Université Abdelmalek Essaadi
Faculté des Sciences et Technique De Tanger
Département Génie Informatique

Architecture et Technologies Big Data

Mini-Projet Big Data : Installation et Analyse sur un Cluster Hadoop/Spark

Réalisé par :

GAROUD Fatima Ezzahraa

BOUASSAB Chaimae

ABABRI Chaimae

Encadré par :

Pr. Anouar Abdelhakim BOUDHIR

15/06/2025

Table des matières

Introduction

Dans le cadre du mini-projet Big Data, ce travail vise à mettre en place un environnement Big Data fonctionnel basé sur un cluster Hadoop/Spark, en utilisant deux machines virtuelles nommées ***ubuntumasterfch*** et ***ubuntuworkerfch***. L'objectif est de configurer un cluster opérationnel, capable de stocker, traiter et analyser de grands volumes de données, tout en simulant un cas d'usage réel. À travers les phases d'installation, de manipulation, d'analyse et de présentation, ce projet permet de maîtriser les technologies Hadoop et Spark, de comprendre leur intégration dans un environnement distribué, et de développer des compétences en analyse de données et en communication technique. En utilisant ***ubuntumasterfch*** comme nœud maître et ***ubuntuworkerfch*** comme nœud travailleur, ce projet illustre le fonctionnement d'un cluster Big Data dans un contexte pratique.

Rôle des machines virtuelles

Le cluster Hadoop/Spark est configuré avec deux machines virtuelles, chacune ayant un rôle spécifique pour assurer le fonctionnement distribué du système :

1. ubuntumasterfch (Nœud Maître) :
    - Rôle principal : Cette machine virtuelle agit comme le nœud maître (Master Node) du cluster Hadoop/Spark. Elle est responsable de la coordination et de la gestion des opérations du cluster.
    - Composants installés :
        - Hadoop NameNode : Gère les métadonnées du système de fichiers distribué HDFS, stockant les informations sur la localisation des données à travers le cluster.
        - Hadoop ResourceManager : Coordonne les ressources du cluster (CPU, mémoire) pour les tâches distribuées via YARN (Yet Another Resource Negotiator).
        - Spark Master : Supervise l'exécution des applications Spark, en répartissant les tâches entre les nœuds travailleurs.

- Responsabilités :
  - Orchestrer les opérations du cluster, comme la gestion des fichiers dans HDFS et la planification des tâches Spark.
  - Héberger les services de gestion (NameNode, ResourceManager, Spark Master).
  - Maintenir la communication avec le nœud travailleur pour assurer la cohérence des opérations.
- Configuration : Cette VM nécessite une configuration réseau stable (SSH, hostname, DNS local), un accès root sécurisé, et les variables d'environnement pour Hadoop et Spark correctement définies.

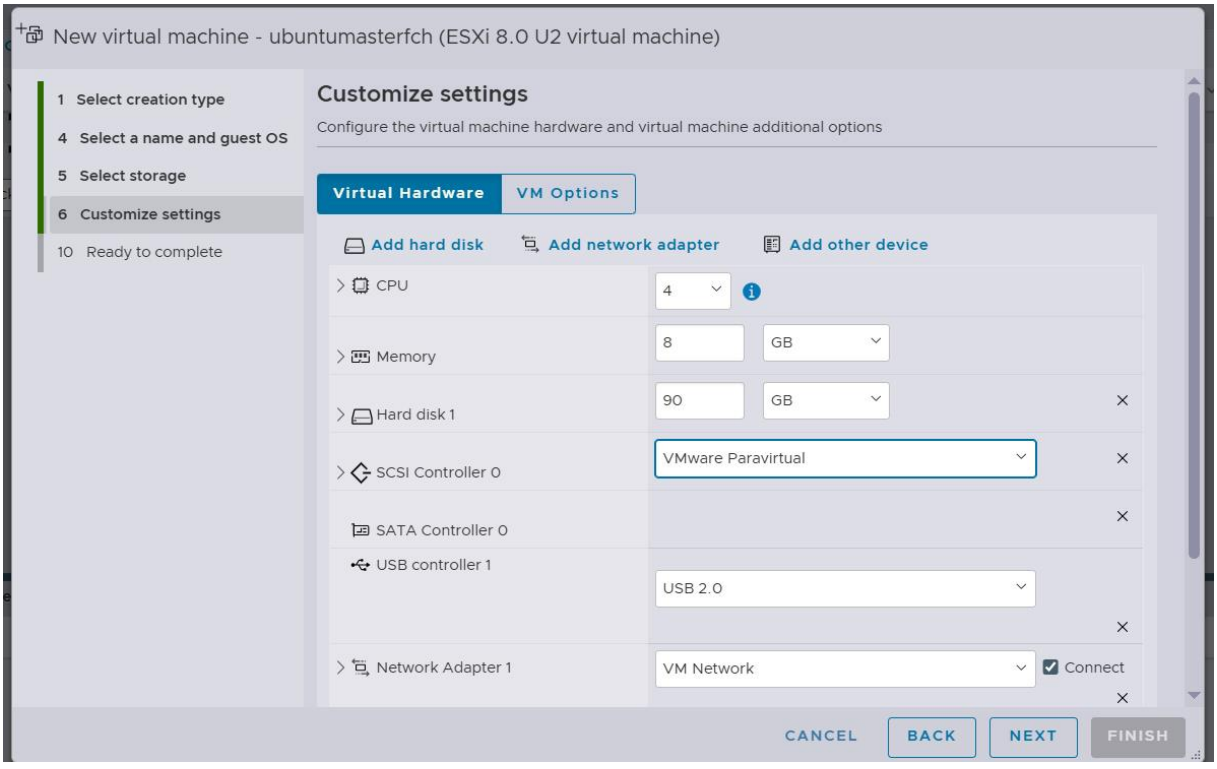
2. ubuntuworkerfch (Nœud Travailleur) :
   - Rôle principal : Cette machine virtuelle agit comme le nœud travailleur (Worker Node) du cluster. Elle exécute les tâches de calcul et de stockage confiées par le nœud maître.
   - Composants installés :
     - Hadoop DataNode : Stocke les blocs de données réels dans HDFS et répond aux requêtes du NameNode pour lire ou écrire des données.
     - Hadoop NodeManager : Gère les ressources locales (CPU, mémoire) pour exécuter les tâches distribuées par le ResourceManager via YARN.
     - Spark Worker : Exécute les tâches Spark assignées par le Spark Master, en effectuant des calculs parallèles sur les données.
   - Responsabilités :
     - Stocker les données dans HDFS (sous forme de blocs).
     - Effectuer les calculs distribués pour les jobs Spark, comme le filtrage, l'agrégation ou les jointures.
     - Communiquer avec le nœud maître pour recevoir les instructions et renvoyer les résultats.

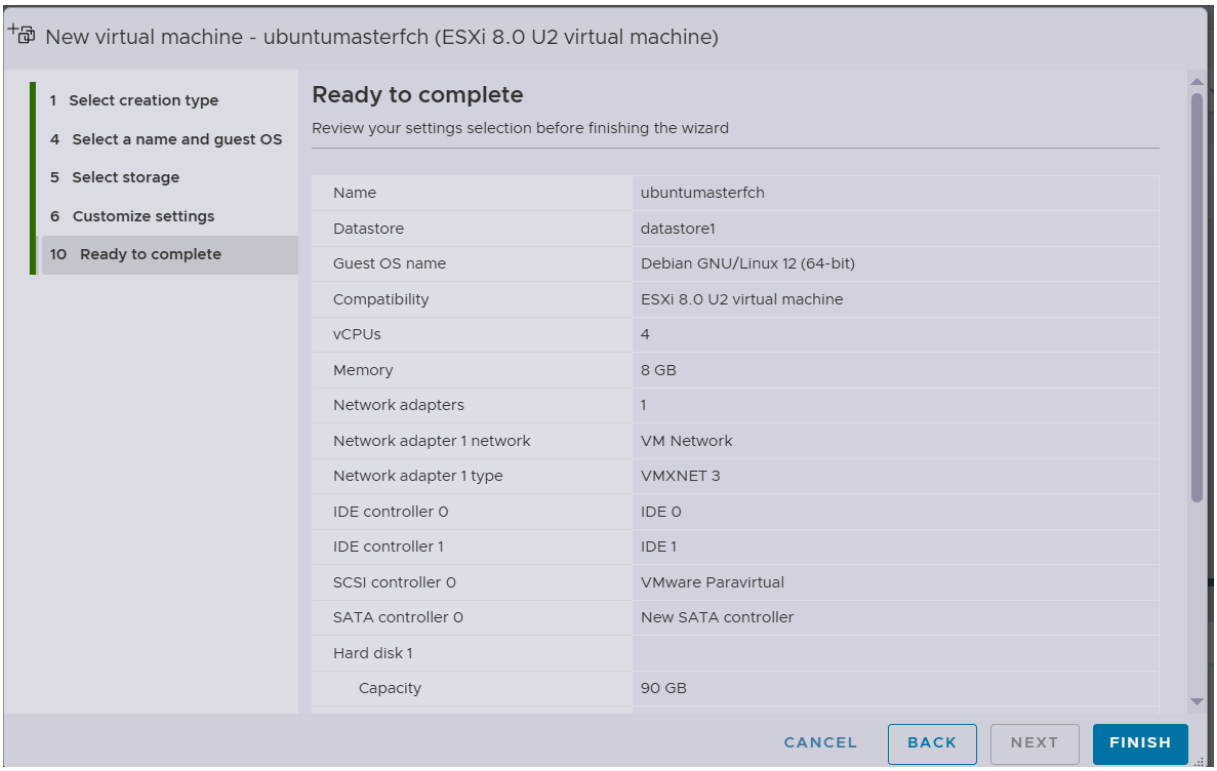
- Configuration : Cette VM doit être configurée pour communiquer avec *ubuntumasterfch* (via SSH et configuration réseau), et avoir les mêmes versions de Java, Hadoop et Spark installées pour garantir la compatibilité.

Interaction entre les VM :

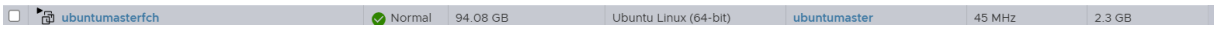
- Le nœud maître (*ubuntumasterfch*) contrôle et coordonne les opérations, tandis que le nœud travailleur (*ubuntuworkerfch*) exécute les tâches de stockage et de calcul.
- La communication entre les deux VM repose sur une configuration réseau appropriée (SSH, ports ouverts, DNS local) pour permettre l'échange de données et d'instructions.
- Dans un cluster réel, il pourrait y avoir plusieurs nœuds travailleurs, mais ici, *ubuntuworkerfch* simule un environnement distribué minimal.

