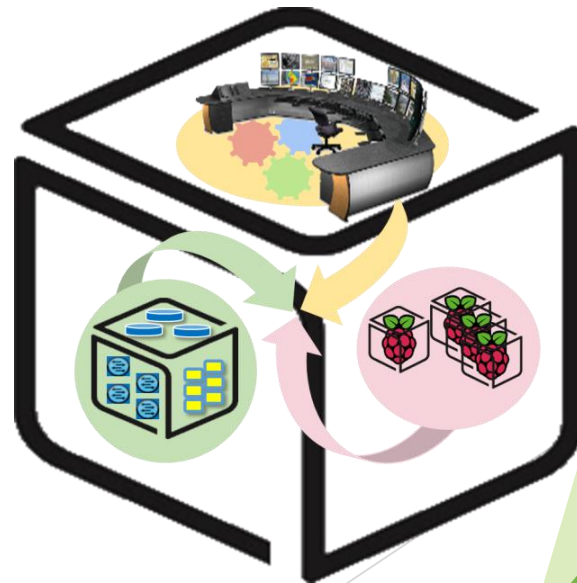


# SmartX Labs for Computer Systems

Cluster Lab

(2016, Spring)

NetCS Lab



# History and Contributor of Cluster Lab

(2016. 06. 30.)

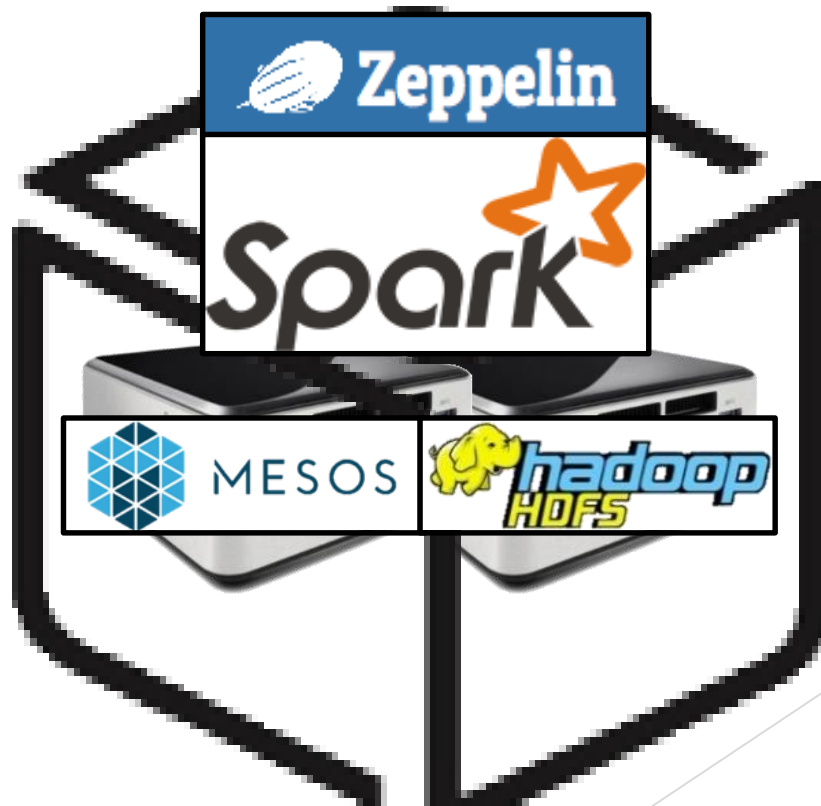
Version	Updated Date	Updated Contents	Contributor
-	2015/10	(구) Analytics Lab 작성	송지원
v1	2016/04	Cluster Lab 초안 작성	김승룡
v2	2016/05	Cluster Lab 수정	송지원
v3r3	2016/05/28	Cluster Lab 2차 수정 (내용 수정 및 추가)	송지원
v4r1	2016/05/30	Cluster Lab 3차 수정 (피드백 반영)	송지원
v5	2016/06/01	HDFS를 옵션으로 변경, 기타 문제 수정	송지원
v6r1	2016/06/03	실습자 검수 후 수정	송지원, 윤희범, 남택호
v6r2	2016/06/29	HDFS 설치 과정 등 수정	송지원
v6r3	2016/06/30	Zeppelin 독립 실행 모드 설명 추가	송지원

# CSLab: Cluster LAB

## - Goal

SETUP to run data processing and visualization

- Install Mesos, HDFS, Spark, Zeppelin on NUC



# Apache Mesos

## - Concept



### What is Mesos?

#### A distributed systems kernel

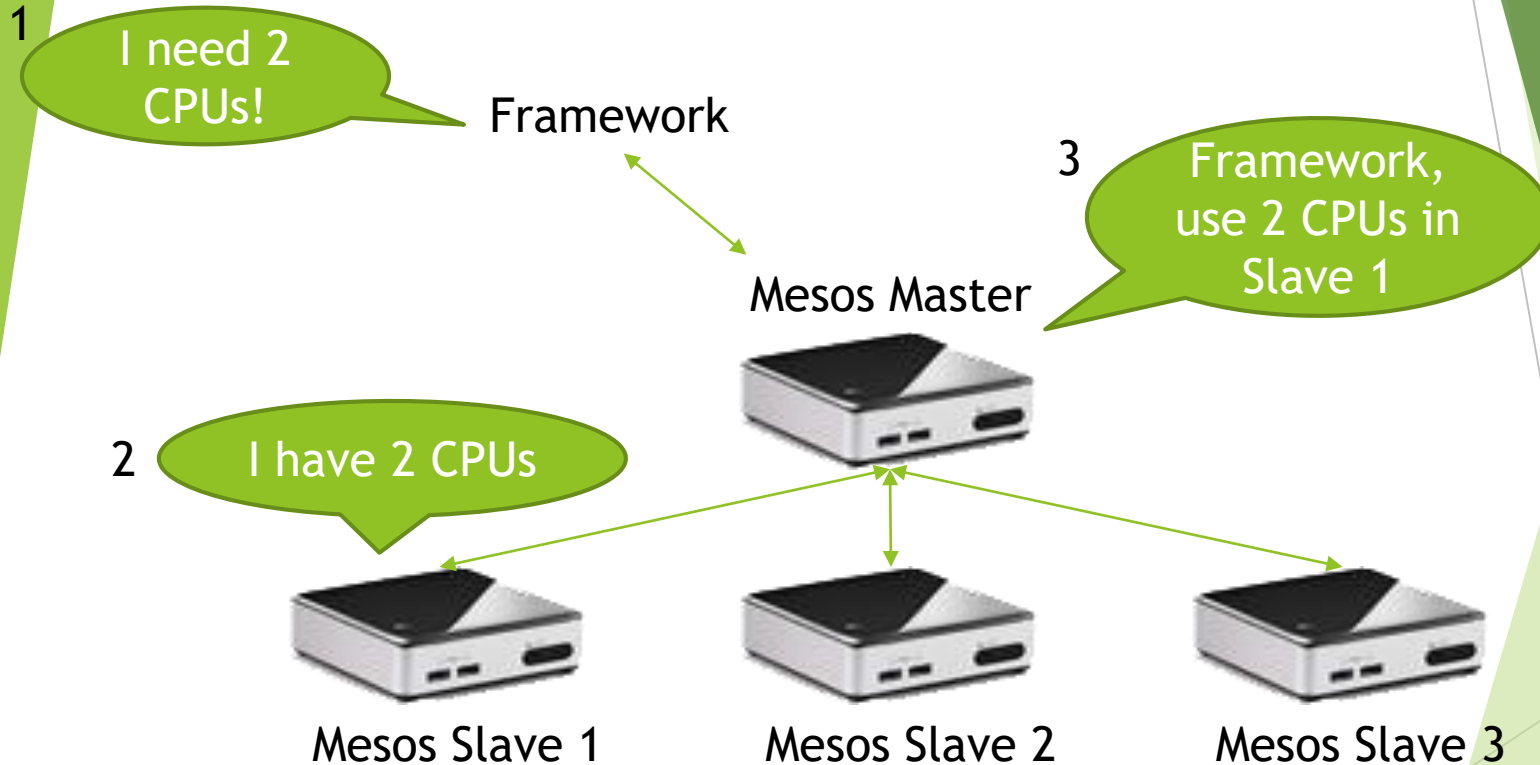
Mesos is built using the same principles as the Linux kernel, only at a different level of abstraction. The Mesos kernel runs on every machine and provides applications (e.g., Hadoop, Spark, Kafka, Elastic Search) with API's for resource management and scheduling across entire datacenter and cloud environments.

