Using Simple Storage Service (S3)

Overview:

Amazon Simple Storage Service (Amazon S3) is storage in the cloud. Data stored on S3 are accessible to those with appropriate permissions from any Internet connected device. Sending and receiving data to and from S3 can be accomplished through the AWS Management Console, AWS CLI or through the use of programming language specific Software Development Kits(SDKs).

This guide introduces you to Amazon S3 and how to use the AWS Management Console along with the Cloud9 IDE within AWS. The S3 User guide can be downloaded from AWS but is also available in the weekly content.

Be reminded Amazon updates their user interfaces frequently. You should expect some differences between the screen captures shown in this document and actual screen results while navigating the services at Amazon through the Management Console.

By default, AWS allows a user to create up to 100 buckets in each of their AWS accounts. Additional buckets may be achieved through requesting additional buckets through the AWS support team.

Bucket names should be DNS compliant. They can contain lowercase letters, numbers, hyphens and periods. Bucket names can only start and end with a letter or number, and cannot contain a period next to a hyphen or another period.

S3 through the AWS Management Console:

The AWS Management console can be used to create, delete and use S3 buckets. For a simple use case, we will create and use an S3 bucket to store family photos. Once created, we will upload some photos to an appropriate folder within S3 and then use the bucket to download images as needed.

In addition to setting up S3 buckets, permissions (policies) are needed to ensure only those users with the correct privileges will have access to the data. Even though we might not consider the photos to represent a security risk, we will make sure data on the S3 bucket is encrypted at rest.

To create an S3 bucket, login to your AWS Educate account, navigate to your assigned classroom, and click on the S3 service available in the storage category. **See the "EnteringYourAWSClassroom.pdf" document in your LEO classroom if you need a refresher on how to login and navigate to your assigned classroom. Contact your instructor immediately, if you are not able to enter your classroom.**

Figure 1 shows the results after selecting the S3 service within the AWS management console. Note, if you have previously created S3 buckets, those buckets will be displayed on the screen.

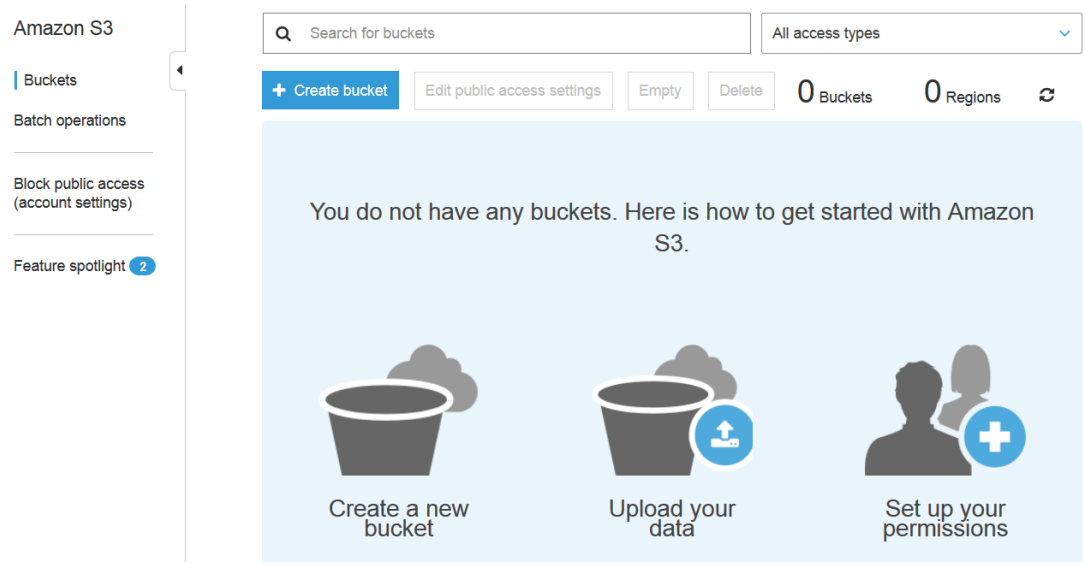

Figure 1 S3 Service Management Console View

To create a bucket, click on the Create bucket button and follow steps to name the bucket, select the region, set the properties, set the permissions, and review. For this exercise, name the bucket edu.umuc.sdev400.yourname.photos. The bucket name must be unique for all AWS environments and contain all lowercase letters. Figure 2 illustrates the successfully completing the first step in creating the bucket.

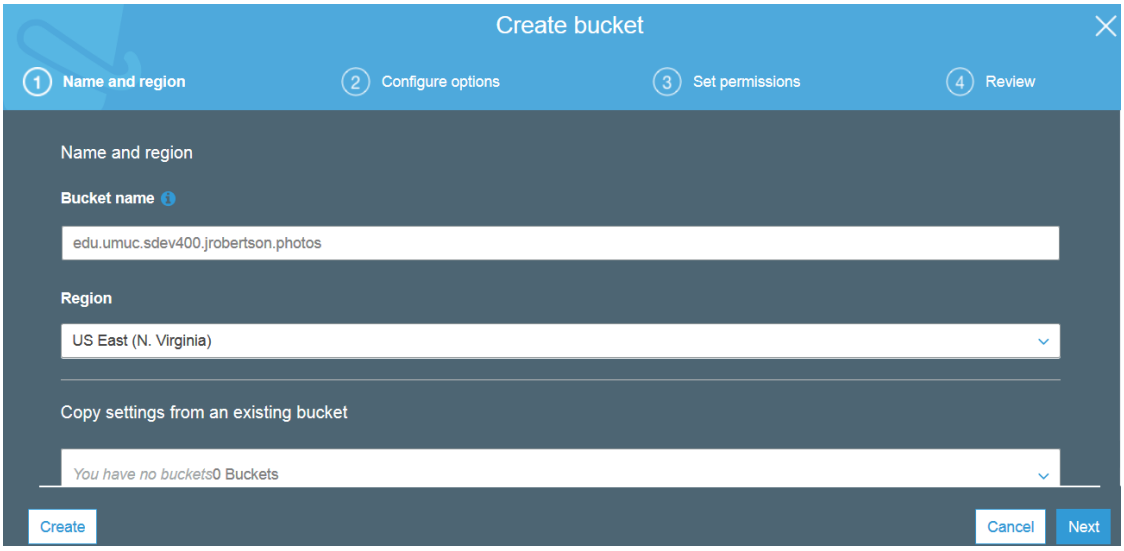

Figure 2 Step 1 to create an S3 bucket

Click next to move to the subsequent steps. For now, we will not tag the instance or adding any Cloud trail object level logging. However; scroll down to automatically encrypt objects using AES-256 Server Side encryption. Then, as shown in figure 3, click Next to continue.

