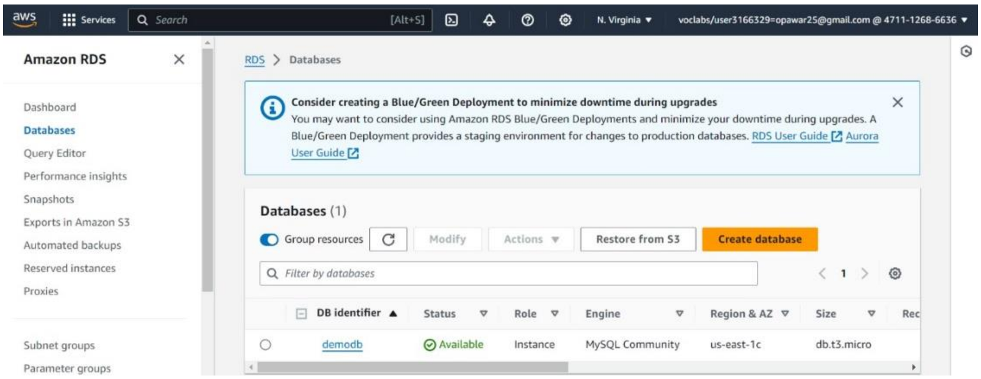
TASK 1

Create an RDS instance in your AWS account and upload the data to the RDS instance. Since the dataset is huge, you need to upload the data from only two files

Step 1. Creating an RDS instance

- Search for RDS on Amazon Web Server.
- Click on create database under database menu.
- Further choose “Standard create” from the database creation method.
- Next click on MySQL from Engine Options to run a MySQL engine.
- Click on Free tier under the Templates option to use the engine for free.
- Now, give a name to your database under DB cluster identifier.
- Next, give a public access to your database by clicking on yes under the Public Access option.
- Ensure to make the database password authentication under database authentication for security.
- At last, click on Create Database to create the RDS Database.
- The database will take 5-10 minutes to get active.



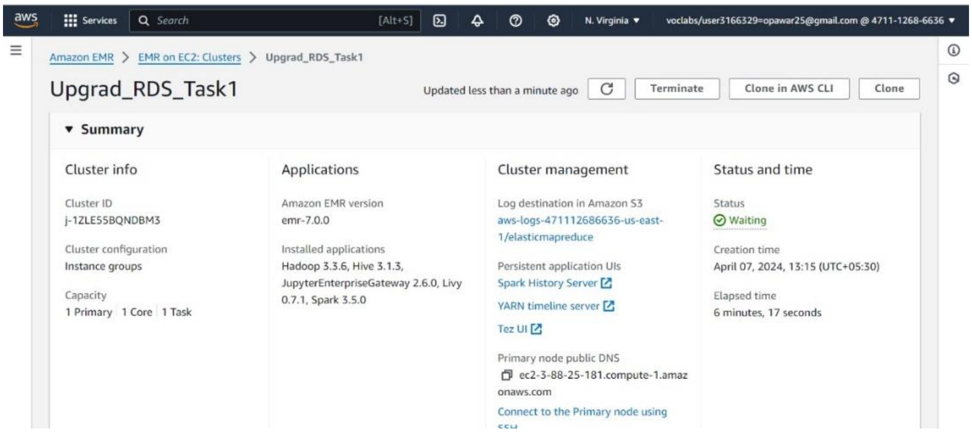
Consider creating a Blue/Green Deployment to minimize downtime during upgrades
You may want to consider using Amazon RDS Blue/Green Deployments and minimize your downtime during upgrades. A Blue/Green Deployment provides a staging environment for changes to production databases. RDS User Guide Aurora User Guide

Databases (1)

	DB identifier ▲	Status	Role	Engine	Region & AZ	Size	Rec
○	demodb	✓ Available	Instance	MySQL Community	us-east-1c	db.t3.micro	

