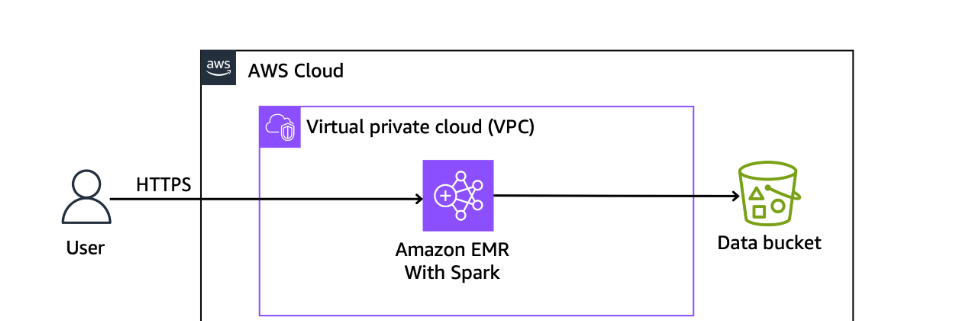
# Lab 1 - Low-Latency Data Analytics Using Apache Spark on Amazon EMR

Apache Spark, a powerful tool for processing big data

Amazon EMR (Elastic MapReduce) is a managed service from AWS for processing big data using frameworks like Apache Spark.



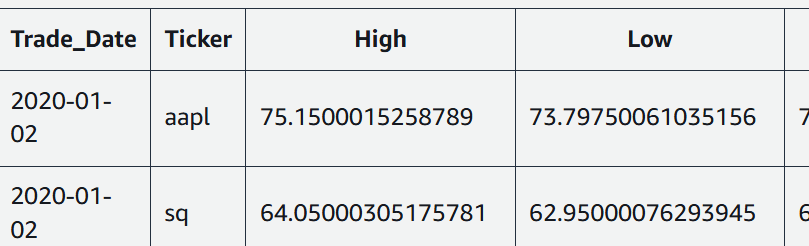
1. EMR Cluster: Think of this as a group of computers working together to process large amounts of data. EMR makes it easy to set up and manage these clusters.
2. Apache Spark: This is the tool you'll use to process data. It's fast and designed to handle big data tasks efficiently.

Spark runs on EMR cluster instances

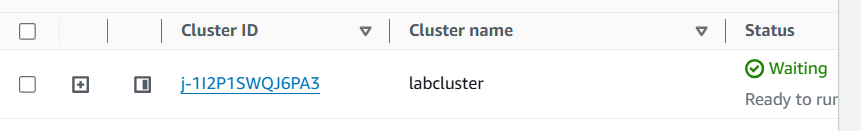
PySpark: This is a way to use Spark with Python.

**Task 1: Explore the lab environment**

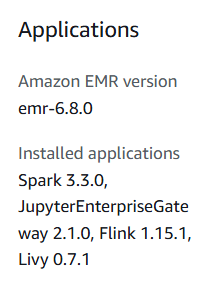
1.1 Open S3, visit data

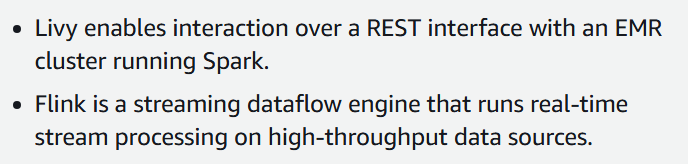


1.2 emr cluster



1.3 spark is preloaded





**Task 2: Connect to an EMR cluster**

2.1 open CLI using link

2.2 run

*# 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*

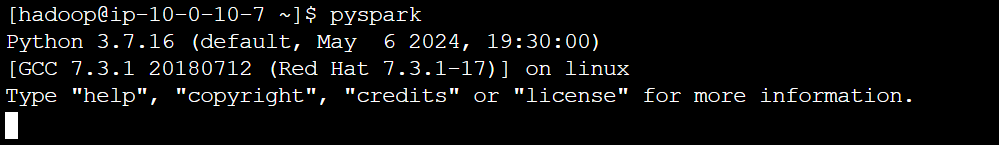
1. Retrieves the ID of the first EMR cluster.

2. Fetches the Master Public DNS name of the EMR cluster using its ID.

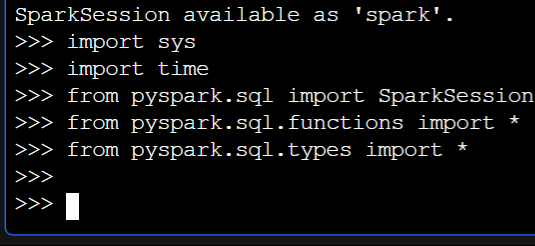
3. Initiates an SSH connection to the EMR cluster using the fetched DNS name and specified credentials.

2.2 after connected, run this to interact using pyspark

Pyspark

2.3 

2.4 run commands



2.5 To create an application with name stock-summary, paste the following command

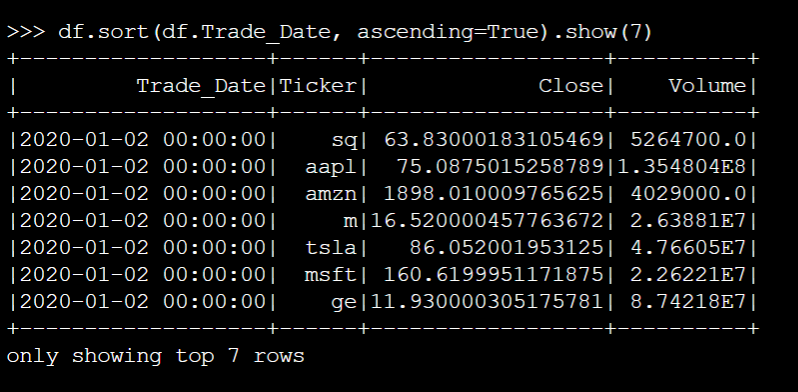


2.6 give the data location

dataBucket = ‘S3\_loc’

2.7 read the csv file





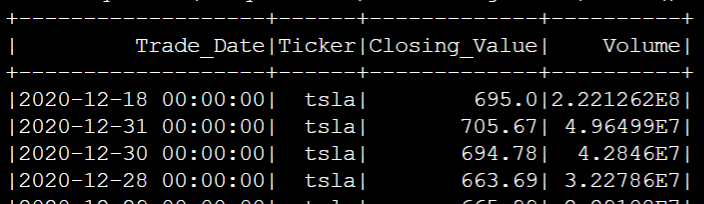
2.8 this command Creates a temporary view named "stockprice" in Spark DataFrame df, allowing SQL queries to be performed on it.

df.createOrReplaceTempView("stockprice")

2.9 we can now run sql queries

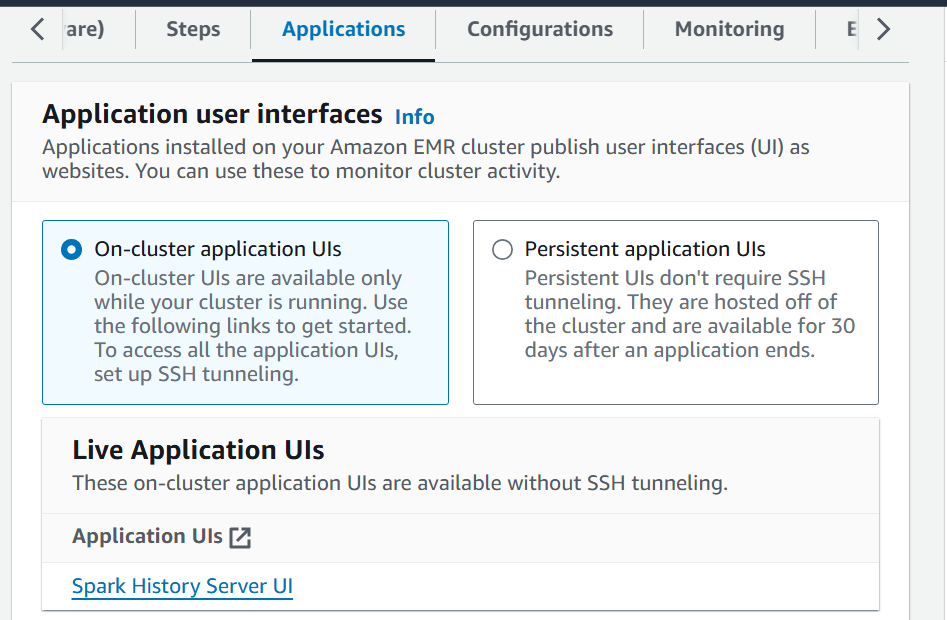
dfSql = spark.sql("SELECT Trade\_Date, Ticker, round(DOUBLE(Close),2) AS Closing\_Value, Volume FROM stockprice WHERE Volume > 10000000 ORDER BY Close DESC LIMIT 10")

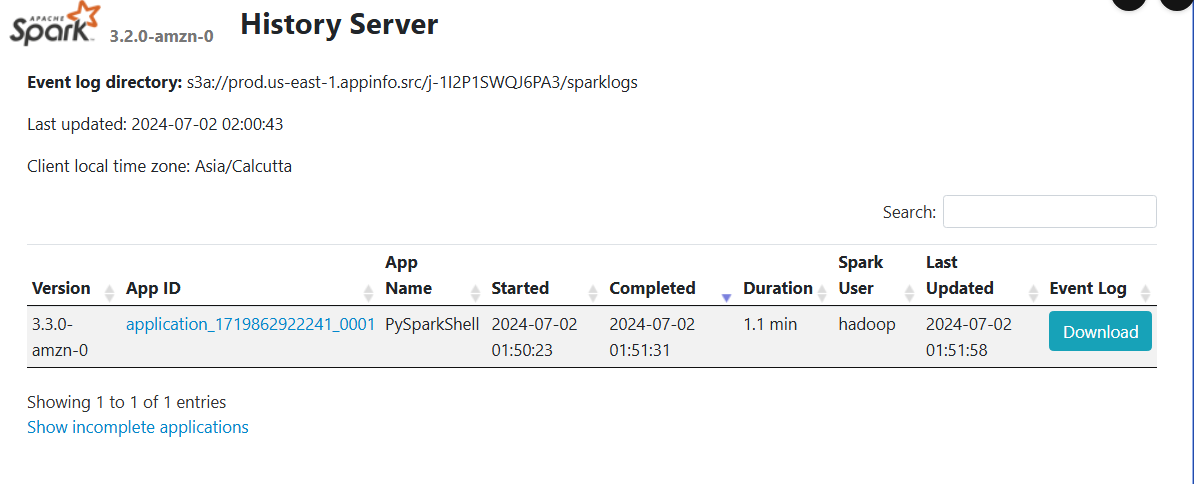
dfSql.sort(dfSql.Volume, ascending=False).show()

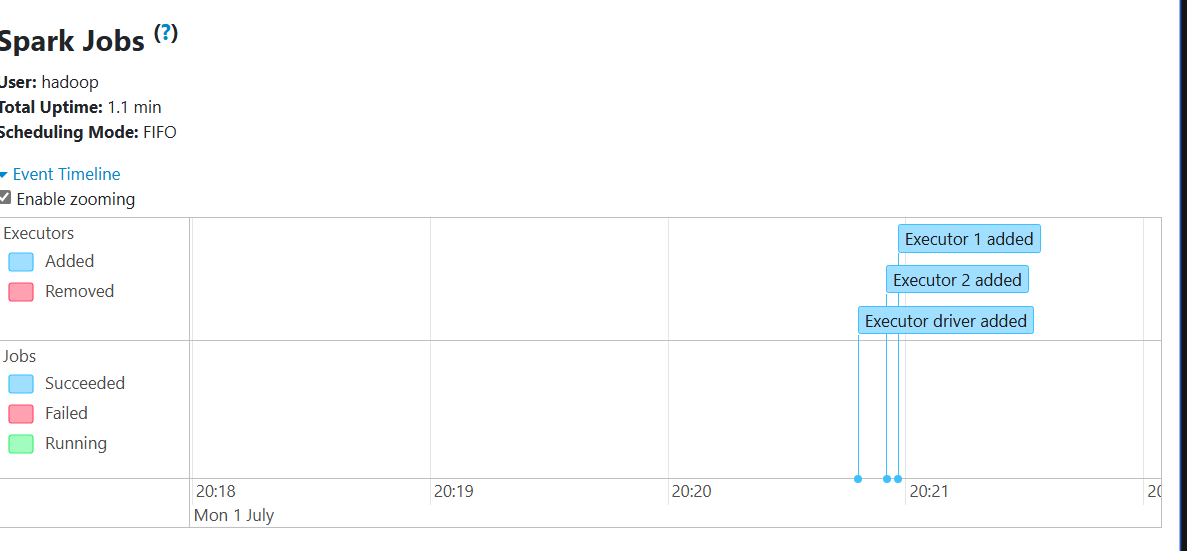


**Task 3: Review Spark jobs in the Spark history server**

3.1 go here and click the link



3.2 



An executor in Spark is a distributed agent responsible for executing tasks on worker nodes within a Spark cluster.