

Metropolitan State University

ICS 432 - 01: Distributed and Cloud Computing

Fall 2021

Lab 07: Programmatic Access to AWS and GCP using CLI & SDK

Total points: 25

Out: Saturday, October 16, 2021

Due: 11:59 PM on Friday, October 22, 2021

What to submit?

The objective of this lab is to practice using virtual machines on the cloud. To complete this lab:

- Read this lab assignment carefully.
- At various parts of the lab, you are asked to **take screen shots** of your work. Open a word document and paste the screen shots in this document in the same order as mentioned in the lab. Make sure to highlight the screen shot number.
- After you complete all the lab exercises, upload the word document to the designated D2L folder by 11:59 PM on Friday, October 22, 2021.

NOTE: On Windows machines, you may consider using [Snip & Sketch](#) for screenshot handling.

Exercise 1: Programmatically working with AWS S3 using CLI & SDK

Introduction

AWS Cloud9 is a web-based integrated development environment (IDE) that contains a collection of tools that you use to code, build, run, test, debug, and release software in the cloud. The AWS Cloud9 IDE provides you with a console to work from and filesystem to use. An Amazon Elastic Compute Cloud (Amazon EC2) instance is the driving force behind this AWS Cloud9 environment. However, this underlying system is hidden from you behind the AWS Cloud9 console, allowing you to focus on interacting with your code and command line, and the respective AWS resources, for example, Amazon S3 and Amazon EC2.

References: you may refer to the following:

- AWS Cloud9 Developer Guide for help with Cloud9:
<https://docs.aws.amazon.com/cloud9/latest/user-guide/aws-cloud9-ug.pdf#tutorial-tour-ide>
- Python boto3 or working with AWS S3:
https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3.html#S3.Client.create_bucket

Objective

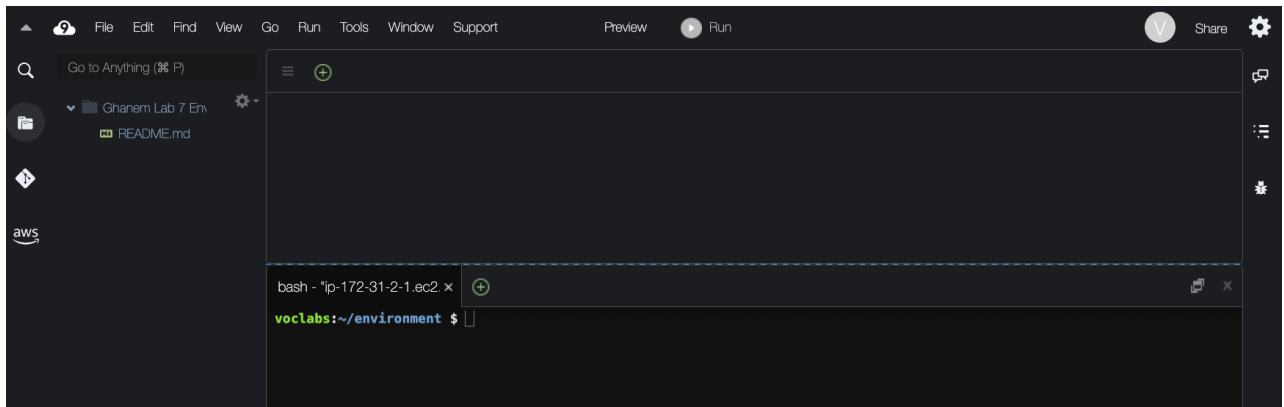
In this exercise, you will use AWS Cloud9 to write Python code host a static website on AWS S3 and to create/list with EC2 instances. This exercise consists of the following steps:

- 1- Setting up a Cloud9 Environment to be used for writing Python code.
- 2- Running simple Python code in Cloud9 Environment.
- 3- Installing the AWS SDK for Python, Boto3
- 4- Write Python code to create an S3 bucket and upload files
- 5- Making the web site publicly accessible.
- 6- Working with EC2 programmatically
- 7- Challenge: Write a python program to upload the contents of a specified directory to an AWS S3 bucket

