Loading the datasets into an AWS RDS instance (Task 1)

Loading the datasets into an AWS RDS instance (Task 1)

Creation of RDS Instance

This document is purposely created for steps related to loading of datasets, however steps involved (including screenshots) for creating the EMR instance, creating the RDS instance and connecting RDS instance to EMR cluster are added in Annexure-I (at end of this document) for reference.

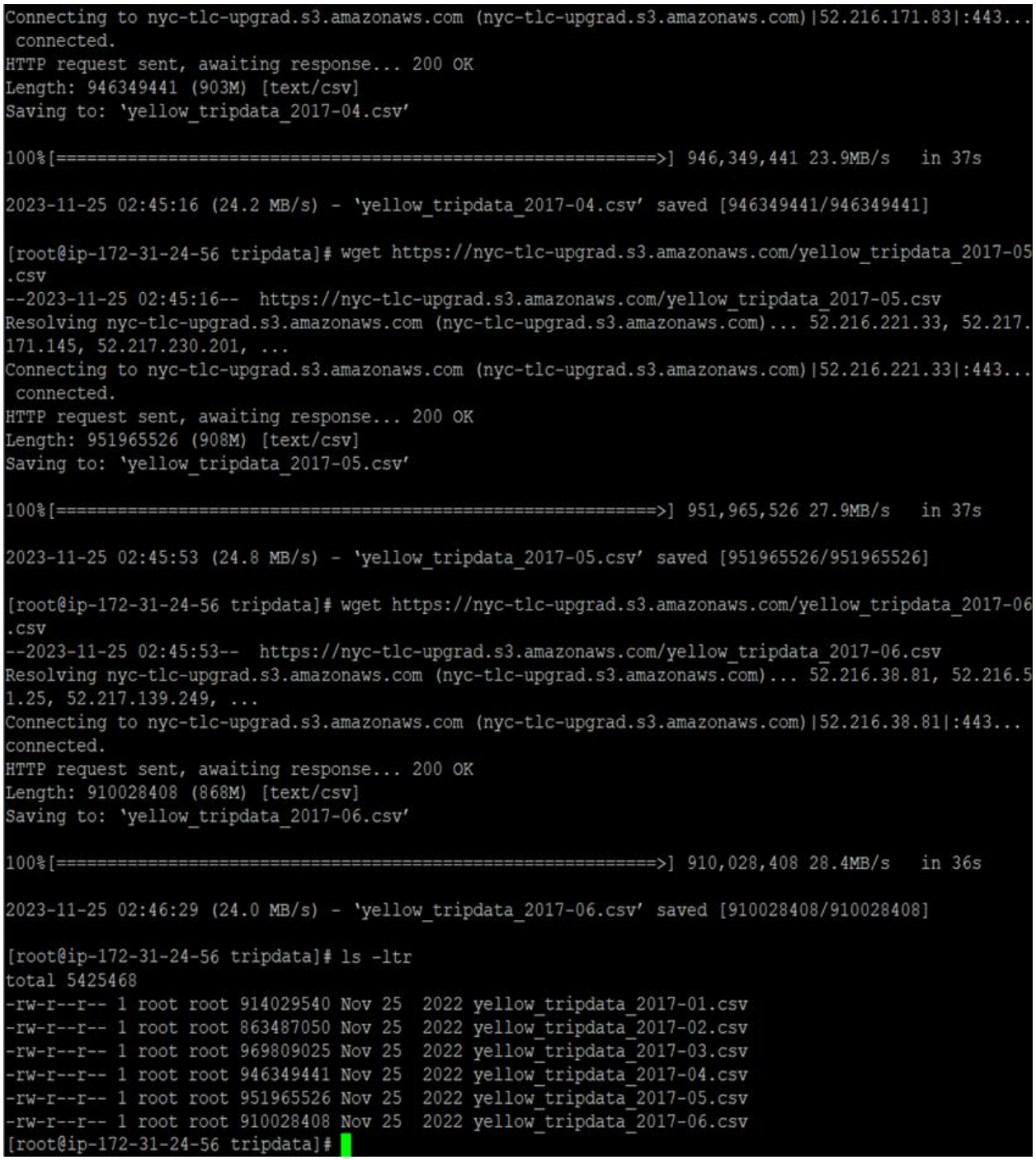
Downloading the datasets to master node of EMR cluster

Download the datasets using 'wget' command on EMR cluster's shell. This will download the files to local filesystem. For ease all the files are downloaded to a new folder "tripdata" in home directory.

```
sudo -i
cd
pwd
mkdir tripdata
cd tripdata
wget https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-01.csv
wget https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-02.csv
wget https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-03.csv
wget https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-04.csv
wget https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-05.csv
wget https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-06.csv
ls -ltr
wc -l yellow_tripdata_2017-*.csv
```

Screenshot of commands for downloading dataset

```
[root@ip-172-31-24-56 tripdata]# wget https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-01
.csv
--2023-11-25 02:42:51--  https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-01.csv
Resolving nyc-tlc-upgrad.s3.amazonaws.com (nyc-tlc-upgrad.s3.amazonaws.com)... 52.217.200.249, 52.217
.226.145, 54.231.193.249, ...
Connecting to nyc-tlc-upgrad.s3.amazonaws.com (nyc-tlc-upgrad.s3.amazonaws.com)|52.217.200.249|:443..
. connected.
HTTP request sent, awaiting response... 200 OK
Length: 914029540 (872M) [text/csv]
Saving to: 'yellow_tripdata_2017-01.csv'

100%[=============================================================>] 914,029,540 27.6MB/s   in 37s

2023-11-25 02:43:28 (23.6 MB/s) - 'yellow_tripdata_2017-01.csv' saved [914029540/914029540]

[root@ip-172-31-24-56 tripdata]# wget https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-02
.csv
--2023-11-25 02:43:28--  https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-02.csv
Resolving nyc-tlc-upgrad.s3.amazonaws.com (nyc-tlc-upgrad.s3.amazonaws.com)... 54.231.203.81, 54.231.
225.1, 54.231.228.121, ...
Connecting to nyc-tlc-upgrad.s3.amazonaws.com (nyc-tlc-upgrad.s3.amazonaws.com)|54.231.203.81|:443...
 connected.
HTTP request sent, awaiting response... 200 OK
Length: 863487050 (823M) [text/csv]
Saving to: 'yellow_tripdata_2017-02.csv'

100%[=============================================================>] 863,487,050 24.4MB/s   in 33s

2023-11-25 02:44:01 (24.9 MB/s) - 'yellow_tripdata_2017-02.csv' saved [863487050/863487050]

[root@ip-172-31-24-56 tripdata]# wget https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-03
.csv
--2023-11-25 02:44:01--  https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-03.csv
Resolving nyc-tlc-upgrad.s3.amazonaws.com (nyc-tlc-upgrad.s3.amazonaws.com)... 52.216.79.4, 52.217.86
.20, 52.217.191.25, ...
Connecting to nyc-tlc-upgrad.s3.amazonaws.com (nyc-tlc-upgrad.s3.amazonaws.com)|52.216.79.4|:443... c
onnected.
HTTP request sent, awaiting response... 200 OK
Length: 969809025 (925M) [text/csv]
Saving to: 'yellow_tripdata_2017-03.csv'

100%[=============================================================>] 969,809,025 24.4MB/s   in 37s

2023-11-25 02:44:38 (25.1 MB/s) - 'yellow_tripdata_2017-03.csv' saved [969809025/969809025]

[root@ip-172-31-24-56 tripdata]# wget https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-04
.csv
--2023-11-25 02:44:38--  https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-04.csv
Resolving nyc-tlc-upgrad.s3.amazonaws.com (nyc-tlc-upgrad.s3.amazonaws.com)... 52.216.171.83, 52.216.
```



Loading the datasets

Step 1. Initializing DNS names for RDS instance and EMR cluster

For the purpose of assignment, EMR cluster and RDS instance is created temporarily and doesn't have permanent DMS. Therefore, for ease of development and execution of required commands, two shell local variables are created before executing any command,

for storing DNS of RDS instance and EMR cluster, by using below commands on shell of EMR cluster:

```
DNS_EMR="ec2-100-26-47-42.compute-1.amazonaws.com"
DNS_RDS="tripdb.cl3grgmbpn1n.us-east-1.rds.amazonaws.com"
echo $DNS_EMR
echo $DNS_RDS
```

This will also help to test the assignment code/commands, as there will not be any requirement of changing DNS in each of command separately.

Screenshot of local variable initialization

Step 2. Login to RDS instance from EMR cluster

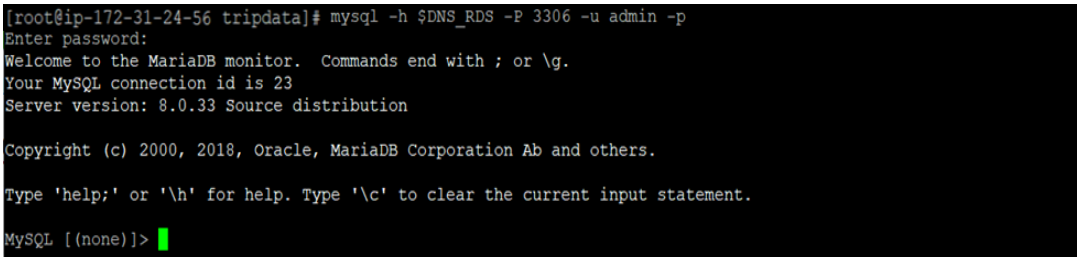
Login to RDS instance of MySQL using the credentials used to create the database service:

Username : admin
Password : user1234

```
mysql -h $DNS_RDS -P 3306 -u admin -p
```

Enter the password which is mentioned above, when prompt for it.

Screenshot for connecting to database



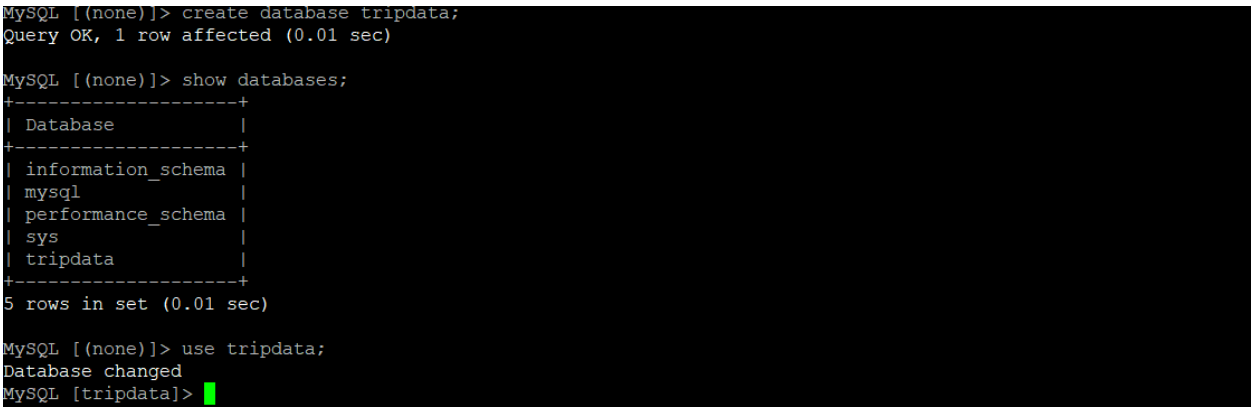
Step 3. Creation of database

Following actions are taken:

- Create new database in MySQL instance with name “tripdata”.
- Validate database has been created by using ‘show databases’ command. It should include “tripdata” as database along with few default databases came with MySQL for administration and demonstration purposes.
- Switch to new database created, to ensure that all DDLs and DMLs run on “tripdata” database.

```
create database tripdata;
show databases;
use tripdata;
```

Screenshot of commands for creating database



Step 4. Creation of table schema

Few important observations/actions about datasets: -

- Data is not clean and many fields contain outliers. For example, fields like “fare_amount”, “total_amount” contains negative (< 0) values, which cannot be a valid amount. Again, for categorical variables like “payment_type” value contains other than 1 to 6, i.e., 99, which may be representing the missing value. However, As EDA analysis is not part of assignment, currently no data cleaning is performed, and data is imported in RDS instance without cleaning.
- For all decimal fields range is taken as (16, 4), which means it can store upto 4 decimal places and 12-digit decimal number. This is because the range of data field cannot be determined due to presence of outlier.
- Only columns “congestion_surcharge” and “airport_fee” contains null value.

- A field “RECORD_ID” has been added as auto incremental integer primary key, to uniquely identify each record.

A table with following columns and datatype is created:

#	Column Name	Data Type	Is Null	Column Description
1	RECORD_ID	VARCHAR	N	A Primary Key is created which is combination of "SOURCE" + "YEAR" + "MONTH" + "ROW_NUMBER".
2	VendorID	INT	N	A code indicating the TPEP provider that provided the record. 1= Creative Mobile Technologies, LLC; 2= VeriFone Inc.
3	tpep_pickup_datetime	DATETIME	N	The date and time when the meter was engaged.
4	tpep_dropoff_datetime	DATETIME	N	The date and time when the meter was disengaged.
5	passenger_count	INT	N	The number of passengers in the vehicle. This is a driver-entered value.
6	trip_distance	DECIMAL	N	The elapsed trip distance in miles reported by the taximeter.
7	RatecodeID	INT	N	The final rate code in effect at the end of the trip. **1= Standard rate** **2=JFK** **3=Newark** **4=Nassau or Westchester** **5=Negotiated fare** **6=Group ride** Also, it is found that it contains value 99.
8	store_and_fwd_flag	VARCHAR	N	This flag indicates whether the trip record was held in vehicle memory before sending to the vendor, aka “store and forward,” because the vehicle did not have a connection to the server. **Y= store and forward trip** **N= not a store and forward trip**
9	PULocationID	INT	N	TLC Taxi Zone in which the taximeter was engaged.
10	DOLocationID	INT	N	TLC Taxi Zone in which the taximeter was disengaged.
11	payment_type	INT	N	A numeric code signifying how the passenger paid for the trip. **1= Credit card** **2= Cash** **3= No charge** **4= Dispute** **5= Unknown** **6= Voided trip**

12	fare_amount	DECIMAL	N	The time-and-distance fare calculated by the meter. Few outliers like negative values are observed.
13	extra	DECIMAL	N	Miscellaneous extras and surcharges. Currently, this only includes the $0.50 and $1 rush hour and overnight charges.
14	mta_tax	DECIMAL	N	$0.50 MTA tax that is automatically triggered based on the metered rate in use.
15	tip_amount	DECIMAL	N	Tip amount – This field is automatically populated for credit card tips. Cash tips are not included.
16	tolls_amount	DECIMAL	N	Total amount of all tolls paid in trip.
17	improvement_surcharge	DECIMAL	N	$0.30 improvement surcharge assessed trips at the flag drop. The improvement surcharge began being levied in 2015.
18	total_amount	DECIMAL	N	The total amount charged to passengers. Does not include cash tips.
19	congestion_surcharge	DECIMAL	Y	Total amount collected in trip for NYS congestion surcharge.
20	airport_fee	DECIMAL	Y	$1.25 for pick up only at LaGuardia and John F. Kennedy Airports

