
Introduction to Hadoop Yarn
BIG DATA
HADOOP & SPARK

What is YARN?

YARN= Yet Another Resource Negotiator

YARN is a resource manager

Created by separating the processing engine and the management function of MapReduce

Monitors and manages workloads, maintains a multi-tenant environment, manages the high availability features of Hadoop, and implements security controls

Need for YARN

Before 2012

Users could write MapReduce programs using scripting languages



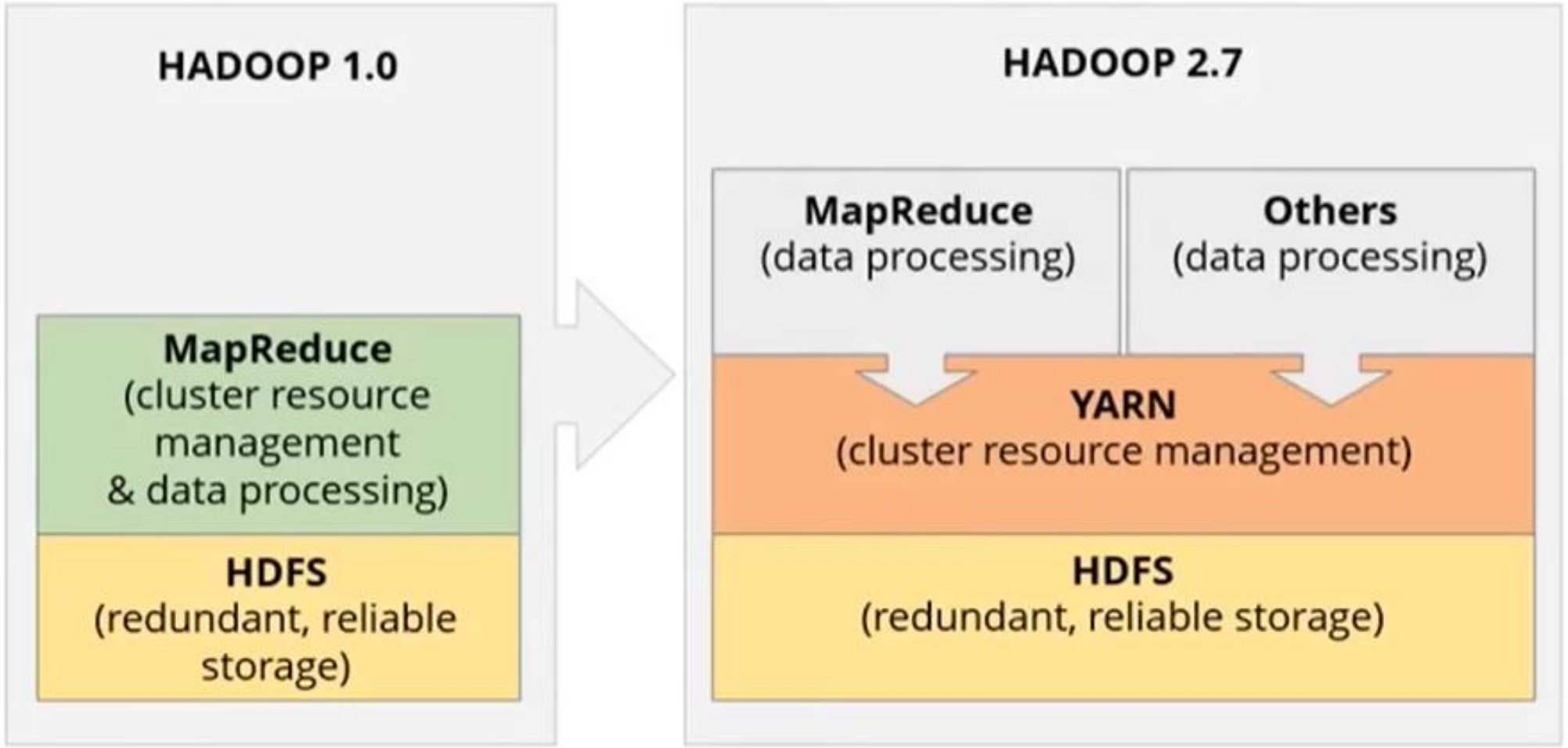
Need for YARN

Before 2012

Users could write MapReduce programs using scripting languages

Since 2012

Users could work on multiple processing models in addition to MapReduce



YARN—Use Case

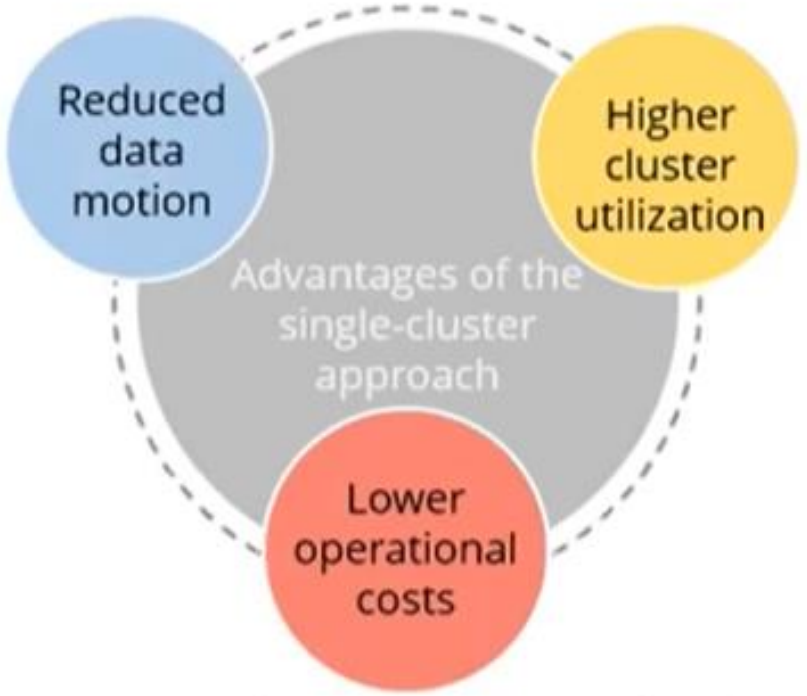
YAHOO!

- Yahoo was the first company to embrace Hadoop in a big way, and it is a trendsetter within the Hadoop ecosystem. In late 2012, it struggled to handle iterative and stream processing of data on Hadoop infrastructure due to MapReduce limitations.
- After implementing YARN in the first quarter of 2013, Yahoo has installed more than 30,000 production nodes on
  - Spark for iterative processing
  - Storm for stream processing
  - Hadoop for batch processing

YARN—Advantages

The single-cluster approach provides a number of advantages, including:

There's no need to move data between Hadoop YARN and systems running on different clusters of computers

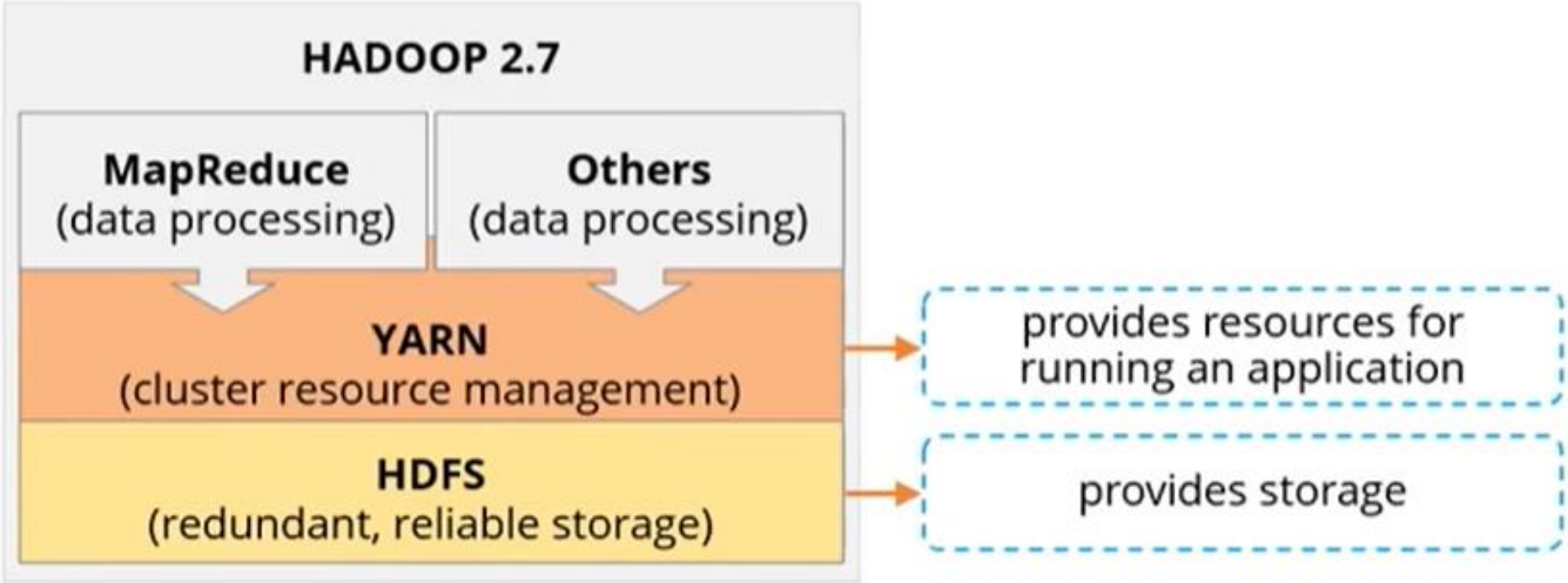


Resources unutilized by a framework can be consumed by another

Only one "do-it-all" cluster needs to be managed

YARN Infrastructure

The YARN Infrastructure is responsible for providing computational resources for application executions.



