Building Batch Data Analytics Solutions on AWS:

Lab 1 - Low-Latency Data Analytics Using Apache Spark on Amazon EMR

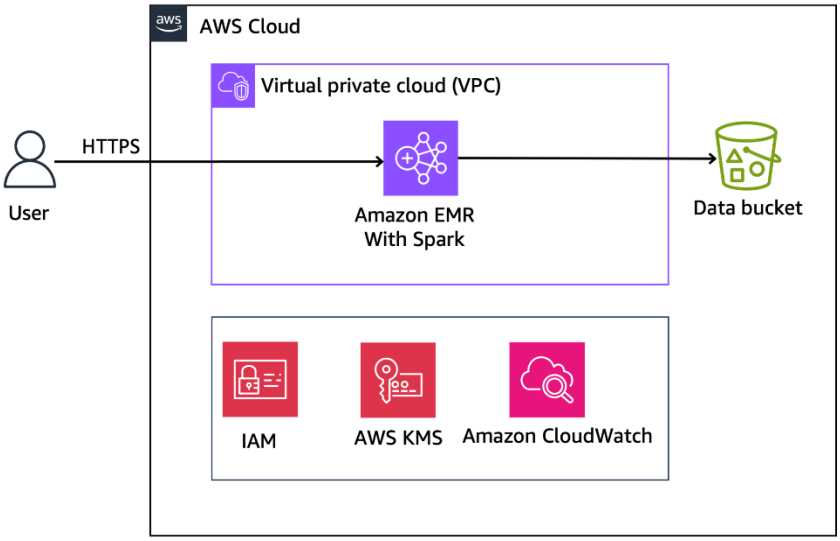
Note: Do not include any personal, identifying, or confidential information into the lab environment. Information entered may be visible to others.

Corrections, feedback, or other questions? Contact us at *AWS Training and Certification*.

Lab overview

Your company, AnyCompany Financials, is creating a platform for users to perform customized stock queries. You are asked to investigate low-latency data analytics for the dynamic stock application. You've decided to use Apache Spark to process the data and investigate different monitoring methods.

Your task is to connect to an EMR cluster and create an application using PySpark, and research the operational visibility of your stock application using the Spark history server.



OBJECTIVES

By the end of this lab, you will be able to:

Review an EMR cluster with Apache Spark

Connect to an EMR cluster

Use PySpark to interact with an EMR cluster

Access a Spark history server and review different Spark job metrics

AWS SERVICES NOT USED IN THIS LAB

AWS service capabilities used in this lab are limited to what the lab requires. Expect errors when accessing other services or performing actions beyond those provided in this lab guide.

ICON KEY

Various icons are used throughout this lab to call attention to certain aspects of the guide. The following list explains the purpose for each one:

Specifies the command you must run.

Verify the output of a command or edited file.

Specifies important hints, tips, guidance, or advice.

Calls attention to information of special interest or importance. Failure to read the note does not result in physical harm to the equipment or data, but it could result in the need to repeat certain steps.

Start lab

To launch the lab, at the top of the page, choose Start lab.

Caution: You must wait for the provisioned AWS services to be ready before you can continue.

To open the lab, choose Open Console.

You are automatically signed in to the AWS Management Console in a new web browser tab.

WARNING: Do not change the Region unless instructed.

COMMON SIGN-IN ERRORS

Error: You must first sign out

If you see the message, **You must first log out before logging into a different AWS account:**

Choose the **click here** link.

Close your **Amazon Web Services Sign In** web browser tab and return to your initial lab page.

Choose Open Console again.

Error: Choosing Start Lab has no effect

In some cases, certain pop-up or script blocker web browser extensions might prevent the **Start Lab** button from working as intended. If you experience an issue starting the lab:

Add the lab domain name to your pop-up or script blocker's allow list or turn it off.

Refresh the page and try again.

Task 1: Explore the lab environment

In this task, you review the account resources created when the lab was started.

