

[Serverless Computing](https://cloud.google.com/products/serverless/) (https://cloud.google.com/products/serverless/)

[App Engine](https://cloud.google.com/appengine/) (https://cloud.google.com/appengine/)

[Documentation](https://cloud.google.com/appengine/docs/) (https://cloud.google.com/appengine/docs/)

[Flexible Environment](https://cloud.google.com/appengine/docs/flexible/) (https://cloud.google.com/appengine/docs/flexible/)

[Python](https://cloud.google.com/appengine/docs/flexible/python/) (https://cloud.google.com/appengine/docs/flexible/python/) [Guides](#)

Quickstart for Python in the App Engine Flexible Environment

Python | [Java](https://cloud.google.com/appengine/docs/flexible/java/quickstart) (https://cloud.google.com/appengine/docs/flexible/java/quickstart) | [Node.js](https://cloud.google.com/appengine/docs/flexible/nodejs/quickstart)

(https://cloud.google.com/appengine/docs/flexible/nodejs/quickstart) | [Go](https://cloud.google.com/appengine/docs/flexible/go/quickstart)

(https://cloud.google.com/appengine/docs/flexible/go/quickstart) | [Ruby](https://cloud.google.com/appengine/docs/flexible/ruby/quickstart)

(https://cloud.google.com/appengine/docs/flexible/ruby/quickstart) | [PHP](https://cloud.google.com/appengine/docs/flexible/php/quickstart)

(https://cloud.google.com/appengine/docs/flexible/php/quickstart) | [.NET](https://cloud.google.com/appengine/docs/flexible/dotnet/quickstart)

(https://cloud.google.com/appengine/docs/flexible/dotnet/quickstart)

This quickstart shows you how to create a small App Engine app that displays a short message.

Before you begin

Before running and deploying this quickstart, install the Cloud SDK and then set up a Google Cloud project for App Engine:

1. Download and install Cloud SDK:

[DOWNLOAD THE SDK](https://cloud.google.com/sdk/docs/) (HTTPS://CLOUD.GOOGLE.COM/SDK/DOCS/)

Note: If you already have the Cloud SDK installed, update it by running the following command:

```
gcloud components update
```



2. Create a new project:

```
gcloud projects create [YOUR_PROJECT_ID] --set-as-default
```



Verify the project was created:

```
gcloud projects describe [YOUR_PROJECT_ID]
```



You see project details that might look like the following:

```
createTime: year-month-hour
lifecycleState: ACTIVE
name: project-name
parent:
id: '433637338589'
type: organization
projectId: project-name-id
projectNumber: 499227785679
```



3. Initialize your App Engine app with your project and choose its region:

```
gcloud app create --project=[YOUR_PROJECT_ID]
```



When prompted, select the region (#before-you-begin) where you want your App Engine application located.

! **Caution:** You cannot change an app's region once it has been set.

4. Make sure billing is enabled for your project. A billing account needs to be linked to your project in order for the application to be deployed to App Engine.

ENABLE BILLING ([HTTPS://CONSOLE.CLOUD.GOOGLE.COM/PROJECTSELECTOR/BILLING?LANG=P](https://console.cloud.google.com/projectselector/billing?lang=P))

Your flexible environment deployment will incur costs while it is active. Clean up your project (#clean-up) when you are finished to avoid ongoing costs.

5. Install the following prerequisites:

- [Download and install Git](https://git-scm.com/) (<https://git-scm.com/>).
- Run the following command to install the [gcloud component](https://cloud.google.com/sdk/docs/managing-components) (<https://cloud.google.com/sdk/docs/managing-components>) that includes the App Engine extension for Python:

```
gcloud components install app-engine-python
```



6. Prepare your environment for Python development. It is recommended that you have the latest version of Python, `pip`, and other related tools installed on your system. For instructions, refer to the [Python Development Environment Setup Guide](https://cloud.google.com/python/setup) (<https://cloud.google.com/python/setup>).

This quickstart demonstrates a simple Python app written with the [Flask](http://flask.pocoo.org/) (<http://flask.pocoo.org/>) web framework that can be deployed to App Engine. Although this sample uses Flask, you can use any web framework that satisfies the requirements above. Alternative frameworks include [Django](https://www.djangoproject.com) (<https://www.djangoproject.com>), [Pyramid](http://www.pylonsproject.org/) (<http://www.pylonsproject.org/>), [Bottle](http://bottlepy.org/) (<http://bottlepy.org/>), and [web.py](http://webpy.org/) (<http://webpy.org/>).

Download the Hello World app

We've created a simple Hello World app for Python so you can quickly get a feel for deploying an app to the Google Cloud.

1. Clone the Hello World sample app repository to your local machine.

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



Alternatively, you can [download the sample](https://github.com/GoogleCloudPlatform/python-docs-samples/archive/master.zip)

(<https://github.com/GoogleCloudPlatform/python-docs-samples/archive/master.zip>) as a zip file and extract it.

