Big Data Hadoop and Spark Developer

Deep Dive into Apache Spark Framework

Learning Objectives

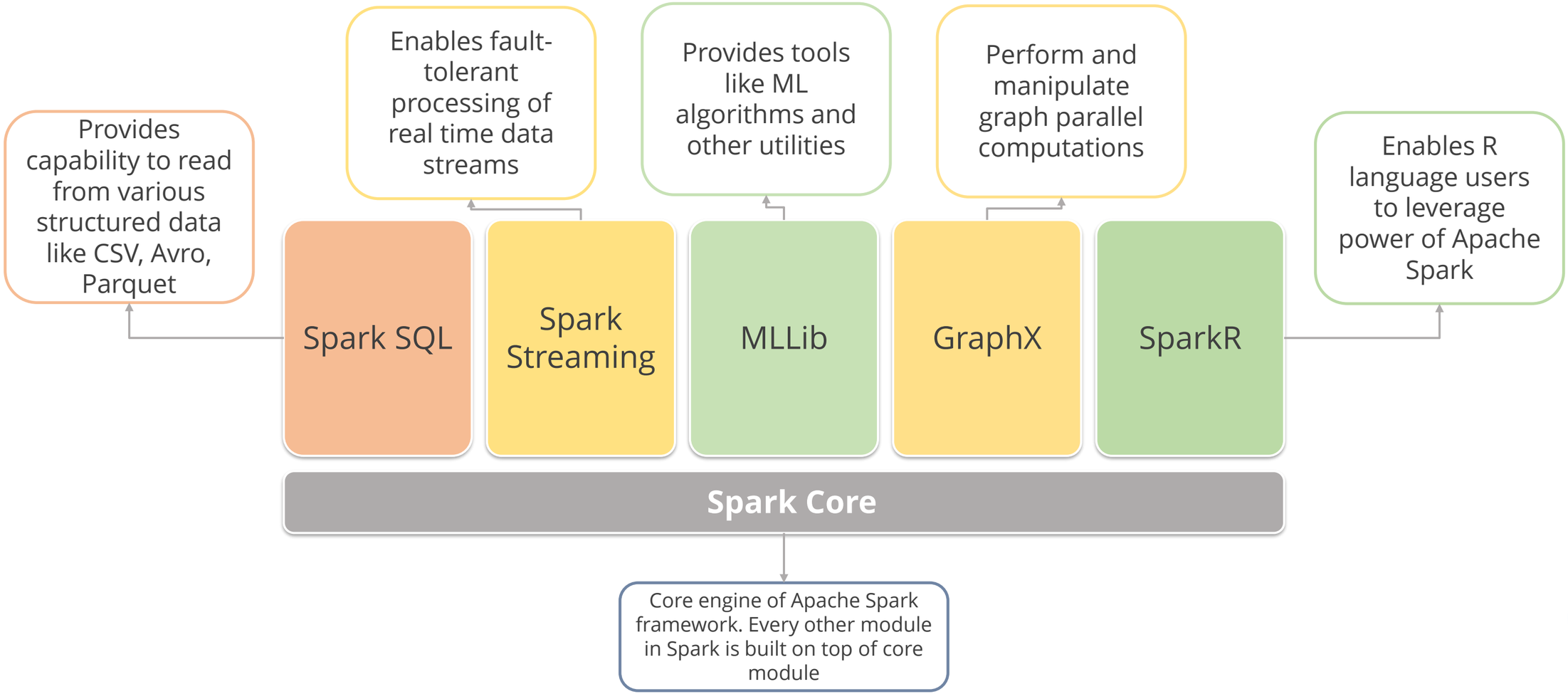
By the end of this lesson, you will be able to:

- Define Spark components and its architecture
- Summarize Spark Deployment Modes
- Work with PySpark Shell
- Submit PySpark Job in a program
- Work with Spark Web UI

Spark Components

Components of a Spark Project

Apache Spark is an alternative to Hadoop MapReduce that includes six components. These are:



Components of a Spark Project

Spark Core and RDDs

- It provides basic I/O, distributed task dispatching, and scheduling as the foundation.
- RDDs can be created by applying coarse-grained transformations or referencing external datasets.

Spark SQL

- As a component lying on the top of Spark Core, it introduces SchemaRDD, which can be manipulated.
- It supports SQL with ODBC/JDBC server and command-line interfaces.

Components of a Spark Project

Spark Streaming

- It leverages the fast-scheduling capability of Spark Core.
- It ingests data in small batches and performs RDD transformations on them.

MLlib

- It is a distributed machine learning framework built on top of Spark.
- It is nine times faster than the Hadoop disk-based version of Apache Mahout.

