INSTALLATION DE HADOOP ET SPARK SOUS WSL ET TEST.

Objectif du TP :

Installer Hadoop et Spark sous Windows Subsystem for Linux (WSL) et exécuter un test de fonctionnement.

Prérequis :

- WSL installé sur Windows.
- Ubuntu ou une autre distribution Linux sous WSL.
- Java JDK installé (version 8 ou supérieure).

1- INSTALLATION DES PREREQUIS

- **Installation de WSL sous Windows**

WSL (Windows Subsystem for Linux) en français sous-système Windows pour Linux. Bien qu'il soit intégré dans les systèmes d'exploitation Windows 10 et Windows 11, la fonctionnalité est désactivée par défaut. Pour l'utiliser, vous devez l'activer manuellement.

Étapes pour activer WSL :

1. Ouvrir PowerShell en tant qu'administrateur :

Cliquez avec le bouton droit sur le menu Démarrer et sélectionnez "Windows PowerShell (Admin)".

2. Activer WSL :

Exécutez la commande suivante pour activer WSL :

```
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
```

3. Mettre à jour WSL :

Exécutez la commande : `wsl --update`

La commande "`wsl --update`" est utilisée pour mettre à jour les composants de WSL. Elle est plus pertinente si vous avez déjà WSL installé et que vous souhaitez vous assurer que vous disposez de la dernière version de WSL.

4. Vérifier la virtualisation :

Pour que WSL 2 fonctionne, la virtualisation doit être activée dans le BIOS. Vous pouvez vérifier cela dans le Gestionnaire des tâches :

- Appuyez sur Ctrl + Shift + Esc pour ouvrir le Gestionnaire des tâches.
- Allez dans l'onglet "Performances" et sélectionnez "Processeur". En bas à droite, vous verrez une mention indiquant si la virtualisation est activée.

5. Activer la plateforme de machine virtuelle (pour WSL 2) :

Cette étape est nécessaire pour faire fonctionner WSL 2 :

- Exécutez cette commande :

```
dism.exe /online /enable-feature
/featurename:VirtualMachinePlatform /all /norestart
```

Il peut arriver que vous vous posiez la question de savoir quelle est la raison d'activer la Plateforme de Machine Virtuelle.

Voici les raisons :

1. **Fonctionnalité Spécifique à WSL 2** :
    - La plateforme de machine virtuelle (Virtual Machine Platform) est une fonctionnalité spécifique requise par WSL 2 pour fonctionner correctement. Bien que la virtualisation soit activée dans le BIOS, cela ne suffit pas à activer les fonctionnalités de virtualisation nécessaires pour WSL 2. Cette plateforme permet à WSL 2 d'utiliser Hyper-V, qui gère les machines virtuelles sous Windows.
2. **Isolation et Gestion des Ressources** :
    - L'activation de cette fonctionnalité permet à WSL 2 d'exécuter un noyau Linux complet dans un environnement virtualisé. Cela offre une meilleure isolation et une gestion plus efficace des ressources entre Windows et les distributions Linux. Sans cette activation, WSL 2 ne peut pas fonctionner en tant que machine virtuelle légère.
3. **Compatibilité avec les Applications** :
    - Certaines applications Linux nécessitent des fonctionnalités spécifiques du noyau que WSL 2 peut fournir uniquement si la plateforme de machine virtuelle est activée. Cela améliore la compatibilité avec divers outils et logiciels qui pourraient ne pas fonctionner correctement sous WSL 1.
4. **Amélioration des Performances** :
    - L'utilisation de la plateforme de machine virtuelle permet d'optimiser les performances, notamment pour les opérations d'E/S (entrées/sorties) et l'exécution de programmes nécessitant des appels système spécifiques.

Conclusion

En résumé, même si la virtualisation est activée dans le BIOS, l'activation de la plateforme de machine virtuelle est indispensable pour que WSL 2 fonctionne correctement. Cela permet d'exploiter pleinement les avantages offerts par WSL 2, notamment en termes de performances, de compatibilité et d'accès aux fonctionnalités avancées du noyau Linux.

6. Redémarrer votre ordinateur :

Après avoir exécuté ces commandes, il est généralement nécessaire de redémarrer votre ordinateur pour que les modifications prennent effet.

7. Définir WSL 2 comme version par défaut :

Une fois votre système redémarré, ouvrez à nouveau PowerShell en tant qu'administrateur et exécutez la commande suivante pour définir WSL 2 comme version par défaut :

```
wsl --set-default-version 2
```

- **Installation de Ubuntu ou autre distribution linux sous WSL**

Si vous voulez voir quelles distributions peuvent être installées, exécutez :

```
wsl --list --online
```

Où

```
wsl -l -o
```

Pour ma part j'ai installé la version de Ubuntu 22.04 LTS. Parce qu'après recherche, il est dit que cette version de Ubuntu est la plus stable.

Toujours dans PowerShell, tapez la commande suivante pour installer spécifiquement Ubuntu 22.04 LTS :

```
wsl --install -d Ubuntu-22.04
```

Cette commande va automatiquement télécharger et installer Ubuntu 22.04 LTS depuis le Microsoft Store.

Pour vérifier que l'installation s'est bien déroulée et que tu utilises bien Ubuntu 22.04 avec WSL 2, exécute la commande suivante :

```
wsl --list --verbose
```

Cette commande devrait afficher "Ubuntu-22.04" dans la liste avec la version WSL 2.

Et voilà ! Tu as maintenant Ubuntu 22.04 LTS installé dans WSL et prêt à l'emploi.

NB : Après avoir activé WSL et installé Ubuntu 22.04 LTS, chaque fois que vous ouvrez WSL, vous accédez directement à un terminal Ubuntu fonctionnant comme un environnement Linux intégré à Windows. Cet environnement vous permet d'utiliser toutes les fonctionnalités d'Ubuntu depuis votre système Windows.

- **Installation de Java JDK (version 8 ou supérieure).**

Avant d'installer Java, mets à jour la liste des paquets pour t'assurer d'avoir les dernières versions :

```
sudo apt update
```

Pour installer une version spécifique de Java, utilise la commande suivante :

```
sudo apt install openjdk-8-jdk
```

Une fois l'installation terminée, vérifie la version de Java pour t'assurer que tout est correctement installé :

```
java -version
```

Et voilà ! Java JDK est maintenant installé sur ton Ubuntu dans WSL. Tu peux désormais compiler et exécuter des programmes Java.

2- INSTALLATION DE HADOOP

- **Téléchargement d'Hadoop**

Pour le téléchargement de Hadoop nous pouvons utiliser deux méthodes :

1ère Méthode : Sous WSL.

Dans la barre des tâches, vous tapez dans la barre de recherches wsl, quand il apparait vous l'ouvrez.
Une fois dans wsl avec une connexion internet vous tapez la commande suivante :

wget https://downloads.apache.org/hadoop/common/hadoop-3.4.1/hadoop-3.4.1.tar.gz

Cette commande téléchargera le fichier compressé hadoop-3.4.1.tar.gz depuis le site officiel d'Apache.

2ème Méthode : En cliquant sur le lien suivant :

https://downloads.apache.org/hadoop/common/hadoop-3.4.1/hadoop-3.4.1.tar.gz

En cliquant sur le lien cela vous permettra de télécharger directement le fichier compresser d'Hadoop. Pour retrouver le fichier compresser d'Hadoop, vous ouvrez l'explorateur de fichier, dans la barre à gauche vous verrez Téléchargements, vous

cliquez sur Téléchargement pour l'ouvrir et vous verrez à l'intérieur le fichier compresser d'Hadoop. Donc tous les fichiers que vous télécharger sont stockés par défaut dans Téléchargement.

Le Téléchargement de Hadoop étant terminé nous allons passer aux étapes suivantes.

Pour mon cas lorsque j'installais Ubuntu 22.04 j'ai créé un nom d'utilisateur et un mot de passe. Comme j'avais déjà le fichier compressé de Hadoop dans un dossier sur mon disque, j'ai donc copier le fichier compressé de Hadoop et je suis allé le collé dans Ubuntu-22.04\home\elie.

Toujours dans WSL nous allons exécuter les commandes suivantes :

- `tar -xzvf hadoop-3.4.1.tar.gz`

Cette commande extrait les fichiers de l'archive hadoop-3.4.1.tar.gz dans un dossier nommé hadoop-3.4.1.

- `sudo mv hadoop-3.4.1 /usr/local/hadoop`

Cette commande déplace le dossier extrait hadoop-3.4.1 dans le répertoire /usr/local et le renomme en hadoop pour simplifier son accès.
`sudo` permet d'exécuter la commande avec les privilèges administratifs nécessaires pour accéder au dossier /usr/local.
`mv` est la commande pour déplacer ou renommer des fichiers et dossiers.

- `nano $HADOOP_HOME/etc/hadoop/log4j.properties`

Cette commande ouvre le fichier log4j.properties dans l'éditeur nano pour modifier les paramètres de journalisation d'Hadoop.
`nano` est un éditeur de texte en ligne de commande.
`$HADOOP_HOME` est la variable d'environnement qui pointe vers le répertoire d'installation d'Hadoop (/usr/local/Hadoop).
`log4j.properties` est le fichier de configuration pour la journalisation des logs d'Hadoop. En modifiant ce fichier, tu peux ajuster les niveaux de log (INFO, DEBUG, ERROR) pour contrôler la quantité d'informations que Hadoop enregistre.

- **Configuration des variables d'environnement :**

Ouvrir le fichier '~/.bashrc' (en exécutant : `nano ~/.bashrc`) et ajouter :

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

export HADOOP_HOME=/usr/local/hadoop

export PATH=$PATH:$HADOOP_HOME/bin

export HADOOP_SBIN=$HADOOP_HOME/sbin

export PATH=$PATH:$HADOOP_SBIN

export HADOOP_MAPRED_HOME=$HADOOP_HOME

export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop

export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native

export HADOOP_OPTS=" Djava.library.path=$HADOOP_HOME/lib/native"

export HADOOP_STREAMING=$HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.4.1.jar

export HADOOP_LOG_DIR=$HADOOP_HOME/logs

export PDSH_RCMD_TYPE=ssh

Après avoir copier puis coller les codes ci-dessus dans ~/.bashrc,

NB : Lorsque vous ouvrez un fichier avec nano et que vous le modifié pour terminer :

Appuyez sur **Ctrl + O** pour sauvegarder le fichier.

Appuyez sur **Enter** pour confirmer.

Appuyez sur **Ctrl + X** pour quitter Nano.

Appliquer les modifications :

source ~/.bashrc

Ouvrir le fichier 'log4j.properties' :

```
nano $HADOOP_HOME/etc/hadoop/log4j.properties
```

et ajouter :

log4j.rootLogger=INFO, console

log4j.appender.console=org.apache.log4j.ConsoleAppender

log4j.appender.console.layout=org.apache.log4j.PatternLayout

log4j.appender.console.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %p %c{1} - %m%n

- **Configurer Hadoop :**

Créer le fichier hadoop-env.sh : Si ce fichier n'existe pas dans $HADOOP_HOME/etc/hadoop/, créez-le et assurez-vous d'y ajouter cette configuration Java :

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

Explication : Pour la création du fichier hadoop-env.sh on peut utiliser la commande suivante :

```
nano $HADOOP_HOME/etc/hadoop/hadoop-env.sh
```

1. Si le fichier hadoop-env.sh existe déjà :

- La commande ouvrira ce fichier pour modification dans l'éditeur de texte `nano`.

2. Si le fichier hadoop-env.sh n'existe pas mais que le dossier $HADOOP_HOME/etc/hadoop/ existe :

- La commande créera un fichier vide nommé hadoop-env.sh dans ce dossier, et vous pourrez le modifier.

3. Si le dossier $HADOOP_HOME/etc/hadoop/ n'existe pas :

- Vous obtiendrez une erreur indiquant que le fichier ou le répertoire n'existe pas.

Éditer le fichier 'core-site.xml' :

sudo nano /usr/local/hadoop/etc/hadoop/core-site.xml

<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://localhost:9000</value>

</property>

</configuration>

Éditer le fichier 'hdfs-site.xml' :

sudo nano /usr/local/hadoop/etc/hadoop/hdfs-site.xml

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

</configuration>

5. Formatage du système de fichiers HDFS :

```
hdfs namenode -format
```

Rappel : Avant de démarrer Hadoop il peut y avoir un problème entre Hadoop et le serveur SSH. J'ai rencontré ce problème. Parce que Hadoop va tenter de se connecter à des services via SSH pour démarrer les nœuds (NameNode, DataNode, Secondary NameNode). Donc il faut vérifier l'état de notre serveur SSH, c'est-à-dire vérifier s'il est connecté et aussi vous assurer que la connexion au serveur SSH fonctionne sans mot de passe, parce que Hadoop nécessite que SSH fonctionne sans mot de passe pour automatiser la communication entre les nœuds.

Je vous explique ces étapes là à travers les commandes suivantes :

1. Vérifier l'état de SSH

Commandes pour vérifier :

1. Si le client SSH est disponible :

```
ssh -V
```

Si cette commande renvoie une version de SSH (par exemple, OpenSSH_8.4), le client est installé.

2. Vérifiez si le serveur SSH est en cours d'exécution :

```
sudo service ssh status
```

Si le serveur SSH est actif, vous verrez un message indiquant qu'il fonctionne.

- Sinon, démarrez-le avec :

```
sudo service ssh start
```

3. Tester une connexion SSH locale :

Après avoir confirmé que le serveur SSH fonctionne, testez la connexion entre le client et le serveur sur localhost :

```
ssh localhost
```

- Si cette commande vous demande un mot de passe ou affiche un avertissement (comme "l'authenticité de l'hôte ne peut pas être établie"), SSH fonctionne. Continuez avec la configuration de l'authentification SSH sans mot de passe.
- Si vous obtenez une erreur comme Connection refused, passez à l'étape suivante :

2. Installer et configurer SSH si nécessaire

Si le serveur SSH n'est pas installé ou configuré correctement, procédez comme suit :

1. Installer le serveur SSH :

```
sudo apt update
```

Ensuite :

```
sudo apt install openssh-server -y
```

2. Démarrer le serveur SSH :

```
sudo service ssh start
```

3. Configurer SSH pour démarrer automatiquement :

```
sudo systemctl enable ssh
```

4. Vérifiez si SSH écoute sur le port 22 :

```
netstat -tlnp | grep :22
```

- Cela devrait afficher une ligne indiquant que SSH écoute sur 0.0.0.0:22 ou 127.0.0.1:22.

3. Configurer l'authentification SSH sans mot de passe

Hadoop nécessite que SSH fonctionne sans mot de passe pour automatiser la communication entre les nœuds.

Étapes :

1. Générer une clé SSH (si elle n'existe pas déjà) :

```
ssh-keygen -t rsa -P ""
```

- Cela génère une clé publique et privée dans ~/.ssh/id_rsaet ~/.ssh/id_rsa.pub.

2. Ajouter la clé publique à la liste des clés autorisées :

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

Ensuite :

```
chmod 600 ~/.ssh/authorized_keys
```

3. Tester la connexion SSH sans mot de passe :

```
ssh localhost
```

- Si cela fonctionne sans demander de mot de passe, la configuration est correcte.

6. Démarrer Hadoop :

```
start-dfs.sh
```

7. Vérification du fonctionnement :

Accéder à l'interface Web de Hadoop :

- Ouvrir un navigateur et aller sur 'http://localhost:9870'

Pour créer un répertoire et le vérifier :

`hdfs dfs -mkdir /test` (Cette commande crée un répertoire nommé /test dans HDFS)

`hdfs dfs -ls /` (Cette commande permet de lister le contenu du répertoire racine dans HDFS pour vérifier que le répertoire a bien été créé)

Conclusion
Après ces étapes, vous aurez installé Hadoop sous WSL et effectué un test de création d'un répertoire.

3- INSTALLATION DE SPARK

- **Téléchargement de Spark**

Pour le téléchargement de Spark nous pouvons utiliser deux méthodes :

1ère Méthode : Sous WSL.

Dans la barre des tâches, vous tapez dans la barre de recherches wsl, quand il apparait vous l'ouvrez.
Une fois dans wsl avec une connexion internet vous tapez la commande suivante :

wget https://downloads.apache.org/spark/spark-3.5.3/spark-3.5.3-bin-hadoop3.tgz

Cette commande téléchargera le fichier compressé *spark-3.5.3-bin-hadoop3.tgz* depuis le site officiel d'Apache.

2ème Méthode : En cliquant sur le lien suivant :

https://downloads.apache.org/spark/spark-3.5.3/spark-3.5.3-bin-hadoop3.tgz

En cliquant sur le lien cela vous permettra de télécharger directement le fichier compresser de Spark. Pour retrouver le fichier compresser de Spark, vous ouvrez l'explorateur de fichier, dans la barre à gauche vous verrez Téléchargements, vous cliquez sur Téléchargement pour l'ouvrir et vous verrez à l'intérieur le fichier compresser de Spark.

Pour mon cas lorsque j'installais ubuntu 22.04 j'ai créer un nom d'utilisateur et un mot de passe. Comme j'avais déjà le fichier compressé de Spark dans un dossier sur mon disque, j'ai donc copier le fichier compressé de Spark et je suis allé le collé dans Ubuntu-22.04\home\elie.

Le Téléchargement de Spark étant terminé nous allons passer aux étapes suivantes.

Toujours dans wsl nous allons exécuter les commandes suivantes :

```
tar xvf spark-3.5.3-bin-hadoop3.tgz
sudo mv spark-3.5.3-bin-hadoop3 /usr/local/spark
```

3. Configurer les variables d'environnement :

Ajoutez les lignes suivantes à votre fichier `~/.bashrc` :

export SPARK_HOME=/usr/local/spark

export PATH=$PATH:$SPARK_HOME/bin

4. Charger les nouvelles configurations :

```
source ~/.bashrc
```

5. Test d'un job simple :

1. Créer un fichier Python (test_spark.py) :

```
mkdir ~/spark_projects
```

Signification :

`mkdir` : Commande utilisée pour créer un répertoire (dossier) dans le système de fichiers.

`~/spark_projects` : Chemin du répertoire à créer.

`~` : Représentez le personnel de votre répertoire (par exemple, /home/elie sur Linux/WSL).

`spark_projects` : Le nom du nouveau répertoire.

Effet : Crée un répertoire nommé spark_projects dans votre répertoire personnel.

`cd ~/spark_projects/` # ou le répertoire de votre choix

Signification :

`cd` : Commande utilisée pour changer de répertoire (Change Directory).

`~/spark_projects/` : Chemin du répertoire dans lequel vous voulez naviguer.

Effet : Vous placez dans le répertoire spark_projects que vous venez de créer. Vous travaillez désormais à l'intérieur de ce dossier.

```
touch test_spark.py
```

Effet : Crée un fichier python nommer `test_spark.py`

Exécuter la commande : `nano test_spark.py` pour ouvrir le fichier python.

Copier et y coller le code ci-dessous :

```
from pyspark import SparkContext
sc = SparkContext("local", "Simple App")
data = [1, 2, 3, 4, 5]
rdd = sc.parallelize(data)
total = rdd.reduce(lambda a, b: a + b)
print("Total :", total)
sc.stop()
```

Appuyez sur **Ctrl + O** pour sauvegarder le fichier.

Appuyez sur **Enter** pour confirmer.

Appuyez sur **Ctrl + X** pour quitter Nano.

Un autre fichier 'test2_spark.py'

Copier et y coller le code ci-dessous :

```
from pyspark.sql import SparkSession
# Créer une session Spark
spark = SparkSession.builder \
.appName("Test Spark") \
.getOrCreate()
# Créer un DataFrame
data = [("BAKAYOKO", 1500000), ("YAO", 278000), ("TIEKOURA", 150000)]
```

df = spark.createDataFrame(data, ["Nom", "Salaire"])

 # Afficher le DataFrame

df.show()

 # Arrêter la session Spark

spark.stop()

2. Exécuter le job Spark :

```
spark-submit test_spark.py
```

En cas d'erreur d'adresse éditer fichier '~/.bashrc' ajouter le script ci-dessous puis charger avec la commande *'source ~/.bashrc*'

export SPARK_LOCAL_IP=10.255.255.254 # Remplacez cette adresse par une adresse IP de votre réseau

Cela devrait afficher `Total : 15`, indiquant que Spark fonctionne correctement.

Conclusion
Après ces étapes, vous aurez installé Spark sous WSL et effectué un test.