

# Introduction into Big Data analytics

## Lecture 3a – Hadoop installation on Ubuntu

**Janusz Szwabiński**

Outlook:

1. VirtualBox
2. Downloading Hadoop
3. Stand-alone installation
4. Pseudo-distributed mode

## VirtualBox ¶

- <https://www.virtualbox.org/> (<https://www.virtualbox.org/>)
- follow the installation steps in the User Manual (<https://www.virtualbox.org/manual/UserManual.html> (<https://www.virtualbox.org/manual/UserManual.html>))

### New virtual machine

- download Ubuntu from <https://www.ubuntu.com/download/desktop> (<https://www.ubuntu.com/download/desktop>) and install it within the virtual machine
- follow the installation tutorial at <https://tutorials.ubuntu.com/tutorial/tutorial-install-ubuntu-desktop> (<https://tutorials.ubuntu.com/tutorial/tutorial-install-ubuntu-desktop>) in case you are not familiar with GNU/Linux systems
- as an alternative, you may use one of the existing Virtual Ubuntu Images from <https://www.osboxes.org/ubuntu/> (<https://www.osboxes.org/ubuntu/>)
- use fixed size disks, because
  - they help to avoid hadoop's unhealthy node issues related to storage size
  - they speed up your virtual machines

### Resizing the virtual drive

- <https://technology.amis.nl/2017/01/30/ubuntu-vm-virtualbox-increase-size-disk-make-smaller-exports-distribution/> (<https://technology.amis.nl/2017/01/30/ubuntu-vm-virtualbox-increase-size-disk-make-smaller-exports-distribution/>)

## Accessing virtual machine from host OS

### Via ssh

1. Install ssh server on a guest box.
2. Go to the network settings of the virtual machine and click the *Port Forwarding button*.
3. Add a new rule:

Host port 3022, guest port 22, name ssh, protocol TCP

4. To ssh into the guest VM, write

```
ssh -p 3022 user@127.0.0.1
```

5. To copy files between host and guest, use sftp instead:

```
sftp -P 3022 user@127.0.0.1
```

### Important note

Other solutions are possible. Check <https://www.virtualbox.org/manual/ch06.html> (<https://www.virtualbox.org/manual/ch06.html>) for more details.

### Important note #2

If your host OS is a flavor of MS Windows, you may be interested in installing:

- putty, <https://www.putty.org/> (<https://www.putty.org/>)
- WInSCP, <https://winscp.net/eng/docs/lang:pl> (<https://winscp.net/eng/docs/lang:pl>)

## Via shared folders

1. Install Guest Additions on your virtual machine.
2. Creating a shared folder:
  - A. Create a folder on the Host computer that you would like to share, for example ~/share.
  - B. Boot the Guest operating system in VirtualBox.
  - C. Select Devices -> Shared Folders....
  - D. Choose the Add button. Select ~/share.
  - E. Optionally select the Make permanent option.
3. Preparing the folder:
  - Linux guest

```
sudo mount -t vboxsf -o uid=$UID,gid=$(id -g) share ~/host
```

- Windows guest

```
net use x: \\vboxsvr\share
```

### Important note

In case you get a Permission denied error while trying to access the folder on a Linux guest, add your user to the vboxsf group:

```
sudo usermod -a -G vboxsf szwabin
```

## Software installation

### Java

```
sudo apt-get install openjdk-8-jdk
```

For setting up JAVA\_HOME and PATH variables, add the following lines to the .bashrc file in your \$HOME directory:

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/
export PATH=$PATH:$JAVA_HOME/bin
```

(on Ubuntu 16.04)

Apply changes into the current running system with

```
source ~/.bashrc
```

### SSH server

```
sudo apt-get install openssh-server
```

You may want to generate RSA keys for passphraseless usage:

```
ssh-keygen -t rsa
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod 0600 ~/.ssh/authorized_keys
```

### ViM text editor (optionally)

```
sudo apt-get install vim
```

**Important note**

In case you do not know an exact package name, you can search the available packages in the Ubuntu repository with

```
apt-cache search keyword
```

## Hadoop

- go to <http://hadoop.apache.org/releases.html> (<http://hadoop.apache.org/releases.html>) and download the latest stable release
- extract the archive in a directory of your choice (e.g. /usr/local) with

```
tar xzf hadoop-your-version.tar.gz
```

## Hadoop operation modes

- **Local/Standalone** mode
  - default configuration of fresh installed Hadoop
  - Hadoop can be run as a single java process
- **Pseudo Distributed** mode
  - a distributed simulation on single machine
  - each Hadoop daemon such as hdfs, yarn, MapReduce etc. will run as a separate java process
  - useful for development and experimenting with Hadoop
- **Fully Distributed** mode
  - minimum two or more machines as a cluster

## Stand-alone Hadoop installation

- no daemons running
- everything runs in a single JVM
- suitable for running MapReduce programs during development (easy to test and debug them)

### Installation

Unpack the Hadoop distribution to a directory of your choice, for instance `/usr/local/hadoop`

### Configuration

- `hadoop-env.sh`

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

- `.bashrc`