REVIEW YOUR FOLDERS IN THE AMAZON S3 BUCKET

At the top of the page, in the unified search bar, search for and choose

S3

Select the bucket with **databucket** in its name.

Select the name of the **data/** folder. You will see a **stock_prices.csv** file.

This file contains the information of stock prices of some of the big tech companies (AAPL, SQ, AMZN, GE, M, TSLA, and MSFT) for the year 2020. Data columns you can find include **Trade_Date**, **Ticker**, **High**, **Low**, **Open**, **Close**, **Volume**, and **Adj_Close**.

Trade_Date	Ticker	High	Low	Open	Close	Volume	Adj_Close
2020-01-02	aapl	75.1500015258789	73.79750061035156	74.05999755859375	75.0875015258789	135480400.0	74.20746612548828
2020-01-02	sq	64.05000305175781	62.95000076293945	62.99000167 84668	63.83000183105469	5264700	63.83000183105469
2020-01-02	amzn	1898.010009765625	1864.1500244140625	1875.0	1898.010009765625	4029000	1898.010009765625
2020-01-02	ge	11.960000038146973	11.229999542236328	11.229999542236328	11.930000305175781	87421800.0	11.861019134521484
2020-01-02	m	17.270000457763672	16.389999389648438	17.180000305175 78	16.520000457763672	26388100.0	15.861986160 27832
2020-01-02	tsla	86.13999938964844	84.34200286865234	84.9000015258789	86.052001953125	47660500.0	86.052001953125
2020-01-02	msft	160.72999572753906	158.33000183105 47	158.77999877929688	160.61999511718 75	22622100.0	158.20576477050 78

REVIEW YOUR EMR CLUSTER CONFIGURATION

At the top of the page, in the unified search bar, search for and choose

EMR

In the left navigation pane, in the **EMR on EC2** section, choose **Clusters**.

The **labcluster** is in a **Waiting** status. This means that the cluster has started and is ready to use.

Select **labcluster** to view more details.

A summary page of the EMR cluster is presented. In this instance, the cluster is preloaded with Spark, JupyterEnterpriseGateway, Flink, and Livy, applications.

Livy enables interaction over a REST interface with an EMR cluster running Spark.

Flink is a streaming dataflow engine that runs real-time stream processing on high-throughput data sources.

Task complete: You have successfully explored the lab environment.

Task 2: Connect to an EMR cluster

In this task, you connect to the EMR cluster using SSH into CommandHost Session Manager and interact with cluster using PySpark.

Copy the **CommandHostSessionManagementUrl** from the left of this instruction and paste it on a new tab of your browser.

Expected output: You will be redirected to the command host terminal.

Command: In the terminal, copy and paste below commands:

Note: Using the below command, you are retrieving the EMR cluster ID and then use that Cluster ID to get the public DNS of the cluster. Finally you are using the Cluster DNS value to SSH to the EMR Cluster terminal.

```
# Get EMR Cluster ID and export to the Environment.

export ID=$(aws emr list-clusters | jq '.Clusters[0].Id' | tr -d "")


# Use the ID to get the PublicDNS name of the EMR Cluster

# and export to the Environment.

export HOST=$(aws emr describe-cluster --cluster-id $ID | jq
'.Cluster.MasterPublicDnsName' | tr -d "")
```

```
# SSH to the EMR cluster
ssh -i ~/EMRKey.pem hadoop@$HOST
```

For the prompt, enter

```
yes
```

.

