



# Fruitylicious

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 CLASSIFIER

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 a 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.

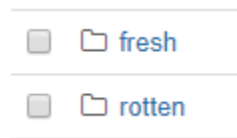
### A. Techniques and Tools

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

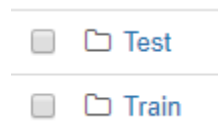
application backend. Jupyter Notebook has been selected as the development environment. Kaggle dataset contains fresh and rotted apple images.

## 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.

Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d_5 (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_6 (Conv2D)	(None, 29, 29, 32)	9248
max_pooling2d_6 (MaxPooling2D)	(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		

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

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.

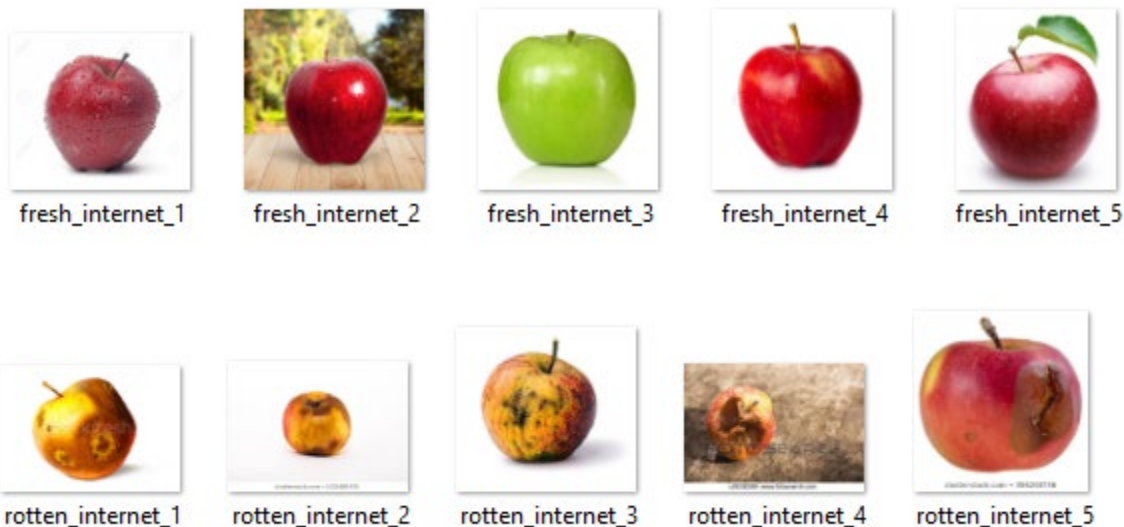
```
history = model.fit_generator(generator=training_set,
                             steps_per_epoch=STEP_PER_TRAIN,
                             epochs=50, verbose=1,
                             callbacks=callb_1, |
                             validation_data=test_set,
                         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

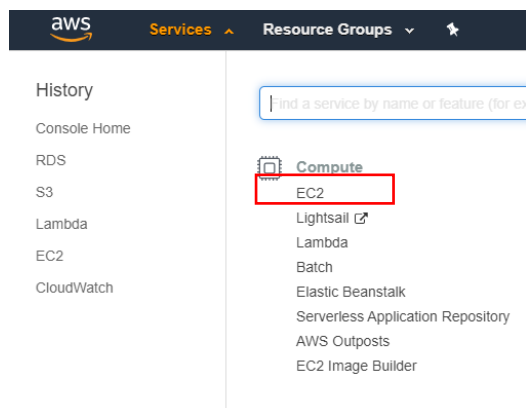


### III. Amazon Elastic Compute Cloud – EC2

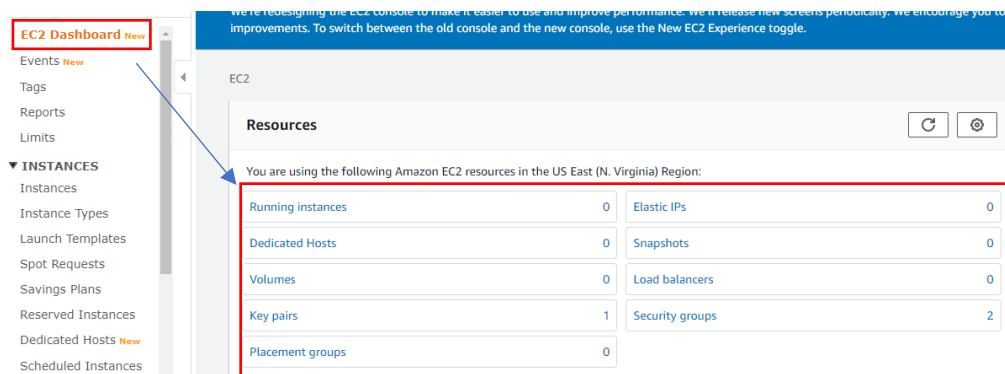
Amazon Elastic Compute Cloud (Amazon EC2) is a cloud-based computing service that offers stable, resizable compute functionality. It is planned to make cloud computing on a network scale simpler for users. It reduces the time taken to get and boot new server instances to minutes, helping you to rapidly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

## Launch instance

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

**Launch instance** ▼

Note: Your instances will launch in the US East (N. Virginia) Region

- 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.

### Step 1: Choose an Amazon Machine Image (AMI)

[Cancel and Exit](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Q Search for an AMI by entering a search term e.g. "Windows" X

Quick Start

My AMIs

AWS Marketplace

Community AMIs

**Amazon Linux 2 AMI (HVM), SSD Volume Type** - ami-0323c3dd2da7fb37d (64-bit x86) / ami-0ce2e5b7d27317779 (64-bit Arm)

**Amazon Linux**  
Free tier eligible

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

**Select**

64-bit (x86)  
64-bit (Arm)

- 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.

### Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All instance types Current generation Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
<input type="checkbox"/>	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	General purpose	<b>t2.micro</b> Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.xlarge	4	16	EBS only	-	Moderate	Yes
<input type="checkbox"/>	General purpose	t2.2xlarge	8	32	EBS only	-	Moderate	Yes

[Cancel](#) [Previous](#) [Review and Launch](#) **Next: Configure Instance Details**

- 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'

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

### Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances	1	Launch into Auto Scaling Group
Purchasing option	<input type="checkbox"/> Request Spot instances	
Network	vpc-f1d2c85 (default)	Create new VPC
Subnet	No preference (default subnet in any Availability Zone)	Create new subnet
Auto-assign Public IP	Use subnet setting (Enable)	
Placement group	<input type="checkbox"/> Add instance to placement group	
Capacity Reservation	Open	Create new Capacity Reservation
IAM role	S3_Admin_Access	Create new IAM role
Shutdown behavior	Stop	
Stop - Hibernate behavior	<input type="checkbox"/> Enable hibernation as an additional stop behavior	
Enable termination protection	<input type="checkbox"/> Protect against accidental termination	

Cancel Previous **Review and Launch** 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'

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

### Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/xvda	snap-0e1167baa50e9c0ff	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted

Add New Volume

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

Cancel Previous **Review and Launch** **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’

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

### Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver.  
A copy of a tag can be applied to volumes, instances or both.  
Tags will be applied to all instances and volumes. [Learn more](#) about tagging your Amazon EC2 resources.

Key (128 characters maximum)	Value (256 characters maximum)	Instances (i)	Volumes (i)
Name	CloudCruise WebServer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

[Add another tag](#) (Up to 50 tags maximum)

[Cancel](#) [Previous](#) [Review and Launch](#) [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.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

### Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: ☐ Create a new security group ☒ Select an existing security group

Security Group ID	Name	Description	Actions
<input type="checkbox"/> sg-86da9bad	default	default VPC security group	<a href="#">Copy to new</a>
<input checked="" type="checkbox"/> sg-0ae1efb0c90969891	WebDMZ	WebDMZ	<a href="#">Copy to new</a>

Inbound rules for sg-0ae1efb0c90969891 (Selected security groups: sg-0ae1efb0c90969891)

Type (i)	Protocol (i)	Port Range (i)	Source (i)	Description (i)
HTTP	TCP	80	0.0.0.0/0	
HTTP	TCP	80	:::0	
SSH	TCP	22	0.0.0.0/0	

[Cancel](#) [Previous](#) [Review and Launch](#)

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

## Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

**⚠ Improve your instances' security. Your security group, WebDMZ, is open to the world.**  
Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only.  
You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. [Edit security groups](#)

### AMI Details

[Edit AMI](#)

**Free tier eligible** **Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0323c3dd2da7fb37d**  
Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras.  
Root Device Type: ebs Virtualization type: hvm

### Instance Type

[Edit instance type](#)

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	Variable	1	1	EBS only	-	Low to Moderate

### Security Groups

[Edit security groups](#)[Cancel](#) [Previous](#) [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.

### Select an existing key pair or create a new key pair



A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Create a new key pair

Key pair name

Download Key Pair

After downloading the key pair we can launch the instance.

### Select an existing key pair or create a new key pair



A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Choose an existing key pair

Select a key pair

MyUSE1KP

☒ I acknowledge that I have access to the selected private key file (MyUSE1KP.pem), and that without this file, I won't be able to log into my instance.

[Cancel](#) [Launch Instances](#)

We can see launch status and launch log as shown below

Launch Status

Initiating Instance Launches  
Please do not close your browser while this is loading  
Creating security groups... Successful  
Authorizing inbound rules...

Launch Status

✓ Your instances are now launching  
The following instance launches have been initiated: i-04a28651232d32e5 View launch log

Launch Instance Connect Actions

Filter by tags and attributes or search by keyword

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP
	i-0ad3c9459f0d45838	t2.micro	us-east-1b	running	2/2 checks passed	None	ec2-18-212-69-128.co...	18.212.69...

Instance: i-0ad3c9459f0d45838 Public DNS: ec2-18-212-69-128.compute-1.amazonaws.com

Description Status Checks Monitoring Tags

Instance ID: i-0ad3c9459f0d45838  
Instance state: running  
Instance type: t2.micro  
Finding: Opt-in to AWS Compute Optimizer for recommendations. Learn more  
Private DNS: ip-172-31-92-121.ec2.internal  
Private IPs: 172.31.92.121  
Secondary private IPs:  
VPC ID: vpc-ff1d2c85  
Subnet ID: subnet-eaad4ecb  
Network interfaces: eth0  
IAM role: S3 Admin Access

Public DNS (IPv4): ec2-18-212-69-128.compute-1.amazonaws.com  
IPv4 Public IP: 18.212.69.128  
IPv6 IPs: -  
Elastic IPs:  
Availability zone: us-east-1b  
Security groups: WebDMZ. view inbound rules. view outbound rules  
Scheduled events: No scheduled events  
AMI ID: amzn2-ami-hvm-2.0.20200406.0-x86\_64-gp2 (ami-0323c3dd2da7fb37d)  
Platform details: -  
Usage operation: -  
Source/dest check: True

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'

Secure Shell App | chrome-extension://pnhechapfaiddjhompbfnfcldbbghjo/html/nassh.html

Welcome to Secure Shell App version 0.28.  
Answers to Frequently Asked Questions: <https://goo.gl/mupppj> (ctrl+click on links to open)  
[Pro Tip] Use 'Open as Window' or 'Fullscreen' to prevent Ctrl+W from closing your terminal!  
[Pro Tip] See <https://goo.gl/mupppj> for more information.  
ChangeLog/release notes: </html/changelog.html>  
Random Pro Tip #10: When running in a window, Ctrl+Shift+N can open another session.

[New Connection]

ec2-user@18.212.69.128

ec2-user@18.212.69.128

SSH relay server options

Identity: MyKP Import...

SSH Arguments: extra command line arguments

Current profile: default

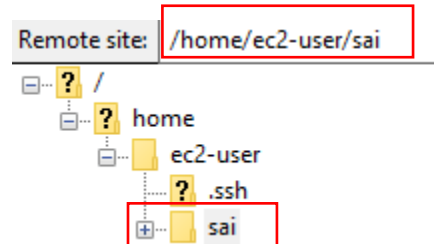
[Delete] Options [SFTP] [ENTER] Connect



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

Filename	Filesize	Filetype	Last modified	Filename	Filesize	Filetype	Last modified	Permissions	Owner/Group
..				..					
.ipynb_checkpoints		File folder	4/16/2020 3:59:51 ...	.ssh		File folder	4/16/2020 8:25:...	drwx-----	ec2-user ec...
code from ec2		File folder	3/29/2020 4:15:34 ...	.bash_logout	18	BASH_LOG...	1/15/2020 6:56:...	-rw-r--r--	ec2-user ec...
sai		File folder	3/29/2020 4:15:27 ...	.bash_profile	193	BASH_PRO...	1/15/2020 6:56:...	-rw-r--r--	ec2-user ec...
Tried Code		File folder	3/29/2020 4:25:37 ...	.bashrc	231	BASHRC File	1/15/2020 6:56:...	-rw-r--r--	ec2-user ec...
Frustricious_CHNNModel...	45,466	IPYNB File	4/16/2020 4:00:18 ...						

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)