Step 1: Setting up an AWS Cloud9 Environment

- 1- Log in to the AWS Management Console using your AWS Educate account.
- 2- Click on Services→ Cloud9.
- 3- From the Cloud9 console, click Create environment.
- 4- On the Name environment page, enter **your-last-name Lab7 Environment** and enter a description for the environment (e.g., Using Python to work with AWS resources). Click Next Step. On the Configure settings page:
 - a. **Environment type** → Create a new EC2 instance for environment (direct access).
 - b. **Instance type** → `t2.micro`.
 - c. **Platform** → Amazon Linux 2.
 - d. **Cost-saving** setting → After 30 minutes (default)
 - i. This sets the hibernation time for your environment. This setting will shut down the EC2 instance after a set period of no activity. This feature is useful because it helps you save money when you're not using your environment. You can set this to 30 minutes (which is the default), 1 hour, 4 hours, 1 day, 1 week, or never. In this demo, leave it set to the default time.
 - e. **IAM role** → keep the default AWSServiceRoleforAWSCloud9
 - i. provide an IAM role if you want Cloud9 to call other AWS services on your behalf.
 - f. **Network settings (advanced)** → keep the default.
 - i. this is where you would specify an Amazon VPC and subnet that you want to launch your EC2 instance in. However, for this lab, we will use the default network.
 - g. Click Next step.
- 5- On the Review page, review the Environment name and settings for your Cloud9 environment, and then click Create environment.
- 6- An EC2 instance is instantiated with the Cloud9 environment. It might take a few minutes to launch, and then the environment should be ready to go. If you receive a project setting file update warning, click Accept.

- 7- Take a few minutes to read the AWS Toolkit – Quick Start tutorial that introduces you to the AWS Cloud9 development then closed the tutorial. Close the Welcome Screen. You should then see the following Cloud9 IDE.



- 8- Notice the following components of the IDE:
- Menu bar (File, Edit, Find, etc.)
 - Editor, tabs, and panes
 - Command window (voclabs:~/environment)
 - File system at the left side of the IDE (including a folder with your environment name).
- 9- To see the full pathname of your current working directory, in the command window, run the following Print Working Directory command and notice the current working directory.

```
pwd
```

- 10- To check that you can access your AWS resources from AWS Cloud9, run the following command from the AWS Cloud9 terminal. You should see a list of your S3 buckets. If you do not have any buckets on your AWS S3 account, go to S3 dashboard and create a bucket and then run the command again.

```
aws s3 ls
```

- 11- Create a Text file: From the **File** System menu, right click in your environment's folder and choose **New File**. In the **Filename** box, enter `yourlastname_hello_world.txt`. Double click on the file name to open the file some text to the file and click Save.
- 12- In the command window, type `ls` and you should see your filed listed.

Lab screenshot #1: take a screenshot to show your Cloud9 IDE showing the text file and the command window.

Step 2: Running simple Python code in Cloud9 Environment

- 1- Confirm Python is installed in the environment by running the following command in the Terminal window. The output should include the version of python.

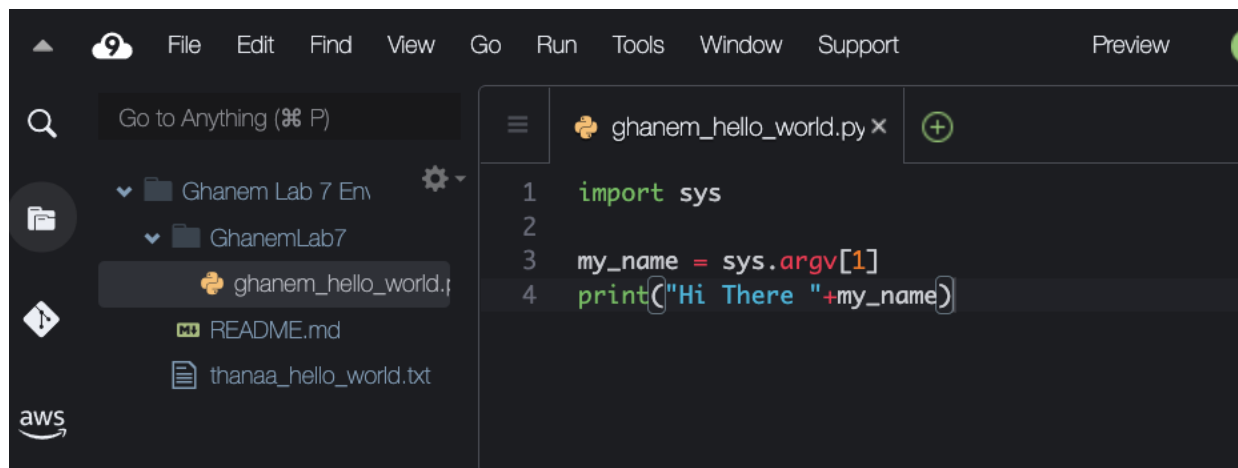
```
python --version
```

Lab screenshot #2: take a screenshot to show the command window showing python version.

- 2- In Terminal window, run the following `yum` command to help ensure the latest security updates and bug fixes are installed.

```
sudo yum -y update
```

- 3- In the file system menu on the left, right click on the environment's folder and create a new folder called `your_last_nameLab7`. Create a file called `your_last_name_hello_world.py` inside that folder.
- 4- Write the following python code in the file and save it. Note that I want you to write the code instead of just copying it so that you pay attention to the syntax.



