INF 729
Introduction au framework Hadoop

Accès aux VM de travail : ***Réseau école :*** *ssh <login>@ssh.enst.fr* ***Bridge :*** *ssh ubuntu@137.194.211.146* ***Connection machine :*** *ssh tp-hadoop-<No>*

Sommaire :

Installation d'Hadoop

Sur chaque machine :

Vérification de mise à jour de l'environnement :

sudo apt update
sudo apt upgrade
sudo apt install ssh
sudo apt install pdsh
sudo su
echo "ssh" > /etc/pdsh/rcmd_default
sudo apt install openjdk-8-jdk

Vérification du fichier hostname et hosts :

Dans hosts : <adresseip des machines du cluster> <hostname>
Dans hostname : <nom machine>

Générer clé publique : ssh-keygen -t rsa -P ""
Enregistrement clé pub : cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
Test local host : ssh localhost

Si mot de passe demandé :

ssh-keygen -t rsa -P " -f ~/.ssh/id_rsa
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod 0600 ~/.ssh/authorized_keys

Téléchargement et décompression :

cd
wget https://mirrors.sonic.net/apache/hadoop/common/hadoop-3.3.1/hadoop-3.3.1.tar.gz && tar -zxvf hadoop-3.3.1.tar.gz && rm -rf hadoop-3.3.1.tar.gz
mv hadoop-3.3.1/ hadoop
cd hadoop

mkdir -p data/dataNode data/nameNode

Sur la machine maitre :

cd hadoop

nano etc/hadoop/hadoop-env.sh → export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/

nano etc/hadoop/core-site.xml

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://<hostname machine maître>:9000</value>
  </property>
</configuration>
```

nano etc/hadoop/hdfs-site.xml

```
<configuration>
<property>
<name>dfs.namenode.name.dir</name>
<value>file://<chemin data/nameNode></</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>file://<chemin data/dataNode></value>
</property>
<property>
<name>dfs.replication</name>
<value>2</value>
</property>
</configuration>
```

nano etc/hadoop/mapred-site.xml

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
  <property>
    <name>mapreduce.application.classpath</name>
<value>$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*:$HADOOP_
MAPRED_HOME/share/hadoop/mapreduce/lib/*</value>
  </property>
</configuration>
```

nano etc/hadoop/yarn-site.xml

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

nano etc/hadoop/workers : lister les hostnames des datanodes

envoi des conf aux workers : scp ~/hadoop/etc/hadoop/* <worker>:~/hadoop/etc/hadoop/

Paramétrage de Yarn sur la machine maitre :

```
export HADOOP_HOME="~/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
```

Dans etc/hadoop/yarn-site.xml :

```
<property>
<name>yarn.resourcemanager.hostname</name>
<value>tp-hadoop-9</value>
</property>
```

formatage en HDFS : hdfs namenode -format

Test du cluster :

démarrage hdfs : sbin/start-dfs.sh
démarrage yarn : sbin/start-yarn.sh

vérification sur API : http://137.194.211.146/<maitre>/9870/dfshealth.html
vérification par terminal (sur chaque machine) : jps
vérification des logs : cat logs/*.log

création d'un dossier de travail sur hdfs : bin/hadoop fs -mkdir -p /user/***<username>***
création du dossier input : bin/hadoop fs -mkdir -p input

Lancement d'un appli Wordcount : *bin/hadoop fs jar WC/wc.jar WordCount input/file01 outputX*

Rappel des variables à mettre dans .bashrc

```
export JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64/"
export PATH=${JAVA_HOME}/bin:${PATH}
export HIVE_HOME="/home/ubuntu/hive"
export PATH=$PATH:$HIVE_HOME/bin
export CLASSPATH=$CLASSPATH:/usr/local/Hadoop/lib/*:.
export CLASSPATH=$CLASSPATH:/usr/local/hive/lib/*:.
export DERBY_HOME="/home/ubuntu/derby"
export PATH=$PATH:$DERBY_HOME/bin
export
CLASSPATH=$CLASSPATH:$DERBY_HOME/lib/derby.jar:$DERBY_HO
ME/lib/derbytools.jar
```

création d'un fichier app.java : copier/coller le code dans <app>.java
compilation (dans le dossier hadoop/app) :

