è per le richieste API associate al nome di dominio personalizzato da indirizzare ad API Gateway tramite la distribuzione CloudFront mappata.

Considerazioni

Di seguito sono riportate alcune considerazioni relative al nome di dominio personalizzato ottimizzato per l'edge:

- Per configurare un nome di dominio personalizzato ottimizzato per i dispositivi periferici o per aggiornarne il certificato, è necessario disporre dell'autorizzazione per aggiornare le distribuzioni. CloudFront

  Per aggiornare le distribuzioni sono necessarie le seguenti autorizzazioni: CloudFront

  JSON

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

```
            "Action": [
                "cloudfront:updateDistribution"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

- È necessario richiedere o importare un certificato per il nome di dominio personalizzato ottimizzato per l'edge nella Regione Stati Uniti orientali (Virginia settentrionale) – `us-east-1`.
- La CloudFront distribuzione creata da API Gateway è di proprietà di un account specifico della regione affiliato ad API Gateway. Quando si tracciano le operazioni per creare e aggiornare tale CloudFront distribuzione CloudTrail, è necessario utilizzare questo ID account API Gateway. Per ulteriori informazioni, consulta Registra la creazione di un nome di dominio personalizzato CloudTrail.
- API Gateway supporta nomi di dominio personalizzati ottimizzati per i dispositivi perimetrali sfruttando la Server Name Indication (SNI) sulla distribuzione. CloudFront Per ulteriori informazioni sull'utilizzo di nomi di dominio personalizzati su una CloudFront distribuzione, incluso il formato di certificato richiesto e la dimensione massima della lunghezza della chiave del certificato, consulta Using Alternate Domain Names and HTTPS nella Amazon CloudFront Developer Guide
- Affinché un nome di dominio personalizzato ottimizzato per l'edge sia pronto sono necessari circa 40 minuti.
- Dopo aver creato il nome di dominio personalizzato ottimizzato per i dispositivi periferici, devi creare un record DNS per mappare il nome di dominio personalizzato al nome di distribuzione. CloudFront

Creazione di un nome di dominio personalizzato ottimizzato per l'edge

La procedura seguente descrive come creare un nome di dominio personalizzato ottimizzato per l'edge per un'API.

Console di gestione AWS

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere Custom Domain Names (Nomi di dominio personalizzati) nel riquadro di navigazione principale.

3. Scegli Aggiungi nome di dominio.
4. In Domain name (Nome di dominio), immettere un nome di dominio.
5. Per la modalità Routing, scegli _ONLY. API_MAPPING
6. Per Tipo di endpoint API, scegli Ottimizzato per l'edge.
7. Scegliere una versione di TLS minima.
8. Scegliere un certificato ACM.
9. Scegli Aggiungi nome di dominio.

REST API

1. Chiamare domainname:create, specificando il nome di dominio personalizzato e l'ARN di un certificato archiviato in AWS Certificate Manager.

   La chiamata API riuscita restituisce una `201 Created` risposta contenente l'ARN del certificato e il nome di CloudFront distribuzione associato nel relativo payload.
2. Annota il nome del dominio di CloudFront distribuzione mostrato nell'output. Ti servirà nel passaggio successivo per impostare il target dell'alias del record A del dominio personalizzato nel DNS.

Per esempi di codice di questa chiamata all'API REST, consulta domainname:create.

Un nome di dominio personalizzato ottimizzato per l'edge impiega circa 40 minuti per essere pronto, ma la console visualizza immediatamente il nome di dominio di CloudFront distribuzione associato, sotto forma di*`distribution-id`*`.cloudfront.net`, insieme all'ARN del certificato. Nel frattempo, puoi creare una mappatura del percorso di base o una regola di routing e quindi configurare l'alias del record DNS per mappare il nome di dominio personalizzato al nome di dominio di distribuzione associato. CloudFront

Configurazione della mappatura del percorso di base di un'API con un nome di dominio personalizzato come nome host

Poiché la modalità di routing è impostata su`API_MAPPING_ONLY`, è possibile utilizzare la mappatura del percorso di base per utilizzare un singolo nome di dominio personalizzato come nome host di più nomi. APIs In questo modo, un'API è accessibile tramite la combinazione di nome di dominio personalizzato e percorso di base associato.

Ad esempio, se in Gateway API hai creato un'API denominata `PetStore` e un'altra denominata `Dogs` e hai configurato un nome di dominio personalizzato `api.example.com`, puoi impostare l'URL dell'API `PetStore` come `https://api.example.com`.

In questo modo, l'API `PetStore` viene associata al percorso di base di una stringa vuota. Se imposti l'URL dell'API `PetStore` come `https://api.example.com/PetStore`, l'API `PetStore` viene associata al percorso di base di `PetStore`. Puoi assegnare un percorso di base `MyDogList` per l'API `Dogs`. L'URL `https://api.example.com/MyDogList` è quindi l'URL root dell'API `Dogs`.

Per configurare le mappature delle API su più livelli, è possibile utilizzare solo un nome di dominio personalizzato regionale. I nomi di dominio personalizzati ottimizzati per l'edge non sono supportati. Per ulteriori informazioni, consulta the section called “Mappature API”.

La procedura seguente consente di impostare le mappature dell'API per mappare percorsi dal nome di dominio personalizzato alle fasi API.

Console di gestione AWS

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Seleziona Custom Domain Names (Nomi di dominio personalizzati) nel riquadro di navigazione principale della console API Gateway.
3. Scegliere un nome di dominio personalizzato.
4. Scegliere Configure API mappings (Configura mappature API).
5. Scegliere Add new mapping (Aggiungi nuova mappatura).
6. Specificare API, Stage (Fase) e Path (Percorso) (facoltativo) per la mappatura.
7. Scegli Save (Salva).

REST API

Richiamare basepathmapping:create su un nome di dominio personalizzato specifico, indicando le proprietà `basePath`, `restApiId` e `stage` della distribuzione nel payload della richiesta.

In caso di esito positivo, la chiamata API restituisce una risposta `201 Created`.

Per esempi di codice della chiamata all'API REST, consulta basepathmapping:create.

Per una maggiore flessibilità su come indirizzare il traffico verso il proprio APIs, è possibile modificare la modalità di routing in `ROUTING_RULE_ONLY` o `ROUTING_RULE_THEN_API_MAPPING` e creare

una regola di routing. Per ulteriori informazioni, consulta the section called “Invia traffico al tuo APIs tramite il tuo nome di dominio personalizzato in API Gateway”.

Creazione di un record DNS per il nome di dominio personalizzato ottimizzato per l'edge

Dopo aver avviato la creazione del nome di dominio personalizzato ottimizzato per l'edge, configura l'alias del record DNS.

Ti consigliamo di utilizzare Route 53 per creare un alias di record A per il tuo nome di dominio personalizzato e specificare il nome di dominio di CloudFront distribuzione come destinazione dell'alias. Ciò significa che Route 53 è in grado di instradare il nome di dominio personalizzato anche se si tratta di un apex di zona. Per ulteriori informazioni, consulta la pagina relativa alla scelta tra set di record della risorsa alias e non alias nella Guida per gli sviluppatori di Amazon Route 53.

Per istruzioni su Amazon Route 53, consulta Instradamento del traffico a un'API Amazon API Gateway utilizzando il nome di dominio nella Guida per gli sviluppatori di Amazon Route 53.

Registra la creazione di un nome di dominio personalizzato CloudTrail

Quando CloudTrail è abilitato per la registrazione delle chiamate API Gateway effettuate dal tuo account, API Gateway registra gli aggiornamenti di CloudFront distribuzione associati quando viene creato o aggiornato un nome di dominio personalizzato per un'API. Questi log sono disponibili in `us-east-1`. Poiché queste CloudFront distribuzioni sono di proprietà di API Gateway, ognuna di queste CloudFront distribuzioni segnalate è identificata da uno dei seguenti account API Gateway specifici della regione, anziché dall'ID dell' IDsaccount del proprietario dell'API.

Region	ID account
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

Region	ID account
eu-west-3	061510835048
eu-central-1	474240146802
eu-central-2	166639821150
eu-north-1	394634713161
eu-south-1	753362059629
eu-south-2	359345898052
ap-northeast-1	969236854626
ap-northeast-2	020402002396
ap-northeast-3	360671645888
ap-southeast-1	195145609632
ap-southeast-2	798376113853
ap-southeast-3	652364314486
ap-southeast-4	849137399833
ap-south-1	507069717855
ap-south-2	644042651268
ap-east-1	174803364771
sa-east-1	287228555773
me-south-1	855739686837
me-central-1	614065512851

Rotazione di un certificato importato in ACM

ACM gestisce automaticamente il rinnovo dei certificati emessi. Non è necessario ruotare alcun certificato emesso da ACM per i nomi di dominio personalizzati. CloudFront lo gestisce per tuo conto.

Se tuttavia importi un certificato in ACM e lo usi per un nome di dominio personalizzato, devi ruotarlo prima della scadenza. A tale scopo, è necessario importare un nuovo certificato di terze parti per il nome di dominio e ruotare il certificato esistente in quello nuovo. Il processo deve essere ripetuto quando il nuovo certificato importato scade. In alternativa, puoi richiedere a ACM di emettere un nuovo certificato per il nome di dominio e ruotare il certificato esistente in quello nuovo emesso da ACM. Dopodiché, puoi lasciare ACM e CloudFront gestire automaticamente la rotazione dei certificati. Per creare o importare un nuovo certificato ACM, segui la procedura riportata in the section called “Per creare o importare un SSL/TLS certificato in ACM”.

La procedura seguente descrive come ruotare un certificato per un nome di dominio.

> **Note**
>
> Per ruotare un certificato importato in ACM sono necessari circa 40 minuti.

Console di gestione AWS

1. Richiedi o importa un certificato in ACM.
2. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
3. Seleziona Custom Domain Names (Nomi di dominio personalizzati) nel riquadro di navigazione principale della console API Gateway.
4. Scegli un nome di dominio personalizzato ottimizzato per l'edge.
5. Per Configurazione endpoint scegli Modifica.
6. Seleziona un certificato nell'elenco a discesa Certificato ACM.
7. Scegli Salva per iniziare la rotazione del certificato per il nome di dominio personalizzato.

REST API

Richiama l'operazione domainname:update indicando l'ARN del nuovo certificato ACM per il nome di dominio specificato.

AWS CLI

Quanto segue update-domain-nameaggiorna il certificato ACM per un nome di dominio ottimizzato per l'edge.

```
aws apigateway update-domain-name \
    --domain-name edge.example.com \
    --patch-operations "op='replace',path='/certificateArn',value='arn:aws:acm:us-
east-2:111122223333:certificate/CERTEXAMPLE123EXAMPLE'"
```

Quanto segue update-domain-nameaggiorna il certificato ACM per un nome di dominio regionale.

```
aws apigateway update-domain-name \
    --domain-name regional.example.com \
    --patch-operations "op='replace',path='/
regionalCertificateArn',value='arn:aws:acm:us-east-2:111122223333:certificate/
CERTEXAMPLE123EXAMPLE'"
```

Chiamata dell'API con nomi di dominio personalizzati quando si utilizza una mappatura percorso di base

Chiamare un'API con un nome di dominio personalizzato equivale a chiamare l'API con il nome di dominio predefinito, a condizione che venga usato l'URL corretto.

Gli esempi seguenti confrontano e contrappongono un insieme di impostazioni personalizzate predefinite URLs e corrispondenti URLs di due APIs (`udxjefeqf3duz`) in una regione specificata (`us-east-1`) e di un determinato nome di dominio personalizzato (`api.example.com`).

ID API	Fase	URL predefinito	Percorso di base	URL personalizzato
udxjef	prod	https://udxjef.execute-api.us-east-1.amazonaws.com/prod	/petstore	https://api.example.com/negozio di animali

ID API	Fase	URL predefinito	Percorso di base	URL personalizzato
udxjef	tst	https://udxjef.execute-api.us-east-1.amazonaws.com/tst	/petdepot	https://api.example.com/petdepot
qf3duz	dev	https://qf3duz.execute-api.us-east-1.amazonaws.com/dev	/bookstore	https://api.example.com/libreria
qf3duz	tst	https://qf3duz.execute-api.us-east-1.amazonaws.com/tst	/bookstand	https://api.example.com/leggio

Per una maggiore flessibilità su come indirizzare il traffico verso il tuo APIs, puoi creare una regola di routing. Per ulteriori informazioni, consulta the section called “Invia traffico al tuo APIs tramite il tuo nome di dominio personalizzato in API Gateway”.

API Gateway supporta i nomi di dominio personalizzati per un'API usando SNI (Server Name Indication, Indicazione nome server). Puoi richiamare l'API con un nome di dominio personalizzato usando un browser o una libreria client che supporta SNI.

API Gateway applica SNI sulla CloudFront distribuzione. Per informazioni su come vengono CloudFront utilizzati i nomi di dominio personalizzati, consulta Amazon CloudFront Custom SSL.

Migrazione di un nome di dominio personalizzato a un tipo di endpoint API diverso in Gateway API

Puoi eseguire la migrazione di un nome di dominio personalizzato tra endpoint ottimizzati per edge ed endpoint regionali. Non è possibile eseguire la migrazione di un nome di dominio personalizzato pubblico a un nome di dominio personalizzato privato. Aggiungi prima di tutto il nuovo tipo di

configurazione dell'endpoint all'elenco `endpointConfiguration.types` esistente per il nome di dominio personalizzato. Configura quindi un record DNS in modo che il nome di dominio personalizzato punti all'endpoint di cui è appena stato effettuato il provisioning. Infine, si rimuove l'endpoint del nome di dominio personalizzato obsoleto.

Considerazioni

Di seguito sono riportate alcune considerazioni relative alla migrazione del dominio personalizzato tra un endpoint API regionale e un endpoint API ottimizzato per l'edge:

- Un nome di dominio personalizzato ottimizzato per l'edge richiede un certificato fornito da ACM della Regione Stati Uniti orientali (Virginia settentrionale), `us-east-1`. Questo certificato è distribuito un tutte le posizioni geografiche.
- Un nome di dominio personalizzato regionale richiede un certificato fornito da ACM nella stessa Regione che ospita l'API. È possibile eseguire la migrazione di un nome di dominio personalizzato ottimizzato per l'edge che non si trova nella Regione `us-east-1` a un nome di dominio personalizzato regionale richiedendo un nuovo certificato ACM della Regione locale dell'API.
- Per completare la migrazione tra un nome di dominio personalizzato ottimizzato per l'edge e un nome di dominio personalizzato regionale possono essere necessari fino a 60 secondi. Il tempo necessario per la migrazione dipende anche da quando si aggiornano i record DNS.
- È possibile aggiungere una configurazione aggiuntiva dell'endpoint solo se la modalità di accesso all'endpoint è impostata su. `BASIC` Una volta che sono disponibili due configurazioni degli endpoint, non è possibile modificare la modalità di accesso agli endpoint. Per ulteriori informazioni, consulta the section called “Modalità di accesso agli endpoint”.
- Se il nome di dominio personalizzato utilizza una politica di sicurezza che inizia con`SecurityPolicy_`, quando aggiungi un nuovo tipo di configurazione dell'endpoint, la modalità di accesso all'endpoint è la stessa per entrambi i tipi di endpoint e devi scegliere una policy di sicurezza che inizi con `SecurityPolicy_` per il nuovo tipo di configurazione dell'endpoint.

Migrazione di nomi di dominio personalizzati

> **Note**
>
> Per completare la migrazione, è necessario rimuovere l'endpoint obsoleto dal nome di dominio personalizzato.

La seguente procedura illustra come eseguire la migrazione di un nome di dominio personalizzato ottimizzato per l'edge a un nome di dominio personalizzato regionale.

Console di gestione AWS

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere Custom Domain Names (Nomi di dominio personalizzati) nel riquadro di navigazione principale.
3. Scegli un nome di dominio personalizzato ottimizzato per l'edge.
4. Per Configurazione endpoint scegli Modifica.
5. Scegli Aggiungi endpoint regionale.
6. Per Certificato ACM scegli un certificato.

   Il certificato regionale deve trovarsi nella stessa Regione dell'API regionale.
7. Scegli Save changes (Salva modifiche).
8. Configura un record DNS in modo che il nome di dominio personalizzato regionale punti al nome host regionale. Per ulteriori informazioni, consulta la configurazione di Route 53 per instradare il traffico in Gateway API.
9. Dopo aver verificato che la configurazione DNS utilizza l'endpoint corretto, elimina la configurazione dell'endpoint ottimizzato per l'edge. Scegli il nome di dominio personalizzato, quindi per Configurazione dell'endpoint (ottimizzato per edge) seleziona Elimina.
10. Conferma la scelta ed elimina l'endpoint.

AWS CLI

Il update-domain-namecomando seguente migra un nome di dominio personalizzato ottimizzato da edge a un nome di dominio personalizzato regionale:

```
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" }
      ]'
```

Il certificato regionale deve trovarsi nella stessa regione dell'API regionale.

L'output sarà simile al seguente:

```
{
    "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"
}
```

Per il nome di dominio personalizzato regionale migrato, la proprietà `regionalDomainName` risultante restituisce il nome host dell'API regionale. Devi configurare un record DNS in modo che il nome di dominio personalizzato regionale punti a questo nome host regionale. Questo consente al traffico destinato al nome di dominio personalizzato di essere instradato all'host regionale.

Dopo avere impostato il record DNS, è possibile rimuovere il nome di dominio personalizzato ottimizzato per l'edge. Il update-domain-namecomando seguente rimuove il nome di dominio personalizzato ottimizzato per i bordi:

```
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"}
        ]'
```

La procedura seguente mostra come migrare un nome di dominio personalizzato ottimizzato per i dispositivi periferici che utilizza una politica di sicurezza avanzata verso un nome di dominio personalizzato regionale che utilizza anche una politica di sicurezza avanzata.

Console di gestione AWS

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere Custom Domain Names (Nomi di dominio personalizzati) nel riquadro di navigazione principale.
3. Scegli un nome di dominio personalizzato ottimizzato per l'edge.
4. Per Configurazione endpoint scegli Modifica.
5. Scegli Aggiungi endpoint regionale.
6. Per Certificato ACM scegli un certificato.

   Il certificato regionale deve trovarsi nella stessa Regione dell'API regionale.
7. Per Politica di sicurezza, scegli una politica di sicurezza che inizi con. `SecurityPolicy_`
8. Per la modalità di accesso Endpoint, scegli Basic.
9. Scegli Save changes (Salva modifiche).
10. Configura un record DNS in modo che il nome di dominio personalizzato regionale punti al nome host regionale. Per ulteriori informazioni, consulta la configurazione di Route 53 per instradare il traffico in Gateway API.
11. Dopo aver verificato che la configurazione DNS utilizza l'endpoint corretto, elimina la configurazione dell'endpoint ottimizzato per l'edge. Scegli il nome di dominio personalizzato, quindi per Configurazione dell'endpoint (ottimizzato per edge) seleziona Elimina.
12. Conferma la scelta ed elimina l'endpoint.

AWS CLI

Il update-domain-namecomando seguente migra un nome di dominio personalizzato ottimizzato dai bordi in un nome di dominio personalizzato regionale:

```
aws apigateway update-domain-name \
    --domain-name 'api.example.com' \
    --patch-operations  '[
        { "op":"add", "path": "/endpointConfiguration/types","value": "REGIONAL" },
        { "op":"replace", "path": "/securityPolicy",
 "value":"SecurityPolicy_TLS13_1_3_2025_09"},
        { "op":"add", "path": "/regionalCertificateArn", "value": "arn:aws:acm:us-
west-2:123456789012:certificate/cd833b28-58d2-407e-83e9-dce3fd852149" }
      ]'
```

Il certificato regionale deve trovarsi nella stessa regione dell'API regionale.

L'output sarà simile al seguente:

```
{
    "certificateArn": "arn:aws:acm:us-
east-1:123456789012:certificate/34a95aa1-77fa-427c-aa07-3a88bd9f3c0a",
    "certificateName": "edge-cert",
    "certificateUploadDate": "2017-10-16T23:22:57Z",
    "distributionDomainName": "d1frvgze7vy1bf.cloudfront.net",
    "domainName": "api.example.com",
    "endpointConfiguration": {
        "types": [
            "EDGE",
            "REGIONAL"
        ]
    },
    "securityPolicy": "SecurityPolicy_TLS13_1_3_2025_09",
    "endpointAccessMode": "BASIC",
    "regionalCertificateArn": "arn:aws:acm:us-west-2:123456789012:certificate/
cd833b28-58d2-407e-83e9-dce3fd852149",
    "regionalDomainName": "d-fdisjghyn6.execute-api.us-west-2.amazonaws.com"
}
```

Per il nome di dominio personalizzato regionale migrato, la proprietà `regionalDomainName` risultante restituisce il nome host dell'API regionale. Devi configurare un record DNS in modo che il nome di dominio personalizzato regionale punti a questo nome host regionale. Questo consente al traffico destinato al nome di dominio personalizzato di essere instradato all'host regionale.

Dopo avere impostato il record DNS, è possibile rimuovere il nome di dominio personalizzato ottimizzato per l'edge. Il update-domain-namecomando seguente rimuove il nome di dominio personalizzato ottimizzato per i bordi:

```
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"}
        ]'
```

La seguente procedura mostra come eseguire la migrazione di un nome di dominio personalizzato regionale a un nome di dominio personalizzato ottimizzato per l'edge.

Console di gestione AWS

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel pannello di navigazione principale, scegli Nomi di dominio personalizzati.
3. Scegli un nome di dominio personalizzato regionale.
4. Per Configurazione endpoint scegli Modifica.
5. Scegli Aggiungi endpoint ottimizzato per edge.
6. Per Certificato ACM scegli un certificato.

   Il certificato di dominio ottimizzato per edge deve essere creato nella regione `us-east-1`.
7. Scegli Save (Salva).
8. Configura un record DNS in modo che il nome di dominio personalizzato ottimizzato per l'edge punti al nome host ottimizzato per l'edge. Per ulteriori informazioni, consulta la configurazione di Route 53 per instradare il traffico in Gateway API.
9. Dopo aver verificato che la configurazione DNS utilizza l'endpoint corretto, elimina la configurazione dell'endpoint regionale. Scegli il nome di dominio personalizzato, quindi per Configurazione dell'endpoint (regionale) seleziona Elimina.
10. Conferma la scelta ed elimina l'endpoint.

AWS CLI

Il update-domain-namecomando seguente migra il nome di dominio personalizzato regionale in un nome di dominio personalizzato ottimizzato per Edge:

```
aws apigateway update-domain-name \
    --domain-name 'api.example.com' \
    --patch-operations  '[
        { "op":"add", "path": "/endpointConfiguration/types","value": "EDGE" },
        { "op":"add", "path": "/certificateName", "value": "edge-cert" },
 {"op":"add", "path": "/certificateArn", "value": "arn:aws:acm:us-
east-1:738575810317:certificate/34a95aa1-77fa-427c-aa07-3a88bd9f3c0a"}
      ]'
```

Il certificato di dominio ottimizzato per edge deve essere creato nella regione `us-east-1`.

L'output sarà simile al seguente:

```
{
    "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"
}
```

Per il nome di dominio personalizzato specificato, API Gateway restituisce il nome host dell'API ottimizzata per l'edge come valore della proprietà `distributionDomainName`. Devi impostare un record DNS in modo che il nome di dominio personalizzato ottimizzato per i confini punti al nome di dominio di questa distribuzione. Questo consente al traffico destinato al nome di dominio personalizzato ottimizzato per gli edge di essere instradato al nome host dell'API ottimizzata per edge.

Dopo avere impostato il record DNS, è possibile rimuovere il tipo di endpoint `REGION` del nome di dominio personalizzato. Il update-domain-namecomando seguente rimuove il tipo di endpoint regionale:

```
aws apigateway update-domain-name \
    --domain-name api.example.com \
    --patch-operations '[
        {"op":"remove", "path":"/endpointConfiguration/types", value:"REGIONAL"},
        {"op":"remove", "path":"regionalCertificateArn"}
      ]'
```

L'output sarà simile al seguente:

```
{
```

```
    "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"
    }
}
```

Invia traffico al tuo APIs tramite il tuo nome di dominio personalizzato in API Gateway

Quando configuri la modalità di routing per il tuo nome di dominio personalizzato, imposti il modo in cui il traffico in entrata viene indirizzato al tuo. APIs Invii traffico al tuo indirizzo APIs utilizzando regole di routing, mappature API o regole di routing e mappature API. La sezione seguente illustra quando utilizzare le regole di routing e le mappature API, nonché come impostare la modalità di routing per il nome di dominio personalizzato.

Quando utilizzare le regole di routing

Quando si utilizzano le regole di routing, si indirizzano le richieste in entrata che soddisfano determinate condizioni verso fasi REST specifiche. APIs Ad esempio, una regola può instradare una richiesta alla fase `production` della REST API `users` se contiene l'intestazione `version:v1` e il percorso di base `/users`. Utilizza le regole di routing per creare topologie di routing dinamiche avanzate che supportino casi d'uso come i A/B test o l'aumento dell'utilizzo di nuove versioni del tuo. APIs

Quando si indirizza il traffico a una REST API, è consigliabile utilizzare le regole di routing per il nome di dominio personalizzato. È possibile creare nuovamente qualsiasi mappatura API utilizzando le regole di routing. Per ulteriori informazioni, consulta the section called “Nuova creazione di una mappatura API utilizzando le regole di routing”.

Per REST APIs, puoi anche utilizzare insieme le regole di routing e le mappature delle API. Quando si usano insieme le regole di routing e le mappature API, Gateway API valuta sempre le regole di routing prima di qualsiasi mappatura API. Le regole di routing e le mappature API si utilizzano insieme per migrare gli attuali nomi di dominio personalizzati o per esplorare le regole di routing.

Considerazioni sulle regole di routing

Le seguenti considerazioni potrebbero influire sull'utilizzo delle regole di routing:

- WebSocket oppure HTTP APIs non sono supportati come destinazione APIs per le regole di routing.
- Se il nome di dominio personalizzato presenta mappature API sia su REST che su HTTP APIs, le regole di routing non sono supportate.
- È possibile creare una regola di routing di un dominio personalizzato privato per una REST API privata. È possibile creare una regola di routing di un dominio personalizzato pubblico per un'API regionale oppure ottimizzata per l'edge.
- Non è possibile creare una regola di routing di un dominio personalizzato pubblico per un'API privata. Non è possibile creare una regola di routing di un nome di dominio personalizzato privato per un'API pubblica.

Scelta tra regole di routing e mappature API

Quando possibile, è consigliabile utilizzare le regole di routing. Utilizza le mappature API solo per inviare traffico a un HTTP o a un'API. WebSocket

Impostazione della modalità di routing per il nome di dominio personalizzato

Puoi scegliere la modalità di routing utilizzata da API Gateway per indirizzare il traffico verso il tuo APIs. Per ulteriori informazioni, consulta the section called "Invia traffico al tuo APIs tramite il tuo nome di dominio personalizzato in API Gateway". In questa sezione vengono illustrate le modalità di routing per i nomi di dominio personalizzati. Devi impostare una modalità di routing per il tuo nome di dominio personalizzato per indirizzare il traffico verso il tuo. APIs Sono supportate le seguenti modalità di routing:

- ROUTING_RULE_THEN_ API_MAPPING: utilizza questa modalità per inviare traffico al tuo APIs con regole di routing e mappature API. In questa modalità, tutte le regole di routing hanno la priorità su qualsiasi mappatura API. Per un esempio di questa modalità, consultare the section called "Esempio 2: regole di routing e mappature API".
- ROUTING_RULE_ONLY: utilizza questa modalità per consentire solo alle regole di routing di inviare traffico al tuo. APIs Quando il nome di dominio personalizzato utilizza questa modalità, non è possibile creare una mappatura delle API, ma è possibile utilizzare il comando per visualizzarla. get-api-mappings I chiamanti delle API non possono utilizzare le mappature API per accedere a questo nome di dominio.

- API_MAPPING_ONLY: utilizza questa modalità per consentire solo alle mappature delle API di inviare traffico al tuo. APIs Quando il nome di dominio personalizzato usa questa modalità, non è possibile creare una regola di routing, ma è possibile utilizzare il comando `list-routing-rules` per visualizzarla. I chiamanti delle API non possono utilizzare le regole di routing per accedere a questo nome di dominio.

  Questa è la modalità di routing predefinita per tutti i nomi di dominio esistenti e per tutti i nuovi nomi di dominio che vengono creati.

Quando si crea un nome di dominio personalizzato utilizzando `apigateway`, `API_MAPPING_ONLY` è chiamato `BASE_PATH_MAPPING_ONLY` e `ROUTING_RULE_THEN_API_MAPPING` è chiamato `ROUTING_RULE_THEN_BASE_PATH_MAPPING`. Questo comportamento è presente solo per AWS CLI, o qualsiasi CloudFormation SDKs, non in. Console di gestione AWS

La procedura seguente illustra come modificare la modalità di routing per un nome di dominio personalizzato esistente. Quando si modifica la modalità di routing del nome di dominio personalizzato, i chiamanti delle API non possono accedere al nome di dominio utilizzando una modalità di routing non supportata.

Console di gestione AWS

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere Custom Domain Names (Nomi di dominio personalizzati) nel riquadro di navigazione principale.
3. Scegliere un nome di dominio personalizzato.
4. Per Dettagli del dominio, scegliere Modifica.
5. Per la modalità Routing, scegliete ROUTING_RULE_THEN_. API_MAPPING
6. Scegli Save (Salva).

   Se si modifica la modalità di routing in `ROUTING_RULE_ONLY` o `API_MAPPING_ONLY`, tutte le mappature API o le regole di routing create vengono rimosse dalla pagina dei dettagli del nome di dominio della console. Se si modifica la modalità di routing per supportare regole di routing o mappature API, queste risorse torneranno a essere presenti nella pagina.

AWS CLI - apigatewayv2

Il update-domain-namecomando seguente aggiorna un nome di dominio per utilizzare la modalità di routing: ROUTING_RULE_THEN_API_MAPPING

```
aws apigatewayv2 update-domain-name \
  --domain-name 'api.example.com' \
  --routing-mode "ROUTING_RULE_THEN_API_MAPPING"
```

L'output sarà simile al seguente:

```
{
"ApiMappingSelectionExpression": "$request.basepath",
"DomainName": "api.example.com",
"DomainNameArn": "arn:aws:apigateway:us-west-2::/domainnames/api.example.com",
"DomainNameConfigurations": [
  {
      "ApiGatewayDomainName": "d-abcdefg.execute-api.us-west-2.amazonaws.com",
      "CertificateArn": "arn:aws:acm:us-west-2:111122223333:certificate/
abcdefg-123456-abcdefg",
      "DomainNameStatus": "AVAILABLE",
      "EndpointType": "REGIONAL",
      "HostedZoneId": "Z2OJLYMUO9EFXC",
      "SecurityPolicy": "TLS_1_2"
   }
 ],
"RoutingMode": "ROUTING_RULE_THEN_API_MAPPING",
"Tags": {}
}
```

AWS CLI - apigateway

Il update-domain-namecomando seguente aggiorna un nome di dominio personalizzato privato per utilizzare la modalità di routing: ROUTING_RULE_THEN_BASE_PATH_MAPPING

```
aws apigateway update-domain-name \
  --domain-name 'private.example.com' \
  --patch-operations "op='replace',path='/
routingMode',value='ROUTING_RULE_THEN_BASE_PATH_MAPPING'"
```

L'output sarà simile al seguente:

```
{
"domainName": "private.example.com",
"domainNameId": "abcd1234",
"domainNameArn": "arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234",
"certificateArn": "arn:aws:acm:us-west-2:111122223333:certificate/
a1b2c3d4-5678-90ab-cdef",
"certificateUploadDate": "2024-09-10T10:31:20-07:00",
"endpointConfiguration": {
  "types": [
    "PRIVATE"
   ],
  "ipAddressType": "dualstack"
  },
"domainNameStatus": "AVAILABLE",
"securityPolicy": "TLS_1_2",
"policy": "...",
"routingMode" : "ROUTING_RULE_THEN_BASE_PATH_MAPPING"
}
```

Regole di routing per connettere le fasi API a un nome di dominio personalizzato per REST API

Una regola di routing è un insieme di condizioni che, se soddisfatte, invocano un'azione. Ad esempio, una regola può instradare qualsiasi richiesta in entrata per un nome di dominio personalizzato che contiene l'intestazione `Hello:World` e il percorso di base `users` alla fase `production` di una REST API.

Le regole vengono valutate in ordine di priorità; impostando la modalità di routing su `ROUTING_RULE_THEN_API_MAPPING`, Gateway API valuta sempre tutte le regole di routing prima di valutare qualsiasi mappatura API. Di seguito viene illustrato come una regola di routing utilizza condizioni, azioni e priorità.

Condizioni

Quando le condizioni di una regola vengono soddisfatte, l'operazione viene eseguita. Gateway API supporta fino a due condizioni di intestazione e una condizione di percorso. Gateway API valuta insieme le condizioni di intestazione e le condizioni di percorso di base.

È possibile creare una regola senza condizioni. Quando Gateway API valuta questa regola, l'azione viene sempre eseguita. È possibile creare una regola senza condizioni come regola catch-all.

Per ulteriori informazioni sulle condizioni di intestazione, consultare the section called “Corrispondenza delle condizioni di intestazione”. Per ulteriori informazioni sulle condizioni di percorso, consultare the section called “Corrispondenza delle condizioni di percorso di base”.

Azioni

Le azioni sono il risultato della corrispondenza delle condizioni a una regola di routing. Attualmente, l’unica azione supportata consiste nell’invocare una fase di una REST API.

Ogni regola può avere una sola azione.

Priorità

La priorità determina l’ordine in cui vengono valutate le regole, dal valore più basso a quello più alto. Le regole non possono avere la stessa priorità.

È possibile impostare la priorità su un valore compreso tra 1 e 1.000.000. Se una regola ha la priorità pari a uno, Gateway API la valuta per prima. Quando si crea una regola, è consigliabile aggiungere degli spazi vuoti nelle priorità. Questo approccio consente di cambiare la priorità delle regole e aggiungere nuove regole. Per ulteriori informazioni, consulta the section called “Modifica della priorità di una regola di routing”.

Per esempi di come Gateway API valuta le regole di routing, consulta the section called “Esempi di come Gateway API valuta le regole di routing”.

Tipi di condizione delle regole di routing di Gateway API

La sezione seguente illustra i tipi di condizione delle regole di routing. Gateway API soddisfa una regola solo se tutte le condizioni sono vere.

Corrispondenza delle condizioni di intestazione

Quando si crea una condizione di intestazione, è possibile eseguire la corrispondenza al nome dell’intestazione e al valore glob dell’intestazione, ad esempio `Hello:World`. Gateway API utilizza una corrispondenza letterale per convalidare la corrispondenza delle condizioni di intestazione. La condizione può utilizzare fino a due intestazioni specificando AND tra loro. Ad esempio, la condizione può essere soddisfatta se una richiesta in entrata contiene `Hello:World` e `x-version:beta`.

La corrispondenza del nome dell’intestazione non fa distinzione tra maiuscole e minuscole, ma il valore glob dell’intestazione rispetta la distinzione tra maiuscole e minuscole. `Hello:World` corrisponde a `hello:World`, ma non a `Hello:world`.

Per l'elenco dei valori di intestazione con limitazioni, consultare the section called "Restrizioni".

Utilizzo di caratteri jolly con le condizioni di intestazione

È possibile utilizzare i caratteri jolly solo nel valore glob dell'intestazione e il carattere jolly deve essere `*prefix-match`, `suffix-match*` o `*contains*`. La tabella seguente mostra degli esempi su come utilizzare i caratteri jolly per la corrispondenza delle condizioni di intestazione.

Condizioni di intestazione	Richieste che soddisfano la regola di routing	Richieste che non soddisfano la regola di routing
`x-version: a*`	• `x-version: account` • `x-version: alpha`	• `x-version: backup` • `x-version: beta` • `x-version: users`
`x-version: *a`	• `x-version: alpha` • `x-version: beta`	• `x-version: account` • `x-version: backup` • `x-version: users`
`x-version: *a*`	• `x-version: account` • `x-version: alpha` • `x-version: backup` • `x-version: beta`	• `x-version: users`
`x-version: *a*` e `x-version: *b*`	• `x-version: backup` • `x-version: beta`	• `x-version: account` • `x-version: alpha` • `x-version: users`
`x-version: b*` e `x-version: *a`	• `x-version: beta`	• `x-version: account` • `x-version: alpha` • `x-version: backup` • `x-version: users`
`x-version: *`	• `x-version: account` • `x-version: alpha` • `x-version: backup`	Nessuno

Condizioni di intestazione	Richieste che soddisfano la regola di routing	Richieste che non soddisfano la regola di routing
	• `x-version: beta` • `x-version: users`	

Se si creano condizioni per più valori di intestazione, ad esempio `Accept:application/json,text/xml`, è consigliabile utilizzare `*contains*` per le condizioni di intestazione ed evitare di creare condizioni usando la virgola (`,`).

Poiché Gateway API utilizza una corrispondenza letterale per le condizioni di intestazione, le corrispondenze semantiche potrebbero essere instradate in modo diverso. La tabella riportata di seguito mostra la differenza dei risultati delle regole di routing.

Condizioni di intestazione	Richieste che soddisfano la regola di routing	Richieste che non soddisfano la regola di routing
`Accept: *json`	• `Accept:application/json Accept:text/xml`	• `Accept:application/json,text/xml`
`Accept: *json*`	• `Accept:application/json Accept:text/xml` • `Accept:application/json,text/xml`	Nessuno

Corrispondenza delle condizioni di percorso di base

Quando si crea una condizione di percorso di base, se la richiesta in entrata contiene il percorso specificato, la regola viene soddisfatta. La corrispondenza rispetta la distinzione tra maiuscole e minuscole, quindi il percorso `New/Users` non corrisponde a `new/users`.

È possibile creare una condizione di percorso di base per un solo percorso di base.

Per l'elenco delle condizioni di percorso di base con limitazioni, consultare the section called "Restrizioni".

Eliminazione del percorso di base con le condizioni di percorso base

Quando create una condizione del percorso base, potete scegliere di rimuovere il percorso base. Quando si elimina il percorso di base, Gateway API rimuove il percorso di base corrispondente in entrata quando invoca l'API di destinazione. Questo è lo stesso comportamento di quando si utilizza una mappatura API. Se non si elimina il percorso di base, Gateway API inoltra l'intero percorso di base all'API di destinazione. È consigliabile eliminare il percorso di base solo quando si crea nuovamente una mappatura API.

La tabella seguente mostra esempi di come Gateway API valuta la condizione di eliminazione del percorso di base.

Condizione	Eliminazione del percorso di base	Richiesta in entrata	Risultato
Se il percorso di base contiene `PetStoreShopper/dogs`	True	`GET https://example.com/PetStoreShopper/dogs`	Gateway API chiama il metodo `GET` della risorsa `/`.
Se il percorso di base contiene `PetStoreShopper/dogs` .	False	`GET https://example.com/PetStoreShopper/dogs`	Gateway API chiama il metodo `GET` della risorsa `PetStoreShopper/dogs` .
Se il percorso di base contiene `PetStoreShopper`	True	`GET https://example.com/PetStoreShopper/dogs`	Gateway API chiama il metodo `GET` della risorsa `dogs`.
Se il percorso di base contiene `PetStoreShopper`	False	`GET https://example.com/PetStoreShopper/dogs`	Gateway API chiama il metodo `GET` della risorsa `PetStoreShopper/dogs` .
Se il percorso di base contiene `PetStoreShopper`	True	`GET https://example.com/PetStor`	Gateway API chiama il metodo `GET` della risorsa `/` con il parametro della

Condizione	Eliminazione del percorso di base	Richiesta in entrata	Risultato
		`eShopper?birds=available`	stringa di query `birds=available` .
Se il percorso di base contiene `PetStoreShopper`	False	`GET https://example.com/PetStoreShopper?birds=available`	Gateway API chiama il metodo `GET` della risorsa `/PetStoreShopper` con il parametro della stringa di query `birds=available` .

Restrizioni

- L'API di destinazione e il nome di dominio personalizzato devono trovarsi nello stesso AWS account.
- Ogni regola può avere una sola API di destinazione.
- È possibile creare una regola di routing di un nome di dominio personalizzato privato per un'API privata e di un nome di dominio personalizzato pubblico per un'API pubblica. Non è possibile combinare insieme risorse pubbliche e private.
- Se il nome di dominio personalizzato dispone di mappature API su REST API e API HTTP, le regole di routing non sono supportate.
- Il numero massimo per la priorità è 1.000.000.
- Limitazioni di intestazione:
  - Ogni condizione `anyOf` può contenere un solo valore di intestazione.
  - Gli unici caratteri consentiti per i nomi delle intestazioni e i valori globali delle intestazioni sono specificati da RFC 7230, ossia `a-z`, `A-Z`, `0-9` e i seguenti caratteri speciali: `*?-!#$%&'.^_`|~`.
  - È possibile utilizzare un carattere jolly nel valore glob dell'intestazione, ma deve essere `*prefix-match`, `suffix-match*` o `*contains*`. Non è possibile utilizzare `*` all'interno di un valore glob dell'intestazione.
  - I nomi delle intestazioni con caratteri jolly non sono supportati.
  - Il nome dell'intestazione non può contenere più di 40 caratteri.

- Il valore glob dell'intestazione non può contenere più di 128 caratteri.
- Il valore glob dell'intestazione per una corrispondenza infissa non può contenere più di 40 caratteri.
- Le seguenti intestazioni non sono supportate come condizioni:
  - `access-control-*`
  - `apigw-*`
  - `Authorization`
  - `Connection`
  - `Content-Encoding`
  - `Content-Length`
  - `Content-Location`
  - `Forwarded`
  - `Keep-Alive`
  - `Origin`
  - `Proxy-Authenticate`
  - `Proxy-Authorization`
  - `TE`
  - `Trailers`
  - `Transfer-Encoding`
  - `Upgrade`
  - `x-amz-*`
  - `x-amzn-*`
  - `x-apigw-api-id`
  - `X-Forwarded-For`
  - `X-Forwarded-Host`
  - `X-Forwarded-Proto`
  - `x-restAPI`
  - `Via`
- Limitazioni di percorso di base:
  - Il percorso di base non può contenere più di 128 caratteri.

- Il percorso di base deve contenere solo lettere, numeri e i seguenti caratteri: `$-_.+!*'()/`.

  Questi caratteri non sono supportati per le espressioni regolari (regex).
- Il percorso di base non può iniziare né terminare con una barra rovesciata (\).

Esempi di come Gateway API valuta le regole di routing

La sezione seguente illustra quattro esempi di come Gateway API valuta le regole di routing e le mappature API.

Esempio 1: solo regole di routing

In questo esempio, il nome di dominio personalizzato `https://petstore.example.com` ha la modalità di routing impostata su `ROUTING_RULE_ONLY` con le regole di routing e le priorità riportate di seguito.

ID della regola	Priorità	Condizioni	Azione
`abc123`	10	Se la richiesta contiene l'intestazione `Hello:World`	API di destinazione 1
`zzz000`	50	Se la richiesta contiene le intestazioni `Accept:image/webp` e `Pet:Dog-*` e se il percorso di base contiene `PetStoreShopper`	API di destinazione 2
`efg456`	100	Nessuno	API di destinazione 3

La tabella seguente mostra come Gateway API applica le regole di routing precedenti alle richieste di esempio.

Richiesta	API selezionata	Spiegazione
`https://petstore.example.com -h "Hello:World"`	API di destinazione 1	La richiesta soddisfa la regola di routing `abc123`.
`https://petstore.example.com/PetStoreShopper -h "Hello:World", "Pet:Dog-Bella", "Accept:image/webp"`	API di destinazione 1	Gateway API valuta tutte le regole di routing in ordine di priorità. La regola di routing `abc123` ha la priorità assoluta e le condizioni sono soddisfatte, quindi Gateway API invoca API di destinazione 1. Sebbene le condizioni della richiesta corrispondano anche alla regola di routing `zzz000`, Gateway API non valuta nessun'altra regola di routing dopo aver effettuato una corrispondenza.
`https://petstore.example.com/PetStoreShopper -h "Pet:Dog-Bella", "Accept:image/webp"`	API di destinazione 2	La richiesta soddisfa la regola di routing `zzz000`. Questa è una corrispondenza perché la stringa `Pet:Dog-Bella` corrisponde a `Pet:Dog-*`
`https://petstore.example.com/PetStoreShopper -h "Pet:Dog-Bella"`	API di destinazione 3	La richiesta non soddisfa la regola di routing `abc123`. La richiesta non soddisfa la regola di routing `zzz000` perché non sono presenti tutte le intestazioni necessarie. La regola di priorità successiva corrisponde a tutte le richieste

Richiesta	API selezionata	Spiegazione
		in entrata, quindi Gateway API invoca API di destinazione 3.

Esempio 2: regole di routing e mappature API

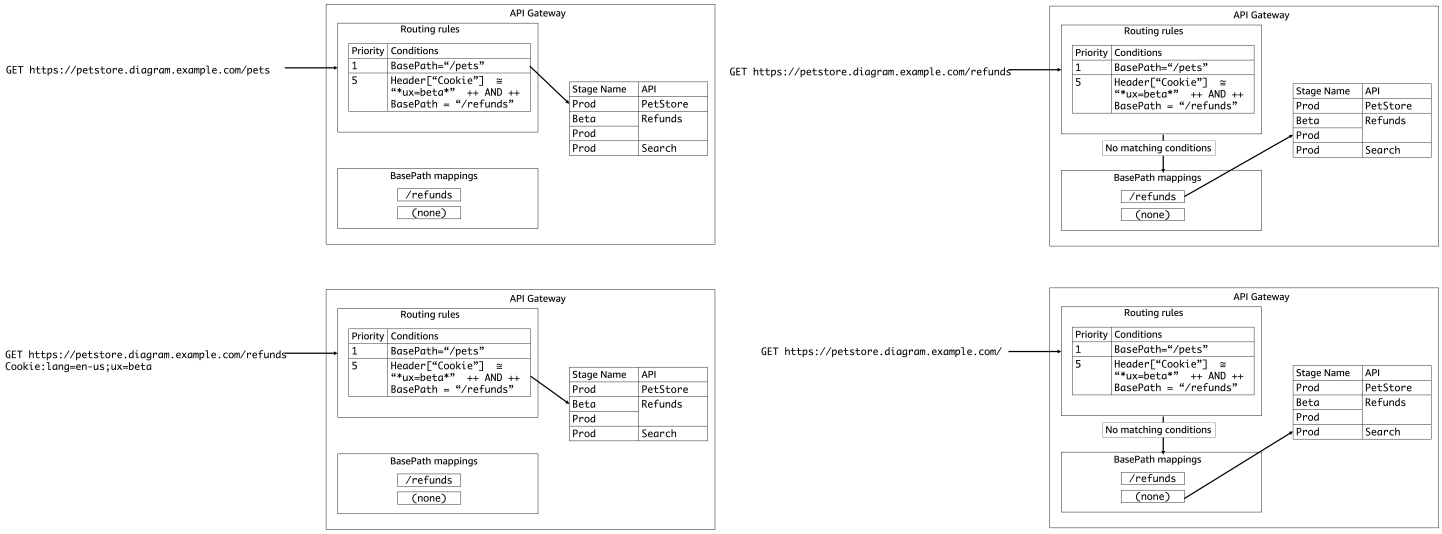
In questo esempio, il nome di dominio personalizzato `https://petstore.diagram.example.com` ha la modalità di routing impostata su `ROUTING_RULE_THEN_API_MAPPING` e le seguenti regole di routing e mappature API.

ID della regola	Priorità	Condizioni	Azione
`abc123`	1	Se la richiesta contiene `pets`	Invoca la fase `Prod` dell'API `PetStore`.
`000zzz`	5	Se la richiesta contiene l'intestazione `Cookie:*ux=beta*` e se il percorso di base contiene `/refunds`	Invoca la fase `Beta` dell'API `Refunds`.

La tabella riportata di seguito mostra le mappature API per `https://petstore.backup.example.com`.

Mappatura API	API selezionata
`/refunds`	Invoca la fase `Prod` dell'API `Refunds`.
`(none)`	Invoca la fase `Prod` dell'API `Search`.

Il diagramma seguente mostra come Gateway API applica le regole di routing e le mappature API precedenti alle richieste di esempio. Le richieste di esempio sono riepilogate nella tabella che segue il diagramma.



La tabella seguente mostra come Gateway API applica le regole di routing e le mappature API precedenti alle richieste di esempio.

Richiesta	API selezionata	Spiegazione
`https://petstore.diagram.com/pets`	La fase `Prod` dell'API `PetStore`.	La richiesta soddisfa la regola di routing `abc123`.
`https://petstore.diagram.example.com/refunds -h "Cookie:lang=en-us;ux=beta"`	La fase `Beta` dell'API `Refunds`.	La richiesta soddisfa la regola di routing `000zzz`. L'intestazione `Cookie` contiene la corrispondenza `*contains*` e la corrispondenza del percorso di base corrette per questa condizione.
`https://petstore.diagram.example.com/refunds`	La fase `Prod` dell'API `Refunds`.	La richiesta non ha le intestazioni necessarie per soddisfare la regola di routing `zzz000`. Se Gateway API non riesce a soddisfare correttamente una regola di routing, utilizza le mappature API. Gateway API può mappare il percorso

Richiesta	API selezionata	Spiegazione
		di base alla fase `Prod` dell'API `Refunds`.
`https://petstore.diagram.example.com/`	La fase `Prod` dell'API `Search`.	La richiesta esegue la corrispondenza della mappatura API al percorso vuoto (`none`).

Esempio 3: regole di routing e mappature API con più livelli

In questo esempio, il nome di dominio personalizzato `https://petstore.backup.example.com` ha la modalità di routing impostata su `ROUTING_RULE_THEN_API_MAPPING` e le seguenti regole di routing e mappature API.

La tabella seguente mostra le regole di routing per `https://petstore.backup.example.com`.

ID della regola	Priorità	Condizioni	Azione
`abc123`	10	Se la richiesta contiene l'intestazione `Hello:World`	API di destinazione 1
`000zzz`	50	Se la richiesta contiene le intestazioni `Accept:image/webp` e `Pet:Dog-*` e se il percorso di base contiene `PetStoreShopper`	API di destinazione 2

La tabella riportata di seguito mostra le mappature API per `https://petstore.backup.example.com`.

Mappatura API	API selezionata
`PetStoreShopper`	API di destinazione 3
`PetStoreShopper/cats`	API di destinazione 4

La tabella seguente mostra come Gateway API applica le regole di routing e le mappature API precedenti alle richieste di esempio.

Richiesta	API selezionata	Spiegazione
`https://petstore.example.com/PetStoreShopper -h "Accept:image/webp", "Pet:Cats"`	API di destinazione 3	La richiesta non ha le intestazioni necessarie per soddisfare la regola di routing `zzz000`. Se Gateway API non riesce a soddisfare correttamente una regola di routing, utilizza le mappature API. Gateway API può mappare il percorso di base all'API di destinazione 3.
`https://petstore.example.com/PetStoreShopper/cats -h "Hello:World"`	API di destinazione 1	La richiesta soddisfa la regola di routing `abc123`. Se la modalità di routing è impostata su `ROUTING_RULE_THEN_API_MAPPING` , le regole di routing hanno sempre la priorità sulle mappature API.
`https://petstore.example.com/Admin -h "Pet:Dog-Bella"`	Nessuno	La richiesta non soddisfa nessuna regola di routing o mappatura API. Poiché non esiste una regola di routing predefinita, Gateway API rifiuta la chiamata e invia al

Richiesta	API selezionata	Spiegazione
		chiamante un codice di stato `403 Forbidden` .

Esempio 4: regole di routing per i nomi di dominio con caratteri jolly

In questo esempio, il nome di dominio personalizzato `https://*.example.com` è un nome di dominio con caratteri jolly. Il carattere jolly supporta tutti i sottodomini che vengono instradati nuovamente allo stesso dominio. Le regole di routing di esempio seguenti modificano questo comportamento per consentire l'instradamento dei sottodomini a diverse API di destinazione utilizzando l'intestazione `Host`.

La tabella seguente mostra le regole di routing per `https://*.example.com`.

ID della regola	Priorità	Condizioni	Azione
`abc123`	10	Se la richiesta contiene l'intestazione `Host:a.example.com`	API di destinazione 1
`000zzz`	50	Se la richiesta contiene l'intestazione `Host:b.example.com`	API di destinazione 2
`efg456`	500	Nessuno	API di destinazione 3

La tabella seguente mostra come Gateway API applica le regole di routing precedenti alle richieste di esempio.

Richiesta	API selezionata	Spiegazione
`https://a.example.com`	API di destinazione 1	L'intestazione `Host` è `a.example.com` . Questa

Richiesta	API selezionata	Spiegazione
		richiesta soddisfa la regola di routing `abc123`.
`https://b.example.com`	API di destinazione 2	L'intestazione `Host` è `b.example.com` . Questa richiesta soddisfa la regola di routing `000zzz`.
`https://testing.example.com`	API di destinazione 3	Questa richiesta soddisfa la regola di routing catch-all `efg456`.

Come utilizzare le regole di routing

È possibile creare una regola di routing utilizzando o qualsiasi AWS SDK. Console di gestione AWS AWS CLI Dopo aver creato una regola, è possibile modificarne la priorità.

Creazione di una regola di routing

La procedura seguente mostra come creare una regola di routing per un nome di dominio personalizzato con la modalità di routing impostata su `ROUTING_RULE_THEN_API_MAPPING` o `ROUTING_RULE_ONLY`.

Console di gestione AWS

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere Custom Domain Names (Nomi di dominio personalizzati) nel riquadro di navigazione principale.
3. Scegliere un nome di dominio personalizzato.
4. Nella scheda Dettagli di routing, scegliere Aggiungi una regola di routing.
5. Scegliere Aggiungi una nuova condizione per aggiungere una nuova condizione.

   È possibile aggiungere una condizione di intestazione o di percorso di base. Per eseguire la corrispondenza di tutte le richieste in entrata al nome di dominio personalizzato, non aggiungere condizioni.
6. Per Azione, selezionare nel menu a discesa l'API e la fase di destinazione.

7. Scegli Next (Successivo).
8. Nel campo Priorità, immettere un numero per la priorità.

   Gateway API valuta le regole in ordine di priorità, dal valore più basso a quello più alto.

   Se si crea una regola senza condizioni, è consigliabile utilizzare una priorità con valore elevato.
9. Scegliere Crea una regola di routing.

AWS CLI

Il comando create-routing-rule seguente crea una regola di routing con la priorità 50. In questo esempio, Gateway API instrada tutte le richieste in entrata che hanno le intestazioni `Hello:World` e `x-version:beta` e il percorso di base `PetStoreShopper` all'API di destinazione `a1b2c3`.

```
aws apigatewayv2 create-routing-rule \
 --domain-name 'api.example.com' \
 --priority 50 \
 --conditions '[
   {
     "MatchHeaders": {
       "AnyOf": [
         {
           "Header": "Hello",
           "ValueGlob": "World"
         }
       ]
     }
   },
   {
     "MatchHeaders": {
       "AnyOf": [
         {
           "Header": "x-version",
           "ValueGlob": "beta"
         }
       ]
     }
   },
   {
```

```
      "MatchBasePaths": {
        "AnyOf": [
          "PetStoreShopper"
        ]
      }
    }
  ]'\
  --actions '[
  {
    "InvokeApi": {
      "ApiId": "a1b2c3",
      "Stage": "prod"
    }
  }
 ]'
```

L'output sarà simile al seguente.

```
{
    "Actions": [
        {
            "InvokeApi": {
                "ApiId": "a1b2c3",
                "Stage": "prod",
                "StripBasePath": false
            }
        }
    ],
    "Conditions": [
        {
            "MatchHeaders": {
                "AnyOf": [
                    {
                        "Header": "Hello",
                        "ValueGlob": "World"
                    }
                ]
            }
        },
        {
            "MatchHeaders": {
                "AnyOf": [
                    {
```

```
                        "Header": "x-version",
                        "ValueGlob": "beta"
                    }
                ]
            }
        },
        {
            "MatchBasePaths": {
                "AnyOf": [
                    "PetStoreShopper"
                ]
            }
        }
    ],
    "Priority": 50,
    "RoutingRuleArn": "arn:aws:apigateway:us-west-2:111122223333:/domainnames/
api.example.com/routingrules/abc123",
    "RoutingRuleId": "abc123"
}
```

Modifica della priorità di una regola di routing

È possibile modificare la priorità di una regola di routing. La modifica ha effetto immediato e potrebbe influire sul modo in cui gli utenti delle API invocano i nomi di dominio personalizzati. Quando si impostano le priorità delle regole di routing, è consigliabile lasciare degli spazi vuoti tra le regole.

Ad esempio, si considerino due regole di routing, una regola `abc123` con la priorità 50 e una regola `zzz000` con la priorità 150. Per modificare la priorità delle regole in modo che Gateway API valuti prima la regola `zzz000`, è possibile impostare la priorità della regola `zzz000` su 30.

Console di gestione AWS

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere Custom Domain Names (Nomi di dominio personalizzati) nel riquadro di navigazione principale.
3. Scegliere un nome di dominio personalizzato.
4. Nella scheda Dettagli di routing, scegliere la regola di routing, quindi scegliere Modifica.
5. Scegli Next (Successivo).
6. Per Priorità, immettere la nuova priorità.

7. Scegli Save changes (Salva modifiche).

AWS CLI

Il comando put-routing-rule seguente modifica la priorità della regola di routing `abc123`.

```
aws apigatewayv2 put-routing-rule \
 --domain-name 'api.example.com' \
 --priority 30 \
 --routing-rule-id abc123 \
 --conditions '[
   {
     "MatchHeaders": {
       "AnyOf": [
         {
           "Header": "Hello",
           "ValueGlob": "World"
         }
       ]
     }
   },
   {
     "MatchHeaders": {
       "AnyOf": [
         {
           "Header": "x-version",
           "ValueGlob": "beta"
         }
       ]
     }
   },
   {
     "MatchBasePaths": {
       "AnyOf": [
         "PetStoreShopper"
       ]
     }
   }
 ]'\
 --actions '[
 {
   "InvokeApi": {
     "ApiId": "a1b2c3",
```

```
      "Stage": "prod"
    }
  }
 ]'
```

L'output sarà simile al seguente:

```
{
    "Actions": [
        {
            "InvokeApi": {
                "ApiId": "a1b2c3",
                "Stage": "prod",
                "StripBasePath": false
            }
        }
    ],
    "Conditions": [
        {
            "MatchHeaders": {
                "AnyOf": [
                    {
                        "Header": "Hello",
                        "ValueGlob": "World"
                    }
                ]
            }
        },
        {
            "MatchHeaders": {
                "AnyOf": [
                    {
                        "Header": "x-version",
                        "ValueGlob": "beta"
                    }
                ]
            }
        },
        {
            "MatchBasePaths": {
                "AnyOf": [
                    "PetStoreShopper"
                ]
```

```
            }
        }
    ],
    "Priority": 38,
    "RoutingRuleArn": "arn:aws:apigateway:us-west-2:111122223333:/domainnames/
api.example.com/routingrules/abc123",
    "RoutingRuleId": "abc123"
}
```

Nuova creazione di una mappatura API utilizzando le regole di routing

È possibile creare nuovamente una mappatura API utilizzando le regole di routing. Per creare nuovamente una mappatura API, è necessario attivare l'eliminazione del percorso di base. Questo approccio preserva il comportamento delle mappature API. Per ulteriori informazioni, consulta the section called "Eliminazione del percorso di base con le condizioni di percorso base".

Il seguente tutorial mostra come creare nuovamente la mappatura API `https://api.example.com/orders/v2/items/categories/5` come regola di routing e come aggiornare i log di accesso per registrare l'ID della regola di routing che Gateway API utilizza per inviare il traffico all'API.

Console di gestione AWS

Per impostare la modalità di routing su ROUTING_RULE_THEN_API_MAPPING

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere Custom Domain Names (Nomi di dominio personalizzati) nel riquadro di navigazione principale.
3. Scegliere il nome di dominio personalizzato.
4. Per Dettagli del dominio, scegliere Modifica.
5. Per Modalità di routing, scegliere ROUTING_RULE_THEN_API_MAPPING.
6. Seleziona Salva

Dopo aver impostato la modalità di routing, si crea la regola di routing.

Per creare la regola di routing

1. Nella scheda Dettagli di routing, scegliere Aggiungi una regola di routing.

2. Scegliere Aggiungi una nuova condizione e quindi scegliere Percorso.
3. Per Percorso, immettere **`orders/v2/items/categories/5`**.
4. Per Percorso base della striscia, scegliere Attivo.
5. Per API di destinazione, scegliere l'API di destinazione desiderata.
6. Per Fase di destinazione, scegliere la fase di destinazione desiderata.
7. Scegli Next (Successivo).
8. Per Priorità, immettere una priorità.

   Anche se si mantiene la mappatura API esistente, Gateway API utilizzerà sempre la nuova regola di routing poiché le regole di routing hanno sempre la priorità sulle mappature API.
9. Scegli Save changes (Salva modifiche).

Dopo aver creato la regola di routing, si aggiorna il formato del log di accesso per la fase o si crea un nuovo log per verificare che Gateway API utilizzi la regola di routing per inviare traffico all'API.

Per aggiornare i log di accesso

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere l'API.
3. Nel riquadro di navigazione principale scegli Fasi.
4. Per Log e tracciamento, scegliere Modifica.

   Se non è disponibile un gruppo di log, consulta the section called "CloudWatch registri".
5. Aggiungere **`$context.customDomain.routingRuleIdMatched`** al formato dei log.

   Questo gruppo di log registra l'ID della regola di routing utilizzato da Gateway API per inviare il traffico all'API. Per ulteriori informazioni, consulta the section called "Impossibile determinare in che modo Gateway API invia il traffico alle API".
6. Scegli Save (Salva).

Dopo aver aggiornato i log di accesso, si invoca del nome di dominio personalizzato. Di seguito è riportato un esempio di comando curl per invocare il nome di dominio personalizzato `https://api.example.com` con il percorso di base `orders/v2/items/categories/5`.

```
curl "https://api.example.com/orders/v2/items/categories/5"
```

Dopo aver richiamato con successo il nome di dominio personalizzato, conferma che CloudWatch Logs mostri il. `routingRuleIdMatched` Per informazioni su come utilizzare la console CloudWatch Logs per visualizzare un gruppo di log, consulta. the section called “Visualizza gli eventi di registro nella CloudWatch console”

AWS CLI

1. Utilizza il seguente comando update-domain-name per aggiornare il nome di dominio *api.example.com* in modo da utilizzare la modalità di routing `ROUTING_RULE_THEN_API_MAPPING`.

```
aws apigatewayv2 update-domain-name \
  --domain-name 'api.example.com' \
  --routing-mode ROUTING_RULE_THEN_API_MAPPING
```

2. Utilizza il seguente comando create-routing-rule per creare una nuova regola di routing al fine di creare nuovamente la mappatura API `https://`*api.example.com*`/orders/v2/items/categories/5`.

```
aws apigatewayv2 create-routing-rule \
  --domain-name 'api.example.com' \
  --priority 50 \
  --conditions '[
  {
    "MatchBasePaths": {
      "AnyOf": [
        "orders/v2/items/categories/5"
      ]
    }
  }
]' \
  --actions '[
  {
    "InvokeApi": {
      "ApiId": "a1b2c3",
      "Stage": "prod",
      "StripBasePath": true
    }
  }
]'
```

3. Utilizza il seguente comando update-stage per aggiornare il formato dei log di accesso in modo da includere la variabile `$context.customDomain.routingRuleIdMatched`. Questa variabile registra l'ID della regola di routing utilizzata da Gateway API per inviare il traffico all'API. Questo log si usa per verificare che Gateway API utilizzi la regola di routing per inviare traffico all'API. Per ulteriori informazioni, consulta the section called "Impossibile determinare in che modo Gateway API invia il traffico alle API".

```
aws apigateway update-stage \
  --rest-api-id a1bc2c3 \
  --stage-name prod \
  --patch-operations "op=replace,path=/accessLogSettings/format,value='\
$context.path \$context.customDomain.routingRuleIdMatched \$context.requestId \
$context.extendedRequestId'"
```

   Se non è disponibile un gruppo di log, consulta the section called "CloudWatch registri".

4. Utilizza il seguente comando curl di esempio per invocare il nome di dominio personalizzato con il percorso di base `orders/v2/items/categories/5`.

```
curl "https://api.example.com/orders/v2/items/categories/5
```

5. Utilizza il seguente comando filter-log-events seguente per ottenere gli eventi dei log dal gruppo di log `access-log-group-orders` che contiene l'ID della regola di routing `abc123`.

```
aws logs filter-log-events --log-group-name access-log-group-orders --filter-
pattern abc123
```

   Questo comando conferma che Gateway API utilizza la regola di routing per inviare traffico all'API.

Risoluzione dei problemi relativi alle regole di routing

La seguente guida può aiutare a risolvere i problemi relativi alle regole di routing.

Impossibile determinare in che modo Gateway API invia il traffico alle API

È possibile utilizzare i log di accesso per la fase della REST API per registrare e risolvere i problemi delle regole di routing. È possibile visualizzare l'ID della regola di routing utilizzato da Gateway API per inviare il traffico all'API usando la variabile

`$context.customDomain.routingRuleIdMatched`. Per visualizzare la mappatura API utilizzata da Gateway API per inviare il traffico all'API, si usa la variabile `$context.customDomain.basePathMatched`.

Per registrare le regole di routing, devi configurare un ARN del ruolo CloudWatch Logs appropriato per il tuo account e creare un gruppo di log.

Il gruppo di log di accesso di esempio seguente può recuperare le informazioni pertinenti per la risoluzione dei problemi relativi alle regole di routing e alle mappature API. Gateway API popola solo la variabile di contesto per il meccanismo di routing utilizzato, in caso contrario la variabile di contesto è `-`.

CLF

```
$context.path $context.customDomain.routingRuleIdMatched
 $context.customDomain.basePathMatched $context.requestId $context.extendedRequestId
```

JSON

```
{"requestPath": "$context.path", "routingRuleId" :
 "$context.customDomain.routingRuleIdMatched", "API mapping" :
 "$context.customDomain.basePathMatched", "requestId":"$context.requestId",
 "extendedRequestId":"$context.extendedRequestId"}
```

XML

```
<request id="$context.requestId"> <requestPath>$context.path</requestPath>
 <ruleId>$context.customDomain.routingRuleIdMatched</ruleId> <ApiMapping>
$context.customDomain.basePathMatched</ApiMapping> <extendedRequestId>
$context.extendedRequestId</extendedRequestId> </request>
```

CSV

```
$context.path,$context.customDomain.routingRuleIdMatched,
$context.customDomain.basePathMatched,$context.requestId,$context.extendedRequestId
```

È consigliabile inoltre verificare la modalità di routing per il nome di dominio personalizzato. Per ulteriori informazioni, consulta the section called "Impostazione della modalità di routing per il nome di dominio personalizzato".

Impossibile abilitare le regole di routing sul nome di dominio personalizzato

È possibile ricevere da Gateway API il seguente errore:

```
Your account doesn’t have permission to use RoutingRules.
This might be caused by an IAM policy in your account with a deny statement on
 BasePathMapping or ApiMapping.
To grant permission for this account to use RoutingRules, use the UpdateAccount API.
This will impact any existing IAM policies that deny access to BasePathMapping or
 ApiMapping.
See API Gateway documentation for further details.
```

Riceverai questo errore se disponi o disponi di una policy IAM che nega l'accesso a o. BasePathMappingApiMapping Quando si abilitano le regole di routing per un nome di dominio personalizzato, anche se la policy continua a negare l’accesso a `BasePathMapping` o `ApiMapping`, è possibile utilizzare la stessa policy per accedere a `RoutingRule`. Ciò consente a un utente di modificare il comportamento di routing del nome di dominio personalizzato.

Ad esempio, se si dispone di una policy simile alla seguente:

```
{
    "Sid": "DenyCreatingApiMappings",
    "Effect": "Deny",
    "Action": "apigateway:POST",
    "Resource": [
        "arn:aws:apigateway:us-west-2::/domainnames/example.com/apimappings"
    ]
}
```

Quando si abilitano le regole di routing per `example.com`, questa policy continua a negare l’accesso alla creazione di `ApiMapping` ma non nega l’accesso alla creazione di `RoutingRule`.

È consigliabile controllare le policy IAM dell’account. La policy di esempio seguente nega l’accesso alla creazione di `ApiMapping`, `BasePathMapping` e `RoutingRule`:

```
{
    "Sid": "DenyCreatingBasePathMappingsApiMappings",
    "Effect": "Deny",
    "Action": "apigateway:POST",
    "Resource": [
        "arn:aws:apigateway:us-west-2::/domainnames/example.com/basepathmappings",
        "arn:aws:apigateway:us-west-2::/domainnames/example.com/apimappings"
```

```
    ]
},
{
    "Sid": "DenyCreatingRoutingRules",
    "Effect": "Deny",
    "Action": "apigateway:CreateRoutingRule",
    "Resource": [
        "arn:aws:apigateway:us-west-2:111122223333:/domainnames/example.com/
routingrules/*"
    ]
}
```

Dopo aver verificato che tutte le policy sono state aggiornate, è possibile aggiornare le impostazioni dell'API a livello di account per abilitare le regole di routing per una Regione.

Utilizza il seguente comando update-account per aggiornare le impostazioni dell'API a livello di account per una Regione:

```
aws apigateway update-account --patch-operations 'op=remove,path=/
features,value=BlockedForRoutingRules' --region us-west-2
```

Dopo aver aggiornato le impostazioni dell'API a livello di account, è possibile modificare la modalità di routing del nome di dominio personalizzato. È anche possibile continuare a utilizzare le policy IAM per negare l'accesso a `RoutingRules`, `ApiMapping` o `BasePathMapping`.

Utilizzo delle mappature API per connettere le fasi API a un nome di dominio personalizzato per REST API

È possibile utilizzare le mappature API per connettere le fasi API a un nome di dominio personalizzato. In questo modo il traffico viene inviato alle API tramite il nome di dominio personalizzato.

Una mappatura API specifica un'API, una fase e, facoltativamente, un percorso da utilizzare per la mappatura. Ad esempio, è possibile mappare `https://api.example.com/orders` alla fase `production` di un'API.

È possibile mappare le fasi HTTP e API REST allo stesso nome di dominio personalizzato.

Prima di creare una mappatura API, è necessario disporre di un'API, di una fase e di un nome di dominio personalizzato. Per ulteriori informazioni sulla creazione di un nome di dominio personalizzato, consulta the section called "Configurazione di un nome di dominio personalizzato regionale".

Richieste in entrata per il nome di dominio personalizzato

Quando si mappa un nome di dominio personalizzato a una fase dell'API, Gateway API elimina il percorso di base in entrata. In tal modo si rimuove il percorso di base mappato dall'invocazione all'API. Ad esempio, se la mappatura percorso di base è `https://api.example.com/orders/shop/5` per la fase `test` e si utilizza la richiesta `https://api.example.com/orders/shop/5/hats`, Gateway API invoca la risorsa `/hats` della fase `test` dell'API, non la risorsa `orders/shop/5/hats`.

Richieste di mappatura API

Di seguito viene illustrato come Gateway API valuta le mappature API.

È possibile creare una mappatura API utilizzando mappature a livello singolo, ad esempio una mappatura API da `orders` alla fase `beta` di un'API e una mappatura API da `shipping` alla fase `alpha` di un'API. Per i nomi di dominio personalizzati regionali con la policy di sicurezza TLS 1.2, Gateway API supporta le mappature API a più livelli. È possibile creare una mappatura API da `orders/v1/items` alla fase `alpha` di un'API e da `orders/v2/items` alla fase `beta` di un'API. Quando si crea una mappatura a più livelli, Gateway API invia le richieste alla mappatura API che ha il percorso corrispondente più lungo.

È possibile creare una mappatura API per il percorso vuoto (`none`). Se nessun percorso corrisponde alla richiesta, Gateway API invia la richiesta al percorso vuoto (`none`).

In questo esempio, il nome di dominio personalizzato `https://api.example.com` ha le seguenti mappature API.

Mappatura API	API selezionata
`(none)`	API 1
`orders`	API 2
`orders/v1/items`	API 3
`orders/v2/items`	API 4
`orders/v1/items/categories`	API 5

La tabella seguente mostra come Gateway API applica le mappature API precedenti alle richieste di esempio.

Richiesta	API selezionata	Spiegazione
`https://api.example.com/orders`	API 2	La richiesta corrisponde esattamente a questa mappatura API.
`https://api.example.com/orders/v1/items`	API 3	La richiesta corrisponde esattamente a questa mappatura API.
`https://api.example.com/orders/v2/items`	API 4	La richiesta corrisponde esattamente a questa mappatura API.
`https://api.example.com/orders/v1/items/123`	API 3	API Gateway sceglie la mappatura con il percorso corrispondente più lungo. `123` alla fine della richiesta non influisce sulla selezione . Consulta the section called “Richieste in entrata per il nome di dominio personalizzato”.
`https://api.example.com/orders/v1/items/categories/5`	API 5	API Gateway sceglie la mappatura con il percorso corrispondente più lungo.
`https://api.example.com/customers`	API 1	API Gateway utilizza la mappatura vuota come catch-all.
`https://api.example.com/ordersandmore`	API 2	API Gateway sceglie la mappatura con il prefisso corrispondente più lungo.

Richiesta	API selezionata	Spiegazione
		Per un nome di dominio personalizzato configurato con mappature a livello singolo, ad esempio solo `https://api.example.com/orders` e `https://api.example.com/`, API Gateway sceglie `API 1`, poiché non esiste un percorso corrispondente con `ordersandmore`.

Restrizioni

- In una mappatura API, il nome di dominio personalizzato e le API mappate devono trovarsi nello stesso account. AWS
- Le mappature API devono contenere solo lettere, numeri e i seguenti caratteri: `$-_.+!*'()/`.
- La lunghezza massima per il percorso in una mappatura API è di 300 caratteri.
- È possibile disporre di 200 mappature API con più livelli per ogni nome di dominio. Questo limite non include le mappature API con livelli singoli, ad esempio `/prod`.
- È possibile mappare le API HTTP a un nome di dominio personalizzato regionale solo con la policy di sicurezza TLS 1.2.
- Non puoi mappare le WebSocket API allo stesso nome di dominio personalizzato di un'API HTTP o un'API REST.
- Dopo aver creato le mappature API, è necessario creare o aggiornare il record di risorse del provider DNS per eseguire la mappatura all'endpoint API.
- Se si crea una mappatura API con più livelli, Gateway API converte tutti i nomi di intestazione in lettere minuscole.

Creare una mappatura API

Per creare una mappatura API, innanzitutto è necessario creare un nome di dominio personalizzato, un'API e una fase. Il nome di dominio personalizzato deve avere la modalità di routing impostata su

`ROUTING_RULE_THEN_API_MAPPING` o `API_MAPPING_ONLY`. Per informazioni su come impostare la modalità di routing, consultare the section called “Impostazione della modalità di routing per il nome di dominio personalizzato”.

Ad esempio, AWS Serverless Application Model modelli che creano tutte le risorse, vedi Sessions With SAM on GitHub.

Console di gestione AWS

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere Custom Domain Names (Nomi di dominio personalizzati) nel riquadro di navigazione principale.
3. Scegliere un nome di dominio personalizzato.
4. Nella scheda Dettagli di routing, scegliere Configura delle mappature API.
5. Specifica API, Fase e Percorso per la mappatura.
6. Selezionare Salva.

AWS CLI

Il comando create-api-mapping seguente crea una mappatura API. In questo esempio, API Gateway invia le richieste `api.example.com/v1/orders` all'API e alla fase specificate.

> **Note**
>
> Per creare mappature API con più livelli, è necessario utilizzare `apigatewayv2`.

```
aws apigatewayv2 create-api-mapping \
    --domain-name api.example.com \
    --api-mapping-key v1/orders \
    --api-id a1b2c3d4 \
    --stage test
```

CloudFormation

L' CloudFormation esempio seguente crea una mappatura API.

> **Note**
>
> Per creare mappature API con più livelli, è necessario utilizzare `AWS::ApiGatewayV2`.

```
MyApiMapping:
  Type: 'AWS::ApiGatewayV2::ApiMapping'
  Properties:
    DomainName: api.example.com
    ApiMappingKey: 'orders/v2/items'
    ApiId: !Ref MyApi
    Stage: !Ref MyStage
```

Tipi di indirizzi IP per nomi di dominio personalizzati in API Gateway

Quando crei un nome di dominio personalizzato, specifichi il tipo di indirizzi IP che possono richiamare il tuo dominio. Puoi scegliere di risolvere IPv4 gli indirizzi IPv4 per richiamare il tuo dominio oppure puoi scegliere dualstack per consentire sia IPv4 agli indirizzi che agli IPv6 indirizzi di richiamare il tuo dominio. Ti consigliamo di impostare il tipo di indirizzo IP su dualstack per ridurre l'esaurimento dello spazio IP o per il tuo livello di sicurezza. Per ulteriori informazioni sui vantaggi di un tipo di indirizzo IP dualstack, consulta on. IPv6 AWS

È possibile modificare il tipo di indirizzo IP aggiornando la configurazione dell'endpoint del nome di dominio.

Considerazioni sui tipi di indirizzi IP

Le seguenti considerazioni potrebbero influire sull'utilizzo dei tipi di indirizzi IP.

- Il tipo di indirizzo IP predefinito per i nomi di dominio personalizzati per il pubblico di API Gateway APIs è IPv4.
- I nomi di dominio personalizzati privati possono avere solo un tipo di indirizzo IP dualstack.
- Non è necessario che il nome di dominio personalizzato abbia lo stesso tipo di indirizzo IP per tutti gli indirizzi APIs mappati su di esso. Se disabiliti l'endpoint API predefinito, ciò potrebbe influire sul modo in cui i chiamanti possono richiamare il tuo dominio.

Modifica il tipo di indirizzo IP di un nome di dominio personalizzato

È possibile modificare il tipo di indirizzo IP aggiornando la configurazione dell'endpoint del nome di dominio. È possibile aggiornare la configurazione dell'endpoint utilizzando il Console di gestione AWS AWS CLI CloudFormation, o un AWS SDK.

Console di gestione AWS

Per modificare il tipo di indirizzo IP di un nome di dominio personalizzato

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli un nome di dominio pubblico personalizzato.
3. Scegli la configurazione dell'endpoint.
4. Per il tipo di indirizzo IP, seleziona uno IPv4o Dualstack.
5. Scegli Save (Salva).

AWS CLI

Il update-domain-namecomando seguente aggiorna un'API in modo che abbia un tipo di indirizzo IP dualstack:

```
aws apigateway update-domain-name \
    --domain-name dualstack.example.com \
    --patch-operations "op='replace',path='/endpointConfiguration/
ipAddressType',value='dualstack'"
```

L'output sarà simile al seguente:

```
{
    "domainName": "dualstack.example.com",
    "certificateUploadDate": "2025-02-04T14:46:10-08:00",
    "regionalDomainName": "d-abcd1234.execute-api.us-east-1.amazonaws.com",
    "regionalHostedZoneId": "Z3LQWSYCGH4ADY",
    "regionalCertificateArn": "arn:aws:acm:us-east-1:111122223333:certificate/
a1b2c3d4-5678-90ab-cdef",
    "endpointConfiguration": {
        "types": [
            "REGIONAL"
        ],
```

```
        "ipAddressType": "dualstack"
    },
    "domainNameStatus": "AVAILABLE",
    "securityPolicy": "TLS_1_2",
    "tags": {}
}
```

Scegli una politica di sicurezza per il tuo dominio personalizzato in API Gateway

Una policy di sicurezza è una combinazione predefinita di versione TLS minima e suite di crittografia offerta da Gateway API. Quando i client stabiliscono un handshake TLS sull'API o sul nome di dominio personalizzato, la policy di sicurezza applica la versione di TLS e il pacchetto di crittografia accettato da API Gateway. Le policy di sicurezza proteggono le API e i nomi di dominio personalizzati da problemi di sicurezza della rete, ad esempio manomissioni e eavesdropping tra un client e un server.

API Gateway supporta policy di sicurezza legacy e policy di sicurezza avanzate. `TLS_1_0`e `TLS_1_2` sono politiche di sicurezza obsolete. Utilizza queste politiche di sicurezza per carichi di lavoro generalizzati o per iniziare a creare un'API. Qualsiasi politica che inizia con `SecurityPolicy_` è una politica di sicurezza avanzata. Utilizza queste policy per carichi di lavoro regolamentati, governance avanzata o per utilizzare la crittografia post-quantistica. Quando utilizzi una policy di sicurezza avanzata, devi anche impostare la modalità di accesso agli endpoint per una governance aggiuntiva. Per ulteriori informazioni, consulta the section called “Modalità di accesso agli endpoint”.

Considerazioni

Di seguito sono riportate le considerazioni relative alle politiche di sicurezza per i nomi di dominio personalizzati per le API REST in API Gateway:

- Non è possibile abilitare il TLS reciproco su un nome di dominio che utilizza una politica di sicurezza avanzata.
- Non è possibile mappare un'API HTTP a un nome di dominio che utilizza una politica di sicurezza avanzata.
- Se abiliti la mappatura del percorso di base a più livelli su un'API REST che utilizza una politica di sicurezza avanzata, non puoi creare una mappatura del percorso di base su un'API HTTP per lo stesso nome di dominio.
- La tua API può essere mappata su un nome di dominio personalizzato con una politica di sicurezza diversa dalla tua API. Quando richiami quel nome di dominio personalizzato, API Gateway utilizza

la politica di sicurezza del dominio personalizzato per negoziare l'handshake TLS. Se si disabilita l'endpoint API predefinito, si potrebbe influire sul modo in cui i chiamanti possono invocare l'API.

- API Gateway supporta le policy di sicurezza su tutte le API. Tuttavia, puoi scegliere solo una policy di sicurezza per le API REST. API Gateway supporta solo la politica `TLS_1_2` di sicurezza per HTTP o WebSocket API.
- API Gateway non supporta l'aggiornamento di una policy di sicurezza per un nome di dominio con più tipi di endpoint. Se disponi di più tipi di endpoint per un nome di dominio, eliminane uno per aggiornare la politica di sicurezza.

In che modo API Gateway applica le politiche di sicurezza

L'esempio seguente mostra come API Gateway applica le policy di `SecurityPolicy_TLS13_1_3_2025_09` sicurezza utilizzando la policy di sicurezza come esempio.

La politica `SecurityPolicy_TLS13_1_3_2025_09` di sicurezza accetta il traffico TLS 1.3 e rifiuta il traffico TLS 1.2 e TLS 1.0. Per il traffico TLS 1.3, la politica di sicurezza accetta le seguenti suite di crittografia:

- `TLS_AES_128_GCM_SHA256`
- `TLS_AES_256_GCM_SHA384`
- `TLS_CHACHA20_POLY1305_SHA256`

API Gateway non accetta altre suite di crittografia. Ad esempio, la politica di sicurezza rifiuterebbe qualsiasi traffico TLS 1.3 che utilizza la suite di crittografia. `AES128-SHA`

Per monitorare il protocollo TLS e i client di crittografia utilizzati per accedere al tuo API Gateway, puoi utilizzare le variabili `$context.tlsVersion` e di `$context.cipherSuite` contesto nei tuoi log di accesso. Per ulteriori informazioni, consulta the section called “Monitoraggio”.

Per visualizzare le politiche di sicurezza predefinite per tutte le API REST e i nomi di dominio personalizzati, consulta. the section called “Politiche di sicurezza predefinite” Per visualizzare le politiche di sicurezza supportate per tutte le API REST e i nomi di dominio personalizzati, consulta. the section called “Policy di sicurezza supportate”

Modifica la politica di sicurezza del tuo nome di dominio personalizzato

Se modifichi la politica di sicurezza, il completamento dell'aggiornamento richiede circa 15 minuti. Puoi monitorare il tuo nome `lastUpdateStatus` di dominio personalizzato. Man mano che il tuo nome di dominio personalizzato `lastUpdateStatus` viene aggiornato, lo sarà `PENDING` e quando sarà completato, lo sarà`AVAILABLE`.

Quando utilizzi una politica di sicurezza che inizia con`SecurityPolicy_`, devi anche attivare la modalità di accesso agli endpoint. Per ulteriori informazioni, consulta the section called “Modalità di accesso agli endpoint”.

Console di gestione AWS

Per modificare la politica di sicurezza di un nome di dominio personalizzato

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli un nome di dominio personalizzato che invii il traffico alle API REST.

   Assicurati che ci sia un solo tipo di endpoint associato al tuo nome di dominio personalizzato.
3. Scegli Impostazioni personalizzate del nome di dominio, quindi scegli Modifica.
4. Per Politica di sicurezza, seleziona una nuova politica.
5. Per la modalità di accesso agli endpoint, scegli Strict.
6. Scegli Save changes (Salva modifiche).

AWS CLI

Il seguente comando update-domain-name aggiorna un nome di dominio per utilizzare la politica di sicurezza: `SecurityPolicy_TLS13_1_3_2025_09`

```
aws apigateway update-domain-name \
    --domain-name example.com \
    --patch-operations '[
        {
            "op": "replace",
            "path": "/securityPolicy",
            "value": "SecurityPolicy_TLS13_1_3_2025_09"
        },
        {
            "op": "replace",
```

```
            "path": "/endpointAccessMode",
            "value": "STRICT"
        }
    ]'
```

L'output sarà simile al seguente:

```
{
   "domainName": "example.com",
   "endpointConfiguration": {
      "types": [ "REGIONAL" ],
      "ipAddressType": "dualstack"
   },
   "regionalCertificateArn": "arn:aws:acm:us-west-2:111122223333:certificate/
a1b2c3d4-5678-90ab-cdef",
   "securityPolicy": "SecurityPolicy_TLS13_1_3_2025_09",
   "endpointAccessMode": "STRICT"
}
```

Informazioni su API e API HTTP WebSocket

Per ulteriori informazioni sulle API e le API HTTP, WebSocket consulta e. the section called “Politica di sicurezza per HTTP APIs in API Gateway” the section called “Politica di sicurezza per WebSocket APIs API Gateway”

Disabilitazione dell'endpoint predefinito per REST API

Per impostazione predefinita, i client possono richiamare l'API utilizzando l'endpoint `execute-api` generato da API Gateway per l'API. Per garantire che i client possano accedere all'API solo utilizzando un nome di dominio personalizzato con l'autenticazione TLS reciproca, disattivare l'endpoint `execute-api` predefinito. I client possono comunque connettersi all'endpoint predefinito, ma riceveranno un codice di stato `403 Forbidden`. La disabilitazione dell’endpoint predefinito influisce su tutte le fasi dell’API. Questa impostazione ha effetto quando si aggiorna un'impostazione in qualsiasi fase, ad esempio l'aggiornamento della distribuzione sullo stage.

La procedura seguente mostra come disabilitare l'endpoint predefinito per una REST API.

Console di gestione AWS

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.

2. Scegliere una REST API.
3. Nel pannello di navigazione principale scegli Impostazioni API.
4. Scegliere un'API.
5. In Dettagli API seleziona Modifica.
6. Per Endpoint predefinito seleziona Inattivo.
7. Scegli Save changes (Salva modifiche).
8. Nel pannello di navigazione principale scegli Risorse.
9. Seleziona Deploy API (Distribuisci API).
10. Implementa nuovamente l'API in una fase o aggiorna un'impostazione su una fase per rendere effettivo l'aggiornamento.

AWS CLI

Il comando update-rest-api seguente disabilita l'endpoint predefinito:

```
aws apigateway update-rest-api \
    --rest-api-id abcdef123 \
    --patch-operations op=replace,path=/disableExecuteApiEndpoint,value='True'
```

Dopo aver disabilitato l'endpoint predefinito, è necessario distribuire l'API per rendere effettiva la modifica.

Il comando create-deployment seguente crea un'implementazione e la associa a una fase:

```
aws apigateway create-deployment \
    --rest-api-id abcdef123 \
    --stage-name dev
```

Configurazione dei controlli dell'integrità personalizzati per failover DNS per un'API di Gateway API

Puoi utilizzare i controlli di integrità di Amazon Route 53 per controllare il failover DNS da un'API API Gateway in una regione primaria Regione AWS a una in una regione secondaria. Questo può aiutare a mitigare gli impatti in caso di problema regionale. Se si utilizza un dominio personalizzato, è possibile eseguire il failover senza richiedere ai client di modificare gli endpoint API.

Quando si sceglie Evaluate Target Health (Valutazione dello stato target) per un record alias, tali record non riescono solo quando il servizio API Gateway non è disponibile nella regione. In alcuni casi, il tuo API Gateway APIs può subire interruzioni prima di quel momento. Per controllare direttamente il failover DNS, configura i controlli di integrità personalizzati di Route 53 per il tuo API Gateway. APIs Per questo esempio, si utilizza un CloudWatch allarme che aiuta gli operatori a controllare il failover DNS. Per altri esempi e altre considerazioni sulla configurazione del failover, consulta Creazione di meccanismi di disaster recovery utilizzando Route 53 ed Esecuzione dei controlli di integrità di Route 53 su risorse private in un VPC con e. AWS Lambda CloudWatch

Argomenti

- Prerequisiti
- Fase 1: Configurazione delle risorse
- Fase 2: Avvio del failover nella regione secondaria
- Fase 3: Test del failover
- Fase 4: Ritorno alla regione primaria
- Fasi successive: personalizzare e verificare periodicamente

Prerequisiti

Per completare questa procedura, è necessario creare e configurare le risorse seguenti:

- Un nome di dominio di cui si è proprietari.
- Un certificato ACM per quel nome di dominio in due. Regioni AWS Per ulteriori informazioni, consulta the section called “Prerequisiti per i nomi di dominio personalizzati”.
- Una zona ospitata Route 53 per il nome di dominio in uso. Per ulteriori informazioni, consulta Utilizzo delle zone ospitate nella Guida per gli sviluppatori di Amazon Route 53.

Per ulteriori informazioni su come creare record DNS di failover Route 53 per i nomi di dominio, consulta Scelta una policy di instradamento nella Guida per sviluppatori Amazon Route 53. Per ulteriori informazioni su come monitorare un CloudWatch allarme, consulta Monitoring a CloudWatch alarm nella Amazon Route 53 Developer Guide.

Fase 1: Configurazione delle risorse

In questo esempio, vengono create le seguenti risorse per configurare il failover DNS per il nome di dominio in uso:

- API Gateway APIs in due Regioni AWS
- Nomi di dominio personalizzati API Gateway con lo stesso nome in due Regioni AWS
- Mappature API Gateway che collegano l'API Gateway APIs ai nomi di dominio personalizzati
- Record DNS di failover di Route 53 per i nomi di dominio
- Un CloudWatch allarme nella regione secondaria
- Un controllo dello stato di salute della Route 53 basato sull' CloudWatch allarme nella regione secondaria

Innanzitutto, accertarsi di disporre di tutte le risorse richieste nelle regioni primaria e secondaria. La regione secondaria deve contenere l'allarme e il controllo dell'integrità. In questo modo, non si dipende dalla regione primaria per eseguire il failover. Ad esempio, CloudFormation i modelli che creano queste risorse, vedi `primary.yaml`e `secondary.yaml`.

> **Important**
>
> Prima del failover nella regione secondaria, accertarsi che tutte le risorse necessarie siano disponibili. In caso contrario, l'API non sarà pronta per il traffico nella regione secondaria.

Fase 2: Avvio del failover nella regione secondaria

Nell'esempio seguente, la regione di standby riceve una CloudWatch metrica e avvia il failover. Utilizziamo una metrica personalizzata che richiede l'intervento dell'operatore per avviare il failover.

```
aws cloudwatch put-metric-data \
    --metric-name Failover \
    --namespace HealthCheck \
    --unit Count \
    --value 1 \
    --region us-west-1
```

Sostituisci i dati metrici con i dati corrispondenti per l'allarme che hai configurato. CloudWatch

Fase 3: Test del failover

Richiamare l'API e verificare di ricevere una risposta dalla regione secondaria. Se si sono utilizzati i modelli di esempio della fase 1, la risposta cambia da `{"message": "Hello from the primary Region!"}` a `{"message": "Hello from the secondary Region!"}` dopo il failover.

```
curl https://my-api.example.com

{"message": "Hello from the secondary Region!"}
```

Fase 4: Ritorno alla regione primaria

Per tornare alla regione principale, invia una CloudWatch metrica che determini il superamento del controllo sanitario.

```
aws cloudwatch put-metric-data \
    --metric-name Failover \
    --namespace HealthCheck \
    --unit Count \
    --value 0 \
    --region us-west-1
```

Sostituisci i dati metrici con i dati corrispondenti per l' CloudWatch allarme che hai configurato.

Richiamare l'API e verificare di ricevere una risposta dalla regione primaria. Se si sono utilizzati i modelli di esempio della fase 1, la risposta cambia da `{"message": "Hello from the secondary Region!"}` a `{"message": "Hello from the primary Region!"}`.

```
curl https://my-api.example.com

{"message": "Hello from the primary Region!"}
```

Fasi successive: personalizzare e verificare periodicamente

Questo esempio dimostra un modo per configurare il failover DNS. È possibile utilizzare una varietà di CloudWatch metriche o endpoint HTTP per i controlli di integrità che gestiscono il failover. Verificare periodicamente i meccanismi di failover per accertarsi che funzionino come previsto e che gli operatori conoscano le procedure di failover.

Ottimizzazione delle prestazioni di REST APIs

Dopo aver reso disponibile la tua API per essere chiamata, potrebbe essere necessario ottimizzarla per renderla più reattività. API Gateway fornisce alcune strategie per ottimizzare l'API, come il caching delle risposte e la compressione del payload. In questa sezione viene descritto come abilitare queste funzionalità.

Argomenti

- Impostazioni della cache per REST API in Gateway API
- Compressione del payload per REST APIs in API Gateway

Impostazioni della cache per REST API in Gateway API

Puoi abilitare il caching delle API in Gateway API per memorizzare nella cache le risposte dell'endpoint. Con il caching, è possibile ridurre il numero di chiamate effettuate all'endpoint e migliorare la latenza delle richieste all'API.

Quando abiliti il caching per una fase, le risposte dall'endpoint vengono memorizzate da API Gateway nella cache per un periodo TTL (time-to-live) specificato espresso in secondi. Per rispondere alla richiesta, API Gateway ricerca la risposta dell'endpoint nella cache anziché effettuare una richiesta all'endpoint. Il valore predefinito di TTL per il caching dell'API è 300 secondi. Il valore massimo di TTL è 3600 secondi. TTL=0 indica che il caching è disabilitato.

> **Note**
>
> Il caching è il miglior tentativo. Puoi utilizzare le `CacheMissCount` metriche `CacheHitCount` e in Amazon CloudWatch per monitorare le richieste che API Gateway fornisce dalla cache delle API.

La dimensione massima di una risposta che può essere memorizzata nella cache è 1048576 byte. la crittografia dei dati della cache può aumentare le dimensioni della risposta quando viene memorizzata nella cache.

Questo è un servizio idoneo ai fini HIPAA. Per ulteriori informazioni su AWS, U.S. L'Health Insurance Portability and Accountability Act del 1996 (HIPAA) e l'utilizzo di AWS servizi per elaborare, archiviare e trasmettere informazioni sanitarie protette (PHI), vedere Panoramica HIPAA.

> **Important**
>
> Quando abiliti il caching per una fase, il sistema di caching sarà abilitato per impostazione predefinita solo per i metodi `GET`. Ciò consente di garantire la sicurezza e la disponibilità dell'API. Puoi abilitare il caching per altri metodi ignorando le impostazioni del metodo.

> **Important**
>
> Per il caching viene applicato un addebito orario in base alle dimensioni della cache selezionata. Il caching non è idoneo per il piano gratuito. AWS Per ulteriori informazioni, consulta Prezzi di API Gateway.

Abilitazione del caching di Amazon API Gateway

In Gateway API è possibile abilitare il caching per una fase specifica.

Quando abiliti il caching, devi scegliere una capacità di cache. In generale, una capacità più ampia garantisce prestazioni migliori ma comporta costi più alti. Per le dimensioni della cache supportate, consulta la sezione cache ClusterSize nell'API Gateway API Reference.

Il caching in API Gateway è consentito mediante la creazione di un'istanza di cache dedicata. Questo processo può richiedere fino a 4 minuti.

La capacità di caching può essere modificata in API Gateway rimuovendo l'istanza di cache esistente e creandone una nuova con una capacità modificata. Tutti i dati esistenti memorizzati nella cache vengono eliminati.

> **Note**
>
> La capacità della cache influisce sulla CPU, sulla memoria e sulla larghezza di banda della rete dell'istanza della cache. Di conseguenza, la capacità della cache può influire sulle prestazioni della cache.
> API Gateway consiglia di eseguire un test di caricamento di 10 minuti per verificare che la capacità della cache sia appropriata per il carico di lavoro. Assicurati che il traffico durante il test di carico rispecchi il traffico di produzione. Ad esempio, includi un aumento graduale, un traffico costante e picchi di traffico. Il test di caricamento deve includere risposte che possono essere fornite dalla cache, nonché risposte univoche che aggiungono elementi alla cache. Durante il test di caricamento, monitora i parametri di latenza, 4xx, 5xx, hit della cache e mancato riscontro nella cache. In base a questi parametri, regola la capacità della cache a seconda delle esigenze. Per ulteriori informazioni sul test di carico, consulta Come faccio a selezionare la migliore capacità di Amazon API Gateway Cache per evitare di raggiungere un limite di velocità?.

Console di gestione AWS

Nella console Gateway API, è possibile configurare il caching nella pagina Fasi. Viene effettuato il provisioning della cache della fase desiderata e specificata un'impostazione di cache predefinita a livello di metodo. Attivando la cache predefinita a livello di metodo, il caching a livello di metodo viene attivato per tutti i metodi `GET` nella fase, a meno che il metodo in questione non disponga di una sostituzione dedicata. Tutti i metodi `GET` aggiuntivi che implementi nella fase disporranno di una cache a livello di metodo. Per configurare l'impostazione del caching a livello di metodo per metodi specifici nella fase, è possibile utilizzare le sostituzioni dei metodi. Per ulteriori informazioni sulle sostituzioni dei metodi, consulta the section called “Come ignorare il caching di fase per il caching dei metodi”.

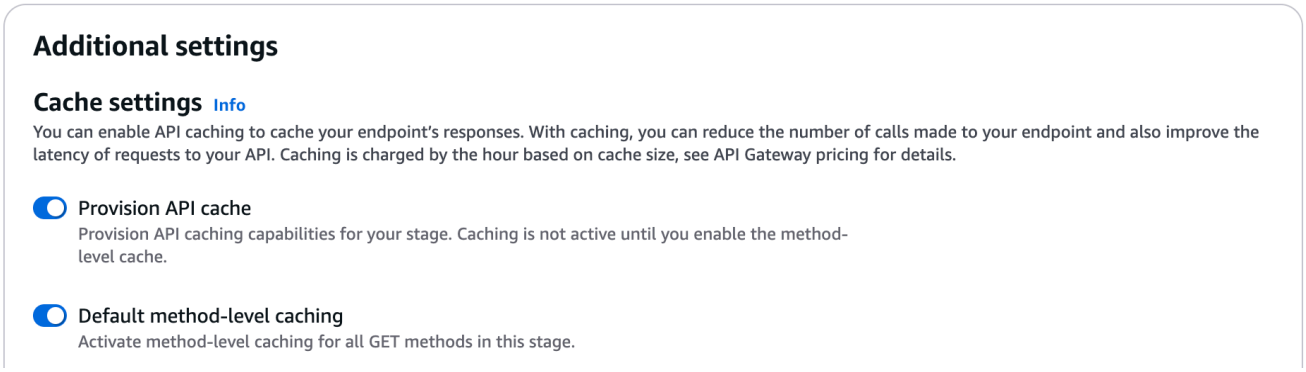
Per configurare il caching dell'API per una fase specifica:

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere Stages (Fasi).
3. Nell'elenco Stages (Fasi) dell'API, selezionare la fase.
4. Nella sezione Dettagli fase scegli Modifica.
5. In Impostazioni aggiuntive, per Impostazioni cache attiva Cache API con provisioning.

   In questo modo viene effettuato il provisioning di un cluster di cache per la fase.
6. Per attivare il caching per la fase, attiva Memorizzazione nella cache predefinita a livello di metodo.

   In questo modo viene attivato il caching a livello di metodo per tutti i metodi `GET` nella fase. Tutti i metodi `GET` aggiuntivi che implementi nella fase disporranno di una cache a livello di metodo.

> **Note**
>
> Se hai un'impostazione esistente per una cache a livello di metodo, la modifica dell'impostazione del caching predefinito a livello di metodo non influisce sull'impostazione esistente.



7. Scegli Save changes (Salva modifiche).

AWS CLI

Il comando update-stage seguente aggiorna una fase per allocare una cache e attiva il caching a livello di metodo per tutti i metodi GET della fase:

```
aws apigateway update-stage \
    --rest-api-id a1b2c3 \
    --stage-name 'prod' \
    --patch-operations file://patch.json
```

Il contenuto di `patch.json` è il seguente:

```
[
     {
          "op": "replace",
          "path": "/cacheClusterEnabled",
          "value": "true"
     },
     {
          "op": "replace",
          "path": "/cacheClusterSize",
          "value": "0.5"
     },
     {
        "op": "replace",
        "path": "/*/*/caching/enabled",
        "value": "true"
     }
```

```
]
```

> **Note**
>
> Se hai un'impostazione esistente per una cache a livello di metodo, la modifica dell'impostazione del caching predefinito a livello di metodo non influisce sull'impostazione esistente.

> **Note**
>
> Per completare la creazione o l'eliminazione di una cache, API Gateway impiega circa 4 minuti.
> Una volta creata la cache, il valore Cluster di cache cambia da `Create in progress` a `Active`. Quando invece viene completata l'eliminazione della cache, il valore Cluster di cache cambia da `Delete in progress` a `Inactive`.
> Quando attivi il caching a livello di metodo per tutti i metodi della fase, il valore Memorizzazione nella cache predefinita a livello di metodo cambia in `Active`. Se disattivi il caching a livello di metodo per tutti i metodi della fase, il valore Memorizzazione nella cache predefinita a livello di metodo cambia in `Inactive`. Se hai un'impostazione esistente per una cache a livello di metodo, la modifica dello stato della cache non influisce su tale impostazione.

Se abiliti il caching in Impostazioni cache di una fase, vengono memorizzati nella cache solo i metodi `GET`. Per garantire la sicurezza e la disponibilità dell'API, ti consigliamo di non modificare questa impostazione. Tuttavia, puoi abilitare il caching per altri metodi ignorando le impostazioni del metodo.

Puoi verificare che il caching funzioni come previsto in due modi:

- Controlla le CloudWatch metriche di CacheHitCounte CacheMissCountper la tua API e il tuo stage.
- Inserisci un timestamp nella risposta.

> **Note**
>
> Non utilizzare l'`X-Cache`intestazione della CloudFront risposta per determinare se l'API viene fornita dall'istanza della cache di API Gateway.

Sostituzione del caching a livello di fase di Gateway API per il caching a livello di metodo

Puoi ignorare le impostazioni della cache a livello di fase attivando o disattivando il caching per un metodo specifico. È possibile anche modificare il periodo TTL oppure attivare o disattivare la crittografia per le risposte memorizzate nella cache.

Se si prevede che nelle risposte di un metodo che si sta memorizzando nella cache siano inclusi dati sensibili, crittografare i dati della cache. Potrebbe essere necessario eseguire questa operazione per rispettare vari framework di conformità. Per ulteriori informazioni, consultare Amazon API Gateway controls nella Guida per l'utente di AWS Security Hub .

Console di gestione AWS

La modifica dell'impostazione del caching predefinito a livello di metodo nei dettagli della fase non influisce sulle impostazioni della cache a livello di metodo che dispongono di sostituzioni.

Se prevedi che nelle risposte di un metodo che si sta memorizzando nella cache siano inclusi dati sensibili, in Cache Settings (Impostazioni cache), scegliere Encrypt cache data (Crittografa dati cache).

Per configurare il caching dell'API per i singoli metodi utilizzando la console:

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Selezionare l'API.
3. Scegliere Stages (Fasi).
4. Nell'elenco Stages (Fasi) dell'API, espandere la fase e scegliere un metodo nell'API.
5. Nella sezione Sostituzioni del metodo, scegli Modifica.
6. Nella sezione Impostazioni metodo, attiva o disattiva Abilita cache metodo o personalizza le altre opzioni desiderate.

> **Note**
>
> Il caching non è attivo finché non si effettua il provisioning di un cluster di cache per la fase.

7. Scegli Save (Salva).

AWS CLI

Il comando update-stage seguente disattiva la cache solo per il metodo `GET /pets`:

```
aws apigateway update-stage /
    --rest-api-id a1b2c3 /
    --stage-name 'prod' /
    --patch-operations file://patch.json
```

Il contenuto di `patch.json` è il seguente:

```
[{
        "op": "replace",
        "path": "/~1pets/GET/caching/enabled",
        "value": "false"
}]
```

Uso dei parametri di metodo o di integrazione come chiavi di cache per indicizzare le risposte memorizzate nella cache

Puoi utilizzare parametri di metodo o di integrazione come chiavi di cache per indicizzare le risposte memorizzate nella cache Sono inclusi: intestazioni personalizzate, percorsi URL o stringhe di query. È possibile specificare alcuni o tutti i parametri come chiave di cache, ma è necessario specificare almeno un valore. Se disponi di una chiave di cache, Gateway API memorizza nella cache le risposte di ciascun valore di chiave separatamente, anche quando la chiave di cache non è presente.

> **Note**
>
> Le chiavi di cache sono necessarie per la configurazione del caching su una risorsa.

Ad esempio, supponiamo di avere una richiesta espressa nel seguente formato:

```
GET /users?type=... HTTP/1.1
host: example.com
...
```

In questa richiesta `type` può assumere il valore `admin` o `regular`. Se includi il parametro `type` come parte della chiave di cache, le risposte da `GET /users?type=admin` vengono memorizzate nella cache separatamente da quelle di `GET /users?type=regular`.

Quando una richiesta di metodo o di integrazione usa più di un parametro, puoi scegliere di includere alcuni di essi o tutti per creare la chiave di cache. Ad esempio puoi includere solo il parametro `type` nella chiave di cache per la richiesta seguente, eseguita nell'ordine elencato in un periodo TTL:

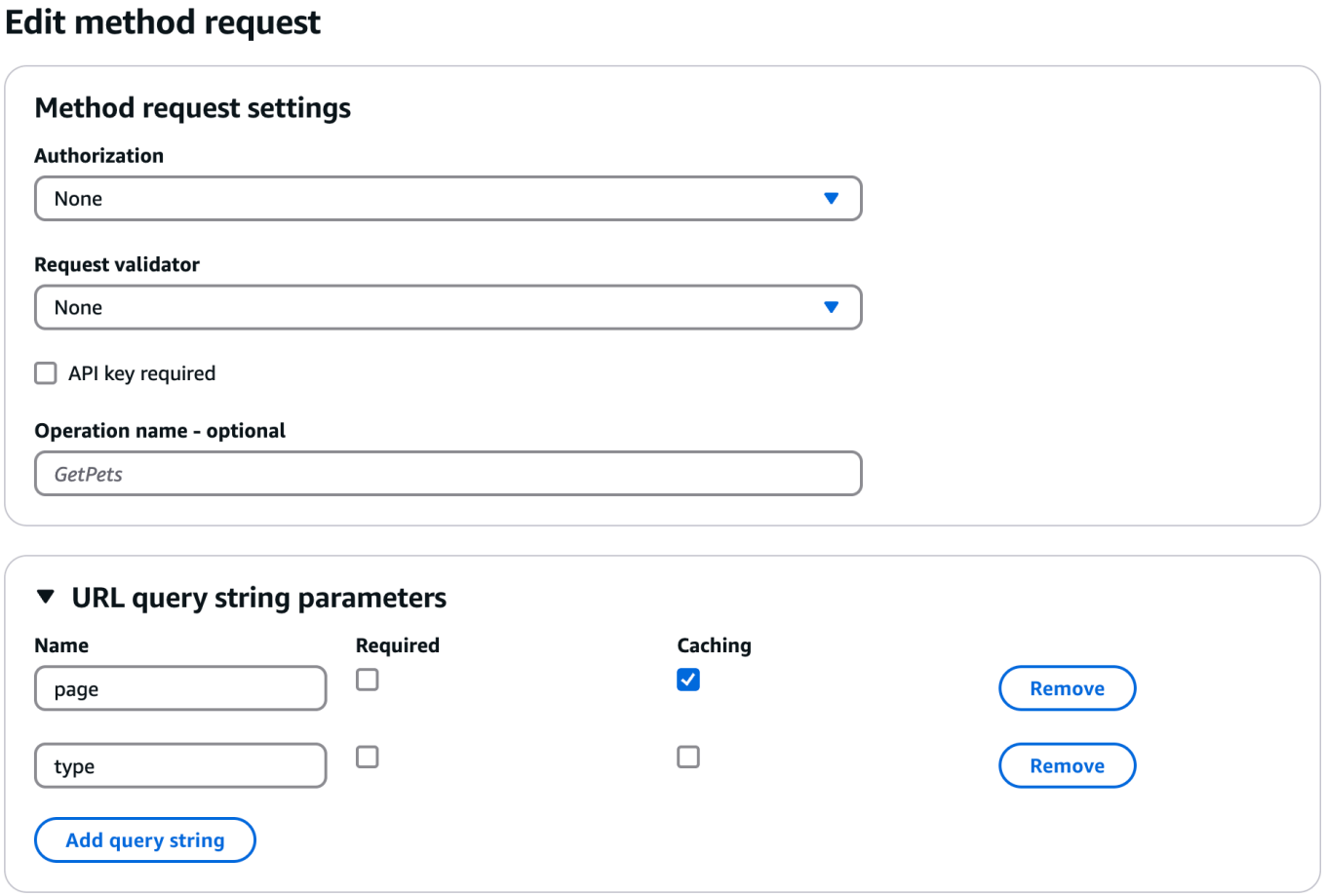
```
GET /users?type=admin&department=A HTTP/1.1
host: example.com
...
```

La risposta da questa richiesta viene memorizzata nella cache e utilizzata per servire la seguente richiesta:

```
GET /users?type=admin&department=B HTTP/1.1
host: example.com
...
```

Console di gestione AWS

Per includere un parametro della richiesta di metodo o di integrazione in una chiave di cache nella console API Gateway, seleziona Caching dopo avere aggiunto il parametro.



AWS CLI

Il comando put-method seguente crea un metodo GET e richiede il parametro della stringa di query `type`:

```
aws apigateway put-method /
    --rest-api-id a1b2c3 /
    --resource-id aaa111 /
    --http-method GET /
    --authorization-type "NONE" /
    --request-parameters "method.request.querystring.type=true"
```

Il comando put-integration seguente crea un'integrazione per il metodo GET con un endpoint HTTP e specifica che Gateway API memorizza nella cache il parametro della richiesta di metodo `type`.

```
aws apigateway put-integration /
    --rest-api-id a1b2c3 /
```

```
    --resource-id aaa111 /
    --http-method GET /
    --type HTTP /
    --integration-http-method GET /
    --uri 'https://example.com' /
    --cache-key-parameters "method.request.querystring.type"
```

Per specificare che Gateway API memorizzi nella cache un parametro della richiesta di integrazione, utilizzare `integration.request.location.name` come parametro della chiave di cache.

Scaricare la cache della fase API in API Gateway

Quando il caching dell'API è abilitato, puoi scaricare la cache della fase API per garantire che i client dell'API ricevano le risposte più recenti dagli endpoint di integrazione.

Console di gestione AWS

Per svuotare la cache della fase API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere un'API che abbia una fase con una cache.
3. Nel riquadro di navigazione principale scegliere Fasi, quindi scegliere una fase con una cache.
4. Scegliere il menu Operazioni fase, quindi selezionare Svuota cache della fase.

AWS CLI

Il comando flush-stage-cache seguente svuota la cache della fase:

```
aws apigateway flush-stage-cache \
    --rest-api-id a1b2c3 \
    --stage-name prod
```

> **Note**
>
> Una volta scaricata la cache, le risposte vengono servite dall'endpoint di integrazione fino a quando la cache non viene creata nuovamente. Durante questo periodo, il numero di

richieste inviate all'endpoint di integrazione potrebbe aumentare. L'operazione potrebbe aumentare temporaneamente la latenza complessiva delle API.

Invalidare una voce della cache di API Gateway

Un client dell'API può invalidare una voce cache esistente e ricaricarla dall'endpoint di integrazione per le singole richieste. Il client deve inviare una richiesta contenente l'intestazione `Cache-Control: max-age=0`. Il client riceve la risposta direttamente dall'endpoint di integrazione anziché dalla cache, a condizione che il client sia autorizzato a eseguire questa operazione. In questo modo, la voce di cache esistente viene sostituita con la nuova risposta recuperata dall'endpoint di integrazione.

Per concedere l'autorizzazione per un client, collegare una policy del formato seguente a un ruolo di esecuzione IAM per l'utente.

> **Note**
>
> Cross-account l'invalidazione della cache non è supportata.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "execute-api:InvalidateCache"
            ],
            "Resource": [
                "arn:aws:execute-api:us-east-1:111111111111:api-id/stage-name/
GET/resource-path-specifier"
            ]
        }
    ]
}
```

Questa policy consente al servizio di esecuzione API Gateway di invalidare la cache per le richieste nella risorsa o nelle risorse specificate. Per specificare un gruppo di risorse di destinazione, usa un carattere jolly (*) per `account-id`, `api-id` e altre voci nel valore ARN di `Resource`. Per ulteriori informazioni su come impostare le autorizzazioni per il servizio di esecuzione API Gateway, consulta Controllo degli accessi a una REST API con le autorizzazioni IAM.

Se non imponi una policy `InvalidateCache` (o scegli la casella di controllo Require authorization (Richiedi autorizzazione) nella console), qualsiasi client può invalidare la cache di API. Se tutti i client o la maggior parte di essi invalidano la cache dell'API, si potrebbe avere un aumento significativo sulla latenza dell'API.

Quando la policy è attiva, il caching è abilitato e l'autorizzazione è richiesta.

È possibile specificare il modo in cui Gateway API gestisce le richieste non autorizzate scegliendo tra le seguenti opzioni:

Col codice stato 403 la richiesta ha esito negativo

Gateway API restituisce una risposta `403 Unauthorized`.

Per impostare questa opzione utilizzando l'API, usa `FAIL_WITH_403`.

Ignora intestazione controllo cache, aggiungi un avviso nell'intestazione della risposta

Gateway API elabora la richiesta e aggiunge un'intestazione di avviso nella risposta.

Per impostare questa opzione utilizzando l'API, usa `SUCCEED_WITH_RESPONSE_HEADER`.

Ignora intestazione controllo cache

Gateway API elabora la richiesta senza aggiungere un'intestazione di avviso nella risposta.

Per impostare questa opzione utilizzando l'API, usa `SUCCEED_WITHOUT_RESPONSE_HEADER`.

È possibile impostare il comportamento di gestione delle richieste non autorizzate tramite la console Gateway API o AWS CLI.

Console di gestione AWS

Per specificare come vengono gestite le richieste non autorizzate

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere un'API che abbia una fase con una cache.

3. Nel riquadro di navigazione principale scegliere Fasi, quindi scegliere una fase con una cache.
4. Per Dettagli fase, scegliere Modifica.
5. Per Gestione delle richieste non autorizzate, selezionare un'opzione.
6. Scegli Continua.
7. Esamina le modifiche e scegli Salva modifiche.

AWS CLI

Il comando update-stage seguente aggiorna una fase per gestire le richieste non autorizzate con esito negativo della richiesta e codice di stato 403:

```
aws apigateway update-stage /
    --rest-api-id a1b2c3 /
    --stage-name 'prod' /
    --patch-operations 'op=replace,path=/*/*/caching/
unauthorizedCacheControlHeaderStrategy,value="FAIL_WITH_403"'
```

CloudFormation esempio di stage con cache

Il CloudFormation modello seguente crea un'API di esempio, fornisce una cache `0.5` da GB per lo `Prod` stage e attiva la memorizzazione nella cache a livello di metodo per tutti i metodi. `GET`

> **Important**
>
> Per il caching viene applicato un addebito orario in base alle dimensioni della cache selezionata. Il caching non rientra nel piano gratuito di AWS . Per ulteriori informazioni, consulta Prezzi di API Gateway.

Esempio CloudFormation modello

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
```

```
      Name: cache-example
  PetsResource:
    Type: 'AWS::ApiGateway::Resource'
    Properties:
      RestApiId: !Ref Api
      ParentId: !GetAtt Api.RootResourceId
      PathPart: 'pets'
  PetsMethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref PetsResource
      HttpMethod: GET
      ApiKeyRequired: true
      AuthorizationType: NONE
      Integration:
        Type: HTTP_PROXY
        IntegrationHttpMethod: GET
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
  ApiDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn:
      - PetsMethodGet
    Properties:
      RestApiId: !Ref Api
  ApiStage:
    Type: 'AWS::ApiGateway::Stage'
    Properties:
      StageName: Prod
      Description: Prod Stage with a cache
      RestApiId: !Ref Api
      DeploymentId: !Ref ApiDeployment
      CacheClusterEnabled: True
      CacheClusterSize: 0.5
      MethodSettings:
        - ResourcePath: /*
          HttpMethod: '*'
          CachingEnabled: True
```

Compressione del payload per REST APIs in API Gateway

API Gateway permette al client di chiamare l'API con payload compressi usando una delle codifiche di contenuto supportate. Per impostazione predefinita, API Gateway supporta la decompressione del

payload di richiesta del metodo. È tuttavia necessario configurare l'API per abilitare la compressione del payload di risposta del metodo.

Per abilitare la compressione in un'API, imposta la proprietà `minimumCompressionsSize` su un valore intero non negativo compreso tra 0 e 10485760 (10 milioni di byte) quando crei l'API oppure dopo averla creata. Per disabilitare la compressione nell'API, imposta la proprietà `minimumCompressionSize` su null oppure rimuovila. È possibile abilitare o disabilitare la compressione per un'API utilizzando la console API Gateway AWS CLI, o l'API REST di API Gateway.

Se desideri che la compressione venga applicata a payload di qualsiasi dimensione, imposta il valore di `minimumCompressionSize` su zero. La compressione di dati di piccole dimensioni può tuttavia comportare un aumento della dimensione finale dei dati. La compressione in API Gateway e la decompressione nel client possono inoltre comportare un aumento della latenza globale e richiedere tempi di elaborazione maggiori. Esegui test case sull'API per determinare un valore ottimale.

Il client può inviare una richiesta API con un payload compresso e un'intestazione `Content-Encoding` appropriata per consentire a Gateway API di decomprimere e applicare i modelli di mappatura appropriati prima di passare la richiesta all'endpoint di integrazione. Dopo che la compressione è stata abilitata e l'API distribuita, il client può ricevere una risposta API con un payload compresso se specifica un'intestazione `Accept-Encoding` appropriata nella richiesta del metodo.

Quando l'endpoint di integrazione prevede e restituisce payload JSON non compressi, un modello di mappatura configurato per un payload JSON non compresso è applicabile al payload compresso. Per un payload di richiesta del metodo compresso, API Gateway decomprime il payload, applica il modello di mappatura e passa la richiesta mappata all'endpoint di integrazione. Per un payload di risposta di integrazione non compresso, API Gateway applica il modello di mappatura, comprime il payload mappato e restituisce il payload compresso al client.

Argomenti

- Abilitazione della compressione del payload per un'API in Gateway API
- Chiamata di un metodo API con un payload compresso in Gateway API
- Ricezione di una risposta API con un payload compresso in Gateway API

Abilitazione della compressione del payload per un'API in Gateway API

Puoi abilitare la compressione per un'API utilizzando la console API Gateway AWS CLI, o un AWS SDK.

Per un'API esistente, dopo aver abilitato la compressione, è necessario distribuire l'API per rendere effettiva la modifica. Per una nuova API, puoi distribuirla dopo aver terminato la configurazione.

> **Note**
>
> La codifica dei contenuti con la massima priorità deve essere supportata da API Gateway. In caso contrario, la compressione non viene applicata al payload della risposta.

Argomenti

- Abilitazione della compressione dei payload per un'API mediante la console API Gateway
- Abilita la compressione del payload per un'API utilizzando il AWS CLI
- Codifiche di contenuto supportate da API Gateway

Abilitazione della compressione dei payload per un'API mediante la console API Gateway

Nella procedura seguente viene descritto come abilitare la compressione del payload per un'API.

Per abilitare la compressione del payload usando la console API Gateway

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Seleziona un'API esistente o creane una nuova.
3. Nel riquadro di navigazione principale, scegli Impostazioni API.
4. Nella sezione Dettagli API, scegli Modifica.
5. Attiva l'opzione Codifica contenuto per abilitare la compressione del payload. In Dimensione corpo minima, immetti un numero per la dimensione di compressione minima (in byte). Per disattivare la compressione, disattiva l'opzione Codifica contenuto.
6. Scegli Save changes (Salva modifiche).

Abilita la compressione del payload per un'API utilizzando il AWS CLI

Il create-rest-apicomando seguente crea un'API con compressione del payload:

```
aws apigateway create-rest-api \
    --name "My test API" \
```

```
    --minimum-compression-size 0
```

Il update-rest-apicomando seguente abilita la compressione del payload per un'API esistente:

```
aws apigateway update-rest-api \
    --rest-api-id 1234567890 \
    --patch-operations op=replace,path=/minimumCompressionSize,value=0
```

La proprietà `minimumCompressionSize` ha un valore intero non negativo compreso tra 0 e 10485760 (10M byte). Misura la soglia di compressione. Se la dimensione del payload è inferiore a questo valore, la compressione o la decompressione non vengono applicate al payload. Impostando il valore su zero, la compressione viene applicata per qualsiasi dimensione di payload.

Il update-rest-apicomando seguente disattiva la compressione del payload:

```
aws apigateway update-rest-api \
    --rest-api-id 1234567890 \
    --patch-operations op=replace,path=/minimumCompressionSize,value=
```

È anche possibile impostare `value` su una stringa vuota `""` o omettere completamente la proprietà `value` nella chiamata precedente.

Codifiche di contenuto supportate da API Gateway

API Gateway supporta le codifiche di contenuto seguenti:

- `deflate`
- `gzip`
- `identity`

API Gateway supporta anche il formato di intestazione `Accept-Encoding` seguente, in base alla specifica 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`

Chiamata di un metodo API con un payload compresso in Gateway API

Per effettuare una richiesta API con un payload compresso, il client deve impostare l'intestazione `Content-Encoding` con una delle codifiche di contenuto supportate.

Supponiamo che tu sia un client API e desideri chiamare il metodo PetStore API (`POST /pets`). Il metodo non deve essere chiamato usando l'output JSON seguente:

```
POST /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Content-Length: ...

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

Deve invece essere chiamato con lo stesso payload compresso usando la codifica GZIP:

```
POST /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Content-Encoding:gzip
Content-Length: ...

[binary]
```

Quando API Gateway riceve la richiesta, verifica se la codifica di contenuto specificata è supportata. Tenta quindi di decomprimere il payload con la codifica di contenuto specificata. Se la decompressione ha esito positivo, la richiesta viene inviata all'endpoint di integrazione. Se la codifica specificata non è supportata oppure se il payload fornito non è compresso con la codifica specificata, API Gateway restituisce la risposta di errore `415 Unsupported Media Type`. L'errore non viene registrato in CloudWatch Logs, se si verifica nella fase iniziale della decompressione prima che l'API e la fase vengano identificate.

Ricezione di una risposta API con un payload compresso in Gateway API

Quando effettua una richiesta in un'API abilitata per la compressione, il client può scegliere di ricevere un payload di risposta compresso con un determinato formato specificando un'intestazione `Accept-Encoding` con una codifica di contenuto supportata.

API Gateway comprime il payload di risposta solo quando vengono soddisfatte le condizioni seguenti:

- La richiesta in ingresso ha l'intestazione `Accept-Encoding` con una codifica di contenuto e un formato supportati.

  > **Note**
  >
  > Se l'intestazione non è impostata, il valore predefinito è `*`, come definito in RFC 7231. In tal caso, API Gateway non comprime il payload. Alcuni browser o client possono aggiungere `Accept-Encoding` (ad esempio, `Accept-Encoding:gzip, deflate, br`) automaticamente alle richieste abilitate per la compressione. Ciò può attivare la compressione del payload in Gateway API. Se i valori dell'intestazione `Accept-Encoding` supportati non sono specificati in modo esplicito, API Gateway non comprime il payload.

- La proprietà `minimumCompressionSize` è impostata nell'API per abilitare la compressione.
- La risposta di integrazione non ha un'intestazione `Content-Encoding`.
- La dimensione di un payload di risposta di integrazione, dopo l'applicazione del modello di mappatura applicabile, è maggiore o uguale al valore di `minimumCompressionSize` specificato.

API Gateway applica l'eventuale modello di mappatura configurato per la risposta di integrazione prima della compressione del payload. Se la risposta di integrazione contiene un'intestazione `Content-Encoding`, API Gateway presuppone che il payload di risposta di integrazione sia già compresso e ignora l'elaborazione della compressione.

Un esempio è l'esempio di PetStore API e la seguente richiesta:

```
GET /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Accept: application/json
```

Il back-end risponde alla richiesta con un payload JSON non compresso simile a quanto segue:

```
200 OK

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

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

Per ricevere l'output come payload compresso, il client API può inviare una richiesta come illustrato di seguito:

```
GET /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Accept-Encoding:gzip
```

Il client riceve la risposta con un'intestazione `Content-Encoding` e un payload con codifica GZIP simile a quanto segue:

```
200 OK
Content-Encoding:gzip
...