- Cloud as a single computer
- Share resources across the machines



# Apache Mesos

## - Architecture



# HDFS

## - Concept

### Hadoop Distributed FileSystem

- A distributed file system that provides high-throughput access to application data.

### Features

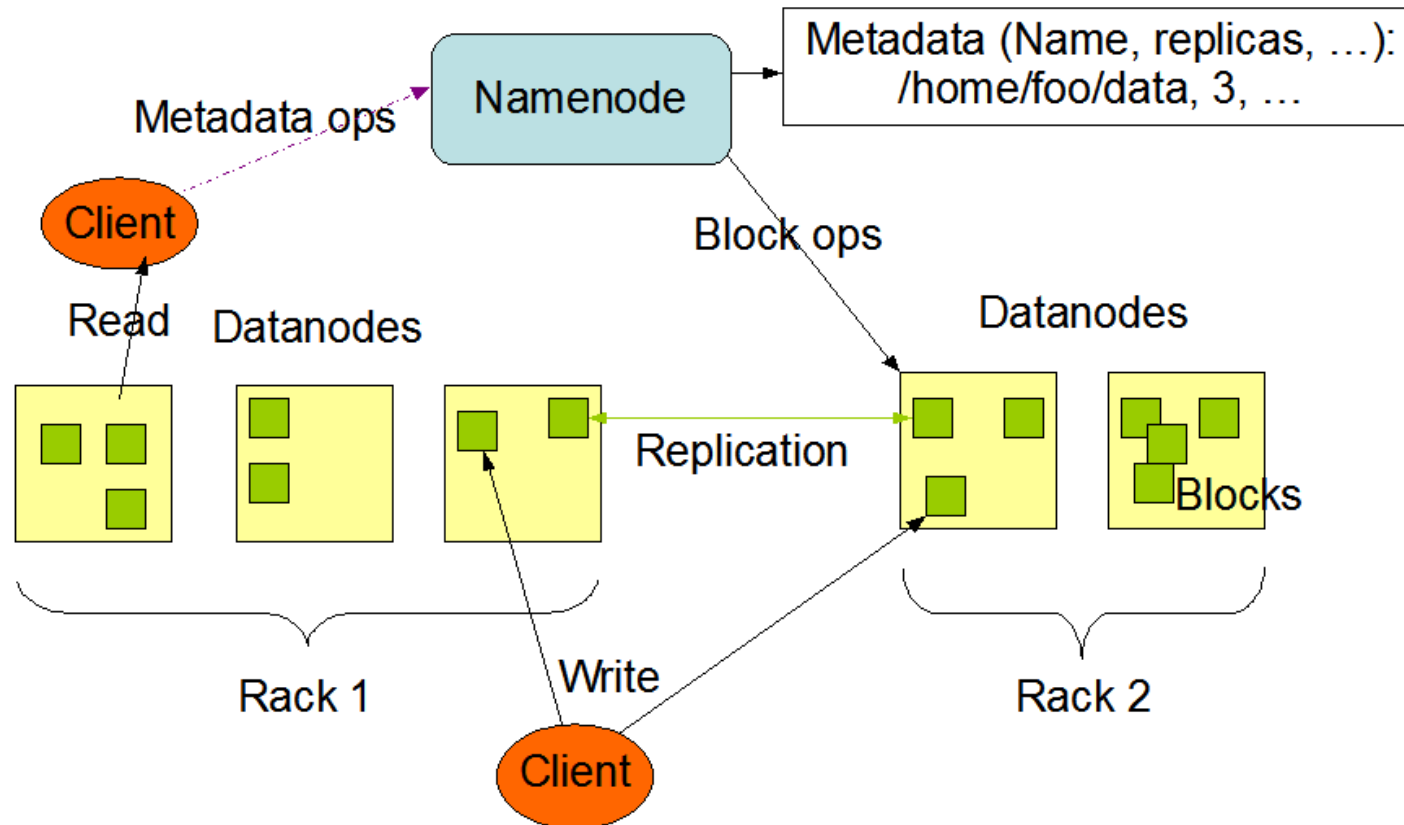
- **Fault tolerance** by detecting faults and applying quick, automatic recovery
- **Portability** across heterogeneous commodity hardware and operating systems
- **Scalability** to reliably store and process large amounts of data
- **Economy** by distributing data and processing across clusters of commodity personal computers
- **Efficiency** by distributing data and logic to process it in parallel on nodes where data is located
- **Reliability** by automatically maintaining multiple copies of data and automatically redeploying processing logic in the event of failures

# HDFS

## - Architecture

### <Master/Slave architecture>

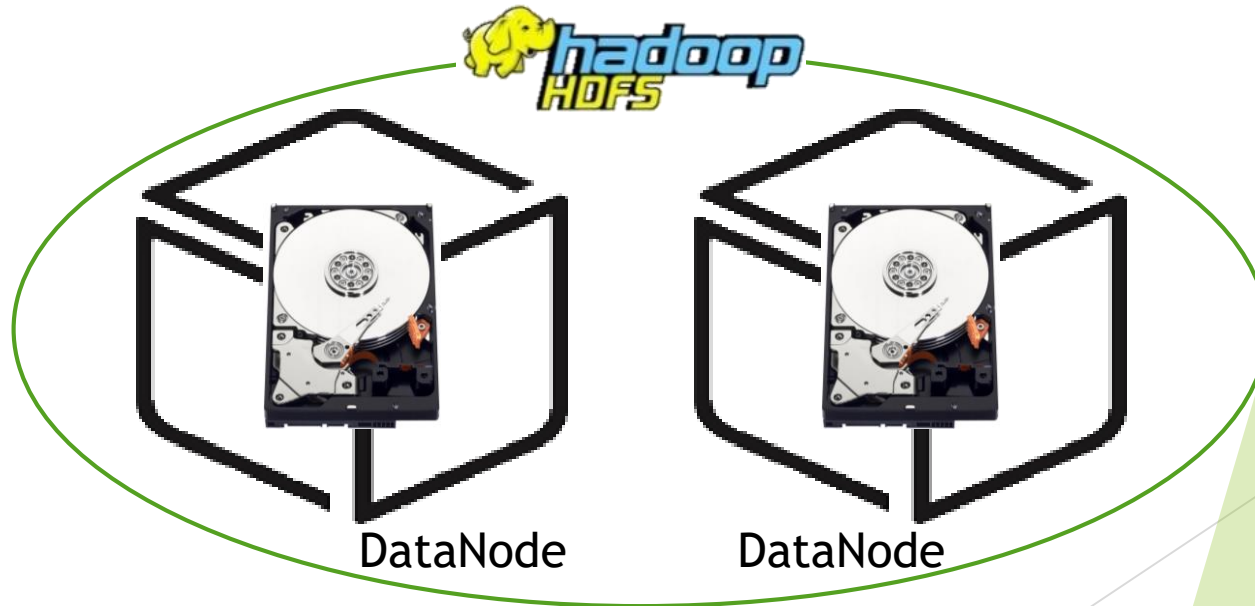
- NameNode: A single node which manages the file system namespace and regulates access to files by clients.
- DataNode: DataNodes manage storage attached to the nodes that they run on.



