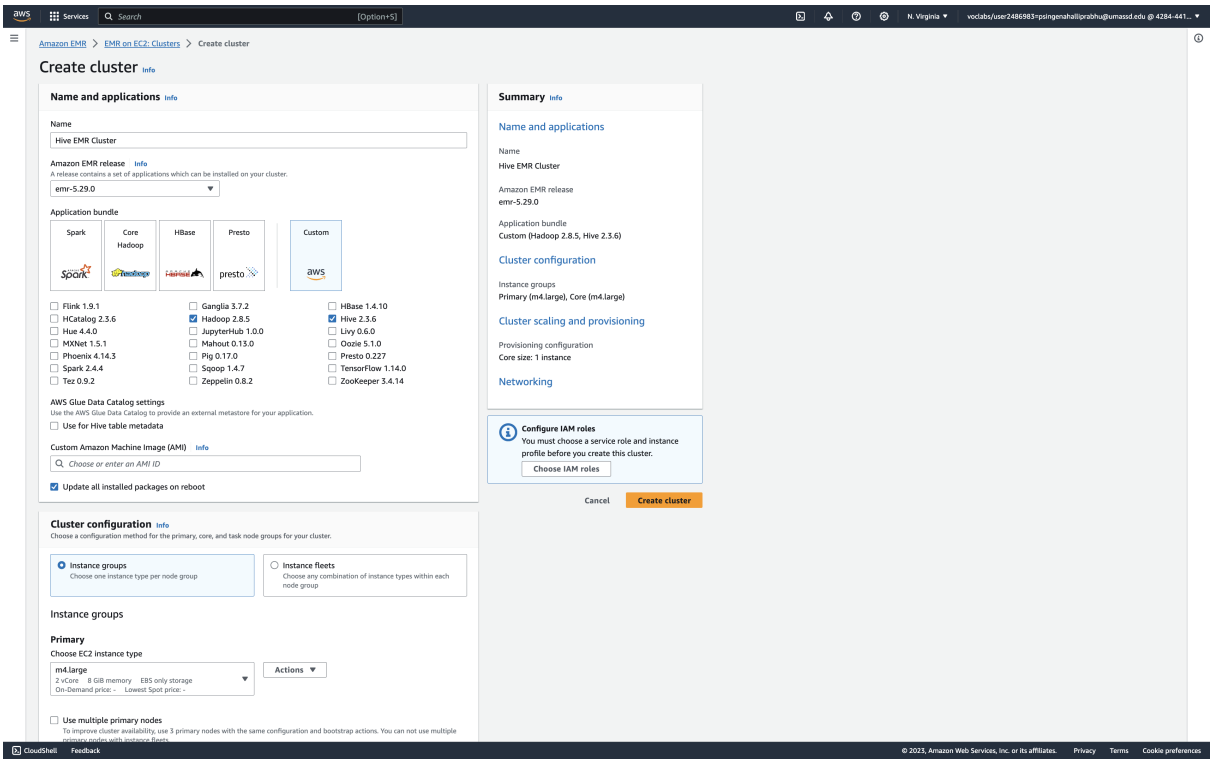
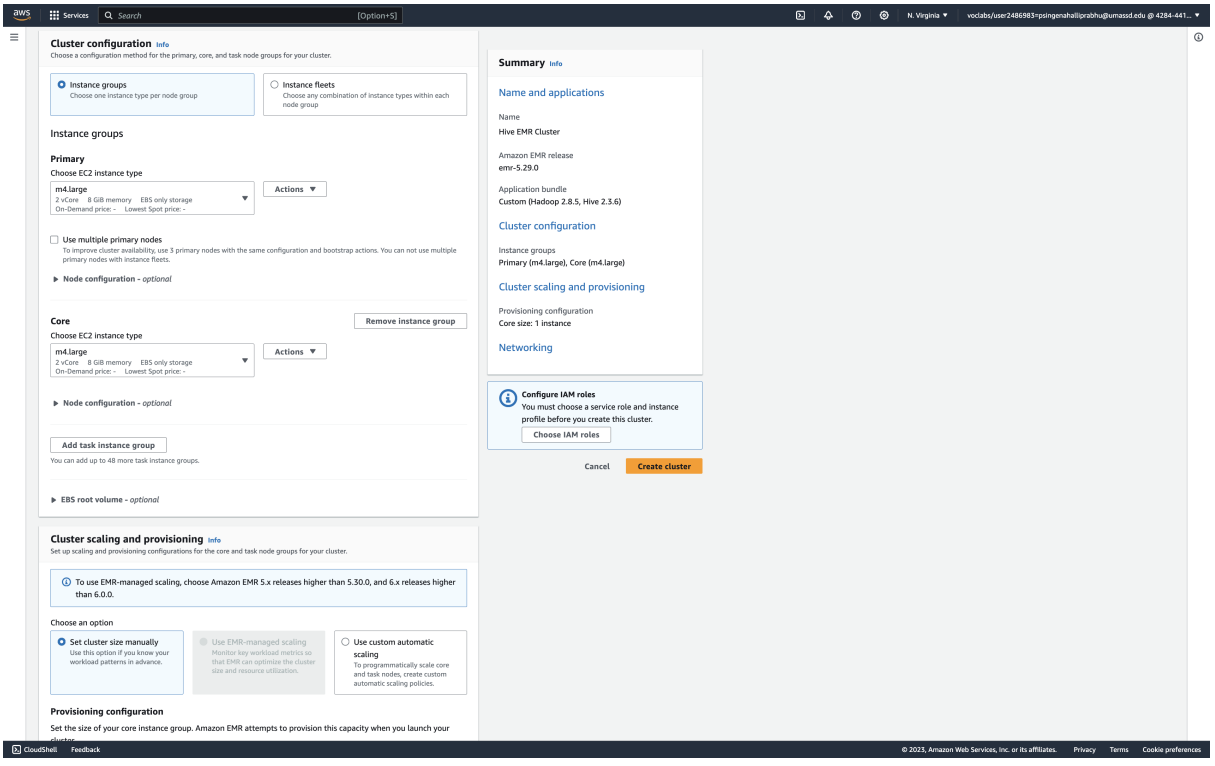
CIS 602: Big Data – Homework 5

Pradyoth Singenahalli Prabhu - 02071847

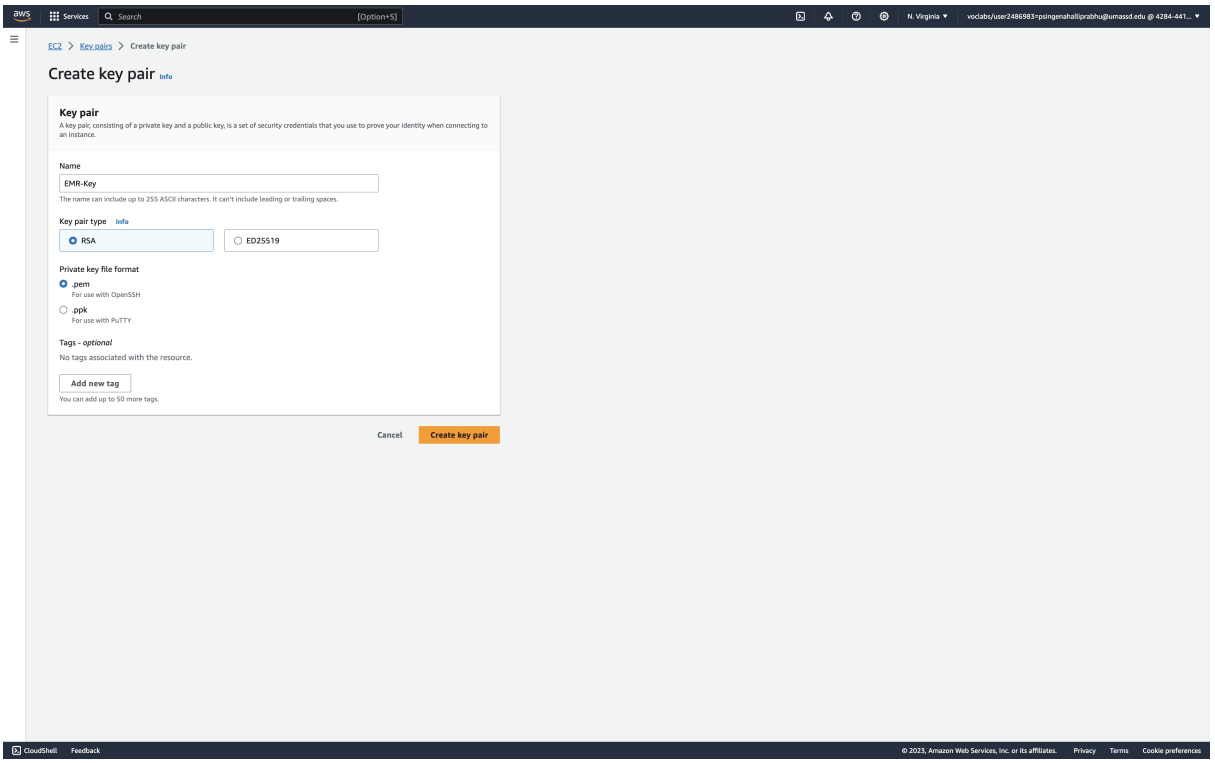
Task 1: Launching an Amazon EMR cluster

Creating EMR cluster:

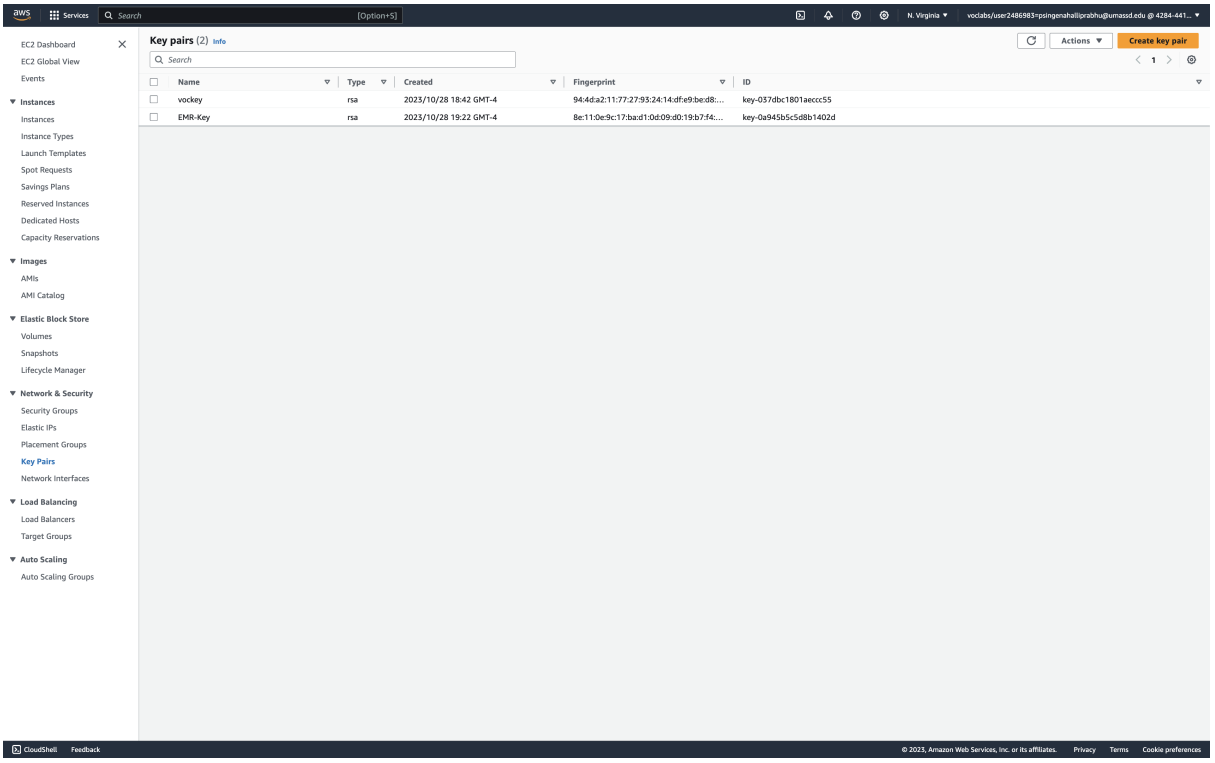


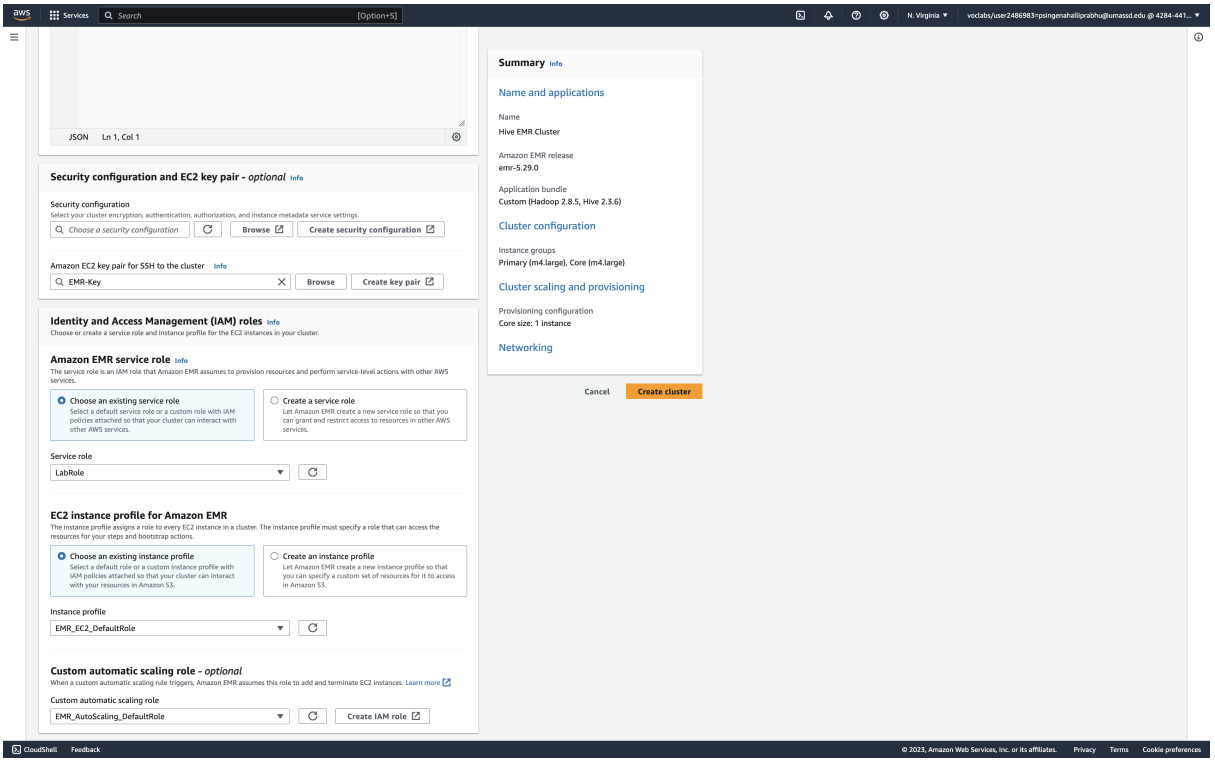


Creating EMR Key:

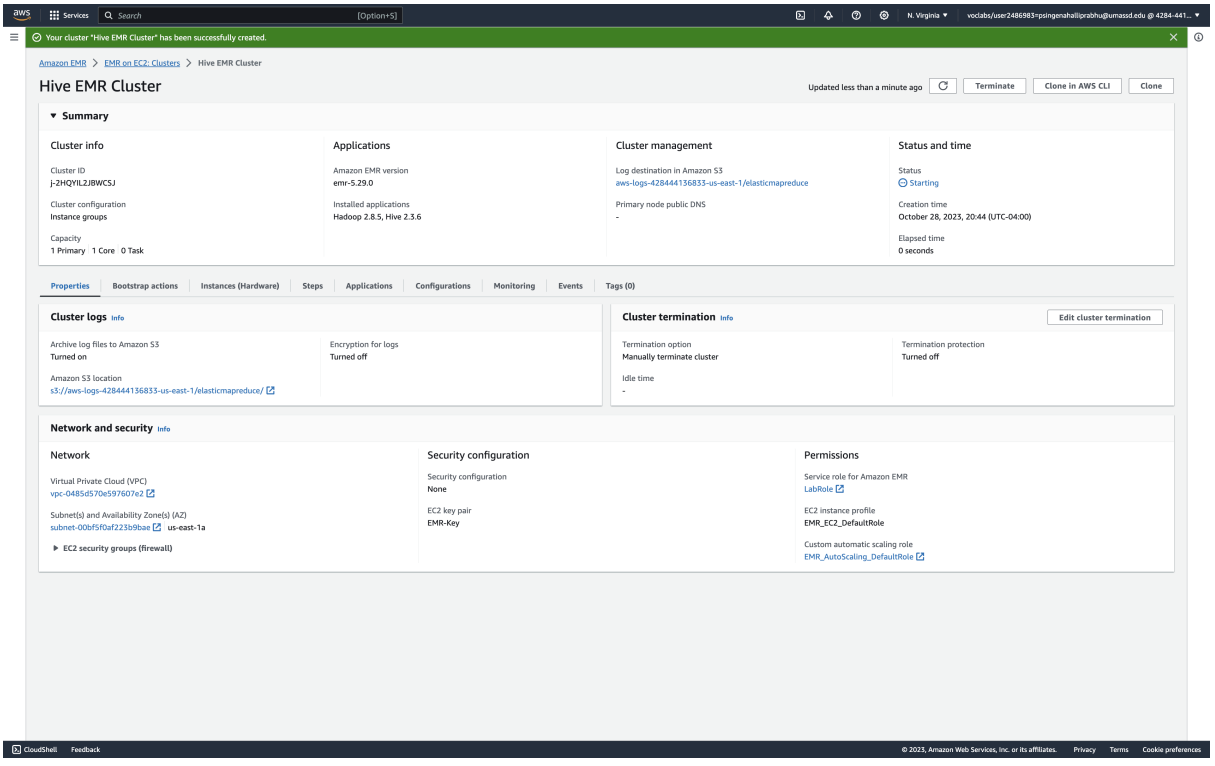


Successfully Created the EMR-Key:

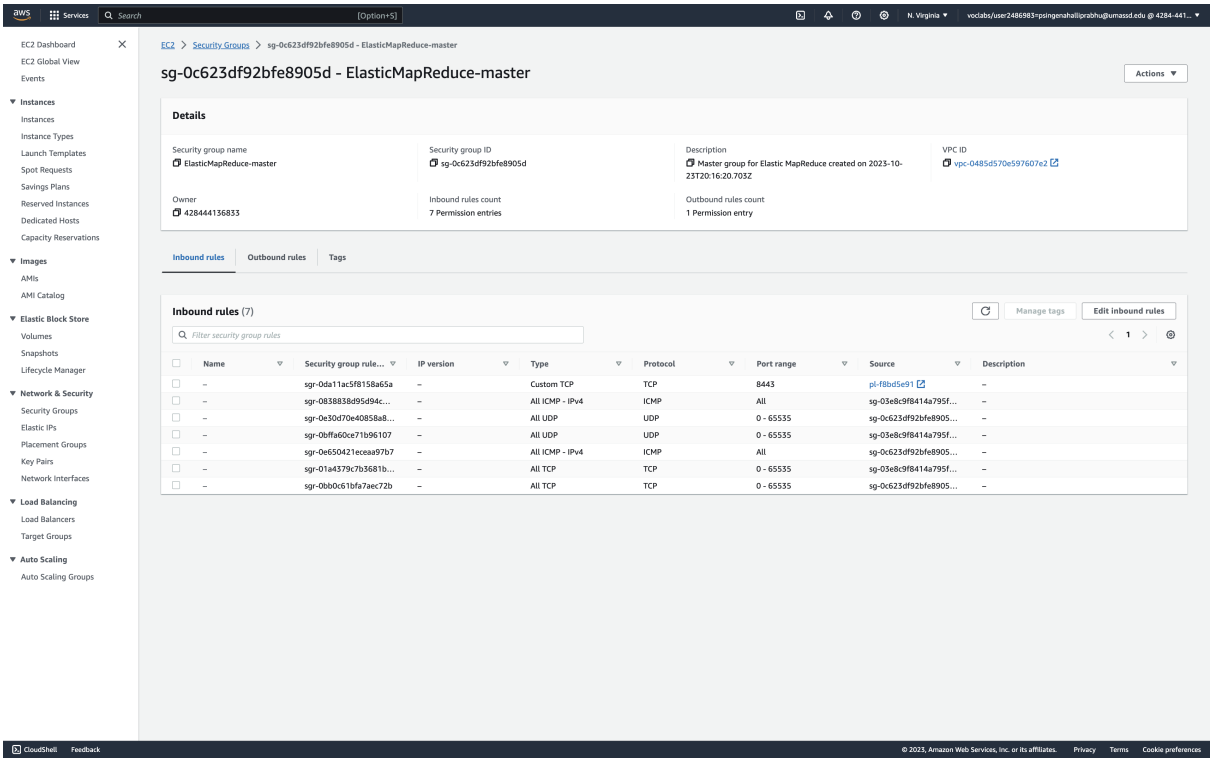


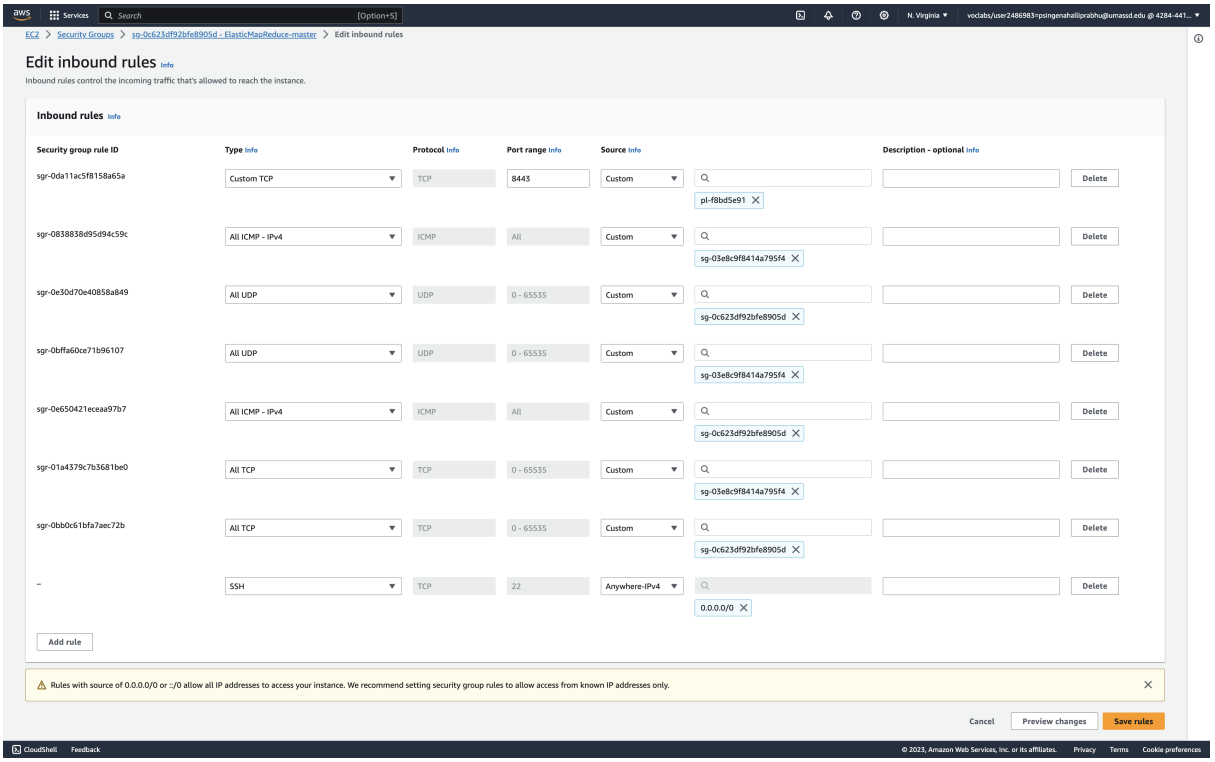


Created the cluster successfully:

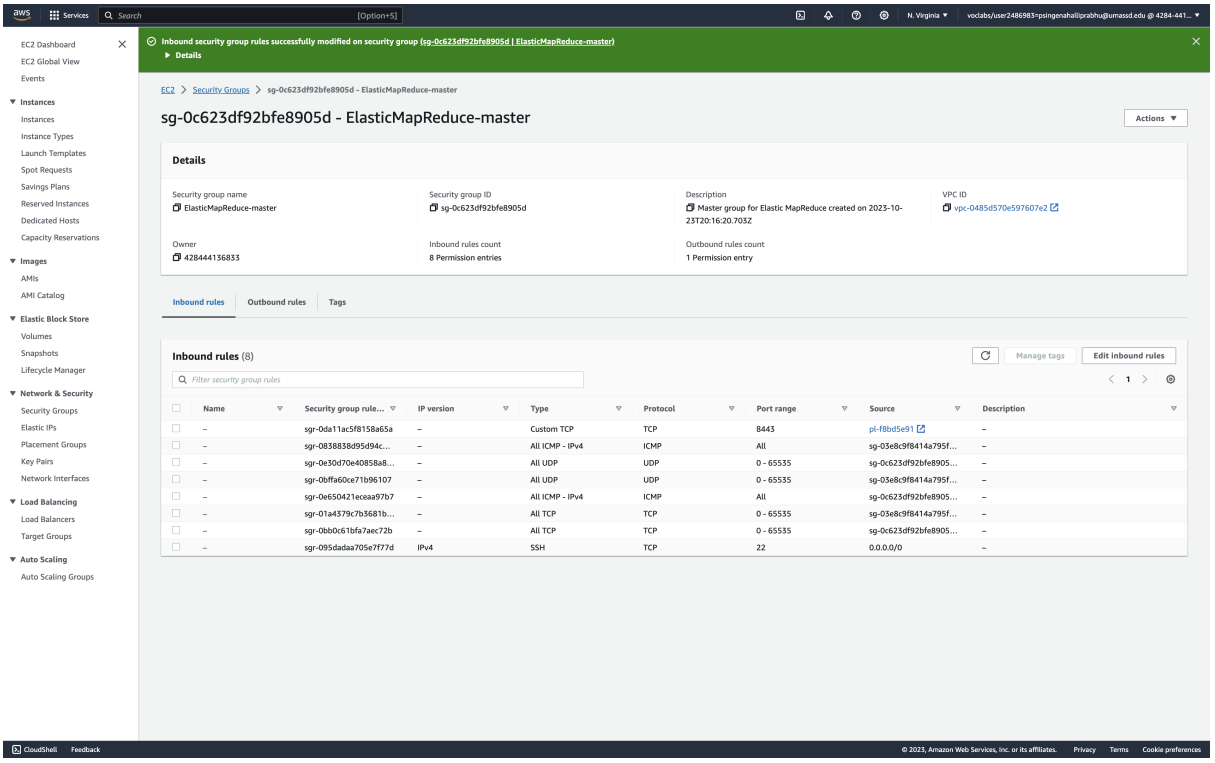


Configure the security group for the main node to allow SSH connections:



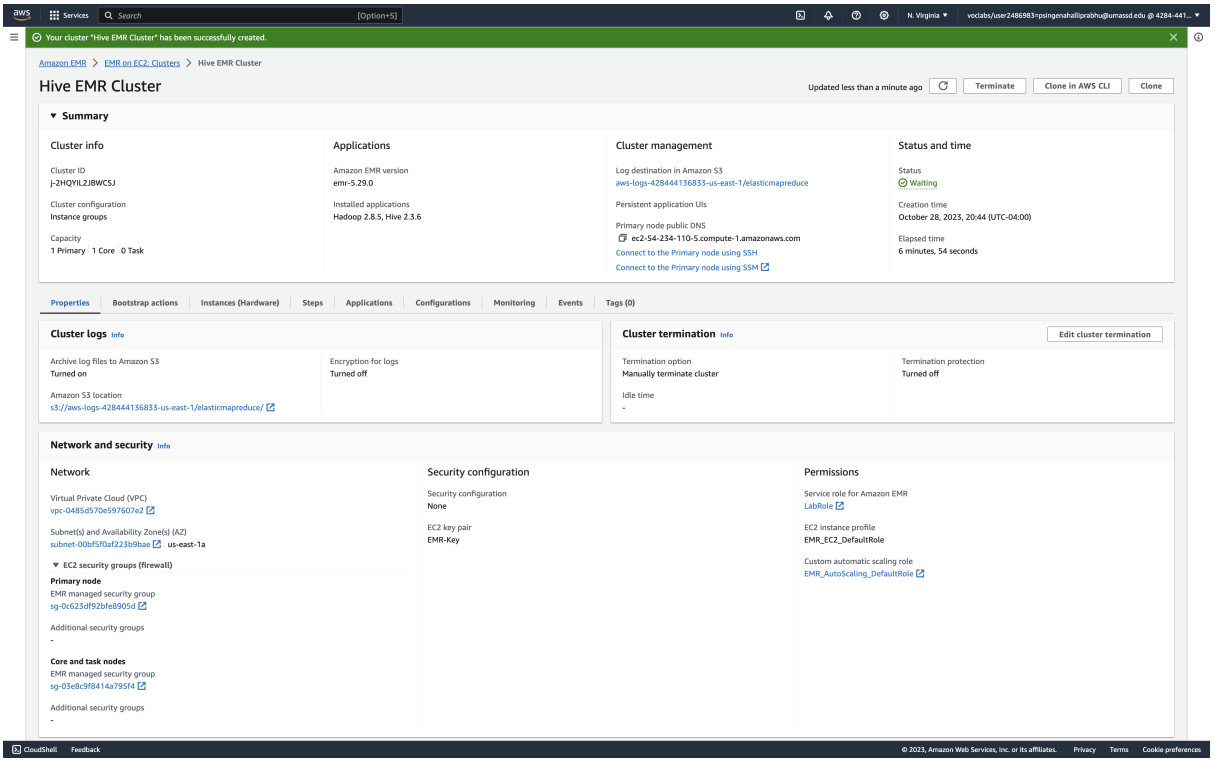


Successfully Created:



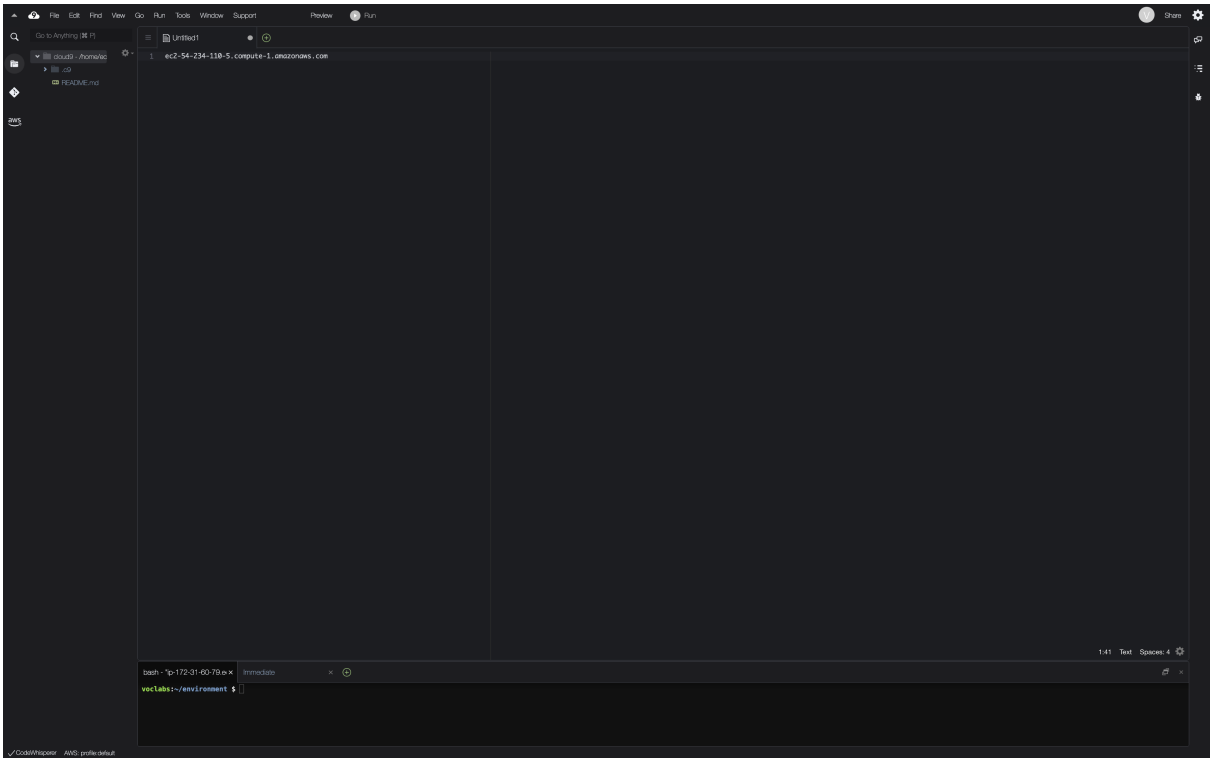
Successfully Created EMR Cluster (Hive EMR Cluster):

Status changed to **Waiting**.

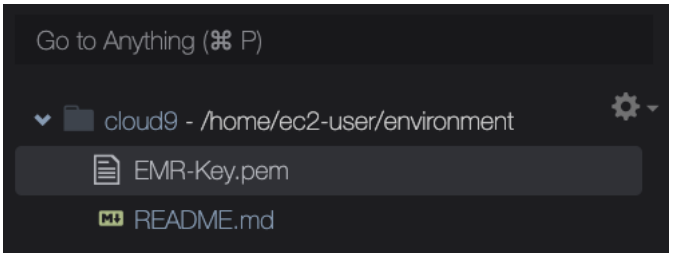


Task 2: Connecting to the Hadoop main node by using SSH

Copied Primary node public DNS: `ec2-54-234-110-5.compute-1.amazonaws.com`



Uploading the EMR-Key:



Modify the permissions on the key pair file so that the SSH software will use it:

Command:

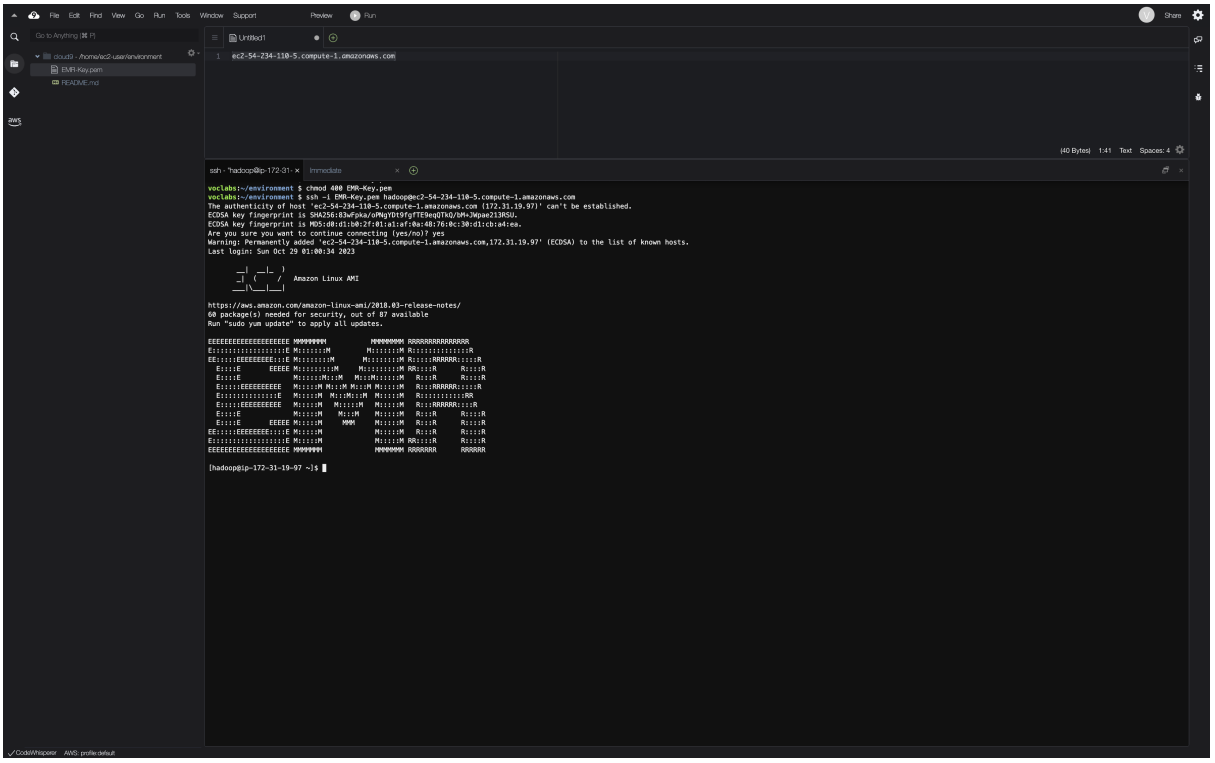
```
chmod 400 EMR-Key.pem
```

Establish an SSH connection to the cluster's main node:

Bash Command:

```
ssh -i EMR-Key.pem hadoop@ec2-54-234-110-5.compute-1.amazonaws.com
```

Successfully established the connection:



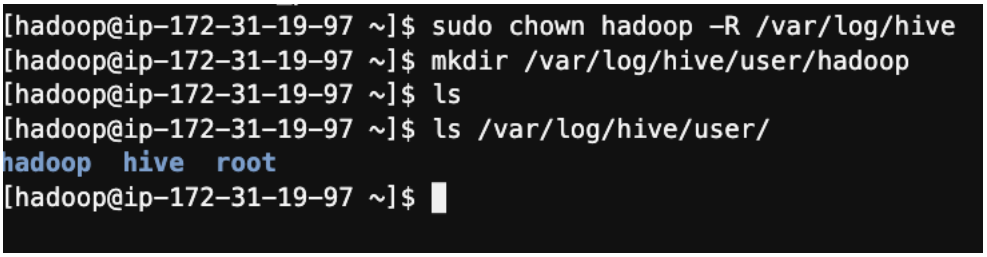
Task 3: Running Hive interactively

To create a logging directory for Hive, running the following commands:

Command:

```
sudo chown hadoop -R /var/log/hive
mkdir /var/log/hive/user/hadoop
```

Successfully Created the folder:



Configure Hive:

To retrieve the name of the S3 bucket, we are running the following command:

Command:

```
aws s3 ls /
```

```
[hadoop@ip-172-31-19-97 ~]$ aws s3 ls /
2023-10-28 23:31:30 aws-logs-428444136833-us-east-1
[hadoop@ip-172-31-19-97 ~]$
```

So S3 bucket name is: `aws-logs-428444136833-us-east-1`

Running the hive

```
hive -d SAMPLE=s3://aws-tc-largeobjects/CUR-TF-200-ACDENG-1/emr-lab -d DAY=2009-04-13 -d HOUR=08 -d NEXT_DAY=2009-04-13 -d NEXT_HOUR=0
```

Output:

```
[hadoop@ip-172-31-19-97 ~]$ hive -d SAMPLE=s3://aws-tc-largeobjects/CUR-TF-200-ACDENG-1/emr-lab -d DAY=2009-04-13 -d HOUR=08 -d NEXT_DAY=2009-04-13 -d NEXT_HOUR=09 -d OUTPUT=s3://aws-logs-428444136833-us-east-1

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.properties Async: true
hive>
```

Task 4: Creating tables out of source data by using Hive

To create an external table named `impressions`:

Command:

```
CREATE EXTERNAL TABLE impressions (
requestBeginTime string,
adId string,
impressionId string,
referrer string,
userAgent string,
userCookie string,
ip string)
PARTITIONED BY (dt string)
ROW FORMAT  serde 'org.apache.hive.hcatalog.data.JsonSerDe' with serdeproperties
('paths'='requestBeginTime, adId, impressionId, referrer, userAgent, userCookie, ip')
LOCATION '${SAMPLE}/tables/impressions';
```

```
hive> CREATE EXTERNAL TABLE impressions (
    > requestBeginTime string,
    > adId string,
    > impressionId string,
    > referrer string,
    > userAgent string,
    > userCookie string,
    > ip string)
    > PARTITIONED BY (dt string)
    > ROW FORMAT  serde 'org.apache.hive.hcatalog.data.JsonSerDe' with serdeproperties
    > ('paths'='requestBeginTime, adId, impressionId, referrer, userAgent, userCookie, ip')
    > LOCATION '${SAMPLE}/tables/impressions';
OK
Time taken: 18.788 seconds
hive>
    >
```

Output:

```
OK
Time taken: 18.788 seconds
```

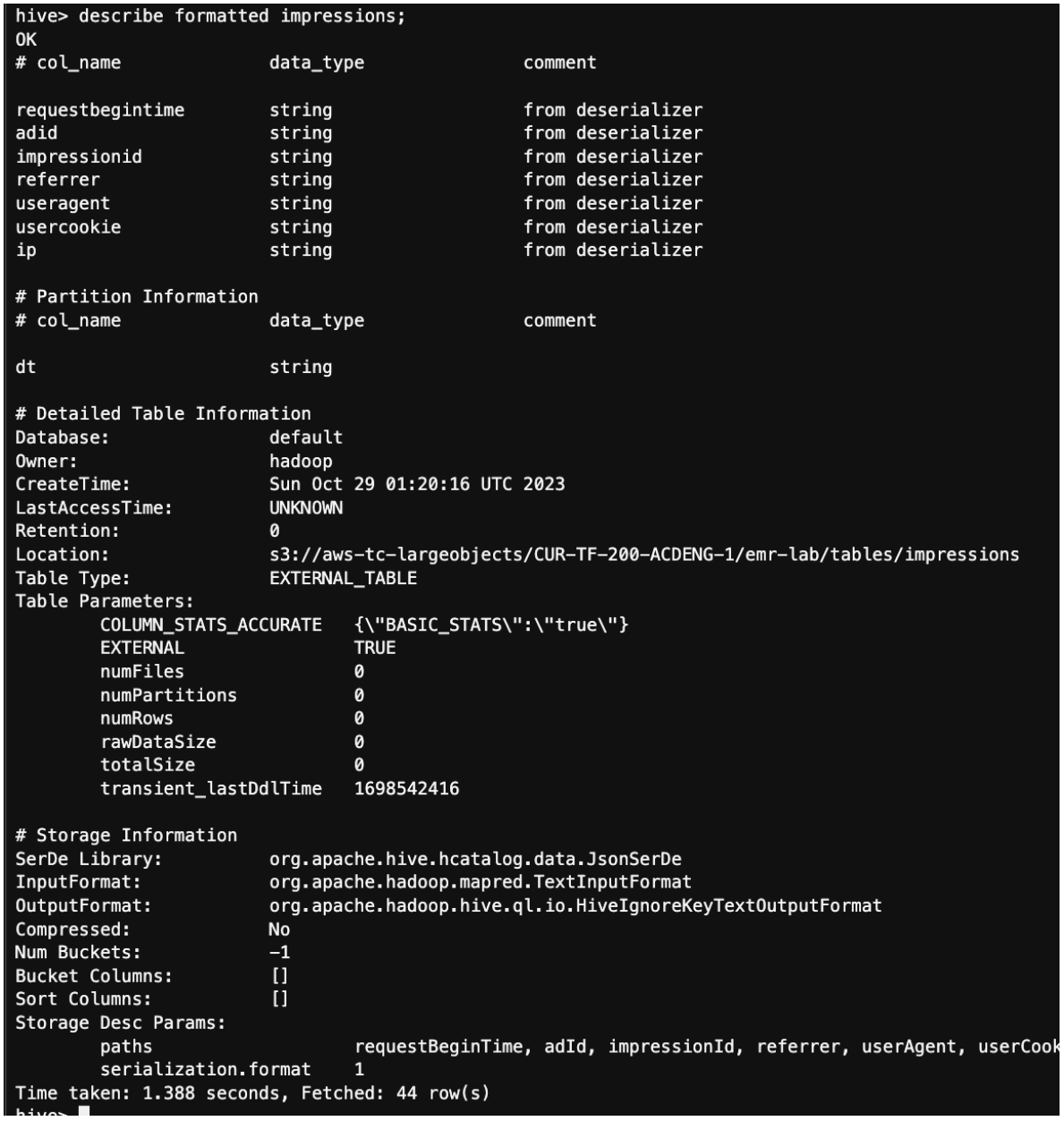
Updating the Hive metadata for the `impressions` table to include all partitions:

To see how many partitions the `impressions` table currently has:

Command:

```
describe formatted impressions;
```

Output:



Number of partitions is `0`:

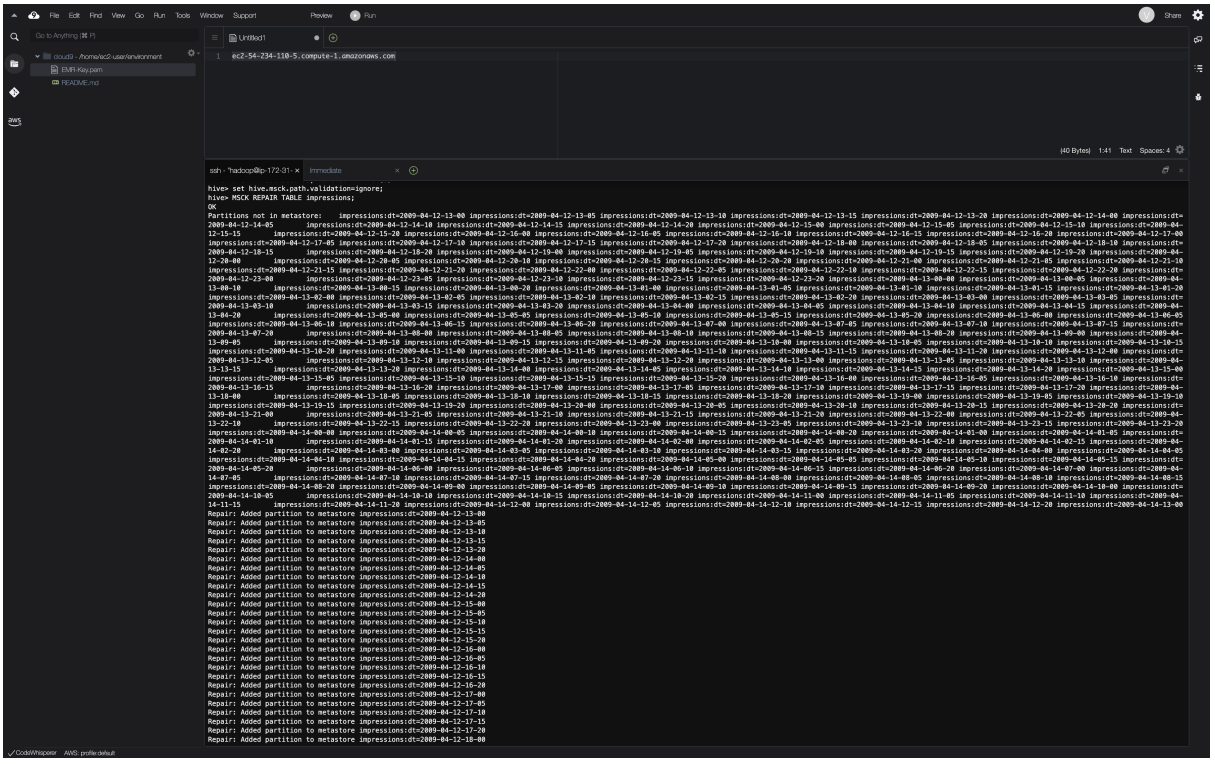
```
Table Parameters:
        COLUMN_STATS_ACCURATE   {\"BASIC_STATS\":\"true\"}
        EXTERNAL                TRUE
        numFiles                0
        numPartitions           0
        numRows                 0
        rawDataSize             0
        totalSize               0
        transient_lastDdlTime   1698542416
```

To inspect the input data and apply partitions to the metastore, running the following command:

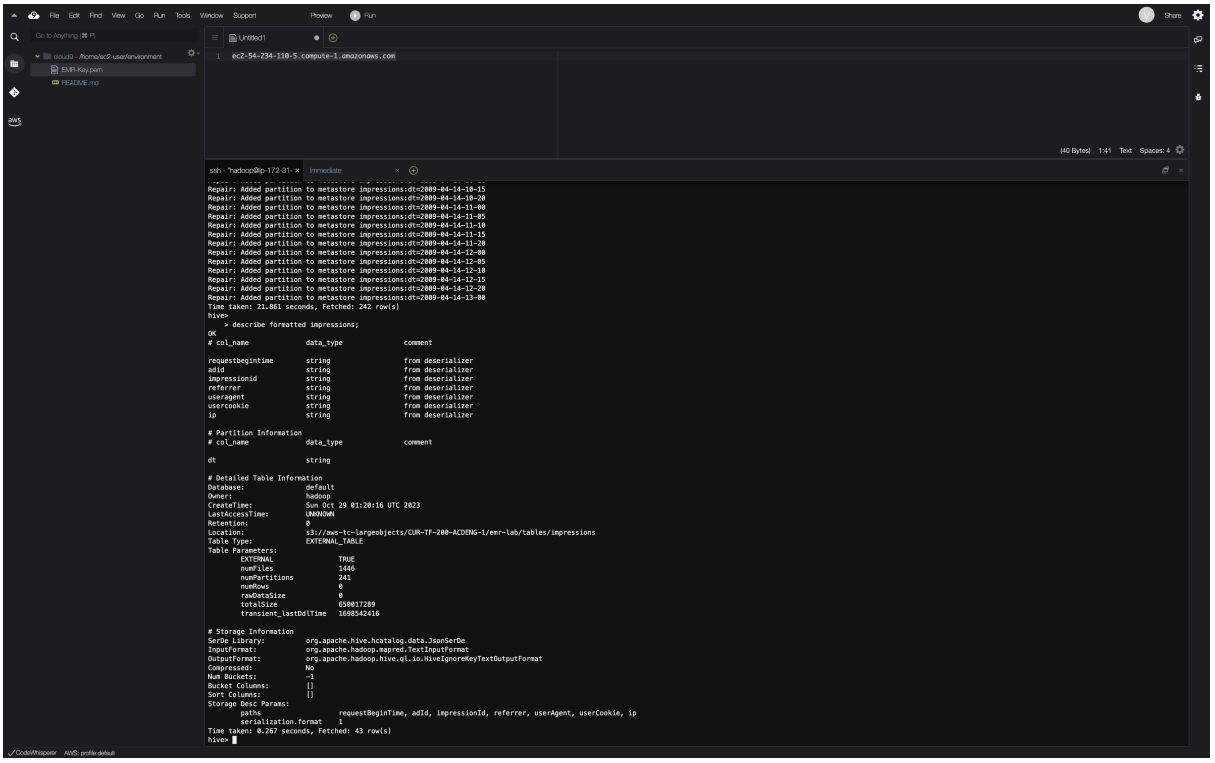
Command:

```
set hive.msck.path.validation=ignore;
MSCK REPAIR TABLE impressions;
```

Output:



After running the command `describe formatted impressions;`:



We can see that now partitions are created.

Creating another external table and again discover its partitions. This table will be named `clicks`, and it references click-stream logs

Command:

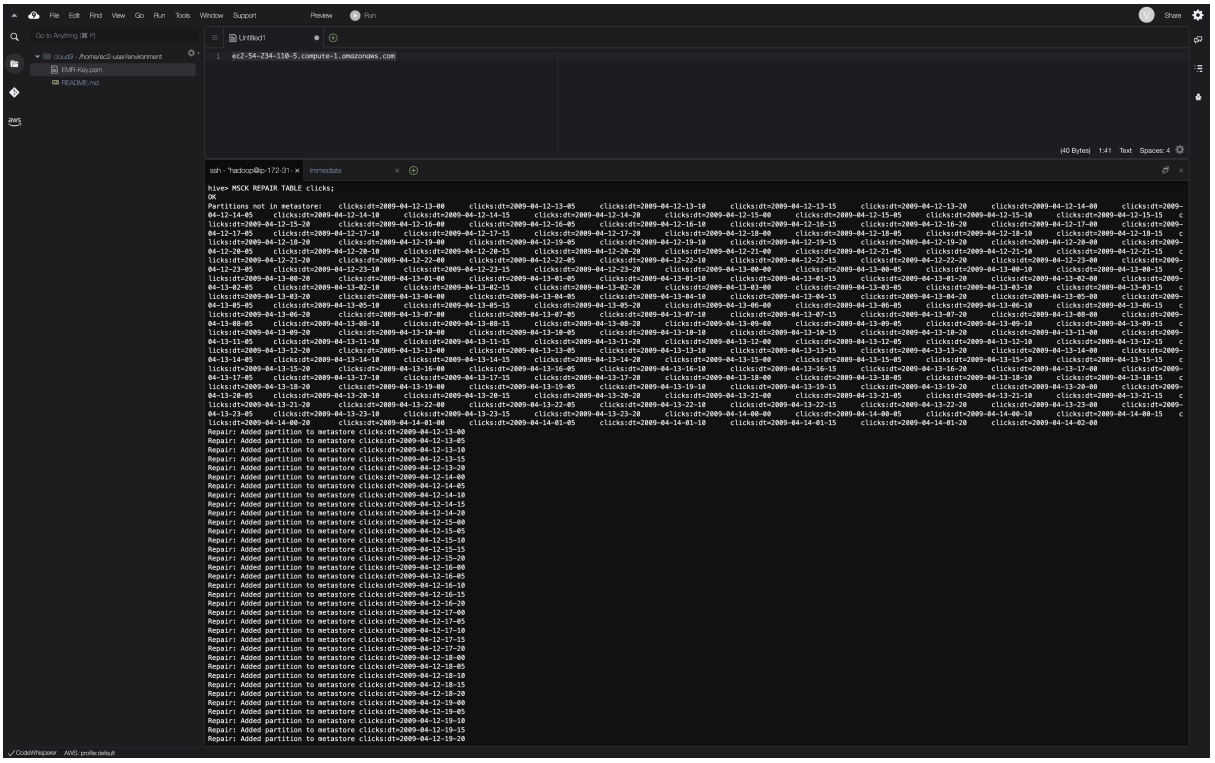
```
CREATE EXTERNAL TABLE clicks (impressionId string, adId string)
PARTITIONED BY (dt string)
ROW FORMAT  serde 'org.apache.hive.hcatalog.data.JsonSerDe'
WITH SERDEPROPERTIES ('paths'='impressionId')
LOCATION '${SAMPLE}/tables/clicks';
```

Successfully Created the table:

To return all partitions from the click-stream log data, running the following command:

```
MSCK REPAIR TABLE clicks;
```

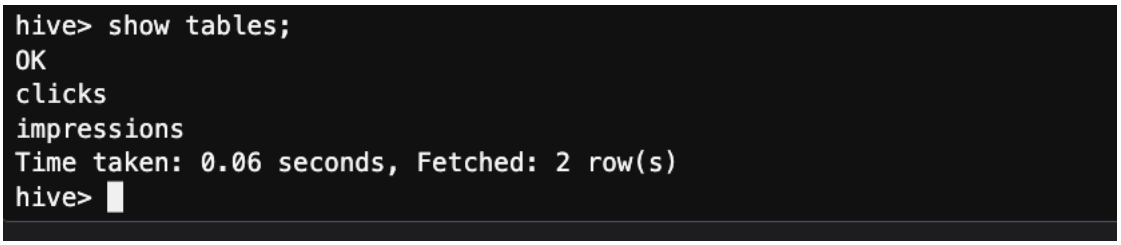
Output:



Verifying that the two tables now exist:

Command:

```
show tables;
```

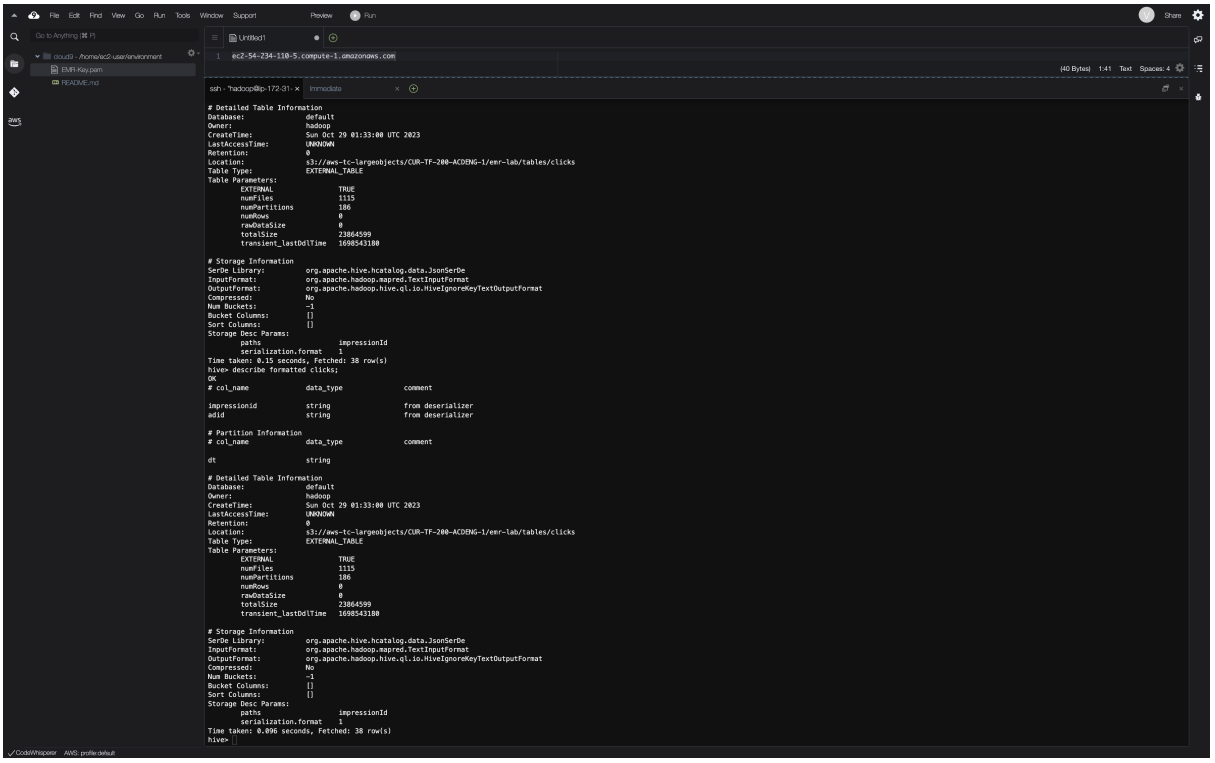


To describe the `clicks` table:

Command:

```
describe formatted clicks;
```

Output:



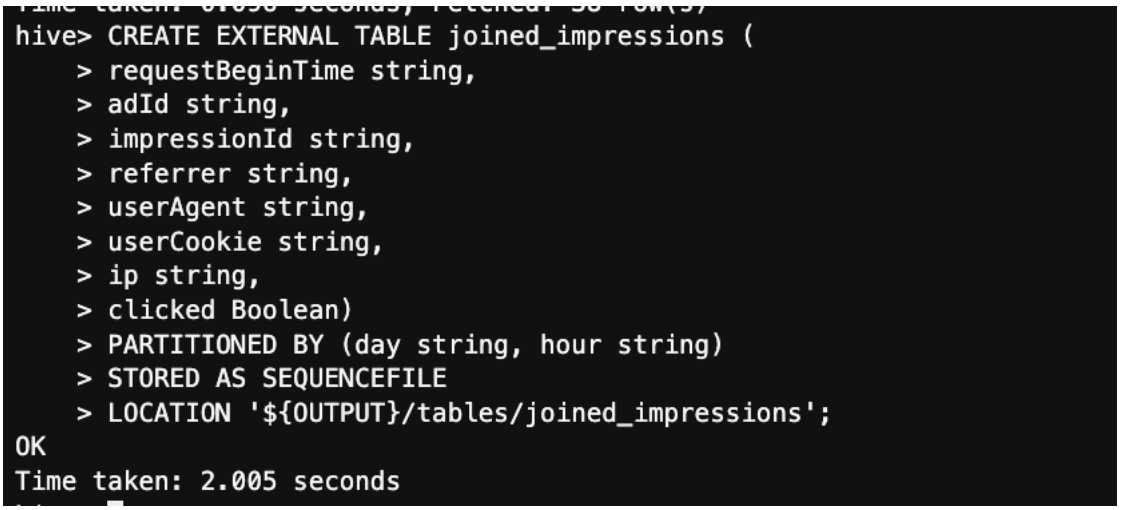
Task 5: Joining tables by using Hive

To create a new external table named `joined_impressions`, running the following command:

Command:

```
CREATE EXTERNAL TABLE joined_impressions (
requestBeginTime string,
adId string,
impressionId string,
referrer string,
userAgent string,
userCookie string,
ip string,
clicked Boolean)
PARTITIONED BY (day string, hour string)
STORED AS SEQUENCEFILE
LOCATION '${OUTPUT}/tables/joined_impressions';
```

Output:

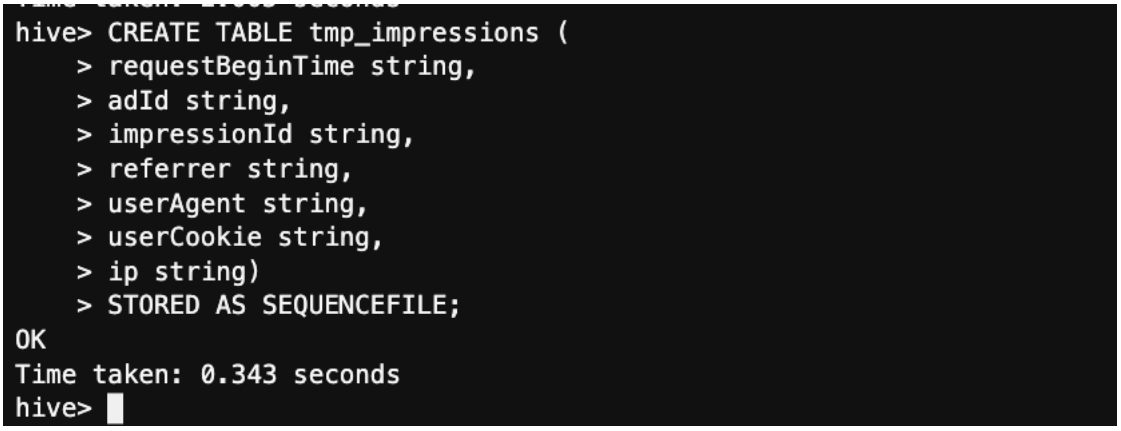


To create a new temporary table in HDFS named `tmp_impressions` to store intermediate data, running the following command:

Command:

```
CREATE TABLE tmp_impressions (
requestBeginTime string,
adId string,
impressionId string,
referrer string,
userAgent string,
userCookie string,
ip string)
STORED AS SEQUENCEFILE;
```

Output:



To insert impression log data for the time duration referenced, running the following command:

Command:

```
INSERT OVERWRITE TABLE tmp_impressions
SELECT
from_unixtime(cast((cast(i.requestBeginTime as bigint) / 1000) as int)) requestBeginTime,
i.adId,
i.impressionId,
i.referrer,
i.userAgent,
i.userCookie,
i.ip
   FROM impressions i
   WHERE i.dt >= '${DAY}-${HOUR}-00'
AND i.dt < '${NEXT_DAY}-${NEXT_HOUR}-00';
```

Output:

```
hive> INSERT OVERWRITE TABLE tmp_impressions
    > SELECT
    > from_unixtime(cast((cast(i.requestBeginTime as bigint) / 1000) as int)) requestBeginTime,
    > i.adId,
    > i.impressionId,
    > i.referrer,
    > i.userAgent,
    > i.userCookie,
    > i.ip
    >    FROM impressions i
    >    WHERE i.dt >= '${DAY}-${HOUR}-00'
    > AND i.dt < '${NEXT_DAY}-${NEXT_HOUR}-00';
Query ID = hadoop_20231029014340_33c7e875-1b0f-4f07-bac7-c721c08276c3
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1698540576797_0002)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 01/01  [==========================>>] 100%  ELAPSED TIME: 37.95 s
--------------------------------------------------------------------------------
Loading data to table default.tmp_impressions
OK
Time taken: 52.369 seconds
hive>
```

Create a temporary table named `tmp_clicks` and insert data into the table.

To create the table, running the following command:

Command:

```
CREATE TABLE tmp_clicks ( impressionId string, adId string )
STORED AS SEQUENCEFILE;
```

Output:

```
Time taken: 52.369 seconds
hive> CREATE TABLE tmp_clicks ( impressionId string, adId string )
    > STORED AS SEQUENCEFILE;
OK
Time taken: 0.062 seconds
hive>
```

To insert data into the `tmp_clicks` table, running the following command:

Command:

```
INSERT OVERWRITE TABLE tmp_clicks
SELECT impressionId, adId
FROM clicks c
WHERE c.dt >= '${DAY}-${HOUR}-00'
AND c.dt < '${NEXT_DAY}-${NEXT_HOUR}-20';
```

```
hive> INSERT OVERWRITE TABLE tmp_clicks
    > SELECT impressionId, adId
    > FROM clicks c
    > WHERE c.dt >= '${DAY}-${HOUR}-00'
    > AND c.dt < '${NEXT_DAY}-${NEXT_HOUR}-20';
Query ID = hadoop_20231029014655_dc53b181-f4b9-4647-ad8c-4c9131c23adf
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1698540576797_0002)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 01/01  [==========================>>] 100%  ELAPSED TIME: 36.88 s
----------------------------------------------------------------------------------------------
Loading data to table default.tmp_clicks
OK
Time taken: 38.345 seconds
hive>
```

To create a left outer join of `tmp_clicks` and `tmp_impressions` that writes the resulting data set to the `joined_impressions` table in your S3 output bucket, running the following command:

Command:

```
INSERT OVERWRITE TABLE joined_impressions
PARTITION (day='${DAY}', hour='${HOUR}')
SELECT
i.requestBeginTime,
i.adId,
i.impressionId,
i.referrer,
i.userAgent,
i.userCookie,
i.ip,
(c.impressionId is not null) clicked
FROM tmp_impressions i
LEFT OUTER JOIN tmp_clicks c ON i.impressionId=c.impressionId;
```

Output:

```
hive> INSERT OVERWRITE TABLE joined_impressions
    > PARTITION (day='${DAY}', hour='${HOUR}')
    > SELECT
    > i.requestBeginTime,
    > i.adId,
    > i.impressionId,
    > i.referrer,
    > i.userAgent,
    > i.userCookie,
    > i.ip,
    > (c.impressionId is not null) clicked
    > FROM tmp_impressions i
    > LEFT OUTER JOIN tmp_clicks c ON i.impressionId=c.impressionId;
Query ID = hadoop_20231029014934_adb2c69a-65eb-4eec-84aa-c223c157fa07
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1698540576797_0002)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Map 2 .......... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 17.82 s
----------------------------------------------------------------------------------------------
Loading data to table default.joined_impressions partition (day=2009-04-13, hour=08)
OK
Time taken: 22.701 seconds
hive>
```

Task 6: Querying the resulting dataset

Using SQL-like queries to query the resulting dataset.

To instruct the HiveSQL client to print the names of the columns as headers in result sets, running the following command:

Command:

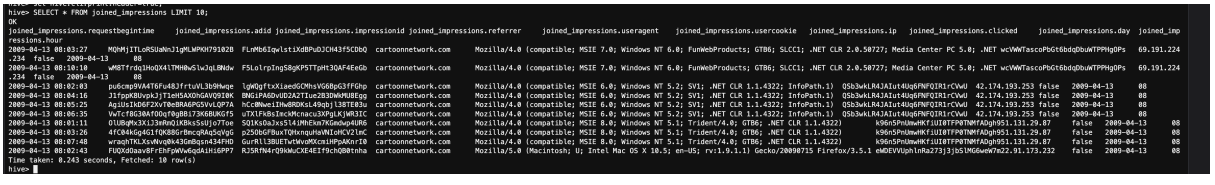
```
set hive.cli.print.header=true;
```

To return the first 10 rows of data in the `joined_impressions` table, running the following SELECT command:

Command:

```
SELECT * FROM joined_impressions LIMIT 10;
```

Output:



Query:

```
SELECT adid, count(*) AS hits FROM joined_impressions WHERE clicked = true GROUP BY adid ORDER BY hits DESC LIMIT 10;
```

Output:

```
hive> SELECT adid, count(*) AS hits FROM joined_impressions WHERE clicked = true GROUP BY adid ORDER BY hits DESC LIMIT 10;
Query ID = hadoop_20231029015516_71ce43e4-023c-4e70-a099-fba2153013e6
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1698540576797_0002)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 03/03  [==========================>>] 100%  ELAPSED TIME: 11.94 s
--------------------------------------------------------------------------------
OK
adid    hits
70n4fCvgAV0wfojgw5xw3QAEiaULMg  5
FxSntm0CqAx6LejjR2qVElsXofCXL7  4
2ikJc5AbFI6IA3LGrhATQGpdO7KhAk  4
7p32sreTnFKiK5glLSkCmexjXiXL8P  4
HdQw0xhN7o3hQUJIVnoSS4JumGN79w  4
tbXLLXhArCiqGMj3f1I0TrpGuidIKJ  3
98kBcfJejuljxpHDfBMxQlk8S41ql6  3
8NjKIm78sc6i9cciNxHLT3v6Ua44N2  3
vgAQhCFRswb4cFrwGJoaTUl1elqFQc  3
5CiFcHhj8mk2js8SERhCLofcUrIEdN  3
Time taken: 12.805 seconds, Fetched: 10 row(s)
hive>
```

Exiting the hive:

```
exit;
```

Query:

```
hive -e "SELECT referrer, count(*) as hits FROM joined_impressions WHERE clicked = true GROUP BY referrer ORDER BY hits DESC LIMIT 10;"
```

Output:

```
hive> exit;
[hadoop@ip-172-31-19-97 ~]$ hive -e "SELECT referrer, count(*) as hits FROM joined_impressions WHERE clicked = true GROUP BY referrer ORDER BY hits DESC LIMIT 10;" > /home/hadoop/result.txt

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.properties Async: true
Query ID = hadoop_20231029015756_252a089d-ffe2-47d8-9d52-e6bb962e8148
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1698540576797_0004)

Map 1: -/-      Reducer 2: 0/1  Reducer 3: 0/1
Map 1: 0/1      Reducer 2: 0/1  Reducer 3: 0/1
Map 1: 0/1      Reducer 2: 0/1  Reducer 3: 0/1
Map 1: 0(+1)/1  Reducer 2: 0/1  Reducer 3: 0/1
Map 1: 0(+1)/1  Reducer 2: 0/1  Reducer 3: 0/1
Map 1: 0(+1)/1  Reducer 2: 0/1  Reducer 3: 0/1
Map 1: 1/1      Reducer 2: 0/1  Reducer 3: 0/1
Map 1: 1/1      Reducer 2: 0(+1)/1      Reducer 3: 0/1
Map 1: 1/1      Reducer 2: 1/1  Reducer 3: 0/1
Map 1: 1/1      Reducer 2: 1/1  Reducer 3: 0(+1)/1
Map 1: 1/1      Reducer 2: 1/1  Reducer 3: 1/1
OK
Time taken: 34.456 seconds, Fetched: 10 row(s)
[hadoop@ip-172-31-19-97 ~]$
```

Displaying the content of the file `result.txt`:

```
cat /home/hadoop/result.txt
```

Output:

```
[hadoop@ip-172-31-19-97 ~]$ cat /home/hadoop/result.txt
ibibo.com       49
lowes.com       33
odnoklassniki.ru        33
files.wordpress.com     30
media-servers.net       29
tmz.com 29
tattoodle.com   29
tiscali.it      28
mpnrs.com       28
barnesandnoble.com      28
[hadoop@ip-172-31-19-97 ~]$
```

To exit the SSH session, running the following command:

```
exit
```

Analyzing one of the log files that makes up the output from Amazon S3:

To rediscover the name of the bucket that contains the *joined_impressions* table data, running the following command:

Command:

```
aws s3 ls /
```

Output:

```
voclabs:~/environment $ aws s3 ls /
2023-10-28 23:31:30 aws-logs-428444136833-us-east-1
voclabs:~/environment $
```

To download the file that contains the data that you have been querying, running the following command:

Command:

```
aws s3 cp s3://aws-logs-428444136833-us-east-1/output/tables/joined_impressions/day=2009-04-13/hour=08/000000_0 example-log.txt
```

Output:

```
voclabs:~/environment $ aws s3 cp s3://aws-logs-105193538101-us-east-1/output/tables/joined_impressions/day=2009-04-13/hour=08/000000_0 example-log.txt
download: s3://aws-logs-105193538101-us-east-1/output/tables/joined_impressions/day=2009-04-13/hour=08/000000_0 to ./example-log.txt
voclabs:~/environment $
```

Conclusion Remarks:

Launching an Amazon EMR cluster with Hive installed is a fundamental step in setting up a data processing environment. By leveraging the capabilities of EMR, you can efficiently handle large-scale data processing tasks and leverage Hive for data warehousing and analysis.

Establishing a secure connection to the Amazon EMR main node using an SSH client and an Amazon EC2 key pair private key ensures secure access to the cluster. This step is crucial for administering and monitoring the EMR cluster and carrying out various tasks efficiently.

Configuring a logging directory for Hive and initiating an interactive Hive session on the EMR cluster's main node enables seamless data querying and manipulation. This interactive session facilitates data exploration and analysis, providing insights critical for decision-making and further data processing steps.

Creating Hive tables that reference data stored in Amazon S3 forms the backbone of a well-structured data warehouse. Using HiveQL to load data and run queries efficiently organizes the data, laying the groundwork for subsequent analysis and extraction of valuable insights.

The process of joining the impressions and clicks tables provides comprehensive insights into user behavior and aids in optimizing advertising services. Leveraging partitioned tables enhances the efficiency of data retrieval and analysis, enabling more accurate and targeted decision-making for advertising strategies and campaign optimizations.

Completing these tasks demonstrates your proficiency in handling complex data processing and analysis workflows, crucial skills in the dynamic and evolving field of data science.