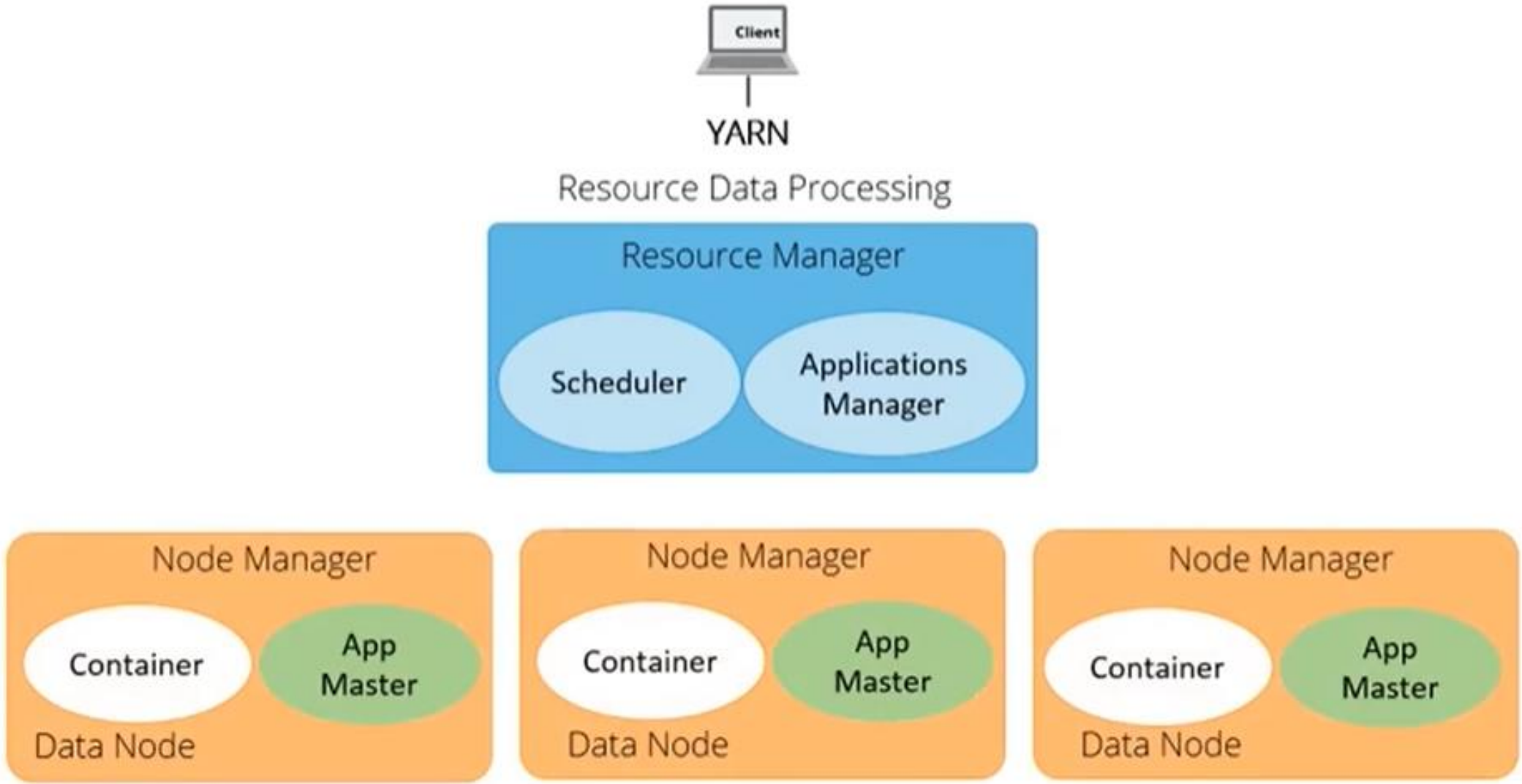
HDFS and YARN

YARN and its Architecture

BIG DATA
HADOOP & SPARK

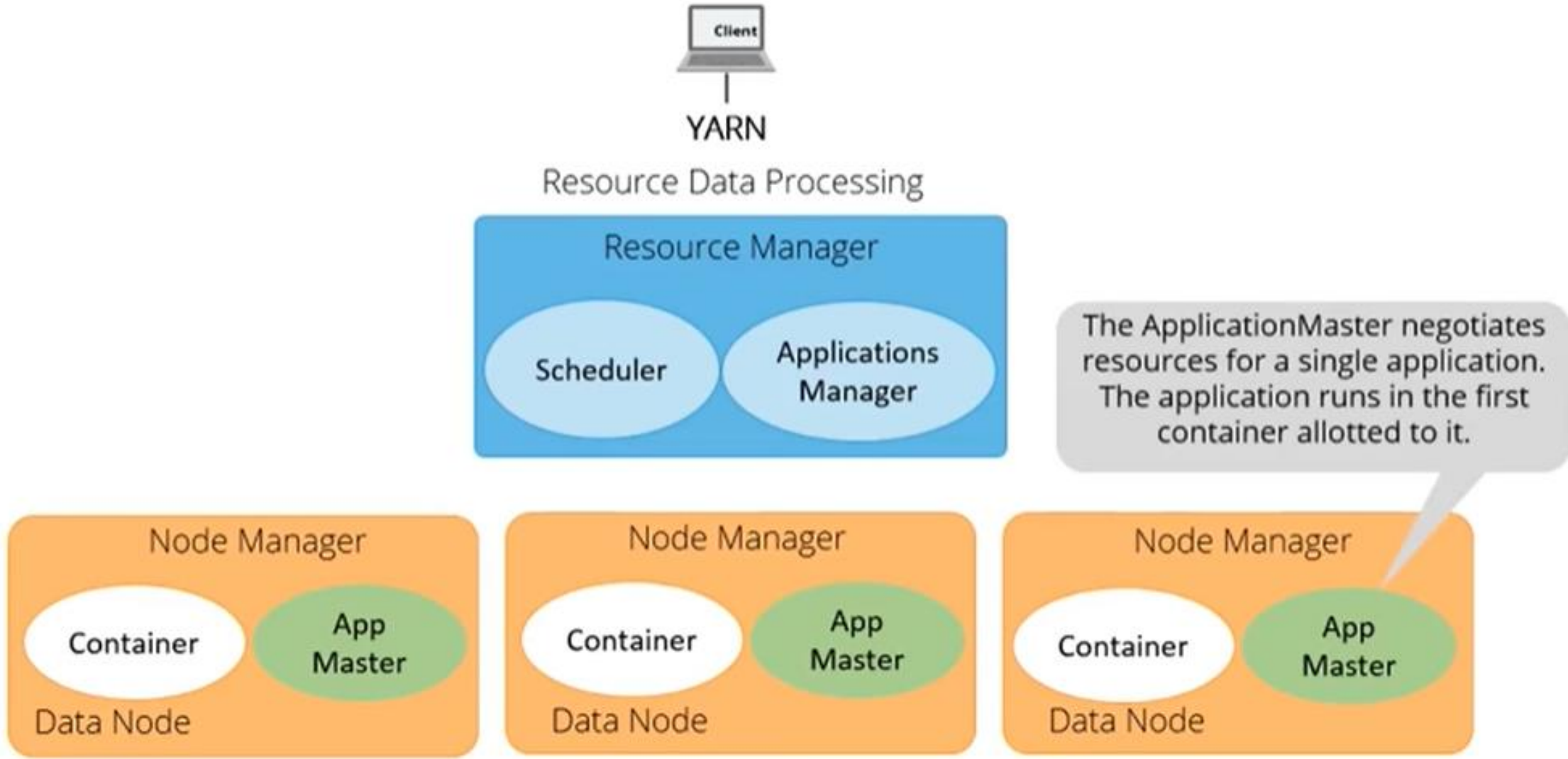
Three Elements of YARN Architecture

The three important elements of the YARN architecture are the ResourceManager, ApplicationMaster, and NodeManager.



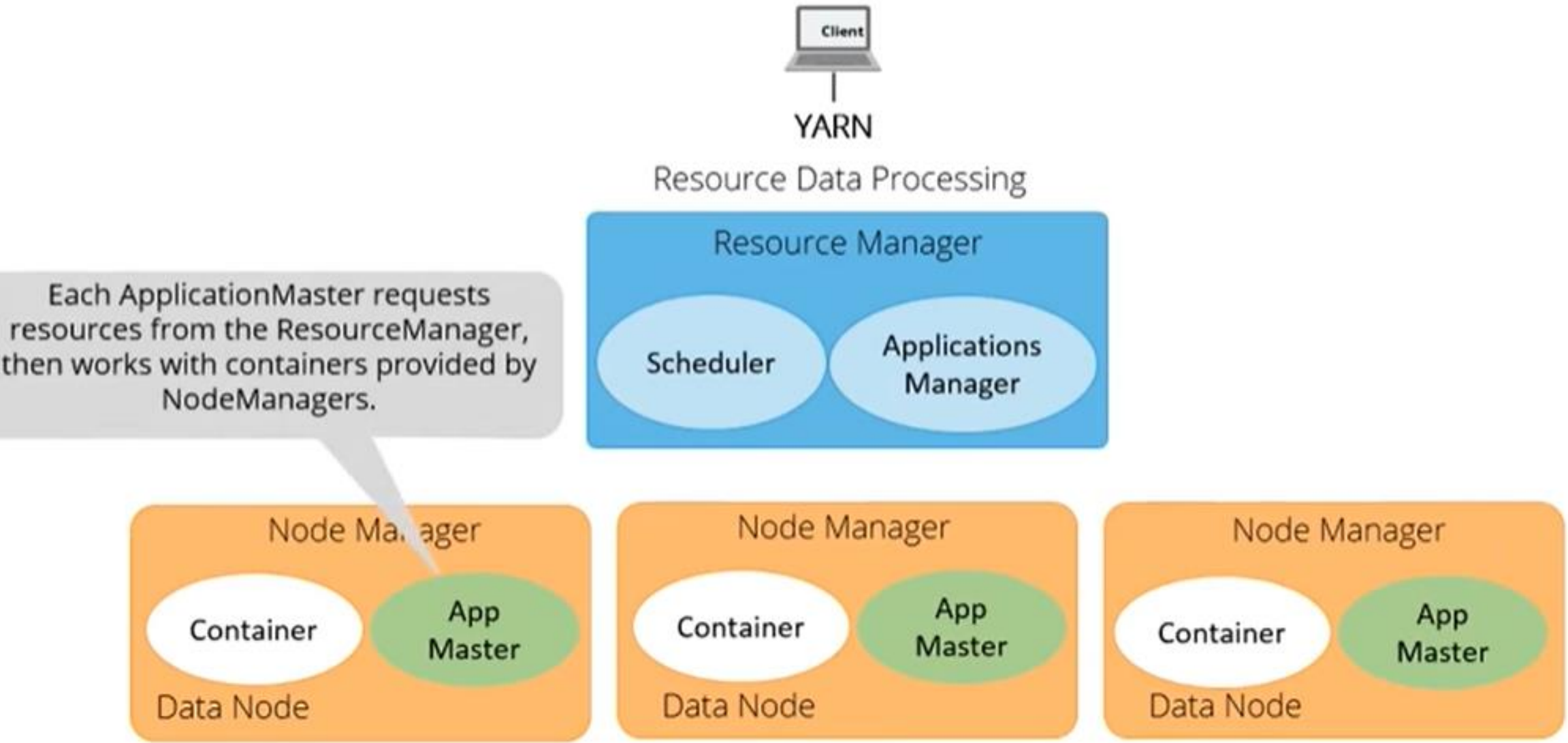
Three Elements of YARN Architecture

The three important elements of the YARN architecture are the ResourceManager, ApplicationMaster, and NodeManager.



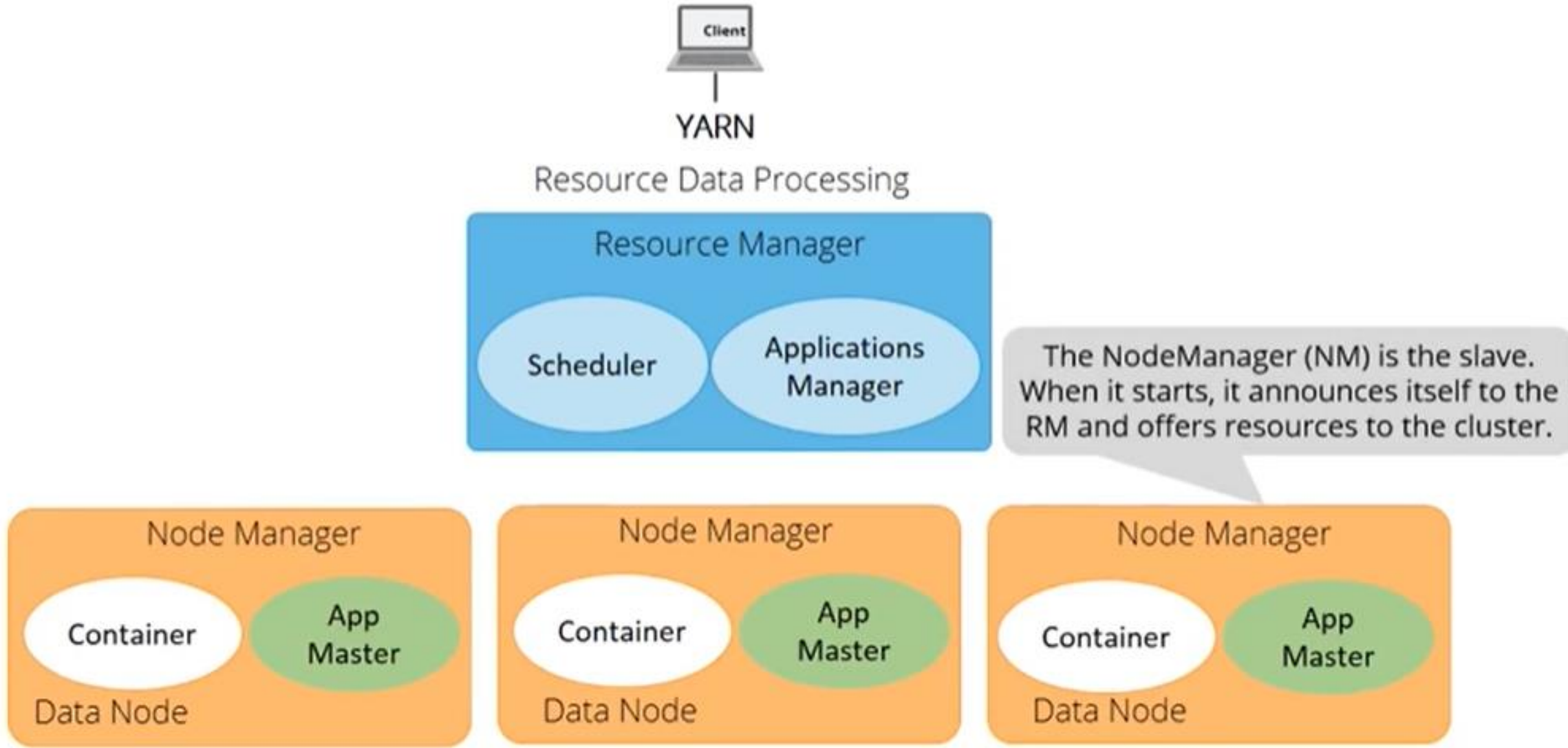
Three Elements of YARN Architecture

The three important elements of the YARN architecture are the ResourceManager, ApplicationMaster, and NodeManager.



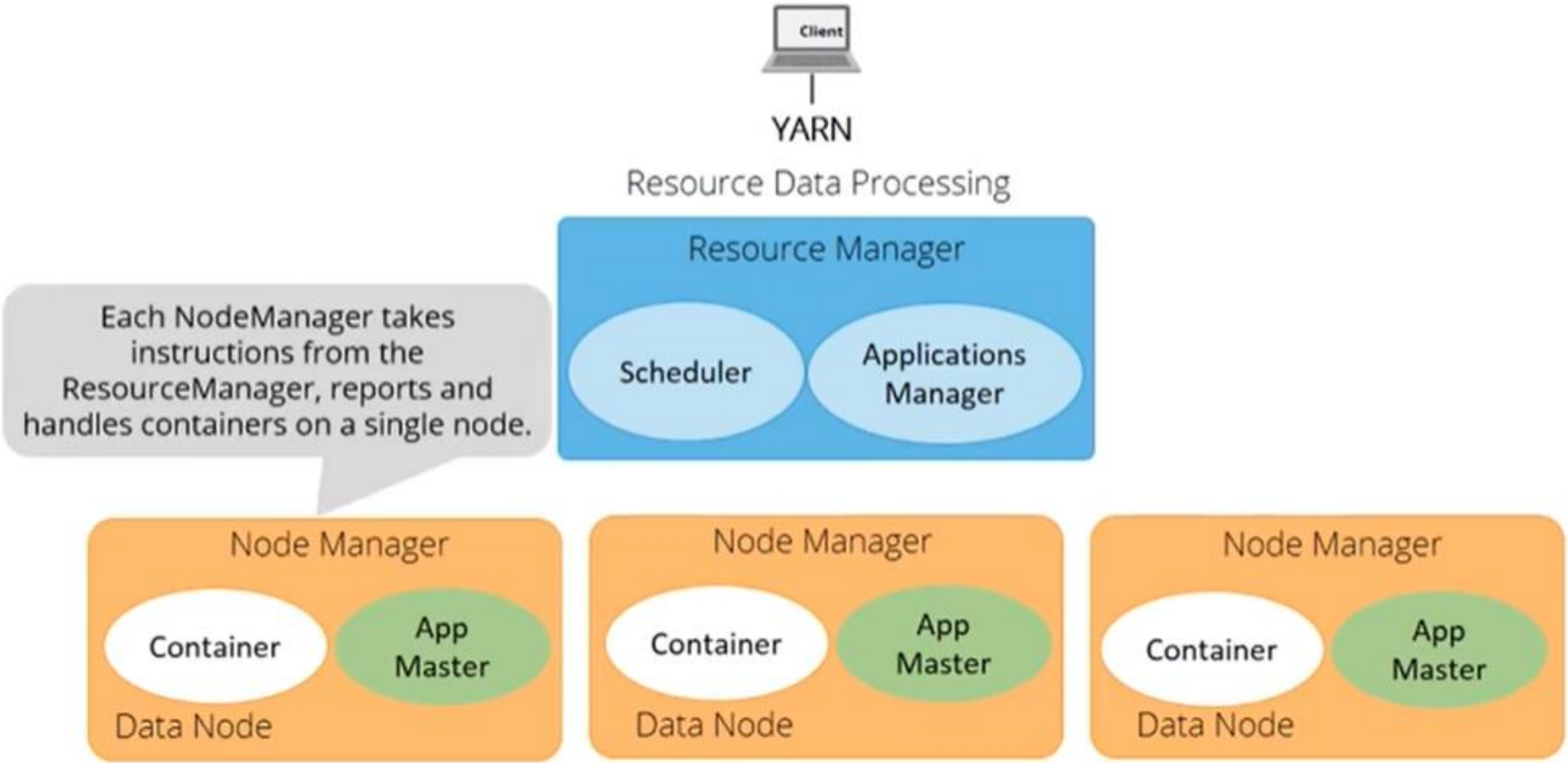
Three Elements of YARN Architecture

The three important elements of the YARN architecture are the ResourceManager, ApplicationMaster, and NodeManager.



