

[Python](https://cloud.google.com/python/) (<https://cloud.google.com/python/>) [Guides](#)

Handling sessions with Firestore

Many apps need session handling for authentication and user preferences. The [Flask framework](http://flask.pocoo.org/) (<http://flask.pocoo.org/>) comes with a memory-based implementation to perform this function. However, this implementation is unsuitable for an app that can be served from multiple instances, because the session that is recorded in one instance might differ from other instances. This tutorial shows how to handle sessions on [App Engine](https://cloud.google.com/appengine/docs/standard/) (<https://cloud.google.com/appengine/docs/standard/>).

Objectives

- Write the app.
- Run the app locally.
- Deploy the app on App Engine.

Costs

This tutorial uses the following billable components of Google Cloud:

- [App Engine](https://cloud.google.com/appengine/pricing) (<https://cloud.google.com/appengine/pricing>)
- [Firestore](https://cloud.google.com/firestore/pricing) (<https://cloud.google.com/firestore/pricing>)

To generate a cost estimate based on your projected usage, use the [pricing calculator](https://cloud.google.com/products/calculator) (<https://cloud.google.com/products/calculator>). New Google Cloud users might be eligible for a [free trial](https://cloud.google.com/free-trial) (<https://cloud.google.com/free-trial>).

When you finish this tutorial, you can avoid continued billing by deleting the resources you created. For more information, see [Cleaning up](#) (#clean-up).

Before you begin

1. [Sign in](https://accounts.google.com/Login) (https://accounts.google.com/Login) to your Google Account.

If you don't already have one, [sign up for a new account](https://accounts.google.com/SignUp) (https://accounts.google.com/SignUp).

2. In the 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 THE PROJECT SELECTOR PAGE](https://console.cloud.google.com/projectselector) (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/PROJECTSELECTOR)

3. Make sure that billing is enabled for your Google Cloud project. [Learn how to confirm billing is enabled for your project](https://cloud.google.com/billing/docs/how-to/modify-project) (https://cloud.google.com/billing/docs/how-to/modify-project).
4. Enable the Firestore API.

[ENABLE THE API](https://console.cloud.google.com/flows/enableapi?apiid=firestore) (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/FLOWS/ENABLEAPI?APIID=FIREFSTORE)

5. [Install and initialize the Cloud SDK](https://cloud.google.com/sdk/docs/) (https://cloud.google.com/sdk/docs/).
6. Install Python, pip, and virtualenv on your system. For instructions, see [Setting up a Python development environment](https://cloud.google.com/python/setup) (https://cloud.google.com/python/setup) for Google Cloud.

Setting up the project

1. In your terminal window, start in a directory of your choosing and create a new directory named `sessions`. All of the code for this tutorial is inside the `sessions` directory.
2. Change into the `sessions` directory:

```
cd sessions
```



3. Create the `requirements.txt` with the following contents:

```
sessions/requirements.txt
```

```
(https://github.com/GoogleCloudPlatform/getting-started-python/blob/master/sessions/requirements.txt)
```

```
GOOGLECLOUDPLATFORM/GETTING-STARTED-PYTHON/BLOB/MASTER/SESSIONS/REQUIREMENTS.TXT)
```

```
google-cloud-firestore==1.6.0  
flask==1.1.1
```

