



Introduction to **YARN**

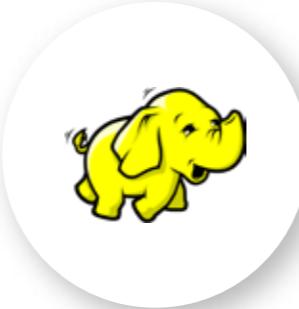
Module 3 : Hadoop, HDFS,
& YARN



WHERE ARE WE in this journey?



INTRODUCTION
TO BIG DATA



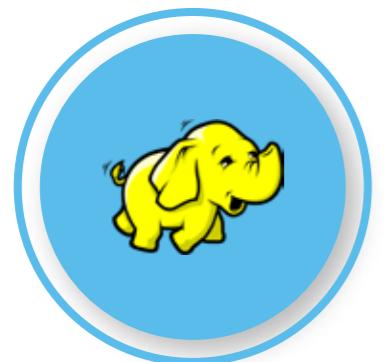
SCALING, STORAGE &
COMPUTATION WITH
APACHE HADOOP



PYTHON



HDFS



INTRODUCTION
TO YARN



MAPREDUCE



PIG



HIVE

LEARNT SO FAR



HOW HADOOP OVERCOMES CHALLENGES FACED IN TRADITIONAL COMPUTING?



GETTING STARTED WITH HADOOP



INTRODUCTION TO HADOOP & HDFS



COMMONLY USED LINUX & HDFS COMMANDS

TAKEAWAYS

from today's session

- **DRAWBACKS OF EARLIER HADOOP VERSIONS**
- **HOW YARN SOLVES DRAWBACKS OF EARLIER HADOOP VERSIONS**
- **WHERE DOES YARN FIT IN HADOOP ECOSYSTEM?**
- **ARCHITECTURE OF YARN**

AGENDA

Introduction to YARN



CHALLENGES OF EARLIER HADOOP VERSIONS



WHAT IS APACHE YARN?



FEATURES OF YARN



ARCHITECTURE OF YARN

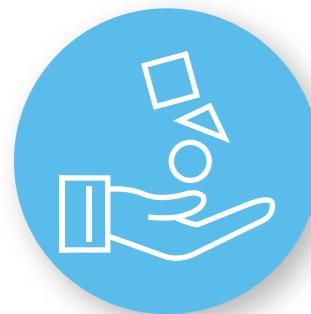


COMPONENTS OF YARN

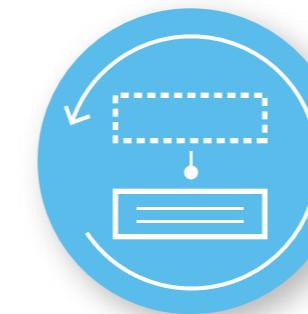
CHALLENGES

of earlier versions of Hadoop

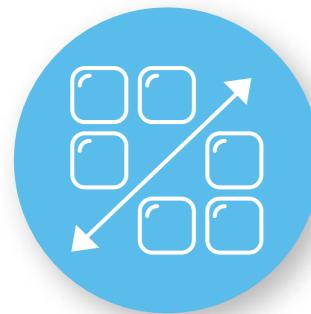
DRAWBACKS IN EARLIER HADOOP VERSIONS



Underutilization of resources



Lack of recovery mechanism



Limited scalability



Processing using only
MapReduce

Let us expand on these topics..

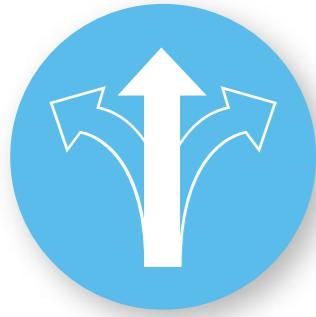
SOME DEFINITIONS

that are useful

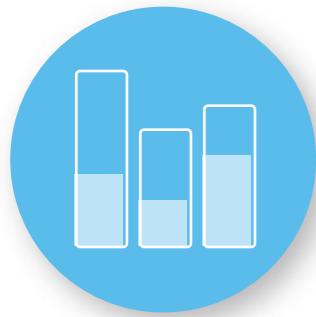
- ▶ A process needs an **area** of computation where actual processing happens
- ▶ In Hadoop, this **computation area** is called **SLOT**
- ▶ **Types of slots in Hadoop** framework:
 - ▶  MAP SLOT
 - ▶  REDUCE SLOT

CHALLENGE 1

Under utilization of resources in earlier Hadoop Versions



Earlier Hadoop had ***predefined number of map & reduce slots*** which ***lead to inflexible slots*** configured on DataNodes.



Under utilization occurs because ***map slots*** might be '***full***' while ***reduce slots*** are ***empty*** (and vice-versa)

CHALLENGE 2:

Scalability => Limitations of adding resources in earlier Hadoop versions

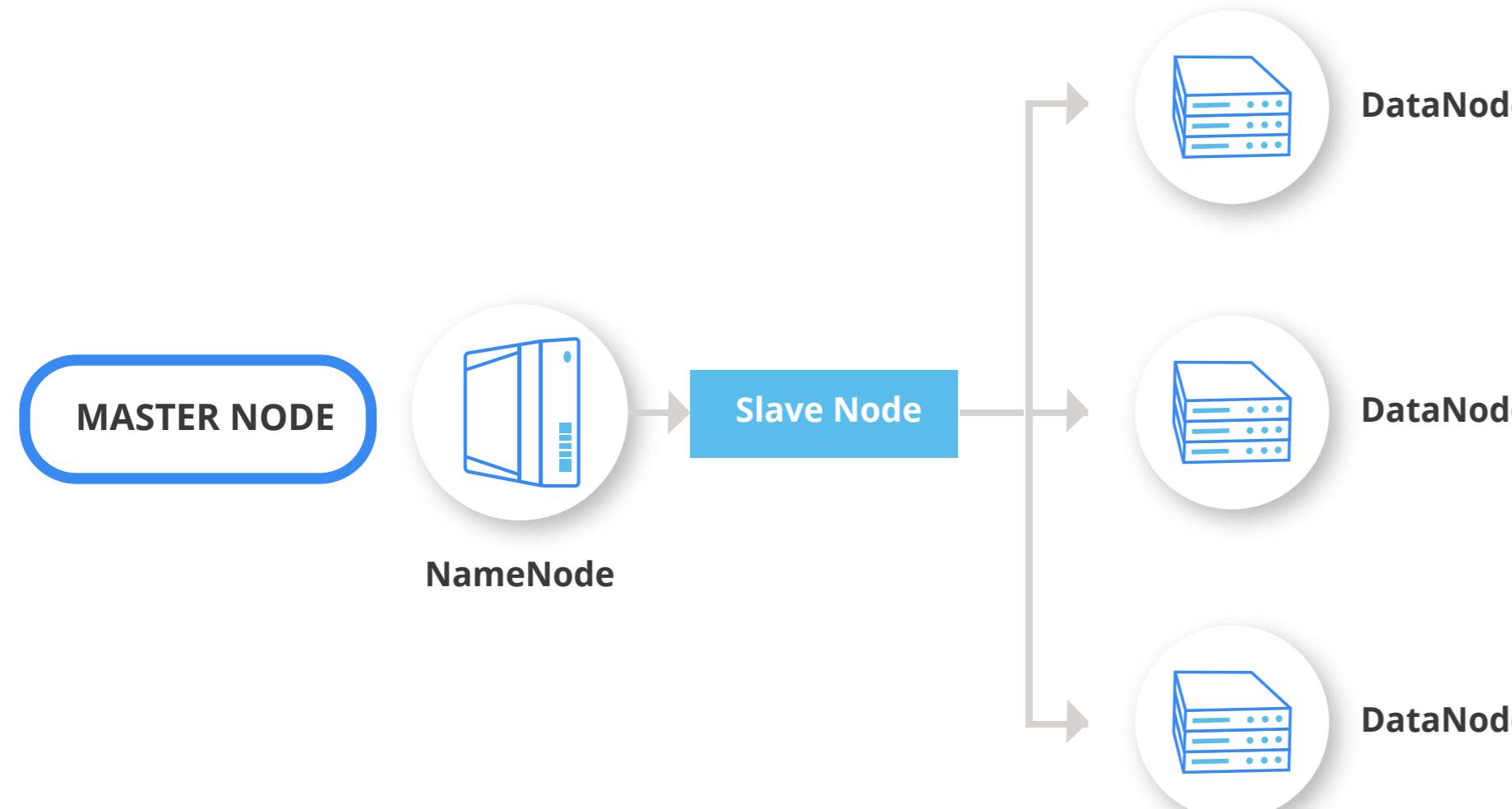
In earlier Hadoop versions, we had

- ▶ Maximum cluster size of **4,000** nodes
- ▶ At most **40,000 concurrent tasks** could be executed

CHALLENGE 3:

Lack of recovery mechanism: Single point of failure

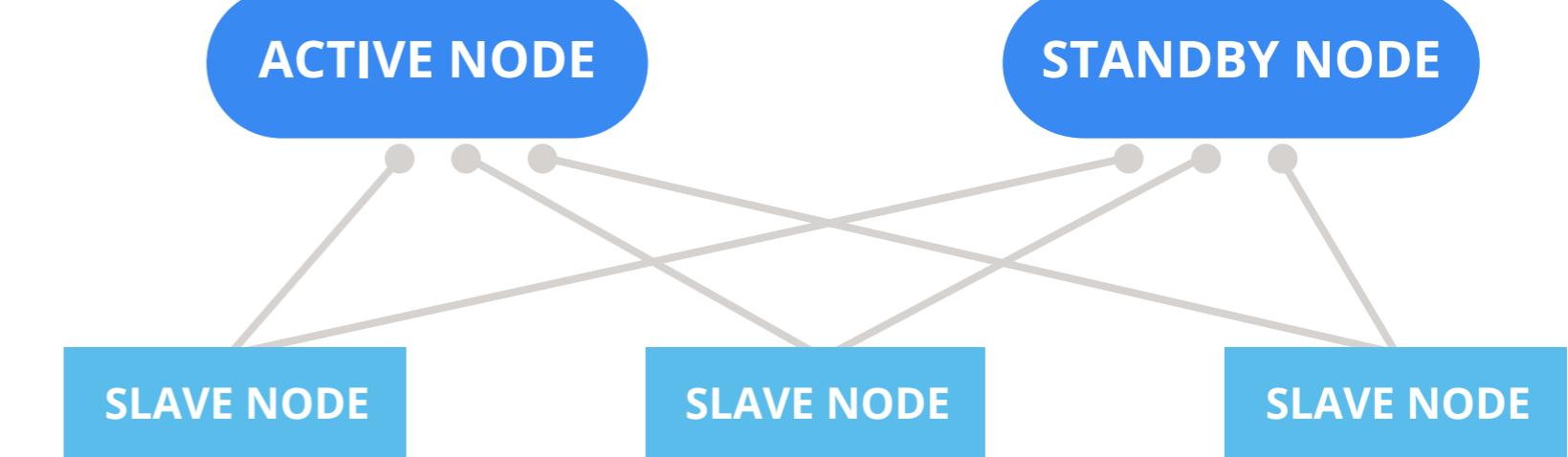
EARLIER HADOOP VERSIONS



- ▶ One NameNode
- ▶ In case of failure, entire data is lost

This is called **SINGLE POINT OF FAILURE**

CURRENT HADOOP VERSIONS



- ▶ More than one NameNode
- ▶ No loss of data

CHALLENGE 4:

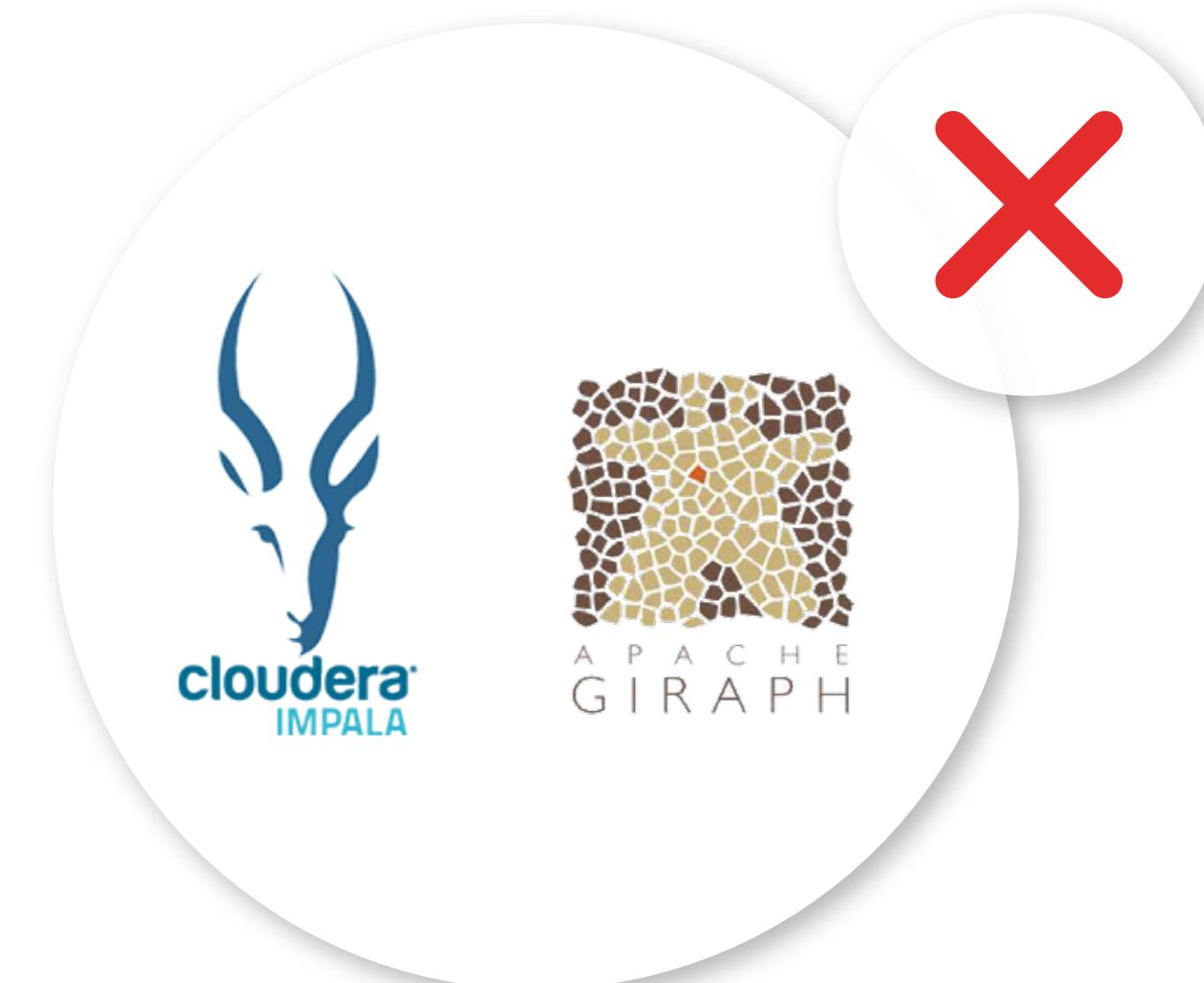
Processing using only MapReduce

EARLIER HADOOP VERSIONS

Supported (*MR-based*)



Not supported (*Non-MR based*)



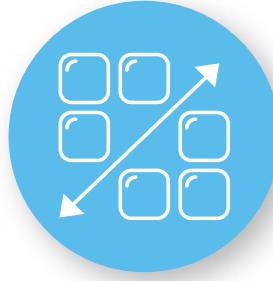
YARN: SOLUTION TO PROBLEMS

in earlier Hadoop versions

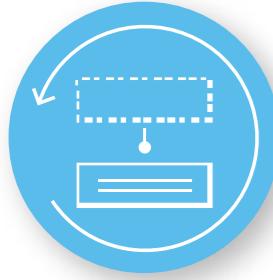
PROBLEMS



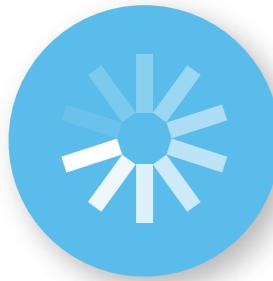
Underutilization of resources



Limited scalability



Lack of recovery mechanism



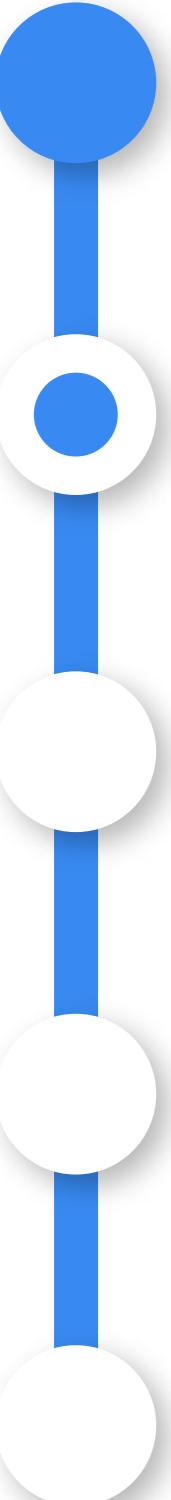
Processing using only MapReduce

SOLUTION



AGENDA

Introduction to YARN



CHALLENGES OF EARLIER HADOOP VERSIONS

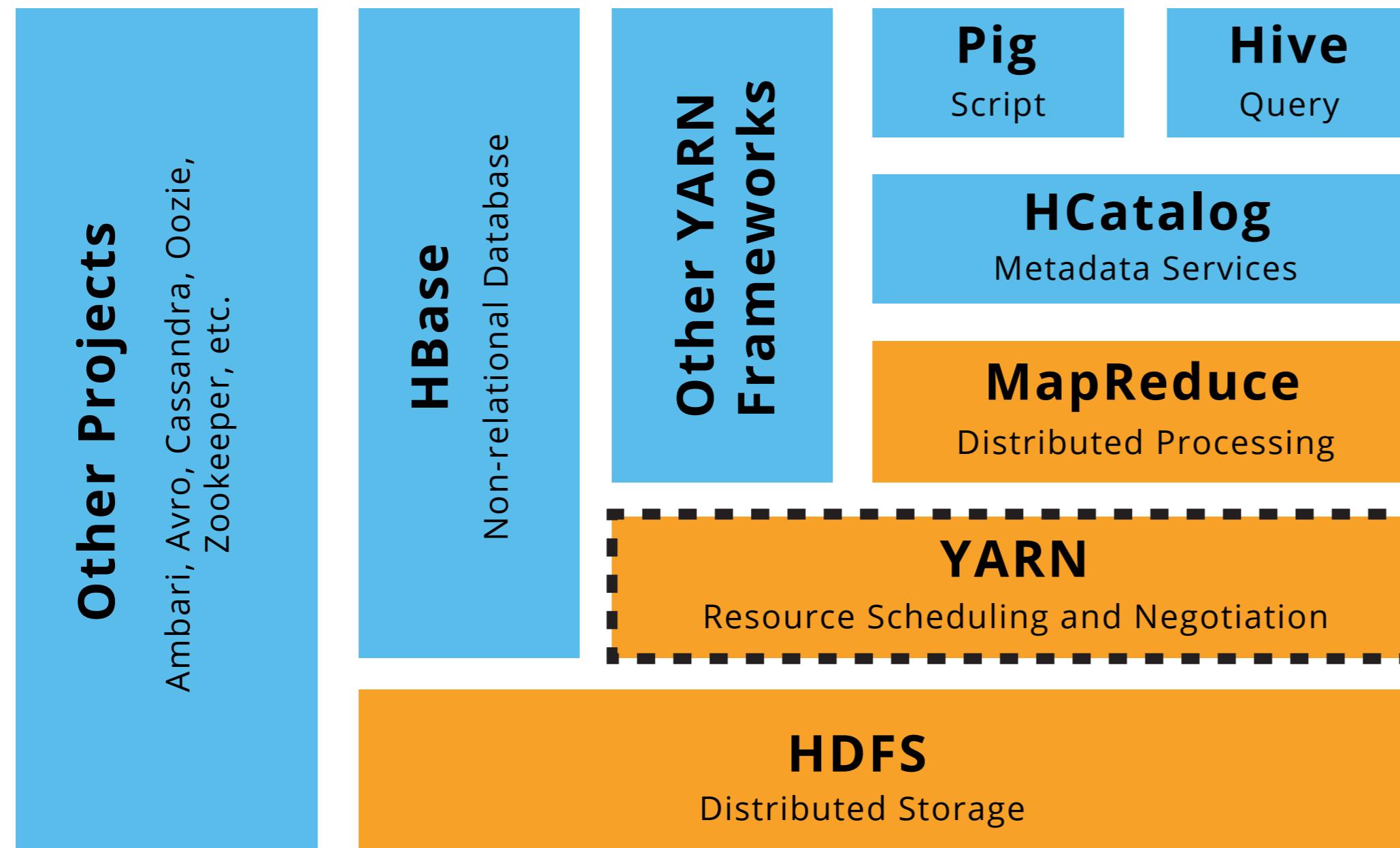
WHAT IS APACHE YARN?

FEATURES OF YARN

ARCHITECTURE OF YARN

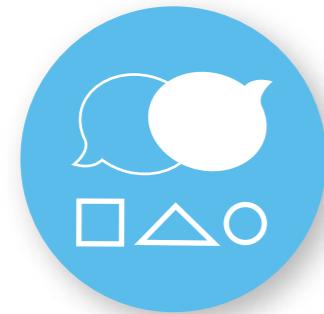
COMPONENTS OF YARN

WHAT IS YARN?



Apache YARN is a ***Resource Management Layer*** of Hadoop

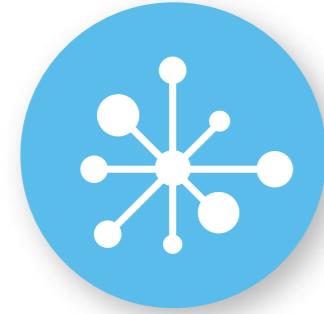
WHAT IS YARN?



Apache **YARN** stands for **Y**et **A**nother **R**esource **N**egotiator



Separates ***resource management & processing components***

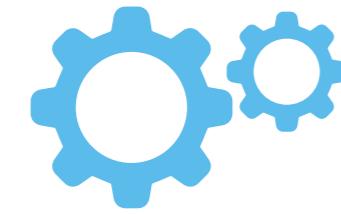


Is considered as a ***data operating system*** of Hadoop

WHAT IS YARN?

Continued

It is a *central platform for*



CONSISTENT OPERATIONS

Multiple look ups onto the same non actively manipulated file should return the same content irrespective of its location.



DATA GOVERNANCE

YARN manages the lifecycle of data and its accessibility to users.



SECURITY

Hadoop is a distributed system and requires authentication between the different components while they communicate with each other.

AGENDA

Introduction to YARN



CHALLENGES OF EARLIER HADOOP VERSIONS

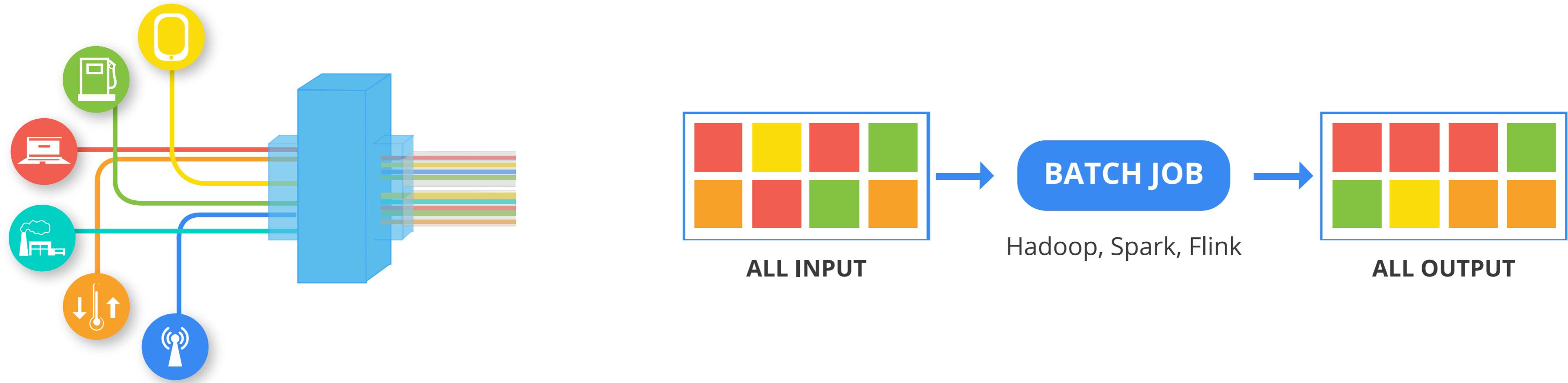
WHAT IS APACHE YARN?

FEATURES OF YARN

ARCHITECTURE OF YARN

COMPONENTS OF YARN

FEATURES OF YARN



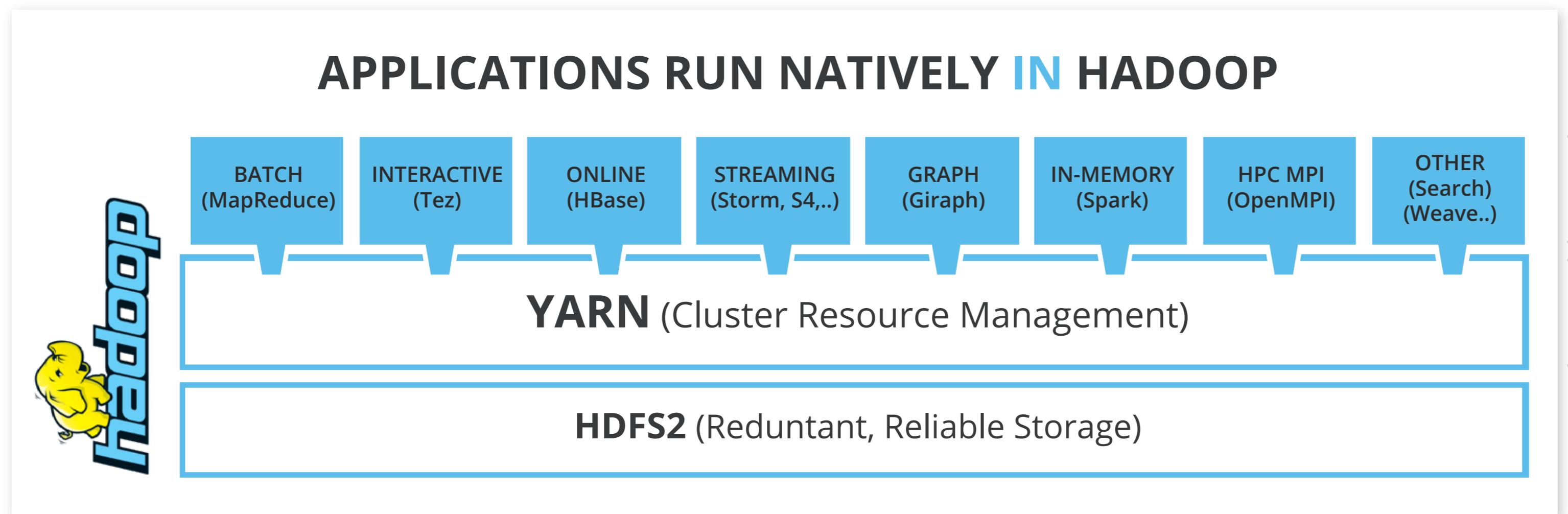
Allows multiple data processing engines such as

- ▶ Interactive SQL
- ▶ Real-time streaming
- ▶ Batch processing

to handle data stored in a single platform

FEATURES OF YARN

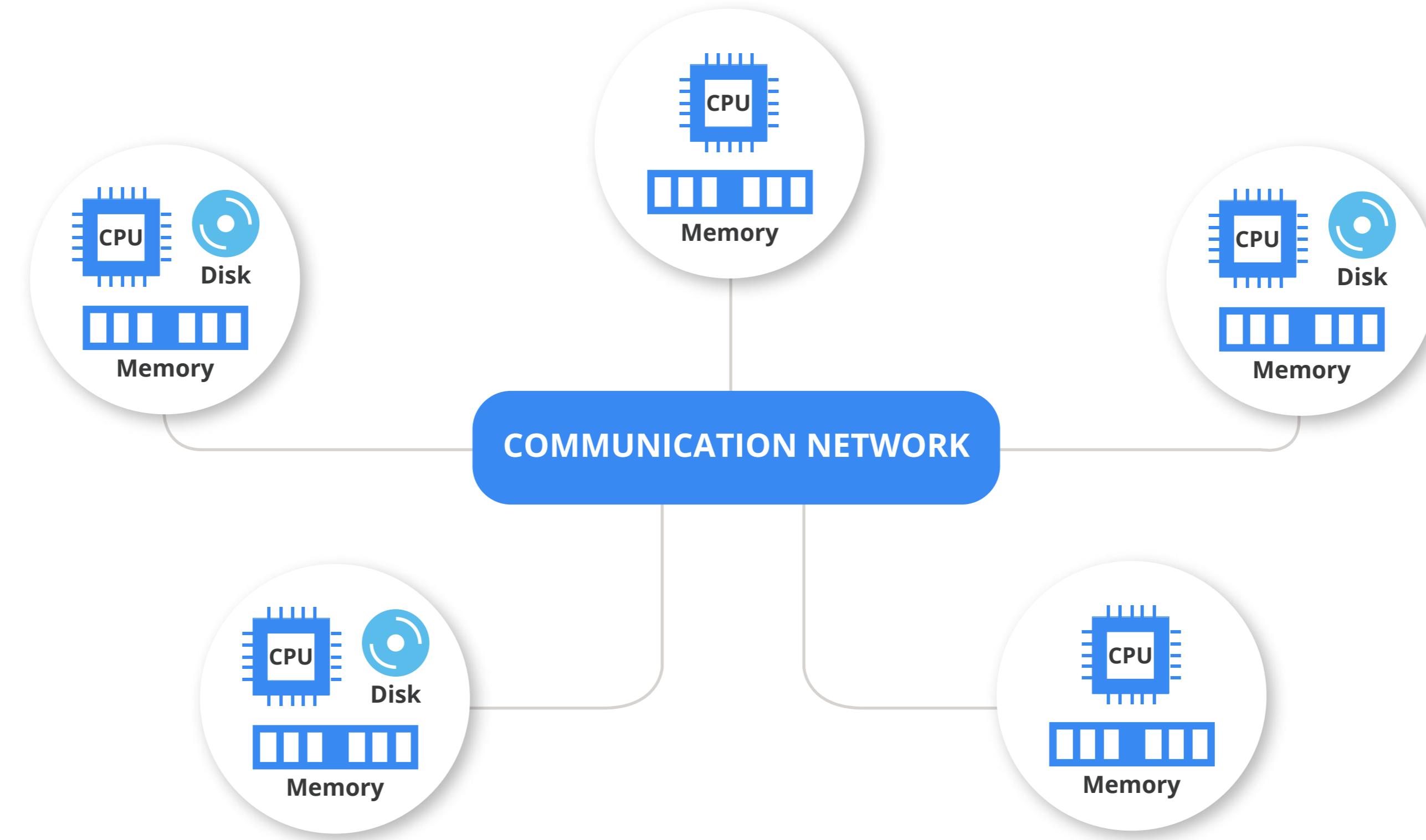
Takes *Hadoop beyond Batch*



More applications than MapReduce
YARN for better resource utilization

FEATURES OF YARN

YARN is *the distributed OS of Hadoop*



AGENDA

Introduction to YARN



CHALLENGES OF EARLIER HADOOP VERSIONS

WHAT IS APACHE YARN?

FEATURES OF YARN

ARCHITECTURE OF YARN

COMPONENTS OF YARN



Diving deep into YARN!

OVERVIEW

of YARN architecture

YARN architecture has ***four components*** namely:



RESOURCE MANAGER (RM)



Schedulers



CONTAINERS



NODE MANAGER (NM)



APPLICATION MASTER (AM)

OVERVIEW

of YARN architecture

Continued



RESOURCE MANAGER (RM)

- Keep track of available resources (primarily CPU, memory and data location) on each node of the cluster and schedules jobs
- Arbitrates system resources between competing applications
- Runs on master node
- Has a core component named ***scheduler***



NODE MANAGER (NM)

- Communicate with RM
- Runs on slave nodes

OVERVIEW OF YARN ARCHITECTURE

Continued



CONTAINERS

- Allocate a certain amount of resources (memory, CPU) on a slave node
- Created by the RM upon request
- Applications run in one or more containers



APPLICATION MASTER (AM)

- Requests resources (containers) from RM, tracks their status and monitors progress
- One per application
- Requests more containers from RM when required



What are schedulers in YARN?

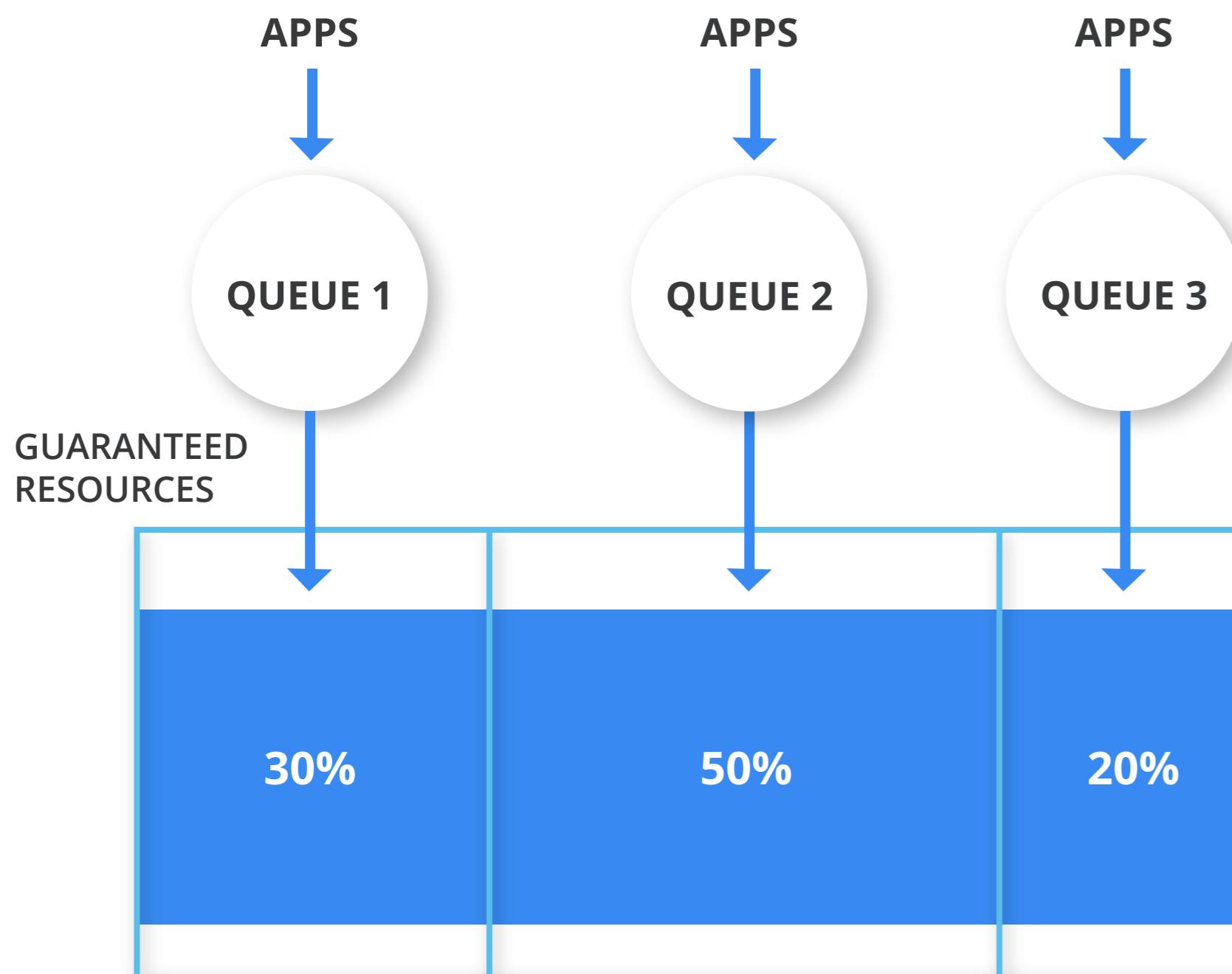
YARN SCHEDULER

Planner of Hadoop

- A Hadoop job consists of Map tasks and Reduce tasks
- Only one job in entire cluster => it occupies cluster
- Multiple customers with multiple jobs
 - ▶ Users/jobs = "tenants"
 - ▶ Multi-tenant system
- Need a way to schedule all these jobs (and their constituent tasks)
- Need to be fair across the different tenants
- Hadoop YARN has two popular schedulers
 - ▶ Hadoop Capacity Scheduler
 - ▶ Hadoop Fair Scheduler