```

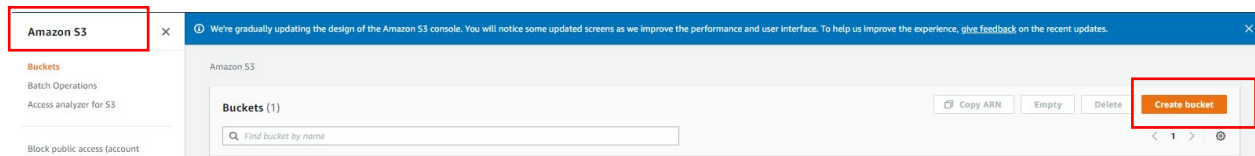
Finally, we can see the libraries installed using the following command and verify the needed libraries and highlighted.

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

```
absl-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-instance-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
jsonpinter==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
pyparsing==2.14
pycurl==7.19.0
pygments==0.3
pylibzma==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 “clouducrise” bucket in S3 folder with public access and then choose “Create bucket”

**Bucket name**  
clouducrise  
Bucket name must be unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#)

**Region**  
US East (N. Virginia) us-east-1

**Bucket settings for Block Public Access**  
Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

☐ **Block all public access**  
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

☐ **Block public access to buckets and objects granted through new access control lists (ACLs)**  
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

☐ **Block public access to buckets and objects granted through any access control lists (ACLs)**  
S3 will ignore all ACLs that grant public access to buckets and objects.

☐ **Block public access to buckets and objects granted through new public bucket or access point policies**  
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

☐ **Block public and cross-account access to buckets and objects through any public bucket or access point policies**  
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

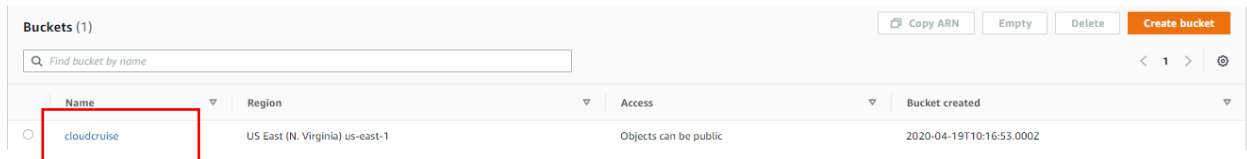
**Warning:** Turning off block all public access might result in this bucket and the objects within becoming public. AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

☐ I acknowledge that the current settings might result in this bucket and the objects within becoming public.

**Advanced settings**

Cancel **Create bucket**

Thus, we have created “cloudcruise” bucket

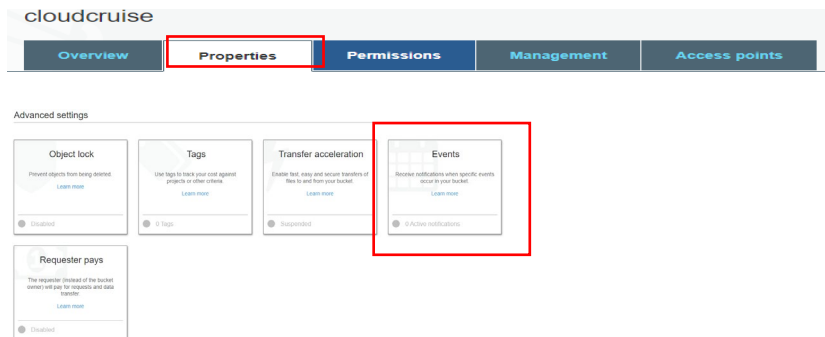


The screenshot shows the AWS S3 Buckets console. At the top, there's a 'Buckets (1)' header with buttons for 'Copy ARN', 'Empty', 'Delete', and 'Create bucket'. Below this is a search bar labeled 'Find bucket by name'. A table lists the bucket 'cloudcruise' with columns for Name, Region (US East (N. Virginia) us-east-1), Access (Objects can be public), and Bucket created (2020-04-19T10:16:53.000Z). The 'cloudcruise' name is highlighted with a red box.

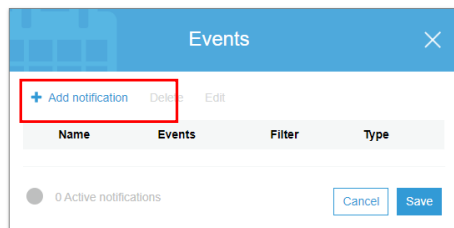
Name	Region	Access	Bucket created
cloudcruise	US East (N. Virginia) us-east-1	Objects can be public	2020-04-19T10:16:53.000Z

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.

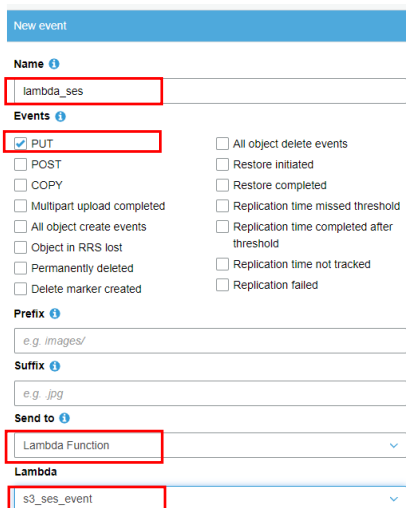
Choose ‘Properties’ tab and select ‘Events’ under Advanced settings.



Choose Events and click on ‘Add Notification’



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



The screenshot shows the 'New event' configuration form. The 'Name' field is 'lambda\_ses' (highlighted with a red box). The 'Events' section has 'PUT' checked (highlighted with a red box). The 'Prefix' field is 'e.g. images/'. The 'Suffix' field is 'e.g. .jpg'. The 'Send to' dropdown is 'Lambda Function' (highlighted with a red box). The 'Lambda' dropdown is 's3\_ses\_event' (highlighted with a red box).

**New event**

**Name** ⓘ  
lambda\_ses

**Events** ⓘ

☒ PUT

☐ POST

☐ COPY

☐ Multipart upload completed

☐ All object create events

☐ Object in RRS lost