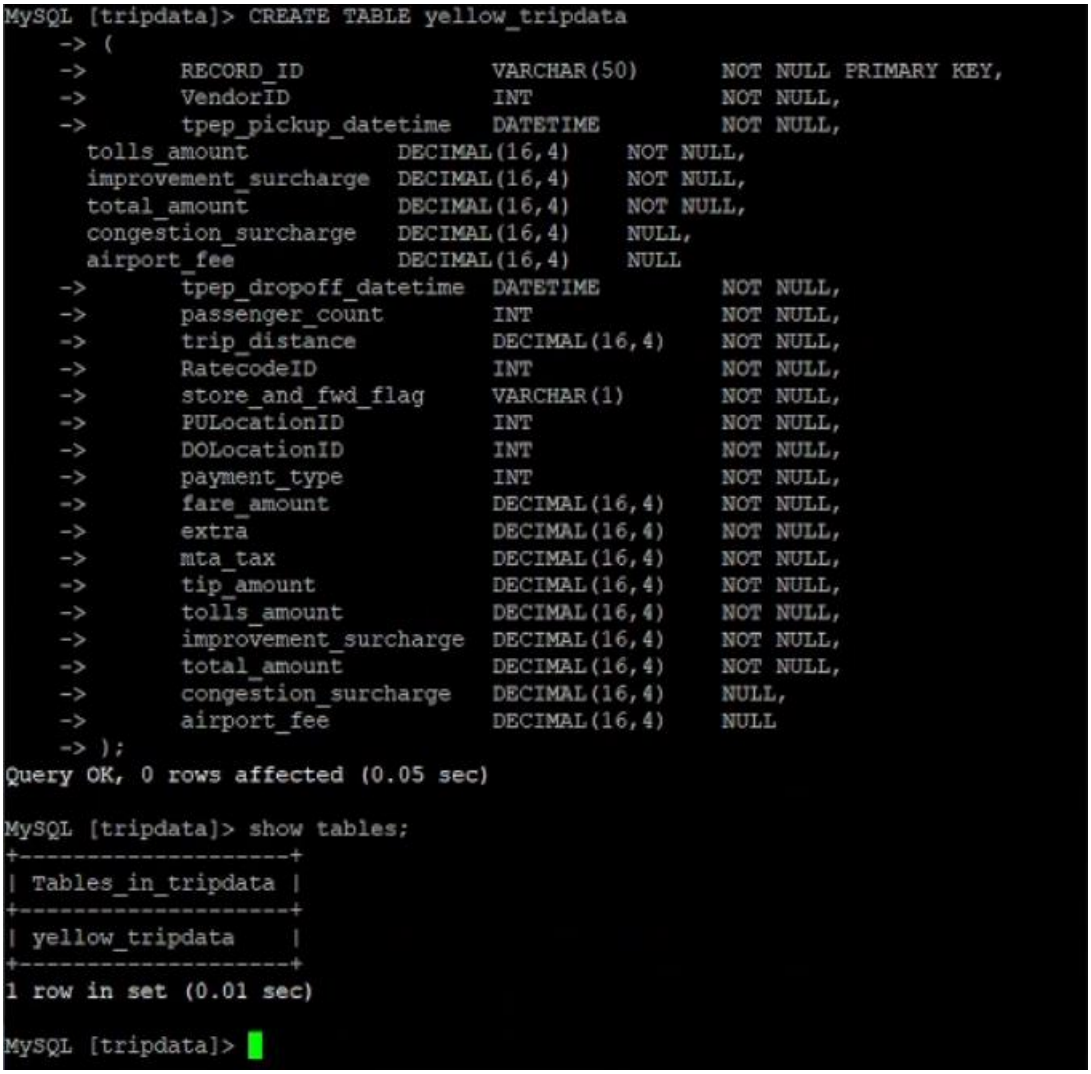
Below DDL is used to create the desired table to store data: -

```
CREATE TABLE yellow_tripdata
(
      RECORD_ID              VARCHAR(50)      NOT NULL PRIMARY KEY,
      VendorID               INT              NOT NULL,
      tpep_pickup_datetime   DATETIME         NOT NULL,
      tpep_dropoff_datetime  DATETIME         NOT NULL,
      passenger_count        INT              NOT NULL,
      trip_distance          DECIMAL(16,4)    NOT NULL,
      RatecodeID             INT              NOT NULL,
      store_and_fwd_flag     VARCHAR(1)       NOT NULL,
      PULocationID           INT              NOT NULL,
      DOLocationID           INT              NOT NULL,
      payment_type           INT              NOT NULL,
      fare_amount            DECIMAL(16,4)    NOT NULL,
      extra                  DECIMAL(16,4)    NOT NULL,
      mta_tax                DECIMAL(16,4)    NOT NULL,
      tip_amount             DECIMAL(16,4)    NOT NULL,
      tolls_amount           DECIMAL(16,4)    NOT NULL,
      improvement_surcharge  DECIMAL(16,4)    NOT NULL,
      total_amount           DECIMAL(16,4)    NOT NULL,
      congestion_surcharge   DECIMAL(16,4)    NULL,
      airport_fee            DECIMAL(16,4)    NULL
);
```

After above SQL, table should be created on database. To validate checked through below SQL command:

```
show tables;
```

Screenshot of commands for creating table



Step 5. Global variable for importing data from file

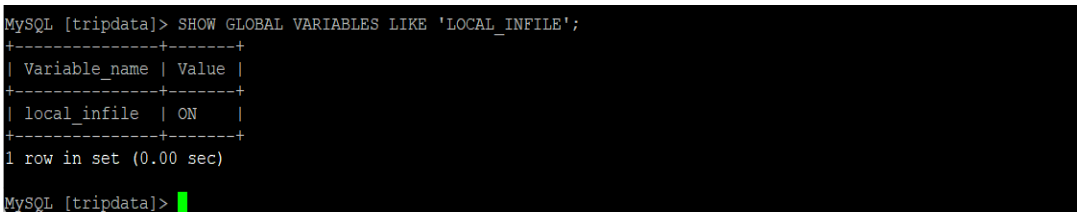
By default, in MySQL importing data from local client is disabled. Before loading dataset, we need to ensure that flag “LOCAL_INFILE” in MySQL is set to TRUE. To check the value, following command is executed on MySQL:

```
SHOW GLOBAL VARIABLES LIKE 'LOCAL_INFILE';
```

If above command doesn't show the flag value as "TRUE" or "ON", then following command is run to enable the flag:

```
SET GLOBAL LOCAL_INFILE=TRUE;
```

Screenshot of global flag set



Step 6. Loading the dataset

MySQL support loading a table using local CSV or other files. This will directly load the table. Any datatype conversion can also be performed inline.

Following command for setting MySQL variable for row number to 0 to help creating PK:

```
set @myrow_numb=0;
```

Following commands are used to load file in database:

```
LOAD DATA LOCAL INFILE '/root/tripdata/yellow_tripdata_2017-01.csv'
INTO TABLE yellow_tripdata
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(     @VendorID, @tpep_pickup_datetime, @tpep_dropoff_datetime,
      @passenger_count, @trip_distance, @RatecodeID, @store_and_fwd_flag,
      @PULocationID, @DOLocationID, @payment_type, @fare_amount, @extra,
      @mta_tax, @tip_amount, @tolls_amount, @improvement_surcharge,
      @total_amount, @congestion_surcharge, @airport_fee
)
SET
RECORD_ID            = CONCAT('MYSQL-', SUBSTRING(@tpep_pickup_datetime, 1, 8),
LPAD(@myrow_numb:=@myrow_numb+1, 10, '0')),
VendorID             = @VendorID,
tpep_pickup_datetime = STR_TO_DATE(@tpep_pickup_datetime,'%Y-%m-%d %H:%i:%s'),
tpep_dropoff_datetime= STR_TO_DATE(@tpep_dropoff_datetime,'%Y-%m-%d %H:%i:%s'),
passenger_count      = @passenger_count,
trip_distance        = CAST(@trip_distance AS CHAR)+0,
RatecodeID           = @RatecodeID,
store_and_fwd_flag   = @store_and_fwd_flag,
PULocationID         = @PULocationID,
DOLocationID         = @DOLocationID,
payment_type         = @payment_type,
fare_amount          = CAST(@fare_amount AS CHAR)+0,
extra                = CAST(@extra AS CHAR)+0,
mta_tax              = CAST(@mta_tax AS CHAR)+0,
tip_amount           = CAST(@tip_amount AS CHAR)+0,
tolls_amount         = CAST(@tolls_amount AS CHAR)+0,
improvement_surcharge= CAST(@improvement_surcharge AS CHAR)+0,
total_amount         = CAST(@total_amount AS CHAR)+0,
congestion_surcharge = CASE WHEN @congestion_surcharge = '' THEN NULL ELSE
CAST(@congestion_surcharge AS CHAR)+0 END,
airport_fee          = CASE WHEN TRIM(REPLACE(REPLACE(@airport_fee, '\r', ''),
'\n', '')) = '' THEN NULL ELSE CAST(TRIM(REPLACE(REPLACE(@airport_fee, '\r',
''), '\n', '')) AS CHAR)+0 END;
```

```
LOAD DATA LOCAL INFILE '/root/tripdata/yellow_tripdata_2017-02.csv'
INTO TABLE yellow_tripdata
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(     @VendorID, @tpep_pickup_datetime, @tpep_dropoff_datetime,
      @passenger_count, @trip_distance, @RatecodeID, @store_and_fwd_flag,
      @PULocationID, @DOLocationID, @payment_type, @fare_amount, @extra,
      @mta_tax, @tip_amount, @tolls_amount, @improvement_surcharge,
      @total_amount, @congestion_surcharge, @airport_fee
)
SET
RECORD_ID            = CONCAT('MYSQL-', SUBSTRING(@tpep_pickup_datetime, 1, 8),
LPAD(@myrow_numb:=@myrow_numb+1, 10, '0')),
VendorID             = @VendorID,
tpep_pickup_datetime = STR_TO_DATE(@tpep_pickup_datetime,'%Y-%m-%d %H:%i:%s'),
tpep_dropoff_datetime= STR_TO_DATE(@tpep_dropoff_datetime,'%Y-%m-%d %H:%i:%s'),
passenger_count      = @passenger_count,
trip_distance        = CAST(@trip_distance AS CHAR)+0,
RatecodeID           = @RatecodeID,
store_and_fwd_flag   = @store_and_fwd_flag,
PULocationID         = @PULocationID,
DOLocationID         = @DOLocationID,
payment_type         = @payment_type,
fare_amount          = CAST(@fare_amount AS CHAR)+0,
extra                = CAST(@extra AS CHAR)+0,
mta_tax              = CAST(@mta_tax AS CHAR)+0,
tip_amount           = CAST(@tip_amount AS CHAR)+0,
tolls_amount         = CAST(@tolls_amount AS CHAR)+0,
improvement_surcharge= CAST(@improvement_surcharge AS CHAR)+0,
total_amount         = CAST(@total_amount AS CHAR)+0,
congestion_surcharge = CASE WHEN @congestion_surcharge = '' THEN NULL ELSE
CAST(@congestion_surcharge AS CHAR)+0 END,
airport_fee          = CASE WHEN TRIM(REPLACE(REPLACE(@airport_fee, '\r', ''),
'\n', '')) = '' THEN NULL ELSE CAST(TRIM(REPLACE(REPLACE(@airport_fee, '\r',
''), '\n', '')) AS CHAR)+0 END;
```

Screenshot of loading the dataset in RDS

```
MySQL [tripdata]> LOAD DATA LOCAL INFILE '/root/tripdata/yellow_tripdata_2017-01
.csv'
    -> INTO TABLE yellow_tripdata
    -> FIELDS TERMINATED BY ','
    -> LINES TERMINATED BY '\n'
    -> IGNORE 1 LINES
    -> (      @VendorID, @tpep_pickup_datetime, @tpep_dropoff_datetime,
    ->        @passenger_count, @trip_distance, @RatecodeID, @store_and_fwd_flag,
    ->        @PULocationID, @DOLocationID, @payment_type, @fare_amount, @extra,
    ->        @mta_tax, @tip_amount, @tolls_amount, @improvement_surcharge,
    ->        @total_amount, @congestion_surcharge, @airport_fee
    -> )
    -> SET
    -> RECORD_ID            = CONCAT('MYSQL-', SUBSTRING(@tpep_pickup_datetime,
1, 8), LPAD(@myrow_numb:=@myrow_numb+1, 10, '0')),
    -> VendorID             = @VendorID,
    -> tpep_pickup_datetime = STR_TO_DATE(@tpep_pickup_datetime,'%Y-%m-%d %H:%i:
%s'),
    -> tpep_dropoff_datetime= STR_TO_DATE(@tpep_dropoff_datetime,'%Y-%m-%d %H:%i
:%s'),
    -> passenger_count      = @passenger_count,
    -> trip_distance        = CAST(@trip_distance AS CHAR)+0,
    -> RatecodeID           = @RatecodeID,
    -> store_and_fwd_flag   = @store_and_fwd_flag,
    -> PULocationID         = @PULocationID,
    -> DOLocationID         = @DOLocationID,
    -> payment_type         = @payment_type,
    -> fare_amount          = CAST(@fare_amount AS CHAR)+0,
    -> extra                = CAST(@extra AS CHAR)+0,
    -> mta_tax              = CAST(@mta_tax AS CHAR)+0,
    -> tip_amount           = CAST(@tip_amount AS CHAR)+0,
    -> tolls_amount         = CAST(@tolls_amount AS CHAR)+0,
    -> improvement_surcharge= CAST(@improvement_surcharge AS CHAR)+0,
    -> total_amount         = CAST(@total_amount AS CHAR)+0,
    -> congestion_surcharge = CASE WHEN @congestion_surcharge = '' THEN NULL ELS
E CAST(@congestion_surcharge AS CHAR)+0 END,
    -> airport_fee          = CASE WHEN TRIM(REPLACE(REPLACE(@airport_fee, '\r',
 ''), '\n', '')) = '' THEN NULL ELSE CAST(TRIM(REPLACE(REPLACE(@airport_fee, '\r
', ''), '\n', '')) AS CHAR)+0 END;
Query OK, 9710820 rows affected, 1 warning (6 min 6.68 sec)
Records: 9710820  Deleted: 0  Skipped: 0  Warnings: 1

MySQL [tripdata]>
```

```
MySQL [tripdata]> LOAD DATA LOCAL INFILE '/root/tripdata/yellow_tripdata_2017-02
.csv'
    -> INTO TABLE yellow_tripdata
    -> FIELDS TERMINATED BY ','
    -> LINES TERMINATED BY '\n'
    -> IGNORE 1 LINES
    -> (     @VendorID, @tpep_pickup_datetime, @tpep_dropoff_datetime,
    ->       @passenger_count, @trip_distance, @RatecodeID, @store_and_fwd_flag,
    ->       @PULocationID, @DOLocationID, @payment_type, @fare_amount, @extra,
    ->       @mta_tax, @tip_amount, @tolls_amount, @improvement_surcharge,
    ->       @total_amount, @congestion_surcharge, @airport_fee
    -> )
    -> SET
    -> RECORD_ID            = CONCAT('MYSQL-', SUBSTRING(@tpep_pickup_datetime,
1, 8), LPAD(@myrow_numb:=@myrow_numb+1, 10, '0')),
    -> VendorID             = @VendorID,
    -> tpep_pickup_datetime = STR_TO_DATE(@tpep_pickup_datetime,'%Y-%m-%d %H:%i:
%s'),
    -> tpep_dropoff_datetime= STR_TO_DATE(@tpep_dropoff_datetime,'%Y-%m-%d %H:%i
:%s'),
    -> passenger_count      = @passenger_count,
    -> trip_distance        = CAST(@trip_distance AS CHAR)+0,
    -> RatecodeID           = @RatecodeID,
    -> store_and_fwd_flag   = @store_and_fwd_flag,
    -> PULocationID         = @PULocationID,
    -> DOLocationID         = @DOLocationID,
    -> payment_type         = @payment_type,
    -> fare_amount          = CAST(@fare_amount AS CHAR)+0,
    -> extra                = CAST(@extra AS CHAR)+0,
    -> mta_tax              = CAST(@mta_tax AS CHAR)+0,
    -> tip_amount           = CAST(@tip_amount AS CHAR)+0,
    -> tolls_amount         = CAST(@tolls_amount AS CHAR)+0,
    -> improvement_surcharge= CAST(@improvement_surcharge AS CHAR)+0,
    -> total_amount         = CAST(@total_amount AS CHAR)+0,
    -> congestion_surcharge = CASE WHEN @congestion_surcharge = '' THEN NULL ELS
E CAST(@congestion_surcharge AS CHAR)+0 END,
    -> airport_fee          = CASE WHEN TRIM(REPLACE(REPLACE(@airport_fee, '\r',
 ''), '\n', '')) = '' THEN NULL ELSE CAST(TRIM(REPLACE(REPLACE(@airport_fee, '\r
', ''), '\n', '')) AS CHAR)+0 END;
Query OK, 9169775 rows affected, 1 warning (6 min 41.17 sec)
Records: 9169775  Deleted: 0  Skipped: 0  Warnings: 1

MySQL [tripdata]>
```

