BITS Pilani presentation

BITS Pilani
Pilani Campus

Dr. Vivek V. Jog
Dept. Of Computer Engineering

innovate achieve lead

BITS Pilani
Pilani Campus

Big Data Systems (S1-24_CCZG522)
Lecture No.3

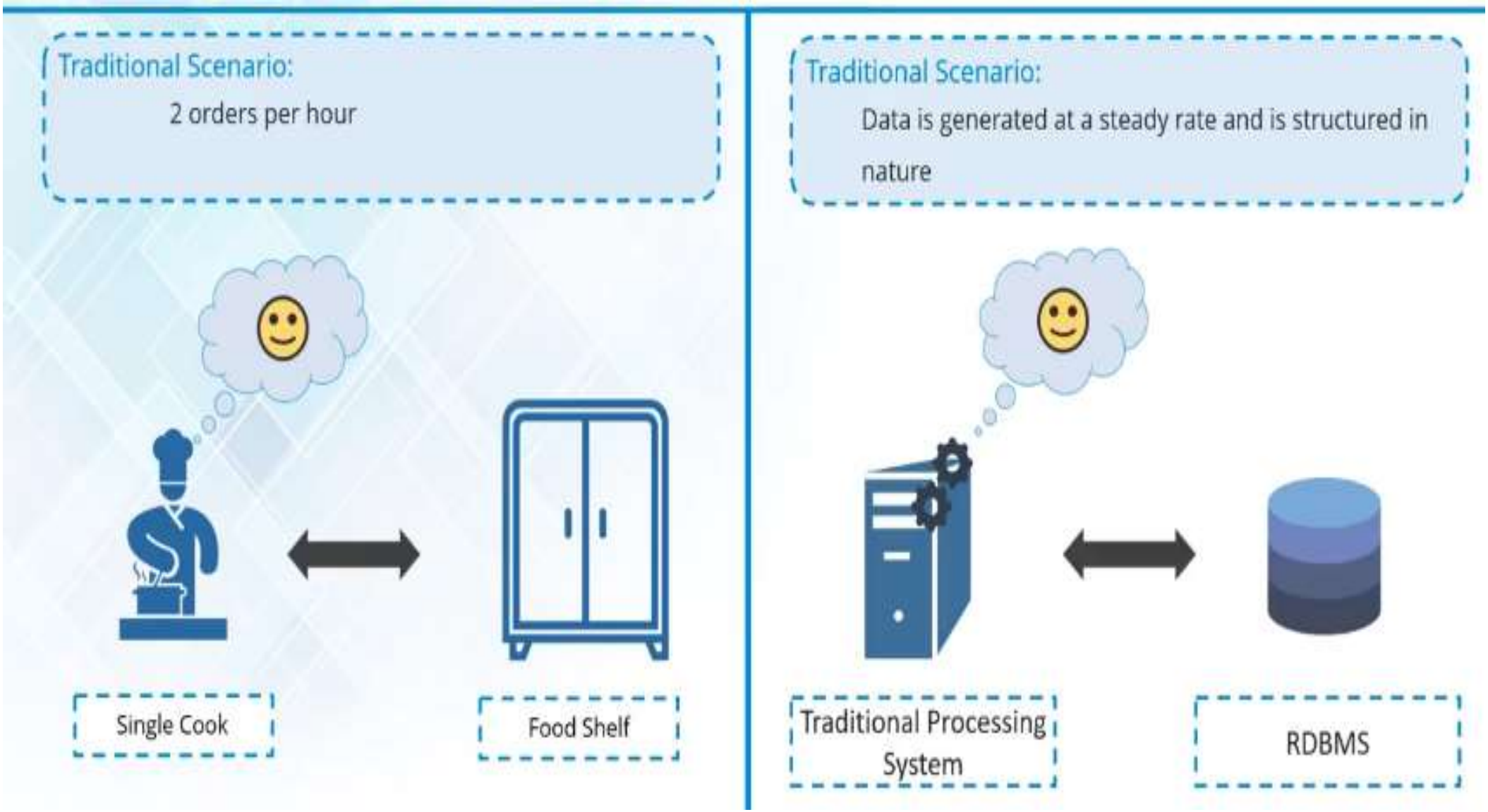
INDEX

- Big Data Growth Drivers
- What is Big Data?
- Hadoop Introduction
- Hadoop Master/Slave Architecture
- Hadoop Core Components
- HDFS Data Blocks
- HDFS Read/Write Mechanism
- What is MapReduce
- MapReduce Program
- MapReduce Job Workflow
- Hadoop Ecosystem

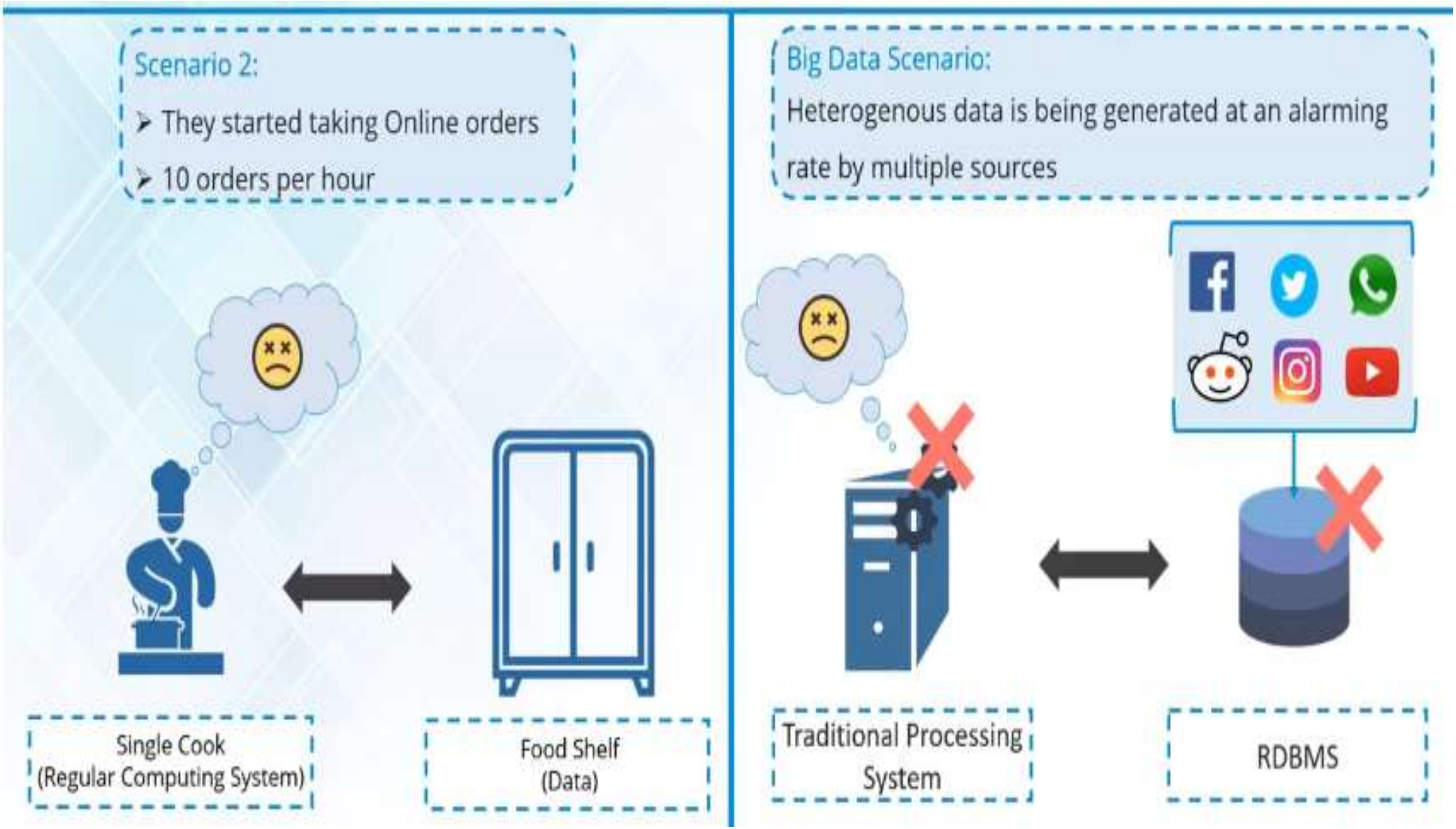
Five V's

"Big data is the term for a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications"

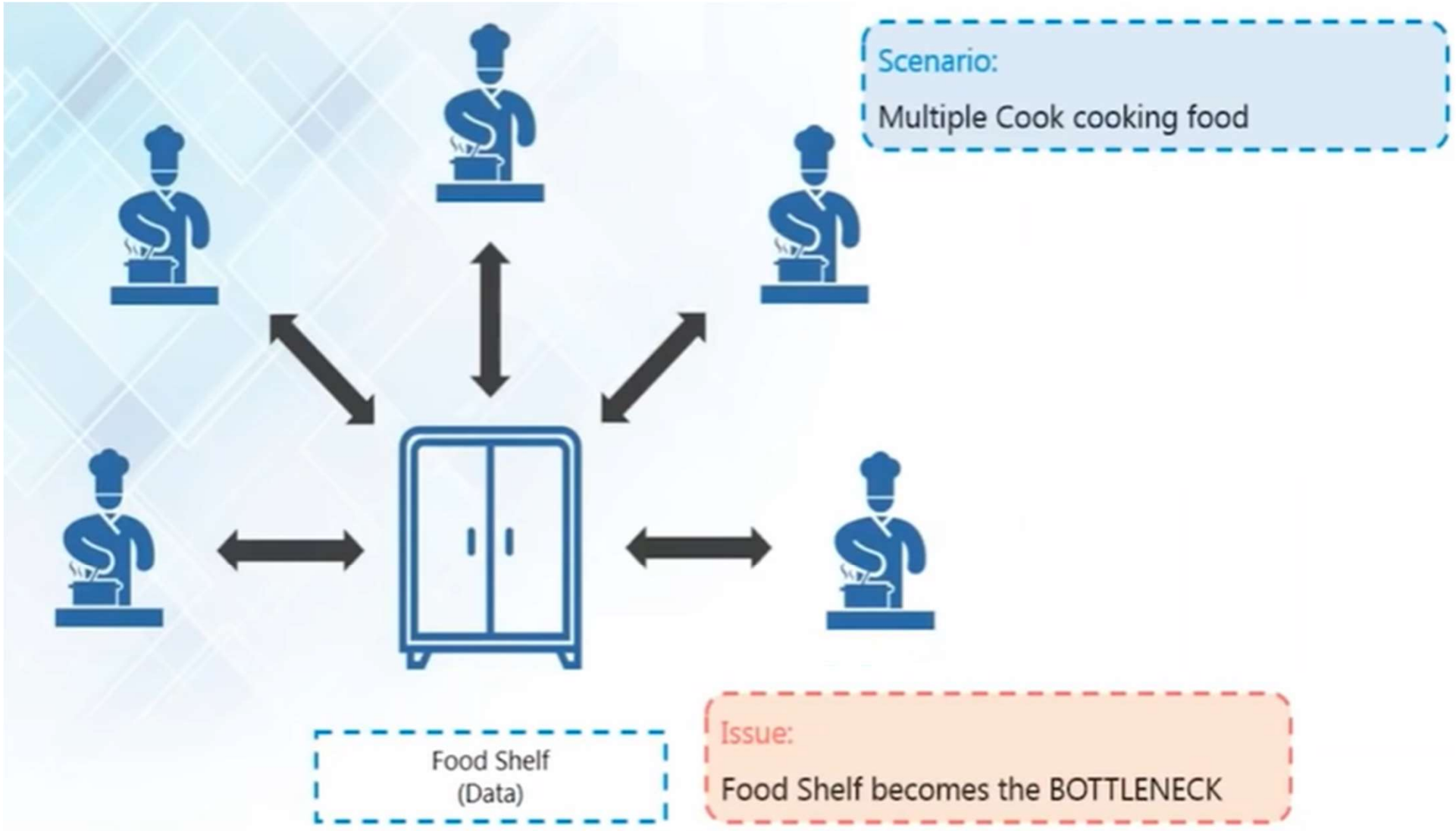
Traditional System V/s Big Data



Contd..


Scenario 2:
They started taking Online orders
10 orders per hour
Single Cook
(Regular Computing System)
Food Shelf
(Data)
Big Data Scenario:
Heterogenous data is being generated at an alarming rate by multiple sources
Traditional Processing System
RDBMS

Bottleneck


Scenario:
Multiple Cook cooking food
Food Shelf
(Data)
Issue:
Food Shelf becomes the BOTTLENECK

Bottleneck

Scenario:

Multiple Processing Unit for data processing

Issue:

Bringing data to processing generated lots of Network overhead

More number of orders = More number of cooks

Scenario:

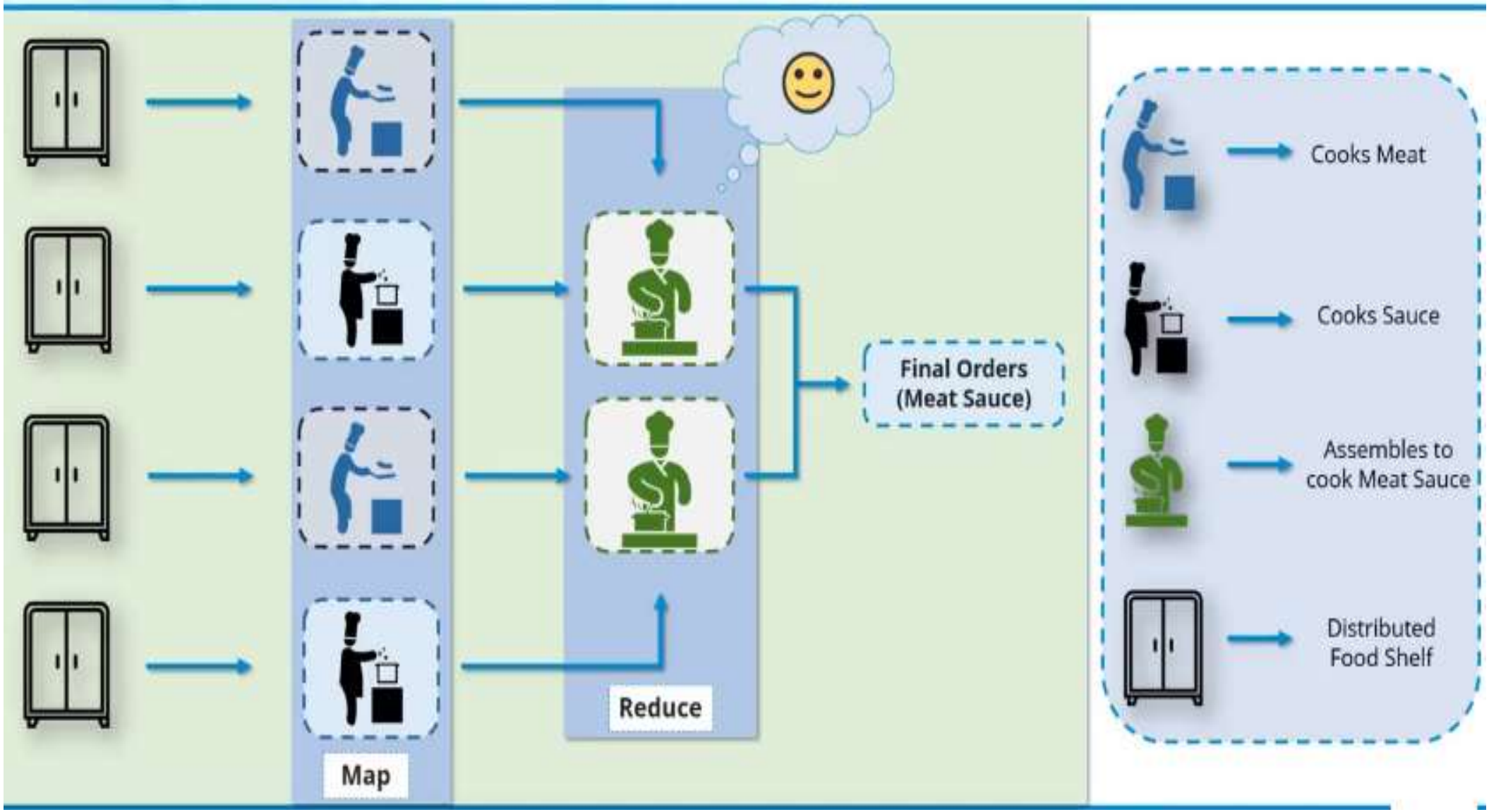
Multiple Processing Unit for data processing

Issue:

Bringing data to processing generated lots of Network overhead

Check for the Bottleneck

Effective Solution

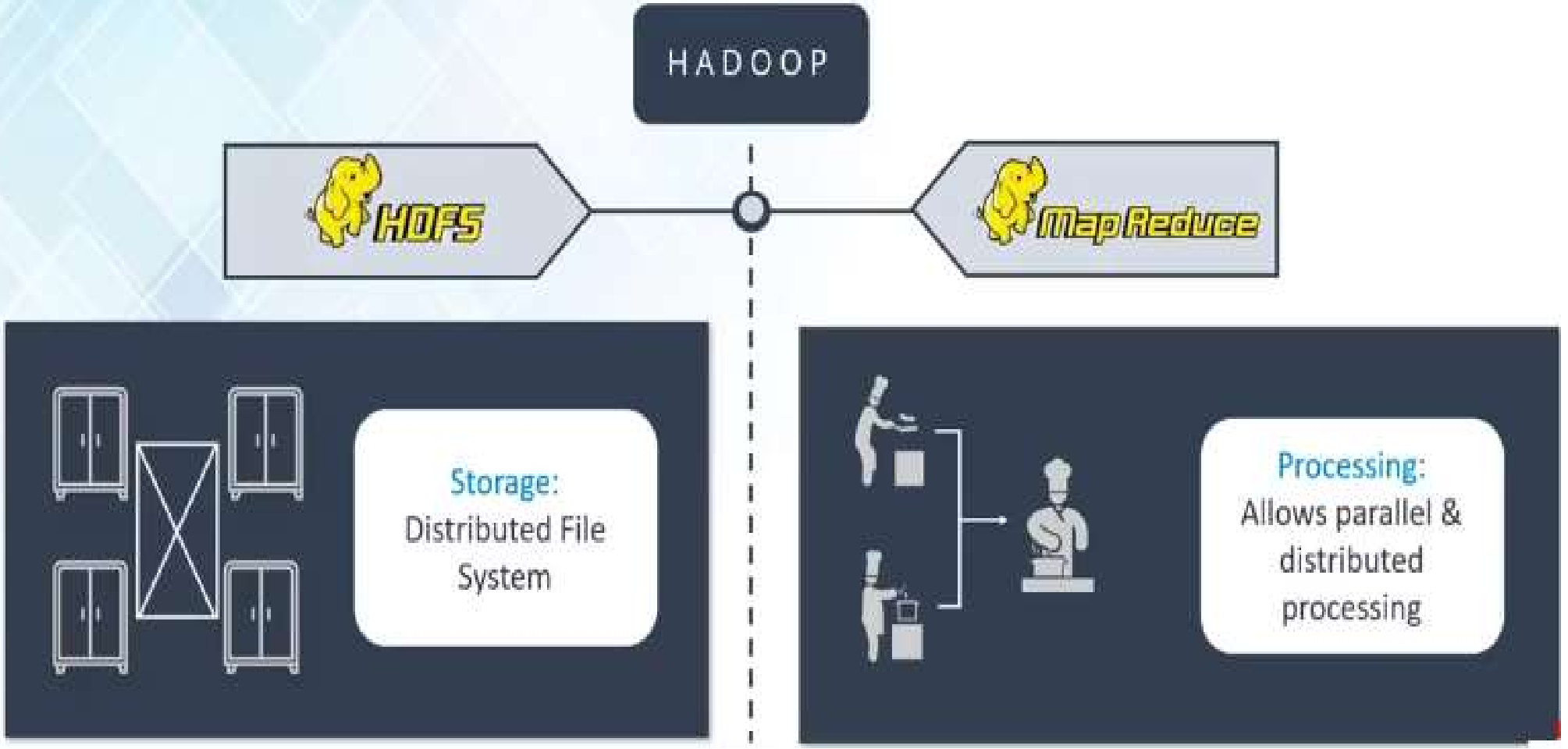


Solution Framework

WE NEED SOME FRAMEWORK THAT CAN PROVIDES SOLUTION

Apache Hadoop

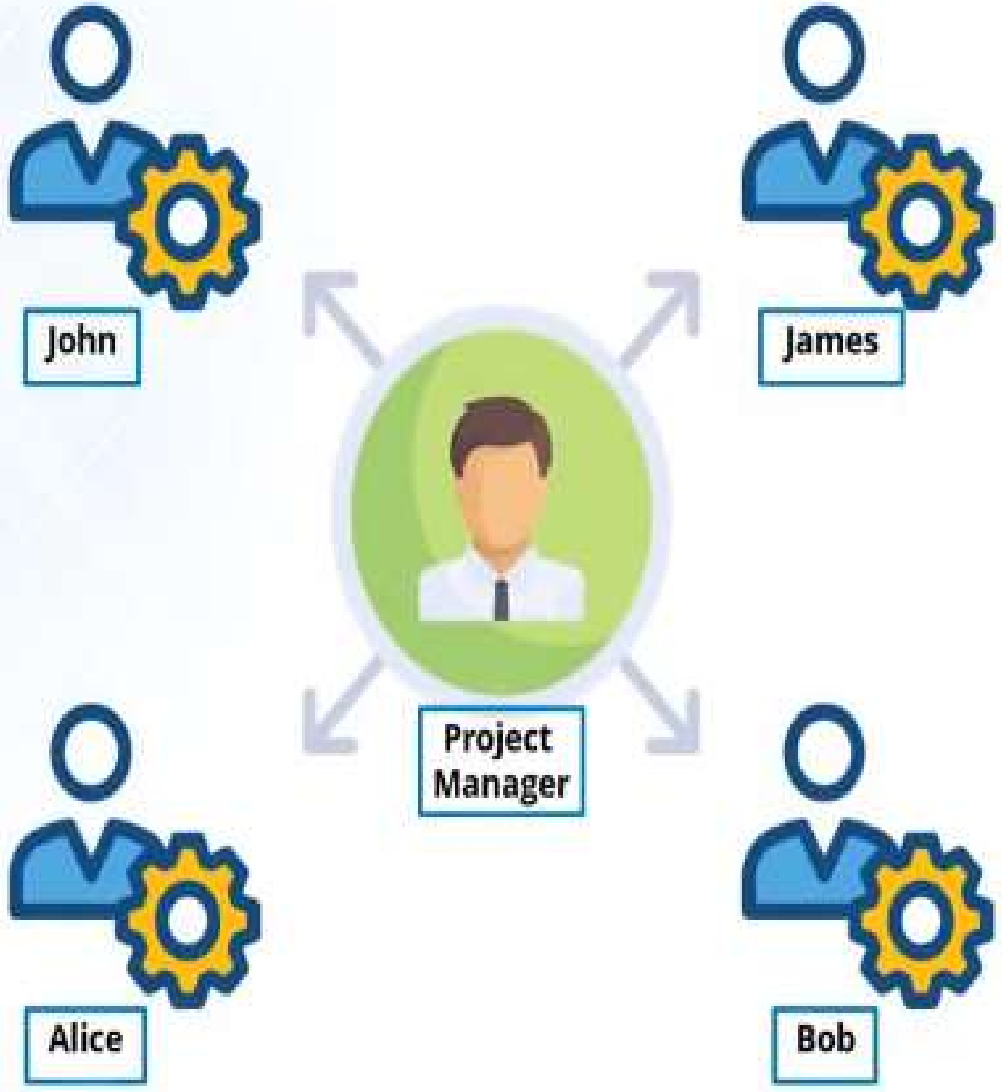
Hadoop is a framework that allows us to store and process large data sets in parallel and distributed fashion



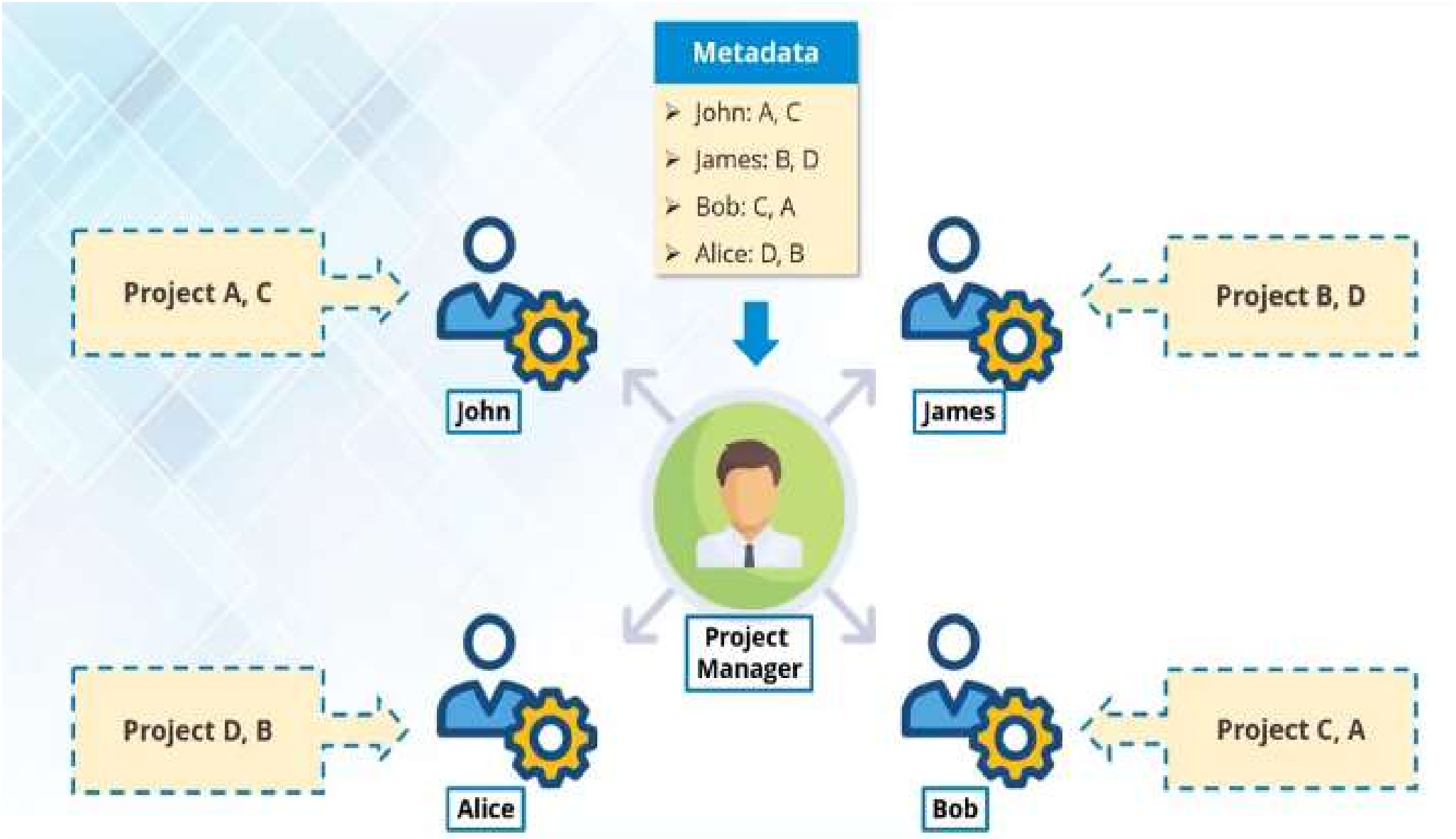
Master/Slave Approach

Scenario:

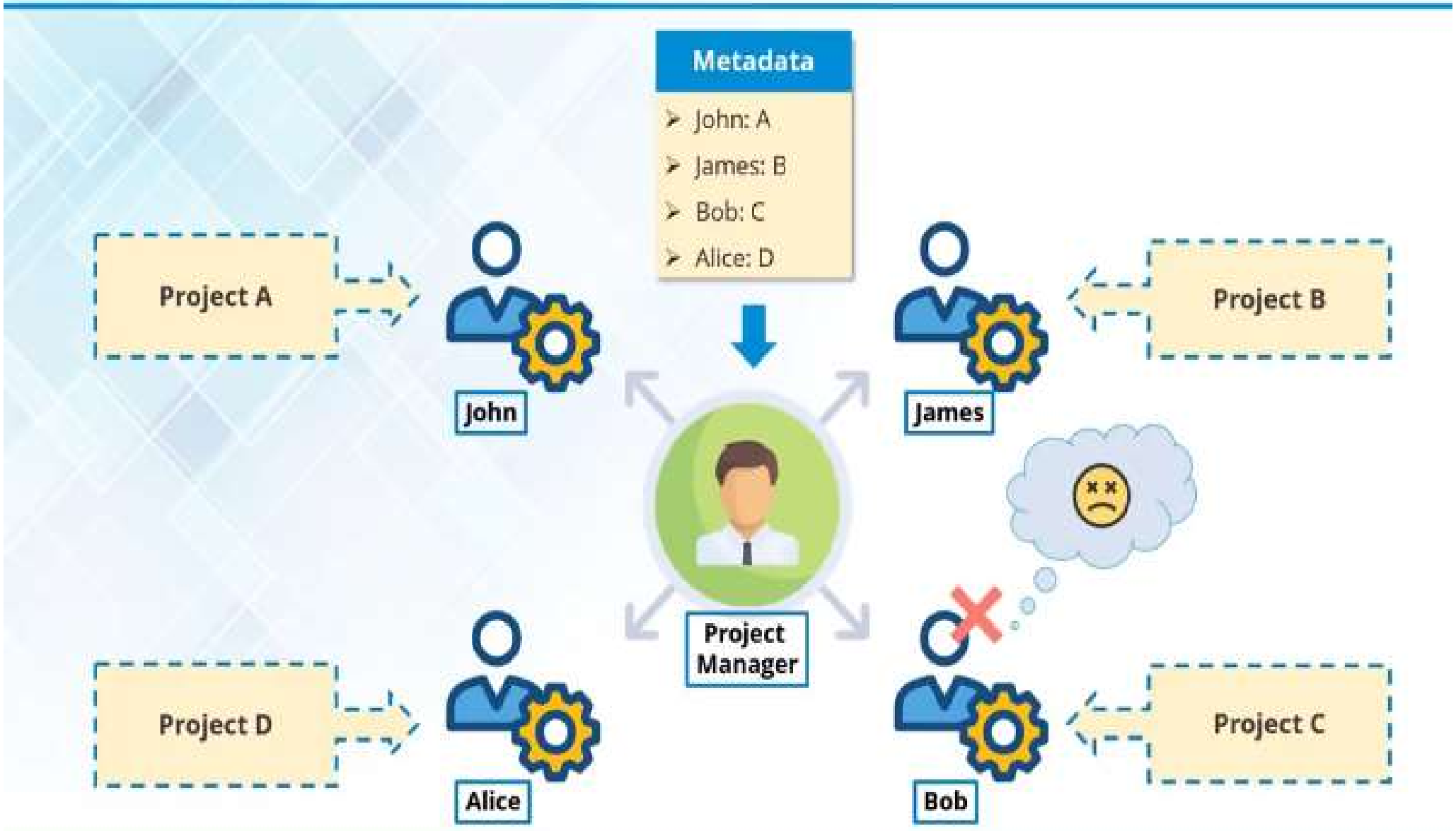
A project Manager managing a team of four employees. He assigns project to each of them and tracks the progress



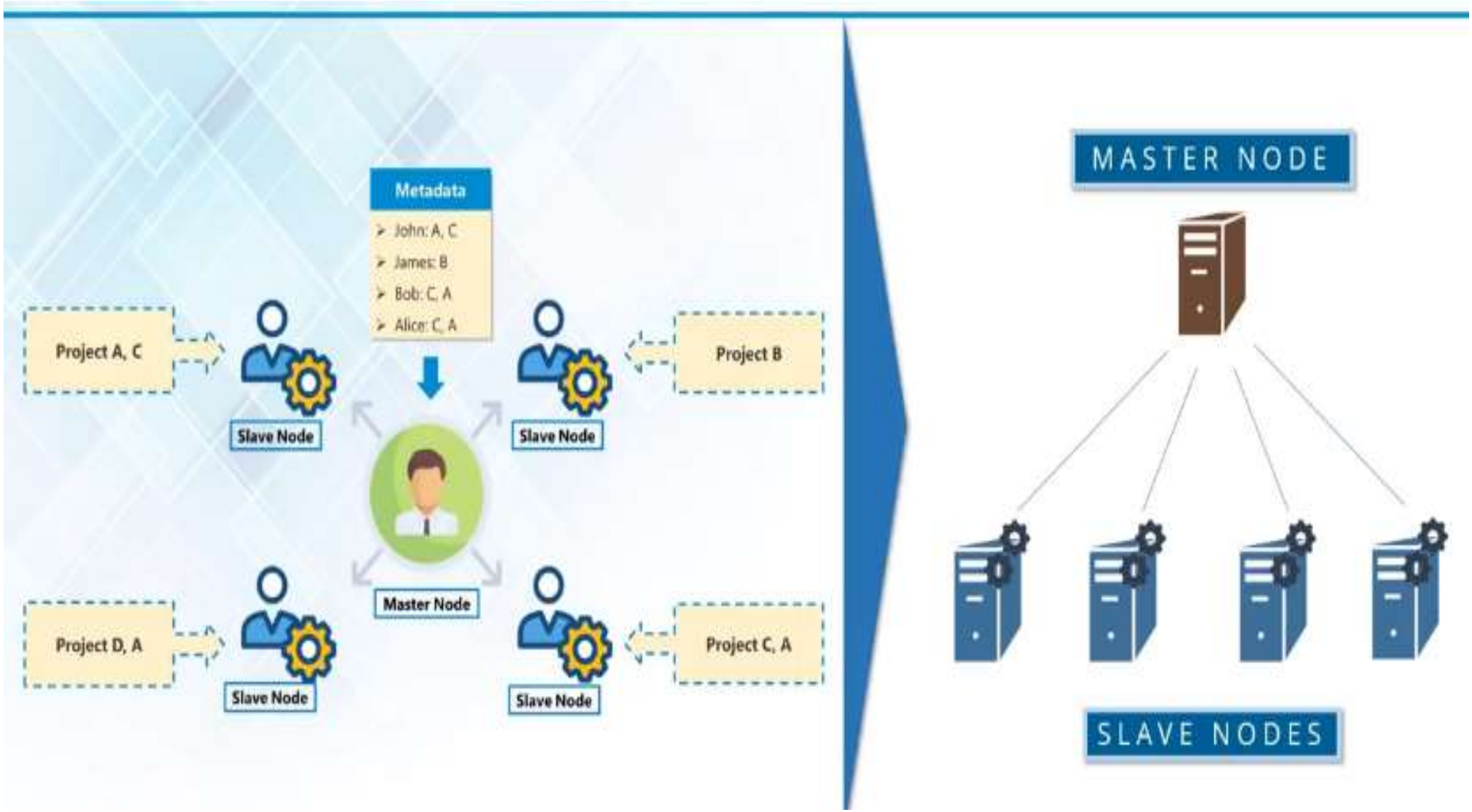
Maintain Metadata



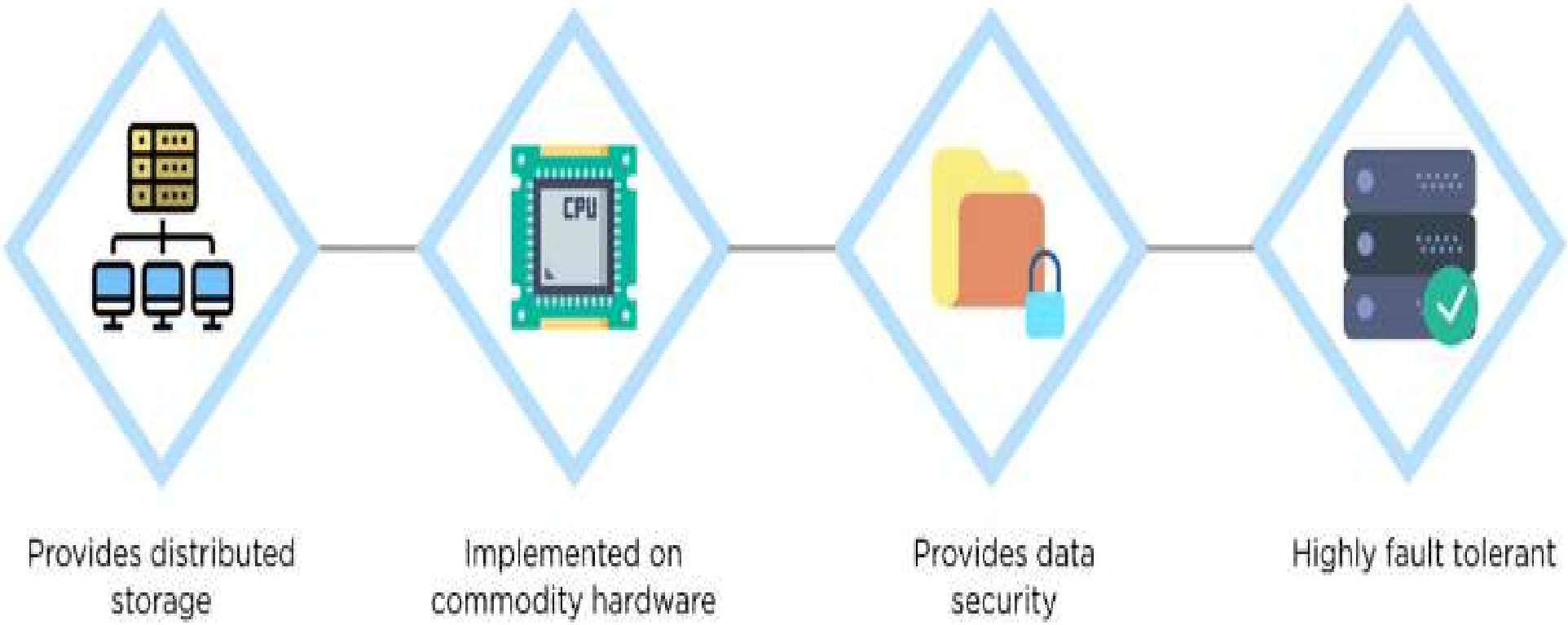
Backup Failover Solution



Evolution of Architecture

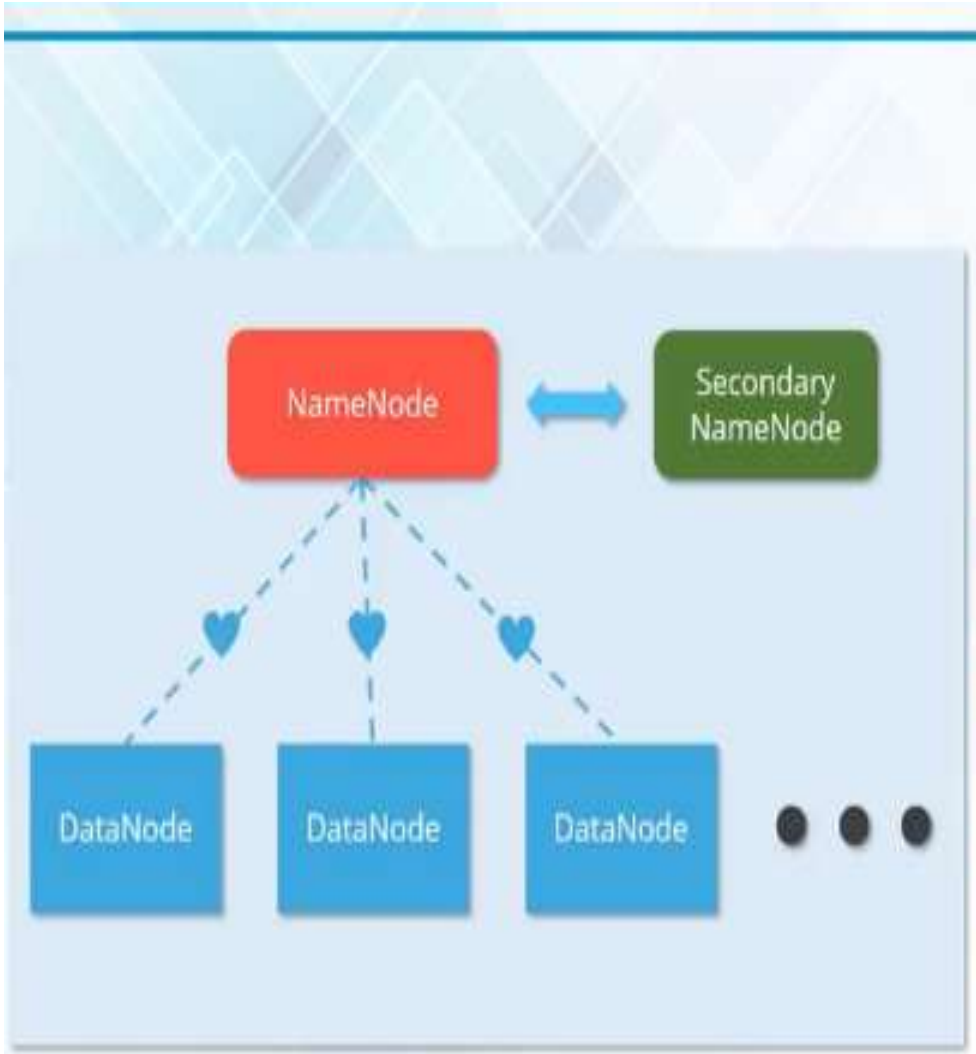


Evolution of HDFS



3 Core Components

Namenode & Datanode



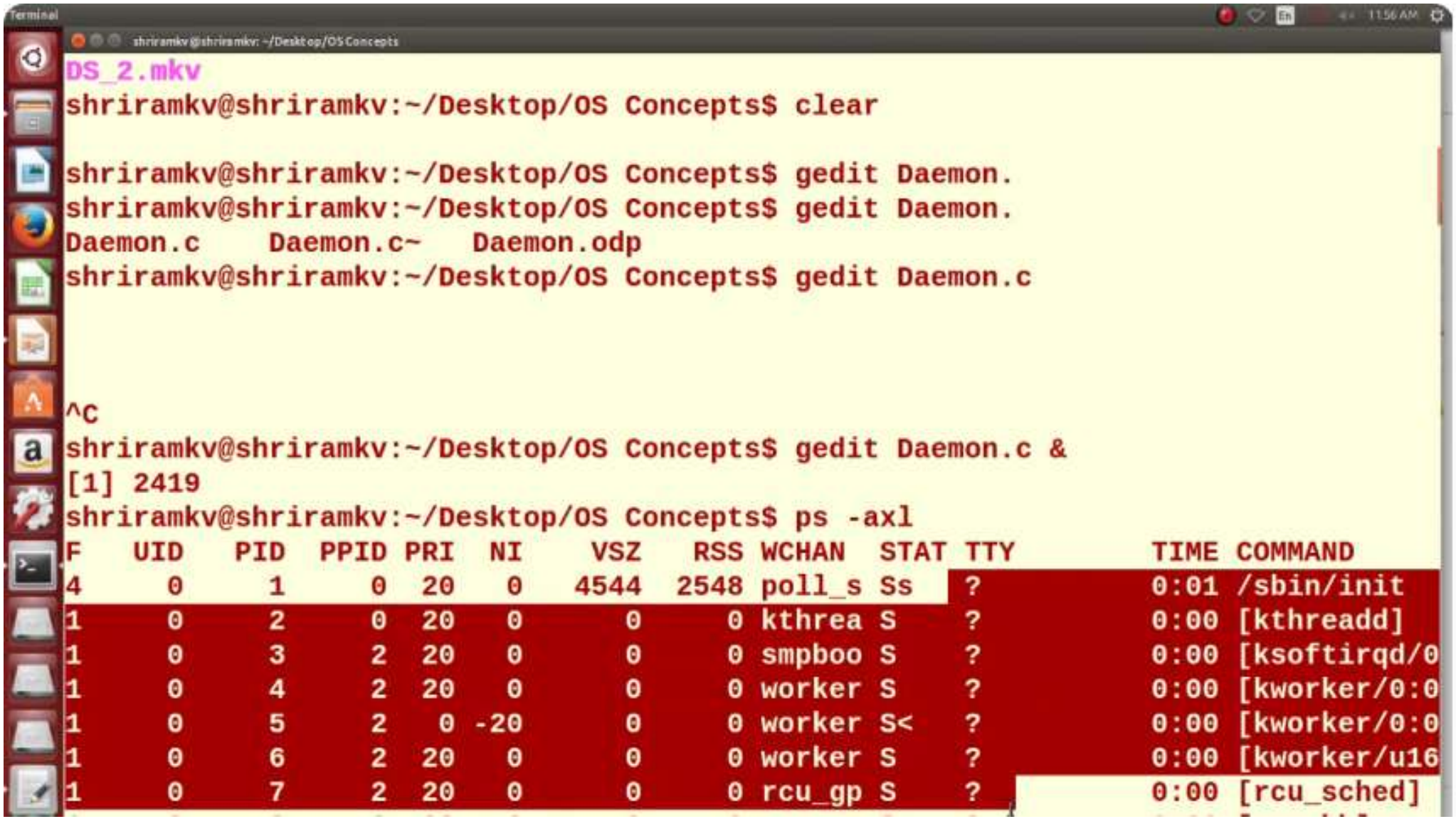
NameNode:

- Maintains and Manages DataNodes
- Records metadata i.e. information about data blocks e.g. location of blocks stored, the size of the files, permissions, hierarchy, etc.
- Receives heartbeat and block report from all the DataNodes

DataNode:

- Slave daemons
- Stores actual data
- Serves read and write requests from the clients

Daemon Services

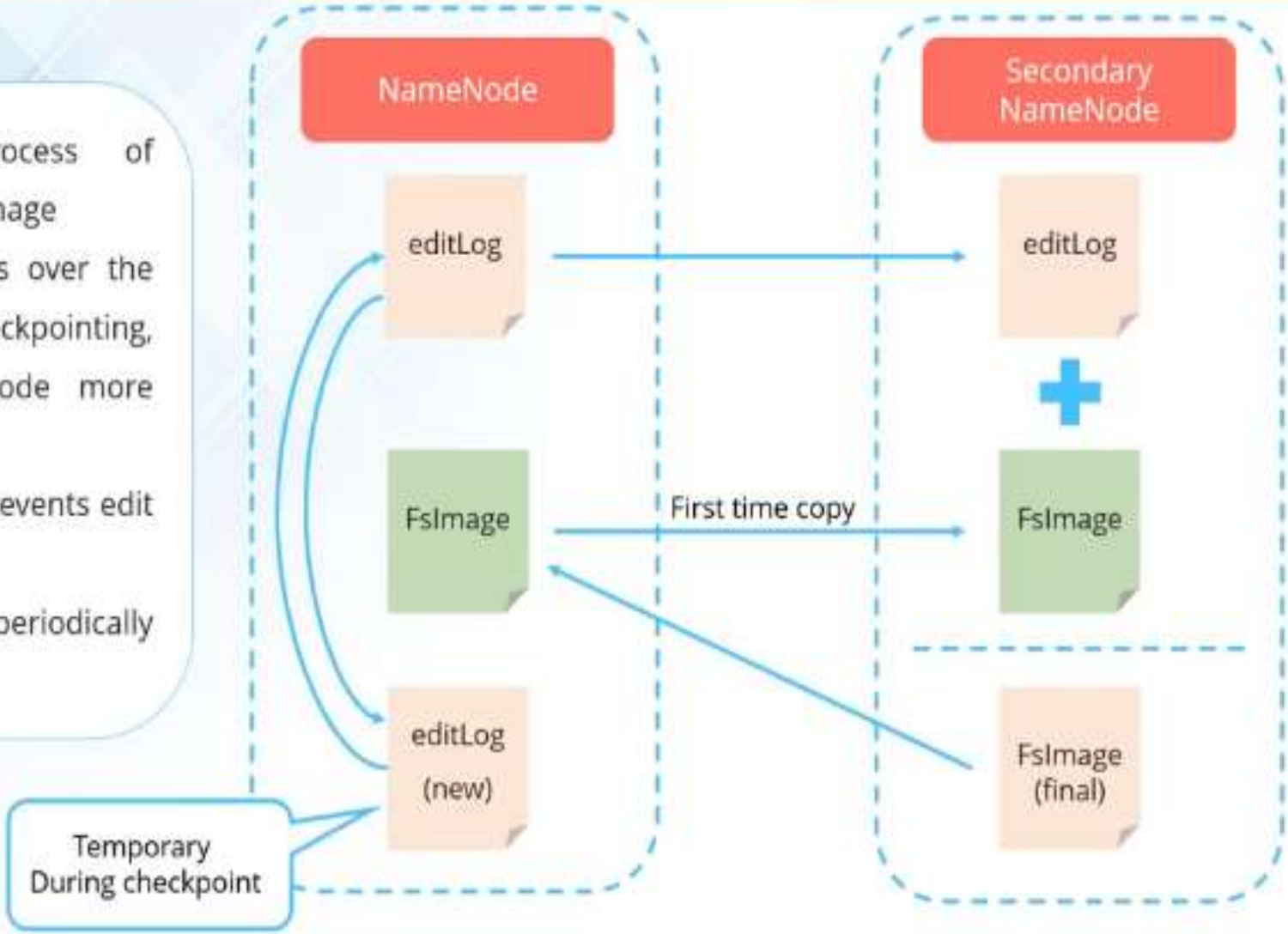


About Daemon Process

- It is a program.
- Runs in Unix/Linux without interruption or user intiation to happen.
- Executed in the background.
- Recollect, orphans.
- No terminal usage
- Command to see the daemons
- Ps -axl (results will have ? Under TTY, it tells Daemon's exisitence)

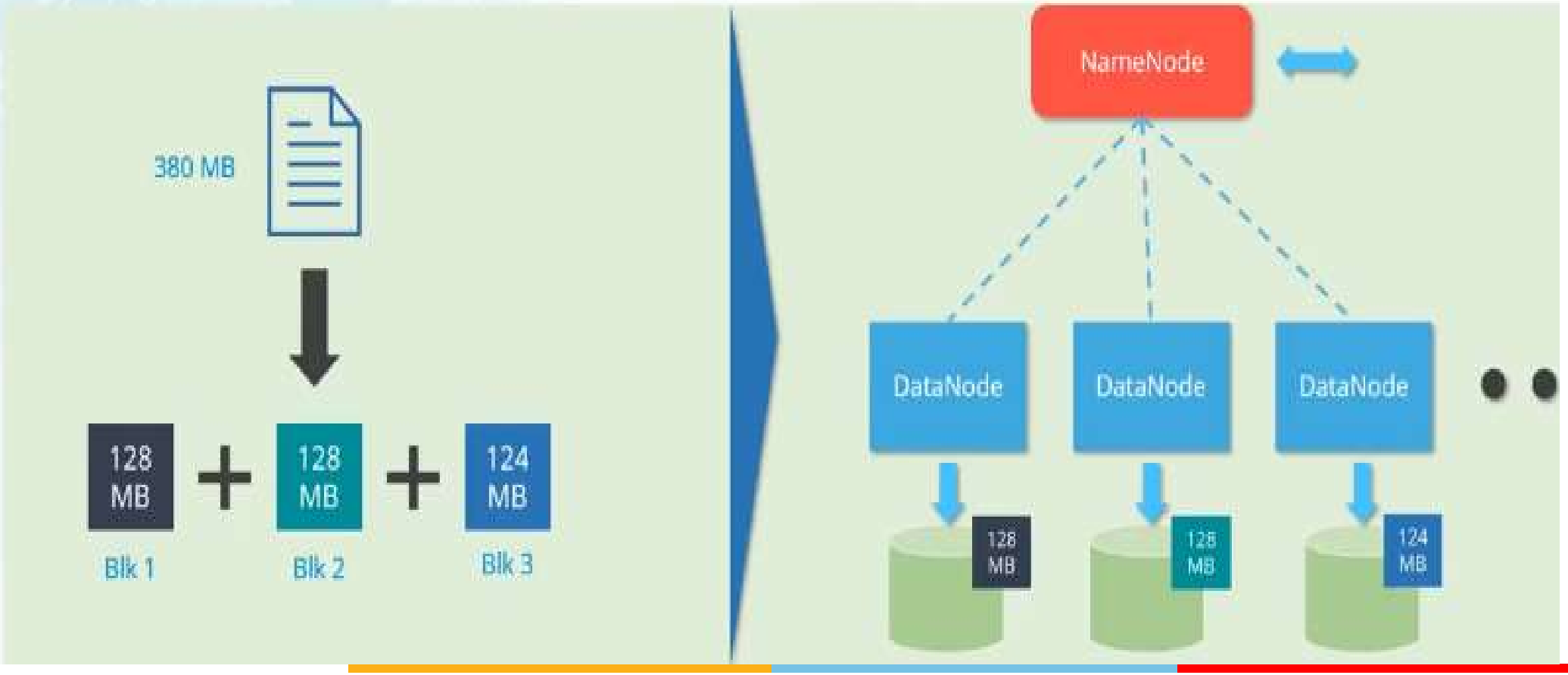
Secondary NameNode & Checkpoint

- Checkpointing is a process of combining edit logs with FsImage
- Secondary NameNode takes over the responsibility of checkpointing, therefore, making NameNode more available
- Allows faster Failover as it prevents edit logs from getting too huge
- Checkpointing happens periodically (default: 1 hour)

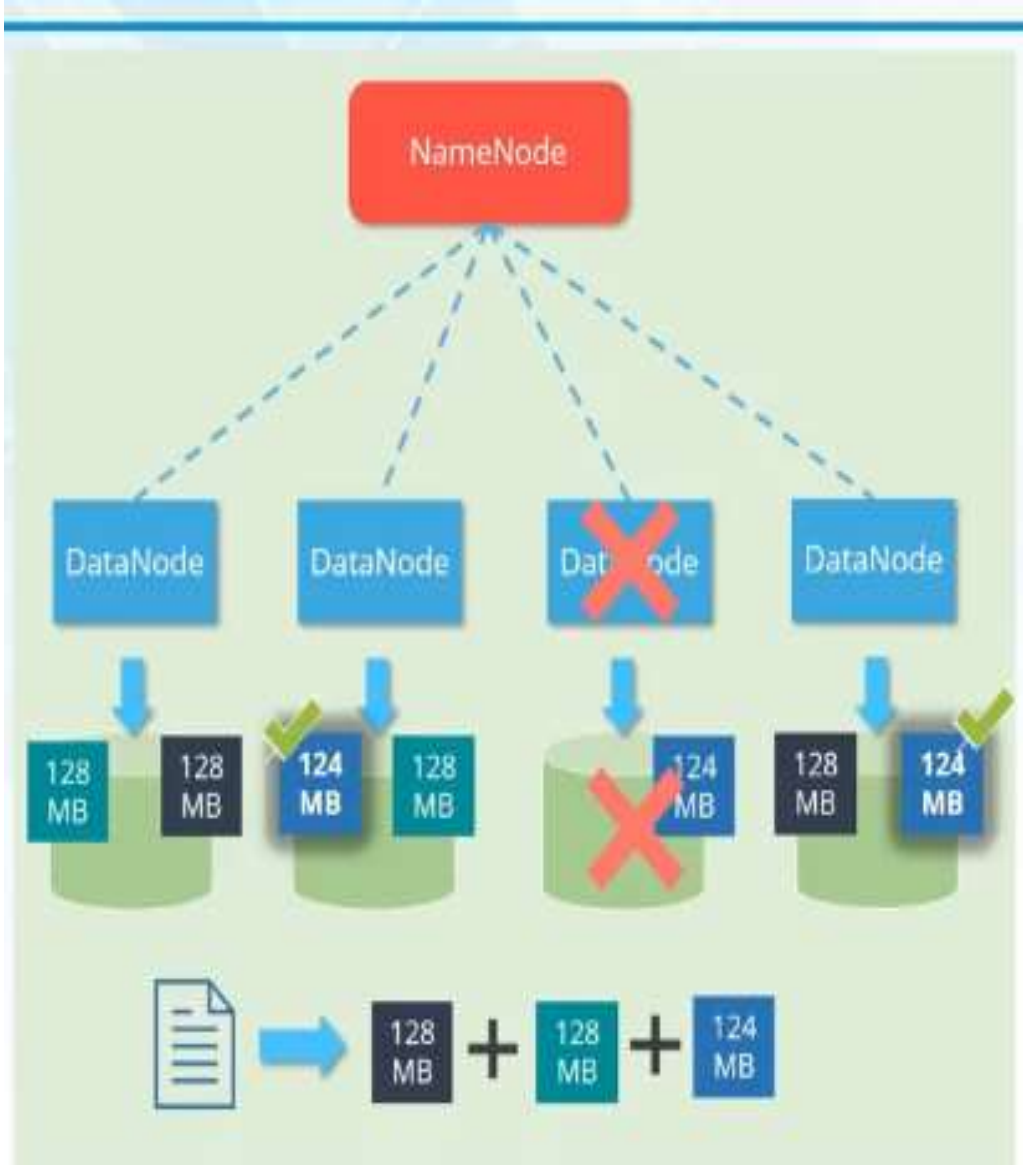


HDFS DataNode Blocks

- Each file is stored on HDFS as blocks
- The default size of each block is 128 MB in Apache Hadoop 2.x (64 MB in Apache Hadoop 1.x)



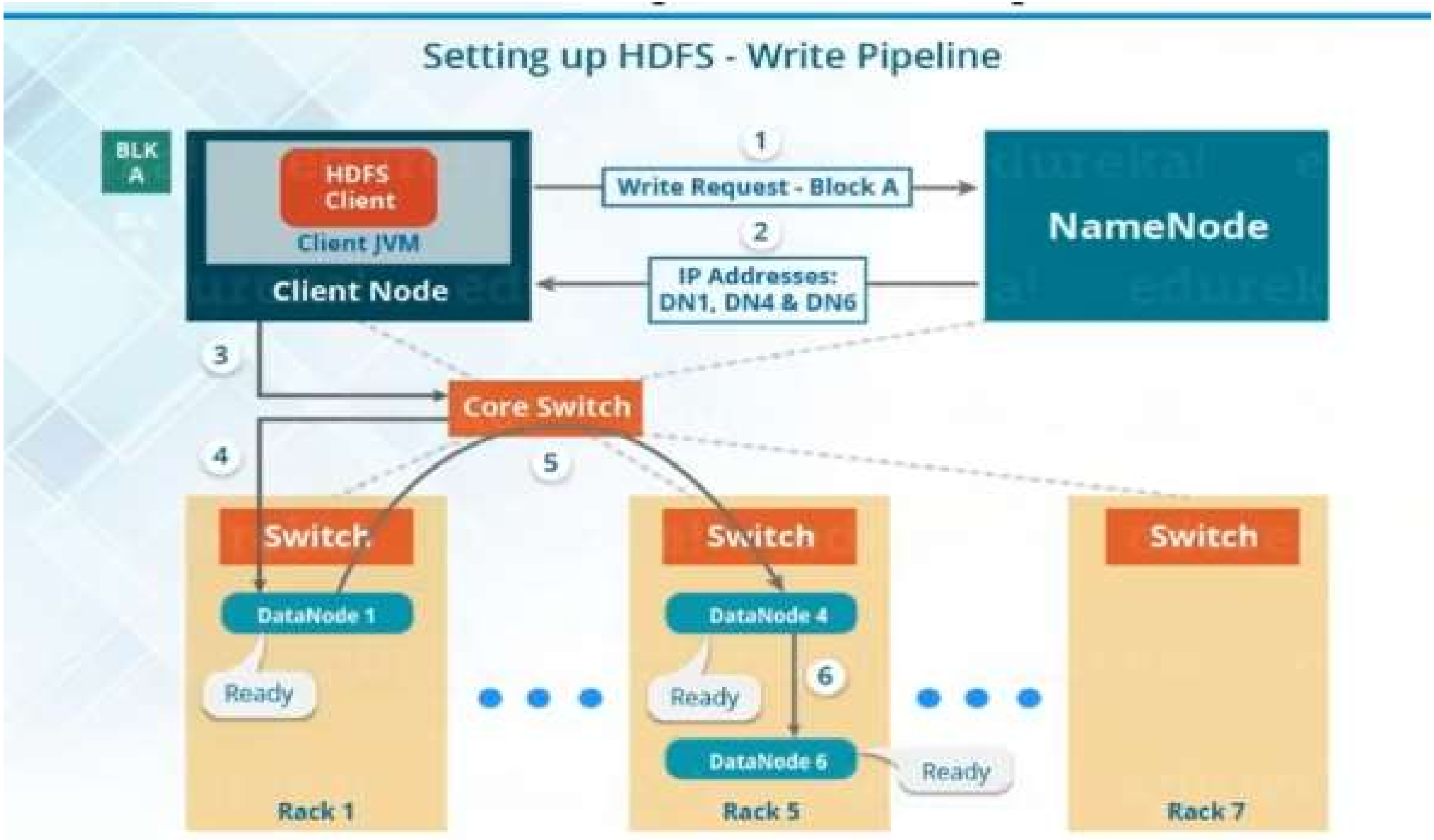
Replication



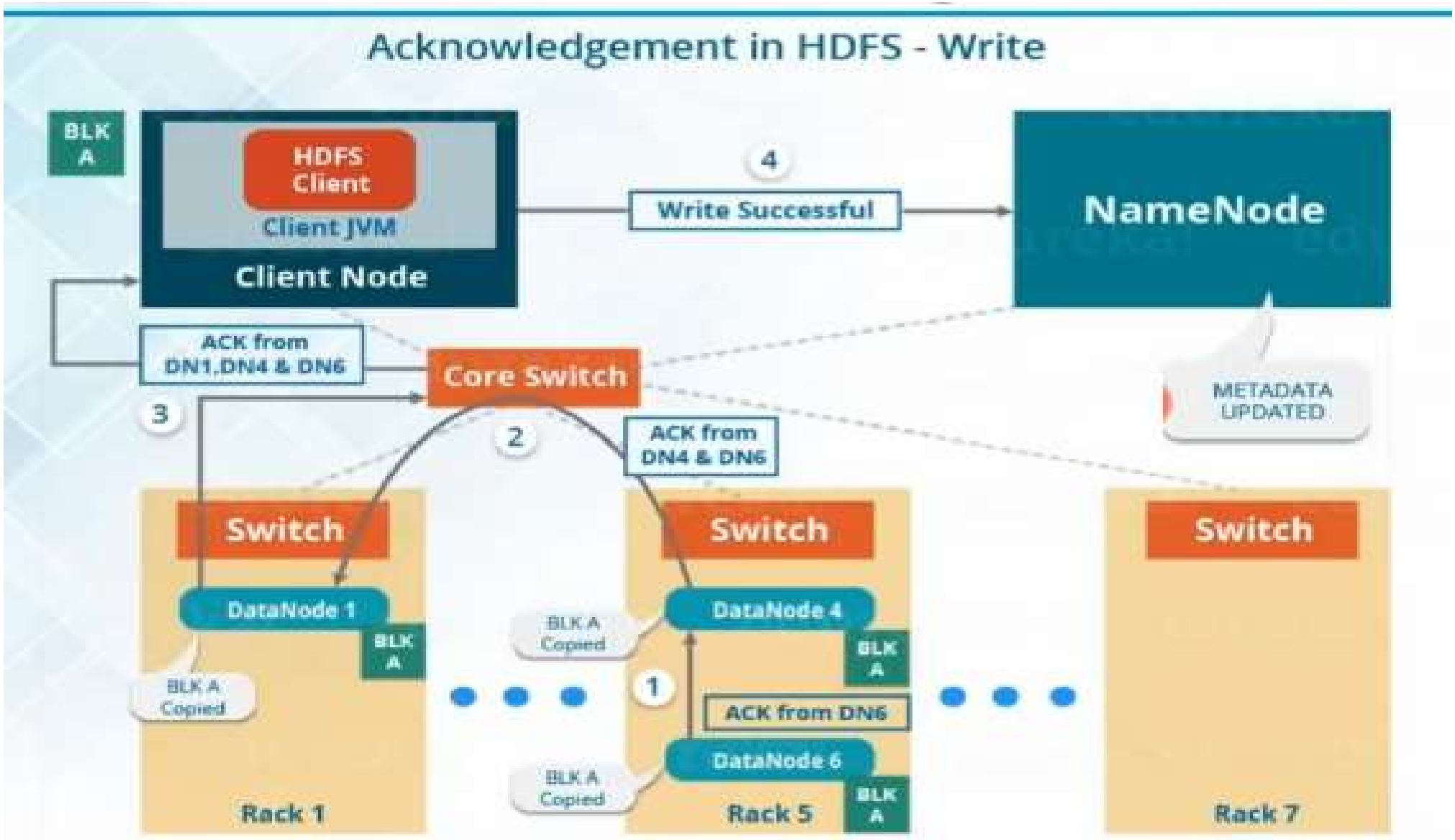
Solution:

Each data blocks are replicated (thrice by default) and are distributed across different DataNodes

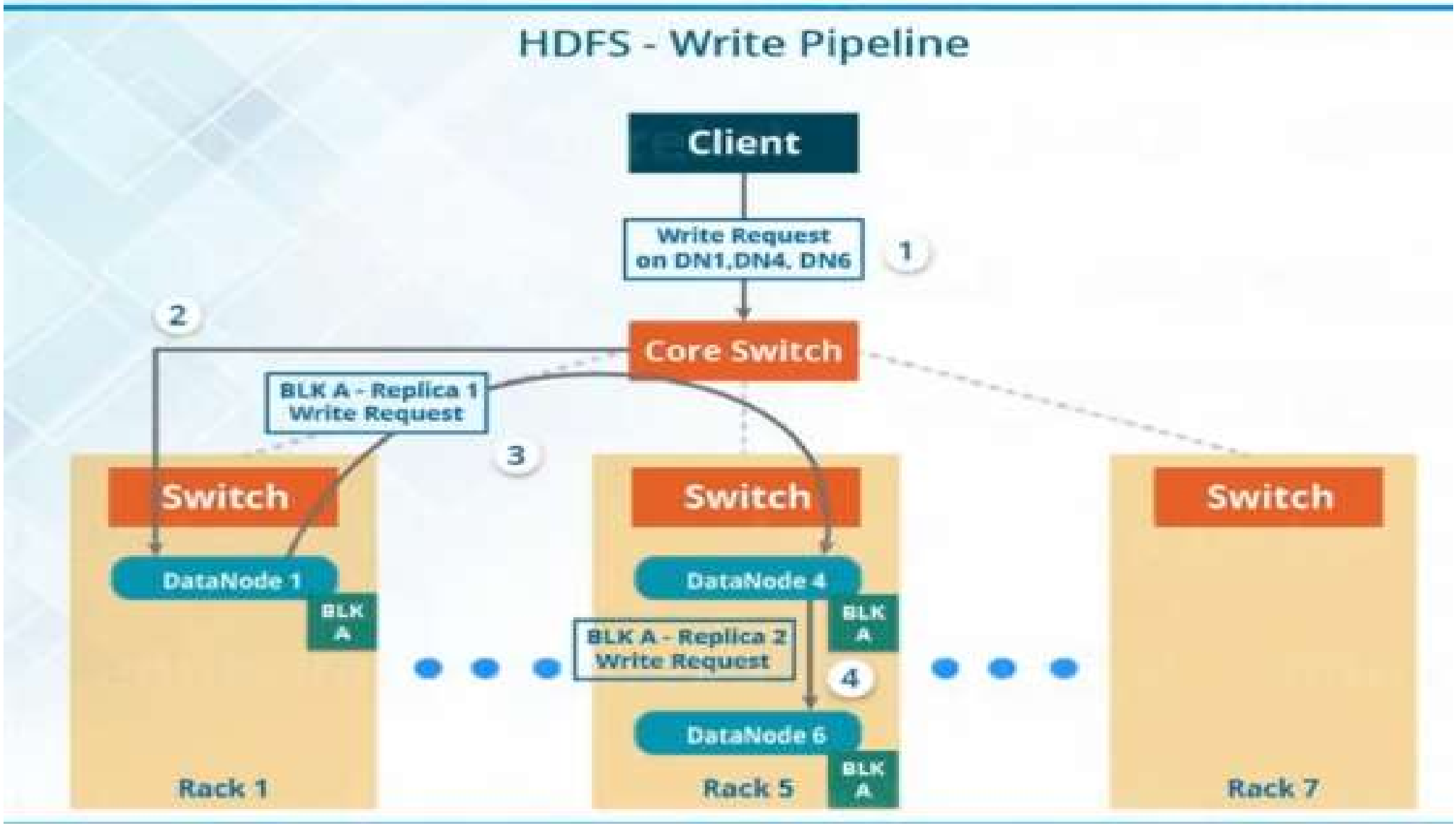
HDFS Write


Setting up HDFS - Write Pipeline
BLK A
HDFS Client
Client JVM
Client Node
1
Write Request - Block A
2
IP Addresses: DN1, DN4 & DN6
NameNode
3
Core Switch
4
5
Switch
DataNode 1
Ready
Rack 1
Switch
DataNode 4
Ready
6
DataNode 6
Ready
Rack 5
Switch
Rack 7

