Quickstart: Deploy an API on API Gateway using the gcloud command-line tool

cloud.google.com/api-gateway/docs/quickstart

This page shows you how to deploy an API on API Gateway to secure traffic to a backend service.

Follow the steps below to deploy a new API to access a backend service on Cloud Functions using the `gcloud` command-line tool. This quickstart also describes how to use an API key to protect your backend from unauthorized access.

Before you begin

1. In the Cloud Console, go to the **Dashboard** page and select or create a Google Cloud project.

   Go to the Dashboard page

   Note: If you don't plan to keep the resources you create in this quickstart, create a new project instead of selecting an existing project. After you finish these steps, you can delete the project, removing all resources associated with the project.
2. Confirm that billing is enabled for your project.

   Learn how to enable billing

3. Ensure that the Cloud SDK is downloaded and installed on your machine.

   Download the Cloud SDK

4. Update `gcloud` components:

   ```
   gcloud components update
   ```

5. Set the default project. Replace *PROJECT_ID* with your Google Cloud project ID.

   ```
   gcloud config set project PROJECT_ID
   ```

Enabling required services

API Gateway requires that you enable the following Google services:

Name	Title
`apigateway.googleapis.com`	API Gateway API
`servicemanagement.googleapis.com`	Service Management API
`servicecontrol.googleapis.com`	Service Control API

To confirm that the required services are enabled:

```
gcloud services list
```

If you do not see the required services listed, enable them:

```
gcloud services enable apigateway.googleapis.com
gcloud services enable servicemanagement.googleapis.com
gcloud services enable servicecontrol.googleapis.com
```

For more information about the `gcloud` services, see gcloud services.

Deploying an API backend

API Gateway sits in front of a deployed backend service and handles all incoming requests. In this quickstart, API Gateway routes incoming calls to a Cloud Function backend named `helloGET` that contains the function shown below:

```
/**
 * HTTP Cloud Function.
 * This function is exported by index.js, and is executed when
 * you make an HTTP request to the deployed function's endpoint.
 *
 * @param {Object} req Cloud Function request context.
 *                     More info: https://expressjs.com/en/api.html#req
 * @param {Object} res Cloud Function response context.
 *                     More info: https://expressjs.com/en/api.html#res
 */exports.helloGET = (req, res) => {
  res.send('Hello World!');
};
```

Follow the steps in Quickstart: Using the gcloud command-line tool to download the sample Cloud Functions code and deploy the Cloud Function backend service.

Creating an API

Now you are ready to create your API on API Gateway.

1. Enter the following command, where:

    - *API_ID* specifies the name of your API. See API ID requirements for API naming guidelines.
    - *PROJECT_ID* specifies the name of your Google Cloud project.

    ```
    gcloud api-gateway apis create API_ID --project=PROJECT_ID
    ```

    For example:

    ```
    gcloud api-gateway apis create my-api --project=my-project
    ```

2. On successful completion, you can use the following command to view details about the new API:

   ```
   gcloud api-gateway apis describe API_ID --project=PROJECT_ID
   ```

   For example:

   ```
   gcloud api-gateway apis describe my-api --project=my-project
   ```

   This command returns the following:

   ```
     createTime: '2020-02-29T21:52:20.297426875Z'
     displayName: my-api
     managedService: my-api-123abc456def1.apigateway.my-project.cloud.goog
     name: projects/my-project/locations/global/apis/my-api
     state: ACTIVE
     updateTime: '2020-02-29T21:52:20.647923711Z'
   ```

Note the value of the `managedService` property. This value is used to enable your API in a subsequent step.

Creating an API config

Before API Gateway can be used to manage traffic to your deployed API backend, it needs an API config.

You can create an API config using an OpenAPI spec that contains specialized annotations to define the desired API Gateway behavior. The OpenAPI spec used for this quickstart contains routing instructions to our Cloud Function backend:

```
# openapi2-functions.yaml
swagger: '2.0'
info:
  title: API_ID optional-string
  description: Sample API on API Gateway with a Google Cloud Functions backend
  version: 1.0.0
schemes:
  - https
produces:
  - application/json
paths:
  /hello:
    get:
      summary: Greet a user
      operationId: hello
      x-google-backend:
        address: https://GCP_REGION-PROJECT_ID.cloudfunctions.net/helloGET
      responses:
        '200':
          description: A successful response
          schema:
            type: string
```

To upload this OpenAPI spec and create an API config using the `gcloud` command line tool:

1. From the command line, create a new file named `openapi2-functions.yaml` .
2. Copy and paste the contents of the OpenAPI spec shown above into the newly created file.
3. Edit the file as follows:
    1. In the `title` field, replace *API_ID* with the name of your API and replace *optional-string* with a brief description of your choosing. The value of this field is used when minting API keys that grant access to this API.
    2. In the `address` field, replace *GCP_REGION* with the GCP region of the deployed function and *PROJECT_ID* with the name of your Google Cloud project.
4. Enter the following command, where:
    - *CONFIG_ID* specifies the name of your API config.
    - *API_ID* specifies the name of your API.
    - *PROJECT_ID* specifies the name of your Google Cloud project.
    - *SERVICE_ACCOUNT_EMAIL* specifies the service account created explicitly for creating API configs. For more information, see Configuring a service account.

```
gcloud api-gateway api-configs create CONFIG_ID \
  --api=API_ID --openapi-spec=API_DEFINITION \
  --project=PROJECT_ID --backend-auth-service-account=SERVICE_ACCOUNT_EMAIL
```

For example:

```
gcloud api-gateway api-configs create my-config \
  --api=my-api --openapi-spec=openapi2-functions.yaml \
  --project=my-project --backend-auth-service-account=0000000000000-
compute@developer.gserviceaccount.com
```

This operation may take several minutes to complete as the API config is propagated to downstream systems. Creation of a complex API config could take up to ten minutes to complete successfully.

Note: If you run this command and receive an error in the form `FAILED_PRECONDITION: Service Account ... does not exist` , see Configuring a service account to ensure that a service account is enabled for your project.

5. After the API config is created, you can view its details by running this command:

   ```
   gcloud api-gateway api-configs describe CONFIG_ID \
     --api=API_ID --project=PROJECT_ID
   ```

   For example:

   ```
   gcloud api-gateway api-configs describe my-config \
     --api=my-api --project=my-project
   ```

   The output shows your API config details, including name and state, as shown in the example below:

   ```
   createTime: '2020-02-07T18:17:01.839180746Z'
   displayName: my-config
   gatewayConfig:
   backendConfig:
     googleServiceAccount: 0000000000000-compute@developer.gserviceaccount.com
   name: projects/my-project/locations/global/apis/my-api/configs/my-config
   serviceRollout:
   rolloutId: 2020-02-07r0
   state: ACTIVE
   updateTime: '2020-02-07T18:17:02.173778118Z'
   ```

Creating a gateway

Now deploy the API config on a gateway. Deploying an API config on a gateway defines an external URL that API clients can use to access your API.

Run the following command to deploy the API config you just created to API Gateway:

```
gcloud api-gateway gateways create GATEWAY_ID \
  --api=API_ID --api-config=CONFIG_ID \
  --location=GCP_REGION --project=PROJECT_ID
```

where:

- *GATEWAY_ID* specifies the name of the gateway.
- *API_ID* specifies the name of the API Gateway API associated with this gateway.
- *CONFIG_ID* specifies the name of the API config deployed to the gateway.

- *GCP_REGION* is the Google Cloud region for the deployed gateway.

  Note: Allowed values are:
    - `asia-east1`
    - `asia-northeast1`
    - `australia-southeast1`
    - `europe-west1`
    - `europe-west2`
    - `us-east1`
    - `us-east4`
    - `us-central1`
    - `us-west2`
    - `us-west3`
    - `us-west4`
- *PROJECT_ID* specifies the name of your Google Cloud project.

For example:

```
gcloud api-gateway gateways create my-gateway \
  --api=my-api --api-config=my-config \
  --location=us-central1 --project=my-project
```

On successful completion, use the following command to view details about the gateway:

```
gcloud api-gateway gateways describe GATEWAY_ID \
  --location=GCP_REGION --project=PROJECT_ID
```

For example:

```
gcloud api-gateway gateways describe my-gateway \
  --location=us-central1 --project=my-project
```

This command returns the following:

```
apiConfig: projects/my-project/locations/global/apis/my-api/configs/my-config
createTime: '2020-02-05T13:44:12.997862831Z'
defaultHostname: my-gateway-a12bcd345e67f89g0h.uc.gateway.dev
displayName: my-gateway
name: projects/my-project/locations/us-central1/gateways/my-gateway
serviceAccount:
      email: 0000000000000-compute@developer.gserviceaccount.com
state: ACTIVE
updateTime: '2020-02-05T13:45:00.844705087Z'
```

Note the value of the `defaultHostname` property. This is the hostname portion of the gateway URL you use to test your deployment in the next step.

Testing your API deployment

Now you can send requests to your API using the URL generated upon deployment of your gateway.

Enter the following `curl` command, where:

- *DEFAULT_HOSTNAME* specifies the hostname portion of your deployed gateway URL.
- `hello` is the path specified in your API config.

```
curl https://DEFAULT_HOSTNAME/hello
```

For example:

```
curl https://my-gateway-a12bcd345e67f89g0h.uc.gateway.dev/hello
```

The output is:

```
Hello World!
```

You have successfully created and deployed an API Gateway!

Securing access by using an API key

To secure access to your API backend, generate an API key associated with your project and grant that key access to call your API. See Restricting API access with API keys for more information.

If you do not already have an API key associated with the Google Cloud project you are using in this quickstart, you can add one by following the steps at Creating an API Key.

To secure access to your gateway using an API key:

1. Enable API key support for your service. Enter the following command, where:
    - *API_ID* specifies the name of your API.
    - *HASH* is the unique hash code generated when you deployed the API.
    - *PROJECT_ID* specifies the name of your Google Cloud project.

    ```
    gcloud services enable API_ID-HASH.apigateway.PROJECT_ID.cloud.goog
    ```

    For example:

    ```
    gcloud services enable my-api-123abc456def1.apigateway.my-project.cloud.goog
    ```

2. Modify the OpenAPI spec used to create your API config to include instructions to enforce an API key validation security policy on all traffic. Add the `security` type and `securityDefinitions` as shown below:

```
  # openapi2-functions.yaml
  swagger: '2.0'
  info:
    title: API_ID optional-string
    description: Sample API on API Gateway with a Google Cloud Functions
backend
    version: 1.0.0
  schemes:
    - https
  produces:
    - application/json
  paths:
    /hello:
      get:
        summary: Greet a user
        operationId: hello
        x-google-backend:
          address: https://GCP_REGION-PROJECT_ID.cloudfunctions.net/helloGET
        security:
        - api_key: []
        responses:
          '200':
            description: A successful response
            schema:
              type: string
  securityDefinitions:
    # This section configures basic authentication with an API key.
    api_key:
      type: "apiKey"
      name: "key"
      in: "query"
```

The `securityDefinition` configures your API to require an API key passed as a query parameter named `key` when requesting access to all paths defined in the spec.

3. Create a new API config with the modified OpenAPI spec using the following command:

```
gcloud api-gateway api-configs create NEW_CONFIG_ID \
--api=API_ID --openapi-spec=NEW_API_DEFINITION \
--project=PROJECT_ID --backend-auth-service-account=SERVICE_ACCOUNT_EMAIL
```

For example:

```
gcloud api-gateway api-configs create my-config-key \
  --api=my-api --openapi-spec=openapi2-functions.yaml \
  --project=my-project --backend-auth-service-
account=0000000000000compute@developer.gserviceaccount.com
```

4. Run the following command to update your existing gateway with the new API config:

   ```
   gcloud api-gateway gateways update GATEWAY_ID \
     --api=API_ID --api-config=NEW_CONFIG_ID \
     --location=GCP_REGION --project=PROJECT_ID
   ```

   For example:

   ```
   gcloud api-gateway gateways update my-gateway \
     --api=my-api --api-config=my-config-key \
     --location=us-central1 --project=my-project
   ```

Testing your API key

Once you have created and deployed the modified API, try making a request to it.

Enter the following `curl` command, where:

- *DEFAULT_HOSTNAME* specifies the hostname portion of your deployed gateway URL.
- `hello` is the path specified in your API config.

```
curl https://DEFAULT_HOSTNAME/hello
```

For example:

```
curl https://my-gateway-a12bcd345e67f89g0h.uc.gateway.dev/hello
```

This should result in the following error:

```
UNAUTHENTICATED:Method doesn't allow unregistered callers (callers without
established identity). Please use API Key or other form of API consumer identity
to call this API.
```

Now, enter the following `curl` command where:

- *DEFAULT_HOSTNAME* specifies the hostname portion of your deployed gateway URL.
- `hello` is the path specified in your API config.
- *API_KEY* specifies the API key you created in the previous step.

```
curl https://DEFAULT_HOSTNAME/hello?key=API_KEY
```

Now you should see `Hello World!` in the response from your API.

Congratulations! You have successfully protected your API backend with an API Gateway. Now you can start onboarding new API clients by generating additional API keys.

Clean up

To avoid incurring charges to your Google Cloud account for the resources used in this quickstart, you can delete your API and delete your gateways. You can also delete the Google Cloud project used for this tutorial.

What's next

- Learn more About API Gateway
- Walk through Configuring the development environment

Rate and review