Fruitilicious

Cloud Cruise

Deepika Srinivasan 700693073

Abstract— **Identifying the quality of fresh produce while procuring is a major task that involves time and effort in Retail industry. In addition, human error may also result in wrong choices while differentiating better stock. The main objective of this project is to identify and classify whether the apple fruit is fresh or rotten using Convolutional Neural Networks based Machine Learning model. In this project, we will use AWS EC2 to deploy image classification model in a web-based environment. Once the image is classified, the image file will be renamed to the class defined and dropped in the S3 folder. This, in turn, will trigger AWS Lambda to make use of Simple Email Service (SES) to send a notification email along with the image as an attachment and other information details.**

Keywords-Convolutional Neural Network, Amazon EC2, Amazon S3, Amazon Lambda, Amazon RDS, Amazon SES, SQL Workbench, FileZilla

I. INTRODUCTION

Retail and many supermarkets actually require manual labor to sort and classify fruits depending on maturity level. This includes not just labor costs, but also the time spent in these operations. In this rapidly expanding sector, fruit identification is required followed by maturity level determination.

II. IMAGE CLASSIFER

An image classifier is a type of machine learning for identifying pictures. When you send it a picture, it is reacting to the picture with a name. We will train an image classifier by showing it several examples of already labelled images. In this analysis, for example, we will train an image classifier to identify fresh and rotted apples by showing them a range of apples. After the training is done by the image classifier, the accuracy is tested and, if it performs well enough, it is stored as an Machine Learning model file. If the model needs to be retrained, after tuning the model we will replicate the training cycle again until we have the desired accuracy. To use our image classifier in a web-based environment, this Core ML model file is imported into EC2 instance.

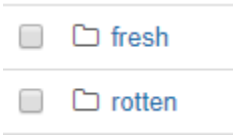
A. Techniques and Tools

Python was chosen as the programming language in this project along with Keras as the platform, which is a high-level neural network API written in Python. Google's TensorFlow is chosen as the software library for low-level

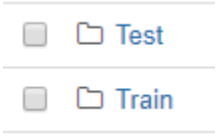
application backend. Jupyter Notebook has been selected as the development environment. Kaggle dataset contains fresh and rotted apple images.

B. Dataset Preparation

Initially, we prepare the data that the classifier will use to learn and evaluate. The training data set is generated from about 80% of the apple images for each label (fresh and rotten), while the evaluation data set is created from the remaining 20% of the images. 'Shutil' library is used for this purpose to copy required files from 'fresh' and 'rotten' folder to 'train' and 'test' folder.



After we have copied the images to 'train' and 'test' folder based on train:test split ratio, we will have two new folders in the current parent folder as shown below



Each of the the above folder will have their own Fresh and Rotten folder with images and the count is shown as follows.

```
Number of training fresh images 1670
Number of training rotten images 2354
Number of Training images  4024
**********************************************************************
Number of testing fresh images 418
Number of testing rotten images 589
Number of Testing images  1007
```

C. CNN Model

Convolutional Neural Networks (CNN), introduced by Yann LeCun in 1988, is a special model of artificial neural networks. CNN takes advantage of some visual cortex features. Image classification is one of the most common uses of this architecture. For instance, Facebook uses CNN for automated tagging algorithms, Amazon — for product recommendations generation, and Google — for searching through photos among users. In more detail: the image is

passed through a series of convolutional, nonlinear, pooling layers and completely connected layers, and then the output is produced.

A convolutional layer includes a series of filters we need to know the parameters of. The filter height and weight are less than the volume of the data. To compute an activation map consisting of neurons, each filter is transformed with the amount of inputs. After each process of the convolution the nonlinear layer is inserted. It has a function to enable which brings nonlinear properties. Under this property a network will not be large enough and cannot model the response variable (as a class label). The pattern of pooling continues after the nonlinear layer. It deals with image width and height and conducts down sampling on them. Consequently, the volume of the image is reduced. Once the sequence of convolutional, nonlinear and pooling layers is complete, a completely connected layer must be added. This layer takes input from convolutional network layers. Attaching a fully connected layer to the end of the network results in a N dimensional matrix, where N is the number of classes the model selects the desired class from.

First step to create such CNN model is called Model Construction. Below CNN model is built along with the chosen hyperparameters in order to maximize the performance.

```
#hyperparameters
FILTER_SIZE = 3
NUM_FILTERS = 32
INPUT_SIZE  = 64
MAXPOOL_SIZE = 2
BATCH_SIZE = 32
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 62, 62, 32)        896
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 29, 29, 32)        9248
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
flatten_3 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_5 (Dense)              (None, 128)               802944
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 129
=================================================================
Total params: 813,217
Trainable params: 813,217
Non-trainable params: 0
_________________________________________________________________
```

Secondly, we will perform Model training to train the model based on the expected output. Model progress is noticeable while the script is running. In the end the model's final accuracy will be recorded. In order to record the best model in our training process we repeat the training phase for 50 iterations and keep saving the best model obtained so far.

```
history = model.fit_generator(generator=training_set,
                              steps_per_epoch=STEP_PER_TRAIN,
                              epochs=50, verbose=1,
                              callbacks=callb_1,
                              validation_data=test_set,
                              validation_steps=STEP_PER_TEST)
```

```
Epoch 00050: val_acc improved from 0.98359 to 0.98667, saving model to FruitStatus_cnn_0.99.h5
```

Thirdly, after the model is trained, model testing can be carried out. A second collection of data is loaded during this process. This data set was never used by the model and must thus be checked for its true accuracy.

```
Found 10 images belonging to 2 classes.
Evaluation loss over Test dataset is :0.1344
Evaluation accuracy over Test dataset is :90.00%
```

From the above experiments, we find the model is capable of correctly classifying all fresh and rotten apples as seen below.

Finally, once the model training is complete, and it is known that the model gives the correct outcome, we can save it locally. The model name is FruitStatus_cnn_0.99.h5 in our experiment. Eventually, the saved model will be used in the real world. This process is called model evaluation. This means the model can be used for analyzing new results.

```
# Loading best Model
my_model = load_model(filepath=r'C:\Users\Deepika\CS5000\Model_Callbacks_Checkpoint_SGD\FruitStatus_cnn_0.99.h5')
```

Instead of a folder containing the images, we can check the model with single file as illustrated below

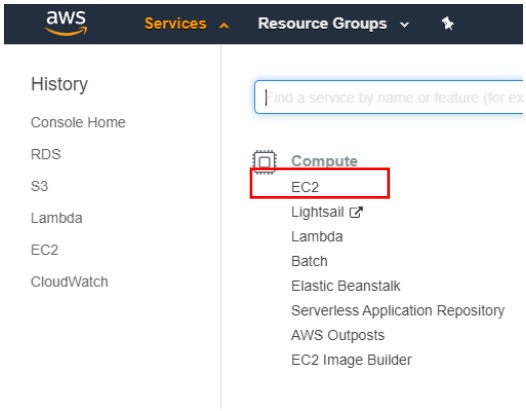
```
My model predicted as [[1.]]
Apple Test Image is predicted as rotten
Apple is predicted as Rotten with probability 100.00%
```

III. Amazon Elastic Compute Cloud – EC2

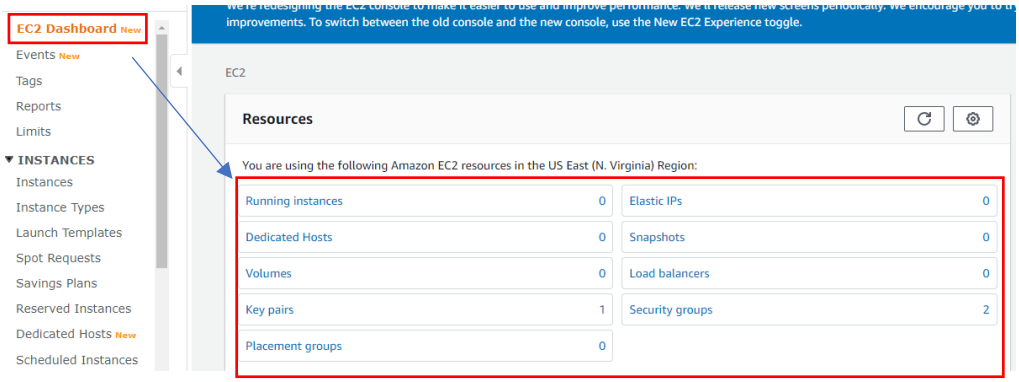
Amazon Elastic Compute Cloud (Amazon EC2) is a cloud-based computing service that offers stable, resizable compute functionality. It is planned to make cloud computing on a network scale simpler for users. It reduces the time taken to get and boot new server instances to minutes, helping you to rapidly scale up and down capacity as computing requirements shift.

Steps to Create EC2 Instance

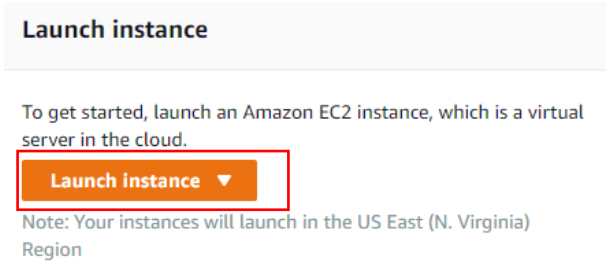
- Login to AWS account, and head to the top left corner of the AWS Services page.
- We can see here all the AWS Products classified according to their region namely Computing, Processing, Archive, etc. To create an EC2 instance, as in the next section, we must select choose EC2 from Compute.This will in turn launch EC2 dashboard.



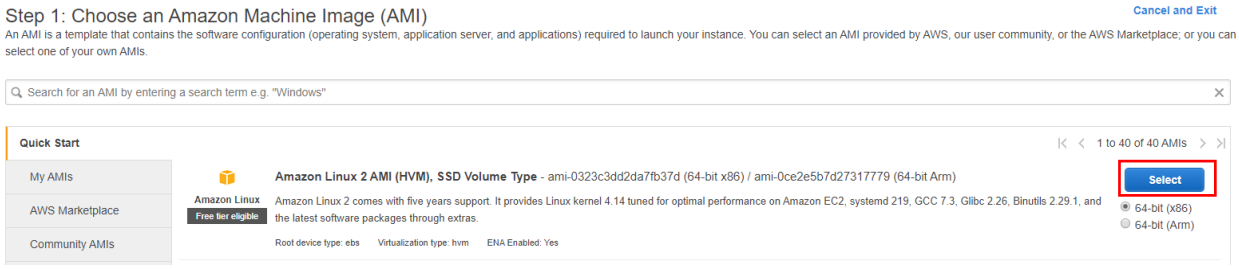
- Below is the EC2 dashboard with all the details listed like running instances, security groups and key pairs available etc.



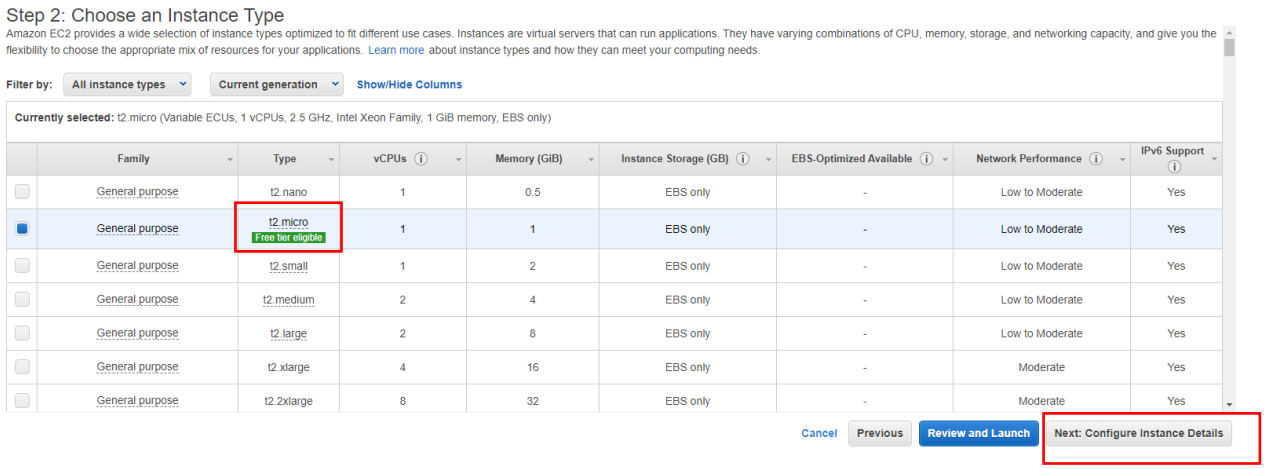
- Click on 'Launch Instance' button in the Build EC2 Instance segment (as seen below) once your desired Region is chosen. When you click 'Launch Instance,' the instance development wizard page will open.



