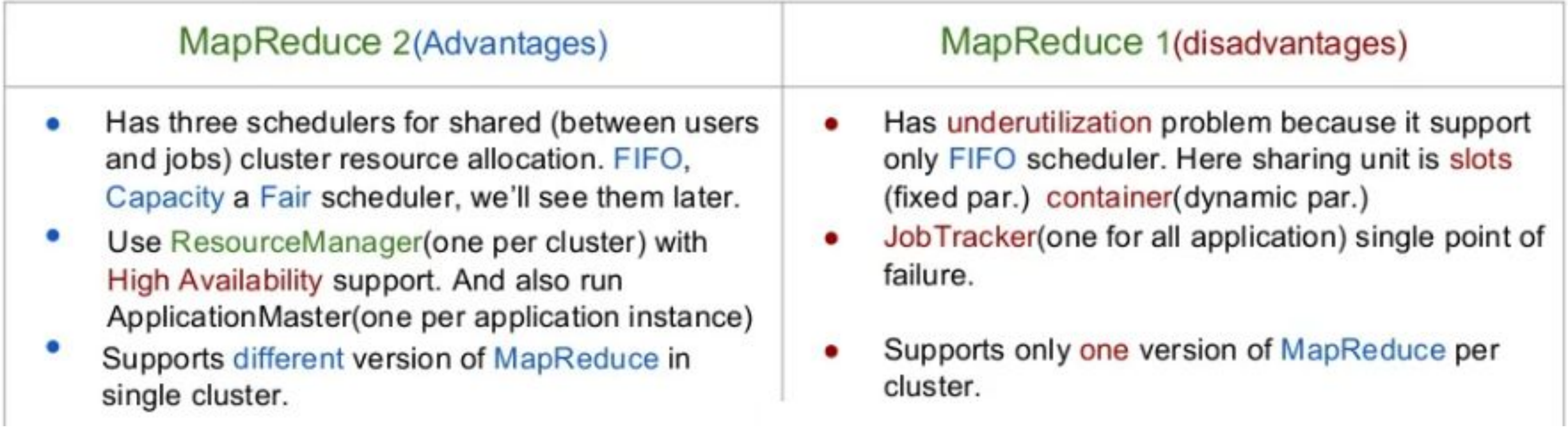
YARN

Yet Another Resource Negotiator

MapReduce 2 vs MapReduce 1

MapReduce 2 became an application on top of the YARN, which use Yarn to manage resources. We look at advantages of MapReduce 2 in table below:



MapReduce 2(Advantages)	MapReduce 1(disadvantages)
● Has three schedulers for shared (between users and jobs) cluster resource allocation. FIFO, Capacity a Fair scheduler, we'll see them later. ● Use ResourceManager(one per cluster) with High Availability support. And also run ApplicationMaster(one per application instance) ● Supports different version of MapReduce in single cluster.	● Has underutilization problem because it support only FIFO scheduler. Here sharing unit is slots (fixed par.) container(dynamic par.) ● JobTracker(one for all application) single point of failure. ● Supports only one version of MapReduce per cluster.

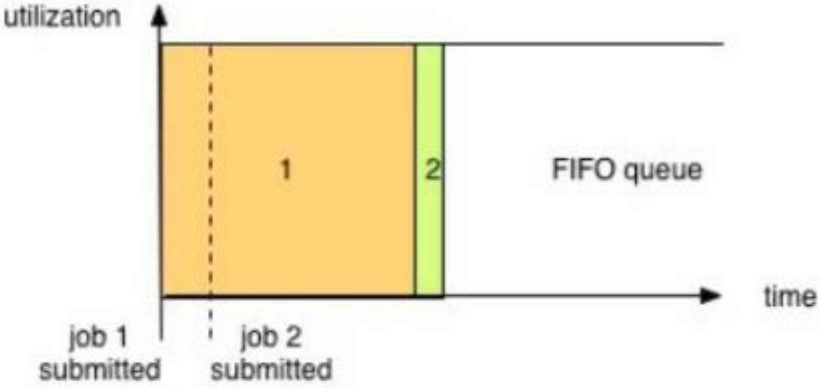
Scheduling in Yarn

- Scheduling is an important task when Hadoop cluster is shared between many users and tasks. Problem is which user's task should be run first or which task should be run first, big one or small one.
- Hadoop 1 which use FIFO scheduler with slot (fixed cpu, memory and disk count) based model was very inefficient for shared clusters, whereas Hadoop 2 introduced Capacity (by Yahoo) and Fair (by Facebook) schedulers with container (dynamic cpu, memory and disk count) allocation model as part of the Yarn, which is more efficient than FIFO scheduler.