Step 7. Verifying the imported data

The data in MySQL has been loaded, we can verify the record count of inserted data. Following commands to check the total rows imported in table:

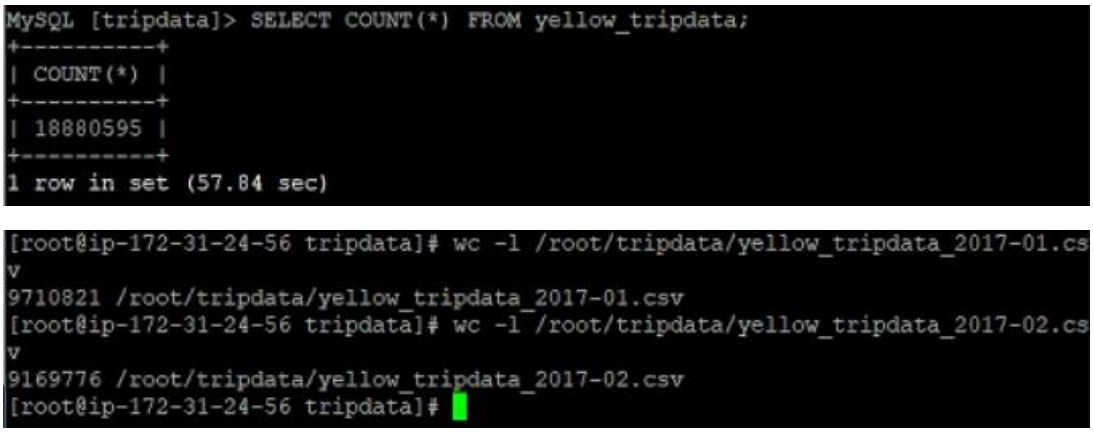
```
SELECT COUNT(*) FROM yellow_tripdata;
```

```
quit;
```

Also, check the total records in two CSV files on local file systems:

```
wc -l /root/tripdata/yellow_tripdata_2017-01.csv
wc -l /root/tripdata/yellow_tripdata_2017-02.csv
```

Screenshot of verification of the dataset

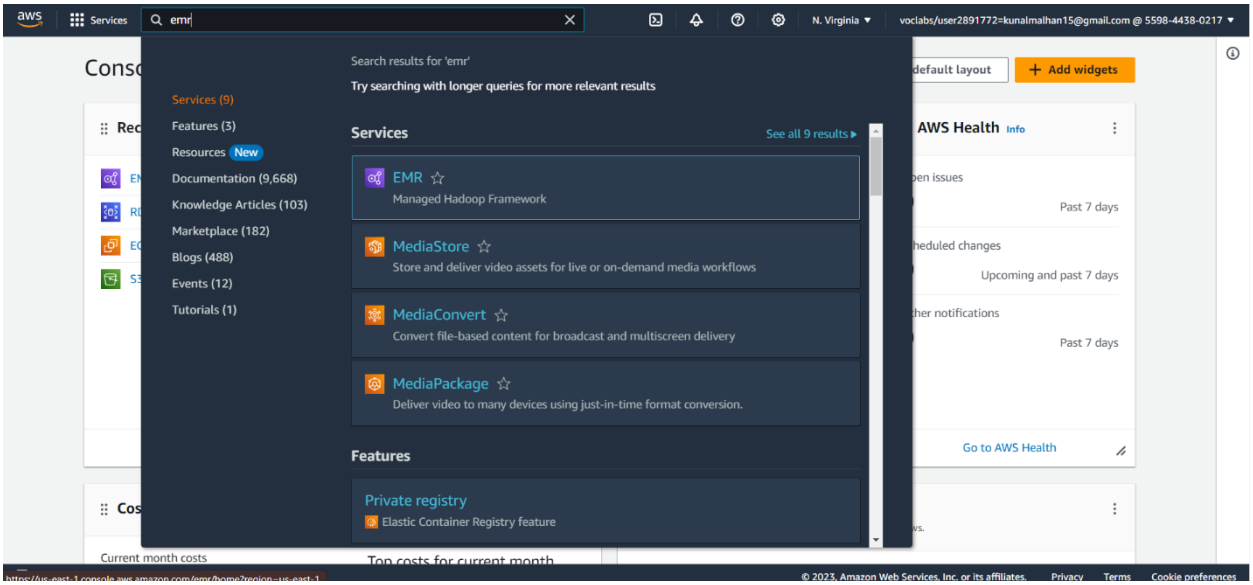


Sum of lines (excluding header row) is same as the records added in database.

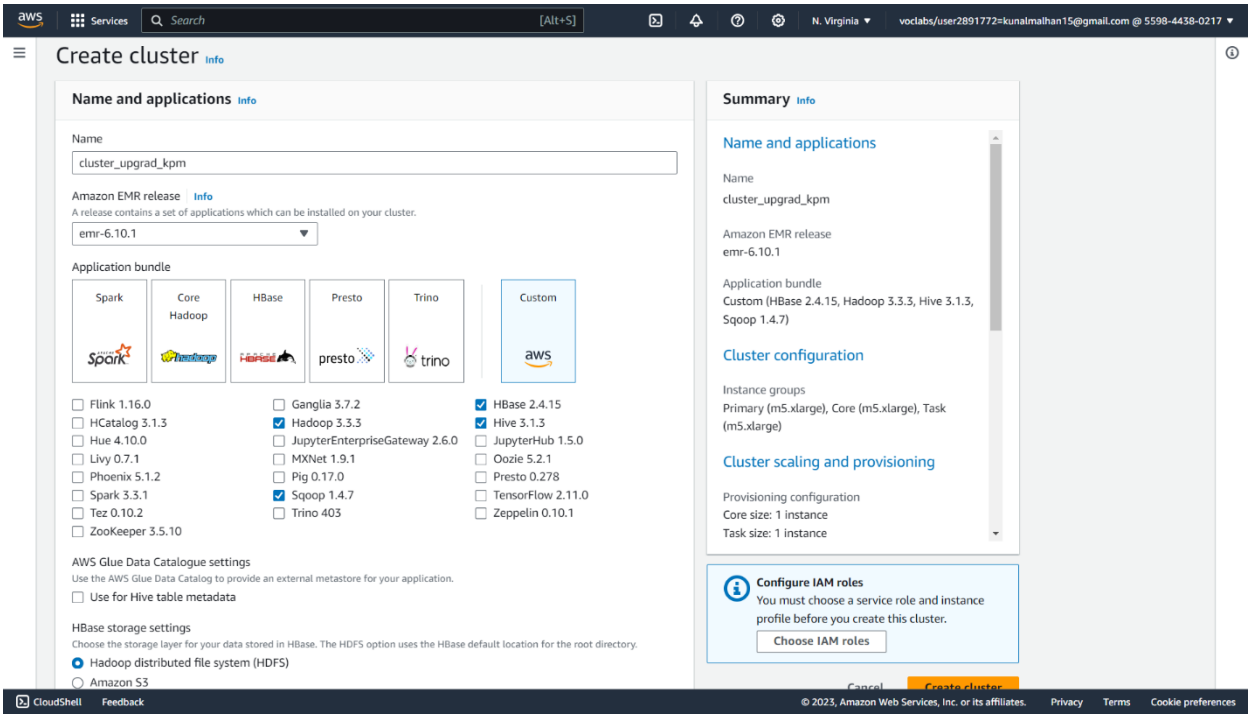
********** END OF TASK-1 **********

Annexure – I: Creating the EMR cluster

1. Search “EMR” or select “EMR” from recent visited on Console Home of AWS.

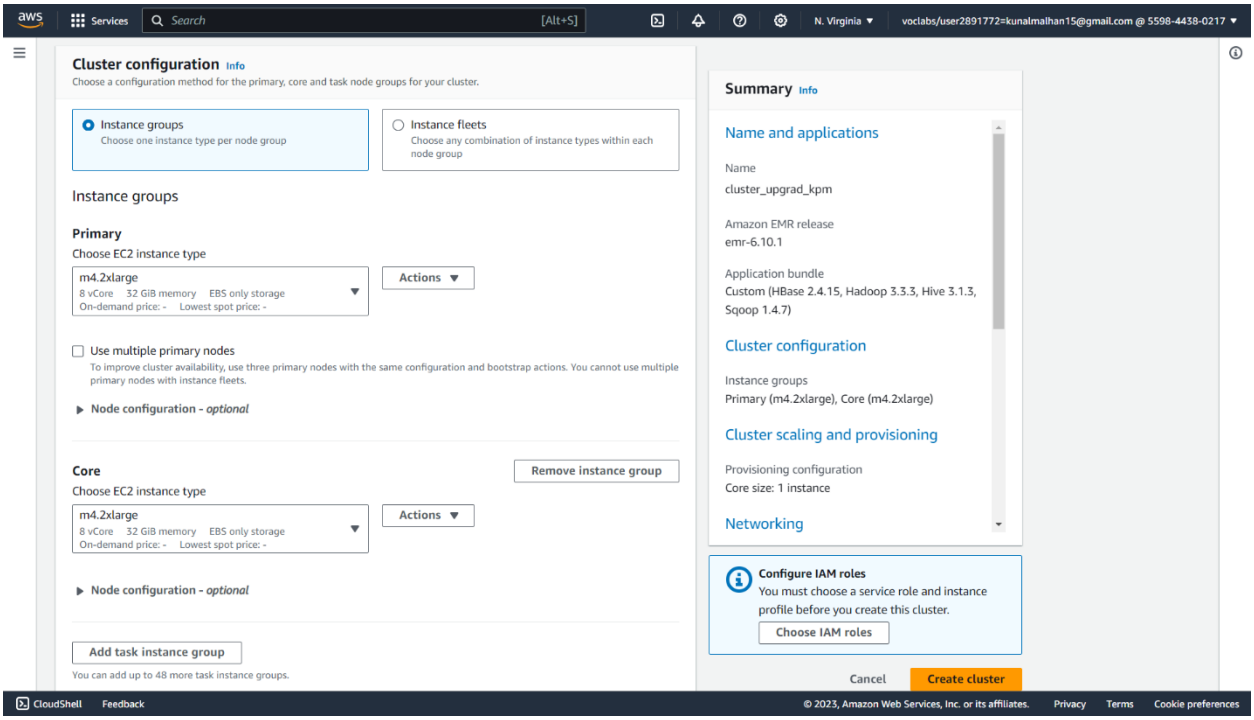


2. Creating “EMR” cluster with following configurations:
   a. Name : cluster_upgrad_kpm
   b. Amazon EMR release : emr-6.10.1
   c. Application bundle : Custom <Hadoop, HBase, Sqoop, Hive>

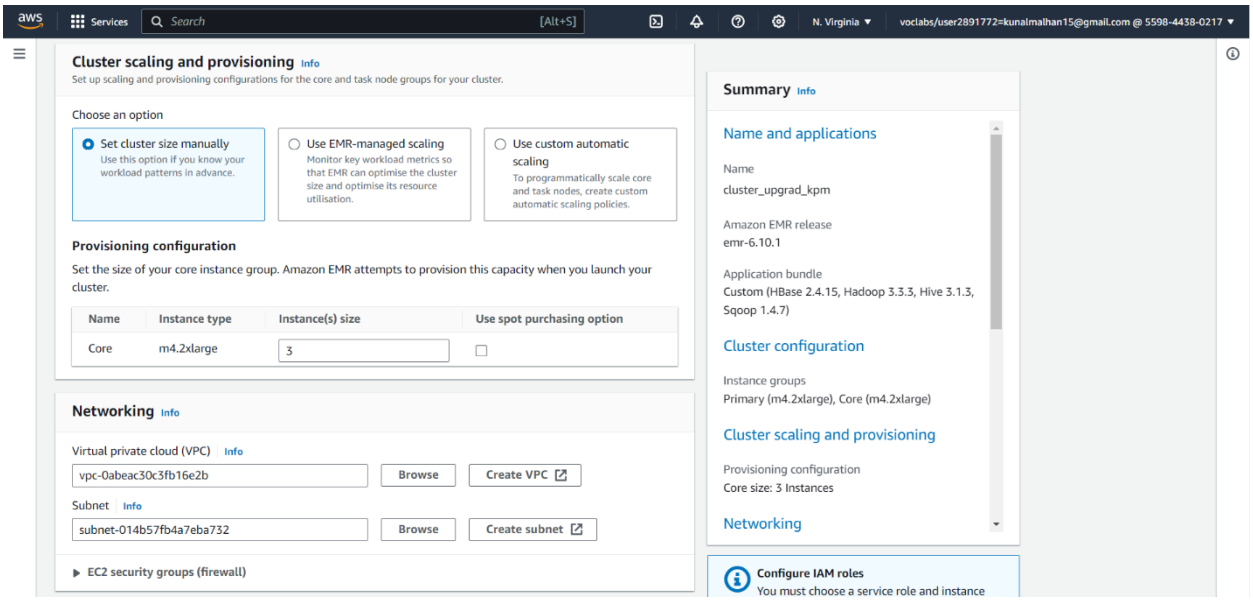


   d. Cluster Configuration:

i. Primary EC2 instance – m4.xlarge with EBS size of 100 GB
ii. Core Nodes – m4.xlarge with EBS size of 100 GB
iii. Task Nodes – None (Removed Task Nodes)