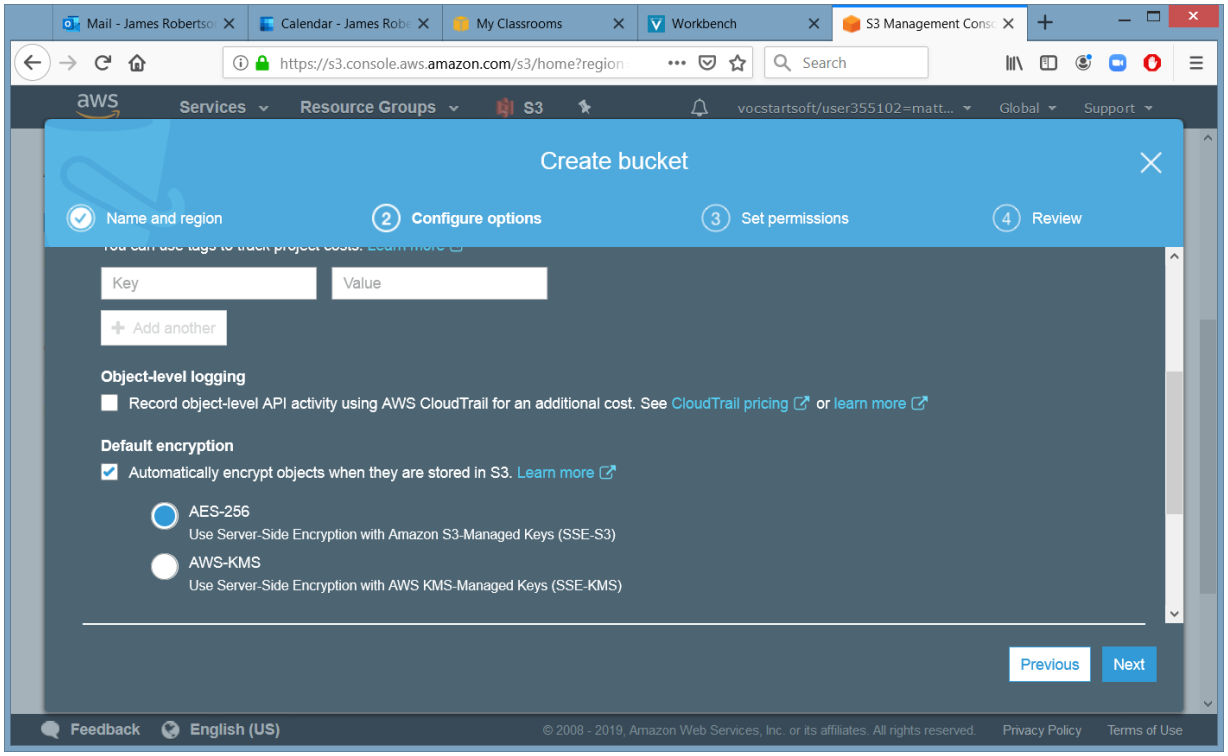

Figure 3 Step 2 in Creating S3 Instance

For step 3, accept the Block all public access and click Next as shown in Figure 4. Blocking all public access should be the default for most of your buckets to avoid accidental exposure of sensitive data to the public.

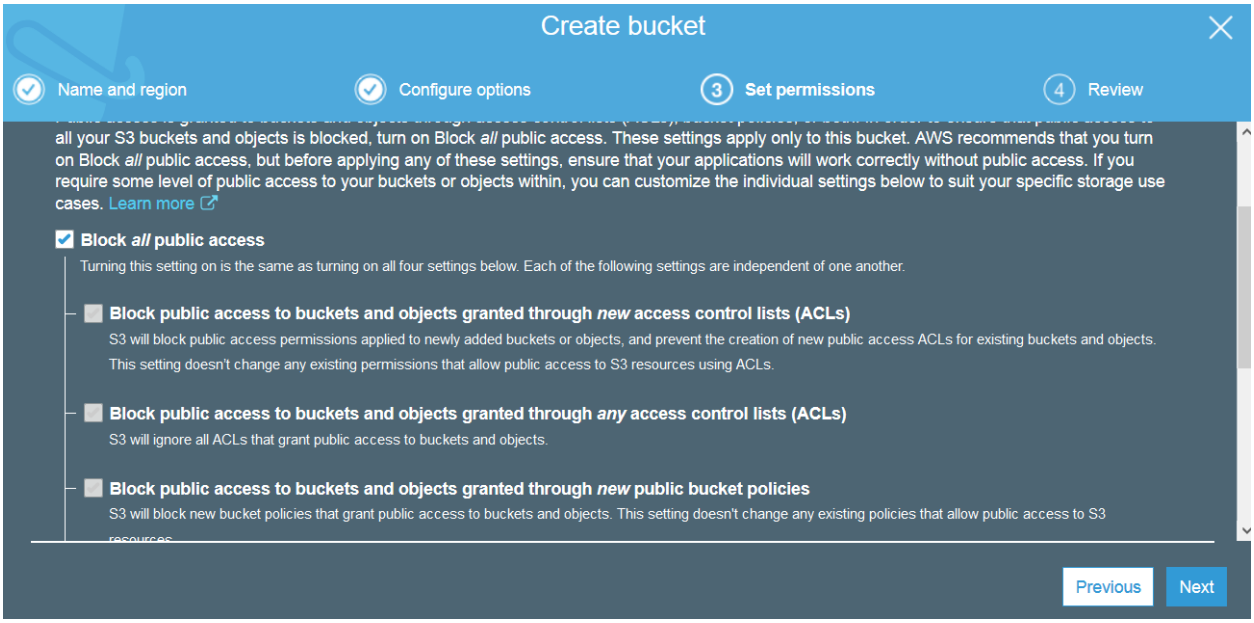

Figure 4 Step 3 for Creating an S3 Bucket

Review the bucket settings. Figure 5 shows the review screen associated with step 4. Click Create Bucket to complete the S3 bucket.

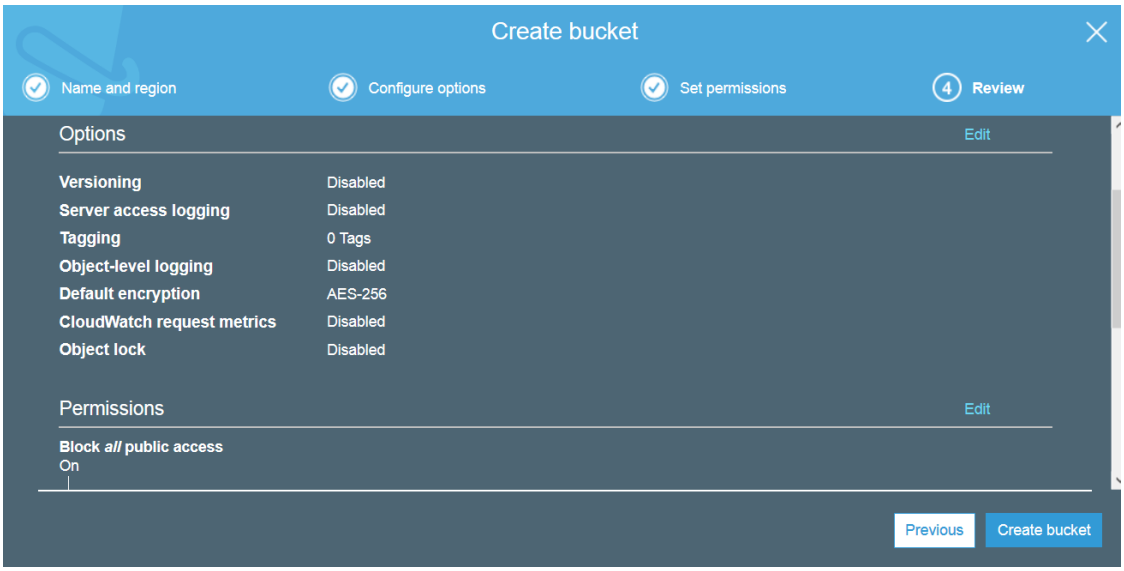

Figure 5 Step 4 for Creating an S3 Bucket

Successful creation of the S3 bucket will result in the image shown in Figure 6.