```
# setting up Java environment
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export PATH=$PATH:$JAVA_HOME/bin
```

```
# setting up Hadoop
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
```

In order to update the variables set up in the `.bashrc` file, issue the command

```
source .bashrc
```

### Test

- Hadoop installation delivers an example MapReduce jar file

In [1]:

```
!hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.0.0.jar
```

An example program must be given as the first argument.

Valid program names are:

aggregatewordcount: An Aggregate based map/reduce program that counts the words in the input files.

aggregatewordhist: An Aggregate based map/reduce program that computes the histogram of the words in the input files.

bbp: A map/reduce program that uses Bailey-Borwein-Plouffe to compute exact digits of Pi.

dbcoun: An example job that count the pageview counts from a database.

distbbp: A map/reduce program that uses a BBP-type formula to compute exact bits of Pi.

grep: A map/reduce program that counts the matches of a regex in the input.

join: A job that effects a join over sorted, equally partitioned datasets

multifilewc: A job that counts words from several files.

pentomino: A map/reduce tile laying program to find solutions to pentomino problems.

pi: A map/reduce program that estimates Pi using a quasi-Monte Carlo method.

randomtextwriter: A map/reduce program that writes 10GB of random textual data per node.

randomwriter: A map/reduce program that writes 10GB of random data per node.

secondarysort: An example defining a secondary sort to the reducer.

sort: A map/reduce program that sorts the data written by the random writer.

sudoku: A sudoku solver.

teragen: Generate data for the terasort

terasort: Run the terasort

teravalidate: Checking results of terasort

wordcount: A map/reduce program that counts the words in the input files.

wordmean: A map/reduce program that counts the average length of the words in the input files.

wordmedian: A map/reduce program that counts the median length of the words in the input files.

wordstandarddeviation: A map/reduce program that counts the standard deviation of the length of the words in the input files.

- it provides basic functionality of MapReduce
- it can be used for computations (e.g. Pi value), word counts in a given list of files, etc.

Let us have an input directory where we will push a few text files. Our goal is to count the total number of words in those files:

- we do not need to write a MapReduce application
- word count functionality is included in the example jar

```
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.0.0.jar wordcount MyMR/ output
```

# Pseudo-distributed mode

## First attempt

- default Ubuntu installation within VirtualBox
- 10 GB virtual disc with a dynamic size

## Configuration

- .bashrc in your home directory

```
# setting up Java environment
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export PATH=$PATH:$JAVA_HOME/bin

# setting up Hadoop
export HADOOP_HOME=/home/szwabin/Tools/hadoop-3.0.0
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_INSTALL=$HADOOP_HOME
```

- core-site.xml in \$HADOOP\_HOME/etc/hadoop directory

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

- hdfs-site.xml in \$HADOOP\_HOME/etc/hadoop directory

```
<configuration>

  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>

  <property>
    <name>dfs.name.dir</name>
    <value>file:///home/szwabin/hadoopinfra/hdfs/namenode</value>
  </property>

  <property>
    <name>dfs.data.dir</name>
    <value>file:///home/szwabin/hadoopinfra/hdfs/datanode</value>
  </property>

</configuration>
```

- mapred-site.xml in \$HADOOP\_HOME/etc/hadoop directory

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

- yarn-site.xml in \$HADOOP\_HOME/etc/hadoop directory

```
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
</configuration>
```

## Formatting the HDFS

```
hdfs namenode -format
```

You should see something like

```
2018-03-08 11:46:33,556 INFO namenode.NameNode: STARTUP_MSG:
/*****
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = enterprise/127.0.1.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 3.0.0
<snip>
2018-03-08 11:46:34,160 INFO namenode.NameNode: SHUTDOWN_MSG:
/*****
SHUTDOWN_MSG: Shutting down NameNode at enterprise/127.0.1.1
*****/
```



## Starting DFS

```
start-dfs.sh
```

The result should be similar to

```
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [enterprise]
2018-03-08 11:50:47,280 WARN util.NativeCodeLoader: Unable to load native
-hadoop library for your platform... using builtin-java classes where app
licable
```

## Starting Yarn

```
start-yarn.sh
```

The result should be as follows:

```
Starting resourcemanager
Starting nodemanagers
```

## Accessing Hadoop

We can access Hadoop via browser at

```
http://localhost:9870/
```

(in older versions of Hadoop the port is 50070)

The screenshot shows a web browser window with the URL `localhost:9870/dfshealth.html#tab-overview`. The browser's address bar and tabs are visible. The main content area shows the Hadoop Overview page for 'localhost:9000' (active). The page has a green navigation bar with tabs for 'Hadoop', 'Overview', 'Datanodes', 'Datanode Volume Failures', 'Snapshot', 'Startup Progress', and 'Utilities'. Below the navigation bar, the 'Overview' section is active, displaying a table with the following information:

Started:	Thu Mar 08 12:04:00 +0100 2018
Version:	3.0.0, rc25427ceca461ee979d30edd7a4b0f50718e6533
Compiled:	Fri Dec 08 20:16:00 +0100 2017 by andrew from branch-3.0.0
Cluster ID:	CID-e048bfc9-3706-4218-b0de-c3cba865b021
Block Pool ID:	BP-933626661-127.0.1.1-1520505994072

Below the table, there is a 'Summary' section with the following text:

Security is off.  
Safemode is off.  
1 files and directories, 0 blocks = 1 total filesystem object(s).  
Heap Memory used 93.39 MB of 457.5 MB Heap Memory. Max Heap Memory is 3.45 GB.  
Non Heap Memory used 45.9 MB of 47.09 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

Below the summary text, there is another table with the following information:

Configured Capacity:	416.16 GB
DFS Used:	24 KB (0%)
Non DFS Used:	192.13 GB
DFS Remaining:	202.87 GB (48.75%)

The default port number to access all applications of the cluster is 8088:

```
http://localhost:8088/
```

The screenshot shows the Hadoop All Applications web interface. The browser address bar is localhost:8088/cluster. The page title is "All Applications". On the left, there is a navigation menu with options like "Cluster", "About Nodes", "Node Labels", "Applications", "NEW", "NEW\_SAVING", "SUBMITTED", "ACCEPTED", "RUNNING", "FINISHED", "FAILED", "KILLED", and "Scheduler". The main content area displays "Cluster Metrics" and "Cluster Nodes Metrics".

Cluster Metrics										
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	
0	0	0	0	0	0 B	8 GB	0 B	0	8	

Cluster Nodes Metrics							
Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes		
1	0	0	0	0	0		

**Scheduler Metrics**

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Ap
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

Show 20 entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Reserved CPU VCores	Reserved Memory MB	% of Queue	% of Cluster	Progress
No data available in table																	

Showing 0 to 0 of 0 entries

## Creating working directory

```
hadoop fs -mkdir /input
```

## Preparing example data

```
hadoop fs -put $HADOOP_HOME/*.txt /input
```

## Starting a MapReduce example

```
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.0.0.jar wordcount /input /output
```

- usually, the job will be submitted but not finished
- a closer look at the status of the cluster reveals, that there are
  - 0 active nodes
  - 1 unhealthy node

The screenshot shows the Hadoop All Applications web interface. The browser address bar is localhost:8088/cluster. The page title is "All Applications". On the left, there is a navigation menu with options like "Cluster", "About Nodes", "Node Labels", "Applications", "NEW", "NEW\_SAVING", "SUBMITTED", "ACCEPTED", "RUNNING", "FINISHED", "FAILED", "KILLED", and "Scheduler". The main content area displays "Cluster Metrics" and "Cluster Nodes Metrics".

Cluster Metrics										
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	
1	1	0	0	0	0 B	0 B	0 B	0	0	

Cluster Nodes Metrics							
Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes		
0	0	0	0	1	0		

**Scheduler Metrics**

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Ap
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

Show 20 entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Reserved CPU VCores	Reserved Memory MB	% of Queue	% of Cluster	Progress
application_1522149379810_0001	szwab	word count	MAPREDUCE	default	0	Tue Mar 27 13:21:00 +0200 2018	N/A	ACCEPTED	UNDEFINED	0	0	0	0	0	0,0	0,0	

Showing 1 to 1 of 1 entries

- the node is unhealthy due to the fact that there is no more usable disk space

## Second attempt

- 40 GB virtual disc (fixed size)
- same Hadoop configuration
- in this case, you usually get the following (or similar) error message while running the mapreduce job:

```
Could not find or load main class org.apache.hadoop.mapred.YarnChild
```

## Third attempt

To solve the above issue, simply update the classpath in your `.bashrc` file:

```
# setting up Java environment
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export PATH=$PATH:$JAVA_HOME/bin

# setting up Hadoop
export HADOOP_HOME=/home/szwabin/Tools/hadoop-3.0.0
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_CLASSPATH=$(hadoop classpath):$HADOOP_CLASSPATH
```

## Fourth attempt

After saving the file and updating the environment, the above error disappeared, but I got the following message instead:

```
Container [pid=...] is running beyond virtual memory limits.
```

To solve that, we should properly configure the maximum memory allocations for MapReduce (file `mapred-site.xml`):

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

  <property>
    <name>mapreduce.admin.user.env</name>
    <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
  </property>

  <property>
    <name>yarn.app.mapreduce.am.env</name>
    <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
  </property>

  <property>
    <name>mapreduce.map.memory.mb</name>
    <value>2048</value>
  </property>

  <property>
    <name>mapreduce.reduce.memory.mb</name>
    <value>4096</value>
  </property>

  <property>
    <name>mapreduce.map.java.opts</name>
    <value>-Xmx1024m</value>
  </property>

  <property>
    <name>mapreduce.reduce.java.opts</name>
    <value>-Xmx3072m</value>
  </property>

</configuration>
```

In [ ]: