

House Rules

- Welcome! Thank you for joining us for **Day 2 of Big Data Processing on AWS Elastic MapReduce (EMR) using Hadoop and Spark.**
- Please feel free to pop your questions in the Q&A and our Engineers will answer them as we go along.
- Our Presenter will also answer some of the questions at different points.
- Please check your emails from Nikki for your AWS Account (Hash Link). Please check your Junk-Mail or Spam.

Welcome to Day 2

- Day 2 Instructors



Mahlatse



Odwa



Emmanuel

An introduction to Big Data processing using Spark Applications

Mahlatse Baholo
Cloud Support Engineer
AWS



Contents

What is Apache Spark?

How does it work?

How to monitor Spark jobs?

Spark Lab demonstration

Key Terms

- ❑ Application - A self-contained computation that runs user-supplied code to compute a result
- ❑ SparkSession - A unified entry point of a spark application from Spark 2.0
- ❑ Cluster manager - An external service for acquiring resources on the cluster
- ❑ Worker node - Any node that can run application code in the cluster
- ❑ Executor - A process launched for an application on a worker node, that runs tasks and keeps data in memory or disk storage across them
- ❑ Task - A unit of work that will be sent to one executor
- ❑ Job - A parallel computation consisting of multiple tasks that gets spawned in response to a Spark action (e.g. save, collect)
- ❑ Stage - Each job gets divided into smaller sets of tasks called stages that depend on each other (similar to the map and reduce stages in MapReduce)
- ❑ REPL (Read–Eval–Print–Loop) – Also termed an interactive top level or language shell, is a simple interactive computer programming environment that takes single user inputs, executes them, and returns the result to the user

Spark Overview

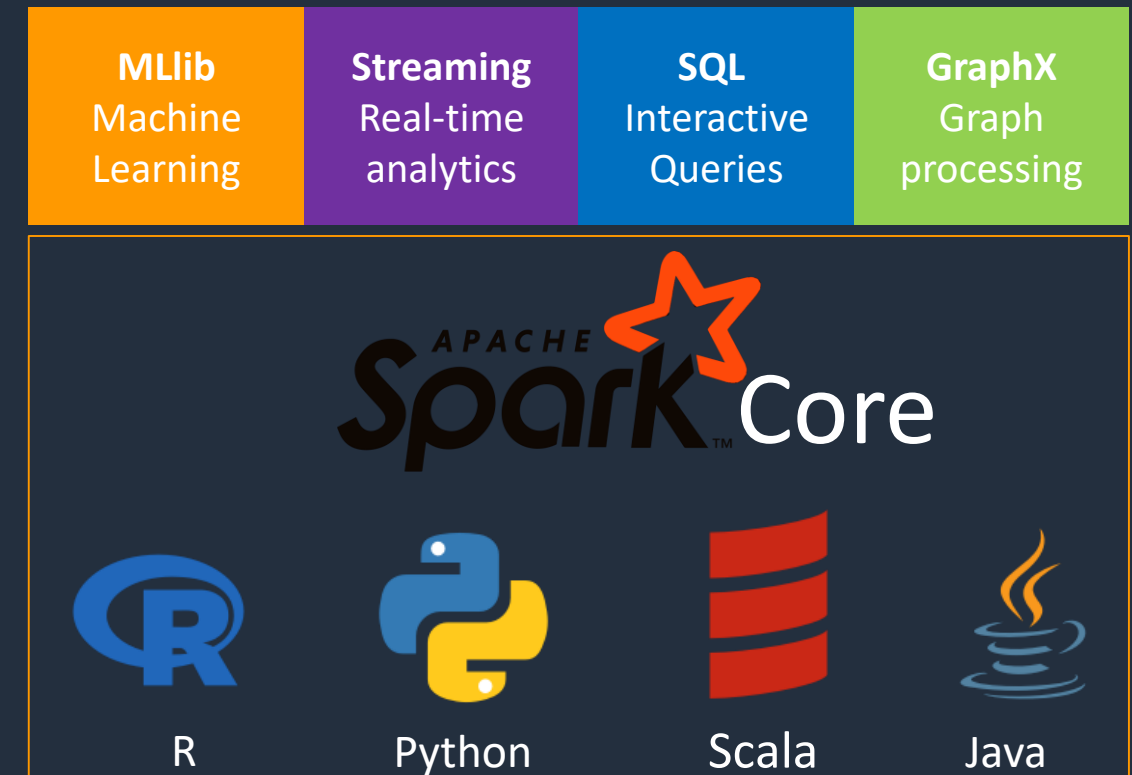
What is Apache Spark ?

Apache Spark is an open-source, distributed processing system used for big data workloads

The Spark framework includes:

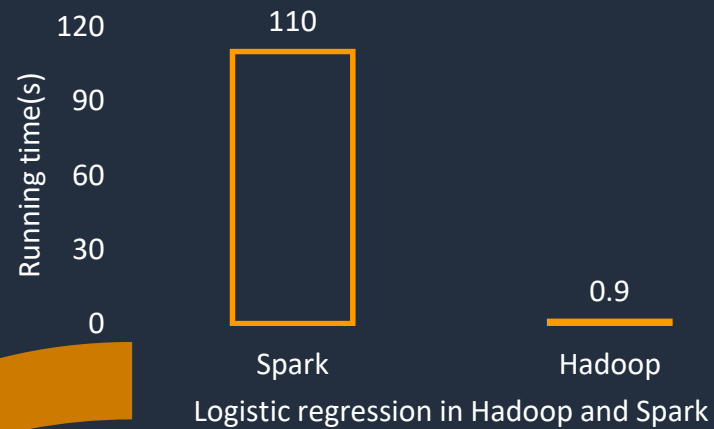
- ❑ Spark Core as the foundation for the platform which provides high-level APIs in R, Python, Scala and Java
- ❑ Spark MLlib for machine learning
- ❑ Spark Streaming for real-time analytics
- ❑ Spark SQL for interactive queries
- ❑ Spark GraphX for graph processing

Spark Components



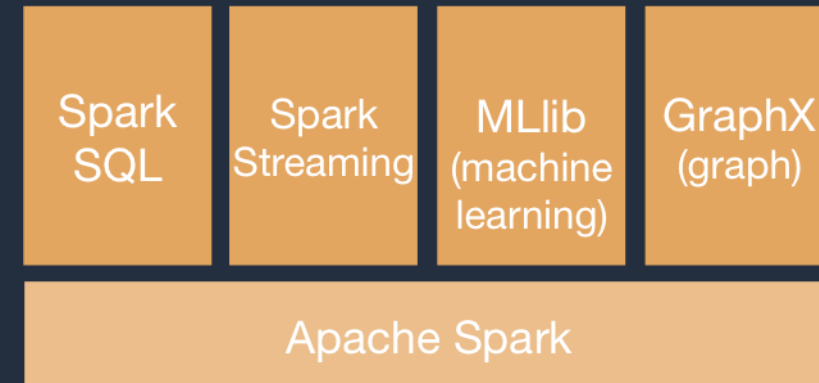
Benefits of Spark

Speed - Run workloads 100x faster



It is fast because it utilizes in-memory processing and caching and optimized query execution for fast analytical queries

Multiple workloads



Ease of Use –
Write applications quickly in



R



Python



Scala



Java

Spark Deployment options



Apache YARN is the cluster resource manager of Hadoop 2



Apache Mesos is an open project to manage computer clusters and can also run Hadoop applications



