Create a EC2-Cluster

In this example, we will create a cluster with 3 Amazon EC2 instances.

Assign each of instances a name: Master, Slave1, Slave2.

Login all three instances:

```
ssh -i key.pem ec2-user@Instance-public-ip-address
```

Modify their hosts file `/etc/hosts`:

```
master 172.31.14.138
slave1 172.31.12.252
slave2 172.31.10.80
```

(This is an example, you need replace the IP addresses with your instances' private IP addresses)

Copy your `key.pem` file from local to Master:

```
scp –i key.pem key.pem ec2-user@Master-public-ip-address
```

SSH to Master, go to the `~/.ssh/` directory and generate a keyless RSA keypair:

```
ssh-keygen –t rsa –P ‘'
```

Accept the default location and filename.

Copy the contents of `id_rsa.pub` and paste at the end of `authorized_keys` three times.

For the second two pastes, alter the IP addresses to match that of the slaves:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQC64oVfh9bbX+CBmQfUV8tzbHbzmi
4+CTHcpcicsURKl6joT9ZmZXk7SjzB9nVTMGzI8vvJqtIiwGXIXQLCeSL1
q0JBwZBunZnq0IG99BX0RlD1KWl8yWJLufSFj7IdfTApqa6Y9VZfBwAIpo
6sW21p8mnQZFb97liBFRd+Si67KoQNs6OXDqsaJk9tK2tMVayJbUhvbUfe
u21+YUBU/qK1t21Y3kfuMFi5Jr6Ozq41VQ8S2VGkVc+VLr00jP0cz8YxMj
n2hGrr1tPquA4FGLZbd6y/QsXQTe6ENhS/q1qZRxDrKYeSMABasbRTCfMI
J1muVRuUBeXpvJ+fbc12fa5B HadoopKeyPair
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAt5grkm1DDcbdk1X0Pz1TWVOsvqbAIo
sUfjQXwpRkAO/YZL3GG8+sWJ9v4OH3N8UWmUz77Q+T/Ql2z+IcHTFJq68R
HzbD7z/4dJUD2STE4RzfSqu/QD+zyZXrQ8Orty7r7APlKB2rqeh6zaVX1m
aqefERUbvkKnybHUUcZgZGsSgL2I89gYmh7SrNMLSbjw7rfhbn1vSk3bpD
HxyqjIF4ow2sifWB/BD5ScWgs0JnJguSzKKyy65AvwaVE9jv9bwVsI3otP
q/b2fCpR7J+RVImqUj9BuVd+nmzPxmFKHc7Zbt/Mnx4yP9t0Ke+V5PHxUf
m1SiSgQC4/8z4ShoxzcNnw== ec2-user@ip-X-X-X-138
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAt5grkm1DDcbdk1X0Pz1TWVOsvqbAIo
sUfjQXwpRkAO/YZL3GG8+sWJ9v4OH3N8UWmUz77Q+T/Ql2z+IcHTFJq68R
HzbD7z/4dJUD2STE4RzfSqu/QD+zyZXrQ8Orty7r7APlKB2rqeh6zaVX1m
aqefERUbvkKnybHUUcZgZGsSgL2I89gYmh7SrNMLSbjw7rfhbn1vSk3bpD
HxyqjIF4ow2sifWB/BD5ScWgs0JnJguSzKKyy65AvwaVE9jv9bwVsI3otP
q/b2fCpR7J+RVImqUj9BuVd+nmzPxmFKHc7Zbt/Mnx4yP9t0Ke+V5PHxUf
m1SiSgQC4/8z4ShoxzcNnw== ec2-user@ip-X-X-X-252
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAQEAt5grkm1DDcbdk1X0Pz1TWVOsvqbAIo
sUfjQXwpRkAO/YZL3GG8+sWJ9v4OH3N8UWmUz77Q+T/Ql2z+IcHTFJq68R
HzbD7z/4dJUD2STE4RzfSqu/QD+zyZXrQ8Orty7r7APlKB2rqeh6zaVX1m
aqefERUbvkKnybHUUcZgZGsSgL2I89gYmh7SrNMLSbjw7rfhbn1vSk3bpD
HxyqjIF4ow2sifWB/BD5ScWgs0JnJguSzKKyy65AvwaVE9jv9bwVsI3otP
q/b2fCpR7J+RVImqUj9BuVd+nmzPxmFKHc7Zbt/Mnx4yP9t0Ke+V5PHxUf
m1SiSgQC4/8z4ShoxzcNnw== ec2-user@ip-X-X-X-80
```

Copy `authorized_keys` and `id_rsa` keypair to the slaves:

```
scp –i key.pem authorized_keys ec2-user@slave1
scp –i key.pem authorized_keys ec2-user@slave2
scp id_rsa* ec2-user@slave1
scp id rsa* ec2-user@slave2
```

Then you can directly SSH to slaves without key:

```
ssh slave1
ssh slave2
ssh master
```

Java Environment

Download jdk1.8 from Oracle on each of the instances, unzip it and add `JAVA_HOME` to the `/etc/profile` file:

```
export JAVA_HOME=/home/ec2-user/jdk1.8.0_144
export PATH=$JAVA_HOME/bin:$PATH
```

After modifying, applicate the new profile:

```
Source /etc/profile
```

Maven Environment

Install maven on master. Samza needs Maven to package projects.

```
sudo wget http://repos.fedorapeople.org/repos/dchen/apache-maven/epel-
apachemaven.
repo -O /etc/yum.repos.d/epel-apache-maven.repo
sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo
sudo yum install -y apache-maven
mvn --version
```

YARN Deployment

Download Hadoop and unzip:

```
wget http://www-us.apache.org/dist/hadoop/common/hadoop-
2.8.1/hadoop-2.8.1.tar.gz
tar –xvf hadoop-2.8.1.tar.gz
cd hadoop-2.8.1
```

Add `HADOOP_HOME` to `/etc/profile` in all instances

```
export HADOOP_YARN_HOME=/home/ec2-user/hadoop-2.8.1
export HADOOP_CONF_DIR=$HADOOP_YARN_HOME/conf
```

```
Source /etc/profile
```

Modify following configuration files in hadoop directory:

Add these to the begining of `etc/hadoop/hadoop-env.sh`

```
export JAVA_HOME=/home/ec2-user/jdk1.8.0_144
export HADOOP_PREFX=/home/ec2-user/hadoop-2.8.1
```

Add these to the begining of `etc/hadoop/yarn-env.sh`

```
export JAVA_HOME=/home/ec2-user/jdk1.8.0_144
```

Add these to `etc/hadoop/core-site.xml`

```
<configuration>
        <property>
                <name>fs.defaultFS</name>
                <value>hdfs://master:9000</value>
        </property>
        <property>
                <name>hadoop.tmp.dir</name>
                <value>/home/ec2-user/hadoop-2.8.1/tmp</value>
        </property>
</configuration>
```

Add these to `etc/hadoop/hdfs-site.xml`

```
<configuration>
        <property>
                <name>dfs.permissions.enabled</name>
                <value>false</value>
        </property>
        <property>
                <name>dfs.http.address</name>
                <value>master:50070</value>
        </property>
        <property>
                <name>dfs.namenode.secondary.http-address</name>
                <value>master:50090</value>
        </property>
        <property>
                <name>dfs.replication</name>
                <value>3</value>
        </property>
</configuration>
```

Add these to `etc/hadoop/mapred-site.xml.template`

```
<configuration>
        <property>
                <name>mapred.job.tracker</name>
                <value>master:9001</value>
        </property>
        <property>
                <name>mapred.map.tasks</name>
                <value>20</value>
        </property>
        <property>
                <name>mapred.reduce.tasks</name>
                <value>4</value>
        </property>
        <property>
                <name>mapreduce.framework.name</name>
                <value>yarn</value>
        </property>
        <property>
                <name>mapreduce.jobhistory.address</name>
                <value>master:10020</value>
        </property>
        <property>
                <name>mapreduce.jobhistory.webapp.address</name>
                <value>master:19888</value>
        </property>
</configuration>
```

Add these to `etc/hadoop/yarn-site.xml`

```
<configuration>
        <property>
                <name>yarn.resourcemanager.address</name>
                <value>master:8032</value>
        </property>
        <property>
                <name>yarn.resourcemanager.scheduler.address</name>
                <value>master:8030</value>
        </property>
        <property>
                <name>yarn.resourcemanager.webapp.address</name>
                <value>master:8088</value>
        </property>
        <property>
                <name>yarn.resourcemanager.resource-
tracker.address</name>
                <value>master:8031</value>
        </property>
        <property>
                <name>yarn.resourcemanager.admin.address</name>
                <value>master:8033</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.nodemanager.resource.memory-mb</name>
                <value>8192</value>
        </property>
        <property>
                <name>yarn.scheduler.maximum-allocation-mb</name>
                <value>4096</value>
        </property>
        <property>
                <name>yarn.nodemanager.vmem-pmem-ratio</name>
                <value>5</value>
        </property>
</configuration>
```

Create a `conf/` directory under the hadoop's home directory and copy all files above into it

```
mkdir conf
cp hadoop/etc/core-site.xml conf/core-site.xml
cp hadoop/etc/hdfs-site.xml conf/hdfs-site.xml
cp hadoop/etc/mapred-site.xml.template conf/mapred-site.xml
cp hadoop/etc/yarn-site.xml conf/yarn-site.xml
cp hadoop/etc/hadoop-env.sh conf/hadoop-env.sh
cp hadoop/etc/yarn-env.sh conf/yarn-env.sh
```

Download capacity-scheduler's configuration file:

```
curl http://svn.apache.org/viewvc/hadoop/common/trunk/hadoop-yarn-
project/hadoop-yarn/hadoop-yarn-server/hadoop-yarn-server-
tests/src/test/resources/capacityscheduler.xml?view=co >
conf/capacity-scheduler.xml
```

Now copy the whole Hadoop directory to slaves

```
scp –r hadoop-2.8.1 slave1:~/
scp –r hadoop-2.8.1 slave2:~/
```

Create and add these in the slaves list file `hadoop-2.8.1/conf/slaves`:

For master node:

```
slave1
slave2
```

For slave nodes: (it's an empty file)

```

```

Format the HDFS in master node:

```
hadoop-2.8.1/bin/hdfs namenode -format
```

Start Hadoop

```
hadoop-2.8.1/sbin/start-all.sh
```

Visit the Hadoop UI (Master-public-IP-address:8088) to see whether YARN is working. Check the nodes status.

You can test the Hadoop with Tom's Hello-world APP.

(https://github.com/fuzhengjia/Simple_resa_YARN_App, there is a documentation)

Run the project:

```
git clone https://github.com/fuzhengjia/Simple_resa_YARN_App.git
mvn clean package
mv target/tom-simple-yarn-app.jar YARNAPP.jar
hadoop fs -rm -r -f /apps
hadoop fs -mkdir -p /apps
hadoop fs -copyFromLocal YARNAPP.jar /apps/YARNAPP.jar
hadoop jar YARNAPP.jar com.resa.yarn.Client
```

Check the application status on the website. If it is **finished**, then you can check the `hadoop-2.8.1/logs/userlog/applicationXXXX_XXX/containerXXXX/stdout` file to see the result. It should be a 'Hello world' in it.

Zookeeper Deployment

Download and unzip zookeeper:

```
wget www-us.apache.org/dist/zookeeper/zookeeper-3.4.10/zookeeper-3.4.10.tar.gz
tar –xvf zookeeper-3.4.10.tar.gz
```

Rename the `zoo_sample.cfg` file:

```
cd zookeeper-3.4.10
mv conf/zoo_sample.cfg conf/zoo.cfg
```

Start zookeeper on Master:

```
./bin/zkServer.sh start
```

Kafka Deployment

Download and unzip Kafka:

```
wget www-us.apache.org/dist/kafka/0.11.0.1/kafka_2.11-0.11.0.1.tgz
tar –xvf kafka_2.11-0.11.0.1.tgz
cd kafka 2.11-0.11.0.1
```

Start Kafka on Master:

```
bin/kafka-server-start.sh config/server.properties
```

Adding Virtual Memory to EC2 instances

By default, there are no disks for instances. So there are no virtual memory for YARN to allocate.

Create EBS-volumes for each of instances and attach them.

Login into instances and attach the new drive by shell instructions:

```
mkswap –f /dev/xvdf
swapon /dev/xvdf
free -m
```

See if you successfully add the virtual memory

Samza Deployment

Download and deploy samza:

```
git clone http://git-wip-us.apache.org/repos/asf/samza.git
cd samza
./gradlew clean publishToMavenLocal
```

Now you have samza in your local maven repository.

Hello-Samza example

Using hello-samza's samza application to test whether the cluster's configuration is right.

Download Hello-Samza project:

```
git clone https://git.apache.org/samza-hello-samza.git hello-samza
cd hello-samza
```

Modify the exmaple's configuration file:

```
vim src/main/config/wikipedia-feed.properties
```

Modify these fields in it:

```
yarn.package.path=hdfs://master:9000/path/for/tgz/hello-samza-
0.13.0-dist.tar.gz

systems.kafka.consumer.zookeeper.connect=master:2181
systems.kafka.producer.bootstrap.servers=master:9092
```

Compile hello-samza project:

```
mvn clean package
```

The generated project package file will be here in the `target/` directory.

Put the generated package file into HDFS so that the whole cluster have access to it:

```
~/hadoop-2.8.1/bin/hadoop fs –rm –r –f /path/for/tgz
~/hadoop-2.8.1/bin/hadoop fs –mkdir –p /path/for/tgz/
~/hadoop-2.8.1/bin/hadoop fs –copyFromLocal target/hello-samza-
0.13.0-dist.tar.gz /path/for/tgz/hello-samza-0.13.0-dist.tar.gz
```

Unzip the project package to get the Samza job submitter

```
mkdir -p deploy/samza
tar –xvf target/hello-samza-0.13.0-dist.tar.gz –C deploy/samza
```

Submit the wikipedia-feed job by using the submitter

```
deploy/samza/bin/run-job.sh --config-
factory=org.apache.samza.config.factories.PropertiesConfigFactory --
config-path=file://$PWD/deploy/samza/config/wikipedia-feed.properties
```

Check the application's status on YARN's web UI. When the application status is **running,** you can check its result by running a kafka consumer:

```
~/kafka_2.11-0.11.0.1/bin/kafka-console-consumer.sh --zookeeper
master:2181 --topic wikipedia-raw
```

If everything goes fine, tuples of wikipedia's information will keeps coming.