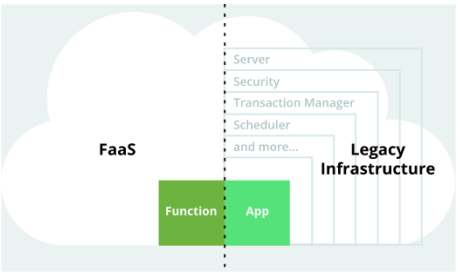
Deploy HTTP Cloud Functions in Java on API Gateway using the `gcloud` command-line tool

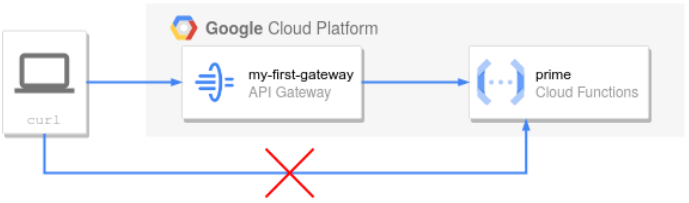
Lorenzo Ferron (20024182)

Overview



This tutorial shows you how to deploy an API on API Gateway to access a backend service on Cloud Functions using the `gcloud` command-line tool. In particular our HTTP cloud function will check if a number is prime.

Cloud Functions is an event-driven serverless compute platform. Cloud Functions allows you to write your code without worrying about provisioning resources or scaling to handle changing requirements. Concerns like "how many instances to run" and "what operating system to use" are all managed by Function as a Service platform (or FaaS), leaving developers free to focus on business logic.



So you will be able to send an HTTP request to an API Gateway that will route it to backend service. Nevertheless even if the HTTP cloud function will have an URL you **won't** be able to send an HTTP request directly to it.

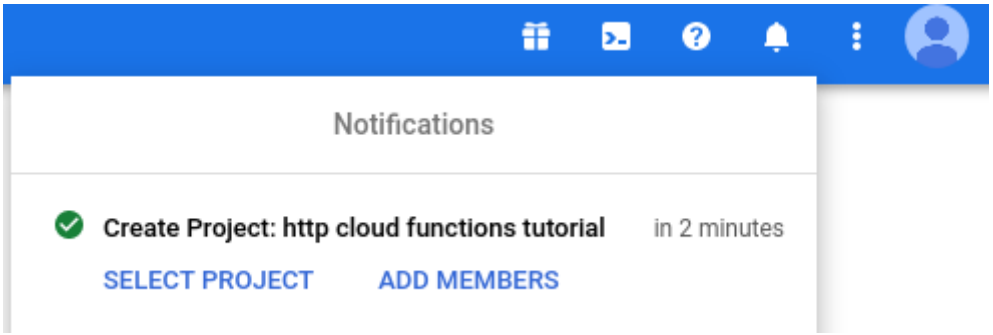
Before you begin

(Optional) Creating a new project

It is really necessary?

Well, if you don't plan to keep the resources you create in this tutorial, create a new project instead of selecting an existing one. After you finish these steps, you can delete the project, removing all resources associated with the project.

1. Go to the Manage resources page in the Cloud Console.
2. Click **Create Project**.
3. In the **New Project** window that appears, enter a project name, in this tutorial "http cloud functions tutorial".
4. Click **Create**.



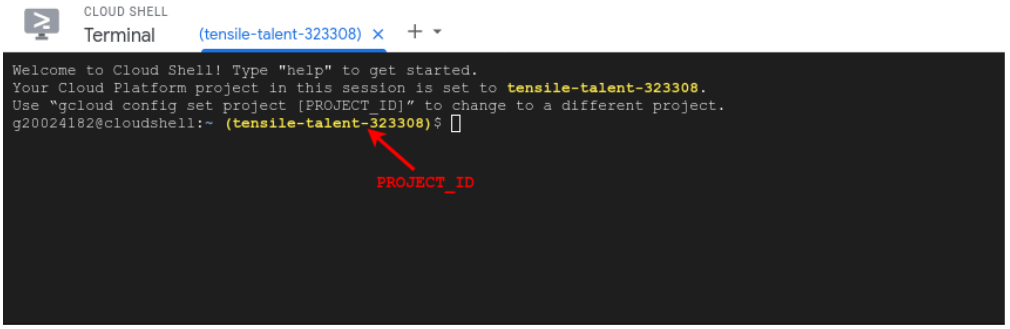
5. Click **Select Project** from notification alert.

