Pi Hadoop Cluster

**Install Raspbian and prepare environment for Hadoop**

**Rasbian installation**

Download Raspbian Jessie Lite:

<https://www.raspberrypi.org/downloads/raspbian/>

Write to SD card (use any tool of choice) for windows I use:

<https://sourceforge.net/projects/win32diskimager/>

Plugin in SD card and fire up your PI.

For inital configuration (raspi-config)

* Expand filesystem
* Under *9 Advanced Options -> A3 Memory Split*Choose 16MB to give as much RAM as possible for Hadoop

If you wish you may also try to overlcock the PI a bit to improve performance. For this tutorial I use default.

**Configure network**

Login as pi (default password: raspberry)

Edit (as root with sudo -s):

/etc/dhcpcd.conf

At the bottom of the file add:

interface eth0

static ip\_address=192.168.29.109/24

# static ip\_address=192.168.29.110/24

# static ip\_address=192.168.29.111/24

# static ip\_address=192.168.29.112/24

# static ip\_address=192.168.29.113/24

# static ip\_address=192.168.29.114/24

# static ip\_address=192.168.29.115/24

# static ip\_address=192.168.29.116/24

# static ip\_address=192.168.29.117/24

Change domain name servers to your environment. Note that we are not required to make any changes to /etc/interfaces in Raspbian Jessie.

**Update system and install Oracle Java**

sudo apt-get update && sudo apt-get install oracle-java7-jdk

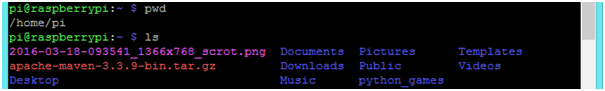
Run *update-alternatives*, ensure *jdk-8-oracle-\*\*\** is selected:

sudo update-alternatives --config java

**Install maven:**

1.  Connect to Raspberry Pi using SSH, with your valid credentials.  
2.  The Raspberry Pi‘s raspbian Jessie image comes with oracle-java8-jdk already installed and I wanted to use this with maven. However if you install maven using apt-get it will install all kinds of other JDKs and things you don’t need. The solution to this is to install maven from its binary distribution. This is quite easy as everything it needs is already install.  
3.  Download the Binary tar.gz version the mavenwebsite. Pick the latest version. At the time of writing this document, 3.3.9 is the latest version of Apache Maven available. Also in your case, the URL might be different, the Apache Maven Website might suggest a different URL mirror, closer to your geographical location.  
     **wget http://apache.**[**mirrors.pair.com**](http://mirrors.pair.com/)**/maven/maven-3/3.3.9/binaries/apache-maven-3.3.9-bin.tar.gz**

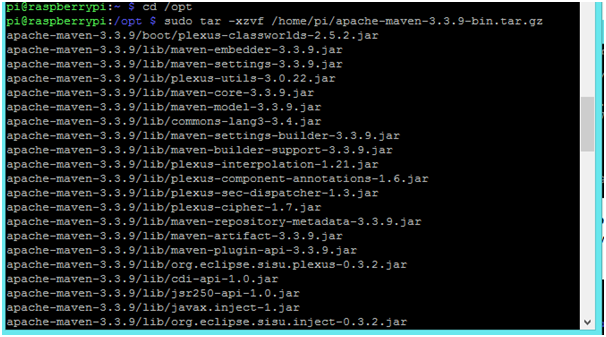
4.  Once it is downloaded, please check if you could find the **apache-maven-3.3.9-bin.tar.gz** file in the path that you downloaded with **ls** command and it is available in the path /home/pi in my case.

[](https://3.bp.blogspot.com/-zTQrg1g9BLo/V1Txw4W68OI/AAAAAAAAAPA/qv-8rC5Qrbk0pgVhrlSkuPUqWNoqpIdSwCKgB/s1600/image_2.png)

5.  Now I would like to untar(unzip) this to /opt folder. So change to /opt folder and untar using the following command.

**cd /opt**

**sudo tar -xzvf /home/pi/apache-maven-3.3.9-bin.tar**

[](https://1.bp.blogspot.com/-v_jzXKY5F1s/V1Txw_veJ4I/AAAAAAAAAPE/z3BhjWWkkdEdZFm3eToIoVTQQNo-Z8lvACKgB/s1600/image_3.png)

6.   Now it’s time to tell our shell where to find maven. We will do this in the system profile settings so it is available to all users.

