Udacity's Google App Engine Installation Guide

In this document, we will go through the different steps needed to get the Google App Engine (GAE) up and running for the Web Development course. You should consider this text as guiding material to supplement what is available on the web regarding the GAE.

GAE setup

In the Web Development course, we will work with Google App Engine using the Python software development kit (SDK) associated with it. Here are the three steps needed to get GAE set up:

- Sign up for GAE.
- Have Python installed in your system.
- Install the GAE Python SDK.

You can sign up for Google App Engine here. Notice once you sign up, you will be taken to Google's tour of the GAE. This installation guide follows basically the same flow, so you should not find the two too different. At the end of the day, whichever way you decide to go about it, make sure you have met the three steps outline above for a correct setup.

If you do not have Python installed already in your computer, make sure that you install Python 2.7 for your platform from the Python Web Site.

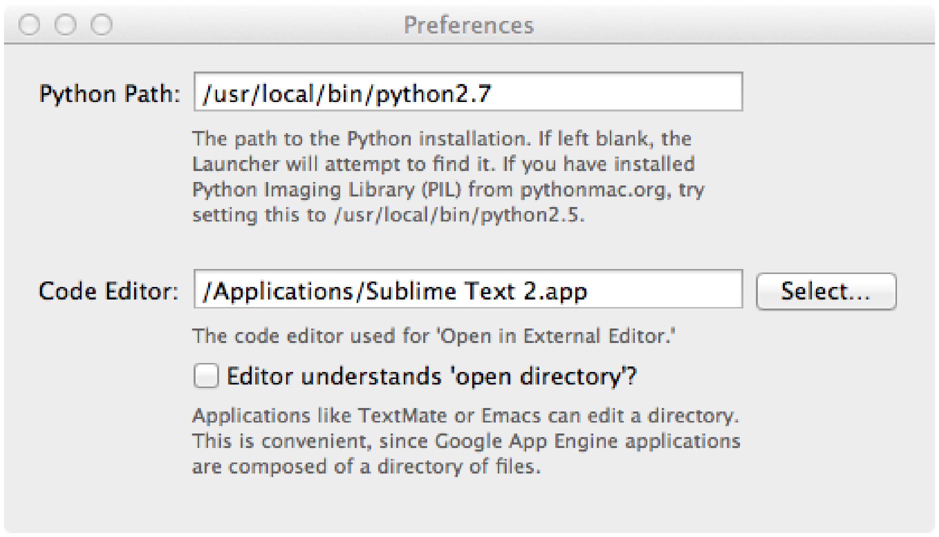
In order to install the GAE Python SDK (Google Cloud SDK, another name you might encounter on the web), you can download it from here. The Python SDK includes a web server application that simulates the App Engine environment. The Python SDK runs on any computer with Python 2.7 and versions are available for Windows, Mac OSX, and Linux. (Note: The Python SDK is not compatible with Python 3.x)

Have you checked off the three steps below needed for the installation? If so, you should have successfully set up the GAE, now let's learn how to get a web app running with it.

✓ Sign up for GAE.
✓ Have Python installed in your system.
✓ Install the GAE Python SDK.

Instructions for Command-Line Users
To configure Google App Engine Launcher for the command line, you will want to perform a couple of configurations first.

First, you will need to set the preferences in Google App Engine so that it knows where Python is installed. To do this, go to the Google App Engine preferences, and input the Python path. You may also set your preferred text editor. These settings define how you can interact with Google App Engine from the command line.



Python Path: /usr/local/bin/python2.7

The path to the Python installation. If left blank, the Launcher will attempt to find it. If you have installed Python Imaging Library (PIL) from pythonmac.org, try setting this to /usr/local/bin/python2.5.

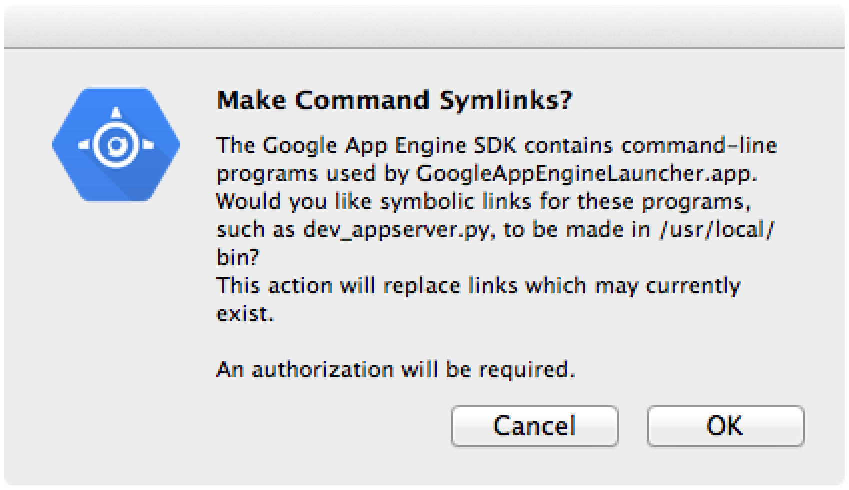
Code Editor: /Applications/Sublime Text 2.app

The code editor used for 'Open in External Editor.'

☐ Editor understands 'open directory'?

Applications like TextMate or Emacs can edit a directory. This is convenient, since Google App Engine applications are composed of a directory of files.

Next, you need to create symlinks for GAE. To do this, click on the GoogleAppEngineLauncher menu, and select ‘Make Symlinks’ and approve the request by clicking ‘OK’.


Make Command Symlinks?

The Google App Engine SDK contains command-line programs used by GoogleAppEngineLauncher.app. Would you like symbolic links for these programs, such as dev_appserver.py, to be made in /usr/local/bin?
This action will replace links which may currently exist.

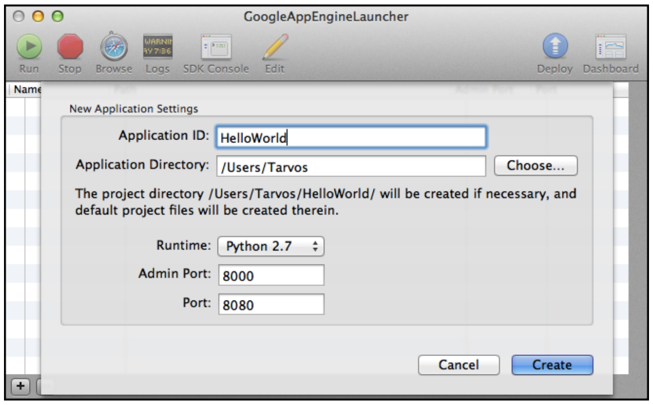
An authorization will be required.

GAE in action

Now that we have the GAE setup, let's see it in action. Remember the GAE allows you to run your web apps locally for testing and on the web once it's ready for the public view; and you can do each one of these options either via the GAE Launcher or via the command-line. We will take a look at how to do all of these.

Running your app locally
Web developers usually feel comfortable in the localhost, and so will you. Running your app locally means you can use the localhost as a playground for testing, and it should be a place you are familiar with. Let's take a look at two different ways GAE lets you run locally.

Option 1: Using the GAE Launcher
Open the Google App Engine Launcher and Select **File > New Application**



New Application Settings

Application ID: HelloWorld

Application Directory: /Users/Tarvos

The project directory /Users/Tarvos/HelloWorld/ will be created if necessary, and default project files will be created therein.

Runtime: Python 2.7

Admin Port: 8000

Port: 8080

Assign a unique name for your app and hit create. You need to make sure the name matches the one on your GAE account. This will automatically create all the files you need to run your application.

At this point, you can run the application by selecting the application in the app list, click the **Run** button to start the application, then click the **Browse** button to view it. Clicking **Browse** simply loads (or reloads) http://localhost:8080/ in your default web browser. Feel free to give this a try!

Option 2: Using the command-line
Create a directory named helloworld. All files for this application reside in this directory.

1) Creating the Request Handler
Inside the helloworld directory, create a file named main.py, and give it the following contents:

```
import webapp2
class MainPage(webapp2.RequestHandler):
    def get(self):
        self.response.write('Hello World!')

app = webapp2.WSGIApplication([
    ('/', MainPage),
], debug=True)
```

Notice the indentation looks a bit weird in this piece of code, but don’t let it throw you off. You can simply copy and paste the piece of code directly into your text editor.

2) Creating the Configuration Files
An App Engine application has a configuration file called app.yaml. Among other things, this file describes which handler scripts should be used for which URLs. Inside the helloworld directory, create a file named **app.yaml** with the content below. You need to make sure the application name matches the one on your GAE account.

```
application: HelloWorld
version: 1
runtime: python27
api_version: 1
threadsafe: yes

handlers:
- url: /favicon\.ico
  static_files: favicon.ico
  upload: favicon\.ico

- url: .*
  script: main.app
```

```
libraries:
- name: webapp2
  version: "2.5.2"
```

3)Testing the Application
In the terminal, move to your helloworld directory. Now you can run your app locally with the following command:

```
dev_appserver.py .
```

Don't forget to include the dot, this tells the GAE to run locally with the files in the current directory. The web server is now running, listening for requests on port 8080. You can test the application by visiting the following URL in your web browser: http://localhost:8080/

Note: If you get the error message `"command not found"` when attempting to run the `dev_appserver.py .` command, you need to setup the symlinks to allow GAE to communicate with your command-line. Check out the instructions for command-line users for how to set this up.

For more information about running the development web server, including how to change which port it uses, see the Dev Web Server reference or run the command with the option `----help`.

Deploying your app on the web
Once you have tested your app locally and it's working properly in your machine, it is ready for deploying. What this means is that you will be able to upload your web app to the internet, and have it be accessible to the whole world. Let's take a the couple of different ways you can deploy your app with GAE:

Option 1: Using the GAE Launcher
Click **Deploy** in the Google App Engine Launcher and enter your Google username and password at the prompts. Now you should be able to see your application live at: http://***your-app-id***.appspot.com (Note: replace yourappid with an application name you created here).

Option 2: Using the command-line
To upload your finished application to Google App Engine, run the following command from your working directory:

```
appcfg.py update .
```

Don't forget to include the dot, this tells the GAE to run locally with the files in the current directory. Now you should be able to see your application live at: http://***your- app- id.***appspot.com (Note: replace yourappid with an application name you created here).

Note: If you get the error message `“command not found”` when attempting to run the `appcfg.py update .` command, you need to setup the symlinks to allow GAE to communicate with your command-line. Check out the instructions for command-line users for how to set this up.

Additional notes and readings

- Info about the Configuration File (app.yaml)
- Info about the High Replication Datastore