Open Cloud Shell

For this tutorial, we will use Cloud Shell: a command-line prompt inside Cloud Console to avoid installing software on your local machine. But if you prefer installing software locally see Cloud SDK. In the latter case you **must** add `--project=PROJECT_ID` option to commands shown in the next sections.

1. Click the **Activate Cloud Shell** button at the top of the Cloud Console window.

A Cloud Shell session opens inside a new frame at the bottom of the Cloud Console and displays



a command-line prompt.

Get source code

Copy in command-line prompt:

```
$ cd /tmp/
$ git clone https://gitlab.di.unipmn.it/20024182/http-functions-gcp.git
$ cd http-functions-gcp/
```

Setup services

Now you are ready to deploy a Cloud Function, create API, create API config, deploy it on API Gateway and test everything. To do these things you can use:

- a **non-interactive** `bash` script, or

- the instructions below.

TL;DR

To do nothing you can run these commands:

```
$ bash ./deploy.sh # only ./deploy.sh not work, because FS is mounted with
                   # "noexec" flag
```

It takes several minutes. In any case, for the **first time it is recommended** to read the following section to understand the steps.

Long version

In this section we illustrate the functions and constants into `deploy.sh` that you can read with

```
$ cat deploy.sh
```

To set up services you can copy commands—that start with **`$`**—into the command-line prompt.

A bit of constants

```
$ readonly API_ID='prime'
$ readonly API_DEFINITION='swagger.yaml'
$ readonly CONFIG_ID='prime-config'
$ readonly GATEWAY_ID='my-first-gateway'
```

where:

- `API_ID` specifies the name of your API.
- `API_DEFINITION` specifies the path of your API config.
- `CONFIG_ID` specifies the name of your API config.
- `GATEWAY_ID` specifies the name of the gateway.

You will find these constants in the next code snippets (light gray box).

`enable_apis()`

Before proceeding you must enable some Google services. In particular API Gateway requires:

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

Whereas Cloud Function requires:

Name	Title
`cloudfunctions.googleapis.com`	Cloud Functions API
`cloudbuild.googleapis.com`	Cloud Build API

So you can run following commands:

```
1 $ gcloud services enable apigateway.googleapis.com
2 $ gcloud services enable servicemanagement.googleapis.com
3 $ gcloud services enable servicecontrol.googleapis.com
4 $ gcloud services enable cloudfunctions.googleapis.com
5 $ gcloud services enable cloudbuild.googleapis.com
```

Note that in line 4 not only you enable the service, but implicitly you create a default service account named "App Engine default service account". This is not a user account.
A service account is a special type of Google account intended to represent a non-human user that needs to authenticate and be authorized to access data in Google APIs. So service accounts do not have passwords, and **cannot log in** via browsers or cookies. For our purpose "App Engine default service account" is sufficient. However you can create your own services account with specific permissions. But remember the principle of least privilege.

`deploy_cloud_function()`

Now we deploy an HTTP cloud function named "prime" that requires authentication.

```
1 $ gcloud functions deploy prime --quiet --region=europe-west1 \
    --entry-point 'functions.Prime' --trigger-http --runtime java11
```

where:

- `--region` specifies the Cloud region for the function.
- `--entry-point` is the fully-qualified class name that implements `service` method from `HttpFunction` interface.
- `--trigger-http` assigns endpoint to function. Any HTTP request (of a supported type) to the endpoint will trigger function execution. Supported HTTP request types are: `POST`, `PUT`, `GET`, `DELETE`, and `OPTIONS`.
- `--runtime` specifies the runtime in which to run the function.

`setup_api_gateway()`

```
1 $ gcloud api-gateway apis create "${API_ID}"
2 $ sed -ie "s@\${ADDR}@$(gcloud functions describe prime
  --region=europe-west1 --format='value(httpsTrigger.url)')@"
  "${API_DEFINITION}"
3 $ gcloud api-gateway api-configs create "${CONFIG_ID}" \
      --api="${API_ID}" --openapi-spec="${API_DEFINITION}" \
      --backend-auth-service-account="$(gcloud iam service-accounts list
  --filter="displayName:App Engine default service account"
  --format="value(email)")"
4 $ gcloud api-gateway gateways create "${GATEWAY_ID}" \
      --api="${API_ID}" --api-config="${CONFIG_ID}" \
      --location=us-central1
```

In line 1 we create a new API named "prime", see `${API_ID}` constant above. In line 2 `sed` replaces the placeholder `${ADDR}` in `swagger.yaml` with the URL generated during deploying HTTP cloud function in the previous section. In line 3 we create an API config for "prime" named "prime-config". To do this we use OpenAPI spec that contains specialized annotations to define the desired API Gateway behavior. The OpenAPI spec used for this tutorial contains routing instructions to our Cloud Function backend. In particular, the OpenAPI spec is included in `swagger.yaml`. Moreover we use `--backend-auth-service-account` option to specify the email of "App Engine default service account". So API Gateway will use this account to execute "prime" cloud function. Finally in line 4, we create a new gateway for `${API_ID}` with `${CONFIG_ID}` config in `us-central1`, see `--location` option.
Note that the HTTP cloud function is in `europe-west1`, while API Gateway is in `us-central1`.

`print_endpoint()`

```
1 $ echo "Your endpoint is https://$(gcloud api-gateway gateways describe
  "${GATEWAY_ID}" --location=us-central1
  --format="value(defaultHostname)")/prime"
```

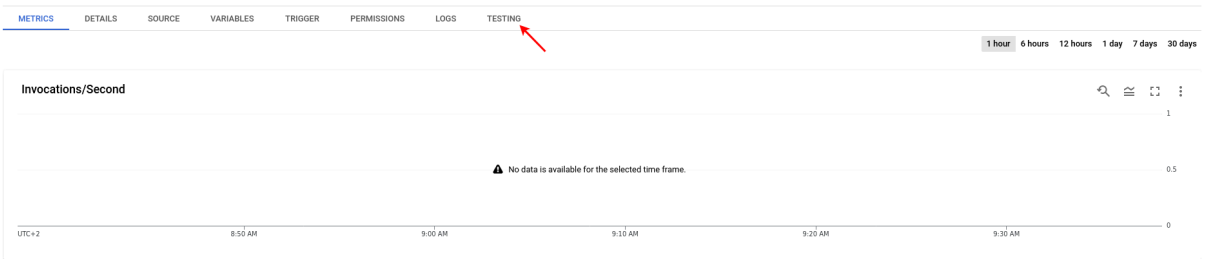
Prints the endpoint URL that you have just created.

Testing

We can perform two types of test using Cloud Console or `curl`.

Cloud console

1. Go to the Cloud Functions page in the Cloud Console.
2. Select the function named "prime".
3. Click tab **TESTING**



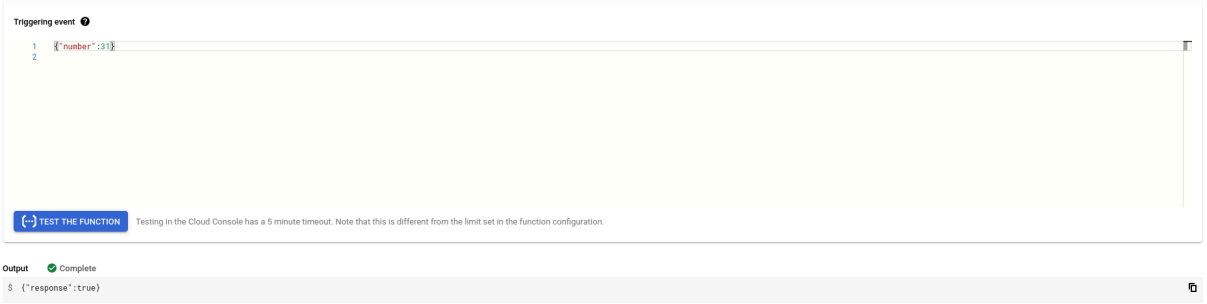
4. Copy in **Triggering event**

```
{"number":31}
```

5. Click **TEST THE FUNCTION**. So you achieve as output

```
{"response":true}
```

Because 31 is a prime number.



`curl`

Open a terminal on **your local machine** and type:

```
$ curl --location --request POST ${ENDPOINT} \
    -s -w " | HTTP Status-Code: %{response_code}\n" \
    --header 'Content-Type: application/json' \
    --data-raw '{
      "number": "5832158674351"
    }'
```

where:

- `${ENDPOINT}` is the output of `echo` command in the `print_endpoint()` function.

Output achieved

```
<response> | <HTTP Status-Code>
```

where:

- `<response>` is a JSON string that contains `true` if the number in `--data-raw` option is prime, `false` otherwise. If an error occurred this field is empty.
- `<HTTP Status-Code>` a numerical constant defined in RFC2616.

Cleaning up

Note

If you **didn't create** a new project as explain in section "Creating a new project", you will find created resources in **API Gateway** and **Cloud Functions** services.

Finally, we are now ready to delete our project.

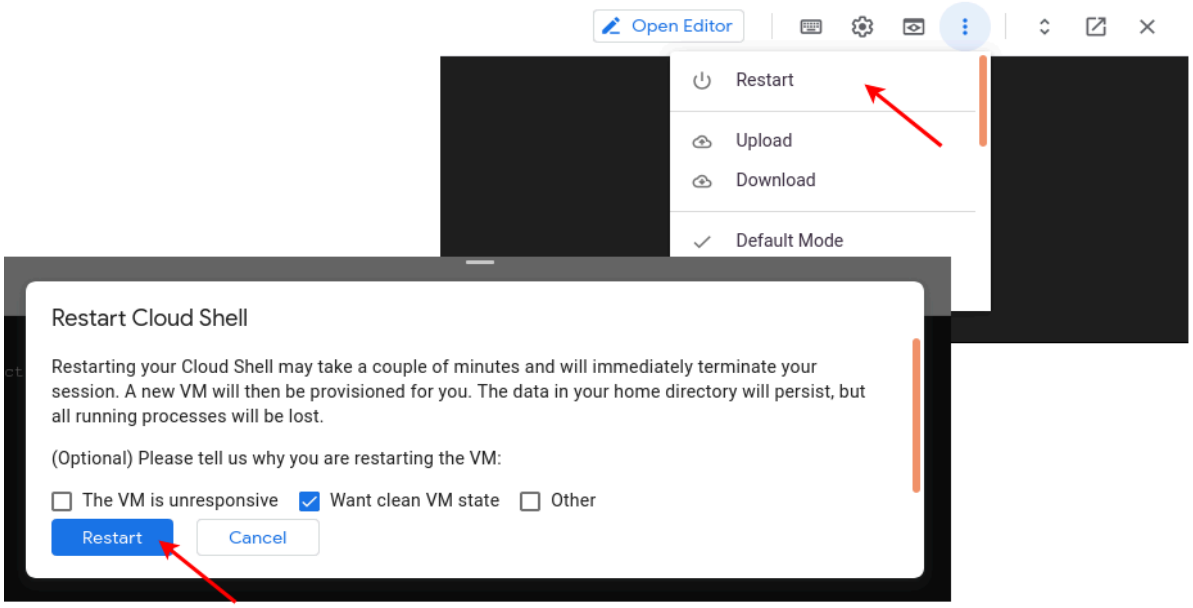
1. Go to the Manage resources page in the Cloud Console.
2. Click the **Activate Cloud Shell** button at the top of the Cloud Console window.

3. Copy in command-line prompt

```
1 $ PROJECT_ID=$(gcloud projects list --filter="name:'http cloud
functions tutorial'" --format="value(projectId)")
2 $ gcloud endpoints services delete --quiet \
    $(gcloud endpoints services list
--filter="serviceConfig.title:prime" --format="value(serviceName)"
--project="${PROJECT_ID}")
3 $ gcloud projects delete $(gcloud projects list --filter="name:'http
cloud functions tutorial'" --format="value(projectId)") --quiet
4 $ gcloud beta billing projects unlink ${PROJECT_ID}
```

   In line 2 we remove the endpoint from the project. Whereas in line 4 we unlink the project from bill to not risk exceeding the quota.

4. (Optional) To clean `/tmp/` folder used in Cloud Shell you can restart the VM.



Further details

If you are curious about "prime" cloud function you can download source code here. You can find some instructions to deploy function on your local machine.

Troubleshooting

If there are some problems with the `gcloud` command you can use `--verbosity=debug` option. But in general a problem can be:

- deprecated option in `gcloud`;
- a subcommand not anymore in beta, see `gcloud beta billing projects unlink`; or
- deprecated/decommissioned runtime in `--runtime` option.