HADOOP SCHEDULER TYPE 1

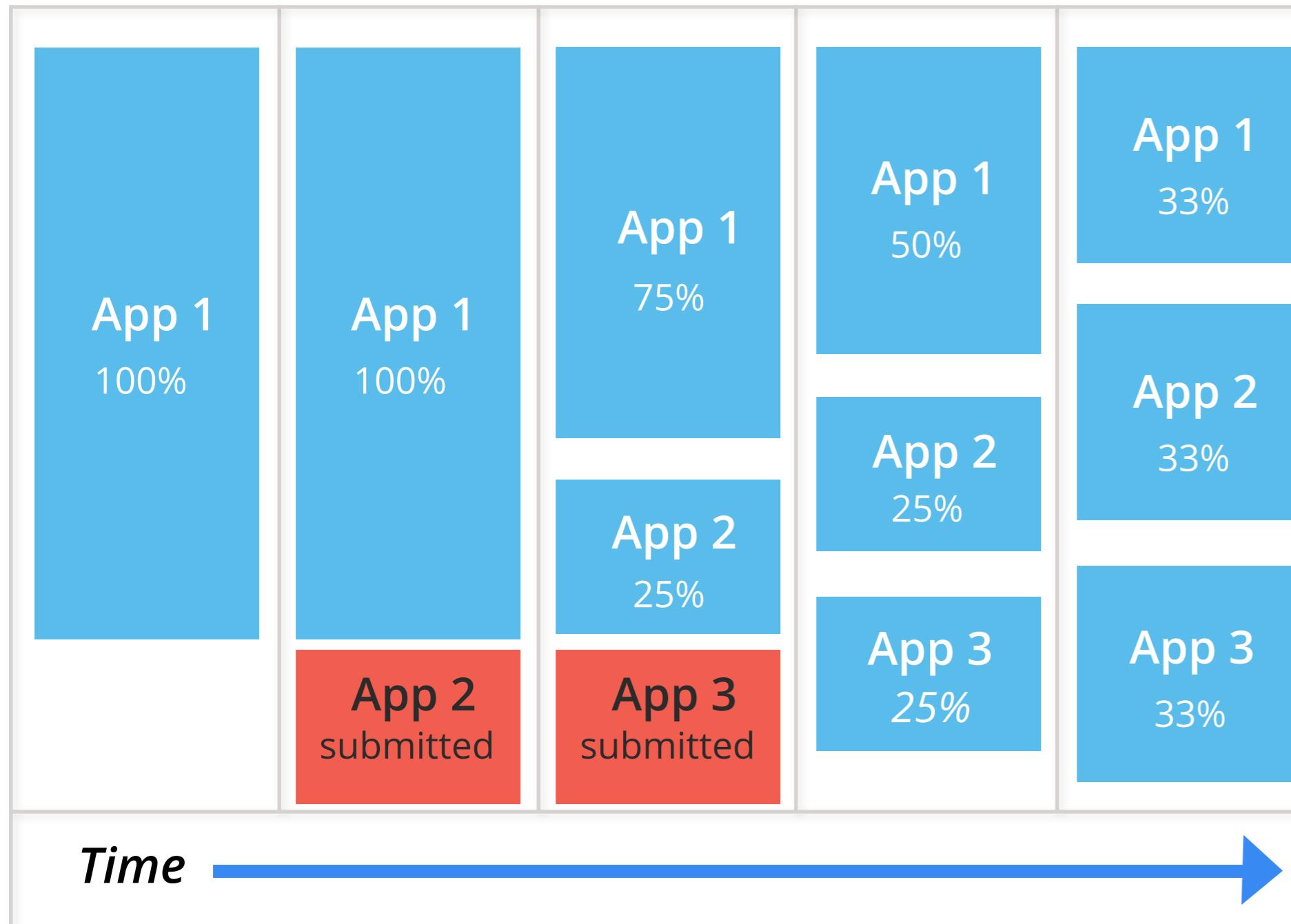
Capacity scheduler



- ▶ Contains multiple queues
- ▶ Each queue contains multiple jobs
- ▶ Each queue guaranteed some portion of the cluster capacity
 - Eg:
 - Queue 1 is given 80% of cluster
 - Queue 2 is given 20% of cluster
 - Higher-priority jobs go to Queue 1
- ▶ For jobs within same queue, FIFO typically used
- ▶ Administrators can configure queues

HADOOP SCHEDULER TYPE 2

Fair scheduler



- ▶ All jobs get equal share of resources
- ▶ When only one job present, occupies entire cluster
- ▶ As other jobs arrive, each job given equal % of cluster
 - Eg:
 - *Each job might be given equal number of cluster-wide YARN containers*
 - *Each container = 1 task of job*

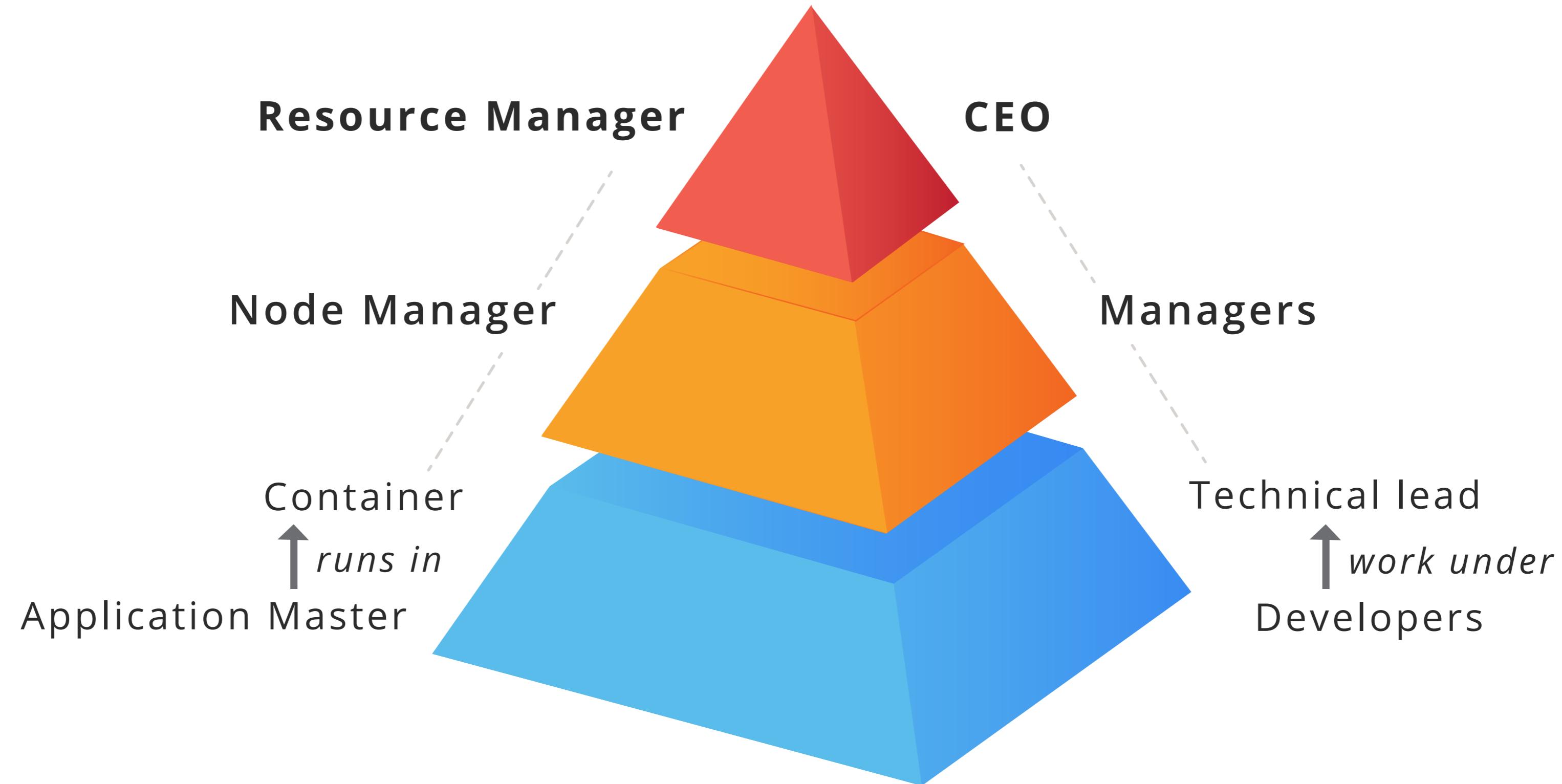
HADOOP SCHEDULER TYPE 2

Fair scheduler

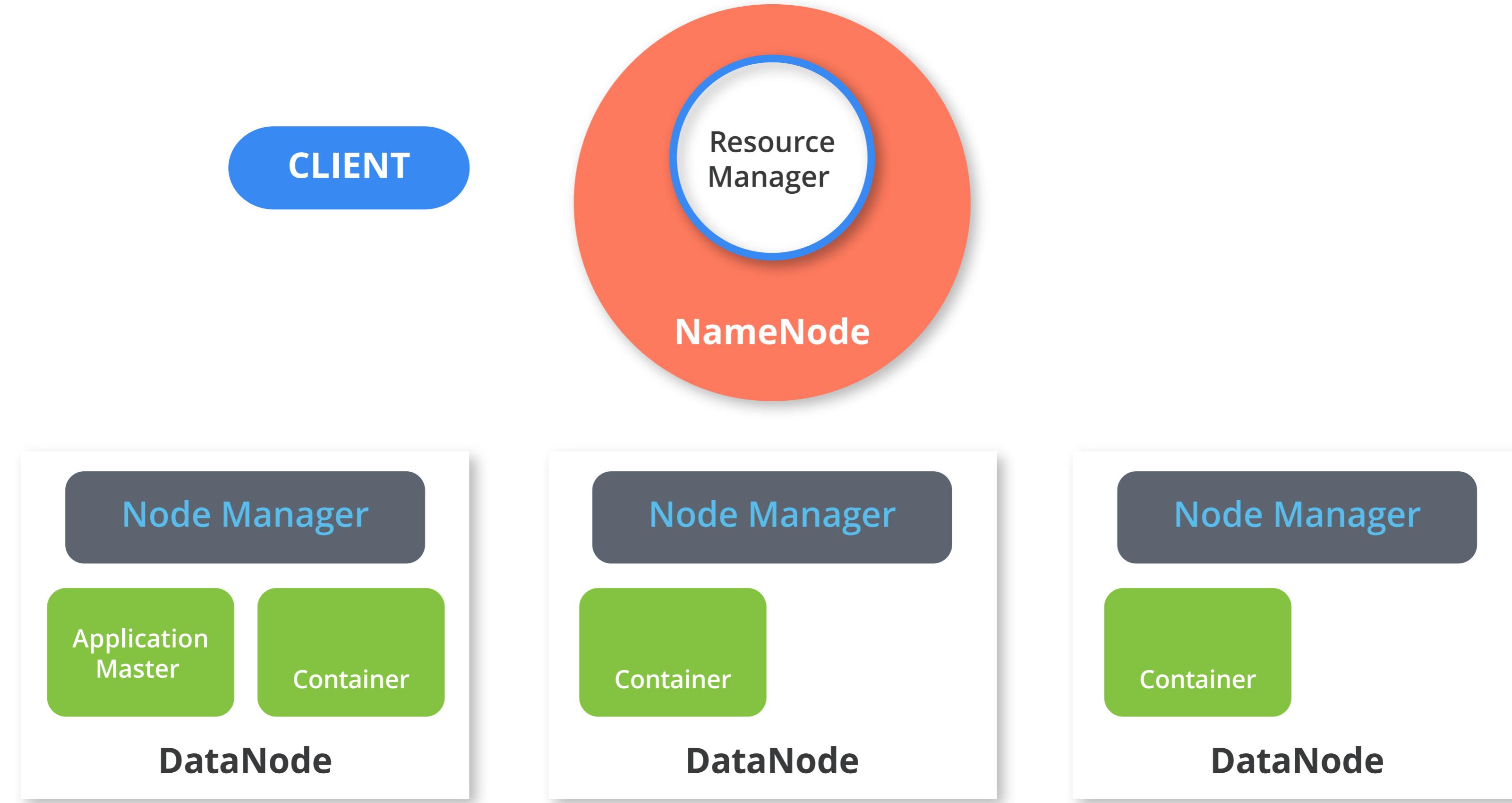
- ➊ Divides cluster into pools (jobs)
 - *Typically one pool per user*
- ➋ Resources divided equally among pools
 - *Gives each user fair share of cluster*
- ➌ Within each pool, can use either
 - *Fair share scheduling, or*
 - *FIFO/FCS*
 - *(Configurable)*