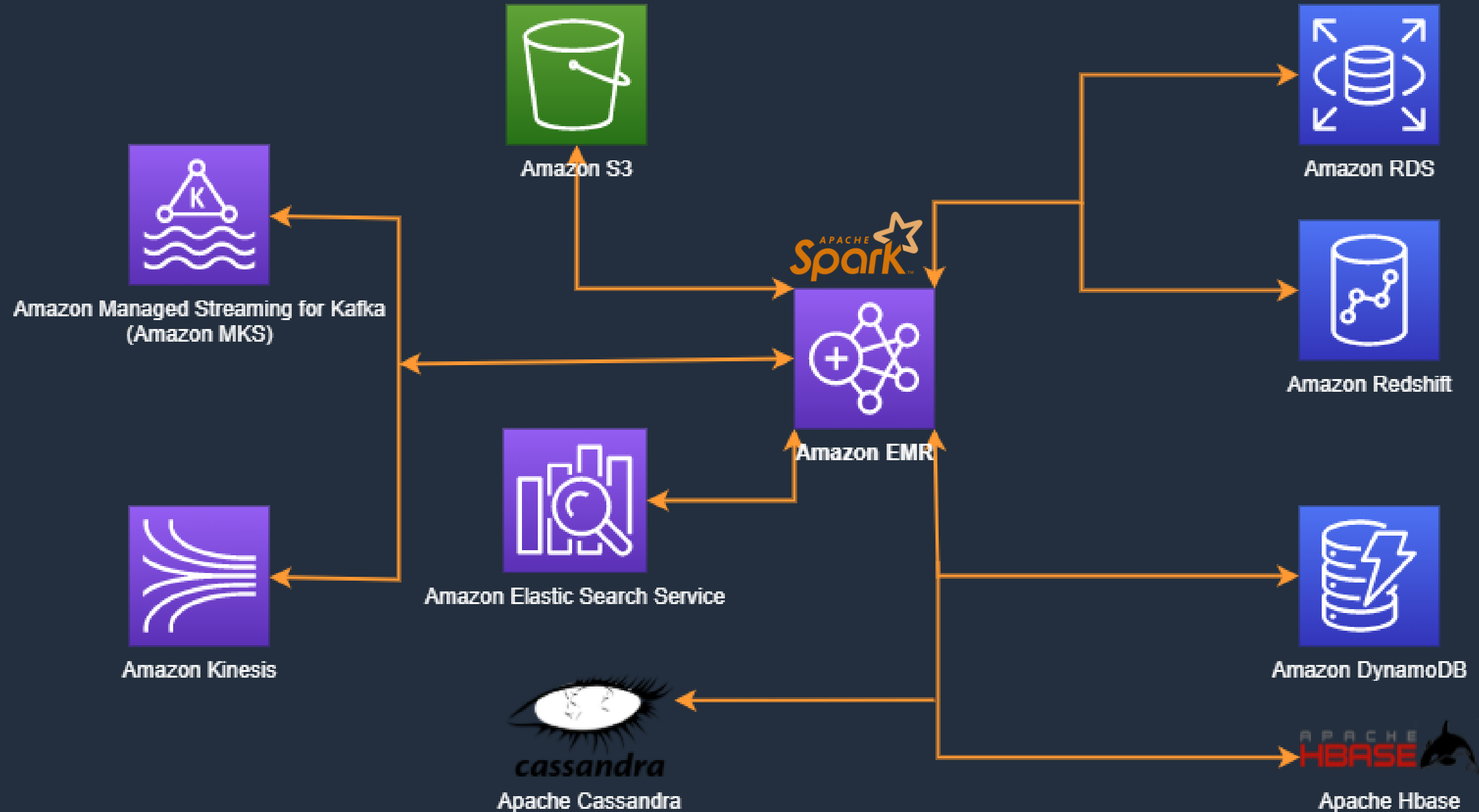
Open source system for automating deployment, scaling, and management of containerized applications



Standalone mode

By default, applications submitted to the standalone cluster will run in FIFO order, each application will try to use all available nodes

Many Storage Layers to choose from



Apache Spark Use Cases



Financial Services

Spark is used in banking to predict customer churn, and recommend new financial products

In investment banking, Spark is used to analyze stock prices to predict future trends



Manufacturing

Spark is used to eliminate downtime of internet-connected equipment, by recommending when to do preventive maintenance



Healthcare

Spark is used to build comprehensive patient care, by making data available to front-line health workers

Spark can also be used to predict/recommend patient treatment



Retail

Spark is used to attract, and keep customers through personalized services and offers

Poll

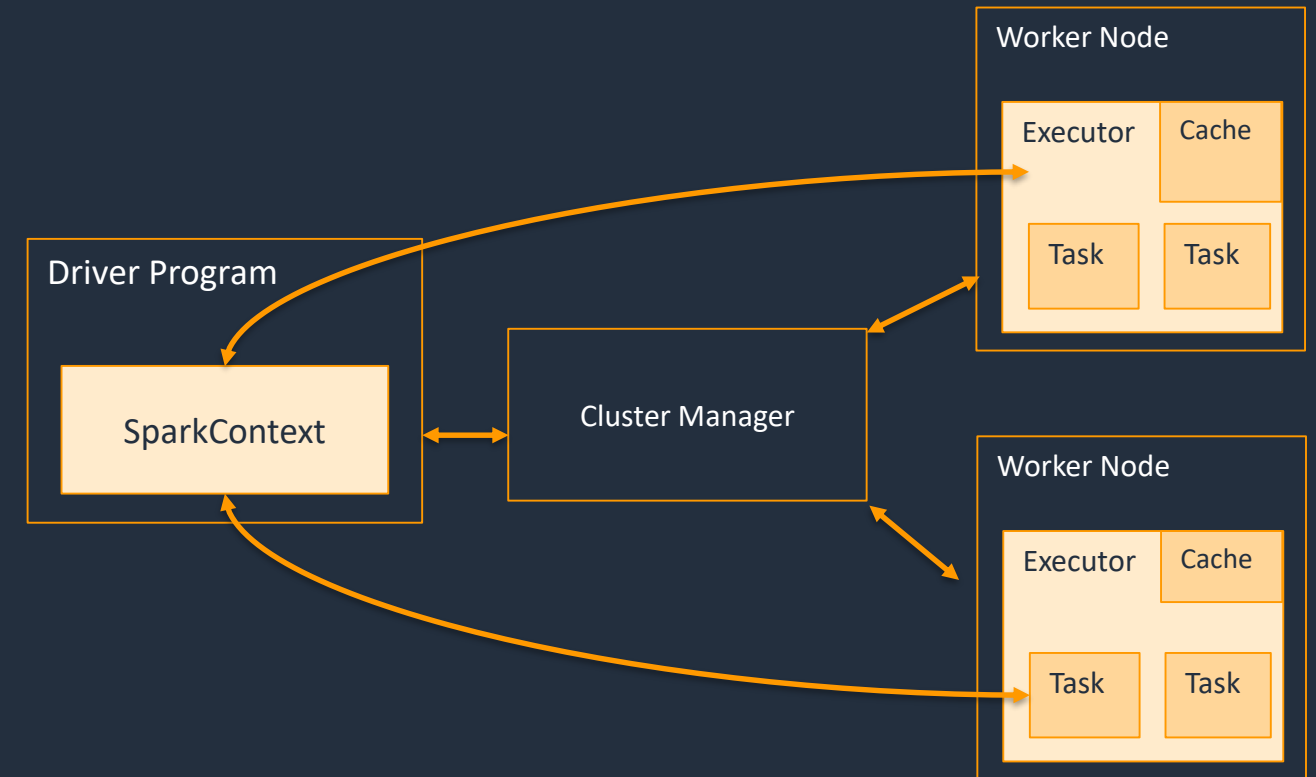


What is the latest version of Spark?