☐ Permanently deleted

☐ Delete marker created

☐ All object delete events

☐ Restore initiated

☐ Restore completed

☐ Replication time missed threshold

☐ Replication time completed after threshold

☐ Replication time not tracked

☐ Replication failed

**Prefix** ⓘ  
e.g. images/

**Suffix** ⓘ  
e.g. .jpg

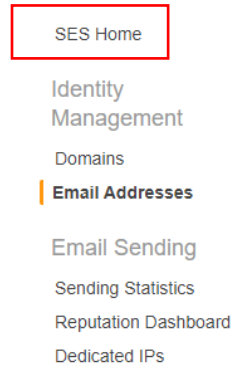
**Send to** ⓘ  
Lambda Function

**Lambda**  
s3\_ses\_event

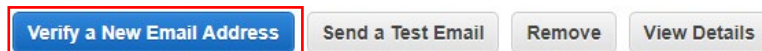


## V. Amazon Simple Email Service – SES

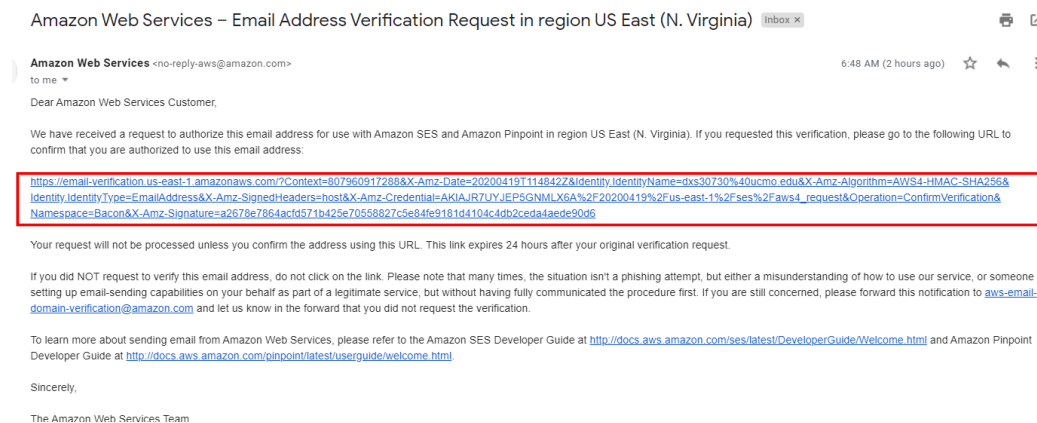
Initially, we need to make sure that we verify the email addresses that we will be using .Choose SES from the services and select ‘Email Addresses’






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



We will get confirmation email like the one below



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

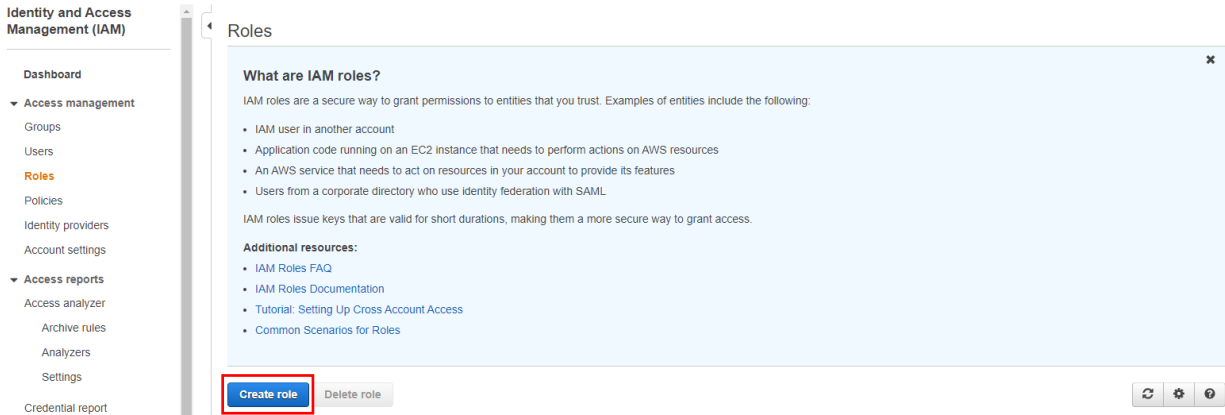
		deepi4u@gmail.com	verified
		dxs30730@ucmo.edu	verified

## VI. Identity Access Management - IAM

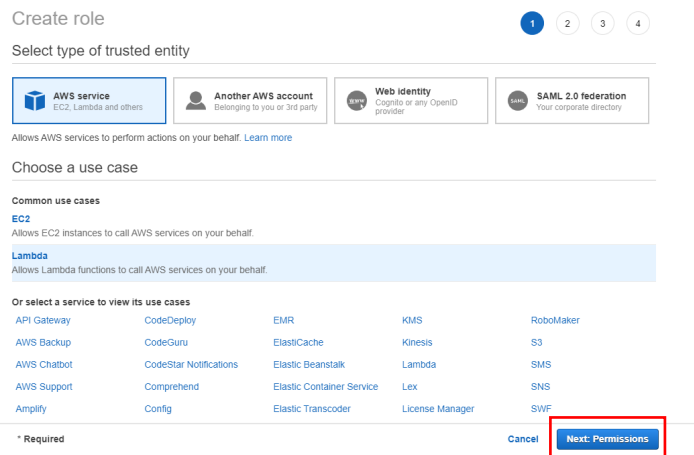
Firstly, we will create IAM role to have access to Lambda and SES. So choose IAM from the list of EC2 Services.



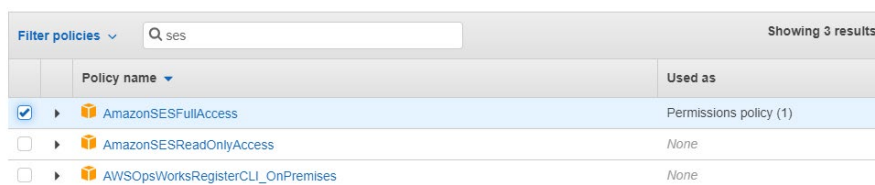
Secondly, in IAM dashboard click 'Create role'



Choose Lambda and click on 'Next: Permissions'



Choose appropriate policy as shown below and then create the role called 'lambda\_se3\_ses'



Filter policies  Showing 18 results

	Policy name	Used as
<input type="checkbox"/>	AWSDeepLensLambdaFunctionAccessPolicy	None
<input type="checkbox"/>	AWSLambdaBasicExecutionRole	None
<input type="checkbox"/>	AWSLambdaBasicExecutionRole-ad5c379a-b44a-4e36-8b88-71e625b5d60c	Permissions policy (1)
<input type="checkbox"/>	AWSLambdaDynamoDBExecutionRole	None
<input type="checkbox"/>	AWSLambdaENIManagementAccess	None
<input checked="" type="checkbox"/>	AWSLambdaExecute	Permissions policy (1)
<input type="checkbox"/>	AWSLambdaFullAccess	None
<input type="checkbox"/>	AWSLambdaInvocationDynamoDB	None

Role name\*

Policies AmazonSESFullAccess [↗](#)  
 AWSLambdaExecute [↗](#)

Permissions boundary Permissions boundary is not set

[Cancel](#) [Previous](#) [Create role](#)

We have also created S3\_Admin\_Access role by following the steps as shown above

## Summary

[Delete role](#)

**Role ARN** `arn:aws:iam::807960917288:role/S3_Admin_Access` [↗](#)  
**Role description** Allows EC2 instances to call AWS services on your behalf. | [Edit](#)  
**Instance Profile ARNs** `arn:aws:iam::807960917288:instance-profile/S3_Admin_Access` [↗](#)  
**Path** /  
**Creation time** 2020-02-18 04:56 CDT  
**Last activity** 2020-04-19 08:09 CDT (Today)  
**Maximum CLI/API session duration** 1 hour [Edit](#)

[Permissions](#) [Trust relationships](#) [Tags](#) [Access Advisor](#) [Revoke sessions](#)

▼ Permissions policies (2 policies applied)

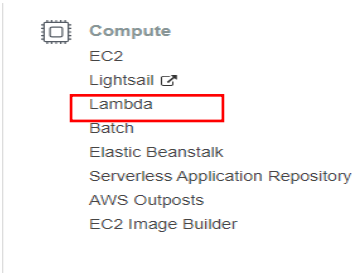
[Attach policies](#)

[Add inline policy](#)

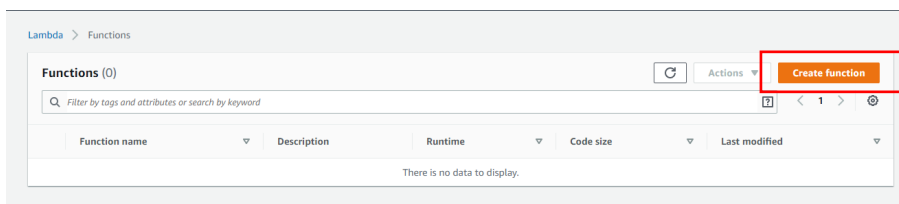
	Policy name	Policy type	
▶	AmazonS3FullAccess	AWS managed policy	✕
▶	CloudWatchFullAccess	AWS managed policy	✕

## VII. AWS Lambda

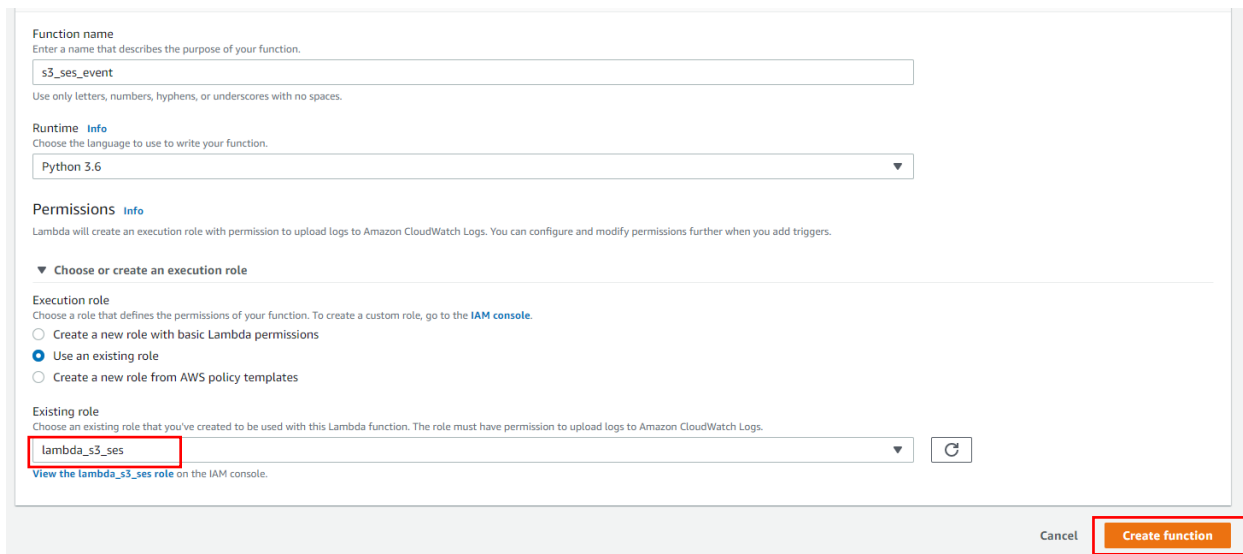
Select Lambda from Compute services



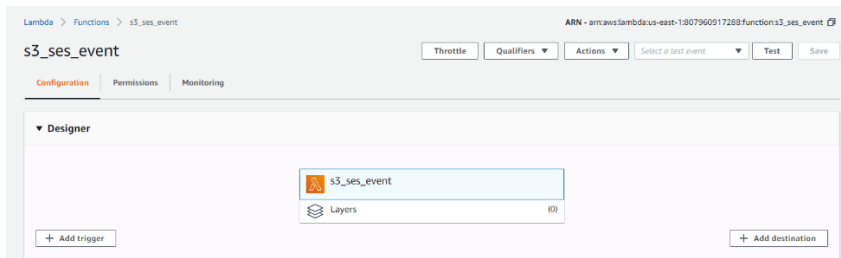
Click on 'Create Function'



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'

s3\_ses\_event Throttle Qualifiers ▼ Actions ▼ Select a test event ▼ Test Save

**Function code** [Info](#)

Code entry type: Edit code inline Runtime: Python 3.6 Handler: lambda\_function.lambda\_handler

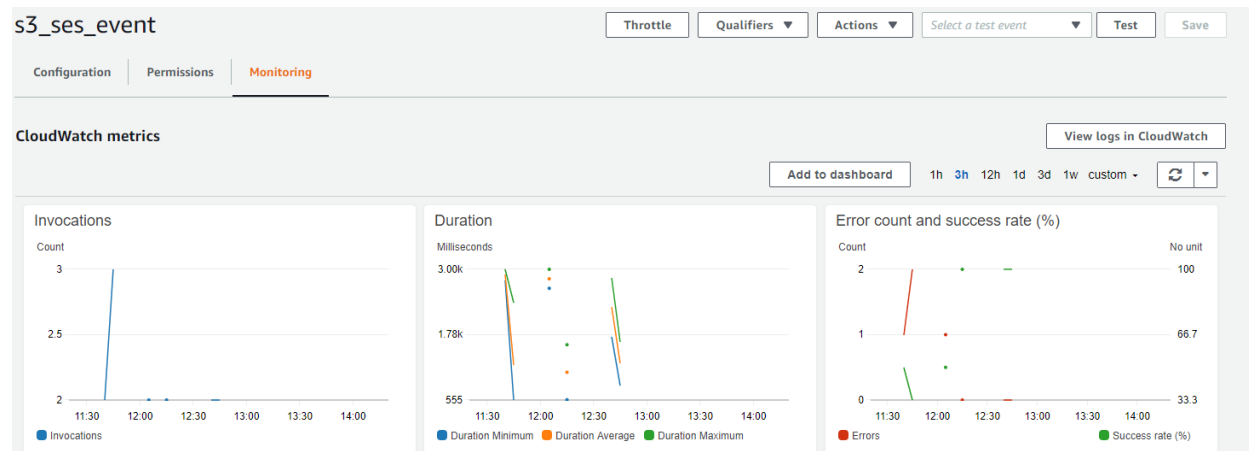
```