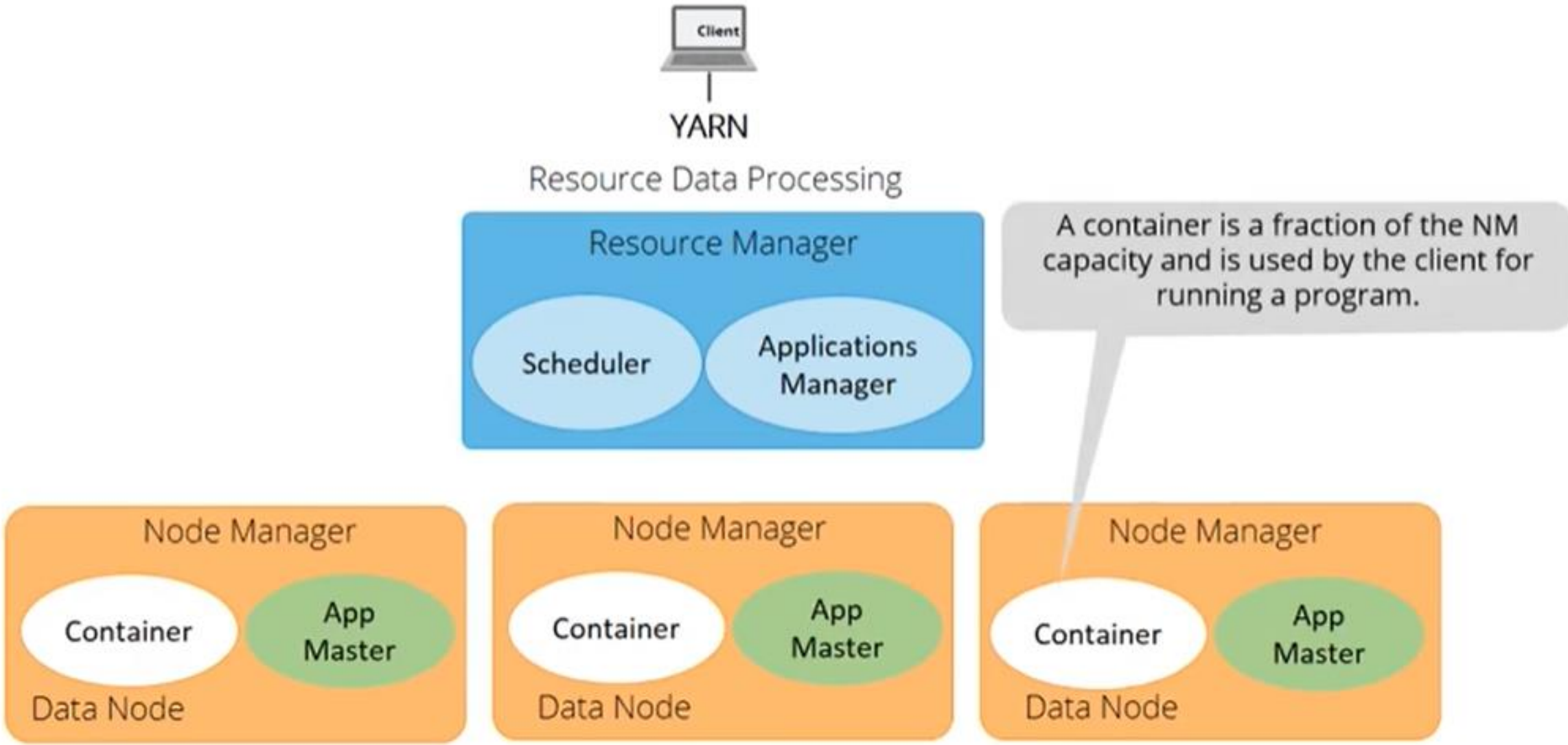
Three Elements of YARN Architecture

The three important elements of the YARN architecture are the ResourceManager, ApplicationMaster, and NodeManager.



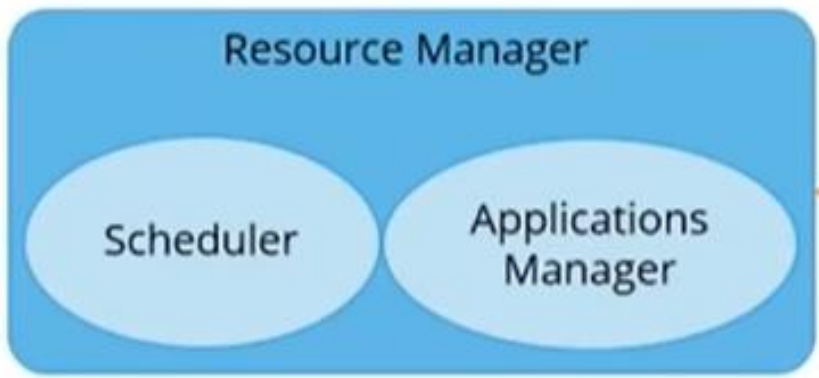
Three Elements of YARN Architecture

The three important elements of the YARN architecture are the ResourceManager, ApplicationMaster, and NodeManager.



YARN Architecture Element—ResourceManager

The RM mediates the available resources in the cluster among competing applications—to maximum cluster utilization.



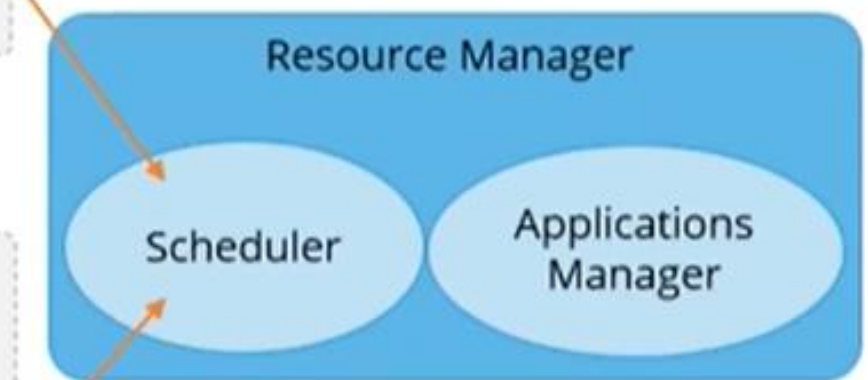
It includes a pluggable scheduler called the YarnScheduler, which allows different policies for managing constraints such as capacity, fairness, and Service Level Agreements.

ResourceManager Component—Scheduler

The Scheduler is responsible for allocating resources to various running applications.

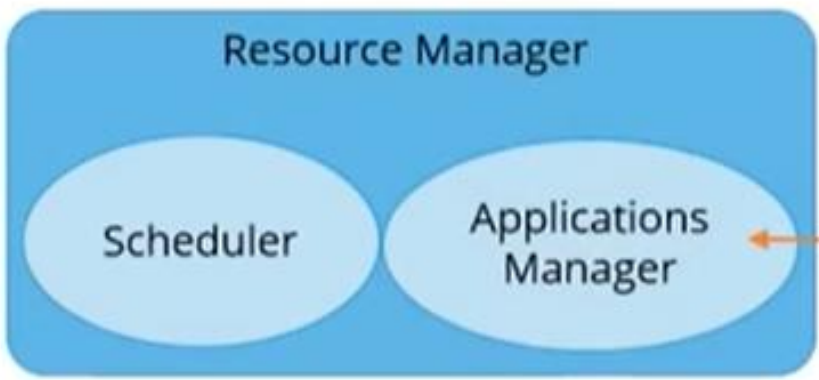
The Scheduler does not monitor or track the status of the application; nor does it restart failed tasks.

The Scheduler has a policy plug-in to partition cluster resources among various applications. Examples: CapacityScheduler, FairScheduler.



ResourceManager Component—ApplicationManager

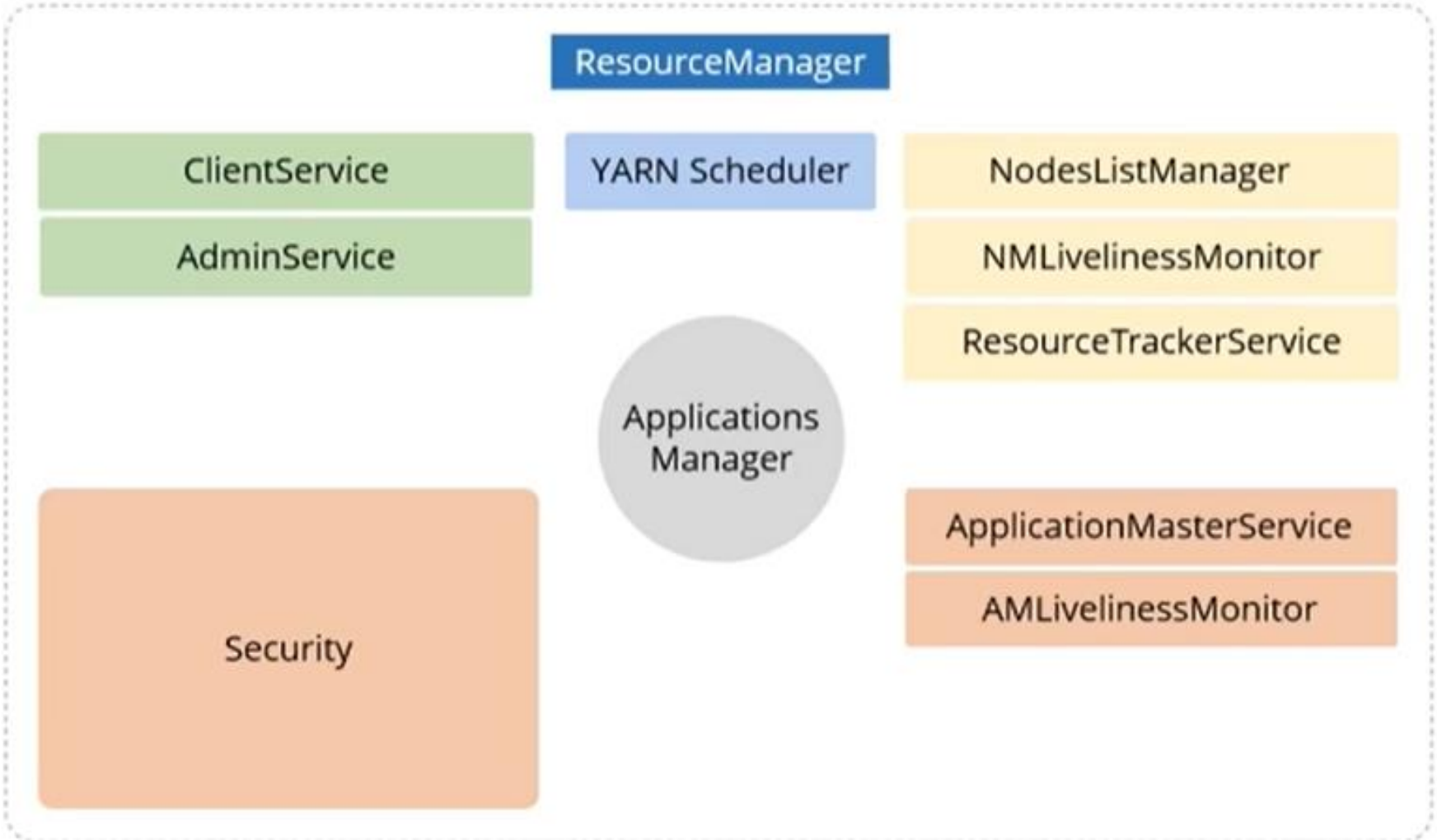
The ApplicationsManager is an interface which maintains a list of applications that have been submitted, currently running, or completed.



The ApplicationsManager accepts job submissions, negotiates the first container for executing the application, and restarts the ApplicationMaster container on failure.