Step 2. Creating an EMR Cluster

- Search for EMR on Amazon Web Server.
- Click on “create cluster” to create a EMR cluster
- Now, give name to your cluster under the name column.
- Next, tick the required applications under the application bundle for example Hadoop, HBase, Sqoop etc.
- Further choose an EC2 instance type for Primary node.
- Next enter the Size required (30 GiB) under EBS root volume.
- Now, select the EC2 key pair for SSH to the cluster by creating a new key pair.
- Now at last, select the service role and instance profile as default role
- Finally, click on Create Cluster to create an EMR Cluster
- The cluster will take 5-10 minutes to get active


aws
Services
Search [Alt+S]
N. Virginia
voclabs/user3166329=opawar25@gmail.com @ 4711-1268-6636
Amazon EMR > EMR on EC2: Clusters > Upgrad_RDS_Task1
Updated less than a minute ago
Terminate
Clone in AWS CLI
Clone

Upgrad_RDS_Task1

Summary

Cluster info

Cluster ID
j-1ZLE55BQNDBM3

Cluster configuration
Instance groups

Capacity
1 Primary | 1 Core | 1 Task

Applications

Amazon EMR version
emr-7.0.0

Installed applications
Hadoop 3.3.6, Hive 3.1.3, JupyterEnterpriseGateway 2.6.0, Livy 0.7.1, Spark 3.5.0

Cluster management

Log destination in Amazon S3
aws-logs-471112686636-us-east-1/elasticmapreduce

Persistent application UIs
Spark History Server
YARN timeline server
Tez UI

Primary node public DNS
ec2-3-88-25-181.compute-1.amazonaws.com
Connect to the Primary node using SSH

Status and time

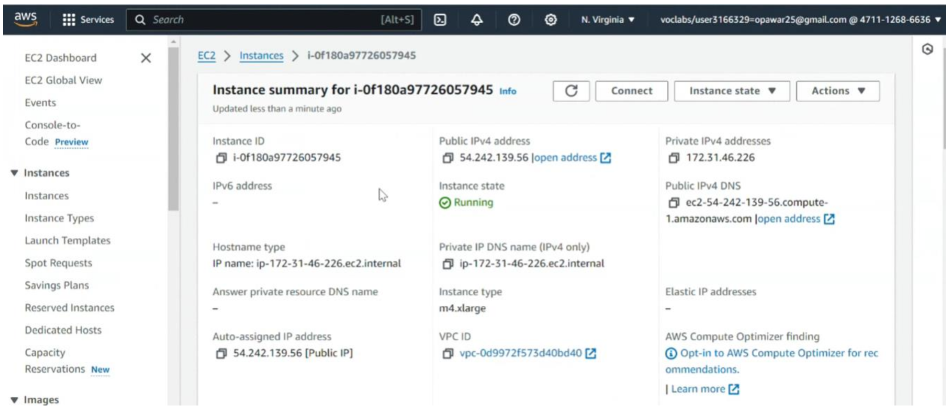
Status
Waiting

Creation time
April 07, 2024, 13:15 (UTC+05:30)

Elapsed time
6 minutes, 17 seconds

Step 3. Creating EC2 Instance

- Search for EC2 on Amazon Web Server
- Click on Launch instances to create a EC2 instance
- Now, give a name to your EC2 instance under the Name and Tags option
- Now, Select a desired Amazon Machine Image for example, Amazon Linux, macOS etc.
- Now select the instance type which is required for example t2.micro with 1 vCPU and 1GiB memory
- Now, select a key pair name from the drop-down menu or create a new key pair
- At last, click on Launch Instance to create a EC2 Instance
- The creation of the instance may take around 5-10 minutes to get active



Instance summary for i-0f180a97726057945

Updated less than a minute ago

Instance ID
i-0f180a97726057945

IPv6 address
–

Hostname type
IP name: ip-172-31-46-226.ec2.internal

Answer private resource DNS name
–

Auto-assigned IP address
54.242.139.56 [Public IP]