Expected output: You will see the EMR terminal.

```
************************
**** EXAMPLE OUTPUT ****
************************

Warning: Permanently added 'ec2-35-160-218-246.us-west-2.compute.amazonaws.com,10.1.12.14' (ECDSA) to the list of known hosts.
Last login: Tue Jul 11 17:08:58 2023

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
No packages needed for security; 1 packages available
Run "sudo yum update" to apply all updates.

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 MMMMMMMM MMMMMMMM RRRRRRR RRRRRR

INTERACT WITH THE EMR CLUSTER USING PYSPARK

Command: Type

`pyspark` in the terminal and press *enter*.

Expected output: You will be redirected to the Spark application.

****

**** EXAMPLE OUTPUT ****

****

[hadoop@ip-10-0-10-164 ~]$ pyspark

...

Welcome to

```
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.1.1-amzn-0
      /_/
```

Using Python version 3.7.16 (default, Mar 10 2023 03:25:26)

Spark context Web UI available at http://ip-10-0-10-164.us-west-2.compute.internal:4040

Spark context available as 'sc' (master = yarn, app id = application_1690557266848_0001).

SparkSession available as 'spark'.

>>>

To import the required modules, paste the following command and press *enter*.

```
import sys
import time
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *
```

Expected output:

None, unless there is an error.

To create an application with name **stock-summary**, paste the following command and press *enter*.

```
spark =  SparkSession.builder.appName("stock-summary").getOrCreate()
```

Expected output:

None, unless there is an error.

To create a dataBucket variable that points to the Amazon S3 bucket containing the data file, paste the following command and press *enter*.

This action simplifies future commands that also need to point to the bucket.

Replace **DATA_BUCKET** with the **DataBucket** value shown to the left of these instructions. Make sure to leave the single quotes.

```
dataBucket = 'DATA_BUCKET'
```

Expected output:

None, unless there is an error.

To read the **stockprice.csv** file from Amazon S3 and import it to the cluster as a DataFrame, paste the following command and press enter.

```
df = spark.read.csv("s3://"+dataBucket+"/data/stock_prices.csv", header=True, inferSchema=True).select('Trade_Date', 'Ticker', 'Close', 'Volume')

df.sort(df.Trade_Date, ascending=True).show(7)
```

Expected output:

```
*************************

**** EXAMPLE OUTPUT ****

*************************
```

```
+----------+------+-----------------+----------+
|Trade_Date|Ticker|            Close|    Volume|
+----------+------+-----------------+----------+
|2020-01-02|    sq| 63.83000183105469| 5264700.0|
|2020-01-02|  aapl|  75.0875015258789|1.354804E8|
|2020-01-02|  amzn| 1898.010009765625| 4029000.0|
|2020-01-02|     m|16.520000457763672| 2.63881E7|
|2020-01-02|  tsla|   86.052001953125| 4.76605E7|
|2020-01-02|  msft| 160.6199951171875| 2.26221E7|
|2020-01-02|    ge|11.930000305175781| 8.74218E7|
+----------+------+-----------------+----------+
only showing top 7 rows
```

To find the total number of records, paste the following command and press *enter*.

```
("Total number of stocks records: " + str(df.count()))
```

Expected output:

```
************************
**** EXAMPLE OUTPUT ****
************************

'Total number of stocks records: 1771'
```

To find dates when transaction volume was more than 1 million stocks, and display 10 results in descending order, paste the following command and press *enter*.

```
dfVol = df.filter( (df.Volume > 10000000)).sort(df.Volume, ascending=False)
dfVol.show(10)
```

Expected output:

```
************************
**** EXAMPLE OUTPUT ****
************************
```

```
+----------+------+-----------------+----------+
|Trade_Date|Ticker|            Close|    Volume|
+----------+------+-----------------+----------+
|2020-02-28|  aapl| 68.33999633789062|  4.2651E8|
|2020-03-12|  aapl|62.057498931884766| 4.18474E8|
|2020-03-20|  aapl|57.310001373291016|4.016932E8|
|2020-07-31|  aapl|106.26000213623047|3.743368E8|
|2020-03-13|  aapl| 69.49250030517578| 3.70732E8|
|2020-08-24|  aapl|125.85749816894531|3.459376E8|
|2020-03-02|  aapl| 74.70249938964844|3.413972E8|
|2020-08-21|  aapl|124.37000274658203|3.380548E8|
|2020-03-23|  aapl|56.092498779296875|3.367528E8|
|2020-09-04|  aapl|120.95999908447266|3.326072E8|
+----------+------+-----------------+----------+
only showing top 10 rows
```

To create a `stockprice` query view, paste the following command and press *enter*.

Using local temporary views, you can use SQL syntax to query your data.

```
df.createOrReplaceTempView("stockprice")
```

Expected output:

None, unless there is an error.

The lifetime of this temporary table is tied to the SparkSession that is used to create this DataFrame.

To list `10` results where the transaction `Volume` was more than `10000000` stocks, paste the following command and press *enter*.

```
dfSql = spark.sql("SELECT Trade_Date, Ticker, round(DOUBLE(Close),2) AS Closing_Value, Volume  FROM stockprice WHERE Volume > 10000000 ORDER BY Close DESC LIMIT 10")
```

```
dfSql.sort(dfSql.Volume, ascending=False).show()
```

Expected output:

```
************************
**** EXAMPLE OUTPUT ****
************************
+----------+------+------------+----------+
|Trade_Date|Ticker|Closing_Value|    Volume|
+----------+------+------------+----------+
|2020-12-18|  tsla|       695.0|2.221262E8|
|2020-12-31|  tsla|      705.67| 4.96499E7|
|2020-12-30|  tsla|      694.78|  4.2846E7|
|2020-12-28|  tsla|      663.69| 3.22786E7|
|2020-12-29|  tsla|      665.99| 2.29108E7|
|2020-01-31|  amzn|     2008.72| 1.55673E7|
|2020-04-16|  amzn|     2408.19| 1.20382E7|
|2020-03-12|  amzn|     1676.61| 1.13462E7|
|2020-03-17|  amzn|     1807.84| 1.09171E7|
|2020-03-19|  amzn|     1880.93| 1.03999E7|
+----------+------+------------+----------+
```

Task complete: You successfully connected to the EMR cluster using SSH tool, accessed the EMR cluster, and switched to Spark application to run PySpark examples.

Task 3: Challenge – analyze movie data with Spark

We have uploaded movie data to the **ChallengeBucket**. Your task is to find the number of movies that actor

Jodie Fost is associated with as an actor. List the

year,

title, and

rating in chronological ascending order (

year low to high).

Navigate **here** for solution.

Hint: The column names are:

year

title

directors_0

rating

genres_0

genres_1

rank

running_time_secs

actors_0

actors_1

actors_2

directors_1

directors_2

Task complete: You successfully analyzed movie data with Spark.

Task 4: Review Spark jobs in the Spark history server

In this task, you access the Spark history server and review job metrics that ran previously.

The Spark history server is an extension of the Apache Spark web user interface (UI). It presents a visual interface with detailed information about completed and running Spark jobs on a cluster. You can dive into job-specific metrics and information about scheduler stages, tasks, and running executors.

At the top of the page, in the unified search bar, search for and choose

EMR

In the left navigation pane, in the **EMR on EC2** section, choose **Clusters**.

Select **labcluster** to view more details.

Under the Summary pane, choose **Applications**.

In the **Application user interfaces** section:

Select **Persistent application UIs**

Choose the **Spark History Server** link.

You are redirected to the **Spark History Server** page.

If you see the **Popup Blocked** banner, please follow your browser configuration to allow pop-ups.

If you see the **No completed applications found!** banner, choose the **Show incomplete applications** link.

Select any **App ID**.

You are directed to a **Spark Jobs** page. Here you can see a snapshot of the job.

Expand **Event Timeline** to see the various stages of the run.

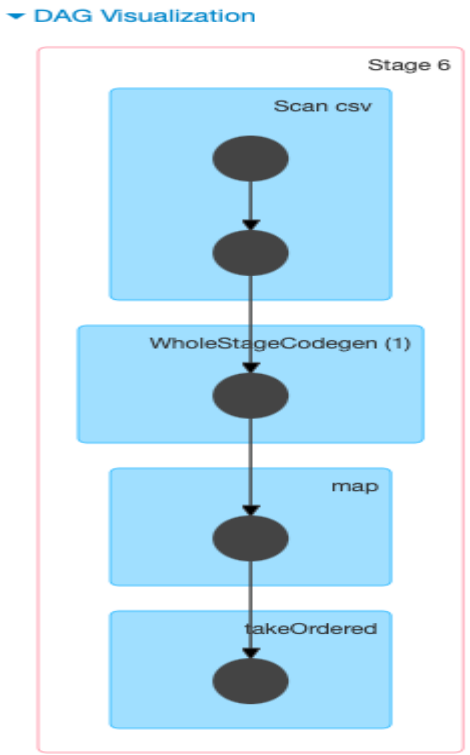
You can also view the **Completed Jobs** status in tabular format.

Sort the **Job Id (Job Group)** column in ascending order, and then choose the highlighted link description of the last job.

You are directed to the **Details for Job** page. On this page, you see **DAG Visualization** (DAG stands for directed acyclic graph).

Expand **DAG Visualization** to see the tasks that were just run.

You can use the DAG to view the query plan and see the logic Spark is using to run each task. Here is a sample of what a DAG looks like:



By using a directed acyclic graph (DAG) execution engine, Spark can create efficient query plans for data transformations.

Task complete: You successfully reviewed Spark jobs in the Spark history server.

Conclusion

Congratulations! You now have successfully:

Reviewed an EMR cluster with Apache Spark

Connected to an EMR cluster

Used PySpark to interact with an EMR cluster

Accessed a Spark history server and reviewed different Spark job metrics

End lab

Follow these steps to close the console and end your lab.

Return to the **AWS Management Console**.

At the upper-right corner of the page, choose **AWSLabsUser**, and then choose **Sign out**.

Choose End lab and then confirm that you want to end your lab.

For more information about AWS Training and Certification,
see *https://aws.amazon.com/training/*.

Your feedback is welcome and appreciated.
If you would like to share any feedback, suggestions, or corrections, please provide the details in our *AWS Training and Certification Contact Form*.

Appendix

TASK 3 CHALLENGE SOLUTION

Replace **CHALLENGE_BUCKET** with the **ChallengeBucket** value shown to the left of these instructions.

