Part 1: Launch AWS instances

Step 1:
Login to aws console

Step 2:
Choose ubuntu machines

Step 3:
Make open security

Step 4:
Download the private key file and store it at a safe place as this is the only chance you would be able to download it. Amazon doesn't store it anywhere.
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html#retrieving-the-public-key-windows

Step 5:
Launch all nodes and modify the names

Step 6:
Go to the aws console and make a note of public DNS for each node
namenode:
datanode1:
datanode2:
datanode3:

Part 2: Connect to AWS instances through Putty

Step 7:
Generate private key using Puttygen tool which can be understood by Putty
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/putty.html

Step 8:
Open putty and start a connection with each node:
Use generated new ppk file in SSH->auth
Ip address is public DNS

Part 3: Install Java

Step 9: All instances
Update repository
sudo apt-get update

Step 10: All instances
Install Java
sudo apt-get -y install openjdk-8-jdk-headless

Part 4: Configure SSH

Step 11: namenode
ssh-keygen

Step 12: namenode
Copy the content of file */home/ubuntu/.ssh/id_rsa.pub* on clipboard

Step 13: all datanodes
Paste that content in file *~/.ssh/authorized_keys*

Step 14: namenode
Configure node details in *~/.ssh/config* file.

```
Host namenode
  HostName <DNS of namenode>
  User ubuntu
  IdentityFile ~/.ssh/id_rsa

Host datanode1
  HostName <DNS of datanode1>
  User ubuntu
  IdentityFile ~/.ssh/id_rsa

Host datanode2
  HostName <DNS of datanode2>
  User ubuntu
  IdentityFile ~/.ssh/id_rsa

Host datanode3
  HostName <DNS of datanode3>
  User ubuntu
  IdentityFile ~/.ssh/id_rsa
```

Step 15: namenode
check SSH connectivity
ssh namenode
ssh datanode1
ssh datanode2
ssh datanode3

Part 5: Download Hadoop

Step 11: All nodes
sudo wget http://archive.apache.org/dist/hadoop/common/hadoop-2.7.3/hadoop-2.7.3.tar.gz

Step 12: All nodes
Extract files
tar xvzf hadoop-2.7.3.tar.gz
mv *hadoop-2.7.3 hadoop*

Part 6: Modify .bashrc file

Step 13: All nodes
Add following variables in *~/.bashrc* file

#====== HADOOP VAR =======#
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_INSTALL=/home/ubuntu/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib/native"

Step 14: All nodes
Apply these changes
source ~/.bashrc

Part 7: Create HDFS storage dir

Step 15: All nodes
sudo mkdir -p /usr/local/hadoop/hdfs/data
sudo chown -R ubuntu:ubuntu /usr/local/hadoop/hdfs/data

Part 8: Modify configuration files

Step 16: All nodes
Go to the configuration directory
cd ~/hadoop/etc/hadoop

Step 17: All nodes
Set JAVA_HOME in hadoop-env.sh file
JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

Step 18: All nodes
Modify core-site.xml

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value><DNS of namenode>:9000</value>
  </property>
</configuration>
```

Step 19: Namenode
Modify hdfs-site.xml

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>3</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:///usr/local/hadoop/hdfs/data</value>
  </property>
</configuration>
```

Step 20: datanodes
Modify hdfs-site.xml

```
<property>
    <name>dfs.replication</name>
    <value>3</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:///usr/local/hadoop/hdfs/data</value>
  </property>
```

Step 21: Namenode
Modify mapred-site.xml
First you need to copy template file to create actual

cp mapred-site.xml.template mapred-site.xml

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

Note: No need of mentioning job tracker address as its no longer used in MR2
https://www.cloudera.com/documentation/enterprise/5-4-x/topics/cdh_ig_mapreduce_to_yarn_migrate.html

Step 22: Namenode
yarn-site.xml

```
<configuration>
  <!-- Site specific YARN configuration properties -->
  <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.resourcemanager.hostname</name>
    <value><DNS of namenode></value>
  </property>
</configuration>
```

Part 9: set up masters and slaves

Step 23: Namenode

Modify files as following:

1. masters
   <DNS of namenode>
2. slaves
   <DNS of datanode1>
   <DNS of datanode2>
   <DNS of datanode3>

Part 10: Format namenode

Step 24: Namenode
hdfs namenode -format

Part 11: Start services

Step 25: Namenode
Start HDFS service
./start-dfs.sh

Step 26: Namenode
Start YARN service
./start-yarn.sh

==========

Notes

References:
https://www.novixys.com/blog/setup-apache-hadoop-cluster-aws-ec2/

https://letsdobigdata.wordpress.com/2014/01/13/setting-up-hadoop-multi-node-cluster-on-amazon-ec2-part-1/

Check public IPv4 on console:
`curl v4.ident.me`

Block location for a file in HDFS
`hdfs fsck /myfile.txt -files -blocks -locations`

Minimal Replicated Block:
https://issues.apache.org/jira/browse/HDFS-8720

Install Spark:
https://data-flair.training/blogs/install-apache-spark-multi-node-cluster/