Introduction to HADOOP YARN

NIELIT Chandigarh/Ropar

"Big Data is at the foundation of all the megatrends that are happening today."

Yet Another Resource Negotiator (YARN)

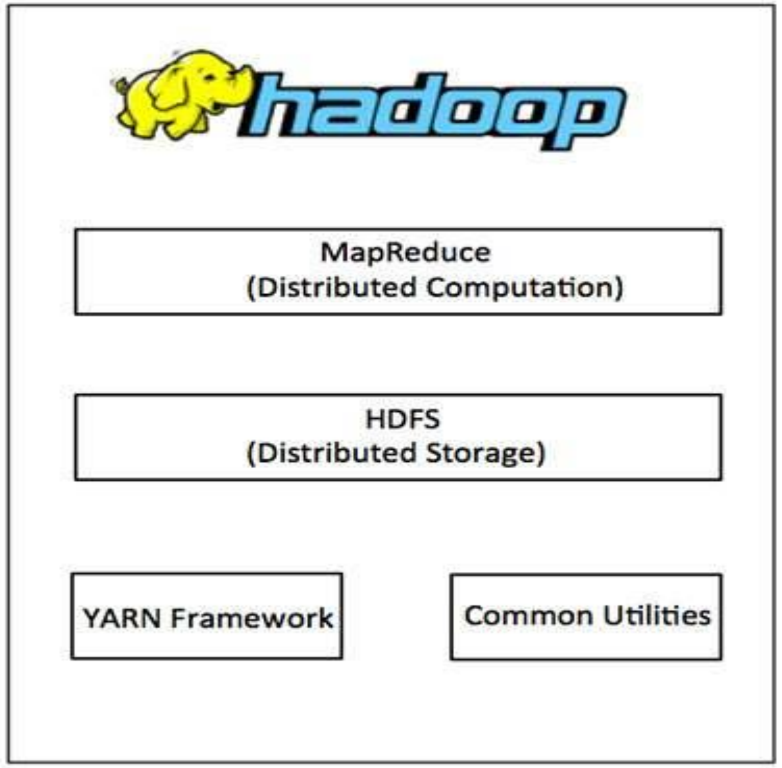
- **YARN** is the resource management layer of the Hadoop ecosystem.
- It decouples resource management from data processing, significantly improving scalability and system performance.
- Acts as an operating system for Hadoop clusters, allowing multiple applications to share resources efficiently.

Benefits of YARN

- **Enhanced Scalability**: Supports a larger number of nodes and applications.
- **Flexibility**: Allows integration of different processing engines like Spark, Tez, and Storm.
- **Efficient Resource Utilization**: Dynamically allocates resources based on demand, ensuring minimal waste.
- **Multi-tenancy**: Enables multiple frameworks to run simultaneously on the same cluster.

Yarn

- YARN is the prerequisite for Enterprise Hadoop providing resource management and a central platform to deliver consistent operations, security, and data governance tools across Hadoop clusters.



YARN is responsible for:

- Assigning computational resources like CPU, memory, and I/O to applications
- Scheduling tasks to be executed on different cluster nodes
- Monitoring jobs submitted to a Hadoop cluster

Features of YARN:

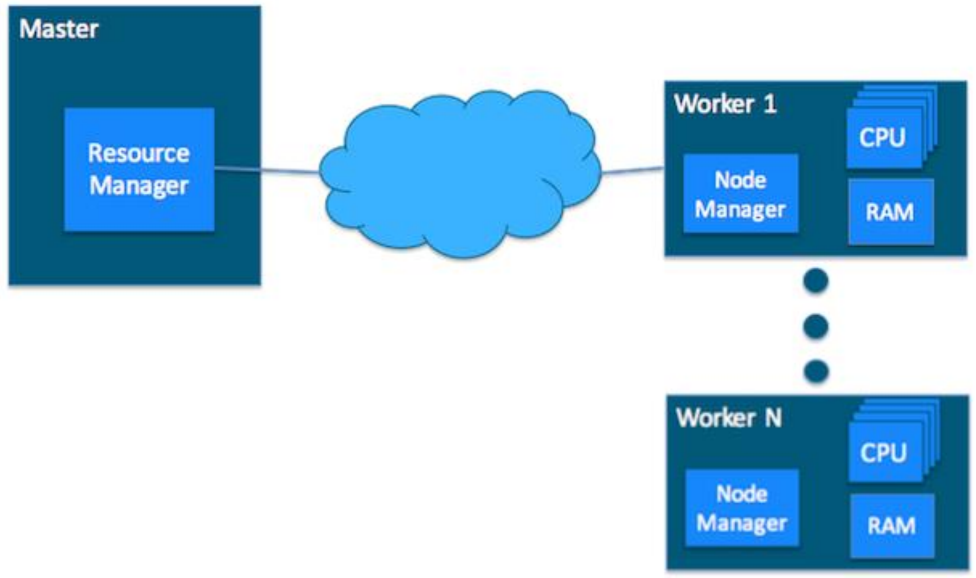
- YARN includes:
- Multi-tenancy
- A global ResourceManager (RM) and per-application ApplicationMaster (AM)
- Automatic and manual failover for RMs

Key Components of YARN

- **ResourceManager (RM)**
  - The central authority for resource allocation in the Hadoop cluster.
  - Tracks available resources and assigns them to applications based on requirements.
  - Ensures optimized utilization of cluster resources.
- **NodeManager (NM)**
  - Deployed on every node in the Hadoop cluster.
  - Monitors the resource usage (CPU, memory, etc.) of containers running on the node.
  - Reports the status of resources to the ResourceManager.
- **ApplicationMaster (AM)**
  - Manages the **lifecycle of applications** within YARN.
  - Coordinates with the ResourceManager to request necessary resources.
  - Ensures the execution of application-specific tasks within the allocated containers.

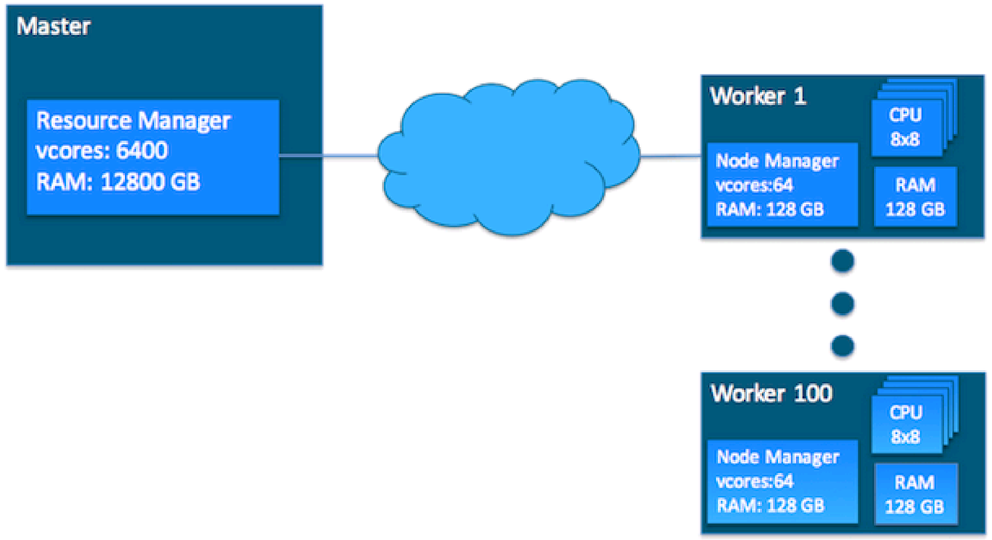
YARN Cluster Basics

- In a YARN cluster, there are two types of hosts:
  - The *ResourceManager* is the master daemon that communicates with the client, tracks resources on the cluster, and orchestrates work by assigning tasks to *NodeManagers*.
  - A *NodeManager* is a worker daemon that launches and tracks processes spawned on worker hosts.



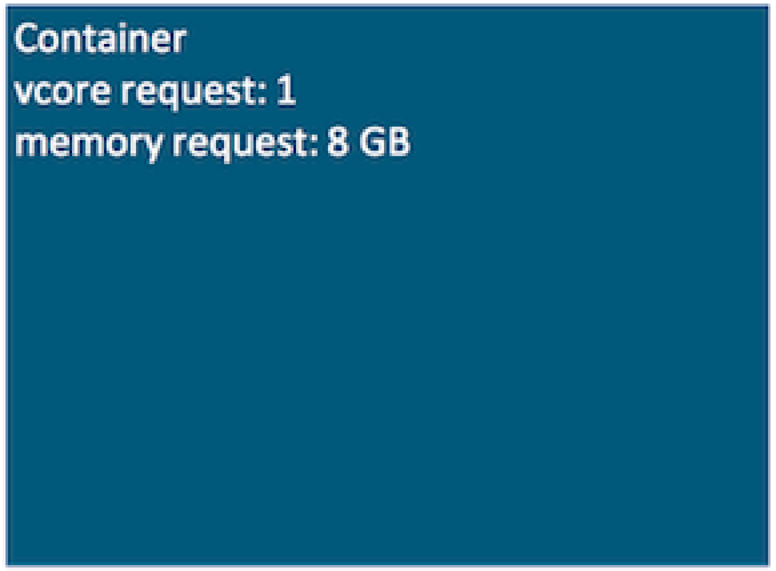
Yarn Resource Monitoring

- YARN currently defines two resources:
  - *v-cores*
  - *memory*.
- Each NodeManager tracks
  - its own local resources and
  - communicates its resource configuration to the ResourceManager
- The ResourceManager keeps
  - a running total of the cluster's available resources.

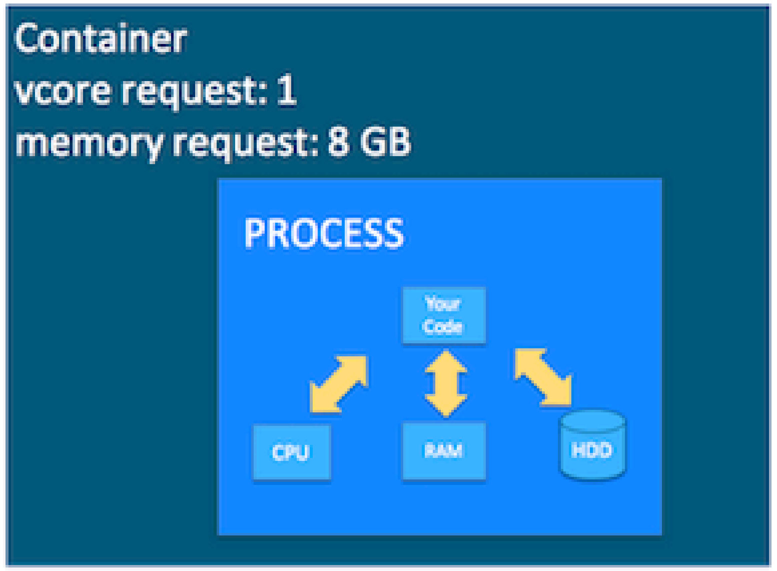


Yarn Container

- Containers
  - a request to hold resources on the YARN cluster.
  - a container hold request consists of vcore and memory



Container as a hold



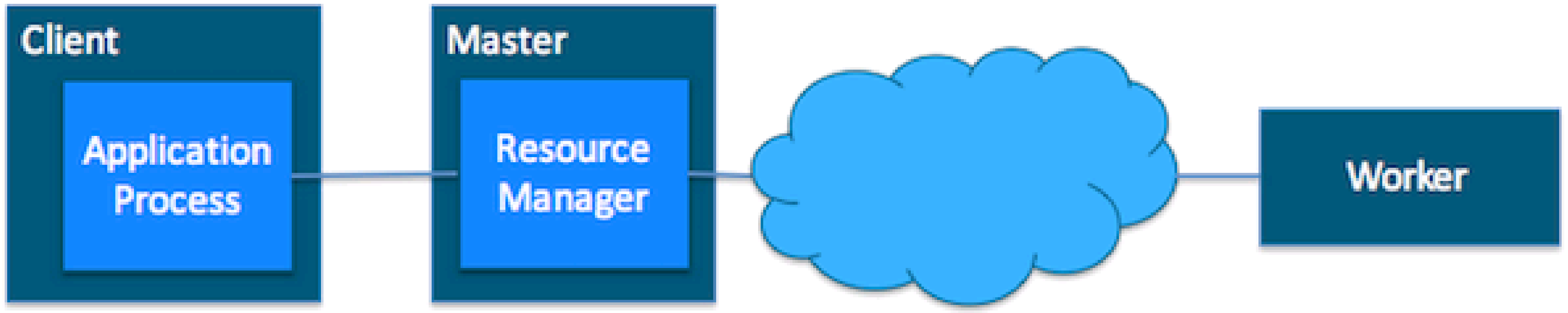
The task running as a process inside a container

Yarn Application and ApplicationMaster

- Yarn application
  - It is a YARN client program that is made up of one or more tasks
  - Example: MapReduce Application
- ApplicationMaster
  - It helps coordinate tasks on the YARN cluster for each running application
  - It is the first process run after the application starts.

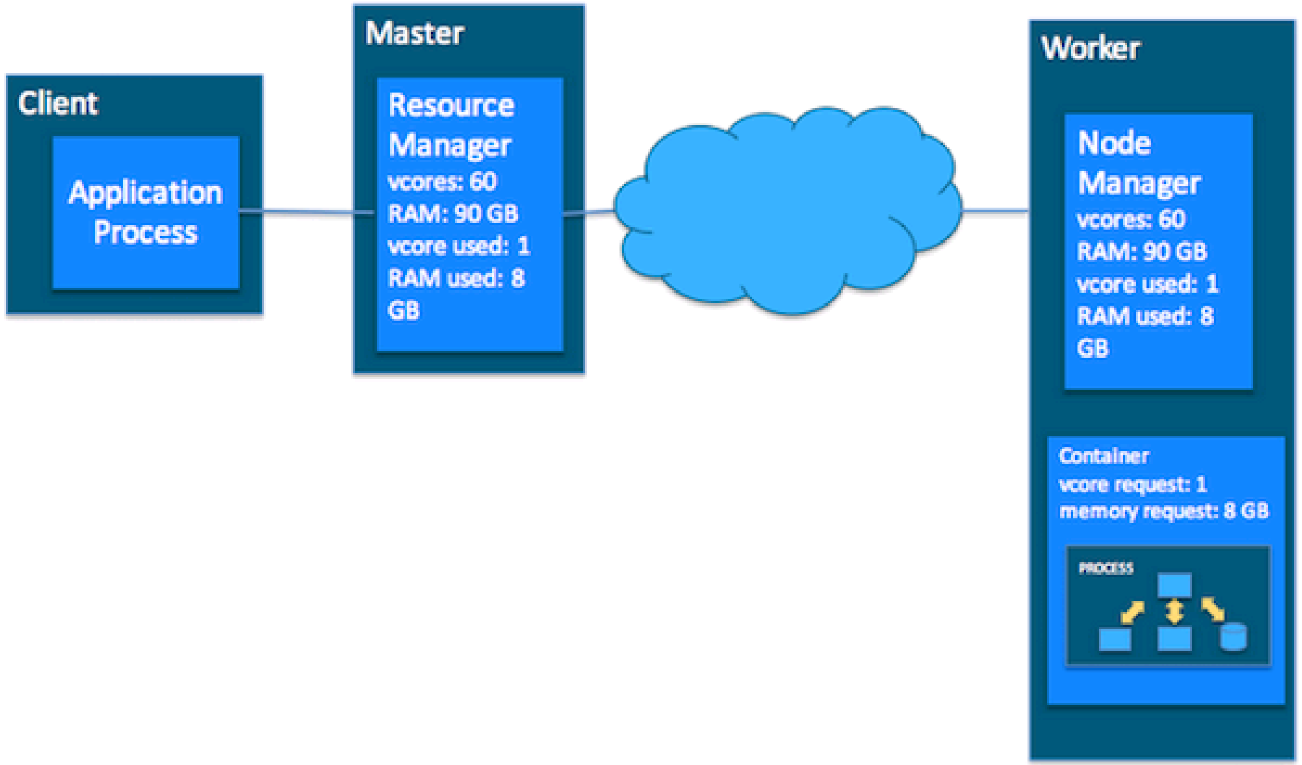
Interactions among Yarn Components

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



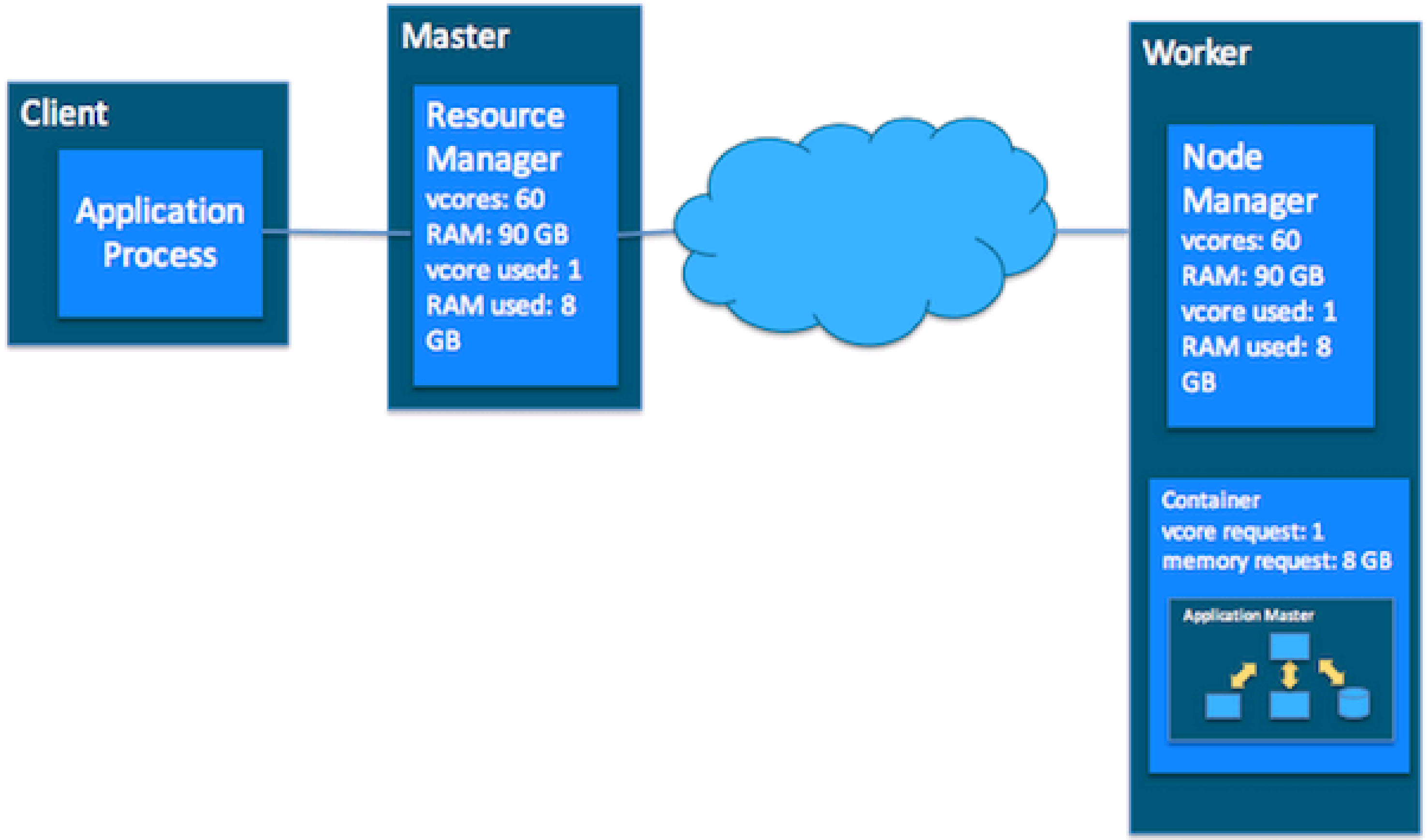
Interactions among Yarn Components

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



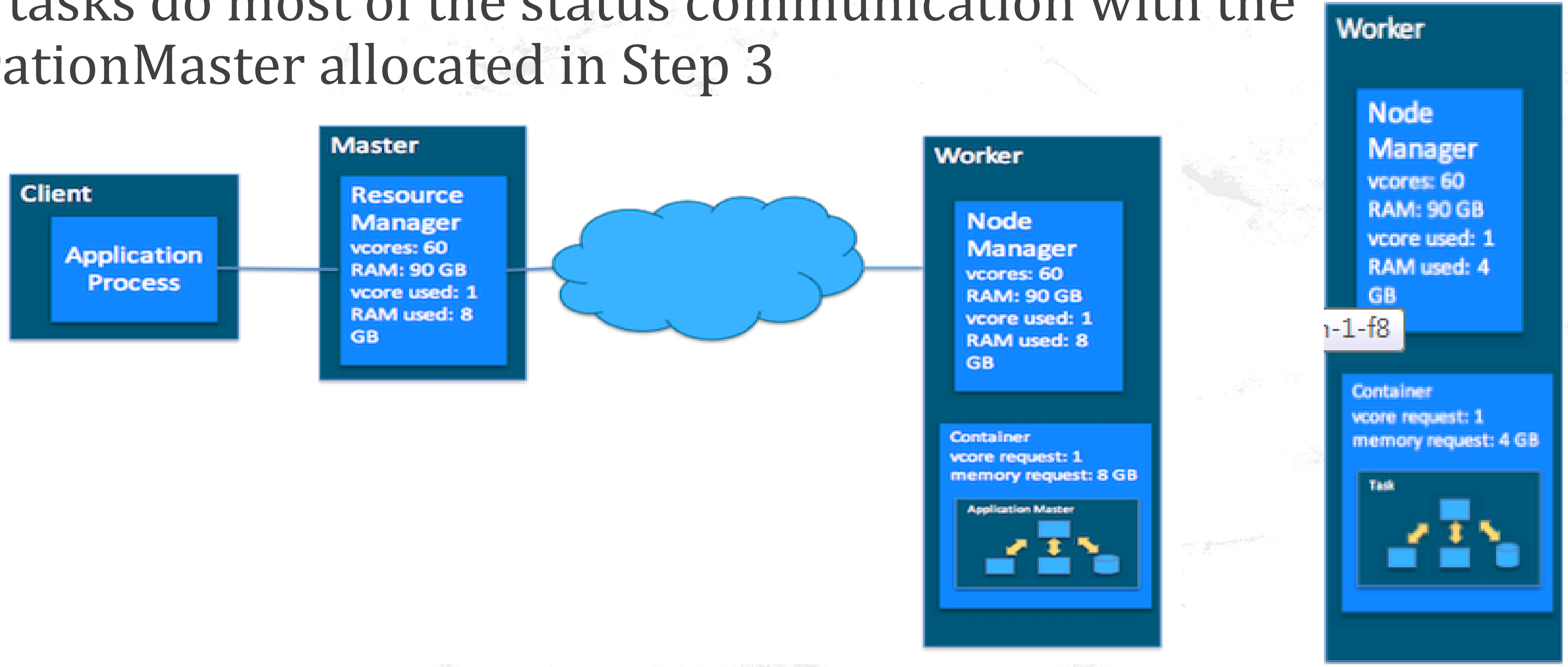
Interactions among Yarn Components

3. The ApplicationMaster starts running within that container



Interactions among Yarn Components

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



Interactions among Yarn Components

5. Once all tasks are finished, the ApplicationMaster exits. The last container is de-allocated from the cluster.

6. The application client exits. (The ApplicationMaster launched in a container is more specifically called a managed AM. Unmanaged ApplicationMasters run outside of YARN’s control.)

Practical: Running a Word Count Job on YARN