- 5- To set the AWS Cloud9 terminal path to your folder, in the command window, change directory to go inside your Lab 7 directory. For example, my command will be as follows:

```
cd GhanemLab7
```

- 6- From the AWS Cloud9 terminal, run your python code (using `python3`) using the following run command. Note that the run command requires you to pass in a variable, that is, your name. For example, my command will be as follows:

```
python3 ghanem_hello_world.py Thanaa
```

Lab screenshot #3: take a screenshot to show the command window with the output of your program

Step 3: Installing the AWS SDK for Python, Boto3

- 1- Type `clear` in your command window to delete all the previous commands.
- 2- You will install pip and Boto3 by running the following command

```
sudo python3 -m pip install boto3
```

- 3- Verify boto3 is installed and show the current version

```
python -m pip show boto3
```

Lab screenshot #4: take a screenshot to show the command window after running the previous two commands.

- 4- Take few minutes to explore the following link for the boto3 library commands.

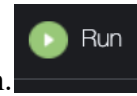
https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3.html#S3.Client.create_bucket

Step 4: Write Python code to create a bucket on S3 and upload files

- 1- Download and unzip the following file from D2L: `lab-07-files.zip`. Unzip the file to a directory.
- 2- On Cloud9 File System, select your Lab7 folder then click on **File→Upload Local Files**. Upload the folder you downloaded from D2L that includes the following three files: `index.html`, `logo.jog`, and `index.jpg` files. These are the same ICS432 web site files that were used in Lab 3.
- 3- On Cloud9 file explorer, under your Lab 7 folder, open a new file and save it with the name `create_s3_bucket.py`.
- 4- The following screen shot show Python code that can be used to create an S3 bucket and make the bucket available for web site hosting. Write this code in your `create_s3_bucket.py` file and change the bucket name to include your last name.

```
1 import boto3
2
3 s3 = boto3.client("s3", region_name="us-east-1")
4 bucket_name = "ghanemla7website"
5
6 s3.create_bucket(Bucket=bucket_name)
7
8 website_configuration = {
9     "ErrorDocument": {"Key": "error.html"},
10    "IndexDocument": {"Suffix": "index.html"}
11 }
12
13 s3.put_bucket_website(
14     Bucket = bucket_name,
15     WebsiteConfiguration = website_configuration
16 )
17
```

Lab screenshot #5: take a screenshot of your python code.



- 5- Run `create_s3_bucket.py` by clicking on the Run button.
- 6- Go to the AWS S3 dashboard and make sure that the bucket is created.

Lab screenshot #6: take a screenshot of your S3 dashboard showing the newly created bucket.

- 7- Create another file called `upload_files.py` and write the following code to upload the `index.jpg` file to the buckets. Add code to upload the other two files as well. Note that the content type for images should be `image/jpg`



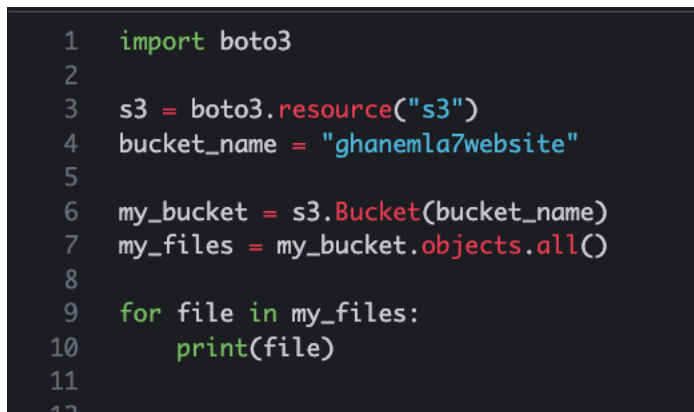
```
1 import boto3
2
3 s3 = boto3.client("s3", region_name="us-east-1")
4 bucket_name = "ghanemla7website"
5
6
7
8 file_name_1 = '/home/ec2-user/environment/GhanemLab7/s3-website-lab-files/index.html'
9 response = s3.upload_file(file_name_1, bucket_name, "index.html", ExtraArgs={"ContentType": "text/html"})
```

Lab screenshot #7: take a screenshot of your python code.

- 8- Run the `upload_files.py` code. Open S3 console and make sure that the three files are uploaded to the bucket.

Lab screenshot #8: take a screenshot of your S3 bucket dashboard.

