



# YARN

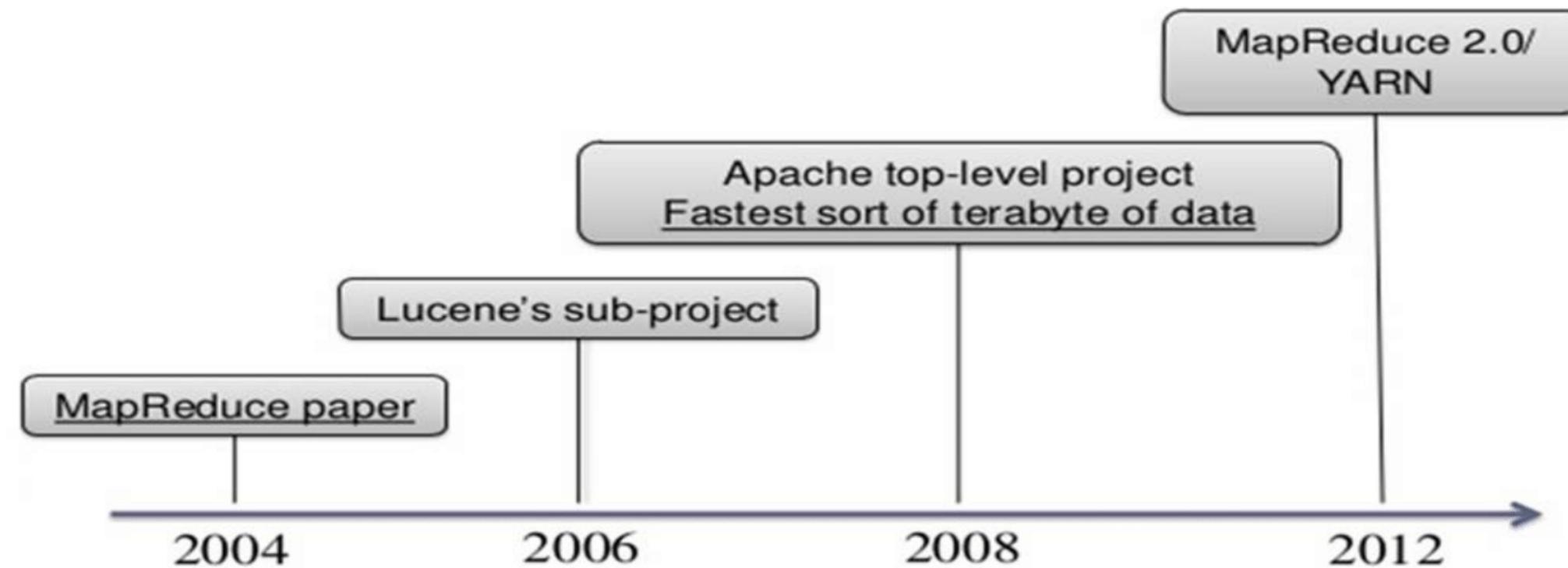
---

## Yet Another Resource Negotiator

- ✓ Evolution of YARN
- ✓ Architecture
- ✓ Ecosystem
- ✓ Introduction to YARN
- ✓ Daemons
- ✓ Application Execution
- ✓ Container
- ✓ MapReduce Application Execution



# Evolution of yarn



## The Data Operating System for Hadoop 2.0

### Flexible

Enables other purpose-built data processing models beyond MapReduce (batch), such as interactive and streaming

### Efficient

Increase processing **IN** Hadoop on the same hardware while providing predictable performance & quality of service

### Shared

Provides a stable, reliable, secure foundation and shared operational services across multiple workloads

- Apache™ Hadoop® YARN is a sub-project of Hadoop at the Apache Software Foundation introduced in Hadoop 2.0 that separates the resource management and processing components.
- The YARN-based architecture of Hadoop 2.0 provides a more general processing platform that is not constrained to MapReduce.
- YARN takes the resource management capabilities that were in MapReduce and packages them so they can be used by new engines.

- The fundamental idea of YARN is to split up the two major responsibilities of the JobTracker i.e. resource management and job scheduling / monitoring, into separate daemons: a global ResourceManager and per-application ApplicationMaster (AM).
- The ResourceManager and per-node slave, the NodeManager (NM), form the new, and generic, **system** for managing applications in a distributed manner.

- Framework supporting multiple applications
  - Separate generic resource brokering from application logic
  - Define protocols/libraries and provide a framework for custom application development
  - Share same Hadoop Cluster across applications
- Cluster Utilization
  - Generic resource container model replaces fixed Map/Reduce slots. Container allocations based on locality, memory
  - Sharing cluster among multiple application

- Scalability
  - Removed complex app logic from RM, scale further
  - State machine, message passing based loosely coupled design
  - Compact scheduling protocol
- Application Agility and Innovation
  - Map Reduce becomes an application in user space unlocking safe innovation
  - Multiple versions of an app can co-exist leading to experimentation
  - Easier upgrade of framework and application

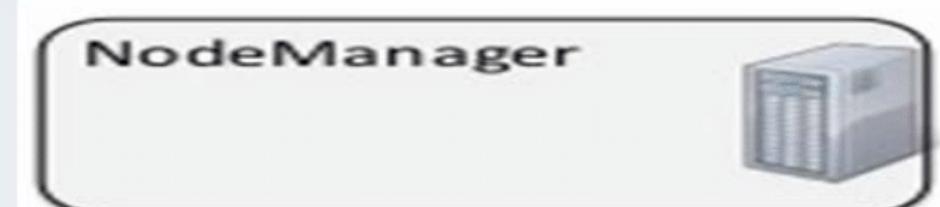
## Resource Manager (RM )

- Run on the master node.
- Global Resource scheduler
- Arbitrate System resources between competing application



## Node Manger

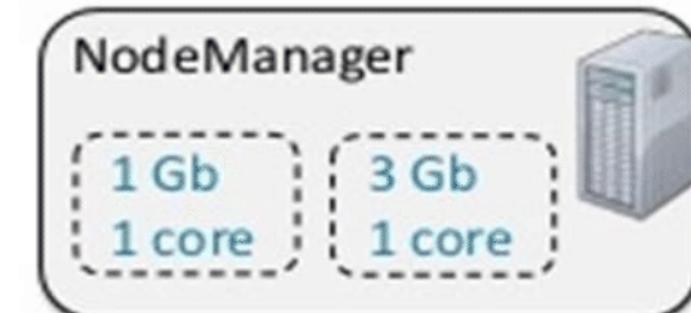
- Runs on the slave node.
- Communicates with RM



