Running Spark on a Cluster

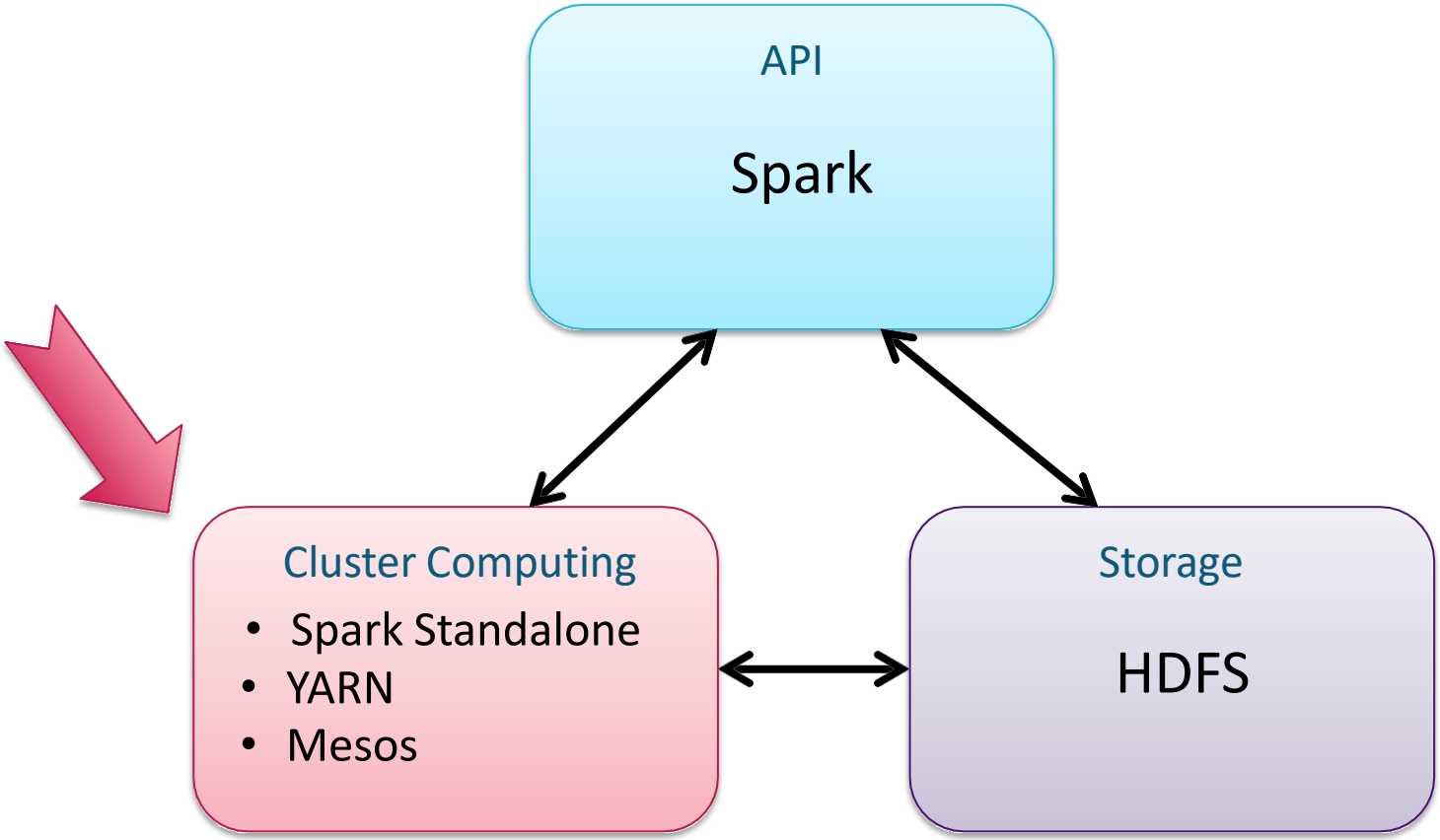
Spark Cluster Options

- **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?

