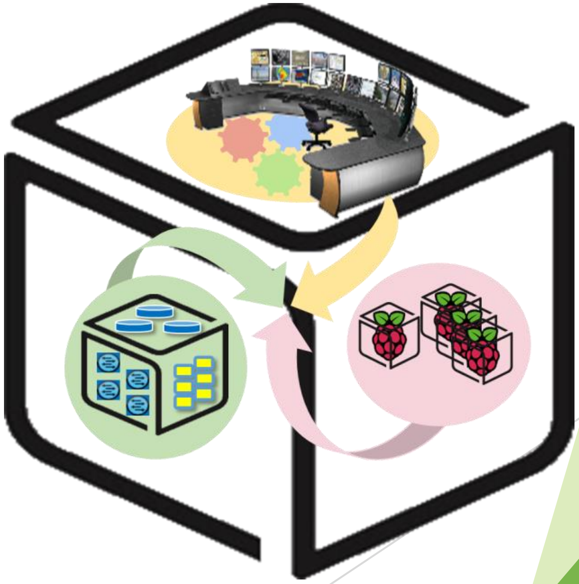
SmartX Labs
for Computer Systems

Cluster &
Analytics Lab

(2018, Spring)

NetCS Lab



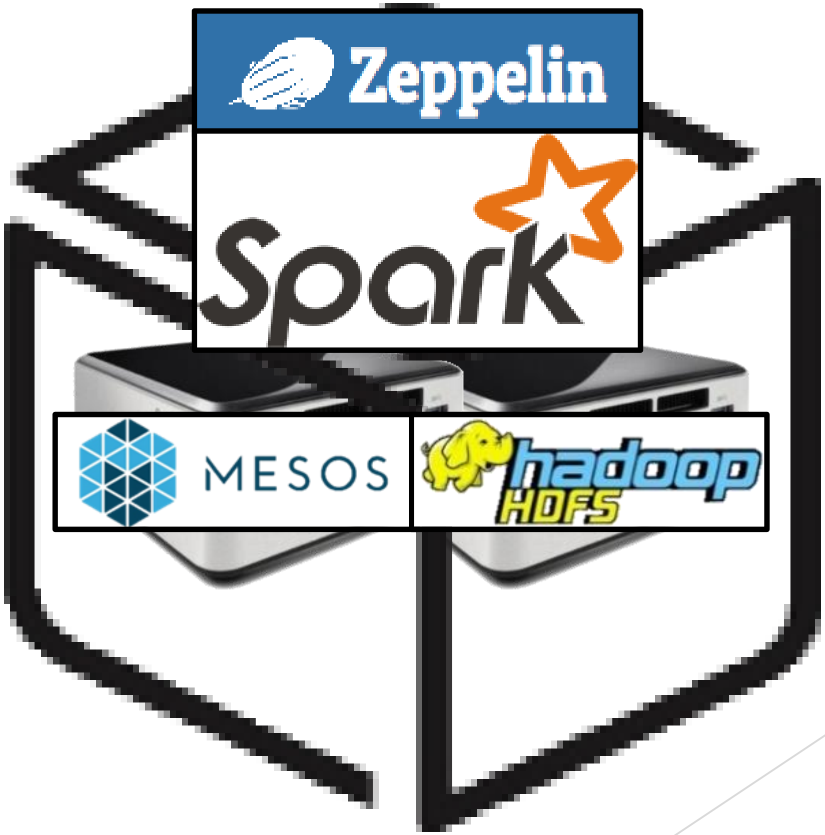
History and Contributor of Cluster Lab (2017. 05. 20.)

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
0.6.4	2016/07/04	Zeppelin 설치 방법 누락된 부분 추가	송지원
0.6.5	2017/05/20	Ubuntu 16.04, Spark 2.1.1-Hadoop-2.7, Zeppelin 0.7.1, Hadoop 2.8.0 대응 업데이트	강문중
0.6.6	2018/05/30	소프트웨어 버전 업데이트, 강의 시나리오 반영하여 그림 변경, 이론 보강	권진철

CSLab: Cluster & Analytics Lab. - Goal

SETUP to run data processing and visualization

- with Mesos, Spark, Zeppelin, (HDFS)



Apache Mesos
- Concept

MESOS

What is Mesos?
Apache Mesos is an open-source project to manage **computer clusters.**

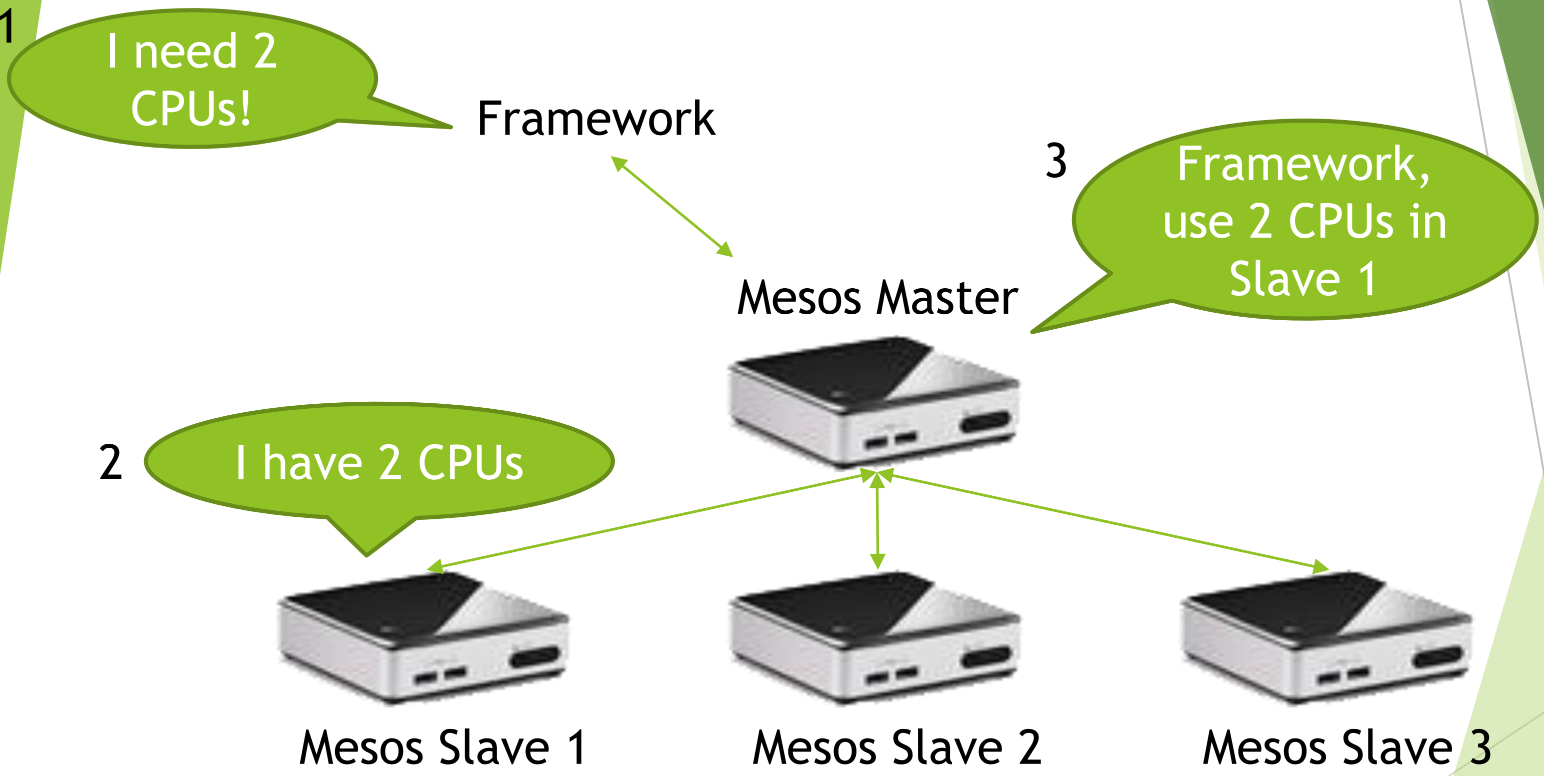
Mesos abstracts CPU, memory, storage, and other compute resources away from machines (physical or virtual), enabling fault-tolerant and elastic distributed systems to easily be built and run effectively.