���RP�

J�)JV
�:P^IeA*������+(�L ��X�YZ�ku0L0B7!9��C#�&����Y��a���^�X
```

Quando il payload di risposta è compresso, solo la dimensione dei dati compressi viene fatturata per il trasferimento dati.

Distribuisci il tuo REST APIs ai clienti in API Gateway

Questa sezione fornisce dettagli sulla distribuzione del tuo API Gateway APIs ai tuoi clienti. La distribuzione dell'API include la generazione SDKs per i clienti di applicazioni da scaricare e integrare con le loro applicazioni client, la documentazione dell'API in modo che i clienti sappiano come richiamarla dalle loro applicazioni client e la messa a disposizione dell'API come parte delle offerte di prodotti.

Argomenti

- Piani di utilizzo e chiavi API per REST API in Gateway API
- Documentazione delle REST API in Gateway API
- Genera SDKs per REST APIs in API Gateway
- Vendi il tuo API Gateway APIs tramite Marketplace AWS

Piani di utilizzo e chiavi API per REST API in Gateway API

Dopo aver creato, sottoposto a test e distribuito le API, puoi usare i piani di utilizzo di API Gateway per renderle disponibili come offerte per i clienti. Puoi configurare piani di utilizzo e chiavi API per permettere a clienti di accedere ad API selezionate e iniziare a limitare le richieste a tali API sulla base di limiti e quote definiti. Questi possono essere impostati a livello API o metodo API.

Cosa sono i piani di utilizzo e le chiavi API?

Un piano di utilizzo consente di specificare chi può accedere a una o più fasi e metodi API distribuiti, inclusa la frequenza di richiesta di destinazione per iniziare a limitare le richieste. Il piano utilizza chiavi API per identificare client API e chi può accedere akke fasi API associate a ciascuna chiave.

Le chiavi API sono valori stringa alfanumerici che vengono distribuiti ai clienti degli sviluppatori di applicazioni per concedere l'accesso all'API. Puoi utilizzare le chiavi API insieme ai provider di autorizzazioni per Lambda, ruoli IAM o Amazon Cognito per controllare l'accesso alle API. API Gateway può generare chiavi API per tuo conto o puoi importarle da un file CSV. Puoi generare una chiave API in API Gateway o importarla in API Gateway da un'origine esterna. Per ulteriori informazioni, consulta the section called “Configurazione di chiavi API per REST API in Gateway API”.

Una chiave API ha un nome e un valore. I termini "chiave API" e "valore di chiave API" vengono spesso usati in modo intercambiabile. Il nome non può superare 1024 caratteri. Il valore è una stringa alfanumerica tra 20 e 128 caratteri, ad esempio, `apikey1234abcdefghij0123456789`.

> **Important**
>
> I valori di chiave API devono essere univoci. Se provi a creare due chiavi API con nomi diversi e lo stesso valore, queste vengono considerate da API Gateway la stessa chiave API.

Una chiave API può essere associata a più piani di utilizzo. Un piano di utilizzo può essere associato a più fasi. Tuttavia, una determinata chiave API può essere associata a un solo piano di utilizzo per ogni fase dell'API.

Un limite di throttling imposta il punto di destinazione in cui dovrebbe iniziare la limitazione delle richieste. Può essere impostato a livello API o metodo API.

Un limite di quota è il numero massimo di richieste con una determinata chiave API che possono essere inviate in un intervallo di tempo specificato. Puoi configurare singoli metodi API per richiedere l'autorizzazione per le chiavi API in base alla configurazione del piano di utilizzo.

I limiti di throttling e di quote si applicano alle richieste per le singole chiavi API che vengono aggregate nelle fasi API in un piano di utilizzo.

Note

I piani di utilizzo della limitazione (della larghezza di banda della rete) e le quote non sono limiti rigidi e vengono applicati sulla base del miglior tentativo. In alcuni casi, i client possono superare le quote impostate. Non fare affidamento sulle quote del piano di utilizzo o limitazione (della larghezza di banda della rete) per controllare i costi o bloccare l'accesso a un'API. Considerare l'utilizzo di Budget AWS per monitorare i costi e AWS WAF gestire le richieste API.

Best practice per le chiavi API e i piani di utilizzo

Di seguito vengono fornite le best practice da seguire quando usi chiavi API e piani di utilizzo.

Important

- Non utilizzare le chiavi API per l'autenticazione o l'autorizzazione per il controllo degli accessi alle API. Se un piano di utilizzo include più API, un utente con una chiave API valida per un'API nel piano di utilizzo potrà accedere a tutte le API nel piano di utilizzo. Per controllare gli accessi all'API, utilizza invece un ruolo IAM, un sistema di autorizzazione Lambda o un pool di utenti di Amazon Cognito.

- Utilizzare le chiavi API generate da API Gateway. Le chiavi API non devono includere informazioni riservate; i client in genere le trasmettono in intestazioni che possono essere registrate.

- Se si utilizza un portale per gli sviluppatori per pubblicare le API, occorre tenere presente che tutte le API in un piano di utilizzo specifico sono sottoscrivibili dai clienti, anche se non sono visibili ai clienti.
- In alcuni casi, i client possono superare le quote impostate. Non fare affidamento sui piani di utilizzo per controllare i costi. Considera l'utilizzo di Budget AWS per monitorare i costi e di AWS WAF per gestire le richieste API.
- Dopo aver aggiunto una chiave API a un piano di utilizzo, il completamento dell'operazione di aggiornamento potrebbe richiedere alcuni minuti.

Scelta dell'origine di una chiave API in Gateway API

Quando associ un piano di utilizzo a un'API e abiliti chiavi API su metodi API, ogni richiesta in ingresso all'API deve contenere una chiave API. API Gateway legge la chiave e la confronta con le chiavi nel piano di utilizzo. Se esiste una corrispondenza, API Gateway esegue il throttling delle richieste in base alla quota e al limite delle richieste del piano. In caso contrario, genera un'eccezione `InvalidKeyParameter` e l'intermediario riceve una risposta `403 Forbidden`.

L'API di API Gateway è in grado di ricevere chiavi API da una delle due seguenti origini:

`HEADER`

Devi distribuire le chiavi API ai clienti e chiedere loro di passare la chiave API come intestazione `X-API-Key` di ogni richiesta in ingresso.

`AUTHORIZER`

Puoi fare in modo che la chiave API venga restituita da un'autorizzazione Lambda come parte della risposta di autorizzazione. Per ulteriori informazioni sulla risposta delle autorizzazioni, consulta the section called “Output da un sistema di autorizzazione Lambda di Gateway API”.

> **Note**
>
> Per informazioni sulle best practice da prendere in considerazione, consulta the section called "Best practice per le chiavi API e i piani di utilizzo".

La procedura seguente descrive come scegliere un'origine della chiave API per un'API.

Console di gestione AWS

Per scegliere un'origine della chiave API per un'API

1. Accedere alla console API Gateway.
2. Seleziona un'API esistente o creane una nuova.
3. Nel riquadro di navigazione principale, scegli Impostazioni API.
4. Nella sezione Dettagli API, scegli Modifica.
5. In Origine chiave API, seleziona `Header` o `Authorizer` nell'elenco a discesa.
6. Scegli Save changes (Salva modifiche).

AWS CLI

Il comando update-rest-api seguente aggiorna un'API per impostare l'origine della chiave API su `AUTHORIZER`:

```
aws apigateway update-rest-api --rest-api-id 1234123412 --patch-operations
 op=replace,path=/apiKeySource,value=AUTHORIZER
```

Per fare in modo che il client invii una chiave API, impostare `value` su `HEADER` nel comando precedente.

REST API

Per scegliere un'origine di chiave API per un'API utilizzando l'API REST API Gateway, invoca `restapi:update` come segue:

```
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"
    }
  ]
}
```

Per fare in modo che un'autorizzazione restituisca una chiave API, imposta `value` su `AUTHORIZER` nell'input `patchOperations` precedente.

Formato file chiave API di API Gateway

API Gateway può importare le chiavi API dai file esterni in formato CSV (comma-separated value), quindi associare le chiavi importate a uno o più piani di utilizzo. Il file importato deve contenere le colonne `Name` e `Key`. Per i nomi delle intestazioni di colonna non viene rilevata la distinzione tra maiuscole e minuscole e le colonne possono essere disposte in qualsiasi ordine, come mostrato nel'esempio seguente:

```
Key,name
apikey1234abcdefghij0123456789,MyFirstApiKey
```

Il valore di `Key` deve essere costituito da un numero di caratteri compreso tra 20 e 128. Un valore `Name` non può superare 1024 caratteri.

Un file di chiavi API può includere anche la colonna `Description`, `Enabled` o `UsagePlanIds`, come mostrato nell'esempio seguente:

```
Name,key,description,Enabled,usageplanIds
MyFirstApiKey,apikey1234abcdefghij0123456789,An imported key,TRUE,c7y23b
```

Quando una chiave è associata a più di un piano di utilizzo il valore `UsagePlanIds` è una stringa separata da virgole di ID di piani di utilizzo racchiusi tra una coppia di apici singoli o doppi, come mostrato nell'esempio seguente:

```
Enabled,Name,key,UsageplanIds
true,MyFirstApiKey,apikey1234abcdefghij0123456789,"c7y23b,glvrsr"
```

Le colonne non riconosciute sono consentite ma vengono ignorate. Il valore predefinito è una stringa vuota o il valore booleano `true`.

La stessa chiave API può essere importata più volte e la versione più recente sovrascrive la precedente. Due chiavi API sono identiche se hanno lo stesso valore `key`.

> **Note**
>
> Per informazioni sulle best practice da prendere in considerazione, consulta the section called “Best practice per le chiavi API e i piani di utilizzo”.

Configurazione di chiavi API per REST API in Gateway API

Per configurare le chiavi API, completa queste operazioni:

- Configura i metodi API in modo che richiedano una chiave API.
- Crea o importa in una Regione una chiave API per l'API.

Prima di impostare le chiavi API, è necessario aver creato un'API e averla distribuita in una fase. Una volta creata una chiave API, non può essere modificata.

Per istruzioni su come creare e distribuire un'API utilizzando la console API Gateway, consulta rispettivamente Sviluppo di REST API in Gateway API e Implementazione di REST API in Gateway API.

Dopo aver creato una chiave API, è necessario associarla a un piano di utilizzo. Per ulteriori informazioni, consulta Configurazione di piani di utilizzo per REST API in Gateway API.

> **Note**
>
> Per informazioni sulle best practice da prendere in considerazione, consulta the section called “Best practice per le chiavi API e i piani di utilizzo”.

Argomenti

- Richiesta di una chiave API per un metodo
- Crea una chiave API
- Importazione di chiavi API

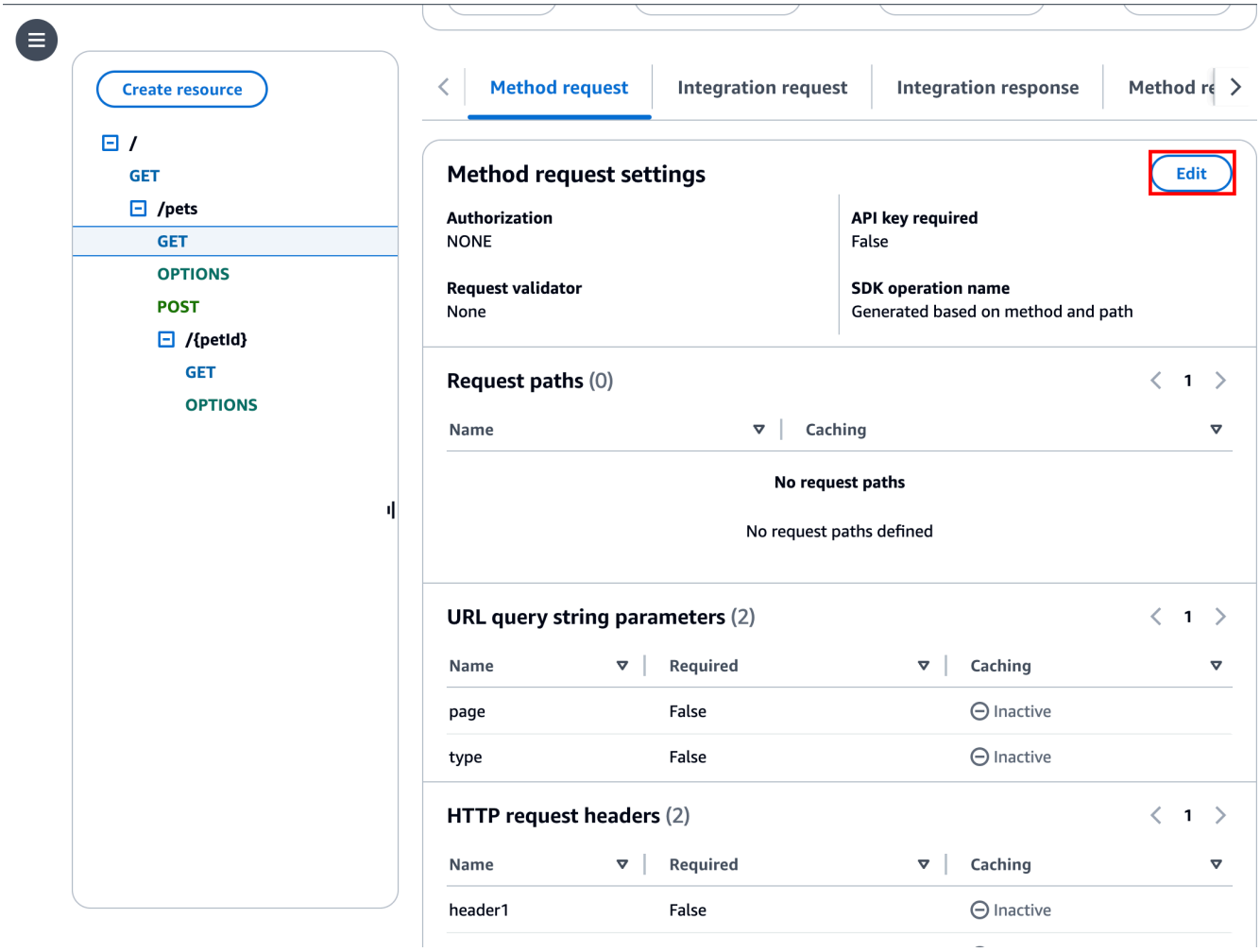
Richiesta di una chiave API per un metodo

La procedura seguente mostra come configurare un metodo API in modo che richieda una chiave API.

Console di gestione AWS

Per configurare un metodo API in modo che richieda una chiave API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere una REST API.
3. Nel riquadro di navigazione principale di API Gateway, scegliere Resources (Risorse).
4. In Resources (Risorse) creare un nuovo metodo o sceglierne uno esistente.
5. Nella scheda Richiesta metodo, in Impostazioni richiesta metodo, scegli Modifica.



6. Seleziona Chiave API necessaria.
7. Scegli Save (Salva).
8. Distribuisci o ridistribuisci l'API per rendere effettivo il requisito.

Se l'opzione Chiave API necessaria è impostata su `false` e non si esegue la procedura precedente, la chiave API associata a una fase API non viene utilizzata per il metodo.

AWS CLI

Il comando put-method seguente crea un metodo PUT che richiede una chiave API:

```
aws apigateway put-method \
    --rest-api-id 1234123412 \
    --resource-id a1b2c3 \
    --http-method PUT \
    --authorization-type "NONE" \
```

```
    --api-key-required
```

Il comando update-method seguente aggiorna un metodo esistente in modo che richieda una chiave API:

```
aws apigateway update-method \
    --rest-api-id 1234123412 \
    --resource-id a1b2c3 \
    --http-method PUT \
    --patch-operations op="replace",path="/apiKeyRequired",value="true"
```

REST API

Per configurare un metodo in modo che richieda una chiave API, procedi in uno dei seguenti modi:

- Invoca `method:put` per creare un metodo. Imposta `apiKeyRequired` un `true` nel payload di richiesta.
- Invoca `method:update` per impostare `apiKeyRequired` su `true`.

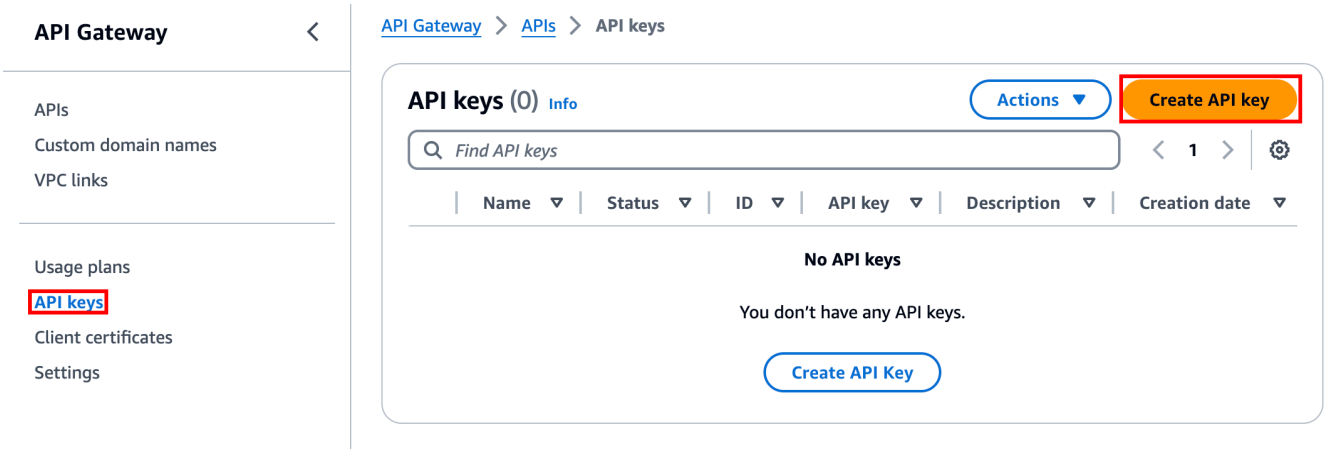
Crea una chiave API

La procedura seguente mostra come creare una chiave API. Non tenere conto di questo passaggio se desideri importare la chiave API.

Console di gestione AWS

Per creare una chiave API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere una REST API.
3. Nel riquadro di navigazione principale di Gateway API, scegli Chiavi API.
4. Scegli Crea chiave API.



5. In Nome, immetti un nome.
6. (Facoltativo) In Descrizione, immetti una descrizione.
7. In Chiave API, scegli Genera automaticamente per fare in modo che Gateway API generi il valore della chiave oppure scegli Personalizza per creare il tuo valore di chiave.
8. Scegli Save (Salva).

AWS CLI

Il comando create-api-key seguente crea una chiave API:

```
aws apigateway create-api-key \
   --name 'Dev API key' \
   --description 'API key for Devs' \
   --enabled
```

REST API

Invoca `apikey:create` per creare una chiave API.

Importazione di chiavi API

La procedura seguente descrive come importare le chiavi API. Non tenere conto di questo passaggio se è già stata creata una chiave API.

Console di gestione AWS

Per importare chiavi API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere una REST API.
3. Nel riquadro di navigazione principale, scegli Chiavi API.
4. Scegli il menu a discesa Operazioni e quindi Importa chiavi API.
5. Per caricare un file di chiavi con valori separati da virgole, scegli Scegli file. Puoi anche immettere le chiavi nell'editor di testo. Per informazioni sul formato del file, consulta the section called “Formato file chiave API di API Gateway”.
6. Scegli Errore su avvertenze per arrestare l'importazione se si verifica un errore oppure Ignora avvisi per continuare a importare voci di chiavi valide in caso di presenza di avvisi.
7. Scegli Importa per importare le tue chiavi API.

AWS CLI

Il comando import-api-keys seguente importa una chiave API:

```
aws apigateway import-api-key \
    a--body fileb://keys.csv \
    --format csv
```

REST API

Invoca `apikey:import` per importare una chiave API da un file. Per il formato file, consulta Formato file chiave API di API Gateway.

Non è possibile modificare il valore della nuova chiave API. Dopo aver creato l’API, si configura un piano di utilizzo. Per ulteriori informazioni, consulta Configurazione di piani di utilizzo per REST API in Gateway API.

Configurazione di piani di utilizzo per REST API in Gateway API

Prima di creare un piano di utilizzo, è necessario avere configurato le chiavi API desiderate. Per ulteriori informazioni, consulta Configurazione di chiavi API per REST API in Gateway API.

Argomenti

- Eseguire la migrazione dell'API a Piani di utilizzo predefiniti (se necessario)
- Creazione di un piano di utilizzo
- Aggiunta di una fase a un piano di utilizzo
- Aggiunta di una chiave API a un piano di utilizzo

Eseguire la migrazione dell'API a Piani di utilizzo predefiniti (se necessario)

Se hai iniziato a usare API Gateway dopo l'introduzione della caratteristica dei piani di utilizzo l'11 agosto 2016, troverai che i piani di utilizzo sono abilitati automaticamente in tutte le regioni supportate.

Se hai iniziato a utilizzare API Gateway prima di quella data, potrebbe essere necessario migrare ai piani di utilizzo predefiniti. Ti verrà proposta l'opzione Enable Usage Plans (Abilita piani di utilizzo) prima di usare i piani di utilizzo per la prima volta nella regione selezionata. Quando abiliti questa opzione, vengono creati piani di utilizzo predefiniti per ogni fase API univoca associata alle chiavi API esistenti. Nel piano di utilizzo predefinito, non vengono impostati inizialmente throttle o limiti di quota e le associazioni tra le chiavi API e le fasi API sono copiate sui piani di utilizzo. Il comportamento dell'API resta inalterato. Tuttavia, è necessario utilizzare la `UsagePlan` `apiStages` proprietà per associare i valori della fase API specificati (`apiId` and `stage`) alle chiavi API incluse (via `UsagePlanKey`), anziché utilizzare la `ApiKey` `stageKeys` proprietà.

Per controllare se hai già eseguito la migrazione ai piani di utilizzo predefiniti, utilizza il comando della CLI `get-account`. Nell'output del comando, l'elenco `features` include una voce `"UsagePlans"` quando i piani di utilizzo sono abilitati.

Puoi anche migrare le tue API ai piani di utilizzo predefiniti utilizzando AWS CLI quanto segue:

Per migrare ai piani di utilizzo predefiniti utilizzando il AWS CLI

1. Invoca questo comando della CLI: `update-account`.
2. Per il parametro `cli-input-json`, utilizza il seguente JSON:

```
[
    {
        "op": "add",
        "path": "/features",
        "value": "UsagePlans"
    }
]
```

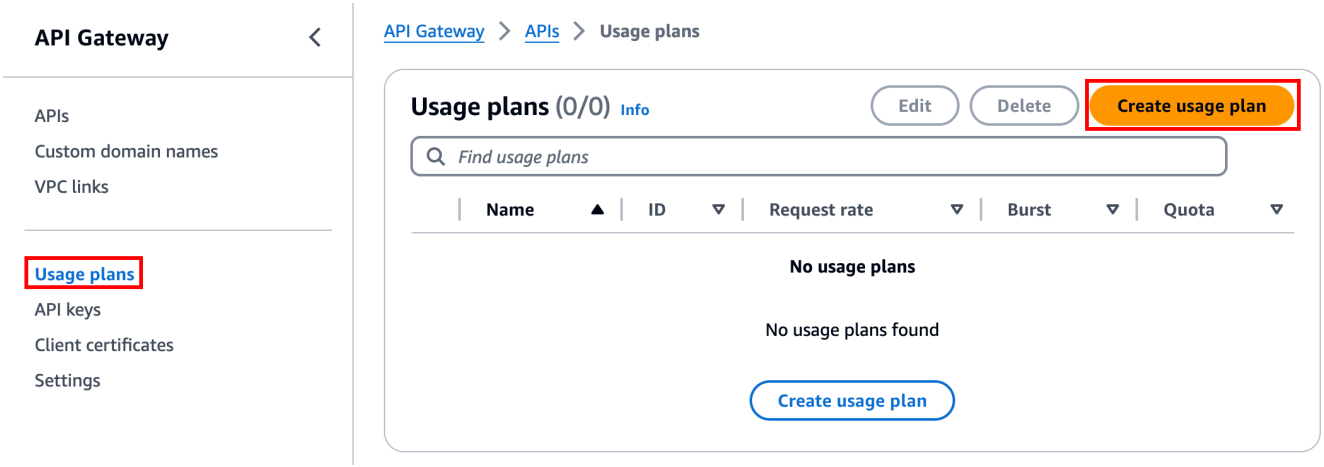
Creazione di un piano di utilizzo

La procedura seguente illustra come creare un piano di utilizzo.

Console di gestione AWS

Per creare un piano di utilizzo

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel riquadro di navigazione principale di Gateway API, scegli Piani di utilizzo e quindi Crea piano di utilizzo.



3. In Nome, immetti un nome.
4. (Facoltativo) In Descrizione, immetti una descrizione.
5. Per impostazione predefinita, i piani di utilizzo abilitano la limitazione (della larghezza di banda della rete). Immetti un valore in Tariffa e Ottimizzazione per il tuo piano di utilizzo. Scegli Throttling per disattivare la limitazione (della larghezza di banda della rete).
6. Per impostazione predefinita, i piani di utilizzo abilitano una quota per un periodo di tempo. In Richieste, immetti il numero totale di richieste che un utente può effettuare nel periodo di validità del tuo piano di utilizzo. Scegli Quota per disattivare la quota.
7. Scegli Crea piano di utilizzo.

AWS CLI

Il comando create-usage-plan seguente crea un piano di utilizzo che si ripristina all'inizio del mese:

```
aws apigateway create-usage-plan \
```

```
    --name "New Usage Plan" \
    --description "A new usage plan" \
    --throttle burstLimit=10,rateLimit=5 \
    --quota limit=500,offset=0,period=MONTH
```

REST API

Invoca `usageplan:create` per creare un piano di utilizzo.

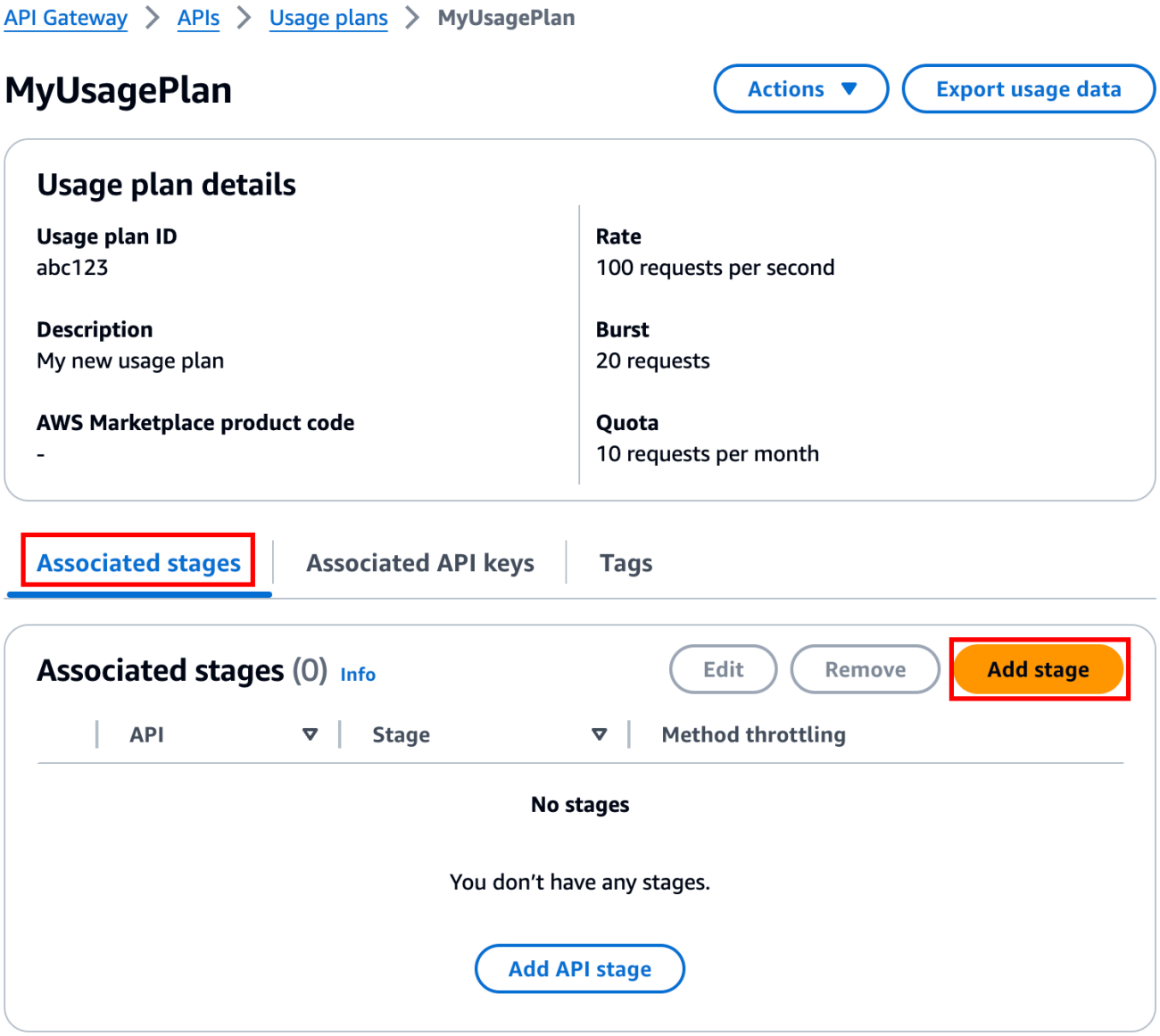
Aggiunta di una fase a un piano di utilizzo

La procedura seguente illustra come aggiungere una fase a un piano di utilizzo.

Console di gestione AWS

Per aggiungere una fase a un piano di utilizzo

1. Seleziona il piano di utilizzo.
2. Nella scheda Fasi associate, scegli Aggiungi fase.



3. In API, seleziona un'API.
4. In Fase, seleziona una fase.
5. (Facoltativo) Per attivare la limitazione (della larghezza di banda della rete) a livello di metodo, esegui le operazioni indicate di seguito:
   a. Scegli Method-level la limitazione, quindi scegli Aggiungi metodo.
   b. InRisorsa, seleziona una risorsa nella tua API.
   c. In Metodo, seleziona un metodo nella tua API.
   d. Immetti un valore in Tariffa e Ottimizzazione per il tuo piano di utilizzo.
6. Scegli Aggiungi al piano di utilizzo.

AWS CLI

Il comando update-usage-plan seguente aggiunge la fase `Prod` di un'API a un piano di utilizzo:

```
aws apigateway update-usage-plan \
    --usage-plan-id abc123 \
    --patch-operations op="add",path="/apiStages",value="a1b1c2:Prod"
```

REST API

Chiama `usageplan:update` per aggiornare un piano di utilizzo.

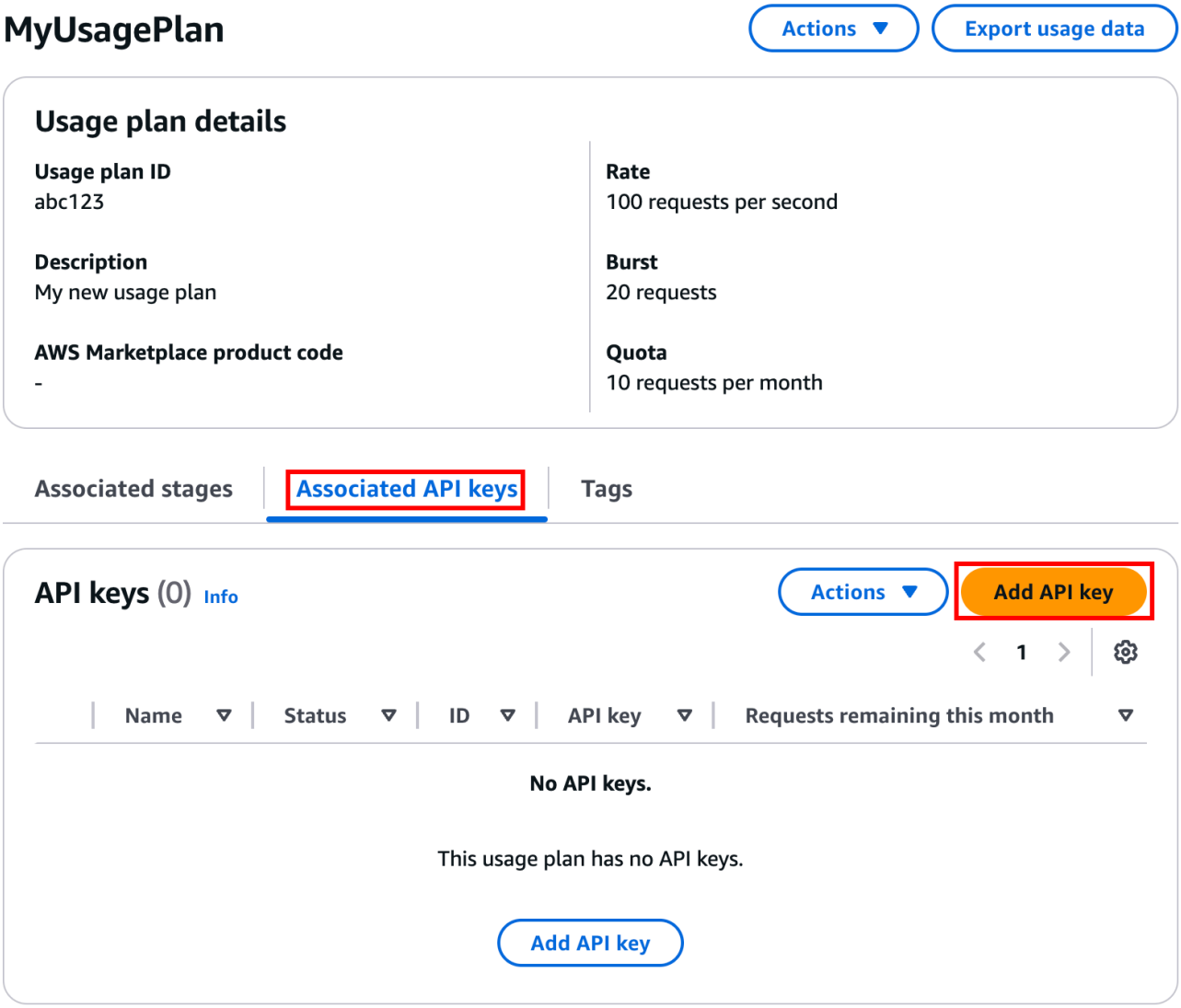
Aggiunta di una chiave API a un piano di utilizzo

La procedura seguente descrive come aggiungere una chiave API a un piano di utilizzo.

Console di gestione AWS

Per aggiungere una chiave a un piano di utilizzo

1. Nella scheda Chiavi API associate, scegli Aggiungi chiave API.



2. a. Per associare una chiave esistente al tuo piano di utilizzo, seleziona Aggiungi chiave esistente, quindi seleziona la chiave esistente nel menu a discesa.

   b. Per creare una nuova chiave API, seleziona Crea e aggiungi nuova chiave, quindi crea una nuova chiave. Per ulteriori informazioni su come creare una nuova chiave, consulta Crea una chiave API.

3. Scegli Aggiungi chiave API.

AWS CLI

Il comando create-usage-plan-key seguente associa una chiave API esistente a un piano di utilizzo:

```
aws apigateway create-usage-plan-key \
    --usage-plan-id a1b2c3 \
    --key-type "API_KEY" \
    --key-id aaa111bbb
```

REST API

Chiama `usageplankey:create` per associare una chiave API esistente a un piano di utilizzo.

È anche possibile associare direttamente le chiavi API a un piano di utilizzo quando vengono importate. Invoca `apikey:import` per aggiungere una o più chiavi API direttamente al piano di utilizzo specificato. Il payload della richiesta deve contenere valori di chiavi API, l'identificativo del piano di utilizzo associato, i flag booleani che indicano che le chiavi sono abilitate per il piano di utilizzo e, possibilmente, nomi e descrizioni delle chiavi API.

L'esempio che segue della richiesta `apikey:import` aggiunge tre chiavi API (identificate da `key`, `name` e `description`) a un piano di utilizzo (identificato da `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
```

Di conseguenza, vengono create tre risorse `UsagePlanKey` che vengono aggiunte a `UsagePlan`.

Puoi aggiungere chiavi API anche a più di un piano di utilizzo. A questo scopo, modifica ogni valore della colonna `usageplanIds` in una stringa separata da virgole contenente gli identificatori del piano di utilizzo selezionato e racchiusa tra apici (`"n371pt,m282qs"` o `'n371pt,m282qs'`).

> **Note**
>
> Una chiave API può essere associata a più piani di utilizzo. Un piano di utilizzo può essere associato a più fasi. Tuttavia, una determinata chiave API può essere associata a un solo piano di utilizzo per ogni fase dell'API.

Gestione di un piano di utilizzo per REST API in Gateway API

La gestione di un piano di utilizzo implica il monitoraggio delle quote utilizzate e rimanenti in un determinato periodo di tempo, se necessario, e l'estensione delle quote rimanenti di una quantità specificata. Le procedure seguenti descrivono come monitorare le quote.

Console di gestione AWS

Per monitorare le quote utilizzate e rimanenti

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel riquadro di navigazione principale di Gateway API, scegli Piani di utilizzo.
3. Seleziona un piano di utilizzo.
4. Scegli la scheda Chiavi API associate per visualizzare il numero di richieste rimanenti per il periodo di tempo per ciascuna chiave.
5. (Facoltativo) Scegli Esporta dati di utilizzo, quindi scegli una data in Da e una data in A. Quindi, scegli JSON o CSV per il formato dei dati esportati e infine scegli Esporta.

   L'esempio di seguito mostra un file esportato.

```
{
    "px1KW6...qBazOJH": [
      [
        0,
        5000
      ],
      [
        0,
        5000
      ],
      [
        0,
        10
```

```
      ]
    ]
  }
```

I dati di utilizzo nell'esempio mostrano i dati di utilizzo per un client API identificato dalla chiave API (`px1KW6...qBaz0JH`) tra il 1 e il 3 agosto 2016. I dati di utilizzo giornalieri mostrano le quote utilizzate e rimanenti. In questo esempio il sottoscrittore non ha ancora usato le quote riservate e l'amministratore o il proprietario dell'API ha ridotto la quota rimanente da 5000 a 10 il terzo giorno.

Le procedure seguenti descrivono come modificare le quote.

Per estendere le quote rimanenti

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel riquadro di navigazione principale di Gateway API, scegli Piani di utilizzo.
3. Seleziona un piano di utilizzo.
4. Scegli la scheda Chiavi API associate per visualizzare il numero di richieste rimanenti per il periodo di tempo per ciascuna chiave.
5. Seleziona una chiave API, quindi scegli Concedi estensione utilizzo.
6. Immetti un numero di quote in Richieste rimanenti. Puoi aumentare o diminuire le richieste rimanenti per il periodo di validità del tuo piano di utilizzo.
7. Scegli Aggiorna quota.

AWS CLI

I comandi di esempio update-usage-plan seguenti mostrano come aggiungere, rimuovere o modificare le impostazioni di limitazione (della larghezza di banda della rete) a livello di metodo in un piano di utilizzo.

> **Note**
>
> Assicurarsi di modificare `us-east-1` nel valore di regione appropriato per l'API.

Per aggiungere o sostituire un limite di tasso per il throttling di una risorsa e un metodo singoli:

```
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"
```

Per aggiungere o sostituire un limite di ottimizzazione per il throttling di una risorsa e un metodo singoli:

```
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"
```

Per rimuovere le impostazioni di throttling a livello di metodo per una risorsa e un metodo singoli:

```
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=""
```

Per rimuovere tutte le impostazioni di throttling a livello di metodo per un'API:

```
aws apigateway --region us-east-1 update-usage-plan --usage-plan-id planId --patch-
operations op="remove",path="/apiStages/apiId:stage/throttle ",value=""
```

Di seguito è riportato un esempio utilizzando l'API di esempio 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}}"'
```

REST API

Chiama `usageplan:update` per gestire un piano di utilizzo.

Crea e configura chiavi API e piani di utilizzo con CloudFormation

Puoi utilizzare CloudFormation per richiedere chiavi API sui metodi API e creare un piano di utilizzo per un'API. Il CloudFormation modello di esempio esegue le seguenti operazioni:

- Crea un'API Gateway API con i metodi `GET` e `POST`.

- Richiede una chiave API per i metodi `GET` e `POST`. Questa API riceve le chiavi dall'intestazione `X-API-KEY` di ogni richiesta in entrata.
- Creazione di una chiave API.
- Crea un piano di utilizzo per specificare una quota mensile di 1000 richieste al mese, un limite di velocità di limitazione di 100 richieste al secondo e un limite di limitazione di 200 richieste al secondo.
- Specifica un limite di velocità di limitazione a livello di metodo di 50 richieste al secondo e un limite a livello di metodo di 100 richieste al secondo per il metodo `GET`.
- Associa le fasi API e le chiavi API al piano di utilizzo.

```
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}'
```

Configurazione di un metodo per utilizzare le chiavi API con una definizione OpenAPI

È possibile utilizzare una definizione OpenAPI per richiedere le chiavi API per un metodo.

Crea l'oggetto requisito di sicurezza di ogni metodo per richiedere una chiave API che invochi il metodo. Quindi, definisci `api_key` nella definizione di sicurezza. Dopo aver creato l'API, aggiungi la nuova fase API al piano di utilizzo.

L'esempio seguente crea un'API e richiede una chiave API per i metodi `POST` e `GET`:

OpenAPI 2.0

```
{
  "swagger" : "2.0",
  "info" : {
    "version" : "2024-03-14T20:20:12Z",
    "title" : "keys-api"
  },
  "basePath" : "/v1",
  "schemes" : [ "https" ],
  "paths" : {
    "/pets" : {
      "get" : {
        "responses" : { },
        "security" : [ {
          "api_key" : [ ]
```

```
        } ],
        "x-amazon-apigateway-integration" : {
          "type" : "http_proxy",
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets/",
          "passthroughBehavior" : "when_no_match"
        }
      },
      "post" : {
        "responses" : { },
        "security" : [ {
          "api_key" : [ ]
        } ],
        "x-amazon-apigateway-integration" : {
          "type" : "http_proxy",
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets/",
          "passthroughBehavior" : "when_no_match"
        }
      }
    }
  },
  "securityDefinitions" : {
    "api_key" : {
      "type" : "apiKey",
      "name" : "x-api-key",
      "in" : "header"
    }
  }
}
```

OpenAPI 3.0

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "keys-api",
    "version" : "2024-03-14T20:20:12Z"
  },
  "servers" : [ {
    "url" : "{basePath}",
    "variables" : {
      "basePath" : {
```

```
        "default" : "v1"
      }
    }
  } ],
  "paths" : {
    "/pets" : {
      "get" : {
        "security" : [ {
          "api_key" : [ ]
        } ],
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets/",
          "passthroughBehavior" : "when_no_match",
          "type" : "http_proxy"
        }
      },
      "post" : {
        "security" : [ {
          "api_key" : [ ]
        } ],
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets/",
          "passthroughBehavior" : "when_no_match",
          "type" : "http_proxy"
        }
      }
    }
  },
  "components" : {
    "securitySchemes" : {
      "api_key" : {
        "type" : "apiKey",
        "name" : "x-api-key",
        "in" : "header"
      }
    }
  }
}
```

Test di piani di utilizzo per REST API in Gateway API

Ad esempio, utilizziamo l' PetStore API, che è stata creata in[Tutorial: creazione di un'API REST mediante l'importazione di un esempio](). Presupponiamo che l'API sia configurata per l'uso della chiave API `Hiorr45VR...c4GJc`. La procedura seguente illustra come testare un piano di utilizzo.

Per testare il piano di utilizzo

- Fai una richiesta `GET` nella risorsa Pets (`/pets`), con i parametri di query `?type=...&page=...` dell'API (ad esempio, `xbvxlpijch`) in un piano di utilizzo:

```
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**
>
> Devi inviare questa richiesta al componente `execute-api` di API Gateway e fornire la chiave API necessaria (ad esempio `Hiorr45VR...c4GJc`) nell'intestazione `x-api-key` obbligatoria.

Se riesce, la risposta restituisce un codice di stato `200 OK` e un payload contenente i risultati richiesti dal back-end: Se dimentichi di impostare l'intestazione `x-api-key` o se la imposti con una chiave sbagliata, ottieni una risposta `403 Forbidden`. Tuttavia, se non hai configurato il metodo in modo che richieda una chiave API, probabilmente otterrai la risposta `200 OK`, anche se imposti l'intestazione `x-api-key` in modo non corretto e se vengono superati i imiti di throttling e di quota del piano di utilizzo.

A volte, quando si verifica un errore interno in cui API Gateway non riesce ad applicare i limiti di throttling o le quote del piano di utilizzo per la richiesta, API Gateway elabora la richiesta senza applicare i limiti di throttling o le quote specificate nel piano i utilizzo. Tuttavia, registra un messaggio di errore `Usage Plan check failed due to an internal error` in CloudWatch. Puoi ignorare questi errori occasionali.

Chiamata di un metodo tramite una chiave API

A seconda del tipo di origine della chiave API scelta, usa una delle procedure seguenti per utilizzare chiavi API con origine intestazione o chiavi API restituite dall'autorizzazione nell'invocazione di metodo:

Per utilizzare chiavi API con origine intestazione:

1. Crea un'API con i metodi API desiderati e quindi implementa l'API in una fase.
2. Crea un nuovo piano di utilizzo o scegline uno esistente. Aggiungi la fase API distribuita al piano di utilizzo. Collega una chiave API al piano di utilizzo o scegli una chiave API esistente nel piano. Prendi nota del valore della chiave API scelta.
3. Configura i metodi API in modo che richiedano una chiave API.
4. Ridistribuisci l'API nella stessa fase. Se distribuisci l'API in una nuova fase, assicurati di aggiornare il piano di utilizzo per collegare la nuova fase API.
5. Chiama l'API utilizzando la chiave API. Il seguente comando curl di esempio invoca il metodo `GET` sulla risorsa `getUsers` della fase `prod` di un'API utilizzando una chiave API.

```
curl -H "X-API-Key: abcd1234" 'https://b123abcde4.execute-api.us-
west-2.amazonaws.com/prod/getUsers'
```

Il client può ora chiamare i metodi API fornendo l'intestazione `x-api-key` con la chiave API scelta come valore di intestazione. Di seguito è riportato un esempio di chiamata:

Per utilizzare chiavi API con origine autorizzazione:

1. Crea un'API con i metodi API desiderati e quindi implementa l'API in una fase.
2. Crea un nuovo piano di utilizzo o scegline uno esistente. Aggiungi la fase API distribuita al piano di utilizzo. Collega una chiave API al piano di utilizzo o scegli una chiave API esistente nel piano. Prendi nota del valore della chiave API scelta.
3. Crea una funzione di autorizzazione Lambda basata su token. Includi `usageIdentifierKey:`*`{api-key}`* come proprietà a livello di root della risposta di autorizzazione. Per istruzioni sulla creazione di un sistema di autorizzazione basato su token, consulta the section called “Esempio di funzione Lambda dell'autorizzatore TOKEN”.
4. Configura i metodi API in modo che richiedano una chiave API e abilita anche l'autorizzazione Lambda per i metodi.

5. Ridistribuisci l'API nella stessa fase. Se distribuisci l'API in una nuova fase, assicurati di aggiornare il piano di utilizzo per collegare la nuova fase API.

Il client può ora chiamare i metodi che richiedono la chiave API senza fornire esplicitamente una chiave API. La chiave API restituita dall'autorizzazione viene utilizzata automaticamente.

Documentazione delle REST API in Gateway API

Per aiutare i clienti a comprendere e utilizzare la tua API, è fondamentale documentarla. Per aiutarti a documentare l'API, API Gateway ti consente di aggiungere e aggiornare il contenuto della guida per le singole entità API come parte integrante del processo di sviluppo dell'API. API Gateway memorizza il contenuto di origine e ti consente di archiviare diverse versioni della documentazione. Puoi associare una versione della documentazione a una fase API, esportare uno snapshot di documentazione specifico della fase in un file OpenAPI esterno e distribuire il file come pubblicazione della documentazione.

Per documentare l'API, è possibile chiamare la REST API di Gateway API, utilizzare un AWS SDK o AWS CLI per Gateway API oppure usare la console Gateway API. Inoltre, puoi importare o esportare le parti della documentazione definite in un file OpenAPI esterno.

Per condividere la documentazione API con gli sviluppatori, puoi utilizzare un portale per sviluppatori. Per un esempio, consulta Integrazione ReadMe con API Gateway per mantenere aggiornato il tuo Developer Hub o Come semplificare lo sviluppo di API su Amazon API Gateway Using SmartBear SwaggerHub sul blog AWS Partner Network (APN).

Argomenti

- Rappresentazione della documentazione dell'API in API Gateway
- Documentare un'API utilizzando la console API Gateway
- Pubblicare la documentazione dell'API utilizzando la console API Gateway
- Documentare un'API utilizzando l'API REST di API Gateway
- Pubblicare la documentazione dell'API utilizzando l'API REST di API Gateway
- Importare la documentazione dell'API
- Controllo dell'accesso alla documentazione dell'API in Gateway API

Rappresentazione della documentazione dell'API in API Gateway

La documentazione dell'API in API Gateway è costituita da singole parti associate a specifiche entità API che includono API, risorsa, metodo, richiesta, risposta parametri del messaggio (ad esempio, percorso, query, intestazione), nonché autorizzazioni e modelli.

In API Gateway, una parte della documentazione è rappresentata da una DocumentationPartrisorsa. L'intera documentazione dell'API è rappresentata dalla DocumentationPartsraccolta.

La documentazione di un'API implica la creazione di istanze di `DocumentationPart`, aggiungendole alla raccolta `DocumentationParts` e mantenendo le versioni delle parti della documentazione man mano che l'API evolve.

Argomenti

- Parti della documentazione
- Versioni della documentazione

Parti della documentazione

Una DocumentationPartrisorsa è un oggetto JSON che memorizza il contenuto della documentazione applicabile a una singola entità API. Ciò include qualsiasi UTF-8 contenuto e tutte le principali lingue di localizzazione per la documentazione. Il campo `properties` contiene il contenuto della documentazione sotto forma di mappa di coppie chiave-valore. La proprietà `location` identifica l'entità API associata.

La forma della mappa di contenuti è determinata da te, lo sviluppatore dell'API. Il valore di una coppia chiave-valore può essere una stringa, un numero, un booleano, un oggetto o una matrice. La forma dell'oggetto `location` dipende dal tipo di entità di destinazione.

La risorsa `DocumentationPart` supporta l'ereditarietà del contenuto: il contenuto della documentazione di un'entità API è applicabile agli elementi figlio dell'entità API. Per ulteriori informazioni sulla definizione di entità figlio ed ereditarietà del contenuto, consulta Ereditare il contenuto da un'entità API di più specifiche generali.

Posizione di una parte della documentazione

La proprietà location di un'DocumentationPartistanza identifica un'entità API a cui si applica il contenuto associato. L'entità API può essere una risorsa API REST di API Gateway, ad esempio Resource RestApi, Method MethodResponse, Authorizer o Model. L'entità può anche essere un

parametro di messaggio, ad esempio un parametro di percorso URL, un parametro di stringa di query, un parametro di intestazione di richiesta o risposta, un corpo di richiesta o risposta o un codice di stato di risposta.

Per specificare un'entità API, impostare l'attributo type dell'oggetto `location` su uno dei seguenti valori `API`, `AUTHORIZER`, `MODEL`, `RESOURCE`, `METHOD`, `PATH_PARAMETER`, `QUERY_PARAMETER`, `REQUEST_HEADER`, `REQUEST_BODY`, `RESPONSE`, `RESPONSE_HEADER` o `RESPONSE_BODY`.

A seconda dell'attributo `type` di un'entità API, è possibile specificare altri attributi `location`, tra cui method, name, path e statusCode. Non tutti questi attributi sono validi per una determinata entità API. Ad esempio, `type`, `path`, `name` e `statusCode` sono attributi validi dell'entità `RESPONSE`, solo `type` e `path` sono attributi di posizione validi dell'entità `RESOURCE`. Non è corretto includere un campo non valido nell'attributo `location` di `DocumentationPart` per una determinata entità API.

Non tutti i campi `location` validi sono obbligatori. Ad esempio, `type` è il campo `location` valido e obbligatorio di tutte le entità API. Tuttavia, `method`, `path` e `statusCode` sono attributi validi ma non obbligatori per l'entità `RESPONSE`. Se non è specificato esplicitamente, un campo `location` valido assume il valore predefinito. Il valore predefinito di `path` è `/`, vale a dire la risorsa radice di un'API. Il valore predefinito di `method` o `statusCode` è `*`, ossia qualsiasi valore di metodo o codice di stato, rispettivamente.

Contenuto di una parte della documentazione

Il valore di `properties` è codificato come stringa JSON. Il valore `properties` contiene tutte le informazioni che hai scelto per soddisfare i tuoi requisiti di documentazione. Ad esempio, la seguente è una mappa di contenuti valida:

```
{
  "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."
}
```

Anche se API Gateway accetta qualsiasi stringa JSON valida come mappa di contenuti, gli attributi del contenuto sono trattati come due categorie: quelli che possono e quelli che non possono essere riconosciuti da OpenAPI. Nell'esempio precedente, `info`, `description` e `x-custom-info` sono riconosciuti da OpenAPI come oggetto, proprietà o estensione OpenAPI standard. Di contro, `my-`

`info` non è conforme alle specifiche OpenAPI. API Gateway propaga gli attributi di OpenAPI-compliant contenuto nelle definizioni delle entità API dalle `DocumentationPart` istanze associate. API Gateway non propaga gli attributi di contenuto non conformi nelle definizioni di entità API.

Per un altro esempio, ecco un `DocumentationPart` mirato per un'entità `Resource`:

```
{
    "location" : {
        "type" : "RESOURCE",
        "path": "/pets"
    },
    "properties" : {
        "summary" : "The /pets resource represents a collection of pets in PetStore.",
        "description": "... a child resource under the root...",
    }
}
```

Qui, `type` e `path` sono entrambi campi validi per l'identificazione della destinazione del tipo `RESOURCE`. Per la risorsa radice (`/`), puoi omettere il campo `path`.

```
{
    "location" : {
        "type" : "RESOURCE"
    },
    "properties" : {
        "description" : "The root resource with the default path specification."
    }
}
```

È equivalente alla seguente istanza di `DocumentationPart`:

```
{
    "location" : {
        "type" : "RESOURCE",
        "path": "/"
    },
    "properties" : {
        "description" : "The root resource with an explicit path specification"
    }
}
```

Ereditare il contenuto da un'entità API di specifiche più generali

Il valore predefinito di un campo `location` facoltativo fornisce una descrizione di modello di un'entità API. Usando il valore predefinito dell'oggetto `location`, puoi aggiungere una descrizione generale nella mappa di `properties` a un'istanza di `DocumentationPart` con questo tipo di modello `location`. API Gateway estrae gli attributi della documentazione OpenAPI applicabili da `DocumentationPart` dell'entità API generica e li inserisce in un'entità API specifica con i campi `location` corrispondenti al modello `location` generale o corrispondenti al valore esatto, a meno che l'entità specifica non disponga già di un'istanza `DocumentationPart` associata. Questo comportamento è anche noto come ereditarietà del contenuto da un'entità API con specifiche più generali.

L'ereditarietà del contenuto non si applica a determinati tipi di entità API. Per informazioni dettagliate, consulta la tabella seguente.

Quando un'entità API corrisponde a più di un modello di posizione di `DocumentationPart`, l'entità erediterà la parte della documentazione con i campi di posizione con la precedenza e le specificità più elevate. L'ordine di precedenza è `path` > `statusCode`. Per la corrispondenza con il campo `path`, API Gateway sceglie l'entità con il valore del percorso più specifico. La seguente tabella mostra questo comportamento con alcuni esempi.

Caso	**path**	**state**	**name**	Remar
1	`/pets`	`*`	`id`	La documentazione associata a questo modello di posizione verrà ereditata dalle entità che

Caso	`path`	`state`	`name`	Remar
				corrispondono al modello di posizione.

Caso	**path**	**state**	**name**	Remar
2	/pets	200	id	La documentazione associata a questo modello di posizione verrà ereditata dalle entità che corrispondono al modello di posizione in caso di corrispondenza con il caso 1 e il caso 2, in quanto il caso

Caso	`path`	`state`	`name`	Remar
				2 è più specifico del caso 1.

Caso	**path**	**state**	**name**	Remar
3	`/pets/petId`	`*`	`id`	La documentazione associata a questo modello di posizione verrà ereditata dalle entità che corrispondono al modello di posizione quando i casi 1, 2 e 3 sono corrispondenti, poiché il caso 3 ha una precedenz

Caso	**path**	**state**	**name**	Remar
				a maggiore rispetto al caso 2 ed è più specifico del caso 1.

Ecco un altro esempio per contrapporre un'istanza di `DocumentationPart` più generica con una più specifica. Il seguente messaggio di errore generale `"Invalid request error"` è inserito nelle definizioni di OpenAPI delle risposte dell'errore `400`, se non sostituite.

```
{
    "location" : {
        "type" : "RESPONSE",
        "statusCode": "400"
    },
    "properties" : {
        "description" : "Invalid request error."
    }"
}
```

Con la seguente sostituzione, l'errore `400` che risponde ai metodi della risorsa `/pets` ha la descrizione `"Invalid petId specified"`.

```
{
    "location" : {
        "type" : "RESPONSE",
        "path": "/pets",
        "statusCode": "400"
    },
    "properties" : "{
```

```
        "description" : "Invalid petId specified."
    }"
}
```

Campi di posizione di valid **`DocumentationPart`**

La tabella seguente mostra i campi validi e obbligatori, nonché i valori predefiniti applicabili di una DocumentationPartrisorsa associata a un determinato tipo di entità API.

Entità API	Campi di posizione validi	Campi di posizione obbligatori	Valori dei campi predefiniti	Contenuto ereditabile
API	`{ "location": { "type": "API" }, ... }`	`type`	N/A	No
Risorsa	`{ "location": { "type": "RESOURCE", "path": "resource_path " }, ... }`	`type`	Il valore predefinito di `path` è `/`.	No
Metodo	`{ "location": { "type": "METHOD", "path": "resource_path ", "method": "http_verb " }, ...`	`type`	I valori predefiniti di `path` e `method` sono `/` e `*`, rispettivamente.	Sì, con corrispondenza di `path` per il prefisso e corrispondenza di `method` per

Entità API	Campi di posizione validi	Campi di posizione obbligatori	Valori dei campi predefiniti	Contenuto ereditabile
	`}`			qualsiasi valore.
Parametro di query	`{ "location": { "type": "QUERY_PARAMETER", "path": "resource_path ", "method": "HTTP_verb ", "name": "query_parameter_name " }, ... }`	`type`	I valori predefiniti di `path` e `method` sono `/` e `*`, rispettivamente.	Sì, con corrispondenza di `path` per il prefisso e corrispondenza di `method` per i valori esatti.
Corpo di richiesta	`{ "location": { "type": "REQUEST_BODY", "path": "resource_path ", "method": "http_verb " }, ... }`	`type`	I valori predefiniti di `path` e `method` sono `/` e `*`, rispettivamente.	Sì, con corrispondenza di `path` per il prefisso e corrispondenza di `method` per i valori esatti.

Entità API	Campi di posizione validi	Campi di posizione obbligatori	Valori dei campi predefiniti	Contenuto ereditabile
Parametro di intestazione di richiesta	`{ "location": { "type": "REQUEST_HEADER", "path": "resource_path ", "method": "HTTP_verb ", "name": "header_name " }, ... }`	`type`, `name`	I valori predefiniti di `path` e `method` sono `/` e `*`, rispettivamente.	Sì, con corrispondenza di `path` per il prefisso e corrispondenza di `method` per i valori esatti.
Parametro di percorso richiesta	`{ "location": { "type": "PATH_PARAMETER", "path": "resource/{path_parameter_name }", "method": "HTTP_verb ", "name": "path_parameter_name " }, ... }`	`type`, `name`	I valori predefiniti di `path` e `method` sono `/` e `*`, rispettivamente.	Sì, con corrispondenza di `path` per il prefisso e corrispondenza di `method` per i valori esatti.

Entità API	Campi di posizione validi	Campi di posizione obbligatori	Valori dei campi predefiniti	Contenuto ereditabile
Risposta	`{ "location": { "type": "RESPONSE", "path": "resource_path", "method": "http_verb", "statusCode": "status_code" }, ... }`	`type`	I valori predefiniti di `path`, `method` e `statusCode` sono `/`, `*` e `*`, rispettivamente.	Sì, con corrispondenza di `path` per il prefisso e corrispondenza di `method` e `statusCode` per i valori esatti.
Intestazione della risposta	`{ "location": { "type": "RESPONSE_HEADER", "path": "resource_path", "method": "http_verb", "statusCode": "status_code", "name": "header_name" }, ... }`	`type`, `name`	I valori predefiniti di `path`, `method` e `statusCode` sono `/`, `*` e `*`, rispettivamente.	Sì, con corrispondenza di `path` per il prefisso e corrispondenza di `method` e `statusCode` per i valori esatti.

Entità API	Campi di posizione validi	Campi di posizione obbligatori	Valori dei campi predefiniti	Contenuto ereditabile
Corpo di risposta	`{ "location": { "type": "RESPONSE_BODY", "path": "resource_path ", "method": "http_verb ", "statusCode": "status_code " }, ... }`	`type`	I valori predefiniti di `path`, `method` e `statusCode` sono `/`, `*` e `*`, rispettivamente.	Sì, con corrispondenza di `path` per il prefisso e corrispondenza di `method` e `statusCode` per i valori esatti.
Authorizer	`{ "location": { "type": "AUTHORIZER", "name": "authorizer_name " }, ... }`	`type`	N/A	No
Modello	`{ "location": { "type": "MODEL", "name": "model_name " }, ... }`	`type`	N/A	No

Versioni della documentazione

Una versione della documentazione è un'istantanea della DocumentationPartsraccolta di un'API ed è contrassegnata con un identificatore di versione. La pubblicazione della documentazione di un'API implica la creazione di una versione della documentazione, l'associazione con una fase API e l'esportazione della versione specifica della fase della documentazione dell'API in un file OpenAPI esterno. In API Gateway, un'istantanea della documentazione è rappresentata come una DocumentationVersionrisorsa.

Quando aggiorni un'API, crei nuove versioni dell'API. In API Gateway, gestisci tutte le versioni della documentazione utilizzando la DocumentationVersionsraccolta.

Documentare un'API utilizzando la console API Gateway

In questa sezione, viene descritto come creare e gestire la parti della documentazione di un'API tramite la console API Gateway.

Un prerequisito per la creazione e la modifica della documentazione di un'API è che l'API deve essere già stata creata. In questa sezione, utilizziamo l'PetStoreAPI come esempio. Per creare un'API utilizzando la console API Gateway, seguire le istruzioni in Tutorial: creazione di un'API REST mediante l'importazione di un esempio.

Argomenti

- Documenta l'entità API
- Documenta un'entità RESOURCE
- Documenta un'entità METHOD
- Documenta un'entità QUERY_PARAMETER
- Documenta un'entità PATH_PARAMETER
- Documenta un'entità REQUEST_HEADER
- Documenta un'entità REQUEST_BODY
- Documenta un'entità RESPONSE
- Documenta un'entità RESPONSE_HEADER
- Documenta un'entità RESPONSE_BODY
- Documenta un'entità MODEL
- Documenta un'entità AUTHORIZER

Documenta l'entità **`API`**

Per aggiungere una nuova parte della documentazione per l'entità `API`, procedi come segue:

1. Nel riquadro di navigazione principale scegli Documentazione, quindi seleziona Crea parte della documentazione.
2. Per Tipo di documentazione seleziona API.

   Se una parte della documentazione non è stata creata per l'`API`, viene visualizzato l'editor della mappa di `properties` della parte della documentazione. Inserisci la seguente mappa di `properties` nell'editor di testo.

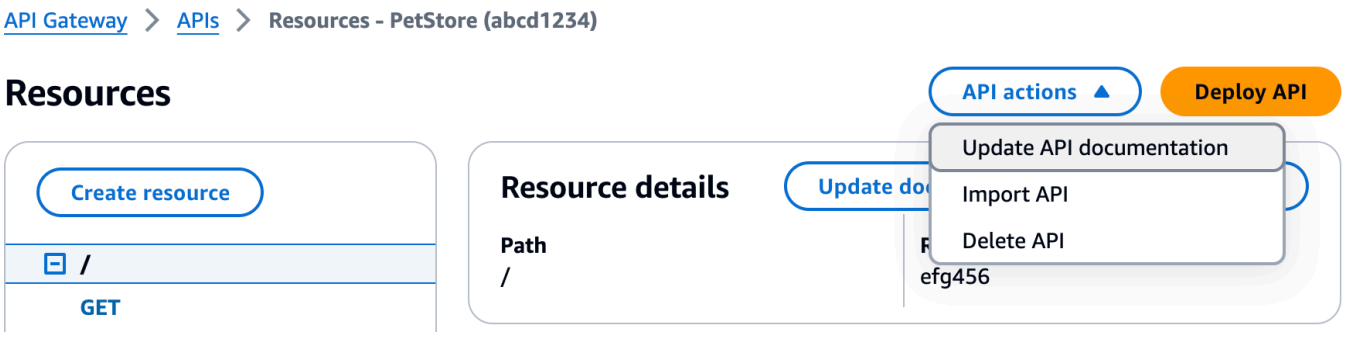
   ```
   {
     "info": {
       "description": "Your first API Gateway API.",
       "contact": {
           "name": "John Doe",
           "email": "john.doe@api.com"
       }
     }
   }
   ```

   > **Note**
   >
   > Non è necessario codificare la mappa `properties` in una stringa JSON. La console API Gateway trasforma in stringa l'oggetto JSON per tuo conto.

3. Scegli Crea parte della documentazione.

Per aggiungere una nuova parte della documentazione per l'entità `API` nel riquadro Risorse, procedi come segue:

1. Nel riquadro di navigazione principale scegli Risorse.
2. Scegli il menu Operazioni API, quindi seleziona Aggiorna documentazione dell'API.



Per modificare una parte della documentazione esistente, procedi come segue:

1. Nel riquadro Documentazione scegli la scheda Risorse e metodi.
2. Seleziona il nome dell'API, quindi nella scheda dell'API scegli Modifica.

Documenta un'entità **`RESOURCE`**

Per aggiungere una nuova parte della documentazione per un'entità `RESOURCE`, procedi come segue:

1. Nel riquadro di navigazione principale scegli Documentazione, quindi seleziona Crea parte della documentazione.
2. Per Tipo di documentazione seleziona Risorsa.
3. Per Percorso inserisci un percorso.
4. Immetti una descrizione nell'editor di testo, ad esempio:

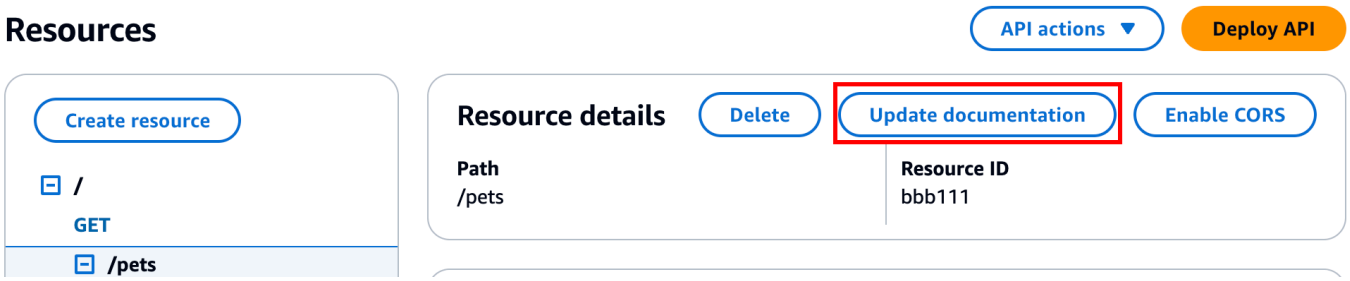
```
{
    "description": "The PetStore's root resource."
}
```

5. Scegli Crea parte della documentazione. È possibile creare la documentazione per una risorsa non elencata.
6. Se necessario, ripeti queste fasi per aggiungere o modificare un'altra parte della documentazione.

Per aggiungere una nuova parte della documentazione per un'entità `RESOURCE` nel riquadro Risorse, procedi come segue:

1. Nel riquadro di navigazione principale scegli Risorse.

2. Scegli la risorsa, quindi seleziona Aggiorna documentazione.



Per modificare una parte della documentazione esistente, procedi come segue:

1. Nel riquadro Documentazione scegli la scheda Risorse e metodi.
2. Seleziona la risorsa contenente la parte della documentazione, quindi scegli Modifica.

Documenta un'entità `METHOD`

Per aggiungere una nuova parte della documentazione per un'entità `METHOD`, procedi come segue:

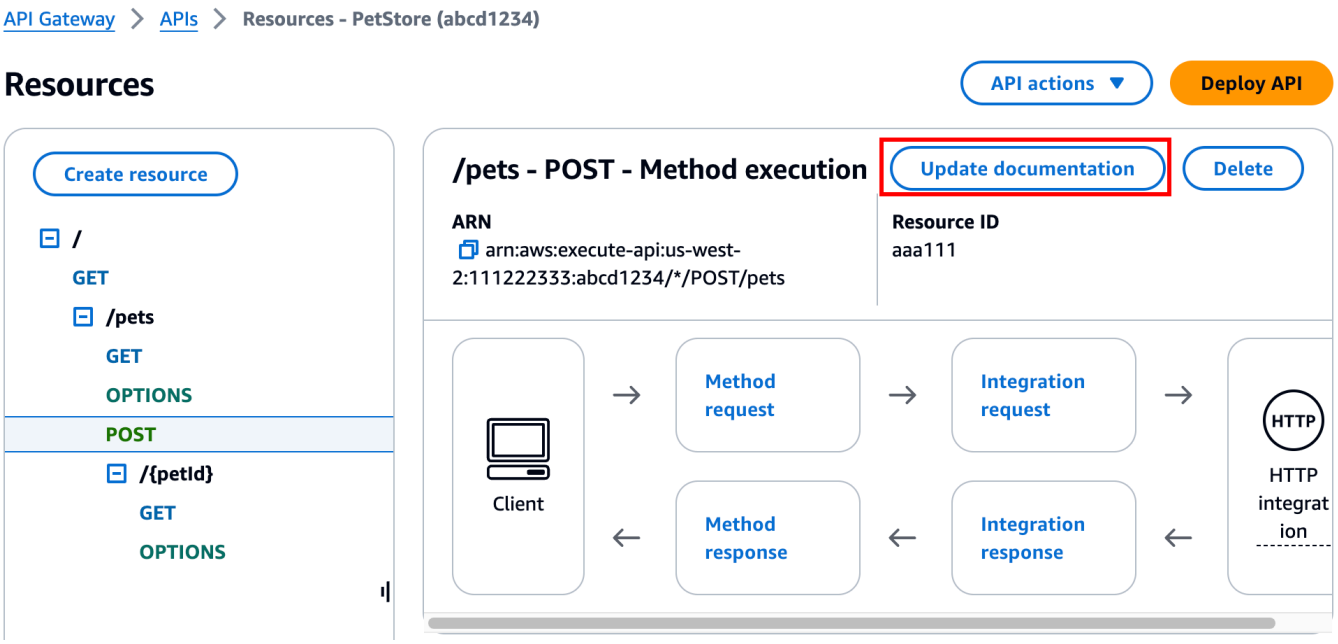
1. Nel riquadro di navigazione principale scegli Documentazione, quindi seleziona Crea parte della documentazione.
2. Per Tipo di documentazione seleziona Metodo.
3. Per Percorso inserisci un percorso.
4. Per Metodo seleziona un verbo HTTP.
5. Immetti una descrizione nell'editor di testo, ad esempio:

```
{
  "tags" : [ "pets" ],
  "summary" : "List all pets"
}
```

6. Scegli Crea parte della documentazione. È possibile creare la documentazione per un metodo non elencato.
7. Se necessario, ripeti queste fasi per aggiungere o modificare un'altra parte della documentazione.

Per aggiungere una nuova parte della documentazione per un'entità `METHOD` nel riquadro Risorse, procedi come segue:

1. Nel riquadro di navigazione principale scegli Risorse.
2. Scegli il metodo, quindi seleziona Aggiorna documentazione.



Per modificare una parte della documentazione esistente, procedi come segue:

1. Nel riquadro Documentazione scegli la scheda Risorse e metodi.
2. È possibile selezionare il metodo o la risorsa contenente il metodo, quindi utilizzare la barra di ricerca per trovare e scegliere la parte della documentazione.
3. Scegli Modifica.

Documenta un'entità `QUERY_PARAMETER`

Per aggiungere una nuova parte della documentazione per un'entità `QUERY_PARAMETER`, procedi come segue:

1. Nel riquadro di navigazione principale scegli Documentazione, quindi seleziona Crea parte della documentazione.
2. Per Tipo di documentazione seleziona Parametro di query.

3. Per Percorso inserisci un percorso.
4. Per Metodo seleziona un verbo HTTP.
5. In Nome, immetti un nome.
6. Immetti una descrizione nell'editor di testo.
7. Scegli Crea parte della documentazione. È possibile creare la documentazione per un parametro di query non elencato.
8. Se necessario, ripeti queste fasi per aggiungere o modificare un'altra parte della documentazione.

Per modificare una parte della documentazione esistente, procedi come segue:

1. Nel riquadro Documentazione scegli la scheda Risorse e metodi.
2. È possibile selezionare il parametro di query o la risorsa contenente il parametro di query, quindi utilizzare la barra di ricerca per trovare e scegliere la parte della documentazione.
3. Scegli Modifica.

Documenta un'entità PATH_PARAMETER

Per aggiungere una nuova parte della documentazione per un'entità `PATH_PARAMETER`, procedi come segue:

1. Nel riquadro di navigazione principale scegli Documentazione, quindi seleziona Crea parte della documentazione.
2. Per Tipo di documentazione seleziona Parametro di percorso.
3. Per Percorso inserisci un percorso.
4. Per Metodo seleziona un verbo HTTP.
5. In Nome, immetti un nome.
6. Immetti una descrizione nell'editor di testo.
7. Scegli Crea parte della documentazione. È possibile creare la documentazione per un parametro di percorso non elencato.
8. Se necessario, ripeti queste fasi per aggiungere o modificare un'altra parte della documentazione.

Per modificare una parte della documentazione esistente, procedi come segue:

1. Nel riquadro Documentazione scegli la scheda Risorse e metodi.
2. È possibile selezionare il parametro di percorso o la risorsa contenente il parametro di percorso, quindi utilizzare la barra di ricerca per trovare e scegliere la parte della documentazione.
3. Scegli Modifica.

Documenta un'entità REQUEST_HEADER

Per aggiungere una nuova parte della documentazione per un'entità `REQUEST_HEADER`, procedi come segue:

1. Nel riquadro di navigazione principale scegli Documentazione, quindi seleziona Crea parte della documentazione.
2. Per Tipo di documentazione seleziona Intestazione della richiesta.
3. Per Percorso inserisci il percorso dell'intestazione della richiesta.
4. Per Metodo seleziona un verbo HTTP.
5. In Nome, immetti un nome.
6. Immetti una descrizione nell'editor di testo.
7. Scegli Crea parte della documentazione. È possibile creare la documentazione per un'intestazione della richiesta non elencata.
8. Se necessario, ripeti queste fasi per aggiungere o modificare un'altra parte della documentazione.

Per modificare una parte della documentazione esistente, procedi come segue:

1. Nel riquadro Documentazione scegli la scheda Risorse e metodi.
2. È possibile selezionare l'intestazione della richiesta o la risorsa contenente l'intestazione della richiesta, quindi utilizzare la barra di ricerca per trovare e scegliere la parte della documentazione.
3. Scegli Modifica.

Documenta un'entità REQUEST_BODY

Per aggiungere una nuova parte della documentazione per un'entità `REQUEST_BODY`, procedi come segue:

1. Nel riquadro di navigazione principale scegli Documentazione, quindi seleziona Crea parte della documentazione.
2. Per Tipo di documentazione seleziona Corpo della richiesta.
3. Per Percorso inserisci il percorso del corpo della richiesta.
4. Per Metodo seleziona un verbo HTTP.
5. Immetti una descrizione nell'editor di testo.
6. Scegli Crea parte della documentazione. È possibile creare la documentazione per un corpo della richiesta non elencato.
7. Se necessario, ripeti queste fasi per aggiungere o modificare un'altra parte della documentazione.

Per modificare una parte della documentazione esistente, procedi come segue:

1. Nel riquadro Documentazione scegli la scheda Risorse e metodi.
2. È possibile selezionare il corpo della richiesta o la risorsa contenente il corpo della richiesta, quindi utilizzare la barra di ricerca per trovare e scegliere la parte della documentazione.
3. Scegli Modifica.

Documenta un'entità RESPONSE

Per aggiungere una nuova parte della documentazione per un'entità `RESPONSE`, procedi come segue:

1. Nel riquadro di navigazione principale scegli Documentazione, quindi seleziona Crea parte della documentazione.
2. Per Tipo di documentazione seleziona Risposta (codice di stato).
3. Per Percorso inserisci un percorso per la risposta.
4. Per Metodo seleziona un verbo HTTP.
5. Per Codice di stato inserisci un codice di stato HTTP.
6. Immetti una descrizione nell'editor di testo.
7. Scegli Crea parte della documentazione. È possibile creare la documentazione per un codice di stato della risposta non elencato.
8. Se necessario, ripeti queste fasi per aggiungere o modificare un'altra parte della documentazione.

Per modificare una parte della documentazione esistente, procedi come segue:

1. Nel riquadro Documentazione scegli la scheda Risorse e metodi.
2. È possibile selezionare il codice di stato della risposta o la risorsa contenente il codice di stato della risposta, quindi utilizzare la barra di ricerca per trovare e scegliere la parte della documentazione.
3. Scegli Modifica.

Documenta un'entità **RESPONSE_HEADER**

Per aggiungere una nuova parte della documentazione per un'entità `RESPONSE_HEADER`, procedi come segue:

1. Nel riquadro di navigazione principale scegli Documentazione, quindi seleziona Crea parte della documentazione.
2. Per Tipo di documentazione seleziona Intestazione della risposta.
3. Per Percorso inserisci un percorso per l'intestazione della risposta.
4. Per Metodo seleziona un verbo HTTP.
5. Per Codice di stato inserisci un codice di stato HTTP.
6. Immetti una descrizione nell'editor di testo.
7. Scegli Crea parte della documentazione. È possibile creare la documentazione per un'intestazione della risposta non elencata.
8. Se necessario, ripeti queste fasi per aggiungere o modificare un'altra parte della documentazione.

Per modificare una parte della documentazione esistente, procedi come segue:

1. Nel riquadro Documentazione scegli la scheda Risorse e metodi.
2. È possibile selezionare l'intestazione della risposta o la risorsa contenente l'intestazione della risposta, quindi utilizzare la barra di ricerca per trovare e scegliere la parte della documentazione.
3. Scegli Modifica.

Documenta un'entità RESPONSE_BODY

Per aggiungere una nuova parte della documentazione per un'entità `RESPONSE_BODY`, procedi come segue:

1. Nel riquadro di navigazione principale scegli Documentazione, quindi seleziona Crea parte della documentazione.
2. Per Tipo di documentazione seleziona Corpo della risposta.
3. Per Percorso inserisci un percorso per il corpo della risposta.
4. Per Metodo seleziona un verbo HTTP.
5. Per Codice di stato inserisci un codice di stato HTTP.
6. Immetti una descrizione nell'editor di testo.
7. Scegli Crea parte della documentazione. È possibile creare la documentazione per un corpo della risposta non elencato.
8. Se necessario, ripeti queste fasi per aggiungere o modificare un'altra parte della documentazione.

Per modificare una parte della documentazione esistente, procedi come segue:

1. Nel riquadro Documentazione scegli la scheda Risorse e metodi.
2. È possibile selezionare il corpo della risposta o la risorsa contenente il corpo della risposta, quindi utilizzare la barra di ricerca per trovare e scegliere la parte della documentazione.
3. Scegli Modifica.

Documenta un'entità MODEL

La documentazione di un'entità `MODEL` comporta la creazione e la gestione delle istanze di `DocumentPart` per il modello e gli elementi `properties` del modello. Ad esempio, per il modello `Error` fornito con ogni API per impostazione predefinita ha la seguente definizione dello schema:

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

```
}
```

e richiede due istanze di `DocumentationPart`, una per `Model` e l'altra per la relativa proprietà `message`:

```
{
  "location": {
    "type": "MODEL",
    "name": "Error"
  },
  "properties": {
    "title": "Error Schema",
    "description": "A description of the Error model"
  }
}
```

e

```
{
  "location": {
    "type": "MODEL",
    "name": "Error.message"
  },
  "properties": {
    "description": "An error message."
  }
}
```

Quando l'API viene esportata, le proprietà di `DocumentationPart` sostituiscono i valori dello schema originale.

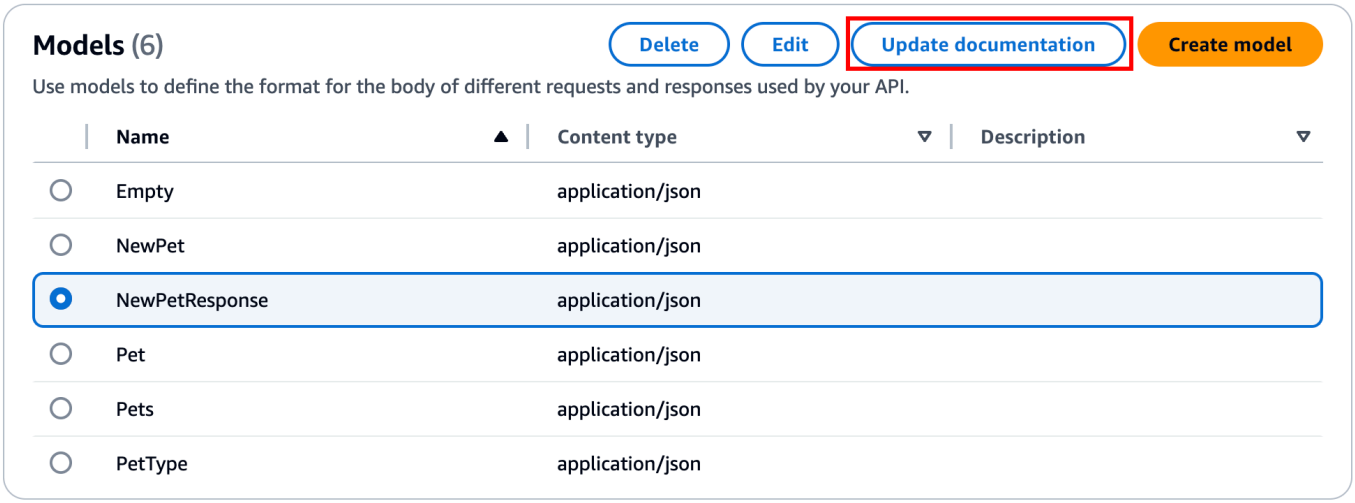
Per aggiungere una nuova parte della documentazione per un'entità `MODEL`, procedi come segue:

1. Nel riquadro di navigazione principale scegli Documentazione, quindi seleziona Crea parte della documentazione.
2. Per Tipo di documentazione seleziona Modello.
3. Per Nome inserisci un nome per il processo.
4. Immetti una descrizione nell'editor di testo.
5. Scegli Crea parte della documentazione. È possibile creare la documentazione per modelli non elencati.

6. Se necessario, ripeti queste fasi per aggiungere o modificare una parte della documentazione per altri modelli.

Per aggiungere una nuova parte della documentazione per un'entità `MODEL` nel riquadro Modelli, procedi come segue:

1. Nel riquadro di navigazione principale seleziona Modelli.
2. Scegli il modello, quindi seleziona Aggiorna documentazione.



Per modificare una parte della documentazione esistente, procedi come segue:

1. Nel riquadro Documentazione scegli la scheda Modelli.
2. Utilizza la barra di ricerca o seleziona il modello, quindi scegli Modifica.

Documenta un'entità **`AUTHORIZER`**

Per aggiungere una nuova parte della documentazione per un'entità `AUTHORIZER`, procedi come segue:

1. Nel riquadro di navigazione principale scegli Documentazione, quindi seleziona Crea parte della documentazione.
2. Per Tipo di documentazione seleziona Sistema di autorizzazione.
3. Per Nome immetti il nome del sistema di autorizzazione.

4. Immetti una descrizione nell'editor di testo. Specifica un valore per il campo `location` valido per il sistema di autorizzazione.
5. Scegli Crea parte della documentazione. È possibile creare la documentazione per sistemi di autorizzazione non elencati.
6. Se necessario, ripeti queste fasi per aggiungere o modificare una parte della documentazione per altre autorizzazioni.

Per modificare una parte della documentazione esistente, procedi come segue:

1. Nel riquadro Documentazione scegli la scheda Sistemi di autorizzazione.
2. Utilizza la barra di ricerca o seleziona il sistema di autorizzazione, quindi scegli Modifica.

Pubblicare la documentazione dell'API utilizzando la console API Gateway

La procedura seguente mostra come pubblicare una versione della documentazione.

Per pubblicare una versione della documentazione utilizzando la console API Gateway

1. Nel riquadro di navigazione principale scegli Documentazione.
2. Seleziona Pubblica documentazione.
3. Configura la pubblicazione:
    a. In Fase, seleziona una fase.
    b. Per Versione inserisci un identificatore di versione, ad esempio `1.0.0`.
    c. (Facoltativo) In Descrizione, immetti una descrizione.
4. Seleziona Publish (Pubblica).

Ora puoi procedere con il download della documentazione pubblicata esportando la documentazione in un file OpenAPI esterno. Per ulteriori informazioni, consulta the section called “Esportazione di un'API REST”.

Documentare un'API utilizzando l'API REST di API Gateway

In questa sezione viene descritto come creare e gestire la parti della documentazione di un'API tramite l'API REST di API Gateway.

Un prerequisito per la creazione e la modifica della documentazione di un'API è aver già creato l'API. In questa sezione, utilizziamo l'PetStoreAPI come esempio. Per creare un'API utilizzando la console API Gateway, seguire le istruzioni in Tutorial: creazione di un'API REST mediante l'importazione di un esempio.

Argomenti

- Documenta l'entità API
- Documenta un'entità RESOURCE
- Documenta un'entità METHOD
- Documenta un'entità QUERY_PARAMETER
- Documenta un'entità PATH_PARAMETER
- Documenta un'entità REQUEST_BODY
- Documenta un'entità REQUEST_HEADER
- Documenta un'entità RESPONSE
- Documenta un'entità RESPONSE_HEADER
- Documenta un'entità AUTHORIZER
- Documenta un'entità MODEL
- Aggiornare le parti della documentazione
- Elencare le parti della documentazione

Documenta l'entità **API**

Per aggiungere la documentazione per un'API, aggiungi una DocumentationPartrisorsa per l'entità 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}"
}
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Ad esempio:

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

Se la documentazione è già stata aggiunta, viene restituita una risposta `409 Conflict` contenente il messaggio di errore `Documentation part already exists for the specified location: type 'API'.`". In questo caso è necessario eseguire l'operazione documentationpart:update.

```
PATCH /restapis/4wk1k4onj3/documentation/parts/part_id HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "/properties",
    "value" : "{\n\t\"info\": {\n\t\t\"description\" : \"Your first API with Amazon API
 Gateway.\"\n\t}\n}"
  } ]
```

```
}
```

La risposta con esito positivo restituisce un codice di stato `200 OK` con un payload che include l'istanza `DocumentationPart` aggiornata nel payload.

Documenta un'entità **`RESOURCE`**

Per aggiungere la documentazione per la risorsa principale di un'API, aggiungi una DocumentationPartrisorsa destinata alla risorsa Resource corrispondente:

```
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}"
}
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Ad esempio:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/p76vqo"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/p76vqo"
    },
    "documentationpart:update": {
```

```
      "href": "/restapis/4wk1k4onj3/documentation/parts/p76vqo"
    }
  },
  "id": "p76vqo",
  "location": {
    "path": "/",
    "method": null,
    "name": null,
    "statusCode": null,
    "type": "RESOURCE"
  },
  "properties": "{\n\t\"description\" : \"The PetStore root resource.\"\n}"
}
```

Quando il percorso della risorsa non è specificato, si presume che la risorsa sia la risorsa radice. Puoi aggiungere `"path": "/"` a `properties` per rendere esplicita la specifica.

Per creare la documentazione per una risorsa figlia di un'API, aggiungi una DocumentationPartrisorsa destinata alla risorsa Resource corrispondente:

```
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}"
}
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Ad esempio:

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

Per aggiungere la documentazione per una risorsa secondaria specificata da un parametro path, aggiungi una DocumentationPartrisorsa destinata alla risorsa Resource:

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "RESOURCE",
         "path" : "/pets/{petId}"
```

```
    },
    "properties": "{\n\t\"description\" : \"A child resource specified by the petId
 path parameter.\"\n}"
}
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Ad esempio:

```
{
  "_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**
>
> L'DocumentationPartistanza di un'RESOURCEentità non può essere ereditata da nessuna delle sue risorse secondarie.

Documenta un'entità **METHOD**

Per aggiungere la documentazione per un metodo di un'API, aggiungi una DocumentationPartrisorsa destinata alla risorsa Method corrispondente:

```
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}"
}
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Ad esempio:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    },
```

```
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    }
  },
  "id": "o64jbj",
  "location": {
    "path": "/pets",
    "method": "GET",
    "name": null,
    "statusCode": null,
    "type": "METHOD"
  },
  "properties": "{\n\t\"summary\" : \"List all pets.\"\n}"
}
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Ad esempio:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64jbj"
    }
  },
  "id": "o64jbj",
  "location": {
    "path": "/pets",
    "method": "GET",
```

```
    "name": null,
    "statusCode": null,
    "type": "METHOD"
  },
  "properties": "{\n\t\"summary\" : \"List all pets.\"\n}"
}
```

Se il campo `location.method` non è specificato nella richiesta precedente, si presume che sia il metodo ANY rappresentato da un carattere jolly `*`.

Per aggiornare il contenuto della documentazione di un'entità `METHOD`, chiamare l'operazione documentationpart:update, fornendo una nuova mappa di `properties`:

```
PATCH /restapis/4wk1k4onj3/documentation/parts/part_id HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "/properties",
    "value" : "{\n\t\"tags\" : [ \"pets\" ], \n\t\"summary\" : \"List all pets.\"\n}"
  } ]
}
```

La risposta con esito positivo restituisce un codice di stato `200 OK` con un payload che include l'istanza `DocumentationPart` aggiornata nel payload. Ad esempio:

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

Documenta un'entità **QUERY_PARAMETER**

Per aggiungere la documentazione per un parametro di query di richiesta, aggiungi una DocumentationPartrisorsa destinata al QUERY_PARAMETER tipo, con i campi validi di and. `path` `name`

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "QUERY_PARAMETER",
         "path" : "/pets",
         "method" : "GET",
         "name" : "page"
    },
    "properties": "{\n\t\"description\" : \"Page number of results to return.\"\n}"
}
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Ad esempio:

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

La mappa di `properties` della parte della documentazione di un'entità `QUERY_PARAMETER` può essere ereditata da uno delle relative entità figlio `QUERY_PARAMETER`. Ad esempio se aggiungi una risorsa `treats` dopo `/pets/{petId}`, abiliti il metodo `GET` su `/pets/{petId}/treats` ed esponi il parametro di query `page`, questo parametro di query figlio eredita la mappa di `DocumentationPart` di `properties` dal parametro di query con nome simile del metodo `GET /pets`, a meno che non aggiungi esplicitamente una risorsa `DocumentationPart` al parametro di query `page` del metodo `GET /pets/{petId}/treats`.

Documenta un'entità **PATH_PARAMETER**

Per aggiungere la documentazione per un parametro di percorso, aggiungi una DocumentationPartrisorsa per l'PATH_PARAMETERentità.

```
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}"
}
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Ad esempio:

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

Documenta un'entità **`REQUEST_BODY`**

Per aggiungere documentazione per un corpo della richiesta, aggiungi una DocumentationPartrisorsa per il corpo della richiesta.

```
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}"
}
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Ad esempio:

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

Documenta un'entità **REQUEST_HEADER**

Per aggiungere documentazione per un'intestazione di richiesta, aggiungi una DocumentationPartrisorsa per l'intestazione della richiesta.

```
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}"
}
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Ad esempio:

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

Documenta un'entità RESPONSE

Per aggiungere la documentazione per una risposta a un codice di stato, aggiungi una DocumentationPartrisorsa destinata alla MethodResponserisorsa corrispondente.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
```

```
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location": {
      "path": "/",
      "method": "*",
      "name": null,
      "statusCode": "200",
      "type": "RESPONSE"
    },
    "properties": "{\n  \"description\" : \"Successful operation.\"\n}"
}
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Ad esempio:

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

Documenta un'entità **RESPONSE_HEADER**

Per aggiungere la documentazione per un'intestazione di risposta, aggiungi una DocumentationPartrisorsa per l'intestazione della risposta.

```
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}"
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Ad esempio:

```
{
  "_links": {
    "curies": {
      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
      "name": "documentationpart",
      "templated": true
    },
    "self": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/fev7j7"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/fev7j7"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/fev7j7"
    }
  },
```

```
  "id": "fev7j7",
  "location": {
    "path": "/",
    "method": "GET",
    "name": "Content-Type",
    "statusCode": "200",
    "type": "RESPONSE_HEADER"
  },
  "properties": "{\n  \"description\" : \"Media type of request\"\n}"
}
```

La documentazione di questa intestazione di risposta `Content-Type` è la documentazione predefinita per le intestazioni `Content-Type` di qualsiasi risposta dell'API.

Documenta un'entità AUTHORIZER

Per aggiungere la documentazione per un autorizzatore API, aggiungi una DocumentationPartrisorsa destinata all'autorizzatore specificato.

```
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}"
}
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Ad esempio:

```
{
  "_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**
>
> L'DocumentationPartistanza di un'AUTHORIZERentità non può essere ereditata da nessuna delle sue risorse secondarie.

Documenta un'entità **`MODEL`**

La documentazione di un'entità `MODEL` comporta la creazione e la gestione delle istanze di `DocumentPart` per il modello e gli elementi `properties` del modello. Ad esempio, per il modello `Error` fornito con ogni API per impostazione predefinita ha la seguente definizione dello schema:

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "Error Schema",
```

```
  "type" : "object",
  "properties" : {
    "message" : { "type" : "string" }
  }
}
```

e richiede due istanze di `DocumentationPart`, una per `Model` e l'altra per la relativa proprietà `message`:

```
{
  "location": {
    "type": "MODEL",
    "name": "Error"
  },
  "properties": {
    "title": "Error Schema",
    "description": "A description of the Error model"
  }
}
```

e

```
{
  "location": {
    "type": "MODEL",
    "name": "Error.message"
  },
  "properties": {
    "description": "An error message."
  }
}
```

Quando l'API viene esportata, le proprietà di `DocumentationPart` sostituiscono i valori dello schema originale.

Per aggiungere la documentazione per un modello API, aggiungi una DocumentationPartrisorsa destinata al modello specificato.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
```

```
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "MODEL",
         "name" : "Pet"
    },
    "properties": "{\n\t\"description\" : \"Data structure of a Pet object.\"\n}"
}
```

Se con esito positivo, l'operazione restituisce una risposta `201 Created` contenente la nuova istanza `DocumentationPart` creata nel payload. Ad esempio:

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

Ripetete lo stesso passaggio per creare un' DocumentationPart istanza per una qualsiasi delle proprietà del modello.

> **Note**
>
> L'DocumentationPartistanza di un'`MODEL`entità non può essere ereditata da nessuna delle sue risorse secondarie.

Aggiornare le parti della documentazione

Per aggiornare le parti della documentazione di qualsiasi tipo di entità API, invia una richiesta PATCH su un'DocumentationPartistanza di un identificatore di parte specificato per sostituire la `properties` mappa esistente con una nuova.

```
PATCH /restapis/4wk1k4onj3/documentation/parts/part_id HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "RESOURCE_PATH",
    "value" : "NEW_properties_VALUE_AS_JSON_STRING"
  } ]
}
```

La risposta con esito positivo restituisce un codice di stato `200 OK` con un payload che include l'istanza `DocumentationPart` aggiornata nel payload.

Puoi aggiornare più parti della documentazione in una singola richiesta `PATCH`.

Elencare le parti della documentazione

Per elencare le parti della documentazione di qualsiasi tipo di entità API, invia una richiesta GET su una DocumentationPartsraccolta.

```
GET /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret
```

La risposta con esito positivo restituisce un codice di stato `200 OK` con un payload che include le istanze `DocumentationPart` disponibili nel payload.

Pubblicare la documentazione dell'API utilizzando l'API REST di API Gateway

Per pubblicare la documentazione per un'API, crea, aggiorna o ottieni uno snapshot della documentazione, quindi associa lo snapshot della documentazione a una fase dell'API. Quando crei uno snapshot della documentazione, puoi anche contemporaneamente associarlo a una fase dell'API.

Argomenti

- Creare uno snapshot della documentazione e associarlo a una fase dell'API
- Creare uno snapshot della documentazione
- Aggiornare uno snapshot della documentazione
- Ottenere uno snapshot della documentazione
- Associare uno snapshot della documentazione a una fase dell'API
- Scaricare uno snapshot della documentazione associato a una fase

Creare uno snapshot della documentazione e associarlo a una fase dell'API

Per creare uno snapshot delle parti della documentazione di un'API e associarlo contemporaneamente a una fase dell'API, invia la seguente richiesta `POST`:

```
POST /restapis/restapi_id/documentation/versions HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret
```

```
{
    "documentationVersion" : "1.0.0",
    "stageName": "prod",
    "description" : "My API Documentation v1.0.0"
}
```

Se con esito positivo, l'operazione restituisce una risposta `200 OK` contenente la nuova istanza `DocumentationVersion` creata come payload.

In alternativa, è possibile creare uno snapshot della documentazione senza associarlo prima a una fase dell'API e quindi chiamare restapi:update per associare lo snapshot a una fase dell'API specificata. Puoi inoltre aggiornare o interrogare uno snapshot della documentazione esistente e quindi aggiornare la relativa associazione della fase. Le fasi vengono illustrate nelle prossime quattro sezioni.

Creare uno snapshot della documentazione

Per creare un'istantanea delle parti della documentazione di un'API, crea una nuova DocumentationVersionrisorsa e aggiungila alla DocumentationVersionsraccolta dell'API:

```
POST /restapis/restapi_id/documentation/versions HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "documentationVersion" : "1.0.0",
    "description" : "My API Documentation v1.0.0"
}
```

Se con esito positivo, l'operazione restituisce una risposta `200 OK` contenente la nuova istanza `DocumentationVersion` creata come payload.

Aggiornare uno snapshot della documentazione

È possibile aggiornare un'istantanea della documentazione solo modificando la `description` proprietà della risorsa corrispondente. DocumentationVersion L'esempio seguente mostra come

aggiornare la descrizione dello snapshot della documentazione come identificata dal relativo identificatore di versione, *version*, ad esempio `1.0.0`.

```
PATCH /restapis/restapi_id/documentation/versions/version HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "patchOperations": [{
        "op": "replace",
        "path": "/description",
        "value": "My API for testing purposes."
    }]
}
```

Se con esito positivo, l'operazione restituisce una risposta `200 OK` contenente l'istanza `DocumentationVersion` aggiornata come payload.

Ottenere uno snapshot della documentazione

Per ottenere un'istantanea della documentazione, invia una `GET` richiesta relativa alla risorsa specificata. DocumentationVersion L'esempio seguente mostra come ottenere uno snapshot della documentazione di un dato identificatore di versione, 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
```

Associare uno snapshot della documentazione a una fase dell'API

Per pubblicare la documentazione dell'API, associa uno snapshot della documentazione a una fase dell'API. Devi aver già creato la fase dell'API prima di associare la versione della documentazione alla fase.

Per associare uno snapshot della documentazione a una fase dell'API utilizzando l'API REST di API Gateway, esegui l'operazione stage:update per impostare a versione della documentazione desiderata sulla proprietà `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"
    }]
}
```

Scaricare uno snapshot della documentazione associato a una fase

Dopo che una versione della documentazione delle parti è associata a una fase, puoi esportare le parti della documentazione insieme alle definizioni dell'entità API in un file esterno, utilizzando la console API Gateway, l'API REST di API Gateway, uno dei relativi SDK o la AWS CLI per API Gateway. Il processo è uguale a quello utilizzato per esportare l'API. Il formato del file esportato può essere JSON o YAML.

Utilizzando l'API REST di API Gateway, puoi anche impostare esplicitamente il parametro di query `extension=documentation,integrations,authorizers` per includere le parti della documentazione dell'API, le integrazioni API e le autorizzazioni in un'esportazione API. Per impostazione predefinita, le parti della documentazione sono incluse, ma le integrazioni e le autorizzazioni sono escluse quando si esporta un'API. L'output predefinito di un'esportazione API è adatto per la distribuzione della documentazione.

Per esportare la documentazione dell'API in un file OpenAPI JSON esterno usando l'API REST di API Gateway, inviare la seguente richiesta `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
```

Qui, l'oggetto `x-amazon-apigateway-documentation` contiene le parti della documentazione e la definizione dell'entità API contiene le proprietà della documentazione supportate da OpenAPI. L'output non include dettagli sull'integrazione o sulle autorizzazioni Lambda (note in precedenza come autorizzazioni ad hoc). Per includere entrambi i dettagli, imposta `extensions=integrations,authorizers,documentation`. Per includere i dettagli delle integrazioni ma non delle autorizzazioni, imposta `extensions=integrations,documentation`.

Imposta l'intestazione `Accept:application/json` nella richiesta per inserire l'output del risultato in un file JSON. Per produrre l'output YAML, imposta l'intestazione della richiesta su `Accept:application/yaml`.

Ad esempio, esamineremo un'API che espone un semplice metodo `GET` sulla risorsa radice (`/`). Questa API ha quattro entità API definite in un file di definizione OpenAPI, uno per ciascuno dei tipi `API`, `MODEL`, `METHOD` e `RESPONSE`. Una parte della documentazione è stata aggiunta a ciascuna delle entità `API`, `METHOD` e `RESPONSE`. Chiamando il precedente comando di esportazione della documentazione, otteniamo il seguente output, con le parti della documentazione elencate nell'oggetto `x-amazon-apigateway-documentation` come estensione di un file OpenAPI standard.

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"
}
```

Per un OpenAPI-compliant attributo definito nella `properties` mappa di una parte della documentazione, API Gateway inserisce l'attributo nella definizione dell'entità API associata. Un attributo di `x-`*`something`* è un'estensione OpenAPI standard. Questa estensione viene propagata nella definizione dell'entità API. Ad esempio, consulta l'attributo `x-example` per il metodo `GET`. Un attributo come `foo` non fa parte delle specifiche OpenAPI e non viene inserito nelle definizioni delle entità API associate.

Se uno strumento di rendering della documentazione (ad esempio, l'interfaccia utente OpenAPI) analizza le definizioni delle entità API per estrarre gli attributi della documentazione, gli eventuali non OpenAPI-compliant `properties` attributi di un'`DocumentationPart`istanza non sono disponibili per lo strumento. Tuttavia, se uno strumento di rendering della documentazione analizza l'oggetto `x-amazon-apigateway-documentation` per ottenere i contenuti o se lo strumento chiama restapi:documentation-parts e documenationpart:by-id per recuperare le parti della documentazione da API Gateway, tutti gli attributi della documentazione sono disponibili per la visualizzazione con lo strumento.

Per esportare la documentazione con definizioni di entità API contenenti i dettagli delle integrazioni in un file OpenAPI JSON, inviare la seguente richiesta `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
```

Per esportare la documentazione con definizioni di entità API contenenti dettagli delle integrazioni e delle autorizzazioni in un file OpenAPI YAML, inviare la seguente richiesta `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
```

Per usare la console API Gateway per esportare e scaricare la documentazione pubblicata di un'API, segui le istruzioni in Esportazione di un'API REST tramite la console API Gateway.

Importare la documentazione dell'API

Come con l'importazione delle definizioni di entità API, puoi importare le parti della documentazione da un file OpenAPI esterno in un'API di API Gateway. Specificare le parti della documentazione da importare all'interno dell'estensione Oggetto x-amazon-apigateway-documentation in un file di definizione OpenAPI valido. L'importazione della documentazione non modifica le definizioni delle entità API esistenti.

Esiste la possibilità di unire le parti della documentazione appena specificate in parti della documentazione esistenti in API Gateway o per sovrascrivere le parti della documentazione esistenti. Nella modalità `MERGE`, una nuova parte della documentazione definita nel file OpenAPI viene aggiunta alla raccolta `DocumentationParts` dell'API. Se un elemento `DocumentationPart` importato esiste già, un attributo importato sostituisce quello esistente se i due sono diversi. Altri attributi della documentazione esistenti rimangono inalterati. Nella modalità `OVERWRITE`, l'intera raccolta `DocumentationParts` viene sostituita in base al file di definizione OpenAPI importato.

Importazione delle parti della documentazione tramite l'API REST di API Gateway

Per importare la documentazione dell'API usando l'API REST di API Gateway, esegui l'operazione documentationpart:import. L'esempio seguente mostra come sovrascrivere parti della documentazione esistenti di un'API con un singolo metodo `GET /` , restituendo una risposta `200 OK` per l'esito positivo.

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"
        }
      }
    ]
  }
}
```

In caso di esito positivo, questa richiesta restituisce una risposta 200 OK contenente l'elemento `DocumentationPartId` importato nel payload.

```
{
  "ids": [
    "kg3mth",
    "796rtf",
    "zhek4p",
    "5ukm9s"
  ]
}
```

Inoltre, è possibile anche invocare restapi:import o restapi:put, fornendo le parti della documentazione nell'oggetto `x-amazon-apigateway-documentation` come parte del file OpenAPI di input della definizione dell'API. Per escludere le parti della documentazione dall'importazione dell'API, imposta `ignore=documentation` nei parametri di query della richiesta.

Importazione di parti di documentazione tramite la console API Gateway

Le seguenti istruzioni descrivono come importare le parti della documentazione.

Per utilizzare la console per importare le parti della documentazione di un'API da un file esterno

1. Nel riquadro di navigazione principale scegli Documentazione.
2. Scegli Importa.
3. Se è disponibile la documentazione esistente, seleziona Sovrascrivi o Unisci per la nuova documentazione.
4. Seleziona Scegli il file per caricare un file da un'unità oppure immettere il contenuto di un file nella vista del file. Per un esempio, consulta il payload della richiesta di esempio in Importazione delle parti della documentazione tramite l'API REST di API Gateway.

5. Scegli in che modo gestire gli avvisi durante l'importazione. Seleziona Avvisi di errore o Ignora avvisi. Per ulteriori informazioni, consulta the section called “Errori e avvisi relativi all'importazione dell'API in Gateway API”.
6. Scegli Importa.

Controllo dell'accesso alla documentazione dell'API in Gateway API

Se hai un team dedicato alla documentazione per scrivere e modificare la documentazione dell'API, puoi configurare autorizzazioni di accesso separate per i tuoi sviluppatori (per lo sviluppo dell'API) e per i tuoi scrittori o editori (per lo sviluppo del contenuto). Questo è particolarmente appropriato quando un fornitore di terze parti è coinvolto nella creazione della documentazione.

Per concedere al team di documentazione l'accesso per creare, aggiornare e pubblicare la documentazione dell'API, puoi assegnare al team di documentazione un ruolo IAM con la seguente politica IAM, dove si *account_id* trova l'ID AWS account del team di documentazione.

JSON

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

    {
      "Sid": "StmtDocPartsAddEditViewDelete",
      "Effect": "Allow",
      "Action": [
        "apigateway:GET",
        "apigateway:PUT",
        "apigateway:POST",
        "apigateway:PATCH",
        "apigateway:DELETE"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1:111111111111:/restapis/*/documentation/*"
      ]
    }
  ]
}
```

Per informazioni sull'impostazione delle autorizzazioni per accedere alle risorse di API Gateway, consultare the section called “Come funziona Amazon API Gateway con IAM”.

Genera SDKs per REST APIs in API Gateway

Per chiamare la tua API REST in un modo specifico per la piattaforma o la lingua, devi generare l'SDK specifico della piattaforma o della lingua dell'API. Si genera l'SDK dopo aver creato, testato e implementato l'API in una fase. Attualmente, API Gateway supporta la generazione di un SDK per un'API in Java, Java per Android JavaScript, Objective-C o Swift per iOS e Ruby.

In questa sezione viene descritto come generare un SDK di un'API di API Gateway. Dimostra inoltre come utilizzare l'SDK generato in un'app Java, un'app Java per Android, Objective-C e Swift per app iOS e un'app. JavaScript

Per facilitare la discussione, usiamo questa API di API Gateway, che espone questa funzione Lambda calcolatore semplice.

Prima di procedere, crea o importa l'API e distribuiscila almeno una volta in API Gateway. Per istruzioni, consulta Implementazione di REST API in Gateway API.

Argomenti

- Funzione Lambda del calcolatore semplice
- API calcolatore semplice in API Gateway
- Definizione OpenAPI dell'API del calcolatore semplice
- Generazione dell'SDK Java di un'API in Gateway API
- Generazione dell'SDK Android di un'API in Gateway API
- Generazione dell'SDK iOS di un'API in Gateway API
- Genera l' JavaScript SDK di un'API REST in API Gateway
- Generazione dell'SDK Ruby di un'API in Gateway API
- Generazione SDKs per un'API utilizzando AWS CLI i comandi in API Gateway

Funzione Lambda del calcolatore semplice

A scopi illustrativi, verrà utilizzata una funzione Lambda in Node.js che esegue le operazioni binarie di addizione, sottrazione, moltiplicazione e divisione.

Argomenti

- Formato di input della funzione Lambda del calcolatore semplice
- Formato di output della funzione Lambda del calcolatore semplice
- Implementazione della funzione Lambda del calcolatore semplice

Formato di input della funzione Lambda del calcolatore semplice

Questa funzione riceve un input nel formato seguente:

```
{ "a": "Number", "b": "Number", "op": "string"}
```

dove op può essere un valore (`+, -, *, /, add, sub, mul, div`) qualsiasi.

Formato di output della funzione Lambda del calcolatore semplice

Quando un'operazione ha esito positivo, restituisce il risultato nel formato seguente:

```
{ "a": "Number", "b": "Number", "op": "string", "c": "Number"}
```

dove c contiene il risultato del calcolo.

Implementazione della funzione Lambda del calcolatore semplice

L'implementazione della funzione Lambda è la seguente:

```
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";
  }

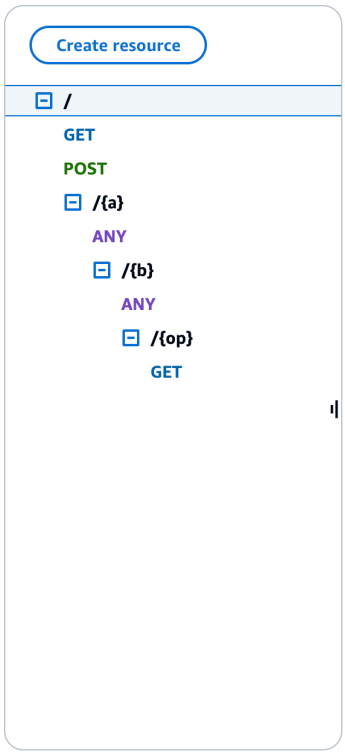
  const res = {};
  res.a = Number(event.a);
  res.b = Number(event.b);
```

```
  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 calcolatore semplice in API Gateway

L'API calcolatore semplice espone tre metodi (GET, POST, GET) per richiamare la the section called “Funzione Lambda del calcolatore semplice”. Di seguito è illustrata una rappresentazione grafica di quest'API:



Questi tre metodi mostrano modi diversi per fornire l'input per la funzione Lambda di back-end per eseguire la stessa operazione:

- Il metodo `GET /?a=...&b=...&op=...` usa i parametri di query per specificare l'input.
- Il metodo `POST /` usa un payload JSON `{"a":"Number", "b":"Number", "op":"string"}` per specificare l'input.
- Il metodo `GET /{a}/{b}/{op}` usa i parametri di percorso per specificare l'input.

Se non è definito, API Gateway genera il nome del metodo SDK corrispondente combinando le parti relative al metodo HTTP e al percorso. La parte del percorso root (`/`) è detta `Api Root`. Ad esempio, il nome del metodo SDK Java predefinito per il metodo API `GET /?a=...&b=...&op=...` è `getAB0p`, il nome del metodo SDK predefinito per `POST /` è `postApiRoot` e il nome del metodo SDK predefinito per `GET /{a}/{b}/{op}` è `getAB0p`. La convenzione può essere personalizzata per i singoli SDK. Consulta la documentazione dell'origine dell'SDK generato per informazioni sui nomi di metodi SDK specifici.

È possibile e consigliabile sostituire i nomi dei metodi SDK predefiniti specificando la proprietà operationName in ogni metodo API. Questa operazione viene eseguita durante la creazione del

metodo API o l'aggiornamento del metodo API con l'API REST di API Gateway. Nella definizione Swagger dell'API è possibile impostare `operationId` per ottenere lo stesso risultato.

Prima di illustrare come invocare questi metodi usando un SDK generato da API Gateway per questa API, esaminiamo brevemente come eseguire la configurazione. Per istruzioni dettagliate, consulta Sviluppo di REST API in Gateway API. Se non hai mai usato API Gateway, consulta prima Scegli un tutorial di integrazione AWS Lambda.

Creazione di modelli per l'input e l'output

Per specificare un input fortemente tipizzato nell'SDK, creiamo un modello `Input` per l'API. Per descrivere il tipo di dati del corpo della risposta, creiamo un modello `Output` e un modello `Result`.

Per creare i modelli per l'input, l'output e il risultato

1. Nel riquadro di navigazione principale seleziona Modelli.
2. Scegli Crea modello.
3. In Nome, inserisci **`input`**.
4. Per Tipo di contenuto inserisci **`application/json`**.

   Se non viene trovato alcun tipo di contenuto corrispondente, la convalida della richiesta non viene eseguita. Per utilizzare lo stesso modello indipendentemente dal tipo di contenuti, inserisci **`$default`**.
5. Per Schema modello immetti il seguente modello:

```
{
    "$schema" : "$schema": "http://json-schema.org/draft-04/schema#",
    "type":"object",
    "properties":{
        "a":{"type":"number"},
        "b":{"type":"number"},
        "op":{"type":"string"}
    },
    "title":"Input"
}
```

6. Scegli Crea modello.
7. Ripeti le seguenti fasi per creare un modello `Output` e un modello `Result`.

   Per il modello `Output` immetti in Schema modello quanto segue:

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "properties": {
        "c": {"type":"number"}
    },
    "title": "Output"
}
```

Per il modello `Result` immetti in Schema modello quanto segue. Sostituisci l'ID API `abc123` con il tuo ID API.

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type":"object",
    "properties":{
        "input":{
            "$ref":"https://apigateway.amazonaws.com/restapis/abc123/models/Input"
        },
        "output":{
            "$ref":"https://apigateway.amazonaws.com/restapis/abc123/models/Output"
        }
    },
    "title":"Result"
}
```

Configurazione dei parametri di query del metodo GET /

Per il metodo `GET /?a=..&b=..&op=..`, i parametri di query sono dichiarati in Method Request (Richiesta metodo):

Per configurare i parametri della stringa di query GET/URL

1. Nella sezione Richiesta metodo scegli Modifica per il metodo `GET` sulla risorsa root (`/`).
2. Scegli Parametri della stringa di query URL ed effettua le seguenti operazioni:
    a. Scegliere Add query string (Aggiungi stringa di query).
    b. In Nome, inserisci **a**.
    c. Mantieni Obbligatorio e Caching disattivati.

d. Mantieni disattivata l'opzione Caching.

Ripeti le stesse fasi e crea una stringa di query denominata **b** e una stringa di query denominata **op**.

3. Scegli Save (Salva).

Configurazione del modello di dati per il payload come input nel back-end

Per il metodo `POST /`, creiamo il modello `Input` e lo aggiungiamo alla richiesta del metodo per definire la forma dei dati di input.

Per configurare il modello di dati per il payload come input nel back-end

1. Nella sezione Richiesta metodo scegli Modifica per il metodo `POST` sulla risorsa root (`/`).
2. Scegli Corpo della richiesta.
3. Scegliere Add model (Aggiungi modello).
4. Per Tipo di contenuto inserisci **`application/json`**.
5. Per Modello seleziona Input.
6. Scegli Save (Salva).

Con questo modello, i clienti dell'API possono chiamare l'SDK per specificare l'input creando un'istanza di un oggetto `Input`. Senza questo modello, i clienti dovrebbero creare un oggetto dizionario per rappresentare l'input JSON per la funzione Lambda.

Configurazione del modello di dati per l'output del risultato dal back-end

Per tutti e tre i metodi, creiamo il modello `Result` e lo aggiungiamo in `Method Response` per il metodo per definire la forma dell'output restituito dalla funzione Lambda.

Per configurare il modello di dati per l'output del risultato dal back-end

1. Seleziona la risorsa /{a}/{b}/{op}, quindi scegli il metodo GET.
2. Nella scheda Risposta metodo scegli Modifica in Risposta 200.
3. In Corpo della risposta scegli Aggiungi modello.
4. Per Tipo di contenuto inserisci **`application/json`**.

5. Per Modello seleziona Risultato.
6. Scegli Save (Salva).

Con questo modello, i clienti dell'API possono analizzare un output con esito positivo leggendo le proprietà di un oggetto `Result`. Senza questo modello, i clienti dovrebbero creare un oggetto dizionario per rappresentare l'output JSON.

Definizione OpenAPI dell'API del calcolatore semplice

Di seguito è riportata una definizione OpenAPI dell'API del calcolatore semplice. Puoi importarla nel tuo account. Tuttavia, dopo l'importazione, dovrai reimpostare le autorizzazioni basate su risorse per la funzione Lambda. A questo scopo, riseleziona la funzione Lambda creata nel tuo account da Integration Request (Richiesta di integrazione) nella console API Gateway. La console API Gateway reimposterà le autorizzazioni richieste. In alternativa, puoi utilizzare AWS Command Line Interface per il comando Lambda add-permission.

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2016-09-29T20:27:30Z",
    "title": "SimpleCalc"
  },
  "host": "t6dve4zn25.execute-api.us-west-2.amazonaws.com",
  "basePath": "/demo",
  "schemes": [
    "https"
  ],
  "paths": {
    "/": {
      "get": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "op",
```

```
            "in": "query",
            "required": false,
            "type": "string"
          },
          {
            "name": "a",
            "in": "query",
            "required": false,
            "type": "string"
          },
          {
            "name": "b",
            "in": "query",
            "required": false,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Result"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "requestTemplates": {
            "application/json": "#set($inputRoot = $input.path('$'))\n{\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"
          }
        }
      }
    }
  }
```

```
}
```

Generazione dell'SDK Java di un'API in Gateway API

La procedura seguente mostra come generare l'SDK Java di un'API in Gateway API.

Per generare l'SDK Java di un'API di API Gateway

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere un'API REST.
3. Scegliere Stages (Fasi).
4. Nel riquadro Fasi, seleziona il nome della fase.
5. Apri il menu Azioni fase, quindi scegli Genera SDK.
6. In Piattaforma, scegli la piattaforma Java ed effettua le seguenti operazioni:

    a. In Service Name (Nome servizio) specificare il nome dell'SDK. Ad esempio, **`SimpleCalcSdk`**. Questo diventa il nome della classe del client SDK. Il nome corrisponde al tag `<name>` in `<project>` nel file pom.xml presente nella cartella di progetto dell'SDK. Non includere i trattini.

    b. In Java Package Name (Nome pacchetto Java) specificare il nome di un pacchetto per l'SDK. Ad esempio, **`examples.aws.apig.simpleCalc.sdk`**. Questo nome di pacchetto viene utilizzato come namespace della libreria dell'SDK. Non includere i trattini.

    c. In Java Build System (Sistema di compilazione Java) immettere **`maven`** o **`gradle`** per specificare il sistema di compilazione.

    d. In Java Group Id (ID gruppo Java), immettere l'identificatore di un gruppo per il progetto SDK. Ad esempio, immetti **`my-apig-api-examples`**. L'identificatore corrisponde al tag `<groupId>` di `<project>` nel file `pom.xml` presente nella cartella di progetto dell'SDK.

    e. In Java Artifact Id (ID artefatto Java), immettere l'identificatore di un artefatto per il progetto SDK. Ad esempio, immetti **`simple-calc-sdk`**. L'identificatore corrisponde al tag `<artifactId>` di `<project>` nel file `pom.xml` presente nella cartella di progetto dell'SDK.

    f. In Java Artifact Version (Versione artefatto Java), immettere la stringa identificatore della versione. Ad esempio, **`1.0.0`**. L'identificatore della versione corrisponde al tag `<version>` di `<project>` nel file `pom.xml` presente nella cartella di progetto dell'SDK.

    g. In Source Code License Text (Testo della licenza del codice sorgente), immettere il testo della licenza del codice sorgente, se disponibile.

7. Seleziona Generate SDK (Genera SDK) e segui le istruzioni sullo schermo per scaricare l'SDK generato da API Gateway.

Segui le istruzioni nell'articolo Utilizzo di un SDK Java generato da API Gateway per un'API REST per usare l'SDK generato.

Ogni volta che aggiorni un'API, devi ridistribuire l'API e rigenerare l'SDK per includere gli aggiornamenti.

Generazione dell'SDK Android di un'API in Gateway API

La procedura seguente mostra come generare l'SDK Android di un'API in Gateway API.

Per generare l'SDK Android di un'API in API Gateway

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere un'API REST.
3. Scegliere Stages (Fasi).
4. Nel riquadro Fasi, seleziona il nome della fase.
5. Apri il menu Azioni fase, quindi scegli Genera SDK.
6. In Piattaforma, scegli la piattaforma Android ed effettua le seguenti operazioni:
   a. In Group ID (ID gruppo), immettere l'identificatore univoco per il progetto corrispondente che viene usato nel file `pom.xml`, ad esempio **`com.mycompany`**.
   b. In Invoker package (Pacchetto di invoker), immettere lo spazio dei nomi per le classi client generate, ad esempio **`com.mycompany.clientsdk`**.
   c. In Artifact ID (ID artefatto), immettere il nome del file .jar compilato senza la versione che viene usato nel file `pom.xml`, ad esempio **`aws-apigateway-api-sdk`**.
   d. In Artifact version (Versione artefatto), immettere il numero di versione dell'artefatto per il cliente generato che Viene utilizzato nel `pom.xml` file e dovrebbe seguire un. *`major`* *`minor`*. *`patch`*modello (ad esempio,**`1.0.0`**).
7. Seleziona Generate SDK (Genera SDK) e segui le istruzioni sullo schermo per scaricare l'SDK generato da API Gateway.

Segui le istruzioni nell'articolo Utilizzo di un SDK Android generato da API Gateway per un'API REST per usare l'SDK generato.

Ogni volta che aggiorni un'API, devi ridistribuire l'API e rigenerare l'SDK per includere gli aggiornamenti.

Generazione dell'SDK iOS di un'API in Gateway API

La procedura seguente mostra come generare l'SDK iOS di un'API in Gateway API.

Per generare l'SDK iOS di un'API in API Gateway

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere un'API REST.
3. Scegliere Stages (Fasi).
4. Nel riquadro Fasi, seleziona il nome della fase.
5. Apri il menu Azioni fase, quindi scegli Genera SDK.
6. In Piattaforma, scegli la piattaforma iOS (Objective-C) o iOS (Swift) ed effettua le seguenti operazioni:

   - Digitare un prefisso univoco nella casella Prefix (Prefisso).

     L'effetto del prefisso è il seguente: se si assegna, ad esempio, **`SIMPLE_CALC`** come prefisso per l'SDK all'SimpleCalcAPI con, e `result` modelli`input`, `output` l'SDK generato conterrà la `SIMPLE_CALC``SimpleCalcClient` classe che incapsula l'API, incluso il metodo richieste/risposte. Inoltre, l'SDK generato conterrà le classi `SIMPLE_CALC``input`, `SIMPLE_CALC``output` e `SIMPLE_CALC``result` per rappresentare l'input, l'output e i risultati, rispettivamente, per l'input di richiesta e l'output di risposta. Per ulteriori informazioni, consulta Usa l'SDK iOS generato da API Gateway per un'API REST in Objective-C o Swift.
7. Seleziona Generate SDK (Genera SDK) e segui le istruzioni sullo schermo per scaricare l'SDK generato da API Gateway.

Segui le istruzioni nell'articolo Usa l'SDK iOS generato da API Gateway per un'API REST in Objective-C o Swift per usare l'SDK generato.

Ogni volta che aggiorni un'API, devi ridistribuire l'API e rigenerare l'SDK per includere gli aggiornamenti.

Genera l' JavaScript SDK di un'API REST in API Gateway

La procedura seguente mostra come generare l' JaveScript SDK di un'API in API Gateway.

Per generare l' JavaScript SDK di un'API in API Gateway

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere un'API REST.
3. Scegliere Stages (Fasi).
4. Nel riquadro Fasi, seleziona il nome della fase.
5. Apri il menu Azioni fase, quindi scegli Genera SDK.
6. Per Piattaforma, scegli la piattaforma. JavaScript
7. Seleziona Generate SDK (Genera SDK) e segui le istruzioni sullo schermo per scaricare l'SDK generato da API Gateway.

Segui le istruzioni nell'articolo Usa un JavaScript SDK generato da API Gateway per un'API REST per usare l'SDK generato.

Ogni volta che aggiorni un'API, devi ridistribuire l'API e rigenerare l'SDK per includere gli aggiornamenti.

Generazione dell'SDK Ruby di un'API in Gateway API

La procedura seguente mostra come generare l'SDK Ruby di un'API in Gateway API.

Per generare l'SDK Ruby di un'API in API Gateway

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere un'API REST.
3. Scegliere Stages (Fasi).
4. Nel riquadro Fasi, seleziona il nome della fase.
5. Apri il menu Azioni fase, quindi scegli Genera SDK.
6. In Piattaforma, scegli la piattaforma Ruby ed effettua le seguenti operazioni:

    a. In Service Name (Nome servizio) specificare il nome dell'SDK. Ad esempio, **`SimpleCalc`**. Questo nome viene utilizzato come namespace Ruby Gem dell'API. Deve essere composto solo da lettere, (`a-zA-Z`), senza caratteri speciali o numeri.

b. In Ruby Gem Name (Nome Ruby Gem) specificare il nome Ruby Gem per contenere il codice sorgente dell'SDK generato per l'API. Per impostazione predefinita, è il nome del servizio in minuscolo più il suffisso `-sdk`, ad esempio **`simplecalc-sdk`**.

c. In Ruby Gem Version (Versione Ruby Gem) specificare il numero di versione del Ruby Gem generato. Per impostazione predefinita, è impostato su `1.0.0`.

7. Seleziona Generate SDK (Genera SDK) e segui le istruzioni sullo schermo per scaricare l'SDK generato da API Gateway.

Segui le istruzioni nell'articolo Utilizzo di un SDK Ruby generato da API Gateway per un'API REST per usare l'SDK generato.

Ogni volta che aggiorni un'API, devi ridistribuire l'API e rigenerare l'SDK per includere gli aggiornamenti.

Generazione SDKs per un'API utilizzando AWS CLI i comandi in API Gateway

Puoi usarlo AWS CLI per generare e scaricare un SDK di un'API per una piattaforma supportata chiamando il comando get-sdk. Di seguito è riportata la dimostrazione per alcune delle piattaforme supportate.

Argomenti

- Genera e scarica l'SDK Java per Android utilizzando il AWS CLI
- Genera e scarica l' JavaScript SDK utilizzando il AWS CLI
- Genera e scarica l'SDK di Ruby usando AWS CLI

Genera e scarica l'SDK Java per Android utilizzando il AWS CLI

Per generare e scaricare un SDK Java per Android generato da API Gateway di un'API (`udpuvvzbkc`) in una specifica fase (`test`), invoca il comando seguente:

```
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
```

L'ultimo input di `~/apps/myApi/myApi-android-sdk.zip` è il percorso del file SDK scaricato denominato `myApi-android-sdk.zip`.

Genera e scarica l' JavaScript SDK utilizzando il AWS CLI

Per generare e scaricare un JavaScript SDK generato da API Gateway di un'API (`udpuvvzbkc`) in una determinata fase (`test`), chiamate il comando come segue:

```
aws apigateway get-sdk \
            --rest-api-id udpuvvzbkc \
            --stage-name test \
            --sdk-type javascript \
            ~/apps/myApi/myApi-js-sdk.zip
```

L'ultimo input di `~/apps/myApi/myApi-js-sdk.zip` è il percorso del file SDK scaricato denominato `myApi-js-sdk.zip`.

Genera e scarica l'SDK di Ruby usando AWS CLI

Per generare e scaricare un SDK Ruby di un'API (`udpuvvzbkc`) in una specifica fase (`test`), chiama il comando come segue:

```
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
```

L'ultimo input di `~/apps/myApi/myApi-ruby-sdk.zip` è il percorso del file SDK scaricato denominato `myApi-ruby-sdk.zip`.

Successivamente, mostreremo come utilizzare l'SDK generato per chiamare l'API sottostante. Per ulteriori informazioni, consulta Invocazione di REST API in Gateway API.

Vendi il tuo API Gateway APIs tramite Marketplace AWS

Dopo aver creato, testato e distribuito il tuo APIs, puoi inserirlo in un piano di utilizzo API Gateway e venderlo come prodotto Software as a Service (SaaS). Marketplace AWS Gli acquirenti di API che si

abbonano alla tua offerta di prodotti vengono fatturati Marketplace AWS in base al numero di richieste effettuate al piano di utilizzo.

Per vendere APIs il tuo prodotto Marketplace AWS, devi configurare il canale di vendita per l'integrazione Marketplace AWS con API Gateway. In generale, ciò comporta l'offerta del prodotto Marketplace AWS, l'impostazione di un ruolo IAM con politiche appropriate per consentire ad API Gateway di inviare metriche di utilizzo a Marketplace AWS, l'associazione di un Marketplace AWS prodotto a un piano di utilizzo API Gateway e l'associazione di un Marketplace AWS acquirente a una chiave API Gateway. I dettagli vengono approfonditi nelle sezioni seguenti.

Per ulteriori informazioni sulla vendita della tua API come prodotto SaaS Marketplace AWS, consulta la Guida per l'Marketplace AWS utente.

Argomenti

- Inizializza l' Marketplace AWS integrazione con API Gateway
- Gestione della sottoscrizione del cliente ai piani di utilizzo

Inizializza l' Marketplace AWS integrazione con API Gateway

Le seguenti attività riguardano l'inizializzazione unica dell' Marketplace AWS integrazione con API Gateway, che consente di vendere il prodotto APIs come SaaS.

Pubblica un prodotto su Marketplace AWS

Per inserire il tuo piano di utilizzo come prodotto SaaS, invia un modulo di caricamento prodotto tramite Marketplace AWS. Il prodotto deve contenere una dimensione denominata `apigateway` del tipo `requests`. Questa dimensione definisce price-per-request e viene utilizzata da API Gateway per misurare le richieste al tuo APIs.

Creazione del ruolo di misurazione

Crea un ruolo IAM denominato `ApiGatewayMarketplaceMeteringRole` con le seguenti policy di esecuzione e policy di attendibilità. Questo ruolo consente ad API Gateway di inviare metriche di utilizzo Marketplace AWS a tuo nome.

Policy di esecuzione del ruolo di misurazione

JSON

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "aws-marketplace:BatchMeterUsage",
        "aws-marketplace:ResolveCustomer"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

Policy di relazione di fiducia del ruolo di misurazione

JSON

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

Associa il piano di utilizzo al prodotto Marketplace AWS

Quando offri un prodotto su Marketplace AWS, ricevi un codice Marketplace AWS prodotto. Per integrare API Gateway con Marketplace AWS, associa il tuo piano di utilizzo al codice Marketplace AWS del prodotto. Puoi abilitare l'associazione impostando il `productCode`campo `UsagePlan` dell'API Gateway sul codice del Marketplace AWS prodotto, utilizzando la console API Gateway, l'API REST API Gateway, l'API AWS CLI for API Gateway o l' AWS SDK per API Gateway. Nell'esempio di codice seguente viene utilizzata l'API REST dell'API Gateway:

```
PATCH /usageplans/USAGE_PLAN_ID
Host: apigateway.region.amazonaws.com
Authorization: ...

