University of East London School of Architecture, Computing and Engineering (ACE)
Author: Dr Amin Karami (a.karami@uel.ac.uk) 19-20

Tutorial 1: Deploying Hadoop on Single Machine

CN7022 - Big Data Analytics

Dr Amin Karami (a.karami@uel.ac.uk)

LEARNING OUTCOMES: After completing this tutorial, you should:

- Have gotten a hands-on experience in installing and deploying Hadoop
- Be able to determine and install prerequisite packages and tools for deploying Hadoop
- Practice Linux commands
- Have gotten a hands-on experience in employing Hadoop commands
- Be able to read and write data from/in, and check the block sizes and the relevant information in HDFS

Hadoop Cluster modes

Standard (local) mode: No daemons, everything runs in a single JVM, has no HDFS.

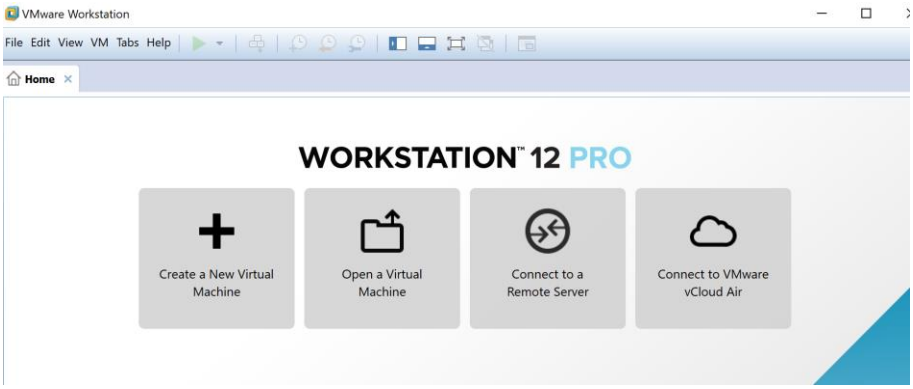
Pseudo-Distributed mode: Hadoop daemons run on the local machine and each components has its own JVM engine, plus a SSH connection between them.

Fully Distributed mode: Hadoop daemons run on a cluster of machines.

Where is Big Data files in KD labs?

(1) Login to the KD's machine:
   a. Username: **Student**
   b. Password: **acestudent**

(2) Find VMWare in the Desktop



Select "Open a Virtual Machine" → find "BigDataClass" folder at Drive D → launch the Ubuntu by selecting "Ubuntu 16.04 - Desktop.vmx" file. **Make sure to log into your own group ID.**

Phase 1: Deploying Hadoop

Step 1: Getting ready

The aim of this section is to describe how to deploy and setup Hadoop on a single node in Pseudo-Distributed Mode. Go through the following steps to complete Hadoop installation

- Find Ubuntu 16.04 (Big Data) from VMWare in UEL machines or open it in your own machine.
- Configure RAM to 8GB or 12GB, CPU to 4 Processors, 30GB HDD
- Launch Ubuntu and press **I copied!**, if a prompt shows. Then, press **Yes**.
- Usernames:
  - Tuesday Group (password: **Tuesday**)
  - Wednesday Group (password: **Wednesday**)
  - Friday Group (password: **Friday**)

Please, sign in to your own week for lab.

Step 2: Download Java 1.8

Open a new Command Prompt, then (The **sudo** gives permission to use specific system commands at the root (most powerful) level of the system. **apt-get update** downloads the

package lists from the repositories and updates them to **get** information on the newest versions of packages and their dependencies.):

- Run the following commands to delete the lock files from memory and cache:

```
$ sudo rm /var/cache/apt/archives/lock
$ sudo rm /var/lib/dpkg/lock
$ sudo rm /var/lib/apt/lists/lock
```

- Update Linux packages and install Java:

```
$ sudo apt-get update
$ sudo apt-get install openjdk-8-jdk openjdk-8-jre
```

Step 3: Turn off firewall and iptables (or add security policy)

```
$ sudo ufw disable
$ sudo iptables -F
```

Step 4: Install a secure SSH

Hadoop components cooperate with together via a secure channel. To do so, we need to set up the SSH connection as follows:

```
$ sudo apt-get install ssh
```

The SSH protocol uses public key cryptography for authenticating hosts and users.

Step 5: Generate new authentication key pairs for SSH

`ssh-keygen` is a tool for creating new authentication key pairs for SSH. Such key pairs are used for automating logins, single sign-on, and for authenticating hosts.

```
$ ssh-keygen -t rsa
```

- `-t` specifies the type of key
- If it asks you "Enter file in which to save the key (/root/.ssh/id_rsa): only press **Enter**".
- Also, for any question comes up, only press **Enter.**

- copy generated public key into an authorized key list:

```
$ cat ~/.ssh/id_rsa.pub > ~/.ssh/authorized_keys
```

`rsa` is a public key algorithm for generating key pairs. `ecdsa` and `ed25519` are also newest ones and recommend to use.

Now, let's check the `ssh` connectivity. Open a new terminal, and run `ssh localhost`. You will get a message like "`Are you sure you want to continue connecting (yes/no)?`" for the first time, so **type and enter "yes"**. If you can connect without asking password, it means that you successfully installed and configured SSH certificate. Otherwise, there is a problem and must be fixed.

Step 6: Download Hadoop files

Fetch and Install Hadoop v3.2.0 from https://archive.apache.org/dist/hadoop/common/. you can copy the web-link of the desired version and get it as follows:

- Download it:

```
$ wget https://www.dropbox.com/s/subianzkgumcpel/hadoop-3.2.0.tar.gz
```

- Unzip it:

```
$ tar xfzv hadoop-3.2.0.tar.gz
```

Step 7: Modify *bashrc* file

The `bashrc` file contains a series of configurations for the terminal session. Since Hadoop reads this file for running Hadoop engine and its components, we need to configure this file accordingly. Firstly, we need to know the physical location of java packages. They are usually in **/usr/lib/jvm**. However, if you cannot find it in this path, you need to run a shell command to find the right path by `update-alternatives --config java`

Then, we can open `bashrc` file with three different tools: `gedit`, `nano`, `vi`. Our preference would be `gedit` that is a graphical tool. If it does not work, you can apply `nano`.

Make sure to have the correct Hadoop installation path in the configuration file, based on your lab day: **tuesday_group**, **wednesday_group**, or **friday_group**.

```
$ gedit ~/.bashrc
```

- Then, add the following commands at end of the file

```
#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_INSTALL=/home/friday_group/hadoop-3.2.0/
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"
#HADOOP VARIABLES END
```

- Then, add the following permission commands, where the user can work with Hadoop components. (use `whoami` to find the current user's ID)

```
export HDFS_NAMENODE_USER="friday_group"
export HDFS_DATANODE_USER="friday_group"
export HDFS_SECONDARYNAMENODE_USER="friday_group"
export YARN_RESOURCEMANAGER_USER="friday_group"
export YARN_NODEMANAGER_USER="friday_group"

Please exchange the "friday_group" to your own lab day: tuesday_group
or wednesday_group or friday_group
```

- Finally, restart `bashrc` file by `source ~/.bashrc`

Step 8: Hadoop configuration

In this step we need to configure Hadoop. Since we copied Hadoop files into `/home directory`, it is better to move in this path for our convenient. All the configuration files are placed in `/etc/hadoop` folder.

```
$ cd /home/friday_group/hadoop-3.2.0/etc/hadoop/
```

- Modify `gedit hadoop-env.sh` by adding (**the JAVA_HOME path should be similar to JAVA_HOME in previous step**):

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

- Modify `gedit core-site.xml` by adding the following tags between "configuration" tags. (we set port 9000 (default) for listening Hadoop traffic)

```
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9000</value>
</property>
```

- Modify `gedit yarn-site.xml` by adding the following tags between "configuration" tags. (we add *shuffling* property into Hadoop)

```
<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.vmem-check-enabled</name>
<value>false</value>
</property>
```

- Modify `gedit mapred-site.xml` by adding the following tags between "configuration" tags. (we add *yarn* as Hadoop engine)

```
 <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
</property>
<property>
  <name>yarn.app.mapreduce.am.env</name>
  <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>
<property>
  <name>mapreduce.map.env</name>
  <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>
<property>
  <name>mapreduce.reduce.env</name>
  <value>HADOOP_MAPRED_HOME=${HADOOP_HOME}</value>
</property>
```

- Create a folder to hold `namenode` and `datanode` files and folders.

```
$ sudo mkdir -p /home/friday_group/hadoop_store/hdfs/
```

- Modify `gedit hdfs-site.xml` by adding the following tags between "configuration" tags.

```
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.namenode.name.dir</name>
<value>file:/home/friday_group/hadoop_store/hdfs/namenode</value>
</property>

<property>
<name>dfs.datanode.data.dir</name>
<value>file:/home/friday_group/hadoop_store/hdfs/datanode</value>
</property>
```

- Grant hadoop folder with full permission (**1:** execute; **2:** write; **4:** read for owner/group/other)

```
$ sudo chmod -R 777 /home/friday_group/hadoop-3.2.0/*

$ sudo chmod -R 777 /home/friday_group/hadoop_store/*
```

- Check the Hadoop version by `hadoop version`. If it does not show the Hadoop version correctly, it means that you missed some of configuration commands. Double check all the commands again.

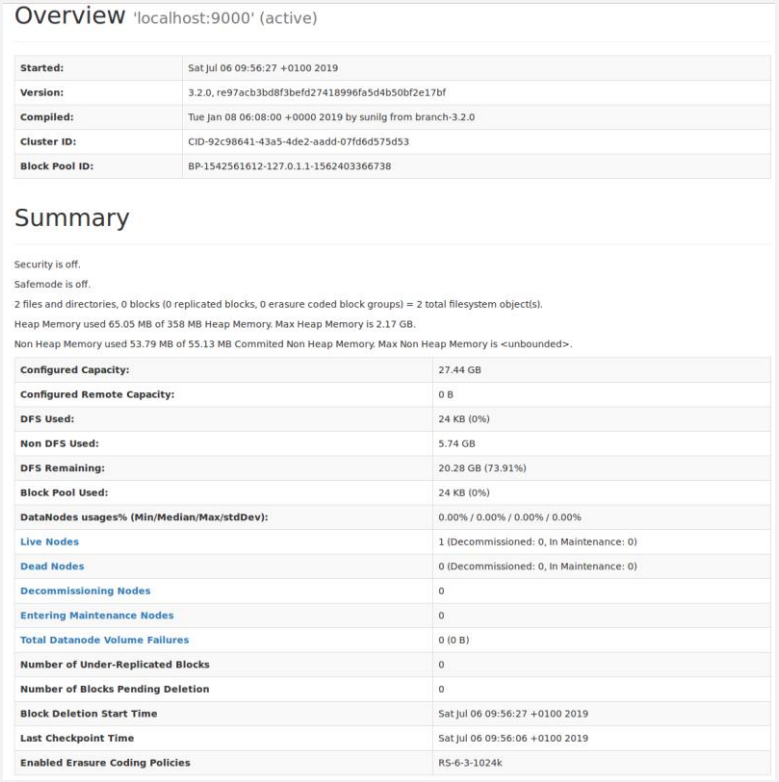
- Format the new Hadoop Filesystem by `hdfs namenode -format`
- Start Hadoop by `start-all.sh`
- Check whether all Hadoop daemons (e.g. Task Trackers, Job Trackers, DataNodes and NameNode) are already running by `jps`

```
uel@uel-Deskop-VM:/usr/local/hadoop/etc/hadoop$ jps
17440 ResourceManager
17571 NodeManager
17188 SecondaryNameNode
16824 NameNode
17916 Jps
16959 DataNode
```

Step 9: Hadoop HTTP-based monitoring console/API

A high-level overview of Hadoop components is available in:

- YARN ResourceManager: http://localhost:8088
- NameNode and DataNode: http://localhost:9870



Overview 'localhost:9000' (active)

Started:	Sat Jul 06 09:56:27 +0100 2019
Version:	3.2.0, re97acb3bd8f3befd27418996fa5d4b50bf2e17bf
Compiled:	Tue Jan 08 06:08:00 +0000 2019 by sunilg from branch-3.2.0
Cluster ID:	CID-92c98641-43a5-4de2-aadd-07fd6d575d53
Block Pool ID:	BP-1542561612-127.0.1.1-1562403366738

Summary

Security is off.
Safemode is off.
2 files and directories, 0 blocks (0 replicated blocks, 0 erasure coded block groups) = 2 total filesystem object(s).
Heap Memory used 65.05 MB of 358 MB Heap Memory. Max Heap Memory is 2.17 GB.
Non Heap Memory used 53.79 MB of 55.13 MB Commited Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	27.44 GB
Configured Remote Capacity:	0 B
DFS Used:	24 KB (0%)
Non DFS Used:	5.74 GB
DFS Remaining:	20.28 GB (73.91%)
Block Pool Used:	24 KB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	1 (Decommissioned: 0, In Maintenance: 0)
Dead Nodes	0 (Decommissioned: 0, In Maintenance: 0)
Decommissioning Nodes	0
Entering Maintenance Nodes	0
Total Datanode Volume Failures	0 (0 B)
Number of Under-Replicated Blocks	0
Number of Blocks Pending Deletion	0
Block Deletion Start Time	Sat Jul 06 09:56:27 +0100 2019
Last Checkpoint Time	Sat Jul 06 09:56:06 +0100 2019
Enabled Erasure Coding Policies	RS-6-3-1024k

Make sure to “Configured Capacity” and “Non DFS Used” are NOT 0. If they are, you will need to double check your Hadoop configurations step-by-step.

(Note 1: localhost is the Hadoop URL. If it is in another machine, easily replace localhost with the correct IP Address and/or URL)

(Note 2: If these ports (8088 and 9870) blocked by firewall/IDS, you need to add the access list policy for activation, such as by `iptables -A INPUT -p tcp --dport 50070 -j ACCEPT`)

Phase 2: Working with HDFS

Hadoop file system (fs) shell commands are used to perform various file operations such as creating directories, copying files, changing permissions, viewing the contents, changing ownership of files and so on. The syntax of fs shell command is expressed in two ways:

- `hadoop fs <args>`: FS relates to a generic file system which can point to any file sys- tems like local, HDFS etc. So this can be used when you are dealing with different file systems such as Local FS, HFTP FS, S3 FS, and others.
- `hdfs dfs <args>`: It is very specific for HDFS. It is highly recommended fscommand instead of first one.

In this section, we will learn a list of main commands that we need for Hadooping.

1. **`mkdir`:** It is used for creating directories in HDFS.
   Syntax: `$ hdfs dfs -mkdir [-p] <path>`
   `[-p]` Do not fail if the directory already exists.

   Try:

   ```
   $ hdfs dfs -mkdir /user
   ```

   ```
   $ hdfs dfs -mkdir /user/folder1
   ```

2. **`ls`:** It is used for listing directories in HDFS.
   Syntax: `$ hdfs dfs -ls [-d] [-h] [-R] <path>`
   `[-d]` Directories are listed as plain files.
   `[-h]` Formats the sizes of files in a human-readable fashion rather than a number of bytes.
   `[-R]` Recursively list the contents of directories.

   Try:

   ```
   $ hdfs dfs -ls /
   ```

   ```
   $ hdfs dfs -ls /user
   ```

   ```
   $ hdfs dfs -lsr /
   ```

3. **`put`:** Copies files from local fs to HDFS. This is similar to `-copyFromLocal` command.
   Syntax: `$ hdfs dfs -put [-f] [-p] <localsrc>...<hdfs-dst>`

   Copying fails if the file already exists, unless the `-f` flag is given. Passing `-p` preserves access and modification times, ownership and the mode. Passing `-f` overwrites the destination if it already exists.

   Try:

   ```
   $ hdfs dfs -put <put the file path> /user
   ```

4. **get:** Copies files from HDFS to local file system. This is similar to `-copyToLocal` command.
   Syntax: `$ hdfs dfs -get <hdfs-dst>...<localsrc>`

Try:

```
$ hdfs dfs -get /user/textfile <put the destination path>
```

5. **cat:** similar to `Unix cat` command, it is used for displaying contents of a file.
   Syntax: `$ hdfs dfs -cat <hdfs-file>`

Try:

```
$ hdfs dfs -cat /user/textfile
```

6. **cp:** It is used for copying files from one directory to another within HDFS.
   Syntax: `$ hdfs dfs -cp <hdfs-src-file> ...<hdfs-dst>`

Try:

```
$ hdfs dfs -mkdir /user/folder2
$ hdfs dfs -cp /user/textfile /user/folder2
$ hdfs dfs -ls -R /
```

7. **mv:** it is used for moving a file from one directory to another within HDFS.
   Syntax: `$ hdfs dfs -mv <hdfs-src-file>...<hdfs-dst>`

8. **rm:** It is used for removing a file from HDFS.
   Syntax: `$ hdfs dfs -rm [-f] [-r` or `-R] [-skipTrash] <src>`
   `[-skipTrash]` bypasses trash, if enabled, and immediately deletes `<src>`
   `[-f]` If the file does not exist, do not display a diagnostic message or modify the exit status to reflect an error.
   `[-r|R]` Recursively deletes directories.
   Note: We can use `hdfs dfs -rmdir <dst>` command to delete directories.

Try:

```
$ hdfs dfs -rm -R /user/folder2
```

Phase 3: HDFS File Block and Input Split

In this phase, we want to see how we can track the number of mappers (i.e., the split data) and the relevant information in HDFS.

1. **Firstly download a big sized data into Desktop.** `wget` is a command line utility for downloading files from the Internet. The file is UNSW-NB15 dataset for cybersecurity purposes. Its size is **560MB**.

```
$ wget https://www.dropbox.com/s/tfuwfncqrl03p0a/UNSW-NB15.csv
```

2. **Put the file in HDFS.** We make a folder, then put the file inside it. [it would take a few minutes to copy data into HDFS]

```
$ hdfs dfs -mkdir /DataAnalysis
$ hdfs dfs -put <put UNSW-NB15.csv path> /DataAnalysis
```

3. **Check the number of blocks for a file, block size, block sequence, block names and block locations.** Now, we want to see how the UNSW-NB15 dataset has been split into different blocks (maps) in HDFS.

```
$ hadoop fsck /DataAnalysis -files -blocks -locations
```

- There are 5 mappers (blocks): 560MB/128MB (block default size) = 4.375 ≈ 5
- There is only one replication per each block. We set it before in Hadoop configuration files.

```
Connecting to namenode via http://localhost:9870/fsck?ugi=uel&files=1&blocks=1&locations=1&path=%2FDataAnalysis
FSCK started by uel (auth:SIMPLE) from /127.0.0.1 for path /DataAnalysis at Sun Jul 14 13:03:06 BST 2019
/DataAnalysis <dir>
/DataAnalysis/UNSW-NB15.csv 586377597 bytes, replicated: replication=1, 5 block(s):  OK
0. BP-1676033978-127.0.1.1-1563104572746:blk_1073741825_1001 len=134217728 Live_repl=1  [DatanodeInfoWithStorage[127.0.0.1:9866,DS-6a3dc4b4-b4a0-489e-b0aa-168b789bfa9a,DISK]]
1. BP-1676033978-127.0.1.1-1563104572746:blk_1073741826_1002 len=134217728 Live_repl=1  [DatanodeInfoWithStorage[127.0.0.1:9866,DS-6a3dc4b4-b4a0-489e-b0aa-168b789bfa9a,DISK]]
2. BP-1676033978-127.0.1.1-1563104572746:blk_1073741827_1003 len=134217728 Live_repl=1  [DatanodeInfoWithStorage[127.0.0.1:9866,DS-6a3dc4b4-b4a0-489e-b0aa-168b789bfa9a,DISK]]
3. BP-1676033978-127.0.1.1-1563104572746:blk_1073741828_1004 len=134217728 Live_repl=1  [DatanodeInfoWithStorage[127.0.0.1:9866,DS-6a3dc4b4-b4a0-489e-b0aa-168b789bfa9a,DISK]]
4. BP-1676033978-127.0.1.1-1563104572746:blk_1073741829_1005 len=49506685 Live_repl=1  [DatanodeInfoWithStorage[127.0.0.1:9866,DS-6a3dc4b4-b4a0-489e-b0aa-168b789bfa9a,DISK]]

Status: HEALTHY
 Number of data-nodes:  1
 Number of racks:               1
 Total dirs:                    1
 Total symlinks:                0

Replicated Blocks:
 Total size:    586377597 B
 Total files:   1
 Total blocks (validated):      5 (avg. block size 117275519 B)
 Minimally replicated blocks:   5 (100.0 %)
 Over-replicated blocks:        0 (0.0 %)
 Under-replicated blocks:       0 (0.0 %)
 Mis-replicated blocks:         0 (0.0 %)
 Default replication factor:    1
 Average block replication:     1.0
 Missing blocks:                0
 Corrupt blocks:                0
 Missing replicas:              0 (0.0 %)

Erasure Coded Block Groups:
 Total size:    0 B
 Total files:   0
 Total block groups (validated):        0
 Minimally erasure-coded block groups:  0
 Over-erasure-coded block groups:       0
 Under-erasure-coded block groups:      0
 Unsatisfactory placement block groups: 0
 Average block group size:      0.0
 Missing block groups:          0
 Corrupt block groups:          0
 Missing internal blocks:       0
FSCK ended at Sun Jul 14 13:03:06 BST 2019 in 22 milliseconds

The filesystem under path '/DataAnalysis' is HEALTHY
```

Phase 4: Extra tips for Hadoop

1. **To get help with all commands in the Hadoop file system:** type `hadoop fs -help`
2. **working with files and folders manually:** you can use the user-friendly web interface (http://localhost:9870) rather than shell commands for manipulating files and folders into/from HDFS. The advantage of using shell command is that you can speed up execution of commands for big size data.
3. **Stop Hadoop:** `Hadoop-start` and `Hadoop-stop` shells are available in `/usr/local/hadoop/sbin`, where we installed it. You can go into this directory for running or stopping Hadoop by `stop-dfs.sh` and `stop-yarn.sh` together, or easily type `stop-all.sh` everywhere to stop all the daemons running.
4. **Start Hadoop:** if you want to start Hadoop again, type `start-all.sh` in the command line.
5. **Remove Hadoop completely:** Firstly, remove recursively all folders associated with Hadoop with `rm -rf /usr/local/hadoop`. Then, clean up all Hadoop paths from "PATH" variable in "`.bashrc`" file.