Scheduling in Yarn

FIFO - First Input First Output:

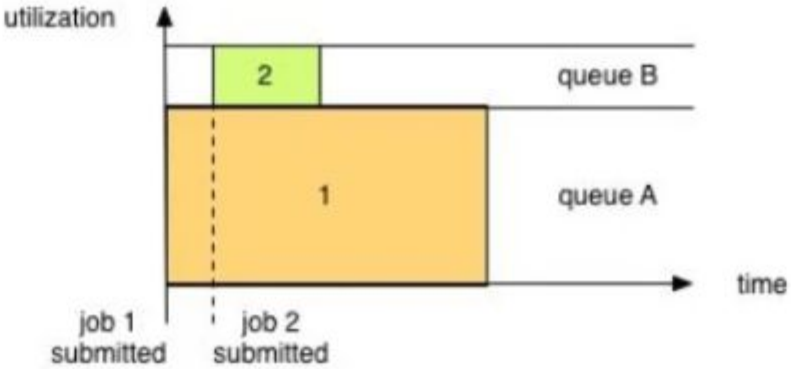
- the simplest and most understandable scheduler
- It doesn’t needing any configuration.
- But it’s not suitable for shared clusters because big applications eat all resources.



Scheduling in Yarn

Capacity scheduler - allows sharing of a Hadoop cluster along organizational lines (each one is a queue). Queues may be further divided in hierarchical fashion.

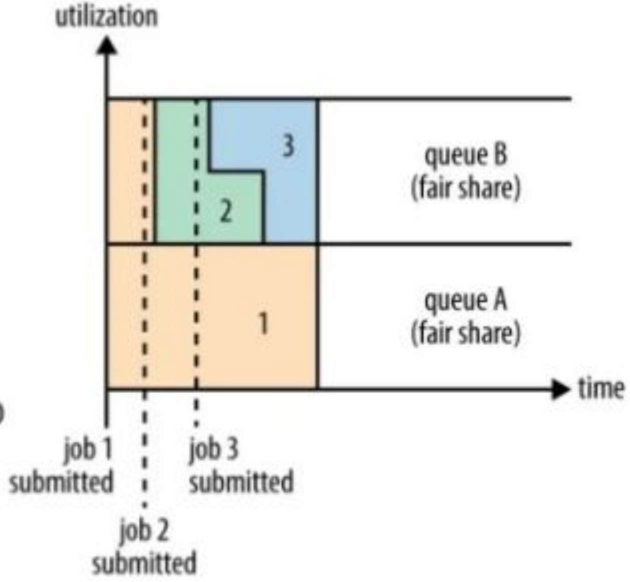
- each organization is allocated a certain capacity of the overall cluster.
- if there is more than one job Capacity Scheduler may allocate the spare resources to jobs in the queue, even if that causes the queue's capacity to be exceeded. (queue elasticity.)
- when demand increases, the queue will only return to capacity as resources are released from other queues as containers complete. If it's not have configured policy



Scheduling in Yarn

Fair Scheduler dynamically balance resources (evenly between all tasks) between all running jobs. There is also queue hierarchy for organisations.

- If queue policy is not configured it is Fair (50/50% or 1:1)
- Preemption allows the scheduler to kill containers for queues that are running with more than their fair share of resources
- Delay scheduling allows allocating container in same node where application was submitted.
- Dominant Resource Fairness (drf) gives priority to tasks which have the most dominant resources



Running an Application through YARN

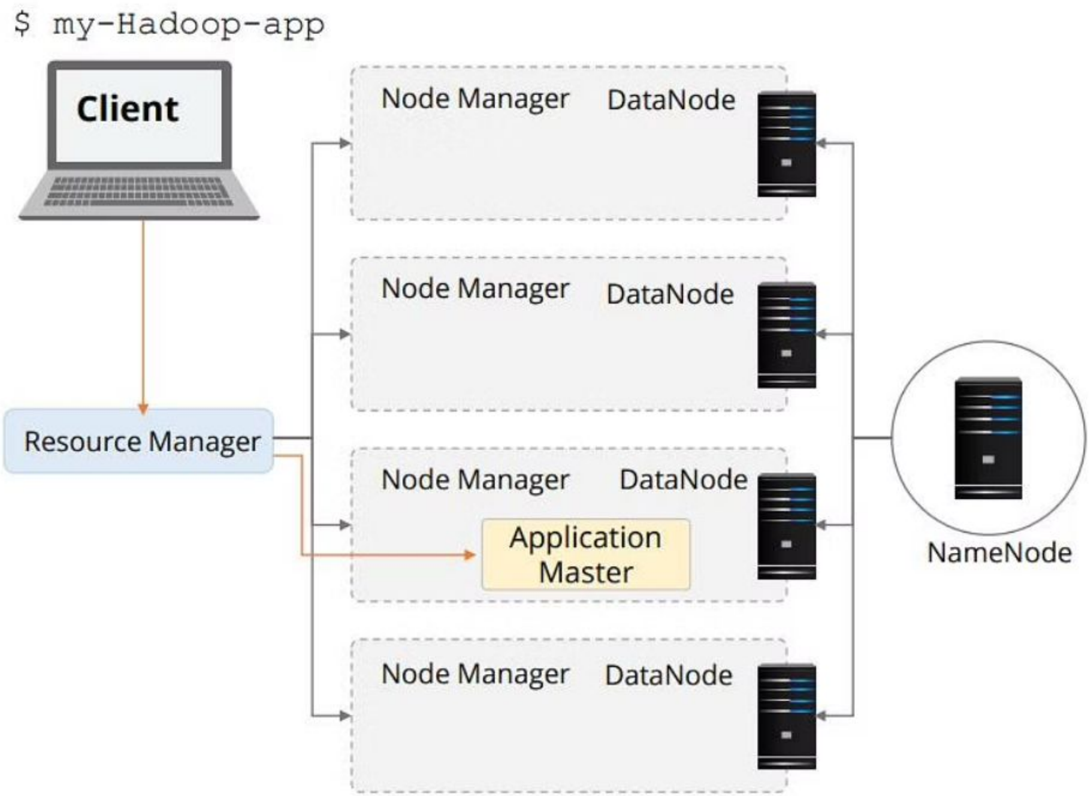
Steps

1. The client submits an application to the Resource Manager
2. The ResourceManager allocates a container
3. The Application Master contacts the related Node Manager
4. The Node Manager launches the container
5. The container executes the Application Master

Step 1 - Application submitted to the Resource Manager

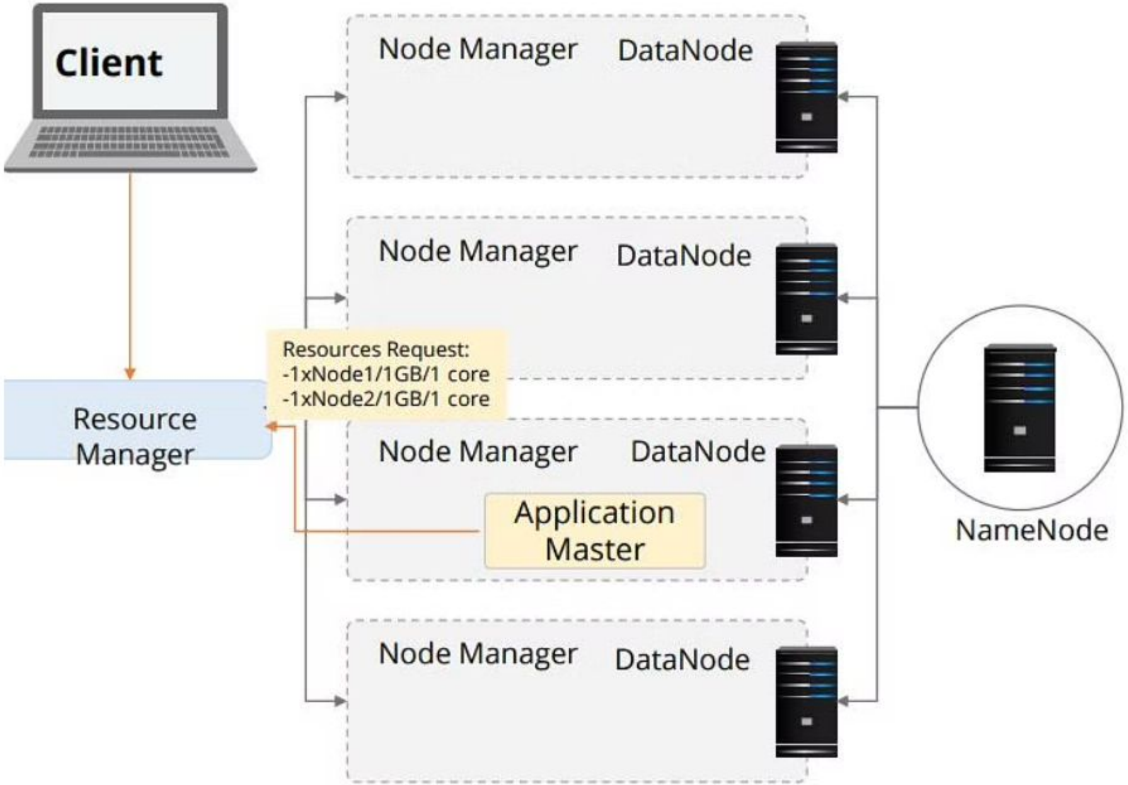
The Resource Manager maintains the list of applications on the cluster and available resources on the Node Manager. The Resource Manager determines the next application that receives a portion of the cluster resource. The decision is subject to many constraints such as queue capacity, Access Control Lists, and fairness.

Users submit applications to the Resource Manager by typing the Hadoop jar command.

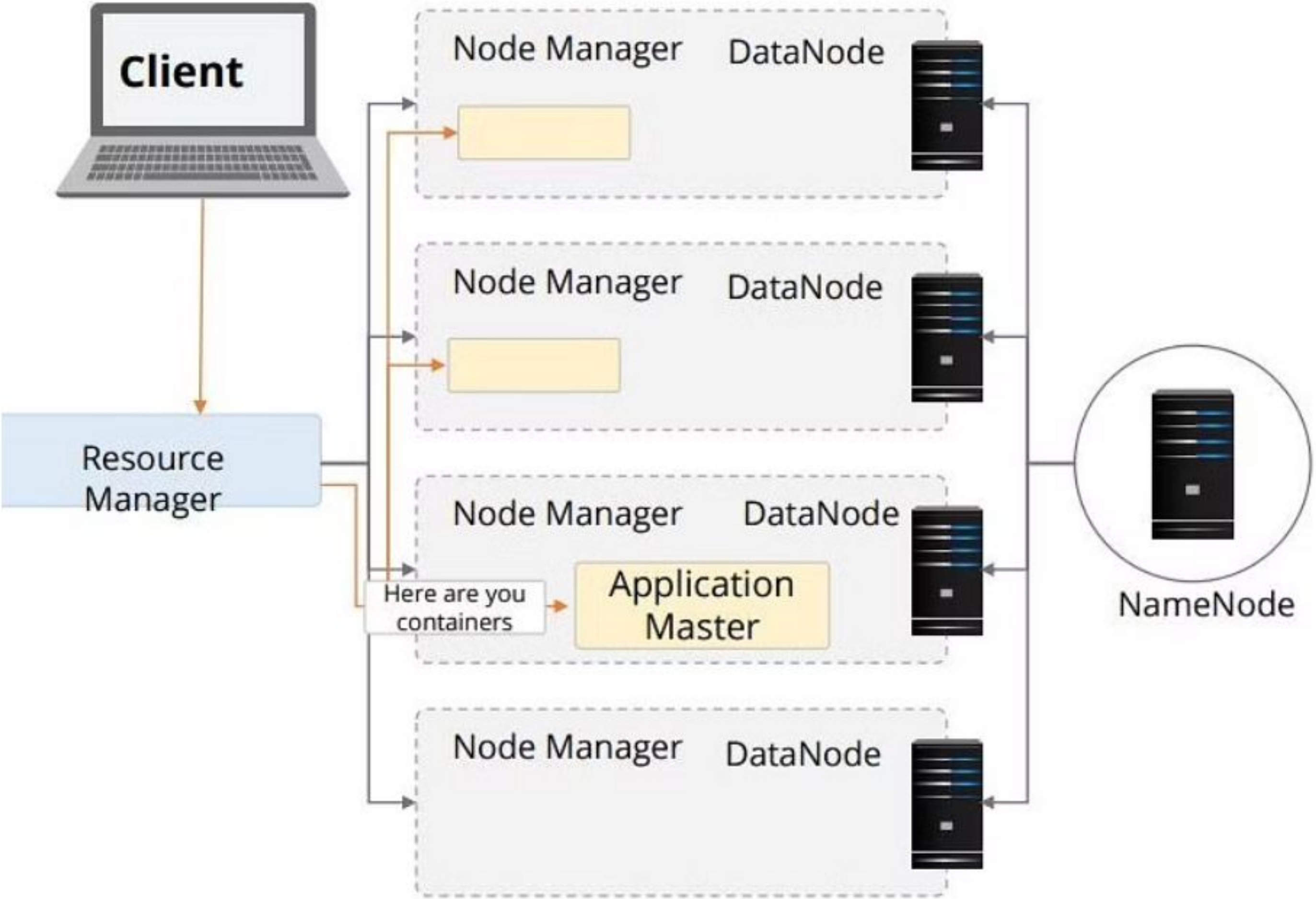


Step 2 - Resource Manager allocates Container

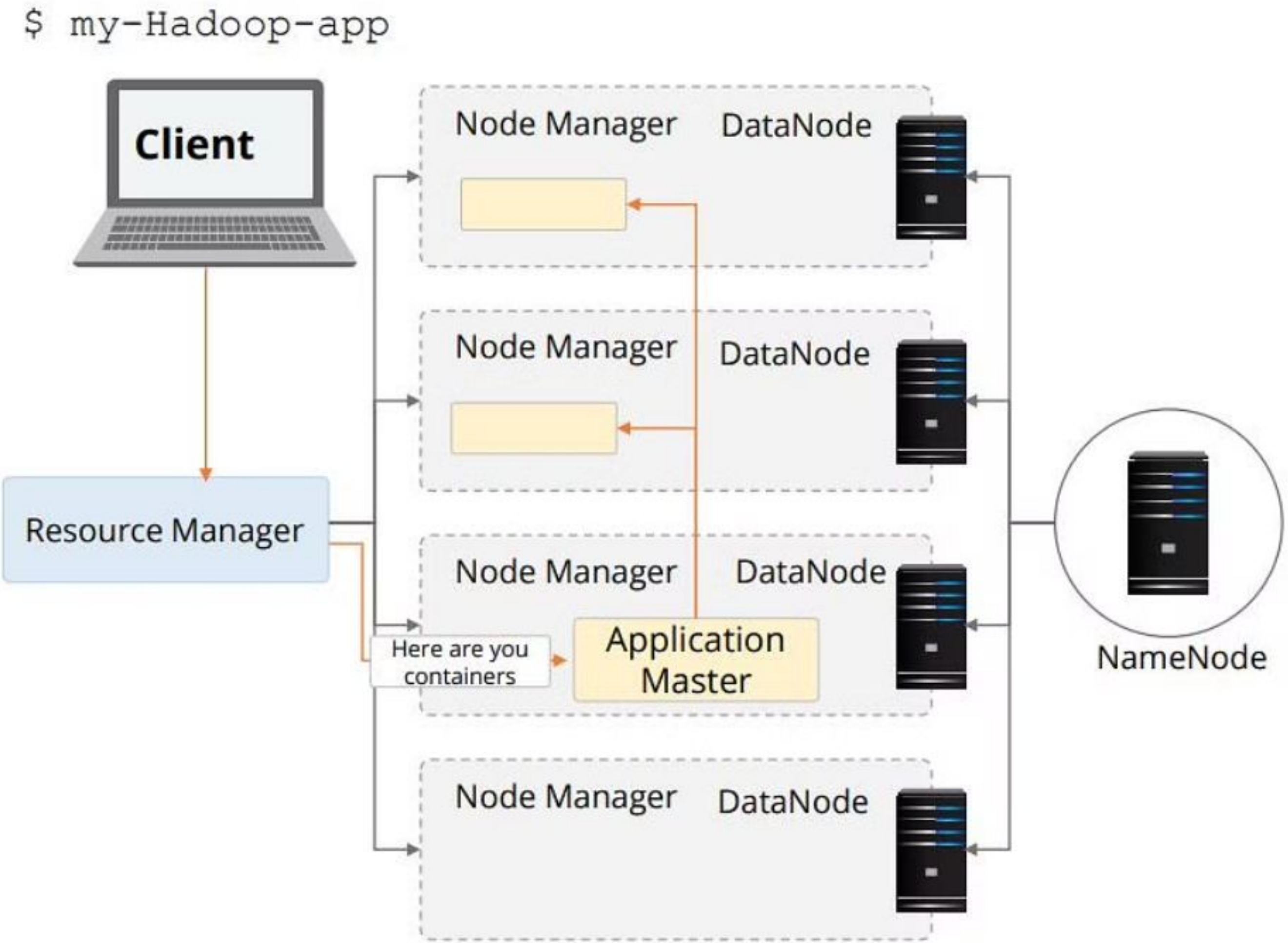
When the Resource Manager accepts a new application submission, one of the first decisions the Scheduler makes is selecting a container. Then, the Application Master is started and is responsible for the entire life-cycle of that particular application.



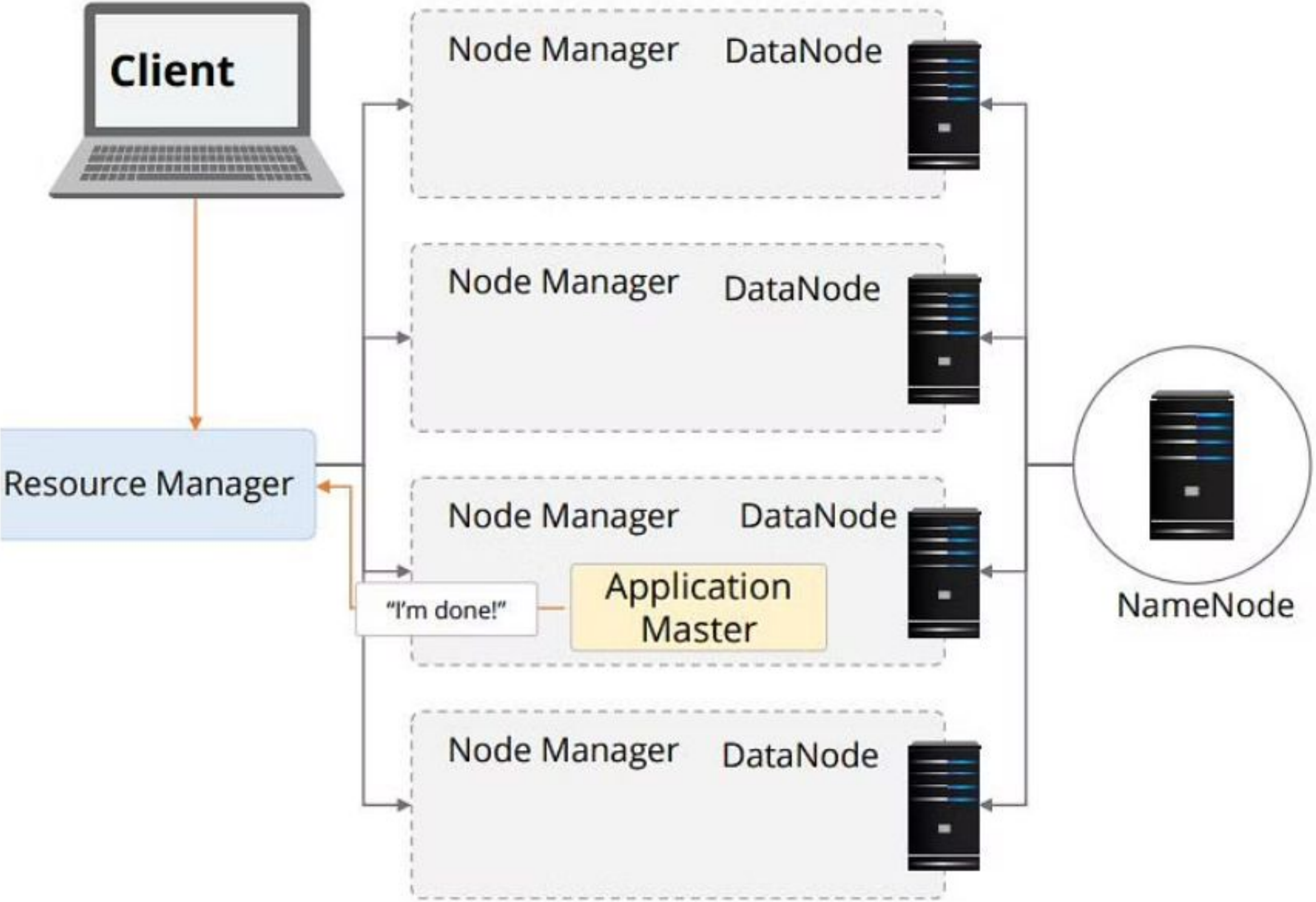
Step 3 - Application Master contacts Node Manager



Step 4 -Resource Manager Launches Container



Step 5 - Container Executes the Application Master


Client
Resource Manager
Node Manager
DataNode
Node Manager
DataNode
Node Manager
DataNode
"I'm done!"
Application Master
Node Manager
DataNode
NameNode