Python (https://cloud.google.com/python/) Guides

Running Django on the App Engine standard environment

Django apps that run on App Engine standard environment (https://cloud.google.com/appengine) scale dynamically according to traffic.

This tutorial assumes that you're familiar with Django web development. If you're new to Django development, it's a good idea to work through writing your first Django app (https://docs.djangoproject.com/en/1.11/intro/tutorial01/) before continuing. In that tutorial, the app's models represent polls that contain questions, and you can interact with the models by using the Django admin console.

This tutorial requires Python 3.7 (https://www.python.org/).

Before you begin

1. Sign in (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).

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/PROJECTSELECT

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).
4. Install and initialize the Cloud SDK (https://cloud.google.com/sdk/docs/).
5. Enable the Datastore, Pub/Sub, Cloud Storage JSON, Stackdriver Logging, and Google+ APIs.

   ENABLE THE APIS (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/FLOWS/ENABLEAPI?APIID=DATASTOR

Log in to gcloud

Acquire new credentials to use the Cloud SQL Admin API:

```
gcloud auth application-default login
```

Downloading and running the app

After you've completed the prerequisites, download and deploy the Django sample app. The following sections guide you through configuring, running, and deploying the app.

Cloning the Django app

The code for the Django sample app is in the `GoogleCloudPlatform/python-docs-samples` (https://github.com/GoogleCloudPlatform/python-docs-samples) repository on GitHub.

1. You can either download the sample (https://github.com/GoogleCloudPlatform/python-docs-samples/archive/master.zip) as a zip file and extract it or clone the repository to your local machine:

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

2. Go to the directory that contains the sample code:

   LINUX/MACOS WINDOWS

   ```
   cd python-docs-samples/appengine/standard_python37/django
   ```

Setting up your local environment

When deployed, your app uses the Cloud SQL Proxy that is built in to the App Engine environment to communicate with your Cloud SQL instance. However, to test your app locally, you must install and use a local copy of the proxy in your development environment.

Learn more about the Cloud SQL Proxy (https://cloud.google.com/sql/docs/mysql/sql-proxy).

To perform basic admin tasks on your Cloud SQL instance, you can use the MySQL client.

Note: You must authenticate `gcloud` (https://cloud.google.com/sql/docs/mysql/sql-proxy#gcloud) before you can use the Cloud SQL Proxy to connect from your local machine.

Enable the Cloud SQL Admin API

Before using Cloud SQL, you must enable the Cloud SQL Admin API:

```
gcloud services enable sqladmin
```

Installing the Cloud SQL Proxy

Download and install the Cloud SQL Proxy. The Cloud SQL Proxy connects to your Cloud SQL instance when running locally.

LINUX 64-BIT   LINUX 32-BIT   MORE

1. Download the proxy:

```
wget https://dl.google.com/cloudsql/cloud_sql_proxy.linux.amd64 -O cloud_sql_
```

2. Make the proxy executable:

```
chmod +x cloud_sql_proxy
```

If your operating system isn't included here, you can also compile the proxy from source (http://github.com/GoogleCloudPlatform/cloudsql-proxy).

Creating a Cloud SQL instance

1. Create a Cloud SQL for MySQL Second Generation instance.
   (https://cloud.google.com/sql/docs/mysql/create-instance)

   Name the instance `polls-instance` or similar. It can take a few minutes for the instance to be ready. When the instance is ready, it's visible in the instances list.

★ Make sure that you create a **Second Generation** instance.

2. Use the Cloud SDK to run the following command where `[YOUR_INSTANCE_NAME]` represents the name of your Cloud SQL instance:

```
gcloud sql instances describe [YOUR_INSTANCE_NAME]
```

   In the output, note the value shown for `[CONNECTION_NAME]`.

   The `[CONNECTION_NAME]` value is in the format `[PROJECT_NAME]:[REGION_NAME]:[INSTANCE_NAME]`.

Initializing your Cloud SQL instance

1. Start the Cloud SQL Proxy by using the `[CONNECTION_NAME]` value from the previous step:

   LINUX/MACOS WINDOWS

```
./cloud_sql_proxy -instances="[YOUR_INSTANCE_CONNECTION_NAME]"=tcp:3306
```

   Replace `[YOUR_INSTANCE_CONNECTION_NAME]` with the `[CONNECTION_NAME]` value that you recorded in the previous step.

   This step establishes a connection from your local computer to your Cloud SQL instance for local testing purposes. Keep the Cloud SQL Proxy running the entire time you test your app locally.

2. Create a Cloud SQL user and database:

   CLOUD CONSOLE MYSQL CLIENT

a. Create a new database by using the Cloud Console (https://cloud.google.com/sql/docs/mysql/create-manage-databases#create) for your Cloud SQL instance `polls-instance`. For example, you can use the name `polls`.

b. Create a new user by using the Cloud Console (https://cloud.google.com/sql/docs/mysql/create-manage-users#creating) for your Cloud SQL instance `polls-instance`.

Configuring the database settings

1. Open `mysite/settings.py` for editing.
   a. To help set up the connection to the database for both App Engine deployment and local testing, set `<your-database-user>` and `<your-database-password>` to the username and password you created previously in the step Creating a Cloud SQL instance (#creating_a_cloud_sql_instance).
   b. Get the values for your instance:

   ```
   gcloud sql instances describe [YOUR_INSTANCE_NAME]
   ```

   c. From the output, copy the `connectionName` value for use in the next step.
   d. Set `<your-cloudsql-connection-string>` to the `connectionName` from the previous step.
   e. Set `[YOUR-DATABASE]` to the name you chose during the Initialize your Cloud SQL instance (#initializing_your_cloud_sql_instance) step.
2. Close and save `settings.py`.

Running the app on your local computer

1. To run the Django app on your local computer, set up a Python development environment (https://cloud.google.com/python/setup), including Python, `pip`, and `virtualenv`.
2. Create an isolated Python environment, and install dependencies:

LINUX/MACOS WINDOWS

```
virtualenv env
source env/bin/activate
pip install -r requirements.txt
```

3. Run the Django migrations to set up your models:

```
python manage.py makemigrations
python manage.py makemigrations polls
python manage.py migrate
```

4. Start a local web server:

```
python manage.py runserver
```

5. In your browser, go to http://localhost:8000 (http://localhost:8000):

```
http://localhost:8000
```

   The page displays the following text: "No polls are available." The Django web server running on your computer delivers the sample app pages.

6. Press `Control+C` to stop the local web server.

Using the Django admin console

1. Create a superuser. You need to define a username and password.

```
python manage.py createsuperuser
```

2. Start a local web server:

```
python manage.py runserver
```

3. In your browser, go to http://localhost:8000/admin (http://localhost:8000/admin).

```
http://localhost:8000/admin
```

4. Log in to the admin site using the username and password you used when you ran `createsuperuser`.

Deploying the app to the App Engine standard environment

1. Gather all the static content into one folder by moving all of the app's static files into the folder specified by `STATIC_ROOT` in `settings.py`:

```
python manage.py collectstatic
```

2. Upload the app by running the following command from within the `python-docs-samples/appengine/standard/django` directory of the app where the `app.yaml` file is located:

```
gcloud app deploy
```

   Wait for the message that notifies you that the update has completed.

Seeing the app run in Google Cloud

The following command deploys the app as described in `app.yaml` and sets the newly deployed version as the default version, causing it to serve all new traffic.

- In your browser, enter the following address. Replace `<your-project-id>` with your Google Cloud project ID.

```
https://<your-project-id>.appspot.com
```

Your request is served by a web server running in the App Engine standard environment.

If you update your app, you deploy the updated version by entering the same command that you used to deploy the app. The deployment creates a new version (https://console.cloud.google.com/project/_/appengine/versions) of your app and promotes it to the default version. The earlier versions of your app remain. All of these app versions are billable resources. To reduce costs, delete the non-default versions of your app.

For information about deleting the non-default versions of your app, see Cleaning up (https://cloud.google.com/python/getting-started/delete-tutorial-resources).

Production

When you are ready to serve your content in production, in `mysite/settings.py`, change the `DEBUG` variable to `False`.

Note: In this tutorial, credentials for connecting to your SQL database are added to the code. In a production setting, however, you should use Cloud Key Management Service (https://cloud.google.com/kms) and Cloud IAM (https://cloud.google.com/iam) to manage and control access securely.

Understanding the code

The Django sample app was created using standard Django tooling.

- The following commands create the project and the polls app:

```
django-admin startproject mysite

python manage.py startapp polls
```

- The `settings.py` file contains the configuration for your SQL database. The code in `settings.py` uses the `GAE_APPLICATION` environment variable to determine whether the app is running on App Engine or running on your local computer:
  - When the app runs on App Engine, it connects to the MySQL host by using the `/cloudsql` Unix socket.
  - When the app runs on your local computer, it connects to the MySQL host by using TCP, which requires a username and password.

appengine/standard_python37/django/mysite/settings.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/appengine/standard_python37/django/mysite/settings.py)

N-DOCS-SAMPLES/BLOB/MASTER/APPENGINE/STANDARD_PYTHON37/DJANGO/MYSITE/SETTINGS.PY)

```
if os.getenv('GAE_APPLICATION', None):
    # Running on production App Engine, so connect to Google Cloud SQL using
    # the unix socket at /cloudsql/<your-cloudsql-connection string>
    DATABASES = {
        'default': {
            'ENGINE': 'django.db.backends.mysql',
            'HOST': '/cloudsql/[YOUR-CONNECTION-NAME]',
            'USER': '[YOUR-USERNAME]',
            'PASSWORD': '[YOUR-PASSWORD]',
            'NAME': '[YOUR-DATABASE]',
        }
    }
else:
    # Running locally so connect to either a local MySQL instance or connect to
    # Cloud SQL via the proxy. To start the proxy via command line:
    #
    #     $ cloud_sql_proxy -instances=[INSTANCE_CONNECTION_NAME]=tcp:3306
    #
    # See https://cloud.google.com/sql/docs/mysql-connect-proxy
    DATABASES = {
        'default': {
            'ENGINE': 'django.db.backends.mysql',
            'HOST': '127.0.0.1',
            'PORT': '3306',
            'NAME': '[YOUR-DATABASE]',
            'USER': '[YOUR-USERNAME]',
            'PASSWORD': '[YOUR-PASSWORD]',
        }
    }
```

- The `app.yaml` (https://cloud.google.com/appengine/docs/standard/python3/config/appref) file contains configuration information for deployment to App Engine. This `app.yaml` file specifies that App Engine serves static files from the `static/` directory:

appengine/standard_python37/django/app.yaml (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/appengine/standard_python37/django/app.yaml)

)RM/PYTHON-DOCS-SAMPLES/BLOB/MASTER/APPENGINE/STANDARD_PYTHON37/DJANGO/APP.YAML)

```
runtime: python37

handlers:
# This configures Google App Engine to serve the files in the app's static
# directory.
```

```
- url: /static
  static_dir: static/

# This handler routes all requests not caught above to your main app. It is
# required when static routes are defined, but can be omitted (along with
# the entire handlers section) when there are no static files defined.
- url: /.*
  script: auto
```