{
    "patchOperations" : [{
        "path" : "/productCode",
        "value" : "MARKETPLACE_PRODUCT_CODE",
        "op" : "replace"
    }]
}
```

Gestione della sottoscrizione del cliente ai piani di utilizzo

Le seguenti attività sono gestite dall'applicazione del portale sviluppatore.

Quando un cliente si abbona al tuo prodotto tramite Marketplace AWS, Marketplace AWS inoltra una `POST` richiesta all'URL degli abbonamenti SaaS che hai registrato al momento dell'offerta del prodotto. Marketplace AWS La richiesta `POST` presenta un parametro `x-amzn-marketplace-token` contenente le informazioni sull'acquirente. Segui le istruzioni riportate in Onboarding dei clienti SaaS per gestire questo reindirizzamento nell'applicazione del portale per gli sviluppatori.

In risposta alla richiesta di iscrizione di un cliente, Marketplace AWS invia una `subscribe-success` notifica a un argomento di Amazon SNS a cui puoi abbonarti. (Consulta Onboarding dei clienti SaaS). Per accettare la richiesta di abbonamento del cliente, gestisci la `subscribe-success` notifica creando o recuperando una chiave API Gateway per il cliente, associando la chiave API Marketplace AWS-provisioned `customerId` del cliente alle chiavi API e quindi aggiungendo la chiave API al tuo piano di utilizzo.

Una volta completata la richiesta di abbonamento del cliente, l'applicazione del portale per sviluppatori deve presentare al cliente la chiave API associata e informare il cliente che la chiave API deve essere inclusa nell'intestazione delle richieste inviate a. `x-api-key` APIs

Quando un cliente annulla un abbonamento a un piano di utilizzo, Marketplace AWS invia una `unsubscribe-success` notifica all'argomento SNS. Per portare a termine il processo di cancellazione di un cliente, devi gestire la notifica `unsubscribe-success` eliminando le chiavi API del cliente dal piano di utilizzo.

Per autorizzare un cliente ad accedere a un piano di utilizzo

Per autorizzare un cliente ad accedere al tuo piano di utilizzo, usa l'API di API Gateway per recuperare o creare una chiave API per il cliente e aggiungerla al piano di utilizzo.

L'esempio seguente mostra come chiamare l'API REST di API Gateway per creare una nuova chiave API con un Marketplace AWS `customerId` valore specifico (*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"
}
```

L'esempio seguente mostra come ottenere una chiave API con un Marketplace AWS `customerId` valore specifico (*MARKETPLACE_CUSTOMER_ID*).

```
GET apikeys?customerId=MARKETPLACE_CUSTOMER_ID HTTP/1.1
Host: apigateway.region.amazonaws.com
Authorization: ...
```

Per aggiungere una chiave API a un piano di utilizzo, crea un `UsagePlanKey` con la chiave API per il relativo piano di utilizzo. Il seguente esempio mostra come ottenere tale risultato utilizzando l'API REST di API Gateway, dove `n371pt` è l'ID del piano di utilizzo e `q5ugs7qjjh` è un esempio di `keyId` API restituito dagli esempi precedenti.

```
POST /usageplans/n371pt/keys HTTP/1.1
Host: apigateway.region.amazonaws.com
Authorization: ...

{
    "keyId": "q5ugs7qjjh",
```

```
    "keyType": "API_KEY"
}
```

Per associare un cliente a una chiave API

È necessario aggiornare il `customerId` campo `ApiKey`'s con l'ID Marketplace AWS cliente del cliente. In questo modo, la chiave API viene associata al cliente Marketplace AWS , attivando così misurazione e fatturazione per l'acquirente. Il seguente esempio di codice invoca l'API REST API Gateway per ottenere tale risultato.

```
PATCH /apikeys/q5ugs7qjjh
Host: apigateway.region.amazonaws.com
Authorization: ...

{
    "patchOperations" : [{
        "path" : "/customerId",
        "value" : "MARKETPLACE_CUSTOMER_ID",
        "op" : "replace"
    }]
}
```

Protezione delle REST API in Gateway API

API Gateway fornisce diversi modi per proteggere l'API da determinate minacce, ad esempio utenti malintenzionati o picchi di traffico. Puoi proteggere l'API utilizzando strategie come la generazione di certificati SSL, la configurazione di un firewall per applicazioni Web, l'impostazione di destinazioni di limitazione e l'accesso all'API solo da un cloud privato virtuale (VPC, Virtual Private Cloud). In questa sezione viene descritto come abilitare queste funzionalità utilizzando API Gateway.

Argomenti

- Richiedi certificati client per la tua API con autenticazione TLS reciproca in API Gateway
- Presenta i certificati client ai servizi di backend con TLS reciproco in API Gateway
- Utilizzalo AWS WAF per proteggere il tuo REST APIs in API Gateway
- Limita le richieste al tuo REST APIs per una migliore velocità di trasmissione in API Gateway
- REST API private in Gateway API

Richiedi certificati client per la tua API con autenticazione TLS reciproca in API Gateway

L'autenticazione TLS reciproca richiede l'autenticazione bidirezionale tra il client e il server. Con il protocollo TLS reciproco, i clienti devono presentare X.509 certificati per verificare la propria identità per accedere alla tua API. L'autenticazione TLS reciproca è un requisito comune per Internet of Things (IoT) e le applicazioni business-to-business.

È possibile utilizzare l'autenticazione TLS reciproca insieme ad altre operazioni di autorizzazione e autenticazione supportate da API Gateway. API Gateway inoltra i certificati forniti dai client alle autorizzazioni Lambda e alle integrazioni di back-end.

> **Important**
>
> Per impostazione predefinita, i client possono richiamare l'API utilizzando l'endpoint `execute-api` generato da API Gateway per l'API. Per garantire che i client possano accedere all'API solo utilizzando un nome di dominio personalizzato con l'autenticazione TLS reciproca, disattivare l'endpoint `execute-api` predefinito. Per ulteriori informazioni, consulta the section called “Disabilitazione dell'endpoint predefinito”.

Argomenti

- Prerequisiti per l'autenticazione TLS reciproca
- Configurazione dell'autenticazione TLS reciproca per un nome di dominio personalizzato
- Richiamo di un'API utilizzando un nome di dominio personalizzato che richiede l'autenticazione TLS reciproca.
- Aggiornamento del truststore
- Disabilitazione dell'autenticazione TLS reciproca
- Risoluzione dei problemi relativi all'autenticazione TLS reciproca per la REST API

Prerequisiti per l'autenticazione TLS reciproca

Per configurare l'autenticazione TLS reciproca hai bisogno di:

- Un nome di dominio personalizzato regionale

- Almeno un certificato configurato AWS Certificate Manager per il tuo nome di dominio personalizzato
- Un truststore configurato e caricato in Amazon S3

Nomi di dominio personalizzati

Per abilitare l'autenticazione TLS reciproca per un'API REST, è necessario configurare un nome di dominio personalizzato per l'API. È possibile abilitare l'autenticazione TLS reciproca per un nome di dominio personalizzato e quindi fornire il nome di dominio personalizzato ai client. Per accedere a un'API utilizzando un nome di dominio personalizzato con l'autenticazione TLS reciproca attivata, i client devono presentare certificati attendibili nelle richieste API. Puoi trovare ulteriori informazioni all'indirizzo the section called “Nomi di dominio personalizzati”.

Utilizzo AWS Certificate Manager certificati emessi

Puoi richiedere un certificato pubblicamente attendibile direttamente da ACM o importare certificati pubblici o autofirmati. Per configurare un certificato in ACM, accedi ad ACM. Se desideri importare un certificato, continua a leggere la sezione seguente.

Utilizzando un file importato o AWS Autorità di certificazione privata certificato

Per utilizzare un certificato importato in ACM o un certificato AWS Autorità di certificazione privata con TLS reciproco, API Gateway necessita di un certificato `ownershipVerificationCertificate` rilasciato da ACM. Questo certificato di proprietà viene utilizzato solo per verificare che disponi delle autorizzazioni per l'utilizzo del nome di dominio. Non viene utilizzato per l'handshake TLS. Se non ne hai già uno`ownershipVerificationCertificate`, vai su per https://console.aws.amazon.com/acm/ configurarne uno.

Dovrai mantenere la validità di questo certificato per tutta la durata del nome di dominio. Se un certificato scade e il rinnovo automatico non riesce, tutti gli aggiornamenti al nome di dominio verranno bloccati. Dovrai aggiornare il `ownershipVerificationCertificateArn` con un `ownershipVerificationCertificate` valido prima di poter apportare altre modifiche. Il `ownershipVerificationCertificate` non può essere utilizzato come certificato del server per un altro dominio TLS reciproco in API Gateway. Se un certificato viene reimportato direttamente in ACM, l'emittente deve rimanere invariato.

Configurazione del truststore

I truststore sono file di testo con un'estensione `.pem`. Sono elenchi attendibili di certificati provenienti da autorità di certificazione. Per utilizzare il protocollo TLS reciproco, crea un truststore di X.509 certificati affidabili per accedere alla tua API.

Devi includere nel truststore l'intera catena di attendibilità, a partire dal certificato della CA emittente, fino al certificato emesso da una CA radice. API Gateway accetta certificati del client emessi da qualsiasi CA presente nella catena di attendibilità. I certificati possono provenire da autorità di certificazione pubbliche o private. I certificati possono avere una lunghezza massima della catena di quattro. È inoltre possibile fornire certificati autofirmati. Nel truststore sono supportati i seguenti algoritmi:

- SHA-256 o più forte
- RSA-2048 o più forte
- ECDSA-256 oppure ECDSA-384

API Gateway convalida un certo numero di proprietà del certificato. È possibile utilizzare le autorizzazioni Lambda per eseguire ulteriori controlli quando un client richiama un'API, incluso il controllo della revoca di un certificato. API Gateway convalida le seguenti proprietà:

Validation	Description
X.509 sintassi	Il certificato deve soddisfare i requisiti di X.509 sintassi.
Integrità	Il contenuto del certificato non deve essere stato modificato rispetto a quello firmato dall'autorità di certificazione del truststore.
Validity	Il periodo di validità del certificato deve essere attuale.
Concatenamento di nomi/concatenamento di chiavi	I nomi e gli oggetti dei certificati devono formare una catena ininterrotta. I certificati possono avere una lunghezza massima della catena di quattro.

Caricare il truststore in un bucket Amazon S3 in un singolo file

Di seguito è riportato un esempio di come potrebbe presentarsi un file .pem.

Example certificates.pem

```
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
...
```

Il seguente AWS CLI comando cp viene caricato nel tuo `certificates.pem` bucket Amazon S3:

```
aws s3 cp certificates.pem s3://bucket-name
```

Configurazione dell'autenticazione TLS reciproca per un nome di dominio personalizzato

Per configurare l'autenticazione TLS reciproca per una REST API, è necessario utilizzare un nome di dominio personalizzato regionale per l'API con una policy di sicurezza `TLS_1_2`. Per ulteriori informazioni sulla scelta di una policy di sicurezza, consulta the section called “Scelta di una policy di sicurezza”.

> **Note**
>
> L'autenticazione TLS reciproca non è supportata per le API private.

Dopo aver caricato il truststore su Amazon S3, puoi configurare il tuo nome di dominio personalizzato per utilizzare l'autenticazione TLS reciproca. Il comando create-domain-name seguente crea un nome di dominio personalizzato con l’autenticazione TLS reciproca:

```
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
```

Dopo aver creato il nome di dominio, devi configurare i record DNS e le mappature del percorso di base per le operazioni API. Per ulteriori informazioni, consulta Configurazione di un nome di dominio personalizzato regionale in Gateway API.

Richiamo di un'API utilizzando un nome di dominio personalizzato che richiede l'autenticazione TLS reciproca.

Per richiamare un'API con l'autenticazione TLS reciproca abilitata, i client devono presentare un certificato attendibile nella richiesta API. Quando un client tenta di richiamare la tua API, API Gateway cerca l'emittente del certificato del client nel tuo truststore. Per permettere ad API Gateway di procedere con la richiesta, l'emittente del certificato e l'intera catena di attendibilità fino al certificato emesso da una CA root devono trovarsi nel truststore.

Il seguente comando `curl` di esempio seguente invia una richiesta `api.example.com`, che include `my-cert.pem` nella richiesta. `my-key.key` è la chiave privata del certificato.

```
curl -v --key ./my-key.key --cert ./my-cert.pem api.example.com
```

L'API viene richiamata solo se il truststore considera attendibile il certificato. Le seguenti condizioni impediranno ad API Gateway di eseguire l'handshake TLS e lo porteranno a negare la richiesta con il codice di stato `403`. Se il tuo certificato:

- non è attendibile
- è scaduto
- non utilizza un algoritmo supportato

> **Note**
>
> API Gateway non verifica se un certificato è stato revocato.

Aggiornamento del truststore

Per aggiornare i certificati nel truststore, carica un nuovo bundle di certificati in Amazon S3. Potrai quindi aggiornare il nome di dominio personalizzato per utilizzare il certificato aggiornato.

Usa il Controllo delle versioni di Amazon S3 per mantenere più versioni del truststore. Quando si aggiorna il nome di dominio personalizzato per utilizzare una nuova versione del truststore, API Gateway restituisce avvisi se i certificati non sono validi.

API Gateway produce avvisi di certificati solo quando si aggiorna il nome di dominio. API Gateway non invia notifiche se un certificato caricato in precedenza scade.

Il comando update-domain-name seguente aggiorna un nome di dominio personalizzato per utilizzare una nuova versione del truststore:

```
aws apigateway update-domain-name \
    --domain-name api.example.com \
    --patch-operations op='replace',path='/mutualTlsAuthentication/
truststoreVersion',value='abcdef123'
```

Disabilitazione dell'autenticazione TLS reciproca

Il comando update-domain-name seguente disabilita l'autenticazione TLS reciproca:

```
aws apigateway update-domain-name \
    --domain-name api.example.com \
    --patch-operations op='replace',path='/mutualTlsAuthentication/
truststoreUri',value=''
```

Risoluzione dei problemi relativi all'autenticazione TLS reciproca per la REST API

Di seguito vengono forniti i consigli per la risoluzione dei problemi e la correzione degli errori che possono verificarsi durante l'attivazione dell'autenticazione TLS reciproca.

Risoluzione dei problemi relativi agli avvisi dei certificati

Quando crei un nome di dominio personalizzato con l'autenticazione TLS reciproca, API Gateway genera avvisi se i certificati nel truststore non sono validi. Questo può verificarsi anche quando aggiorni un nome di dominio personalizzato all'utilizzo di un nuovo truststore. Gli avvisi indicano il problema con il certificato e l'oggetto del certificato che ha generato l'avviso. L'autenticazione TLS

reciproca è ancora abilitata per l'API, ma alcuni client potrebbero non essere in grado di accedere all'API.

Per identificare il certificato che ha generato l'avviso, devi decodificare i certificati nel truststore. Puoi utilizzare strumenti come `openssl` per decodificare i certificati e identificarne gli oggetti.

Il comando seguente visualizza il contenuto di un certificato, incluso l'oggetto:

```
openssl x509 -in certificate.crt -text -noout
```

Aggiorna o rimuovi i certificati che hanno prodotto gli avvisi, quindi carica un nuovo truststore su Amazon S3. Dopo aver caricato il nuovo truststore, aggiorna il nome di dominio personalizzato per utilizzarlo.

Risoluzione dei problemi relativi ai conflitti dei nomi di dominio

L'errore `"The certificate subject <certSubject> conflicts with an existing certificate from a different issuer."` indica che più autorità di certificazione hanno emesso un certificato per questo dominio. Per ogni oggetto del certificato, può esistere un solo emittente in API Gateway per domini TLS reciproci. Dovrai ottenere tutti i certificati per tale oggetto tramite un unico emittente. Se il problema riguarda un certificato di cui non hai il controllo ma puoi provare la proprietà del nome di dominio, contatta Supporto per aprire un ticket.

Risoluzione dei problemi relativi ai messaggi di stato dei nomi di dominio

`PENDING_CERTIFICATE_REIMPORT`: questo significa che hai reimportato un certificato in ACM e la convalida non è riuscita perché il nuovo certificato ha un SAN (subject alternative name, nome alternativo dell'oggetto) che non è coperto dal `ownershipVerificationCertificate` o l'oggetto o i SAN nel certificato non coprono il nome di dominio. Potrebbe esserci una configurazione errata o è stato importato un certificato non valido. Devi reimportare un certificato valido in ACM. Per ulteriori informazioni sulla convalida, consulta Convalida della proprietà del dominio.

`PENDING_OWNERSHIP_VERIFICATION`: significa che il certificato verificato in precedenza è scaduto e che ACM non è stato in grado di rinnovarlo in automatico. Dovrai rinnovare il certificato o richiederne uno nuovo. Ulteriori informazioni sul rinnovo del certificato si trovano nella guida alla Risoluzione dei problemi relativi al rinnovo dei certificati gestiti di ACM.

Risoluzione dei problemi relativi alla restituzione di un certificato errato

Durante la migrazione di un certificato dedicato da un nome di dominio completo (FQDN) a un nome di dominio con caratteri jolly del cliente, Gateway API potrebbe restituire il certificato per il nome FQDN anziché per il nome di dominio con caratteri jolly.

Il comando seguente mostra quale certificato viene restituito da Gateway API:

```
openssl s_client -connect hostname:port
```

Se il certificato risultante è per il nome FQDN, contatta Supporto per aprire un ticket.

Presenta i certificati client ai servizi di backend con TLS reciproco in API Gateway

API Gateway può presentare un certificato client al tuo backend durante l'handshake TLS. Il tuo backend può quindi verificare che le richieste provengano da API Gateway. Questo crea una connessione TLS reciproca tra API Gateway e il servizio di backend.

Puoi scegliere tra due approcci:

- AWS Certificate Manager Certificati (ACM): importa il tuo CA-signed certificato da ACM e allegalo alla tua fase API. Con questo approccio, ottieni catene complete di certificati, rinnovo automatico e integrazione con la tua infrastruttura PKI esistente.
- Gateway-generated Certificati API: genera un certificato autofirmato in API Gateway. Questo approccio non richiede un'infrastruttura di certificati esterna. Il certificato scade dopo 365 giorni e deve essere ruotato manualmente.

Argomenti

- Come funziona l'autenticazione dei certificati del client backend
- Usa il tuo certificato ACM per il TLS reciproco di backend in API Gateway
- Usa un Gateway-generated certificato API per l'autenticazione backend in API Gateway
- Autorità di certificazione supportate da Gateway API per le integrazioni HTTP e proxy HTTP in Gateway API

Come funziona l'autenticazione dei certificati del client backend

Con Outbound Mutual TLS (MTLS), API Gateway funge da client TLS. Presenta un certificato client al server di backend quando si connette. Il tuo backend controlla questo certificato per confermare che le richieste provengono da API Gateway.

Gli MTL in uscita sono diversi dagli MTL in entrata. Gli MTL in entrata richiedono che i client API presentino i certificati ad API Gateway. Gli MTL in uscita proteggono il collegamento tra API Gateway e il tuo backend.

Sequenza di handshake TLS

I passaggi seguenti mostrano come API Gateway imposta una connessione TLS reciproca con il tuo backend:

1. API Gateway invia un ClientHello messaggio al backend.
2. Il backend invia il certificato del server ad API Gateway.
3. API Gateway controlla il certificato del server.
4. Il backend invia un messaggio CertificateRequest a API Gateway.
5. API Gateway invia la catena di certificati del client al backend.
6. Il backend verifica il certificato del client rispetto al suo archivio di fiducia.
7. La connessione TLS reciproca è pronta. API Gateway inoltra la richiesta API.

Se il backend non richiede un certificato client nel passaggio 4, API Gateway termina l'handshake TLS senza inviarne uno. La connessione utilizza il protocollo TLS unidirezionale standard.

Confronta le opzioni di certificato

La tabella seguente confronta i due approcci per la configurazione dei certificati client di backend.

ACM-managed rispetto ai certificati API Gateway-generated

Funzionalità	ACM-managed certificati	Gateway-generated certificati API
Fonte del certificato	La tua PKI o AWS Autorità di certificazione privata	API Gateway autofirmato

Funzionalità	ACM-managed certificati	Gateway-generated certificati API
Configurazione	Fai riferimento al certificato sul palco tramite il relativo ARN ACM; gestisci il ciclo di vita del certificato con le API ACM.	Gestisci nella pagina dei certificati client o con le API dei certificati client API Gateway.
Rinnovo	Dipende da come è stato rilasciato il certificato. Per i certificati emessi da ACM AWS Autorità di certificazione privata, ACM li rinnova automaticamente e API Gateway propaga l'aggiornamento senza ridistribuzione e senza tempi di inattività. Per i certificati importati in ACM, è necessario reimportarli prima della scadenza; API Gateway quindi propaga la reimportazione automaticamente.	Manuale. Il certificato scade dopo 365 giorni e deve essere ruotato manualmente.
Supporto a catena	Catena completa (fino a 5 certificati)	Certificato unico autofirmato
Modello di fiducia backend	Il backend si fida della tua CA	Il backend deve bloccare il certificato API Gateway

Usa il tuo certificato ACM per il TLS reciproco di backend in API Gateway

Puoi configurare API Gateway per presentare il tuo CA-signed certificato ai servizi di backend. Importa il tuo certificato in AWS Certificate Manager (ACM) o emettine uno tramite. AWS Autorità di certificazione privata Quindi, collega il certificato ACM ARN alla tua fase API.

Prerequisiti

Prima di configurare un certificato client ACM, devi disporre di quanto segue:

- AWS Certificate Manager accesso nella stessa regione della tua API.

- Autorizzazioni IAM: `acm:ImportCertificate` e `acm:DescribeCertificate` (per l'importazione dell'opzione A), o `acm:RequestCertificate` (per l'opzione B), o `acm-pca:IssueCertificateacm-pca:GetCertificate`, e `acm:ImportCertificate` (per l'opzione C).
- Un'API REST implementata in una fase.

Fase 1: Importa un certificato o emettine uno tramite AWS Autorità di certificazione privata

Puoi importare un certificato dalla tua PKI esistente o emettere un nuovo certificato tramite AWS Autorità di certificazione privata. Entrambi i percorsi producono un certificato ACM ARN da utilizzare nel passaggio successivo.

> **Note**
>
> I certificati pubblici ACM non sono supportati per l'autenticazione del client backend. A partire dall'11 giugno 2025, AWS Certificate Manager non emette più certificati pubblici con l'uso `clientAuth` esteso delle chiavi (EKU). Questa funzionalità richiede l'EKU, quindi utilizza un certificato che importi in ACM o uno emesso tramite. AWS Autorità di certificazione privata

> **Note**
>
> Crea il certificato ACM nella stessa AWS regione dell'API REST che lo utilizzerà. I certificati ACM sono risorse regionali, quindi il certificato deve esistere nella regione dell'API.

Opzione A: importazione dalla tua PKI esistente

Per importare un certificato client e la relativa chiave privata in ACM, esegui il comando seguente. Per ulteriori informazioni, consulta Importazione di certificati nella Guida per l'utente di AWS Certificate Manager .

```
aws acm import-certificate \
  --certificate fileb://client-cert.pem \
  --private-key fileb://private-key.pem \
  --certificate-chain fileb://ca-chain.pem \
  --region region
```

Il comando restituisce il certificato ACM ARN. Registrate questo valore per il passaggio successivo.

Opzione B: richiedi un certificato tramite AWS Autorità di certificazione privata (ACM-managed)

Per richiedere un certificato privato che ACM gestisce e può rinnovare automaticamente, esegui il seguente comando. Per ulteriori informazioni, consulta Richiesta di un certificato privato nella Guida per l'utente. AWS Certificate Manager

```
aws acm request-certificate \
  --domain-name www.example.com \
  --certificate-authority-arn arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/12345678-1234-1234-1234-123456789012 \
  --region region
```

Il comando restituisce il certificato ACM ARN. Registrate questo valore per il passaggio successivo.

Opzione C: emissione tramite AWS Autorità di certificazione privata e importazione in ACM

Se hai bisogno di un controllo diretto sui parametri del certificato (come estensioni personalizzate o algoritmi di firma), puoi emettere un certificato tramite ACM AWS Autorità di certificazione privata e poi importarlo in ACM. I certificati importati in questo modo non vengono rinnovati automaticamente da ACM. Assicurati che il certificato soddisfi i requisiti del certificato. Per ulteriori informazioni sull'emissione di certificati privati, vedere Emissione di un certificato di entità finale privata nella Guida per l'utente. AWS Autorità di certificazione privata

```
aws acm-pca issue-certificate \
  --certificate-authority-arn arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/12345678-1234-1234-1234-123456789012 \
  --csr fileb://csr.pem \
  --signing-algorithm SHA256WITHRSA \
  --validity Value=365,Type=DAYS
```

Recupera e importa il certificato

Il `issue-certificate` comando restituisce un AWS Autorità di certificazione privata certificato ARN, non un ARN ACM. Per utilizzare questo certificato con API Gateway, recuperalo utilizzando `aws acm-pca get-certificate` e quindi importalo in ACM using. `aws acm import-certificate` L'importazione produce il certificato ACM ARN che utilizzerai nel passaggio successivo. Quando eseguiaws `acm import-certificate`, imposta `--region` la regione dell'API in modo che il certificato ACM venga creato lì.

Passaggio 2: configura una fase API per utilizzare il certificato ACM

Dopo aver ottenuto un certificato ACM ARN, configura la fase API per presentare il certificato al tuo backend.

Per configurare una fase (console)

1. Apri la console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli la tua API REST.
3. Scegliere Stages (Fasi).
4. Nella sezione Dettagli fase scegli Modifica.
5. Per il certificato client, seleziona il tuo certificato ACM dall'elenco a discesa.
6. Scegli Save changes (Salva modifiche).

Per configurare una fase (AWS CLI)

Esegui il comando seguente:

```
aws apigateway update-stage \
  --rest-api-id abc123 \
  --stage-name prod \
  --patch-operations op='replace',path=/clientCertificateId,value=arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
```

> **Note**
>
> API Gateway utilizza lo stesso `clientCertificateId` campo per i Gateway-generated certificati ACM e API. Quando fornisci un certificato ACM ARN, API Gateway rileva automaticamente il formato e utilizza il flusso di lavoro. ACM-managed

Fase 3: Verifica la configurazione

Per verificare che API Gateway invii il certificato al tuo backend, completa i seguenti passaggi:

⚠ Il backend deve richiedere un certificato client

Il backend deve essere configurato per richiedere un certificato client durante l'handshake TLS. Se il backend non ne richiede uno, API Gateway non presenta il certificato.

1. Richiama il tuo endpoint API.
2. Verifica che il tuo backend riceva il certificato client durante l'handshake TLS.
3. Verifica che il tuo backend accetti il certificato e restituisca una risposta positiva.

Se il backend rifiuta il certificato, verifica che la catena di certificati possa essere convalidata rispetto al trust store del backend.

Requisiti del certificato

Il certificato leaf che configuri deve soddisfare i seguenti requisiti.

Requisiti del certificato client ACM

Requisito	Description
Lunghezza massima della catena	5 certificati
Validity	Il certificato non deve essere scaduto o essere ancora valido al momento della configurazione
Region	Il certificato ACM deve trovarsi nella stessa regione dell'API
Account	Il certificato ACM deve trovarsi nello stesso account dell'API
Utilizzo esteso delle chiavi (EKU)	Se presente, deve include`clientAuth` . Se assente, il certificato è accettato.
Utilizzo delle chiavi (KU)	Se presente, deve includere `digitalSignature` o`keyAgreement` . Se assente, il certificato è accettato.

Requisito	Description
Algoritmo chiave	Deve essere uno tra: RSA 2048, RSA 3072, RSA 4096, ECDSA (EC_Prime256v1), ECDSA (EC_SECP384R1) o ECDSA P-256 (EC_SECP521R1) P-384 P-521
Stato del certificato ACM	Deve essere `ISSUED`

> **Note**
>
> API Gateway non convalida la catena di fiducia tra i certificati leaf e intermedi. Inoltre, API Gateway non convalida l'intento del certificato o i vincoli di base (ad esempio) sui certificati intermedi. `CA:TRUE` Il tuo backend esegue queste convalide durante l'handshake TLS.

Rinnovo e propagazione del certificato

Quando un certificato viene modificato in ACM, API Gateway rileva l'aggiornamento e propaga automaticamente il nuovo certificato. Non è necessario ridistribuire lo stage e l'API non subisce tempi di inattività durante la rotazione.

La propagazione dei certificati alla fine è coerente. Durante l'aggiornamento, il backend potrebbe ricevere il vecchio o il nuovo certificato fino al completamento della propagazione.

La modalità di rinnovo del certificato dipende da come è stato emesso:

- Certificati emessi tramite AWS Autorità di certificazione privata (ACM-managed) (Opzione B): ACM rinnova automaticamente questi certificati. API Gateway rileva il rinnovo e si aggiorna automaticamente.
- Certificati emessi AWS Autorità di certificazione privata e importati (opzione C): ACM non rinnova automaticamente i certificati importati. È necessario reimportare il certificato rinnovato. Dopo aver reimportato il certificato, API Gateway rileva la modifica e si aggiorna automaticamente.
- Certificati importati dalla tua PKI (opzione A): devi reimportare il certificato rinnovato in ACM. Dopo aver reimportato il certificato, API Gateway rileva la modifica e si aggiorna automaticamente.

ACM invia notifiche di scadenza del certificato tramite Amazon. EventBridge Puoi utilizzare queste notifiche per impostare allarmi prima della scadenza di un certificato.

Comportamento e limitazioni del certificato ACM

Visualizzazione del certificato configurato

I certificati ACM non vengono visualizzati nelle risposte `GetClientCertificate` o `GetClientCertificates` API. Per visualizzare il certificato ACM ARN configurato su una fase, usa. GetStage Per visualizzare i dettagli del certificato, utilizza le API ACM e. DescribeCertificate GetCertificate

Riutilizzo in più fasi

È possibile allegare lo stesso certificato ACM a più fasi. Ogni fase fa riferimento in modo indipendente al certificato tramite il relativo ARN.

Le API dei certificati client non si applicano ai certificati ACM

I certificati ACM non sono risorse API. Gateway-managed Le `DeleteClientCertificate` API `GetClientCertificateUpdateClientCertificate`, e restituiscono a `NotFoundException` quando vengono chiamate con un certificato ACM ARN. Utilizza le API ACM per gestire il ciclo di vita del certificato.

Pulizia automatica delle associazioni di certificati

Quando rimuovi un certificato ACM da una fase, aggiorni una fase per utilizzare un certificato diverso o elimini una fase o un'API REST, API Gateway pulisce automaticamente l'associazione dei certificati. Non è richiesta alcuna azione manuale.

Eliminazione del certificato ACM

ACM non consente di eliminare un certificato mentre API Gateway ha un'associazione attiva con esso. Per eliminare il certificato da ACM, rimuovilo innanzitutto da tutte le fasi che lo fanno riferimento.

Usa un Gateway-generated certificato API per l'autenticazione backend in API Gateway

Puoi usare API Gateway per generare un certificato SSL e quindi usare la relativa chiave pubblica nel back-end per verificare che le richieste HTTP al sistema back-end provengano da API Gateway. Ciò consente al back-end HTTP di controllare e accettare solo le richieste che provengono da Amazon API Gateway, anche se il back-end è accessibile pubblicamente.

> **Note**
>
> Alcuni server back-end potrebbero non supportare l'autenticazione client SSL come fa API Gateway e potrebbero restituire un errore di certificato SSL. Per un elenco di server back-end non compatibili, consulta the section called “Note importanti”.

> **Tip**
>
> Per utilizzare il tuo CA-signed certificato ACM anziché un certificato Gateway-generated autofirmato API, consulta. the section called “Usa il tuo certificato ACM”

I certificati SSL generati da API Gateway sono autofirmati e solo la chiave pubblica di un certificato è visibile nella console API Gateway o tramite le API.

Argomenti

- Generazione di un certificato client tramite la console API Gateway
- Configurazione di un'API per usare i certificati SSL
- Test della chiamata per verificare la configurazione del certificato del client
- Configurazione di un server di back-end HTTPS per verificare il certificato del client
- Rotazione di un certificato client in scadenza

Generazione di un certificato client tramite la console API Gateway

1. Apri la console API Gateway all'indirizzo. https://console.aws.amazon.com/apigateway/
2. Scegli un REST o WebSocket un'API.
3. Nel riquadro di navigazione principale scegli Certificati client.
4. Nel riquadro Certificati client scegli Genera certificato client.
5. (Facoltativo) In Descrizione, immetti una descrizione.
6. Scegli Genera certificato per generare il certificato. API Gateway genera un nuovo certificato e restituisce il nuovo GUID del certificato, insieme alla chiave PEM-encoded pubblica.

A questo punto, puoi configurare un'API per usare il certificato.

Configurazione di un'API per usare i certificati SSL

Queste istruzioni presuppongono che la procedura in sia già stata completat Generazione di un certificato client tramite la console API Gateway.

1. Nella console API Gateway, crea o apri un WebSocket REST o un'API per cui desideri utilizzare il certificato client. Assicurati che l'API sia stata distribuita in una fase.
2. Nel riquadro di navigazione principale scegli Fasi.
3. Nella sezione Dettagli fase scegli Modifica.
4. Per Certificato client seleziona un certificato.
5. Scegli Save changes (Salva modifiche).

Dopo aver selezionato un certificato per l'API e avere eseguito il salvataggio, API Gateway usa il certificato per tutte le chiamate alle integrazioni HTTP nell'API.

Test della chiamata per verificare la configurazione del certificato del client

1. Scegli un metodo REST API. Seleziona la scheda Test. Potrebbe essere necessario scegliere il pulsante freccia destra per visualizzare la scheda Test.
2. Per Certificato client seleziona un certificato.
3. Scegli Test (Esegui test).

API Gateway presenta il certificato SSL scelto per il back-end HTTP per l'autenticazione dell'API.

Configurazione di un server di back-end HTTPS per verificare il certificato del client

Queste istruzioni presuppongono che Generazione di un certificato client tramite la console API Gateway sia già stata completata e che sia stata scaricata una copia del certificato client. Per scaricare un certificato client, chiama `clientcertificate:by-id` dell'API REST di API Gateway o `get-client-certificate` della AWS CLI.

Prima di configurare un server HTTPS di backend per verificare il certificato SSL client di API Gateway, devi aver ottenuto la chiave PEM-encoded privata e un certificato lato server forniti da un'autorità di certificazione affidabile.

Se il nome di dominio del server è `myserver.mydomain.com`, il valore CNAME del certificato del server deve essere `myserver.mydomain.com` o `*.mydomain.com`.

Tra le autorità di certificazione supportate figurano Let's Encrypt o una delle the section called "Autorità di certificazione supportate per l'integrazione HTTP e proxy HTTP".

Ad esempio, supponiamo che il file del certificato client sia `apig-cert.pem` e che la chiave privata del server e i file di certificato siano, rispettivamente, `server-key.pem` e `server-cert.pem`. Per un Node.js server nel backend, puoi configurare il server in modo simile a quanto segue:

```
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);
```

Per un'app node-express, puoi utilizzare i moduli client-certificate-auth per autenticare le richieste dei client con i certificati. PEM-encoded

Per gli altri server HTTPS, consulta la documentazione per il server.

Rotazione di un certificato client in scadenza

Il certificato client generato da API Gateway è valido per 365 giorni. È necessario ruotare il certificato prima che un certificato client in una fase API scada per evitare tempi di inattività per l'API.

Ruota un certificato client in scadenza utilizzando Console di gestione AWS

La seguente procedura mostra come ruotare un certificato client nella console per un'API implementata in precedenza.

1. Nel riquadro di navigazione principale scegli Certificati client.
2. Nel riquadro Certificati client scegli Genera certificato.
3. Aprire l'API per cui si desidera usare il certificato client.
4. Scegliere Stages (Fasi) per l'API selezionata e quindi scegliere una fase.

5. Nella sezione Dettagli fase scegli Modifica.
6. Per Certificato client seleziona il nuovo certificato.
7. Per salvare le impostazioni, scegli Salva modifiche.

Ruota un certificato client in scadenza utilizzando AWS CLI

È possibile verificare la data di scadenza del certificato chiamando il client dell'API REST API Gateway o il AWS CLI comando Certificate:by-id di get-client-certificate e controllando la proprietà expirationDate restituita. https://docs.aws.amazon.com/apigateway/latest/api/API_ClientCertificate.html#expirationDate

Per ruotare un certificato client, procedere come indicato di seguito:

1. Genera un nuovo certificato client chiamando clientcertificate:generate dell'API REST di API Gateway o il comando generate-client-certificate della AWS CLI . In questo tutorial si presuppone che l'ID del nuovo certificato client sia `ndiqef`.
2. Aggiorna il server di back-end per includere il nuovo certificato client. Non rimuovere il certificato client esistente.

   Alcuni server potrebbero richiedere un riavvio per completare l'aggiornamento. Per ulteriori informazioni, consulta la documentazione del server per vedere se è necessario riavviare il server durante l'aggiornamento.
3. Aggiornare la fase API per utilizzare il nuovo certificato client chiamando stage:update dell'API REST di API Gateway, con il nuovo ID del certificato client (`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"
    }
  ]
```

```
}
```

È anche possibile utilizzare il comando update-stage.

Se stai usando un'API, usa il comando update-stage. WebSocket `apigatewayv2` https://docs.aws.amazon.com/cli/latest/reference/apigatewayv2/update-stage.html

4. Aggiorna il server di back-end per eliminare il vecchio certificato.
5. Elimina il vecchio certificato da API Gateway chiamando clientcertificate:delete dell'API REST di API Gateway, specificando il client () del vecchio certificato: CertificateId `a1b2c3`

```
DELETE /clientcertificates/a1b2c3
```

È anche possibile chiamare il comando delete-client-certificate:

```
aws apigateway delete-client-certificate --client-certificate-id a1b2c3
```

Autorità di certificazione supportate da Gateway API per le integrazioni HTTP e proxy HTTP in Gateway API

Nell'elenco riportato di seguito sono indicate le autorità di certificazione supportate da API Gateway per le integrazioni HTTP, proxy HTTP e private.

```
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: usertrusteccccertificationauthority
  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
```

Utilizzalo AWS WAF per proteggere il tuo REST APIs in API Gateway

AWS WAF è un firewall per applicazioni Web che aiuta a proteggere le applicazioni Web e APIs dagli attacchi. Consente di configurare un set di regole denominato lista di controllo accessi Web (web ACL) per consentire, bloccare o contare le richieste Web in base a condizioni e regole di sicurezza Web personalizzabili definite dall'utente. Per ulteriori informazioni, consulta How AWS WAF Works.

Puoi utilizzarla AWS WAF per proteggere l'API REST di API Gateway da exploit web comuni, come attacchi SQL injection e cross-site scripting (XSS). Questi potrebbero influire sulla disponibilità e sulle prestazioni delle API, compromettere la sicurezza o consumare risorse eccessive. Ad esempio, puoi creare regole per consentire o bloccare richieste da intervalli di indirizzi IP specificati, richieste da blocchi CIDR, richieste provenienti da un paese o una regione specifici, richieste che contengono codice SQL dannoso o richieste contenenti script dannoso.

È anche possibile creare regole che corrispondono a una stringa specificata o un modello di espressione regolare in intestazioni HTTP, metodo, stringa di query, URI e il corpo della richiesta (entro i primi 64 KB). Inoltre, puoi creare regole per bloccare attacchi da utenti-agenti, bad bot e scraper di contenuti. Ad esempio, puoi usare le regole basate sulla frequenza per specificare il numero di richieste Web consentite da ogni IP client in un periodo di 5 minuti, costantemente aggiornato, finale.

> **Important**
>
> AWS WAF è la tua prima linea di difesa contro gli exploit web. Quando AWS WAF è abilitato su un'API, AWS WAF le regole vengono valutate prima di altre funzionalità di controllo degli accessi, come le policy delle risorse, le policy IAM, gli autorizzatori Lambda e gli autorizzatori Amazon Cognito. Ad esempio, se AWS WAF blocca l'accesso da un blocco CIDR consentito da una politica delle risorse, ha la AWS WAF precedenza e la politica delle risorse non viene valutata.

AWS WAF Per abilitare la tua API, devi fare quanto segue:

1. Usa la AWS WAF console, l' AWS SDK o la CLI per creare un ACL Web che contenga la combinazione desiderata AWS WAF di regole gestite e regole personalizzate. Per ulteriori informazioni, consulta la sezione Guida introduttiva AWS WAF e elenchi di controllo degli accessi Web (web). ACLs

   > **Important**
   >
   > API Gateway richiede un ACL AWS WAFV2 web per un'applicazione regionale o un ACL AWS WAF Classic regionale web.

2. Associa l'ACL AWS WAF web a una fase API. Puoi farlo utilizzando la AWS WAF console, l' AWS SDK, la CLI o utilizzando la console API Gateway.

Per associare un ACL AWS WAF Web a uno stadio API Gateway API utilizzando la console API Gateway

Per utilizzare la console API Gateway per associare un ACL AWS WAF Web a uno stadio API Gateway API esistente, attenersi alla seguente procedura:

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Seleziona un'API esistente o creane una nuova.
3. Nel riquadro di navigazione principale scegli Fasi, quindi seleziona una fase.
4. Nella sezione Dettagli fase scegli Modifica.
5. In Web Application Firewall (AWS WAF), seleziona l'ACL web.

   Se lo stai utilizzando AWS WAFV2, seleziona un ACL AWS WAFV2 web per un'applicazione regionale. L'ACL web e tutte AWS WAFV2 le altre risorse che utilizza devono trovarsi nella stessa regione dell'API.

   Se lo stai utilizzando AWS WAF Classic regionale, seleziona un ACL web regionale.
6. Scegli Save changes (Salva modifiche).

Associa un ACL AWS WAF Web a una fase API Gateway utilizzando il AWS CLI

Il associate-web-aclcomando seguente associa un ACL AWS WAFV2 Web per un'applicazione regionale a uno stadio API Gateway API esistente:

```
aws wafv2 associate-web-acl \
--web-acl-arn  arn:aws:wafv2:{region}:111122223333:regional/webacl/test-cli/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \
--resource-arn arn:aws:apigateway:{region}::/restapis/4wk1k4onj3/stages/prod
```

Il associate-web-aclcomando seguente associa un ACL AWS WAF Classic regionale Web a uno stadio API Gateway API esistente:

```
aws waf-regional associate-web-acl \
--web-acl-id 'aabc123a-fb4f-4fc6-becb-2b00831cadcf' \
--resource-arn 'arn:aws:apigateway:{region}::/restapis/4wk1k4onj3/stages/prod'
```

Associa un ACL AWS WAF web a uno stadio API utilizzando l'API REST AWS WAF

Per utilizzare l'API AWS WAFV2 REST per associare un ACL AWS WAFV2 Web per un'applicazione regionale a uno stadio API Gateway API esistente, utilizzate il comando AssociateWebACL, come nell'esempio seguente:

```
import boto3

wafv2 = boto3.client('wafv2')

wafv2.associate_web_acl(
    WebACLArn='arn:aws:wafv2:{region}:111122223333:regional/webacl/test/abc6aa3b-
fc33-4841-b3db-0ef3d3825b25',
    ResourceArn='arn:aws:apigateway:{region}::/restapis/4wk1k4onj3/stages/prod'
)
```

Per utilizzare l'API AWS WAF REST per associare un ACL AWS WAF Classic regionale Web a uno stadio API Gateway API esistente, utilizzate il comando AssociateWebACL, come nell'esempio seguente:

```
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'
)
```

Limita le richieste al tuo REST APIs per una migliore velocità di trasmissione in API Gateway

Puoi configurare la limitazione e le quote per evitare che vengano sopraffatti da troppe richieste. APIs Sia le limitazioni che le quote sono applicate sulla base del massimo sforzo e dovrebbero essere considerate come obbiettivi piuttosto che massimali garantiti delle richieste.

API Gateway limita la larghezza di banda della rete delle richieste nell'API mediante l'algoritmo di token bucket, dove un token rappresenta una richiesta. In particolare, API Gateway esamina la frequenza e il numero di richieste inviate rispetto APIs a tutti gli utenti del tuo account, per regione.

Nell'algoritmo di bucket token, un picco può consentire il superamento predefinito di tali limiti, ma anche altri fattori possono causare il superamento dei limiti in alcuni casi.

Quando le richieste inviate superano i limiti impostati per il tasso a regime e per i limiti di picco, API Gateway inizia a limitare le richieste. I clienti potrebbero ricevere risposte agli errori `429 Too Many Requests` a questo punto. Quando rileva queste eccezioni, il client può reinviare le richieste non riuscite in modo da limitare la velocità.

In qualità di sviluppatore di API, puoi impostare i limiti target per le singole fasi o metodi dell'API per migliorare le prestazioni complessive di tutto APIs il tuo account. In alternativa, puoi abilitare i piani di utilizzo per impostare limitazioni sugli invii di richieste al client in base a quote e tassi di richiesta specificati.

Argomenti

- Come sono applicate le impostazioni del limite del throttling in API Gateway
- Limitazione a livello di account per regione
- Configurazione del throttling a livello di API e destinazioni delle limitazioni a livello di fase in un piano di utilizzo
- Configurazione delle destinazioni di limitazione della larghezza di banda della rete a livello di fase
- Configurazione delle destinazioni della limitazione a livello di metodo in un piano di utilizzo

Come sono applicate le impostazioni del limite del throttling in API Gateway

Prima di configurare le impostazioni di limitazione (della larghezza di banda della rete) e quota per l'API, è utile capire quali sono i tipi di impostazioni relative alla limitazione (della larghezza di banda della rete) e come vengono applicate da Gateway API.

Amazon API Gateway fornisce due tipi base di impostazioni correlate alle impostazioni delle limitazioni:

- AWS i limiti di limitazione vengono applicati a tutti gli account e i clienti di una regione. Queste impostazioni del limite servono a evitare di sovraccaricare l'API, e di conseguenza il tuo account, con troppe richieste. Questi limiti sono stabiliti AWS e non possono essere modificati da un cliente.
- I limiti per account vengono applicati a tutti i membri APIs di un account in una regione specificata. Il limite di tariffa a livello di account può essere aumentato su richiesta: sono possibili limiti più elevati con APIs timeout più brevi e carichi utili inferiori. Per richiedere un aumento dei limiti

di limitazione a livello di account per Regione, contatta il Centro assistenza AWS. Per ulteriori informazioni, consulta Quote e note importanti. Tieni presente che questi limiti non possono essere superiori ai limiti di limitazione. AWS

- I limiti di throttling per API e per fase vengono applicati a livello di metodo API per una fase. È possibile configurare le stesse impostazioni per tutti i metodi o configurare impostazioni di throttling diverse per ciascun metodo. Tieni presente che questi limiti non possono essere superiori ai limiti di AWS limitazione.
- Limiti di throttling per client vengono applicati ai client che utilizzano le chiavi API associate al piano di utilizzo come identificatore client. Ricorda che questi limiti non possono essere più alti dei limiti per account.

Le impostazioni correlate alla limitazione (della larghezza di banda della rete) di Gateway API vengono applicate nell'ordine seguente:

1. Limiti di throttling per metodo o per client impostati per una fase API in un piano di utilizzo
2. Limiti di throttling per metodo impostati per una fase API
3. Limitazione a livello di account per regione
4. AWS Limitazione regionale

Limitazione a livello di account per regione

Per impostazione predefinita, API Gateway limita le richieste allo stato stazionario (RPS) APIs all'interno di un AWS account, per regione. Inoltre, limita il burst (ovvero la dimensione massima del bucket) per tutti gli elementi di un account, per regione APIs . AWS In API Gateway, il limite dei picchi rappresenta il numero massimo di invii di richieste che API Gateway può gestire prima di restituire risposte di errore `429 Too Many Requests`. Per ulteriori informazioni sulla limitazione delle quote, vedere Quote e note importanti.

Configurazione del throttling a livello di API e destinazioni delle limitazioni a livello di fase in un piano di utilizzo

In un piano di utilizzo puoi impostare una destinazione della limitazione (della larghezza di banda della rete) per tutti i metodi a livello di API o di fase. È possibile specificare una velocità di limitazione (della larghezza di banda della rete)ovvero la frequenza, in richieste al secondo, con cui i token vengono aggiunti al bucket di token. È anche possibile specificare un burst di limitazione (della larghezza di banda della rete), ovvero la capacità del bucket di token.

È possibile utilizzare la AWS CLI e creare un Console di gestione AWS piano di utilizzo. SDKs Per ulteriori informazioni su come creare un piano di utilizzo, consulta ???.

Configurazione delle destinazioni di limitazione della larghezza di banda della rete a livello di fase

È possibile utilizzare la AWS CLI e la Console di gestione AWS per creare obiettivi SDKs di throttling a livello di fase.

Per ulteriori informazioni su come utilizzare per creare obiettivi di limitazione a livello Console di gestione AWS di fase, consulta. ??? Per ulteriori informazioni su come utilizzare la AWS CLI per creare obiettivi di throttling a livello di fase, consulta create-stage.

Configurazione delle destinazioni della limitazione a livello di metodo in un piano di utilizzo

Puoi impostare destinazioni della limitazione aggiuntive a livello di metodo in Usage Plans (Piani di utilizzo) come mostrato in Creazione di un piano di utilizzo. Nella console API Gateway, questi vengono impostati specificando `Resource=<resource>`, `Method=<method>` nell'impostazione Configure Method Throttling (Configurazione del throttling). Ad esempio, ad esempio, è possibile specificare,. PetStore `Resource=/pets` `Method=GET`

REST API private in Gateway API

Un'API privata è una REST API richiamabile solo all'interno di un VPC Amazon. Puoi accedere all'API utilizzando un endpoint VPC di interfaccia, che è un'interfaccia di rete endpoint creata nel VPC. Gli endpoint dell'interfaccia sono alimentati daAWS PrivateLink, una tecnologia che consente di accedere privatamente ai AWS servizi utilizzando indirizzi IP privati.

Puoi anche utilizzarla Direct Connect per stabilire una connessione da una rete locale ad Amazon VPC e quindi accedere alla tua API privata tramite quella connessione. In ogni caso, il traffico alla tua API privata utilizza connessioni sicure ed è isolato dalla rete Internet pubblica. Il traffico non esce dalla rete Amazon.

Best practice per le API private

Di seguito sono riportate le best practice consigliate per la creazione di un'API privata:

- Usa un unico endpoint VPC per accedere a più API private. In questo modo si riduce il numero di endpoint VPC di cui potresti aver bisogno.

- Associa il tuo endpoint VPC alla tua API. In questo modo viene creato un record DNS alias di Route 53 e risulta più semplice invocare l'API privata.
- Attiva il DNS privato per il tuo VPC. Quando attivi il DNS privato per il tuo VPC, puoi invocare l'API all'interno di un VPC senza passare l'intestazione `Host` o `x-apigw-api-id`.

  Se attivi il DNS privato, non puoi accedere all'endpoint predefinito per le API pubbliche. Per accedere all'endpoint predefinito per le API pubbliche, puoi disattivare il DNS privato, creare una zona ospitata privata per ogni API privata nel tuo VPC e quindi effettuare il provisioning dei record richiesti in Route 53. Questo consente la risoluzione della tua API e ti permette comunque di invocare l'endpoint pubblico predefinito dal tuo VPC. Per ulteriori informazioni consulta Creating a private hosted zone (Creazione di una zona ospitata privata).
- Limita l'accesso alla tua API privata a VPC o endpoint VPC specifici. Aggiungi le condizioni `aws:SourceVpc` o `aws:SourceVpce` alla policy delle risorse dell'API per limitare l'accesso.
- Per la massima sicurezza del perimetro dati, puoi creare una policy degli endpoint VPC. Questo controlla l'accesso agli endpoint VPC che possono invocare la tua API privata.

Considerazioni relative alle API private

Le seguenti considerazioni potrebbero influire sull'utilizzo delle API private:

- Sono supportate solo le REST API.
- Non è possibile convertire un'API privata in un'API ottimizzata per l'edge.
- Le API private supportano solo TLS 1.2. Le versioni precedenti di TLS non sono supportate.
- Se effettui una richiesta utilizzando il HTTP/2 protocollo, alla richiesta viene applicato il protocollo. HTTP/1.1
- Non è possibile impostare il tipo di indirizzo IP per le API private per consentire solo agli indirizzi IPv4 di invocare l'API privata. È supportato solo il tipo dualstack. Per ulteriori informazioni, consulta the section called "Tipi di indirizzo IP per REST API in Gateway API".
- Per inviare il traffico utilizzando l'API privata, è possibile usare tutti i tipi di indirizzo IP supportati da Amazon VPC. È possibile inviare traffico dualstack e IPv6 configurando le impostazioni sull'endpoint VPC. Questa configurazione non può essere modificata tramite Gateway API. Per ulteriori informazioni, consulta Aggiungere il supporto IPv6 per il VPC.
- Gli endpoint VPC per le API private sono soggetti alle stesse limitazioni degli altri endpoint VPC dell'interfaccia. Per ulteriori informazioni, consulta Accedere a un AWS servizio utilizzando un endpoint VPC di interfaccia nella Guida. AWS PrivateLink Per ulteriori informazioni sull'utilizzo di

API Gateway con VPC condivisi e sottoreti condivise, consulta Shared subnetsnella Guida AWS PrivateLink .

Passaggi successivi per le API private

Per informazioni su come creare un'API privata e associare un endpoint VPC, consulta the section called “Creazione di un'API privata”. Per seguire un tutorial in cui crei dipendenze CloudFormation e un'API privata in, vedi. Console di gestione AWSthe section called “Tutorial: creazione di una REST API privata”

Creazione di un'API privata

Prima di creare un'API privata, devi creare un endpoint VPC per Gateway API. Quindi devi creare la tua API privata e collegarvi una policy delle risorse. Facoltativamente, puoi associare il tuo endpoint VPC alla tua API privata per semplificare l'invocazione dell'API. Infine, implementa la tua API.

Nelle seguenti procedure viene descritto come effettuare questa operazione. Puoi creare un'API REST privata utilizzando Console di gestione AWS, AWS CLI o un AWS SDK.

Prerequisiti

Per eseguire questi passaggi, devi disporre di un VPC completamente configurato. Per informazioni su come creare un VPC, consulta Creare solo un VPC nella Guida per l'utente di Amazon VPC. Per seguire tutti i passaggi consigliati durante la creazione del VPC, abilita il DNS privato. In questo modo puoi invocare la tua API all'interno di un VPC senza dover passare l'host o l'intestazione `x-apigw-api-id`.

Per abilitare il DNS privato, è necessario che gli attributi `enableDnsSupport` e `enableDnsHostnames` del VPC siano impostati su `true`. Per ulteriori informazioni, consulta DNS Support in Your VPC e Updating DNS Support for Your VPC.

Passaggio 1: creare un endpoint VPC per Gateway API nel VPC

La procedura seguente descrive come creare un endpoint VPC per Gateway API. Per creare un endpoint VPC per API Gateway, specifichi il `execute-api` dominio Regione AWS in cui creare la tua API privata. Il dominio `execute-api` è il servizio del componente Gateway API per l'esecuzione dell'API.

Quando crei l'endpoint VPC per Gateway API, devi specificare le impostazioni DNS. Se disattivi il DNS privato, puoi accedere all'API solo tramite DNS pubblico. Per ulteriori informazioni, consulta

the section called “Problema: non riesco a connettermi alla mia API pubblica da un endpoint VPC di Gateway API”.

Console di gestione AWS

Per creare un endpoint VPC dell'interfaccia per Gateway API

1. Accedi Console di gestione AWS e apri la console Amazon VPC all'indirizzo. https://console.aws.amazon.com/vpc/
2. Dal riquadro di navigazione, in Cloud privato virtuale, scegli Endpoint.
3. Seleziona Crea endpoint.
4. (Facoltativo) Per Tag nome, immetti un nome che consenta di identificare l'endpoint VPC.
5. Per Service category (Categoria servizio), scegli AWS services.
6. In Servizi, sulla barra di ricerca, inserisci **`execute-api`**. Quindi, scegli l'endpoint del servizio API Gateway in Regione AWS cui creerai la tua API. Il nome del servizio deve essere simile a `com.amazonaws.us-east-1.execute-api` e il Tipo deve essere Interfaccia.
7. Per VPC, scegliere il VPC in cui si desidera creare l'endpoint.
8. (Facoltativo) Per disattivare Abilita nome DNS privato, scegli Impostazioni aggiuntive e quindi deseleziona Abilita nome DNS privato.
9. In Sottoreti, scegli le zone di disponibilità in cui hai creato le interfacce di rete dell'endpoint. Per migliorare la disponibilità della tua API, scegli più sottoreti.
10. In Security group (Gruppo di sicurezza), selezionare il gruppo di sicurezza da associare alle interfacce di rete dell'endpoint VPC.

    Il gruppo di sicurezza scelto deve essere impostato in modo da consentire il traffico HTTPS in entrata sulla porta TCP 443 da un range di IP nel VPC o da un altro gruppo di sicurezza nel tuo VPC.
11. Per Policy, esegui una delle seguenti operazioni:
    - Se non hai creato la tua API privata o non desideri configurare una policy personalizzata degli endpoint VPC, scegli Accesso completo.
    - Se hai già creato un'API privata e desideri configurare una policy personalizzata degli endpoint VPC, puoi inserire tale policy. Per ulteriori informazioni, consulta the section called “Utilizzo di policy di endpoint VPC per API private”.

    Puoi aggiornare la policy degli endpoint VPC dopo aver creato l'endpoint VPC. Per ulteriori informazioni, consulta Update a VPC endpoint policy.

12. Seleziona Crea endpoint.
13. Copia l'ID dell'endpoint VPC risultante, poiché potresti utilizzarlo nei passaggi successivi.

AWS CLI

Il comando create-vpc-endpoint seguente crea un endpoint VPC:

```
aws ec2 create-vpc-endpoint \
    --vpc-id vpc-1a2b3c4d \
    --vpc-endpoint-type Interface \
    --service-name com.amazonaws.us-east-1.execute-api \
    --subnet-ids subnet-7b16de0c \
    --security-group-id sg-1a2b3c4d
```

Copia l'ID dell'endpoint VPC risultante, poiché potresti utilizzarlo nei passaggi successivi.

Fase 2: creare un'API privata

Dopo aver creato l'endpoint VPC, creerai una REST API privata. La procedura seguente mostra come creare un'API privata.

Console di gestione AWS

Per creare un'API privata

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Seleziona Create API (Crea API).
3. In API REST, scegliere Crea.
4. In Nome, immetti un nome.
5. (Facoltativo) In Descrizione, immetti una descrizione.
6. Per Tipo di endpoint API scegli Privato.
7. (Facoltativo) Per ID endpoint VPC, inserisci un ID di endpoint VPC.

   Se associ un ID endpoint VPC alla tua API privata, puoi invocarla dall'interno del VPC senza sostituire un'intestazione `Host` o passare un `x-apigw-api-id header`. Per ulteriori informazioni, consulta the section called “(Facoltativo) Associazione o dissociazione di un endpoint VPC da una REST API privata”.

8. Per Tipo di indirizzo IP, scegli Dualstack.
9. Seleziona Create API (Crea API).

Dopo aver completato i passaggi precedenti, puoi seguire le istruzioni indicate alla sezione the section called “Nozioni di base sulla console REST API” per configurare i metodi e le integrazioni per l'API. Per distribuire l'API, segui il passaggio 3 e collega una policy delle risorse all'API.

AWS CLI

Il comando create-rest-api seguente crea un’API privata:

```
aws apigateway create-rest-api \
        --name 'Simple PetStore (AWS CLI, Private)' \
        --description 'Simple private PetStore API' \
        --region us-west-2 \
        --endpoint-configuration '{ "types": ["PRIVATE"], "ipAddressType":
 "dualstack" }'
```

In caso di esito positivo, la chiamata restituisce un output simile al seguente:

```
{
    "createdDate": "2017-10-13T18:41:39Z",
    "description": "Simple private PetStore API",
    "endpointConfiguration": {
        "types": [
            "PRIVATE"
        ],
        "ipAddressType": "dualstack"
    },
    "id": "0qzs2sy7bh",
    "name": "Simple PetStore (AWS CLI, Private)"
}
```

Dopo aver completato i passaggi precedenti, puoi seguire le istruzioni indicate alla sezione the section called “Tutorial: creare un'API REST utilizzando AWS SDK o AWS CLI” per configurare i metodi e le integrazioni per l'API. Per distribuire l'API, segui il passaggio 3 e collega una policy delle risorse all'API.

SDK JavaScript v3

L'esempio seguente mostra come creare un'API privata utilizzando l' AWS SDK per la JavaScript v3:

```
import {APIGatewayClient, CreateRestApiCommand} from "@aws-sdk/client-api-gateway";
const apig = new APIGatewayClient({region:"us-east-1"});

const input = { // CreateRestApiRequest
  name: "Simple PetStore (JavaScript v3 SDK, private)", // required
  description: "Demo private API created using the AWS SDK for JavaScript v3",
  version: "0.00.001",
  endpointConfiguration: { // EndpointConfiguration
    types: [ "PRIVATE"],
  },
};

export const handler = async (event) => {
const command = new CreateRestApiCommand(input);
try {
  const result = await apig.send(command);
  console.log(result);
} catch (err){
  console.error(err)
 }
};
```

In caso di esito positivo, la chiamata restituisce un output simile al seguente:

```
{
  apiKeySource: 'HEADER',
  createdDate: 2024-04-03T17:56:36.000Z,
  description: 'Demo private API created using the AWS SDK for JavaScript v3',
  disableExecuteApiEndpoint: false,
  endpointConfiguration: { types: [ 'PRIVATE' ] },
  id: 'abcd1234',
  name: 'Simple PetStore (JavaScript v3 SDK, private)',
  rootResourceId: 'efg567',
  version: '0.00.001'
}
```

Dopo aver completato i passaggi precedenti, puoi seguire le istruzioni indicate alla sezione the section called “Tutorial: creare un'API REST utilizzando AWS SDK o AWS CLI” per configurare i metodi e le integrazioni per l'API. Per distribuire l'API, segui il passaggio 3 e collega una policy delle risorse all'API.

Python SDK

L'esempio seguente mostra come creare un'API privata utilizzando l' AWS SDK per Python:

```
import json
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

def lambda_handler(event, context):
    try:
      result = apig.create_rest_api(
      name='Simple PetStore (Python SDK, private)',
      description='Demo private API created using the AWS SDK for Python',
      version='0.00.001',
      endpointConfiguration={
          'types': [
             'PRIVATE',
          ],
      },
      )
    except botocore.exceptions.ClientError as error:
            logger.exception("Couldn't create private API %s.", error)
            raise
    attribute=["id", "name", "description", "createdDate", "version",
 "apiKeySource", "endpointConfiguration"]
    filtered_data ={key:result[key] for key in attribute}
    result = json.dumps(filtered_data, default=str, sort_keys='true')
    return result
```

In caso di esito positivo, la chiamata restituisce un output simile al seguente:

```
"{\"apiKeySource\": \"HEADER\", \"createdDate\": \"2024-04-03 17:27:05+00:00\",
 \"description\": \"Demo private API created using the AWS SDK for \",
 \"endpointConfiguration\": {\"types\": [\"PRIVATE\"]}, \"id\": \"abcd1234\", \"name
\": \"Simple PetStore (Python SDK, private)\", \"version\": \"0.00.001\"}"
```

Dopo aver completato i passaggi precedenti, puoi seguire le istruzioni indicate alla sezione the section called “Tutorial: creare un'API REST utilizzando AWS SDK o AWS CLI” per configurare

i metodi e le integrazioni per l'API. Per distribuire l'API, segui il passaggio 3 e collega una policy delle risorse all'API.

Passaggio 3: impostare una policy delle risorse per un'API privata.

La tua API privata corrente è inaccessibile a tutti i VPC. Utilizza una policy delle risorse per concedere ai tuoi VPC ed endpoint VPC l'accesso alle tue API private. Puoi concedere l'accesso a un endpoint VPC in qualsiasi account. AWS

La tua policy delle risorse deve contenere le condizioni `aws:SourceVpc` o `aws:SourceVpce` per limitare l'accesso. È consigliabile identificare specifici VPC ed endpoint VPC e non creare una policy delle risorse che consenta l'accesso a tutti i VPC e gli endpoint VPC.

La procedura seguente descrive come collegare una policy delle risorse all'API.

Console di gestione AWS

1. Accedi alla console API Gateway all'indirizzo. https://console.aws.amazon.com/apigateway
2. Scegliere una REST API.
3. Nel riquadro di navigazione principale, scegli Policy delle risorse.
4. Scegli Crea policy.
5. Scegli Seleziona un modello e quindi VPC di origine.
6. Sostituisci `{{vpcID}}` (incluse le parentesi graffe) con l'ID VPC.
7. Scegli Save changes (Salva modifiche).

AWS CLI

Il comando update-rest-api seguente collega una policy di risorse a un'API esistente:

```
aws apigateway update-rest-api \
    --rest-api-id a1b2c3 \
    --patch-operations op=replace,path=/
policy,value='"{\"jsonEscapedPolicyDocument\"}"'
```

Può essere utile anche controllare quali risorse hanno accesso al tuo endpoint VPC. Per controllare quali risorse hanno accesso all'endpoint VPC, collega una policy di endpoint all'endpoint VPC. Per ulteriori informazioni, consulta the section called "Utilizzo di policy di endpoint VPC per API private".

(Facoltativo) Associazione o dissociazione di un endpoint VPC da una REST API privata

Quando si associa un endpoint VPC all'API privata, Gateway API genera un nuovo record DNS alias di Route 53. Puoi utilizzare questo record per invocare le API private come accade per le API pubbliche, senza sostituire un'intestazione `Host` o passare un'intestazione `x-apigw-api-id`.

L'URL di base generato è nel formato seguente:

```
https://{rest-api-id}-{vpce-id}.execute-api.{region}.amazonaws.com/{stage}
```

Associate a VPC endpoint (Console di gestione AWS)

Puoi associare un endpoint VPC alla tua API privata durante o dopo la sua creazione. La procedura seguente spiega come associare un endpoint VPC a un'API creata in precedenza.

Per associare un endpoint VPC a un'API privata

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli l'API privata.
3. Nel riquadro di navigazione principale, scegli Policy delle risorse.
4. Modifica la policy delle risorse per consentire le chiamate dall'endpoint VPC aggiuntivo.
5. Nel riquadro di navigazione principale, scegli Impostazioni API.
6. Nella sezione Dettagli API, scegli Modifica.
7. Per ID endpoint VPC seleziona gli ID degli endpoint VPC aggiuntivi.
8. Selezionare Salva.
9. Implementa nuovamente l'API per rendere effettive le modifiche.

Dissociate a VPC endpoint (Console di gestione AWS)

Per annullare l'associazione di endpoint VPC da una REST API privata

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli l'API privata.
3. Nel riquadro di navigazione principale, scegli Policy delle risorse.
4. Modifica la policy delle risorse per rimuovere le menzioni dell'endpoint VPC di cui desideri annullare l'associazione all'API privata.

5. Nel riquadro di navigazione principale, scegli Impostazioni API.
6. Nella sezione Dettagli API, scegli Modifica.
7. Per ID endpoint VPC scegli la X per annullare l'associazione dell'endpoint VPC.
8. Selezionare Salva.
9. Implementa nuovamente l'API per rendere effettive le modifiche.

Associate a VPC endpoint (AWS CLI)

Il comando create-rest-api seguente associa gli endpoint VPC al momento della creazione dell'API:

```
aws apigateway create-rest-api \
    --name Petstore \
    --endpoint-configuration '{ "types": ["PRIVATE"], "vpcEndpointIds" :
 ["vpce-0212a4ababd5b8c3e", "vpce-0393a628149c867ee"] }' \
    --region us-west-2
```

L'output sarà simile al seguente:

```
{
    "apiKeySource": "HEADER",
    "endpointConfiguration": {
        "types": [
            "PRIVATE"
        ],
        "vpcEndpointIds": [
            "vpce-0212a4ababd5b8c3e",
            "vpce-0393a628149c867ee"
        ]
    },
    "id": "u67n3ov968",
    "createdDate": 1565718256,
    "name": "Petstore"
}
```

Il comando update-rest-api seguente associa gli endpoint VPC a un'API creata in precedenza:

```
aws apigateway update-rest-api \
```

```
    --rest-api-id u67n3ov968 \
    --patch-operations "op='add',path='/endpointConfiguration/
vpcEndpointIds',value='vpce-01d622316a7df47f9'" \
    --region us-west-2
```

L'output sarà simile al seguente:

```
{
    "name": "Petstore",
    "apiKeySource": "1565718256",
    "tags": {},
    "createdDate": 1565718256,
    "endpointConfiguration": {
        "vpcEndpointIds": [
            "vpce-0212a4ababd5b8c3e",
            "vpce-0393a628149c867ee",
            "vpce-01d622316a7df47f9"
        ],
        "types": [
            "PRIVATE"
        ]
    },
    "id": "u67n3ov968"
}
```

Implementa nuovamente l'API per rendere effettive le modifiche.

Disassociate a VPC endpoint (AWS CLI)

Il comando update-rest-api seguente dissocia un endpoint VPC da un'API privata:

```
aws apigateway update-rest-api \
    --rest-api-id u67n3ov968 \
    --patch-operations "op='remove',path='/endpointConfiguration/
vpcEndpointIds',value='vpce-0393a628149c867ee'" \
    --region us-west-2
```

L'output sarà simile al seguente:

```
{
    "name": "Petstore",
    "apiKeySource": "1565718256",
    "tags": {},
```

```
    "createdDate": 1565718256,
    "endpointConfiguration": {
        "vpcEndpointIds": [
            "vpce-0212a4ababd5b8c3e",
            "vpce-01d622316a7df47f9"
        ],
        "types": [
            "PRIVATE"
        ]
    },
    "id": "u67n3ov968"
}
```

Implementa nuovamente l'API per rendere effettive le modifiche.

Passaggio 4: implementare un'API privata

Per implementare la tua API, crea un'implementazione API e associala a una fase. La procedura seguente mostra come implementare la tua API privata.

Console di gestione AWS

Per distribuire un'API privata

1. Scegliere l'API.
2. Seleziona Deploy API (Distribuisci API).
3. In Fase, seleziona Nuova fase.
4. Per Nome fase immetti il nome di una fase.
5. (Facoltativo) In Descrizione, immetti una descrizione.
6. Seleziona Implementa.

AWS CLI

Il comando create-deployment seguente implementa un'API privata:

```
aws apigateway create-deployment --rest-api-id a1b2c3 \
  --stage-name test \
  --stage-description 'Private API test stage' \
  --description 'First deployment'
```

Risoluzione dei problemi relativi all'API privata

Di seguito sono forniti alcuni consigli per la risoluzione dei problemi relativi a errori e problemi che potrebbero verificarsi durante la creazione di un'API privata.

Problema: non riesco a connettermi alla mia API pubblica da un endpoint VPC di Gateway API

Durante la creazione del VPC puoi configurare le impostazioni DNS. È consigliabile attivare il DNS privato per il VPC. Se decidi di disattivare il DNS privato, puoi accedere all'API solo tramite DNS pubblico.

Se abiliti il DNS privato, non puoi accedere all'endpoint predefinito di un'API pubblica di Gateway API dal tuo endpoint VPC. Puoi accedere a un'API con un nome di dominio personalizzato.

Se crei un nome di dominio personalizzato regionale utilizzi un record alias di tipo A, mentre se crei un nome di dominio personalizzato ottimizzato per l'edge non ci sono restrizioni per il tipo di record. Puoi accedere a queste API pubbliche con il DNS privato abilitato. Per ulteriori informazioni, consulta Problema: non riesco a connettermi alla mia API pubblica da un endpoint VPC di Gateway API.

Problema: la mia API restituisce **{"Message» :"Utente: anonymous non è autorizzato a eseguire: execute-api:Invoke on resource: arn:aws:execute-api:us-east-1: ********/****/****/ "}**

Nella tua politica sulle risorse, AWS se imposti il Principal su un Principal, come il seguente:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:role/developer",
                    "arn:aws:iam::111122223333:role/Admin"
                ]
            },
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/stage/GET/pets"
            ]
```

```
        }
    ]
}
```

Devi utilizzare l'autorizzazione `AWS_IAM` per ogni metodo dell'API, altrimenti l'API restituirà il messaggio di errore precedente. Per ulteriori indicazioni su come attivare l'autorizzazione `AWS_IAM` per un metodo, consulta the section called “Metodi”.

Problema: non riesco a capire se il mio endpoint VPC è associato alla mia API

Se associ o dissoci un endpoint VPC dalla tua API privata, devi implementarla nuovamente. Il completamento dell'operazione di aggiornamento potrebbe richiedere alcuni minuti a causa della propagazione DNS. Durante questo periodo, l'API è disponibile, ma la propagazione DNS per gli URL DNS appena generati potrebbe essere ancora in corso. Se dopo alcuni minuti i nuovi URL non vengono risolti nel DNS, ti consigliamo di reimplementare l'API.

Nomi di dominio personalizzati per API private in Gateway API

Puoi creare un nome di dominio personalizzato per la tua API privata. Un nome di dominio personalizzato privato consente di fornire un URL più semplice e intuitivo ai chiamanti dell'API. Con un nome di dominio personalizzato privato, puoi ridurre la complessità, configurare le misure di sicurezza durante l'handshake TLS e controllare il ciclo di vita del certificato del tuo nome di dominio utilizzando AWS Certificate Manager (ACM). Per ulteriori informazioni, consulta the section called “Protezione della chiave privata del certificato per il nome di dominio personalizzato”.

I nomi di dominio personalizzati per le API private non devono essere necessariamente univoci tra più account. È possibile creare `example.private.com` nell'account 111122223333 e nell'account 555555555555, purché il certificato ACM copra il nome di dominio. Per identificare un nome di dominio personalizzato privato, usa l'ARN del nome di dominio personalizzato privato. Questo identificatore è univoco per i nomi di dominio personalizzati privati.

Quando crei un nome di dominio personalizzato privato in Gateway API, sei un provider di API. Puoi fornire il tuo nome di dominio personalizzato privato ad altri Account AWS utilizzando API Gateway o (). AWS Resource Access Manager AWS RAM

Quando invochi un nome di dominio personalizzato privato, sei un consumatore di API. Puoi utilizzare un nome di dominio personalizzato privato dal tuo Account AWS o da un altro Account AWS.

Quando utilizzi un nome di dominio personalizzato privato, crei un'associazione di accesso al nome di dominio tra un endpoint VPC e un nome di dominio personalizzato privato. Con un'associazione

di accesso al nome di dominio, i consumatori di API possono invocare il tuo nome di dominio personalizzato privato mentre sono isolati dalla rete Internet pubblica. Per ulteriori informazioni, consulta the section called “Provider di API e consumatori di API”.

Protezione della chiave privata del certificato per il nome di dominio personalizzato

Quando richiedi un SSL/TLS certificato utilizzando ACM per creare il tuo nome di dominio personalizzato per le API private, ACM genera una public/private coppia di chiavi. La coppia di chiavi viene generata al momento dell'importazione di un certificato. La chiave di accesso pubblica diventa parte del certificato. Per archiviare in modo sicuro la chiave privata, ACM crea un'altra chiave utilizzando AWS KMS, chiamata chiave KMS, l'alias. aws/acm AWS KMS utilizza questa chiave per crittografare la chiave privata del certificato. Per ulteriori informazioni, consulta Data protection in AWS Certificate Manager nella Guida per l'utente di AWS Certificate Manager .

API Gateway utilizza AWS TLS Connection Manager, un servizio accessibile solo a Servizi AWS, per proteggere e utilizzare le chiavi private del certificato. Quando utilizzi il certificato ACM per creare un nome di dominio personalizzato API Gateway, API Gateway associa il certificato a AWS TLS Connection Manager. A tale scopo, creiamo un grant in base alla tua AWS chiave AWS KMS gestita. Questa concessione consente a TLS Connection Manager di utilizzare AWS KMS per decrittografare la chiave privata del certificato. TLS Connection Manager utilizza il certificato e la chiave privata decrittografata (testo in chiaro) per stabilire una connessione sicura (SSL/TLS sessione) con i client dei servizi API Gateway. Quando l'associazione del certificato a un servizio Gateway API viene annullata, la concessione viene ritirata. Per ulteriori informazioni, consulta Grants nella Guida per gli sviluppatori di AWS Key Management Service .

Per ulteriori informazioni, consulta the section called “Crittografia dei dati inattivi”.

Considerazioni sui nomi di dominio personalizzati

Le seguenti considerazioni potrebbero influire sull’utilizzo dei nomi di dominio personalizzati:

- Gateway API impiega circa 15 minuti per allocare il nome di dominio personalizzato privato.
- Se si aggiorna il certificato ACM, Gateway API impiega circa 15 minuti per completare l’operazione. Durante questo periodo, il nome di dominio è nello stato `UPDATING` ed è ancora possibile accedervi.
- Per invocare un nome di dominio personalizzato privato, è necessario creare un'associazione di accesso al nome di dominio personalizzato. Affinché un'associazione di accesso al nome di dominio sia pronta sono necessari circa 15 minuti.

- Il nome di dominio personalizzato privato ARN contiene i e i. *account-id domain-name-id* Quando si crea un nome di dominio, Gateway API utilizza il formato ARN `arn:partition:apigateway:region::/domainnames/domain-name`. Quando si accede a un nome di dominio personalizzato privato, si utilizza il formato ARN `arn:partition:apigateway:region:account-id:/domainnames/domain-name` +*domain-name-id*.

  Potrebbe essere necessario modificare le autorizzazioni IAM per consentire l'accesso a un nome di dominio privato dopo averlo creato.
- Non è possibile invocare nomi di dominio personalizzati privati con lo stesso nome dallo stesso endpoint VPC. Ad esempio, per invocare `arn:aws:apigateway:us-west-2:111122223333:/domainnames/private.example.com+abcd1234` e `arn:aws:apigateway:us-west-2:111122223333:/domainnames/private.example.com+xyz000`, è necessario associare ciascun nome di dominio personalizzato privato a un endpoint VPC diverso.
- Sono supportati i certificati con caratteri jolly, ad esempio un certificato per `*.private.example.com`.
- I nomi di dominio personalizzati con caratteri jolly non sono supportati.
- Sono supportati solo i certificati RSA con una lunghezza di chiave di 2048 bit e i certificati ECDSA con lunghezze di chiave di 256 bit e 384 bit.
- Non è possibile impostare il tipo di indirizzo IP per le API private per consentire solo agli indirizzi IPv4 di invocare l'API privata. È supportato solo il tipo dualstack. Per ulteriori informazioni, consulta the section called "Tipi di indirizzo IP per REST API in Gateway API".
- Per inviare il traffico utilizzando l'API privata, è possibile usare tutti i tipi di indirizzo IP supportati da Amazon VPC. È possibile inviare traffico dualstack e IPv6 configurando le impostazioni sull'endpoint VPC. Questa configurazione non può essere modificata tramite Gateway API. Per ulteriori informazioni, consulta Aggiungere il supporto IPv6 per il VPC.
- Multi-level la mappatura del percorso di base, ad esempio la mappatura dell'API privata a `/developers/feature`, non è supportata, ma puoi utilizzare una regola di routing per creare una condizione di percorso a più livelli. Per ulteriori informazioni, consulta the section called "Invia traffico al tuo APIs tramite il tuo nome di dominio personalizzato in API Gateway".
- Non è possibile impostare una versione TLS minima per un nome di dominio personalizzato privato. A tutti i nomi di dominio personalizzati privati è associata la policy di sicurezza `TLS-1-2`.

- È possibile utilizzare la policy degli endpoint VPC per controllare l'accesso a un nome di dominio personalizzato privato. Per ulteriori informazioni, consulta gli esempi 4 e 5 in the section called “Utilizzo di policy di endpoint VPC per API private”.
- È necessario creare una policy delle risorse separata per l'API privata e per il nome di dominio personalizzato privato. Per invocare un nome di dominio personalizzato privato, un consumatore di API deve disporre dell'accesso dalla policy delle risorse dei nomi di dominio personalizzati privati, dalla policy delle risorse delle API private e da qualsiasi autorizzazione o policy di endpoint VPC sull'API privata.

Considerazioni sull'utilizzo di nomi di dominio personalizzati privati con altre risorse Gateway API

Le seguenti considerazioni potrebbero influire sull’utilizzo dei nomi di dominio personalizzati privati con altre risorse Gateway API:

- Non è possibile inviare il traffico da un nome di dominio personalizzato privato a un’API pubblica.
- Quando un'API privata viene mappata a un nome di dominio personalizzato privato, non è possibile modificare il tipo di endpoint dell'API.
- Non è possibile eseguire la migrazione di un nome di dominio personalizzato pubblico a un nome di dominio personalizzato privato.
- Se disponi di un endpoint VPC che utilizzi per accedere a un nome di dominio personalizzato pubblico, non utilizzarlo per creare un'associazione di accesso al nome di dominio con un nome di dominio personalizzato privato.

Differenze tra nomi di dominio personalizzati privati e nomi di dominio personalizzati pubblici

Di seguito sono descritte le differenze tra i nomi di dominio personalizzati privati e pubblici:

- I nomi di dominio personalizzati privati non devono essere necessariamente univoci tra più account.
- Un nome di dominio privato ha un ID del nome di dominio. Questo ID identifica in modo univoco un nome di dominio personalizzato privato e non viene generato per i nomi di dominio personalizzati pubblici.
- Quando utilizzi AWS CLI per aggiornare o eliminare il tuo nome di dominio personalizzato privato, devi fornire l'ID del nome di dominio. Se il nome di dominio personalizzato privato è `example.com` e il nome di dominio personalizzato pubblico è `example.com` e non si fornisce l'ID del nome di dominio, Gateway API modificherà o eliminerà il nome di dominio personalizzato pubblico.

Passaggi successivi per nomi di dominio personalizzati per la API private

Per informazioni sulle attività di un provider e di un consumatore di API, consulta the section called “Provider di API e consumatori di API”.

Per istruzioni sulla creazione di un nome di dominio personalizzato privato che puoi richiamare con il tuo Account AWS, consultathe section called “Tutorial: creazione e invocazione di un nome di dominio personalizzato per API private”.

Per istruzioni su come fornire un altro Account AWS accesso al tuo nome di dominio personalizzato privato, consultathe section called “Provider API: condividi il tuo nome di dominio personalizzato privato utilizzando AWS RAM”. Per istruzioni su come associare il tuo endpoint VPC a un nome di dominio personalizzato privato in un altro Account AWS, consulta. the section called “Consumatore di API: associazione dell'endpoint VPC a un nome di dominio personalizzato privato condiviso”

Attività dei provider di API e dei consumatori di API per i nomi di dominio personalizzati per le API private

Quando crei un nome di dominio personalizzato privato, sei un provider di API. Quando invochi un nome di dominio personalizzato privato, sei un consumatore di API. Puoi utilizzare un nome di dominio privato personalizzato dal tuo Account AWS o da un altro. Account AWS

La sezione seguente spiega le attività che il provider e il consumatore di API devono eseguire per utilizzare un nome di dominio personalizzato privato. Se desideri richiamare un nome di dominio personalizzato privato per conto tuo Account AWS, sei sia il fornitore dell'API che il consumatore dell'API. Se desideri richiamare un dominio personalizzato privato in un altro Account AWS, a seconda della relazione di fiducia tra il fornitore di API e il consumatore dell'API AWS Organizations, AWS RAM potresti completare alcune attività per te.

Attività di un provider di API

I provider di API creano API private e le mappano a nomi di dominio personalizzati.

I provider di API gestiscono due policy delle risorse per proteggere i propri nomi di dominio personalizzati privati. La prima policy riguarda il servizio `execute-api` e controlla quali endpoint VPC possono invocare il nome di dominio personalizzato privato. Nella configurazione del nome di dominio personalizzato privato è denominata `policy`.

La seconda policy riguarda il servizio Amazon API Gateway Management e controlla quali endpoint VPC di altri endpoint Account AWS possono formare un'associazione di accesso al nome di dominio

con il tuo nome di dominio privato personalizzato. Un endpoint VPC deve formare un'associazione di accesso al nome di dominio con un nome di dominio personalizzato privato per invocarlo. Nella configurazione del nome di dominio personalizzato privato è denominata `managementPolicy`. Puoi utilizzare AWS RAM o API Gateway per aggiornare questa policy. Se non prevedi di consentire agli endpoint VPC di altri di Account AWS richiamare il tuo nome di dominio personalizzato, non modifichi il. `managementPolicy`

Se sei un provider di API, devi eseguire le seguenti operazioni:

1. Crea un'API privata.
2. Aggiorna la `policy` dell'API privata per concedere all'endpoint VPC l'accesso all'API privata.
3. Crea un nome di dominio personalizzato privato.
4. Aggiorna la `policy` del nome di dominio personalizzato privato per concedere all'endpoint VPC l'accesso al nome di dominio personalizzato privato.
5. Crea una mappatura percorso di base o una regola di routing per inviare il traffico dall'API privata al nome di dominio personalizzato privato. Per ulteriori informazioni, consulta the section called "Invia traffico al tuo APIs tramite il tuo nome di dominio personalizzato in API Gateway".

Se desideri consentire agli utenti di API di altri utenti di accedere Account AWS al tuo nome di dominio personalizzato privato, procedi come segue:

1. Aggiorna la `managementPolicy` del nome di dominio personalizzato privato per consentire ai consumatori di API in altri account di associare i loro endpoint VPC al tuo nome di dominio personalizzato privato. Puoi farlo utilizzando i seguenti metodi:

   AWS RAM

   Con AWS RAM, se il provider e il consumatore dell'API fanno parte della stessa organizzazione che utilizza AWS Organizations, la condivisione delle risorse tra fornitore e consumatore viene accettata automaticamente. In caso contrario, è necessario attendere che il consumatore di API accetti la condivisione delle risorse. Ti consigliamo di AWS RAM utilizzarlo per condividere il tuo nome di dominio personalizzato privato.

   Gateway API

   Con API Gateway, AWS CLI è supportato solo il. Devi aggiornare il nome di dominio personalizzato privato utilizzando un'operazione di patch e fornire il documento di policy per `managementPolicy`.

2. Aggiorna la `policy` del tuo nome di dominio personalizzato privato e di tutte le API private mappate a esso per concedere l'accesso all'endpoint VPC del consumatore di API.

Per istruzioni su come fornire la tua API a un altro Account AWS, consultathe section called “Provider API: condividi il tuo nome di dominio personalizzato privato utilizzando AWS RAM”.

Attività di un consumatore di API

I consumatori di API associano i propri endpoint VPC a un ARN di nome di dominio per invocare un nome di dominio personalizzato privato. I consumatori di API non devono necessariamente creare un'API di Gateway API.

Se sei un consumatore di API, esegui le seguenti operazioni:

1. Crea un endpoint VPC con DNS privato abilitato in Amazon VPC.
2. (Facoltativo, se AWS RAM utilizzato) Accetta una condivisione di risorse di dominio personalizzata privata AWS RAM entro 12 ore dalla condivisione delle risorse. Se fai parte della stessa organizzazione del provider di API, la condivisione di risorse viene accettata automaticamente.
3. Ottieni l'ARN del nome di dominio personalizzato privato. Poiché l'URL del nome di dominio personalizzato privato non è univoco, devi utilizzare l'ARN del nome di dominio personalizzato privato per creare l'associazione di accesso al nome di dominio tra l'endpoint VPC e il nome di dominio personalizzato privato. Puoi utilizzarlo AWS RAM per recuperare l'ARN del nome di dominio personalizzato privato.
4. Associa l'ARN del dominio personalizzato privato all'endpoint VPC in Gateway API. In questo modo viene creata una connessione sicura tra l'endpoint VPC e il nome di dominio personalizzato privato. Il traffico non esce dalla rete Amazon.
5. Attendi che il provider di API conceda all'endpoint VPC l'accesso al nome di dominio personalizzato privato e a qualsiasi API privata mappata al nome di dominio personalizzato privato. Se sei sia il provider che il consumatore di API, sei tu a concedere l'accesso per l'invocazione all'endpoint VPC.
6. Crea una zona ospitata privata di Route 53 e un record Route 53 per risolvere il nome di dominio personalizzato privato in Route 53.

Per istruzioni su come utilizzare un'API in un'altra Account AWS, consulta. the section called “Consumatore di API: associazione dell'endpoint VPC a un nome di dominio personalizzato privato condiviso”

Tutorial: creazione e invocazione di un nome di dominio personalizzato per API private

In questo tutorial creerai un nome di dominio personalizzato privato da invocare in un VPC nel tuo account. A questo scopo, sei il provider e il consumatore di API. Per completare questo tutorial devi disporre di un'API privata e di un endpoint VPC. Se disponi di un endpoint VPC che utilizzi per accedere a un nome di dominio personalizzato pubblico, non utilizzarlo per questo tutorial o per creare associazioni di accesso a nomi di dominio.

Passaggio 1: creare un nome di dominio personalizzato privato

Puoi creare un nome di dominio personalizzato privato specificando il nome di dominio, il certificato ACM e la policy per il servizio `execute-api` per controllare quali endpoint VPC possono invocarlo.

Console di gestione AWS

Per creare un nome di dominio personalizzato privato

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel pannello di navigazione principale, scegli Nomi di dominio personalizzati.
3. Scegli Aggiungi nome di dominio.
4. In Domain name (Nome di dominio), immettere un nome di dominio.

   Il certificato ACM deve coprire questo nome di dominio, ma non è necessario che il nome di dominio sia univoco.
5. Seleziona Privato.
6. Per Modalità di routing, scegli Solo mappature API.
7. Per Certificato ACM, seleziona un certificato.
8. Scegli Aggiungi nome di dominio.

Gateway API effettua il provisioning di un nome di dominio con una policy `deny` per tutte le risorse. Questa è la policy delle risorse per il servizio `execute-api`. Devi aggiornare questa policy delle risorse per concedere l'accesso agli endpoint VPC per invocare il tuo nome di dominio personalizzato privato.

Per aggiornare la policy delle risorse

1. Scegli la scheda Policy delle risorse, quindi scegli Modifica policy delle risorse.

2. Inserisci la policy delle risorse seguente nell'editor di codice. Sostituisci l'endpoint VPC *vpce-abcd1234efg* con il tuo ID endpoint VPC.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ],
            "Condition" : {
                "StringNotEquals": {
                    "aws:SourceVpce": "vpce-abcd1234"
                }
            }
        }
    ]
}
```

3. Scegli Save changes (Salva modifiche).

AWS CLI

Quando crei un nome di dominio personalizzato privato utilizzando il AWS CLI, fornisci una politica delle risorse affinché il `execute-api` servizio conceda l'accesso agli endpoint VPC per richiamare il tuo nome di dominio personalizzato privato, utilizzando il parametro. `--policy file://policy.json` È possibile modificare questa policy in un secondo momento.

In questo esempio si collega la seguente policy di risorse come `policy` caricando i parametri da un file. Copia e salva questo file come `policy.json`. Questa policy consente solo il

traffico in entrata verso un nome di dominio personalizzato privato dall'endpoint VPC: *vpce-abcd1234efg*

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ],
            "Condition" : {
                "StringNotEquals": {
                    "aws:SourceVpce": "vpce-abcd1234"
                }
            }
        }
    ]
}
```

Il seguente comando create-domain-name crea un nome di dominio personalizzato privato:

```
aws apigateway create-domain-name \
    --domain-name 'private.example.com' \
    --certificate-arn 'arn:aws:acm:us-west-2:111122223333:certificate/
a1b2c3d4-5678-90ab-cdef' \
    --security-policy 'TLS_1_2' \
    --endpoint-configuration '{"types":["PRIVATE"]}' \
    --policy file://policy.json
```

L'output sarà simile al seguente:

```
{
    "domainName": "private.example.com",
    "domainNameId": "abcd1234",
    "domainNameArn": "arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234",
    "certificateArn": "arn:aws:acm:us-west-2:111122223333:certificate/
a1b2c3d4-5678-90ab-cdef",
    "certificateUploadDate": "2024-09-10T10:31:20-07:00",
    "endpointConfiguration": {
        "types": [
            "PRIVATE"
        ]
    },
    "domainNameStatus": "AVAILABLE",
    "securityPolicy": "TLS_1_2",
    "routingMode" : "API_MAPPING_ONLY",
    "policy": "..."
}
```

Passaggio 2: creare una mappatura del percorso di base per mappare l'API privata al nome di dominio personalizzato privato

Dopo aver creato il nome di dominio personalizzato privato, mappa un'API privata a tale nome. Con una mappatura del percorso di base, un'API è accessibile tramite la combinazione di nome di dominio personalizzato privato e percorso di base associato. È consigliabile usare un singolo nome di dominio personalizzato privato come nome host di più API.

Tutti i provider di API devono creare una mappatura del percorso di base, anche se non hai intenzione di invocare la tua API. Inoltre, devi concedere l'accesso agli endpoint VPC per invocare qualsiasi API privata mappata al tuo nome di dominio personalizzato privato.

Console di gestione AWS

Per creare una mappatura del percorso di base

1. Accedi alla console API Gateway all'indirizzo. https://console.aws.amazon.com/apigateway
2. Nel pannello di navigazione principale, scegli Nomi di dominio personalizzati.
3. Scegli un nome di dominio personalizzato privato.
4. Nella scheda Mappature API scegli Configura mappature.

5. Scegliere Add new mapping (Aggiungi nuova mappatura).
6. Immettere un'API, uno Stage (Fase)e, facoltativamente, un Path (Percorso).
7. Selezionare Salva.

AWS CLI

Il seguente comando create-base-path-mapping crea una mappatura tra un'API privata e un nome di dominio personalizzato privato:

```
aws apigateway create-base-path-mapping \
    --domain-name-id abcd1234 \
    --domain-name 'private.example.com' \
    --rest-api-id a1b2c3 \
    --stage prod \
    --base-path v1
```

L'output sarà simile al seguente.

```
{
    "basePath": "v1",
    "restApiId": "a1b2c3",
    "stage": "prod"
}
```

Per una maggiore flessibilità su come instradare il traffico alle API, è possibile modificare la modalità di routing in `ROUTING_RULE_ONLY` o `ROUTING_RULE_THEN_API_MAPPING` e creare una regola di routing. Per ulteriori informazioni, consulta the section called “Invia traffico al tuo APIs tramite il tuo nome di dominio personalizzato in API Gateway”.

> **Note**
>
> Se desideri che altri Account AWS invochino il tuo nome di dominio personalizzato privato, dopo aver completato questo tutorial, segui i passaggi indicatithe section called “Provider API: condividi il tuo nome di dominio personalizzato privato utilizzando AWS RAM”.

Passaggio 3: creare un'associazione di accesso al nome di dominio tra il nome di dominio personalizzato e un endpoint VPC

A questo punto, creerai un'associazione di accesso al nome di dominio tra il nome di dominio personalizzato privato e l'endpoint VPC. L'endpoint VPC utilizza l'associazione di accesso al nome di dominio per invocare il nome di dominio personalizzato privato mentre è isolato dalla rete Internet pubblica.

Console di gestione AWS

Per creare un'associazione di accesso al nome di dominio

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel pannello di navigazione principale, scegli Nomi di dominio personalizzati.
3. Scegli un nome di dominio personalizzato privato.
4. Nella scheda Condivisione delle risorse, per le Associazioni di accesso ai nomi di dominio, scegli Crea associazione di accesso al nome di dominio.
5. In ARN del nome di dominio digita il tuo nome di dominio.
6. In ID endpoint VPC seleziona l'ID endpoint VPC a cui hai fornito l'accesso nel passaggio 1.
7. Scegli Associazione di accesso al nome di dominio.

Puoi anche creare l'associazione di accesso al nome di dominio utilizzando la pagina Associazioni di accesso ai nomi di dominio della console.

AWS CLI

Il seguente comando `create-domain-name-access-association` crea un'associazione di accesso al nome di dominio tra il nome di dominio personalizzato privato e l'endpoint VPC.

```
aws apigateway create-domain-name-access-association \
    --domain-name-arn arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234 \
    --access-association-source vpce-abcd1234efg \
    --access-association-source-type VPCE \
    --region us-west-2
```

L'output sarà simile al seguente.

```
{
```

```
    "domainNameAccessAssociationARN": "arn:aws:apigateway:us-west-2:111122223333:/
domainnameaccessassociations/domainname/private.example.com+abcd1234/vpcesource/
vpce-abcd1234efg",
    "accessAssociationSource": "vpce-abcd1234efg",
    "accessAssociationSourceType": "VPCE",
    "domainNameARN" : "arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234"
}
```

Affinché l'associazione di accesso al nome di dominio sia pronta sono necessari circa 15 minuti. Durante l'attesa, puoi eseguire i passaggi seguenti.

Passaggio 4: creare una zona ospitata Route 53

Dopo aver aggiornato la policy delle risorse e associato il nome di dominio personalizzato privato all'endpoint VPC, crea una zona ospitata privata in Route 53 per risolvere il nome di dominio personalizzato. Una zona ospitata è un container che contiene informazioni su come instradare il traffico per un dominio all'interno di uno o più VPC senza esporre le risorse su Internet. Per ulteriori informazioni, consulta Utilizzo delle zone ospitate private.

Console di gestione AWS

Per utilizzare il Console di gestione AWS, consulta Creazione di una zona ospitata privata nella Amazon Route 53 Developer Guide.

Per Nome, usa il tuo nome di dominio personalizzato privato. Per ID VPC, usa il VPC contenente l'endpoint VPC utilizzato nei passaggi precedenti.

AWS CLI

Il seguente comando create-hosted-zone crea una zona ospitata privata:

```
aws route53 create-hosted-zone --name private.example.com \
    --caller-reference 2014-04-01-18:47 \
    --hosted-zone-config Comment="command-line version",PrivateZone=true \
    --vpc VPCRegion=us-west-2,VPCId=vpc-abcd1234
```

L'output contiene l'ID della zona ospitata. L'ID della zona ospitata viene utilizzato nei passaggi seguenti.

Passaggio 5: creare un record DNS di Route 53

Dopo aver creato la zona ospitata, devi creare un record per risolvere il tuo nome di dominio personalizzato privato. Userai l'ID della zona ospitata che hai creato nel passaggio precedente. In questo esempio viene creato un tipo di record A. Se utilizzi IPv6 per il tuo endpoint VPC, crea un tipo di record AAAA. Se utilizzi dualstack per il tuo endpoint VPC, crea un tipo di record sia AAAA che A.

Console di gestione AWS

Per utilizzare il Console di gestione AWS, consulta Routing del traffico verso un'API Amazon API Gateway utilizzando il tuo nome di dominio.

Usa Creazione rapida e attiva Alias. Per l'endpoint, utilizza il nome DNS dell'endpoint VPC.

AWS CLI

Per configurare i record DNS in modo da mappare il nome di dominio personalizzato privato al relativo nome host dell'ID della zona ospitata, devi creare un file JSON contenente la configurazione per configurare un record DNS per il nome di dominio privato.

Il seguente `setup-dns-record.json` mostra come creare un record `A` DNS per mappare un nome di dominio personalizzato privato al relativo nome host privato. Fornisci il `DNSName` del tuo ID DNS VPC e l'ID della zona ospitata che hai creato nel passaggio precedente.

```
{
  "Changes": [
    {
      "Action": "UPSERT",
      "ResourceRecordSet": {
        "Name": "private.example.com",
        "Type": "A",
        "AliasTarget": {
          "DNSName": "vpce-abcd1234.execute-api.us-west-2.vpce.amazonaws.com",
          "HostedZoneId": "Z2OJLYMUO9EFXC",
          "EvaluateTargetHealth": false
        }
      }
    }
  ]
}
```

Il seguente comando change-resource-record-sets crea un record DNS per il nome di dominio personalizzato privato:

```
aws route53 change-resource-record-sets \
    --hosted-zone-id ZABCDEFG1234 \
    --change-batch file://path/to/your/setup-dns-record.json
```

Sostituisci `hosted-zone-id` con l'ID della zona ospitata di Route 53 del record DNS impostato nel tuo account. Il valore del parametro `change-batch` punta a un file JSON.

Se non intendi invocare il tuo nome di dominio personalizzato privato, dopo aver verificato che il nome di dominio personalizzato privato funzioni, puoi eliminare queste risorse.

Passaggio 6: invocare il nome di dominio personalizzato privato

Ora puoi invocare il tuo nome di dominio personalizzato privato nel tuo Account AWS. Nel tuo VPC, usa il seguente comando curl per accedere al tuo nome di dominio personalizzato privato.

```
curl https://private.example.com/v1
```

Per ulteriori informazioni su altri modi per invocare la tua API privata, consulta the section called “Invocazione di un'API privata utilizzando un nome di dominio personalizzato”.

Passaggio 7: eseguire l'eliminazione

Per evitare costi inutili, elimina l'associazione tra l'endpoint VPC e il nome di dominio personalizzato privato, quindi elimina il nome di dominio personalizzato privato.

Console di gestione AWS

Per eliminare l'associazione di accesso al nome di dominio

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel pannello di navigazione principale, scegli Associazioni di accesso ai nomi di dominio.
3. Seleziona l'associazione di accesso al nome di dominio, quindi scegli Elimina.
4. Conferma la tua scelta, quindi scegli Elimina.

Dopo aver eliminato l'associazione di accesso al nome di dominio, puoi eliminare il tuo nome di dominio personalizzato privato.

Per eliminare il nome di dominio personalizzato privato

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel pannello di navigazione principale, scegli Nomi di dominio personalizzati.
3. Scegli il tuo nome di dominio personalizzato privato.
4. Scegliere Elimina.
5. Conferma la tua scelta, quindi scegli Elimina.

Se necessario, puoi anche eliminare il tuo endpoint VPC. Per ulteriori informazioni, consulta Eliminazione di un endpoint dell'interfaccia.

AWS CLI

Per eliminare

1. Il seguente comando `delete-access-association` elimina l'associazione di accesso al nome di dominio:

```
aws apigateway delete-domain-name-access-association \
    --domain-name-access-association-arn 'arn:aws:apigateway:us-
west-2:111122223333:/domainnameaccessassociations/domainname/private.example.com
+abcd1234/vpcesource/vpce-abcd1234efg' \
    --region us-west-2
```

2. Il seguente comando `delete-domain-name` elimina il nome di dominio personalizzato privato. Questo comando rimuove anche tutte le mappature dei percorsi di base.

```
aws apigateway delete-domain-name \
    --domain-name test.private.com \
    --domain-name-id abcd1234
```

Se necessario, puoi anche eliminare il tuo endpoint VPC. Per ulteriori informazioni, consulta Eliminazione di un endpoint dell'interfaccia.

Best practice

Di seguito sono riportate le best practice consigliate per la creazione di un nome di dominio personalizzato privato:

- Si utilizza la mappatura percorso di base o le regole di routing per inviare il traffico da un nome di dominio privato personalizzato a più API private.
- Quando un endpoint VPC non ha più bisogno di accedere a un nome di dominio personalizzato privato, elimina l'associazione. Inoltre, rimuovi l'endpoint VPC da `execute-api` per il servizio `policy` per il dominio personalizzato privato.
- Configura almeno due zone di disponibilità per endpoint VPC.
- Disabilita l'endpoint predefinito. È consigliabile disabilitare l'endpoint predefinito per consentire ai consumatori dell'API di chiamare l'API solo dal nome di dominio personalizzato. Per ulteriori informazioni, consulta the section called “Disabilitazione dell'endpoint predefinito”.
- Ti consigliamo di eseguire il provisioning di una zona ospitata privata di Route 53 e di un A-type record quando configuri il tuo nome di dominio personalizzato privato. Se non hai intenzione di invocare il tuo nome di dominio personalizzato privato, puoi eliminare queste risorse in un secondo momento.

Utilizzo di nomi di dominio personalizzati privati in più account

Questa sezione descrive come utilizzare nomi di dominio personalizzati privati in più account. Puoi fornire un nome di dominio personalizzato privato a un altro Account AWS e utilizzarne un altro Account AWS per richiamare un nome di dominio personalizzato privato.

Puoi condividere il tuo nome di dominio personalizzato privato Account AWS con un altro utilizzando AWS Resource Access Manager il nostro API Gateway. AWS Resource Access Manager (AWS RAM) ti aiuta a condividere in modo sicuro le tue risorse tra Account AWS e all'interno dell'organizzazione o delle unità organizzative (OU). Per ulteriori informazioni, vedi Che cos'è AWS Resource Access Manager.

Per istruzioni su come condividere un nome di dominio personalizzato privato Account AWS con un altro utente AWS RAM, consultathe section called “Provider API: condividi il tuo nome di dominio personalizzato privato utilizzando AWS RAM”.

Per istruzioni su come condividere un nome di dominio personalizzato privato con un altro utente Account AWS utilizzando API Gateway, consultathe section called “Provider API: condividi il tuo nome di dominio personalizzato privato utilizzando l'API Gateway AWS CLI”.

Per istruzioni su come utilizzare un nome di dominio personalizzato privato in un altro Account AWS, consultathe section called “Consumatore di API: associazione dell'endpoint VPC a un nome di dominio personalizzato privato condiviso”.

Best practice per l'uso di nomi di dominio personalizzati privati in più account

Di seguito sono riportate le best practice consigliate per l'utilizzo di nomi di dominio personalizzati privati in più account:

- AWS RAM Usalo per condividere i tuoi nomi di dominio personalizzati privati. Quando lo usi AWS RAM, puoi ridurre il sovraccarico operativo e non devi crearne uno `managementPolicy` per il servizio Amazon API Gateway Management.
- Usa il parametro `resource-owner` quando elenchi i tuoi nomi di dominio personalizzati privati o le associazioni di accesso ai nomi di dominio. Usa il parametro `resource-owner` per elencare solo le risorse che appartengono a te o ad altri Account AWS.

  L'esempio seguente mostra come ottenere tutte le associazioni di accesso ai nomi di dominio di tua proprietà:

  ```
  aws apigateway get-domain-name-access-associations --resource-owner SELF
  ```

  Usa `--resource-owner OTHER_ACCOUNTS` per elencare tutte le associazioni di accesso ai nomi di dominio che altri account hanno creato con il tuo nome di dominio personalizzato privato.

Provider API: condividi il tuo nome di dominio personalizzato privato utilizzando AWS RAM

Puoi fornire ai consumatori di API un altro Account AWS accesso al tuo nome di dominio personalizzato privato. In questa sezione, scopri come condividere il tuo nome di dominio personalizzato privato utilizzando AWS RAM e come controllare l'accesso al tuo nome di dominio personalizzato privato.

Considerazioni sulla condivisione del nome di dominio personalizzato privato

Le seguenti considerazioni potrebbero influire sul modo in cui fornisci l'accesso al tuo nome di dominio personalizzato privato tramite AWS RAM. Per sapere come condividere il tuo nome di dominio personalizzato privato senza utilizzarlo AWS RAM, consultathe section called “Provider API: condividi il tuo nome di dominio personalizzato privato utilizzando l'API Gateway AWS CLI”.

- I nomi di dominio personalizzati privati sono condivisi a Regione AWS livello. Il nome di dominio personalizzato privato e l'endpoint VPC devono trovarsi nello stesso Regione AWS.
- Puoi utilizzare una condivisione di risorse con più principali e, dopo aver creato la condivisione di risorse, aggiungervi altri principali. È consigliabile riutilizzare la condivisione di risorse, quando possibile.

- Devi sempre concedere l'accesso all'endpoint VPC del consumatore di API per invocare il nome di dominio personalizzato privato e le API private mappate a esso.
- Se il consumatore e il fornitore dell'API fanno parte della stessa organizzazione che utilizza AWS Organizations, la condivisione delle risorse viene accettata automaticamente. È comunque necessario creare la condivisione di risorse tramite AWS RAM.
- Se il consumatore e il fornitore di API fanno parte della stessa organizzazione che utilizza AWS Organizations e la condivisione delle risorse all'interno dell'organizzazione è abilitata, a tutti i principali dell'organizzazione con cui si condivide viene automaticamente concesso l'accesso alle condivisioni di risorse. Non è necessario un invito e puoi evitare la condivisione di risorse.
- Se il consumatore di API non accetta la condivisione di risorse entro 12 ore, il provider di API deve condividere nuovamente la risorsa.
- Dopo aver creato la condivisione delle risorse, AWS RAM aggiorna il `managementPolicy` servizio Amazon API Gateway Management per il tuo nome di dominio personalizzato privato per impedire l'accesso ai principali senza accesso esplicito`allow`. Per ulteriori informazioni, consulta Determining whether a request is allowed or denied within an account nella Guida per l'utente di IAM.

  La `managementPolicy` aggiornata sarà simile alla seguente:

  JSON

```
{
    "Version":"2012-10-17",
    "Id": "abcd1234-1234-abcd-abcd-1234abcdefg",
    "Statement": [
        {
            "Sid":
 "APIGatewayPrivateDomainNameManagementPolicyDefaultPermission-org",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "apigateway:CreateAccessAssociation",
            "Resource": "arn:aws:apigateway:us-west-2:111122223333:/
domainnames/private.example.com+abcd1234",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": "o-1234abcd"
                },
                "StringNotEquals": {
                    "aws:PrincipalAccount": "111122223333"
                }
```

```
            }
        }
    ]
}
```

AWS RAM ha impedito ai principali senza accesso esplicito di creare associazioni di `allow` accesso con il tuo nome di dominio personalizzato privato, aggiungendo quanto segue:

```
"StringNotEquals": {
    "aws:PrincipalAccount": "111122223333"
}
```

Puoi comunque utilizzare il principale indicato nella Account AWS persona che ha creato il nome di dominio personalizzato privato per creare associazioni di accesso ai nomi di dominio.

Consentire ad altri account di creare associazioni di accesso ai nomi di dominio con il tuo nome di dominio personalizzato privato

Innanzitutto, concedi l'accesso a un altro utente Account AWS per creare associazioni di accesso ai nomi di dominio con il tuo nome di dominio personalizzato privato.

Console di gestione AWS

Per utilizzare il Console di gestione AWS, consulta Creazione di una condivisione di risorse AWS RAM nella Guida AWS RAM per l'utente.

In Seleziona il tipo di risorsa, scegli Domini personalizzati privati Gateway API.

AWS CLI

Il seguente comando create-resource-share crea una condivisione di risorse per il tuo nome di dominio personalizzato privato. Il completamento dell'associazione tra la risorsa e il principale può richiedere alcuni minuti. Per i principali, fornisci un ID account o un ID organizzazione, ad esempio `arn:aws:organizations::123456789012:organization/o-1234abcd`. Puoi fornire più principali per la condivisione di risorse.

```
aws ram create-resource-share \
    --region us-west-2 \
    --name privateCustomDomain-resource-share \
    --permission-arns arn:aws:ram::aws:permission/
APIGatewayPrivateDomainNameManagementPolicyDefaultPermission \
```

```
    --resource-arns arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234 \
    --principals 222222222222
```

Dopo aver fornito l'accesso a un altro Account AWS, i consumatori di API di quell'account devono creare un'associazione di accesso al nome di dominio tra il loro endpoint VPC e il tuo nome di dominio personalizzato privato. Non puoi creare l'associazione di accesso al nome di dominio al loro posto. Per ulteriori informazioni, consulta the section called “Associazione dell'endpoint VPC a un nome di dominio personalizzato privato condiviso”.

Consentire ad altri account di invocare il nome di dominio personalizzato privato

Successivamente, concedi l'accesso all'endpoint VPC del consumatore di API per invocare il tuo nome di dominio personalizzato privato e le API private mappate ad esso.

Console di gestione AWS

Per consentire agli endpoint VPC in altri account di invocare il nome di dominio personalizzato privato

1. Accedi alla console API Gateway all'indirizzo. https://console.aws.amazon.com/apigateway
2. Nel pannello di navigazione principale, scegli Nomi di dominio personalizzati.
3. Scegli il nome di dominio personalizzato privato che hai condiviso con altri Account AWS.
4. Nella scheda Policy delle risorse, scegli Modifica policy delle risorse.
5. Aggiungi l'ID dell'endpoint VPC del consumatore di API alla tua policy delle risorse.

   Puoi trovare l'ID dell'endpoint VPC del consumatore di API nella sezione Associazioni di accesso ai nomi di dominio della scheda Condivisione delle risorse nella pagina Dettagli del dominio del tuo nome di dominio personalizzato privato.
6. Scegli Save changes (Salva modifiche).

AWS CLI

La seguente `policy` per il servizio `execute-api` consente il traffico in entrata verso un nome di dominio personalizzato privato dagli endpoint VPC *`vpce-abcd1234efg`* e *`vpce-xyz000abc`*.

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ],
            "Condition" : {
                "StringNotEquals": {
                    "aws:SourceVpce": [
                    "vpce-abcd1234",
                    "vpce-xyzz0000"
                    ]
                }
            }
        }
    ]
}
```

Il seguente comando update-domain-name utilizza un'operazione di patch per aggiornare la `policy` per un nome di dominio personalizzato privato:

```
aws apigateway update-domain-name
    --domain-name private.example.com \
    --domain-name-id abcd1234 \
    --patch-operations op=replace,path=/policy,value='"{\"Version\": \"2012-10-17\",
       \"Statement\": [{\"Effect\": \"Allow\",\"Principal\": \"*\",\"Action\":
 \"execute-api:Invoke\",\"Resource\":[\"execute-api:/*\"]},{\"Effect\": \"Deny\",
\"Principal\": \"*\",\"Action\": \"execute-api:Invoke\",\"Resource\":[\"execute-
api:/*\"],\"Condition\":{\"StringNotEquals\":[\"vpce-abcd1234efg\", \"vpce-xyz000abc
\"]}}}]}"
```

Provider di API: interrompi la condivisione di un nome di dominio personalizzato privato utilizzando AWS RAM

Per interrompere la condivisione del nome di dominio personalizzato privato, devi innanzitutto impedire al consumatore di API di creare altre associazioni di accesso ai nomi di dominio dissociando la condivisione delle risorse. Quindi, devi rifiutare l'associazione di accesso al nome di dominio e rimuovere l'endpoint VPC del consumatore di API dalla tua `policy` per il servizio `execute-api`. Il consumatore di API può quindi eliminare la propria associazione di accesso al nome di dominio.

Interruzione della condivisione del nome di dominio personalizzato privato

Innanzitutto, interrompi l'utilizzo AWS RAM delle risorse condivise.

Console di gestione AWS

Per utilizzare il Console di gestione AWS, vedi Aggiornare una risorsa condivisa in AWS RAM.

AWS CLI

Il seguente comando disassociate-resource-share dissocia una condivisione di risorse per il tuo nome di dominio personalizzato privato.

```
aws ram disassociate-resource-share \
    --region us-west-2 \
    --resource-arns arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234 \
    --principals 222222222222
```

Rifiuto dell'associazione di accesso al nome di dominio

Dopo aver interrotto la condivisione dell'utilizzo delle risorse AWS RAM, si rifiuta l'associazione di accesso al nome di dominio tra un endpoint VPC in un altro account e il nome di dominio personalizzato privato.

> **Note**
>
> Non puoi rifiutare un'associazione di accesso a un nome di dominio nel tuo account. Per interrompere la condivisione di risorse, elimina l'associazione di accesso al nome di dominio. Per ulteriori informazioni, consulta la sezione su come eliminare un'associazione di accesso al nome di dominio.

Quando rifiuti un'associazione di accesso al nome di dominio con un endpoint VPC, se un consumatore di API tenta di chiamare il tuo nome di dominio personalizzato privato, Gateway API rifiuta la chiamata e restituisce un codice di stato `403`.

Console di gestione AWS

Per rifiutare un'associazione di accesso al nome di dominio

1. Accedi alla console API Gateway all'indirizzo. https://console.aws.amazon.com/apigateway
2. Nel pannello di navigazione principale, scegli Nomi di dominio personalizzati.
3. Scegli il nome di dominio personalizzato privato che hai condiviso con altri Account AWS.
4. Nella sezione Condivisione delle risorse, scegli l'associazione di accesso al nome di dominio che desideri rifiutare.
5. Scegli Rifiuta l'associazione.
6. Conferma la tua scelta, quindi scegli Rifiuta.

AWS CLI

Il seguente comando `reject-domain-name-access-association` rifiuta l'associazione di accesso al nome di dominio tra l'endpoint VPC e il nome di dominio personalizzato privato.

```
aws apigateway reject-domain-name-access-association \
    --domain-name-access-association-arn arn:aws:apigateway:us-west-2:444455556666:/
domainnameaccessassociations/domainname/private.example.com+abcd1234/vpcesource/
vpce-abcd1234efg \
    --domain-name-arn arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234
```

Negazione dell'accesso al provider di API per invocare il nome di dominio personalizzato privato

Dopo avere rifiutato l'associazione di accesso al nome di dominio, rimuovi l'endpoint VPC del consumatore di API dalla tua `policy` per il servizio `execute-api`.

Console di gestione AWS

Per rimuovere l'endpoint VPC del consumatore di API dalla tua policy delle risorse

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.

2. Nel pannello di navigazione principale, scegli Nomi di dominio personalizzati.
3. Scegli il nome di dominio personalizzato privato che hai condiviso con altri Account AWS.
4. Nella scheda Policy delle risorse, seleziona Modifica.
5. Rimuovi l'endpoint VPC dalla policy.
6. Scegli Save changes (Salva modifiche).

AWS CLI

Il seguente comando update-domain-name utilizza un'operazione di patch per aggiornare la `execute-api` per il servizio `policy` per un nome di dominio personalizzato privato. Questa nuova `policy` rimuove un ulteriore ID endpoint VPC aggiunto in the section called “Consentire ad altri account di invocare il nome di dominio personalizzato privato”:

```
aws apigateway update-domain-name
    --domain-name private.example.com \
    --domain-name-id abcd1234 \
    --patch-operations op=replace,path=/policy,value='"{\"Version\": \"2012-10-17\",
       \"Statement\": [{\"Effect\": \"Allow\",\"Principal\": \"*\",\"Action\":
 \"execute-api:Invoke\",\"Resource\":[\"execute-api:/*\"]},{\"Effect\": \"Deny\",
\"Principal\": \"*\",\"Action\": \"execute-api:Invoke\",\"Resource\":[\"execute-
api:/*\"],\"Condition\":{\"StringNotEquals\":{\"aws:SourceVpce\": \"vpce-abcd1234efg
\"}}}]}"
```

Il consumatore di API deve quindi eliminare l'associazione di accesso al nome di dominio. Non puoi farlo tu al suo posto. Per ulteriori informazioni, consulta the section called “Consumatore di API: eliminazione dell'associazione di accesso al nome di dominio con un nome di dominio personalizzato privato”.

Provider API: condividi il tuo nome di dominio personalizzato privato utilizzando l'API Gateway AWS CLI

Puoi condividere un nome di dominio personalizzato privato utilizzando l'API Gateway AWS CLI, ma ti consigliamo di AWS RAM utilizzarlo per ridurre il sovraccarico operativo. Per istruzioni su come utilizzare per AWS RAM condividere il tuo nome di dominio personalizzato privato, consultathe section called “Provider API: condividi il tuo nome di dominio personalizzato privato utilizzando AWS RAM”.

Per condividere un nome di dominio personalizzato privato utilizzando l'API Gateway AWS CLI, concedi un altro Account AWS accesso per creare associazioni di accesso ai nomi di dominio e richiamare il tuo nome di dominio personalizzato privato. A tale scopo, è necessario aggiornare la `managementPolicy` del servizio di gestione di Gateway API e la `policy` per il servizio `execute-api` per il tuo nome di dominio personalizzato privato. Devi anche concedere l'accesso all'endpoint VPC del consumatore di API nella policy delle risorse per tutte le API private mappate al tuo nome di dominio personalizzato privato.

Il consumatore di API dovrà comunque creare un'associazione di accesso al nome di dominio tra il suo endpoint VPC e il tuo nome di dominio personalizzato privato. Non puoi farlo tu al suo posto.

Concedere l'accesso al nome di dominio personalizzato privato

Per concedere l'accesso al tuo nome di dominio personalizzato privato

1. Per aggiornare la `managementPolicy` del servizio di gestione di Gateway API, è necessario creare un file JSON contenente l'operazione di patch per l'aggiornamento della policy. Quanto segue `patch-managementPolicy.json` sostituisce quello corrente `managementPolicy` con un criterio di esempio che concede a Account AWS 111122223333 e 444455556666 l'accesso per creare associazioni di accesso ai nomi di dominio con il nome di dominio personalizzato privato. `private.example.com`

   ```
   [{
       "op": "replace",
       "path": "/managementPolicy",
       "value": "{\"Version\":\"2012-10-17\",       \"Statement\":[{\"Effect
   \":\"Allow\",\"Principal\":{\"AWS\":[\"arn:aws:iam::111122223333:root
   \", \"arn:aws:iam::444455556666:root\"]},\"Action\":
   \"apigateway:CreateAccessAssociation\",\"Resource\":\"arn:aws:apigateway:us-
   west-2:111122223333:/domainnames/private.example.com+abcd1234\"}]}"
   }]
   ```

   Il seguente comando update-domain-name aggiorna la `managementPolicy` tramite `patch-managementPolicy.json`.

   ```
   aws apigateway update-domain-name \
       --domain-name private.example.com \
       --domain-name-id abcd1234 \
       --patch-operations file://patch-managementPolicy.json
   ```

Una volta concesso l'accesso, devi notificare al consumatore di API che può formare l'associazione di accesso ai nomi di dominio. Se lo usi AWS RAM, AWS RAM eseguirà questo passaggio per te.

2. Per aggiornare la `policy` per il servizio `execute-api`, è necessario creare un file JSON contenente l'operazione di patch per l'aggiornamento della policy. Il seguente `patch-policy.json` sostituisce la `policy` corrente con una policy di esempio che consente a due endpoint VPC di invocare il nome di dominio personalizzato privato `private.example.com`.

```
[{
    "op": "replace",
    "path": "/policy",
    "value": "{\"Version\": \"2012-10-17\",       \"Statement\": [{\"Effect\":
 \"Allow\",\"Principal\": \"*\",\"Action\": \"execute-api:Invoke\",\"Resource\":
 \"arn:aws:execute-api:us-west-2:111122223333:/domainnames/private.example.com
+abcd1234\"},{\"Effect\": \"Deny\",\"Principal\": \"*\",\"Action\": \"execute-
api:Invoke\",\"Resource\": \"arn:aws:execute-api:us-west-2:111122223333:/
domainnames/private.example.com+abcd1234\",\"Condition\": {\"StringNotEquals\":
 {\"aws:SourceVpce\": [\"vpce-abcd1234\",\"vpce-xyzz0000\"]}}}]}"
}]
```

Il seguente comando update-domain-name aggiorna la `policy` tramite `patch-policy.json`.

```
aws apigateway update-domain-name \
    --domain-name private.example.com \
    --domain-name-id abcd1234 \
    --patch-operations file://patch-policy.json
```

Negare l'accesso al nome di dominio personalizzato privato

Per interrompere la condivisione del tuo nome di dominio personalizzato privato, devi rifiutare l'associazione di accesso al nome di dominio tra il tuo nome di dominio personalizzato privato e l'endpoint VPC del consumatore di API.

Per negare l'accesso al nome di dominio personalizzato privato

1. Il seguente comando `reject-domain-name-access-association` rifiuta l'associazione di accesso al nome di dominio.

```
aws apigateway reject-domain-name-access-association \
    --domain-name-access-association-arn arn:aws:apigateway:us-
west-2:444455556666:/domainnameaccessassociations/domainname/private.example.com
+abcd1234/vpcesource/vpce-abcd1234 \
    --domain-name-arn arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234
```

2. Modifica `patch-managementPolicy.json` per rimuovere l'accesso che consente all'account del provider di API di creare un'associazione di accesso al nome di dominio con il tuo nome di dominio personalizzato privato. Il seguente `patch-managementPolicy.json` rimuove un account dalla `managementPolicy`:

```
[{
     "op": "replace",
     "path": "/managementPolicy",
     "value": "{\"Version\":\"2012-10-17\",       \"Statement\":[{\"Effect\":
\"Allow\",\"Principal\":\"*\",\"Action\":\"apigateway:CreateAccessAssociation
\",\"Resource\":\"arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234\"}]}"
}]
```

   Il seguente comando update-domain-name aggiorna la `managementPolicy` tramite `patch-managementPolicy.json`.

```
aws apigateway update-domain-name \
    --domain-name private.example.com \
    --domain-name-id abcd1234 \
    --patch-operations file://patch-managementPolicy.json
```

3. Modifica `patch-policy.json` per rimuovere l'accesso che consente all'endpoint VPC del provider di API di invocare il tuo nome di dominio personalizzato privato. Il seguente `patch-policy.json` rimuove l'ID dell'endpoint VPC dalla `policy`:

```
[{
    "op": "replace",
    "path": "/policy",
    "value": "{\"Version\":\"2012-10-17\",       \"Statement\":[{\"Effect\":
\"Allow\",\"Principal\":\"*\",\"Action\":\"execute-api:Invoke\",\"Resource\":
\"arn:aws:execute-api:us-west-2:111122223333:/domainnames/private.example.com
+abcd1234\"},{\"Effect\":\"Deny\",\"Principal\":\"*\",\"Action\":\"execute-
```

```
api:Invoke\",\"Resource\":\"arn:aws:execute-api:us-west-2:111122223333:/
domainnames/private.example.com+abcd1234\",\"Condition\":{\"StringNotEquals\":
{\"aws:SourceVpce\":\"vpce-abcd1234\"}}}]}"
}]
```

Il seguente comando update-domain-name aggiorna la `policy` tramite `patch-policy.json`.

```
aws apigateway update-domain-name \
    --domain-name private.example.com \
    --domain-name-id abcd1234 \
    --patch-operations file://patch-policy.json
```

Policy di esempio utilizzate in questa procedura

Nella sezione seguente sono illustrate le policy di esempio utilizzate nella procedura precedente.

La seguente policy di esempio riguarda `managementPolicy` per il servizio di gestione di Gateway Amazon API. Questa politica concede a Account AWS 111122223333 e 444455556666 l'accesso per creare associazioni di accesso ai nomi di dominio con il nome di dominio personalizzato privato. `private.example.com`

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "111122223333",
                    "444455556666"
                ]
            },
            "Action": "apigateway:CreateAccessAssociation",
            "Resource": "arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+a1b2c3"
        }
    ]
}
```

La seguente policy di esempio riguarda la `policy` per il servizio `execute-api`. Questa policy concede l'accesso agli endpoint VPC `vpce-abcd1234` e `vpce-xyzz0000` per invocare il nome di dominio personalizzato privato.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "arn:aws:execute-api:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234"
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "arn:aws:execute-api:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234",
            "Condition": {
                "StringNotEquals": {
                    "aws:SourceVpce": [
                        "vpce-abcd1234",
                        "vpce-xyzz0000"
                    ]
                }
            }
        }
    ]
}
```

Consumatore di API: associazione dell'endpoint VPC a un nome di dominio personalizzato privato condiviso

La procedura seguente mostra come utilizzare un nome di dominio privato in un altro Account AWS. A seconda del rapporto di fiducia che avete con il provider di API, AWS RAM potrebbe completare alcune attività al posto vostro.

Quando ti trovi in un nome di dominio personalizzato diverso Account AWS da quello privato, puoi solo associare il tuo endpoint VPC a un nome di dominio personalizzato privato e richiamarlo. Non puoi visualizzare la `policy` o qualsiasi altro parametro del nome di dominio personalizzato privato.

Prerequisiti

I seguenti prerequisiti sono necessari per utilizzare un nome di dominio personalizzato privato in un altro: Account AWS

- Un VPC e un endpoint VPC per il servizio `execute-api`. Per il VPC, è necessario che `enableDnsHostnames` e `enableDnsSupport` siano impostati su `true`.
- È consigliabile configurare almeno due zone di disponibilità per endpoint VPC.

(Facoltativo) Accettare la condivisione di risorse del dominio personalizzato privato

Se il tuo provider AWS RAM di API ha creato una condivisione di risorse, hai 12 ore per accettarla. Se fai parte della stessa organizzazione che utilizza AWS Organizations come provider API, la condivisione viene accettata automaticamente. Se fai parte di un'organizzazione in cui è abilitata la condivisione automatica delle risorse, la risorsa viene condivisa automaticamente con te.

Console di gestione AWS

Per utilizzare il Console di gestione AWS, consulta Accettare e rifiutare gli inviti alla condivisione di risorse nella Guida per l'AWS RAM utente.

AWS CLI

Per trovare tutte le risorse condivise con te, usa il seguente comando get-resource-share-invitations:

```
aws ram get-resource-share-invitations \
    --region us-west-2
```

Utilizza l'ARN di condivisione risorse risultante per accettare l'invito alla condivisione di risorse. Il seguente comando accept-resource-share-invitation accetta la condivisione di risorse.

```
aws ram accept-resource-share-invitation \
    --resource-share-invitation-arn arn:aws:ram:us-west-2:123456789012:resource-
share-invitation/1e3477be-4a95-46b4-bbe0-c4001EXAMPLE \
    --region us-west-2
```

Associazione dell'endpoint VPC a un nome di dominio personalizzato privato condiviso

Poiché i nomi di dominio personalizzati privati non sono univoci, devi associare il tuo endpoint VPC all'ARN del nome di dominio personalizzato univoco. Dopo aver creato l'associazione di accesso al nome di dominio, possono essere necessari fino a 15 minuti prima che l'endpoint VPC invochi correttamente il nome di dominio personalizzato privato. Se disponi di un endpoint VPC che utilizzi per accedere a un nome di dominio personalizzato pubblico, non utilizzarlo per creare associazioni di accesso a nomi di dominio.

Console di gestione AWS

Per associare l'endpoint VPC a un nome di dominio personalizzato privato condiviso

1. Accedi alla console API Gateway all'indirizzo. https://console.aws.amazon.com/apigateway
2. Nel pannello di navigazione principale, scegli Associazioni di accesso ai nomi di dominio.
3. Scegli Crea associazione di accesso al nome di dominio.
4. Per ARN del nome di dominio, seleziona l'ARN del nome di dominio che il provider di API ha condiviso con te.

   L'ARN del nome di dominio potrebbe non essere visualizzato nell'elenco a discesa. Puoi utilizzare la AWS RAM console per visualizzare i nomi di dominio condivisi con te, quindi copiare l'ARN del nome di dominio e inserirlo in questo campo.
5. Per ID endpoint VPC, seleziona l'ID endpoint VPC con cui desideri creare l'associazione di accesso al nome di dominio.
6. Scegli Crea associazione di accesso al nome di dominio.

AWS CLI

Poiché i nomi di dominio personalizzati privati non sono univoci, devi associare il tuo endpoint VPC all'ARN del nome di dominio personalizzato univoco. Per trovare l'ARN del nome di dominio, utilizza uno dei comandi seguenti.

1. AWS RAM

   Il seguente comando list-resources elenca le risorse condivise con te. Il provider di API deve aver utilizzato AWS RAM per condividere con te il proprio dominio personalizzato privato per utilizzare questo comando.

```
aws ram list-resources \
    --resource-owner OTHER-ACCOUNTS \
    --region us-west-2
    --resource-type apigateway:Domainnames
```

Gateway API

Il `get-domain-names` comando seguente elenca tutti i nomi di dominio personalizzati privati di proprietà di altri con Account AWS cui è possibile formare associazioni di accesso ai nomi di dominio.

```
aws apigateway get-domain-names \
    --resource-owner OTHER_ACCOUNTS \
    --region us-west-2
```

2. Dopo aver recuperato l'ARN, utilizza Gateway API per creare l'associazione di accesso al nome di dominio tra l'endpoint VPC e un nome di dominio personalizzato privato condiviso. Utilizza il seguente comando `create-domain-name-access-association`:

```
aws apigateway create-domain-name-access-association \
    --access-association-source-type VPCE \
    --access-association-source 'vpce-1a2b3c4d5e6f1a2b3' \
    --domain-name-arn arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234"
```

L'output sarà simile al seguente.

