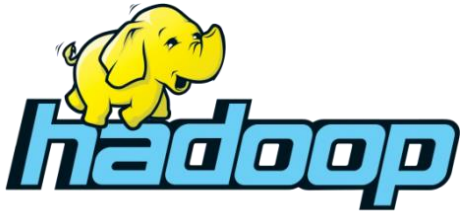

BIG DATA
HADOOP
ENTERPRISE
ANALYTICS
OPEN SOURCE
TECHNOLOGIES
BUSINESS
SECURITY
MACHINE
PRIVACY
LEARNING
RELATIONAL
CHIEF DATA OFFICER
INSIDE DATA
TOOLS
LEADS
MANAGER
RISK
ANALYSIS
SOLUTIONS
MARKETING
PROCESSING
SUPPORT
HPC
BUYS
SYSTEMS
SPACE
SOLVED
PROJECTS
CREDIBILITY
COMMUNITY
THINGS
MAJORITY
OUTSIDE DATA
DATA BLENDING
THINK BIG ANALYTICS
INFORMATION
INVESTMENT
FEELINGS
IN MEMORY
ACHIEVING
STANDARD
ADOPTION
DATA SCIENCE
SOFTWARE
NUMBER
SQL ON HADOOP
DECISION MAKING
CLICKSTREAM
ETL
INSIGHTS
URGENCY
IGNORANCE
DEFINING
BIGGER
CROWDED
ADVANCED
PENTAHO
UPGRADE
AS A SERVICE
WEB BASED
SHORTAGE
IMPACT
REFERENCE
ORGANIZATIONS
HYPOTHESIS
XPLENTY
INVOLVED
PREDICTIVE
COMPETITIVE
INNOVATION
COMBINE
TAKING
COMPLEX
MOVING
INJECT
ENVIRONMENT
ALPINE DATA LABS
SCIENTIST
BIGGEST
REFINE
OFFLINE
ORIGINAL
REAL TIME
STUDENTS
PREFERENCES
LITTLE DATA
RISK MANAGEMENT
TOTAL COST OF OWNERSHIP
DATA CENTRIC
UNSTRUCTURED QUERY LANGUAGE
HIGH SCHOOL
COMPLIANCE
MITIGATED
PROMINENT
SIMPLIFY
FORCED
RETAIL
SEARCH
SUCCESS
SCIENCE
OFFLOAD
VENDORS
FUTURE
FINANCE
PROMISE
CHEAPER
DATA HUB
DRAMATIC
INITIATIVES
ANALYZED
UNIVERSE
EXACTLY
PERFORM
CURRICULUM
ACQUISITIONS
DISAPPOINTMENT
UPSELL
ACTING
INDIVIDUAL
CONCERNS
DEVELOPERS
ADDITIONAL
ISSUES
EFFECT
DIVERSE
PRODUCE
PUNDITS
TASKING
LEADER
FREQUENCY
FAILED
OPPORTUNITIES
PRINCIPLES
INCREASINGLY
INFRASTRUCTURE

Apache Hadoop

Experis™ Academy
ManpowerGroup

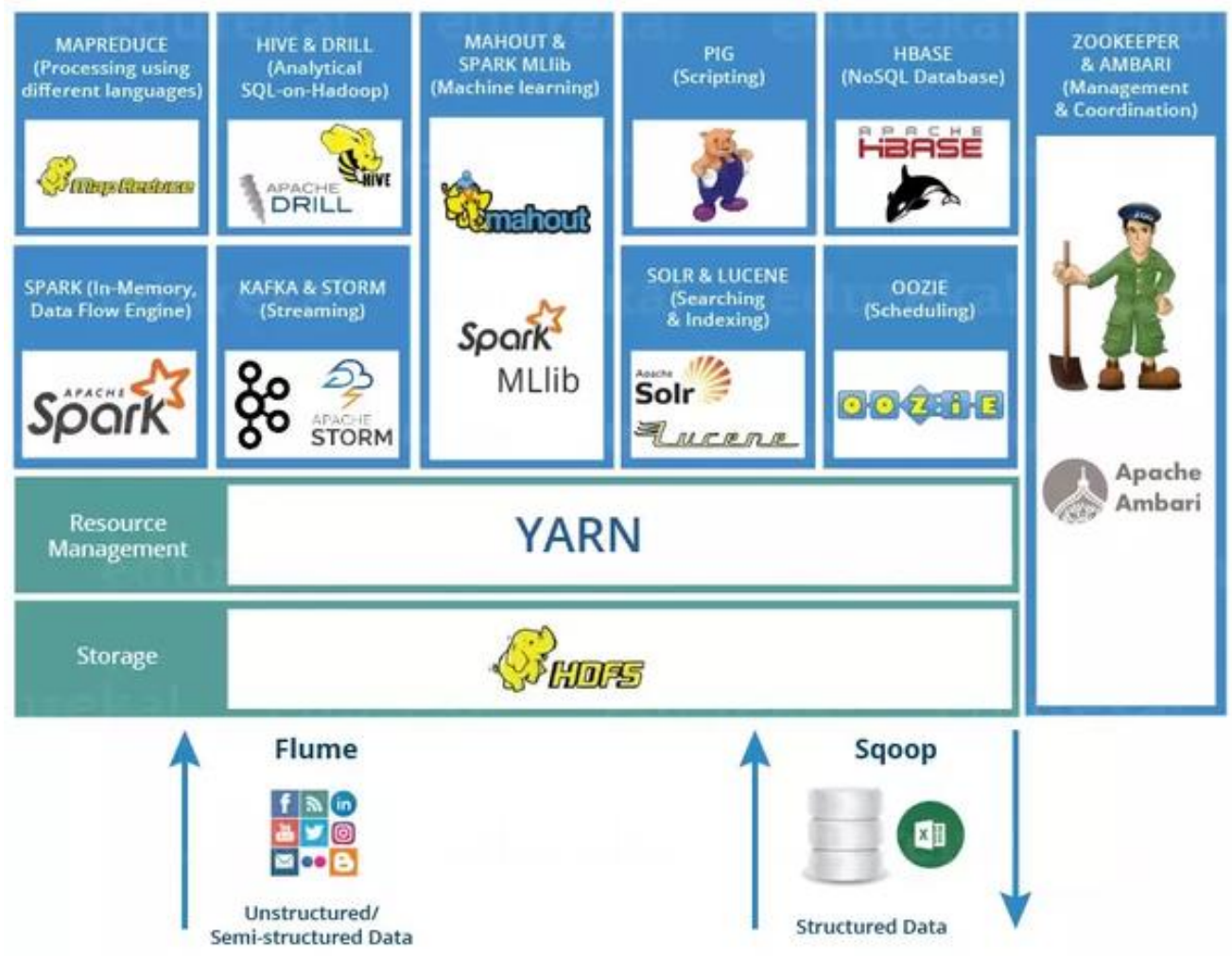
- Hadoop is an open source framework that facilitates using a network of many computers to solve problems involving massive amounts of data and computation
- Hadoop consists of two components:
  - Hadoop Distributed File System (**HDFS**)
  - Distributed computing engine that lets you implement and run programs as **MapReduce** jobs
- The Hadoop infrastructure takes care of all complex aspects of distributed processing: parallelization, scheduling, resource management, inter-machine communication, handling software and hardware failures
- Thanks to this clean abstraction, implementing distributed applications that process terabytes of data on hundreds of machines has never been so easy



Apache Hadoop

- A wide variety of companies and organizations use Hadoop for both research and production
- For example, in Yahoo More than 40,000 computers running Hadoop
  - The biggest cluster contains 4500 nodes
  - Used to support research for Ad Systems and Web Search

Hadoop EcoSystem

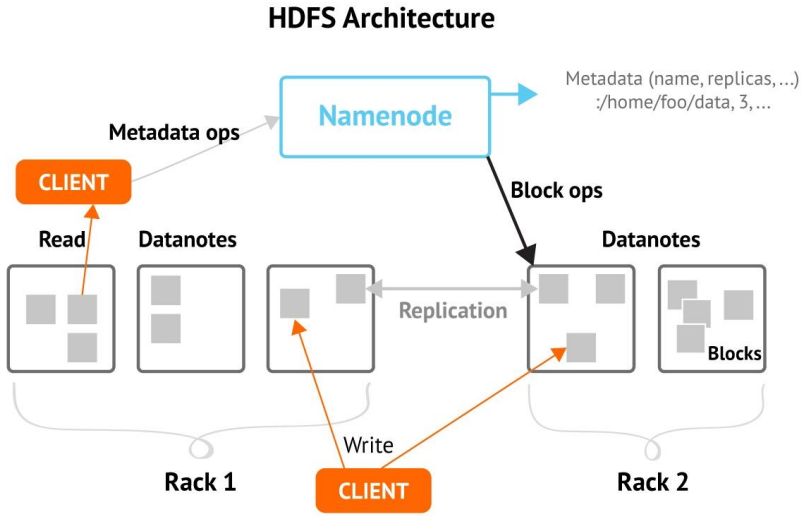


Distributed File Systems

- A distributed file system is similar to a normal file system, except that it runs on multiple servers at once
- Because it's a file system, you can do almost all the same things you'd do on a normal file system, such as storing, reading, and deleting files and adding security to files
- Distributed file systems have significant advantages:
  - They can store files larger than any one computer disk
  - Files get automatically replicated across multiple servers for redundancy or parallel operations while hiding the complexity of doing so from the user
  - The system can scale horizontally (by adding small servers to the network) instead of vertically (moving everything to a stronger server with more storage and better CPUs)
- The best-known distributed file system is the **Hadoop File System** (HDFS)
  - It is an open source implementation of the Google File System (GFS)

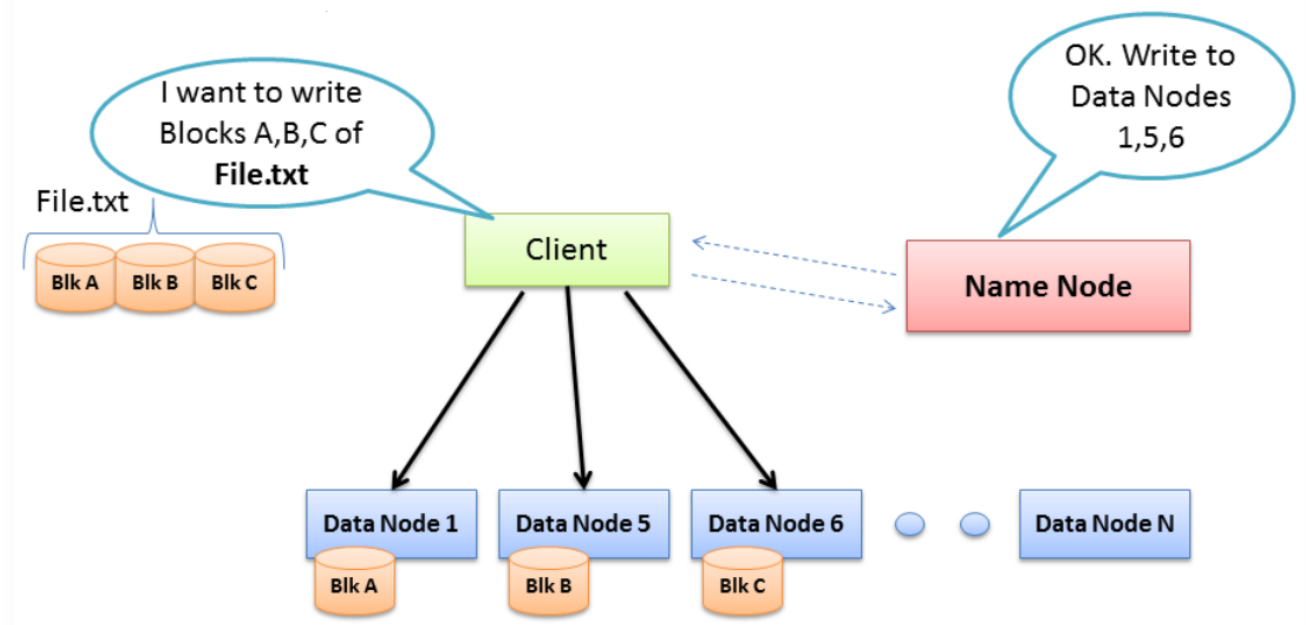
HDFS Architecture

- The master (**NameNode**) manages the file system namespace operations like opening, closing, and renaming files and directories and determines the mapping of blocks to DataNodes along with regulating access to files by clients
- The slaves (**DataNodes**) are responsible for serving read and write requests from the file system's clients along with perform block creation, deletion, and replication upon instruction from the Master (NameNode)



