Big Data Project (Hadoop)

Saturday, January 21, 2023 12:43 PM

1. Our First steps

a. Installations :

For our project , we will start by installing distributed Hadoop using a cluster of 2 Nodes and Master , and this installation will be performed locally using virtuelle machines containing :

- Ubuntu server 22.04 (3GB RAM , 30 GB Disk ,3 vCPUs) => 1 Master and 2 Nodes

b. Configuration initial de cluster :

- Update & Upgrade OS :[**sudo apt update**] [**sudo apt-get upgrade**]
- Install ssh-server and ssh-client .
  - [sudo apt-get install openssh-server]
  - Enable SSH service : [sudo systemctl enable ssh]
  - Enable and start SSH immediately : [sudo systemctl enable ssh --now]
  - Start the ssh service by typing : [sudo systemctl start ssh]
  - Test it by login into the system : [ssh userName@Your-server-name-IP] or [ssh ec2-user@ec2-aws-ip-here] if u using remote machine.
  - Another way to test : [sudo systemctl status ssh]
- Configure firewall and open port 22 (in our case by default no fiewall exist)
  - [sudo ufw allow ssh]
  - [sudo ufw enable]
  - [sudo ufw status]
- Set up static address for our server ubuntu so we can work using wireless network :
  - Listing interfaces (don't choose LOOPBACK interface) . (for our case we chose enp0s3)
  - Verify that this interface not managed by the cloud (a feature of ubuntu server) :
    - [sudo nano /etc/cloud/cloud.cfg.d/subiquity-disable-cloudinit-networking.cfg] => verify is disabled .
  - Now go to this file [sudo nano /etc/netplan/00-installer-config.yaml]

```
network:
  version: 2
  ethernets:
    enp0s3:
      addresses: [192.168.1.10/24]
      gateway4: 192.168.1.1
      nameservers:
        addresses: [1.1.1.1, 1.0.0.1]_
```

  - After this run this command and see the change [ip a]
  - Run this command to kill the network (restart the interfaces) : [sudo netplan try] then [sudo netplan apply]
  - In the end reboot your system .[sudo reboot]
  - Other alternatives is to add hostname to /etc/hosts :

Adding a hostname to the /etc/hosts file is a way to manually map a hostname to an IP address. The /etc/hosts file is a simple text file that maps hostnames to IP addresses, and is used by the operating system to resolve hostnames to IP addresses before it queries DNS (Domain Name System) servers. By adding a hostname to the /etc/hosts file, you can override the DNS resolution and force the operating system to use a specific IP address for a given hostname. This can be useful for testing or troubleshooting network issues.

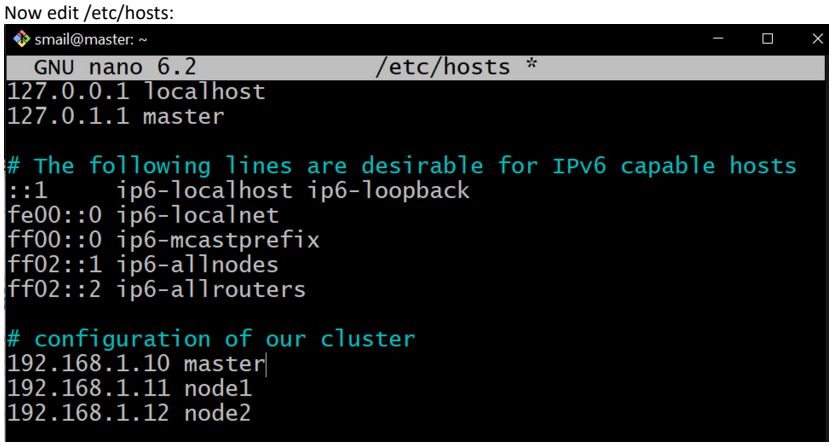
To change the hostname on a Linux or Unix-based operating system, you can use the "hostnamectl" command or manually edit the "/etc/hostname" file.

Using hostnamectl:
hostnamectl set-hostname new_hostname

Manually editing the /etc/hostname file:
1. Open a terminal window
2. Open the /etc/hostname file in a text editor (e.g. nano, vim)
3. Replace the current hostname with the new hostname
4. Save and close the file
5. Reboot the system for the changes to take effect

For master vm => master
For node1 vm => node1
For node2 vm => node2

Now edit /etc/hosts:



- Un peu de geographique sur ssh :) => The .ssh directory in your home directory is used by SSH to store various files related to authentication and encryption. Some of the commonly used files in this directory include:

- **`authorized_keys`**: This file contains a list of public keys that are authorized to access the current user's account. The public keys in this file are used to authenticate the corresponding private keys on the client side.
- **`id_rsa`** and **`id_rsa.pub`**: These files are the private and public key pair generated using the **`ssh-keygen`** command. The private key is stored on the client side and is used to authenticate the user. The public key is stored on the server side and is used to verify the authenticity of the client.
- **`config`** : This file is used to define the ssh config settings like hostname, username, and ports to be used for different hosts.

- `known_hosts`: This file contains a list of server's host keys that the client has previously connected to. It is used to verify the identity of the server during future connections and protect against man-in-the-middle attacks.
- `known_hosts2`: This file is similar to known_hosts, but it stores the host keys in a different format that is compatible with other SSH implementations.
- `id_dsa`, `id_ecdsa`, `id_ed25519` : These files are similar to id_rsa, but they use different algorithms for generating the key pairs.

Note: The files have the same name but different extensions, the one with the .pub extension is the public key and the one without it is the private key.

It's important to note that the permissions on the **.ssh** directory and its files should be set correctly. The .ssh directory should have 700 permissions and the files inside it should have 600 permissions.

- Now we will use our host machine Windows and connect to ubuntu server :
  - Windows [ssh-keygen -t rsa]
  - Windows [ssh-copy-id user@ubuntu_server_ip]
  - Windows test connection [ssh user@ubuntu_server_ip]
  - If u wanna connect from the other side [ubuntu to windows] u can do the same steps that have been done for windows . And make sure that ssh server is set up for winodws (as we have done for ubuntu) .

c. Installing Hadoop

i. Requirements

- SSH
- Java 8 (MUST java 8 for Hadoop 3.x.x)

  - [sudo apt install openjdk-8-jre-headless] and [sudo apt install openjdk-8-jdk-headless] then verify [java -version] and [javac -version]

- 3 server
  - Master server (192.168.1.10/ubuntu22.04,hostname=Smail)
  - Slave server (192.168.1.11/ubuntu22.04,hostname=Smail1)
  - Slave server (192.168.1.12/ubuntu22.04,hostname=Smail2)
- Hadoop 3.2.4
  - Download : [wget https://dlcdn.apache.org/hadoop/common/hadoop-3.2.4/hadoop-3.2.4.tar.gz]
  - Untar the archive and move it to /usr/local :
    - [tar -xzvf hadoop-3.2.4.tar.gz]
    - Remane [mv hadoop-3.2.4 hadoop]
    - Move [mv hadoop /usr/local]
  - We need to update our default environment variables to include the JAVA HOME and Hadoop binary directories:
    - Where java is installed [update-alternatives --display java]

```
smail@hadoop:~$ update-alternatives --display java
java - auto mode
  link best version is /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java
  link currently points to /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java
  link java is /usr/bin/java
  slave java.1.gz is /usr/share/man/man1/java.1.gz
/usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java - priority 1081
  slave java.1.gz: /usr/lib/jvm/java-8-openjdk-amd64/jre/man/man1/java.1.gz
```

  - Now add PATH /etc/environment : [sudo nano /etc/enviornment]
    - 1st modification : [PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/local/hadoop/bin:/usr/local/hadoop/sbin"]
    - 2ed modification : [JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64"]