# HDFS

## - Architecture

HDFS makes storages of separate machines in cluster into a single storage.





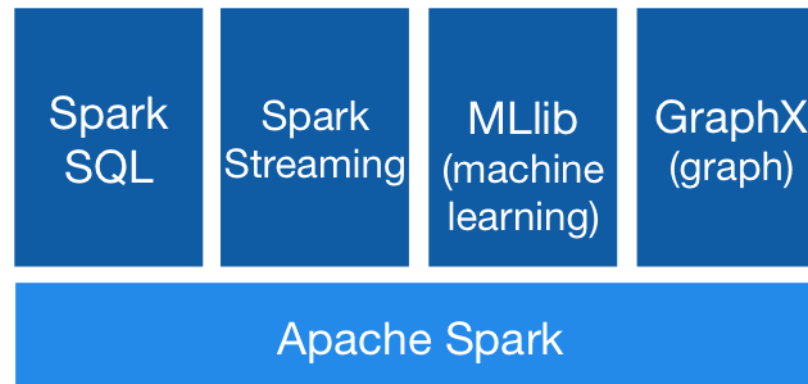
# Apache Spark

## - Concept



**Apache Spark™** is a fast and general engine for large-scale data processing.

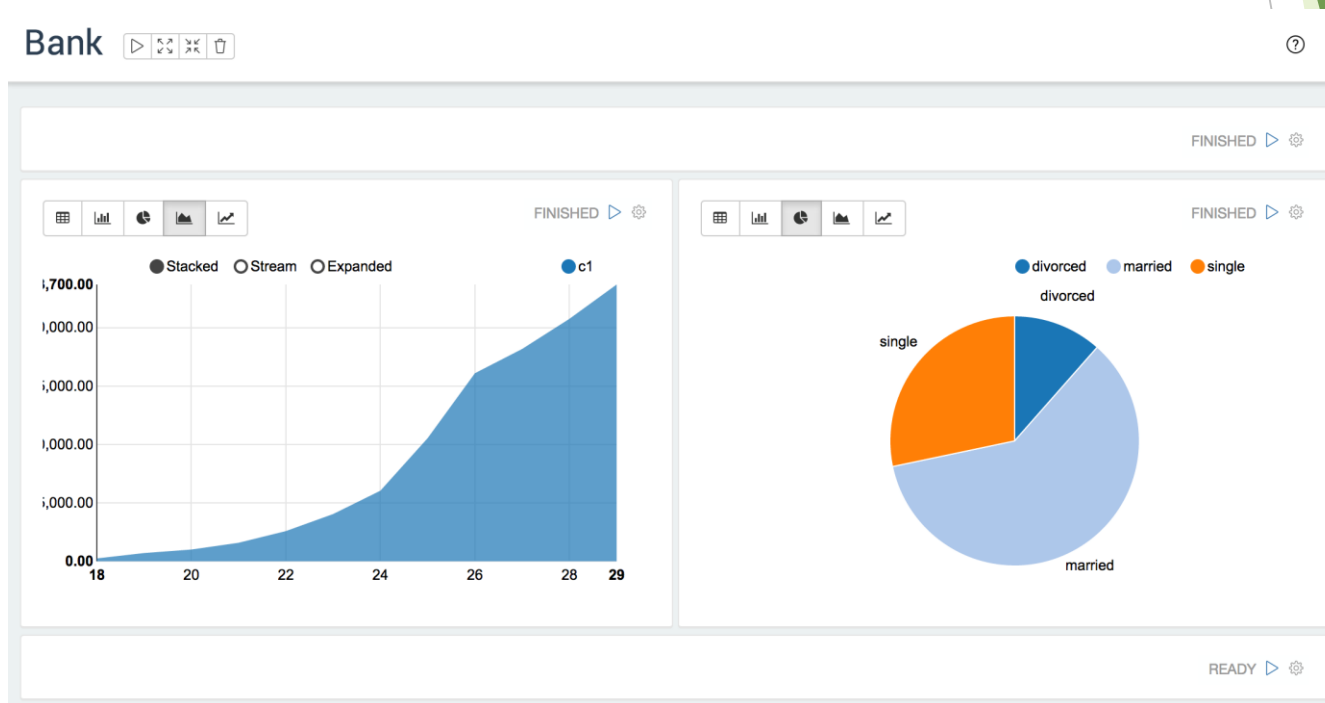
- In-memory data processing framework: Fast!
- Easy to use, community fastly growing
- Libraries: SQL and DataFrame, Streaming, MLlib, GraphX
- Run on standalone or Mesos, Yarn, etc
- Scala, Java, Python



# Apache Zeppelin -Concept

A web-based notebook that enables interactive data analytics.

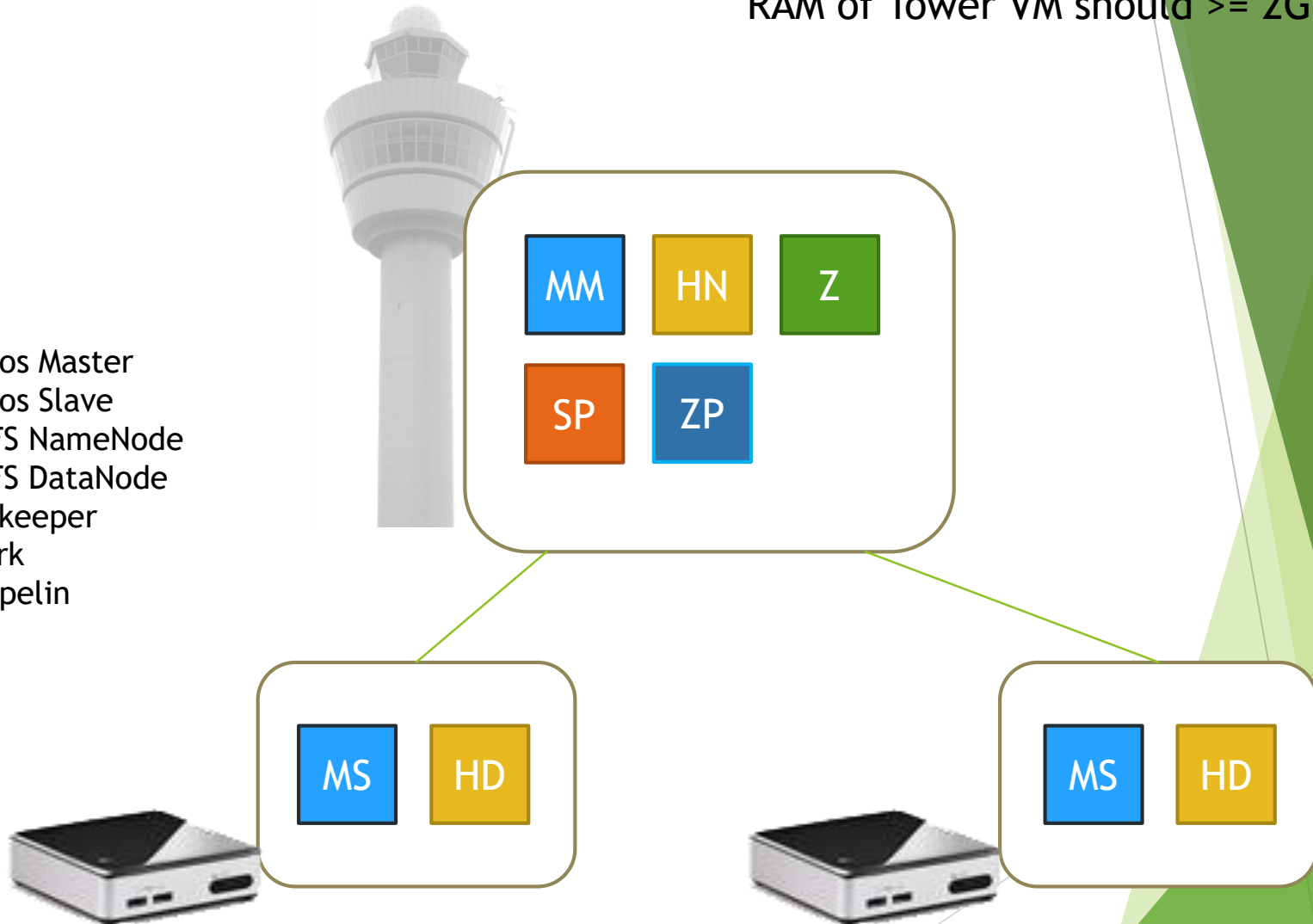
Support Spark



# 0. Cluster Overview

Prerequisite: Ubuntu 14.04 - 64bit.  
(To install Java, see AppServer Lab.)  
RAM of Tower VM should  $\geq$  2GB.

- MM Mesos Master
- MS Mesos Slave
- HN HDFS NameNode
- HD HDFS DataNode
- Z Zookeeper
- SP Spark
- ZP Zeppelin



# 0. Preparation

## - Install Java



### Install JAVA JDK 8

```
$ sudo add-apt-repository ppa:webupd8team/java  
$ sudo apt-get update  
$ sudo apt-get install oracle-java8-installer
```

If fail to install JAVA like

error: sudo add-apt-repository: command not found

Follow command

```
$ sudo apt-get install software-properties-common python-software-properties  
$ sudo apt-get update
```

Do this for all tower and NUCs.

If you already installed Java 8 in previous labs, you can skip it.

# 0. Preparation

## - Configure accounts

Do this for all tower and NUCs.

### 1. Set root password

- `sudo passwd`

### 2. Create Hadoop account and login

- `sudo -s`
- `adduser hadoop`
- `adduser hadoop sudo`
- `su hadoop`      *#login as 'hadoop'*

This procedure is required when you install HDFS.



# 0. Preparation

## - Configure hostnames

Do this for all tower and NUCs.

### 1. Edit /etc/hosts

- `sudo vi /etc/hosts`

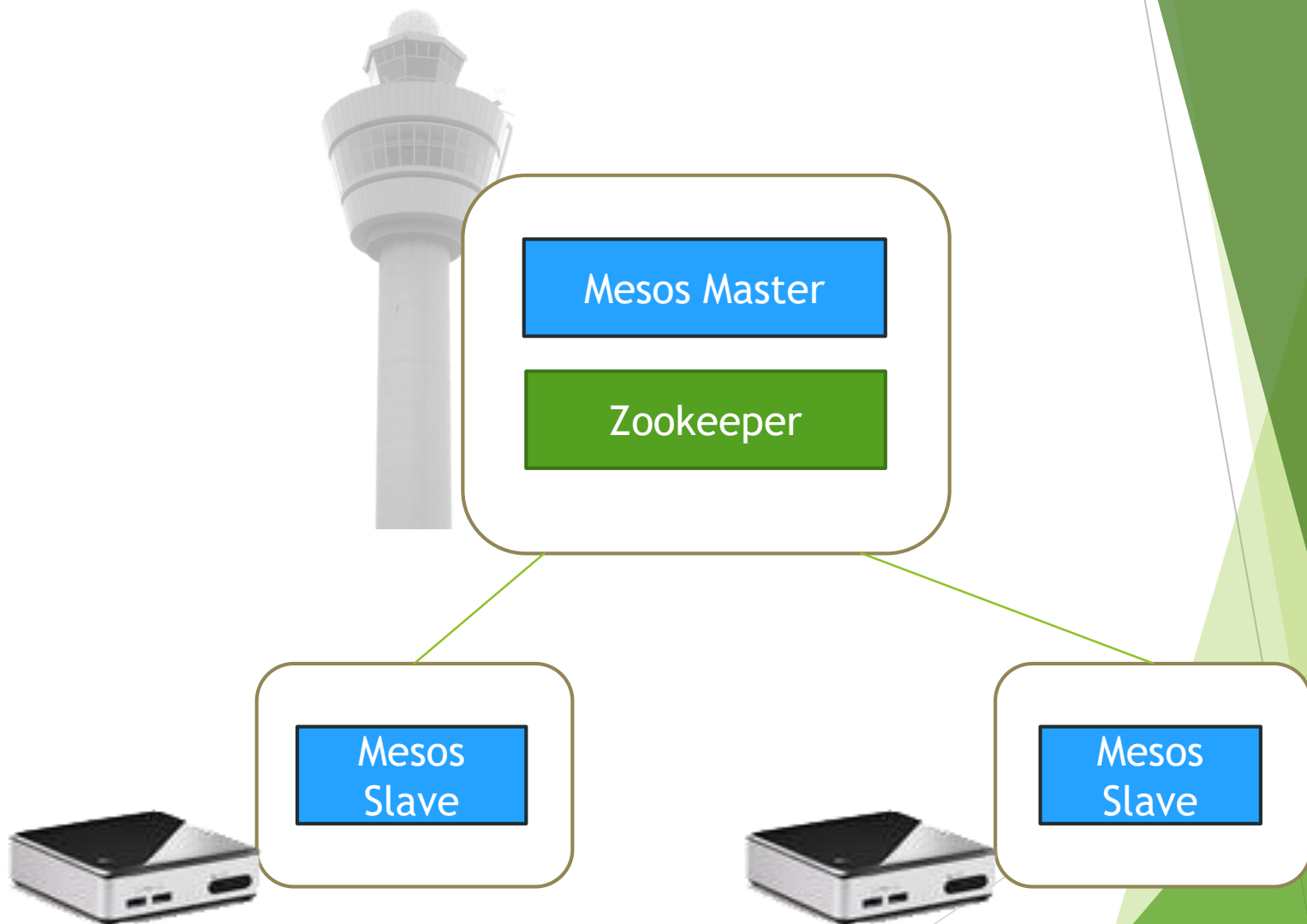
Edit:

```
EX) 127.0.0.1      localhost
     192.168.0.1    tower
     192.168.0.2    nuc01
     192.168.0.3    nuc02
```



# 1. Apache Mesos

- Install



# 1. Apache Mesos

## - Installation Procedure

1. Add Mesosphere repository
2. Install Mesos Master
3. Install Mesos Slave
4. Check on the Web UI



# 1. Apache Mesos

## - Install: Add Mesosphere repository

Add the repository to Tower and NUCs.



```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv E56151BF
```

```
DISTRO=$(lsb_release -is | tr '[:upper:]' '[:lower:]')  
CODENAME=$(lsb_release -cs)
```

To check you correctly input, use echo command.

```
echo $DISTRO $CODENAME
```

```
ubuntu trusty
```

```
echo "deb http://repos.mesosphere.io/${DISTRO} ${CODENAME} main" | sudo  
tee /etc/apt/sources.list.d/mesosphere.list
```

```
sudo apt-get -y update
```

# 1. Apache Mesos

## - Install: Mesos Master

```
sudo apt-get -y install mesos  
sudo reboot
```

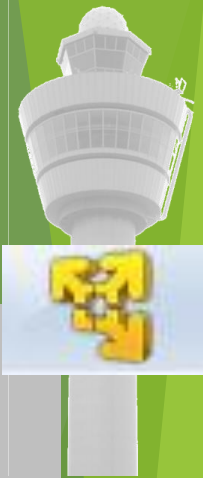
```
sudo service mesos-slave stop
```

```
echo manual | sudo tee /etc/init/mesos-slave.override  
echo <TOWER_IP_ADDR> | sudo tee /etc/mesos-master/ip  
echo <TOWER_HOSTNAME> | sudo tee /etc/mesos-master/hostname  
echo zk://<TOWER_IP_ADDR>:2181/mesos | sudo tee /etc/mesos/zk  
echo <NAME> | sudo tee /etc/mesos-master/cluster
```

```
sudo service zookeeper restart  
sudo service mesos-master restart
```

```
echo 1 | sudo tee /etc/zookeeper/conf/myid
```

NAME: anything you want



# 1. Apache Mesos

## - Install: Mesos Slave

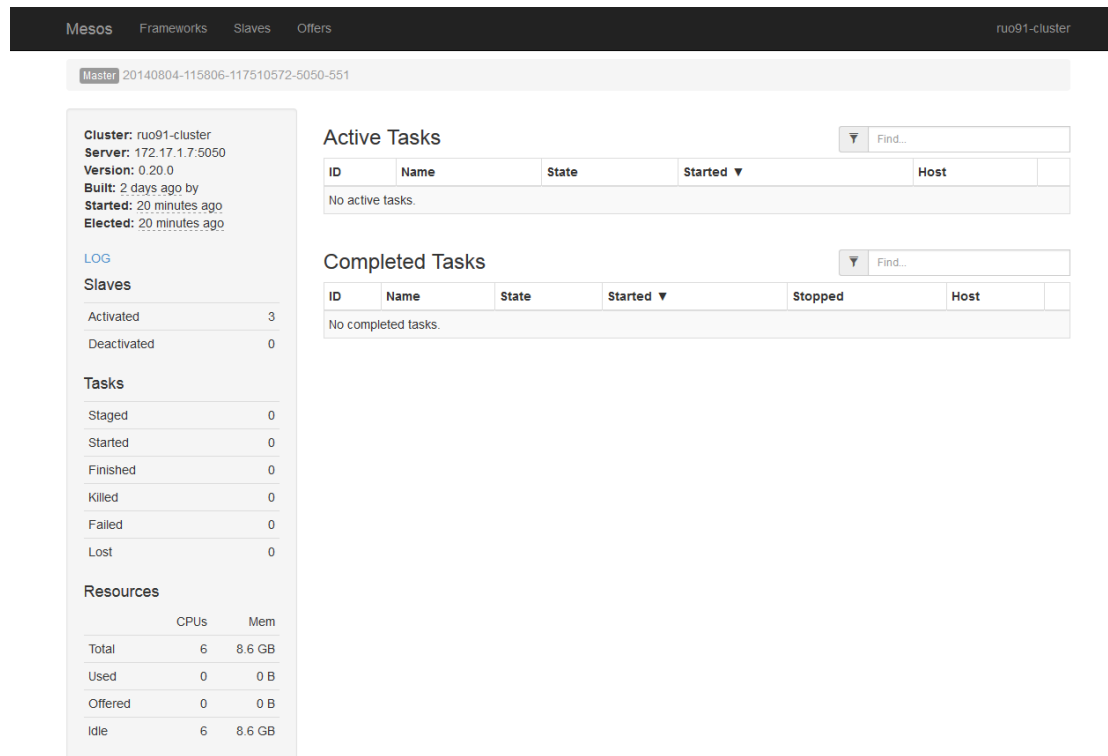


- `sudo apt-get -y install mesos`
  - `sudo reboot`
  - `sudo service mesos-master stop`
  - `echo manual | sudo tee /etc/init/mesos-master.override`
  - `sudo service zookeeper stop`
  - `echo manual | sudo tee /etc/init/zookeeper.override`
  - `sudo apt-get -y remove --purge zookeeper`
  - `echo <NUC_IP_ADDR> | sudo tee /etc/mesos-slave/ip`
  - `echo <NUC_HOSTNAME> | sudo tee /etc/mesos-slave/hostname`
  - `echo zk://<TOWER_IP_ADDR>:2181/mesos | sudo tee /etc/mesos/zk`
  - `echo HADOOP_HOME=/usr/local/hadoop | sudo tee -a /etc/default/mesos-slave`
  - `sudo reboot`
- HADOOP\_HOME** will be needed later if you want to use Spark with HDFS.

# 1. Apache Mesos

## - Check on the Web UI

In your web browser, go to  
`http://<MASTER-IP-ADDR>:5050`



The screenshot displays the Apache Mesos Web UI for a cluster named 'ruo91-cluster'. The interface includes a navigation bar with links to Mesos, Frameworks, Slaves, and Offers. The main content area is divided into several sections:

- Cluster Information:** Displays the cluster name 'ruo91-cluster', server IP '172.17.1.7:5050', version '0.20.0', and build time '2 days ago by'. It also shows the start time '20 minutes ago' and election time '20 minutes ago'.
- LOG:** A link to view the cluster log.
- Slaves:** A table showing the status of slaves in the cluster.
- Tasks:** A table showing the status of tasks in the cluster.
- Resources:** A table showing the total, used, offered, and idle resources.
- Active Tasks:** A table showing the status of active tasks.
- Completed Tasks:** A table showing the status of completed tasks.

The 'Slaves' table shows 3 activated slaves and 0 deactivated slaves. The 'Tasks' table shows 0 staged, 0 started, 0 finished, 0 killed, 0 failed, and 0 lost tasks. The 'Resources' table shows 6 CPUs and 8.6 GB of memory. The 'Active Tasks' and 'Completed Tasks' tables are currently empty.

