# Big Data Analytics and Reasoning - Practice 01

Giuseppe Mazzotta

giuseppe.mazzotta@unical.it

# Hadoop Ecosystem

Hadoop is an open source framework for reliable, scalable and distributed computing

Allow us to deal with huge amount of data

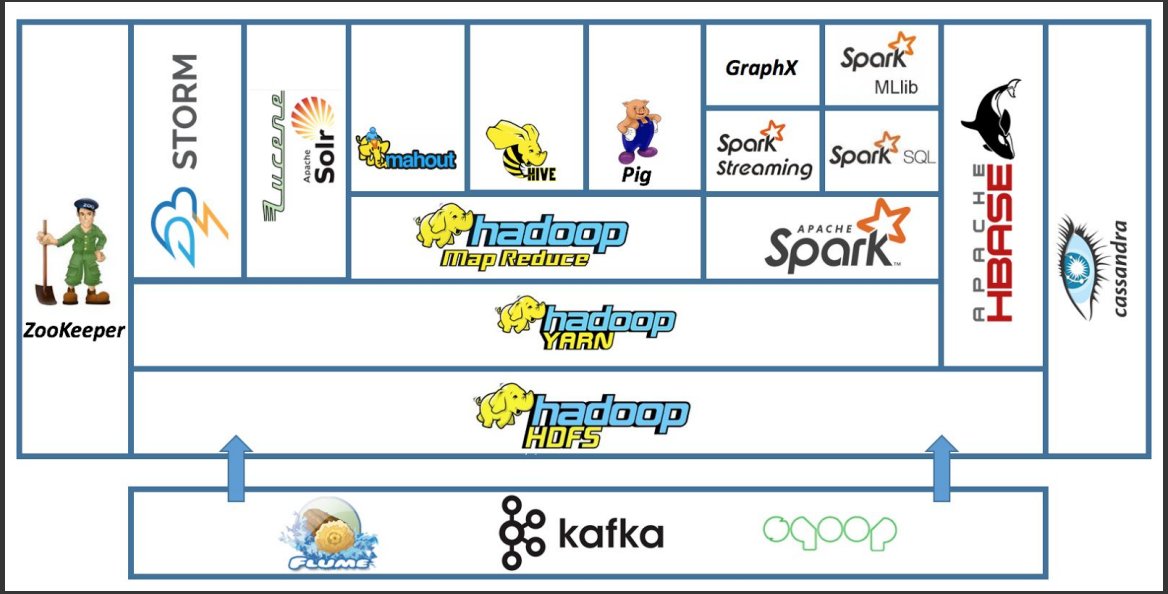
Move computation rather than move data

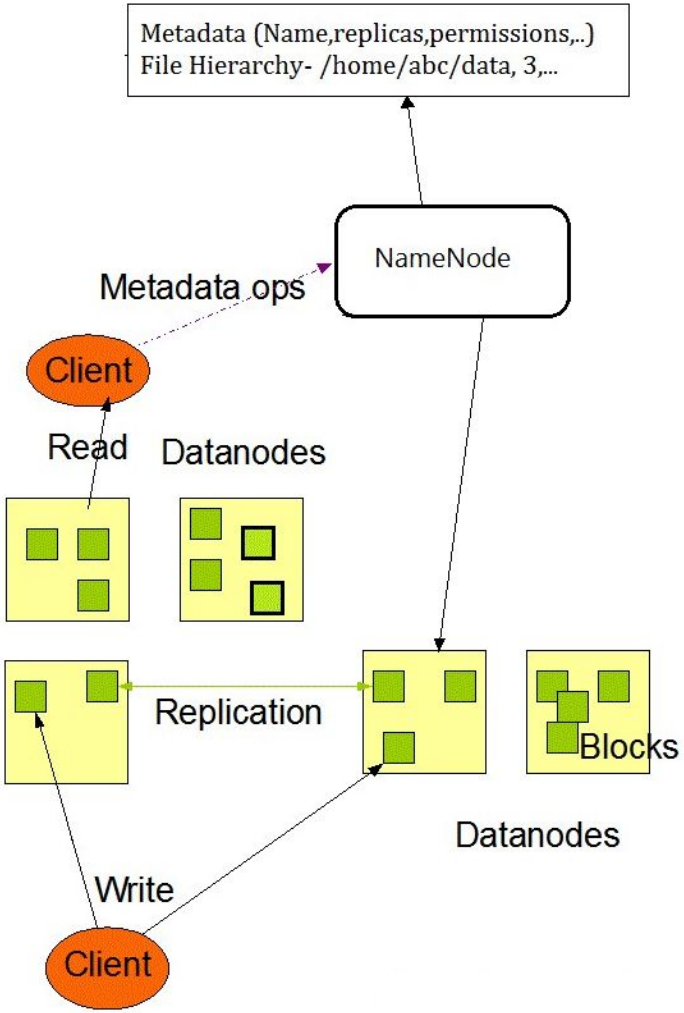
Distributed clusters of commodity hardware

# 1. Overall Panorama

- **HDFS**
  Distributed file system
  Abstract interface to be used as a classical file system
- **Yarn**
  Distributed service for resource and job management.
- **DBMS and Data Analysis**
  Support many service to efficient storing, querying and analyzing data

# What are we dealing with?






# HDFS

HDFS is a distributed file system

It has be designed to scale up by adding commodity machines

Hierarchical architecture:

- **Namenode**
  Top level service
  Coordinate subordinates
  Handle client requests
- **Datanodes**
  Bottom level services
  Slave nodes that store data and perform assigned task

# YARN - Yet Another Resource Negotiator

YARN is a framework for resource management and job scheduling

Main motivation: Split resource management and job scheduling/monitoring to obtain a more scalable parallel computation system

Basic concepts

- **Resource Manager**
- **Node Manager**
- **Application Master**
- **Container**

# YARN

**Resource Manager**
Resource Manager is the master service that manages all the resources of our cluster

**Container**
A container is a collection of physical resources, such as RAM, CPU cores and so on, on a precise machine

**Application Master**
The application master is the process that coordinates the execution of client applications
It is a particular container that negotiates for other containers with the resource manager
It has to monitor the application execution, tracking its progresses

**Node Manager**
A node manager is a slave service that will hosts containers and communicates with resource manager to track resource availability

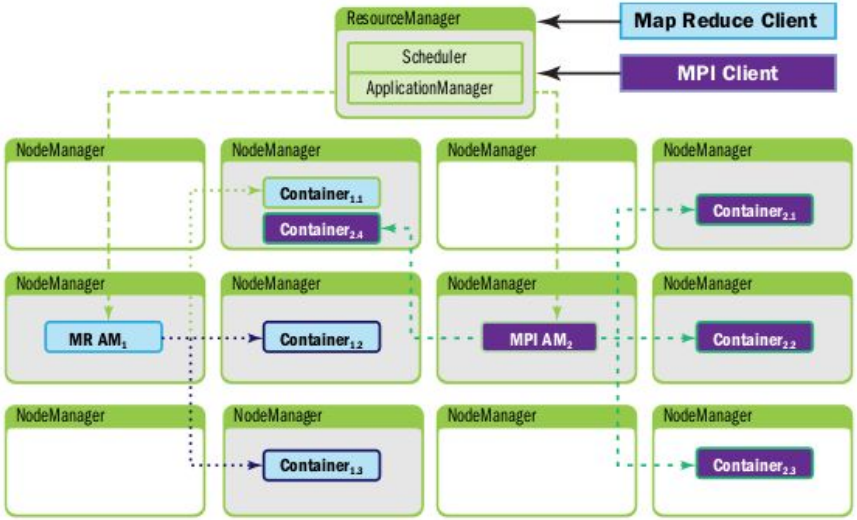



Figure 4.1 YARN architecture with two clients (MapReduce and MPI). The client MPI $AM_2$ is running an MPI application and the client MR $AM_1$ is running a MapReduce application.

# HIVE

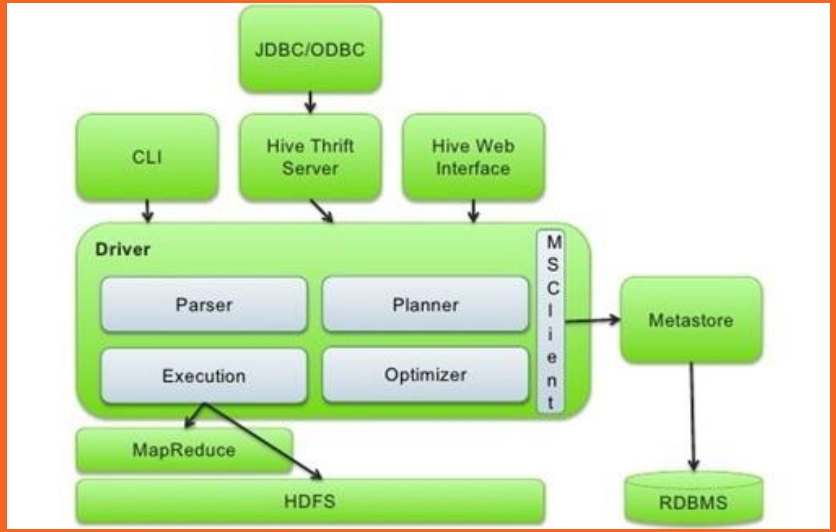
HIVE is a data warehouse framework that stores table as files into hdfs

Support many different file formats that efficiently index rows

Use relational database for storing schemas metadata

Provide a SQL-like language to create and query tables

Manly used as data warehouse system not as a transactional database



# HBASE

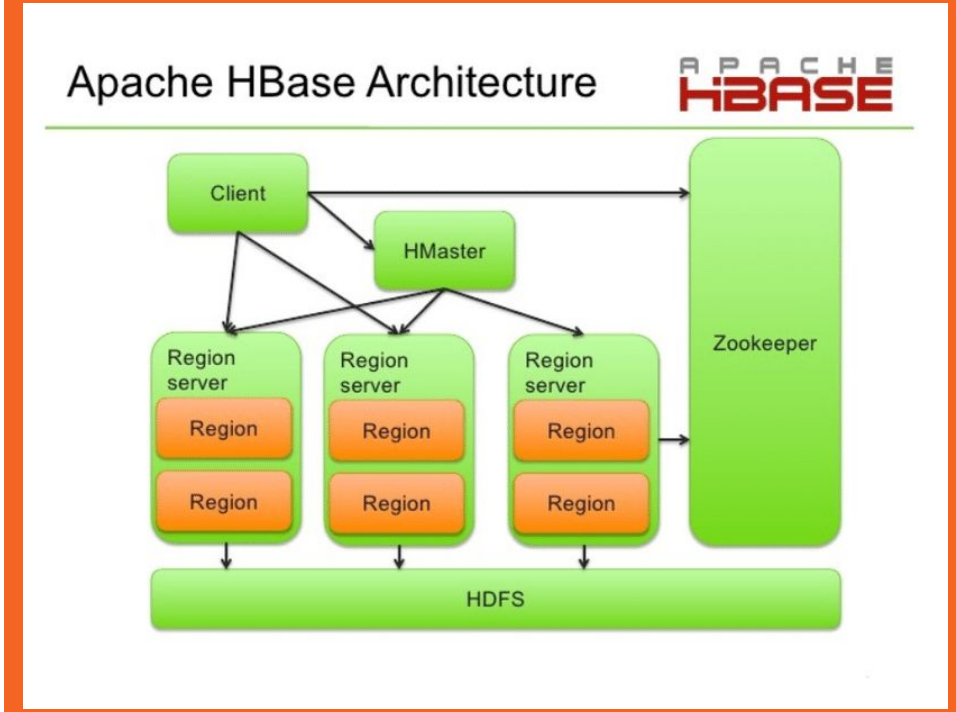
HBASE is a NoSQL database that stores data directly into hdfs

Data are distributed among region servers

Automatic regionserver failover

Suitable for MapReduce application

Provide a Java client API



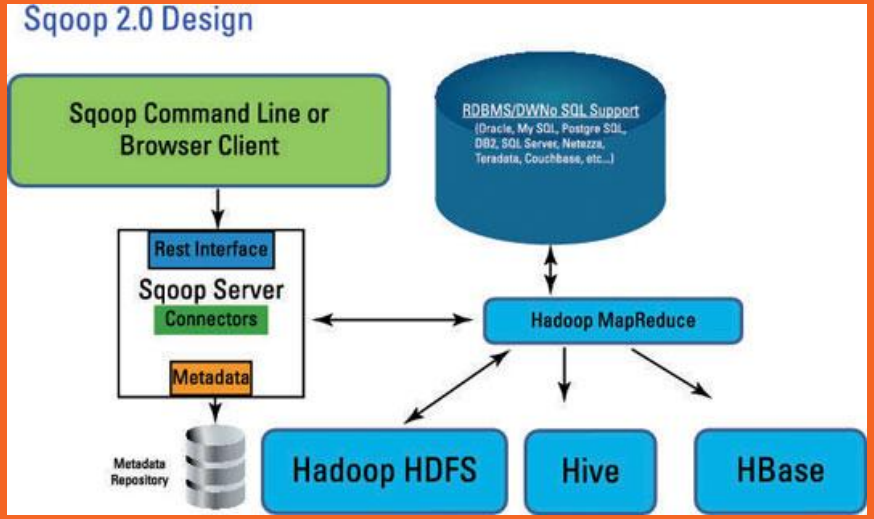
# SQOOP

ETL (Extract Transform Load) Tool

Import/Export data from RDBMS to HDFS

Use map-reduce for storing data

Filtering and aggregate



# SPARK

In-memory framework for large-scale data processing

Load data in main memory instead of reading from disk

Ingestion, cleaning, transformation and republish

Provides API for many programming languages such as Scala, Python, Java, R

Spark SQL, MLlib, GraphX

# New York City Taxi Fare Prediction

## Objective

Build a system that is able to store data about taxi ride in New York City and makes fare prediction of the cost of possible taxi ride

# 2. Overall Description

### ➔ Data source

Daily taxi rides are stored into mysql database

### ➔ Data format

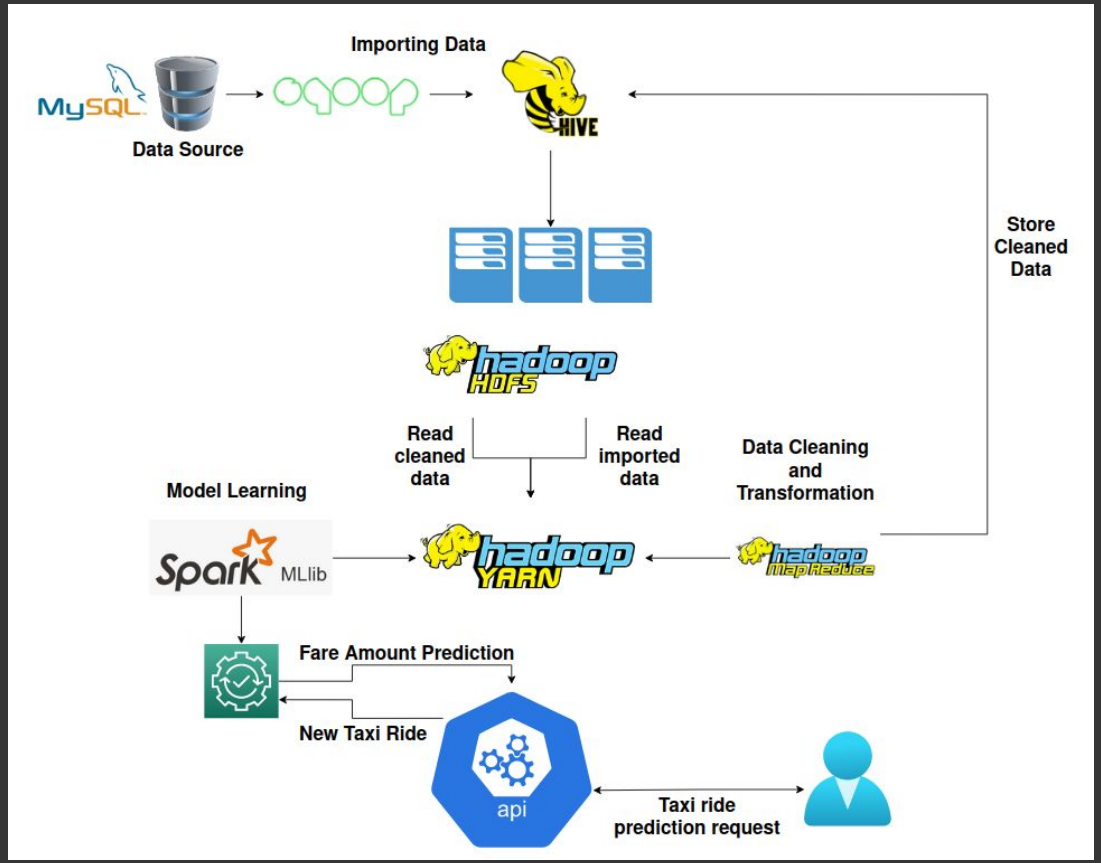
Each taxi ride is described by:

- Pickup/dropoff timestamp
- Pickup/dropoff latitude and longitude
- Passenger count
- Fare amount

### ➔ Problem

Storage is limited to two days
Provide users a system to estimate the cost of a taxi ride

# Possible Big Data Solution



# Let’s start building our cluster.




# How to build a virtual cluster?

Each node of the cluster is a virtual machine

Each machine should be able to communicate with all the others

SSH password less access

Operating system suggested: Ubuntu 18.04.6 live server

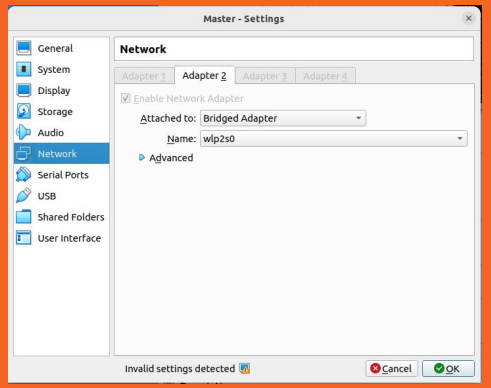
**Tip**

Our cluster is composed by three machines. One will be used as master node and the other as slave nodes

# How to build a virtual cluster?

## Main steps

- Create master, slave1 and slave2 virtual machine using virtualbox
- Use the same username to each machine (e.g. “hadoop”)
- Bridge Network Configuration:
  - Each machine has a “Bridged Adapter”
  - Configure network interfaces with ip address in the subnet of the host machine
  - **This setting requires to use your own network**
- Nat Network
  - Create a nat network
  - Each machine has a “Network with NAT” adapter
  - Assign an IP belonging to the NAT network


```
hadoop@giuseppe-Prestige-15-A10SC:~$ ifconfig -a
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 109782  bytes 10476963 (10.4 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 109782  bytes 10476963 (10.4 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.4  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::be7a:caa1:1c46:d693  prefixlen 64  scopeid 0x20<link>
        ether 50:eb:71:d9:6f:c5  txqueuelen 1000  (Ethernet)
        RX packets 788133  bytes 754863138 (754.8 MB)
        RX errors 0  dropped 1  overruns 0  frame 0
        TX packets 363227  bytes 116675993 (116.6 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
hadoop@master:~$ ifconfig -a
enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.10  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::a00:27ff:fe4a:15c  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:4a:01:5c  txqueuelen 1000  (Ethernet)
        RX packets 94273  bytes 21198805 (21.1 MB)
        RX errors 0  dropped 8  overruns 0  frame 0
        TX packets 48612  bytes 13243643 (13.2 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 152823  bytes 11125330 (11.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 152823  bytes 11125330 (11.1 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
hadoop@master:~$ cat /etc/network/interfaces
# ifupdown has been replaced by netplan(5) on this system.  See
# /etc/netplan for current configuration.
# To re-enable ifupdown on this system, you can run:
#    sudo apt install ifupdown
# ifupdown has been replaced by netplan(5) on this system.  See
# /etc/netplan for current configuration.
# To re-enable ifupdown on this system, you can run:
#    sudo apt install ifupdown

auto enp0s8
iface enp0s8 inet static
address 192.168.1.10
netmask 255.255.255.0
gateway 192.168.1.4
```

# How to build a virtual cluster?

Main steps

- Configure /etc/hosts files:
  - Add record <ip hostname> for each machine
- Generate a public key in the master node (e.g. use ssh-keygen)
- Share public key to slave nodes

Test accessibility from the master:

- ssh slave1
- ssh slave2

```
hadoop@master:~$ cat /etc/hosts
#127.0.0.1 localhost

192.168.1.10    master
192.168.1.11    slave1
192.168.1.12    slave2

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

```
hadoop@master:~$ ssh-keygen -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hadoop/.ssh/id_rsa): /home/hadoop/.ssh/master_key
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/hadoop/.ssh/master_key.
Your public key has been saved in /home/hadoop/.ssh/master_key.pub.
The key fingerprint is:
SHA256:ecpospU9LXIoGi3pNfw387wFPsVzoP8IPbD8/XGsSh4 hadoop@master
The key's randomart image is:
+---[RSA 4096]----+
|                 |
|                 |
|            .    |
|         . o .   |
|        S = + .  |
|   +   * * B o . |
|  + B B B B E  .o|
| . = O o++ B * .o|
|  o . .. ++.=.+..|
+----[SHA256]-----+
```

```
hadoop@master:~$ cat .ssh/master_key.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQCgeMkfUp2i4NJ4xz8jcosCL3n0uAmwT4+cCYizB54+z1TBxTpcbFcBXug4LGQnM98Cn
wawk/ZvB7rvkBh991PHVr0viaauUTghu6QgPygcu/KIZyj9ZL4fptakkC2MsRHVQTUTWNkvmKBtdb8LIYOAyz8Ye+eIdPhc/LMTNz1N1l
+TKOQ7c+PtwqyxZnW69swCJAPtn1XF8wKB6D1e9sxki6lriLENKrqBJ9wuVwPV9MuTGdN4BD72JlmmkJM09+ENGl8XgxsMVvZcK1PBWaC
d3dLKrK8ZYCpIrP4mU+lbqCi0hdY0oAvDfBzri9OM5V66TfQveMmffGpzsAsDBXaE9W3X3D/ZuC8E8+A6u/1aisezbVU1EoGsKBreZEXd
YN2oXC6hwUSZXuP4x64aulQ1kkfnTsirucCvUqXJ3auyeY+nYlDfkxjNJePiLAPoHW0QM2IGgNNUzroo8/buxuGyLBAE/ybj7H9wYnGEQ
OqM0rVnrOXRuouDhNyMNDM1STUhUFq4i0Dsnw9SmNcCrLJ+A6Bo7eh4B8x4SkK7oT7Pfp2V88r3/y+eyCw+5RLXC8ofbPIgjGI7J7UQTJ
Rbi4KkC7glWVQjTqQjJ2/BgEwB84R3kXVY0VGI2zJUrXpRT4RYdcZAvff2JvQ0uobeUpPl9zIf+/p6epUHfiuTJc9gRQ== hadoop@mas
ter
```

```
hadoop@slave1:~$ cat .ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQCgeMkfUp2i4NJ4xz8jcosCL3n0uAmwT4+cCYizB54+z1TBxTpcbFcBXug4LGQnM98Cn
wawk/ZvB7rvkBh991PHVr0viaauUTghu6QgPygcu/KIZyj9ZL4fptakkC2MsRHVQTUTWNkvmKBtdb8LIYOAyz8Ye+eIdPhc/LMTNz1N1l
+TKOQ7c+PtwqyxZnW69swCJAPtn1XF8wKB6D1e9sxki6lriLENKrqBJ9wuVwPV9MuTGdN4BD72JlmmkJM09+ENGl8XgxsMVvZcK1PBWaC
d3dLKrK8ZYCpIrP4mU+lbqCi0hdY0oAvDfBzri9OM5V66TfQveMmffGpzsAsDBXaE9W3X3D/ZuC8E8+A6u/1aisezbVU1EoGsKBreZEXd
YN2oXC6hwUSZXuP4x64aulQ1kkfnTsirucCvUqXJ3auyeY+nYlDfkxjNJePiLAPoHW0QM2IGgNNUzroo8/buxuGyLBAE/ybj7H9wYnGEQ
OqM0rVnrOXRuouDhNyMNDM1STUhUFq4i0Dsnw9SmNcCrLJ+A6Bo7eh4B8x4SkK7oT7Pfp2V88r3/y+eyCw+5RLXC8ofbPIgjGI7J7UQTJ
Rbi4KkC7glWVQjTqQjJ2/BgEwB84R3kXVY0VGI2zJUrXpRT4RYdcZAvff2JvQ0uobeUpPl9zIf+/p6epUHfiuTJc9gRQ== hadoop@mas
ter
```

# That’s it.