- Now create hadoop user and give him the right permissions :

```
# adduser hadoop
# usermod -aG hadoop hadoop
# chown hadoop:root -R /usr/local/hadoop
# chmod g+rwx -R /usr/local/hadoop
```

- Now log in as hadoop user then generate an ssh key (only for the master node) :

```
# su - hadoop
# ssh-keygen -t rsa
```

- Now login as the hadoop user and copy the SSH key to all Hadoop Nodes. Again, you only need to complete this step on the Hadoop Master:

```
# su - hadoop
$ ssh-copy-id hadoop@master
$ ssh-copy-id hadoop@node1
$ ssh-copy-id hadoop@node2
```

ii. Configuring the Hadoop Master :

- Open the /usr/local/hadoop/etc/hadoop/core-site.xml file and enter the following :

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

- Next, open the /usr/local/hadoop/etc/hadoop/hdfs-site.xml file and add the the follwing :

```
<configuration>
 <property> # only for master
  <name>dfs.namenode.name.dir</name>
  <value>/usr/local/hadoop/data/nameNode</value>
 </property>
 <property> # only for nodes
  <name>dfs.datanode.data.dir</name>
  <value>/usr/local/hadoop/data/dataNode</value>
 </property>
 <property> # for master and nodes
  <name>dfs.replication</name>
  <value>2</value>
 </property>
</configuration>
```

- Open the /usr/local/hadoop/etc/hadoop/workers file and add these two lines (your Nodes) and don't forget to remove localhost if you want to work with two slaves and one master not containing DataNode .

```
Node1
Node2
```

- And now we will copy the configuration in /etc/hadoop to the nodes :

```
Scp /usr/local/hadoop/etc/hadoop/* node1:/usr/local/hadoop/etc/hadoop/
Scp /usr/local/hadoop/etc/hadoop/* node2:/usr/local/hadoop/etc/hadoop/
```

- Then format the HDFS file system :

```
Source /etc/environment
Hdfs namenode -format
```

- And now we can start hdfs :

```
start-dfs.sh
```

- Now you should see this in :
  - Master => [jps] => (jps)&(SecondaryNameNode)&(NameNode)
  - Nodes => [jps] => (jps)&(DataNode)
  - And you can visit this Web UI on your browser within your network : http://192.168.1.10:9870/

iii. Now configuring the scheduler of the jobs (Yarn) for Hadoop :

Hadoop , on its own , can schedule any jobs so we need to run Yarn so we can schedule jobs on our Hadoop cluster (so we need to configure this 2 files) :

- Yarn-site.xml (each node and master)

```
<property>
  <name>yarn.resourcemanager.hostname</name>
  <value>master</value>
 </property>

Or

<property>
        <name>yarn.acl.enable</name>
        <value>0</value>
</property>
<property>
        <name>yarn.resourcemanager.hostname</name>
        <value>master</value>
</property>
<property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
</property>
<property>
        <name>yarn.nodemanager.resource.memory-mb</name>
        <value>2826</value>
</property>
<property>
        <name>yarn.scheduler.maximum-allocation-mb</name>
```

```
        <value>2726</value>
</property>
<property>
        <name>yarn.scheduler.minimum-allocation-mb</name>
        <value>128</value>
</property>
<property>
        <name>yarn.nodemanager.vmem-check-enabled</name>
        <value>false</value>
</property>
<property>
        <name>yarn.nodemanager.resource.cpu-vcores</name>
        <value>2</value>
</property>
```

- Mapred-site.xml (each node and master)

```
<property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
</property>
<property>
        <name>yarn.app.mapreduce.am.resource.mb</name>
        <value>416</value>
</property>
<property>
        <name>mapreduce.map.memory.mb</name>
        <value>256</value>
</property>
<property>
        <name>mapreduce.reduce.memory.mb</name>
        <value>256</value>
</property>
```

And we set up these environment variables:

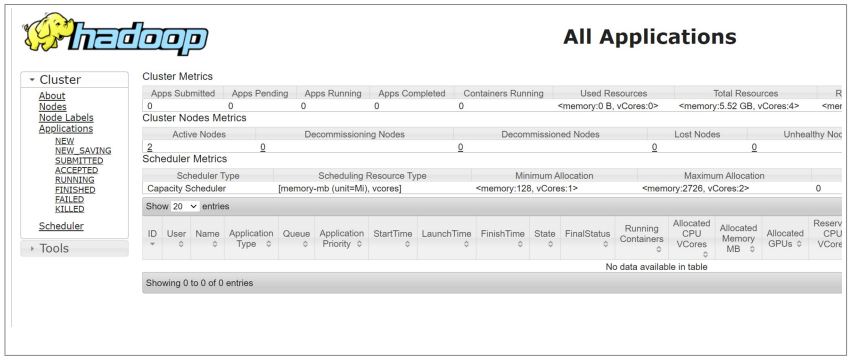
```
export HADOOP_HOME="/usr/local/hadoop"
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME
```

- And run this command to start yarn : [start-yarn.sh]
- And verify that everything work correctly : [yarn node -list]

```
hadoop@master:/usr/local/hadoop/etc/hadoop$ yarn node -list
2023-01-22 13:49:53,162 INFO client.RMProxy: Connecting to ResourceManager at ma
ster/192.168.1.10:8032
Total Nodes:2
         Node-Id             Node-State Node-Http-Address       Number-of-Runnin
g-Containers
     node2:42269                RUNNING        node2:8042
               0
     node1:43671                RUNNING        node1:8042
               0
```

There are no containers running because we haven't started any jobs yet.

- In the end we can now access Web Interface for Job Scheduler [http://192.168.1.10:8088/cluster] :



- Running an Example of Hadoop Jobs :

We can now run a sample Hadoop job and schedule it on our cluster.The example we will run is to use MapReduce to calculate PI.Run the following command to run the job:

```
[ yarn jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.4.jar pi 16 1000 ]
```

```
Job Finished in 44.361 seconds
Estimated value of Pi is 3.14250000000000000000
```

It will take several minutes to complete.

- I have faced some problems with mapreduce application (giving me this error) :
  - Error :
  - Solutions :

- Edit the file mapred-site.xml :

```
<!-- fixing the issues of mapreduce application-->
<property>
  <name>yarn.app.mapreduce.am.env</name>
  <value>HADOOP_MAPRED_HOME=/usr/local/hadoop</value>
</property>
<property>
  <name>mapreduce.map.env</name>
  <value>HADOOP_MAPRED_HOME=/usr/local/hadoop</value>
</property>
<property>
  <name>mapreduce.reduce.env</name>
  <value>HADOOP_MAPRED_HOME=/usr/local/hadoop</value>
</property>
<!-- end of fixing issues -->
```

And add this 5th property and filled its value by the result of this command :

```
export HADOOP_CLASSPATH=$(hadoop classpath)
echo $HADOOP_CLASSPATH
```

```
/usr/local/hadoop/etc/hadoop:/usr/local/hadoop/share/hadoop/common/lib/
*:/usr/local/hadoop/share/hadoop/common/
*:/usr/local/hadoop/share/hadoop/hdfs:/usr/local/hadoop/share/hadoop/hdfs/lib/
*:/usr/local/hadoop/share/hadoop/hdfs/*:/usr/local/hadoop/share/hadoop/mapreduce/lib/
*:/usr/local/hadoop/share/hadoop/mapreduce/
*:/usr/local/hadoop/share/hadoop/yarn:/usr/local/hadoop/share/hadoop/yarn/lib/
*:/usr/local/hadoop/share/hadoop/yarn/*
```

```
<property>
<name>mapreduce.application.classpath</name>
<value></value>
</property>
```

