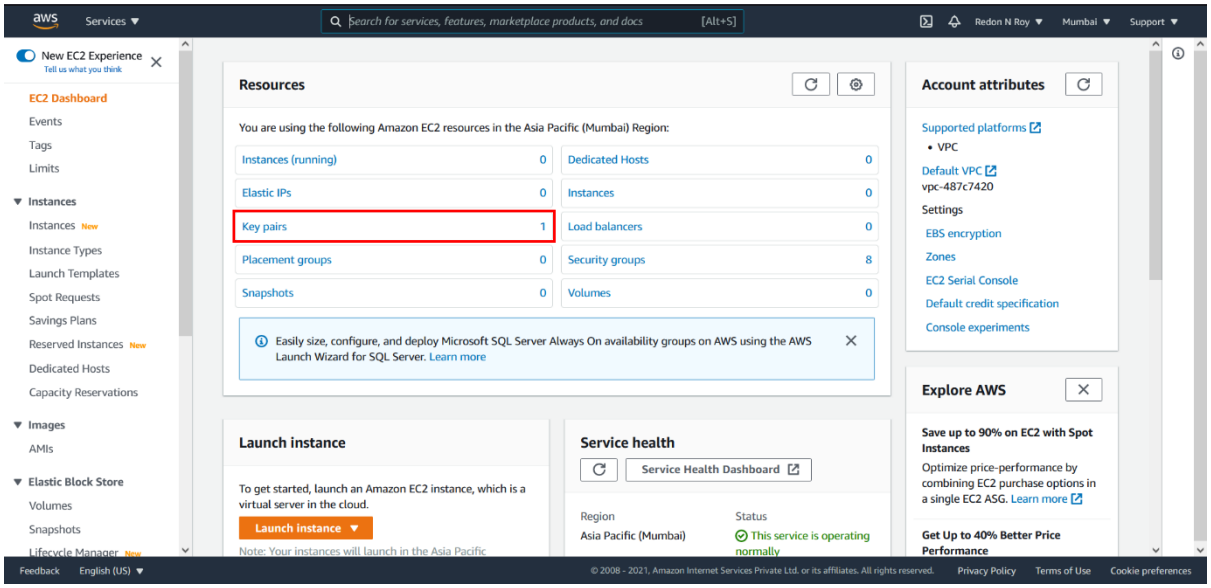
AWS EMR Demo

This file will help you to set up and perform hands-on with AWS EMR. We'll be creating the cluster and then run spark-submit in the EMR cluster.

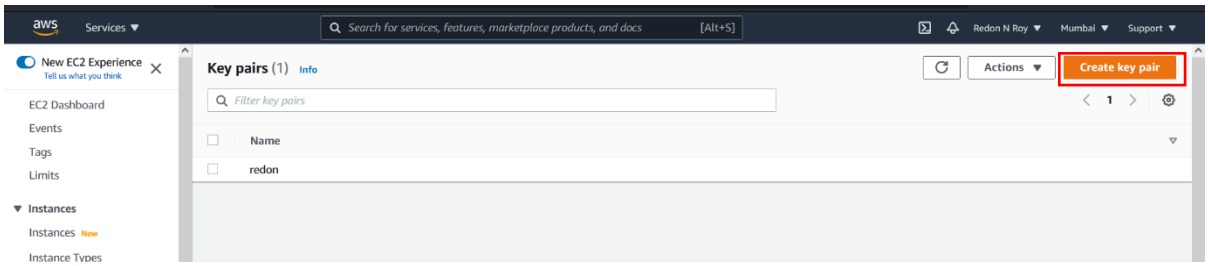
1. Creating key pair

Before starting with the creation of the cluster, we will be creating the key pair so that we can use it to access the cluster via SSH (Secure Shell). The steps to create a key pair are: -

a. In the **AWS dashboard**, search for "**EC2**" in the search bar and navigate to the EC2 dashboard.



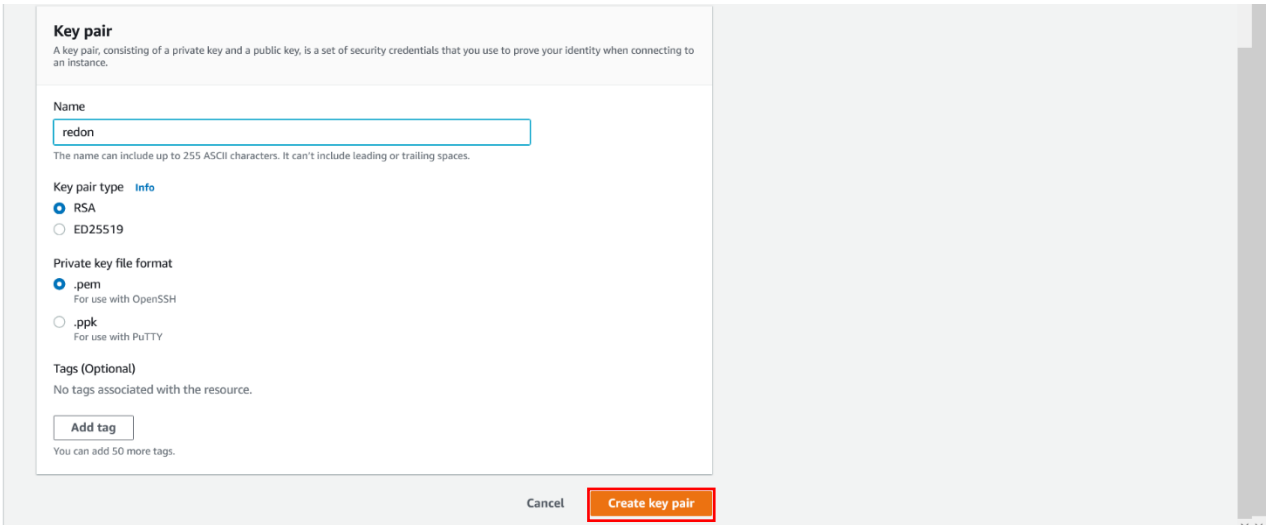
b. On the EC2 dashboard, choose the **key pairs** option under the **resources** section. Inside that, choose the **Create key pair** option on the top right corner.



c. Give a name to the key pair. Leave the type as **RSA**. The file format can be either **.pem** (Privacy Enhanced Mail) or **.ppk**.

If you are using **PuTTY**, then you must go with .ppk format or if you are using **OpenSSH**, then choose the .pem format.

For this demo, I'll be using OpenSSH (Instructions for PuTTY can be fount at the end of the document). Then click on **Create key pair**.

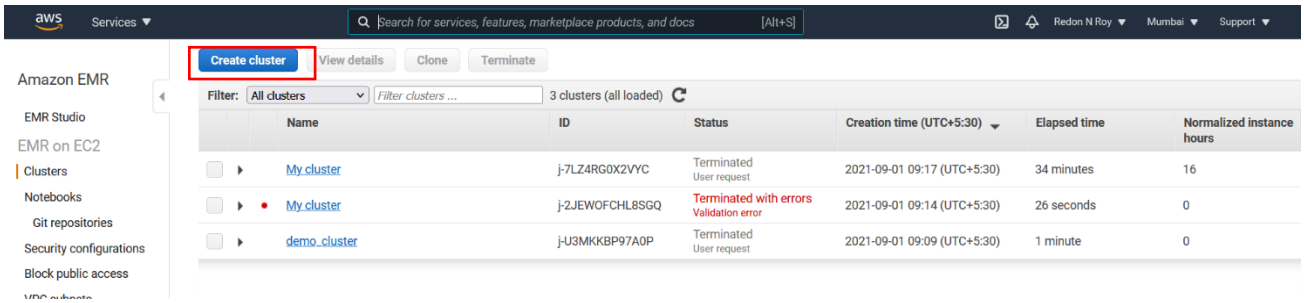


d. This will generate a key pair which we ill be using while cluster creation.

