My objective is to run gsutil commands from python script, using .boto credentials to authorize my request.  
  
1. **What is "gs\_access\_key\_id" and "gs\_secret\_access\_key"?**

From the GCP API Manager, under Credentials tab, I generated a OAuth2.0 client ID and client secret pair.

Is this the same as gs\_access\_key\_id" and "gs\_secret\_access\_key"?

When I run the command "gsutil config -a" , I am prompted to enter gs\_access\_key\_id" and "gs\_secret\_access\_key", which go and get automatically saved under the [Credentials] section .boto file.

What should I enter? Please tell me how this method of authorization works.

What exactly is the difference between (gs\_access\_key\_id, gs\_secret\_access\_key) and (client ID and client secret)? And what to use when?

**2. Authorize using boto Python library**

I have also set the client ID and client secret under [OAuth2] section using the same OAuth2 key pair as above:

client\_id =

client\_secret =

Python code to authorize

from boto import storage\_uri  
from gcs\_oauth2\_boto\_plugin import oauth2\_plugin  
import gcs\_oauth2\_boto\_plugin  
content = storage\_uri('gs://mybucket').get\_contents\_as\_string()  
print(content)  
  
I am getting a GSResponseError: 403 Access denied to............  error

I am following the examples from

<https://cloud.google.com/storage/docs/xml-api/gspythonlibrary?hl=en>

**3. Use of ACL bucket permissions**

If I am already using authorizing myself using these credentials, would I still need to explicitly grant ACL permissions on buckets?

PS: I am facing these authorization issues only while using .boto config file. In both the scenarios below, I am getting a 200 OK success response.

**Scenario 1: Using Service Account Credentials**

# --------------- Authorize using service account --------------------

client\_email = 'newbucket@datalab1-159607.iam.gserviceaccount.com'

private\_key\_password = 'notasecret'

scopes = ['https://www.googleapis.com/auth/cloud-platform']

credentials = ServiceAccountCredentials.from\_p12\_keyfile(client\_email, 'C:\Users\dell\Downloads\keyfile.p12', private\_key\_password, scopes)

http\_auth = credentials.authorize(Http())

service = build('storage', 'v1', http\_auth)

request = service.buckets().get(bucket='lavabucket1')

**Scenario 2: Using OAuth2.0 client credentials**

#--------------------- Authorize using OAuth2 credentials --------------------------------

CLIENT\_SECRET\_FILE = 'C:\Users\dell\Downloads\client\_secret.json'

SCOPES='https://www.googleapis.com/auth/cloud-platform'

APPLICATION\_NAME = 'Lava Client Other'

credential\_path = 'C:\Users\dell\Downloads\start.json'

store = Storage(credential\_path)

credentials = store.get()

if not credentials or credentials.invalid:

flow = client.flow\_from\_clientsecrets(CLIENT\_SECRET\_FILE, SCOPES)

flow.user\_agent = APPLICATION\_NAME

if flags:

credentials = tools.run\_flow(flow, store, flags)

else: # Needed only for compatibility with Python 2.6

credentials = tools.run(flow, store)

print('Storing credentials to ' + credential\_path)

http = credentials.authorize(httplib2.Http())

service = build('storage', 'v1', http)

request = service.buckets().get(bucket='lavabucket1')