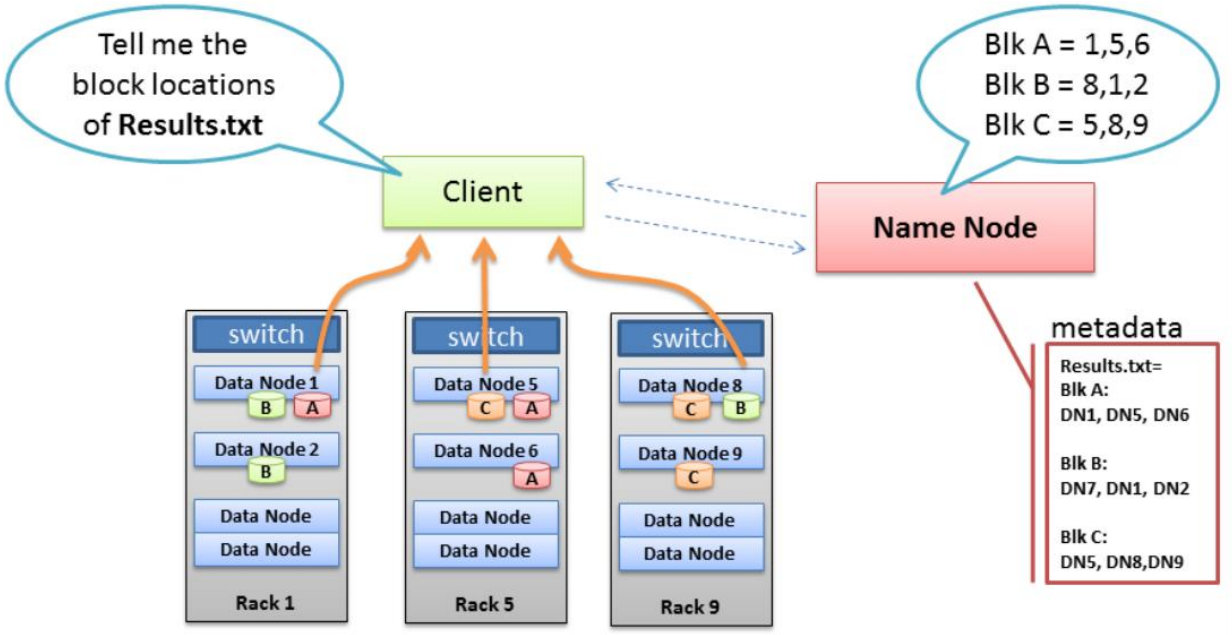
Writing a File to HDFS

- When a client wants to write a file to HDFS:
  - The client asks the NameNode for a list of data nodes that will store the file blocks
  - The client writes the blocks to the DataNodes
  - The DataNodes replicate the block
- The standard setting for Hadoop is to have 3 copies of each block in the cluster



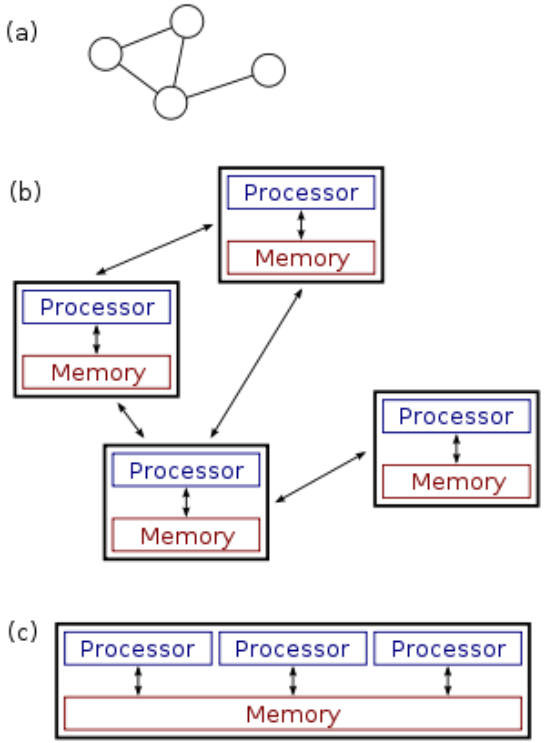
Reading a File from HDFS

- When a Client wants to retrieve a file from HDFS:
  - It consults the NameNode and asks for the block locations of the file
  - The NameNode returns a list of each DataNode holding a block, for each block
  - The client picks a DataNode from each block list and reads one block at a time with TCP on port 50010 (the default port number for the Data Node daemon)



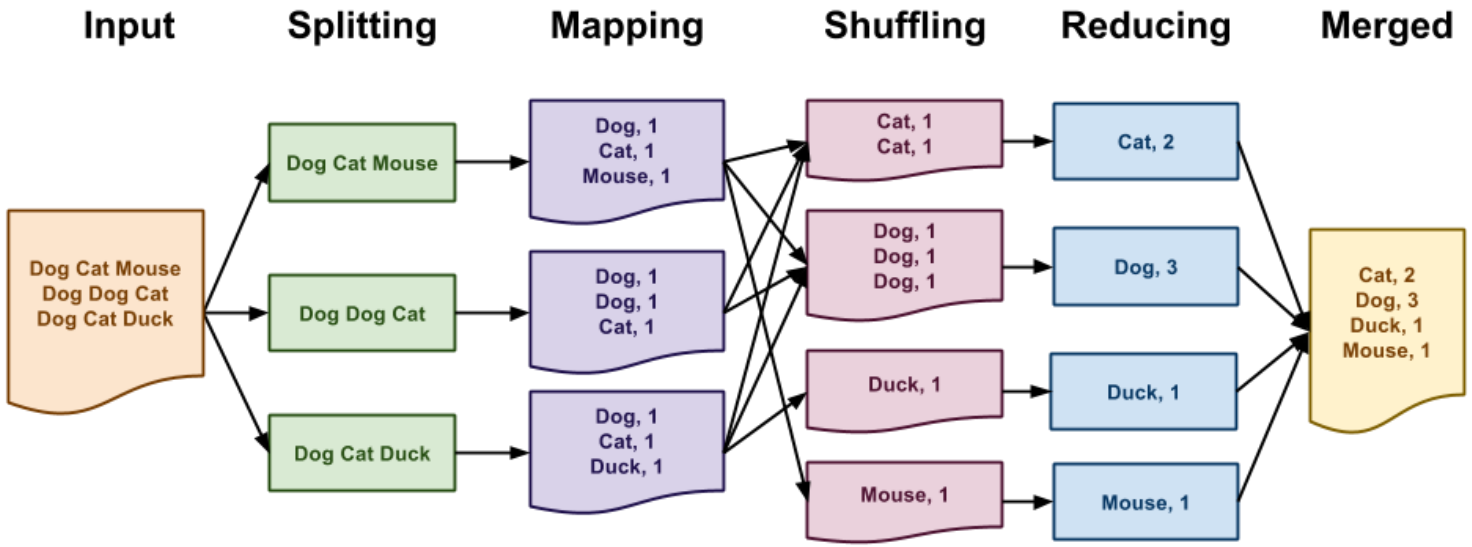
Distributed Computing

- In **distributed computing**, a problem is divided into many tasks, each of which is solved by a single computer
  - The computers communicate with each other via message passing



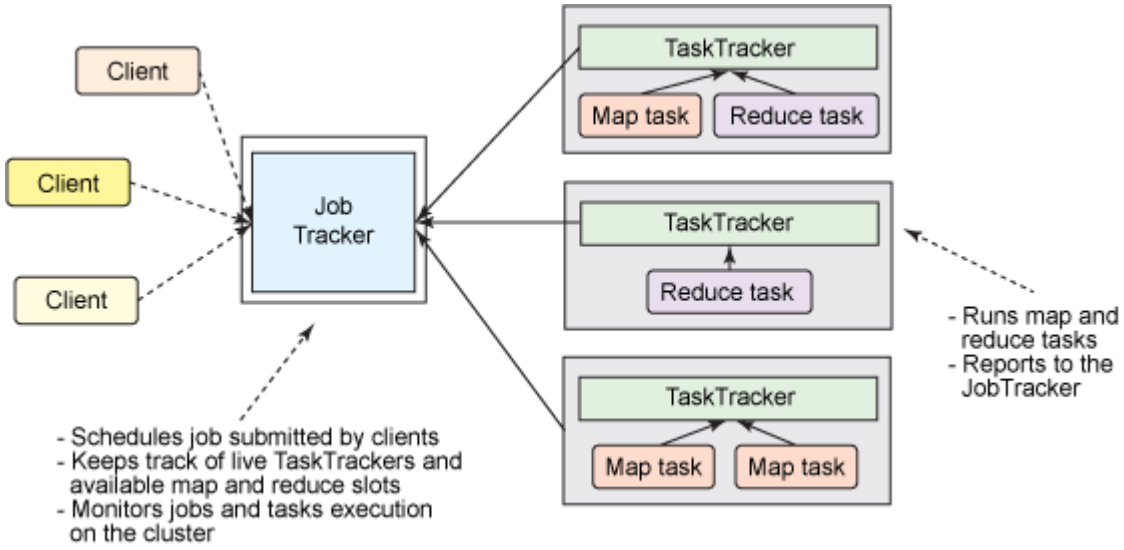
MapReduce

- MapReduce is a simple functional programming operation and it can be applied, in parallel, to gigabytes or terabytes of data
- *Map tasks* invoke map functions over subsets of input data
- After they are done, *reduce tasks* start calling reduce functions on the intermediate data, generated by map functions, to produce the final output



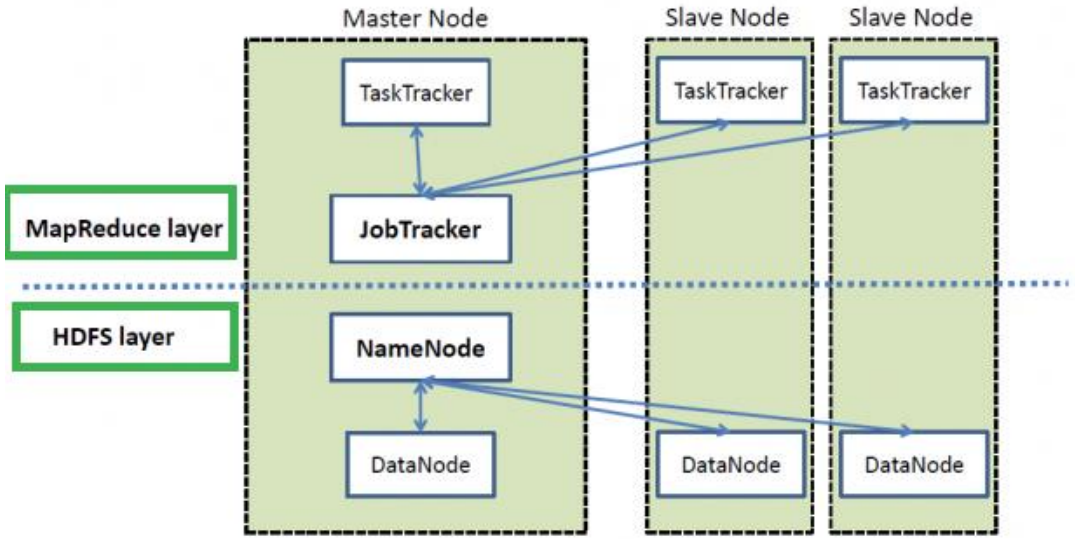
Hadoop MapReduce

- In the classical Hadoop MapReduce framework (MRv1), the job execution is controlled by two types of processes:
  - A single master process called **JobTracker**, which coordinates all jobs running on the cluster and assigns map and reduce tasks to run on the TaskTrackers
  - A number of subordinate processes called **TaskTrackers**, which run assigned tasks and periodically report the progress to the JobTracker



Hadoop MapReduce

- TaskTracker instances are typically deployed on the same servers that host DataNode instances, so that MapReduce operations are performed close to the data

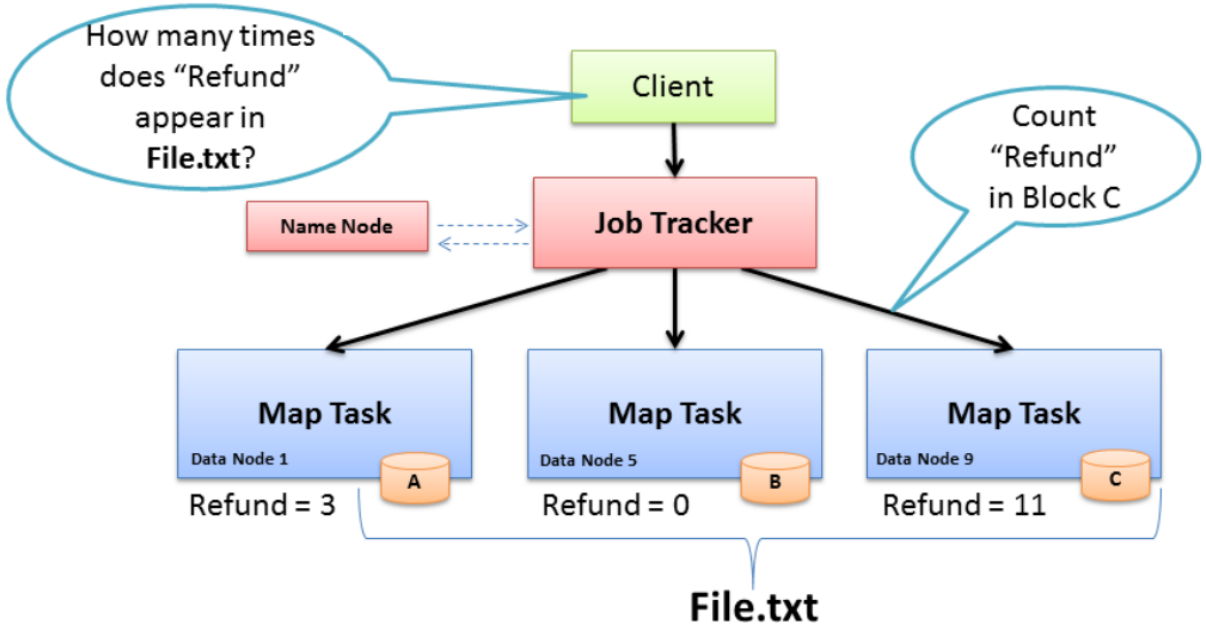


Typical Workflow

- Load data into the cluster (HDFS writes)
- Analyze the data (Map Reduce)
- Store results in the cluster (HDFS writes)
- Read the results from the cluster (HDFS reads)

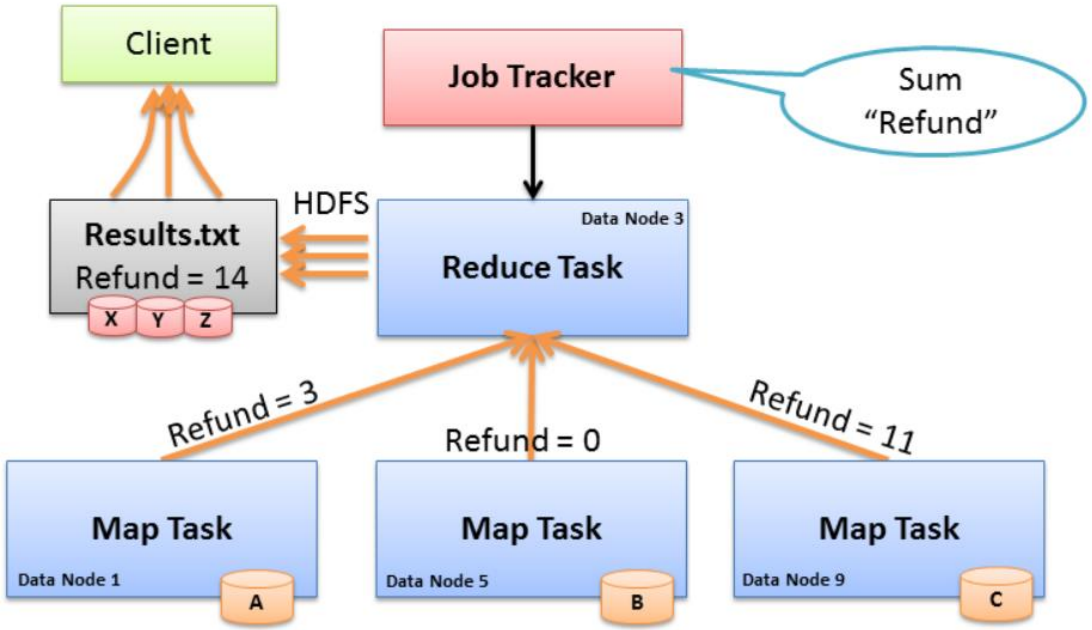
Data Processing: Map

- The Client machine submits the Map Reduce job to the Job Tracker
- The Job Tracker consults the Name Node to learn which Data Nodes have blocks of the file
- The Job Tracker then provides the Task Trackers running on those nodes with the Java code required to execute the Map computation on their local data
- Each node stores the result of its local computation in temporary local storage



Data Processing: Reduce

- The Job Tracker starts a Reduce task on any one of the nodes in the cluster
- The Reduce task grabs the intermediate data from all of the completed Map tasks.
- The Reducer summarizes the results from all the map tasks (e.g., sums up the total occurences of the word “Refund”) and writes the final result to HDFS



Limitations of Classical MapReduce

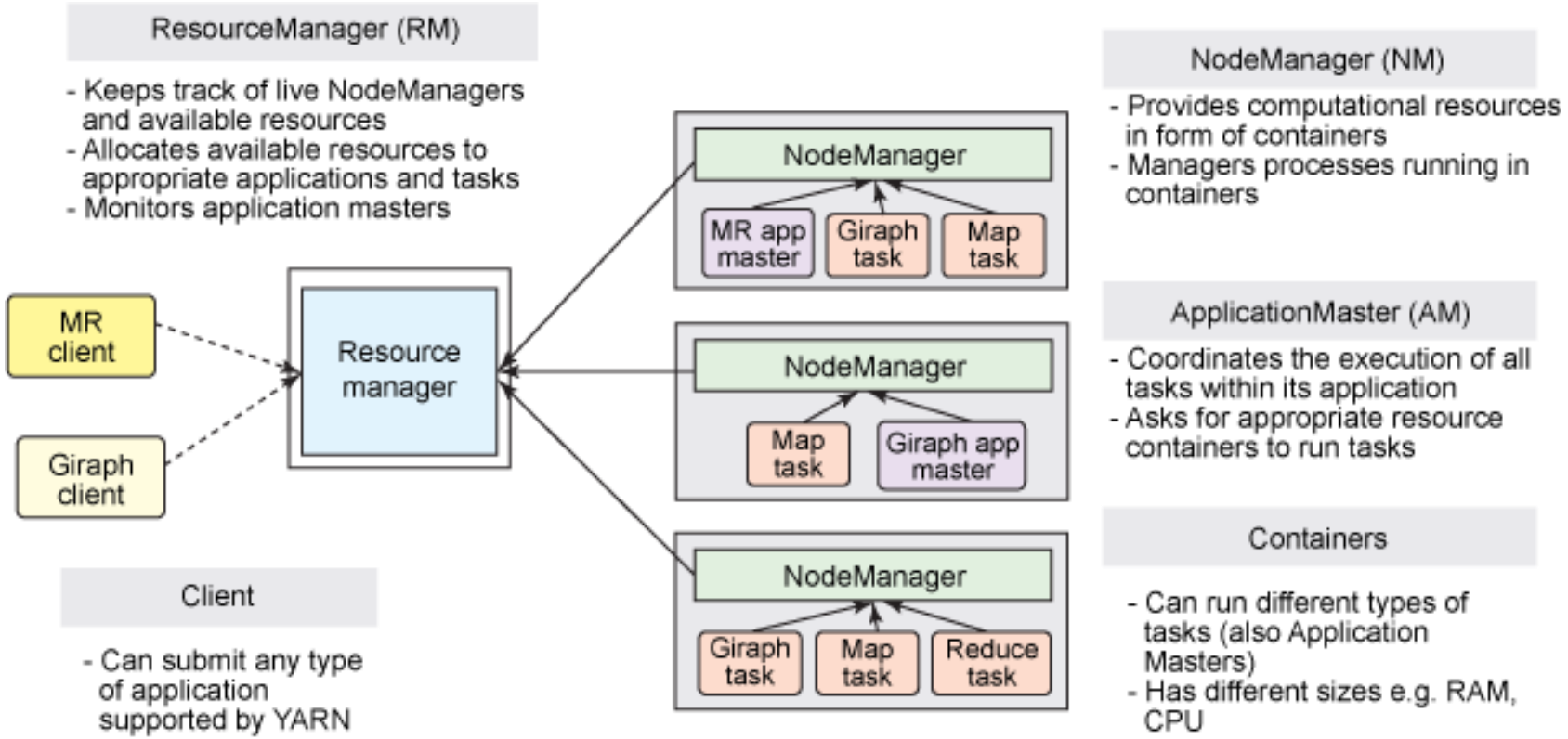
- The large Hadoop clusters revealed a limitation involving a scalability bottleneck caused by having a single JobTracker
- A single JobTracker had to constantly keep track of thousands of TaskTrackers, hundreds of jobs, and tens of thousands of map and reduce tasks
- According to Yahoo!, the practical limits of such a design are reached with a cluster of 5,000 nodes and 40,000 tasks running concurrently
- Due to this limitation, smaller and less-powerful clusters had to be created and maintained
- In addition, Hadoop was designed to run MapReduce jobs only
- With the advent of alternative programming models (such as graph processing), there was a need to support programming paradigms that could run on the same cluster and share resources in an efficient and fair manner

Hadoop YARN

Experis™ Academy
ManpowerGroup

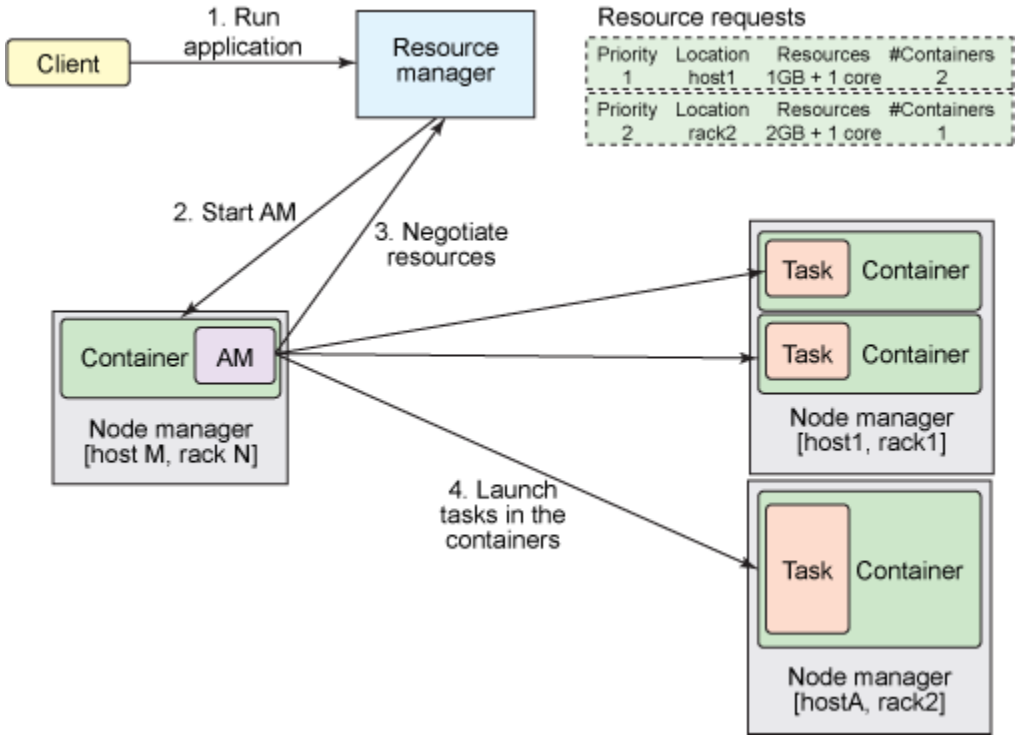
- YARN (Yet Another Resource Neogetiator) is the resource management and job scheduling technology used in Hadoop from version 2.0
- YARN offers clear advantages in scalability, efficiency, and flexibility compared to the classical MapReduce engine in the first version of Hadoop
- The YARN-based architecture is not constrained to MapReduce
- The following name changes give a bit of insight into the design of YARN:
  - ResourceManager instead of a cluster manager
  - ApplicationMaster instead of a dedicated and short-lived JobTracker
  - NodeManager instead of TaskTracker
  - A distributed application instead of a MapReduce job

YARN Architecture



Application Submission in YARN

- The following diagram shows how the ResourceManager, ApplicationMaster, NodeManagers, and containers interact together when an application is submitted to a YARN cluster



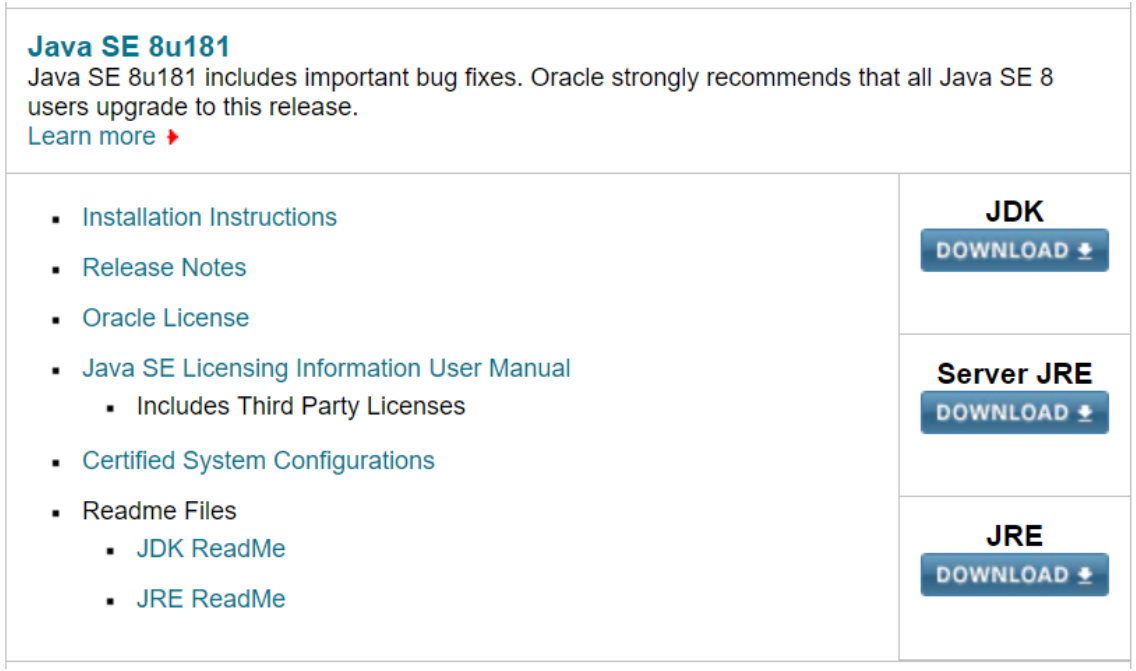
Hadoop Deployment Methods

Experis™ Academy
ManpowerGroup

- **Standalone**: In this mode, there are no daemons running, everything runs as in a single JVM. This mode is suitable for running the MapReduce program during development, as it is easy to test and debug.
- **Pseudo-distributed**: The Hadoop daemon process runs on a local machine simulating a cluster on a small scale.
- **Fully distributed**: Hadoop runs on a cluster of machines providing a production environment.
- Client machines have Hadoop installed with all the cluster settings, but are neither a Master or a Slave

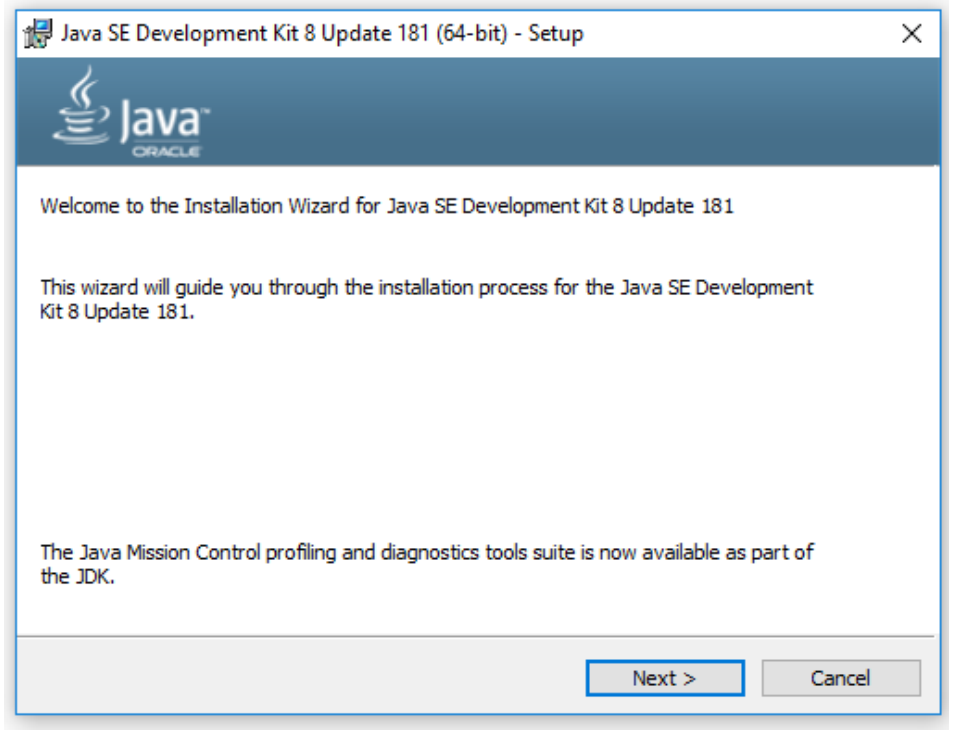
Install Java

- Running Hadoop requires Java
- Download JDK (Java Development Kit) 8 from http://www.oracle.com/technetwork/java/javase/downloads/index.html
- Note: JDK 10 is still not supported by current version of Hadoop

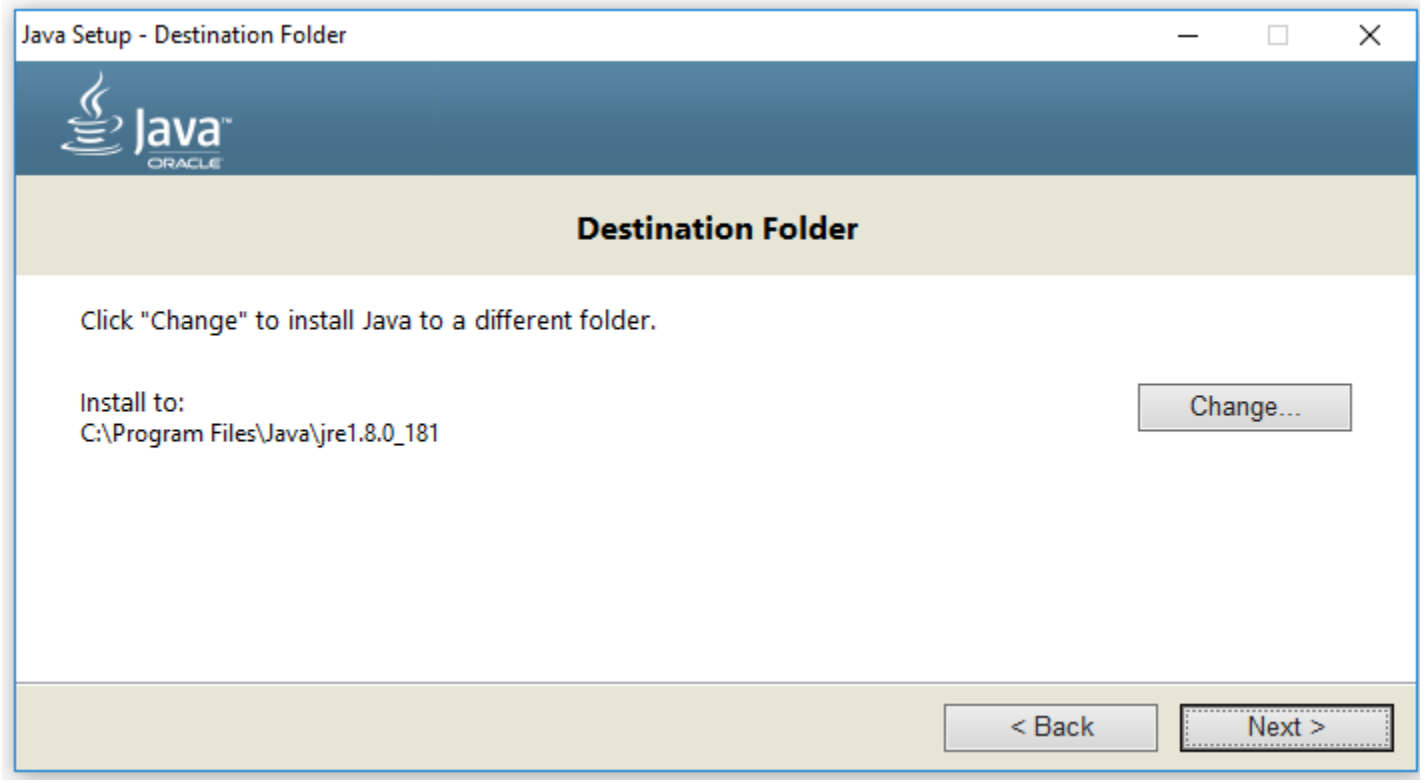


Install Java

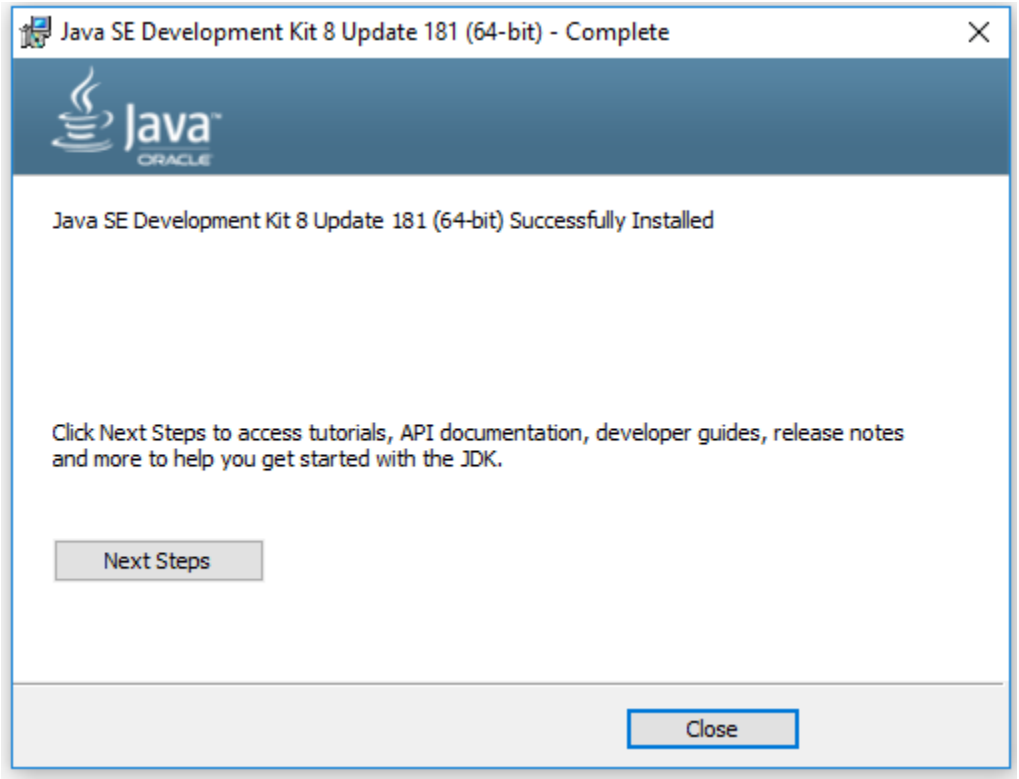
- Double-click the file jdk-8u181-windows-x64.exe to start installation



Install Java

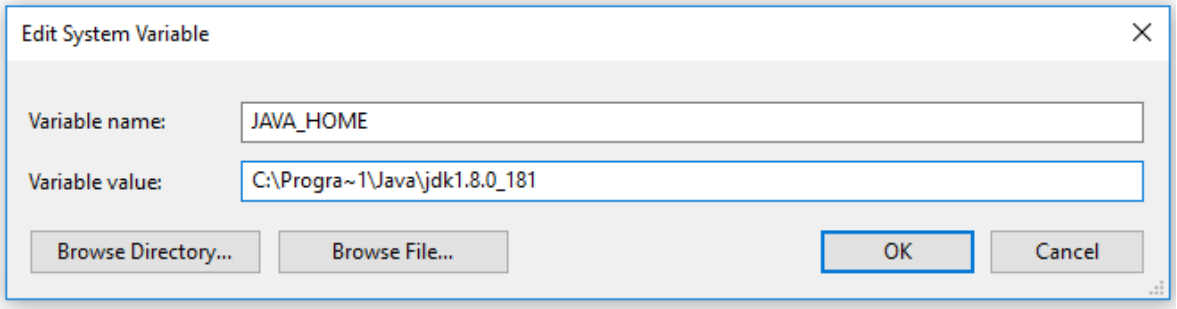


Install Java



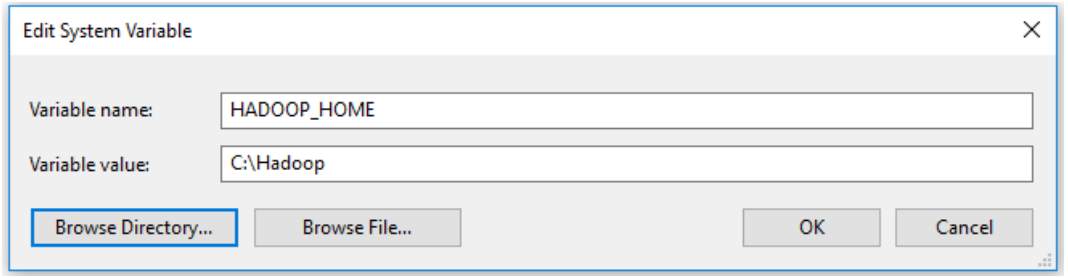
Install Java

- Set the JAVA_HOME environment variable
  - In the **Variable Name** field, enter JAVA_HOME
  - In the **Variable Value** field, enter your JDK installation path
    - Write Progra~1 instead of 'Program Files' (hadoop has some issues with spaces)



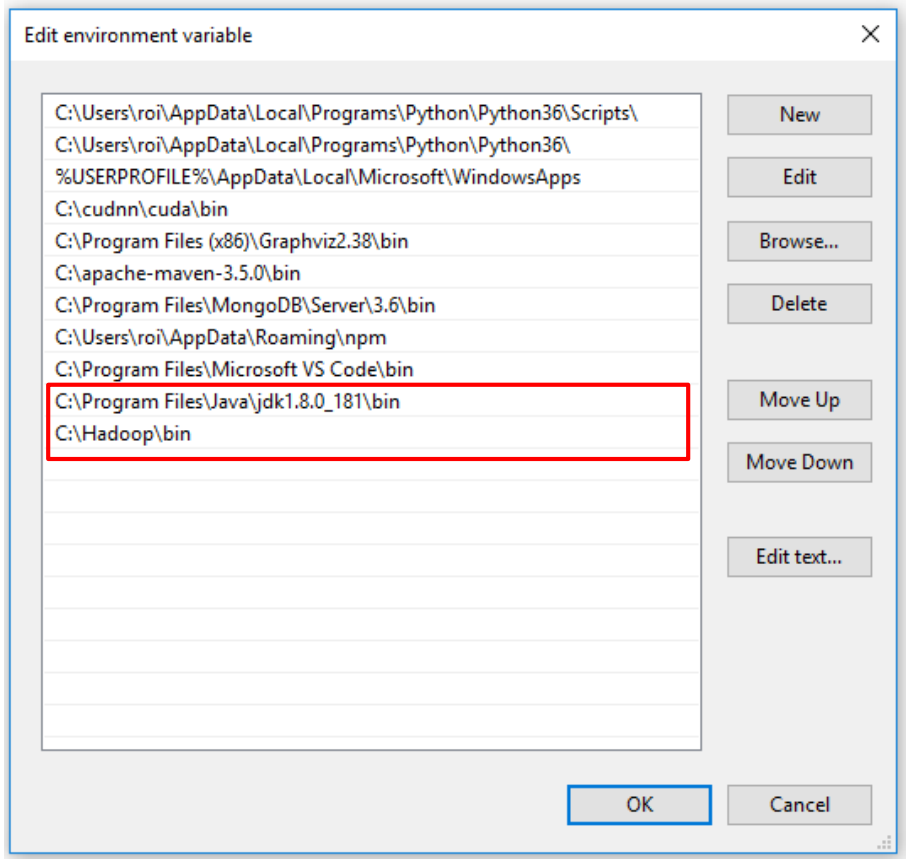
Install Hadoop

- Download the latest Hadoop binary from http://hadoop.apache.org/releases.html
- Extract the downloaded file (hadoop-3.0.3.tar.gz) into the folder C:\Hadoop
- You might get some warning during the extraction process, you can ignore them
- Set the HADOOP_HOME environment variable to C:\Hadoop



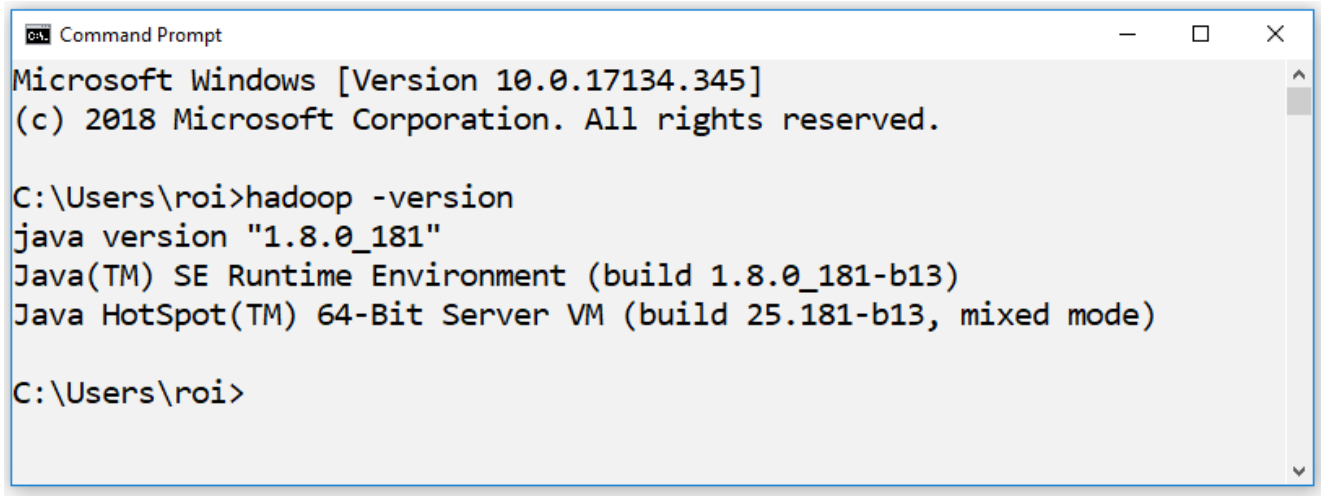
Install Hadoop

- Add %JAVA_HOME%/bin and %HADOOP_HOME%/bin into the Path variable:



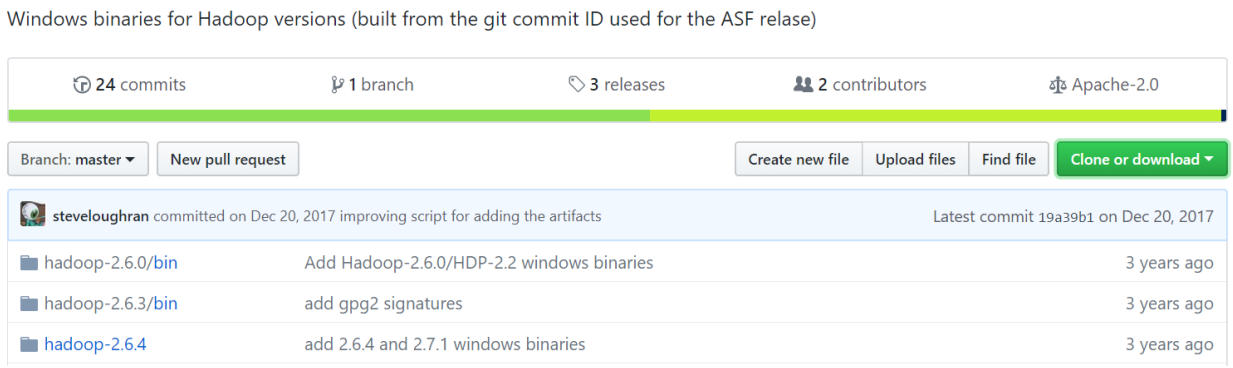
Verify Setup

- You should be able to verify your settings via the following command:



Install WinUtils

- WinUtils are Windows binaries for Hadoop versions
- Download winutils from https://github.com/steveloughran/winutils



- Extract the file winutils-master.zip
- Choose the subfolder that corresponds to the version of Hadoop you downloaded
- Copy the files from this subfolder\bin to C:\Hadoop\bin
  - e.g., copy winutils-master\hadoop-3.0.0\bin to C:\Hadoop\bin
  - Choose replace the files in the destination

Hadoop Configuration

- Before running Hadoop, we need to configure some files in *C:\Hadoop\etc\hadoop*
- First, edit the file *C:\Hadoop\etc\hadoop\core-site.xml*
  - This file contains the name of the default file system
  - All Hadoop services and clients use this file to locate the namenode
- Add the following configuration key to the file:

```
<configuration>
    <property>
        <name>fs.default.name</name>
        <value>hdfs://0.0.0.0:19000</value>
    </property>
</configuration>
```

Hadoop Configuration

- Now edit the file *hdfs-site.xml*
  - This file contains the HTTP addresses for namenode and datanode
- Add the following configuration keys to the file:

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
    <property>
        <name>dfs.name.dir</name>
        <value>file:///C:/Hadoop/dfs/name</value>
    </property>
    <property>
        <name>dfs.data.dir</name>
        <value>file:///C:/Hadoop/dfs/data</value>
    </property>
</configuration>
```

- Create the directories C:\Hadoop\dfs\name and C:\Hadoop\dfs\data

Hadoop Configuration

- Edit *mapred-site.xml*
  - This file contains the framework for executing MapReduce jobs
- Add the following, replacing *%USERNAME%* with your Windows username:

```
<configuration>
    <property>
        <name>mapreduce.job.user.name</name>
        <value>%USERNAME%</value>
    </property>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
    <property>
        <name>yarn.apps.stagingDir</name>
        <value>/user/%USERNAME%/staging</value>
    </property>
    <property>
        <name>mapreduce.jobtracker.address</name>
        <value>local</value>
    </property>
</configuration>
```

Hadoop Configuration

- Edit *yarn-site.xml* and add the following entries:

```
<configuration>
    <property>
        <name>yarn.server.resourcemanager.address</name>
        <value>0.0.0.0:8020</value>
    </property>
    <property>
<name>yarn.server.resourcemanager.application.expiry.interval</name>
        <value>60000</value>
    </property>
    <property>
        <name>yarn.server.nodemanager.address</name>
        <value>0.0.0.0:45454</value>
    </property>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
    <property>
        <name>yarn.nodemanager.aux-
services.mapreduce.shuffle.class</name>
        <value>org.apache.hadoop.mapred.ShuffleHandler</value>
    </property>
    <property>
        <name>yarn.server.nodemanager.remote-app-log-dir</name>
        <value>/app-logs</value>
    </property>
    <property>
        <name>yarn.nodemanager.log-dirs</name>
        <value>/dep/logs/userlogs</value>
    </property>
```

```
    <property>
        <name>yarn.server.mapreduce-appmanager.attempt-
listener.bindAddress</name>
        <value>0.0.0.0</value>
    </property>
    <property>
        <name>yarn.server.mapreduce-appmanager.client-
service.bindAddress</name>
        <value>0.0.0.0</value>
    </property>
    <property>
        <name>yarn.log-aggregation-enable</name>
        <value>true</value>
    </property>
    <property>
        <name>yarn.log-aggregation.retain-seconds</name>
        <value>-1</value>
    </property>
    <property>
        <name>yarn.application.classpath</name>
<value>%HADOOP_CONF_DIR%,%HADOOP_COMMON_HOME%/share/hadoop/common/*,%HADO
OP_COMMON_HOME%/share/hadoop/common/lib/*,%HADOOP_HDFS_HOME%/share/hadoop
/hdfs/*,%HADOOP_HDFS_HOME%/share/hadoop/hdfs/lib/*,%HADOOP_MAPRED_HOME%/s
hare/hadoop/mapreduce/*,%HADOOP_MAPRED_HOME%/share/hadoop/mapreduce/lib/*
,%HADOOP_YARN_HOME%/share/hadoop/yarn/*,%HADOOP_YARN_HOME%/share/hadoop/y
arn/lib/*</value>
    </property>
</configuration>
```

Hadoop Configuration

- Finally, change the file hadoop-env.cmd to add the following lines at the end of file:

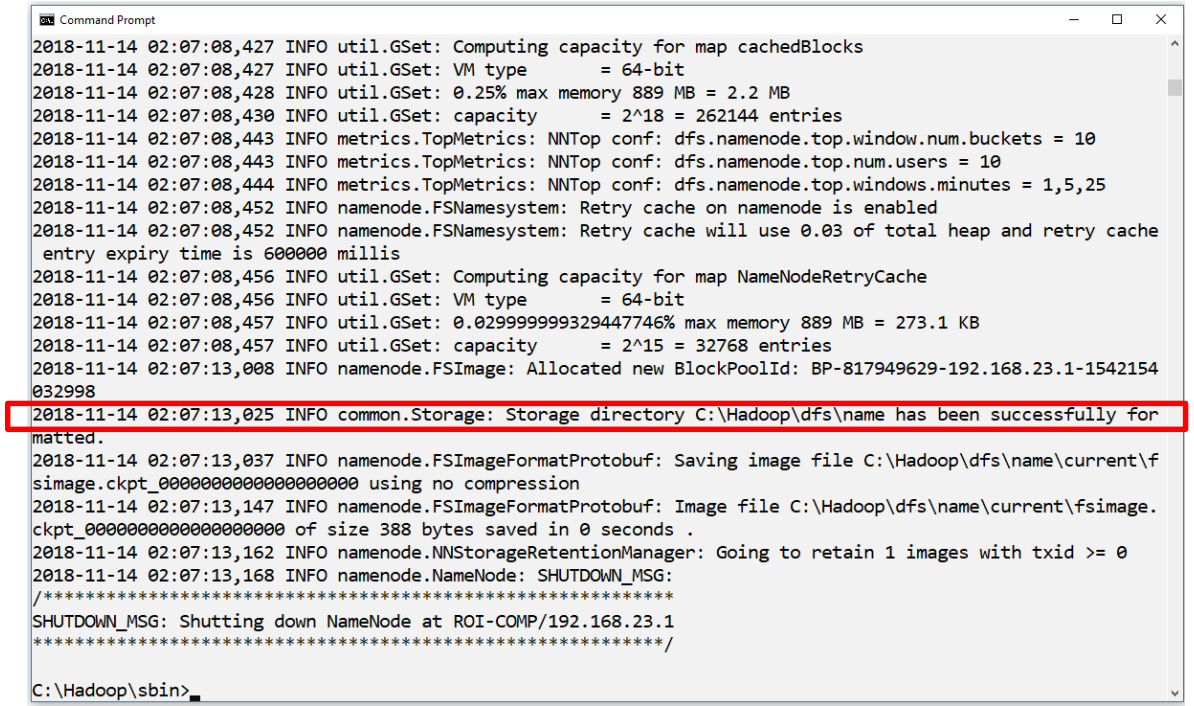
```
set HADOOP_PREFIX=%HADOOP_HOME%
set HADOOP_CONF_DIR=%HADOOP_PREFIX%\etc\hadoop
set YARN_CONF_DIR=%HADOOP_CONF_DIR%
set PATH=%PATH%;%HADOOP_PREFIX%\bin
```

Format HDFS File System

- Open command prompt in administrator mode
- Run the following command to format the file system:

```
hdfs namenode -format
```

- The output should contain log messages indicating the success of the format:



Startup Scripts

Experis Academy
ManpowerGroup

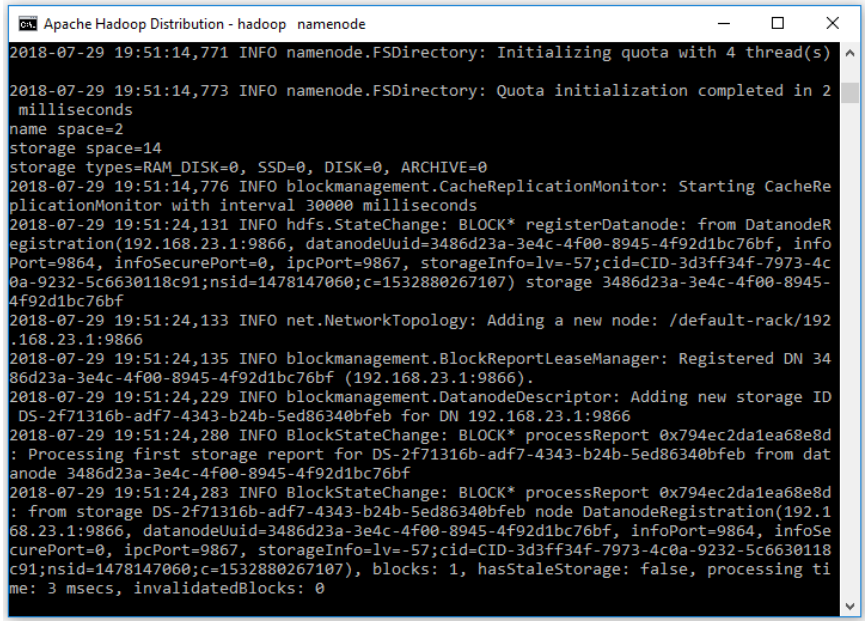
- The C:\Hadoop\sbin directory contains some scripts used to launch Hadoop DFS and Hadoop Map/Reduce daemons:
  - **start-dfs.sh** - Starts the Hadoop DFS daemons, the namenode and datanodes. Use this before start-mapred.sh
  - **stop-dfs.sh** - Stops the Hadoop DFS daemons.
  - **start-mapred.sh** - Starts the Hadoop Map/Reduce daemons, the jobtracker and tasktrackers.
  - **stop-mapred.sh** - Stops the Hadoop Map/Reduce daemons.
  - **start-all.sh** - Starts all Hadoop daemons, the namenode, datanodes, the jobtracker and tasktrackers. Deprecated; use start-dfs.sh then start-mapred.sh
  - **stop-all.sh** - Stops all Hadoop daemons. Deprecated; use stop-mapred.sh then stop-dfs.sh

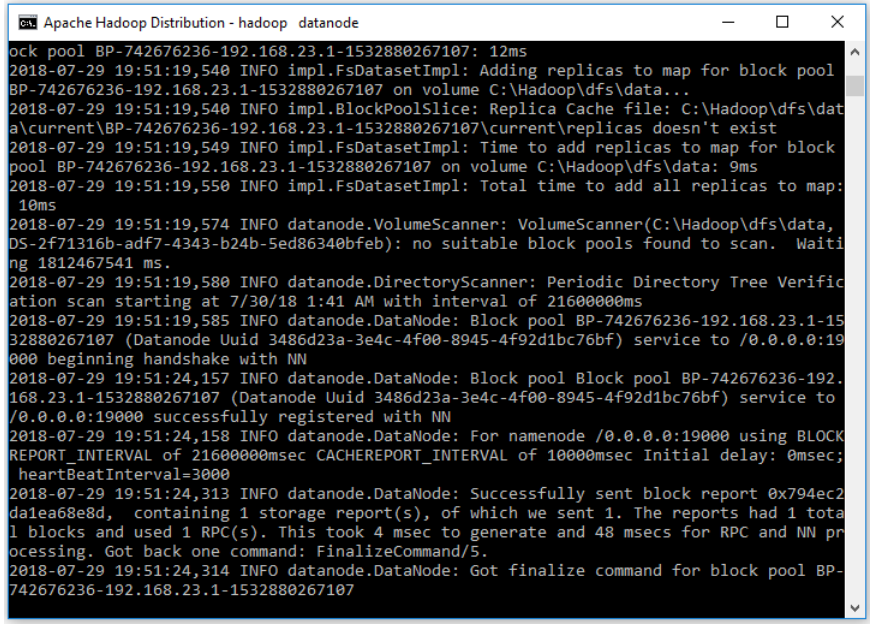
Starting HDFS Daemons

- Start namenode and datanode by running the following command:

```
cd C:\Hadoop\sbin
C:\Hadoop\sbin>start-dfs.cmd
```

- Two command prompts named namenode and datanode will open:





HDFS Commands

- The File System (FS) shell includes various shell-like commands that directly interact with the Hadoop Distributed File System (HDFS) The FS shell is invoked by
- Most of the commands in FS shell behave like the corresponding Unix commands
- A list of HDFS commands can be found here
- For example, to copy a file from a local directory to HDFS type:

```
hdfs dfs -put /localfs/source/path /hdfs/destination/path
```

```
C:\Hadoop\sbin>hdfs dfs -put C:/test.txt /
```

- To display the contents of the HDFS root folder type:

```
hdfs dfs –ls /
```

```
C:\Hadoop\sbin>hdfs dfs -ls /
Found 1 items
-rw-r--r--   1 roi supergroup         20 2018-11-14 02:12 /test.txt
```

HDFS Commands

- To view the contents of a file in HDFS:

```
hdfs dfs -cat /hdfs/path
```

```
C:\Hadoop\sbin>hdfs dfs -cat /test.txt
Hello world
Goodbye
```

- To copy a file from HDFS to a local directory:

```
hdfs dfs -get /hdfs/source/path /localfs/destination/path
```

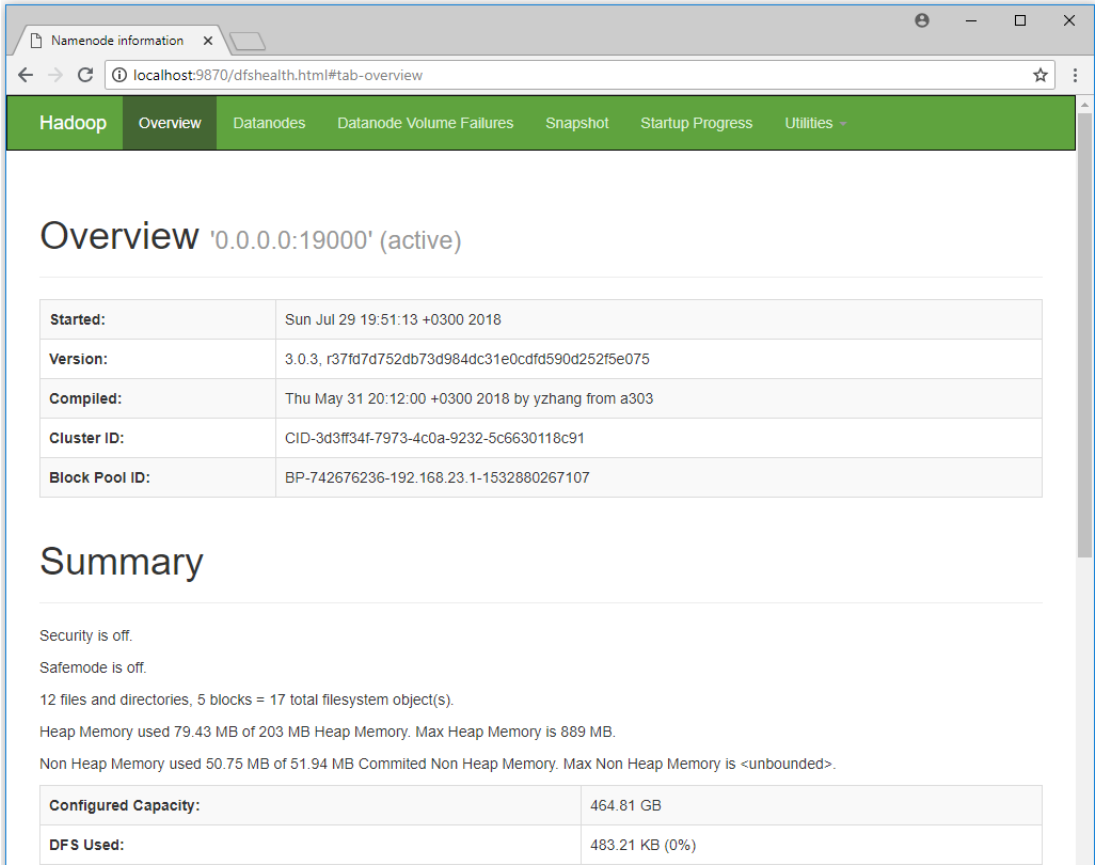
```
C:\Hadoop\sbin>hdfs dfs -get /test.txt C:\temp