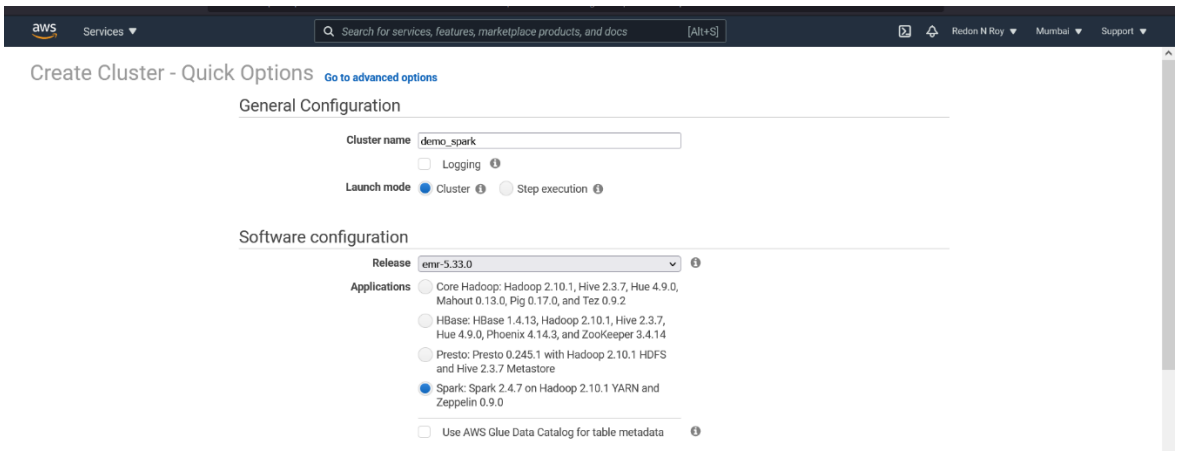
2. Creating EMR cluster

Now that we have the key pair setup, we can proceed with the cluster creation. The steps for that are:-

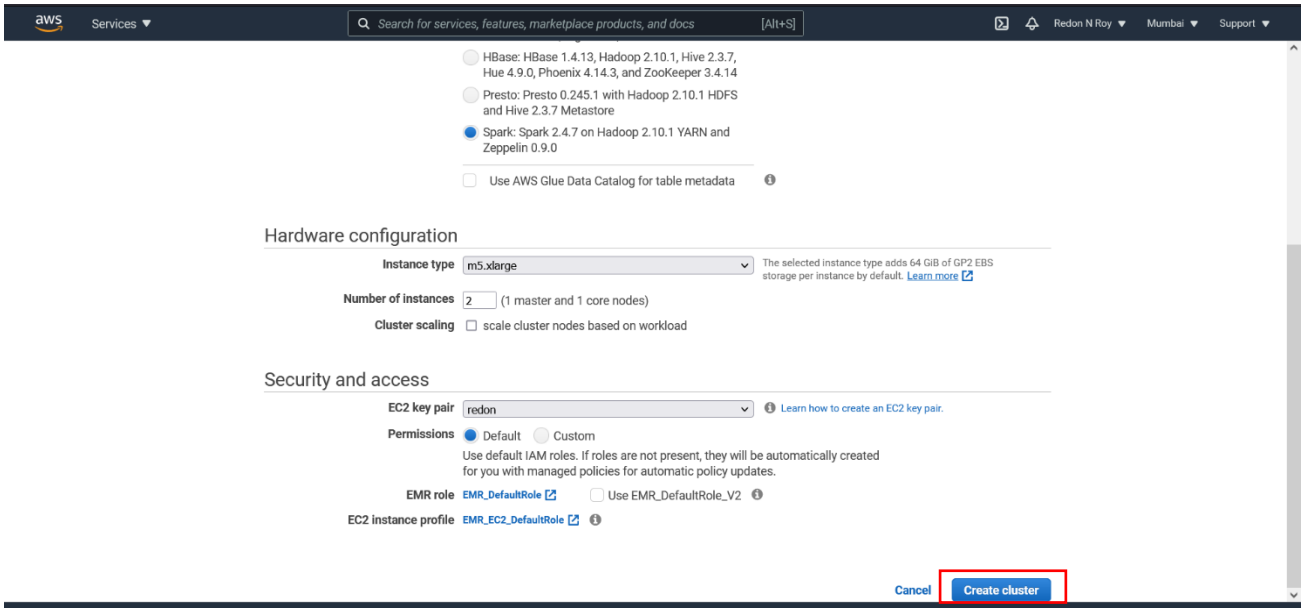
a. Go to the **EMR dashboard** and select the **Create cluster** option on the top left corner.



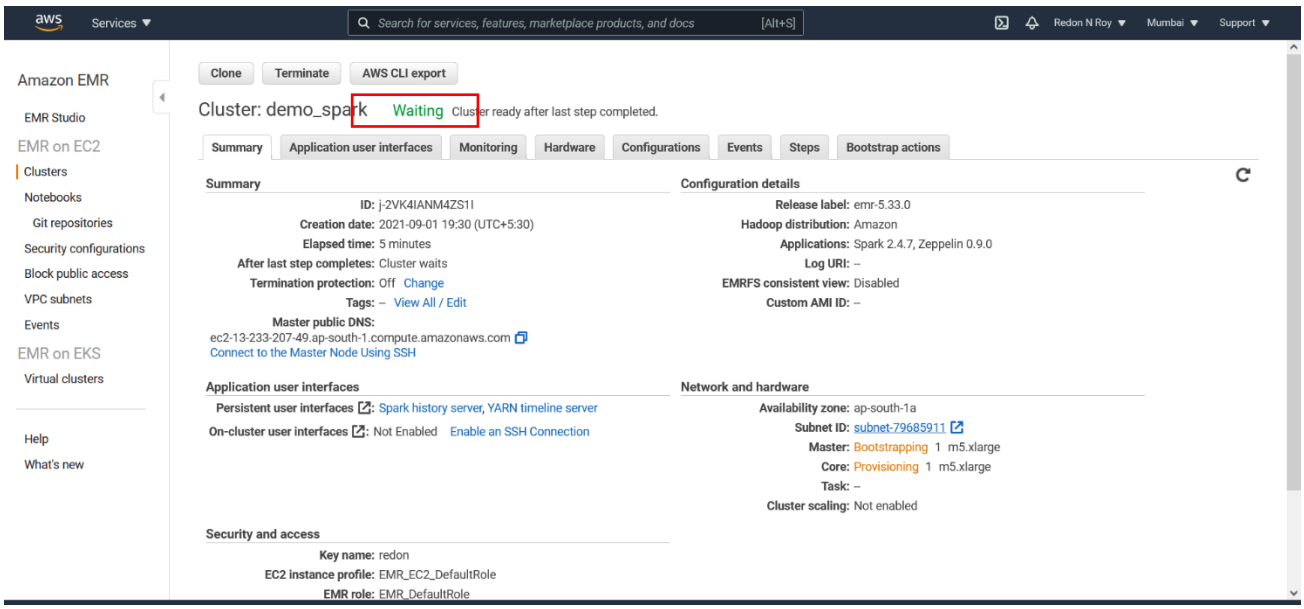
b. Now provide the **cluster name** for the EMR cluster. You can turn **off** the **Logging** option. Select the **Launch mode** as **Cluster**. The select the EMR configuration, here we'll be using the cluster for operations on **Spark**.



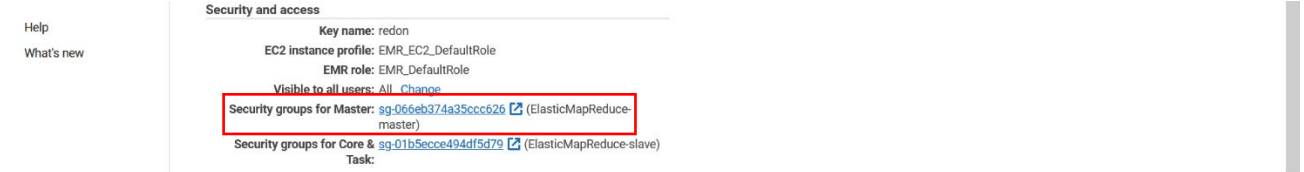
c. Select the **Instance type** as per the requirement. Here we'll go with the default one. Set the **number of instances** according to the need. For the demo I'll go with 2. The select the key pair that we have created earlier in the **EC2 key pair**. Leave everything else at default. Finally click on Create cluster.



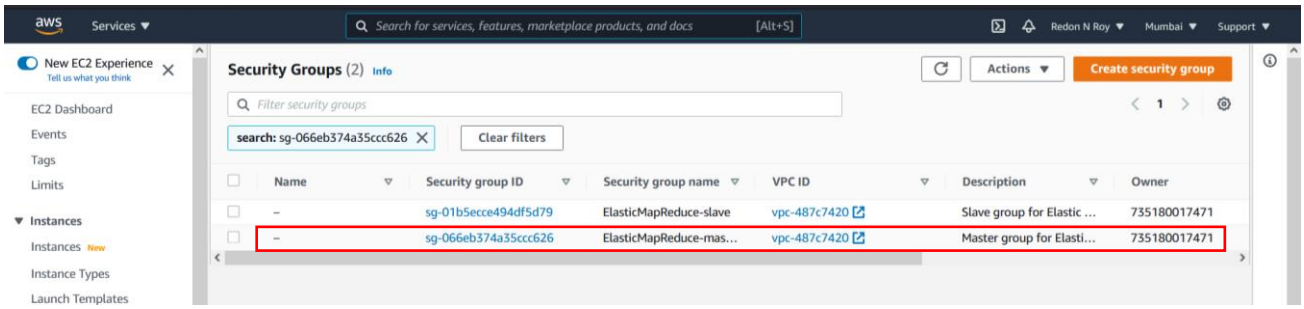
d. Now the cluster will be created and started. It will take a **few minutes** to get the cluster up and running. We will be able to use the cluster once the status changes from **Starting** to **Waiting**.