../bin/hadoop com.sun.tools.javac.Main <app>.java
jar cf <app>.jar WordCount*.class

remplir le dossier input : bin/hadoop fs -put <file> input
lancer l'app.jar : bin/hadoop jar <app>.jar app input output

arrêt hdfs : sbin/stop-dfs.sh
arrêt yarn : sbin/stop-yarn.sh

Affichage des rapports : bin/hadoop fs -report

au besoin :

suppression de la partition hdfs : rm -rf /tmp/hadoop-<username>/dfs
trouver le clusterId : cat ~/hadoop/data/nameNode/current

Installation de Zookeeper

Apache Zookeeper

Téléchargement de zookeeper : wget https://mirroir.wptheme.fr/apache/zookeeper/zookeeper-3.6.3/apache-zookeeper-3.6.3-bin.tar.gz && tar -zxvf apache-zookeeper-3.6.3-bin.tar.gz && mv apache-zookeeper-3.6.3-bin zookeeper && rm -rf apache-zookeeper-3.6.3-bin.tar.gz

Création d'un fichier de config en stand alone : cd zookeeper && nano conf/zoo.cfg

tickTime=2000
initLimit=10
syncLimit=5
dataDir=/home/ubuntu/zookeeper/data
dataLogDir=/home/ubuntu/zookeeper/logs
clientPort=2181
autopurge.snapRetainCount=10
autopurge.purgeInterval=24
server.1=<***hostname 1***>*:*2888:3888
server.2=<***hostname 2***>:2888:3888
server.3=<***hostname 3***>:2888:3888
server.4=<***hostname 4***>:2888:3888

Définition des variables d'environnement à mettre dans bin/zkEnv.sh

export ZK_HOME=~/zookeeper/
export PATH=$PATH:$ZK_HOME/bin:$ZK_HOME/conf
export JAVA_HOME= usr/lib/jvm/java-8-openjdk-amd64/

Création du fichier myid : echo <***no de serveur***> > ~/zookeeper/data/myid

Démarrer zookeeper : bin/zkServer.sh start

Connection à zookeeper : bin/zkCli.sh comme client

Lister les znodes : ls /
Créer un znode : create /workers "" ("" signifie qu'on ne veut pas insérer de données)
Supprimer un zonde : delete /orkers
Quitter le client : quit

Vérifier son statut : echo srvr | nc localhost 2181 | grep Mode

Arrêter Zookeeper : bin/zkServer.sh stop

Installation de HBASE

Téléchargement : wget https://dlcdn.apache.org/hbase/2.4.6/hbase-2.4.6-bin.tar.gz

Dans ~/.bashrc :
Export HBASE_HOME=/home/ubuntu/hbase
Export HBASE_CONF=$HBASE_HOME/conf
Export HBASE_WEBAPPS=$HBASE_HOME/hbase-webapps
Export PATH=$PATH :$HBASE_HOME/bin

Dans *conf/hbase-env.sh*, précision du JAVA HOME

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/
export HBASE_REGIONSERVERS=/home/ubuntu/hbase/conf/regionservers

dans conf/hbase-site.xml :

```
<configuration>
        <property>
                <name>hbase.rootdir</name>
                <value>hdfs://<hadoop-master>:9000/hbase</value>
        </property>
        <property>
                <name>hbase.cluster.distributed</name>
                <value>true</value>
        </property>
        <property>
                <name>hbase.zookeeper.quorum</name>
                <value>hostname1, hostname2, hostname3…</value>
        </property>
        <property>
                <name>dfs.replication</name>
                <value>1</value>
        </property>
        <property>
                <name>hbase.zookeeper.property.clientPort</name>
                <value>2181</value>
        </property>
        <property>
                <name>hbase.zookeeper.property.dataDir</name>
                <value>/home/ubuntu/hbase/zookeeper</value>
        </property>
</configuration>
```

Dans conf/regionservers : liste des hostnames des serveurs

Démarrer HBASE : *bin/start-hbase.sh*

Connection à l'interface hbase : ./bin/hbase shell

Arrêt de HBASE : ./bin/stop-hbase.sh

Installation de HIVE

Au besoin, voir à la fin le traitement des erreurs.

Téléchargement, décompression et renommage du dossier : *wget https://dlcdn.apache.org/hive/hive-3.1.2/apache-hive-3.1.2-bin.tar.gz && tar -zxf apache-hive-3.1.2-bin.tar.gz && rm -rf apache-hive-3.1.2-bin.tar.gz && mv apache-hive-3.1.2-bin hive*

Vérification des chemins dans ~/.bashrc :

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/
export PATH=${JAVA_HOME}/bin:${PATH}
export HIVE_HOME=/home/ubuntu/hive
export PATH=$PATH:$HIVE_HOME/bin
export CLASSPATH=$CLASSPATH:/usr/local/Hadoop/lib/*:.
export CLASSPATH=$CLASSPATH:/usr/local/hive/lib/*:.
```

Edition de conf/hive-env.sh :

```
cp hive-env.sh.template hive-env.sh
export HADOOP_HOME=/home/ubuntu/hadoop
```

Edition de bin/hive-config.sh :

```
export HADOOP_HOME=/home/ubuntu/hadoop
```

Initialiser la Derby database en rentrant dans le terminal la commande suivante :
$HIVE_HOME/bin/schematool -dbType derby -initSchema

Installation de Derby (en remplacement de HBase ?) *wget https://dlcdn.apache.org//db/derby/db-derby-10.14.2.0/db-derby-10.14.2.0-bin.tar.gz && tar -zxf db-derby-10.14.2.0-bin.tar.gz && mv db-derby-10.14.2.0-bin derby*

remplissage du .bashrc :

```
export DERBY_HOME=/home/ubuntu/derby
export PATH=$PATH:$DERBY_HOME/bin
export
CLASSPATH=$CLASSPATH:$DERBY_HOME/lib/derby.jar:$DERBY_HOME/lib/derbytools.jar
```

Création d'un dossier data dans derby : *mkdir -p ~/derby/data*

Configurer le métastore de Hive : *cd $HIVE_HOME/conf && cp hive-default.xml.template hive-site.xml && nano template hive-site.xml*

Créer un fichier jpox.properties dans lequel insérer les lignes suivantes : (pas sûr que ce soit utile)

```
javax.jdo.PersistenceManagerFactoryClass =
org.jpox.PersistenceManagerFactoryImpl
org.jpox.autoCreateSchema = false
org.jpox.validateTables = false
org.jpox.validateColumns = false
org.jpox.validateConstraints = false
org.jpox.storeManagerType = rdbms
org.jpox.autoCreateSchema = true
```

org.jpox.autoStartMechanismMode = checked
org.jpox.transactionIsolation = read_committed
javax.jdo.option.DetachAllOnCommit = true
javax.jdo.option.NontransactionalRead = true
javax.jdo.option.ConnectionDriverName = org.apache.derby.jdbc.ClientDriver
javax.jdo.option.ConnectionURL = jdbc:derby://hadoop1:1527/metastore_db;create = true
javax.jdo.option.ConnectionUserName = APP
javax.jdo.option.ConnectionPassword = mine

Lancer hadoop et créer dans hdfs un dossier /tmp et /user/hive/warehouse : *cd ~/hadoop && bin/hadoop fs -mkdir -p /tmp && bin/hadoop fs -mkdir -p /user/hive/warehouse*

Attribuer les droits à ces dossiers : *bin/hadoop fs -chmod g+w /tmp && bin/hadoop fs -chmod g+w /user/hive/warehouse*

Vérifier l'installation de hive : *cd ~/hive && bin/hive*

Test : lancement du terminal hive et création d'une table :

Lancer la commande (n'import où) : *hive*

```
>create table product(product int, pname string, price float)
>row format delimited
>fields terminated by ',';
```

Normalement le terminal renvoie « OK ». Puis taper : *describe product ;*

Message d'erreur java au lancement :

- *SLF4J: Found binding in [jar:file:/home/ubuntu/hive/lib/slf4j-log4j12-1.7.30.jar…* : supprimer le fichier

- *com.ctc.wstx.exc.WstxParsingException: Illegal character entity: expansion character (code 0x8 at [row,col,system-id]: [3215,96,"file:/home/ubuntu/hive/conf/hive-site.xml"] :* supprimer le caractère dans le fichier (rechercher « exclusive locks for… » et supprimer les caractères spéciaux juste après.

- *Exception in thread "main" java.lang.IllegalArgumentException: java.net.URISyntaxException: Relative path in absolute URI: ${system:java.io.tmpdir%7D/$%7Bsystem:user.name%7D :* insérer juste après <configuration> de hive-site.xml les lignes suivantes :

```
<property>
  <name>system:java.io.tmpdir</name>
  <value>/tmp/hive/java</value>
 </property>
 <property>
  <name>system:user.name</name>
  <value>${user.name}</value>
 </property>
```

- *hive > FAILED: HiveException java.lang.RuntimeException: Unable to instantiate org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreClient :* modifier le fichier sql comme suit :
  nano ~/hive/scripts/metastore/upgrade/derby/hive-schema-3.1.0.derby.sql
  commenter les deux premières lignes de définition de

```
#CREATE FUNCTION "APP"."NUCLEUS_ASCII"
#CREATE FUNCTION "APP"."NUCLEUS_MATCHES"
```

  Relancer *$HIVE_HOME/bin/schematool -dbType derby -initSchema*

Installation de SPark

Téléchargement et installation : *wget https://archive.apache.org/dist/spark/spark-3.2.0/spark-3.2.0-bin-hadoop3.2.tgz && rm -rf spark-3.2.0-bin-hadoop3.2.tgz*

Dans .bashrc :

export SPARK_HOME=/home/ubuntu/spark
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin

Créer spark-env.sh dans spark/conf : *cp spark-env.sh.template spark-env.sh*

Tester spark : dans spark, taper *bin/spark-shellre*

Tester spark via url : http://ip-address:4040/

Installer pySpark : s*udo apt install pip && pip install pyspark*

Dans le fichier spark-env.sh :

Ajout du JAVA_HOME**

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

Ajout de l'adresse IP du master**

export SPARK_MASTER='192.168.3.32'

Création du fichier slaves depuis ~/spark/conf/slaves

Ajouter les lignes suivantes au fichier slave
tp-hadoop-9 # master
tp-hadoop-10 # slave01
tp-hadoop-11 # slave02
tp-hadoop-12 # slave03

Lancement de Spark:

~spark/sbin/start-all.sh # pour lancer Spark
~spark/sbin/stop-all.sh # pour arrêter Spark

Ouverture du shell : *bin /spark-shell*

Lancer pySpark : pyspark

Programme Wordcount PySpark**

Creation de ~/spark/WC_spark/WC_pyspark.py
Lancement du programme: bin/spark-submit ./WC_pyspark/WC_pyspark.py
bin/spark-submit ./WC_pyspark/WC_pypsark_file_arg.py ../hadoop/data/file02

WC_pyspark_file_arg.py

```
import sys
from operator import add

from pyspark.sql import SparkSession


if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Usage: wordcount <file>", file=sys.stderr)
        sys.exit(-1)

    spark = SparkSession\
        .builder\
        .appName("PythonWordCount")\
        .getOrCreate()

    lines = spark.read.text(sys.argv[1]).rdd.map(lambda r: r[0])
    counts = lines.flatMap(lambda x: x.split(' ')) \
                .map(lambda x: (x, 1)) \
                .reduceByKey(add)
    output = counts.collect()
    for (word, count) in output:
        print("%s: %i" % (word, count))

    spark.stop()
```