2. Project :

For our project our goal is to perform a number of analysis using brut data from Kaggle website [https://www.kaggle.com/datasets/sobhanmoosavi/us-accidents] called US accidents (2016-2021).

i. Get to know your data :

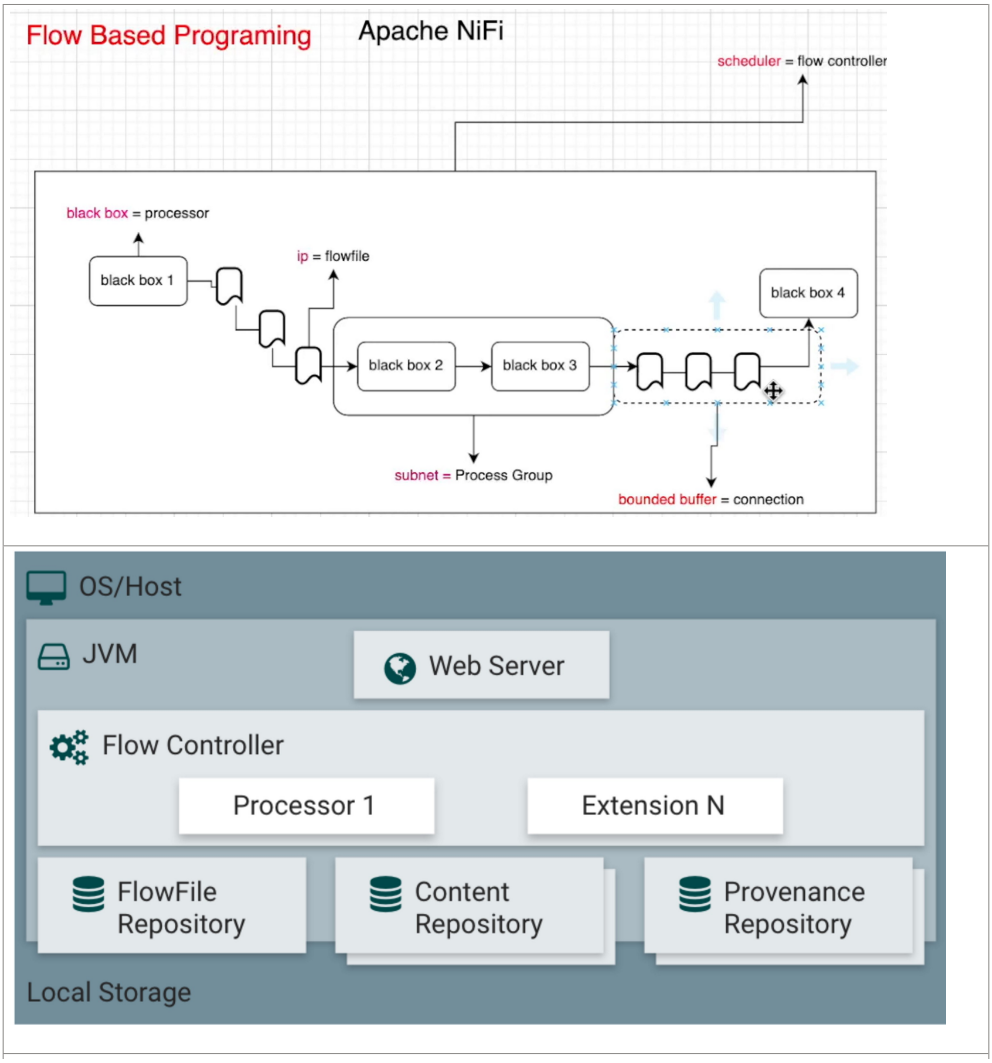
ID	This is a unique identifier of the accident record.
Severity	Shows the severity of the accident, a number between 1 and 4, where 1 indicates the least impact on traffic (i.e., short delay
Start_Time	Shows start time of the accident in local time zone.
End_Time	Shows end time of the accident in local time zone. End time here refers to when the impact of accident on traffic flow was dismissed.
Start_Lat	Shows latitude in GPS coordinate of the start point.
Start_Lng	Shows longitude in GPS coordinate of the start point.
End_Lat	Shows latitude in GPS coordinate of the end point
End_Lng	Shows longitude in GPS coordinate of the end point.
Distance(mi)	The length of the road extent affected by the accident.
Description	Shows a human provided description of the accident
Number	Shows a human provided description of the accident.
Street	Shows the street name in address field.
Side	Shows the relative side of the street (Right/Left) in address field.
City	Shows the city in address field.
County	Shows the county in address field.
State	Shows the state in address field.
Zipcode	Shows the zipcode in address field.
Timezone	Shows timezone based on the location of the accident (eastern, central, etc.).
Airport_Code	Denotes an airport-based weather station which is the closest one to location of the accident.
Weather_Timestamp	Shows the time-stamp of weather observation record (in local time).
Temperature(F)	Shows the temperature (in Fahrenheit).
Wind_Chill(F)	Shows the wind chill (in Fahrenheit).
Humidity(%)	Shows the humidity (in percentage).
Pressure(in)	Shows the air pressure (in inches).
Visibility(mi)	Shows visibility (in miles).
Wind_Direction	Shows wind direction.
Wind_Speed(mph)	Shows wind speed (in miles per hour).
Precipitation(in)	Shows precipitation amount in inches, if there is any.
Weather_Condition	Shows the weather condition (rain, snow, thunderstorm, fog, etc.)
Amenity	A POI annotation which indicates presence of amenity in a nearby location.

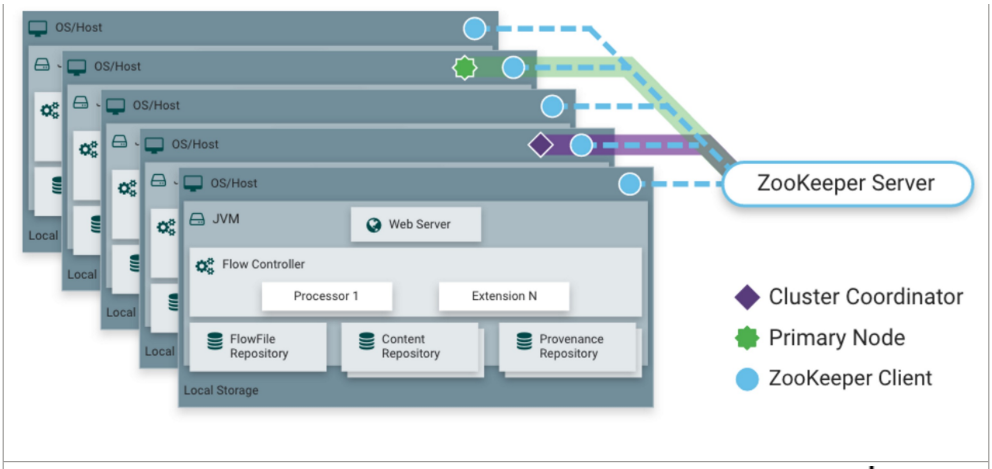
Bump	A POI annotation which indicates presence of speed bump or hump in a nearby location.
Crossing	A POI annotation which indicates presence of crossing in a nearby location.
Give_Way	A POI annotation which indicates presence of give_way in a nearby location.
Junction	A POI annotation which indicates presence of junction in a nearby location.
No_Exit	A POI annotation which indicates presence of no_exit in a nearby location.
Railway	A POI annotation which indicates presence of railway in a nearby location.
Roundabout	A POI annotation which indicates presence of roundabout in a nearby location.
Station	A POI annotation which indicates presence of station in a nearby location.
Stop	A POI annotation which indicates presence of stop in a nearby location.
Traffic_Calming	A POI annotation which indicates presence of traffic_calming in a nearby location.
Traffic_Signal	A POI annotation which indicates presence of traffic_signal in a nearby location.
Turning_Loop	A POI annotation which indicates presence of turning_loop in a nearby location.
Sunrise_Sunset	Shows the period of day (i.e. day or night) based on sunrise/sunset.
Civil_Twilight	Shows the period of day (i.e. day or night) based on civil twilight.
Nautical_Twilight	Shows the period of day (i.e. day or night) based on nautical twilight.
Astronomical_Twilight	Shows the period of day (i.e. day or night) based on astronomical twilight.