Public IPv4 address
54.242.139.56 |open address

Instance state
Running

Private IP DNS name (IPv4 only)
ip-172-31-46-226.ec2.internal

Instance type
m4.xlarge

VPC ID
vpc-0d9972f573d40bd40

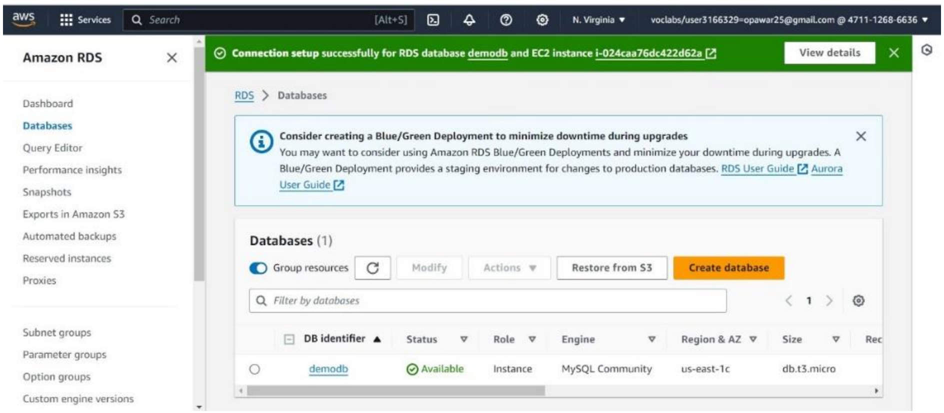
Private IPv4 addresses
172.31.46.226

Public IPv4 DNS
ec2-54-242-139-56.compute-1.amazonaws.com |open address

Elastic IP addresses
–

AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations.
| Learn more

Step 4. Connecting a setup between RDS database and EC2 instance



Connection setup successfully for RDS database demodb and EC2 instance i-024caa76dc422d62a

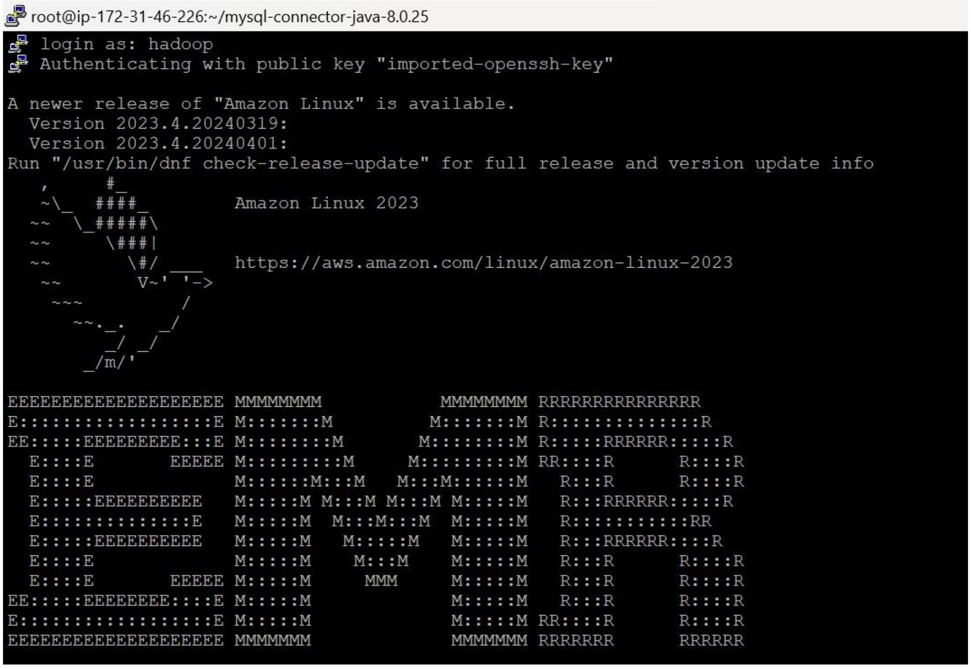
Consider creating a Blue/Green Deployment to minimize downtime during upgrades
You may want to consider using Amazon RDS Blue/Green Deployments and minimize your downtime during upgrades. A Blue/Green Deployment provides a staging environment for changes to production databases. RDS User Guide Aurora User Guide

Databases (1)

DB identifier	Status	Role	Engine	Region & AZ	Size	Rec
demodb	Available	Instance	MySQL Community	us-east-1c	db.t3.micro	

Step 5. Connecting EMR instance with the local computer using Putty.

