Serverless Applications
Google Cloud Functions

Outline

- Google Cloud Functions + API Management
- Documentation available at:

  https://cloud.google.com/functions
- Scalable pay as you go Functions-as-a-Service (FaaS) to run your code with zero server management.
  - No servers to provision, manage, or upgrade
  - Automatically scale based on the load
  - Integrated monitoring, logging, and debugging capability
  - Built-in security at role and per function level based on the principle of least privilege
  - Key networking capabilities for hybrid and multi-cloud scenarios

Google Cloud Functions

Why Google Cloud Functions?

- **Serverless Applications on Google's Infrastructure.** Construct applications from bite-sized business logic **billed** to the nearest **100 milliseconds**, only while your code is running. Serve users from zero to planet-scale, all without managing any infrastructure.

Microservices Over Monoliths

- Developer agility comes from building systems composed of small, independent units of functionality focused on doing one thing well. Cloud Functions lets you build and deploy services at the level of a single function, not at the level of entire applications, containers, or VMs.

Connect & Extend Cloud Services

- Cloud Functions provides a connective layer of logic that lets you write code to connect and extend cloud services. Listen and respond to events such as a file upload to Cloud Storage, an incoming message on a Cloud Pub/Subtopic, a log change in Stackdriver Logging, or a mobile-related event from Firebase.

Serverless Economics

- Cloud Functions are **ephemeral**, spinning up on-demand and back down in response to events in the environment. Pay only while your function is executing, metered to the nearest 100 milliseconds, and pay nothing after your function finishes.

Google Cloud Functions (cont'd)

Mobile Ready

- Mobile app developers can use Cloud Functions directly from **Firebase**, Google Cloud's mobile platform. Firebase natively emits events to which Cloud Functions can respond, including from Firebase Analytics, Realtime Database, Authentication, and Storage.

Just Add Code

- Run in a fully-managed, serverless environment where Google handles servers, operating systems, and runtime environments completely on your behalf. Each Cloud Function runs in its own isolated secure execution context, scales automatically, and has a lifecycle independent from other functions.

Open and Familiar

- **Cloud Functions are written in JavaScript and** execute in a standard **Node.js runtime** environment. **Python** and **Go** are also supported. We don't assume anything proprietary all the way down to the operating system, which means your functions will just work—including native libraries you bring to the platform. Discover a superior, open developer experience that comes from working hand-in-hand with the Node.js Foundation, with our Google colleagues and with the community through the **open source V8 engine**.

Google Cloud Functions Use Cases

Mobile Backend

- Use Google's mobile platform for app developers, Firebase, and extend your mobile backend with Cloud Functions. Listen and respond to events from Firebase Analytics, Realtime Database, Authentication, and Storage.

APIs & Microservices

- Compose applications from lightweight, loosely coupled bits of logic that are quick to build and scale automatically. Your functions can be event-driven or invoked directly over HTTP/S.

Data Processing / ETL

- Listen and respond to Cloud Storage events such as when a file is created, changed, or removed. Process images, do video transcoding, validate or transform data, and invoke any service on the Internet from your Cloud Function.

Webhooks

- Via a simple HTTP trigger, respond to events originating from 3rd party systems like GitHub, Slack, Stripe, or from anywhere that can send HTTP/S requests.

IoT

- Imagine tens or hundreds of thousands of devices streaming data into Cloud Pub/Sub automatically launching Cloud Functions to process, transform and store data. Cloud Functions lets you do this in a way that's completely serverless.

Google Cloud Functions Features

Cloud Pub/Sub Triggers

- Cloud Functions can be triggered by messages on a Cloud Pub/Sub topic, and multiple functions can subscribe to the same topic.

Cloud Storage Triggers

- You can associate a Cloud Function to mutation events on a Cloud Storage bucket. Every time a file in your bucket is created, deleted or modified, your function will execute.

Firebase Triggers

- Mobile developers will find first-class integration between Firebase and Cloud Functions.

HTTP/S Invocation

- Functions deployed with an HTTP trigger are given a fully qualified domain together with a dynamically generated TLS certificate for secure communication.

GitHub/Bitbucket

- Using Cloud Source Repositories you can deploy Cloud Functions directly from your Github or Bitbucket repository without needing to upload code or manage versions yourself.

Logging, Monitoring & Debugging

- Logs emitted from your Cloud Functions are automatically written to Stackdriver Logging and performance telemetry is recorded in Stackdriver Monitoring. Stackdriver Debugger lets you investigate your code’s behavior in production.

Google Cloud Functions Pricing

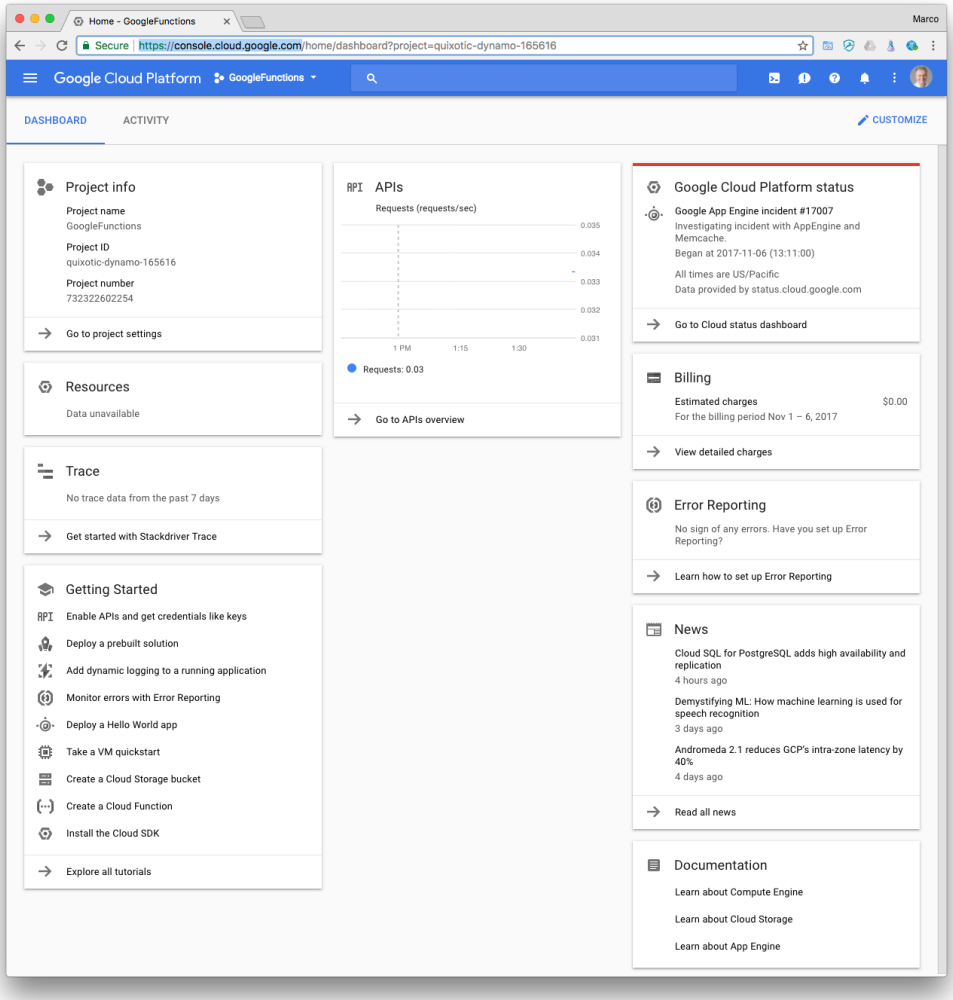
	FREE *LIMIT PER MONTH*	PRICE *ABOVE FREE LIMIT (PER UNIT)*	PRICE UNIT
Invocations *	2 million invocations	$0.40	per million invocations
Compute Time	400,000 GB-seconds	$0.0000025	per GB-Second
	200,000 GHz seconds	$0.0000100	per GHz-Second
Outbound Data (Egress)	5GB	$0.12	per GB
Inbound Data (Ingress)	Unlimited	Free	per GB
Outbound Data to Google APIs in same region	Unlimited	Free	per GB

* Includes both Background and HTTP Functions.

Create a Simple HTTP service using Google Cloud Functions

In this exercise you will demonstrate writing, deploying, and triggering an HTTP Cloud Function. The Cloud Function is triggered by and HTTP request and outputs a “Hello World” in our browser. This tutorial uses billable components of Cloud Platform, including Google Cloud Functions.

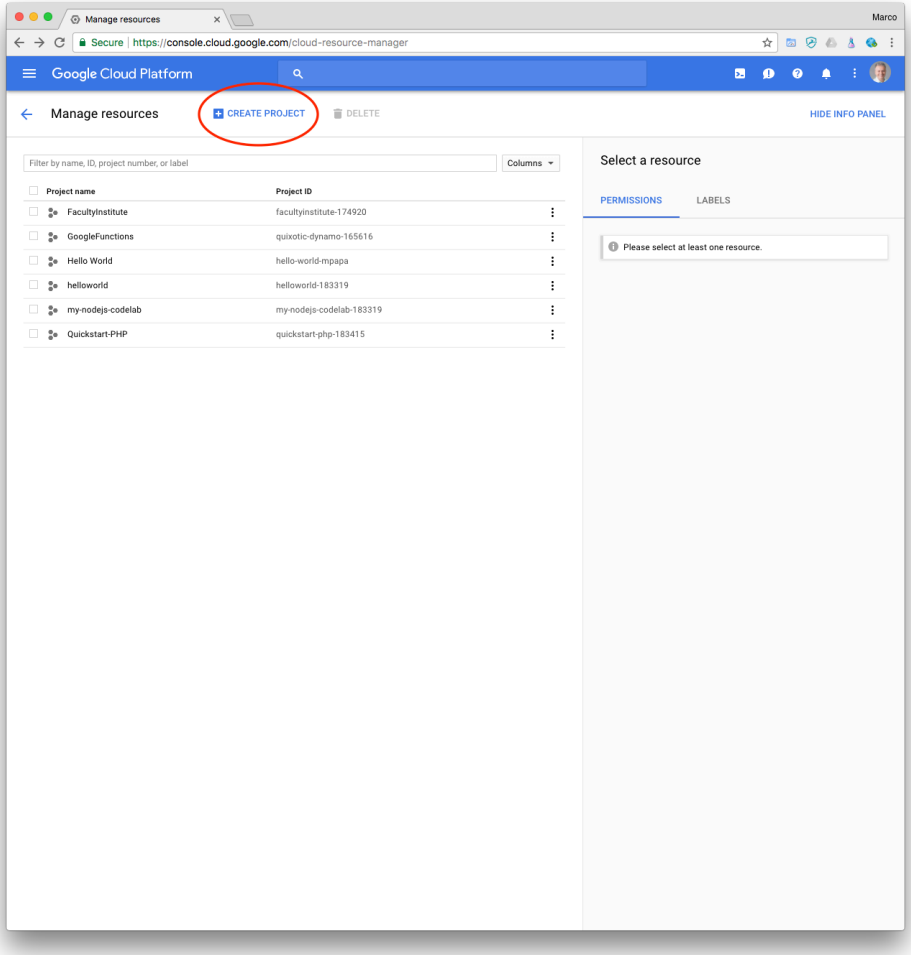
Google Cloud Functions (cont'd)



Follow the steps in this section to create a new Google Cloud function and an API endpoint to trigger it:

1. **Sign in** to the Google Cloud Platform at: https://console.cloud.google.com

Google Cloud Functions (cont'd)