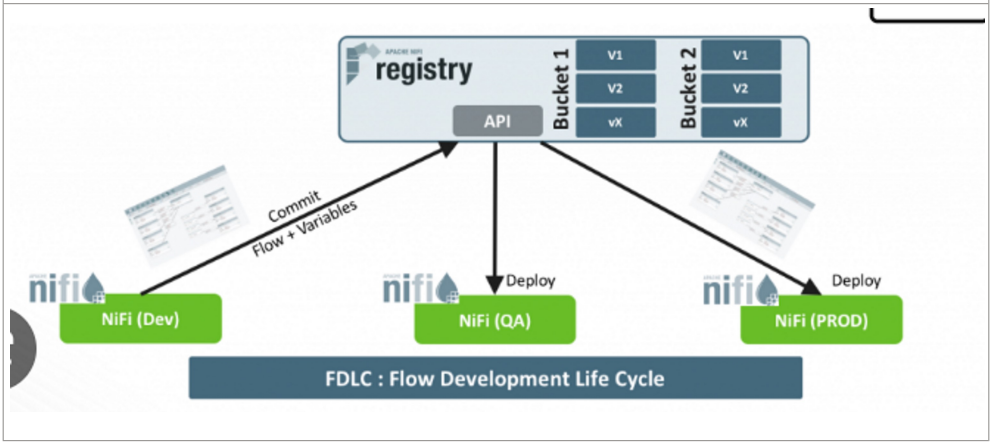
ii. Tools :

a. Nifi

Our first tool that we will use for exploring and using this dataset is Nifi , which is a tool for automatize the work flow of data from the producer to the consumer , and it is build on Flow Based Programming (indicated in the pictures below) . for more information about this tool we can consult the documentation of Nifi [https://nifi.apache.org/docs.html] or if you like getting information by following visual videos we can follow this playlist in this case : [Apache NiFi - 2022]



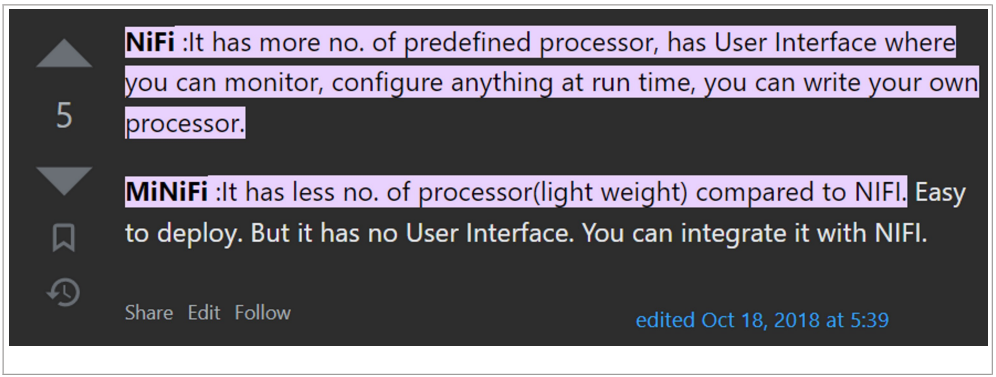




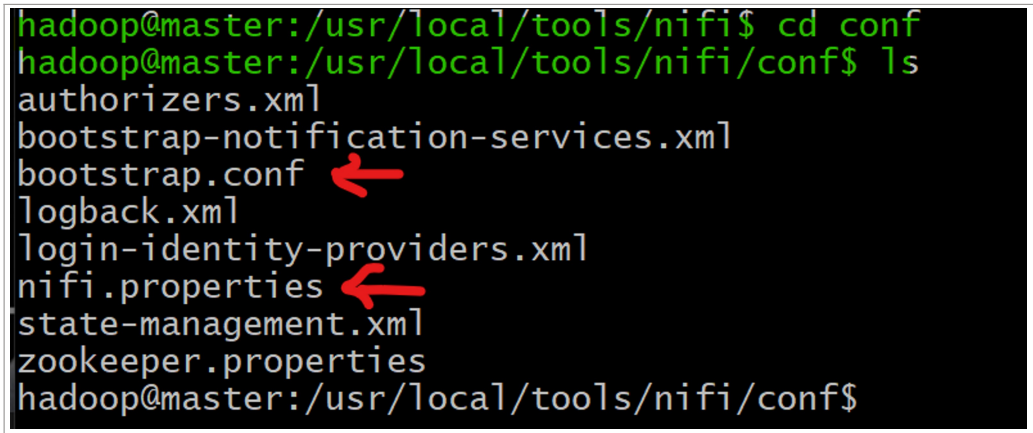
i. The installation of Nifi tool:

Consult the Nifi Apache website and download the binary version (1.2.0 for February 2017) => [wget https://archive.apache.org/dist/nifi/1.1.2/nifi-1.1.2-bin.tar.gz] in the home directory of Hadoop user .

Note : as we need an interface for our processing data to interact with and understand more the work flow , we going to use the full featured version of Nifi and not the mini one :



After downloading the Nifi binary tar gz file , we going unzip it and put it into a directory called **tools as nifi directory** in **/usr/local** : [tar -xzvf ...] & [sudo mv nifi /usr/local/tools/]



- Bootstrap.conf : allow us to configure how nifi should be started and that's from the perspective of resources , and this file will also help us to optimize the way nifi works in the cases when we run out of the memory for example , and this file also help us to optimize the memory giving to for JVM (cause the more resources given to JVM more the more heavy collector garbage will be , ...)

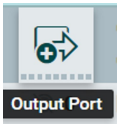
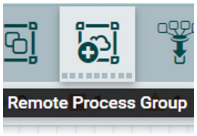
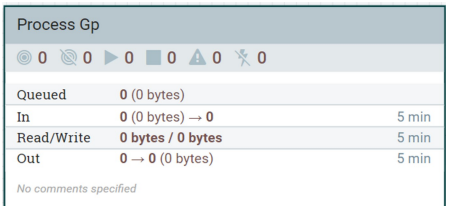
```
# How long to wait after telling NiFi to shutdown before >
graceful.shutdown.seconds=20

# Disable JSR 199 so that we can use JSP's without runnin>
java.arg.1=-Dorg.apache.jasper.compiler.disablejsr199=true

# JVM memory settings
java.arg.2=-Xms512m
java.arg.3=-Xmx512m
```

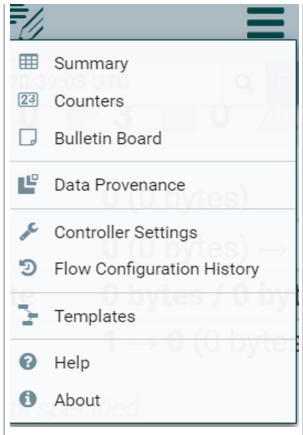
- Nifi.properties: by default in the latest versions nifi securely generate user and password for you out of the box , but we can change that . And this some properties of web ui :

```
# web properties #
nifi.web.war.directory=./lib
nifi.web.http.host=
nifi.web.http.port=8080
nifi.web.https.host=
nifi.web.https.port=
nifi.web.jetty.working.directory=./work/jetty
nifi.web.jetty.threads=200
```