- 9- Create a new python file called `list_bucket_objects.py`. Write code in the file to print the names of all objects (i.e., files) in the bucket.



```
1 import boto3
2
3 s3 = boto3.resource("s3")
4 bucket_name = "ghanemla7website"
5
6 my_bucket = s3.Bucket(bucket_name)
7 my_files = my_bucket.objects.all()
8
9 for file in my_files:
10     print(file)
11
12
```

Lab screenshot #9: take a screenshot of your code.

Lab screenshot #10: run your code and take a screenshot of the command window showing the output.

Note: `boto3.client("s3")` and `boto3.resource("s3")` are two different libraries that allows you to work with AWS S3 buckets and each library supports a different set of methods.

10 – You can also list all buckets and objects in a bucket using AWS CLI by running the following command in the command window.

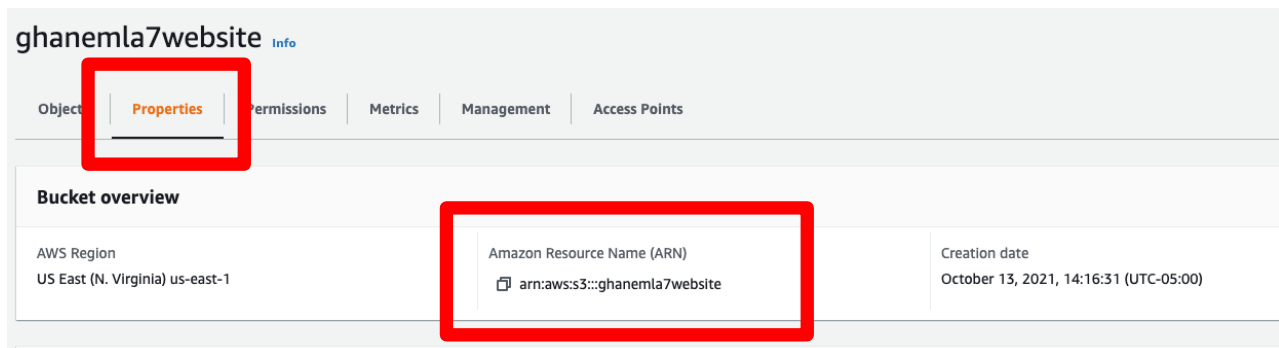
```
aws s3api list-objects --bucket "ghanemla7website"
```

Note the output is in JSON format where each JSON object represents one file along with the meta data for that file.

Step 5: Making the web site publicly accessible

In this step, you will create a bucket policy that gives anyone access to your S3 bucket

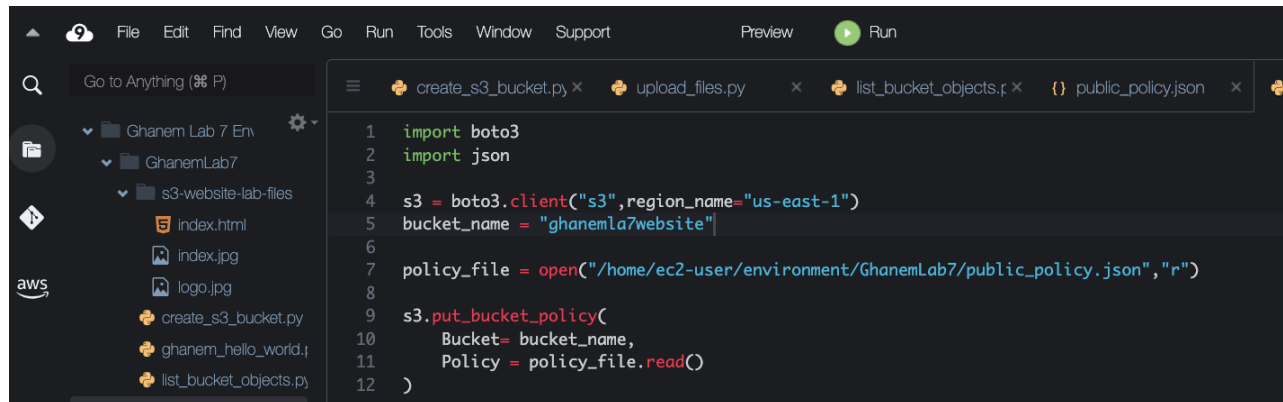
- 1- On S3's dashboard, click on index.html and find the Object URL from the Properties tab. Copy the URL and open it from the browser. You will get access denied error.
- 2- To create a bucket policy that gives anyone permission to read from your bucket, in Cloud9 file browser, create a new file in your Lab 7 folder and call it `public_policy.json`
- 3- In this file, paste the following JSON record and replace the Resource with your Bucket's **ARN** and save the file. Note ARN is Amazon Resource Number that is assigned to any resource created on AWS. You can retrieve the bucket's ARN from the bucket's Properties tab.



```
{
  "Version": "2012-10-17",
  "Statement": [ {
    "Sid": "PublicReadGetObject",
    "Effect": "Allow",
    "Principal": "*",
    "Action": [ "s3:GetObject" ],
    "Resource": [ "arn:aws:s3:::ics432-web-site/*" ]
  } ]
}
```

- 4- In your Lab7 folder, create a python file called `permissions.py` and write the following code.

