# Your First Function: Python

cloud.google.com/functions/docs/first-python

This guide takes you through the process of writing a Cloud Function using the Python runtime. There are two types of Cloud Functions:

- An HTTP function, which you invoke from standard HTTP requests.
- An event-driven function, which you use to handle events from your Cloud infrastructure, such as messages on a Cloud Pub/Sub topic, or changes in a Cloud Storage bucket.

The sample shows how to create a simple HTTP function.

**Learn more**: For more details, read about HTTP functions and event-driven functions.

## Guide structure

## Creating a GCP project using Cloud SDK

1. In the Google Cloud Console, on the project selector page, select or create a Google Cloud project.

   **Note**: If you don't plan to keep the resources that you create in this procedure, create a project instead of selecting an existing project. After you finish these steps, you can delete the project, removing all resources associated with the project.
   Go to project selector

2. Make sure that billing is enabled for your Cloud project. Learn how to confirm that billing is enabled for your project.

3. Enable the Cloud Functions and Cloud Build APIs.
   Enable the APIs

4. Install and initialize the Cloud SDK.
5. Update and install `gcloud` components:

   ```
   gcloud components update
   ```

   **Note**: Need a command prompt? You can use the Google Cloud Shell. The Google Cloud Shell is a command line environment that already includes the Google Cloud SDK, so you don't need to install it. The Google Cloud SDK also comes preinstalled on Google Compute Engine Virtual Machines.
6. Prepare your development environment.
   Go to the Python setup guide

## Creating a function

1. Create a directory on your local system for the function code:

```
mkdir ~/helloworld
cd ~/helloworld
```

2. Create a `main.py` file in the `helloworld` directory with the following contents:

```
from flask import

 escape


def hello_http(request):
    """HTTP Cloud Function.
    Args:
        request (flask.Request): The request object.
        <https://flask.palletsprojects.com/en/1.1.x/api/#incoming-request-
data>
    Returns:
        The response text, or any set of values that can be turned into a
        Response object using `make_response`

<https://flask.palletsprojects.com/en/1.1.x/api/#flask.make_response>.
    """
    request_json = request.get_json(silent=True)
    request_args = request.

args


    if request_json and 'name' in request_json:
        name = request_json['name']
    elif request_args and 'name' in request_args:
        name = request_args['name']
    else:
        name = 'World'
    return 'Hello {}!'.format(escape(name))
```

This example function takes a name supplied in the HTTP request and returns a greeting, or "Hello World!" when no name is supplied.

**Note:** Cloud Functions looks for deployable functions in `main.py` by default. Use the `--source` flag when deploying your function via `gcloud` to specify a different directory containing a `main.py` file.

## Specifying dependencies

Dependencies in Python are managed with pip and expressed in a metadata file called `requirements.txt`. This file must be in the same directory as the `main.py` file that contains your function code.

1. Create a `requirements.txt` file in the `helloworld` directory.

2. Add the function's dependency, in this case the Flask package, to your `requirements.txt` file by adding the following line:

   ```
   Flask==1.0.2
   ```

**Learn more**: For more details, read about specifying dependencies.

## Deploying the function

To deploy the function with an HTTP trigger, run the following command in the `helloworld` directory:

```
gcloud functions deploy hello_http --runtime python39 --trigger-http --allow-unauthenticated
```

The `--allow-unauthenticated` flag lets you reach the function without authentication. To require authentication, omit the flag.

**Learn more**: For more details, read about deploying Cloud Functions.

## Testing the function

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

   ```
   gcloud functions describe hello_http
   ```

   It should look like this:

   ```
   https://GCP_REGION-PROJECT_ID.cloudfunctions.net/hello_http
   ```

2. Visit this URL in your browser. You should see a "Hello World!" message.

   Try passing a name in the HTTP request, for example by using the following URL:

   ```
   https://GCP_REGION-PROJECT_ID.cloudfunctions.net/hello_http?name=NAME
   ```

   You should see the message "Hello *NAME*!"

## Viewing logs

### Using the command-line tool

Logs for Cloud Functions are viewable in the Cloud Logging UI, and via the `gcloud` command-line tool.

To view logs for your function with the `gcloud` tool, use the `logs read` command, followed by the name of the function:

```
gcloud functions logs read hello_http
```

The output should resemble the following:

```
LEVEL  NAME        EXECUTION_ID  TIME_UTC                 LOG
D      hello_http  pdb5ys2t022n  2019-09-18 23:29:09.791  Function execution
started
D      hello_http  pdb5ys2t022n  2019-09-18 23:29:09.798  Function execution took
7 ms, finished with status code: 200
```

**Note:** There is typically a slight delay between when log entries are created and when they show up in Cloud Logging.

## Using the Logging dashboard

You can also view logs for Cloud Functions from the Cloud Console.

**Learn more**: For more details, read about writing, viewing, and responding to logs.
Rate and review