**sudo nano /etc/profile.d/maven.sh**

      and paste the following and save (Ctrl+X) and Enter Y to save the file.

export M2\_HOME=/opt/apache-maven-3.3.9

export PATH=$PATH:$M2\_HOME/bin

export MAVEN\_OPTS="-Xmx2048m -XX:MaxPermSize=512m"

7.   Now reboot your raspberry pi using the command:

**sudo reboot**

8.   Log back in to Raspberry Pi with the credentials and enter the following command to ensure Maven set up is complete.

**mvn -version**

[](https://3.bp.blogspot.com/-CCEW-47U3rw/V1TxxUNvFPI/AAAAAAAAAPQ/5DyZZrBbEXAsBq98w7JuWslwChjHK-hBQCKgB/s1600/image_4.png)

       This command will display the Maven Home, Java version that is installed.

8.    That's it, now Maven is successfully installed on Raspberry Pi.

**Configure Hadoop user**

sudo addgroup hadoop

sudo adduser --ingroup hadoop hduser

sudo adduser hduser sudo

Create SSH paris keys with blank password. This will enable nodes to communicate with each other in the cluster.

su hduser

mkdir ~/.ssh

ssh-keygen -t rsa -P ""

cat ~/.ssh/id\_rsa.pub > ~/.ssh/authorized\_keys

Login as hduser (answer yes when prompted to trust certificate key – otherwise Hadoop will fail to login later)

su hduser

ssh localhost

exit

**Compile Native Hadoop 2.7.2 for Raspberry PI (ARM)**

Ensure you have logged out as hduser and logged in as pi user. (for sudo command below to work properly)

## Install protobuf 2.5.0

This is required to build Hadoop.

wget https://github.com/google/protobuf/releases/download/v2.5.0/protobuf-2.5.0.tar.gz

tar xzvf protobuf-2.5.0.tar.gz

cd protobuf-2.5.0

./configure --prefix=/usr

make

make check

sudo make install

## Install snappy

### **Download Snappy**

wget <http://pkgs.fedoraproject.org/repo/pkgs/snappy/snappy-1.1.1.tar.gz/8887e3b7253b22a31f5486bca3cbc1c2/snappy-1.1.1.tar.gz>

tar xzvf snappy-1.1.1.tar.gz

cd snappy-1.1.1/

./configure

make

sudo make install

### **Check snappy**

ls -lh /usr/local/lib |grep snappy

-rw-r--r--  1 root root 229K Jun 21 15:46 libsnappy.a  
-rwxr-xr-x  1 root root  953 Jun 21 15:46 libsnappy.la  
lrwxrwxrwx  1 root root   18 Jun 21 15:46 libsnappy.so -> libsnappy.so.1.2.0  
lrwxrwxrwx  1 root root   18 Jun 21 15:46 libsnappy.so.1 -> libsnappy.so.1.2.0  
-rwxr-xr-x  1 root root 145K Jun 21 15:46 libsnappy.so.1.2.0

## Install cmake snappy libsnappy-dev & bzip2 libbz2-dev

sudo apt-get install cmake snappy libsnappy-dev bzip2 libbz2-dev

## Install zlib1g-dev & libssl-dev

sudo apt-get install zlib1g-dev libssl-dev

## Hadoop 2.7.2 download and build

wget http://apache.claz.org/hadoop/common/hadoop-2.7.2/hadoop-2.7.2-src.tar.gz

tar xzvf hadoop-2.7.2-src.tar.gz

### For Hadoop 2.7.2 to build properly we also need to apply a patch.

cd hadoop-2.7.2-src/hadoop-common-project/hadoop-common/src

wget https://issues.apache.org/jira/secure/attachment/12570212/HADOOP-9320.patch

patch < HADOOP-9320.patch

### Start build Hadoop-2.7.2

cd ~/hadoop-2.7.2-src/

mvn package -Pdist,native -DskipTests -Dtar

if “Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:2.10:test” error then try:

mvn clean install –U package -Pdist,native -DskipTests -Dtar

Give access to hduser

## Install Hadoop

Copy compiled binaries to /opt

cd hadoop-dist/target/

sudo cp -R hadoop-2.7.2 /opt/hadoop

Give access to hduser

sudo chown -R hduser.hadoop /opt/hadoop/

## Configure the environment

In ~/.bashrc, add to bottom of file:

# Set Hadoop-related environment variables

export HADOOP\_HOME=/opt/hadoop

# Set JAVA\_HOME (we will also configure JAVA\_HOME directly for Hadoop later on)

export JAVA\_HOME=/usr/lib/jvm/jdk-7-oracle-arm-vfp-hflt

# Some convenient aliases and functions for running Hadoop-related commands

unalias fs &> /dev/null

alias fs="hadoop fs"

unalias hls &> /dev/null

alias hls="fs -ls"

# If you have LZO compression enabled in your Hadoop cluster and

# compress job outputs with LZOP (not covered in this tutorial):

# Conveniently inspect an LZOP compressed file from the command

# line; run via:

#

# $ lzohead /hdfs/path/to/lzop/compressed/file.lzo

#

# Requires installed 'lzop' command.

#

lzohead () {

hadoop fs -cat $1 | lzop -dc | head -1000 | less

}

# Add Hadoop bin/ directory to PATH

#export PATH=$PATH:$HADOOP\_HOME/bin

# HADOOP VARIABLES START

export JAVA\_HOME=/usr/lib/jvm/java-7-oracle-arm-vfp-hflt

export HADOOP\_INSTALL=/opt/hadoop

export PATH=$PATH:$HADOOP\_INSTALL/sbin

export PATH=$PATH:$HADOOP\_INSTALL/bin

export HADOOP\_MAPRED\_HOME=$HADOOP\_INSTALL

export HADOOP\_COMMON\_HOME=$HADOOP\_INSTALL

export HADOOP\_HDFS\_HOME=$HADOOP\_INSTALL

export HADOOP\_YARN\_HOME=$HADOOP\_INSTALL

export HADOOP\_COMMON\_LIB\_NATIVE\_DIR=$HADOOP\_INSTALL/lib/native

export HADOOP\_OPTS="-Djava.library.path=$HADOOP\_INSTALL/lib/native"

export HADOOP\_CONF\_DIR=${HADOOP\_HOME}"/etc/hadoop"

Afterwards source the .profile file source ~/.bashrc.

Verify installation and native libraries

su hduser

cd /opt/hadoop/bin

hadoop checknative -a

16/03/24 20:20:03 INFO bzip2.Bzip2Factory: Successfully loaded & initialized native-bzip2 library system-native  
16/03/24 20:20:03 INFO zlib.ZlibFactory: Successfully loaded & initialized native-zlib library  
Native library checking:  
hadoop: true /opt/hadoop/lib/native/libhadoop.so.1.0.0  
zlib: true /lib/arm-linux-gnueabihf/libz.so.1  
snappy: true /usr/lib/libsnappy.so.1  
lz4: true revision:99  
bzip2: true /lib/arm-linux-gnueabihf/libbz2.so.1  
openssl: true /usr/lib/arm-linux-gnueabihf/libcrypto.so

Check version

hadoop version

Hadoop 2.7.2  
Subversion Unknown -r Unknown  
Compiled by root on 2016-02-21T19:05Z  
Compiled with protoc 2.5.0  
From source with checksum d0fda26633fa762bff87ec759ebe689c  
This command was run using /opt/hadoop/share/hadoop/common/hadoop-common-2.7.2.jar

## Edit and change varibales in Hadoop environment.sh (/opt/hadoop/etc/hadoop/)

Find out your java home (readlink -f /usr/bin/java | sed “s:jre/bin/java::).

export JAVA\_HOME=/usr/lib/jvm/jdk-7-oracle-arm32-vfp-hflt

Enable the use of native hadoop library and IPv4 stack:

export HADOOP\_OPTS="$HADOOP\_OPTS -Djava.library.path=$HADOOP\_INSTALL/lib/native -Djava.net.preferIPv4Stack=true"