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

MESOS

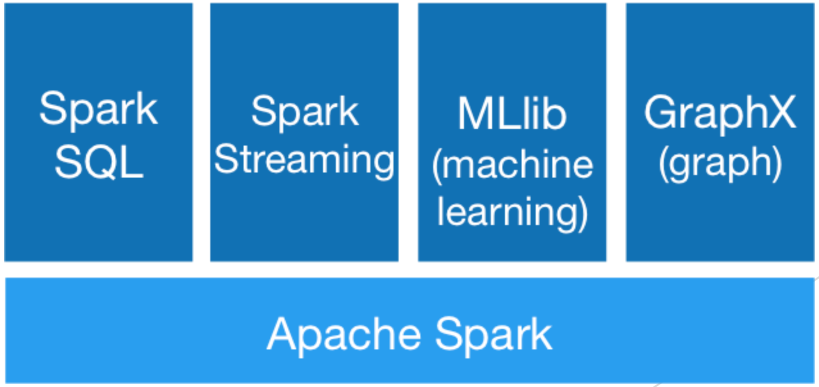


Apache Spark
- Concept

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



Spark SQL

Spark Streaming

MLlib (machine learning)

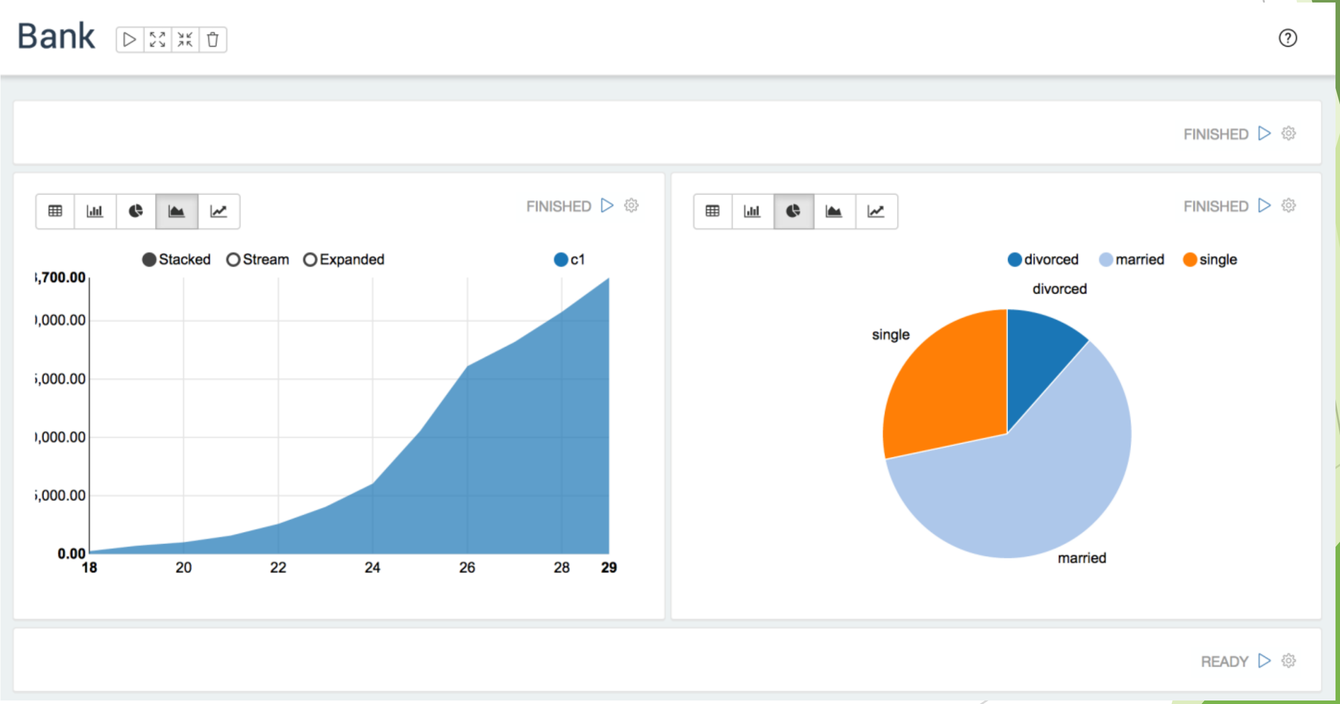
GraphX (graph)

Apache Spark

Apache Zeppelin
-Concept

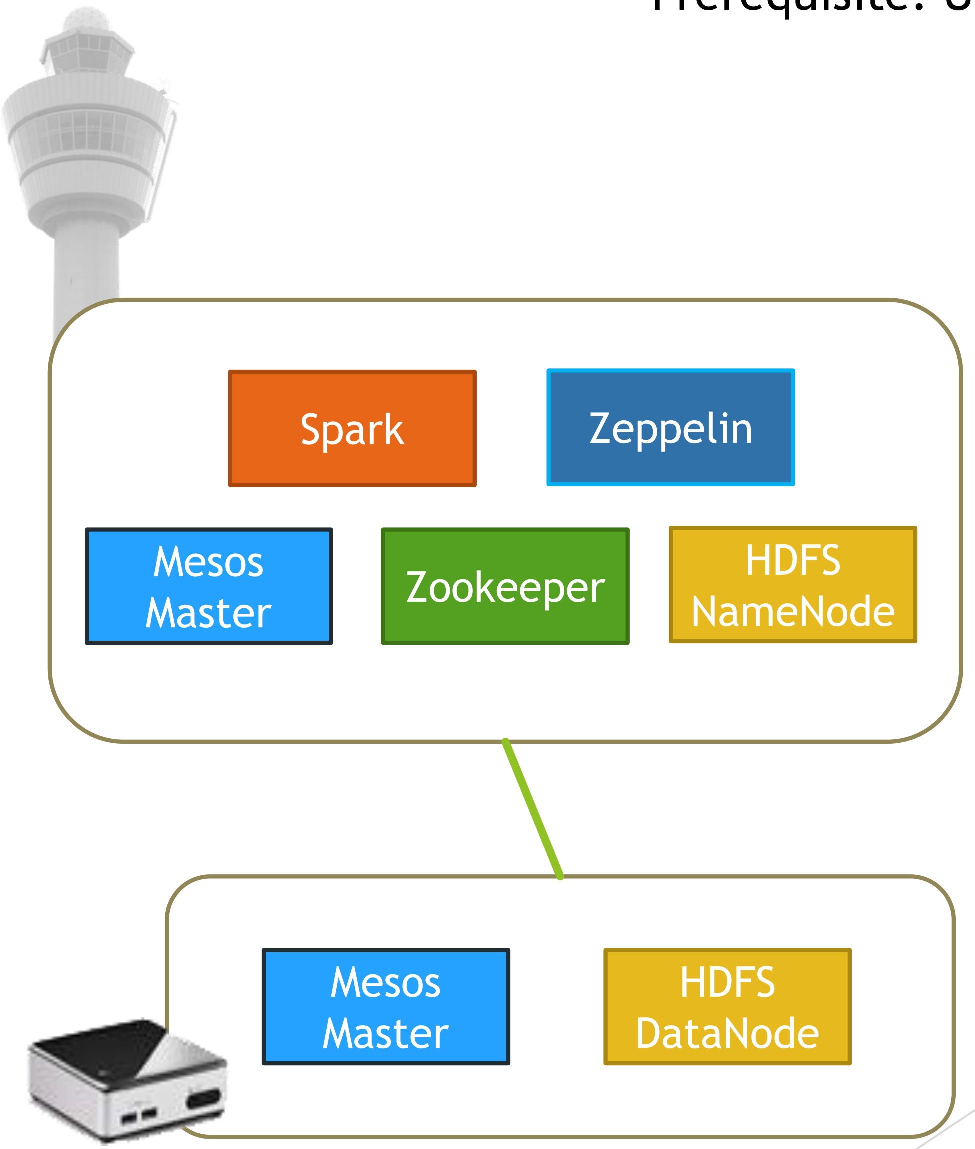
A web-based notebook that enables interactive data analytics.