Phase 1 – Prise en main et installation





Des étapes sur VM ubuntumasterfch :



Étape1 : Création de l'utilisateur hadoopuser :

1. Connection à la VM

```
C:\Users\DELL>ssh masterubuntu@192.168.180.49
```

2. Création de l'utilisateur hadoopuser

La création de l'utilisateur avec le repertoire personnel :

```
masterubuntu@masterubuntu:~$ sudo adduser hadoopuser
[sudo] password for masterubuntu:
Adding user `hadoopuser' ...
Adding new group `hadoopuser' (1001) ...
Adding new user `hadoopuser' (1001) with group `hadoopuser' ...
Creating home directory `/home/hadoopuser' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for hadoopuser
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
```

Cela crée un répertoire personnel /home/hadoopuser.

- En Donne à hadoopuser des privilèges sudo (pour installer des paquets et gérer les configurations système) :

```
masterubuntu@masterubuntu:~$ sudo usermod -aG sudo hadoopuser
```

Hadoop et Spark nécessitent des commandes sudo pour certaines installations (par exemple, apt install) et configurations de répertoires.

3. Configurer SSH pour hadoopuser

```
masterubuntu@masterubuntu:~$ su - hadoopuser
```

- Génération d'une clé SSH pour permettre la communication sans mot de passe avec le Worker (et pour Hadoop/Spark) :

```
hadoopuser@masterubuntu:~$ ssh-keygen -t rsa -P "" -f ~/.ssh/id_rsa
Generating public/private rsa key pair.
Created directory '/home/hadoopuser/.ssh'.
Your identification has been saved in /home/hadoopuser/.ssh/id_rsa
Your public key has been saved in /home/hadoopuser/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:x8rGrccH5RbcCijg1iTvpZsjB67pwmbaF/Sja09XcQc hadoopuser@masterubuntu
The key's randomart image is:
+---[RSA 3072]----+
|            E    |
|    o .      .   |
|   . *    ......  |
|    + + o..o+..   |
|   o o +S ++ o    |
|    o =o =. +     |
|.  . +.+*..o      |
|.= .*.=o .o .     |
|=o=+.=.... .      |
+----[SHA256]-----+
```

Cela crée :

- Clé privée : /home/hadoopuser/.ssh/id_rsa
- Clé publique : /home/hadoopuser/.ssh/id_rsa.pub

➢ Autorisation de la clé publique pour SSH local

```
hadoopuser@masterubuntu:~$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
hadoopuser@masterubuntu:~$ chmod 600 ~/.ssh/authorized_keys
```

➢ Teste de SSH localement

```
hadoopuser@masterubuntu:~$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ED25519 key fingerprint is SHA256:uz9IQYRw/fbvS5rU+vU1+b4m9B4WsONwidTGEzCyRCk.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-140-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Thu May 29 12:59:57 PM UTC 2025

  System load:  0.0               Processes:             203
  Usage of /:   16.2% of 42.49GB  Users logged in:       1
  Memory usage: 7%                IPv4 address for ens34: 192.168.180.49
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

55 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '24.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.
```

cela fonctionne sans demander de mot de passe, SSH est correctement configuré pour hadoopuser.

4. Vérifier les permissions

- Pour que hadoopuser a les permissions nécessaires sur son répertoire personnel :

```
hadoopuser@masterubuntu:~$ sudo chown -R hadoopuser:hadoopuser /home/hadoopuser
[sudo] password for hadoopuser:
```

```
hadoopuser@masterubuntu:~$ chmod 700 /home/hadoopuser/.ssh
```

Étape 2 : Configuration réseau

2.1 Configurer le nom d'hôte

➢ Connection en tant que hadoopuser :

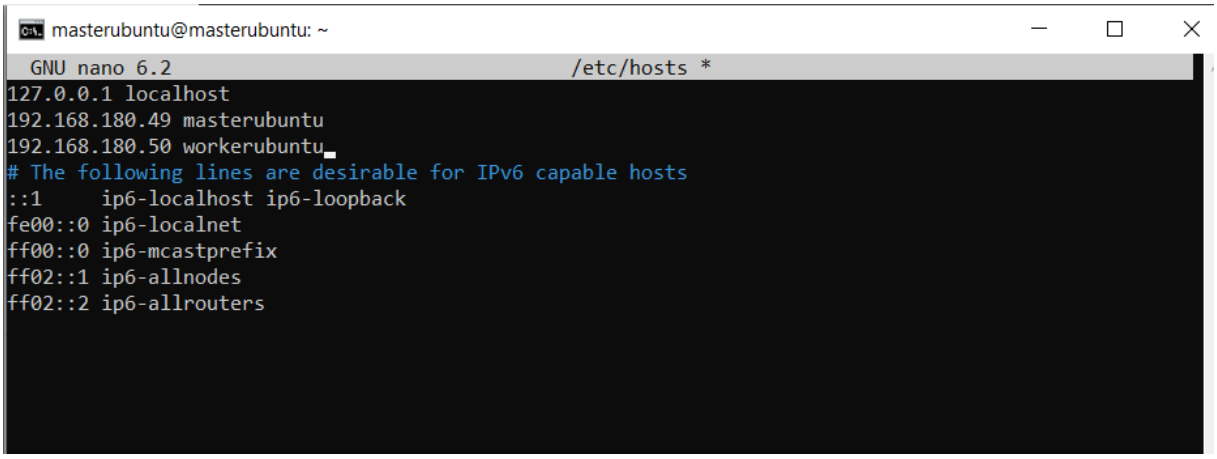
```
masterubuntu@masterubuntu:~$ su - hadoopuser
```

➢ Définition du nom d'hôte :

```
masterubuntu@masterubuntu:~$ sudo hostnamectl set-hostname ubuntumaster
[sudo] password for masterubuntu:
```

➢ Mettre à jour /etc/hosts

```
masterubuntu@masterubuntu:~$ sudo nano /etc/hosts
```



2.2 Configurer l'IP statique

- Vérification de l'interface réseau

```
masterubuntu@masterubuntu:~$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1
000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens34: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qle
n 1000
    link/ether 00:0c:29:7c:78:c6 brd ff:ff:ff:ff:ff:ff
    altname enp2s2
```

- Modification de /etc/netplan/00-installer-config.yaml

```
masterubuntu@masterubuntu:~$ sudo nano /etc/netplan/00-installer-config.yaml
```

masterubuntu@masterubuntu: ~

```
 GNU nano 6.2                    /etc/netplan/00-installer-config.yaml *
network:
  ethernets:
    ens34:
      addresses:
        - 192.168.180.49/24
      gateway4: 192.168.180.1
      nameservers:
        addresses: [8.8.8.8, 8.8.4.4]
  version: 2
```

- Application:

```
masterubuntu@masterubuntu:~$ sudo netplan apply

** (generate:2172): WARNING **: 14:29:51.596: Permissions for /etc/netplan/00-installer-config.yaml a
re too open. Netplan configuration should NOT be accessible by others.

** (generate:2172): WARNING **: 14:29:51.596: `gateway4` has been deprecated, use default routes inst
ead.
See the 'Default routes' section of the documentation for more details.
WARNING:root:Cannot call Open vSwitch: ovsdb-server.service is not running.

** (process:2170): WARNING **: 14:29:52.029: Permissions for /etc/netplan/00-installer-config.yaml ar
e too open. Netplan configuration should NOT be accessible by others.

** (process:2170): WARNING **: 14:29:52.030: `gateway4` has been deprecated, use default routes inste
ad.
See the 'Default routes' section of the documentation for more details.

** (process:2170): WARNING **: 14:29:52.283: Permissions for /etc/netplan/00-installer-config.yaml ar
e too open. Netplan configuration should NOT be accessible by others.

** (process:2170): WARNING **: 14:29:52.283: `gateway4` has been deprecated, use default routes inste
ad.
See the 'Default routes' section of the documentation for more details.

** (process:2170): WARNING **: 14:29:52.284: Permissions for /etc/netplan/00-installer-config.yaml ar
e too open. Netplan configuration should NOT be accessible by others.

** (process:2170): WARNING **: 14:29:52.284: `gateway4` has been deprecated, use default routes inste
ad.
See the 'Default routes' section of the documentation for more details.
```

- Vérification

```
masterubuntu@masterubuntu:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens34: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:0c:29:7c:78:c6 brd ff:ff:ff:ff:ff:ff
    altname enp2s2
    inet 192.168.180.49/24 metric 100 brd 192.168.180.255 scope global dynamic ens34
       valid_lft 86334sec preferred_lft 86334sec
    inet6 fe80::20c:29ff:fe7c:78c6/64 scope link
       valid_lft forever preferred_lft forever
```

2.3 Tester la connectivité

- Teste de la connexion réseau :

```
masterubuntu@masterubuntu:~$ ping -c 4 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=114 time=22.9 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=114 time=22.8 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=114 time=22.9 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=114 time=22.6 ms

--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 22.577/22.803/22.928/0.140 ms
masterubuntu@masterubuntu:~$ ping -c 4 google.com
PING google.com (216.58.209.78) 56(84) bytes of data.
64 bytes from mad07s22-in-f14.1e100.net (216.58.209.78): icmp_seq=1 ttl=114 time=21.3 ms
64 bytes from waw02s06-in-f14.1e100.net (216.58.209.78): icmp_seq=2 ttl=114 time=22.0 ms
64 bytes from waw02s06-in-f78.1e100.net (216.58.209.78): icmp_seq=3 ttl=114 time=20.9 ms
64 bytes from mad07s22-in-f14.1e100.net (216.58.209.78): icmp_seq=4 ttl=114 time=21.7 ms

--- google.com ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 20.904/21.469/21.970/0.405 ms
masterubuntu@masterubuntu:~$
```

Étape 3 : Configurer le pare-feu

- Installation de UFW :

```
masterubuntu@masterubuntu:~$ sudo apt install ufw -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ufw is already the newest version (0.36.1-4ubuntu0.1).
ufw set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 57 not upgraded.
```

- Autorisation des ports nécessaires :

```
masterubuntu@masterubuntu:~$ sudo ufw allow 22
Rules updated
Rules updated (v6)
masterubuntu@masterubuntu:~$ sudo ufw allow 9000
Rules updated
Rules updated (v6)
masterubuntu@masterubuntu:~$ sudo ufw allow 9870
Rules updated
Rules updated (v6)
masterubuntu@masterubuntu:~$ sudo ufw allow 8088
Rules updated
Rules updated (v6)
masterubuntu@masterubuntu:~$ sudo ufw allow 7077
Rules updated
Rules updated (v6)
masterubuntu@masterubuntu:~$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
```

- Vérification :

```
masterubuntu@masterubuntu:~$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22                         ALLOW       Anywhere
9000                       ALLOW       Anywhere
9870                       ALLOW       Anywhere
8088                       ALLOW       Anywhere
7077                       ALLOW       Anywhere
22 (v6)                    ALLOW       Anywhere (v6)
9000 (v6)                  ALLOW       Anywhere (v6)
9870 (v6)                  ALLOW       Anywhere (v6)
8088 (v6)                  ALLOW       Anywhere (v6)
7077 (v6)                  ALLOW       Anywhere (v6)
```