Mesos				Frameworks				Slaves				Offers				ruo91-cluster			
Master 20140804-115806-117510572-5050-551																			
<b>Cluster:</b> ruo91-cluster																			
<b>Server:</b> 172.17.1.7:5050																			
<b>Version:</b> 0.20.0																			
<b>Built:</b> 2 days ago by																			
<b>Started:</b> 20 minutes ago																			
<b>Elected:</b> 20 minutes ago																			
<a href="#">LOG</a>																			
<b>Slaves</b>																			
Activated 3																			
Deactivated 0																			
<b>Tasks</b>																			
Staged 0																			
Started 0																			
Finished 0																			
Killed 0																			
Failed 0																			
Lost 0																			
<b>Resources</b>																			
CPUs Mem																			
Total 6 8.6 GB																			
Used 0 0 B																			
Offered 0 0 B																			
Idle 6 8.6 GB																			
<b>Active Tasks</b>																			
Find...																			
ID Name State Started Host																			
No active tasks.																			
<b>Completed Tasks</b>																			
Find...																			
ID Name State Started Stopped Host																			
No completed tasks.																			

Check the activated slaves and resources.



# 2. Apache Spark

## - Install



1. Download and unarchive Spark binary for all Tower and NUCs.

- `cd ~`
- `wget http://mirror.apache-kr.org/spark/spark-1.6.2/spark-1.6.2-bin-hadoop2.6.tgz`
- `tar xzf spark-1.6.2-bin-hadoop2.6.tgz`

2. On Tower, configure `spark-env.sh` file.

- `cd spark-1.6.2-bin-hadoop2.6/conf`
- `cp spark-env.sh.template spark-env.sh`
- `vi spark-env.sh`

**Edit:** `export MESOS_NATIVE_JAVA_LIBRARY=/usr/local/lib/libmesos.so`  
`export MASTER=mesos://<TOWER_IP>:5050`

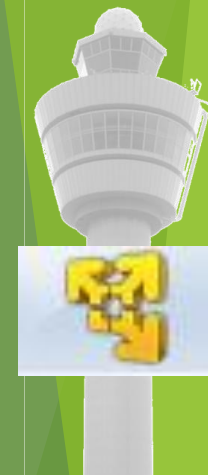
# 2. Apache Spark

## - Test

# Test Spark: In Tower

- `cd ..`
- `bin/pyspark`

Go to Mesos web UI  
and see Spark  
framework running.



```
>>> data = range(1, 10001)
>>> distData = sc.parallelize(data)
>>> distData.filter(lambda x: x < 10).collect()
```

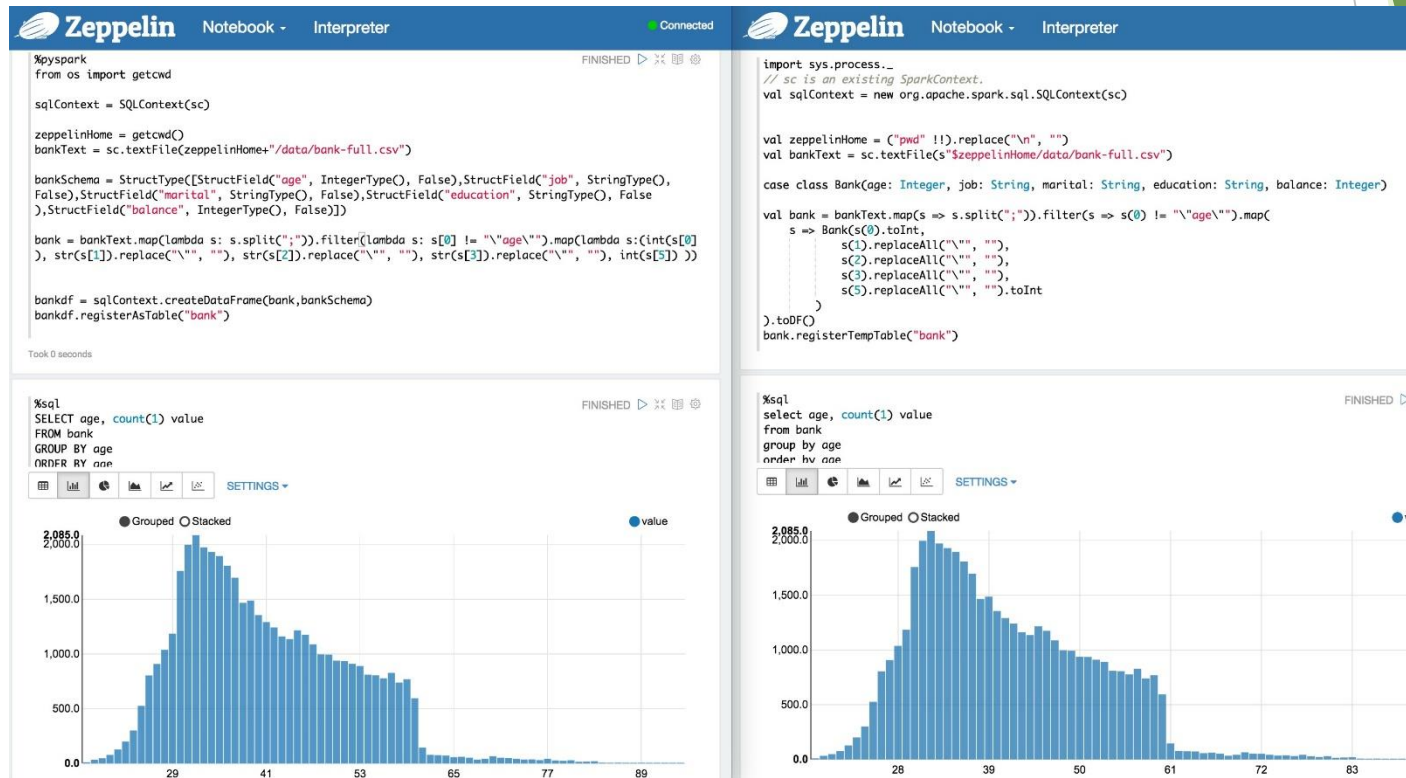
```
>>> distData.filter(lambda x: x < 10).collect()
16/06/29 16:57:41 INFO SparkContext: Starting job: collect at <stdin>:1
16/06/29 16:57:42 INFO DAGScheduler: Got job 1 (collect at <stdin>:1) with 2 output partitions
16/06/29 16:57:42 INFO DAGScheduler: Final stage: ResultStage 1 (collect at <stdin>:1)
16/06/29 16:57:42 INFO DAGScheduler: Parents of final stage: List()
16/06/29 16:57:42 INFO DAGScheduler: Missing parents: List()
16/06/29 16:57:42 INFO DAGScheduler: Submitting ResultStage 1 (PythonRDD[2] at collect at <stdin>:1), which has no missing parents
16/06/29 16:57:42 INFO MemoryStore: Block broadcast_1 stored as values in memory (estimated size 3.4 KB, free 9.1 KB)
16/06/29 16:57:42 INFO MemoryStore: Block broadcast_1_piece0 stored as bytes in memory (estimated size 2.3 KB, free 11.4 KB)
16/06/29 16:57:42 INFO BlockManagerInfo: Added broadcast_1_piece0 in memory on 192.168.88.147:37555 (size: 2.3 KB, free: 511.1 MB)
16/06/29 16:57:42 INFO SparkContext: Created broadcast 1 from broadcast at DAGScheduler.scala:1006
16/06/29 16:57:42 INFO DAGScheduler: Submitting 2 missing tasks from ResultStage 1 (PythonRDD[2] at collect at <stdin>:1)
16/06/29 16:57:42 INFO TaskSchedulerImpl: Adding task set 1.0 with 2 tasks
16/06/29 16:57:42 INFO TaskSetManager: Starting task 0.0 in stage 1.0 (TID 2, nuc08, partition 0, PROCESS_LOCAL, 17269 bytes)
16/06/29 16:57:42 INFO TaskSetManager: Starting task 1.0 in stage 1.0 (TID 3, nuc07, partition 1, PROCESS_LOCAL, 16802 bytes)
16/06/29 16:57:42 INFO BlockManagerInfo: Added broadcast_1_piece0 in memory on nuc08:40305 (size: 2.3 KB, free: 511.1 MB)
16/06/29 16:57:42 INFO TaskSetManager: Finished task 0.0 in stage 1.0 (TID 2) in 40 ms on nuc08 (1/2)
16/06/29 16:57:42 INFO BlockManagerInfo: Added broadcast_1_piece0 in memory on nuc07:33340 (size: 2.3 KB, free: 511.1 MB)
16/06/29 16:57:42 INFO TaskSetManager: Finished task 1.0 in stage 1.0 (TID 3) in 446 ms on nuc07 (2/2)
16/06/29 16:57:42 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool
16/06/29 16:57:42 INFO DAGScheduler: ResultStage 1 (collect at <stdin>:1) finished in 0.447 s
16/06/29 16:57:42 INFO DAGScheduler: Job 1 finished: collect at <stdin>:1, took 0.464302 s
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

# 3. Apache Zeppelin

## - Run Example



In Zeppelin tutorial, Press 'Run' button to test.



# 3. Apache Zeppelin

## - Tip: Zeppelin Standalone mode

If you have trouble running Zeppelin on Mesos, or have only one machine, then you can run Zeppelin in standalone mode.

If you already made configuration file, remove it first.

- `rm conf/zeppelin-env.sh`

Without any configuration, just start Zeppelin daemon.

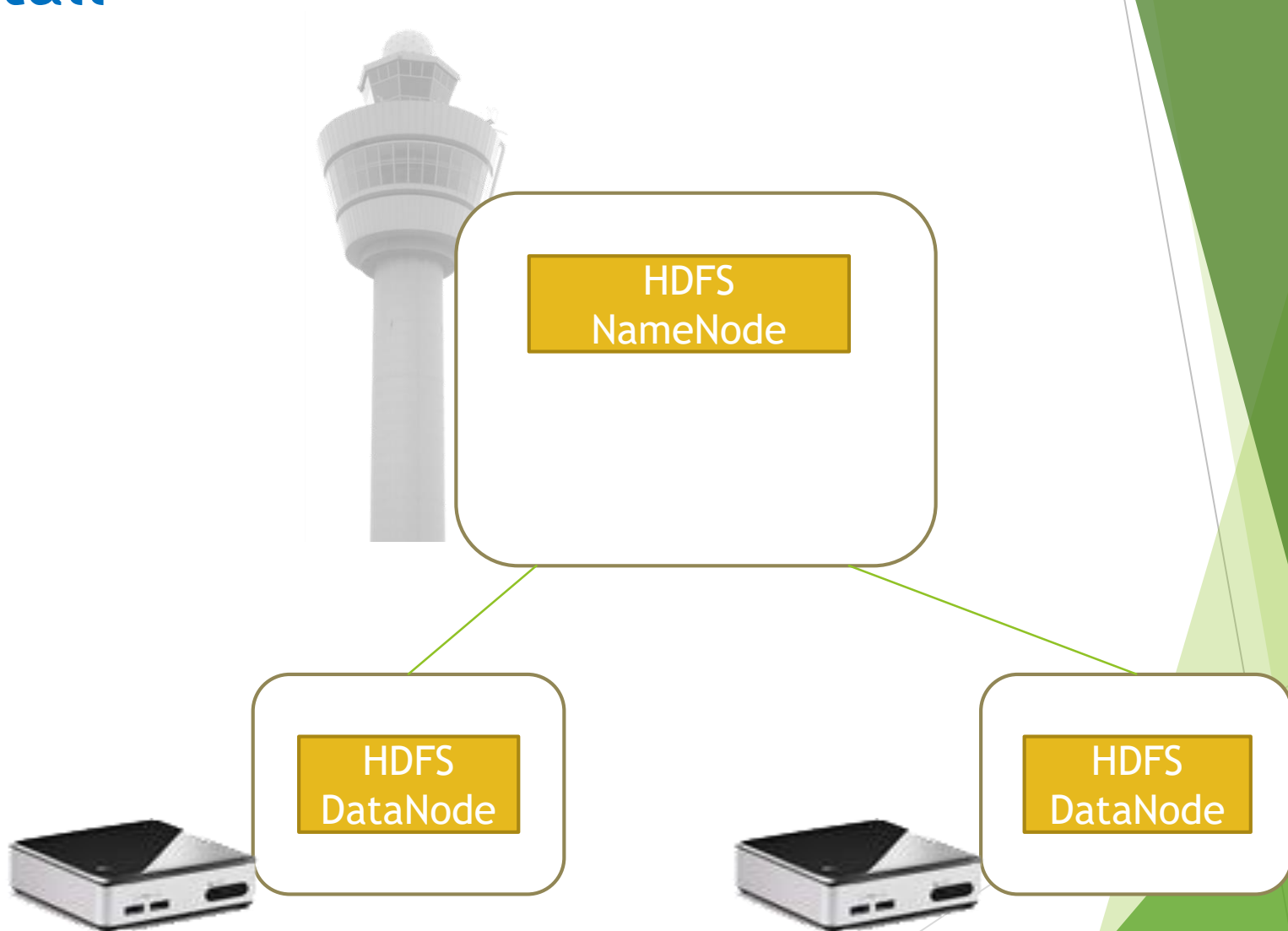
- `bin/zeppelin-daemon.sh start`

*#(or if daemon is already running, use 'restart' instead of 'start.')*



# 4. HDFS (Optional)

## - Install



# 4. HDFS (Optional)

## - Installation Procedure

1. Set hostnames
2. Configure accounts and SSH settings
3. Download and Unzip Hadoop
4. Configure HDFS
5. Start and test

# 4-1. HDFS

## - Configure accounts and SSH settings



1. Log in to user 'hadoop'.