4. Install the dependencies:

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

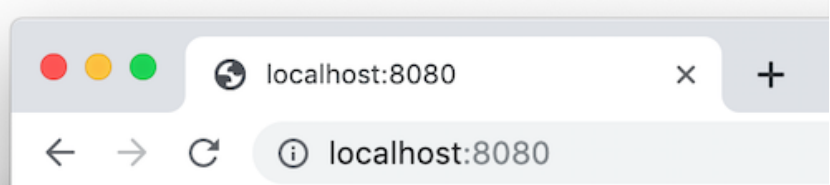
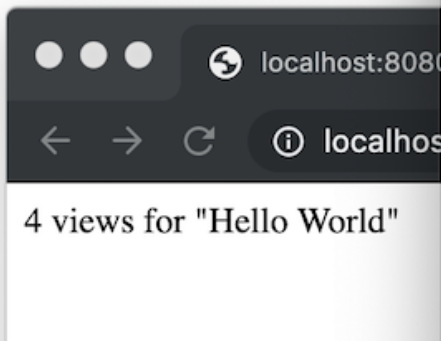
At the end of this tutorial, the final file structure is similar to the following:

```
sessions  
├─ app.yaml  
├─ main.py  
└─ requirements.txt
```

Writing the web app

This app displays greetings in different languages for every user. Returning users are always greeted in the same language.

2 views for "Salut le Monde"



Before your app can store preferences for a user, you need a way to store information about the current user in a session. This sample app uses Firestore to store session data.

- In your terminal window, create a file called `main.py` with the following content:

```
sessions/main.py
```

```
(https://github.com/GoogleCloudPlatform/getting-started-python/blob/master/sessions/main.py)
```

```
HUB.COM/GOOGLECLOUDPLATFORM/GETTING-STARTED-PYTHON/BLOB/MASTER/SESSIONS/MAIN.PY)
```

```
import random
from uuid import uuid4

from flask import Flask, make_response, request
from google.cloud import firestore

app = Flask(__name__)
db = firestore.Client()
sessions = db.collection('sessions')
greetings = [
    'Hello World',
    'Hallo Welt',
    'Ciao Mondo',
    'Salut le Monde',
    'Hola Mundo',
]

@firestore.transactional
def get_session_data(transaction, session_id):
    """ Looks up (or creates) the session with the given session_id.
        Creates a random session_id if none is provided. Increments
        the number of views in this session. Updates are done in a
        transaction to make sure no saved increments are overwritten.
    """
    if session_id is None:
        session_id = str(uuid4()) # Random, unique identifier

    doc_ref = sessions.document(document_id=session_id)
    doc = doc_ref.get(transaction=transaction)
    if doc.exists:
        session = doc.to_dict()
    else:
        session = {
```

```
        'greeting': random.choice(greetings),
        'views': 0
    }

    session['views'] += 1    # This counts as a view
    transaction.set(doc_ref, session)

    session['session_id'] = session_id
    return session

@app.route('/', methods=['GET'])
def home():
    template = '<body>{} views for "{}"</body>'

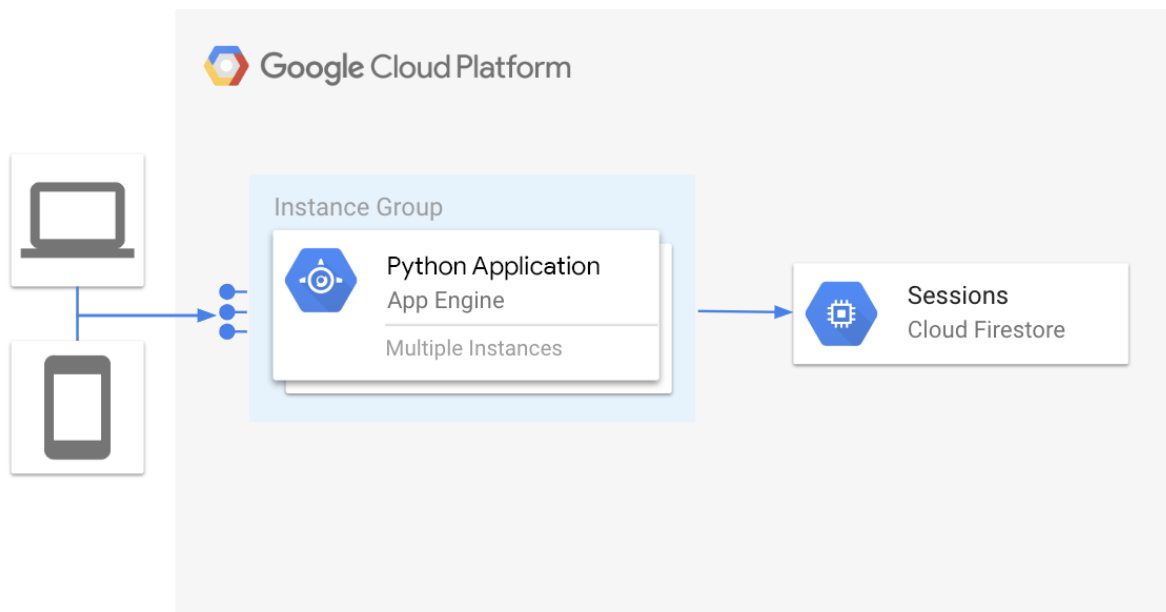
    transaction = db.transaction()
    session = get_session_data(transaction, request.cookies.get('session_id'))

    resp = make_response(template.format(
        session['views'],
        session['greeting']
    ))
    resp.set_cookie('session_id', session['session_id'], httponly=True)
    return resp

if __name__ == '__main__':
    app.run(host='127.0.0.1', port=8080)
```

- The following diagram illustrates how Firestore handles sessions for the App Engine app.

Architecture: Session Handling with Firestore



After you've set `app.use(session)`, every request has a `req.session` property that you can use to identify recurring users. The session data is stored in Firestore.

Deleting sessions

You can [delete session data](https://cloud.google.com/firestore/docs/using-console#delete_data) (https://cloud.google.com/firestore/docs/using-console#delete_data) or implement an automated deletion strategy. If you use storage solutions for sessions such as Memcache or Redis, expired sessions are automatically deleted.

Running locally

1. In your terminal window, install the [Gunicorn HTTP server](https://gunicorn.org/) (https://gunicorn.org/):

```
pip install gunicorn
```

2. Run the Gunicorn HTTP server:

```
gunicorn -b :8080 main:app
```

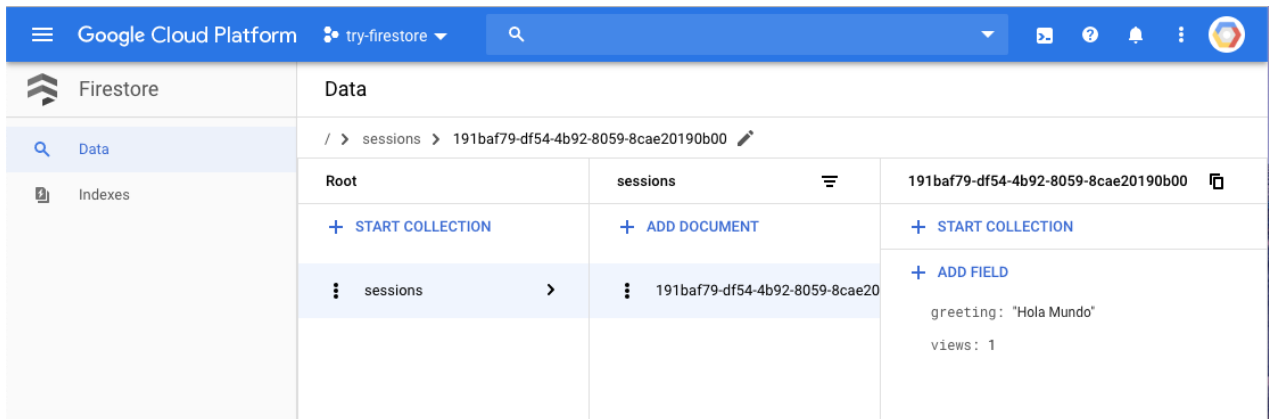


3. View the app in your web browser:

CLOUD SHELL
LOCAL MACHINE

In the Cloud Shell toolbar, click **Web preview** and select **Preview on port 8080**.

You see one of five greetings: "Hello World", "Hallo Welt", "Hola mundo", "Salut le Monde", or "Ciao Mondo." The language changes if you open the page in a different browser or in incognito mode. You can see and edit the session data in the [Google Cloud Console](https://console.cloud.google.com/firestore/) (<https://console.cloud.google.com/firestore/>).



4. To stop the HTTP server, in your terminal window, press **Control+C**.

Deploying and running on App Engine

You can use the [App Engine standard environment](https://cloud.google.com/appengine/docs/standard/) (<https://cloud.google.com/appengine/docs/standard/>) to build and deploy an app that runs reliably under heavy load and with large amounts of data.

This tutorial uses the App Engine standard environment to deploy the server.

1. In your terminal window, create an `app.yaml` file and copy the following:

```
sessions/app.yaml
(https://github.com/GoogleCloudPlatform/getting-started-python/blob/master/sessions/app.yaml)
```

IUB.COM/GOOGLECLOUDPLATFORM/GETTING-STARTED-PYTHON/BLOB/MASTER/SESSIONS/APP.YAML)

```
runtime: python37
```

2. Deploy the app on App Engine:

```
gcloud app deploy
```

3. View the live app at <https://your-project-id.appspot.com>:

```
gcloud app browse
```

Where **your-project-id** is your Google Cloud project ID.

The greeting is now delivered by a web server running on an App Engine instance.

Debugging the app

If you cannot connect to your App Engine app, check the following:

1. Check that the `gcloud` deploy commands successfully completed and didn't output any errors. If there were errors (for example, `message=Build failed`), fix them, and try [deploying the App Engine app](#) (`#deploying_the_web_app`) again.
2. In the Cloud Console, go to the **Logs Viewer** page.

[GO TO LOGS VIEWER PAGE \(HTTPS://CONSOLE.CLOUD.GOOGLE.COM/LOGS/VIEWER\)](https://console.cloud.google.com/logs/viewer)

- a. In the **Recently selected resources** drop-down list, click **App Engine Application**, and then click **All module_id**. You see a list of requests from when you visited your app. If you don't see a list of requests, confirm you selected **All module_id** from the drop-down list. If you see error messages printed to the Cloud Console, check that your app's code matches [the code in the section about writing the web app](#) (`#writing_the_web_app`).
- b. Make sure that the Firestore API is enabled.

Cleaning up

Delete the 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.

If you plan to explore multiple tutorials and quickstarts, reusing projects can help you avoid exceeding project quota limits.

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.

Delete the App Engine instance


1. In the Cloud Console, go to the **Versions** page for App Engine.

[GO TO THE VERSIONS PAGE \(HTTPS://CONSOLE.CLOUD.GOOGLE.COM/APPENGINE/VERSIONS\)](https://console.cloud.google.com/appengine/versions)

2. Select the checkbox for the non-default app version you want to delete.

Note: The only way you can delete the default version of your App Engine app is by deleting your project. However, you can [stop the default version in the Cloud Console \(https://console.cloud.google.com/appengine/versions\)](https://console.cloud.google.com/appengine/versions). This action shuts down all instances associated with the version. You can restart these instances later if needed.

In the App Engine standard environment, you can stop the default version only if your app has manual or basic scaling.

3. Click **Delete**  to delete the app version.

What's next

- Try additional [Cloud Functions tutorials](https://cloud.google.com/functions/docs/tutorials/) (https://cloud.google.com/functions/docs/tutorials/).
- [Learn more about App Engine](https://cloud.google.com/appengine/docs/) (https://cloud.google.com/appengine/docs/).
- [Try Cloud Run](https://cloud.google.com/run/docs/quickstarts/prebuilt-deploy) (https://cloud.google.com/run/docs/quickstarts/prebuilt-deploy), which lets you run stateless containers on a fully managed environment or in your own Google Kubernetes Engine cluster.

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 5, 2020.