Étape 4 : Installer Java

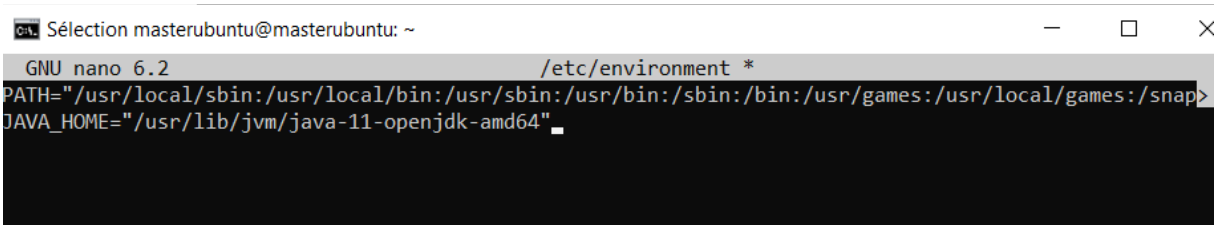
- Installation d'OpenJDK 11 :

```
masterubuntu@masterubuntu:~$ sudo apt install openjdk-11-jdk -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  alsa-topology-conf alsa-ucm-conf at-spi2-core ca-certificates-java dconf-gsettings-backend
  dconf-service fontconfig-config fonts-dejavu-core fonts-dejavu-extra gsettings-desktop-schemas
  java-common libasound2 libasound2-data libatk-bridge2.0-0 libatk-wrapper-java
  libatk-wrapper-java-jni libatk1.0-0 libatk1.0-data libatspi2.0-0 libavahi-client3
  libavahi-common-data libavahi-common3 libcups2 libdconf1 libdrm-amdgpu1 libdrm-intel1
  libdrm-nouveau2 libdrm-radeon1 libfontconfig1 libfontenc1 libgif7 libgl1 libgl1-amber-dri
  libgl1-mesa-dri libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libgraphite2-3 libharfbuzz0b
  libice-dev libice6 libjpeg-turbo8 libjpeg8 liblcms2-2 libllvm15 libpciaccess0 libpcsclite1
  libpthread-stubs0-dev libsensors-config libsensors5 libsm-dev libsm6 libx11-dev libx11-xcb1
  libxau-dev libxaw7 libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0 libxcb-present0 libxcb-randr0
  libxcb-shape0 libxcb-shm0 libxcb-sync1 libxcb-xfixes0 libxcb1-dev libxcomposite1 libxdmcp-dev
  libxfixes3 libxft2 libxi6 libxinerama1 libxkbfile1 libxmu6 libxpm4 libxrandr2 libxrender1
  libxshmfence1 libxt-dev libxt6 libxtst6 libxv1 libxxf86dga1 libxxf86vm1 openjdk-11-jdk-headless
  openjdk-11-jre openjdk-11-jre-headless session-migration x11-common x11-utils x11proto-dev
```

- Vérification :

```
masterubuntu@masterubuntu:~$ java -version
openjdk version "11.0.27" 2025-04-15
OpenJDK Runtime Environment (build 11.0.27+6-post-Ubuntu-0ubuntu122.04)
OpenJDK 64-Bit Server VM (build 11.0.27+6-post-Ubuntu-0ubuntu122.04, mixed mode, sharing)
```

- Configuration JAVA_HOME



Application :

```
masterubuntu@masterubuntu:~$ source /etc/environment
```

Vérification :

```
masterubuntu@masterubuntu:~$ echo $JAVA_HOME
/usr/lib/jvm/java-11-openjdk-amd64
```

Étape 5 : Installer et configurer Hadoop

5.1 Installer Hadoop

- Téléchargement de Hadoop 3.3.6

```
masterubuntu@masterubuntu:~$ wget https://downloads.apache.org/hadoop/common/hadoop-3.3.6/hadoop-3.3.
6.tar.gz
--2025-05-29 14:48:49--  https://downloads.apache.org/hadoop/common/hadoop-3.3.6/hadoop-3.3.6.tar.gz
Resolving downloads.apache.org (downloads.apache.org)... 135.181.214.104, 88.99.208.237, 2a01:4f9:3a:
2c57::2, ...
Connecting to downloads.apache.org (downloads.apache.org)|135.181.214.104|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 730107476 (696M) [application/x-gzip]
Saving to: ‘hadoop-3.3.6.tar.gz’

hadoop-3.3.6.tar.gz        37%[============>                    ] 263.96M  5.83MB/s    eta 9m 5s
```

- Décompression du fichier :

```
masterubuntu@masterubuntu:~$ sudo tar -xzf hadoop-3.3.6.tar.gz -C /usr/local
masterubuntu@masterubuntu:~$ sudo mv /usr/local/hadoop-3.3.6 /usr/local/hadoop
masterubuntu@masterubuntu:~$ sudo chown -R hadoopuser:hadoopuser /usr/local/hadoop
```

- Configurer les variables d'environnement :

```
masterubuntu@masterubuntu:~$ nano ~/.bashrc
```

- L'Ajoute :

```
# === Java environment ===
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
export PATH=$PATH:$JAVA_HOME/bin

export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
```

- Application :

```
masterubuntu@masterubuntu:~$ source ~/.bashrc
```

- Vérification :

```
hadoopuser@ubuntumaster:~$ hadoop version
Hadoop 3.3.6
Source code repository https://github.com/apache/hadoop.git -r 1be78238728da9266a4f88195058f08fd012bf
9c
Compiled by ubuntu on 2023-06-18T08:22Z
Compiled on platform linux-x86_64
Compiled with protoc 3.7.1
From source with checksum 5652179ad55f76cb287d9c633bb53bbd
This command was run using /usr/local/hadoop/share/hadoop/common/hadoop-common-3.3.6.jar
```

5.2 Configurer Hadoop

- L'Accès au répertoire de configuration :

```
hadoopuser@ubuntumaster:/usr/local/hadoop/etc/hadoop$
```

- Modification des fichiers :
  - **core-site.xml** :

```
hadoopuser@ubuntumaster:/usr/local/hadoop/etc/hadoop$ nano core-site.xml
```

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

  - **hdfs-site.xml :**

```
hadoopuser@ubuntumaster:/usr/local/hadoop/etc/hadoop$ nano hdfs-site.xml
```

```
<configuration>
<property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:///usr/local/hadoop/hdfs/namenode</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:///usr/local/hadoop/hdfs/datanode</value>
  </property>
</configuration>
```

  - **yarn-site.xml**

```
hadoopuser@ubuntumaster:/usr/local/hadoop/etc/hadoop$ nano yarn-site.xml
```

```
<configuration>

<!-- Site specific YARN configuration properties -->
<property>
    <name>yarn.resourcemanager.hostname</name>
    <value>ubuntumaster</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
</configuration>
```

- **mapred-site.xml**

```
hadoopuser@ubuntumaster:/usr/local/hadoop/etc/hadoop$ nano mapred-site.xml
```

```
<configuration>
<property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

- **workers**

```
hadoopuser@ubuntumaster:/usr/local/hadoop/etc/hadoop$ nano workers
```

```
hadoopuser@ubuntumaster: /usr/local/hadoop/etc/hadoop
 GNU nano 6.2                                  workers *
workerubuntu
```

- Création des répertoires HDFS :

```
hadoopuser@ubuntumaster:/usr/local/hadoop/etc/hadoop$ sudo mkdir -p /usr/local/hadoop/hdfs/namenode
[sudo] password for hadoopuser:
```

```
hadoopuser@ubuntumaster:/usr/local/hadoop/etc/hadoop$ sudo mkdir -p /usr/local/hadoop/hdfs/datanode
hadoopuser@ubuntumaster:/usr/local/hadoop/etc/hadoop$ sudo chown -R hadoopuser:hadoopuser /usr/local/
hadoop
```

5.3 Formater le NameNode

- Formater le NameNode :

```
hadoopuser@ubuntumaster:/usr/local/hadoop/etc/hadoop$ hdfs namenode -format
WARNING: /usr/local/hadoop/logs does not exist. Creating.
2025-05-29 15:48:43,217 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = ubuntumaster/192.168.180.49
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 3.3.6
STARTUP_MSG:   classpath = /usr/local/hadoop/etc/hadoop:/usr/local/hadoop/share/hadoop/common/lib/net
ty-codec-stomp-4.1.89.Final.jar:/usr/local/hadoop/share/hadoop/common/lib/zookeeper-3.6.3.jar:/usr/lo
cal/hadoop/share/hadoop/common/lib/paranamer-2.3.jar:/usr/local/hadoop/share/hadoop/common/lib/jetty-
webapp-9.4.51.v20230217.jar:/usr/local/hadoop/share/hadoop/common/lib/netty-transport-4.1.89.Final.ja
r:/usr/local/hadoop/share/hadoop/common/lib/checker-qual-2.5.2.jar:/usr/local/hadoop/share/hadoop/com
mon/lib/kerb-simplekdc-1.0.1.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-math3-3.1.1.jar:/u
sr/local/hadoop/share/hadoop/common/lib/javax.servlet-api-3.1.0.jar:/usr/local/hadoop/share/hadoop/co
mmon/lib/kerby-util-1.0.1.jar:/usr/local/hadoop/share/hadoop/common/lib/jetty-servlet-9.4.51.v2023021
7.jar:/usr/local/hadoop/share/hadoop/common/lib/netty-codec-socks-4.1.89.Final.jar:/usr/local/hadoop/
share/hadoop/common/lib/slf4j-api-1.7.36.jar:/usr/local/hadoop/share/hadoop/common/lib/netty-codec-sm
```

Étape 6 : Installer et configurer Spark

6.1 Installer Spark

- Téléchargement Spark 3.4.2 :

```
hadoopuser@ubuntumaster:~$ wget https://archive.apache.org/dist/spark/spark-3.4.2/spark-3.4.2-bin-had
oop3.tgz
--2025-05-29 16:31:01--  https://archive.apache.org/dist/spark/spark-3.4.2/spark-3.4.2-bin-hadoop3.tg
z
Resolving archive.apache.org (archive.apache.org)... 65.108.204.189, 2a01:4f9:1a:a084::2
Connecting to archive.apache.org (archive.apache.org)|65.108.204.189|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 388664780 (371M) [application/x-gzip]
Saving to: 'spark-3.4.2-bin-hadoop3.tgz'

spark-3.4.2-bin-hadoop3.t  15%[====>                            ]  55.79M  9.87MB/s    eta 34s
```

- Vérification que le fichier est téléchargé :

```
hadoopuser@ubuntumaster:~$ ls -lh spark-3.4.2-bin-hadoop3.tgz
-rw-rw-r-- 1 hadoopuser hadoopuser 371M Nov 25  2023 spark-3.4.2-bin-hadoop3.tgz
```

- Décompression du fichier