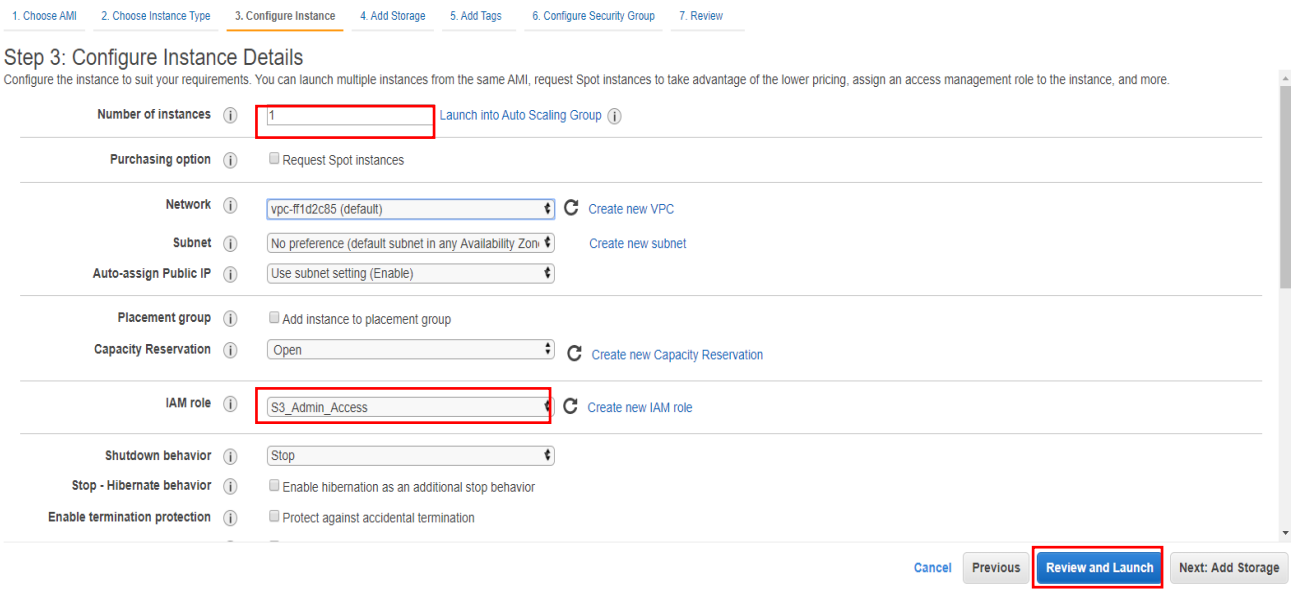
- We will be asked to pick an AMI of our choice in the below page. AMI is an Amazon Software File and it is essentially a simulation of an Operating System framework that we can use as a basis for building our instance. If we start an EC2 instance based on our AMI need, it instantly boots the instance with the appropriate Operating system. Here we select the default Amazon Linux (64 bit) AMI.



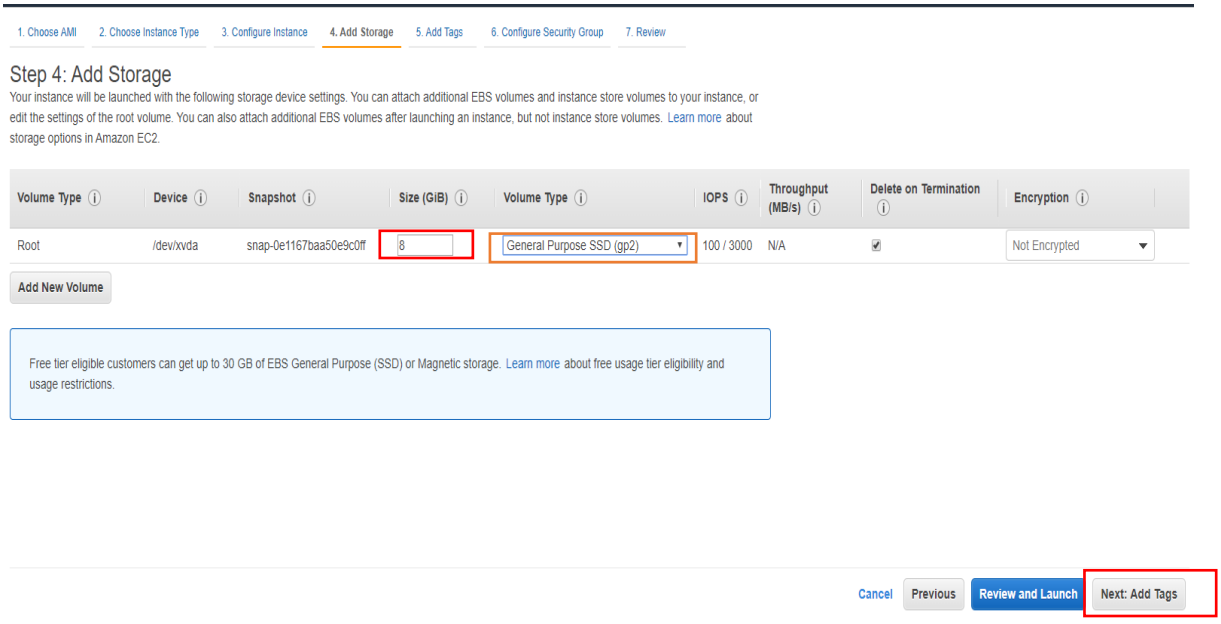
- In the next step, we will pick the type of instance that we need based on the requirements. We will choose the type of t2.micro instance, which is provided with 1vCPU and 1 GB memory server by AWS. Click on "Configure Instance Info" to proceed with next steps.



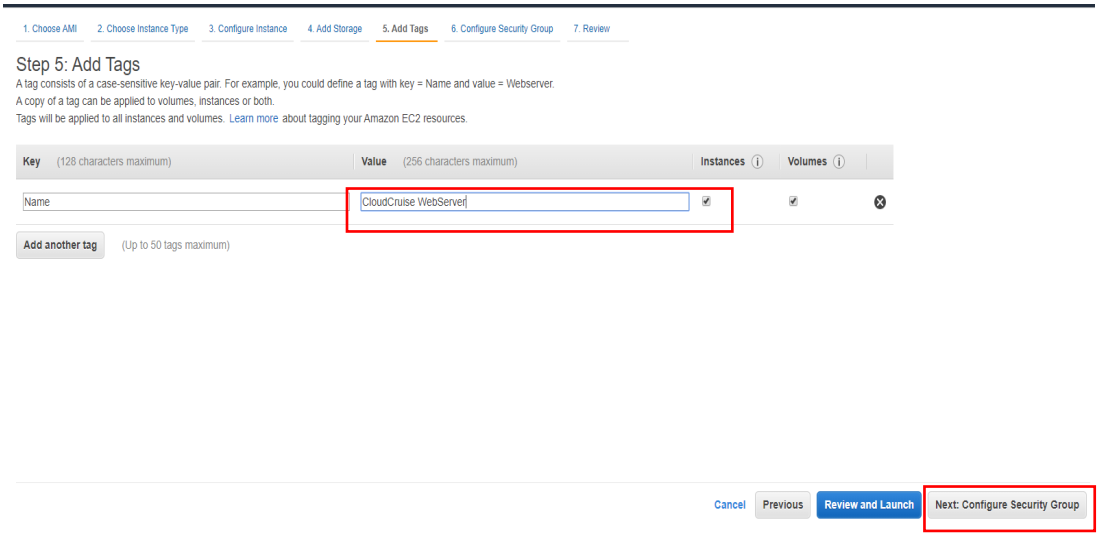
- In configure Instance page we can indicate number of instances we want to launch and the IAM role we will be using in our project. Click on ‘Next: Add Storage’



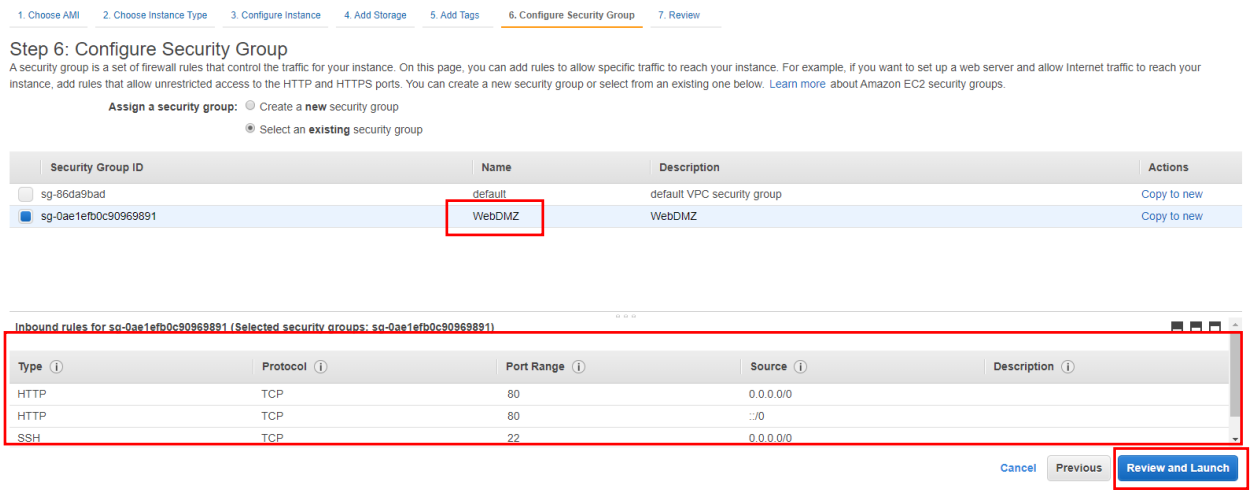
- In the Add Storage stage, we will see that a General Purpose SSD root capacity of 8 GB was automatically given to the instance. (Maximum volume size is 16 GB) We can change the volume size, install new volumes, adjust the volume form, etc. AWS offers three types of volumes EBS- Magnetic, General Purpose SSD, Provided IOPs. Click on 'Next: Add Tags'



- In this step, we can tag instance with a pair of key-values. It gives the AWS account administrator insight when there are loads of instances open. Here we have labelled the instance as ‘CloudCruise WebServer’.Then click on ‘Next: Configure Security Group’



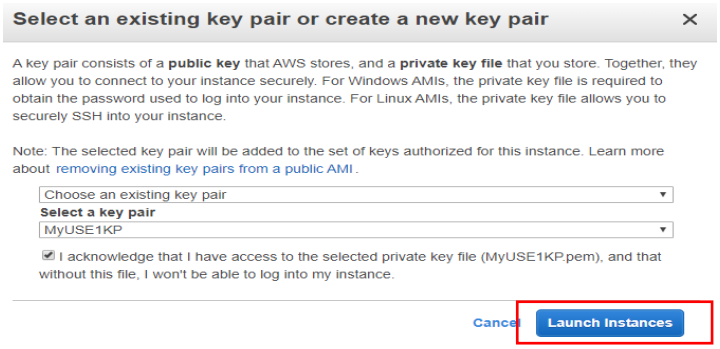
- We will limit traffic on our instance ports in this next phase by configuring the Security Groups. It is an optional function of firewall supported by AWS separately from the firewall of the instance’s default OS firewall. Since our site is a web server, we will build a separate security group named WebDMZ for easier reference. We will also establish the protocols that we want to allow in our scenario. We will also assign IPs to reach our instance on the mentioned protocols. Once the firewall rules are set, we will click ‘Review and launch’ button.



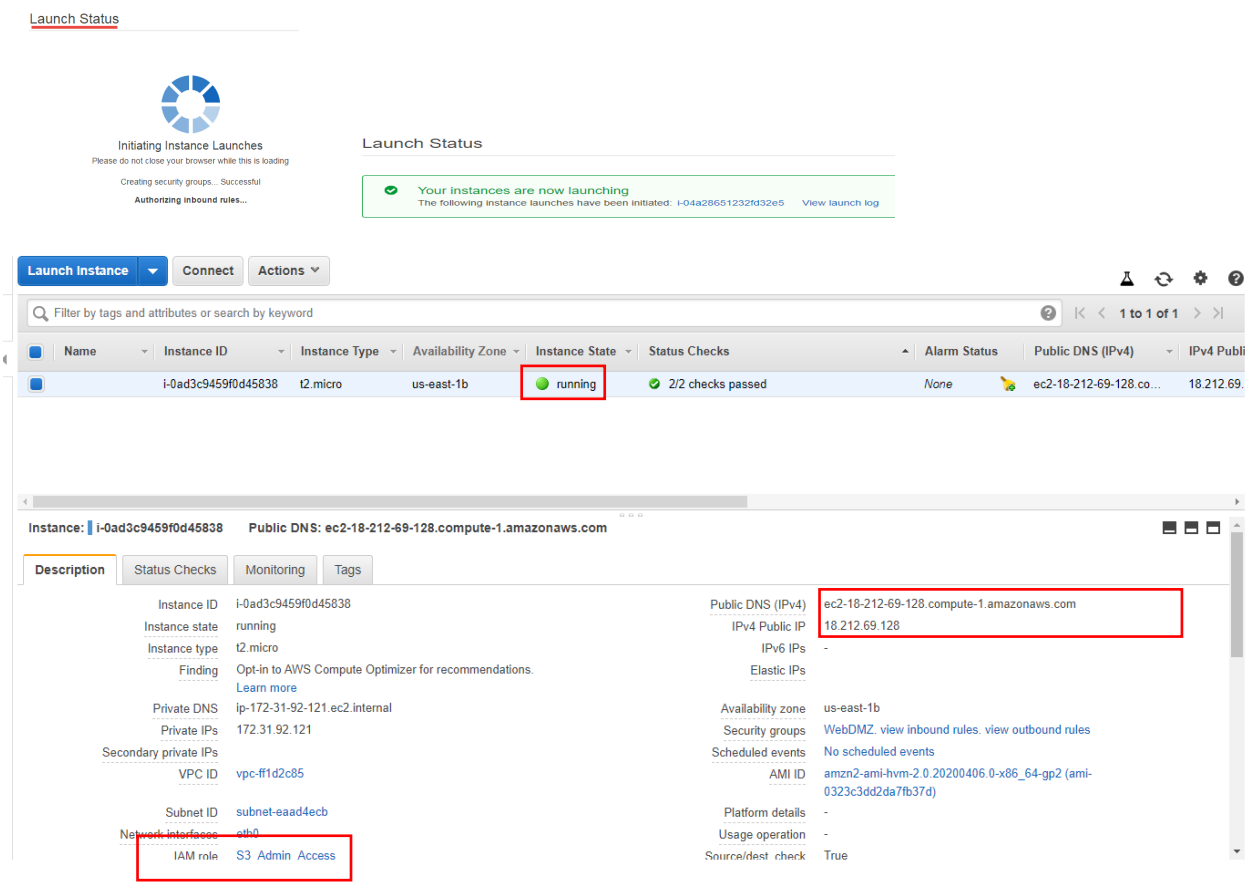
- In this final step we will review all the settings and go ahead and click ‘Launch’.

We will be asked in the next stage to build a key pair to access the instance. A key pair is a combination of public-private keys. In this case, AWS holds the public key, and we are asked to keep our private key safe. We need to first generate a new key pair and then give our key a name. Eventually this can be copied and stored in our protected folder.