- Firstly, copy the Primary node public DNS from the EMR cluster and paste it as a host name or IP Address in Putty
- Now, click on credentials under the Auth menu from SSH and browse for private key file for authentication.
- After inserting the private key, click open and the connection between EMR instance with the local computer will be executed.

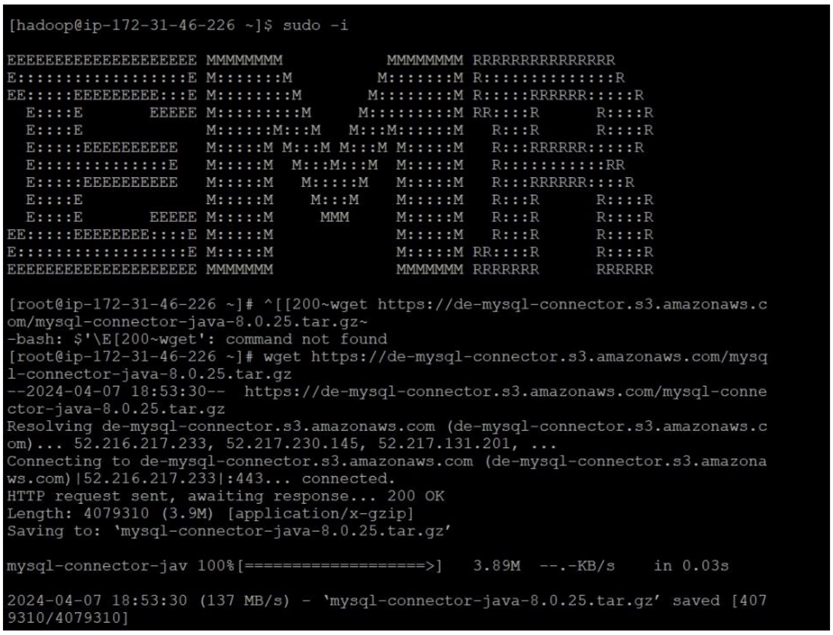


root@ip-172-31-46-226:~/mysql-connector-java-8.0.25

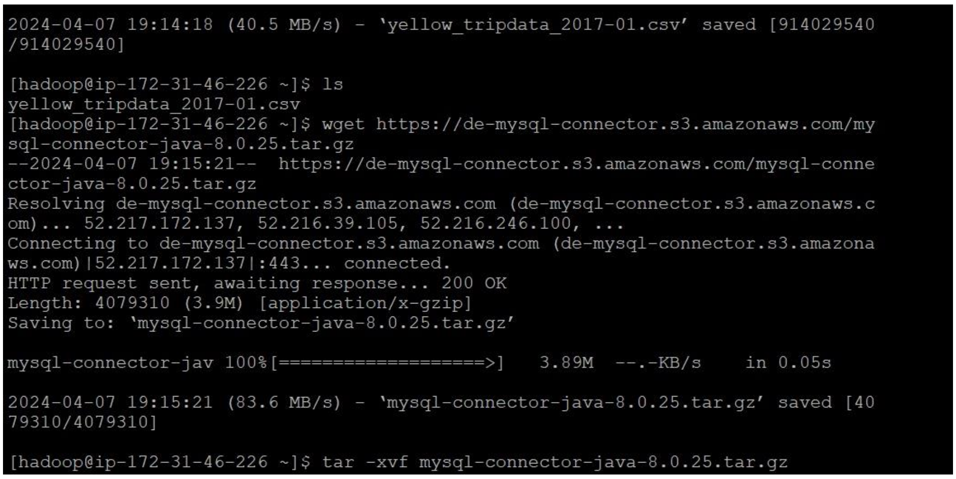
```
login as: hadoop
Authenticating with public key "imported-openssh-key"

A newer release of "Amazon Linux" is available.
  Version 2023.4.20240319:
  Version 2023.4.20240401:
Run "/usr/bin/dnf check-release-update" for full release and version update info
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'

EEEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M:::::::M         M:::::::M R::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R:::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E:::::EEEEEEEEEE   M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R:::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE::::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR
```