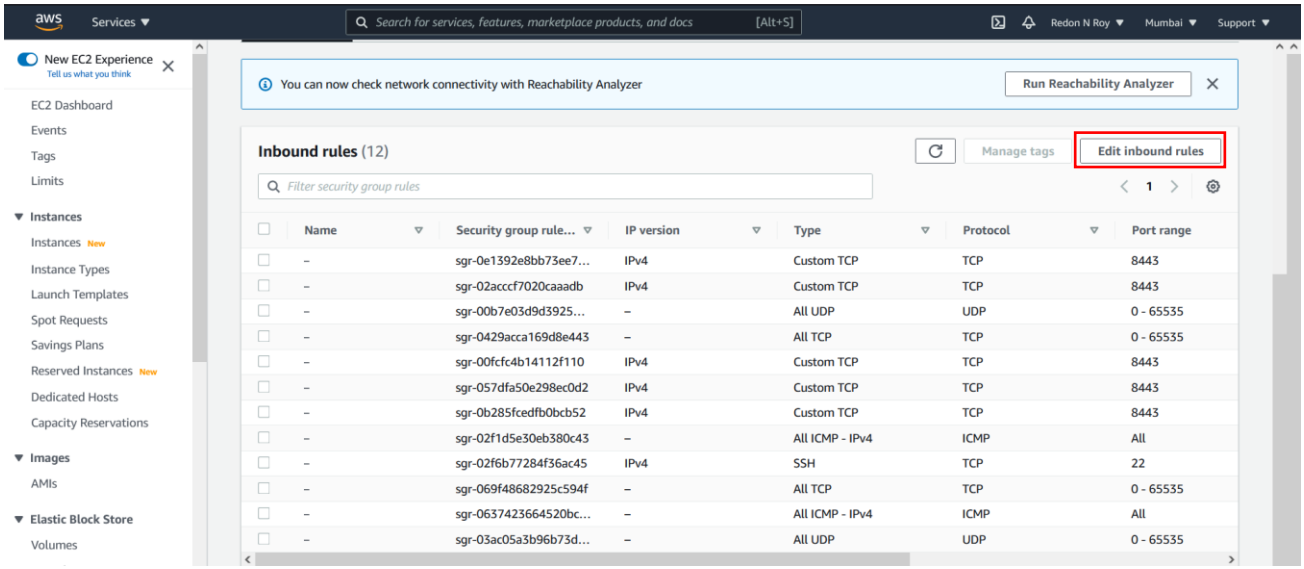
e. Now we need to configure the Security groups to allow connection over SSH. For that go to the bottom on the cluster dashboard and choose the **Security group for Master**.



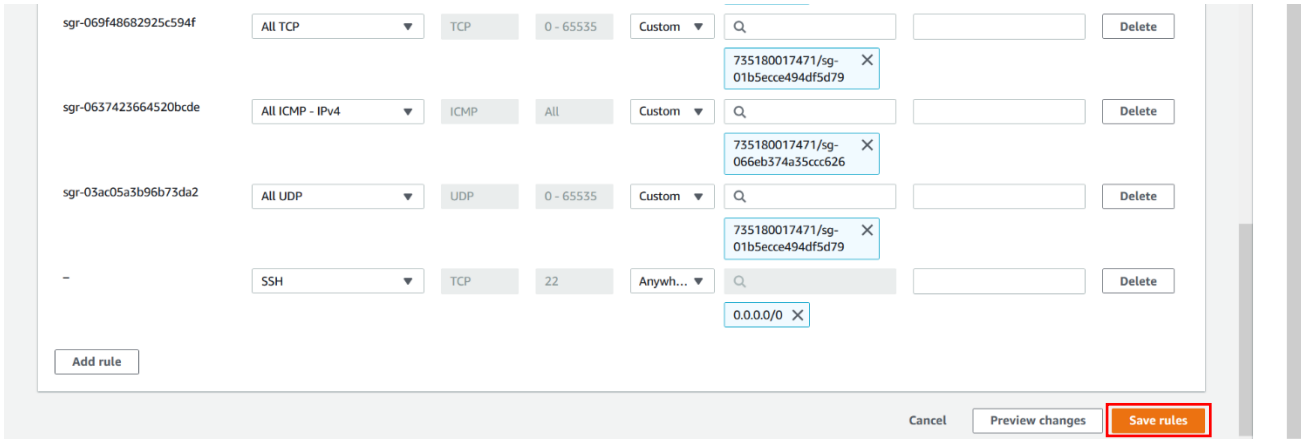
f. In there, choose the security group for the **master node**.



g. In the Inbound rules section, choose **Edit inbound rules** option on the right corner.



h. Go to the bottom of the Inbound rules and click on **Add rules**. Now select the **Type** as **SSH** and the **connection** to be **IPv4 – Anywhere**. Finally click on **Save rules**.



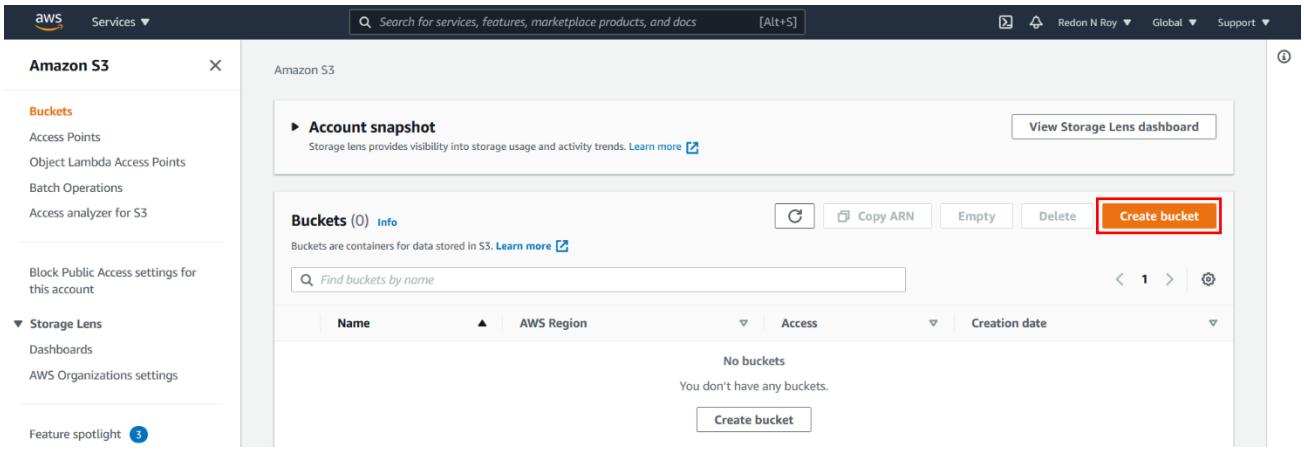
i. That’s it. The EMR cluster is fully configured.

3. Uploading Dataset and Program to S3

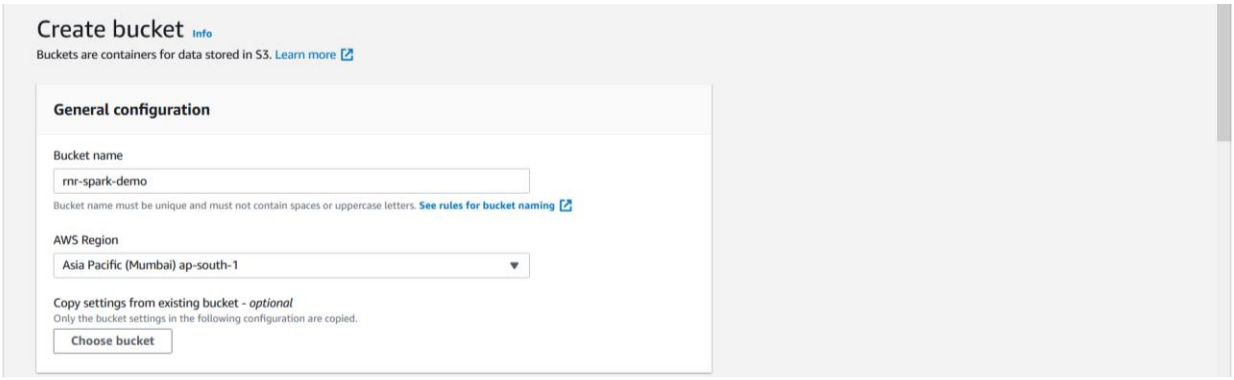
Before we can start processing the data on the EMR cluster, we need to put the dataset and the pyspark program into the S3 bucket so that it can be accessed easily by the EMR cluster. Since we need the file path to read the CSV file into the dataframe in pyspark program, we'll be uploading the dataset followed by the pyspark program. The steps include: -

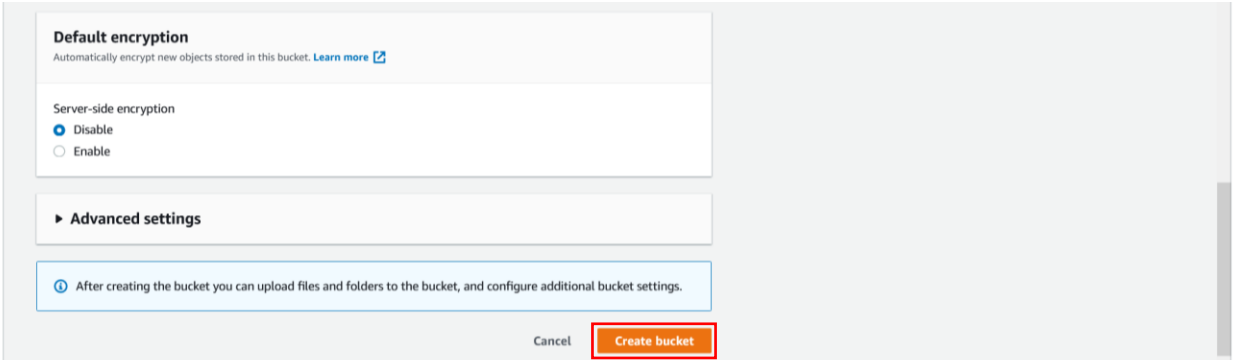
Note: - The dataset and Pyspark programs can be found here https://tinyurl.com/aws-emr-demo

a. We need to go to the S3 dashboard and setup a bucket there to hold all our data. Click on Create bucket option to create a new S3 bucket.

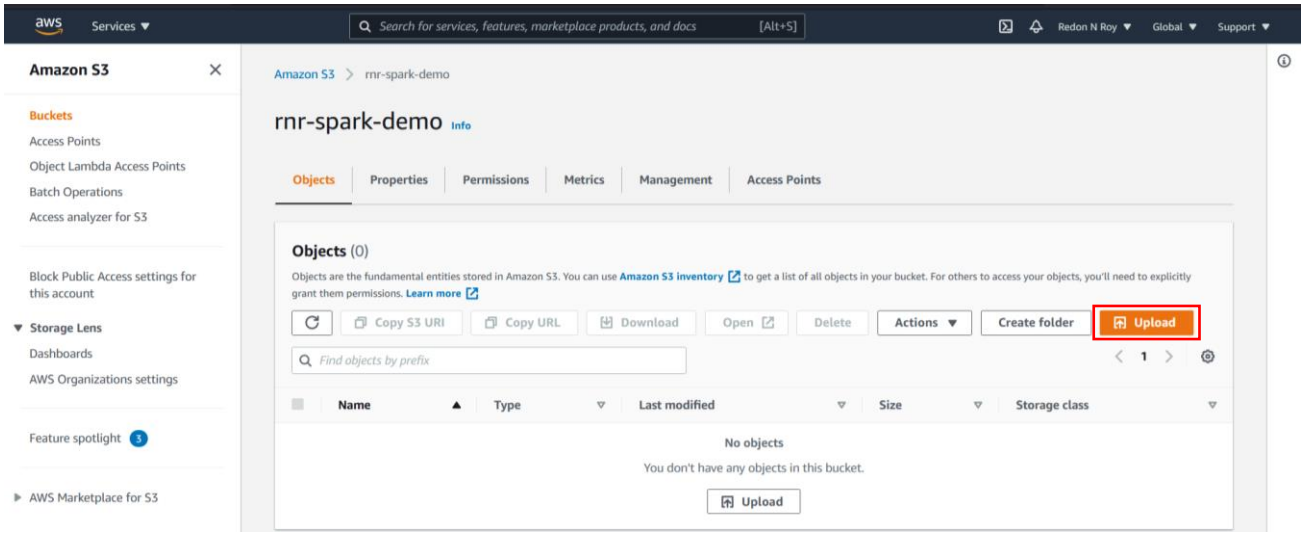


b. In the create bucket option, we need to give a bucket name that is globally unique. Leave all the other options as default and press the Create bucket option at the bottom of the screen.

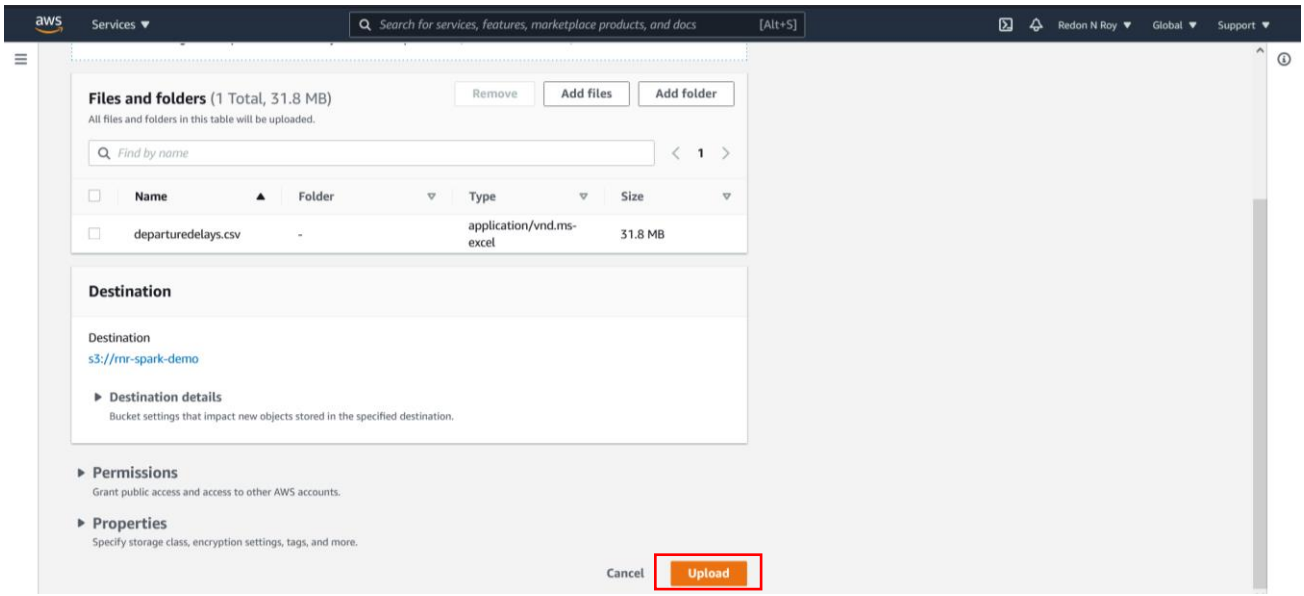




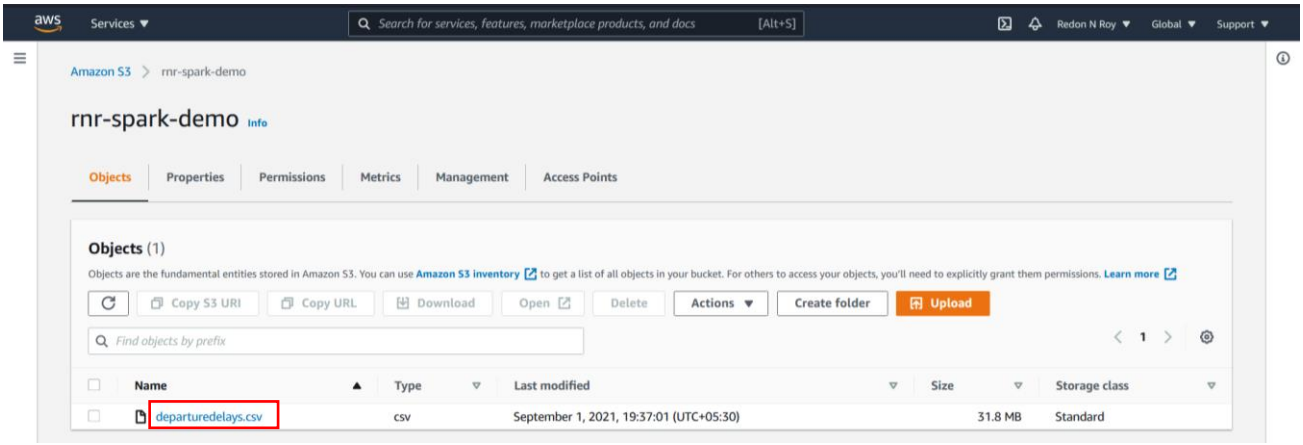
c. Now that we have created a bucket, we can upload the dataset into the bucket. For that, we can click on the **Upload** button.



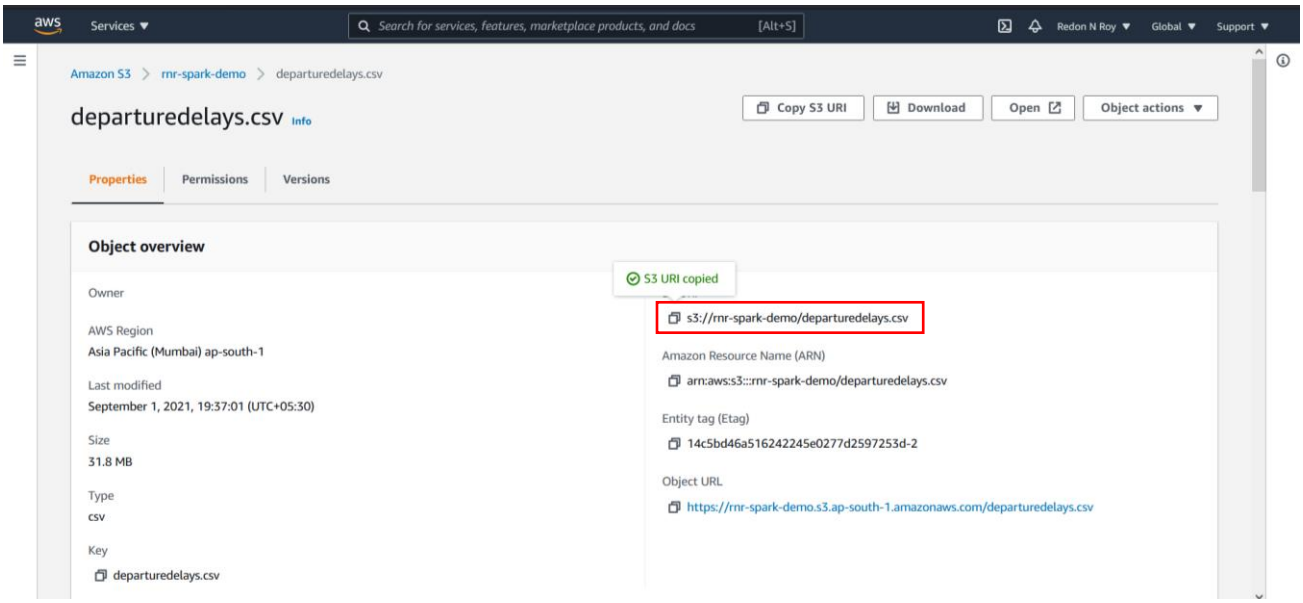
d. Now choose the CSV file that you want to upload. Keep all other option as default. Finally click **Upload**.



e. You would be able to see the file in the S3 bucket that you have created. From here go into the dataset to get the S3 URI.



f. Now copy the S3 URI on the right side so that the path to the dataset can be added to the pyspark program.



g. Now let's add this URI to both of the pyspark program. The path should be added to the variable named **"csv_file"** in both the programs.

```
from pyspark.sql import SparkSession
spark = (SparkSession
  .builder
  .appName("SparkExample")
  .getOrCreate())

spark.sparkContext.setLogLevel('ERROR')

csv_file = "s3://rnr-spark-demo/departuredelays.csv"

df = (spark.read.format("csv")
  .option("inferSchema", "true")
  .option("header", "true")
  .load(csv_file))
```

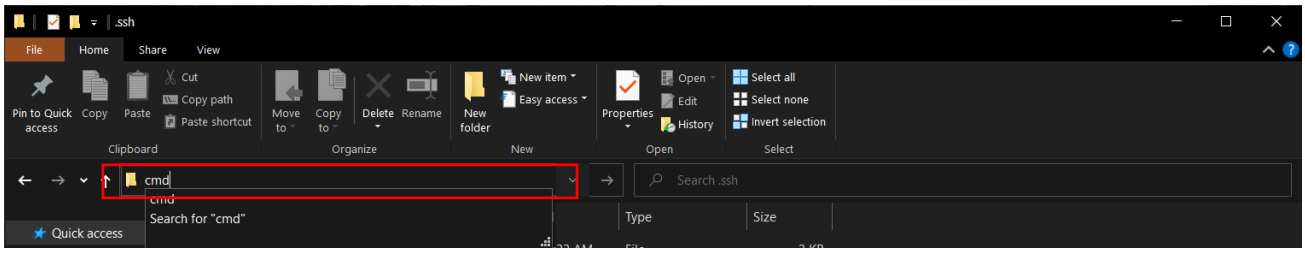
h. Now we can upload both pyspark files to the S3 bucket in the same way we uploaded the dataset.

4. Running spark-submit on EMR cluster

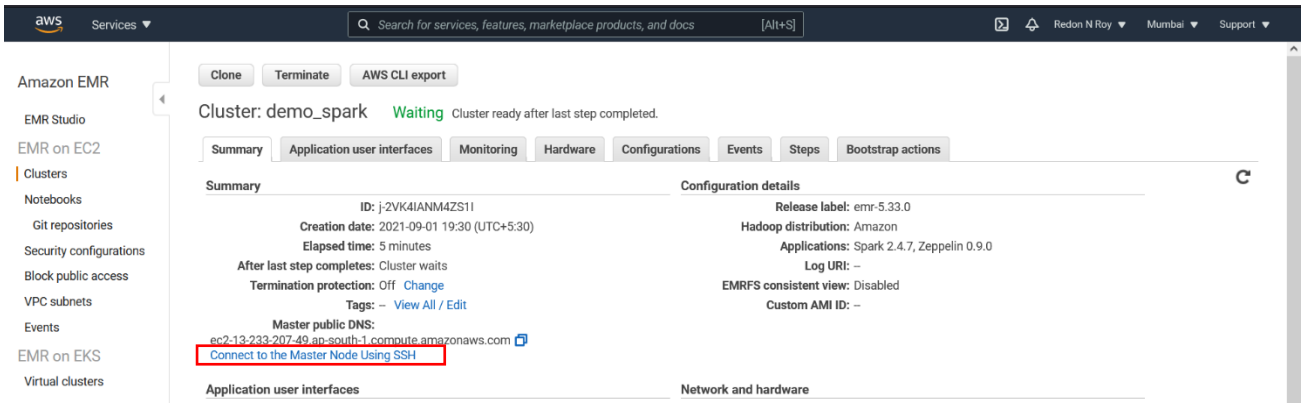
Now we are all set to run the pyspark program on the EMR cluster. For that we need to connect to the master node of the cluster via SSH. As discussed before, we can do it either through OpenSSH or with PuTTY. To connect to the cluster, follow the steps according to your preference of SSH connection.

Using OpenSSH

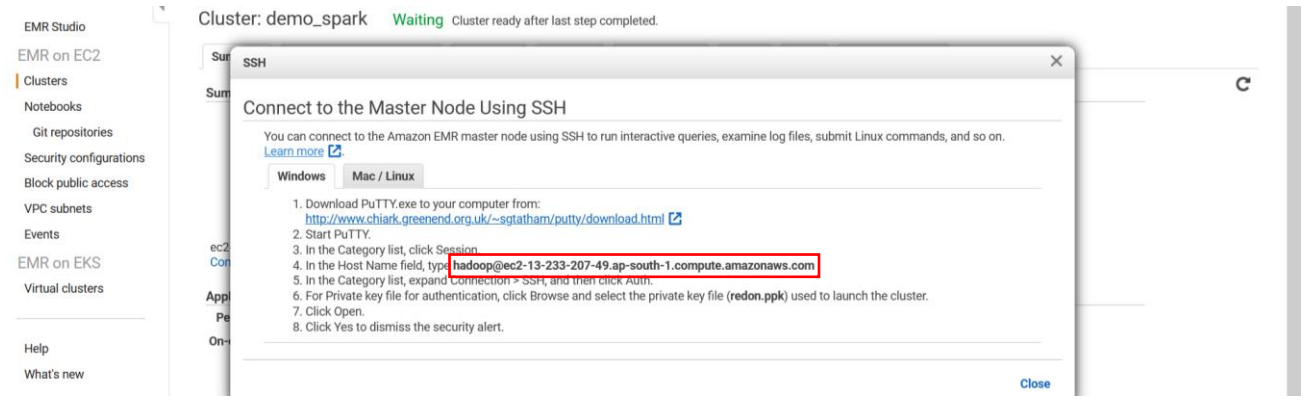
1. Open the folder where you have downloaded the key for the cluster (usually the downloads folder). Go to the address bar and type "**cmd**" and press enter. This will open a Command Prompt in that location. For me I had the key in a different folder.



2. Once you have the CMD open, go to the EMR cluster dashboard and click on the option "**Connect to the Master Node Using SSH**".



3. In there, find and copy the Host name as we need it to connect to the cluster.

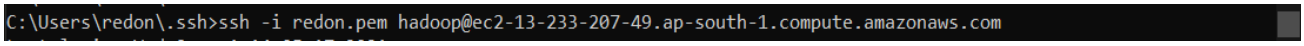


4. Now back in the CMD type the command in the following format and press enter.

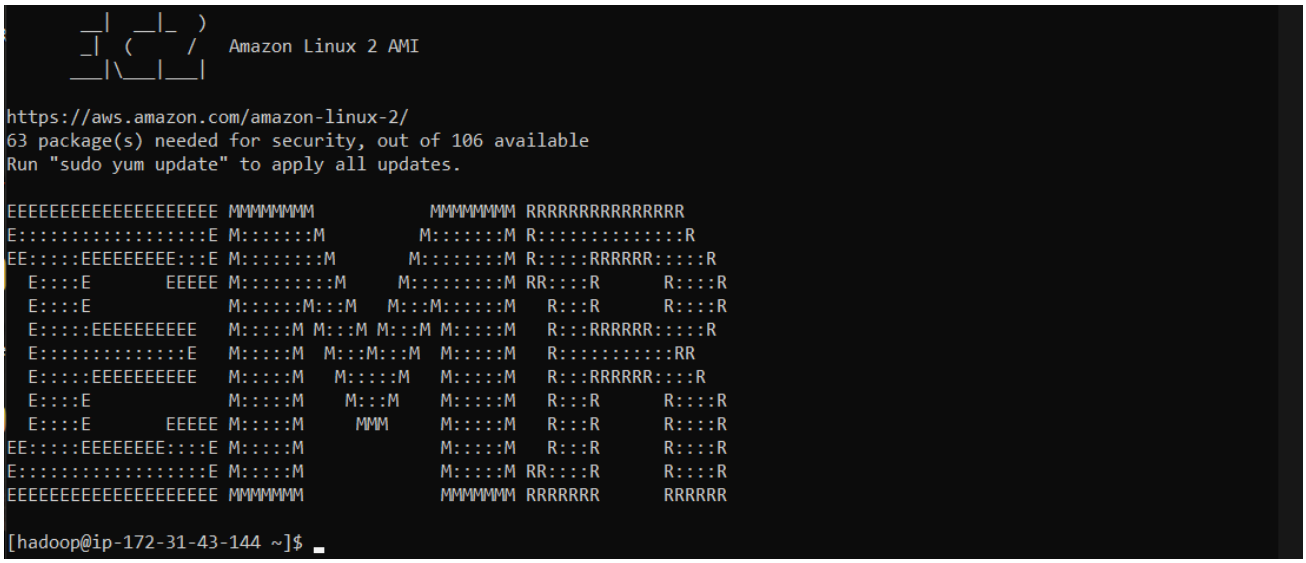
ssh -i key_file_name.pem host_name

E.g.: ssh -i redon.pem hadoop@ec2-13-233-207-49.ap-south-1.compute.amazonaws.com

When connecting for the first time, it will ask permission to add to known host. Type yes and press enter.



5. Now you'll be welcomed with a similar screen that confirms that you are connected to the EMR cluster.



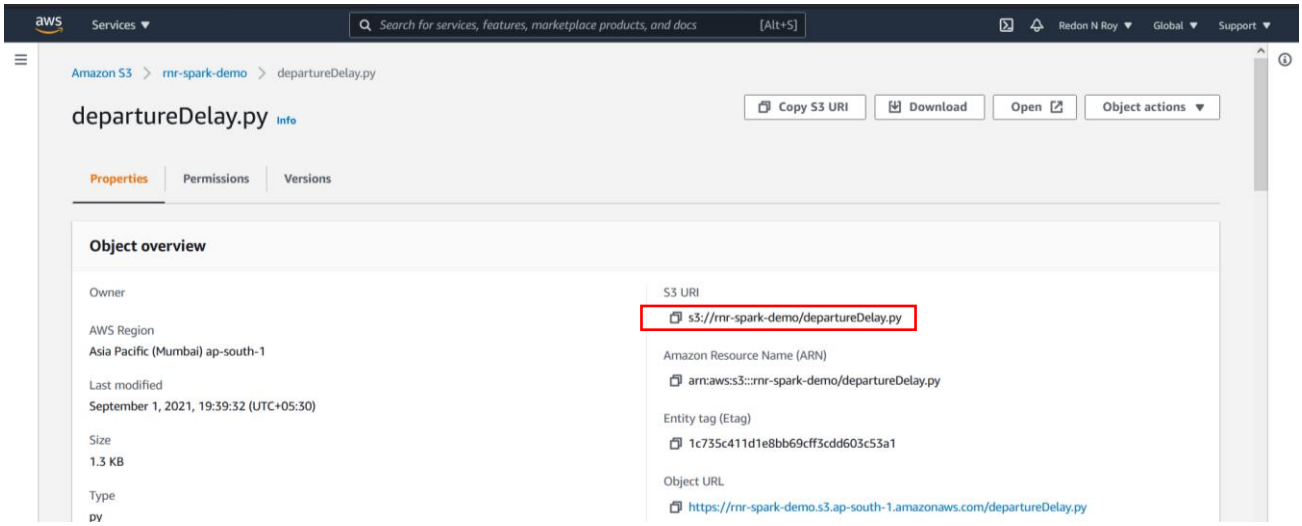
Using PuTTY

(I haven't tried this method personally; the instructions are being provided by AWS.)

1. Open PuTTY.
2. In the Category list, click Session.
3. In the Host Name field, type your host name, for example: ***hadoop@ec2-13-233-207-49.ap-south-1.compute.amazonaws.com***
4. In the Category list, expand Connection and choose SSH. Then click on Auth.
5. Now click on browse and select the .ppk file downloaded earlier. Then click open to add the key to PuTTY.
6. Finally click Yes to dismiss the security alert.
7. This should connect you to the cluster through PuTTY.

Once connected to the cluster by any one of the two methods, we can now run the spark-submit through the terminal. For that the steps are: -

a. First go to the S3 bucket and get the S3 URI of the pyspark program so that we can download the same into the cluster for running it.



b. Now go back to the terminal and use the following command to download the pyspark program into the cluster.

Aws s3 cp S3_URI .

E.g. : aws s3 cp s3://rnr-spark-demo/departureDelay.py .

(DON'T FORGET TO PUT A SPACE AND A DOT(.) AFTER THE S3 URI)

```
[hadoop@ip-172-31-43-144 ~]$ aws s3 cp s3://rnr-spark-demo/departureDelay.py .
download: s3://rnr-spark-demo/departureDelay.py to ./departureDelay.py
```

c. Now run spark-submit with the pyspark program and it will produce the output of the queries.

```
[hadoop@ip-172-31-43-144 ~]$
[hadoop@ip-172-31-43-144 ~]$ spark-submit departureDelay.py
```

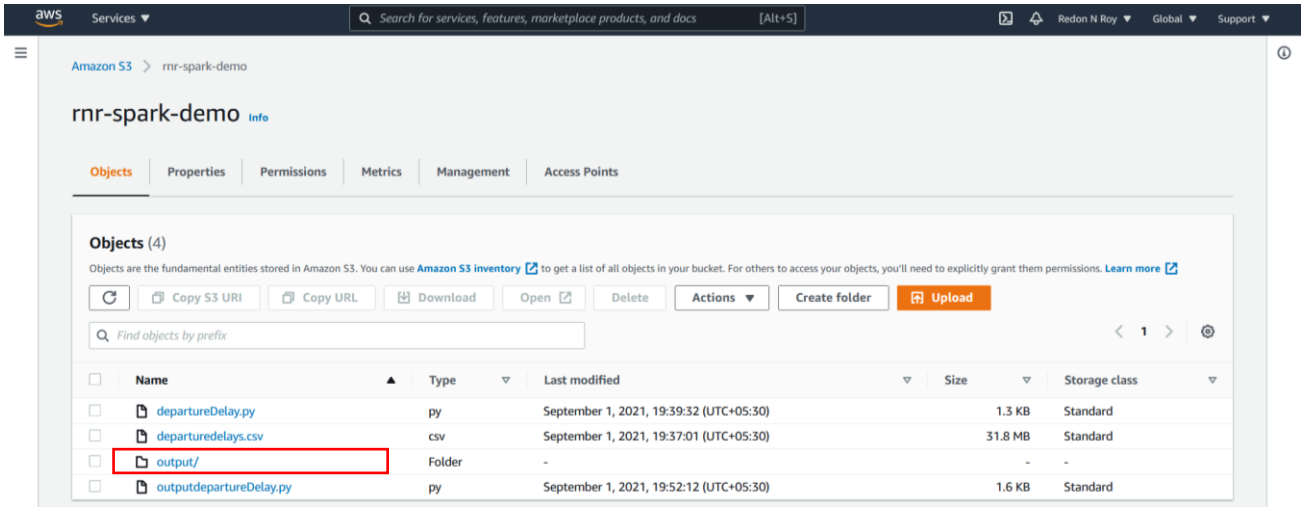
```
hadoop@ip-172-31-43-144:~

+--------+------+-----------+
|distance|origin|destination|
+--------+------+-----------+
|    4330|   HNL|        JFK|
|    4330|   HNL|        JFK|
|    4330|   HNL|        JFK|
|    4330|   HNL|        JFK|
|    4330|   HNL|        JFK|
|    4330|   HNL|        JFK|
|    4330|   HNL|        JFK|
|    4330|   HNL|        JFK|
|    4330|   HNL|        JFK|
|    4330|   HNL|        JFK|
+--------+------+-----------+
only showing top 10 rows

+-------+-----+------+-----------+
|   date|delay|origin|destination|
+-------+-----+------+-----------+
|2190925| 1638|   SFO|        ORD|
|1031755|  396|   SFO|        ORD|
|1022330|  326|   SFO|        ORD|
|1051205|  320|   SFO|        ORD|
|1190925|  297|   SFO|        ORD|
|2171115|  296|   SFO|        ORD|
|1071040|  279|   SFO|        ORD|
|1051550|  274|   SFO|        ORD|
|3120730|  266|   SFO|        ORD|
|1261104|  258|   SFO|        ORD|
+-------+-----+------+-----------+
only showing top 10 rows

+-----+------+-----------+-------------+
|delay|origin|destination|Flight_Delays|
+-----+------+-----------+-------------+
|  333|   ABE|        ATL|  Long Delays|
|  305|   ABE|        ATL|  Long Delays|
|  275|   ABE|        ATL|  Long Delays|
|  257|   ABE|        ATL|  Long Delays|
|  247|   ABE|        DTW|  Long Delays|
|  247|   ABE|        ATL|  Long Delays|
|  219|   ABE|        ORD|  Long Delays|
|  211|   ABE|        ATL|  Long Delays|
|  197|   ABE|        DTW|  Long Delays|
|  192|   ABE|        ORD|  Long Delays|
+-----+------+-----------+-------------+
only showing top 10 rows

[hadoop@ip-172-31-43-144 ~]$
```

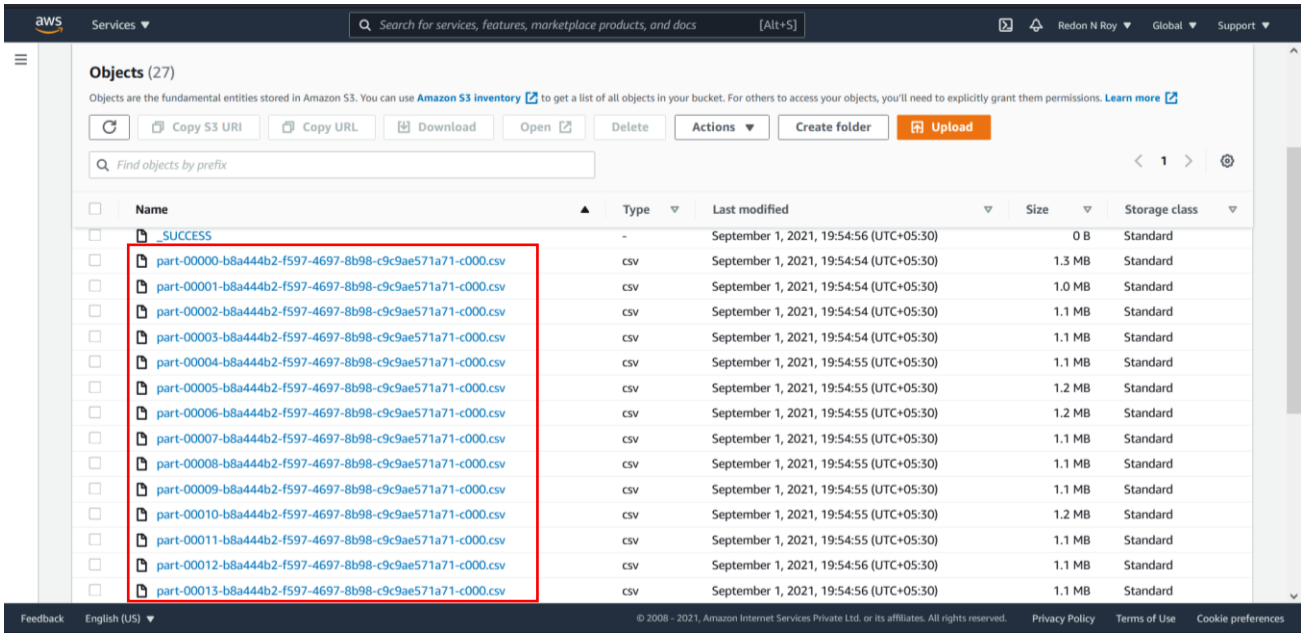
Getting output to S3

d. Now we can run the pyspark program to get output to the S3 bucket. For that let's get the second pyspark program in the same way as the first program using S3 URI.

```
hadoop@ip-172-31-43-144:~
[hadoop@ip-172-31-43-144 ~]$ aws s3 cp s3://rnr-spark-demo/outputdepartureDelay.py .
download: s3://rnr-spark-demo/outputdepartureDelay.py to ./outputdepartureDelay.py
[hadoop@ip-172-31-43-144 ~]$
[hadoop@ip-172-31-43-144 ~]$
[hadoop@ip-172-31-43-144 ~]$
[hadoop@ip-172-31-43-144 ~]$
[hadoop@ip-172-31-43-144 ~]$ spark-submit outputdepartureDelay.py
```

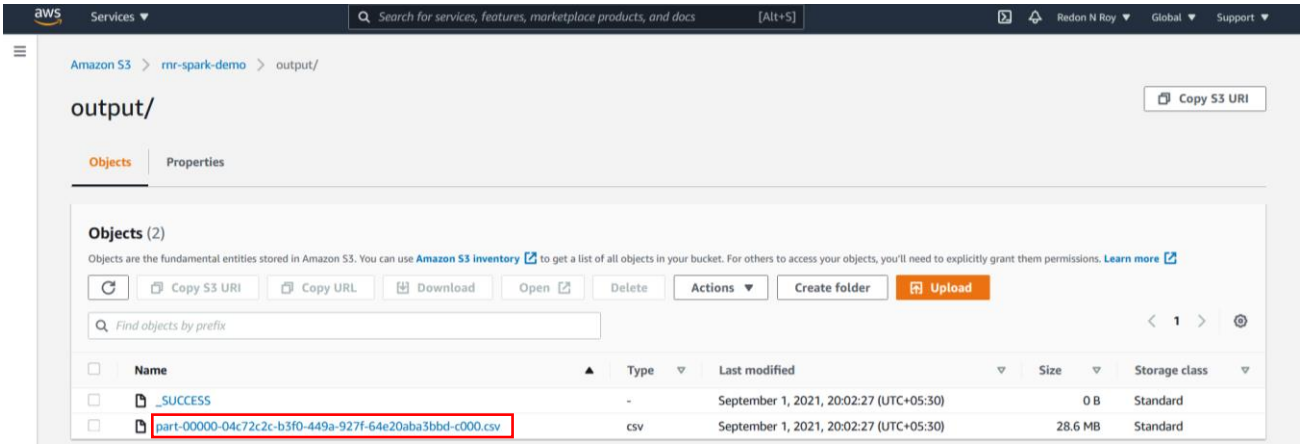
e. It will store the output into the S3 bucket in the folder we have mentioned. The file will be stored in parts if the coalesce(1) is not given along with the write command.