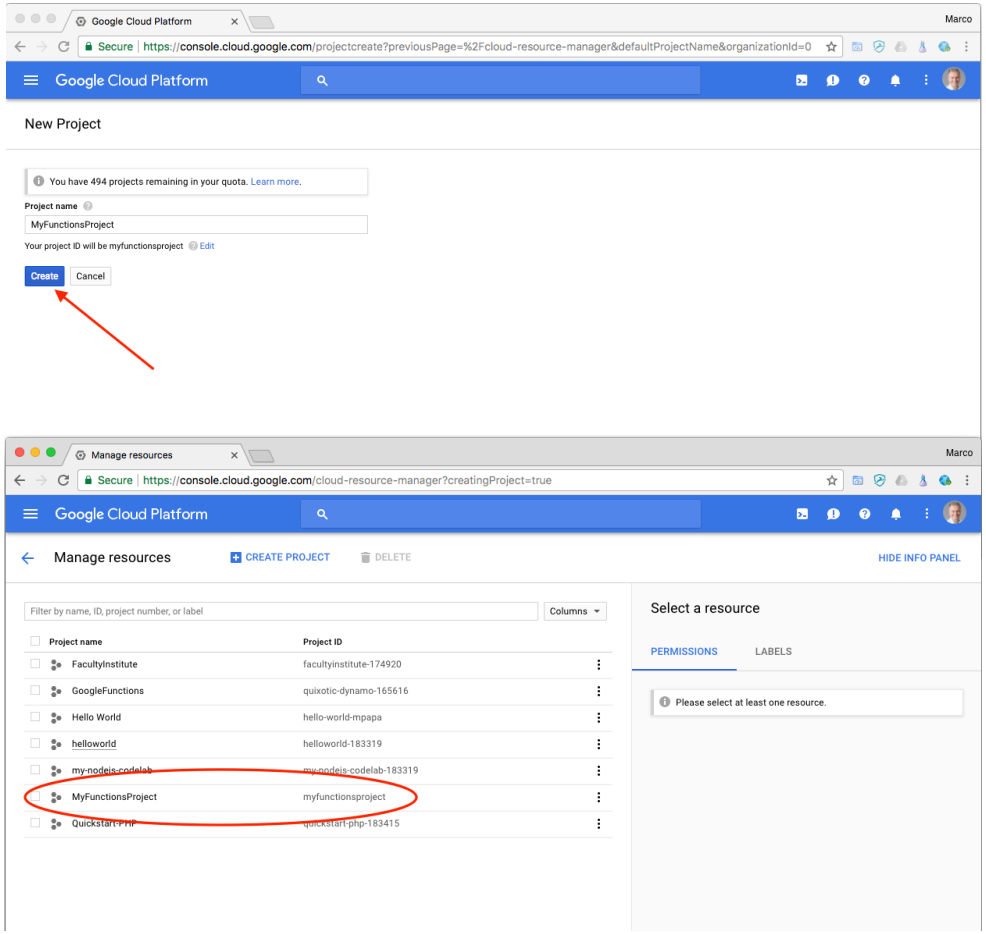


2. Select or create a Cloud Platform project. Go to the Projects page at:

https://console.cloud.google.com/project

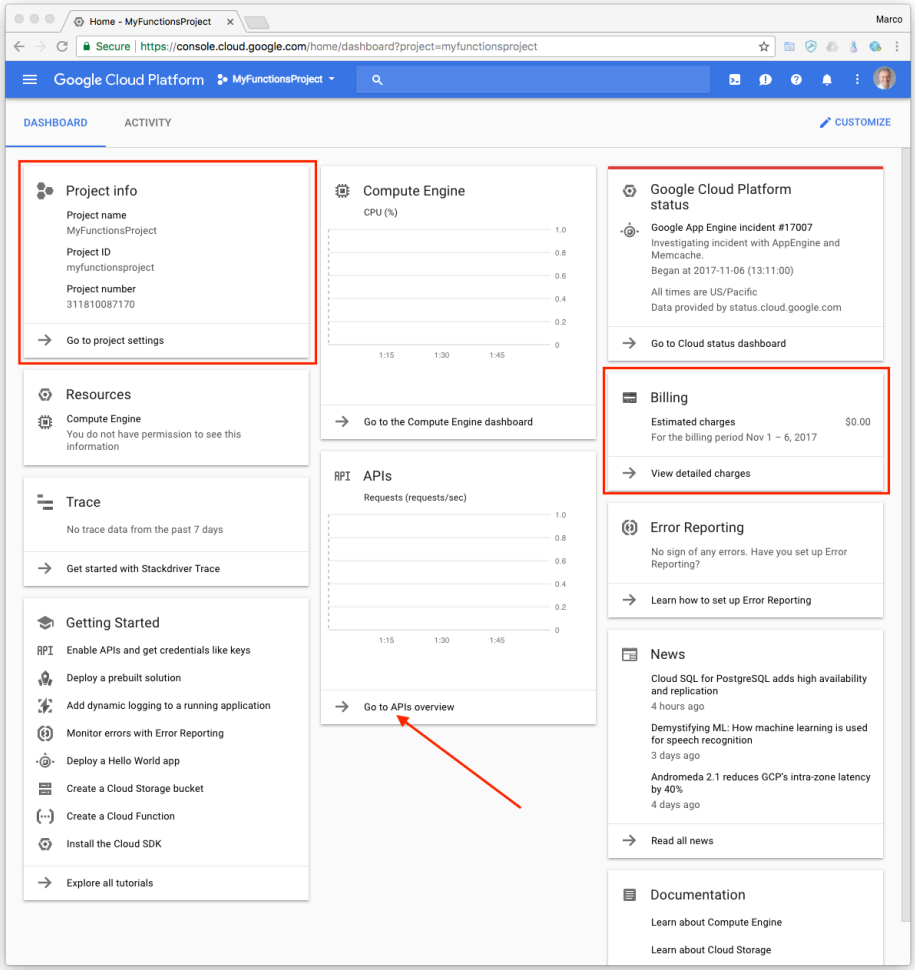
3. Select **CREATE PROJECT**.

Google Cloud Functions (cont'd)



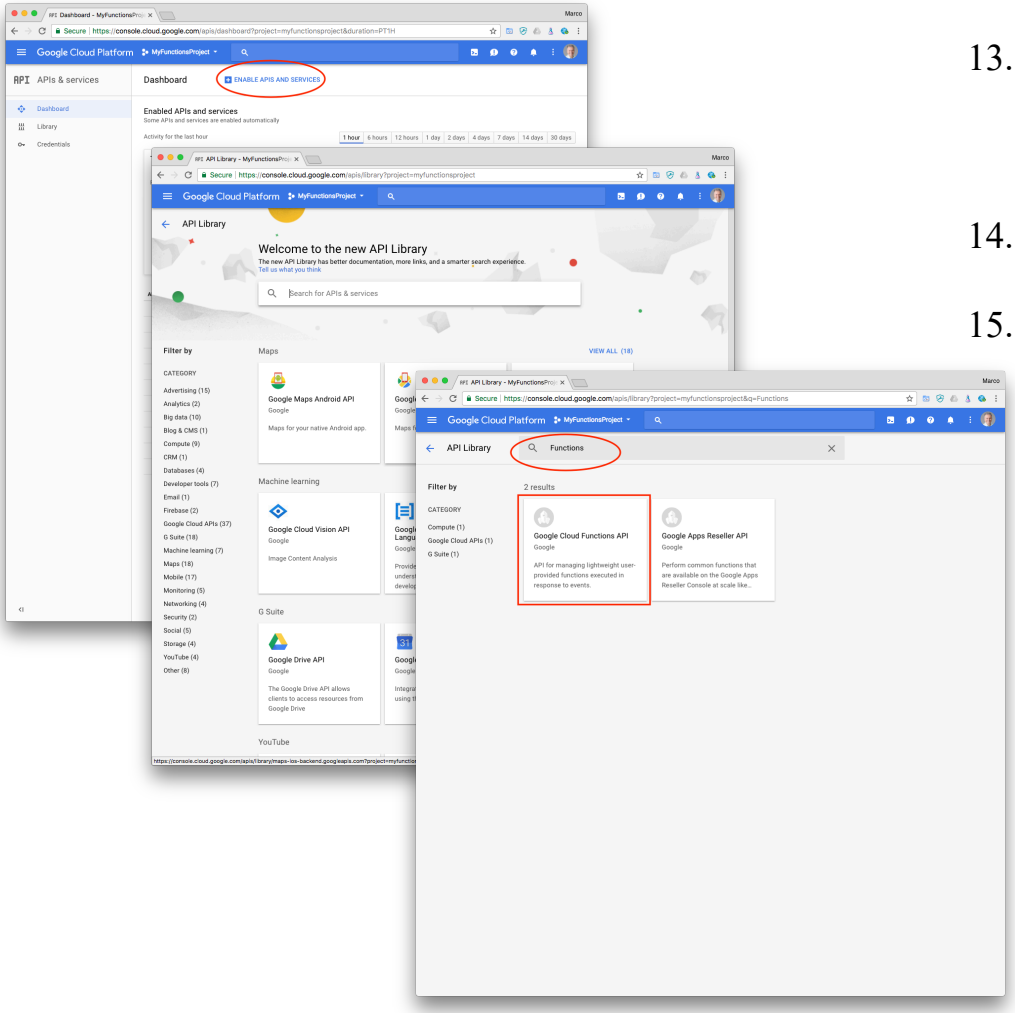
4. Enter your project name, such as *MyFunctionsProject*. Notice the project ID.
5. Click **Create**.
6. You may have to refresh the page to see your new project.
7. Click on the project name, *MyFunctionsProject*, in this example.
8. Click the **Products and Services** "3 bars" icon on top left and select **Home**.

Google Cloud Functions (cont'd)

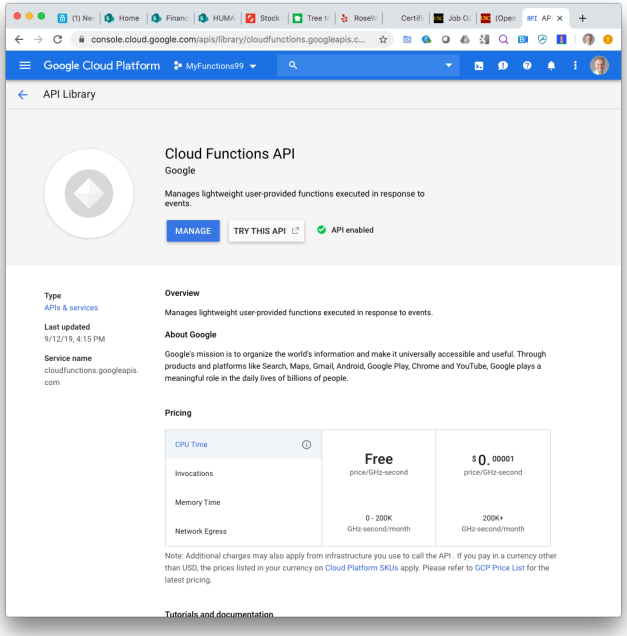


9. **Enable billing** for a project
10. How you enable billing depends on whether you're creating a new project or you're re-enabling billing for an existing project.
11. When you create a new project, you're prompted to choose which of your billing accounts you want to link to the project. If you have only one billing account, that account is automatically linked to your project.
12. Click on **Go to APIs overview**.

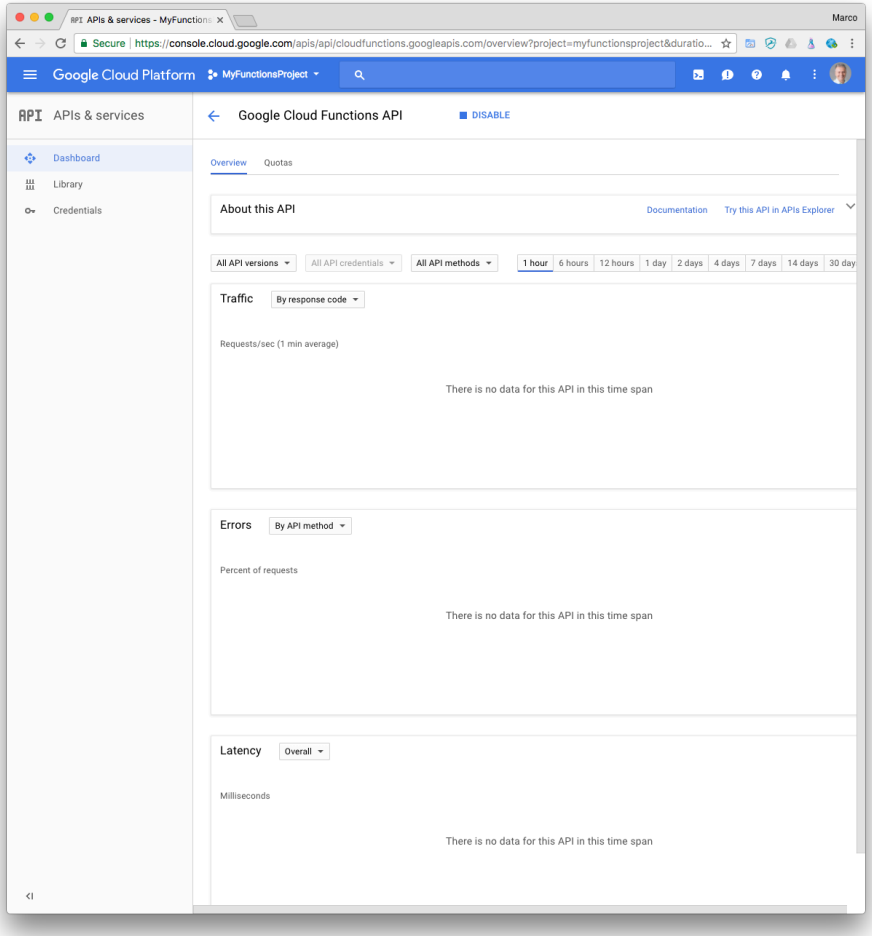
Google Cloud Functions (cont'd)



13. Enable the Cloud Functions API. In the **API & Services Dashboard**, click **ENABLE APIS AND SERVICES**. The page titled "*Welcome to the new API Library*" appears.
14. Enter "Functions" in the edit box. Then click **Cloud Functions API**.
15. Click **ENABLE**. Wait while "Enabling API."