Support Spark



0. Cluster Overview

Prerequisite: Ubuntu 16.04 - 64bit



HDFS (optional)
- Concept

Hadoop Distributed FileSystem

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

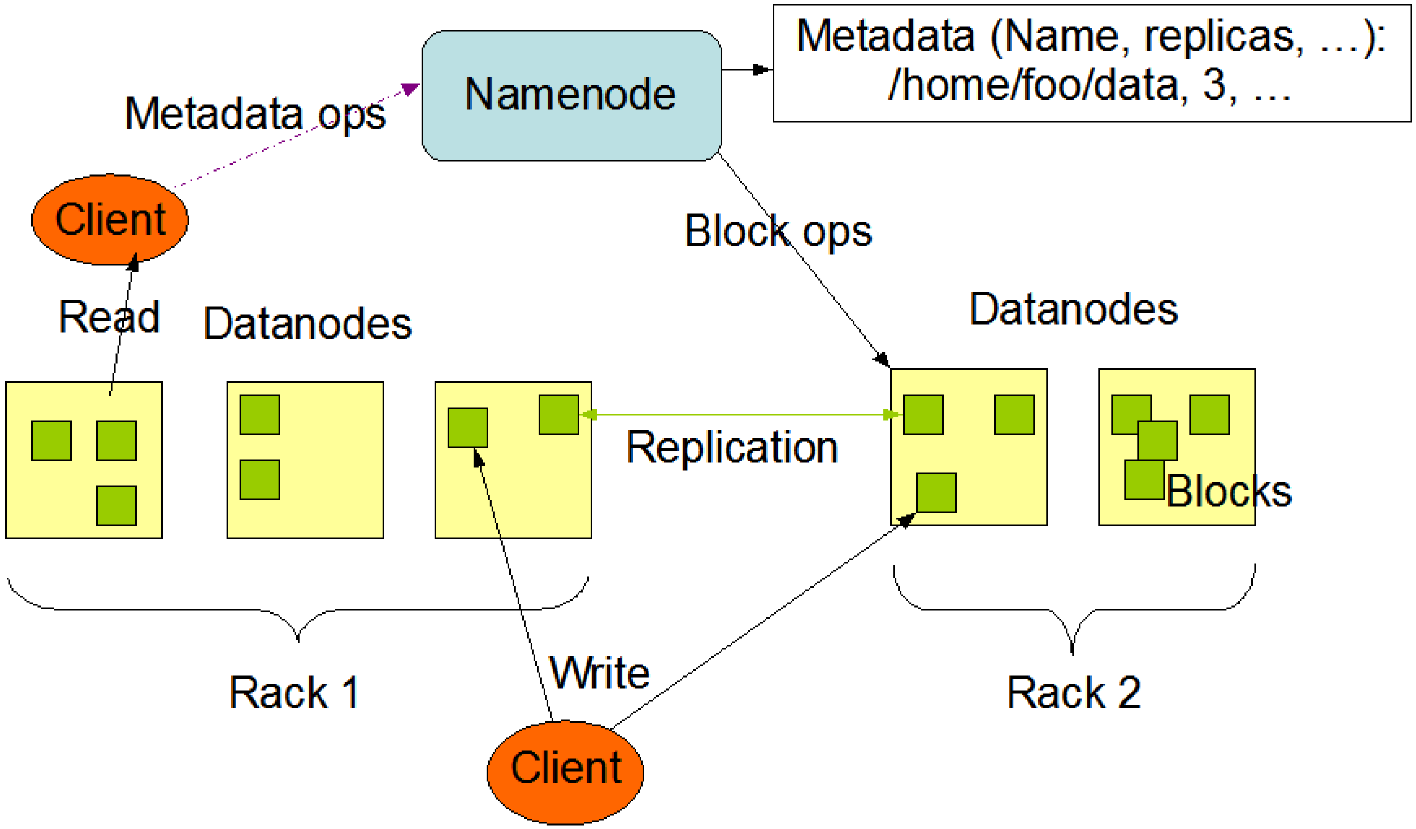
Features

- 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

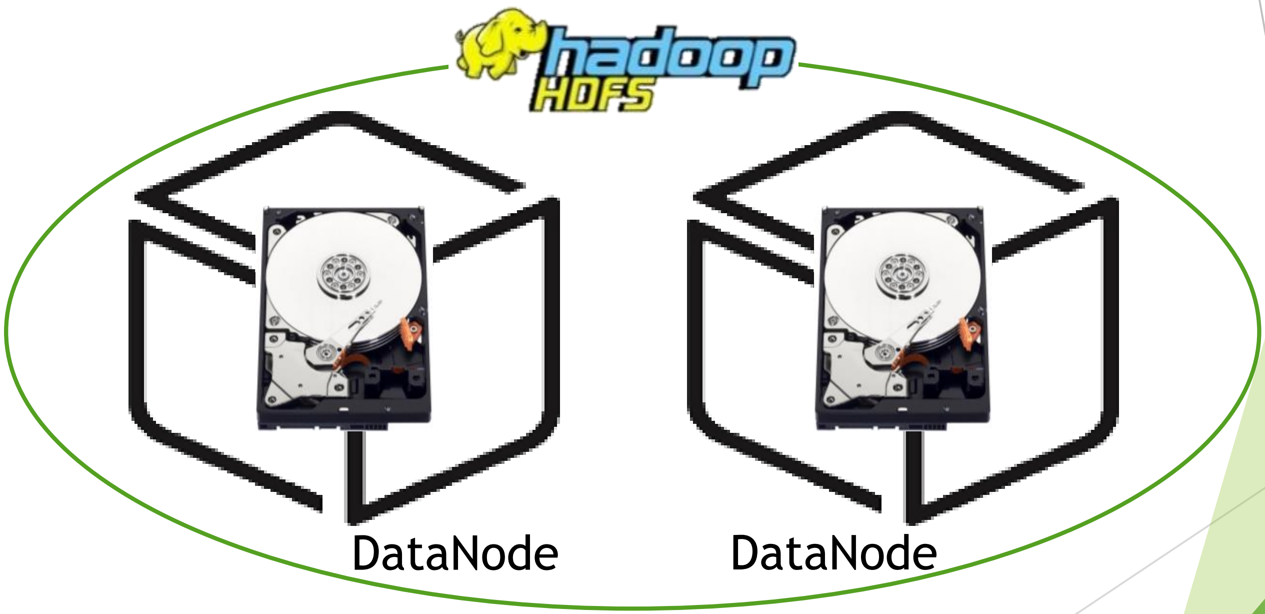
<Master/Slave architecture>

- 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.



0. Preparation
- Install Java and Mesos Dependencies

Install JDK 8 and other Apache Mesos Dependencies

```
$   sudo apt update
$   sudo apt-get install -y openjdk-8-jdk
$   sudo apt-get -y install build-essential python-dev python-six python-virtualenv
    libcurl4-nss-dev libsasl2-dev libsasl2-modules maven libapr1-dev libsvn-dev zlib1g-dev
    iputils-ping
```

Do this for **all NUCs**.

0. Preparation
- Configure hostnames

1. From **NUC 1** :

```
$  sudo hostname nuc01
```

2. From **NUC 2** :

```
$  sudo hostname nuc02
```

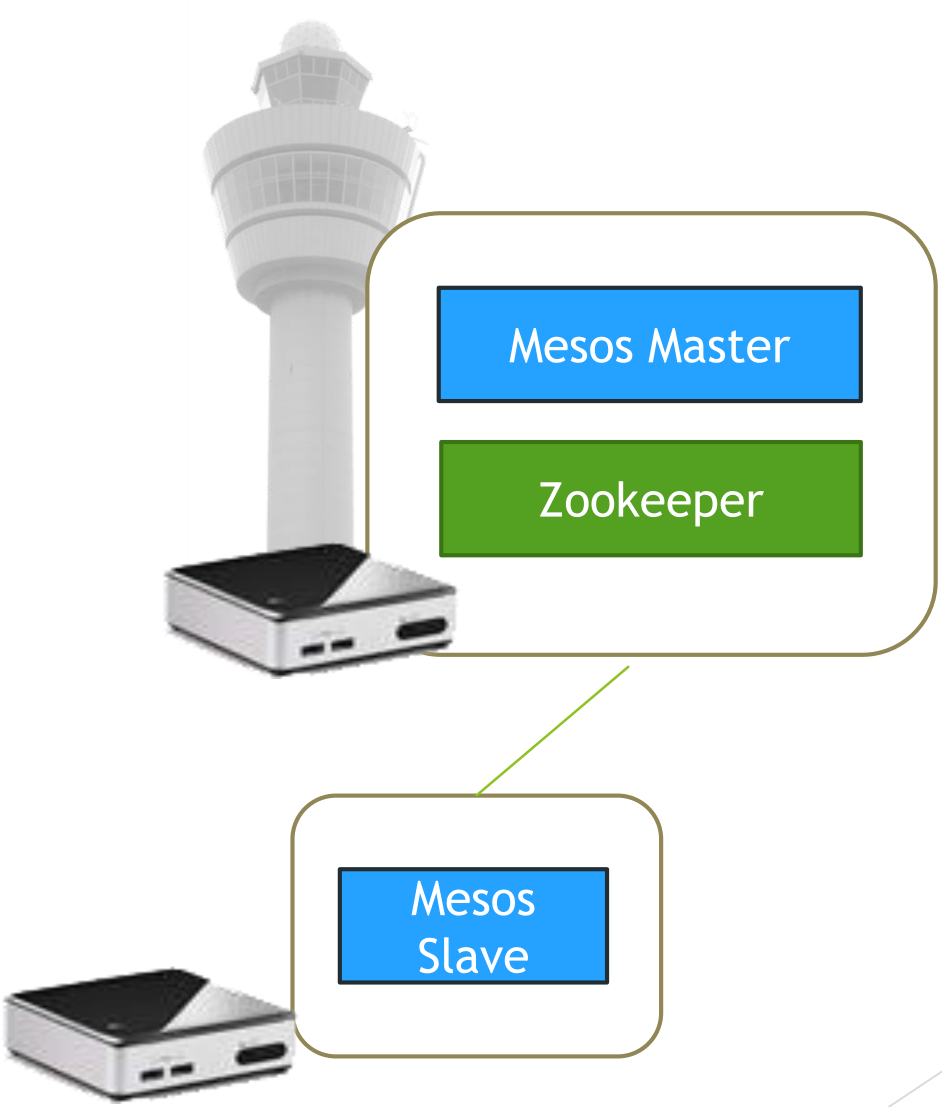
3. Edit /etc/hosts from **all NUCs** :

```
$  sudo vi /etc/hosts
```