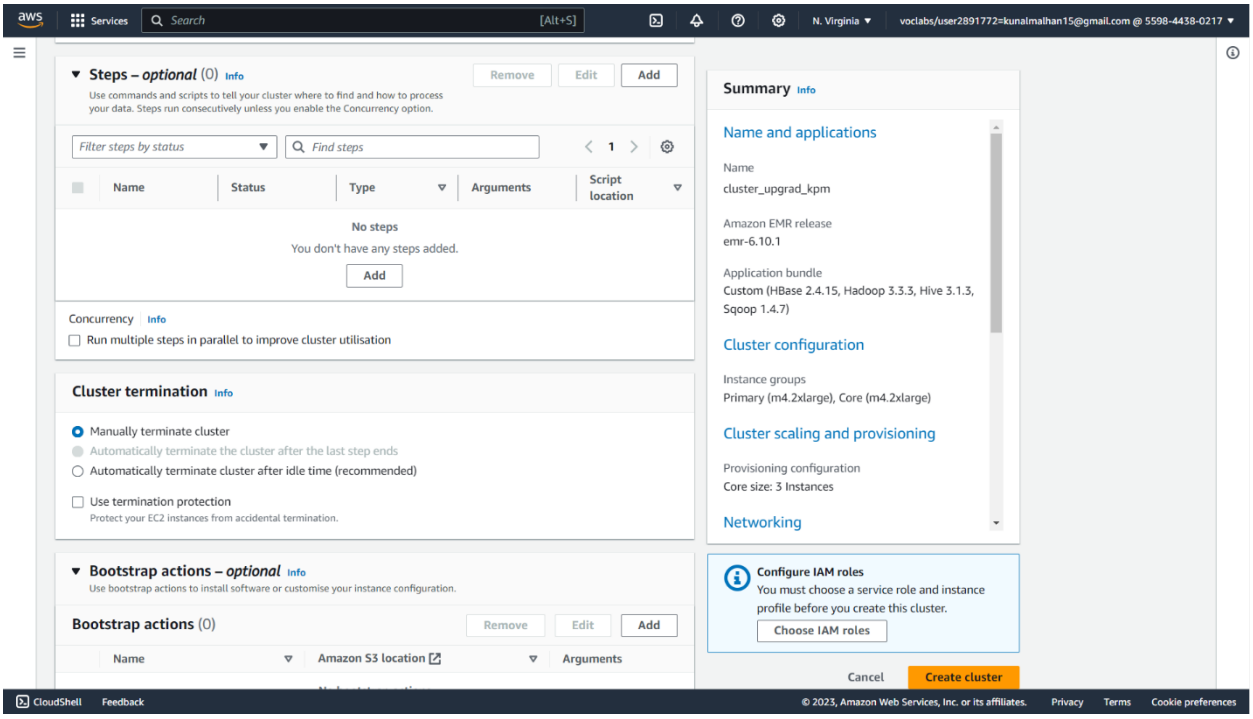
e. Cluster scaling and provisioning:
   i. Option : Set cluster size manually
   ii. Provisioning : 3 Instance(s) size
f. Networking : Default <No Change>



g. Steps : Default <No Change>

h. Cluster Termination : Manually terminate cluster (to avoid timeout problem)

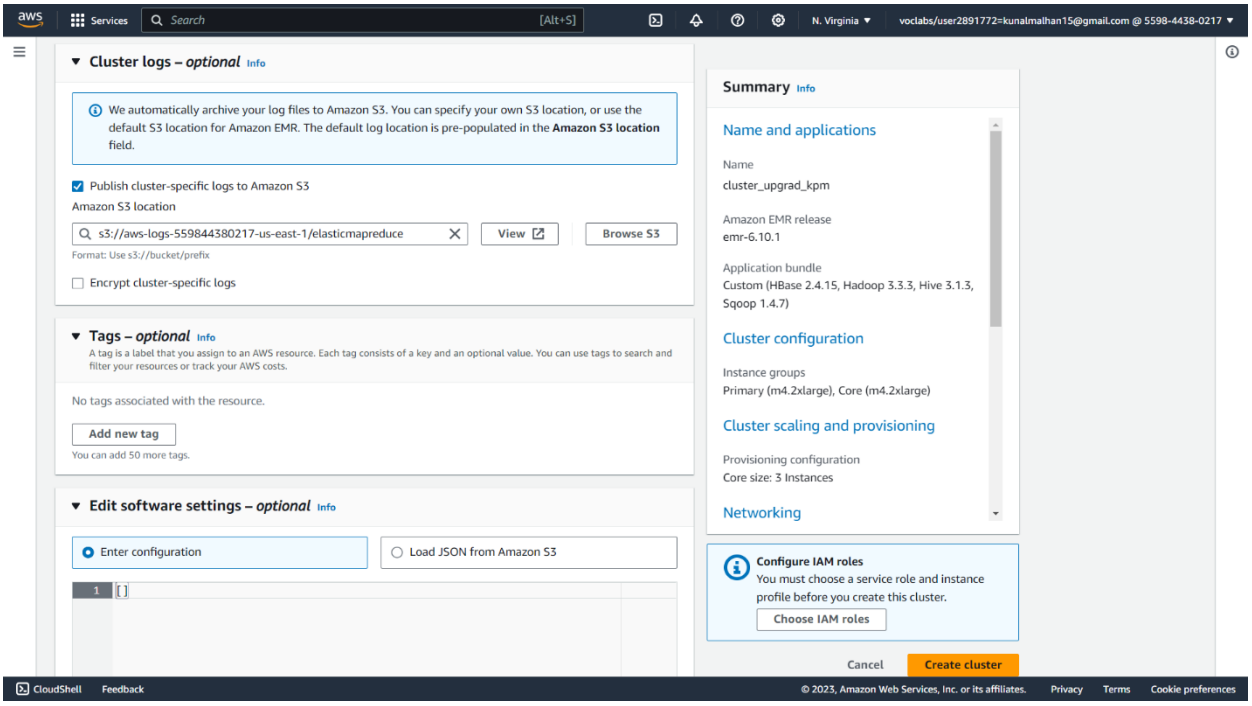
i. Bootstrap actions : Default <No Change>



j. Cluster logs : Default <No Change>

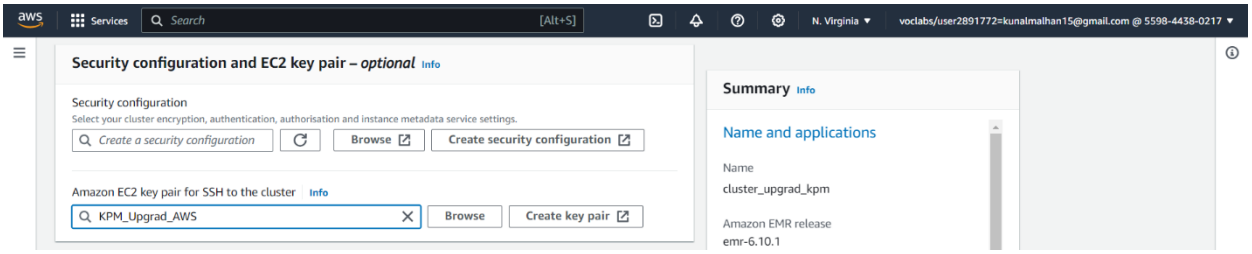
k. Tags : Default <No Change>

l. Edit software settings : Default <No Change>



m. Security configuration and EC2 key pair

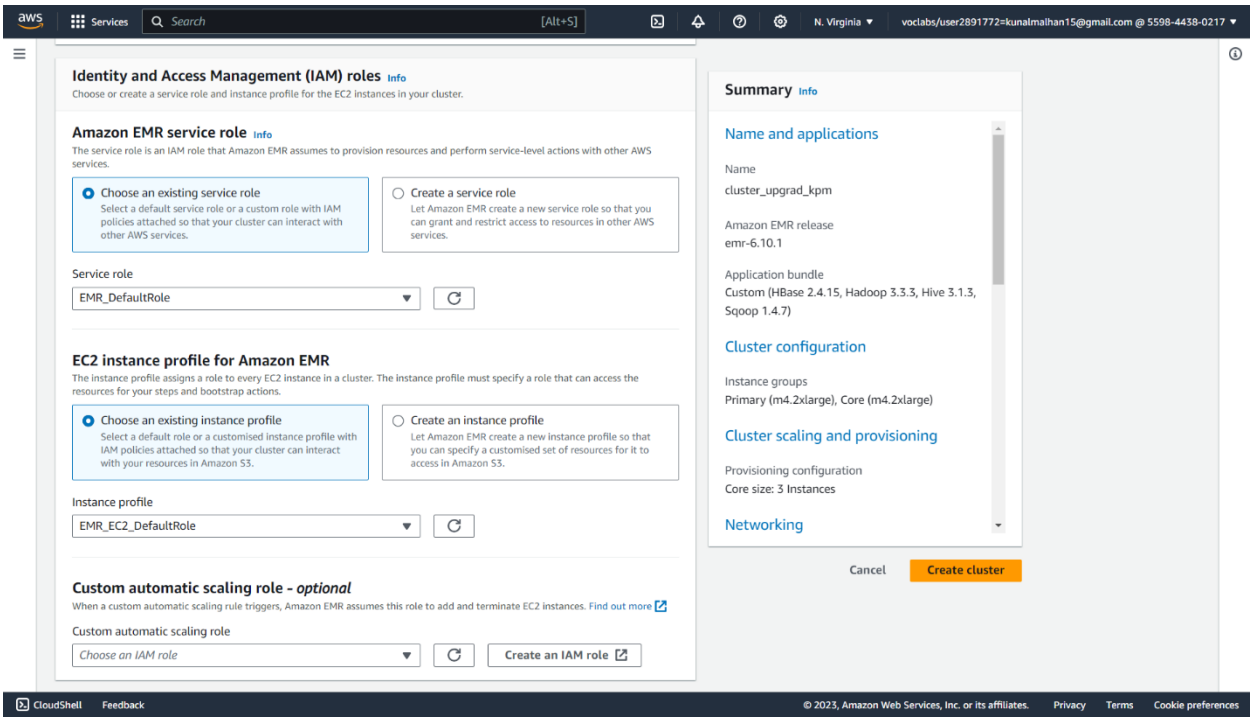
i. Amazon EC2 key pair for SSH : KPM_Upgard_AWS (Select your own key)



n. Identity and Access Management roles (IAM)

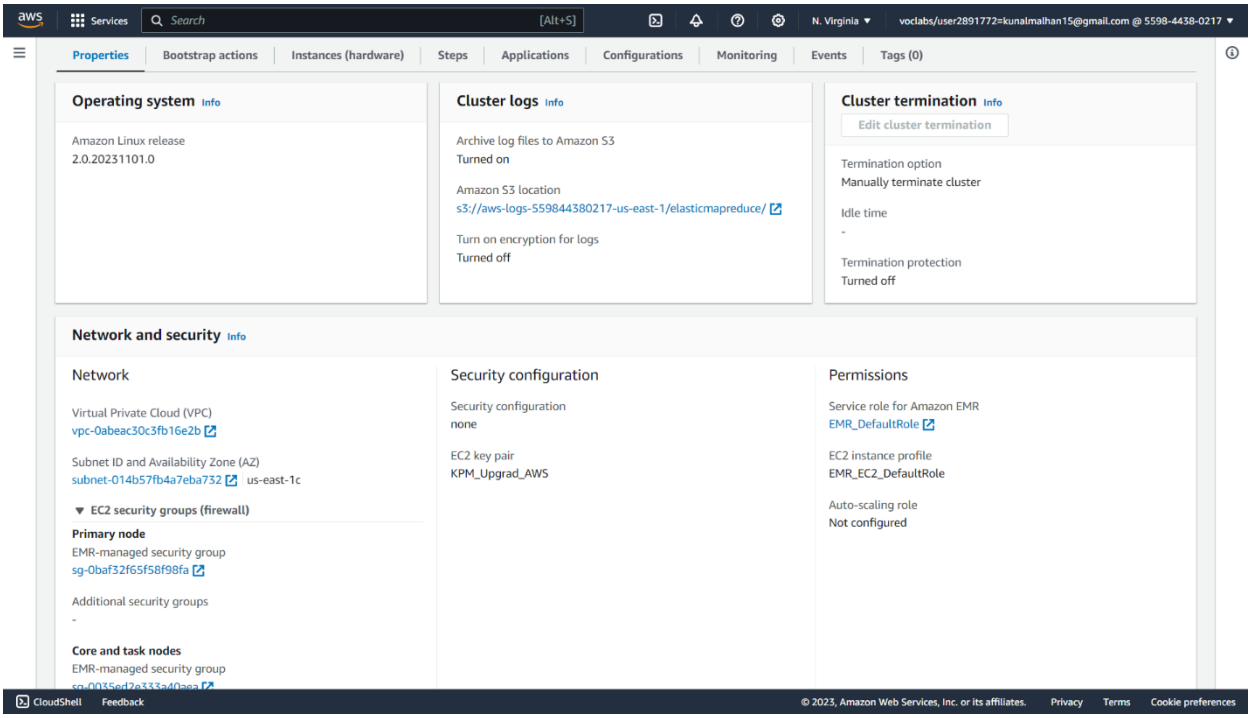
i. Amazon EMR service Role : EMR_Default_Role

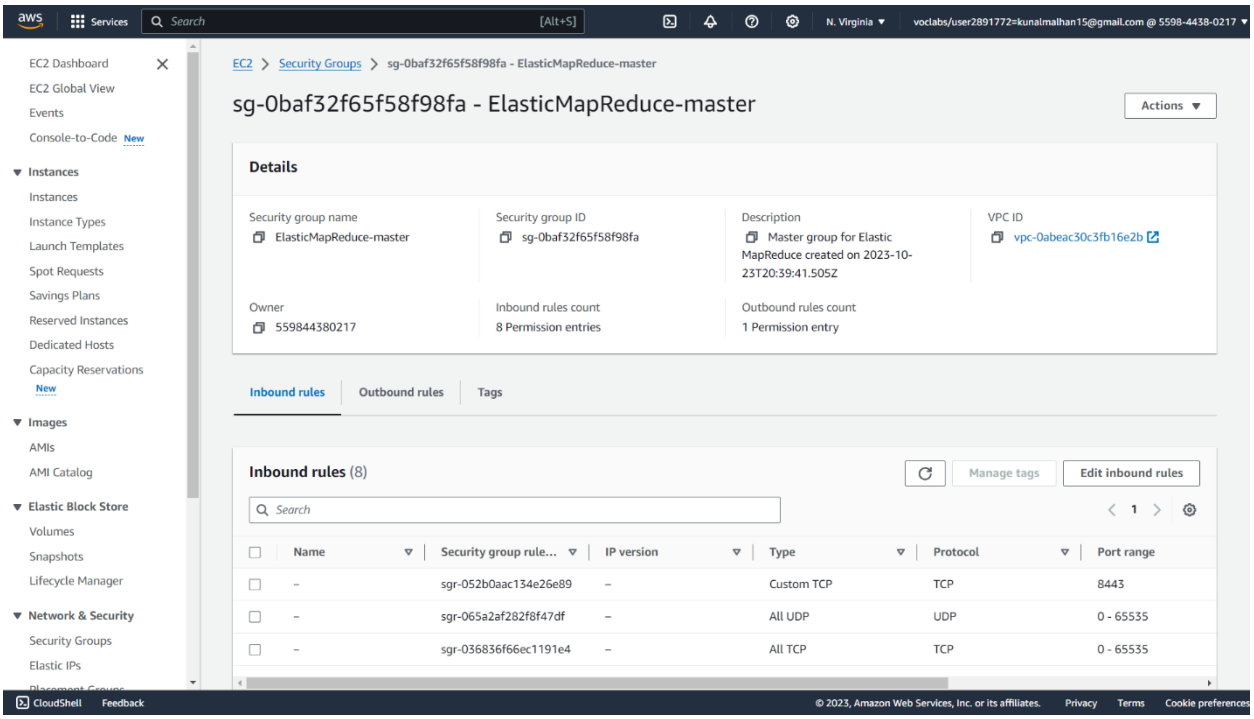
ii. EC2 instance profile for EMR : EMR_Default_Role



3. Allow access to EMR through Security Groups:

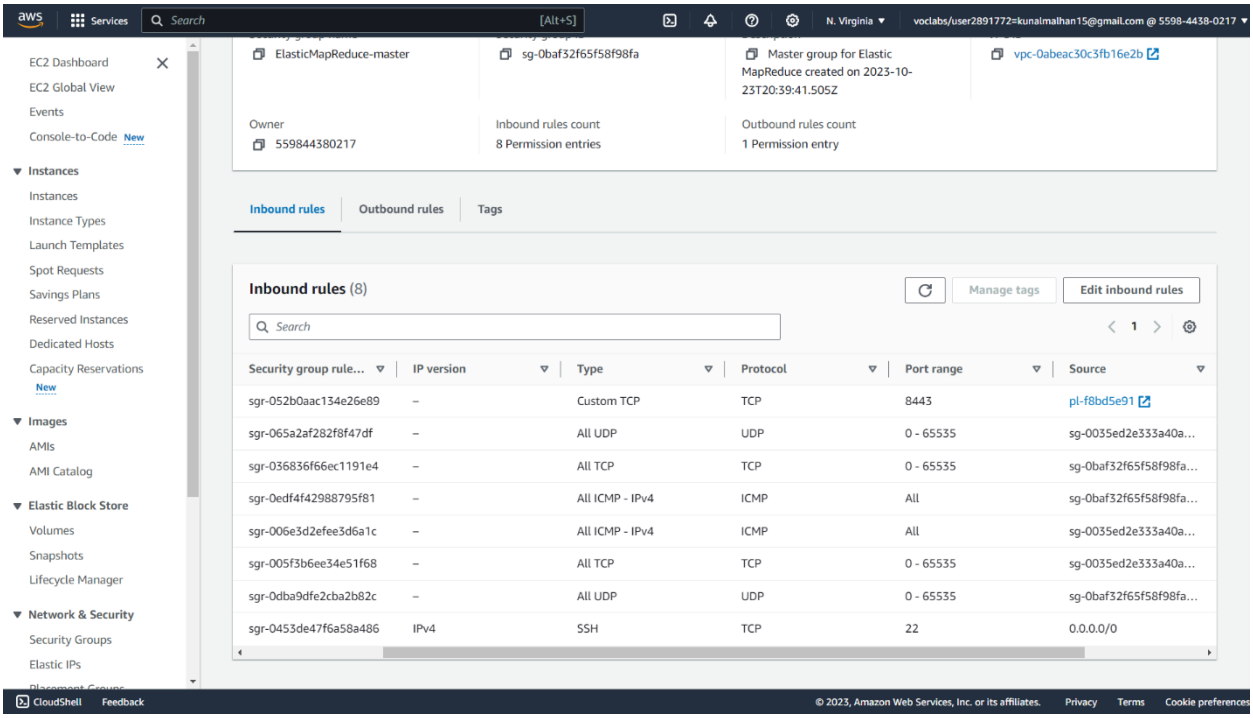
a. On EMR – Properties – Network and Security, click on EC2 security groups for Primary Node.





b. Enter and/or ensure Inbound rule with following configuration exists:
   i. IP version : IPv4
   ii. Type : SSH
   iii. Protocol : TCP
   iv. Port range : 22
   v. Source : 0.0.0.0/0 (this will allow all source to connect through

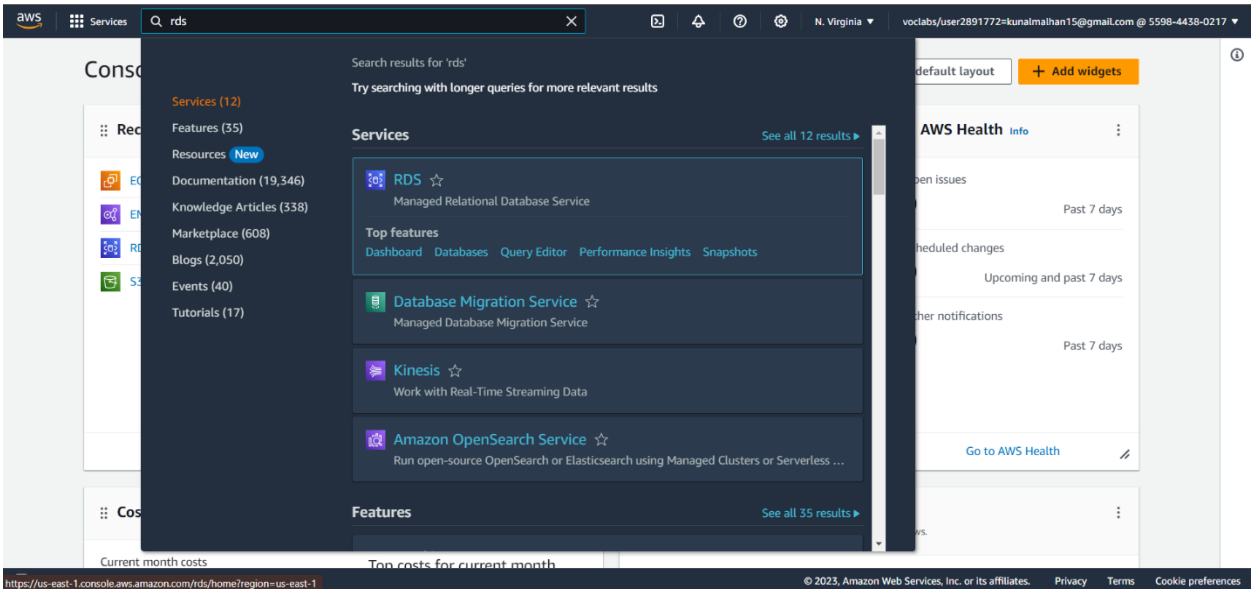
SSH. One can choose to enter own IP if its fixed IP.)



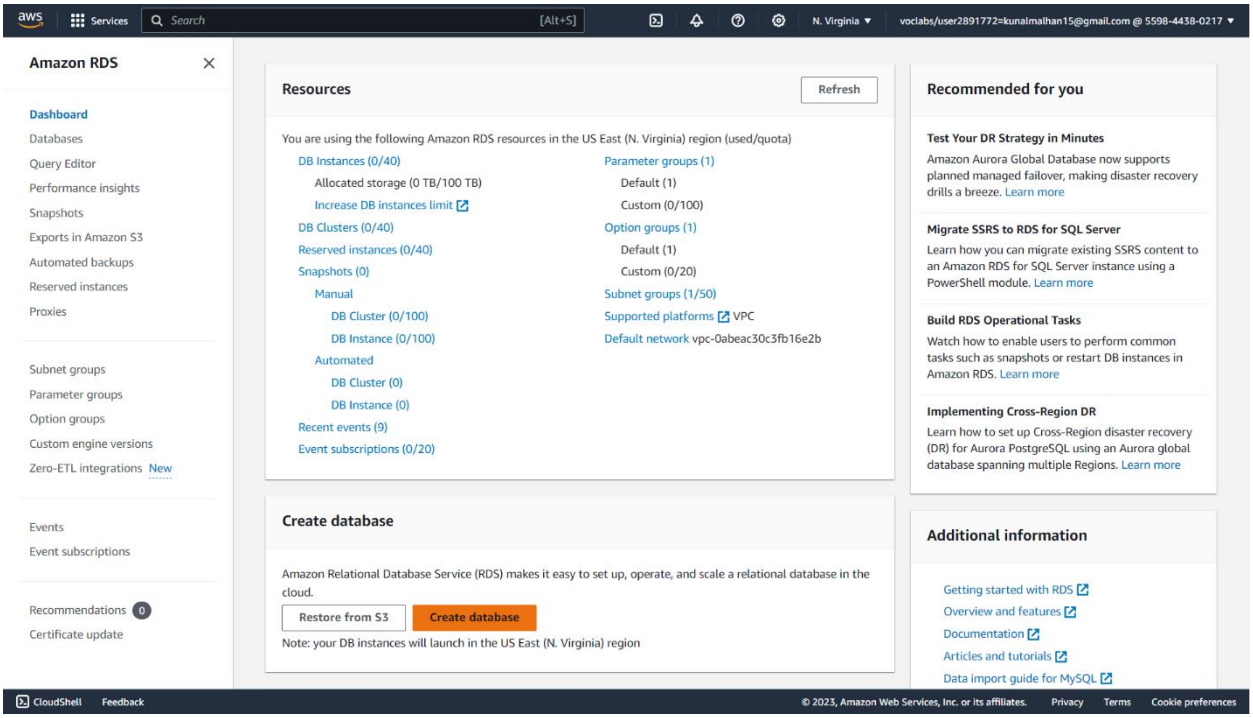
Successful Installation:

Annexure – II: Creating the RDS instance on AWS

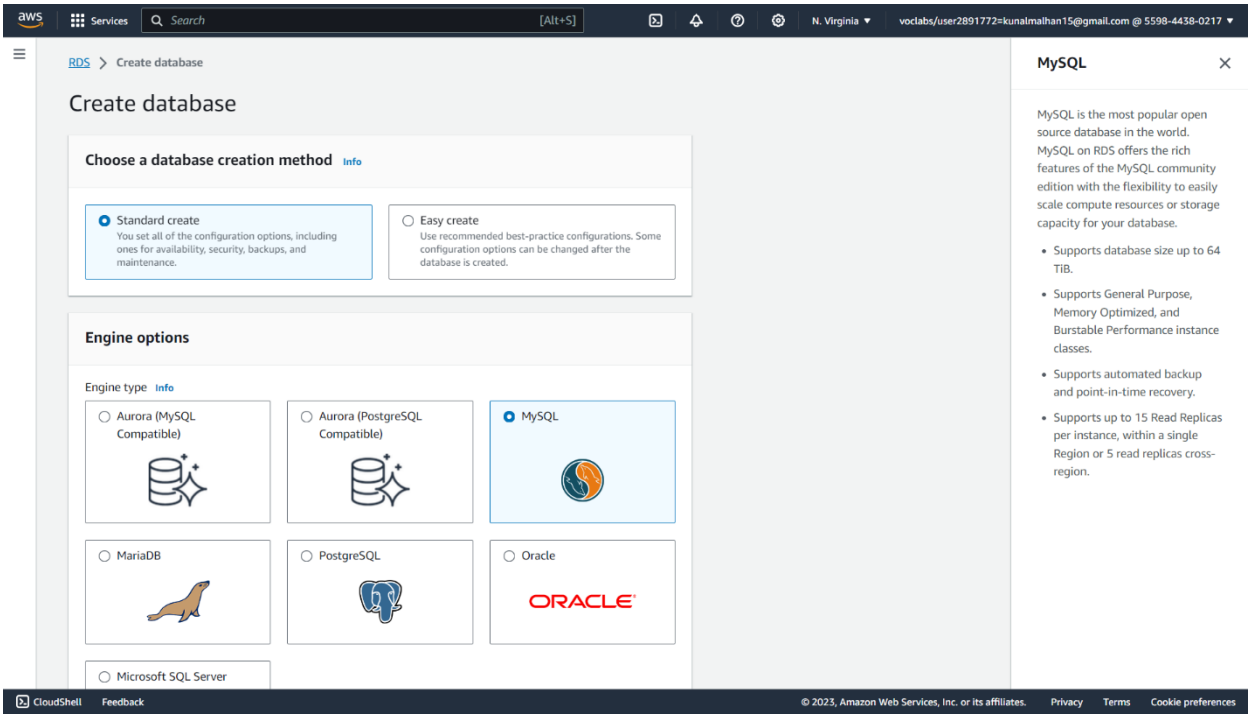
1. Search “RDS” or select “RDS” from recent visited on Console Home of AWS.



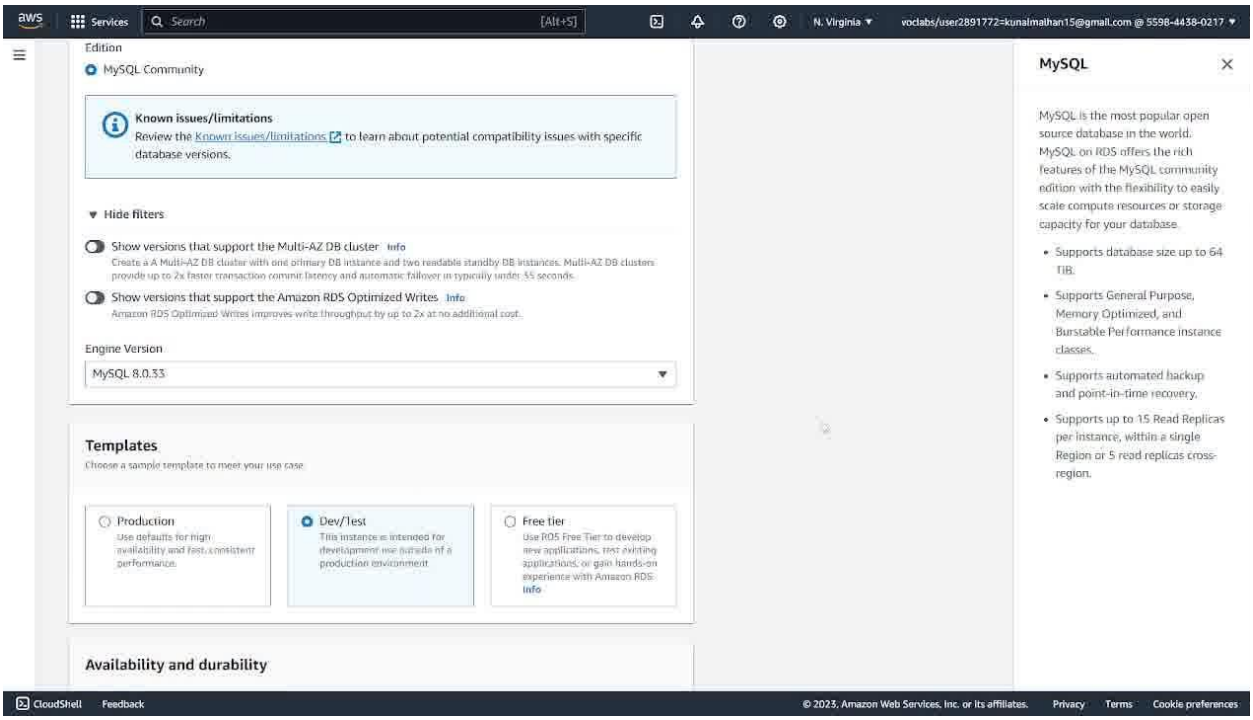
2. Click on Create Database.



