

Université Virtuelle du Sénégal (UVS)



Master 1 **Big Data & Analytics**

Unité d'enseignement **Infrastructures Big data**

Module: **Virtualisation/Cloud/Plateforme Big data** | Instructeur : M. TINE

**Année académique : 2020-2021**

---

## **Activité 3**

### **INITIATION A HADOOP ET MAPREDUCE**

---

Instructeur: Mr. TINE Ing., MS., MSc.

Objectifs de l'activité:

- Ce document décrit comment installer et configurer une installation Hadoop à un seul nœud afin que vous puissiez effectuer rapidement des opérations simples à l'aide de Hadoop MapReduce et du système de fichiers distribués Hadoop (HDFS).

# I. Hadoop

## I.1 Présentation

Apache Hadoop ([hadoop.apache.org](https://hadoop.apache.org)) est un Framework open-source pour stocker et traiter les données volumineuses sur un cluster. Il est utilisé par un grand nombre de contributeurs et utilisateurs. Il a une licence Apache 2.0.



Figure 1 : Hadoop

## I.2 Installation

Ce TP est une adaptation d'un TP proposé par Institut National des Sciences Appliquées et de Technologie Tunisie, lui-même inspiré de la formation "**Intro to Hadoop and Map Reduce**" fait par Cloudera<sup>1</sup> et publié sur Udacity<sup>2</sup>. Cloudera fournit une machine virtuelle où Hadoop, ainsi qu'un grand nombre d'outils de son écosystème, sont préinstallés. Cependant nous allons installer la dernière distribution d'Hadoop pour effectuer cette activité.

L'installation d'Hadoop en mode **Single Node Cluster**.

### a) Prérequis :

Nous allons travailler en environnement Linux (**Ubuntu** de préférence ou CentOS)

Java doit être installé dans le système (vous pouvez vérifier en tapant la commande **java -version**).

Si Java n'est pas installé, vous pouvez le faire avec la commande suivante :

```
$ sudo apt-get update
$ sudo apt-get install openjdk-8-jdk
```

Vérifier l'installation de Java avec la commande **java -version**, le résultat devrait ressembler à ceci :

```
openjdk version "1.8.0_242"
OpenJDK Runtime Environment (build 1.8.0_242-b09)
OpenJDK 64-Bit Server VM (build 25.242-b09, mixed mode)
```

<sup>1</sup> **Cloudera** : Plateforme de BigData <https://www.cloudera.com/>

<sup>2</sup> **Udacity** : Plateforme de eLearning <https://www.udacity.com/>

Si votre machine ne dispose pas du logiciel ssh requis, vous devrez l'installer avec les commandes suivantes :

```
$ sudo apt-get install ssh
$ sudo apt-get install pdsh
```

## b) Téléchargement de la distribution d'Hadoop

Pour obtenir une distribution Hadoop, téléchargez une version stable récente à partir de l'un des miroirs de téléchargement Apache : <http://www.apache.org/dyn/closer.cgi/hadoop/common/>

Prendre la version hadoop-3.2.2 <https://downloads.apache.org/hadoop/common/hadoop-3.2.2/>

# Index of /hadoop/common/hadoop-3.2.2

Name	Last modified	Size	Description
 <a href="#">Parent Directory</a>		-	
 <a href="#">CHANGELOG.md</a>	2021-01-13 18:48	95K	
 <a href="#">CHANGELOG.md.asc</a>	2021-01-13 18:48	833	
 <a href="#">CHANGELOG.md.sha512</a>	2021-01-13 18:48	143	
 <a href="#">RELEASENOTES.md</a>	2021-01-13 18:48	5.2K	
 <a href="#">RELEASENOTES.md.asc</a>	2021-01-13 18:48	833	
 <a href="#">RELEASENOTES.md.sha512</a>	2021-01-13 18:48	146	
 <a href="#">hadoop-3.2.2-rat.txt</a>	2021-01-13 18:48	1.8M	
 <a href="#">hadoop-3.2.2-rat.txt.asc</a>	2021-01-13 18:48	833	
 <a href="#">hadoop-3.2.2-rat.txt.sha512</a>	2021-01-13 18:48	151	
 <a href="#">hadoop-3.2.2-site.tar.gz</a>	2021-01-13 18:48	43M	
 <a href="#">hadoop-3.2.2-site.tar.gz.asc</a>	2021-01-13 18:48	833	
 <a href="#">hadoop-3.2.2-site.tar.gz.sha512</a>	2021-01-13 18:48	155	
 <a href="#">hadoop-3.2.2-src.tar.gz</a> 	2021-01-13 18:48	31M	
 <a href="#">hadoop-3.2.2-src.tar.gz.asc</a>	2021-01-13 18:48	833	
 <a href="#">hadoop-3.2.2-src.tar.gz.sha512</a>	2021-01-13 18:48	154	
 <a href="#">hadoop-3.2.2.tar.gz</a>	2021-01-13 18:48	377M	
 <a href="#">hadoop-3.2.2.tar.gz.asc</a>	2021-01-13 18:48	833	
 <a href="#">hadoop-3.2.2.tar.gz.sha512</a>	2021-01-13 18:48	150	

