1. Installation de docker.
2. Vérifier que docker a bien été installé en lançant la commande : docker.
3. Télécharger l’image docker sur dockerHub avec la commande suivant :

docker pull liliasfaxi/spark-hadoop:hv-2.7.2



1. Création de trois contenaires (1 maitre et 2 esclave) avec les commande suivant :

docker run -itd --net=hadoop -p 50070:50070 -p 8088:8088 -p 7077:7077 -p 16010:16010 --name hadoop-master --hostname hadoop-master liliasfaxi /spark-hadoop:hv-2.7.2

docker run -itd -p 8040:8042 --net=hadoop --name hadoop-slave1 --hostname hadoop-slave1 liliasfaxi /spark-hadoop:hv-2.7.2

docker run -itd -p 8041:8042 --net=hadoop --name hadoop-slave1 --hostname hadoop-slave2 liliasfaxi /spark-hadoop:hv-2.7.2

A la fin on a les 3 contenaire crée comme ca :

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

PS : Moi j’ai un problème dans la création de master sur le port 50070, du coup je l’ai supprimée.

1. On va entrai dans le contenaire master et commencé a exécuter des commande :



Le résultat de l’ exécution doit être comme cela : root@hadoop-master:~#

1. Après avoir entré dans le contenaire master , la prochaine étape et de lancé hadoop et yarn avec la commande suivant :

./start-hadoop.sh

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

1. Après avoir lancé le contenaire master, on va créer un répertoire input ou on va stocker notre fichier d’entrée et de déplacer le fichier purchases.txt avec cette commande :

hadoop fs –put purchases.txt input

On peut voir le fichier a été bien déplacé on affichant les dernier ligne de notre fichier comme cela :

Une image contenant texte, capture d’écran, menu, noir et blanc

Description générée automatiquement

* les commandes les plus utilisées pour manipuler les fichiers dans HDFS:

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Map Reduce[¶](https://insatunisia.github.io/TP-BigData/tp1/" \l "map-reduce" \o "Permanent link)

**Présentation**[**¶**](https://insatunisia.github.io/TP-BigData/tp1/#presentation_1)

Un Job Map-Reduce se compose principalement de deux types de programmes :

* **Mappers** : permettent d’extraire les données nécessaires sous forme de clef/valeur, pour pouvoir ensuite les trier selon la clef
* **Reducers** : prennent un ensemble de données triées selon leur clef, et effectuent le traitement nécessaire sur ces données (somme, moyenne, total...)

**Wordcount[¶](https://insatunisia.github.io/TP-BigData/tp1/" \l "wordcount" \o "Permanent link)**

Nous allons tester un programme MapReduce grâce à un exemple très simple, le *WordCount*, l'équivalent du *HelloWorld* pour les applications de traitement de données. Le Wordcount permet de calculer le nombre de mots dans un fichier donné, en décomposant le calcul en deux étapes:

* L'étape de *Mapping*, qui permet de découper le texte en mots et de délivrer en sortie un flux textuel, où chaque ligne contient le mot trouvé, suivi de la valeur 1 (pour dire que le mot a été trouvé une fois)
* L'étape de *Reducing*, qui permet de faire la somme des 1 pour chaque mot, pour trouver le nombre total d'occurrences de ce mot dans le texte.