How a ResourceManager operates

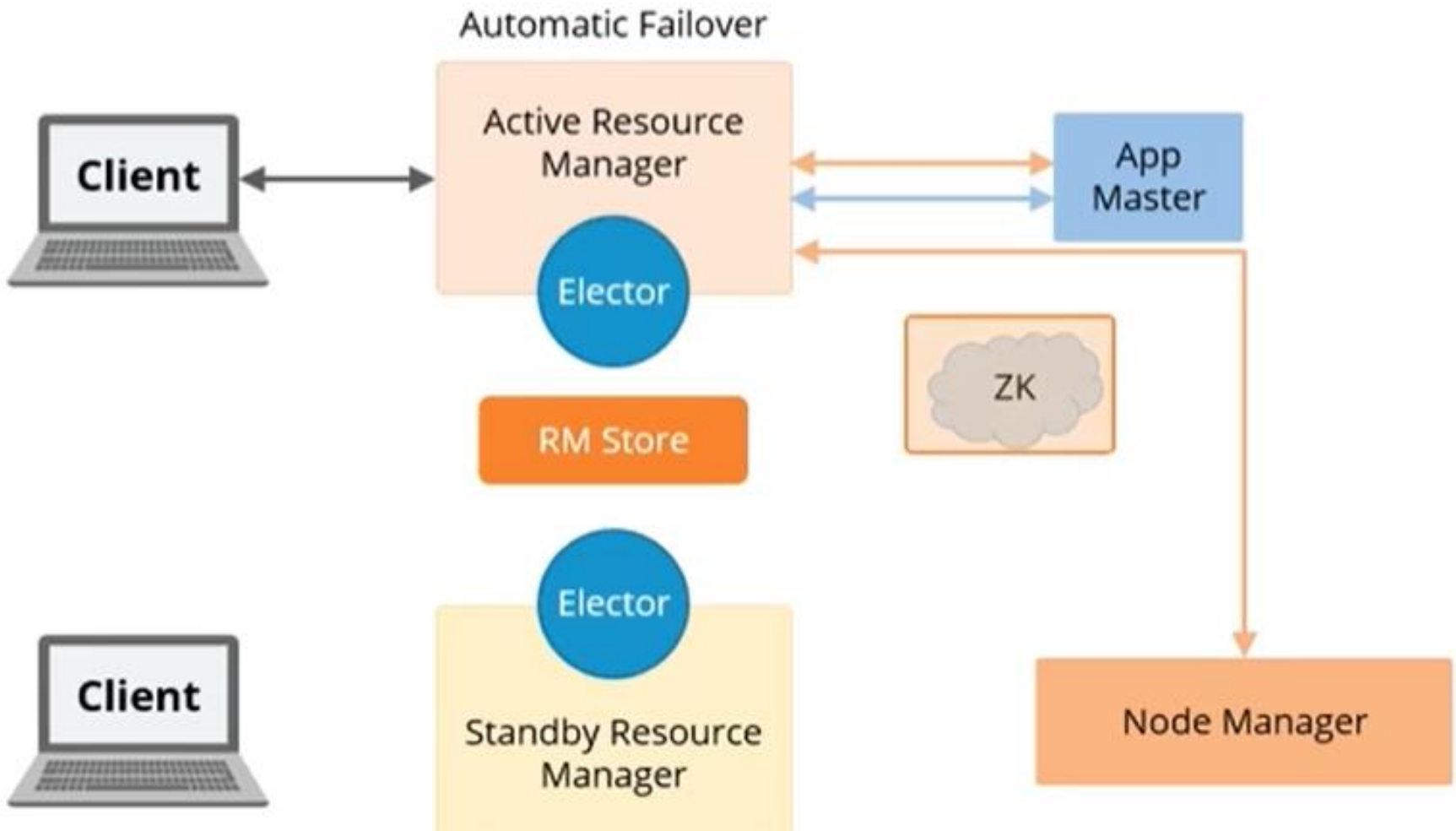
The figure shown here displays all the internal components of the ResourceManager.



ResourceManager in High Availability Mode

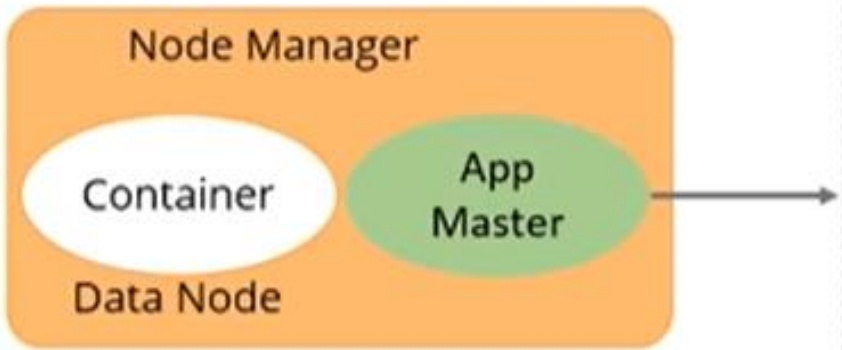
Before Hadoop 2.4, the ResourceManager was the single point of failure in a YARN cluster.

The High Availability, or HA, feature an Active/Standby ResourceManager pair to remove this single point of failure.



YARN Architecture Element—ApplicationMaster

The ApplicationMaster in YARN is a framework-specific library, which negotiates resources from the RM and works with the NodeManager or Managers to execute and monitor containers and their resource consumption.



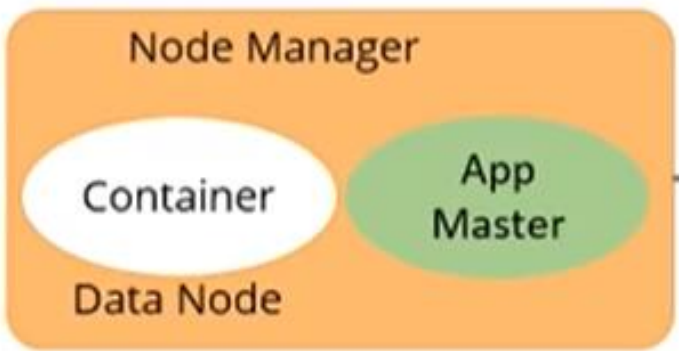
The ApplicationMaster:

- manages the application lifecycle
- makes dynamic adjustments to resource consumption
- manages execution flow
- manages faults
- provides status and metrics to the RM
- interacts with NodeManager and RM using extensible communication protocols
- Is not run as a trusted service

While every application has its own instance of an AppMaster, it is possible to implement an AppMaster for a set of applications as well.

YARN Architecture Element—NodeManager

When a container is leased to an application, the NodeManager sets up the container's environment, including the resource constraints specified in the lease and any dependencies.

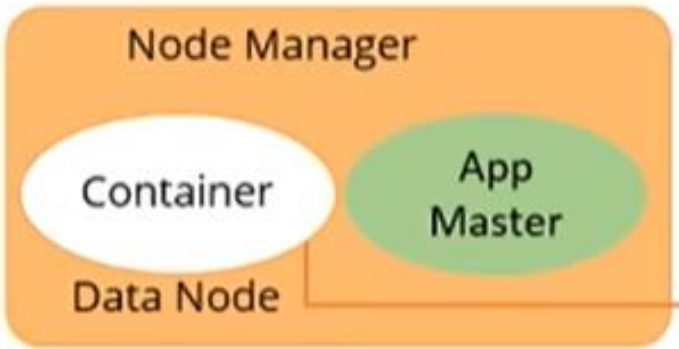


The NodeManager runs on each node and manages the following:

- Container lifecycle management
- Container dependencies
- Container leases
- Node and container resource usage
- Node health
- Log management
- Reporting node and container status to the RM

YARN Container

A YARN container is a result of a successful resource allocation, that is, the RM has granted an application a lease to use specified resources on a specific node.

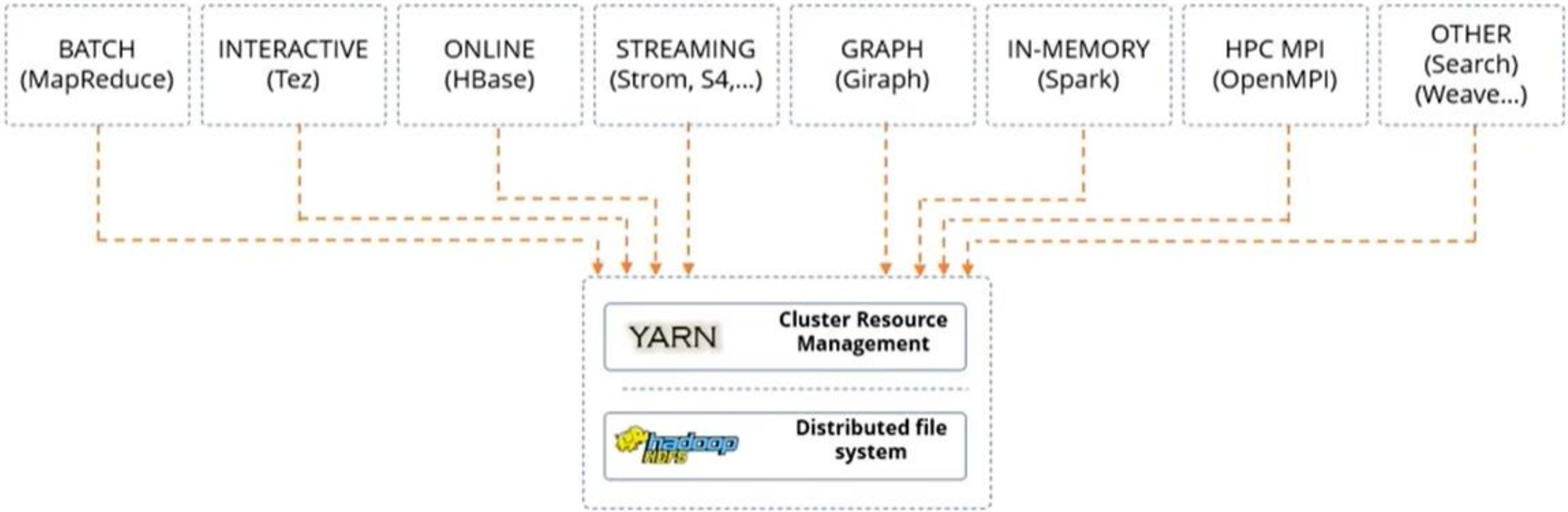


To launch the container, the ApplicationMaster must provide a container launch context (CLC) that includes the following information:

- Environment variables
- Dependencies, that is, local resources such as data files or shared objects needed prior to launch
- Security tokens
- The command necessary to create the process the application wants to launch

Applications on YARN

There can be many different workloads running on a Hadoop YARN cluster.