**c) Préparation de l'environnement pour le démarrage du cluster Hadoop**

Décompressez la distribution Hadoop téléchargée. Dans la distribution, éditez le fichier **etc/hadoop/hadoop-env.sh** pour définir certains paramètres comme suit:

```
# set to the root of your Java installation
export JAVA_HOME=/usr/java/latest
```

Dans le cas d'une distribution Ubuntu le répertoire d'installation de Java est **/usr/lib/jvm/java-8-openjdk-amd64**

La modification dans le fichier **etc/hadoop/hadoop-env.sh** ressemble à ceci :

```
# set to the root of your Java installation
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

Pour une modification permanente, modifier le fichier **.bashrc** avec les entrées ci-dessous

Dans mon cas je suppose que la distribution Hadoop a été décompressé dans mon répertoire personnel /home/pmtine, utiliser l'éditeur **\$ nano ~/.bashrc**

```
#Java variables
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export PATH=${JAVA_HOME}/bin:${PATH}
export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar
# Hadoop variables
export HADOOP_HOME=/home/pmtine/hadoop-3.2.2
export PATH=${PATH}:${HADOOP_HOME}/bin
export PATH=${PATH}:${HADOOP_HOME}/sbin
```

## I.3 Premiers Pas avec Hadoop

### Tester l'installation d'Hadoop

#### a) Fonctionnement autonome (Standalone Operation)

Par défaut, Hadoop est configuré pour s'exécuter en mode non distribué, en tant que processus Java unique. Ceci est utile pour le débogage.

L'exemple suivant copie le répertoire conf décompressé à utiliser comme entrée, puis recherche et affiche chaque correspondance de l'expression régulière donnée. La sortie est écrite dans le répertoire de sortie donné.

```
$ mkdir input
$ cp etc/hadoop/*.xml input
$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.2.jar grep input output 'dfs[a-z.]+'
$ cat output/*
```

#### b) Fonctionnement pseudo-distribuée (Pseudo-Distributed Operation)

Hadoop peut également être exécuté sur un nœud unique dans un mode pseudo-distribué où chaque démon Hadoop s'exécute dans un processus Java distinct.

### Configuration

Editer le fichier **etc/hadoop/core-site.xml** comme suit :

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

Puis le fichier **etc/hadoop/hdfs-site.xml** comme suit :

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
</configuration>
```

## ✚ Configurer SSH sans mot de passe

Maintenant, vérifiez que vous pouvez ssh sur l'hôte local sans mot de passe:

```
$ ssh localhost
```

Si vous ne pouvez pas ssh vers localhost sans phrase de passe, exécutez les commandes suivantes:

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

## ✚ Exécution

Suivez les étapes suivantes pour tester le mode d'opération pseudo-distribué. Les instructions suivantes permettent d'exécuter un travail **MapReduce localement**.

### 1. Formatage du système de fichiers pour prise en compte de HDFS

```
$ hdfs namenode -format
```

### 2. Démarrage des démons NameNode et DataNode :

```
$ start-dfs.sh
```

La sortie du journal du démon hadoop est écrite dans le répertoire \$ HADOOP\_LOG\_DIR (par défaut, \$ HADOOP\_HOME / logs).

### 3. Parcourez l'interface Web pour le NameNode; par défaut, il est disponible sur:

```
http://localhost:9870/
```

Vous devriez avoir ce type d'interface :

**Hadoop** Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities ▾

## Overview 'localhost:9000' (active)

<b>Started:</b>	Mon May 24 12:30:11 +0000 2021
<b>Version:</b>	3.2.2, r7a3bc90b05f257c8ace2f76d74264906f0f7a932
<b>Compiled:</b>	Sun Jan 03 09:26:00 +0000 2021 by hexiaoqiao from branch-3.2.2
<b>Cluster ID:</b>	CID-0b9ef5b0-ca57-4a24-a48c-9e8de17ddfe4
<b>Block Pool ID:</b>	BP-146139384-10.0.0.4-1621859398849

## Summary

Security is off.  
Safemode is off.

1 files and directories, 0 blocks (0 replicated blocks, 0 erasure coded block groups) = 1 total filesystem object(s).

Heap Memory used 97.05 MB of 300 MB Heap Memory. Max Heap Memory is 1.73 GB.

**4. Créez les répertoires HDFS requis pour exécuter les tâches MapReduce:**

```
$ hdfs dfs -mkdir /user
$ hdfs dfs -mkdir /user/<username>
```

Dans mon cas username=**pmtine**

**5. Copiez les fichiers d'entrée dans le système de fichiers distribué:**

```
$ hdfs dfs -mkdir input
$ hdfs dfs -put etc/hadoop/*.xml input
```

**6. Exécutez certains des exemples fournis avec l'installation**

```
$ hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.2.jar grep input output 'dfs[a-z.]+'
```

**7. Examinez les fichiers de sortie: copiez les fichiers de sortie du système de fichiers distribué vers le système de fichiers local et examinez-les:**

```
$ hdfs dfs -get output output
$ cat output/*
```

Ou avec la commande suivante

```
$ hdfs dfs -cat output/*
```

**8. Lorsque vous avez terminé, arrêtez les démons (processus) avec la commande**

```
$ stop-dfs.sh
```

**c) Configuration de YARN sur un seul nœud**

Vous pouvez exécuter un travail MapReduce sur YARN dans un mode **pseudo-distribué** en définissant quelques paramètres et en exécutant en plus le démon ResourceManager et le démon NodeManager.

Les instructions suivantes ne supposent que les étapes 1. à 4. des instructions ci-dessus sont déjà exécutées.

**1. Configurez les paramètres comme suit:**

**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>
```

etc/hadoop/yarn-site.xml:

```
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.env-whitelist</name>

    <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF
    _DIR,CLASSPATH_PREPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_H
    OME</value>
  </property>
</configuration>
```

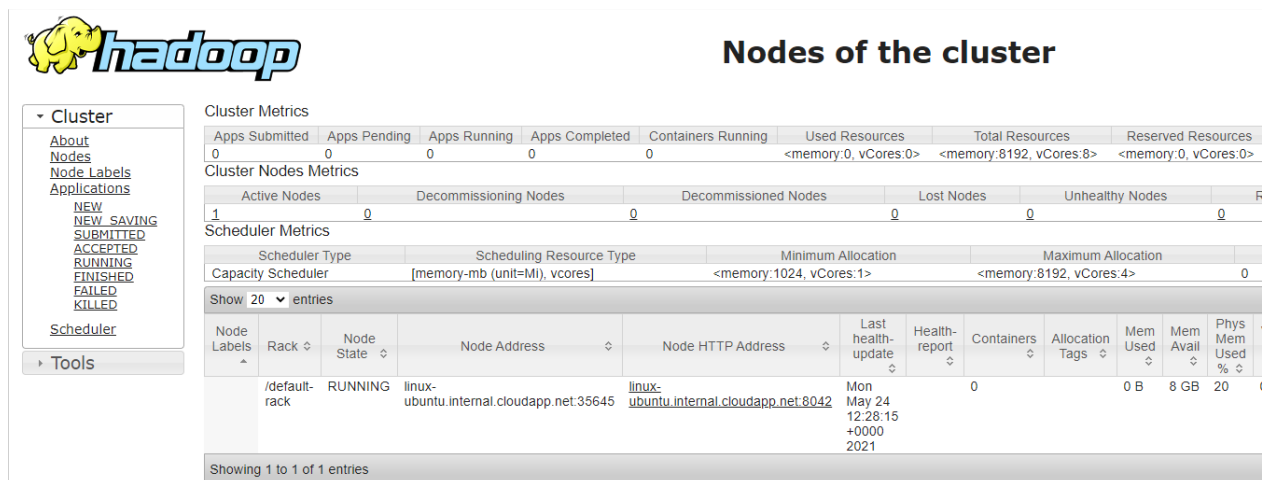
## 2. Démarrez le démon ResourceManager et le démon NodeManager:

```
$ start-yarn.sh
```

## 3. Parcourez l'interface Web du ResourceManager; par défaut, il est disponible sur:

Lien de ResourceManager : <http://localhost:8088/>

Vous accédez à une interface suivante, naviguez sur les différents liens dans le menu à gauche.



The screenshot shows the Hadoop ResourceManager web interface. On the left is a navigation menu with options like Cluster, About, Nodes, Node Labels, Applications, and Scheduler. The main content area is titled "Nodes of the cluster" and displays various metrics and a table of nodes.

**Cluster Metrics**

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resources	Total Resources	Reserved Resources
0	0	0	0	0	<memory:0, vCores:0>	<memory:8192, vCores:8>	<memory:0, vCores:0>

**Cluster Nodes Metrics**

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes
1	0	0	0	0

**Scheduler Metrics**

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>

Showing 20 entries

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Allocation Tags	Mem Used	Mem Avail	Phys Mem Used %
/default-rack		RUNNING	linux-ubuntu.internal.cloudapp.net:35645	linux-ubuntu.internal.cloudapp.net:8042	Mon May 24 12:28:15 +0000 2021		0		0 B	8 GB	20 %

Showing 1 to 1 of 1 entries

## 4. Exécutez un travail MapReduce.

## 5. Lorsque vous avez terminé, arrêtez les démons avec:

```
$ stop-yarn.sh
```



## I.4 Manipulation des données de l'étude de cas : analyse de ventes

### a) Organisation du répertoire de travail

Dans cette activité nous n'avons pas utilisé la machine virtuelle préconfigurée avec Hadoop et les différents répertoires de travail. Il faudra donc organiser le répertoire de travail, pour faire simple et suivre le tutoriel d'Udacity, vous pouvez créer un répertoire `udacity_training` dans votre repertoire personnel (`/home/pmtine` dans mon cas) :

```
$ mkdir ~/udacity_training
```

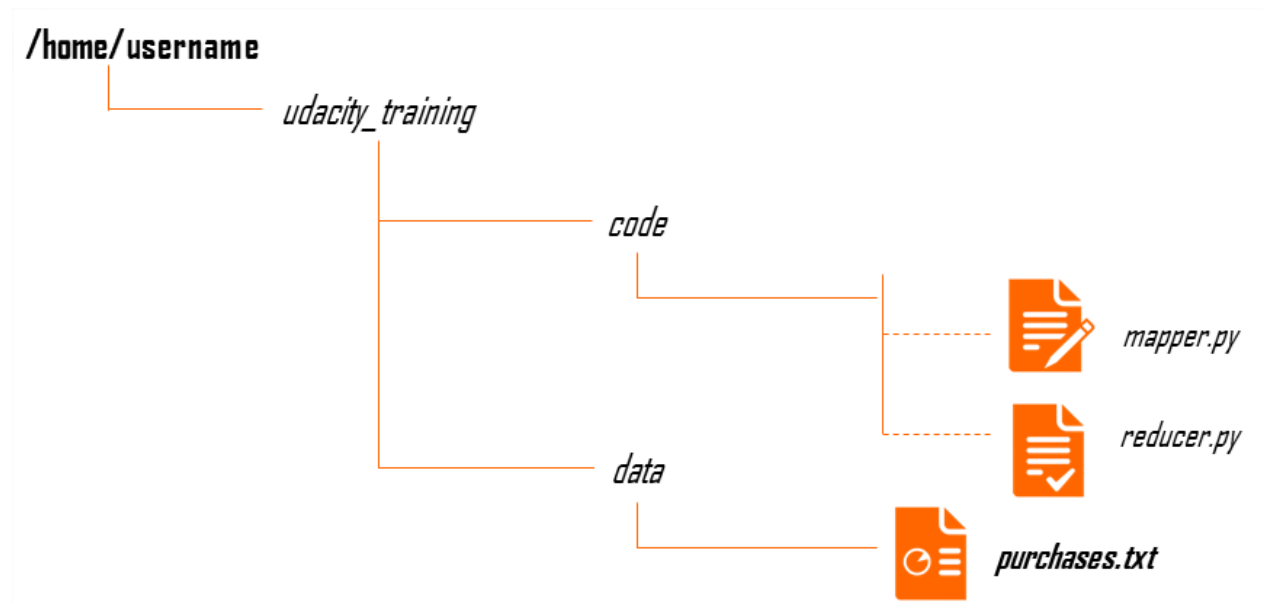
Vous allez créer deux sous-répertoires: `code` et `data` dans lesquels on trouvera et on sauvegardera respectivement les codes de nos mappers et reducers, et les données sources (`purchases.txt`) et résultat.

```
$ mkdir ~/udacity_training/data
```

```
$ mkdir ~/udacity_training/code
```

Ensuite télécharger le fichier de données de travail `purchases.txt`, partagé avec le responsable de classe.

A l'arrivé le répertoire de travail devrait ressembler à ceci (username=nom d'utilisateur Unix).



Dans mon cas username=pmtine

Toutes les commandes interagissant avec le système Hadoop/HDFS commencent par **hdfs dfs**. Ensuite, les options rajoutées sont très largement inspirées des commandes Unix standard.

- Créez les répertoires HDFS requis pour exécuter les tâches MapReduce:

```
$ hdfs dfs -mkdir /user
$ hdfs dfs -mkdir /user/<username>
```

Dans mon cas le <username>=pmtine

```
$ hdfs dfs -mkdir /user
$ hdfs dfs -mkdir /user/pmtine
```

- Créer un répertoire dans HDFS, appelé **myinput**. Pour cela, taper:

**hdfs dfs -mkdir myinput**

- Pour copier le fichier *purchases.txt* dans HDFS sous le répertoire **myinput**, taper la commande:

**hdfs dfs -put data/purchases.txt myinput/**

- Pour afficher le contenu du répertoire *myinput*, la commande est:

**hdfs dfs -ls myinput**

On obtiendra alors le résultat suivant:

```
pmtine@Linux-Ubuntu:~/udacity_training$ hdfs dfs -ls myinput
Found 1 items
-rw-r--r--  1 pmtine supergroup  211312924 2021-05-24 13:47 myinput/purchases.txt
```

Pour visualiser les dernières lignes du fichier, taper:

**hdfs dfs -tail myinput/purchases.txt**

On obtient alors:

```
pmtine@Linux-Ubuntu:~/udacity_training$ hdfs dfs -tail myinput/purchases.txt
31      17:59  Norfolk Toys  164.34  MasterCard
2012-12-31  17:59  Chula Vista  Music  380.67  Visa
2012-12-31  17:59  Hialeah Toys  115.21  MasterCard
2012-12-31  17:59  Indianapolis  Men's Clothing  158.28  MasterCard
2012-12-31  17:59  Norfolk Garden  414.09  MasterCard
2012-12-31  17:59  Baltimore  DVDs  467.3  Visa
2012-12-31  17:59  Santa Ana  Video Games  144.73  Visa
2012-12-31  17:59  Gilbert Consumer Electronics  354.66  Discover
2012-12-31  17:59  Memphis Sporting Goods  124.79  Amex
2012-12-31  17:59  Chicago Men's Clothing  386.54  MasterCard
2012-12-31  17:59  Birmingham  CDs  118.04  Cash
2012-12-31  17:59  Las Vegas  Health and Beauty  420.46  Amex
2012-12-31  17:59  Wichita Toys  383.9  Cash
2012-12-31  17:59  Tucson Pet Supplies  268.39  MasterCard
2012-12-31  17:59  Glendale  Women's Clothing  68.05  Amex
2012-12-31  17:59  Albuquerque  Toys  345.7  MasterCard
2012-12-31  17:59  Rochester  DVDs  399.57  Amex
2012-12-31  17:59  Greensboro  Baby  277.27  Discover
2012-12-31  17:59  Arlington  Women's Clothing  134.95  MasterCard
2012-12-31  17:59  Corpus Christi  DVDs  441.61  Discover
```

Le CLI hdfs fournit différents types de commandes de type Shell qui interagissent directement avec les données HDFS. Vous pouvez lire ou écrire des données de fichier avec l'outil shell. En outre, vous pouvez accéder aux données stockées dans d'autres systèmes de stockage tels que HFTP, S3 et FS que HDFS prend désormais en charge.

Dans le tableau suivant, nous résumons les commandes les plus utilisées avec Hadoop/HDFS depuis un terminal (**hdfs dfs -help** permet d'avoir de l'aide sur la signification des commandes) :

Commande	Description
<b>hdfs dfs -ls</b>	Afficher le contenu du répertoire racine
<b>hdfs dfs -put file.txt</b>	Upload un fichier dans hadoop (à partir du répertoire courant linux)
<b>hdfs dfs -get file.txt</b>	Download un fichier à partir de hadoop sur votre disque local
<b>hdfs dfs -tail file.txt</b>	Lire les dernières lignes du fichier
<b>hdfs dfs -cat file.txt</b>	Affiche tout le contenu du fichier
<b>hdfs dfs -mv file.txt newfile.txt</b>	Renommer le fichier
<b>hdfs dfs -rm newfile.txt</b>	Supprimer le fichier
<b>hdfs dfs -mkdir myinput</b>	Créer un répertoire
<b>hdfs dfs -cat file.txt   less</b>	Lire le fichier page par page

**Activité 1.** Tester les différentes fonctions citées ci-dessus pour:

- Créer un répertoire appelé *myinput*
- Copier le fichier *purchases.txt* dans le répertoire *myinput*
- Afficher les dernières lignes du fichier

## II. Map Reduce

### II.1 Présentation

Map Reduce est un patron d'architecture de développement permettant de traiter les données volumineuses de manière parallèle et distribuée.

Il se compose principalement de deux types de programmes:

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

Pour notre activité, nous utilisons le langage Python pour développer les Mappers et les Reducers. Les traitements intermédiaires (comme le tri par exemple) sont effectués automatiquement par Hadoop.

## II.2 Mapper

Soit un code comportant 6 champs, séparés par des tabulations. Le Mapper doit:

- Séparer les différents champs par tabulation
- Extraire les éléments voulus à partir de ces champs, sous forme de clef/valeur

Pour ce premier exercice, notre but est de déterminer le total des ventes par magasin, pour un fichier log dont les champs sont de la forme suivante :

**date → temps → magasin → produit → coût → paiement**

Pour calculer les ventes par magasin, le couple (clef, valeur) à extraire est (magasin,coût).

Pour faire cela, le code du Mapper est le suivant :

```
#!/usr/bin/python3

# Format of each line is:
# date\ttime\tstore name\titem description\tcost\tmethod of payment
#
# We want elements 2 (store name) and 4 (cost)
# We need to write them out to standard output, separated by a tab

import sys

for line in sys.stdin:
    data = line.strip().split("\t")
    if len(data) == 6:
        date, time, store, item, cost, payment = data
        print("{0}\t{1}".format(store, cost))
```

Ce code se trouve sous le répertoire `~/udacity_training/code` dans le fichier *mapper.py*.

*Remarque : Python est un langage qui délimite les différents blocs en utilisant les tabulations, faites alors bien attention à vos indentations.*

### Activité 2.

- Que permet de faire chaque ligne de ce code?
- Tester ce mapper en local sur les 50 premières lignes du fichier *purchases.txt* en tapant l'instruction suivante, directement à partir de votre répertoire *code*:

```
head -50 ../data/purchases.txt | mapper.py
```

## II.3 Reducer

Le Reducer permet de faire le traitement désiré sur des entrées sous forme de clef/valeur, préalablement triées par Hadoop (on n'a pas à s'occuper du tri manuellement).

Dans l'exemple précédent, une fois que le Mapper extrait les couples **(store,cost)**, le Reducer aura comme tâche de faire la somme de tous les couts pour un même magasin. Le code du Reducer est le suivant :

```
#!/usr/bin/python3

import sys

salesValue = 0
salesCount = 0

# Loop around the data
# It will be in the format key\tval
# Where key is the store name, val is the sale amount
#
# All the sales for a particular store will be presented,
# then the key will change and we'll be dealing with the next store

for line in sys.stdin:
    data_mapped = line.strip().split("\t")
    if len(data_mapped) != 2:
        # Something has gone wrong. Skip this line.
        continue

    thisStore, thisSale = data_mapped
    salesValue += float(thisSale)
    salesCount += 1

    print (salesValue, "\t", salesCount)
```

### Activité 3.

- Expliquer ce code.
- Tester ce Reducer sur le disque local, en utilisant cette instruction.

```
head -50 ../data/purchases.txt |./mapper.py |sort |./reducer.py
```

## II.4 Lancer un Job entier

Lancer un job entier sur Hadoop implique qu'on fera appel au mapper puis au reducer sur une entrée volumineuse, et obtenir à la fin un résultat, directement sur HDFS. Pour faire cela, l'instruction à exécuter est:

```
$ mapred streaming -mapper mapper.py -reducer reducer.py -file mapper.py -file reducer.py -input myinput -output joboutput
```

Vous pouvez vérifier le travail dans l'interface du Ressource Manager, lien « RUNNING »



**RUNNING**

Cluster

About

Nodes

Node Labels

Applications

NEW

NEW SAVING

SUBMITTED

ACCEPTED

RUNNING

FINISHED

FAILED

KILLED

Scheduler

Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Used Resource
1	0	1	0	1	<memory:2048, vCores:1>

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes
1	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>

Show 20 entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State
application_1621865236902_0002	pmtine	streamjob1407778825539111060.jar	MAPREDUCE	default	0	Mon May 24 14:08:31 +0000 2021	Mon May 24 14:08:33 +0000 2021	N/A	RL

Showing 1 to 1 of 1 entries

Cette instruction donne en paramètres les fichiers correspondant aux Mappers et Reducers, et les répertoires contenant le fichier d'entrée (*myinput*) et la sortie à générer (*joboutput*). Le répertoire de sortie, après exécution, contiendra un fichier appelé *part-00000*, représentant la sortie désirée.

Faite **\$ hdfs dfs -get joboutput output** et **\$ cat output/\*** (ou **hdfs dfs -cat joboutput/\***) pour afficher le résultat du Job. Le résultat doit ressembler à ceci avec

```
pmtine@Linux-Ubuntu:~/udacity_training/code$ hdfs dfs -get joboutput
pmtine@Linux-Ubuntu:~/udacity_training/code$ cat output/*
1034457953.2599708      4138476
```

*Remarque* : Le répertoire d'entrée doit contenir un seul fichier. Le répertoire de sortie ne doit pas exister avant l'exécution de l'instruction.

---

#### Activité 4.

- Exécuter un job hadoop sur le fichier *purchases.txt* en utilisant les fichiers *mapper.py* et *reducer.py* déjà fournis. Stocker le résultat dans un répertoire *joboutput*. Sauvegarde ensuite le fichier part-00000 dans votre répertoire local.
  - Quelle est la totalité des ventes du magasin de **Buffalo** ?
- 

## II.5 Applications : à vous de jouer !

Nous allons continuer à travailler avec le même fichier en entrées (**purchases.txt**), mais pour obtenir des résultats différents. **Le but est donc d'écrire vos propres Mappers et Reducers.**

---

#### Activité 5.

- Donner la liste des ventes par catégorie de produits.
  - Quelle est la valeur des ventes pour la catégorie **Toys**?
  - Et pour la catégorie **Consumer Electronics**?
- 

---

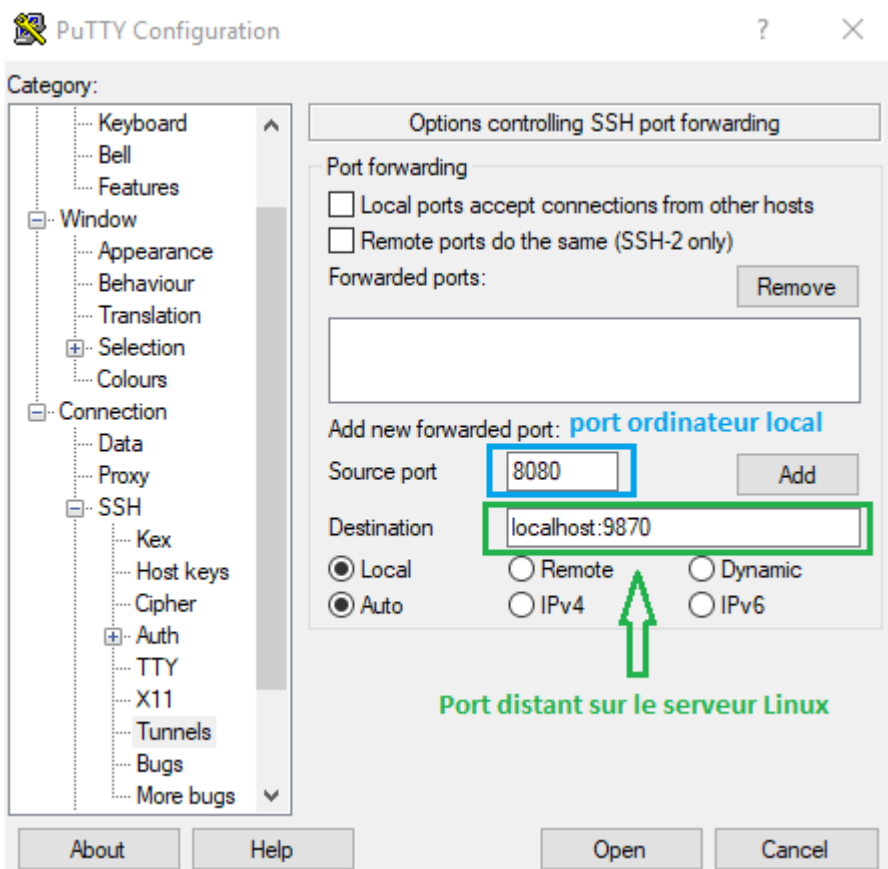
#### Activité 6.

- Donner le montant de la vente le plus élevé pour chaque magasin
  - Quelle cette valeur pour les magasins suivants:
    - o *Reno*
    - o *Toledo*
    - o *Chandler*
-

### III. Annexes

#### III.1 Configuration tunnels SSH

Au cas où votre VM est hébergée dans le Cloud ou ne dispose pas d'interface graphique (Desktop), il est possible de configurer des tunnels pour le forward des URL distants vers la machine locale. La configuration se fera comme suit pour les NameNodes avec l'adresse localhost:9870

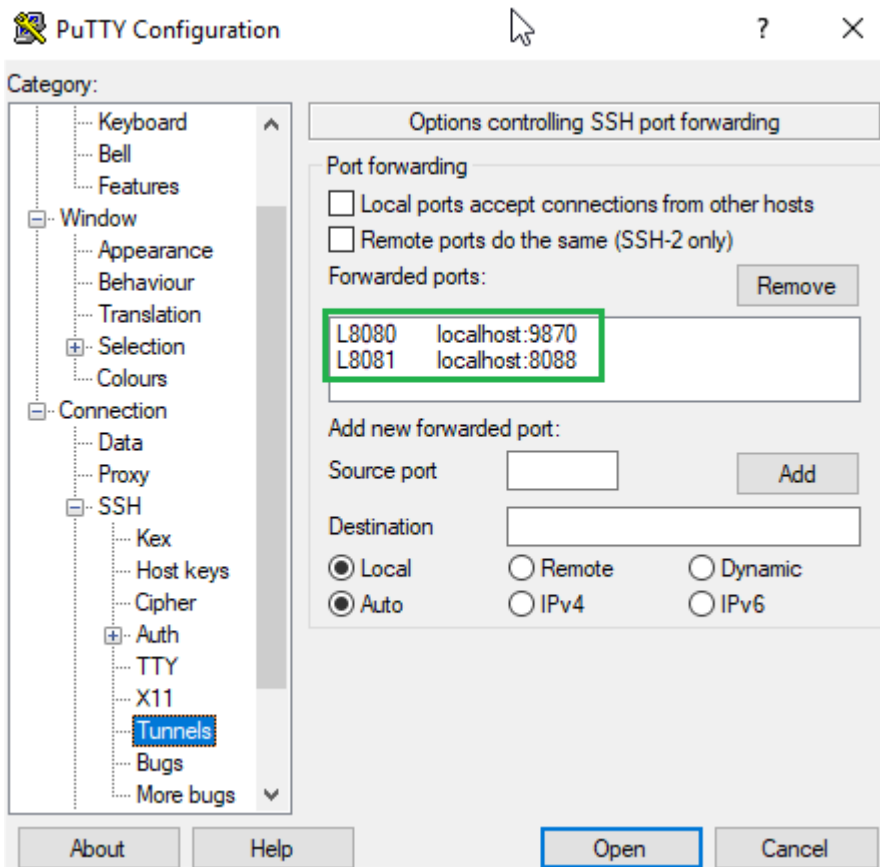


Ainsi on après avoir ouvert la session SSH, l'interface d'administration du **NameNode** est accessible via le lien <http://localhost:8080/>.



Pour le **Ressource Manager** (YARN), c'est le même type de configuration, nous choisiront comme port local 8081 avec accès via le lien <http://localhost:8081>

Au final nous devront avoir 2 tunnels ajoutés dans la session Putty.



## IV. Références

Références pour l'installation d'Hadoop (version 3.3) et tutorial MapReduce :

<https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html>

<https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html>

<https://hadoop.apache.org/docs/current/hadoop-streaming/HadoopStreaming.html>