Tutorial linux

# Prérequis :

* Etre sur linux
* Avoir télécharger et dézipper le fichier
* Avoir installer Docker
* Avoir démarrer Docker ( systemctl start docker )

# Compiler L’image Docker:

cd hadoop-cluster-docker

./build-image.sh

# Démarrer L’image:

./start-container.sh

# Lancer hadoop :

./start-hadoop.sh

# Lancer HBase:

./start-hbase.sh

# Lancer Spark :

spark-shell

# Tuto Hadoop (comment l’utiliser):

Après avoir démarrer Hadoop,

On va créer un répertoire input :

**hadoop fs –mkdir -p input**

On va charger le fichier purshase.txt :

**hadoop fs –put purchases.txt input**

On affiche le contenu du répertoire input pour vérifier :

**hadoop fs –ls input**

On va afficher les dernières lignes du fichier avec :

**hadoop fs -tail input/purchases.txt**

Notons que sur <http://localhost:50070> affiche les informations de nôtre namenode

Et que <http://localhost:8088> permet d'afficher les informations du ressource manager de Yarn et visualiser le comportement des différents jobs

Ouvrir le fichier pom.xml et ajouter les dépendances :

**<dependencies>**

**<dependency>**

**<groupId>org.apache.hadoop</groupId>**

**<artifactId>hadoop-common</artifactId>**

**<version>2.7.2</version>**

**</dependency>**

**<!-- https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-mapreduce-client-core -->**

**<dependency>**

**<groupId>org.apache.hadoop</groupId>**

**<artifactId>hadoop-mapreduce-client-core</artifactId>**

**<version>2.7.2</version>**

**</dependency>**

**<!-- https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-hdfs -->**

**<dependency>**

**<groupId>org.apache.hadoop</groupId>**

**<artifactId>hadoop-hdfs</artifactId>**

**<version>2.7.2</version>**

**</dependency>**

**<dependency>**

**<groupId>org.apache.hadoop</groupId>**

**<artifactId>hadoop-mapreduce-client-common</artifactId>**

**<version>2.7.2</version>**

**</dependency>**

**</dependencies>**

**A suivre**

# Tuto Spark:

Une fois avoir lancer les containers et Hadoop,

On crée un fichier file.txt contenant un texte :

**touch file.txt**

**echo hello > file.txt**

On va ensuite chager le fichier dans le HDFS :

**hadoop fs -put file.txt**

On lance le shell spark:

**spark-shell**

Ensuite, pour s’entrainer à spark, on va :

Séparer les mots selon les caractères d'espacement

**val lines = sc.textFile("file1.txt")**

Appliquer un *map* sur les mots obtenus qui produit le couple (*<mot>*, 1)

**val words = lines.flatMap(\_.split("\\s+"))**

Puis un *reduce* qui permet de faire la somme des 1 des mots identiques

**val wc = words.map(w => (w, 1)).reduceByKey(\_ + \_)**

Puis, on sauvegarde le résultat dans un fichier file1.count

**wc.saveAsTextFile("file.count")**

On vas télécharger le fichier file.count dans le HDFS:

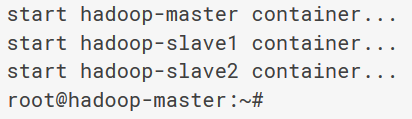
**hadoop fs -get file1.count**

Puis on va regarder le contenu du fichier « part-00000 » crée par le chargement :

**cat part-00000**

# Tuto HBase shell:

Une fois avoir Démarrer l’image avec ./start-container.sh :



On lance Hadoop avec la commande **./start-hadoop.sh**

On lance ensuite HBase avec la commande **start-hbase.sh**

On démarre le shell de HBase en tapant la commande **hbase shell**



On vas crée une table en tapant la commande :

**create ‘table1 ’,’colonne1’,’colonne2’**

En tapant la commande **list** , on vas lister les tables créent.

Pour insérer une ligne dans la table, on tape la commande :

**put ‘table1 ’,’1’,’colonne1:souscolonne1’,’valeur1’**

**put ‘table1 ’,’1’,’colonne1:souscolonne2’,’valeur2’**

**put ‘table1 ’,’1’,’colonne2:souscolonne3’,’valeur3’**

**put ‘table1 ’,’1’,’colonne2:souscolonne4’,’valeur4’**

Cette série de ligne à permis de crée un objet table1 ayant une colonne 1 et une colonne 2 comprenant respectivement une souscolonne1 et une souscolonne2 pour la première colonne et une souscolonne3 et une souscolonne4 pour la deuxième.

Pour afficher la table « table1 », on peut exécuter la commande :

**scan ‘table1 ’**

Pour afficher une ligne en particulier, on peut faire la commande :

**get ‘table1’,'1',{COLUMN => 'colonne1:souscolonne1'}**

On récupèrera donc l’information « valeur1 ».