Installation and Configuration Documentation

Release 1.0.1

Oshin Prem

June 30, 2014

CONTENTS

1 HADOOP INSTALLATION **3**
1.1 SINGLE-NODE INSTALLATION 3
1.2 MULTI-NODE INSTALLATION 8

2 HIVE INSTALLATION **13**
2.1 INTRODUCTION 13
2.2 Hive Installation 13

3 SQOOP INSTALLATION **17**
3.1 INTRODUCTION 17
3.2 Stable release and Download 17
3.3 Prerequisites 17
3.4 Installation 17

4 IMPORTING DATA FROM HADOOP TO MYSQL **19**
4.1 Steps to install mysql 19
4.2 Using sqoop to perform import to hadoop from sql 19
4.3 Error points 19

5 Indices and tables **21**

Contents:

CHAPTER

ONE

HADOOP INSTALLATION

This section refers to the installation settings of Hadoop on a standalone system as well as on a system existing as a node in a cluster.

1.1 SINGLE-NODE INSTALLATION

1.1.1 Running Hadoop on Ubuntu (Single node cluster setup)

The report here will describe the required steps for setting up a single-node Hadoop cluster backed by the Hadoop Distributed File System, running on Ubuntu Linux. Hadoop is a framework written in Java for running applications on large clusters of commodity hardware and incorporates features similar to those of the Google File System (GFS) and of the MapReduce computing paradigm. Hadoop's HDFS is a highly fault-tolerant distributed file system and, like Hadoop in general, designed to be deployed on low-cost hardware. It provides high throughput access to application data and is suitable for applications that have large data sets.

Before we start, we will understand the meaning of the following:

DataNode:

A DataNode stores data in the Hadoop File System. A functional file system has more than one DataNode, with the data replicated across them.

NameNode:

The NameNode is the centrepiece of an HDFS file system. It keeps the directory of all files in the file system, and tracks where across the cluster the file data is kept. It does not store the data of these file itself.

Jobtracker:

The Jobtracker is the service within hadoop that farms out MapReduce to specific nodes in the cluster, ideally the nodes that have the data, or atleast are in the same rack.

TaskTracker:

A TaskTracker is a node in the cluster that accepts tasks- Map, Reduce and Shuffle operatons – from a Job Tracker.

Secondary Namenode:

Secondary Namenode whole purpose is to have a checkpoint in HDFS. It is just a helper node for namenode.

1.1.2 Prerequisites

Java 6 JDK

Hadoop requires a working Java 1.5+ (aka Java 5) installation.

Update the source list

```
user@ubuntu:~$  sudo apt-get update
```

or

Install Sun Java 6 JDK

Note:

If you already have Java JDK installed on your system, then you need not run the above command.

To install it

```
user@ubuntu:~$ sudo apt-get install sun-java6-jdk
```

The full JDK which will be placed in /usr/lib/jvm/java-6-openjdk-amd64 After installation, check whether java JDK is correctly installed or not, with the following command

```
user@ubuntu:~$ java -version
```

Adding a dedicated Hadoop system user

We will use a dedicated Hadoop user account for running Hadoop.

```
user@ubuntu:~$ sudo addgroup hadoop_group
user@ubuntu:~$ sudo adduser --ingroup hadoop_group hduser1
```

This will add the user hduser1 and the group hadoop_group to the local machine. Add hduser1 to the sudo group

```
user@ubuntu:~$ sudo adduser hduser1 sudo
```

Configuring SSH

The hadoop control scripts rely on SSH to peform cluster-wide operations. For example, there is a script for stopping and starting all the daemons in the clusters. To work seamlessly, SSh needs to be etup to allow password-less login for the hadoop user from machines in the cluster. The simplest ay to achive this is to generate a public/private key pair, and it will be shared across the cluster.

Hadoop requires SSH access to manage its nodes, i.e. remote machines plus your local machine. For our single-node setup of Hadoop, we therefore need to configure SSH access to localhost for the hduser user we created in the earlier.

We have to generate an SSH key for the hduser user.

```
user@ubuntu:~$ su - hduser1
hduser1@ubuntu:~$ ssh-keygen -t rsa -P ""
```

The second line will create an RSA key pair with an empty password.

Note:

P "", here indicates an empty password

You have to enable SSH access to your local machine with this newly created key which is done by the following command.

```
hduser1@ubuntu:~$   cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

The final step is to test the SSH setup by connecting to the local machine with the hduser1 user. The step is also needed to save your local machine's host key fingerprint to the hduser user's known hosts file.

```
hduser@ubuntu:~$ ssh localhost
```

If the SSH connection fails, we can try the following (optional):

- Enable debugging with ssh -vvv localhost and investigate the error in detail.
- Check the SSH server configuration in /etc/ssh/sshd_config. If you made any changes to the SSH server configuration file, you can force a configuration reload with sudo /etc/init.d/ssh reload.

1.1.3 INSTALLATION

Main Installation

- Now, I will start by switching to hduser

  ```
  hduser@ubuntu:~$ su - hduser1
  ```

- Now, download and extract Hadoop 1.2.0
- Setup Environment Variables for Hadoop

Add the following entries to .bashrc file

```
# Set Hadoop-related environment variables
export HADOOP_HOME=/usr/local/hadoop
# Add Hadoop bin/ directory to PATH
export PATH= $PATH:$HADOOP_HOME/bin
```

Configuration

hadoop-env.sh

Change the file: conf/hadoop-env.sh

```
#export JAVA_HOME=/usr/lib/j2sdk1.5-sun
```

to in the same file

```
# export JAVA_HOME=/usr/lib/jvm/java-6-openjdk-amd64  (for 64 bit)
# export JAVA_HOME=/usr/lib/jvm/java-6-openjdk-amd64  (for 32 bit)
```

conf/*-site.xml

Now we create the directory and set the required ownerships and permissions

```
hduser@ubuntu:~$ sudo mkdir -p /app/hadoop/tmp
hduser@ubuntu:~$ sudo chown hduser:hadoop /app/hadoop/tmp
hduser@ubuntu:~$ sudo chmod 750 /app/hadoop/tmp
```

The last line gives reading and writing permissions to the /app/hadoop/tmp directory

- Error: If you forget to set the required ownerships and permissions, you will see a java.io.IO Exception when you try to format the name node.

Paste the following between <configuration>

- In file conf/core-site.xml

```
<property>
    <name>hadoop.tmp.dir</name>
    <value>/app/hadoop/tmp</value>
    <description>A base for other temporary directories.</description>
</property>

<property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:54310</value>
    <description>The name of the default file system.  A URI whose
    scheme and authority determine the FileSystem implementation.  The
    uri's scheme determines the config property (fs.SCHEME.impl) naming
    the FileSystem implementation class.  The uri's authority is used to
    determine the host, port, etc. for a filesystem.</description>
</property>
```

- In file conf/mapred-site.xml

```
<property>
<name>mapred.job.tracker</name>
    <value>localhost:54311</value>
    <description>The host and port that the MapReduce job tracker runs
    at.  If "local", then jobs are run in-process as a single map
    and reduce task.
    </description>
</property>
```

- In file conf/hdfs-site.xml

```
<property>
    <name>dfs.replication</name>
    <value>1</value>
    <description>Default block replication.
    The actual number of replications can be specified when the file is created.
    The default is used if replication is not specified in create time.
    </description>
</property>
```

Formatting the HDFS filesystem via the NameNode

To format the filesystem (which simply initializes the directory specified by the dfs.name.dir variable). Run the command

```
hduser@ubuntu:~$ /usr/local/hadoop/bin/hadoop namenode -format
```

Starting your single-node cluster

Before starting the cluster, we need to give the required permissions to the directory with the following command

```
hduser@ubuntu:~$ sudo chmod -R 777 /usr/local/hadoop
```

Run the command

```
hduser@ubuntu:~$ /usr/local/hadoop/bin/start-all.sh
```

This will startup a Namenode, Datanode, Jobtracker and a Tasktracker on the machine.

```
hduser@ubuntu:/usr/local/hadoop$ jps
```

Errors:

1. **If by chance your datanode is not starting, then you have to erase the contents of the folder /app/hadoop/tmp** The command that can be used

   ```
   hduser@ubuntu:~:$ sudo rm -Rf /app/hadoop/tmp/*
   ```

2. **You can also check with netstat if Hadoop is listening on the configured ports.** The command that can be used

   ```
   hduser@ubuntu:~$ sudo netstat -plten | grep java
   ```

3. Errors if any, examine the log files in the /logs/ directory.

Stopping your single-node cluster

Run the command to stop all the daemons running on your machine.

```
hduser@ubuntu:~$ /usr/local/hadoop/bin/stop-all.sh
```

ERROR POINTS:

If datanode is not starting, then clear the tmp folder before formatting the namenode using the following command

```
hduser@ubuntu:~$ rm -Rf /app/hadoop/tmp/*
```

Note:

- The masters and slaves file should contain localhost.
- In /etc/hosts, the ip of the system should be given with the alias as localhost.
- Set the java home path in hadoop-env.sh as well bashrc.

1.2 MULTI-NODE INSTALLATION

1.2.1 Running Hadoop on Ubuntu Linux (Multi-Node Cluster)

From single-node clusters to a multi-node cluster

We will build a multi-node cluster merge two or more single-node clusters into one multi-node cluster in which one Ubuntu box will become the designated master but also act as a slave , and the other box will become only a slave.

1.2.2 Prerequisites

Configuring single-node clusters first,here we have used two single node clusters. Shutdown each single-node cluster with the following command

```
user@ubuntu:~$  bin/stop-all.sh
```

1.2.3 Networking

- The easiest is to put both machines in the same network with regard to hardware and software configuration.
- Update /etc/hosts on both machines .Put the alias to the ip addresses of all the machines. Here we are creating a cluster of 2 machines , one is master and other is slave 1

  ```
  hduser@master:$  cd /etc/hosts
  ```

- Add the following lines for two node cluster

  ```
  10.105.15.78    master  (IP address of the master node)
  10.105.15.43    slave1   (IP address of the slave node)
  ```

1.2.4 SSH access

The hduser user on the master (aka hduser@master) must be able to connect:

1. to its own user account on the master - i.e. ssh master in this context.
2. to the hduser user account on the slave (i.e. hduser@slave1) via a password-less SSH login.

- Add the hduser@master public SSH key using the following command

  ```
  hduser@master:~$  ssh-copy-id -i $HOME/.ssh/id_rsa.pub hduser@slave1
  ```

- Connect with user hduser from the master to the user account hduser on the slave.

1. From master to master

   ```
   hduser@master:~$  ssh master
   ```

2. From master to slave

   ```
   hduser@master:~$  ssh slave1
   ```

1.2.5 Hadoop

Cluster Overview

This will describe how to configure one Ubuntu box as a master node and the other Ubuntu box as a slave node.

Configuration

conf/masters

The machine on which bin/start-dfs.sh is running will become the primary NameNode. This file should be updated on all the nodes. Open the masters file in the conf directory

```
hduser@master/slave :~$ /usr/local/hadoop/conf
hduser@master/slave :~$ sudo gedit masters
```

Add the following line

```
Master
```

conf/slaves

This file should be updated on all the nodes as master is also a slave. Open the slaves file in the conf directory

```
hduser@master/slave :~$ /usr/local/hadoop/conf$ sudo gedit slaves
```

Add the following lines

```
Master
Slave1
```

conf/*-site.xml (all machines)

Open this file in the conf directory

```
hduser@master:~$ /usr/local/hadoop/conf$ sudo gedit core-site.xml
```

Change the fs.default.name parameter (in conf/core-site.xml), which specifies the NameNode (the HDFS master) host and port.

conf/core-site.xml (ALL machines .ie. Master as well as slave)

```
<property>
    <name>fs.default.name</name>
    <value>hdfs://master:54310</value>
    <description>The name of the default file system.  A URI whose
    scheme and authority determine the FileSystem implementation.  The
    uri’s scheme determines the config property (fs.SCHEME.impl) naming
    the FileSystem implementation class.  The uri’s authority is used to
    determine the host, port, etc. for a filesystem.</description>
</property>
```

conf/mapred-site.xml

Open this file in the conf directory

```
hduser@master:~$  /usr/local/hadoop/conf
hduser@master:~$  sudo gedit mapred-site.xml
```

Change the mapred.job.tracker parameter (in conf/mapred-site.xml), which specifies the JobTracker (MapReduce master) host and port.

conf/mapred-site.xml (ALL machines)

```
<property>
    <name>mapred.job.tracker</name>
    <value>master:54311</value>
    <description>The host and port that the MapReduce job tracker runs
    at.  If "local", then jobs are run in-process as a single map
    and reduce task.
    </description>
</property>
```

conf/hdfs-site.xml

Open this file in the conf directory

```
hduser@master:~$  /usr/local/hadoop/conf
hduser@master:~$  sudo gedit hdfs-site.xml
```

Change the dfs.replication parameter (in conf/hdfs-site.xml) which specifies the default block replication. We have two nodes available, so we set dfs.replication to 2.

conf/hdfs-site.xml (ALL machines)

Changes to be made

```
<property>
<name>dfs.replication</name>
<value>2</value>
<description>Default block replication.
    The actual number of replications can be specified when the file is created.
    The default is used if replication is not specified in create time.
    </description>
</property>
```

Formatting the HDFS filesystem via the NameNode

Format the cluster's HDFS file system

```
hduser@master:~$  /usr/local/hadoop$ bin/hadoop namenode -format
```

Starting the multi-node cluster

Starting the cluster is performed in two steps.

1. We begin with starting the HDFS daemons: the NameNode daemon is started on master, and DataNode daemons are started on all slaves (here: master and slave).
2. Then we start the MapReduce daemons: the JobTracker is started on master, and TaskTracker daemons are started on all slaves (here: master and slave).

Cluster is started by running the commnd on master

```
hduser@master:~$   /usr/local/hadoop
hduser@master:~$   bin/start-all.sh
```

By this command:

- The NameNode daemon is started on master, and DataNode daemons are started on all slaves (here: master and slave).
- The JobTracker is started on master, and TaskTracker daemons are started on all slaves (here: master and slave)

To check the daemons running , run the following commands

```
hduser@master:~$  jps
```

On slave, datanode and jobtracker should run.

```
hduser@slave:~$  /usr/local/hadoop$ jps
```

Stopping the multi-node cluster

To stop the multinode cluster , run the following command on master pc

```
hduser@master:~$   /usr/local/hadoop
hduser@master:~$   bin/stop-all.sh
```

ERROR POINTS:

1. **Number of slaves = Number of replications in hdfs-site.xml** also number of slaves = all slaves + master(if master is also considered to be a slave)
2. When you start the cluster, clear the tmp directory on all the nodes (master+slaves) using the following command

   ```
   hduser@master:~$   rm -Rf /app/hadoop/tmp/*
   ```

3. Configuration of /etc/hosts , masters and slaves files on both the masters and the slaves nodes should be the same.
4. If namenode is not getting started run the following commands:
   - To give all permissions of hadoop folder to hduser

     ```
     hduser@master:~$  sudo chmod -R 777 /app/hadoop
     ```

   - This command deletes the junk files which gets stored in tmp folder of hadoop

     ```
     hduser@master:~$  sudo rm -Rf /app/hadoop/tmp/*
     ```

CHAPTER TWO

HIVE INSTALLATION

This section refers to the installation settings of Hive on a standalone system as well as on a system existing as a node in a cluster.

2.1 INTRODUCTION

Apache Hive is a data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis. Apache Hive supports analysis of large datasets stored in Hadoop's HDFS and compatible file systems such as Amazon S3 filesystem. It provides an SQL-like language called HiveQL(Hive Query Language) while maintaining full support for map/reduce.

2.2 Hive Installation

2.2.1 Installing HIVE:

- Browse to the link: http://apache.claz.org/hive/stable/
- Click the apache-hive-0.13.0-bin.tar.gz
- Save and Extract it

  Commands

  ```
  user@ubuntu:~$  cd  /usr/lib/
  user@ubuntu:~$  sudo mkdir hive
  user@ubuntu:~$  cd Downloads
  user@ubuntu:~$  sudo mv apache-hive-0.13.0-bin /usr/lib/hive
  ```

2.2.2 Setting Hive environment variable:

Commands

```
user@ubuntu:~$  cd
user@ubuntu:~$  sudo gedit  ~/.bashrc
```

Copy and paste the following lines at end of the file

```
# Set HIVE_HOME
export HIVE_HOME="/usr/lib/hive/apache-hive-0.13.0-bin"
PATH=$PATH:$HIVE_HOME/bin
export PATH
```

2.2.3 Setting HADOOP_PATH in HIVE config.sh

Commands

```
user@ubuntu:~$ cd  /usr/lib/hive/apache-hive-0.13.0-bin/bin
user@ubuntu:~$ sudo gedit hive-config.sh
```

Go to the line where the following statements are written

```
# Allow alternate conf dir location.
HIVE_CONF_DIR="${HIVE_CONF_DIR:-$HIVE_HOME/conf"
export HIVE_CONF_DIR=$HIVE_CONF_DIR
export HIVE_AUX_JARS_PATH=$HIVE_AUX_JARS_PATH
```

Below this write the following

```
export HADOOP_HOME=/usr/local/hadoop    (write the path where hadoop file is there)
```

2.2.4 Create Hive directories within HDFS

Command

```
user@ubuntu:~$   hadoop fs -mkdir /usr/hive/warehouse
```

2.2.5 Setting READ/WRITE permission for table

Command

```
user@ubuntu:~$  hadoop fs -chmod g+w /usr/hive/warehouse
```

2.2.6 HIVE launch

Command

```
user@ubuntu:~$  hive
```

Hive shell will prompt:

OUTPUT ::

Shell will look like

```
Logging initialized using configuration in jar:file:/usr/lib/hive/apache-hive-0.13.0-bin/lib/hive- co
hive>
```

2.2.7 Creating a database

Command

```
hive> create database mydb;
```

OUTPUT

```
OK
Time taken: 0.369 seconds
hive>
```

2.2.8 Configuring hive-site.xml:

Open with text-editor and change the following property

```
<property>
    <name>hive.metastore.local</name>
    <value>TRUE</value>
    <description>controls whether to connect to remove metastore server or open a new metastore serve
</property>

<property>
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:mysql://usr/lib/hive/apache-hive-0.13.0-bin/metastore_db? createDatabaseIfNotExist=tr
    <description>JDBC connect string for a JDBC metastore</description>
</property>

<property>
    <name>javax.jdo.option.ConnectionDriverName</name>
    <value>com.mysql.jdbc.Driver</value>
    <description>Driver class name for a JDBC metastore</description>
</property>

<property>
    <name>hive.metastore.warehouse.dir</name>
    <value>/usr/hive/warehouse</value>
    <description>location of default database for the warehouse</description>
 </property>
```

2.2.9 Writing a Script

Open a new terminal (CTRL+ALT+T)

```
user@ubuntu:~$      sudo gedit sample.sql

create database sample;
use sample;
create table product(product int, productname string, price float)[row format delimited fields termin
describe product;
```

load data local inpath '/home/hduser/input_to_product.txt' into table product

```
select * from product;
```

SAVE and CLOSE

```
user@ubuntu:~$ sudo gedit input_to_product.txt
user@ubuntu:~$ cd /usr/lib/hive/apache-hive-0.13.0-bin/ $ bin/hive -f /home/hduser/sample.sql
```

CHAPTER

THREE

SQOOP INSTALLATION

This section refers to the installation settings of Sqoop.

3.1 INTRODUCTION

- Sqoop is a tool designed to transfer data between Hadoop and relational databases.
- You can use Sqoop to import data from a relational database management system (RDBMS) such as MySQL or Oracle into the Hadoop Distributed File System (HDFS), transform the data in Hadoop MapReduce, and then export the data back into an RDBMS. Sqoop automates most of this process, relying on the database to describe the schema for the data to be imported. Sqoop uses MapReduce to import and export the data, which provides parallel operation as well as fault tolerance. This document describes how to get started using Sqoop to move data between databases and Hadoop and provides reference information for the operation of the Sqoop command-line tool suite.

3.2 Stable release and Download

Sqoop is an open source software product of the Apache Software Foundation. Sqoop source code is held in the Apache Git repository.

3.3 Prerequisites

Before we can use Sqoop, a release of Hadoop must be installed and con?gured. Sqoop is currently supporting 4 major Hadoop releases - 0.20, 0.23, 1.0 and 2.0. We have installed Hadoop 2.2.0 and it is compatible with sqoop 1.4.4.We are using a Linux environment Ubuntu 12.04 to install and run sqoop. The basic familiarity with the purpose and operation of Hadoop is required to use this product.

3.4 Installation

To install the sqoop 1.4.4 we followed the given sequence of steps :

1. Download the sqoop-1.4.4.bin_hadoop-1.0.0.tar.gz file from www.apache.org/dyn/closer.cgl/sqoop/1.4.4
2. Unzip the tar ?le: sudo tar -zxvf sqoop-1.4.4.bin hadoop1.0.0.tar.gz
3. Move sqoop-1.4.4.bin hadoop1.0.0 to sqoop using command

   user@ubuntu:~$ sudo mv sqoop 1.4.4.bin hadoop1.0.0 /usr/local/sqoop

4. Create a directory sqoop in usr/lib using command

   user@ubuntu:~$ sudo mkdir /usr/lib/sqoop

5. Go to the zipped folder sqoop-1.4.4.bin_hadoop-1.0.0 and run the command

   user@ubuntu:~sudo mv ./* /usr/lib/sqoop

6. Go to root directory using cd command

   user@ubuntu:~$ cd

7. Open bashrc file using

   user@ubuntu:~$ sudo gedit ~/.bashrc

8. Add the following lines

   ```
   export SQOOP_HOME=¡usr/lib/sqoop
   export PATH=$PATH:$SQOOP_HOME/bin
   ```

9. To check if the sqoop has been installed successfully type the command

   ```
   sqoop version
   ```

CHAPTER

FOUR

IMPORTING DATA FROM HADOOP TO MYSQL

4.1 Steps to install mysql

- Run the command :sudo apt-get install mysql-server and give appropriate username and password.

4.2 Using sqoop to perform import to hadoop from sql

1. Download mysql-connector-java-5.1.28-bin.jar and move to /usr/lib/sqoop/lib using command

   ```
   user@ubuntu:~$ sudo cp mysql-connnectpr-java-5.1.28-bin.jar /usr/lib/sqoop/lib/
   ```

2. Login to mysql using command

   ```
   user@ubuntu:~$   mysql -u root -p
   ```

3. Login to secure shell using command

   ```
   user@ubuntu:~$  ssh localhost
   ```

4. Start hadoop using the command

   ```
   user@ubuntu:~$  bin/hadoop start-all.sh
   ```

5. Run the command

   ```
   user@ubuntu:~$ sqoop import -connect jdbc:mysql://localhost:3306/sqoop -username root -pasword a
   ```

This command imports the employees table from the sqoop directory of myql to hdfs.

4.3 Error points

1. Do check if the hadoop is in safe mode using command

   ```
   user@ubuntu:~$hadoop dfsadmin -safemode get
   ```

If you are getting safemode is on, run the command

```
user@ubuntu:~$hadoop dfsadmin -safemode leave
```

and again run the command

```
user@ubuntu:~$hadoop dfsadmin -safemode get
```

and confirm that you are getting safemode is off.

2. Do make sure that haoop is running before performing the import action.

CHAPTER

FIVE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*