ANALOGY

YARN architecture *vs* Corporate Flow

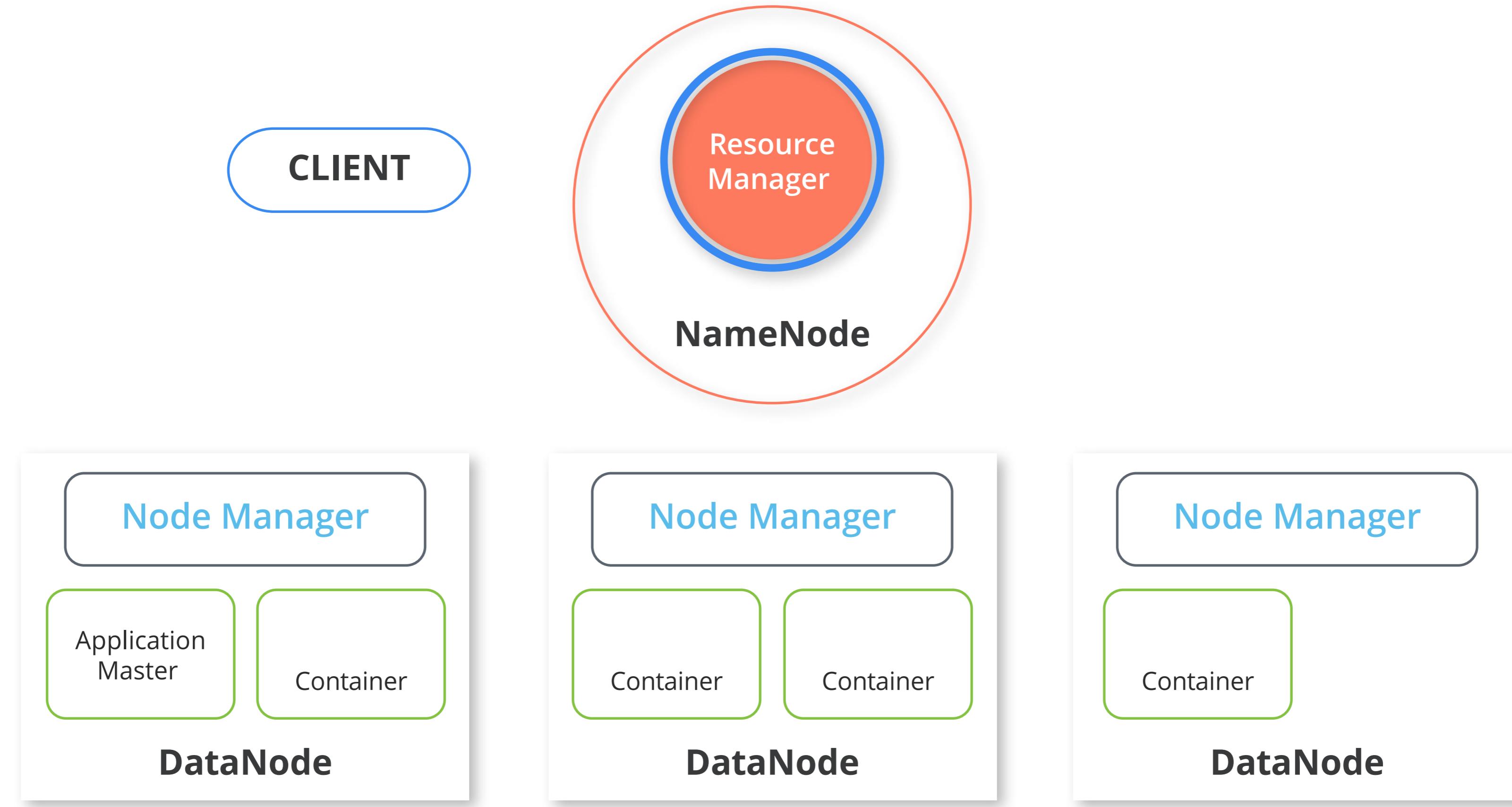


HOW THESE COMPONENTS FIT into Hadoop cluster?



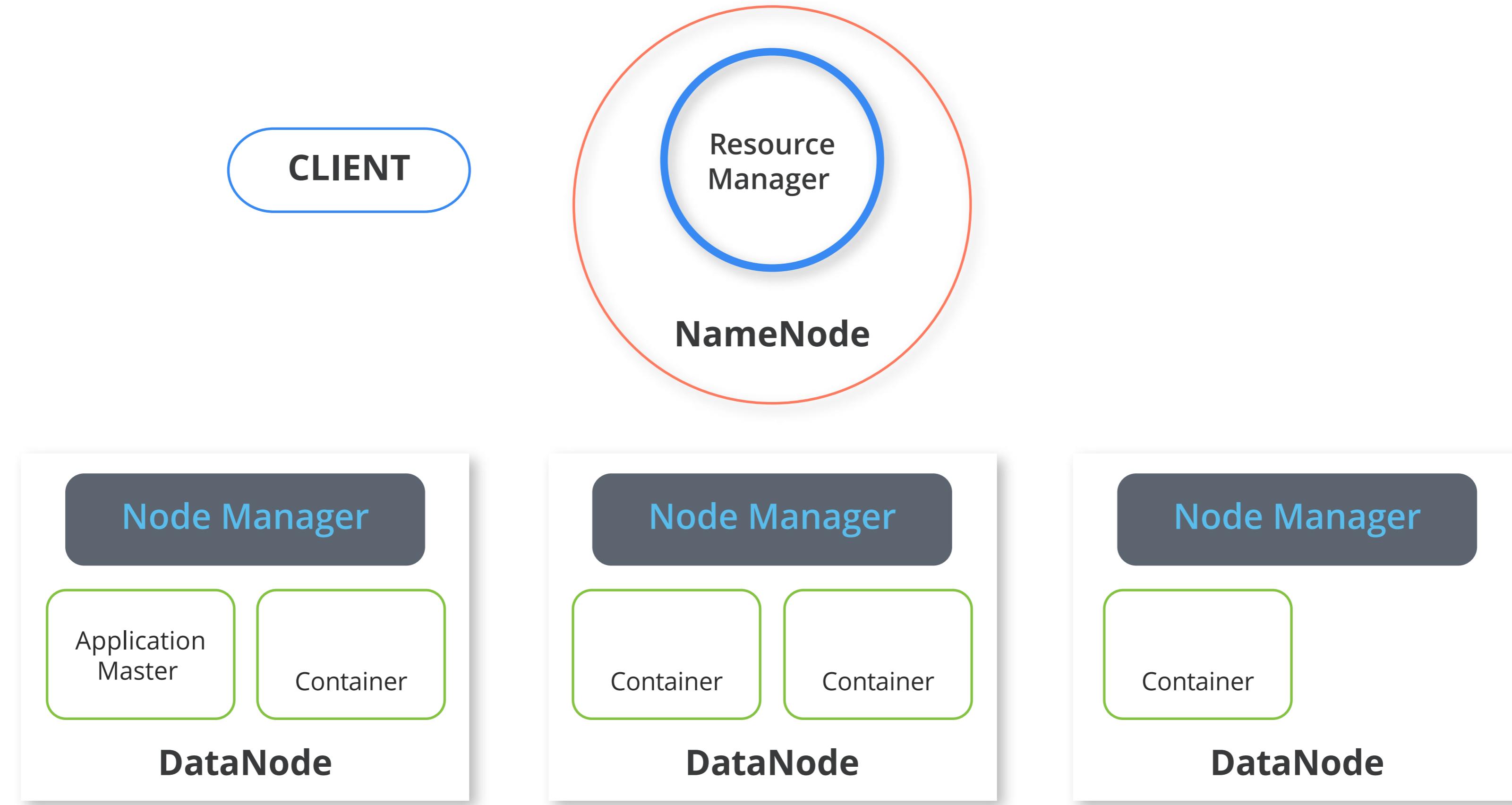
RESOURCE MANAGER

The brain of the entire process



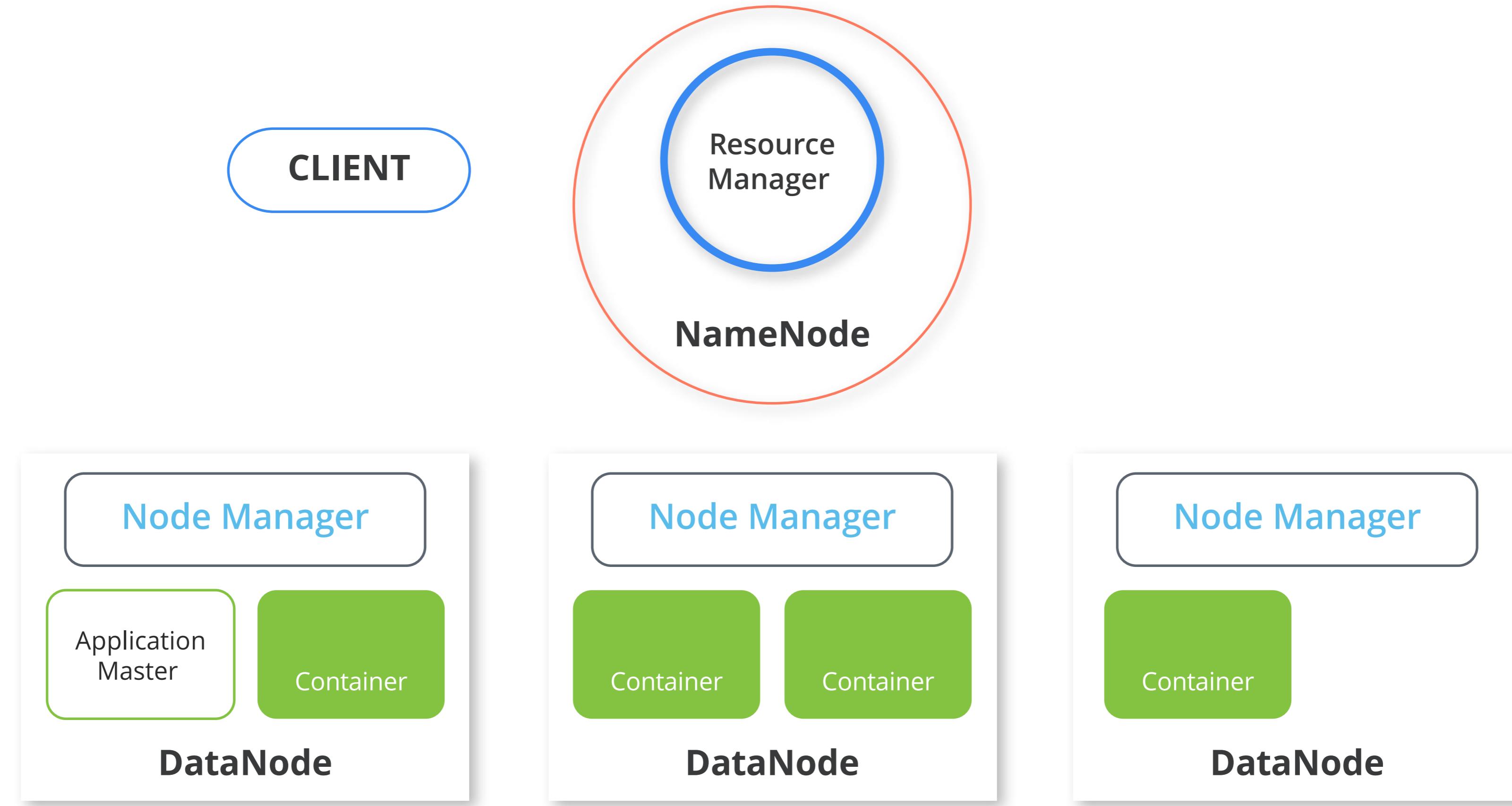
NODE MANAGER

Supervisor of the process



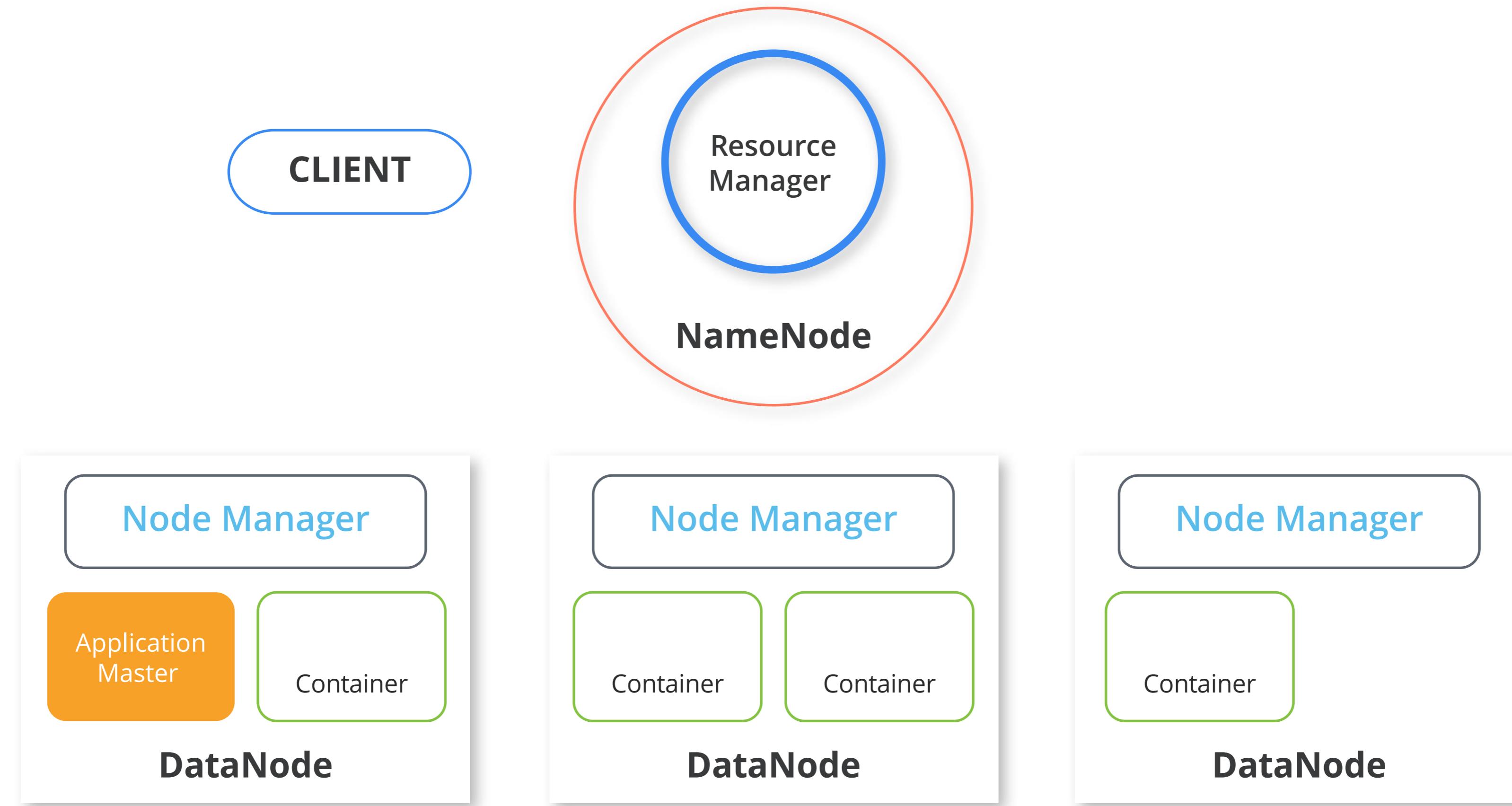
CONTAINERS

Place of execution



APPLICATION MASTER

Supplier of a process





How a job/application is submitted to YARN?

