

# Running Spark on a Cluster

---

# Spark Cluster Options

---

## Spark can run

- Locally
  - No distributed processing
- Locally with multiple worker threads
- On a cluster
  - Spark Standalone
  - Apache Hadoop YARN (Yet Another Resource Negotiator)
  - Apache Mesos

# Why Run on a Cluster?

---

 **Run Spark on a cluster to get the advantages of distributed processing**

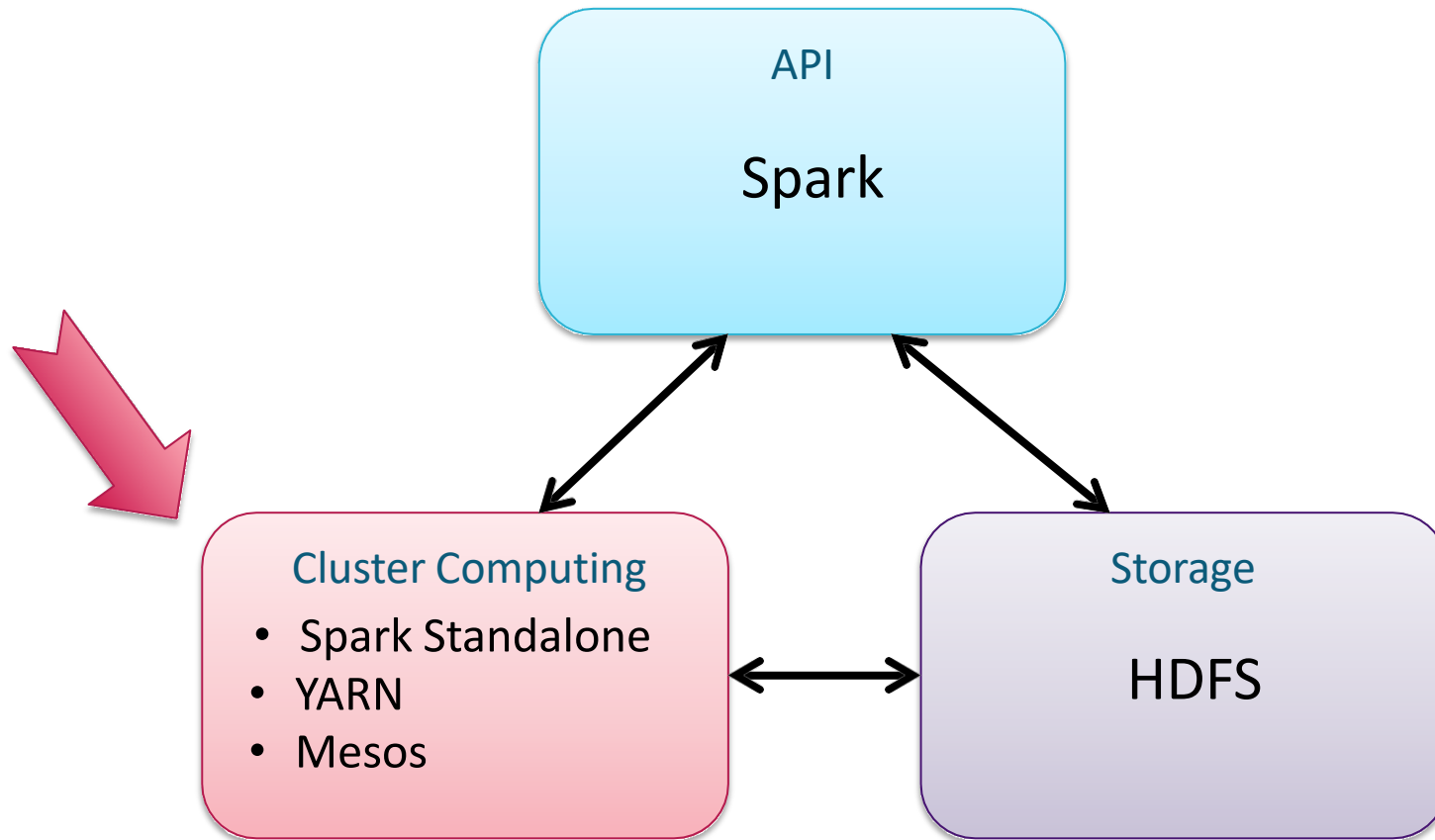
- Ability to process large amounts of data efficiently
- Fault tolerance and scalability

 **Local mode is useful for development and testing**

 **Production use is almost always on a cluster**

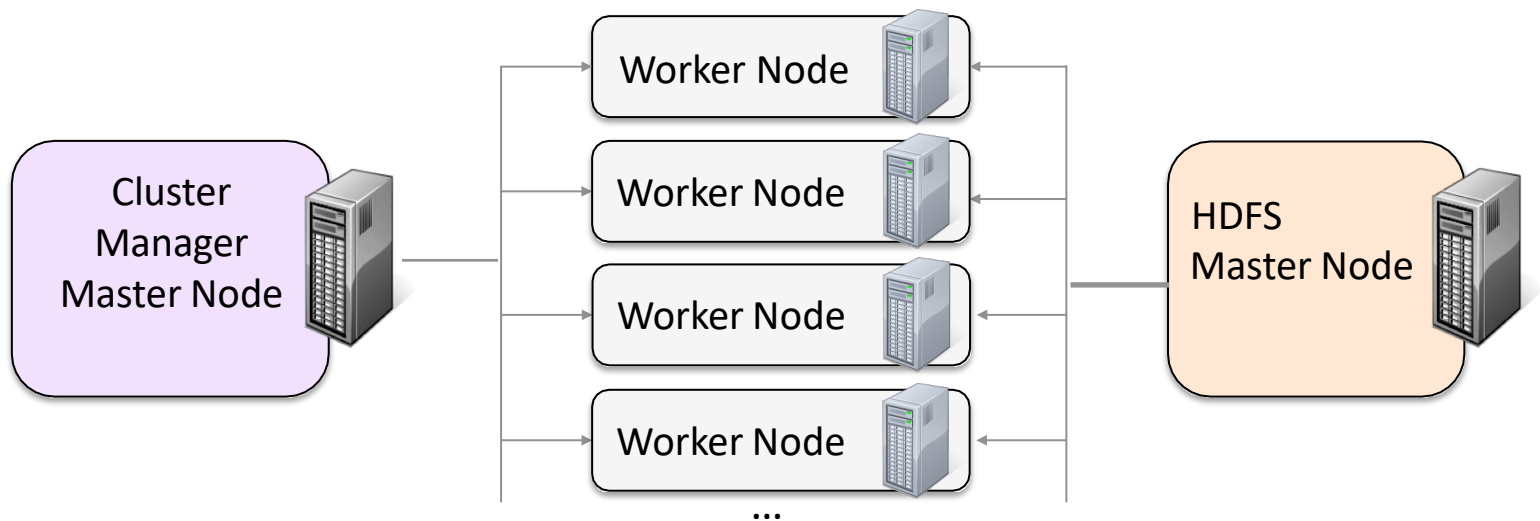
# Distributed Processing with the Spark Framework

---



# Spark Cluster Terminology

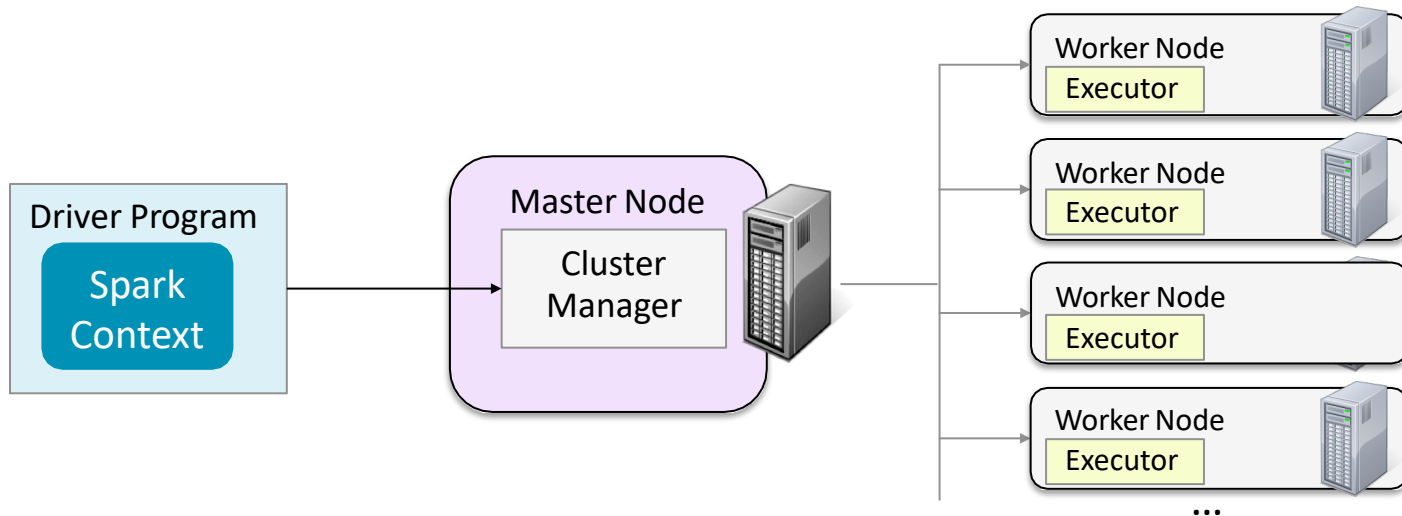
- ✎ **A cluster is a group of computers working together**
  - Usually runs HDFS in addition to Spark Standalone, YARN, or Mesos
- ✎ **A node is an individual computer in the cluster**
  - *Master* nodes manage distribution of work and data to *worker* nodes
- ✎ **A daemon is a program running on a node**
  - Each performs different functions in the cluster



# The Spark Driver Program

## A Spark Driver

- The “main” program
  - Either the Spark Shell or a Spark application
- Creates a Spark Context configured for the cluster
- Communicates with Cluster Manager to distribute tasks to executors



# Starting the Spark Shell on a Cluster

---

## Set the Spark Shell master to

- **url** – the URL of the cluster manager
- **local[\*]** – run with as many threads as cores (default)
- **local[n]** – run locally with *n* worker threads
- **local** – run locally without distributed processing

## This configures the `SparkContext.master` property

Python

```
$ MASTER=spark://masternode:7077 pyspark
```

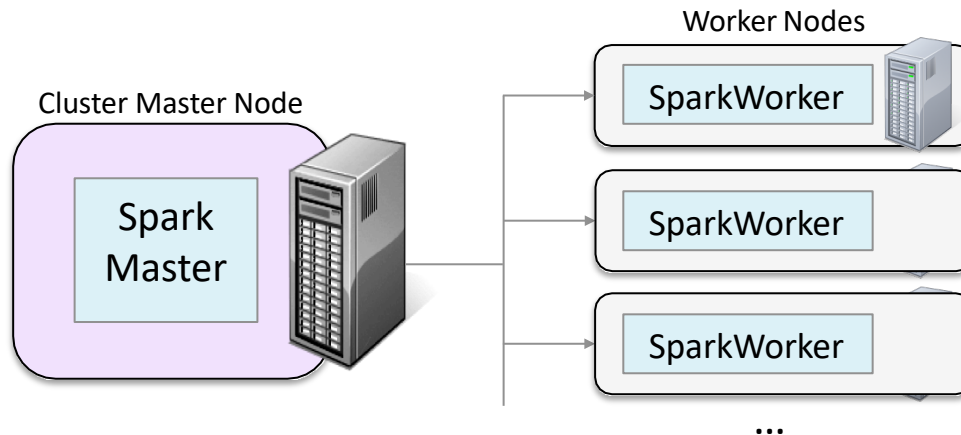
Scala

```
$ spark-shell --master spark://masternode:7077
```

# Spark Standalone Daemons

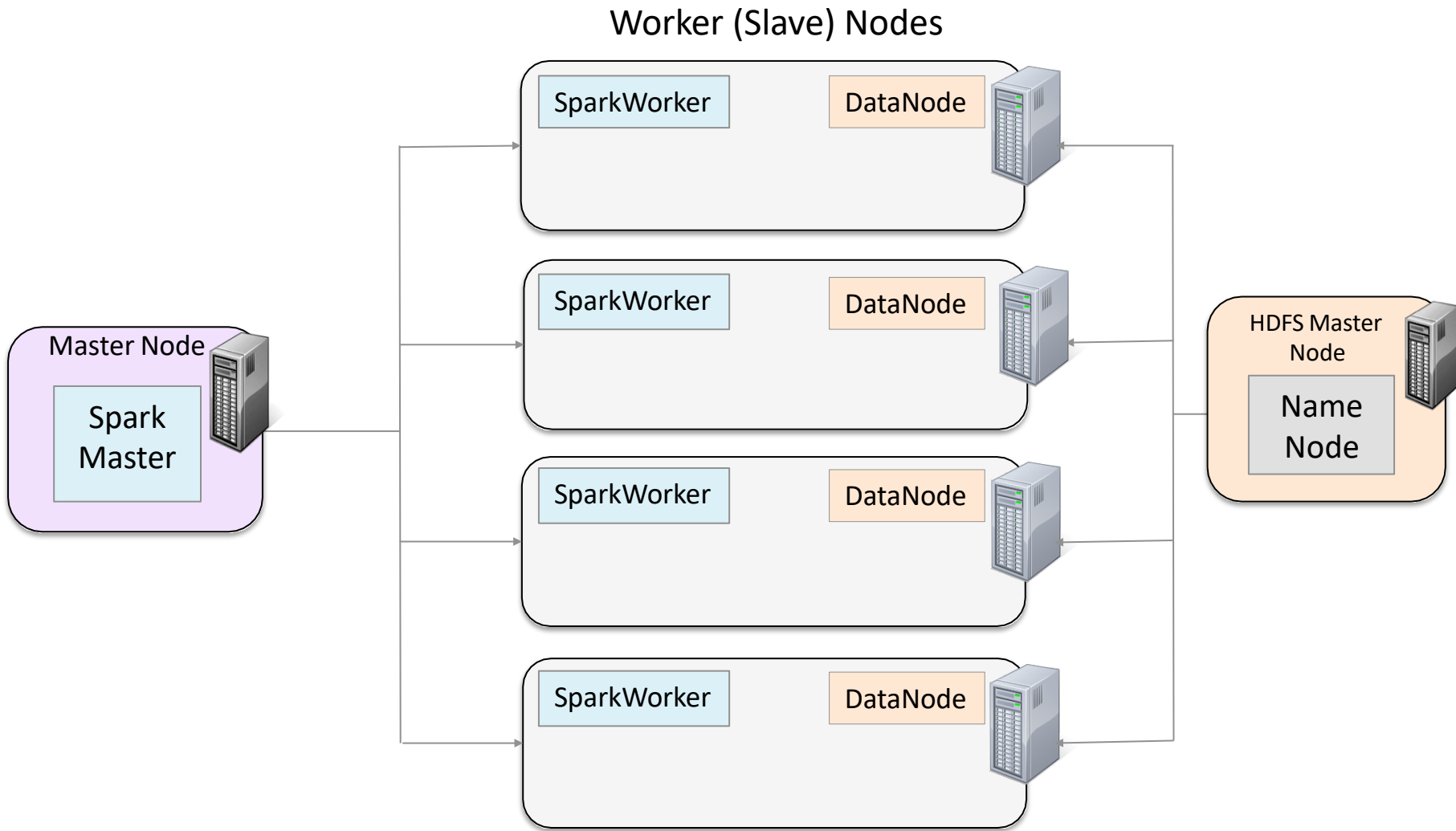
## Spark Standalone daemons

- Spark Master
  - One per cluster
  - Manages applications, distributes individual tasks to Spark Workers
- Spark Worker
  - One per worker node
  - Starts and monitors Executors for applications

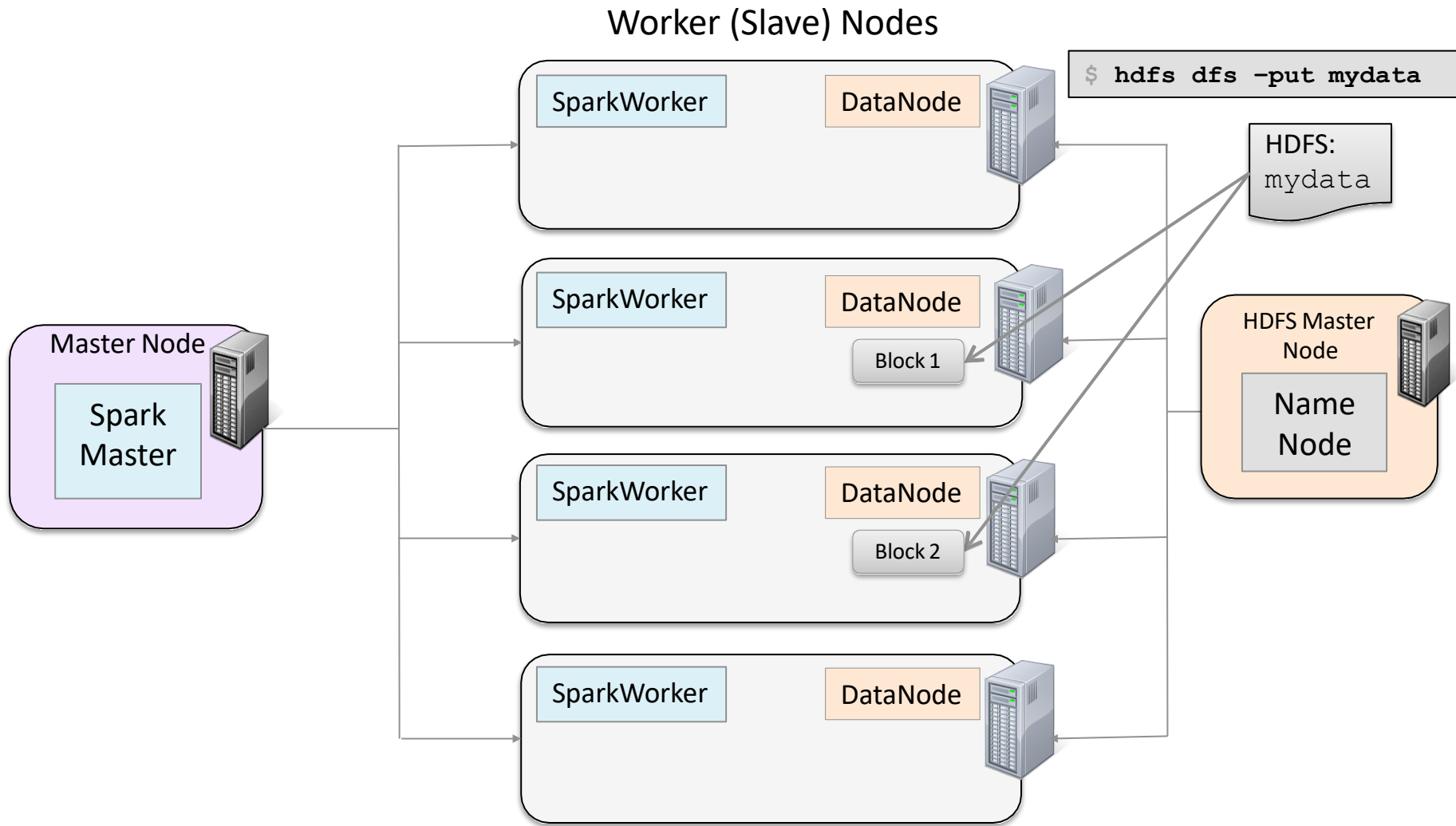




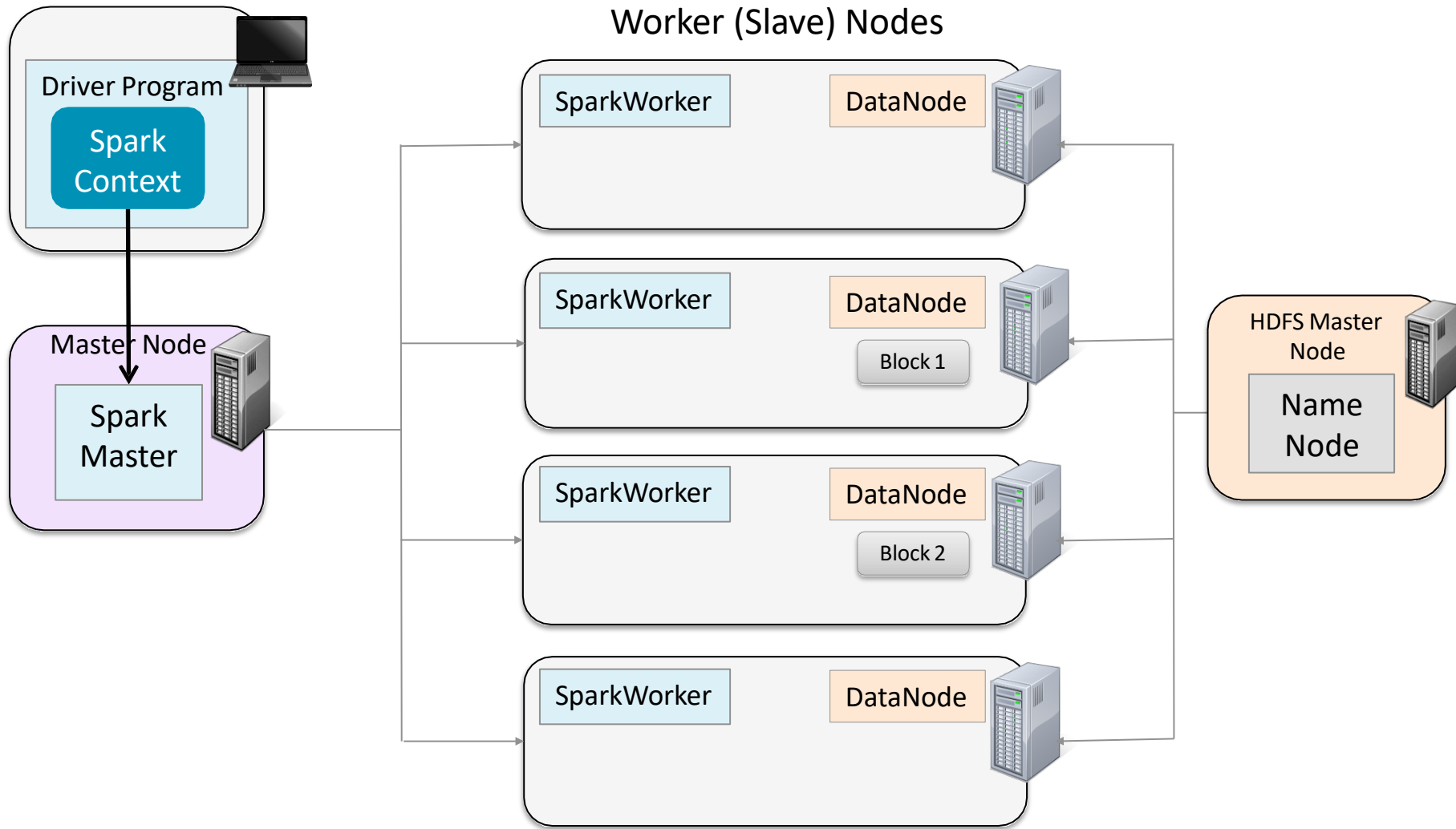
# Running Spark on a Standalone Cluster (1)



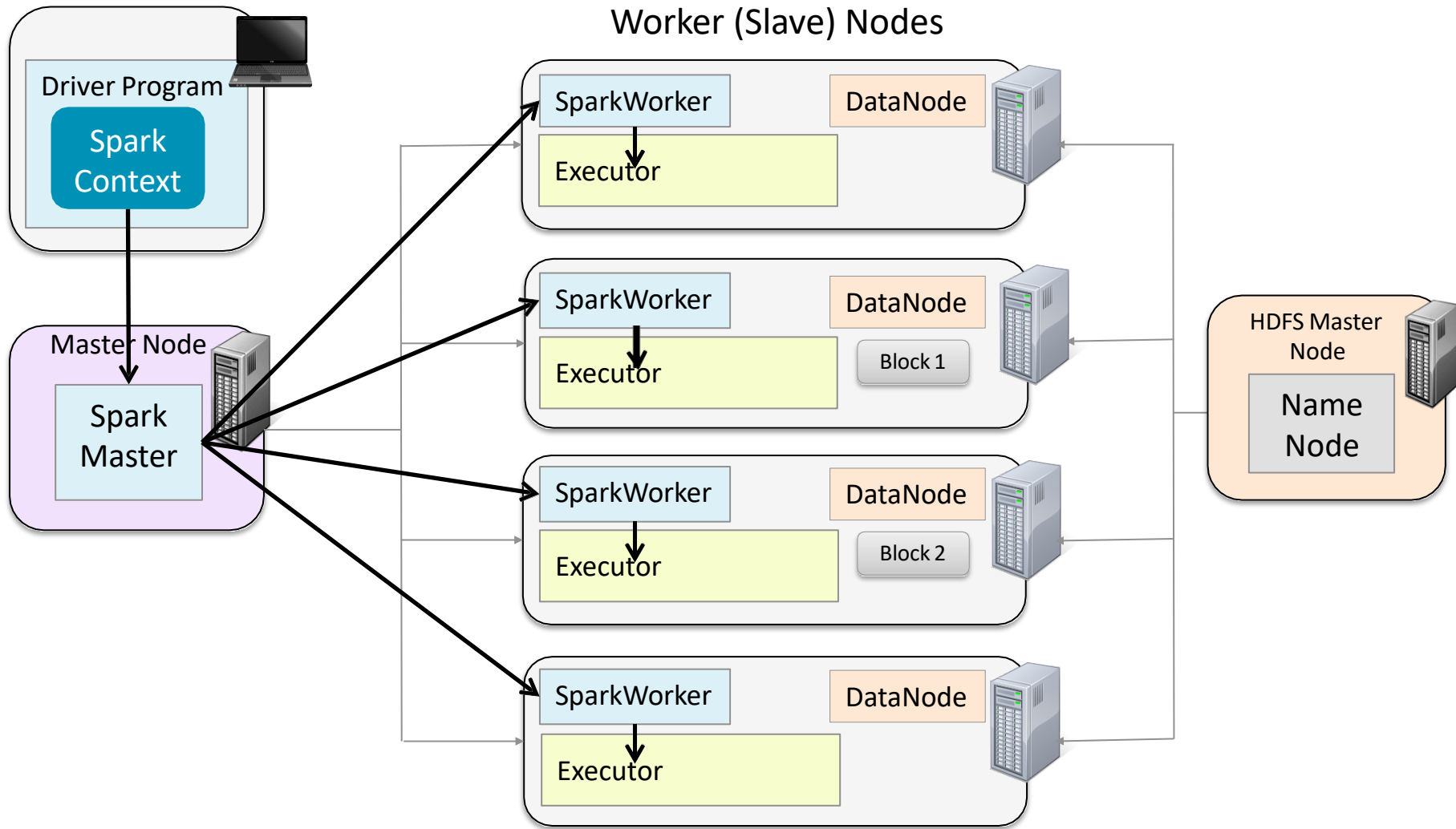
## Running Spark on a Standalone Cluster (2)



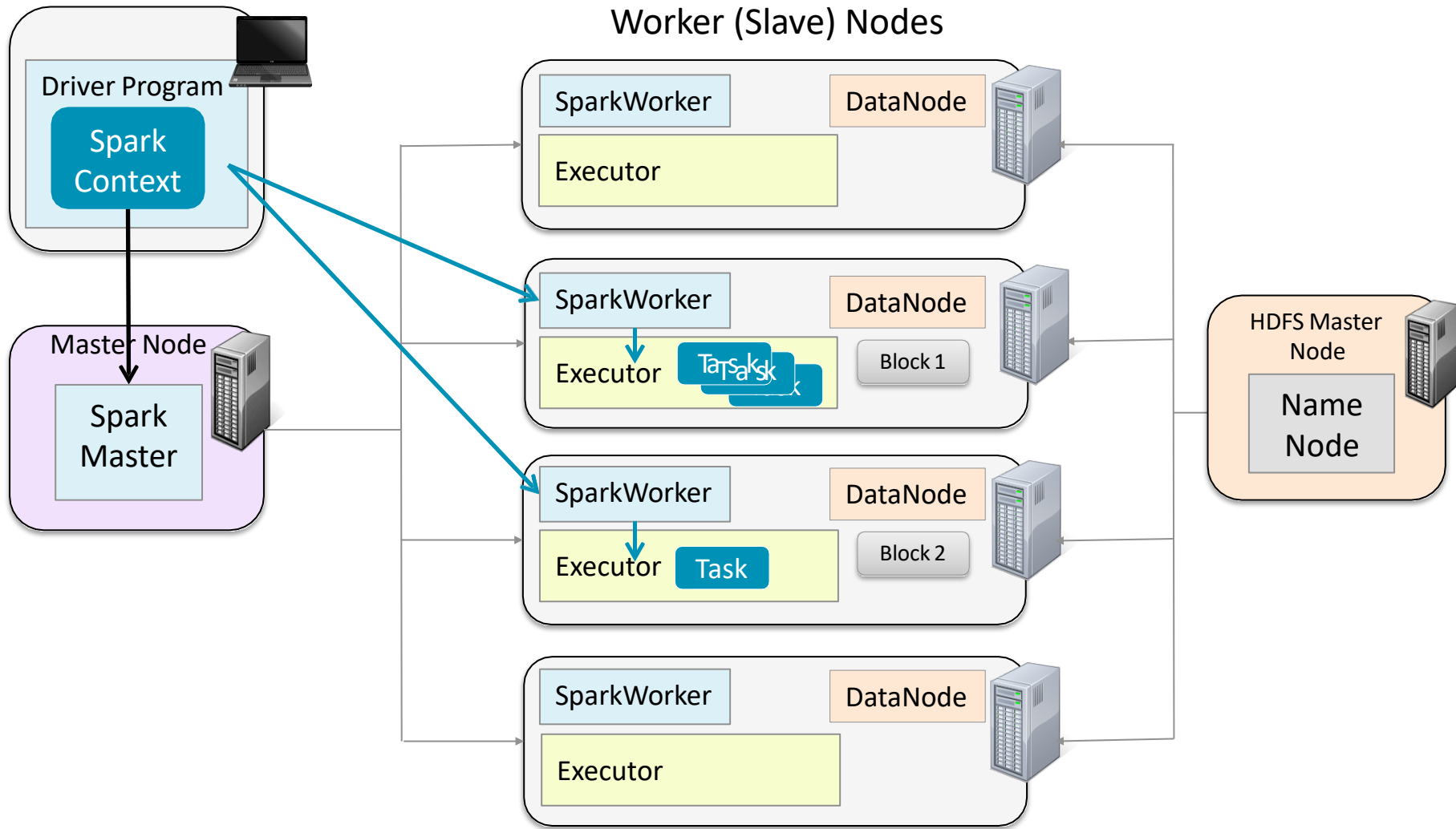
## Running Spark on a Standalone Cluster (3)



## Running Spark on a Standalone Cluster (4)



# Running Spark on a Standalone Cluster (5)



# Spark Standalone Web UI

- ✍ Spark Standalone clusters offer a Web UI to monitor the cluster
  - `http://masternode:uiport`
    - e.g., in our class environment, `http://localhost:18080`

The screenshot displays the Spark Master Web UI for a standalone cluster. At the top, it shows the Spark logo and the master URL: `spark://ec2-23-20-24-104.compute-1.amazonaws.com:7077`. Below this, summary statistics are provided: URL, 5 workers, 20 total/used cores, 68.2 GB total/63.2 GB used memory, and 1 running/2 completed applications. The 'Workers' section contains a table of 5 worker nodes, all in an 'ALIVE' state with 4 cores and 13.6 GB memory. The 'Running Applications' section shows one application named 'PageRank' in a 'RUNNING' state. The 'Completed Applications' section shows two finished applications: 'SparkPi' and 'Spark shell'. Three callout boxes highlight key features: 'Worker Nodes' points to the worker table, 'Master URL' points to the top URL, and 'Applications' points to the completed applications table.

**Spark Master at `spark://ec2-23-20-24-104.compute-1.amazonaws.com:7077`**

URL: `spark://ec2-23-20-24-104.compute-1.amazonaws.com:7077`  
Workers: 5  
Cores: 20 Total, 20 Used  
Memory: 68.2 GB Total, 63.2 GB Used  
Applications: 1 Running, 2 Completed

**Workers**

Id	Address	State	Cores	Memory
<a href="#">worker-20140121065745-ip-10-236-129-42.ec2.internal-60105</a>	<code>ip-10-236-129-42.ec2.internal:60105</code>	ALIVE	4 (4 Used)	13.6 GB (12.6 GB Used)
<a href="#">worker-20140121065747-ip-10-137-18-53.ec2.internal-54087</a>	<code>ip-10-137-18-53.ec2.internal:54087</code>	ALIVE	4 (4 Used)	13.6 GB (12.6 GB Used)
<a href="#">worker-20140121065747-ip-10-138-3-46.ec2.internal-50661</a>	<code>ip-10-138-3-46.ec2.internal:50661</code>	ALIVE	4 (4 Used)	13.6 GB (12.6 GB Used)
<a href="#">worker-20140121065747-ip-10-236-151-85.ec2.internal-60016</a>	<code>ip-10-236-151-85.ec2.internal:60016</code>	ALIVE	4 (4 Used)	13.6 GB (12.6 GB Used)
<a href="#">worker-20140121065748-ip-10-238-128-41.ec2.internal-42252</a>	<code>ip-10-238-128-41.ec2.internal:42252</code>	ALIVE	4 (4 Used)	13.6 GB (12.6 GB Used)

**Running Applications**

ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
<a href="#">app-20140121215958-0002</a>	<a href="#">PageRank</a>	20	12.6 GB	2014/01/21 21:59:58	root	RUNNING	13 s

**Completed Applications**

ID	Name	Cores	Memory	Submitted Time	User	State	Duration
<a href="#">app-20140121215522-0001</a>	<a href="#">SparkPi</a>	20	12.6 GB	2014/01/21 21:55:22	root	FINISHED	10 s
<a href="#">app-20140121215016-0000</a>	<a href="#">Spark shell</a>	20	12.6 GB	2014/01/21 21:50:16	root	FINISHED	1.2 min

# Spark Standalone Web UI: Application Overview

**Running Applications**

ID	Name	Cores
<a href="#">app-20140121215958-0002</a>	<a href="#">PageRank</a>	20

**Completed Applications**

**Link to Spark Application UI**

**Spark** **Application: PageRank**

ID: app-20140121220952-0006  
Name: PageRank  
User: root  
Cores: Unlimited (20 granted)  
Executor Memory: 12.6 GB  
Submit Date: Tue Jan 21 22:09:52 UTC 2014  
State: RUNNING  
[Application Detail UI](#)

**Executors for this application**


**Executor Summary**

ExecutorID	Worker	Cores	Memory	State	Logs
3	<a href="#">worker-20140121065747-ip-10-138-3-46.ec2.internal-50661</a>	4	12936	RUNNING	<a href="#">stdout stderr</a>
4	<a href="#">worker-20140121065745-ip-10-236-129-42.ec2.internal-60105</a>	4	12936	RUNNING	<a href="#">stdout stderr</a>
1	<a href="#">worker-20140121065748-ip-10-238-128-41.ec2.internal-42252</a>	4	12936	RUNNING	<a href="#">stdout stderr</a>
2	<a href="#">worker-20140121065747-ip-10-236-151-85.ec2.internal-60016</a>	4	12936	RUNNING	<a href="#">stdout stderr</a>
0	<a href="#">worker-20140121065747-ip-10-137-18-53.ec2.internal-54087</a>	4	12936	RUNNING	<a href="#">stdout stderr</a>

# Spark Standalone Web UI: Worker Detail

**Workers**

Id	Address
<a href="#">worker-20140121065745-ip-10-236-129-42.ec2.internal-60105</a>	ip-10-236-129-42.ec2.internal:60105
<a href="#">worker-20140121065747-ip-10-137-18-53.ec2.internal-54087</a>	ip-10-137-18-53.ec2.internal:54087
<a href="#">worker-20140121065747-ip-10-138-3-46.ec2.internal-50661</a>	ip-10-138-3-46.ec2.internal:50661

 **Spark Worker at ip-10-236-129-42.ec2.internal:60105**

**ID:** worker-20140121065745-ip-10-236-129-42.ec2.internal-60105  
**Master URL:** spark://ec2-23-20-24-104.compute-1.amazonaws.com:7077  
**Cores:** 4 (4 Used)  
**Memory:** 13.6 GB (12.6 GB Used)  
[Back to Master](#)

**Running Executors 1**

ExecutorID	Cores	Memory	Job Details	Logs
4	4	12.6 GB	<b>ID:</b> app-20140121220135-0003 <b>Name:</b> PageRank <b>User:</b> root	<a href="#">stdout stderr</a>

**Finished Executors**

4	4	12.6 GB	<b>ID:</b> app-20140121215522-0001 <b>Name:</b> SparkPi <b>User:</b> root	<a href="#">stdout stderr</a>
4	4	12.6 GB	<b>ID:</b> app-20140121215958-0002 <b>Name:</b> PageRank <b>User:</b> root	<a href="#">stdout stderr</a>

**All executors on this node**

**Log files**



# Supported Cluster Resource Managers

---

## **Spark Standalone**

- Included with Spark
- Easy to install and run
- Limited configurability and scalability
- Useful for testing, development, or small systems

## **Hadoop YARN**

- Included in CDH
- Most common for production sites
- Allows sharing cluster resources with other applications (MapReduce, Impala, etc.)

## **Apache Mesos**

- First platform supported by Spark
- Now used less often

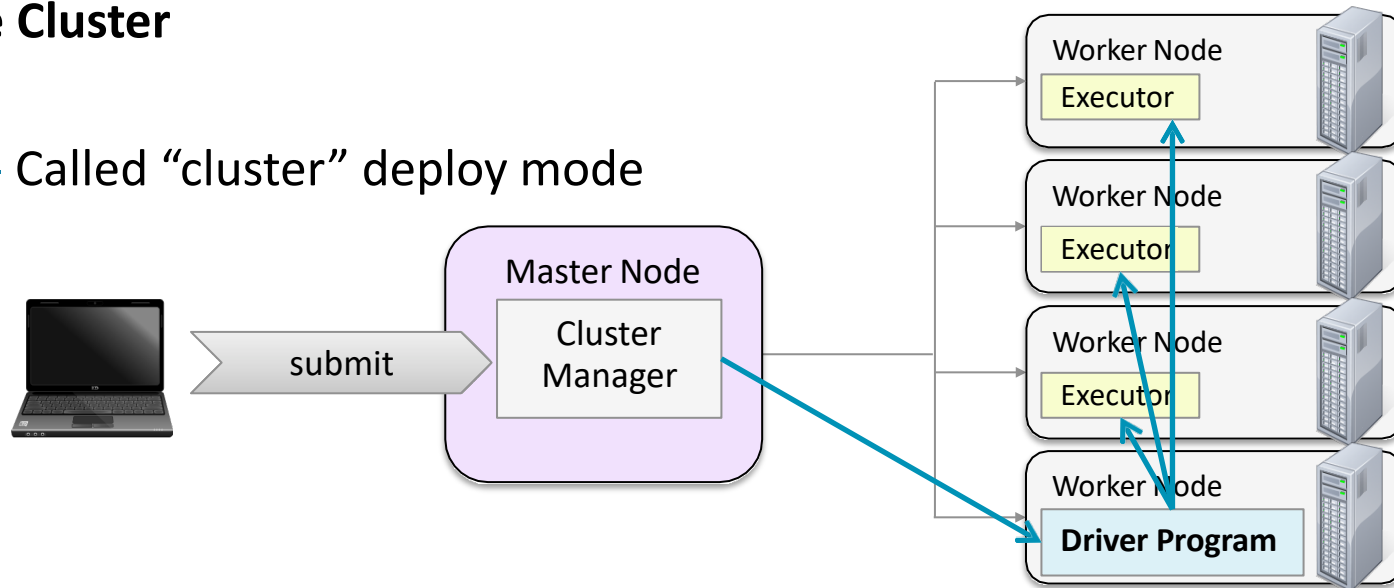
# Client Mode and Cluster Mode

 **By default, the driver program runs outside the cluster**

- Called “client” deploy mode
- Most common
- Required for interactive use (e.g., the Spark Shell)

 **It is also possible to run the driver program on a worker node in the Cluster**

- Called “cluster” deploy mode



# Installing a Spark Cluster (2)

---

## **Difficult:**

- Download and install Spark and HDFS directly from Apache

## **Easier: CDH**

- Cloudera's Distribution, including Apache Hadoop
- Includes HDFS, Spark API, Spark Standalone, and YARN
- Includes many patches, backports, bug fixes

## **Easiest: Cloudera Manager**

- Wizard9based UI to install, configure, and manage a cluster
- Included with Cloudera Express (free) or Cloudera Enterprise
- Supports Spark deployment as Standalone or YARN

---

Exercise:  
Spark Two Nodes Cluster – 90 Minutes