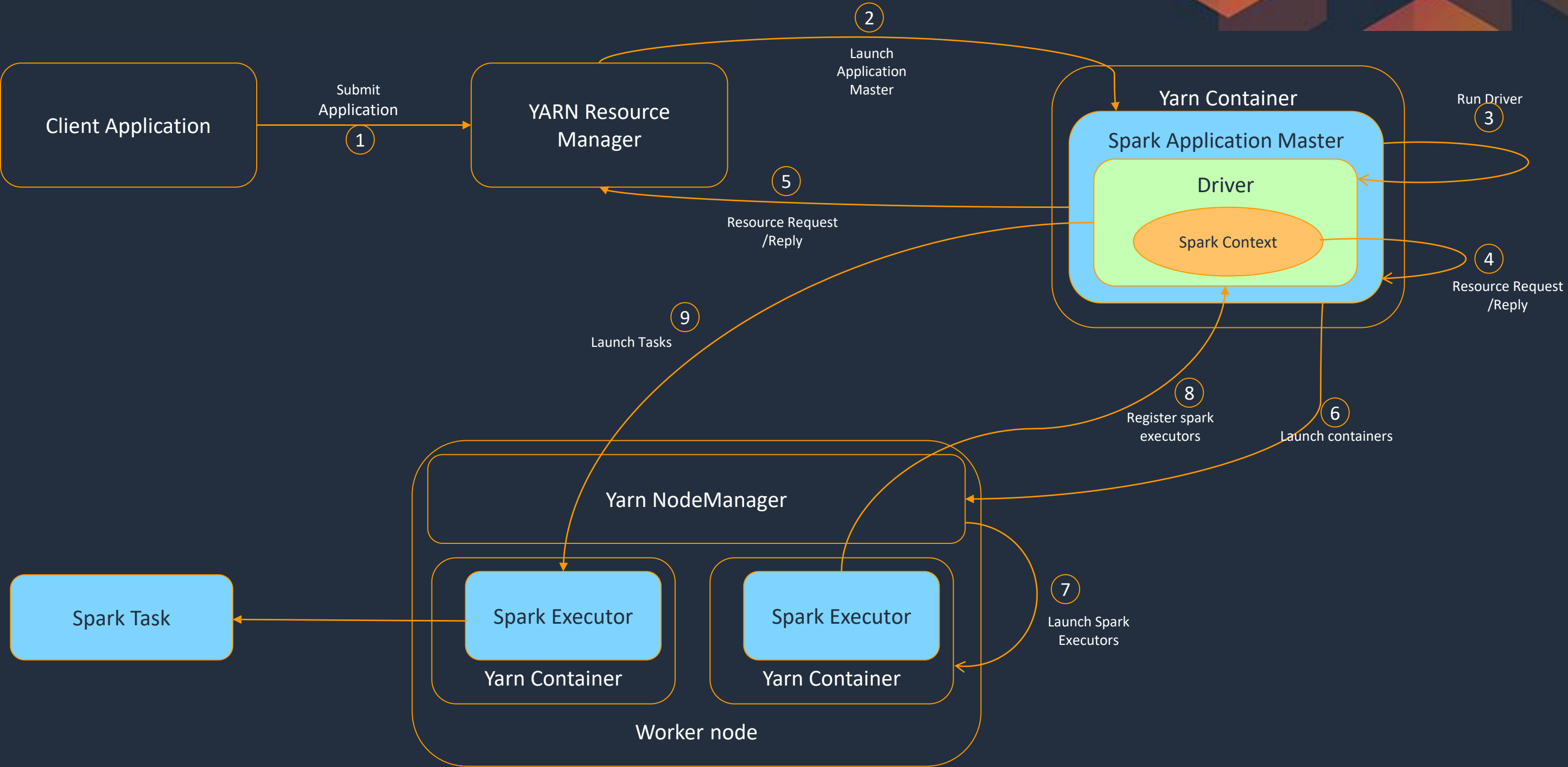
Why is Spark fast?

Spark Terminology

- ❑ Driver program - The process running the main() function of the application and also creates the SparkContext
- ❑ SparkContext - The entry point to any spark functionality
- ❑ Cluster manager - An external service for acquiring resources on the cluster
- ❑ Worker node - Any node that can run application code in the cluster
- ❑ Executor - A process launched for an application on a worker node, that runs tasks and keeps data in memory or disk storage across them
- ❑ Task - A unit of work that will be sent to one executor



Spark Execution Model

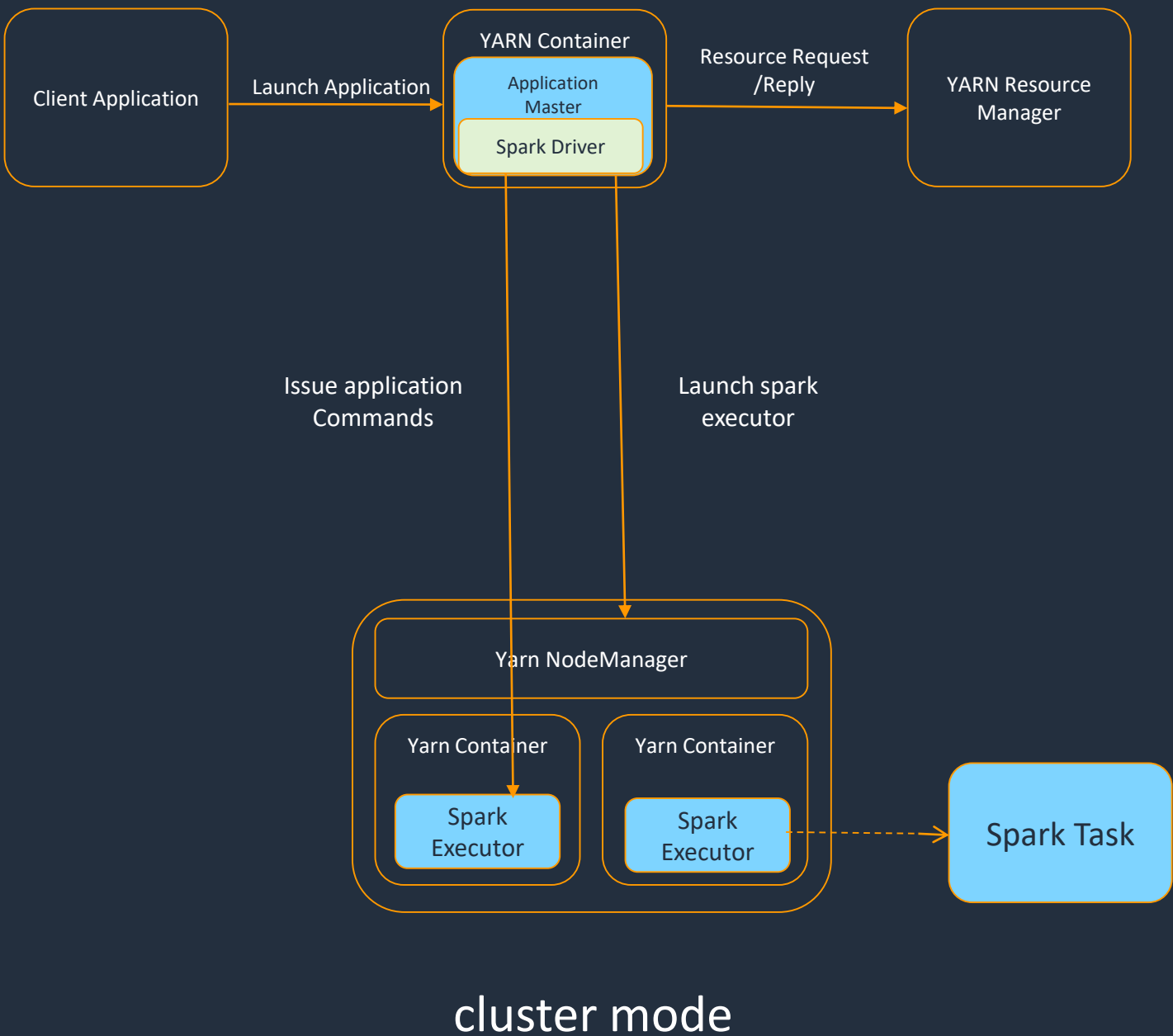
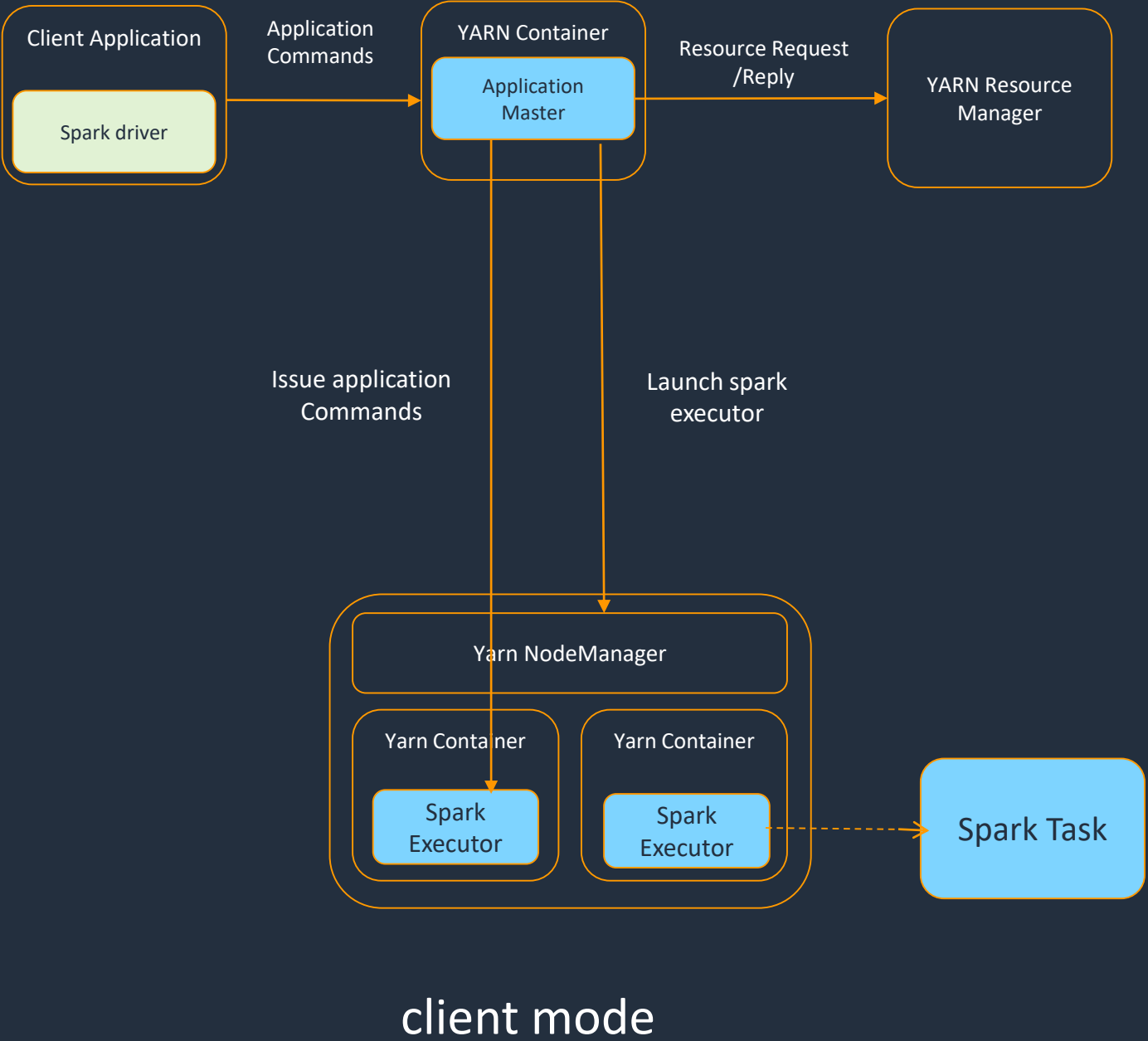