# Other Important key element in yarn

## Containers:

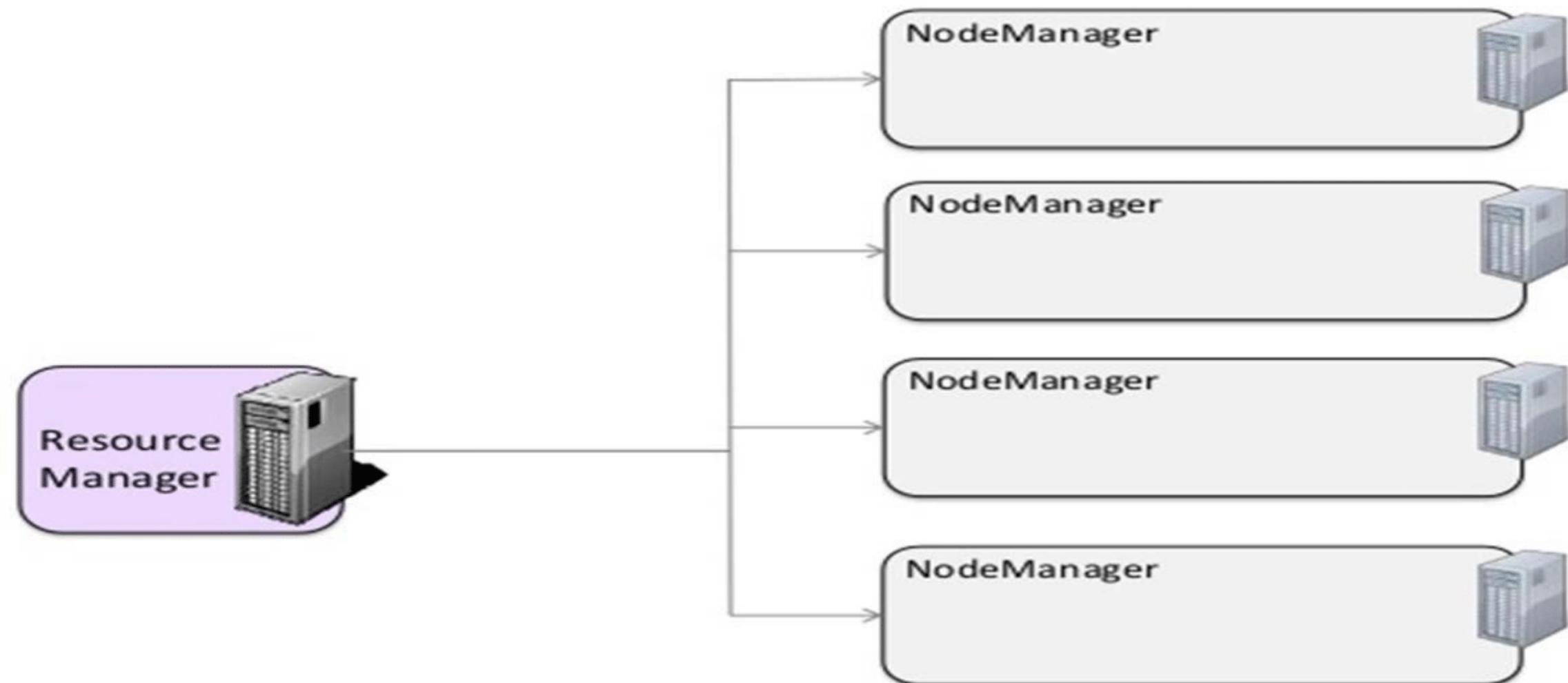
- Allocates a certain amount of resources (memory, cpu) on the slave node
- Application runs on one or more containers



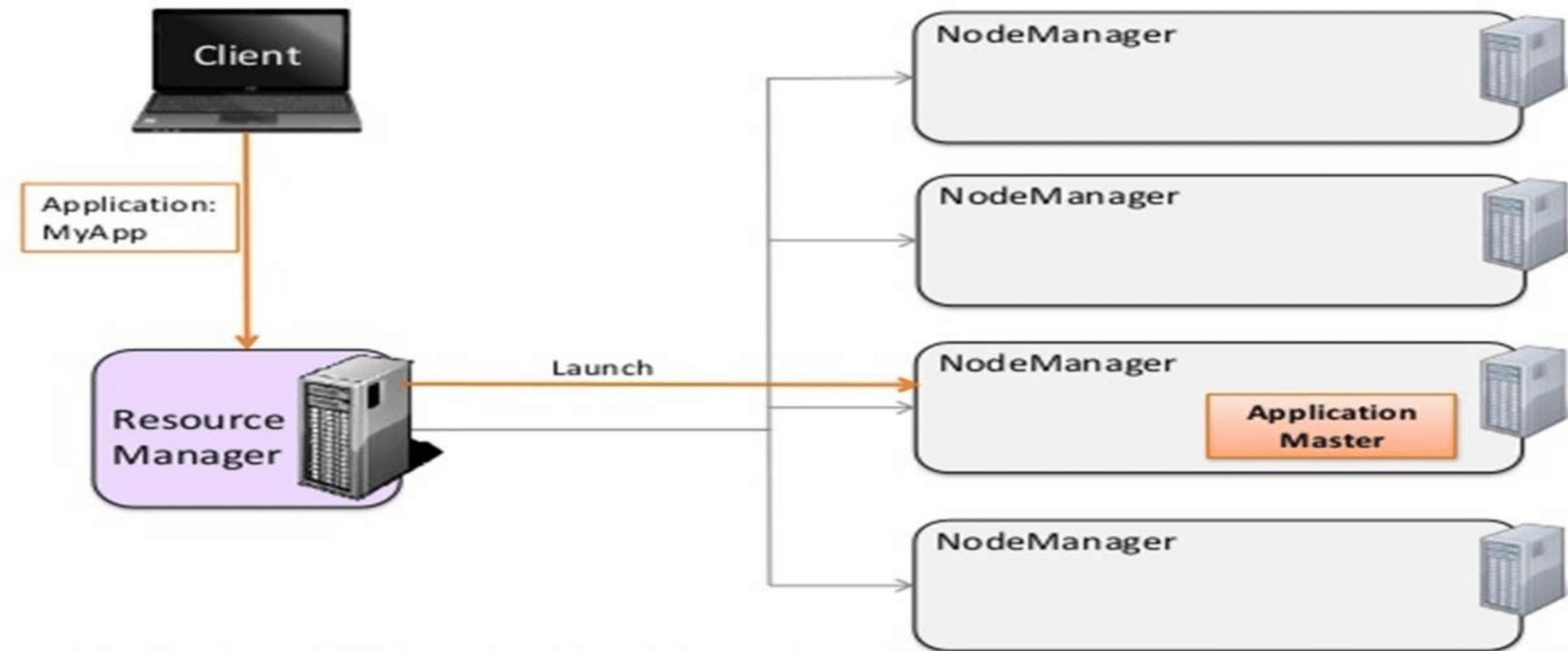
## Application master:

- One per application
- Runs in the container
- Request more container to run job