C:\Hadoop\sbin>more C:\temp\test.txt
Hello world
Goodbye
```

- To a remove a file from HDFS:

```
hdfs dfs –rm /hdfs/path
```

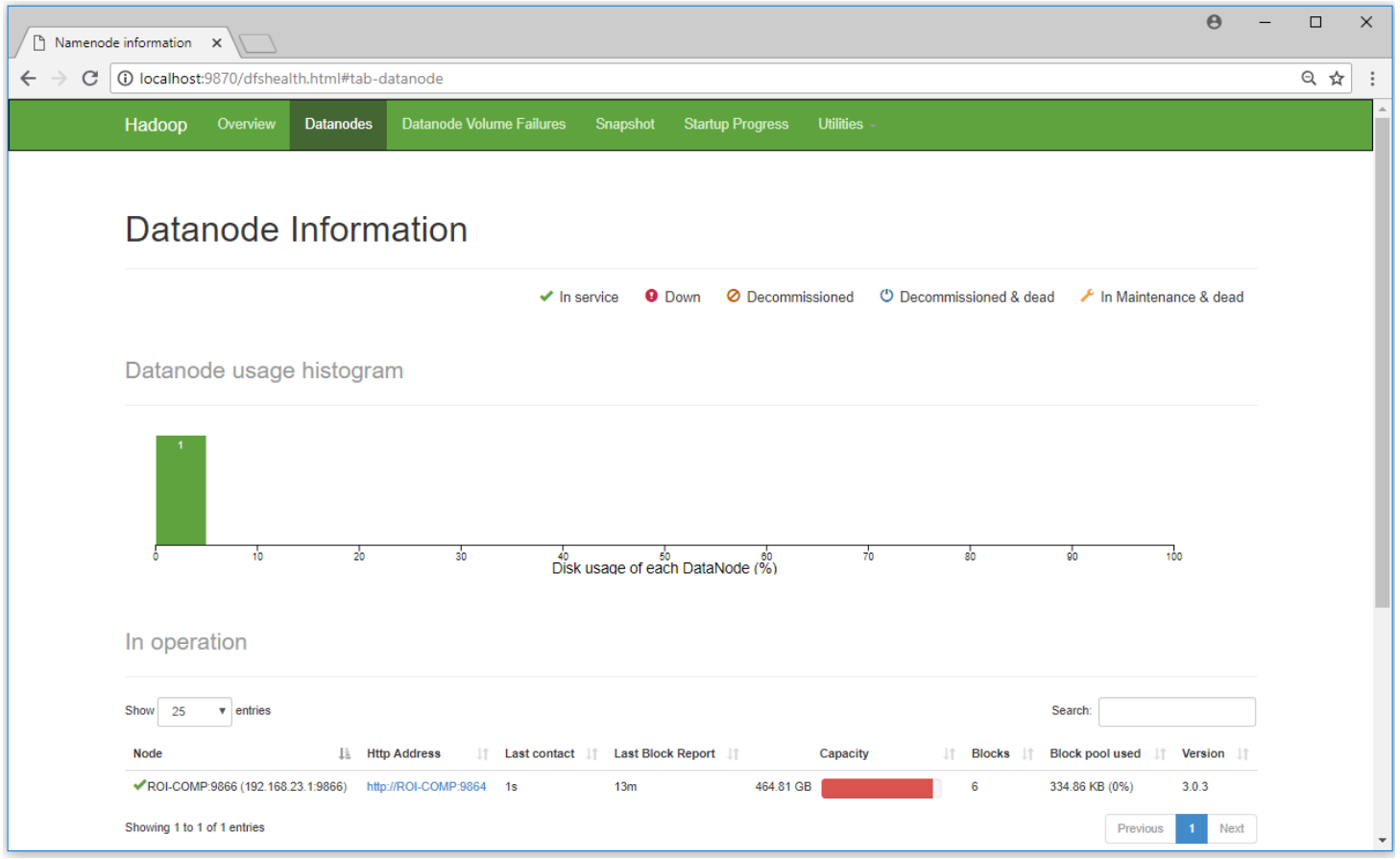
Namenode UI

- Default URL: http://localhost:9870



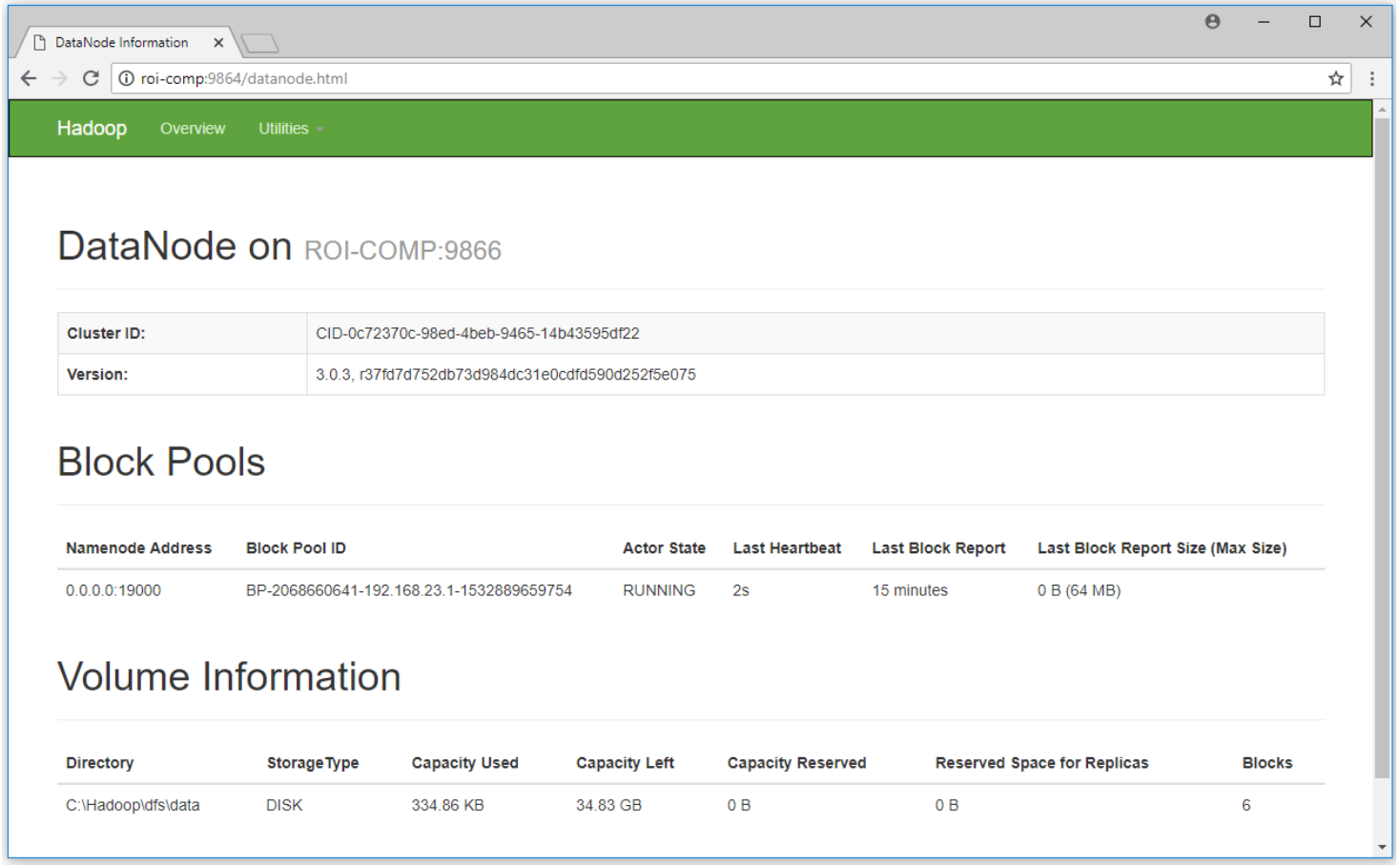
Namenode UI

- Click the Datanodes tab to find out all the data nodes:



Datanode UI

- In this case, we only have a single data node with UI URL as http://localhost:9864



Exercise

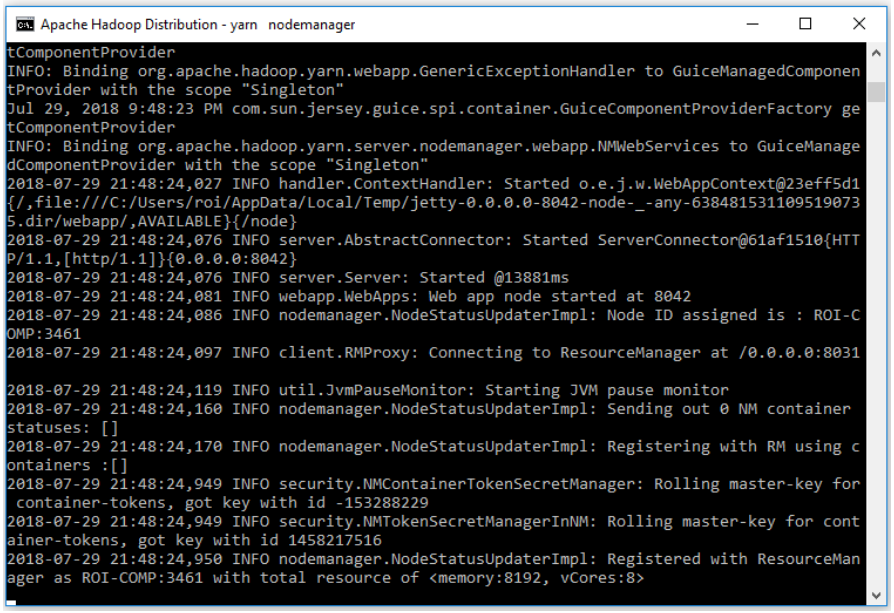
- Create a directory named mydir under HDFS root /
- Copy a few files from your local directory to mydir
- Delete a file in mydir
- Download a file from HDFS to your local directory

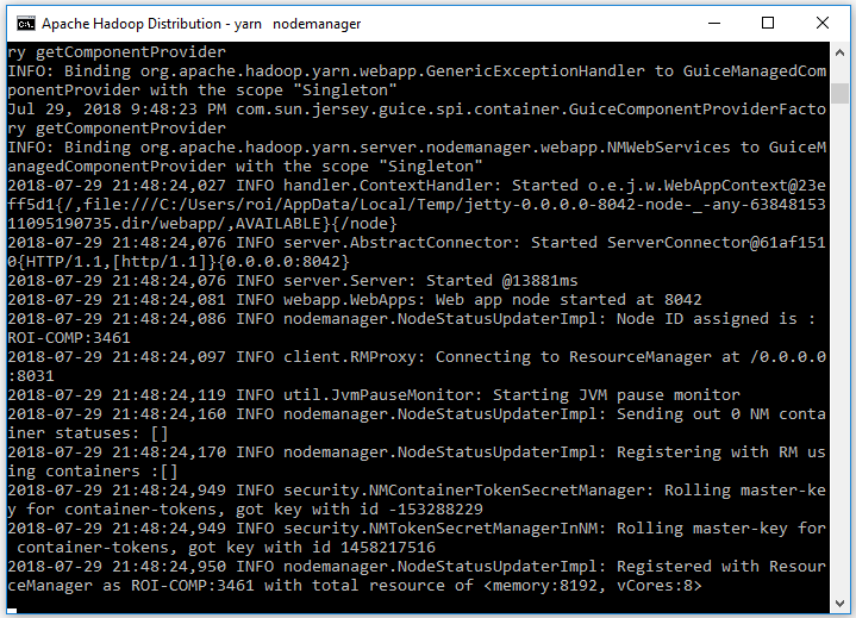
Start YARN Daemons

- Copy the file hadoop-yarn-server-timelineservice-3.0.3.jar from C:\Hadoop\share\hadoop\yarn\timelineservice to C:\Hadoop\share\hadoop\yarn
- Start YARN through the following command in admin mode:

```
C:\Hadoop\sbin\start-yarn.cmd
```

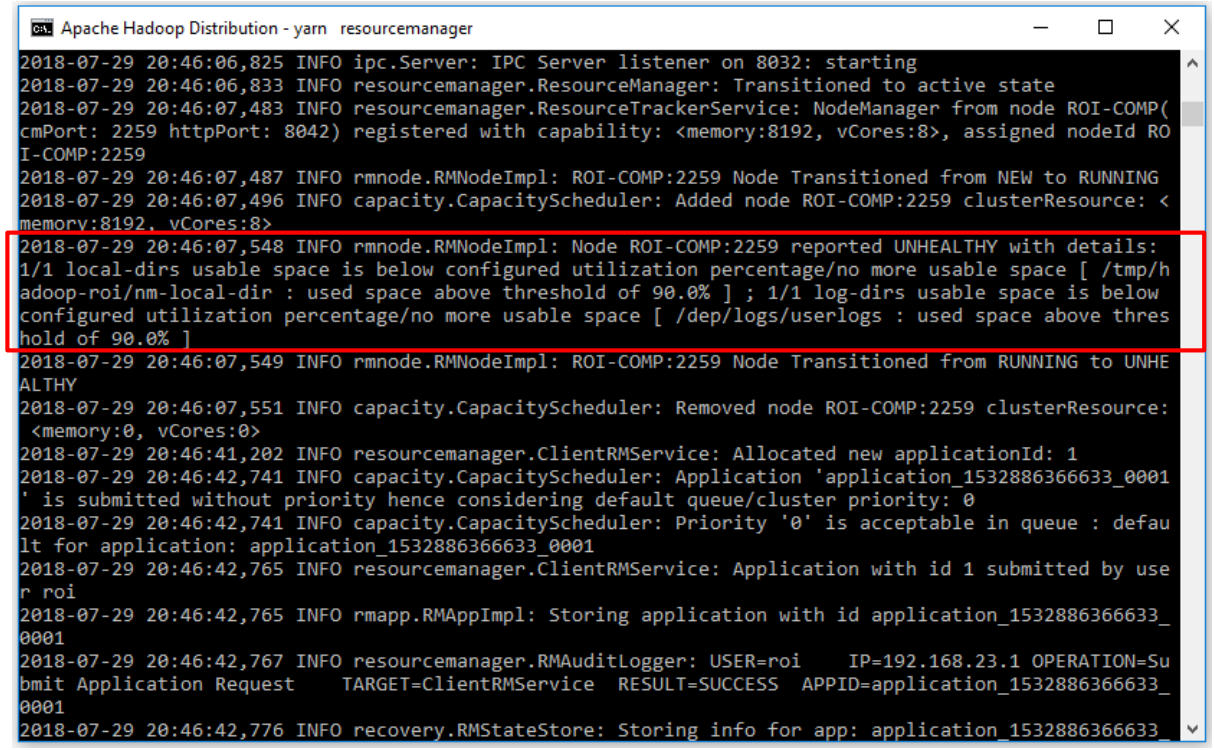
- Similar to HDFS, two windows will open:





Start YARN Daemons

- Make sure you have at least 10% free disk space, otherwise you might get the following error:



Start YARN Daemons

- You can increase the threshold in the config file yarn-site.xml:

```
<property>
    <name>yarn.nodemanager.disk-health-checker.max-disk-utilization-per-disk-percentage</name>
    <value>98</value>
</property>
```

YARN Commands

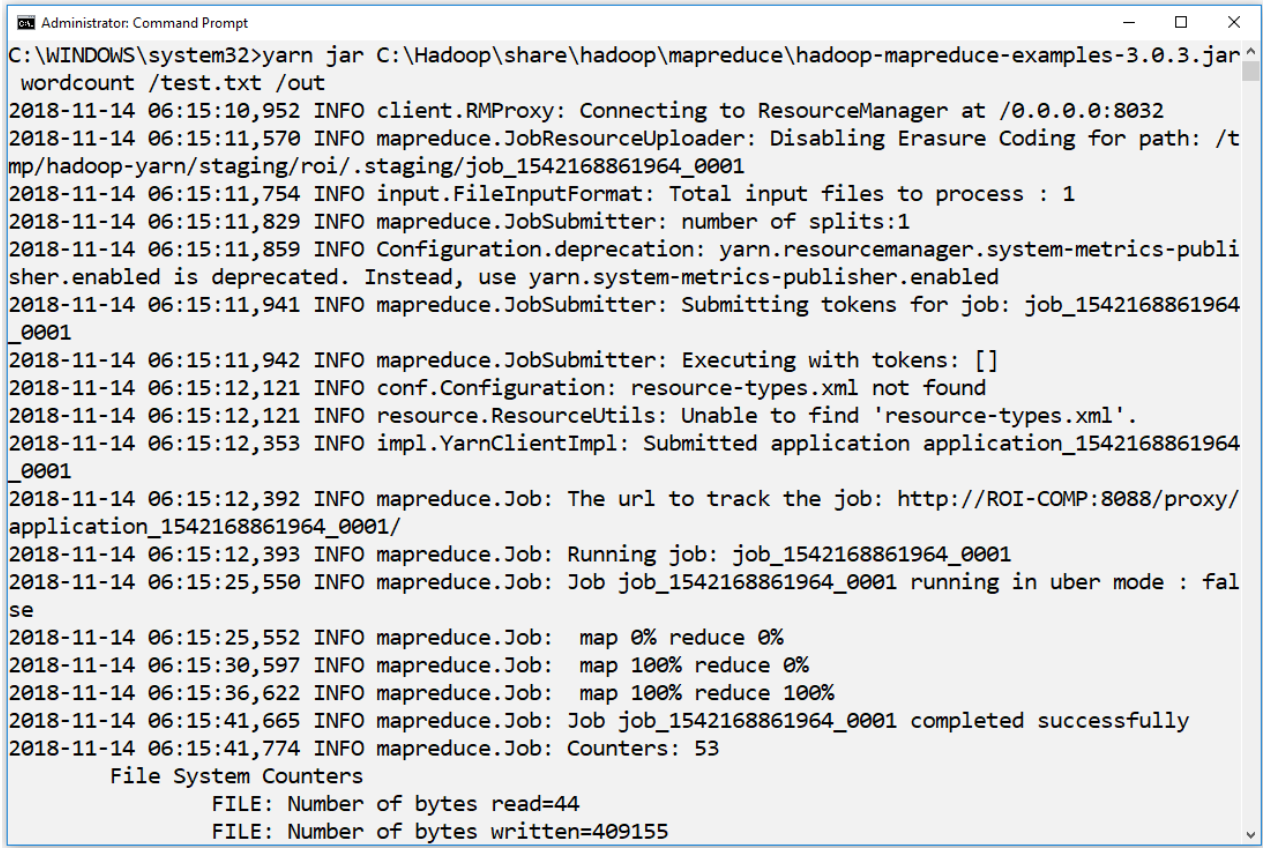
- Type yarn in command line to see a list of available commands:

```
C:\Users\roi>yarn
Usage: yarn [--config confdir] [--loglevel loglevel] COMMAND
       where COMMAND is one of:
  resourcemanager      run the ResourceManager
  nodemanager          run a nodemanager on each slave
  router               run the Router daemon
  timelineserver       run the timeline server
  timelinereader       run the timeline reader server
  rmadmin              admin tools
  version              print the version
  jar <jar>          run a jar file
  application          prints application(s) report/kill application
  applicationattempt   prints applicationattempt(s) report
  cluster              prints cluster information
  container            prints container(s) report
  node                 prints node report(s)
  queue                prints queue information
  logs                 dump container logs
  schedulerconf        updates scheduler configuration
  classpath            prints the class path needed to get the
                       Hadoop jar and the required libraries
  daemonlog            get/set the log level for each daemon
  or
  CLASSNAME            run the class named CLASSNAME
Most commands print help when invoked w/o parameters.
```

Running a Map-Reduce Job

- Run the following sample job (under administrator) to perform word count:

```
yarn jar C:\Hadoop\share\hadoop\mapreduce\hadoop-mapreduce-examples-3.0.3.jar
wordcount /test.txt /out
```



Running a Map-Reduce Job

- To the result of this job is stored in the /out HDFS folder:

```
C:\WINDOWS\system32>hdfs dfs -ls /out
Found 2 items
-rw-r--r--   1 roi supergroup          0 2018-11-14 06:15 /out/_SUCCESS
-rw-r--r--   1 roi supergroup         26 2018-11-14 06:15 /out/part-r-00000

C:\WINDOWS\system32>hdfs dfs -cat /out/part-r-00000
Goodbye 1
Hello   1
world   1
```

Running a Map-Reduce Job

- You can find the Java source code of this example under C:\Hadoop\share\hadoop\mapreduce\sources
- The class that implements the mapper task:

```
public class WordCount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context) throws
         IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }
```

Running a Map-Reduce Job

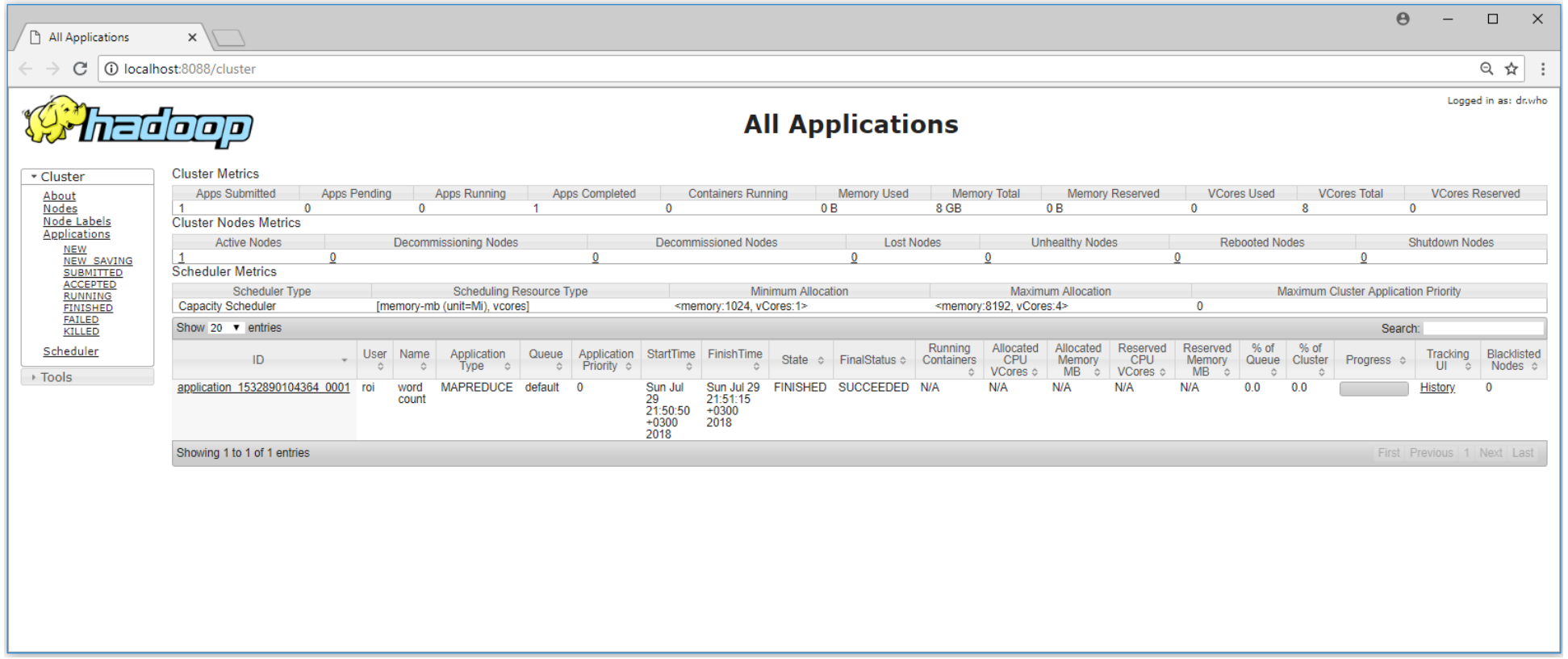
Experis Academy
ManpowerGroup

- The class implementing the reducer task:

```
public static class IntSumReducer
     extends Reducer<Text,IntWritable,Text,IntWritable> {
  private IntWritable result = new IntWritable();

  public void reduce(Text key, Iterable<IntWritable> values,
                     Context context
                     ) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) {
      sum += val.get();
    }
    result.set(sum);
    context.write(key, result);
  }
}
```

Application Status UI

- You can also view your job status through YRAN website. The default path is http://localhost:8088



Cluster Status UI

- Click Nodes to see the status of the nodes in the cluster: