DTSC 701: INTRODUCTION TO BIG DATA

PROJECT ASSIGNMENT

Names:

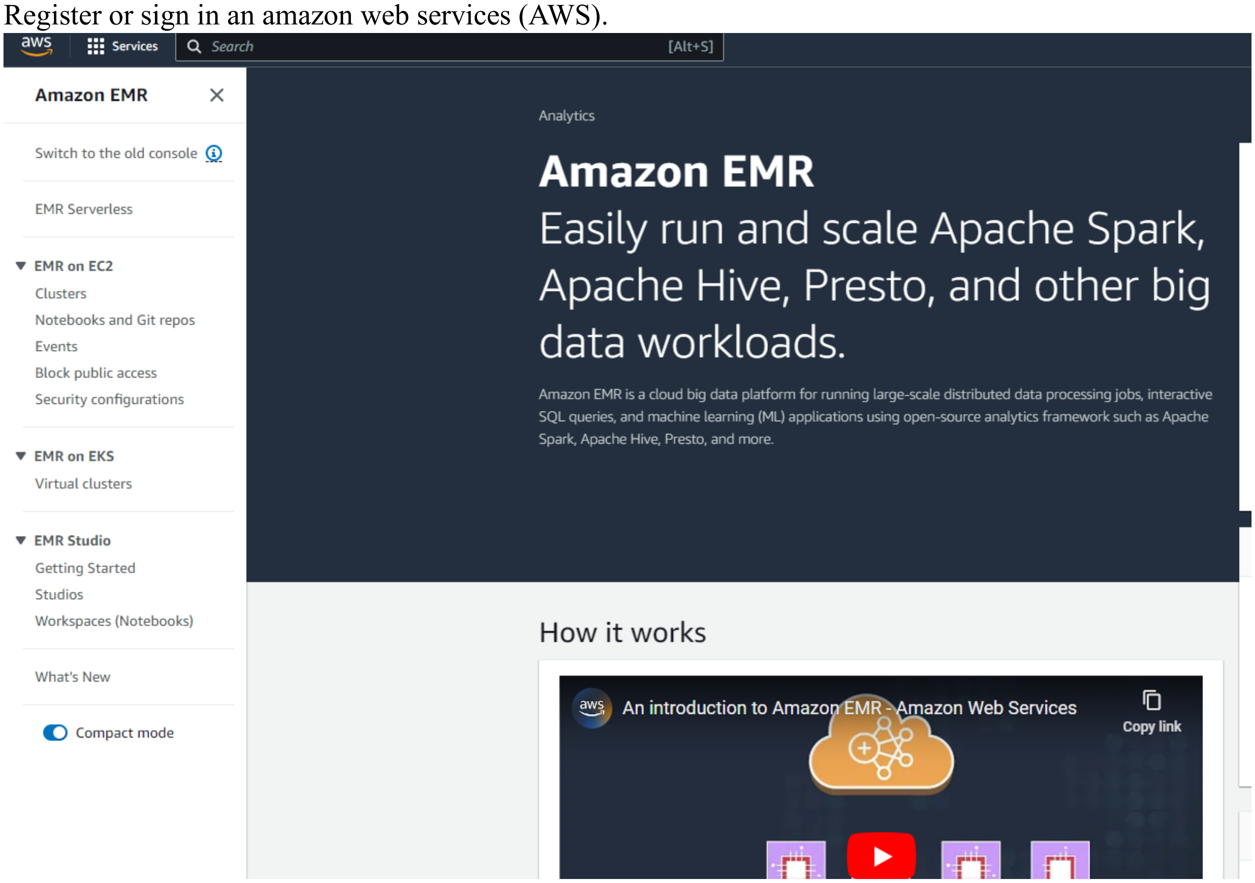
AISHWARYA KANAKAMEDALA (ID: 1322532)

TOYAZ VAMSI INAGANTI (ID: 1316806)

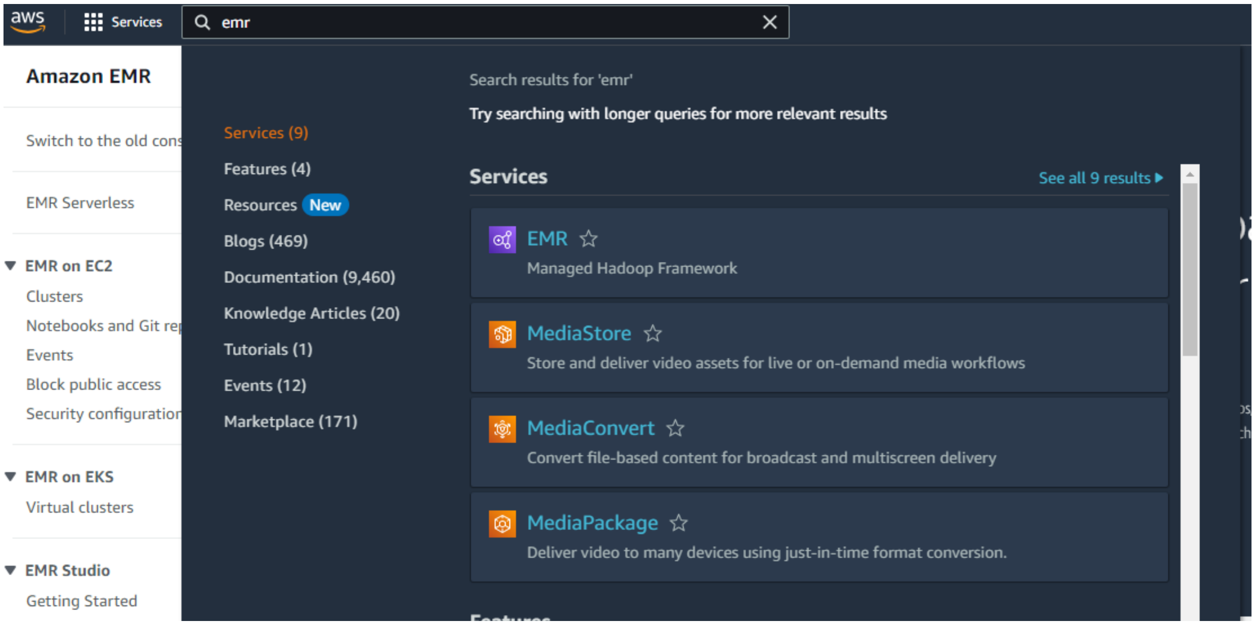
1. A data lake can include structured data from relational databases (rows and columns), semi-structured data (CSV, logs, XML, JSON), unstructured data (emails, documents, PDFs) and binary data (images, audio, video). Therefore a CSV as data lake downloaded from Kaggle, which has fuel consumption from year 2000 to 2022, with more than 27000 rows.

1	ticker	commodity	date	open	high	low	close	volume
2	CL=F	Crude Oil	8/23/2000	31.95000076	32.79999924	31.95000076	32.04999924	79385
3	CL=F	Crude Oil	8/24/2000	31.89999962	32.24000168	31.39999962	31.62999916	72978
4	CL=F	Crude Oil	8/25/2000	31.70000076	32.09999847	31.31999969	32.04999924	44601
5	CL=F	Crude Oil	8/28/2000	32.04000092	32.91999817	31.86000061	32.86999893	46770
6	CL=F	Crude Oil	8/29/2000	32.81999969	33.02999878	32.56000137	32.72000122	49131
7	CL=F	Crude Oil	8/30/2000	32.75	33.40000153	32.09999847	33.40000153	79214
8	CL=F	Crude Oil	8/31/2000	33.25	33.70000076	32.97000122	33.09999847	56895
9	CL=F	Crude Oil	9/1/2000	33.04999924	33.45000076	32.75	33.38000107	45869
10	CL=F	Crude Oil	9/5/2000	33.95000076	33.99000168	33.41999817	33.79999924	55722
11	CL=F	Crude Oil	9/6/2000	33.99000168	34.95000076	33.83000183	34.95000076	74692
12	CL=F	Crude Oil	9/7/2000	34.5	35.5	34.45000076	35.33000183	74105
13	CL=F	Crude Oil	9/8/2000	34.54999924	34.77999878	33.40000153	33.70000076	88415
14	CL=F	Crude Oil	9/11/2000	33.79999924	35.84999847	33.75	35.09999847	101518
15	CL=F	Crude Oil	9/12/2000	35.45000076	35.5	34.09999847	34.20000076	91911
16	CL=F	Crude Oil	9/13/2000	34	34.74000168	33.5	33.79999924	94630
17	CL=F	Crude Oil	9/14/2000	33.77999878	34.5	33.11999893	34.09999847	98068
18	CL=F	Crude Oil	9/15/2000	34.5	36.09999847	34.45000076	35.84999847	85839
19	CL=F	Crude Oil	9/18/2000	36.20000076	37.15000153	36.15000153	36.88000107	59663
20	CL=F	Crude Oil	9/19/2000	36.54999924	37	36.15000153	36.5	62731
21	CL=F	Crude Oil	9/20/2000	37.5	37.79999924	36.5	37.5	119080
22	CL=F	Crude Oil	9/21/2000	34.65000153	35.5	33.34999847	33.95000076	110851
23	CL=F	Crude Oil	9/22/2000	34	34.40000153	32.5	32.65000153	85083

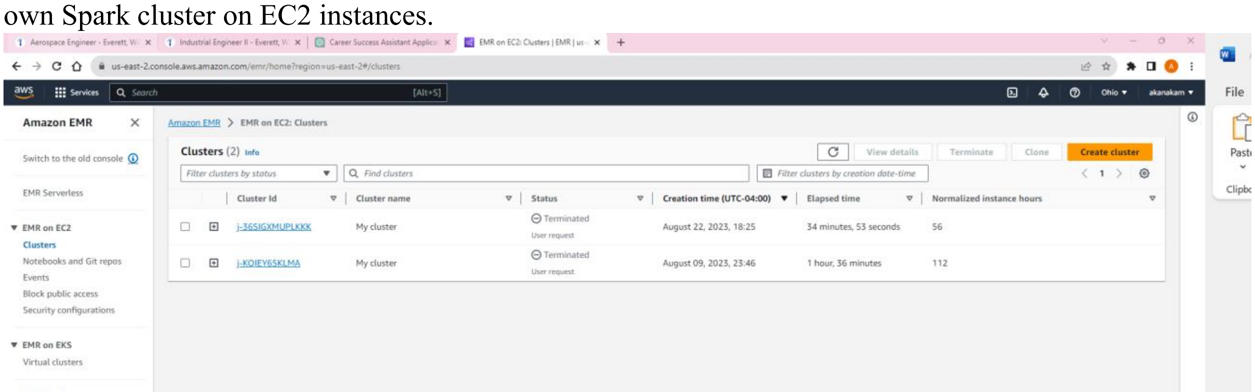
2. Register or sign in an amazon web services (AWS).