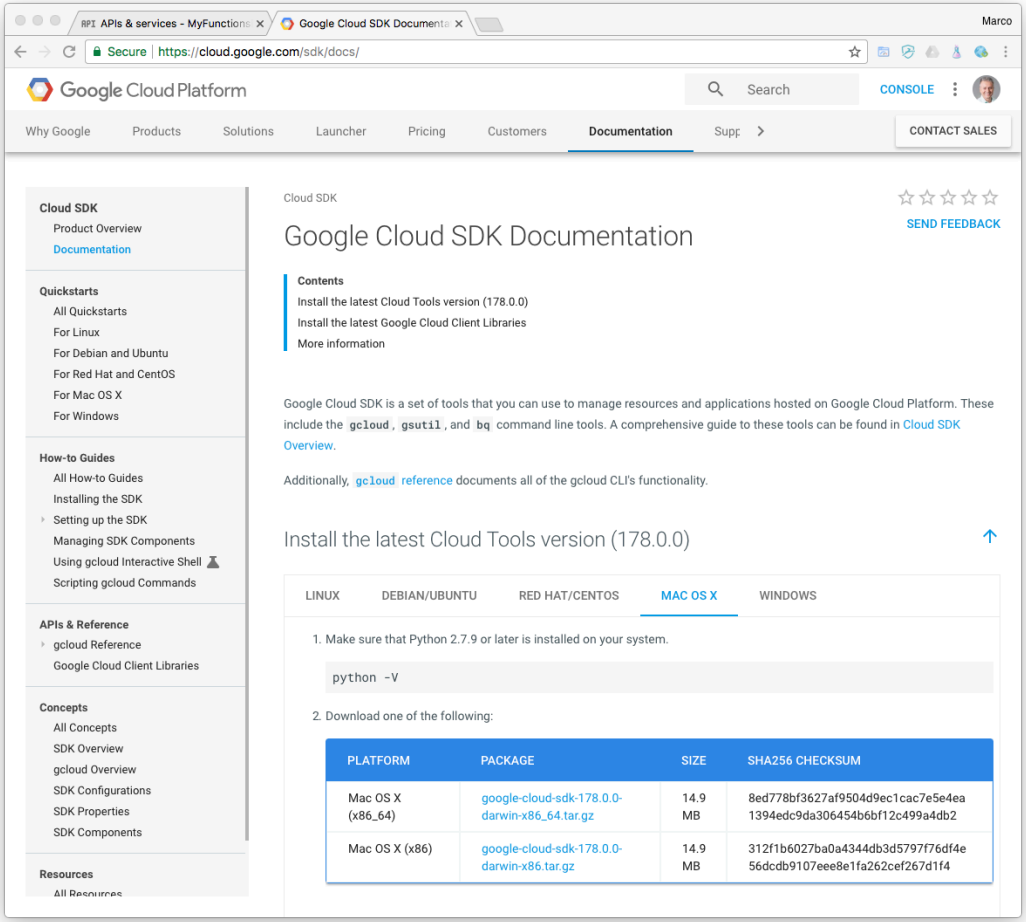


Google Cloud Functions (cont'd)



16. The **Google Cloud Functions API** "Overview" page is shown from the **Dashboard**. [if not showm, click on the **Navigation Menu** > **COMPUTE** > **Cloud Functions**. If you have previously created Functions, they will show on the list.

Google Cloud Functions (cont'd)



17. Install and initialize the Cloud SDK at:

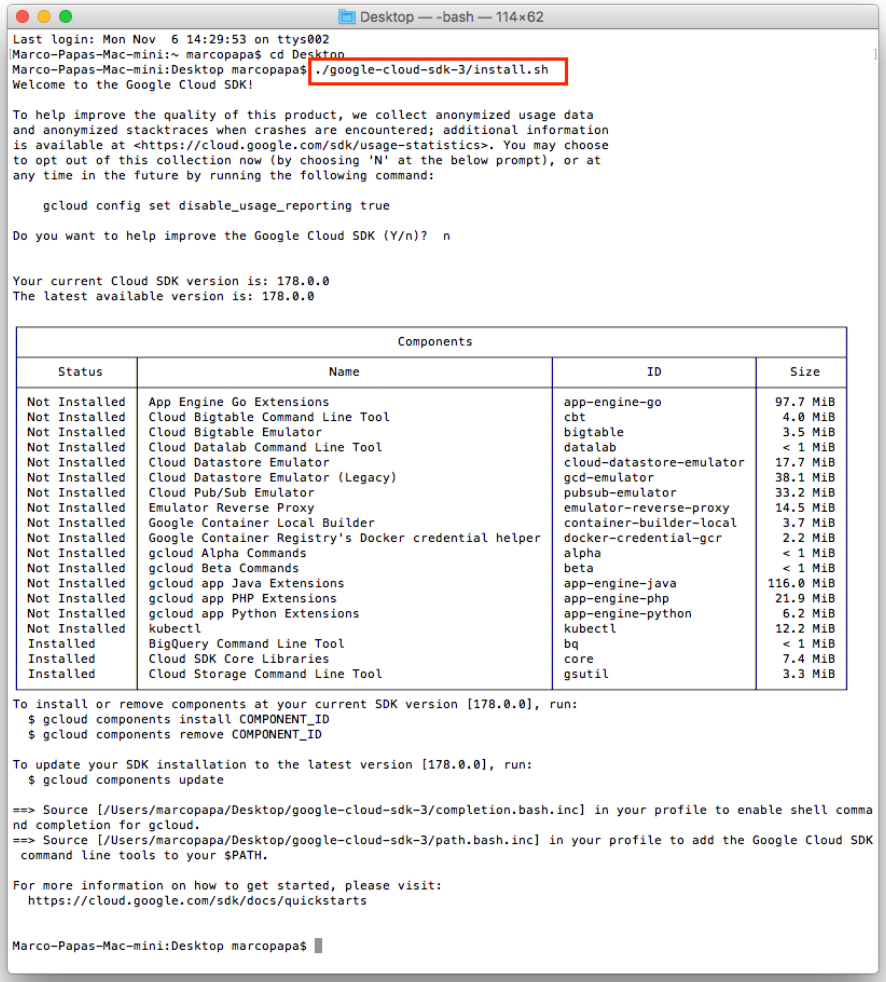
https://cloud.google.com/sdk/docs/

18. In the section titled *Install the latest Cloud Tools version 270.0.0)*, select your platform several versions of Linux, Mac OSX or Windows)
19. Make sure that Python 3.7 or later is installed on your system:

```
python -V
```

20. Download your selected package. (google-cloud-sdk-270.0.0-darwin-x86_64.tar on macOS)
21. Extract the file to any location on your file system.

Google Cloud Functions (cont'd)



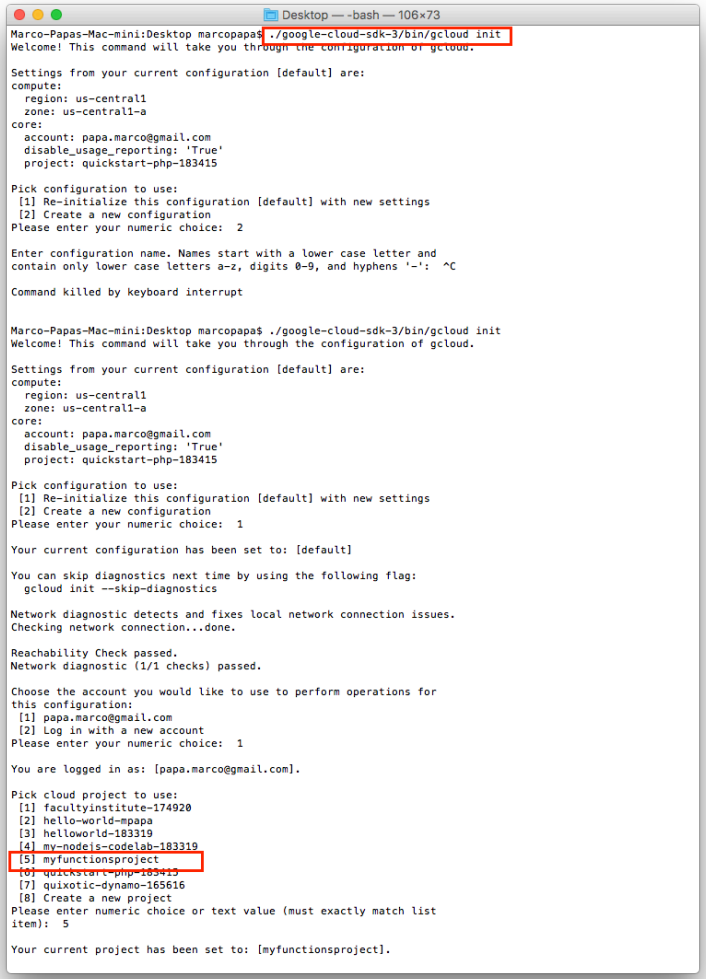
22. Run the **install script** to add SDK tools to your path, enable command completion in your bash shell, and/or and enable usage reporting.

```
./google-cloud-sdk-3/install.sh
```

Note: you may have to rename the SDK folder google-cloud-sdk-3 from "google-cloud-sdk 3".

18. Open a new terminal so that the changes take effect.

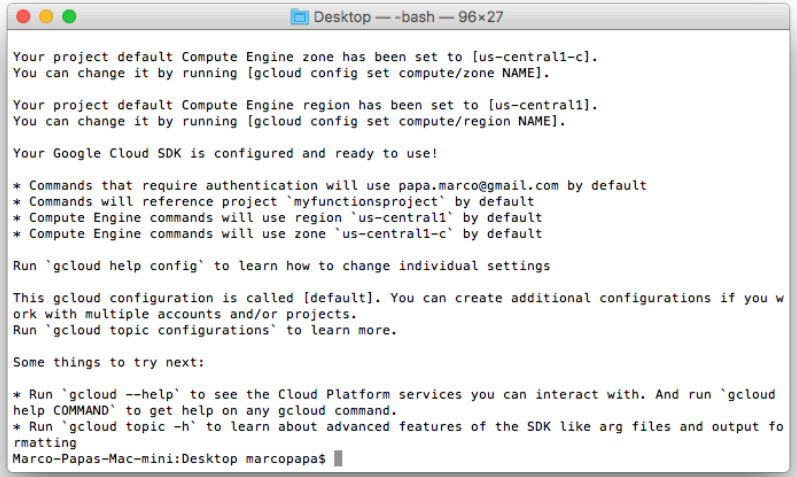
Google Cloud Functions (cont'd)



23. Run **gcloud init** to initialize the SDK:

```
./google-cloud-sdk/bin/gcloud init
```

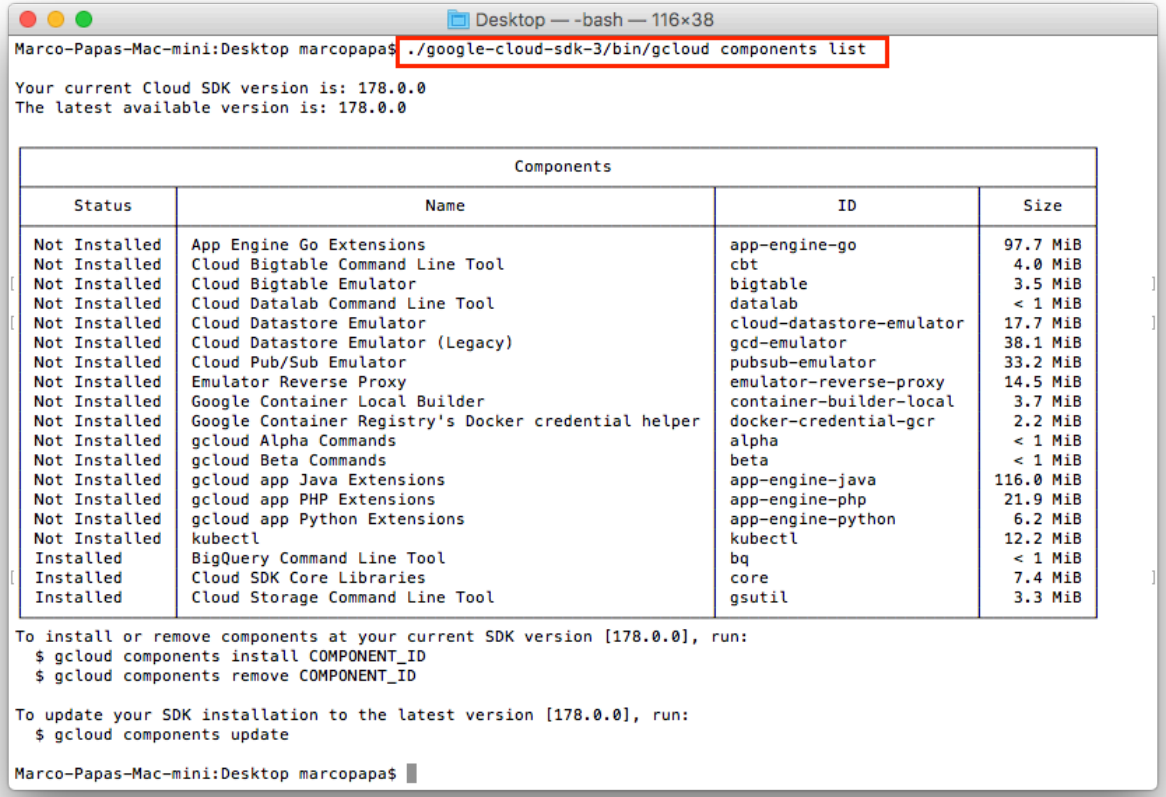
24. You will be asked to select the project.
25. You maybe asked to "enable" **API [compute.googleapis.com]** and "configure" **Google Compute Engine.** Answer **Y** to both.



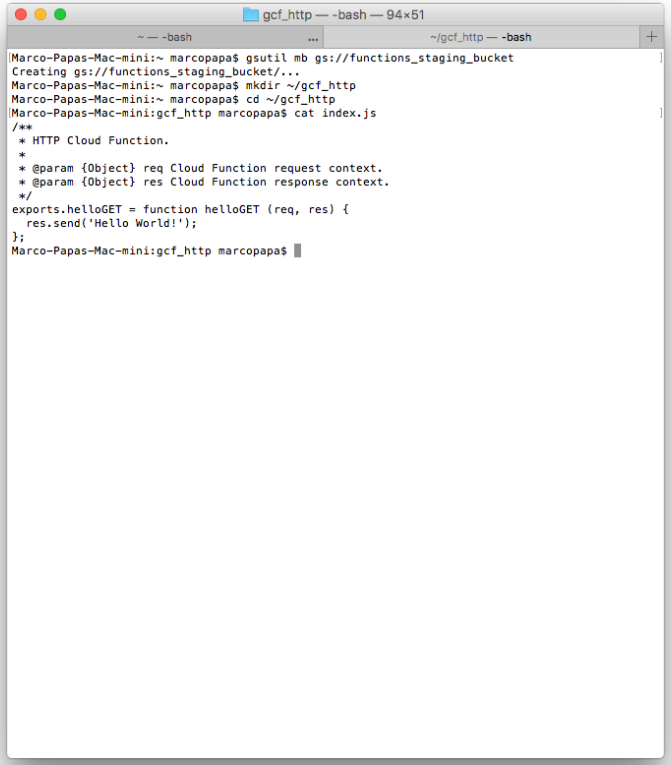
Google Cloud Functions (cont'd)