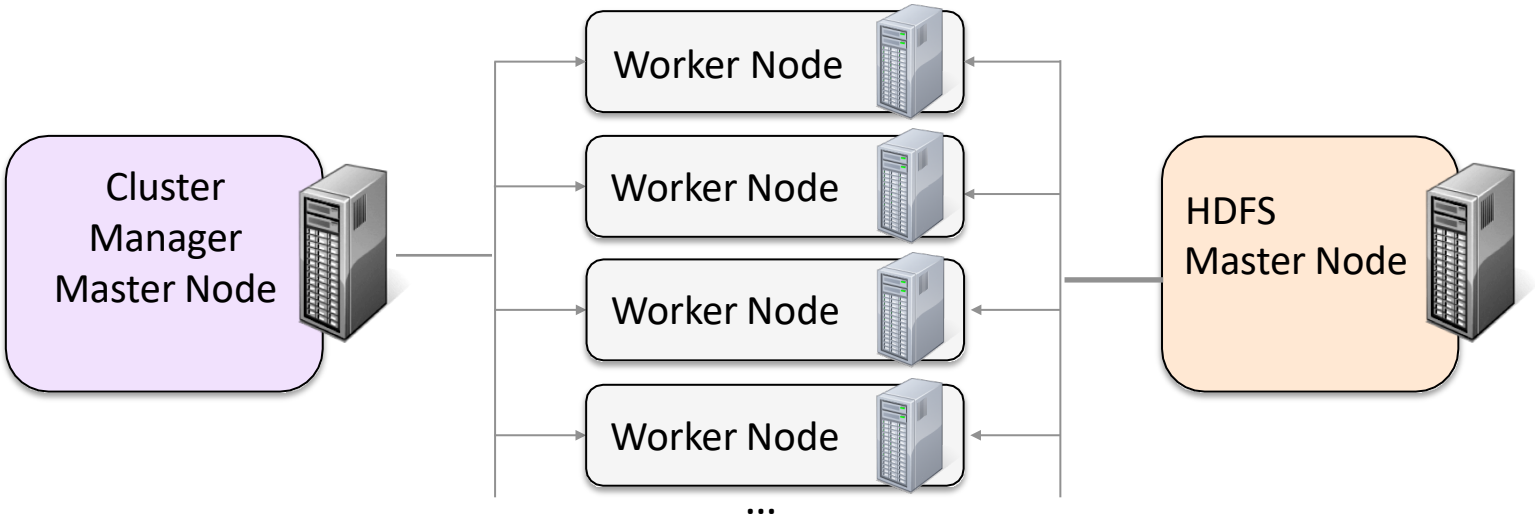
- **Run Spark on a cluster to get the advantages of distributed processing**
  - Ability to process large amounts of data efficiently
  - Fault tolerance and scalability
- **Local mode is useful for development and testing**
- **Production use is almost always on a cluster**

