Installing GCP SDK python

Google Developer Console pre-requisite tasks

Create a project, and make note of this metadata:
Project ID: **some-gcp-project-id**

Ensure that these Google Cloud APIs are enabled on your project:
- BigQuery API
- Cloud Storage API
- Cloud Datastore API

(if you want to build out this developer environment on a Google Compute Engine, enable that API as well).

Create a Google Cloud Platform (GCP) Service Account. And make note of this metadata:
Client ID
some-gcp-service-acct.apps.googleusercontent.com
Email address
some-gcp-service-acct@developer.gserviceaccount.com

For that service account generate a new P12 key (for use by gcloud.datastore, gcloud.storage), and generate a new JSON key (for use with google api client used to access BigQuery). In the process of creating these key credential files are placed in $HOME/Downloads:
- some-service-acct-key.json
- some-service-acct-key.p12

Subsequently, you will use the 'gcloud' command to create a key credential JSON file (which will likewise be downloaded into $HOME/Downloads) for your individual gmail account. Initially, your individual gmail account becomes the 'active', or default, credentialed user account with GCP which gcloud uses. However, it is best practice to create a credentialed GCP service account, and to configure gcloud to work with the service account. That service account becomes the new default gcloud account. As such, all Python scripts which import gcloud.datastore, gcloud.storage, and gcloud.pubsub.

Fedora 22 pre-requsite tasks

You cannot install gcloud without a C compiler installed on host
:q:sudo yum groupinstall 'Development Tools'

Debian
sudo apt-get update && apt-get upgrade
sudo apt-get install build-essential

you cannot install gcloud using pip without first installing 'pycrypto'
sudo yum install pycrypto

```
sudo apt-get install python-crypto
sudo apt-get install python-dev
```

To complete the base modules needed by the GCP SDK, also run
sudo yum install python-devel
$ sudo wget https://bootstrap.pypa.io/get-pip.py

$ sudo python get-pip.py

$ sudo pip install --upgrade google-api-python-client httplib2 argparse

CentOS 6.6. GCE

cd /usr/src
sudo wget https:/www.python.org/ftp/python/2.7.6/Python-2.7.6.tgz
sudo tar xzf Python-2.7.6.tgz
sudo cd Python-2.7.6
sudo ./configure
sudo make altinstall

Environment Variables

Set and or add the following environment variables. Most importantly, GCP SDK Python works with Python 2.7.x, which is pre-installed on Fedora 22.

The Linux account running the gcloud-python scripts must use Python 2.7, and within its .bashrc file, it must set around a dozen environment variables. First, create the $PYTHONPATH environment variable and set it to the local Python 2.7 site-packages directory.

PYTHON 2.7
#export PYTHONHOME=/usr/bin/python2.7
export PYTHONPATH=/usr/lib/python2.7/site-packages

Next, in order to use the GCP SDK consoles and command line tools, their location has to be appended to the existing $PATH environment variable

add gcloud bin to the $PATH

Next, in order for the Python shell or a Python script to import GCP SDK modules, their locations have to be appended to the $PTYHONPATH environment variable.

PYTHON modules for GCP SDK

```
export PYTHONPATH=$PYTHONPATH:/usr/lib/python2.7/site-packages/gcloud
export PYTHONPATH=$PYTHONPATH:/usr/lib/python2.7/site-packages/gcloud/datastore
export PYTHONPATH=$PYTHONPATH:/usr/lib/python2.7/site-packages/gcloud/pubsub
export PYTHONPATH=$PYTHONPATH:/usr/lib/python2.7/site-packages/gcloud/storage
export PYTHONPATH=$PYTHONPATH:/home/yourhomedir/google-cloud-sdk/platform/bq
export PYTHONPATH=$PYTHONPATH:/home/yourhomedir/google-cloud-sdk/platform/gcutil
export PYTHONPATH=$PYTHONPATH:/home/yourhomedir/google-cloud-sdk/platform/gcutil/lib
export PYTHONPATH=$PYTHONPATH:/home/yourhomedir/google-cloud-sdk/platform/google_appengine
export PYTHONPATH=$PYTHONPATH:/home/yourhomedir/google-cloud-sdk/platform/google_appengine/lib
export PYTHONPATH=$PYTHONPATH:/home/yourhomedir/google-cloud-sdk/platform/gsutil
export PYTHONPATH=$PYTHONPATH:/home/yourhomedir/google-cloud-sdk/platform/gsutil/gslib
export PYTHONPATH=$PYTHONPATH:/home/yourhomedir/google-cloud-sdk/lib/googlecloudsdk/lib
export PYTHONPATH=$PYTHONPATH:/home/yourhomedir/google-cloud-sdk/lib/googlecloudsdk/gcloud
export PYTHONPATH=$PYTHONPATH:/home/yourhomedir/google-cloud-sdk/lib/googlecloudsdk/bigquery
export PYTHONPATH=$PYTHONPATH:/home/yourhomedir/google-cloud-sdk/lib/googlecloudsdk/bigquery/lib
```

Next, append $PYTHONPATH to $PATH

```
export PATH=$PATH:$PYTHONPATH
```

Next, there are a number of fundamental Google Cloud SDK environment variables to add and set. First, set the CLOUDSDK_PYTHON environment variable to point to the link to Python 2.7

```
# Google Cloud SDK
export CLOUDSDK_PYTHON=/usr/bin/python2.7
```

In a later set, when the 'gcloud auth login' command is run, you will be authenticating your personal gmail account with GCP, in the process 2 events happen (for which 2 different environment variables need to be set):

1- a configuration directory is created, and

```
export CLOUDSDK_CONFIG=/home/yourhomedir/.config/gcloud
```

2 - a key file is downloaded to the local host.

When 'gcloud auth login' runs it will ask to modify your .bashrc file, so allow it to do so. The gcloud command will add the CLOUDSDK_CONFIG environment variable to .bashrc

To finish setting the credentials your personal gmail account will use when connecting to GCP, you have to add and set the GOOGLE_APPLICATION_CREDENTIALS environment variable and set it to the location of the key file that was downloaded.

```
export GOOGLE_APPLICATION_CREDENTIALS=/home/yourhomedir/initially-your-acct-key.json
```

```
export CLOUDSDK_PYTHON_SITEPACKAGES=1
```

Install GCP SDK

Pip is used to install a number of Python modules, so if the pip command is not installed then:

```
sudo easy_install pip
```

sudo pip install gcloud

that should install gcloud in /usr/lib/python2.7/site-packages/

If installing on a Google Compute Engine (GCE), the Google Cloud SDK installation directory is /usr/local/share/google/google-cloud-sdk

If not, then download GCP SDK and unzip in home directory, and run this command
./google-cloud-sdk/install.sh

You also have to install google-apps, else the 'bq' command line tool will not work.
sudo pip install google-apputils

Authenticate personal gmail with GCP

then run
gcloud auth login

Ensure that these environment variables have been added and set correctly within your .bashrc file:

Google Cloud SDK
export CLOUDSDK_PYTHON=/usr/bin/python2.7
export CLOUDSDK_CONFIG=/home/yourhomedir/.config/gcloud
export GOOGLE_APPLICATION_CREDENTIALS=/home/yourhomedir/initially-your-acct-key.json

Configure gcloud for GCP Project

Next, set the default GCP project for all gcloud scripts, so run

gcloud config set project some-gcp-project-id

Update gcloud Components

then run
gcloud components update pkg-python

then run
gcloud components update pkg-java

then run
gcloud components update pkg-go

Configure GCP Service Account with gcloud

then run to configure the service account
$ gcloud auth activate-service-account some-gcp-service-acct@developer.gserviceaccount.com --format json --key-file /tmp/some-service-acct-key.json --project some-gcp-project-id

which should return something like this:

Activated service account credentials for: [some-gcp-service-acct@developer.gserviceaccount.com]

```
{
  "MAX_TOKEN_LIFETIME_SECS": 3600,
  "NON_SERIALIZED_MEMBERS": [
    "store"
  ],
  "access_token": null,
  "access_token_expired": false,
  "assertion_type": null,
  "client_id": null,
  "client_secret": null,
  "id_token": null,
  "invalid": false,
  "refresh_token": null,
  "revoke_uri": "https://accounts.google.com/o/oauth2/revoke",
  "serialization_data": {
    "client_email": "some-gcp-service-acct@developer.gserviceaccount.com",
    "client_id": "some-gcp-service-acct.apps.googleusercontent.com",
    "private_key": "-----BEGIN PRIVATE KEY-----\n....
.....msyET7fNQ==\n-----END PRIVATE KEY-----\n",
    "private_key_id": "2.....3c",
    "type": "service_account"
  },
  "service_account_email": "some-gcp-service-acct@developer.gserviceaccount.com",
  "service_account_name": "some-gcp-service-acct@developer.gserviceaccount.com",
  "store": {},
  "token_expiry": null,
  "token_response": null,
  "token_uri": "https://accounts.google.com/o/oauth2/token",
  "user_agent": "Cloud SDK Command Line Tool"
}
```


Next, you will check to confirm that the GCP Service Account is 'active' and is therefore the default credentials that will be provided to gcloud Python scripts, have been downloaded into the $HOME/Downloads directory. Rename that JSON credentials file to something like some-service-acct-key.json, and place a copy of it within $HOME

Next, check that the service account is the default, or active, account within gcloud.

```
$ gcloud auth list
Credentialed accounts:
 - some-gcp-service-acct@developer.gserviceaccount.com (active)
 - youracct@gmail.com

To set the active account, run:
  $ gcloud config set account ``ACCOUNT"
```

Next, go back into your .bashrc file and set the GOOGLE_APPLICATION_CREDENTIALS environment variable to the key (JSON file) used by the service account. The default behavior of a gcloud-python application is to reference this environment variable.

```
export GOOGLE_APPLICATION_CREDENTIALS=/home/yourhomedir/some-service-acct-key.json
```

gcloud storage Query test

Next, validate the GCP SDK gcloud-python configuration(s). It is critically important to note that the literal data value assigned to the datastore dataset_id is an exact literal match to the project id, in this case 'some-gcp-project-id'. If the 2 values are not an exact match the query will throw a ServiceUnavailable error, and return a stack trace.

```
$ python
Python 2.7.8 (default, Apr 15 2015, 09:26:43)
[GCC 4.9.2 20150212 (Red Hat 4.9.2-6)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from gcloud import datastore
>>> datastore.set_default_dataset_id('some-gcp-project-id')
>>> datastore_conn = datastore.get_connection()
>>> datastore.set_default_connection(datastore_conn)
>>> q=datastore.Query('foo')
>>> print list(q.fetch())
```

If everything is configured correctly, the response will be:

```
[]
```

Now, all we have accomplished so far is installing the Python modules that enable us to into act with Google Cloud Storage buckets and with Google Cloud Datastore. If you want to connect to Google BigQuery there are many more tasks to accomplish.

~~$ pip install pyopenssl ndg-httpsclient pyasn1~~