How YARN Runs an Application

There are five steps involved in YARN to run an application:

The client submits an application to the ResourceManager

The ResourceManager allocates a container

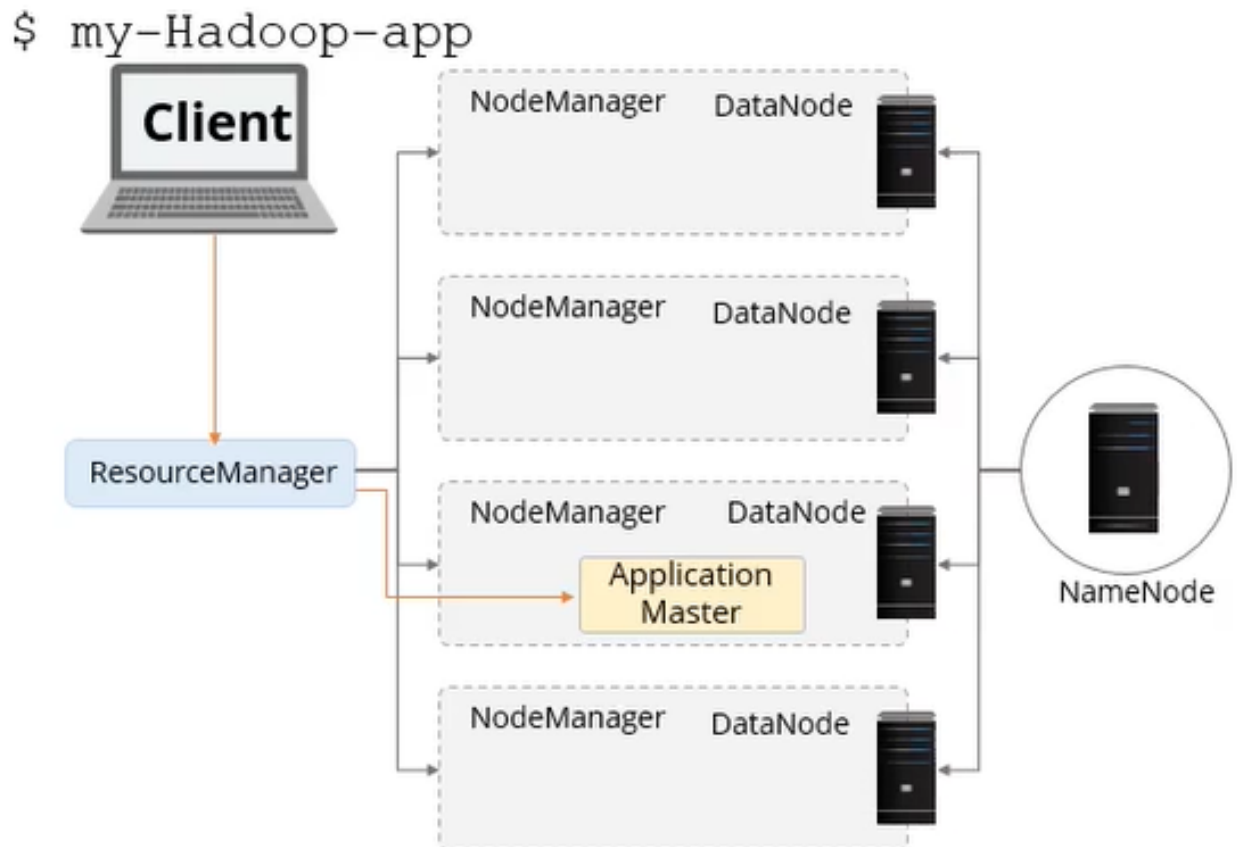
The ApplicationMaster contacts the related NodeManager

The NodeManager launches the container

The container executes the ApplicationMaster

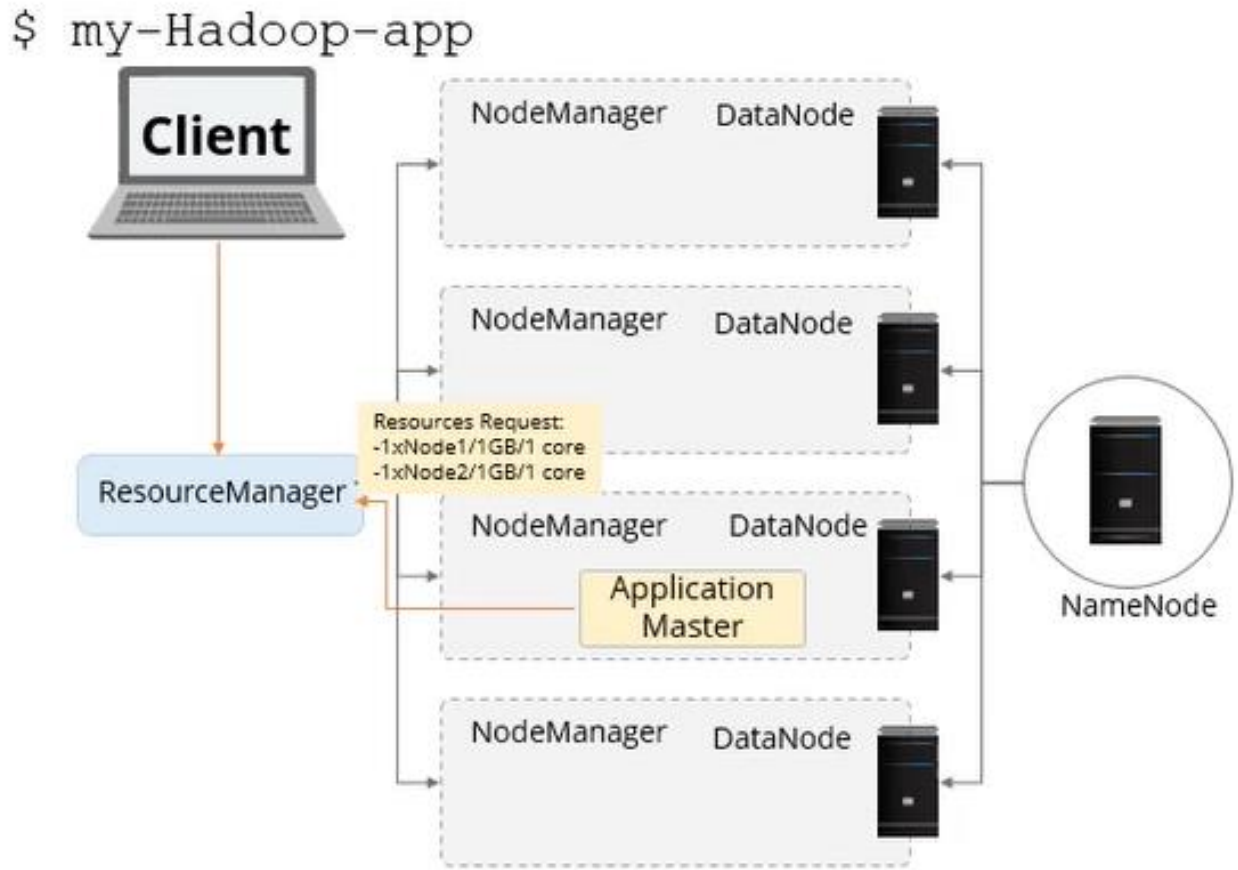
Step1—Application Submitted to ResourceManager

Users submit applications to the ResourceManager by typing the hadoop jar command.



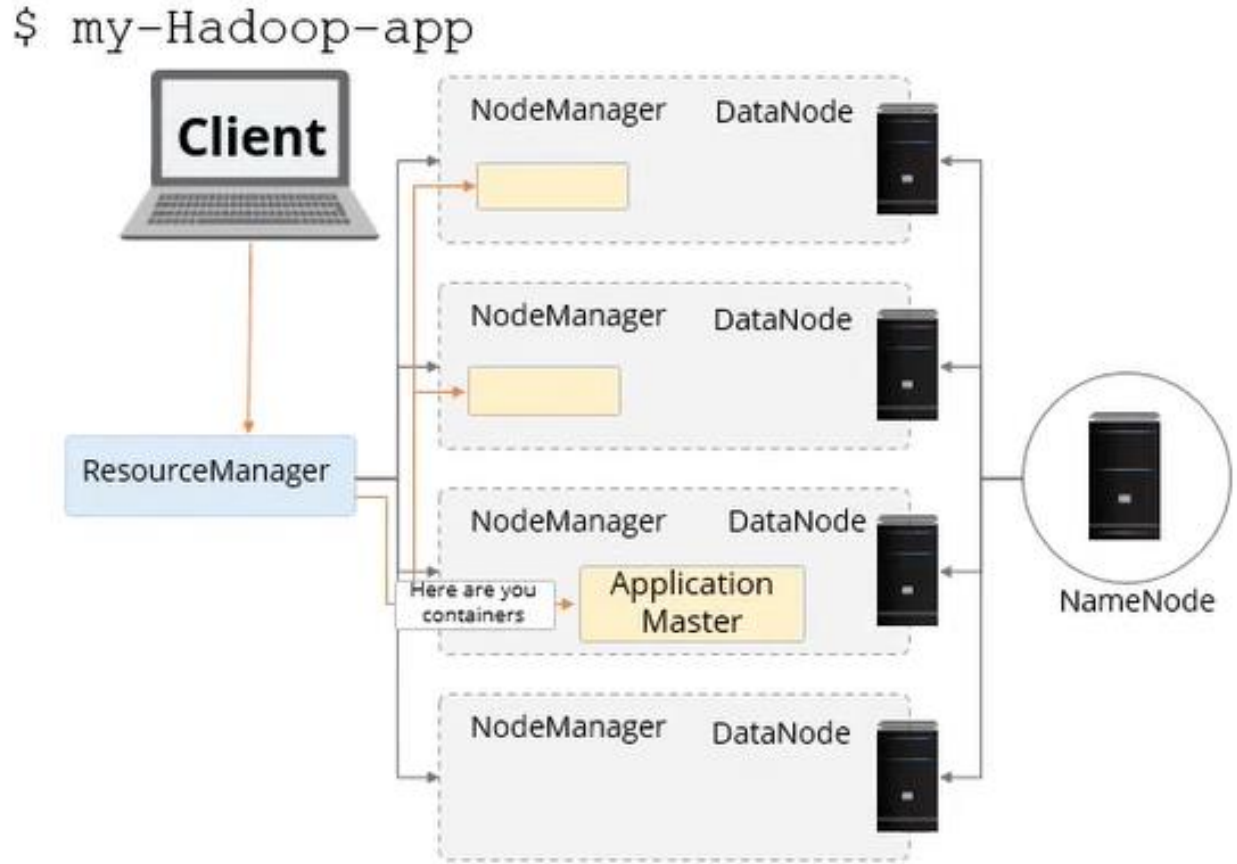
Step2—ResourceManager Allocates a Container

When the ResourceManager accepts a new application submission, one of the first decisions the Scheduler makes is selecting a container.