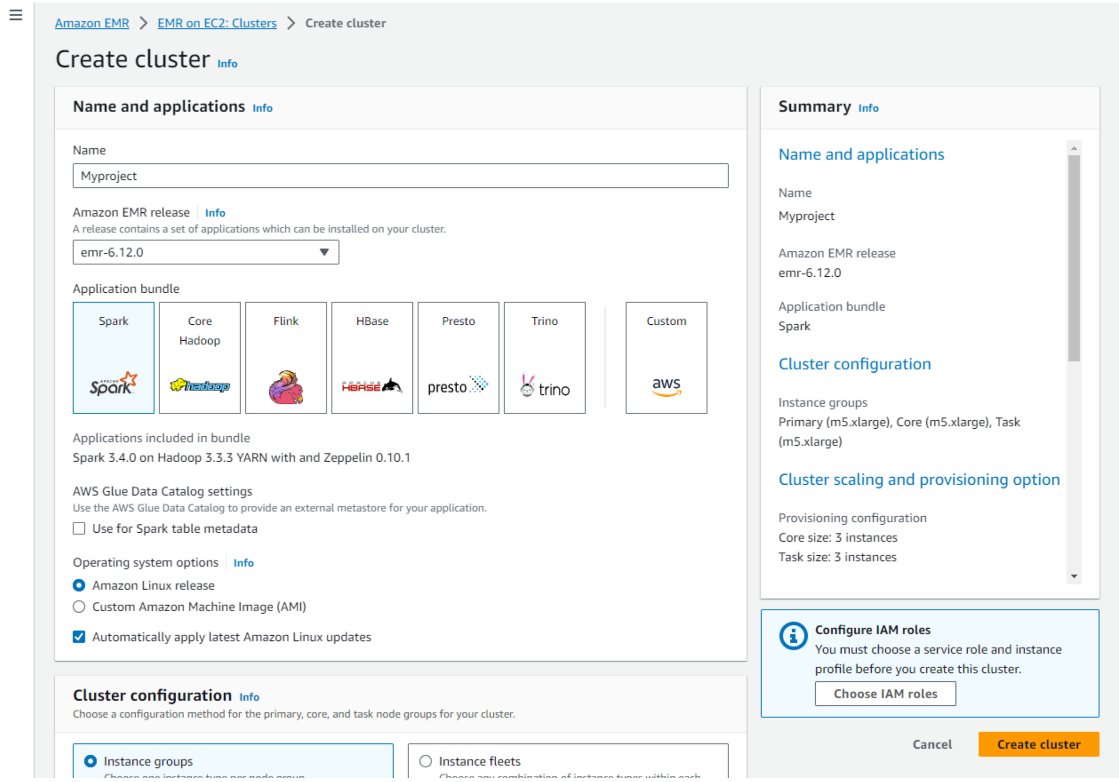
3. Navigate to EMR in the search bar to create a new cluster



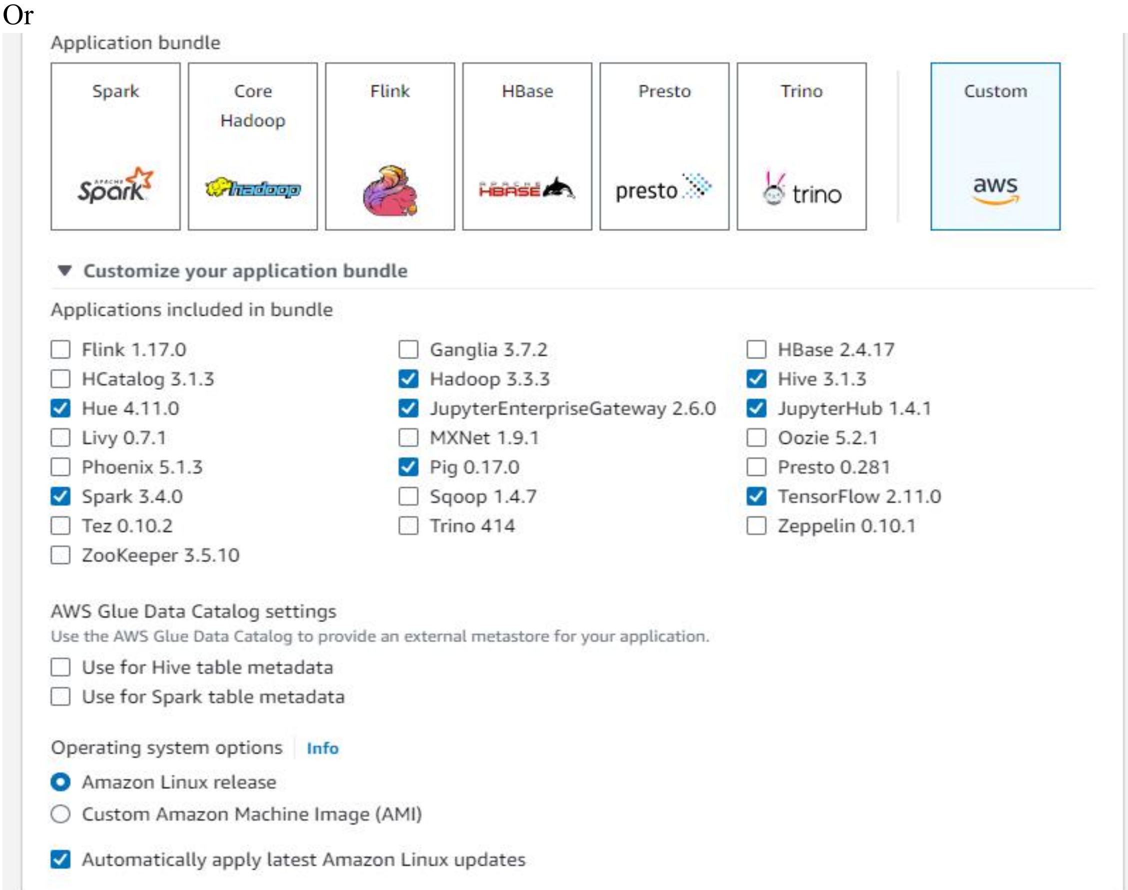
4. set up a Spark cluster on AWS using Amazon EMR (Elastic MapReduce) or by deploying your own Spark cluster on EC2 instances.



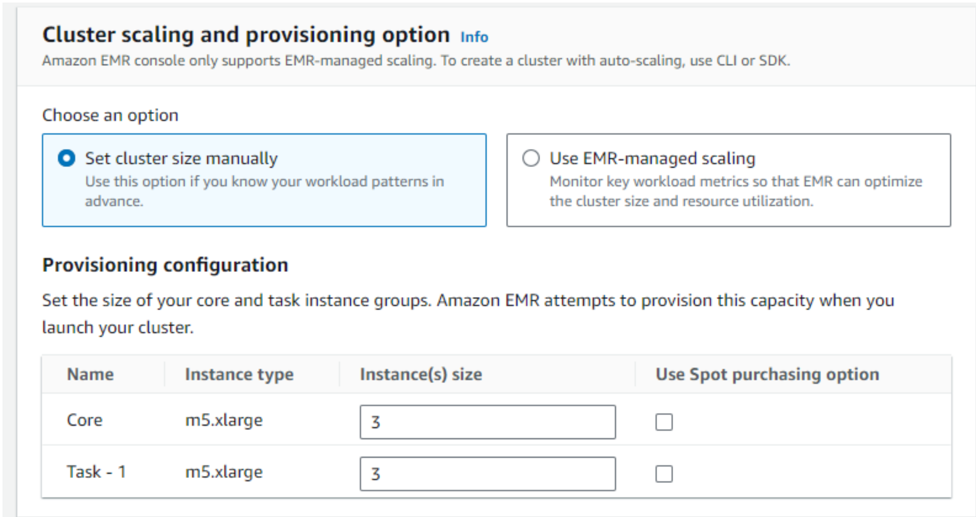
5. Name a new cluster, where we choose Spark or customize depending on the requirement as Application bundle, and cluster configuration as Primary(m5.xlarge), Core (m5.xlarge), Task (m5.xlarge) and increasing Cluster scaling and provisioning option as Core size as 3 instances and Task size as 3 instances. For creating a cluster for machine learning purpose, we select additional resources in the customized bundle as shown in fig below


Amazon EMR > EMR on EC2: Clusters > Create cluster
Create cluster Info
Name and applications Info
Name
Myproject
Amazon EMR release Info
A release contains a set of applications which can be installed on your cluster.
emr-6.12.0
Application bundle
Spark
Core Hadoop
Flink
HBase
Presto
Trino
Custom
Applications included in bundle
Spark 3.4.0 on Hadoop 3.3.3 YARN with and Zeppelin 0.10.1
AWS Glue Data Catalog settings
Use the AWS Glue Data Catalog to provide an external metastore for your application.
Use for Spark table metadata
Operating system options Info
Amazon Linux release
Custom Amazon Machine Image (AMI)
Automatically apply latest Amazon Linux updates
Cluster configuration Info
Choose a configuration method for the primary, core, and task node groups for your cluster.
Instance groups
Instance fleets
Summary Info
Name and applications
Name
Myproject
Amazon EMR release
emr-6.12.0
Application bundle
Spark
Cluster configuration
Instance groups
Primary (m5.xlarge), Core (m5.xlarge), Task (m5.xlarge)
Cluster scaling and provisioning option
Provisioning configuration
Core size: 3 instances
Task size: 3 instances
Configure IAM roles
You must choose a service role and instance profile before you create this cluster.
Choose IAM roles
Cancel
Create cluster