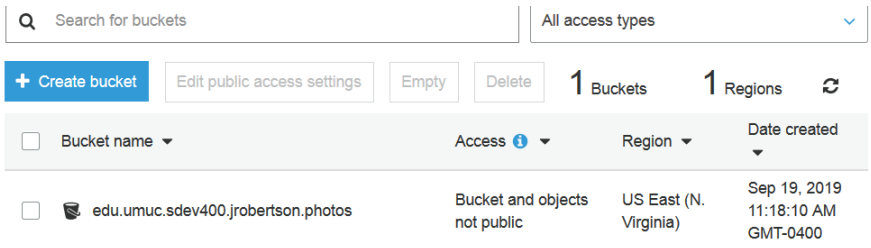

Figure 6 Successful Creation of the S3 Bucket

To create folders in the bucket click on the photos bucket and click create folder as shown in Figure 7. Folders can be thought of just like folders or directories on your computer. You can place files and documents in the folders.

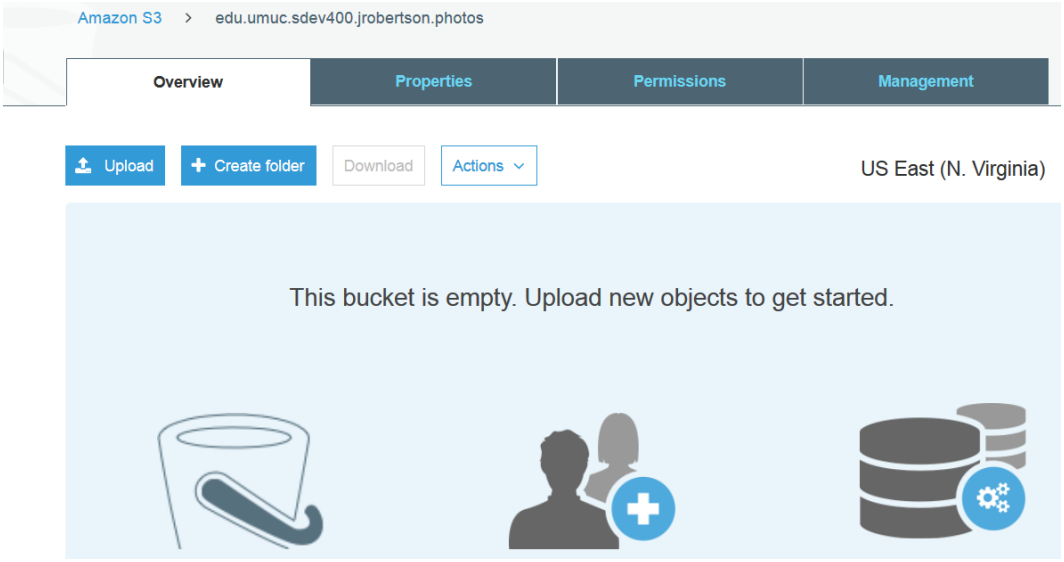

Figure 7 Adding Folders to the S3 Bucket

Create three folders to store photos for 2017, 2018, and 2019 as illustrated in Figure 8. Click Save after creating each folder. Note, if you select “None” for the encryption setting it defaults to the bucket encryption settings which we already defined as AES-256 server side encryption.

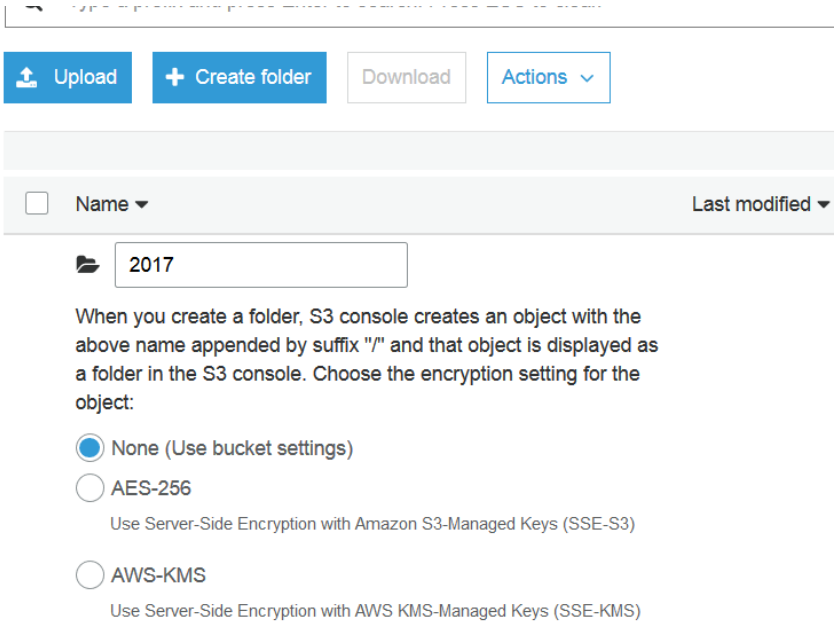

Figure 8 Creating Folders on S3

Once the folders are created you will see them directly on the AWS console as shown in figure 9.

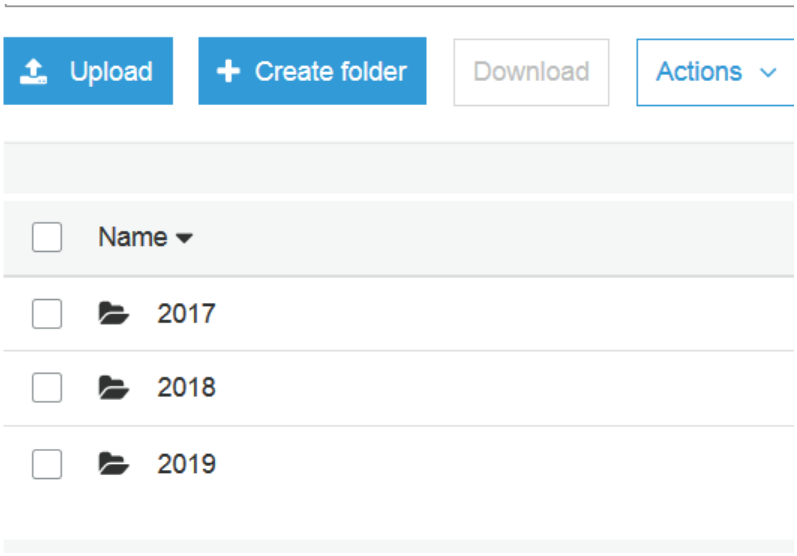

Figure 9 Creating 3 folders

To upload a photo or document from your desktop, select the folder you want to upload to in your S3 bucket, then select "Upload". You can either drop and drag to the folder or use the Add files button as shown in figure 10.

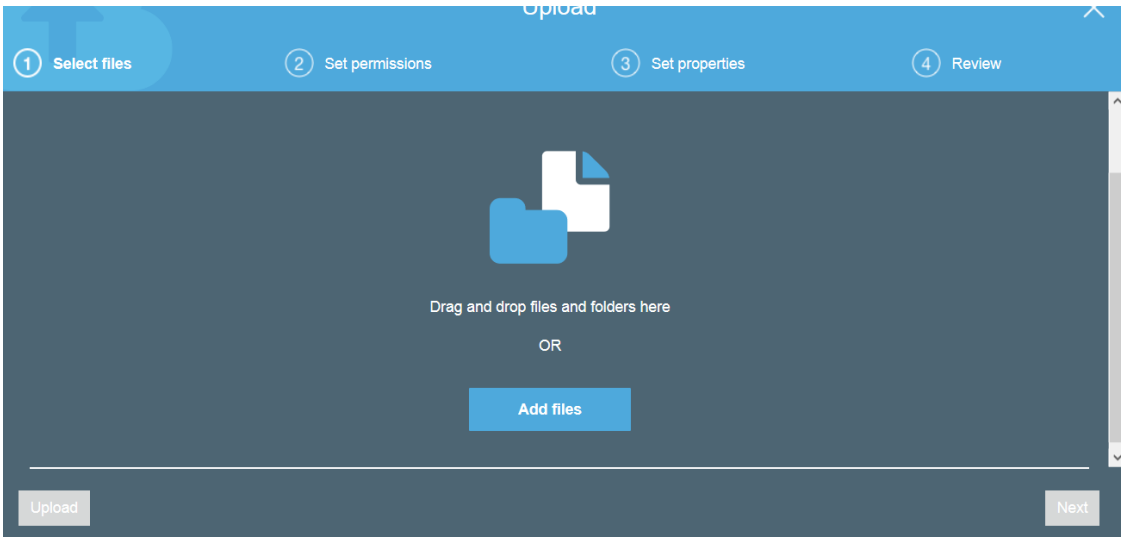

Figure 10 Uploading a File to an S3 folder

If you select Add files, you will prompted to select and file from your desktop computer for upload. Choose an appropriate file and then select Open. As shown in figure 11, select Upload to upload the file to the S3 bucket folder.

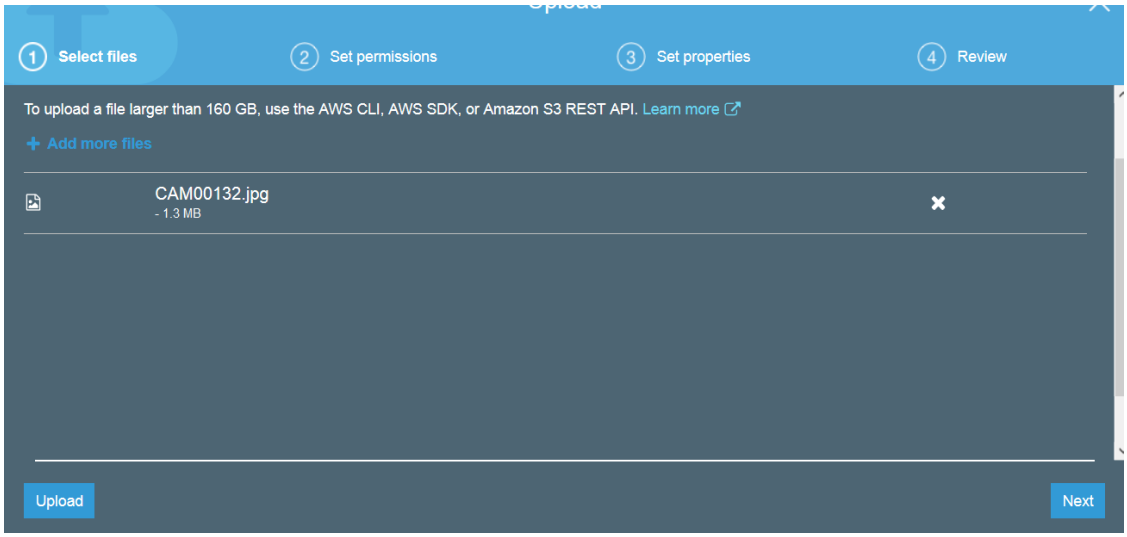

Figure 11 Upload Images to the S3 bucket

Upon successful upload, the new images will be stored in your S3 bucket in the folder you specified. Figure 12 shows the screen output resulting from three images that were successfully transferred to the 2017 folder in the S3 photos bucket.

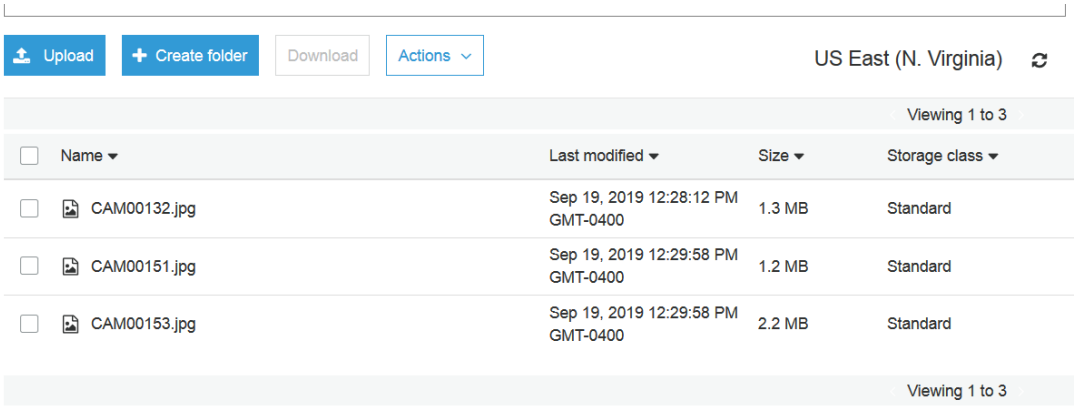

Figure 12 Successful Upload of Vacation Photos

You can download the files the AWS management console by navigating to the folder containing the file you want to download, selecting it and then clicking Download as shown in figure 13.