Components of a Spark Project

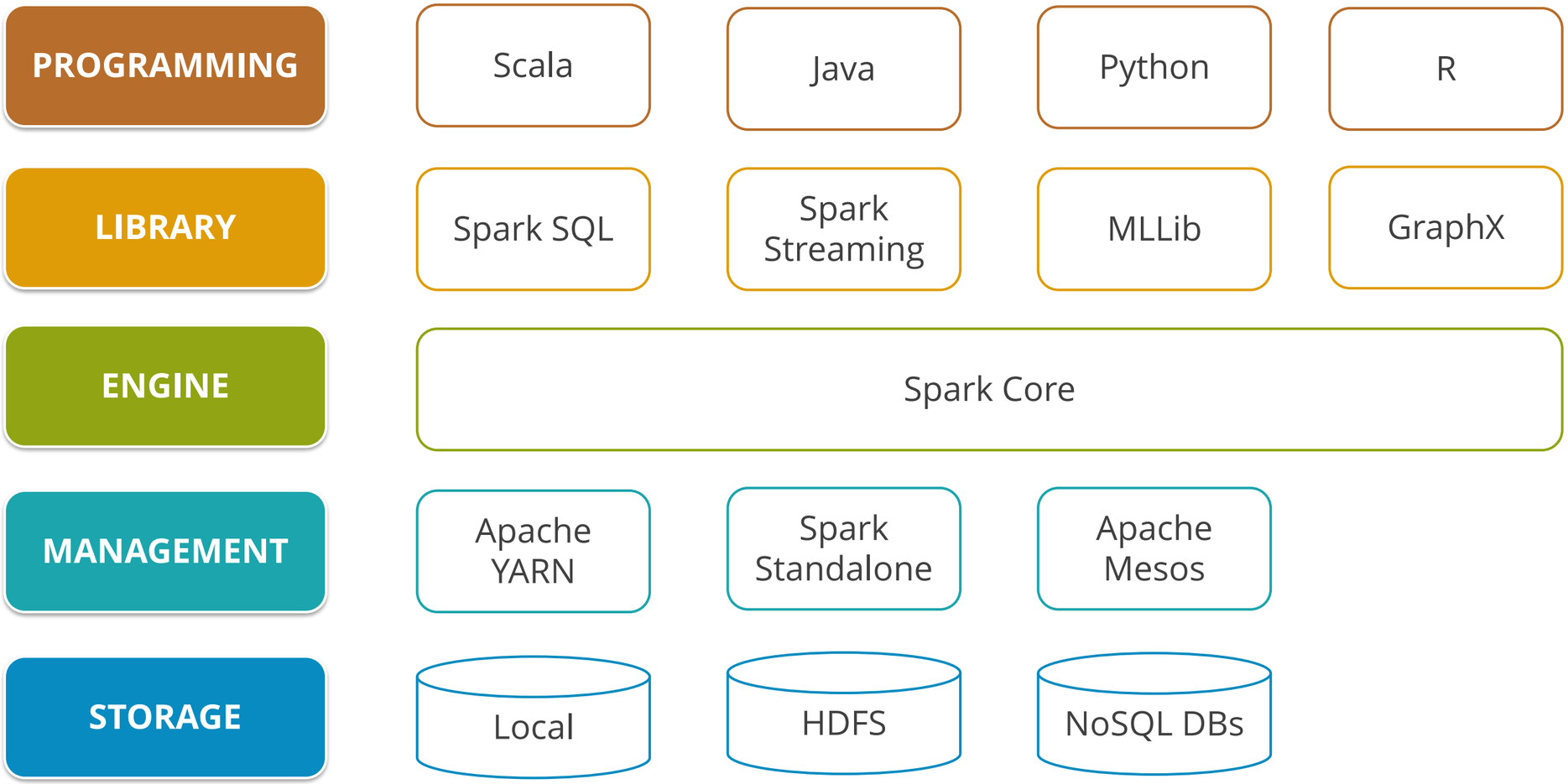
GraphX

- It is a distributed graph processing framework built on top of Spark.
- It provides an API and an optimized runtime for the Pregel abstraction.

SparkR

- It provides a simple interface for using Apache Spark from R.
- It provides a distributed data frame implementation for huge datasets.
- It supports operations like selection, filtering, and aggregation.

Category of Spark Components



Application of In-Memory Processing

In column-centric databases, similar pieces of information can be stored together. The working of in-memory processing can be explained as below:

The entire information is loaded into memory, eliminating the need for indexes, aggregates, optimized databases, star schemas, and cubes.

Compression algorithms are used by most in-memory tools, thereby reducing the in-memory size.

Application of In-Memory Processing

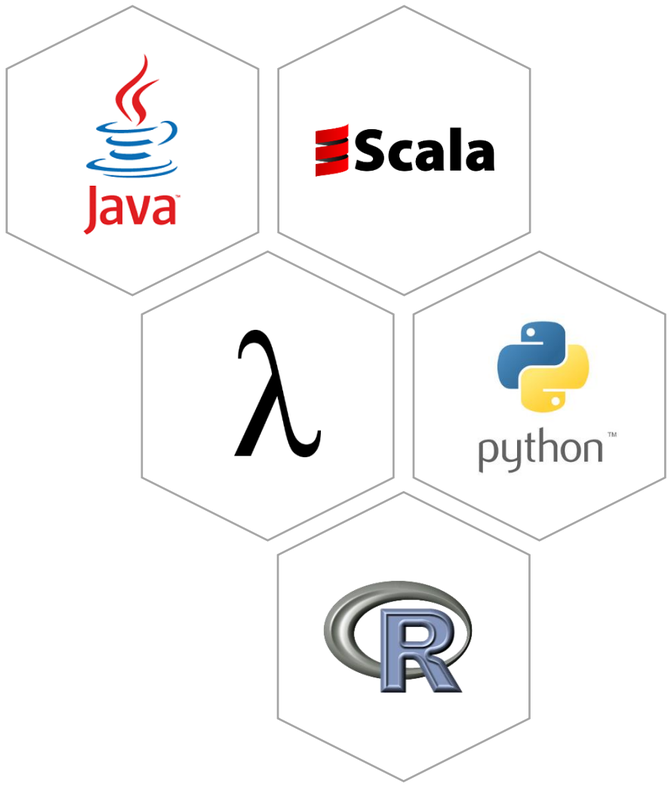
The analysis of data can be flexible in size and can be accessed within seconds by concurrent users with excellent analytics potential.

It is possible to access visually rich dashboards and existing data sources.

Querying the data loaded into memory is different from caching.

Language Flexibility in Spark

Spark is popular for its performance benefits over MapReduce.
Another important benefit is language flexibility, as explained below.



Support for various development languages

Spark supports popular development languages like Java, Scala, and Python and will support R.

Capability to define functions in-line

With the temporary exception of Java, a common element in these languages is that they provide methods to express operations using lambda functions and closures.

Hadoop Ecosystem vs. Spark

Every type of data processing can be done using Spark that is executed in Hadoop. They are:

Batch Processing: Spark batch can be used over Hadoop MapReduce.

Structured Data Analysis: Spark SQL can be used with SQL.

Machine Learning Analysis: MLlib can be used for clustering, recommendations, and classification.

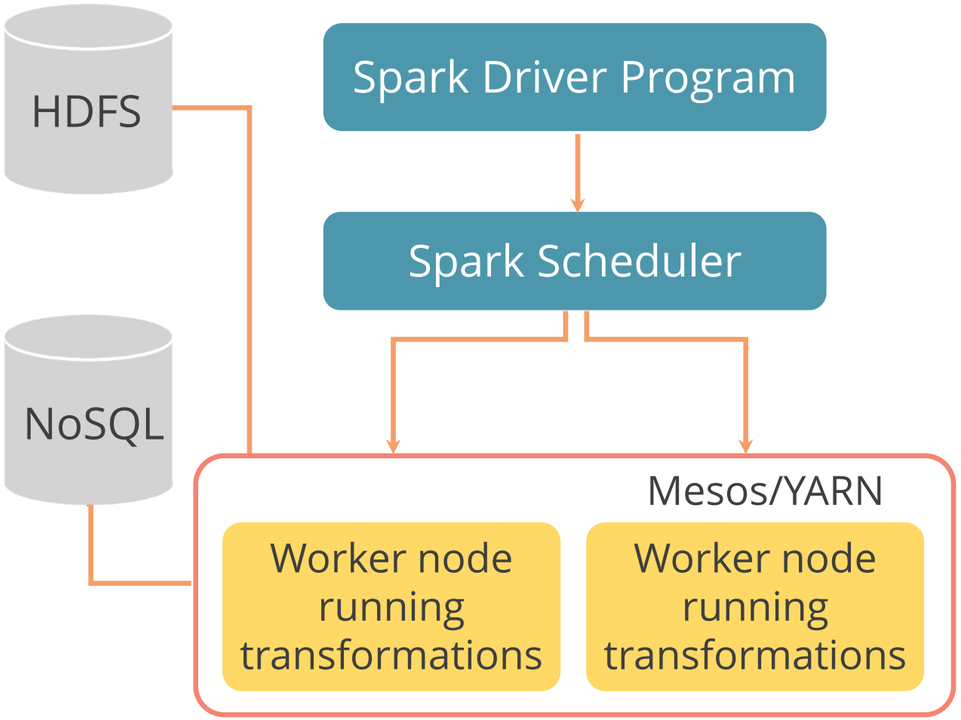
Interactive SQL Analysis: Spark SQL can be used over Stringer, Tez, or Impala.

Real-time Streaming Data Analysis: Spark streaming can be used over specialized libraries like Storm.

Spark Architecture

Spark Architecture

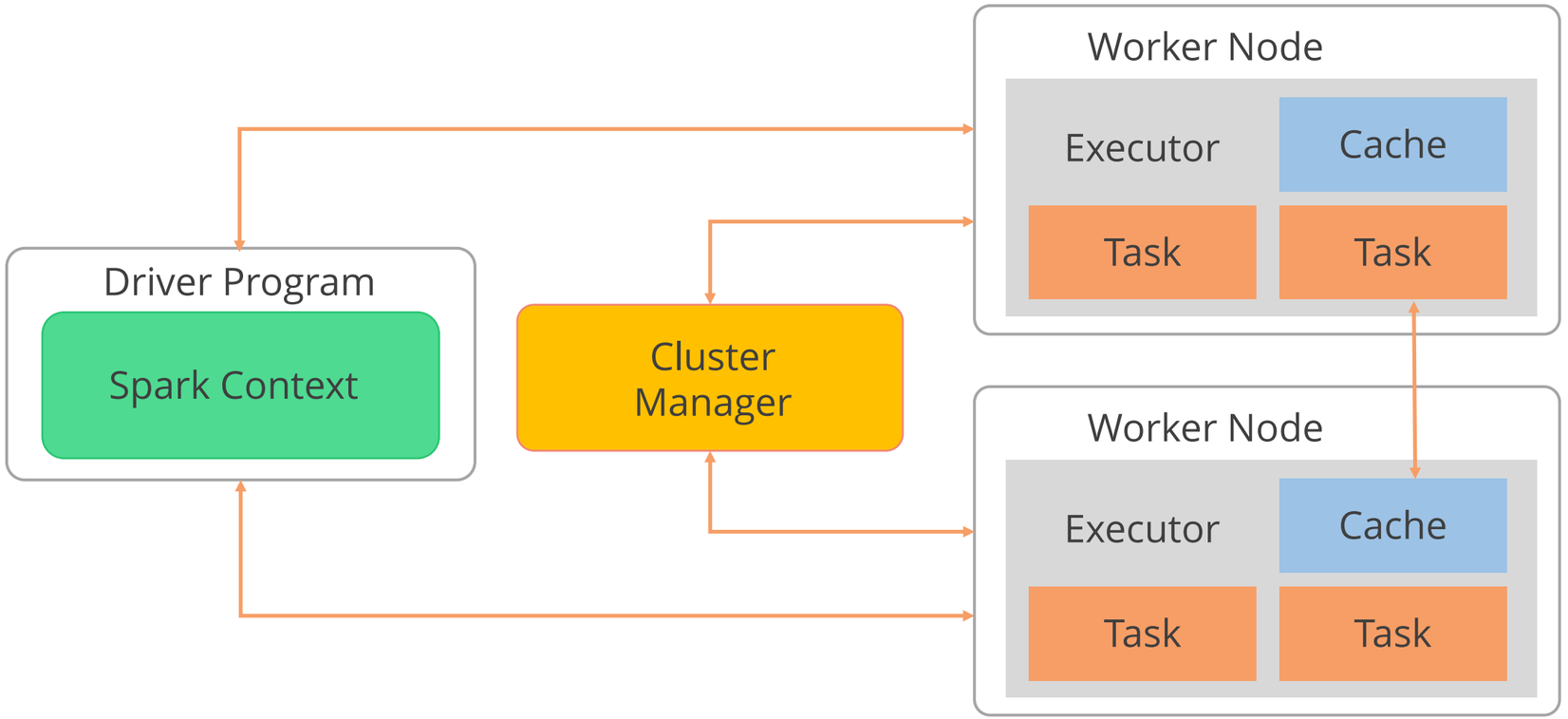
The components of the Spark execution architecture are:



- **Spark-submit script:** Used to launch applications on a cluster; can use all cluster managers through a uniform interface
- **Spark applications:** Run as independent sets of processes on a cluster and are coordinated by the SparkContext object in the driver program
- **Cluster managers:** Supported cluster managers are Standalone, Apache Mesos, and Hadoop YARN
- **Spark's EC2 launch scripts:** Make launching a standalone cluster easy on Amazon EC2

Working of Apache Spark Architecture

The following diagram depicts the basic Apache Spark architecture:



Working of Apache Spark Architecture

The components of the Spark execution architecture are explained below:

1. **Driver Program**
   It is the execution point of the program. It creates SparkContext to schedule job execution and negotiates with the cluster manager.

2. **SparkContext object in Driver Program**
   It helps in coordinating with all the distributed processes and allows resource allocation.

3. **Cluster Manager**
   It provides executors for the execution of code.

Working of Apache Spark Architecture

The components of the Spark execution architecture are explained below:

4

Executors

It uses cache slots to keep data in memory, and task slots are Java threads that run the code. Executors run tasks scheduled by the driver.

5

Worker Node

They are the slave nodes that execute the tasks.

Spark Cluster in Real World

Running Spark in Different Modes

The different deployment modes of Spark are:

Can be launched manually by using launching scripts or starting a master and workers, which are used for development and testing

Has advantages like scalable partitioning among different Spark instances and dynamic partitioning between Spark and other frameworks

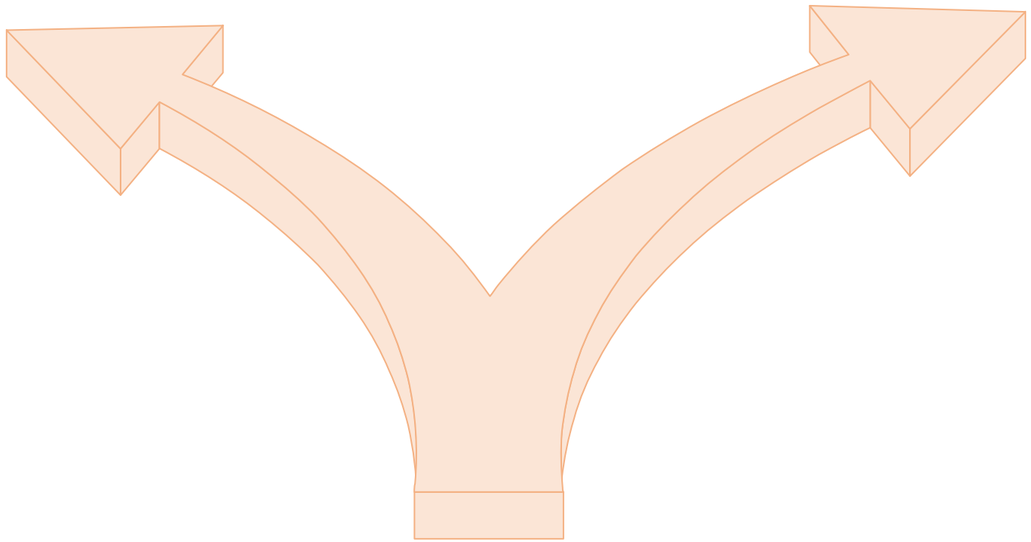
Has all the parallel processing and benefits of the Hadoop cluster

Has key-value pair benefits of Amazon

Deployment Modes: YARN

Two deployment modes can be used to launch Spark applications on YARN.

Client mode



Cluster mode

Deployment Modes: YARN

Client mode

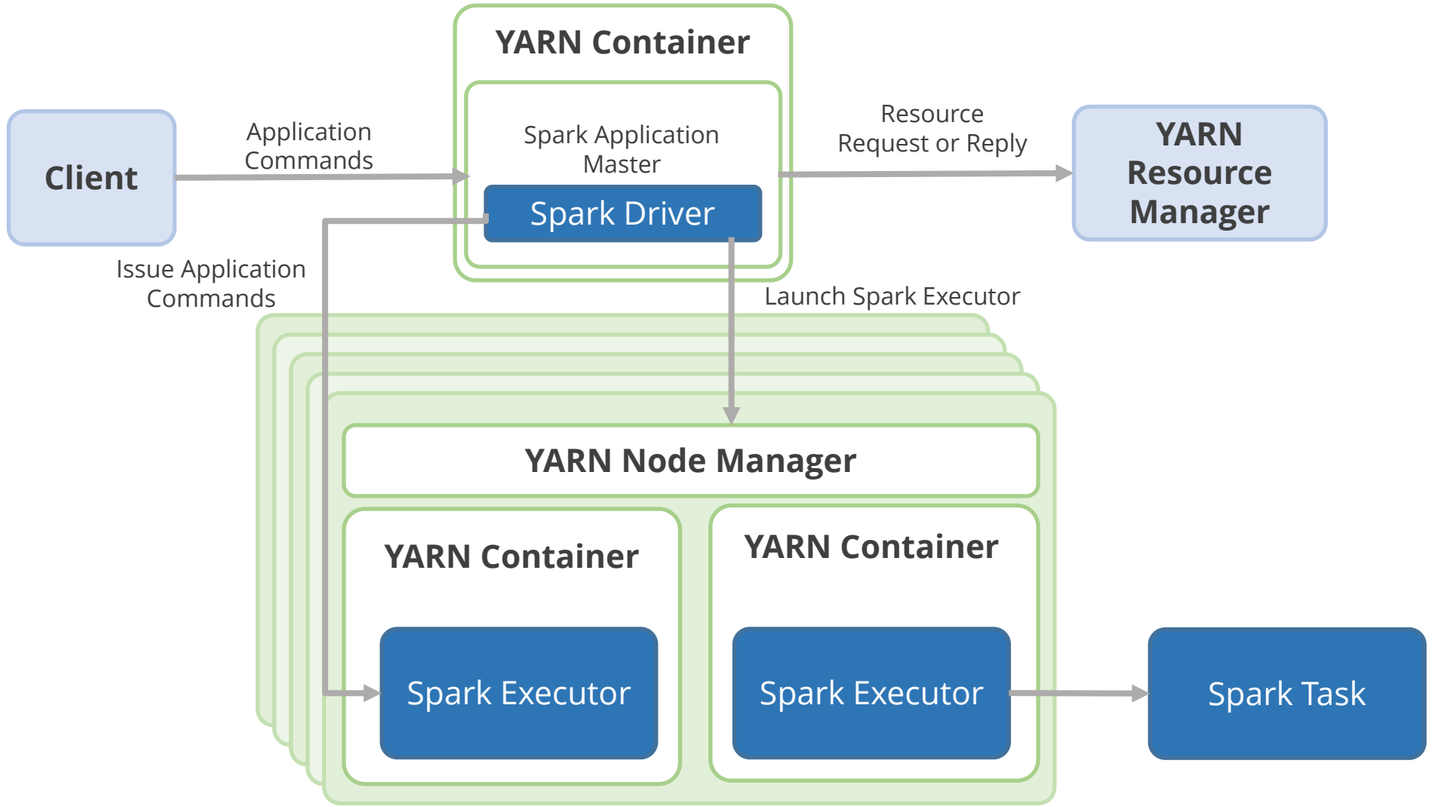
```
/bin/spark-submit --class org.apache.spark.examples.SparkPi --master yarn --deploy-mode client /path/to/examples.jar
```

Cluster mode

```
/bin/spark-submit --class org.apache.spark.examples.SparkPi --master yarn --deploy-mode cluster /path/to/examples.jar
```

Client Mode: YARN

The architecture of YARN in client mode is depicted in the diagram below.



Client Mode: YARN

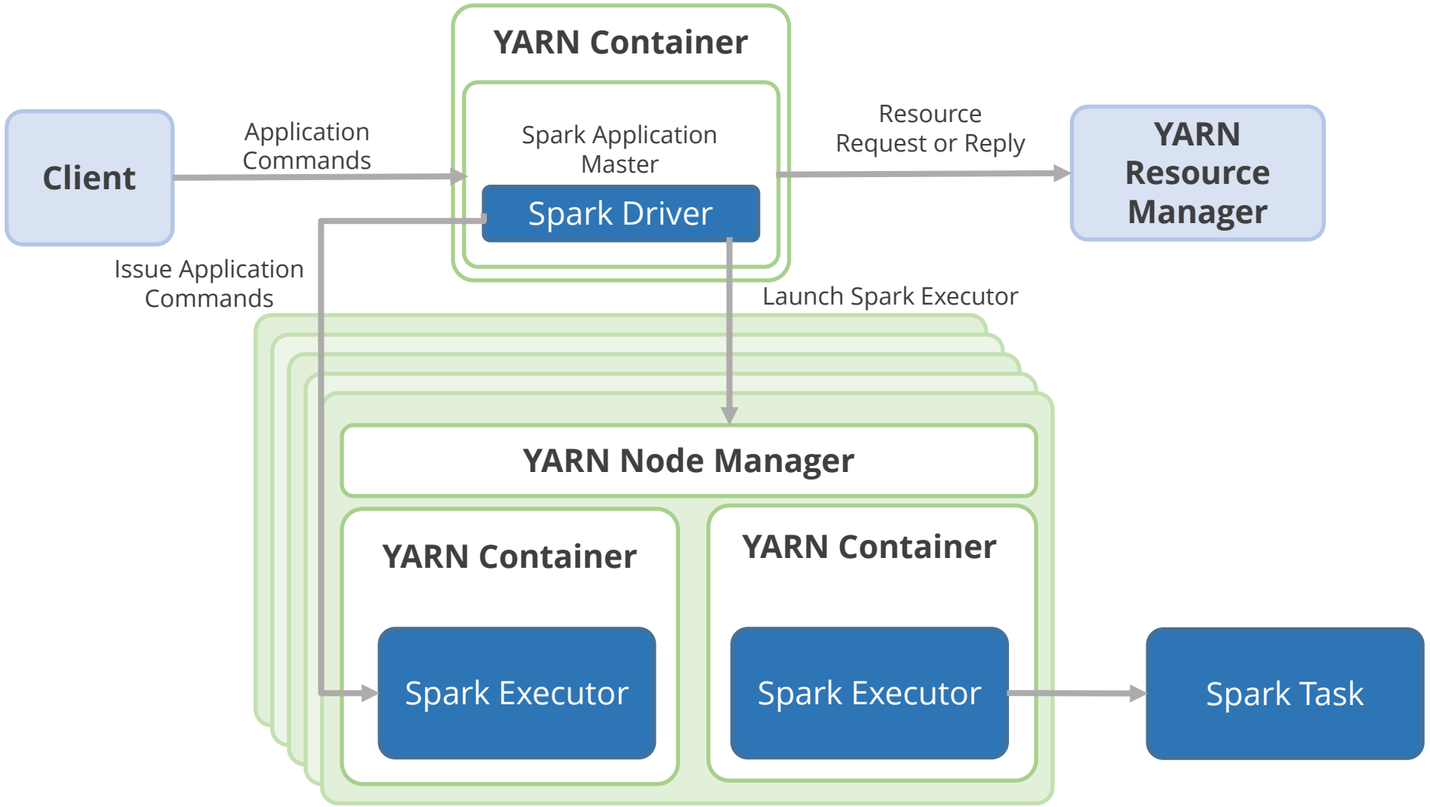
The Spark Driver runs on the host from where the job is submitted in client mode.

The Application Master's job is to request resources (executor containers) from YARN.

The client communicates with the containers to coordinate the work after they start.

Cluster Mode: YARN

The architecture of YARN in cluster mode is depicted in the diagram below.



Cluster Mode: YARN

Spark Driver runs in the Application Master on a cluster machine.

A single process in a YARN container is responsible for both driving the application and requesting resources from YARN.

The client that launches the application does not run for the lifetime of the application.

Introduction to PySpark Shell

Spark Shell

PySpark

- The Spark Shell provides interactive data exploration (REPL).
- It is used for the execution of PySpark statements.
- Spark Shell is available for Scala, Python, and R.
- The pyspark command is used to launch Spark from the Python shell, which is also known as PySpark.

SparkContext

PySpark

- It is the main entry point of Spark API.
- Every Spark application requires a SparkContext.
- Spark Shell provides a preconfigured SparkContext called sc.
- A driver program starts when we execute any Spark application.
- The operations are then executed by the driver program inside the executors on worker nodes.

PySpark Shell in Simplilearn Lab

```
labuser@apachehadooplab:~$ pyspark3
Python 3.8.10 (default, Nov 14 2022, 12:59:47)
[GCC 9.4.0] 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 setLogLeve
l(newLevel).
2024-02-14 07:20:50,516 WARN util.NativeCodeLoader: Unable to load native-hadoop
 library for your platform... using builtin-java classes where applicable
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.2.3
      /_/

Using Python version 3.8.10 (default, Nov 14 2022 12:59:47)
Spark context Web UI available at http://localhost:4040
Spark context available as 'sc' (master = local[*], app id = local-1707895252255
).
SparkSession available as 'spark'.
>>>
```

Submitting PySpark Job

Submitting PySpark Job

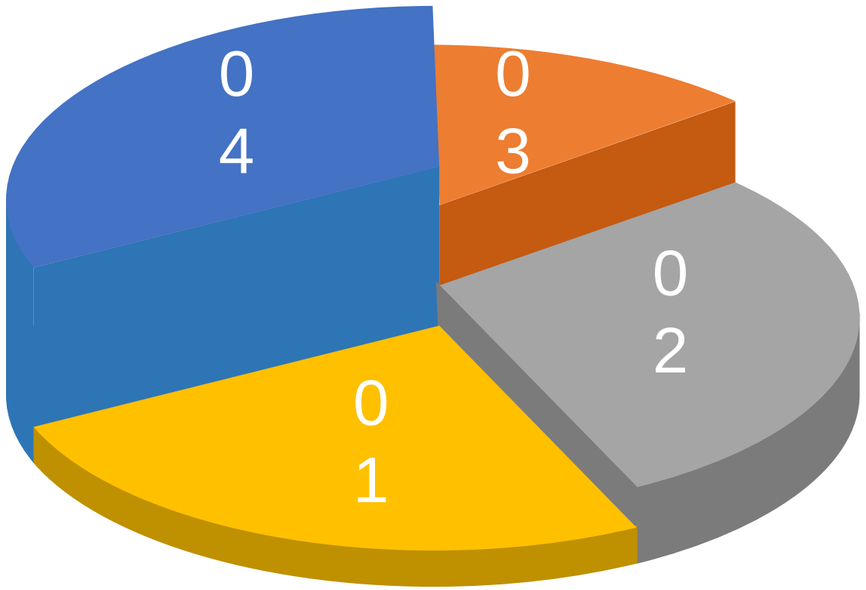
The spark-submit command submits PySpark jobs on the spark cluster.

Syntax:

```
spark-submit \
--master yarn \
--deploy-mode cluster  \
example123.py
```

Submitting PySpark Job

The common options used are:



--class
The entry point for an application

--master
The master URL for the cluster

--deploy-mode
Whether the driver should be deployed on the worker nodes

--conf
Arbitrary Spark configuration property in key=value format

Execution of Spark Jar File: Example

This example illustrates how to execute a Spark jar file with the spark-submit command. Before submitting the jar, users should ensure that the application has a total of 8 cores.

Run the application locally on 8 cores

```
./bin/spark-submit \
  --class org.apache.spark.examples.SparkPi \
  --master local[8] \
  /path/to/example123.jar \
  100
```

Execution of Spark Jar File: Example

This example shows how to execute a Spark jar file with the spark-submit command. Before submitting the jar, it must ensure that the application is deployed in client mode. The deployment mode is specified using the –deploy-mode property.

Run on a Spark standalone cluster in client deploy mode

```
./bin/spark-submit \
  --class org.apache.spark.examples.SparkPi \
  --master spark://207.184.161.138:7077 \
  --executor-memory 20G \
  --total-executor-cores 100 \
  /path/to/examples.jar \
  1000
```

Execution of Spark Jar File: Example

This example illustrates how to run a Spark jar file with the spark-submit command. Ensure that the application is in cluster deploy mode with the supervise flag enabled before submitting the jar.

Run on a Spark standalone cluster in cluster deploy mode with supervise

```
./bin/spark-submit \
  --class org.apache.spark.examples.SparkPi \
  --master spark://207.184.161.138:7077 \
  --deploy-mode cluster \
  --supervise \
  --executor-memory 20G \
  --total-executor-cores 100 \
  /path/to/examples.jar \
  1000
```

Execution of Spark Jar File: Example

This example shows how to execute a Spark jar file with the spark-submit command. While submitting the jar, users must ensure that the application is running on a YARN cluster in cluster deploy mode.

Run on a YARN cluster in cluster deploy mode

```
export HADOOP_CONF_DIR=XXX
./bin/spark-submit \
  --class org.apache.spark.examples.SparkPi \
  --master yarn \
  --deploy-mode cluster \
  --executor-memory 20G \
  --num-executors 50 \
  /path/to/examples.jar \
  1000
```

Execution of Spark Jar File: Example

This example illustrates how to run a spark jar file with the spark-submit command. One must ensure that the application is submitted on a Spark standalone cluster while submitting the jar.

Run a Python application on a Spark standalone cluster

```
./bin/spark-submit \
  --master spark://207.184.161.138:7077 \
  examples/src/main/python/pi.py \
  1000
```

Execution of Spark Jar File: Example

This case illustrates how to execute a spark jar file with the spark submit command. While submitting the jar, users must ensure that the application is running on a Mesos cluster in cluster deploy mode with the supervise flag.

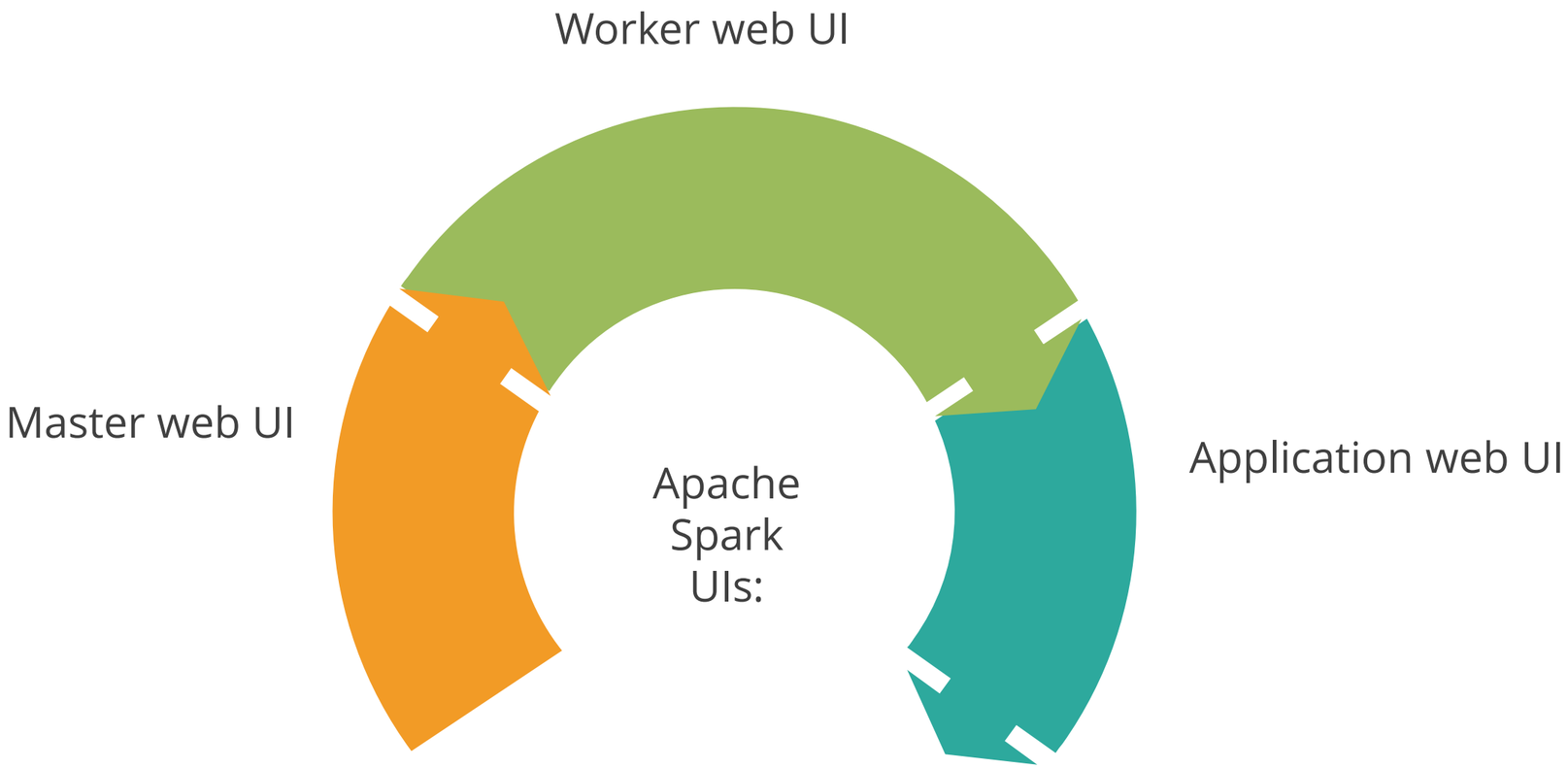
Run on a Mesos cluster in cluster deploy mode with supervise

```
./bin/spark-submit \
  --class org.apache.spark.examples.SparkPi \
  --master mesos://207.184.161.138:7077 \
  --deploy-mode cluster \
  --supervise \
  --executor-memory 20G \
  --total-executor-cores 100 \
  http://path/to/examples.jar \
  1000
```

Spark Web UI

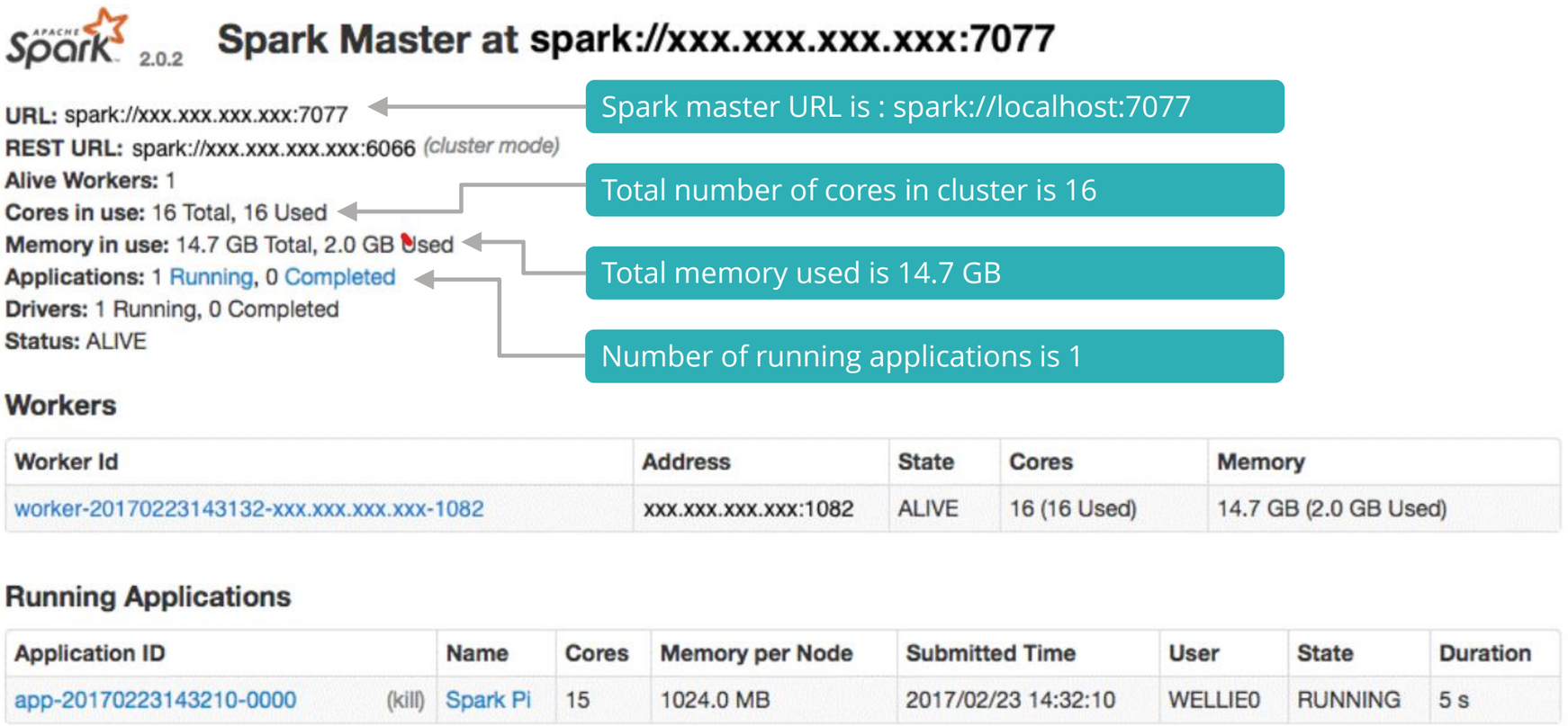
Spark Web UI

Apache Spark provides several web user interfaces that can be used to monitor the status and resource consumption of a Spark cluster.



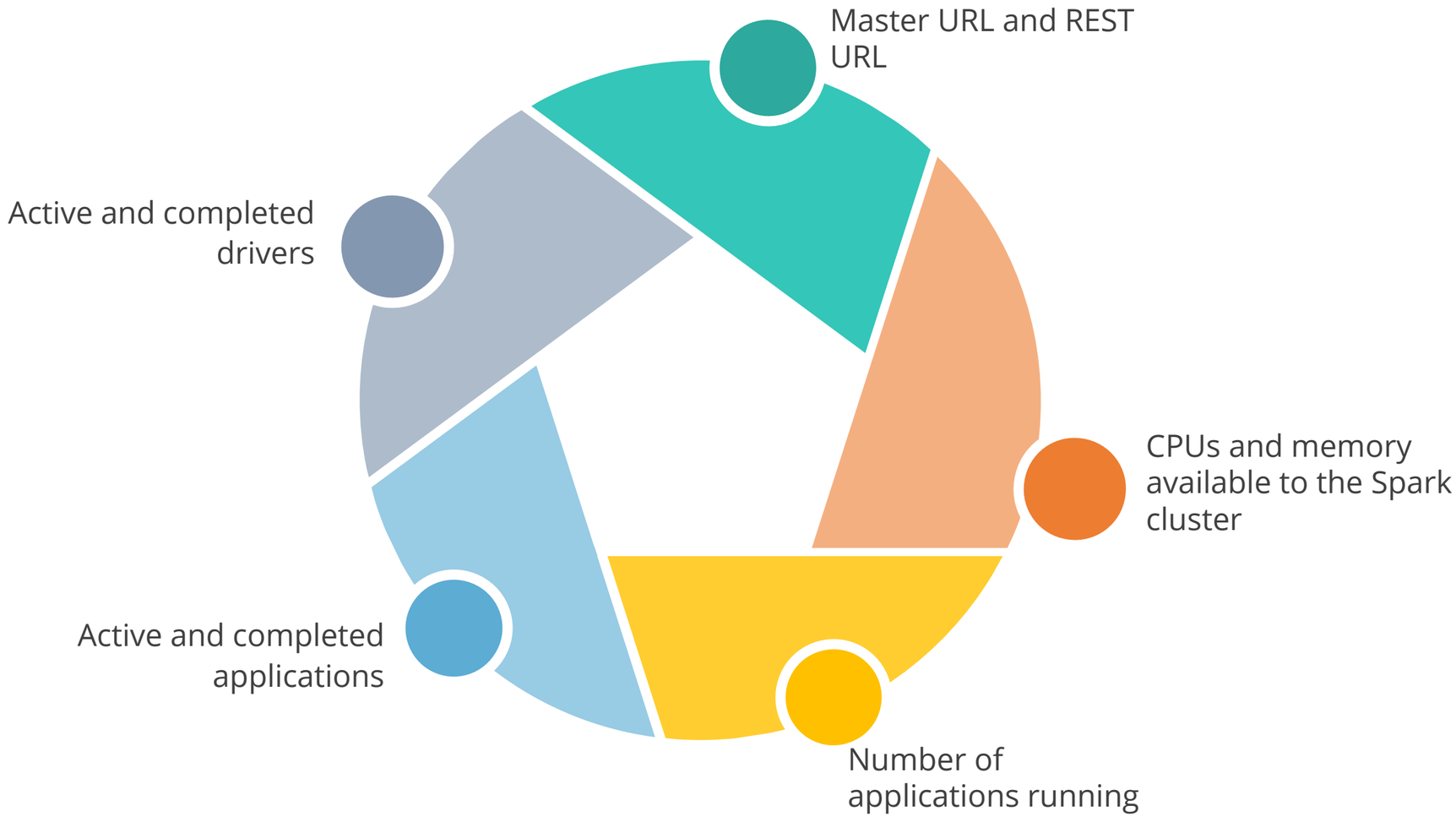
Master Web UI

It is used to monitor CPU and memory resources that are allotted to the Spark cluster and for each application.



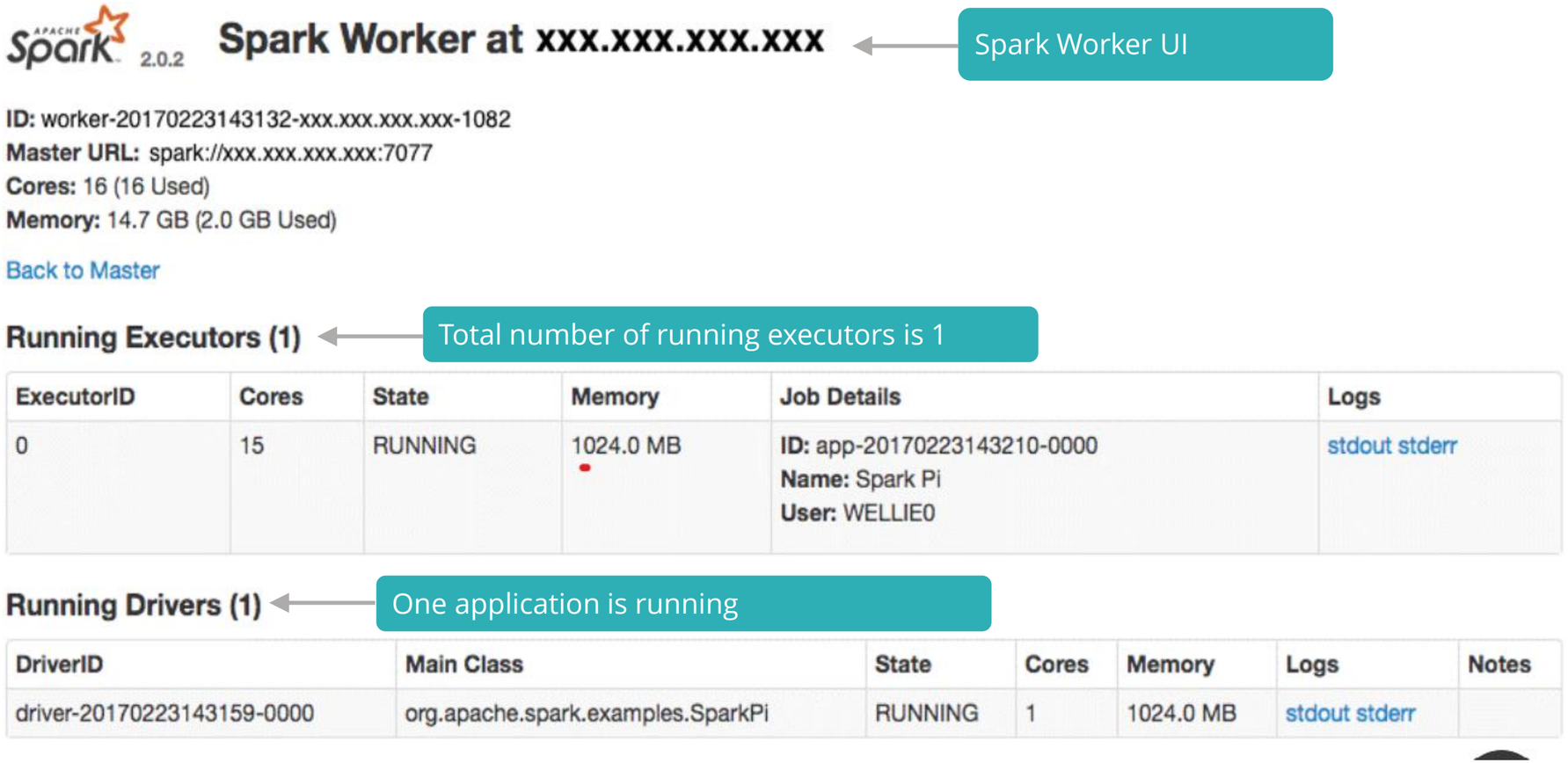
Master Web UI

It provides an overview of the Spark cluster and displays the following information:



Worker Web UI

The Worker Web UI provides information about the executors and applications.



Worker Web UI

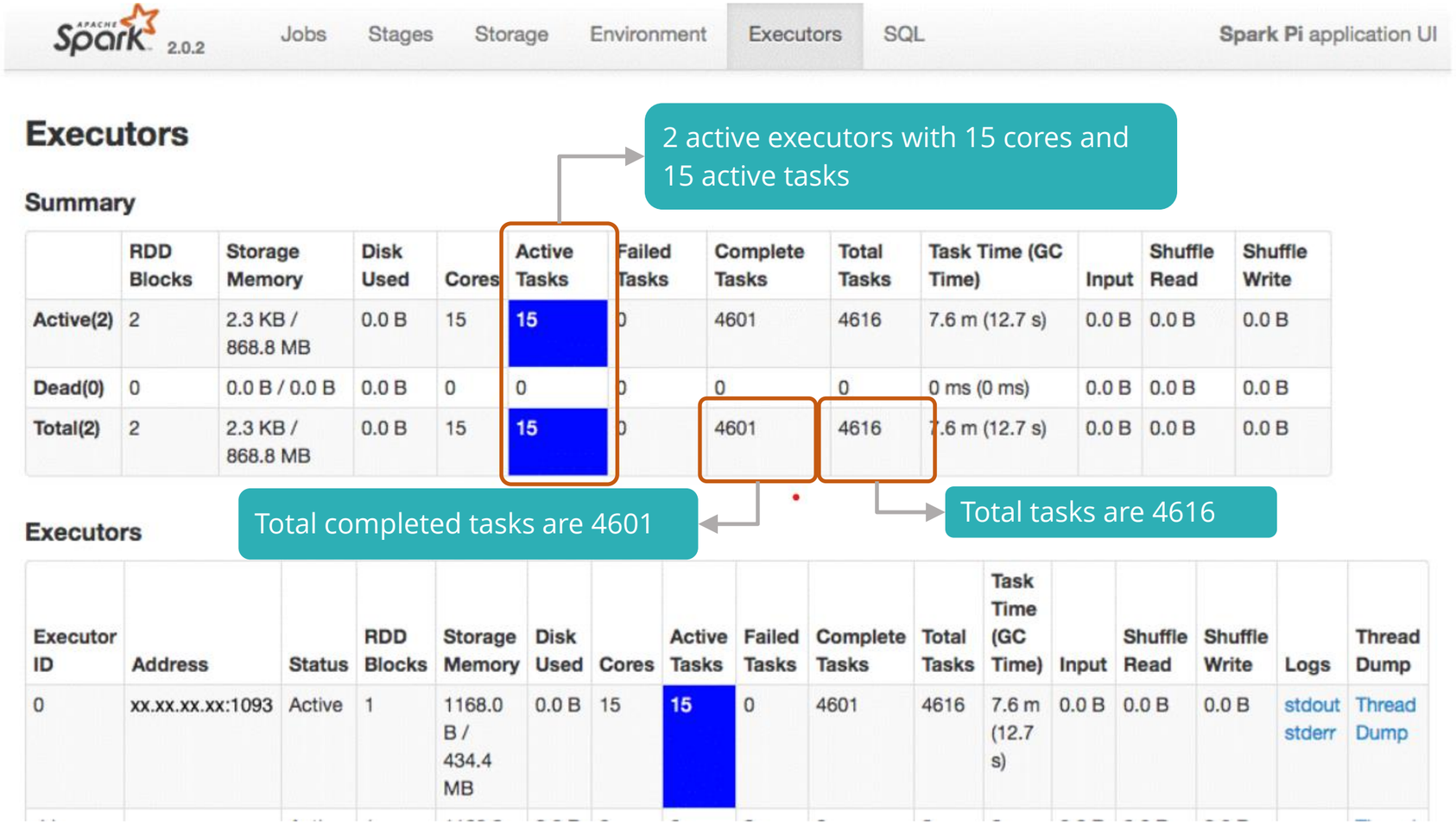
The Worker Web UI provides an overview of the executors and drivers that are spawned by the worker process.

The Web UI can see the number of executors that are currently running, and the number of resources allotted to them.

It displays the status of the allotted resources.

Application Web UI

The Application Web UI provides information about the Executors.



	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write
Active(2)	2	2.3 KB / 868.8 MB	0.0 B	15	15	0	4601	4616	7.6 m (12.7 s)	0.0 B	0.0 B	0.0 B
Dead(0)	0	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B	0.0 B
Total(2)	2	2.3 KB / 868.8 MB	0.0 B	15	15	0	4601	4616	7.6 m (12.7 s)	0.0 B	0.0 B	0.0 B

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Logs	Thread Dump
0	xx.xx.xx.xx:1093	Active	1	1168.0 B / 434.4 MB	0.0 B	15	15	0	4601	4616	7.6 m (12.7 s)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump

Application Web UI

1. Each Spark application launches its instance of the web UI.

2. The Application Web UI provides a wealth of information about the Spark application and can be a useful tool to debug the application.

Assisted Practice: Deployment of PySpark Job

Duration: 10 Minutes

ASSISTED PRACTICE

Problem Scenario:

Andrew, employed as a data engineer at a bank, has been tasked with evaluating the performance of a Spark application in cluster and client modes. He is asked to submit the same job in both modes and conduct a comparative analysis of the outcomes.

Objective: To showcase how to deploy a PySpark job effectively in client and cluster modes

Assisted Practice: Deployment of PySpark Job

Duration: 10 Minutes

Steps Overview:

Step 1: Create a Python file in the Web desktop using vi editor

Step 2: Import the required libraries and create a Spark Session to initialize the code

Step 3: Submit the job in **client mode** using the **spark-submit --deploy-mode client map.py** command

Step 4: Submit the job in **cluster mode** using the **spark-submit --deploy-mode cluster map.py** command

Note: The solution to this assisted practice is provided under the Reference Materials section.

ASSISTED PRACTICE

Key Takeaways

- Spark supports popular development languages like Java, Scala, R, and Python.
- In client mode, the Spark driver runs on the host from where the job is submitted.
- In cluster mode, the Spark Driver runs in the Application Master on a cluster machine.
- Apache Spark provides several web user interfaces that we can use to monitor the status and resource consumption of the Spark cluster.

Knowledge Check

Knowledge Check

1

Which of the following are the components of the Spark project?

A. Spark Core and RDDs

B. Spark SQL

C. Spark Streaming

D. All of the above

Knowledge Check 1

Which of the following are the components of the Spark project?

A. Spark Core and RDDs

B. Spark SQL

C. Spark Streaming

D. All of the above

The correct answer is **D**

Spark Core and RDDs, Spark SQL, and Spark Streaming are some of the components of the Spark project.

Knowledge Check 2

The Spark driver runs on the host from where the job is submitted. Which mode is being referred to?

A. Client Mode

B. Cluster Mode

C. Standalone Mode

D. Mesos

Knowledge Check 2

The Spark driver runs on the host from where the job is submitted. Which mode is being referred to?

A. Client Mode

B. Cluster Mode

C. Standalone Mode

D. Mesos

The correct answer is **A**

The driver is launched directly within the spark-submit process in client mode, which acts as a client to the cluster. The input and output of the application are attached to the console.

Knowledge Check

3

Which of the following are the supported Cluster Managers?

A. Standalone

B. Apache Mesos

C. Hadoop Yarn

D. All of the above

Knowledge Check

3

Which of the following are the supported Cluster Managers?

A. Standalone

B. Apache Mesos

C. Hadoop Yarn

D. All of the above

The correct answer is **D**

Standalone, Apache Mesos, and Hadoop Yarn are all supported Cluster Managers.

Knowledge Check 4

Assume you want to run a Spark job on your production server to collect data from all bike stations with more than 10 vehicles at any given moment. This job will be sent to YARN via:

A. Spark-shell console

B. SparkR way

C. Jar file using spark-submit

D. Executable file via spark-shell

Knowledge Check 4

Assume you want to run a Spark job on your production server to collect data from all bike stations with more than 10 vehicles at any given moment. This job will be sent to YARN via:

A. Spark-shell console

B. SparkR way

C. Jar file using spark-submit

D. Executable file via spark-shell

The correct answer is **C**

The spark-submit script in Spark's bin directory is used to launch applications on a cluster. It can use all the Spark's supported cluster managers through a uniform interface so each application need not be configured individually.

Thank You