1 Hadoop Primer

Hadoop is essentially an operating system for distributed processing. Its primary subsystems are HDFS and MapReduce (and Yarn).

2 Passwordless SSH

Before setting up Hadoop, setup passwordless SSH into your account.

Install an SSH server. If you're on Ubuntu based system, you can run:

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

If you haven't generated an SSH key, run:

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

This generates 2 files: `/.ssh/id_rsa` (the private key), and `/.ssh/id_rsa.pub` (the public key).

Add the public key to the authorized keys:

```
cat ~/.ssh/id_rsa.pub  >> ~/.ssh/authorized_keys2
```

You should now be able to login into your localhost:

```
ssh localhost
```

and it shouldn't ask you for a password.

3 JAVA_HOME

Find the actual location of java, and set `JAVA_HOME` environment variable:

At the top of `~/.bashrc`, add:

```
export JAVA_HOME=/usr/lib/jvm/default-java
```

4 Downloading Hadoop

Grab the latest version from: `http://hadoop.apache.org/`

As of this writing, I found: `hadoop-3.1.0.tar.gz`

You can unzip that into your home folder:

```
tar -xzf ~/Downloads/hadoop-3.1.0.tar.gz
```

This created `/home/alex/hadoop-3.1.0`. Now define a variable `HADOOP_HOME` to point to it. At the top of `~/.bashrc`, add:

```
export HADOOP_HOME=/home/alex/hadoop-3.1.0
export PATH="$HADOOP_HOME/bin:$PATH"
```

5 Configuring Hadoop

Unfortunately, Hadoop requires a few configuration files to be tweaked before it can run.

5.1 HDFS config

This goes into: `$HADOOP_HOME/etc/hadoop/hdfs-site.xml`:

```
<configuration>
    <property>
        <name>dfs.datanode.data.dir</name>
        <value>file:///home/alex/hadoop-3.1.0/hdfs/datanode</value>
    </property>
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>file:///home/alex/hadoop-3.1.0/hdfs/namenode</value>
    </property>
</configuration>
```

The `dfs.datanode.data.dir` tells datanodes where to store file blocks.

Thje `dfs.namenode.name.dir` tells namenodes where to persist the file metadata and transaction log.

You should create these directories:

```
mkdir -p /home/alex/hadoop-3.1.0/hdfs/datanode
mkdir -p /home/alex/hadoop-3.1.0/hdfs/namenode
```

5.2 Hadoop config

This goes into: `$HADOOP_HOME/etc/hadoop/core-site.xml`:

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

The `fs.defaultFS` indicates default file system, HDFS in our case.

5.3 Format HDFS

Just run:

```
hdfs namenode -format
```

6 Running Stuff

Go to `$HADOOP_HOME/sbin` directory, and run:

```
./start-all.sh
```

This starts HDFS, etc.

Notice there is also `stop-all.sh` script. As well as scripts to start/stop individual services.

7 Using HDFS

You should now have HDFS running. You can create your own home folder:

```
hdfs dfs -mkdir /user
hdfs dfs -mkdir /user/alex
```

Now when you run:

```
hdfs dfs -ls /
```

You should see:

```
Found 1 items
drwxr-xr-x   - alex supergroup          0 2018-04-09 18:54 /user
```

Lets create a file `hello.csv`:

```
1,john,doe,1997-02-09
2,jane,doe,1982-03-14
3,bob,johnson,2002-04-21
4,john,jackson,1999-12-31
5,jack,johnson,1998-11-03
```

We can dump it into our home folder via:

```
hdfs dfs -copyFromLocal hello.csv hello.csv
```

The file is now in HDFS:

```
$ hdfs dfs -cat hdfs://localhost/user/alex/hello.csv
1,john,doe,1997-02-09
2,jane,doe,1982-03-14
3,bob,johnson,2002-04-21
4,john,jackson,1999-12-31
5,jack,johnson,1998-11-03
```

8 Hive Primer

Hive is a SQL engine. It lets us run SQL queries against HDFS files.

9 Downloading Hive

Grab the latest version from: `http://hive.apache.org/`

As of this writing, I found: `apache-hive-2.3.3-bin.tar.gz`

You can unzip that into your home folder:

```
tar -xzf ~/Downloads/apache-hive-2.3.3-bin.tar.gz
```

This created `/home/alex/apache-hive-2.3.3-bin`. Now define a variable `HIVE_HOME` to point to it. At the top of `~/.bashrc`, add:

```
export HIVE_HOME=/home/alex/apache-hive-2.3.3-bin
export PATH="$HIVE_HOME/bin:$PATH"
```

10 Configuring Hive

Create the warehouse folder: where Hive will save data.

```
hdfs dfs -mkdir /user/hive
hdfs dfs -mkdir /user/hive/warehouse
```

We now need to give Hive a place to save metadata on tables. For this tutorial, we'll use PostgreSQL.

```
sudo su - postgres
psql
create role hivemetastore with login password 'hivemetastore321';
create database hivemetastore with owner hivemetastore;
```

Now we need: `$HIVE_HOME/conf/hive-site.xml`:

```
<configuration>
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:postgresql://localhost/hivemetastore</value>
</property>
<property>
  <name>javax.jdo.option.ConnectionDriverName</name>
  <value>org.postgresql.Driver</value>
</property>
<property>
```

```
  <name>javax.jdo.option.ConnectionUserName</name>
  <value>hivemetastore</value>
</property>
<property>
  <name>javax.jdo.option.ConnectionPassword</name>
  <value>hivemetastore321</value>
</property>
</configuration>
```

The above are pretty self explanatory.

Now initialize schema for Hive to use:

```
schematool -dbType postgres -initSchema
schematool -dbType postgres -info
```

11 Running Hive

We are finally ready to run Hive:

```
hive
```

To exit, just type `exit`.

Let's adjust the location of our `hello.csv` file.

```
hdfs dfs -mkdir hello
hdfs dfs -mv hello.csv hello
```

Now run `hive` again, and then type:

```
CREATE EXTERNAL TABLE hello
    ( uid INT, fname STRING, lname STRING, dob DATE)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LOCATION '/user/alex/hello';
```

Now we can run queries against this table:

```
hive> show tables;
OK
hello
Time taken: 0.03 seconds, Fetched: 1 row(s)
```

```
hive> describe hello;
OK
uid                     int
fname                   string
lname                   string
dob                     date
Time taken: 0.08 seconds, Fetched: 4 row(s)
```

```
hive> select * from hello;
OK
1 john doe 1997-02-09
2 jane doe 1982-03-14
3 bob johnson 2002-04-21
4 john jackson 1999-12-31
5 jack johnson 1998-11-03
Time taken: 0.135 seconds, Fetched: 5 row(s)

hive> select lname,fname from hello where dob > '1999-01-01';
OK
johnson bob
jackson john
Time taken: 0.37 seconds, Fetched: 2 row(s)
```

Let us now create another table, using the results from our external table:

```
create table hello1999 as
    select lname,fname
    from hello
    where dob > '1999-01-01';
```

Now exit and see where Hive stores the data. The original file is still there:

```
hdfs dfs -ls -R
drwxr-xr-x   - alex supergroup          0 2018-04-09 19:31 hello
-rw-r--r--   3 alex supergroup        121 2018-04-09 19:01 hello/hello.csv
```

The new table went into the `/user/hive/warehouse` folder:

```
hdfs dfs -ls -R /user/hive/
drwxr-xr-x   - alex supergroup          0 2018-04-09 19:39 /user/hive/warehouse
drwxr-xr-x   - alex supergroup          0 2018-04-09 19:39 /user/hive/warehouse/hello199
-rwxr-xr-x   3 alex supergroup         25 2018-04-09 19:39 /user/hive/warehouse/hello199
```

We can read this data directly:

```
hdfs dfs -cat /user/hive/warehouse/hello1999/000000_0
johnson bob
jackson john
```

There's a delimiter there... it's binary $0x01$. It's configurable—when you create a table, you have a lot of control over the format of the data.