Assignment 1 - Introduction to cloud storage and Google cloud

Name: Shweta Pathak

UTA id: 1001154572

Net id: ssp4572

Section: 13:00 to 15:00

Functionality:

1. Upload file on Google cloud.

2. Download a file from Google cloud.

3. List of files on cloud.

4. Delete a file from cloud.

# Assignment 1 - Introduction to cloud storage and Google cloud

# References:

# Assignment1Prototype.py (Code prototype provided by professor in class)

# http://stackoverflow.com/questions/20852664/python-pycrypto-encrypt-decrypt-text-files-with-aes4

# https://cloud.google.com/storage/docs/json\_api/v1/json-api-python-samples for listing objects in bucket

# http://oblalex.blogspot.com/2014/09/google-drive-api-upload-files-to-folder.html

#import statements.

import argparse

import httplib2

import os

import sys

import json

import time

import datetime

import io

import hashlib

from apiclient import discovery

from oauth2client import file

from oauth2client import client

from oauth2client import tools

from apiclient.http import MediaIoBaseDownload

from Crypto import Random

from Crypto.Cipher import AES

password = raw\_input("Enter the password :\n")

key = hashlib.sha256(password).digest()

def pad(s):

return s + b"\0" \* (AES.block\_size - len(s) % AES.block\_size)

# Encrypting file data

def encrypt(message, key, key\_size=256):

message = pad(message)

iv = Random.new().read(AES.block\_size)

cipher = AES.new(key, AES.MODE\_CBC, iv)

return iv + cipher.encrypt(message)

#Decrypting file data

def decrypt(ciphertext, key):

iv = ciphertext[:AES.block\_size]

cipher = AES.new(key, AES.MODE\_CBC, iv)

plaintext = cipher.decrypt(ciphertext[AES.block\_size:])

return plaintext.rstrip(b"\0")

#Function to encrypt a given file

def encrypt\_file(file\_name, key):

with open(file\_name, 'rb') as new:

plaintext = new.read()

encFile = encrypt(plaintext, key)

with open("encFile" + file\_name, 'wb') as new\_file:

new\_file.write(encFile)

return "encFile" + file\_name

#Function to decrypt a given file.

def decrypt\_file(file\_name, key):

with open(file\_name, 'rb') as fo:

ciphertext = fo.read()

dec = decrypt(ciphertext, key)

with open(file\_name[:-4], 'wb') as fo:

fo.write(dec)

\_BUCKET\_NAME = 'assg1\_bucket'

\_API\_VERSION = 'v1'

parser = argparse.ArgumentParser(

description=\_\_doc\_\_,

formatter\_class=argparse.RawDescriptionHelpFormatter,

parents=[tools.argparser])

CLIENT\_SECRETS = os.path.join(os.path.dirname(\_\_file\_\_), 'client\_secret.json')

FLOW = client.flow\_from\_clientsecrets(CLIENT\_SECRETS,

scope=[

'https://www.googleapis.com/auth/devstorage.full\_control',

'https://www.googleapis.com/auth/devstorage.read\_only',

'https://www.googleapis.com/auth/devstorage.read\_write',

],

message=tools.message\_if\_missing(CLIENT\_SECRETS))

# Download the given file and decrypt and saves to local machine. Also deletes the file from bucket

def get(service):

downloadfile = raw\_input("Enter file name to download with file type \n")

try:

request = service.objects().get(

bucket=\_BUCKET\_NAME,

object=downloadfile,

fields='bucket,name,metadata(my-key)',

)

response = request.execute()

print json.dumps(response, indent=2)

request = service.objects().get\_media(

bucket=\_BUCKET\_NAME ,

object=downloadfile,

)

fh = io.BytesIO()

downloader = MediaIoBaseDownload(fh, request, chunksize=1024\*1024) #show progress at download

done = False

while not done:

status, done = downloader.next\_chunk()

if status:

print 'Download %d%%.' % int(status.progress() \* 100)

print 'Download Complete!'

dec = decrypt(fh.getvalue(),key)

with open(downloadfile, 'wb') as dfile:

dfile.write(dec)

print json.dumps(response, indent=2)

except Exception as e:

print "\n\nFile not found"

except client.AccessTokenRefreshError:

print ("Error in the credentials")

#Puts a object into file after encryption and deletes the object from the local PC.

def put(service):

try:

fileToEncrypt = raw\_input("Enter file name to upload \n")

encFile = encrypt\_file(fileToEncrypt, key)

request = service.objects().insert(

bucket=\_BUCKET\_NAME,

name=fileToEncrypt,

media\_body=encFile)

response = request.execute()

os.remove(fileToEncrypt) #to remove the local copies

os.remove(encFile)

fields\_to\_return = 'nextPageToken,items(bucket,name,metadata(my-key))'

print json.dumps(response, indent=2)

while request is not None:

response = request.execute()

print json.dumps(response, indent=2)

request = service.objects().list\_next(request, response)

except Exception as e:

print "\nFile not found in local machine."

except client.AccessTokenRefreshError:

print ("Error in the credentials")

# Objects in the bucket will be displayed

def list\_objects(service):

try:

fields\_to\_return = 'nextPageToken,items(name,size,contentType,metadata(my-key))'

request = service.objects().list(bucket=\_BUCKET\_NAME, fields=fields\_to\_return)

while request is not None:

response = request.execute()

print json.dumps(response, indent=2)

request = service.objects().list\_next(request, response)

except Exception as e:

print "\n\n No files present."

#This deletes the object from the bucket

def delete\_objects(service):

objectToDelete = raw\_input("Enter object name to delete with extension \n")

try:

service.objects().delete(

bucket=\_BUCKET\_NAME,

object=objectToDelete).execute()

print objectToDelete+" deleted"

except Exception as e:

print "File not found "

except client.AccessTokenRefreshError:

print ("Error in the credentials")

def main(argv):

flags = parser.parse\_args(argv[1:])

storage = file.Storage('sample.dat')

credentials = storage.get()

if credentials is None or credentials.invalid:

credentials = tools.run\_flow(FLOW, storage, flags)

http = httplib2.Http()

http = credentials.authorize(http)

service = discovery.build('storage', \_API\_VERSION, http=http)

options\_toselect = {1: put, 2: get, 3:list\_objects, 4:delete\_objects}

while(True):

print "1. Upload file on google cloud \n"

print "2. Download a file from google cloud \n"

print "3. List of files on cloud \n"

print "4. Delete a file from cloud \n"

print "5. Exit \n"

option = raw\_input("Select one option : ")

if option =="1":

options\_toselect[1](service)

elif option =="2":

options\_toselect[2](service)

elif option =="3":

options\_toselect[3](service)

elif option =="4":

options\_toselect[4](service)

elif option =="5":

sys.exit(0)

else:

print "Please select a valid choice !!!\n"

if \_\_name\_\_ == '\_\_main\_\_':

main(sys.argv)