```
hadoopuser@ubuntumaster:~$ sudo tar -xzf spark-3.4.2-bin-hadoop3.tgz -C /usr/local
[sudo] password for hadoopuser:
```

```
hadoopuser@ubuntumaster:~$ sudo mv /usr/local/spark-3.4.2-bin-hadoop3 /usr/local/spark
hadoopuser@ubuntumaster:~$ sudo chown -R hadoopuser:hadoopuser /usr/local/spark
```

- Configuration des variables d'environnement :

```
hadoopuser@ubuntumaster:~$ nano ~/.bashrc
```

```
export SPARK_HOME=/usr/local/spark
export PATH=$PATH:$SPARK_HOME/bin
```

- Application :

```
hadoopuser@ubuntumaster:~$ source ~/.bashrc
```

- Vérification :

```
hadoopuser@ubuntumaster:~$ spark-shell --version
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.4.2
      /_/

Using Scala version 2.12.17, OpenJDK 64-Bit Server VM, 11.0.27
Branch HEAD
Compiled by user centos on 2023-11-25T07:25:47Z
Revision 0c0e7d4087c64efca259b4fb656b8be643be5686
Url https://github.com/apache/spark
Type --help for more information.
```

6.2 Configurer Spark

- L'Accès au répertoire de configuration :

```
hadoopuser@ubuntumaster:~$ cd $SPARK_HOME/conf
hadoopuser@ubuntumaster:/usr/local/spark/conf$
```

- Modifier spark-env.sh

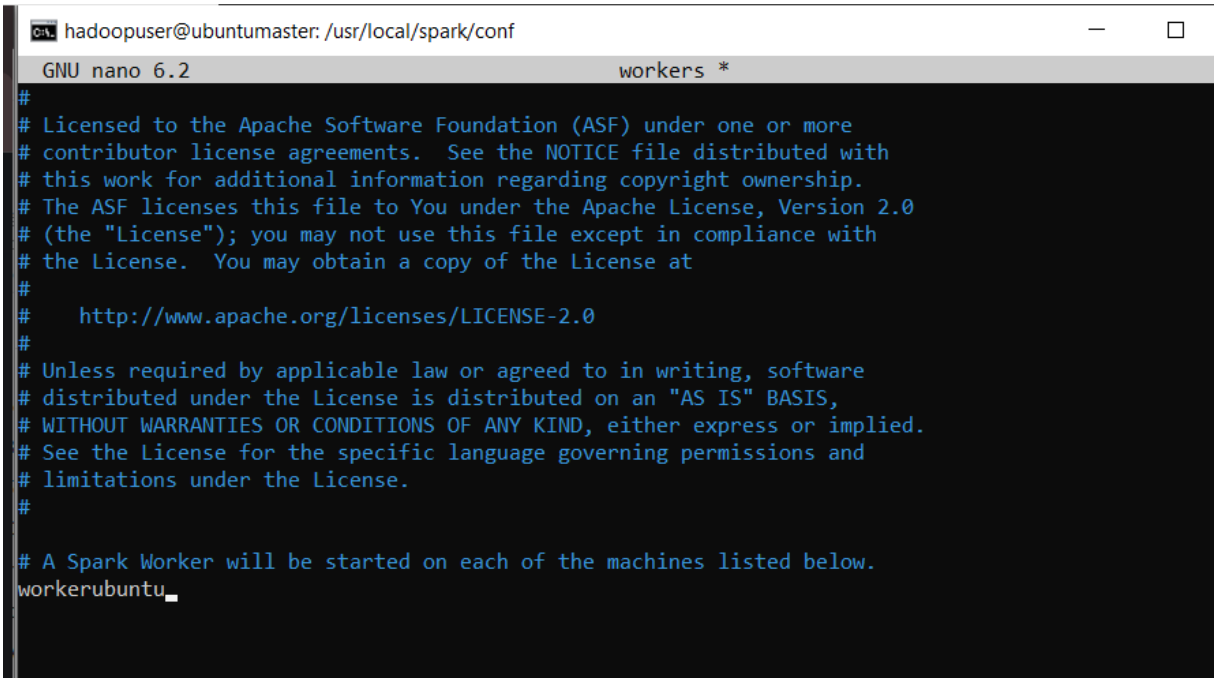
```
hadoopuser@ubuntumaster:/usr/local/spark/conf$ cp spark-env.sh.template spark-env.sh
hadoopuser@ubuntumaster:/usr/local/spark/conf$ nano spark-env.sh
```

- Contenu :

```
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
export HADOOP_CONF_DIR=/usr/local/hadoop/etc/hadoop
export SPARK_MASTER_HOST=ubuntumaster
```

- Modifier workers :

```
hadoopuser@ubuntumaster:/usr/local/spark/conf$ cp workers.template workers
hadoopuser@ubuntumaster:/usr/local/spark/conf$ nano workers
```



Des etapes sur VM ubuntuworkerfch :

Objectif :

Intégrer un serveur Ubuntu comme **nœud worker** dans un cluster Hadoop/Spark afin qu'il puisse exécuter les tâches de traitement de données distribuées sous la supervision du nœud master.

1. Installation de la distribution Ubuntu Server

Nous avons utilisé une **machine dédiée** dans la salle E26 pour installer **Ubuntu Server 22.04 LTS** (version minimale, sans interface graphique).

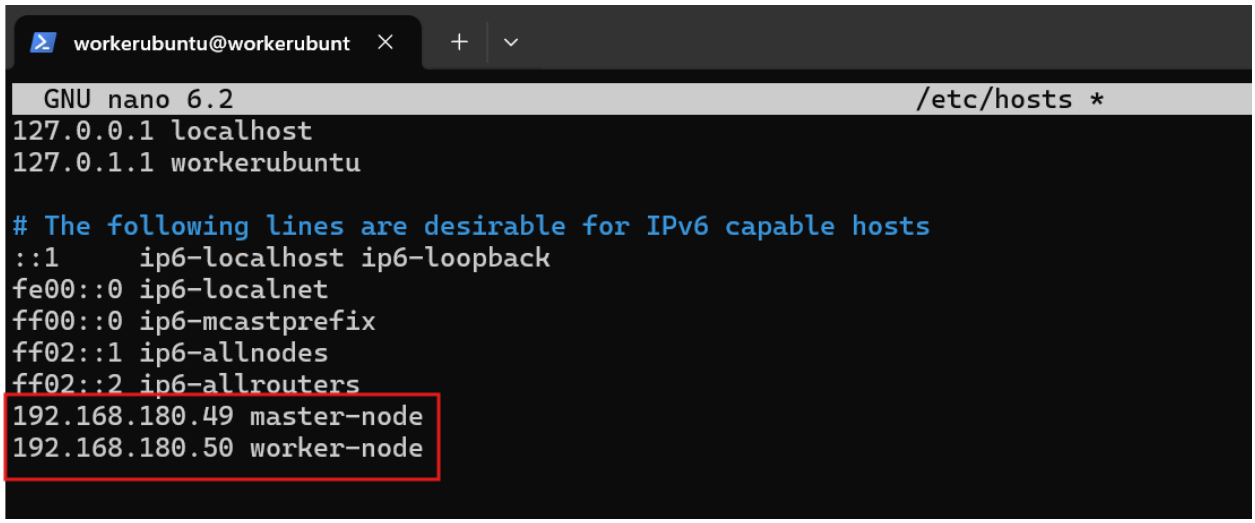
```
PS C:\Users\surface> ssh workerubuntu@192.168.180.50
workerubuntu@192.168.180.50's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-140-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Thu May 29 01:36:55 PM UTC 2025

  System load:  0.0                Processes:             194
  Usage of /:   26.0% of 27.86GB   Users logged in:       1
  Memory usage: 4%                 IPv4 address for ens34: 192.168.180.50
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

135 updates can be applied immediately.
80 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable
```

2. Configuration système initiale

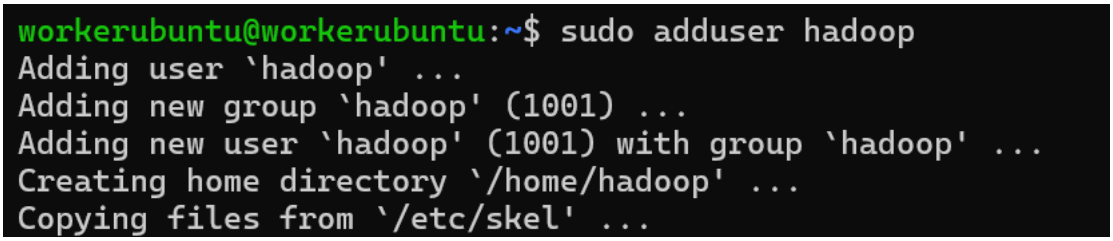
- **Nom d'hôte personnalisé** :

```
workerubuntu@workerubuntu:~$ sudo hostnamectl set-hostname worker-node
```

- **Configuration du fichier /etc/hosts** :

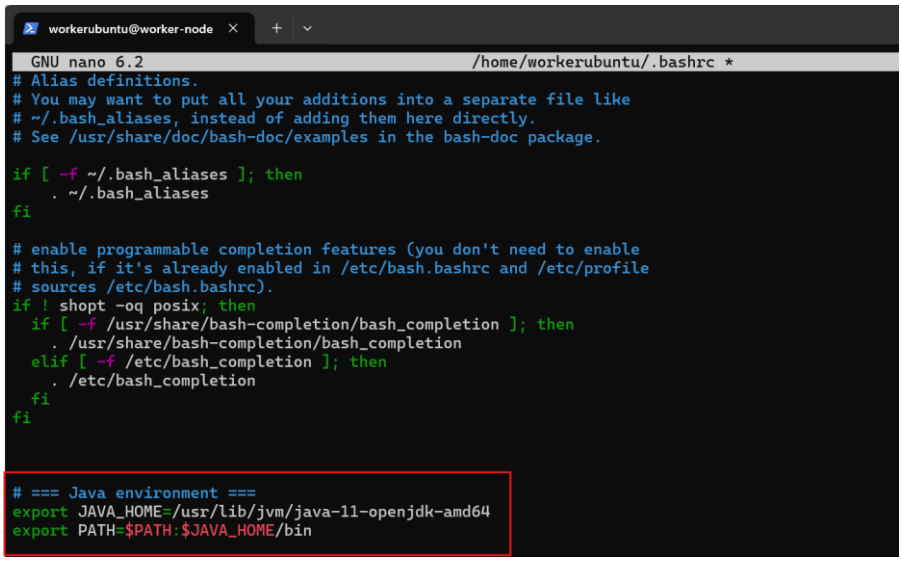


- **Création d'un utilisateur hadoop** :



3. Installation de Java

```
workerubuntu@worker-node:~$ java -version
openjdk version "11.0.27" 2025-04-15
OpenJDK Runtime Environment (build 11.0.27+6-post-Ubuntu-0ubuntu122.04)
OpenJDK 64-Bit Server VM (build 11.0.27+6-post-Ubuntu-0ubuntu122.04, mixed mode, sharing)
```




```
workerubuntu@worker-node:~$ source ~/.bashrc
```

4. Installation de Hadoop et Spark

- **Déplacement dans /usr/local**

```
workerubuntu@worker-node:~$ wget https://dlcdn.apache.org/hadoop/common/hadoop-3.3.6/hadoop-3.3.6.tar.gz
wget https://dlcdn.apache.org/spark/spark-3.5.0/spark-3.5.0-bin-hadoop3.tgz
--2025-05-29 14:27:20--  https://dlcdn.apache.org/hadoop/common/hadoop-3.3.6/hadoop-3.3.6.tar.gz
Resolving dlcdn.apache.org (dlcdn.apache.org)... 151.101.2.132, 2a04:4e42::644
Connecting to dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 730107476 (696M) [application/x-gzip]
Saving to: ‘hadoop-3.3.6.tar.gz’

hadoop-3.3.6.tar.gz          100%[=============================================>] 696.28M  2.24MB/s    in 8m 50s
```

- **Décompression des fichiers téléchargés**

```
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.1.3.xml
hadoop-3.3.6/share/hadoop/hdfs/hadoop-hdfs-client-3.3.6-tests.jar
hadoop-3.3.6/share/hadoop/hdfs/hadoop-hdfs-httpfs-3.3.6.jar
tar (child): spark-3.5.0-bin-hadoop3.tgz: Cannot open: No such file or directory
```

```
spark-3.5.0-bin-hadoop3/bin/spark-shell
spark-3.5.0-bin-hadoop3/bin/find-spark-home
workerubuntu@worker-node:~$ sudo mv spark-3.5.0-bin-hadoop3 /usr/local/spark
```

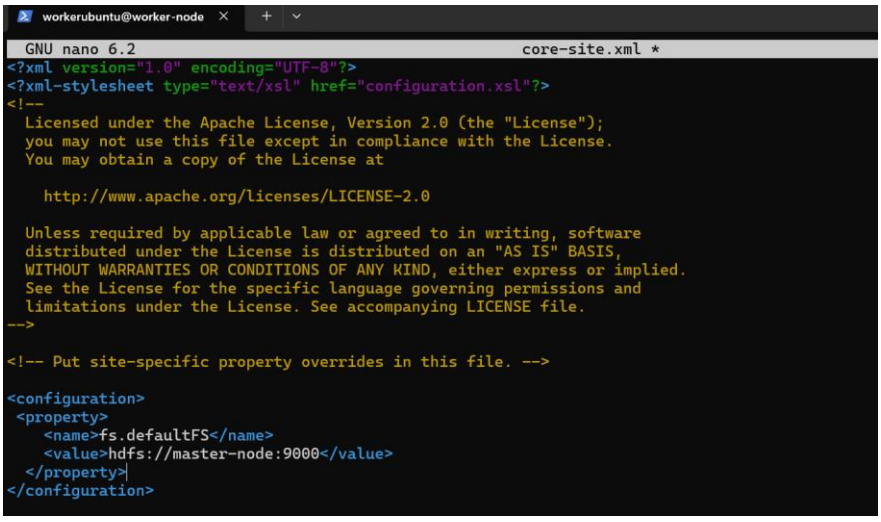
```
# === Hadoop & Spark ===
export HADOOP_HOME=/usr/local/hadoop
export SPARK_HOME=/usr/local/spark
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$SPARK_HOME/bin:$SPARK_HOME/sbin
```

```
workerubuntu@worker-node:~$ hadoop version
Hadoop 3.3.6
Source code repository https://github.com/apache/hadoop.git -r 1be78238728da9266a4f88195058f08fd012bf9c
Compiled by ubuntu on 2023-06-18T08:22Z
Compiled on platform linux-x86_64
Compiled with protoc 3.7.1
From source with checksum 5652179ad55f76cb287d9c633bb53bbd
This command was run using /usr/local/hadoop/share/hadoop/common/hadoop-common-3.3.6.jar
workerubuntu@worker-node:~$ spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/05/29 15:13:46 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java
asses where applicable
Spark context Web UI available at http://worker-node:4040
Spark context available as 'sc' (master = local[*], app id = local-1748531627013).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.0
      /_/

Using Scala version 2.12.18 (OpenJDK 64-Bit Server VM, Java 11.0.27)
Type in expressions to have them evaluated.
Type :help for more information.
```

5. Configuration des fichiers Hadoop

Dans le dossier $HADOOP_HOME/etc/hadoop :

- **core-site.xml** :



- **hdfs-site.xml** :

```
<configuration>
  <property>
    <name>dfs.data.dir</name>
    <value>/home/hadoop/hdfs/data</value>
  </property>
</configuration>
```

- **yarn-site.xml** :

```
<configuration>

<!-- Site specific YARN configuration properties -->

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

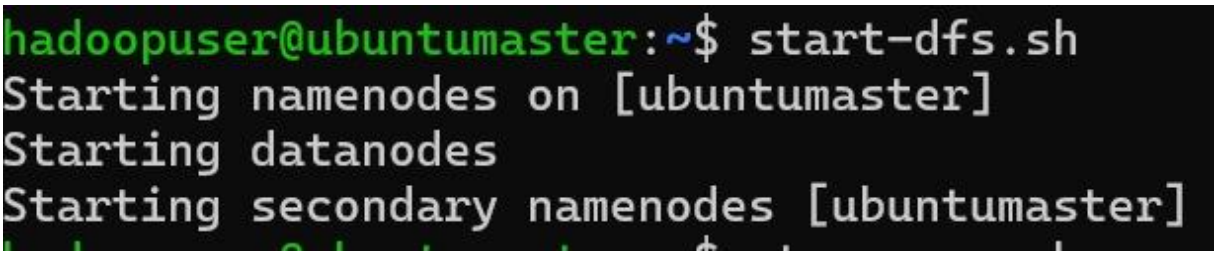
Phase 2 – Manipulation et stockage

Étape 1 : Formatage de hdfs :

```
hadoopuser@ubuntumaster:~$ hdfs namenode -format
2025-05-30 13:02:31,867 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = ubuntumaster/192.168.180.49
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 3.3.6
STARTUP_MSG:   classpath = /usr/local/hadoop/etc/hadoop:/usr/local/hadoop/share/hadoop/common/lib/netty-codec-stomp-4.1.
89.Final.jar:/usr/local/hadoop/share/hadoop/common/lib/zookeeper-3.6.3.jar:/usr/local/hadoop/share/hadoop/common/lib/par
anamer-2.3.jar:/usr/local/hadoop/share/hadoop/common/lib/jetty-webapp-9.4.51.v20230217.jar:/usr/local/hadoop/share/hadoo
p/common/lib/netty-transport-4.1.89.Final.jar:/usr/local/hadoop/share/hadoop/common/lib/checker-qual-2.5.2.jar:/usr/loca
l/hadoop/share/hadoop/common/lib/kerb-simplekdc-1.0.1.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-math3-3.1.1.
jar:/usr/local/hadoop/share/hadoop/common/lib/javax.servlet-api-3.1.0.jar:/usr/local/hadoop/share/hadoop/common/lib/kerb
y-util-1.0.1.jar:/usr/local/hadoop/share/hadoop/common/lib/jetty-servlet-9.4.51.v20230217.jar:/usr/local/hadoop/share/ha
doop/common/lib/netty-codec-socks-4.1.89.Final.jar:/usr/local/hadoop/share/hadoop/common/lib/slf4j-api-1.7.36.jar:/usr/l
ocal/hadoop/share/hadoop/common/lib/netty-codec-smtp-4.1.89.Final.jar:/usr/local/hadoop/share/hadoop/common/lib/avro-1.7
.7.jar:/usr/local/hadoop/share/hadoop/common/lib/jetty-util-9.4.51.v20230217.jar:/usr/local/hadoop/share/hadoop/common/l
ib/jsp-api-2.1.jar:/usr/local/hadoop/share/hadoop/common/lib/netty-codec-dns-4.1.89.Final.jar:/usr/local/hadoop/share/ha
doop/common/lib/slf4j-reload4j-1.7.36.jar:/usr/local/hadoop/share/hadoop/common/lib/netty-transport-native-epoll-4.1.89.
Final-linux-aarch_64.jar:/usr/local/hadoop/share/hadoop/common/lib/jackson-mapper-asl-1.9.13.jar:/usr/local/hadoop/share
/hadoop/common/lib/netty-handler-4.1.89.Final.jar:/usr/local/hadoop/share/hadoop/common/lib/jsch-0.1.55.jar:/usr/local/h
adoop/share/hadoop/common/lib/zookeeper-jute-3.6.3.jar:/usr/local/hadoop/share/hadoop/common/lib/netty-transport-rxtx-4.
1.89.Final.jar:/usr/local/hadoop/share/hadoop/common/lib/jetty-server-9.4.51.v20230217.jar:/usr/local/hadoop/share/hadoo
p/common/lib/jackson-databind-2.12.7.1.jar:/usr/local/hadoop/share/hadoop/common/lib/jersey-json-1.20.jar:/usr/local/had
oop/share/hadoop/common/lib/hadoop-shaded-protobuf_3_7-1.1.1.jar:/usr/local/hadoop/share/hadoop/common/lib/netty-resolve
```

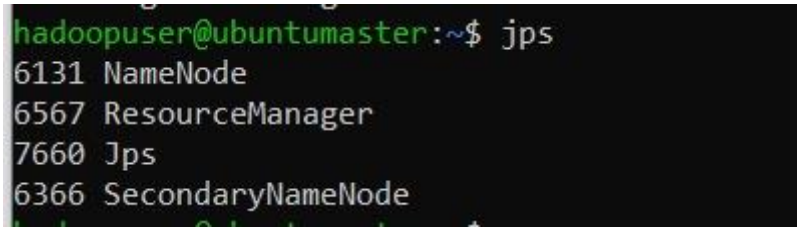
Étape 2 : Tester les services

- **Démarrage de HDFS :**

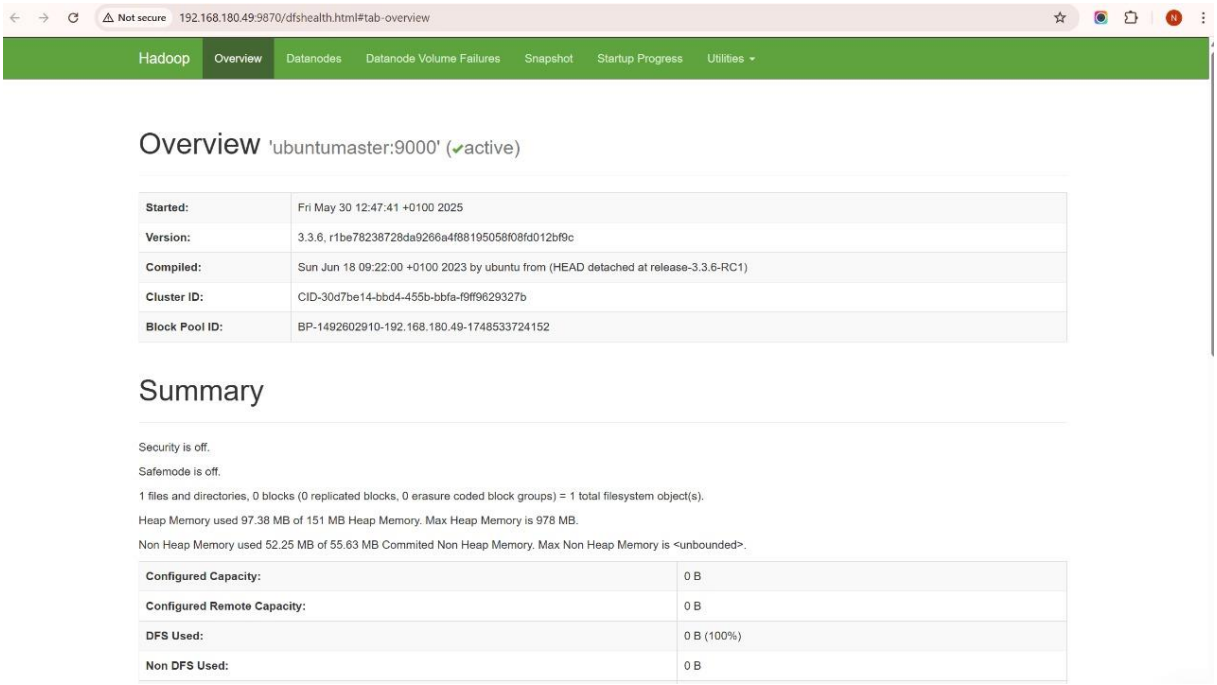


- **Démarrage de YARN :**

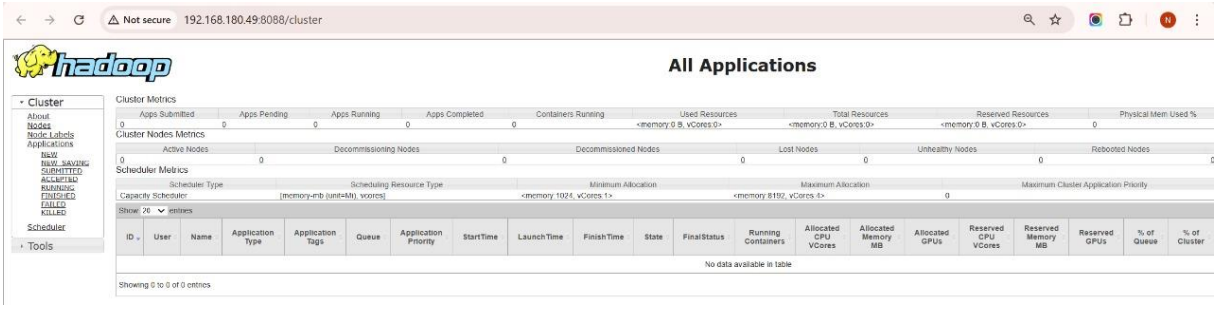
- **Vérification des processus :**



- **Vérification dans le navigateur :**
  - HDFS : http://192.168.180.49:9870



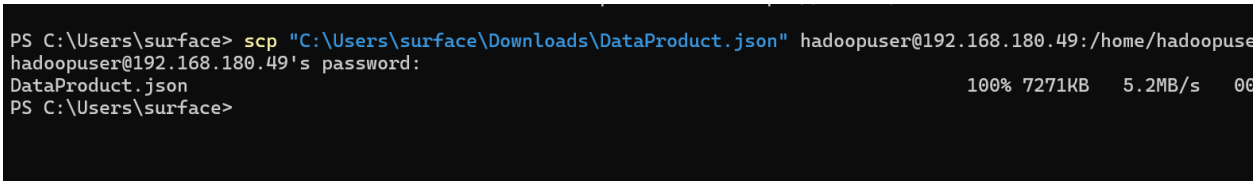
- YARN : http://192.168.180.49:8088



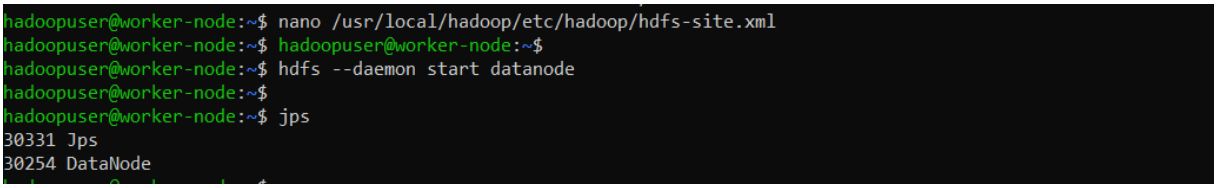
Etape 3 : Transfert du fichier .csv dans MasterUbuntu

Dans cette étape, nous avons transféré un fichier de données nommé mental_disorders_reddittttt.csv depuis un poste local Windows vers le serveur Ubuntu Master, via le protocole SCP (Secure Copy Protocol).

Commande utilisée :



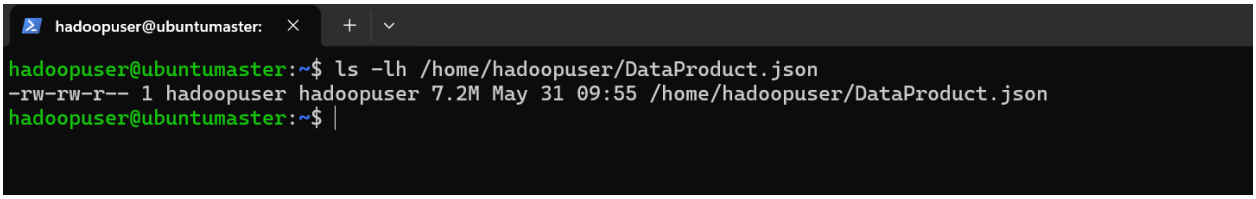
Activation réussie du service DataNode sur le nœud worker :



La commande jps confirme maintenant que DataNode est bien actif (PID : 30254), validant ainsi la participation du nœud worker au cluster Hadoop.

Etape 4 : Ingestion dans HDFS

Une fois le fichier présent sur le master, nous avons préparé le stockage dans HDFS pour permettre un traitement distribué via Spark.



```
hadoopuser@ubuntumaster:~$ cd /usr/local/hadoop/sbin
./start-dfs.sh
Starting namenodes on [ubuntumaster]
Starting datanodes
workerubuntu: ssh: connect to host workerubuntu port 22: No route to host
Starting secondary namenodes [ubuntumaster]
hadoopuser@ubuntumaster:/usr/local/hadoop/sbin$
```

```
hadoopuser@ubuntumaster:~$ hdfs dfs -ls /data/product_reviews/
Found 1 items
-rw-r--r--   1 hadoopuser supergroup    7445886 2025-05-31 10:12 /data/product_reviews/DataProduct.json
hadoopuser@ubuntumaster:~$
```

Objectif de cette phase :

Préparer les données brutes pour analyse avec PySpark, en les plaçant dans un système distribué (HDFS). Cette phase garantit que les données sont accessibles par tous les nœuds du cluster.

Étape 5 : Traitement des données avec PySpark

Objectif : Supprimer les valeurs nulles, doublons, champs inutiles, normaliser le texte (pour NLP).

Dans le cadre de la Phase 2 (« Manipulation et stockage »), nous avons dû :

1. Charger un fichier JSON volumineux (≈ 7 Mo) dans HDFS.
2. Écrire un script PySpark pour lire ce fichier depuis HDFS.
3. Effectuer des opérations de base (filter, groupBy, agg, join) et nettoyer les données.
4. Sauvegarder le résultat traité au format Parquet dans HDFS.

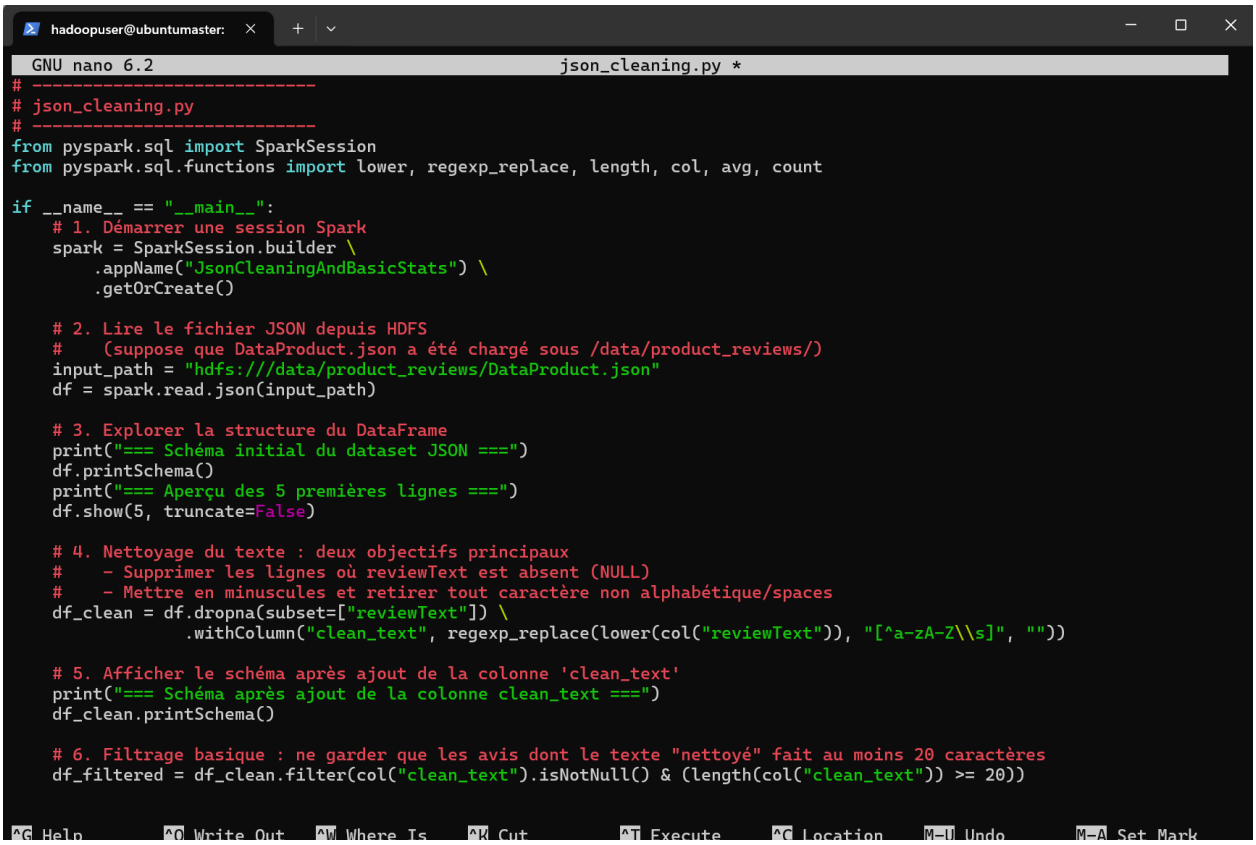
Le jeu de données choisi s'appelle DataProduct.json et contient des avis clients (Amazon reviews) comprenant notamment les champs suivants :

- asin : identifiant produit
- overall : note (notation globale)
- reviewText : texte de l'avis

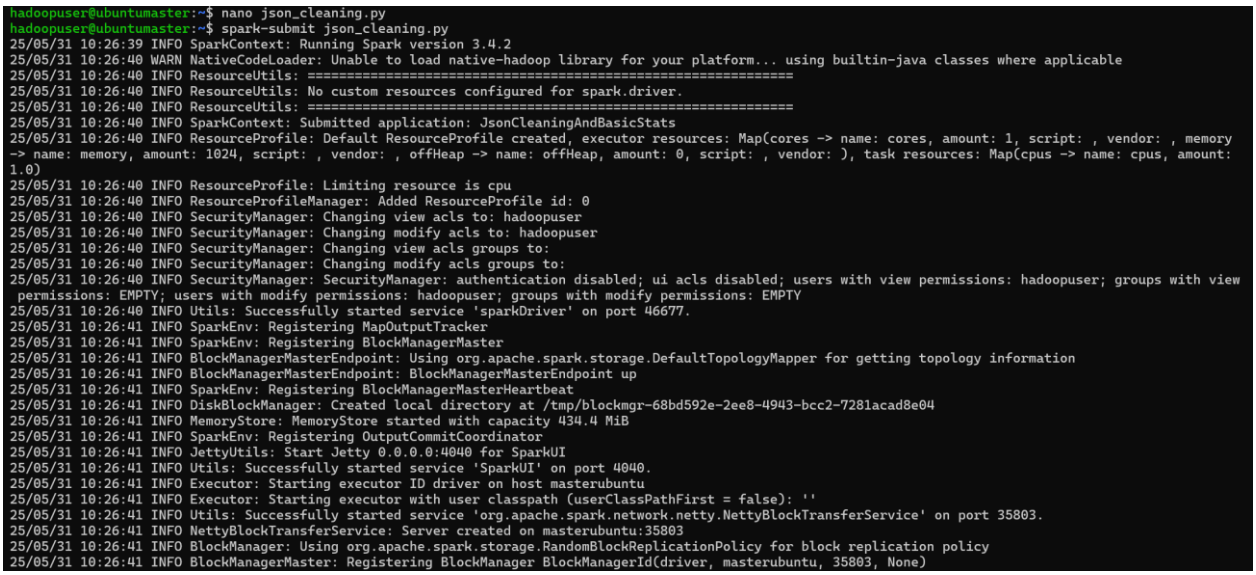
- reviewerID : identifiant du client
- reviewTime : date
- etc.

L'objectif du script est donc de lire ces avis, de :

- Nettoyer le texte (reviewText) pour ne conserver que des caractères alphabétiques (a–z) et des espaces,
- Filtrer les avis trop courts pour que l'analyse textuelle soit pertinente,
- Calculer des statistiques descriptives sur la longueur du texte nettoyé,
- Regrouper (groupBy) les avis par produit (asin) pour calculer, pour chaque produit, le nombre d'avis et la note moyenne,
- Effectuer une jointure (join) pour associer, à chaque avis individuel, les statistiques « nombre d'avis » et « note moyenne » du produit concerné,
- Sauvegarder en sortie deux jeux de résultats au format Parquet :
    1. Les avis nettoyés (avec une colonne clean_text et la longueur text_length).
    2. Les statistiques agrégées par produit (asin, nb_reviews, avg_rating).



Lancement de script avec spark



```
25/05/31 11:01:45 INFO CodeGenerator: Code generated in 7.39936 ms
+----------+--------------+-----------+----------+-----------------+
|asin      |reviewerID    |text_length|nb_reviews|avg_rating       |
+----------+--------------+-----------+----------+-----------------+
|1384719342|A2IBPI20UZIR0U|261        |5         |5.0              |
|1384719342|A14VAT5EAX3D9S|529        |5         |5.0              |
|1384719342|A195EZSQDW3E21|431        |5         |5.0              |
|1384719342|A2C00NNG1ZQQG2|204        |5         |5.0              |
|1384719342|A94QU4C90B1AX |155        |5         |5.0              |
|B00004Y2UT|A2A039TZMZHH9Y|227        |6         |4.666666666666667|
|B00004Y2UT|A1UPZM995ZAH90|184        |6         |4.666666666666667|
|B00004Y2UT|AJNFQI3YR6XJ5 |795        |6         |4.666666666666667|
|B00004Y2UT|A3M1PLEYNDEYO8|195        |6         |4.666666666666667|
|B00004Y2UT|AMNTZU1YQN1TH |214        |6         |4.666666666666667|
+----------+--------------+-----------+----------+-----------------+
only showing top 10 rows
```

Interprétation du tableau après jointure :

Chaque ligne correspond à un avis individuel (identifié par sa colonne reviewerID) associé au produit (asin) qu'il concerne. Les colonnes suivantes détaillent, pour ce même produit :

- **text_length** : la longueur (en caractères) du texte nettoyé de cet avis.
- **nb_reviews** : le nombre total d'avis recensés pour le produit asin (agrégation count(*)).
- **avg_rating** : la note moyenne (champ overall) pour ce produit (agrégation avg(overall)).

Par exemple, on voit que pour le produit 1384719342 (premières lignes) :

- Cet avis a un texte net d'une longueur de 261 caractères.
- Au total, ce produit a 5 avis, et sa note moyenne est de 5,0.

Pour le produit B00004Y2UT (lignes suivantes) :

- Un avis donné a une longueur de 227 caractères.
- Ce produit compte 6 avis au total, avec une note moyenne de 4,6666…

En résumé, cette jointure a « enrichi » chaque avis individuel en lui associant les statistiques globales du produit correspondant (nombre d'avis et note moyenne). Cela permet, par exemple, de filtrer ou de trier les avis non seulement selon leur propre contenu (longueur du texte) mais aussi en fonction de la popularité ou de la qualité générale du produit (via nb_reviews et avg_rating).

Après cette étape, nous pouvons enregistrer le résultat complet (avis nettoyés + statistiques produit) au format Parquet pour une utilisation ultérieure (visualisation, analyses plus avancées, etc.).

```
25/05/31 11:01:43 INFO DAGScheduler: Job 7 finished
25/05/31 11:01:43 INFO CodeGenerator: Code generate
25/05/31 11:01:43 INFO CodeGenerator: Code generate
+----------+----------+------------------+
|asin      |nb_reviews|avg_rating        |
+----------+----------+------------------+
|B003VWJ2K8|162       |4.685185185185185 |
|B0002E1G5C|142       |4.577464788732394 |
|B0002F7K7Y|116       |4.620689655172414 |
|B003VWKPHC|114       |4.649122807017544 |
|B0002H0A3S|92        |4.6521739130434785|
|B0002CZVXM|74        |4.5               |
|B0006NDF8A|71        |4.492957746478873 |
|B0009G1E0K|69        |4.492753623188406 |
|B0002E2KPC|68        |4.5               |
|B0002GLDQM|67        |4.373134328358209 |
+----------+----------+------------------+
only showing top 10 rows
```

Ce tableau présente les **10 produits** (colonne asin) ayant le **plus grand nombre d'avis** (nb_reviews). Pour chacun, on affiche également la **note moyenne** (avg_rating). Par exemple

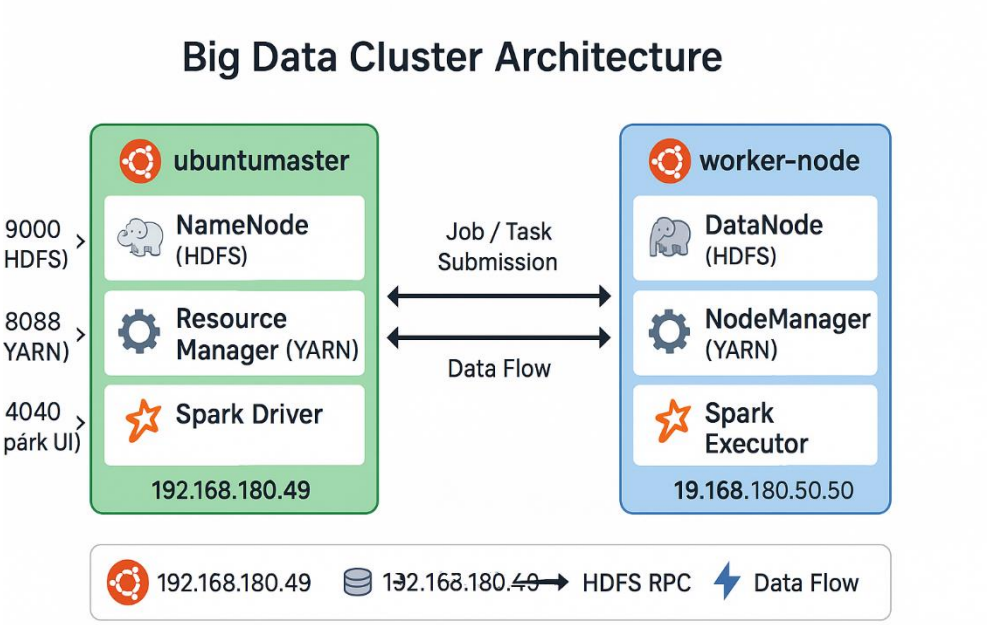
- Le produit B003VWJ2K8 compte **162 avis** et a une note moyenne d'environ **4,685**.
- Le produit B0002E1G5C en compte **142**, avec une note moyenne de **4,577**.

```
25/05/31 11:01:41 INFO DAGScheduler: ResultStage 6 (showString at NativeMethodAccessorImpl.java:0) finished in 0.084 s
25/05/31 11:01:41 INFO DAGScheduler: Job 5 is finished. Cancelling potential speculative or zombie tasks for this job
25/05/31 11:01:41 INFO TaskSchedulerImpl: Killing all running tasks in stage 6: Stage finished
25/05/31 11:01:41 INFO DAGScheduler: Job 5 finished: showString at NativeMethodAccessorImpl.java:0, took 0.089246 s
25/05/31 11:01:41 INFO BlockManagerInfo: Removed broadcast_7_piece0 on masterubuntu:32891 in memory (size: 34.6 KiB, free: 434.3 MiB)
25/05/31 11:01:41 INFO CodeGenerator: Code generated in 9.4378 ms
+----------+--------------+-----------+--------------------------------------------------------------------------------+
|      asin|    reviewerID|text_length|                                                                      clean_text|
+----------+--------------+-----------+--------------------------------------------------------------------------------+
|B000068NSX|A2EZWZ8MBEDOLN|       1513|ive been using these cables for more than  months and they are holding up pre...|
|B0000AQRSS| A2MR43RDPZX3J|       3496|it is so hard to get an honest review of any mike that isnt a shure sm this i...|
|B0000AQRSS|  A6FIAB28IS79|       1472|i started with the shure pg and most recently acquired the shure sm  id proba...|
|B0000AQRST|A2KI91IR3RA7D0|       2369|although this mic is now considered primarily an instrument mic that was not ...|
|B0000AQRST|A34IJACMU8C3IM|       1704|considering that the sm was introduced to the world in  a year after the shur...|
+----------+--------------+-----------+--------------------------------------------------------------------------------+
only showing top 5 rows

25/05/31 11:01:42 INFO BlockManagerInfo: Removed broadcast_8_piece0 on masterubuntu:32891 in memory (size: 16.5 KiB, free: 434.3 MiB)
=== Top 10 produits par nombre d'avis ===
25/05/31 11:01:42 INFO FileSourceStrategy: Pushed Filters: IsNotNull(reviewText)
```