```
{
    "domainNameAccessAssociationARN": "arn:aws:apigateway:us-
west-2:444455556666:/domainnameaccessassociations/domainname/private.example.com
+abcd1234/vpcesource/vpce-abcd1234efg",
    "accessAssociationSource": "vpce-1a2b3c4d5e6f1a2b3",
    "accessAssociationSourceType": "VPCE",
    "domainNameARN" : "arn:aws:apigateway:us-west-1:111122223333:/domainnames/
private.example.com+a1b2c3"
}
```

Dopo aver associato il tuo endpoint VPC al nome di dominio personalizzato privato, verifica che il tuo provider di API abbia aggiornato la policy del proprio nome di dominio personalizzato privato per

consentire al tuo endpoint VPC di invocare il suo nome di dominio. Per ulteriori informazioni, consulta the section called “Consentire ad altri account di invocare il nome di dominio personalizzato privato”.

Creazione di una zona ospitata Route 53

Per risolvere il nome di dominio personalizzato privato, devi creare una zona ospitata privata Route 53. Una zona ospitata è un container che contiene informazioni su come instradare il traffico per un dominio all'interno di uno o più VPC senza esporre le risorse su Internet. Per ulteriori informazioni, consulta Utilizzo delle zone ospitate private.

Console di gestione AWS

Per utilizzare il Console di gestione AWS, consulta Creazione di una zona ospitata privata nella Amazon Route 53 Developer Guide.

Per Nome, usa il nome di dominio personalizzato privato. Per ID VPC, usa il VPC contenente l'endpoint VPC che hai usato per l'associazione di accesso al nome di dominio.

AWS CLI

Il seguente comando create-hosted-zone crea una zona ospitata privata:

```
aws route53 create-hosted-zone --name private.example.com \
    --caller-reference 2014-04-01-18:47 \
    --hosted-zone-config Comment="command-line version",PrivateZone=true \
    --vpc VPCRegion=us-west-2,VPCId=vpc-abcd1234
```

L'output contiene l'ID della zona ospitata. L'ID della zona ospitata viene utilizzato nei passaggi seguenti.

Creazione di un record DNS di Route 53

Dopo aver creato la zona ospitata, devi creare un record per risolvere il dominio personalizzato privato. In questo esempio viene creato un tipo di record A. Se utilizzi IPv6 per il tuo endpoint VPC, crea un tipo di record AAAA. Se utilizzi dualstack per il tuo endpoint VPC, crea un tipo di record sia AAAA che A.

Console di gestione AWS

Per utilizzare il Console di gestione AWS, consulta Routing del traffico verso un'API Amazon API Gateway utilizzando il tuo nome di dominio.

Usa Creazione rapida e attiva Alias. Per l'endpoint, utilizza il nome DNS dell'endpoint VPC.

AWS CLI

Per configurare i record DNS in modo da mappare il nome di dominio personalizzato privato al relativo nome host dell'ID della zona ospitata, devi prima creare un file JSON contenente la configurazione per configurare un record DNS per il nome di dominio privato.

Il seguente `setup-dns-record.json` mostra come creare un record `A` DNS per mappare un nome di dominio personalizzato privato al relativo nome host privato. Fornisci il `DNSName` del tuo ID DNS VPC e l'ID della zona ospitata che hai creato nel passaggio precedente.

```
{
  "Changes": [
    {
      "Action": "UPSERT",
      "ResourceRecordSet": {
        "Name": "private.example.com",
        "Type": "A",
        "AliasTarget": {
          "DNSName": "vpce-abcd1234.execute-api.us-west-2.vpce.amazonaws.com",
          "HostedZoneId": "Z2OJLYMUO9EFXC",
          "EvaluateTargetHealth": false
        }
      }
    }
  ]
}
```

Il seguente comando change-resource-record-sets crea un record DNS per il nome di dominio personalizzato privato:

```
aws route53 change-resource-record-sets \
    --hosted-zone-id ZABCDEFG1234 \
    --change-batch file://path/to/your/setup-dns-record.json
```

Sostituisci `hosted-zone-id` con l'ID della zona ospitata di Route 53 del record DNS impostato nel tuo account. Il valore del parametro `change-batch` punta a un file JSON.

Passaggi successivi per un consumatore di API

Ora puoi invocare l'API privata nel tuo Account AWS. Nel tuo VPC, puoi usare il seguente comando curl per accedere al tuo nome di dominio personalizzato privato.

```
curl https://private.example.com/v1
```

Per ulteriori informazioni su altri modi per invocare la tua API privata, consulta the section called “Invocazione di un'API privata utilizzando un nome di dominio personalizzato”.

Consumatore di API: eliminazione dell'associazione di accesso al nome di dominio con un nome di dominio personalizzato privato

Se sei un consumatore di API, puoi eliminare la risorsa di associazione di accesso in qualsiasi momento. Il provider di API non può eliminare l'associazione di accesso al nome di dominio per te.

È consigliabile eliminare sempre un'associazione di accesso al nome di dominio quando non viene più utilizzata.

Console di gestione AWS

Per eliminare l'associazione di accesso al nome di dominio

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel pannello di navigazione principale, scegli Associazioni di accesso ai nomi di dominio.
3. Seleziona l'associazione di accesso al nome di dominio, quindi scegli Elimina.
4. Conferma la tua scelta, quindi scegli Elimina.

AWS CLI

Il seguente comando `delete-access-association` elimina l'associazione di accesso:

```
aws apigateway delete-domain-name-access-association \
    --domain-name-access-association-arn 'arn:aws:apigateway:us-
west-2:444455556666:/domainnameaccessassociations/domainname/private.example.com
+abcd1234/vpcesource/vpce-abcd1234efg'
```

Crea un nome di dominio personalizzato per le API private utilizzando CloudFormation

Il seguente CloudFormation modello di esempio crea un'API privata e un nome di dominio personalizzato privato, associa l'API privata al nome di dominio personalizzato e quindi crea un'associazione di accesso al nome di dominio. Devi specificare l'endpoint VPC, il nome di dominio e l'ARN del certificato.

Le seguenti considerazioni potrebbero influire sull'utilizzo di CloudFormation per creare un nome di dominio personalizzato privato:

- Non puoi rifiutare un'associazione di accesso al nome di dominio utilizzando. CloudFormation Per rifiutare un'associazione di accesso al nome di dominio, utilizza la AWS CLI.
- Usa la `AWS::ApiGateway::DomainNameV2` CloudFormation proprietà per creare un nome di dominio personalizzato privato.
- Usa la `AWS::ApiGateway:BasePathMappingV2` CloudFormation proprietà per creare una mappatura del percorso di base.

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  EndpointID:
    Type: String
    Default: vpce-abcd1234567efg
    Description: A VPC endpoint with enableDnsHostnames and enableDnsSupport set to
 true.
  DomainName:
    Type: String
    Default: private.example.com
    Description: A domain name that you own.
  CertificateArn:
    Type: String
    Default: arn:aws:acm:us-west-2:123456789:certificate/abcd-000-1234-0000-000000abcd
    Description: An ACM certificate that covers the domain name.
Resources:
  PrivateApi:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      EndpointConfiguration:
        Types:
          - PRIVATE
        VpcEndpointIds:
          - !Ref EndpointID
```

```
      Name: private-api
      Policy:
        Statement:
          - Action: 'execute-api:Invoke'
            Effect: Allow
            Principal: '*'
            Resource: 'execute-api:/*'
          - Action: 'execute-api:Invoke'
            Condition:
              StringNotEquals:
                'aws:SourceVpce': !Ref EndpointID
            Effect: Deny
            Principal: '*'
            Resource: 'execute-api:/*'
        Version: 2012-10-17
  PrivateApiDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    Properties:
      RestApiId: !Ref PrivateApi
      Description: Private API deployment
    DependsOn:
      - PrivateApiMethod
  PrivateApiStage:
    Type: 'AWS::ApiGateway::Stage'
    Properties:
      RestApiId: !Ref PrivateApi
      DeploymentId: !Ref PrivateApiDeployment
      StageName: prod
  PrivateApiMethod:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      HttpMethod: ANY
      ResourceId: !GetAtt PrivateApi.RootResourceId
      RestApiId: !Ref PrivateApi
      AuthorizationType: NONE
      Integration:
        Type: MOCK
        RequestTemplates:
          application/json: "{\"statusCode\": 200}"
        IntegrationResponses:
          - StatusCode: '200'
      MethodResponses:
        - StatusCode: '200'
  PrivateDomainName:
```

```
    Type: AWS::ApiGateway::DomainNameV2
    Properties:
      DomainName: !Ref DomainName
      CertificateArn: !Ref CertificateArn
      EndpointConfiguration:
        Types:
          - PRIVATE
      SecurityPolicy: TLS_1_2
      Policy:
        Statement:
            - Action: 'execute-api:Invoke'
              Effect: Allow
              Principal: '*'
              Resource: 'execute-api:/*'
            - Action: 'execute-api:Invoke'
              Condition:
                StringNotEquals:
                  'aws:SourceVpce': !Ref EndpointID
              Effect: Deny
              Principal: '*'
              Resource: 'execute-api:/*'
        Version: 2012-10-17
  PrivateBasePathMapping:
    Type: AWS::ApiGateway::BasePathMappingV2
    DependsOn:
      - PrivateApiStage
    Properties:
      BasePath: prod
      DomainNameArn: !GetAtt PrivateDomainName.DomainNameArn
      RestApiId: !Ref PrivateApi
      Stage: prod
  DomainNameAccessAssociation:
    Type: AWS::ApiGateway::DomainNameAccessAssociation
    Properties:
      DomainNameArn: !GetAtt PrivateDomainName.DomainNameArn
      AccessAssociationSource: !Ref EndpointID
      AccessAssociationSourceType: VPCE
```

Invocazione di un'API privata

Puoi invocare un'API privata solo dall'interno di un VPC utilizzando un endpoint VPC. La tua API privata deve avere una policy delle risorse che consenta a specifici VPC ed endpoint VPC di invocare l'API.

Se richiami un'API privata senza utilizzare un nome di dominio personalizzato o nomi DNS privati e le API o il nome di dominio utilizzano una politica di sicurezza che inizia con`SecurityPolicy_`, devi impostare la modalità di accesso agli endpoint su. `BASIC` Per ulteriori informazioni, consulta the section called “Modalità di accesso agli endpoint”.

Invocazione di un'API privata utilizzando un nome di dominio personalizzato

Per invocare un'API privata utilizzando un nome di dominio personalizzato, è necessario che l'endpoint VPC disponga di un'associazione di accesso ai nomi di dominio con un nome di dominio personalizzato e il nome di dominio personalizzato deve consentire l'accesso affinché l'endpoint VPC possa invocarla. Per ulteriori informazioni, consulta the section called “Nomi di dominio personalizzati per API private”.

Non ci sono differenze tra l'invocazione di un nome di dominio personalizzato privato in un VPC personale o in un altro. Account AWS Account AWS

Uso del nome di dominio personalizzato

All'interno del tuo VPC, puoi invocare la tua API utilizzando il nome di dominio personalizzato. L'esempio seguente è un comando curl per invocare il tuo nome di dominio personalizzato privato:

```
curl https://private.example.com
```

Uso di nomi host DNS privati specifici dell'endpoint

È possibile invocare l’API utilizzando il nome di dominio personalizzato come nome host DNS privato specifico dell’endpoint.

```
curl https://private-dns-hostname.execute-api.region.vpce.amazonaws.com/basepath -H
 'Host:custom-domain-name'
```

L’esempio seguente è un comando curl per invocare il nome di dominio personalizzato utilizzando il nome host DNS privato specifico dell’endpoint:

```
curl https://vpce-123456-abc000.execute-api.us-east-2.vpce.amazonaws.com/test -H
 'Host:private.example.com'
```

Invocazione di un'API privata senza utilizzare un nome di dominio personalizzato

Per invocare la tua API privata senza utilizzare un nome di dominio personalizzato, devi identificare i nomi DNS per l'API. La procedura seguente mostra come trovare i nomi DNS.

Console di gestione AWS

Per trovare i nomi DNS

1. Accedi Console di gestione AWS e apri la console Amazon VPC all'indirizzo. https://console.aws.amazon.com/vpc/
2. Nel pannello di navigazione principale, scegli Endpoint, quindi seleziona l'endpoint VPC dell'interfaccia per Gateway API.
3. Nel riquadro Dettagli verranno visualizzati 5 valori nel campo Nomi DNS. I primi tre sono i nomi DNS pubblici per l'API. Gli altri due sono i nomi DNS privati.

AWS CLI

Usa il seguente comando describe-vpc-endpoints per elencare i tuoi valori DNS.

```
aws ec2 describe-vpc-endpoints --vpc-endpoint-ids vpce-01234567abcdef012
```

I primi tre sono i nomi DNS pubblici per l'API. Gli altri due sono i nomi DNS privati.

Invocazione di un'API privata utilizzando un alias Route53

È possibile associare o dissociare un endpoint VPC da una REST API privata. Per ulteriori informazioni, consulta the section called “(Facoltativo) Associazione o dissociazione di un endpoint VPC da una REST API privata”.

Dopo aver associato gli endpoint VPC alla tua API privata, puoi utilizzare il seguente URL di base per invocare l'API:

```
https://{rest-api-id}-{vpce-id}.execute-api.{region}.amazonaws.com/{stage}
```

Ad esempio, se hai configurato il metodo `GET /pets` per la fase `test` e se l'ID REST API era `01234567ab`, il tuo ID endpoint VPC era `vpce-01234567abcdef012` e la Regione era `us-west-2`, puoi invocare la tua API come:

```
curl -v https://01234567ab-vpce-01234567abcdef012.execute-api.us-west-2.amazonaws.com/test/pets
```

Invocazione di un'API privata utilizzando i nomi DNS privati

Se hai abilitato il DNS privato, puoi accedere all'API privata utilizzando i seguenti nomi DNS privati:

```
{restapi-id}.execute-api.{region}.amazonaws.com
```

L'URL di richiamata di base dell'API è nel formato seguente:

```
https://{restapi-id}.execute-api.{region}.amazonaws.com/{stage}
```

Ad esempio, se hai configurato il metodo `GET /pets` per la fase `test` e se l'ID REST API era `01234567ab` e la tua Regione era `us-west-2`, puoi invocare la tua API privata inserendo il seguente URL in un browser:

```
https://01234567ab.execute-api.us-west-2.amazonaws.com/test/pets
```

In alternativa, puoi utilizzare il seguente comando cURL per invocare l'API privata:

```
curl -X GET https://01234567ab.execute-api.us-west-2.amazonaws.com/test/pets
```

> **Warning**
>
> Se abiliti il DNS privato per il tuo endpoint VPC, non sarai in grado di accedere all'endpoint predefinito per le API pubbliche. Per ulteriori informazioni, consulta Perché non riesco a connettere un endopoint VPC di API Gateway alla mia API pubblica?.

Richiama un'API privata utilizzando Direct Connect

Puoi utilizzarla Direct Connect per stabilire una connessione privata dedicata da una rete locale ad Amazon VPC e accedere al tuo endpoint API privato tramite quella connessione utilizzando nomi DNS pubblici.

Puoi anche utilizzare nomi DNS privati per accedere alla tua API privata da una rete locale configurando un endpoint in Amazon Route 53 Resolver ingresso e inoltrando tutte le query DNS del DNS privato dalla tua rete remota. Per ulteriori informazioni, consulta Inoltro delle query DNS in entrata nella Guida per sviluppatori di Amazon Route 53.

Invocazione di un'API privata tramite nomi host DNS pubblici specifici dell'endpoint

Puoi accedere all'API privata tramite nomi host DNS specifici dell'endpoint Si tratta di nomi host del DNS pubblico che contengono l'ID dell'endpoint VPC o l'ID dell'API per la tua API privata.

L'URL di base generato è nel formato seguente:

```
https://{public-dns-hostname}.execute-api.{region}.vpce.amazonaws.com/{stage}
```

Ad esempio, se hai configurato il metodo `GET /pets` per la fase `test` e se l'ID REST API era `abc1234`, il nome host DNS pubblico era `vpce-def-01234567` e la Regione era `us-west-2`, puoi invocare l'API privata tramite il relativo ID VPCe utilizzando l'intestazione `Host` in un comando cURL:

```
curl -v https://vpce-def-01234567.execute-api.us-west-2.vpce.amazonaws.com/test/pets -H
 'Host: abc1234.execute-api.us-west-2.amazonaws.com'
```

In alternativa, puoi invocare la tua API privata tramite il relativo ID utilizzando l'intestazione `x-apigw-api-id` in un comando cURL con il seguente formato:

```
curl -v https://{public-dns-hostname}.execute-api.{region}.vpce.amazonaws.com/{stage} -
H 'x-apigw-api-id:{api-id}'
```

Monitoraggio delle REST API in Gateway API

In questa sezione, puoi imparare a monitorare la tua API utilizzando CloudWatch metriche, CloudWatch log, Firehose e. AWS X-Ray Combinando log di CloudWatch esecuzione e CloudWatch metriche, puoi registrare errori e tracce di esecuzione e monitorare le prestazioni della tua API. È possibile anche registrare i log delle chiamate API in Firehose. Puoi anche utilizzarlo AWS X-Ray per tracciare le chiamate tramite i servizi downstream che compongono la tua API.

> **Note**
>
> L'API Gateway potrebbe non generare log e parametri nei seguenti casi:
>
> - Errori 413 per richiesta troppo grande
> - Errori 431 Campi intestazione richiesta troppo grandi
> - Errori 429 per troppe richieste

- Errori serie 400 per richieste inviate a un dominio personalizzato che non dispone del mapping API
- Errori serie 500 per malfunzionamenti interni

API Gateway non genererà log e parametri durante il test di un metodo REST API. Le CloudWatch voci vengono simulate. Per ulteriori informazioni, consulta the section called “Utilizzo della console per il test di un metodo API REST.”.

Argomenti

- Monitora l'esecuzione delle API REST con i CloudWatch parametri di Amazon
- Configurare la CloudWatch registrazione per le API REST in API Gateway
- Traccia delle richieste degli utenti alle REST API utilizzando X-Ray in Gateway API

Monitora l'esecuzione delle API REST con i CloudWatch parametri di Amazon

Puoi monitorare l'esecuzione delle API utilizzando CloudWatch, che raccoglie ed elabora i dati grezzi da API Gateway in metriche leggibili. near-real-time Queste statistiche vengono registrate per un periodo di 15 mesi, per permettere l'accesso alle informazioni storiche e offrire una prospettiva migliore sulle prestazioni del servizio o dell'applicazione Web. Per impostazione predefinita, i dati metrici di API Gateway vengono inviati automaticamente CloudWatch in periodi di un minuto. Per ulteriori informazioni, consulta What Is Amazon CloudWatch? nella Amazon CloudWatch User Guide.

I parametri forniti dall'API Gateway offrono informazioni che possono essere analizzate in diversi modi. L'elenco seguente mostra alcuni usi comuni dei parametri che sono suggerimenti per iniziare:

- Monitora le IntegrationLatencymetriche per misurare la reattività del backend.
- Monitora i parametri Latency per misurare la velocità di risposta complessiva delle chiamate API.
- Monitora le metriche CacheHitCounte le CacheMissCountmetriche per ottimizzare le capacità della cache e ottenere le prestazioni desiderate.

Argomenti

- Dimensioni e parametri di Amazon API Gateway

- Visualizzazione dei parametri di CloudWatch con il pannello di controllo delle API in API Gateway
- Visualizza le metriche dell'API Gateway nella console CloudWatch
- Visualizza gli eventi di registro dell'API Gateway nella CloudWatch console
- Strumenti di monitoraggio in AWS API Gateway

Dimensioni e parametri di Amazon API Gateway

Le metriche e le dimensioni che API Gateway invia ad Amazon CloudWatch sono elencate di seguito. Per ulteriori informazioni, consulta Monitora l'esecuzione delle API REST con i CloudWatch parametri di Amazon.

Parametri di API Gateway

Amazon API Gateway invia dati metrici CloudWatch ogni minuto.

Lo spazio dei nomi `AWS/ApiGateway` include le metriche descritte di seguito.

Metrica	Descrizione
4XXError	Il numero di errori lato client catturati in un dato periodo. API Gateway conta i codici di stato di risposta del gateway modificati come 4 XXError errori. La statistica `Sum` rappresenta questo parametro , ovvero il numero totale di errori 4XXError nel dato periodo. La statistica `Average` rappresenta la percentuale di errori 4XXError nel dato periodo, ovvero il numero totale di errori 4XXError diviso per il numero totale di richieste durante il periodo. Il denominatore corrisponde al parametro Count (di seguito). Unit: Count
5XXError	Il numero di errori lato server catturati in un dato periodo.

Metrica	Descrizione
	La statistica `Sum` rappresenta questo parametro , ovvero il numero totale di errori 5XXError nel dato periodo. La statistica `Average` rappresenta la percentuale di errori 5XXError nel dato periodo, ovvero il numero totale di errori 5XXError diviso per il numero totale di richieste durante il periodo. Il denominatore corrisponde al parametro Count (di seguito). Unit: Count
CacheHitCount	Il numero di richieste servite dalla cache API in un dato periodo. La statistica `Sum` rappresenta questo parametro, ovvero il numero totale di riscontri nella cache nel dato periodo. La statistica `Average` rappresenta la percentuale di riscontri nella cache, ovvero il numero totale di riscontri nella cache diviso per il numero totale di richieste durante il periodo. Il denominatore corrisponde al parametro Count (di seguito). Unit: Count

Metrica	Descrizione
CacheMissCount	Il numero di richieste servite dal back-end in un dato periodo, quando la memorizzazione nella cache delle API è abilitata. La statistica `Sum` rappresenta questo parametro , ovvero il numero totale di mancati riscontri nella cache nel dato periodo. La statistica `Average` rappresenta la percentuale di mancati riscontri nella cache, ovvero il numero totale di mancati riscontri nella cache diviso per il numero totale di richieste durante il periodo. Il denominatore corrisponde al parametro Count (di seguito). Unit: Count
Count	Il numero totale di richieste API in un dato periodo. La statistica `SampleCount` rappresenta questo parametro. Unit: Count
IntegrationLatency	Il tempo che intercorre tra il momento in cui API Gateway ritrasmette una richiesta al back-end e quando riceve da questo una risposta. Unit: Millisecond
Latency	Il tempo che intercorre tra il momento in cui API Gateway riceve una richiesta dal client e quando restituisce ad esso una risposta. La latenza include la latenza di integrazione e altri sovraccarichi dell'API Gateway. Unit: Millisecond

Dimensioni per i parametri

Per filtrare i parametri di API Gateway, puoi utilizzare le dimensioni nella seguente tabella.

> **Note**
>
> API Gateway rimuove i caratteri non ASCII dalla ApiName dimensione prima di inviare le metriche a. CloudWatch Se APIName non contiene caratteri ASCII, API ID viene utilizzato come ApiName.

Dimensione	Descrizione
ApiName	Filtra i parametri dell'API Gateway per l'API REST con il nome API specificato.
ApiName, Method, Resource, Stage	Filtra i parametri dell'API Gateway per il metodo API con nome API, fase, risorsa e metodo specificati. API Gateway non invierà queste metriche a meno che tu non abbia abilitato esplicitamente le metriche dettagliate CloudWatch . Nella console, scegli una fase, quindi seleziona Modifica per Log e tracciame nto. Seleziona Parametri dettagliati, quindi scegli Salva modifiche. In alternativa, puoi chiamare il AWS CLI comando update-stage per aggiornare la proprietà a. `metricsEnabled true` L'abilitazione di questi parametri comporta addebiti aggiuntivi sul tuo account. Per informazioni sui prezzi, consulta la pagina CloudWatchdei prezzi di Amazon.
ApiName, Stage	Filtra i parametri di API Gateway per la risorsa della fase API con il nome e la fase specificati per l'API.

Visualizzazione dei parametri di CloudWatch con il pannello di controllo delle API in API Gateway

È possibile utilizzare il pannello di controllo delle API nella console API Gateway per visualizzare i parametri di CloudWatch dell'API distribuita in API Gateway. Tali parametri sono visualizzati come un riepilogo delle attività delle API nel corso del tempo.

Argomenti

- Prerequisiti
- Esame delle attività nel pannello di controllo delle API

Prerequisiti

1. Devi aver creato un'API in API Gateway. Segui le istruzioni in Sviluppo di REST API in Gateway API.
2. Devi avere distribuito l'API almeno una volta. Segui le istruzioni in Implementazione di REST API in Gateway API.

Esame delle attività nel pannello di controllo delle API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere un'API.
3. Nel pannello di navigazione principale, seleziona Pannello di controllo.
4. In Fase, scegli la fase desiderata.
5. Scegli Intervallo di date per specificare un intervallo di date.
6. Aggiorna, se necessario, e visualizza le singole metriche presenti nei grafici intitolati Chiamate API, Latenza, Latenza di integrazione, Latenza, Errore 4xx ed Errore 5xx.

> **Tip**
>
> Per esaminare i parametri di CloudWatch a livello di metodo, assicurarsi di aver attivato CloudWatch Logs a livello di metodo. Per ulteriori informazioni su come configurare la registrazione dei log a livello di metodo, consultare Sostituzione delle impostazioni a livello di fase.

Visualizza le metriche dell'API Gateway nella console CloudWatch

I parametri vengono raggruppati prima in base allo spazio dei nomi del servizio e successivamente in base alle diverse combinazioni di dimensioni all'interno di ogni spazio dei nomi. Per visualizzare le metriche a livello di metodo per l'API, attiva le metriche dettagliate. Per ulteriori informazioni, consulta the section called “Aggiornamento delle impostazioni della fase”.

Per visualizzare le metriche dell'API Gateway utilizzando la console CloudWatch

1. Apri la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Se necessario, modifica Regione AWS. Dalla barra di navigazione, seleziona la regione in cui risiedono AWS le tue risorse.
3. Nel riquadro di navigazione, seleziona Parametri.
4. Nella scheda All metrics (Tutti i parametri), scegliere API Gateway (Gateway API).
5. Per visualizzare i parametri in base alla fase, selezionare il pannello By Stage (Per fase). Quindi, seleziona il tuo nome APIs e quello delle metriche.
6. Per visualizzare i parametri in base a un'API specifica, selezionare il pannello API Name (Nome API). Quindi, seleziona il tuo nome APIs e quello delle metriche.

Per visualizzare le metriche utilizzando la CLI AWS

1. Utilizza il seguente comando list-metrics per elencare le metriche:

   ```
   aws cloudwatch list-metrics --namespace "AWS/ApiGateway"
   ```

   Dopo aver creato una metrica, occorrono fino a 15 minuti perché venga visualizzata. Per visualizzare più rapidamente le statistiche delle metriche, usa o. get-metric-dataget-metric-statistics
2. Utilizzate il seguente get-metrics-statisticscomando per visualizzare la media su un periodo di tempo utilizzando intervalli di 5 minuti:

   ```
   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
   ```

Visualizza gli eventi di registro dell'API Gateway nella CloudWatch console

La sezione seguente spiega i prerequisiti necessari e come visualizzare gli eventi di registro di API Gateway nella CloudWatch console.

Prerequisiti

1. Devi aver creato un'API in API Gateway. Segui le istruzioni in Sviluppo di REST API in Gateway API.
2. È necessario avere implementato e richiamato l'API almeno una volta. A tale scopo, segui le istruzioni riportate in Implementazione di REST API in Gateway API e Invocazione di REST API in Gateway API.
3. È necessario che CloudWatch i log siano abilitati per una fase. Segui le istruzioni in Configurare la CloudWatch registrazione per le API REST in API Gateway.

Per visualizzare le richieste e le risposte API registrate utilizzando la console CloudWatch

1. Apri la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Se necessario, modifica Regione AWS. Dalla barra di navigazione, seleziona la regione in cui risiedono AWS le tue risorse. Per ulteriori informazioni, consulta Regioni ed endpoint.
3. Nel pannello di navigazione a sinistra, scegli Log, Gruppi di log.
4. Nella tabella Log Groups, scegliete un gruppo di log del nome API-Gateway-Execution-Logs_ {}/{stage-name}. rest-api-id
5. Nella tabella Log Streams (Flussi di log), scegliere un flusso di log. Puoi utilizzare il timestamp per individuare il flusso di log che ti interessa.
6. Selezionare Text (Testo) per vedere il testo non elaborato o scegliere Row (Riga) per visualizzare l'evento riga dopo riga.

> **Important**
>
> CloudWatch consente di eliminare gruppi o stream di log. Non eliminare manualmente i gruppi o i flussi di log delle API di API Gateway, lasciare che sia API Gateway a gestire queste risorse. L'eliminazione manuale dei flussi e dei gruppi di log potrebbe causare la mancata registrazione delle richieste e delle risposte dell'API. In tal caso puoi eliminare

l'intero gruppo di log dell'API e ridistribuire l'API. Ciò accade perché il Gateway API crea flussi o gruppi di log per una fase API nel momento in cui viene distribuita.

Strumenti di monitoraggio in AWS API Gateway

AWS fornisce vari strumenti che è possibile utilizzare per monitorare API Gateway. Alcuni di questi strumenti possono essere configurati in modo che eseguano automaticamente il monitoraggio, mentre altri richiedono l'intervento manuale. Si consiglia di automatizzare il più possibile i processi di monitoraggio.

Strumenti di monitoraggio automatizzati in AWS

Per controllare l'API Gateway e segnalare l'eventuale presenza di problemi è possibile usare i seguenti strumenti di monitoraggio automatici:

- Amazon CloudWatch Alarms: monitora una singola metrica in un periodo di tempo specificato ed esegui una o più azioni in base al valore della metrica rispetto a una determinata soglia in diversi periodi di tempo. L'azione è una notifica inviata a un argomento di Amazon Simple Notification Service (Amazon SNS) o a una policy di Amazon EC2 Auto Scaling. CloudWatch gli allarmi non richiamano azioni semplicemente perché si trovano in uno stato particolare; lo stato deve essere cambiato e mantenuto per un determinato numero di periodi. Per ulteriori informazioni, consulta Monitora l'esecuzione delle API REST con i CloudWatch parametri di Amazon.
- Amazon CloudWatch Logs: monitora, archivia e accedi ai tuoi file di registro da AWS CloudTrail o altre fonti. Per ulteriori informazioni, consulta What is CloudWatch Logs? nella Amazon CloudWatch User Guide.
- Amazon EventBridge (precedentemente chiamato CloudWatch Events): abbina gli eventi e li indirizza a una o più funzioni o flussi di destinazione per apportare modifiche, acquisire informazioni sullo stato e intraprendere azioni correttive. Per ulteriori informazioni, consulta What Is Amazon EventBridge? nella Guida EventBridge per l'utente.
- AWS CloudTrail Monitoraggio dei registri: condividi i file di CloudTrail registro tra account, monitora i file di registro in tempo reale inviandoli a CloudWatch Logs, scrivi applicazioni di elaborazione dei log in Java e verifica che i file di registro non siano stati modificati dopo la consegna da parte di. CloudTrail Per ulteriori informazioni, consulta Lavorare con i file di CloudTrail registro nella Guida per l'AWS CloudTrail utente.

Strumenti di monitoraggio manuali

Un'altra parte importante del monitoraggio di API Gateway riguarda il monitoraggio manuale degli elementi che gli CloudWatch allarmi non coprono. L'API Gateway e altre dashboard della AWS console forniscono una at-a-glance panoramica dello stato dell' AWS ambiente. CloudWatch Consigliamo anche di controllare i file di log nell'esecuzione dell'API.

- Il pannello di controllo di API Gateway mostra le seguenti statistiche per una determinata fase API nel corso di un determinato periodo di tempo:
  - API Calls (Chiamate API)
  - Cache HIt (Riscontri nella cache) solo quando è abilitato il caching dell'API.
  - Cache Miss (Mancato riscontro nella cache) solo quando è abilitato il caching dell'API.
  - Latenza
  - Integration Latency (Latenza di integrazione)
  - 4XX Error (Errore 4XX)
  - 5XX Error (Errore 5XX)
- La CloudWatch home page mostra:
  - Stato e allarmi attuali
  - Grafici degli allarmi e delle risorse
  - Stato di integrità dei servizi

  Inoltre, è possibile utilizzare CloudWatch per effettuare le seguenti operazioni:
  - Crea pannelli di controllo personalizzati per monitorare i servizi di interesse.
  - Crea grafici dei dati dei parametri per la risoluzione di problemi e il rilevamento di tendenze.
  - Cerca e sfoglia tutte le metriche AWS delle tue risorse
  - Crea e modifica gli allarmi per ricevere le notifiche dei problemi.

Creazione di CloudWatch allarmi per monitorare API Gateway

Puoi creare un CloudWatch allarme che invia un messaggio Amazon SNS quando l'allarme cambia stato. Un allarme controlla un singolo parametro in un periodo di tempo specificato ed esegue una o più operazioni in base al valore del parametro relativo a una determinata soglia in una serie di periodi di tempo. L'operazione corrisponde all'invio di una notifica a un argomento di Amazon SNS o a una policy di Auto Scaling. Gli allarmi richiamano azioni solo per cambiamenti di stato

sostenuti. CloudWatch gli allarmi non richiamano azioni semplicemente perché si trovano in uno stato particolare; lo stato deve essere cambiato e mantenuto per un determinato numero di periodi.

Configurare la CloudWatch registrazione per le API REST in API Gateway

Per aiutare a risolvere i problemi relativi all'esecuzione delle richieste o all'accesso del client alla tua API, puoi abilitare Amazon CloudWatch Logs per registrare le chiamate API. Per ulteriori informazioni su CloudWatch, consultathe section called “CloudWatch metriche”.

Esistono due tipi di accesso alle API: CloudWatch

- Registrazione dell'esecuzione: API Gateway registra le azioni intraprese per elaborare le richieste API, inclusi errori e tracce di esecuzione. Puoi anche configurare la consegna di Amazon CloudWatch Logs per indirizzare i log di esecuzione verso le tue destinazioni.
- Registrazione degli accessi: registri chi ha effettuato l'accesso alla tua API e in che modo. Crei il tuo gruppo di log, scegli un formato di registro e specifichi quali `$context` variabili includere.

Puoi abilitare la registrazione dell'esecuzione e quella degli accessi in modo reciprocamente indipendente.

Autorizzazioni per la registrazione CloudWatch

Per abilitare CloudWatch i log, devi concedere ad API Gateway l'autorizzazione per leggere e scrivere i log del tuo CloudWatch account. La policy AmazonAPIGatewayPushToCloudWatchLogs gestita dispone di tutte le autorizzazioni richieste.

> **Note**
>
> API Gateway chiama Servizio di token di sicurezza AWS per assumere il ruolo IAM, quindi assicurati che AWS STS sia abilitato per la regione. Per ulteriori informazioni, consulta Gestione di AWS STS in una AWS regione.

Per concedere queste autorizzazioni al tuo account, crea un ruolo IAM con `apigateway.amazonaws.com` entità attendibile, allega la policy precedente al ruolo IAM e imposta l'ARN del ruolo IAM nella WatchRoleArn proprietà cloud del tuo account. https://docs.aws.amazon.com/apigateway/latest/api/API_GetAccount.html È necessario impostare la WatchRoleArn proprietà cloud separatamente per ogni AWS regione in cui si desidera abilitare i log. CloudWatch

Se ricevi un errore durante l'impostazione del ruolo IAM ARN, controlla le impostazioni del tuo Servizio di token di sicurezza AWS account per assicurarti che AWS STS sia abilitato nella regione che stai utilizzando. Per ulteriori informazioni sull'abilitazione AWS STS, consulta Managing AWS STS in an AWS Region nella IAM User Guide.

Argomenti

- Registrazione dell'esecuzione per le API REST
- Registrazione degli accessi per le API REST

Registrazione dell'esecuzione per le API REST

La registrazione dell'esecuzione registra le azioni intraprese da API Gateway per elaborare una richiesta API. I dati registrati includono errori, tracce di esecuzione, valori o payload dei parametri di richiesta e risposta, dati utilizzati dagli autorizzatori Lambda, se sono necessarie le chiavi API e se i piani di utilizzo sono abilitati. Gateway API oscura le intestazioni di autorizzazione, i valori delle chiavi API e parametri di richiesta sensibili simili dai dati registrati nei log.

Per migliorare il livello di sicurezza, è consigliabile utilizzare la registrazione nei log dell'esecuzione a livello di `ERROR` o `INFO`. Potrebbe essere necessario eseguire questa operazione per rispettare vari framework di conformità. Per ulteriori informazioni, consultare Amazon API Gateway controls nella Guida per l'utente di AWS Security Hub .

Registrazione standard delle esecuzioni

Quando abiliti la registrazione delle esecuzioni su uno stage, API Gateway crea e gestisce automaticamente un gruppo di CloudWatch log Logs. Il gruppo di log è denominato`API-Gateway-Execution-Logs_{rest-api-id}/{stage_name}`. All'interno di ciascun gruppo di log, i log sono suddivisi in flussi di log ordinati in base all'ora dell'ultimo evento.

Con registrazione di esecuzione standard:

- Gli eventi di registro vengono troncati a 1 KB.
- API Gateway gestisce il nome del gruppo di log. Non puoi scegliere il nome del gruppo di log o inviare i log direttamente ad Amazon S3 o Firehose.

Per abilitare la registrazione di esecuzione standard, imposta `loggingLevel` on your stage su o. `ERROR` `INFO` Non è richiesta alcuna configurazione aggiuntiva.

Permissions

Per abilitare CloudWatch i log, devi concedere ad API Gateway l'autorizzazione per leggere e scrivere CloudWatch i log del tuo account. La policy AmazonAPIGatewayPushToCloudWatchLogs gestita dispone di tutte le autorizzazioni richieste.

Crea un ruolo IAM con `apigateway.amazonaws.com` entità attendibile, allega la policy precedente e imposta l'ARN del ruolo IAM sulla WatchRoleArn proprietà cloud del tuo account. https://docs.aws.amazon.com/apigateway/latest/api/API_GetAccount.html È necessario impostare questa proprietà separatamente per ogni AWS regione in cui si desidera abilitare i log. CloudWatch

> **Note**
>
> API Gateway chiama Servizio di token di sicurezza AWS per assumere il ruolo IAM, quindi assicurati che AWS STS sia abilitato per la regione. Per ulteriori informazioni, consulta Gestire AWS STS in una AWS regione.

Abilita la registrazione delle esecuzioni (console)

Prima di iniziare, implementa l'API in una fase e configura un ARN del ruolo CloudWatch Logs appropriato per il tuo account.

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway
2. Nel pannello di navigazione principale, scegli Impostazioni, quindi in Registrazione seleziona Modifica.
3. Per il ruolo di CloudWatch registro ARN, inserisci l'ARN di un ruolo IAM con le autorizzazioni appropriate. È necessario eseguire questa operazione una volta per ogni utente Account AWS che crea API utilizzando API Gateway.
4. Nel pannello di navigazione principale, scegli le API, scegli la tua API REST e quindi scegli Stages.
5. Scegli una fase, quindi nella sezione Log and tracing, scegli Modifica.
6. Seleziona un livello di registrazione dal menu a discesa CloudWatch Logs:
   - Disattivato: API Gateway non registra le richieste per questa fase.
   - Solo errori: API Gateway registra solo gli errori.
   - Errori e registri informativi: API Gateway registra tutti gli eventi.

7. (Facoltativo) Seleziona Data tracing per registrare i dati completi di richieste e risposte. Ciò può comportare la registrazione di dati sensibili.

   > **Note**
   >
   > Si consiglia di non utilizzare Tracciamento dei dati per le API di produzione.

8. (Facoltativo) Seleziona Metriche dettagliate per attivare le metriche dettagliate CloudWatch . Per ulteriori informazioni, consulta the section called “CloudWatch metriche”.
9. Scegli Salva modifiche.

Non è necessario ridistribuire l'API quando si aggiornano le impostazioni della fase.

Abilita la registrazione delle esecuzioni (CloudFormation)

Utilizzate la `MethodSettings` proprietà sulla risorsa dello stage per abilitare la registrazione delle esecuzioni.

```
TestStage:
  Type: AWS::ApiGateway::Stage
  Properties:
    StageName: test
    RestApiId: !Ref MyAPI
    DeploymentId: !Ref Deployment
    MethodSettings:
      - ResourcePath: "/*"
        HttpMethod: "*"
        LoggingLevel: INFO
```

Per abilitare CloudWatch i log, devi concedere ad API Gateway l'autorizzazione per leggere e scrivere CloudWatch i log del tuo account. Per ulteriori informazioni, consulta Associate account with IAM role nella Guida per l'utente di AWS CloudFormation .

CloudWatch Distribuzione dei log di esecuzione da Amazon Logs

La distribuzione di Amazon CloudWatch Logs fornisce un percorso alternativo per i log di esecuzione. Invece di scrivere su un gruppo di Gateway-managed log API, configuri una consegna che invia i log di esecuzione alle destinazioni che possiedi e controlli.

Con la distribuzione di Amazon CloudWatch Logs:

- Gli eventi di registro possono pesare fino a 1 MB e consentono di acquisire dati più completi su richieste e risposte. I payload che superano 1 MB vengono comunque troncati.
- Scegli la destinazione: i tuoi gruppi di CloudWatch log Logs, bucket Amazon S3 o stream Firehose.
- Puoi inviare i log a più destinazioni contemporaneamente.
- Sei tu a controllare la configurazione del gruppo di log, inclusa la denominazione e la conservazione.

> **Important**
>
> CloudWatch La consegna dei log sostituisce la registrazione di esecuzione standard. Quando si configura una distribuzione, API Gateway interrompe la scrittura nel gruppo di log gestito automaticamente. Non è possibile utilizzare entrambi i percorsi contemporaneamente per la stessa fase.

Quando crei una consegna, API Gateway indirizza i log di esecuzione alle destinazioni configurate anziché al gruppo di log gestito automaticamente. Quando si elimina una consegna, i log di esecuzione riprendono automaticamente il flusso verso il gruppo di log gestito automaticamente. Non sono necessarie modifiche alla configurazione di API Gateway oltre al prerequisito di impostazione `loggingLevel` su o. `ERROR` `INFO`

> **Note**
>
> La consegna dei log è la soluzione migliore. In rari casi, alcuni eventi di registro potrebbero non essere consegnati.

> **Warning**
>
> Prima di abilitare la distribuzione dei log, aggiorna eventuali dashboard, allarmi o filtri di abbonamento che fanno riferimento al gruppo di log gestito automaticamente (). `API-Gateway-Execution-Logs_{rest-api-id}/{stage_name}` Una volta che la consegna è attiva, quel gruppo di log non riceve più nuovi eventi.

La tabella seguente confronta la registrazione di esecuzione standard con la distribuzione di Amazon CloudWatch Logs.

	Registrazione di esecuzione standard	Consegna di Amazon CloudWatch Logs
Dimensione massima degli eventi di registro	1 KB	1 MB
Destinazioni	CloudWatch Log (gruppo di log gestito)	CloudWatch Logs, Amazon S3, Firehose (risorse proprie)
Diverse destinazioni	No	Sì
Nome del gruppo di log	Risolto da API Gateway	Scegli tu
Configurazione	Automatico quando la registrazione è abilitata	Richiede la creazione di una consegna Amazon CloudWatch Logs
Prerequisito	Impostato `loggingLevel` su `ERROR` o `INFO`	Impostato `loggingLevel` su `ERROR` o `INFO`

> **Note**
>
> Se si imposta su `loggingLevel` `OFF` dopo aver configurato una consegna, non viene generato o consegnato alcun registro di esecuzione. Le destinazioni di consegna rimangono configurate ma non ricevono dati.

Per creare una spedizione, consultathe section called “Crea una consegna di log”.

Crea una distribuzione di log per i log di esecuzione dell'API REST

Puoi indirizzare i log di esecuzione ai tuoi gruppi di CloudWatch log Logs, ai bucket Amazon S3 o ai tuoi stream Firehose. A tale scopo, crea una consegna Amazon Logs. CloudWatch Ciò prevede tre passaggi: creare una fonte di consegna, creare una destinazione di consegna e collegarli a una consegna.

Prerequisiti

Prima di creare un registro di consegna, assicurati di disporre di quanto segue:

- La registrazione dell'esecuzione è abilitata nella fase dell'API REST. Imposta `loggingLevel` su `ERROR` o `INFO`. Se la registrazione non è abilitata, la creazione dell'origine di consegna non riesce.
- Una risorsa di destinazione: un gruppo di CloudWatch log Logs, un bucket Amazon S3 o uno stream Firehose di tua proprietà.
- Autorizzazioni per chiamare le API di distribuzione di Amazon Logs CloudWatch . Per ulteriori informazioni, consulta Abilitare la registrazione dai AWS servizi nella Amazon CloudWatch Logs User Guide.

Gateway-specific Valori API per la consegna di Amazon CloudWatch Logs

Quando crei una distribuzione per i log di esecuzione di API Gateway, utilizza i seguenti valori:

ARN risorsa

```
arn:aws:apigateway:{region}:{account-id}:/restapis/{rest-api-id}/
stages/{stage-name}
```

Tipo di log

```
EXECUTION_LOGS
```

Crea una consegna di log (AWS CLI)

L'esempio seguente crea una consegna che invia i log di esecuzione da un'API REST a un gruppo di CloudWatch log Logs di tua proprietà.

1. Crea la fonte di consegna

   Registra la tua API REST come fonte di consegna.

```
aws logs put-delivery-source \
    --name my-apigw-source \
    --log-type EXECUTION_LOGS \
    --resource-arn arn:aws:apigateway:{region}:{account-id}:/restapis/{rest-api-
id}/stages/{stage-name}
```

2. Crea la destinazione di consegna

   Registra il tuo gruppo di log come destinazione di consegna.

```
aws logs put-delivery-destination \
    --name my-log-destination \
    --delivery-destination-configuration
 "destinationResourceArn=arn:aws:logs:{region}:{account-id}:log-group:{log-group-
name}"
```

3. Crea la consegna

   Collega l'origine alla destinazione.

```
aws logs create-delivery \
    --delivery-source-name my-apigw-source \
    --delivery-destination-arn arn:aws:logs:{region}:{account-id}:delivery-
destination:my-log-destination
```

Per inviare i log ad Amazon S3 o Firehose, specifica l'ARN della risorsa appropriata nel comando. `put-delivery-destination` Per ulteriori informazioni sul flusso completo di distribuzione di Amazon CloudWatch Logs, consulta Enable logging from AWS services nella Amazon CloudWatch Logs User Guide.

Per inviare i log a più destinazioni, crea consegne aggiuntive utilizzando la stessa fonte di consegna con destinazioni diverse.

Campi di registrazione

Con la distribuzione di Amazon CloudWatch Logs, i log di esecuzione includono campi strutturati che puoi personalizzare per destinazione. Sono disponibili i seguenti campi:

- `resource_arn`— L'ARN della fase API
- `event_timestamp`— L'ora dell'evento di registro
- `api_id`— L'identificatore dell'API REST
- `stage`— Il nome della fase
- `resource_path`— Il percorso della risorsa della richiesta
- `http_method`— Il metodo HTTP
- `payload`— Il contenuto del registro di esecuzione (uguale ai registri di esecuzione standard)

È possibile selezionare i campi da includere e il formato di output per ciascuna destinazione utilizzando il `--record-fields` parametro on `create-delivery` e il `--output-format` parametro on`put-delivery-destination`.

I formati di output disponibili dipendono dalla destinazione:

- CloudWatch Registri: json, semplici
- Amazon S3: json, semplice, w3c, parquet
- Firehose — json, semplice

> **Corrisponde all'output del registro di esecuzione standard**
>
> Se desideri un output di consegna che corrisponda alla registrazione di esecuzione standard (solo payload, senza campi aggiuntivi), aggiungi `--output-format "plain"` a `put-delivery-destination` e `--record-fields "payload" --field-delimiter ""` a. `create-delivery`

Crea un registro di consegna (CloudFormation)

Quando assemblate lo stage ARN per i CloudFormation modelli, utilizzate il seguente schema:

```
  DeliverySource:
    Type: AWS::Logs::DeliverySource
    Properties:
      Name: my-apigw-source
      LogType: EXECUTION_LOGS
      ResourceArn: !Sub "arn:${AWS::Partition}:apigateway:${AWS::Region}:
${AWS::AccountId}:/restapis/${MyApi}/stages/${MyStageName}"
```

Crea una consegna del registro (console API Gateway)

Puoi creare una consegna dei log dalla sezione Log and tracing della pagina dei dettagli della fase.

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli la tua API REST, quindi scegli Stages.
3. Scegliere una fase.
4. Nella sezione Log and tracing, in Destinazioni di consegna dei log, scegli Aggiungi destinazione.

5. Scegli un tipo di destinazione (CloudWatch Logs, Amazon S3 o Firehose) e specifica la risorsa di destinazione.
6. Selezionare Salva.

> **Important**
>
> È necessario abilitare la registrazione dell'esecuzione sullo stage prima di poter aggiungere le destinazioni di consegna. Se la registrazione è impostata su Off, la configurazione di consegna non genera registri.

> **Note**
>
> Per configurare le destinazioni di consegna tra più account, utilizza la CLI AWS . È necessario creare una politica di destinazione della consegna nell'account di destinazione prima di creare la consegna nell'account di origine. Per ulteriori informazioni, consulta l'esempio Cross-account di consegna nella Guida per l'utente di Amazon CloudWatch Logs.

Verifica la consegna del registro di esecuzione

Dopo aver creato una consegna di log, verifica che la consegna sia attiva e che i log di esecuzione arrivino a destinazione.

Per confermare che una consegna è attiva, usa il `describe-deliveries` comando:

```
aws logs describe-deliveries
```

Quindi invia una richiesta di prova e verifica la tua destinazione:

1. Invia una richiesta di test alla tua API REST.

   ```
   curl https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/{resource}
   ```

2. Attendi 1—2 minuti per la propagazione dei log.
3. Controlla la tua destinazione per gli eventi di registro:

- CloudWatch Registri: apri la console CloudWatch Logs e accedi al tuo gruppo di log. Dovresti vedere uno stream di log con gli eventi recenti.
- Amazon S3: apri la console Amazon S3 e accedi al tuo bucket. Cerca oggetti con un prefisso di percorso che includa il tuo ID API REST.
- Firehose: controlla le metriche del flusso di consegna nella console Firehose per confermare il flusso dei dati.

Risoluzione dei problemi

Se i log non arrivano a destinazione, controlla quanto segue:

- La registrazione è abilitata: conferma che `loggingLevel` sia impostata `INFO` su `ERROR` o sullo stage. Se la registrazione è attiva`OFF`, non viene generato alcun registro indipendentemente dalla configurazione di consegna.
- Stato della consegna: utilizza il comando `get-delivery` CLI per controllare lo stato della consegna e confermare che è attiva.

```
aws logs get-delivery --id {delivery-id}
```

- Autorizzazioni: verifica che la politica delle risorse di destinazione consenta ad Amazon CloudWatch Logs di scrivere su di esse. Per ulteriori informazioni, consulta Abilitare la registrazione dai AWS servizi nella Amazon CloudWatch Logs User Guide.

Eliminare una consegna di log per i log di esecuzione dell'API REST

Quando elimini una consegna di log, API Gateway riprende a scrivere i log di esecuzione nel gruppo di log gestito automaticamente (`API-Gateway-Execution-Logs_{rest-api-id}/{stage_name}`), a condizione che la registrazione delle esecuzioni sia ancora abilitata sullo stage.

Eliminare una consegna di log (AWS CLI)

Per eliminare una consegna, è necessario eliminarla e quindi, facoltativamente, eliminare l'origine e la destinazione della consegna.

1. Elimina la spedizione

```
aws logs delete-delivery --id {delivery-id}
```

2. (Facoltativo) Elimina la fonte di consegna

```
aws logs delete-delivery-source --name my-apigw-source
```

3. (Facoltativo) Eliminare la destinazione di consegna

```
aws logs delete-delivery-destination --name my-log-destination
```

> **Note**
>
> L'eliminazione dell'origine o della destinazione della consegna è facoltativa. Puoi riutilizzarli per creare nuove consegne in un secondo momento.

Eliminare una consegna di log (console API Gateway)

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli la tua API REST, quindi scegli Stages.
3. Scegliere una fase.
4. Nella sezione Log and tracing, in Destinazioni di consegna dei log, seleziona la destinazione che desideri rimuovere.
5. Scegliere Elimina.

Mantieni la continuità dei log

> **Tip**
>
> Per mantenere i log nello stesso gruppo di log e mantenere le impostazioni di conservazione e i filtri di abbonamento esistenti, imposta il gruppo di log gestito automaticamente (`API-Gateway-Execution-Logs_{rest-api-id}/{stage_name}`) come destinazione di consegna. I nomi dei flussi di log saranno diversi da quelli creati dalla registrazione standard.

Registrazione degli accessi per le API REST

Nella registrazione degli accessi, registri chi ha avuto accesso alla tua API e in che modo il chiamante ha effettuato l'accesso all'API. Puoi creare il tuo gruppo di log o scegliere un gruppo di log esistente.

Per specificare i dettagli di accesso, seleziona le variabili `$context`, un formato di log e un gruppo di log come destinazione.

Formati di log di accesso

Il formato del log di accesso deve includere almeno `$context.requestId` o `$context.extendedRequestId`. È consigliabile includerli entrambi nel formato di registro.

`$context.requestId`

In questo modo il valore dell'intestazione `x-amzn-RequestId` viene registrato nel log. I client possono sostituire il valore nell'intestazione `x-amzn-RequestId` con un valore nel formato di un identificatore univoco universale (UUID). API Gateway restituisce questo ID richiesta nell'intestazione della risposta di `x-amzn-RequestId`. Gateway API sostituisce gli ID di richiesta sovrascritti che non sono nel formato di un UUID con *`UUID`*`_REPLACED_INVALID_REQUEST_ID` nei log di accesso.

`$context.extendedRequestId`

Il valore di extendedRequestID è un ID univoco generato da Gateway API. API Gateway restituisce questo ID richiesta nell'intestazione della risposta di `x-amz-apigw-id`. Un caller API non può fornire o ignorare questo ID di richiesta. Potrebbe essere necessario fornire questo valore all' AWS assistenza per risolvere i problemi relativi all'API. Per ulteriori informazioni, consulta the section called “Variabili per la registrazione dei log degli accessi per Gateway API”.

Seleziona un formato di log adottato anche dal back-end analitico, ad esempio Common Log Format (CLF), JSON, XML o CSV. Quindi puoi alimentarlo direttamente con i log di accesso per l'elaborazione e il rendering dei parametri. Per definire il formato del registro, imposta l'ARN del gruppo di log sulla `accessLogSettings/destinationArn` proprietà sullo stage. https://docs.aws.amazon.com/apigateway/latest/api/API_Stage.html È possibile ottenere un ARN del gruppo di log nella CloudWatch console. Per definire il formato del registro degli accessi, impostate un formato scelto sulla `accessLogSetting/format` proprietà sullo `stage`.

Esempi di alcuni formati di log delle operazioni di accesso utilizzati con maggiore frequenza sono mostrati nella console API Gateway ed elencati qui di seguito.

- `CLF` (Common Log Format):

```
$context.identity.sourceIp $context.identity.caller $context.identity.user
 [$context.requestTime]"$context.httpMethod $context.resourcePath
```

```
 $context.protocol" $context.status $context.responseLength $context.requestId
 $context.extendedRequestId
```

- JSON:

```
{ "requestId":"$context.requestId",
 "extendedRequestId":"$context.extendedRequestId","ip": "$context.identity.sourceIp",
 "caller":"$context.identity.caller", "user":"$context.identity.user",
 "requestTime":"$context.requestTime", "httpMethod":"$context.httpMethod",
 "resourcePath":"$context.resourcePath", "status":"$context.status",
 "protocol":"$context.protocol", "responseLength":"$context.responseLength" }
```

- XML:

```
<request id="$context.requestId"> <extendedRequestId>$context.extendedRequestId</
extendedRequestId> <ip>$context.identity.sourceIp</ip> <caller>
$context.identity.caller</caller> <user>$context.identity.user</user> <requestTime>
$context.requestTime</requestTime> <httpMethod>$context.httpMethod</httpMethod>
 <resourcePath>$context.resourcePath</resourcePath> <status>$context.status</status>
 <protocol>$context.protocol</protocol> <responseLength>$context.responseLength</
responseLength> </request>
```

- CSV (valori separati da virgola):

```
$context.identity.sourceIp,$context.identity.caller,$context.identity.user,
$context.requestTime,$context.httpMethod,$context.resourcePath,$context.protocol,
$context.status,$context.responseLength,$context.requestId,$context.extendedRequestId
```

Configura la registrazione degli accessi utilizzando la console API Gateway

Prima di configurare la registrazione degli accessi, implementa l'API in una fase e configura un ARN del ruolo CloudWatch Logs appropriato per il tuo account.

1. Accedi alla console API Gateway all'indirizzo. https://console.aws.amazon.com/apigateway
2. Nel riquadro di navigazione principale scegli API, quindi esegui una delle seguenti operazioni:
   a. Seleziona un'API esistente e quindi scegli una fase.
   b. Crea un'API e implementala in una fase.
3. Nel riquadro di navigazione principale scegli Fasi.
4. Nella sezione Log e tracciamento scegli Modifica.

5. Attiva Registrazione accesso personalizzato.
6. Immetti l'ARN di un gruppo di log in ARN di destinazione del log degli accessi. Il formato dell'ARN è `arn:aws:logs:{region}:{account-id}:log-group:log-group-name`.
7. In Formato dei log immetti un formato di log. Puoi scegliere CLF, JSON, XML o CSV. Per ulteriori informazioni sui formati di log di esempio, consulta the section called “Formati di log di accesso”.
8. Scegli Save changes (Salva modifiche).

API Gateway ora è pronto a registrare i log delle richieste all'API. Non è necessario ridistribuire l'API quando si aggiornano le impostazioni delle fasi, i log o le variabili delle fasi.

Configura la registrazione degli accessi utilizzando CloudFormation

Utilizza il seguente CloudFormation modello di esempio per creare un gruppo di CloudWatch log di Amazon Logs e configurare la registrazione degli accessi per una fase. Per abilitare CloudWatch i log, devi concedere ad API Gateway l'autorizzazione per leggere e scrivere CloudWatch i log del tuo account. Per ulteriori informazioni, consulta Associate account with IAM role nella Guida per l'utente di AWS CloudFormation .

```
 TestStage:
   Type: AWS::ApiGateway::Stage
   Properties:
     StageName: test
     RestApiId: !Ref MyAPI
     DeploymentId: !Ref Deployment
     AccessLogSetting:
       DestinationArn: !GetAtt MyLogGroup.Arn
       Format: $context.extendedRequestId $context.identity.sourceIp
$context.identity.caller $context.identity.user [$context.requestTime]
"$context.httpMethod $context.resourcePath $context.protocol" $context.status
$context.responseLength $context.requestId
 MyLogGroup:
   Type: AWS::Logs::LogGroup
   Properties:
     LogGroupName: !Join
       - '-'
       - - !Ref MyAPI
         - access-logs
```

Registrazione dei log delle chiamate REST API su Amazon Data Firehose in Gateway API

Per facilitare il debug dei problemi relativi all'accesso dei client all'API, è possibile registrare i log delle chiamate API su Amazon Data Firehose. Per ulteriori informazioni su Firehose, consulta What Is Amazon Data Firehose?

Per la registrazione degli accessi, puoi solo abilitare CloudWatch o FireHose, non puoi abilitare entrambi. Tuttavia, è possibile abilitare CloudWatch la registrazione dell'esecuzione e Firehose la registrazione degli accessi.

Argomenti

- Formati di log di Firehose per Gateway API
- Autorizzazioni per la registrazione dei log di Firehose
- Configurazione della registrazione dei log degli accessi di Firehose tramite la console Gateway API

Formati di log di Firehose per Gateway API

La registrazione Firehose utilizza lo stesso formato della registrazione. CloudWatch

Autorizzazioni per la registrazione dei log di Firehose

Quando la registrazione dei log degli accessi di Firehose è abilitata su una fase, Gateway API crea nell'account un ruolo collegato al servizio, se non è già esistente. Il ruolo è denominato `AWSServiceRoleForAPIGateway` ed è collegato alla policy gestita `APIGatewayServiceRolePolicy`. Per ulteriori informazioni sui ruoli collegati ai servizi, consulta Utilizzo dei ruoli collegati ai servizi.

> **Note**
>
> Il nome del flusso di Firehose deve essere `amazon-apigateway-{your-stream-name}`.

Configurazione della registrazione dei log degli accessi di Firehose tramite la console Gateway API

Per configurare la registrazione API, devi aver distribuito l'API in una fase. È inoltre necessario aver creato un flusso di Firehose.

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.

2. Esegui una delle seguenti operazioni:

   a. Seleziona un'API esistente e quindi scegli una fase.

   b. Crea un'API e distribuiscila in una fase.

3. Nel riquadro di navigazione principale scegli Fasi.

4. Nella sezione Log e tracciamento scegli Modifica.

5. Per abilitare la registrazione dei log degli accessi in un flusso Firehose:

   a. Attiva Registrazione accesso personalizzato.

   b. In ARN di destinazione del log degli accessi inserisci l'ARN di un flusso Firehose. Il formato dell'ARN è `arn:aws:firehose:{region}:{account-id}:deliverystream/amazon-apigateway-{your-stream-name}`.

      > **Note**
      >
      > Il nome del flusso di Firehose deve essere `amazon-apigateway-{your-stream-name}`.

   c. In Formato dei log immetti un formato di log. Puoi scegliere CLF, JSON, XML o CSV. Per ulteriori informazioni sui formati di log di esempio, consulta the section called “Formati di log di accesso”.

6. Scegli Save changes (Salva modifiche).

Gateway API è ora pronto per registrare i log delle richieste all'API su Firehose. Non è necessario ridistribuire l'API quando si aggiornano le impostazioni delle fasi, i log o le variabili delle fasi.

Variabili per la registrazione dei log degli accessi per Gateway API

Nella registrazione degli accessi, in qualità di sviluppatore dell'API puoi registrare chi ha avuto accesso alla tua API e in che modo l'intermediario ha avuto accesso all'API. Puoi creare un gruppo di log personalizzato o sceglierne uno esistente che potrebbe essere gestito da API Gateway. Per specificare i dettagli di accesso, è possibile utilizzare le seguenti variabili `$context` con distinzione tra maiuscole e minuscole.

Per l’elenco di tutte le variabili di riferimento per le trasformazioni dei dati, consultare the section called “Variabili per le trasformazioni dei dati”.

Parametro	Description
`$context.accountId`	L'ID dell' AWS account del proprietario dell'API.
`$context.apiId`	Identificatore assegnato da API Gateway all'API.
`$context.authorize.error`	Il messaggio di errore di autorizzazione.
`$context.authorize.latency`	La latenza di autorizzazione in ms.
`$context.authorize.status`	Il codice di stato restituito da un tentativo di autorizzazione.
`$context.authorizer.claims.` *`property`*	Proprietà delle richieste restituite dal pool di utenti di Amazon Cognito dopo che l'interme diario del metodo viene autenticato correttam ente. Per ulteriori informazioni, consulta the section called “Utilizza un pool di utenti di Amazon Cognito come autorizzazione per un'API REST”. **Note** La chiamata di `$context.authorizer.claims` restituisce null.
`$context.authorizer.error`	Il messaggio di errore restituito da un'autori zzazione.
`$context.authorizer.integrationLatency`	Latenza di integrazione del sistema di autorizza zione in ms.
`$context.authorizer.integrationStatus`	Il codice di stato restituito da un'autorizzazione Lambda.
`$context.authorizer.latency`	La latenza di autorizzazione in ms.

Parametro	Description
`$context.authorizer.principalId`	Identificazione dell'utente dell'entità principale associata al token inviata dal client e restituita da un'autorizzazione Lambda in API Gateway (precedentemente noto come autorizzazione ad hoc). Per ulteriori informazioni, consulta the section called “Uso di autorizzazioni Lambda”.

`$context.authorizer.` *`property`*	Valore in formato stringa della coppia chiave/valore specificata della mappa `context` restituita da una funzione delle autorizzazioni Lambda di API Gateway. Ad esempio, se le autorizzazioni restituiscono la mappa `context` seguente:
	`"context" : {`
	`"key": "value",`
	`"numKey": 1,`
	`"boolKey": true`
	`}`
	La chiamata di `$context.authorizer.key` restituisce la stringa `"value"`, la chiamata di `$context.authorizer.numKey` restituisce la stringa `"1"` e la chiamata di `$context.authorizer.boolKey` restituisce la stringa `"true"`.
	Infatti*`property`*, l'unico carattere speciale supportato è il carattere di sottolineatura(`_`).
	Per ulteriori informazioni, consulta the section called “Uso di autorizzazioni Lambda”.

`$context.authorizer.requestId`	L'ID della richiesta dell' AWS endpoint.
`$context.authorizer.status`	Il codice di stato restituito da un'autorizzazione.

Parametro	Description
`$context.authenticate.error`	Il messaggio di errore restituito da un tentativo di autenticazione.
`$context.authenticate.latency`	La latenza di autenticazione in ms.
`$context.authenticate.status`	Il codice di stato restituito da un tentativo di autenticazione.
`$context.awsEndpointRequestId`	L'ID della richiesta dell' AWS endpoint.
`$context.cipherSuite`	Il codice, in formato IANA, negoziato durante l'handshake TLS tra il client e API Gateway.
`$context.customDomain.basePathMatched`	Il percorso per una mappatura API a cui corrisponde una richiesta in ingresso. Applicabile quando un client utilizza un nome di dominio personalizzato per accedere a un'API. Ad esempio, se un client invia una richiesta a `https://api.example.com/v1/orders/1234` e la richiesta corrisponde alla mappatura API con il percorso `v1/orders`, il valore è `v1/orders`. Per ulteriori informazioni, consulta the section called “Mappature API”.
`$context.customDomain.routingRuleIdMatched`	La regola di routing a cui corrisponde una richiesta in ingresso. Applicabile quando un client utilizza un nome di dominio personalizzato per accedere a un'API. Per ulteriori informazioni, consulta the section called “Regole di routing”.
`$context.deploymentId`	ID dell'implementazione API.

Parametro	Description
`$context.domainName`	Nome di dominio completo usato per richiamare l'API. Deve essere lo stesso dell'intestazione `Host` in ingresso.
`$context.domainPrefix`	Prima etichetta di `$context.domainName` .
`$context.endpointType`	Il tipo di endpoint dell'API.
`$context.error.message`	Stringa contenente un messaggio di errore di API Gateway. Questa variabile può essere utilizzata solo per la semplice sostituzione di variabili in un modello di GatewayResponsebody mapping, che non viene elaborato dal motore Velocity Template Language, e nella registrazione degli accessi. Per ulteriori informazioni, consultare the section called “Metriche” e the section called “Configurazione delle risposte del gateway per la personalizzazione delle risposte agli errori”.
`$context.error.messageString`	Valore `$context.error.message` tra virgolette, ovvero `"$context.error.message"` .
`$context.error.responseType`	Un tipo di. GatewayResponse Questa variabile può essere utilizzata solo per la semplice sostituzione di variabili in un modello di GatewayResponsebody mapping, che non viene elaborato dal motore Velocity Template Language, e nella registrazione degli accessi. Per ulteriori informazioni, consultare the section called “Metriche” e the section called “Configurazione delle risposte del gateway per la personalizzazione delle risposte agli errori”.

Parametro	Description
`$context.error.validationErrorString`	Stringa contenente un messaggio dettagliato di errore di convalida.
`$context.extendedRequestId`	ID esteso generato da API Gateway e assegnato alla richiesta API. L'ID della richiesta esteso contiene ulteriori informazioni utili per il debug e la risoluzione dei problemi.
`$context.httpMethod`	Metodo HTTP usato. I valori validi sono: `DELETE`, `GET`, `HEAD`, `OPTIONS`, `PATCH`, `POST` e `PUT`.
`$context.identity.accountId`	L'ID dell'account associato alla richiesta. AWS
`$context.identity.apiKey`	Per i metodi API che richiedono una chiave API, questa variabile è la chiave API associata alla richiesta del metodo. Per i metodi che non richiedono una chiave API, questa variabile è null. Per ulteriori informazioni, consulta the section called "Piani di utilizzo".
`$context.identity.apiKeyId`	ID chiave API associato a una richiesta API che richiede una chiave API.
`$context.identity.caller`	Identificatore dell'entità principale dell'intermediario che ha firmato la richiesta. Supportato per risorse che utilizzano l'autorizzazione IAM.

Parametro	Description
`$context.identity.cognitoAuthenticationProvider`	Un elenco separato da virgole con tutti i provider di autenticazione Amazon Cognito utilizzati dal chiamante che effettua la richiesta . Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito. Ad esempio, per un'identità di un pool di utenti Amazon Cognito, `cognito-idp.` *`region`*`.amazonaws.com/` *`user_pool_id`* `,cognito-idp.` *`region`*`.amazonaws.com/` *`user_pool_id`* `:CognitoSignIn:` *`token subject claim`* Per informazioni sui provider di autenticazione Amazon Cognito disponibili, consulta Using Federated Identities nella Guida per gli sviluppatori di Amazon Cognito.
`$context.identity.cognitoAuthenticationType`	Tipo di autenticazione Amazon Cognito dell'intermediario da cui proviene la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito. I valori possibili includono `authenticated` per le identità autenticate e `unauthenticated` per le identità non autenticate.
`$context.identity.cognitoIdentityId`	ID identità di Amazon Cognito dell'intermediario da cui proviene la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito.
`$context.identity.cognitoIdentityPoolId`	ID pool di identità di Amazon Cognito dell'intermediario da cui proviene la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito.

Parametro	Description
`$context.identity.principalOrgId`	L'ID organizzazione AWS.
`$context.identity.sourceIp`	L'indirizzo IP di origine della connessione TCP immediata da cui proviene la richiesta all'endpoint di Gateway API.
`$context.identity.clientCert.clientCertPem`	Il certificato PEM-encoded client presentato dal client durante l'autenticazione TLS reciproca. Presente quando un client accede a un'API utilizzando un nome di dominio personalizzato che ha attivato l'autenticazione TLS reciproca.
`$context.identity.clientCert.subjectDN`	Nome distinto dell'oggetto del certificato presentato da un client. Presente quando un client accede a un'API utilizzando un nome di dominio personalizzato che ha attivato l'autenticazione TLS reciproca.
`$context.identity.clientCert.issuerDN`	Nome distinto dell'approvatore del certificato presentato da un cliente. Presente quando un client accede a un'API utilizzando un nome di dominio personalizzato che ha attivato l'autenticazione TLS reciproca.
`$context.identity.clientCert.serialNumber`	Il numero di serie del certificato. Presente quando un client accede a un'API utilizzando un nome di dominio personalizzato che ha attivato l'autenticazione TLS reciproca.
`$context.identity.clientCert.validity.notBefore`	La data prima della quale il certificato non è valido. Presente quando un client accede a un'API utilizzando un nome di dominio personalizzato che ha attivato l'autenticazione TLS reciproca.

Parametro	Description
`$context.identity.clientCert.validity.notAfter`	La data dopo la quale il certificato non è valido. Presente quando un client accede a un'API utilizzando un nome di dominio personalizzato che ha attivato l'autenticazione TLS reciproca.
`$context.identity.vpcId`	L'ID VPC del VPC da cui proviene la richiesta all'endpoint Gateway API.
`$context.identity.vpceId`	L'ID endpoint VPC dell'endpoint VPC da cui proviene la richiesta all'endpoint Gateway API. Presente solo quando si dispone di un'API privata.
`$context.identity.user`	Identificatore dell'entità principale dell'utente che sarà autorizzato per l'accesso alle risorse. Supportato per risorse che utilizzano l'autorizzazione IAM.
`$context.identity.userAgent`	Intestazione `User-Agent` del chiamante API.
`$context.identity.userArn`	Amazon Resource Name (ARN) dell'utente valido identificato dopo l'autenticazione. Per ulteriori informazioni, consulta https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html.
`$context.integration.error`	Il messaggio di errore restituito da un'integrazione.
`$context.integration.integrationStatus`	Per l'integrazione del proxy Lambda, il codice di stato restituito dal codice della funzione Lambda di backend AWS Lambda, non dal codice della funzione Lambda.
`$context.integration.latency`	Latenza di integrazione in ms. Equivalente a `$context.integrationLatency` .

Parametro	Description
`$context.integration.requestId`	L'ID della AWS richiesta dell'endpoint. Equivalente a `$context.awsEndpointRequestId` .
`$context.integration.responseTransferMode`	La modalità di trasferimento delle risposte della tua integrazione. Ciò può essere `BUFFERED` o `STREAMED`.
`$context.integration.status`	Il codice di stato restituito da un'integrazione. Per le integrazioni proxy Lambda, questo è il codice di stato restituito dal codice della funzione Lambda.
`$context.integration.timeToAllHeaders`	Il tempo che intercorre tra il momento in cui API Gateway stabilisce la connessione di integrazione e il momento in cui riceve tutte le intestazioni di risposta all'integrazione dal client.
`$context.integration.timeToFirstContent`	Il tempo che intercorre tra il momento in cui API Gateway stabilisce la connessione di integrazione e il momento in cui riceve i primi byte di contenuto.
`$context.integrationLatency`	Latenza di integrazione in ms.
`$context.integrationStatus`	Per l'integrazione del proxy Lambda, questo parametro rappresenta il codice di stato restituito dal codice della funzione Lambda di AWS Lambda backend e non dal codice della funzione Lambda.
`$context.isCanaryRequest`	Restituisce `true` se la richiesta era indirizzata al canary e `false` se la richiesta non era indirizzata al canary. Presente solo quando è abilitato un canary.

Parametro	Description
`$context.path`	Percorso della richiesta. Ad esempio, per un URL di richiesta non proxy `https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/root/child` , il valore di `$context.path` è `/{stage}/root/child` .
`$context.protocol`	Protocollo della richiesta, ad esempi, `HTTP/1.1`. **Note** Le API API Gateway possono accettare HTTP/2 richieste, ma API Gateway invia le richieste alle integrazioni di backend utilizzando. HTTP/1.1 Di conseguenza, il protocollo di richiesta viene registrato HTTP/1.1 anche se un client invia una richiesta che utilizza. HTTP/2
`$context.requestId`	L'ID della richiesta. I client possono sovrascrivere questo ID di richiesta. Utilizza `$context.extendedRequestId` per un ID di richiesta univoco generato da API Gateway.
`$context.requestOverride.header.` *`header_name`*	Sovrascrittura intestazione della richiesta. Se definito, questo parametro contiene le intestazioni da utilizzare al posto delle intestazioni HTTP che sono definite nel riquadro Integration Request (Richiesta di integrazione). Per ulteriori informazioni, consulta Sostituzione dei parametri della richiesta e della risposta e dei codici di stato per REST API in Gateway API.

Parametro	Description
`$context.requestOverride.path. path_name`	Sovrascrittura percorso della richiesta. Se definito, questo parametro contiene il percorso della richiesta da utilizzare al posto dei parametri di percorso URL che sono definiti nel riquadro Integration Request (Richiesta di integrazione). Per ulteriori informazioni, consulta Sostituzione dei parametri della richiesta e della risposta e dei codici di stato per REST API in Gateway API.
`$context.requestOverride.querystring. querystring_name`	Sovrascrittura stringa di query della richiesta. Se definito, questo parametro contiene la stringa di query della richiesta da utilizzare al posto dei URL Query String Parameters (Parametri stringa di query URL) che sono definiti nel riquadro Integration Request (Richiesta di integrazione). Per ulteriori informazioni, consulta Sostituzione dei parametri della richiesta e della risposta e dei codici di stato per REST API in Gateway API.
`$context.responseLatency`	Latenza della risposta in ms.
`$context.responseLength`	Lunghezza del payload della risposta in byte.
`$context.responseOverride.header. header_name`	Sovrascrittura intestazione della risposta. Se definito, questo parametro contiene l'intestazione da restituire al posto della Response header (Intestazione di risposta) che è definita come la Default mapping (mappatura predefinita) nel riquadro Integration Response (Risposta integrazione). Per ulteriori informazioni, consulta Sostituzione dei parametri della richiesta e della risposta e dei codici di stato per REST API in Gateway API.

Parametro	Description
`$context.responseOverride.status`	Sovrascrittura codice di stato della risposta. Se definito, questo parametro contiene il codice di stato da restituire al posto di Method response status (Stato risposta metodo) che è definito come la Default mapping (Mappatura predefinita) nel riquadro Integration Response (Risposta integrazione). Per ulteriori informazioni, consulta Sostituzione dei parametri della richiesta e della risposta e dei codici di stato per REST API in Gateway API.
`$context.requestTime`	Ora della richiesta in formato CLF (`dd/MMM/yyyy:HH:mm:ss +-hhmm`).
`$context.requestTimeEpoch`	L'ora della richiesta in formato epoca (Unix epoch) in millisecondi.
`$context.resourceId`	Identificatore assegnato da API Gateway alla risorsa.
`$context.resourcePath`	Percorso della risorsa. Ad esempio, per l'URI della richiesta non proxy di `https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/root/child` , il valore di `$context.resourcePath` è `/root/child` . Per ulteriori informazioni, consulta Tutorial: creazione di una REST API con un'integrazione non proxy HTTP.
`$context.stage`	La fase di distribuzione della richiesta API (ad esempio `Beta` o `Prod`).
`$context.status`	Stato della risposta del metodo.
`$context.tlsVersion`	La versione TLS negoziata durante l'handshake TLS tra il client e API Gateway.

Parametro	Description
`$context.waf.error`	Il messaggio di errore restituito da. AWS WAF
`$context.waf.latency`	La AWS WAF latenza in ms.
`$context.waf.status`	Il codice di stato restituito da AWS WAF.
`$context.xrayTraceId`	L'ID di traccia per la X-Ray traccia. Per ulteriori informazioni, consulta the section called “Configurare AWS X-Ray”.
`$context.wafResponseCode`	La risposta ricevuta da AWS WAF: `WAF_ALLOW` o `WAF_BLOCK` . Non verrà impostata se la fase non è associata a un ACL Web. Per ulteriori informazioni, consulta the section called “AWS WAF”.
`$context.webaclArn`	ARN completo della lista di controllo accessi Web usata per stabilire se consentire o bloccare la richiesta. Non verrà impostata se la fase non è associata a un ACL Web. Per ulteriori informazioni, consulta the section called “AWS WAF”.

Traccia delle richieste degli utenti alle REST API utilizzando X-Ray in Gateway API

Puoi utilizzare AWS X-Ray per monitorare e analizzare le richieste degli utenti durante il passaggio attraverso le API REST di Amazon API Gateway verso i servizi sottostanti. API Gateway supporta il monitoraggio attraverso X-Ray per tutti i tipi di endpoint API REST di API Gateway: regionale, ottimizzato per l’edge e privato. Puoi usare X-Ray con Amazon API Gateway in tutte le Regioni AWS in cui X-Ray è disponibile.

Poiché X-Ray offre una panoramica completa di un'intera richiesta, puoi analizzare le latenze nelle API e i relativi servizi di back-end. È possibile utilizzare una mappa del servizio di X-Ray per visualizzare la latenza di un'intera richiesta e quella dei servizi downstream integrati con X-Ray.

Inoltre, puoi configurare le regole di campionamento per comunicare a X-Ray quali richieste registrare e a quale frequenza in base alle policy specificate.

Se chiami un'API di API Gateway da un servizio già tracciato, API Gateway passa subito attraverso il monitoraggio, anche se X-Ray non è abilitato sull'API.

È possibile abilitare X-Ray per una fase API utilizzando la console API Gateway o utilizzando l'API o l'interfaccia a riga di comando di API Gateway.

Argomenti

- Configurazione AWS X-Ray con API Gateway REST APIs
- Utilizzo AWS X-Ray mappe dei servizi e viste di tracciamento con API Gateway
- Configurazione delle regole AWS X-Ray di campionamento per API Gateway APIs
- AWS X-Ray tracce per Amazon API Gateway APIs

Configurazione AWS X-Ray con API Gateway REST APIs

In questa sezione puoi trovare informazioni dettagliate su come configurare AWS X-Raycon API Gateway REST APIs.

Argomenti

- Modalità di monitoraggio di X-Ray per API Gateway
- Autorizzazioni per il monitoraggio di X-Ray
- Abilitazione del monitoraggio tramite X-Ray nella console API Gateway
- Abilitazione del AWS X-Ray tracciamento utilizzando l'API Gateway CLI

Modalità di monitoraggio di X-Ray per API Gateway

Il percorso di una richiesta nell'applicazione viene tracciato mediante un ID traccia. Una traccia raccoglie tutti i segmenti generati da una singola richiesta, in genere una richiesta HTTP `GET` o `POST`.

Per un'API di API Gateway sono disponibili due modalità di monitoraggio:

- Passiva: questa è l'impostazione predefinita se non hai abilitato il monitoraggio X-Ray su una fase API. Questo approccio significa che l'API di API Gateway viene tracciata solo se X-Ray è stato abilitato in un servizio upstream.

- Attiva: quando per una fase API di API Gateway si sceglie questa impostazione, API Gateway esegue automaticamente il campionamento delle richieste di invocazione dell'API in base all'algoritmo di campionamento specificato da X-Ray.

  Quando su una fase il monitoraggio è abilitato in modalità attiva, API Gateway crea un ruolo collegato ai servizi nel tuo account se questo non è già esistente. Il ruolo è denominato `AWSServiceRoleForAPIGateway` e sarà collegato alla policy gestita `APIGatewayServiceRolePolicy`. Per ulteriori informazioni sui ruoli collegati ai servizi, consulta Utilizzo dei ruoli collegati ai servizi.

  > **Note**
  >
  > X-Ray applica un algoritmo di campionamento per garantire che il monitoraggio avvenga in modo efficiente, continuando allo stesso tempo a fornire un campione rappresentativo delle richieste ricevute dall'API. L'algoritmo di campionamento di default corrisponde a una richiesta al secondo, con il 5% di richieste campionate oltre tale limite.

Puoi modificare la modalità di tracciamento per la tua API utilizzando la console di gestione API Gateway, l'API Gateway CLI o AWS un SDK.

Autorizzazioni per il monitoraggio di X-Ray

Quando su una fase è abilitato il monitoraggio tramite X-Ray, API Gateway crea un ruolo collegato ai servizi nel tuo account se questo non è già esistente. Il ruolo è denominato `AWSServiceRoleForAPIGateway` e sarà collegato alla policy gestita `APIGatewayServiceRolePolicy`. Per ulteriori informazioni sui ruoli collegati ai servizi, consulta Utilizzo dei ruoli collegati ai servizi.

Abilitazione del monitoraggio tramite X-Ray nella console API Gateway

Puoi utilizzare la console di Amazon API Gateway per abilitare il monitoraggio in modalità attiva in una fase API.

In queste fasi si presuppone che l'API sia già stata distribuita a una fase.

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli l'API, quindi nel pannello di navigazione principale, seleziona Fasi.
3. Nel riquadro Fasi scegli una fase.

4. Nella sezione Log e tracciamento scegli Modifica.
5. Per abilitare il tracciamento X-Ray seleziona Tracciamento X-Ray per attivarlo.
6. Scegli Save changes (Salva modifiche).

Una volta abilitato X-Ray per la fase API, è possibile utilizzare la console di gestione di X-Ray per visualizzare il monitoraggio e le mappe di servizio.

Abilitazione del AWS X-Ray tracciamento utilizzando l'API Gateway CLI

Il comando create-stage seguente crea una fase con tracciamento X-Ray attivo:

```
aws apigateway create-stage \
    --rest-api-id rest-api-id \
    --stage-name stage-name \
    --deployment-id deployment-id \
    --region region \
    --tracing-enabled=true
```

L'output sarà simile al seguente:

```
{
    "tracingEnabled": true,
    "stageName": stage-name,
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": deployment-id,
    "lastUpdatedDate": 1533849811,
    "createdDate": 1533849811,
    "methodSettings": {}
}
```

Il comando create-stage seguente crea una fase senza tracciamento X-Ray attivo:

```
aws apigateway create-stage \
    --rest-api-id rest-api-id \
    --stage-name stage-name \
    --deployment-id deployment-id \
    --region region \
    --tracing-enabled=false
```

L'output sarà simile al seguente:

```
{
    "tracingEnabled": false,
    "stageName": stage-name,
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": deployment-id,
    "lastUpdatedDate": 1533849811,
    "createdDate": 1533849811,
    "methodSettings": {}
}
```

Il comando update-stage seguente attiva il tracciamento X-Ray per un'API implementata:

```
aws apigateway update-stage \
    --rest-api-id rest-api-id \
    --stage-name stage-name \
    --patch-operations op=replace,path=/tracingEnabled,value=true
```

Il comando update-stage seguente disattiva il tracciamento X-Ray per un'API implementata:

```
aws apigateway update-stage \
    --rest-api-id rest-api-id \
    --stage-name stage-name \
    --region region \
    --patch-operations op=replace,path=/tracingEnabled,value=false
```

L'output sarà simile al seguente:

```
{
    "tracingEnabled": false,
    "stageName": stage-name,
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": deployment-id,
    "lastUpdatedDate": 1533850033,
    "createdDate": 1533849811,
    "methodSettings": {}
}
```

Una volta abilitato X-Ray per la fase API, utilizzare l'interfaccia a riga di comando di X-Ray per recuperare le informazioni sul monitoraggio. Per ulteriori informazioni, vedere Utilizzo dell'API X-Ray con la CLI AWS.

Utilizzo AWS X-Ray mappe dei servizi e viste di tracciamento con API Gateway

In questa sezione sono disponibili informazioni dettagliate sull'utilizzo delle mappe di servizio e delle viste di monitoraggio di AWS X-Ray con API Gateway.

Argomenti

- Esempio X-Ray di mappa dei servizi
- Esempio di visualizzazione della X-Ray traccia

Esempio X-Ray di mappa dei servizi

AWS X-Ray le mappe dei servizi mostrano informazioni sull'API e su tutti i relativi servizi a valle. Quando X-Ray è abilitato per una fase API in API Gateway, vedrai un nodo nella mappa dei servizi contenente informazioni sul tempo complessivo trascorso nel servizio API Gateway. Puoi ottenere informazioni dettagliate sullo stato della risposta e un istogramma del tempo di risposta dell'API nel periodo di tempo selezionato. Per le API che si integrano con AWS servizi come AWS Lambda Amazon DynamoDB, vedrai più nodi che forniscono metriche prestazionali relative a tali servizi. C'è una mappa di servizio per ogni fase API.

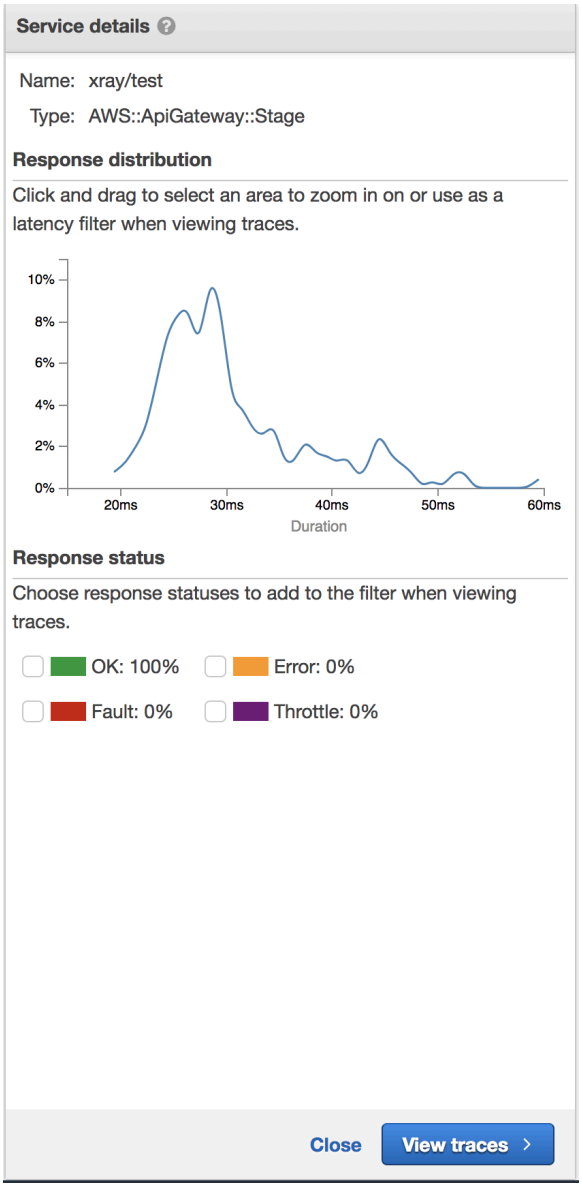
L'esempio seguente mostra una mappa di servizio per la fase `test` di un'API denominata `xray`. Quest'API dispone di due integrazioni Lambda. I nodi rappresentano il servizio Gateway API e le due funzioni Lambda.

Per una spiegazione dettagliata della struttura della mappa dei servizi, consulta Utilizzare la mappa di tracciamento. X-Ray



La mappa dei servizi può essere ingrandita per mostrare una vista della traccia della fase API. La traccia visualizza informazioni approfondite riguardanti l'API, rappresentate come segmenti e segmenti secondari. Ad esempio, il monitoraggio della mappa di servizio mostrata sopra include i segmenti del servizio Lambda e della funzione Lambda. Per ulteriori informazioni, vedere AWS Lambda e AWS X-Ray.

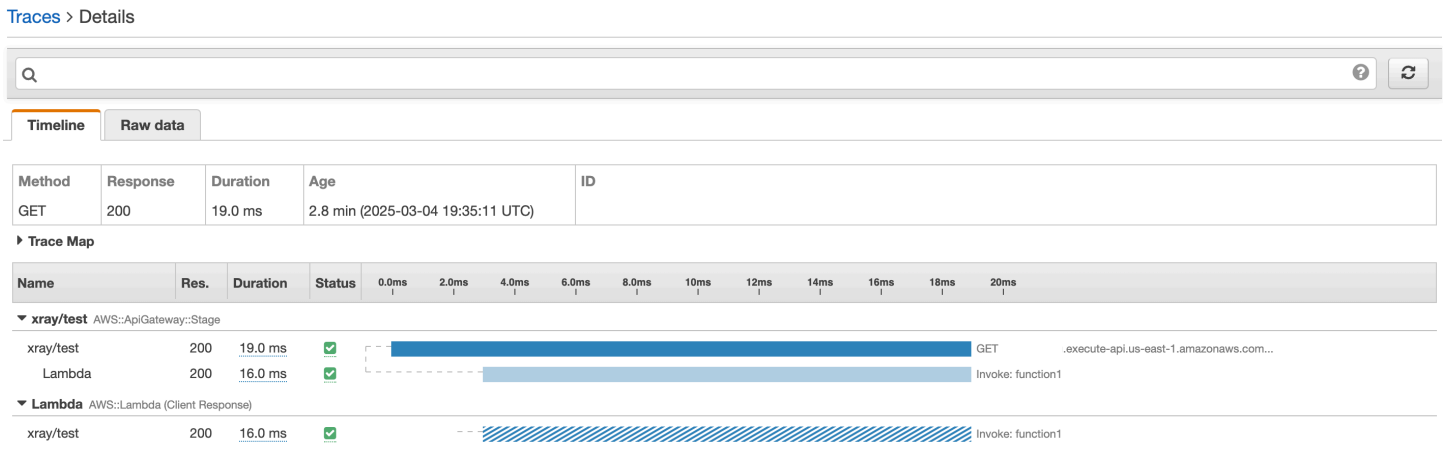
Se scegli un nodo o un edge su una mappa dei X-Ray servizi, la X-Ray console mostra un istogramma di distribuzione della latenza. Puoi utilizzare un istogramma della latenza per visualizzare il tempo impiegato da un servizio per completare le richieste. Di seguito è riportato un istogramma della fase dell'API Gateway denominata `xray/test` nella precedente mappa di servizio. Per una spiegazione dettagliata degli istogrammi di distribuzione della latenza, consulta Use Latency Histograms.



Esempio di visualizzazione della X-Ray traccia

Il seguente diagramma mostra una vista del monitoraggio generato per l'API di esempio descritta in precedenza, con una funzione di backend Lambda. Viene mostrata una richiesta metodo API andata a buon fine con codice di risposta pari a 200.

Per una spiegazione dettagliata delle viste di traccia, consulta View traces and trace details.



Configurazione delle regole AWS X-Ray di campionamento per API Gateway APIs

Puoi utilizzare la AWS X-Ray console o l'SDK per configurare le regole di campionamento per la tua API Amazon API Gateway. Una regola di campionamento specifica le richieste che X-Ray deve registrare per l'API. Personalizzando le regole di campionamento è possibile controllare la quantità di dati da registrare e modificare immediatamente il campionamento senza dover cambiare o ridistribuire il codice.

Prima di specificare le regole di campionamento di X-Ray, leggere i seguenti argomenti nella Guida per gli sviluppatori di X-Ray:

- Configurazione delle regole di campionamento
- Utilizzo delle regole di campionamento con l'API X-Ray

Argomenti

- Valori delle opzioni della regola di campionamento a raggi X per API Gateway APIs
- Esempi di regole di campionamento di X-Ray

Valori delle opzioni della regola di campionamento a raggi X per API Gateway APIs

Le seguenti opzioni di campionamento di X-Ray sono rilevanti per API Gateway. I valori di stringa possono utilizzare caratteri jolly per corrispondere a un solo carattere (?) o a zero o più caratteri (*). Per ulteriori informazioni, compresa una spiegazione dettagliata di come vengono utilizzate le impostazioni del reservoir e della frequenza, consulta Configure sampling rules.

- Nome regola (stringa): un nome univoco per la regola.

- Priorità (numero intero compreso tra 1 e 9999): la priorità della regola di campionamento. I servizi valutano le regole in ordine crescente di priorità e prendono una decisione sul campionamento in base alla prima regola corrispondente.
- Riserva (numero intero non negativo): un numero fisso di richieste che rispettano il filtro da analizzare ogni secondo, prima di applicare la percentuale fissa. Il reservoir non viene utilizzato direttamente dai servizi, ma si applica a tutti i servizi che utilizzano la regola nel loro complesso.
- Percentuale (numero tra 0 e 100): la percentuale di richieste che rispettano il filtro da analizzare dopo l'esaurimento della riserva.
- Nome servizio (stringa): nome della fase API, nel formato ***{api-name}*/*{stage-name}***. Ad esempio, se si dovesse distribuire l'API di PetStoreesempio in una fase denominata`test`, il valore del nome del servizio da specificare nella regola di campionamento sarebbe. **`pets/test`**
- Tipo servizio (stringa): per un'API di API Gateway è possibile specificare sia **`AWS::ApiGateway::Stage`** che **`AWS::ApiGateway::*`**.
- Host (stringa): il nome host ricavato dall'intestazione HTTP host. Per la corrispondenza a tutti i nome host, va impostato su **`*`**. Si può anche specificare un nome host completo o parziale di corrispondenza, ad esempio **`api.example.com`** o **`*.example.com`**.
- Resource ARN (ARN risorsa) (stringa): l'ARN della fase API, ad esempio **`arn:aws:apigateway:`*region*`::/restapis/`*api-id*`/stages/`*stage-name***.

  Il nome di fase si può ricavare dalla console, dall'interfaccia a riga di comando o dall'API di API Gateway. Per ulteriori informazioni sui formati degli ARN, consulta Riferimenti generali di Amazon Web Services.
- Metodo HTTP (stringa): il metodo da campionare, ad esempio **`GET`**.
- URL path (Percorso URL) (stringa): il percorso dell'URL della richiesta.
- (opzionale) Attributi (chiave e valore): intestazioni provenienti dalla richiesta HTTP originale, ad esempio **`Connection`**, **`Content-Length`** o **`Content-Type`**. Ogni valore dell'attributo può contenere fino a 32 caratteri.

Esempi di regole di campionamento di X-Ray

Esempio 1 di regola di campionamento

Questa regola campiona tutte le richieste `GET` per l'API `testxray` nella fase `test`.

- Nome regola — **`test-sampling`**
- Priorità — **`17`**

- Dimensioni riserva — **10**
- Percentuale fissa — **10**
- Nome servizio — **testxray/test**
- Tipo servizio — **AWS::ApiGateway::Stage**
- Metodo HTTP — **GET**
- ARN risorsa — *****
- Host — *****

Esempio 2 di regola di campionamento

Questa regola campiona tutte le richieste `testxray` per l'API nella fase `prod`.

- Nome regola — **prod-sampling**
- Priorità — **478**
- Dimensioni riserva — **1**
- Percentuale fissa — **60**
- Nome servizio — **testxray/prod**
- Tipo servizio — **AWS::ApiGateway::Stage**
- Metodo HTTP — *****
- ARN risorsa — *****
- Host — *****
- Attributi — **{}**

AWS X-Ray tracce per Amazon API Gateway APIs

Questa sezione illustra i segmenti di AWS X-Ray traccia, i sottosegmenti e altri campi di traccia per Amazon API Gateway. APIs

Prima di leggere questa sezione, rivedere i seguenti argomenti nella Guida per gli sviluppatori di X-Ray:

- Usa un Console di gestione AWS
- Documenti sui segmenti di X-Ray
- Concetti

Argomenti

- Esempi di oggetti di monitoraggio per un'API di API Gateway
- Comprendere la traccia

Esempi di oggetti di monitoraggio per un'API di API Gateway

Questa sezione illustra alcuni degli oggetti visibili in un monitoraggio di un'API di API Gateway.

Annotazioni

Le annotazioni possono essere visualizzate nei segmenti e nei segmenti secondari. Servono da espressioni di filtro nelle regole di campionamento per filtrare le tracce. Per ulteriori informazioni, consulta Configure sampling rules.

Di seguito è riportato un esempio di un oggetto `annotations`, in cui una fase API è identificata dall'ID dell'API e dal nome della fase API:

```
"annotations": {
    "aws:api_id": "a1b2c3d4e5",
    "aws:api_stage": "dev"
}
```

Per ulteriori informazioni sulle annotazioni, consulta X-Ray segment documents e scegli Annotations in X-Ray segment documents.

AWS dati relativi alle risorse

L'oggetto `aws` viene visualizzato solo nei segmenti. Di seguito è riportato l'esempio di un oggetto `aws` corrispondente alla regola di campionamento predefinita. Per una spiegazione dettagliata delle regole di campionamento, consulta Configure sampling rules.

```
"aws": {
    "xray": {
        "sampling_rule_name": "Default"
    },
    "api_gateway": {
        "account_id": "123412341234",
        "rest_api_id": "a1b2c3d4e5",
        "stage": "dev",
        "request_id": "a1b2c3d4-a1b2-a1b2-a1b2-a1b2c3d4e5f6"
    }
```

```
}
```

Per ulteriori informazioni sull'oggetto `aws`, consulta X-Ray segment documents e AWS resource data in X-Ray segment documents.

Comprendere la traccia

Di seguito è riportato un segmento di monitoraggio per una fase dell'API Gateway. Per una spiegazione dettagliata dei campi che compongono il segmento di traccia, consulta X-Ray segment documents.

```
        {
            "Document": {
                "id": "a1b2c3d4a1b2c3d4",
                "name": "testxray/dev",
                "start_time": 1533928226.229,
                "end_time": 1533928226.614,
                "metadata": {
                    "default": {
                        "extended_request_id": "abcde12345abcde=",
                        "request_id": "a1b2c3d4-a1b2-a1b2-a1b2-a1b2c3d4e5f6"
                    }
                },
                "http": {
                    "request": {
                        "url": "https://example.com/dev?
username=demo&message=hellofromdemo/",
                        "method": "GET",
                        "client_ip": "192.0.2.0",
                        "x_forwarded_for": true
                    },
                    "response": {
                        "status": 200,
                        "content_length": 0
                    }
                },
                "aws": {
                    "xray": {
                        "sampling_rule_name": "Default"
                    },
                    "api_gateway": {
                        "account_id": "123412341234",
                        "rest_api_id": "a1b2c3d4e5",
```

```
                        "stage": "dev",
                        "request_id": "a1b2c3d4-a1b2-a1b2-a1b2-a1b2c3d4e5f6"
                    }
                },
                "annotations": {
                    "aws:api_id": "a1b2c3d4e5",
                    "aws:api_stage": "dev"
                },
                "trace_id": "1-a1b2c3d4-a1b2c3d4a1b2c3d4a1b2c3d4",
                "origin": "AWS::ApiGateway::Stage",
                "resource_arn": "arn:aws:apigateway:us-east-1::/restapis/a1b2c3d4e5/
stages/dev",
                "subsegments": [
                    {
                        "id": "abcdefgh12345678",
                        "name": "Lambda",
                        "start_time": 1533928226.233,
                        "end_time": 1533928226.6130002,
                        "http": {
                            "request": {
                                "url": "https://example.com/2015-03-31/functions/
arn:aws:lambda:us-east-1:123412341234:function:xray123/invocations",
                                "method": "GET"
                            },
                            "response": {
                                "status": 200,
                                "content_length": 62
                            }
                        },
                        "aws": {
                            "function_name": "xray123",
                            "region": "us-east-1",
                            "operation": "Invoke",
                            "resource_names": [
                                "xray123"
                            ]
                        },
                        "namespace": "aws"
                    }
                ]
            },
            "Id": "a1b2c3d4a1b2c3d4"
        }
```

Portali API Gateway

Puoi utilizzare un portale API Gateway per condividere la tua documentazione APIs e quella sull'API con i consumatori di API in una pagina Web. Un portale contiene una serie di prodotti del portale. Ogni prodotto del portale è un raggruppamento logico di REST APIs e contiene la documentazione creata e pubblicata per gli utenti delle API. Le pagine dei prodotti all'interno di un portale contengono la documentazione personalizzata a livello di prodotto del portale. Le pagine degli endpoint REST del prodotto contengono la documentazione per ciascuno dei REST APIs e il percorso, il metodo e lo stadio di un'API REST. La combinazione di pagine di prodotto e pagine degli endpoint REST dei prodotti fornisce la documentazione completa per consentire ai consumatori di API di imparare come iniziare a utilizzare il tuo REST. APIs

Utilizza un portale per effettuare le seguenti operazioni:

- Utilizza API Gateway per l'intero ciclo di vita delle API, dalla creazione APIs alla documentazione e alla distribuzione.
- Condividi i prodotti del portale tra più account. AWS
- Personalizza e condividi una posizione centrale per i prodotti del portale e fornisci pagine di prodotto.

Considerazioni

Le seguenti considerazioni potrebbero influire sull'utilizzo dei portali API Gateway:

- È possibile creare solo portali per condividere REST. APIs
- I portali vengono creati a livello di AWS regione. Il portale può contenere solo REST APIs che si trovano nella stessa regione in cui è stato creato il portale.
- Puoi utilizzare il Console di gestione AWS AWS CLI, o solo AWS SDKs per creare e condividere i tuoi portali.
- Puoi controllare l'accesso al tuo portale solo utilizzando i pool di utenti di Amazon Cognito. Gli utenti del portale possono accedere direttamente tramite un pool di utenti oppure possono federarsi tramite un provider di identità (IdP) di terze parti. Se utilizzi un pool di utenti Amazon Cognito, non puoi impostare un segreto client per il tuo pool di utenti. Per ulteriori informazioni, consulta i pool di utenti di Amazon Cognito.

Prodotti del portale in API Gateway

Un prodotto del portale rappresenta un servizio o una funzionalità che si desidera condividere. Il prodotto del portale è una raccolta di endpoint REST e pagine di prodotto del prodotto. Gli endpoint REST del prodotto sono i punti di accesso al prodotto del portale e consistono nel percorso e nel metodo di un'API REST e nella fase in cui viene implementata. Le pagine dei prodotti sono la documentazione fornita per spiegare come i consumatori di API possono utilizzare gli endpoint del prodotto. Un prodotto del portale può contenere l'intera `Prod` fase come raccolta di endpoint REST del prodotto o solo la `GET /pets` risorsa distribuita sullo `Prod` stage come endpoint REST di un singolo prodotto.

Il prodotto del portale è personalizzabile. Puoi aggiungere documentazione personalizzata, rinominare gli endpoint REST del prodotto, riorganizzare l'ordine di visualizzazione, aggiungere nuove sezioni e condividere prodotti tra account. AWS Affinché le modifiche apportate al prodotto del portale abbiano effetto, è necessario ripubblicare tutti i portali che utilizzano il prodotto del portale.

Esempio di prodotto del portale sull'adozione di animali domestici

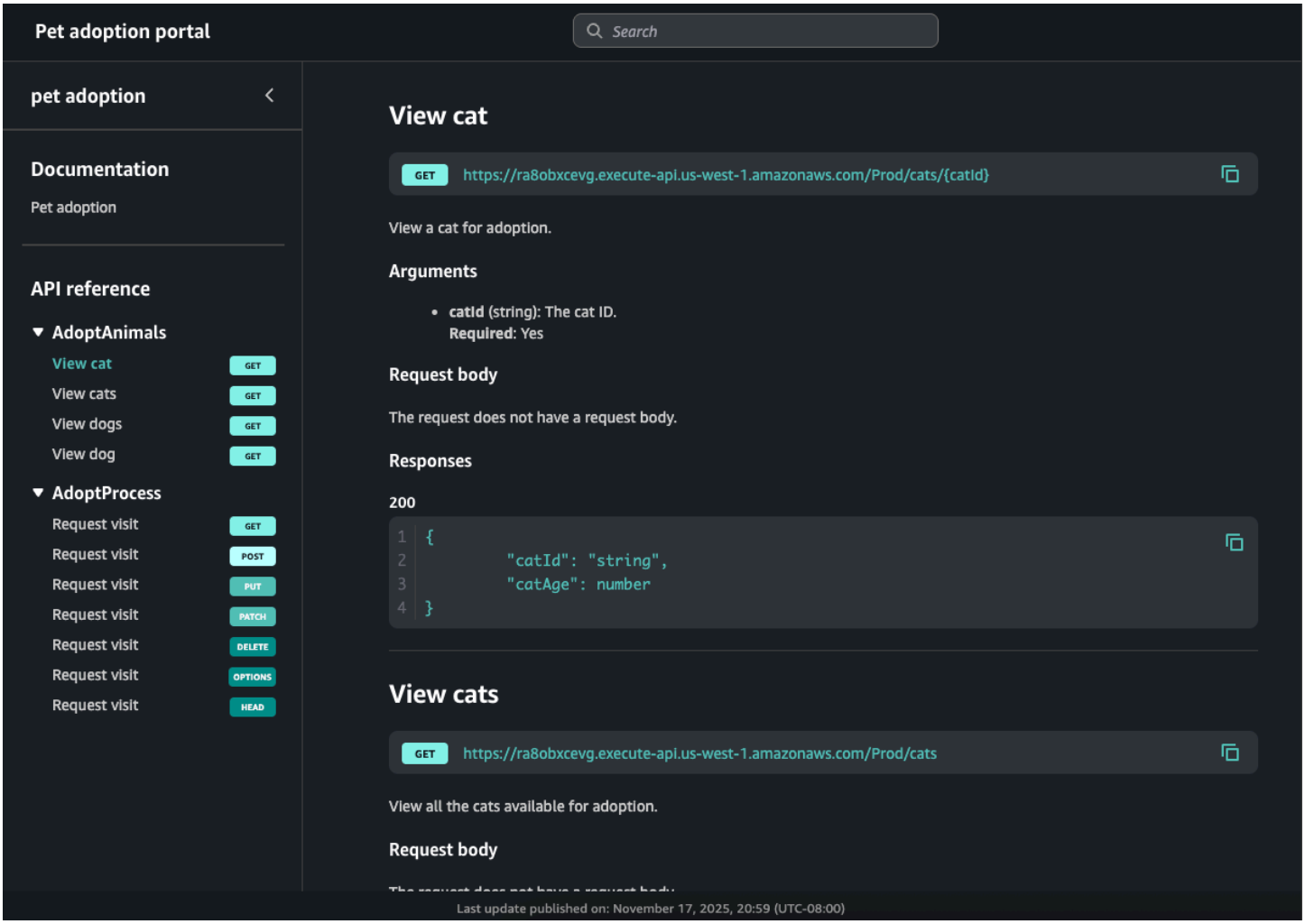
Ad esempio, potresti avere più API REST che rappresentano un servizio di adozione di animali domestici. È possibile utilizzare API Gateway per creare un prodotto del `pet adoption` portale. Questo prodotto del portale aiuterebbe i clienti a scoprire quali API utilizzare per incontrare e adottare animali domestici. Questo prodotto del portale utilizza le API REST già create dall'utente, ma consente di raggrupparle e organizzarle. Puoi anche fornire documentazione sui termini e le condizioni di utilizzo del tuo prodotto del portale per l'adozione di animali domestici e consentire ai clienti di provare le tue API. Queste informazioni sono tutte memorizzate nel prodotto del portale.

La tabella seguente mostra tre API che rappresentano un prodotto del portale per l'adozione di animali domestici e i nomi delle operazioni degli endpoint REST del prodotto corrispondenti e i nomi delle sezioni di pagina.

ID API REST	percorso e metodo dell'API REST	Fase dell'API REST	Nome operazione	Sezione della pagina
`kf5387miad`	`GET /dogs`	Prod	`View dogs`	`AdoptAnimals`

ID API REST	percorso e metodo dell'API REST	Fase dell'API REST	Nome operazione	Sezione della pagina
kf5387miad	GET /dogs/{dogId}	Prod	View dog	AdoptAnimals
ra8obxcevg	GET /cats	Prod	View cats	AdoptAnimals
ra8obxcevg	GET /cats/{catId}	Prod	View cat	AdoptAnimals
h0rpx9cm62	ANY /user/{userId}/{petId+}	Beta	Request visit	AdoptProcess

In questo esempio, due API REST `ra8obxcevg` e `kf5387miad`, sono raggruppate nella `AdoptAnimals` sezione. Il risultato di questa navigazione di ordinamento sarebbe simile al seguente in un portale:



Poiché il ANY metodo è un metodo generico, il portale mostra tutti i metodi HTTP supportati. Questo portale contiene anche la documentazione creata dal proprietario del prodotto del portale.

Fasi successive

Per iniziare a usare i prodotti del portale, puoi fare quanto segue:

- Per creare un prodotto del portale, consultathe section called “Crea un prodotto del portale”.
- Per maggiori informazioni sulla funzionalità Try it, consultathe section called “Abilita l'endpoint REST di prova per un prodotto nel tuo portale”.
- Per maggiori informazioni sulle pagine dei prodotti, consultathe section called “Crea una pagina di prodotto”.
- Per ulteriori informazioni sulla condivisione del prodotto del portale, consultathe section called “Condividi i prodotti del portale”.

Dopo aver creato un prodotto del portale, puoi pubblicarlo su un portale. Per ulteriori informazioni, consulta the section called “Crea un portale”.

Crea un prodotto portale in API Gateway

La procedura seguente mostra come creare un prodotto del portale. Un portale è una raccolta di prodotti del portale. Dopo aver creato il prodotto del portale, crei gli endpoint REST e le pagine di prodotto del prodotto. Per ulteriori informazioni sui prodotti del portale, consultathe section called “Prodotto del portale”.

Considerazioni

Le seguenti considerazioni potrebbero influire sul modo in cui crei un prodotto del portale:

- Il prodotto del portale può contenere API REST pubbliche e private. Le API private non sono supportate per la funzionalità try it e, di conseguenza, hanno una differenza visiva nel portale. In qualità di proprietario del portale, potrebbe essere necessario fornire una documentazione che spieghi questo aspetto.
- Se crei il prodotto del portale utilizzando gli AWS SDK AWS CLI o, il portale non avrà endpoint di prodotto o pagine di prodotto. Devi aggiungere queste risorse utilizzando la console AWS CLI or. Per informazioni su come creare un endpoint REST del prodotto, consultathe section called “Crea un endpoint REST del prodotto”. Per informazioni su come creare una pagina di prodotto, consultathe section called “Crea una pagina di prodotto”.

Crea un prodotto del portale

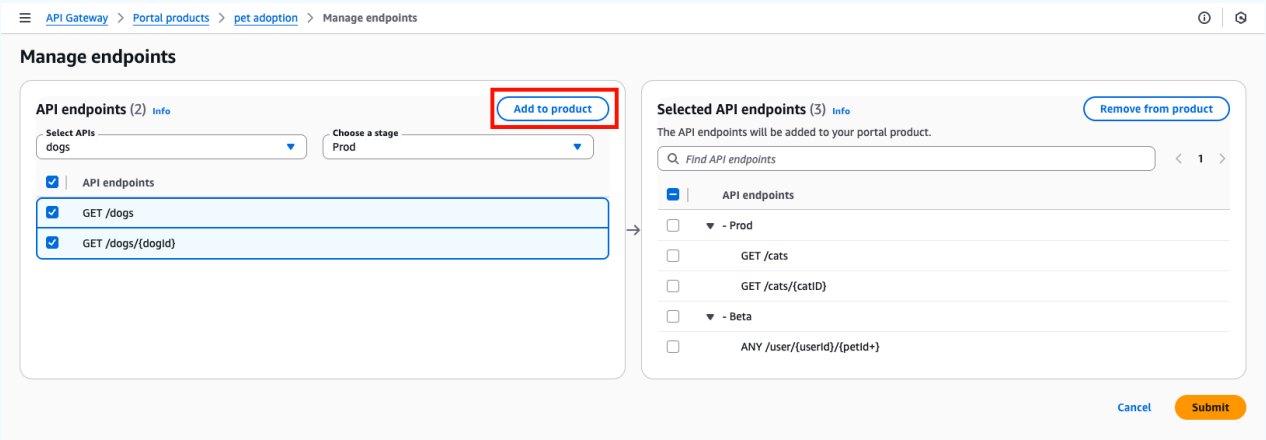
La procedura seguente mostra come creare un prodotto del portale.

Per creare un prodotto del portale

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel riquadro di navigazione principale, scegli i prodotti Portal.
3. Scegli Crea prodotto.
4. Per Nome prodotto, inserisci il nome del prodotto del portale.
5. Per la descrizione del prodotto, inserisci una descrizione.
6. Scegli Next (Successivo).

7. Per selezionare gli endpoint REST del prodotto, in Endpoint API scegli un'API, quindi scegli una fase.
8. Per aggiungere un endpoint agli endpoint REST del prodotto, seleziona l'endpoint API, quindi scegli Aggiungi al prodotto.

> **Note**
>
> Non scegliere Avanti senza prima aver scelto Aggiungi al prodotto.



L'endpoint API verrà visualizzato nell'elenco degli endpoint API selezionati.

9. Scegli Next (Successivo).
10. Controlla la selezione e scegli Crea prodotto.

Dopo aver creato il prodotto del portale utilizzando la console, tutte le pagine di prodotto e le pagine degli endpoint REST del prodotto sono bozze e non verranno visualizzate in un portale. Per rendere visibili ai consumatori le pagine dei prodotti e le pagine degli endpoint REST dei prodotti, devi aggiungere la bozza a una sezione. Se create il prodotto del portale utilizzando gli AWS SDK AWS CLI o, aggiungete la bozza alla sezione del comando. AWS CLI Indipendentemente dal modo in cui aggiungete la bozza a una sezione, dovete pubblicare il portale che utilizza i prodotti del portale affinché sia visibile ai consumatori.

Per aggiungere la bozza a una sezione della pagina

1. Le bozze sono elencate nella scheda Documentazione. Sono disponibili pagine di documentazione Draft per le pagine di prodotto e pagine di riferimento Draft API per le pagine degli endpoint REST del prodotto. Scegli le pagine di riferimento di Draft API.

2. Scegli una bozza di pagina di riferimento dell'API.

   Se non disponi di endpoint REST del prodotto, non avrai alcuna bozza di pagine di riferimento dell'API. Per informazioni su come creare un endpoint REST di un prodotto, consulta. the section called “Crea un endpoint REST del prodotto”
3. Scegli Modifica pagina.
4. In questa pagina, puoi sovrascrivere qualsiasi parte della documentazione API esistente o utilizzare la documentazione di API Gateway. Per consentire ai consumatori di visualizzare il contenuto della pagina dell'endpoint REST del prodotto, inserisci un nome nella sezione Nome della sezione. Se questo fosse l'`/pets-GET`endpoint, il nome della pagina potrebbe essere. `Pets`
5. Scegli Save changes (Salva modifiche).
6. Il nuovo nome di pagina che hai creato viene visualizzato nella sezione delle pagine di riferimento dell'API.

   Per consentire alla nuova pagina di essere visibile ai consumatori, devi comunque ripubblicare il tuo portale. Per ulteriori informazioni, consulta the section called “Pubblica un portale”.

Crea un endpoint REST del prodotto in API Gateway

Un endpoint REST del prodotto è un punto di accesso al prodotto del portale. Ogni endpoint è costituito dal percorso e dal metodo di un'API REST e dalla fase in cui viene distribuito. L'endpoint include tutta la documentazione che hai definito per la tua API REST o può essere aggiornato con documentazione personalizzata. Quando crei l'endpoint REST del prodotto, configuri le seguenti impostazioni:

- Il nome dell'operazione dell'endpoint REST del prodotto. Se l'API REST ha il percorso e il metodo`GET /pets`, in un portale, il nome è`/pets`. Utilizzando il nome dell'operazione, è possibile modificare il nome dell'endpoint REST del prodotto in`Pet summaries`. Ciò non ha alcun impatto sull'API esistente.
- La funzionalità di prova per l'endpoint del tuo prodotto. In questo modo i consumatori di API possono provare la vostra APIs applicazione nel vostro portale. Se crei un endpoint di prodotto per un'API privata, la funzionalità try it è impostata su. `UNAVAILABLE` Per ulteriori informazioni, consulta the section called “Abilita l'endpoint REST di prova per un prodotto nel tuo portale”.
- Se attualmente disponi di parti di documentazione definite per la tua API REST, puoi importare la documentazione o sovrascrivere la documentazione esistente e creare nuova documentazione a

livello di prodotto utilizzando Flavored Markdown. GitHub Eventuali modifiche alla documentazione non si propagano alle parti della documentazione dell'API API Gateway. Se importi le parti della documentazione corrente, API Gateway sincronizza le modifiche tra l'API e l'endpoint REST del prodotto.

Crea un endpoint REST del prodotto

La procedura seguente mostra come aggiungere un endpoint REST del prodotto a un prodotto del portale esistente. Per informazioni su come creare insieme un prodotto del portale e un endpoint REST del prodotto, consulta. the section called “Crea un prodotto del portale”

Per creare un endpoint REST di prodotto per un portale di prodotti

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel riquadro di navigazione principale, scegli i prodotti del portale.
3. Scegli un prodotto del portale.
4. Scegli la scheda Endpoint associati, quindi scegli Aggiungi endpoint.
5. Per selezionare gli endpoint REST del prodotto, in Endpoint API scegli un'API, quindi scegli una fase.
6. Per aggiungere un endpoint agli endpoint REST del prodotto, seleziona l'endpoint API, quindi scegli Aggiungi al prodotto.

   L'endpoint API verrà visualizzato nell'elenco degli endpoint API selezionati.
7. Scegli Invia.
8. La pagina dell'endpoint REST del prodotto è ora una bozza. Scegli le pagine di riferimento dell'API Draft, quindi scegli la pagina dell'endpoint REST del prodotto per finalizzare il contenuto della pagina.
9. Scegli Modifica pagina.
10. Per sovrascrivere la documentazione esistente di API Gateway, per Documentation source, attiva Sostituisci la documentazione e procedi come segue:

    a. Per Visualizza contenuto, scegli Create override.

    b. Per Nome dell'operazione, inserisci un nuovo nome dell'operazione.

    c. Per il corpo della pagina, inserisci la tua documentazione personalizzata.

    d. (Facoltativo) Per Endpoint, inserisci un endpoint personalizzato che appare nel portale.

11. Per consentire ai clienti di richiamare l'API nel portale, seleziona Prova la funzionalità. Per ulteriori informazioni, consulta the section called “Abilita l'endpoint REST di prova per un prodotto nel tuo portale”.
12. Per Nome della sezione, inserisci un nome o scegli una sezione esistente. Quando aggiungi l'endpoint REST del prodotto a una sezione, i consumatori possono visualizzare i tuoi contenuti su un portale pubblicato.
13. Scegli Save changes (Salva modifiche).

Per consentire alla nuova pagina di essere visibile ai consumatori, devi comunque ripubblicare il tuo portale. Per ulteriori informazioni, consulta the section called “Pubblica un portale”.

Crea una pagina di prodotto in API Gateway

Una pagina di prodotto aiuta i consumatori a comprendere e utilizzare i prodotti. La documentazione è a livello di prodotto, quindi tutti i portali che includono il prodotto dispongono di questa documentazione. API Gateway supporta due tipi di pagine di prodotto scritte in GitHub Flavored Markdown:

Pagina di panoramica

Quando crei il prodotto, API Gateway genera automaticamente una pagina di panoramica del prodotto in base alle informazioni fornite durante la creazione del prodotto. Questa pagina ha una formattazione aggiuntiva che non può essere modificata. È possibile modificare il testo o rimuovere la pagina dal prodotto del portale.

Pagine personalizzate

Una pagina personalizzata è una pagina scritta in Markdown. Puoi utilizzare questa sezione per i termini e le condizioni o per un tutorial su come utilizzare il tuo prodotto. Puoi modificare qualsiasi pagina personalizzata in una pagina di panoramica. Quando si modifica una pagina personalizzata in una pagina di panoramica, il contenuto della pagina viene formattato in modo da corrispondere alla pagina di panoramica.

Considerazioni

Le seguenti considerazioni potrebbero influire sull'utilizzo delle pagine dei prodotti:

- Questa documentazione non è a livello di API e non include metodi, risorse o parametri di richiesta. Per informazioni su come modificare la documentazione a livello di API per un prodotto, consulta. the section called “Aggiorna l'endpoint REST di un prodotto”
- Non puoi caricare direttamente un `.md` file sul tuo prodotto.
- Puoi modificare l'ordine delle pagine dei prodotti modificando l'elenco delle pagine nell'ordine di visualizzazione del prodotto del portale. Per ulteriori informazioni, consulta the section called “Aggiornare un prodotto del portale”.
- È necessario ripubblicare tutti i portali che includono il prodotto affinché le modifiche alla documentazione abbiano effetto.
- Se disponi di più portali che utilizzano lo stesso prodotto del portale, le pagine dei prodotti sono le stesse in tutti i portali. Per avere gli stessi endpoint REST del prodotto ma pagine di prodotto diverse, crea più prodotti.

Crea una pagina di prodotto

La procedura seguente mostra come creare una pagina di prodotto.

Per creare una pagina di prodotto

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel riquadro di navigazione principale, scegli i prodotti del portale.
3. Scegli un prodotto.
4. In Documentazione, scegli Crea pagina personalizzata.
5. Per Titolo della pagina, inserisci il titolo della pagina.
6. Per Body, inserisci la tua documentazione supplementare utilizzando GitHub Flavored Markdown.

   La sezione Anteprima mostra come i tuoi contenuti appaiono in un portale. Lo stile visivo finale potrebbe cambiare in base alle impostazioni del portale.
7. Scegli Crea pagina di prodotto.
8. La pagina del prodotto è ora una bozza. Scegli Bozze di documentazione, quindi scegli la pagina del prodotto per finalizzare il contenuto della pagina.
9. Scegli Modifica pagina.

10. Per Nome sezione, inserisci un nome o scegli una sezione esistente. Quando aggiungi la pagina del prodotto a una sezione, i consumatori possono visualizzare i tuoi contenuti su un portale pubblicato.
11. Scegli Save changes (Salva modifiche).

Per consentire alla nuova pagina di essere visibile ai consumatori, devi comunque ripubblicare il portale. Per ulteriori informazioni, consulta the section called “Pubblica un portale”.

Aggiornare un prodotto del portale in API Gateway

Quando aggiorni un prodotto del portale, puoi modificare il nome, la descrizione o l'ordine di visualizzazione degli endpoint REST e delle pagine dei prodotti. Per modificare l'ordine di visualizzazione, modifichi l'elenco delle pagine per le pagine dei prodotti e il nome della sezione per gli endpoint dei prodotti. Se si esegue questa operazione utilizzando AWS CLI o SDKs, si specifica l'ARN della pagina del prodotto o l'ARN dell'endpoint REST del prodotto. Devi ripubblicare tutti i portali che utilizzano i tuoi prodotti per consentire ai consumatori delle tue API di vedere le modifiche riflesse in un portale.

Aggiorna un prodotto del portale

La procedura seguente mostra come modificare l'elenco delle sezioni per riordinare gli endpoint REST del prodotto così come verranno visualizzati in un portale. In questa procedura, si presuppone che nel prodotto del portale siano presenti almeno due endpoint REST di prodotto.

Per aggiornare un prodotto del portale

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel riquadro di navigazione principale, scegli i prodotti del portale.
3. Scegli un prodotto.
4. Nella scheda Documentazione, per le pagine dei prodotti, scegli Gestisci.
5. Trascina e rilascia gli endpoint REST del prodotto per riordinarli. Puoi anche spostare gli endpoint REST e le pagine dei prodotti nella sezione delle bozze delle pagine. Le pagine delle bozze non saranno visibili nel portale.
6. (Facoltativo) Rinomina i nomi degli endpoint REST del prodotto o aggiungi una nuova sezione di riferimento dell'API. Queste modifiche non influiranno sul tuo REST. APIs
7. Scegli Save changes (Salva modifiche).

Aggiorna un endpoint REST del prodotto in API Gateway

Quando aggiorni un endpoint REST del prodotto, puoi modificare tutte le impostazioni dell'endpoint del prodotto ad eccezione del percorso e del metodo di un'API REST e della fase in cui viene distribuita. Qualsiasi modifica agli endpoint del prodotto viene aggiornata automaticamente, ma devi ripubblicare tutti i portali che utilizzano i tuoi prodotti per vedere le modifiche riflesse in un portale.

Aggiorna l'endpoint REST di un prodotto

La procedura seguente mostra come aggiornare l'endpoint REST di un prodotto per sovrascrivere la documentazione API corrente con una documentazione di riferimento API personalizzata.

Per aggiornare l'endpoint REST di un prodotto

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel riquadro di navigazione principale, scegli i prodotti del portale.
3. Scegli un prodotto.
4. Nella scheda Documentazione, in Pagine di riferimento dell'API, scegli il nome dell'endpoint REST di un prodotto, ad esempio /dogs - GET.
5. Per Anteprima, scegli Modifica pagina.
6. Per la fonte della documentazione, attiva Sostituisci la documentazione.
7. Per Visualizza contenuto, scegli Sovrascrivi il contenuto esistente.

   Se scegli Rimuovi tutto il contenuto, il contenuto viene rimosso dalla pagina, ma puoi scegliere Sostituisci il contenuto esistente per accedere nuovamente alla documentazione.
8. Nel corpo della pagina, inserisci la documentazione di riferimento dell'API personalizzata utilizzando GitHub Flavored Markdown.

   Questa documentazione non verrà propagata ad API Gateway.
9. Scegli Save changes (Salva modifiche).
10. È necessario ripubblicare tutti i portali che utilizzano questo prodotto del portale affinché le modifiche abbiano effetto.

Aggiornare una pagina di prodotto in API Gateway

Quando aggiorni una pagina di prodotto, puoi modificare tutte le impostazioni della pagina del prodotto. Qualsiasi modifica alla pagina del prodotto viene aggiornata automaticamente, ma devi ripubblicare tutti i portali che utilizzano i tuoi prodotti per vedere le modifiche riflesse in un portale.

Condividi i prodotti del portale in API Gateway

Puoi condividere un prodotto del portale Account AWS utilizzando AWS RAM. Quando si condivide un prodotto del portale, l'altro account può utilizzare il prodotto del portale nel proprio portale. Con i prodotti per portali condivisi, puoi creare un unico catalogo delle API della tua organizzazione e applicare gli standard di governance in tutto il tuo ecosistema di API. Allo stesso tempo, i prodotti del portale di condivisione offrono ai provider di API la flessibilità di sviluppare, testare e gestire le API nei propri account.

Considerazioni

Le seguenti considerazioni potrebbero influire sul modo in cui condividi le risorse del portale:

- Quando condividi il prodotto con un altro account, quell'account non può modificare alcuna proprietà della tua API REST. Ciò include gli endpoint di integrazione, la strategia di autorizzazione o la configurazione della fase.
- Quando si aggiunge il prodotto del portale di un altro account nel portale, il proprietario del prodotto del portale non può visualizzare o controllare altre proprietà del portale. Il proprietario del prodotto del portale sa solo che il prodotto viene utilizzato nel portale.
- I prodotti del portale API Gateway sono condivisi a Regione AWS livello.
- Puoi utilizzare una condivisione di risorse con più principali e, dopo aver creato la condivisione di risorse, aggiungervi altri principali. È consigliabile riutilizzare la condivisione di risorse, quando possibile.
- Se entrambi gli account fanno parte della stessa organizzazione che utilizza AWS Organizations, la condivisione delle risorse viene accettata automaticamente. È comunque necessario creare la condivisione di risorse tramite AWS RAM.
- Se entrambi gli account fanno parte della stessa organizzazione che utilizza AWS Organizations e la condivisione delle risorse all'interno dell'organizzazione è abilitata, a tutti i responsabili dell'organizzazione con cui si condivide viene automaticamente concesso l'accesso alle condivisioni di risorse. Non è necessario un invito e puoi evitare la condivisione di risorse.

- Se l'account con cui hai condiviso il prodotto non accetta la condivisione delle risorse entro 12 ore, devi condividere nuovamente la risorsa.
- Dopo aver creato la condivisione delle risorse, AWS RAM aggiorna la politica di condivisione del prodotto per impedire l'accesso ai principali utenti senza un accesso esplicito`allow`. Per ulteriori informazioni, consulta Determining whether a request is allowed or denied within an account nella Guida per l'utente di IAM.

  La politica delle risorse aggiornata sarà simile alla seguente:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "apigateway:GetProduct",
                "apigateway:ListEndpoints",
                "apigateway:ListPages",
                "apigateway:GetEndpoint",
                "apigateway:GetPage"
            ],
            "Resource": [
                "arn:aws:apigateway:us-east-1:111122223333:/portalproducts/product-id",
                "arn:aws:apigateway:us-east-1:111122223333:/portalproducts/product-id/*"
            ]
        }
    ]
}
```

  AWS RAM ha impedito ai committenti senza autorizzazione esplicita di aggiungere il prodotto al loro portale, aggiungendo quanto segue:

```
"StringNotEquals": { "aws:PrincipalAccount": "555555555555" }
```

Per sapere come condividere un prodotto, consulta. the section called “Condividi il prodotto del tuo portale con un proprietario del portale” Per sapere come aggiungere un prodotto condiviso con te al tuo portale, consultathe section called “Aggiungi un prodotto del portale condiviso al tuo portale”.

Condividi il prodotto del portale con un proprietario del portale in API Gateway

In qualità di proprietario del prodotto, puoi condividere il tuo prodotto con un proprietario del portale in un altro Account AWS utilizzo AWS RAM. Mantieni il pieno controllo di tutte le pagine dei prodotti e delle pagine degli endpoint REST dei prodotti.

Considerazioni

Le seguenti considerazioni potrebbero influire sul modo in cui condividi i prodotti del portale:

- Puoi condividere o annullare la condivisione dei prodotti del portale in qualsiasi momento. Se si annulla la condivisione di un prodotto del portale mentre viene utilizzato in un portale pubblicato, il proprietario del portale non potrà visualizzare il prodotto del portale, visualizzare gli eventuali aggiornamenti apportati o ripubblicare o visualizzare in anteprima il portale finché non rimuove il prodotto del portale.
- Puoi visualizzare i prodotti del portale che hai condiviso con altri account.
- Non puoi modificare i portali che contengono il tuo prodotto del portale, a meno che tu non crei il portale tu stesso.

Condividi il prodotto del tuo portale

La procedura seguente mostra come creare una condivisione di risorse.

Console di gestione AWS

Per utilizzare il Console di gestione AWS, vedere Creazione di una condivisione di risorse AWS RAM nella Guida AWS RAM per l'utente.

Per Seleziona il tipo di risorsa, scegli il prodotto API Gateway.

AWS CLI

Il seguente comando create-resource-share crea una condivisione di risorse per il tuo nome di dominio personalizzato privato. Il completamento dell'associazione tra la risorsa e il principale può richiedere alcuni minuti. Per i principali, fornisci un ID account o un ID organizzazione, ad esempio `arn:aws:organizations::123456789012:organization/o-1234abcd`. Puoi fornire più principali per la condivisione di risorse.

```
aws ram create-resource-share \
 --region us-west-2 \
 --name portal-product-resource-share \
```

```
 --permission-arns arn:aws:ram::aws:permission/
AWSRAMDefaultPermissionAPIGatewayDeveloperPortalProduct \
 --resource-arns arn:aws:apigateway:us-west-2:111122223333:/portalproducts/
p000000000 \
 --principals 222222222222
```

Per annullare la condivisione del prodotto del portale, usa AWS RAM per eliminare la condivisione delle risorse.

In qualsiasi momento, è possibile modificare la politica di condivisione dei prodotti per modificare quali committenti possono utilizzare i prodotti del portale nei rispettivi portali.

Per aggiornare la politica di condivisione dei prodotti

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel pannello di navigazione principale, scegli Prodotti Portal.
3. Scegli un prodotto.
4. Nella scheda Condivisione del prodotto, per Politica di condivisione del prodotto, scegli Gestisci politica di condivisione del prodotto.
5. Se non hai mai condiviso il tuo prodotto in precedenza, la politica di condivisione dei prodotti impedirà l'accesso ad altri account ai prodotti del tuo portale. È necessario aggiornare la politica per consentire l'accesso di determinati account al prodotto del portale.
6. Dopo aver aggiornato la politica di condivisione del prodotto, scegli Salva modifiche.

Aggiungi un prodotto del portale condiviso al tuo portale in API Gateway

In qualità di proprietario del portale, puoi utilizzare i prodotti del portale condivisi con te da altri account nel tuo portale. L'utente mantiene il pieno controllo del portale.

Considerazioni

Le seguenti considerazioni potrebbero influire sulla condivisione delle risorse del portale:

- È necessario accettare la condivisione delle risorse del prodotto dal proprietario del prodotto per utilizzare il prodotto nel portale. Se tu e il proprietario del prodotto appartenete alla stessa organizzazione, AWS RAM potete completare alcuni passaggi di condivisione al posto vostro.
- Se il proprietario del prodotto annulla la condivisione di un prodotto mentre si trova nel portale pubblicato, il prodotto sarà comunque visibile sul portale, ma non potrai pubblicare nuovamente

il portale finché non rimuovi il prodotto. È tua responsabilità in qualità di proprietario del portale rimuovere il prodotto dal portale, poiché il proprietario del prodotto non può farlo per te.

- È possibile visualizzare i prodotti del portale, le pagine dei prodotti e le pagine degli endpoint REST del prodotto, ma non è possibile modificare nessuna di queste risorse.
- Se un prodotto è condiviso con te, non puoi condividerlo con un altro account.

(Facoltativo) Accetta la condivisione delle risorse

Dopo che il proprietario del prodotto ha creato una condivisione di risorse, hai 12 ore per accettarla. Se fai parte della stessa organizzazione che utilizzi AWS Organizations come proprietario del prodotto, la condivisione viene accettata automaticamente. Se fai parte di un'organizzazione in cui è abilitata la condivisione automatica delle risorse, la risorsa viene condivisa automaticamente con te.

Console di gestione AWS

Per utilizzare il Console di gestione AWS, consulta Accettare e rifiutare gli inviti alla condivisione di risorse nella Guida per l'AWS RAM utente.

AWS CLI

Per trovare tutte le risorse condivise con te, usa il seguente comando https://docs.aws.amazon.com/cli/latest/reference/ram/get-resource-share-invitations.html get-resource-share-invitations:

```
aws ram get-resource-share-invitations \
    --region us-west-2
```

Utilizza l'ARN di condivisione risorse risultante per accettare l'invito alla condivisione di risorse. Il seguente comando accept-resource-share-invitation accetta la condivisione di risorse.

```
aws ram accept-resource-share-invitation \
    --resource-share-invitation-arn arn:aws:ram:us-west-2:123456789012:resource-
share-invitation/1e3477be-4a95-46b4-bbe0-c4001EXAMPLE \
    --region us-west-2
```

Aggiungi un prodotto condiviso al tuo portale

Dopo aver accettato la condivisione delle risorse, aggiungi il prodotto al tuo portale.

Console di gestione AWS

Per aggiungere un prodotto condiviso al tuo portale

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel pannello di navigazione principale, scegli Portals.
3. Scegli un portale.
4. Nella scheda Prodotti, per i prodotti del portale, scegli Aggiungi prodotti.
5. Aggiungi un prodotto al tuo portale. I prodotti condivisi vengono visualizzati come `shared` nell'elenco dei prodotti.
6. Scegli Save changes (Salva modifiche).

AWS CLI

Per trovare tutti i prodotti del portale condivisi con te, usa il seguente `get-portal-products` comando:

```
aws apigatewayv2 get-portal-products \
    --resource-owner OTHER_ACCOUNTS \
    --region us-west-2
```

Per aggiungere un prodotto del portale condiviso al tuo portale, usa il seguente `update-portal` comando:

```
aws apigateway update-portal \
    --included-portal-product-arns arn:aws:apigateway:us-west-2:111122223333:/
portalproducts/p000000000 \
    --region us-west-2
```

Abilita «prova» per un endpoint REST del prodotto API Gateway nel tuo portale

Usa try it per consentire a un utente di API di richiamare l'endpoint del prodotto dal tuo portale. Quando un utente di API utilizza try it, inserisce i parametri di richiesta del metodo e richiama l'endpoint del prodotto. Quindi, API Gateway richiama l'API sulla rete Internet pubblica e restituisce la risposta nella finestra di prova. Puoi utilizzare una pagina di prodotto personalizzata per fornire qualsiasi informazione aggiuntiva sui parametri richiesti per accedere all'API.

API Gateway utilizza i seguenti limiti per proteggere APIs:

- API Gateway consente solo 3 richieste al secondo alla tua API.
- API Gateway utilizza un limite di timeout integrato di 29000 ms. L'API effettiva potrebbe avere un limite di timeout più elevato, ma API Gateway non applica questo timeout quando un cliente utilizza try it.
- API Gateway limita il payload di risposta a 6 MB.

Considerazioni

Le seguenti considerazioni potrebbero influire sul modo in cui utilizzi try it:

- Try it non è supportato quando si visualizza l'anteprima di un portale.
- Prova che non è supportato per il REST APIs con le seguenti funzionalità:
  - Privato APIs
  - APIs che utilizzano il TLS reciproco
  - APIs che utilizzano certificati SSL privati o autofirmati

  In qualità di proprietario del portale, sei responsabile della comunicazione con i tuoi utenti di API. Il motivo del pulsante «perché provarlo» non è disponibile per i REST APIs che non sono supportati. API Gateway non te lo spiega.

Abilita «prova» per un endpoint REST del prodotto

La procedura seguente mostra come abilitare try it per un endpoint del prodotto.

Per sapere come utilizzarlo, provatelo in un portale, consultatethe section called “Usa un portale”.

Abilita «try it» per un endpoint REST del prodotto

1. Nel riquadro di navigazione principale, scegli i prodotti del portale.
2. Scegli un prodotto.
3. Nella scheda Documentazione, nelle pagine di riferimento delle API, scegli il nome dell'endpoint REST di un prodotto, ad esempio /dogs - GET.
4. Scegli Modifica pagina.
5. Per consentire ai clienti di richiamare la tua API nel portale, seleziona Prova la funzionalità.

6. Scegli Save changes (Salva modifiche).
7. È necessario ripubblicare tutti i portali che utilizzano questo prodotto del portale affinché le modifiche abbiano effetto.

Eliminare un prodotto del portale in API Gateway

Quando si elimina un prodotto del portale, questo non può essere recuperato.

Eliminare un prodotto del portale

La procedura seguente mostra come eliminare un prodotto del portale.

Per eliminare un prodotto del portale

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel riquadro di navigazione principale, scegli i prodotti del portale.
3. Scegli un prodotto del portale.
4. Scegli Elimina.
5. Conferma la scelta e seleziona Elimina.
6. È necessario ripubblicare tutti i portali che utilizzano questo prodotto del portale affinché le modifiche abbiano effetto.

Crea un portale in API Gateway

Un portale è una raccolta di prodotti. Sei tu a controllare l'autorizzazione, il marchio e la pubblicazione del tuo portale. Quando crei un portale, il relativo stato di pubblicazione è`Disabled`. Ciò significa che il portale non è individuabile sul Web. Per consentire ai consumatori di accedere al tuo portale, devi pubblicarlo. Prima di pubblicare il portale, puoi anche visualizzarlo in anteprima.

API Gateway supporta i portali sui seguenti browser:

- Firefox
- Google Chrome
- Microsoft Edge
- Safari

Scegli un nome di dominio

Per impostazione predefinita, l'URL del portale è ospitato nel seguente nome di dominio di proprietà di Amazon:

```
https://p-portalId.apigw-portal.us-east-1.on.aws
```

Per personalizzare il nome di dominio, puoi fornire un nome di dominio di tua proprietà e un certificato SSL per il dominio personalizzato. Il certificato deve essere gestito con AWS Certificate Manager(ACM) negli Stati Uniti orientali (Virginia settentrionale) ed essere supportato da Amazon. CloudFront Per ulteriori informazioni, consulta Requisiti per l'utilizzo dei SSL/TLS certificati con. CloudFront

Quando utilizzi un nome di dominio personalizzato, API Gateway crea una CloudFront distribuzione, protetta in transito con il tuo certificato ACM. API Gateway fornisce quindi un record di alias da aggiungere alla configurazione DNS. Il record di alias è il nome di dominio di proprietà di Amazon. Devi aggiungere questo record di alias per indirizzare il traffico verso la CloudFront distribuzione del tuo dominio personalizzato. Il nome di dominio fornito deve essere univoco e non essere già utilizzato dall'elenco degli alias di una distribuzione esistente CloudFront .

Se utilizzi il tuo nome di dominio, API Gateway utilizza la politica `TLS_1_2` di sicurezza per proteggere il tuo portale. Per ulteriori informazioni, consulta Protocolli e cifrari supportati tra i visualizzatori. Non puoi modificare questo valore.

Crea un portale

Utilizzi la console API Gateway per creare e configurare il tuo portale. Le seguenti procedure guidano l'utente attraverso tutti i passaggi richiesti. Puoi anche scegliere Vai alla revisione e crea alla fine della prima procedura per creare un portale vuoto.

La procedura seguente mostra come creare un portale. Innanzitutto, definisci i dettagli del tuo portale.

Per definire i dettagli del portale

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel pannello di navigazione principale, scegli Portali.
3. Selezionare Create portal (Crea portale).
4. In Dettagli del portale, per Nome del portale, inserisci un nome.

5. Per Descrizione del portale, inserisci una descrizione.
6. Per l'autorizzazione del portale, seleziona come autorizzare il portale.
   - Se è la prima volta che crei un portale, ti consigliamo di scegliere Nessuno. Se prevedi di aggiungere una produzione APIs al tuo portale, usa Richiedi autorizzazione.
   - Per richiedere agli utenti del portale di autenticarsi per utilizzare il portale, scegli Richiedi autorizzazione ed esegui le seguenti operazioni:
     a. Per il pool di utenti di Amazon Cognito, seleziona il tuo pool di utenti.

        Il pool di utenti deve trovarsi nella stessa regione del portale.
     b. Per l'ID client dell'applicazione, scegli l'ID client dell'app.

        Nel client dell'app, devi impostare l'URL di callback sull'URL predefinito del portale.
     c. Per il dominio del pool di utenti, inserisci il dominio del pool di utenti. Per ulteriori informazioni, vedere Configurazione di un dominio del pool di utenti.
7. Per la configurazione del dominio, seleziona il nome di dominio per il tuo portale.
   - Se è la prima volta che crei un portale, ti consigliamo di scegliere Dominio predefinito. Una volta pubblicato, il portale sarà disponibile all'indirizzo`https://p-`*`portalId`*`.apigw-portal.`*`us-east-1`*`.on.aws`.
   - Per fornire un nome di dominio personalizzato di tua proprietà, seleziona Nome di dominio personalizzato e procedi come segue:
     a. Per Nome di dominio, inserisci un nome di dominio completo.
     b. Per il certificato ACM, scegli un certificato ACM che copra il nome di dominio.

        Dovrai aggiungere il record di alias fornito da API Gateway al tuo record DNS, per servire il traffico proveniente dal tuo portale.
8. Per aggiungere un app monitor CloudWatch RUM al portale, per RUM app monitor, scegli un app monitor. Per ulteriori informazioni, consulta CloudWatch RUM.
9. Scegli Next (Successivo).

Dopo aver definito i dettagli del portale, aggiungi i prodotti del portale. È possibile modificare i prodotti del portale nel portale in qualsiasi momento, ma è necessario ripubblicare il portale affinché le modifiche abbiano effetto.

Per selezionare i prodotti del portale

1. Dall'elenco delle risorse, seleziona i prodotti da aggiungere al portale.
2. Scegli Next (Successivo).

Dopo aver selezionato i prodotti, personalizzi il design del portale. Per caricare un logo del portale, devi avere un'immagine caricata su Amazon S3. Puoi modificare questi valori in qualsiasi momento, ma devi ripubblicare il portale affinché le modifiche abbiano effetto.

(Facoltativo) Per personalizzare il portale

1. Per il logo di Portal, fornisci l'URI S3 per il tuo file.

   Sono `.jpg` supportati solo `.svg` o. `.png`
2. Per personalizzare il design del tuo portale, hai due opzioni:

   a. Puoi scegliere il tema Portal per applicare un tema cromatico predefinito al tuo portale.

   b. Puoi usare i selettori di colore per selezionare tutte le opzioni di progettazione del tuo portale.
3. Scegli Next (Successivo).

Dopo aver selezionato il design del portale, esamina le informazioni relative al portale.

Per rivedere il tuo portale

- Rivedi le impostazioni nella pagina di revisione e creazione. Quando si è soddisfatti delle impostazioni, selezionare Create (Crea).

Se utilizzi un pool di utenti Amazon Cognito per controllare l'accesso al tuo portale, devi impostare l'URL di callback sull'URL predefinito del portale nel client dell'app. Per ulteriori informazioni, consulta Impostazioni specifiche dell'applicazione con i client dell'app.

Il portale non sarà più disponibile per i consumatori dopo la sua creazione. È necessario pubblicare il portale per renderlo accessibile su Internet. Per ulteriori informazioni, consulta the section called “Pubblica un portale”.

Aggiornare un portale in API Gateway

Puoi aggiornare il tema, il logo o i prodotti del portale nel tuo portale. Quando aggiorni il portale, le modifiche non hanno effetto finché non lo ripubblichi.

Aggiorna un portale

La procedura seguente mostra come aggiornare il design di un portale.

Per aggiornare il design di un portale

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel pannello di navigazione principale, scegli Portali.
3. Scegli un portale.
4. Nella scheda Progettazione del portale, scegli Modifica.
5. Per aggiornare il logo del portale, scegli Aggiorna logo del portale. È possibile rimuovere il logo esistente o utilizzare un nuovo logo.
6. Apporta modifiche al tema o allo sfondo del design del tuo portale.
7. Scegli Save changes (Salva modifiche).
8. (Facoltativo) Scegli Genera anteprima del portale per vedere le modifiche.

   Dopo aver confermato le modifiche, pubblica il portale.
9. Scegli Pubblica portale.
10. Inserisci una descrizione per l'aggiornamento del portale e scegli Pubblica.

Visualizza l'anteprima di un portale in API Gateway

È possibile visualizzare in anteprima il portale prima di pubblicarlo. API Gateway crea un URL di breve durata che puoi utilizzare per visualizzare in anteprima la versione del tuo portale che i clienti vedranno una volta pubblicato il portale.

Considerazioni

Le seguenti considerazioni potrebbero influire sulla modalità di anteprima di un portale:

- Prova che non sia abilitato per il tuo portale di anteprima. Per ulteriori informazioni, consulta the section called “Abilita l'endpoint REST di prova per un prodotto nel tuo portale”.

Visualizzare l'anteprima di un portale

La procedura seguente mostra come visualizzare l'anteprima di un portale.

Per visualizzare l'anteprima di un portale

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel pannello di navigazione principale, scegli Portali.
3. Scegli un portale.
4. Scegli Genera anteprima del portale.

   Non aggiornare la pagina.
5. Dopo che API Gateway ha generato l'anteprima, scegli Apri anteprima del portale.

   L'anteprima del portale si aprirà in una nuova scheda.

Pubblica un portale in API Gateway

Per consentire agli utenti di API di accedere al portale, è necessario pubblicarlo. L'URL di un portale può essere scoperto da chiunque su Internet. Ti consigliamo di visualizzare in anteprima e proteggere il portale prima di pubblicarlo.

Considerazioni

API Gateway potrebbe impiegare alcuni minuti per pubblicare il portale. Puoi monitorare lo stato di pubblicazione nella console.

Pubblica un portale

La procedura seguente mostra come pubblicare un portale.

Per pubblicare un portale

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel pannello di navigazione principale, scegli Portali.
3. Scegli un portale.
4. Scegli Pubblica portale.

5. (Facoltativo) In Descrizione delle modifiche, inserisci una descrizione della modifica.

   Quando pubblicate un portale, vi consigliamo di fornire sempre una breve descrizione delle modifiche.

6. Seleziona Pubblica.

   API Gateway impiega alcuni minuti per completare la pubblicazione del portale. API Gateway fornisce un collegamento al portale quando è disponibile.

Per eliminare il portale, devi prima disabilitarlo. Per ulteriori informazioni, consulta the section called “Disabilita un portale”.

Usa un portale API Gateway

Dopo la pubblicazione, il portale è disponibile sul Web. Per utilizzare il portale, copia il nome di dominio predefinito del portale o il nome di dominio e inseriscilo in un browser. Se il portale utilizza un nome di dominio, il nome di dominio predefinito non può essere utilizzato per accedere al portale.

Usa un portale

La procedura seguente mostra come accedere a un portale e utilizzare try it per l'endpoint REST di un prodotto.

Per utilizzare un portale

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel pannello di navigazione principale, scegli Portali.
3. Scegli un portale.
4. Copia il nome di dominio predefinito del portale o il nome di dominio e inseriscilo in un browser.
5. Nella home page del portale, puoi cercare i prodotti del portale e scoprire le pagine dei prodotti. Utilizza la ricerca nella parte superiore del portale per cercare tutti i contenuti del portale. Utilizza la barra di ricerca sotto Prodotti per cercare i prodotti del portale.
6. Scegli un prodotto del portale.
7. Scegli un endpoint REST del prodotto.
8. Nella finestra Try it, inserisci i parametri per provare a chiamare la tua API.
9. Scegli Provalo. API Gateway richiamerà la tua API e mostrerà i risultati nella finestra.

Risolvi i problemi del tuo portale

Di seguito vengono forniti consigli per la risoluzione di errori e problemi che potrebbero verificarsi durante l'utilizzo di un portale.

Il sito della pagina Web del portale non può essere raggiunto

Se ricevi un messaggio di errore che indica che il sito della pagina Web del portale non è raggiungibile, assicurati di aver pubblicato il portale. Se hai utilizzato un nome di dominio, assicurati di aver aggiunto il record di alias fornito da API Gateway per indirizzare il traffico verso la CloudFront distribuzione del tuo dominio personalizzato.

Il tentativo non è riuscito

Se non riesci a utilizzare try it per un endpoint REST del prodotto, assicurati che lo stato try it sia`ACTIVE`. Inoltre, assicurati di aver inserito i parametri corretti per richiamare la tua API.

Le pagine non vengono visualizzate nel portale

Se hai creato una pagina di endpoint REST o una pagina di prodotto e questa non è visibile nel portale, assicurati che le pagine non siano bozze. Per verificarlo, procedi come segue:

Per verificare che le pagine non contengano bozze

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel riquadro di navigazione principale, scegli i prodotti del portale.
3. Scegli il prodotto del portale.
4. Nella scheda Documentazione, verifica che le tue pagine non si trovino nella sezione Bozze di documentazione o Pagine di riferimento Draft API, ma nelle pagine di documentazione personalizzata o nelle pagine di riferimento API.

Disabilitare un portale in API Gateway

Quando si disattiva un portale, il portale non è più disponibile sul Web. Dopo aver disattivato un portale, è possibile eliminarlo o ripubblicarlo se necessario.

Disabilitare un portale

La procedura seguente mostra come disattivare un portale.

Per disattivare un portale

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel pannello di navigazione principale, scegli Portali.
3. Scegli un portale.
4. Scegli Azioni, quindi scegli Disabilita portale.
5. Conferma la tua scelta e scegli Disattiva il portale.

   Lo stato di pubblicazione cambierà in Disabilitato.

Per rimuovere lo stato di disabilitazione del portale, pubblicate nuovamente il portale.

Eliminare un portale in API Gateway

Quando si elimina un portale, questo non può essere recuperato ed è necessario prima disabilitarlo. Se desideri rimuovere il portale dal Web, puoi disabilitarlo. Ciò consente di modificare il portale e ripubblicarlo in un secondo momento.

Eliminare un portale

La procedura seguente mostra come eliminare un portale.

Come eliminare un portale

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Nel pannello di navigazione principale, scegli Portali.
3. Scegli un portale.
4. Se il portale non è disabilitato, disattivalo scegliendo Azioni, Disattiva portale.

   La disattivazione del portale richiede alcuni minuti. È possibile monitorare lo stato di pubblicazione per vedere quando lo stato di pubblicazione del portale è disabilitato.
5. Dopo aver annullato la pubblicazione del portale, puoi eliminarlo. Per eliminare il portale, scegli Azioni, Elimina portale. Conferma la scelta e seleziona Elimina.

API HTTP di Gateway API

REST API e API HTTP sono entrambe prodotti dell'API RESTful. Le REST API supportano più funzionalità rispetto alle API HTTP, mentre le API HTTP sono progettate con funzionalità minime in modo che possano essere offerte a un prezzo inferiore. Per ulteriori informazioni, consulta the section called “Scelta tra REST API e API HTTP ”.

È possibile utilizzare le API HTTP per inviare richieste alle funzioni AWS Lambda o a qualsiasi endpoint HTTP instradabile. Ad esempio, puoi creare un'API HTTP che si integra con una funzione Lambda sul back-end. Quando un client chiama l'API, API Gateway invia la richiesta alla funzione Lambda e restituisce la risposta della funzione al client.

API HTTP supportano l'autorizzazione OpenID Connect e OAuth 2.0. Sono dotate di supporto integrato per CORS (cross-origin resource sharing) e le distribuzioni automatiche.

Puoi creare API HTTP utilizzando la Console di gestione AWS, la AWS CLI, le API, CloudFormation o gli SDK.

Argomenti

- Sviluppa HTTP APIs in API Gateway
- Pubblicazione di API HTTP che i clienti possono invocare
- Protezione delle API HTTP in Gateway API
- Monitora HTTP APIs nell'API Gateway
- Risoluzione dei problemi con le API HTTP in Gateway API

Sviluppa HTTP APIs in API Gateway

Questa sezione fornisce dettagli sulle funzionalità di API Gateway di cui hai bisogno durante lo sviluppo del tuo API Gateway APIs.

Durante lo sviluppo delle API di API Gateway è possibile impostare una serie di caratteristiche dell'API. Queste caratteristiche dipendono dal caso d'uso dell'API. Ad esempio, è possibile consentire a solo a determinati client di richiamare l'API oppure che questa sia disponibile per tutti. È possibile decidere che una chiamata API esegua una funzione Lambda, una query a un database o richiami un'applicazione.

Argomenti

- Creazione di un'API HTTP
- Crea percorsi per HTTP APIs in API Gateway
- Tipi di indirizzo IP per API HTTP in Gateway API
- Controlla e gestisci l'accesso a HTTP APIs in API Gateway
- Creazione delle integrazioni per API HTTP in Gateway API
- Configurazione di CORS per HTTP APIs in API Gateway
- Trasformazione delle richieste e delle risposte API per API HTTP in Gateway API
- Usa le definizioni OpenAPI per HTTP APIs in API Gateway

Creazione di un'API HTTP

Per creare un'API funzionale, è necessario disporre di almeno una route, integrazione, fase e distribuzione.

Gli esempi seguenti mostrano come creare un'API con un'integrazione AWS Lambda o HTTP, una route e una fase predefinita configurata per distribuire automaticamente le modifiche.

In questa guida si presuppone che tu abbia già familiarità con API Gateway e Lambda. Per una guida più dettagliata, consulta Inizia a usare.

Argomenti

- Crea un'API HTTP utilizzando il Console di gestione AWS
- Crea un'API HTTP utilizzando la AWS CLI

Crea un'API HTTP utilizzando il Console di gestione AWS

1. Apri la console API Gateway.
2. Seleziona Create API (Crea API).
3. In HTTP API, scegliere Build (Compila).
4. Scegliere Add integration (Aggiungi integrazione), quindi scegliere una funzione AWS Lambda o immettere un endpoint HTTP.
5. Per Name (Nome) immetti un nome per il TAG.
6. Scegliere Review and create (Rivedi e crea).

7. Seleziona Crea.

Ora la tua API è pronta per effettuare le chiamate. È possibile testare l'API inserendo il relativo URL di chiamata in un browser o utilizzando Curl.

```
curl https://api-id.execute-api.us-east-2.amazonaws.com
```

Crea un'API HTTP utilizzando la AWS CLI

È possibile utilizzare la creazione rapida per creare un'API con un'integrazione Lambda o HTTP, una route catch-all di default e una fase predefinita configurata per implementare automaticamente le modifiche. Il comando create-api seguente utilizza la funzione di creazione rapida per creare un'API che si integra con una funzione Lambda sul backend.

> **Note**
>
> Per richiamare un'integrazione Lambda, API Gateway deve disporre delle autorizzazioni necessarie. È possibile utilizzare una policy basata sulle risorse o un ruolo IAM per concedere le autorizzazioni di API Gateway per richiamare una funzione Lambda. Per ulteriori informazioni, consulta AWS Lambda Autorizzazioni nella Guida per gli AWS Lambda sviluppatori.

Example

```
aws apigatewayv2 create-api --name my-api --protocol-type HTTP --target
 arn:aws:lambda:us-east-2:123456789012:function:function-name
```

Ora la tua API è pronta per effettuare le chiamate. È possibile testare l'API inserendo il relativo URL di chiamata in un browser o utilizzando Curl.

```
curl https://api-id.execute-api.us-east-2.amazonaws.com
```

Crea percorsi per HTTP APIs in API Gateway

Esegui il routing delle richieste API in entrata dirette alle risorse di back-end. Le route sono costituite da due parti: un metodo HTTP e un percorso delle risorse, ad esempi, `GET /pets`. È possibile definire metodi HTTP specifici per il percorso. In alternativa, è possibile utilizzare il metodo ANY per

abbinare tutti i metodi non definiti per una risorsa. È possibile creare una route `$default` che funga da catch-all per le richieste che non corrispondono ad altre route.

> **Note**
>
> Gateway Amazon API decodifica i parametri di richiesta con codifica URL prima di passarli all'integrazione back-end.

Lavorare con le variabili di percorso

Nelle route delle API HTTP è possibile utilizzare delle variabili di percorso.

Ad esempio, il percorso `GET /pets/{petID}` cattura una richiesta `GET` inviata da un client `https://api-id.execute-api.us-east-2.amazonaws.com/pets/6`.

Una variabile di percorso greedy cattura tutte le risorse figlio di un percorso. Per creare una variabile di percorso greedy, aggiungere + al nome della variale, ad esempio `{proxy+}`. La variabile di percorso greedy deve trovarsi alla fine del percorso della risorsa.

Utilizzo dei parametri della stringa di query

Per impostazione predefinita, API Gateway invia i parametri della stringa di query all'integrazione back-end se sono inclusi in una richiesta a un oggetto API HTTP.

Ad esempio, quando un client invia una richiesta a `https://api-id.execute-api.us-east-2.amazonaws.com/pets?id=4&type=dog`, i parametri della stringa di query `?id=4&type=dog` vengono inviati all'integrazione.

Lavorare con il percorso **`$default`**

Il percorso `$default` cattura le richieste che non corrispondono esplicitamente ad altri percorsi nell'API.

Quando la route `$default` riceve una richiesta, API Gateway invia il percorso completo della richiesta all'integrazione. Ad esempio, è possibile creare un'API con solo un percorso `$default` e integrarlo nel metodo `ANY` con l'endpoint `https://petstore-demo-endpoint.execute-api.com` HTTP. Quando si invia una richiesta a `https://api-id.execute-api.us-east-2.amazonaws.com/store/checkout`, API Gateway invia una richiesta a `https://petstore-demo-endpoint.execute-api.com/store/checkout`.

Per ulteriori informazioni sulle integrazioni HTTP, consultare Crea integrazioni proxy HTTP per HTTP APIs.

Routing delle richieste API

Quando un client invia una richiesta API, API Gateway stabilisce innanzitutto a quale fase instradare la richiesta. Se la richiesta corrisponde esplicitamente a una fase, API Gateway invia la richiesta a tale fase. Se nessuna fase corrisponde completamente alla richiesta, API Gateway invia la richiesta alla fase `$default`. Se non è presente alcuna `$default` fase, l'API restituisce `{"message":"Not Found"}` e non genera CloudWatch log.

Dopo aver selezionato una fase, API Gateway seleziona una route. API Gateway seleziona la route con la corrispondenza più specifica, utilizzando le seguenti priorità:

1. Corrispondenza completa per un percorso e un metodo.
2. Corrispondenza per un percorso e un metodo con una variabile di percorso greedy (`{proxy+}`).
3. Instradamento `$default`.

Se nessuna route corrisponde a una richiesta, API Gateway restituisce al client il valore `{"message":"Not Found"}`.

Ad esempio, si consideri un'API con una fase `$default` e le seguenti route di esempio:

1. `GET /pets/dog/1`
2. `GET /pets/dog/{id}`
3. `GET /pets/{proxy+}`
4. `ANY /{proxy+}`
5. `$default`

Nella tabella seguente viene riepilogato il modo in cui API Gateway instrada le richieste alle route di esempio.

Richiesta	Percorso selezionato	Spiegazione
`GET https://api-id.execute-`	`GET /pets/dog/1`	La richiesta corrisponde completamente a questo instradamento statico.

Richiesta	Percorso selezionato	Spiegazione
`api. region.amazonaws.com/pets/dog/1`		
`GET https://api-id.execute-api. region.amazonaws.com/pets/dog/2`	`GET /pets/dog/{id}`	La richiesta corrisponde completamente a questo percorso.
`GET https://api-id.execute-api. region.amazonaws.com/pets/cat/1`	`GET /pets/{proxy+}`	La richiesta non corrisponde completamente a un percorso. Il percorso con un metodo `GET` e una variabile di percorso greedy cattura questa richiesta .
`POST https://api-id.execute-api. region.amazonaws.com/test/5`	`ANY /{proxy+}`	Il metodo `ANY` corrisponde a tutti i metodi che non sono stati definiti per un percorso. I percorsi con variabili di percorso greedy hanno priorità più alta rispetto al percorso `$default`.

Tipi di indirizzo IP per API HTTP in Gateway API

Quando si crea un'API, è possibile specificare il tipo di indirizzo IP che può invocarla. È possibile scegliere IPv4 per risolvere gli indirizzi IPv4 per invocare l'API oppure dualstack per consentire a entrambi gli indirizzi IPv4 e IPv6 di invocare l'API. Si potrebbe voler impostare il tipo di indirizzo IP su dualstack per ridurre l'esaurimento dello spazio IP o per il livello di sicurezza. Per ulteriori informazioni sui vantaggi del tipo di indirizzo IP dualstack, consultare IPv6 on AWS.

Considerazioni sui tipi di indirizzo IP

Le seguenti considerazioni potrebbero influire sull'utilizzo dei tipi di indirizzo IP:

- Il tipo di indirizzo IP predefinito per le API HTTP è IPv4.

- Se per un'API esistente si modifica il tipo di indirizzo IP da IPv4 a dualstack, è necessario verificare che tutte le policy che controllano l'accesso alle API siano aggiornate in modo da tenere conto delle chiamate IPv6. Quando si modifica il tipo di indirizzo IP, la modifica diventa immediatamente effettiva.
- L'API può essere mappata su un nome di dominio personalizzato con un tipo di indirizzo IP diverso da quello dell'API. Se si disabilita l'endpoint API predefinito, si potrebbe influire sul modo in cui i chiamanti possono invocare l'API.

Modifica del tipo di indirizzo IP di un'API HTTP

È possibile modificare il tipo di indirizzo IP aggiornando la configurazione dell'API. È possibile aggiornare la configurazione dell'API utilizzando la Console di gestione AWS, AWS CLI, CloudFormation o un AWS SDK. Se si modifica il tipo di indirizzo IP dell'API, non è necessario eseguire di nuovo l'implementazione dell'API per rendere effettive le modifiche.

Console di gestione AWS

Per modificare il tipo di indirizzo IP di un'API HTTP

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere un'API HTTP.
3. Per Impostazioni API, scegliere Modifica.
4. Per Tipo di indirizzo IP, selezionare IPv4 o Dualstack.
5. Selezionare Salva.

   La modifica alla configurazione dell'API ha effetto immediato.

AWS CLI

Il comando update-api seguente aggiorna un'API in modo che abbia un tipo di indirizzo IP dualstack:

```
aws apigatewayv2 update-api \
    --api-id abcd1234 \
    --ip-address-type dualstack
```

L'output sarà simile al seguente:

```
{
    "ApiEndpoint": "https://abcd1234.execute-api.us-east-1.amazonaws.com",
    "ApiId": "abcd1234",
    "ApiKeySelectionExpression": "$request.header.x-api-key",
    "CreatedDate": "2025-02-04T22:20:20+00:00",
    "DisableExecuteApiEndpoint": false,
    "Name": "My-HTTP-API",
    "ProtocolType": "HTTP",
    "RouteSelectionExpression": "$request.method $request.path",
    "Tags": {},
    "NotificationUris": [],
    "IpAddressType": "dualstack"
}
```


Controlla e gestisci l'accesso a HTTP APIs in API Gateway

API Gateway supporta più meccanismi per controllare e gestire l'accesso a un'API HTTP:

- Gli autorizzatori Lambda utilizzano le funzioni Lambda per controllare l'accesso a. APIs Per ulteriori informazioni, consulta Controlla l'accesso a HTTP APIs con gli AWS Lambda autorizzatori.
- Gli autorizzatori JWT utilizzano token web JSON per controllare l'accesso a. APIs Per ulteriori informazioni, consulta Controllo degli accessi alle API HTTP con provider di autorizzazioni JWT in Gateway API.
- I ruoli e le politiche AWS IAM standard offrono controlli di accesso flessibili e robusti. Puoi utilizzare i ruoli e le policy IAM per controllare chi può creare e gestire APIs le tue e chi può richiamarle. Per ulteriori informazioni, consulta Controlla l'accesso a HTTP APIs con l'autorizzazione IAM in API Gateway.

Per migliorare il livello di sicurezza, è consigliabile configurare un sistema di autorizzazioni per tutti gli instradamenti per l'API HTTP. Potrebbe essere necessario eseguire questa operazione per rispettare vari framework di conformità. Per ulteriori informazioni, consultare Amazon API Gateway controls nella Guida per l'utente di AWS Security Hub .

Controlla l'accesso a HTTP APIs con gli AWS Lambda autorizzatori

Si sfrutta un'autorizzazione Lambda al fine di utilizzare una funzione Lambda per controllare l'accesso all'API HTTP. Quindi, quando un client invoca un'API, API Gateway richiama la funzione Lambda. API Gateway utilizza la risposta dalla funzione Lambda per determinare se il client può accedere all'API.

Tipo di formato payload

Il tipo di formato del payload dell'autorizzazione specifica il formato dei dati che API Gateway invia a un'autorizzazione Lambda e il modo in cui API Gateway interpreta la risposta ricevuta da Lambda. Se non specifichi una versione del formato di payload, Console di gestione AWS utilizza la versione più recente per impostazione predefinita. Se crei un autorizzatore Lambda utilizzando AWS CLI, o un SDK CloudFormation, devi specificare un. `authorizerPayloadFormatVersion` I valori supportati sono `1.0` e `2.0`.

Se hai bisogno di compatibilità con REST APIs, usa la versione. `1.0`

Gli esempi seguenti mostrano la struttura di ogni tipo di formato 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"
  }
}
```

Formato della risposta dell'autorizzazione

Il tipo di formato del payload determina anche la struttura della risposta che deve essere restituita dalla funzione Lambda.

Risposta della funzione Lambda per il formato 1.0

Se si sceglie il tipo di formato `1.0`, le autorizzazioni Lambda devono restituire una policy IAM che consenta o rifiuti l'accesso alla route dell'API. Nella policy è possibile utilizzare la sintassi standard delle policy IAM. Per alcuni esempi di policy IAM, consultare the section called “Controllo degli accessi per invocare un'API”. È possibile passare le proprietà del contesto alle integrazioni Lambda o ai log di accesso utilizzando `$context.authorizer.property`. L'oggetto `context` è facoltativo e `claims` è un segnaposto riservato che non può essere utilizzato come oggetto contesto. Per ulteriori informazioni, consulta the section called “Variabili di logging”.

Example

JSON

```
{
  "principalId": "abcdef",
  "policyDocument": {
    "Version":"2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Allow|Deny",
        "Resource": "arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/
{httpVerb}/[{resource}/[{child-resources}]]"
      }
    ]
  },
  "context": {
    "exampleKey": "exampleValue"
  }
}
```

Risposta della funzione Lambda per il formato 2.0

Se si sceglie il tipo di formato `2.0`, dalla funzione Lambda è possibile restituire un valore booleano o una policy IAM che utilizza la sintassi standard della policy di IAM. Per restituire un valore booleano, abilitare risposte semplici per l'autorizzazione. Negli esempi seguenti viene illustrato il formato in cui è necessario codificare il risultato restituito dalla funzione Lambda. L'oggetto `context` è facoltativo. È possibile passare le proprietà del contesto alle integrazioni Lambda o ai log di accesso utilizzando

`$context.authorizer.property`. Per ulteriori informazioni, consulta the section called “Variabili di logging”.

Simple response

```
{
  "isAuthorized": true/false,
  "context": {
    "exampleKey": "exampleValue"
  }
}
```

IAM policy

```
{
  "principalId": "abcdef",
  "policyDocument": {
    "Version":"2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Allow|Deny",
        "Resource": "arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/
{httpVerb}/[{resource}/[{child-resources}]]"
      }
    ]
  },
  "context": {
    "exampleKey": "exampleValue"
  }
}
```

Esempio di funzioni di autorizzazione Lambda

Le seguenti funzioni Lambda Node.js di esempio illustrano i formati di risposta richiesti che è necessario restituire dalla funzione Lambda per il tipo di formato del 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
```

Origini di identità

È facoltativamente possibile specificare le origini di identità per un'autorizzazione Lambda. Le origini di identità specificano la posizione dei dati necessari per autorizzare una richiesta. Ad esempio, è possibile specificare i valori di intestazione o di stringa di query come origini di identità. Se si specificano le origini di identità, i client devono includerle nella richiesta. Se la richiesta del client non include le origini di identità, API Gateway non richiama l'autorizzazione Lambda e il client riceve un errore `401`.

La tabella seguente illustra le origini di identità supportate per un sistema di autorizzazione Lambda.

Tipo	Esempio	Note
Valore intestazione	$request.header. *`name`*	I nomi delle intestazioni non fanno distinzione tra maiuscole e minuscole.
Valore della stringa di query	$request.querystring. *`name`*	I nomi delle stringhe di query fanno distinzione tra maiuscole e minuscole.
Variabile di contesto	$contesto. *`variableName`*	Valore di una variabile di contesto supportata.
Variabile di fase	$stageVariabili. *`variableName`*	Il valore di una variabile di fase.

È anche possibile ottenere direttamente `{"errorMessage" : "Unauthorized"}` dalla funzione Lambda per restituire un errore `401` ai client. Se si restituisce direttamente ai client un errore `401` della funzione Lambda, non specificare alcuna origine di identità quando si crea il sistema di autorizzazione Lambda.

Caching delle risposte delle autorizzazioni

Puoi abilitare la memorizzazione nella cache per un autorizzatore Lambda specificando un. authorizerResultTtlInSeconds Quando il caching è abilitato per un'autorizzazione, API Gateway utilizza le origini di identità dell'autorizzazione come chiave della cache. Se un client specifica gli stessi parametri nelle origini di identità all'interno del TTL configurato, API Gateway utilizza il risultato dell'autorizzazione memorizzato nella cache, anziché richiamare la funzione Lambda.

Per abilitare il caching, l'autorizzazione deve disporre di almeno un'origine di identità.

Se si abilitano risposte semplici per un'autorizzazione, la risposta dell'autorizzazione consente o rifiuta completamente tutte le richieste API che corrispondono ai valori dell'origine di identità memorizzata nella cache. Per autorizzazioni più granulari, disabilitare le risposte semplici e restituire una policy IAM. A seconda del sistema di autorizzazione, la policy IAM potrebbe dover controllare l'accesso a più utenti.

Per impostazione predefinita, API Gateway utilizza la risposta dell'autorizzazione memorizzata nella cache per tutte le route dell'API che utilizzano l'autorizzazione. Per inserire nella cache le risposte per l'instradamento, aggiungere `$context.routeKey` alle origini di identità dell'autorizzazione.

Creare un'autorizzazione Lambda

Quando si crea un'autorizzazione Lambda, si specifica la funzione Lambda che API Gateway deve utilizzare. È necessario concedere ad API Gateway l'autorizzazione per richiamare la funzione Lambda utilizzando la policy basata su risorse della funzione o un ruolo IAM. Il comando create-authorizer seguente crea un sistema di autorizzazione Lambda:

```
aws apigatewayv2 create-authorizer \
    --api-id abcdef123 \
    --authorizer-type REQUEST \
    --identity-source '$request.header.Authorization' \
    --name lambda-authorizer \
    --authorizer-uri 'arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-west-2:123456789012:function:my-function/invocations' \
    --authorizer-payload-format-version '2.0' \
    --enable-simple-responses
```

Il comando add-permission seguente aggiorna la policy di risorse della funzione Lambda per fornire a Gateway API l'autorizzazione per invocare la funzione. Se l'API Gateway non dispone dell'autorizzazione a richiamare la funzione, i client ricevono un errore `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"
```

Dopo aver creato un'autorizzazione e concesso ad API Gateway l'autorizzazione per richiamarla, aggiornare la route affinché utilizzi l'autorizzazione. Il comando update-route seguente aggiunge il sistema di autorizzazione Lambda all'instradamento. Se il sistema di autorizzazione Lambda utilizza il caching delle policy, è necessario aggiornare la policy per controllare l'accesso all'instradamento aggiuntivo.

```
aws apigatewayv2 update-route \
    --api-id abcdef123 \
    --route-id abc123 \
    --authorization-type CUSTOM \
    --authorizer-id def123
```

Risoluzione dei problemi relativi alle autorizzazioni Lambda

Se l'API Gateway non è in grado di richiamare l'autorizzazione Lambda o l'autorizzazione Lambda restituisce una risposta in un formato non valido, i client ricevono una risposta `500 Internal Server Error`.

Per risolvere gli errori, abilitare la registrazione degli accessi per la fase dell'API. Includere la variabile di registrazione `$context.authorizer.error` nel formato di log.

Se i log indicano che API Gateway non dispone dell'autorizzazione per richiamare la funzione, aggiornare la policy basata su risorse della funzione o fornire un ruolo IAM per concedere ad API Gateway il permesso di per richiamare l'autorizzazione.

Se i log indicano che la funzione Lambda restituisce una risposta non valida, verificare che la funzione Lambda restituisca una risposta nel formato richiesto.

Controllo degli accessi alle API HTTP con provider di autorizzazioni JWT in Gateway API

È possibile utilizzare JSON Web Tokens (JWT) come parte di framework OpenID Connect (OIDC) e OAuth 2.0 per limitare l'accesso client alle API.

Se si configura un autorizzatore JWT per una route dell'API, API Gateway convalida i JWT inviati dai client con le richieste API. API Gateway consente o nega le richieste in base alla convalida dei token e, facoltativamente, gli ambiti nel token. Se si configurano gli ambiti per un percorso, il token deve includere almeno uno degli ambiti del percorso.

È possibile configurare autorizzazioni distinte per ogni percorso di un'API o utilizzare lo stesso autorizzatore per più route.

> **Note**
>
> Non esiste un meccanismo standard per differenziare i token di accesso JWT da altri tipi di JWT come token ID OpenID Connect. A meno che non si richiedano token ID per l'autorizzazione API, si consiglia di configurare le route per richiedere gli ambiti di autorizzazione. È inoltre possibile configurare gli autorizzatori JWT per richiedere emittenti o gruppi di destinatari che il provider di identità utilizza solo quando si emettono token di accesso JWT.

Autorizzazione delle richieste API con un provider di autorizzazioni JWT

API Gateway utilizza il seguente flusso di lavoro generale per autorizzare le richieste alle route configurate per l'utilizzo di un'autorizzazione JWT.

1. Controllare la presenza di `identitySource` per un token. Il `identitySource` può includere solo il token o il token con il prefisso `Bearer`.
2. Decodifica del token.
3. Controlla l'algoritmo e la firma del token utilizzando la chiave pubblica recuperata dal dell'emittent `jwks_uri`. Attualmente sono supportati solo RSA-based gli algoritmi. API Gateway memorizza nella cache la chiave pubblica per un massimo di due ore nel migliore dei modi. In caso di perdita della cache, API Gateway recupera la chiave pubblica dall'emittente sulla rete, aggiungendo latenza alla richiesta che la attiva. È più probabile che le API con una frequenza di richiesta bassa o intermittente subiscano questi recuperi. Poiché una chiave pubblica può rimanere memorizzata nella cache per un massimo di due ore, concedi un periodo di tolleranza durante la rotazione delle chiavi, durante il quale sia la vecchia che la nuova chiave sono valide.
4. Convalida delle richieste. API Gateway valuta le seguenti richieste dei token:
   - `kid`: il token deve presentare una richiesta di intestazione che corrisponde alla chiave nel `jwks_uri` che ha firmato il token.
   - `iss`: deve corrispondere all'`issuer` configurato per l'autorizzazione.
   - `aud` o `client_id`: devono corrispondere a una delle voci `audience` configurate per l'autorizzazione. Gateway API convalida `client_id` solo se `aud` non è presente. Quando `aud` e `client_id` sono entrambi presenti, Gateway API valuta `aud`.
   - `exp` – deve essere successivo all'ora corrente in UTC.
   - `nbf` – deve essere precedente all'ora corrente in UTC.
   - `iat` – deve essere precedente all'ora corrente in UTC.

- `scope` o `scp`: il token deve includere almeno uno degli ambiti negli `authorizationScopes` della route.

Se uno di questi passaggi non riesce, API Gateway nega la richiesta API.

Dopo aver convalidato il JWT, API Gateway passa le registrazioni nel token all'integrazione della route API. Le risorse di back-end, come le funzioni Lambda, possono accedere alle attestazioni JWT. Ad esempio, se JWT include un'attestazione di identità `emailID`, è disponibile per un'integrazione Lambda in `$event.requestContext.authorizer.jwt.claims.emailID`. Per ulteriori informazioni sul payload che API Gateway invia alle integrazioni Lambda, consulta the section called “AWS Lambda integrazioni”.

Creazione di un'autorizzazione JWT

Prima di creare un autorizzatore JWT, è necessario registrare un'applicazione client con un provider di identità. È anche necessario aver creato un'API HTTP. Per esempi di creazione di un'API HTTP, consultare Creazione di un'API HTTP.

Creazione di un provider di autorizzazioni JWT tramite la console

Nei seguenti passaggi viene illustrato come creare un provider di autorizzazioni JWT utilizzando la console.

Per creare un provider di autorizzazioni JWT tramite la console

1. Accedi alla console API Gateway all'indirizzo. https://console.aws.amazon.com/apigateway
2. Scegliere un'API HTTP.
3. Nel pannello di navigazione principale, scegli Autorizzazione.
4. Seleziona la scheda Gestisci provider di autorizzazioni.
5. Scegli Crea.
6. Per Tipo di autorizzazione scegli JWT.
7. Configurare il provider di autorizzazioni JWT e specificare una Origine identità che definisca l'origine del token.
8. Scegli Crea.

Crea un autorizzatore JWT utilizzando AWS CLI

Il comando create-authorizer seguente crea un'autorizzazione JWT. Per `jwt-configuration`, specificare `Audience` e `Issuer` per il provider di identità. Se si utilizza Amazon Cognito come provider di identità, `IssuerUrl` è `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
```

Crea un autorizzatore JWT usando AWS CloudFormation

Il CloudFormation modello seguente crea un'API HTTP con un autorizzatore JWT che utilizza Amazon Cognito come provider di identità.

L'output del CloudFormation modello è un URL per un'interfaccia utente ospitata da Amazon Cognito in cui i clienti possono registrarsi e accedere per ricevere un JWT. Dopo l'accesso, il client viene reindirizzato all'API HTTP con un token di accesso nell'URL. Per invocare l'API con il token di accesso, modifica # nell'URL in ? per utilizzare il token come parametro della stringa di query.

Esempio CloudFormation modello

```
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
```

Aggiornamento di un instradamento per utilizzare un provider di autorizzazioni JWT

Puoi utilizzare la console AWS CLI, l'SDK o un AWS SDK per aggiornare un percorso in modo da utilizzare un autorizzatore JWT.

Aggiornamento di un instradamento per utilizzare un provider di autorizzazioni JWT tramite la console

Nei seguenti passaggi viene illustrato come aggiornare un instradamento per utilizzare un provider di autorizzazioni JWT utilizzando la console.

Per creare un provider di autorizzazioni JWT tramite la console

1. Accedi alla console API Gateway all'indirizzo. https://console.aws.amazon.com/apigateway
2. Scegliere un'API HTTP.
3. Nel pannello di navigazione principale, scegli Autorizzazione.
4. Scegli un metodo, quindi seleziona il provider di autorizzazioni dal menu a discesa e scegli Collega provider di autorizzazioni.

Aggiorna un percorso per utilizzare un autorizzatore JWT utilizzando AWS CLI

Il comando update-route seguente aggiorna un instradamento per utilizzare un provider di autorizzazioni JWT:

```
aws apigatewayv2 update-route \
   --api-id api-id  \
   --route-id route-id  \
   --authorization-type JWT \
   --authorizer-id authorizer-id \
   --authorization-scopes user.email
```

Controlla l'accesso a HTTP APIs con l'autorizzazione IAM in API Gateway

È possibile abilitare l'autorizzazione IAM per route di API HTTP. Quando l'autorizzazione IAM è abilitata, i client devono utilizzare Signature Version 4 (SigV4) per firmare le loro richieste con AWS le credenziali. API Gateway richiama la route API solo se il client dispone dell'autorizzazione `execute-api` per la route.

L'autorizzazione IAM per HTTP APIs è simile a quella per REST. APIs

> **Note**
>
> Le politiche relative alle risorse non sono attualmente supportate per HTTP APIs.

Per esempi di politiche IAM che concedono ai client il permesso di invocazione APIs, consultathe section called “Controllo degli accessi per invocare un'API”.

Abilitazione dell'autorizzazione IAM per una route

Il comando update-route seguente abilita l’autorizzazione IAM per un instradamento di un’API HTTP:

```
aws apigatewayv2 update-route \
    --api-id abc123 \
    --route-id abcdef \
    --authorization-type AWS_IAM
```

Creazione delle integrazioni per API HTTP in Gateway API

Le integrazioni collegano una route alle risorse di back-end. Le API HTTP supportano proxy Lambda, il servizio AWS e le integrazioni proxy HTTP. Ad esempio, è possibile configurare una richiesta `POST` per la route `/signup` dell'API per l'integrazione con una funzione Lambda che gestisce la registrazione dei clienti.

Argomenti

- Crea integrazioni AWS Lambda proxy per HTTP APIs in API Gateway
- Crea integrazioni proxy HTTP per HTTP APIs
- Crea integrazioni AWS di servizi per HTTP APIs in API Gateway
- Crea integrazioni private per HTTP APIs in API Gateway

Crea integrazioni AWS Lambda proxy per HTTP APIs in API Gateway

Un'integrazione proxy Lambda consente di integrare una route API con una funzione Lambda. Quando un client chiama l'API, API Gateway invia la richiesta alla funzione Lambda e restituisce la risposta della funzione al client. Per esempi di creazione di un'API HTTP, consultare Creazione di un'API HTTP.

Tipo di formato payload

La versione del formato del payload specifica il formato dell'evento che Gateway API invia a un'integrazione Lambda e il modo in cui Gateway API interpreta la risposta ricevuta da Lambda. Se non specifichi una versione del formato di payload, Console di gestione AWS utilizza la versione più recente per impostazione predefinita. Se crei un'integrazione Lambda utilizzando AWS CLI CloudFormation, o un SDK, devi specificare un. `payloadFormatVersion` I valori supportati sono `1.0` e `2.0`.

Per ulteriori informazioni su come impostare `payloadFormatVersion`, consulta create-integration. Per ulteriori informazioni su come determinare l'entità `payloadFormatVersion` di un'integrazione esistente, consulta get-integration.

Differenze di formato del payload

L'elenco seguente mostra le differenze tra le versioni del formato del payload `1.0` e `2.0`:

- Il formato `2.0` non contiene i campi `multiValueHeaders` o `multiValueQueryStringParameters`. Le intestazioni duplicate sono combinate con virgole e

incluse nel campo `headers`. Le stringhe di query duplicate sono combinate con virgole e incluse nel campo `queryStringParameters`.

- Il formato `2.0` include `rawPath`. Se si utilizza una mappatura API per collegare la fase a un nome di dominio personalizzato, `rawPath` non fornirà il valore di mappatura API. Utilizza il formato `1.0` e `path` per accedere alla mappatura API per il nome di dominio personalizzato.
- Il formato `2.0` include un nuovo campo `cookies`. Tutte le intestazioni dei cookie nella richiesta vengono combinate con virgole e aggiunte al campo `cookies`. Nella risposta al client, ogni cookie diventa un'intestazione `set-cookie`.

Struttura del formato del payload

Gli esempi seguenti mostrano la struttura di ogni tipo di formato payload. Per tutti i nomi delle intestazioni si utilizzano le minuscole.

2.0

```
{
  "version": "2.0",
  "routeKey": "$default",
  "rawPath": "/my/path",
  "rawQueryString": "parameter1=value1&parameter1=value2&parameter2=value",
  "cookies": [
    "cookie1",
    "cookie2"
  ],
  "headers": {
    "header1": "value1",
    "header2": "value1,value2"
  },
  "queryStringParameters": {
    "parameter1": "value1,value2",
    "parameter2": "value"
  },
  "requestContext": {
    "accountId": "123456789012",
    "apiId": "api-id",
    "authentication": {
      "clientCert": {
        "clientCertPem": "CERT_CONTENT",
        "subjectDN": "www.example.com",
        "issuerDN": "Example issuer",
```

```
        "serialNumber": "a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1",
        "validity": {
          "notBefore": "May 28 12:30:02 2019 GMT",
          "notAfter": "Aug  5 09:36:04 2021 GMT"
        }
      }
    },
    "authorizer": {
      "jwt": {
        "claims": {
          "claim1": "value1",
          "claim2": "value2"
        },
        "scopes": [
          "scope1",
          "scope2"
        ]
      }
    },
    "domainName": "id.execute-api.us-east-1.amazonaws.com",
    "domainPrefix": "id",
    "http": {
      "method": "POST",
      "path": "/my/path",
      "protocol": "HTTP/1.1",
      "sourceIp": "192.0.2.1",
      "userAgent": "agent"
    },
    "requestId": "id",
    "routeKey": "$default",
    "stage": "$default",
    "time": "12/Mar/2020:19:03:58 +0000",
    "timeEpoch": 1583348638390
  },
  "body": "Hello from Lambda",
  "pathParameters": {
    "parameter1": "value1"
  },
  "isBase64Encoded": false,
  "stageVariables": {
    "stageVariable1": "value1",
    "stageVariable2": "value2"
  }
```

```
}
```

1.0

```
{
  "version": "1.0",
  "resource": "/my/path",
  "path": "/my/path",
  "httpMethod": "GET",
  "headers": {
    "header1": "value1",
    "header2": "value2"
  },
  "multiValueHeaders": {
    "header1": [
      "value1"
    ],
    "header2": [
      "value1",
      "value2"
    ]
  },
  "queryStringParameters": {
    "parameter1": "value1",
    "parameter2": "value"
  },
  "multiValueQueryStringParameters": {
    "parameter1": [
      "value1",
      "value2"
    ],
    "parameter2": [
      "value"
    ]
  },
  "requestContext": {
    "accountId": "123456789012",
    "apiId": "id",
    "authorizer": {
      "claims": null,
      "scopes": null
    },
    "domainName": "id.execute-api.us-east-1.amazonaws.com",
```

```
    "domainPrefix": "id",
    "extendedRequestId": "request-id",
    "httpMethod": "GET",
    "identity": {
      "accessKey": null,
      "accountId": null,
      "caller": null,
      "cognitoAuthenticationProvider": null,
      "cognitoAuthenticationType": null,
      "cognitoIdentityId": null,
      "cognitoIdentityPoolId": null,
      "principalOrgId": null,
      "sourceIp": "192.0.2.1",
      "user": null,
      "userAgent": "user-agent",
      "userArn": null,
      "clientCert": {
        "clientCertPem": "CERT_CONTENT",
        "subjectDN": "www.example.com",
        "issuerDN": "Example issuer",
        "serialNumber": "a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1",
        "validity": {
          "notBefore": "May 28 12:30:02 2019 GMT",
          "notAfter": "Aug  5 09:36:04 2021 GMT"
        }
      }
    },
    "path": "/my/path",
    "protocol": "HTTP/1.1",
    "requestId": "id=",
    "requestTime": "04/Mar/2020:19:15:17 +0000",
    "requestTimeEpoch": 1583349317135,
    "resourceId": null,
    "resourcePath": "/my/path",
    "stage": "$default"
  },
  "pathParameters": null,
  "stageVariables": null,
  "body": "Hello from Lambda!",
  "isBase64Encoded": false
}
```

Formato di risposta della funzione Lambda

Il tipo di formato del payload determina la struttura della risposta che deve essere restituita dalla funzione Lambda.

Risposta della funzione Lambda per il formato 1.0

Con la versione di formato `1.0`, le integrazioni Lambda devono restituire una risposta nel formato JSON seguente:

Example

```
{
    "isBase64Encoded": true|false,
    "statusCode": httpStatusCode,
    "headers": { "headername": "headervalue", ... },
    "multiValueHeaders": { "headername": ["headervalue", "headervalue2", ...], ... },
    "body": "..."
}
```

Risposta della funzione Lambda per il formato 2.0

Con il tipo di formato `2.0`, API Gateway può dedurre autonomamente il formato della risposta. Se la funzione Lambda restituisce un JSON valido e non restituisce un , API Gateway fa le seguenti ipotes `statusCode`:

- `isBase64Encoded` è `false`.
- `statusCode` è `200`.
- `content-type` è `application/json`.
- `body` è la risposta della funzione.

Gli esempi seguenti mostrano l'output di una funzione Lambda e l'interpretazione da parte di API Gateway.

Risultato della funzione Lambda	Interpretazione di API Gateway
`"Hello from Lambda!"`	`{`

Risultato della funzione Lambda	Interpretazione di 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>

Per personalizzare la risposta, la funzione Lambda deve restituire una risposta con il seguente formato.

```
{
    "cookies" : ["cookie1", "cookie2"],
    "isBase64Encoded": true|false,
    "statusCode": httpStatusCode,
    "headers": { "headername": "headervalue", ... },
    "body": "Hello from Lambda!"
}
```

Crea integrazioni proxy HTTP per HTTP APIs

Un'integrazione proxy HTTP consente di collegare una route API a un endpoint HTTP instradabile pubblicamente. Con questo tipo di integrazione, API Gateway passa l'intera richiesta e risposta tra il front-end e il back-end.

Per creare un'integrazione HTTP, fornire l'URL di un endpoint HTTP instradabile pubblicamente.

Integrazione proxy HTTP con variabili di percorso

Nelle route delle API HTTP è possibile utilizzare delle variabili di percorso.

Ad esempio, la route `/pets/{petID}` cattura le richieste a `/pets/6`. Puoi fare riferimento alle variabili di percorso nell'URI di integrazione per inviare il contenuto della variabile a un'integrazione. Un esempio è `/pets/extendedpath/{petID}`.

Puoi utilizzare variabili di percorso greedy per catturare tutte le risorse figlio di una route. Per creare una variabile di percorso greedy, aggiungere + al nome della variale, ad esempio `{proxy+}`.

Per impostare una route con un'integrazione proxy HTTP che cattura tutte le richieste, crea una route API con una variabile di percorso greedy (ad esempio, `/parent/{proxy+}`). Integra la route con un endpoint HTTP (ad esempio `https://petstore-demo-endpoint.execute-api.com/petstore/{proxy}`) nel metodo `ANY`. La variabile di percorso greedy deve trovarsi alla fine del percorso della risorsa.

Crea integrazioni AWS di servizi per HTTP APIs in API Gateway

Puoi integrare la tua API HTTP con AWS i servizi utilizzando integrazioni di prima classe. Un'integrazione di prima classe collega una route di un'API HTTP a un'API di un servizio AWS . Quando un client richiama un percorso supportato da un'integrazione di prima classe, API Gateway richiama un' AWS API di servizio per te. Ad esempio, puoi utilizzare integrazioni di prima classe per inviare un messaggio a una coda di Amazon Simple Queue Service o per avviare una macchina a stati. AWS Step Functions Per le operazioni di servizio supportate, consulta the section called “AWSRiferimento alle integrazioni di servizi ”.

Mappatura dei parametri delle richieste

Le integrazioni di prima classe hanno parametri obbligatori e facoltativi. È necessario configurare tutti i parametri obbligatori per creare un'integrazione. È possibile utilizzare valori statici o mappare i parametri che vengono valutati dinamicamente in fase di runtime. Per un elenco completo delle integrazioni e dei parametri supportati, consulta the section called “AWSRiferimento alle integrazioni di servizi ”.

Nella tabella seguente vengono descritti i parametri supportati della richiesta di mappatura.

Tipo	Esempio	Note
Valore intestazione	$request.header. *name*	I nomi delle intestazioni non fanno distinzione tra

Tipo	Esempio	Note
		maiuscole e minuscole. API Gateway combina più valori di intestazione con virgole, ad esempio `"header1": "value1,value2"` .
Valore della stringa di query	$request.querystring. *name*	I nomi delle stringhe di query fanno distinzione tra maiuscole e minuscole . API Gateway combina più valori con virgole, ad esempio `"querystring1": "Value1,Value2"` .
Parametro del percorso	$richiesta.percorso. *name*	Il valore di un parametro del percorso nella richiesta. Ad esempio, se l'instradamento è `/pets/{petId}` , è possibile mappare il parametro `petId` della richiesta con *`$request.path.petId`* .
Passthrough corpo richiesta	$request.body	API Gateway passa l'intero corpo della richiesta.

Tipo	Esempio	Note
Corpo della richiesta	$request.body. *name*	Un'espressione di percorso JSON. Le espressioni di discesa ricorsiva (`$request.body..` *name*) e di filtro (`?(`*expression*` )`) non sono supportate. **Note** Quando si specifica un percorso JSON, API Gateway tronca il corpo della richiesta a 100 KB e quindi applica l'espressione di selezione. Per inviare payload superiori a 100 KB, specificare `$request.body` .
Variabile di contesto	$contesto. *variableName*	Valore di una variabile di contesto supportata.
Variabile di fase	$stageVariabili. *variableName*	Il valore di una variabile di fase.
Valore statico	*string*	Un valore costante.

Creazione di un'integrazione di prima classe

Prima di creare un'integrazione di prima classe, devi creare un ruolo IAM che conceda le autorizzazioni API Gateway per richiamare l'azione di AWS servizio con cui stai effettuando l'integrazione. Per ulteriori informazioni, consulta Creazione di un ruolo per un servizio AWS.

Per creare un'integrazione di prima classe, scegli un'azione di AWS servizio supportata`SQS-SendMessage`, ad esempio configura i parametri di richiesta e fornisci un ruolo che conceda le autorizzazioni API Gateway per richiamare l'API del servizio integrato. AWS A seconda del sottotipo di integrazione, sono necessari diversi parametri di richiesta. Per ulteriori informazioni, consulta the section called “AWSRiferimento alle integrazioni di servizi ”.

Il comando create-integration seguente crea un’integrazione che invia un messaggio 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"}'
```

Crea un'integrazione di prima classe utilizzando CloudFormation

L'esempio seguente mostra uno CloudFormation snippet che crea un `/{source}/{detailType}` percorso con un'integrazione di prima classe con Amazon. EventBridge

Il parametro `Source` viene mappato al parametro del percorso `{source}`, `DetailType` viene mappato al parametro del percorso `{DetailType}` e il parametro `Detail` viene mappato al corpo della richiesta.

Lo snippet non mostra il router di eventi o il ruolo IAM che concede le autorizzazioni Gateway API per richiamare l'operazione `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"
```

Riferimento al sottotipo di integrazione

I seguenti sottotipi di integrazione sono supportati per le API HTTP.

Sottotipi di integrazione

- EventBridge-PutEvents 1.0
- SQS-SendMessage 1.0
- SQS-ReceiveMessage 1.0
- SQS-DeleteMessage 1.0
- SQS-PurgeQueue 1.0
- AppConfig-GetConfiguration 1.0
- Kinesis-PutRecord 1.0
- StepFunctions-StartExecution 1.0
- StepFunctions-StartSyncExecution 1.0
- StepFunctions-StopExecution 1.0

EventBridge-PutEvents 1.0

Invia eventi personalizzati ad Amazon EventBridge in modo che possano essere abbinati alle regole.

Parametro	Campo obbligatorio
Dettaglio	True
DetailType	True

Parametro	Campo obbligatorio
Crea	True
Orario	False
EventBusName	False
Risorse	False
Regione	False
TraceHeader	False

Per ulteriori informazioni, consulta PutEvents nella Guida di riferimento per le API di Amazon EventBridge.

SQS-SendMessage 1.0

Recapita un messaggio alla coda specificata.

Parametro	Campo obbligatorio
QueueUrl	True
MessageBody	True
DelaySeconds	False
MessageAttributes	False
MessageDeduplicationId	False
MessageGroupId	False
MessageSystemAttributes	False
Regione	False

Per ulteriori informazioni, consulta SendMessage nella Guida di riferimento per le API di Amazon Simple Queue Service.

SQS-ReceiveMessage 1.0

Recupera uno o più messaggi (fino a 10) dalla coda specificata.

Parametro	Campo obbligatorio
QueueUrl	True
AttributeNames	False
MaxNumberOfMessages	False
MessageAttributeNames	False
ReceiveRequestAttemptId	False
VisibilityTimeout	False
WaitTimeSeconds	False
Regione	False

Per ulteriori informazioni, consulta ReceiveMessage nella Guida di riferimento per le API di Amazon Simple Queue Service.

SQS-DeleteMessage 1.0

Elimina il messaggio specificato dalla coda specificata.

Parametro	Campo obbligatorio
ReceiptHandle	True
QueueUrl	True
Regione	False

Per ulteriori informazioni, consulta DeleteMessage nella Guida di riferimento per le API di Amazon Simple Queue Service.

SQS-PurgeQueue 1.0

Elimina tutti i messaggi nella coda specificata.

Parametro	Campo obbligatorio
QueueUrl	True
Regione	False

Per ulteriori informazioni, consulta PurgeQueue nella Guida di riferimento per le API di Amazon Simple Queue Service.

AppConfig-GetConfiguration 1.0

Ricezione di informazioni su una configurazione.

Parametro	Campo obbligatorio
Applicazione	True
Ambiente	True
Configurazione	True
ClientId	True
ClientConfigurationVersion	False
Regione	False

Per ulteriori informazioni, consulta GetConfiguration in Documentazione di riferimento delle API AWS AppConfig.

Kinesis-PutRecord 1.0

Scrive un singolo record di dati in un flusso di dati di Amazon Kinesis.

Parametro	Campo obbligatorio
StreamName	True
Dati	True
PartitionKey	True
SequenceNumberForOrdering	False
ExplicitHashKey	False
Regione	False

Per ulteriori informazioni, consulta PutRecord nella Guida di riferimento per le API di Amazon Kinesis Data Streams.

StepFunctions-StartExecution 1.0

Avvia l'esecuzione di una macchina a stati.

Parametro	Campo obbligatorio
StateMachineArn	True
Nome	False
Input	False
Regione	False

Per ulteriori informazioni, consulta StartExecution nella Documentazione di riferimento delle API AWS Step Functions.

StepFunctions-StartSyncExecution 1.0

Avvia un'esecuzione sincrona di una macchina a stati.

Parametro	Campo obbligatorio
StateMachineArn	True
Nome	False
Input	False
Regione	False
TraceHeader	False

Per ulteriori informazioni, consulta StartSyncExecution nella Documentazione di riferimento delle API AWS Step Functions.

StepFunctions-StopExecution 1.0

Interrompe un'esecuzione.

Parametro	Campo obbligatorio
ExecutionArn	True
Causa	False
Errore	False
Regione	False

Per ulteriori informazioni, consulta StopExecution nella Documentazione di riferimento delle API AWS Step Functions.

Crea integrazioni private per HTTP APIs in API Gateway

Le integrazioni private consentono di creare integrazioni API con risorse private in un VPC, come ad esempio Application Load Balancer o applicazioni basate su container di Amazon ECS.

Puoi esporre le risorse in un VPC per l'accesso da parte di client esterni al VPC utilizzando integrazioni private. È possibile controllare l'accesso all'API utilizzando uno qualsiasi dei metodi di autorizzazione supportati da API Gateway.

> **Note**
>
> Per creare un'integrazione privata, devi innanzitutto creare un collegamento VPC. I link VPC V2 sono ora supportati sia per HTTP che per REST. APIs Per ulteriori informazioni sui collegamenti VPC V2, consulta. the section called “Collegamenti VPC V2”

Dopo aver creato un VPC link V2, puoi configurare integrazioni private che si connettono a un Application Load Balancer, Network Load Balancer o a risorse registrate con un servizio. AWS Cloud Map

Considerazioni

Le seguenti considerazioni potrebbero influire sull'utilizzo delle integrazioni private:

- Tutte le risorse devono essere di proprietà dello stesso. Account AWS Ciò include il AWS Cloud Map servizio o il servizio di bilanciamento del carico, il collegamento VPC e l'API HTTP.
- Per impostazione predefinita, il traffico di integrazione privata utilizza il protocollo HTTP. Per utilizzare HTTPS, specifica un. `tlsConfig` A tale scopo Console di gestione AWS, quando crei l'integrazione privata, scegli Impostazioni avanzate e quindi inserisci un nome di server sicuro.
- Per le integrazioni private, API Gateway include la porzione fase dell'endpoint API nella richiesta alle risorse back-end. Ad esempio, una richiesta alla fase `test` di un'API include `test/`*`route-path`* nella richiesta all'integrazione privata. Per rimuovere il nome della fase dalla richiesta alle risorse di back-end, utilizzare parameter mapping (mappatura dei parametri) per sovrascrivere il percorso della richiesta a `$request.path`.

Creare un'integrazione privata utilizzando un Application Load Balancer o un Network Load Balancer

Prima di creare un'integrazione privata, è necessario creare un collegamento VPC V2. Per ulteriori informazioni sui collegamenti VPC V2, consulta. the section called “Collegamenti VPC V2”

Per creare un'integrazione privata con un Application Load Balancer o un Network Load Balancer, creare un'integrazione proxy HTTP, specificare il collegamento VPC da utilizzare e fornire l'ARN del listener del sistema di bilanciamento del carico.

Il comando create-integration seguente crea un'integrazione privata che si connette a un bilanciatore del carico utilizzando un collegamento 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
```

Crea un'integrazione privata utilizzando Service Discovery AWS Cloud Map

Prima di creare un'integrazione privata, è necessario creare un collegamento VPC V2. Per ulteriori informazioni sui collegamenti VPC, consultare the section called "Collegamenti VPC V2".

Per le integrazioni con AWS Cloud Map, API Gateway utilizza `DiscoverInstances` per identificare le risorse. È possibile utilizzare i parametri di query per fare riferimento a risorse specifiche. Gli attributi delle risorse registrate devono includere indirizzi IP e porte. API Gateway distribuisce le richieste tra le risorse integre restituite da `DiscoverInstances`. Per ulteriori informazioni, consulta DiscoverInstancesl' AWS Cloud Map API Reference.

> **Note**
>
> Se utilizzi Amazon ECS per compilare le voci AWS Cloud Map, devi configurare il tuo task Amazon ECS per utilizzare i record SRV con Amazon ECS Service Discovery o attivare Amazon ECS Service Connect. Per ulteriori informazioni, consulta Interconnecting services nella Guida per gli sviluppatori di Amazon Elastic Container Service.

Per creare un'integrazione privata con AWS Cloud Map, crea un'integrazione proxy HTTP, specifica il link VPC da utilizzare e fornisci l'ARN del servizio. AWS Cloud Map

Il seguente comando create-integration crea un'integrazione privata che utilizza il rilevamento dei AWS Cloud Map servizi per identificare le risorse:

```
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
```

Configurazione di CORS per HTTP APIs in API Gateway

CORS (Cross-origin resource sharing) è una caratteristica di sicurezza del browser che limita le richieste HTTP avviate da script in esecuzione nel browser. Se non riesci ad accedere all'API e ricevi un messaggio di errore che contiene `Cross-Origin Request Blocked`, potresti dover abilitare CORS. Per ulteriori informazioni, consulta Che cos'è CORS?

CORS è in genere necessario per creare applicazioni Web che accedano a APIs server ospitati su un dominio o un'origine diversi. È possibile abilitare CORS per consentire le richieste all'API da un'applicazione Web ospitata in un dominio diverso. Ad esempio, se l'API è ospitata in `https://{api_id}.execute-api.{region}.amazonaws.com/` e si desidera chiamare l'API da un'applicazione Web ospitata in `example.com`, l'API deve supportare CORS.

Se si configura CORS per un'API, API Gateway invia automaticamente una risposta alle richieste OPTIONS preliminari, anche se non esiste una route OPTIONS configurata per l'API. Per una richiesta CORS, API Gateway aggiunge le intestazioni CORS configurate alla risposta da un'integrazione.

> **Note**
>
> Se si configura CORS per un'API, API Gateway ignora le intestazioni CORS restituite dall'integrazione back-end.

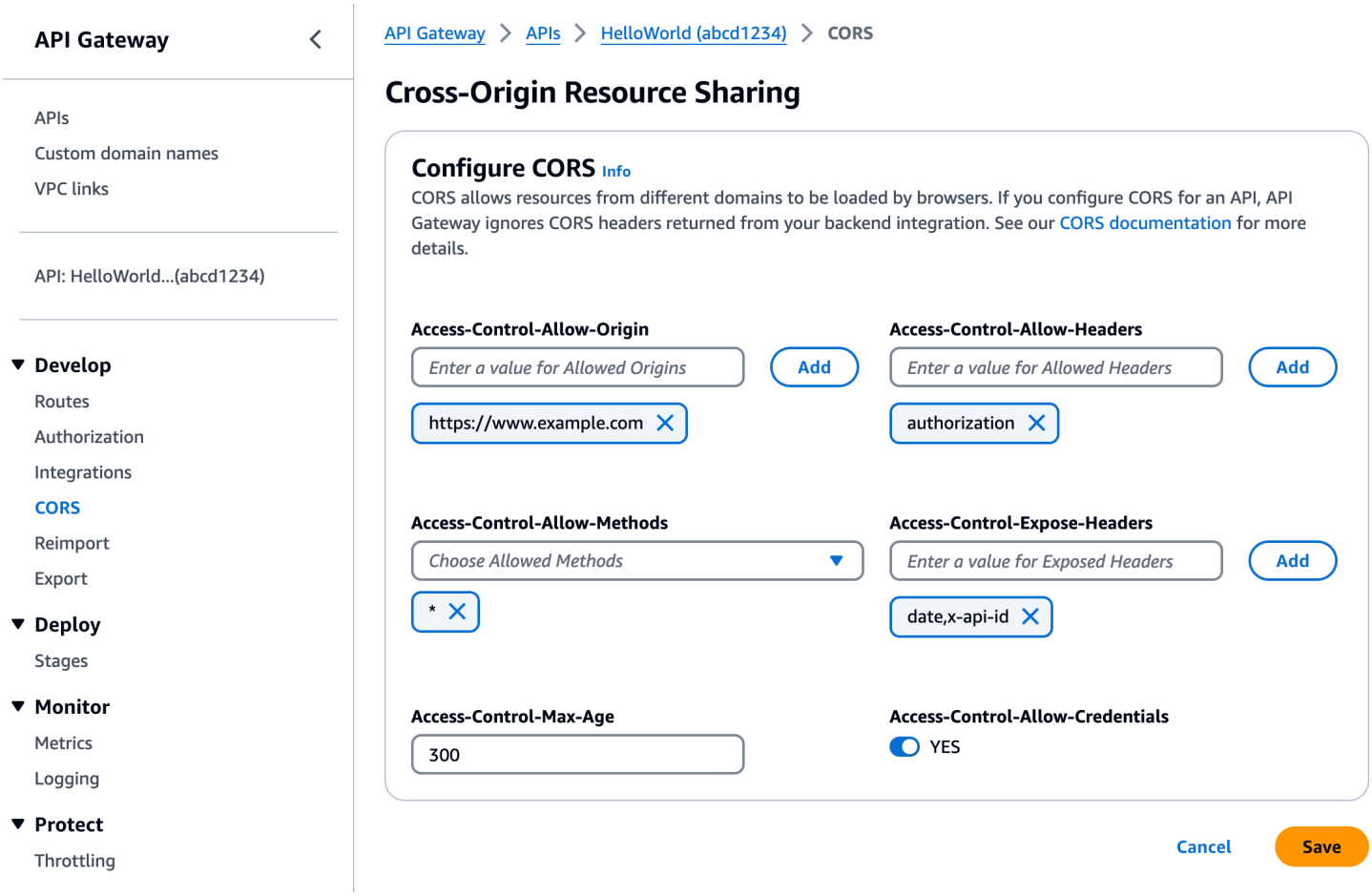
È possibile specificare i seguenti parametri in una configurazione CORS. Per aggiungere questi parametri utilizzando la console Gateway API, scegli Aggiungi dopo aver inserito il valore.

Intestazioni CORS	Proprietà di configurazione CORS	Valori di esempio
Access-Control-Allow-Origin	allowOrigins	• `https://www.example.com` • `*` (consente tutte le origini) • `https://*` (consente qualsiasi origine che inizia con https://)

Intestazioni CORS	Proprietà di configurazione CORS	Valori di esempio
		• `http://*` (consente qualsiasi origine che inizia con http://)
Access-Control-Allow-Credentials	allowCredentials	true
Access-Control-Expose-Headers	exposeHeaders	data,, * x-api-id
Access-Control-Max-Age	maxAge	300
Access-Control-Allow-Methods	allowMethods	GET, POST, DELETE, *
Access-Control-Allow-Headers	allowHeaders	authorization, *

Per restituire le intestazioni CORS, la richiesta deve contenere un'intestazione `origin`. Per il metodo `OPTIONS`, la richiesta deve contenere un'intestazione `origin` e un'intestazione `Access-Control-Request-Method`.

La configurazione CORS potrebbe essere simile all'immagine seguente:



Configurazione di CORS per un'API HTTP con un instradamento **`$default`** e un provider di autorizzazioni

È possibile abilitare CORS e configurare l'autorizzazione per qualsiasi route di un'API HTTP. Quando si abilitano CORS e l'autorizzazione per la route `$default`, ci sono alcune considerazioni speciali di cui tenere conto. Il percorso `$default` cattura le richieste per tutti i metodi e i percorsi che non sono stati definiti in modo esplicito, incluse le richieste `OPTIONS`. Per supportare le richieste `OPTIONS` non autorizzate, aggiungi all'API una route `OPTIONS /{proxy+}` che non richiede l'autorizzazione e collega un'integrazione alla route. Il percorso `OPTIONS /{proxy+}` ha priorità più alta rispetto al percorso `$default`. Di conseguenza, consente ai client di inviare richieste `OPTIONS` all'API senza autorizzazione. Per ulteriori informazioni sulle priorità di instradamento, consultare Routing delle richieste API.

Configurare CORS per un'API HTTP utilizzando la CLI AWS

Il comando update-api seguente abilita le richieste CORS da `https://www.example.com`:

Example

```
aws apigatewayv2 update-api --api-id api-id --cors-configuration AllowOrigins="https://
www.example.com"
```

Per maggiori informazioni, consultare CORS nella Guida di riferimento delle API di Amazon API Gateway Versione 2.

Trasformazione delle richieste e delle risposte API per API HTTP in Gateway API

È possibile modificare le richieste API dai client prima che raggiungano le integrazioni back-end. È inoltre possibile modificare la risposta dalle integrazioni prima che API Gateway restituisca la risposta ai client. È possibile utilizzare la mappatura dei parametri per modificare le richieste e le risposte API per le API HTTP. Per utilizzare la mappatura dei parametri, specificare i parametri di richiesta o risposta API da modificare e la modalità per modificare tali parametri.

Trasformazione delle richieste API

È possibile utilizzare i parametri di richiesta per modificare le richieste prima che raggiungano le integrazioni back-end. È possibile modificare le intestazioni, le stringhe di query o il percorso della richiesta.

I parametri di richiesta sono una mappa chiave-valore. La chiave identificherà la posizione del parametro di richiesta da modificare e la modalità per modificarlo. Il valore specifica i nuovi dati per il parametro.

Nella tabella seguente sono illustrate le chiavi supportate.

Tipo	Sintassi
Intestazione	`append|overwrite|remove:header.` *`headername`*
Stringa di query	`append|overwrite|remove:querystring.` *`querystring-name`*
Percorso	`overwrite:path`

Nella tabella seguente sono illustrati i valori supportati che è possibile mappare ai parametri.

Tipo	Sintassi	Note
Valore intestazione	$request.header.*name* o ${request.header.*name*}	I nomi delle intestazioni non fanno distinzione tra maiuscole e minuscole. API Gateway combina più valori di intestazione con virgole, ad esempio `"header1": "value1,value2"` . Alcune intestazioni sono riservate. Per ulteriori informazioni, consulta the section called “Intestazioni riservate”.
Valore della stringa di query	$request.querystring.*name* o ${request.querystring.*name*}	I nomi delle stringhe di query fanno distinzione tra maiuscole e minuscole . API Gateway combina più valori con virgole, ad esempio `"querystring1" "Value1,Value2"` .
Corpo della richiesta	$request.body.*name* o ${request.body.*name*}	Un'espressione di percorso JSON. Le espressioni di discesa ricorsiva (`$request.body..name`) e di filtro (`?(expression)`) non sono supportate. **Note** Quando si specifica un percorso JSON, API Gateway tronca il corpo della richiesta a 100 KB e quindi

Tipo	Sintassi	Note
		applica l'espress ione di selezione. Per inviare payload superiori a 100 KB, specificare `$request.body` .
Percorso della richiesta	$request.path o ${request .path}	Il percorso della richiesta, senza il nome della fase.
Parametro del percorso	$request.path.*name* o ${request.path.*name*}	Il valore di un parametro del percorso nella richiesta. Ad esempio, se l'instradamento è `/pets/{petId}` , è possibile mappare il parametro `petId` della richiesta con `$request.path.petId` .
Variabile di contesto	$context.*variableName* o ${context.*variableName* }	Il valore di una variabile di contesto. **Note** Sono supportati solo i caratteri speciali `.` e `_`.
Variabile di fase	$stageVariables.*variableName* o ${stageVariables.*variableName* }	Il valore di una variabile di fase.
Valore statico	*stringa*	Un valore costante.

> **Note**
>
> Per utilizzare più variabili in un'espressione di selezione, racchiudere la variabile tra parentesi. Ad esempio, `${request.path.name} ${request.path.id}`.

Trasformazione delle risposte API

È possibile utilizzare i parametri di risposta per trasformare la risposta HTTP da un'integrazione back-end prima di restituire la risposta ai client. È possibile modificare le intestazioni o il codice di stato di una risposta prima che Gateway API restituisca la risposta ai client.

Configurare i parametri di risposta per ogni codice di stato restituito dall'integrazione. I parametri di risposta sono una mappa chiave-valore. La chiave identificherà la posizione del parametro di richiesta da modificare e la modalità per modificarlo. Il valore specifica i nuovi dati per il parametro.

Nella tabella seguente sono illustrate le chiavi supportate.

Tipo	Sintassi
Intestazione	`append|overwrite|remove:header.` *`headername`*
Codice di stato	`overwrite:statuscode`

Nella tabella seguente sono illustrati i valori supportati che è possibile mappare ai parametri.

Tipo	Sintassi	Note
Valore intestazione	$response.header.*name* o ${response.header.*name*}	I nomi delle intestazioni non fanno distinzione tra maiuscole e minuscole. API Gateway combina più valori di intestazione con virgole, ad esempio `"header1": "value1,value2"` . Alcune intestazioni sono riservate. Per

Tipo	Sintassi	Note
		ulteriori informazioni, consulta the section called “Intestazioni riservate”.
Corpo di risposta	$response.body.*name* o ${response.body.*name*}	Un'espressione di percorso JSON. Le espressioni di discesa ricorsiva (`$response.body..name`) e di filtro (`?(expression)`) non sono supportate. **Note** Quando si specifica un percorso JSON, API Gateway tronca il corpo della risposta a 100 KB e quindi applica l'espressione di selezione. Per inviare payload superiori a 100 KB, specificare `$response.body`.
Variabile di contesto	$context.*variableName* o ${context.*variableName*}	Valore di una variabile di contesto supportata.
Variabile di fase	$stageVariables.*variableName* o ${stageVariables.*variableName*}	Il valore di una variabile di fase.
Valore statico	*stringa*	Un valore costante.

> **Note**
>
> Per utilizzare più variabili in un'espressione di selezione, racchiudere la variabile tra parentesi. Ad esempio, `${request.path.name} ${request.path.id}`.

Intestazioni riservate

Le intestazioni seguenti sono riservate. Per queste intestazioni, non è possibile configurare le mappature di richiesta o di risposta.

- access-control-*
- apigw-*
- Autorizzazione
- Connessione
- Content-Encoding
- Content-Length
- Content-Location
- Forwarded
- Keep-Alive
- Origin
- Proxy-Authenticate
- Proxy-Authorization
- TE
- Trailers
- Transfer-Encoding
- Upgrade
- x-amz-*
- x-amzn-*
- X-Forwarded-For
- X-Forwarded-Host
- X-Forwarded-Proto
- Via

Esempi

Gli esempi della AWS CLI seguenti configurano le mappature dei parametri. Per modelli di esempio CloudFormation, vedi GitHub.

Aggiungere un'intestazione a una richiesta API

Il comando create-integration seguente crea un'intestazione denominata `header1` per una richiesta API prima che raggiunga l'integrazione del backend. API Gateway popola l'intestazione con l'ID di richiesta.

```
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" }'
```

Rinominare un'intestazione di richiesta

Il comando create-integration seguente rinomina l'intestazione di una richiesta da `header1` a `header2`:

```
aws apigatewayv2 create-integration \
    --api-id abcdef123 \
    --integration-type HTTP_PROXY \
    --payload-format-version 1.0 \
    --integration-uri 'https://api.example.com' \
    --integration-method ANY \
    --request-parameters '{ "append:header.header2": "$request.header.header1",
 "remove:header.header1": "''"}'
```

Modificare la risposta da un'integrazione

Il comando create-integration seguente configura i parametri di risposta per un'integrazione. Quando le integrazioni restituiscono un codice di stato 500, API Gateway modifica il codice di stato in 403 e aggiunge `header11` alla risposta. Quando l'integrazione restituisce un codice di stato 404, API Gateway aggiunge un'intestazione `error` alla risposta.

```
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"}  }'
```

Rimuovere le mappature dei parametri configurati

Il comando update-integration seguente rimuove i parametri di richiesta configurati in precedenza per `append:header.header1`. Il comando rimuove anche i parametri di risposta configurati in precedenza per un codice di stato 200.

```
aws apigatewayv2 update-integration \
    --api-id abcdef123 \
    --integration-id hijk456 \
    --request-parameters '{"append:header.header1" : ""}' \
    --response-parameters '{"200" : {}}'
```

Usa le definizioni OpenAPI per HTTP APIs in API Gateway

È possibile definire l'API HTTP utilizzando un file di definizione OpenAPI 3.0. Quindi è possibile importare la definizione in API Gateway per creare un'API. Per ulteriori informazioni sulle estensioni di API Gateway verso OpenAPI, consultare Estensioni OpenAPI.

Importazione di un'API HTTP

È possibile creare un'API HTTP importando un file di definizione OpenAPI 3.0.

Per eseguire la migrazione da un'API REST a un'API HTTP, è possibile esportare l'API REST come file di definizione OpenAPI 3.0. Quindi importare la definizione dell'API come un'API HTTP. Per ulteriori informazioni sull'esportazione di un'API REST, consultare Esportazione di un'API REST da API Gateway.

> **Note**
>
> HTTP APIs supporta le stesse AWS variabili di REST APIs. Per ulteriori informazioni, consulta AWS variabili per l'importazione OpenAPI.

Informazioni di convalida dell'importazione

Quando si importa un'API, API Gateway fornisce tre categorie di informazioni di convalida.

Informazioni

Una proprietà è valida in base alla specifica OpenAPI, ma tale proprietà non è supportata per HTTP. APIs

Ad esempio, il seguente frammento di codice OpenAPI 3.0 produce informazioni sull'importazione perché HTTP APIs non supporta la convalida delle richieste. API Gateway ignora i campi `requestBody` e `schema`.

```
"paths": {
  "/": {
    "get": {
      "x-amazon-apigateway-integration": {
        "type": "AWS_PROXY",
        "httpMethod": "POST",
        "uri": "arn:aws:lambda:us-east-2:123456789012:function:HelloWorld",
        "payloadFormatVersion": "1.0"
      },
      "requestBody": {
        "content": {
          "application/json": {
            "schema": {
              "$ref": "#/components/schemas/Body"
            }
          }
        }
      }
    }
  }
  ...
},
"components": {
  "schemas": {
    "Body": {
      "type": "object",
      "properties": {
        "key": {
          "type": "string"
        }
```

```
      }
    }
    ...
  }
  ...
}
```

Attenzione

Una proprietà o una struttura non è valida in base alla specifica OpenAPI, ma non blocca la creazione di API. È possibile specificare se l'API Gateway deve ignorare questi avvisi e continuare a creare l'API o interrompere la creazione dell'API n caso di presenza di avvisi.

Il seguente documento OpenAPI 3.0 produce avvisi sull'importazione perché HTTP supporta APIs solo le integrazioni del proxy Lambda e del proxy HTTP.

```
"x-amazon-apigateway-integration": {
  "type": "AWS",
  "httpMethod": "POST",
  "uri": "arn:aws:lambda:us-east-2:123456789012:function:HelloWorld",
  "payloadFormatVersion": "1.0"
}
```

Errore

La specifica OpenAPI non è valida o è malformata. API Gateway non è in grado di creare alcuna risorsa dal documento malformato. È necessario correggere gli errori e quindi riprovare.

La seguente definizione dell'API produce errori durante l'importazione perché HTTP APIs supporta solo la specifica OpenAPI 3.0.

```
{
  "swagger": "2.0.0",
  "info": {
    "title": "My API",
    "description": "An Example OpenAPI definition for Errors/Warnings/ImportInfo",
    "version": "1.0"
  }
  ...
}
```

Come altro esempio, mentre OpenAPI consente agli utenti di definire un'API con più requisiti di sicurezza associati a una particolare operazione, API Gateway non supporta questa operazione. Ogni operazione può avere solo un'autorizzazione IAM, un provider di autorizzazioni Lambda o un provider di autorizzazioni JWT. Il tentativo di modellare più requisiti di sicurezza genera un errore.

Importa un'API utilizzando il AWS CLI

Il comando import-api seguente importa il file di definizione OpenAPI 3.0 `api-definition.json` come API HTTP:

Example

```
aws apigatewayv2 import-api --body file://api-definition.json
```

Example

È possibile importare la seguente definizione OpenAPI 3.0 di esempio per creare un'API HTTP.

```
{
  "openapi": "3.0.1",
  "info": {
    "title": "Example Pet Store",
    "description": "A Pet Store API.",
    "version": "1.0"
  },
  "paths": {
    "/pets": {
      "get": {
        "operationId": "GET HTTP",
        "parameters": [
          {
            "name": "type",
            "in": "query",
            "schema": {
              "type": "string"
            }
          },
          {
            "name": "page",
            "in": "query",
            "schema": {
              "type": "string"
```

```
          }
        }
      ],
      "responses": {
        "200": {
          "description": "200 response",
          "headers": {
            "Access-Control-Allow-Origin": {
              "schema": {
                "type": "string"
              }
            }
          },
          "content": {
            "application/json": {
              "schema": {
                "$ref": "#/components/schemas/Pets"
              }
            }
          }
        }
      },
      "x-amazon-apigateway-integration": {
        "type": "HTTP_PROXY",
        "httpMethod": "GET",
        "uri": "http://petstore.execute-api.us-west-1.amazonaws.com/petstore/pets",
        "payloadFormatVersion": 1.0
      }
    },
    "post": {
      "operationId": "Create Pet",
      "requestBody": {
        "content": {
          "application/json": {
            "schema": {
              "$ref": "#/components/schemas/NewPet"
            }
          }
        },
        "required": true
      },
      "responses": {
        "200": {
          "description": "200 response",
```

```
            "headers": {
              "Access-Control-Allow-Origin": {
                "schema": {
                  "type": "string"
                }
              }
            },
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/NewPetResponse"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "HTTP_PROXY",
          "httpMethod": "POST",
          "uri": "http://petstore.execute-api.us-west-1.amazonaws.com/petstore/pets",
          "payloadFormatVersion": 1.0
        }
      }
    },
    "/pets/{petId}": {
      "get": {
        "operationId": "Get Pet",
        "parameters": [
          {
            "name": "petId",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "headers": {
              "Access-Control-Allow-Origin": {
                "schema": {
                  "type": "string"
```

```
                }
              }
            },
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Pet"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "HTTP_PROXY",
          "httpMethod": "GET",
          "uri": "http://petstore.execute-api.us-west-1.amazonaws.com/petstore/pets/
{petId}",
          "payloadFormatVersion": 1.0
        }
      }
    }
  },
  "x-amazon-apigateway-cors": {
    "allowOrigins": [
      "*"
    ],
    "allowMethods": [
      "GET",
      "OPTIONS",
      "POST"
    ],
    "allowHeaders": [
      "x-amzm-header",
      "x-apigateway-header",
      "x-api-key",
      "authorization",
      "x-amz-date",
      "content-type"
    ]
  },
  "components": {
    "schemas": {
      "Pets": {
        "type": "array",
```

```
        "items": {
          "$ref": "#/components/schemas/Pet"
        }
      },
      "Empty": {
        "type": "object"
      },
      "NewPetResponse": {
        "type": "object",
        "properties": {
          "pet": {
            "$ref": "#/components/schemas/Pet"
          },
          "message": {
            "type": "string"
          }
        }
      },
      "Pet": {
        "type": "object",
        "properties": {
          "id": {
            "type": "string"
          },
          "type": {
            "type": "string"
          },
          "price": {
            "type": "number"
          }
        }
      },
      "NewPet": {
        "type": "object",
        "properties": {
          "type": {
            "$ref": "#/components/schemas/PetType"
          },
          "price": {
            "type": "number"
          }
        }
      },
      "PetType": {
```

```
        "type": "string",
        "enum": [
          "dog",
          "cat",
          "fish",
          "bird",
          "gecko"
        ]
      }
    }
  }
}
```

Esporta HTTP APIs da API Gateway

Dopo aver creato un'API HTTP, è possibile esportare una definizione OpenAPI 3.0 della propria API da API Gateway. È possibile scegliere una fase da esportare o esportare l'ultima configurazione dell'API. È inoltre possibile importare una definizione API esportata in API Gateway per creare un'altra API identica. Per ulteriori informazioni sull'importazione delle definizioni API, consultare Importazione di un'API HTTP.

Esportazione di una definizione OpenAPI 3.0 di uno stage utilizzando la CLI AWS

Il comando export-api seguente esporta una definizione OpenAPI di una fase API denominata `prod` in un file YAML denominato `stage-definition.yaml`. Per impostazione predefinita il file di definizione esportato include le estensioni API Gateway.

```
aws apigatewayv2 export-api \
    --api-id api-id  \
    --output-type YAML  \
    --specification OAS30 \
    --stage-name prod \
    stage-definition.yaml
```

Esporta una definizione OpenAPI 3.0 delle ultime modifiche della tua API utilizzando la CLI AWS

Il comando export-api seguente esporta una definizione OpenAPI di un'API HTTP in un file JSON denominato `latest-api-definition.json`. Poiché il comando non specifica una fase, API Gateway esporta la configurazione più recente dell'API, indipendentemente dal fatto che sia stata distribuita in una fase o meno. Il file di definizione esportato non include le estensioni API Gateway.

```
aws apigatewayv2 export-api \
```

```
    --api-id api-id  \
    --output-type JSON  \
    --specification OAS30 \
    --no-include-extensions \
    latest-api-definition.json
```

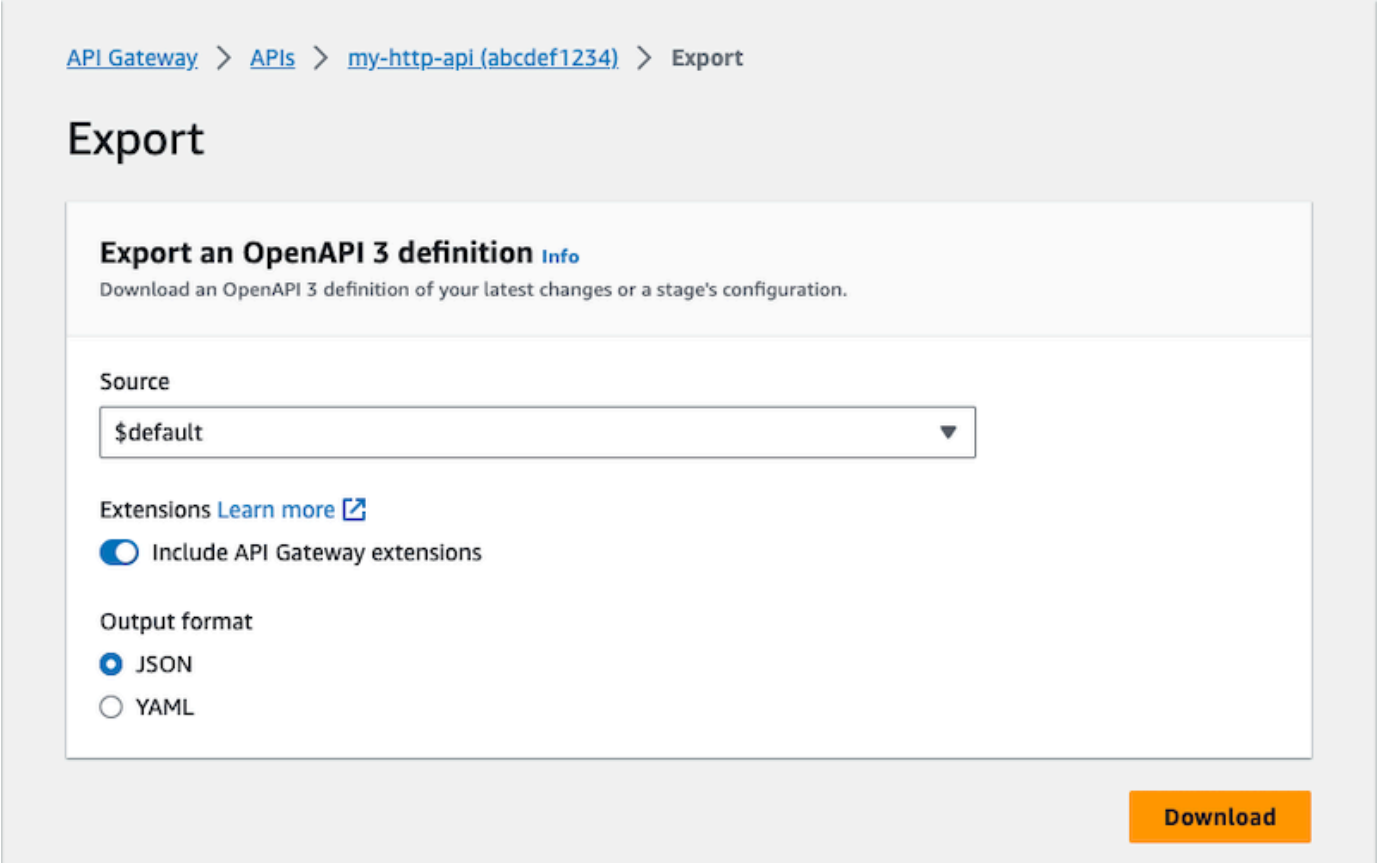
Per ulteriori informazioni, consulta ExportAPI nella Guida di riferimento per le API di Amazon API Gateway Versione 2.

Esportazione di una definizione OpenAPI 3.0 utilizzando la console Gateway API

La seguente procedura mostra come esportare una definizione OpenAPI di un'API HTTP.

Esportazione di una definizione OpenAPI 3.0 utilizzando la console Gateway API

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere un'API HTTP.
3. Nel pannello di navigazione principale, in Sviluppa, scegli Esporta.
4. Seleziona una delle seguenti opzioni per esportare la tua API:



a. In Origine, seleziona un'origine per la definizione OpenAPI 3.0. Puoi scegliere una fase da esportare o esportare l'ultima configurazione dell'API.

b. Attiva Includi le estensioni API Gateway per includere le estensioni Gateway API.

c. In Formato di output, seleziona un formato di output.

5. Scegli Scarica.

Pubblicazione di API HTTP che i clienti possono invocare

Puoi utilizzare fasi e nomi di dominio personalizzati per pubblicare l'API richiamabile dai client.

Una fase API è un riferimento logico a uno stato del ciclo di vita dell'API (ad esempio, `dev`, `prod`, `beta` o `v2`). Ogni fase è un riferimento con nome a una distribuzione dell'API e viene resa disponibile per le applicazioni client da chiamare. Puoi configurare integrazioni e impostazioni differenti per ogni fase di un'API.

Puoi utilizzare nomi di dominio personalizzati per fornire un URL più semplice e intuitivo all'API richiamabile dai client, rispetto all'URL predefini, `https://`*`api-id`*`.execute-api.`*`region`*`.amazonaws.com/`*`stage`*.

> **Note**
>
> Per aumentare la sicurezza delle API API Gateway, il dominio `execute-api.{`*`region`*`}.amazonaws.com`è registrato nella Public Suffix List (PSL). Per una maggiore sicurezza, consigliamo di utilizzare i cookie con un prefisso `__Host-` se hai bisogno di impostare cookie sensibili nel nome di dominio predefinito per le API API Gateway. Questa pratica ti aiuterà a difendere il tuo dominio dai tentativi CSRF (cross-site request forgery). Per ulteriori informazioni, consulta la pagina Impostazione cookie nella pagina Mozilla Developer Network.

Argomenti

- Fasi per HTTP APIs in API Gateway
- Politica di sicurezza per HTTP APIs in API Gateway
- Nomi di dominio personalizzati per API HTTP in Gateway API

Fasi per HTTP APIs in API Gateway

Una fase API è un riferimento logico a uno stato del ciclo di vita dell'API (ad esempio, `dev`, `prod`, `beta` o `v2`). Le fasi API sono identificate dal rispettivo ID API e dal nome della fase e sono incluse nell'URL utilizzato per richiamare l'API. Ogni fase è un riferimento con nome a una distribuzione dell'API e viene resa disponibile per le applicazioni client da chiamare.

È possibile creare una fase `$default` che viene servita dalla base dell'URL dell'API, ad esempio `https://{api_id}.execute-api.{region}.amazonaws.com/`. Utilizzare questo URL per richiamare una fase API.

Una distribuzione è uno snapshot della configurazione dell'API. Dopo essere stata distribuita a una fase, l'API è disponibile per i client da richiamare. È necessario distribuire un'API per attivare le modifiche apportate. Se si abilitano le distribuzioni automatiche, le modifiche apportate a un'API vengono rilasciate automaticamente.

Usa le variabili di fase per HTTP APIs in API Gateway

Le variabili di fase sono coppie chiave-valore che è possibile definire per una fase di un'API HTTP. Fungono da variabili di ambiente e possono essere utilizzate nella configurazione dell'API.

Le variabili di fase non sono destinate ad essere utilizzate per i dati sensibili, come le credenziali. Per passare dati sensibili alle integrazioni, usa un AWS Lambda autorizzatore. È possibile passare dati sensibili alle integrazioni nell'output del provider di autorizzazioni Lambda. Per ulteriori informazioni, consulta the section called “Formato della risposta dell'autorizzazione”.

Esempio: utilizzo di una variabile di fase per personalizzare l'endpoint di integrazione HTTP

Ad esempio, puoi definire una variabile di fase e quindi impostare il suo valore come un endpoint HTTP per un'integrazione proxy HTTP. Successivamente, puoi fare riferimento all'endpoint utilizzando il nome della variabile di fase associata. In questo modo, puoi utilizzare la stessa configurazione API con un endpoint diverso in ogni fase. Allo stesso modo, puoi utilizzare le variabili di fase per specificare un'integrazione di AWS Lambda funzioni diversa per ogni fase dell'API.

Per utilizzare una variabile di fase per personalizzare l'endpoint di integrazione HTTP, è necessario innanzitutto impostare il nome e il valore della variabile di fase (ad esempio `url`) con un valore pari a `example.com`. Quindi, impostare un'integrazione proxy HTTP. Anziché inserire l'URL dell'endpoint, è possibile comunicare ad API Gateway di usare il valore della variabile di fase, ossi, **`http://${stageVariables.url}`**. Questo valore indica ad API Gateway di sostituire la variabile di fase `${}` al runtime, a seconda della fase dell'API.

È possibile fare riferimento alle variabili di fase in modo simile per specificare il nome di una funzione Lambda o un ruolo AWS ARN.

Quando si specifica un nome di funzione Lambda come valore della variabile di fase, è necessario configurare manualmente le autorizzazioni per la funzione Lambda. Il comando add-permission seguente configura l'autorizzazione per la funzione Lambda:

```
aws lambda add-permission --function-name arn:aws:lambda:XXXXXX:your-lambda-function-
name --source-arn arn:aws:execute-api:us-east-1:YOUR_ACCOUNT_ID:api_id/*/HTTP_METHOD/
resource --principal apigateway.amazonaws.com --statement-id apigateway-access --action
 lambda:InvokeFunction
```

Riferimento alle variabili di fase API Gateway per HTTP APIs in API Gateway

È possibile utilizzare le variabili di fase API Gateway per HTTP APIs nei seguenti casi.

Integrazione HTTP URIs

Puoi utilizzare una variabile di fase come parte di un URI di integrazione HTTP, come mostrato negli esempi seguenti.

- Un URI completo senza protocoll – `http://${stageVariables.<variable_name>}`
- Un dominio completo – `http://${stageVariables.<variable_name>}/resource/operation`
- Un sottodominio – `http://${stageVariables.<variable_name>}.example.com/resource/operation`
- Un percorso – `http://example.com/${stageVariables.<variable_name>}/bar`
- Una stringa di query – `http://example.com/foo?q=${stageVariables.<variable_name>}`

Funzioni Lambda

Puoi utilizzare una variabile di fase al posto di un nome di integrazione di funzione o alias Lambda, come illustrato negli esempi seguenti.

- `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**
>
> Per utilizzare una variabile di fase per una funzione Lambda, la funzione deve essere nello stesso account dell'API. Le variabili di fase non supportano le funzioni Lambda tra più account.

AWS credenziali di integrazione

È possibile utilizzare una variabile stage come parte di un ARN di credenziali AWS utente o di ruolo, come illustrato nell'esempio seguente.

- `arn:aws:iam::<account_id>:${stageVariables.<variable_name>}`

Politica di sicurezza per HTTP APIs in API Gateway

Gateway API applica la policy di sicurezza `TLS_1_2` per tutti gli endpoint delle API HTTP.

Una policy di sicurezza è una combinazione predefinita di una versione TLS minima e un pacchetto di crittografia offerta da Gateway Amazon API. Il protocollo TLS affronta i problemi di sicurezza della rete, ad esempio manomissioni e intercettazioni tra un client e un server. Quando i client stabiliscono un handshake TLS sull'API tramite il dominio personalizzato, la policy di sicurezza applica la versione di TLS e le opzioni del pacchetto di crittografia che i client possono scegliere di utilizzare. Questa policy di sicurezza accetta il traffico TLS 1.2 e TLS 1.3 e rifiuta il traffico TLS 1.0.

Protocolli e cifrari TLS supportati per HTTP APIs

La tabella seguente descrive i protocolli TLS supportati per HTTP. APIs

Protocolli TLS	Policy di sicurezza TLS_1_2
TLSv13.	Sì

Protocolli TLS	Policy di sicurezza TLS_1_2
TLSv12.	Sì

La tabella seguente descrive i codici TLS disponibili per la politica di sicurezza TLS 1_2 per HTTP. APIs

Crittografie TLS	Policy di sicurezza TLS_1_2
TLS-AES-128-GCM- SHA256	Sì
TLS-AES-256-GCM- SHA384	Sì
TLS- - - CHACHA20 POLY1305 SHA256	Sì
ECDHE-ECDSA- -GCM- AES128 SHA256	Sì
ECDH-RSA- AES128 -GCM- SHA256	Sì
ECDHE-ECSA AES128 - - SHA256	Sì
ECDHE-RSA- - AES128 SHA256	Sì
ECDHE-ECDSA- -GCM AES256 - SHA384	Sì
ECDH-RSA- AES256 -GCM- SHA384	Sì
ECDHE-ECSA AES256 - - SHA384	Sì
ECDHE-RSA- - AES256 SHA384	Sì
AES128-GCM- SHA256	Sì
AES128-SHA256	Sì
AES256-GCM- SHA384	Sì
AES256-SHA256	Sì

OpenSSL e nomi crittografia RFC

OpenSSL e IETF RFC 5246 utilizzano nomi diversi per le stesse crittografie. Per l'elenco dei nomi delle crittografie, consulta the section called “OpenSSL e nomi crittografia RFC”.

Informazioni su REST e APIs WebSocket APIs

Per ulteriori informazioni su REST APIs e WebSocket APIs, vedere the section called “Scelta di una policy di sicurezza” ethe section called “Politica di sicurezza per WebSocket APIs API Gateway”.

Nomi di dominio personalizzati per API HTTP in Gateway API

I nomi di dominio personalizzati sono URL più semplici e più intuitivi che è possibile fornire agli utenti delle API.

Dopo aver distribuito un'API, tu e i tuoi clienti potete richiamarla usando l'URL di base predefinito nel formato seguente:

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

dove *`api-id`* è generato da Gateway API, *`region`* è la Regione AWS e *`stage`* è il valore che è stato specificato al momento dell'implementazione dell'API.

La parte del nome host dell'URL, *`api-id`*`.execute-api.`*`region`*`.amazonaws.com`, fa riferimento a un endpoint API. Il nome dell'endpoint API predefinito viene generato casualmente, è complesso da richiamare e non è intuitivo.

Con i nomi di dominio personalizzati, è possibile impostare il nome host dell'API e scegliere un percorso base (ad esempio `myservice`) per mappare l'URL alternativo all'API. Ad esempio, un URL di base dell'API più intuitivo può diventare:

```
https://api.example.com/myservice
```

Considerazioni

Le seguenti considerazioni potrebbero influire sull'utilizzo di un nome di dominio personalizzato.

- Un nome di dominio personalizzato regionale può essere associato alle REST API e alle API HTTP. È possibile utilizzare le API di Gateway API Versione 2 per creare e gestire nomi di dominio personalizzati regionali per le REST API.

- L'unica versione minima di TLS supportata è TLS 1.2.
- È necessario creare o aggiornare il record di risorse del provider DNS per eseguire la mappatura all'endpoint API. In assenza di questa mappatura, le richieste API destinate al nome di dominio personalizzato non possono raggiungere API Gateway.
- È possibile supportare un numero pressoché infinito di nomi di dominio senza superare la quota predefinita con un certificato jolly. Per ulteriori informazioni, consulta the section called “Nomi di dominio personalizzati con caratteri jolly”.

Prerequisiti

Di seguito sono riportati i prerequisiti per la creazione di un nome di dominio personalizzato.

Registrare un nome di dominio

Per configurare nomi di dominio personalizzati per le API, devi avere un nome di dominio Internet registrato. Puoi registrare un nome di dominio Internet utilizzando Amazon Route 53 oppure un registrar di dominio di terze parti a tua scelta. Il nome di dominio personalizzato può essere il nome di un sottodominio o del dominio root (detto anche "apex di zona") di un dominio Internet registrato.

Il nome di dominio deve seguire la specifica RFC 1035 e può avere un massimo di 63 ottetti per etichetta e 255 ottetti in totale.

Certificati per nomi di dominio personalizzati

Prima di configurare un nome di dominio personalizzato per un'API, è necessario che sia già presente un certificato SSL/TLS in ACM. Se ACM non è disponibile nella Regione AWS in cui si sta creando il nome di dominio personalizzato, è necessario importare nella Regione un certificato in Gateway API.

Per importare un certificato SSL/TLS, devi fornire il corpo del certificato SSL/TLS in formato PEM, la sua chiave privata e la catena di certificati per il nome di dominio personalizzato.

Ogni certificato archiviato in ACM è identificato dal relativo ARN. Con i certificati emessi da ACM, non devi preoccuparti di esporre dettagli sensibili del certificato, come la chiave privata. Per usare un certificato gestito da AWS per un nome di dominio, devi semplicemente fare riferimento al relativo ARN.

Se l’applicazione utilizza il blocco dei certificati, talvolta denominato associazione SSL, per associare un certificato ACM, è possibile che l'applicazione non riesca a connettersi al tuo dominio dopo che

AWS ha rinnovato il certificato. Per ulteriori informazioni, consulta Probemi nel blocco dei certificati nella Guida per l'utente di AWS Certificate Manager.

Nomi di dominio personalizzati con caratteri jolly

Con i nomi di dominio personalizzati con caratteri jolly, è possibile supportare un numero quasi infinito di nomi di dominio senza superare la quota predefinita. Ad esempio, è possibile dare a ciascuno dei clienti il proprio nome di dominio, *customername*`.api.example.com`.

Per creare un nome di dominio personalizzato con caratteri jolly, specificare un carattere jolly (`*`) come primo sottodominio di un dominio personalizzato che rappresenta tutti i possibili sottodomini di un dominio radice.

Ad esempio, il nome di dominio personalizzato con caratteri jolly `*.example.com` genera sottodomini quali `a.example.com`, `b.example.com` e `c.example.com`, indirizzati tutti allo stesso dominio.

I nomi di dominio personalizzati con caratteri jolly supportano configurazioni distinte dai nomi di dominio personalizzati standard di API Gateway. Ad esempio, in un singolo account AWS, è possibile configurare `*.example.com` e `a.example.com` affinché si comportino in modo diverso.

Per creare un nome di dominio personalizzato con caratteri jolly, è necessario fornire un certificato emesso da ACM che sia stato convalidato utilizzando il DNS o il metodo di convalida della posta elettronica.

> **Note**
>
> Non è possibile creare un nome di dominio personalizzato con caratteri jolly se un account AWS diverso ha creato un nome di dominio personalizzato in conflitto con il nome di dominio personalizzato con caratteri jolly. Ad esempio, se l'account A ha creato `a.example.com`, l'account B non può creare il nome di dominio personalizzato con caratteri jolly `*.example.com`.
> Se l'account A e l'account B condividono un proprietario, è possibile contattare il Centro assistenza AWS per richiedere un'eccezione.

Passaggi successivi per nomi di dominio personalizzati

Per configurare un nome di dominio personalizzato per un'API HTTP, consulta la documentazione disponibile nella sezione REST API della Guida per gli sviluppatori di Gateway API.

Per prima cosa, si specifica un certificato per il nome di dominio personalizzato. Per ulteriori informazioni, consulta the section called “Prepara i certificati in AWS Certificate Manager”. Successivamente, si crea un nome di dominio personalizzato regionale. Per ulteriori informazioni, consulta the section called “Configurazione di un nome di dominio personalizzato regionale”.

Mappatura delle fasi API a un nome di dominio personalizzato per le API HTTP

È possibile utilizzare le mappature API per connettere le fasi API a un nome di dominio personalizzato. Dopo aver creato un nome di dominio e aver configurato i record DNS, è possibile utilizzare le mappature API per inviare il traffico alle API tramite il nome di dominio personalizzato.

Una mappatura API specifica un'API, una fase e, facoltativamente, un percorso da utilizzare per la mappatura. Ad esempio, è possibile mappare la fase `production` di un'API su `https://api.example.com/orders`.

È possibile mappare le fasi HTTP e API REST allo stesso nome di dominio personalizzato.

Prima di creare una mappatura API, è necessario disporre di un'API, di una fase e di un nome di dominio personalizzato. Per ulteriori informazioni sulla creazione di un nome di dominio personalizzato, consulta the section called “Configurazione di un nome di dominio personalizzato regionale”.

Routing delle richieste API

È possibile configurare le mappature API con più livelli, ad esempio `orders/v1/items` e `orders/v2/items`.

Per mappature API con più livelli, API Gateway instrada le richieste alla mappatura API che ha il percorso corrispondente più lungo. API Gateway considera solo i percorsi configurati per le mappature API, e non i percorsi API, per selezionare l'API da richiamare. Se nessun percorso corrisponde alla richiesta, API Gateway invia la richiesta all'API mappata al percorso vuoto (`none`).

Per i nomi di dominio personalizzati che usano mappature API con più livelli, API Gateway instrada le richieste alla mappatura API che ha il percorso corrispondente più lungo.

Ad esempio, se si considera un nome di dominio personalizzato `https://api.example.com` con le seguenti mappature API:

1. (`none`) mappato all'API 1.
2. `orders` mappato all'API 2.
3. `orders/v1/items` mappato all'API 3.

4. `orders/v2/items` mappato all'API 4.

5. `orders/v2/items/categories` mappato all'API 5.

Richiesta	API selezionata	Spiegazione
`https://api.example.com/orders`	`API 2`	La richiesta corrisponde esattamente a questa mappatura API.
`https://api.example.com/orders/v1/items`	`API 3`	La richiesta corrisponde esattamente a questa mappatura API.
`https://api.example.com/orders/v2/items`	`API 4`	La richiesta corrisponde esattamente a questa mappatura API.
`https://api.example.com/orders/v1/items/123`	`API 3`	API Gateway sceglie la mappatura con il percorso corrispondente più lungo. `123` alla fine della richiesta non influisce sulla selezione.
`https://api.example.com/orders/v2/items/categories/5`	`API 5`	API Gateway sceglie la mappatura con il percorso corrispondente più lungo.
`https://api.example.com/customers`	`API 1`	API Gateway utilizza la mappatura vuota come catch-all.
`https://api.example.com/ordersandmore`	`API 2`	API Gateway sceglie la mappatura con il prefisso corrispondente più lungo. Per un nome di dominio personalizzato configurato con mappature a livello singolo,

Richiesta	API selezionata	Spiegazione
		ad esempio solo `https://api.example.com/orders` e `https://api.example.com/`, API Gateway sceglie API 1, poiché non esiste un percorso corrispondente con `ordersandmore`.

Restrizioni

- In una mappatura API, il nome di dominio personalizzato e le API mappate devono trovarsi nello stesso account AWS.
- Le mappature API devono contenere solo lettere, numeri e i seguenti caratteri: `$-_.+!*'()/`.
- La lunghezza massima per il percorso in una mappatura API è di 300 caratteri.
- È possibile disporre di 200 mappature API con più livelli per ogni nome di dominio. Questo limite non include le mappature API con livelli singoli, ad esempio `/prod`.
- È possibile mappare le API HTTP a un nome di dominio personalizzato regionale solo con la policy di sicurezza TLS 1.2.
- Non è possibile mappare le API WebSocket allo stesso nome di dominio personalizzato di un'API HTTP o un'API REST.
- Se si crea una mappatura API con più livelli, Gateway API converte tutti i nomi di intestazione in lettere minuscole.

Creare una mappatura API

Per creare una mappatura API, innanzitutto è necessario creare un nome di dominio personalizzato, un'API e una fase. Per informazioni sulla creazione di un nome di dominio personalizzato, consulta the section called “Configurazione di un nome di dominio personalizzato regionale”.

Per modelli di esempio di AWS Serverless Application Model che creano tutte le risorse, consulta Sessioni con SAM su GitHub.

Console di gestione AWS

Per creare una mappatura API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere Nomi di dominio personalizzati.
3. Selezionare un nome di dominio personalizzato già creato.
4. Scegliere API mappings (mappature API).
5. Scegliere Configure API mappings (Configura mappature API).
6. Scegliere Add new mapping (Aggiungi nuova mappatura).
7. Immettere un'API, uno Stage (Fase)e, facoltativamente, un Path (Percorso).
8. Selezionare Salva.

AWS CLI

Il comando create-api-mapping seguente crea una mappatura API. In questo esempio, API Gateway invia le richieste `api.example.com/v1/orders` all'API e alla fase specificate.

```
aws apigatewayv2 create-api-mapping \
    --domain-name api.example.com \
    --api-mapping-key v1/orders \
    --api-id a1b2c3d4 \
    --stage test
```

CloudFormation

Il seguente esempio di CloudFormation consente di creare una mappatura API.

```
MyApiMapping:
  Type: 'AWS::ApiGatewayV2::ApiMapping'
  Properties:
    DomainName: api.example.com
    ApiMappingKey: 'orders/v2/items'
    ApiId: !Ref MyApi
    Stage: !Ref MyStage
```

Disabilitazione dell'endpoint predefinito per API HTTP

Per impostazione predefinita, i client possono richiamare l'API utilizzando l'endpoint `execute-api` generato da API Gateway per l'API. Per garantire che i client possano accedere all'API solo utilizzando un nome di dominio personalizzato con l'autenticazione TLS reciproca, disattivare l'endpoint `execute-api` predefinito. Quando si disattiva l'endpoint predefinito, questa operazione influisce su tutte le fasi di un'API.

La seguente procedura mostra come disabilitare l'endpoint predefinito per un'API HTTP.

Console di gestione AWS

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere un'API HTTP.
3. Scegli l'ID dell'API per aprire la pagina Dettagli API.
4. In Dettagli API seleziona Modifica.
5. Per Endpoint predefinito seleziona Disabilita.
6. Selezionare Salva.

   Se si attivano le implementazioni automatiche per la fase, non è necessario implementare nuovamente l'API per rendere effettiva la modifica. In caso contrario, è necessario implementare nuovamente l'API.
7. (Facoltativo) Scegli Distribuzione, quindi implementa nuovamente l'API o crea una nuova fase per rendere effettiva la modifica.

AWS CLI

Il comando update-domain-name seguente disabilita l'endpoint predefinito per un'API HTTP:

```
aws apigatewayv2 update-api \
    --api-id abcdef123 \
    --disable-execute-api-endpoint
```

Dopo aver disattivato l'endpoint predefinito, è necessario distribuire l'API per rendere effettiva la modifica, a meno che non siano abilitate le distribuzioni automatiche.

Il seguente comando create-deployment crea l'implementazione:

```
aws apigatewayv2 create-deployment \
    --api-id abcdef123 \
    --stage-name dev
```

Tipi di indirizzo IP per nomi di dominio personalizzati per API HTTP

Quando si crea un'API, è possibile specificare il tipo di indirizzo IP che può invocare il dominio. È possibile scegliere IPv4 per risolvere gli indirizzi IPv4 per invocare il dominio oppure dualstack per consentire a entrambi gli indirizzi IPv4 e IPv6 di invocare il dominio. È consigliabile impostare il tipo di indirizzo IP su dualstack per ridurre l'esaurimento dello spazio IP o per il livello di sicurezza. Per ulteriori informazioni sui vantaggi del tipo di indirizzo IP dualstack, consultare IPv6 on AWS.

Considerazioni sui tipi di indirizzo IP

Le seguenti considerazioni potrebbero influire sull'utilizzo dei tipi di indirizzo IP.

- Il tipo di indirizzo IP predefinito per i nomi di dominio personalizzati di Gateway API è IPv4.
- Per il nome di dominio personalizzato, non è necessario che tutte le API mappate ad esso abbiano lo stesso tipo di indirizzo IP. Se si disabilita l'endpoint API predefinito, si potrebbe influire sul modo in cui i chiamanti possono invocare l'API.

Modifica del tipo di indirizzo IP di un nome di dominio personalizzato

È possibile modificare il tipo di indirizzo IP aggiornando la configurazione dell'endpoint del dominio. È possibile aggiornare la configurazione dell'endpoint del dominio utilizzando la Console di gestione AWS, AWS CLI, CloudFormation o un AWS SDK.

Console di gestione AWS

Per modificare il tipo di indirizzo IP di un nome di dominio personalizzato

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere un nome di dominio personalizzato pubblico.
3. Scegliere Configurazione endpoint.
4. Per Tipo di indirizzo IP, selezionare IPv4 o Dualstack.
5. Selezionare Salva.

AWS CLI

Il comando update-domain-name seguente aggiorna un'API in modo che abbia un tipo di indirizzo IP dualstack:

```
aws apigatewayv2 update-domain-name \
   --domain-name dualstack.example.com \
   --domain-name-configurations CertificateArn=arn:aws:acm:us-
east-1:111122223333:certificate/abcd1234-5678-abc,IpAddressType=dualstack
```

L'output sarà simile al seguente:

```
{
    "ApiMappingSelectionExpression": "$request.basepath",
    "DomainName": "dualstack.example.com",
    "DomainNameConfigurations": [
        {
            "ApiGatewayDomainName": "d-abcd1234.execute-api.us-
east-1.amazonaws.com",
            "CertificateArn": "arn:aws:acm:us-east-1:111122223333:certificate/
abcd1234-5678-abc",
            "DomainNameStatus": "AVAILABLE",
            "EndpointType": "REGIONAL",
            "HostedZoneId": "Z3LQWSYCGH4ADY",
            "SecurityPolicy": "TLS_1_2",
            "IpAddressType": "dualstack"
        }
    ],
    "Tags": {}
}
```

Protezione delle API HTTP in Gateway API

API Gateway fornisce diversi modi per proteggere l'API da determinate minacce, ad esempio utenti malintenzionati o picchi di traffico. È possibile proteggere l'API usando strategie come l'impostazione delle destinazioni di limitazione e l'abilitazione dell'autenticazione TLS reciproca. In questa sezione viene descritto come abilitare queste funzionalità utilizzando API Gateway.

Argomenti

- Limita le richieste al tuo HTTP APIs per una migliore velocità di trasmissione in API Gateway

- Come attivare l'autenticazione TLS reciproca per il tuo HTTP APIs in API Gateway

Limita le richieste al tuo HTTP APIs per una migliore velocità di trasmissione in API Gateway

Puoi configurare il throttling per APIs evitare che vengano sopraffatti da troppe richieste. Le limitazioni della larghezza di banda della rete sono applicate sulla base del massimo sforzo e dovrebbero essere considerate come obiettivi e non limiti massimi garantiti delle richieste.

API Gateway limita la larghezza di banda della rete delle richieste nell'API mediante l'algoritmo di token bucket, dove un token rappresenta una richiesta. In particolare, API Gateway esamina la frequenza e il numero di richieste inviate rispetto APIs a tutti gli utenti del tuo account, per regione. Nell'algoritmo di bucket token, un picco può consentire il superamento predefinito di tali limiti, ma anche altri fattori possono causare il superamento dei limiti in alcuni casi.

Quando le richieste inviate superano i limiti impostati per il tasso a regime e per i limiti di picco, API Gateway inizia a limitare le richieste. I clienti potrebbero ricevere risposte agli errori `429 Too Many Requests` a questo punto. Quando rileva queste eccezioni, il client può reinviare le richieste non riuscite in modo da limitare la velocità.

In qualità di sviluppatore di API, puoi impostare i limiti target per le singole fasi o percorsi dell'API per migliorare le prestazioni complessive di tutto APIs il tuo account.

Limitazione a livello di account per regione

Per impostazione predefinita, API Gateway limita le richieste allo stato stazionario (RPS) APIs all'interno di un AWS account, per regione. Inoltre, limita il burst (ovvero la dimensione massima del bucket) per tutti gli elementi APIs all'interno di un account, per regione. AWS In API Gateway, il limite dei picchi rappresenta il numero massimo di invii di richieste che API Gateway può gestire prima di restituire risposte di errore `429 Too Many Requests`. Per ulteriori informazioni sulla limitazione delle quote, vedere Quote e note importanti.