Step 6: Connecting to MySQL database remotely through putty using following commands

1. wget https://de-mysql-connector.s3.amazonaws.com/mysql-connector-java-8.0.25.tar.gz
2. tar -xvf mysql-connector-java-8.0.25.tar.gz
3. cd mysql-connector-java-8.0.25/
4. sudo cp mysql-connector-java-8.0.25.jar /usr/lib/sqoop/lib/

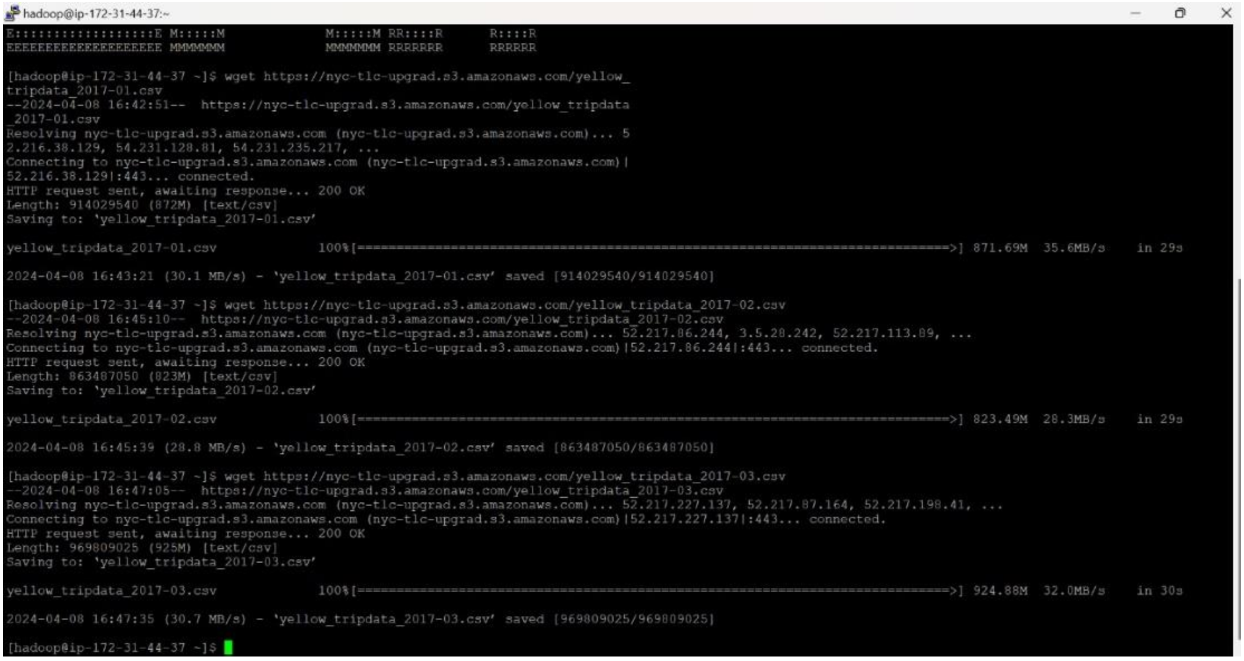


```
[hadoop@ip-172-31-46-226 ~]$ sudo -i

EEEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M:::::::M         M:::::::M R::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R:::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E:::::EEEEEEEEEE   M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R:::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE::::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR

[root@ip-172-31-46-226 ~]# ^[[200~wget https://de-mysql-connector.s3.amazonaws.c
om/mysql-connector-java-8.0.25.tar.gz~
-bash: $'\E[200~wget': command not found
[root@ip-172-31-46-226 ~]# wget https://de-mysql-connector.s3.amazonaws.com/mysq
l-connector-java-8.0.25.tar.gz
--2024-04-07 18:53:30--  https://de-mysql-connector.s3.amazonaws.com/mysql-conne
ctor-java-8.0.25.tar.gz
Resolving de-mysql-connector.s3.amazonaws.com (de-mysql-connector.s3.amazonaws.c
om)... 52.216.217.233, 52.217.230.145, 52.217.131.201, ...
Connecting to de-mysql-connector.s3.amazonaws.com (de-mysql-connector.s3.amazona
ws.com)|52.216.217.233|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4079310 (3.9M) [application/x-gzip]
Saving to: ‘mysql-connector-java-8.0.25.tar.gz’

mysql-connector-jav 100%[===================>]   3.89M  --.-KB/s    in 0.03s

2024-04-07 18:53:30 (137 MB/s) - ‘mysql-connector-java-8.0.25.tar.gz’ saved [407
9310/4079310]
```