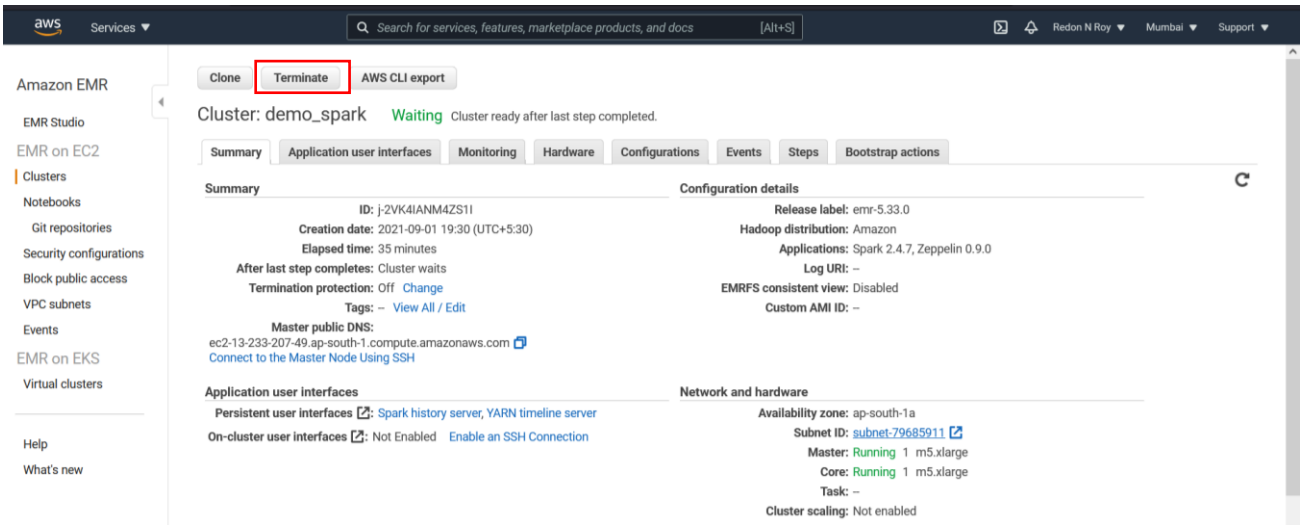


f. With the addition of coalesce(1), the entire output will be stored in a single file.

```
45
46  df2.coalesce(1).write.mode("overwrite").csv("s3://rnr-spark-demo/output")
```

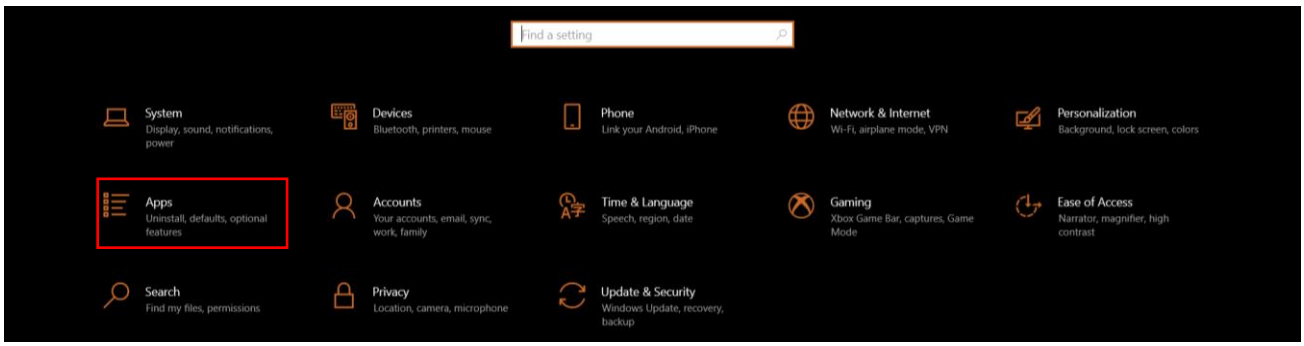


g. Once you are done with working on the EMR cluster, you can got to the EMR cluster management dashboard and terminate the session.

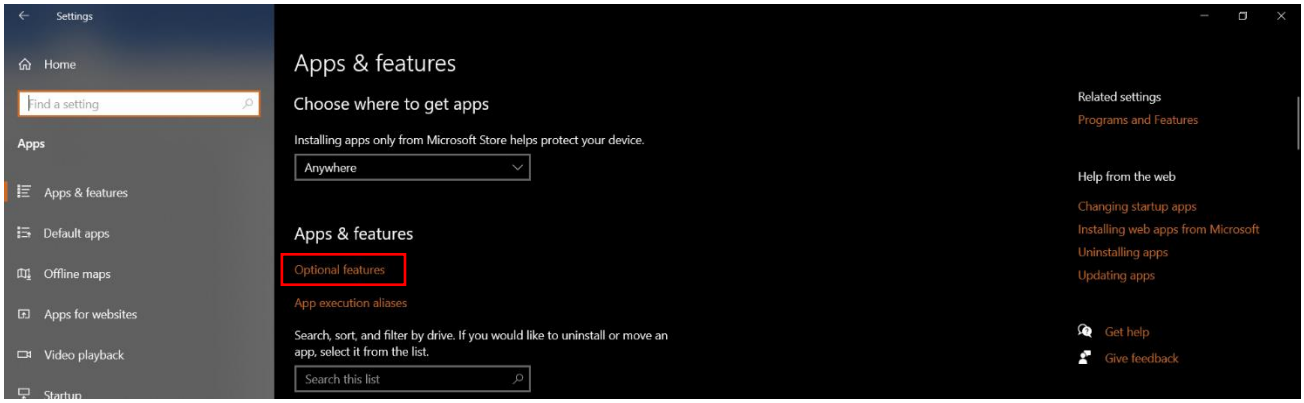


How to install OpenSSH in Windows 10

1. Go to setting in Windows 10 and select the Apps option.

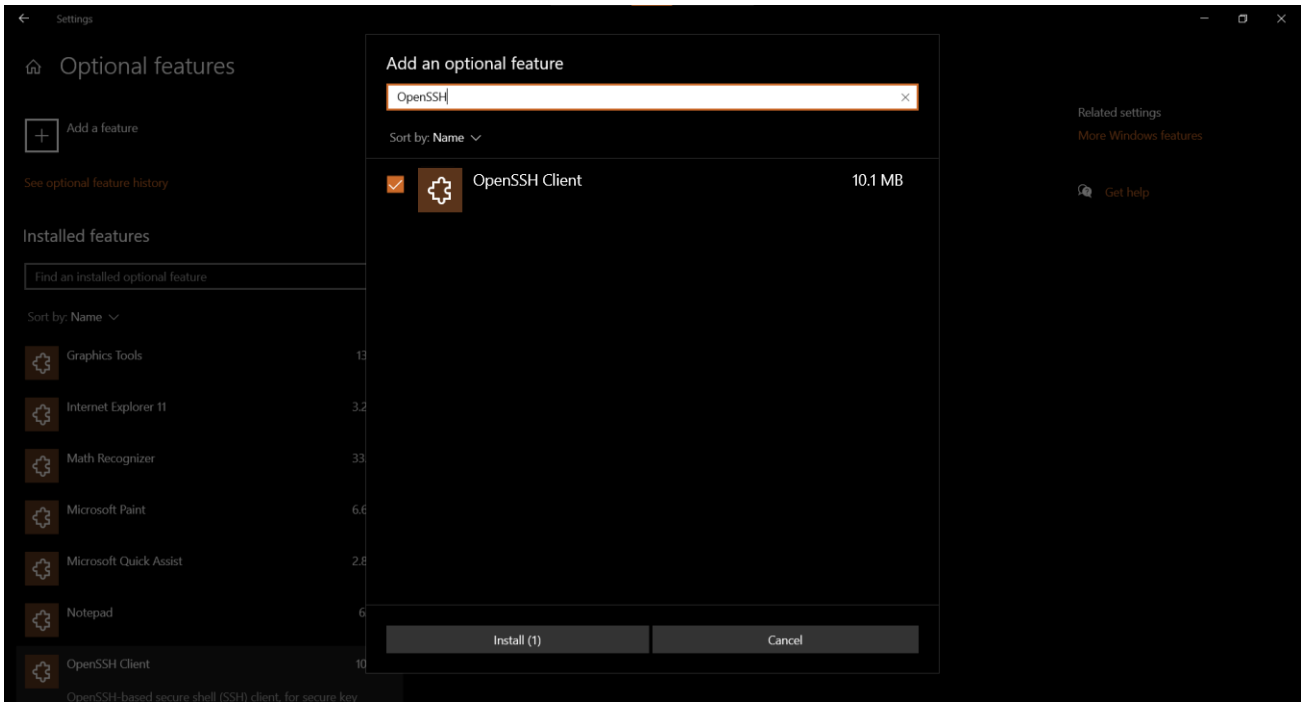


2. In Apps option, go to the optional features.



3. Click on Add a feature

4. Now search and install **<u>OpenSSH client.</u>**



5. Now OpenSSH is installed on your system successfully. You can use it directly from the CMD.

   In case if it is not working, just restart your system and it will be available.