


API Gateway: Qwik Start

Activate Cloud Shell

Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

1. Click **Activate Cloud Shell**  at the top of the Google Cloud console.
2. Click through the following windows:
 - Continue through the Cloud Shell information window.
 - Authorize Cloud Shell to use your credentials to make Google Cloud API calls.

When you are connected, you are already authenticated, and the project is set to your **Project_ID**, `qwiklabs-gcp-03-f4b490688069`. The output contains a line that declares the **Project_ID** for this session:

```
Your Cloud Platform project in this session is set to qwiklabs-gcp-03-f4b490688069
```

`gcloud` is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

3. (Optional) You can list the active account name with this command:
`gcloud auth list`

4. Click **Authorize**.
Output:

```
ACTIVE: *
ACCOUNT: student-01-281f827c45c4@qwiklabs.net

To set the active account, run:
$ gcloud config set account `ACCOUNT`
```

5. (Optional) You can list the project ID with this command:
`gcloud config list project`

Output:

```
[core]  
project = qwiklabs-gcp-03-f4b490688069
```

Note: For full documentation of `gcloud`, in Google Cloud, refer to [the gcloud CLI overview guide](#).

Set the region

Set the project region for this lab:

```
gcloud config set compute/region us-west2
```

Enable the required APIs

1. In the Cloud Console, on the Navigation menu (☰) click **APIs & Services > Library**.
2. Start typing "api gateway" in the **Search** bar, then select the **API Gateway API** tile.
3. Now click the **Enable** button on the next screen.

Task 1. Deploying an API backend

API Gateway sits in front of a deployed backend service and handles all incoming requests. In this lab, 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!');
};

```

1. In Cloud Console, clone the Cloud Function sample repository:

```
git clone https://github.com/GoogleCloudPlatform/nodejs-docs-samples.git
```

2. Change to the directory that contains the Cloud Functions sample code:

```
cd nodejs-docs-samples/functions/helloworld/helloworldGet
```

3. To deploy the function with an HTTP trigger, run the following command in the directory containing your function:

```
gcloud functions deploy helloGET --runtime nodejs20 --trigger-http --allow-unauthenticated --region us-west2
```

- If prompted, enter **Y** to enable the API.

Note: If you receive a request to permit the gcloud command with your credentials click **Authorize**. It will take a few minutes to deploy the cloud function. **Wait for the operation to complete before proceeding.** **Warning:** If you receive an Error as **IamPermissionDeniedException** rerun the above command.

Task 2. Test the API backend

1. When the function finishes deploying, take note of the `httpsTrigger`'s url property or find it using the following command:

```
gcloud functions describe helloGET --region us-west2
```

The output should look similar to the URL below where `PROJECT_ID` is a value specific to your project.

2. Set your `PROJECT_ID` as a variable:

```
export PROJECT_ID=qwiklabs-gcp-03-f4b490688069
```

Visit the URL to invoke the Cloud Function. You should see the message `Hello World!` as the response:

```
curl -v https://us-west2-qwiklabs-gcp-03-  
f4b490688069.cloudfunctions.net/helloGET
```

Create the API definition

API Gateway uses an API definition to route calls to the backend service. You can use an OpenAPI spec that contains specialized annotations to define the desired API Gateway behavior. The OpenAPI spec for this quickstart contains routing instructions to the Cloud Function backend.

1. From Cloud Shell, navigate back to your home directory:
`cd ~`

2. Create a new file named `openapi2-functions.yaml`:
`touch openapi2-functions.yaml`

Copy and paste the contents of the OpenAPI spec shown below into the newly created file:

```
# openapi2-functions.yaml  
swagger: '2.0'  
info:  
  title: API_ID description  
  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://us-west2-qwiklabs-gcp-03-  
f4b490688069.cloudfunctions.net/helloGET  
      responses:  
        '200':  
          description: A successful response  
          schema:  
            type: string
```

4. Set the following environment variables:
`export API_ID="hello-world-$(cat /dev/urandom | tr -dc 'a-z' | fold -w
${1:-8} | head -n 1)"`

5. Run the following commands to replace the variables set in the last step in the OpenAPI spec file:

```
sed -i "s/API_ID/${API_ID}/g" openapi2-functions.yaml
sed -i "s/PROJECT_ID/${PROJECT_ID}/g" openapi2-functions.yaml
```

Task 3. Creating a gateway

Now you are ready to create and deploy a gateway on API Gateway.

1. In the top search bar enter **API Gateway** and select it from the options that appear.

2. Click **Create Gateway**. Then, in the **APIs** section:

- Ensure the **Select an API** input is set to **Create new API**.
- For **Display Name** enter `Hello World API`
- For **API ID**, run the following command to once again obtain the API ID and enter it into the **API ID** field:

```
export API_ID="hello-world-$(cat /dev/urandom | tr -dc 'a-z' | fold -w
${1:-8} | head -n 1)"
echo $API_ID
```

3. In the **API Config** section:

- Ensure the **Select a Config** input is set to **Create new API config**.
- Do the following to upload the `openapi2-functions.yaml` file previously created.

4. In Cloud Shell, run the following command:

```
cloudshell download $HOME/openapi2-functions.yaml
```

5. Click **Download**.

Note: The file `openapi2-functions.yaml` is now downloaded to your local machine.

6. Select **Browse** and select the file from the browser's download location:

- Enter `Hello World Config` in the **Display Name** field.
- Ensure the **Select a Service Account** input is set to **Compute Engine default service account**.

7. In the **Gateway details** Section:

- Enter `Hello Gateway` in the **Display Name** field.
- Set the **Location** drop down to `us-west2`.

8. Click **Create Gateway**.

Note: It will take several minutes (~10 minutes) for the **Create Gateway** operation to complete. To check the status of the creation and deployment process, you can click the Notification icon in the main navigation bar to display a status notification, as shown in the image below. Please ensure that the icon status has a green check next to it before proceeding.

Testing your API Deployment

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

1. In Cloud Shell, enter the following command to retrieve the `GATEWAY_URL` of the newly created API hosted by API Gateway:

```
export GATEWAY_URL=$(gcloud api-gateway gateways describe hello-gateway
--location us-west2 --format json | jq -r .defaultHostname)
```

2. Run the following command to ensure that the `GATEWAY_URL` environment variable is set:

```
echo $GATEWAY_URL
```

If it is not, that means you will need to **wait longer** for the API Gateway to be deployed.

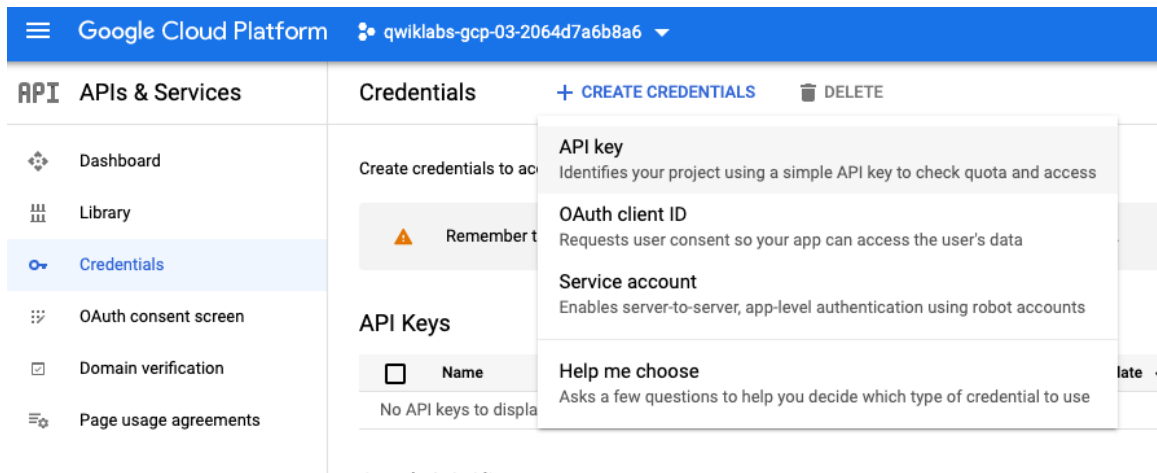
3. Run the following curl command and validate that the response returned is `Hello World!!`:

```
curl -s -w "\n" https://$GATEWAY_URL/hello
```

Task 4. Securing access by using an API key

To secure access to your API backend, you can generate an API key associated with your project and grant that key access to call your API. To create an API Key you must do the following:

1. In the Cloud Console, navigate to **APIs & Services > Credentials**.
2. Select **Create credentials**, then select **API Key** from the dropdown menu. The **API key created** dialog box displays your newly created key.



3. Copy the API Key from the dialog, then click on **close**.

4. Store the API Key value in Cloud Shell by running the following command:

```
export API_KEY=REPLACE_WITH_COPIED_API_KEY
```

Now, enable the API Key support for your service.

1. In Cloud Shell, obtain the name of the Managed Service you just created using the following command:

```
MANAGED_SERVICE=$(gcloud api-gateway apis list --format json | jq -r
.[0].managedService | cut -d'/' -f6)
echo $MANAGED_SERVICE
```

2. Then, using the Managed Service name of the API you just created, run this command to **enable** API key support for the service:

```
gcloud services enable $MANAGED_SERVICE
```

Modify the OpenAPI Spec to leverage API Key Security

In this section, modify the API config of the deployed API to enforce an API key validation security policy on all traffic.

1. Add the `security` type and `securityDefinitions` sections to a new file called `openapi2-functions2.yaml` file as shown below:

```
touch openapi2-functions2.yaml
```

2. Copy and paste the contents of the OpenAPI spec shown below into the newly created file:

```
# openapi2-functions.yaml
```

```

swagger: '2.0'
info:
  title: API_ID description
  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://us-west2-qwiklabs-gcp-03-
f4b490688069.cloudfunctions.net/helloGET
      security:
        - api_key: []
      responses:
        '200':
          description: A successful response
          schema:
            type: string
securityDefinitions:
  api_key:
    type: "apiKey"
    name: "key"
    in: "query"

```

3. Run the following commands to replace the variables set in the last step in the OpenAPI spec file:

```

sed -i "s/API_ID/${API_ID}/g" openapi2-functions2.yaml
sed -i "s/PROJECT_ID/${PROJECT_ID}/g" openapi2-functions2.yaml

```

4. Download the updated API spec file, you will use it to update the Gateway config in the next step:

```
cloudshell download $HOME/openapi2-functions2.yaml
```

5. Click **Download**.

Task 5. Create and deploy a new API config to your existing gateway

1. Open the **API Gateway** page in Cloud Console. (Click **Navigation Menu > API Gateway**.)
2. Select your API from the list to view details.

3. Select the **Gateways** tab.
4. Select `Hello Gateway` from the list of available **Gateways**.
5. Click on `Edit` at the top of the Gateway page.
6. Under **API Config** change the drop down to `Create new API config`.
7. Click **Browse** in the **Upload an API Spec** input box and select the `openapi2-functions2.yaml` file.
8. Enter `Hello Config` for **Display Name**.
9. Select `Qwiklabs User Service Account` for **Select a Service Account**.
10. Click **Update**.

Note: It may take a few minutes for the **Update Gateway** operation to complete. To check the status of the creation and deployment process, you can click the Notification icon in the main navigation bar to display a status notification, as shown in the image below. Please ensure that the icon status has a green check next to it before proceeding.

Click **Check my progress** to verify the objective.

Please update a Hello Gateway by creation new api config named as 'Hello Config' with specific configurations.

Create and deploy a new API config to your existing gateway

Check my progress

Please update a Hello Gateway by creation new api config named as 'Hello Config' with specific configurations.

Task 6. Testing calls using your API key

1. To test using your API key run the following command:

```
export GATEWAY_URL=$(gcloud api-gateway gateways describe hello-gateway
--location us-west2 --format json | jq -r .defaultHostname)
curl -sL $GATEWAY_URL/hello
```

You should see a response similar to the following error as an API key was not supplied with the `curl` call: `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.

2. Run the following `curl` command with the `key` query parameter and use the API key previously created to call the API:

```
curl -sL -w "\n" $GATEWAY_URL/hello?key=$API_KEY
```

If you do not have the `API_KEY` environment variable set you can get your API key from the left menu by navigating **APIs & Services > Credentials**. The key will be available under the **API Keys** section.

The response returned from the API should now be `Hello World!`.

Note: You may need to run this command more than once to obtain the desired result.