2. Change to the directory that contains the sample code.

```
cd python-docs-samples/appengine/flexible/hello_world
```



Run Hello World on your local machine

To run the Hello World app on your local computer:

MAC OS / LINUX

WINDOWS

Note: These instructions describe how to set up a virtual environment in Python 3. For Python 2 apps, use [virtualenv](#)

(<https://packaging.python.org/guides/installing-using-pip-and-virtual-environments/#installing-virtualenv>)

to set up a virtual environment.

1. Create an isolated Python environment in a directory external to your project and activate it:

```
python3 -m venv env
source env/bin/activate
```

2. Navigate to your project directory and install dependencies:

```
cd YOUR_PROJECT
pip install -r requirements.txt
```

3. Run the application:

```
python main.py
```

4. In your web browser, enter the following address:

```
http://localhost:8080
```

★ **Note:** If you are using Cloud Shell, in the toolbar, click **Web Preview** and select **Preview on port 8080** instead.

The **Hello World** message from the sample app displays on the page. In your terminal window, press **Ctrl+C** to exit the web server.

Deploy and run Hello World on App Engine

To deploy your app to the App Engine flexible environment:

1. Deploy the Hello World app by running the following command from the `hello_world` directory:

```
gcloud app deploy
```

Learn about the [optional flags \(#deploy_and_run_hello_world_on_app_engine\)](#).

2. Launch your browser to view the app at `http://YOUR_PROJECT_ID.appspot.com`

gcloud app browse



where `YOUR_PROJECT_ID` represents your Google Cloud project ID.

This time, the page that displays the Hello World message is delivered by a web server running on an App Engine instance.

Congratulations! You've deployed your first Python app to App Engine flexible environment!

See the following sections for information about cleaning up as well as links to possible next steps that you can take.

Clean up


To avoid incurring charges, you can delete your Google Cloud project to stop billing for all the resources used within that project.

Caution: Deleting a project has the following effects:

- **Everything in the project is deleted.** If you used an existing project for this tutorial, when you delete it, you also delete any other work you've done in the project.
- **Custom project IDs are lost.** When you created this project, you might have created a custom project ID that you want to use in the future. To preserve the URLs that use the project ID, such as an `appspot.com` URL, delete selected resources inside the project instead of deleting the whole project.

1. In the Cloud Console, go to the **Manage resources** page.

[GO TO THE MANAGE RESOURCES PAGE \(HTTPS://CONSOLE.CLOUD.GOOGLE.COM/IAM-ADMIN/PROJ...](https://console.cloud.google.com/iam-admin/projects)

2. In the project list, select the project you want to delete and click **Delete** .

3. In the dialog, type the project ID, and then click **Shut down** to delete the project.

What's next

Learn the whole platform

Now that you know what it's like to develop and deploy App Engine apps, you can explore the rest of Google Cloud. You already have the Cloud SDK installed which gives you the tools to interact with products like Cloud SQL, Cloud Storage, Firestore, and more.

For a guided walkthrough that teaches you how to create an app that uses the entire platform, not just App Engine, check out our quickstart on creating [the Bookshelf app](https://cloud.google.com/python/getting-started/tutorial-app) (<https://cloud.google.com/python/getting-started/tutorial-app>).

Learn about the App Engine flexible environment

Here are some topics to help continue your learning about App Engine.

- [An overview of App Engine](https://cloud.google.com/appengine/docs/flexible/python/an-overview-of-app-engine)
(<https://cloud.google.com/appengine/docs/flexible/python/an-overview-of-app-engine>)
- [How requests are routed](https://cloud.google.com/appengine/docs/flexible/python/how-requests-are-routed)
(<https://cloud.google.com/appengine/docs/flexible/python/how-requests-are-routed>)
- [How requests are handled](https://cloud.google.com/appengine/docs/flexible/python/how-requests-are-handled)
(<https://cloud.google.com/appengine/docs/flexible/python/how-requests-are-handled>)
- [How instances are managed](https://cloud.google.com/appengine/docs/flexible/python/how-instances-are-managed)
(<https://cloud.google.com/appengine/docs/flexible/python/how-instances-are-managed>)

Hello World code review

Hello World is the simplest possible App Engine app, as it contains only one service, has only one version, and all of the code is located within the app's root directory. This section describes each of the app files in detail.

`main.py`

The Hello World app is a basic one-file Flask app.



[appengine/flexible/hello_world/main.py](https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/appengine/flexible/hello_world/main.py)

(https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/appengine/flexible/hello_world/main.py)

PLATFORM/PYTHON-DOCS-SAMPLES/BLOB/MASTER/APPENGINE/FLEXIBLE/HELLO_WORLD/MAIN.PY)

```
import logging

from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello():
    """Return a friendly HTTP greeting."""
    return 'Hello World!'

@app.errorhandler(500)
def server_error(e):
    logging.exception('An error occurred during a request.')
    return """
    An internal error occurred: <pre>{}</pre>
    See logs for full stacktrace.
    """.format(e), 500

if __name__ == '__main__':
    # This is used when running locally. Gunicorn is used to run the
    # application on Google App Engine. See entrypoint in app.yaml.
    app.run(host='127.0.0.1', port=8080, debug=True)
```

app.yaml

The `app.yaml`

(<https://cloud.google.com/appengine/docs/flexible/python/configuring-your-app-with-app-yaml>) file describes an app's deployment configuration:

[appengine/flexible/hello_world/app.yaml](https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/appengine/flexible/hello_world/app.yaml)

(https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/appengine/flexible/hello_world/app.yaml)

```
PLATFORM/PYTHON-DOCS-SAMPLES/BLOB/MASTER/APPENGINE/FLEXIBLE/HELLO_WORLD/APP.YAML)
```

```
runtime: python
env: flex
entrypoint: gunicorn -b :$PORT main:app

runtime_config:
  python_version: 3

# This sample incurs costs to run on the App Engine flexible environment.
# The settings below are to reduce costs during testing and are not appropriate
# for production use. For more information, see:
# https://cloud.google.com/appengine/docs/flexible/python/configuring-your-app-with-
manual_scaling:
  instances: 1
resources:
  cpu: 1
  memory_gb: 0.5
  disk_size_gb: 10
```

Here, `app.yaml` specifies the runtime used by the app, and sets `env: flex`, specifying that the app uses the [flexible environment](https://cloud.google.com/appengine/docs/flexible/) (<https://cloud.google.com/appengine/docs/flexible/>).

The `entrypoint` tells App Engine how to start the app. This app uses [gunicorn](http://gunicorn.org/) (<http://gunicorn.org/>) to serve the Python app. The `$PORT` variable is set by App Engine when it starts the app. For more information about `entrypoint`, see [App startup](https://cloud.google.com/appengine/docs/flexible/python/runtime#application_startup) (https://cloud.google.com/appengine/docs/flexible/python/runtime#application_startup).

Note: This is the simplest way to get gunicorn running. However this approach runs the application with a single blocking worker, which means all HTTP requests are handled in serial, including health checks. In practice, this results in the application becoming unresponsive in the event of slow requests. For more details on how to configure gunicorn for production, see [Recommended gunicorn configuration](https://cloud.google.com/appengine/docs/flexible/python/runtime#recommended_gunicorn_configuration) (https://cloud.google.com/appengine/docs/flexible/python/runtime#recommended_gunicorn_configuration).

Additionally, the optional `runtime_config` section sets `python_version` to use Python 3. If `python_version` is not specified, then Python 2 is used by default. You can also specify `python_version: 2` explicitly.

- For more information on how the Python runtime works, see [The Python runtime](https://cloud.google.com/appengine/docs/flexible/python/runtime) (<https://cloud.google.com/appengine/docs/flexible/python/runtime>).

- For more details about how to design your app to take advantage of versions and services, see [An overview of App Engine](https://cloud.google.com/appengine/docs/flexible/python/an-overview-of-app-engine) (<https://cloud.google.com/appengine/docs/flexible/python/an-overview-of-app-engine>).
- For more details about the configuration settings for App Engine, see [Configuring your app with app.yaml](https://cloud.google.com/appengine/docs/flexible/python/configuring-your-app-with-app-yaml) (<https://cloud.google.com/appengine/docs/flexible/python/configuring-your-app-with-app-yaml>).

requirements.txt

`requirements.txt` (http://pip.readthedocs.org/en/stable/user_guide/#requirements-files) and the Python package manager `pip` (<http://pip.readthedocs.org>) are used to declare and install application dependencies. Hello World requires `Flask` (<http://flask.pocoo.org/>), a web framework, and `Gunicorn` (<http://gunicorn.org/>), a WSGI server.

```
appengine/flexible/hello_world/requirements.txt  
(https://github.com/GoogleCloudPlatform/python-docs-  
samples/blob/master/appengine/flexible/hello_world/requirements.txt)
```

```
/PYTHON-DOCS-SAMPLES/BLOB/MASTER/APPENGINE/FLEXIBLE/HELLO_WORLD/REQUIREMENTS.TXT)
```

```
Flask==1.1.1  
gunicorn==20.0.4
```

`requirements.txt` defines the libraries that will be installed both locally and when deploying to App Engine.

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](https://creativecommons.org/licenses/by/4.0/) (<https://creativecommons.org/licenses/by/4.0/>), and code samples are licensed under the [Apache 2.0 License](https://www.apache.org/licenses/LICENSE-2.0) (<https://www.apache.org/licenses/LICENSE-2.0>). For details, see our [Site Policies](https://developers.google.com/terms/site-policies) (<https://developers.google.com/terms/site-policies>). Java is a registered trademark of Oracle and/or its affiliates.

Last updated January 2, 2020.