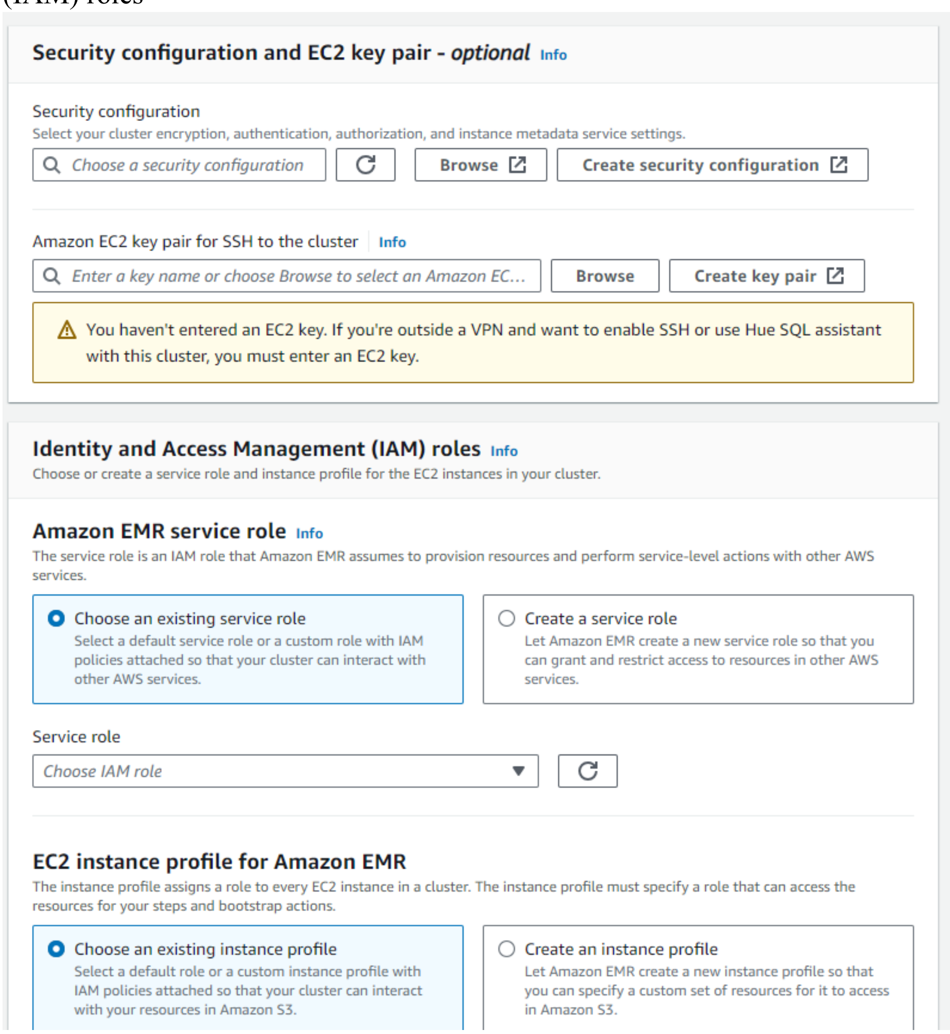
Or


Application bundle
Spark
Core Hadoop
Flink
HBase
Presto
Trino
Custom
Customize your application bundle
Applications included in bundle
Flink 1.17.0
Ganglia 3.7.2
HBase 2.4.17
HCatalog 3.1.3
Hadoop 3.3.3
Hive 3.1.3
Hue 4.11.0
JupyterEnterpriseGateway 2.6.0
JupyterHub 1.4.1
Livy 0.7.1
MXNet 1.9.1
Oozie 5.2.1
Phoenix 5.1.3
Pig 0.17.0
Presto 0.281
Spark 3.4.0
Sqoop 1.4.7
TensorFlow 2.11.0
Tez 0.10.2
Trino 414
Zeppelin 0.10.1
ZooKeeper 3.5.10
AWS Glue Data Catalog settings
Use the AWS Glue Data Catalog to provide an external metastore for your application.
Use for Hive table metadata
Use for Spark table metadata
Operating system options Info
Amazon Linux release
Custom Amazon Machine Image (AMI)
Automatically apply latest Amazon Linux updates

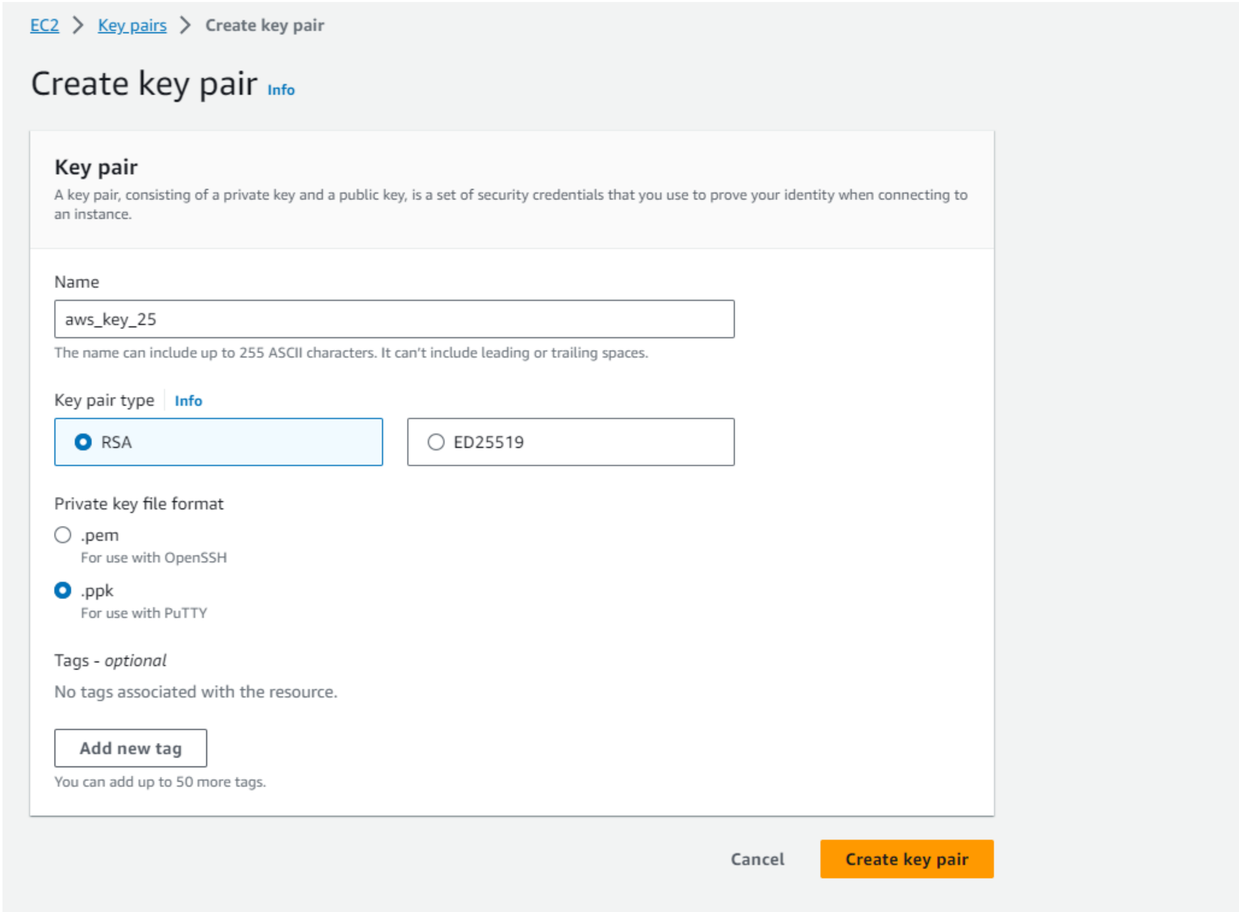
6. Scale the cluster to the size of 3 instances for core and task nodes.



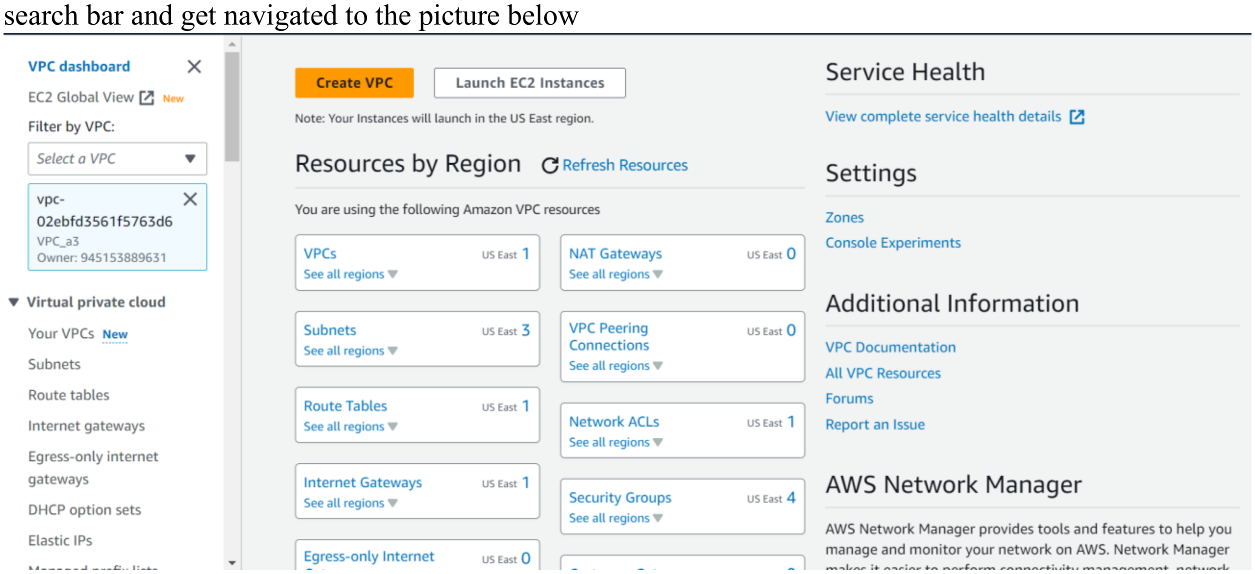
7. Create or security configuration and EC2 key pair as well as Identity and Access Management (IAM) roles



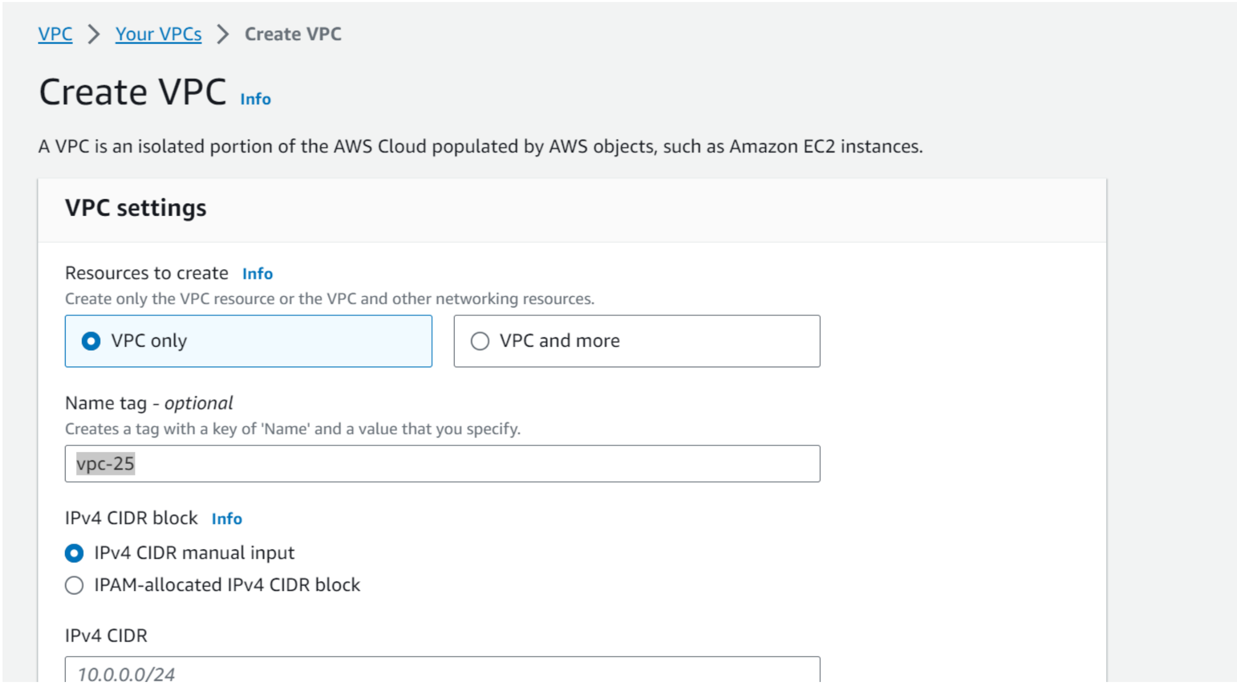
8. Create a key pair by clicking on **create key pair** as shown in the picture below and then browse it and select it for the EC2 pair. This step is completely option