- To run nifi : [/usr/local/tools/nifi/bin$ **./nifi.sh help] then [** ./nifi.sh start] then go to [/usr/local/tools/nifi/logs] then consult [logs of nifi bootstrap] and get the user and password credentials [cat nifi-app.log | grep Generated] .
- In the end consult nifi using Web interface by using the port specified in the nifi.properties file , here is an example [http://192.168.1.10:8080/nifi/]
- To change the user's credentials auto-generated :
  - Step 1 : go to binary file /bin and then write : [./nifi.sh set-single-user-credentials admin 0000].
  - Step 2 : restart your nifi app . (you can consult last modification of credentials by typing cat login-identity-providers.xml | grep admin]

ii. Overview of Nifi web interface :

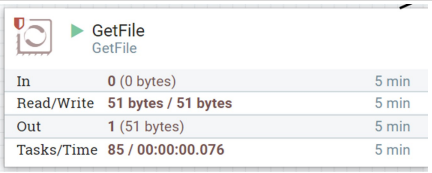
Component	Description	How to work with it
	Input Port : they act as a mechanism for transforming data into a processor group	To bring it to canvas , you need only to click on it and drag it to your canvas , when you drag it to canvas , you won't get the same amount of options as the processors cause this is a simple reusable component all it ask you is to give it a name (unique name) .
	Same mechanism : the output port provide a mechanism on transferring data between process groups (outside process group , so input it's brings data inside and outputs brings data outside)	Same steps .
	Remote process group : when you put an input port and output port on the main canvas , nifi will consider it as side to side communication (we're not gonna go over that concept , we gonna convert it later)	
	 Double click to enter Process group or right click on it and then enter group .	
	This component allows you to duplicate the data passing throw a component in the data flow (this is at the fly - streaming data) .	

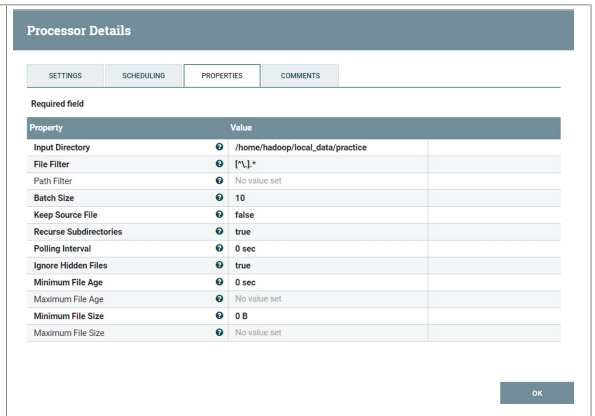


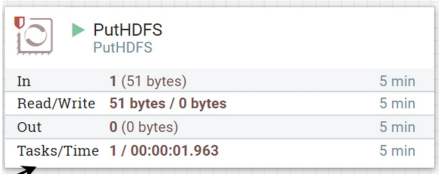
• Summary : give the summary of your nifi instance .

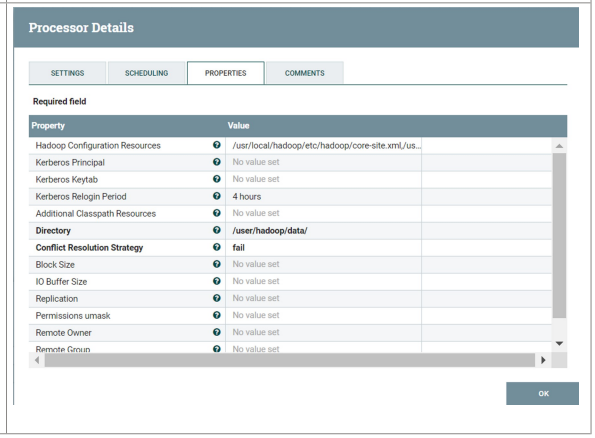
iii. Moving the data from our Host machine to the HDFS:

1) **Step 1**: create a directory in the machine containing Nifi : [mkdir local_data]
2) **Step 2**: create 2 processor :

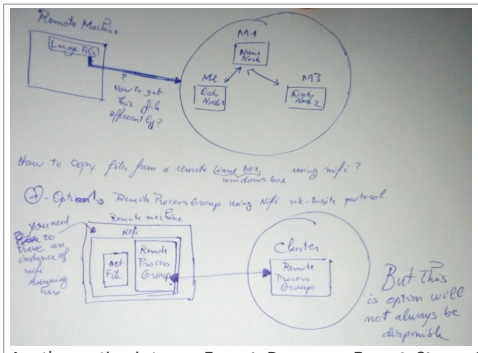




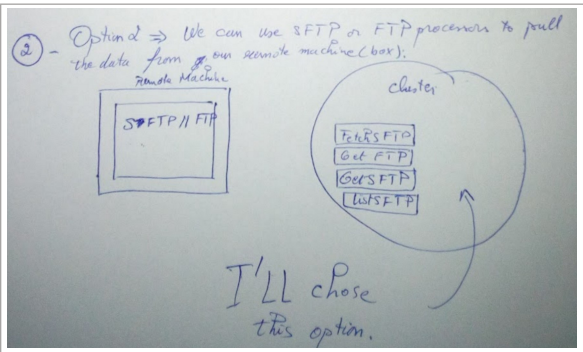




3) **Step 3: preparing the source file environnement :**



Another option is to use **ExecuteProcess** or **ExecuteStreamCommand** to execute a custom script that will SCP to your remote Linux instance. Otherwise there is a JIRA for a SCP processor: https://issues.apache.org/jira/browse/NIFI-539



[How to set up an sftp server on windows 10]
[Data Cleaning In Python (Practical Examples)]
[Clean Excel Data with Python and Pandas - 5 Minute Python Scripts - Full Code Along Walkthrough]

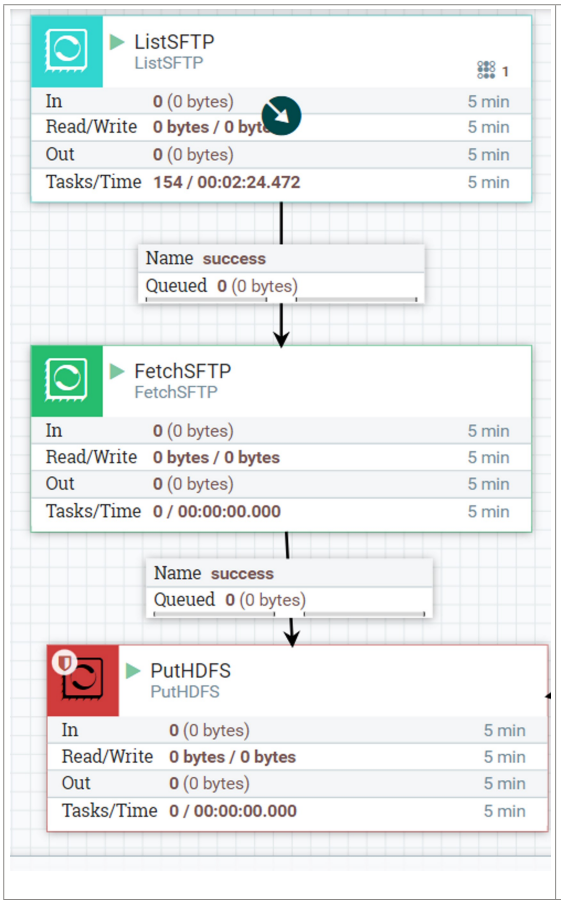
Tasks for installing SFTP server on windows 10:

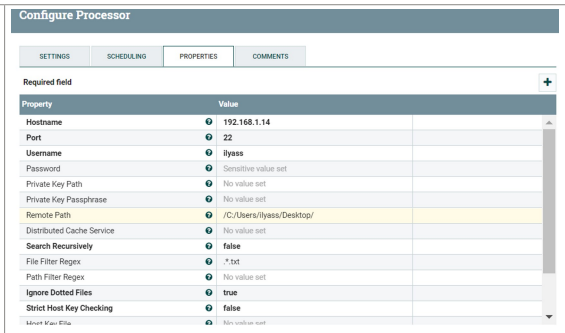
1- Install the Open SSH server . (search on Features in Windows search)
2- Create an inbound rule in Firewall for port no 22.(control panel => windows defender firewall=> advanced settings => Inbound Rules => new Rule => select Port => next => TCP & type the port 22 => next => next => next => type the rule [ssh server] => finish)
3- Start the Open SSH services (services => OpenSSH Server => properties => Startup type = automatic)
4- Download and install WinSCP tool as SFTP Client . (google it and download it)

5- Connect to SFTP Server .

For the sake of simplicity , I'm not going to work with ssh but I'll use only password … .

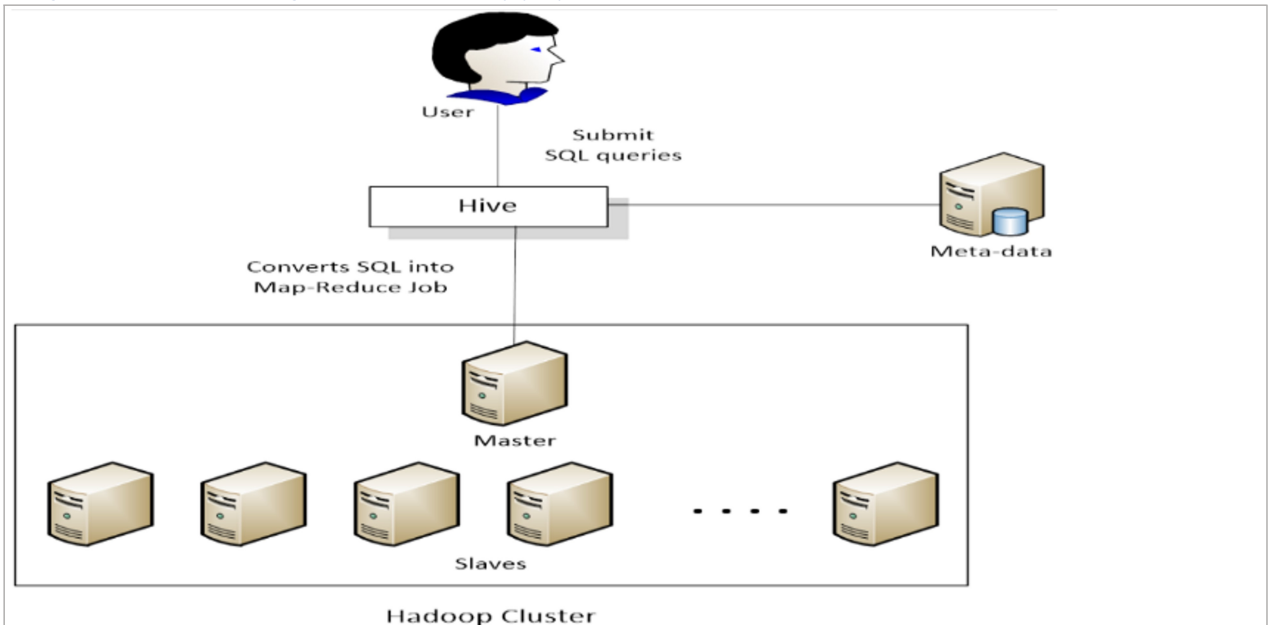
4) **Step 4 : configuring nifi processes to retrieve our dataset from the SFTP server that is already set up : [why exactly I have used this processors** https://community.cloudera.com/t5/Community-Articles/How-to-Retrieve-files-from-a-SFTP-server-using-NiFi-GetSFTP/ta-p/245430]



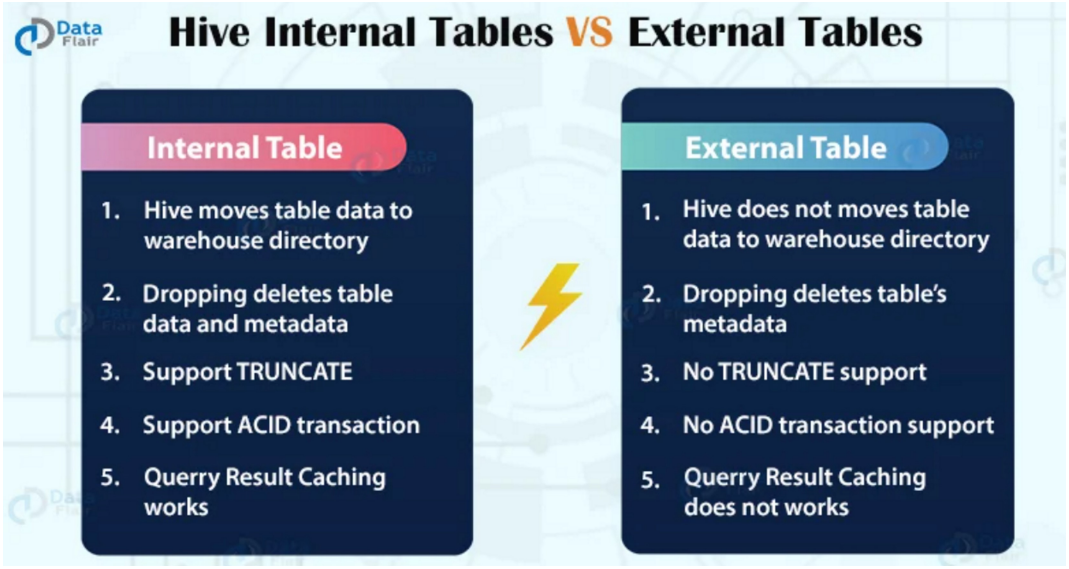


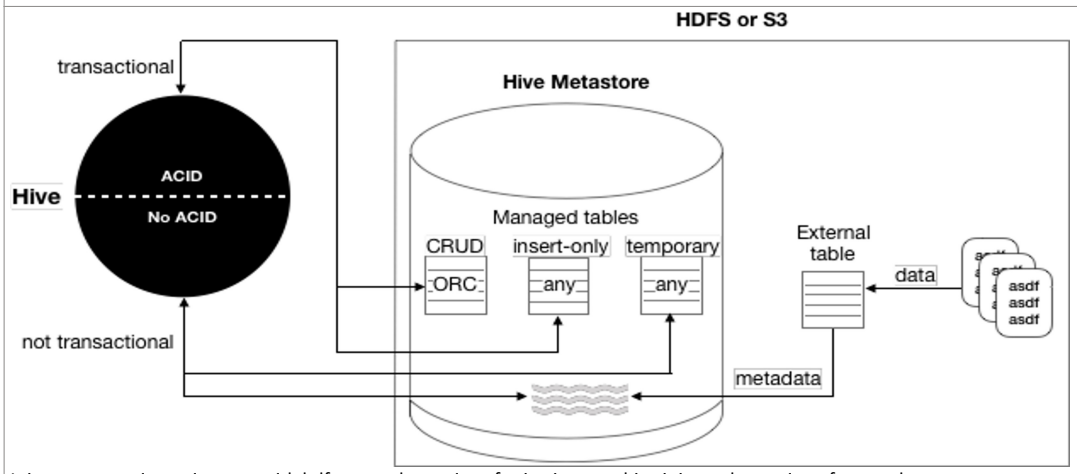
This is the configuration for ListSFTP , and it is the same configuration for FetchSFTP . Furthermore , we will use the PutHDFS that is already set up from the last example .

iv. Using Hive to retrieve data using HiveQL (like a result of query on RDBMS):



We will use Mysql as an RDBMS to store Meta-data of hive tables . (in this project we not going to use the embedded Derby RDBMS , because it is not centralized , if we use for example 2 sessions of hive on our cluster , each one will store different meta data than the other , so whenever I create hive table from the first session , I cannot access it then from the second session …)





It is a query engine to interact with hdfs as an alternative of using java , so hive it is an abstraction of mapreduce .

Example 1 :without partitioning

```
CREATE TABLE mytable (col1 INT, col2 STRING, col3 DATE) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
LOAD DATA INPATH '/path/to/data.txt' INTO TABLE mytable;
```

Example 2 : using static partitioning

```
CREATE TABLE mytable_partitioned (col1 INT, col2 STRING, col3 DATE) PARTITIONED BY (col4 STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
LOAD DATA INPATH '/path/to/data.txt' INTO TABLE mytable_partitioned PARTITION (col4='value');
```

Example 3: using dynamic partitioning

```
CREATE TABLE mytable_dynamic (col1 INT, col2 STRING, col3 DATE) PARTITIONED BY (col4 STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
# enable dynamic partitioning mode
SET hive.exec.dynamic.partition=true;
SET hive.exec.dynamic.partition.mode=nonstrict;
# load data ( note that this mechanism can slow down for large amount of data )
FROM mytable INSERT INTO TABLE mytable_dynamic PARTITION (col4) SELECT col1, col2, col3, col4;
```

Note => you can specify the location where the data will be stored in hdfs , in case you didn't specified it , the tables will be created in default path [/user/hive/warehouse] , you can also see this partitions by hdfs commands or using hive [show partitions tableName]

1) **Step 1** : download hive 3.1.2 [wget https://downloads.apache.org/hive/hive-3.1.2/apache-hive-3.1.1-bin.tar.gz]
2) **Step 2** : Now we will extract this archive and move it to /usr/local/tools directory and then add it to environnement variables . [tar -xzvf apache-hive-3.1.2-bin.tar.gz] => [mv apache-hive-3.1.2-bin hive] => [mv hive /usr/local/tools/] => [sudo mv hive/ /usr/local/tools/]
3) **Step 3**: install Mysql [sudo apt-get install mysql-server] => username=root & password= => [sudo mysql -u root -p] => [CREATE USER 'hadoop'@'localhost' IDENTIFIED BY '0000';]=>[GRANT ALL PRIVILEGES ON *.* TO 'hadoop'@'localhost';]=>[FLUSH PRIVILEGES;]
4) **Step 4**: to be able to connect hive to mysql for storing meta-data you need to have Mysql Connector (JDBC) in lib directory of hive : [wget https://repo1.maven.org/maven2/mysql/mysql-connector-java/8.0.28/mysql-connector-java-8.0.28.jar] => [mv mysql-connector-j-8.0.32.jar hive/lib/]
5) **Step 5**: configuration [cd /usr/local/tools/hive/conf/] => [nano hive-site.xml]

```
<configuration>
    <property>
        <name>javax.jdo.option.ConnectionURL</name>
        <value>jdbc:mysql://localhost/metastore?createDatabaseIfNotExist=true</value>
        <description>metadata is stored in a MySQL server</description>
    </property>
```

```
    <property>
        <name>javax.jdo.option.ConnectionDriverName</name>
        <value>com.mysql.jdbc.Driver</value>
        <description>MySQL JDBC driver class</description>
    </property>
    <property>
        <name>javax.jdo.option.ConnectionUserName</name>
        <value>hadoop</value>
        <description>user name for connecting to mysql server</description>
    </property>
    <property>
        <name>javax.jdo.option.ConnectionPassword</name>
        <value>0000</value>
        <description>password for connecting to mysql server</description>
    </property>
    <property>
        <name>hive.metastore.client.cache.maxSize</name>
        <value>10Mb</value>
        <description>Max size for hive metastore client local cache</description>
    </property>
</configuration>
```

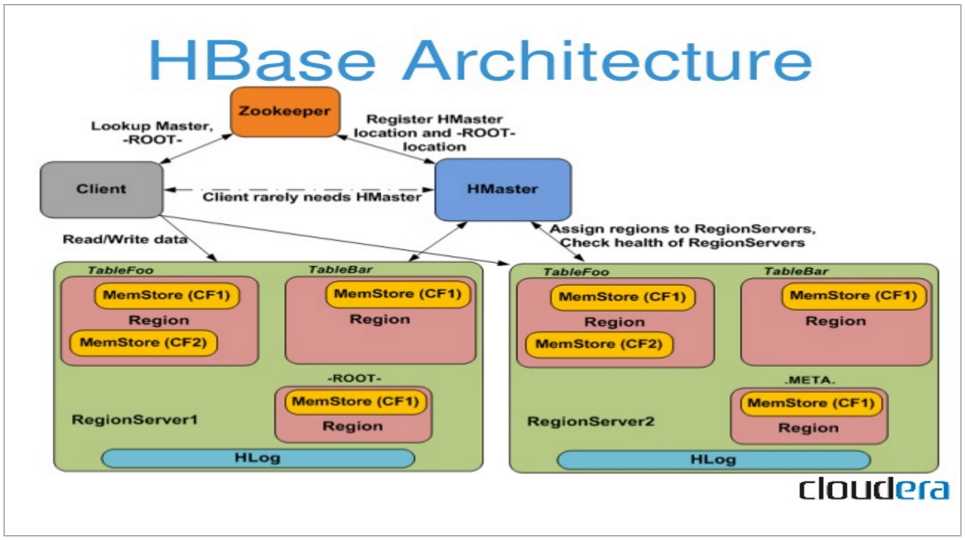
6) **Step 6:** this is a command [schematool -dbType mysql -initSchema] when it is triggered , it creates the database declared in the configuration and all tables under this DB for storing our meta-deta. Pay attention of the libraries used by hive and hadoop are compatible like (guava and Slfqj) , if you faced any issue try to use those libraries that is exist in the classpath (like hadoop) in hive library .
7) **Step 7**: add hive to Path and then , execute hive command [hive]:
   a) **Sub-step 7.1**: cat only the 1st row to file [hdfs dfs -cat /user/hadoop/data/US_Accidents_Dec21_updated.csv | head -1 >> header.txt]
   b) **Sub-step 7.2** :
   c) **Sub-step 7.3** : modifiy the file header.txt

```
CREATE External TABLE  us_accidents(
        ID string,
        Severity INT,
        Start_Time Date,
        End_Time Date,
        Start_Lat Double,
        Start_Lng Double,
        End_Lat Double,
        End_Lng Double,
        Distance_mi Double,
        Description varchar(300),
        Number float,
        Street string,
        Side char(1),
        City string,
        County string,
        State string,
        Zipcode string,
        Country string ,
        Timezone string,
        Airport_Code string,
        Weather_Timestamp date,
        Temperature_F float,
        Wind_Chill_F float,
        Humidity_Per float,
        Pressure_in float,
        Visibility_mi float,
        Wind_Direction string,
        Wind_Speed_mph float,
        Precipitation_in float,
        Weather_Condition string,
        Amenity BOOLEAN,
        Bump BOOLEAN,
        Crossing BOOLEAN,
        Give_Way BOOLEAN,
        Junction BOOLEAN,
        No_Exit BOOLEAN,
        Railway BOOLEAN,
        Roundabout BOOLEAN,
        Station BOOLEAN,
        Stop BOOLEAN,
        Traffic_Calming BOOLEAN,
        Traffic_Signal BOOLEAN,
        Turning_Loop BOOLEAN,
        Sunrise_Sunset string,
        Civil_Twilight string,
        Nautical_Twilight string,
        Astronomical_Twilight string
) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

   d) **Sub-step 7.4**: [LOAD DATA INPATH '/user/hadoop/data/US_Accidents_Dec21_updated.csv' INTO TABLE us_accidents;]
   e) **Sub-step 7.4**: test your data set [select state from us_accidents where id ="A-1";]
   f) **Sub-step 7.5**: perform a partition over the table

v. Hbase

Hbase is a data model that is designed to provide us with quick random access to huge amounts of structured or semi structured data . Whereas , Hdfs is good for sequential data access , but it lacks the random read/write capability . Hbase runs on top of the HDFS and provides us with a read and write to it. (so Data procedures => can store data into => HDFS) and (Data consumers <= can access data from <= HDFS)



1) Downloading Hbase :
   [wget https://dlcdn.apache.org/hbase/2.5.2/hbase-2.5.2-bin.tar.gz]
2) Unzip the file and configure it for Master Node of our cluster:
   [tar -xzvf hbase-2.5.2-bin.tar.gz]
   [mv hbase-2.5.2 hbase]
   [sudo mv hbase /usr/local/tools]
3) Open hbase-env.sh in conf directory (in our case we've already set up the JAVA_HOME in bashrc file):