Distributed Processing with the Spark Framework



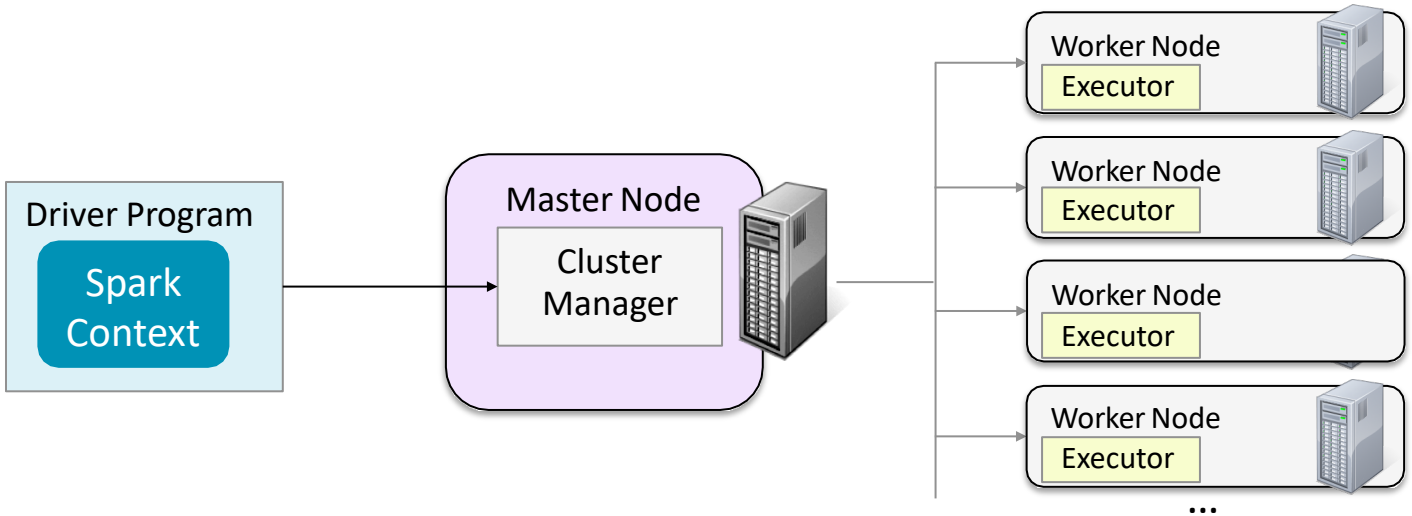
Spark Cluster Terminology

- **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

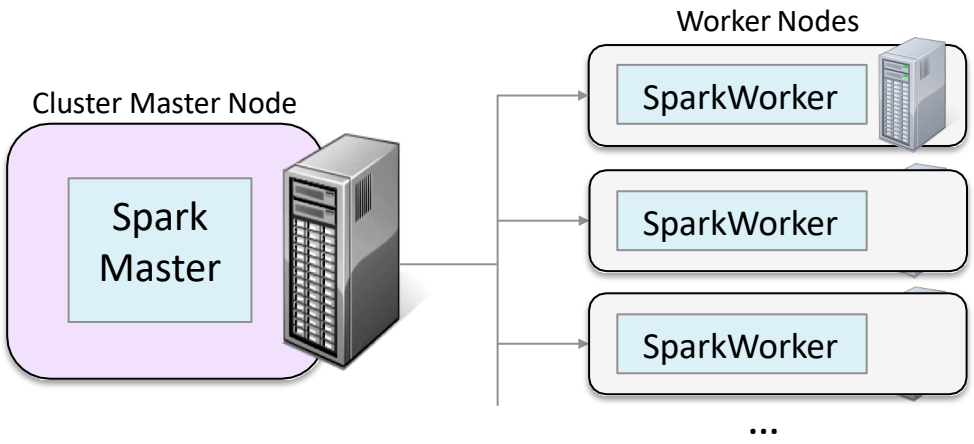
```
$ MASTER=spark://masternode:7077 pyspark
```

Scala

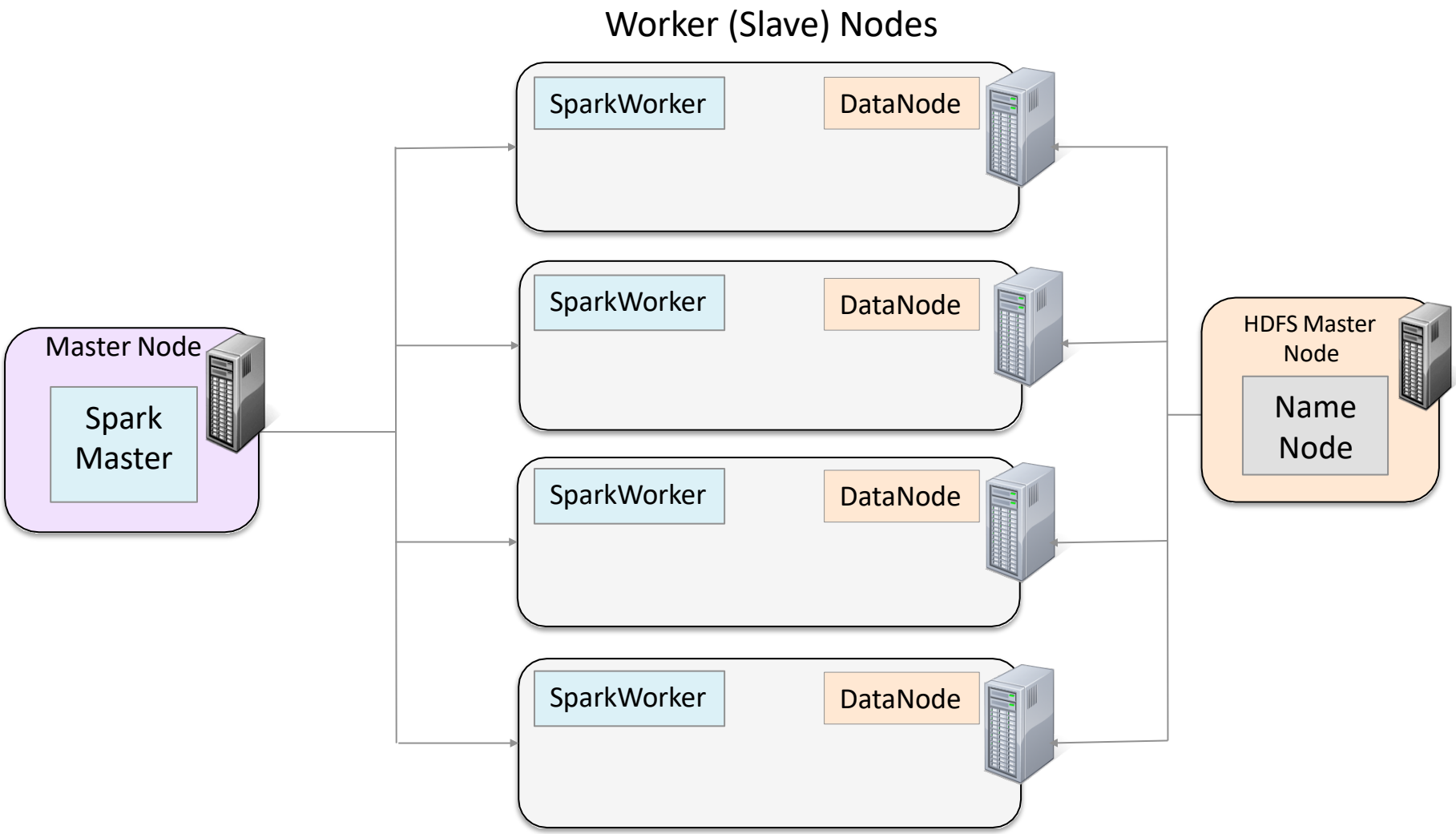
```
$ spark-shell --master spark://masternode:7077
```

Spark Standalone Daemons

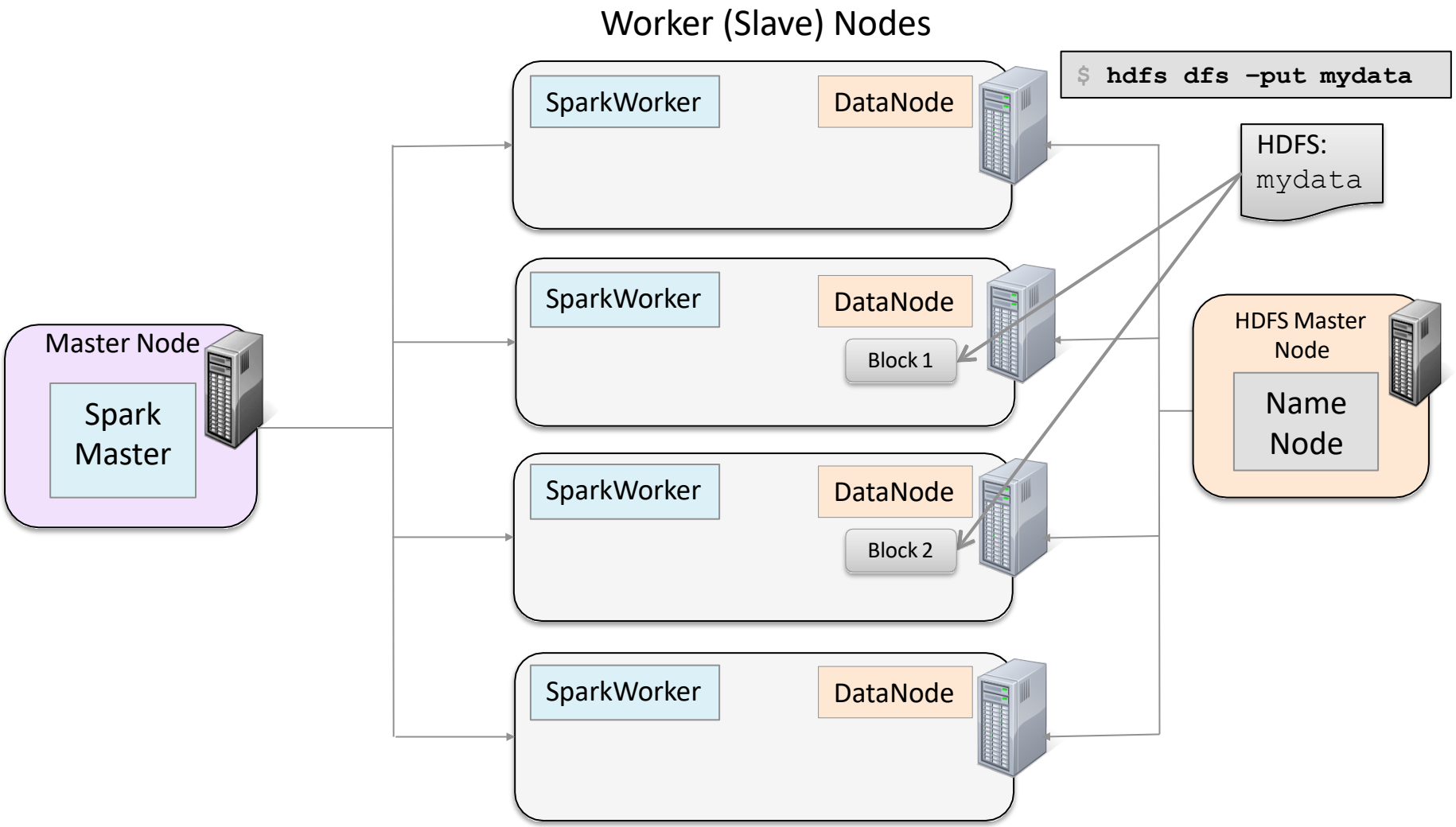
- **Spark Standalone daemons**
  - Spark Master
    - One per cluster
    - Manages applications, distributes individual tasks to Spark Workers
  - Spark Worker
    - One per worker node
    - Starts and monitors Executors for applications