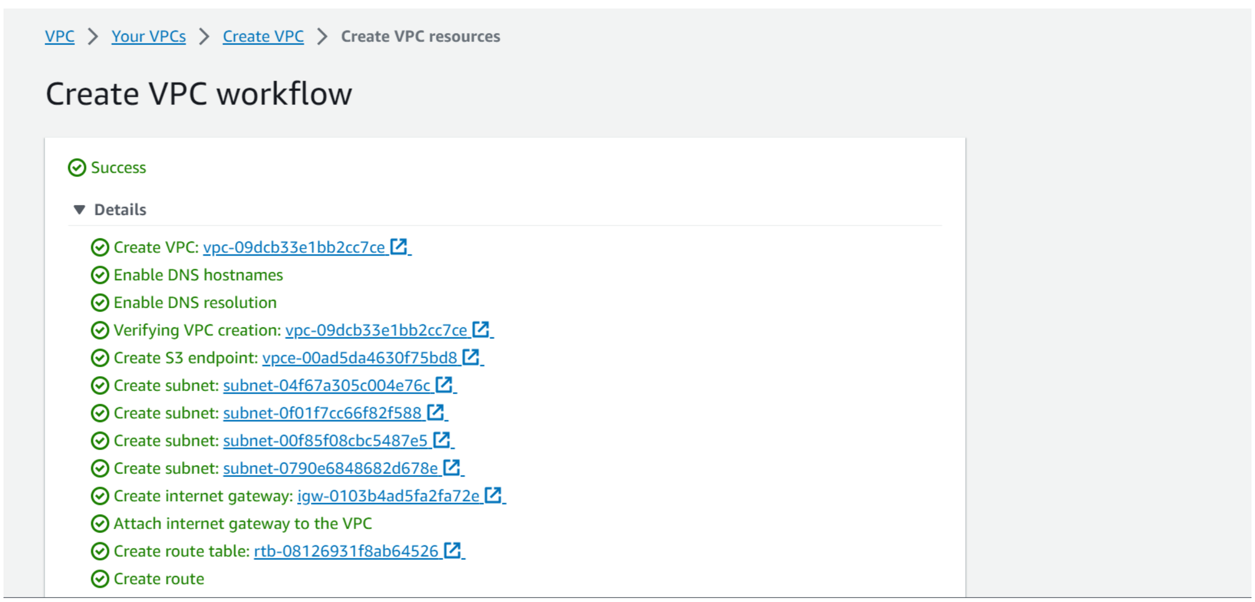


9. Next step is to Create a service role or use an existing service role. Below are the pictures showing how to create a VPC and subnet and security group for the first time.
By creating new **Virtual Private Cloud (VPC) and subnet and Security group.** Search VPC on search bar and get navigated to the picture below

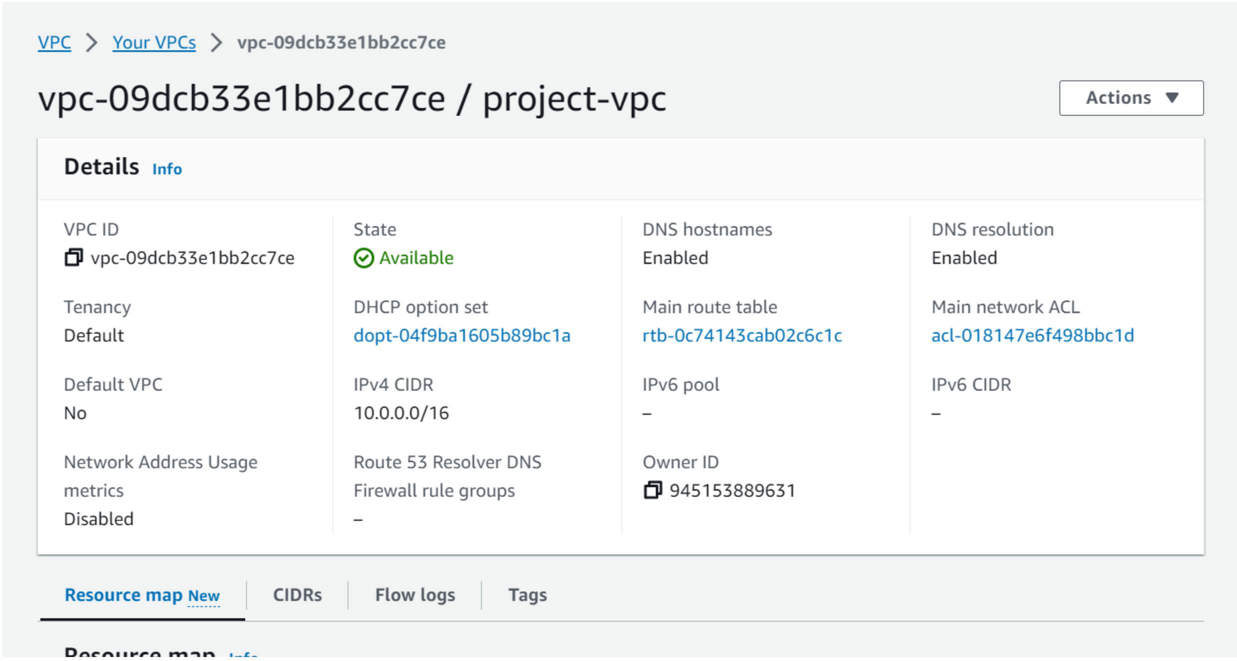


Choose the option of VPC and more, and name the VPC. The option of VPC and more will create the subnet and security group key also as shown in the below pictures.

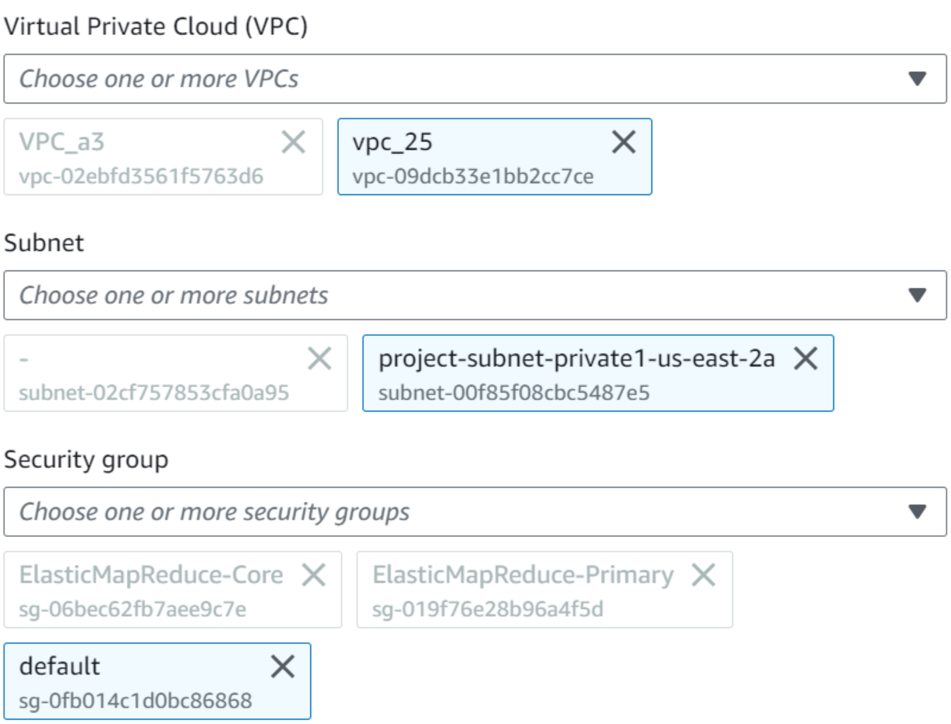




The image below shows the VPC details that is created and can be used in the cluster

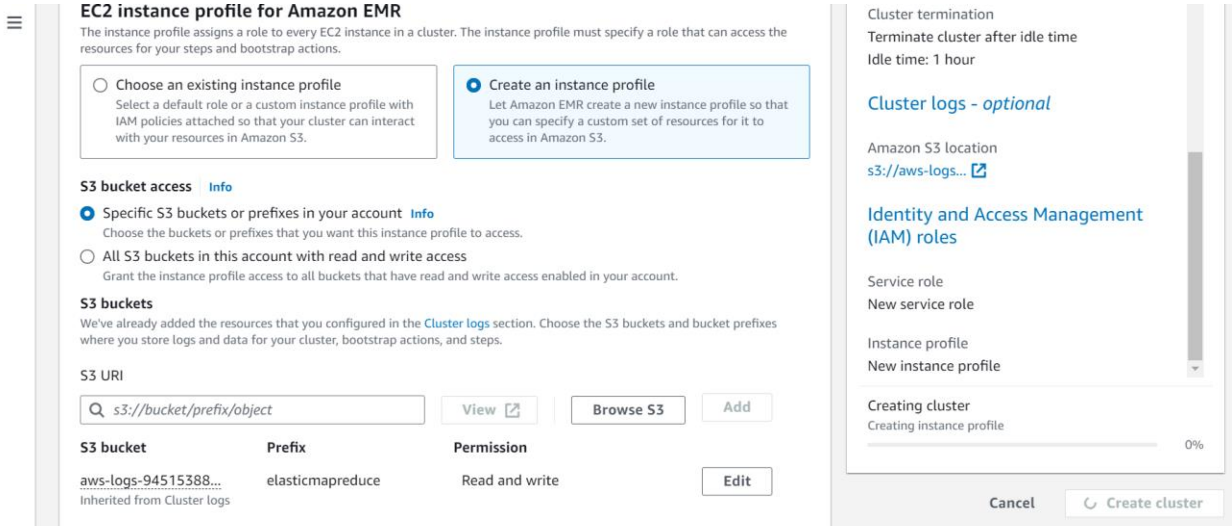


10. After selecting create a service role, one has to select the VPC, Subnet and security group options by clicking on them and selecting from the dropdown menu. The image shows the selected ones highlighted in blue.

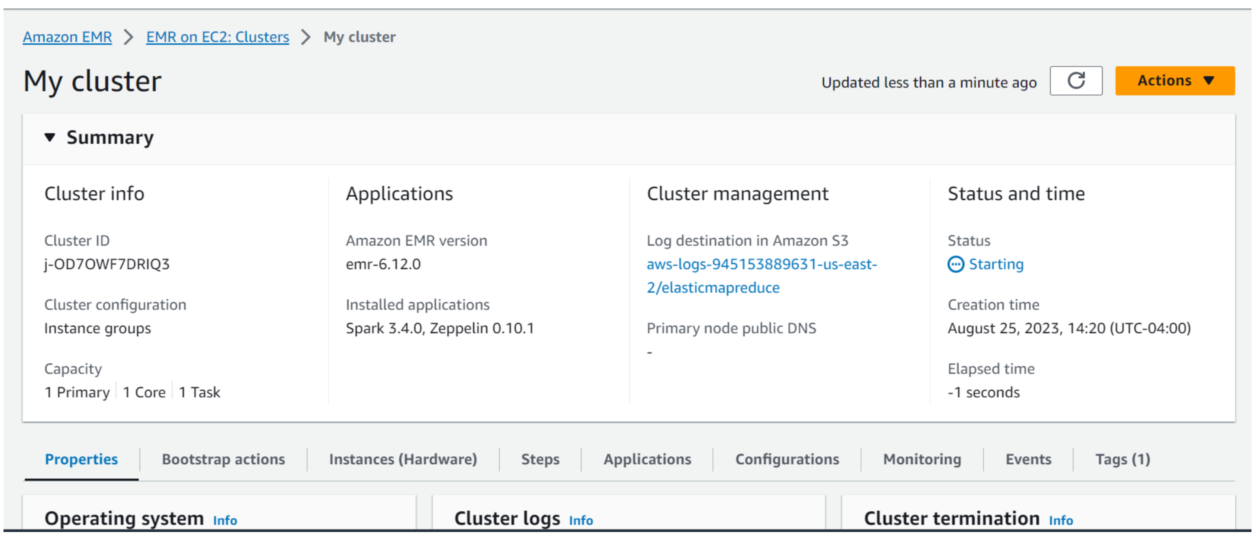


11. One can choose an existing instance profile or create an instance profile. If chosen, the option of create an instance profile, the Amazon EMR gives an option of specifying a custom set of

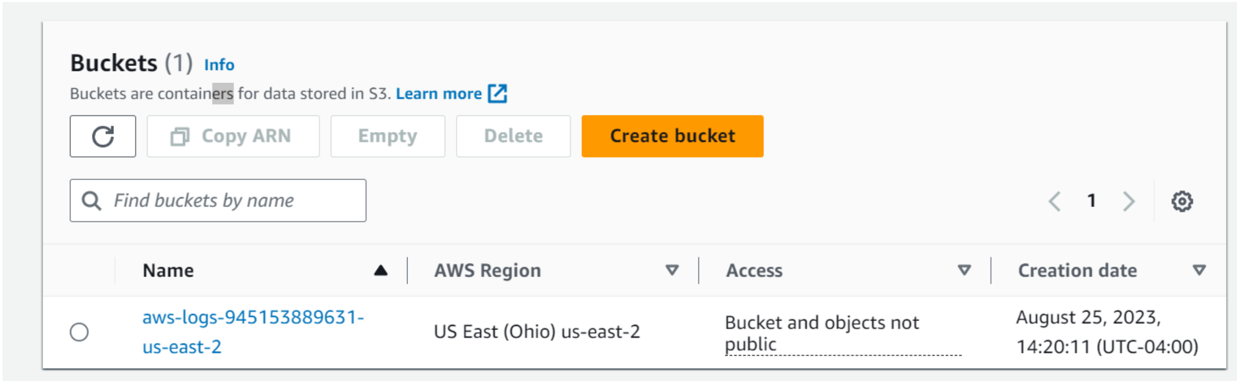
resources for the S3 bucket that will be created later on. It also provides the option of accessing all the buckets or the specific bucket that will be created with read and write access or giving only read access to the bucket



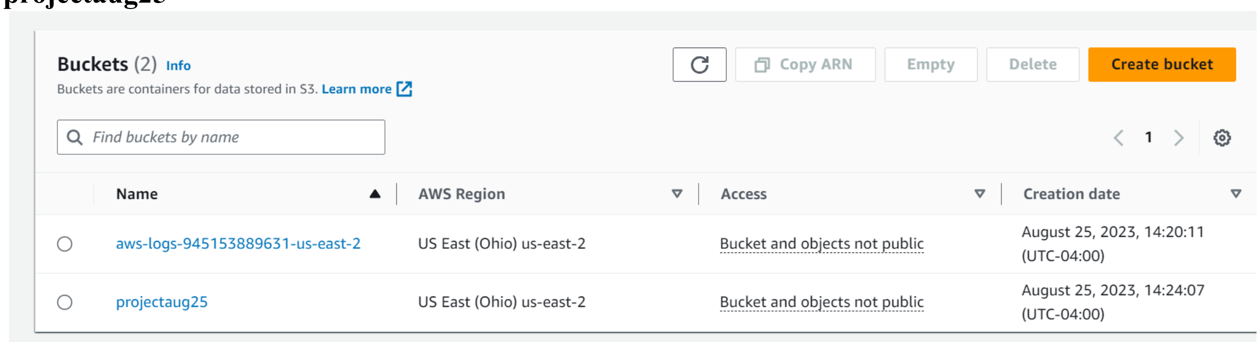
12. After completing all the steps, we can click on create cluster, and cluster is created as shown in the image below.



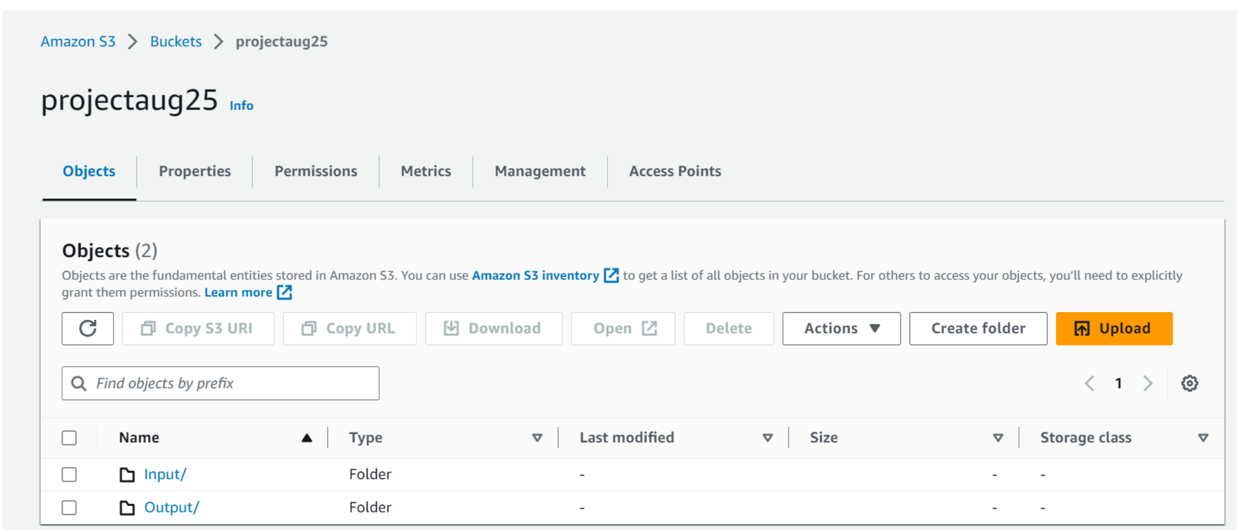
13. Create a bucket now by searching **S3** in search bar. It navigates to the page as shown below which shows the existing buckets if present.



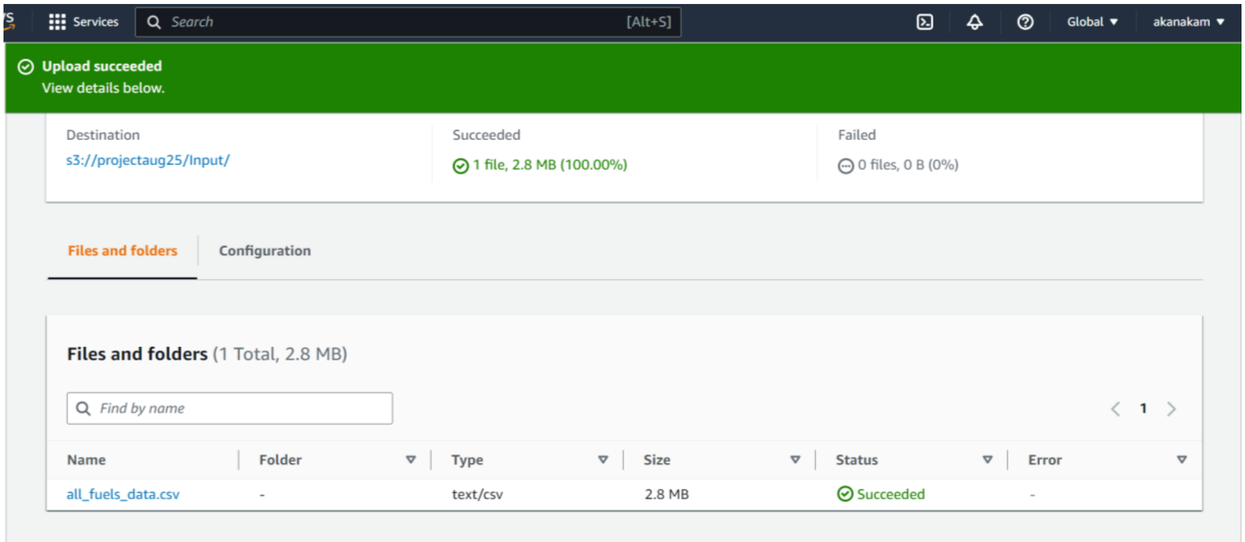
14. After clicking create bucket, one can write the name of the bucket and choose the option of giving access of the bucket to the public or not. As shown below, a new bucket is created **projectaug25**



15. After creating the bucket, we have to upload the data files. To upload the files, two folders are created Input and Output folders by clicking on Create folder



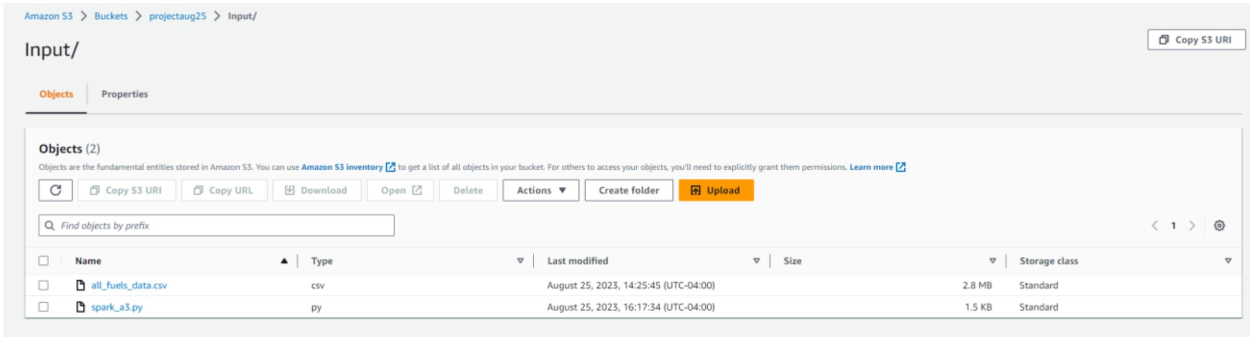
16. After clicking on upload button one can Upload the data file (all_fuels_data.csv) and python file for spark in Input folder.



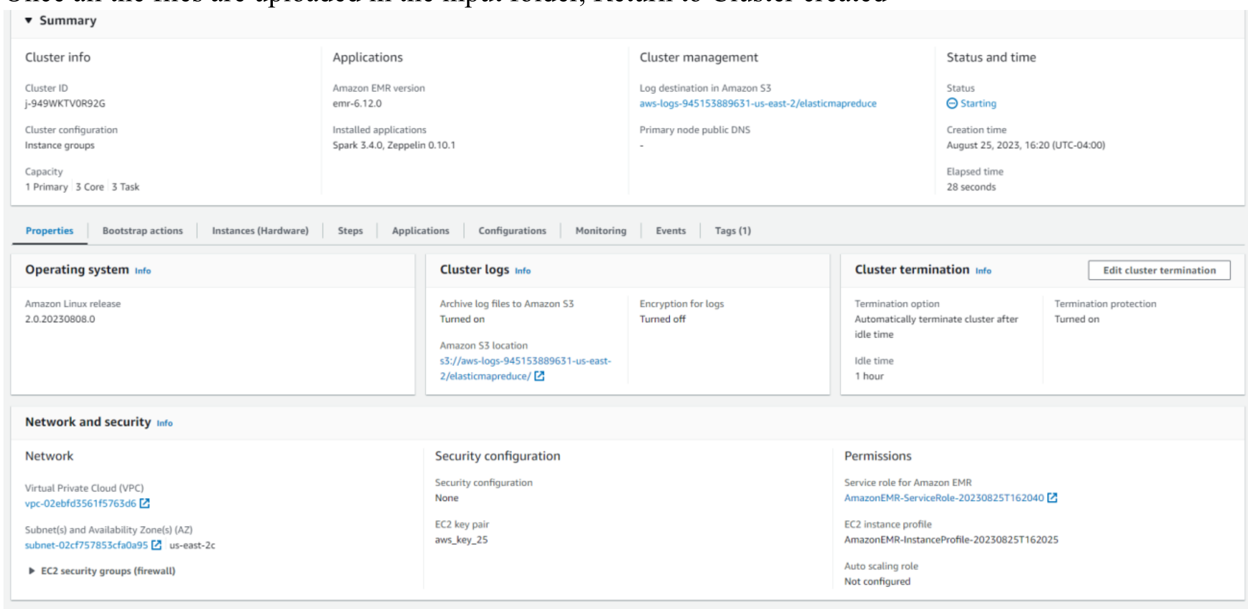
17. The spark script written is shown below:

```
from pyspark.sql import SparkSession
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.regression import RandomForestRegressor
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.sql.functions import col

# Initialize Spark session
spark = SparkSession.builder.appName("FuelPricePrediction").getOrCreate()

# Load data from CSV into a Spark DataFrame
data_path = "s3://projectaug25/Input/all_fuels_data.csv"
data_df = spark.read.csv(data_path, header=True, inferSchema=True)

# Select relevant columns and rename 'close' column to 'label'
selected_data = data_df.select("open", "high", "low", "volume", "close") \
                        .withColumnRenamed("close", "label")

# Data Preprocessing and Feature Engineering
assembler = VectorAssembler(inputCols=["open", "high", "low", "volume"], outputC
assembled_df = assembler.transform(selected_data)

# Split Data into Training and Testing Sets
train_ratio = 0.8
test_ratio = 1.0 - train_ratio
train_data, test_data = assembled_df.randomSplit([train_ratio, test_ratio], seed

# Build and Train a Machine Learning Model (Random Forest Regressor)
rf_regressor = RandomForestRegressor(featuresCol="features", labelCol="label", n
model = rf_regressor.fit(train_data)

# Make predictions on the test data
predictions = model.transform(test_data)

# Evaluate the model's performance
evaluator = RegressionEvaluator(labelCol="label", predictionCol="prediction", me
rmse = evaluator.evaluate(predictions)
output_file_path = "s3://projectaug25/Output/output.txt"
with open(output_file_path, "w") as f:
    f.write(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
```

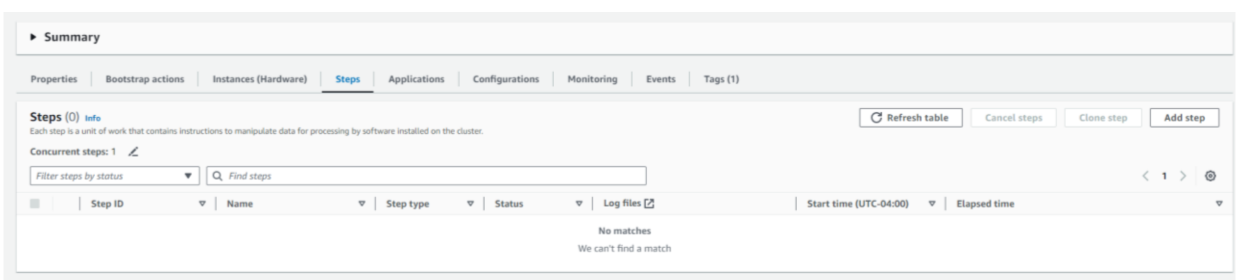
The spark script written is uploaded in the same input folder



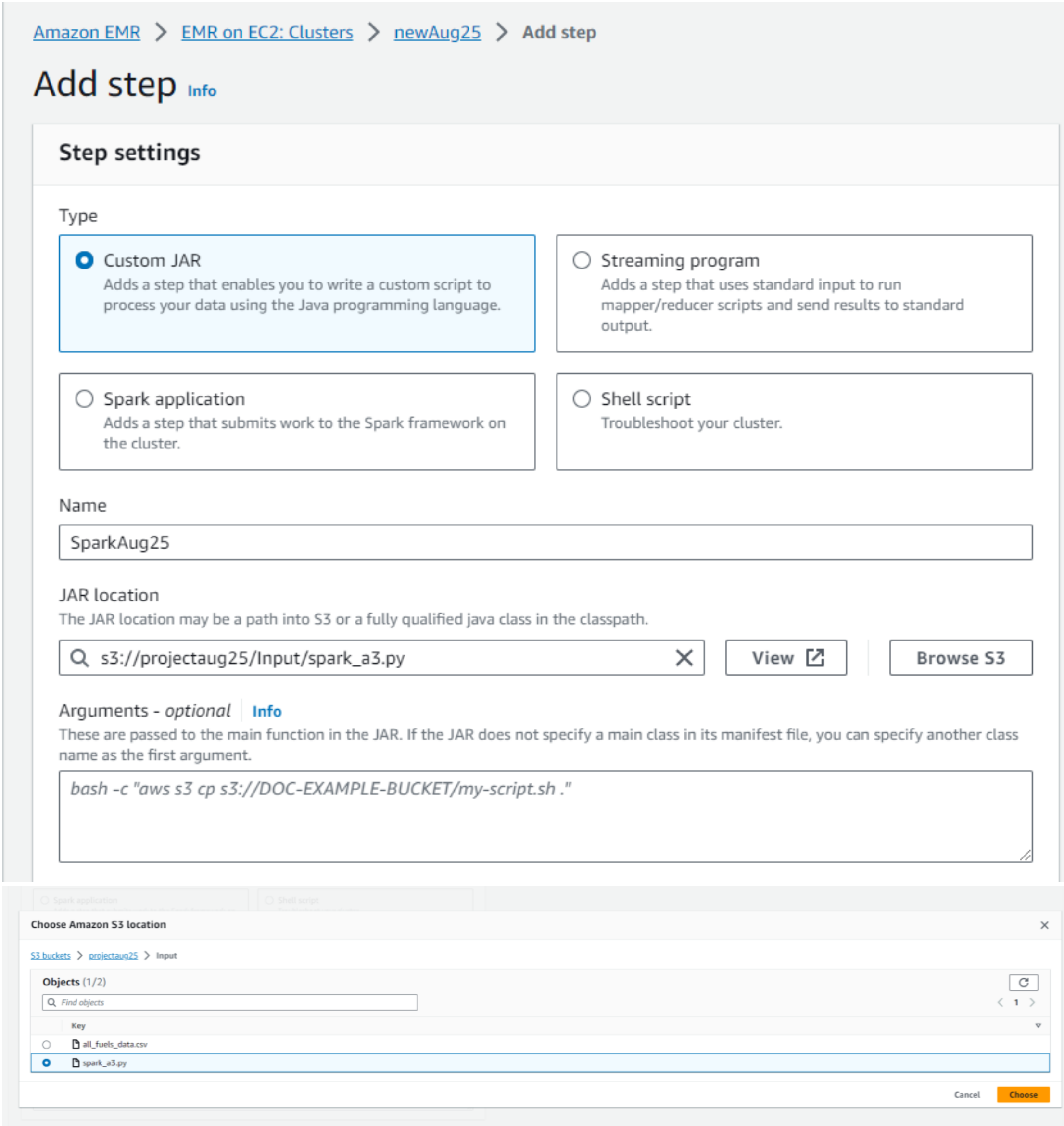
18. Once all the files are uploaded in the input folder, Return to Cluster created



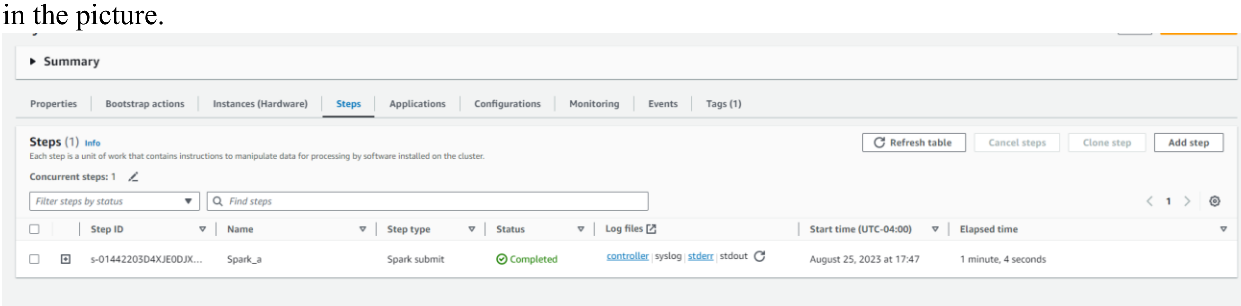
19. Go the steps shown in the menu bar



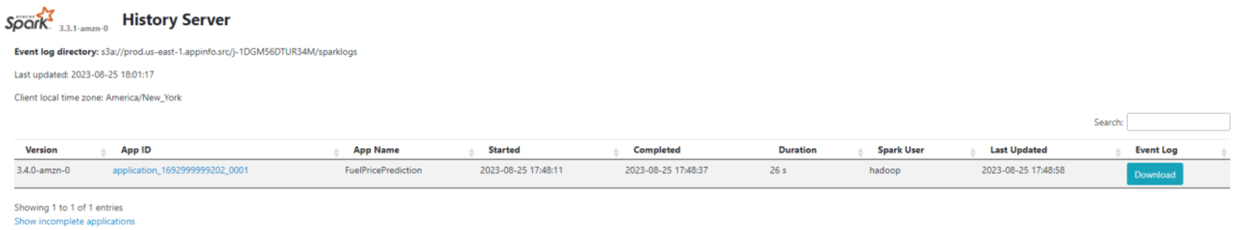
20. Add step, where you can choose the step settings by choosing **Spark application** , and give a name to the step as Spark_a / SparkAug25 and provide or set the JAR location to spark.py or the python script kept in the bucket.



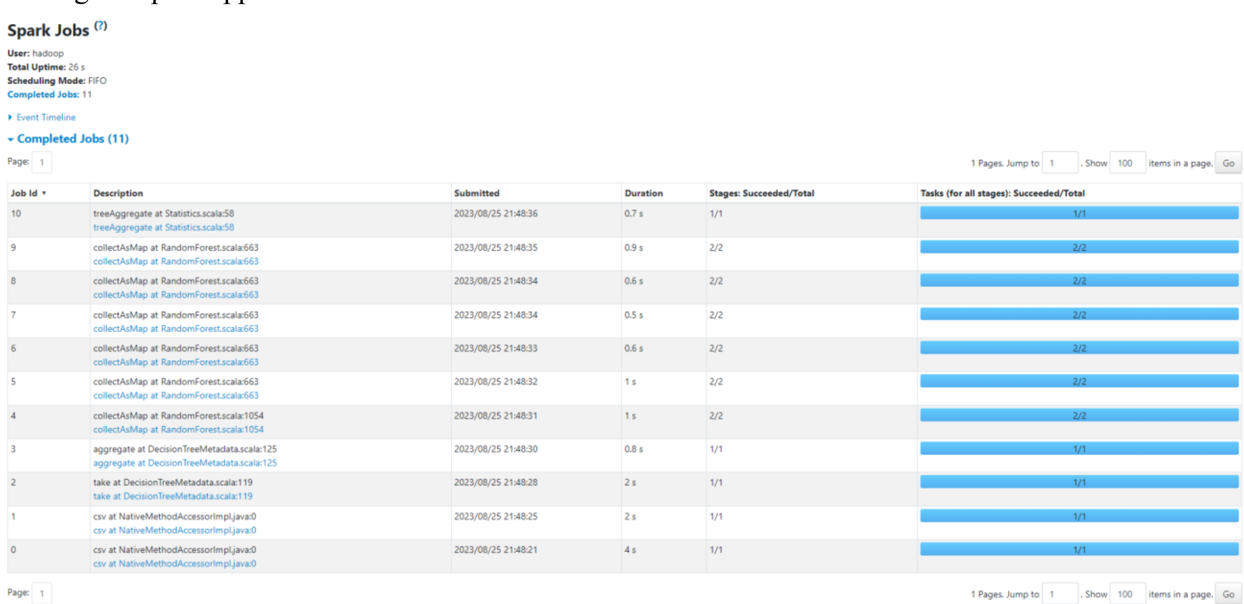
21. The step starts the process of running the script. Once completed its shown in the status as shown in the picture.



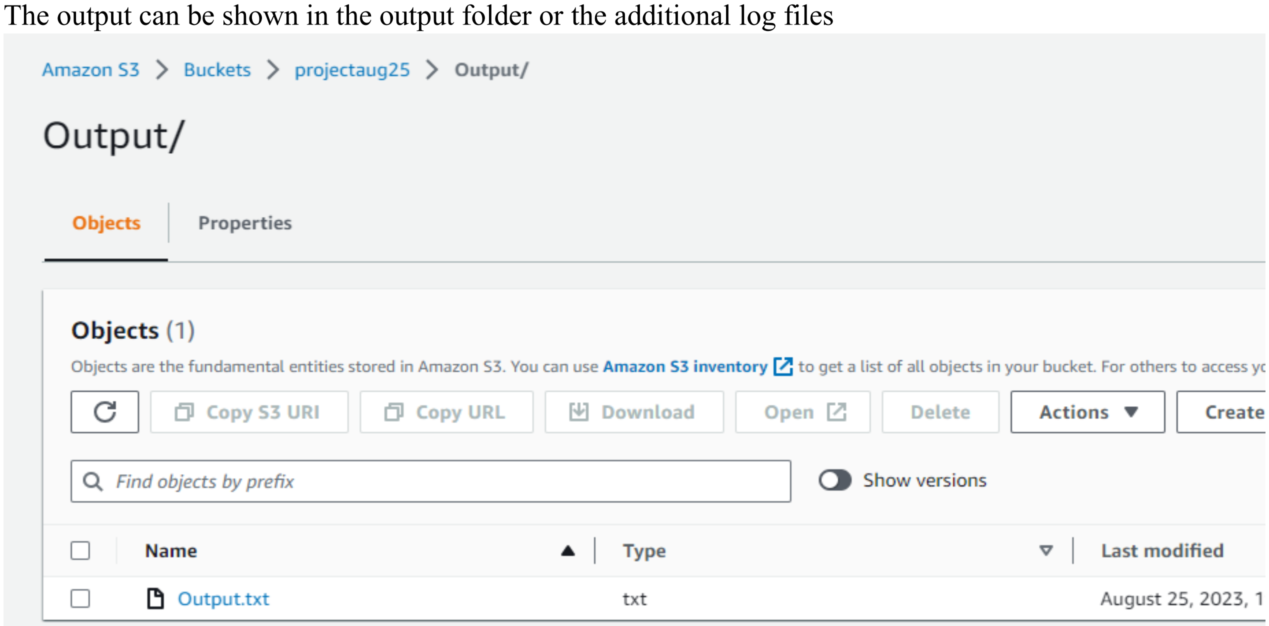
22. The details regarding the spark session can be seen in the history of the server.



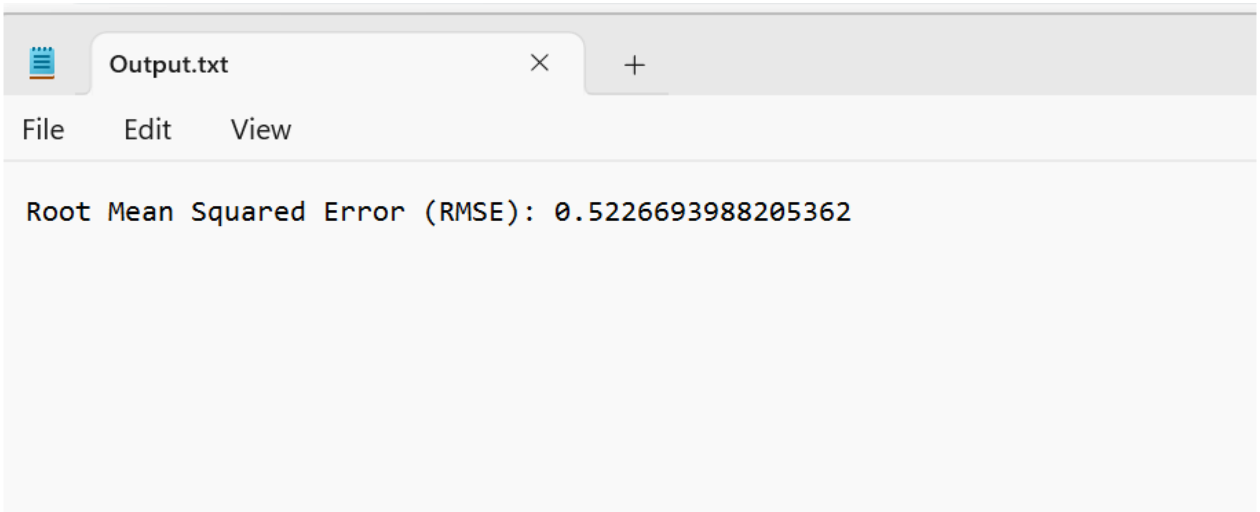
23. One can click on Application Id and look into the completed jobs and tasks that are completed for running the spark application



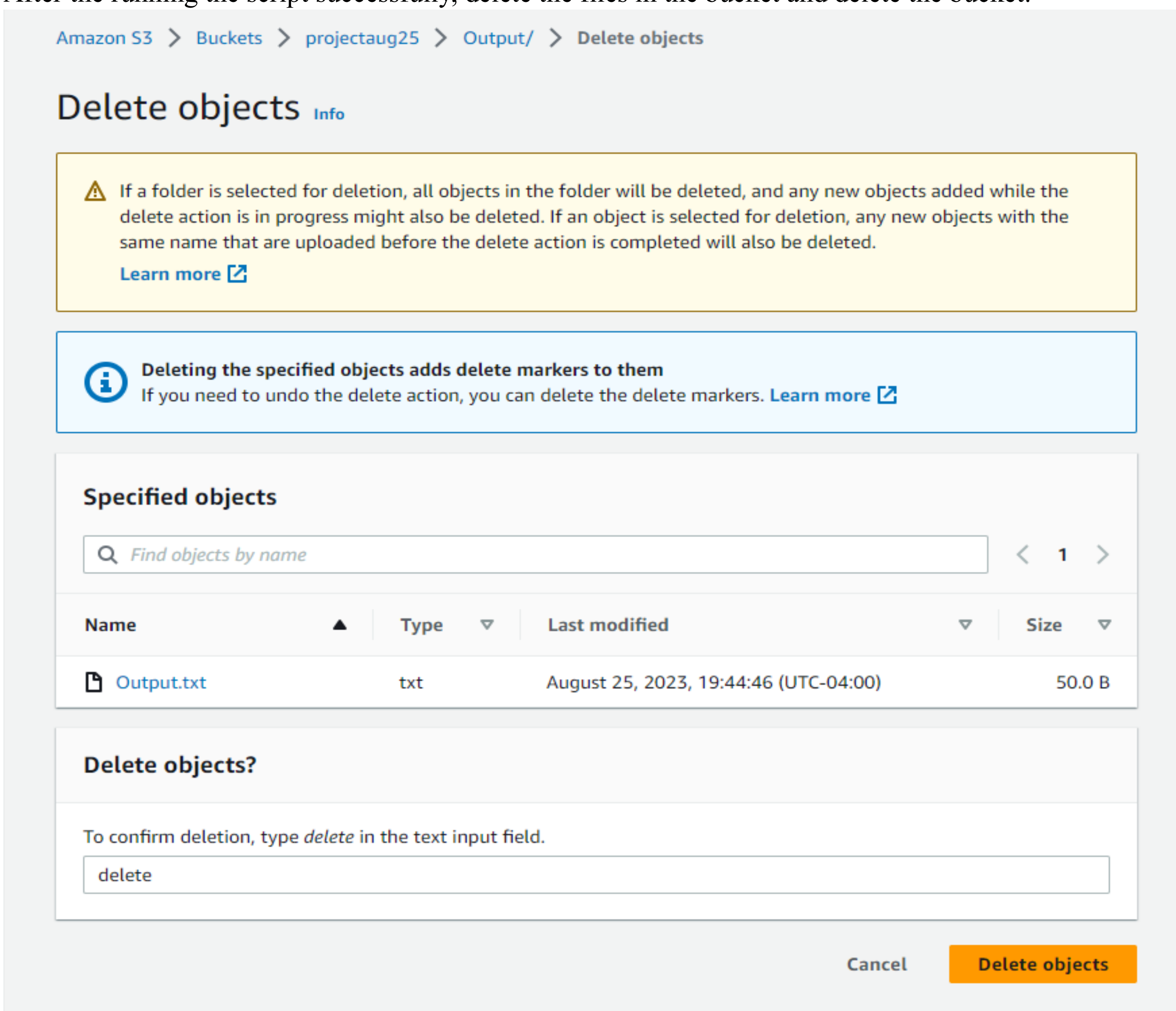
24. The output can be shown in the output folder or the additional log files

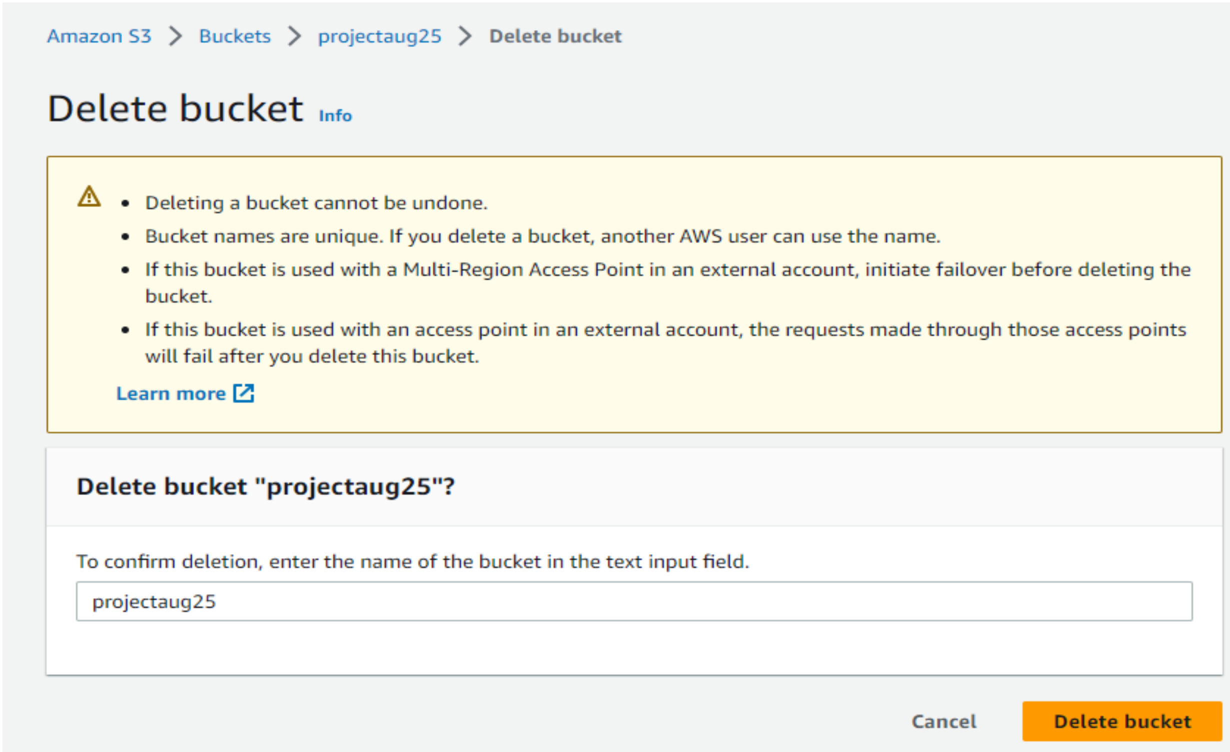


25. The output of the machine learning script is shown below

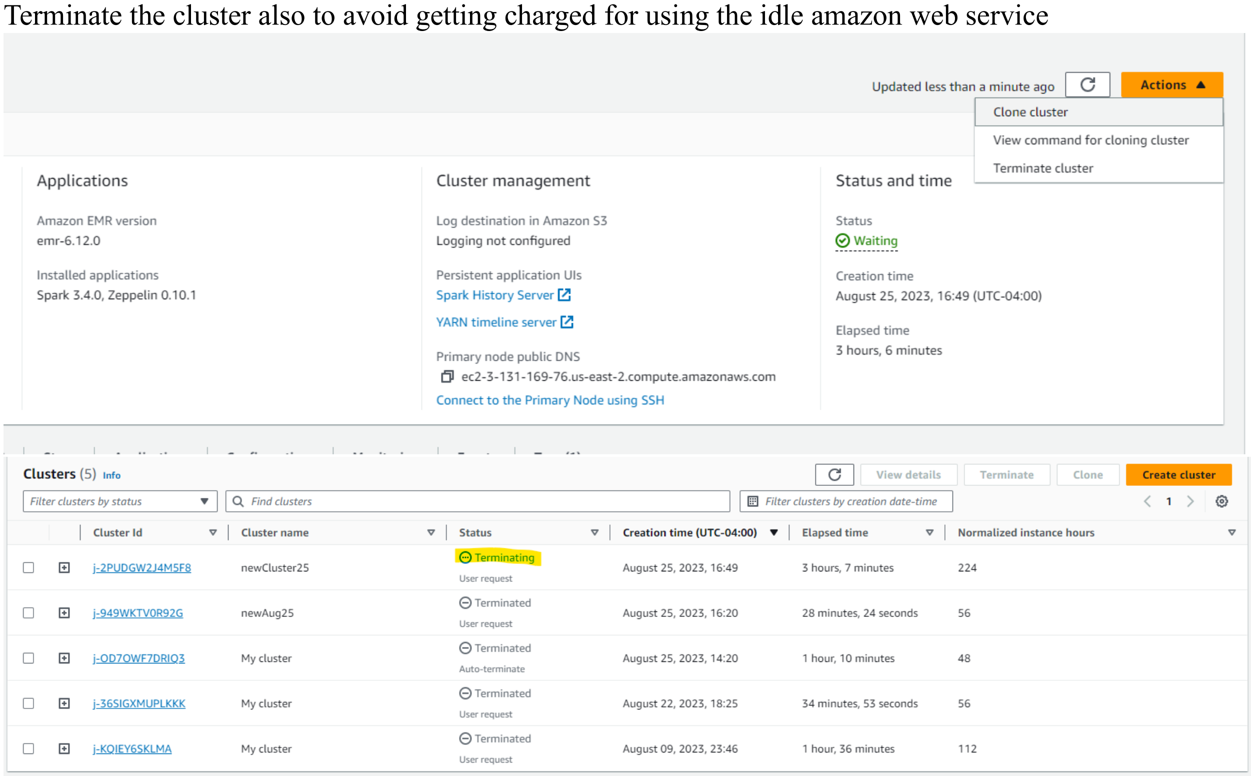


26. After the running the script successfully, delete the files in the bucket and delete the bucket.

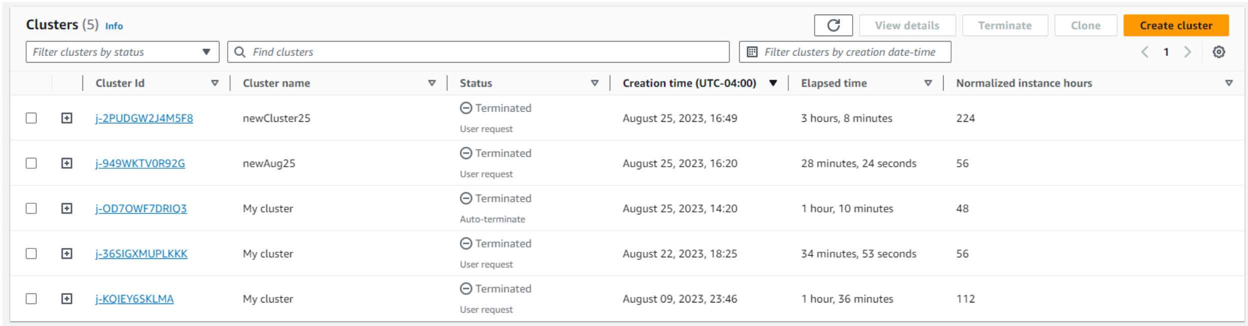




27. Terminate the cluster also to avoid getting charged for using the idle amazon web service



28. As shown the cluster is terminated



Clusters (5) Info

Cluster Id	Cluster name	Status	Creation time (UTC-04:00)	Elapsed time	Normalized instance hours
j-2PUDGW2J4M5F8	newCluster25	Terminated User request	August 25, 2023, 16:49	3 hours, 8 minutes	224
j-949WKTV0R92G	newAug25	Terminated User request	August 25, 2023, 16:20	28 minutes, 24 seconds	56
j-OD7OWF7DRIQ3	My cluster	Terminated Auto-terminate	August 25, 2023, 14:20	1 hour, 10 minutes	48
j-36SIGXMUPLKKK	My cluster	Terminated User request	August 22, 2023, 18:25	34 minutes, 53 seconds	56
j-KQIEY6SKLMA	My cluster	Terminated User request	August 09, 2023, 23:46	1 hour, 36 minutes	112

<u>CORE COMPONENTS OF THE PROJECT</u>

1. As mentioned the data needs to have more than 1000 rows, the data we got from Kaggle contains more than 27000 rows. The csv file is chosen as data lake as it presents structured data and easy to handle.
2. The data lake is uploaded or connected to AWS distributed cloud services. The choice of aws is primarily because of
    - The familiarity with the interface during the coursework and assignments given.
    - AWS has greater resources, infrastructure amd superior s, scalable services than Azure.
    - The pricing model of AWS is on hourly basis.
    - It has rich and uder friendly interface.
3. The objective was to predict the closing prices of fuel data by utilizing daily attributes such as fuel rates, consumption, and the high and low costs on the following day. This involved training the data and subsequently assessing the predictive capability. The evaluation encompassed computing the root mean square error or accuracy of the chosen model. The models under consideration included the random forest regressor as well as other machine learning approaches like linear regression. At the moment, the focus is on showcasing a single model, with the resultant output being stored within a text file housed in the S3 bucket.

All the step-by-step procedures have been mentioned in the file with pictures.

Thankyou.