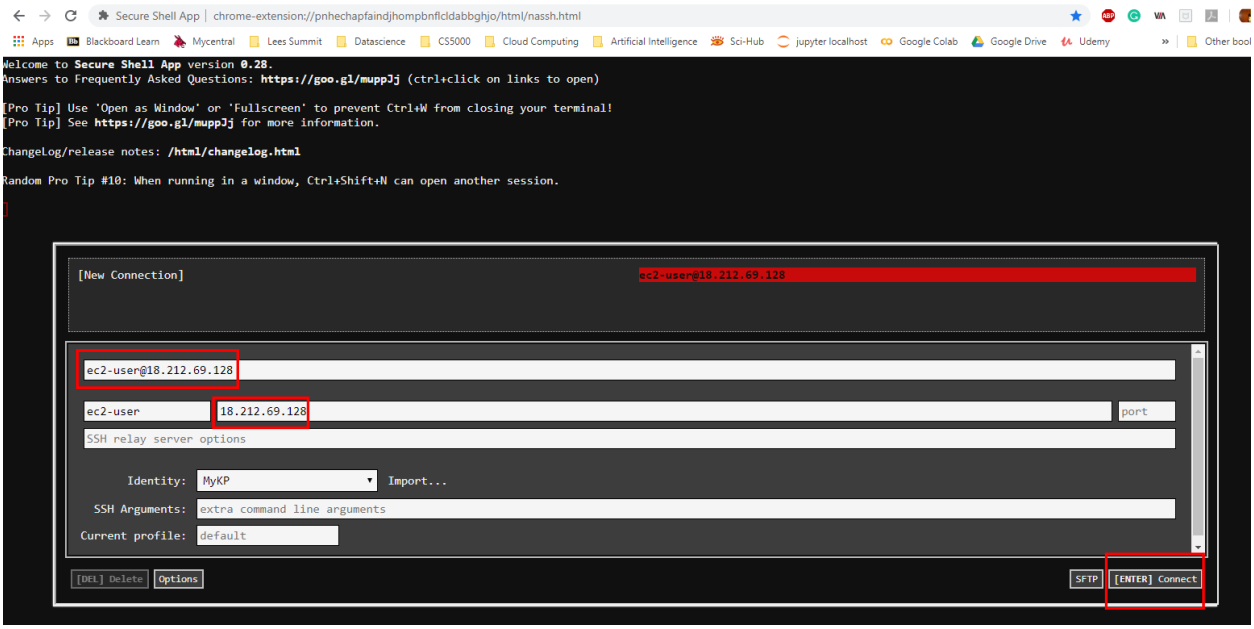
After downloading the key pair we can launch the instance.

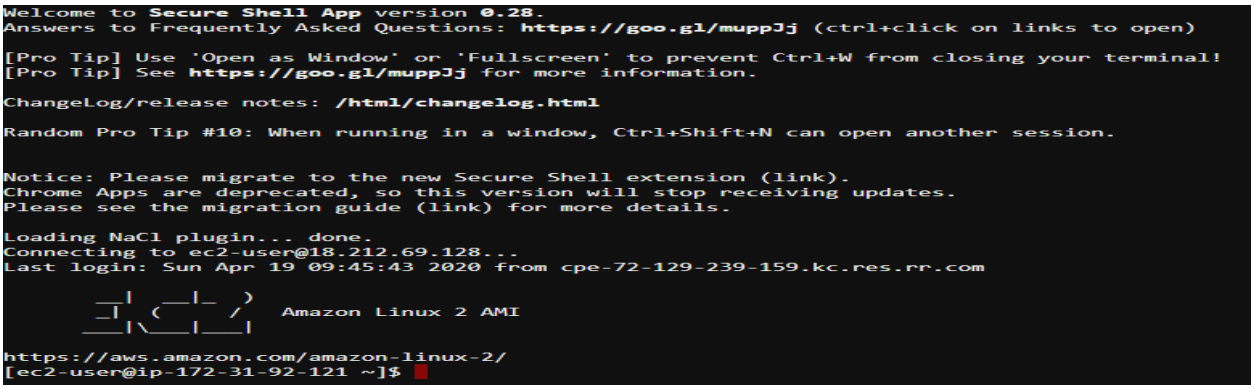


We can see launch status and launch log as shown below



In order to connect to EC2 instance created just now, we can ssh using chrome extension. we will copy the Public IP and enter in the below page as shown and then click '[Enter] Connect'



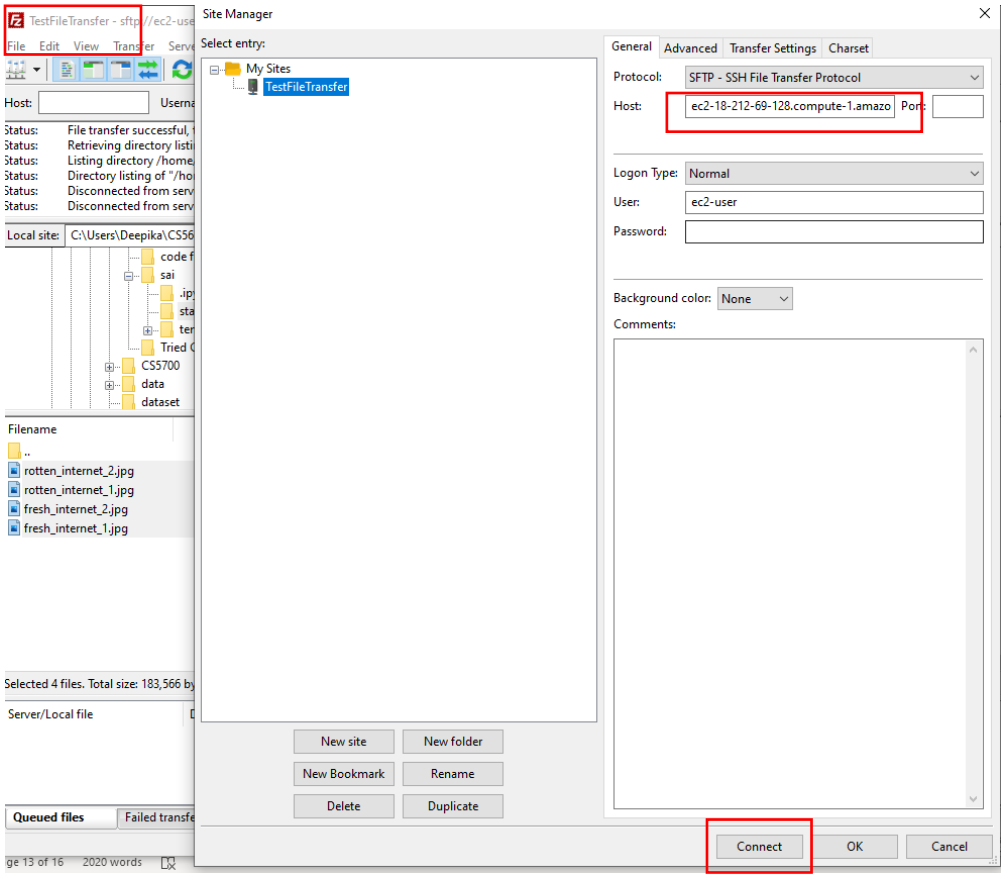


Now we have logged in to our EC2 instance

FileZilla

We will make use of FileZilla to transfer the files from our local machine to EC2 instance.

Go to ‘File’ and select ‘Site Manager’. Establish connection with EC2 instance by indicating the Public DNS(IPv4) in Host tab



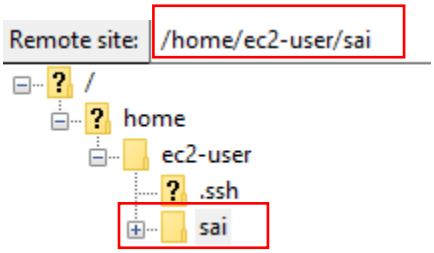
After we establish successful connection with EC2 server we will be able to view below status in FileZilla

In the next step we will transfer the folder named ‘sai’ to EC2 instance

Filename	Filesize	Filetype	Last modified
..			
.ipynb_checkpoints		File folder	4/16/2020 3:59:51 ...
code from ec2		File folder	3/29/2020 4:15:34 ...
sai		File folder	3/29/2020 4:15:27 ...
Tried Code		File folder	3/29/2020 4:25:37 ...
Fruitilicious_CNNModel.i...	45,466	IPYNB File	4/16/2020 4:00:18 ...

Filename	Filesize	Filetype	Last modified	Permissions	Owner/Group
..					
.ssh		File folder	4/16/2020 8:25:...	drwx------	ec2-user ec...
.bash_logout	18	BASH_LOG...	1/15/2020 6:56:...	-rw-r--r--	ec2-user ec...
.bash_profile	193	BASH_PRO...	1/15/2020 6:56:...	-rw-r--r--	ec2-user ec...
.bashrc	231	BASHRC File	1/15/2020 6:56:...	-rw-r--r--	ec2-user ec...

Once the files are transferred we can see the folder appearing in EC2 instance.



In order to start using our EC2 instance we may need to perform certain steps as mentioned below

To obtain root privileges we can make use of the below command

```
[ec2-user@ip-172-31-44-222 ~]$ sudo su
```

Yum update command with no packages, can update every program currently installed. Yum will guarantee that all requirements are met when upgrading packages.

```
[root@ip-172-31-44-222 ec2-user]# yum update
```

```
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
No packages marked for update
```

Python will be installed in EC2 instance using below command

```
[root@ip-172-31-92-121 sai]# yum install python-pip
```

```
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
---> Package python3-pip.noarch 0:9.0.3-1.amzn2.0.2 will be installed
--> Processing Dependency: python(abi) = 3.7 for package: python3-pip-9.0.3-1.amzn2.0.2.noarch
--> Processing Dependency: python3-setuptools for package: python3-pip-9.0.3-1.amzn2.0.2.noarch
--> Processing Dependency: /usr/bin/python3 for package: python3-pip-9.0.3-1.amzn2.0.2.noarch
--> Running transaction check
---> Package python3.x86_64 0:3.7.6-1.amzn2.0.1 will be installed
--> Processing Dependency: python3-libs(x86-64) = 3.7.6-1.amzn2.0.1 for package: python3-3.7.6-1.amzn2.0.1.x86_64
--> Processing Dependency: libpython3.7m.so.1.0()(64bit) for package: python3-3.7.6-1.amzn2.0.1.x86_64
---> Package python3-setuptools.noarch 0:38.4.0-3.amzn2.0.6 will be installed
--> Running transaction check
---> Package python3-libs.x86_64 0:3.7.6-1.amzn2.0.1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package                 Arch          Version                  Repository    Size
================================================================================
Installing:
 python3-pip             noarch        9.0.3-1.amzn2.0.2        amzn2-core   1.9 M
Installing for dependencies:
 python3                 x86_64        3.7.6-1.amzn2.0.1        amzn2-core    71 k
 python3-libs            x86_64        3.7.6-1.amzn2.0.1        amzn2-core   9.1 M
 python3-setuptools      noarch        38.4.0-3.amzn2.0.6       amzn2-core   617 k

Transaction Summary
================================================================================
Install  1 Package (+3 Dependent packages)

Total download size: 12 M
Installed size: 50 M
Is this ok [y/d/N]: y
```

```
Downloading packages:
(1/4): python3-3.7.6-1.amzn2.0.1.x86_64.rpm                                                    |  71 kB  00:00:00
(2/4): python3-pip-9.0.3-1.amzn2.0.2.noarch.rpm                                                |  1.9 MB  00:00:00
(3/4): python3-setuptools-38.4.0-3.amzn2.0.6.noarch.rpm                                        |  617 kB  00:00:00
(4/4): python3-libs-3.7.6-1.amzn2.0.1.x86_64.rpm                                               |  9.1 MB  00:00:00
--------------------------------------------------------------------------------------------------------------------
Total                                                                          26 MB/s |  12 MB  00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : python3-pip-9.0.3-1.amzn2.0.2.noarch                                                1/4
  Installing : python3-3.7.6-1.amzn2.0.1.x86_64                                                    2/4
  Installing : python3-setuptools-38.4.0-3.amzn2.0.6.noarch                                        3/4
  Installing : python3-libs-3.7.6-1.amzn2.0.1.x86_64                                               4/4
  Verifying  : python3-setuptools-38.4.0-3.amzn2.0.6.noarch                                        1/4
  Verifying  : python3-pip-9.0.3-1.amzn2.0.2.noarch                                                2/4
  Verifying  : python3-3.7.6-1.amzn2.0.1.x86_64                                                    3/4
  Verifying  : python3-libs-3.7.6-1.amzn2.0.1.x86_64                                               4/4

Installed:
  python3-pip.noarch 0:9.0.3-1.amzn2.0.2

Dependency Installed:
  python3.x86_64 0:3.7.6-1.amzn2.0.1                  python3-libs.x86_64 0:3.7.6-1.amzn2.0.1                  python3-setuptools.noarch 0:38.4.0-3.amzn2.0.6

Complete!
[root@ip-172-31-44-222 ec2-user]#
```

```
[root@ip-172-31-92-121 sai]# pip --version
pip 20.0.2 from /usr/lib/python2.7/site-packages/pip (python 2.7)
```

```
[root@ip-172-31-92-121 sai]# python --version
Python 2.7.16
```

In order to install Tensorflow we need to execute the following lines of code

```
[root@ip-172-31-92-121 sai]# pip install --no-code-dir --user tensorflow==1.14.0
```

Now, we will install all necessary libraries to run our CNN Model in EC2 instance. We have these libraries stored in requirements.txt file

```
[root@ip-172-31-92-121 sai]# cat requirements.txt
flask
keras
numpy
pandas
wtforms
scikit-image
boto3
mysql-connector
```

```
[root@ip-172-31-92-121 sai]# pip install --user -r requirements.txt
```

Or we can run the files individually as follows with library replaced with the list provided in requirements.txt

```
[root@ip-172-31-92-121 sai]# pip install --user flask
```