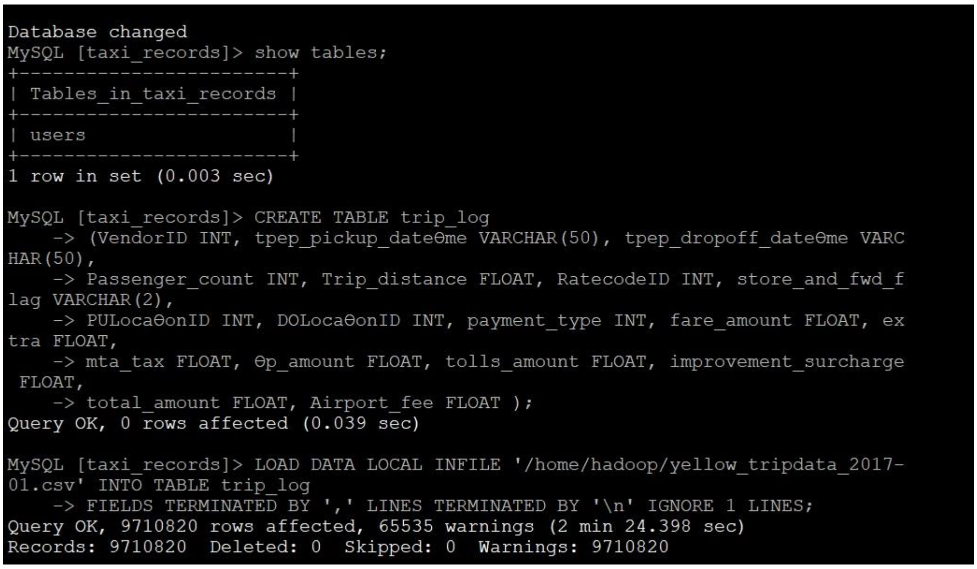
```
2024-04-07 19:14:18 (40.5 MB/s) - ‘yellow_tripdata_2017-01.csv’ saved [914029540
/914029540]

[hadoop@ip-172-31-46-226 ~]$ ls
yellow_tripdata_2017-01.csv
[hadoop@ip-172-31-46-226 ~]$ wget https://de-mysql-connector.s3.amazonaws.com/my
sql-connector-java-8.0.25.tar.gz
--2024-04-07 19:15:21--  https://de-mysql-connector.s3.amazonaws.com/mysql-conne
ctor-java-8.0.25.tar.gz
Resolving de-mysql-connector.s3.amazonaws.com (de-mysql-connector.s3.amazonaws.c
om)... 52.217.172.137, 52.216.39.105, 52.216.246.100, ...
Connecting to de-mysql-connector.s3.amazonaws.com (de-mysql-connector.s3.amazona
ws.com)|52.217.172.137|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4079310 (3.9M) [application/x-gzip]
Saving to: ‘mysql-connector-java-8.0.25.tar.gz’

mysql-connector-jav 100%[===================>]   3.89M  --.-KB/s    in 0.05s

2024-04-07 19:15:21 (83.6 MB/s) - ‘mysql-connector-java-8.0.25.tar.gz’ saved [40
79310/4079310]

[hadoop@ip-172-31-46-226 ~]$ tar -xvf mysql-connector-java-8.0.25.tar.gz
```