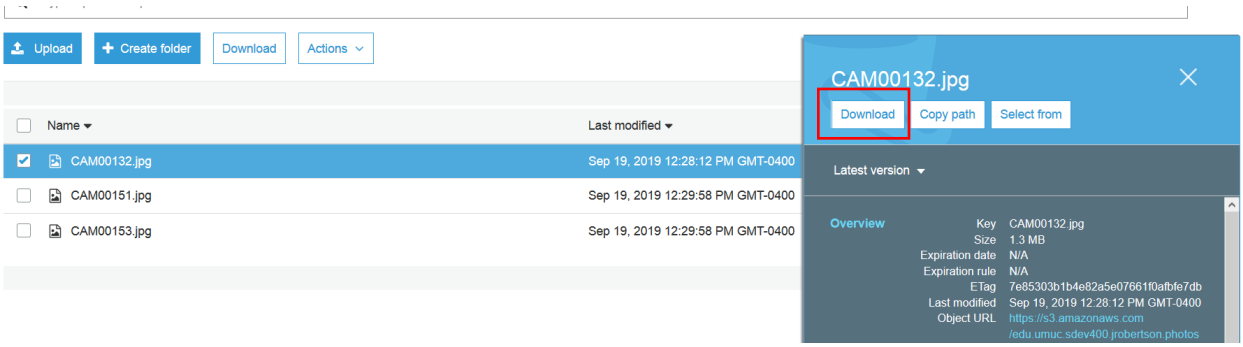

Figure 13 Downloading files from the S3 bucket folder.

If you view the file you will see that even though encryption is enabled, the results are transparent to an authorized user. As shown in figure 14, the image downloaded as expected.

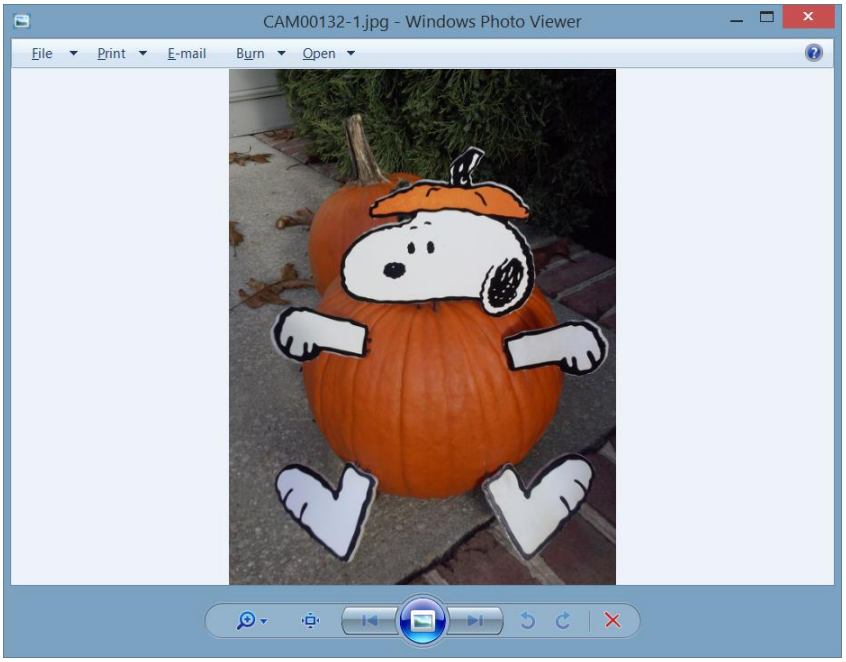

Figure 14 Successful Image Download

Server-side encryption protects your data at rest by preventing unauthorized users from being able to view the image or document. This means if someone gains access to the disks or hardware containing the encrypted documents, they will not be able to read it without the key.

S3 through the AWS SDK and Cloud 9:

If you successfully completed previous SDEV courses at UMGC, you more than likely had to use the Cloud9 IDE for creating, compiling and running code within the AWS cloud. This section shows you how to programmatically use Cloud9 and Python to interact with Amazon's S3 service.

As before, login to your AWS educate account, navigate to your assigned AWS classroom and search for/navigate to the Cloud 9 IDE service. If you already have Cloud9 IDE environment running, you can launch that one. If not, click on "Create Environment". As shown in figure 15, name and describe your environment and then click "Next step".

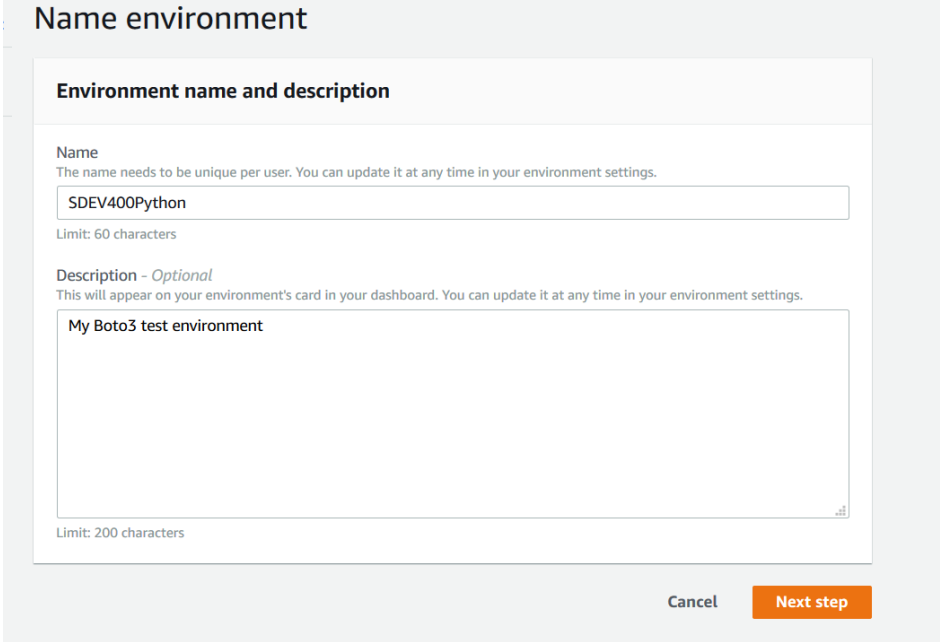

Figure 15 Name your Cloud9 IDE

For the environment, you should accept all of the default settings as shown in Figure 16. Click "Next step" to continue.

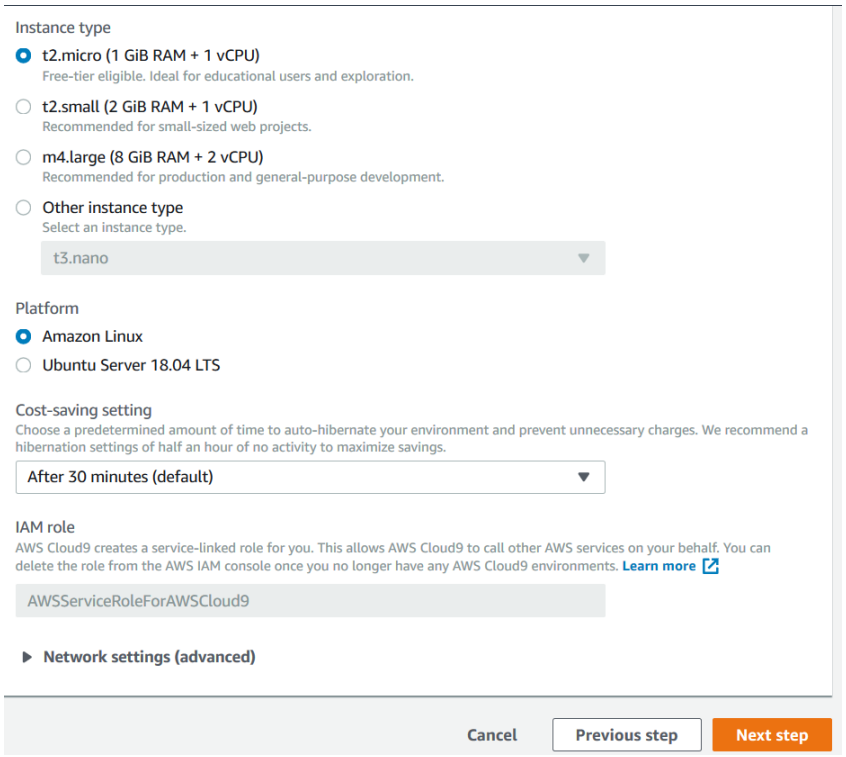

Figure 16 Accept the Default Environment Settings

Once the review settings step appears, click the "Create Environment" as shown in figure 17.

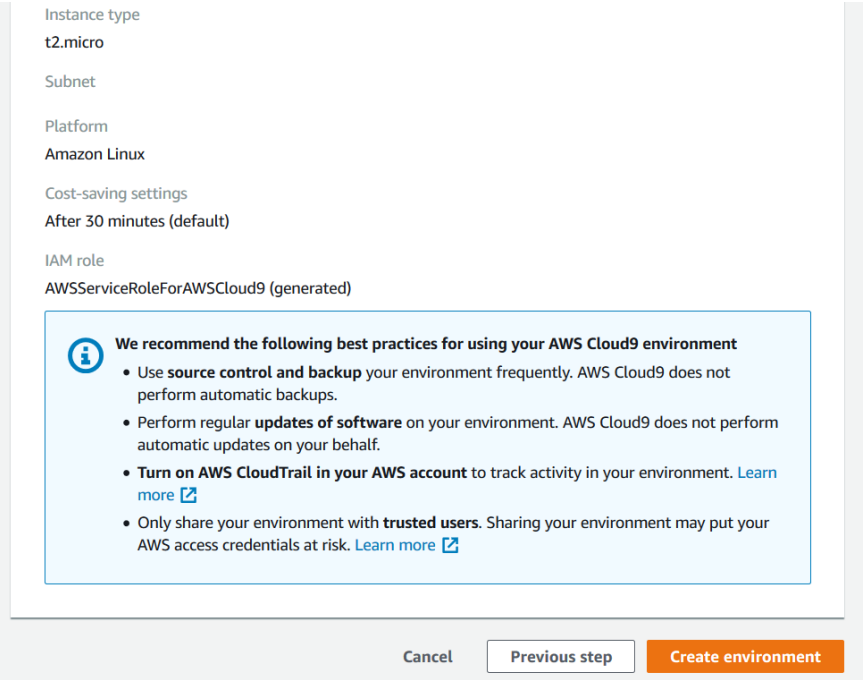

Figure 17 Review and Create Environment

It will take a few minutes for your environment to be ready. See figure 18.

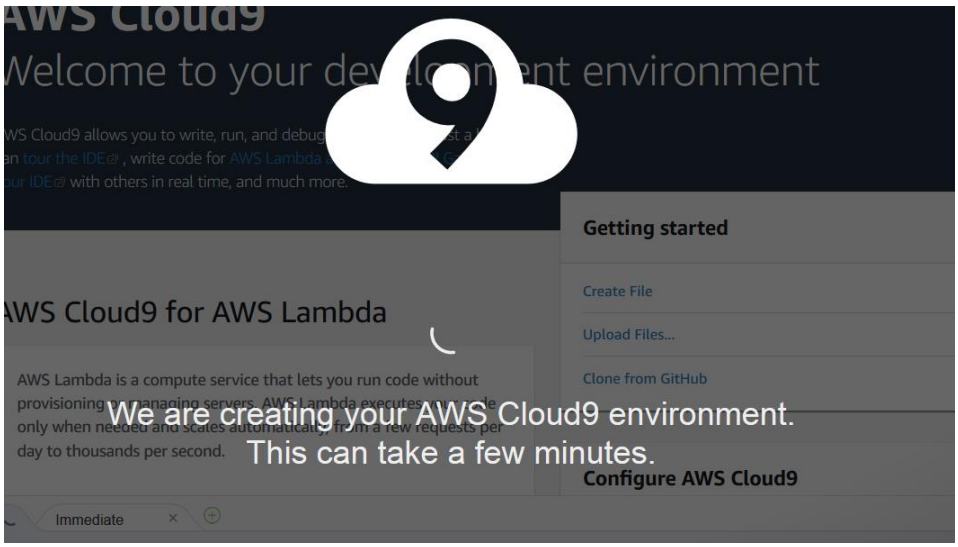

Figure 18 Waiting for the Cloud9 environment

Once the environment is ready the initial welcome screen will appear as shown in figure 19.

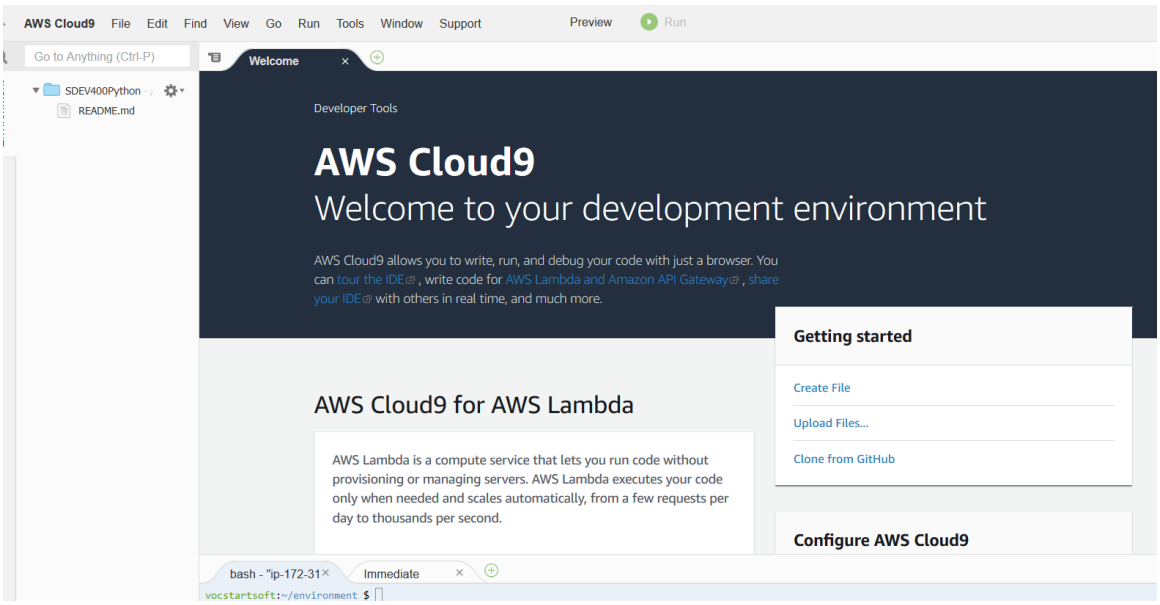

Figure 19 Cloud9 IDE Initial Welcome Screen

Any new software package will take some time to become comfortable using it. However; in most cases, you only need a few basic commands and navigation tips to become proficient. However; I encourage you to look at the IDE tour found at this URL to become comfortable with the basic features:

https://docs.aws.amazon.com/cloud9/latest/user-guide/tour-ide.html

Installing boto3 in your Cloud9 IDE

Since we will be using Python3 for most of the course, we need to set-up the environment to have available the Python modules and packages. Most functionality allowing communication with AWS services such as S3 are included in the boto3 package. To install the boto3 package, use the bash shell at the bottom of the screen and type:

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

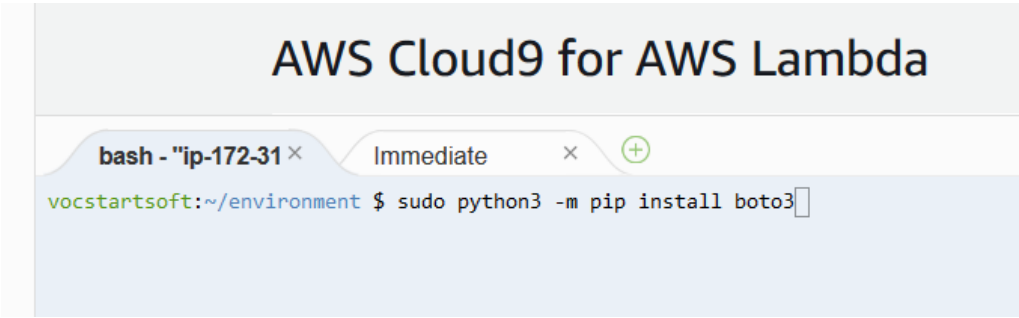

Figure 20 Installing boto3 in Cloud9

Successful results are shown in figure 21.

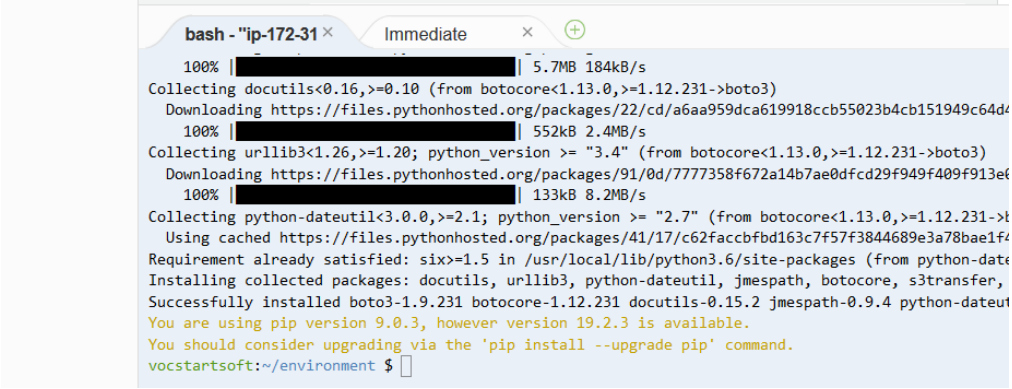

Figure 21 Successful installation of boto3

A couple of important notes:

1. Make sure you use Python3. Both Python(2) and Python3 are installed on the IDE. Python3 is much more secure. You should be using Python3 for compiling and running programs.
2. **Do not** update the pip installer on your Cloud9 environment. I know this is attempting to do since you see that nice suggestion when you install, but if you do, you will have all sorts of compatibility problems which will cause significant frustration.
3. From time to time, you may need to install other Python3 modules to support a set of functionality provided in some of the examples. You can do this the same way you installed the boto3. Just substitute the name of the needed module. For example, to install numpy, use this statement:

```
sudo python3 -m pip install numpy
```

Connecting to an S3 bucket

Now that you have boto3 installed, you can quickly create a Python file to test your environment. A good practice is to separate your projects into folders in your IDE. To create a folder named "FirstTest", click on the Main folder in the upper left side of the IDE, use your right-mouse button to select New Folder. Enter the folder name and it will be created. (See figure 22)

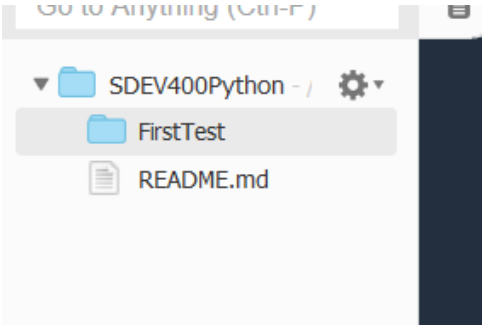

Figure 22 Creating a Project Folder

Similar to the way the folder was created, select the FirstTest folder, click the right mouse button and then select New File. Name the file ListBuckets.py as shown in figure 23.

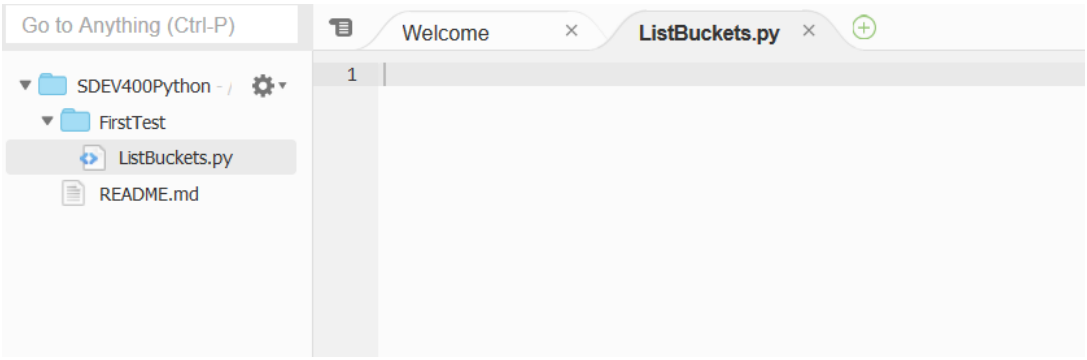

Figure 23 Creating a File

Next, we type (or copy and paste) the sample code that shows how to connect to your S3 buckets and list the names of each one.

```
import boto3

# Create an S3 client

s3 = boto3.client('s3')

# Call S3 to list current buckets

response = s3.list_buckets()

# Get a list of all bucket names from the response

buckets = [bucket['Name'] for bucket in response['Buckets']]

# Print out the bucket list

print("Bucket List: %s" % buckets)
```

As with any coding project, save early and save often. You can save in the Cloud9 IDE, by using CTRL+S or select File->Save menu option.

To test the code, click the green run button at the top of the IDE.

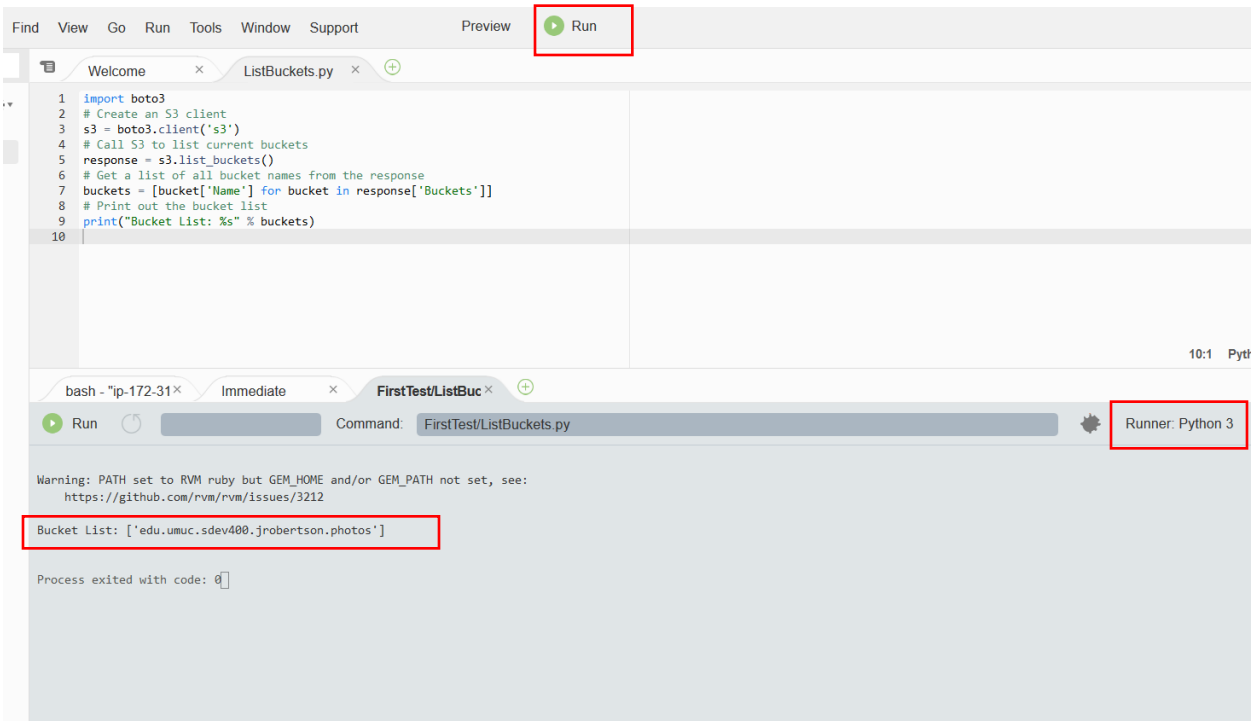

Figure 24 Successful Run Listing S3 buckets

A couple of points are worth mentioning for this example and resulting output:

1. There is warning message that will be printed each time you run a script from this environment. You can ignore this warning.
2. The Runner is listed as Python 3 on the lower right side of output section. This tells us we are running using Python 3. If this is not listed, select the Runner and select Python 3 from the list of options.
3. The output lists the bucket name of "edu.umuc.sdev400.photos" which we created earlier in this document.
4. The code is short but has includes much functionality:
    a. `import boto3` – This statement imports the boto3 Python module. Boto3 is an object-oriented API providing access to AWS services. Without this statement, the program would not know how to communicate with AWS.
    b. `s3 = boto3.client('s3')` – This statement provides a low level client representing an S3 service. As we investigate more options you will see that 's3' can be replaced with other services such as 'ec2', 'rds' and others.
    c. `response = s3.list_buckets()`- This statement calls the list_buckets() method and returns the results in the variable named response. The boto3 API has multiple methods available for each AWS service. Often these methods are in the form of list_XXX(), put_XXX() and get_XXX(). A critical factor in understanding how to handle the results is to know the format and structure of the return type. Often, the return type is a dictionary item with multiple components of varying types. For example, the syntax and structure for the response variable returned in this case are:

```
Response Syntax
    {
     'Buckets': [
            {
                'Name': 'string',
                'CreationDate': datetime(2015, 1, 1)
            },
        ],
     'Owner': {
            'DisplayName': 'string',
            'ID': 'string'
        }
    }


Response Structure

   (dict) --
       Buckets (list) --
           (dict) --
               Name (string) --

               The name of the bucket.
               CreationDate (datetime) --

               Date the bucket was created.
       Owner (dict) --
           DisplayName (string) --
           ID (string) –
```

Python has a very powerful set of data structures that are used extensively by AWS. We explored these data structures in SDEV 300. However; it is recommended to **review this topic as this is the foundation for most of the AWS SDK functionality**.

Notice the syntax uses JavaScript Object Notation(JSON) format. Also notice the combination of dictionary, list and string components in the structure. You can access each element be referring to the item and using the required data structure format. For example, lists, use [ ] to enclose the list of elements. Extracting the specific element and associated value will take some practice.

d. `buckets = [bucket['Name'] for bucket in response['Buckets']] –` This statement retrieves a list of buckets based on the 'Name' attribute. There is actually quite a bit happening in this one statement. The `response['Buckets']` code is referring to the list of Buckets in the response. You could have also used `response['Owner']`. The `for` statement just cycles through each of the Buckets with `bucket['Name']` looks specifically for the `Name` of the bucket. If you wanted to retrieve the CreationDate, use `bucket['CreationDate']`. Note the [ ] enclosing the syntax. This is required as

the results will be a list of bucket names. **Experimentation and a detailed review of Python data structure use is the best approach for truly understanding how this powerful syntax works**.

e. `print("Bucket List: %s" % buckets)` – Finally, this statement takes the list of buckets generate in the above statement and prints them to the screen.

This week's content has a link to the Python digital textbook used in SDEV 300. This is a great resource for reviewing Python data structures. In addition, the following boto3 API reference should be used frequently to locate available methods as well as the corresponding response syntax and structure.

https://boto3.amazonaws.com/v1/documentation/api/latest/index.html