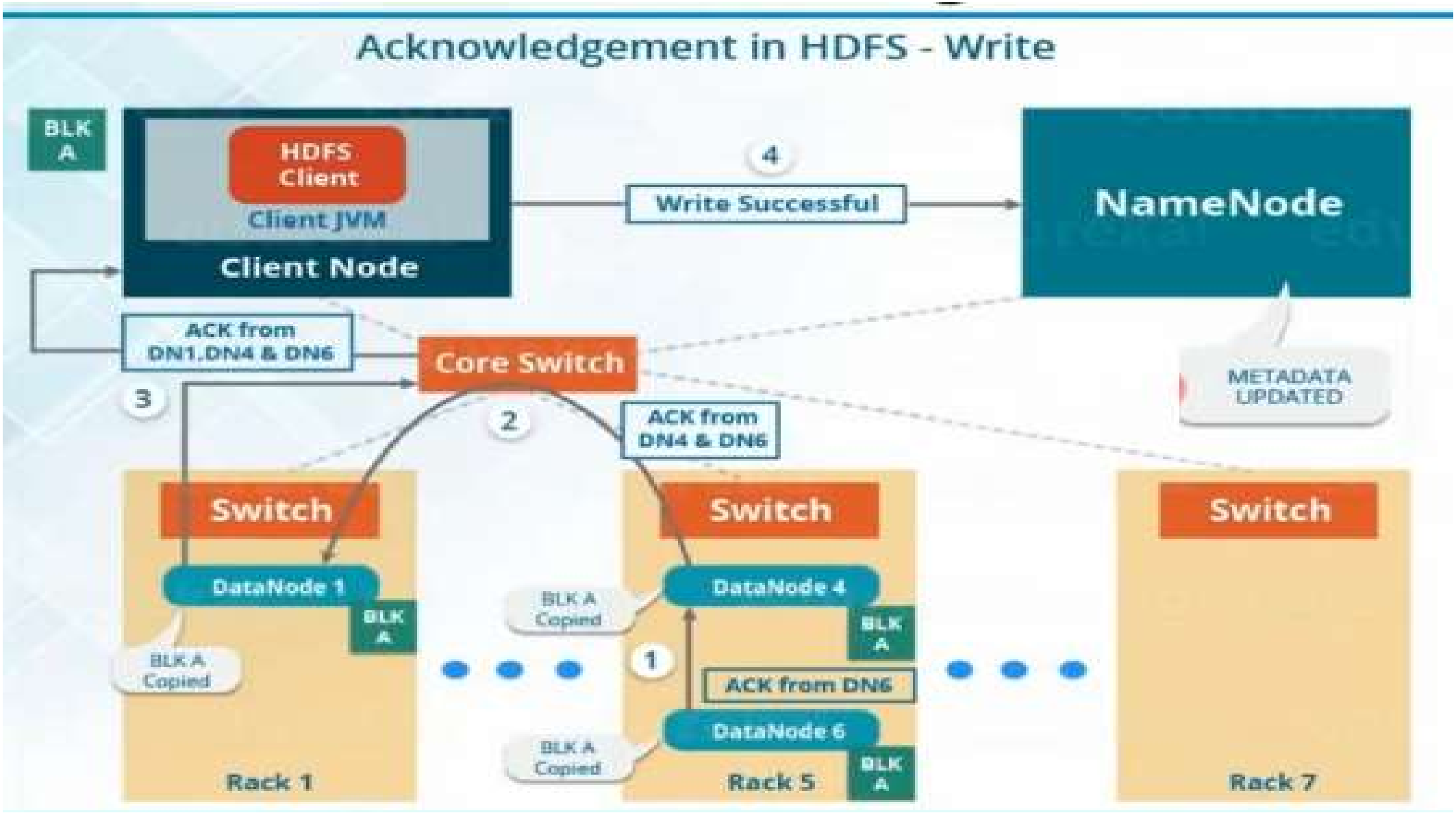
HDFS Write - Ack



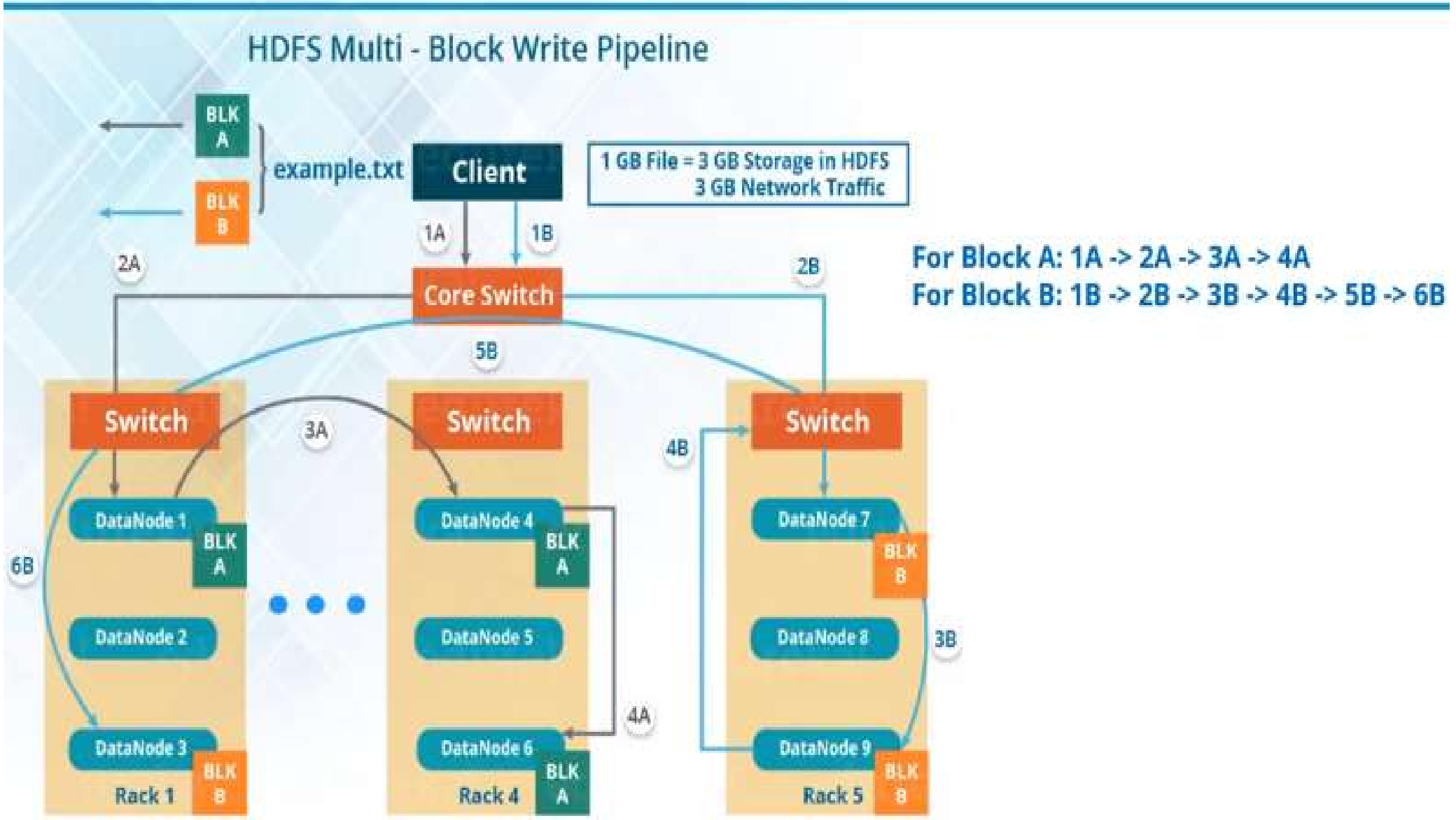
Write Request



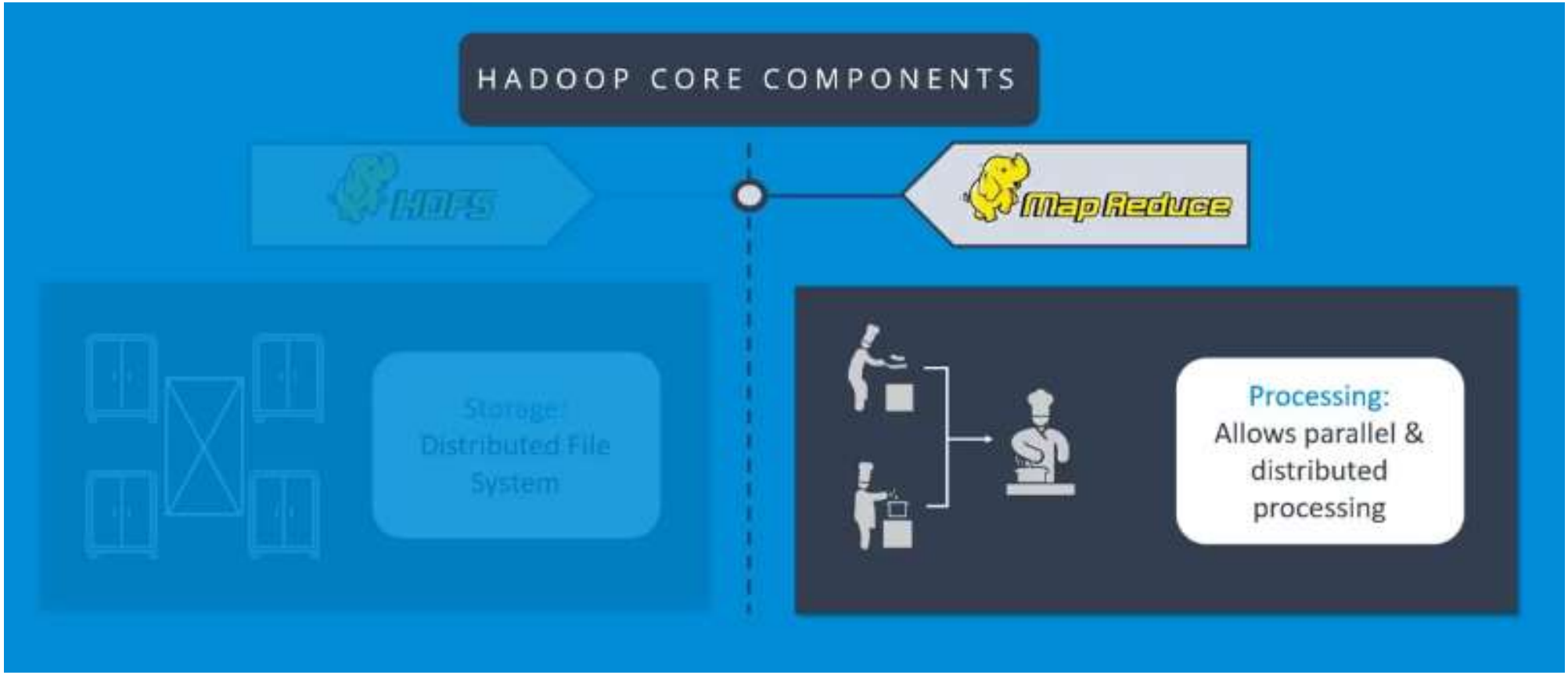
Ack


Acknowledgement in HDFS - Write
BLK A
HDFS Client
Client JVM
Client Node
4
Write Successful
NameNode
ACK from DN1,DN4 & DN6
Core Switch
3
2
ACK from DN4 & DN6
METADATA UPDATED
Switch
Switch
Switch
DataNode 1
BLK A
BLK A Copied
BLK A Copied
DataNode 4
BLK A
1
ACK from DN6
DataNode 6
BLK A Copied
BLK A
Rack 1
Rack 5
Rack 7

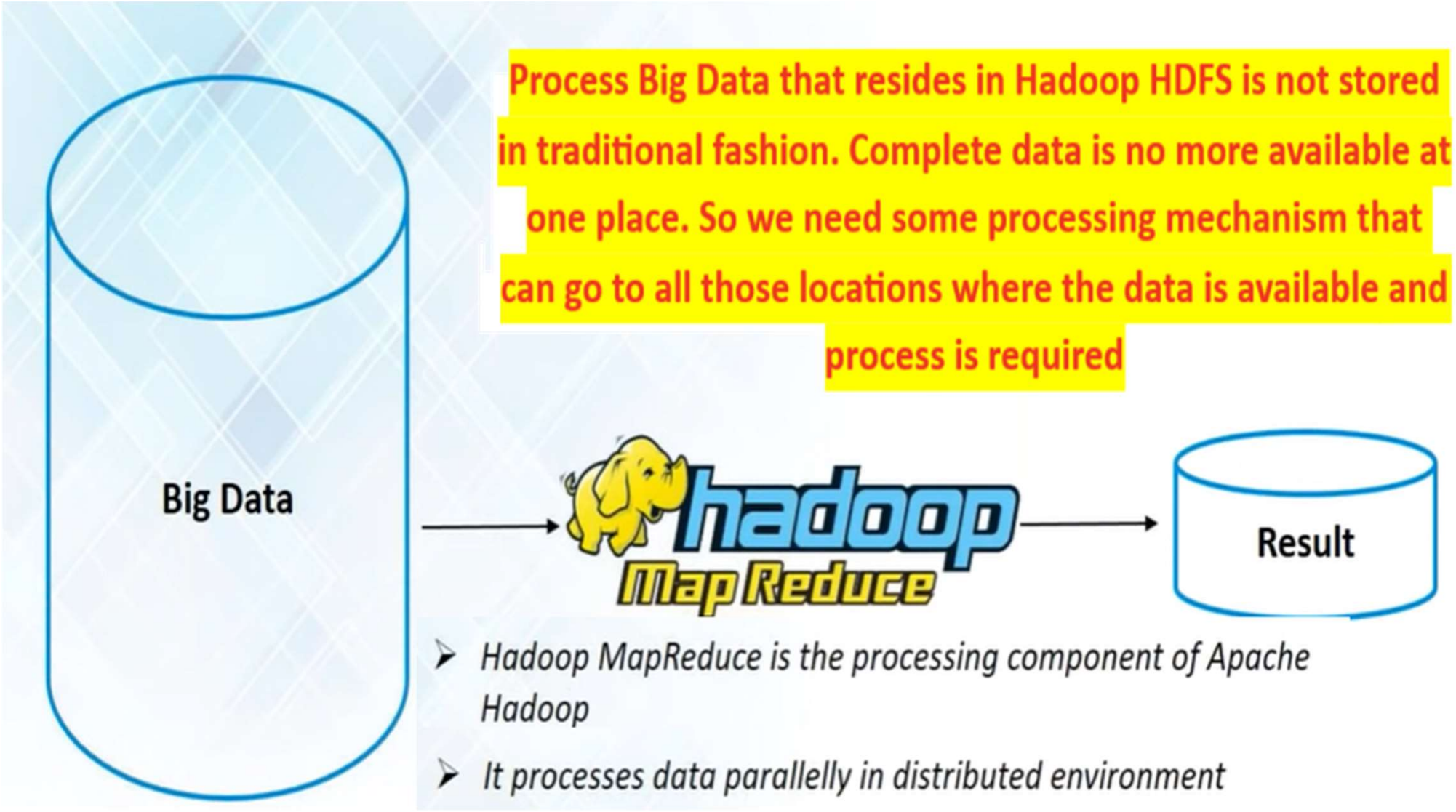
Multi Data Block write



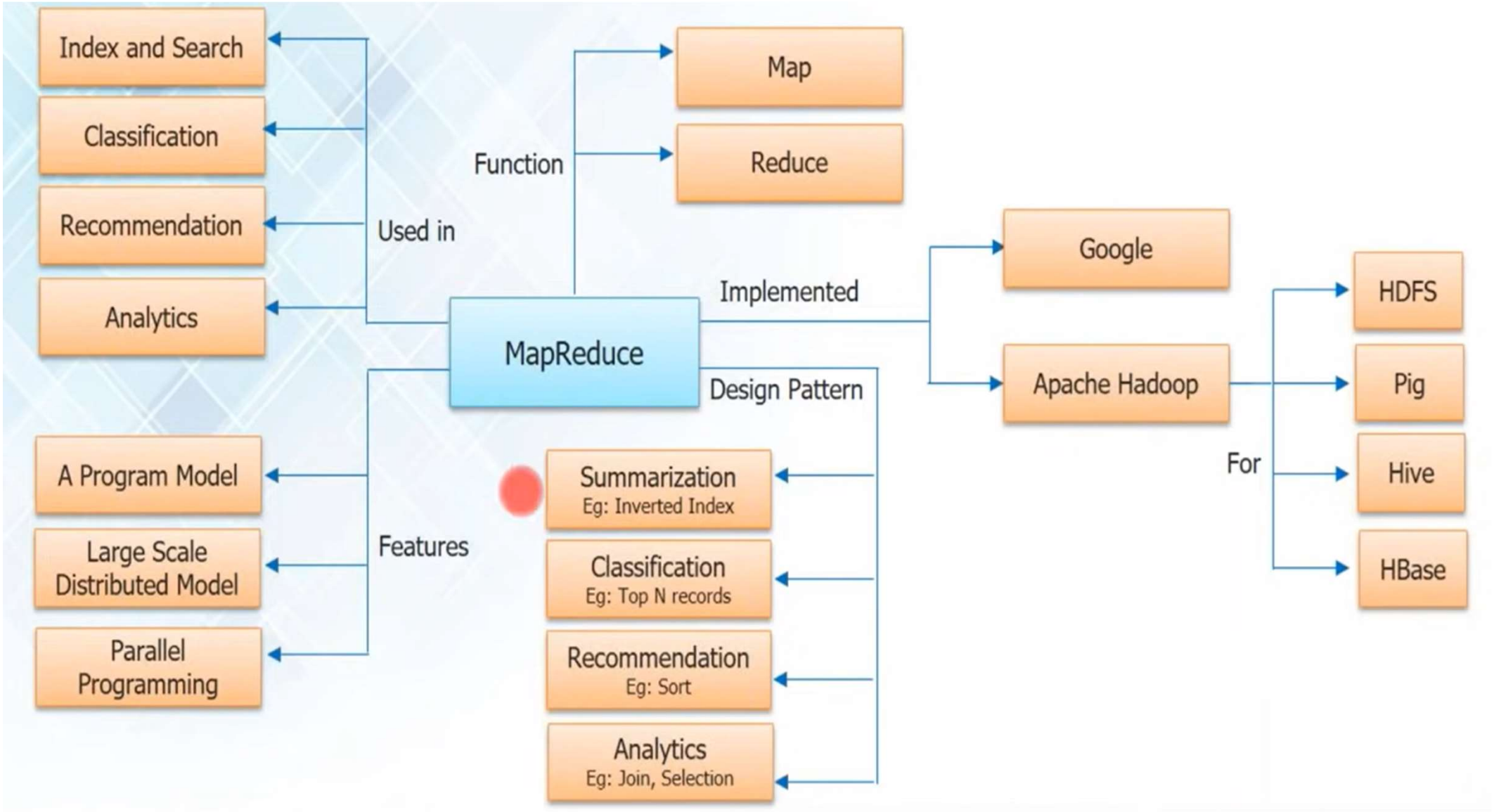
MapReduce Layer



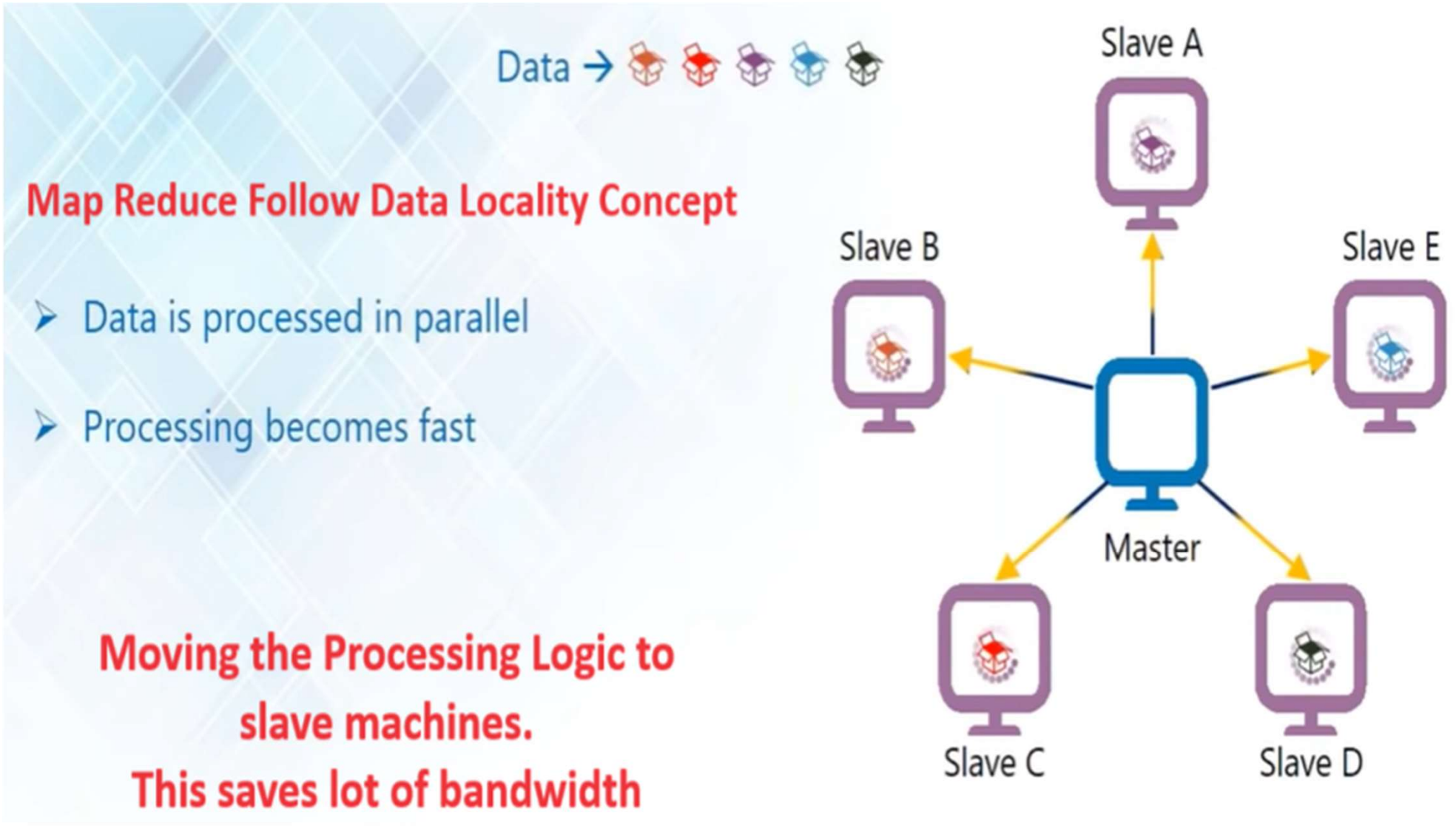
Why Map-Reduce?



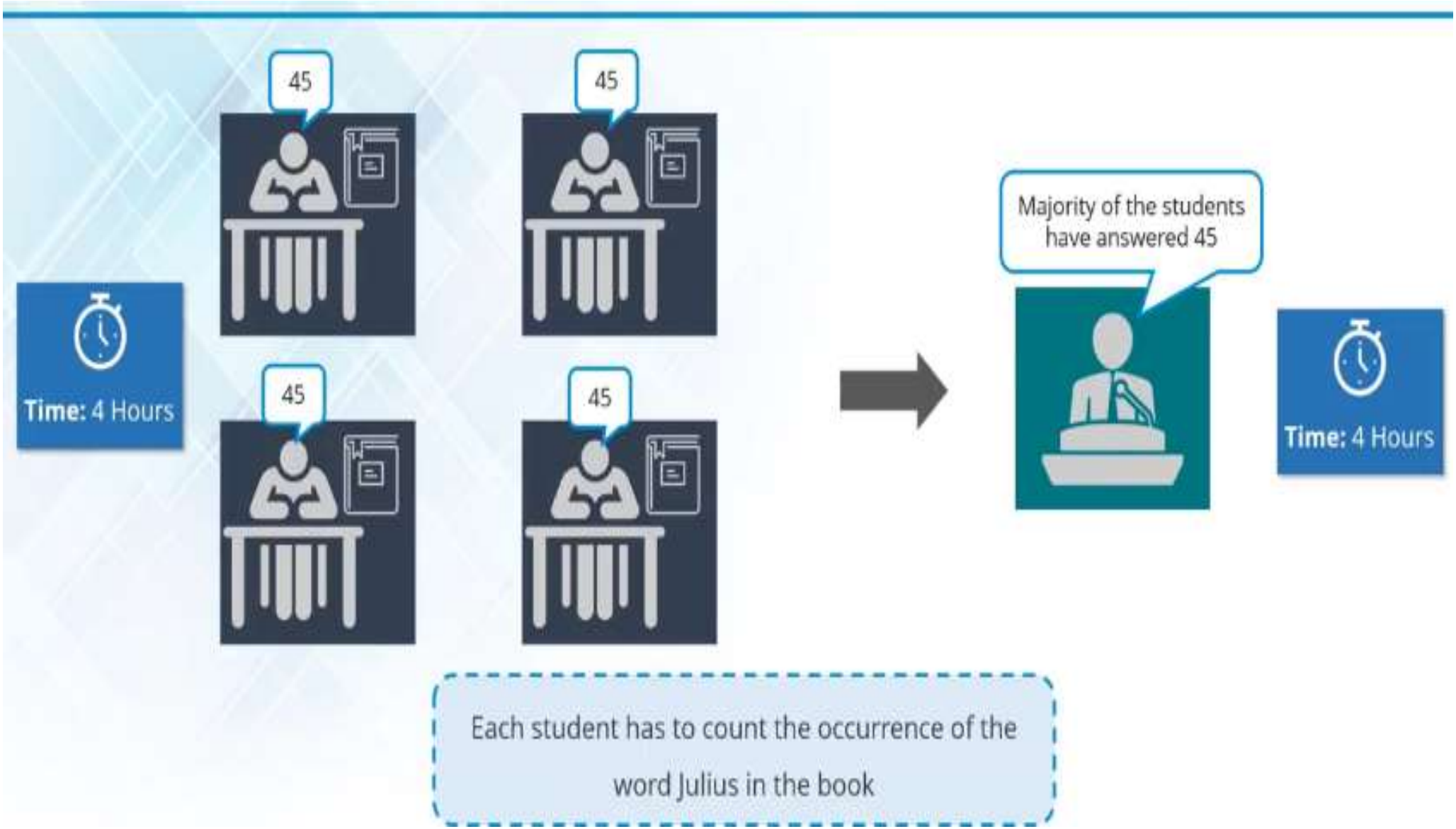
Bird Eye View



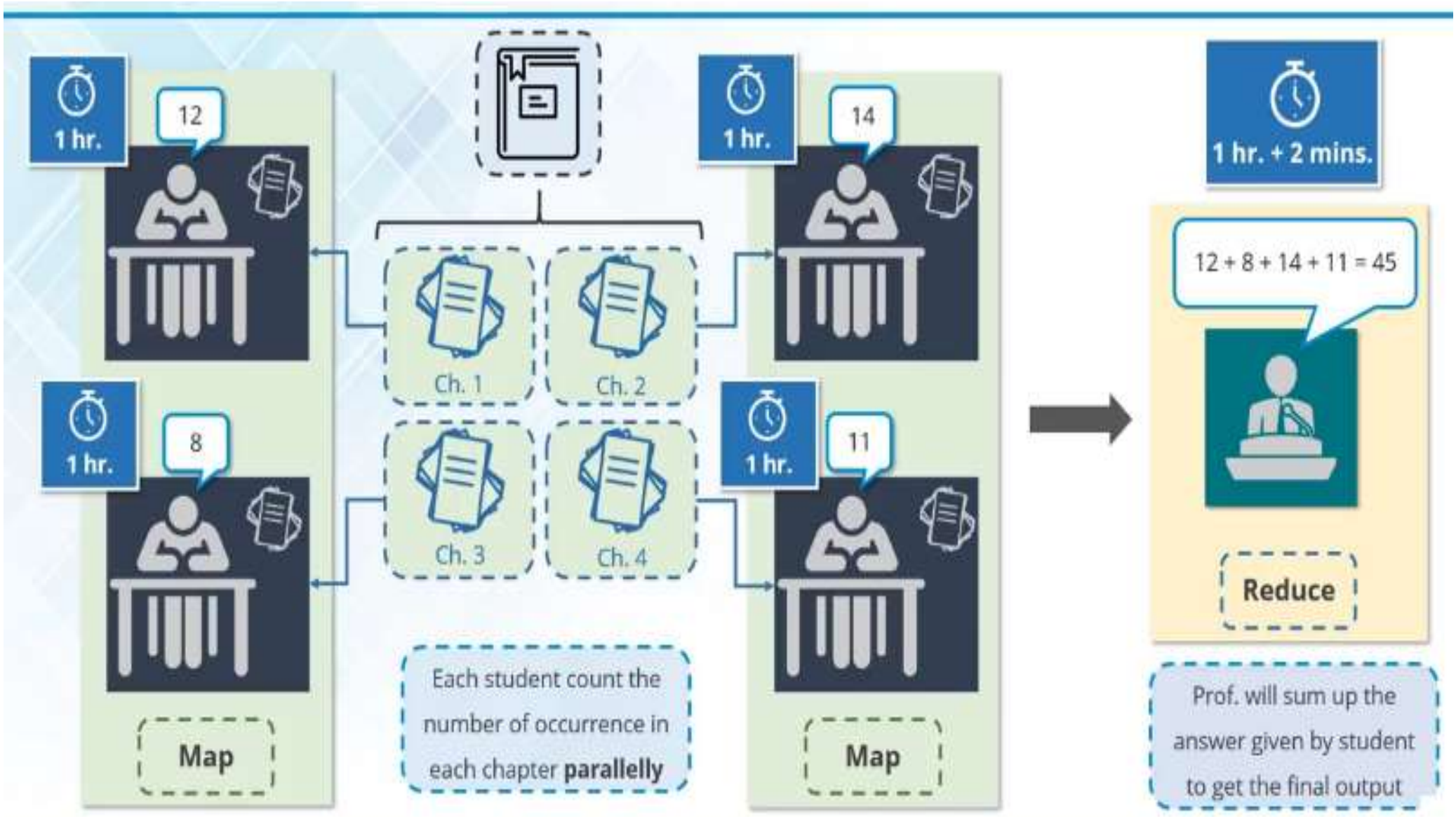
Parallel Processing scenario



Word Count in a Book – Normal mode

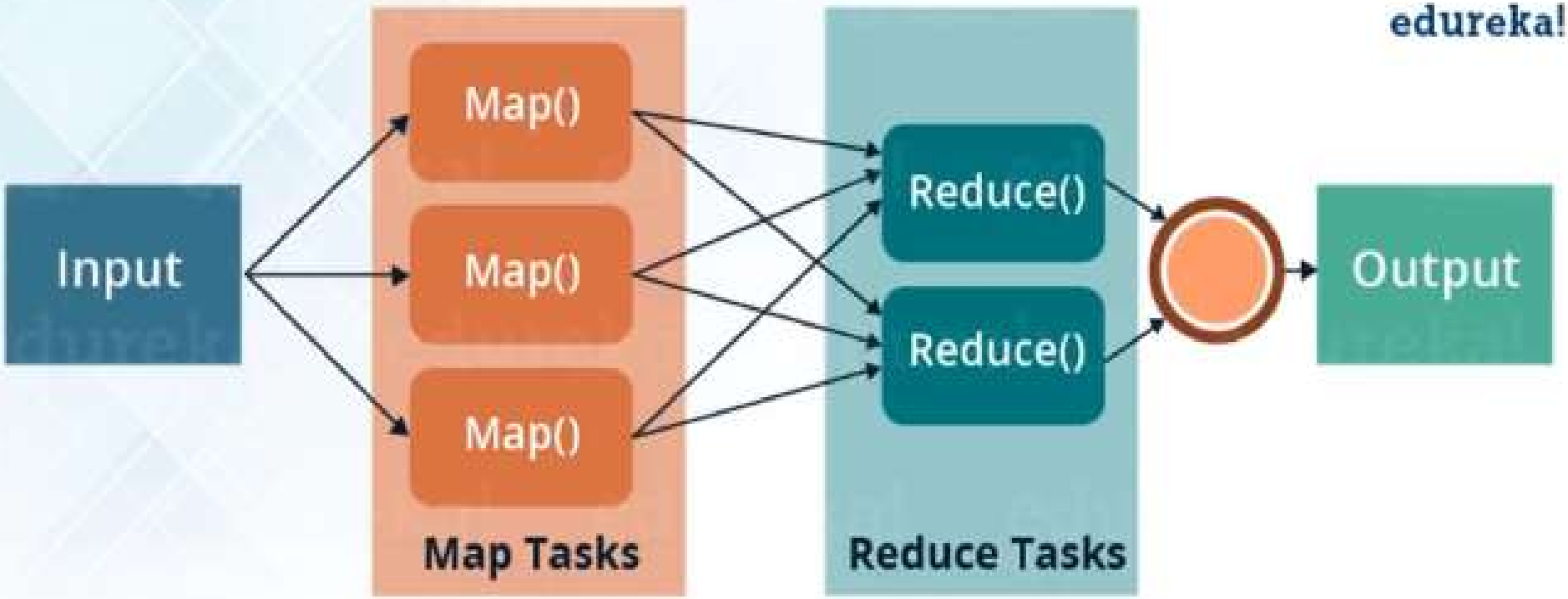


MapReduce Operation - Example

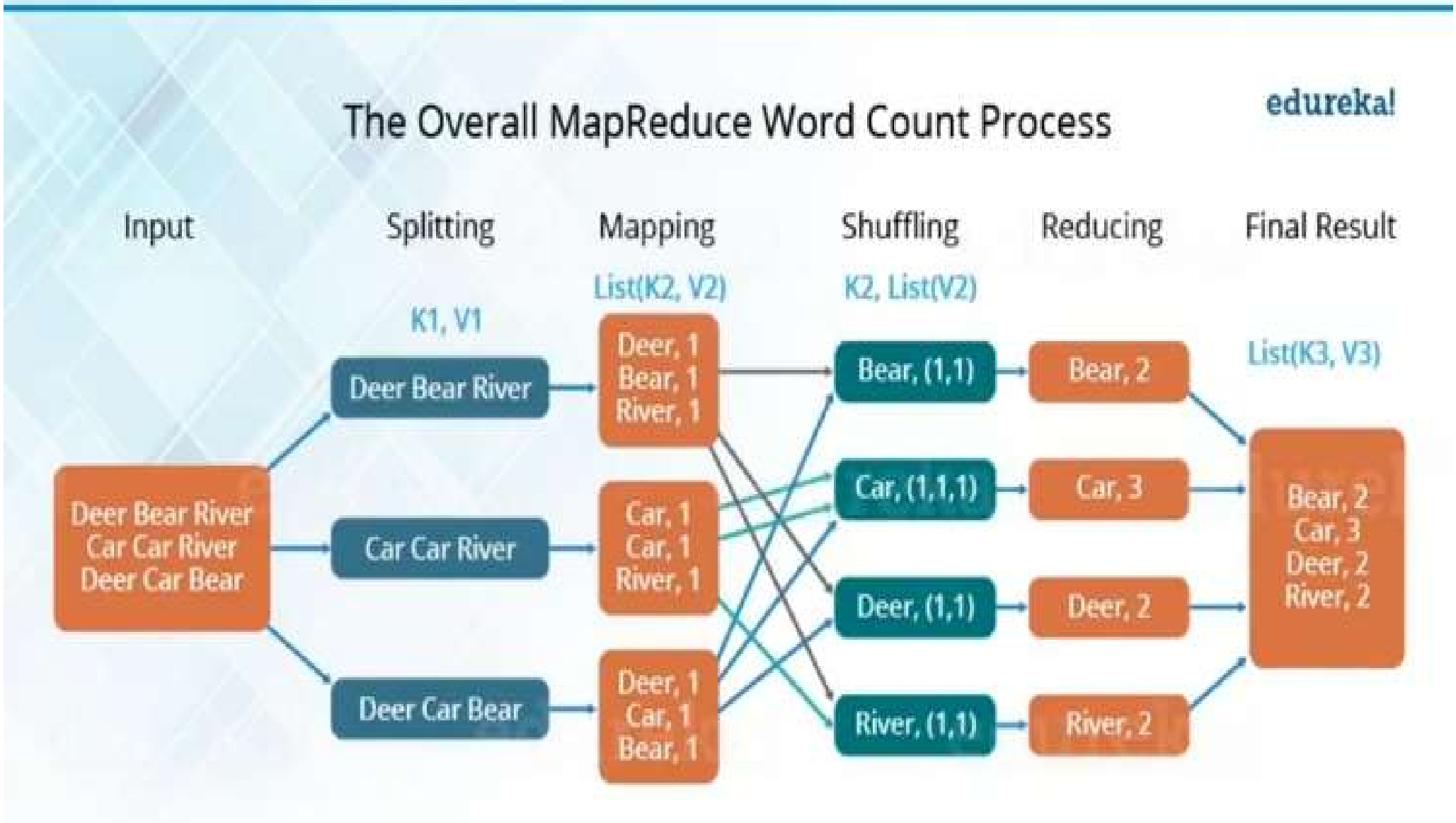


Exact Flow - MapReduce

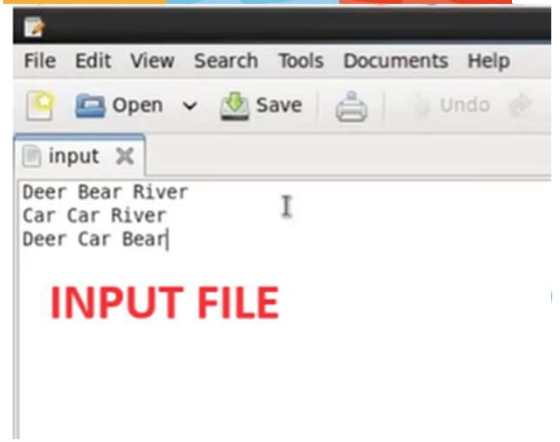
MapReduce is a programming framework that allows us to perform distributed and parallel processing on large data sets in a distributed environment

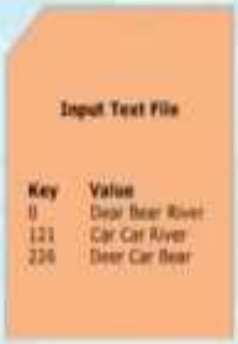


Steps with example



Mapper Code –Word Count Example





Byte Offset Type

Mapper Value Input Type

Mapper Key Output Type

Mapper Value Output Type

```
public static class Map extends Mapper<LongWritable,Text,Text,IntWritable> {

public void map(LongWritable key, Text value, Context context) throws IOException,InterruptedException {

String line = value.toString();
StringTokenizer tokenizer = new StringTokenizer(line);
while (tokenizer.hasMoreTokens()) {
value.set(tokenizer.nextToken());
context.write(value, new IntWritable(1));
}
```

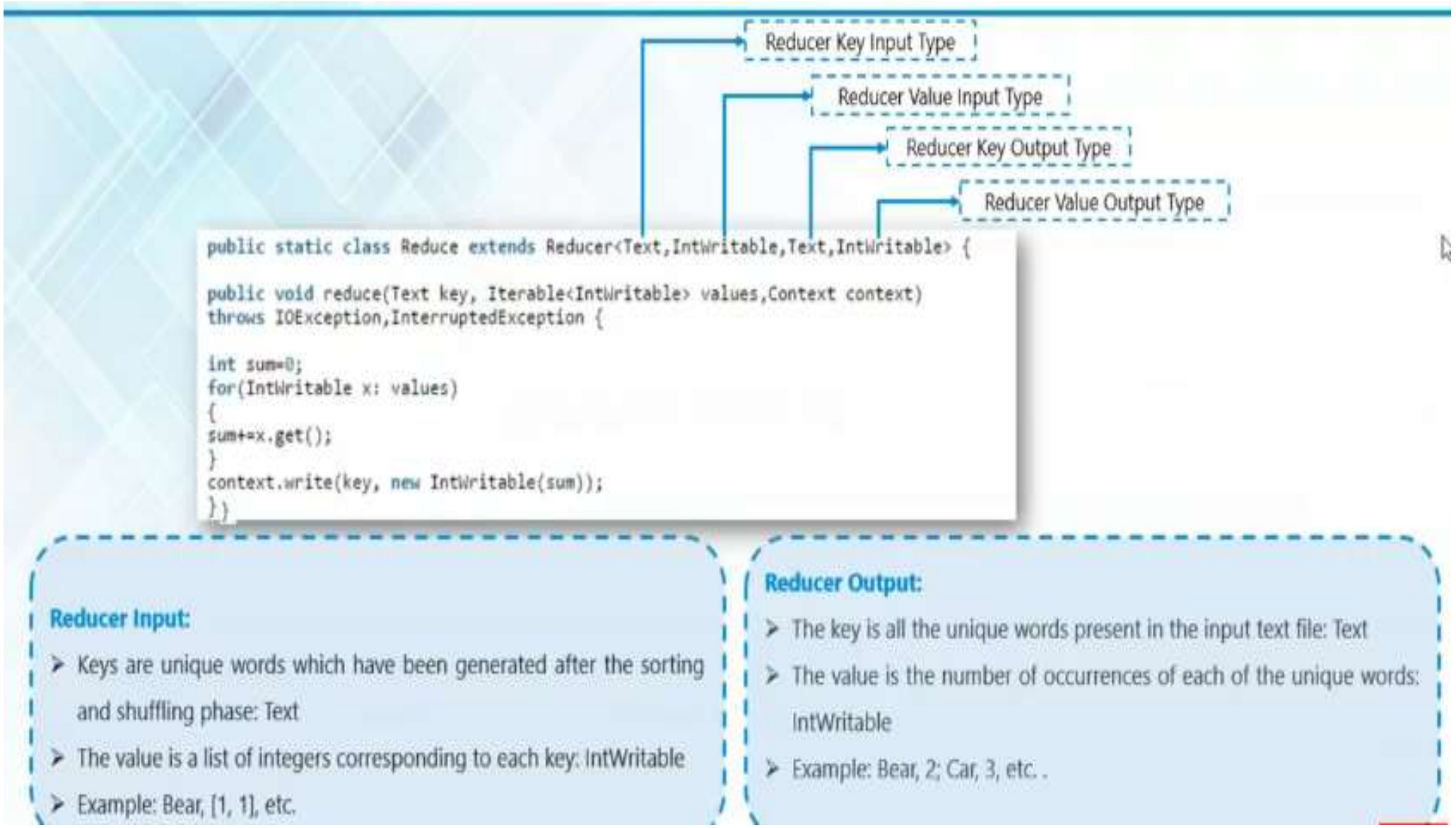
Mapper Input:

- The key is nothing but the offset of each line in the text file: *LongWritable*
- The value is each individual: *Text*

Mapper Output:

- The key is the tokenized words: Text
- We have the hardcoded value in our case which is 1: IntWritable
- Example – Dear 1, Bear 1, etc.