STARTING AN APPLICATION

in 6 easy steps

Step 1

Client submits a job/application

Step 2

RM asks Node Manager to launch Application Master

Step 3

Node Manager launches Application Master

Step 4

Application Master requests containers

Step 5

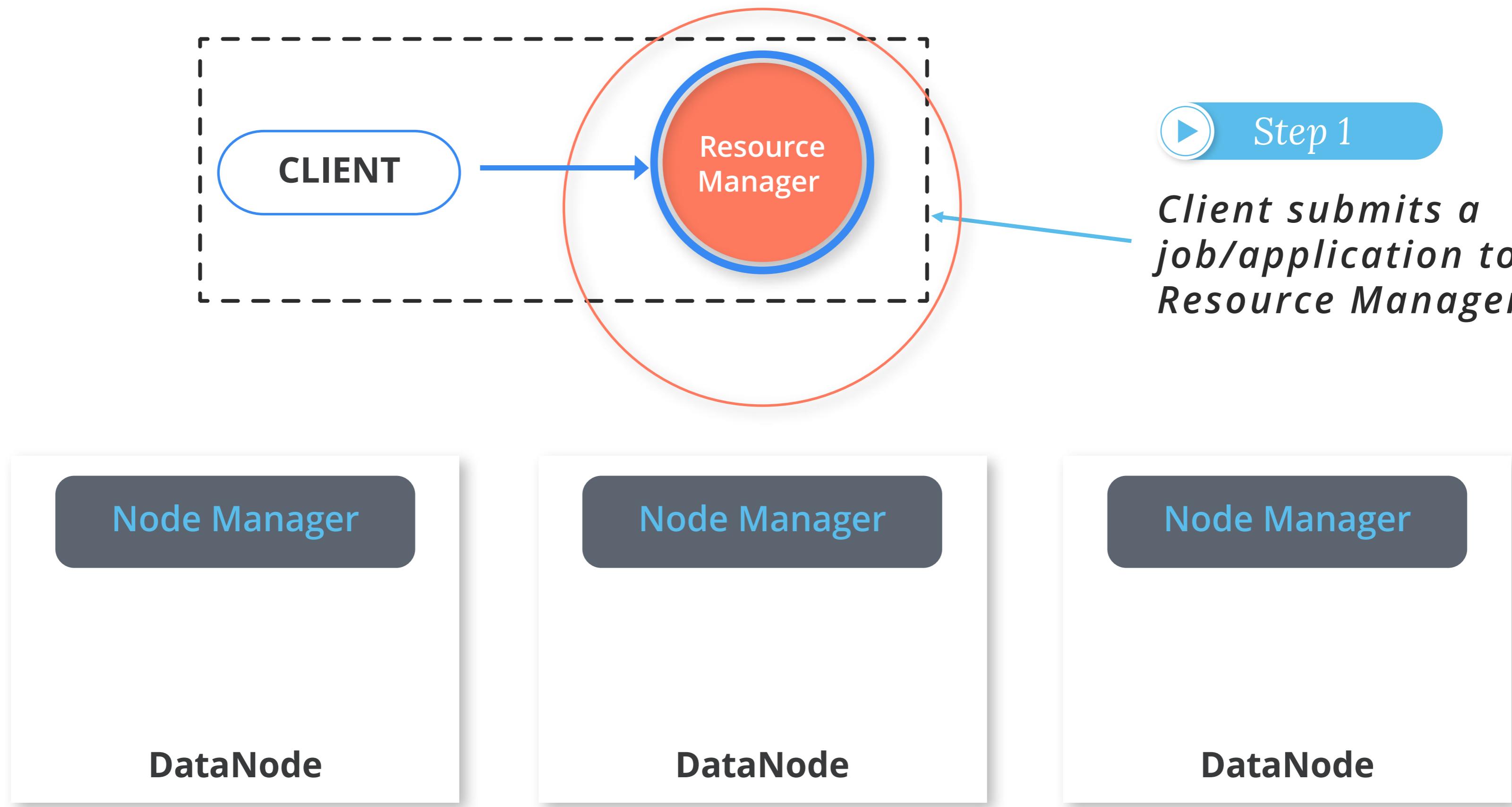
RM assigns containers

Step 6

Application Master launches application

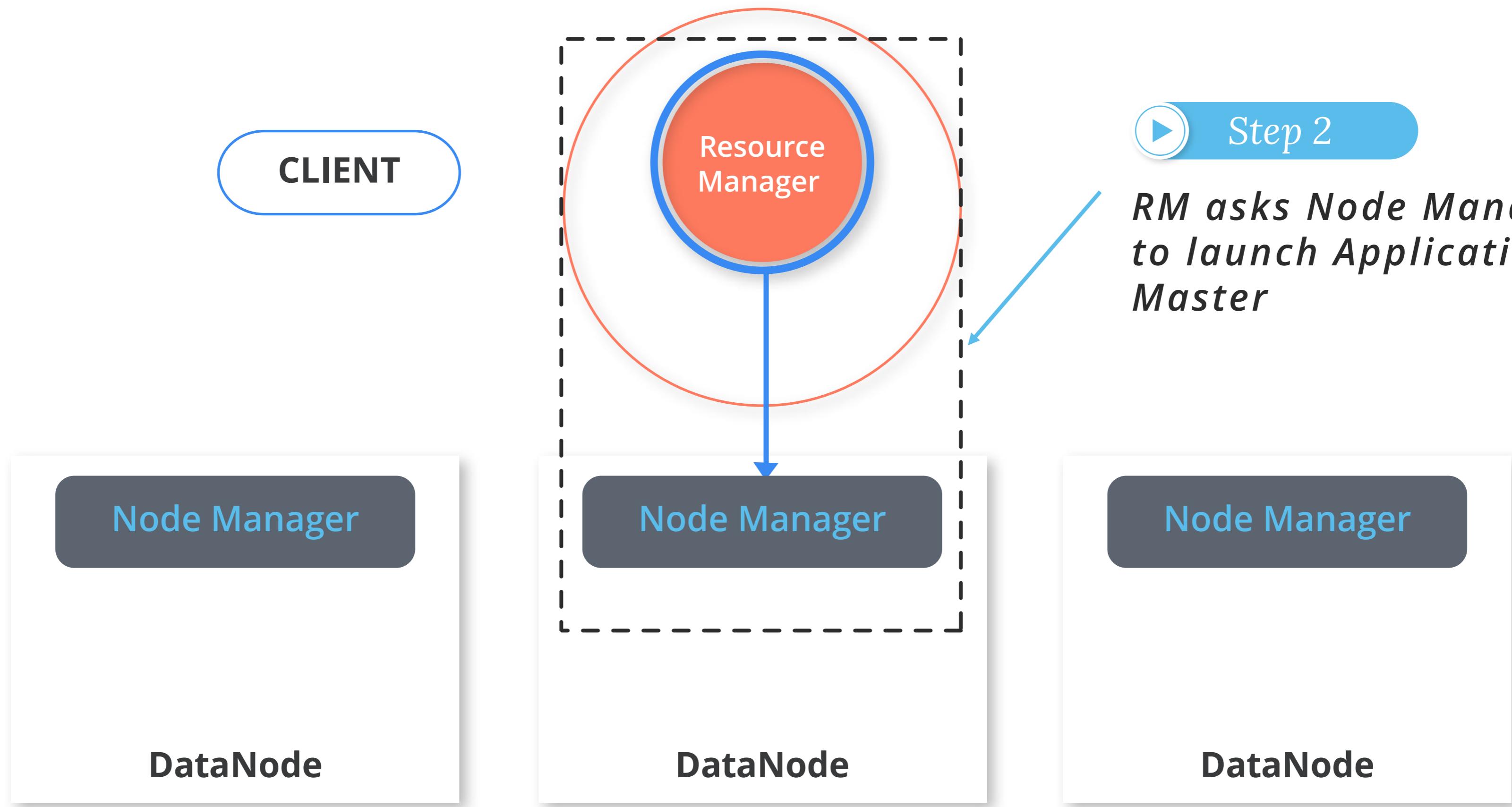
STEP 1

Client submits a job/application to RM



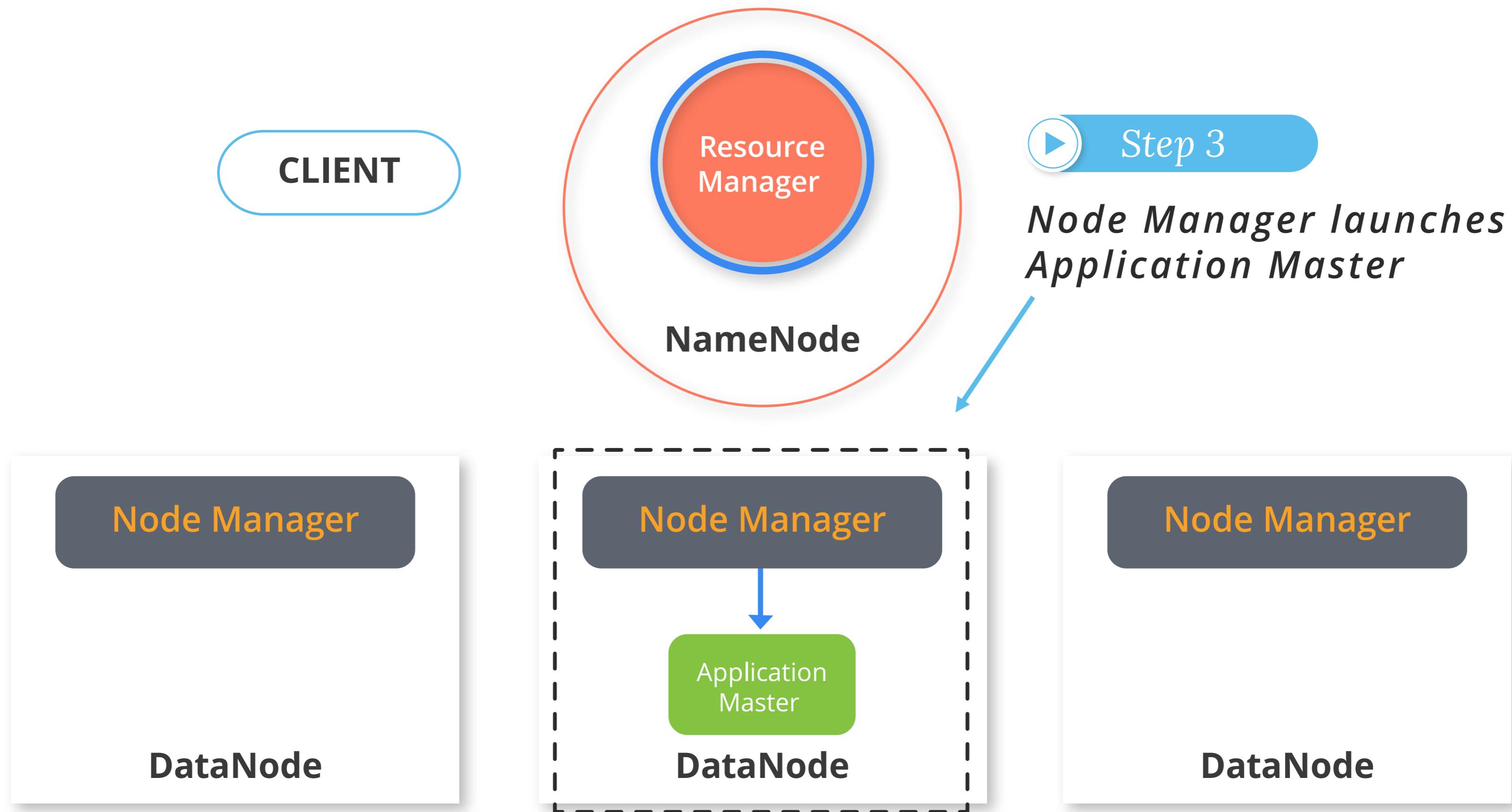
STEP 2

RM asks Node Manager to launch Application Master



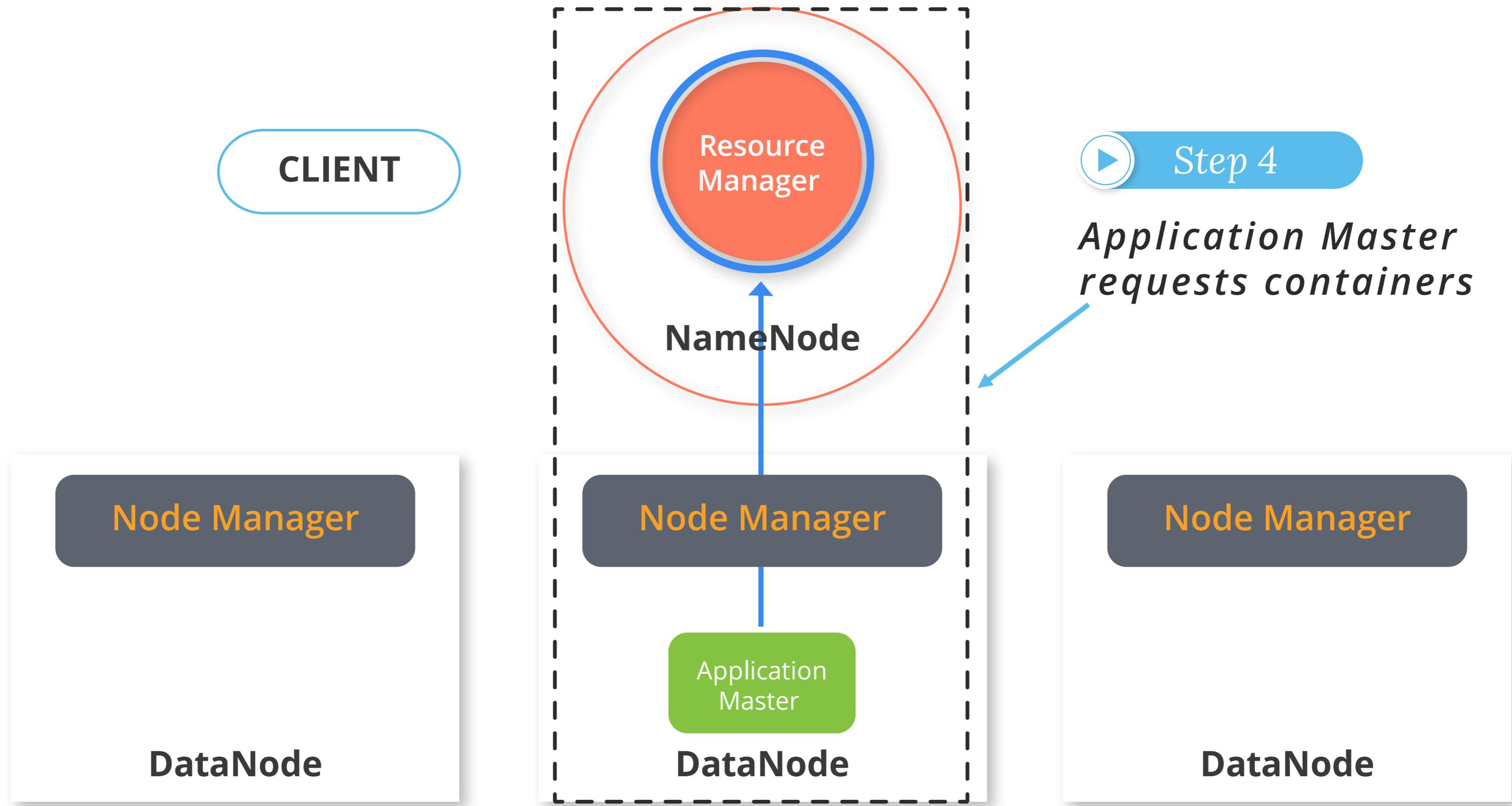
STEP 3

Node Manager launches Application Master



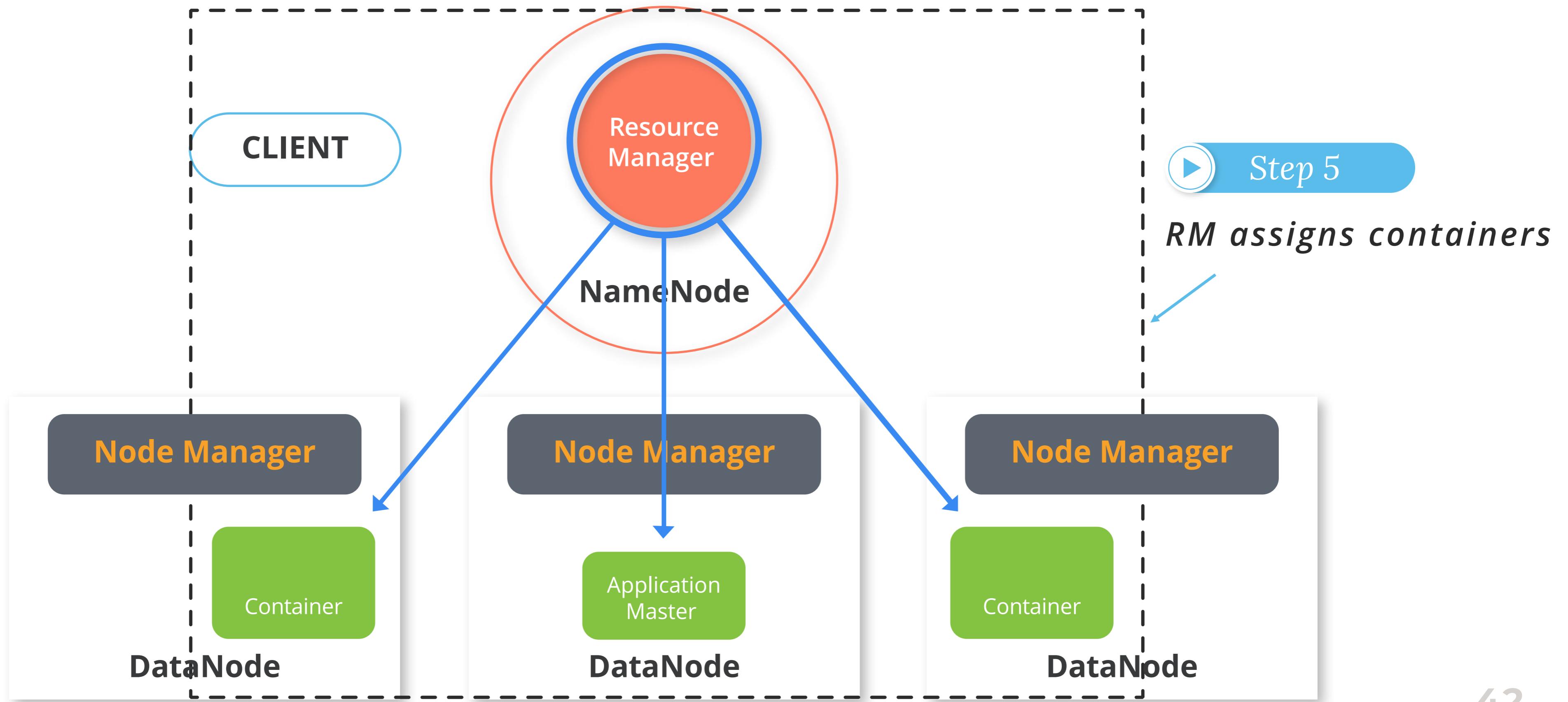
STEP 4

Application Master requests containers



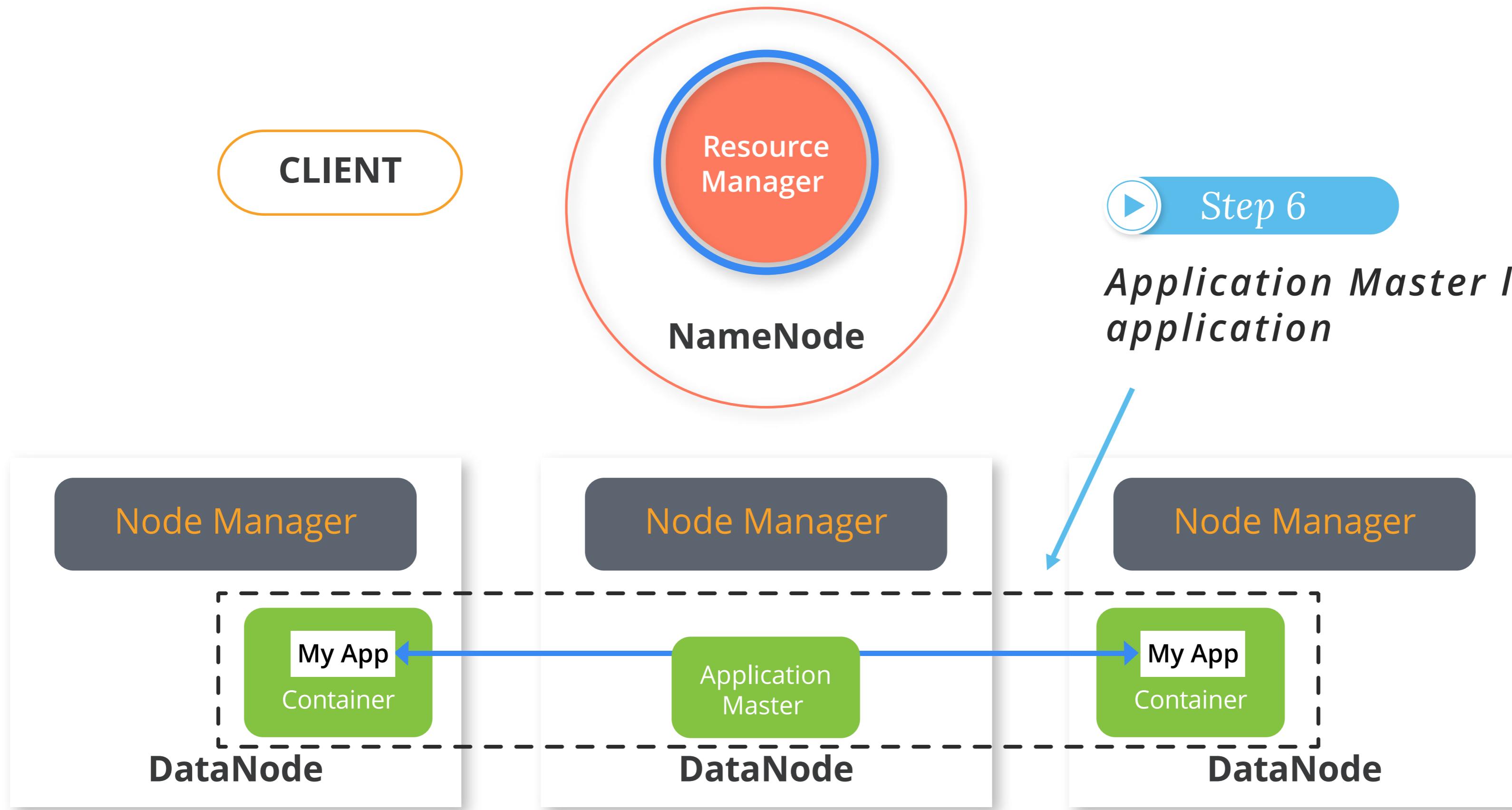
STEP 5

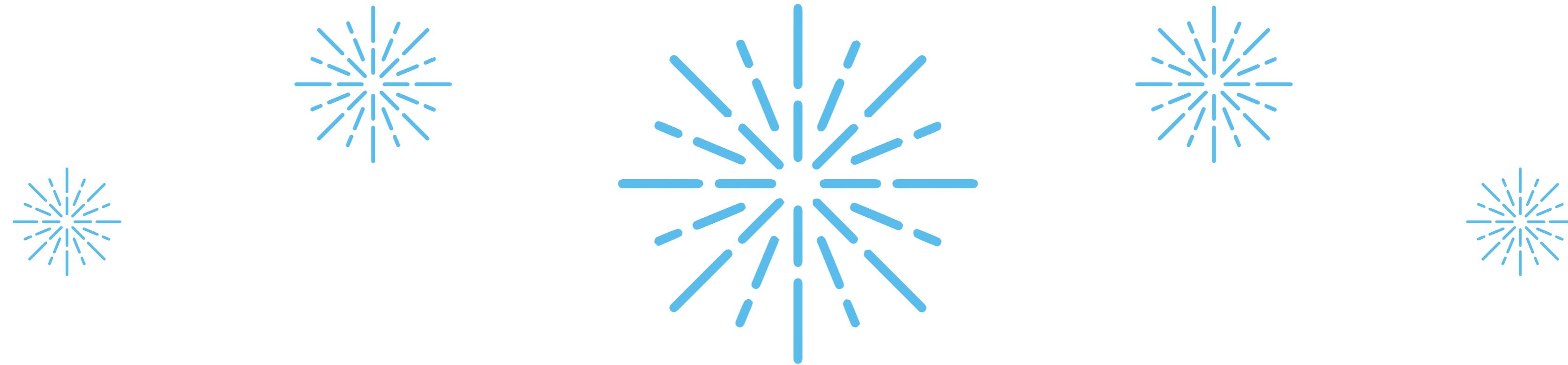
RM launches containers in DataNode and container IDs to AM



STEP 6

Application Master launches application in the containers





Congratulations!
Your Application is now running



Now, we will have a look at the
Resource manager UI

RESOURCE MANAGER UI

<https://ipaddress:8088>

The screenshot shows the 'Nodes of the cluster' page of the Hadoop Resource Manager UI. The sidebar on the left has a 'Cluster' section with links for About, Nodes (which is highlighted with a red box), Node Labels, Applications, and Scheduler. Below these are buttons for Tools and Help. The main content area is titled 'Nodes of the cluster'. It displays various metrics for the cluster, such as Cluster Metrics (Apps Submitted: 1, Apps Pending: 0, Apps Running: 1, Apps Completed: 0, Containers Running: 0, Memory Used: 0 B, Memory Total: 6 GB, Memory Reserved: 0 B, VCores Used: 0, VCores Total: 4, VCores Reserved: 0) and Cluster Nodes Metrics (Active Nodes: 1, Decommissioning Nodes: 0, Decommissioned Nodes: 0, Lost Nodes: 0, Unhealthy Nodes: 0, Rebooted Nodes: 0, Shutdown Nodes: 0). The Scheduler Metrics section shows Capacity Scheduler as the type, Scheduling Resource Type as [MEMORY], and allocation ranges <memory:32, vCores:1> to <memory:6144, vCores:4>. The main table lists one node: /default-rack, RUNNING, ip-172-31-4-20.ap-south-1.compute.internal:8041, ip-172-31-4-20.ap-south-1.compute.internal:8042, Fri Mar 23 13:25:04 +0000 2018. The table includes columns for Node Labels, Rack, Node State, Node Address, Node HTTP Address, Last health-update, Health-report, Containers, Mem Used, Mem Avail, VCores Used, VCores Avail, and Version.