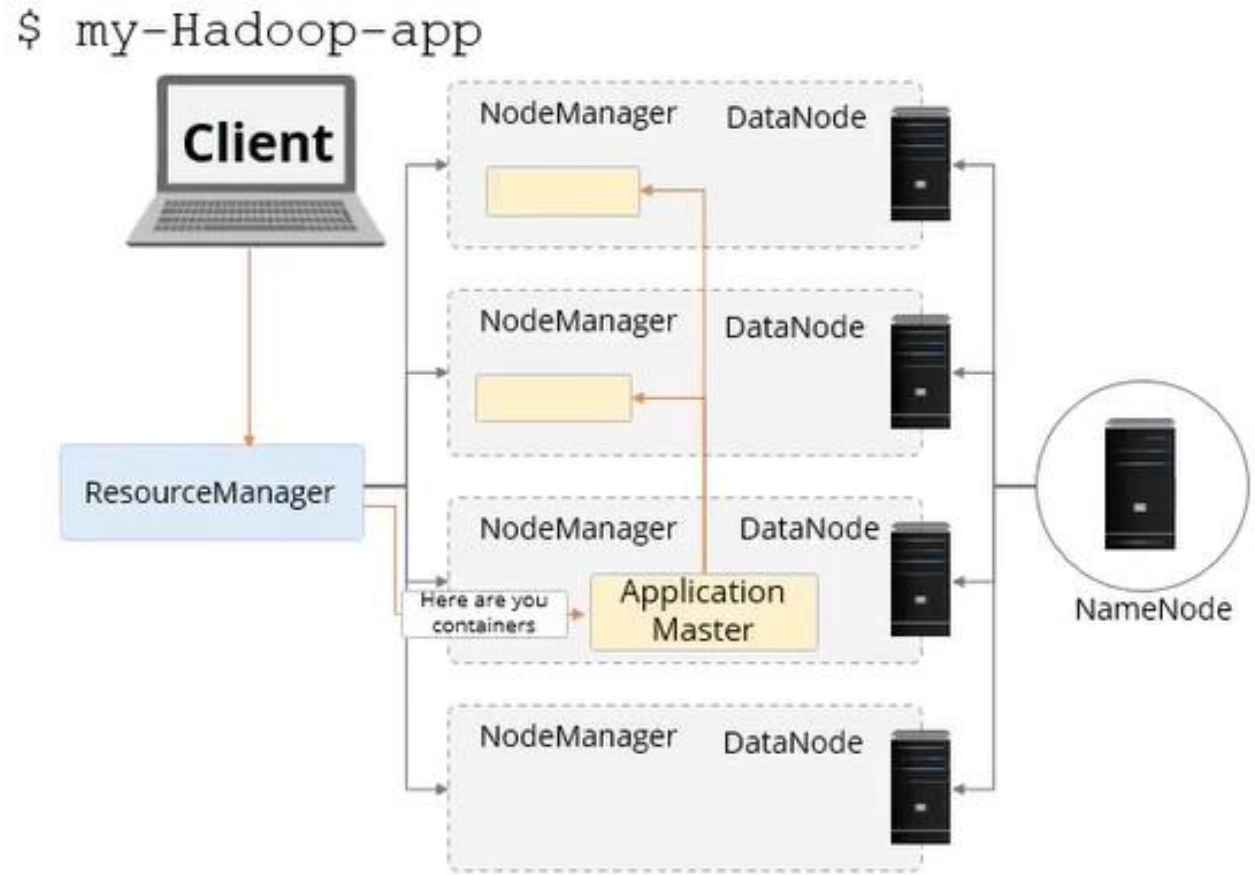
Step3 —ApplicationMaster Contacts NodeManager

After a container is allocated, the ApplicationMaster asks the NodeManager managing the host on which the container was allocated to use these resources to launch an application-specific task.



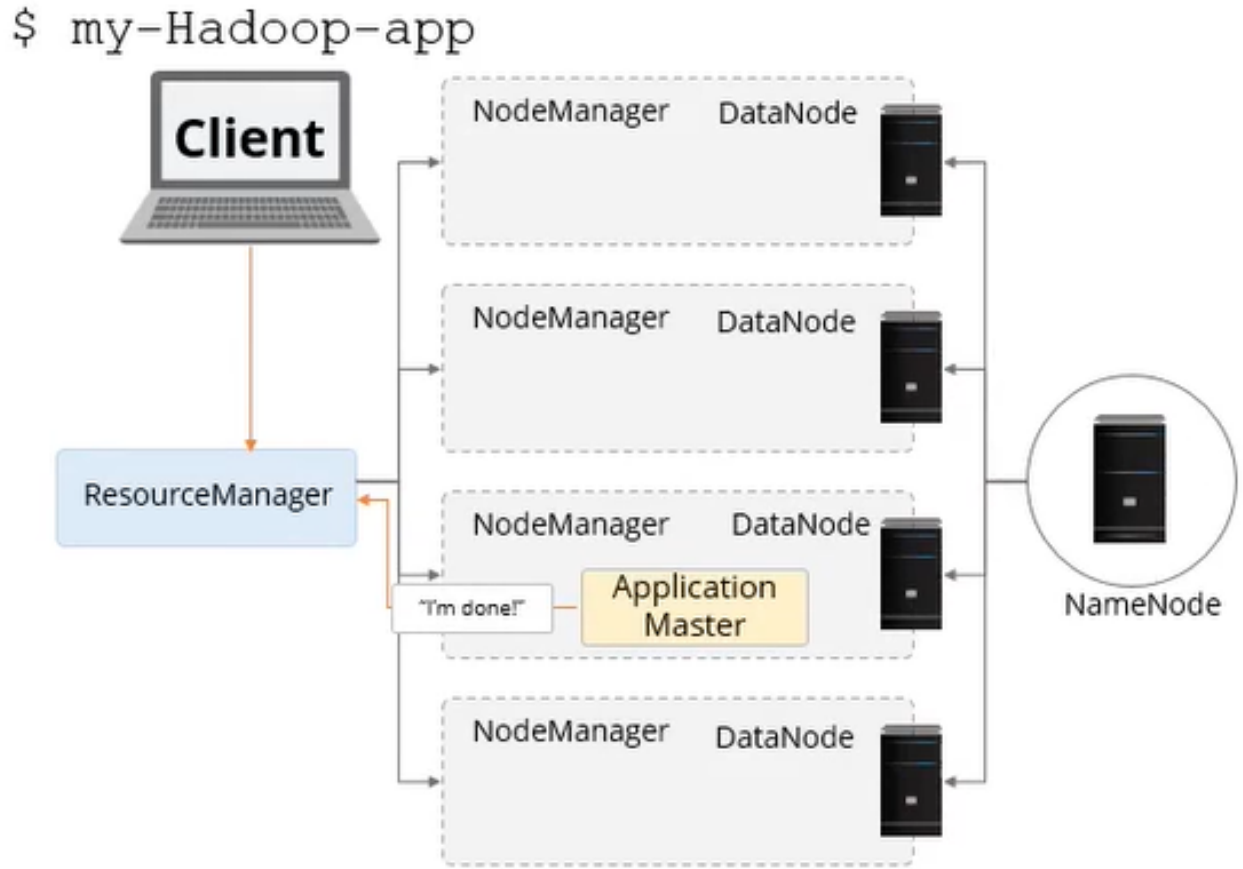
Step4—ResourceManager Launches a Container

The NodeManager does not monitor tasks; it only monitors the resource usage in the containers.



Step5—Container Executes the ApplicationMaster

After the application is complete, the ApplicationMaster shuts itself and releases its own container.



Three Tools for YARN Developers

Hadoop includes three tools for YARN developers:

YARN Web UI

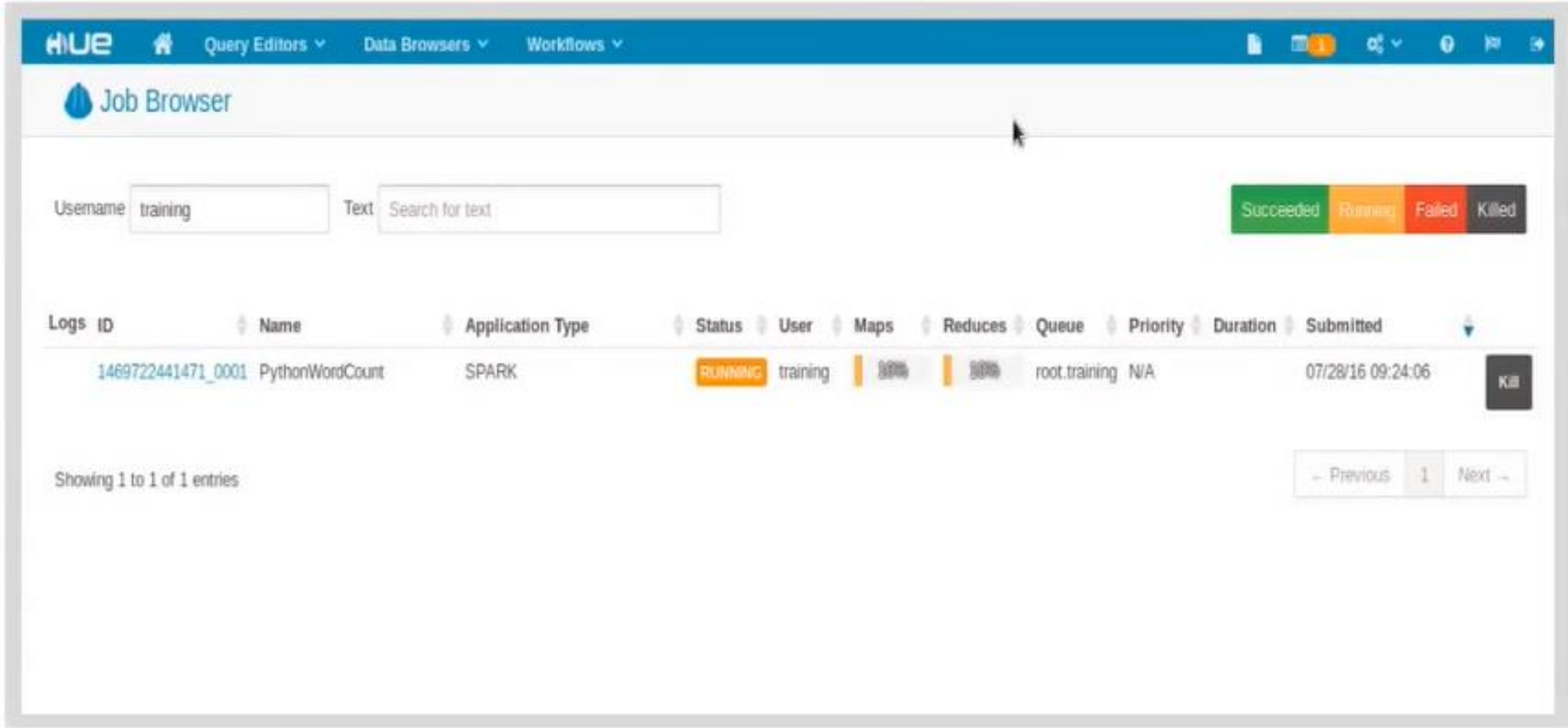
YARN web UI runs on 8088 port, by default.