We can see the status of library installation as follows for each library indicated in requirements.txt file.

```
WARNING: Running pip install with root privileges is generally not a good idea. Try `pip3 install --user` instead.
Collecting flask (from -r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/f2/28/2a03252dfb9ebf377f40fba6a7841b47083260bf8bd8e737b0c6952df83f/Flask-1.1.2-py2.py3-none-any.whl (94kB)
    100% |████████████████████████████████| 102kB 7.6MB/s
Collecting keras (from -r requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/ad/fd/6bfe87920d7f4fd475acd28500a42482b6b84479832bdc0fe9e589a60ceb/Keras-2.3.1-py2.py3-none-any.whl (377kB)
    100% |████████████████████████████████| 378kB 3.0MB/s
Collecting numpy (from -r requirements.txt (line 3))
  Downloading https://files.pythonhosted.org/packages/b7/ce/d0b92f0283faa4da76ea82587ff9da70104e81f59ba14f76c87e4196254e/numpy-1.18.2-cp37-cp37m-manylinux1_x86_64.whl (20.2MB)
    100% |████████████████████████████████| 20.2MB 64kB/s
Collecting pandas (from -r requirements.txt (line 4))
  Downloading https://files.pythonhosted.org/packages/4a/6a/94b219b8ea0f2d580169e85ed1edc0163743f55aaeca8a44c2e8fc1e344e/pandas-1.0.3-cp37-cp37m-manylinux1_x86_64.whl (10.0MB)
```

Finally, we can see the libraries installed using the following command and verify the needed libraries and highlighted.

```
[root@ip-172-31-92-121 sai]# pip freeze
```

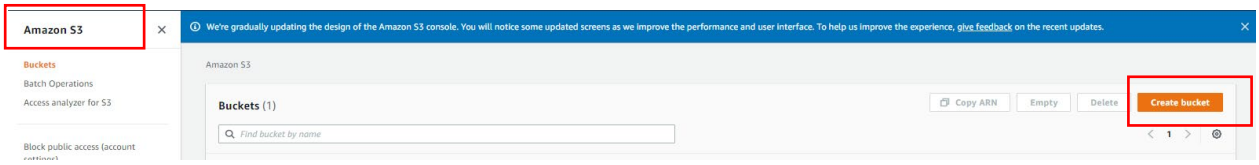
```
abs1-py==0.9.0
amazon-linux-extras==1.6.10
astor==0.8.1
aws-cfn-bootstrap==1.4
awscli==1.16.300
Babel==0.9.6
backports.ssl-match-hostname==3.5.0.1
backports.weakref==1.0.post1
boto3==1.12.41
botocore==1.15.41
cffi==1.6.0
chardet==2.2.1
click==7.1.1
cloud-init==19.3
colorama==0.3.2
configobj==4.7.2
cryptography==1.7.2
docutils==0.12
ec2-hibinit-agent==1.0.2
enum34==1.1.10
ez-setup==0.9
Flask==1.1.2
funcsigs==1.0.2
futures==3.0.5
gast==0.3.3
google-pasta==0.2.0
grpcio==1.28.1
h5py==2.10.0
hibagent==1.1.0
idna==2.4
iniparse==0.4
ipaddress==1.0.16
itsdangerous==1.1.0
Jinja2==2.11.2
jmespath==0.9.3
jsonpatch==1.2
```

```
jsonpointer==1.9
jsonschema==2.5.1
jwcrypto==0.4.2
Keras==2.3.1
Keras-Applications==1.0.8
Keras-Preprocessing==1.1.0
kitchen==1.1.1
langtable==0.0.31
lockfile==0.9.1
Markdown==3.1.1
MarkupSafe==1.1.1
mock==3.0.5
mysql-connector==2.2.9
numpy==1.16.6
oauthlib==2.0.1
pandas==0.24.2
Pillow==2.0.0
ply==3.4
protobuf==3.11.3
pyasn1==0.1.9
pycparser==2.14
pycurl==7.19.0
pygpgme==0.3
pyliblzma==0.5.3
pystache==0.5.3
python-daemon==1.6
python-dateutil==2.6.0
pytz==2019.3
```

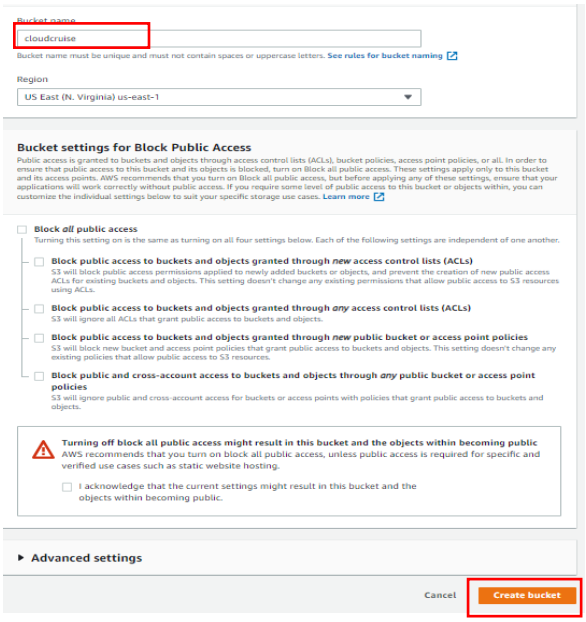
```
pyxattr==0.5.1
PyYAML==3.10
repoze.lru==0.4
requests==2.6.0
rpm-python==4.11.3
rsa==3.4.1
s3transfer==0.3.3
scipy==1.2.3
simplejson==3.2.0
six==1.14.0
tensorboard==1.14.0
tensorflow==1.14.0
tensorflow-estimator==1.14.0
termcolor==1.1.0
urlgrabber==3.10
urllib3==1.24.3
Werkzeug==1.0.1
wrapt==1.12.1
WTForms==2.2.1
yum-langpacks==0.4.2
yum-metadata-parser==1.1.4
```

IV. Amazon Simple Storage - S3

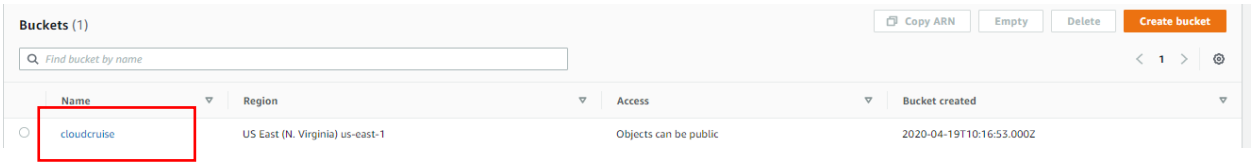
Select S3 from Management Console. Click on ‘Create Bucket’



Create “cloudcruise” bucket in S3 folder with public access and then choose “Create bucket”

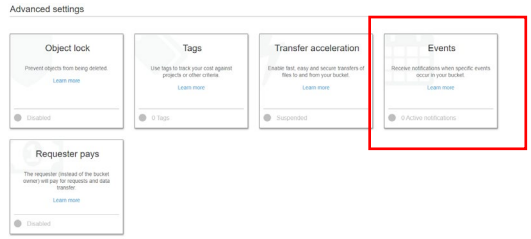


Thus, we have created “cloudcruise” bucket

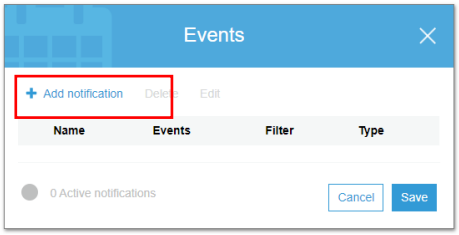


We will also create the event notification step for S3 folder so that when an image is added into ‘cloudcruise’ bucket, lambda function will be triggered.

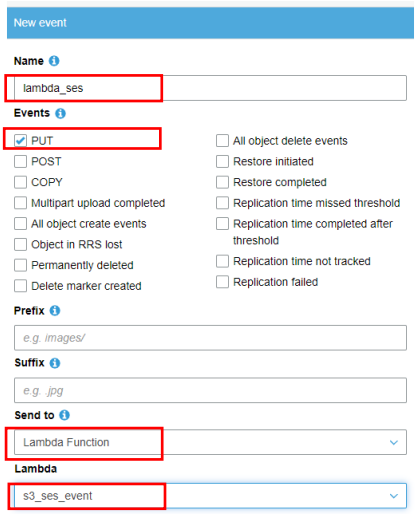
Choose ‘Properties’ tab and select ‘Events’ under Advanced settings.



Choose Events and click on ‘Add Notification’

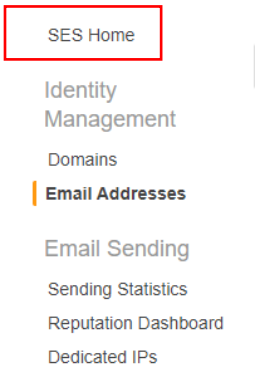


Configure the event to trigger lambda on any files added into ‘cloudcruise’

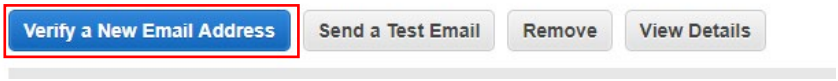


V. Amazon Simple Email Service – SES

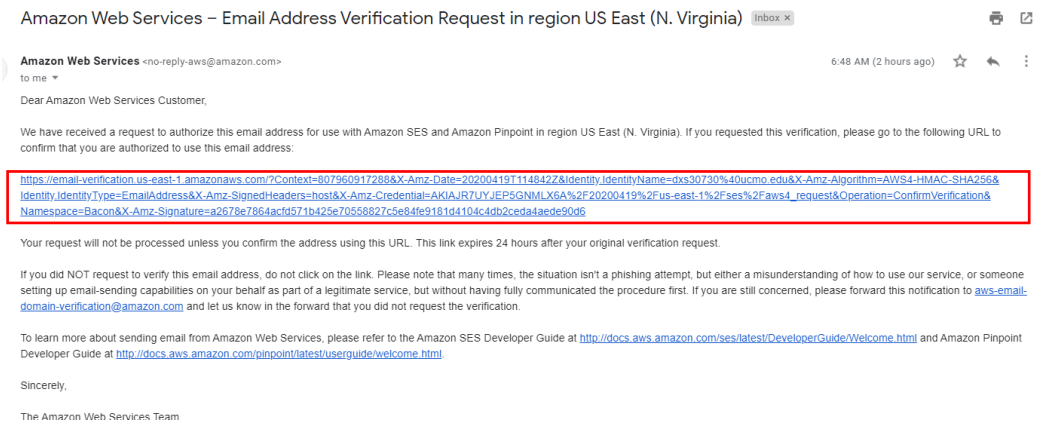
Initially, we need to make sure that we verify the email addresses that we will be using .Choose SES from the services and select 'Email Addresses'



Click on 'Verify a New Email Address' and provide the email address



We will get confirmation email like the one below


Amazon Web Services – Email Address Verification Request in region US East (N. Virginia) Inbox ×

Amazon Web Services <no-reply-aws@amazon.com> 6:48 AM (2 hours ago)
to me

Dear Amazon Web Services Customer,

We have received a request to authorize this email address for use with Amazon SES and Amazon Pinpoint in region US East (N. Virginia). If you requested this verification, please go to the following URL to confirm that you are authorized to use this email address:

https://email-verification.us-east-1.amazonaws.com/?Context=807960917288&X-Amz-Date=20200419T114842Z&Identity.IdentityName=dxs30730%40ucmo.edu&X-Amz-Algorithm=AWS4-HMAC-SHA256&Identity.IdentityType=EmailAddress&X-Amz-SignedHeaders=host&X-Amz-Credential=AKIAJR7UYJEP5GNMLX6A%2F20200419%2Fus-east-1%2Fses%2Faws4_request&Operation=ConfirmVerification&Namespace=Bacon&X-Amz-Signature=a2678e7864acfd571b425e70558827c5e84fe9181d4104c4db2ceda4aede90d6

Your request will not be processed unless you confirm the address using this URL. This link expires 24 hours after your original verification request.

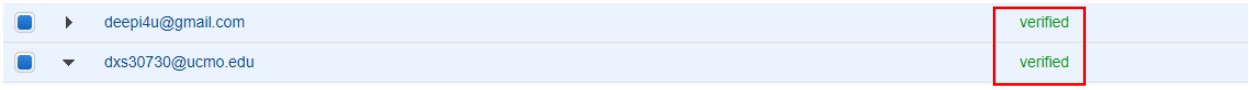
If you did NOT request to verify this email address, do not click on the link. Please note that many times, the situation isn't a phishing attempt, but either a misunderstanding of how to use our service, or someone setting up email-sending capabilities on your behalf as part of a legitimate service, but without having fully communicated the procedure first. If you are still concerned, please forward this notification to aws-email-domain-verification@amazon.com and let us know in the forward that you did not request the verification.

To learn more about sending email from Amazon Web Services, please refer to the Amazon SES Developer Guide at http://docs.aws.amazon.com/ses/latest/DeveloperGuide/Welcome.html and Amazon Pinpoint Developer Guide at http://docs.aws.amazon.com/pinpoint/latest/userguide/welcome.html.