Poll



What are two different modes for spark to run on YARN?

Spark on YARN – client mode vs cluster mode



Running Spark

Binary	Description
spark-shell	Runs Scala REPL (Read-Eval-Print-Loop)
pyspark	Runs Python REPL (Read-Eval-Print-Loop)
spark-sql	Runs SQL REPL (Read-Eval-Print-Loop)
sparkR	Runs R REPL (Read-Eval-Print-Loop)
spark-submit	Submit jar or Python application for execution on cluster

Running Spark(Continued)



```
[hadoop@ip-10-10-10-174 ~]$ pyspark
Python 3.7.9 (default, Feb 18 2021, 03:10:35)
[GCC 7.3.1 20180712 (Red Hat 7.3.1-12)] on linux
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
21/06/21 08:55:24 WARN HiveConf: HiveConf of name hive.server2.thrift.url does not exist
21/06/21 08:55:27 WARN Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading libraries under SPARK_HOME.
Welcome to
```



```
Using Python version 3.7.9 (default, Feb 18 2021 03:10:35)
SparkSession available as 'spark'.
>>> 5 + 5
10
>>>
>>> sc
<SparkContext master=yarn appName=PySparkShell>
```

```
[hadoop@ip-10-10-10-153 ~]$ spark-submit --class org.apache.spark.examples.SparkPi --master yarn --deploy-mode cluster /usr/lib/spark/examples/jars/spark-examples.jar
```

Poll



Where can I deploy Spark?

RDD, Data Frames, Datasets

Resilient Distributed Dataset (RDD)

RDD was the primary user-facing API in Spark since its inception. At the core, an RDD is an immutable distributed collection of elements of your data, partitioned across nodes in your cluster that can be operated in parallel with a low-level API that offers transformations and actions.

DataFrames

Like an RDD, a DataFrame is an immutable distributed collection of data. Unlike an RDD, data is organized into named columns, like a table in a relational database.

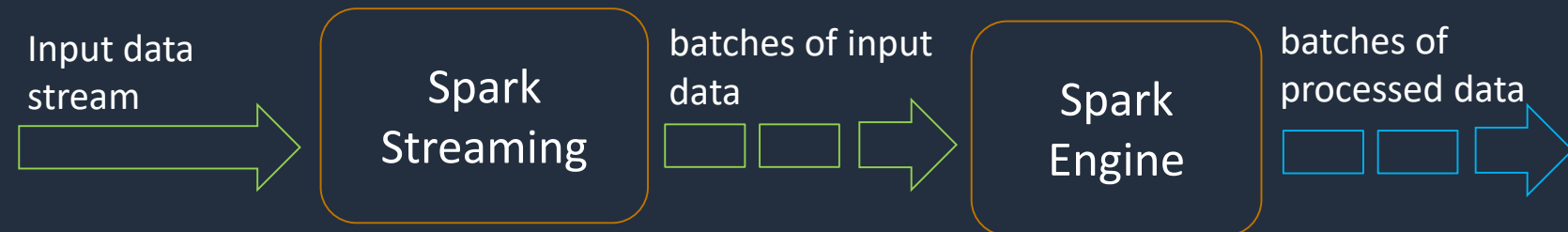
Datasets

Dataset was introduced in spark-1.6 as experimental feature. Dataset takes on two distinct APIs characteristics: a strongly-typed API and an untyped API.

Note : you can move between DataFrame or Dataset and RDDs — by simple API method calls—and DataFrames and Datasets are built on top of RDDs.

Spark Streaming

- ❑ An extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams
- ❑ Processing data streams using DStreams (old API)
- ❑ Dstream or *discretized stream* is a high-level abstraction which represents a continuous stream of data
- ❑ DStreams can be created either from input data streams from sources such as Kafka, and Kinesis, or by applying high-level operations on other DStreams. Internally, a DStream is represented as a sequence of [RDDs](#).



Structured Streaming

- ❑ A scalable, high-throughput, and fault-tolerant stream processing engine built on the Spark SQL engine
- ❑ Processing structured data streams with relation queries (using Datasets and DataFrames, newer API than DStreams)