It also provides a better view than Hue; however you can't control or configure from YARN web UI.

Hue Job Browser

Hue Job browser

The Hue Job Browser allows you to monitor status of job, kill a running job, and view logs.



YARN Command Line

Most of the YARN commands are for administrator rather than developer

Few useful commands for developer:

- yarn –help
list all command of yarn

- yarn –version
print the version

- yarn logs -applicationId <app-id>
views logs of specified application ID

Running MapReduce Examples on Hadoop YARN

Running MapReduce Examples on Hadoop YARN

Run Cmd as administrator

```
hdfs namenode -format
```

Run Cmd as administrator

```
cd\
cd hadoop\sbin
start-dfs
start-yarn
```

Run Cmd as administrator

```
cd\
cd hadoop\share\hadoop\mapreduce
yarn jar hadoop-mapreduce-examples-3.3.1.jar
```

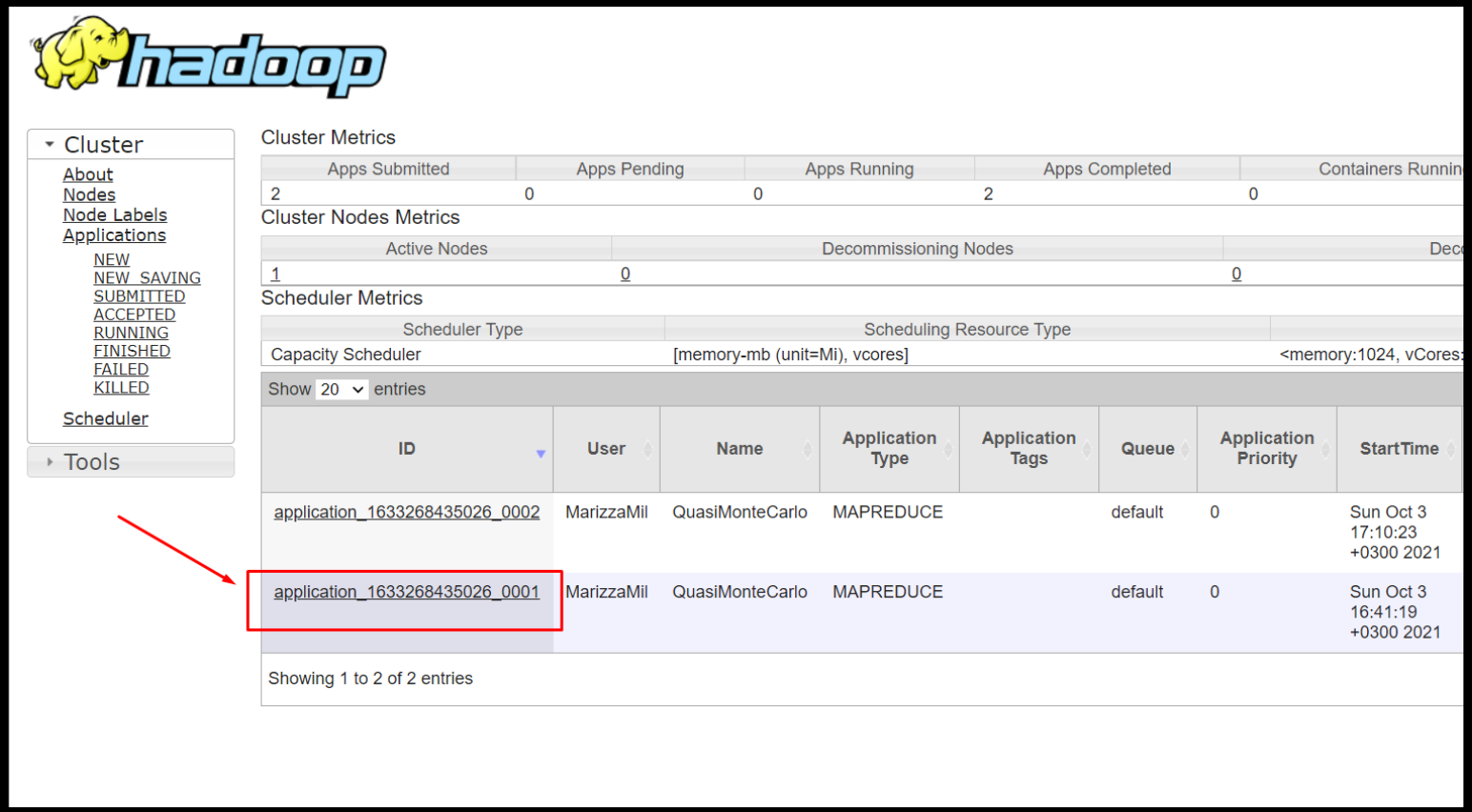
Running the pi Example

To run the pi example with 16 maps and 100000 samples, run the following command:
Run Cmd as administrator

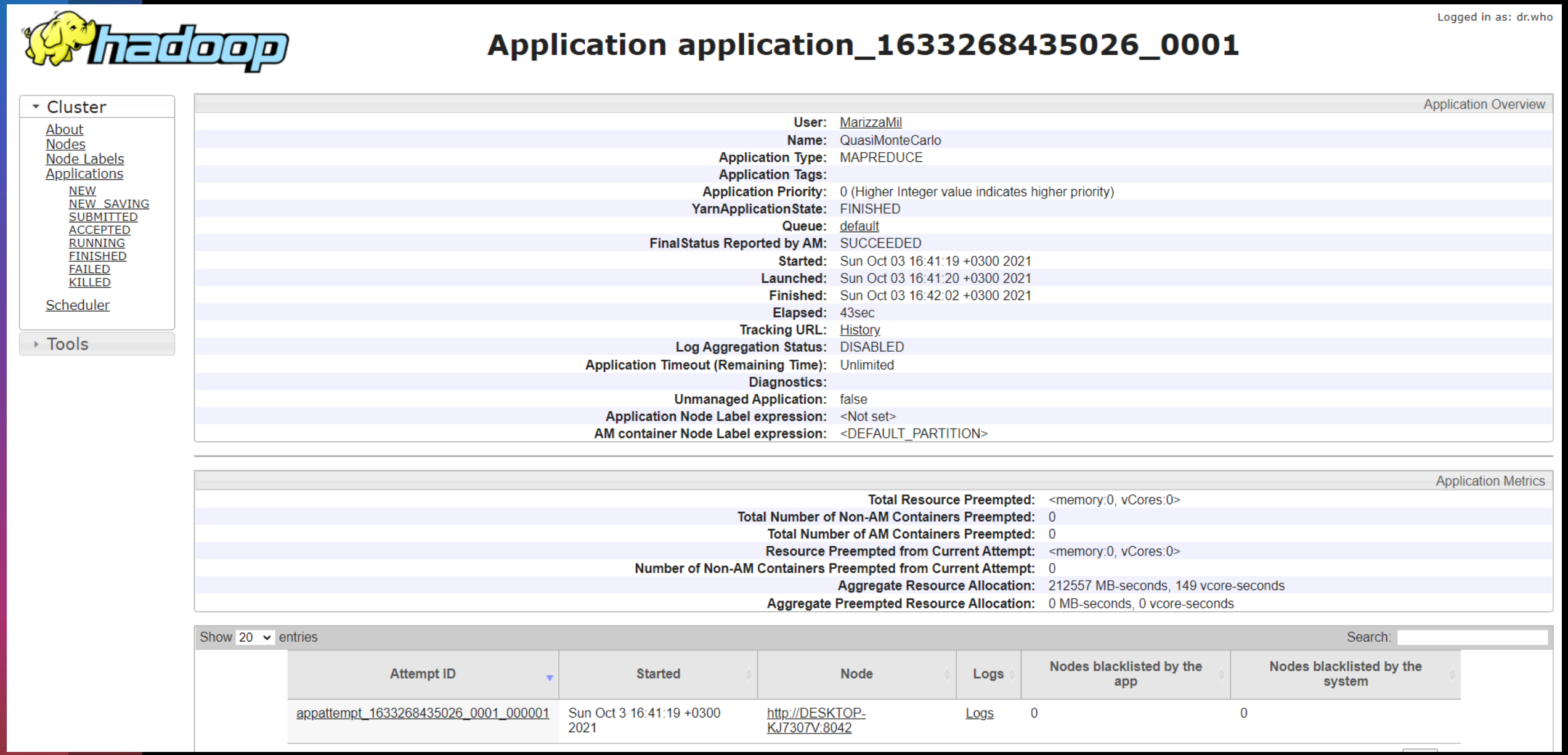
```
cd\
cd hadoop\share\hadoop\mapreduce
yarn jar hadoop-mapreduce-examples-3.3.1.jar  pi 16 100000
```

Using the Web GUI to Monitor Examples

The following figure shows the main YARN web interface (http://hostname:8088)



This page provides information similar to that on the Running Applications page, but only for the selected job.



Run wordcount MapReduce job

Run Cmd as administrator

```
hdfs namenode -format
```

Run Cmd as administrator

```
cd\
cd hadoop\sbin
start-dfs
start-yarn
// 1.  Create a text file with some content. We'll pass this file as input to the wordcount MapReduce job for counting words.
C:\file1.txt
// 2. Create a directory ('new_demo') in HDFS to keep all the text files ('file1.txt') to be used for counting words.
hadoop fs -mkdir /new_demo
// 3. Copy the text file(say 'file1.txt') from local disk to the newly created 'new_demo' directory in HDFS.
hdfs dfs -copyFromLocal C:\temp\file1.txt /new_demo
// 4. Check content of the copied file
hdfs dfs -ls /new_demo
```

Run Cmd as administrator

```
cd\
cd hadoop\share\hadoop\mapreduce
// 5. Run the wordcount MapReduce job provided
yarn jar hadoop-mapreduce-examples-3.3.1.jar wordcount /new_demo output
// 6. Check output.
hdfs dfs -cat output/*
```

http://hostname:8088