```
• su hadoop
```

2. Generate key (just press enter x 3) in tower and NUCs

```
• ssh-keygen -t rsa  
• cp /home/hadoop/.ssh/id_rsa.pub  
  /home/hadoop/.ssh/authorized_keys
```

3. Modify key permission

```
• cd .ssh  
• chmod 644 authorized_keys
```

4. Copy key from Tower to all NUCs

```
• scp authorized_keys hadoop@<NUC_IP>:~/.ssh/
```

5. Try to login via SSH to check if you can login to NUC without password.

# 4-1. HDFS

## - Download and Unzip Hadoop



### 1. Download and Unzip in all Tower and NUCs.

- `cd`
- `wget http://mirror.apache-kr.org/hadoop/common/hadoop-2.7.2/hadoop-2.7.2.tar.gz`
- `tar -xvzf hadoop-2.7.2.tar.gz`
- `sudo mv hadoop-2.7.2 /usr/local/hadoop`

### 2. Go to the directory which contains configuration files.

- `cd /usr/local/hadoop/etc/hadoop`
- We will edit these files:  
`hadoop-env.sh, core-site.xml, hdfs-site.xml, slaves`

# 4-1. HDFS

## - Configuration



Edit these files.

1. <hadoop-env.sh> file

Edit: export JAVA\_HOME=/usr/lib/jvm/java-8-oracle

2. <core-site.xml> file

```
Edit: <configuration>
      <property>
        <name>fs.defaultFS</name>
        <value>hdfs://tower_hostname:9000/</value>
      </property>
    </configuration>
```

This option must  
be a hostname,  
not IP address.

# 4-1. HDFS

## - Configuration



### 4. <hdfs-site.xml> file

Edit: `<configuration>`  
    `<property>`  
        `<name>dfs.replication</name>`  
        `<value>2</value>`  
    `</property>`  
    `<property>`  
        `<name>dfs.namenode.name.dir</name>`  
        `<value>file:///usr/local/hadoop/namenode</value>`  
    `</property>`  
    `<property>`  
        `<name>dfs.datanode.data.dir</name>`  
        `<value>file:///usr/local/hadoop/datanode</value>`  
    `</property>`  
`</configuration>`

### 5. <slaves> file: Add hostname or IP address of all NUCs, one per line.

Ex)

nuc1  
nuc2

# 4-1. HDFS

## - Configuration



6. Deploy configuration files from Tower to all slaves.

- `cd ..`
- `scp -r hadoop hadoop@<NUC_IP>:/usr/local/hadoop/etc/`

7. In NUCs, make DataNode directory.

- `mkdir /usr/local/hadoop/datanode`

8. In all Tower and NUCs, edit /etc/environment file.

- `vi /etc/environment`

Add this line at the end of the paths, and close with ”.

`:/usr/local/hadoop/bin`

Ex) `PATH="/usr/local/sbin:/usr/local/bin:  
... :/sbin:/bin:/usr/local/hadoop/bin"`

9. Reboot.

# 4-1. HDFS

## - Start and Test

1. Login to user 'hadoop'.

- `su hadoop`

2. Format NameNode.

- `hdfs namenode -format`

3. Start HDFS.

- `hadoop/sbin/start-dfs.sh`

4. Make a directory and upload a file to HDFS to check if it is working.

- `hadoop fs -mkdir /user`
- `hadoop fs -put ~/hadoop-2.7.2.tar.gz /user/`
- `hadoop fs -ls hdfs://<TOWER_IP>:9000/user/`

# Try the last command  
on both tower and slaves.

You can also see on the web:  
[http://<TOWER\\_IP>:50070](http://<TOWER_IP>:50070)





# 4-2. Apache Spark with HDFS

## - Configuration



In home directory where Spark directory is located

- `hadoop fs -put spark-1.6.2-bin-hadoop2.6.tgz /user/`
- `cd spark-1.6.2-bin-hadoop2.6/conf`
- `vi spark-env.sh`

**Edit:** `export MESOS_NATIVE_JAVA_LIBRARY=/usr/local/lib/libmesos.so`  
`export MASTER=mesos://<TOWER_IP>:5050`  
`export SPARK_EXECUTOR_URI=hdfs://<TOWER_IP>:9000/user/spark-1.6.2-bin-hadoop2.6.tgz`

### # Test Spark

- `cd ..`
- `bin/pyspark`

```
>>> data = range(1, 10001)
>>> distData = sc.parallelize(data)
>>> distData.filter(lambda x: x < 10).collect()
```

Go to Mesos web UI and see Spark framework running.

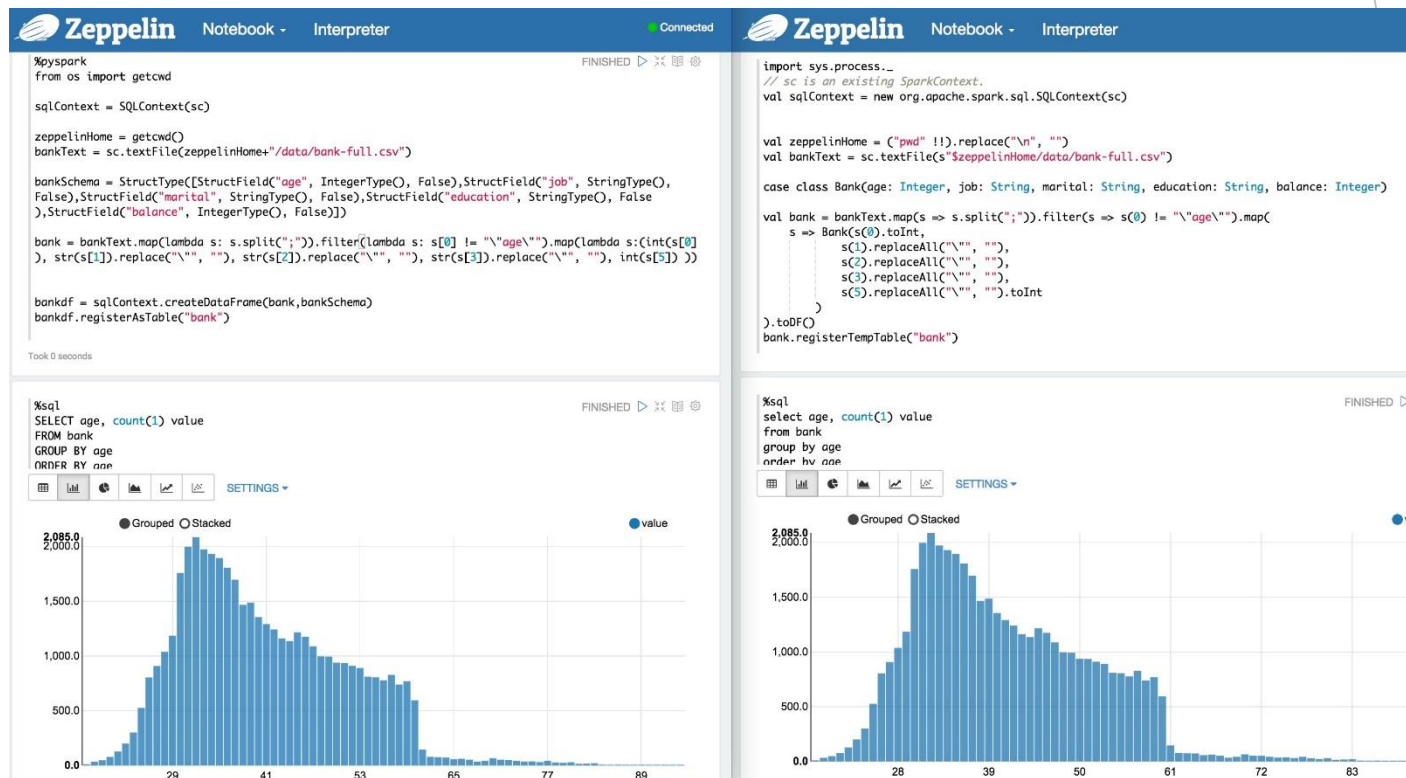
# 3. Apache Zeppelin

## - Run Example

In Zeppelin directory,

- `bin/zeppelin-daemon.sh restart`

In Zeppelin tutorial, Press 'Run' button to test changed configurations.



Thank You for  
Your Attention  
Any Questions?

