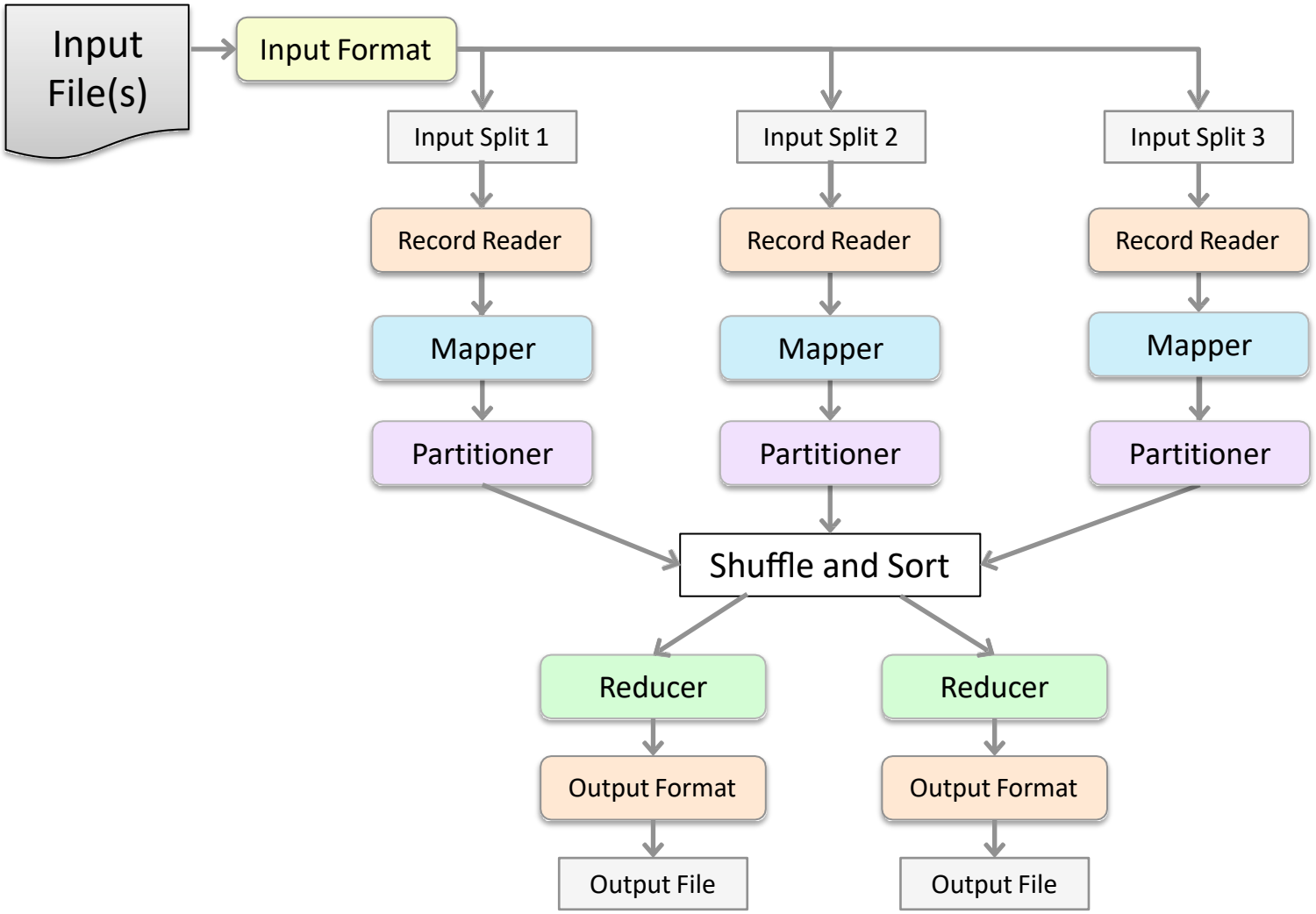
Understanding MapReduce on YARN

What Is MapReduce?