I limiti per account vengono applicati a tutti i membri di un account APIs in una regione specificata. Il limite di tariffa a livello di account può essere aumentato su richiesta: sono possibili limiti più elevati con APIs timeout più brevi e carichi utili inferiori. Per richiedere un aumento dei limiti di limitazione a livello di account per Regione, contatta il Centro assistenza AWS. Per ulteriori informazioni, consulta Quote e note importanti. Tieni presente che questi limiti non possono essere superiori ai limiti di limitazione. AWS

Throttling a livello di instradamento

Puoi impostare il throttling a livello di route per sostituire i limiti di throttling delle richieste a livello di account per una fase specifica o per singoli route nell'API. I limiti della limitazione (della larghezza di banda della rete) della route di default non possono superare i limiti di velocità a livello di account.

È possibile configurare la limitazione a livello di percorso utilizzando AWS CLI. Il comando update-stage seguente configura la limitazione (della larghezza di banda della rete) personalizzata per la fase e l'instradamento specificati di un'API:

```
aws apigatewayv2 update-stage \
    --api-id a1b2c3d4 \
    --stage-name dev \
    --route-settings '{"GET /pets":
{"ThrottlingBurstLimit":100,"ThrottlingRateLimit":2000}}'
```

Come attivare l'autenticazione TLS reciproca per il tuo HTTP APIs in API Gateway

L'autenticazione TLS reciproca richiede l'autenticazione bidirezionale tra il client e il server. Con l'autenticazione TLS reciproca, i client devono presentare certificati X.509 per verificare la propria identità per accedere all'API. Il TLS reciproco è un requisito comune per l'Internet of Things (IoT) e business-to-business le applicazioni.

È possibile utilizzare l'autenticazione TLS reciproca insieme ad altre operazioni di autorizzazione e autenticazione supportate da API Gateway. API Gateway inoltra i certificati forniti dai client alle autorizzazioni Lambda e alle integrazioni di back-end.

> **Important**
>
> Per impostazione predefinita, i client possono richiamare l'API utilizzando l'endpoint `execute-api` generato da API Gateway per l'API. Per garantire che i client possano accedere all'API solo utilizzando un nome di dominio personalizzato con l'autenticazione TLS reciproca, disattivare l'endpoint `execute-api` predefinito. Per ulteriori informazioni, consulta the section called "Disabilitazione dell'endpoint predefinito".

Prerequisiti per l'autenticazione TLS reciproca

Per configurare l'autenticazione TLS reciproca hai bisogno di:

- Un nome di dominio personalizzato
- Almeno un certificato configurato AWS Certificate Manager per il tuo nome di dominio personalizzato
- Un truststore configurato e caricato in Amazon S3

Nomi di dominio personalizzati

Per abilitare l'autenticazione TLS reciproca per un'API HTTP, è necessario configurare un nome di dominio personalizzato per l'API. È possibile abilitare l'autenticazione TLS reciproca per un nome di dominio personalizzato e quindi fornire il nome di dominio personalizzato ai client. Per accedere a un'API utilizzando un nome di dominio personalizzato con l'autenticazione TLS reciproca attivata, i client devono presentare certificati attendibili nelle richieste API. Puoi trovare ulteriori informazioni all'indirizzo the section called “Nomi di dominio personalizzati”.

Utilizzo di certificati AWS Certificate Manager emessi

Puoi richiedere un certificato pubblicamente attendibile direttamente da ACM o importare certificati pubblici o autofirmati. Per configurare un certificato in ACM, accedi ad ACM. Se desideri importare un certificato, continua a leggere la sezione seguente.

Utilizzo di un AWS Autorità di certificazione privata certificato o importato

Per utilizzare un certificato importato in ACM o un certificato AWS Autorità di certificazione privata con TLS reciproco, API Gateway necessita di un certificato `ownershipVerificationCertificate` rilasciato da ACM. Questo certificato di proprietà viene utilizzato solo per verificare che disponi delle autorizzazioni per l'utilizzo del nome di dominio. Non viene utilizzato per l'handshake TLS. Se non ne hai già uno`ownershipVerificationCertificate`, vai a https://console.aws.amazon.com/acm/configurarne uno.

Dovrai mantenere la validità di questo certificato per tutta la durata del nome di dominio. Se un certificato scade e il rinnovo automatico non riesce, tutti gli aggiornamenti al nome di dominio verranno bloccati. Dovrai aggiornare il `ownershipVerificationCertificateArn` con un `ownershipVerificationCertificate` valido prima di poter apportare altre modifiche. Il `ownershipVerificationCertificate` non può essere utilizzato come certificato del server per

un altro dominio TLS reciproco in API Gateway. Se un certificato viene reimportato direttamente in ACM, l'emittente deve rimanere invariato.

Configurazione del truststore

I truststore sono file di testo con un'estensione `.pem`. Sono elenchi attendibili di certificati provenienti da autorità di certificazione. Per utilizzare l'autenticazione TLS reciproca, crea un truststore di certificati X.509 attendibili per accedere all'API.

Devi includere nel truststore l'intera catena di attendibilità, a partire dal certificato della CA emittente, fino al certificato emesso da una CA radice. API Gateway accetta certificati del client emessi da qualsiasi CA presente nella catena di attendibilità. I certificati possono provenire da autorità di certificazione pubbliche o private. I certificati possono avere una lunghezza massima della catena di quattro. È inoltre possibile fornire certificati autofirmati. Nel truststore sono supportati i seguenti algoritmi di hashing:

- SHA-256 o superiore
- RSA-2048 o superiore
- ECDSA-256 o superiore

API Gateway convalida un certo numero di proprietà del certificato. È possibile utilizzare le autorizzazioni Lambda per eseguire ulteriori controlli quando un client richiama un'API, incluso il controllo della revoca di un certificato. API Gateway convalida le seguenti proprietà:

Validation	Descrizione
Sintassi X.509	Il certificato deve soddisfare i requisiti di sintassi X.509.
Integrità	Il contenuto del certificato non deve essere stato modificato rispetto a quello firmato dall'autorità di certificazione del truststore.
Validity	Il periodo di validità del certificato deve essere attuale.
Concatenamento di nomi/concatenamento di chiavi	I nomi e gli oggetti dei certificati devono formare una catena ininterrotta. I certificati

Validation	Descrizione
	possono avere una lunghezza massima della catena di quattro.

Carica il truststore in un bucket Amazon S3 in un singolo file

Example certificates.pem

```
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
...
```

Il seguente AWS CLI comando cp viene caricato nel tuo `certificates.pem` bucket Amazon S3:

```
aws s3 cp certificates.pem s3://bucket-name
```

Configurazione dell'autenticazione TLS reciproca per un nome di dominio personalizzato

Per configurare l'autenticazione TLS reciproca per un'API HTTP, devi utilizzare un nome di dominio personalizzato regionale per l'API, con la versione 1.2 di TLS o successiva. Per ulteriori informazioni sulla creazione e la configurazione di un nome di dominio personalizzato, consulta the section called "Configurazione di un nome di dominio personalizzato regionale".

> **Note**
>
> Il TLS reciproco non è supportato per uso privato. APIs

Dopo aver caricato il truststore su Amazon S3, puoi configurare il tuo nome di dominio personalizzato per utilizzare l'autenticazione TLS reciproca. Quanto segue create-domain-namecrea un nome di dominio personalizzato con TLS reciproco:

```
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
```

Dopo aver creato il nome di dominio, devi configurare i record DNS e le mappature del percorso di base per le operazioni API. Per ulteriori informazioni, consulta Configurazione di un nome di dominio personalizzato regionale in Gateway API.

Richiamo di un'API utilizzando un nome di dominio personalizzato che richiede l'autenticazione TLS reciproca.

Per richiamare un'API con l'autenticazione TLS reciproca abilitata, i client devono presentare un certificato attendibile nella richiesta API. Quando un client tenta di richiamare la tua API, API Gateway cerca l'emittente del certificato del client nel tuo truststore. Per permettere ad API Gateway di procedere con la richiesta, l'emittente del certificato e l'intera catena di attendibilità fino al certificato emesso da una CA root devono trovarsi nel truststore.

Il seguente comando `curl` di esempio seguente invia una richiesta `api.example.com,` che include `my-cert.pem` nella richiesta. `my-key.key` è la chiave privata del certificato.

```
curl -v --key ./my-key.key --cert ./my-cert.pem api.example.com
```

L'API viene richiamata solo se il truststore considera attendibile il certificato. Le seguenti condizioni impediranno ad API Gateway di eseguire l'handshake TLS e lo porteranno a negare la richiesta con il codice di stato `403`. Se il tuo certificato:

- non è attendibile
- è scaduto
- non utilizza un algoritmo supportato

> **Note**
>
> API Gateway non verifica se un certificato è stato revocato.

Aggiornamento del truststore

Per aggiornare i certificati nel truststore, carica un nuovo bundle di certificati in Amazon S3. Potrai quindi aggiornare il nome di dominio personalizzato per utilizzare il certificato aggiornato.

Usa il Controllo delle versioni di Amazon S3 per mantenere più versioni del truststore. Quando si aggiorna il nome di dominio personalizzato per utilizzare una nuova versione del truststore, API Gateway restituisce avvisi se i certificati non sono validi.

API Gateway produce avvisi di certificati solo quando si aggiorna il nome di dominio. API Gateway non invia notifiche se un certificato caricato in precedenza scade.

Il update-domain-namecomando seguente aggiorna un nome di dominio personalizzato per utilizzare una nuova versione di truststore:

```
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'
```

Disabilitazione dell'autenticazione TLS reciproca

Per disabilitare l'autenticazione TLS reciproca per un nome di dominio personalizzato, rimuovere il truststore dal nome di dominio personalizzato, come mostrato nel comando seguente.

Il update-domain-namecomando seguente aggiorna un nome di dominio personalizzato per rimuovere il truststore dal nome di dominio personalizzato:

```
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=''
```

Risoluzione dei problemi relativi all'autenticazione TLS reciproca per l'API HTTP

Di seguito vengono forniti i consigli per la risoluzione dei problemi e la correzione degli errori che possono verificarsi durante l'attivazione dell'autenticazione TLS reciproca.

Risoluzione dei problemi relativi agli avvisi dei certificati

Quando crei un nome di dominio personalizzato con l'autenticazione TLS reciproca, API Gateway genera avvisi se i certificati nel truststore non sono validi. Questo può verificarsi anche quando aggiorni un nome di dominio personalizzato all'utilizzo di un nuovo truststore. Gli avvisi indicano il problema con il certificato e l'oggetto del certificato che ha generato l'avviso. L'autenticazione TLS reciproca è ancora abilitata per l'API, ma alcuni client potrebbero non essere in grado di accedere all'API.

Per identificare il certificato che ha generato l'avviso, devi decodificare i certificati nel truststore. Puoi utilizzare strumenti come `openssl` per decodificare i certificati e identificarne gli oggetti.

Il comando seguente visualizza il contenuto di un certificato, incluso l'oggetto:

```
openssl x509 -in certificate.crt -text -noout
```

Aggiorna o rimuovi i certificati che hanno prodotto gli avvisi, quindi carica un nuovo truststore su Amazon S3. Dopo aver caricato il nuovo truststore, aggiorna il nome di dominio personalizzato per utilizzarlo.

Risoluzione dei problemi relativi ai conflitti dei nomi di dominio

L'errore `"The certificate subject <certSubject> conflicts with an existing certificate from a different issuer."` indica che più autorità di certificazione hanno emesso un certificato per questo dominio. Per ogni oggetto del certificato, può esistere un solo emittente in API Gateway per domini TLS reciproci. Dovrai ottenere tutti i certificati per tale oggetto tramite un unico emittente. Se il problema riguarda un certificato di cui non hai il controllo ma puoi provare la proprietà del nome di dominio, contatta Supporto per aprire un ticket.

Risoluzione dei problemi relativi ai messaggi di stato dei nomi di dominio

`PENDING_CERTIFICATE_REIMPORT`: Ciò significa che hai reimportato un certificato in ACM e la convalida non è riuscita perché il nuovo certificato ha un SAN (nome alternativo del soggetto) che non è coperto dal `ownershipVerificationCertificate` o l'oggetto o SANs nel certificato non copre il nome di dominio. Potrebbe esserci una configurazione errata o è stato importato un certificato

non valido. Devi reimportare un certificato valido in ACM. Per ulteriori informazioni sulla convalida, consulta Convalida della proprietà del dominio.

`PENDING_OWNERSHIP_VERIFICATION`: significa che il certificato verificato in precedenza è scaduto e che ACM non è stato in grado di rinnovarlo in automatico. Dovrai rinnovare il certificato o richiederne uno nuovo. Ulteriori informazioni sul rinnovo del certificato si trovano nella guida alla Risoluzione dei problemi relativi al rinnovo dei certificati gestiti di ACM.

Monitora HTTP APIs nell'API Gateway

Puoi utilizzare CloudWatch metriche e CloudWatch registri per monitorare HTTP. APIs Combinando log e parametri, puoi registrare gli errori e monitorare le prestazioni dell'API.

> **Note**
>
> L'API Gateway potrebbe non generare log e parametri nei seguenti casi:
>
> - Errori 413 per richiesta troppo grande
> - Errori 429 per troppe richieste
> - Errori serie 400 per richieste inviate a un dominio personalizzato che non dispone del mapping API
> - Errori serie 500 per malfunzionamenti interni

Argomenti

- Monitoraggio delle metriche CloudWatch per API HTTP in Gateway API
- Configurare la registrazione per HTTP APIs in API Gateway

Monitoraggio delle metriche CloudWatch per API HTTP in Gateway API

È possibile monitorare l'esecuzione dell'API usando CloudWatch, che raccoglie i dati grezzi dall'API Gateway e li elabora trasformandoli in parametri leggibili quasi in tempo reale. Queste statistiche vengono registrate per un periodo di 15 mesi, per permettere l'accesso alle informazioni storiche e offrire una prospettiva migliore sulle prestazioni del servizio o dell'applicazione Web. Per impostazione predefinita, i dati dei parametri dell'API Gateway vengono inviati automaticamente a CloudWatch in periodi di un minuto. Per monitorare le metriche, creare un dashboard CloudWatch

per l'API. Per ulteriori informazioni su come creare un dashboard CloudWatch, consultare l'argomento relativo alla creazione di un dashboard CloudWatch nella Guida per l'utente di Amazon CloudWatch. Per ulteriori informazioni, consulta Che cos'è Amazon CloudWatch nella Guida per l'utente di Amazon CloudWatch.

I parametri seguenti sono supportati per API HTTP. Inoltre, puoi abilitare parametri dettagliati per scrivere parametri a livello di route in Amazon CloudWatch.

Parametro	Descrizione
4xx	Il numero di errori lato client catturati in un dato periodo.
5xx	Il numero di errori lato server catturati in un dato periodo.
Conteggio	Il numero totale di richieste API in un dato periodo.
IntegrationLatency	Il tempo che intercorre tra il momento in cui API Gateway ritrasmette una richiesta al back-end e quando riceve da questo una risposta.
Latenza	Il tempo che intercorre tra il momento in cui API Gateway riceve una richiesta dal client e quando restituisce ad esso una risposta. La latenza include la latenza di integrazi one e altri sovraccarichi dell'API Gateway.
DataProcessed	La quantità di dati elaborati in byte.

Per filtrare i parametri di API Gateway, puoi utilizzare le dimensioni nella seguente tabella.

Dimensione	Descrizione
ApiId	Filtra i parametri API Gateway per un'API con l'ID API specificato.
ApiId, Fase	Filtra i parametri API Gateway per una fase API con l'ID API e l'ID fase specificati.
ApiId, Metodo, Risorsa, Fase	Filtra le metriche di Gateway API per un metodo API con l'ID API, l'ID fase, il percorso della risorsa e l'ID instradamento specificati. API Gateway non invia questi parametri a meno che non siano stati abilitati esplicitamente parametri CloudWatch dettagliati. A questo scopo, richiama l'operazione UpdateStage dell'API REST API Gateway V2 per aggiornare la proprietà `detailedMetricsEnabled` su `true`. In alternativa, puoi chiamare il comando update-stage della AWS CLI per aggiornare la proprietà `DetailedMetricsEnabled` a `true`. L'abilitazione di tali parametri comporta addebiti aggiuntivi sul tuo account. Per informazioni sui prezzi, consulta la pagina Prezzi di Amazon CloudWatch.

Configurare la registrazione per HTTP APIs in API Gateway

È possibile attivare la registrazione per scrivere i log nei registri. CloudWatch È possibile utilizzare variabili di registrazione per personalizzare il contenuto dei log.

Per migliorare il tuo livello di sicurezza, ti consigliamo di scrivere i log in CloudWatch Logs per tutte le fasi della tua API HTTP. Potrebbe essere necessario eseguire questa operazione per rispettare vari framework di conformità. Per ulteriori informazioni, consultare Amazon API Gateway controls nella Guida per l'utente di AWS Security Hub .

Per attivare la registrazione per un'API HTTP, è necessario effettuare le seguenti operazioni.

1. Assicurarsi che l'utente disponga delle autorizzazioni necessarie per attivare la registrazione.
2. Crea un gruppo di CloudWatch log Logs.
3. Fornisci l'ARN del gruppo di log CloudWatch Logs per una fase della tua API.

Autorizzazioni per attivare la registrazione

Per attivare la registrazione per un'API, l'utente deve disporre delle seguenti autorizzazioni.

Example

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:GetLogEvents",
                "logs:FilterLogEvents"
            ],
            "Resource": "arn:aws:logs:us-east-2:123456789012:log-group:*"
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "logs:CreateLogDelivery",
                "logs:PutResourcePolicy",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:CreateLogGroup",
                "logs:DescribeResourcePolicies",
                "logs:GetLogDelivery",
                "logs:ListLogDeliveries"
            ],
            "Resource": "*"
        }
    ]
}
```

Crea un gruppo di log e attiva la registrazione per HTTP APIs

È possibile creare un gruppo di log e attivare la registrazione degli accessi utilizzando Console di gestione AWS o il. AWS CLI

Console di gestione AWS

1. Creazione di un gruppo di log.

   Per informazioni su come creare un gruppo di log utilizzando la console, consulta Create a Log Group in Amazon CloudWatch Logs User Guide.
2. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
3. Scegliere un'API HTTP.
4. Nella scheda Monitor (Monitoraggio) nel riquadro di navigazione principale, scegliere Logging (Registrazione).
5. Seleziona una fase per attivare la registrazione e scegliere Select (Seleziona).
6. Scegliere Edit (Modifica) per attivare la registrazione degli accessi.
7. Attiva la registrazione degli accessi, inserisci un CloudWatch registro e seleziona un formato di registro.
8. Scegli Save (Salva).

AWS CLI

Il create-log-groupcomando seguente crea un gruppo di log:

```
aws logs create-log-group --log-group-name my-log-group
```

Per attivare la registrazione, è necessario il nome della risorsa Amazon (ARN) del gruppo di log. Il formato ARN è arn:aws:logs: :log-group:. *region account-id log-group-name*

Il comando update-stage seguente attiva la registrazione dei log per la fase `$default` di un'API HTTP:

```
aws apigatewayv2 update-stage --api-id abcdef \
    --stage-name '$default' \
    --access-log-settings '{"DestinationArn": "arn:aws:logs:region:account-
id:log-group:log-group-name", "Format": "$context.identity.sourceIp - -
 [$context.requestTime] \"$context.httpMethod $context.routeKey $context.protocol\"
 $context.status $context.responseLength $context.requestId"}'
```

Formati di log di esempio

Esempi di formati di log di accesso comuni utilizzati con maggiore frequenza sono disponibili nella console API Gateway ed elencati come segue.

- `CLF` (Common Log Format):

```
$context.identity.sourceIp - - [$context.requestTime] "$context.httpMethod
 $context.routeKey $context.protocol" $context.status $context.responseLength
 $context.requestId $context.extendedRequestId
```

- `JSON`:

```
{ "requestId":"$context.requestId", "ip": "$context.identity.sourceIp",
 "requestTime":"$context.requestTime",
 "httpMethod":"$context.httpMethod","routeKey":"$context.routeKey",
 "status":"$context.status","protocol":"$context.protocol",
 "responseLength":"$context.responseLength", "extendedRequestId":
 "$context.extendedRequestId" }
```

- `XML`:

```
<request id="$context.requestId"> <ip>$context.identity.sourceIp</ip> <requestTime>
$context.requestTime</requestTime> <httpMethod>$context.httpMethod</httpMethod>
 <routeKey>$context.routeKey</routeKey> <status>$context.status</status> <protocol>
$context.protocol</protocol> <responseLength>$context.responseLength</responseLength>
 <extendedRequestId>$context.extendedRequestId</extendedRequestId> </request>
```

- CSV (valori separati da virgola):

```
$context.identity.sourceIp,$context.requestTime,$context.httpMethod,
$context.routeKey,$context.protocol,$context.status,$context.responseLength,
$context.requestId,$context.extendedRequestId
```

Personalizzazione dei log degli accessi delle API HTTP

Puoi utilizzare le seguenti variabili per personalizzare i log di accesso per API HTTP. Per ulteriori informazioni sui log di accesso per HTTP APIs, consultaConfigurare la registrazione per HTTP APIs in API Gateway.

Parametro	Description
`$context.accountId`	L'ID dell' AWS account del proprietario dell'API.
`$context.apiId`	Identificatore assegnato da API Gateway all'API.
`$context.authorizer.claims.` *`property`*	Proprietà delle richieste restituite da JSON Web Token (JWT) dopo che il chiamante del metodo è stato autenticato correttamente, ad esempio `$context.authorizer.claims.username` . Per ulteriori informazioni, consulta Controllo degli accessi alle API HTTP con provider di autorizzazioni JWT in Gateway API.

Parametro	Description
	Note La chiamata di `$context.authorizer.claims` restituisce null.
`$context.authorizer.error`	Il messaggio di errore restituito da un'autorizzazione.
`$context.authorizer.`*`property`*	Valore della coppia chiave/valore specificata della mappa `context` restituita da una funzione delle autorizzazioni Lambda di API Gateway. Ad esempio, se le autorizzazioni restituiscono la mappa `context` seguente: `"context" : { "key": "value", "numKey": 1, "boolKey": true }` chiamando `$context.authorizer.key` viene restituita la stringa `"value"`, chiamando `$context.authorizer.numKey` viene restituito 1 e chiamando `$context.authorizer.boolKey` viene restituito `true`.
`$context.awsEndpointRequestId`	L'ID della richiesta dell' AWS endpoint dall'`x-amzn-requestId` intestazione `x-amz-request-id` o.
`$context.awsEndpointRequestId2`	L'ID della richiesta dell' AWS endpoint dall'intestazione. `x-amz-id-2`

Parametro	Description
`$context.customDomain.basePathMatched`	Il percorso per una mappatura API a cui corrisponde una richiesta in ingresso. Applicabile quando un client utilizza un nome di dominio personalizzato per accedere a un'API. Ad esempio, se un client invia una richiesta a `https://api.example.com/v1/orders/1234` e la richiesta corrisponde alla mappatura API con il percorso `v1/orders`, il valore è `v1/orders`. Per ulteriori informazioni, consulta the section called “Mappature API”.
`$context.dataProcessed`	La quantità di dati elaborati in byte.
`$context.domainName`	Nome di dominio completo usato per richiamare l'API. Deve essere lo stesso dell'intestazione `Host` in ingresso.
`$context.domainPrefix`	Prima etichetta di `$context.domainName`.
`$context.error.message`	Una stringa contenente un messaggio di errore API Gateway.
`$context.error.messageString`	Valore `$context.error.message` tra virgolette, ovvero `"$context.error.message"`.
`$context.error.responseType`	Un tipo di `GatewayResponse`. Per ulteriori informazioni, consultare the section called “Metriche” e the section called “Configurazione delle risposte del gateway per la personalizzazione delle risposte agli errori”.
`$context.extendedRequestId`	Equivalente a `$context.requestId`.

Parametro	Description
`$context.httpMethod`	Metodo HTTP usato. I valori validi sono: `DELETE`, `GET`, `HEAD`, `OPTIONS`, `PATCH`, `POST` e `PUT`.
`$context.identity.accountId`	L'ID AWS dell'account associato alla richiesta . Supportato per route che utilizzano l'autoriz zazione IAM.
`$context.identity.caller`	Identificatore dell'entità principale dell'inte rmediario che ha firmato la richiesta. Supportat o per route che utilizzano l'autorizzazione IAM.
`$context.identity.cognitoAu thenticationProvider`	Un elenco separato da virgole con tutti i provider di autenticazione Amazon Cognito utilizzati dal chiamante che effettua la richiesta . Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito. Ad esempio, per un'identità di un pool di utenti Amazon Cognito, `cognito-idp.` *`region`*`.amazonaws.com/` *`user_pool _id`* `,cognito-idp.` *`region`*`.amazonaw s.com/` *`user_pool_id`* `:CognitoS ignIn:` *`token subject claim`* Per informazioni sui provider di autentica zione Amazon Cognito disponibili, consulta Using Federated Identities nella Guida per gli sviluppatori di Amazon Cognito.

Parametro	Description
`$context.identity.cognitoAuthenticationType`	Tipo di autenticazione Amazon Cognito dell'intermediario da cui proviene la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito. I valori possibili includono `authenticated` per le identità autenticate e `unauthenticated` per le identità non autenticate.
`$context.identity.cognitoIdentityId`	ID identità di Amazon Cognito dell'intermediario da cui proviene la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito.
`$context.identity.cognitoIdentityPoolId`	ID pool di identità di Amazon Cognito dell'intermediario da cui proviene la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito.
`$context.identity.principalOrgId`	L'ID organizzazione AWS. Supportato per route che utilizzano l'autorizzazione IAM.
`$context.identity.clientCert.clientCertPem`	Certificato client codificato PEM che il client ha presentato durante l'autenticazione TLS reciproca. Presente quando un client accede a un'API utilizzando un nome di dominio personalizzato che ha attivato l'autenticazione TLS reciproca.
`$context.identity.clientCert.subjectDN`	Nome distinto dell'oggetto del certificato presentato da un client. Presente quando un client accede a un'API utilizzando un nome di dominio personalizzato che ha attivato l'autenticazione TLS reciproca.

Parametro	Description
`$context.identity.clientCert.issuerDN`	Nome distinto dell'approvatore del certificato presentato da un cliente. Presente quando un client accede a un'API utilizzando un nome di dominio personalizzato che ha attivato l'autenticazione TLS reciproca.
`$context.identity.clientCert.serialNumber`	Il numero di serie del certificato. Presente quando un client accede a un'API utilizzando un nome di dominio personalizzato che ha attivato l'autenticazione TLS reciproca.
`$context.identity.clientCert.validity.notBefore`	La data prima della quale il certificato non è valido. Presente quando un client accede a un'API utilizzando un nome di dominio personalizzato che ha attivato l'autenticazione TLS reciproca.
`$context.identity.clientCert.validity.notAfter`	La data dopo la quale il certificato non è valido. Presente quando un client accede a un'API utilizzando un nome di dominio personalizzato che ha attivato l'autenticazione TLS reciproca.
`$context.identity.sourceIp`	L'indirizzo IP di origine della connessione TCP immediata da cui proviene la richiesta all'endpoint di API Gateway.
`$context.identity.user`	Identificatore dell'entità principale dell'utente che sarà autorizzato per l'accesso alle risorse. Supportato per route che utilizzano l'autorizzazione IAM.
`$context.identity.userAgent`	Intestazione `User-Agent` del chiamante API.

Parametro	Description
`$context.identity.userArn`	Amazon Resource Name (ARN) dell'uten te valido identificato dopo l'autenticazione. Supportato per route che utilizzano l'autoriz zazione IAM. Per ulteriori informazioni, consulta https://docs.aws.amazon.com/IAM/latest/Us erGuide/id_users.html.
`$context.integration.error`	Il messaggio di errore restituito da un'integr azione. Equivalente a `$context. integrationErrorMessage` .
`$context.integration.integr ationStatus`	Per l'integrazione del proxy Lambda, il codice di stato restituito dal codice della funzione Lambda di backend AWS Lambda, non dal codice della funzione Lambda.
`$context.integration.latency`	Latenza di integrazione in ms. Equivalente a `$context.integrationLatency` .
`$context.integration.requestId`	L'ID della AWS richiesta dell'endpoint. Equivalente a `$context.awsEndpoi ntRequestId` .
`$context.integration.status`	Il codice di stato restituito da un'integrazione. Per le integrazioni proxy Lambda, questo è il codice di stato restituito dal codice della funzione Lambda.
`$context.integrationErrorMessage`	Una stringa contenente un messaggio di errore di integrazione.
`$context.integrationLatency`	Latenza di integrazione in ms.

Parametro	Description
`$context.integrationStatus`	Per l'integrazione del proxy Lambda, questo parametro rappresenta il codice di stato restituito dalla funzione Lambda di backend AWS Lambda, non dalla funzione Lambda di backend.
`$context.path`	Percorso della richiesta. Ad esempio, `/{stage}/root/child` .
`$context.protocol`	Protocollo della richiesta, ad esempi, `HTTP/1.1`. **Note** API Gateway APIs può accettare richieste HTTP/2, ma API Gateway invia richieste alle integrazioni di backend utilizzando HTTP/1.1. Di conseguenza, il protocollo di richiesta viene registrato come HTTP/1.1 anche se un client invia una richiesta che utilizza HTTP/2.
`$context.requestId`	ID assegnato da API Gateway alla richiesta API.
`$context.requestTime`	Ora della richiesta in formato CLF (`dd/MMM/yyyy:HH:mm:ss +-hhmm`).
`$context.requestTimeEpoch`	Ora della richiesta in formato Unix epoch.
`$context.responseLatency`	Latenza della risposta in ms.
`$context.responseLength`	Lunghezza del payload della risposta in byte.

Parametro	Description
`$context.routeKey`	La chiave di route della richiesta API, ad esempio `/pets`.
`$context.stage`	La fase di distribuzione della richiesta API (ad esempio `beta` o `prod`).
`$context.status`	Stato della risposta del metodo.

Risoluzione dei problemi con le API HTTP in Gateway API

Negli argomenti seguenti vengono forniti suggerimenti per la risoluzione dei problemi relativi a errori e problemi che potrebbero verificarsi durante l'utilizzo di API HTTP.

Argomenti

- Risoluzione dei problemi relativi alle integrazione Lambda di API HTTP
- Risoluzione dei problemi relativi ai provider di autorizzazioni JWT per API HTTP

Risoluzione dei problemi relativi alle integrazione Lambda di API HTTP

Di seguito vengono forniti consigli per la risoluzione di errori e problemi che potrebbero verificarsi durante l'utilizzo AWS Lambda integrazioni con HTTP APIs.

Problema: la mia API con un'integrazione Lambda restituisce **`{"message":"Internal Server Error"}`**

Per risolvere l'errore interno del server, aggiungere la `$context.integrationErrorMessage` variabile di registrazione al formato di registro e visualizzare i registri dell'API HTTP. Per raggiungere questo obiettivo, effettuare le seguenti operazioni:

Per creare un gruppo di log utilizzando Console di gestione AWS

1. Aprire la CloudWatch console all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Scegliere Gruppi di log.
3. Scegliere Crea gruppo di log.

4. Immettere un nome di gruppo di log, quindi scegliere Crea.
5. Prendere nota del nome risorsa Amazon (ARN) per il tuo gruppo di log. Il formato ARN è arn:aws:logs: :log-group:. *region account-id log-group-name* È necessario il gruppo di log ARN per abilitare la registrazione degli accessi per la propria API HTTP.

Per aggiungere la variabile di registrazione **$context.integrationErrorMessage**

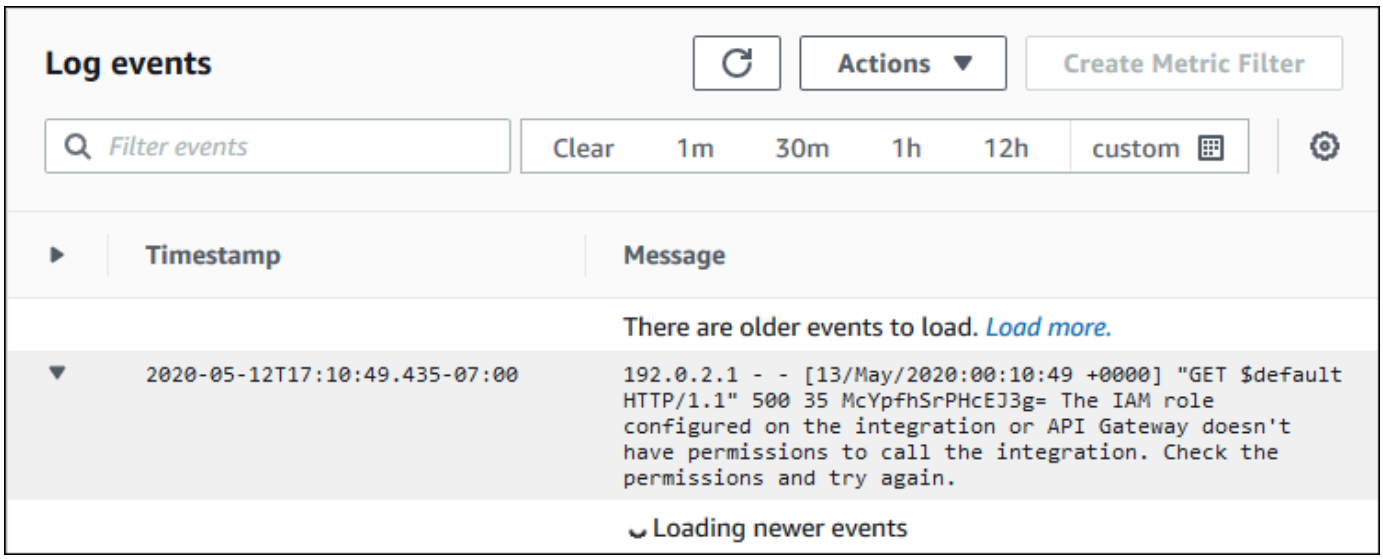
1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere l'API HTTP.
3. In Monitor (Monitora), scegliere Logging (Registrazione).
4. Selezionare una fase della tua API.
5. Scegliere Modifica, quindi abilitare la registrazione degli accessi.
6. Per Log destination (Destinazione dei log), immettere l'ARN per il gruppo di log creato nel passaggio precedente.
7. Per Formato registro, scegliere CLF. Gateway API crea un formato di log di esempio.
8. Aggiungere `$context.integrationErrorMessage` alla fine del formato di registro.
9. Selezionare Salva.

Per visualizzare i log delle API

1. Generare i log Utilizzare un browser o `curl` per richiamare la tua API.

```
$curl https://api-id.execute-api.us-west-2.amazonaws.com/route
```

2. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
3. Scegliere l'API HTTP.
4. In Monitor (Monitora), scegliere Logging (Registrazione).
5. Selezionare la fase dell'API per la quale è stata abilitata la registrazione.
6. Scegli Visualizza i log in. CloudWatch
7. Scegliere il flusso di log più recente per visualizzare i log dell'API HTTP.
8. La voce di registro dovrebbe essere simile alla seguente:



Poiché è stato aggiunto `$context.integrationErrorMessage` al formato di registro, viene visualizzato un messaggio di errore nei nostri registri che riassume il problema.

I registri potrebbero includere un messaggio di errore diverso che indica che si è verificato un problema con il codice della funzione Lambda. In tal caso, controllare il codice della funzione Lambda e verificare che la funzione Lambda restituisca una risposta nel formato richiesto.
Se i log non includono messaggi di errore, aggiungere `$context.error.message` e `$context.error.responseType` al formato del log per ottenere ulteriori informazioni per la risoluzione dei problemi.

In questo caso, i registri mostrano che API Gateway non disponeva delle autorizzazioni necessarie per richiamare la funzione Lambda.

Quando si crea un'integrazione Lambda nella console API Gateway, le autorizzazioni vengono configurate automaticamente per richiamare la funzione Lambda. Quando crei un'integrazione Lambda utilizzando, o un SDK AWS CLI CloudFormation, devi concedere le autorizzazioni ad API Gateway per richiamare la funzione. I comandi add-permission seguenti forniscono l'autorizzazione a diversi instradamenti di API HTTP per invocare una funzione Lambda.

Example Esempio: per la fase **`$default`** e l'instradamento **`$default`** di un'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 Esempio: per la fase **`prod`** e l'instradamento **`test`** di un'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"
```

Conferma la policy della funzione nella scheda Permissions (Autorizzazioni) della console Lambda.

Prova a richiamare nuovamente la tua API. Deve essere visualizzata la risposta della funzione Lambda.

Risoluzione dei problemi relativi ai provider di autorizzazioni JWT per API HTTP

Di seguito vengono forniti consigli per la risoluzione di errori e problemi che potrebbero verificarsi durante l'utilizzo di autorizzatori JSON Web Token (JWT) con HTTP. APIs

Problema: la mia API restituisce **`401 {"message":"Unauthorized"}`**

Controlla l'intestazione `www-authenticate` nella risposta dell'API.

Il comando seguente utilizza `curl` per inviare una richiesta a un'API con un autorizzatore JWT che utilizza `$request.header.Authorization` come origine identità.

```
$curl -v -H "Authorization: token" https://api-id.execute-api.us-
west-2.amazonaws.com/route
```

La risposta dell'API include un'intestazione `www-authenticate`.

```
...
< HTTP/1.1 401 Unauthorized
< Date: Wed, 13 May 2020 04:07:30 GMT
< Content-Length: 26
< Connection: keep-alive
```

```
< www-authenticate: Bearer scope="" error="invalid_token" error_description="the token
 does not have a valid audience"
< apigw-requestid: Mc7UVioPPHcEKPA=
<
* Connection #0 to host api-id.execute-api.us-west-2.amazonaws.com left intact
{"message":"Unauthorized"}}
```

In questo caso, l'intestazione `www-authenticate` mostra che il token non è stato emesso per un pubblico valido. Per consentire ad API Gateway di autorizzare una richiesta, la richiesta `aud` o `client_id` di JWT deve corrispondere a una delle voci destinatario configurate per il provider di autorizzazioni. Gateway API convalida `client_id` solo se `aud` non è presente. Quando `aud` e `client_id` sono entrambi presenti, Gateway API valuta `aud`.

Puoi anche decodificare un JWT e verificare che corrisponda all'emittente, al pubblico e agli ambiti richiesti dall'API. Il sito web jwt.io può eseguire il debug nel browser. JWTs La OpenID Foundation mantiene un elenco di librerie con cui lavorare. JWTs

Per ulteriori informazioni sui provider di autorizzazioni JWT, consulta Controllo degli accessi alle API HTTP con provider di autorizzazioni JWT in Gateway API.

API WebSocket di Gateway API

Un'API WebSocket in API Gateway è una raccolta di route WebSocket integrate con endpoint HTTP di back-end, funzioni Lambda e altri AWS servizi. Puoi utilizzare le caratteristiche API Gateway per semplificare tutti gli aspetti del ciclo di vita dell'API, dalla creazione al monitoraggio delle API di produzione.

Le API WebSocket API Gateway sono bidirezionali. Un client può inviare messaggi a un servizio e i servizi possono inviare messaggi ai client in modo indipendente. Questo comportamento bidirezionale consente interazioni client/servizio più ricche perché i servizi possono inviare i dati ai client senza richiedere ai client di effettuare una richiesta esplicita. Le API WebSocket sono spesso utilizzate in applicazioni in tempo reale come applicazioni di chat, piattaforme di collaborazione, giochi multigiocatore e piattaforme di trading finanziario.

Per un'app di esempio con cui iniziare, consulta Tutorial: crea un'app di WebSocket chat con WebSocket API, Lambda e DynamoDB.

In questa sezione viene descritto come sviluppare, pubblicare, proteggere e monitorare le API WebSocket utilizzando API Gateway.

Argomenti

- Panoramica delle WebSocket API in API Gateway
- Sviluppa WebSocket APIs in API Gateway
- Pubblicazione di API WebSocket che i clienti possono invocare
- Proteggi il tuo WebSocket APIs API Gateway
- Monitoraggio delle API WebSocket in Gateway API

Panoramica delle WebSocket API in API Gateway

In API Gateway puoi creare un' WebSocket API come frontend stateful per un AWS servizio (come Lambda o DynamoDB) o per un endpoint HTTP. L' WebSocket API richiama il backend in base al contenuto dei messaggi che riceve dalle app client.

A differenza di un'API REST, che riceve e risponde alle richieste, un' WebSocket API supporta la comunicazione bidirezionale tra le app client e il backend. Il back-end può inviare messaggi di callback a client connessi.

Nella tua WebSocket API, i messaggi JSON in entrata vengono indirizzati alle integrazioni di backend in base ai percorsi che configuri. (Non-JSON i messaggi vengono indirizzati a un `$default` percorso configurato dall'utente.)

Una route include una chiave di route, che è il valore previsto dopo che un'espressione di selezione della route è stata valutata. `routeSelectionExpression` è un attributo definito a livello di API. Specifica una proprietà JSON che deve essere presente nel payload del messaggio. Per ulteriori informazioni sulle espressioni di selezione della route, consulta the section called “”.

Ad esempio, se i messaggi JSON contengono una proprietà `action` e desideri eseguire operazioni diverse in base a questa proprietà, l'espressione di selezione della route potrebbe essere `${request.body.action}`. La tabella di routing specifica quale operazione eseguire abbinando il valore della proprietà `action` ai valori delle chiavi route personalizzate definiti nella tabella.

Usa i percorsi per un' WebSocket API

Esistono tre route predefinite che possono essere utilizzate: `$connect`, `$disconnect` e `$default`. Inoltre, puoi creare route personalizzate.

- API Gateway chiama il `$connect` percorso quando viene avviata una connessione persistente tra il client e un' WebSocket API.
- API Gateway chiama la route `$disconnect` quando il client o il server si scollega dall'API.
- API Gateway chiama una route personalizzata dopo che l'espressione di selezione della route viene valutata rispetto al messaggio, se viene trovata una route corrispondente; la corrispondenza determina quale integrazione viene richiamata.
- API Gateway chiama la route `$default` se l'espressione di selezione della route non può essere valutata rispetto al messaggio o se non viene trovata una route corrispondente.

Per ulteriori informazioni sulle route `$connect` e `$disconnect`, consulta the section called “Gestione di utenti connessi e app client”.

Per ulteriori informazioni sulla route `$default` e le route personalizzate, consulta the section called “Invocazione dell'integrazione backend”.

Invio dei dati alle app client connesse

I servizi di back-end possono inviare dati alle app client connesse. È possibile inviare i dati nel modo seguente:

- Utilizza un'integrazione per inviare una risposta, che viene restituita al client da una risposta di instradamento definita.
- Puoi utilizzare l'API `@connections` per inviare una richiesta POST. Per ulteriori informazioni, consulta the section called “Utilizzo di comandi `@connections` nel servizio di back-end”.

WebSocket Codici di stato delle API

Le API WebSocket API Gateway utilizzano i seguenti codici di stato per la comunicazione dal server al client, come descritto nel WebSocket Close Code Number Registry:

1001

Gateway API restituisce questo codice di stato quando il client è inattivo per 10 minuti o raggiunge la durata massima della connessione di 2 ore.

1003

Gateway API restituisce questo codice di stato quando un endpoint riceve un tipo di supporto binario. I tipi di supporti binari non sono supportati per le WebSocket API.

1005

Gateway API restituisce questo codice di stato se il client invia un frame chiuso senza codice di chiusura.

1006

API Gateway restituisce questo codice di stato in caso di chiusura imprevista della connessione, ad esempio la connessione TCP chiusa senza un frame WebSocket chiuso.

1008

Gateway API restituisce questo codice di stato quando un endpoint riceve troppe richieste da un determinato client.

1009

Gateway API restituisce questo codice di stato quando un endpoint riceve un messaggio troppo grande per essere elaborato.

1011

Gateway API restituisce questo codice di stato in caso di errore interno del server.

1012

Gateway API restituisce questo codice di stato se il servizio si riavvia.

Gestisci gli utenti connessi e le app client: percorsi $connect e $disconnect

La sezione seguente descrive come utilizzare i `$disconnect` percorsi `$connect` e per l' WebSocket API.

Argomenti

- La rotta $connect
- Trasmissione delle informazioni di connessione dalla rotta $connect
- La rotta $disconnect

La rotta $connect

Le app client si connettono alla tua WebSocket API inviando una richiesta di WebSocket aggiornamento. Se la richiesta ha esito positivo, la route `$connect` viene eseguita mentre la connessione viene stabilita.

Poiché la WebSocket connessione è una connessione con stato, puoi configurare l'autorizzazione solo sulla `$connect` route. `AuthN/AuthZ`verrà eseguito solo al momento della connessione.

Finché l'esecuzione dell'integrazione associata alla route `$connect` non viene completata, la richiesta di aggiornamento è in sospeso e la connessione effettiva non verrà stabilita. Se la richiesta `$connect` ha esito negativo (ad esempio, a causa di un errore `AuthN/AuthZ` o errore di integrazione), la connessione non verrà stabilita.

> **Note**
>
> Se l'autorizzazione ha esito negativo su `$connect`, la connessione non verrà stabilita e il client riceverà una risposta `401` o `403`.

La configurazione di un'integrazione per `$connect` è facoltativa. È opportuno valutare la configurazione di un'integrazione `$connect` se:

- Si desidera consentire ai client di specificare sottoprotocolli utilizzando il campo `Sec-WebSocket-Protocol`. Per il codice di esempio, consulta Imposta una `$connect` route che richiede un WebSocket sottoprotocollo.
- Desideri ricevere una notifica in caso di connessione dei client.
- Desideri limitare le connessioni o controllare chi si connette.
- Desideri che il back-end invii messaggi ai client utilizzando un URL di callback.
- Desideri archiviare gli ID di connessione, i nomi di dominio o altri dettagli di connessione in un database (ad esempio, Amazon DynamoDB). L'ID di connessione è una stringa di lunghezza variabile la cui lunghezza e formato potrebbero cambiare.

Trasmissione delle informazioni di connessione dalla rotta $connect

È possibile utilizzare integrazioni proxy e non proxy per passare informazioni dall'instradamento `$connect` a un database o a un altro Servizio AWS.

Per passare le informazioni di connessione utilizzando un'integrazione proxy

È possibile accedere alle informazioni di connessione da un'integrazione proxy Lambda nell'evento. Usa un'altra AWS Lambda funzione Servizio AWS or per postare sulla connessione.

La seguente funzione Lambda mostra come utilizzare l'oggetto `requestContext` per registrare l'ID di connessione, il nome di dominio, il nome della fase e le stringhe di query.

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}
```

Per passare informazioni di connessione utilizzando un'integrazione proxy

- È possibile accedere alle informazioni di connessione da un'integrazione non proxy. Configura la richiesta di integrazione e fornisci un modello di richiesta WebSocket API. Il seguente modello di mappatura Velocity Template Language (VTL) fornisce una richiesta di integrazione. Questa richiesta invia i seguenti dettagli a un'integrazione non proxy:

  - ID connessione
  - Domain name (Nome dominio)
  - Nome fase
  - Path
  - Intestazioni
  - Stringhe di query

  Questa richiesta invia l'ID di connessione, il nome di dominio, il nome della fase i percorsi, le intestazioni e le stringhe di query a un'integrazione non proxy.

```
{
    "connectionId": "$context.connectionId",
    "domain": "$context.domainName",
```

```
    "stage": "$context.stage",
    "params": "$input.params()"
}
```

Per ulteriori informazioni sulla configurazione delle trasformazioni di dati, consulta the section called “Trasformazioni dei dati”.

Per completare la richiesta di integrazione, impostare `StatusCode: 200` per la risposta di integrazione. Per ulteriori informazioni sulla configurazione di una risposta di integrazione, consulta Configurazione di una risposta di integrazione mediante la console API Gateway.

La rotta **`$disconnect`**

La route `$disconnect` viene eseguita dopo la chiusura della connessione.

La connessione può essere chiusa dal server o dal client. Poiché la connessione è già chiusa quando viene eseguita, `$disconnect` è un evento best-effort. API Gateway farà il massimo per consegnare l'evento `$disconnect` all'integrazione, ma non può garantire la consegna.

Il back-end è in grado di avviare la disconnessione utilizzando l'API `@connections`. Per ulteriori informazioni, consulta the section called “Utilizzo di comandi `@connections` nel servizio di back-end”.

Invoca la tua integrazione di backend con **`$default`** Route e i percorsi personalizzati in API Gateway

La sezione seguente descrive come richiamare l'integrazione del backend utilizzando la `$default` route o una route personalizzata per un'API. WebSocket

Argomenti

- Utilizzo di instradamenti per elaborare messaggi
- La rotta $default
- Instradamenti personalizzati
- Utilizzo delle integrazioni WebSocket API Gateway per connettersi alla logica aziendale
- Differenze importanti tra WebSocket API e API REST

Utilizzo di instradamenti per elaborare messaggi

Nelle API WebSocket API Gateway, i messaggi possono essere inviati dal client al servizio di backend e viceversa. A differenza del request/response modello HTTP, WebSocket nel backend è possibile inviare messaggi al client senza che il client intraprenda alcuna azione.

I messaggi possono essere JSON o non JSON. Tuttavia, solo i messaggi JSON possono essere indirizzati a integrazioni specifiche in base al contenuto del messaggio. Non-JSON i messaggi vengono passati al backend tramite il percorso. `$default`

> **Note**
>
> API Gateway supporta payload dei messaggi fino a 128 KB con dimensione del frame massima di 32 KB. Se un messaggio supera i 32 KB, occorre suddividerlo in più frame, ciascuno di 32 KB o più piccolo. Se si riceve un messaggio (o frame) di dimensioni maggiori, la connessione viene chiusa con codice 1009.
> Al momento i payload binari non sono supportati. Se si riceve un frame binario, la connessione viene chiusa con codice 1003. Tuttavia, è possibile convertire i payload binari in testo. Per informazioni, consulta the section called “Tipi di supporti binari”.

Con le WebSocket API in API Gateway, i messaggi JSON possono essere instradati per eseguire un servizio di backend specifico basato sul contenuto dei messaggi. Quando un client invia un messaggio tramite la sua WebSocket connessione, ciò si traduce in una richiesta di routing all'API. WebSocket La richiesta verrà associata alla route con la chiave route corrispondente in API Gateway. Puoi configurare una richiesta di routing per un' WebSocket API nella console API Gateway AWS CLI, utilizzando o utilizzando un AWS SDK.

> **Note**
>
> Negli AWS SDK AWS CLI e, puoi creare percorsi prima o dopo aver creato le integrazioni. Attualmente la console non supporta il riutilizzo di integrazioni, perciò è necessario creare la route prima e quindi creare l'integrazione per tale route.

Puoi configurare API Gateway per eseguire la convalida su una route prima di passare alla richiesta di integrazione. Se la convalida fallisce, API Gateway fallisce la richiesta senza chiamare il backend,

invia al client una risposta del "Bad request body" gateway simile alla seguente e pubblica i risultati della convalida in Logs: CloudWatch

```
{"message" : "Bad request body", "connectionId": "{connectionId}", "messageId":
 "{messageId}"}
```

Ciò consente di ridurre le chiamate non necessarie al back-end e concentrarsi sugli altri requisiti dell'API.

Puoi anche definire una risposta di instradamento per le route dell'API per abilitare la comunicazione bidirezionale. Una risposta di instradamento descrive quali dati verranno inviati al client al termine dell'integrazione di una particolare route. Non è necessario definire una risposta per una route se, ad esempio, desideri che un client invii messaggi al back-end senza ricevere una risposta (comunicazione unidirezionale). Tuttavia, se non fornisci una risposta di instradamento, API Gateway non invierà informazioni sul risultato dell'integrazione ai client.

La rotta $default

Ogni API WebSocket API Gateway può avere un `$default` percorso. Si tratta di un valore di instradamento speciale che può essere utilizzato nei seguenti modi:

- Puoi usarlo insieme a chiavi di route definite per specificare una route di "fallback", ad esempio un'integrazione fittizia generica che restituisce un determinato messaggio di errore, per i messaggi in ingresso che non corrispondono ad alcuna delle chiavi di route definite.
- Puoi utilizzarlo senza alcuna chiave route definita, per specificare un modello di proxy che delega l'instradamento a un componente di back-end.
- Puoi utilizzarlo per specificare una route per payload non JSON.

Instradamenti personalizzati

Se desideri invocare un'integrazione specifica in base al contenuto del messaggio, puoi farlo creando una route personalizzata.

Una route personalizzata utilizza una chiave route e l'integrazione specificati dall'utente. Quando un messaggio in entrata contiene una proprietà JSON e tale proprietà valuta un valore che corrisponde al valore della chiave route, API Gateway richiama l'integrazione. (Per ulteriori informazioni, consulta the section called “Panoramica delle API WebSocket ”.)

Ad esempio, supponiamo che tu voglia creare un'applicazione chat room. Potresti iniziare creando un' WebSocket API la cui espressione di selezione del percorso è`$request.body.action`. Potresti quindi definire due route: `joinroom` e `sendmessage`. Un'app client potrebbe richiamare la route `joinroom` inviando un messaggio del tipo seguente:

```
{"action":"joinroom","roomname":"developers"}
```

E potrebbe richiamare la route `sendmessage` inviando un messaggio del tipo seguente:

```
{"action":"sendmessage","message":"Hello everyone"}
```

Utilizzo delle integrazioni WebSocket API Gateway per connettersi alla logica aziendale

Dopo aver impostato un percorso per un'API WebSocket API Gateway, devi specificare l'integrazione che desideri utilizzare. Come nel caso di una route, per la quale è possibile avere una richiesta di instradamento e una risposta di instradamento, un'integrazione può avere una richiesta di integrazione e una risposta di integrazione. Una richiesta di integrazione contiene le informazioni previste dal back-end per elaborare la richiesta proveniente dal client. Una risposta di integrazione contiene i dati restituiti dal back-end ad API Gateway e che possono essere utilizzati per costruire un messaggio da inviare al cliente (se è definita una risposta di instradamento).

Per ulteriori informazioni sulla configurazione di integrazioni, consulta the section called “Integrazioni”.

Differenze importanti tra WebSocket API e API REST

Le integrazioni per le WebSocket API sono simili alle integrazioni per le API REST, ad eccezione delle seguenti differenze:

- Al momento, nella console API Gateway è necessario creare prima una route e quindi un'integrazione come destinazione di tale route. Tuttavia, nell'API e nella CLI, puoi creare route e integrazioni in modo indipendente, in qualsiasi ordine.
- Puoi utilizzare una singola integrazione per più route. Ad esempio, se disponi di un set di operazioni tra loro strettamente correlate, puoi fare in modo che tutte queste route vadano in una singola funzione Lambda. Anziché definire più volte i dettagli dell'integrazione, puoi specificarla una sola volta e assegnarla a ciascuna delle route correlate.

> **Note**
>
> Attualmente la console non supporta il riutilizzo di integrazioni, perciò è necessario creare la route prima e quindi creare l'integrazione per tale route.
> Negli AWS CLI and AWS SDK, puoi riutilizzare un'integrazione impostando la destinazione del percorso su un valore di"`integrations/{integration-id}`", dove "`{integration-id}`" è l'ID univoco dell'integrazione da associare alla route.

- API Gateway fornisce più espressioni di selezione che puoi utilizzare nelle route e nelle integrazioni. Non è necessario basarsi sul tipo di contenuto per selezionare un modello di input o una mappatura di output. Analogamente alle espressioni di selezione della route, puoi definire un'espressione di selezione che deve essere valutata da API Gateway per scegliere l'elemento corretto. Tutte verranno ripristinati al modello `$default` se non viene trovato un modello corrispondente.
  - Nelle richieste di integrazione, l'espressione di selezione del modello supporta `$request.body.<json_path_expression>` e valori statici.
  - Nelle risposte di integrazione, l'espressione di selezione del modello supporta `$request.body.<json_path_expression>`, `$integration.response.statuscode`, `$integration.response.header.<headerName>` e valori statici.

Nel protocollo HTTP, in cui richieste e risposte vengono inviate in maniera sincrona, la comunicazione è essenzialmente unidirezionale. Nel WebSocket protocollo, la comunicazione è bidirezionale. Le risposte sono asincrone e non vengono necessariamente ricevute dal client nello stesso ordine di invio dei messaggi del client. Inoltre, il back-end può inviare messaggi al client.

> **Note**
>
> Per una route configurata per utilizzare l'integrazione `AWS_PROXY` o `LAMBDA_PROXY`, la comunicazione è unidirezionale e API Gateway non trasferisce automaticamente la risposta del back-end alla risposta di instradamento. Ad esempio, in caso dell'integrazione `LAMBDA_PROXY`, il corpo restituito dalla funzione Lambda non verrà restituito al client.
> Se desideri che il client riceva risposte di integrazione, devi definire una risposta di instradamento per rendere possibile la comunicazione bidirezionale.

WebSocket espressioni di selezione

API Gateway usa espressioni di selezione per valutare il contesto delle richieste e delle risposte e produrre una chiave. La chiave viene quindi utilizzata per selezionare da un set di valori possibili, in genere fornito da te, lo sviluppatore dell'API. L'esatto set di variabili supportate varierà a seconda del tipo di espressione. Ogni espressione viene descritta in modo dettagliato di seguito.

Per tutte le espressioni, il linguaggio segue lo stesso set di regole:

- Una variabile ha il prefisso `"$"`.
- Le parentesi graffe possono essere utilizzate per definire i limiti delle variabili in modo esplicito, ad esemp., `"${request.body.version}-beta"`.
- Sono supportate numerose variabili, ma la valutazione viene eseguita una sola volta (non esiste la valutazione ricorsiva).
- Il segno di dollaro ($) può essere preceduto dal carattere di escape `"\"`. Questa prassi è molto utile quando si definisce un'espressione associata alla chiave riservata `$default`, ad esempio `"\$default"`.
- In alcuni casi, un formato di modello è obbligatorio. In questo caso, l'espressione deve essere racchiusa tra barre (`"/"`), ad esempio `"/2\d\d/"` per corrispondere ai codici di stato 2*XX*.

Argomenti

- Espressioni di selezione della risposta di instradamento
- Espressioni di selezione della chiave API
- Espressioni di selezione della mappatura dell'API
- WebSocket riepilogo delle espressioni di selezione

Espressioni di selezione della risposta di instradamento

Una risposta di instradamento viene utilizzata per modellare una risposta dal back-end al client. Infatti WebSocket APIs, una risposta di percorso è facoltativa. Una volta definito, segnala ad API Gateway che deve restituire una risposta a un client dopo aver ricevuto un WebSocket messaggio.

La valutazione dell'espressione di selezione della risposta di instradamento produce una chiave di risposta di instradamento. Infine, la chiave viene utilizzata per scegliere tra una delle `RouteResponses` associate all'API. Tuttavia, al momento è supportata solo la chiave `$default`.

Espressioni di selezione della chiave API

Questa espressione viene valutata quando il servizio stabilisce che la richiesta specifica deve procedere solo se il client fornisce una chiave API valida.

Attualmente, i soli due valori supportati sono `$request.header.x-api-key` e `$context.authorizer.usageIdentifierKey`.

Espressioni di selezione della mappatura dell'API

Questa espressione viene valutata per determinare quale fase API viene selezionata quando viene effettuata una richiesta utilizzando un dominio personalizzato.

Attualmente, l'unico valore supportato è `$request.basepath`.

WebSocket riepilogo delle espressioni di selezione

La tabella seguente riassume i casi d'uso delle espressioni di selezione in WebSocket APIs:

Espressione di selezione	Restituisce la chiave per	Note	Esempio di caso d'uso
`Api.RouteSelectionExpression`	`Route.RouteKey`	`$defaul` chiave è supporta a come instradar ento catch-all.	WebSocket Indirizza i messaggi in base al contesto della richiesta di un client.
`Route.ModelSelecti`	Chiave per `Route.RequestModels`	Facoltati vo.	Convalida dinamica della

Espressione di selezione	Restituisce la chiave per	Note	Esempio di caso d'uso
`onExpression`		Se fornita per un'integrazione non proxy, viene effettuata la convalida del modello. `$default` chiave è supportata come catch-all.	richiesta all'interno dello stesso instradamento.

Espressione di selezione	Restituisce la chiave per	Note	Esempio di caso d'uso
`Integration.TemplateSelectionExpression`	Chiave per `Integration.RequestTemplates`	Facoltativo. Può essere fornita per un'integrazione non proxy allo scopo di manipolare i payload in entrata. `${request.body.jsonPath}` supportati e valori statici. `$default` chiave è supportata	Manipolazione della richiesta dell'intermediario in base alle proprietà dinamiche della richiesta.

Espressione di selezione	Restituisce la chiave per	Note	Esempio di caso d'uso
		come catch-all.	

Espressione di selezione	Restituisce la chiave per	Note	Esempio di caso d'uso
`Integration.IntegrationResponseSelectionExpression`	`IntegrationResponse.IntegrationResponseKey`	Facoltativo. Può essere fornita per un'integrazione non proxy. Agisce da corrispondenza di modello per i messagg di errore (da Lambda) o i codici di stato (da integrazioni HTTP).	Manipolazione della risposta del back-end. Scegliere l'azione da eseguire in base alla risposta dinamica del back-end, ad esempio la gestione distinta di determinati errori.

Espressione di selezione	Restituisce la chiave per	Note	Esempio di caso d'uso
		`$defau` necessa a la chiave per far sì che le integrazi oni non proxy agiscano da catch-all per le risposte con esito positivo.	

Espressione di selezione	Restituisce la chiave per	Note	Esempio di caso d'uso
`IntegrationResponse.TemplateSelectionExpression`	Chiave per `IntegrationResponse.ResponseTemplates`	Facoltativo. Può essere fornita per un'integrazione non proxy. La chiave $default è supportata.	In alcuni casi, una proprietà dinamica della risposta potrebbe imporre trasformazioni differenti all'interno dello stesso instradamento e dell'integrazione associata. `${request.body.jsonPath}` , `${integration.response.statuscode}` , `${integra`

Espressione di selezione	Restituisce la chiave per	Note	Esempio di caso d'uso
			`tion.response.header.headerName}` , `${integration.response.multivalueheader.headerName}` , Sono supportati , , , e valori statici. `$defaultLa` chiave è supportata come catch-all.

Espressione di selezione	Restituisce la chiave per	Note	Esempio di caso d'uso
`Route.RouteResponseSelectionExpression`	`RouteResponse.RouteResponseKey`	Dovrebbe essere forniti per avviare una comunicazione bidirezionale per un WebSocket percorso Attualmente, questo valore è limitato solo a `$default`	
`RouteResponse.ModelSelectionExpression`	Chiave per `RouteResponse.RequestModels`	Attualmente non è supportata.	

Sviluppa WebSocket APIs in API Gateway

Questa sezione fornisce dettagli sulle funzionalità di API Gateway di cui hai bisogno durante lo sviluppo del tuo API Gateway APIs.

Durante lo sviluppo delle API di API Gateway è possibile impostare una serie di caratteristiche dell'API. Queste caratteristiche dipendono dal caso d'uso dell'API. Ad esempio, è possibile consentire a solo a determinati client di richiamare l'API oppure che questa sia disponibile per tutti. È possibile decidere che una chiamata API esegua una funzione Lambda, una query a un database o richiami un'applicazione.

Argomenti

- Crea WebSocket APIs in API Gateway
- Tipi di indirizzo IP per API WebSocket in Gateway API
- Crea percorsi per WebSocket APIs API Gateway
- Controlla e gestisci l'accesso WebSocket APIs a API Gateway
- Integrazioni per API WebSocket APIs Gateway
- Richiedi la convalida per WebSocket APIs in API Gateway
- Trasformazioni dei dati per API WebSocket APIs Gateway
- Tipi di supporti binari per API WebSocket in Gateway API
- Invoca WebSocket APIs

Crea WebSocket APIs in API Gateway

Puoi creare un' WebSocket API nella console API Gateway, utilizzando il comando AWS CLI create-api o utilizzando il `CreateApi` comando in un AWS SDK. Le seguenti procedure mostrano come creare una nuova API. WebSocket

> **Note**
>
> WebSocket APIs supporta solo TLS 1.2 e TLS 1.3. Le versioni precedenti di TLS non sono supportate.

Crea un' WebSocket API utilizzando i comandi AWS CLI

Il comando create-api seguente crea un'API con l'espressione di selezione dell'instradamento `$request.body.action`:

```
aws apigatewayv2 --region us-east-1 create-api --name "myWebSocketApi3" --protocol-type
 WEBSOCKET --route-selection-expression '$request.body.action'
```

L'output sarà simile al seguente:

```
{
    "ApiKeySelectionExpression": "$request.header.x-api-key",
    "Name": "myWebSocketApi3",
    "CreatedDate": "2018-11-15T06:23:51Z",
    "ProtocolType": "WEBSOCKET",
    "RouteSelectionExpression": "'$request.body.action'",
    "ApiId": "aabbccddee"
}
```

Creare un' WebSocket API utilizzando la console API Gateway

Puoi creare un' WebSocket API nella console scegliendo il WebSocket protocollo e assegnando un nome all'API.

> **Important**
>
> Dopo aver creato l'API, non puoi modificare il protocollo scelto per la stessa. Non è possibile convertire un' WebSocket API in un'API REST o viceversa.

Per creare un' WebSocket API utilizzando la console API Gateway

1. Accedere alla console API Gateway e scegliere Create API (Crea API).
2. In WebSocket API, scegli Build. Sono supportati solo gli endpoint regionali.
3. In Nome API immetti il nome dell'API.
4. Per Espressione di selezione dell'instradamento immetti un valore. Ad esempio, `$request.body.action`.

   Per ulteriori informazioni sulle espressioni di selezione della route, consulta the section called "".

5. Esegui una delle seguenti operazioni:
   - Scegli Crea API vuota per creare un'API senza instradamenti.
   - Scegli Successivo per collegare gli instradamenti all'API.

   Puoi collegare gli instradamenti dopo aver creato l'API.

Tipi di indirizzo IP per API WebSocket in Gateway API

Quando si crea un'API, è possibile specificare il tipo di indirizzo IP che può invocarla. È possibile scegliere IPv4 per risolvere gli indirizzi IPv4 per invocare l'API oppure dualstack per consentire a entrambi gli indirizzi IPv4 e IPv6 di invocare l'API. È consigliabile impostare il tipo di indirizzo IP su dualstack per ridurre l'esaurimento dello spazio IP o per il livello di sicurezza. Per ulteriori informazioni sui vantaggi del tipo di indirizzo IP dualstack, consultare IPv6 on AWS.

Considerazioni sui tipi di indirizzo IP

Le seguenti considerazioni potrebbero influire sull'utilizzo dei tipi di indirizzo IP:

- Il tipo di indirizzo IP predefinito per tutte le API WebSocket è IPv4.
- Se per un'API esistente si modifica il tipo di indirizzo IP da IPv4 a dualstack, è necessario verificare che tutte le policy che controllano l'accesso alle API siano aggiornate in modo da tenere conto delle chiamate IPv6. Quando si modifica il tipo di indirizzo IP, la modifica diventa immediatamente effettiva.
- L'API può essere mappata su un nome di dominio personalizzato con un tipo di indirizzo IP diverso da quello dell'API. Se si disabilita l'endpoint API predefinito, si potrebbe influire sul modo in cui i chiamanti possono invocare l'API.

Modifica del tipo di indirizzo IP di un'API WebSocket

È possibile modificare il tipo di indirizzo IP aggiornando la configurazione dell'API. È possibile aggiornare la configurazione dell'API utilizzando la Console di gestione AWS, AWS CLI, CloudFormation o un AWS SDK. Se si modifica il tipo di indirizzo IP dell'API, non è necessario eseguire di nuovo l'implementazione dell'API per rendere effettive le modifiche.

Console di gestione AWS

Per modificare il tipo di indirizzo IP di un'API WebSocket

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli un'API WebSocket.
3. Scegliere Impostazioni API, quindi scegliere Modifica.
4. Per Tipo di indirizzo IP, selezionare IPv4 o Dualstack.
5. Selezionare Salva.

   La modifica alla configurazione dell'API ha effetto immediato.

AWS CLI

Il comando update-api seguente aggiorna un'API in modo che abbia un tipo di indirizzo IP dualstack:

```
aws apigatewayv2 update-api \
    --api-id abcd1234 \
    --ip-address-type dualstack
```

L'output sarà simile al seguente:

```
{
    "ApiEndpoint": "https://abcd1234.execute-api.us-east-1.amazonaws.com",
    "ApiId": "abcd1234",
    "ApiKeySelectionExpression": "$request.header.x-api-key",
    "CreatedDate": "2025-02-04T22:20:20+00:00",
    "DisableExecuteApiEndpoint": false,
    "Name": "My-WebSocket-API",
    "ProtocolType": "WEBSOCKET",
    "RouteSelectionExpression": "$request.method $request.path",
    "Tags": {},
    "NotificationUris": [],
    "IpAddressType": "dualstack"
}
```

Crea percorsi per WebSocket APIs API Gateway

Nella tua WebSocket API, i messaggi JSON in arrivo vengono indirizzati alle integrazioni di backend in base ai percorsi che configuri. (Messaggi non JSON sono indirizzati a una route `$default` configurata dall'utente).

Una route include una chiave di route, che è il valore previsto dopo che un'espressione di selezione della route è stata valutata. `routeSelectionExpression` è un attributo definito a livello di API. Specifica una proprietà JSON che deve essere presente nel payload del messaggio. Per ulteriori informazioni sulle espressioni di selezione della route, consulta the section called “”.

Ad esempio, se i messaggi JSON contengono una proprietà `action` e desideri eseguire operazioni diverse in base a questa proprietà, l'espressione di selezione della route potrebbe essere `${request.body.action}`. La tabella di routing specifica quale operazione eseguire abbinando il valore della proprietà `action` ai valori delle chiavi route personalizzate definiti nella tabella.

Esistono tre route predefinite che possono essere utilizzate: `$connect`, `$disconnect` e `$default`. Inoltre, puoi creare route personalizzate.

- API Gateway chiama il `$connect` percorso quando viene avviata una connessione persistente tra il client e un' WebSocket API.
- API Gateway chiama la route `$disconnect` quando il client o il server si scollega dall'API.
- API Gateway chiama una route personalizzata dopo che l'espressione di selezione della route viene valutata rispetto al messaggio, se viene trovata una route corrispondente; la corrispondenza determina quale integrazione viene richiamata.
- API Gateway chiama la route `$default` se l'espressione di selezione della route non può essere valutata rispetto al messaggio o se non viene trovata una route corrispondente.

Espressioni di selezione dell'instradamento

Un'espressione di selezione dell'instradamento viene valutata quando il servizio seleziona l'instradamento da seguire per un messaggio in entrata. Il servizio utilizza la route il cui `routeKey` corrisponde esattamente al valore valutato. Se non trova alcuna corrispondenza ed è disponibile una route con una chiave di routing `$default`, verrà selezionata. Se nessuna route corrisponde al valore valutato e non è disponibile una route `$default`, il servizio restituisce un errore. Per WebSocket based APIs, l'espressione deve avere il seguente formato`$request.body.`*`{path_to_body_element}`*.

Supponiamo, ad esempio, che tu stia inviando il seguente messaggio JSON:

```
{
    "service" : "chat",
    "action" : "join",
    "data" : {
        "room" : "room1234"
   }
}
```

Potresti decidere di selezionare il comportamento dell'API in base alla proprietà `action`. In questo caso, potresti definire la seguente espressione di selezione dell'instradamento:

```
$request.body.action
```

In questo esempio, `request.body` si riferisce al payload JSON del messaggio ed `.action` è un'JSONPathespressione. Puoi utilizzare qualsiasi espressione di percorso JSON dopo `request.body`, ma tieni presente che il risultato sarà in formato stringa. Ad esempio, se l' JSONPath espressione restituisce una matrice di due elementi, questo verrà presentato come stringa. `"[item1, item2]"` Per questo motivo, è opportuno che l'espressione restituisca un valore e non una matrice o un oggetto.

Puoi semplicemente utilizzare un valore statico oppure utilizzare più variabili. La tabella seguente mostra esempi con i relativi risultati valutati per il payload precedente.

Espressione	Risultato valutato	Descrizione
`$request.body.action`	`join`	Una variabile da cui è stato rimosso il wrapping
`${request.body.action}`	`join`	Una variabile

Espressione	Risultato valutato	Descrizione
		con wrapping
`${request.body.service}/${request.body.action}`	`chat/join`	Più variabili con valori statici
`${request.body.action}-${request.body.invalidPath}`	`join-`	Se non JSONPath viene trovato, la variabile viene risolta come «».
`action`	`action`	Valore statico
`\$default`	`$default`	Valore statico

Il risultato valutato viene utilizzato per trovare una route. Se esiste una route con una chiave di routing corrispondente, viene selezionato per elaborare il messaggio. Se non è possibile trovare una route corrispondente, API Gateway cerca di trovare la route `$default`, se disponibile. Se la route `$default` non è definita, API Gateway restituisce un errore.

Configurare i percorsi per un' WebSocket API in API Gateway

Quando crei per la prima volta una nuova WebSocket API, ci sono tre percorsi predefiniti: `$connect``$disconnect`, e`$default`. Puoi crearli utilizzando la console, l'API o AWS CLI. Se lo

desideri, puoi creare instradamenti personalizzati. Per ulteriori informazioni, consulta the section called “Panoramica delle API WebSocket ”.

> **Note**
>
> Nell'interfaccia a riga di comando, puoi creare instradamenti prima o dopo aver creato le integrazioni e puoi riutilizzare la stessa integrazione per più instradamenti.

Creazione di una route tramite la console API Gateway

Per creare una route utilizzando la console API Gateway

1. Accedere alla console API Gateway, scegliere l'API e selezionare Routes (Route).
2. Seleziona Crea instradamento.
3. Per Chiave instradamento inserisci il nome della chiave instradamento. È possibile creare gli instradamenti predefiniti (`$connect`, `$disconnect` e `$default`) o un instradamento personalizzato.

   > **Note**
   >
   > Quando crei un instradamento personalizzato, non utilizzare il prefisso $ nel nome della chiave di instradamento. Questo prefisso è riservato per gli instradamenti predefiniti.

4. Seleziona e configura il tipo di integrazione per l'instradamento. Per ulteriori informazioni, consulta the section called “Configurare una richiesta di integrazione WebSocket API utilizzando la console API Gateway”.

Crea un percorso utilizzando il AWS CLI

Il comando create-route seguente crea un instradamento:

```
aws apigatewayv2 --region us-east-1 create-route --api-id aabbccddee --route-key
 $default
```

L’output sarà simile al seguente:

```
{
```

```
    "ApiKeyRequired": false,
    "AuthorizationType": "NONE",
    "RouteKey": "$default",
    "RouteId": "1122334"
}
```

Specifica delle impostazioni per la richiesta di instradamento per **`$connect`**

Quando configuri l'instradamento `$connect` per la tua API, sono disponibili le seguenti impostazioni facoltative per abilitare l'autorizzazione per l'API. Per ulteriori informazioni, consulta the section called “La rotta `$connect`”.

- Authorization (Autorizzazione): se non è richiesta alcuna autorizzazione, puoi specificare `NONE`. In caso contrario, puoi specificare:
  - `AWS_IAM`per utilizzare le policy AWS IAM standard per controllare l'accesso alla tua API.
  - `CUSTOM` per implementare l'autorizzazione per un'API specificando una funzione di autorizzazione Lambda creata in precedenza. L'autorizzatore può risiedere nel tuo AWS account o in un altro AWS account. Per ulteriori informazioni sui provider di autorizzazioni Lambda, consulta Uso di autorizzazioni Lambda di API Gateway.

    > **Note**
    >
    > Nella console API Gateway, l'impostazione `CUSTOM` è visibile solo dopo aver configurato una funzione di autorizzazione come descritto in the section called “Configurazione di un sistema di autorizzazione Lambda (console)”.

  > **Important**
  >
  > L'impostazione Authorization (Autorizzazione) viene applicata all'intera API, non solo all'instradamento `$connect`. L'instradamento `$connect` protegge gli altri instradamenti, perché viene chiamato per ogni connessione.

- Chiave API necessaria: è possibile richiedere facoltativamente una chiave API per l'instradamento `$connect` di un'API. Puoi utilizzare le chiavi API insieme ai piani di utilizzo per controllare e tracciare l'accesso al tuo. APIs Per ulteriori informazioni, consulta the section called “Piani di utilizzo”.

Configurazione della richiesta di instradamento **`$connect`** mediante la console API Gateway

Per configurare la richiesta di `$connect` routing per un' WebSocket API utilizzando la console API Gateway:

1. Accedere alla console API Gateway, scegliere l'API e selezionare Routes (Route).
2. In Instradamenti scegli `$connect` o crea un instradamento `$connect` seguendo the section called “Creazione di una route tramite la console API Gateway”.
3. Nella sezione Impostazioni della richiesta di instradamento scegli Modifica.
4. Per Autorizzazione seleziona un tipo di autorizzazione.
5. Per richiedere un'API per l'instradamento `$connect` seleziona Richiedi chiave API.
6. Scegli Save changes (Salva modifiche).

Imposta le risposte di routing per WebSocket APIs in API Gateway

WebSocket le rotte possono essere configurate per comunicazioni bidirezionali o unidirezionali. API Gateway non trasferisce la risposta del back-end alla risposta di instradamento, a meno che non venga configurata una risposta di instradamento.

> **Note**
>
> È possibile definire solo la risposta del `$default` percorso per. WebSocket APIs È possibile utilizzare una risposta di integrazione per manipolare la risposta da un servizio back-end. Per ulteriori informazioni, consulta the section called “Panoramica delle risposte di integrazione”.

Puoi configurare le risposte di routing e le espressioni di selezione delle risposte utilizzando la console API Gateway o il AWS CLI o un AWS SDK.

Per ulteriori informazioni sulle espressioni di selezione delle risposte di instradamento, consulta the section called “”.

Argomenti

- Configurazione di una risposta di instradamento mediante la console API Gateway
- Imposta una risposta di routing utilizzando il AWS CLI

Configurazione di una risposta di instradamento mediante la console API Gateway

Dopo aver creato un' WebSocket API e collegato una funzione proxy Lambda alla route predefinita, puoi configurare la risposta alla route utilizzando la console API Gateway:

1. Accedi alla console API Gateway, scegli un' WebSocket API con integrazione della funzione proxy Lambda sul `$default` percorso.
2. In Routes (Route), scegliere l'instradamento `$default`.
3. Scegli Abilita la comunicazione bidirezionale.
4. Seleziona Deploy API (Distribuisci API).
5. Implementa l'API in una fase.

Per connettersi all'API, utilizzare il seguente comando wscat. Per ulteriori informazioni su `wscat`, consultare the section called “Utilizzalo `wscat` per connetterti a un' WebSocket API e inviarle messaggi”.

```
wscat -c wss://api-id.execute-api.us-east-2.amazonaws.com/test
```

Premere il pulsante INVIO per chiamare l'instradamento predefinito. Il comando dovrebbe restituire il corpo della funzione Lambda.

Imposta una risposta di routing utilizzando il AWS CLI

Il create-route-responsecomando seguente crea una risposta di rotta per la `$default` rotta. È possibile identificare l’ID dell’API e l’ID dell’instradamento utilizzando i comandi get-apis e get-routes.

```
aws apigatewayv2 create-route-response \
    --api-id aabbccddee \
    --route-id 1122334  \
    --route-response-key '$default'
```

L'output sarà simile al seguente:

```
{
    "RouteResponseId": "abcdef",
    "RouteResponseKey": "$default"
}
```

Imposta una **`$connect`** route che richiede un WebSocket sottoprotocollo

I client possono utilizzare il `Sec-WebSocket-Protocol` campo per richiedere un WebSocket sottoprotocollo durante la connessione all'API. WebSocket È possibile impostare un'integrazione per la route `$connect` in modo da consentire le connessioni solo se un client richiede un sottoprotocollo supportato dall'API.

La funzione Lambda di esempio seguente restituisce l'intestazione `Sec-WebSocket-Protocol` ai client. La funzione stabilisce una connessione all'API solo se il client specifica il sottoprotocollo `myprotocol`.

Per un CloudFormation modello che crea questo esempio di integrazione tra API e proxy Lambda, consulta. `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;
}
```

È possibile utilizzare `wscat` per verificare che l'API consenta le connessioni solo se un client richiede un sottoprotocollo supportato dall'API. I comandi seguenti utilizzano il flag `-s` per specificare i sottoprotocolli durante la connessione.

Il comando seguente tenta una connessione con un sottoprotocollo non supportato. Poiché il client ha specificato il sottoprotocollo `chat1`, l'integrazione Lambda restituisce un errore 400 e la connessione non riesce.

```
wscat -c wss://api-id.execute-api.region.amazonaws.com/beta -s chat1
error: Unexpected server response: 400
```

Il comando seguente include un sottoprotocollo supportato nella richiesta di connessione. L'integrazione Lambda consente la connessione.

```
wscat -c wss://api-id.execute-api.region.amazonaws.com/beta -s chat1,myprotocol
connected (press CTRL+C to quit)
```

Per ulteriori informazioni sull'invocazione WebSocket APIs, consulta. Invoca WebSocket APIs

Controlla e gestisci l'accesso WebSocket APIs a API Gateway

API Gateway supporta diversi meccanismi per il controllo e la gestione dell'accesso all' WebSocket API.

Puoi usare i seguenti meccanismi per l'autenticazione e l'autorizzazione:

- I ruoli e le policy AWS IAM standard offrono controlli di accesso flessibili e robusti. Puoi utilizzare i ruoli e le policy IAM per controllare chi può creare e gestire APIs i tuoi e chi può richiamarli. Per ulteriori informazioni, consulta Controlla l'accesso a WebSocket APIs con l'autorizzazione IAM.
- I tag IAM possono essere utilizzati insieme alle policy IAM per controllare l'accesso. Per ulteriori informazioni, consulta Utilizzo dei tag per controllare l'accesso alle risorse REST API di Gateway API.

- Gli autorizzatori Lambda sono funzioni Lambda che controllano l'accesso a. APIs Per ulteriori informazioni, consulta Controlla l'accesso WebSocket APIs con gli autorizzatori AWS Lambda REQUEST.

Per migliorare il tuo livello di sicurezza, ti consigliamo di configurare un autorizzatore per il `$connect` percorso su tutti i tuoi. WebSocket APIs Potrebbe essere necessario eseguire questa operazione per rispettare vari framework di conformità. Per ulteriori informazioni, consultare Amazon API Gateway controls nella Guida per l'utente di AWS Security Hub .

Argomenti

- Controlla l'accesso a WebSocket APIs con l'autorizzazione IAM
- Controlla l'accesso WebSocket APIs con gli autorizzatori AWS Lambda REQUEST

Controlla l'accesso a WebSocket APIs con l'autorizzazione IAM

L'autorizzazione IAM in WebSocket APIs è simile a quella per REST APIs, con le seguenti eccezioni:

- L'azione `execute-api` supporta `ManageConnections` oltre alle azioni esistenti (`Invoke`, `InvalidateCache`). `ManageConnections` controlla l'accesso all'API @connections.
- WebSocket i percorsi utilizzano un formato ARN diverso:

```
arn:aws:execute-api:region:account-id:api-id/stage-name/route-key
```

- L'`@connections`API utilizza lo stesso formato ARN di REST: APIs

```
arn:aws:execute-api:region:account-id:api-id/stage-name/POST/@connections
```

> **Important**
>
> Quando usi l'autorizzazione IAM è necessario firmare le richieste con Signature Version 4 (SigV4).

Puoi, ad esempio, configurare la seguente policy per il client. Questo esempio consente a chiunque di inviare un messaggio (`Invoke`) per tutti gli instradamenti ad eccezione di un instradamento

segreto nella fase `prod` e impedisce l'invio di un messaggio da qualunque utente ai client connessi (`ManageConnections`) in tutte le fasi.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "execute-api:Invoke"
            ],
            "Resource": [
                "arn:aws:execute-api:us-east-1:111122223333:api-id/prod/*"
            ]
        },
        {
            "Effect": "Deny",
            "Action": [
                "execute-api:Invoke"
            ],
            "Resource": [
                "arn:aws:execute-api:us-east-1:111122223333:api-id/prod/secret"
            ]
        },
        {
            "Effect": "Deny",
            "Action": [
                "execute-api:ManageConnections"
            ],
            "Resource": [
                "arn:aws:execute-api:us-east-1:111122223333:api-id/*"
            ]
        }
    ]
}
```

Controlla l'accesso WebSocket APIs con gli autorizzatori AWS Lambda REQUEST

Una funzione di autorizzazione Lambda in WebSocket APIs è simile a quella di REST APIs, con le seguenti eccezioni:

- È possibile usare solo una funzione di autorizzazione Lambda per l'instradamento `$connect`.
- Non è possibile utilizzare variabili di percorso (`event.pathParameters`), perché il percorso è fisso.
- `event.methodArn` è diverso dall'API REST equivalente, perché non ha un metodo HTTP. Nel caso di `$connect`, `methodArn` termina con `"$connect"`:

  ```
  arn:aws:execute-api:region:account-id:api-id/stage-name/$connect
  ```

- Le variabili di contesto in `event.requestContext` sono diverse da quelle di REST. APIs

L'esempio seguente mostra un input a un `REQUEST` autorizzatore per un' 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"
    }
}
```

Il seguente esempio di funzione di autorizzazione Lambda è una WebSocket versione della funzione di autorizzazione Lambda per REST in: APIs the section called “Esempi aggiuntivi di funzioni del sistema di autorizzazione Lambda”

Node.js

```
  // A simple REQUEST authorizer example to demonstrate how to use request
  // parameters to allow or deny a request. In this example, a request is
  // authorized if the client-supplied HeaderAuth1 header and QueryString1 query
parameter
  // in the request context match the specified values of
  // of 'headerValue1' and 'queryValue1' respectively.
           export const handler = function(event, context, callback) {
   console.log('Received event:', JSON.stringify(event, null, 2));

  // Retrieve request parameters from the Lambda function input:
  var headers = event.headers;
  var queryStringParameters = event.queryStringParameters;
  var stageVariables = event.stageVariables;
  var requestContext = event.requestContext;

  // Parse the input for the parameter values
  var tmp = event.methodArn.split(':');
  var apiGatewayArnTmp = tmp[5].split('/');
  var awsAccountId = tmp[4];
  var region = tmp[3];
  var ApiId = apiGatewayArnTmp[0];
```

```
    var stage = apiGatewayArnTmp[1];
    var route = apiGatewayArnTmp[2];

    // Perform authorization to return the Allow policy for correct parameters and
    // the 'Unauthorized' error, otherwise.
    var authResponse = {};
    var condition = {};
     condition.IpAddress = {};

    if (headers.HeaderAuth1 === "headerValue1"
        && queryStringParameters.QueryString1 === "queryValue1") {
         callback(null, generateAllow('me', event.methodArn));
     }  else {
         callback(null, generateDeny('me', event.methodArn));
     }
}

// Helper function to generate an IAM policy
var generatePolicy = function(principalId, effect, resource) {
    // Required output:
    var authResponse = {};
     authResponse.principalId = principalId;
    if (effect && resource) {
        var policyDocument = {};
         policyDocument.Version = '2012-10-17       '; // default version
        policyDocument.Statement = [];
        var statementOne = {};
         statementOne.Action = 'execute-api:Invoke'; // default action
        statementOne.Effect = effect;
         statementOne.Resource = resource;
         policyDocument.Statement[0] = statementOne;
         authResponse.policyDocument = policyDocument;
     }
    // Optional output with custom properties of the String, Number or Boolean type.
    authResponse.context = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": true
     };
    return authResponse;
}

var generateAllow = function(principalId, resource) {
    return generatePolicy(principalId, 'Allow', resource);
```

```
}

var generateDeny = function(principalId, resource) {
   return generatePolicy(principalId, 'Deny', resource);
}
```

Python

```
# A simple REQUEST authorizer example to demonstrate how to use request
# parameters to allow or deny a request. In this example, a request is
# authorized if the client-supplied HeaderAuth1 header and QueryString1 query
 parameter
# in the request context match the specified values of
# of 'headerValue1' and 'queryValue1' respectively.

import json


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

    # Retrieve request parameters from the Lambda function input:
    headers = event['headers']
    queryStringParameters = event['queryStringParameters']
    stageVariables = event['stageVariables']
    requestContext = event['requestContext']

    # Parse the input for the parameter values
    tmp = event['methodArn'].split(':')
    apiGatewayArnTmp = tmp[5].split('/')
    awsAccountId = tmp[4]
    region = tmp[3]
    ApiId = apiGatewayArnTmp[0]
    stage = apiGatewayArnTmp[1]
    route = apiGatewayArnTmp[2]

    # Perform authorization to return the Allow policy for correct parameters
    # and the 'Unauthorized' error, otherwise.

    authResponse = {}
    condition = {}
    condition['IpAddress'] = {}
```

```
    if (headers['HeaderAuth1'] ==
            "headerValue1" and queryStringParameters["QueryString1"] ==
 "queryValue1"):
        response = generateAllow('me', event['methodArn'])
        print('authorized')
        return json.loads(response)
    else:
        response = generateDeny('me', event['methodArn'])
        print('unauthorized')
        return json.loads(response)

    # Help function to generate IAM policy


def generatePolicy(principalId, effect, resource):
    authResponse = {}
    authResponse['principalId'] = principalId
    if (effect and resource):
        policyDocument = {}
        policyDocument['Version'] = '2012-10-17       '
        policyDocument['Statement'] = []
        statementOne = {}
        statementOne['Action'] = 'execute-api:Invoke'
        statementOne['Effect'] = effect
        statementOne['Resource'] = resource
        policyDocument['Statement'] = [statementOne]
        authResponse['policyDocument'] = policyDocument

    authResponse['context'] = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": True
    }

    authResponse_JSON = json.dumps(authResponse)

    return authResponse_JSON


def generateAllow(principalId, resource):
    return generatePolicy(principalId, 'Allow', resource)


def generateDeny(principalId, resource):
```

```
    return generatePolicy(principalId, 'Deny', resource)
```

Per configurare la precedente funzione Lambda come funzione di autorizzazione per REQUEST WebSocket un'API, segui la stessa procedura utilizzata per REST. APIs

Per configurare la route`$connect` per l'utilizzo di questo provider di autorizzazioni Lambda nella console, seleziona o crea la route `$connect`. Nella sezione Impostazioni della richiesta di instradamento scegli Modifica. Seleziona l'autorizzazione nel menu a discesa Autorizzazione, quindi scegli Salva modifiche.

Per verificare l'autorizzatore, è necessario creare una nuova connessione. La modifica dell'autorizzatore in `$connect` non influisce sul client già connesso. Quando ti connetti alla tua WebSocket API, devi fornire valori per tutte le fonti di identità configurate. Ad esempio, puoi connetterti inviando una stringa di query e un'intestazione valide utilizzando `wscat` come nell'esempio seguente:

```
wscat -c 'wss://myapi.execute-api.us-east-1.amazonaws.com/beta?
QueryString1=queryValue1' -H HeaderAuth1:headerValue1
```

Se tenti di connetterti senza un valore di identità valido, riceverai una risposta `401`:

```
wscat -c wss://myapi.execute-api.us-east-1.amazonaws.com/beta
error: Unexpected server response: 401
```

Integrazioni per API WebSocket APIs Gateway

Dopo aver impostato una route API, devi integrarla con un endpoint nel back-end. Un endpoint di backend viene anche definito endpoint di integrazione e può essere una funzione Lambda, un endpoint HTTP o un'azione di servizio. AWS L'integrazione dell'API dispone di una richiesta e di una risposta di integrazione.

In questa sezione, puoi imparare come configurare le richieste di integrazione e le risposte di integrazione per la tua API. WebSocket

Argomenti

- Configurare una richiesta di integrazione WebSocket API in API Gateway
- Configurare una risposta di integrazione WebSocket API in API Gateway

Configurare una richiesta di integrazione WebSocket API in API Gateway

La configurazione di una richiesta di integrazione comporta le seguenti operazioni:

- Selezione di una chiave di instradamento da integrare nel back-end.
- Specificazione dell'endpoint di backend da richiamare. WebSocket APIs supporta i seguenti tipi di integrazione:
  - `AWS_PROXY`
  - `AWS`
  - `HTTP_PROXY`
  - `HTTP`
  - `MOCK`

  Per ulteriori informazioni sui tipi di integrazione, consulta IntegrationTypel'API REST di API Gateway V2.
- Configurazione della modalità di trasformazione dei dati della richiesta di instradamento, se necessario, in dati di richiesta di integrazione specificando uno o più modelli di richiesta.

Configurare una richiesta di integrazione WebSocket API utilizzando la console API Gateway

Per aggiungere una richiesta di integrazione a un percorso in un' WebSocket API utilizzando la console API Gateway

1. Accedere alla console API Gateway, scegliere l'API e selezionare Routes (Route).
2. In Routes (Route), scegliere la route.
3. Scegli la scheda Richiesta di integrazione, quindi seleziona Modifica nella sezione Impostazioni della richiesta di integrazione.
4. Per Tipo di integrazione scegli una delle seguenti opzioni:

   - Scegli la funzione Lambda solo se la tua API sarà integrata con una AWS Lambda funzione che hai già creato in questo account o in un altro account.

     Per creare una nuova funzione Lambda in AWS Lambda, impostare un'autorizzazione di risorsa sulla funzione Lambda o eseguire qualsiasi altra azione del servizio Lambda, scegli invece Service.AWS

- Scegliere HTTP se l'API verrà integrata con un endpoint HTTP esistente. Per ulteriori informazioni, consulta Integrazioni HTTP per REST APIs in API Gateway.
- Scegliere Mock (Fittizio) se si desidera generare le risposte API direttamente da API Gateway, senza la necessità di un back-end di integrazione. Per ulteriori informazioni, consulta Integrazioni fittizie per REST APIs in API Gateway.
- Scegli AWS service se la tua API sarà integrata con un servizio. AWS
- Scegli Collegamento VPC se l'API utilizzerà un `VpcLink` come endpoint di integrazione privato. Per ulteriori informazioni, consulta Configura un'integrazione privata.

5. Se scegli Funzione Lambda procedi come segue:

   a. Per Utilizza integrazione proxy Lambda seleziona la casella di controllo desiderata per utilizzare l'integrazione proxy Lambda o l'integrazione proxy Lambda tra più account.

   b. Per Funzione Lambda specifica la funzione in uno dei modi seguenti:

      - Se la funzione Lambda si trova nello stesso account, immetti il nome della funzione e quindi seleziona la funzione dall'elenco a discesa.

        > **Note**
        >
        > Il nome della funzione può facoltativamente includere l'alias o la specifica della versione, come in `HelloWorld`, `HelloWorld:1` o `HelloWorld:alpha`.

      - Se la funzione si trova in un account diverso, immettere l'ARN per la funzione.

   c. Per utilizzare il valore di timeout predefinito di 29 secondi, mantieni attiva l'opzione Timeout predefinito. Per impostare un timeout personalizzato, scegli Timeout predefinito e immetti un valore di timeout compreso tra `50` e `29000` millisecondi.

6. Se si sceglie HTTP, seguire le istruzioni nella fase 4 di the section called “ Configurazione di una richiesta di integrazione tramite la console”.
7. Se si sceglie Mock (Fittizio), andare alla fase Request Templates (Modelli di richiesta).
8. Se scegli Servizio AWS segui le istruzioni della fase 6 in the section called “ Configurazione di una richiesta di integrazione tramite la console”.
9. Se scegli Collegamento VPC procedi come segue:

   a. Per Integrazione proxy VPC seleziona la casella di controllo se desideri eseguire il proxy delle richieste sull'endpoint di `VPCLink`.

b. Per HTTP method (Metodo HTTP) scegliere il tipo di metodo HTTP che corrisponde maggiormente al metodo nel back-end HTTP.

c. Dall'elenco a discesa Collegamento VPC seleziona un collegamento VPC. È possibile selezionare `[Use Stage Variables]` e inserire **`${stageVariables.vpcLinkId}`** nella casella di testo sotto l'elenco.

   È possibile definire la variabile di fase `vpcLinkId` dopo l'implementazione dell'API in una fase e l'impostazione del suo valore sull'ID del `VpcLink`.

d. In Endpoint URL (URL endpoint), immettere l'URL del back-end HTTP che l'integrazione deve utilizzare.

e. Per utilizzare il valore di timeout predefinito di 29 secondi, mantieni attiva l'opzione Timeout predefinito. Per impostare un timeout personalizzato, scegli Timeout predefinito e immetti un valore di timeout compreso tra `50` e `29000` millisecondi.

10. Scegli Save changes (Salva modifiche).
11. In Modelli di richiesta procedi come segue:

    a. Per immettere un'Espressione di selezione del modello scegli Modifica in Modelli di richiesta.

    b. Immetti un'Espressione di selezione del modello. Specifica un'espressione che viene ricercata da Gateway API nel payload del messaggio. Se viene trovata, viene valutata e il risultato è un valore chiave del modello che viene utilizzato per selezionare il modello di mappatura dei dati che deve essere applicato ai dati nel payload del messaggio. Il modello di mappatura dei dati viene creato nel passaggio successivo. Scegli Modifica per salvare le modifiche.

    c. Scegli Crea modello per creare il modello di mappatura dei dati. Per Chiave del modello immetti il valore della chiave del modello utilizzato per selezionare il modello di mappatura dei dati da applicare ai dati nel payload del messaggio. Quindi, specifica un modello di mappatura. Scegli Crea modello.

       Per ulteriori informazioni sulle espressioni di selezione del modello, consulta the section called “Espressioni di selezione del modello”.

Imposta una richiesta di integrazione utilizzando il AWS CLI

È possibile impostare una richiesta di integrazione per un percorso in un' WebSocket API utilizzando AWS CLI come illustrato nell'esempio seguente, che crea un'integrazione fittizia:

1. Crea un file denominato `integration-params.json`, con i seguenti contenuti:

```
{"PassthroughBehavior": "WHEN_NO_MATCH", "TimeoutInMillis": 29000,
 "ConnectionType": "INTERNET", "RequestTemplates": {"application/json":
 "{\"statusCode\":200}"}, "IntegrationType": "MOCK"}
```

2. Utilizza il seguente comando create-integration per creare l'integrazione fittizia.

```
aws apigatewayv2 --region us-east-1 create-integration --api-id aabbccddee --cli-
input-json file://integration-params.json
```

L'output sarà simile al seguente:

```
{
    "PassthroughBehavior": "WHEN_NO_MATCH",
    "TimeoutInMillis": 29000,
    "ConnectionType": "INTERNET",
    "IntegrationResponseSelectionExpression": "${response.statuscode}",
    "RequestTemplates": {
        "application/json": "{\"statusCode\":200}"
    },
    "IntegrationId": "0abcdef",
    "IntegrationType": "MOCK"
}
```

In alternativa, è possibile impostare una richiesta di integrazione per un'integrazione proxy utilizzando il AWS CLI.

1. Creare una funzione Lambda nella console Lambda e assegnarle un ruolo di esecuzione Lambda di base.
2. Utilizza il seguente comando create-integration per creare l'integrazione.

```
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
```

L'output sarà simile al seguente:

```
{
    "PassthroughBehavior": "WHEN_NO_MATCH",
    "IntegrationMethod": "POST",
    "TimeoutInMillis": 29000,
    "ConnectionType": "INTERNET",
    "IntegrationUri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123412341234:function:simpleproxy-echo-e2e/invocations",
    "IntegrationId": "abcdefg",
    "IntegrationType": "AWS_PROXY"
}
```

Formato di input di una funzione Lambda per l'integrazione del proxy per WebSocket APIs

Con l'integrazione proxy Lambda, API Gateway mappa l'intera richiesta client al parametro di input `event` della funzione Lambda di back-end. Nell'esempio seguente viene illustrata la struttura dell'evento di input dall'instradamento `$connect` e dell'evento di input dall'instradamento `$disconnect` che Gateway API invia a un'integrazione proxy Lambda.

Input from the $connect route

```
{
    headers: {
      Host: 'abcd123.execute-api.us-east-1.amazonaws.com',
      'Sec-WebSocket-Extensions': 'permessage-deflate; client_max_window_bits',
      'Sec-WebSocket-Key': '...',
      'Sec-WebSocket-Version': '13',
      'X-Amzn-Trace-Id': '...',
      'X-Forwarded-For': '192.0.2.1',
      'X-Forwarded-Port': '443',
      'X-Forwarded-Proto': 'https'
    },
    multiValueHeaders: {
      Host: [ 'abcd123.execute-api.us-east-1.amazonaws.com' ],
      'Sec-WebSocket-Extensions': [ 'permessage-deflate; client_max_window_bits' ],
      'Sec-WebSocket-Key': [ '...' ],
      'Sec-WebSocket-Version': [ '13' ],
      'X-Amzn-Trace-Id': [ '...' ],
      'X-Forwarded-For': [ '192.0.2.1' ],
      'X-Forwarded-Port': [ '443' ],
      'X-Forwarded-Proto': [ 'https' ]
    },
    requestContext: {
```

```
      routeKey: '$connect',
      eventType: 'CONNECT',
      extendedRequestId: 'ABCD1234=',
      requestTime: '09/Feb/2024:18:11:43 +0000',
      messageDirection: 'IN',
      stage: 'prod',
      connectedAt: 1707502303419,
      requestTimeEpoch: 1707502303420,
      identity: { sourceIp: '192.0.2.1' },
      requestId: 'ABCD1234=',
      domainName: 'abcd1234.execute-api.us-east-1.amazonaws.com',
      connectionId: 'AAAA1234=',
      apiId: 'abcd1234'
    },
    isBase64Encoded: false
  }
```

Input from the $disconnect route

```
{
    headers: {
      Host: 'abcd1234.execute-api.us-east-1.amazonaws.com',
      'x-api-key': '',
      'X-Forwarded-For': '',
      'x-restapi': ''
    },
    multiValueHeaders: {
      Host: [ 'abcd1234.execute-api.us-east-1.amazonaws.com' ],
      'x-api-key': [ '' ],
      'X-Forwarded-For': [ '' ],
      'x-restapi': [ '' ]
    },
    requestContext: {
      routeKey: '$disconnect',
      disconnectStatusCode: 1005,
      eventType: 'DISCONNECT',
      extendedRequestId: 'ABCD1234=',
      requestTime: '09/Feb/2024:18:23:28 +0000',
      messageDirection: 'IN',
      disconnectReason: 'Client-side close frame status not set',
      stage: 'prod',
      connectedAt: 1707503007396,
```

```
    requestTimeEpoch: 1707503008941,
    identity: { sourceIp: '192.0.2.1' },
    requestId: 'ABCD1234=',
    domainName: 'abcd1234.execute-api.us-east-1.amazonaws.com',
    connectionId: 'AAAA1234=',
    apiId: 'abcd1234'
  },
  isBase64Encoded: false
}
```

Configurare una risposta di integrazione WebSocket API in API Gateway

La sezione seguente fornisce una breve panoramica delle risposte di integrazione per l' WebSocket API e su come impostare una risposta di integrazione per un' WebSocket API.

Argomenti

- Panoramica delle risposte di integrazione
- Risposte di integrazione per la comunicazione bidirezionale
- Configurazione di una risposta di integrazione mediante la console API Gateway
- Imposta una risposta di integrazione utilizzando il AWS CLI

Panoramica delle risposte di integrazione

La risposta di integrazione di API Gateway è un metodo di creazione di modelli e di manipolazione della risposta da un servizio di back-end. Esistono alcune differenze nella configurazione di un'API REST rispetto a una risposta di integrazione WebSocket API, ma concettualmente il comportamento è lo stesso.

WebSocket le rotte possono essere configurate per la comunicazione bidirezionale o unidirezionale.

- Quando una route è configurata per la comunicazione bidirezionale, una risposta di integrazione consente di configurare le trasformazioni sul payload del messaggio restituito, in modo simile alle risposte di integrazione per REST. APIs
- Se una route è configurata per la comunicazione unidirezionale, indipendentemente dalla configurazione della risposta di integrazione, non verrà restituita alcuna risposta sul WebSocket canale dopo l'elaborazione del messaggio.

API Gateway non trasferisce la risposta del back-end alla risposta di instradamento, a meno che non venga configurata una risposta di instradamento. Per informazioni sull'impostazione di una risposta di instradamento, consultare the section called “Configura le risposte di routing dell' WebSocket API”.

Risposte di integrazione per la comunicazione bidirezionale

Le integrazioni possono essere divise in integrazioni proxy e integrazioni non proxy.

> **Important**
>
> Per le integrazioni proxy, API Gateway trasferisce automaticamente l'output del back-end all’intermediario come payload completo. Non esiste alcuna risposta di integrazione.

Per integrazioni non proxy, è necessario configurare almeno una risposta di integrazione:

- Idealmente, una delle risposte di integrazione deve agire come un metodo catch-all quando non è possibile effettuare una scelta esplicita. Questo caso predefinito viene rappresentato impostando una chiave di risposta di integrazione di `$default`.
- In tutti gli altri casi, la chiave di risposta di integrazione funziona come un'espressione regolare. Il formato adottato è `"/expression/"`.

Per integrazioni HTTP non proxy:

- API Gateway tenterà di trovare una corrispondenza con il codice di stato HTTP della risposta di back-end. In questo caso, la chiave di risposta di integrazione funzionerà come un'espressione regolare. Se non è possibile trovare una corrispondenza, viene scelta `$default` come risposta di integrazione.
- L'espressione di selezione del modello, come descritto in precedenza, funziona in modo identico. Ad esempio:
  - `/2\d\d/`: ricevi e trasforma risposte andate a buon fine
  - `/4\d\d/`: ricevi e trasforma errori di richiesta non valida
  - `$default`: ricevi e trasforma tutte le risposte impreviste

Per ulteriori informazioni sulle espressioni di selezione del modello, consultare the section called “Espressioni di selezione del modello”.

Configurazione di una risposta di integrazione mediante la console API Gateway

Per configurare una risposta di integrazione del percorso per un' WebSocket API utilizzando la console API Gateway:

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli la tua WebSocket API e scegli il tuo percorso.
3. Scegli la scheda Richiesta di integrazione, quindi seleziona Crea risposta di integrazione nella sezione Impostazioni della risposta di integrazione.
4. Per Chiave della risposta inserisci un valore che verrà individuato nella chiave della risposta del messaggio in uscita dopo aver valutato l'espressione di selezione di risposta. Ad esempio, puoi inserire **`/4\d\d/`** per ricevere e trasformare gli errori di richiesta non valida o **`$default`** per ricevere e trasformare tutte le risposte che corrispondono all'espressione di selezione del modello.
5. Per Espressione di selezione del modello immetti un'espressione di selezione per valutare il messaggio in uscita.
6. Scegli Crea risposta.
7. È inoltre possibile definire un modello di mappatura per configurare le trasformazioni del payload dei messaggi restituiti. Scegli Crea modello.
8. Immettere un nome per la chiave. Se è stata scelta l'espressione di selezione del modello predefinita, immettere**`\$default`**.
9. Per Modello della risposta immetti il modello di mappatura nell'editor di codice.
10. Scegli Crea modello.
11. Scegli Implementa API per implementare l'API.

Per connettersi all'API, utilizzare il seguente comando wscat. Per ulteriori informazioni su `wscat`, consultare the section called “Utilizzalo `wscat` per connetterti a un' WebSocket API e inviarle messaggi”.

```
wscat -c wss://api-id.execute-api.us-east-2.amazonaws.com/test
```

Quando viene richiamato l'instradamento, dovrebbe venire restituito il payload del messaggio restituito.

Imposta una risposta di integrazione utilizzando il AWS CLI

Il create-integration-responsecomando seguente crea una risposta di `$default` integrazione:

```
aws apigatewayv2 create-integration-response \
    --api-id vaz7da96z6 \
    --integration-id a1b2c3 \
    --integration-response-key '$default'
```

Richiedi la convalida per WebSocket APIs in API Gateway

Puoi configurare API Gateway per eseguire la convalida su una route prima di passare alla richiesta di integrazione. Se la convalida fallisce, API Gateway fallisce la richiesta senza chiamare il backend, invia una risposta gateway «Bad request body» al client e pubblica i risultati della convalida in Logs. CloudWatch L'utilizzo della convalida in questo modo riduce le chiamate non necessarie al back-end dell'API.

Espressioni di selezione del modello

Puoi utilizzare un'espressione di selezione del modello per convalidare dinamicamente le richieste all'interno della stessa route. La convalida del modello si verifica se si fornisce un'espressione di selezione del modello per integrazioni proxy o non proxy. Potrebbe essere necessario definire il modello `$default` come un fallback quando non viene trovato alcun modello corrispondente. Se non esiste un modello corrispondente e `$default` non è definito, la convalida non va a buon fine. L'aspetto dell'espressione di selezione è simile a `Route.ModelSelectionExpression` e valuta la chiave per `Route.RequestModels`.

Quando definisci un percorso per un' WebSocket API, puoi facoltativamente specificare un'espressione di selezione del modello. Questa espressione viene valutata per selezionare il modello da utilizzare per la convalida del corpo quando si riceve una richiesta. L'espressione restituisce una delle voci presenti nel di una route `requestmodels`.

Un modello viene espresso come schema JSON e descrive la struttura dati del corpo della richiesta. La natura di queste espressioni di selezione consente di scegliere in modo dinamico il modello in base al quale eseguire la convalida in fase di runtime per una determinata route. Per informazioni su come creare un modello, consulta the section called “Modelli di dati per REST API”.

Configurazione della convalida delle richieste tramite la console Gateway Amazon API

L'esempio seguente mostra come configurare la convalida delle richieste per un instradamento.

Innanzitutto, si crea un modello e quindi un instradamento. Successivamente, si configura la convalida delle richieste sull'instradamento appena creato. Infine, si implementa e si esegue il test dell'API. Per completare questo tutorial, è necessaria un' WebSocket API `$request.body.action` come espressione di selezione del percorso e un endpoint di integrazione per la nuova rotta.

Per la connessione all'API è inoltre necessario `wscat`. Per ulteriori informazioni, consulta the section called “Utilizzalo `wscat` per connetterti a un' WebSocket API e inviarle messaggi”.

Creazione di un modello

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli un'API. WebSocket
3. Nel riquadro di navigazione principale seleziona Modelli.
4. Scegli Crea modello.
5. In Nome, inserisci **`emailModel`**.
6. Per Tipo di contenuto inserisci **`application/json`**.
7. Per Schema modello immetti il seguente modello:

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type" : "object",
    "required" : [ "address"],
    "properties" : {
        "address": {
            "type": "string"
        }
    }
}
```

   Per questo modello è necessario che la richiesta contenga un indirizzo e-mail.
8. Scegli Save (Salva).

In questo passaggio, crei un percorso per la tua WebSocket API.

Come creare una route

1. Nel pannello di navigazione principale scegli Instradamenti.
2. Selezionare Create Route (Crea route).

3. Per Chiave instradamento, inserisci **`sendMessage`**.
4. Scegli un tipo di integrazione e specificare un endpoint di integrazione. Per ulteriori informazioni, consulta the section called “Integrazioni”.
5. Selezionare Create Route (Crea route).

In questo passaggio si imposta la convalida delle richieste per l'instradamento `sendMessage`.

Per impostare la convalida delle richieste

1. Nella scheda Richiesta di instradamento, scegli Modifica in Impostazioni della richiesta di instradamento.
2. Per Espressione di selezione del modello immetti **`${request.body.messageType}`**.

   Gateway API utilizza la proprietà `messageType` per convalidare la richiesta in arrivo.
3. Scegli Aggiungi modello di richiesta.
4. Per Chiave modello inserisci **`email`**.
5. Per Modello scegli emailModel.

   Gateway API convalida i messaggi in arrivo con la proprietà `messageType` impostata su `email` per questo modello.

   > **Note**
   >
   > Se Gateway API non riesce ad abbinare l'espressione di selezione del modello a una chiave del modello, viene selezionato il modello `$default`. Se non esiste un modello `$default`, la convalida non va a buon fine. Per la produzione APIs, ti consigliamo di creare un `$default` modello.
6. Scegli Save changes (Salva modifiche).

In questo passaggio si implementa e si testa l'API.

Per distribuire e testare l'API

1. Seleziona Deploy API (Distribuisci API).
2. Scegliere la fase desiderata dall'elenco a discesa oppure immettere il nome di una nuova fase.
3. Seleziona Implementa.

4. Nel riquadro di navigazione principale scegli Fasi.
5. Copia l' WebSocket URL della tua API. L'URL dovrebbe essere del tipo `wss://abcdef123.execute-api.us-east-2.amazonaws.com/production`.
6. Apri un nuovo terminale ed eseguire il comando wscat con i parametri seguenti.

```
wscat -c wss://abcdef123.execute-api.us-west-2.amazonaws.com/production
```

```
Connected (press CTRL+C to quit)
```

7. Utilizza il seguente comando per testare l'API.

```
{"action": "sendMessage", "messageType": "email"}
```

```
{"message": "Invalid request body", "connectionId":"ABCD1=234",
 "requestId":"EFGH="}
```

Gateway API non riuscirà a completare la richiesta.

Utilizza il seguente comando per inviare una richiesta valida all'API.

```
{"action": "sendMessage", "messageType": "email", "address":
 "mary_major@example.com"}
```

Trasformazioni dei dati per API WebSocket APIs Gateway

In API Gateway, la richiesta del metodo di un' WebSocket API può accettare un payload in un formato diverso dal payload della richiesta di integrazione corrispondente, come richiesto nel backend. Analogamente, il back-end potrebbe restituire un payload delle risposta di integrazione diverso dal payload della risposta di metodo, in base alle aspettative del front-end.

Gateway API consente di utilizzare le trasformazioni dei modelli di mappatura per mappare il payload da una richiesta di metodo alla richiesta di integrazione corrispondente e da una risposta di integrazione alla corrispondente risposta di metodo. Si crea un modello di mappatura e si specifica un'espressione di selezione del modello per determinare quale modello utilizzare per eseguire le trasformazioni dei dati necessarie.

È possibile utilizzare le mappature dei dati per mappare i dati da una richiesta di instradamento a un'integrazione back-end. Per ulteriori informazioni, consulta the section called “Mappatura dei dati”.

Modelli di mappatura e modelli

Un modello di mappatura è uno script espresso in Velocity Template Language (VTL) e applicato al payload utilizzando espressioni. JSONPath Per ulteriori informazioni sui modelli di mappatura API Gateway, consulta Trasformazioni dei modelli di mappatura per REST API in Gateway API.

Il payload può avere un modello di dati in base alla bozza 4 dello schema JSON. Per generare un modello di mappatura, non è necessario definire un modello. Tuttavia, un modello può essere utile per crearne un altro, poiché API Gateway genera un piano di modello in base a un modello fornito. Per ulteriori informazioni sui modelli API Gateway, consulta Modelli di dati per REST API.

Espressioni di selezione del modello

Per trasformare un payload con un modello di mappatura, è necessario specificare un'espressione di selezione del modello WebSocket API in una richiesta di integrazione o in una risposta di integrazione. Questa espressione viene valutata per determinare il modello di input o di output (se disponibili) da usare per trasformare il corpo della richiesta nel corpo della richiesta di integrazione (tramite un modello di input) o il corpo della risposta di integrazione nel corpo della risposta di instradamento (tramite un modello di output).

`Integration.TemplateSelectionExpression` supporta `${request.body.jsonPath}` e valori statici.

`IntegrationResponse.TemplateSelectionExpression` supporta `${request.body.jsonPath}`, `${integration.response.statuscode}`, `${integration.response.header.headerName}`, `${integration.response.multivalueheader.headerName}` e valori statici.

Espressioni di selezione della risposta di integrazione

Quando imposti una risposta di integrazione per un' WebSocket API, puoi facoltativamente specificare un'espressione di selezione della risposta di integrazione. Questa espressione determina quale `IntegrationResponse` deve essere selezionata quando viene restituita un'integrazione. Il valore di questa espressione è attualmente limitato da API Gateway, come definito di seguito. Tieni presente che questa espressione è rilevante solo per le integrazioni non proxy; un'integrazione proxy restituisce semplicemente il payload della risposta all'intermediario senza alcuna modellazione o modifica.

A differenza di altre espressioni di selezione precedenti, questa espressione supporta attualmente un formato di corrispondenza di modelli. L'espressione deve essere racchiusa tra barre.

Attualmente il valore è fisso e dipende dal valore di `integrationType`:

- Per le integrazioni basate su Lambda, è `$integration.response.body.errorMessage`.
- Per le integrazioni `HTTP` e `MOCK`, è `$integration.response.statuscode`.
- Per `HTTP_PROXY` e `AWS_PROXY`, l'espressione non viene utilizzata poiché si richiede che il payload raggiunga il chiamante.

Configurare la mappatura dei dati per API WebSocket APIs Gateway

La mappatura dei dati consente di mappare i dati da una richiesta di instradamento a un'integrazione back-end.

> **Note**
>
> La mappatura dei dati per WebSocket APIs non è supportata in. Console di gestione AWSÈ necessario utilizzare AWS CLI AWS CloudFormation, o un SDK per configurare la mappatura dei dati.

Argomenti

- Come mappare i dati di richiesta di instradamento ai parametri di richiesta di integrazione
- Esempi

Come mappare i dati di richiesta di instradamento ai parametri di richiesta di integrazione

I parametri della richiesta di integrazione possono essere mappati da tutti i parametri di richiesta di instradamento definiti, dal corpo della richiesta `context o` dalle variabili `stage` e dai valori statici.

La tabella seguente mostra le espressioni di mappatura dei dati delle richieste di integrazione. Nella tabella *`PARAM_NAME`* è il nome del parametro di una richiesta di instradamento del tipo di parametro specifico. Deve corrispondere all'espressione regolare. `'^[a-zA-Z0-9._$-]+$]'` *`JSONPath_EXPRESSION`*è un' JSONPath espressione per un campo JSON del corpo della richiesta.

Origine dati mappata	Espressione di mappatura
Stringa di query di richiesta (supportata solo per l'instradamento `$connect`)	`route.request.querystring.` *`PARAM_NAME`*
Intestazione della richiesta (supportata solo per l'instradamento `$connect`)	`route.request.header.` *`PARAM_NAME`*
Stringa di query della richiesta a più valori (supportata solo per l'instradamento `$connect`)	`route.request.multivaluequerystring.` *`PARAM_NAME`*
Intestazione della richiesta a più valori (supportata solo per l'instradamento `$connect`)	`route.request.multivalueheader.` *`PARAM_NAME`*
Corpo di richiesta	`route.request.body.` *`JSONPath_EXPRESSION`*
Variabili di fase	`stageVariables.` *`VARIABLE_NAME`*
Variabili di contesto	`context.`*`VARIABLE_NAME`* che deve essere una delle variabili di contesto supportate.
Valore statico	*`'STATIC_VALUE'`* . La *`STATIC_VALUE`* è una stringa letterale e deve essere racchiusa tra virgolette singole.

Quando crei una mappatura dei dati, AWS CLI assicurati di seguire il formato corretto per utilizzare i valori letterali con stringhe in. AWS CLI Per ulteriori informazioni, consulta Using quotation marks and literals with strings in the AWS CLI nella Guida per l'utente di AWS Command Line Interface .

Esempi

I seguenti AWS CLI esempi configurano le mappature dei dati. Per un CloudFormation modello di esempio, vedere. `websocket-data-mapping.yaml`

Mappare connectionId di un client a un'intestazione in una richiesta di integrazione

Il comando update-integration seguente mappa `connectionId` di un client a un'intestazione `connectionId` nella richiesta a un'integrazione backend.

```
aws apigatewayv2 update-integration \
    --integration-id abc123 \
    --api-id a1b2c3d4 \
    --request-parameters
 'integration.request.header.connectionId'='context.connectionId'
```

Mappatura di un parametro di stringa di query a un'intestazione in una richiesta di integrazione

Il seguente esempio mappa un parametro della stringa di query `authToken` a un'intestazione `authToken` nella richiesta di integrazione.

1. Utilizza il seguente comando update-route per aggiungere il parametro della stringa di query `authToken` ai parametri della richiesta di instradamento.

   ```
   aws apigatewayv2 update-route --route-id 0abcdef \
       --api-id a1b2c3d4 \
       --request-parameters '{"route.request.querystring.authToken": {"Required":
    false}}'
   ```

2. Utilizza il seguente comando update-integration per mappare il parametro della stringa di query all'intestazione `authToken` nella richiesta di integrazione backend.

   ```
   aws apigatewayv2 update-integration \
       --integration-id abc123 \
       --api-id a1b2c3d4 \
       --request-parameters
    'integration.request.header.authToken'='route.request.querystring.authToken'
   ```

3. (Facoltativo) Se necessario, utilizzate quanto segue delete-route-request-parameterper eliminare il parametro della stringa di `authToken` query dai parametri di richiesta del percorso.

   ```
   aws apigatewayv2 delete-route-request-parameter \
       --route-id 0abcdef \
       --api-id a1b2c3d4 \
       --request-parameter-key 'route.request.querystring.authToken'
   ```

WebSocket Riferimento al modello di mappatura delle API per API Gateway

Questa sezione riassume l'insieme di variabili attualmente supportate per le API in WebSocket API Gateway.

Parametro	Description
`$context.connectionId`	Un ID univoco per la connessione che puoi utilizzare per effettuare una richiamata al client. L'ID di connessione è una stringa di lunghezza variabile la cui lunghezza e formato potrebbero cambiare.
`$context.connectedAt`	L'ora della connessione in formato Epoch.
`$context.domainName`	Un nome di dominio per l'API. WebSocket Può essere utilizzato per effettuare un callback al client (invece di un valore hardcoded).
`$context.eventType`	Il tipo di evento: `CONNECT`, `MESSAGE` o `DISCONNECT` .
`$context.messageId`	Un ID univoco sul lato server per un messaggio . Disponibile solo quando `$context.eventType` è `MESSAGE`.
`$context.routeKey`	La chiave di instradamento selezionata.
`$context.requestId`	Come `$context.extendedRequestId` .
`$context.extendedRequestId`	Un ID generato automaticamente per la chiamata API, che contiene altre informazioni utili per debugging/troubleshooting.
`$context.apiId`	Identificatore assegnato da API Gateway all'API.
`$context.authorizer.principalId`	Identificazione dell'utente dell'entità principale associata al token inviato dal client e restituit a da una funzione Lambda del provider di

Parametro	Description
	autorizzazioni Lambda di API Gateway (noto in precedenza come autorizzazioni ad hoc).
`$context.authorizer.` *`property`*	Valore in formato stringa della coppia chiave/valore specificata della mappa `context` restituita da una funzione delle autorizzazioni Lambda di API Gateway. Ad esempio, se le autorizzazioni restituiscono la mappa `context` seguente: `"context" : { "key": "value", "numKey": 1, "boolKey": true }` la chiamata di `$context.authorizer.key` restituisce la stringa `"value"`, la chiamata di `$context.authorizer.numKey` restituisce la stringa `"1"` e la chiamata di `$context.authorizer.boolKey` restituisce la stringa `"true"`.
`$context.error.messageString`	Valore `$context.error.message` tra virgolette, ovvero `"$context.error.message"`.
`$context.error.validationErrorString`	Stringa contenente un messaggio dettagliato di errore di convalida.
`$context.identity.accountId`	L'ID AWS dell'account associato alla richiesta.
`$context.identity.apiKey`	Chiave del proprietario dell'API associata alla richiesta API abilitata dalla chiave.
`$context.identity.apiKeyId`	ID chiave API associato alla richiesta API abilitata dalla chiave

Parametro	Description
`$context.identity.caller`	Identificatore dell'entità principale del chiamante da cui proviene la richiesta.
`$context.identity.cognitoAuthenticationProvider`	Un elenco separato da virgole con tutti i provider di autenticazione Amazon Cognito utilizzati dal chiamante che effettua la richiesta . Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito. Ad esempio, per un'identità di un pool di utenti Amazon Cognito, `cognito-idp.` *`region`*`.amazonaws.com/` *`user_pool_id`* `,cognito-idp.` *`region`*`.amazonaws.com/` *`user_pool_id`* `:CognitoSignIn:` *`token subject claim`* Per informazioni sui provider di autenticazione Amazon Cognito disponibili, consulta Using Federated Identities nella Guida per gli sviluppatori di Amazon Cognito.
`$context.identity.cognitoAuthenticationType`	Tipo di autenticazione Amazon Cognito dell'intermediario da cui proviene la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito. I valori possibili includono `authenticated` per le identità autenticate e `unauthenticated` per le identità non autenticate.
`$context.identity.cognitoIdentityId`	ID identità di Amazon Cognito dell'intermediario da cui proviene la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito.

Parametro	Description
`$context.identity.cognitoId entityPoolId`	ID pool di identità di Amazon Cognito dell'inte rmediario da cui proviene la richiesta. Disponibi le solo se la richiesta è stata firmata con credenziali Amazon Cognito.
`$context.identity.sourceIp`	L'indirizzo IP di origine della connessione TCP immediata da cui proviene la richiesta all'endpo int di API Gateway.
`$context.identity.user`	Identificatore dell'entità principale dell'utente da cui proviene la richiesta.
`$context.identity.userAgent`	Agente utente della chiamata API.
`$context.identity.userArn`	Amazon Resource Name (ARN) dell'utente valido identificato dopo l'autenticazione.
`$context.requestTime`	Ora della richiesta in formato CLF (`dd/MMM/yyyy:HH:mm:ss +-hhmm`).
`$context.requestTimeEpoch`	L'ora della richiesta in formato epoca (Unix epoch) in millisecondi.
`$context.stage`	Fase di distribuzione della chiamata API, ad esempio, beta o di produzione.
`$context.status`	Lo stato della risposta.
`$input.body`	Restituisce il payload non elaborato come stringa.

Parametro	Description
`$input.json(x)`	Questa funzione valuta un'espressione JSONPath e restituisce i risultati come stringa JSON. Ad esempio, `$input.json('$.pets')` restituisce una stringa JSON che rappresenta la struttura di elementi "pet" (animali domestici). Per ulteriori informazioni su JSONPath, consulta la pagina relativa a JSONPath o JSONPath per Java.

Parametro	Description
`$input.path(x)`	Da una stringa di espressione JSONPath (x) restituisce una rappresentazione di oggetto JSON del risultato. In questo modo, puoi accedere agli elementi del payload e modificarli in modo nativo in Apache Velocity Template Language (VTL). Ad esempio, se l'espressione `$input.path('$.pets')` restituisce un oggetto in questo modo: `[ { "id": 1, "type": "dog", "price": 249.99 }, { "id": 2, "type": "cat", "price": 124.99 }, { "id": 3, "type": "fish", "price": 0.99 } ]` `$input.path('$.pets').count()` restituisce "3". Per ulteriori informazioni su JSONPath, consulta la pagina relativa a JSONPath o JSONPath per Java.
`$stageVariables.` *`<variable_name>`*	*`<variable_name>`* rappresenta il nome di una variabile di fase.

Parametro	Description
`$stageVariables['` *`<variable_name>`* `']`	*`<variable_name>`* rappresenta qualsiasi nome di variabile di fase.
`${stageVariables['` *`<variable_name>`* `']}`	*`<variable_name>`* rappresenta qualsiasi nome di variabile di fase.
`$util.escapeJavaScript()`	Sfugge ai caratteri di una stringa utilizzando le regole delle JavaScript stringhe. **Note** Questa funzione trasforma qualsiasi virgoletta singola (`'`) in virgoletta preceduta da un carattere escape (`\'`). Tuttavia, le virgolette singole con escape non sono valide in JSON. Di conseguenza, quando l'output di questa funzione viene usato in una proprietà JSON, devi modificare di nuovo qualsiasi virgoletta singola con carattere escape (`\'`) in virgoletta singola normale (`'`). Questo viene mostrato nell'esempio seguente: `$util.escapeJavaScript( data).replaceAll("\\'",'')`

Parametro	Description

`$util.parseJson()`	Da una stringa JSON restituisce una rappresentazione oggetto del risultato. Puoi usare il risultato di questa funzione per accedere agli elementi del payload e modificarli in modo nativo in Apache Velocity Template Language (VTL). Ad esempio, in presenza del payload seguente:
	`{"errorMessage":"{\"key1\":\"var1\", \"key2\":{\"arr\":[1,2,3]}}"}`
	E se usi il modello di mappatura seguente:
	`#set ($errorMessageObj = $util.parseJson($input.path('$.errorMessage')))`
	`{`
	`"errorMessageObjKey2ArrVal" : $errorMessageObj.key2.arr[0]`
	`}`
	Otterrai l'output seguente:
	`{`
	`"errorMessageObjKey2ArrVal" : 1`
	`}`

`$util.urlEncode()`	Converte una stringa nel formato "application/x-www-form-urlencoded».
`$util.urlDecode()`	Decodifica una stringa application/x "-www-form-urlencoded».
`$util.base64Encode()`	Codifica i dati in una stringa con codifica base64.

Parametro	Description
`$util.base64Decode()`	Decodifica i dati da una stringa con codifica base64.

Tipi di supporti binari per API WebSocket in Gateway API

Le API WebSocket API Gateway non supportano attualmente frame binari nei payload dei messaggi in entrata. Se un'app client invia un frame binario, viene rifiutato da API Gateway e la connessione al client viene annullata con codice 1003.

Esiste una soluzione alternativa a questo comportamento. Se il client invia dati binari con codifica di testo (ad esempio, base64) come un frame di testo, puoi impostare la proprietà `contentHandlingStrategy` dell'integrazione su `CONVERT_TO_BINARY` per convertire il payload da stringa con codifica base64 in binario.

Per restituire una risposta di instradamento per un payload binario in integrazioni non proxy, puoi impostare la proprietà `contentHandlingStrategy` della risposta di integrazione su `CONVERT_TO_TEXT` per convertire il payload da binario in stringa con codifica base64.

Invoca WebSocket APIs

Dopo aver distribuito l' WebSocket API, le applicazioni client possono connettersi ad essa e inviarle messaggi, mentre il servizio di backend può inviare messaggi alle applicazioni client connesse:

- Puoi utilizzarla `wscat` per connetterti alla tua WebSocket API e inviarle messaggi per simulare il comportamento del client. Per informazioni, consulta the section called “Utilizzalo `wscat` per connetterti a un' WebSocket API e inviarle messaggi”.
- Puoi utilizzare l'API @connections dal servizio di back-end per inviare un messaggio di callback a un client connesso, ottenere informazioni di connessione o scollegare il client. Per informazioni, consulta the section called “Utilizzo di comandi `@connections` nel servizio di back-end”.
- Un'applicazione client può utilizzare la propria WebSocket libreria per richiamare l' WebSocketAPI.

Utilizzalo **`wscat`** per connetterti a un' WebSocket API e inviarle messaggi

L'`wscat`utilità è uno strumento utile per testare un' WebSocket API creata e distribuita in API Gateway. Puoi installare e utilizzare `wscat` nel modo seguente:

1. Scarica `wscat` da https://www.npmjs.com/package/wscat.
2. Installa `wscat` eseguendo i comandi seguenti.