Sincerely,

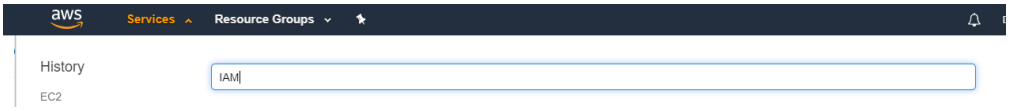
The Amazon Web Services Team

After clicking on the confirmation link, we will be able to use Simple email service

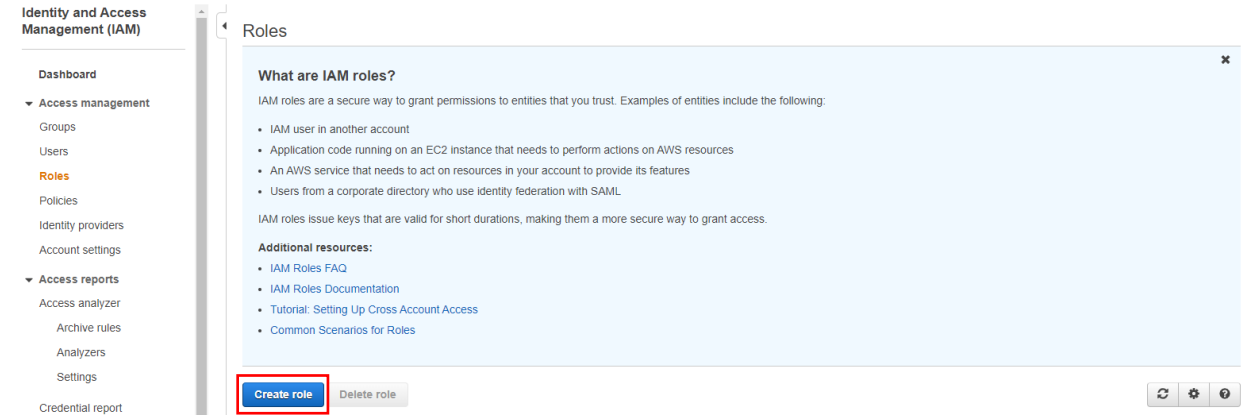


VI. Identitity Access Management - IAM

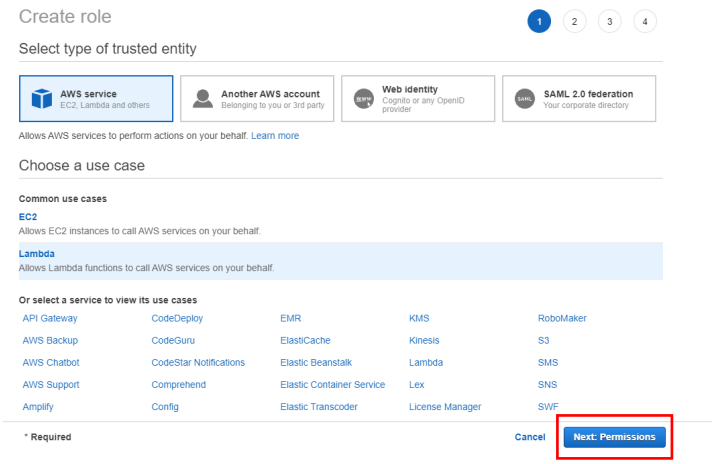
Firstly, we will create IAM role to have access to Lambda and SES.So choose IAM from the list of EC2 Services.



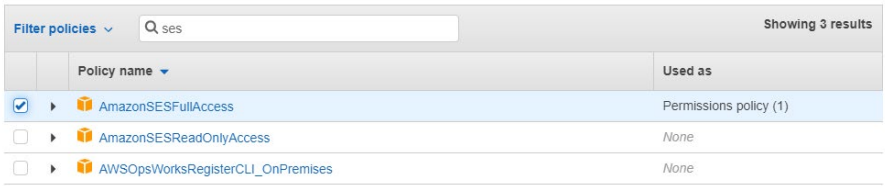
Secondly, in IAM dashboard click ‘Create role’

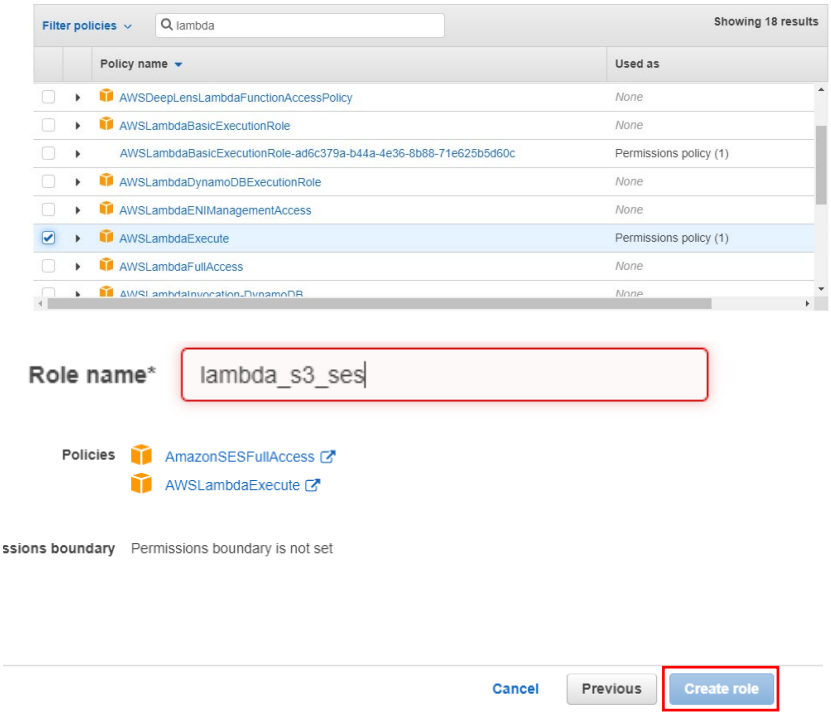


Choose Lambda and click on ‘Next: Permissions’

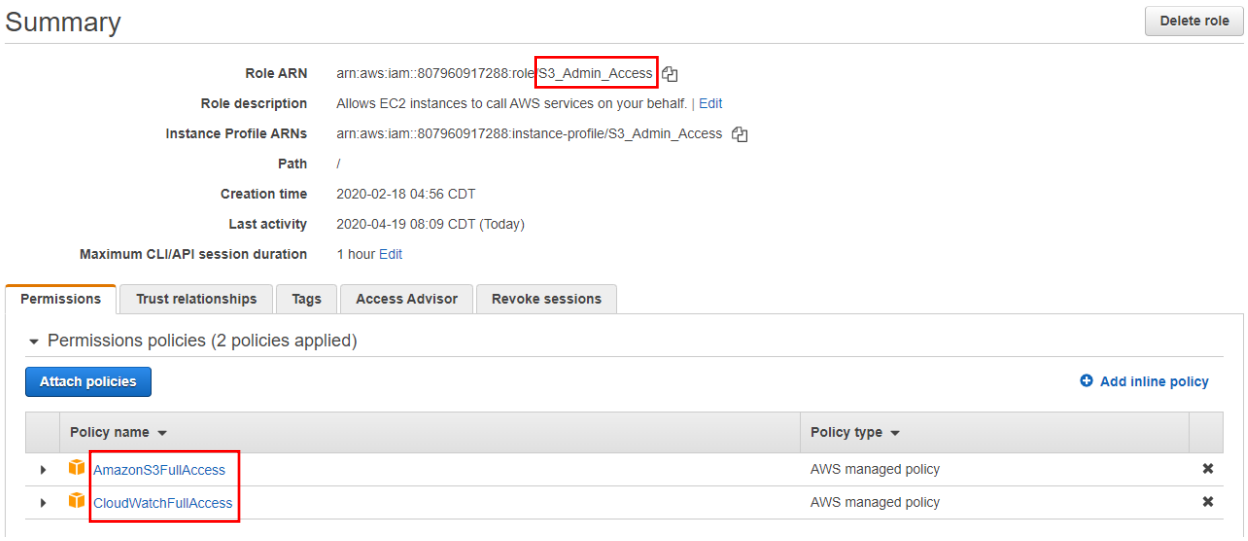


Choose appropriate policy as shown below and then create the role called ‘lambda_se3_ses’



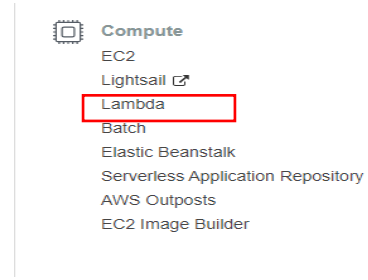


We have also created S3_Admin_Access role by following the steps as shown above

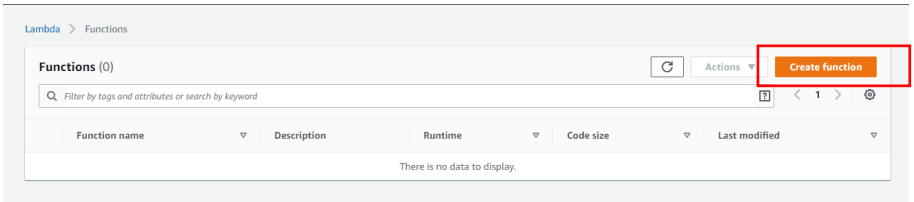


VII. AWS Lambda

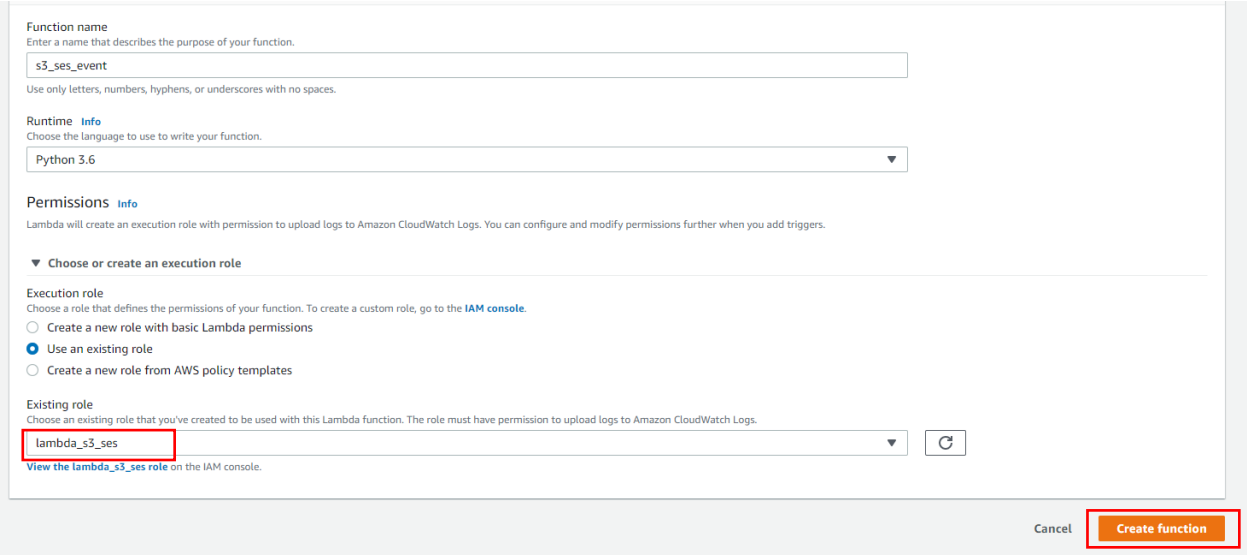
Select Lambda from Compute services



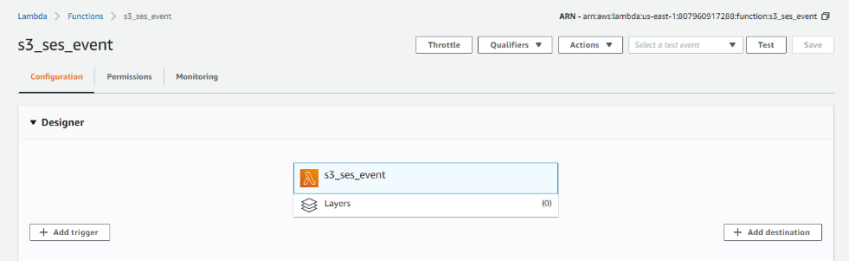
Click on 'Create Function'



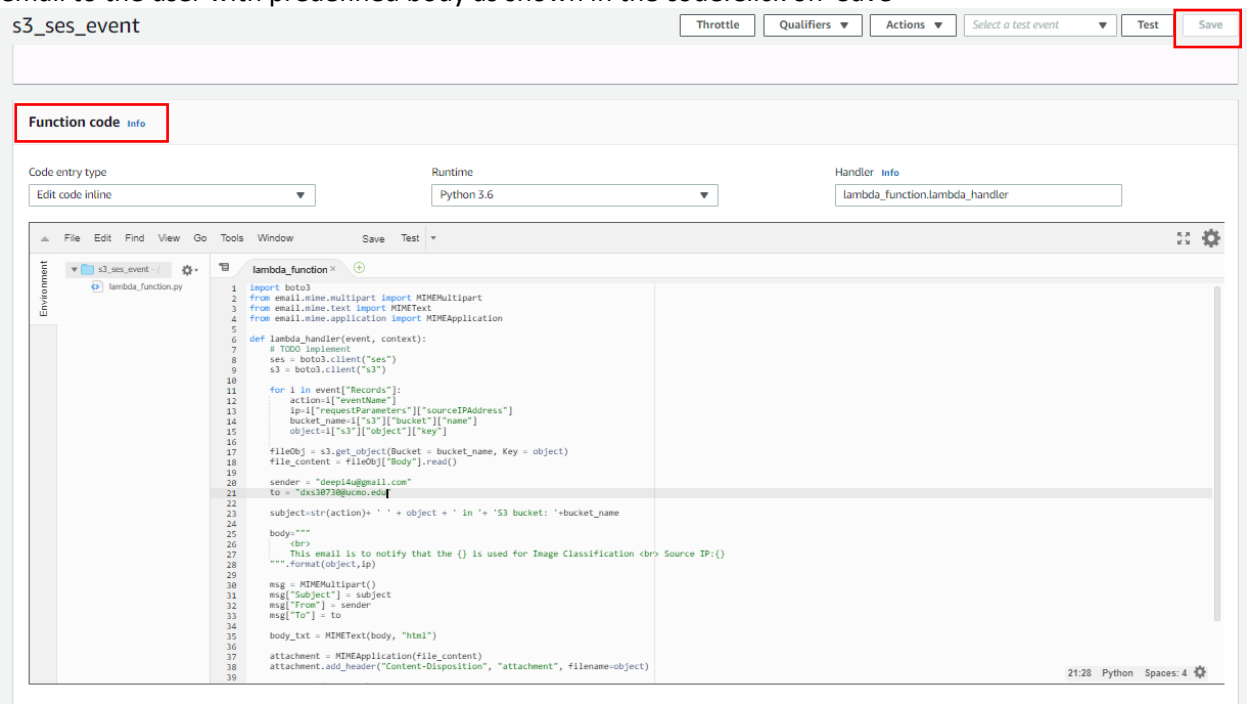
Create a new function called 's3_ses_event' with the role created 'lambda_s3_ses' using IAM step and click on "Create function"