1 import boto3
2 from email.mime.multipart import MIMEMultipart
3 from email.mime.text import MIMEText
4 from email.mime.application import MIMEApplication
5
6 def lambda_handler(event, context):
7     # TODO: implement
8     ses = boto3.client("ses")
9     s3 = boto3.client("s3")
10
11     for i in event["Records"]:
12         actions=i["eventName"]
13         ip=i["requestParameters"]["sourceIPAddress"]
14         bucket_name=i["s3"]["bucket"]["name"]
15         object=i["s3"]["object"]["key"]
16
17         fileObj = s3.get_object(Bucket = bucket_name, Key = object)
18         file_content = fileObj["Body"].read()
19
20         sender = "deepika@gmail.com"
21         to = "dks30730@ucmo.edu"
22
23         subject=str(actions)+ ' ' + object + ' in ' + 'S3 Bucket: '+bucket_name
24
25         body=""
26         <br>
27         This email is to notify that the {} is used for Image Classification <br> Source IP:{}
28         <br>
29         <br>
30         msg = MIMEMultipart()
31         msg["Subject"] = subject
32         msg["From"] = sender
33         msg["To"] = to
34
35         body_txt = MIMEText(body, "html")
36
37         attachment = MIMEApplication(file_content)
38         attachment.add_header("Content-Disposition", "attachment", filename=object)
39

```

21:28 Python Spaces: 4

We can make use of cloud metrics for monitoring purpose



Logs are also updated in CloudWatch for debugging purpose.

CloudWatch > Log Groups > Streams for /aws/lambda/s3\_ses\_event

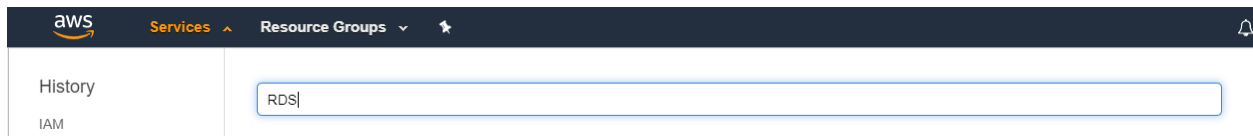
Search Log Group Create Log Stream Delete Log Stream

Filter:  x

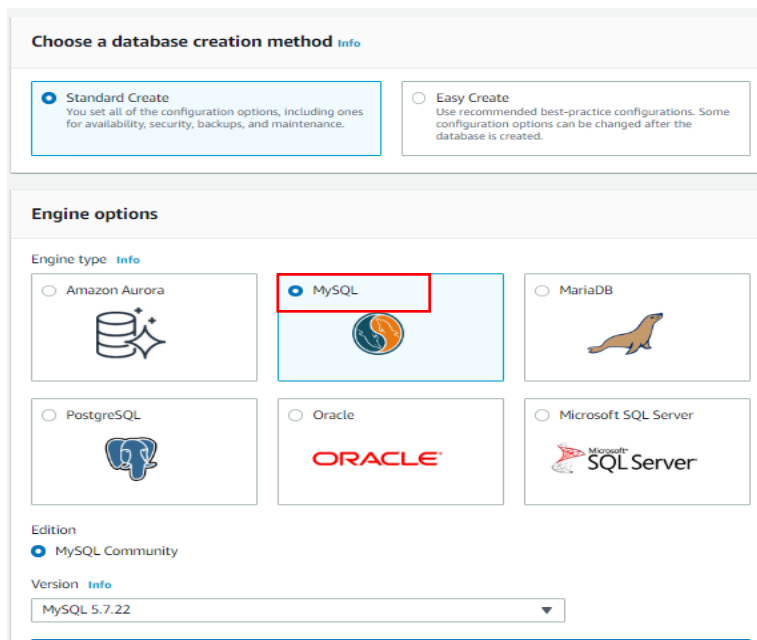
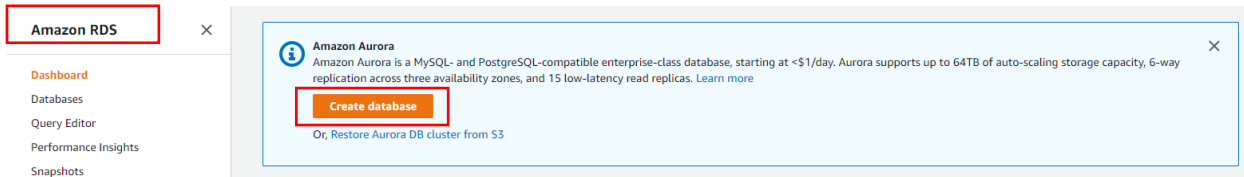
Log Streams	Last Event Time
2020/04/19/[\$LATEST]c04ce824145f448ab22e165980ec5201	2020-04-19 07:47 UTC-5
2020/04/19/[\$LATEST]223d542a97a54b3ba31e6fcdcb185d8	2020-04-19 07:17 UTC-5
2020/04/19/[\$LATEST]d81f361f35e5464c85514b62b7ae4b7d	2020-04-19 06:49 UTC-5
2020/04/19/[\$LATEST]c2b321121a0d45bc9d2a690e792a884b	2020-04-19 06:44 UTC-5

## VIII. Amazon Relational Database Service

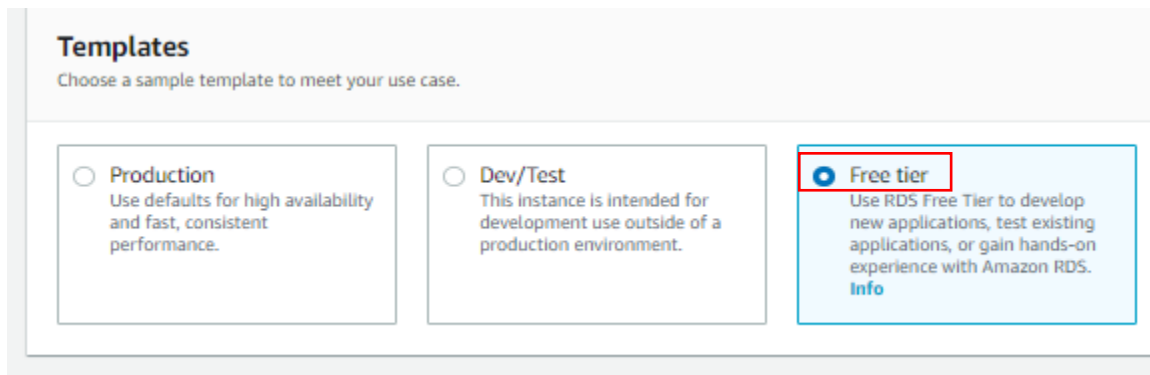
Choose RDS from Services



Click on 'Create Database' and choose 'MySQL'



Select 'Free Tier' in Templates section



Provide 'mysqldb2020' in identifier name,username and password

**Settings**

**DB instance identifier** [Info](#)  
Type a name for your DB instance. The name must be unique cross all DB instances owned by your AWS account in the current AWS Region.

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens (1 to 15 for SQL Server). First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

**▼ Credentials Settings**

**Master username** [Info](#)  
Type a login ID for the master user of your DB instance.

1 to 16 alphanumeric characters. First character must be a letter

☐ Auto generate a password  
Amazon RDS can generate a password for you, or you can specify your own password

**Master password** [Info](#)

Constraints: At least 8 printable ASCII characters. Can't contain any of the following: / (slash), " (double quote) and @ (at sign).

**Confirm password** [Info](#)

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

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IPs	Key Name	Monitoring	Launch Time
	i-0a03c945f910c45038	t2.micro	us-east-1b	running	2/2 checks passed	None	ec2-18-212-88-128.co...	18.212.88.128	-	MyUSE1KP	disabled	April 18, 2020

**▼ Additional connectivity configuration**

**Subnet group** [Info](#)  
DB subnet group that defines which subnets and IP ranges the DB instance can use in the VPC you selected.