Lab screenshot #11: Take a screenshot of your python code.

A screenshot of a code editor window. The left sidebar shows a file explorer with a tree view containing 'Ghanem Lab 7 Env', 'GhanemLab7', and 's3-website-lab-files'. The main editor area shows a Python file named 'permissions.py' with the following code:

```
1 import boto3
2 import json
3
4 s3 = boto3.client("s3", region_name="us-east-1")
5 bucket_name = "ghanemla7website"
6
7 policy_file = open("/home/ec2-user/environment/GhanemLab7/public_policy.json", "r")
8
9 s3.put_bucket_policy(
10     Bucket= bucket_name,
11     Policy = policy_file.read()
12 )
```

- 1- Run `permissions.py`
- 2- You should be able now to access your web site using a link similar to the following link: <http://ics432-web-site.s3-website-us-east-1.amazonaws.com>. Open a browser window and paste your URL.

Lab screenshot #12: Take a screenshot of your browser window and make sure the URL bar is shown in the URL.

Step 6: Programmatically working with EC2 instance

- 1- Use the following AWS CLI to list current EC2 instances in your account:

```
aws ec2 describe-instances
```

The output of this command is JSON records that includes all metadata about instances. You can list only instance ids using the following command

```
aws ec2 describe-instances \
    --filters Name=instance-type,Values=t2.micro \
    --query Reservations[*].Instances[*].[InstanceId] \
    --output text
```

Lab screenshot #13: take a screenshot of the command window showing the output of the two commands.

- 2- Write the following python code to create an EC2 instance and add a tag for you instance with your name attached to the tag. For example, my instance name is set to “ThanaaLab7Instance”.

```
working_with_ec2.py
2
3 ec2_client = boto3.client("ec2",region_name="us-east-1")
4 key_pair = ec2_client.create_key_pair(KeyName="lab7keypair")
5
6 name_tag = [
7     {
8         "ResourceType":"instance",
9         "Tags": [
10             {
11                 "Key": "Name",
12                 "Value": "ThanaaLab7Instance"
13             }
14         ]
15     }
16 ]
17
18
19 instances = ec2_client.run_instances(ImageId="ami-02e136e904f3da870",
20                                     MinCount=1,
21                                     MaxCount=1,
22                                     InstanceType="t2.micro",
23                                     KeyName="lab7keypair",
24                                     TagSpecifications = name_tag
25                                     )
26
27
```

Lab screenshot #14: take a screenshot of your EC2 dashboard showing your newly created instance.

- 3- Create another python file with the following code to display information about your EC2 instances.

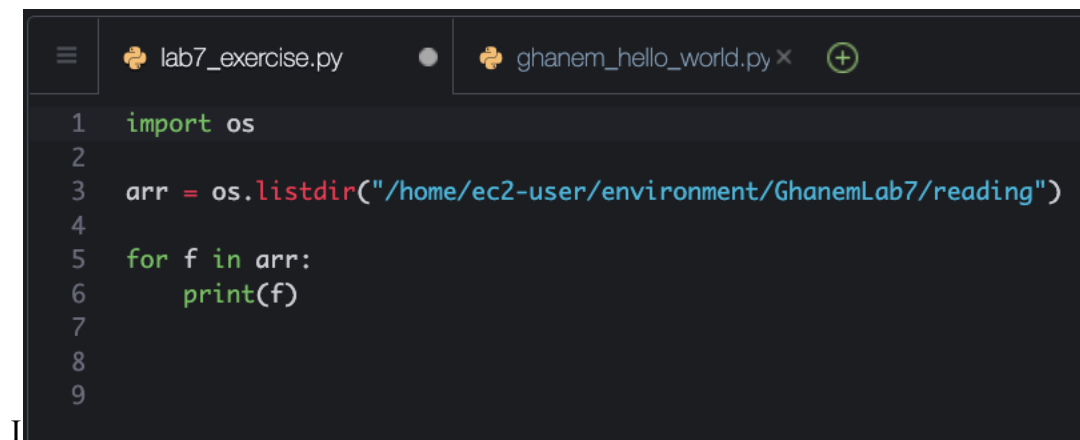
```
33 response = ec2_client.describe_instances()
34 for reservation in response["Reservations"]:
35     for instance in reservation["Instances"]:
36         print(instance["InstanceId"])
37         print(instance["LaunchTime"])
38         print(instance["ImageId"])
39         print("=====")
40
```

Lab screenshot #15: take a screenshot of the output of the python code that shows the details of your instances.

Challenge: Write a python program to upload the contents of a specified directory to an AWS S3 bucket

In this step, you will use what you learned in the previous steps to write a python program to upload the contents of a specified directory to an AWS S3 bucket. Your program should work as follows:

- a. Reads a folder name from the user.
- b. Creates an S3 bucket with the same name as the input folder name.
- c. Opens the folder and finds how many files in the folder and print a message with the number of files. To test this step, upload a folder (that contains at least four files) from your local computer to your Cloud9 environment's file system. In python, the following code is used to list all file names in a directory called 'reading'.



```
lab7_exercise.py  ghanem_hello_world.py x (+)
1  import os
2
3  arr = os.listdir("/home/ec2-user/environment/GhanemLab7/reading")
4
5  for f in arr:
6      print(f)
7
8
9
```

- d. Write a for loop that reads all the file names from the folder and upload these files to the bucket.
- e. Write code to print on the screen the name of the objects in the bucket to make sure all the files are uploaded successfully.

Lab screenshot #16: Take a screenshot of your code.

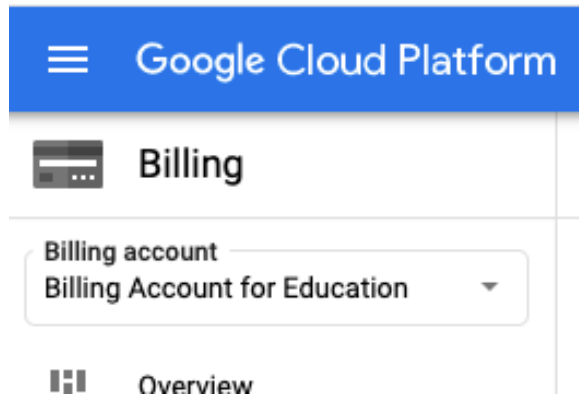
Lab screenshot #17: Take a screenshot of the command window to show the output of running your code.

Lab screenshot #18: Take a screenshot to show the contents of the newly created bucket as shown on AWS S3 dashboard.

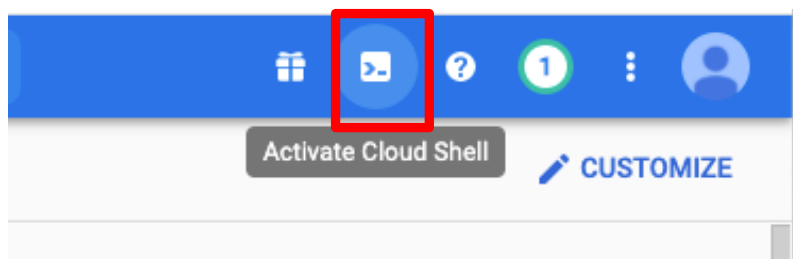
Exercise 2: Programmatic access of GCP Storage

In this exercise, you will write Python code to work with buckets on Google Cloud Storage.

- 1- Log in to GCP console using your ICS432's course credentials.
- 2- Create a new project called <your-last-name>Lab7.
- 3- From the top menu bar, make sure the new project is selected and click on Billing to make sure the 'Billing Account for Education' is selected.



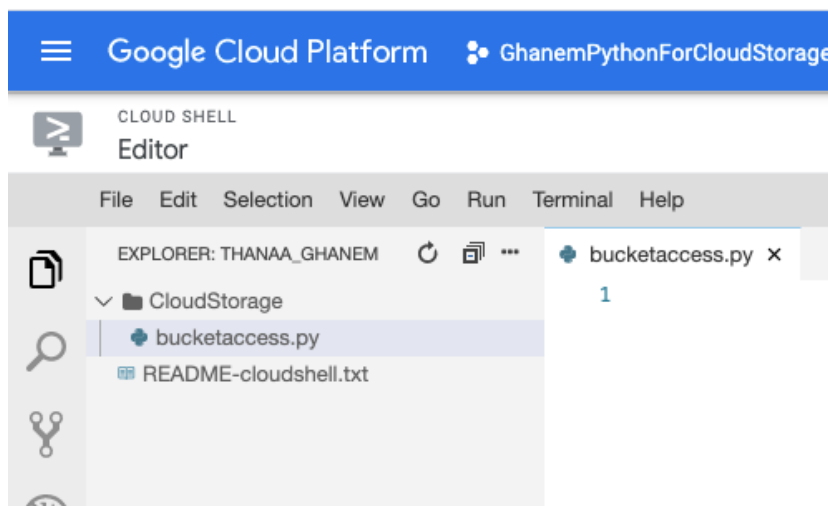
- 4- From the top right corner, click on Activate Cloud Shell.



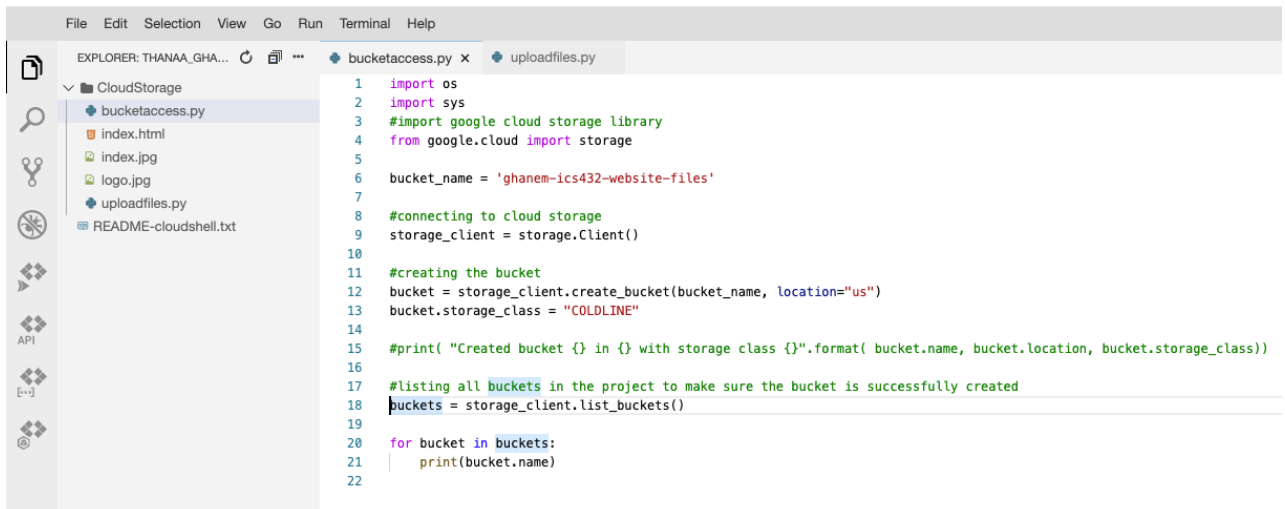
- 5- Click on Open Editor from the top menu of the cloud shell.



- 6- In the editor, click on File → New Folder and create a folder called **Your-last-nameLab7**.
- 7- Right click on the folder name and create a file called, `bucketaccess.py`.

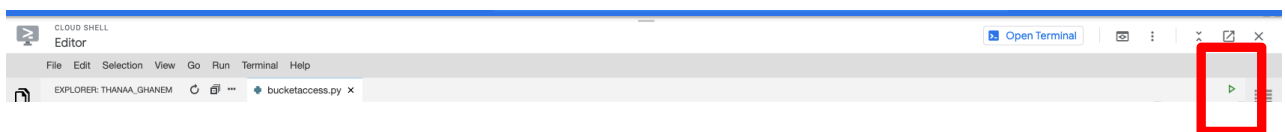


- 8- Write the following python code that does the following:
- creates a new bucket. Change the bucket name to include your last name instead of my last name.
 - lists all the buckets in your account.

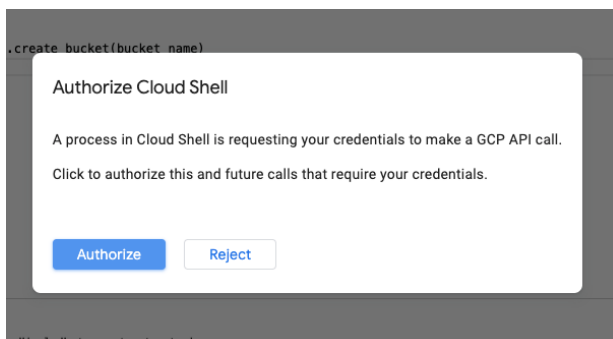


```
1 import os
2 import sys
3 #import google cloud storage library
4 from google.cloud import storage
5
6 bucket_name = 'ghanem-ics432-website-files'
7
8 #connecting to cloud storage
9 storage_client = storage.Client()
10
11 #creating the bucket
12 bucket = storage_client.create_bucket(bucket_name, location="us")
13 bucket.storage_class = "COLDLINE"
14
15 #print( "Created bucket {} in {} with storage class {}".format( bucket.name, bucket.location, bucket.storage_class))
16
17 #listing all buckets in the project to make sure the bucket is successfully created
18 buckets = storage_client.list_buckets()
19
20 for bucket in buckets:
21     print(bucket.name)
22
```

- 9- From the top right menu, click on the small green arrow to run the code.



- 10- If you receive the following message, click Authorize.

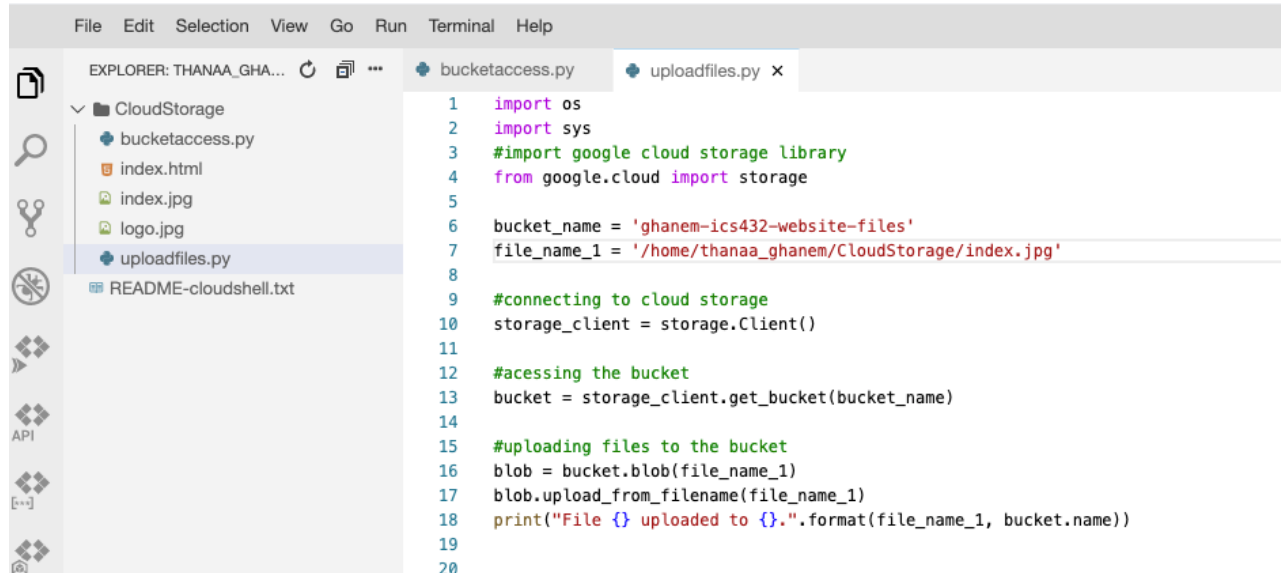


Lab screenshot #19: take a screenshot of terminal window that shows the list of buckets.
Lab screenshot #20: take a screenshot of your cloud storage dashboard with your newly created bucket listed.

- 11- Create another python file called uploadfiles.py
12- Right click on CloudStorage folder in the file browser and upload the three files:
index.html, logo.jpg, and index.html.

- 13- Write the following code in the `uploadfiles.py`. Make sure to include the correct file path in your code. Add code to upload the other two files. You can use `pwd` and `ls` in the terminal window at any time to know the correct path to the file.

```
Problems Python Python x
thanaa_ghanem@cloudshell:~ (ghanempythonforcloudstorage)$ pwd
/home/thanaa_ghanem
thanaa_ghanem@cloudshell:~ (ghanempythonforcloudstorage)$ ls
CloudStorage README-cloudshell.txt
thanaa_ghanem@cloudshell:~ (ghanempythonforcloudstorage)$ cd CloudStorage/
thanaa_ghanem@cloudshell:~/CloudStorage (ghanempythonforcloudstorage)$ ls
bucketaccess.py index.html index.jpg logo.jpg uploadfiles.py
thanaa_ghanem@cloudshell:~/CloudStorage (ghanempythonforcloudstorage)$
```



- 14- Go to cloud storage and make sure the file is correctly uploaded to the bucket.

Lab screenshot #21: take a screenshot of your cloud storage dashboard to show the three files are uploaded to the bucket.

- 15- Use the following python command to list all files in a bucket. Run the command.

Lab screenshot #22: Take a screenshot of the terminal window showing the list of files in the bucket.

- 16- Use the following python code to download a file from a bucket to the editor's file system.
To test this step, upload a file from your local computer to a bucket then download the file to the editor's file system.

```
blob = bucket.blob(source_blob_name)
blob.download_to_filename(destination_file_name)
```

Lab screenshot #23: Take a screenshot of your python file that includes the download file code.