```
# The java implementation to use.  Java 1.8+ required.
# export JAVA_HOME=/path to the home of java
```

4) Open The file ~/.bashrc file and add HBASE_HOME & then add bin of Hbase to path

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


# Hadoop variables ( HDFS and Home )
export HADOOP_HOME="/usr/local/hadoop"
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"

# MapReduce and Yarn
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME

#add Hadoop to path
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin

# Hive
export HIVE_HOME="/usr/local/tools/hive"
export PATH=$PATH:$HIVE_HOME/bin

# Nifi
export NIFI_HOME="/usr/local/tools/nifi"
export PATH=$PATH:$NIFI_HOME/bin

# HBase
export HBASE_HOME="/usr/local/tools/hbase"
export PATH=$PATH:$HBASE_HOME/bin
```

5) Add properties in the file hbase-site.xml :

```
<property>
    <name>hbase.rootdir</name>
    <value>hdfs://master:9000/user/hbase</value>
</property>
<property>
  <name>hbase.cluster.distributed</name>
  <value>true</value>
</property>
<property>
    <name>hbase.zookeeper.quorum</name>
    <value>master,node1,node2</value>
</property>
```

```
 <property>
     <name>hbase.zookeeper.property.clientPort</name>
     <value>2181</value>
 </property>
  <!-- here u have to set the path where u want Hbase to>
 <property>
     <name>hbase.zookeeper.property.dataDir</name>
     <value>usr/local/tools/hbase/zookeeper</value>
 </property>
```

```
# configure regionservers that is exist in hbase/regionservers
Node 1
Node 2
```

6) Add this parameters to hbase-env.sh : (and because I have faced some issues due to the compatibility between hbase and hadoop then I had to prevent hbase from looking for lib in hadoop classpath)

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

```
export HBASE_OPTS="$HBASE_OPTS -XX:+UseConcMarkSweepGC"
```

```
export HBASE_SSH_OPTS="-p 22 -l hadoop"
```

```
export HBASE_MANAGES_ZK=true
```

```
# export HBASE_DISABLE_HADOOP_CLASSPATH_LOOKUP="true"
# issues due version comptability between hbase 2.5.2 and hadoop 3.2.1
( for this moment I have left this comment as it is , but for not having the error of conflicting bindings of slf4j... , I had to remane the slf4j-reload..
To slfs4j-reload..txt so the hbase can use those libraires exist in hadoop classpath /share/hadoop/common/lib/ ) for each node .
```

7) Now copy the whole installation of hbase (from /usr/local/tools into each node in /usr/local/tools) and configure this things :

```
Add hbase to the path (for each node):
# Java
export JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64"


# Hadoop variables ( HDFS and Home )
export HADOOP_HOME="/usr/local/hadoop"
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/nati>
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"

# MapReduce and Yarn
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME

#add Hadoop to path
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin

# HBase
export HBASE_HOME="/usr/local/tools/hbase"
export PATH=$PATH:$HBASE_HOME/bin
```

```
Only specify this properties in the file hbase-site.xml (each node ) :
<property>
     <name>hbase.rootdir</name>
     <value>hdfs://master:9000/user/hbase</value>
 </property>
 <property>
   <name>hbase.cluster.distributed</name>
   <value>true</value>
 </property>
```

```
Repeat the 6th step for each node .

Before u start hbase make sure no other hbase process in the nodes of cluster are running :
kill $(jps | grep <process_name> | awk '{print $1}')
```

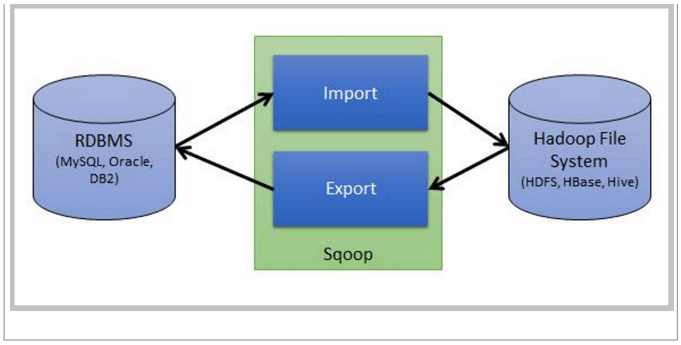
```
http://192.168.1.10:16010/master-status
```

```
hive -e "SELECT * FROM us_accidents INTO OUTFILE '/tmp/hive_data.csv' ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';"
```

```
hbase shell create us_accidents, '<column_family>' put '<hbase_table>', '<row_key>', '<column_family>:<column_name>', '<value>'
```

vi. Sqoop :

It is a tool that allow us to transfer bulk data between Hadoop HDFS and the others databases (RDBMS) .



1) Download : [wget https://archive.apache.org/dist/sqoop/1.4.7/]
2) Unzip the file : [tar -xzvf sqoop-1.4.7.tar.gz]
3) Rename to sqoop : [mv sqoop-1.4.7 sqoop]
4) Move to /usr/local/tools : [sudo mv sqoop /usr/local/tools/]
5) Configure variable environnement :
   a) Access .bashrc : [nano ~/.bashrc]
   b) Add this :

```
# Sqoop
export SQOOP_HOME="/usr/local/tools/sqoop"
export PATH=$PATH:$SQOOP_HOME/bin
```

   c) Source the bashrc file : [source ~/.bashrc]
6) Configure Sqoop :
   a) [cd $SQOOP_HOME/conf]
   b) [mv sqoop-env-template.sh sqoop-env.sh]
   c) [nano sqoop-env.sh] and add :

```
#Set path to where bin/hadoop is available
export HADOOP_COMMON_HOME=/usr/local/hadoop
#Set path to where hadoop-*-core.jar is available
export HADOOP_MAPRED_HOME=/usr/local/hadoop
```

7) We need mysql-connectors cause we will work with Mysql as our Main RDBMS :
   a) Download [wget http://ftp.ntu.edu.tw/MySQL/Downloads/Connector-J/mysql-connector-java-5.1.48.tar.gz]
   b) Unzip [tar -xzvf mysql-connector-java-5.1.48.tar.gz]
   c) [cd mysql-connector-java-5.1.48]
   d) [mv mysql-connector-java-5.1.48-bin.jar /usr/local/tools/sqoop/lib/]
8) Verification : [sqoop-version]

vii. Visualization of Data using ether [Tableau] or [Power BI]:
   1) Load data to Tableau [Fichier Text]