The Model

Input: data from source as an append-only table

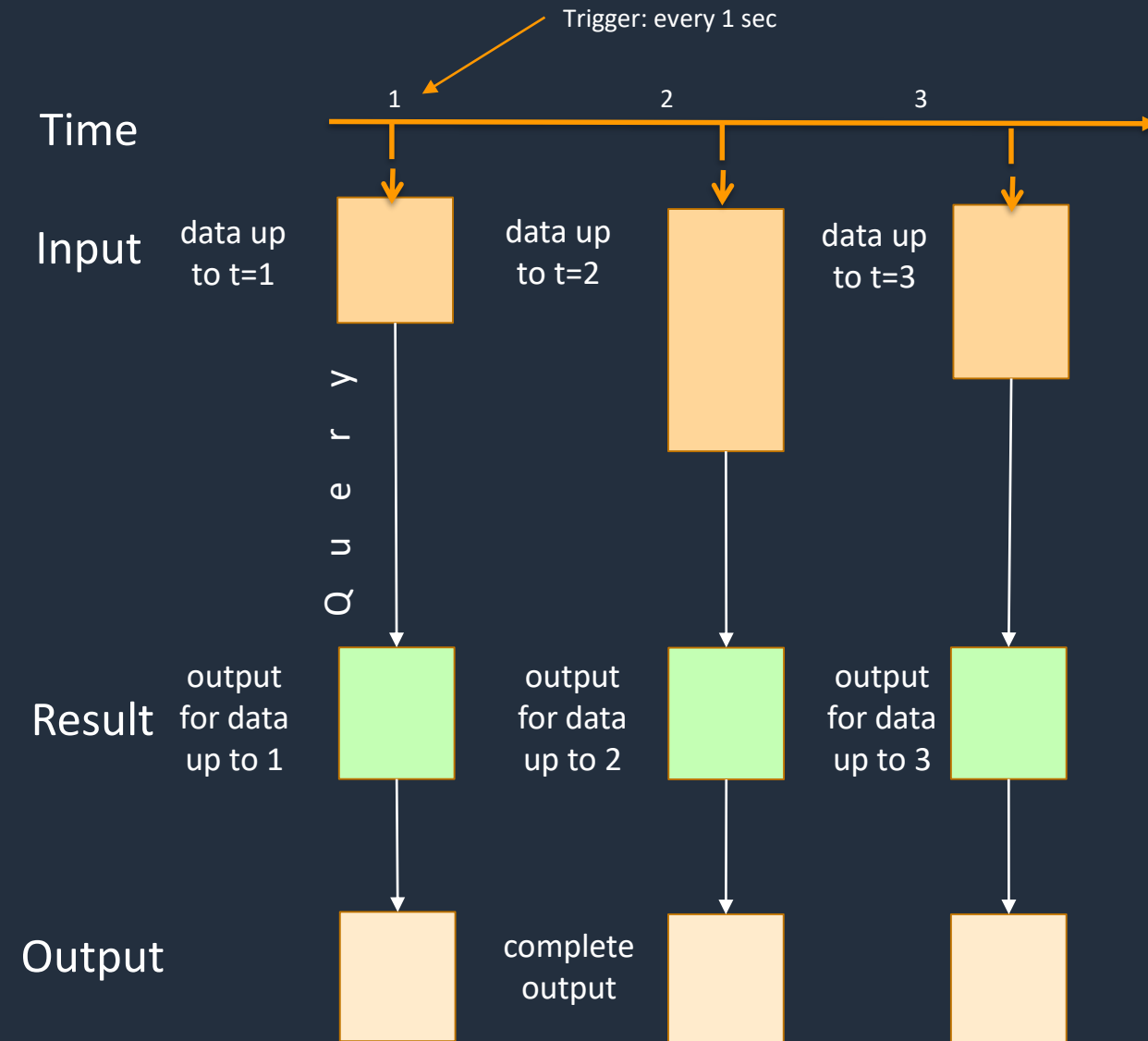
Trigger: how frequently to check for new data

Query: operations on input usual map/filter/reduce

Result: final operated table updated every trigger interval

Output: what part of result to write to data sink after every trigger

Complete output: Write full result table every time



Structured Streaming

- ❑ A scalable, high-throughput, and fault-tolerant stream processing engine built on the Spark SQL engine
- ❑ Processing structured data streams with relation queries (using Datasets and DataFrames, newer API than DStreams)

The Model

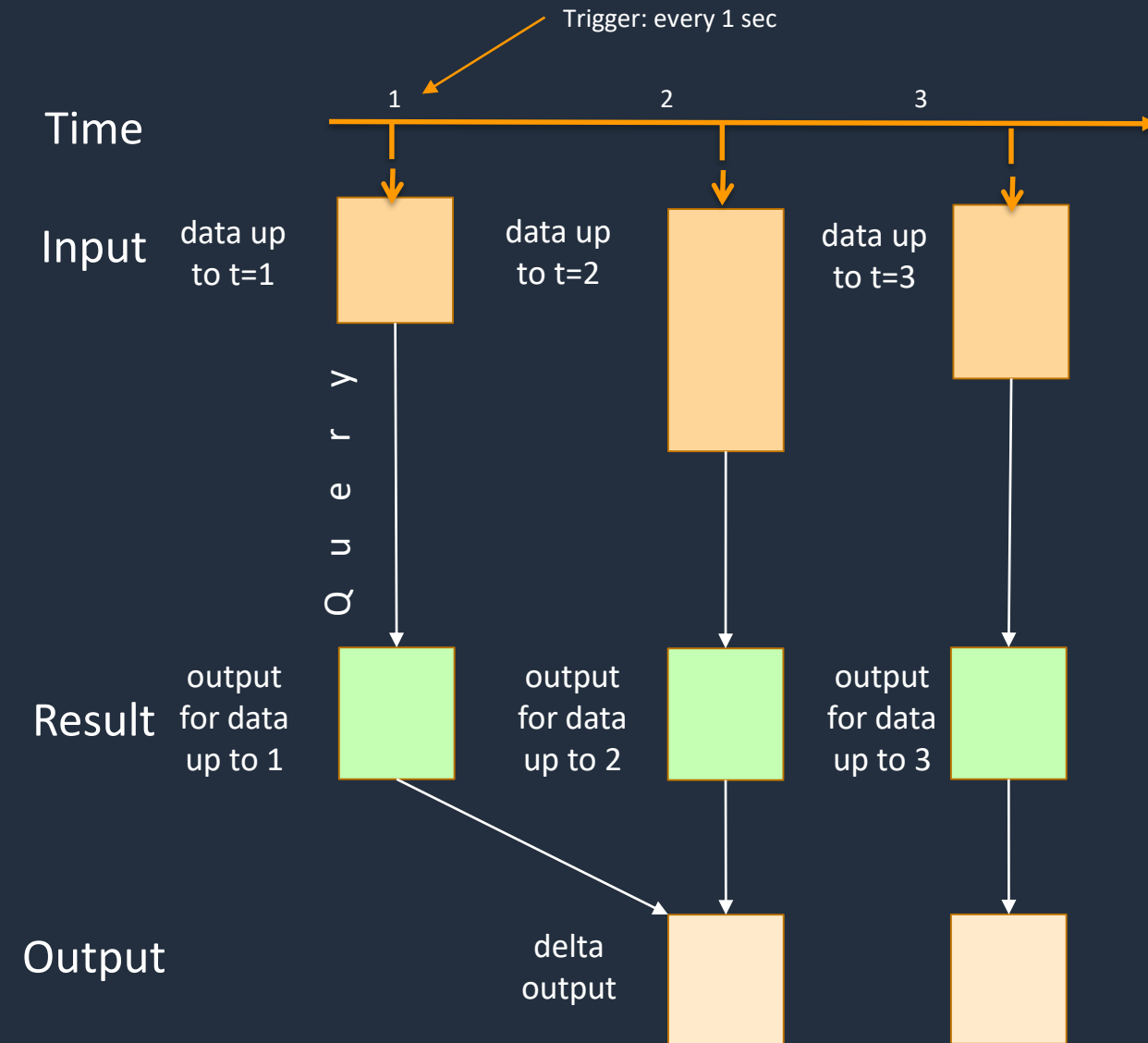
Result: final operated table updated every trigger interval

Output: what part of result to write to data sink after every trigger

Complete output: Write full result table every time

Delta output: Write only the rows that changed in result from previous batch

Append output: Write only new rows



Poll



Between Spark Streaming and Structured streaming, which one is better and why?

Monitoring and Debugging

- ❑ [Spark UI](#)

Job performance, task breakdown of job, information about cached DataFrames, and more



- ❑ [Application History](#)

Application history from EMR console

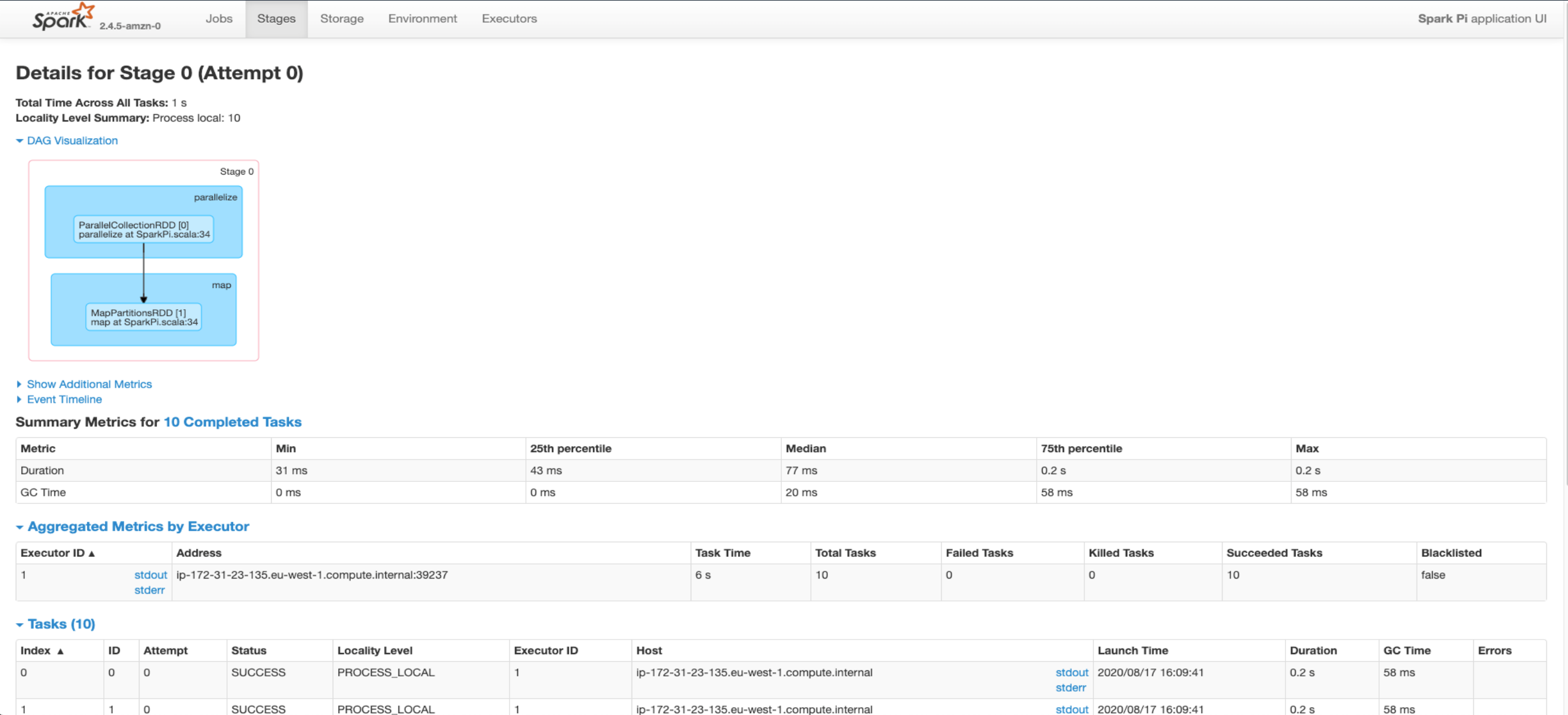
- ❑ [Logs pushing to s3](#)

Logs produced by driver and executors on each node can browse through log folders in EMR console

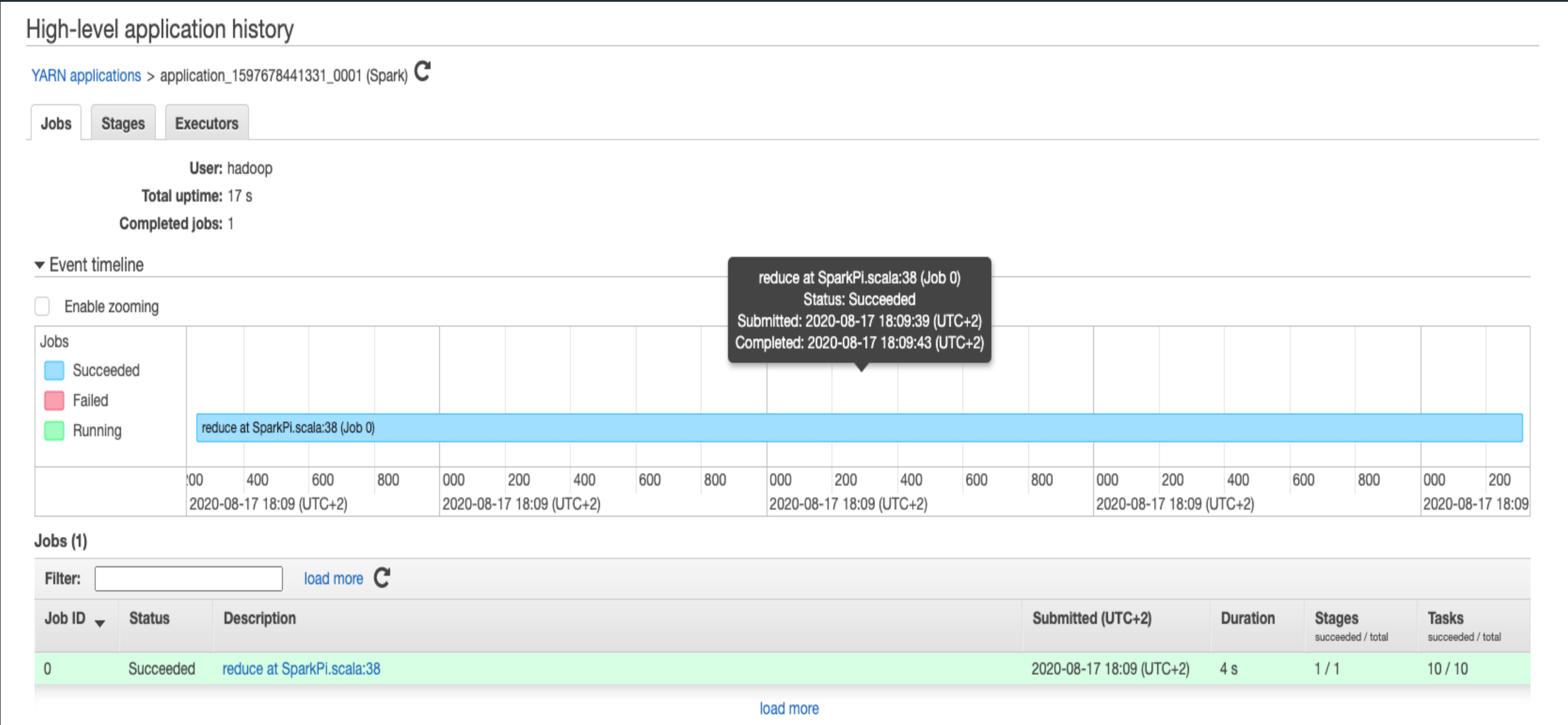
- ❑ [CloudWatch metrics](#)

CloudWatch metrics in the EMR console

Monitoring and Debugging – Using Spark UI



Monitoring and Debugging – Using EMR console



Monitoring and Debugging – Using logs on EMR console

The screenshot displays the AWS EMR console for a cluster named "My cluster" in a "Waiting" state. The console is divided into two main sections: "Summary" and "Configuration details".

Summary:

- ID:** j-22OW7EOLD6BOC
- Creation date:** 2020-08-17 17:30 (UTC+2)
- Elapsed time:** 3 hours, 56 minutes
- After last step completes:** Cluster waits
- Termination protection:** On [Change](#)
- Tags:** -- [View All / Edit](#)
- Master public DNS:** ec2-34-251-5-136.eu-west-1.compute.amazonaws.com [Connect to the Master Node Using SSH](#)

Configuration details:

- Release label:** emr-5.30.1
- Hadoop distribution:** Amazon 2.8.5
- Applications:** Pig 0.17.0, Spark 2.4.5, Livy 0.7.0, Hive 2.3.6
- Log URI:** s3://aws-logs-961197695763-eu-west-1/elasticmapreduce/ [Download](#)
- EMRFS consistent view:** Disabled
- Custom AMI ID:** --

Application user interfaces:

- Persistent user interfaces:** [Spark history server, YARN timeline server, Tez UI](#)
- On-cluster user interfaces:** Not Enabled [Enable an SSH Connection](#)

Security and access:

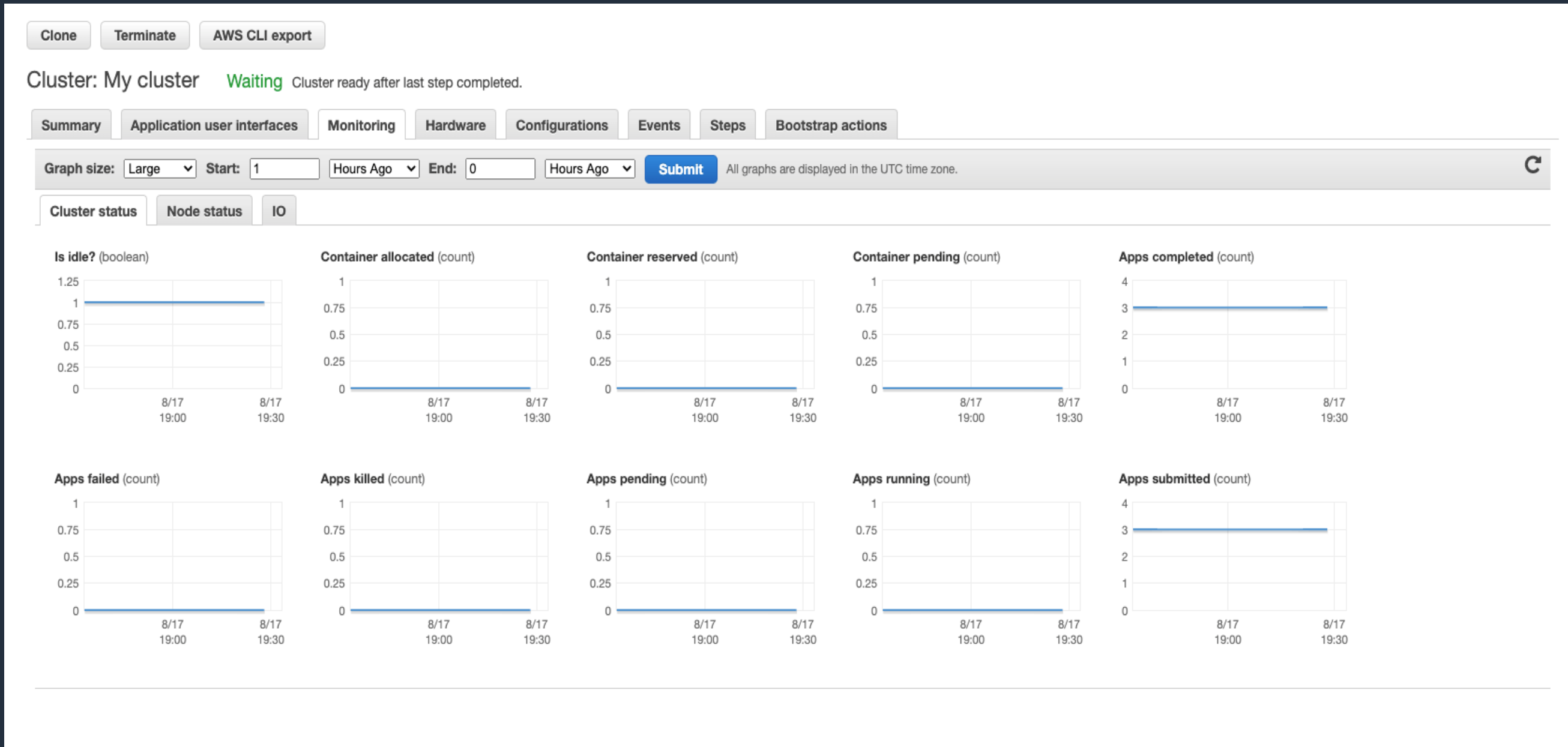
- Key name:** umesh
- EC2 instance profile:** EMR_EC2_DefaultRole
- EMR role:** EMR_DefaultRole
- Auto Scaling role:** EMR_AutoScaling_DefaultRole
- Visible to all users:** All [Change](#)
- Security groups for Master:** sg-0517ec341e3b05025 [\(ElasticMapReduce-master\)](#)
- Security groups for Core & Task:** sg-07a0d7fb554901f24 [\(ElasticMapReduce-slave\)](#)

View S3 overlay:

The "View S3" overlay shows the log URI: s3://aws-logs-961197695763-eu-west-1/elasticmapreduce/j-22OW7EOLD6BOC/containers/application_1597678441331_0001/container_1597678441331_0001_01. The log content is displayed as:

```
.. (go up)
stderr.gz
```

Monitoring and Debugging – Using CW metrics on EMR console





Break

Will resume in 15 minutes

Spark Lab

- ❑ Log in to the EMR Master Node using SSM
- ❑ Run Spark in YARN client/YARN cluster/local mode
- ❑ Python word count running locally on 5 cores word count
- ❑ Python word count hdfs

Find the lab at: <https://github.com/aws-support-bigdata-cpt-vls/2021/tree/main/Day%202/Spark%20Lab>

Questions?



References

<https://spark.apache.org/docs/latest/index.html>

<https://aws.amazon.com/big-data/what-is-spark/>

<https://spark.apache.org/docs/3.1.1/structured-streaming-programming-guide.html>

<https://spark.apache.org/docs/3.1.1/streaming-programming-guide.html>

Consolidated Lab

Odwa Yekela
Cloud Support Engineer
AWS

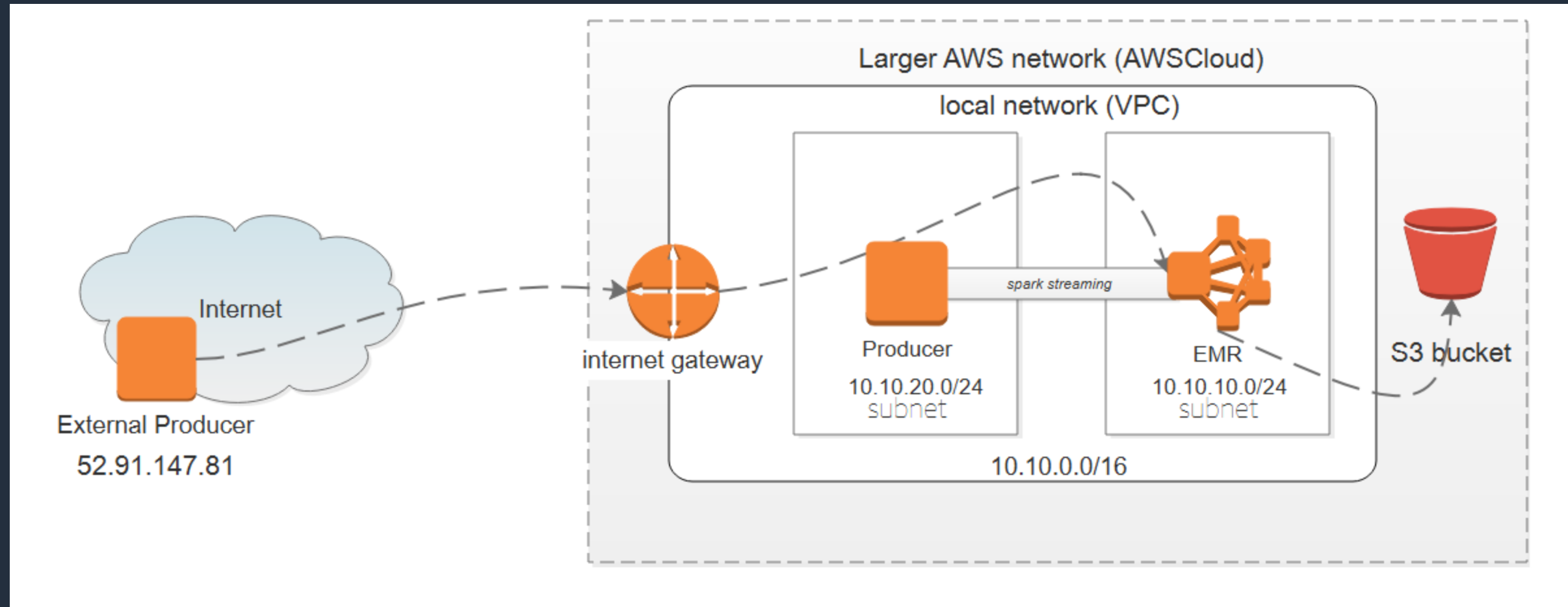


Mission

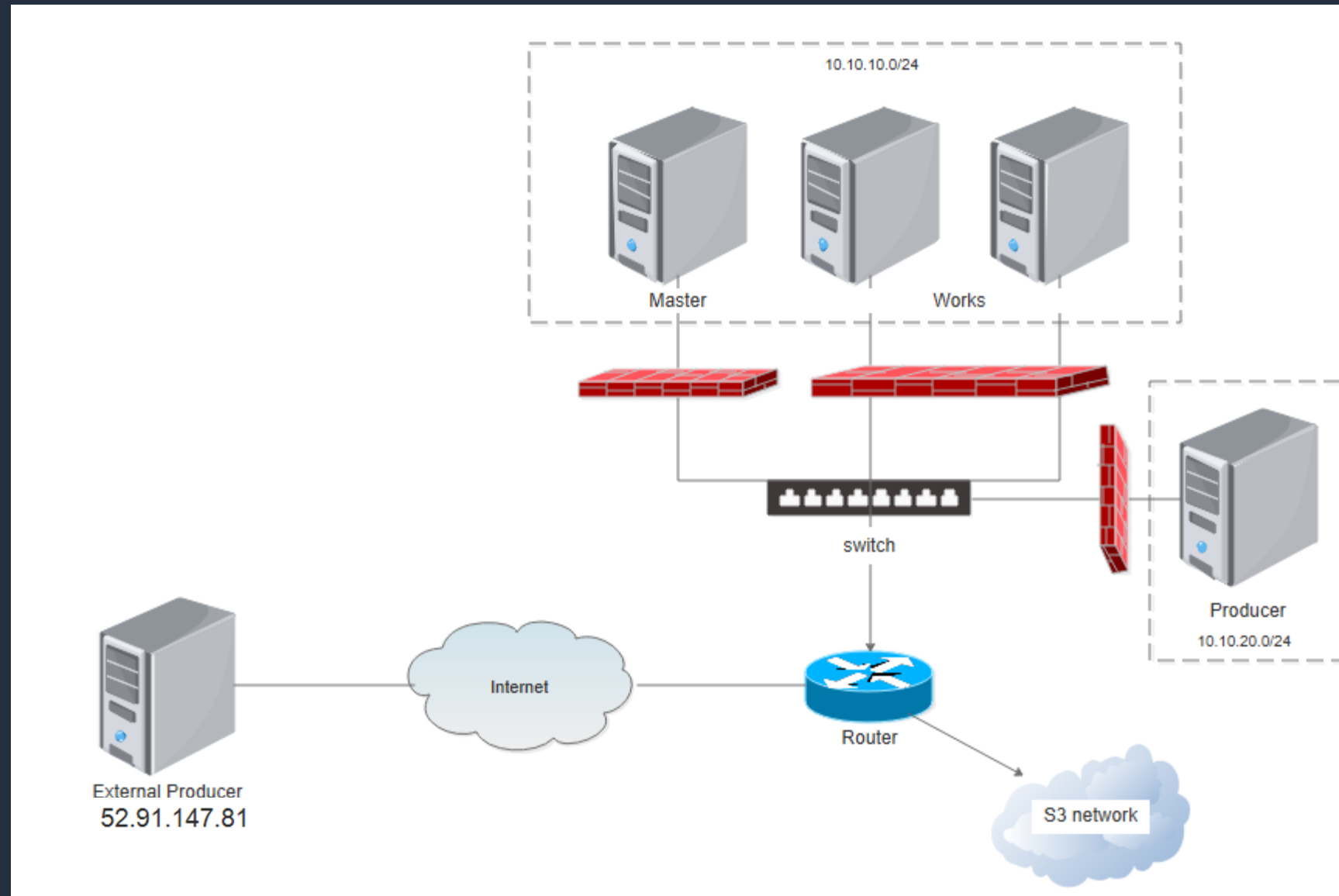
In this Lab we will be setting up our EC2 instance to produce streaming traffic to our Hadoop cluster which will have a spark streaming script acting as the spark application for consuming the data. You will see how something as simple as a wordcount script can be leveraged over a TCP socket connection to allow multiple spark executors to process the data and produce output into HDFS and S3. We will be able to monitor and control this job using YARN commands and interactive web UI applications (Resource Manager webUI, Spark History webUI, HDFS webUI).

1. Setup environment for streaming app
2. Package and submit spark stream application to EMR cluster
3. Monitor our job while it runs
4. Check output from the complete job

Setup



Physical representation (conceptually)



Useful Network commands

System commands

- `lsof -i :9999`
- `ping 10.10.10.1 -c 5`
- `netstat -ntlp | grep 9999`
- `nc -zv 52.91.147.81 9999`

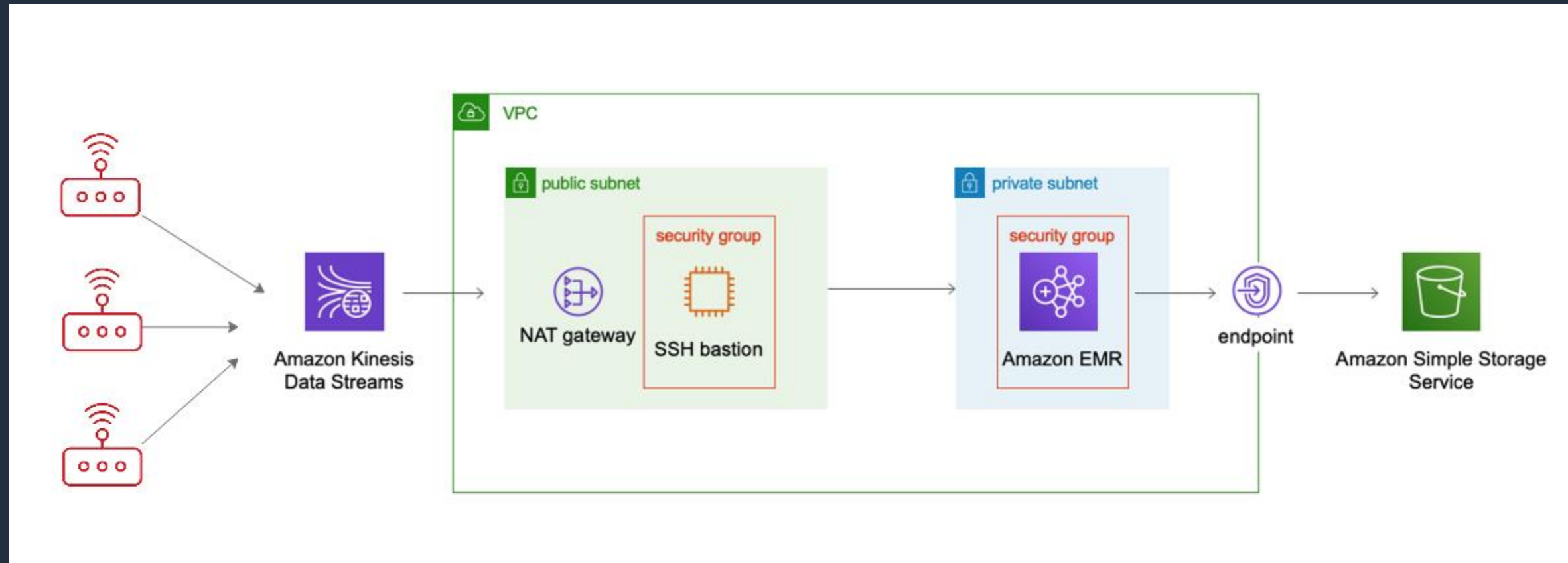
HDFS

- `touch helloworld.txt`
- `hdfs dfs -mkdir /user`
- `hdfs dfs -copyFromLocal helloworld.txt hdfs:///test/`
- `hdfs dfs -ls`
- `hdfs dfs -setrep 3 hdfs:///`

YARN

- `Yarn top`
- `Yarn application list \`
- `Yarn application -kill <application id>`

Production setup



Questions?





Thank You