```
spark =  SparkSession.builder.appName("movie-summary").getOrCreate()

df_challenge = spark.read.csv("s3://CHALLENGE_BUCKET/data/movies.csv",
header=True,
inferSchema=True).select('year','title','directors_0','rating','actors_0','actors_1','actors_2')
```

The number of movies that actor

Jodie Fost is associated with:

```
dfJodieFoster = df_challenge.filter( (df_challenge.actors_0 == "Jodie Foster") | (df_challenge.actors_1 == "Jodie Foster") | (df_challenge.actors_2 == "Jodie Foster") ).sort(df_challenge.year, ascending=True)

rows = dfJodieFoster.count()

dfJodieFoster.show()

print(f"Total number of movies : {rows}")
```

Additionally, for those that who want to solve the challenge using Spark SQL, refer to below solution:

```
spark =  SparkSession.builder.appName("movies").getOrCreate()

dataBucket = 'CHALLENGE_BUCKET'

dfmovies = spark.read.csv("s3://"+dataBucket+"/data/movies.csv", header=True, inferSchema=True).select('year', 'title', 'rating', 'actors_0','actors_1','actors_2')

dfmovies.show(10) ##Not necessary but proves data was loaded in to the data frame

dfmovies.createOrReplaceTempView("movies_view")

dfmovies = spark.sql("SELECT year, title, rating, actors_0, actors_1, actors_2 FROM movies_view WHERE actors_0 = 'Jodie Foster' OR actors_1 = 'Jodie Foster' OR actors_2 = 'Jodie Foster' ORDER BY year ASC")

dfmovies.show()

dfmovies.count()
```

To continue this lab, move on to Task 4.