Step 7: Uploading the data from 2 given files



```
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR

[hadoop@ip-172-31-44-37 ~]$ wget https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_
tripdata_2017-01.csv
--2024-04-08 16:42:51--  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)... 5
2.216.38.129, 54.231.128.81, 54.231.235.217, ...
Connecting to nyc-tlc-upgrad.s3.amazonaws.com (nyc-tlc-upgrad.s3.amazonaws.com)|
52.216.38.129|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 914029540 (872M) [text/csv]
Saving to: 'yellow_tripdata_2017-01.csv'

yellow_tripdata_2017-01.csv           100%[=============================================================================>] 871.69M  35.6MB/s    in 29s

2024-04-08 16:43:21 (30.1 MB/s) - 'yellow_tripdata_2017-01.csv' saved [914029540/914029540]

[hadoop@ip-172-31-44-37 ~]$ wget https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-02.csv
--2024-04-08 16:45:10--  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)... 52.217.86.244, 3.5.28.242, 52.217.113.89, ...
Connecting to nyc-tlc-upgrad.s3.amazonaws.com (nyc-tlc-upgrad.s3.amazonaws.com)|52.217.86.244|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 863487050 (823M) [text/csv]
Saving to: 'yellow_tripdata_2017-02.csv'

yellow_tripdata_2017-02.csv           100%[=============================================================================>] 823.49M  28.3MB/s    in 29s

2024-04-08 16:45:39 (28.8 MB/s) - 'yellow_tripdata_2017-02.csv' saved [863487050/863487050]

[hadoop@ip-172-31-44-37 ~]$ wget https://nyc-tlc-upgrad.s3.amazonaws.com/yellow_tripdata_2017-03.csv
--2024-04-08 16:47:05--  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)... 52.217.227.137, 52.217.87.164, 52.217.198.41, ...
Connecting to nyc-tlc-upgrad.s3.amazonaws.com (nyc-tlc-upgrad.s3.amazonaws.com)|52.217.227.137|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 969809025 (925M) [text/csv]
Saving to: 'yellow_tripdata_2017-03.csv'

yellow_tripdata_2017-03.csv           100%[=============================================================================>] 924.88M  32.0MB/s    in 30s

2024-04-08 16:47:35 (30.7 MB/s) - 'yellow_tripdata_2017-03.csv' saved [969809025/969809025]

[hadoop@ip-172-31-44-37 ~]$
```