Reducer Code –Word Count Example



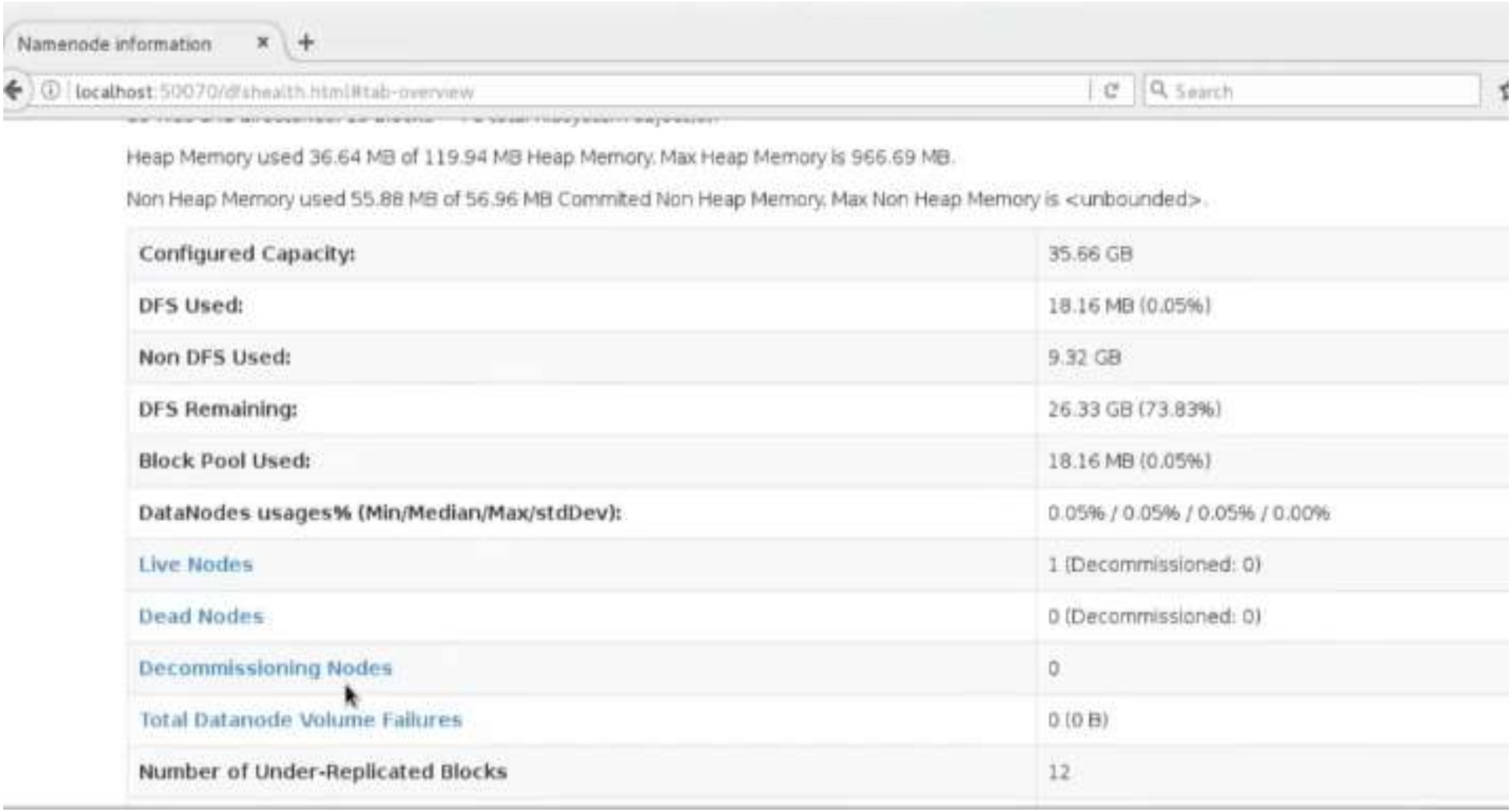
Driver Code for Configuration

In the driver class, we set the configuration of our MapReduce job to run in Hadoop

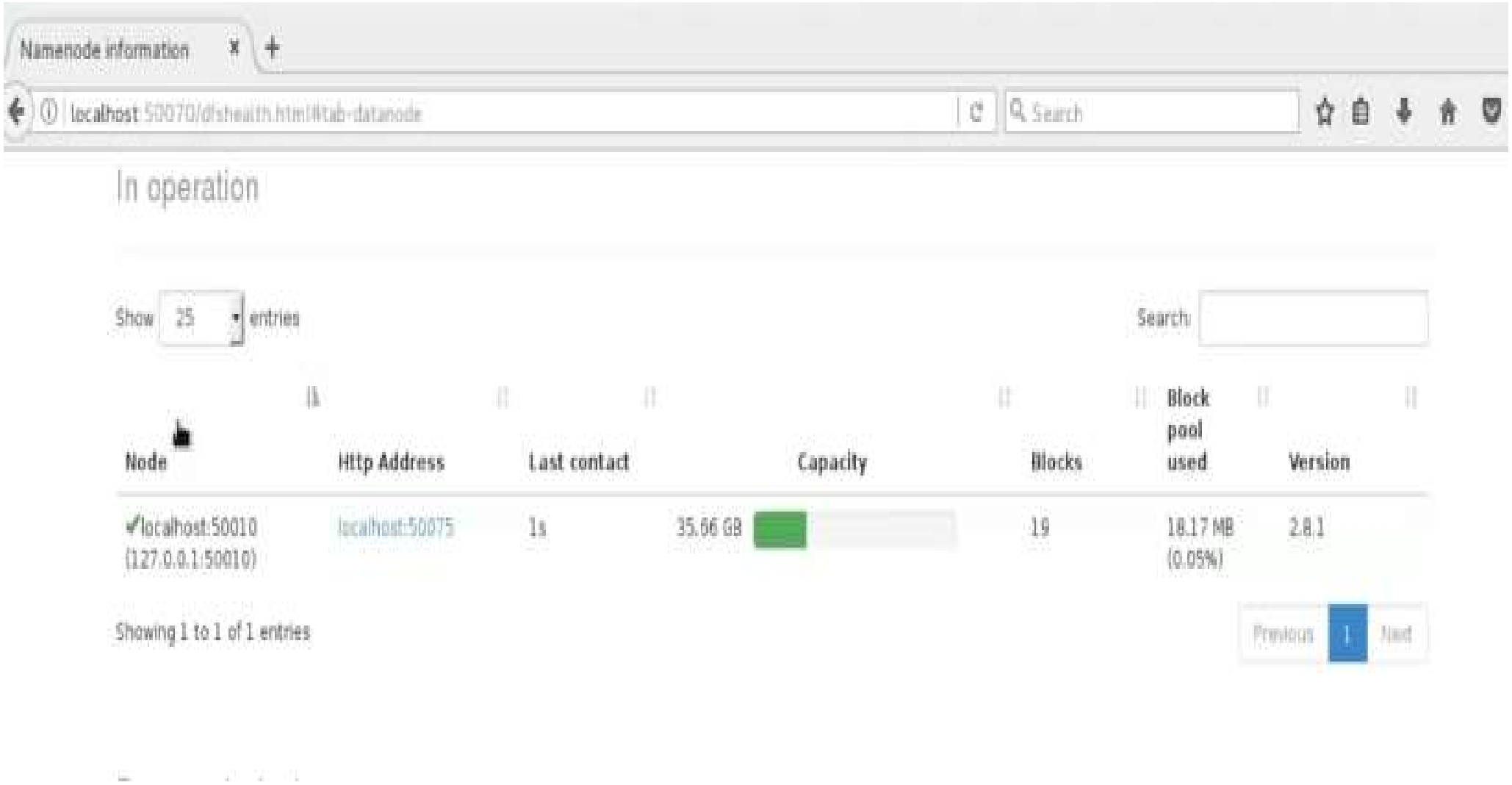
```
Configuration conf= new Configuration();
Job job = new Job(conf,"My Word Count Program");
job.setJarByClass(WordCount.class);
job.setMapperClass(Map.class);
job.setReducerClass(Reduce.class);
job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);
Path outputPath = new Path(args[1]);

//Configuring the input/output path from the filesystem into the job
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
```

- Specify the name of the job , the data type of input/output of the mapper and reducer
- Specify the names of the mapper and reducer classes.
- Path of the input and output folder
- The method setInputFormatClass () is used for specifying the unit of work for mapper
- Main() method is the entry point for the driver

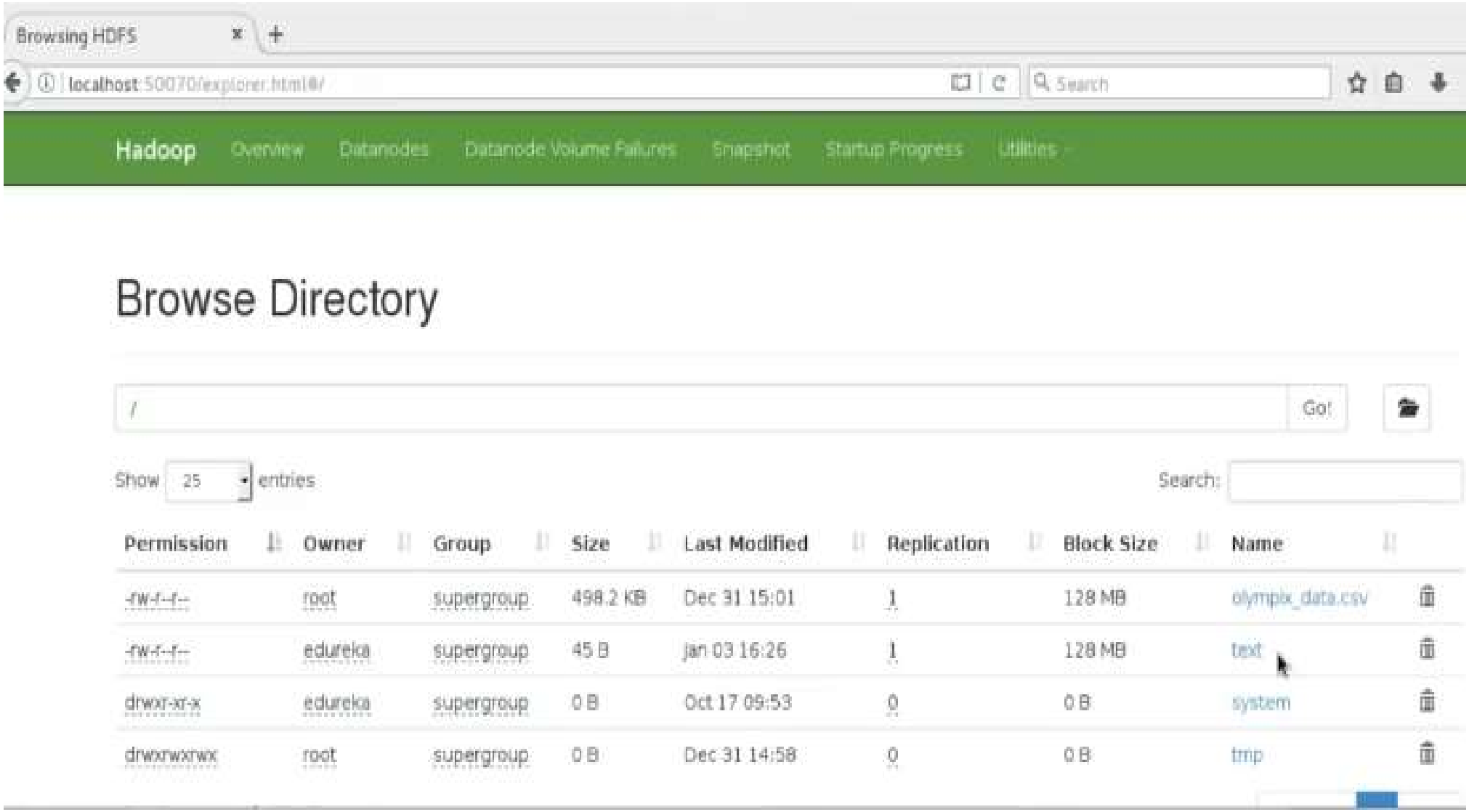
NameNode Summary



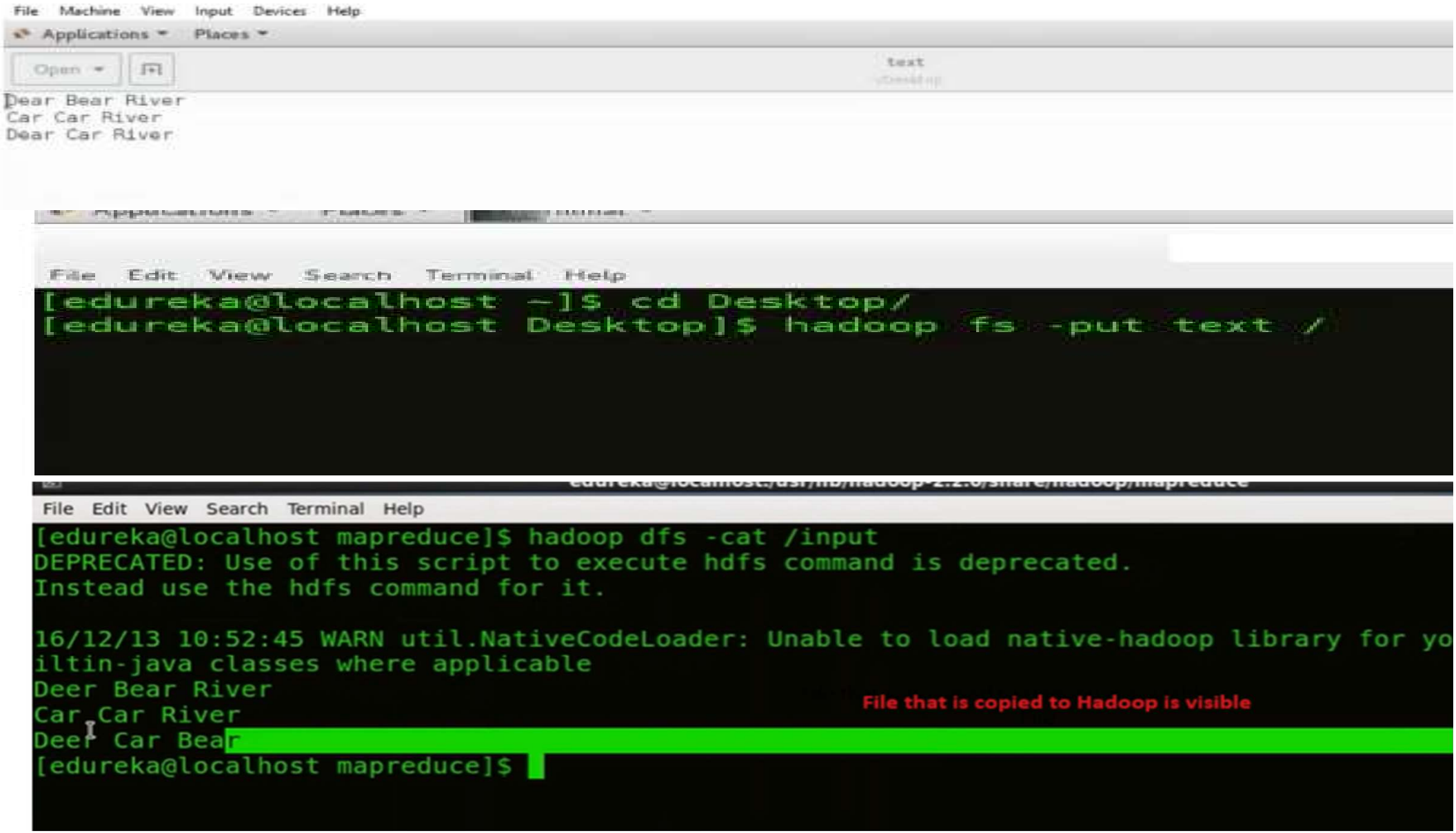
Live Node Details



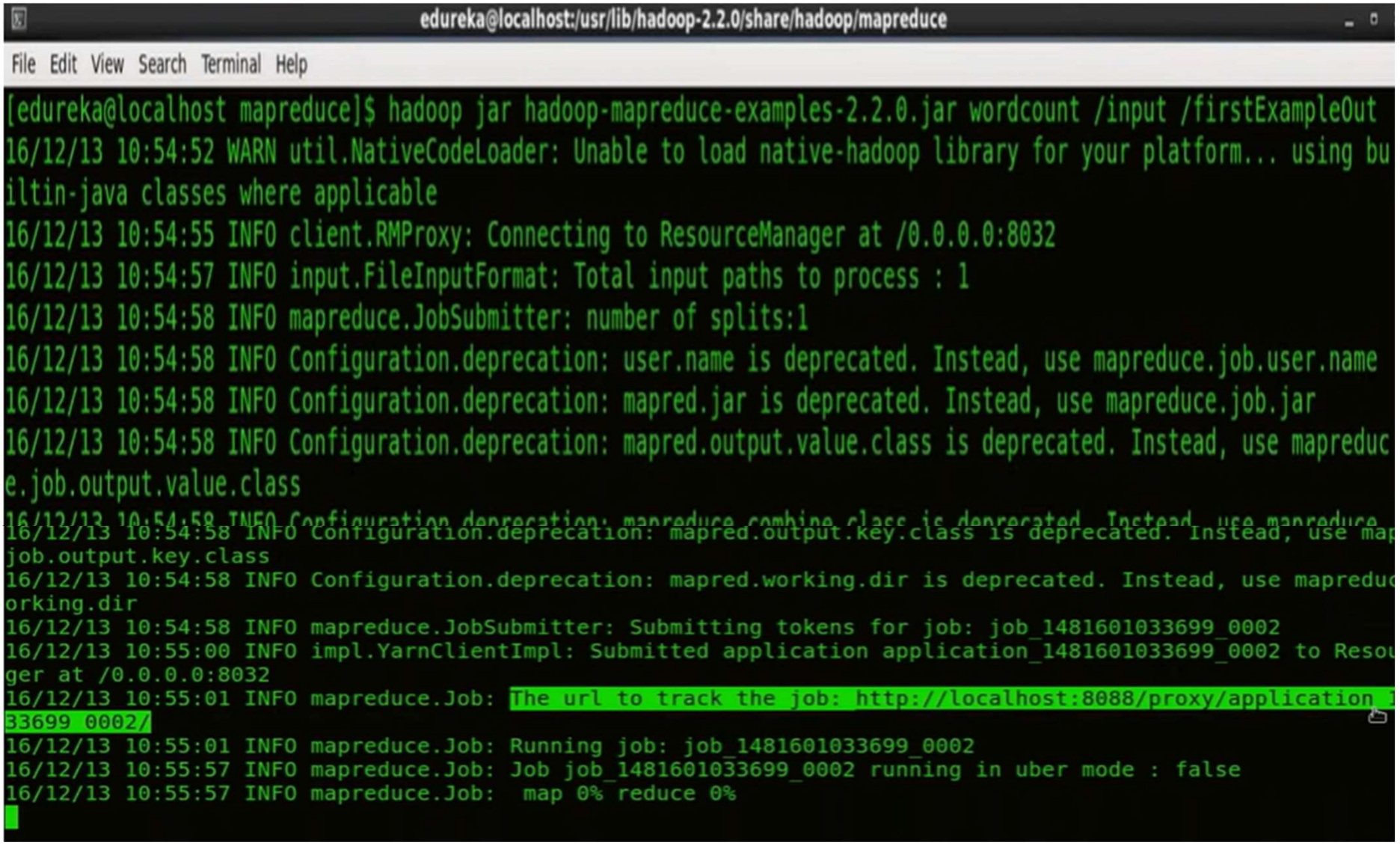
NameNode Files



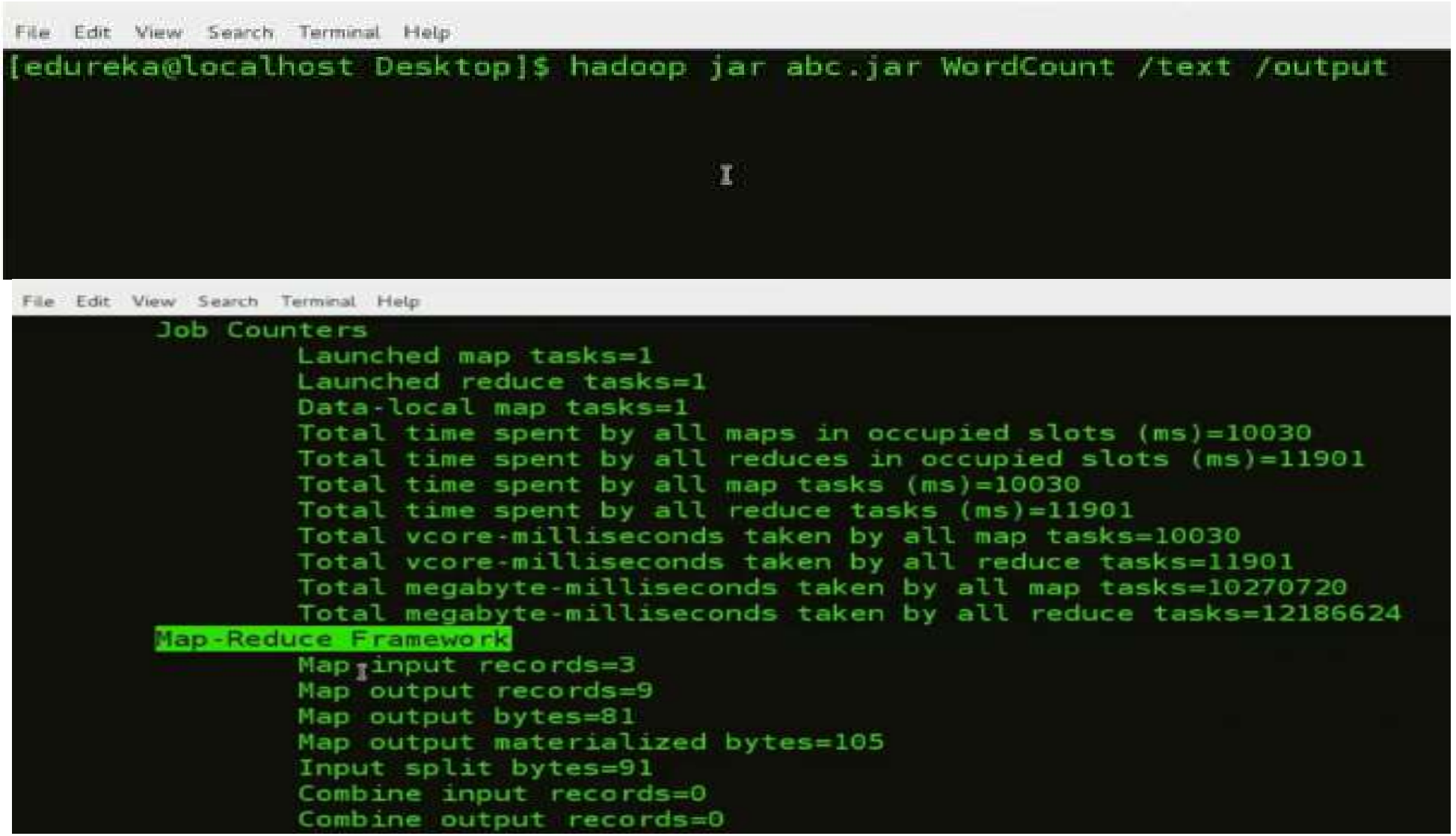
Sample Text File



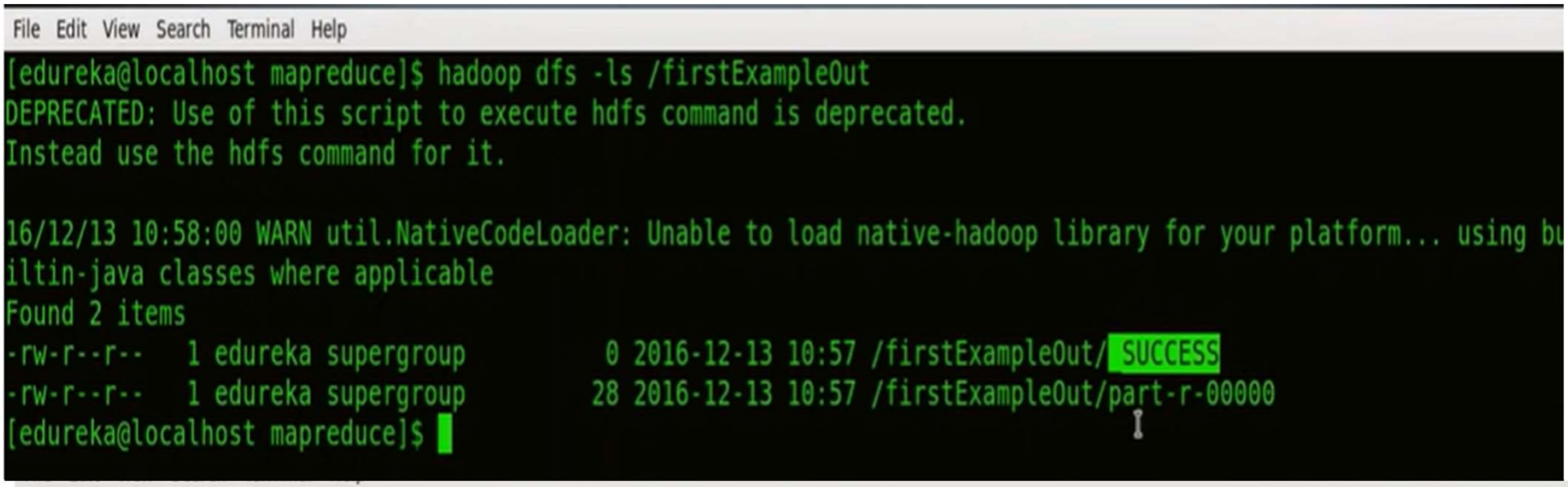
Execution

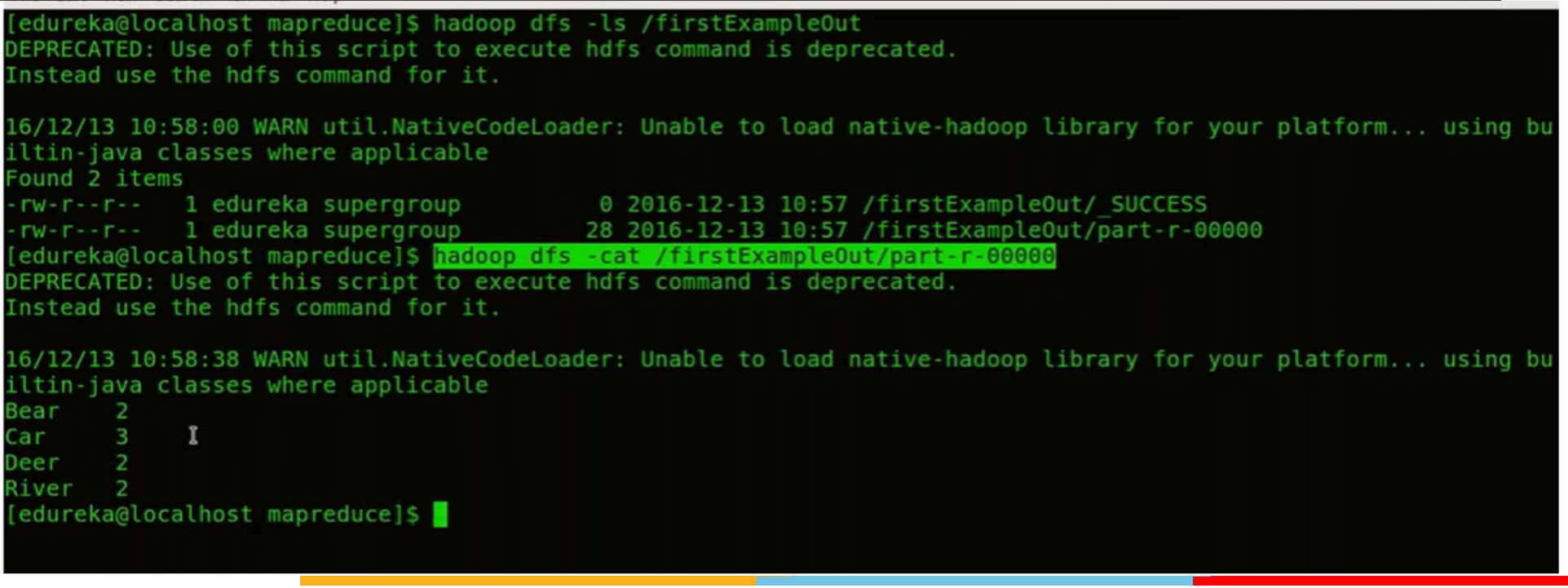

edureka@localhost:/usr/lib/hadoop-2.2.0/share/hadoop/mapreduce
File Edit View Search Terminal Help
[edureka@localhost mapreduce]$ hadoop jar hadoop-mapreduce-examples-2.2.0.jar wordcount /input /firstExampleOut
16/12/13 10:54:52 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using bu
iltin-java classes where applicable
16/12/13 10:54:55 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
16/12/13 10:54:57 INFO input.FileInputFormat: Total input paths to process : 1
16/12/13 10:54:58 INFO mapreduce.JobSubmitter: number of splits:1
16/12/13 10:54:58 INFO Configuration.deprecation: user.name is deprecated. Instead, use mapreduce.job.user.name
16/12/13 10:54:58 INFO Configuration.deprecation: mapred.jar is deprecated. Instead, use mapreduce.job.jar
16/12/13 10:54:58 INFO Configuration.deprecation: mapred.output.value.class is deprecated. Instead, use mapreduc
e.job.output.value.class
16/12/13 10:54:58 INFO Configuration.deprecation: mapred.output.key.class is deprecated. Instead, use map
job.output.key.class
16/12/13 10:54:58 INFO Configuration.deprecation: mapred.working.dir is deprecated. Instead, use mapreduc
orking.dir
16/12/13 10:54:58 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1481601033699_0002
16/12/13 10:55:00 INFO impl.YarnClientImpl: Submitted application application_1481601033699_0002 to Resou
ger at /0.0.0.0:8032
16/12/13 10:55:01 INFO mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_1
33699_0002/
16/12/13 10:55:01 INFO mapreduce.Job: Running job: job_1481601033699_0002
16/12/13 10:55:57 INFO mapreduce.Job: Job job_1481601033699_0002 running in uber mode : false
16/12/13 10:55:57 INFO mapreduce.Job: map 0% reduce 0%

Execution

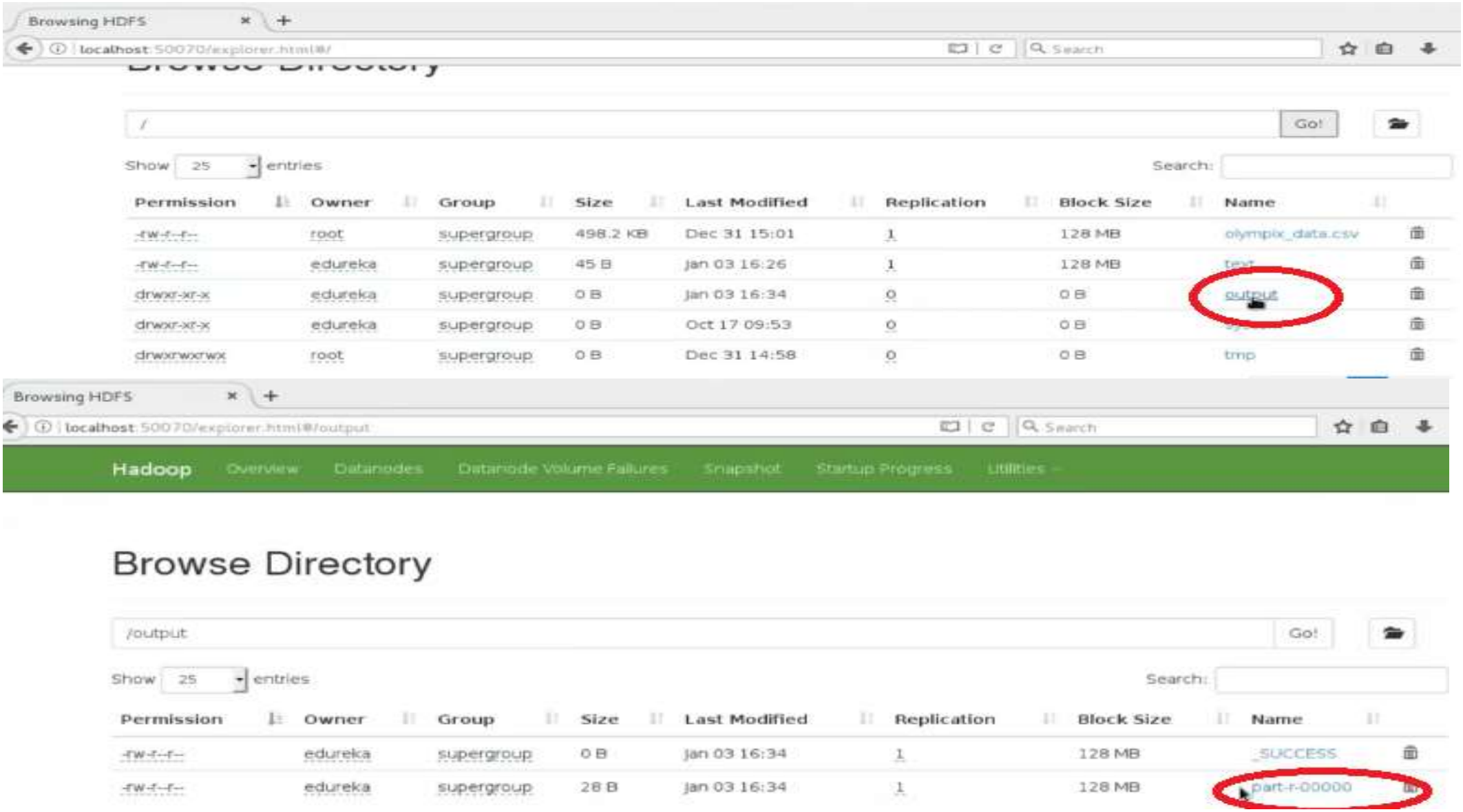

File Edit View Search Terminal Help
[edureka@localhost Desktop]$ hadoop jar abc.jar WordCount /text /output
File Edit View Search Terminal Help
Job Counters
Launched map tasks=1
Launched reduce tasks=1
Data-local map tasks=1
Total time spent by all maps in occupied slots (ms)=10030
Total time spent by all reduces in occupied slots (ms)=11901
Total time spent by all map tasks (ms)=10030
Total time spent by all reduce tasks (ms)=11901
Total vcore-milliseconds taken by all map tasks=10030
Total vcore-milliseconds taken by all reduce tasks=11901
Total megabyte-milliseconds taken by all map tasks=10270720
Total megabyte-milliseconds taken by all reduce tasks=12186624
Map-Reduce Framework
Map input records=3
Map output records=9
Map output bytes=81
Map output materialized bytes=105
Input split bytes=91
Combine input records=0
Combine output records=0

Ack

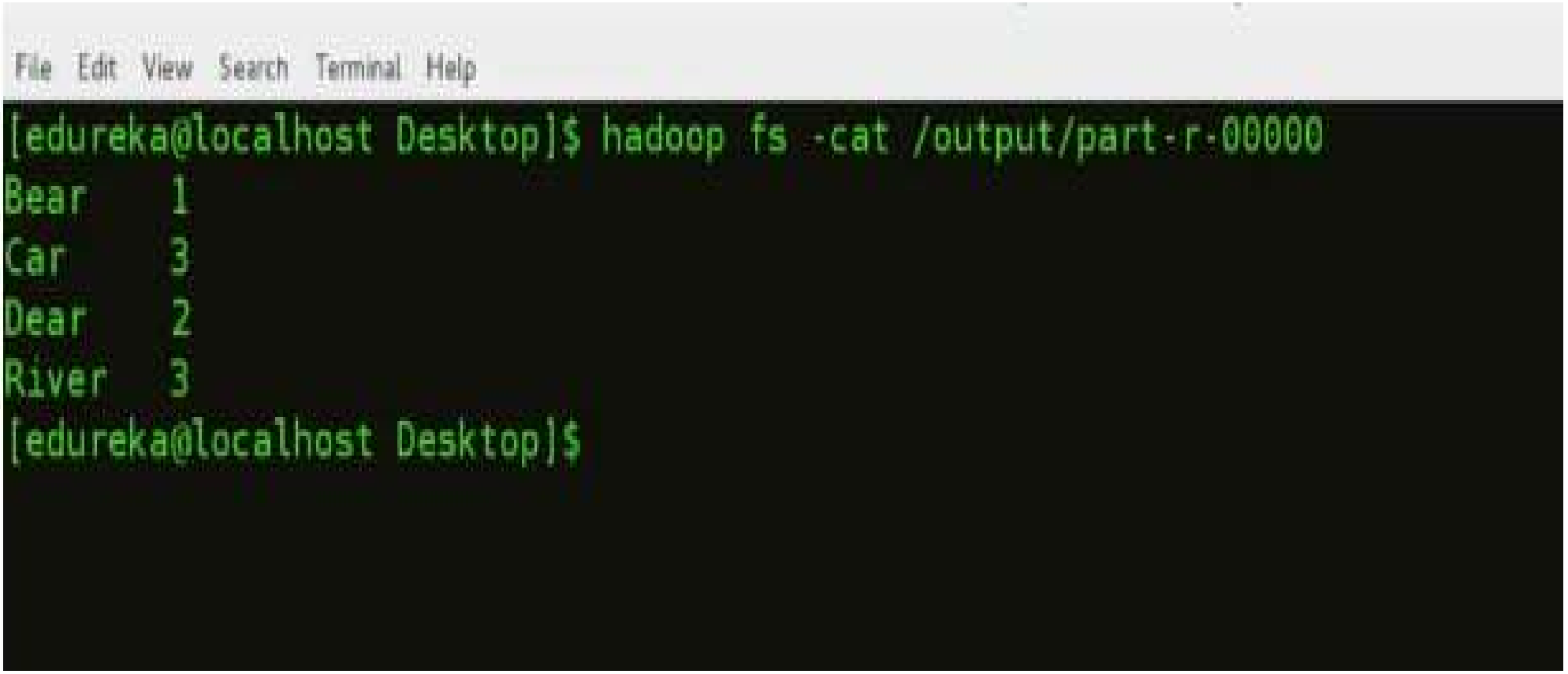

File Edit View Search Terminal Help
[edureka@localhost mapreduce]$ hadoop dfs -ls /firstExampleOut
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
16/12/13 10:58:00 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using bu
iltin-java classes where applicable
Found 2 items
-rw-r--r-- 1 edureka supergroup 0 2016-12-13 10:57 /firstExampleOut/_SUCCESS
-rw-r--r-- 1 edureka supergroup 28 2016-12-13 10:57 /firstExampleOut/part-r-00000
[edureka@localhost mapreduce]$


[edureka@localhost mapreduce]$ hadoop dfs -ls /firstExampleOut
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
16/12/13 10:58:00 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using bu
iltin-java classes where applicable
Found 2 items
-rw-r--r-- 1 edureka supergroup 0 2016-12-13 10:57 /firstExampleOut/_SUCCESS
-rw-r--r-- 1 edureka supergroup 28 2016-12-13 10:57 /firstExampleOut/part-r-00000
[edureka@localhost mapreduce]$ hadoop dfs -cat /firstExampleOut/part-r-00000
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
16/12/13 10:58:38 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using bu
iltin-java classes where applicable
Bear 2
Car 3
Deer 2
River 2
[edureka@localhost mapreduce]$

Result


Browsing HDFS
localhost:50070/explorer.html#/
Search
/
Go!
Show 25 entries
Search:
Permission Owner Group Size Last Modified Replication Block Size Name
-rw-r--r-- root supergroup 498.2 KB Dec 31 15:01 1 128 MB olympix_data.csv
-rw-r--r-- edureka supergroup 45 B Jan 03 16:26 1 128 MB text
drwxr-xr-x edureka supergroup 0 B Jan 03 16:34 0 0 B output
drwxr-xr-x edureka supergroup 0 B Oct 17 09:53 0 0 B
drwxrwxrwx root supergroup 0 B Dec 31 14:58 0 0 B tmp
Browsing HDFS
localhost:50070/explorer.html#/output
Search
Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities
Browse Directory
/output
Go!
Show 25 entries
Search:
Permission Owner Group Size Last Modified Replication Block Size Name
-rw-r--r-- edureka supergroup 0 B Jan 03 16:34 1 128 MB _SUCCESS
-rw-r--r-- edureka supergroup 28 B Jan 03 16:34 1 128 MB part-r-00000

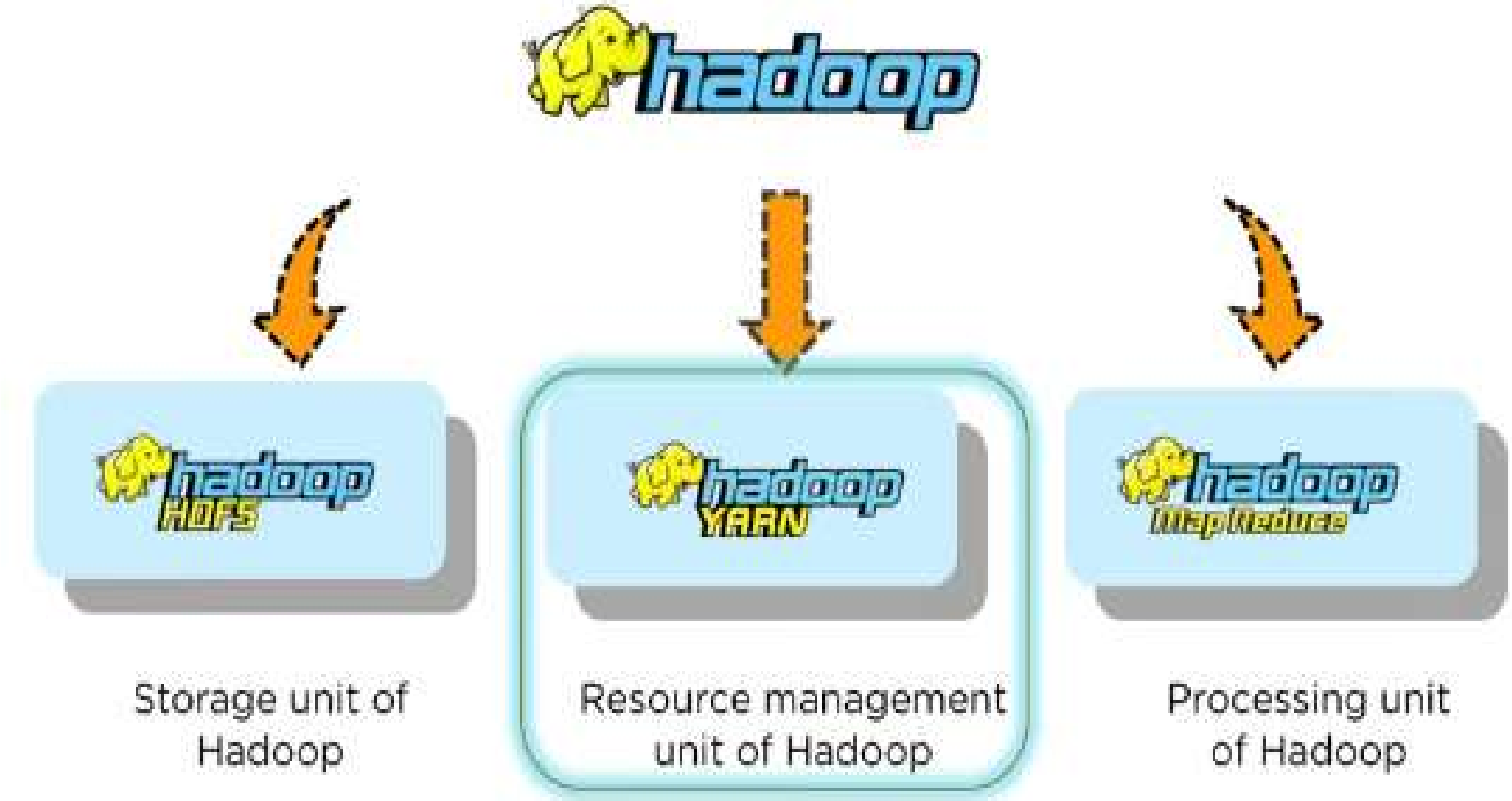
Result-Contd



Better Alternative

Property	Apache Spark	Hadoop MapReduce
Performance	Lightning-fast cluster computing (100 times faster)	Slower than Spark
Read/Write Cycle	Low	High
Usability	Easy	Requires code for every operation
Realtime Analysis	Supported	Not Supported
Latency	Low	High
Fault Tolerance	Supported	Supported
Security	Low	High
Cost	High (requires a lot of RAM)	Low
Developing Language	Scala	Java
Licenses	Free	Free
SQL Support	Spark SQL	Hive
Scalability	High	High
Machine Learning	MLLib	Apache Mahout
Data Caching	Supported	Not Supported
Hardware Requirements	High-Level hardware	Commodity hardware

Basic Hadoop-2 Architecture



YET

ANOTHER

RESOURCE

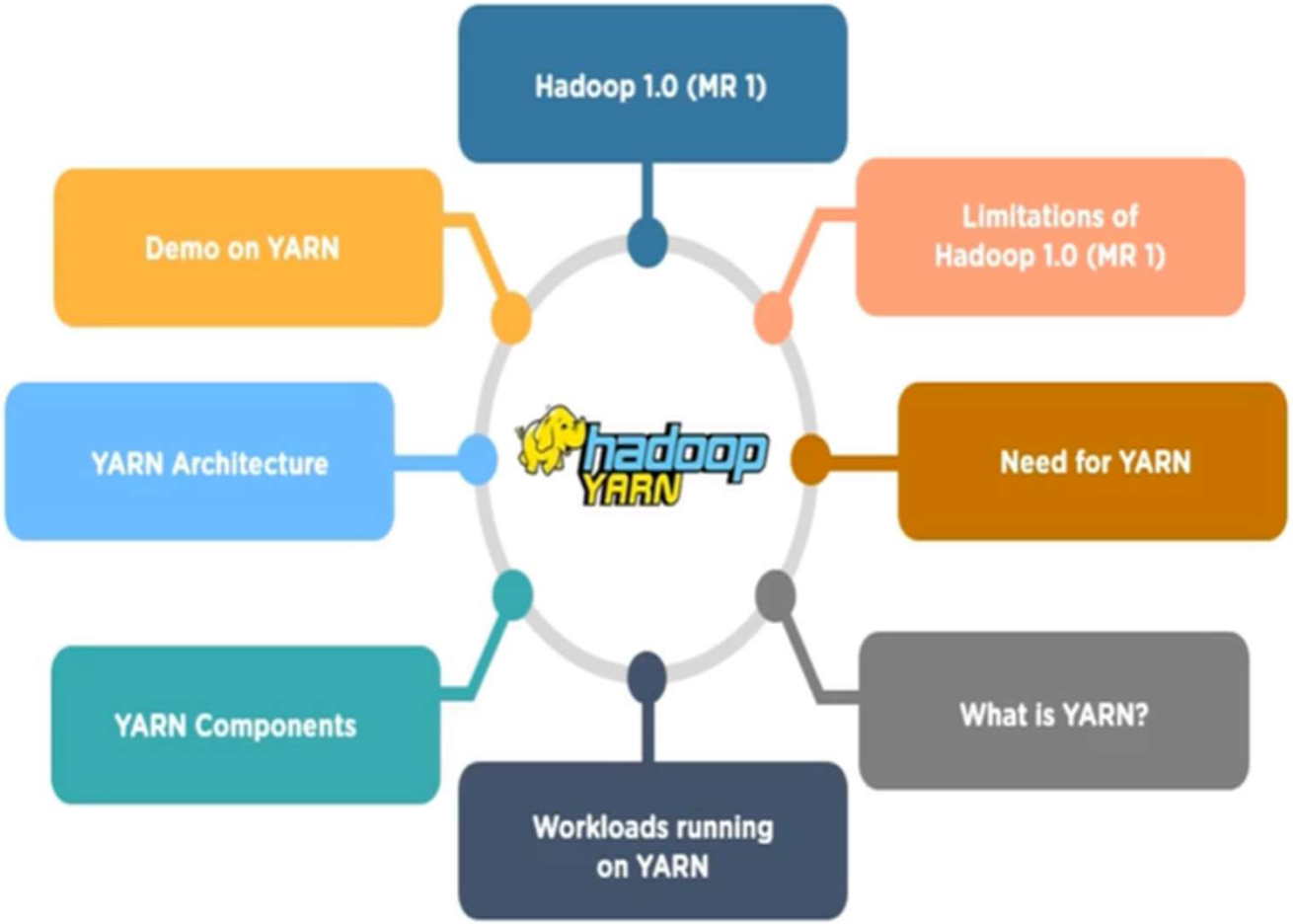
NEGOTIATOR

Coming Up……

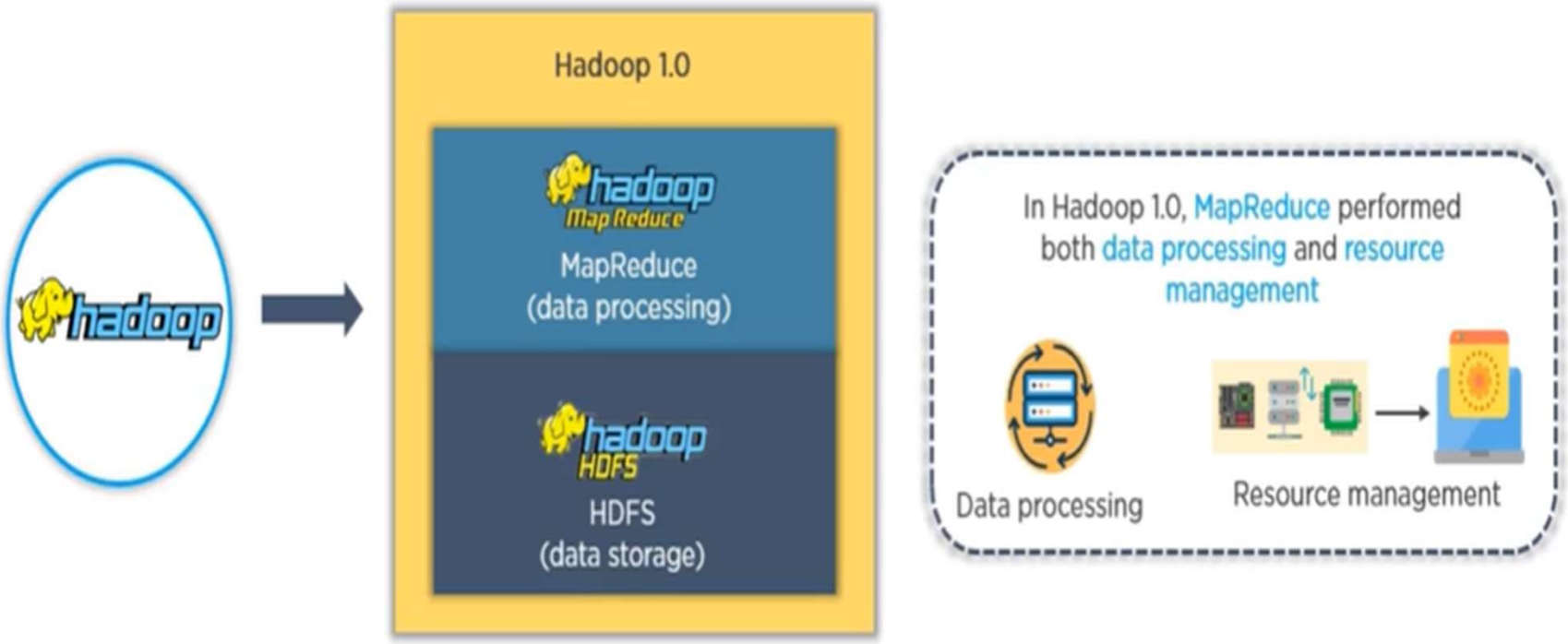
That is all for the day

Thank you

Todays Agenda



Hadoop-1


Hadoop 1.0
MapReduce
(data processing)
HDFS
(data storage)
In Hadoop 1.0, MapReduce performed both data processing and resource management
Data processing
Resource management

Hadoop-1

MapReduce consisted of
Job Tracker and Task Tracker

Job Tracker

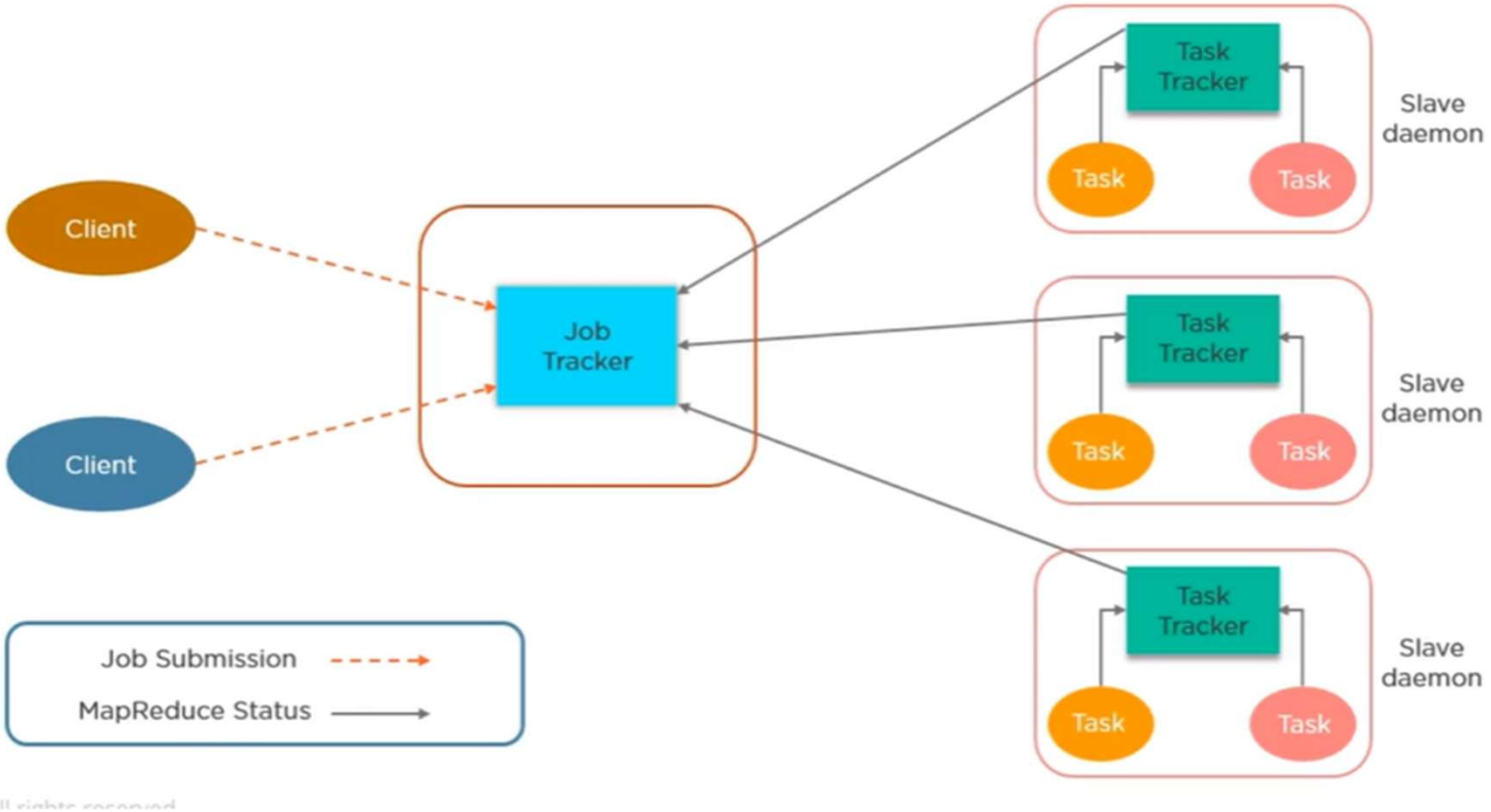
Allocated resources, performed scheduling and monitored jobs

Assigned map and reduce tasks to jobs running on Task Trackers

Task Tracker

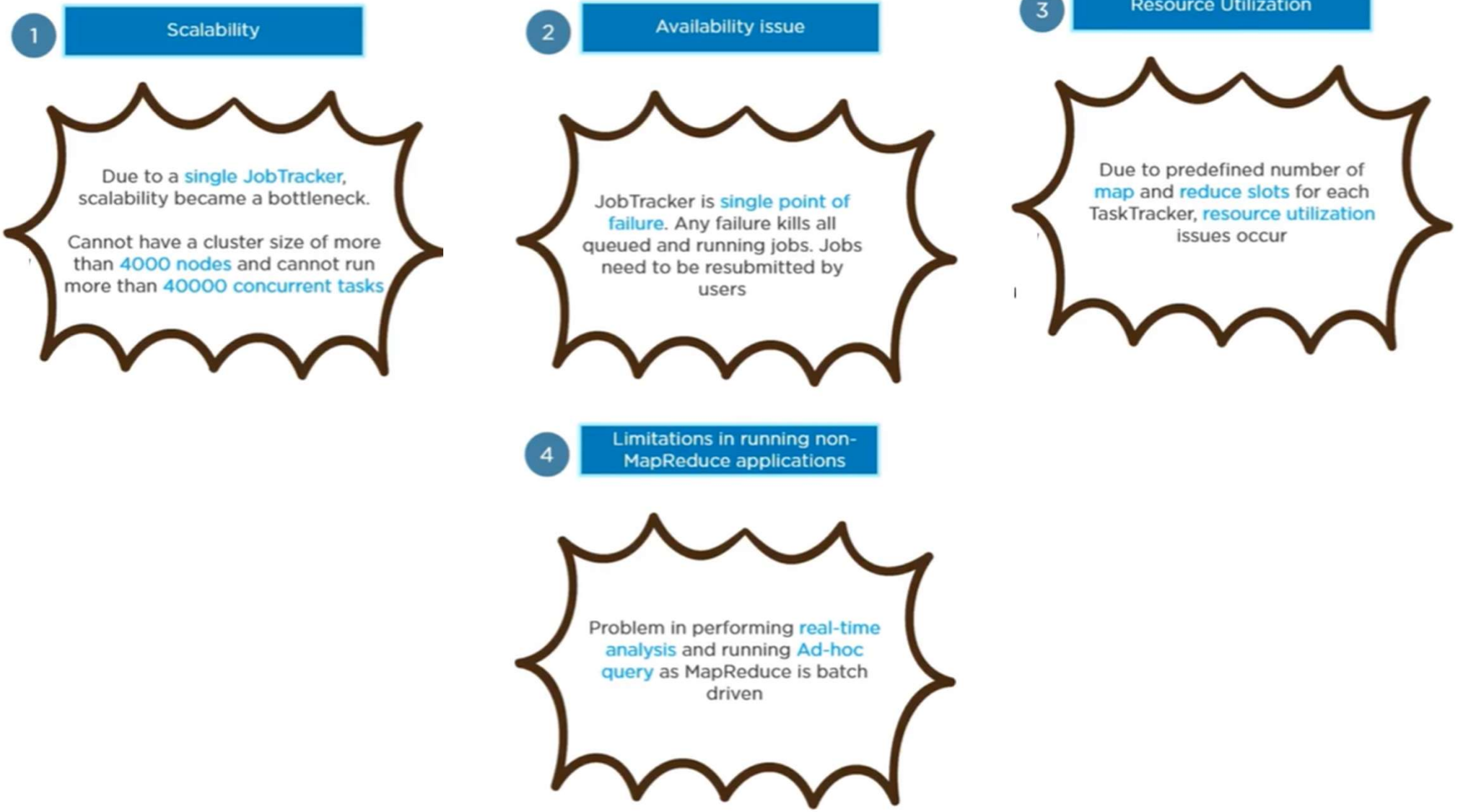
Task Trackers processed the jobs

Task Trackers reported their progress to the Job Tracker



Managing jobs using a single job tracker and utilization of computational resources was inefficient in MR 1

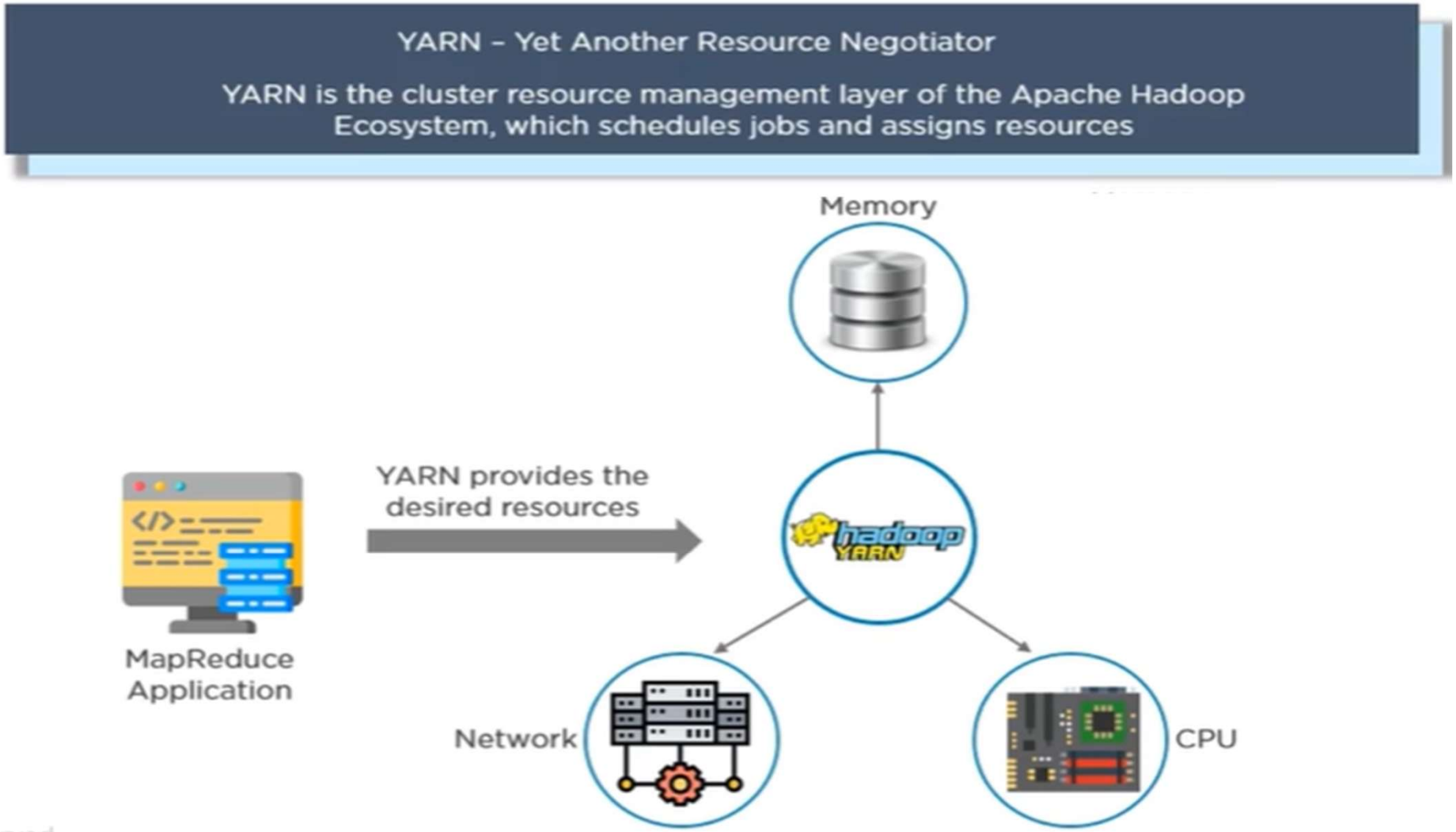
Limitations



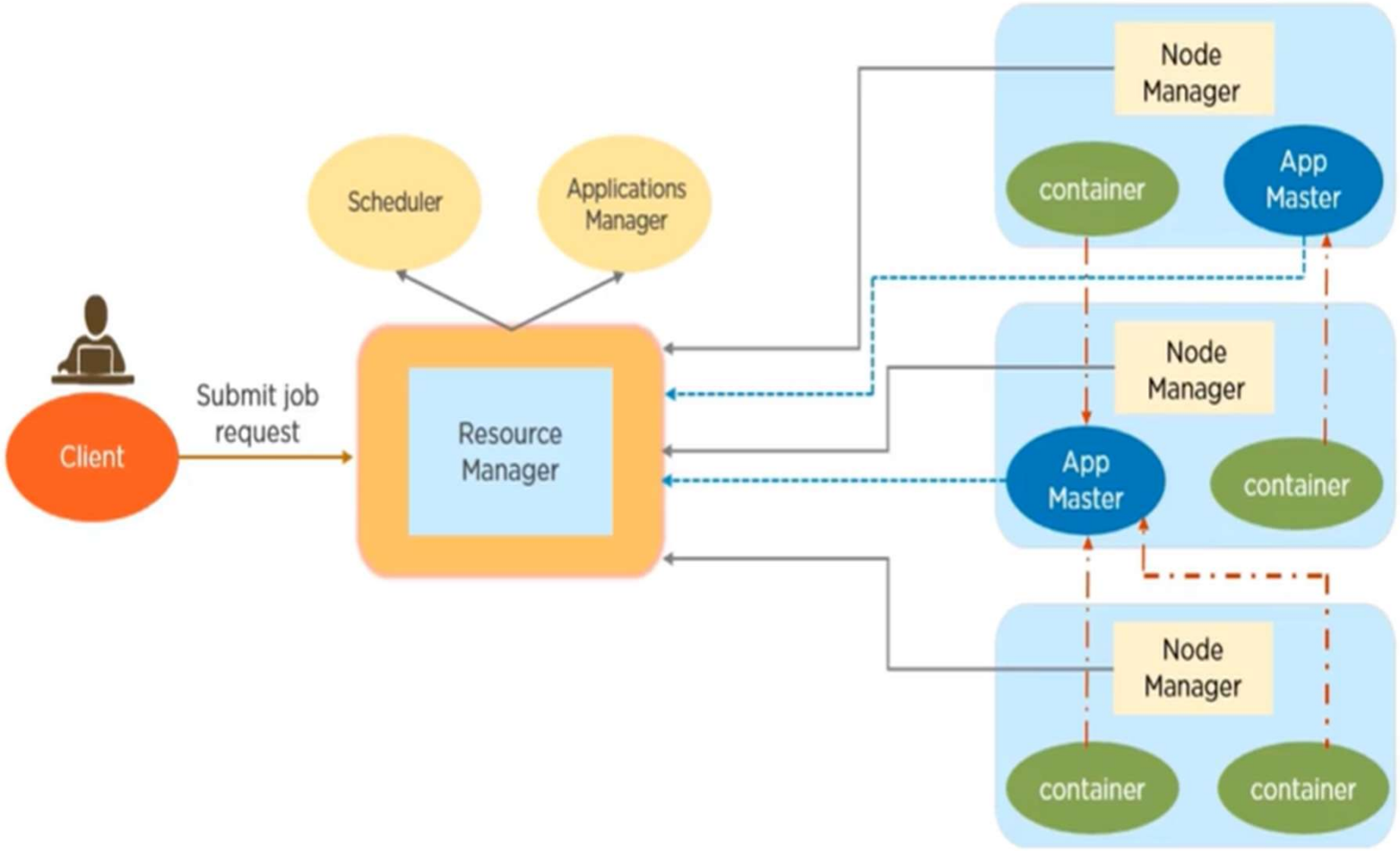
Solution

Scalability	Compatibility	Resource utilization	Multitenancy
			
Can have a cluster size of more than 10,000 nodes and can run more than 1,00,000 concurrent tasks	Applications developed for Hadoop 1 runs on YARN without any disruption or availability issues	Allows dynamic allocation of cluster resources to improve resource utilization	Can use open-source and propriety data access engines and perform real-time analysis and running ad-hoc query

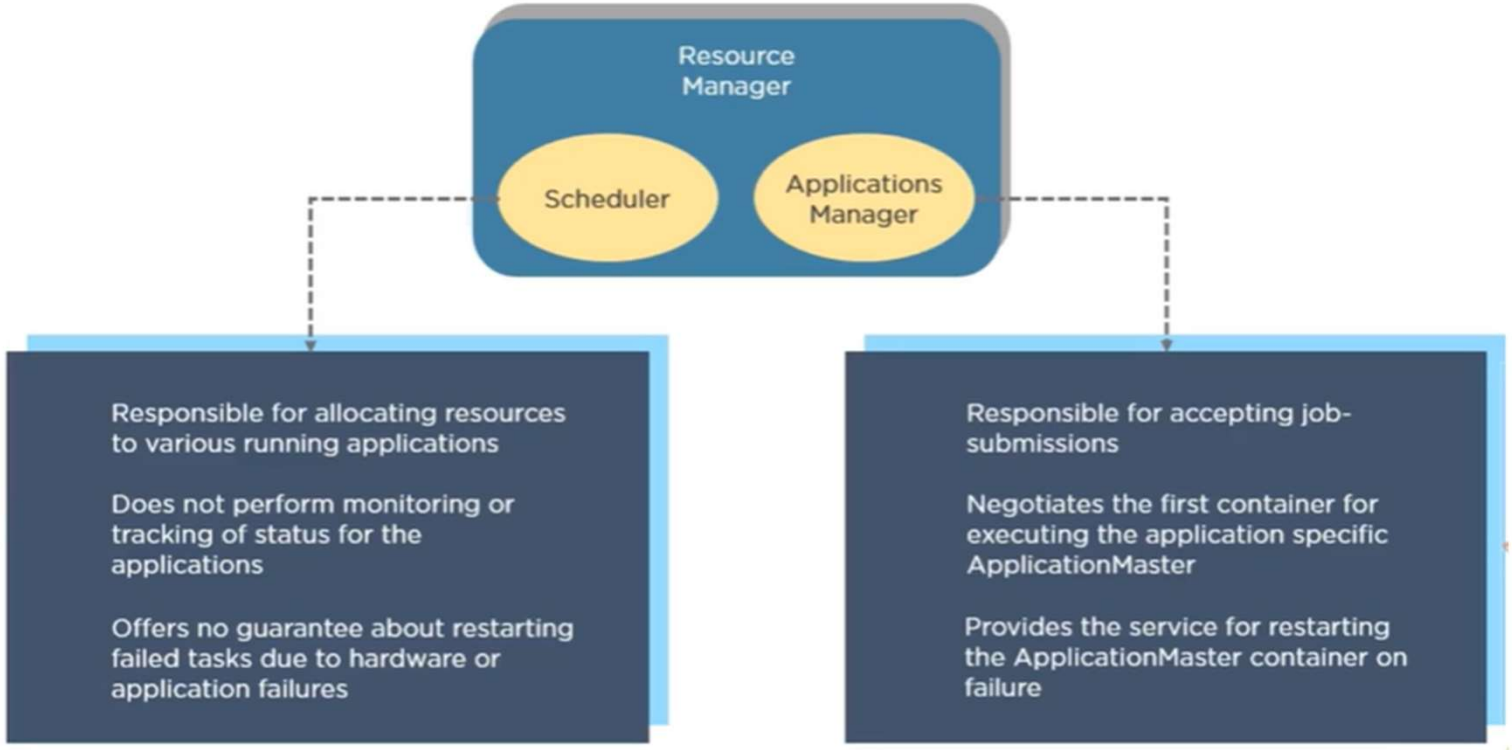
What is YARN ?


YARN – Yet Another Resource Negotiator
YARN is the cluster resource management layer of the Apache Hadoop Ecosystem, which schedules jobs and assigns resources
Memory
YARN provides the desired resources
hadoop YARN
MapReduce Application
Network
CPU

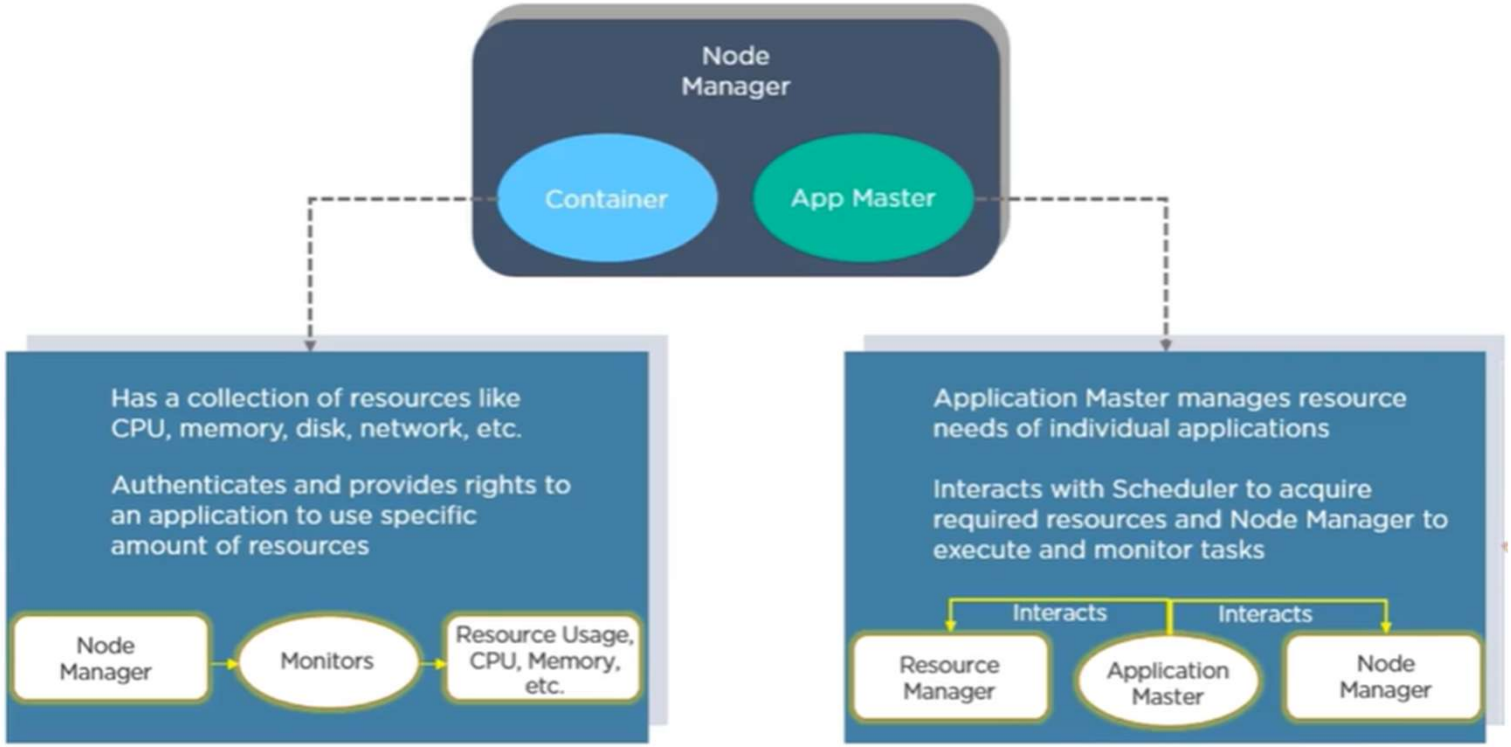
Architecture



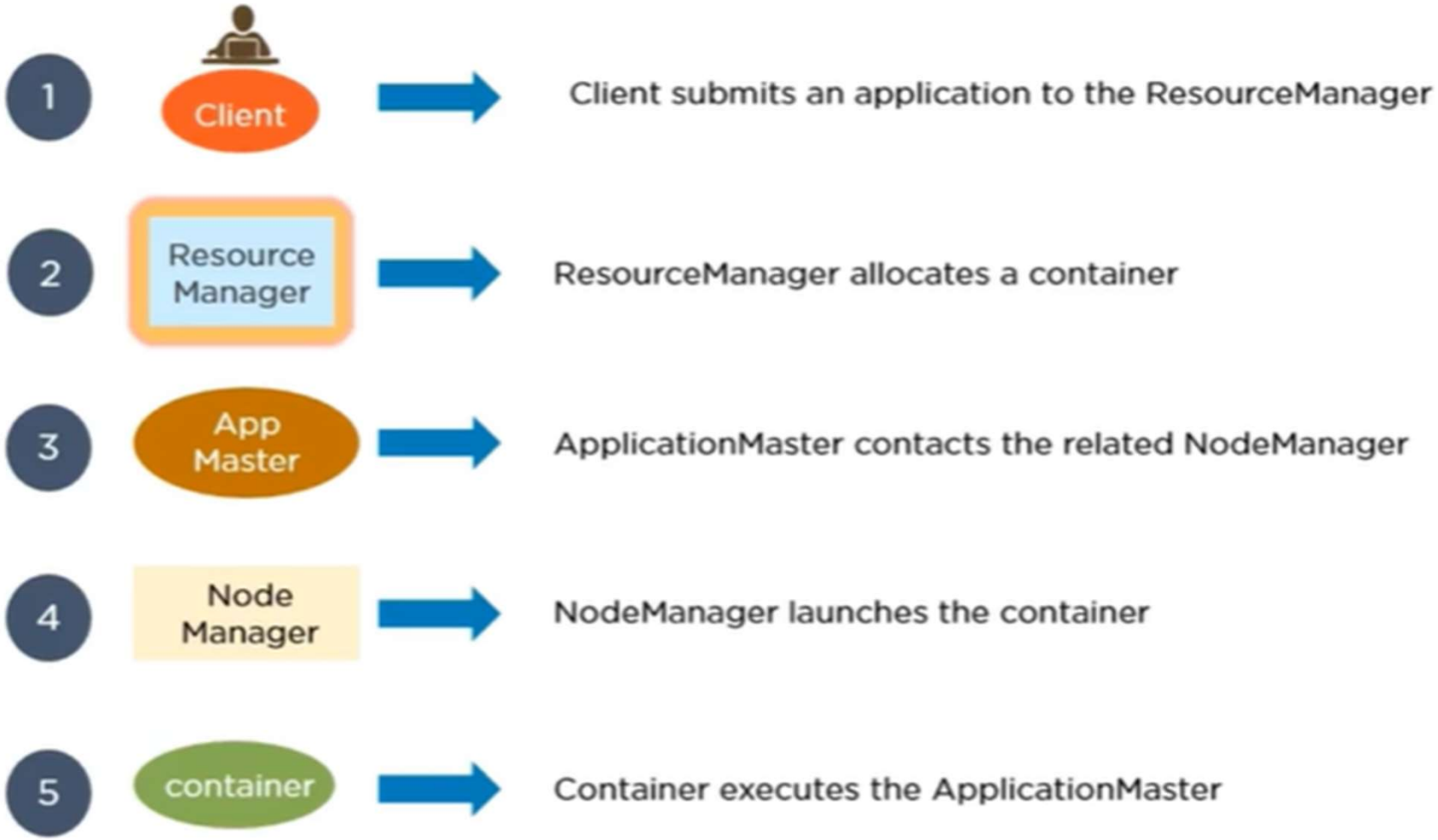
Resource Manager



Node Manager

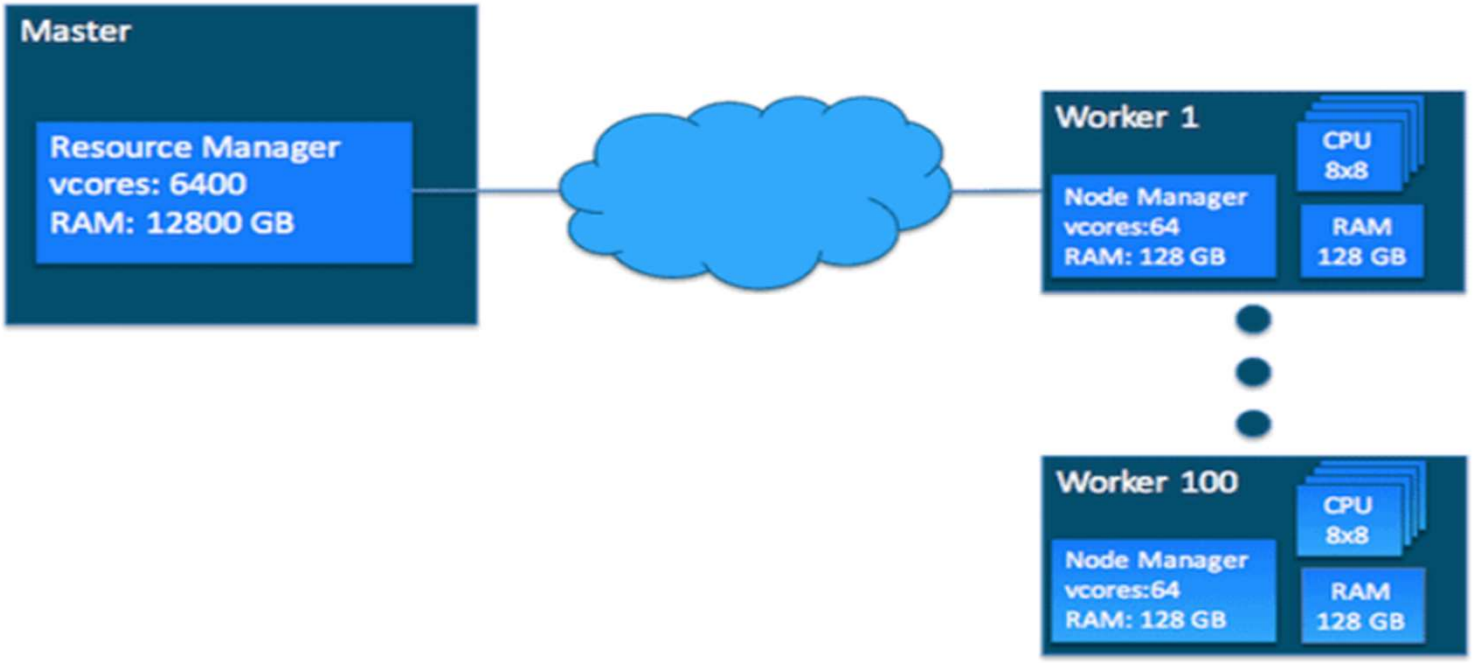


Team work


1
Client
Client submits an application to the ResourceManager
2
Resource Manager
ResourceManager allocates a container
3
App Master
ApplicationMaster contacts the related NodeManager
4
Node Manager
NodeManager launches the container
5
container
Container executes the ApplicationMaster

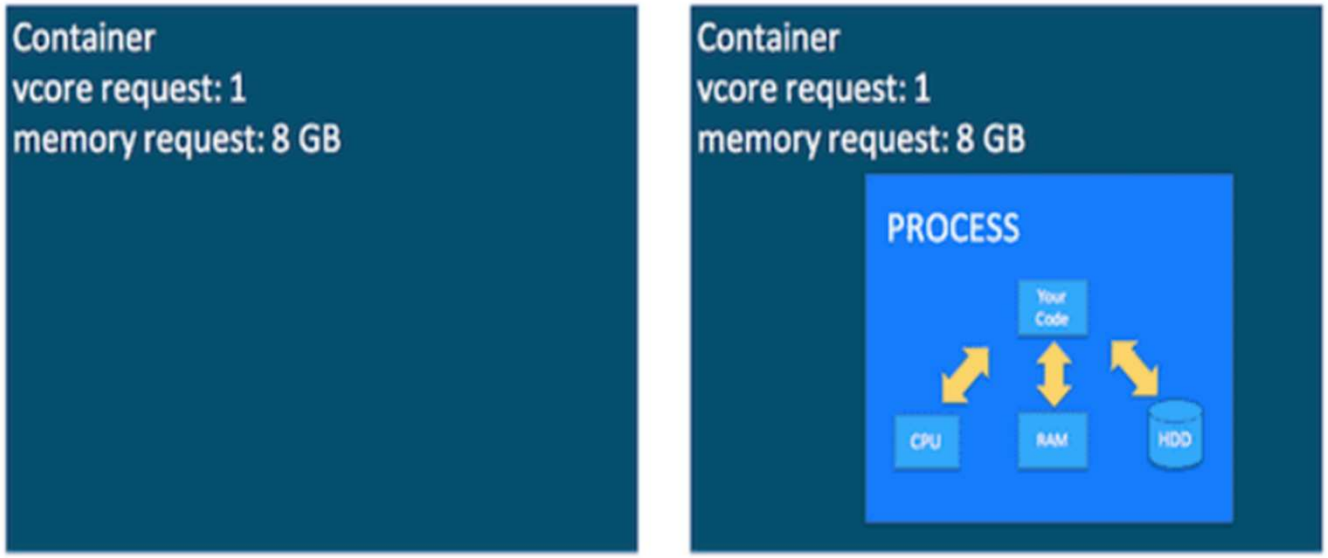
Global View

YARN currently defines two resources, *vcores* and *memory*. Each NodeManager tracks its own local resources and communicates its resource configuration to the ResourceManager, which keeps a running total of the cluster's available resources. By keeping track of the total, the ResourceManager knows how to allocate resources as they are requested. (Vcore has a special meaning in YARN. You can think of it simply as a "usage share of a CPU core." If you expect your tasks to be less CPU-intensive (sometimes called I/O-intensive), you can set the ratio of vcores to physical cores higher than 1 to maximize your use of hardware resources.)



Container

Containers are an important YARN concept. You can think of a container as a request to hold resources on the YARN cluster. Currently, a container hold request consists of vcore and memory shown in Figure



Container as a hold (left), and container as a running process (right)

Once a hold has been granted on a host, the NodeManager launches a process called a *task*. The right side of Figure shows the task running as a process inside a container.

Running Process/App Master

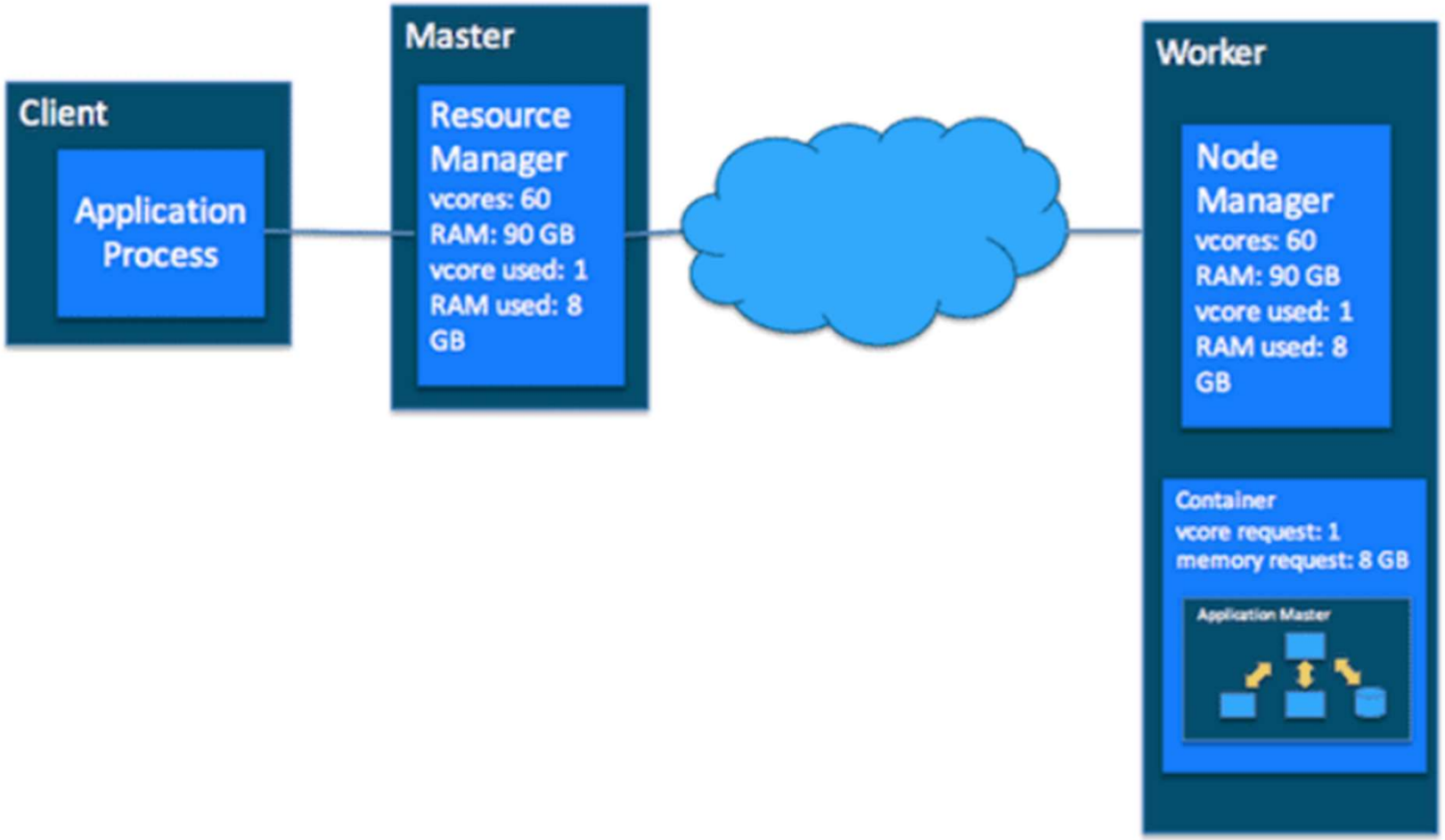
1. The application starts and talks to the ResourceManager for the cluster:

Figure 5: Application starting up before tasks are assigned to the cluster

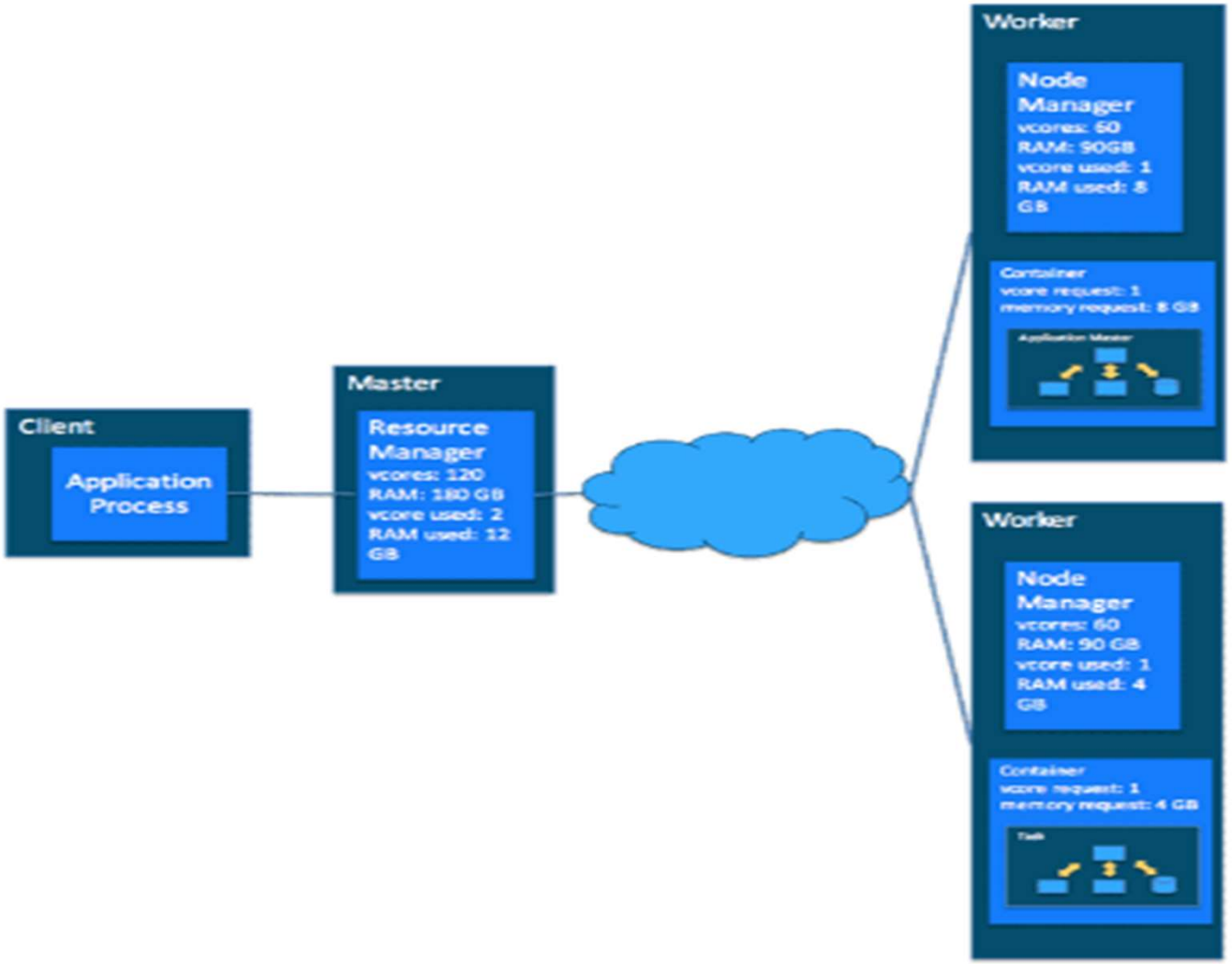
2. The ResourceManager makes a single container request on behalf of the application:

Step 3

3. The ApplicationMaster starts running within that container:



4. The ApplicationMaster requests subsequent containers from the ResourceManager that are allocated to run tasks for the application. Those tasks do most of the status communication with the ApplicationMaster allocated in Step 3):



Putting It Together

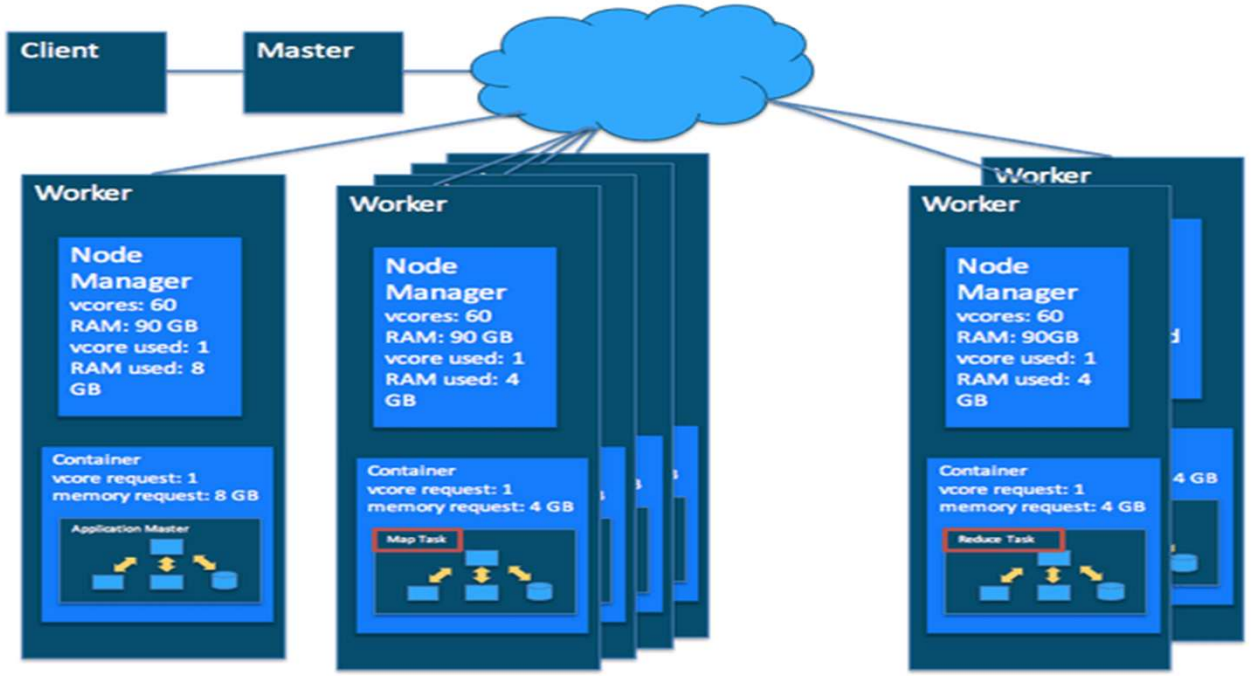


Figure 10: Merged MapReduce/YARN Application Running on a Cluster

In a MapReduce application, there are multiple map tasks, each running in a container on a worker host somewhere in the cluster. Similarly, there are multiple reduce tasks, also each running in a container on a worker host.

Simultaneously on the YARN side, the ResourceManager, NodeManager, and ApplicationMaster work together to manage the cluster's resources and ensure that the tasks, as well as the corresponding application, finish cleanly.

YARN Installation facts

In reality, there are two reasons why the full set of resources on a node cannot be allocated to YARN:

1. Non-Apache Hadoop services are also required to be running on a node (overhead).
2. Other Hadoop-related components require dedicated resources and cannot be shared with YARN (such as when running CDH).

The Intel-backed group develops CDH, a distribution of Hadoop that includes several other open-source projects, such as Impala and Search. It also offers security and integration features. The Impala framework is an interactive SQL query engine that allows direct queries of data stored in HDFS, Apache HBase, or AWS S3.

YARN in RM UI

As mentioned before, the ResourceManager has a snapshot of the available resources on the YARN cluster.

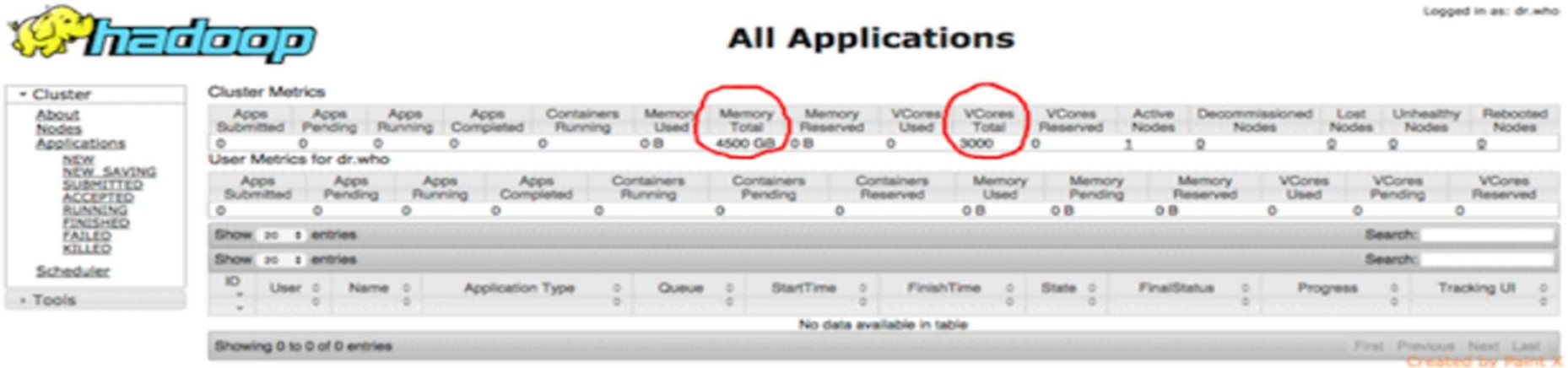
Example: Assume you have the following configuration on your 50 Worker nodes:

1. `yarn.nodemanager.resource.memory-mb` = 90000
2. `yarn.nodemanager.resource.vcores` = 60

Doing the math, your cluster totals should be:

1. memory: 50*90GB=4500GB=4.5TB
2. vcores: 50*60 vcores= 3000 vcores

On the ResourceManager Web UI page, the cluster metrics table shows the total memory and total vcores for the cluster, as seen in Figure 2 below.



Container Config

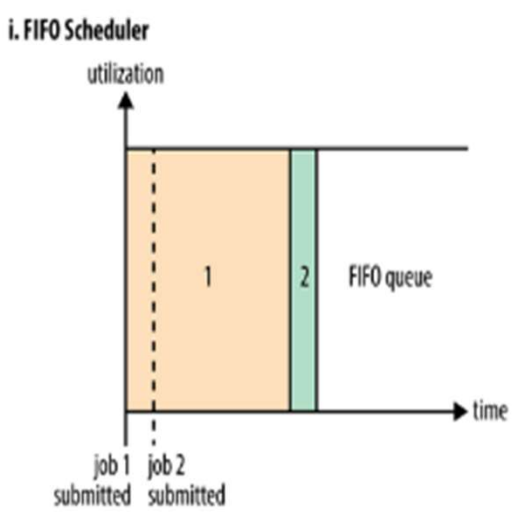
At this point, the YARN Cluster is properly set up in terms of Resources. YARN uses these resource limits for allocation, and enforces those limits on the cluster.

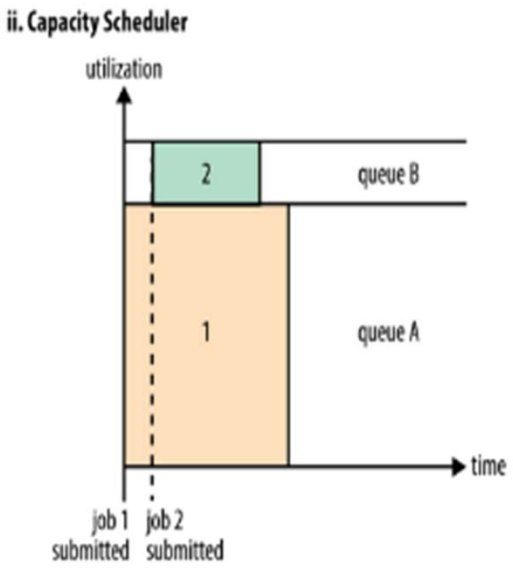
- YARN Container Memory Sizing
  - Minimum: `yarn.scheduler.minimum-allocation-mb`
  - Maximum: `yarn.scheduler.maximum-allocation-mb`
- YARN Container VCore Sizing
  - Minimum: `yarn.scheduler.minimum-allocation-vcores`
  - Maximum: `yarn.scheduler.maximum-allocation-vcores`
- YARN Container Allocation Size Increments
  - Memory Increment: `yarn.scheduler.increment-allocation-mb`
  - VCore Increment: `yarn.scheduler.increment-allocation-vcores`

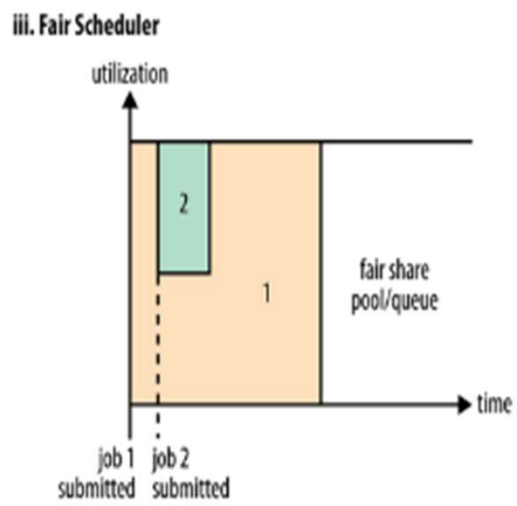
Restrictions

- Memory properties:
  - Minimum required value of 0 for `yarn.scheduler.minimum-allocation-mb`.
  - Any of the memory sizing properties must be less than or equal to `yarn.nodemanager.resource.memory-mb`.
  - Maximum value must be greater than or equal to the minimum value.
- VCore properties:
  - Minimum required value of 0 for `yarn.scheduler.minimum-allocation-vcores`.
  - Any of the vcore sizing properties must be less than or equal to `yarn.nodemanager.resource.vcores`.
  - Maximum value must be greater than or equal to the minimum value.
  - Recommended value of 1 for `yarn.scheduler.increment-allocation-vcores`. Higher values will likely be wasteful.

Scheduling YARN







Jump To Next PPT

Add YARN Service

without directly providing AWS credentials, subject to having the proper permissions defined via Sentry.
protocol.

Sentry	Sentry service stores authorization policy metadata and provides clients concurrent and secure access t
Solr	Solr is a distributed service for indexing and searching data stored in HDFS.
Spark	Apache Spark is an open source cluster computing system. This service runs Spark as an application on
Spark (Standalone)	Apache Spark is an open source cluster computing system. This is the standalone version of the service management. Cloudera recommends using Spark on YARN instead of this standalone version.
Sqoop 1 Client	Configuration and connector management for Sqoop 1.
Sqoop 2	Sqoop is a tool designed for efficiently transferring bulk data between Apache Hadoop and structured da version supported by Cloudera Manager is Sqoop 2.
YARN (MR2 Included)	Apache Hadoop MapReduce 2.0 (MRv2), or YARN, is a data computation framework that supports MapR
ZooKeeper	Apache ZooKeeper is a centralized service for maintaining and synchronizing configuration data.

Add YARN (MR2 Included) Service to Cluster 1

Customize Role Assignments for YARN (MR2 Included)

You can customize the role assignments for your new service here, but note that if assignments are made incorrectly, such as assigning too many suffer.

You can also view the role assignments by host. View By Host

RM ResourceManager × 1 New — ip-172-31-9-240.ap-southeast-1.comput...

JHS JobHistory Server × 1 New — ip-172-31-0-223.ap-southeast-1.comput...

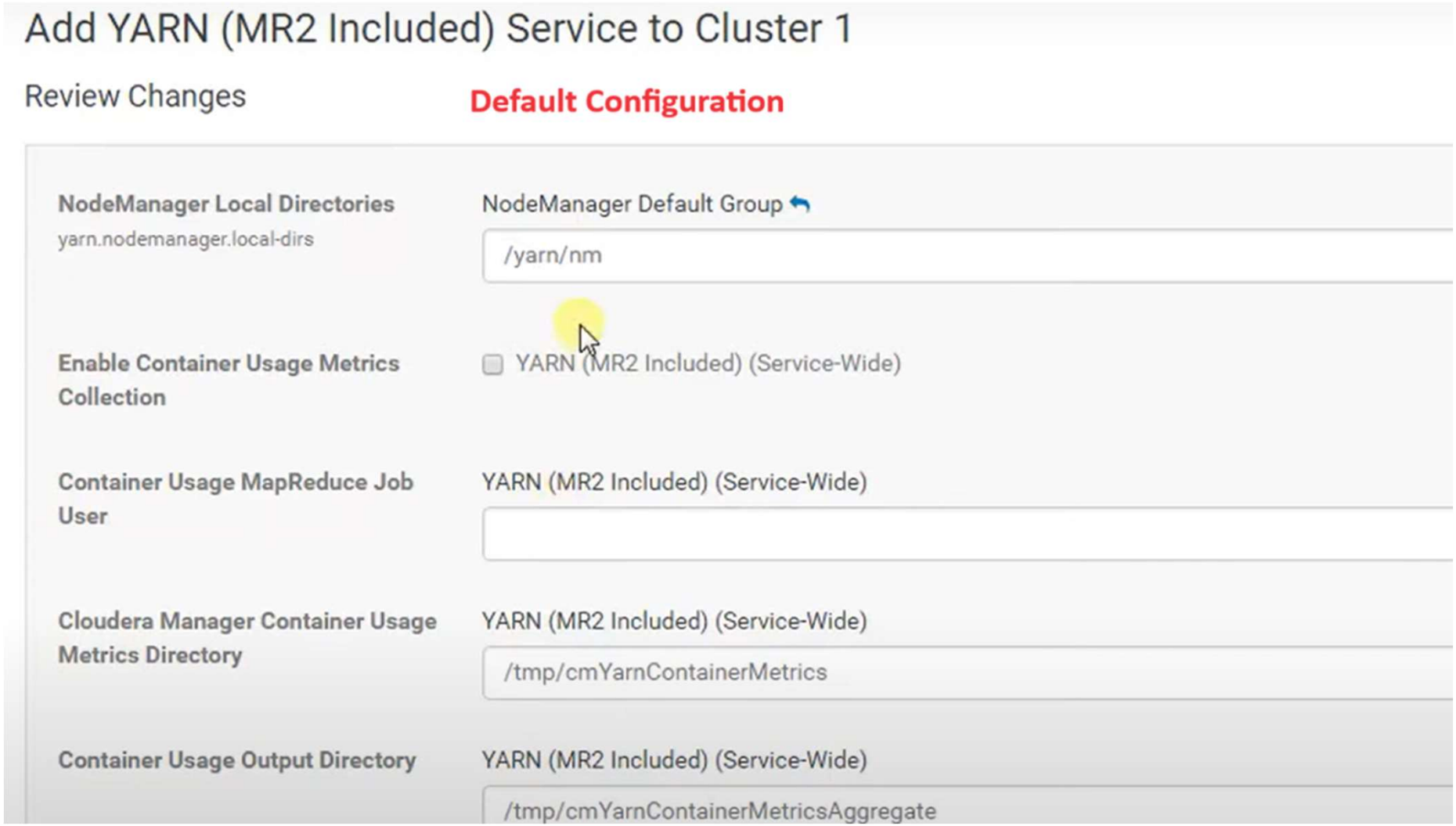
NM NodeManager × 3 New — ip-172-31-0-223.ap-southeast-1.comput...

Select hosts for a new or existing role. The host list is filtered to remove hosts that are not valid candidates; these include hosts that are unhealthy, members of other clusters, or hav incompatible version of CDH installed on them.

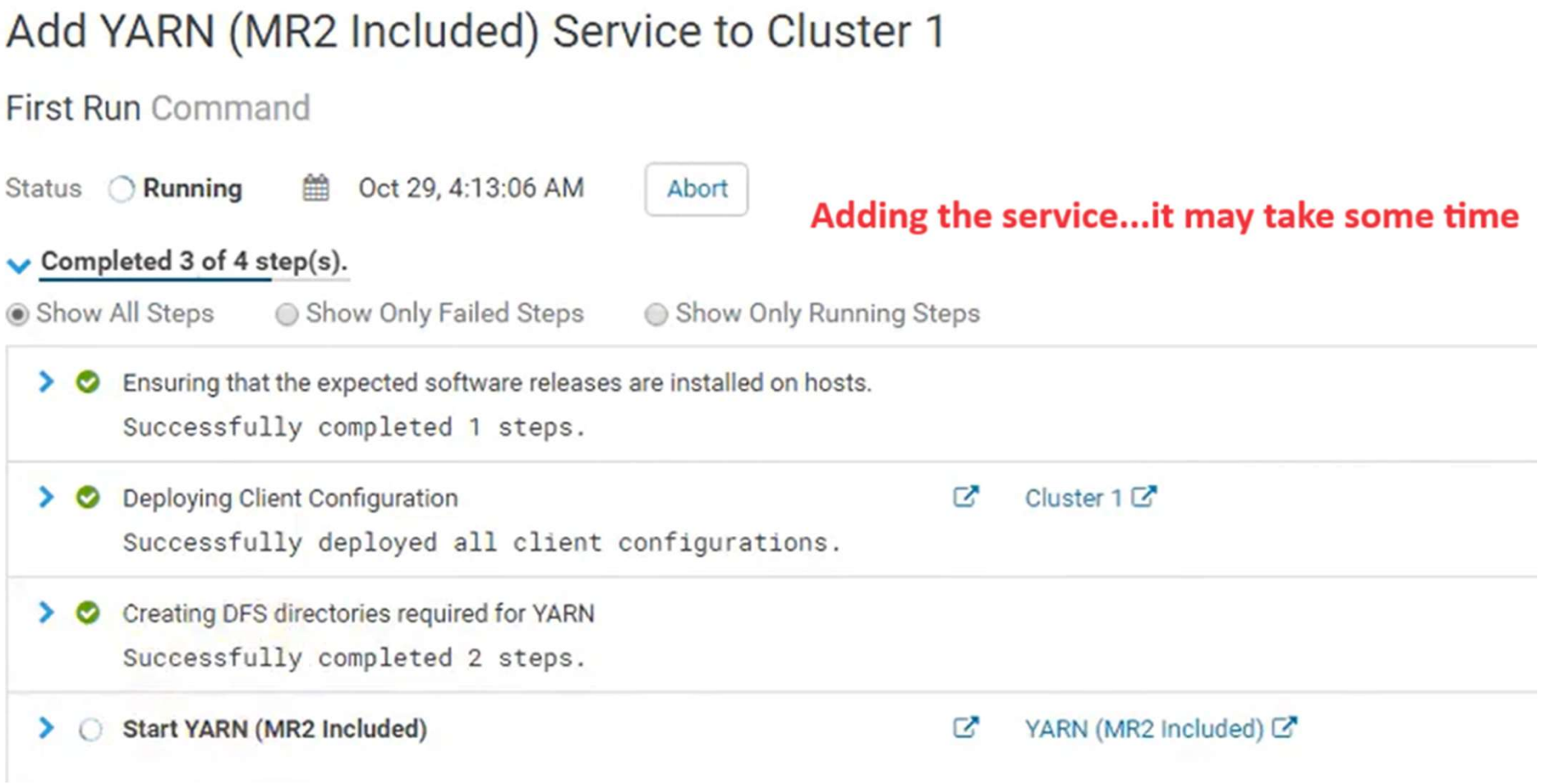
Enter hostnames: host01, host[01-10], IP addresses or rack. Search

	Hostname	IP Address	Rack	Cores	Physical Memory	Existing Roles	Added Role
☐	ip-172-31-0-223.ap-southeast-1.compute.internal	172.31.0.223	/default	2	7.7 GiB	B DN AP SM	NM
☐	ip-172-31-14-33.ap-southeast-1.compute.internal	172.31.14.33	/default	2	7.7 GiB	DN HM	NM
☐	ip-172-31-5-165.ap-southeast-1.compute.internal	172.31.5.165	/default	2	7.7 GiB	DN SNN RM	NM
☑	ip-172-31-9-240.ap-southeast-1.compute.internal	172.31.9.240	/default	2	7.7 GiB	NN ES	RM JHS

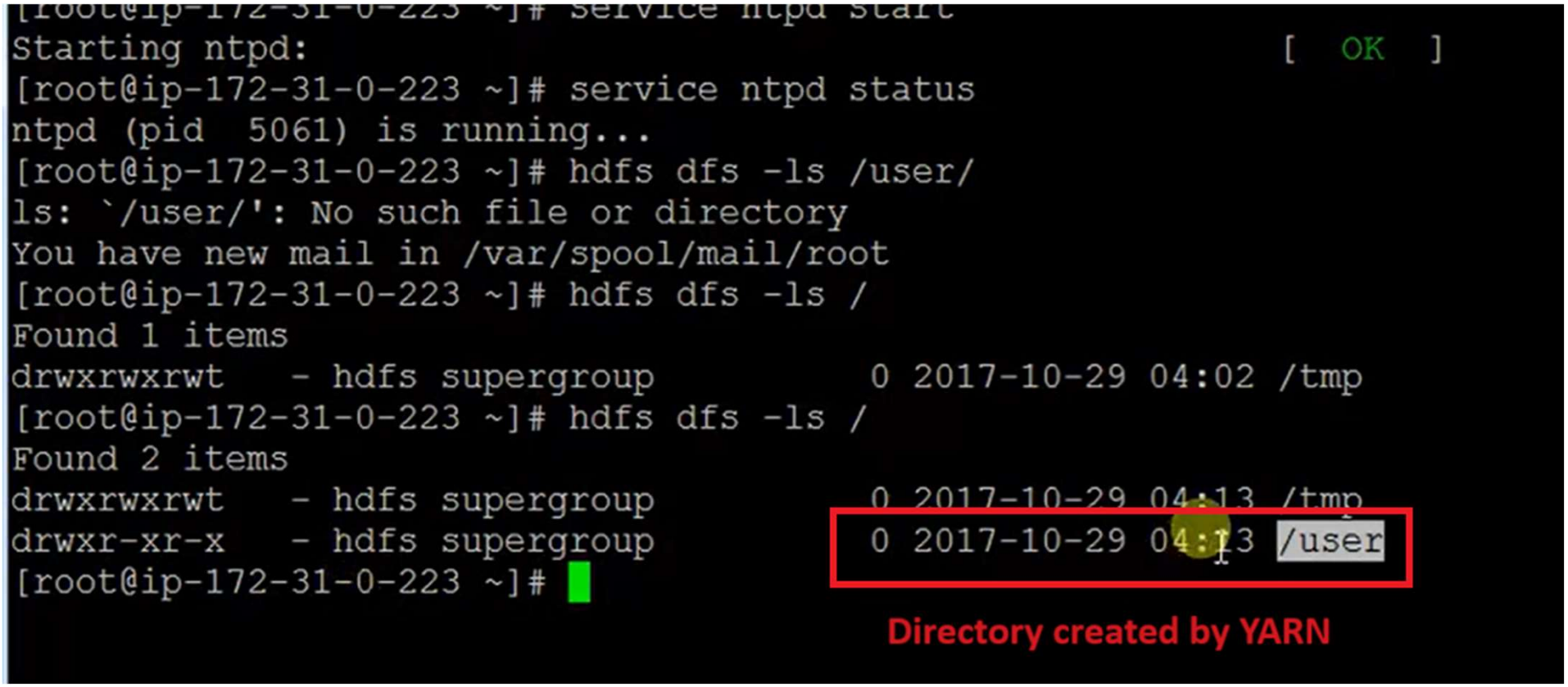
Default config



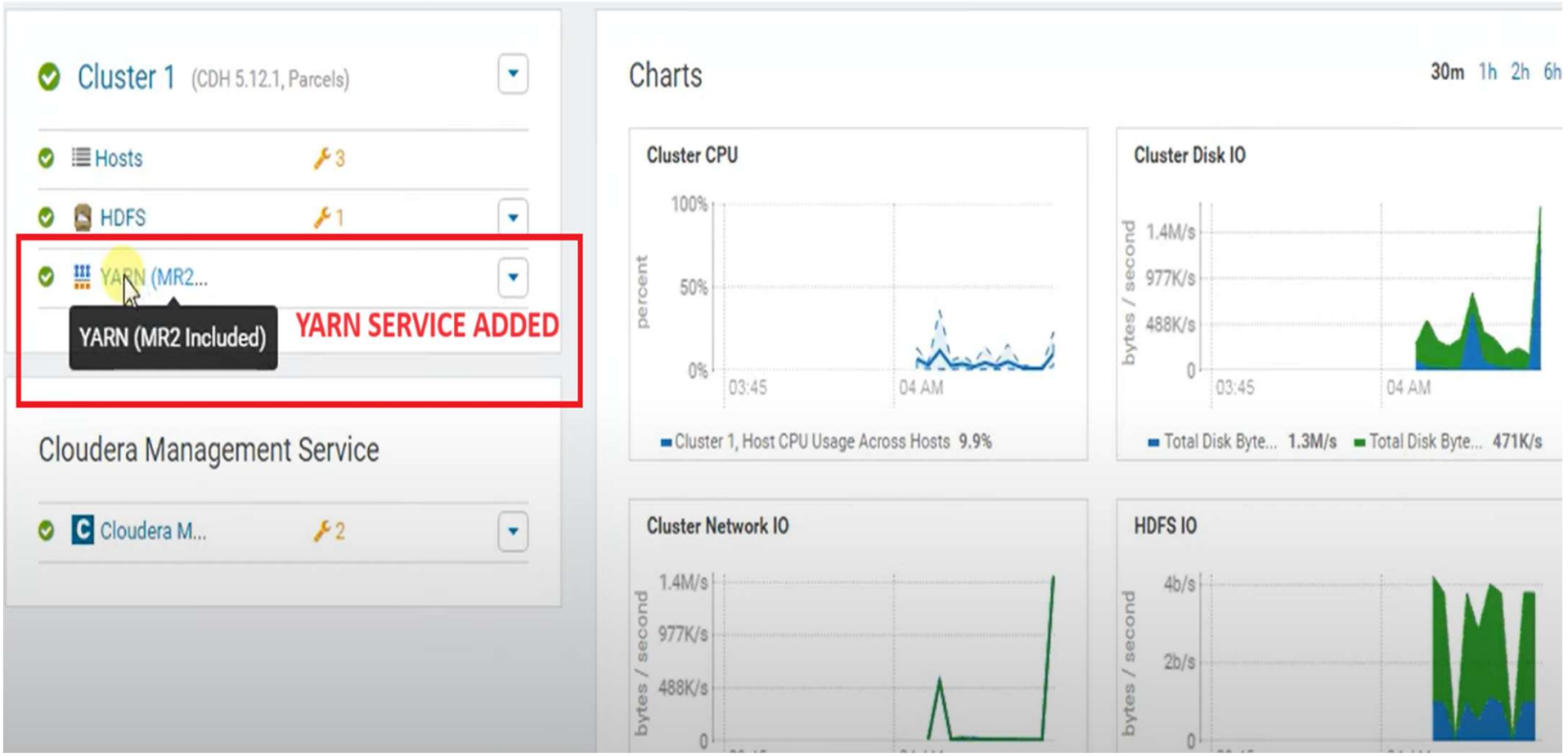
Adding Services

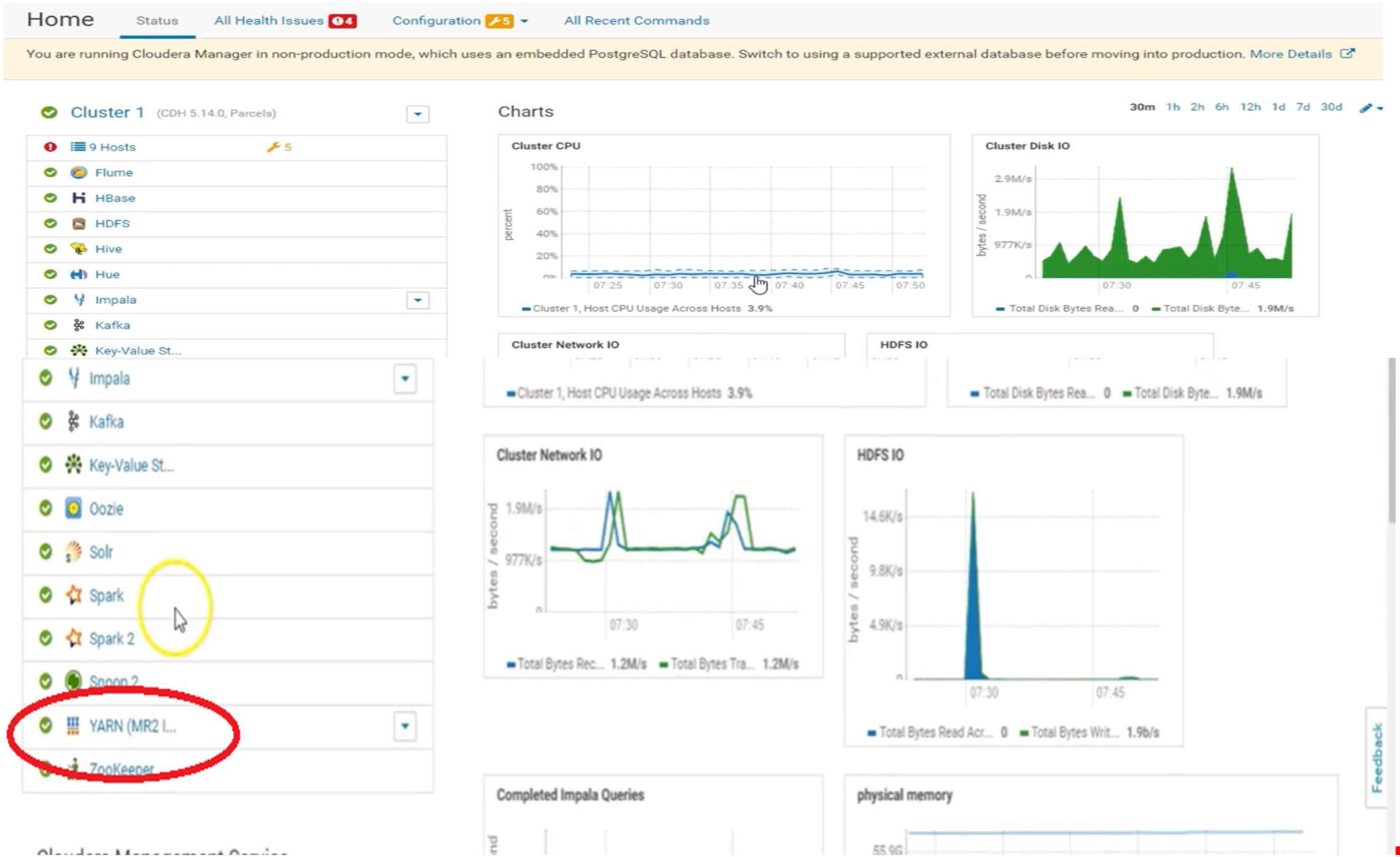


Contd..

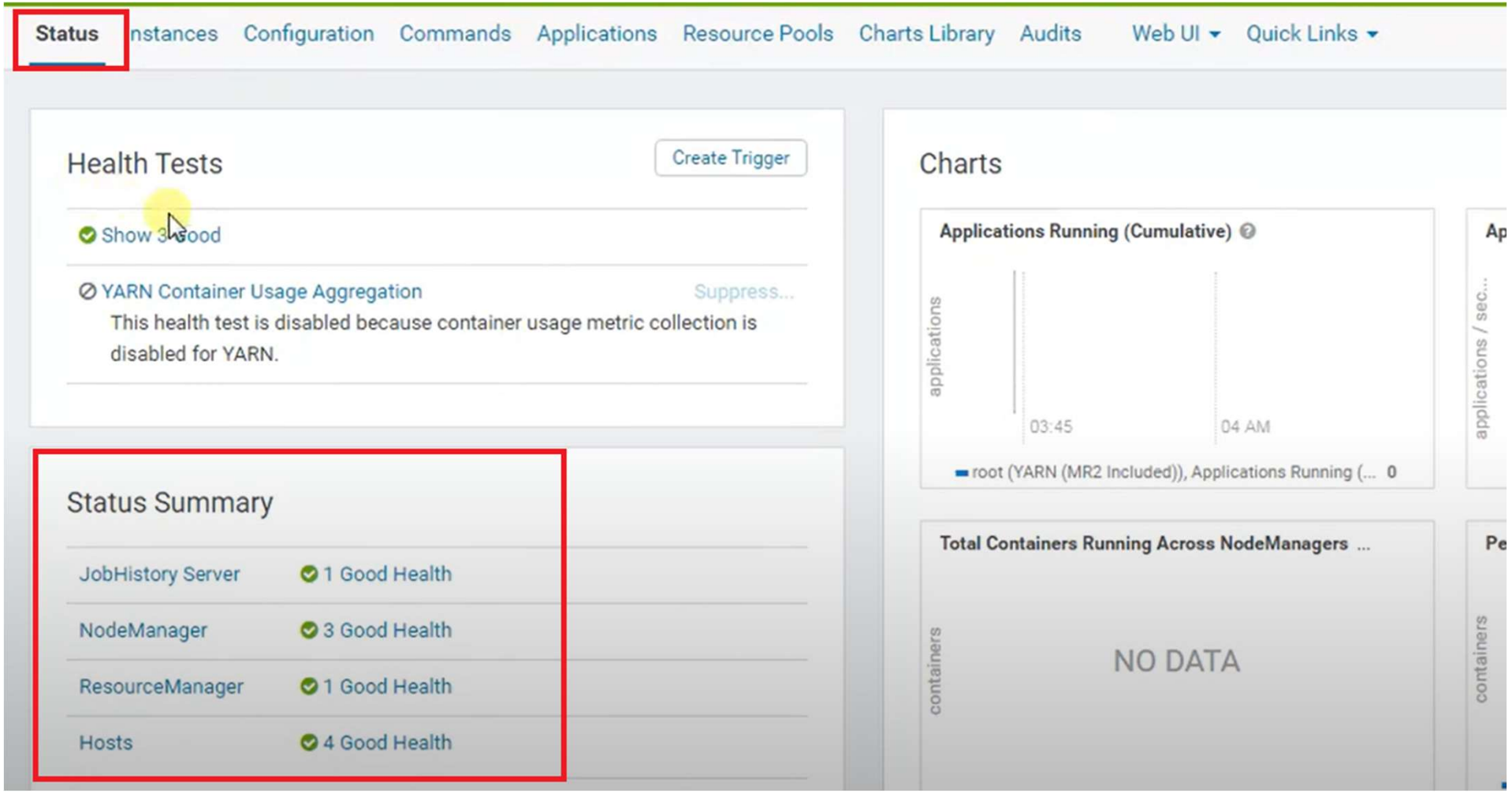

Starting ntpd: [OK]
[root@ip-172-31-0-223 ~]# service ntpd status
ntpd (pid 5061) is running...
[root@ip-172-31-0-223 ~]# hdfs dfs -ls /user/
ls: `/user/': No such file or directory
You have new mail in /var/spool/mail/root
[root@ip-172-31-0-223 ~]# hdfs dfs -ls /
Found 1 items
drwxrwxrwt - hdfs supergroup 0 2017-10-29 04:02 /tmp
[root@ip-172-31-0-223 ~]# hdfs dfs -ls /
Found 2 items
drwxrwxrwt - hdfs supergroup 0 2017-10-29 04:13 /tmp
drwxr-xr-x - hdfs supergroup 0 2017-10-29 04:13 /user
[root@ip-172-31-0-223 ~]#
Directory created by YARN

Service added Successful y

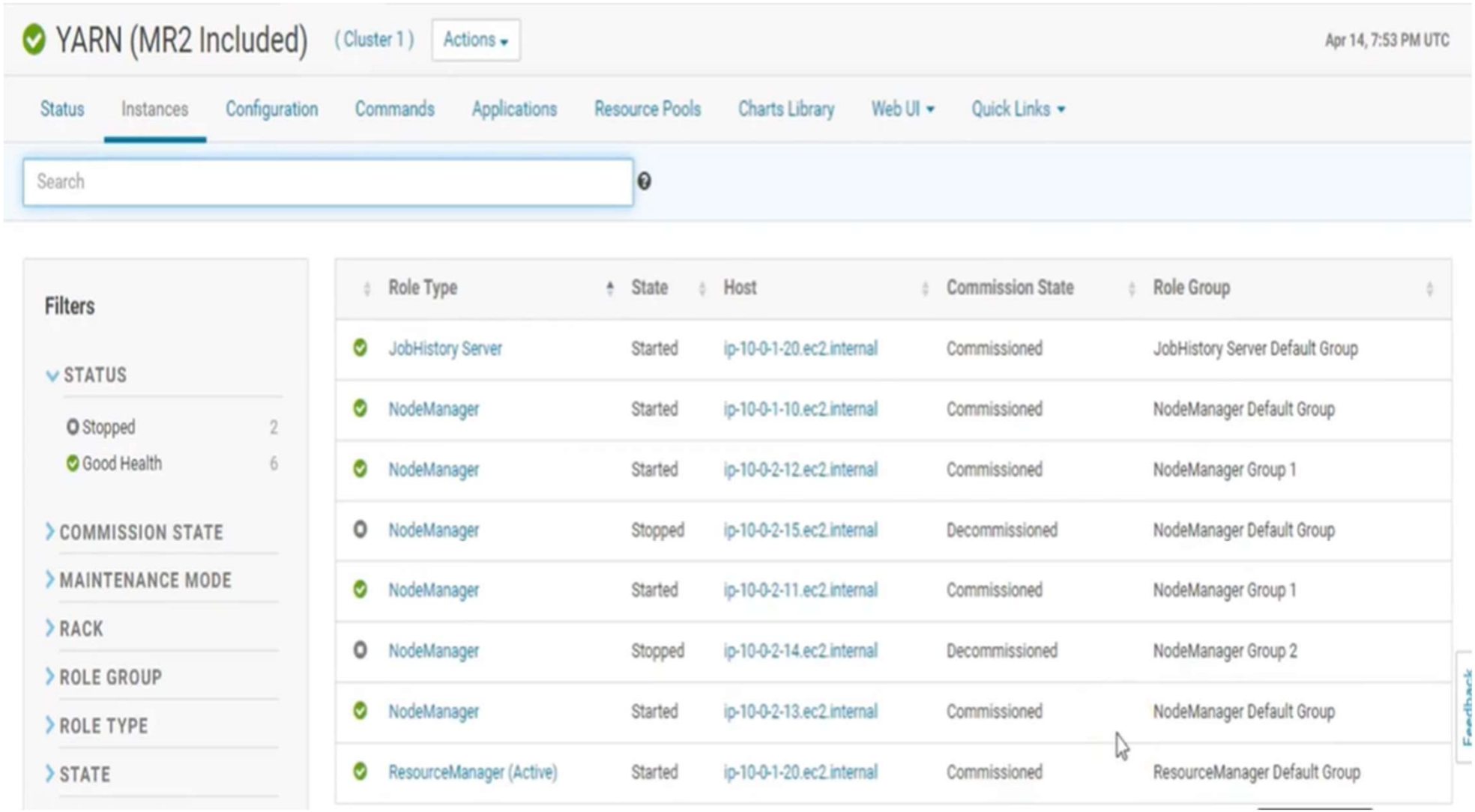



Home
Status
All Health Issues 04
Configuration 5
All Recent Commands
You are running Cloudera Manager in non-production mode, which uses an embedded PostgreSQL database. Switch to using a supported external database before moving into production. More Details
Cluster 1 (CDH 5.14.0, Parcels)
9 Hosts
Flume
HBase
HDFS
Hive
Hue
Impala
Kafka
Key-Value St...
Oozie
Solr
Spark
Spark 2
Spoon 2
YARN (MR2 I...
ZooKeeper
Charts
30m 1h 2h 6h 12h 1d 7d 30d
Cluster CPU
Cluster 1, Host CPU Usage Across Hosts 3.9%
Cluster Disk IO
Total Disk Bytes Rea... 0
Total Disk Byte... 1.9M/s
Cluster Network IO
Total Bytes Rec... 1.2M/s
Total Bytes Tra... 1.2M/s
HDFS IO
Total Bytes Read Acr... 0
Total Bytes Writ... 1.9b/s
Completed Impala Queries
physical memory
Feedback

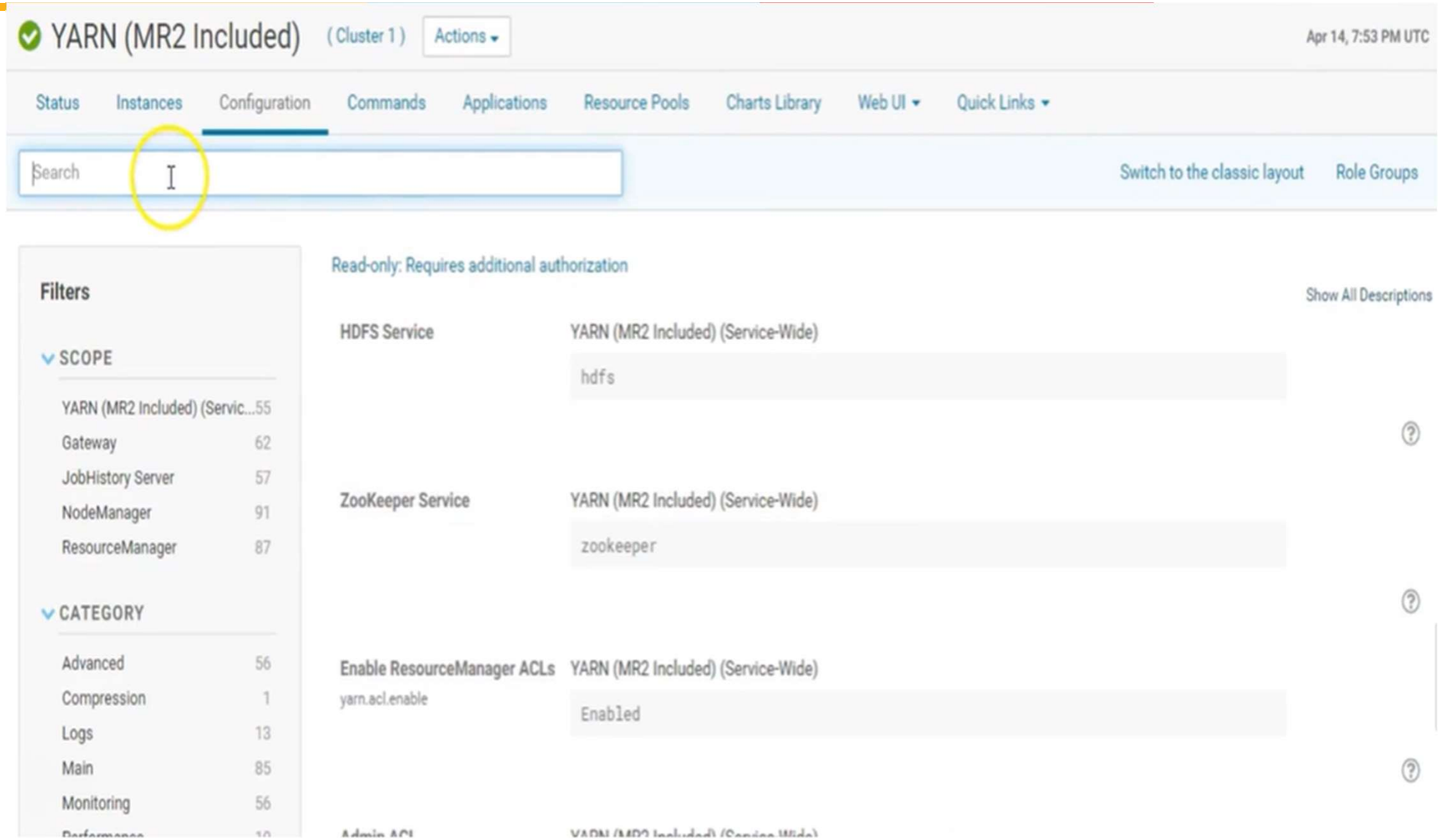
Status page


Status
nstances
Configuration
Commands
Applications
Resource Pools
Charts Library
Audits
Web UI
Quick Links
Health Tests
Create Trigger
Show 3 Good
YARN Container Usage Aggregation
Suppress...
This health test is disabled because container usage metric collection is disabled for YARN.
Status Summary
JobHistory Server
1 Good Health
NodeManager
3 Good Health
ResourceManager
1 Good Health
Hosts
4 Good Health
Charts
Applications Running (Cumulative)
applications
03:45
04 AM
root (YARN (MR2 Included)), Applications Running (... 0
Total Containers Running Across NodeManagers ...
containers
NO DATA

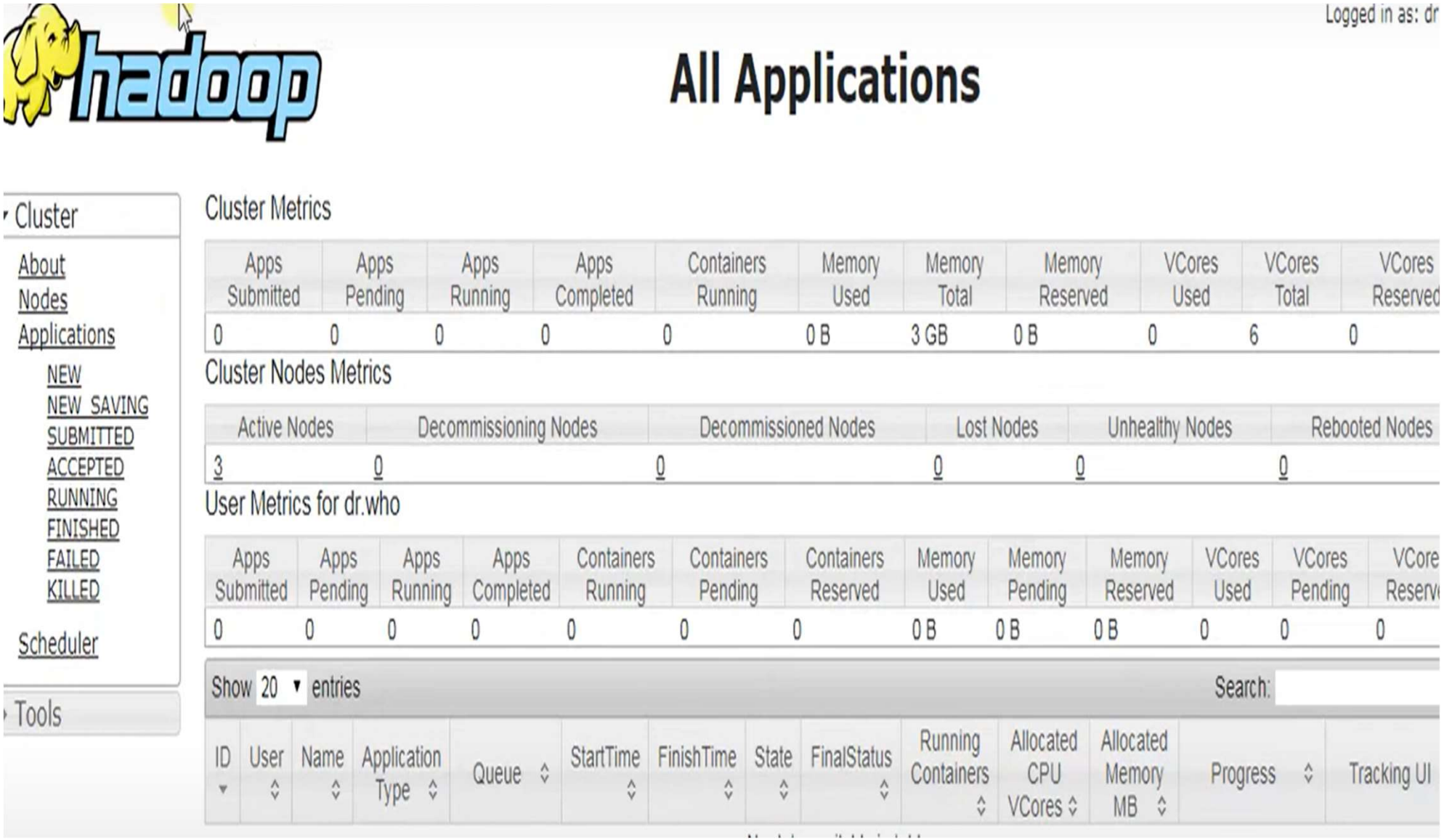
Instances/Roles



Configuration

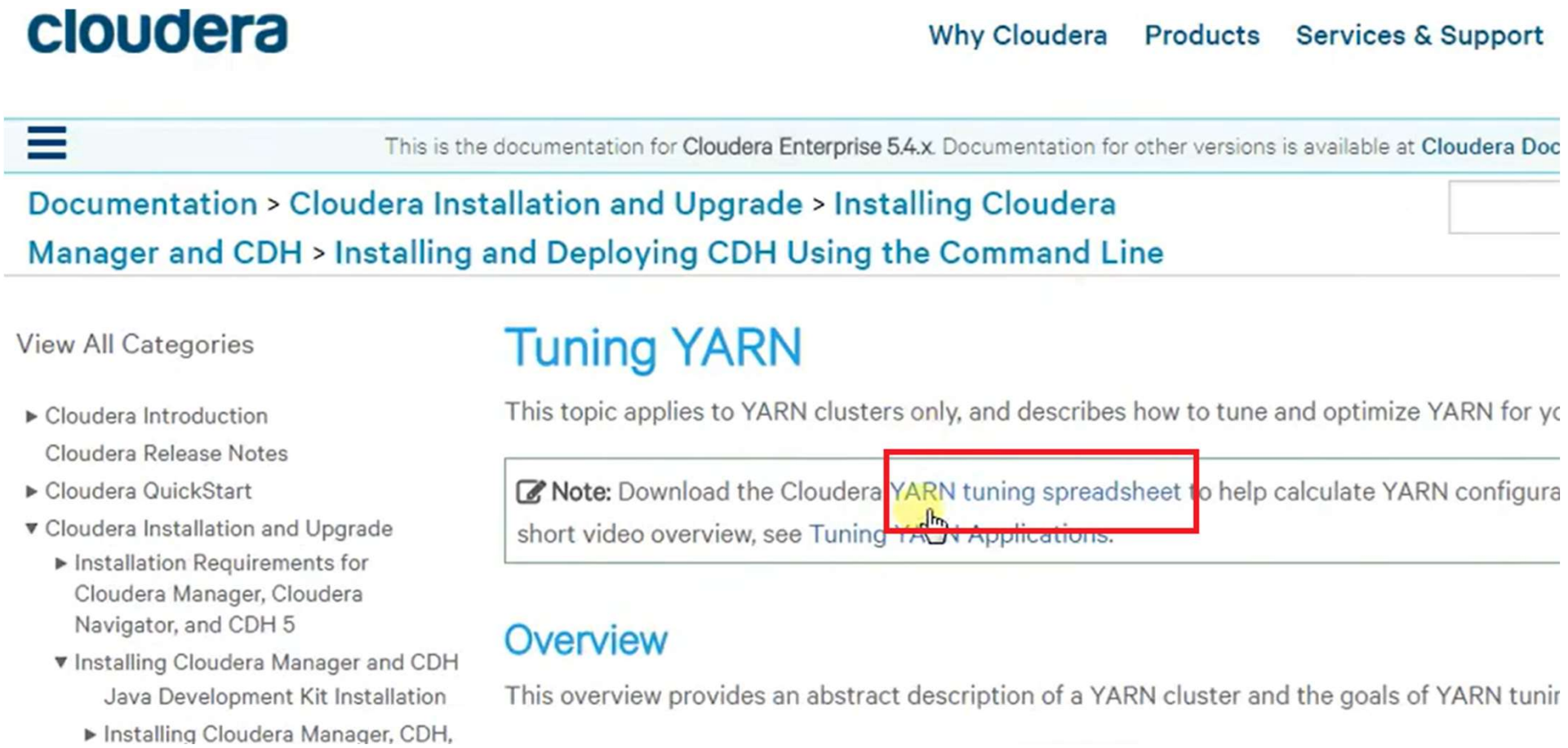


Resource Manager UI



Job History UI

YARN Tunning



Machine Configuration

STEP 1: Worker Host Configuration

Enter your likely machine configuration in the input boxes below. If you are uncertain what machines you plan on buying, put in some minimum values t will suit what you expect to buy.

Host Components	Quantity	Description
RAM	8	Gigabytes
CPU	2	CPUs: 6 cores, 3.5 GHz, 15MB cache
HDD (Hard Disk Drive)	36	12x3TB SATA III Hard Drives in JBOD Configurat
Ethernet	2	Gigabit Ethernet

STEP 2: Worker Host Planning

Cluster Configuration | YARN Configuration | MapReduce Configuration

to allocate resources, mainly CPU and memory, to the various software components that run on the host.

Service	Category	CPU (cores	Memory (M	Notes
Operating System	Overhead	1	1024	Most operating systems use 4-8GB minimum.
Task overhead	Overhead	0	0	Allow additional memory overhead for task buffers such as
Cloudera Manager agent	Overhead	0	1024	Allocate 1GB for Cloudera Manager agents, which track res
Other services	Overhead	0	0	Enter the required cores or memory for services not listed a
HDFS DataNode	CDH	0	1024	Allocate 1GB for the HDFS DataNode.
Impala daemon	CDH	0	0	(Optional Service) Suggestion: Allocate at least 16GB mem
Hbase RegionServer	CDH	0	0	(Optional Service) Suggestion: Allocate no more than 12-16
Solr Server	CDH	0	0	(Optional Service) Suggestion: Minimum 1GB for Solr serve
YARN NodeManager	CDH	0	1024	Allocate 1GB for the YARN NodeManager.
Available Resources		1	4096	
Physical Cores to Vcores Multiplier		4		Set this ratio based on the expected number of concurrent
YARN Available Vcores		4		This value will be used in STEP 4 for YARN Configuration
YARN Available Memory			4096	This value will be used in STEP 4 for YARN Configuration

STEP 3: Cluster Size

Cluster Configuration | YARN Configuration | MapReduce Configuration

Details of worker nodes

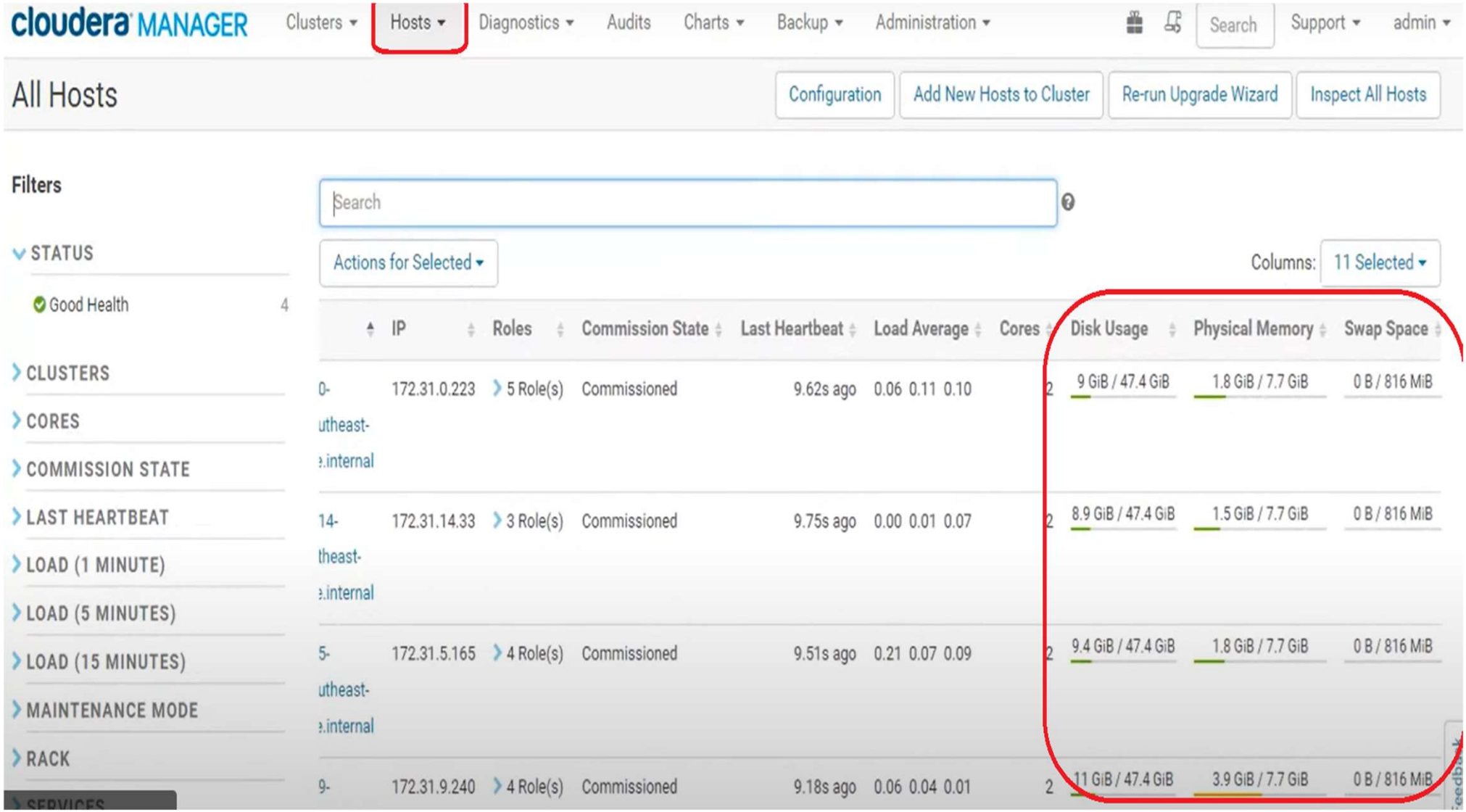
Hbase RegionServer	CDH	0	0	(Optional Service) Suggestion:
Solr Server	CDH	0	0	(Optional Service) Suggestion:
YARN NodeManager	CDH	0	1024	Allocate 1GB for the YARN Nod
Available Resources		1	4096	
Physical Cores to Vcores Multiplier		4		Set this ratio based on the exp
YARN Available Vcores		4		This value will be used in STEP
YARN Available Memory			4096	This value will be used in STEP

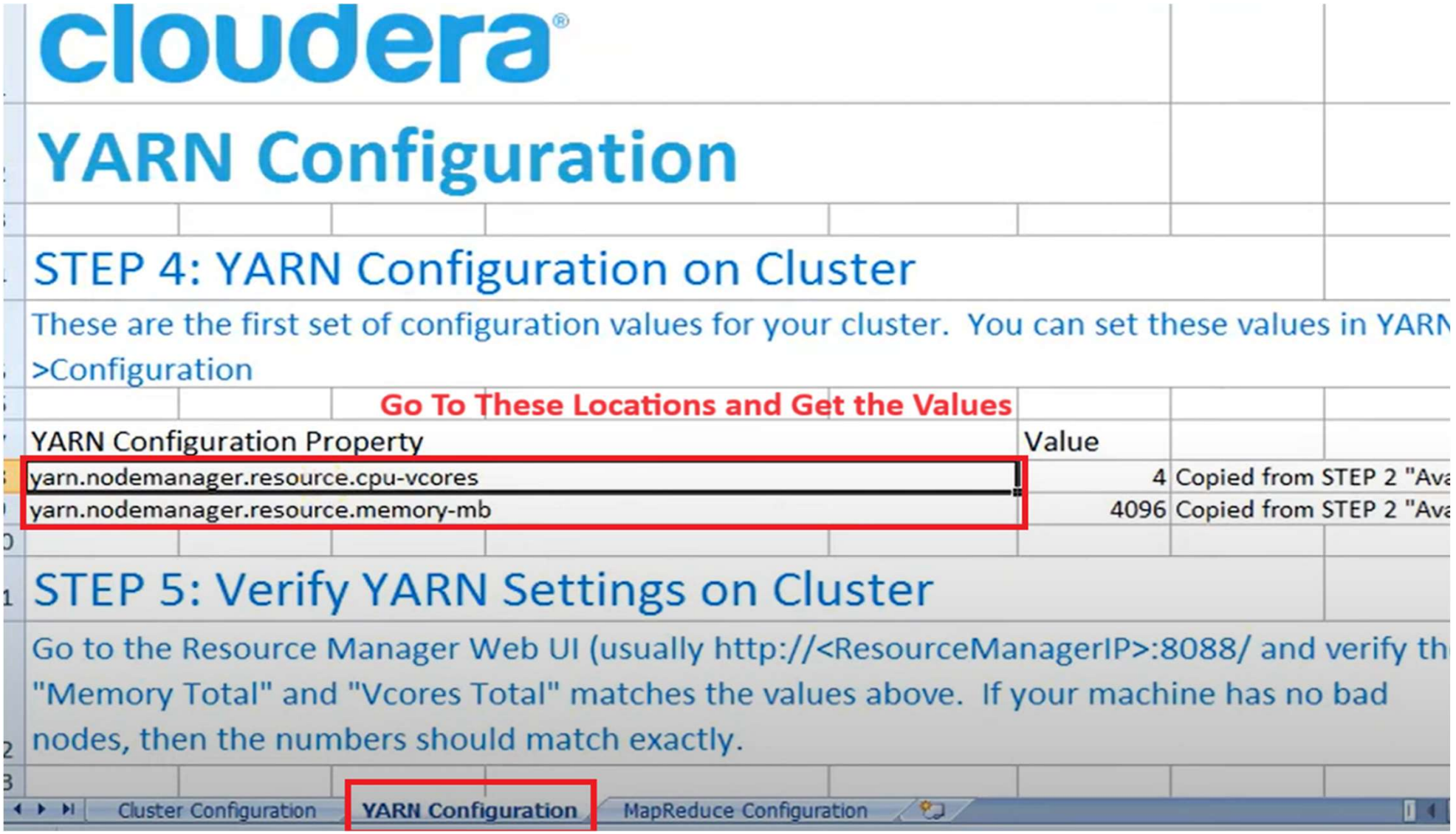
STEP 3: Cluster Size

Enter the number of nodes you have (or expect to have) in the cluster

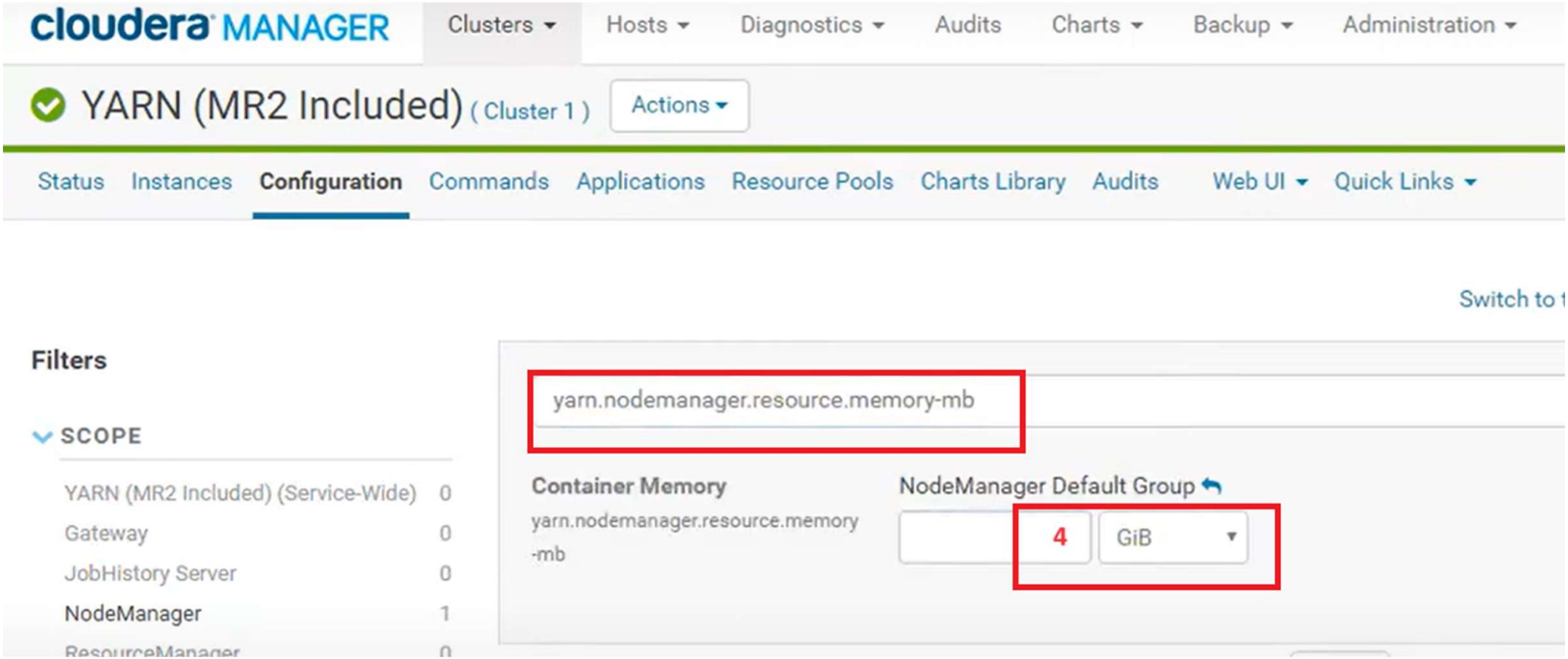
	Quantity
Number of Worker Hosts in the cluster	3

Make note of node statistics

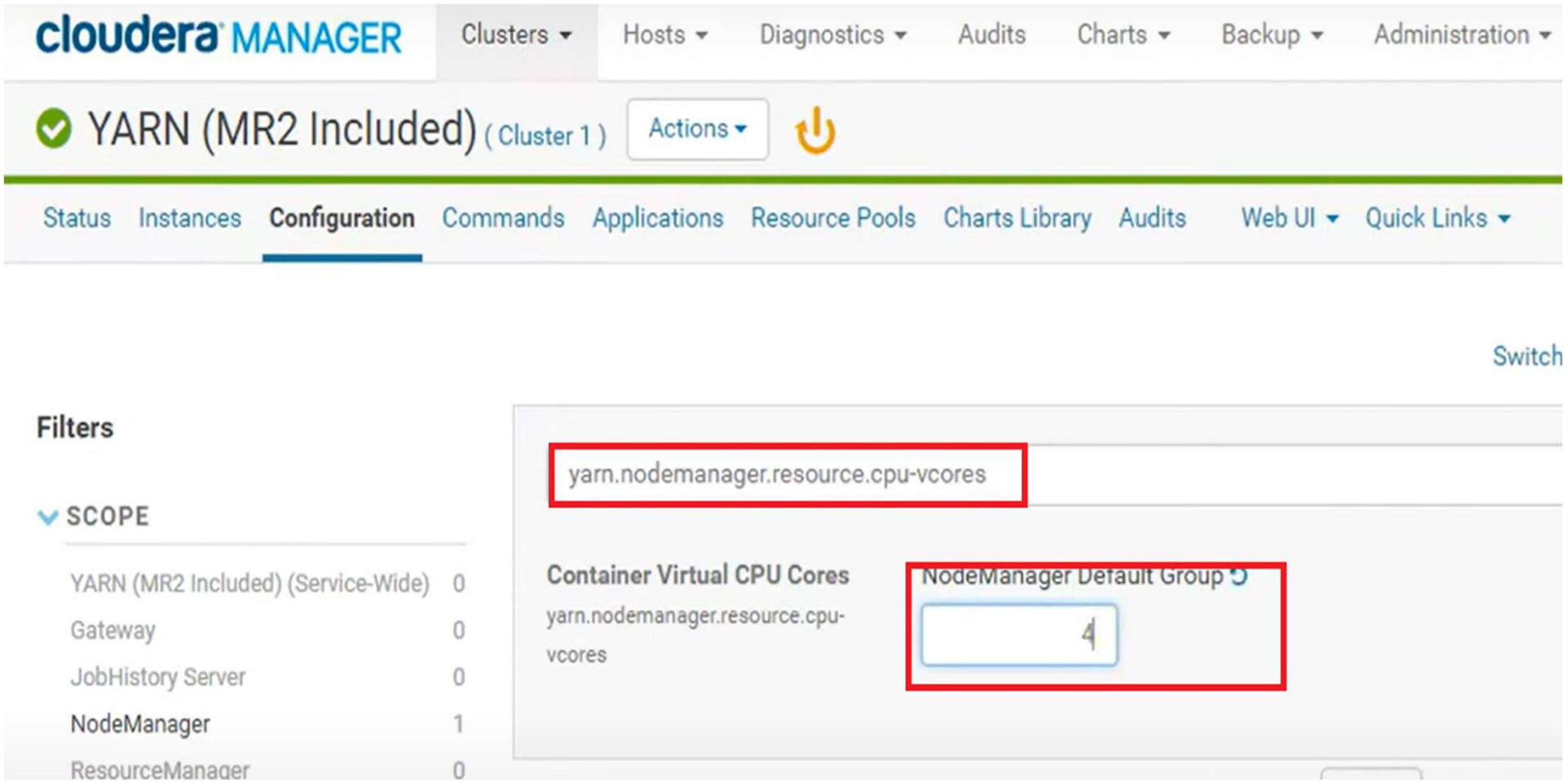



cloudera
YARN Configuration
STEP 4: YARN Configuration on Cluster
These are the first set of configuration values for your cluster. You can set these values in YARN
>Configuration
Go To These Locations and Get the Values
YARN Configuration Property
Value
yarn.nodemanager.resource.cpu-vcores
4 Copied from STEP 2 "Ava
yarn.nodemanager.resource.memory-mb
4096 Copied from STEP 2 "Ava
STEP 5: Verify YARN Settings on Cluster
Go to the Resource Manager Web UI (usually http://<ResourceManagerIP>:8088/ and verify th
"Memory Total" and "Vcores Total" matches the values above. If your machine has no bad
nodes, then the numbers should match exactly.
Cluster Configuration
YARN Configuration
MapReduce Configuration

Memory Available update



CPU Cores Available Update



Few More Details

STEP 5: Verify YARN Settings on Cluster

Go to the Resource Manager Web UI (usually http://<ResourceManagerIP>:8088/ and verify the "Memory Total" and "Vcores Total" matches the values above. If your machine has no bad nodes, then the numbers should match exactly.

Resource Manager Property to Check	Value	Note
Expected Value for "Vcores Total"	12	Calculated from STEP 2 "YARN Availab
Expected Value for "Memory Total" (in GB)	12	Calculated from STEP 2 "YARN Availab

STEP 6: Verify Container Settings on Cluster

In order to have YARN jobs run cleanly, you need to configure the container properties.

YARN Container Configuration Property (Vcores)	Value	Description
yarn.scheduler.minimum-allocation-vcores	1	Minimum vcore reservation for a cont
yarn.scheduler.maximum-allocation-vcores	1	Maximum vcore reservation for a cont
yarn.scheduler.increment-allocation-vcores	1	Vcore allocations must be a multiple
YARN Container Configuration Property (Memory)	Value	

Scheduler Info

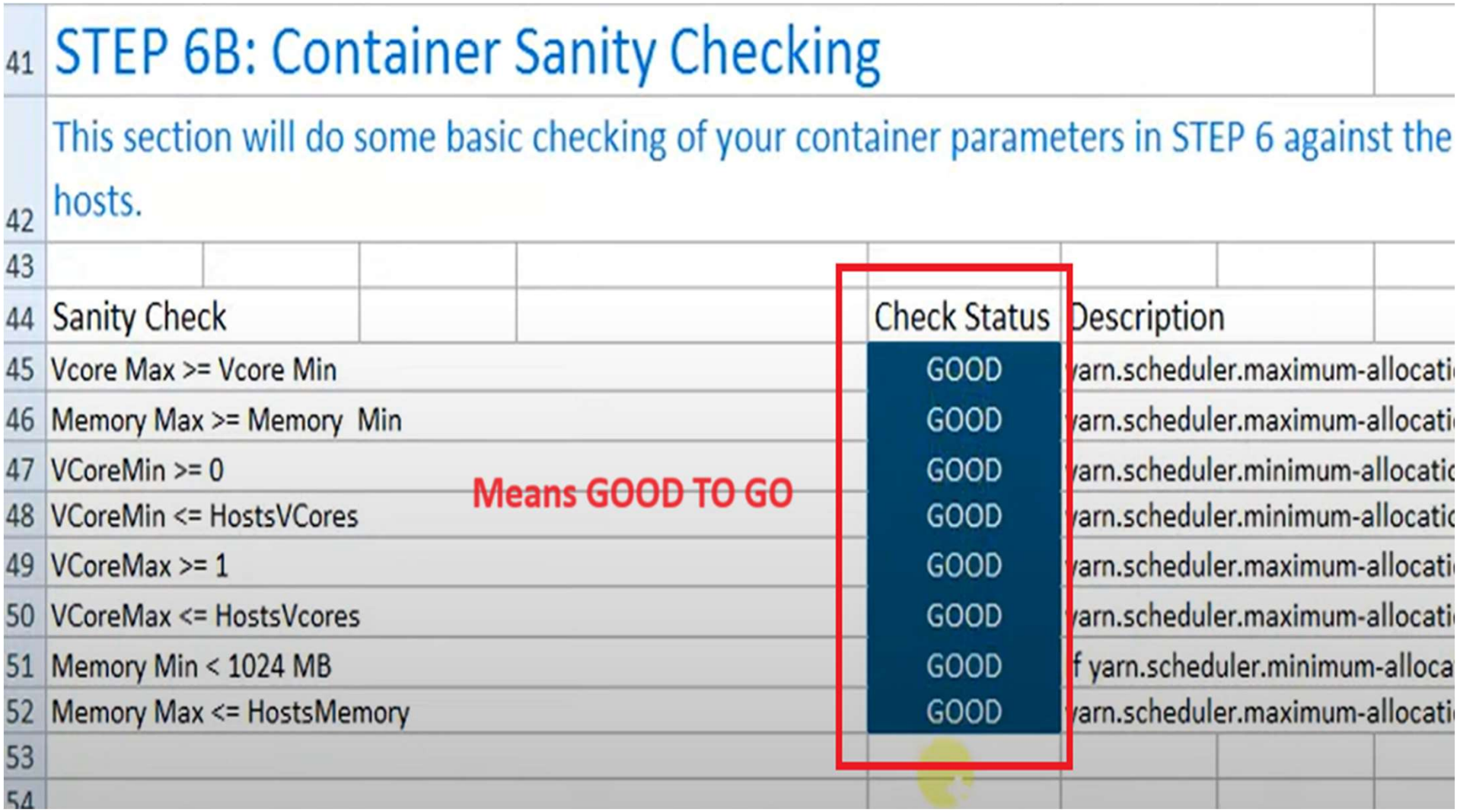
STEP 6: Verify Container Settings on Cluster

In order to have YARN jobs run cleanly, you need to configure the container properties.

YARN Container Configuration Property (Vcores)	Value	Description
yarn.scheduler.minimum-allocation-vcores	1	Minimum vcore reservation for a contai
yarn-scheduler.maximum-allocation-vcores	1	Maximum vcore reservation for a conta
yarn.scheduler.increment-allocation-vcores	1	Vcore allocations must be a multiple of

YARN Container Configuration Property (Memory)	Value	
yarn.scheduler.minimum-allocation-mb	1024	Minimum memory reservation for a con
yarn.scheduler.maximum-allocation-mb	8192	Maximum memory reservation for a cor
yarn.scheduler.increment-allocation-mb	512	Memory allocations must be a multiple

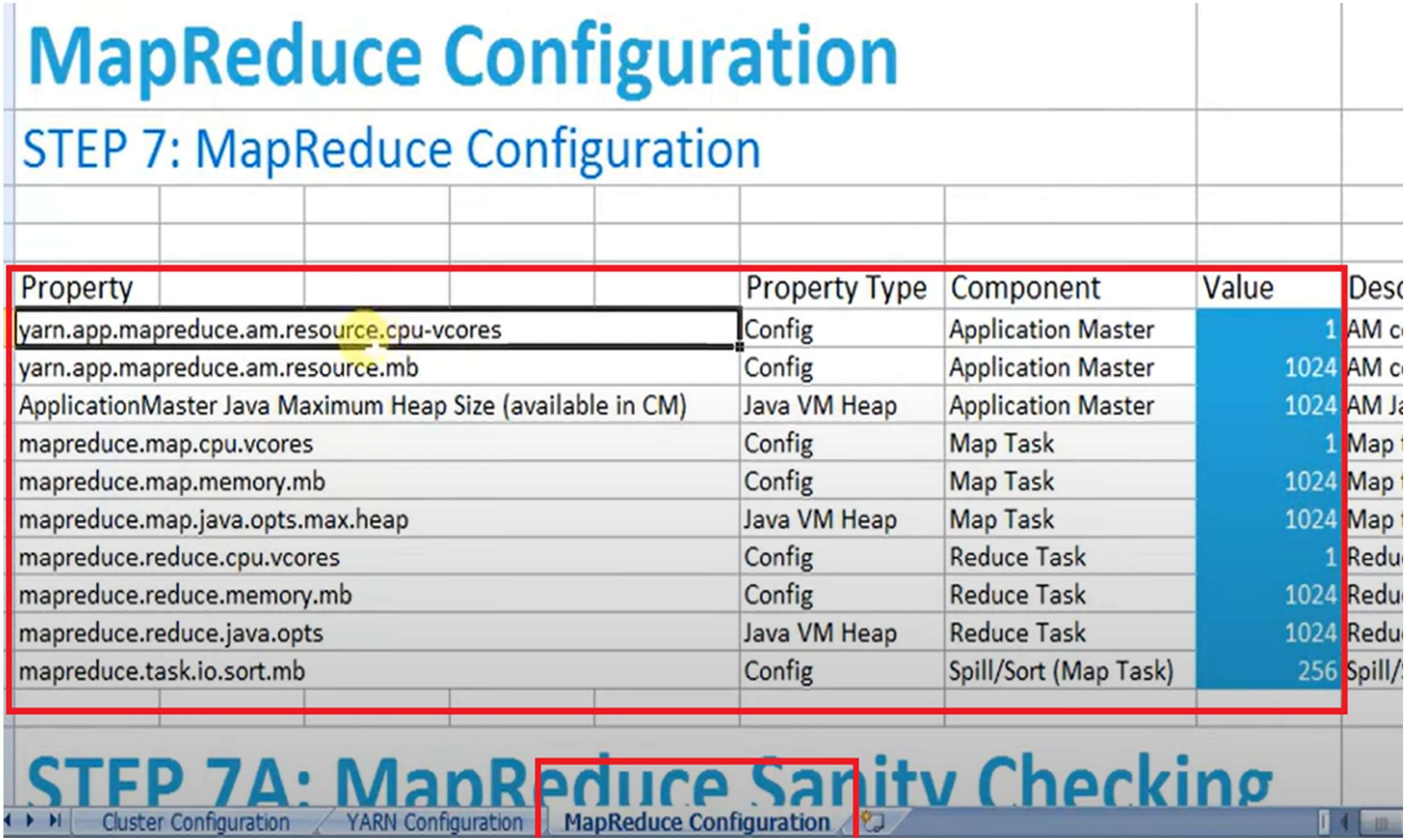
Check



	STEP 6B: Container Sanity Checking		
	This section will do some basic checking of your container parameters in STEP 6 against the hosts.		
	Sanity Check	Check Status	Description
	Vcore Max >= Vcore Min	GOOD	yarn.scheduler.maximum-allocati
	Memory Max >= Memory Min	GOOD	yarn.scheduler.maximum-allocati
	VCoreMin >= 0	GOOD	yarn.scheduler.minimum-allocatio
	VCoreMin <= HostsVCores	GOOD	yarn.scheduler.minimum-allocatio
	VCoreMax >= 1	GOOD	yarn.scheduler.maximum-allocati
	VCoreMax <= HostsVcores	GOOD	yarn.scheduler.maximum-allocati
	Memory Min < 1024 MB	GOOD	f yarn.scheduler.minimum-alloca
	Memory Max <= HostsMemory	GOOD	yarn.scheduler.maximum-allocati

Means GOOD TO GO

Map-Reduce Config



MapReduce Configuration

STEP 7: MapReduce Configuration

Property	Property Type	Component	Value	Desc
yarn.app.mapreduce.am.resource.cpu-vcores	Config	Application Master	1	AM c
yarn.app.mapreduce.am.resource.mb	Config	Application Master	1024	AM c
ApplicationMaster Java Maximum Heap Size (available in CM)	Java VM Heap	Application Master	1024	AM Ja
mapreduce.map.cpu.vcores	Config	Map Task	1	Map
mapreduce.map.memory.mb	Config	Map Task	1024	Map
mapreduce.map.java.opts.max.heap	Java VM Heap	Map Task	1024	Map
mapreduce.reduce.cpu.vcores	Config	Reduce Task	1	Redu
mapreduce.reduce.memory.mb	Config	Reduce Task	1024	Redu
mapreduce.reduce.java.opts	Java VM Heap	Reduce Task	1024	Redu
mapreduce.task.io.sort.mb	Config	Spill/Sort (Map Task)	256	Spill/

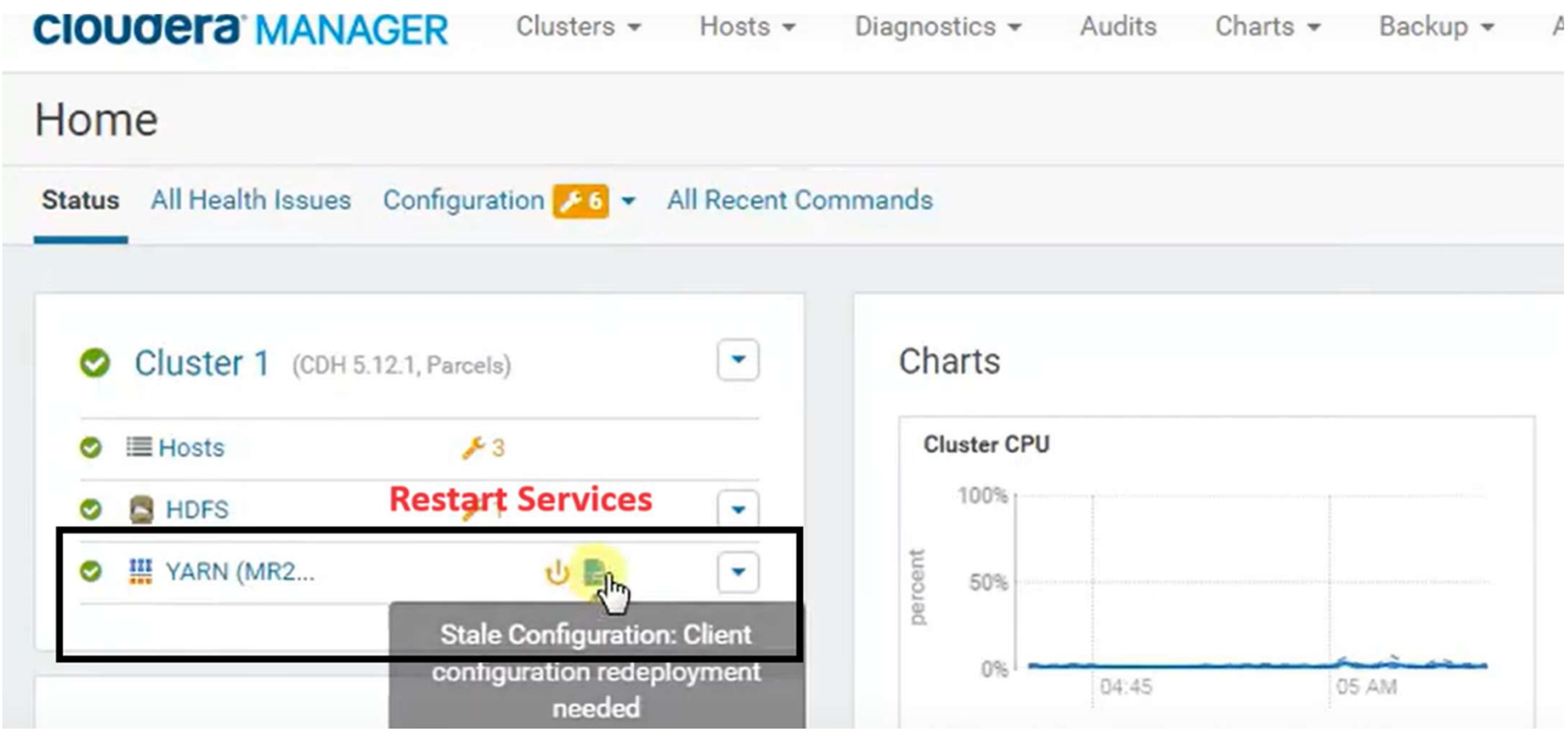
STEP 7A: MapReduce Sanity Checking

Cluster Configuration | YARN Configuration | MapReduce Configuration

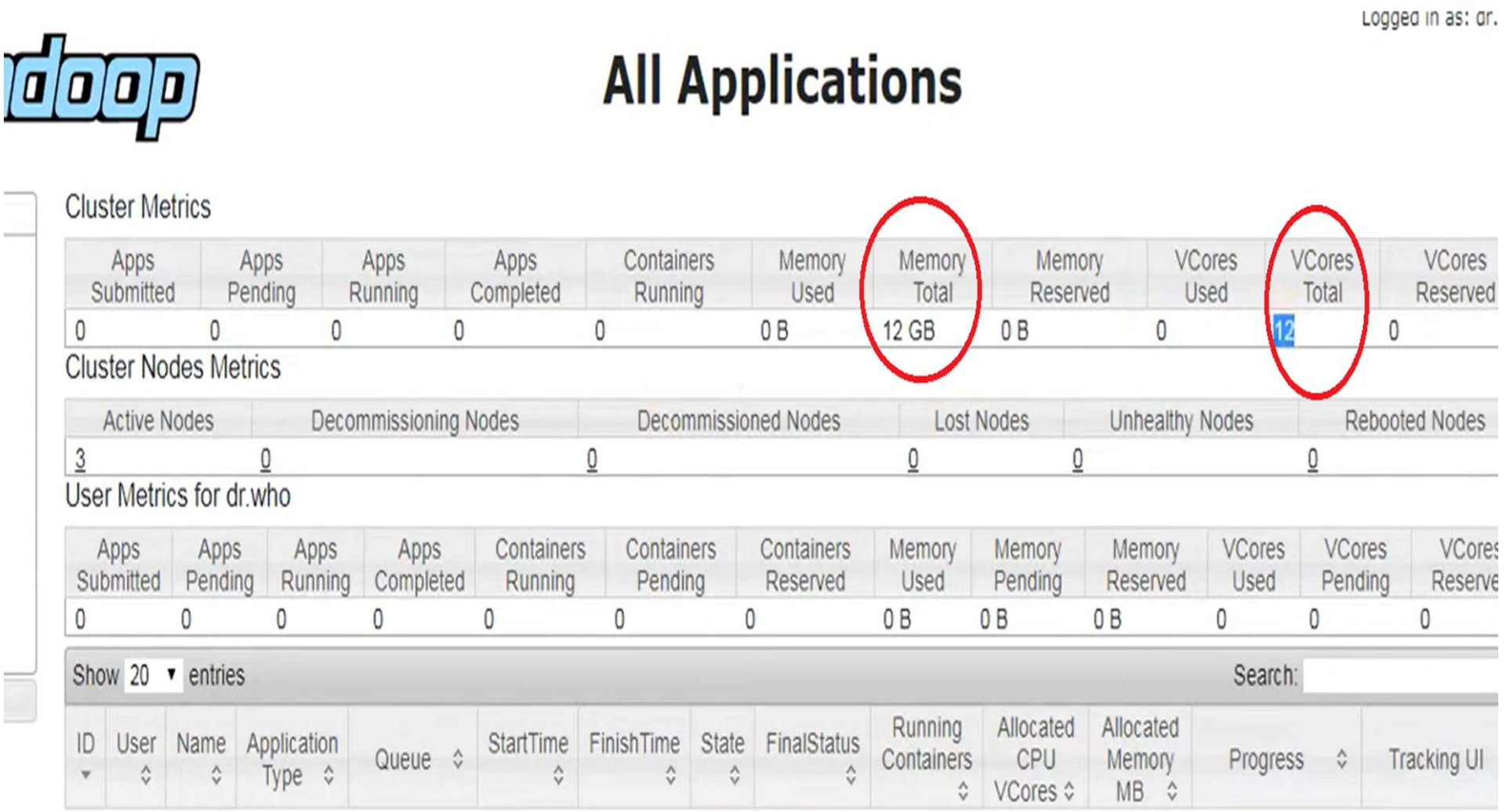
Sanity check MapReduce settings against container minimum/maximum properties.

Application Master Sanity Checks	Value	Description
yarn.app.mapreduce.am.resource.cpu-vcores >= container min	GOOD	Make sure ApplicationMaster vcore request fits within container limits
yarn.app.mapreduce.am.resource.cpu-vcores <= container max	GOOD	Ditto
yarn.app.mapreduce.am.resource.mb >= container min	GOOD	Make sure ApplicationMaster memory request fits within container lim
yarn.app.mapreduce.am.resource.mb <= container max	GOOD	Ditto
ApplicationMaster Java Heap "close" to memory request	GOOD	Make sure ApplicationMaster Java Heap is within 90% to 100% of yarn
Map Task Sanity Checks	**Value**	**Description**
mapreduce.map.cpu.vcores >= container min	GOOD	Make sure Map Task vcore request fits within container limits
mapreduce.map.cpu.vcores <= container max	GOOD	Ditto
mapreduce.map.cpu.memory.mb >= container min	GOOD	Make sure Map Task memory request fits within container limits
mapreduce.map.cpu.memory.mb <= container max	GOOD	Ditto
Map Task Java Heap "close" to memory request	GOOD	Make sure that Map Task Java Heap is within 90% to 100% of mapredu
mapreduce.task.io.sort.mb << Map Task Java Heap	GOOD	Make sure that Spill/Sort memory reservation leaves enough "room" in
Reduce Task Sanity Checks	**Value**	**Description**
mapreduce.reduce.cpu.vcores >= container min	GOOD	Make sure Reduce Task vcore request fits within container limits
mapreduce.reduce.cpu.vcores <= container max	GOOD	Ditto
mapreduce.reduce.cpu.memory.mb >= container min	GOOD	Make sure Reduce Task memory request fits within container limits
mapreduce.reduce.cpu.memory.mb <= container max	GOOD	Ditto
Reduce Task Java Heap "close" to memory request	GOOD	Make sure that Reduce Task Java Heap is within 90% to 100% of mapre

Restart Services



Verify



Logged in as: dr.

doop

All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved
0	0	0	0	0	0 B	12 GB	0 B	0	12	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
3	0	0	0	0	0

User Metrics for dr.who

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending	Memory Reserved	VCores Used	VCores Pending	VCores Reserve
0	0	0	0	0	0	0	0 B	0 B	0 B	0	0	0

Show 20 ▾ entries Search:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Progress	Tracking UI