Link to Node Manager UI

RESOURCE MANAGER UI

Applications

CLUSTER OVERVIEW

The screenshot shows the Hadoop Resource Manager UI with the following sections:

- Cluster Metrics:** Shows 1 application submitted, 0 pending, 0 running, 1 completed, 0 containers running, 0 B memory used, 6 GB total, 0 B reserved, and 0 vcores.
- Cluster Nodes Metrics:** Shows 1 active node, 0 decommissioning, 0 decommissioned, 0 lost nodes, 0 unhealthy nodes, and 0 reboots.
- Scheduler Metrics:** Shows Capacity Scheduler selected, Scheduling Resource Type as [MEMORY], Minimum Allocation as <memory:32, vCores:1>, and Maximum Allocation as <memory:6144, vCores:4>.
- Applications:** A table listing one application entry:

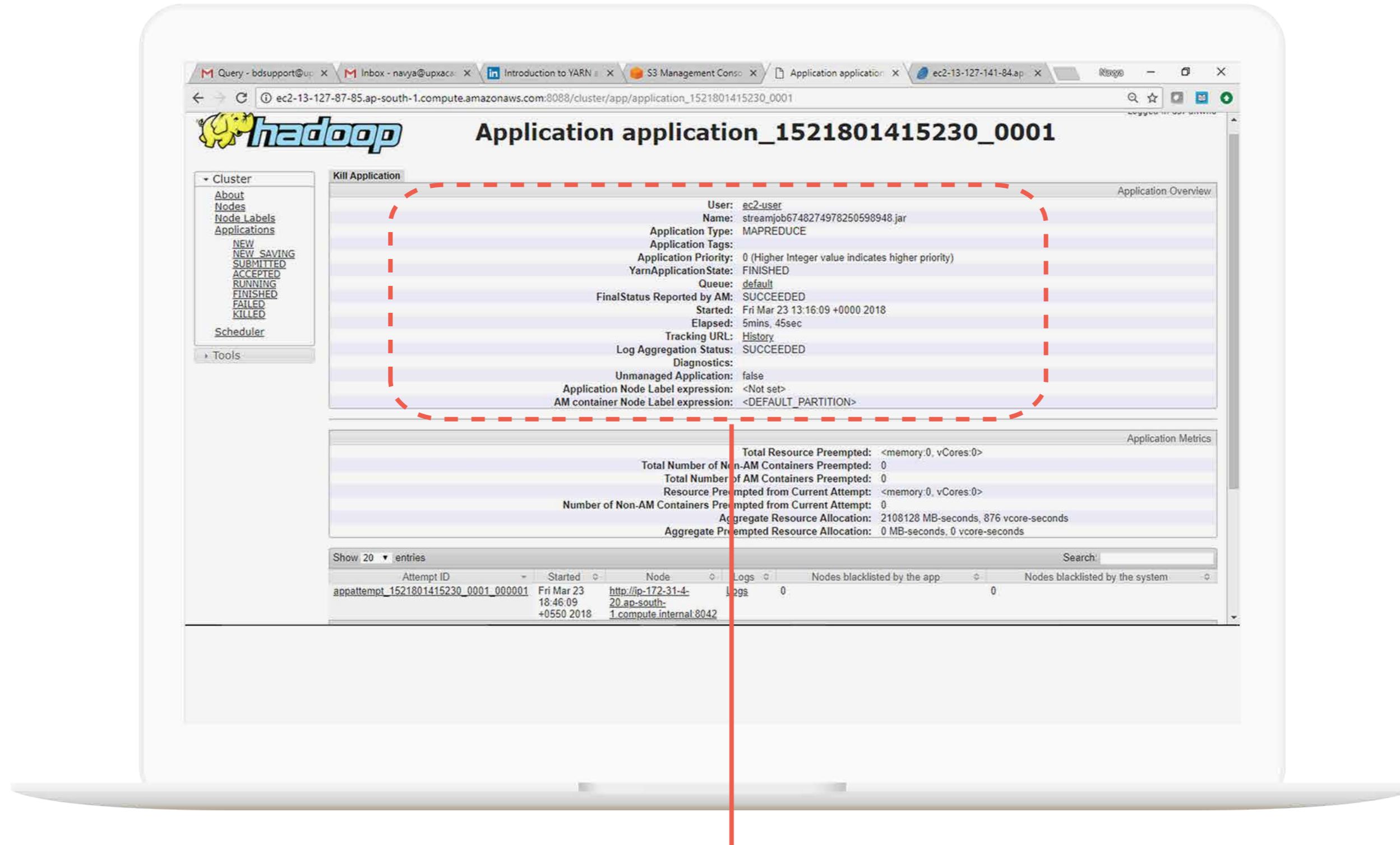
ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU	Allocated Mem
application_1521801415230_0001	ec2-user	streamjob6748274978250598948.jar	MAPREDUCE	default	0	Fri Mar 23 18:46:09 +0550 2018	Fri Mar 23 18:51:54 +0550 2018	FINISHED	SUCCEEDED	N/A	N/A	N/A

A red dashed box highlights the "Applications" section in the sidebar, and a red arrow points from the "Link to Application details" button at the bottom to the application row in the table.

Link to Application details

RESOURCE MANAGER UI

Application Details



Link to Application details

MR APPMASTER UI

Tasks

The screenshot shows the Hadoop MRAppMaster UI interface. At the top, it displays the URL `http://rmhost:8088/proxy/appid/mapreduce/job/jobid`. The main title is "MapReduce Job job_1384200217415_0009". On the left, there's a navigation sidebar with a red dashed box highlighting the "Job" section, which includes links for Overview, Counters, Configuration, Map tasks, Reduce tasks, and AM Logs. The right side contains two main sections: "Job Overview" and "ApplicationMaster". The "Job Overview" section shows details like Job Name: Process Logs, State: RUNNING, Uberized: false, Started: Tue Nov 12 13:54:50 EST 2013, and Elapsed: 1mins, 40sec. The "ApplicationMaster" section shows a table of task types and their counts. Below these sections, there's a footer with the Cloudera logo and copyright information.

MRAppMaster UI: Tasks

`http://rmhost:8088/proxy/appid/mapreduce/job/jobid`

Logged in as: dr.who

hadoop

MapReduce Job
job_1384200217415_0009

Job Overview

Job Name:	Process Logs
State:	RUNNING
Uberized:	false
Started:	Tue Nov 12 13:54:50 EST 2013
Elapsed:	1mins, 40sec

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Tue Nov 12 13:54:43 EST 2013	localhost.localdomain:8042	logs

Task Type	Progress	Total	Pending	Running	Complete
Map	[progress bar]	4	0	0	4
Reduce	[progress bar]	1	0	1	0

Attempt Type	New	Running	Failed	Killed	Successful
Maps	0	0	0	0	4
Reduces	0	1	0	0	0

cloudera

© Copyright 2010-2013 Cloudera. All rights reserved. Not to be reproduced without prior written consent.

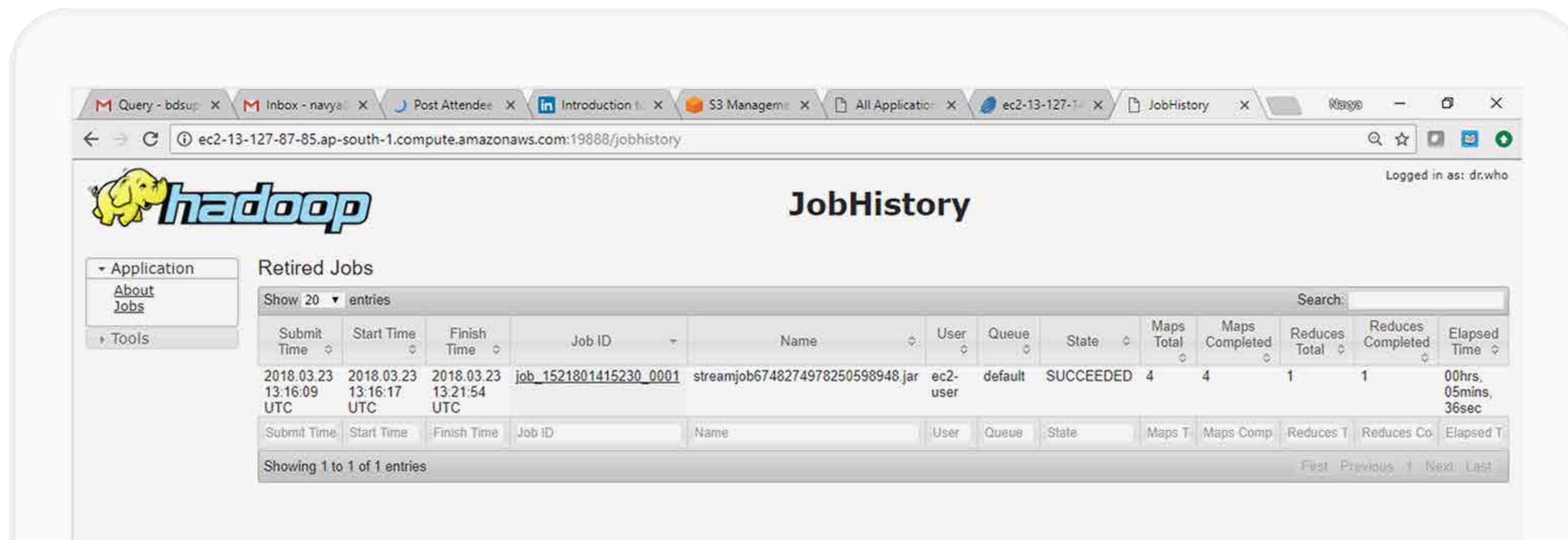
45

MR JOB HISTORY SERVER

► YARN does not keep track of job history

► MapReduce job history server

- Archives jobs metrics & metadata
- Can be accessed through Job history UI or HUE



The screenshot shows a web browser window with the URL `ec2-13-127-87-85.ap-south-1.compute.amazonaws.com:19888/jobhistory`. The page title is "JobHistory". On the left, there's a sidebar with a "hadoop" logo, a "Retired Jobs" section, and navigation links for "Application", "About", "Jobs", and "Tools". The main content area displays a table titled "Retired Jobs" with one entry:

Submit Time	Start Time	Finish Time	Job ID	Name	User	Queue	State	Maps Total	Maps Completed	Reduces Total	Reduces Completed	Elapsed Time
2018.03.23 13:16:09 UTC	2018.03.23 13:16:17 UTC	2018.03.23 13:21:54 UTC	job_1521801415230_0001	streamjob6748274978250598948.jar	ec2-user	default	SUCCEEDED	4	4	1	1	00hrs, 05mins, 36sec

At the bottom of the table, there are filters for "Submit Time", "Start Time", "Finish Time", "Job ID", "Name", "User", "Queue", "State", "Maps T", "Maps Comp", "Reduces T", "Reduces Co", and "Elapsed T". The footer of the table says "Showing 1 to 1 of 1 entries".

DISCLAIMER



*All images used here in the deck are obtained from various sources online, through Google image search. We do not own any copyright claims.
The deck is prepared for educational purpose and no copyright violations were intended.*



www.upxacademy.com

bdsupport@upxacademy.com

1800-123-1260

Follow Us On