- **MapReduce is a programming model**
  - Neither platform- nor language-specific
  - Record-oriented data processing (key and value)
  - Facilitates task distribution across multiple nodes
- **Where possible, each node processes data stored on that node**
- **Consists of two developer-created phases**
  - Map
  - Reduce
- **In between Map and Reduce is the *shuffle and sort***
  - Sends data from the Mappers to the Reducers

MapReduce: The Big Picture



Features of MapReduce

- **Automatic parallelization and distribution**
- **Fault-tolerance**
- **Status and monitoring tools**
- **A clean abstraction for programmers**
  - MapReduce programs are usually written in Java
    - Can be written in other languages using *Hadoop Streaming*
- **MapReduce abstracts all the 'housekeeping' away from the developer**
  - Developer can concentrate simply on writing the Map and Reduce functions

MapReduce Concepts

- **Each Mapper processes a single *input split* from HDFS**
  - Often a single HDFS block
- **Hadoop passes one *record* at a time to the developer's Mapper code**
- **Each record has a *key* and a *value***
- ***Intermediate data* is written by the Mapper to local disk**
- **During the shuffle and sort phase, all the values associated with the same intermediate key are transferred to the same Reducer**
  - The developer specifies the number of Reducers
- **Reducer is passed each key and a list of all its values**
  - Keys are passed in sorted order
- **Output from the Reducers is written to HDFS**

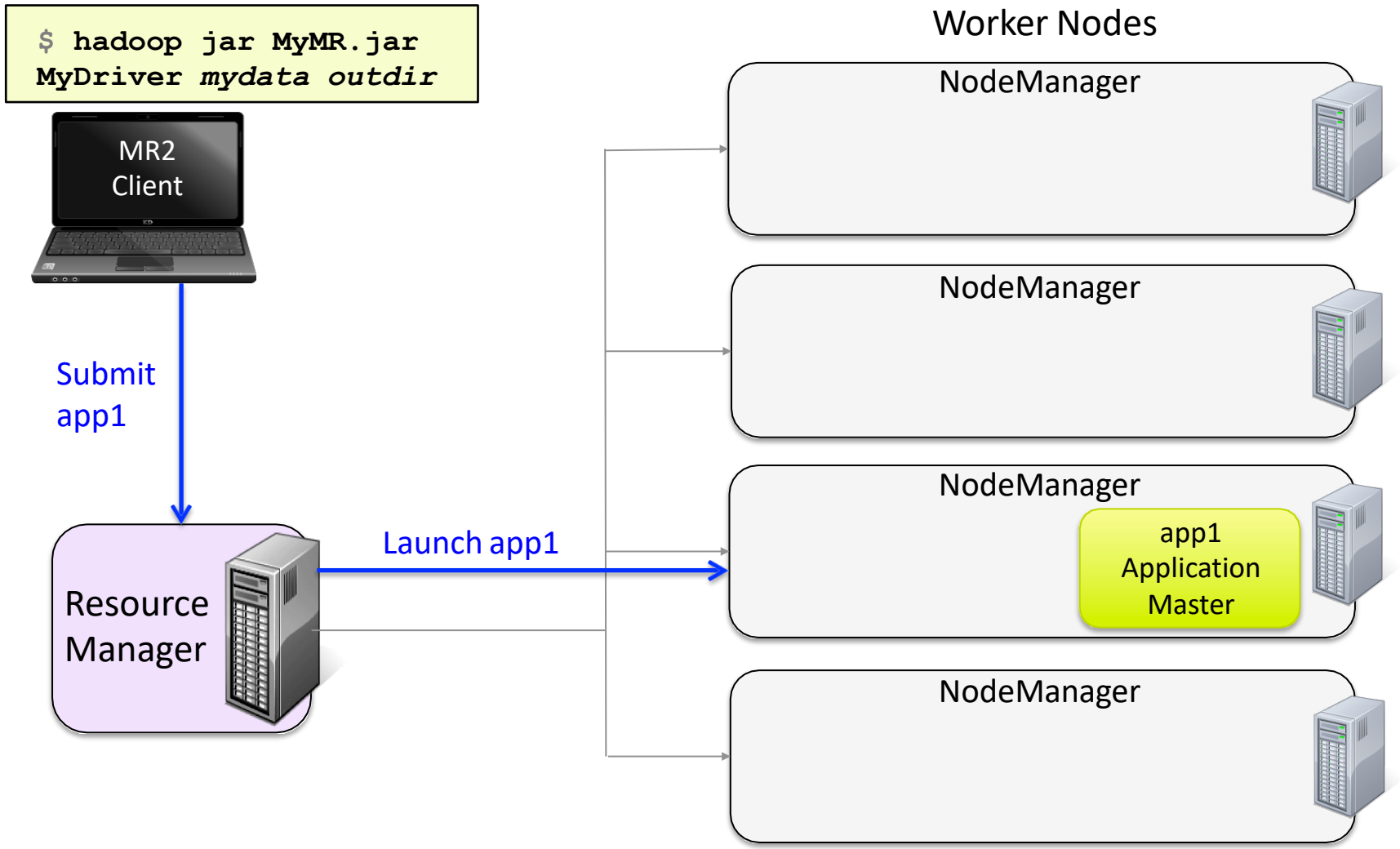
MapReduce Application Terminology

- **Job**
  - A Mapper, a Reducer, and a list of inputs to process
  - The highest-level unit of computation in MapReduce
- **Task**
  - An individual unit of work
  - A job consists one or more tasks
    - Each task is either a Map task or a Reduce task
  - Runs in a container on a worker node
- **Client**
  - The program that submits the job to the YARN ResourceManager
  - Sometimes refers to the machine on which the program runs

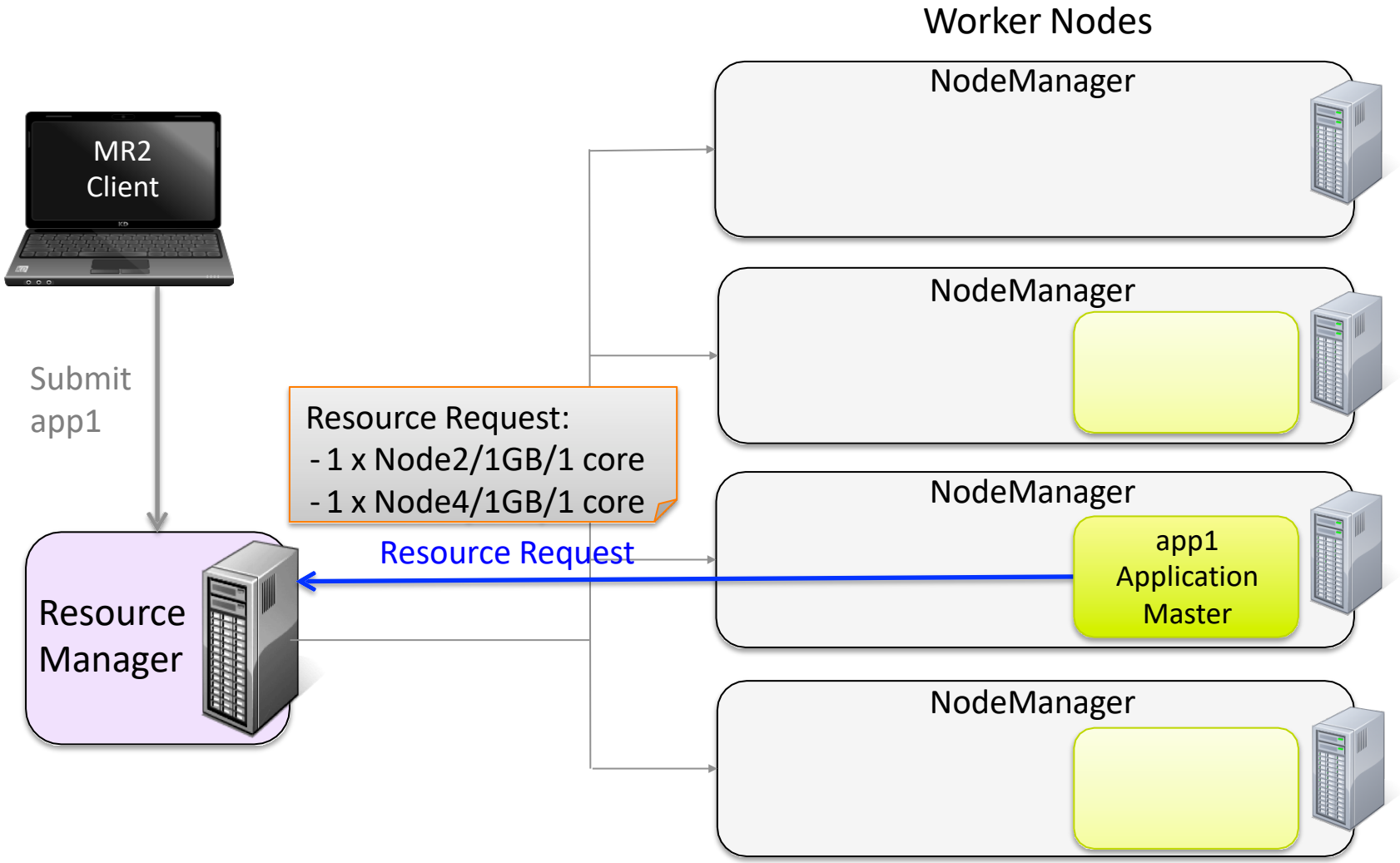
Deploying MapReduce

- **MapReduce Service – two versions available**
  - MapReduce v2
    - Recommended
    - MapReduce applications run as YARN applications
  - MapReduce v1
    - For backward compatibility
    - Does not use YARN
  - Both versions require HDFS

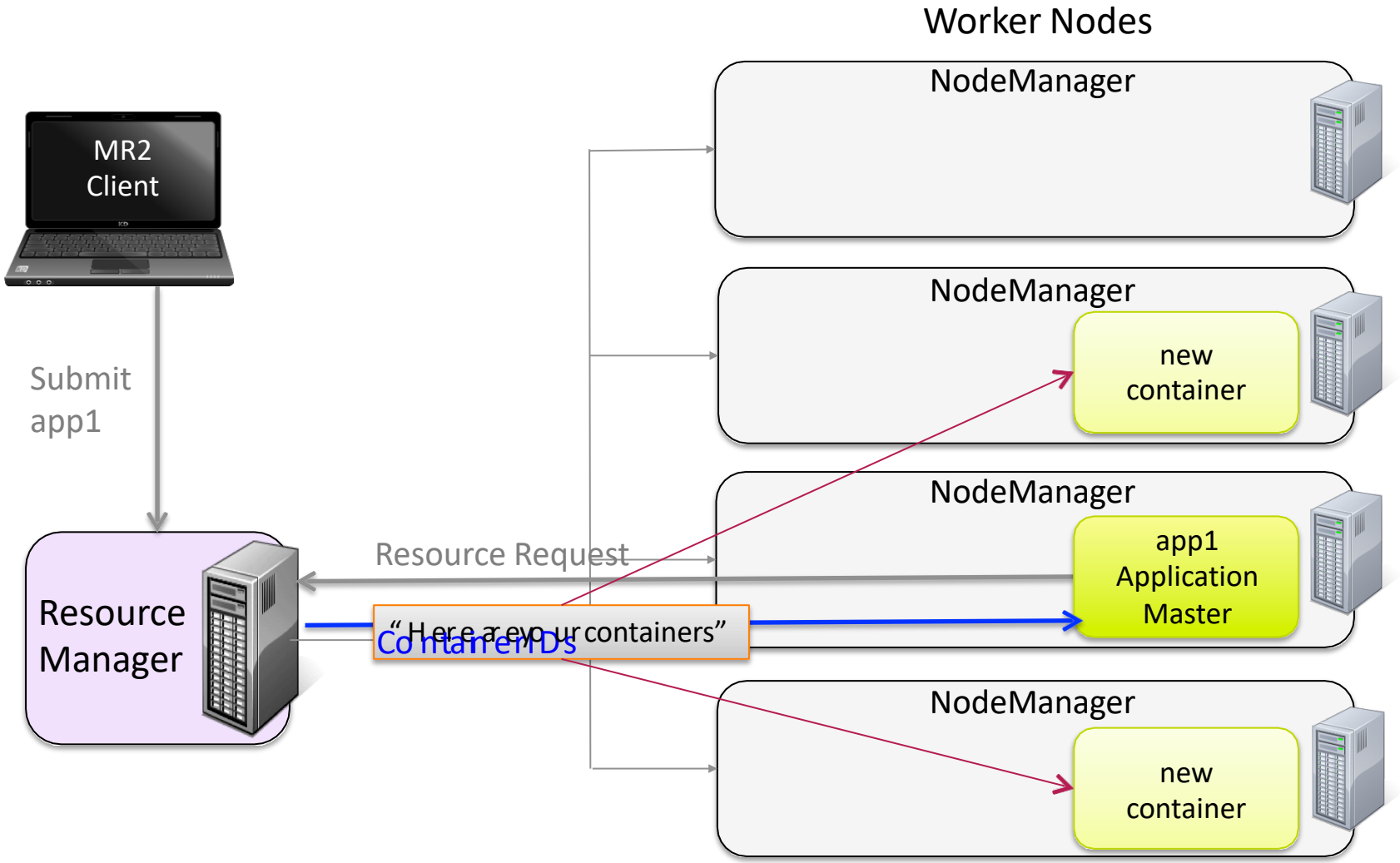
YARN: Submit a MapReduce Application



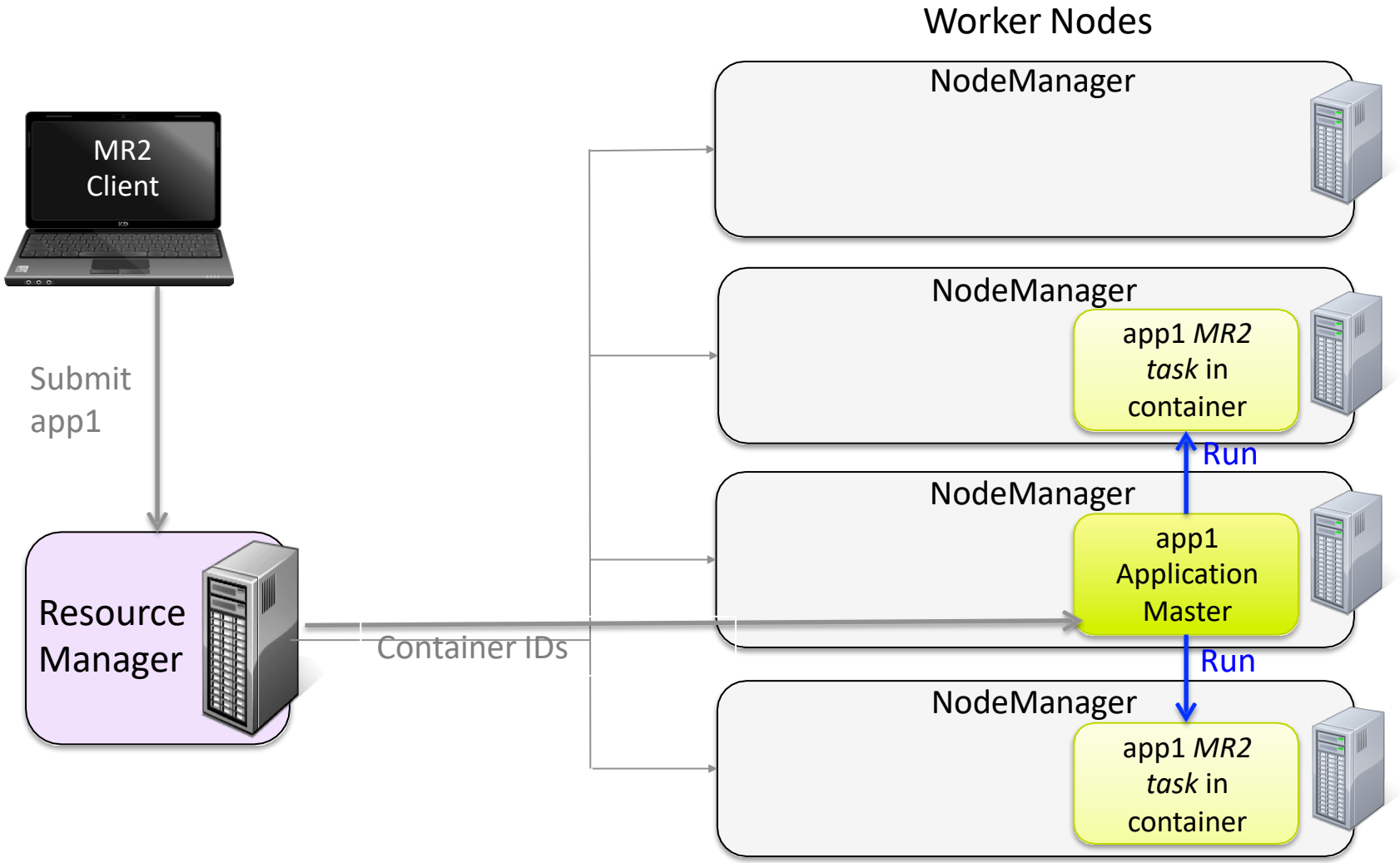
YARN: Application Master Resource Request



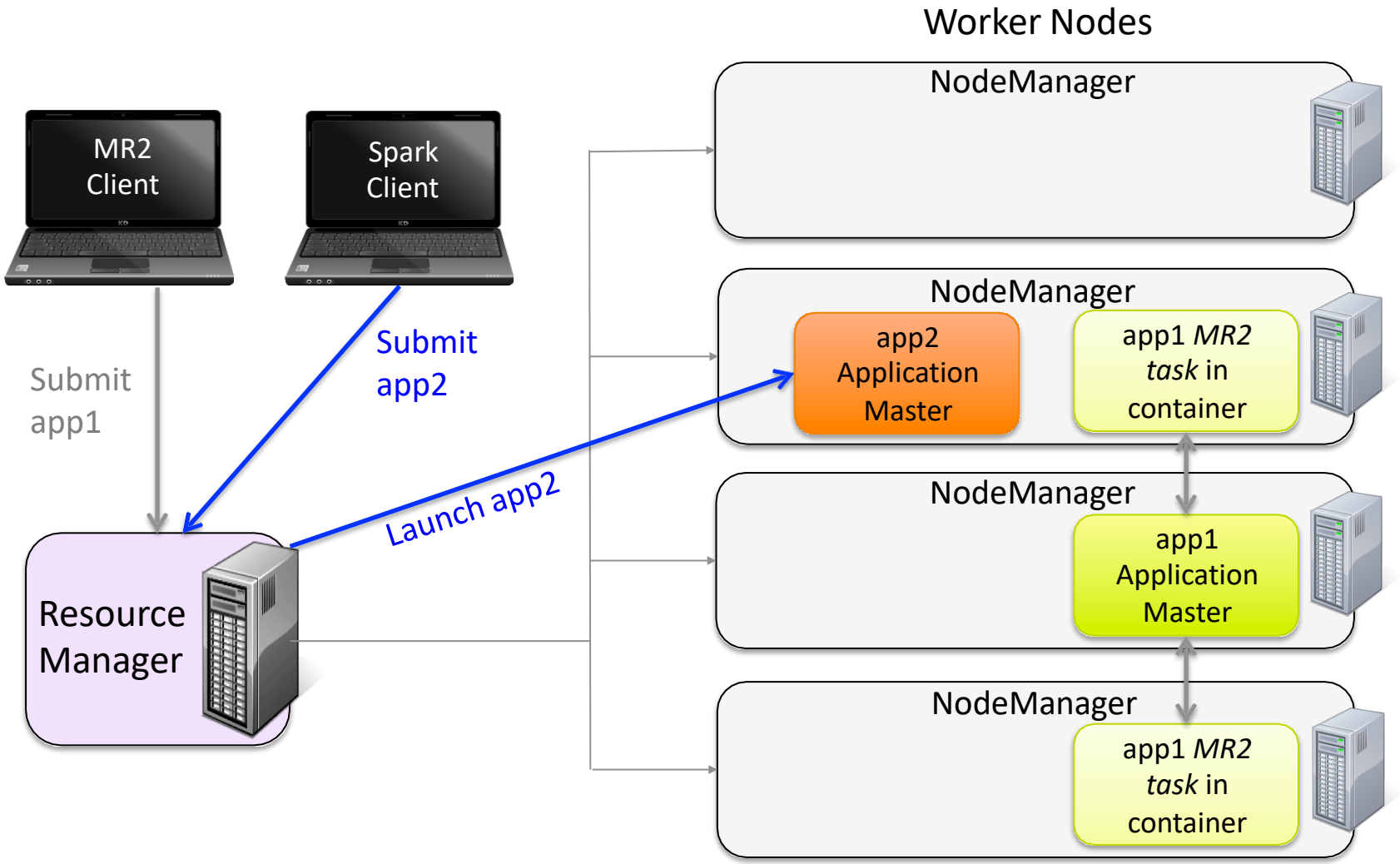
YARN: Container Allocation



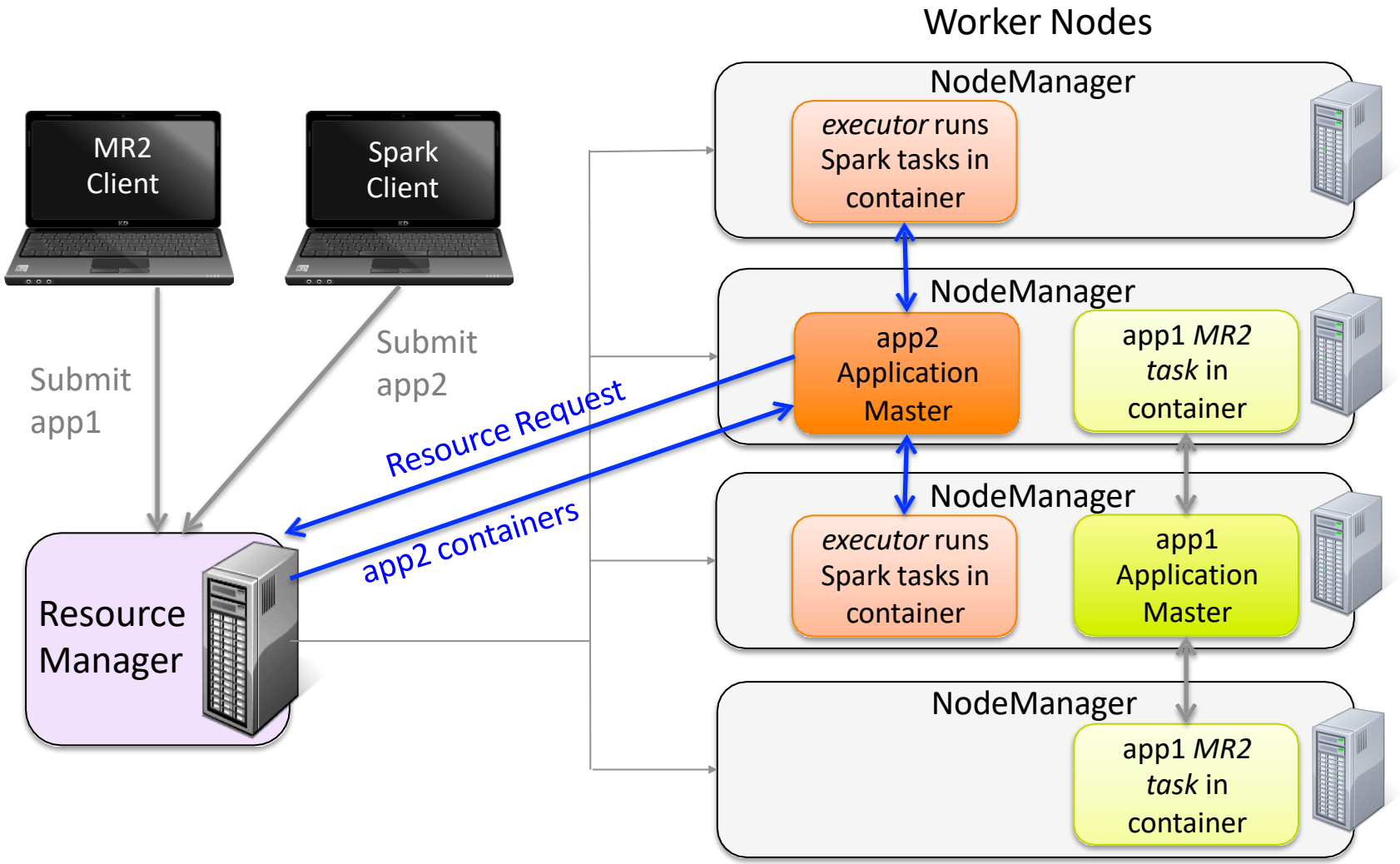
YARN: Application Containers



YARN: Submit a Spark Application



YARN: Containers



YARN Container Lifespan

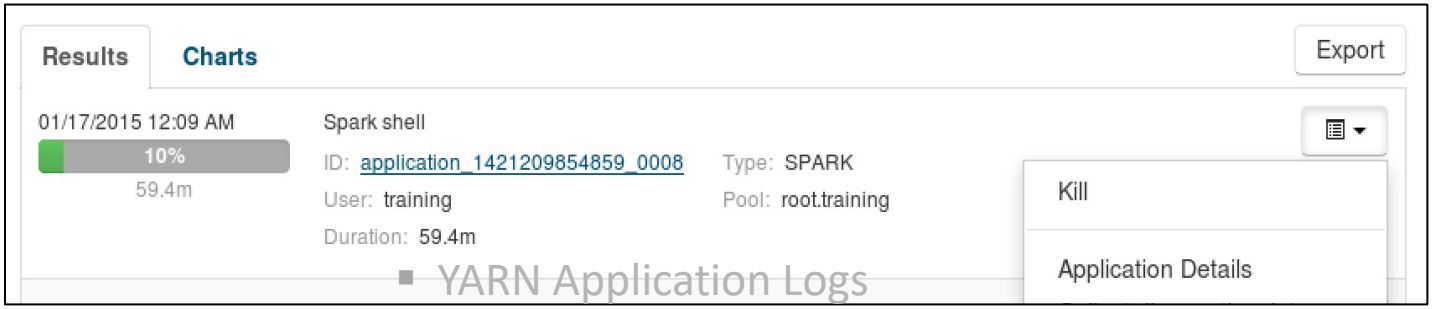
- **MapReduce's use of containers**
  - One container is requested and created *for each task* in a job
  - Each Map or Reduce task gets its own JVM that runs in a container
  - Each container is deleted once a task completes
- **Spark's use of containers**
  - One container is requested and created *for each executor* granted to the Spark application
  - An executor is a JVM
    - Many Spark tasks can run in a single executor – concurrently and over the lifetime of the container
  - Each container stays alive for the lifespan of the application

Discovering YARN Application Details in the Shell

- **Displaying YARN application details in the shell**
  - To view all applications currently running on the cluster
    - `yarn application -list`
    - Returns all running applications, including the application ID for each
  - To view all applications on the cluster, including completed applications
    - `yarn application -list all`
  - To display the status of an individual application
    - `yarn application -status <application_ID>`

Killing YARN Applications

- **To kill a running YARN application from Cloudera Manager**
  - Use the drop down menu for the application in the YARN Applications page for the cluster



- **To kill a running YARN application from the command line**
  - You *can not* kill it just by hitting CTRL-C in the terminal
    - This only kills the client that launched the application
    - The application is still running on the cluster!
  - Utilize the YARN command line utility instead
    - `yarn application –kill <application_ID>`

YARN Application Log Aggregation

- **Debugging distributed processes is intrinsically difficult**
  - Cloudera Manager's log aggregation improves this situation
- **YARN provides application log aggregation services**
  - Recommended (and enabled by default by Cloudera Manager)
  - Logs are aggregated by application
  - Access the logs from Cloudera Manager, any HDFS client, or the shell
- **When YARN log aggregation is enabled:**
  - Container log files are moved from NodeManager hosts' `/var/log/hadoop-yarn/container` directories to HDFS when the application completes
    - Default HDFS directory: `/tmp/logs`
- **Aggregated YARN application log retention**
  - Apache default: log retention disabled (and infinite when enabled)
  - Cloudera Manager default: enabled and 7 day retention

MapReduce Log Details

- **MapReduce jobs produce the following logs:**
  - ApplicationMaster log
  - stdout, stderr, and syslog output for each Map and Reduce task
  - Job configuration settings specified by the developer
  - Counters
- **MapReduce Job History Server log directory**
  - Default: `/var/log/hadoop-mapreduce`

Spark Log Details

- **Spark can write application history logs to HDFS**
  - Default: enabled
- **Spark application history location**
  - Default location in HDFS: `/user/spark/applicationHistory`
  - Find and view the logs from the command line:
    - `$ sudo –u hdfs hdfs dfs /user/spark/\ applicationHistory`
    - `$ sudo –u hdfs hdfs dfs –cat /user/spark/\ application_<application_id>/EVENT_LOG_1`
- **Spark History Server log directory**
  - Default: `/var/log/spark`

Accessing YARN Application Logs From the Command Line

- **Accessing application logs from the command line allows you to use utilities like `grep` for quick log analysis**
- **First find the application ID with `yarn application -list`**

```
$ yarn application -list -appStates FINISHED | grep 'word count'
application_1392918622651_0004              word count                MAPREDUCE
       training     root.training            FINISHED           SUCCEEDED
100% http://monkey:19888/jobhistory/job/job_1392918622651_0004
```

- **Then, once you have the application ID, you can list its logs**

```
$ yarn logs -applicationId application_1392918622651_0004
...
Container: container_1392918622651_0004_01_000002 on elephant_46591
==========================================================
...
LogType: syslog
LogLength: 4085
Log Contents:
2014-02-20 16:59:20,462 WARN [main]
org.apache.hadoop.conf.Configuration: job.xml:an attempt to override
final parameter: mapreduce.job.end-  notification.max.retry.interval;
                                  Ignoring.
```

Lab: YARN Job on Hadoop Cluster – 60 Minutes