```
npm install -g wscat
```

3. Per connetterti all'API, esegui il comando `wscat` come mostrato nell'esempio seguente. Questo esempio presuppone che l'impostazione `Authorization` sia `NONE`.

```
wscat -c wss://aabbccddee.execute-api.us-east-1.amazonaws.com/test/
```

   Sarà necessario sostituire *`aabbccddee`* con l'ID API effettivo, visualizzato nella console API Gateway o restituito dal comando `create-api` della AWS CLI .

   Inoltre, se l'API si trova in una regione diversa da `us-east-1`, sarà necessario sostituire la regione corretta.

4. Per testare l'API, immetti un messaggio, ad esempio il seguente mentre sei connesso:

```
{"{jsonpath-expression}":"{route-key}"}
```

   dove *`{jsonpath-expression}`* è un' JSONPathespressione ed *`{route-key}`* è una chiave di percorso per l'API. Esempio:

```
{"action":"action1"}
{"message":"test response body"}
```

   Per ulteriori informazioni su JSONPath, vedere JSONPathor JSONPath for Java.

5. Per disconnetterti dall'API, immetti `ctrl-C`.

Utilizzo di comandi **`@connections`** nel servizio di back-end

Il tuo servizio di backend può utilizzare le seguenti richieste HTTP di WebSocket connessione per inviare un messaggio di callback a un client connesso, ottenere informazioni sulla connessione o disconnettere il client.

> **Important**
>
> Queste richieste utilizzano l'autorizzazione IAM, pertanto occorre firmarle con Signature Version 4 (SigV4). Per eseguire questa operazione, puoi utilizzare l'API di gestione di API Gateway. Per ulteriori informazioni, consulta ApiGatewayManagementApi.

Nel comando seguente, è necessario sostituirlo *{api-id}* con l'ID API effettivo, che viene visualizzato nella console API Gateway o restituito dal comando AWS CLI create-api. È necessario stabilire la connessione prima di utilizzare questo comando.

Per inviare un messaggio di callback al client, utilizza:

```
POST https://{api-id}.execute-api.us-east-1.amazonaws.com/{stage}/
@connections/{connection_id}
```

Puoi testare questa richiesta utilizzando `Postman` o chiamando `awscurl` come nell'esempio seguente:

```
awscurl --service execute-api -X POST -d "hello world" https://{prefix}.execute-api.us-
east-1.amazonaws.com/{stage}/@connections/{connection_id}
```

Il comando deve essere codificato tramite URL come nell'esempio seguente:

```
awscurl --service execute-api -X POST -d "hello world" https://aabbccddee.execute-
api.us-east-1.amazonaws.com/prod/%40connections/R0oXAdfD0kwCH6w%3D
```

Per ottenere l'ultimo stato di connessione del client, utilizza:

```
GET https://{api-id}.execute-api.us-east-1.amazonaws.com/{stage}/
@connections/{connection_id}
```

Per scollegare il client, utilizza:

```
DELETE https://{api-id}.execute-api.us-east-1.amazonaws.com/{stage}/
@connections/{connection_id}
```

Puoi creare dinamicamente un URL di callback utilizzando le variabili `$context` nell'integrazione. Ad esempio, se si utilizza un'integrazione proxy Lambda con una funzione Lambda `Node.js`, puoi creare l'URL e inviare un messaggio a un client connesso come segue:

```
import {
  ApiGatewayManagementApiClient,
  PostToConnectionCommand,
} from "@aws-sdk/client-apigatewaymanagementapi";

export const handler = async (event) => {
  const domain = event.requestContext.domainName;
  const stage = event.requestContext.stage;
  const connectionId = event.requestContext.connectionId;
  const callbackUrl = `https://${domain}/${stage}`;
  const client = new ApiGatewayManagementApiClient({ endpoint: callbackUrl });

  const requestParams = {
    ConnectionId: connectionId,
    Data: "Hello!",
  };

  const command = new PostToConnectionCommand(requestParams);

  try {
    await client.send(command);
  } catch (error) {
    console.log(error);
  }

  return {
    statusCode: 200,
  };
};
```

Se utilizzi un nome di dominio personalizzato per l' WebSocket API, rimuovi la `stage` variabile dal codice della funzione.

Quando si invia un messaggio di callback, la funzione Lambda deve disporre dell'autorizzazione per chiamare l'API di gestione di Gateway API. Se pubblichi un messaggio prima che venga stabilita la connessione o dopo la disconnessione del client, potresti ricevere un errore contenente `GoneException`.

Pubblicazione di API WebSocket che i clienti possono invocare

La semplice creazione e sviluppo di un'API di API Gateway non la rende automaticamente richiamabile dagli utenti. Per renderla richiamabile, è necessario distribuire l'API in una fase. Inoltre, potrebbe essere necessario personalizzare l'URL che verrà utilizzato dagli utenti per accedere all'API. Puoi assegnarli un dominio che sia coerente con il tuo marchio o che sia più facile da ricordare dell'URL predefinito dell'API.

In questa sezione viene descritto come distribuire l'API e personalizzare l'URL fornito agli utenti per accedervi.

> **Note**
>
> Per aumentare la sicurezza delle API API Gateway, il dominio `execute-api.{region}.amazonaws.com` è registrato nella Public Suffix List (PSL). Per una maggiore sicurezza, consigliamo di utilizzare i cookie con un prefisso `__Host-` se hai bisogno di impostare cookie sensibili nel nome di dominio predefinito per le API API Gateway. Questa pratica ti aiuterà a difendere il tuo dominio dai tentativi CSRF (cross-site request forgery). Per ulteriori informazioni, consulta la pagina Impostazione cookie nella pagina Mozilla Developer Network.

Argomenti

- Crea fasi per WebSocket APIs API Gateway
- Implementazione WebSocket APIs in API Gateway
- Politica di sicurezza per WebSocket APIs API Gateway
- Nomi di dominio personalizzati per API WebSocket in Gateway API

Crea fasi per WebSocket APIs API Gateway

Una fase API è un riferimento logico a uno stato del ciclo di vita dell'API (ad esempio, `dev`, `prod`, `beta` o `v2`). Le fasi API sono identificate dal rispettivo ID API e dal nome della fase e sono incluse nell'URL utilizzato per richiamare l'API. Ogni fase è un riferimento con nome a una distribuzione dell'API e viene resa disponibile per le applicazioni client da chiamare.

Una distribuzione è uno snapshot della configurazione dell'API. Dopo essere stata distribuita a una fase, l'API è disponibile per i client da richiamare. È necessario distribuire un'API per attivare le modifiche apportate.

Variabili di fase

Le variabili di fase sono coppie chiave-valore che puoi definire per una fase di un' WebSocket API. Fungono da variabili di ambiente e possono essere utilizzate nella configurazione dell'API.

Ad esempio, puoi definire una variabile di fase e quindi impostare il suo valore come un endpoint HTTP per un'integrazione proxy HTTP. Successivamente, puoi fare riferimento all'endpoint utilizzando il nome della variabile di fase associata. In questo modo, puoi utilizzare la stessa configurazione API con un endpoint diverso in ogni fase. Allo stesso modo, puoi utilizzare le variabili di fase per specificare un'integrazione di AWS Lambda funzioni diversa per ogni fase dell'API.

> **Note**
>
> Le variabili di fase non sono destinate ad essere utilizzate per i dati sensibili, come le credenziali. Per trasferire dati sensibili alle integrazioni, usa un AWS Lambda autorizzatore. È possibile passare dati sensibili alle integrazioni nell'output del provider di autorizzazioni Lambda. Per ulteriori informazioni, consulta the section called “Formato della risposta dell'autorizzazione”.

Esempi

Per utilizzare una variabile di fase per personalizzare l'endpoint di integrazione HTTP, è necessario innanzitutto impostare il nome e il valore della variabile di fase (ad esempio `url`) con un valore pari a `example.com`. Quindi, impostare un'integrazione proxy HTTP. Anziché inserire l'URL dell'endpoint, è possibile comunicare ad API Gateway di usare il valore della variabile di fase, ossi, **`http://${stageVariables.url}`**. Questo valore indica ad API Gateway di sostituire la variabile di fase `${}` al runtime, a seconda della fase dell'API.

È possibile fare riferimento alle variabili di fase in modo simile per specificare il nome di una funzione Lambda o un ruolo AWS ARN.

Quando si specifica un nome di funzione Lambda come valore della variabile di fase, è necessario configurare manualmente le autorizzazioni per la funzione Lambda. Il comando add-permission seguente aggiunge le autorizzazioni richieste:

```
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
```

Riferimento alle variabili di fase di API Gateway

Integrazione HTTP URIs

Puoi utilizzare una variabile di fase come parte di un URI di integrazione HTTP, come mostrato negli esempi seguenti.

- Un URI completo senza protocoll – `http://${stageVariables.<variable_name>}`
- Un dominio completo – `http://${stageVariables.<variable_name>}/resource/operation`
- Un sottodominio – `http://${stageVariables.<variable_name>}.example.com/resource/operation`
- Un percorso – `http://example.com/${stageVariables.<variable_name>}/bar`
- Una stringa di query – `http://example.com/foo?q=${stageVariables.<variable_name>}`

Funzioni Lambda

Puoi utilizzare una variabile di fase al posto di un nome di funzione o alias Lambda, come illustrato negli esempi seguenti.

- `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**
>
> Per utilizzare una variabile di fase per una funzione Lambda, la funzione deve essere nello stesso account dell'API. Le variabili di fase non supportano le funzioni Lambda tra più account.

AWS credenziali di integrazione

È possibile utilizzare una variabile stage come parte di un ARN di credenziali AWS utente o di ruolo, come illustrato nell'esempio seguente.

- `arn:aws:iam::<account_id>:${stageVariables.<variable_name>}`

Implementazione WebSocket APIs in API Gateway

Dopo aver creato l' WebSocket API, devi distribuirla per renderla disponibile per essere richiamata dagli utenti.

Per distribuire un'API, crea una distribuzione API e associala a una fase. Ogni fase è una snapshot dell'API ed è resa disponibile per essere chiamata dalle app client.

> **Important**
>
> Ogni volta che aggiorni un'API, devi ripeterne l'implementazione. Le modifiche a qualcosa di diverso dalle impostazioni di fase richiedono una nuova implementazione, incluse le modifiche alle seguenti risorse:
>
> - Percorsi
> - Integrazioni
> - Autorizzatori
>
> C'è un limite predefinito di 10 fasi per API. Si consiglia di riutilizzare le fasi per le implementazioni.

Per chiamare un' WebSocket API distribuita, il client invia un messaggio all'URL dell'API. L'URL viene determinato dal nome host e dal nome fase dell'API.

> **Note**
>
> API Gateway supporta payload fino a 128 KB con dimensione del frame massima di 32 KB. Se un messaggio supera i 32 KB, deve essere suddiviso in più frame, ciascuno di 32 KB o più piccolo.

Utilizzando il nome di dominio predefinito dell'API, l'URL di (ad esempio) un' WebSocket API in una determinata fase (*`{stageName}`*) ha il seguente formato:

```
wss://{api-id}.execute-api.{region}.amazonaws.com/{stageName}
```

Per rendere l'URL dell' WebSocket API più intuitivo, puoi creare un nome di dominio personalizzato (ad esempio,`api.example.com`) per sostituire il nome host predefinito dell'API. Il processo di configurazione è lo stesso di REST APIs. Per ulteriori informazioni, consulta the section called “Nomi di dominio personalizzati”.

Le fasi permettono un efficace controllo delle versioni dell'API. Ad esempio, puoi distribuire un'API in una fase `test` e una fase `prod` e utilizzare la fase `test` come build di test e la fase `prod` come build stabile. Dopo che gli aggiornamenti hanno superato il test, puoi promuovere la fase `test` alla fase `prod`. La promozione può essere eseguita ridistribuendo l'API nella fase `prod`. Per ulteriori dettagli sulle fasi, consulta the section called “Configurare una fase”.

Argomenti

- Crea una distribuzione WebSocket API utilizzando il AWS CLI
- Creare una distribuzione WebSocket API utilizzando la console API Gateway

Crea una distribuzione WebSocket API utilizzando il AWS CLI

Il seguente comando create-deployment crea l'implementazione:

```
aws apigatewayv2 --region us-east-1 create-deployment --api-id aabbccddee
```

L'output sarà simile al seguente:

```
{
    "DeploymentId": "fedcba",
```

```
    "DeploymentStatus": "DEPLOYED",
    "CreatedDate": "2018-11-15T06:49:09Z"
}
```

L'API distribuita non può essere chiamata fino a quando non si associa la distribuzione a una fase. Puoi creare una nuova fase o riutilizzare una fase creata in precedenza.

Il comando create-stage seguente crea una nuova fase e la associa all'implementazione:

```
aws apigatewayv2 --region us-east-1 create-stage --api-id aabbccddee --deployment-id
 fedcba --stage-name test
```

L'output sarà simile al seguente:

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

Puoi anche riutilizzare una fase esistente aggiornando la `deploymentId` proprietà della fase con l'ID di distribuzione appena creato (*`deployment-id`*). Il comando update-stage seguente aggiorna l'ID di implementazione della fase:

```
aws apigatewayv2 update-stage --region region \
    --api-id api-id \
    --stage-name stage-name \
    --deployment-id deployment-id
```

Creare una distribuzione WebSocket API utilizzando la console API Gateway

Per utilizzare la console API Gateway per creare una distribuzione per un' WebSocket API:

1. Accedere alla console API Gateway e scegliere l'API.
2. Seleziona Deploy API (Distribuisci API).
3. Scegliere la fase desiderata dall'elenco a discesa oppure immettere il nome di una nuova fase.

Politica di sicurezza per WebSocket APIs API Gateway

API Gateway applica una politica di sicurezza `TLS_1_2` per tutti gli endpoint WebSocket API.

Una policy di sicurezza è una combinazione predefinita di una versione TLS minima e un pacchetto di crittografia offerta da Gateway Amazon API. Il protocollo TLS affronta i problemi di sicurezza della rete, ad esempio manomissioni e intercettazioni tra un client e un server. Quando i client stabiliscono un handshake TLS sull'API tramite il dominio personalizzato, la policy di sicurezza applica la versione di TLS e le opzioni del pacchetto di crittografia che i client possono scegliere di utilizzare. Questa policy di sicurezza accetta il traffico TLS 1.2 e TLS 1.3 e rifiuta il traffico TLS 1.0.

Protocolli e cifrari TLS supportati per WebSocket APIs

La tabella seguente descrive i protocolli TLS supportati per. WebSocket APIs

Protocolli TLS	Policy di sicurezza TLS_1_2
TLSv13.	Sì
TLSv12.	Sì

La tabella seguente descrive i codici TLS disponibili per la politica di sicurezza TLS 1_2 per. WebSocket APIs

Crittografie TLS	Policy di sicurezza TLS_1_2
TLS_AES_128_GCM_ SHA256	Sì
TLS_AES_256_GCM_ SHA384	Sì
TLS_ _ CHACHA20 POLY1305 SHA256	Sì

Crittografie TLS	Policy di sicurezza TLS_1_2
ECDHE-ECDSA- -GCM- AES128 SHA256	Sì
ECDH-RSA- AES128 -GCM- SHA256	Sì
ECDHE-ECSA AES128 - - SHA256	Sì
ECDHE-RSA- - AES128 SHA256	Sì
ECDHE-ECDSA- -GCM AES256 - SHA384	Sì
ECDH-RSA- AES256 -GCM- SHA384	Sì
ECDHE-ECSA AES256 - - SHA384	Sì
ECDHE-RSA- - AES256 SHA384	Sì
AES128-GCM- SHA256	Sì
AES128-SHA256	Sì
AES256-GCM- SHA384	Sì
AES256-SHA256	Sì

OpenSSL e nomi crittografia RFC

OpenSSL e IETF RFC 5246 utilizzano nomi diversi per le stesse crittografie. Per l'elenco dei nomi delle crittografie, consulta the section called “OpenSSL e nomi crittografia RFC”.

Informazioni su REST APIs e HTTP APIs

Per ulteriori informazioni su REST APIs e HTTP APIs, vedere the section called “Scelta di una policy di sicurezza” ethe section called “Politica di sicurezza per HTTP APIs in API Gateway”.

Nomi di dominio personalizzati per API WebSocket in Gateway API

I nomi di dominio personalizzati sono URL più semplici e più intuitivi che è possibile fornire agli utenti delle API.

Dopo aver distribuito un'API, tu e i tuoi clienti potete richiamarla usando l'URL di base predefinito nel formato seguente:

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

dove *api-id* è generato da Gateway API, *region* è la Regione AWS e *stage* è il valore che è stato specificato al momento dell'implementazione dell'API.

La parte del nome host dell'URL, *api-id*`.execute-api.`*region*`.amazonaws.com`, fa riferimento a un endpoint API. Il nome dell'endpoint API predefinito viene generato casualmente, è complesso da richiamare e non è intuitivo.

Con i nomi di dominio personalizzati, è possibile impostare il nome host dell'API e scegliere un percorso base (ad esempio `myservice`) per mappare l'URL alternativo all'API. Ad esempio, un URL di base dell'API più intuitivo può diventare:

```
https://api.example.com/myservice
```

Considerazioni

Le seguenti considerazioni potrebbero influire sull'utilizzo di un nome di dominio personalizzato.

- Se un nome di dominio personalizzato viene mappato a un'API WebSocket, non può essere mappato a una REST API o a un'API HTTP.
- Sono supportati solo i nomi di dominio personalizzati regionali.
- L'unica versione minima di TLS supportata è TLS 1.2.
- È necessario creare o aggiornare il record di risorse del provider DNS per eseguire la mappatura all'endpoint API. In assenza di questa mappatura, le richieste API destinate al nome di dominio personalizzato non possono raggiungere API Gateway.
- È possibile supportare un numero pressoché infinito di nomi di dominio senza superare la quota predefinita con un certificato jolly. Per ulteriori informazioni, consulta the section called “Nomi di dominio personalizzati con caratteri jolly”.

Prerequisiti

Di seguito sono riportati i prerequisiti per un nome di dominio personalizzato.

Registrare un nome di dominio

Per configurare nomi di dominio personalizzati per le API, devi avere un nome di dominio Internet registrato. Puoi registrare un nome di dominio Internet utilizzando Amazon Route 53 oppure un registrar di dominio di terze parti a tua scelta. Il nome di dominio personalizzato può essere il nome di un sottodominio o del dominio root (detto anche "apex di zona") di un dominio Internet registrato.

Il nome di dominio deve seguire la specifica RFC 1035 e può avere un massimo di 63 ottetti per etichetta e 255 ottetti in totale.

Certificati per nomi di dominio personalizzati

Prima di configurare un nome di dominio personalizzato per un'API, è necessario che sia già presente un certificato SSL/TLS in ACM. Se ACM non è disponibile nella Regione AWS in cui si sta creando il nome di dominio personalizzato, è necessario importare nella Regione un certificato in Gateway API.

Per importare un certificato SSL/TLS, devi fornire il corpo del certificato SSL/TLS in formato PEM, la sua chiave privata e la catena di certificati per il nome di dominio personalizzato.

Ogni certificato archiviato in ACM è identificato dal relativo ARN. Con i certificati emessi da ACM, non devi preoccuparti di esporre dettagli sensibili del certificato, come la chiave privata. Per usare un certificato gestito da AWS per un nome di dominio, devi semplicemente fare riferimento al relativo ARN.

Se l'applicazione utilizza il blocco dei certificati, talvolta denominato associazione SSL, per associare un certificato ACM, è possibile che l'applicazione non riesca a connettersi al tuo dominio dopo che AWS ha rinnovato il certificato. Per ulteriori informazioni, consulta Probemi nel blocco dei certificati nella Guida per l'utente di AWS Certificate Manager.

Nomi di dominio personalizzati con caratteri jolly

Con i nomi di dominio personalizzati con caratteri jolly, è possibile supportare un numero quasi infinito di nomi di dominio senza superare la quota predefinita. Ad esempio, è possibile dare a ciascuno dei clienti il proprio nome di dominio, *`customername`*`.api.example.com`.

Per creare un nome di dominio personalizzato con caratteri jolly, specificare un carattere jolly (`*`) come primo sottodominio di un dominio personalizzato che rappresenta tutti i possibili sottodomini di un dominio radice.

Ad esempio, il nome di dominio personalizzato con caratteri jolly `*.example.com` genera sottodomini quali `a.example.com`, `b.example.com` e `c.example.com`, indirizzati tutti allo stesso dominio.

I nomi di dominio personalizzati con caratteri jolly supportano configurazioni distinte dai nomi di dominio personalizzati standard di API Gateway. Ad esempio, in un singolo account AWS, è possibile configurare `*.example.com` e `a.example.com` affinché si comportino in modo diverso.

È possibile utilizzare le variabili di contesto `$context.domainName` e `$context.domainPrefix` per determinare il nome di dominio utilizzato da un client per chiamare l'API. Per ulteriori informazioni sulle variabili di contesto, consulta Variabili per le trasformazioni dei dati per Gateway API.

Per creare un nome di dominio personalizzato con caratteri jolly, è necessario fornire un certificato emesso da ACM che sia stato convalidato utilizzando il DNS o il metodo di convalida della posta elettronica.

> **Note**
>
> Non è possibile creare un nome di dominio personalizzato con caratteri jolly se un account AWS diverso ha creato un nome di dominio personalizzato in conflitto con il nome di dominio personalizzato con caratteri jolly. Ad esempio, se l'account A ha creato `a.example.com`, l'account B non può creare il nome di dominio personalizzato con caratteri jolly `*.example.com`.
> Se l'account A e l'account B condividono un proprietario, è possibile contattare il Centro assistenza AWS per richiedere un'eccezione.

Passaggi successivi per nomi di dominio personalizzati

Per configurare un nome di dominio personalizzato per un'API HTTP, consulta la documentazione disponibile nella sezione REST API della Guida per gli sviluppatori di Gateway API.

Per prima cosa, si specifica un certificato per il nome di dominio personalizzato. Per ulteriori informazioni, consulta the section called “Prepara i certificati in AWS Certificate Manager”. Successivamente, si crea un nome di dominio personalizzato regionale. Per ulteriori informazioni, consulta the section called “Configurazione di un nome di dominio personalizzato regionale”.

Mappatura delle fasi API a un nome di dominio personalizzato per le API WebSocket

È possibile utilizzare le mappature API per connettere le fasi API a un nome di dominio personalizzato. Dopo aver creato un nome di dominio e aver configurato i record DNS, è possibile utilizzare le mappature API per inviare il traffico alle API tramite il nome di dominio personalizzato.

Una mappatura API specifica un'API, una fase e, facoltativamente, un percorso da utilizzare per la mappatura. Ad esempio, è possibile mappare la fase `production` di un'API su `wss://api.example.com/orders`.

Prima di creare una mappatura API, è necessario disporre di un'API, di una fase e di un nome di dominio personalizzato. Per ulteriori informazioni sulla creazione di un nome di dominio personalizzato, consulta the section called “Configurazione di un nome di dominio personalizzato regionale”.

Restrizioni

- In una mappatura API, il nome di dominio personalizzato e le API mappate devono trovarsi nello stesso account AWS.
- Le mappature API devono contenere solo lettere, numeri e i seguenti caratteri: `$-_.+!*'()`.
- La lunghezza massima per il percorso in una mappatura API è di 300 caratteri.
- Non è possibile mappare le API WebSocket allo stesso nome di dominio personalizzato di un'API HTTP o un'API REST.
- Se si crea una mappatura API con più livelli, Gateway API converte tutti i nomi di intestazione in lettere minuscole.

Creare una mappatura API

Per creare una mappatura API, innanzitutto è necessario creare un nome di dominio personalizzato, un'API e una fase. Per informazioni sulla creazione di un nome di dominio personalizzato, consulta the section called “Configurazione di un nome di dominio personalizzato regionale”.

Console di gestione AWS

Per creare una mappatura API

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere Nomi di dominio personalizzati.

3. Selezionare un nome di dominio personalizzato già creato.
4. Scegliere API mappings (mappature API).
5. Scegliere Configure API mappings (Configura mappature API).
6. Scegliere Add new mapping (Aggiungi nuova mappatura).
7. Immettere un'API, uno Stage (Fase)e, facoltativamente, un Path (Percorso).
8. Selezionare Salva.

AWS CLI

Il comando create-api-mapping seguente crea una mappatura API. In questo esempio, API Gateway invia le richieste `api.example.com/v1` all'API e alla fase specificate.

```
aws apigatewayv2 create-api-mapping \
    --domain-name api.example.com \
    --api-mapping-key v1 \
    --api-id a1b2c3d4 \
    --stage test
```

CloudFormation

Il seguente esempio di CloudFormation consente di creare una mappatura API.

```
MyApiMapping:
  Type: 'AWS::ApiGatewayV2::ApiMapping'
  Properties:
    DomainName: api.example.com
    ApiMappingKey: 'v1'
    ApiId: !Ref MyApi
    Stage: !Ref MyStage
```

Tipi di indirizzo IP per nomi di dominio personalizzati per API WebSocket

Quando si crea un nome di dominio personalizzato, è possibile specificare il tipo di indirizzo IP che può invocare il dominio. È possibile scegliere IPv4 per risolvere gli indirizzi IPv4 per invocare il dominio oppure dualstack per consentire a entrambi gli indirizzi IPv4 e IPv6 di invocare il dominio. È consigliabile impostare il tipo di indirizzo IP su dualstack per ridurre l'esaurimento dello spazio IP

o per il livello di sicurezza. Per ulteriori informazioni sui vantaggi del tipo di indirizzo IP dualstack, consultare IPv6 on AWS.

Considerazioni sui tipi di indirizzo IP

Le seguenti considerazioni potrebbero influire sull'utilizzo dei tipi di indirizzo IP.

- Il tipo di indirizzo IP predefinito per i nomi di dominio personalizzati di Gateway API è IPv4.
- Per il nome di dominio personalizzato, non è necessario che tutte le API mappate ad esso abbiano lo stesso tipo di indirizzo IP. Se si disabilita l'endpoint API predefinito, si potrebbe influire sul modo in cui i chiamanti possono invocare l'API.

Modifica del tipo di indirizzo IP del nome di dominio personalizzato

È possibile modificare il tipo di indirizzo IP aggiornando la configurazione dell'endpoint del nome di dominio. È possibile aggiornare la configurazione dell'endpoint utilizzando la Console di gestione AWS, AWS CLI, CloudFormation o un AWS SDK.

Console di gestione AWS

Per modificare il tipo di indirizzo IP di un nome di dominio personalizzato

1. Accedere alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegliere un nome di dominio personalizzato pubblico.
3. Scegliere Configurazione endpoint.
4. Per Tipo di indirizzo IP, selezionare IPv4 o Dualstack.
5. Selezionare Salva.

AWS CLI

Il comando update-domain-name seguente aggiorna un'API in modo che abbia un tipo di indirizzo IP dualstack:

```
aws apigatewayv2 update-domain-name \
   --domain-name dualstack.example.com \
   --domain-name-configurations CertificateArn=arn:aws:acm:us-
east-1:111122223333:certificate/abcd1234-5678-abc,IpAddressType=dualstack
```

L'output sarà simile al seguente:

```
{
    "ApiMappingSelectionExpression": "$request.basepath",
    "DomainName": "dualstack.example.com",
    "DomainNameConfigurations": [
        {
            "ApiGatewayDomainName": "d-abcd1234.execute-api.us-
east-1.amazonaws.com",
            "CertificateArn": "arn:aws:acm:us-east-1:111122223333:certificate/
abcd1234-5678-abc",
            "DomainNameStatus": "AVAILABLE",
            "EndpointType": "REGIONAL",
            "HostedZoneId": "Z3LQWSYCGH4ADY",
            "SecurityPolicy": "TLS_1_2",
            "IpAddressType": "dualstack"
        }
    ],
    "Tags": {}
}
```

Disabilita l'endpoint predefinito per WebSocket APIs

Per impostazione predefinita, i client possono richiamare l'API utilizzando l'endpoint `execute-api` generato da API Gateway per l'API. Per garantire che i client possano accedere all'API solo utilizzando un nome di dominio personalizzato con l'autenticazione TLS reciproca, disattivare l'endpoint `execute-api` predefinito. Quando si disattiva l'endpoint predefinito, questa operazione influisce su tutte le fasi di un'API.

La procedura seguente mostra come disabilitare l'endpoint predefinito per un' WebSocket API.

Console di gestione AWS

1. Accedi alla console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway.
2. Scegli un'API. WebSocket
3. Scegli Impostazioni API.
4. In Dettagli API seleziona Modifica.
5. Per Endpoint predefinito seleziona Inattivo.
6. Scegli Save changes (Salva modifiche).

7. Nel pannello di navigazione principale scegli Instradamenti.
8. Scegli Distribuzione, quindi implementa nuovamente l'API o crea una nuova fase per rendere effettiva la modifica.

AWS CLI

Il seguente comando update-api disabilita l'endpoint predefinito per un'API: WebSocket

```
aws apigatewayv2 update-api \
    --api-id abcdef123 \
    --disable-execute-api-endpoint
```

Dopo aver disabilitato l'endpoint predefinito, è necessario distribuire l'API per rendere effettiva la modifica.

Il AWS CLI comando seguente crea una distribuzione.

```
aws apigatewayv2 create-deployment \
    --api-id abcdef123 \
    --stage-name dev
```

Proteggi il tuo WebSocket APIs API Gateway

API Gateway fornisce diversi modi per proteggere l'API da determinate minacce, ad esempio utenti malintenzionati o picchi di traffico. È possibile proteggere l'API usando strategie come la generazione di certificati SSL o l'impostazione degli obiettivi della limitazione (della larghezza di banda della rete). Per ulteriori informazioni sulla generazione dei certificati SSL, consultare the section called "Certificati API Gateway-generated ". Gli obiettivi della limitazione (della larghezza di banda della rete) sono descritti in dettaglio nella parte restante di questa sezione.

Puoi configurare la limitazione per le tue per APIs evitare che vengano sopraffatte da troppe richieste. Le limitazioni della larghezza di banda della rete sono applicate sulla base del massimo sforzo e dovrebbero essere considerate come obiettivi e non limiti massimi garantiti delle richieste.

API Gateway limita la larghezza di banda della rete delle richieste nell'API mediante l'algoritmo di token bucket, dove un token rappresenta una richiesta. In particolare, API Gateway esamina la frequenza e il numero di richieste inviate rispetto APIs a tutti gli utenti del tuo account, per regione.

Nell'algoritmo di bucket token, un picco può consentire il superamento predefinito di tali limiti, ma anche altri fattori possono causare il superamento dei limiti in alcuni casi.

Quando le richieste inviate superano i limiti impostati per il tasso a regime e per i limiti di picco, API Gateway inizia a limitare le richieste. I clienti potrebbero ricevere risposte agli errori `429 Too Many Requests` a questo punto. Quando rileva queste eccezioni, il client può reinviare le richieste non riuscite in modo da limitare la velocità.

In qualità di sviluppatore di API, puoi impostare i limiti target per le singole fasi o percorsi dell'API per migliorare le prestazioni complessive di tutto APIs il tuo account.

Limitazione a livello di account per regione

Per impostazione predefinita, API Gateway limita le richieste allo stato stazionario (RPS) APIs all'interno di un AWS account, per regione. Inoltre, limita il burst (ovvero la dimensione massima del bucket) per tutti gli elementi di un account, per regione APIs . AWS In API Gateway, il limite dei picchi rappresenta il numero massimo di invii di richieste che API Gateway può gestire prima di restituire risposte di errore `429 Too Many Requests`. Per ulteriori informazioni sulla limitazione delle quote, vedere Quote e note importanti.

I limiti per account vengono applicati a tutti i membri di un account APIs in una regione specificata. Il limite di tariffa a livello di account può essere aumentato su richiesta: sono possibili limiti più elevati con APIs timeout più brevi e carichi utili inferiori. Per richiedere un aumento dei limiti di limitazione a livello di account per Regione, contatta il Centro assistenza AWS. Per ulteriori informazioni, consulta Quote e note importanti. Tieni presente che questi limiti non possono essere superiori ai limiti di limitazione. AWS

Throttling a livello di instradamento

Puoi impostare il throttling a livello di route per sostituire i limiti di throttling delle richieste a livello di account per una fase specifica o per singoli route nell'API. I limiti della limitazione (della larghezza di banda della rete) della route di default non possono superare i limiti di velocità a livello di account.

È possibile configurare la limitazione a livello di percorso utilizzando AWS CLI. Il comando update-stage seguente configura la limitazione (della larghezza di banda della rete) personalizzata per la fase e l'instradamento specificati di un'API:

```
aws apigatewayv2 update-stage \
    --api-id a1b2c3d4 \
    --stage-name dev \
```

```
    --route-settings '{"messages":
{"ThrottlingBurstLimit":100,"ThrottlingRateLimit":2000}}'
```

Monitoraggio delle API WebSocket in Gateway API

Puoi utilizzare i parametri CloudWatch e CloudWatch Logs per monitorare le API WebSocket. Combinando log e parametri, puoi registrare gli errori e monitorare le prestazioni dell'API.

> **Note**
>
> L'API Gateway potrebbe non generare log e parametri nei seguenti casi:
>
> - Errori 413 per richiesta troppo grande
> - Errori 429 per troppe richieste
> - Errori serie 400 per richieste inviate a un dominio personalizzato che non dispone del mapping API
> - Errori serie 500 per malfunzionamenti interni

Argomenti

- Monitora l'esecuzione delle WebSocket API con CloudWatch metriche
- Configurazione della registrazione per API WebSocket APIs Gateway

Monitora l'esecuzione delle WebSocket API con CloudWatch metriche

Puoi utilizzare i CloudWatch parametri di Amazon per monitorare WebSocket APIs. La configurazione è simile a quella utilizzata per REST APIs. Per ulteriori informazioni, consulta Monitora l'esecuzione delle API REST con i CloudWatch parametri di Amazon.

Le seguenti metriche sono supportate per WebSocket APIs:

Metrica	Description
ConnectCount	Il numero di messaggi inviati all'integrazione route `$connect`.

Metrica	Description
MessageCount	Il numero di messaggi inviati all' WebSocket API, da o verso il client.
IntegrationError	Il numero di richieste che restituiscono una risposta 4XX/5XX dall'integrazione.
ClientError	Il numero di richieste con una risposta 4XX restituita da API Gateway prima che l'integra zione venga richiamata.
ExecutionError	Errori che si sono verificati durante la chiamata all'integ razione.
IntegrationLatency	La differenza di tempo tra API Gateway che invia la richiesta all'integrazione e API Gateway che riceve la risposta dall'inte grazione. Soppresso per callback e integrazioni fittizie.

Per filtrare i parametri di API Gateway, puoi utilizzare le dimensioni nella seguente tabella.

Dimensione	Description
ApiId	Filtra i parametri API Gateway per un'API con l'ID API specificato.
ApiId, Fase	Filtra i parametri API Gateway per una fase API con l'ID API e l'ID fase specificati.

Dimensione	Description
ApiId, Metodo, Risorsa, Fase	Filtra le metriche di Gateway API per un metodo API con l'ID API, l'ID fase, il percorso della risorsa e l'ID instradam ento specificati. API Gateway non invierà queste metriche a meno che tu non abbia abilitato esplicitamente le metriche dettagliate CloudWatch . È possibile farlo richiamando l'UpdateStageazione dell'API REST API Gateway V2 a cui aggiornare la `detailedM etricsEnabled` proprietà . `true` In alternativa, puoi chiamare il AWS CLI comando update-stage per aggiornare la `DetailedMetricsEna bled` proprietà a. `true` L'abilitazione di tali parametri comporta addebiti aggiuntivi sul tuo account. Per informazi oni sui prezzi, consulta la pagina CloudWatchdei prezzi di Amazon.

Configurazione della registrazione per API WebSocket APIs Gateway

È possibile abilitare la registrazione per scrivere i log nei registri. CloudWatch Esistono due tipi di accesso tramite API CloudWatch: registrazione dell'esecuzione e registrazione degli accessi. Nella registrazione dell'esecuzione, API Gateway gestisce i CloudWatch log. Il processo include

la creazione di gruppi e flussi di log e la segnalazione ai flussi di log delle richieste e delle risposte dell'intermediario.

Per migliorare il livello di sicurezza, è consigliabile utilizzare la registrazione nei log dell'esecuzione a livello di `ERROR` o `INFO`. Potrebbe essere necessario eseguire questa operazione per rispettare vari framework di conformità. Per ulteriori informazioni, consultare Amazon API Gateway controls nella Guida per l'utente di AWS Security Hub .

Nella registrazione degli accessi, in qualità di sviluppatore dell'API puoi registrare chi ha avuto accesso alla tua API e in che modo l'intermediario ha avuto accesso all'API. Puoi creare un gruppo di log personalizzato o sceglierne uno esistente che potrebbe essere gestito da API Gateway. Per specificare i dettagli di accesso, seleziona variabili `$context` (espresse in un formato a scelta) e scegli un gruppo di log come destinazione.

Per istruzioni su come configurare la CloudWatch registrazione, consulta. the section called "Configura la registrazione degli accessi utilizzando la console API Gateway"

Quando specifichi il Log Format (Formato di log), puoi scegliere quali variabili di contesto registrare. Sono supportate le seguenti variabili.

Parametro	Descrizione
`$context.apiId`	Identificatore assegnato da API Gateway all'API.
`$context.authorize.error`	Il messaggio di errore di autorizzazione.
`$context.authorize.latency`	La latenza di autorizzazione in ms.
`$context.authorize.status`	Il codice di stato restituito da un tentativo di autorizzazione.
`$context.authorizer.error`	Il messaggio di errore restituito da un'autori zzazione.
`$context.authorizer.integra tionLatency`	La latenza del provider di autorizzazioni Lambda in ms.
`$context.authorizer.integra tionStatus`	Il codice di stato restituito da un'autorizzazione Lambda.

Parametro	Descrizione
`$context.authorizer.latency`	La latenza di autorizzazione in ms.
`$context.authorizer.requestId`	L'ID della richiesta dell' AWS endpoint.
`$context.authorizer.status`	Il codice di stato restituito da un'autorizzazione.
`$context.authorizer.principalId`	Identificazione dell'utente principale associata al token inviato dal client e restituita da una funzione Lambda del provider di autorizza zioni di Lambda API Gateway. (Un provider di autorizzazioni Lambda in precedenza noto come autorizzazioni ad hoc).
`$context.authorizer.` *`property`*	Valore in formato stringa della coppia chiave/va lore specificata della mappa `context` restituita da una funzione delle autorizzazioni Lambda di API Gateway. Ad esempio, se le autorizzazioni restituiscono la mappa `context` seguente: `"context" : { "key": "value", "numKey": 1, "boolKey": true }` la chiamata di `$context.authorize r.key` restituisce la stringa `"value"`, la chiamata di `$context.authorize r.numKey` restituisce la stringa `"1"` e la chiamata di `$context.authorize r.boolKey` restituisce la stringa `"true"`.
`$context.authenticate.error`	Il messaggio di errore restituito da un tentativo di autenticazione.

Parametro	Descrizione
`$context.authenticate.latency`	La latenza di autenticazione in ms.
`$context.authenticate.status`	Il codice di stato restituito da un tentativo di autenticazione.
`$context.connectedAt`	L'ora della connessione in formato Epoch.
`$context.connectionId`	Un ID univoco per la connessione, che può essere utilizzato per effettuare un callback al client.
`$context.domainName`	Un nome di dominio per l' WebSocket API. Può essere utilizzato per effettuare un callback al client (invece di un valore codificato).
`$context.error.message`	Una stringa contenente un messaggio di errore API Gateway.
`$context.error.messageString`	Valore `$context.error.message` tra virgolette, ovvero `"$context.error.message"`.
`$context.error.responseType`	Il tipo di risposta di errore.
`$context.error.validationErrorString`	Stringa contenente un messaggio di errore di convalida dettagliato.
`$context.eventType`	Il tipo di evento: `CONNECT`, `MESSAGE` o `DISCONNECT`.
`$context.extendedRequestId`	Equivalente a `$context.requestId`.
`$context.identity.accountId`	L'ID AWS dell'account associato alla richiesta.
`$context.identity.apiKey`	Chiave del proprietario dell'API associata alla richiesta API abilitata dalla chiave.

Parametro	Descrizione
`$context.identity.apiKeyId`	ID chiave API associato alla richiesta API abilitata dalla chiave
`$context.identity.caller`	Identificatore dell'entità principale dell'intermediario che ha firmato la richiesta. Supportato per route che utilizzano l'autorizzazione IAM.
`$context.identity.cognitoAuthenticationProvider`	Un elenco separato da virgole con tutti i provider di autenticazione Amazon Cognito utilizzati dal chiamante che effettua la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito. Ad esempio, per un'identità di un pool di utenti Amazon Cognito, `cognito-idp.`*`region`*`.amazonaws.com/`*`user_pool_id`*`,cognito-idp.`*`region`*`.amazonaws.com/`*`user_pool_id`*`:CognitoSignIn:`*`token subject claim`* Per informazioni sui provider di autenticazione Amazon Cognito disponibili, consulta Using Federated Identities nella Guida per gli sviluppatori di Amazon Cognito.
`$context.identity.cognitoAuthenticationType`	Tipo di autenticazione Amazon Cognito dell'intermediario da cui proviene la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito. I valori possibili includono `authenticated` per le identità autenticate e `unauthenticated` per le identità non autenticate.

Parametro	Descrizione
`$context.identity.cognitoId entityId`	ID identità di Amazon Cognito dell'intermediario da cui proviene la richiesta. Disponibile solo se la richiesta è stata firmata con credenziali Amazon Cognito.
`$context.identity.cognitoId entityPoolId`	ID pool di identità di Amazon Cognito dell'inte rmediario da cui proviene la richiesta. Disponibi le solo se la richiesta è stata firmata con credenziali Amazon Cognito.
`$context.identity.principalOrgId`	L'ID organizzazione AWS. Supportato per route che utilizzano l'autorizzazione IAM.
`$context.identity.sourceIp`	Indirizzo IP di origine della connessione TCP da cui proviene la richiesta ad API Gateway.
`$context.identity.user`	Identificatore dell'entità principale dell'utente che sarà autorizzato per l'accesso alle risorse. Supportato per route che utilizzano l'autoriz zazione IAM.
`$context.identity.userAgent`	Agente utente del chiamante API.
`$context.identity.userArn`	Amazon Resource Name (ARN) dell'utente valido identificato dopo l'autenticazione.
`$context.integration.error`	Il messaggio di errore restituito da un'integr azione.
`$context.integration.integr ationStatus`	Per l'integrazione del proxy Lambda, il codice di stato restituito dal codice della funzione Lambda di backend AWS Lambda, non dal codice della funzione Lambda.
`$context.integration.latency`	Latenza di integrazione in ms. Equivalente a `$context.integrationLatency` .

Parametro	Descrizione
`$context.integration.requestId`	L'ID della AWS richiesta dell'endpoint. Equivalente a `$context.awsEndpointRequestId` .
`$context.integration.status`	Il codice di stato restituito da un'integrazione. Per le integrazioni proxy Lambda, questo è il codice di stato restituito dal codice della funzione Lambda. Equivalente a `$context.integrationStatus` .
`$context.integrationLatency`	La latenza di integrazione in millisecondi, disponibile solo per il log di accesso.
`$context.messageId`	Un ID univoco sul lato server per un messaggio . Disponibile solo quando `$context.eventType` è `MESSAGE`.
`$context.requestId`	Come `$context.extendedRequestId` .
`$context.requestTime`	Ora della richiesta in formato CLF (`dd/MMM/yyyy:HH:mm:ss +-hhmm`).
`$context.requestTimeEpoch`	L'ora della richiesta in formato epoca (Unix epoch) in millisecondi.
`$context.routeKey`	La chiave di instradamento selezionata.
`$context.stage`	Fase di distribuzione della chiamata API, ad esempio, beta o di produzione.
`$context.status`	Lo stato della risposta.
`$context.waf.error`	Il messaggio di errore restituito da AWS WAF.
`$context.waf.latency`	La AWS WAF latenza in ms.
`$context.waf.status`	Il codice di stato restituito da AWS WAF.

Esempi di alcuni formati di log delle operazioni di accesso utilizzati con maggiore frequenza sono mostrati nella console API Gateway ed elencati qui di seguito.

- `CLF` (Common Log Format):

```
$context.identity.sourceIp $context.identity.caller \
$context.identity.user [$context.requestTime] "$context.eventType $context.routeKey
 $context.connectionId" \
$context.status $context.requestId
```

  I caratteri di continuazione (\) sono intesi come un aiuto visivo. Il formato del registro deve essere una singola riga. È possibile aggiungere un carattere di nuova riga (\n) alla fine del formato di registro per includere una nuova riga alla fine di ogni voce di registro.

- `JSON`:

```
{
"requestId":"$context.requestId", \
"ip": "$context.identity.sourceIp", \
"caller":"$context.identity.caller", \
"user":"$context.identity.user", \
"requestTime":"$context.requestTime", \
"eventType":"$context.eventType", \
"routeKey":"$context.routeKey", \
"status":"$context.status", \
"connectionId":"$context.connectionId"
}
```

  I caratteri di continuazione (\) sono intesi come un aiuto visivo. Il formato del registro deve essere una singola riga. È possibile aggiungere un carattere di nuova riga (\n) alla fine del formato di registro per includere una nuova riga alla fine di ogni voce di registro.

- `XML`:

```
<request id="$context.requestId"> \
 <ip>$context.identity.sourceIp</ip> \
 <caller>$context.identity.caller</caller> \
 <user>$context.identity.user</user> \
 <requestTime>$context.requestTime</requestTime> \
 <eventType>$context.eventType</eventType> \
 <routeKey>$context.routeKey</routeKey> \
 <status>$context.status</status> \
```

```
 <connectionId>$context.connectionId</connectionId> \
</request>
```

I caratteri di continuazione (\) sono intesi come un aiuto visivo. Il formato del registro deve essere una singola riga. È possibile aggiungere un carattere di nuova riga (\n) alla fine del formato di registro per includere una nuova riga alla fine di ogni voce di registro.

- CSV (valori separati da virgola):

```
$context.identity.sourceIp,$context.identity.caller, \
$context.identity.user,$context.requestTime,$context.eventType, \
$context.routeKey,$context.connectionId,$context.status, \
$context.requestId
```

I caratteri di continuazione (\) sono intesi come un aiuto visivo. Il formato del registro deve essere una singola riga. È possibile aggiungere un carattere di nuova riga (\n) alla fine del formato di registro per includere una nuova riga alla fine di ogni voce di registro.

Documentazione di riferimento Amazon Resource Name (ARN) API Gateway

Le tabelle seguenti elencano gli Amazon Resource Names (ARNs) per le risorse API Gateway. Per ulteriori informazioni sull'utilizzo ARNs nelle AWS Identity and Access Management politiche, consulta Come funziona Amazon API Gateway con IAM eControllo degli accessi a una REST API con le autorizzazioni IAM.

API HTTP e risorse WebSocket API

Risorsa	ARN
AccessLogSettings	`arn:partition :apigatew ay: region::/apis/api-id/ stages/stage-name /accesslo gsettings`
Api	`arn:partition :apigatew ay: region::/apis/api-id`
API	`arn:partition :apigatew ay: region::/apis`
Authorizer	`arn:partition :apigatew ay: region::/apis/api-id/authoriz ers/ id`
Autorizzatori	`arn:partition :apigatew ay: region::/apis/api-id/authoriz ers`
Cors	`arn:partition :apigatew ay: region::/apis/api-id/cors`

Risorsa	ARN
Implementazione	arn:*partition* :apigateway: *region*::/apis/*api-id*/deployments/ *id*
Distribuzioni	arn:*partition* :apigateway: *region*::/apis/*api-id*/deployments
ExportedAPI	arn:*partition* :apigateway: *region*::/apis/*api-id*/exports/ *specification*
Integrazione	arn:*partition* :apigateway: *region*::/apis/*api-id*/integrations/ *integration-id*
Integrazioni	arn:*partition* :apigateway: *region*::/apis/*api-id*/integrations
IntegrationResponse	arn:*partition* :apigateway: *region*::/apis/*api-id*/integrationresponses/ *integration-response*
IntegrationResponses	arn:*partition* :apigateway: *region*::/apis/*api-id*/integrationresponses
Modello	arn:*partition* :apigateway: *region*::/apis/*api-id*/models/*id*
Modelli	arn:*partition* :apigateway: *region*::/apis/*api-id*/models

Risorsa	ARN
ModelTemplate	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/models/*id*/ template
Route	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/routes/*id*
Percorsi	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*
Stage	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/ stages/*stage-name*
Stage	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/stages
VpcLink	arn:*partition* :apigatew ay: *region*::/vpclinks/ *vpclink-id*

Risorsa	ARN
VpcLinks	arn:*partition* :apigatew ay: *region*::/vpclinks

Risorse API REST

Risorsa	ARN
Account	arn:*partition* :apigatew ay: *region*::/account
ApiKey	arn:*partition* :apigatew ay: *region*::/apikeys/ *id*
ApiKeys	arn:*partition* :apigatew ay: *region*::/apikeys
Authorizer	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ authorizers/ *id*
Autorizzatori	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ authorizers
ClientCertificate	arn:*partition* :apigatew ay: *region*::/clientcertifica tes/ *id*
ClientCertificates	arn:*partition* :apigatew ay: *region*::/clientcertificates
Implementazione	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ deployments/ *id*

Risorsa	ARN
Distribuzioni	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ deployments
DocumentationPart	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ documentation/parts/ *id*
DocumentationParts	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ documentation/parts
DocumentationVersion	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ documentation/versions/ *version*
DocumentationVersions	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ documentation/versions
GatewayResponse	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ gatewayresponses/ *response-type*
GatewayResponses	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ gatewayresponses
Integrazione	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ resources/ *resource-id* /methods/ *http-method* /integration

Risorsa	ARN
IntegrationResponse	arn:*partition* :apigateway: *region*::/restapis/ *api-id*/resources/ *resource-id* /methods/ *http-method* /integration/responses/ *status-code*
Metodo	arn:*partition* :apigateway: *region*::/restapis/ *api-id*/resources/ *resource-id* /methods/ *http-method*
MethodResponse	arn:*partition* :apigateway: *region*::/restapis/ *api-id*/resources/ *resource-id* /methods/ *http-method* /responses/ *status-code*
Modello	arn:*partition* :apigateway: *region*::/restapis/ *api-id*/models/*model-name*
Modelli	arn:*partition* :apigateway: *region*::/restapis/ *api-id*/models
RequestValidator	arn:*partition* :apigateway: *region*::/restapis/ *api-id*/requestvalidators/ *id*
RequestValidators	arn:*partition* :apigateway: *region*::/restapis/ *api-id*/requestvalidators
Risorsa	arn:*partition* :apigateway: *region*::/restapis/ *api-id*/resources/ *id*

Risorsa	ARN
Resources	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
Stage	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ stages/*stage-name*
Stage	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ stages
Tag	arn:*partition* :apigatew ay: *region*::/tags/*url-encoded-resource-arn*
Modello	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*

Risorsa	ARN
UsagePlanKeys	arn:*partition* :apigatew ay: *region*::/usageplans/ *usageplan -id* /keys
VpcLink	arn:*partition* :apigatew ay: *region*::/vpclinks/ *vpclink-id*
VpcLinks	arn:*partition* :apigatew ay: *region*::/vpclinks

Risorse nomi dominio

Risorsa	ARN
ApiMapping	arn:*partition* :apigatew ay: *region*::/domainnames/ *domain-na me* /apimappings/ *id*
ApiMappings	arn:*partition* :apigatew ay: *region*::/domainnames/ *domain-na me* /apimappings
BasePathMapping	arn:*partition* :apigatew ay: *region*::/domainnames/ *domain-na me* /basepathmappings/ *basepath*
BasePathMappings	arn:*partition* :apigatew ay: *region*::/domainnames/ *domain-na me* /basepathmappings
DomainName	arn:*partition* :apigatew ay: *region*::/domainnames/ *domain-na me*

Risorsa	ARN
DomainNameAccessAssociation	arn:*partition* :apigateway: *region*:*account-id* :/domainnameaccessassociations/domainname/ *domain-name* /vpcesource/ *vpce-source-id*
DomainNameAccessAssociations	arn:*partition* :apigateway: *region*:*account-id* :/domainnameaccessassociations
DomainNames	arn:*partition* :apigateway: *region*::/domainnames
RoutingRule	arn:*partition* :apigateway: *region*:*account-id* :/domainnames/ *domain-name* /routingrules/ *routing-rule-id*

execute-api(HTTP APIs e WebSocket APIs REST APIs)

Risorsa	ARN
WebSocket Endpoint API	arn:*partition* :execute-api: *region*:*account-id* :*api-id*/*stage*/*route-key*
Endpoint API HTTP e REST API *	arn:*partition* :execute-api: *region*:*account-id* :*api-id*/*stage*/*http-method* /*resource-path*
Provider di autorizzazioni Lambda **	arn:*partition* :execute-api: *region*:*account-id* :*api-id*/authorizers/ *authorizer-id*

* L'ARN per l'endpoint della `$default` route per HTTP è. APIs `arn:`*`partition`*`:execute-api:`*`region`*`:`*`account-id`*`:`*`api-id`*`/*/$default`

** Questo ARN è applicabile solo quando si imposta la condizione `SourceArn` nella policy delle risorse per una funzione Lambda di autorizzazione. Per un esempio, consulta the section called “Creare un'autorizzazione Lambda”.

Estensioni OpenAPI per Gateway API

Le estensioni API Gateway supportano l'autorizzazione AWS specifica e le integrazioni Gateway-specific API API per API REST e API HTTP. In questa sezione vengono descritte le estensioni API Gateway nella specifica OpenAPI.

> **Tip**
>
> Per comprendere come vengono utilizzate le estensioni API Gateway in un'applicazione, puoi usare la console API Gateway per creare un'API REST o API HTTP ed esportarla in un file di definizione OpenAPI. Per ulteriori informazioni su come esportare un'API, consulta Esportazione di un'API REST da API Gateway e Esporta HTTP APIs da API Gateway.

Argomenti

- Oggetto x-amazon-apigateway-any-method
- x-amazon-apigateway-cors object
- Proprietà x-amazon-apigateway-api-key-source
- Oggetto x-amazon-apigateway-auth
- Oggetto x-amazon-apigateway-authorizer
- Proprietà x-amazon-apigateway-authtype
- Proprietà x-amazon-apigateway-binary-media-types
- Oggetto x-amazon-apigateway-documentation
- x-amazon-apigateway-endpoint-access-mode
- Oggetto x-amazon-apigateway-endpoint-configuration
- Oggetto x-amazon-apigateway-gateway-responses
- Oggetto x-amazon-apigateway-gateway-responses.gatewayResponse
- Oggetto x-amazon-apigateway-gateway-responses.responseParameters
- Oggetto x-amazon-apigateway-gateway-responses.responseTemplates
- x-amazon-apigateway-importexport-version
- Oggetto di x-amazon-apigateway-integration
- Oggetto x-amazon-apigateway-integrations
- Oggetto x-amazon-apigateway-integration.requestTemplates

- Oggetto x-amazon-apigateway-integration.requestParameters
- Oggetto x-amazon-apigateway-integration.responses
- Oggetto x-amazon-apigateway-integration.response
- Oggetto x-amazon-apigateway-integration.responseTemplates
- Oggetto x-amazon-apigateway-integration.responseParameters
- Oggetto x-amazon-apigateway-integration.tlsConfig
- x-amazon-apigateway-minimum-compression-size
- x-amazon-apigateway-policy
- Proprietà x-amazon-apigateway-request-validator
- Oggetto x-amazon-apigateway-request-validators
- Oggetto x-amazon-apigateway-request-validators.requestValidator
- x-amazon-apigateway-security-policy
- Proprietà x-amazon-apigateway-tag-value

Oggetto x-amazon-apigateway-any-method

Specifica l'oggetto OpenAPI Operation per il metodo `ANY` catch-all di API Gateway in un oggetto OpenAPI Path Item. Questo oggetto può coesistere con altri oggetti Operation e rileverà qualsiasi metodo HTTP non esplicitamente dichiarato.

Nella tabella seguente sono elencate le proprietà estese da API Gateway. Per le altre proprietà OpenAPI Operation, consulta le specifiche di OpenAPI.

Nome proprietà	Tipo	Descrizione
`isDefaultRoute`	`Boolean`	Specificare se una route è la route `$default`. Supportat o solo per le API HTTP. Per ulteriori informazioni, consulta Crea percorsi per HTTP APIs in API Gateway.
`x-amazon-apigateway-integration`	Oggetto di x-amazon-apigateway-integration	Specifica l'integrazione del metodo con il back-end. Si tratta di una proprietà

Nome proprietà	Tipo	Descrizione
		estesa dell'oggetto OpenAPI Operation. L'integrazione può essere del tipo `AWS`, `AWS_PROXY` , `HTTP`, `HTTP_PROXY` o `MOCK`.

Esempio di x-amazon-apigateway-any-method

L'esempio seguente integra il metodo `ANY` per una risorsa proxy, `{proxy+}`, con una funzione 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"
        }
```

Nell'esempio seguente viene creata una route `$default` per un'API HTTP che si integra con una funzione 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 object

Specificare la configurazione CORS (Cross-origin Resource Sharing) per un'API HTTP. L'estensione si applica alla struttura OpenAPI a livello root. Per ulteriori informazioni, consulta Configurazione di CORS per HTTP APIs in API Gateway.

Nome proprietà	Tipo	Descrizione
`allowOrigins`	`Array`	Specifica le origini consentite.
`allowCredentials`	`Boolean`	Specifica se le credenziali sono incluse nella richiesta CORS.
`exposeHeaders`	`Array`	Specifica le intestazioni esposte.
`maxAge`	`Integer`	Specifica il numero di secondi in cui il browser deve memorizzare nella cache i risultati delle richieste preliminari.
`allowMethods`	`Array`	Specifica i metodi HTTP consentiti.

Nome proprietà	Tipo	Descrizione
`allowHeaders`	`Array`	Specifica le intestazioni consentite.

Esempio x-amazon-apigateway-cors example

Di seguito è riportato un esempio di configurazione CORS per un'API HTTP.

```
"x-amazon-apigateway-cors": {
    "allowOrigins": [
      "https://www.example.com"
    ],
    "allowCredentials": true,
    "exposeHeaders": [
      "x-apigateway-header",
      "x-amz-date",
      "content-type"
    ],
    "maxAge": 3600,
    "allowMethods": [
      "GET",
      "OPTIONS",
      "POST"
    ],
    "allowHeaders": [
      "x-apigateway-header",
      "x-amz-date",
      "content-type"
    ]
}
```

Proprietà x-amazon-apigateway-api-key-source

Specifica l'origine per la ricezione di una chiave API per il throttling dei metodi API che richiedono una chiave. Questa API-level proprietà è un tipo. `String` Per ulteriori informazioni sulla configurazione di un metodo per richiedere una chiave API, consulta the section called “Configurazione di un metodo per utilizzare le chiavi API con una definizione OpenAPI”.

Specifica l'origine della chiave API per le richieste. I valori validi sono:

- `HEADER` per ricevere la chiave API dall'intestazione `X-API-Key` di una richiesta.
- `AUTHORIZER` per ricevere la chiave API da `UsageIdentifierKey` di un provider di autorizzazioni Lambda (nota in precedenza come autorizzazioni ad hoc).

Esempio di x-amazon-apigateway-api-key-source

L'esempio seguente imposta l'intestazione `X-API-Key` come fonte della chiave 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",
   .
```

```
    .
    .
}
```

Oggetto x-amazon-apigateway-auth

Definisce un tipo di autorizzazione da applicare per l'autorizzazione dei richiami dei metodi in API Gateway.

Nome proprietà	Tipo	Descrizione
`type`	`string`	Specifica il tipo di autorizzazione. Specifica `"NONE"` per l'accesso aperto. Specifica `"AWS_IAM"` per l'utilizzo di autorizzazioni IAM. I valori non rispettano la distinzione tra maiuscole e minuscole.

Esempio di x-amazon-apigateway-auth

Nell'esempio seguente viene impostato il tipo di autorizzazione per un metodo 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"
        }
      }
    }
```

```
  }
}
```

Oggetto x-amazon-apigateway-authorizer

Definisce un sistema di autorizzazione Lambda, pool di utenti Amazon Cognito o provider di autorizzazioni JWT da applicare per l'autorizzazione delle invocazioni dei metodi in API Gateway. Questa estensione si applica alla definizione di sicurezza in OpenAPI 2 e allo schema di sicurezza in OpenAPI 3.

Nome proprietà	Tipo	Descrizione
`type`	`string`	Il tipo di autorizzazione. Questa proprietà è obbligatoria. Per API REST, specificare `token` per un'autorizzazione con l'identità dell'intermediario incorporata in un token di autorizzazione. Specificare `request` per un'autorizzazione con l'identità dell'intermediario contenuta nei parametri della richiesta. Specifica `cognito_user_pools` per un provider di autorizzazioni che utilizza un pool di utenti Amazon Cognito per controllare l'accesso all'API. Per API HTTP, specificare `request` per un'autorizzazione Lambda con l'identità dell'intermediario contenuta nei parametri della richiesta.

Nome proprietà	Tipo	Descrizione
		Specificare `jwt` per un'autorizzazione JWT.
`authorizerUri`	`string`	L'URI (Uniform Resource Identifier) della funzione Lambda del provider di autorizzazioni. La sintassi è esposta di seguito: `"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`	Le credenziali richieste per invocare il provider di autorizzazioni, se presente, nel formato di un ARN di un ruolo di esecuzione IAM. Ad esempio, «arn:aws:iam:::». *account-id IAM_role*
`authorizerPayloadFormatVersion`	`string`	Per API HTTP, specifica il formato dei dati inviati da API Gateway a un provider di autorizzazioni Lambda e come API Gateway interpreta la risposta da Lambda. Per ulteriori informazioni, consulta the section called “Tipo di formato payload”.

Nome proprietà	Tipo	Descrizione
`enableSimpleResponses`	`Boolean`	Per API HTTP, specifica se un provider di autorizzazioni `request` restituisce un valore booleano o una policy IAM. Supportato solo per le autorizzazioni con un `authorizerPayloadFormatVersion` `2.0`. Se abilitata, la funzione del provider di autorizzazioni Lambda restituisce un valore booleano. Per ulteriori informazioni, consulta the section called “Risposta della funzione Lambda per il formato 2.0”.
`identitySource`	`string`	Un elenco separato da virgole di espressioni di mappatura dei parametri di richiesta come origine di identità. Applicabile solo per l'autorizzazione del tipo `request` e `jwt`.
`jwtConfiguration`	`Object`	Specifica l'emittente e i gruppi di destinatari per un autorizzatore JWT. Per ulteriori informazioni, consulta JWTConfiguration nella documentazione di riferimento dell'API Gateway versione 2. Supportato solo per le API HTTP.

Nome proprietà	Tipo	Descrizione
`identityValidationExpression`	`string`	Espressione regolare per la convalida del token come identità in ingresso. Ad esempio, "^x-[a-z]+". Supportata solo per i sistemi di autorizzazione `TOKEN` per REST API.
`authorizerResultTtlInSeconds`	`string`	Numero di secondi durante i quali il risultato dell'auto rizzazione viene memorizzato nella cache.
`providerARNs`	Una matrice di `string`	Un elenco di ARN del pool di utenti Amazon Cognito per `COGNITO_USER_POOLS` .

Esempi di x-amazon-apigateway-authorizer per API REST

Il seguente esempio di definizioni di sicurezza OpenAPI specifica un provider di autorizzazioni Lambda di tipo "token" denominato `test-authorizer`.

```
 "securityDefinitions" : {
   "test-authorizer" : {
     "type" : "apiKey",                          // Required and the value must be
"apiKey" for an API Gateway API.
     "name" : "Authorization",                   // The name of the header containing
the authorization token.
     "in" : "header",                            // Required and the value must be
"header" for an API Gateway API.
     "x-amazon-apigateway-authtype" : "custom", // Specifies the authorization
mechanism for the client.
     "x-amazon-apigateway-authorizer" : {       // An API Gateway Lambda authorizer
definition
       "type" : "token",                         // Required property and the value
must "token"
```

```
        "authorizerUri" : "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:account-id:function:function-name/invocations",
        "authorizerCredentials" : "arn:aws:iam::account-id:role",
        "identityValidationExpression" : "^x-[a-z]+",
        "authorizerResultTtlInSeconds" : 60
      }
    }
  }
```

Il seguente frammento di oggetto operazione OpenAPI imposta `GET /http` per l'uso del provider di autorizzazioni Lambda precedente.

```
  "/http" : {
     "get" : {
       "responses" : { },
       "security" : [ {
         "test-authorizer" : [ ]
       } ],
       "x-amazon-apigateway-integration" : {
         "type" : "http",
         "responses" : {
           "default" : {
             "statusCode" : "200"
           }
         },
         "httpMethod" : "GET",
         "uri" : "http://api.example.com"
       }
     }
   }
```

Il seguente esempio di definizioni di sicurezza OpenAPI specifica un provider di autorizzazioni Lambda di tipo "request", con un parametro di intestazione singolo (`auth`) come origine di identità. Il nome di `securityDefinitions` è `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
      }
    }
}
```

Il seguente esempio di definizioni di sicurezza OpenAPI specifica un provider di autorizzazioni Lambda di tipo "request", con un'intestazione (`HeaderAuth1`) e un parametro di stringa di query `QueryString1` come origini di identità.

```
"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
      }
    }
```

```
}
```

Il seguente esempio di definizioni di sicurezza OpenAPI specifica un provider di autorizzazioni Lambda API Gateway di tipo "request", con una variabile di fase singola (`stage`) come origine di identità.

```
"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
      }
    }
}
```

Il seguente esempio di definizione di sicurezza OpenAPI specifica un pool di utenti Amazon Cognito come provider di autorizzazioni.

```
 "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"
        ]
```

```
      }
    }
```

Il seguente frammento di oggetto dell'operazione OpenAPI imposta `GET /http` per utilizzare il pool di utenti di Amazon Cognito precedente come provider di autorizzazioni, senza ambiti personalizzati.

```
  "/http" : {
     "get" : {
      "responses" : { },
      "security" : [ {
        "cognito-pool" : [ ]
      } ],
      "x-amazon-apigateway-integration" : {
        "type" : "http",
        "responses" : {
          "default" : {
            "statusCode" : "200"
          }
        },
        "httpMethod" : "GET",
        "uri" : "http://api.example.com"
      }
    }
   }
```

Esempi di x-amazon-apigateway-authorizer per API HTTP

Nell'esempio OpenAPI 3.0 riportato di seguito viene creato un provider di autorizzazioni JWT per un'API HTTP che utilizza Amazon Cognito come provider di identità, con l'intestazione `Authorization` come origine identità.

```
"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"
    }
  }
}
```

L'esempio di OpenAPI 3.0 seguente produce lo stesso autorizzatore JWT dell'esempio precedente. Tuttavia, questo esempio utilizza la proprietà `openIdConnectUrl` di OpenAPI per rilevare automaticamente l'emittente. Il `openIdConnectUrl` deve essere completamente formato.

```
"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"
    }
  }
}
```

Nell'esempio seguente viene creato un provider di autorizzazioni Lambda per un'API HTTP. In questo esempio l'autorizzazione utilizza l'intestazione `Authorization` come origine di identità. L'autorizzazione utilizza il tipo di formato payload `2.0` e restituisce il valore booleano, poiché `enableSimpleResponses` è impostato su `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
    }
  }
}
```

Proprietà x-amazon-apigateway-authtype

Per le API REST, questa estensione può essere utilizzata per definire un tipo personalizzato di un provider di autorizzazioni Lambda. In questo caso, il valore è in formato libero. Ad esempio, un'API può avere più provider di autorizzazioni Lambda che utilizzano schemi interni diversi. È possibile utilizzare questa estensione per identificare lo schema interno di un provider di autorizzazioni Lambda.

Più comunemente, nelle API HTTP e nelle API REST, l'estensione può anche essere utilizzata come un modo per definire l'autorizzazione IAM in diverse operazioni che condividono lo stesso schema di sicurezza. In questo caso, il termine `awsSigv4` è un termine riservato, insieme a qualsiasi termine con prefisso `aws`.

Questa estensione si applica allo schema di sicurezza dei tipi `apiKey` in OpenAPI 2 e OpenAPI 3.

Esempio di x-amazon-apigateway-authtype

Il seguente esempio OpenAPI 3 definisce l'autorizzazione IAM su più risorse in un'API REST o in un'API HTTP:

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "openapi3",
    "version" : "1.0"
  },
  "paths" : {
    "/operation1" : {
      "get" : {
        "responses" : {
          "default" : {
            "description" : "Default response"
```

```
          }
        },
        "security" : [ {
          "sigv4Reference" : [ ]
        } ]
      }
    },
    "/operation2" : {
      "get" : {
        "responses" : {
          "default" : {
            "description" : "Default response"
          }
        },
        "security" : [ {
          "sigv4Reference" : [ ]
        } ]
      }
    }
  },
  "components" : {
    "securitySchemes" : {
      "sigv4Reference" : {
        "type" : "apiKey",
        "name" : "Authorization",
        "in" : "header",
        "x-amazon-apigateway-authtype": "awsSigv4"
      }
    }
  }
}
```

Il seguente esempio OpenAPI 3 definisce un provider di autorizzazioni Lambda con uno schema personalizzato per un'API REST:

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "openapi3 for REST API",
    "version" : "1.0"
  },
  "paths" : {
    "/protected-by-lambda-authorizer" : {
```

```
      "get" : {
        "responses" : {
          "200" : {
            "description" : "Default response"
          }
        },
        "security" : [ {
          "myAuthorizer" : [ ]
        } ]
      }
    }
  },
  "components" : {
    "securitySchemes" : {
      "myAuthorizer" : {
        "type" : "apiKey",
        "name" : "Authorization",
        "in" : "header",
        "x-amazon-apigateway-authorizer" : {
          "identitySource" : "method.request.header.Authorization",
          "authorizerUri" : "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:account-id:function:function-name/invocations",
          "authorizerResultTtlInSeconds" : 300,
          "type" : "request",
          "enableSimpleResponses" : false
        },
        "x-amazon-apigateway-authtype": "Custom scheme with corporate claims"
      }
    }
  },
  "x-amazon-apigateway-importexport-version" : "1.0"
}
```

Consulta anche

authorizer.authType

Proprietà x-amazon-apigateway-binary-media-types

Specifica l'elenco di tipi di supporto binari che devono essere supportati da API Gateway, ad esempio `application/octet-stream` e `image/jpeg`. Questa estensione è una matrice JSON. Deve essere inclusa come estensione del fornitore di primo livello nel documento OpenAPI.

Esempio di x-amazon-apigateway-binary-media-types

L'esempio di seguito mostra l'ordine di ricerca della codifica di un'API

```
"x-amazon-apigateway-binary-media-types": [ "application/octet", "image/jpeg" ]
```

Oggetto x-amazon-apigateway-documentation

Definisce le parti della documentazione da importare in API Gateway. Si tratta di un oggetto JSON contenente una matrice di istanze di `DocumentationPart`.

Nome proprietà	Tipo	Descrizione
`documentationParts`	`Array`	Matrice delle istanze di `DocumentationPart` esportate o importate.
`version`	`String`	L'identificatore di versione dello snapshot delle parti di documentazione esportate.

Esempio di x-amazon-apigateway-documentation

L'esempio seguente dell'estensione di API Gateway in OpenAPI definisce le istanze di `DocumentationParts` da importare in o esportare da un'API in API Gateway.

```
{ ...
  "x-amazon-apigateway-documentation": {
    "version": "1.0.3",
    "documentationParts": [
       {
         "location": {
           "type": "API"
       },
        "properties": {
          "description": "API description",
          "info": {
```

```
            "description": "API info description 4",
            "version": "API info version 3"
          }
        }
      },
      {
         … // Another DocumentationPart instance
      }
    ]
  }
}
```

x-amazon-apigateway-endpoint-access-mode

Specifica una modalità di accesso all'endpoint per un'API REST o un nome di dominio personalizzato. Per ulteriori informazioni sulla modalità di accesso agli endpoint, vedere. Modalità di accesso agli endpoint

esempio di x-amazon-apigateway-endpoint-access-mode

L'esempio seguente specifica la modalità di accesso agli endpoint su strict.

```
"x-amazon-apigateway-endpoint-access-mode": "STRICT"
```

Oggetto x-amazon-apigateway-endpoint-configuration

Specifica i dettagli della configurazione dell'endpoint per un'API. Questa estensione è una proprietà estesa dell'oggetto OpenAPI Operation. Questo oggetto deve essere presente nelle estensioni dei fornitori di primo livello per Swagger 2.0. Per OpenAPI 3.0, deve essere presente nelle estensioni dei fornitori dell'oggetto Server.

Nome proprietà	Tipo	Descrizione
`disableExecuteApiEndpoint`	Booleano	Specifica se i client possono richiamare l'API utilizzando l'endpoint `execute-api` predefinito. Per impostazione predefinita, i client possono

Nome proprietà	Tipo	Descrizione
		richiamare l'API con l'endpoint `https://{api_id}.execute-api.{region}.amazonaws.com` predefinito. Per richiedere che i client utilizzino un nome di dominio personalizzato per richiamare l'API, specifica `true`.
`vpcEndpointIds`	Una matrice di `String`	Un elenco di VpcEndpoint identificatori in base ai quali creare record alias Route 53 per un'API REST. È supportato solo per il tipo di endpoint `PRIVATE` di API REST.
`ipAddressType`	`string`	I tipi di indirizzo IP che possono invocare un'API HTTP. Utilizza `ipv4` per consentire ai tipi di indirizzo IPv4 di invocare un'API HTTP. Utilizza `dualstack` per consentire ai tipi di indirizzo IPv4 e IPv6 di invocare un'API HTTP. È supportata solo per le API HTTP.

Esempi di x-amazon-apigateway-endpoint-configuration

L'esempio seguente associa gli endpoint VPC specificati all'API REST.

```
"x-amazon-apigateway-endpoint-configuration": {
    "vpcEndpointIds": ["vpce-0212a4ababd5b8c3e", "vpce-01d622316a7df47f9"]
}
```

Nell'esempio seguente l'endpoint predefinito per un'API viene disattivato.

```
"x-amazon-apigateway-endpoint-configuration": {
    "disableExecuteApiEndpoint": true
}
```

Nell'esempio seguente si imposta il tipo di indirizzo IP su Dualstack per un'API HTTP.

```
"x-amazon-apigateway-endpoint-configuration": {
    "ipAddressType": "dualstack"
}
```

Oggetto x-amazon-apigateway-gateway-responses

Definisce le risposte del gateway per un'API come una stringa da una stringa a una mappa di coppie GatewayResponse chiave-valore. L'estensione si applica alla struttura OpenAPI a livello root.

Nome proprietà	Tipo	Descrizione
responseType	x-amazon-apigateway-gateway-responses.gatewayResponse	A `GatewayResponse` per quanto specificato. *responseType*

Esempio di x-amazon-apigateway-gateway-responses

Il seguente esempio di estensione API Gateway a OpenAPI definisce una GatewayResponses mappa che contiene due GatewayResponse istanze, una per il `DEFAULT_4XX` tipo e l'altra per il tipo. `INVALID_API_KEY`

```
{
  "x-amazon-apigateway-gateway-responses": {
    "DEFAULT_4XX": {
      "responseParameters": {
        "gatewayresponse.header.Access-Control-Allow-Origin": "'domain.com'"
      },
      "responseTemplates": {
        "application/json": "{\"message\": test 4xx b }"
```

```
      }
    },
    "INVALID_API_KEY": {
      "statusCode": "429",
      "responseTemplates": {
        "application/json": "{\"message\": test forbidden }"
      }
    }
  }
}
```

Oggetto x-amazon-apigateway-gateway-responses.gatewayResponse

Definisce una risposta del gateway di un tipo di risposta specificato, inclusi il codice di stato, eventuali modelli di risposta o parametri di risposta applicabili.

Nome proprietà	Tipo	Descrizione
responseParameters	x-amazon-apigateway-gateway-responses.responseParameters	Specifica i parametri, ovvero i GatewayResponse parametri dell'intestazione. I valori di parametro possono assumere qualsiasi valore di parametro di richiesta in ingresso o un valore personalizzato statico.
responseTemplates	x-amazon-apigateway-gateway-responses.responseTemplates	Specifica i modelli di mappatura della risposta del gateway. I modelli non sono elaborati dal motore VTL.
statusCode	`string`	Un codice di stato HTTP per la risposta del gateway.

Esempio di x-amazon-apigateway-gateway-responses.gatewayResponse

L'esempio seguente dell'estensione API Gateway per OpenAPI definisce la personalizzazione della `INVALID_API_KEY` risposta GatewayResponse per restituire il codice di stato di`456`, il valore dell'`api-key`intestazione della richiesta in arrivo e un messaggio. `"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\" }"
  }
}
```

Oggetto x-amazon-apigateway-gateway-responses.responseParameters

Definisce una mappa stringa a stringa di coppie chiave-valore per generare parametri di risposta del gateway dai parametri delle richieste in ingresso o mediante stringhe letterali. Supportato solo per API REST.

Nome proprietà	Tipo	Descrizione
`gatewayresponse. param-position .param-name`	`string`	*`param-position`* può essere `header`, `path` o `querystring` . Per ulteriori informazioni, consulta Mappatura dei parametri per REST API in Gateway API.

Esempio di x-amazon-apigateway-gateway-responses.responseParameters

Il seguente esempio di estensioni OpenAPI mostra un'espressione di mappatura dei parametri di GatewayResponse risposta per abilitare il supporto CORS per le risorse sui domini. `*.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
}
```

Oggetto x-amazon-apigateway-gateway-responses.responseTemplates

Definisce i modelli di GatewayResponse mappatura, come una mappa da stringa a stringa di coppie chiave-valore, per una determinata risposta del gateway. Per ogni coppia chiave-valore, la chiave è il tipo di contenuto. Ad esempio, application/json "e il valore è un modello di mappatura stringato per semplici sostituzioni di variabili. Un modello di mappatura `GatewayResponse` non viene elaborato dal motore Velocity Template Language (VTL).

Nome proprietà	Tipo	Descrizione
`content-type`	`string`	Modello di mappatura del corpo `GatewayResponse` che supporta solo la semplice sostituzione di variabili per personalizzare un corpo della risposta del gateway.

Esempio di x-amazon-apigateway-gateway-responses.responseTemplates

Il seguente esempio di estensioni OpenAPI mostra un modello di GatewayResponse mappatura per personalizzare una risposta di errore generata da API Gateway in un formato specifico dell'app.

```
      "responseTemplates": {
        "application/json": "{ \"message\": $context.error.messageString, \"type\":
$context.error.responseType, \"statusCode\": '488' }"
      }
```

Il seguente esempio di estensioni OpenAPI mostra un modello di GatewayResponse mappatura per sovrascrivere una risposta di errore generata da API Gateway con un messaggio di errore statico.

```
      "responseTemplates": {
        "application/json": "{ \"message\": 'API-specific errors' }"
      }
```

x-amazon-apigateway-importexport-version

Specifica la versione dell'algoritmo di importazione e di esportazione API Gateway per API HTTP. Attualmente, l'unico valore supportato è `1.0`. Per ulteriori informazioni, consulta exportVersion nella documentazione di riferimento di API Gateway versione 2.

esempio di x-amazon-apigateway-importexport-version

Nell'esempio seguente viene impostata la versione di importazione ed esportazione su `1.0`.

```
{
    "openapi": "3.0.1",
    "x-amazon-apigateway-importexport-version": "1.0",
    "info": { ...
```

Oggetto di x-amazon-apigateway-integration

Specifica i dettagli dell'integrazione di back-end utilizzati per questo metodo. Questa estensione è una proprietà estesa dell'oggetto OpenAPI Operation. Il risultato è un oggetto di integrazione API Gateway.

Nome proprietà	Tipo	Descrizione
`cacheKeyParameters`	Una matrice di `string`	Elenco di parametri di richiesta i cui valori devono essere memorizzati nella cache.
`cacheNamespace`	`string`	Un gruppo di API-specific tag con i relativi parametri memorizzati nella cache.
`connectionId`	`string`	L'ID di a VpcLink per l'integrazione privata.
`connectionType`	`string`	Il tipo di connessione dell'integrazione. Il valore valido è `"VPC_LINK"` per l'integrazione privata, in caso contrario, `"INTERNET"` .
`credentials`	`string`	Per le credenziali AWS IAM basate sui ruoli, specifica l'ARN di un ruolo IAM appropriato. Se non specifica te, vengono utilizzate per impostazione predefinita le autorizzazioni basate su risorsa che devono essere aggiunte manualmente per consentire all'API di accedere alla risorsa. Per ulteriori informazioni, consulta l'argomento relativo alla concessione di autorizzazioni mediante una policy per le risorse. Nota: quando usi le credenziali IAM, assicurati che siano

Nome proprietà	Tipo	Descrizione
		abilitati gli endpoint regionali AWS STS per la regione in cui questa API viene distribuita per ottenere le prestazioni migliori.
`contentHandling`	`string`	Tipi di conversione per la codifica del payload delle richieste. Valori validi sono 1) `CONVERT_TO_TEXT` , per la conversione di un payload binario in una stringa con codifica base64 o per la conversione di un payload di testo in una stringa con codifica `utf-8` o per passare il payload di testo in modo nativo senza modifica e 2) `CONVERT_TO_BINARY` , per la conversione di un payload di testo nel BLOB con decodifica base64 o per passare un payload binario in modo nativo senza modifica.
`httpMethod`	`string`	Il metodo HTTP utilizzato nella richiesta di integrazione. Per le invocazioni della funzione Lambda, il valore deve essere `POST`.

Nome proprietà	Tipo	Descrizione
`integrationSubtype`	`string`	Specifica il sottotipo di integrazione per l'integrazione di un servizio. AWS Supportato solo per le API HTTP. Per i sottotipi di integrazione supportati, consulta the section called “AWSRiferimento alle integrazioni di servizi ”.
`integrationTarget`	`string`	Il listener ALB o NLB a cui inviare la richiesta. Supportato solo per le integrazioni private che utilizzano VPC Links V2. Per ulteriori informazioni, consulta the section called “Collegamenti VPC V2”.
`passthroughBehavior`	`string`	Specifica il modo in cui deve essere passato un payload di richiesta del tipo di contenuto non mappato attraverso la richiesta di integrazione senza modifiche. I valori supportati sono `when_no_templates` , `when_no_match` e `never`. Per ulteriori informazioni, consulta Integration.passthroughBehavior.

Nome proprietà	Tipo	Descrizione
`payloadFormatVersion`	`string`	Specifica il formato del payload inviato a un'integr azione. Richiesto per le API HTTP. Per le API HTTP, i valori supportati per le integrazioni proxy Lambda sono `1.0` e `2.0`. Per tutte le altre integrazioni, `1.0` è l'unico valore supportato. Per ulteriori informazioni, consulta the section called “AWS Lambda integrazioni” e the section called “AWSRiferimento alle integrazioni di servizi ”.

Nome proprietà	Tipo	Descrizione
`requestParameters`	Oggetto x-amazon-apigateway-integration.requestParameters	Per API REST, specifica le mappature dai parametri delle richieste dei metodi ai parametri delle richieste di integrazione. I parametri di richiesta supportati sono `querystring` , `path`, `header` e `body`. Per le API HTTP, i parametri di richiesta sono una mappa chiave-valore che specifica i parametri che vengono passati alle integrazioni `AWS_PROXY` con un valore specificato `integrationSubtype` . È possibile fornire valori statici o dati di richiesta di mappatura , variabili di stadio o variabili di contesto che vengono valutate in fase di esecuzione. Per ulteriori informazioni, consulta the section called “AWS integrazioni di servizi”.
`requestTemplates`	Oggetto x-amazon-apigateway-integration.requestTemplates	I modelli di mappatura per un payload di richiesta dei tipi MIME specificati.
`responses`	Oggetto x-amazon-apigateway-integration.responses	Definisce le risposte di metodo e specifica le mappature di payload o dei parametri desiderate dalle risposte di integrazione alle risposte di metodo.

Nome proprietà	Tipo	Descrizione
`responseTransferMode`	`string`	La modalità di trasferimento delle risposte dell'integrazione. Si usa `BUFFERED` per fare in modo che API Gateway attenda di ricevere la risposta completa prima di iniziare la trasmissione. Si usa `STREAM` per fare in modo che API Gateway invii risposte parziali al client non appena sono disponibili. Per ulteriori informazioni, consulta the section called “Trasmetti in streaming la risposta di integrazione per le tue integrazioni proxy”.
`timeoutInMillis`	`integer`	Timeout di integrazione tra 50 ms e 29.000 ms.

Nome proprietà	Tipo	Descrizione
`type`	`string`	Il tipo di integrazione con il back-end specificato. I valori validi sono: • `http` o `http_proxy` , per l'integrazione con un back-end HTTP. • `aws_proxy` , per l'integrazione con le funzioni AWS Lambda. • `aws`, per l'integrazione con funzioni AWS Lambda o altri AWS servizi, come Amazon DynamoDB, Amazon Simple Notification Service o Amazon Simple Queue Service. • `mock`, per l'integrazione con API Gateway senza richiamare alcun back-end. Per ulteriori informazioni sui tipi di integrazione, consulta integration:type.
`tlsConfig`	the section called “x-amazon-apigateway-integration.tlsConfig”	Specifica la configurazione TLS per un'integrazione.

Nome proprietà	Tipo	Descrizione
`uri`	`string`	L'URI dell'endpoint del back-end. Per le integrazioni del tipo `aws`, è un valore ARN. Per l'integrazione HTTP, è l'URL dell'endpoint HTTP che include lo schema `https` o `http`.

Esempio di x-amazon-apigateway-integration

Per API HTTP, è possibile definire le integrazioni nella sezione componenti della definizione OpenAPI. Per ulteriori informazioni, consulta Oggetto x-amazon-apigateway-integrations.

```
"x-amazon-apigateway-integration": {
    "$ref": "#/components/x-amazon-apigateway-integrations/integration1"
}
```

L'esempio seguente crea un'integrazione con una funzione Lambda. A scopo dimostrativo, si presume che i modelli di mappatura `responseTemplates` di esempio mostrati in `requestTemplates` e gli esempi seguenti si applichino al seguente JSON-formatted payload: per generare un output JSON o un `{ "name":"value_1", "key":"value_2", "redirect": {"url" :"..."} }` output XML di. `{ "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'"
            }
        }
    }
}
```

I doppi apici (") della stringa JSON nei modelli di mappatura devono includere il carattere di escape di stringa (\").

Oggetto x-amazon-apigateway-integrations

Definisce una raccolta di integrazioni. Puoi definire le integrazioni nella sezione componenti della definizione OpenAPI e riutilizzare le integrazioni per più route. Supportato solo per le API HTTP.

Nome proprietà	Tipo	Description
integration	Oggetto di x-amazon-apigateway-integration	Una raccolta di oggetti di integrazione.

Esempio x-amazon-apigateway-integrations

Nell'esempio seguente viene creata un'API HTTP che definisce due integrazioni e fa riferimento alle integrazioni utilizzando `$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"
            }
        }
    }
}
```

Oggetto x-amazon-apigateway-integration.requestTemplates

Specifica i modelli di mappatura per un payload di richiesta dei tipi MIME specificati.

Nome proprietà	Tipo	Descrizione
MIME type	`string`	Un esempio del tipo MIME è `application/json` . Per informazioni su come creare un modello di mappatura, consulta the section called “Trasformazioni dei modelli di mappatura”.

Esempio di x-amazon-apigateway-integration.requestTemplates

Il modello seguente imposta i modelli di mappatura per un payload di richiesta de tipi MIME `application/json` e `application/xml`.

```
"requestTemplates" : {
    "application/json" : "#set ($root=$input.path('$')) { \"stage\": \"$root.name\",
 \"user-id\": \"$root.key\" }",
    "application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</stage> "
```

```
}
```

Oggetto x-amazon-apigateway-integration.requestParameters

Per API REST, specifica le mappature dai parametri delle richieste dei metodi denominati ai parametri delle richieste di integrazione. Prima di fare riferimento ai parametri delle richieste dei metodi, è necessario definirli.

Per API HTTP, specifica i parametri che vengono passati alle integrazioni `AWS_PROXY` con un `integrationSubtype` specificato.

Nome proprietà	Tipo	Descrizione
`integration.request.` *`<param-type>`* `.` *`<param-name>`*	`string`	Per API REST, il valore è generalmente un parametro di richiesta di metodo predefinito nel formato `method.request.` *`<param-type>`* `.` *`<param-name>`*, dove `<param-type>` può essere `querystring`, `path`, `header` o `body`. Tuttavia, sono validi anche `$context.` *`VARIABLE_NAME`*, `$stageVariables.` *`VARIABLE_NAME`* e *`STATIC_VALUE`*. Per il parametro `body`, `<param-name>` è un'espressione di percorso JSON senza il prefisso `$.`
`parameter`	`string`	Per le API HTTP, i parametri di richiesta sono una mappa chiave-valore che specifica i

Nome proprietà	Tipo	Descrizione
		parametri che vengono passati alle integrazioni `AWS_PROXY` con un valore specificato `integrationSubtype` . È possibile fornire valori statici o dati di richiesta di mappatura , variabili di stadio o variabili di contesto che vengono valutate in fase di esecuzion e. Per ulteriori informazioni, consulta the section called "AWS integrazioni di servizi".

Esempio di x-amazon-apigateway-integration.requestParameters

L'esempio di mappatura dei parametri di richiesta seguente converte i parametri di query (`version`), intestazione (`x-user-id`) e percorso (`service`) della richiesta del metodo rispettivamente in parametri di query (`stage`), intestazione (`x-userid`) e percorso (op) della richiesta di integrazione.

> **Note**
>
> Se stai creando risorse tramite OpenAPI o, i valori statici devono essere racchiusi tra virgolette singole. CloudFormation
> Per aggiungere questo valore dalla console, immetti `application/json` nella casella, senza virgolette.

```
"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"
},
```

Oggetto x-amazon-apigateway-integration.responses

Definisce le risposte di metodo e specifica le mappature di payload o dei parametri dalle risposte di integrazione alle risposte di metodo.

Nome proprietà	Tipo	Description
`Response status pattern`	Oggetto x-amazon-apigateway-integration.response	Un'espressione regolare utilizzata per abbinare la risposta di integrazione alla risposta del metodo, o `default` per raccogliere qualsiasi risposta che non è stata configurata. Per le integrazioni HTTP, la regex si applica al codice di stato della risposta di integrazione. Per le chiamate Lambda, la regex si applica al `errorMessage` campo dell'oggetto di informazioni sull'errore restituito da AWS Lambda come corpo di risposta al fallimento o quando l'esecuzione della funzione Lambda genera un'eccezione. **Note** Il nome della *`Response status pattern`* proprietà si riferisce a un codice di stato della risposta

Nome proprietà	Tipo	Description
		o a un'espressione regolare che descrive un gruppo di codici di stato della risposta. Non corrisponde ad alcun identificatore di una IntegrationResponse risorsa nell'API REST di API Gateway.

Esempio di x-amazon-apigateway-integration.response

L'esempio seguente mostra un elenco di risposte dalle risposte `2xx` e `302`. Per la risposta `2xx`, la risposta del metodo è mappata dal payload della risposta di integrazione del tipo MIME `application/json` o `application/xml`. Questa risposta usa i modelli di mappatura forniti. Per la risposta `302`, la risposta del metodo restituisce un'intestazione `Location` il cui valore viene derivato dalla proprietà `redirect.url` nel payload della risposta di integrazione.

```
"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"
        }
    }
}
```

Oggetto x-amazon-apigateway-integration.response

Definisce una risposta e specifica le mappature di payload o dei parametri dalla risposta di integrazione alla risposta di metodo.

Nome proprietà	Tipo	Descrizione
`statusCode`	`string`	Codice di stato HTTP per la risposta di metodo, ad esempi, `"200"`. Deve corrispondere a una risposta associata nel campo OpenAPI Operation `responses` .
`responseTemplates`	Oggetto x-amazon-apigateway-integration.responseTemplates	Indica i modelli di mappatura specifici del tipo MIME per il payload della risposta.
`responseParameters`	Oggetto x-amazon-apigateway-integration.responseParameters	Specifica le mappature dei parametri per la risposta. Solo i parametri `header` e `body` della risposta di integrazione possono essere mappati ai parametri `header` del metodo.
`contentHandling`	`string`	Tipi di conversione per la codifica del payload delle risposte. Valori validi sono 1) `CONVERT_TO_TEXT` , per la conversione di un payload binario in una stringa con codifica base64 o per la conversione di un payload di testo in una stringa con

Nome proprietà	Tipo	Descrizione
		codifica `utf-8` o per passare il payload di testo in modo nativo senza modifica e 2) `CONVERT_TO_BINARY` , per la conversione di un payload di testo nel BLOB con decodifica base64 o per passare un payload binario in modo nativo senza modifica.

esempio di x-amazon-apigateway-integration.response

L'esempio che segue definisce una risposta `302` per il metodo che deriva un payload del tipo MIME `application/json` o `application/xml` dal back-end. La risposta usa i modelli di mappatura forniti e restituisce l'URL di reindirizzamento dalla risposta di integrazione nell'intestazione `Location` del metodo.

```
{
    "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"
    }
}
```

Oggetto x-amazon-apigateway-integration.responseTemplates

Specifica i modelli di mappatura per un payload di risposta dei tipi MIME specificati.

Nome proprietà	Tipo	Descrizione
MIME type	string	Specifica un modello di mappatura per trasforma re il corpo della risposta di integrazione nel corpo della risposta di metodo per un tipo MIME specificato. Per informazioni su come creare un modello di mappatura, consulta the section called "Trasformazioni dei modelli di mappatura". Un esempio di è. *MIME type* application/ json

Esempio di x-amazon-apigateway-integration.responseTemplate

Il modello seguente imposta i modelli di mappatura per un payload di richiesta de tipi MIME `application/json` e `application/xml`.

```
"responseTemplates" : {
    "application/json" : "#set ($root=$input.path('$')) { \"stage\": \"$root.name\",
 \"user-id\": \"$root.key\" }",
    "application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</stage> "
}
```

Oggetto x-amazon-apigateway-integration.responseParameters

Specifica le mappature dai parametri delle risposte dei metodi di integrazione ai parametri delle risposte di metodo. Puoi mappare `header`, `body` o valori statici al tipo `header` della risposta dei metodi. Supportato solo per API REST.

Nome proprietà	Tipo	Description
`method.response.header. <param-name>`	`string`	Il valore di parametro denominato può essere derivato dai tipi `header` e `body` di parametri delle risposte di integrazione.

Esempio di X-Amazon-ApiGateway-Integration.ResponseParameters

L'esempio seguente mappa i parametri `body` e `header` della risposta di integrazione a due parametri `header` della risposta di metodo.

```
"responseParameters" : {
    "method.response.header.Location" : "integration.response.body.redirect.url",
    "method.response.header.x-user-id" : "integration.response.header.x-userid"
}
```

Oggetto x-amazon-apigateway-integration.tlsConfig

Specifica la configurazione TLS per un'integrazione.

Nome proprietà	Tipo	Descrizione
`insecureSkipVerification`	`Boolean`	Supportato solo per API REST. Specifica se API Gateway ignora o meno la verifica che il certificato per un endpoint di integrazione sia emesso da un'autorità di certificazione supportata. Questa operazione non è consigliata, ma consente di utilizzare certificati firmati

Nome proprietà	Tipo	Descrizione
		da autorità di certifica zione private o certificati autofirmati. Se abilitato, API Gateway esegue comunque la convalida di base del certifica to, che include il controllo della data di scadenza, del nome host, della presenza di un'autorità di certificazione principale e dell'integrità della catena di certificati. La convalida della catena di certificati include l'applicazione dei vincoli di nome X509v3. Se riscontri errori di convalida con l'`insecureSkipVerifi cation` opzione abilitata, verifica che i certificati CA non includano le estensioni Name Constraints o riemettili senza tale estensione. Il certificato radice appartenente all'autor ità privata deve soddisfare i seguenti vincoli: • L'estensione x509 `keyUsage` deve avere `keyCertSign` . • L'estensione x509 `basicConstraints` deve avere `CA:TRUE`. Supportato solo per integrazi oni `HTTP` e `HTTP_PROXY` .

Nome proprietà	Tipo	Descrizione
		Warning L'abilitazione di `insecureSkipVerification` non è consigliata, soprattutto per le integrazioni con endpoint HTTPS pubblici. Se abiliti `insecureSkipVerification`, aumenti il rischio di attacchi man in the middle.
`serverNameToVerify`	`string`	Supportato solo per le integrazioni private API HTTP. Se si specifica un nome server, API Gateway lo utilizza per verificare il nome host sul certificato dell'integrazione. Il nome del server è incluso anche nell'handshake TLS per supportare SNI (Server Name Indication) o l'hosting virtuale.

Esempi di x-amazon-apigateway-integration.tlsConfig

L'esempio seguente di OpenAPI 3.0 abilita `insecureSkipVerification` per un'integrazione del proxy HTTP API REST.

```
"x-amazon-apigateway-integration": {
  "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
```

```
  "responses": {
     default": {
       "statusCode": "200"
      }
  },
  "passthroughBehavior": "when_no_match",
  "httpMethod": "ANY",
  "tlsConfig" : {
    "insecureSkipVerification" : true
  }
  "type": "http_proxy",
}
```

L'esempio seguente di OpenAPI 3.0 specifica un `serverNameToVerify` per un'integrazione privata API HTTP.

```
"x-amazon-apigateway-integration" : {
  "payloadFormatVersion" : "1.0",
  "connectionId" : "abc123",
  "type" : "http_proxy",
  "httpMethod" : "ANY",
  "uri" : "arn:aws:elasticloadbalancing:us-west-2:123456789012:listener/app/my-load-
balancer/50dc6c495c0c9188/0467ef3c8400ae65",
  "connectionType" : "VPC_LINK",
  "tlsConfig" : {
     "serverNameToVerify" : "example.com"
  }
}
```

x-amazon-apigateway-minimum-compression-size

Specifica la dimensione minima di compressione per un'API REST. Per abilitare la compressione, specificare un numero intero compreso tra 0 e 10485760. Per ulteriori informazioni, consulta Compressione del payload per REST APIs in API Gateway.

x-amazon-apigateway-minimum-compression-size example

Nell'esempio seguente viene specificata una dimensione minima di compressione di `5242880` byte per un'API REST.

```
"x-amazon-apigateway-minimum-compression-size": 5242880
```

x-amazon-apigateway-policy

Specifica una policy delle risorse per un'API REST. Per ulteriori informazioni sulle policy delle risorse, consulta Controllo degli accessi a una REST API con le policy delle risorse Gateway API. Per esempi di policy delle risorse, consulta Esempi di policy delle risorse API Gateway.

esempio di x-amazon-apigateway-policy

Nell'esempio seguente viene specificata una policy delle risorse per un'API REST. La policy delle risorse nega (blocca) il traffico in entrata a un'API da un blocco di indirizzi IP di origine specificato. Al momento dell'importazione, `"execute-api:/*"` viene convertito in`arn:aws:execute-api:`*`region`*`:`*`account-id`*`:`*`api-id`*`/*`, utilizzando la regione corrente, l'ID dell' AWS account e l'attuale ID API REST.

```
"x-amazon-apigateway-policy": {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
               "execute-api:/*"
            ],
            "Condition" : {
                "IpAddress": {
                    "aws:SourceIp": "192.0.2.0/24"
                }
            }
        }
    ]
}
```

Proprietà x-amazon-apigateway-request-validator

Specifica un validatore di richiesta facendo riferimento a *`request_validator_name`* della mappa Oggetto x-amazon-apigateway-request-validators per abilitare la convalida della richiesta in un metodo o nell'API che la contiene. Il valore di questa estensione è una stringa JSON.

Questa estensione può essere specificata a livello di API o a livello di metodo. Il API-level validatore si applica a tutti i metodi a meno che non venga sovrascritto dal validatore a livello di metodo.

Esempio di x-amazon-apigateway-request-validator

L'esempio che segue applica il validatore di richieste `basic` a livello d API, mentre applica il validatore di richieste `parameter-only` per la richiesta `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",
        ...
     }
}
```

Oggetto x-amazon-apigateway-request-validators

Definisce i validatori di richieste supportati per l'API che le contiene come mappa tra un nome di validatore e le regole di convalida delle richieste associate. Questa estensione si applica a un'API REST.

Nome proprietà	Tipo	Descrizione
request_validator_name	Oggetto x-amazon-apigateway-request-validators.requestValidator	Specifica le regole di convalida costituite dal validatore denominato. Ad esempio: `"basic" : { "validateRequestBody" : true, "validateRequestParameters" : true },` Per applicare il validatore a un metodo specifico, fai riferimento al rispettivo nome (`basic`) come al valore della proprietà Proprietà x-amazon-apigateway-request-validator.

esempio di x-amazon-apigateway-request-validators

L'esempio che segue mostra un set di validatori di richieste per un'API come mappa tra un nome di validatore e le regole di convalida delle richieste associate.

OpenAPI 2.0

```
{
  "swagger": "2.0",
  ...
```

```
  "x-amazon-apigateway-request-validators" : {
    "basic" : {
      "validateRequestBody" : true,
      "validateRequestParameters" : true
    },
    "params-only" : {
      "validateRequestBody" : false,
      "validateRequestParameters" : true
    }
  },
  ...
}
```

Oggetto x-amazon-apigateway-request-validators.requestValidator

Specifica le regole di convalida di un validatore di richieste nell'ambito della definizione della mappa Oggetto x-amazon-apigateway-request-validators.

Nome proprietà	Tipo	Descrizione
`validateRequestBody`	`Boolean`	Specifica se convalidare (`true`) o non convalidare (`false`) il corpo della richiesta .
`validateRequestParameters`	`Boolean`	Specifica se convalidare (`true`) o non convalidare (`false`) i parametri di richiesta necessari.

Esempio di x-amazon-apigateway-request-validators.requestValidator

L'esempio seguente mostra un validatore di risposte di soli parametri.

```
"params-only": {
    "validateRequestBody" : false,
    "validateRequestParameters" : true
}
```

x-amazon-apigateway-security-policy

Specifica una politica di sicurezza per un'API REST. Se si crea una politica di sicurezza che inizia con"`SecurityPolicy_`", è necessario impostare anche la modalità di accesso agli endpoint. Per ulteriori informazioni sulle politiche di sicurezza, vederePolitiche di sicurezza per le API REST in API Gateway.

esempio di x-amazon-apigateway-security-policy

L'esempio seguente specifica un'API REST. `SecurityPolicy_TLS13_1_3_2025_09`

```
"x-amazon-apigateway-security-policy": "SecurityPolicy_TLS13_1_3_2025_09"
```

Proprietà x-amazon-apigateway-tag-value

Specifica il valore di un tag AWS per un'API HTTP. È possibile utilizzare la `x-amazon-apigateway-tag-value` proprietà come parte dell'oggetto tag OpenAPI a livello di root per specificare AWS i tag per un'API HTTP. Se specifichi un nome tag senza la proprietà `x-amazon-apigateway-tag-value`, API Gateway crea un tag con una stringa vuota per un valore.

Per ulteriori informazioni sul tagging, consulta Tagging delle risorse API Gateway.

Nome proprietà	Tipo	Descrizione
`name`	`String`	Specifica la chiave di tag.
`x-amazon-apigateway-tag-value`	`String`	Specifica il valore del tag.

Esempio di x-amazon-apigateway-tag-value

L'esempio seguente specifica due tag per un'API HTTP:

- "Owner": "Admin"
- "Prod": ""

```
"tags": [
```

```
    {
      "name": "Owner",
      "x-amazon-apigateway-tag-value": "Admin"
    },
    {
      "name": "Prod"
    }
]
```

Sicurezza in Amazon API Gateway

La sicurezza del cloud AWS è la massima priorità. In qualità di AWS cliente, puoi beneficiare di un data center e di un'architettura di rete progettati per soddisfare i requisiti delle organizzazioni più sensibili alla sicurezza.

La sicurezza è una responsabilità condivisa tra AWS te e te. Il modello di responsabilità condivisa descrive questo modello come sicurezza del cloud e sicurezza nel cloud:

- Sicurezza del cloud: AWS è responsabile della protezione dell'infrastruttura che gestisce AWS i servizi nel AWS cloud. AWS ti fornisce anche servizi che puoi utilizzare in modo sicuro. I revisori esterni testano e verificano regolarmente l'efficacia della nostra sicurezza nell'ambito dei AWS Programmi di AWS conformità dei Programmi di conformità dei di . Per informazioni sui programmi di conformità che si applicano ad Amazon API Gateway, consulta AWS i servizi nell'ambito del programma di conformità, i AWS servizi nell'ambito del programma .
- Sicurezza nel cloud: la tua responsabilità è determinata dal AWS servizio che utilizzi. L'utente è anche responsabile di altri fattori, tra cui la riservatezza dei dati, i requisiti della propria azienda e le leggi e normative vigenti.

Questa documentazione aiuta a comprendere come applicare il modello di responsabilità condivisa quando si usa API Gateway. Gli argomenti seguenti illustrano come configurare API Gateway per soddisfare gli obiettivi di sicurezza e conformità. Scopri anche come utilizzare altri AWS servizi che ti aiutano a monitorare e proteggere le tue risorse API Gateway.

Per ulteriori informazioni, consulta Panoramica della sicurezza in Amazon API Gateway.

Argomenti

- Protezione dei dati in Amazon API Gateway
- Gestione delle identità e degli accessi per Amazon API Gateway
- Logging e monitoraggio in Amazon API Gateway
- Convalida della conformità per Amazon API Gateway
- Resilienza in Amazon API Gateway
- Sicurezza dell'infrastruttura in Amazon API Gateway
- Analisi delle vulnerabilità in Amazon API Gateway
- Best practice di sicurezza in Amazon API Gateway

Protezione dei dati in Amazon API Gateway

Il modello di responsabilità AWS condivisa (modello di) si applica alla protezione dei dati in Amazon API Gateway. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura globale che gestisce tutti i Cloud AWS. L'utente è responsabile del controllo dei contenuti ospitati su questa infrastruttura. L'utente è inoltre responsabile della configurazione della protezione e delle attività di gestione per i Servizi AWS utilizzati. Per ulteriori informazioni sulla privacy dei dati, consulta Domande frequenti sulla privacy dei dati . Per ulteriori informazioni sulla protezione dei dati in Europa, consulta il Centro generale sulla protezione dei dati (GDPR).

Ai fini della protezione dei dati, ti consigliamo di proteggere Account AWS le credenziali e configurare singoli utenti con Centro identità AWS IAM o AWS Identity and Access Management (IAM). In tal modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere i suoi compiti. Suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- SSL/TLS Da utilizzare per comunicare con AWS le risorse. È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Configura l'API e la registrazione delle attività degli utenti con AWS CloudTrail. Per informazioni sull'utilizzo dei CloudTrail percorsi per acquisire AWS le attività, consulta Lavorare con i CloudTrail percorsi nella Guida per l'AWS CloudTrail utente.
- Utilizza soluzioni di AWS crittografia, insieme a tutti i controlli di sicurezza predefiniti all'interno Servizi AWS.
- Utilizza i servizi di sicurezza gestiti avanzati, come Amazon Macie, che aiutano a individuare e proteggere i dati sensibili archiviati in Amazon S3.
- Se hai bisogno di moduli crittografici convalidati FIPS 140-3 per accedere AWS tramite un'interfaccia a riga di comando o un'API, usa un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-3.

Ti consigliamo di non inserire mai informazioni riservate o sensibili, ad esempio gli indirizzi e-mail dei clienti, nei tag o nei campi di testo in formato libero, ad esempio nel campo Nome. Ciò include quando lavori con API Gateway o altro Servizi AWS utilizzando la console, l'API o AWS gli SDK. AWS CLI I dati inseriti nei tag o nei campi di testo in formato libero utilizzati per i nomi possono essere utilizzati per i la fatturazione o i log di diagnostica. Quando si fornisce un URL a un server esterno, suggeriamo vivamente di non includere informazioni sulle credenziali nell'URL per convalidare la richiesta al server.

Crittografia dei dati in Amazon API Gateway

Lo scopo della protezione dati è proteggere i dati sia in transito (durante la trasmissione verso e da API Gateway), sia quando sono inattivi (ovvero quando sono archiviati in AWS).

Crittografia dei dati inattivi in Amazon API Gateway

Se si sceglie di abilitare la memorizzazione nella cache per un'API REST, è possibile abilitare la crittografia della cache. Per ulteriori informazioni, consulta Impostazioni della cache per REST API in Gateway API.

Per ulteriori informazioni sulla protezione dei dati, consulta il post del blog AWS Modello di responsabilità condivisa e GDPR su AWS Security Blog.

Crittografia e decrittografia della chiave privata del certificato

Quando si crea un nome di dominio personalizzato per le API private, il certificato ACM e la chiave privata vengono crittografati utilizzando una chiave KMS gestita da AWS con l'alias aws/acm. È possibile visualizzare l'ID della chiave con questo alias nella console AWS KMS sotto Chiavi gestite da AWS.

Gateway API non accede direttamente alle risorse ACM. Utilizzare AWS TLS Connection Manager per proteggere e accedere alle chiavi private del certificato. Quando si usa il certificato ACM per creare un nome di dominio personalizzato Gateway API per API private, Gateway API associa il certificato ad AWS TLS Connection Manager. Questa operazione viene eseguita creando una concessione in AWS KMS per la chiave gestita da AWS con il prefisso aws/acm. La concessione è uno strumento delle policy che permette a TLS Connection Manager di usare le chiavi KMS nelle operazioni di crittografia. La concessione consente al principale assegnatario (TLS Connection Manager) di chiamare le operazioni di concessione specificate nella chiave KMS per decrittografare la chiave privata del certificato. TLS Connection Manager utilizza quindi il certificato e la chiave privata decrittografata (testo normale) per stabilire una connessione sicura (sessione SSL/TLS) con i client dei servizi Gateway API. Quando l'associazione del certificato a un nome di dominio personalizzato Gateway API per API private viene annullata, la concessione viene ritirata.

Per rimuovere l'accesso alla chiave KMS, si consiglia di sostituire o eliminare il certificato dal servizio utilizzando la Console di gestione AWS o il comando `update-service` in AWS CLI.

Contesto di crittografia per Gateway API

Il contesto di crittografia è un set opzionale di coppie chiave-valore che contiene informazioni contestuali su come può essere utilizzata la chiave privata. AWS KMS associa il contesto di

crittografia ai dati crittografati e lo utilizza come dati autenticati aggiuntivi per supportare la crittografia autenticata.

Quando le chiavi TLS vengono utilizzate con Gateway API e TLS Connection Manager, il nome del servizio Gateway API viene incluso nel contesto di crittografia utilizzato per crittografare la chiave a riposo. È possibile verificare per quale nome di dominio personalizzato Gateway API vengono utilizzati il certificato e la chiave privata esaminando il contesto di crittografia nei log di CloudTrail, come mostrato nella prossima sezione, o consultando la scheda Risorse associate nella console ACM.

Per decrittografare i dati, includere nella richiesta lo stesso contesto di crittografia. Gateway API utilizza lo stesso contesto di crittografia in tutte le operazioni di crittografia di AWS KMS, in cui la chiave è `aws:apigateway:arn` e il valore è il nome della risorsa Amazon (ARN) della risorsa `PrivateDomainName` di Gateway API.

L'esempio illustrato di seguito mostra il contesto di crittografia nell'output di un'operazione come `CreateGrant`.

```
"constraints": {
"encryptionContextEquals": {
"aws:acm:arn": "arn:aws:acm:us-west-2:859412291086:certificate/9177097a-
f0ae-4be1-93b1-19f911ea4f88",
"aws:apigateway:arn": "arn:aws:apigateway:us-west-2:859412291086:/domainnames/denytest-
part1.pdx.sahig.people.aws.dev+cbaeumzjhg"
}
},
"operations": [
"Decrypt"
],
"granteePrincipal": "tlsconnectionmanager.amazonaws.com"
```

Crittografia dei dati in transito in Amazon API Gateway

Le API create con Amazon API Gateway espongono solo endpoint HTTPS. API Gateway non supporta gli endpoint non crittografati (HTTP).

API Gateway gestisce i certificati per gli endpoint predefiniti `execute-api`. Se si configura un nome di dominio personalizzato, si specifica il certificato per il nome di dominio. Come best practice, non bloccare i certificati.

Per una maggiore sicurezza, è possibile scegliere una versione del protocollo Transport Layer Security (TLS) minima da applicare per il dominio API Gateway personalizzato. Le API WebSocket e le API HTTP supportano solo TLS 1.2. Per ulteriori informazioni, consulta Scegli una politica di sicurezza per il tuo dominio personalizzato in API Gateway.

È inoltre possibile impostare una distribuzione Amazon CloudFront con un certificato SSL personalizzato nel proprio account e utilizzarla con API regionali. Puoi quindi configurare la policy di sicurezza per la distribuzione CloudFront con TLS 1.1 o versione successiva in base ai tuoi requisiti di sicurezza e conformità.

Per ulteriori informazioni sulla protezione dei dati, consulta Protezione delle REST API in Gateway API e il post del blog relativo al modello di responsabilità condivisa e GDPR AWS nel Blog sulla sicurezza AWS.

Riservatezza del traffico Internet

Utilizzando Amazon API Gateway, puoi creare API REST private cui è possibile accedere solo da Amazon Virtual Private Cloud (VPC). Il VPC utilizza un endpoint VPC di interfaccia, che è un'interfaccia di rete endpoint creata nel VPC. Utilizzando le policy delle risorse, è possibile consentire o negare l'accesso alle tue API da VPC ed endpoint VPC selezionati, inclusi gli account AWS. Ogni endpoint può essere utilizzato per l'accesso a più API private. È inoltre possibile utilizzare Direct Connect per stabilire una connessione da una rete on-premise ad Amazon VPC e accedere all'API privata durante tale connessione. In tutti i casi, il traffico alla tua API privata utilizza connessioni sicure e non lascia la rete Amazon; è isolato dall'Internet pubblico. Per ulteriori informazioni, consulta the section called “REST API private”.

Gestione delle identità e degli accessi per Amazon API Gateway

AWS Identity and Access Management (IAM) è un programma Servizio AWS che aiuta un amministratore a controllare in modo sicuro l'accesso alle AWS risorse. Gli amministratori IAM controllano chi è autenticato (accesso effettuato) e autorizzato (dispone di autorizzazioni) a utilizzare le risorse di API Gateway. IAM è un software Servizio AWS che puoi utilizzare senza costi aggiuntivi.

Argomenti

- Destinatari
- Autenticazione con identità
- Gestione dell’accesso tramite policy
- Come funziona Amazon API Gateway con IAM

- Esempi di policy basate su identità di Amazon API Gateway
- Esempi di policy basate su risorse di Amazon API Gateway
- Risoluzione dei problemi di identità e accesso di Amazon API Gateway
- Utilizzo dei ruoli collegati ai servizi per API Gateway

Destinatari

Il modo in cui utilizzi AWS Identity and Access Management (IAM) varia in base al tuo ruolo:

- Utente del servizio: richiedi le autorizzazioni all'amministratore se non riesci ad accedere alle funzionalità (consulta Risoluzione dei problemi di identità e accesso di Amazon API Gateway)
- Amministratore del servizio: determina l'accesso degli utenti e invia le richieste di autorizzazione (consulta Come funziona Amazon API Gateway con IAM)
- Amministratore IAM: scrivi policy per gestire l'accesso (consulta Esempi di policy basate su identità di Amazon API Gateway)

Autenticazione con identità

L'autenticazione è il modo in cui accedi AWS utilizzando le tue credenziali di identità. Devi autenticarti come utente IAM o assumendo un ruolo IAM. Utente root dell'account AWS

Puoi accedere come identità federata utilizzando credenziali provenienti da una fonte di identità come Centro identità AWS IAM (IAM Identity Center), autenticazione Single Sign-On o credenziali. Google/ Facebook Per ulteriori informazioni sull'accesso, consulta Come accedere all' Account AWS nella Guida per l'utente di Accedi ad AWS .

Per l'accesso programmatico, AWS fornisce un SDK e una CLI per firmare crittograficamente le richieste. Per ulteriori informazioni, consulta AWS Signature Version 4 per le richieste API nella Guida per l'utente di IAM.

Account AWS utente root

Quando si crea un Account AWS, si inizia con un'identità di accesso denominata utente Account AWS root che ha accesso completo a tutte Servizi AWS le risorse. Consigliamo vivamente di non utilizzare l'utente root per le attività quotidiane. Per le attività che richiedono le credenziali come utente root, consulta Attività che richiedono le credenziali dell'utente root nella Guida per l'utente di IAM.

Utenti e gruppi IAM

Un utente IAM è una identità che dispone di autorizzazioni specifiche per una singola persona o applicazione. Ti consigliamo di utilizzare credenziali temporanee invece di utenti IAM con credenziali a lungo termine. Per ulteriori informazioni, consulta Richiedere agli utenti umani di utilizzare la federazione con un provider di identità per accedere AWS utilizzando credenziali temporanee nella Guida per l'utente IAM.

Un gruppo IAM specifica una raccolta di utenti IAM e semplifica la gestione delle autorizzazioni per gestire gruppi di utenti di grandi dimensioni. Per ulteriori informazioni, consulta Casi d'uso per utenti IAM nella Guida per l'utente di IAM.

Ruoli IAM

Un ruolo IAM è un'identità con autorizzazioni specifiche che fornisce credenziali temporanee. Puoi assumere un ruolo passando da un ruolo utente a un ruolo IAM (console) o chiamando un'operazione AWS CLI o AWS API. Per ulteriori informazioni, consulta Metodi per assumere un ruolo nella Guida per l'utente di IAM.

I ruoli IAM sono utili per l'accesso degli utenti federati, le autorizzazioni utente IAM temporanee, l'accesso multi-account, l'accesso multi-servizio e le applicazioni in esecuzione su Amazon EC2. Per maggiori informazioni, consultare Accesso a risorse multi-account in IAM nella Guida per l'utente IAM.

Gestione dell'accesso tramite policy

Puoi controllare l'accesso AWS creando policy e associandole a AWS identità o risorse. Una policy definisce le autorizzazioni quando è associata a un'identità o a una risorsa. AWS valuta queste politiche quando un preside effettua una richiesta. La maggior parte delle politiche viene archiviata AWS come documenti JSON. Per maggiori informazioni sui documenti delle policy JSON, consulta Panoramica delle policy JSON nella Guida per l'utente IAM.

Utilizzando le policy, gli amministratori specificano chi ha accesso a cosa definendo quale principale può eseguire azioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Un amministratore IAM crea le policy IAM e le aggiunge ai ruoli, che gli utenti possono quindi assumere. Le policy IAM definiscono le autorizzazioni indipendentemente dal metodo utilizzato per eseguirle.

Policy basate sull'identità

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile collegare a un'identità (utente, gruppo o ruolo). Tali policy controllano le operazioni autorizzate per l'identità, nonché le risorse e le condizioni in cui possono essere eseguite. Per informazioni su come creare una policy basata su identità, consultare Definizione di autorizzazioni personalizzate IAM con policy gestite dal cliente nella Guida per l'utente IAM.

Le policy basate su identità possono essere policy in linea (con embedding direttamente in una singola identità) o policy gestite (policy autonome collegate a più identità). Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta Scegliere tra policy gestite e policy in linea nella Guida per l'utente di IAM.

Policy basate sulle risorse

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Gli esempi includono le policy di trust dei ruoli IAM e le policy dei bucket di Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. In una policy basata sulle risorse è obbligatorio specificare un'entità principale.

Le policy basate sulle risorse sono policy inline che si trovano in tale servizio. Non è possibile utilizzare le policy AWS gestite di IAM in una policy basata sulle risorse.

Elenchi di controllo degli accessi () ACLs

Le liste di controllo degli accessi (ACLs) controllano quali principali (membri dell'account, utenti o ruoli) dispongono delle autorizzazioni per accedere a una risorsa. ACLs sono simili alle politiche basate sulle risorse, sebbene non utilizzino il formato del documento di policy JSON.

Amazon S3 e Amazon VPC sono esempi di servizi che supportano. AWS WAF ACLs Per ulteriori informazioni ACLs, consulta la panoramica della lista di controllo degli accessi (ACL) nella Amazon Simple Storage Service Developer Guide.

Altri tipi di policy

AWS supporta tipi di policy aggiuntivi che possono impostare le autorizzazioni massime concesse dai tipi di policy più comuni:

- Limiti delle autorizzazioni: imposta il numero massimo di autorizzazioni che una policy basata su identità ha la possibilità di concedere a un'entità IAM. Per ulteriori informazioni, consulta Limiti delle autorizzazioni per le entità IAM nella Guida per l'utente di IAM.
- Politiche di controllo del servizio (SCPs): specificano le autorizzazioni massime per un'organizzazione o un'unità organizzativa in. AWS Organizations Per ulteriori informazioni, consultare Policy di controllo dei servizi nella Guida per l'utente di AWS Organizations .
- Politiche di controllo delle risorse (RCPs): imposta le autorizzazioni massime disponibili per le risorse nei tuoi account. Per ulteriori informazioni, consulta Politiche di controllo delle risorse (RCPs) nella Guida per l'AWS Organizations utente.
- Policy di sessione: policy avanzate passate come parametro quando si crea una sessione temporanea per un ruolo o un utente federato. Per maggiori informazioni, consultare Policy di sessione nella Guida per l'utente IAM.

Più tipi di policy

Quando a una richiesta si applicano più tipi di policy, le autorizzazioni risultanti sono più complicate da comprendere. Per scoprire come si AWS determina se consentire o meno una richiesta quando sono coinvolti più tipi di policy, consulta Logica di valutazione delle policy nella IAM User Guide.

Come funziona Amazon API Gateway con IAM

Prima di utilizzare IAM per gestire l'accesso ad API Gateway è necessario comprendere quali caratteristiche IAM sono disponibili per l'utilizzo con questo servizio. Per avere una visione di alto livello di come API Gateway e altri AWS servizi funzionano con IAM, consulta AWS Services That Work with IAM nella IAM User Guide.

Argomenti

- Policy basate su identità dei API Gateway
- Policy basate su risorse di API Gateway
- Autorizzazione basata sui tag di API Gateway
- Ruoli IAM di API Gateway

Policy basate su identità dei API Gateway

Con le policy basate su identità di IAM, è possibile specificare quali operazioni e risorse sono consentite o rifiutate, nonché le condizioni in base alle quali le operazioni sono consentite o rifiutate.

API Gateway supporta specifiche operazioni, risorse e chiavi di condizione. Per ulteriori informazioni sulle operazioni, risorse e chiavi di condizione specifiche per API Gateway, consulta Operazioni, risorse e chiavi di condizione per Amazon API Gateway Management e Operazioni, risorse e chiavi di condizione per Amazon API Gateway Management V2. Per informazioni su tutti gli elementi utilizzati in una policy JSON, consulta Documentazione di riferimento degli elementi delle policy JSON IAM nella Guida per l'utente di IAM.

L'esempio seguente mostra una policy basata sull'identità che consente a un utente di creare o aggiornare solo REST privati. APIs Per ulteriori esempi, consulta the section called “Esempi di policy basate su identità”.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ScopeToPrivateApis",
      "Effect": "Allow",
      "Action": [
        "apigateway:PATCH",
        "apigateway:POST",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/restapis",
        "arn:aws:apigateway:us-east-1::/restapis/??????????"
      ],
      "Condition": {
        "ForAllValues:StringEqualsIfExists": {
          "apigateway:Request/EndpointType": "PRIVATE",
          "apigateway:Resource/EndpointType": "PRIVATE"
        }
      }
    },
    {
      "Sid": "AllowResourcePolicyUpdates",
      "Effect": "Allow",
      "Action": [
          "apigateway:UpdateRestApiPolicy"
      ],
      "Resource": [
```

```
            "arn:aws:apigateway:us-east-1::/restapis/*"
        ]
      }
    ]
}
```

Azioni

L' `Action` elemento di una policy JSON descrive le azioni che è possibile utilizzare per consentire o negare l'accesso a un criterio.

Le operazioni delle policy in API Gateway utilizzano il seguente prefisso prima dell'operazione: `apigateway:`. Le istruzioni della policy devono includere un elemento `Action` o `NotAction`. API Gateway definisce un proprio set di operazioni che descrivono le attività che è possibile eseguire con questo servizio.

L'`Action`espressione di gestione dell'API ha il formato`apigateway:`*`action`*, dove *`action`* è una delle seguenti azioni API Gateway: GET, POST, PUT, DELETE, PATCH (per aggiornare le risorse) o *, che sono tutte le azioni precedenti.

Alcuni esempi dell'espressione `Action` includono:

- **`apigateway:*`** per tutte le operazioni API Gateway.
- **`apigateway:GET`** solo per l'operazione GET in API Gateway.

Per specificare più azioni in una sola istruzione, separa ciascuna di esse con una virgola come mostrato di seguito:

```
"Action": [
      "apigateway:action1",
      "apigateway:action2"
```

Per informazioni sui verbi HTTP da utilizzare per operazioni specifiche di API Gateway, consulta Amazon API Gateway Version 1 API Reference (REST APIs) e Amazon API Gateway Version 2 API Reference (WebSocket e HTTP APIs).

Per ulteriori informazioni, consulta the section called “Esempi di policy basate su identità”.

Resources

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento JSON `Resource` della policy specifica l'oggetto o gli oggetti ai quali si applica l'operazione. Come best practice, specifica una risorsa utilizzando il suo nome della risorsa Amazon (ARN). Per le azioni che non supportano le autorizzazioni a livello di risorsa, si utilizza un carattere jolly (*) per indicare che l'istruzione si applica a tutte le risorse.

```
"Resource": "*"
```

Le risorse di API Gateway presentano il seguente formato ARN:

```
arn:aws:apigateway:region::resource-path-specifier
```

Ad esempio, per specificare un'API REST con l'id *api-id* e le relative risorse secondarie, ad esempio gli autorizzatori per l'istruzione, utilizzare il seguente ARN:

```
"Resource": "arn:aws:apigateway:us-east-2::/restapis/api-id/*"
```

Per specificare tutte le REST APIs e le risorse secondarie che appartengono a un account specifico, usa il carattere jolly (*):

```
"Resource": "arn:aws:apigateway:us-east-2::/restapis/*"
```

Per un elenco dei tipi di risorse API Gateway e relativi ARNs, vedereDocumentazione di riferimento Amazon Resource Name (ARN) API Gateway.

Chiavi di condizione

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento `Condition` specifica quando le istruzioni vengono eseguite in base a criteri definiti. È possibile compilare espressioni condizionali che utilizzano operatori di condizione, ad esempio uguale a o minore di, per soddisfare la condizione nella policy con i valori nella richiesta. Per

visualizzare tutte le chiavi di condizione AWS globali, consulta le chiavi di contesto delle condizioni AWS globali nella Guida per l'utente IAM.

API Gateway definisce il proprio set di chiavi di condizione e supporta anche l'uso di alcune chiavi di condizione globali. Per un elenco di chiavi di condizione di API Gateway, consulta Chiavi di condizione per Amazon API Gateway nella Guida per l'utente di IAM. Per informazioni su operazioni e risorse che è possibile utilizzare con una chiave di condizione, consulta Operazioni definite da Amazon API Gateway.

Per informazioni sull'assegnazione di tag, incluso il controllo degli accessi basato sugli attributi, consulta Applicazione di tag.

Esempi

Per esempi di policy basate su identità di API Gateway, consulta Esempi di policy basate su identità di Amazon API Gateway.

Policy basate su risorse di API Gateway

Le policy basate su risorse sono documenti di policy JSON che specificano le operazioni che possono essere eseguite da un'entità specificata sulla risorsa di API Gateway e nel rispetto di quali condizioni possono operare. API Gateway supporta politiche di autorizzazione basate sulle risorse per REST. APIs È possibile utilizzare le policy delle risorse per controllare chi può richiamare un'API REST. Per ulteriori informazioni, consulta the section called “Uso delle policy delle risorse API Gateway”.

Esempi

Per visualizzare esempi di policy basate su risorse di API Gateway, consulta Esempi di policy delle risorse API Gateway.

Autorizzazione basata sui tag di API Gateway

È possibile collegare dei tag alle risorse di API Gateway o passare dei tag in una richiesta ad API Gateway. Per controllare l'accesso basato su tag, fornisci informazioni sui tag nell'elemento condizione di una policy utilizzando le chiavi di condizione `apigateway:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* o `aws:TagKeys`. Per ulteriori informazioni sull'assegnazione di tag alle risorse dell'API Gateway, consulta the section called “Attribute-based controllo degli accessi”.

Per esempi di policy basate su identità per limitare l'accesso a una risorsa in base ai tag di tale risorsa, consulta Utilizzo dei tag per controllare l'accesso alle risorse REST API di Gateway API.

Ruoli IAM di API Gateway

Un ruolo IAM è un'entità all'interno del tuo AWS account che dispone di autorizzazioni specifiche.

Utilizzo di credenziali temporanee con API Gateway

È possibile utilizzare credenziali temporanee per effettuare l'accesso con la federazione, assumere un ruolo IAM o un ruolo multi-account. Ottieni credenziali di sicurezza temporanee chiamando operazioni AWS STS API come AssumeRoleo. GetFederationToken

API Gateway supporta l'utilizzo di credenziali temporanee.

Ruoli collegati ai servizi

I ruoli collegati ai AWS servizi consentono ai servizi di accedere alle risorse di altri servizi per completare un'azione per conto dell'utente. I ruoli collegati ai servizi sono visualizzati nell'account IAM e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati ai servizi, ma non può modificarle.

API Gateway supporta i ruoli collegati ai servizi. Per informazioni sulla creazione o la gestione di ruoli collegati ai servizi di API Gateway, consulta Utilizzo dei ruoli collegati ai servizi per API Gateway.

Ruoli dei servizi

Un servizio può assumere un ruolo del servizio a tuo nome. Un ruolo del servizio consente al servizio di accedere alle risorse in altri servizi per completare un'operazione per conto dell'utente. I ruoli del servizio vengono visualizzati nell'account IAM e sono di proprietà dell'account. Pertanto, un amministratore può modificare le autorizzazioni per questo ruolo. Tuttavia, il farlo potrebbe pregiudicare la funzionalità del servizio.

API Gateway supporta i ruoli del servizio.

Esempi di policy basate su identità di Amazon API Gateway

Per impostazione predefinita, gli utenti e i ruoli IAM non dispongono dell'autorizzazione per creare o modificare le risorse di API Gateway. Inoltre, non possono eseguire attività utilizzando Console di gestione AWS AWS CLI, o. AWS SDKs Un amministratore IAM deve creare policy IAM che concedono a utenti e ruoli l'autorizzazione per eseguire operazioni API specifiche sulle risorse specificate di cui hanno bisogno. L'amministratore devi quindi collegare queste policy a utenti o IAM che richiedono tali autorizzazioni.

Per informazioni sulla creazione di policy IAM, consulta Creazione di policy nella scheda JSON della Guida per l'utente di IAM. Per informazioni sulle operazioni, le risorse e le condizioni specifiche di API Gateway, consulta Operazioni, risorse e chiavi di condizione per Amazon API Gateway Management e Operazioni, risorse e chiavi di condizione per Amazon API Gateway Management V2.

Argomenti

- Best practice per le policy
- Consentire agli utenti di visualizzare le loro autorizzazioni
- Autorizzazioni di lettura semplici
- Creare solo autorizzatori REQUEST o JWT
- Richiedere che l'endpoint predefinito execute-api sia disabilitato
- Consenti agli utenti di creare o aggiornare solo REST privati APIs
- Richiedere che i route API abbiano l'autorizzazione
- Impedisci a un utente di creare o aggiornare un collegamento VPC
- Policy di esempio per l'utilizzo delle regole di routing

Best practice per le policy

Le policy basate su identità determinano se qualcuno può creare, accedere o eliminare risorse API Gateway nell'account. Queste operazioni possono comportare costi aggiuntivi per l' Account AWS. Quando si creano o modificano policy basate sull'identità, seguire queste linee guida e raccomandazioni:

- Inizia con le policy AWS gestite e passa alle autorizzazioni con privilegi minimi: per iniziare a concedere autorizzazioni a utenti e carichi di lavoro, utilizza le politiche AWS gestite che concedono le autorizzazioni per molti casi d'uso comuni. Sono disponibili nel tuo. Account AWS Ti consigliamo di ridurre ulteriormente le autorizzazioni definendo politiche gestite dai AWS clienti specifiche per i tuoi casi d'uso. Per maggiori informazioni, consulta Policy gestite da AWS o Policy gestite da AWS per le funzioni dei processi nella Guida per l'utente di IAM.
- Applicazione delle autorizzazioni con privilegio minimo - Quando si impostano le autorizzazioni con le policy IAM, concedere solo le autorizzazioni richieste per eseguire un'attività. È possibile farlo definendo le azioni che possono essere intraprese su risorse specifiche in condizioni specifiche, note anche come autorizzazioni con privilegio minimo. Per maggiori informazioni sull'utilizzo di IAM per applicare le autorizzazioni, consulta Policy e autorizzazioni in IAM nella Guida per l'utente di IAM.

- Condizioni d'uso nelle policy IAM per limitare ulteriormente l'accesso - Per limitare l'accesso ad azioni e risorse è possibile aggiungere una condizione alle policy. Ad esempio, è possibile scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate utilizzando SSL. Puoi anche utilizzare le condizioni per concedere l'accesso alle azioni del servizio se vengono utilizzate tramite uno specifico Servizio AWS, ad esempio CloudFormation. Per maggiori informazioni, consultare la sezione Elementi delle policy JSON di IAM: condizione nella Guida per l'utente di IAM.
- Utilizzo dello strumento di analisi degli accessi IAM per convalidare le policy IAM e garantire autorizzazioni sicure e funzionali - Lo strumento di analisi degli accessi IAM convalida le policy nuove ed esistenti in modo che aderiscano al linguaggio (JSON) della policy IAM e alle best practice di IAM. Lo strumento di analisi degli accessi IAM offre oltre 100 controlli delle policy e consigli utili per creare policy sicure e funzionali. Per maggiori informazioni, consultare Convalida delle policy per il Sistema di analisi degli accessi IAM nella Guida per l'utente di IAM.
- Richiedi l'autenticazione a più fattori (MFA): se hai uno scenario che richiede utenti IAM o un utente root nel Account AWS tuo, attiva l'MFA per una maggiore sicurezza. Per richiedere la MFA quando vengono chiamate le operazioni API, aggiungere le condizioni MFA alle policy. Per maggiori informazioni, consultare Protezione dell'accesso API con MFA nella Guida per l'utente di IAM.

Per maggiori informazioni sulle best practice in IAM, consulta Best practice di sicurezza in IAM nella Guida per l'utente di IAM.

Consentire agli utenti di visualizzare le loro autorizzazioni

Questo esempio mostra in che modo è possibile creare una policy che consente agli utenti IAM di visualizzare le policy inline e gestite che sono collegate alla relativa identità utente. Questa policy include le autorizzazioni per completare questa azione sulla console o utilizzando programmaticamente l'API o. AWS CLI 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": "*"
        }
    ]
}
```

Autorizzazioni di lettura semplici

Questa politica di esempio fornisce all'utente il permesso di ottenere informazioni su tutte le risorse di un HTTP o di un' WebSocket API con l'identificatore di `a123456789` nella AWS regione us-east-1. La risorsa `arn:aws:apigateway:us-east-1::/apis/a123456789/*` include tutte le risorse secondarie dell'API, ad esempio autorizzatori e distribuzioni.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:GET"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis/a123456789/*"
```

```
      ]
    }
  ]
}
```

Creare solo autorizzatori REQUEST o JWT

Questa politica di esempio consente a un utente di creare solo `REQUEST` o APIs con `JWT` autorizzatori, anche tramite importazione. Nella sezione `Resource` della policy, `arn:aws:apigateway:us-east-1::/apis/??????????` richiede che le risorse abbiano un massimo di 10 caratteri, che escludono le risorse secondarie di un'API. Questo esempio utilizza `ForAllValues` nella sezione `Condition` perché gli utenti possono creare più autorizzatori contemporaneamente importando un'API.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "OnlyAllowSomeAuthorizerTypes",
      "Effect": "Allow",
      "Action": [
        "apigateway:PUT",
        "apigateway:POST",
        "apigateway:PATCH"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis",
        "arn:aws:apigateway:us-east-1::/apis/??????????",
        "arn:aws:apigateway:us-east-1::/apis/*/authorizers",
        "arn:aws:apigateway:us-east-1::/apis/*/authorizers/*"
      ],
      "Condition": {
        "ForAllValues:StringEqualsIfExists": {
          "apigateway:Request/AuthorizerType": [
            "REQUEST",
            "JWT"
          ]
        }
      }
```

```
    }
  ]
}
```

Richiedere che l'endpoint predefinito **execute-api** sia disabilitato

Questa policy di esempio consente agli utenti di creare, aggiornare o importare un'API, con il requisito che `DisableExecuteApiEndpoint` sia `true`. Quando `DisableExecuteApiEndpoint` è `true`, i client non possono utilizzare l'endpoint `execute-api` predefinito per richiamare un'API.

Usiamo la condizione `BoolIfExists` per gestire una chiamata per aggiornare un'API che non ha la chiave di condizione `DisableExecuteApiEndpoint` popolata. Quando un utente tenta di creare o importare un'API, la chiave di condizione `DisableExecuteApiEndpoint` viene sempre popolata.

Poiché la `apis/*` risorsa acquisisce anche risorse secondarie come autorizzatori o metodi, la estendiamo esplicitamente solo con una dichiarazione. APIs `Deny`

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "DisableExecuteApiEndpoint",
      "Effect": "Allow",
      "Action": [
        "apigateway:PATCH",
        "apigateway:POST",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis",
        "arn:aws:apigateway:us-east-1::/apis/*"
      ],
      "Condition": {
        "BoolIfExists": {
          "apigateway:Request/DisableExecuteApiEndpoint": true
        }
      }
    },
    {
```

```
      "Sid": "ScopeDownToJustApis",
      "Effect": "Deny",
      "Action": [
        "apigateway:PATCH",
        "apigateway:POST",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis/*/*"
      ]
    }
  ]
}
```

Consenti agli utenti di creare o aggiornare solo REST privati APIs

Questa policy di esempio utilizza le chiavi di condizione per richiedere che un utente crei solo `PRIVATE` APIs e per impedire aggiornamenti che potrebbero modificare un'API da `PRIVATE` un altro tipo, ad esempio`REGIONAL`.

Utilizziamo `ForAllValues` per richiedere che ogni `EndpointType` aggiunto a un'API sia `PRIVATE`. Utilizziamo una chiave di condizione delle risorse per consentire qualsiasi aggiornamento a un'API fino a quando è `PRIVATE`. `ForAllValues` si applica soltanto quando è presente una chiave di condizione.

Utilizziamo il non-greedy matcher (?) per abbinarlo esplicitamente all'API per impedire l'utilizzo di risorse non API IDs come gli autorizzatori.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ScopePutToPrivateApis",
            "Effect": "Allow",
            "Action": [
                "apigateway:PUT"
            ],
            "Resource": [
                "arn:aws:apigateway:us-east-1::/restapis",
```

```
                "arn:aws:apigateway:us-east-1::/restapis/??????????"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "apigateway:Resource/EndpointType": "PRIVATE"
                }
            }
        },
        {
            "Sid": "ScopeToPrivateApis",
            "Effect": "Allow",
            "Action": [
                "apigateway:DELETE",
                "apigateway:PATCH",
                "apigateway:POST"
            ],
            "Resource": [
                "arn:aws:apigateway:us-east-1::/restapis",
                "arn:aws:apigateway:us-east-1::/restapis/??????????"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "apigateway:Request/EndpointType": "PRIVATE",
                    "apigateway:Resource/EndpointType": "PRIVATE"
                }
            }
        },
        {
            "Sid": "AllowResourcePolicyUpdates",
            "Effect": "Allow",
            "Action": [
                "apigateway:UpdateRestApiPolicy"
            ],
            "Resource": [
                "arn:aws:apigateway:us-east-1::/restapis/*"
            ]
        }
    ]
}
```

Richiedere che i route API abbiano l'autorizzazione

Questa policy fa sì che i tentativi di creare o aggiornare un route (anche attraverso l'importazione) non riescono se il route non dispone di autorizzazione. `ForAnyValue` restituisce false se la chiave non è presente, ad esempio quando un route non viene creato o aggiornato. Usiamo `ForAnyValue` perché più route possono essere creati attraverso l'importazione.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowUpdatesOnApisAndRoutes",
      "Effect": "Allow",
      "Action": [
        "apigateway:POST",
        "apigateway:PATCH",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis",
        "arn:aws:apigateway:us-east-1::/apis/??????????",
        "arn:aws:apigateway:us-east-1::/apis/*/routes",
        "arn:aws:apigateway:us-east-1::/apis/*/routes/*"
      ]
    },
    {
      "Sid": "DenyUnauthorizedRoutes",
      "Effect": "Deny",
      "Action": [
        "apigateway:POST",
        "apigateway:PATCH",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis",
        "arn:aws:apigateway:us-east-1::/apis/*"
      ],
      "Condition": {
        "ForAnyValue:StringEqualsIgnoreCase": {
          "apigateway:Request/RouteAuthorizationType": "NONE"
```

```
        }
      }
    }
  ]
}
```

Impedisci a un utente di creare o aggiornare un collegamento VPC

Questa policy impedisce a un utente di creare o aggiornare un collegamento VPC. Un collegamento VPC consente di esporre risorse all'interno di Amazon VPC ai client esterni a VPC.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "DenyVPCLink",
      "Effect": "Deny",
      "Action": [
        "apigateway:POST",
        "apigateway:PUT",
        "apigateway:PATCH"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/vpclinks",
        "arn:aws:apigateway:us-east-1::/vpclinks/*"
      ]
    }
  ]
}
```

Policy di esempio per l'utilizzo delle regole di routing

I seguenti esempi di policy mostrano come utilizzare le chiavi di RoutingRule condizione per controllare in che modo gli utenti possono indirizzare il traffico dai loro nomi di dominio personalizzati al loro REST. APIs È possibile utilizzare questi esempi per creare policy granulari relative al tipo di

regole di routing che gli utenti possono stabilire. Per ulteriori informazioni, consulta the section called “Regole di routing”.

Impedire a un utente di modificare il modo in cui un nome di dominio personalizzato instrada una richiesta

Questa policy impedisce a un utente di creare o aggiornare `BasePathMapping`, `ApiMapping` o `RoutingRule`. Tutte queste risorse potrebbero modificare il modo in cui un nome di dominio personalizzato indirizza le richieste APIs.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAccessBasePathMappingsApiMappingsRoutingRules",
      "Effect": "Deny",
      "Action": "apigateway:*",
      "Resource": [
        "arn:aws:apigateway:us-east-1::/domainnames/example.com/basepathmappings/
*",
        "arn:aws:apigateway:us-east-1::/domainnames/example.com/apimappings/*",
        "arn:aws:apigateway:us-east-1:111122223333:/domainnames/example.com/
routingrules/*"
      ]
    }
  ]
}
```

Consentire a un utente di aggiornare una regola di routing per determinate priorità

Questa policy consente all’utente di aggiornare una regola di routing solo con un valore di priorità compreso tra 1001 e 2000. È possibile utilizzare questa regola per separare le regole di produzione dalle regole con priorità inferiore e quindi consentire agli utenti di modificare le regole con priorità inferiore senza influire sulle regole di produzione.

JSON

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "UpdatingRoutingRulePriorityBetween1001And2000",
      "Effect": "Allow",
      "Action": "apigateway:UpdateRoutingRule",
      "Resource": "arn:aws:apigateway:us-east-1:111122223333:/domainnames/
example.com/routingrules/*",
      "Condition": {
        "NumericGreaterThanEquals": {
          "apigateway:Resource/Priority": 1001,
          "apigateway:Request/Priority": 1001
        },
       "NumericLessThanEquals": {
          "apigateway:Resource/Priority": 2000,
          "apigateway:Request/Priority": 2000
        }
      }
    }
  ]
}
```

Consentire a un utente di aggiornare una regola di routing o la mappatura percorso di base per un determinato valore del percorso di base

Questa policy consente all'utente di aggiornare solo una mappatura percorso di base per qualsiasi percorso di base che inizia con `orders` o di aggiornare una regola di routing corrispondente a un percorso di base che inizia con `orders`. In questa policy, un utente può aggiornare una mappatura percorso di base o una regola di routing per `orders/create` o `orders123`, ma non per `payment/orders`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
      {
        "Sid": "AllowUpdateRoutingRuleUnderPathOrders",
        "Effect": "Allow",
        "Action": "apigateway:UpdateRoutingRule",
```

```
        "Resource": "arn:aws:apigateway:us-east-1:111122223333:/domainnames/
example.com/routingrules/*",
        "Condition": {
            "ForAllValues:StringLike": {
                "apigateway:Request/ConditionBasePaths": ["orders*"],
                "apigateway:Resource/ConditionBasePaths": ["orders*"]
            },
            "Null":{
                 "apigateway:Request/ConditionBasePaths":"false",
                 "apigateway:Resource/ConditionBasePaths":"false"
          }
        }
      }
  ]
}
```

Consentire a un utente di aggiornare la modalità di routing su valori specifici

Questa policy consente all'utente di aggiornare la modalità di routing solo su `API_MAPPING_ONLY` e `ROUTING_RULE_THEN_API_MAPPING`. Per ulteriori informazioni sulla modalità di routing, consultare the section called "Impostazione della modalità di routing per il nome di dominio personalizzato".

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
        {
           "Sid": "AllowUpdateRoutingModeToAnythingWithApiMapping",
           "Effect": "Allow",
           "Action": ["apigateway:PATCH"],
           "Resource": "arn:aws:apigateway:us-east-1::/domainnames/example.com",
              "Condition": {
                "StringLike": {
                    "apigateway:Request/RoutingMode":"*API_MAPPING*"
                }
           }
       }
    ]
}
```

Esempi di policy basate su risorse di Amazon API Gateway

Per esempi di policy basate su risorse, consult the section called “Esempi di policy delle risorse API Gateway”.

Risoluzione dei problemi di identità e accesso di Amazon API Gateway

Utilizzare le informazioni seguenti per diagnosticare e risolvere i problemi comuni che possono verificarsi durante l'utilizzo di API Gateway e IAM.

Argomenti

- Non sono autorizzato a eseguire un'operazione in API Gateway
- Non sono autorizzato a eseguire iam: PassRole
- Voglio consentire a persone esterne al mio AWS account di accedere alle mie risorse API Gateway

Non sono autorizzato a eseguire un'operazione in API Gateway

Se ricevi un errore che indica che non sei autorizzato a eseguire un’operazione, le tue policy devono essere aggiornate per poter eseguire l’operazione.

L’errore di esempio seguente si verifica quando l’utente IAM `mateojackson` prova a utilizzare la console per visualizzare i dettagli relativi a una risorsa *`my-example-widget`* fittizia ma non dispone di autorizzazioni `apigateway:`*`GetWidget`* fittizie.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 apigateway:GetWidget on resource: my-example-widget because no identity-based policy
 allows the GetWidget action
```

In questo caso, la policy per l’utente `mateojackson` deve essere aggiornata per consentire l’accesso alla risorsa *`my-example-widget`* utilizzando l’azione `apigateway:`*`GetWidget`*.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L’amministratore è la persona che ti ha fornito le credenziali di accesso.

Non sono autorizzato a eseguire iam: PassRole

Se si riceve un errore che indica che non si è autorizzati a eseguire l'operazione `iam:PassRole`, è necessario aggiornare le policy per poter passare un ruolo ad API Gateway.

Alcuni Servizi AWS consentono di passare un ruolo esistente a quel servizio invece di creare un nuovo ruolo di servizio o un ruolo collegato al servizio. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per trasmettere il ruolo al servizio.

L'errore di esempio seguente si verifica quando un utente IAM denominato `marymajor` cerca di utilizzare la console per eseguire un'operazione in API Gateway. Tuttavia, l'azione richiede che il servizio disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone delle autorizzazioni per trasmettere il ruolo al servizio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In questo caso, le policy di Mary devono essere aggiornate per poter eseguire l'operazione `iam:PassRole`.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Voglio consentire a persone esterne al mio AWS account di accedere alle mie risorse API Gateway

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per i servizi che supportano politiche basate sulle risorse o liste di controllo degli accessi (ACLs), puoi utilizzare tali politiche per concedere alle persone l'accesso alle tue risorse.

Per maggiori informazioni, consulta gli argomenti seguenti:

- Per sapere se API Gateway supporta queste caratteristiche, consultare Come funziona Amazon API Gateway con IAM.
- Per scoprire come fornire l'accesso alle tue risorse attraverso Account AWS le risorse di tua proprietà, consulta Fornire l'accesso a un utente IAM in un altro Account AWS di tua proprietà nella IAM User Guide.
- Per scoprire come fornire l'accesso alle tue risorse a terze parti Account AWS, consulta Fornire l'accesso a soggetti Account AWS di proprietà di terze parti nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consulta Fornire l'accesso a utenti autenticati esternamente (federazione delle identità) nella Guida per l'utente IAM.

- Per informazioni sulle differenze di utilizzo tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Accesso a risorse multi-account in IAM nella Guida per l'utente IAM.

Utilizzo dei ruoli collegati ai servizi per API Gateway

Amazon API Gateway utilizza AWS Identity and Access Management ruoli collegati ai servizi (IAM). Un ruolo collegato ai servizi è un tipo di ruolo specifico di IAM collegato direttamente ad API Gateway. I ruoli collegati ai servizi sono predefiniti da API Gateway e includono tutte le autorizzazioni richieste dal servizio per chiamare altri AWS servizi per tuo conto.

Un ruolo collegato ai servizi semplifica la configurazione di API Gateway perché permette di evitare l'aggiunta manuale delle autorizzazioni necessarie. API Gateway definisce le autorizzazioni dei relativi ruoli associati ai servizi e, salvo diversamente definito, solo API Gateway potrà assumere i propri ruoli. Le autorizzazioni definite includono la policy di attendibilità e la policy delle autorizzazioni che non può essere collegata a nessun'altra entità IAM.

È possibile eliminare un ruolo collegato ai servizi solo dopo aver eliminato le risorse correlate. Questa procedura protegge le risorse di API Gateway perché impedisce la rimozione involontaria delle autorizzazioni di accesso alle risorse.

Per informazioni su altri servizi che supportano i ruoli collegati ai servizi, consulta AWS Services That Work with IAM e cerca i servizi che riportano Yes (Sì) nella colonna Service-Linked Role (Ruolo associato ai servizi). Scegli un link Yes (Sì) per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Autorizzazioni del ruolo collegato ai servizi per API Gateway

API Gateway utilizza il ruolo collegato al servizio denominato AWSServiceRoleForAPIGateway—Consente ad API Gateway di accedere a Elastic Load Balancing, Amazon Data Firehose e altre risorse di servizio per tuo conto.

Il ruolo AWSService RoleFor APIGateway collegato al servizio prevede che i seguenti servizi assumano il ruolo:

- `ops.apigateway.amazonaws.com`

La policy delle autorizzazioni del ruolo consente ad API Gateway di eseguire le seguenti operazioni sulle risorse specificate:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:AddListenerCertificates",
                "elasticloadbalancing:RemoveListenerCertificates",
                "elasticloadbalancing:ModifyListener",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:DescribeLoadBalancers",
                "xray:PutTraceSegments",
                "xray:PutTelemetryRecords",
                "xray:GetSamplingTargets",
                "xray:GetSamplingRules",
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries",
                "servicediscovery:DiscoverInstances"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "firehose:DescribeDeliveryStream",
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Resource": "arn:aws:firehose:*:*:deliverystream/amazon-apigateway-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "acm:DescribeCertificate",
                "acm:GetCertificate"
            ],
```

```
            "Resource": "arn:aws:acm:*:*:certificate/*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateNetworkInterfacePermission",
            "Resource": "arn:aws:ec2:*:*:network-interface/*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:*:*:network-interface/*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": [
                        "Owner",
                        "VpcLinkId"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DeleteNetworkInterface",
                "ec2:AssignPrivateIpAddresses",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeNetworkInterfaceAttribute",
                "ec2:DescribeVpcs",
                "ec2:DescribeNetworkInterfacePermissions",
                "ec2:UnassignPrivateIpAddresses",
                "ec2:DescribeSubnets",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "servicediscovery:GetNamespace",
            "Resource": "arn:aws:servicediscovery:*:*:namespace/*"
```

```
        },
        {
            "Effect": "Allow",
            "Action": "servicediscovery:GetService",
            "Resource": "arn:aws:servicediscovery:*:*:service/*"
        }
    ]
}
```

Per consentire a un'entità IAM (come un utente, un gruppo o un ruolo) di creare, modificare o eliminare un ruolo collegato ai servizi devi configurare le relative autorizzazioni. Per ulteriori informazioni, consulta Autorizzazioni del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Creazione di un ruolo collegato ai servizi per API Gateway

Non hai bisogno di creare manualmente un ruolo collegato ai servizi. Quando crei un'API, un nome di dominio personalizzato o un link VPC nell' AWS API Console di gestione AWS, API Gateway crea automaticamente il ruolo collegato al servizio. AWS CLI

Se elimini questo ruolo collegato al servizio, è possibile ricrearlo seguendo lo stesso processo utilizzato per ricreare il ruolo nell'account. Quando si crea un'API, un nome di dominio personalizzato o un collegamento VPC, API Gateway crea nuovamente il ruolo collegato ai servizi per l'utente.

Modifica di un ruolo collegato ai servizi per API Gateway

API Gateway non consente di modificare il ruolo AWSService RoleFor APIGateway collegato al servizio. Dopo aver creato un ruolo collegato al servizio, non è possibile modificarne il nome, perché potrebbero farvi riferimento diverse entità. È possibile tuttavia modificarne la descrizione utilizzando IAM. Per ulteriori informazioni, consulta la sezione Modifica di un ruolo collegato ai servizi nella Guida per l'utente di IAM.

Eliminazione di un ruolo collegato ai servizi per API Gateway

Se non è più necessario utilizzare una caratteristica o un servizio che richiede un ruolo collegato ai servizi, ti consigliamo di eliminare quel ruolo. In questo modo non sarà più presente un'entità non utilizzata che non viene monitorata e gestita attivamente. Tuttavia, è necessario effettuare la pulizia delle risorse associate al ruolo collegato ai servizi prima di poterlo eliminare manualmente.

> **Note**
>
> Se il servizio API Gateway utilizza tale ruolo quando tenti di eliminare le risorse, è possibile che l'eliminazione non abbia esito positivo. In questo caso, attendi alcuni minuti e quindi ripeti l'operazione.

Per eliminare le risorse API Gateway utilizzate da AWSService RoleFor APIGateway

1. Apri la console API Gateway all'indirizzo https://console.aws.amazon.com/apigateway/.
2. Passare all'API, al nome di dominio personalizzato o al collegamento VPC che utilizza il ruolo collegato al servizio.
3. Utilizza la console per eliminare una risorsa.
4. Ripeti la procedura per eliminare tutti APIs i nomi di dominio personalizzati o i link VPC che utilizzano il ruolo collegato al servizio.

Per eliminare manualmente il ruolo collegato ai servizi mediante IAM

Utilizza la console IAM AWS CLI, l'o l' AWS API per eliminare il ruolo collegato al AWSService RoleFor APIGateway servizio. Per ulteriori informazioni, consulta Eliminazione del ruolo collegato ai servizi nella Guida per l'utente di IAM.

Regioni supportate per i ruoli collegati ai servizi di API Gateway

API Gateway supporta l'utilizzo di ruoli collegati ai servizi in tutte le regioni in cui il servizio è disponibile. Per ulteriori informazioni, consulta AWS Endpoint del servizio.

Aggiornamenti API Gateway alle policy AWS gestite

Visualizza i dettagli sugli aggiornamenti delle politiche AWS gestite per API Gateway da quando questo servizio ha iniziato a tenere traccia di queste modifiche. Per gli avvisi automatici sulle modifiche apportate a questa pagina, sottoscrivere il feed RSS nella pagina della cronologia dei documenti di API Gateway.

Modifica	Descrizione	Data
Aggiunto supporto `acm:GetCertificate`	La policy `AWSServiceRoleForAPIGateway`	12 luglio 2021

Modifica	Descrizione	Data
per la policy `AWSServiceRoleForAPIGateway` .	ora include l'autorizzazione per chiamare l'operazione API `GetCertificate` ACM.	
API Gateway ha iniziato a monitorare le modifiche	API Gateway ha iniziato a tracciare le modifiche per le sue politiche AWS gestite.	12 luglio 2021

Logging e monitoraggio in Amazon API Gateway

Il monitoraggio è una parte importante per mantenere l'affidabilità, la disponibilità e le prestazioni di API Gateway e delle soluzioni AWS. È consigliabile raccogliere dati di monitoraggio da tutte le parti della soluzione AWS per eseguire più facilmente il debug dei fallimenti in più punti nel caso si verifichino. AWS fornisce diversi strumenti per il monitoraggio delle risorse API Gateway e di altre risorse, nonché la risposta a potenziali incidenti.

Amazon CloudWatch Logs

Per eseguire il debug di problemi correlati all'esecuzione di richieste o all'accesso client all'API, puoi abilitare CloudWatch Logs per tenere traccia delle chiamate API. Per ulteriori informazioni, consulta the section called “CloudWatch registri”.

Allarmi di Amazon CloudWatch

Utilizzando gli allarmi di CloudWatch, è possibile osservare un singolo parametro per il periodo di tempo specificato. Se il parametro supera una determinata soglia, viene inviata una notifica a un argomento Amazon Simple Notification Service o alla policy AWS Auto Scaling. Gli allarmi CloudWatch non richiamano azioni quando una metrica si trova in uno stato particolare. È necessario invece cambiare lo stato e mantenerlo per un numero di periodi specificato. Per ulteriori informazioni, consulta the section called “CloudWatch metriche”.

Registrazione dei log degli accessi a Firehose

Per eseguire il debug dei problemi correlati all'accesso dei client all'API, è possibile abilitare la registrazione dei log delle chiamate API in Firehose. Per ulteriori informazioni, consulta the section called “Firehose”.

AWS CloudTrail

CloudTrail fornisce un record delle operazioni eseguite da un utente, un ruolo o un servizio AWS in API Gateway. Le informazioni raccolte da CloudTrail consentono di determinare la richiesta effettuata ad API Gateway, l'indirizzo IP da cui è partita la richiesta, l'autore della richiesta, il momento in cui è stata eseguita e altri dettagli. Per ulteriori informazioni, consulta the section called “Lavorare con CloudTrail”.

AWS X-Ray

X-Ray è un servizio AWS che raccoglie i dati sulle richieste che l'applicazione serve e li utilizza per creare una mappa del servizio da utilizzare per identificare i problemi con l'applicazione e le opportunità di ottimizzazione. Per ulteriori informazioni, consulta the section called “Configurare AWS X-Ray”.

AWS Config

AWS Config fornisce una panoramica dettagliata della configurazione delle risorse AWS nel tuo account. Puoi vedere in che modo le risorse sono correlate, ottenere una cronologia delle modifiche alla configurazione ed esaminare come cambiano le relazioni e le configurazioni nel tempo. Puoi utilizzare AWS Config per definire regole che valutano le configurazioni delle risorse per la conformità dei dati. Le regole AWS Config rappresentano le impostazioni di configurazione ideali per le risorse API Gateway. Se una risorsa viola una regola e viene contrassegnata come non conforme, AWS Config può avvisare l'utente utilizzando un argomento Amazon Simple Notification Service (Amazon SNS). Per informazioni dettagliate, consulta the section called “Lavorare con AWS Config”.

Registrazione delle chiamate API di Amazon API Gateway utilizzando AWS CloudTrail

Amazon API Gateway è integrato con AWS CloudTrail, un servizio che fornisce un registro delle azioni intraprese da un utente, ruolo o un Servizio AWS. CloudTrail acquisisce tutte le chiamate API REST per il servizio API Gateway come eventi. Le chiamate acquisite includono chiamate dalla console API Gateway e chiamate di codice al servizio API Gateway APIs. Utilizzando le informazioni raccolte da CloudTrail, è possibile determinare la richiesta effettuata a API Gateway, l'indirizzo IP da cui è stata effettuata la richiesta, quando è stata effettuata e dettagli aggiuntivi.

> **Note**
>
> TestInvokeAuthorizere non TestInvokeMethodhanno effettuato l'accesso. CloudTrail

Ogni evento o voce di log contiene informazioni sull'utente che ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con le credenziali utente root o utente.
- Se la richiesta è stata effettuata per conto di un utente del Centro identità IAM.
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro Servizio AWS.

CloudTrail è attivo nel tuo account Account AWS quando crei l'account e hai automaticamente accesso alla cronologia degli CloudTrail eventi. La cronologia CloudTrail degli eventi fornisce un record visualizzabile, ricercabile, scaricabile e immutabile degli ultimi 90 giorni di eventi di gestione registrati in un. Regione AWSPer ulteriori informazioni, consulta Lavorare con la cronologia degli CloudTrail eventi nella Guida per l'utente.AWS CloudTrail Non sono CloudTrail previsti costi per la visualizzazione della cronologia degli eventi.

Per una registrazione continua degli eventi degli Account AWS ultimi 90 giorni, crea un trail o un data store di eventi CloudTrailLake.

CloudTrail sentieri

Un trail consente di CloudTrail inviare file di log a un bucket Amazon S3. Tutti i percorsi creati utilizzando il Console di gestione AWS sono multiregionali. È possibile creare un trail per una singola Regione o per più Regioni tramite AWS CLI. La creazione di un percorso multiregionale è consigliata in quanto consente di registrare l'intera attività del proprio Regioni AWS account. Se si crea un trail per una singola Regione, è possibile visualizzare solo gli eventi registrati nella Regione AWS del trail. Per ulteriori informazioni sui trail, consulta Creating a trail for your Account AWS e Creating a trail for an organization nella Guida per l'utente di AWS CloudTrail .

Puoi inviare gratuitamente una copia dei tuoi eventi di gestione in corso al tuo bucket Amazon S3 CloudTrail creando un percorso, tuttavia ci sono costi di storage di Amazon S3. Per ulteriori informazioni sui CloudTrail prezzi, consulta la pagina Prezzi.AWS CloudTrail Per informazioni sui prezzi di Amazon S3, consulta Prezzi di Amazon S3.

CloudTrail Archivi di dati sugli eventi di Lake

CloudTrail Lake ti consente di eseguire query basate su SQL sui tuoi eventi. CloudTrail Lake converte gli eventi esistenti in formato JSON basato su righe in formato Apache ORC. ORC è un formato di archiviazione a colonne ottimizzato per il recupero rapido dei dati. Gli eventi vengono aggregati in archivi di dati degli eventi, che sono raccolte di eventi immutabili basate sui criteri selezionati applicando i selettori di eventi avanzati. I selettori applicati a un archivio di dati degli eventi controllano quali eventi persistono e sono disponibili per l'esecuzione della query. Per ulteriori informazioni su CloudTrail Lake, consulta Working with AWS CloudTrail Lake nella Guida per l'utente.AWS CloudTrail

CloudTrail Gli archivi e le richieste di dati sugli eventi di Lake comportano dei costi. Quando crei un datastore di eventi, scegli l'opzione di prezzo da utilizzare per tale datastore. L'opzione di prezzo determina il costo per l'importazione e l'archiviazione degli eventi, nonché il periodo di conservazione predefinito e quello massimo per il datastore di eventi. Per ulteriori informazioni sui CloudTrail prezzi, consulta Prezzi.AWS CloudTrail

Eventi di gestione dell'API Gateway in CloudTrail

Gli eventi di gestione forniscono informazioni sulle operazioni di gestione eseguite sulle risorse di Account AWS. Queste operazioni sono definite anche operazioni del piano di controllo (control-plane). Per impostazione predefinita, CloudTrail registra gli eventi di gestione.

Amazon API Gateway registra tutte le azioni di API Gateway come eventi di gestione, ad eccezione di TestInvokeAuthorizere TestInvokeMethod. Per un elenco delle azioni di Amazon API Gateway a cui API Gateway effettua il log CloudTrail, consulta l'Amazon API Gateway API Reference.

Esempio di evento di Gateway API

Un evento rappresenta una singola richiesta proveniente da qualsiasi fonte e include informazioni sull'operazione API richiesta, la data e l'ora dell'operazione, i parametri della richiesta e così via. CloudTrail i file di registro non sono una traccia stack ordinata delle chiamate API pubbliche, quindi gli eventi non vengono visualizzati in un ordine specifico.

L'esempio seguente mostra un CloudTrail evento che dimostra l'`GetResource`azione API Gateway:

```
{
    Records: [
        {
```

```
            eventVersion: "1.03",
            userIdentity: {
                type: "Root",
                principalId: "AKIAI44QH8DHBEXAMPLE",
                arn: "arn:aws:iam::123456789012:root",
                accountId: "123456789012",
                accessKeyId: "AKIAIOSFODNN7EXAMPLE",
                sessionContext: {
                    attributes: {
                        mfaAuthenticated: "false",
                        creationDate: "2015-06-16T23:37:58Z"
                    }
                }
            },
            eventTime: "2015-06-17T00:47:28Z",
            eventSource: "apigateway.amazonaws.com",
            eventName: "GetResource",
            awsRegion: "us-east-1",
            sourceIPAddress: "203.0.113.11",
            userAgent: "example-user-agent-string",
            requestParameters: {
                restApiId: "3rbEXAMPLE",
                resourceId: "5tfEXAMPLE",
                template: false
            },
            responseElements: null,
            requestID: "6d9c4bfc-148a-11e5-81b6-7577cEXAMPLE",
            eventID: "4d293154-a15b-4c33-9e0a-ff5eeEXAMPLE",
            readOnly: true,
            eventType: "AwsApiCall",
            recipientAccountId: "123456789012"
        },
        ... additional entries ...
    ]
}
```

Per informazioni sul contenuto dei CloudTrail record, consulta il contenuto dei CloudTrail record nella Guida per l'AWS CloudTrail utente.

Monitoraggio della configurazione dell'API API Gateway con AWS Config

Puoi usare AWS Config per registrare le modifiche di configurazione apportate alle risorse API Gateway API e inviare notifiche in base alle modifiche alle risorse. Conservare la cronologia delle

modifiche eseguite sulla configurazione delle risorse di API Gateway è utile per la risoluzione dei problemi operativi, la revisione e i casi d'uso relativi alla conformità.

AWS Config può tenere traccia delle modifiche a:

- Configurazione delle fasi API, ad esempio:
  - Impostazioni del cluster di cache
  - Impostazioni di throttling
  - Impostazioni dei log degli accessi
  - Set di distribuzione attivo nella fase
- Configurazione dell'API, ad esempio:
  - Configurazione dell'endpoint
  - Versione
  - protocol
  - tags

Inoltre, la Regole di AWS Config funzionalità consente di definire regole di configurazione e rilevare, tracciare e avvisare automaticamente le violazioni di tali regole. Tracciando le modifiche a queste proprietà di configurazione delle risorse, puoi anche creare AWS Config regole che generano modifiche per le tue risorse API Gateway e testare le configurazioni delle risorse rispetto alle best practice.

Puoi abilitarlo AWS Config nel tuo account utilizzando la console o il AWS Config . AWS CLI Seleziona i tipi di risorsa per cui vuoi monitorare le modifiche. Se in precedenza hai configurato AWS Config per registrare tutti i tipi di risorse, queste risorse API Gateway verranno registrate automaticamente nel tuo account. Il supporto per Amazon API Gateway in AWS Config è disponibile in tutte le regioni AWS pubbliche e AWS GovCloud (US). Per l'elenco completo delle Regioni supportate, consulta Endpoint e quote Amazon API Gateway nei riferimenti generali Riferimenti generali di AWS.

Argomenti

- Tipi di risorse supportati
- Configurazione AWS Config
- Configurazione AWS Config per registrare le risorse API Gateway
- Visualizzazione dei dettagli di configurazione dell'API Gateway nella AWS Config console

- Valutazione delle risorse API Gateway mediante regole AWS Config

Tipi di risorse supportati

I seguenti tipi di risorse API Gateway sono integrati AWS Config e documentati in AWS Config Supported Resource Types and AWS Resource Relationships:

- `AWS::ApiGatewayV2::Api`(WebSocket e API HTTP)
- `AWS::ApiGateway::RestApi` (API REST)
- `AWS::ApiGatewayV2::Stage`(WebSocket e fase HTTP API)
- `AWS::ApiGateway::Stage` (fase API REST)

Per ulteriori informazioni in merito AWS Config, consulta la Guida per AWS Config gli sviluppatori. Per informazioni sui prezzi, consulta la pagina di informazioni sui prezzi di AWS Config.

> **Important**
>
> Se modifichi una delle proprietà dell'API seguenti dopo avere distribuito l'API, devi ridistribuire l'API per propagare le modifiche. Altrimenti, vedrete le modifiche agli attributi nella AWS Config console, ma le impostazioni delle proprietà precedenti rimarranno valide; il comportamento di runtime dell'API rimarrà invariato.
>
> - **`AWS::ApiGateway::RestApi`** – `binaryMediaTypes`, `minimumCompressionSize`, `apiKeySource`
> - **`AWS::ApiGatewayV2::Api`** – `apiKeySelectionExpression`

Configurazione AWS Config

Per la configurazione iniziale AWS Config, consulta i seguenti argomenti nella Guida per gli AWS Config sviluppatori.

- Configurazione AWS Config con la console
- Configurazione AWS Config con AWS CLI

Configurazione AWS Config per registrare le risorse API Gateway

Per impostazione predefinita, AWS Config registra le modifiche alla configurazione per tutti i tipi di risorse regionali supportate che rileva nella regione in cui è in esecuzione l'ambiente. È possibile personalizzare AWS Config in modo da registrare le modifiche solo per tipi di risorse specifici o le modifiche alle risorse globali.

Per maggiori informazioni sulle risorse regionali e globali e su come personalizzare la AWS Config configurazione, vedi Selezione dei AWS Config record di risorse.

Visualizzazione dei dettagli di configurazione dell'API Gateway nella AWS Config console

Puoi utilizzare la AWS Config console per cercare le risorse di API Gateway e ottenere dettagli attuali e storici sulle relative configurazioni. La procedura seguente descrive come trovare informazioni su un'API di API Gateway.

Per trovare una risorsa API Gateway nella AWS console di configurazione

1. Aprire la console AWS Config.
2. Scegliere Resources (Risorse).
3. Nella pagina Resource inventory (Inventario risorse), scegliere Resources (Risorse).
4. Apri il menu Tipo di risorsa, scorri fino a APIGateway o APIGateway V2, quindi scegli uno o più tipi di risorse API Gateway.
5. Scegliere Look up (Cerca).
6. Scegli un ID di risorsa nell'elenco di risorse AWS Config visualizzato. AWS Config visualizza i dettagli di configurazione e altre informazioni sulla risorsa selezionata.
7. Per visualizzare i dettagli completi della configurazione registrata, scegliere View Details (Visualizza dettagli).

Per ulteriori informazioni su come trovare una risorsa e visualizzare le informazioni in questa pagina, consulta Visualizzazione AWS delle configurazioni e della cronologia delle risorse nella Guida per gli AWS Config sviluppatori.

Valutazione delle risorse API Gateway mediante regole AWS Config

Puoi creare AWS Config regole che rappresentano le impostazioni di configurazione ideali per le tue risorse API Gateway. È possibile utilizzare regole AWS Config gestite predefinite o definire regole

personalizzate. AWS Config monitora continuamente le modifiche alla configurazione delle risorse per determinare se tali modifiche violano alcune delle condizioni delle regole. La AWS Config console mostra lo stato di conformità delle regole e delle risorse.

Se una risorsa viola una regola e viene contrassegnata come non conforme, può AWS Config avvisarti utilizzando un argomento della Amazon Simple Notification Service Developer Guide (Amazon SNS). Per utilizzare i dati a livello di programmazione in questi avvisi AWS Config , usa una coda Amazon Simple Queue Service (Amazon SQS) come l'endpoint di notifica per l'argomento Amazon SNS.

Per ulteriori informazioni sulla configurazione e l'uso di regole, consulta Valutazione delle risorse con le regole nella Guida per gli sviluppatori di AWS Config.

Convalida della conformità per Amazon API Gateway

La nostra nuova esperienza AWS di iscrizione non è progettata per carichi di lavoro regolamentati. Se stai utilizzando la nostra nuova esperienza di AWS registrazione, ma desideri utilizzarla AWS per carichi di lavoro regolamentati, puoi iscriverti a AWS (avanzato) o attivare funzionalità avanzate per il tuo ambiente. AWS

Per sapere se un Servizio AWS rientra nell'ambito di specifici programmi di conformità, consulta Servizi AWS la sezione Scope by Compliance Program e scegli il programma di conformità che ti interessa. Per informazioni generali, vedere Programmi di AWS conformità Programmi di

È possibile scaricare i report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, vedere Scaricamento dei report in AWS Artifact .

La responsabilità dell'utente in materia di conformità durante l'utilizzo Servizi AWS è determinata dalla sensibilità dei dati, dagli obiettivi di conformità dell'azienda e dalle leggi e dai regolamenti applicabili. Per ulteriori informazioni sulla responsabilità dell'utente in materia di conformità durante l'utilizzo Servizi AWS, consulta AWS la Documentazione sulla sicurezza.

Resilienza in Amazon API Gateway

L'infrastruttura globale di AWS è basata su Regioni e zone di disponibilità AWS. Le Regioni AWS Le Regioni forniscono più zone di disponibilità fisicamente separate e isolate che sono connesse tramite reti altamente ridondanti, a bassa latenza e velocità di trasmissione effettiva elevata. Con le zone di disponibilità, puoi progettare e gestire applicazioni e database che eseguono automaticamente

il failover tra zone di disponibilità senza interruzioni. Le zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili rispetto alle infrastrutture a data center singolo o multiplo tradizionali.

In qualità di servizio regionale completamente gestito, Gateway API opera in più zone di disponibilità in ciascuna Regione, utilizzando la ridondanza delle zone di disponibilità per ridurre al minimo gli errori dell'infrastruttura come categoria di rischio di disponibilità. Gateway API è progettato per il ripristino automatico in caso di errore di una zona di disponibilità.

Per ulteriori informazioni sulle Regioni AWS e sulle zone di disponibilità, consulta Infrastruttura globale di AWS.

Per evitare che le API vengano sopraffatte da troppe richieste, API Gateway limita le richieste alle API. Nello specifico, API Gateway imposta un limite per il numero di richieste durante il funzionamento a regime e per i picchi di invio di richieste per tutte le API dell'account. È possibile configurare la limitazione personalizzata per le API. Per ulteriori informazioni, consulta Limita le richieste al tuo REST APIs per una migliore velocità di trasmissione in API Gateway.

È possibile utilizzare i controlli dell'integrità di Route 53 per controllare il failover DNS da un'API di API Gateway in una regione primaria a un'API di Gateway API in un'area secondaria. Per un esempio, consulta the section called “failover DNS”.

Sicurezza dell'infrastruttura in Amazon API Gateway

In quanto servizio gestito, Amazon API Gateway è protetto dalla sicurezza di rete AWS globale. Per informazioni sui servizi AWS di sicurezza e su come AWS protegge l'infrastruttura, consulta AWS Cloud Security. Per progettare il tuo AWS ambiente utilizzando le migliori pratiche per la sicurezza dell'infrastruttura, vedi Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Utilizzi chiamate API AWS pubblicate per accedere ad API Gateway attraverso la rete. I client devono supportare quanto segue:

- Transport Layer Security (TLS). È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Suite di cifratura con Perfect Forward Secrecy (PFS), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Puoi richiamare queste operazioni API da qualsiasi posizione di rete, ma API Gateway non supporta le policy di accesso basate sulle risorse che possono includere limitazioni sull'indirizzo IP di origine.

Puoi anche utilizzare policy basate sulle risorse per controllare l'accesso da endpoint Amazon Virtual Private Cloud (Amazon VPC) specifici o specifici. VPCs In effetti, questo isola l'accesso alla rete a una determinata risorsa API Gateway solo dal VPC specifico all'interno AWS della rete.

Analisi delle vulnerabilità in Amazon API Gateway

Configurazione e controllo IT sono una responsabilità condivisa tra AWSe te, il nostro cliente. Per ulteriori informazioni, consulta il modello di responsabilità condivisa di AWS.

Best practice di sicurezza in Amazon API Gateway

API Gateway fornisce una serie di caratteristiche di sicurezza che occorre valutare durante lo sviluppo e l'implementazione delle policy di sicurezza. Le seguenti best practice sono linee guida generali e non rappresentano una soluzione di sicurezza completa. Poiché queste best practice potrebbero non essere appropriate o sufficienti per l'ambiente, gestiscile come considerazioni utili anziché prescrizioni.

Implementazione dell'accesso con privilegi minimi

Utilizza le policy IAM per implementare l'accesso con privilegi minimi per creare, leggere, aggiornare o eliminare API Gateway. APIs Per ulteriori informazioni, consulta Gestione delle identità e degli accessi per Amazon API Gateway. API Gateway offre diverse opzioni per controllare l'accesso a APIs ciò che create. Per ulteriori informazioni, consulta Controlla e gestisci l'accesso a REST APIs in API Gateway, Controlla e gestisci l'accesso WebSocket APIs a API Gateway e Controllo degli accessi alle API HTTP con provider di autorizzazioni JWT in Gateway API.

Implementare la registrazione

Usa CloudWatch Logs o Amazon Data Firehose per registrare le richieste sul tuo. APIs Per ulteriori informazioni, consulta Monitoraggio delle REST API in Gateway API, Configurazione della registrazione per API WebSocket APIs Gateway e Configurare la registrazione per HTTP APIs in API Gateway.

Implementa gli CloudWatch allarmi Amazon

Utilizzando gli CloudWatch allarmi, controlli una singola metrica per un periodo di tempo specificato. Se la metrica supera una determinata soglia, viene inviata una notifica a un argomento o AWS Auto Scaling una politica di Amazon Simple Notification Service. CloudWatch gli allarmi non richiamano azioni quando una metrica si trova in uno stato particolare. È

necessario invece cambiare lo stato e mantenerlo per un numero di periodi specificato. Per ulteriori informazioni, consulta the section called “CloudWatch metriche”.

Abilita AWS CloudTrail

CloudTrail fornisce un registro delle azioni intraprese da un utente, un ruolo o un AWS servizio in API Gateway. Utilizzando le informazioni raccolte da CloudTrail, è possibile determinare la richiesta effettuata a API Gateway, l'indirizzo IP da cui è stata effettuata la richiesta, chi ha effettuato la richiesta, quando è stata effettuata e dettagli aggiuntivi. Per ulteriori informazioni, consulta the section called “Lavorare con CloudTrail”.

Abilita AWS Config

AWS Config fornisce una visualizzazione dettagliata della configurazione delle AWS risorse del tuo account. Puoi vedere in che modo le risorse sono correlate, ottenere una cronologia delle modifiche alla configurazione ed esaminare come cambiano le relazioni e le configurazioni nel tempo. È possibile utilizzare AWS Config per definire regole che valutano le configurazioni delle risorse per la conformità dei dati. AWS Config le regole rappresentano le impostazioni di configurazione ideali per le risorse API Gateway. Se una risorsa viola una regola e viene contrassegnata come non conforme, può AWS Config avvisarti utilizzando un argomento di Amazon Simple Notification Service (Amazon SNS). Per informazioni dettagliate, vedi the section called “Lavorare con AWS Config”.

Usa AWS Security Hub CSPM

Monitora l'utilizzo di API Gateway riguardo alle best practice di sicurezza utilizzando AWS Security Hub CSPM. Security Hub CSPM utilizza i controlli di sicurezza per valutare le configurazioni delle risorse e gli standard di sicurezza per aiutarti a rispettare vari framework di conformità. Per ulteriori informazioni sull'utilizzo di Security Hub CSPM per valutare le risorse di API Gateway, consulta i controlli di Amazon API Gateway nella Guida per l'AWS Security Hub utente.

Tagging delle risorse API Gateway

Un tag è un'etichetta di metadati che si assegna o che si AWS assegna a una risorsa. AWS Ogni tag è costituito da due parti:

- Una chiave del tag (ad esempio, `CostCenter`, `Environment` o `Project`). Le chiavi dei tag prevedono una distinzione tra lettere maiuscole e minuscole.
- Un campo facoltativo noto come valore del tag (ad esempio, `111122223333` o `Production`). Non specificare il valore del tag equivale a utilizzare una stringa vuota. Analogamente alle chiavi dei tag, i valori dei tag rilevano la distinzione tra maiuscole e minuscole.

I tag consentono di eseguire le seguenti operazioni:

- Controllare gli accessi alle risorse in base ai tag a loro assegnati. Puoi controllare gli accessi specificando chiavi e valori di tag nelle condizioni di una policy AWS Identity and Access Management (IAM). Per ulteriori informazioni sul controllo degli accessi basato su tag, consulta l'argomento relativo al controllo degli accessi mediante tag nella Guida per l'utente di IAM.
- Tieni traccia dei costi. AWS Attivi questi tag sulla Gestione costi e fatturazione AWS dashboard. AWS utilizza i tag per classificare i costi e fornirti un rapporto mensile sull'allocazione dei costi. Per ulteriori informazioni, consulta la pagina sull'utilizzo dei tag per l'allocazione dei costi nella Guida per l'utente di AWS Billing.
- Identifica e organizza le tue risorse. AWS Molti AWS servizi supportano l'etichettatura, quindi puoi assegnare lo stesso tag a risorse di servizi diversi per indicare che le risorse sono correlate. Ad esempio, è possibile assegnare a uno stadio API Gateway lo stesso tag assegnato a una regola CloudWatch Events.

Per suggerimenti sull'utilizzo dei tag, consulta il whitepaper AWS Tagging Strategies.

Nelle sezioni seguenti vengono fornite ulteriori informazioni sui tag di Amazon API Gateway.

Argomenti

- Risorse API Gateway a cui è possibile applicare tag
- Utilizzo dei tag per controllare l'accesso alle risorse REST API di Gateway API

Risorse API Gateway a cui è possibile applicare tag

I tag possono essere impostati sulle seguenti risorse API HTTP o WebSocket API nell'API Amazon API Gateway V2:

- `Api`
- `DomainName`
- `Stage`
- `VpcLink`

Inoltre, i tag possono essere impostati sulle seguenti risorse di API REST nell'API di Amazon API Gateway V1:

- `ApiKey`
- `ClientCertificate`
- `DomainName`
- `DomainNameAccessAssociation`
- `RestApi`
- `Stage`
- `UsagePlan`
- `VpcLink`

I tag non possono essere impostati direttamente su altre risorse. Tuttavia, nell'API di Amazon API Gateway V1, le risorse figlio ereditano i tag impostati sulle risorse padre. Ad esempio:

- Se un tag è impostato su una `RestApi` risorsa, tale tag viene ereditato dalle seguenti risorse secondarie di quella `RestApi` per il controllo degli Attribute-based accessi:
  - `Authorizer`
  - `Deployment`
  - `Documentation`
  - `GatewayResponse`
  - `Integration`
  - `Method`

  - `Model`
  - `Resource`
  - `ResourcePolicy`
  - `Setting`
  - `Stage`
- Se un tag è impostato su un `DomainName`, tale tag viene ereditato da qualsiasi risorsa `BasePathMapping`, `ApiMapping` e `RoutingRule` sottostante.
- Se un tag è impostato su un `UsagePlan`, tale tag è ereditato da qualsiasi risorsa `UsagePlanKey` al di sotto di esso.

> **Note**
>
> L'eredità dei tag viene applicata solo per il controllo degli accessi basato sugli attributi. Ad esempio, non è possibile utilizzare tag ereditati per monitorare i costi in. AWS Cost Explorer Per utilizzare i tag per l'allocazione dei costi, si consiglia di creare i tag sulle risorse figlio, ad esempio la risorsa `Stage`.
> API Gateway non restituisce tag ereditati quando GetTagsrichiedi una risorsa.

Eredità dei tag nell'API di Amazon API Gateway V1

In precedenza, era solo possibile impostare tag su fasi. Ora che è possibile impostarli anche su altre risorse, uno `Stage` può ricevere un tag in due modi:

- Il tag può essere impostato direttamente sullo `Stage`.
- La fase può ereditare il tag dalla sua padr `RestApi`.

Se una fase riceve un tag in entrambi i modi, il tag che è stato impostato direttamente sulla fase ha la precedenza. Ad esempio, supponiamo che una fase erediti i seguenti tag dalla sua API REST padre:

```
{
 'foo': 'bar',
 'x':'y'
}
```

Supponiamo anche che disponga dei seguenti tag impostati direttamente su di essa:

```
{
 'foo': 'bar2',
 'hello': 'world'
}
```

Il risultato netto è che la fase dispone dei seguenti tag, con i seguenti valori:

```
{
 'foo': 'bar2',
 'hello': 'world'
 'x':'y'
}
```

Limitazioni applicate ai tag e convenzioni di utilizzo

Le seguenti limitazioni e convenzioni di utilizzo si applicano all'utilizzo dei tag con risorse API Gateway:

- Ogni risorsa può avere un massimo di 50 tag.
- Per ogni risorsa, ogni chiave di tag deve essere unica e ogni chiave di tag può avere un solo valore.
- La lunghezza massima della chiave del tag è di 128 caratteri Unicode. UTF-8
- La lunghezza massima del valore del tag è di 256 caratteri Unicode in. UTF-8
- I caratteri consentiti per chiavi e valori sono lettere, numeri, spazi rappresentabili in UTF-8 e i seguenti caratteri: .: + = @ _/- (trattino). Le risorse Amazon EC2 consentono qualsiasi carattere.
- Per chiavi e valori di tag viene fatta la distinzione tra maiuscole e minuscole. Come best practice, è consigliabile definire una strategia per l'uso delle lettere maiuscole e minuscole nei tag e implementarla costantemente in tutti i tipi di risorse. Ad esempio, puoi decidere se utilizzare `Costcenter`, `costcenter` o `CostCenter` e utilizzare la stessa convenzione per tutti i tag. Non utilizzare tag simili con lettere maiuscole o minuscole incoerenti.
- Il prefisso `aws:` non può essere utilizzato con i tag; è riservato per l'uso in AWS . Non è possibile modificare né eliminare le chiavi o i valori di tag con tale prefisso. I tag con questo prefisso non vengono conteggiati per il limite del numero di tag per risorsa.

Utilizzo dei tag per controllare l'accesso alle risorse REST API di Gateway API

Le condizioni nelle AWS Identity and Access Management policy fanno parte della sintassi utilizzata per specificare le autorizzazioni per le risorse API Gateway. Per ulteriori informazioni su come specificare le policy IAM, consulta the section called “Uso delle autorizzazioni IAM”. In API Gateway, le risorse possono avere tag e alcune operazioni possono includere tag. Quando si crea una policy IAM, è possibile utilizzare le chiavi di condizione di tag per controllare:

- Quali utenti possono eseguire operazioni su una risorsa API Gateway, in base ai tag di cui la risorsa dispone già.
- Quali tag possono essere passati in una richiesta di operazione.
- Se delle chiavi di tag specifiche possono essere utilizzate in una richiesta.

L'uso dei tag per il controllo degli accessi basato sugli attributi può consentire un controllo più preciso rispetto al controllo, nonché un controllo più dinamico rispetto al API-level controllo degli accessi basato sulle risorse. Possono essere create policy IAM che consentono o negano un'operazione basata sui tag forniti nella richiesta (tag di richiesta) o tag sulla risorsa su cui è eseguita l'operazione (tag delle risorse). In generale, i tag delle risorse sono per risorse già esistenti. I tag di richiesta sono riservati per la creazione di nuove risorse.

Per la sintassi completa e la semantica delle chiavi di condizione di tag, consulta Controllo degli accessi tramite tag nella Guida per l’utente di IAM.

I seguenti esempi mostrano come specificare le condizioni dei tag nelle policy per gli utenti API Gateway.

Limitare le operazioni in base ai tag delle risorse

La seguente policy di esempio concede agli utenti l'autorizzazione a eseguire tutte le operazioni su tutte le risorse, a condizione che tali risorse non dispongano del tag `Environment` con un valore di `prod`.

JSON

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

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "apigateway:*",
      "Resource": "*"
    },
    {
      "Effect": "Deny",
      "Action": [
        "apigateway:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Environment": "prod"
        }
      }
    }
  ]
}
```

Consentire operazioni in base ai tag delle risorse

La seguente policy di esempio consente agli utenti di eseguire tutte le operazioni sulle risorse Gateway API, a condizione che tali risorse dispongano del tag `Environment` con un valore di `Development`. L'istruzione `Deny` impedisce all'utente di modificare il valore del tag `Environment`.

```
{
  "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:StringEqualsIgnoreCase": {
          "aws:TagKeys": "Environment"
        }
      }
    }
  ]
}
```

Negare operazioni di assegnazione di tag

La seguente policy di esempio consente a un utente di eseguire tutte le operazioni Gateway API, ad eccezione della modifica dei tag.

JSON

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

```
        {
            "Effect": "Allow",
            "Action": [
                "apigateway:*"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Deny",
            "Action": [
                "apigateway:*"
            ],
            "Resource": "arn:aws:apigateway:*::/tags*"
        }
    ]
}
```

Consentire operazioni di assegnazione di tag

La seguente policy di esempio consente a un utente di disporre di tutte le risorse Gateway API e di modificare i tag per tali risorse. Per ottenere i tag per una risorsa, l'utente deve disporre delle autorizzazioni `GET` per quella risorsa. Per aggiornare i tag per una risorsa, l'utente deve disporre delle autorizzazioni `PATCH` per quella risorsa.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "apigateway:GET",
                "apigateway:PUT",
                "apigateway:POST",
                "apigateway:DELETE"
            ],
            "Resource": [
                "arn:aws:apigateway:*::/tags/*"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "apigateway:GET",
                "apigateway:PATCH"
            ],
            "Resource": [
                "arn:aws:apigateway:*::*"
            ]
        }
    ]
}
```

riferimenti API

Amazon API Gateway consente APIs di creare e distribuire il proprio HTTP e WebSocket APIs. Inoltre, API Gateway APIs sono disponibili in versione standard AWS SDKs.

Se utilizzi una lingua per cui esiste un AWS SDK, potresti preferire utilizzare l'SDK anziché utilizzare direttamente l'API Gateway APIs REST. Semplificano l'autenticazione, si integrano facilmente con l'ambiente di sviluppo e forniscono un facile accesso ai comandi API Gateway. SDKs

Ecco dove trovare la documentazione di riferimento sull' AWS SDKs API REST di API Gateway:

- Strumenti per Amazon Web Services
- Documentazione di riferimento delle API REST tramite Amazon API Gateway
- Amazon API Gateway WebSocket e riferimento alle API HTTP

Endpoint del servizio Gateway API

Un endpoint è un URL che funge da punto di ingresso per un servizio AWS Web. Gateway API supporta i seguenti tipi di endpoint:

- the section called “IPv4 estremità”
- Endpoint Dualstack (supportano sia che) IPv4 IPv6
- Endpoint FIPS

Quando si effettua una richiesta, è possibile specificare l’endpoint da utilizzare. Se non si specifica un endpoint, l'endpoint viene utilizzato per impostazione IPv4 predefinita. Per utilizzare un tipo di endpoint diverso, devi specificarlo nella richiesta. Per esempi su come eseguire questa operazione, consulta the section called “Specificazione degli endpoint”. Per una tabella degli endpoint disponibili, consulta Amazon API Gateway endpoints.

IPv4 estremità

IPv4 gli endpoint supportano solo il IPv4 traffico. IPv4 gli endpoint sono disponibili per tutte le regioni.

Se specifichi l'endpoint generico, `apigateway.amazonaws.com`, utilizziamo l'endpoint per `us-east-1`. Per utilizzare una Regione diversa, specifica l'endpoint associato. Ad esempio, se

specifichi `apigateway.us-east-2.amazonaws.com` come endpoint, indirizzeremo la tua richiesta all'endpoint `us-east-2`.

IPv4 i nomi degli endpoint utilizzano la seguente convenzione di denominazione:

- `apigateway.`*`region`*`.amazonaws.com`

Ad esempio, il nome dell' IPv4 endpoint per la regione è`eu-west-1`. `apigateway.eu-west-1.amazonaws.com`

Endpoint Dualstack (e) IPv4 IPv6

Gli endpoint Dualstack supportano sia il traffico che il traffico. IPv4 IPv6 Quando effettui una richiesta a un endpoint dualstack, l'URL dell'endpoint si risolve in un indirizzo IPv6 o in un IPv4 indirizzo, a seconda del protocollo utilizzato dalla rete e dal client.

I nomi degli endpoint dual-stack usano la seguente convenzione di denominazione:

- `apigateway.`*`region`*`.api.aws`

Ad esempio, il nome dell'endpoint dual-stack per la Regione `eu-west-1` è `apigateway.eu-west-1.api.aws`.

Specificazione degli endpoint

Gli esempi seguenti mostrano come specificare un endpoint per la regione utilizzando for. `us-east-2` AWS CLI `apigateway`

- Dualstack

```
aws apigateway get-rest-apis --region us-east-2 --endpoint-url https://apigateway.us-east-2.api.aws
```

- IPv4

```
aws apigateway get-rest-apis --region us-east-2 --endpoint-url https://apigateway.us-east-2.amazonaws.com
```

Gli esempi seguenti mostrano come specificare un endpoint per la `us-east-2` Regione utilizzando il AWS CLI for. `apigatewayv2`

- Dualstack

```
aws apigatewayv2 get-apis --region us-east-2 --endpoint-url https://apigateway.us-east-2.api.aws
```

- IPv4

```
aws apigatewayv2 get-apis --region us-east-2 --endpoint-url https://apigateway.us-east-2.amazonaws.com
```

Utilizzo IPv6 degli indirizzi nelle politiche IAM

Se utilizzi policy utente IAM o policy di risorse API Gateway per controllare l'accesso ad API Gateway o a qualsiasi API Gateway APIs, conferma che le policy siano aggiornate per includere intervalli di IPv6 indirizzi. Le policy che non sono aggiornate per gestire IPv6 gli indirizzi potrebbero influire sull'accesso del cliente ad API Gateway quando inizia a utilizzare l'endpoint dualstack. Per ulteriori informazioni sulla gestione delle autorizzazioni di accesso con IAM, consultare Gestione delle identità e degli accessi per Amazon API Gateway.

Le policy IAM che filtrano gli indirizzi IP utilizzano gli operatori di condizione degli indirizzi IP. La seguente politica di identità consente agli indirizzi IP `54.240.143.*` compresi nell'intervallo di ottenere informazioni su tutte le risorse di un HTTP o di un' WebSocket API con l'identificatore di. `a123456789` A qualsiasi indirizzo IP non incluso in questo intervallo verrà negato l'accesso a tutte le risorse dell'API. Poiché tutti IPv6 gli indirizzi non rientrano nell'intervallo consentito, questa politica impedisce IPv6 agli indirizzi di accedere alle informazioni sull'API.

JSON

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

```
      "Action": "apigateway:GET",
      "Resource": "arn:aws:apigateway:us-east-1::/apis/a123456789/*",
      "Condition": {
         "IpAddress": {"aws:SourceIp": "54.240.143.0/24"}
      }
    }
  ]
}
```

È possibile modificare l'`Condition`elemento della policy API per consentire sia gli IPv4 intervalli di indirizzi IPv6 (`2001:DB8:1234:5678::/64`) che (), come illustrato nell'esempio seguente. `54.240.143.0/24` È possibile utilizzare lo stesso tipo di blocco `Condition` mostrato nell'esempio per aggiornare sia le policy utente IAM sia le policy di risorse Gateway API.

```
"Condition": {
  "IpAddress": {
     "aws:SourceIp": [
       "54.240.143.0/24",
        "2001:DB8:1234:5678::/64"
      ]
    }
 }
```

Quote Gateway Amazon API

Le seguenti quote si applicano a tutti i tipi di API di Gateway Amazon API.

Quote a livello di account API Gateway per ogni regione

Di seguito sono elencate le quote per account per ogni regione in Amazon API Gateway.

Operazione o risorsa	Quota predefinita	Può essere aumentata
Limita la quota per account, per regione su HTTP APIs APIs WebSocket APIs, REST e callback WebSocket APIs	10.000 richieste al secondo (RPS) con un'ulteriore capacità di ottimizzazione fornita dall'algoritmo del bucket token, utilizzando una capacità del bucket massima di 5.000 richieste. * **Note** La quota di incremento delle prestazioni dipende dal team di servizio API Gateway in base alla quota RPS complessiva per l'account nella regione. Non è una quota che un cliente può controllare o per la quale può richiedere modifiche.	Sì
Limita la quota senza controllo dell'accesso per account per regione per un portale	250.000 richieste al secondo	No
Limita la quota con controllo degli accessi per account per	10.000 richieste al secondo	No

Operazione o risorsa	Quota predefinita	Può essere aumentata
regione per un portale		

* Nelle Regioni seguenti, la quota di limitazione (della larghezza di banda della rete) predefinita è di 2500 RPS e la quota di espansione predefinita è di 1250 RPS: Africa (Città del Capo), Europa (Milano), Asia Pacifico (Giacarta), Medio Oriente (Emirati Arabi Uniti), Asia Pacifico (Hyderabad), Asia Pacifico (Melbourne), Europa (Spagna), Europa (Zurigo), Israele (Tel Aviv), Canada occidentale (Calgary), Asia Pacifico (Malesia), Asia Pacifico (Thailandia) e Messico (centrale).

Quote di API Gateway per la creazione, la distribuzione e la gestione di un'API

Le seguenti quote fisse si applicano alla creazione, alla distribuzione e alla gestione di un'API in API Gateway, utilizzando la AWS CLI console API Gateway o l'API REST di API Gateway e le relative. SDKs Queste quote non possono essere aumentate.

Operazione	Quota predefinita	Può essere aumentata
CreateApiKey	5 richieste al secondo per account	No
CreateDeployment	1 richiesta ogni 5 secondi per account	No
CreateDocumentationVersion	1 richiesta ogni 20 secondi per account	No
CreateDomainName	1 richiesta ogni 30 secondi per account	No
CreateResource	5 richieste al secondo per account	No

Operazione	Quota predefinita	Può essere aumentata
CreateRestApiper API regionali o private	1 richiesta ogni 3 secondi per account	No
CreateRestApiper API ottimizzate per l'edge	1 richiesta ogni 30 secondi per account	No
CreateVpcLink(V2)	1 richiesta ogni 15 secondi per account	No
DeleteApiKey	5 richieste al secondo per account	No
DeleteDomainName	1 richiesta ogni 30 secondi per account	No
DeleteResource	5 richieste al secondo per account	No
DeleteRestApi	1 richiesta ogni 30 secondi per account	No
GetResources	5 richieste ogni 2 secondi per account	No
DeleteVpcLink(V2)	1 richiesta ogni 30 secondi per account	No
ImportDocumentationParts	1 richiesta ogni 30 secondi per account	No
ImportRestApiper API regionali o private	1 richiesta ogni 3 secondi per account	No
ImportRestApiper API ottimizzate per l'edge	1 richiesta ogni 30 secondi per account	No
PutRestApi	1 richiesta al secondo per account	No

Operazione	Quota predefinita	Può essere aumentata
UpdateAccount	1 richiesta ogni 20 secondi per account	No
UpdateDomainName	1 richiesta ogni 30 secondi per account	No
UpdateUsagePlan	1 richiesta ogni 20 secondi per account	No
Crea un portale	1 richiesta ogni 3 secondi	No
Aggiorna Portal	2 richieste al minuto	No
Ottieni Portal	10 richieste al secondo	No
Elenca portali	10 richieste al secondo	No
Pubblica il portale	2 richieste al minuto	No
DeletePortal	2 richieste al minuto	No
PreviewPortal	1 richiesta ogni 3 secondi	No
DisablePortal	2 richieste al minuto	No
GetPortalProduct	10 richieste al secondo	No
ListPortalProduct	5 richieste al secondo	No
CreatePortalProduct	2 richieste al secondo	No
UpdatePortalProduct	0,5 richieste al secondo	No
DeletePortalProduct	1 richiesta al secondo	No
PutPortalProductSharingPolicy	1 richiesta ogni 3 secondi	No
GetPortalProductSharingPolicy	10 richieste al secondo	No

Operazione	Quota predefinita	Può essere aumentata
DeletePortalProductSharingPolicy	1 richiesta ogni 3 secondi	No
CreateProductRestEndpointPage	1 richiesta ogni 3 secondi	No
GetProductRestEndpointPage	10 richieste al secondo	No
UpdateProductRestEndpointPage	1 richiesta ogni 3 secondi	No
DeleteProductRestEndpointPage	1 richiesta ogni 3 secondi	No
ListProductRestEndpointPages	10 richieste al secondo	No
CreateProductPage	1 richiesta ogni 3 secondi	No
GetProductPage	10 richieste al secondo	No
UpdateProductPage	1 richiesta ogni 3 secondi	No
DeleteProductPage	1 richiesta ogni 3 secondi	No
ListProductPages	10 richieste al secondo	No
Altre operazioni	Nessuna quota fino alla quota totale dell'account.	No
Operazioni totali	10 richieste al secondo con una quota di ottimizzazione di 40 richieste al secondo.	No

Quote per la configurazione e l'esecuzione di una REST API in Gateway API

Le quote seguenti si applicano alla configurazione e all'esecuzione di un'API REST in Amazon API Gateway. Se la quota è relativa a un'API specifica, può essere aumentata solo per determinate API e non per tutte le API di un account.

Per informazioni sulle quote a livello di account, consulta Quote e note importanti.

Operazione o risorsa	Quota predefinita	Può essere aumentata
Nomi di dominio personalizzati pubblici per account per Regione	120	Sì
Nomi di dominio personalizzati privati per account per Regione	50	Sì
API regionali per Regione	600	No
Edge-optimized API per regione	120	No
API private per account per regione	600	No
Associazioni di accesso di nomi di dominio per account	100	Sì

Operazione o risorsa	Quota predefinita	Può essere aumentata
Multi-level Mappature API per dominio	200	No
Regole di routing per dominio	50	Sì
Lunghezza , in caratteri, dell'URL per un'API ottimizzata per gli edge	8192	No
Lunghezza, in caratteri, dell'URL di un'API regionale	10240	No
Lunghezza, in caratteri, dell'URL di un'API privata	8192	No
Lunghezza massima, in caratteri, di una policy delle risorse API Gateway	8192	Sì
Chiavi API per account in ogni regione	10000	No

Operazione o risorsa	Quota predefinita	Può essere aumentata
Certificati Gateway-generated client API per account per regione	60	Sì
Certificati client ACM per account per regione	500	No
Autorizzatori per API (AWS Lambda e Amazon Cognito)	10	Sì Per aumentare questa quota, contattare il Centro Supporto AWS
Parti documentazione per API	2000	Sì Per aumentare questa quota, contattare il Centro Supporto AWS

Operazione o risorsa	Quota predefinita	Può essere aumentata
Risorse per API	300	Sì
Fasi per API	10	Sì
Variabili di fase per ogni fase	100	No
Lunghezza, in caratteri, della chiave in una variabile di fase	64	No
Lunghezza, in caratteri, del valore in una variabile di fase	512	No
Piani di utilizzo per account in ogni regione	300	Sì
Piani di utilizzo per chiave API	10	Sì
Link VPC per account in ogni regione	20	Sì
TTL caching API	300 secondi per impostazione predefinita e configurabile tra 0 e 3600 da un proprietario di API.	Non per il limite superiore (3600)

Operazione o risorsa	Quota predefinita	Può essere aumentata
Dimensione della risposta memorizzata nella cache	1048576 byte. la crittografia dei dati della cache può aumentare le dimensioni della voce quando viene memorizzata nella cache.	No
Timeout di integrazione per API regionali	50 millisecondi: 29 secondi per tutti i tipi di integrazione, inclusi Lambda, Lambda proxy, HTTP, proxy HTTP e integrazioni. AWS	Sì *
Timeout di integrazione per API ottimizzate per l'edge	50 millisecondi: 29 secondi per tutti i tipi di integrazione, inclusi Lambda, Lambda proxy, HTTP, proxy HTTP e integrazioni. AWS	No
Timeout di integrazione per API private	50 millisecondi: 29 secondi per tutti i tipi di integrazione, inclusi Lambda, Lambda proxy, HTTP, proxy HTTP e integrazioni. AWS	Sì *
Dimensione totale combinata di tutti i valori di intestazione, inclusi nomi di intestazione, valori, eventuali terminatori di riga e spazi bianchi	20480 byte	No
Totale dimensioni combinate di tutti i valori dell'integrazione per un'API privaa	8000 byte	No
Dimensione payload	10 MB	No

Operazione o risorsa	Quota predefinita	Può essere aumentata
Tag per fase	50	No
Numero di iterazioni in un loop `#foreach ... #end` nei modelli di mappatura	1000	No
Dimensione massima del modello di mappatura	300 KB	No
Dimensione del modello di log di accesso	3 KB	No
Lunghezza dell'ARN del metodo	1600 byte. Se il metodo contiene un parametro di percorso e un client utilizza un valore che supera la lunghezza dell'ARN, l'API restituirà una risposta `414 Request URI too long`.	No
Method-level impostazioni di limitazione per una fase di un piano di utilizzo	20	Sì Per aumentare questa quota, contattare il Centro Supporto AWS

Operazione o risorsa	Quota predefinita	Può essere aumentata
Dimensione dei modelli per API	400 KB	No
Numero di certificati per truststore	1.000 certificati con dimensioni totali dell'oggetto fino a 1 MB.	No
Timeout per connessione inattiva	310 secondi	No
Dimensione massima del file di definizione dell'API quando si utilizza restapi:import o restapi:put	6 MB	No

* Non è possibile impostare il timeout di integrazione su meno di 50 millisecondi. Puoi aumentare il timeout di integrazione a più di 29 secondi, ma ciò potrebbe richiedere una riduzione della quota di Region-level limitazione per il tuo account.

Best practice per ridurre le quote

Le seguenti best practice consentono di ridurre il numero attuale di risorse per evitare di aumentare la quota. È necessario assicurarsi che questi suggerimenti funzionino per l'architettura delle API in uso.

API per regione

Per ridurre il numero di API per Regione, esporta le API inutilizzate e quindi eliminale da Gateway API. Per ulteriori informazioni, consulta the section called "Esportazione di un'API REST".

Fasi per API

Per ridurre il numero di fasi per API, suddividi l'API in più API.

Risorse per API

Utilizza i percorsi `{proxy+}` per ridurre il numero di risorse. Per ulteriori informazioni, consulta the section called “Configurazione di integrazioni proxy mediante una risorsa proxy”.

Mappature API

Per ridurre il numero di mappature API per un nome di dominio personalizzato, utilizza mappature API con livelli singoli, ad esempio `/prod`. Le mappature API con un singolo livello non vengono conteggiate ai fini della quota delle mappature API.

Autorizzatori per API

Per ridurre il numero di sistemi di autorizzazione per ogni API, riutilizza i sistemi di autorizzazione nei metodi API.

Parti documentazione per API

Per ridurre il numero di parti della documentazione per API, utilizza `ignore=documentation` quando importi l’API. Per ulteriori informazioni, consulta the section called “Importare la documentazione dell'API”.

È anche possibile utilizzare l’ereditarietà dei contenuti per consentire ad alcune parti della documentazione di ereditare i contenuti da un’entità API con specifiche più generali. Per ulteriori informazioni, consulta the section called “Rappresentazione della documentazione dell'API in API Gateway”.

Lunghezza massima, in caratteri, di una policy delle risorse API Gateway

Per ridurre la durata di una policy relativa alle risorse, usala AWS WAFV2 per proteggere la tua API. Per ulteriori informazioni, consulta the section called “AWS WAF”.

Se la policy contiene indirizzi IP, è anche possibile utilizzare intervalli anziché valori specifici per definire gli indirizzi IP.

Piani di utilizzo per chiave API

Per ridurre il numero di piani di utilizzo per chiave API, utilizza una chiave API per piano di utilizzo e associa il piano di utilizzo a più API. Non è consigliabile condividere una chiave API tra più piani di utilizzo.

Quote per la configurazione e l'esecuzione di un'API HTTP in Gateway API

Le quote seguenti si applicano alla configurazione e all'esecuzione di un'API HTTP in Gateway Amazon API. Se la quota è per API, può essere aumentata solo in modo specifico APIs, non per tutti gli APIs account. Per informazioni sulle quote a livello di account, consulta Quote e note importanti

Operazione o risorsa	Quota predefinita	Può essere aumentata
Instradamenti per API	300	Sì
Integrazioni per API	300	No
Timeout massimo dell'integrazione	30 secondi	No
Fasi per API	10	Sì
Mappature API multilivello per dominio	200 mappature API. Questo limite non include le mappature API con livelli singoli, ad esempio `/prod`.	No
Tag per fase	50	No
Dimensione totale combinata dei valori della riga di richiesta e dell'inte stazione	10240 byte	No
Dimensione payload	10 MB	No

Operazione o risorsa	Quota predefinita	Può essere aumentata
Domini personalizzati per account per regione	120	Sì
Dimensione del modello di log di accesso	3 KB	No
Inserimento nel CloudWatch log di Amazon Logs	1 MB	No
Autorizzatori per API	10	Sì Per aumentare questa quota, contattare il Centro Supporto AWS
Destinatari per autorizzatore	50	No
Ambiti per percorso	10	No
Timeout per l'endpoint del set di chiavi Web JSON	1500 ms	No

Operazione o risorsa	Quota predefinita	Può essere aumentata
Dimensione della risposta dall'endpoint del set di chiavi Web JSON	150000 byte	No
Timeout per l'endpoint di individuazione OpenID Connect	1500 ms	No
Timeout per la risposta della funzione Lambda di autorizzazione	10000 ms	No
Link VPC per account in ogni regione	10	Sì
Sottoreti per collegamento VPC	10	Sì
Variabili di fase per ogni fase	100	No
Lunghezza, in caratteri, della chiave in una variabile di fase	64	No
Lunghezza, in caratteri, del valore in una variabile di fase	512	No

Best practice per ridurre le quote

Le seguenti best practice consentono di ridurre il numero attuale di risorse per evitare di aumentare la quota. È necessario assicurarsi che questi suggerimenti funzionino per l'architettura delle API in uso.

APIs per regione

Per ridurre il numero di APIs per regione, esporta quelli inutilizzati APIs e quindi eliminali da API Gateway. Per ulteriori informazioni, consulta the section called "Esportazione".

Fasi per API

Per ridurre il numero di fasi per API, suddividi l'API in più APIs fasi.

Autorizzatori per API

Per ridurre il numero di sistemi di autorizzazione per ogni API, riutilizza i sistemi di autorizzazione nei metodi API.

Quote per la configurazione dei portali in API Gateway

Le seguenti quote si applicano alla creazione di portali in API Gateway. Per ulteriori informazioni, consulta Portali API Gateway.

Operazione o risorsa	Quota predefinita	Può essere aumentata
Portali per account	15	Sì
Prodotti del portale per portale	200	No
Prodotti del portale per account	500	No
Pagine dell'endpoint REST del prodotto per prodotto del portale	40	Sì

Operazione o risorsa	Quota predefinita	Può essere aumentata
Pagine di prodotto per prodotto del portale	40	Sì
Dimensioni del logo	200 KB	No
Dimensioni della pagina di documenta zione per pagina dell'endpoint REST del prodotto	32.000 caratteri	No
Dimensioni di pagina personali zzate per le pagine dei prodotti	32.000 caratteri	No
Nomi di dominio personalizzati per portale	1	No
Autorizzatori per portale	1	No

Quote per la configurazione e l'esecuzione di un API WebSocket Gateway

Le seguenti quote si applicano alla configurazione e all'esecuzione di un' WebSocket API in Amazon API Gateway. Se la quota è per API, può essere aumentata solo in modo specifico APIs, non per tutti i APIs componenti di un account. Per informazioni sulle quote a livello di account, consulta Quote e note importanti

Operazione o risorsa	Quota predefinita	Può essere aumentata
Nuove connessioni al secondo per account (in tutto WebSocket APIs) per regione	500	Sì
Connessioni simultanee	Non applicabile *	Non applicabile
AWS Lambda autorizzatori per API	10	Sì Per aumentare questa quota, contattare il Centro Supporto AWS
AWS Lambda dimensione del risultato dell'autorizzatore	8 KB	No
Instradamenti per API	300	Sì
Integrazioni per API	300	Sì Per aumentare

Operazione o risorsa	Quota predefinita	Può essere aumentata
		questa quota, contattare il Centro Supporto AWS
Timeout integrazione	50 millisecondi - 29 secondi per tutti i tipi di integrazione, inclusi Lambda, proxy Lambda, HTTP, proxy HTTP e integrazioni. AWS	No
Fasi per API	10	Sì
WebSocket dimensione della cornice	32 KB	No
Dimensioni payload del messaggio	128 KB **	No
Dimensione massima del modello di mappatura	300 KB	No
Durata della connessione per l' WebSocket API	2 ore	No
Timeout per connessione inattiva	10 minuti	No

Operazione o risorsa	Quota predefinita	Può essere aumentata
Lunghezza, in caratteri, dell'URL di un' WebSocket API	4096	No
Dimensione del modello di log di accesso	3 KB	No

* Il servizio API Gateway non applica una quota sulle connessioni simultanee. Il numero massimo di connessioni simultanee è determinato dalla velocità delle nuove connessioni al secondo e dalla durata massima della connessione di due ore. Ad esempio, con la quota predefinita di 500 nuove connessioni al secondo, se i client si connettono alla velocità massima per più due ore, API Gateway può servire fino a 3.600.000 connessioni simultanee.

** A causa della quota di WebSocket dimensione del frame di 32 KB, un messaggio di dimensioni superiori a 32 KB deve essere suddiviso in più frame, ciascuno di 32 KB o più piccolo. Si applica ai comandi `@connections`. Se si riceve un messaggio di dimensioni maggiori (o con dimensioni di frame maggiori), la connessione viene chiusa con il codice 1009.

Note importanti Amazon API Gateway

La sezione seguente descrive in dettaglio le note che possono riguardare l'utilizzo di Gateway API.

Argomenti

- Note importanti su Amazon API Gateway per HTTP APIs
- Note importanti su Amazon API Gateway per HTTP e WebSocket APIs
- Note importanti su Amazon API Gateway per REST e WebSocket APIs
- Note importanti su Amazon API Gateway per WebSocket APIs
- Note importanti su Amazon API Gateway per REST APIs

Note importanti su Amazon API Gateway per HTTP APIs

- HTTP APIs tradurrà le `X-Forwarded-*` intestazioni in entrata in un'`Forwarded`intestazione standard e aggiungerà l'IP di uscita, l'host e il protocollo.
- Gateway API aggiunge l'intestazione Content-type alla richiesta se non è presente alcun payload e Content-Length è pari a 0.

Note importanti su Amazon API Gateway per HTTP e WebSocket APIs

- La versione 4A di Signature non è ufficialmente supportata da Amazon API Gateway per HTTP e WebSocket APIs.

Note importanti su Amazon API Gateway per REST e WebSocket APIs

- API Gateway non supporta la condivisione di un nome di dominio personalizzato tra REST e WebSocket APIs.
- I nomi di fasi possono contenere solo caratteri alfanumerici, trattini e caratteri di sottolineatura. La lunghezza massima è 128 caratteri.
- I percorsi di `/ping` e `/sping` sono riservati al controllo dello stato del servizio. Se questi percorsi vengono utilizzati per le risorse API al livello di root con domini personalizzati, non sarà possibile ottenere il risultato previsto.
- Attualmente API Gateway limita gli eventi di log a 1024 byte. Gli eventi di registro di dimensioni superiori a 1024 byte, come i corpi di richiesta e risposta, verranno troncati da API Gateway prima dell'invio ai log. CloudWatch
- CloudWatch Attualmente Metrics limita i nomi e i valori delle dimensioni a 255 caratteri XML validi. Per ulteriori informazioni, consulta la Guida per l'utente di CloudWatch . I valori di dimensione sono una funzione di nomi definiti dall'utente, tra cui nome dell'API, nome dell'etichetta (fase) e nome della risorsa. Quando scegli questi nomi, fai attenzione a non superare i limiti CloudWatch delle metriche.
- La dimensione massima di un modello di mappatura è 300 KB.

Note importanti su Amazon API Gateway per WebSocket APIs

- API Gateway supporta payload dei messaggi fino a 128 KB con dimensione del frame massima di 32 KB. Se un messaggio supera i 32 KB, occorre suddividerlo in più frame, ciascuno di 32 KB o più piccolo. Se si riceve un messaggio di dimensioni maggiori, la connessione viene chiusa con codice 1009.

Note importanti su Amazon API Gateway per REST APIs

- Il carattere barra verticale di testo semplice (`|`) e il carattere parentesi graffa (`{` o `}`) non sono supportati per le stringhe di query dell'URL della richiesta e devono presentare la codifica URL.
- Il punto e virgola (`;`) non è supportato per qualsiasi URL di richiesta della stringa di query e i risultati dei dati del frazionamento.
- REST APIs decodifica i parametri di richiesta con codifica URL prima di passarli alle integrazioni di backend. Per i parametri di richiesta UTF-8, REST APIs decodifica i parametri e poi li passa come unicode alle integrazioni di backend. APIs L'URL REST codifica il carattere percentuale () prima di passarlo alle integrazioni di backend. `%`
- Se effettui il test di un'API tramite la console API Gateway, potresti ricevere la risposta "unknown endpoint errors" (errori endpoint sconosciuti) se un certificato autofirmato viene inviato al back-end, se il certificato intermedio manca nella catena dei certificati o se viene rilevata dal back-end qualsiasi altra eccezione relativa a un certificato non riconoscibile.
- È consigliabile eliminare un'entità API `Resource` o `Method` con un'integrazione privata, dopo aver rimosso eventuali riferimenti hardcoded di un `VpcLink`. In caso contrario, l'integrazione verrà sospesa e riceverai un errore che ti informerà che il collegamento VPC è ancora in uso anche in seguito all'eliminazione dell'entità `Resource` o `Method`. Questo comportamento non si applica quando l'integrazione privata fa riferimento a `VpcLink` tramite una variabile di fase.
- I back-end seguenti potrebbero non supportare l'autenticazione client SSL in un modo che sia compatibile con API Gateway:
  - NGINX
  - Heroku
- API Gateway supporta in gran parte la specifica OpenAPI 2.0 e la specifica OpenAPI 3.0, con le seguenti eccezioni:
  - I segmenti di percorso possono contenere solo caratteri alfanumerici, trattini, punti, virgole, due punti e parentesi graffe. I parametri di percorso devono essere segmenti di

percorso separati. Ad esempio, "risorsa/{nome_parametro_percorso}" è valido, mentre "risorsa{nome_parametro_percorso}" no.

- I nomi di modelli possono contenere solo caratteri alfanumerici.
- Per i parametri di input, sono supportati solo i seguenti attributi: `name`, `in`, `required`, `type`, `description`. Altri attributi vengono ignorati.
- Il tipo `securitySchemes`, se utilizzato, deve essere `apiKey`. Tuttavia, l'autenticazione OAuth 2 e HTTP Basic sono supportate tramite autorizzatori Lambda; la configurazione OpenAPI viene ottenuta tramite le estensioni del fornitore.
- Il `deprecated` campo non è supportato e viene eliminato nella cartella Exported. APIs
- I modelli di API Gateway sono definiti tramite la bozza 4 dello schema JSON, anziché lo schema JSON utilizzato da OpenAPI.
- Il parametro `discriminator` non è supportato in nessun oggetto dello schema.
- Il tag `example` non è supportato.
- Il campo `nullable` non è supportato.
- `exclusiveMinimum` non è supportato da API Gateway.
- I tag `maxItems` e `minItems` non sono inclusi nella convalida della richiesta semplice. Come soluzione alternativa, aggiorna il modello in seguito all'importazione e prima di procedere con la convalida.
- Per OpenAPI 3.0, non è possibile specificare una parola chiave `oneOf`, `anyOf` o `allOf` che utilizza `$ref` per una definizione all'interno dello stesso schema. È possibile immettere direttamente lo schema o definire una risorsa separata del modello Gateway API. Per ulteriori informazioni, consulta the section called “Creazione di modelli più complessi”.
- `oneOf` non è supportato per OpenAPI 2.0 o la generazione di SDK.
- Il campo `readOnly` non è supportato.
- `$ref` non può essere utilizzato per fare riferimento ad altri file.
- Le definizioni di risposta del modulo `"500": {"$ref": "#/responses/UnexpectedError"}` non sono supportate nella root del documento di OpenAPI. Come soluzione alternativa, sostituisci i riferimenti con lo schema inline.
- I numeri del tipo `Int32` o `Int64` non sono supportati. Di seguito viene riportato un esempio:

```
"elementId": {
    "description": "Working Element Id",
    "format": "int32",
    "type": "number"
```

```
}
```

- Il tipo di formato con numeri decimali (`"format": "decimal"`) non è supportato nelle definizioni dello schema.
- Nelle risposte del metodo, la definizione dello schema deve essere di tipo oggetto e non di tipo di base. Ad esempio, `"schema": { "type": "string"}` non è supportato. Tuttavia, puoi trovare una soluzione alternativa al problema utilizzando il tipo di oggetto seguente:

```
"schema": {
     "$ref": "#/definitions/StringResponse"
           }

 "definitions": {
    "StringResponse": {
      "type": "string"
    }
  }
```

- API Gateway non utilizza la sicurezza a livello di radice definita nella specifica di OpenAPI. Pertanto, la sicurezza deve essere definita a livello di operazione per essere applicata in modo appropriato.
- La parola chiave `default` non è supportata.

- In API Gateway vengono applicate le seguenti restrizioni e limitazioni per la gestione dei metodi con l'integrazione Lambda o HTTP.
  - I nomi delle intestazioni e i parametri di query vengono elaborati tenendo presente la distinzione tra lettere maiuscole e minuscole.
  - La tabella seguente elenca le intestazioni che potrebbero venire interrotte, rimappate o modificate quando vengono inviate all'endpoint dell'integrazione o reinviate dall'endpoint dell'integrazione: In questa tabella:
    - `Remapped` significa che il nome dell'intestazione è cambiato da *`<string>`* a `X-Amzn-Remapped-`*`<string>`*.

      `Remapped Overwritten` significa che il nome dell'intestazione è cambiato da *`<string>`* a `X-Amzn-Remapped-`*`<string>`* e il valore viene sovrascritto.

Nome intestazione	Richiesta (**http**/**http_proxy** /**lambda**)	Risposta (**http**/**http_proxy** /**lambda**)
`Age`	Transito	Transito
`Accept`	Transito	Dropped/ Passthrough/ Passthrough
`Accept-Charset`	Transito	Transito
`Accept-Encoding`	Transito	Transito
`Authorization`	Transito *	Rimappata
`Connection`	Passthrough/Passthrough/Dropped	Rimappata
`Content-Encoding`	Passthrough/Dropped/Passthrough	Transito
`Content-Length`	Transito (generato in base al corpo)	Transito
`Content-MD5`	Interrotta	Rimappata
`Content-Type`	Transito	Transito
`Date`	Transito	Rimappata Sovrascritta

Nome intestazione	Richiesta (**http**/**http_proxy** /**lambda**)	Risposta (**http**/**http_proxy** /**lambda**)
`Expect`	Interrotta	Interrotta
`Host`	Sovrascritto nell'endpoint di integrazione	Interrotta
`Max-Forwards`	Interrotta	Rimappata
`Pragma`	Transito	Transito
`Proxy-Authenticate`	Interrotta	Interrotta
`Range`	Transito	Transito
`Referer`	Transito	Transito
`Server`	Interrotta	Rimappata Sovrascritta
`TE`	Interrotta	Interrotta
`Transfer-Encoding`	Dropped/Dropped/Exception	Interrotta
`Trailer`	Interrotta	Interrotta
`Upgrade`	Interrotta	Interrotta
`User-Agent`	Transito	Rimappata

Nome intestazione	Richiesta (**http**/**http_proxy** /**lambda**)	Risposta (**http**/**http_proxy** /**lambda**)
`Via`	Dropped/Dropped/Passthrough	Passthrough/ Dropped/ Dropped
`Warn`	Transito	Transito
`WWW-Authenticate`	Interrotta	Rimappata

* L'intestazione `Authorization` viene eliminata se contiene una firma Signature Version 4 o viene usata un'autorizzazione `AWS_IAM`.

- L'SDK Android di un'API generata da API Gateway utilizza la classe `java.net.HttpURLConnection`. Questa classe genererà un'eccezione non gestita, sui dispositivi con Android 4.4 e versioni precedenti, per la risposta 401 derivante dalla rimappatura dell'intestazione `WWW-Authenticate` su `X-Amzn-Remapped-WWW-Authenticate`.
- A differenza di Java, Android e iOS SDKs di un'API generati da API Gateway, l' JavaScript SDK di un'API generata da API Gateway non supporta tentativi per errori di livello 500.
- La chiamata di test di un metodo utilizza il tipo di contenuto predefinito di `application/json` e ignora le specifiche di altri tipi di contenuto.
- Quando invii richieste a un'API passando l'intestazione `X-HTTP-Method-Override`, API Gateway sostituisce il metodo. Per passare l'intestazione al back-end, è necessario aggiungere l'intestazione all'integrazione richiesta.
- Quando una richiesta contiene più tipi di supporto nell'intestazione `Accept`, API Gateway mantiene la conformità solo al primo tipo di supporto `Accept`. Nella situazione in cui non puoi controllare l'ordine dei tipi di supporto `Accept` e il tipo di supporto del tuo contenuto binario non è il primo dell'elenco, puoi aggiungere il primo tipo di supporto `Accept` nell'elenco `binaryMediaTypes` della tua API e API Gateway restituirà il tuo contenuto come binario. Ad esempio, per inviare un file JPEG utilizzando un elemento `<img>` in un browser, il browser

potrebbe inviare `Accept:image/webp,image/*,*/*;q=0.8` in una richiesta. Aggiungendo `image/webp` all'elenco `binaryMediaTypes`, l'endpoint riceverà il file JPEG come binario.

- La personalizzazione della risposta del gateway predefinito per `413 REQUEST_TOO_LARGE` non è attualmente supportata.
- API Gateway include un'intestazione `Content-Type` per tutte le risposte di integrazione. Per impostazione predefinita, il tipo di contenuto è `application/json`.

Cronologia dei documenti

La tabella seguente descrive le modifiche importanti apportate alla documentazione dall'ultima versione di Amazon API Gateway. Per la notifica sugli aggiornamenti di questa documentazione, puoi abbonarti a un feed RSS selezionando il pulsante RSS nel pannello del menu in alto.

- Ultimo aggiornamento della documentazione: 2 dicembre 2025

Modifica	Descrizione	Data
Aggiungi un'API API Gateway REST come destinazione per Amazon Bedrock AgentCore Gateway	API Gateway ora supporta l'aggiunta di un'API REST come destinazione per Amazon Bedrock AgentCore Gateway. Per ulteriori informazioni, consulta Aggiungere un'API API Gateway REST come destinazione per Amazon Bedrock AgentCore Gateway.	2 dicembre 2025
Integrazione privata con Application Load Balancer	API Gateway ora supporta integrazioni private con Application Load Balancers for REST. APIs Per ulteriori informazioni, consulta Integrazioni private per REST. APIs	21 novembre 2025
Politiche di sicurezza per REST APIs e nomi di dominio personalizzati	API Gateway ora supporta politiche di sicurezza TLS avanzate su REST APIs e nomi di dominio personalizzati. Per ulteriori informazioni, consulta Politiche di	19 novembre 2025

	sicurezza per REST APIs in API Gateway.	
Streaming di risposte per REST APIs	API Gateway REST APIs ora supporta lo streaming progressivo dei payload di risposta ai client non appena diventano disponibili. Per ulteriori informazioni, consulta Trasmetti in streaming la risposta di integrazione per le tue integrazioni proxy.	19 novembre 2025
Portali per sviluppatori	API Gateway ora supporta i portali, che sono una posizione centralizzata in cui i tuoi clienti possono scoprire e testare i tuoi prodotti APIs e quelli del portale che rappresentano un servizio o una funzionalità che desideri condividere. Per ulteriori informazioni, consulta Portali API Gateway.	19 novembre 2025
SIGv4a per REST APIs	API Gateway ora supporta l'utilizzo della versione 4a di Signature per REST APIs. Per ulteriori informazioni, consulta API Gateway permissions model for invoking an API.	19 agosto 2025

Regole di routing per REST APIs	API Gateway ora supporta le regole di routing per REST APIs utilizzando nomi di dominio personalizzati. È possibile instradare dinamicamente le richieste in entrata in base ai valori di intestazione HTTP, ai percorsi di base degli URL o a una combinazione di entrambi. Per ulteriori informazioni, consulta Inviare traffico al tuo APIs tramite il tuo nome di dominio personalizzato.	3 giugno 2025
È stato aggiunto il supporto per gli endpoint dual-stack (IPv4 e) IPv6	API Gateway ora supporta endpoint dual-stack per REST, HTTP e nomi di dominio WebSocket APIs personalizzati. Per ulteriori informazioni, consultare IP address types.	28 marzo 2025
Nomi di dominio personalizzati per uso privato APIs	Aggiunto il supporto per nomi di dominio personalizzati per uso privato APIs. Per ulteriori informazioni, consulta Nomi di dominio personalizzati per uso privato APIs.	21 novembre 2024
Aggiunto il supporto per TLS 1.3	API Gateway ora supporta TLS 1.3 su REST regionale APIs APIs, HTTP e WebSocket APIs. Per ulteriori informazioni, consulta Choosing a security policy for your custom domain in API Gateway.	15 febbraio 2024

Aggiornamenti dell'API REST e della console WebSocket API	Informazioni aggiornate sulla console per REST APIs e WebSocket APIs.	10 dicembre 2023
Aggiornamento della documentazione	Informazioni concettuali aggiornate e creazione di nuovi tutorial per le trasforma zioni dei dati e argomenti di convalida delle richieste per API Gateway REST. APIs Per ulteriori informazioni, consulta Utilizzo della convalida delle richieste in API Gateway e Configurazione delle trasforma zioni dei dati per le REST API.	22 giugno 2023
Configurazione del failover DNS per un'API Gateway su più regioni	È stato aggiunto il supporto per utilizzare i controlli di integrità di Amazon Route 53 per controllare il failover DNS da un'API REST di API Gateway in una regione primaria Regione AWS a una in una regione secondaria. Per ulteriori informazioni, consulta Configurazione dei controlli dell'integrità personalizzati per failover DNS.	31 ottobre 2022
Aggiornamento della documentazione	Riepiloghi delle funzionalità principali aggiornati per API REST e API HTTP. APIs Per ulteriori informazioni, consulta Scelta tra API REST e API APIs HTTP.	31 maggio 2022

Aggiornamento della policy gestita	Aggiunto supporto `acm:GetCertificate` per la policy `AWSServiceRoleForAPIGateway`. Per ulteriori informazioni, consulta la sezione Utilizzo dei ruoli collegati ai servizi per API Gateway.	12 luglio 2021
Mappatura dei parametri per HTTP APIs	È stato aggiunto il supporto per la mappatura dei parametri per HTTP. APIs Per ulteriori informazioni, consulta Trasformazione delle richieste e delle risposte API.	7 gennaio 2021
Disabilitazione dell'endpoint predefinito per un'API REST	È stato aggiunto il supporto per la disabilitazione dell'endpoint predefinito per REST. APIs Per ulteriori informazioni, consulta Disabilitazione dell'endpoint predefinito per un'API REST.	29 ottobre 2020
Autenticazione TLS reciproca	È stato aggiunto il supporto per l'autenticazione TLS reciproca per REST APIs e HTTP. APIs Per ulteriori informazioni, consulta Configurazione dell'autenticazione TLS reciproca per un'API REST e Configurazione dell'autenticazione TLS reciproca per un'API HTTP.	17 settembre 2020

Autorizzatori API AWS Lambda HTTP	È stato aggiunto il supporto per gli AWS Lambda autorizzatori per HTTP. APIs Per ulteriori informazioni, consulta Lavorare con AWS Lambda gli autorizzatori per HTTP. APIs	09 settembre 2020
Integrazioni dei AWS servizi API HTTP	È stato aggiunto il supporto per le integrazioni di AWS servizi per HTTP. APIs Per ulteriori informazioni, consulta Lavorare con le integrazioni AWS di servizi per HTTP. APIs	20 agosto 2020
Domini personalizzati con caratteri jolly API HTTP	È stato aggiunto il supporto per i nomi di dominio personalizzati wildcard per HTTP. APIs Per ulteriori informazioni, consulta Nomi di dominio personalizzati con caratteri jolly.	10 agosto 2020
Miglioramenti del portale per sviluppatori serverless	Aggiunta la gestione degli utenti al pannello di amministrazione e il supporto per l'esportazione delle definizioni API. Per ulteriori informazioni, consulta Usa il portale per sviluppatori serverless per catalogare il tuo API Gateway APIs.	25 giugno 2020

WebSocket Supporto per le API `Sec-WebSocket-Protocol`	Aggiunta del supporto per il campo `Sec-WebSocket-Protocol` Per ulteriori informazioni, vedi Configurazione di una route $connect che richiede un WebSocket sottoprotocollo.	16 giugno 2020
Esportazione di API HTTP	È stato aggiunto il supporto per l'esportazione delle definizioni OpenAPI 3.0 di HTTP. APIs Per ulteriori informazioni, consulta Esportazione di un'API HTTP da API Gateway.	20 aprile 2020
Documentazione sulla sicurezza	Aggiunta documentazione sulla sicurezza Per ulteriori informazioni, consulta Sicurezza in Amazon API Gateway.	31 marzo 2020
Documentazione riorganizzata.	Riorganizzata la guida per sviluppatori.	12 marzo 2020
Disponibilità generale API HTTP	HTTP rilasciato APIs in disponibilità generale. Per ulteriori informazioni, consulta Lavorare con HTTP APIs.	12 marzo 2020
Registrazione API HTTP	Aggiunto il supporto per `$context.integrationErrorMessage` nei log API HTTP. Per ulteriori informazioni, consulta Variabili di registrazione API HTTP.	26 febbraio 2020

AWS variabili per l'importazione OpenAPI	Aggiunto il supporto per AWS le variabili nelle definizioni OpenAPI. Per ulteriori informazioni, consulta Variabili AWS per importazione OpenAPI.	17 febbraio 2020
HTTP APIs	Rilasciato HTTP APIs in versione beta. Per ulteriori informazioni, consulta HTTP APIs.	4 dicembre 2019
Nomi di dominio personalizzati con caratteri jolly	Aggiunto il supporto per i nomi di dominio personalizzati con caratteri jolly. Per ulteriori informazioni, consulta Nomi di dominio personalizzati con caratteri jolly.	21 ottobre 2019
Registrazione dei log su Amazon Data Firehose	Aggiunto il supporto per Amazon Data Firehose come destinazione per i dati di registrazione dei log degli accessi. Per ulteriori informazioni, consulta Using Amazon Data Firehose as a Destination for API Gateway Access Logging.	15 ottobre 2019
Alias Route53 per invocare private APIs	È stato aggiunto il supporto per record DNS di alias Route53 aggiuntivi per l'invocazione di dati privati. APIs Per ulteriori informazioni, consulta la sezione relativa all'accesso all'API privata tramite alias Route53.	18 settembre 2019

Controllo degli accessi basato su tag per WebSocket APIs	È stato aggiunto il supporto per il controllo degli accessi basato su tag per. WebSocket APIs Per ulteriori informazioni, consulta la sezione relativa alle risorse API Gateway a cui possono essere applicati tag.	27 giugno 2019
Selezione della versione di TLS per domini personalizzati	È stato aggiunto il supporto per la selezione delle versioni di Transport Layer Security (TLS) APIs che vengono distribuite su domini personalizzati. Consulta la nota nella sezione relativa alla scelta di una versione di TLS minima per un dominio personalizzato in API Gateway.	20 giugno 2019
Policy degli endpoint VPC per uso privato APIs	È stato aggiunto il supporto per migliorare la sicurezza dei dati privati APIs collegando le policy degli endpoint agli endpoint VPC di interfaccia. Per ulteriori informazioni, consulta Use VPC Endpoint Policies for Private in API APIs Gateway.	4 giugno 2019
Documentazione aggiornata	Riscritta la sezione Nozioni di base su Amazon API Gateway. Tutorial spostati in Tutorial di Amazon API Gateway.	29 maggio 2019

Controllo degli accessi basato su tag per REST APIs	È stato aggiunto il supporto per il controllo degli accessi basato su tag per REST. APIs Per ulteriori informazioni, consulta la sezione relativa all'utilizzo dei tag con policy IAM per controllare l'accesso alle risorse API Gateway.	23 maggio 2019
Documentazione aggiornata	Sono stati riscritti sei argomenti: Che cos'è Amazon API Gateway?, Tutorial: Creazione di un'API con l'integrazione proxy HTTP, Tutorial: Creazione di un'API REST Calc con tre integrazioni non proxy, Informazioni di riferimento sui modelli di mappatura e sulle variabili di logging degli accessi in API Gateway, Utilizzo delle autorizzazioni Lambda di API Gateway e Abilitazione di CORS per una risorsa API REST di API Gateway.	5 aprile 2019
Miglioramenti del portale per sviluppatori serverless	Aggiunto il pannello di amministrazione e altri miglioramenti per semplificare la pubblicazione APIs nel portale per sviluppatori Amazon API Gateway. Per ulteriori informazioni, consulta Usa un portale per sviluppatori per catalogare i tuoi APIs.	28 marzo 2019

Support per AWS Config	È stato aggiunto il supporto per AWS Config. Per ulteriori informazioni, consulta Monitoraggio della configurazione dell'API Gateway API con AWS Config.	20 marzo 2019
Support per CloudFormation	Aggiunta l'API API Gateway V2 al riferimento del CloudFormation modello. Per ulteriori informazioni, consulta l'argomento relativo alla documentazione di riferimento sui tipi di risorse di Amazon API Gateway V2.	7 febbraio 2019
Support per WebSocket APIs	È stato aggiunto il supporto per WebSocket APIs. Per ulteriori informazioni, consulta Creazione di un' WebSocket API in Amazon API Gateway.	18 dicembre 2018
Portale per sviluppatori serverless disponibile tramite AWS Serverless Application Repository	L'applicazione serverless del portale per sviluppatori Amazon API Gateway è ora disponibile presso AWS Serverless Application Repository(in aggiunta a GitHub). Per ulteriori informazioni, consulta Utilizzare un portale per sviluppatori per catalogare il proprio API Gateway APIs.	16 novembre 2018

Support per AWS WAF	Aggiunto il supporto per AWS WAF (Web Applicati on Firewall). Per ulteriori informazioni, consulta Controllare l'accesso a un'API utilizzando AWS WAF.	5 Novembre 2018
Portale per gli sviluppatori serverless	Amazon API Gateway ora fornisce un portale per sviluppatori completam ente personalizzabile come applicazione serverless che puoi implementare per pubblicare il tuo API Gateway. APIs Per ulteriori informazioni, consulta Utilizzare un portale per sviluppatori per catalogare il proprio API Gateway APIs.	29 ottobre 2018
Supporto per le intestazioni multi-valore e i parametri di stringa di query	Amazon API Gateway supporta ora più intestazi oni e parametri di stringa di query con lo stesso nome. Per ulteriori informazioni, consulta la sezione relativa al supporto per le intestazioni multi-val ore e i parametri di stringa di query.	4 ottobre 2018
Supporto OpenAPI	Amazon API Gateway supporta ora OpenAPI 3.0 e OpenAPI (Swagger) 2.0.	27 settembre 2018

Documentazione aggiornata	Aggiunto un nuovo argomento relativo a come le policy delle risorse di Amazon API Gateway influiscono sul flusso di lavoro delle autorizzazioni.	27 settembre 2018
AWS X-Ray Integrazione attiva	Ora puoi utilizzarla AWS X-Ray per tracciare e analizzare le latenze nelle richieste degli utenti man mano che arrivano APIs ai servizi sottostanti. Per ulteriori informazioni, consulta Trace API Gateway API Execution with AWS X-Ray.	6 settembre 2018
Miglioramenti caching	Il sistema di caching sarà abilitato per impostazione predefinita solo per i metodi `GET` quando si abilita il caching per una fase API. Ciò consente di garantire la sicurezza dell'API. Puoi abilitare il caching per altri metodi ignorando le impostazioni del metodo. Per ulteriori informazioni, consulta Abilitazione del caching dell'API per migliorare la velocità di risposta.	20 agosto 2018

Restrizioni dei servizi aggiornate	Sono stati modificati diversi limiti: aumento del numero di APIs account. Limiti di velocità API aumentati per Create/Import/Deploy APIs. Alcuni tassi sono stati corretti da al minuto in al secondo. Per ulteriori informazioni, consulta Limiti.	13 luglio 2018
Sovrascrittura di parametri e intestazioni di richiesta e risposta dell'API	Aggiunto il supporto per sovrascrivere intestazioni di richiesta, stringhe di query e percorsi, nonché intestazioni di risposta e codici di stato. Per ulteriori informazioni, consulta la sezione relativa all'utilizzo di un modello di mappatura per ignorare parametri e intestazioni di richiesta e risposta di un'API.	12 luglio 2018

Throttling a livello di metodo per piani di utilizzo	Aggiunto il supporto per l'impostazione dei limiti di throttling per metodo, nonché l'impostazione dei limiti di throttling per metodi API singoli nelle impostazioni del piano di utilizzo. Queste impostazioni sono in aggiunta ai limiti di throttling a livello di account e ai limiti di throttling a livello di metodo esistenti che puoi definire nelle impostazioni della fase. Per ulteriori informazioni, consulta la sezione relativa al throttling delle richieste API per migliorare le prestazioni.	11 luglio 2018
Le notifiche di aggiornamento della Guida per gli sviluppatori di API Gateway sono ora disponibili tramite RSS	La versione HTML della Guida per gli sviluppatori di API Gateway supporta ora un feed RSS di aggiornamenti che sono documentati nella pagina Cronologia dei documenti. Il feed RSS include gli aggiornamenti effettuati 27 giugno 2018 e quelli successivi. Gli aggiornamenti annunciati in precedenza sono ancora disponibili in questa pagina. Utilizza il pulsante RSS nel pannello del menu in alto per registrarti al feed.	27 giugno 2018

Aggiornamenti precedenti

La tabella seguente descrive le modifiche importanti apportate in ogni versione della Guida per gli sviluppatori di API Gateway prima del 27 giugno 2018.

Modifica	Descrizione	Data della modifica
Privato APIs	È stato aggiunto il supporto per gli endpoint privati APIs, che puoi esporre tramite interfaccia VPC. Il traffico verso la tua rete privata APIs non esce dalla rete Amazon; è isolato dalla rete Internet pubblica.	14 giugno 2018
Provider di autorizzazioni e integrazioni Lambda tra più account e provider di autorizza zioni per il pool di utenti di Amazon Cognito multiacco unt	Usa una AWS Lambda funzione di un AWS account diverso come funzione di autorizzazione Lambda o come backend di integrazione API. Oppure utilizzare un pool di utenti di Amazon Cognito come un provider di autorizza zioni. L'altro account può trovarsi in qualsiasi regione in cui Amazon API Gateway è disponibile. Per ulteriori informazioni, consultare the section called “Configurazione di un'autorizzazione Lambda tra account”, the section called “Tutorial: creazione di una REST API con un'integr azione proxy Lambda multi-account” e the section called “Configurazione dell'autorizzazione Amazon Cognito tra account per un'API REST”.	2 Aprile 2018
Politiche relative alle risorse per APIs	Puoi utilizzare le policy delle risorse API Gateway per consentire agli utenti di un altro AWS account di accedere in modo sicuro alla tua API oppure per autorizzare la chiamata dell'API esclusivamente da intervalli di indirizzi IP di origine specificati o da blocchi CIDR. Per ulteriori informazioni, consulta the section called “Uso delle policy delle risorse API Gateway”.	2 Aprile 2018
Tagging per risorse API Gateway	Come aggiungere fino a 50 tag a una fase API per l'allocaz ione dei costi delle richieste API e il caching in API	19 dicembre 2017

Modifica	Descrizione	Data della modifica
	Gateway. Per ulteriori informazioni, consulta the section called “Configurazione dei tag”.	
Compressione e decompressione dei payload	Come chiamare l'API con payload compressi usando una delle codifiche di contenuto supportate. I payload compressi sono soggetti a mappatura se è specificato il modello di mappatura del corpo. Per ulteriori informazioni, consulta the section called “Codifica dei contenuti”.	19 dicembre 2017
Chiave API restituita da autorizzazioni ad hoc	Come restituire una chiave API da autorizzazioni ad hoc ad API Gateway per applicare un piano di utilizzo per metodi API che richiedono la chiave. Per ulteriori informazioni, consulta the section called “Scelta dell'origine di una chiave API in Gateway API”.	19 dicembre 2017
Autorizzazione con OAuth 2 ambiti	Abilita l'autorizzazione all'invocazione del metodo utilizzan do OAuth 2 ambiti con l'autorizzatore. `COGNITO_U SER_POOLS` Per ulteriori informazioni, consulta the section called “Utilizza un pool di utenti di Amazon Cognito come autorizzazione per un'API REST”.	14 dicembre 2017
Integrazione privata e collegamento VPC	Crea un'API con l'integrazione privata di API Gateway per fornire ai clienti l'accesso alle HTTP/HTTPS risorse in un Amazon VPC dall'esterno del VPC tramite una risorsa. VpcLink Per ulteriori informazioni, consultare the section called “Tutorial: creazione di una REST API con un'integr azione privata” e the section called “Configura un'integr azione privata”.	30 Novembre 2017

Modifica	Descrizione	Data della modifica
Distribuzione Canary per i test dell'API	Come aggiungere una release Canary a una distribuzione API esistente per testare una versione più recente dell'API mantenendo in funzione la versione corrente nella stessa fase. Puoi impostare una percentuale di traffico di stage per la versione Canary e abilitare l'esecuzione e l'accesso specifici per Canary registrati in registri di Logs separati. CloudWatch Per ulteriori informazioni, consulta the section called “Configurazione della distribuzione di una release Canary”.	28 Novembre 2017
Logging degli accessi	Come registrare l'accesso dei client all'API con dati derivati da variabili $context nel formato che preferisci. Per ulteriori informazioni, consulta the section called “CloudWatch registri”.	21 Novembre 2017
SDK Ruby di un'API	Come generare un SDK Ruby per l'API e usarlo per richiamare i metodi API. Per ulteriori informazioni, consulta the section called “Generazione dell'SDK Ruby di un'API in Gateway API” e the section called “Utilizzo di un SDK Ruby generato da API Gateway per un'API REST”.	20 Novembre 2017
Endpoint API regionale	Come specificare un endpoint API regionale per creare un'API per client non mobili. Un client non mobile, come un' EC2 istanza, viene eseguito nella stessa regione in cui viene distribuita l'API. AWS Come per un'API ottimizzata per i confini, puoi creare un nome di dominio personalizzato per un'API regionale. Per ulteriori informazioni, consultar e the section called “Tipi di endpoint di Gateway API” e the section called “Configurazione di un nome di dominio personalizzato regionale”.	2 Novembre 2017

Modifica	Descrizione	Data della modifica
Autorizzazioni ad hoc per le richieste	Come usare autorizzazioni ad hoc per le richieste per fornire informazioni sull'autenticazione utente nei parametri di richiesta in modo da autorizzare le chiamate di metodi API. I parametri di richiesta includono intestazioni e parametri della stringa di query, nonché variabili di fase e di contesto. Per ulteriori informazioni, consulta Uso di autorizzazioni Lambda di API Gateway.	15 settembre 2017
Personalizzazione delle risposte del gateway	Come personalizzare risposte del gateway generate da API Gateway a richieste API che non hanno raggiunto il back-end di integrazione. Un messaggio del gateway personalizzato può fornire al chiamante messaggi di errori personalizzati specifici dell'API, tra cui la restituzione delle intestazioni CORS necessarie, o può trasformare i dati delle risposte del gateway in un formato di uno scambio esterno. Per ulteriori informazioni, consulta Configurazione delle risposte del gateway per la personalizzazione delle risposte agli errori.	6 giugno 2017
Mappatura di proprietà di errore personalizzate di Lambda a intestazioni di risposte dei metodi	Come mappare singole proprietà di errore personalizzate restituite da Lambda ai parametri di intestazione delle risposte dei metodi usando il parametro `integration.response.body`, basandosi su API Gateway per deserializzare l'oggetto errore personalizzato in formato stringa in fase di esecuzione. Per ulteriori informazioni, consulta Gestione degli errori Lambda personalizzati in API Gateway.	6 giugno 2017
Aumento dei limiti di throttling	Come aumentare il limite del tasso di richieste con stato fisso a livello di account a 10.000 richieste al secondo (rps) e il limite di aumento delle prestazioni a 5000 richieste simultanee. Per ulteriori informazioni, consulta Limita le richieste al tuo REST APIs per una migliore velocità di trasmissione in API Gateway.	6 giugno 2017

Modifica	Descrizione	Data della modifica
Convalida delle richieste di metodi	Come configurare validatori di richieste di base a livello di API o a livello di metodo in modo che API Gateway possa convalidare le richieste in ingresso. API Gateway verifica che i parametri obbligatori siano impostati e non siano lasciati vuoti e che il formato dei payload applicabili sia conforme al modello configurato. Per ulteriori informazi oni, consulta Convalida delle richieste per REST API in Gateway API.	11 Aprile 2017
Integrazione con ACM	Come usare certificati ACM per i nomi di dominio personali zzati dell'API. È possibile creare un certificato AWS Certificate Manager o importare un certificato esistente in formato PEM in ACM. Quindi fai riferimento all'ARN del certificato quando imposti un nome di dominio personali zzato per il tuo. APIs Per ulteriori informazioni, consulta Nome di dominio personalizzato per REST pubblico APIs in API Gateway.	9 marzo 2017
Generazione e chiamata di un SDK Java di un'API	Come fare in modo che API Gateway generi l'SDK Java per l'API e come usare l'SDK per chiamare l'API nel client Java. Per ulteriori informazioni, consulta Utilizzo di un SDK Java generato da API Gateway per un'API REST.	13 gennaio 2017
Integrazione con Marketplace AWS	Vendi la tua API in un piano di utilizzo come prodotto SaaS tramite. Marketplace AWS Puoi usare Marketpla ce AWS per estendere la copertura della tua API. Affidati alla fatturazione ai clienti Marketplace AWS per tuo conto. Puoi inoltre lasciare ad API Gateway la gestione dell'auto rizzazione utente e la misurazione dell'utilizzo. Per ulteriori informazioni, consulta Vendi il tuo APIs SaaS.	1 dicembre 2016

Modifica	Descrizione	Data della modifica
Abilitazione del supporto della documentazione per l'API	Aggiungi la documentazione per le entità API nelle DocumentationPartrisorse di API Gateway. Associa un'istantanea delle `DocumentationPart` istanze di raccolta a una fase API per creare un. DocumentationVersion Come pubblicare la documentazione dell'API esportando una versione della documentazione in un file esterno, ad esempio un file Swagger. Per ulteriori informazioni, consulta Documentazione delle REST API in Gateway API.	1 dicembre 2016
Autorizzazioni ad hoc aggiornate	Una funzione Lambda delle autorizzazioni ad hoc restituisce ora l'identificatore dell'entità principale dell'intermediario. La funzione può anche restituire altre informazioni come coppie chiave/valore della mappa `context` e una policy IAM. Per ulteriori informazioni, consulta Output da un sistema di autorizzazione Lambda di Gateway API.	1 dicembre 2016
Supporto di payload binari	binaryMediaTypesImposta la tua API per supportare i payload binari di una richiesta o di una risposta. Imposta la `contentHandling` proprietà su un'integrazione o IntegrationResponsespecifica se gestire un payload binario come blob binario nativo, come stringa codificata in Base64 o come passthrough senza modifiche. Per ulteriori informazioni, consulta Tipi di supporti binari per REST API in Gateway API.	17 Novembre 2016
Abilitazione di un'integrazione proxy con un back-end HTTP o Lambda tramite una risorsa proxy di un'API.	Come creare una risorsa proxy con un parametro di percorso greedy in formato `{proxy+}` e il metodo `ANY` catch-all. La risorsa proxy viene integrata con un back-end HTTP o Lambda usando rispettivamente l'integrazione proxy HTTP o Lambda. Per ulteriori informazioni, consulta Configurazione di un'integrazione proxy mediante una risorsa proxy.	20 settembre 2016

Modifica	Descrizione	Data della modifica
Estendi le offerte di prodotti seleziona ti APIs in API Gateway per i tuoi clienti fornendo uno o più piani di utilizzo.	Come creare un piano di utilizzo in API Gateway per permettere ai client API selezionati di accedere a fasi API specificate a quote e tassi di richiesta concordati. Per ulteriori informazioni, consulta Piani di utilizzo e chiavi API per REST API in Gateway API.	11 agosto 2016
Abilitazione dell'auto rizzazione a livello di metodo con un pool di utenti in Amazon Cognito	Come creare un pool di utenti in Amazon Cognito e usarlo come provider di identità personale. Puoi configurare il pool di utenti come autorizzazione a livello di metodo per concedere l'accesso agli utenti registrati con il pool di utenti. Per ulteriori informazioni, consulta Controllo degli accessi alle REST API utilizzando pool di utenti di Amazon Cognito come sistema di autorizzazione.	28 luglio 2016
Abilitazione CloudWatch delle metriche e delle dimensioni di Amazon nel `AWS/ApiGateway` namespace.	Le metriche dell'API Gateway sono ora standardizzate nel CloudWatch namespace di. `AWS/ApiGateway` Puoi visualizzarli sia nella console API Gateway che nella CloudWatch console Amazon. Per ulteriori informazioni, consulta Dimensioni e parametri di Amazon API Gateway.	28 luglio 2016
Abilitazione della rotazione dei certificati per un nome di dominio personalizzato	La rotazione dei certificati permette di caricare e rinnovare un certificato in scadenza per un nome di dominio personali zzato. Per ulteriori informazioni, consulta Rotazione di un certificato importato in ACM.	27 Aprile 2016

Modifica	Descrizione	Data della modifica
Documentazione delle modifiche per la console Amazon API Gateway aggiornata.	Informazioni su come creare e configurare un'API usando la console API Gateway aggiornata. Per ulteriori informazioni, consulta Tutorial: creazione di un'API REST mediante l'importazione di un esempio e Tutorial: creazione di una REST API con un'integrazione non proxy HTTP.	5 Aprile 2016
Abilitazione della funzionalità dell'API di importazione per creare una nuova API o aggiornarne una esistente da definizioni API esterne.	Con le caratteristiche dell'API di importazione puoi creare una nuova API o aggiornarne una esistente caricando una definizione API esterna espressa in Swagger 2.0 con le estensioni API Gateway. Per ulteriori informazioni sull'API di importazione, consulta Sviluppo di REST API utilizzando OpenAPI in Gateway API.	5 Aprile 2016
Esposizione della variabile `$input.body` per accedere al payload non elaborato come stringa e della funzione `$util.parseJson()` per convertire una stringa JSON in oggetto JSON in un modello di mappatura.	Per ulteriori informazioni su `$input.body` e `$util.parseJson()`, consulta Variabili per le trasformazioni dei dati per Gateway API.	5 Aprile 2016

Modifica	Descrizione	Data della modifica
Abilitazione delle richieste client con invalidamento della cache a livello di metodo e miglioramento della gestione del throttling delle richieste.	Come scaricare una cache a livello di fase API e invalidare singole voci della cache. Per ulteriori informazioni, consulta Scaricare la cache della fase API in API Gateway e Invalidare una voce della cache di API Gateway. Come migliorare l'esperienza della console per gestire il throttling delle richieste API. Per ulteriori informazioni, consulta Limita le richieste al tuo REST APIs per una migliore velocità di trasmissione in API Gateway.	25 marzo 2016
Abilitazione e chiamata di un'API di API Gateway tramite autorizzazioni ad hoc	Crea e configura una AWS Lambda funzione per implementare l'autorizzazione personalizzata. La funzione restituisce il documento di una policy IAM che concede le autorizzazioni per consentire o rifiutare richieste client di un'API di API Gateway. Per ulteriori informazioni, consulta Uso di autorizzazioni Lambda di API Gateway.	11 febbraio 2016
Importazione ed esportazione dell'API di API Gateway usando un file di definizione Swagger e le estensioni	Come creare e aggiornare l'API di API Gateway usando la specifica Swagger con le estensioni API Gateway. Come importare le definizioni Swagger usando lo strumento di importazione di API Gateway. Come esportare un'API di API Gateway in un file di definizione Swagger usando la console API Gateway o l'API di esportazione di API Gateway. Per ulteriori informazioni, consulta Sviluppo di REST API utilizzando OpenAPI in Gateway API e Esportazione di un'API REST da API Gateway.	18 dicembre 2015
Mappatura del corpo della richiesta o della risposta o dei campi JSON del corpo a parametri di richiesta o di risposta.	Come mappare il corpo di richieste di metodi o i relativi i campi JSON nel percorso, nella stringa di query o nelle intestazioni della richiesta di integrazione. Come mappare il corpo della risposta di integrazione o i relativi campi JSON nelle intestazioni della risposta alla richiesta. Per ulteriori informazioni, consulta Esempi di mappatura dei parametri per REST API in Gateway API.	18 dicembre 2015

Modifica	Descrizione	Data della modifica
Utilizzo di variabili di fase in Amazon API Gateway	Informazioni su come associare attributi di configura zione a una fase di distribuzione di un'API in Amazon API Gateway. Per ulteriori informazioni, consulta Utilizzo delle variabili di fase per una REST API in Gateway API.	5 Novembre 2015
Come abilitare CORS per un metodo	È ora più facile abilitare la condivisione di risorse multiorig ine (CORS, Cross-Origin Resource Sharing) per i metodi in Amazon API Gateway. Per ulteriori informazioni, consulta CORS per REST API in Gateway API.	3 Novembre 2015
Come usare l'autenticazione SSL lato client	Come usare Amazon API Gateway per generare certifica ti SSL da usare per autenticare le chiamate al back-end HTTP. Per ulteriori informazioni, consulta Usa un Gateway-generated certificato API per l'autenticazione backend in API Gateway.	22 settembre 2015
Integrazione fittizia di metodi	Informazioni su come eseguire un'integrazione fittizia di un'API con Amazon API Gateway. Questa caratteristica permette agli sviluppatori di generare risposte API da API Gateway direttamente senza che sia prima necessario un back-end di integrazione finale.	1 settembre 2015
Supporto Amazon Cognito Identity	Amazon API Gateway ha ampliato l'ambito della variabile `$context` in modo da restituire informazioni sull'identità di Amazon Cognito Identity quando le richieste sono firmate con credenziali Amazon Cognito. Inoltre, abbiamo aggiunto una `$util` variabile per l'escape dei caratteri e la codifica JavaScript URLs e le stringhe. Per ulteriori informazioni, consulta Variabili per le trasformazioni dei dati per Gateway API.	28 agosto 2015

Modifica	Descrizione	Data della modifica
Integrazione di Swagger	Utilizza lo strumento di importazione Swagger GitHub per importare le definizioni delle API Swagger in Amazon API Gateway. Scopri di più sulla creazione e Estension i OpenAPI per Gateway API la distribuzione e sui metodi di utilizzo dello strumento di APIs importazione. Con lo strumento di importazione Swagger puoi anche aggiornare quelli esistenti. APIs	21 luglio 2015
Informazioni di riferimento sui modelli di mappatura	Informazioni sul parametro `$input` e le relative funzioni in Variabili per le trasformazioni dei dati per Gateway API.	18 luglio 2015
Versione pubblica iniziale	Questa è la versione pubblica iniziale della Guida per gli sviluppatori di API Gateway.	9 luglio 2015

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.