30. Verify all gcloud installed components:

```
./google-cloud-sdk-3/bin/gcloud components list
```



Google Cloud Functions (cont'd)



31. Now prepare the application. Create a Cloud Storage bucket to stage your Cloud Functions files, where [YOUR_STAGING_BUCKET_NAME] is a globally-unique bucket name:

```
gsutil mb gs://[YOUR_STAGING_BUCKET_NAME]
```

As in:

```
gsutil mb gs://functions_staging_bucket
```

32. Create a directory on your local system for the application code:

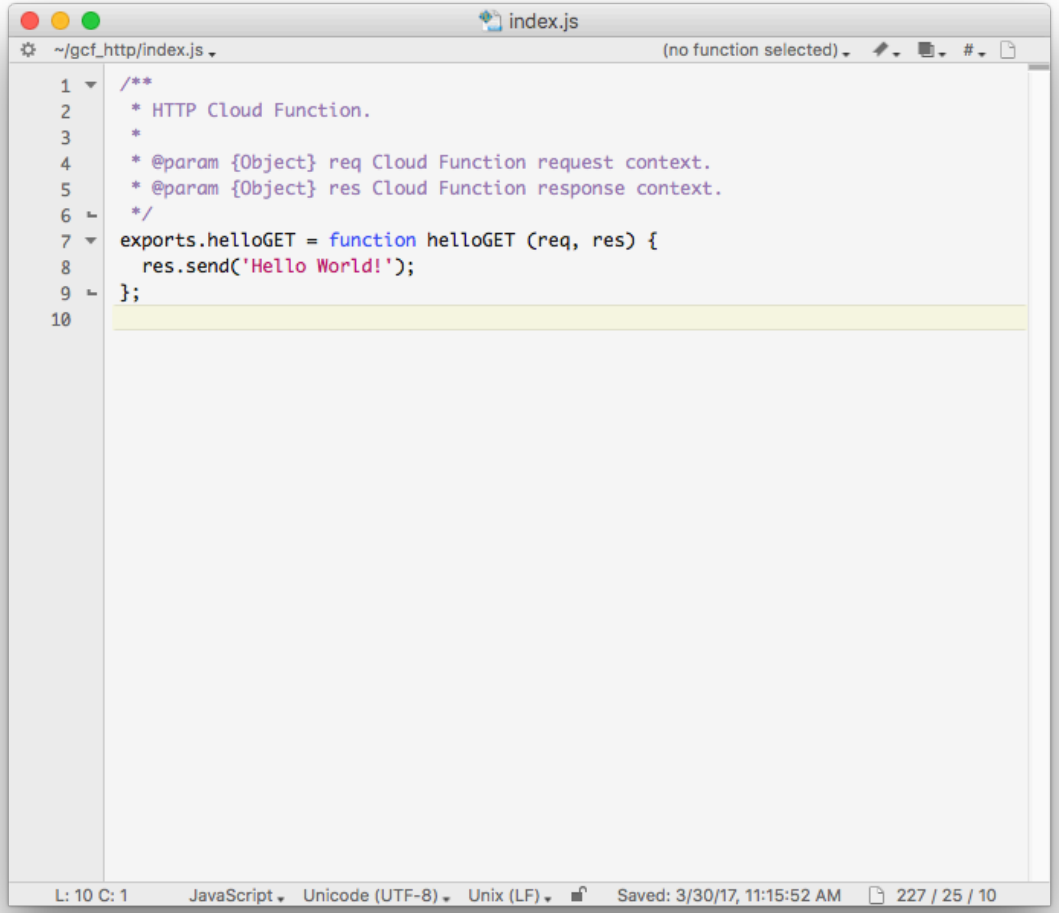
Linux or Mac OS X:

```
mkdir ~/gcf_http
cd ~/gcf_http
```

Windows:

```
mkdir %HOMEPATH%\gcf_http
cd %HOMEPATH%\gcf_http
```

Google Cloud Functions (cont'd)



33. Create an index.js file in the gcf_http directory with the following contents:

```
/**
 * HTTP Cloud Function.
 *
 * @param {Object} req Cloud
Function request context.
 * @param {Object} res Cloud
Function response context.
 */
exports.helloGET = function
helloGET (req, res) {
  res.send('Hello World!');
};
```

34. The **helloGET** function is exported by the module and is executed when you make an HTTP request to the function's endpoint.

Google Cloud Functions (cont'd)

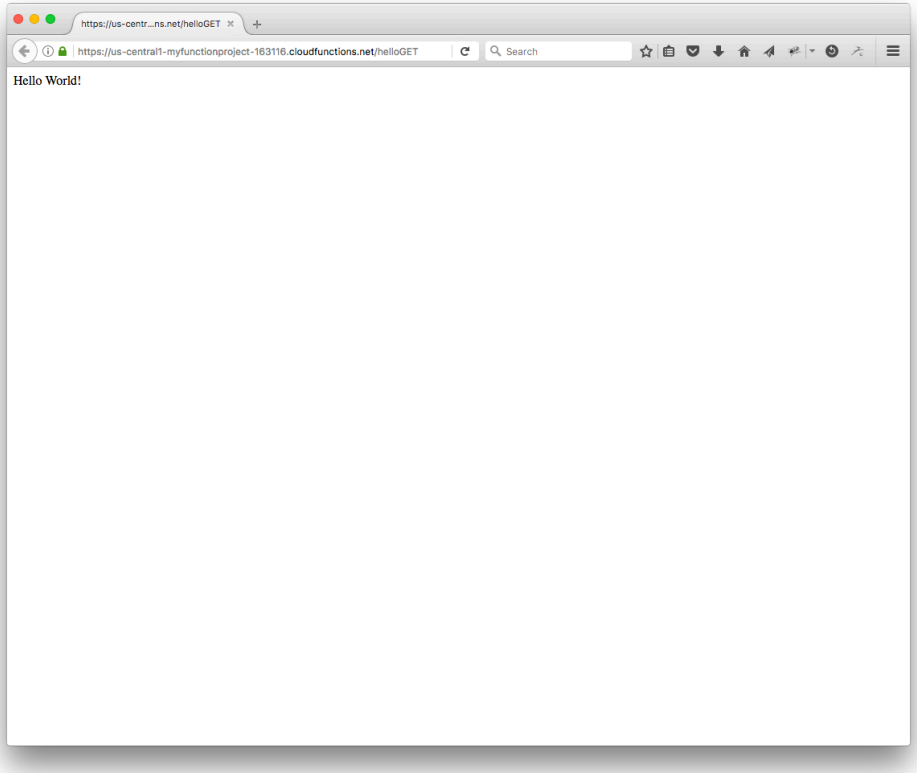
35. **Deploying the Function**. To **deploy** the helloGET function with an HTTP trigger, run the following command in the gcf_http directory:

```
gcloud beta functions deploy helloGET --stage-bucket [YOUR_STAGING_BUCKET_NAME]
--trigger-http
```

where [YOUR_STAGING_BUCKET_NAME] is the name of your staging Cloud Storage Bucket, as in:

```
gcloud beta functions deploy helloGET --stage-bucket functions_staging_bucket --
trigger-http
(you may be asked to install the 'gcloud Beta commands' component. Answer Y.)
```