3. Creating “RDS” instance with following configurations:
    a. Choose a database creation method : Standard Create <No Change>
    b. Engine Options : MySQL

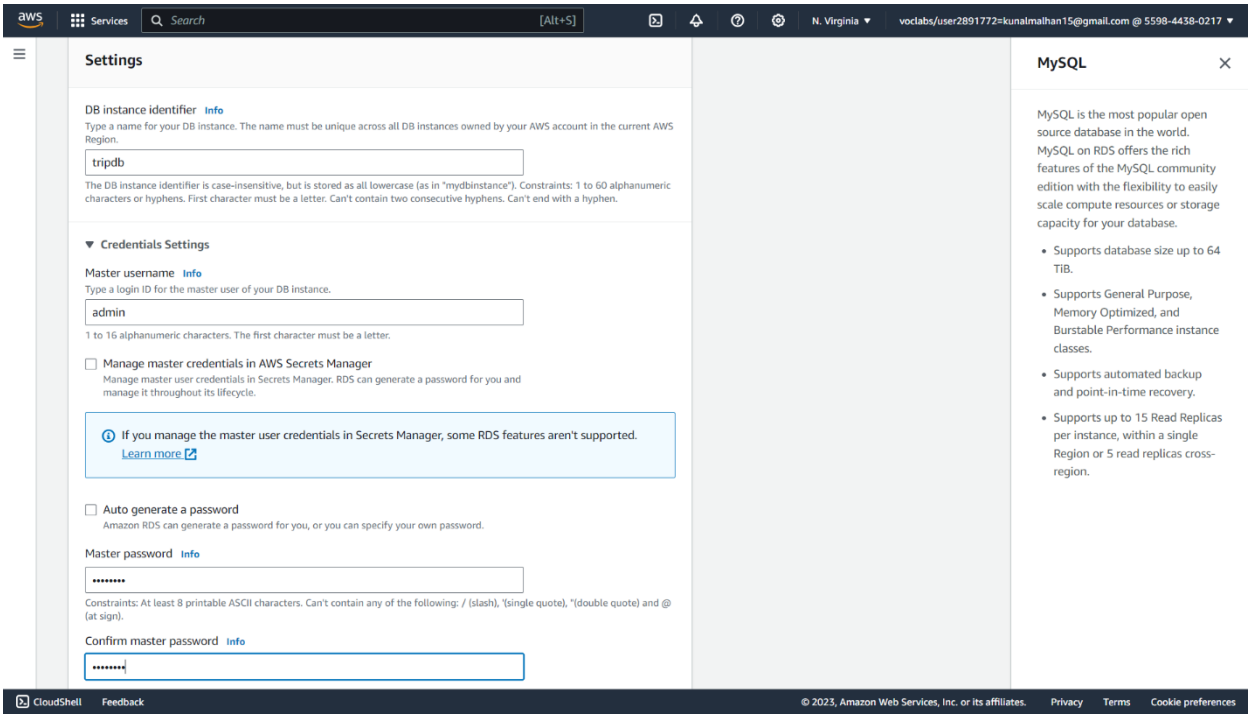


c. Edition : MySQL Community
d. Engine Version : 8.0.33 <No Change>
e. Templates : Dev/Test

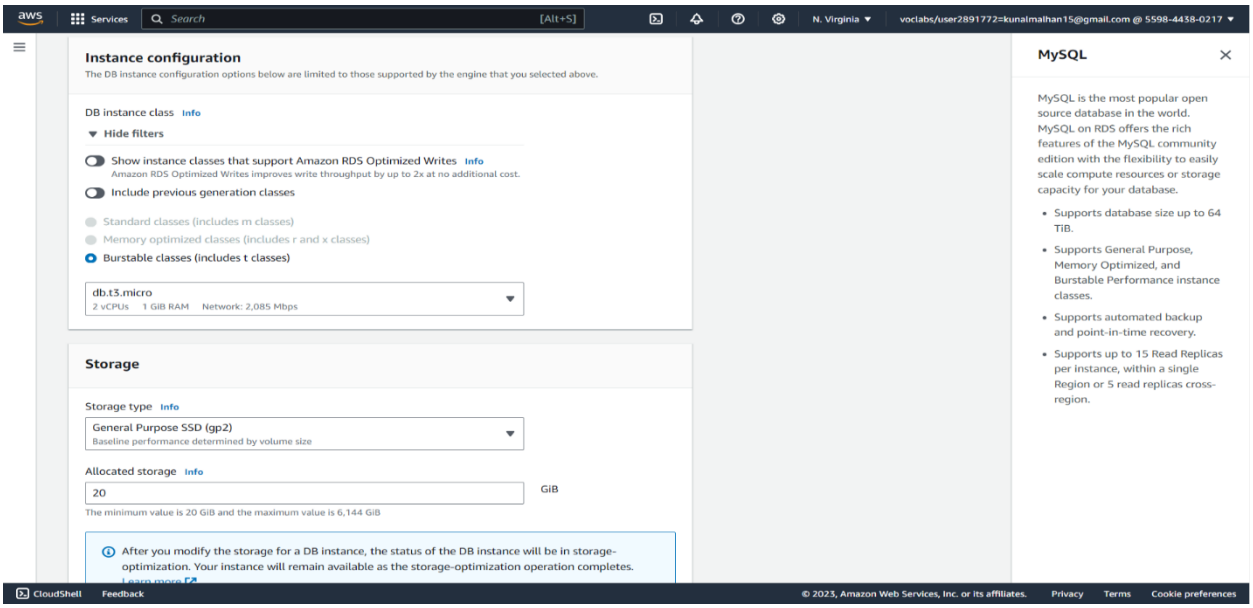


f. Availability and durability : Single DB instance
g. Settings

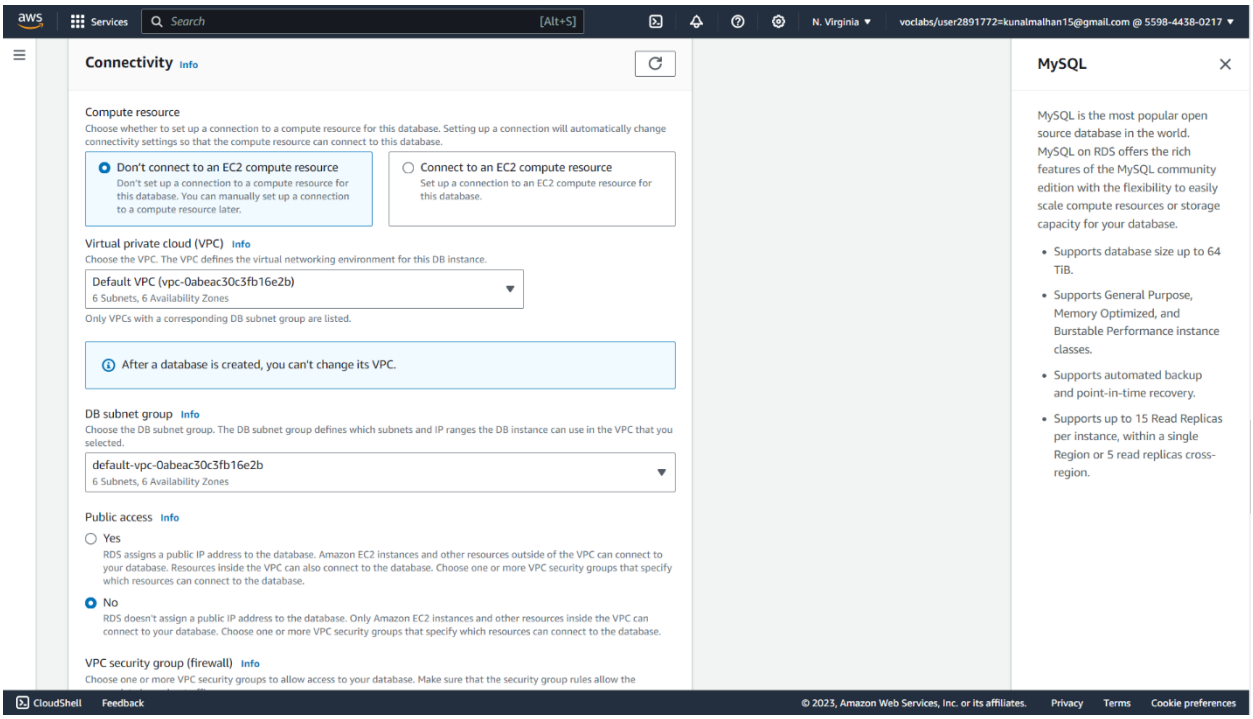
i. DB instance identifier : tripdb
ii. Master Username : admin
iii. Master password : user1234



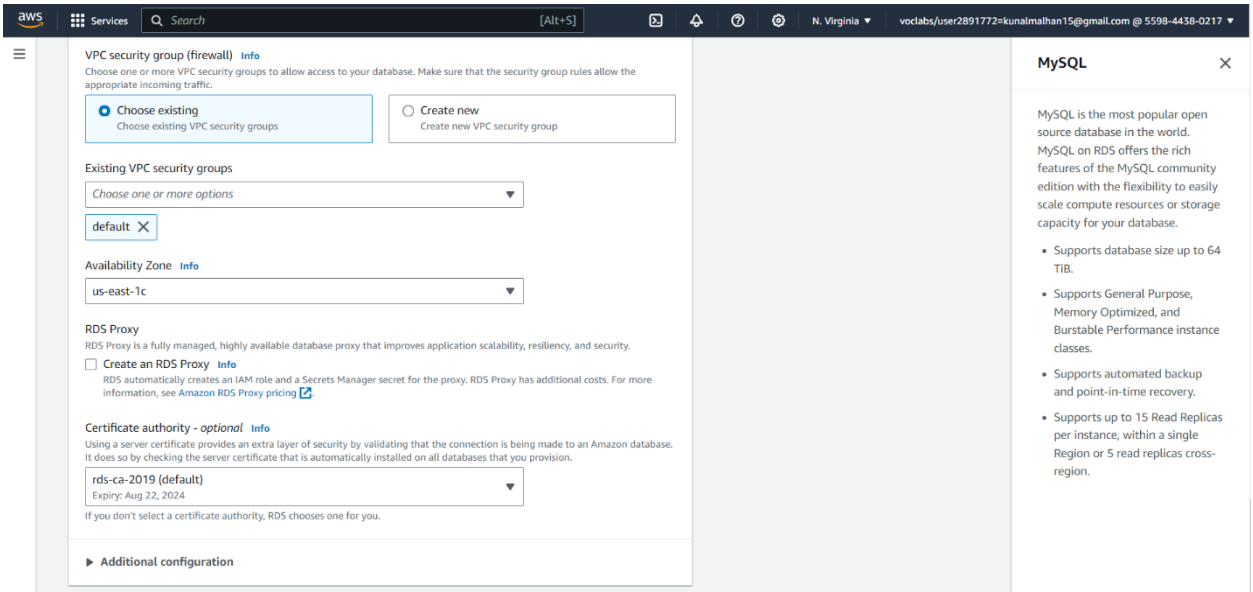
h. DB Instance Class : Burstable Classes (includes t classes)
i. Instance configuration : db.t3.medium
j. Storage
   i. Storage Type : General Purpose SSD (gp2) <No Change>
   ii. Allocated storage : 20 GB <No Change>



k. Connectivity <will perform after creation of instance>
   i. Compute resource : Default <No Change>
   ii. Virtual private cloud (VPC) : Default <No Change>
   iii. DB subnet group : Default <No Change>
   iv. Public access : No <No Change>

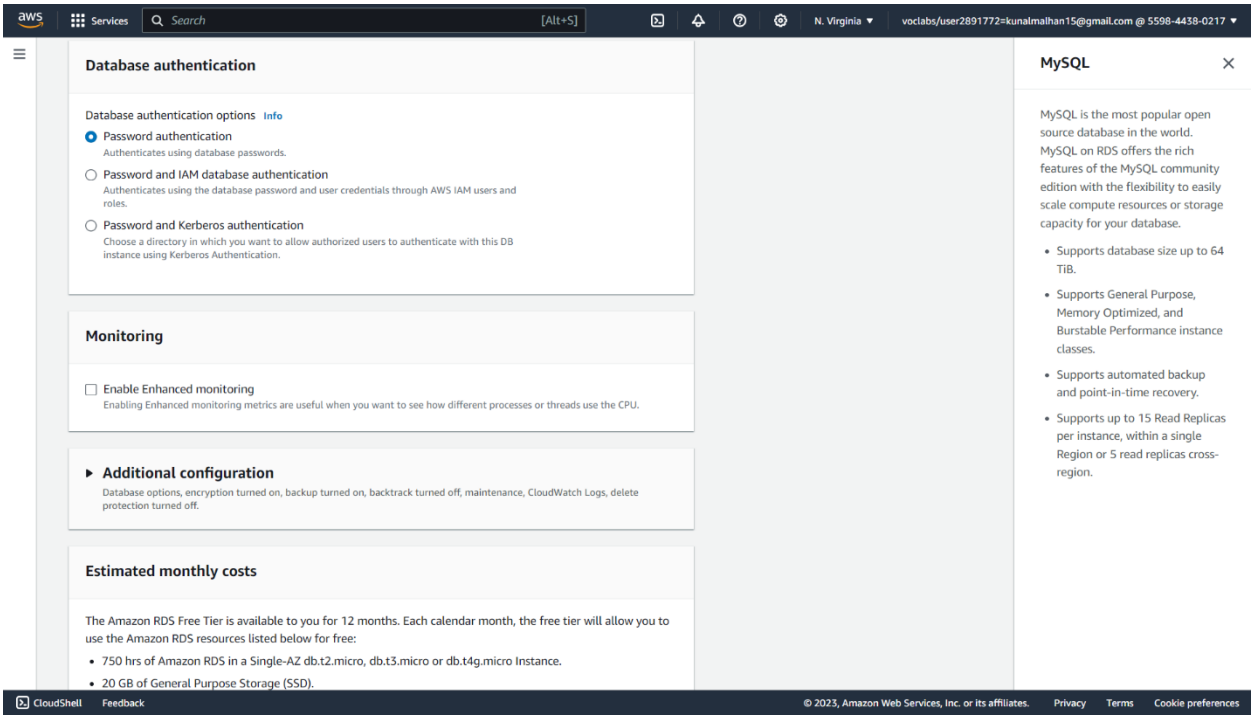


v. VPC security group : Default <No Change>
vi. Availability Zone : <Same as of EMR cluster to save inter AZ NW cost>
vii. RDS proxy : Default <No Change>



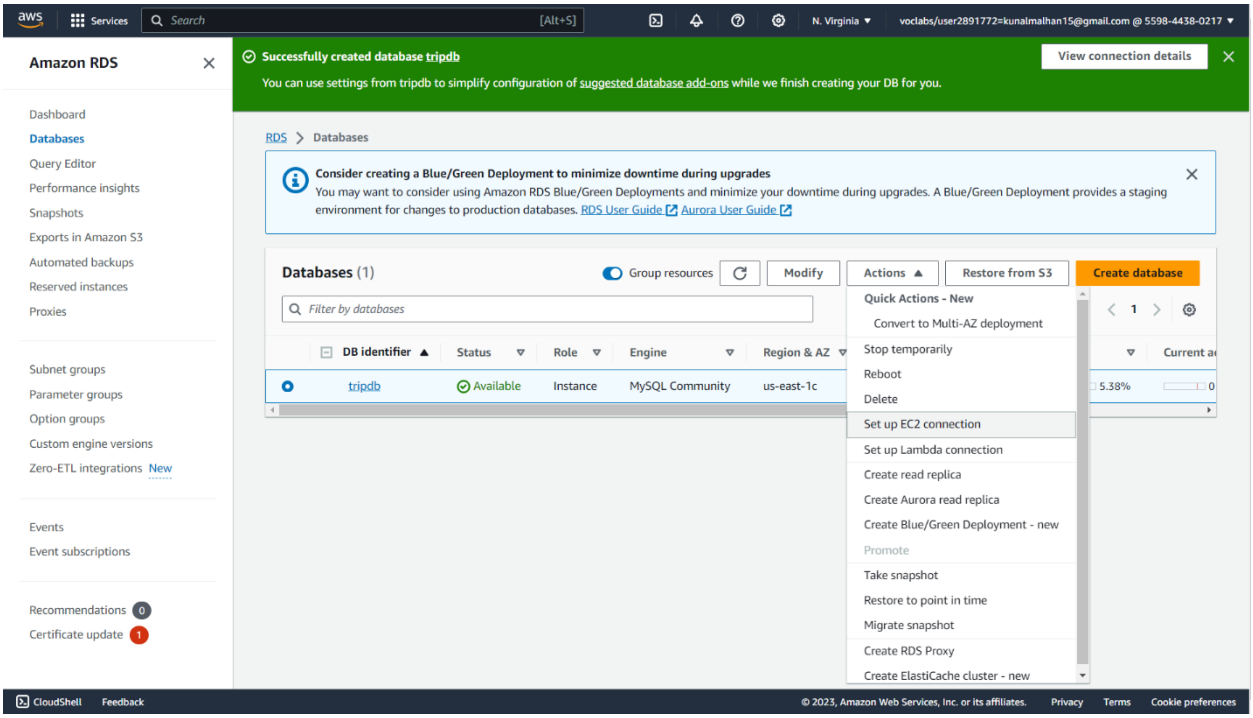
l. Database authentication : Password authentication <No change>
m. Monitoring : Default <No Change>

n. Additional Configuration : Default <No Change>

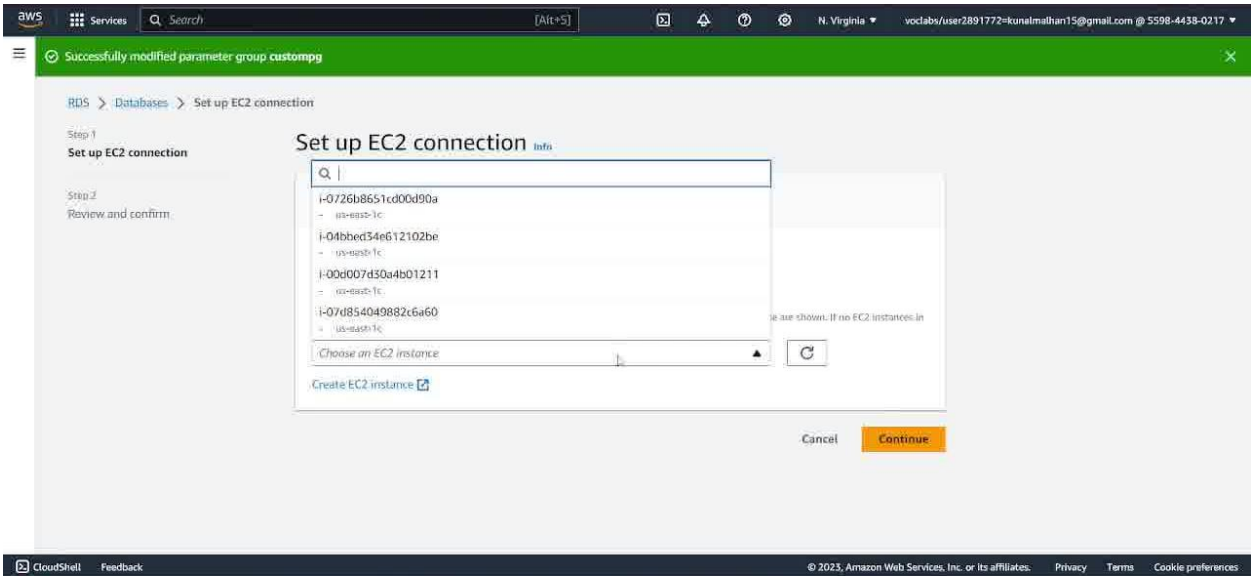


Annexure – III: Connecting the RDS instance to EMR cluster

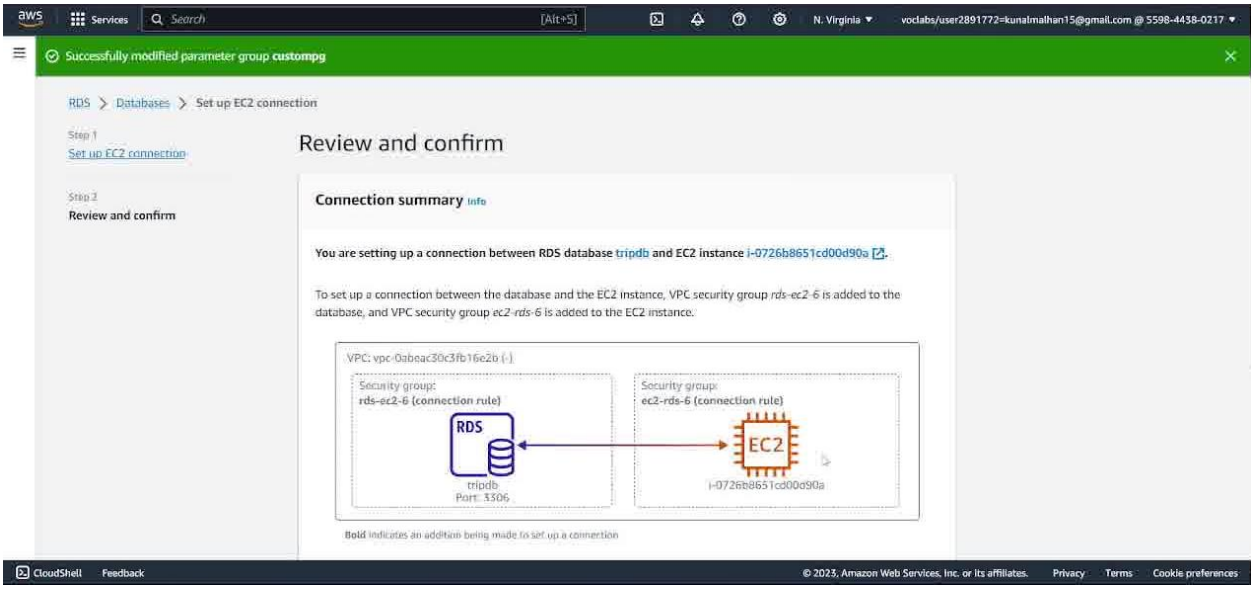
1. Go to RDS – Databases on AWS console, and select the database with DB identifier as “tripdb”.
2. Click on Actions – Setup EC2 connection

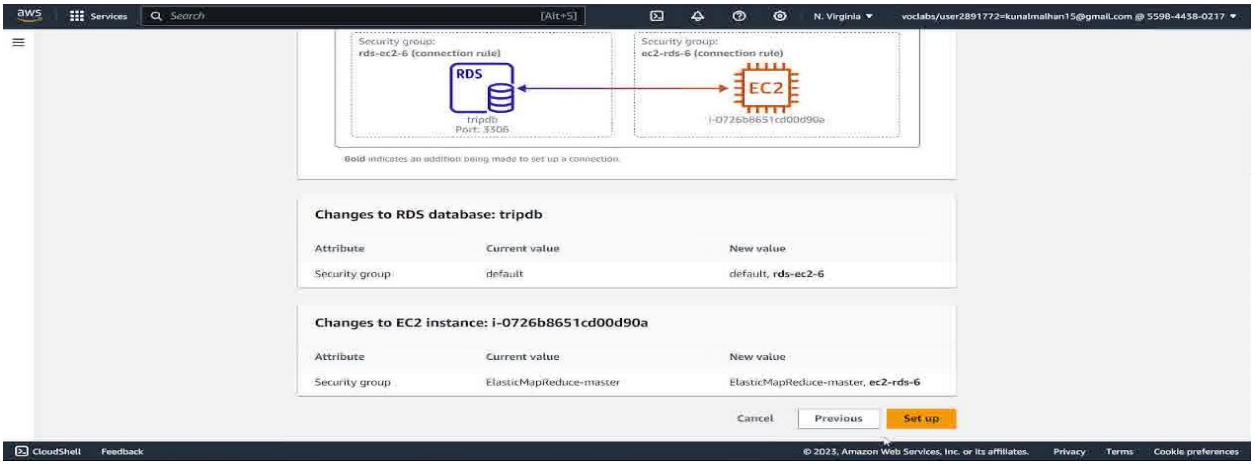


3. Select EC2 instance one by one (Primary and all core nodes):

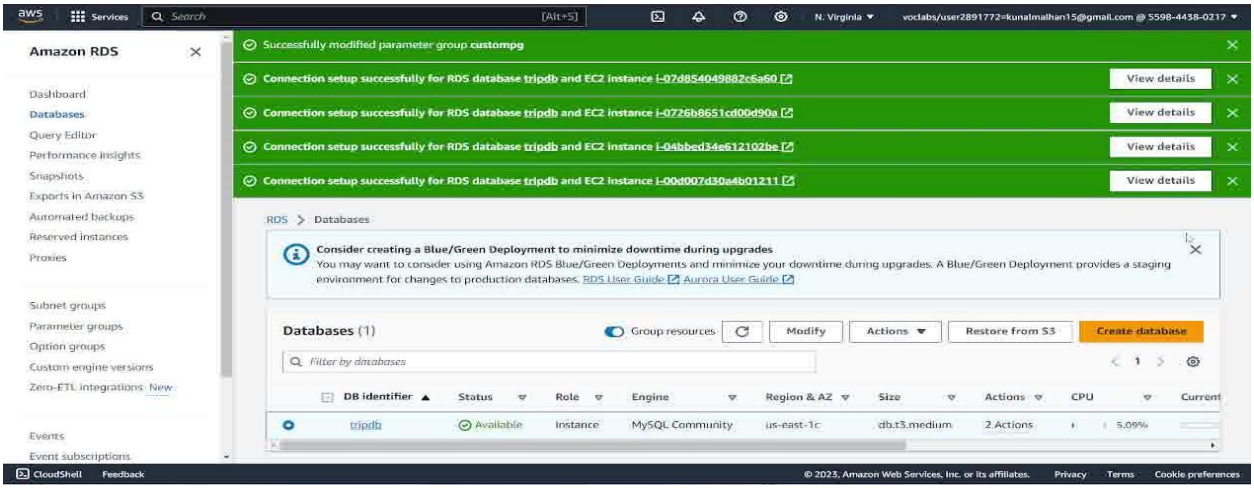


4. Click on Setup Review and confirm.





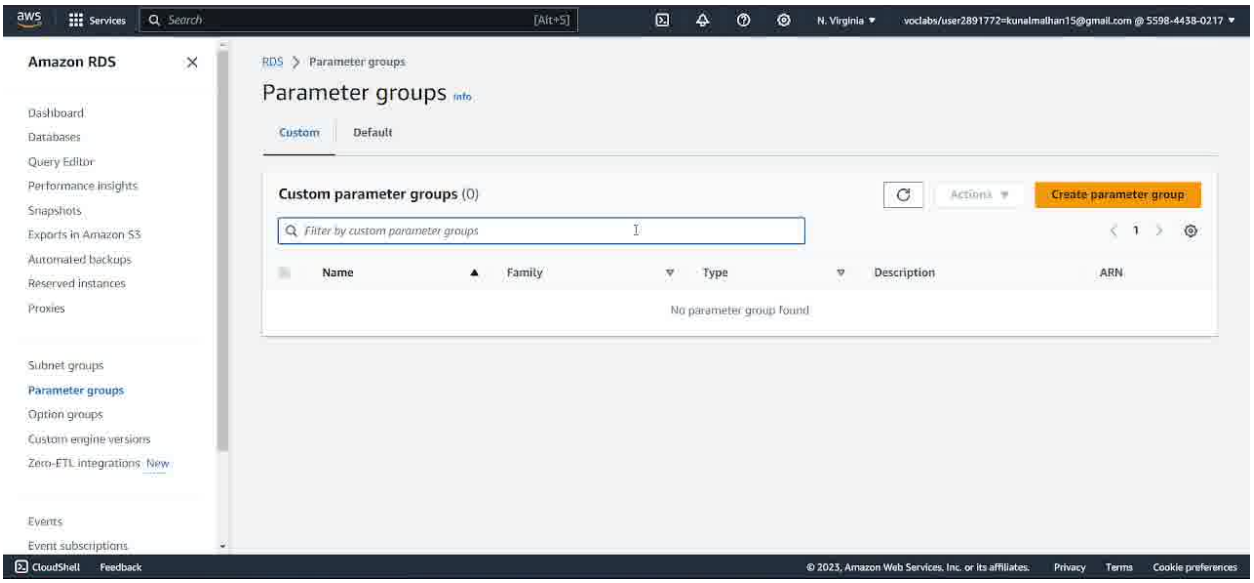
5. Database status will changed to “Modifying”, and then “Available” once completed.



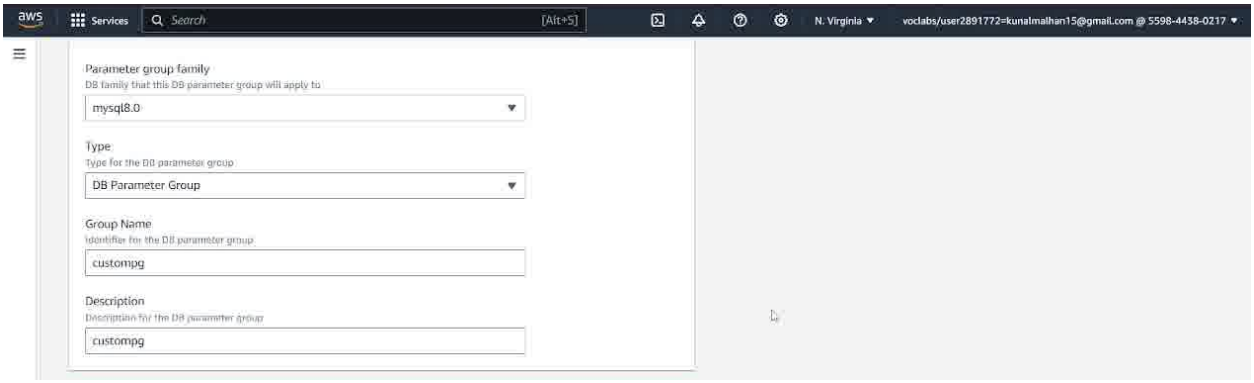
Edit the MySQL Global Variable through Parameter Group

This is required as we are testing using DB instance with only 2 cores and 4 GB memory due to restriction on academy account. To avoid read timeout in parallel fetch, 'net_read_timeout' is set to higher value. This step also provide learning curve for setting any global database variable in similar manner.

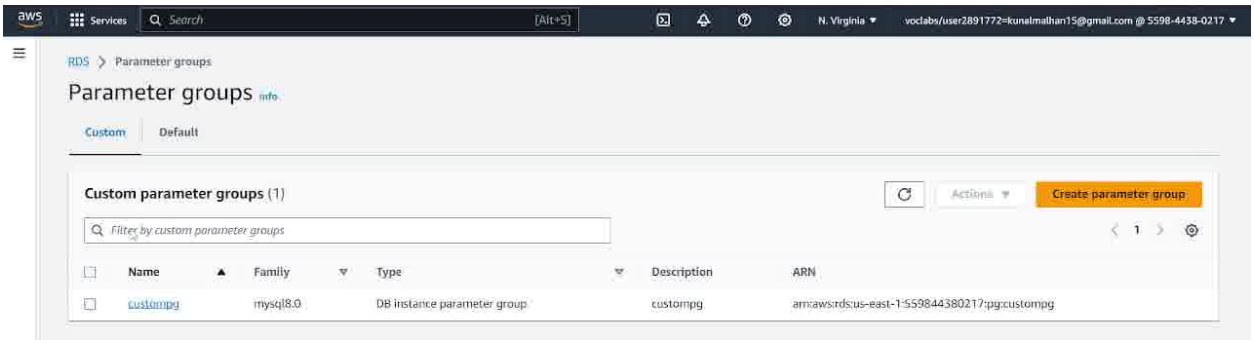
1. Go To RDS > Parameter groups



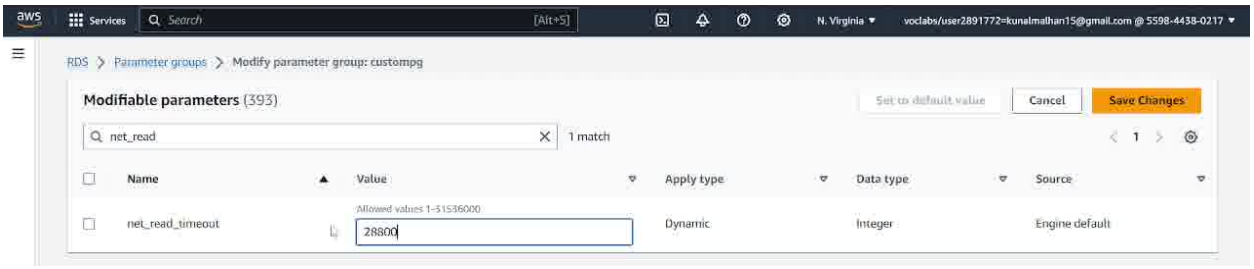
2. Click on 'Create parameter group' and provide required basic details:



3. Select new custom parameter group, and click on Edit under Actions:



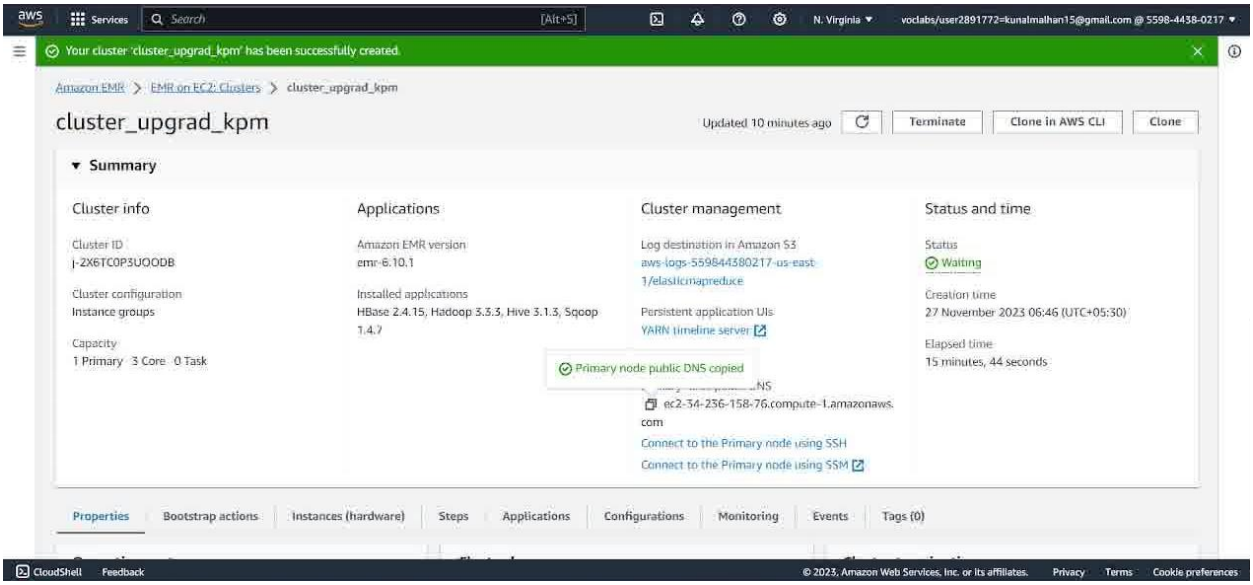
4. Filter and enter modified value of “Global Variable” ‘net_read_timeout’:



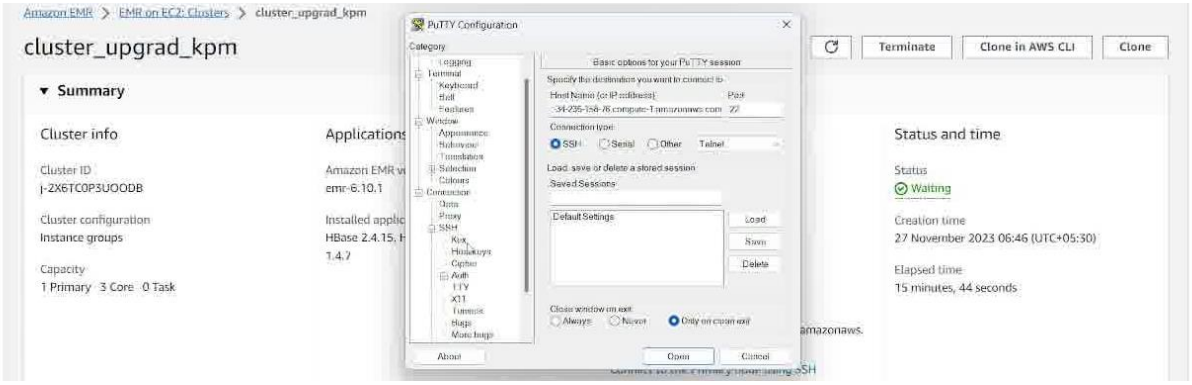
5. Click on Save Changes to complete.

Annexure IV: Login to EMR cluster's master node through putty

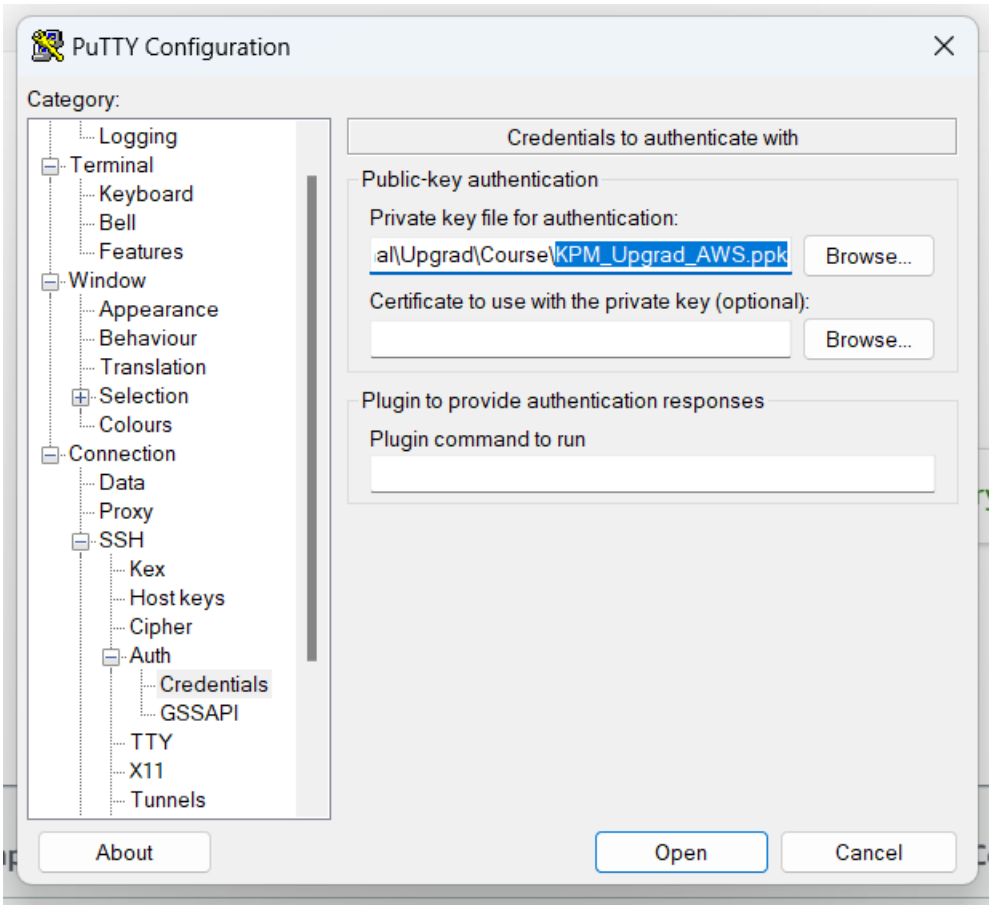
1. Copy the 'Primary node public DNS' from EMR cluster's summary.



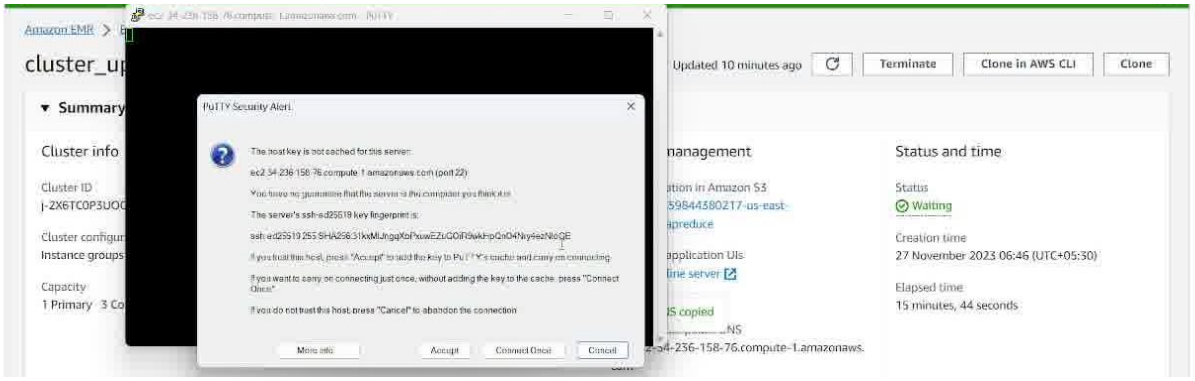
2. Use the Host Name as same public DNS of Primary Node:



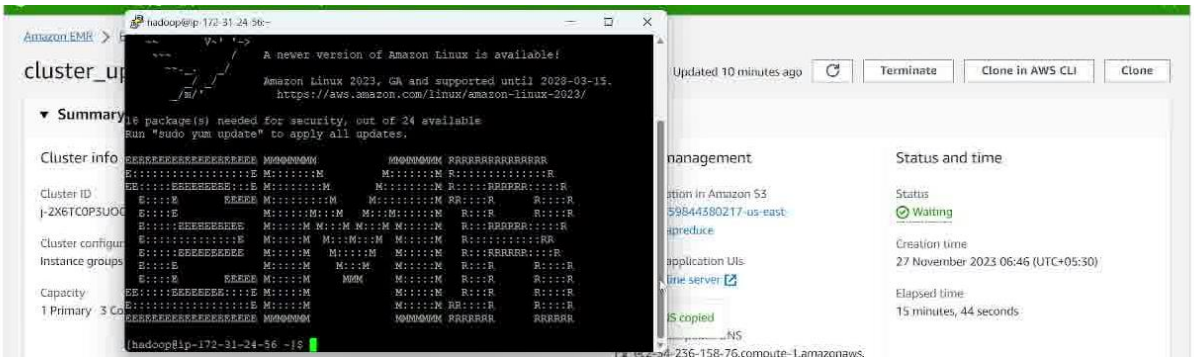
3. Use the ppk file used for creating the EMR cluster:



4. Click on Accept:



5. Default username for EMR cluster is ‘hadoop’. Use to connect successfully:


Amazon EMR
cluster_up
Updated 10 minutes ago
Terminate
Clone in AWS CLI
Clone
Summary
Cluster info
Cluster ID
Capacity
1 Primary
Status and time
Status
Waiting
Creation time
27 November 2023 06:46 (UTC+05:30)
Elapsed time
15 minutes, 44 seconds
hadoop@ip-172-31-24-56:~
A newer version of Amazon Linux is available!
Amazon Linux 2023, GA and supported until 2028-03-15.
https://aws.amazon.com/linux/amazon-linux-2023/
16 package(s) needed for security, out of 24 available
Run "sudo yum update" to apply all updates.
[hadoop@ip-172-31-24-56 ~]$