**Publicly accessible** [Info](#)

☒ Yes  
Amazon EC2 instances and devices outside the VPC can connect to your database. Choose one or more VPC security groups that specify which EC2 instances and devices inside the VPC can connect to the database.

☐ No  
RDS will not assign a public IP address to the database. Only Amazon EC2 instances and devices inside the VPC can connect to your database.

**VPC security group**  
Choose one or more RDS security groups to allow access to your database. Ensure that the security group rules allow incoming traffic from EC2 instances and devices outside your VPC. (Security groups are required for publicly accessible databases.)

☒ Choose existing  
Choose existing VPC security groups

☐ Create new  
Create new VPC security group

**Existing VPC security groups**

**Availability zone** [Info](#)

**Database port** [Info](#)  
TCP/IP port the database will use for application connections.

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

Inbound rules				
Type	Protocol	Port range	Source	Description - optional
HTTP	TCP	80	0.0.0.0/0	-
HTTP	TCP	80	*/0	-
SSH	TCP	22	0.0.0.0/0	-
MySQL/Aurora	TCP	3306	0.0.0.0/0	-

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 'mysqladb2020' to get more details

RDS > Databases	
<div> Databases Group resources Modify Actions Restore from S3 Create database </div>	
<div> Filter databases </div>	
<div> DB identifier Role Engine Region &amp; AZ Size Status </div>	
<div> mysqladb2020 Instance MySQL Community us-east-1b db.t2.micro Available </div>	

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

RDS > Databases > mysqladb2020	
mysqladb2020	
<div> Summary </div>	
<div> <div> DB identifier mysqladb2020 </div> <div> CPU 1.97% </div> <div> Role Instance </div> </div>	
<div> <div> Info Available </div> <div> Engine MySQL Community </div> </div>	
<div> Class db.t2.micro </div>	
<div> Region &amp; AZ us-east-1b </div>	
<div> Connectivity &amp; security Monitoring Logs &amp; events Configuration Maintenance &amp; backups Tags </div>	
<div> Connectivity &amp; security </div>	
<div> Endpoint &amp; port </div>	
<div> <div> Endpoint mysqladb2020.c8zaxnpj5b57.us-east-1.rds.amazonaws.com </div> <div> Port 3306 </div> </div>	
<div> Networking </div>	
<div> <div> Availability zone us-east-1b </div> <div> VPC vpc-f1d2c85 </div> <div> Subnet group default-vpc-f1d2c85 </div> <div> Subnets subnet-2bd3364d subnet-697f5757 subnet-eaad4ecb subnet-484ea817 subnet-0f00b542 subnet-2320ff2d </div> </div>	
<div> Security </div>	
<div> <div> VPC security groups WebDM42 (sg-0ae1efb0c90969891) (active) default (sg-86da9bad) (active) </div> <div> Public accessibility Yes </div> <div> Certificate authority rds-ca-2019 </div> <div> Certificate authority date Aug 22nd, 2024 </div> </div>	

We update the RDS endpoint in our main code 'fruit\_status\_flask\_ec2\_s3\_rds.py' provided in Appendix

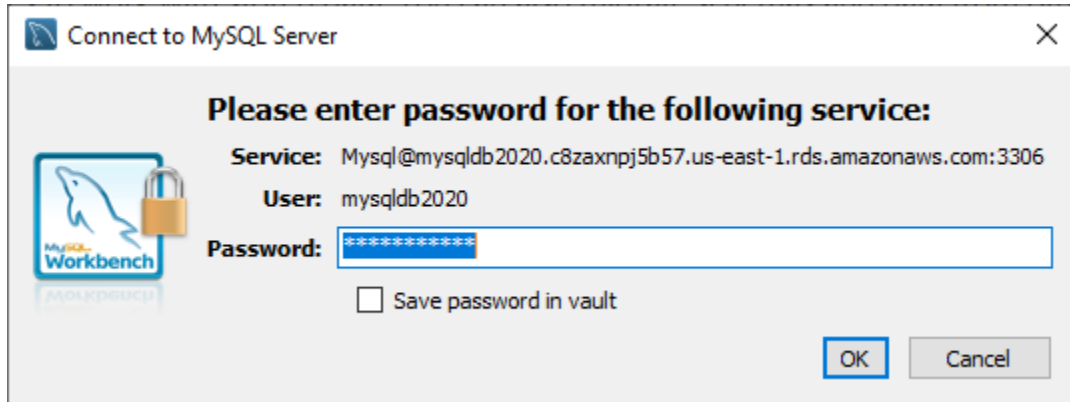
```
#Mysql connection
mydb = mysql.connector.connect(
    host="mysqladb2020.c8zaxnpj5b57.us-east-1.rds.amazonaws.com",
    user="mysqladb2020",
    passwd="mysqladb2020",
    database="cloudcruise"
)
```



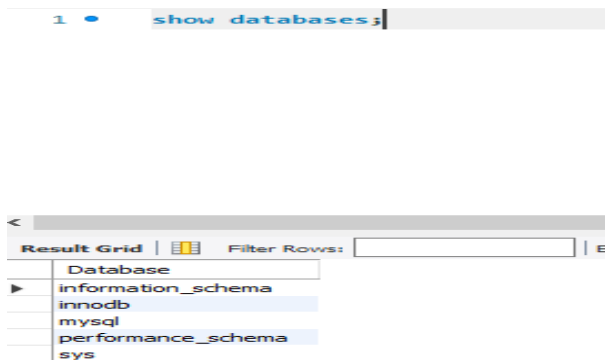
## SQL Workbench

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

Additionally enter database name and password.



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



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
[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;
```

The screenshot shows the SQL Workbench interface. On the left, the 'Database' list includes 'information\_schema', 'cloudcruise' (selected), 'innodb', 'mysql', 'performance\_schema', and 'sys'. In the center, the 'Tables\_in\_cloudcruise' window displays a table named 'logs'. At the bottom, the 'Table' window shows the SQL command to create the 'logs' table: `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
```

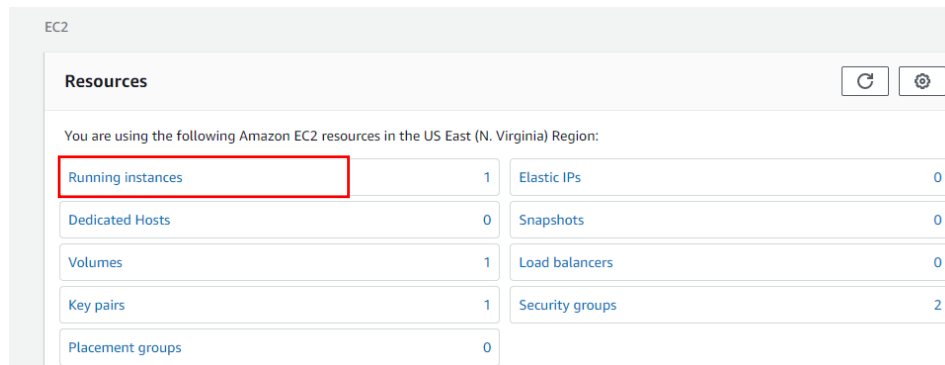
FruitStatus\_cnn\_0.99.h5 is the model we created for Image classification to differentiate fresh and rotten apples.

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)
```

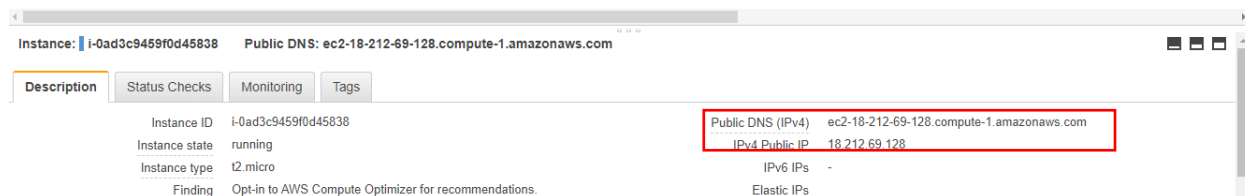
After we run this code, we can go to EC2 dashboard and click on 'Running Instance'



Resources	
You are using the following Amazon EC2 resources in the US East (N. Virginia) Region:	
Running instances	1
Elastic IPs	0
Dedicated Hosts	0
Snapshots	0
Volumes	1
Load balancers	0
Key pairs	1
Security groups	2
Placement groups	0

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

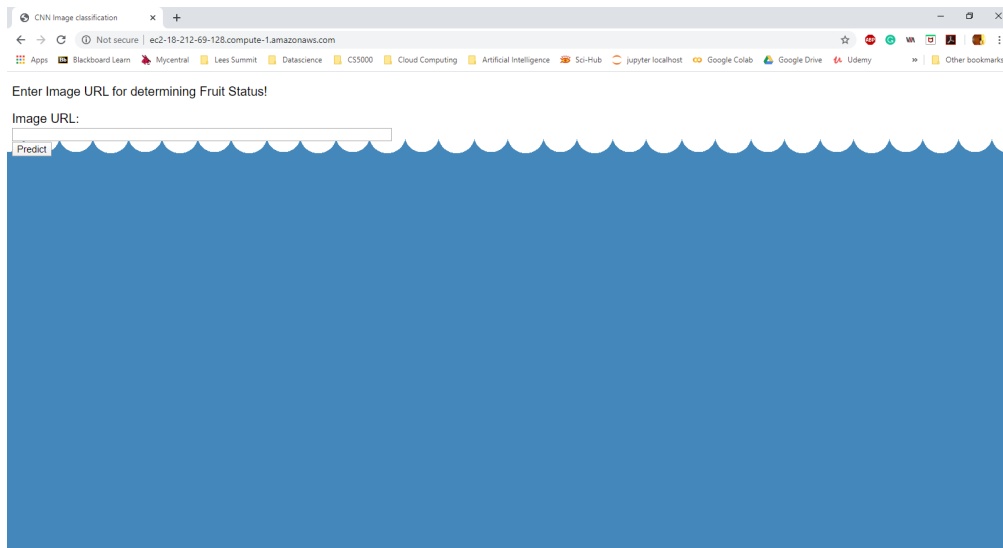
```
i-0ad3c9459f0d45838  t2.micro  us-east-1b  running  2/2 checks passed  None  ec2-18-212-69-128.co...  18.212.1
```



Instance: i-0ad3c9459f0d45838	
Instance ID	i-0ad3c9459f0d45838
Instance state	running
Instance type	t2.micro
Finding	Opt-in to AWS Compute Optimizer for recommendations.
Public DNS (IPv4)	ec2-18-212-69-128.compute-1.amazonaws.com
IPv4 Public IP	18.212.69.128
IPv6 IPs	-
Elastic IPs	

← → ↻

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'

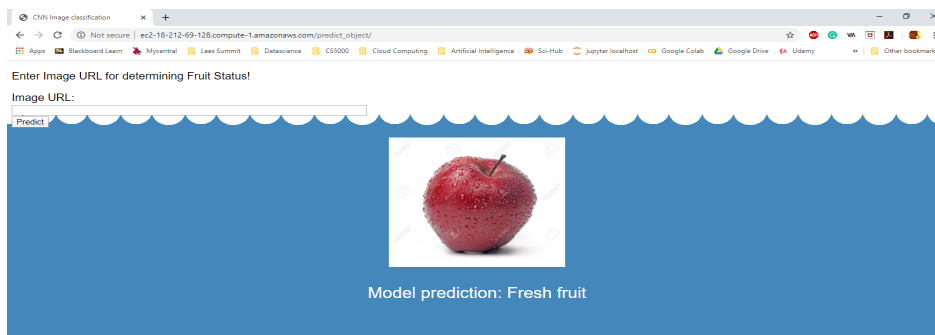
```
[root@ip-172-31-92-121 sai]# ls
db.py          fresh_internet_2.jpg  fruit_status_flask_ec2.py  fruit_status_flask_ec2_s3_rds.py  lambda_code.py  rotten_internet_1.jpg  static
fresh_internet_1.jpg  FruitStatus_cnn_0.99.h5  fruit_status_flask_ec2_s3.py  lambda_code_from_ec2.py  requirements.txt  rotten_internet_2.jpg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

Enter Image URL for determining Fruit Status!

Image URL:



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.

```

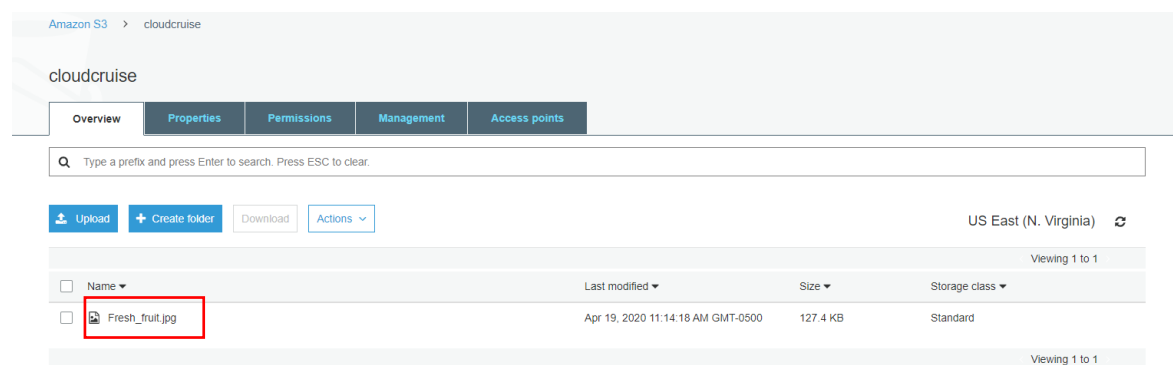
72.129.239.159 - - [19/Apr/2020 16:21:36] "GET / HTTP/1.1" 200 -
WARNING:tensorflow:From /root/.local/lib/python2.7/site-packages/keras/backend/tensorflow_backend.py:4070: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.

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 future 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.

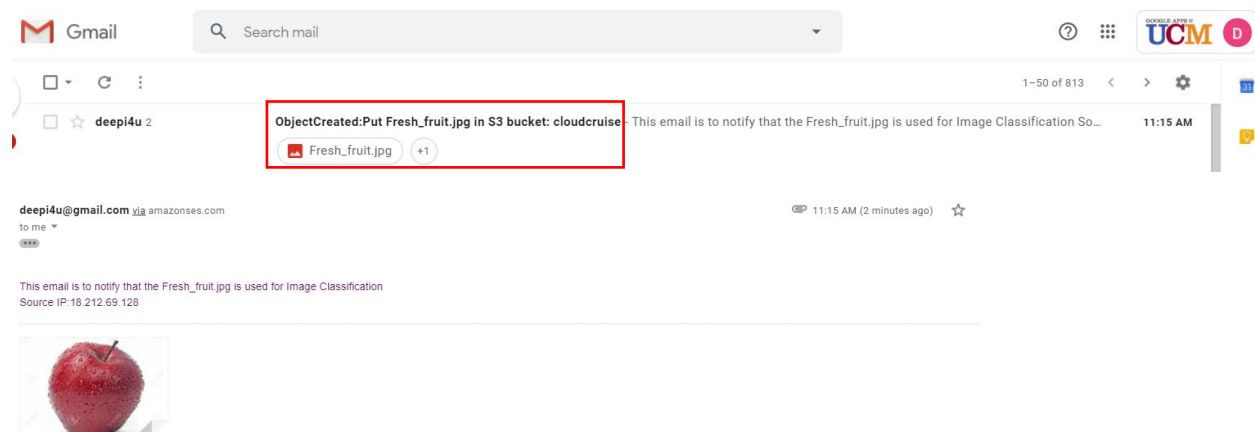
('image_url is:', u'fresh_internet_1.jpg')
('Classification-', 'Fresh fruit')
Fresh fruit
Python module executed successfully
(1, 'record inserted.')

```

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 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;

```

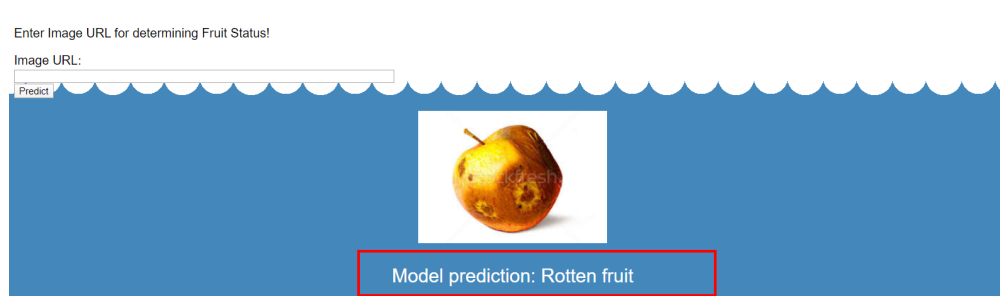
image_id	image_name	image_class
1	fresh_internet_1.jpg	Fresh fruit
NULL	NULL	NULL

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.

```
[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
```


Enter Image URL for determining Fruit Status!

Image URL:

Enter Image URL for determining Fruit Status!

Image URL:

Model prediction: Rotten fruit

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

```
72.129.239.159 - - [19/Apr/2020 16:22:05] "POST /predict_object/ HTTP/1.1" 200 -
('image_url is:', u'rotten_internet_1.jpg')
('Classification-', 'Rotten fruit')
Rotten fruit
Python module executed successfully
(1, 'record inserted.')
```

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

Amazon S3 > cloudcruise

cloudcruise

Overview Properties Permissions Management Access points

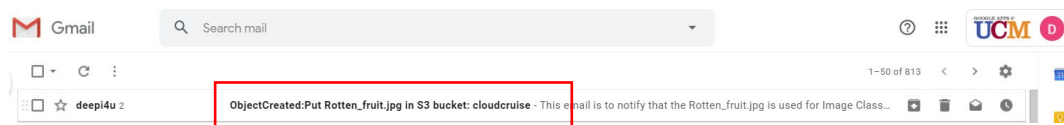
Q Type a prefix and press Enter to search. Press ESC to clear.

US East (N. Virginia)

Viewing 1 to 2

<input type="checkbox"/>	Name	Last modified	Size	Storage class
<input type="checkbox"/>	Fresh_fruit.jpg	Apr 19, 2020 11:22:06 AM GMT-0500	127.4 KB	Standard
<input type="checkbox"/>	Rotten_fruit.jpg	Apr 19, 2020 11:24:52 AM GMT-0500	6.9 KB	Standard

Rotten\_fruit.jpg triggered lambda function which will send email using Simple Email Service.



deepi4u@gmail.com via amazonses.com  
to me

11:24 AM (2 minutes ago)

This email is to notify that the Rotten\_fruit.jpg is used for Image Classification  
Source IP:18.212.69.128



Lets check the logs table for record updates

```
1 • select * from logs;
```

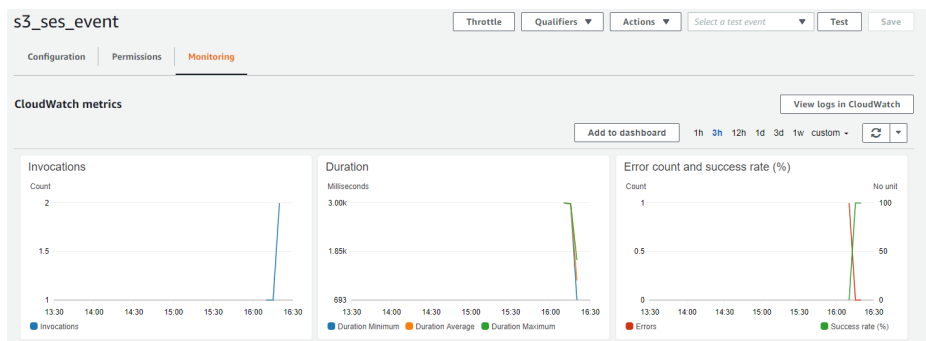
	image_id	image_name	image_class
▶	1	fresh_internet_1.jpg	Fresh fruit
	2	rotten_internet_1.jpg	Rotten fruit
•	NULL	NULL	NULL

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 classification, 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.