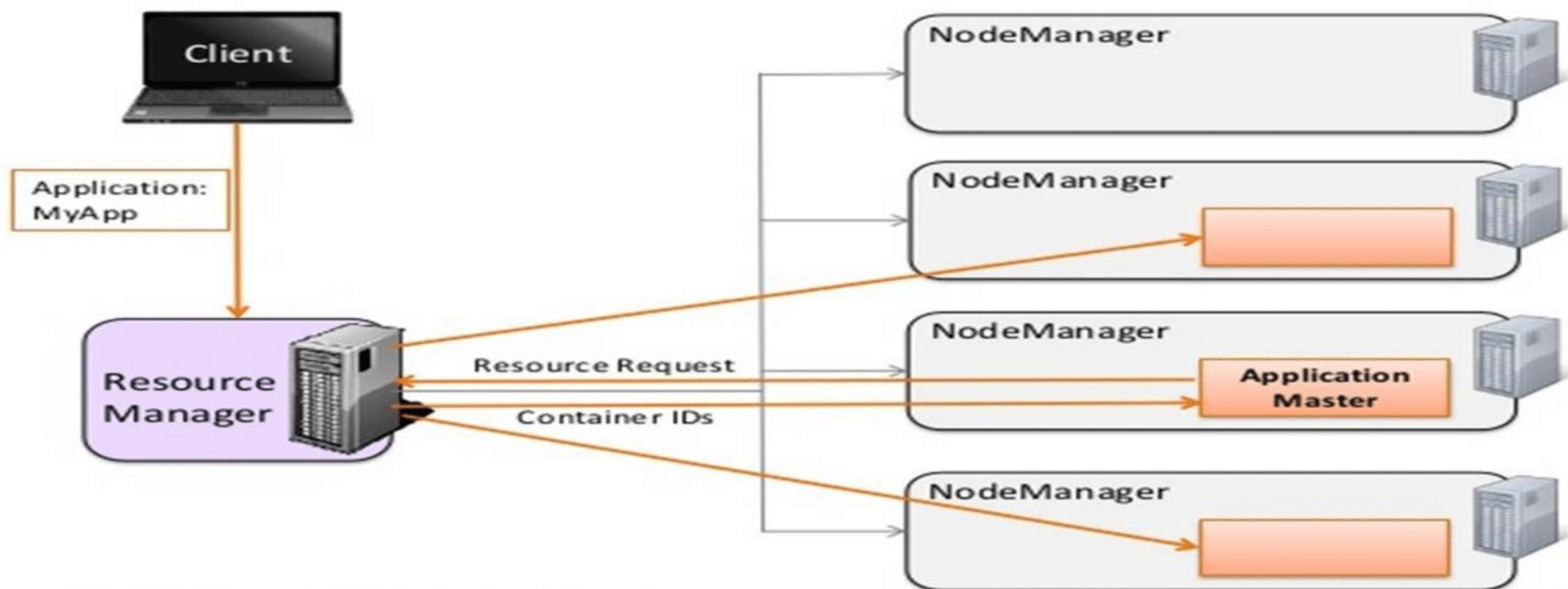




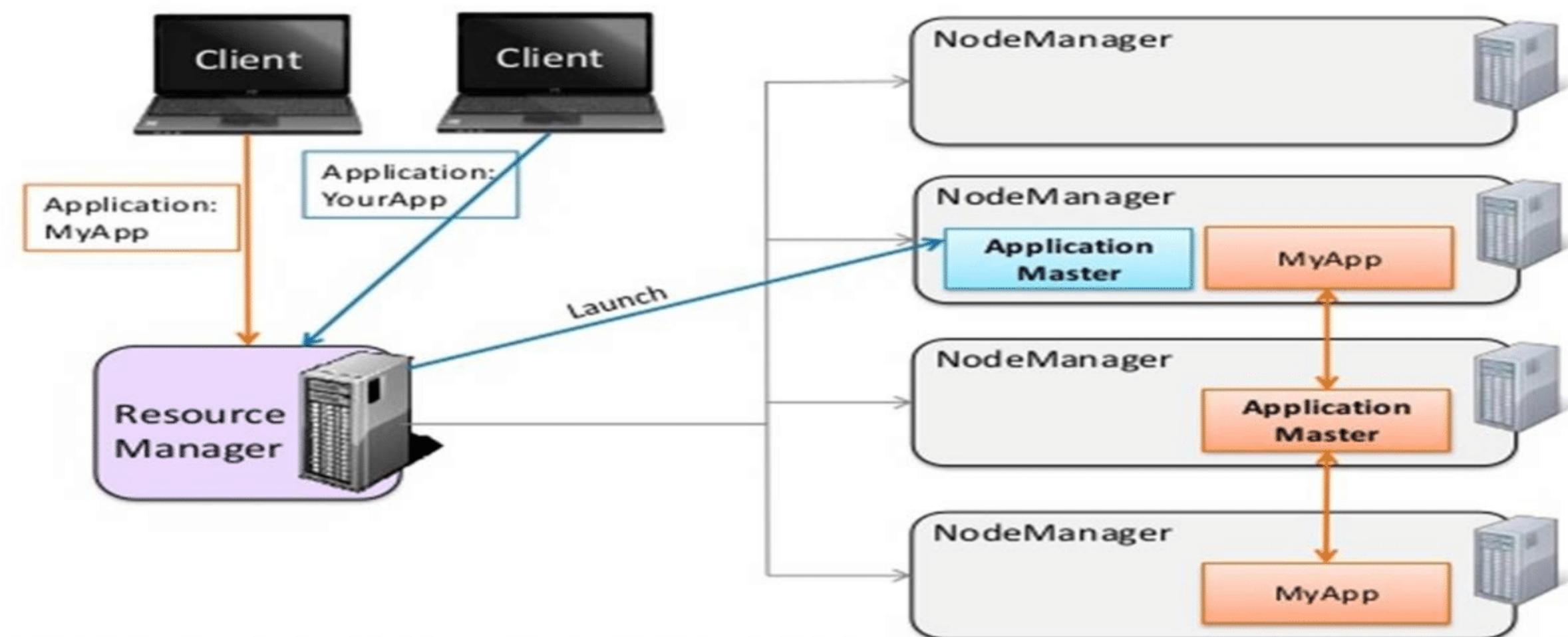
# Yarn Cluster Running an application



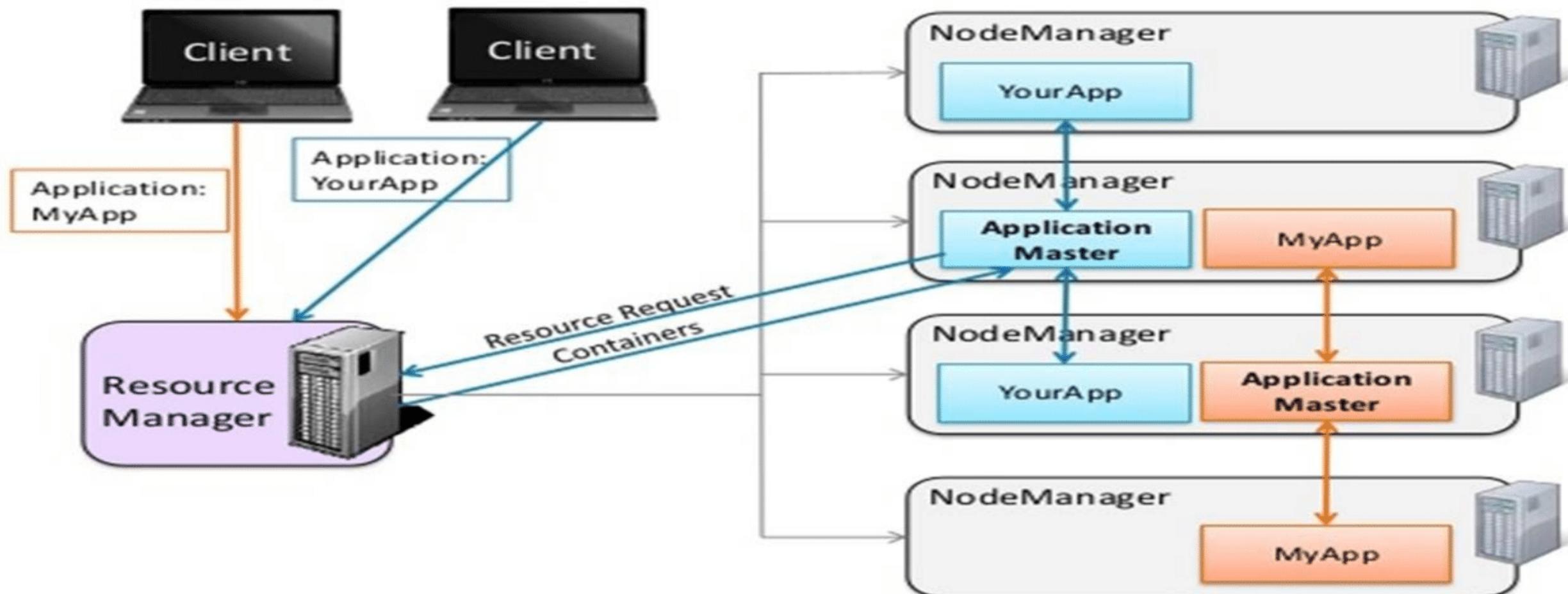
# Yarn Cluster Running an application



# Yarn Cluster Running an application



# Yarn Cluster Running an application



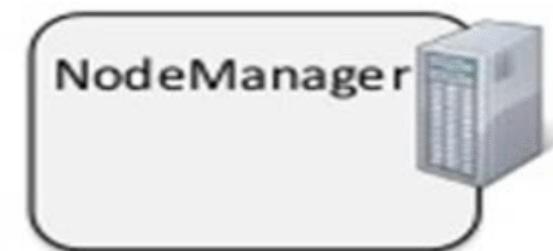
- **What it does**

- Manages nodes
  - Tracks heartbeats from NodeManagers
- Manages containers
  - Handles AM requests for resources
  - De-allocates containers when they expire or the application completes
- Manages ApplicationMasters
  - Creates a container for AMs and tracks heartbeats
- Manages security
  - Supports Kerberos



- **What it does**

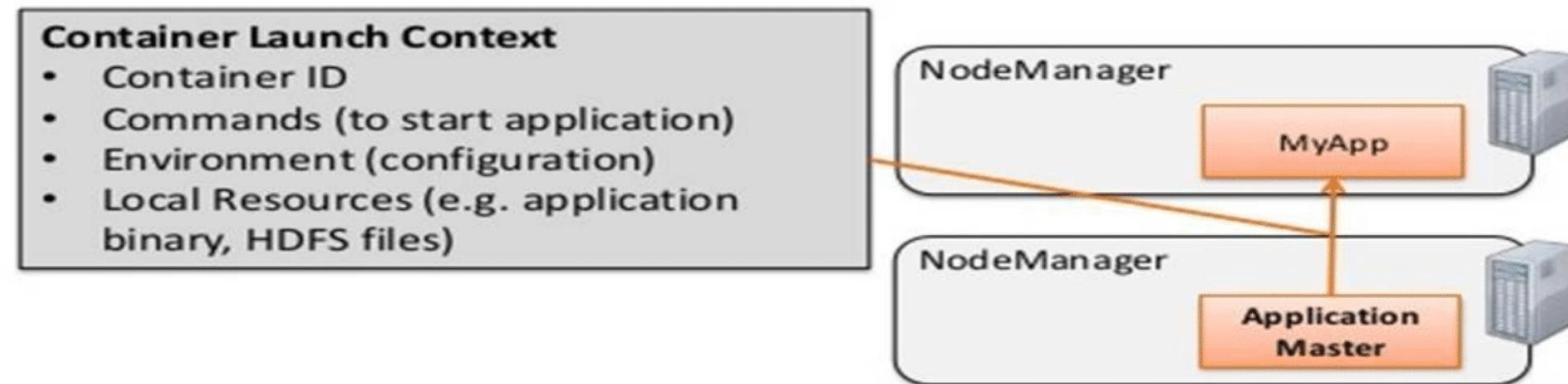
- Communicates with the RM
  - Registers and provides info on node resources
  - Sends heartbeats and container status
- Manages processes in containers
  - Launches AMs on request from the RM
  - Launches application processes on request from AM
  - Monitors resource usage by containers; kills run-away processes
- Provides logging services to applications
  - Aggregates logs for an application and saves them to HDFS
- Runs auxiliary services
- Maintains node level security via ACLs



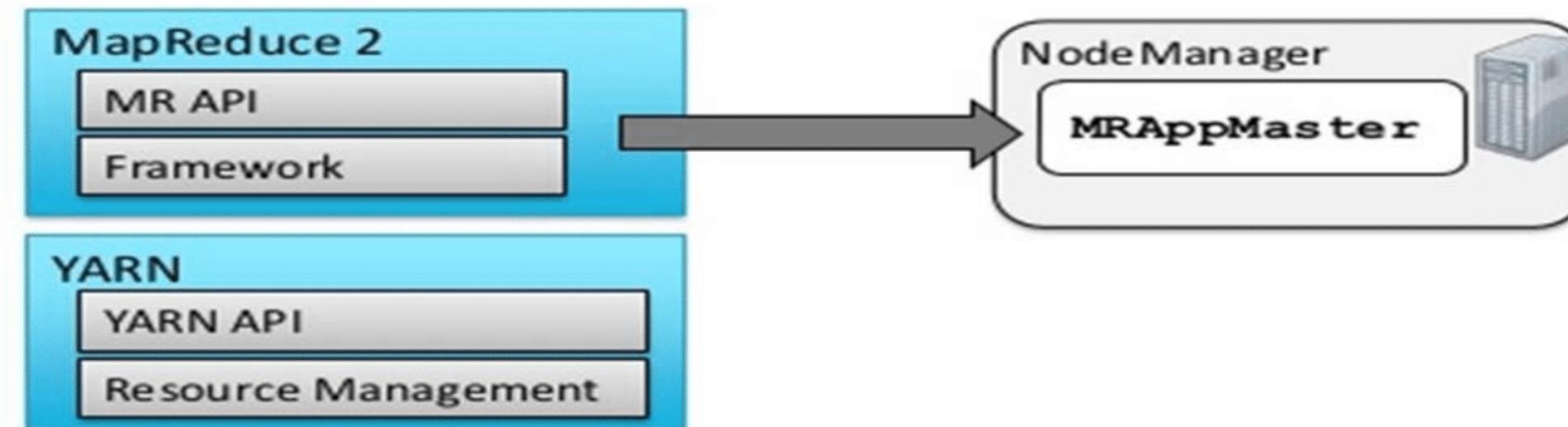
## Resource Request

- Resource name (hostname, rackname or \*)
- Priority (within this application, not between applications)
- Resource requirements
  - memory (MB)
  - CPU (# of cores)
  - more to come, e.g. disk and network I/O, GPUs, etc.
- Number of containers

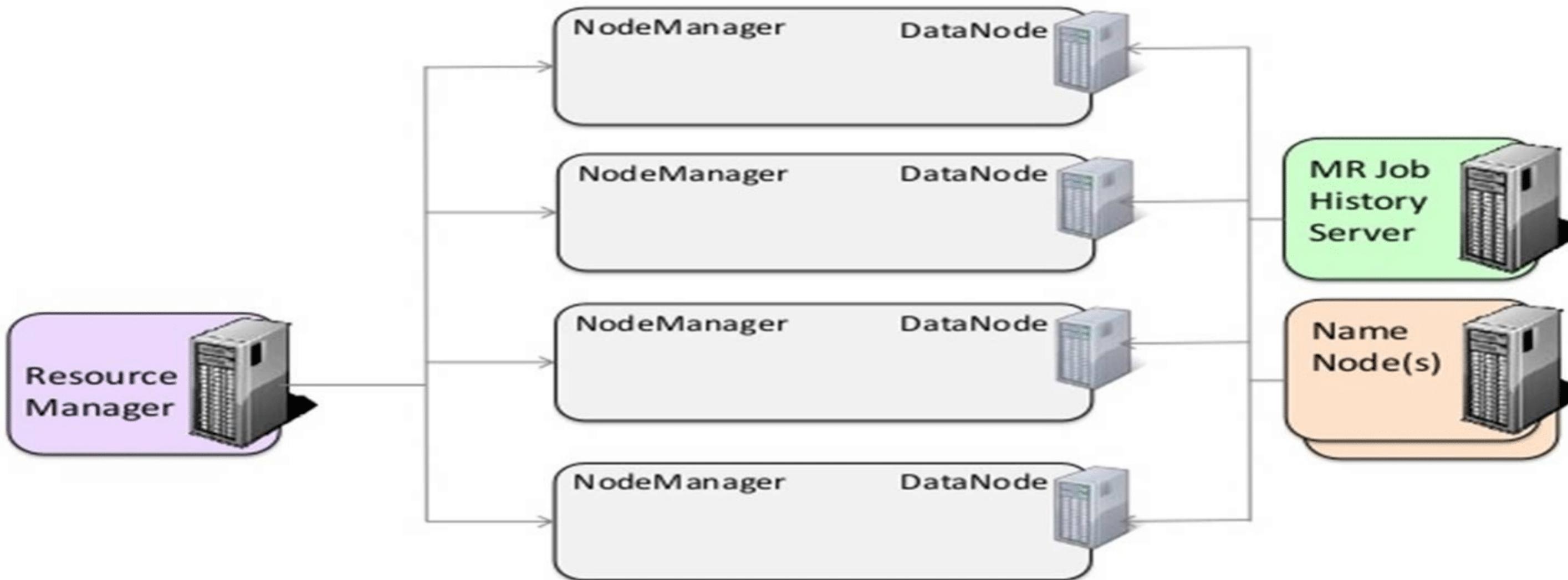




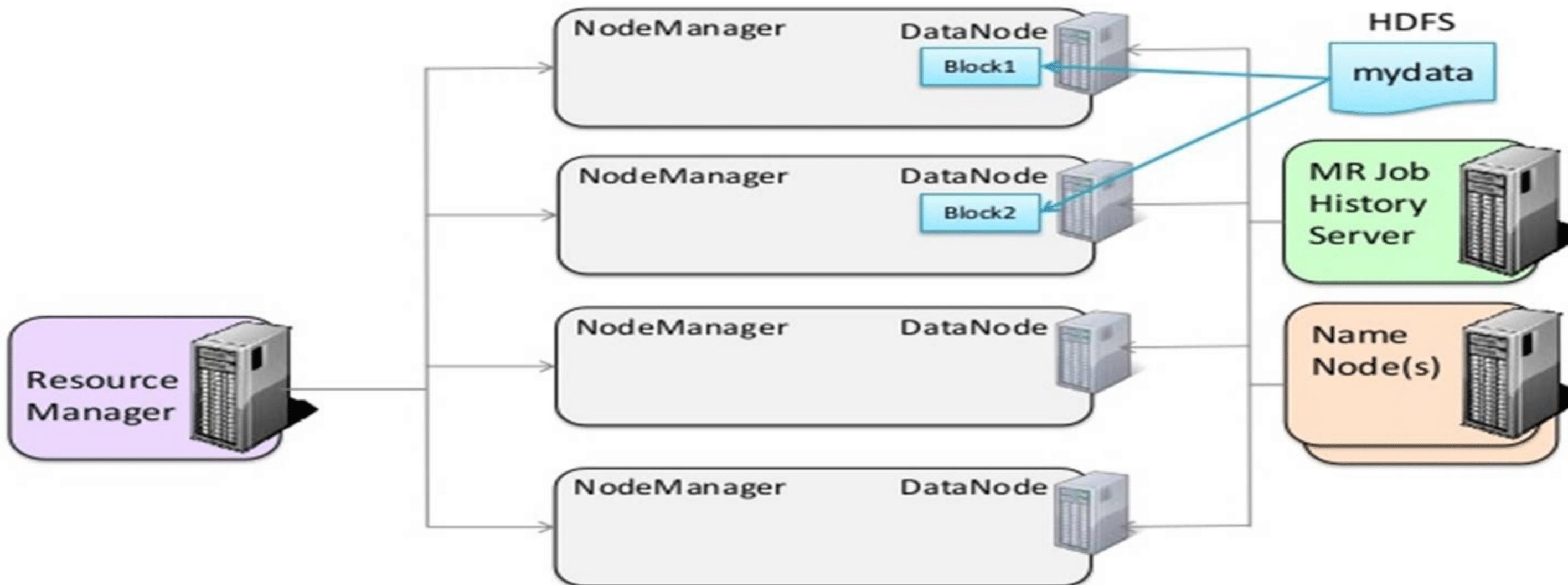
- **YARN does not know or care what kind of application it is running**
  - Could be MR or something else (e.g. Impala)
- **MR2 uses YARN**
  - Hadoop includes a MapReduce ApplicationMaster (**MRAppMaster**) to manage MR jobs
  - Each MapReduce job is an a new instance of an application



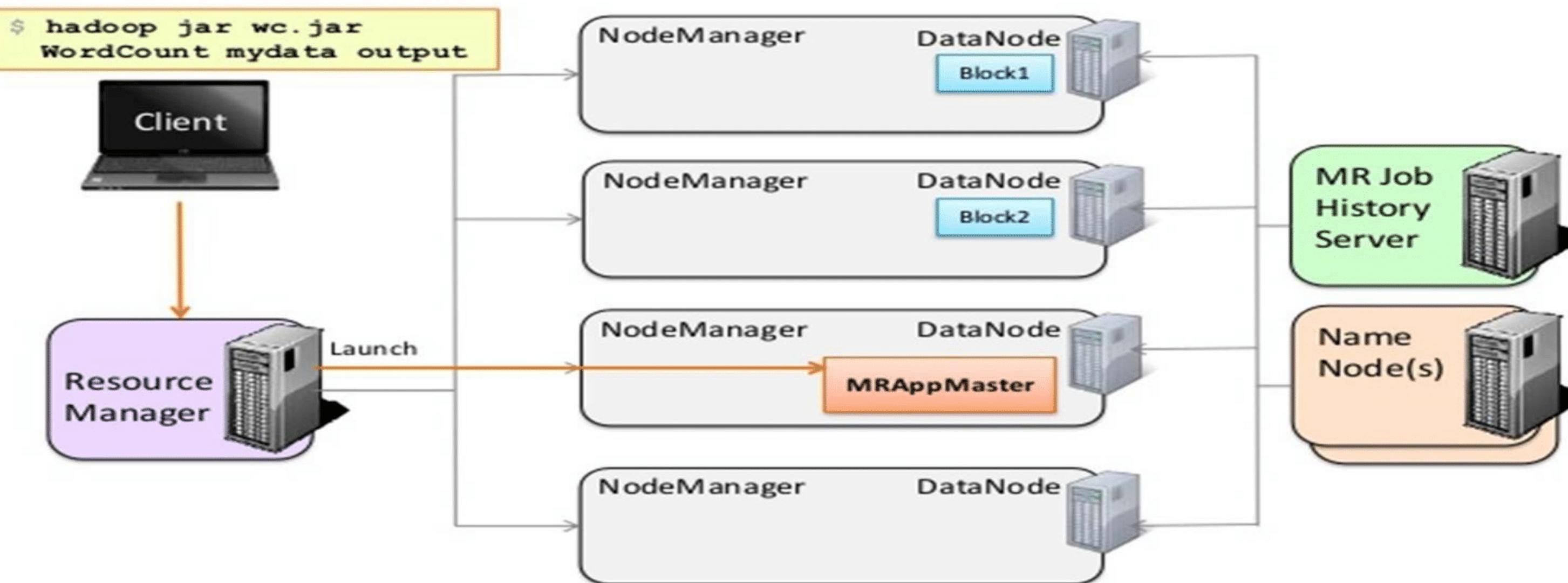
# Running a MapReduce Application



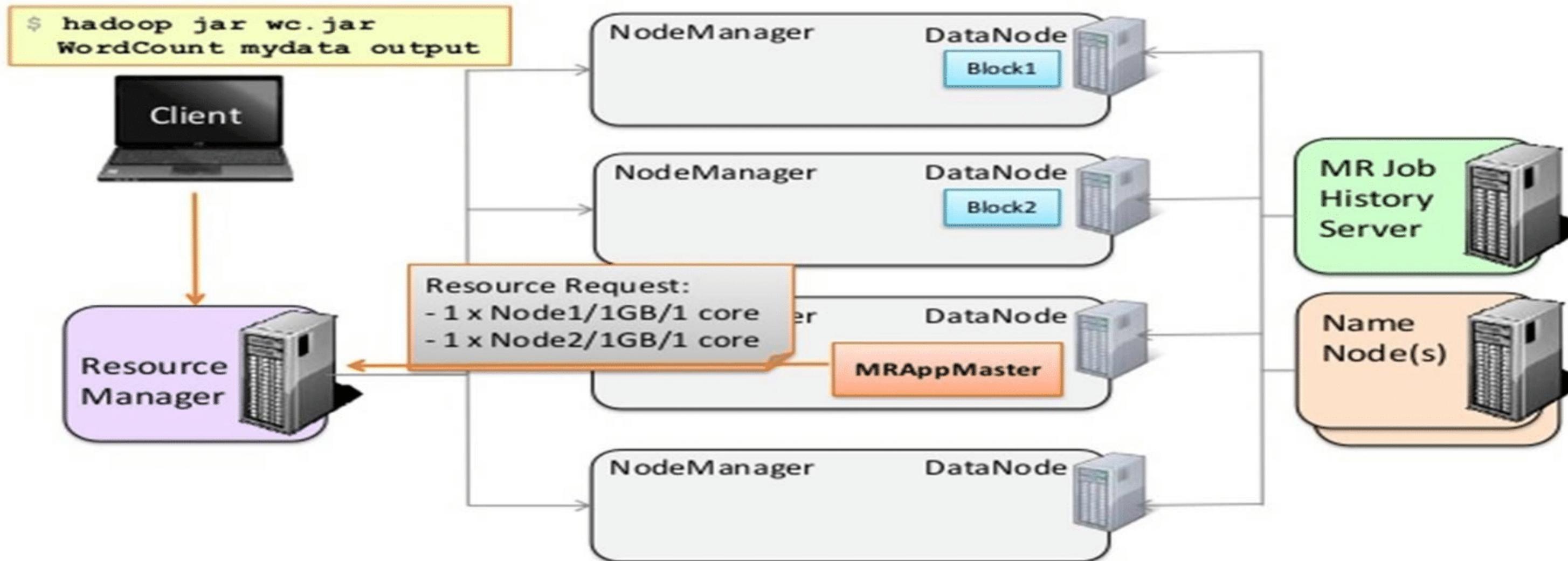
# Running a MapReduce Application



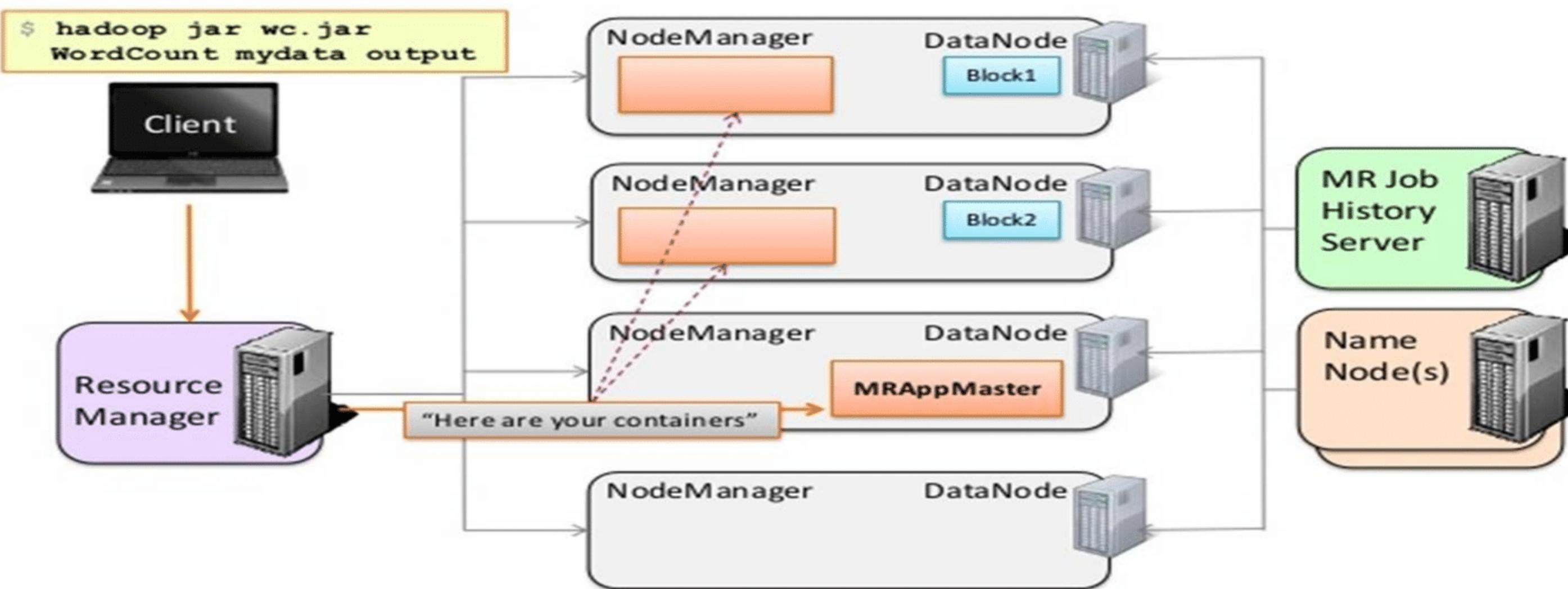
# Running a MapReduce Application



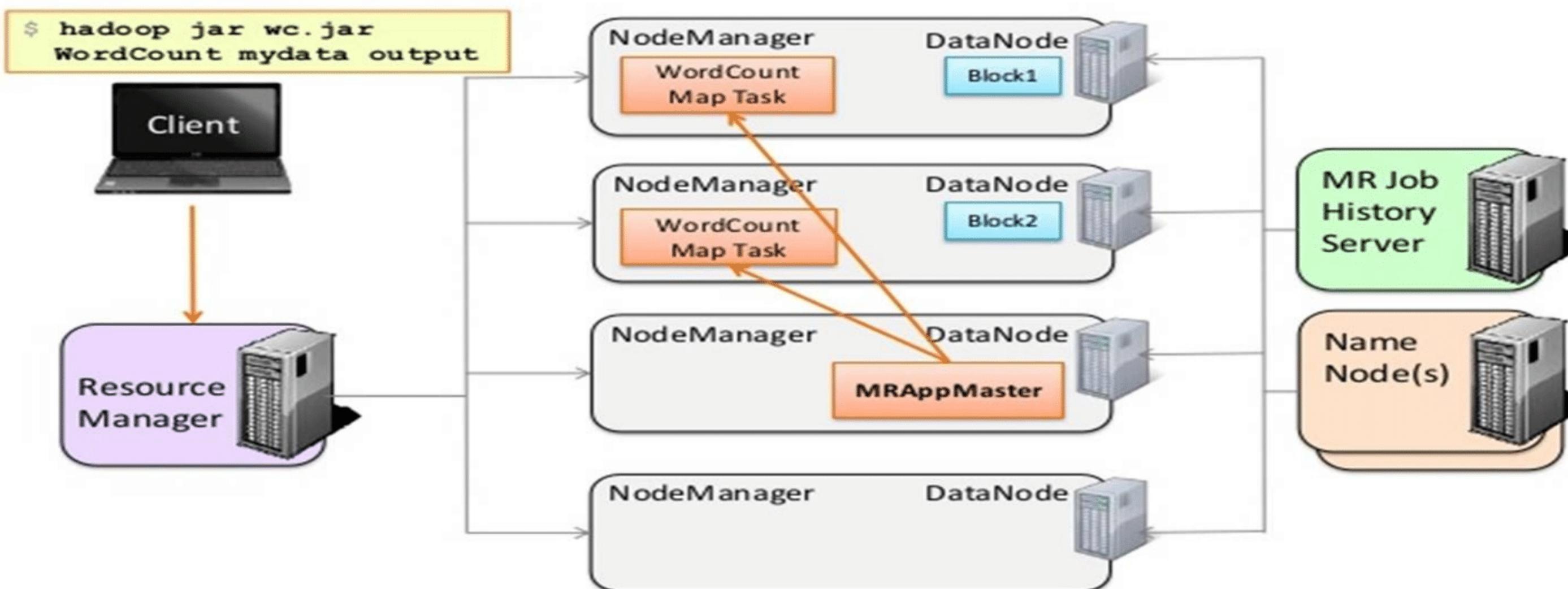
# Running a MapReduce Application



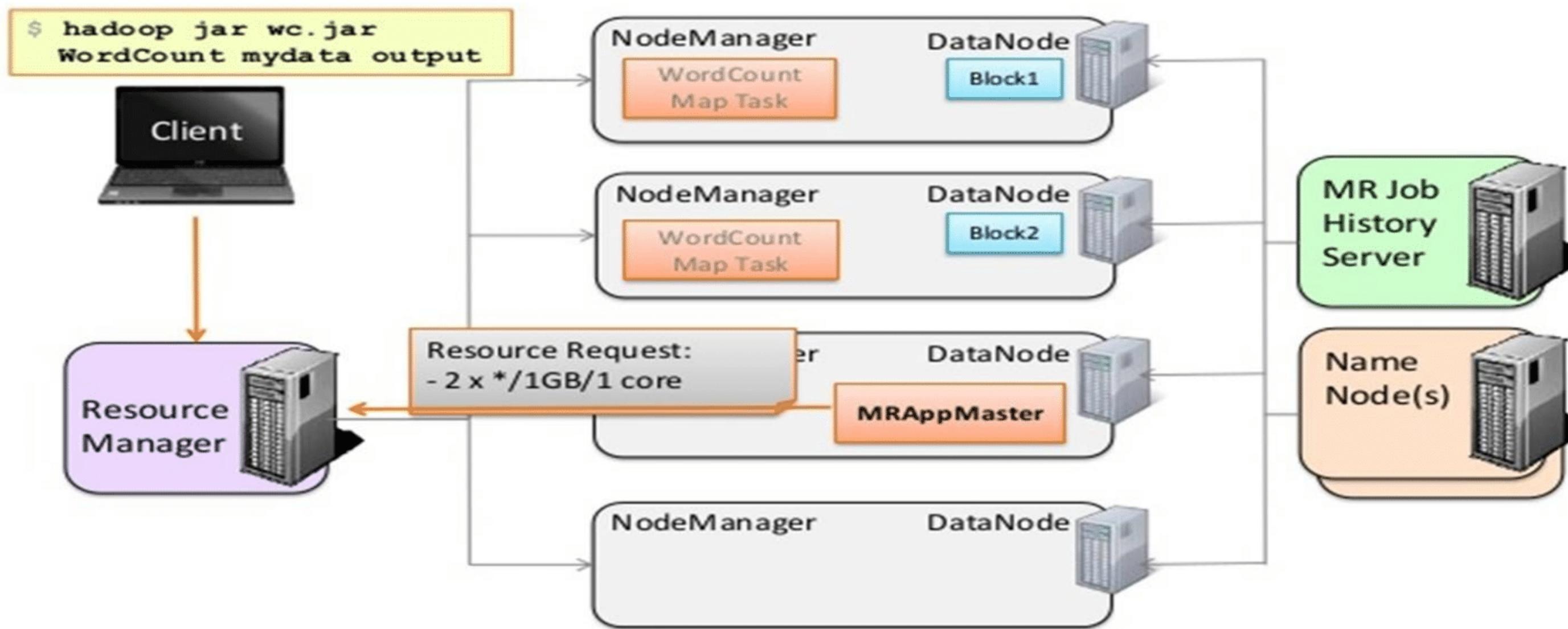
# Running a MapReduce Application



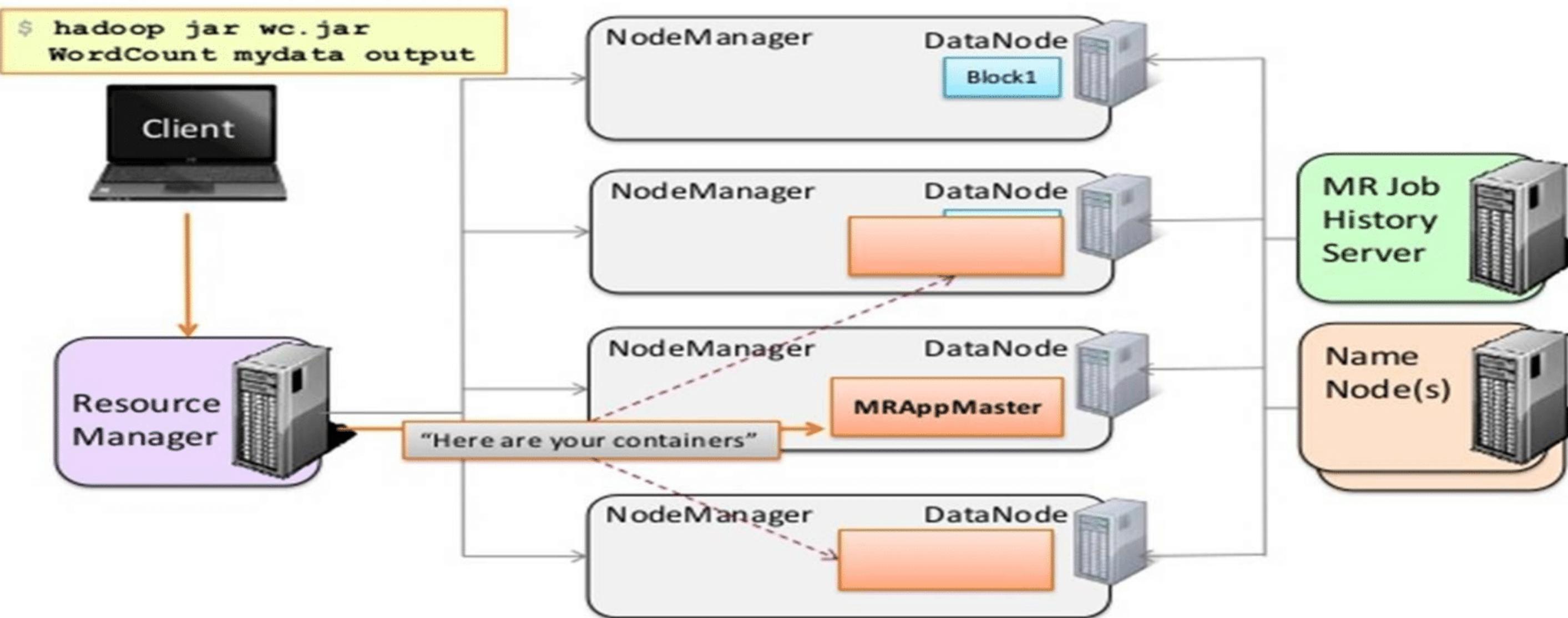
# Running a MapReduce Application



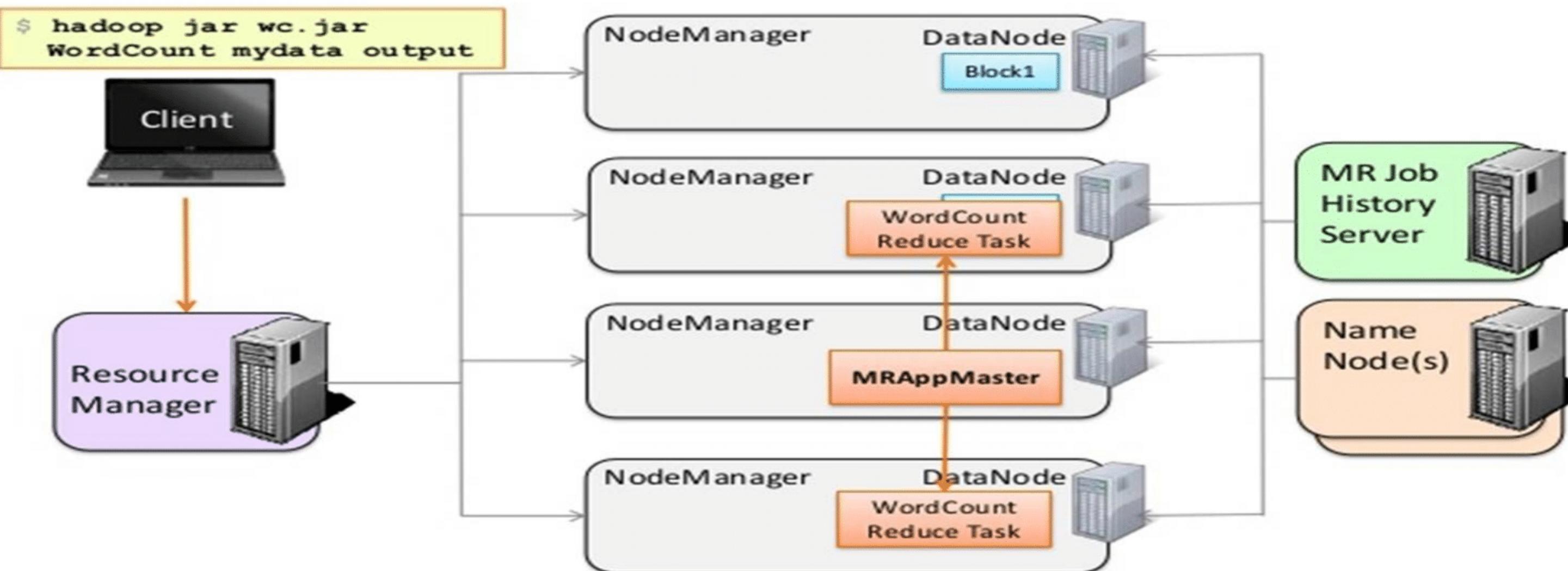
# Running a MapReduce Application



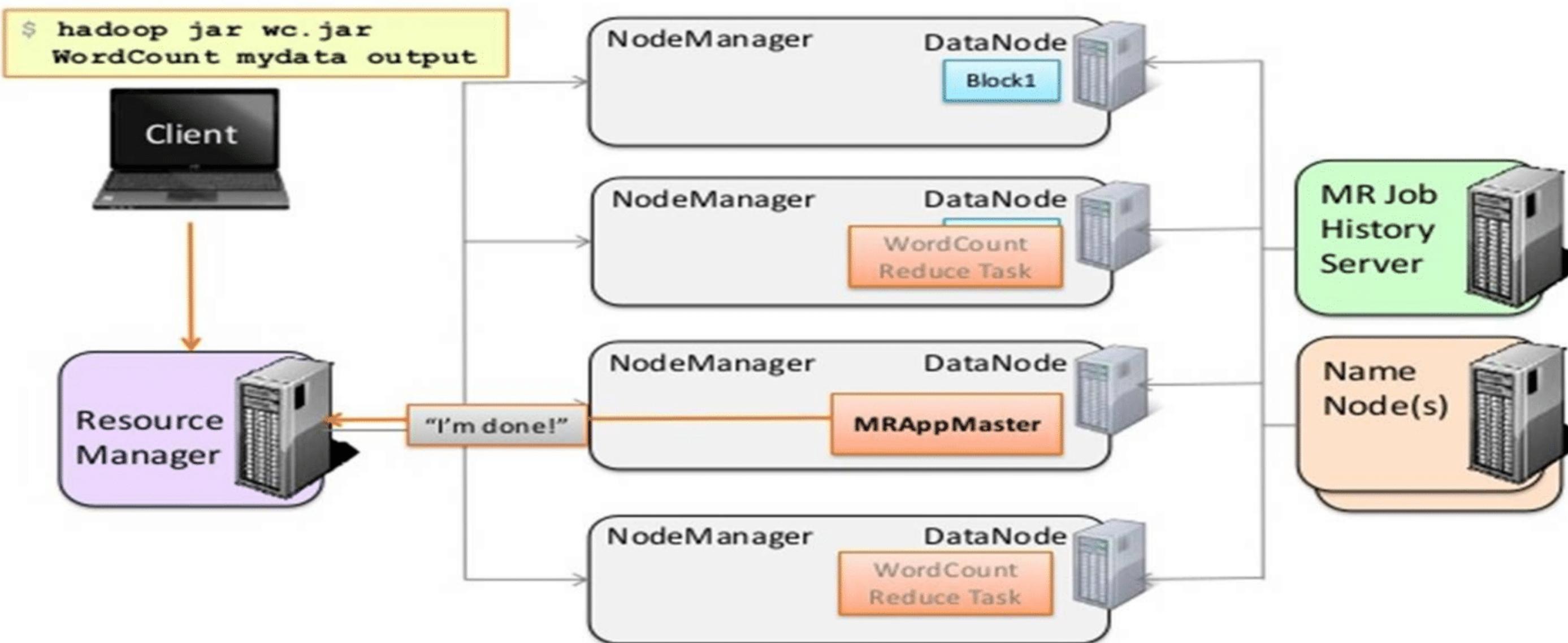
# Running a MapReduce Application



# Running a MapReduce Application



# Running a MapReduce Application



# Thank You

**DataFlair Web Services Pvt Ltd**

+91-8451097879 / +91-7718877477

[info@data-flair.com](mailto:info@data-flair.com)

<http://data-flair.com>

<https://www.facebook.com/DataFlairWS>