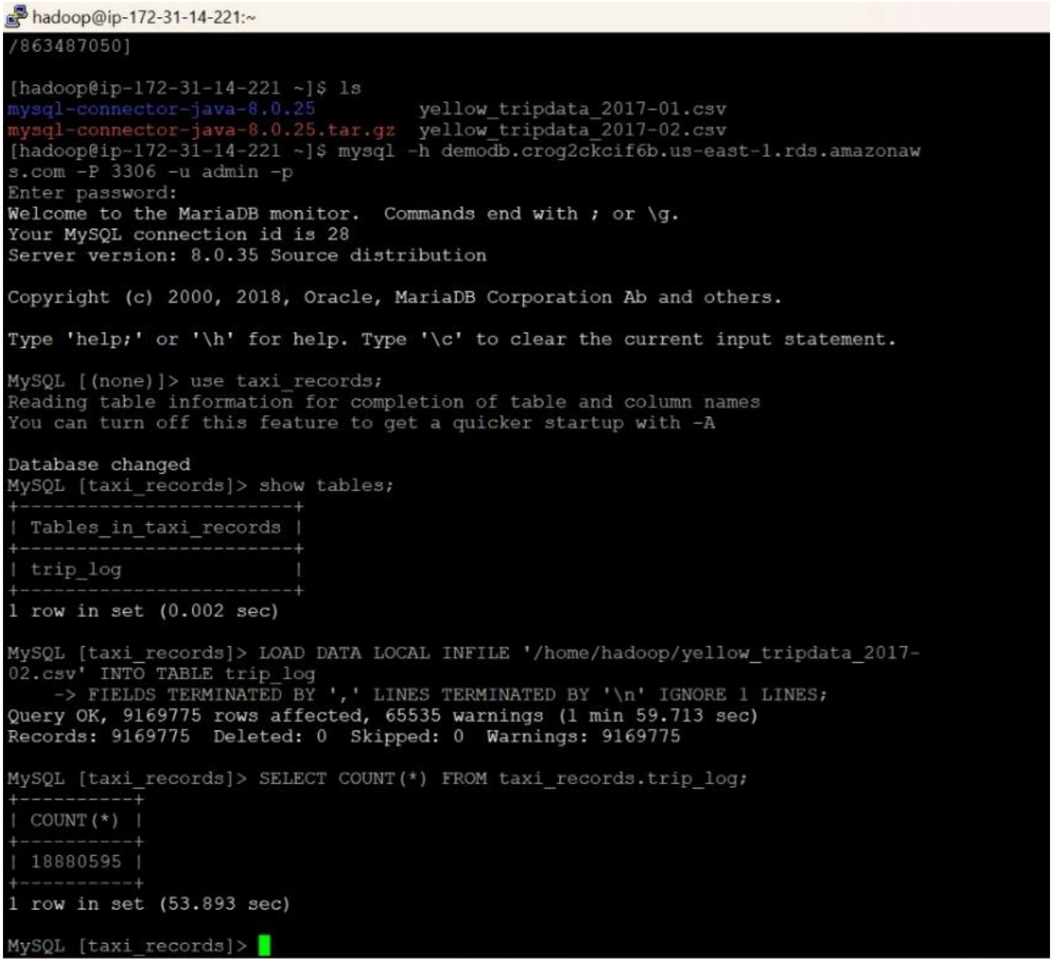
Step 8: Creating schema using MySQL connection and also creating table trip_log



```
Database changed
MySQL [taxi_records]> show tables;
+------------------------+
| Tables_in_taxi_records |
+------------------------+
| users                  |
+------------------------+
1 row in set (0.003 sec)

MySQL [taxi_records]> CREATE TABLE trip_log
    -> (VendorID INT, tpep_pickup_dateΘme VARCHAR(50), tpep_dropoff_dateΘme VARC
HAR(50),
    -> Passenger_count INT, Trip_distance FLOAT, RatecodeID INT, store_and_fwd_f
lag VARCHAR(2),
    -> PULocaΘonID INT, DOLocaΘonID INT, payment_type INT, fare_amount FLOAT, ex
tra FLOAT,
    -> mta_tax FLOAT, Θp_amount FLOAT, tolls_amount FLOAT, improvement_surcharge
 FLOAT,
    -> total_amount FLOAT, Airport_fee FLOAT );
Query OK, 0 rows affected (0.039 sec)

MySQL [taxi_records]> LOAD DATA LOCAL INFILE '/home/hadoop/yellow_tripdata_2017-
01.csv' INTO TABLE trip_log
    -> FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' IGNORE 1 LINES;
Query OK, 9710820 rows affected, 65535 warnings (2 min 24.398 sec)
Records: 9710820  Deleted: 0  Skipped: 0  Warnings: 9710820
```

.

Step 9: Finally checking the count of the given task 1 files.



hadoop@ip-172-31-14-221:~

```
/863487050]

[hadoop@ip-172-31-14-221 ~]$ ls
mysql-connector-java-8.0.25         yellow_tripdata_2017-01.csv
mysql-connector-java-8.0.25.tar.gz  yellow_tripdata_2017-02.csv
[hadoop@ip-172-31-14-221 ~]$ mysql -h demodb.crog2ckcif6b.us-east-1.rds.amazonaw
s.com -P 3306 -u admin -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 28
Server version: 8.0.35 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> use taxi_records;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MySQL [taxi_records]> show tables;
+------------------------+
| Tables_in_taxi_records |
+------------------------+
| trip_log               |
+------------------------+
1 row in set (0.002 sec)

MySQL [taxi_records]> LOAD DATA LOCAL INFILE '/home/hadoop/yellow_tripdata_2017-
02.csv' INTO TABLE trip_log
    -> FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' IGNORE 1 LINES;
Query OK, 9169775 rows affected, 65535 warnings (1 min 59.713 sec)
Records: 9169775  Deleted: 0  Skipped: 0  Warnings: 9169775

MySQL [taxi_records]> SELECT COUNT(*) FROM taxi_records.trip_log;
+----------+
| COUNT(*) |
+----------+
| 18880595 |
+----------+
1 row in set (53.893 sec)

MySQL [taxi_records]>
```