Installing App Engine with Google Cloud SDK

For MacOS and Linux

This document contains instructions to get Google App Engine up and running via the lastest method that is supported by Google. Previous methods are being phased out and support will soon stop. All of Google's cloud developer products now sit in a command line tool called Google Cloud SDK (the `gcloud` command).

If you still have the old, standalone App Engine Launcher (the graphical program), it is advised to uninstall it and switch to the Google Cloud SDK method, as described in this document.

These instructions are based on Google's quickstart guide, but with missing steps added and clarifications for the MacOS and Linux platforms.

Google Account

If you haven't already got a Google account, you can register for one here.

Create App Engine Project

Next you'll create your first App Engine project. Start by navigating to https://console.cloud.google.com/start, select email contact preference and agree to Google Cloud Platform's terms and conditions.

You can dismiss the banner at the top of the page regarding signing up for a free trial. There is no need to signup further in order to create a project or give any billing details.

Click on the `Projects` drop down in the menu bar at the top of the page and click `Create project`. Type hello-world as the name of your project. Note that a unique project ID is created on the line below. This is the ID that you will use when working with the `gcloud` command.

Wait around 30 seconds for your new project to be created and for its dashboard to load. Your unique project ID can also be found at the end of the URL on this page.

Python

Please check you have a version of Python 2.7 with the following command:

```
python -V
```

If the python command is not found or the version is lower than 2.7, please install or update Python. For MacOS, installers are provided on the python.org site, or use the Homebrew package manager. For Linux, use your distribution's package manager (`apt-get` for Ubuntu, `yum` for Red Hat/CentOS) or download from python.org.

Using Python3 alongside Python2

If you find that you have both Python3 and Python2 installed on your machine and the default is Python 3, then there is a way to tell Google Cloud SDK which one to use.

You will need to set the environment variable `CLOUDSDK_PYTHON` to point to the Python 2 executable. You can do this by adding the following line to the file "`~/.bash_profile`" in your user home directory:

```
export CLOUDSDK_PYTHON=/path/to/python2
```

You can find out the full path on your system with the command "`which python2`" and replace the dummy path.

Then run this command to update the current terminal with the new setting (no need to do this step with new terminal windows):

```
source ~/.bash_profile
```

That should allow Google Cloud SDK to install and run properly.

Install Google Cloud SDK

Downloading and extracting on MacOS

The following detailed steps are based on these instructions, provided by Google. Visit that link and download the correct package file, depending whether your OS is 64 or 32-bit. Save it to the root of your home directory (`~/`).

Extract the gzip file here, either with your file manager or with the following console command:

```
tar -zxvf google-cloud-sdk-nnn.n.n-darwin-x86[_64].tar.gz
```

The contents of the archive should now being in the directory `google-cloud-sdk` in the root of your home directory.

Downloading and extracting on Linux

It is recommended to use the general method for installing the Google Cloud SDK (GCS) on to a Linux machine, rather than the packages provided for some distributions. GCS can update itself and its components. It will provide a notice about available updates, when they are available.

The following detailed steps are based on these instructions, provided by Google. Visit that link and download the correct package file, depending whether your OS is 64 or 32-bit. Save it to the root of your home directory (`~/`).

Extract the gzip file here, either with your file manager or with the following console command:

```
tar -zxvf google-cloud-sdk-nnn.n.n-linux-x86[_64].tar.gz
```

The contents of the archive should now being in the directory `google-cloud-sdk` in the root of your home directory.

Run installation script and initialize GCS

Next, run the following install script to add the GCS tools to your path, etc.:

```
./google-cloud-sdk/install.sh
```

Now close your terminal and open a new one, to allow for the changes to take effect.

You now need to initialise the SDK - allowing it access to your Google account and set the default configuration. You should be able now to just issue the command:

```
gcloud init
```

...as the `gcloud` command is now on your path. If it didn't work, you can give the full path (the below path assumes you have installed google-cloud-sdk to your home directory) to the command instead:

```
./google-cloud-sdk/bin/gcloud init
```

You will be asked to login to your Google Account (or if you are already logged in, give GCS access to your account).

Since you have only one project, this project will be your default project. As you add more projects, you can either reset the default (see `gcloud topic configurations` to learn more) to another project or specify which project you want to operate on with the `--project <project id>` option to `gcloud` commands.

Location of configurations and credentials on Linux

If you ever need to delete your GCS configurations and login credentials, these are stored in the following location:

```
~/.config/gcloud
```

Install App Engine Python Extensions

Since the `gcloud` command can manage all of Google's cloud development platforms, not everything is installed by default. You need to install the Python version of App Engine that is used throughout this Nanodegree.

You can list currently installed and available components with the command:

```
gcloud components list
```

Go ahead and install the App Engine Python Extensions with the command:

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

To find out more information about the available components, navigate your browser here.

Your first App Engine webapp

If you have Git installed (highly recommended), you can clone Google's Python sample code repository with the command:

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

If not, you can download a zip archive from here:
https://github.com/GoogleCloudPlatform/python-docs-samples/archive/master.zip

Once you have the code, change your current directory with the following command:

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

To run App Engine locally, you use the command `dev_appserver.py` - a Python script program. If this file is in your user PATH environment variable, the next command to run is:

```
dev_appserver.py .
```

Notice the dot (or full stop) on the end of the command. That's very important and tells the command to use the current directory to search for a file called `app.yaml`. This file

configures the App Engine webapp. Giving `dev_appserver.py` this file directly as an argument will also work.

If you find the command `dev_appserver.py` is not found, either add it's location to your path (instructions here), or pass the file to Python with the command:

```
python ~/google-cloud-sdk/bin/dev_appserver.py .
```

Firewall

Depending on the configuration of your firewall software, you may receive a pop-up notification asking to give Python permission to accept incoming network connections. In order to proceed, please allow these connections for Python.

Browse your first app

To look at the web output of the your first app, go to this address in your browser:

`http://localhost:8080`

Each App Engine app also has an admin server at the following address:

`http://localhost:8000`

This has many useful features, but one you will be using a lot is the Datastore Viewer. There's not a lot to see at the moment, but come back to this page when you start putting data into the Datastore:

`http://localhost:8000/datastore`

Make a change

Open up the file `main.py` in a text editor. Line 21 contains the text that is currently output to the browser. Change it to something else, like:

```
self.response.write('Goodnight, World!')
```

The reload the page `http://localhost:8080` and you should see the updated message.

Stop the server

To stop the local server, press the key combination `Ctrl+C`.

Deploy your app to the cloud

You can deploy your app to the cloud so anyone in the world can view it with the following command:

```
gcloud app deploy
```

This will deploy your default project in the current configuration. You can also specify another project or version of a project. More details on this options are here.

Once deployed, your app will be available to view and use at the address `http://[YOUR_PROJECT_ID].appspot.com`. Or issue the command:

```
gcloud app browse
```

which will open up your default browser to the public URL for the project. This may not work though, depending on your configuration.

Updating indexes

As you develop your app and it gets more complex, the local `index.yaml` file will get populated with the indexes needed by your app. When you deploy your app, you might encounter errors with the deployed app, as it takes time for the indexes to be built.

A way to speed things up is it manually deploy the `index.yaml` file with:

```
gcloud app deploy index.yaml
```

Further details on updating indexes are available here.

Review the Hello World code

Please review the code explanation of both the `main.py` and `app.yaml` files used in this example, in order to better understand what is going on.

Feedback

If you have any comments, questions or suggestions about this document, please mention @swooding on the Forums.

Credits

Lead author….	Steven Wooding
Additional content….	Renee MacDonald

Reviewed by…. Full Stack Nanodegree Mentors