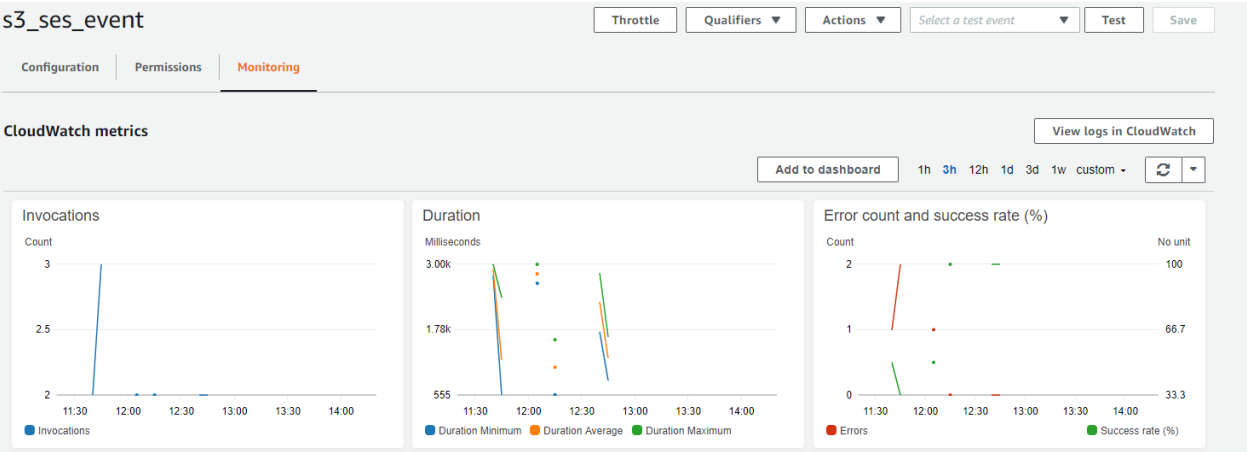
We can see that the lambda function is created in Lambda console



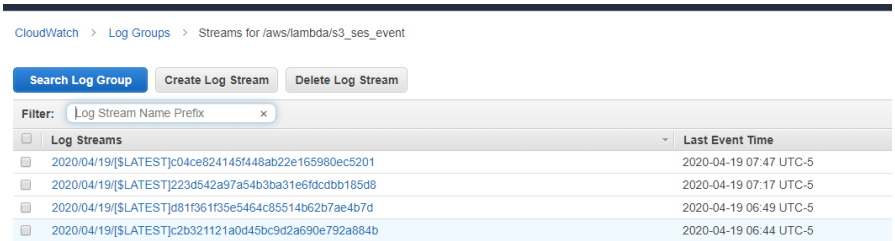
In function code section, add the code provided in appendix ‘lambda_code.py’.This will help to send the email to the user with predefined body as shown in the code.Click on ‘Save’



We can make use of cloud metrics for monitoring purpose

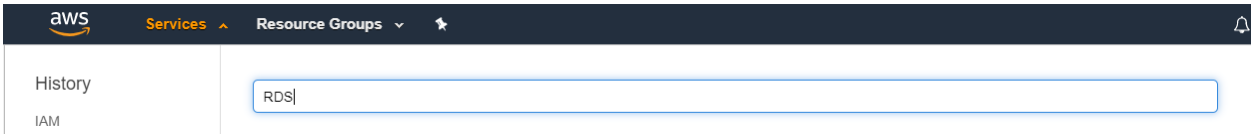


Logs are also updated in CloudWatch for debugging purpose.

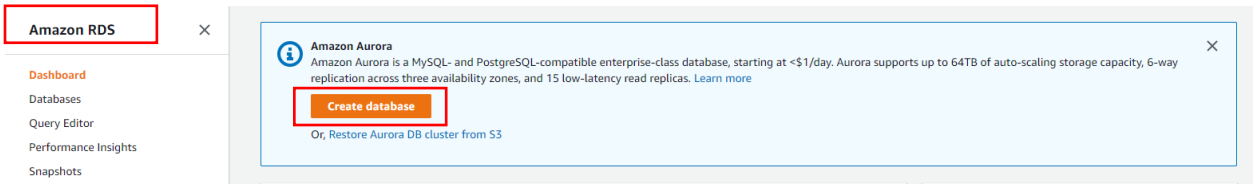


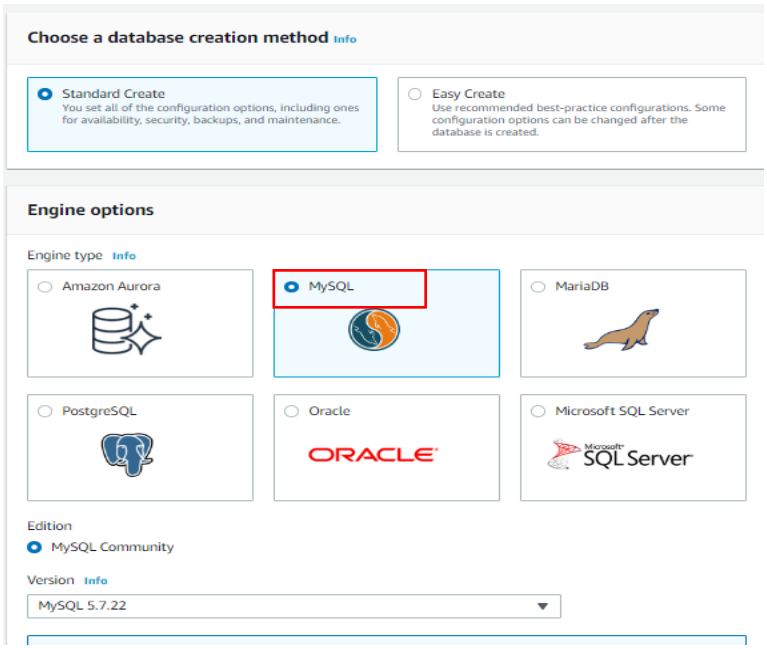
VIII. Amazon Relational Database Service

Choose RDS from Services

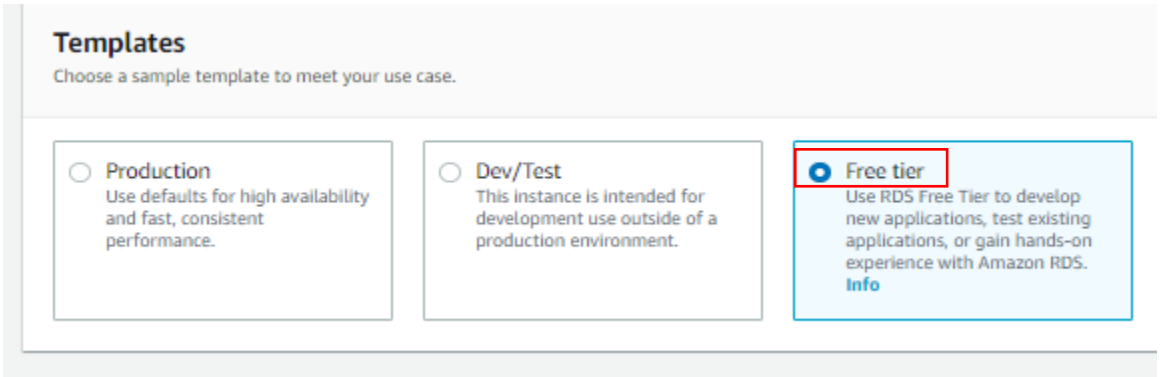


Click on ‘Create Database’ and choose ‘MySQL’

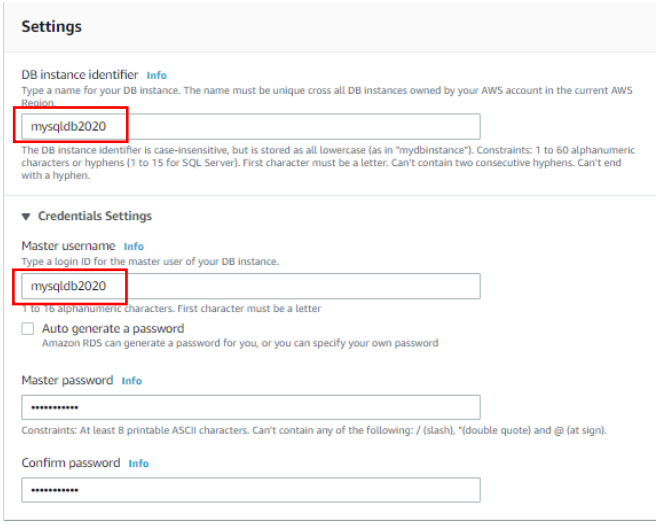




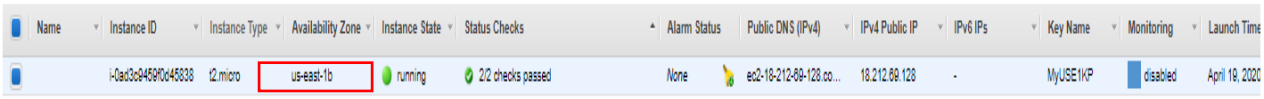
Select ‘Free Tier’ in Templates section

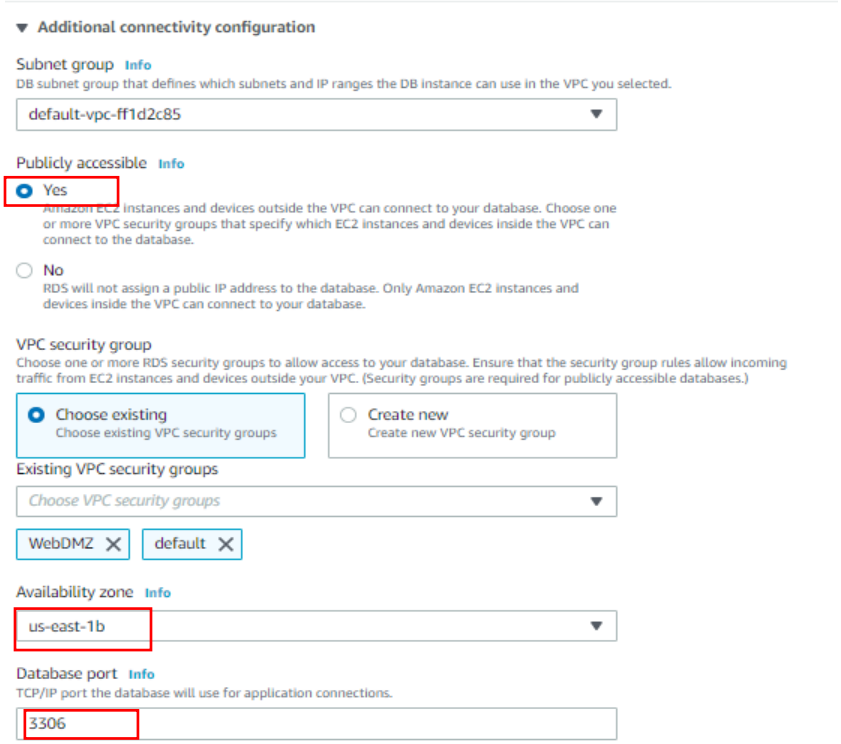


Provide ‘mysqldb2020’ in identifier name,username and password

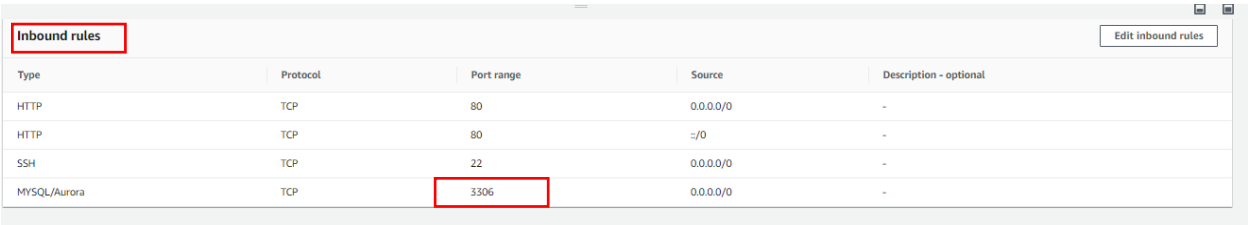


In additional connectivity page,make the Database publicly accessible,update security group as ‘WebDMZ’ and also availability zone same as EC2 as shown below





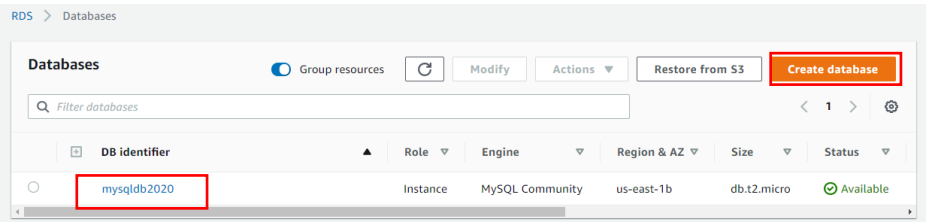
It should be noted that WebDMZ ports are highlighted as shown below for secure connection between services



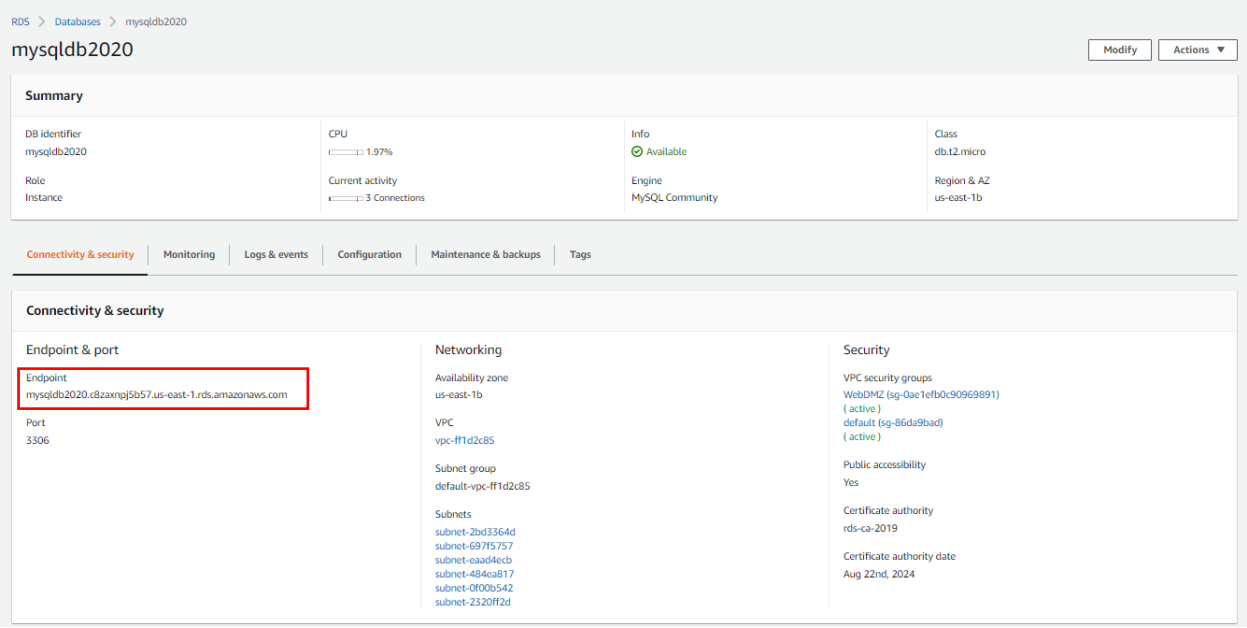
Security group has been updated with inbound rule to accommodate port 3306 as show above.

It should be noted that port 3306 can be used for connecting to this database.

Finally, click on 'Create Database'. Click on 'mysqldb2020' to get more details



After database is created we can find below update in RDS dashboard along with endpoint information.



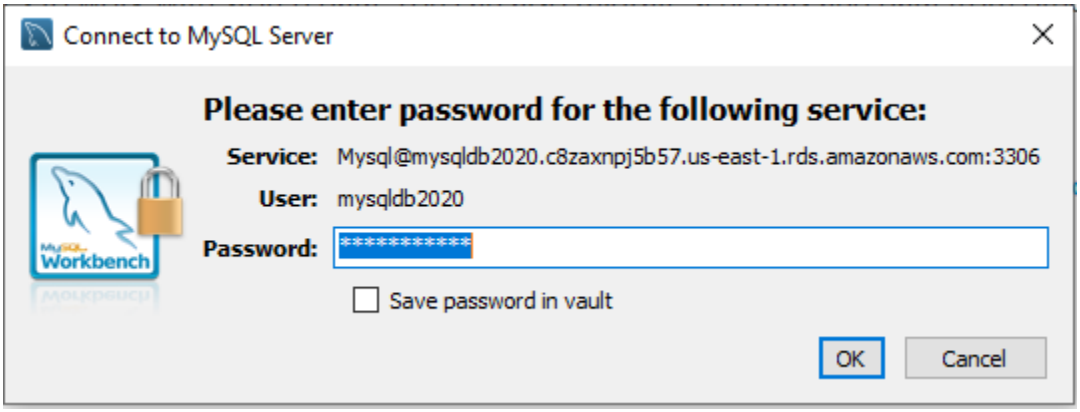
We update the RDS endpoint in our main code 'fruit_status_flask_ec2_s3_rds.py' provided in Appendix

```
#Mysql connection
mydb = mysql.connector.connect(
  host="mysqldb2020.c8zaxnpj5b57.us-east-1.rds.amazonaws.com",
  user="mysqldb2020",
  passwd="mysqldb2020",
  database="cloudcruise"
)
```

SQL Workbench

In SQL Workbench, provide RDS endpoint in host name along with port 3306, and password.Click ‘ok’.

Additionaly enter database name and password.



Currently, following tables are part of the database by default.

```
1  show databases;
```

Result Grid | Filter Rows:

Database
information_schema
innodb
mysql
performance_schema
sys

Lets create table in ‘mysqldb2020’ using db.py provided in Appendix.

```
import mysql.connector

mydb = mysql.connector.connect(
  host="mysqldb2020.c8zaxnpj5b57.us-east-1.rds.amazonaws.com",
  user="mysqldb2020",
  passwd="mysqldb2020"
)

print(mydb)

mycursor = mydb.cursor()
mycursor.execute("CREATE DATABASE cloudcruise")

mycursor.execute("USE cloudcruise")

mycursor.execute("CREATE TABLE logs (image_id INT AUTO_INCREMENT,image_name TEXT,image_class TEXT,PRIMARY KEY (image_id))ENGINE=INNODB")

mycursor.execute("SHOW TABLES")
for x in mycursor:
        print(x)

mycursor.execute("SHOW DATABASES")
for x in mycursor:
  print(x)
```

In the above code, we have first provided connection using our RDS endpoint as host, username and password.Secondly we create database named ‘cloudcruise’ with new table ‘logs’ that have three columns.

image_id	image_name	image_class

```
[root@ip-172-31-92-121 sai]# python db.py
<mysql.connector.connection.MySQLConnection object at 0x7f66d3ed4710>
(u'logs',)
(u'information_schema',)
(u'cloudcruise',)
(u'innodb',)
(u'mysql',)
(u'performance_schema',)
(u'sys',)
[root@ip-172-31-92-121 sai]#
```

We can verify the same from SQL workbench as shown below

```
1 • show databases;
```

```
1  use cloudcruise;
2 • show tables;
```

Result Grid | Filter Rows:

Database
information_schema
cloudcruise
innodb
mysql
performance_schema
sys

Result Grid | Filter Rows:

Tables_in_cloudcruise
logs

Table	Create Table
logs	CREATE TABLE `logs` (`image_id` int(11) NOT NULL AUTO_INCREMENT, `image_name` text, `image_class` text, PRIMARY KEY (`image_id`)) ENGINE=InnoDB DEFAULT CHARSET=latin1

After the table is created we also update this part in our main code 'fruit_status_flask_ec2_s3_rds.py' provided in Appendix

```
#Insert the record into logs table
mycursor = mydb.cursor()
sql = "INSERT INTO logs (image_name, image_class) VALUES (%s, %s)"
val = (image_url, pred_class)
mycursor.execute(sql, val)
mydb.commit()
print(mycursor.rowcount, "record inserted.")
```

IX. Project Implementation

After the file transfer using FileZilla, we have new folder ‘sai’ created in EC2 instance

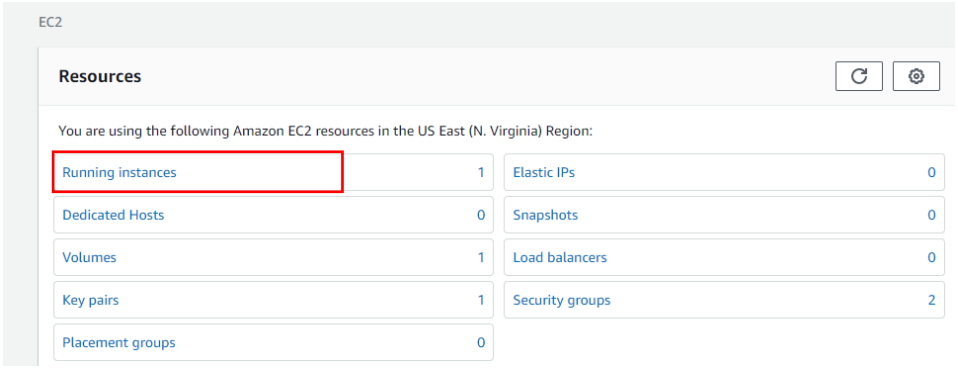
```
[root@ip-172-31-92-121 sai]# ls
db.py                fresh_internet_2.jpg     fruit_status_flask_ec2.py     fruit_status_flask_ec2_s3_rds.py  lambda_code.py     rotten_internet_1.jpg  static
fresh_internet_1.jpg FruitStatus_cnn_0.99.h5  fruit_status_flask_ec2_s3.py  lambda_code_from_ec2.py           requirements.txt   rotten_internet_2.jpg  templates
```

FruitStatus_cnn_0.99.h5 is the model we created for Image classfication to differentiate fresh and rotten apples.

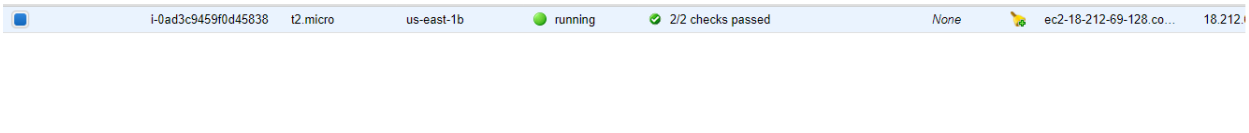
To execute the code that is already transferred to EC2 we will run the following command

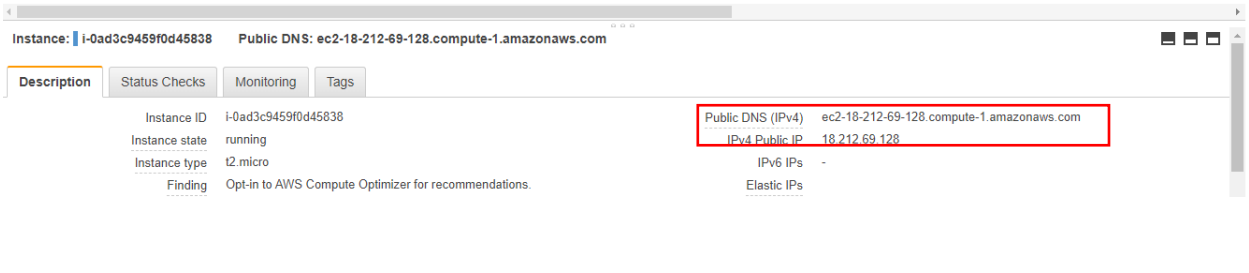
```
[root@ip-172-31-92-121 sai]# python fruit_status_flask_ec2_s3_rds.py
Using TensorFlow backend.
WARNING:tensorflow:From fruit_status_flask_ec2_s3_rds.py:42: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.

 * Serving Flask app "fruit_status_flask_ec2_s3_rds" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:80/ (Press CTRL+C to quit)
```

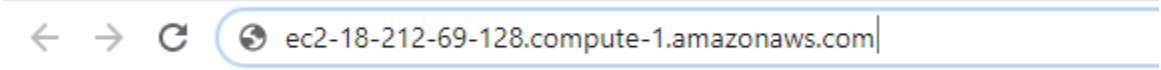
After we run this code, we can go to EC2 dashboard and click on ‘Running Instance’



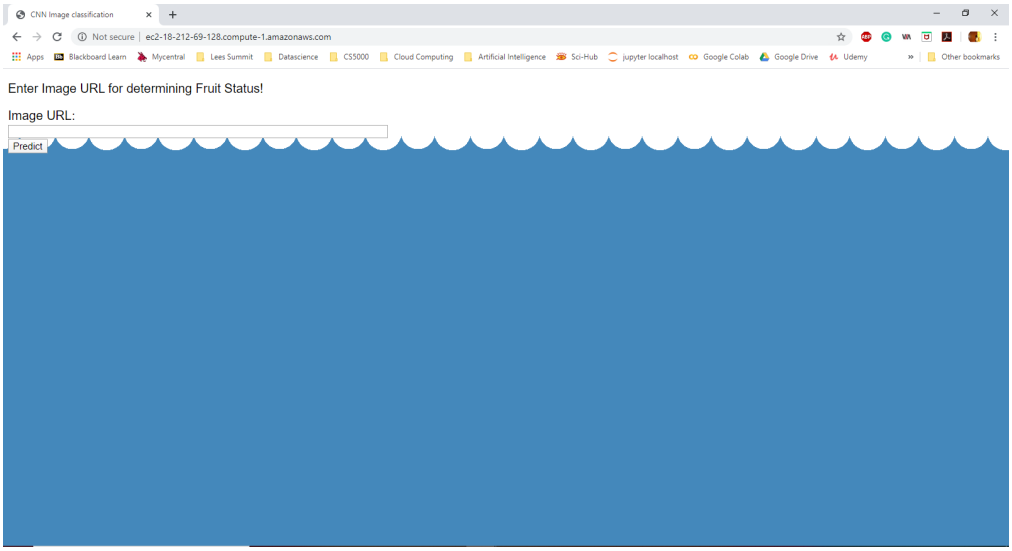
We can either copy Public DNS(IPv4) or IPv4 Public IP and paste it in web browser to access EC2 web browser



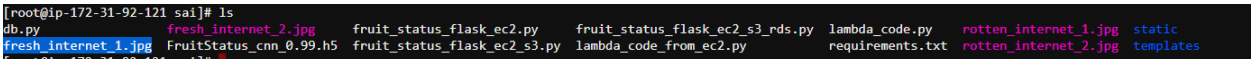


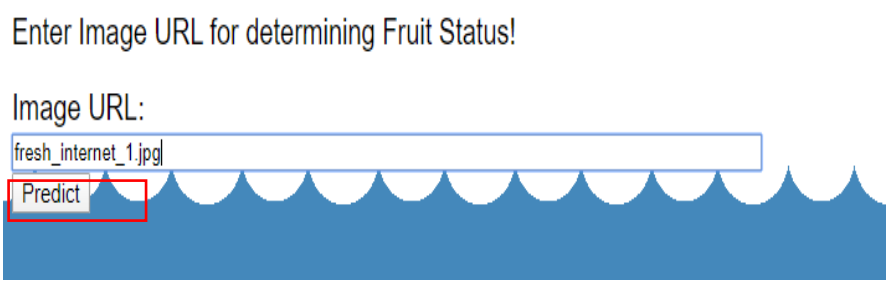


We will get below webpage

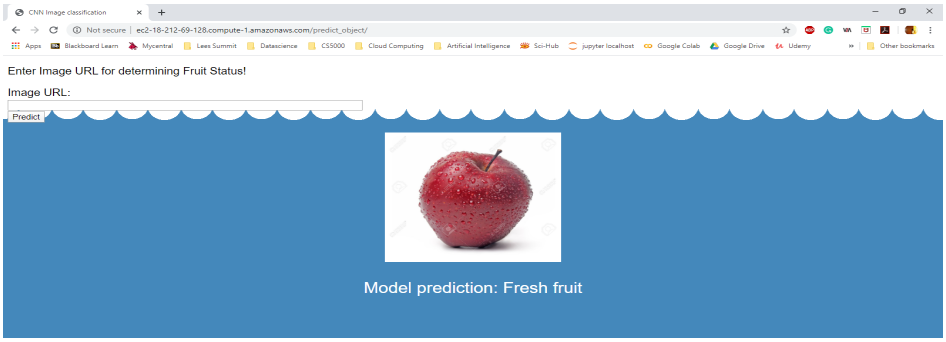


Next we will provide any one of the test image that is available in EC2 ‘sai’ folder in Image URL textbox and click ‘Predict’





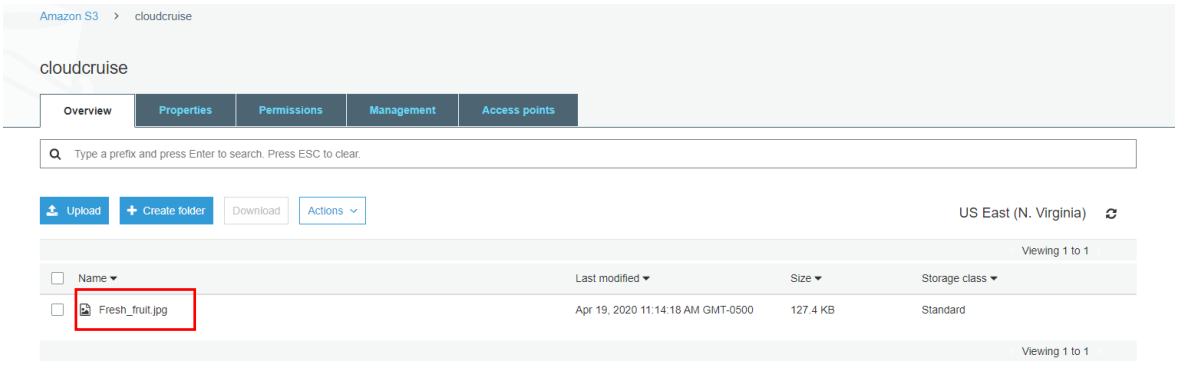
Once we click Predict button we will obtain our Model prediction as shown below



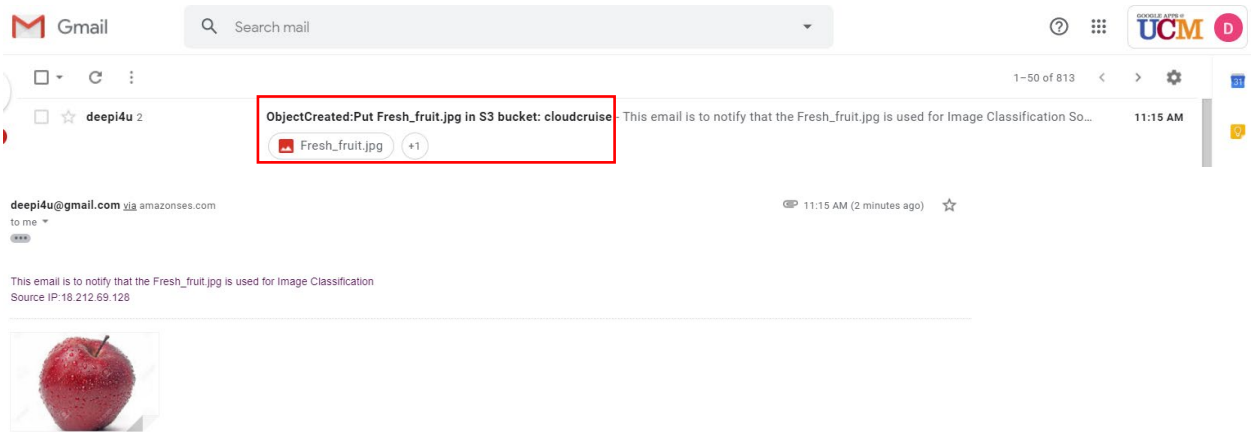
We can also see that EC2 terminal is updated with logs indicating that the apple image uploaded belong to ‘Fresh fruit’ class.In addition it also shows that logs table is also inserted with new record.

```
72.129.239.159 - - [19/Apr/2020 16:21:36] "GET / HTTP/1.1" 200 -
WARNING:tensorflow:From /root/.local/lib/python2.7/site-packages/keras/backend/tensorflow_backend.py:4070: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.

WARNING:tensorflow:From /root/.local/lib/python2.7/site-packages/tensorflow/python/ops/nn_impl.py:180: where (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a futu
re version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /root/.local/lib/python2.7/site-packages/keras/backend/tensorflow_backend.py:422: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables
instead.

('image_url is:', u'fresh_internet_1.jpg')
('Classification-', 'Fresh fruit')
Fresh fruit
Python module executed successfully
(1, 'record inserted.')
```

After model prediction, our python code will send the image to 'cloudcruise' S3 bucket as shown below along with updated name based on model prediction.



This will inturn trigger our lambda function to send email to verified users in Simple email service along with image as an attachment.

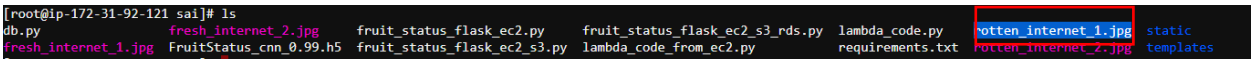


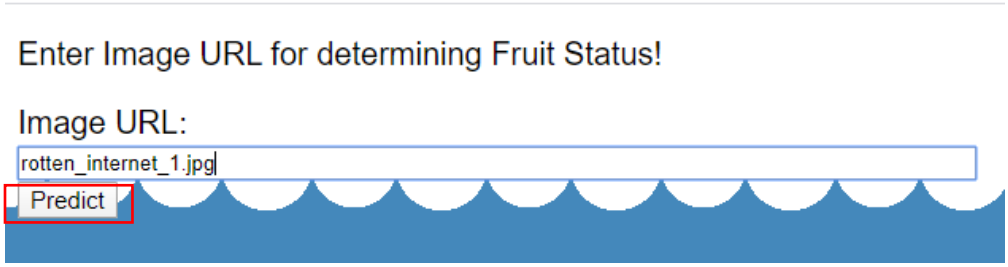
We can also verify that the table is updated with the new record added to our table

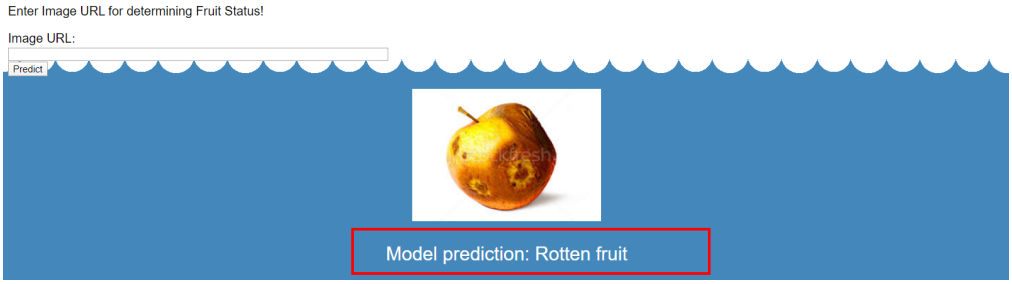
```
select * from logs;
```

image_id	image_name	image_class
1	fresh_internet_1.jpg	Fresh fruit
NULL	NULL	NULL

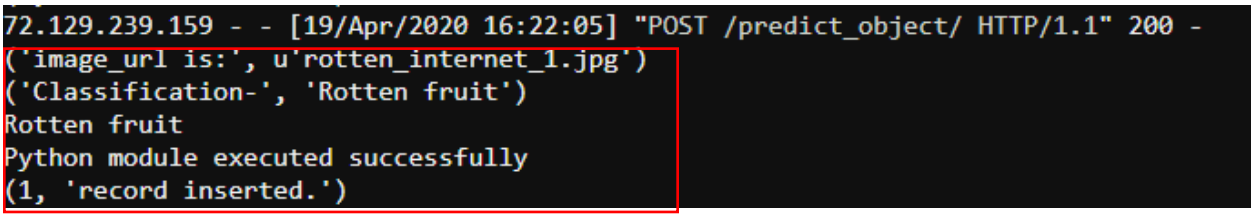
Program is terminated only when we execute Cntrl+Z .So we will try one more test image for rotten apple and check the updates.We will follow the same steps as shown above.



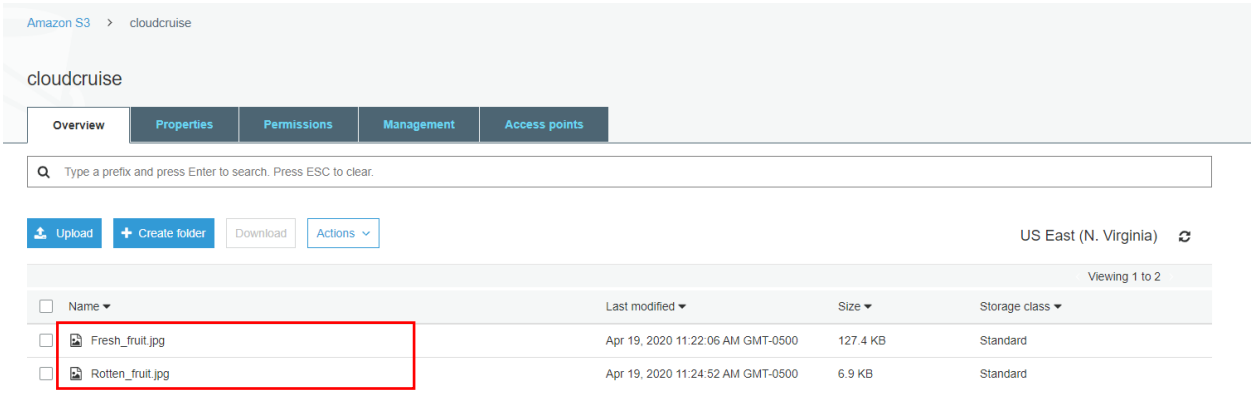




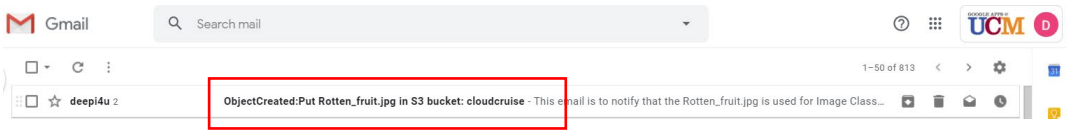
Similar to previous test image we can also see that EC2 console is updated logs for rotten fruit

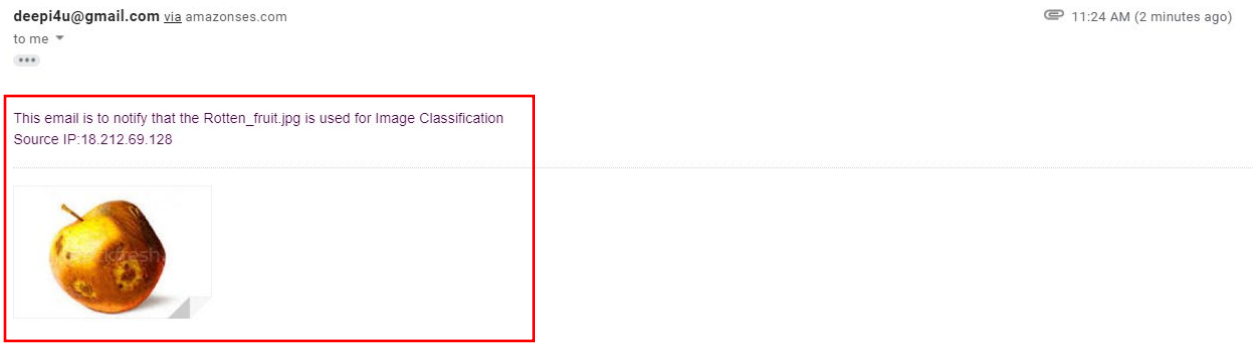


Now we have two images in S3 bucket with current rotten apple image added



Rotten_fruit.jpg triggered lambda function which will send email using Simple Email Service.





Lets check the logs table for record updates

```
1 •  select * from logs;
```

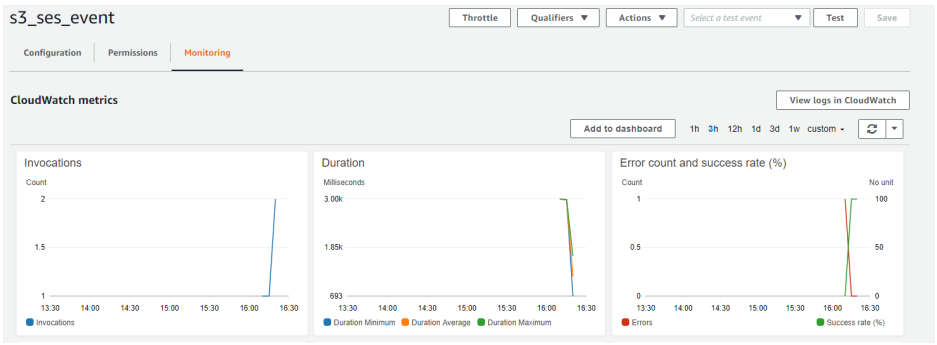
	image_id	image_name	image_class
▶	1	fresh_internet_1.jpg	Fresh fruit
	2	rotten_internet_1.jpg	Rotten fruit
*	NULL	NULL	NULL

It should be noted that EC2 terminal is still running with python code and expecting user input in EC2 web browser.

```
72.129.239.159 - - [19/Apr/2020 16:24:52] "POST /predict_object/ HTTP/1.1" 200 -
```

When we are done with Image classfication, we can terminate the execution.

We can also monitor using the cloudwatch



X. Conclusion

In conclusion, we have made use of Amazon Web Service to deploy Machine Learning model as a Web application.In addition, Simple Storage Service- S3, Simple Email Service- SES , Amazon Lambda, Amazon Relational Database service have been explored and integrated to deliver end-to-end product for consumer.Apart from these services, FileZilla and SQL Workbench have also incorporated in the application for additional features.