Big Data on AWS

Lab Two: Compare query performance in Athena

Table of Contents

[Lab Two: Compare query performance in Athena 3](#_Toc509605129)

[Task One: Create Table 3](#_Toc509605130)

[Task Two: Partitioning data 6](#_Toc509605131)

[Task Three: Convert data to Parquet format 9](#_Toc509605132)

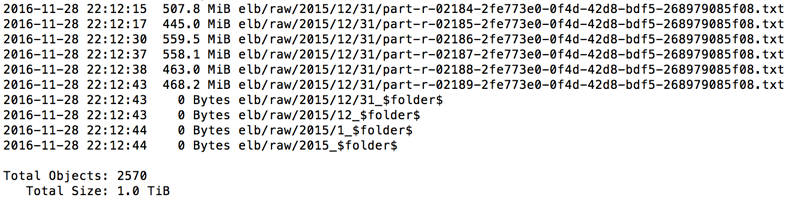
[Task Four: Comparing performance 10](#_Toc509605133)

# Lab Two: Compare query performance in Athena

In this exercise, you will use Athena on logs from Elastic Load Balancers. The logs are generated as text files in a pre-defined format. Through this lab, you will perform the following actions:

* Create a table,
* Partition the data in a format used by Athena,
* Convert it to Parquet format, and
* Compare query performance.

For this exercise, the raw logs are stored on Amazon S3 in the following format. There is a separate prefix for year, month, and date, with 2570 objects and 1 TB of data.



## Task One: Create Table

1. Open Athena and select ***Query Editor***

2. To create database, copy and paste the following DDL statement in the Athena query editor to create a table.

CREATE EXTERNAL TABLE IF NOT EXISTS elb\_logs\_raw\_native (

request\_timestamp string,

elb\_name string,

request\_ip string,

request\_port int,

backend\_ip string,

backend\_port int,

request\_processing\_time double,

backend\_processing\_time double,

client\_response\_time double,

elb\_response\_code string,

backend\_response\_code string,

received\_bytes bigint,

sent\_bytes bigint,

request\_verb string,

url string,

protocol string,

user\_agent string,

ssl\_cipher string,

ssl\_protocol string )

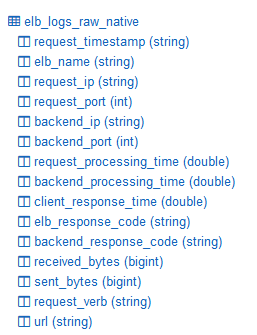
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'

WITH SERDEPROPERTIES (

'serialization.format' = '1','input.regex' = '([^ ]\*) ([^ ]\*) ([^ ]\*):([0-9]\*) ([^ ]\*)[:\-]([0-9]\*) ([-.0-9]\*) ([-.0-9]\*) ([-.0-9]\*) (|[-0-9]\*) (-|[-0-9]\*) ([-0-9]\*) ([-0-9]\*) \\\"([^ ]\*) ([^ ]\*) (- |[^ ]\*)\\\" (\"[^\"]\*\") ([A-Z0-9-]+) ([A-Za-z0-9.-]\*)$' )

LOCATION 's3://athena-examples/elb/raw/';

3. Expand the database to view all the fields



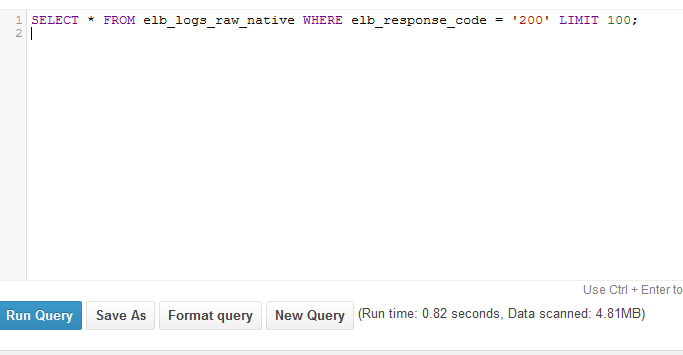
A table is created on the data stored in Amazon S3 and it is now ready to query the data.

Note that table elb\_logs\_raw\_native points towards the prefix s3://athena-examples/elb/raw/.

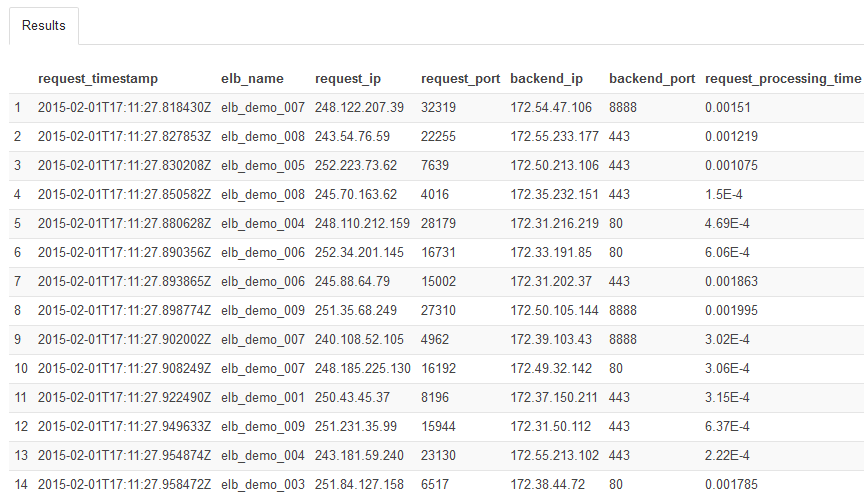
Therefore, when more data is added under the prefix, e.g., a new month’s data, the table automatically grows.

4. Run a simple query

SELECT \* FROM elb\_logs\_raw\_native WHERE elb\_response\_code = '200' LIMIT 100;



5. Now the table can be queried for all logs, without the need to set up any infrastructure or ETL.



## Task Two: Partitioning data

Data can be partitioned across multiple dimensions―e.g. month, week, day, hour, or customer ID―or all of them together.

6. Copy and paste the DDL statement in ***Athena Query Editor*** and select ***Run Query***.

To use partitions, first change the schema definition to include partitions, then load the partition metadata in Athena. Use the same CREATE TABLE statement but with partitioning enabled.

CREATE EXTERNAL TABLE IF NOT EXISTS elb\_logs\_raw\_native\_part (

request\_timestamp string,

elb\_name string,

request\_ip string,

request\_port int,

backend\_ip string,

backend\_port int,

request\_processing\_time double,

backend\_processing\_time double,

client\_response\_time double,

elb\_response\_code string,

backend\_response\_code string,

received\_bytes bigint,

sent\_bytes bigint,

request\_verb string,

url string,

protocol string,

user\_agent string,

ssl\_cipher string,

ssl\_protocol string )

PARTITIONED BY(year string, month string, day string)

ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'

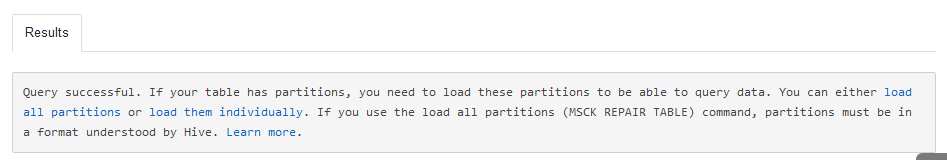
WITH SERDEPROPERTIES (

'serialization.format' = '1','input.regex' = '([^ ]\*) ([^ ]\*) ([^ ]\*):([0-9]\*) ([^ ]\*)[:\-]([0-9]\*) ([-.0-9]\*) ([-.0-9]\*) ([-.0-9]\*) (|[-0-9]\*) (-|[-0-9]\*) ([-0-9]\*) ([-0-9]\*) \\\"([^ ]\*) ([^ ]\*) (- |[^ ]\*)\\\" (\"[^\"]\*\") ([A-Z0-9-]+) ([A-Za-z0-9.-]\*)$' )

LOCATION 's3://athena-examples/elb/raw/';

Note the PARTITIONED BY clause in the CREATE TABLE statement. The data is partitioned by year, month, and day.

7. In the **Results** section, Athena posts a reminder to load partitions for a partitioned table.



8. Copy and paste the following DDL command in the ***Query Editor***.

show partitions elb\_logs\_raw\_native\_part

9. In the **Results** section, the data is partitioned by year, month, and day.

10. Copy and paste the following DDL statement to restrict each query by specifying the partitions in the WHERE clause. In this case, Athena scans less data and finishes faster.

SELECT distinct(elb\_response\_code),

count(url)

FROM elb\_logs\_raw\_native\_part

WHERE year='2015'

AND month= '01'

AND day='01'

GROUP BY elb\_response\_code

11. In the **Results** section, observe the following:

## Task Three: Convert data to Parquet format

12. Copy and paste the following DDL statement to create a table on the parquet data set.

CREATE EXTERNAL TABLE IF NOT EXISTS elb\_logs\_pq (

request\_timestamp string,

elb\_name string,

request\_ip string,

request\_port int,

backend\_ip string,

backend\_port int,

request\_processing\_time double,

backend\_processing\_time double,

client\_response\_time double,

elb\_response\_code string,

backend\_response\_code string,

received\_bytes bigint,

sent\_bytes bigint,

request\_verb string,

url string,

protocol string,

user\_agent string,

ssl\_cipher string,

ssl\_protocol string )

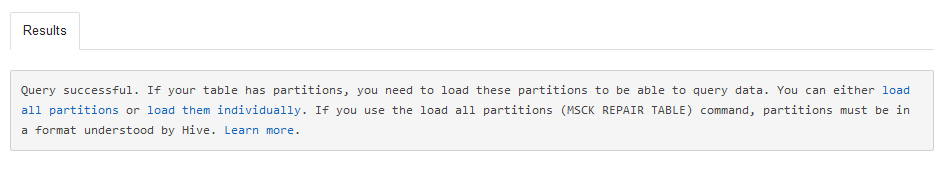
PARTITIONED BY(year int, month int, day int)

STORED AS PARQUET

LOCATION 's3://athena-examples/elb/parquet/'

tblproperties ("parquet.compress"="SNAPPY");

13: Observe the ***Results.***



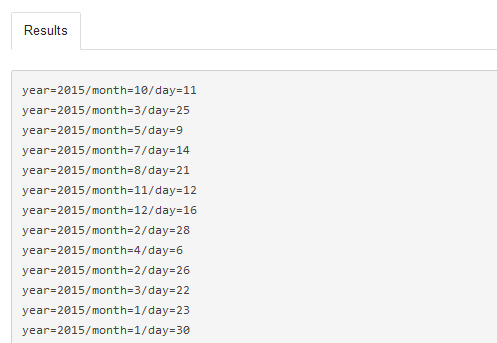
14: Copy and paste the following DDL statement to allow the catalog to recognize all partitions.

msck repair table elb\_logs\_pq

15: Copy and paste the following DDL statement to list all the partitions.

show partitions elb\_logs\_pq

16: Note the list of partitions.



## Task Four: Comparing performance

Compare the performance of the same query between text files and Parquet files.

17: Copy and paste the following DDL statement to run query on compressed, partitioned, and columnar data.

SELECT elb\_name,

uptime,

downtime,

cast(downtime as DOUBLE)/cast(uptime as DOUBLE) uptime\_downtime\_ratio

FROM

(SELECT elb\_name,

sum(case elb\_response\_code

WHEN '200' THEN

1

ELSE 0 end) AS uptime, sum(case elb\_response\_code

WHEN '404' THEN

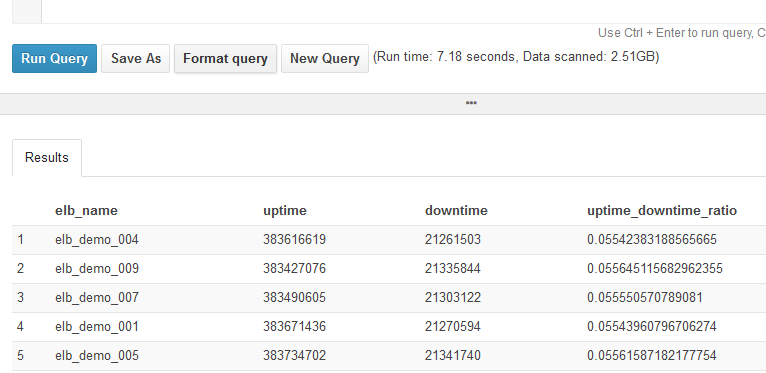
1

ELSE 0 end) AS downtime

FROM elb\_logs\_pq

GROUP BY elb\_name)

18: Note the ***Results,*** Runtime and Data Scanned.



19: Copy and paste the following DDL statement to run query on raw data.

SELECT elb\_name,

uptime,

downtime,

cast(downtime as DOUBLE)/cast(uptime as DOUBLE) uptime\_downtime\_ratio

FROM

(SELECT elb\_name,

sum(case elb\_response\_code

WHEN '200' THEN

1

ELSE 0 end) AS uptime, sum(case elb\_response\_code

WHEN '404' THEN

1

ELSE 0 end) AS downtime

FROM elb\_logs\_raw\_native

GROUP BY elb\_name)

ELSE 0 end) AS downtime

FROM elb\_logs\_raw\_native

GROUP BY elb\_name)

20: Note the ***Results,*** Runtime and Data Scanned.