# Configure Single Node

Edit configuration files in /opt/hadoop/etc/Hadoop

### core-site.xml

<configuration>

<property>

<name>hadoop.tmp.dir</name>

<value>/hdfs/tmp</value>

</property>

<property>

<name>fs.defaultFS</name>

<value>hdfs://localhost:54310</value>

</property>

</configuration>

hdfs-site.xml

</configuration>

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

<property>

<name>dfs.blocksize</name>

<value>5242880</value>

</property>

</configuration>

### yarn-site.xml

<configuration>

<property>

<name>yarn.resourcemanager.resource-tracker.address</name>

<value>node1:8025</value>

</property>

<property>

<name>yarn.resourcemanager.scheduler.address</name>

<value>node1:8035</value>

</property>

<property>

<name>yarn.resourcemanager.address</name>

<value>node1:8050</value>

</property>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.resource.cpu-vcores</name>

<value>4</value>

</property>

<property>

<name>yarn.nodemanager.resource.memory-mb</name>

<value>768</value>

</property>

<property>

<name>yarn.scheduler.minimum-allocation-mb</name>

<value>64</value>

</property>

<property>

<name>yarn.scheduler.maximum-allocation-mb</name>

<value>256</value>

</property>

<property>

<name>yarn.scheduler.minimum-allocation-vcores</name>

<value>1</value>

</property>

<property>

<name>yarn.scheduler.maximum-allocation-vcores</name>

<value>4</value>

</property>

<property>

<name>yarn.nodemanager.vmem-check-enabled</name>

<value>true</value>

</property>

<property>

<name>yarn.nodemanager.pmem-check-enabled</name>

<value>true</value>

</property>

<property>

<name>yarn.nodemanager.vmem-pmem-ratio</name>

<value>2.1</value>

</property>

</configuration>

### mapred-site.xml

<configuration>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

<property>

<name>mapreduce.map.memory.mb</name>

<value>256</value>

</property>

<property>

<name>mapreduce.map.java.opts</name>

<value>-Xmx204m</value>

</property>

<property>

<name>mapreduce.map.cpu.vcores</name>

<value>2</value>

</property>

<property>

<name>mapreduce.reduce.memory.mb</name>

<value>128</value>

</property>

<property>

<name>mapreduce.reduce.java.opts</name>

<value>-Xmx102m</value>

</property>

<property>

<name>mapreduce.reduce.cpu.vcores</name>

<value>2</value>

</property>

<property>

<name>yarn.app.mapreduce.am.resource.mb</name>

<value>128</value>

</property>

<property>

<name>yarn.app.mapreduce.am.command-opts</name>

<value>-Xmx102m</value>

</property>

<property>

<name>yarn.app.mapreduce.am.resource.cpu-vcores</name>

<value>1</value>

</property>

<property>

<name>mapreduce.job.maps</name>

<value>4</value>

</property>

<property>

<name>mapreduce.job.reduces</name>

<value>4</value>

</property>

</configuration>

# Format HDFS filesystem

sudo mkdir -p /hdfs/tmp

sudo chown hduser:hadoop /hdfs/tmp

sudo chmod 750 /hdfs/tmp

hadoop namenode -format

# Configure Multiple Nodes

## Clear HDFS filesystem

rm -rf /hdfs/tmp/\*

hdfs namenode -format

## Edit configuration files:

### /etc/hostname

node1/node2/node3

### /etc/hosts

192.168.0.110 node1

192.168.0.111 node2

192.168.0.112 node3

### /etc/dhcpcd.conf

interface eth0

**static ip\_address=192.168.0.110/24**

**# static ip\_address=192.168.0.111/24**

**# static ip\_address=192.168.0.112/24**

static routers=192.168.0.1

static domain\_name\_servers=123.123.123.123 123.123.123.123

### /opt/hadoop/slaves

node1

node2

node3

### /opt/hadoop/core-site.xml

<configuration>

<property>

<name>hadoop.tmp.dir</name>

<value>/hdfs/tmp</value>

</property>

<property>

<name>fs.defaultFS</name>

<value>hdfs://**node1**:54310</value>

</property>

</configuration>

### /opt/hadoop/hdfs-site.xml

By changing this to 3 we will ensure that we have data locally on each node when we add new files (assuming we have 3 nodes in our cluster)

<property>

<name>dfs.replication</name>

<value>3</value>

</property>

### /opt/hadoop/yarn-site.xml

<property>

  <name>yarn.resourcemanager.resource-tracker.address</name>

  <value>**node1**:8025</value>

</property>

<property>

  <name>yarn.resourcemanager.scheduler.address</name>

  <value>**node1**:8035</value>

</property>

<property>

  <name>yarn.resourcemanager.address</name>

  <value>**node1**:8050</value>

</property>

<property>

  <name>yarn.nodemanager.aux-services</name>

  <value>mapreduce\_shuffle</value>

</property>

# Spark Installation

## Step 1 : Download latest release of Apache Spark from [Downloads | Apache Spark](http://spark.apache.org/downloads.html) and make sure that your Apache Spark build is compatible with Hadoop version installed on your cluster $ cd /opt/ $ wget http://www.apache.org/dyn/closer.lua/spark/spark-1.5.1/spark-1.5.1-bin-hadoop2.6.tgz $ tar –xvf spark-1.5.1-bin-hadoop2.6.tgz

## Step 2. Configuration in "spark-env.sh" file

## Create /opt/spark-1.5.1-bin-hadoop2.6/conf/spark-env.sh file from /opt/spark-1.5.1-bin-hadoop2.6/conf/spark-env.sh.template and add following lines to it

### export SPARK\_MASTER\_HOST=master

### export SPARK\_LOCAL\_IP=master

### export SPARK\_DIST\_CLASSPATH=$(/opt/Hadoop/bin/Hadoop classpath)

### export HADOOP\_CONF\_DIR=/opt/Hadoop/etc/Hadoop

### export YARN\_CONF\_DIR= /opt/Hadoop/etc/Hadoop

### export SPARK\_WORKER\_CORE=4

#### export JAVA\_HOME=/usr/lib/jvm/jdk-7-oracle-arm-vfp-hflt

## Create /opt/spark-1.5.1-bin-hadoop2.6/conf/spark-defaults.conf file from from /opt/spark-1.5.1-bin-hadoop2.6/conf/spark-defaults.conf.template file and add following lines to it.

### spark.master            spark://master:7077

### spark.serializer        org.apache.spark.serializer.KryoSerializer

### spark.io.compression.codec lz4

### spark.driver.cores 4

### spark.executor.cores 4

### spark.executor.extraJavaOptions -XX:+PrintGCDetails –Dkey=value –Dnumbers=”one two three”

### spark.driver.extraJavaOptions -Dhdp.version=current

### spark.yarn.am.extraJavaOptions -Dhdp.version=current

## Create /opt/spark-1.5.1-bin-hadoop2.6/conf/slaves add all the hostnames of spark slave nodes to it. <HOSTNAME OF YOUR MASTER NODE> <HOSTNAME OF YOUR SLAVE NODE 1> ... ... <HOSTNAME OF YOUR SLAVE NODE n> *[Repeat same above step 1 and 2 on other slave nodes]*

## Step 3. Start/Stop Spark using below commands Before starting Spark Demone processes make sure that you kave setup keyless SSH to all of your nodes (Reference: [How to setup Keyless SSH with non root users in CentOS](http://backtobazics.com/linux/how-to-setup-keyless-ssh-with-non-root-users-in-centos/))  and also disable firewall in all cluster nodes.  Than after use following commands for starting and stopping Spark Demons  $ sh /opt/spark-1.5.1-bin-hadoop2.6/sbin/[start-all.sh](http://start-all.sh/) $ sh /opt/spark-1.5.1-bin-hadoop2.6/sbin/[stop-all.sh](http://stop-all.sh/) Step 4. Start Spark shell using YARN Launch Spark Shell using following command $ /opt/spark-1.5.1-bin-hadoop2.6/bin/spark-shell --master yarn-client In case you want to start Spark Shell without using YARN use following command $ /opt/spark-1.5.1-bin-hadoop2.6/bin/spark-shell --master local[2] Here local[2] means start spark shell in standalone mode using two threads. Check [Building Spark Application JAR using Scala and SBT](http://backtobazics.com/big-data/spark/building-spark-application-jar-using-scala-and-sbt/) for building and running Spark Application, after you are done with Installing Apache Spark multi-node cluster with YARN.

**Hive Installation**

**Step 1:**  Download **Hive tar.**

**Command:**wget http://archive.apache.org/dist/hive/hive-2.1.0/apache-hive-2.1.0-bin.tar.gz

**Step 2:**Extract the **tar** file.

**Command:**tar -xzf apache-hive-2.1.0-bin.tar.gz

**Step 3:** Move apache-hive-2.1.0-bin.tar.gz

**Command:**sudo cp –R apache-hive-2.1.0-bin.tar.gz /usr/lib/hive/ apache-hive-2.1.0-bin.tar.gz

**Step 4:** Edit the “.bashrc” file to update the environment variables for user.

**Command:** sudo nano /home/hduser/.bashrc

Add the following at the end of the file:

# Set HIVE\_HOME

export HIVE\_HOME=/opt/apache-hive-2.1.0-bin

export PATH=$HIVE\_HOME/bin:$PATH

export HADOOP\_USER\_CLASSPATH\_FIRST=true

**Step 5:**Create **Hive** directories within**HDFS**. The directory **‘warehouse’** is the location to store the table or data related to hive.

**Command:**

* hdfs dfs -mkdir -p /user/hive/warehouse
* hdfs dfs -mkdir /tmp
* hdfs dfs -chmod g+w /user/hive/warehouse
* hdfs dfs -chmod g+w /tmp

**Step 6:**Run below command to make the changes work in same terminal.

**Command:** source .bashrc

**Step 7:**Set **Hadoop** path in **hive-env.sh**

**Command:** sudo nano conf/hive-env.sh

export HADOOP\_HOME=/opt/Hadoop

export HIVE\_CONF\_DIR=/opt/apache-hive-2.1.1-bin

**Step 8:** Edit **hive-site.xml**

**Command:**gedit conf/hive-site.xml

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?><!--

Licensed to the Apache Software Foundation (ASF) under one or more

contributor license agreements. See the NOTICE file distributed with

this work for additional information regarding copyright ownership.

The ASF licenses this file to You under the Apache License, Version 2.0

(the "License"); you may not use this file except in compliance with

the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software

distributed under the License is distributed on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and

limitations under the License.

-->

<configuration>

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:derby:;databaseName=/opt/apache-hive-2.1.0-bin/metastore\_db;create=true</value>

<description>

JDBC connect string for a JDBC metastore.

To use SSL to encrypt/authenticate the connection, provide database-specific SSL flag in the connection URL.

For example, jdbc:postgresql://myhost/db?ssl=true for postgres database.

</description>

</property>

<property>

<name>hive.metastore.warehouse.dir</name>

<value>/user/hive/warehouse</value>

<description>location of default database for the warehouse</description>

</property>

<property>

<name>hive.metastore.uris</name>

<value/>

<description>Thrift URI for the remote metastore. Used by metastore client to connect to remote metastore.</description>

</property>

<property>

<name>javax.jdo.option.ConnectionDriverName</name>

<value>org.apache.derby.jdbc.EmbeddedDriver</value>

<description>Driver class name for a JDBC metastore</description>

</property>

<property>

<name>javax.jdo.PersistenceManagerFactoryClass</name>

<value>org.datanucleus.api.jdo.JDOPersistenceManagerFactory</value>

<description>class implementing the jdo persistence</description>

</property>

</configuration>

**Step 9:** By default, Hive uses **Derby** database. Initialize Derby database.

**Command:**bin/schematool -initSchema -dbType derby

If permission denied, then give the hduser acouunt access to the /opt/

**Step 10**: Launch **Hive.**

**Command:**hive

**References**

Hadoop:

<http://www.widriksson.com/raspberry-pi-2-hadoop-2-cluster/>

<http://agilerule.blogspot.com/2016/06/how-to-install-maven-on-raspberry-pi.html>

<http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/>

<http://arturmkrtchyan.com/how-to-setup-multi-node-hadoop-2-yarn-cluster>

Spark:

<https://www.quora.com/How-do-I-set-up-Apache-Spark-with-Yarn-Cluster>

http://dblab.xmu.edu.cn/blog/1307-2/

Hive:

<https://www.edureka.co/blog/apache-hive-installation-on-ubuntu>

https://blogs.sap.com/2015/05/03/a-haddop-data-lab-project-on-raspberry-pi-part-24/