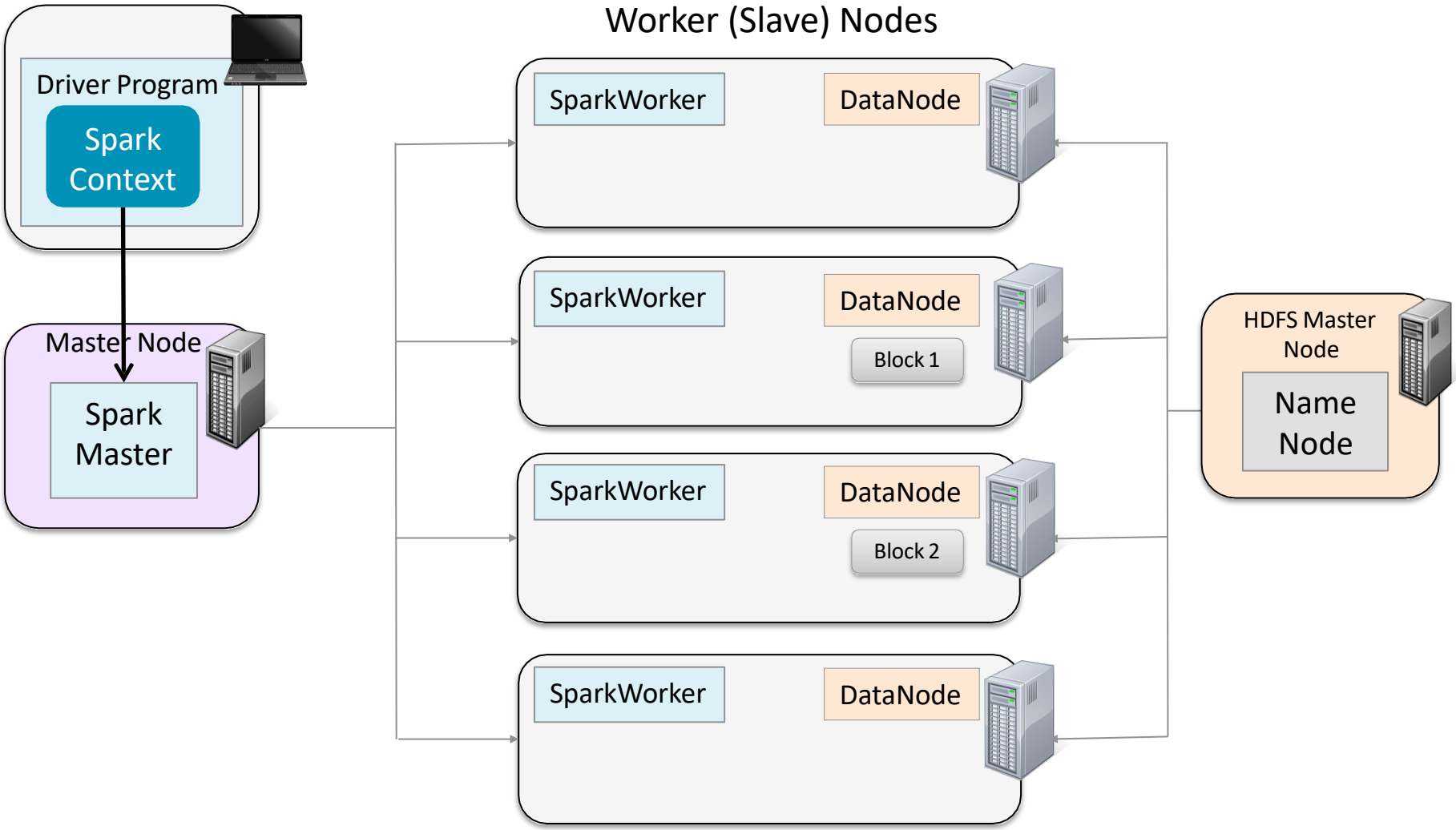
Running Spark on a Standalone Cluster (1)



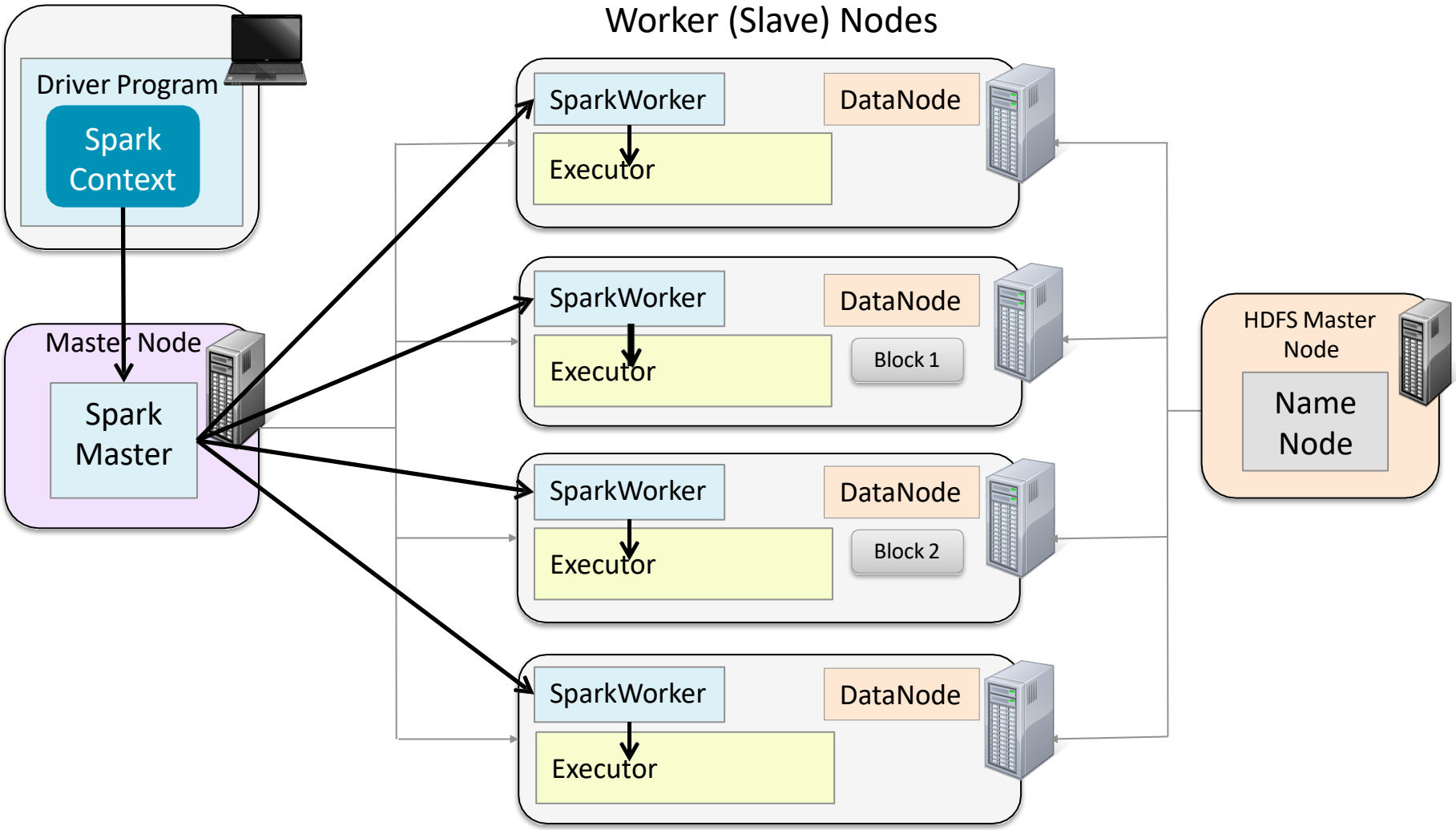
Running Spark on a Standalone Cluster (2)



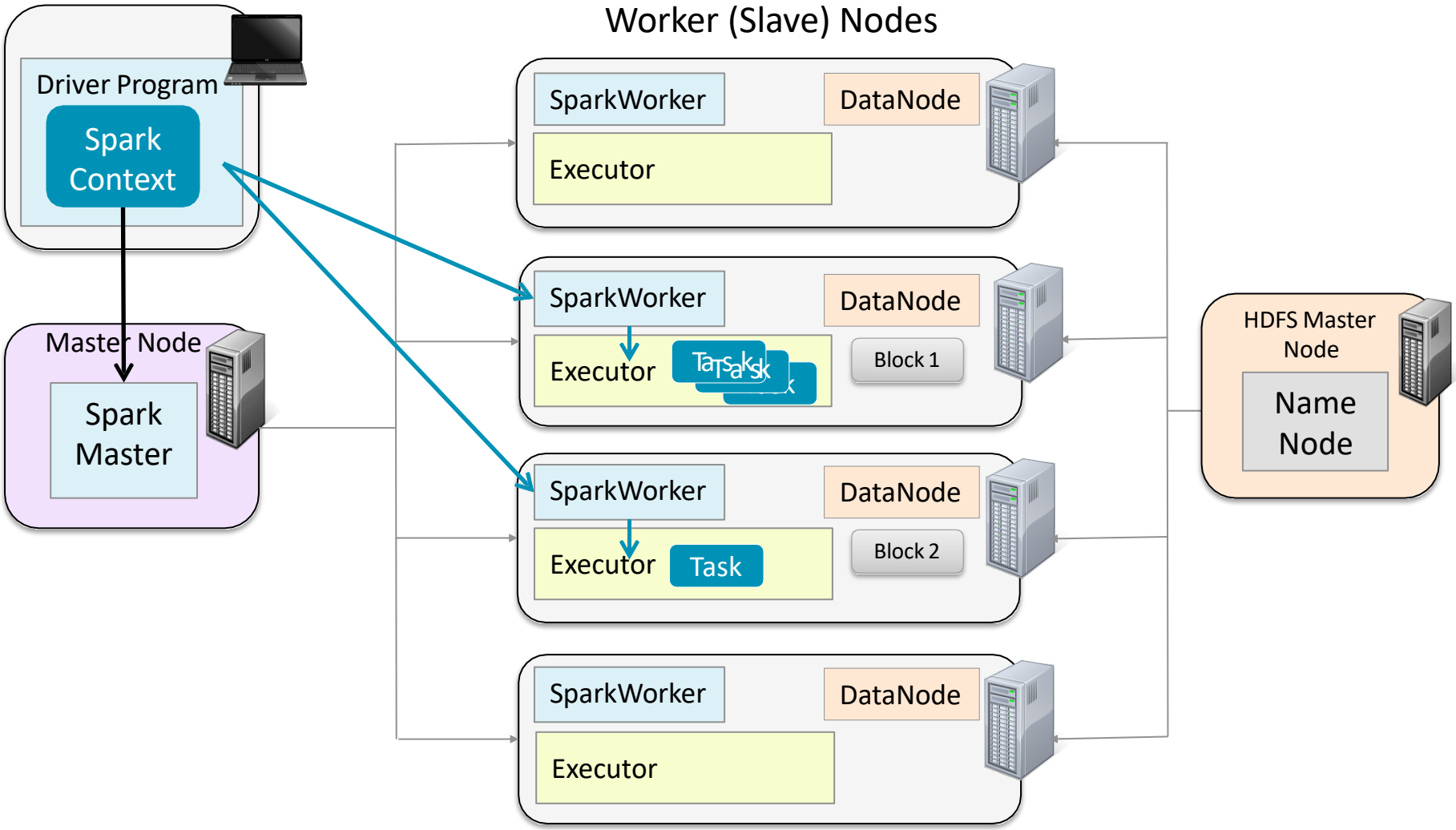
Running Spark on a Standalone Cluster (3)