4. Append the following context into /etc/hosts :

```
    127.0.0.1      localhost
    (IP Address of NUC 1)  nuc01
    (IP Address of NUC 2)  nuc02
```

1. Apache Mesos
- Install



1. Apache Mesos
- Installation Procedure

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

1. Apache Mesos
- Install: Add Mesosphere repository

Add the repository to **all NUCs**.

```
$  sudo apt-key adv --keyserver keyserver.ubuntu.com --recv E56151BF
$  export DISTRO=$(lsb_release -is | tr '[:upper:]' '[:lower:]')
$  export CODENAME=$(lsb_release -cs)

         To check you have correctly inputed, use the command below :
         $  echo $DISTRO $CODENAME
         Its result will be “ubuntu xenial”

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

1. Apache Mesos
- Install: Mesos Master on NUC 1

```
$ sudo apt -y install mesos

$ echo manual | sudo tee /etc/init/mesos-slave.override
$ echo 0.0.0.0 | sudo tee /etc/mesos-master/ip
$ echo nuc01 | sudo tee /etc/mesos-master/hostname
$ echo zk://localhost:2181/mesos | sudo tee /etc/mesos/zk
$ echo <NAME> | sudo tee /etc/mesos-master/cluster

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

$ sudo systemctl restart zookeeper
$ sudo systemctl start mesos-master
```

<NAME>: anything you want

1. Apache Mesos

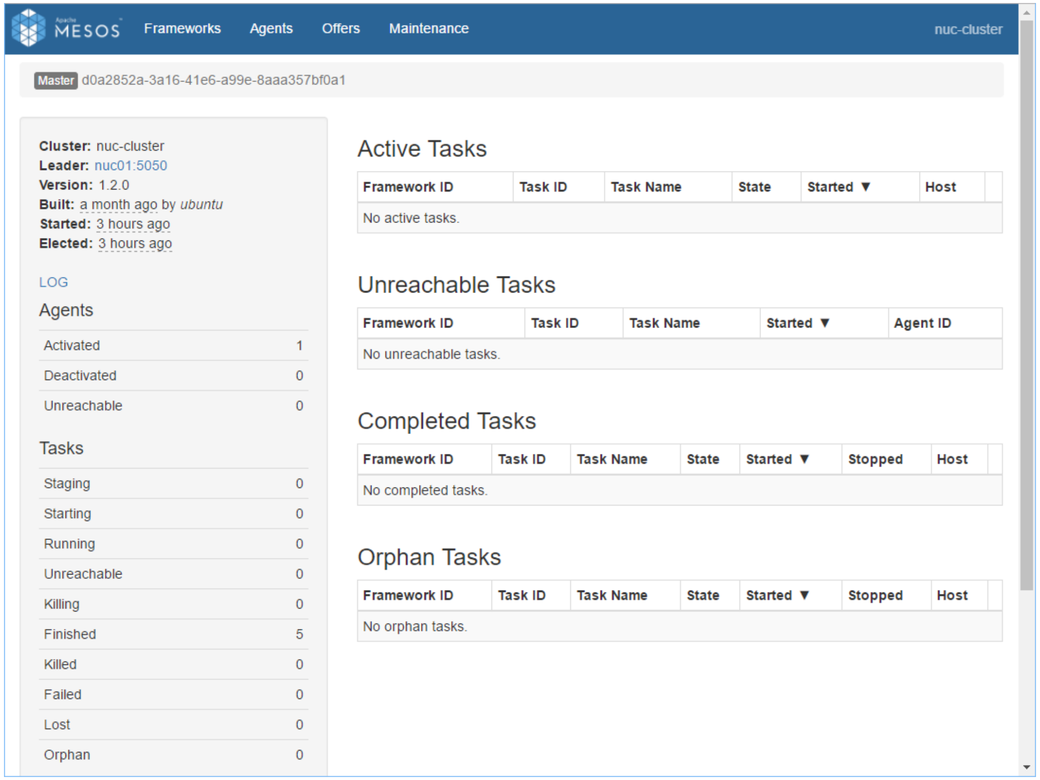
- Install: Mesos Slave on NUC 2

```
$ sudo apt -y install mesos

$ echo manual | sudo tee /etc/init/mesos-master.override
$ echo 0.0.0.0 | sudo tee /etc/mesos-slave/ip
$ echo nuc02 | sudo tee /etc/mesos-slave/hostname
$ echo zk://<NUC1 IP>:2181/mesos | sudo tee /etc/mesos/zk
$ sudo cp /etc/mesos/zk /etc/mesos-slave/master
$ echo HADOOP_HOME=/usr/local/hadoop | sudo tee –a
  /etc/default/mesos-slave

$ sudo systemctl stop zookeeper
$ sudo systemctl start mesos-slave
```

HADOOP_HOME will be needed later if you want to use Spark with HDFS.

1. Apache Mesos
- Check on Mesos Web UI

In your web browser, go to
`http://<NUC 1 IP Address>:5050`



Master d0a2852a-3a16-41e6-a99e-8aaa357bf0a1

Cluster: nuc-cluster
Leader: nuc01:5050
Version: 1.2.0
Built: a month ago by *ubuntu*
Started: 3 hours ago
Elected: 3 hours ago

LOG

Agents

Activated	1
Deactivated	0
Unreachable	0

Tasks

Staging	0
Starting	0
Running	0
Unreachable	0
Killing	0
Finished	5
Killed	0
Failed	0
Lost	0
Orphan	0

Active Tasks

Framework ID	Task ID	Task Name	State	Started ▼	Host
No active tasks.					

Unreachable Tasks

Framework ID	Task ID	Task Name	Started ▼	Agent ID
No unreachable tasks.				

Completed Tasks

Framework ID	Task ID	Task Name	State	Started ▼	Stopped	Host
No completed tasks.						

Orphan Tasks

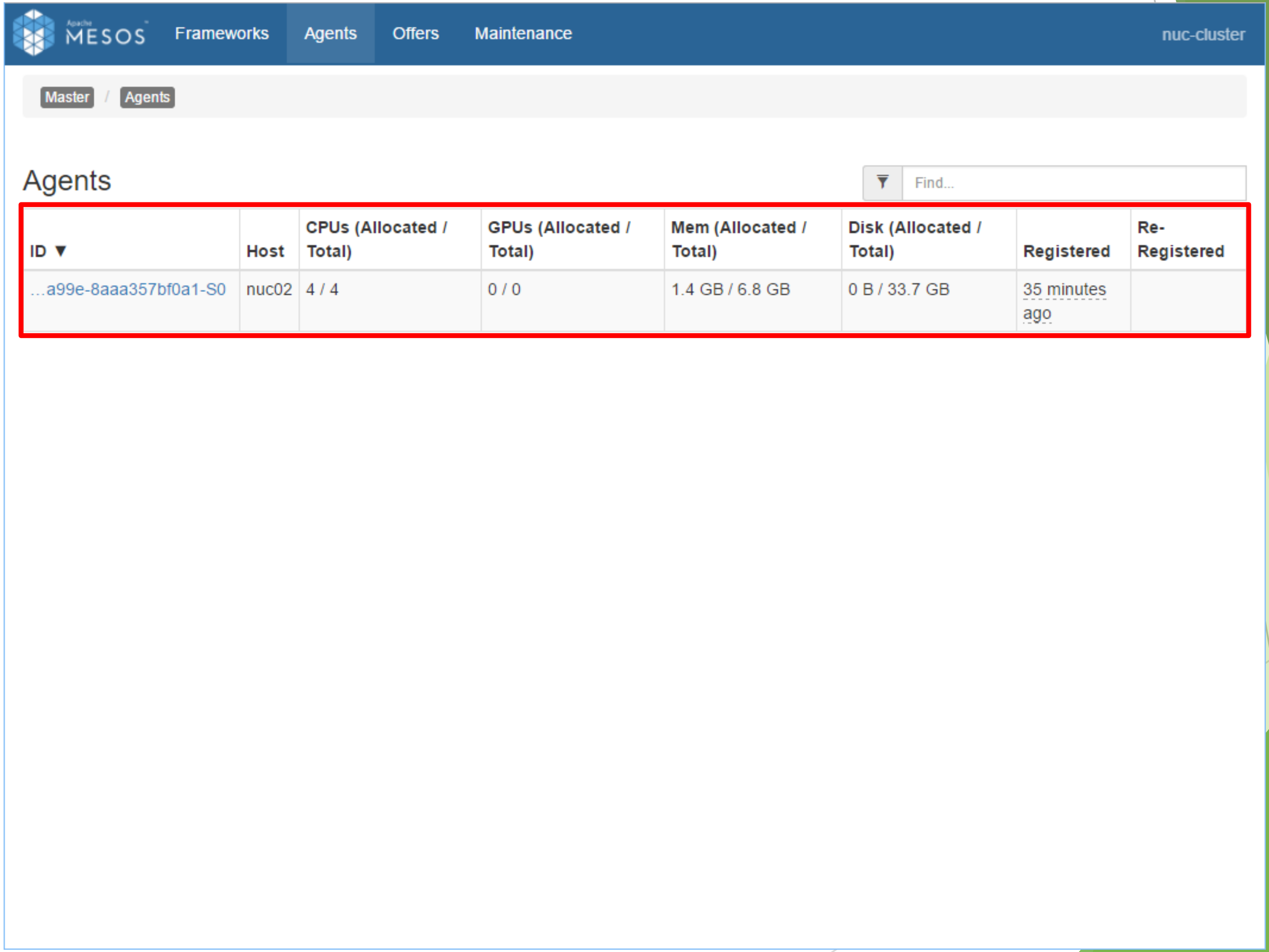
Framework ID	Task ID	Task Name	State	Started ▼	Stopped	Host
No orphan tasks.						

Check activated slaves and resources.

Note : In case of using the browser from other than those 2 NUCs, “hosts” file must be updated for the computer like the previous “Configure hostnames” step.

1. Apache Mesos
- Check on Mesos Web UI

Agents tab shows NUC 2 registered as a slave



Agents

ID ▼	Host	CPUs (Allocated / Total)	GPUs (Allocated / Total)	Mem (Allocated / Total)	Disk (Allocated / Total)	Registered	Re-Registered
...a99e-8aaa357bf0a1-S0	nuc02	4 / 4	0 / 0	1.4 GB / 6.8 GB	0 B / 33.7 GB	35 minutes ago	

2. Apache Spark
- Installation Procedure

1. Install on NUC 1
2. Test on NUC 1

2. Apache Spark
- Install

2. On NUC 1, Download and unarchive Spark, and configure spark-env.sh and spark-default.conf file.

```
$  cd ~
$  wget http://apache.mirror.cdnetworks.com/spark/spark-2.2.1/spark-
   2.2.1-bin-hadoop2.7.tgz
$  tar xzf spark-2.2.1-bin-hadoop2.7.tgz
$  cd spark-2.2.1-bin-hadoop2.7/conf
$  cp spark-env.sh.template spark-env.sh
$  cp spark-defaults.conf.template spark-defaults.conf
```

2.1. For spark-env.sh, append the following with vi command :

Edit:

```
•  export SPARK_LOCAL_IP="<NUC 1 IP Address>"
•  export MESOS_NATIVE_JAVA_LIBRARY="/usr/local/lib/libmesos.so"
•  export
   SPARK_EXECUTOR_URI="http://apache.mirror.cdnetworks.com/spark/sp
   ark-2.2.1/spark-2.2.1-bin-hadoop2.7.tgz"
```

2.2. For spark-defaults.conf, append the following with vi command :

Edit:

```
•  spark.master  mesos://<NUC 1 IP Address>:5050
```

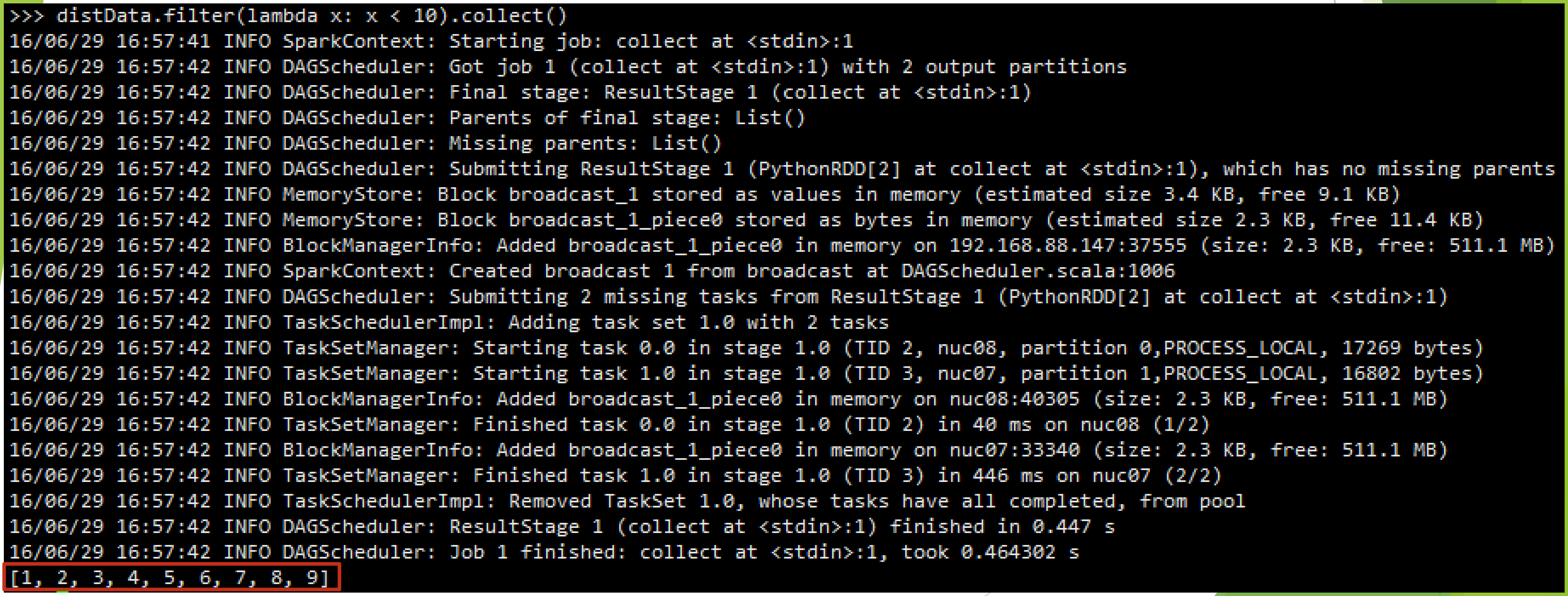
2. Apache Spark
- Test on NUC 1

# Start PySpark

```
$  cd ..
$  bin/pyspark
```

# See if PySpark is running well

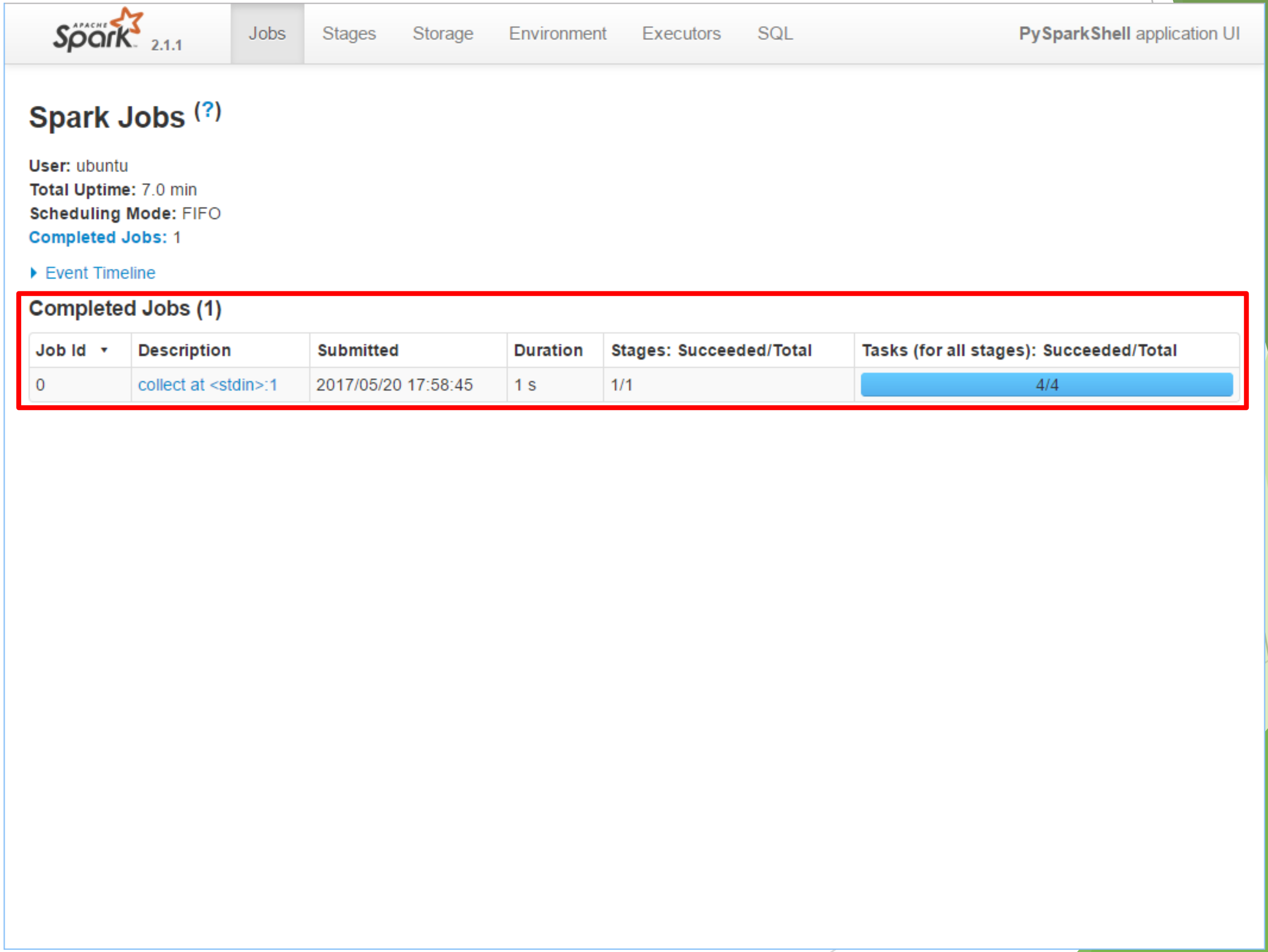
```
>  data = range(1, 10001)
>  distData = sc.parallelize(data)
>  distData.filter(lambda x: x < 10).collect()
```



```
>>> 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]
```

2. Apache Spark
- Test on NUC 1

PySpark Web UI (http://<NUC 1 Address>:4040) showing a job



Spark Jobs (?)

User: ubuntu
Total Uptime: 7.0 min
Scheduling Mode: FIFO
Completed Jobs: 1

Event Timeline

Completed Jobs (1)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	collect at <stdin>:1	2017/05/20 17:58:45	1 s	1/1	4/4

3. Apache Zeppelin on Mesos
- Install on NUC 1

1. Turn off if PySpark is still running

```
>  quit()
```

2. Install Apache Zeppelin and its prequisitories

```
$  cd ~
$  wget http://apache.mirror.cdnetworks.com/zeppelin/zeppelin-
   0.7.3/zeppelin-0.7.3-bin-all.tgz
$  tar xzf zeppelin-0.7.3-bin-all.tgz
$  cd zeppelin-0.7.3-bin-all
$  cp conf/zeppelin-env.sh.template conf/zeppelin-env.sh
```

3. Update “zeppelin-env.sh” file to configure Apache Zeppelin

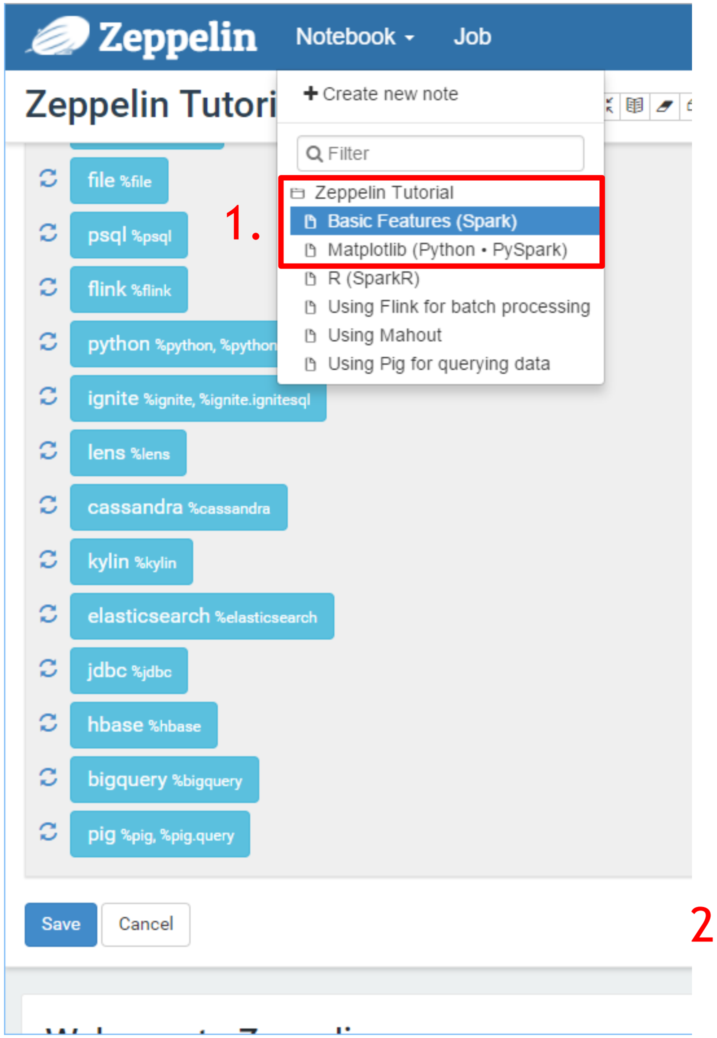
```
$  vi conf/zeppelin-env.sh

•  export MESOS_NATIVE_JAVA_LIBRARY="/usr/local/lib/libmesos.so"
•  export MASTER="mesos://<NUC 01 IP Address>:5050"
•  export SPARK_HOME="/home/<Your Linux ID>/spark-2.2.1-bin-
   hadoop2.7"
```

4. Start Apache Zeppelin daemon

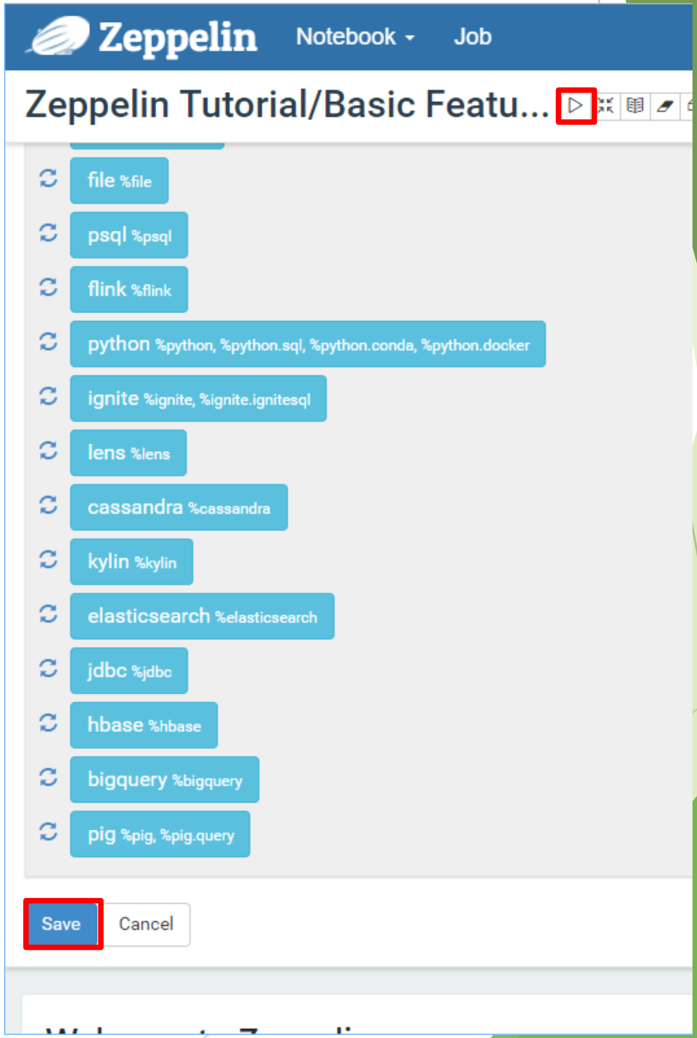
```
$  bin/zeppelin-daemon.sh start
```

3. Apache Zeppelin
- Run Example

Open its Web UI (http://<NUC 1 Address>:8080) and Run tutorial, Press ‘Run’ button to test.



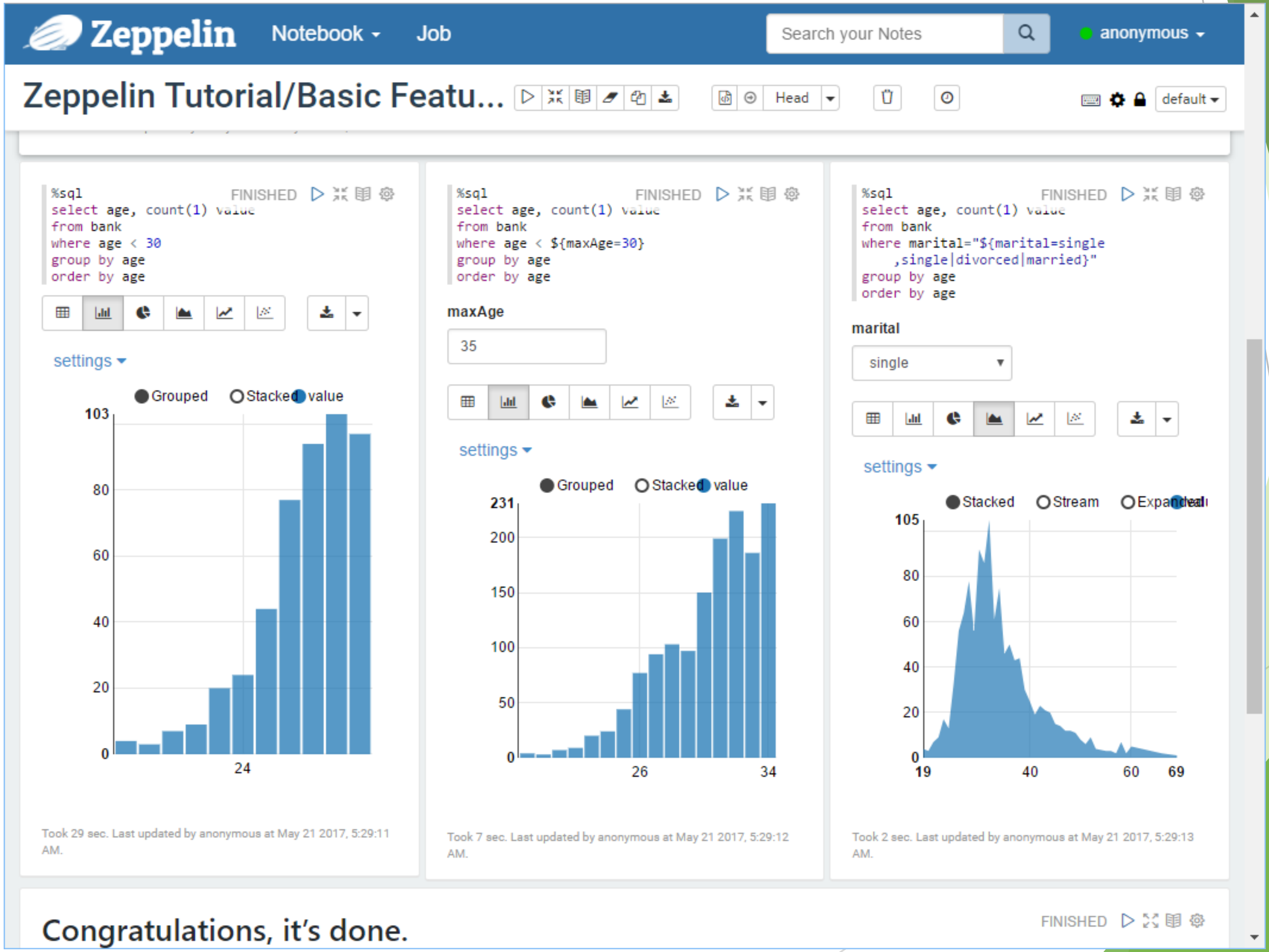
1.



2.

3. Apache Zeppelin
- Run Example

The tutorial will run like this, if all is successful.



Zeppelin Tutorial/Basic Featu...

```
%sql
select age, count(1) value
from bank
where age < 30
group by age
order by age
```

FINISHED

Took 29 sec. Last updated by anonymous at May 21 2017, 5:29:11 AM.

```
%sql
select age, count(1) value
from bank
where age < ${maxAge=30}
group by age
order by age
```

FINISHED

maxAge: 35

Took 7 sec. Last updated by anonymous at May 21 2017, 5:29:12 AM.

```
%sql
select age, count(1) value
from bank
where marital="${marital=single
    ,single|divorced|married}"
group by age
order by age
```

FINISHED

marital: single

Took 2 sec. Last updated by anonymous at May 21 2017, 5:29:13 AM.

Congratulations, it's done.

FINISHED

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.

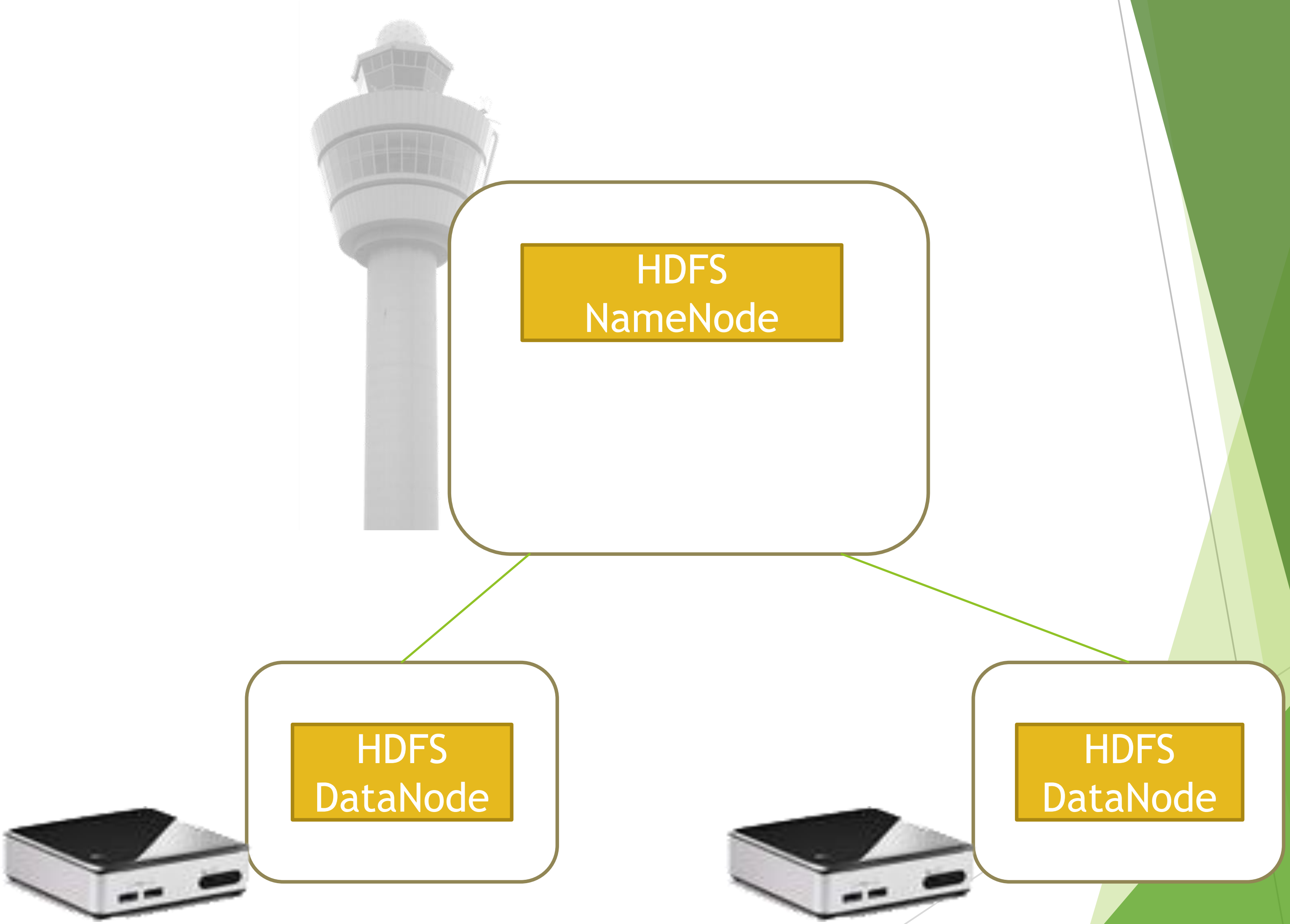
```
$ rm conf/zeppelin-env.sh
```

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



HDFS NameNode

HDFS DataNode

HDFS DataNode

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

04-1. HDFS
- Configure accounts and install SSH

Do this for **all NUCs**.

1. Install SSH package and start SSH daemon

```
$ sudo apt –y install ssh && sudo systemctl start ssh
```

2. Set root password

```
$ sudo passwd
```

3. Create Hadoop account and exit from root account

```
$ sudo -s
$ adduser hadoop
$ adduser hadoop sudo
$ exit
```

We will use only hadoop account for HDFS chapter.

4-1. HDFS
- Configure accounts and SSH settings

From **NUC 1** :

1. Log in to user ‘hadoop’.

```
$ su hadoop
```

2. Generate key (just press enter x 3) in **NUC 1**

```
$ 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 **NUC 1** to NUC 2

```
$ ssh hadoop@<NUC 2 IP Address> mkdir –p \~/.ssh
$ scp authorized_keys hadoop@<NUC 2 IP Address>:~/.ssh/
```

5. Login via SSH with hadoop account to check if you can login to NUC 2 without password and exit.

```
$ ssh hadoop@<NUC 2 IP Address>
$ exit
```

4-1. HDFS
- Download and Unzip Hadoop

1. Download and Unzip in **all NUCs**.

```
$ cd ~
$ wget
  http://apache.mirror.cdnetworks.com/hadoop/common/hadoop-
  2.8.0/hadoop-2.8.0.tar.gz
$ tar -xvzf hadoop-2.8.0.tar.gz
$ sudo mv hadoop-2.8.0 /usr/local/hadoop
```

2. From **NUC 1**, 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

For **all NUCs**, Open these files and update to the followings :

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://nuc01:9000/</value>
    </property>
</configuration>
...
```

The value must specify a hostname, not IP address.

4-1. HDFS
- Configuration

3. “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>
...
```

4. “slaves” file: Add IP address of all NUCs

Edit:

```
(Remove localhost)
<NUC 1 IP Address>
<NUC 2 IP Address>
```

4-1. HDFS
- Configuration

5. Deploy configuration files from **NUC 1** to NUC 2.

```
$  cd ..
$  scp -r hadoop hadoop@<NUC 2 IP
   Address>:/usr/local/hadoop/etc/
```

6. In **all NUCs**, make DataNode directory.

```
$ mkdir /usr/local/hadoop/datanode
```

7. In **all NUCs**, edit /etc/environment file.

```
$  sudo 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”
```

8. And append the hadoop path with the following command :

```
export PATH=$PATH:/usr/local/hadoop/bin
```

4-1. HDFS

- Start and Test

1. From **NUC 1**, login to ‘hadoop’. (Pass if already using it)

```
 $  su hadoop
```

2. Format NameNode.

```
 $  hdfs namenode -format
```

3. Start HDFS.

```
 $  /usr/local/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-2.8.0.tar.gz /user/
 $  hadoop fs -ls hdfs://<NUC 1 IP Address>:9000/user/
```

# Try the last command on both NUCs.

You can also see on the web:
http://<NUC 1 IP Address>:50070

4-2. Apache Spark with HDFS
- Configuration

Put the Apache Spark file into the HDFS and switch back to your ID.

```
$  cd /home/<Your Linux ID>
$  hadoop fs -put spark-2.1.1-bin-hadoop2.7.tgz /user/
$  exit
```

Stop Previous Apache Zeppelin if it's still running.

```
$  ~/zeppelin-0.7.1-bin-all/bin/zeppelin-daemon.sh stop
```

Change the following variable from the previous Apache Spark config :

```
$  cd ~/spark-2.1.1-bin-hadoop2.7
$  vi conf/spark-env.sh
```

Edit:

```
export SPARK_EXECUTOR_URI="hdfs://<NUC 1 IP
Address>:9000/user/spark-2.1.1-bin-hadoop2.7.tgz"
```

Test Spark

```
$  bin/pyspark

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

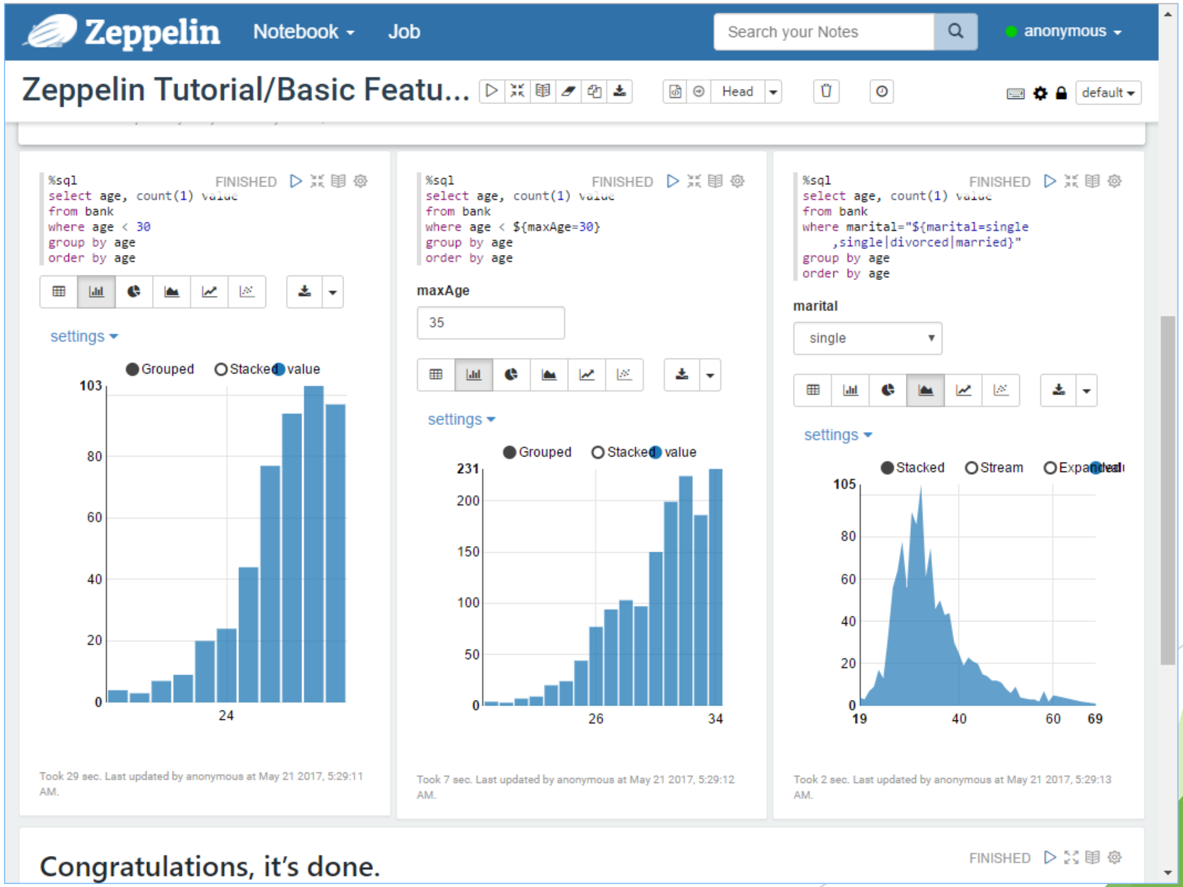
Go to Mesos web UI and see Spark framework running.

4-2. Apache Spark with HDFS
- Run Example from Apache Zeppelin

Close PySpark with quit() and in Zeppelin directory,

```
• bin/zeppelin-daemon.sh start
```

Run Zeppelin tutorial to test changed configurations.



Zeppelin Tutorial/Basic Featu...

```
%sql
select age, count(1) value
from bank
where age < 30
group by age
order by age
```

Took 29 sec. Last updated by anonymous at May 21 2017, 5:29:11 AM.

```
%sql
select age, count(1) value
from bank
where age < ${maxAge=30}
group by age
order by age
```

maxAge: 35

Took 7 sec. Last updated by anonymous at May 21 2017, 5:29:12 AM.

```
%sql
select age, count(1) value
from bank
where marital="${marital=single
    ,single|divorced|married}"
group by age
order by age
```

marital: single

Took 2 sec. Last updated by anonymous at May 21 2017, 5:29:13 AM.

Congratulations, it's done.

Thank You for
Your Attention
Any Questions?