Google Cloud Functions (cont'd)

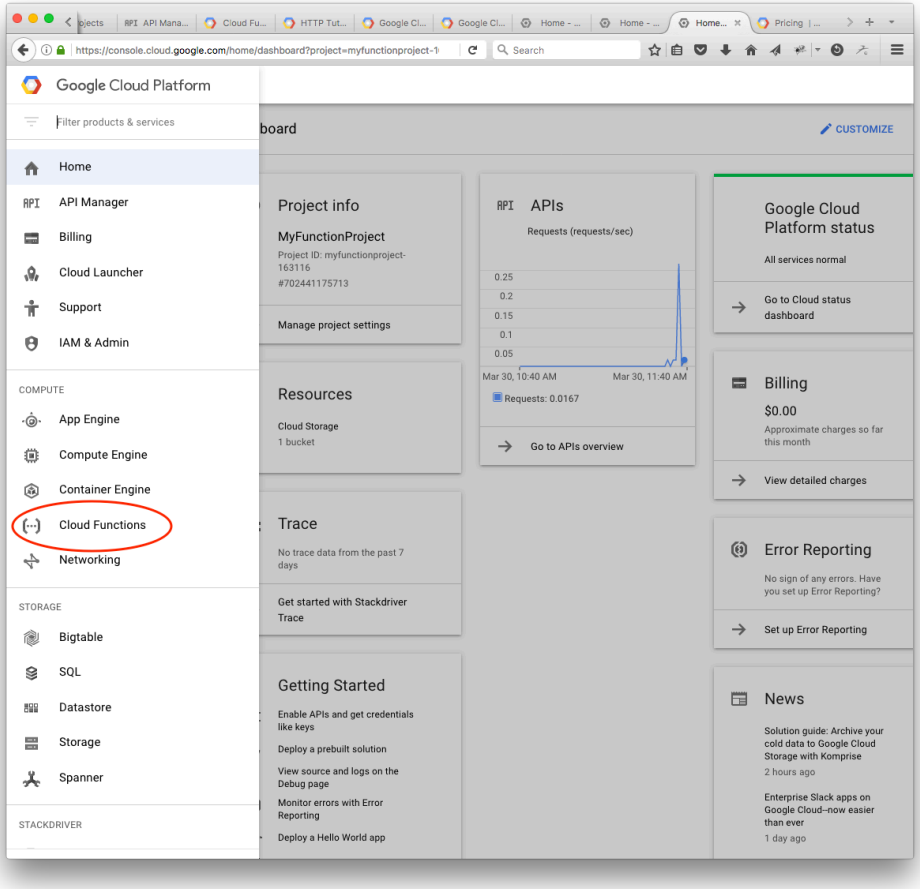


36. **Triggering the function**. Notice of the value of the url of httpsTrigger.

```
https://us-central1-
myfunctionproject-
163116.cloudfunctions.net/helloGET
```

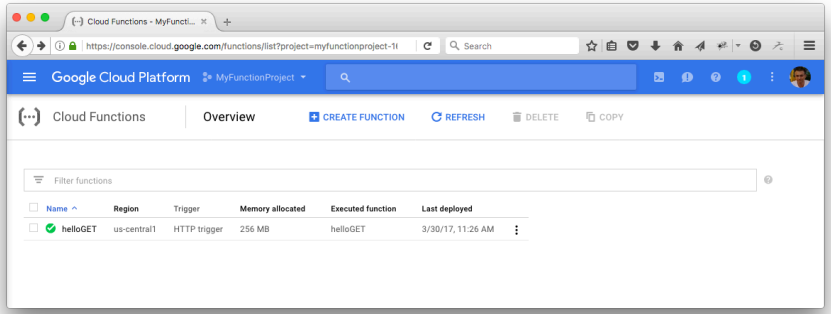
37. Make an **HTTP request** to your function, using curl or visit the function's endpoint in your browser to see the "Hello World!" message.

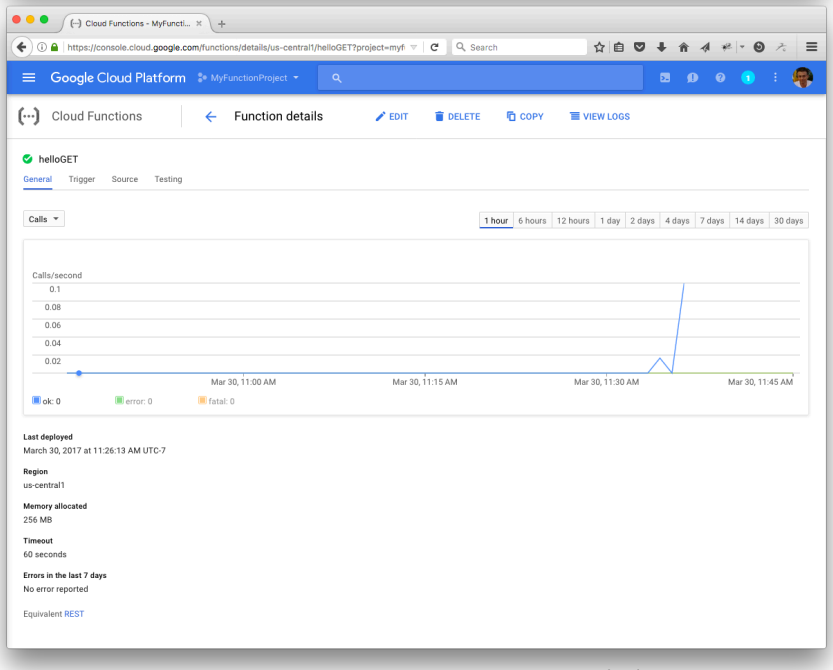
Google Cloud Functions (cont'd)



38. **Monitor the function**. From the Google Cloud Platform menu, select **Cloud Functions**.

Google Cloud Functions (cont'd)





39. Click on **helloGet**.
40. Click on the **Trigger**, **Source** and **Testing** tabs.

Google Cloud Functions (cont'd)

41. Quickstart: "Using the Console" available at: https://cloud.google.com/functions/docs/quickstart-console