Running Spark on a Standalone Cluster (4)

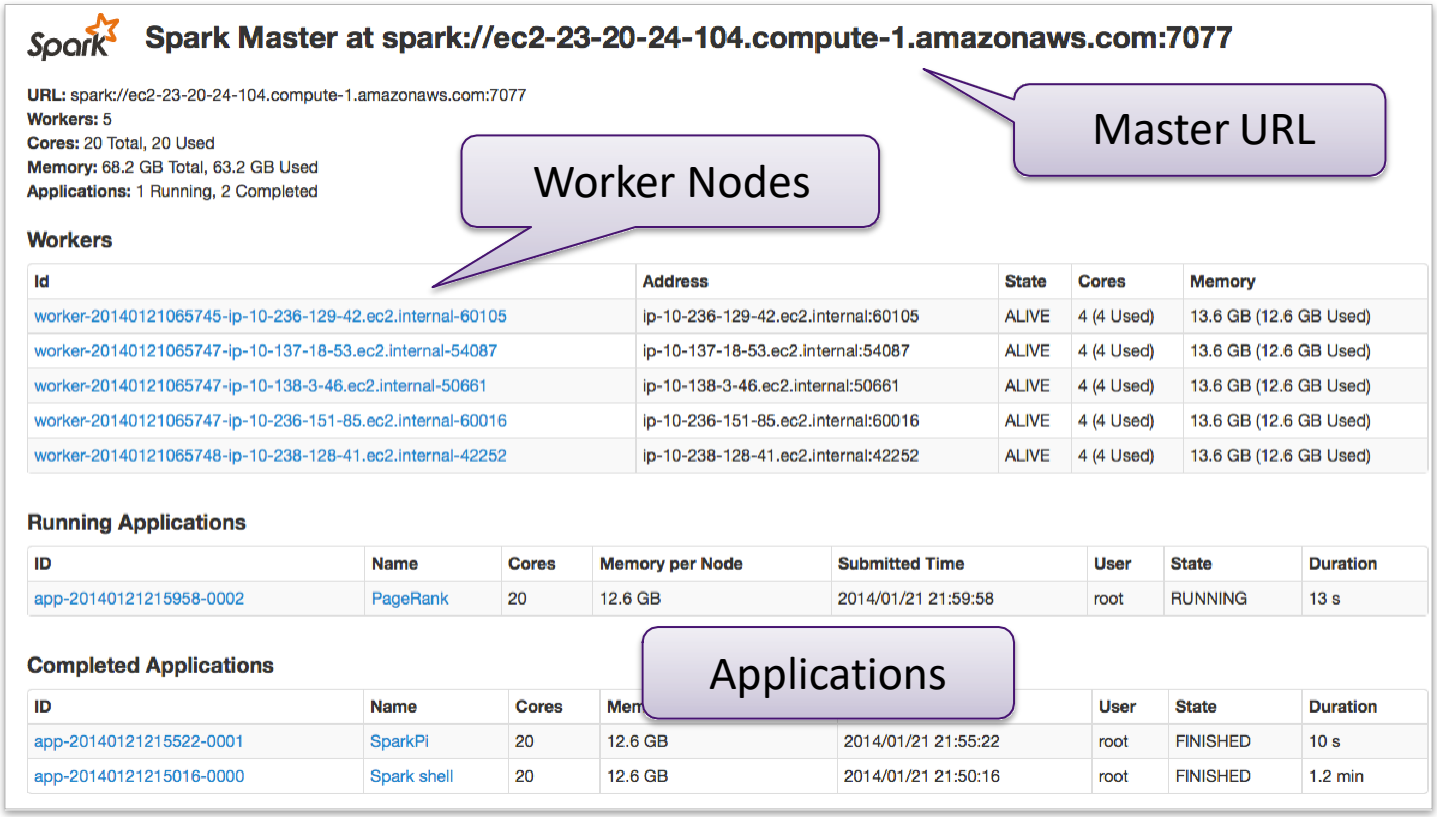


Running Spark on a Standalone Cluster (5)



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`**



Spark Standalone Web UI: Application Overview

Running Applications

ID	Name	Cores
app-20140121215958-0002	PageRank	20

Completed Applications

Link to Spark Application UI

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	worker-20140121065747-ip-10-138-3-46.ec2.internal-50661	4	12936	RUNNING	stdout stderr
4	worker-20140121065745-ip-10-236-129-42.ec2.internal-60105	4	12936	RUNNING	stdout stderr
1	worker-20140121065748-ip-10-238-128-41.ec2.internal-42252	4	12936	RUNNING	stdout stderr
2	worker-20140121065747-ip-10-236-151-85.ec2.internal-60016	4	12936	RUNNING	stdout stderr
0	worker-20140121065747-ip-10-137-18-53.ec2.internal-54087	4	12936	RUNNING	stdout stderr

Spark Standalone Web UI: Worker Detail

Workers

Id	Address
worker-20140121065745-ip-10-236-129-42.ec2.internal-60105	ip-10-236-129-42.ec2.internal:60105
worker-20140121065747-ip-10-137-18-53.ec2.internal-54087	ip-10-137-18-53.ec2.internal:54087
worker-20140121065747-ip-10-138-3-46.ec2.internal-50661	ip-10-138-3-46.ec2.internal:50661
worke	
worke	

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

All executors on this node

Log files

Running Executors 1

ExecutorID	Cores	Memory	Job Details	Logs
4	4	12.6 GB	**ID:** app-20140121220135-0003 **Name:** PageRank **User:** root	stdout stderr

Finished Executors

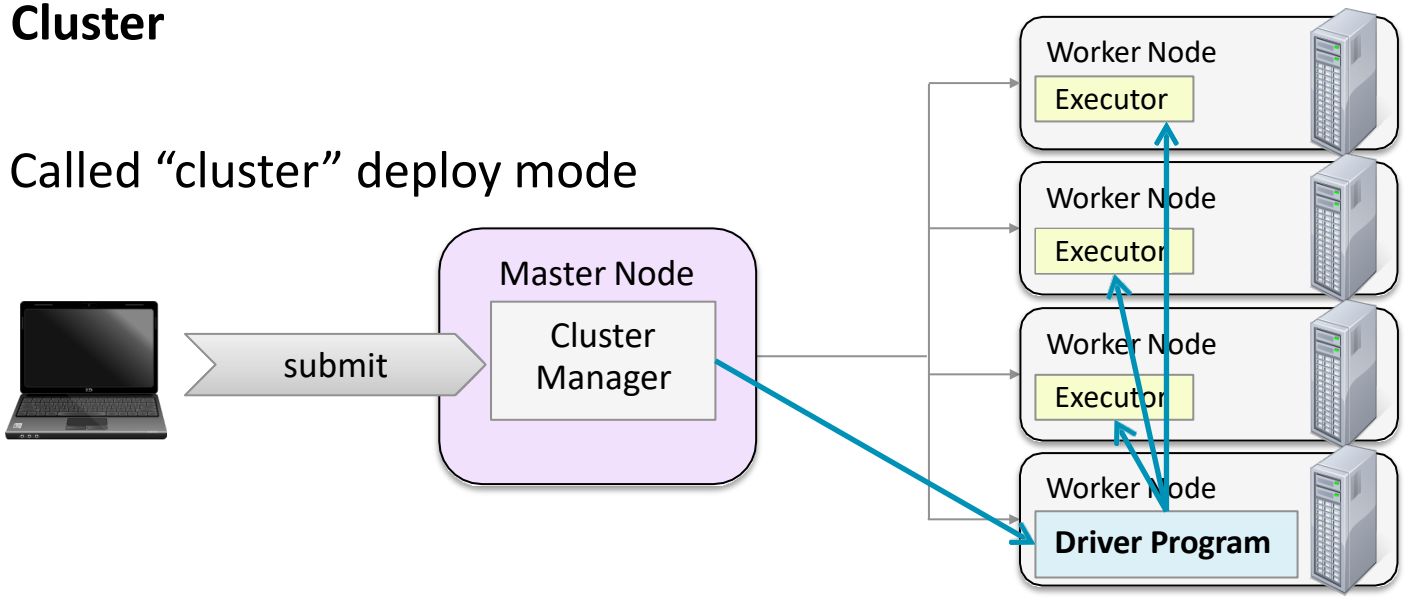
ExecutorID	Cores	Memory	Job Details	Logs
4	4	12.6 GB	**ID:** app-20140121215522-0001 **Name:** SparkPi **User:** root	stdout stderr
4	4	12.6 GB	**ID:** app-20140121215958-0002 **Name:** PageRank **User:** root	stdout stderr

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