On a fait **Sauvegarde des résultats au format Parquet dans HDFS**

- cleaned_reviews_parquet
- product_stats_parquet

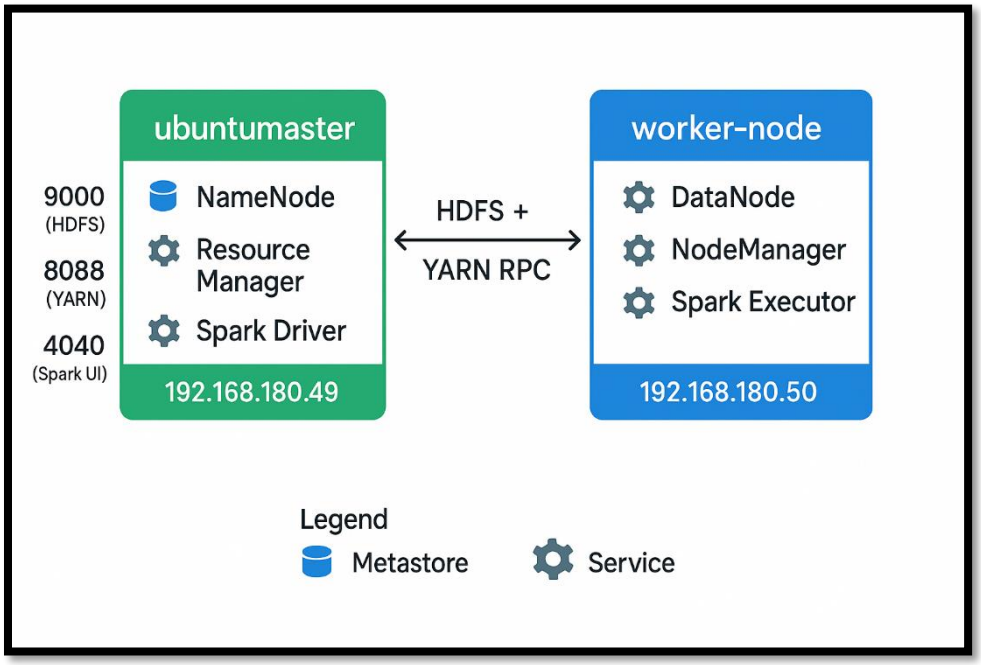
Architecture distribuée du cluster Hadoop/Spark:



Dans l'architecture de notre **cluster Big Data**, l'interaction entre le nœud ubuntumaster et le nœud ubuntuworker repose sur la répartition des responsabilités entre coordination et exécution**. Le nœud maître (ubuntumaster)** agit comme le cerveau du système, assurant la gestion des métadonnées HDFS via le NameNode, la planification des ressources via le **ResourceManager (YARN)** et l'orchestration des tâches Spark grâce au **Spark Driver**. En parallèle, le nœud esclave (ubuntuworker) est dédié à l'exécution : il héberge un DataNode pour stocker physiquement les blocs de données, **un NodeManager** pour exécuter les conteneurs YARN, et un ou plusieurs **Spark Executors** pour effectuer les calculs distribués.

Lorsqu'une commande spark-submit est lancée depuis le master, le **Spark Driver** répartit les tâches à travers les workers **via YARN RPC**. Ces workers lisent les blocs depuis **HDFS** (stockés localement dans leurs DataNodes), traitent les données, puis renvoient les résultats au driver. Ce modèle distribué garantit une scalabilité, une tolérance aux pannes, et une efficacité accrue dans le traitement de gros volumes de données, comme notre fichier **DataProduct.json**. Ainsi, le cluster fonctionne comme une entité unifiée grâce à une communication continue entre le master et ses workers, orchestrée par les couches **Spark et HDFS.**

Architecture de Cluster :

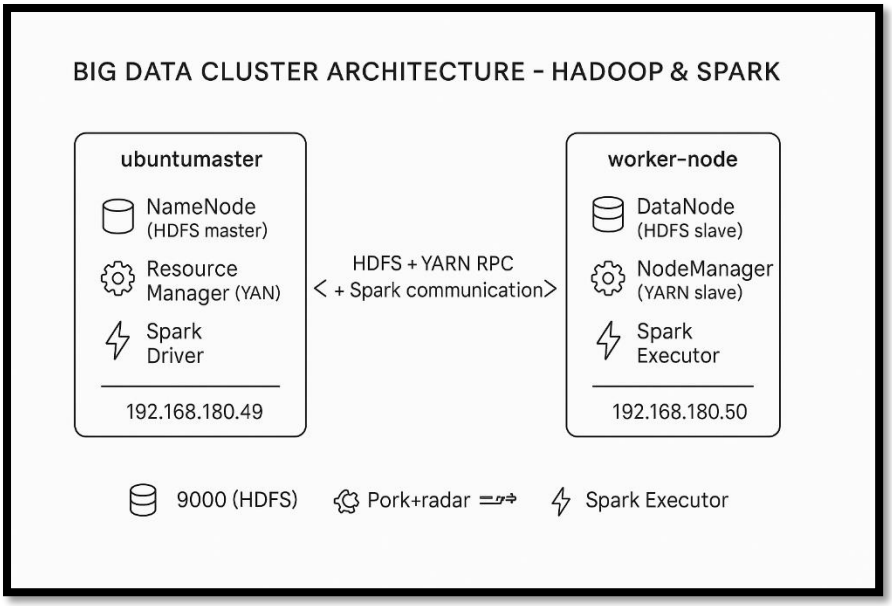


Cette figure illustre l'**architecture du cluster Big Data** déployé pour notre projet. Le nœud **ubuntumaster** (IP : 192.168.180.49) joue le rôle de **maître**, hébergeant trois services essentiels :

- le **NameNode** pour la gestion du système de fichiers HDFS,
- le **ResourceManager** pour la gestion des ressources via YARN,
- et le **Spark Driver** pour le pilotage des tâches Spark.

Le nœud **worker-node** (IP : 192.168.180.50) agit en tant que **nœud esclave**, exécutant :

- un **DataNode** (stockage de blocs HDFS),
- un **NodeManager** (exécution des conteneurs YARN),
- et un **Spark Executor** (exécution des jobs Spark distribués).



Rôles des nœuds dans Notre cluster

❖ **ubuntumaster (Master Node)**

Ce nœud central **coordonne** tout le système :

- **HDFS NameNode** : Gère la structure du système de fichiers (où chaque bloc est stocké)
- **YARN ResourceManager** : Gère les ressources et assigne les tâches aux workers
- **Spark Driver** : Lance et pilote les applications Spark (comme json_cleaning.py)

➢ **Toutes les commandes spark-submit, hdfs dfs, etc., se font ici.**

❖ **ubuntuworker (Worker Node)**

Ce nœud **exécute** les calculs et stocke les données :

- **HDFS DataNode** : Stocke physiquement les blocs de données (JSON, Parquet…)
- **YARN NodeManager** : Exécute les conteneurs pour Spark
- **Spark Executor** : Réalise les transformations, filtrages, agrégations demandées par Spark

➢ C'est ici que **les calculs lourds** sont faits (filter, groupBy, aggregation…) lorsque on lance spark-submit depuis le master.

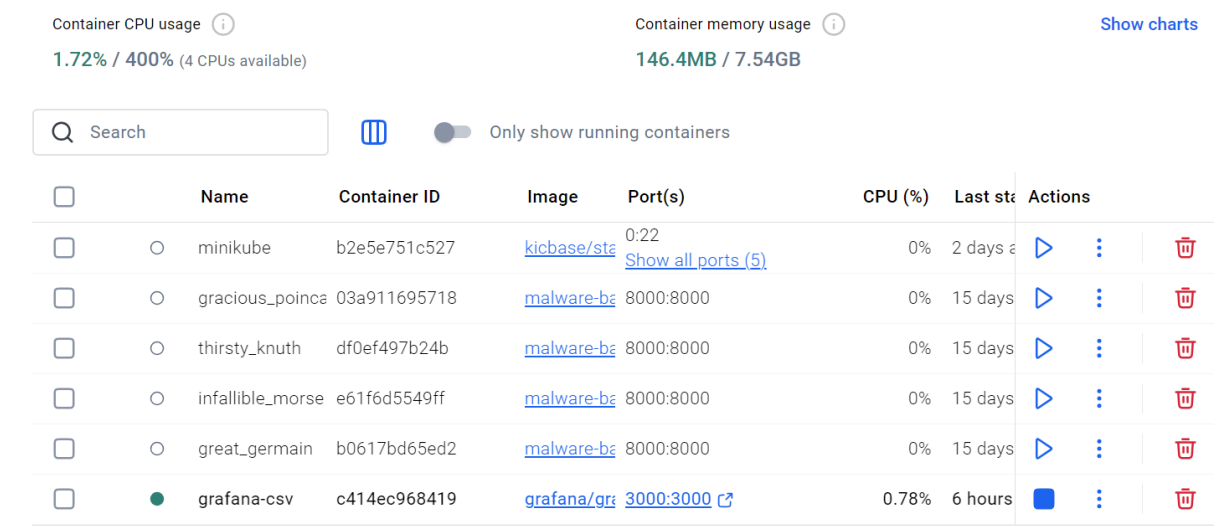
Étape	Fait sur...	Rôle de ubuntuworker
Lancer les services HDFS/YARN	ubuntumaster + worker	Démarrage du DataNode/NodeManager
Copier les données	ubuntumaster	Le DataNode stocke les blocs
spark-submit (nettoyage)	ubuntumaster	Worker exécute les calculs
Sauvegarde Parquet	ubuntumaster	Les fichiers sont écrits via DataNode
Visualisation (Superset)	ubuntumaster	(Pas directement lié au worker)

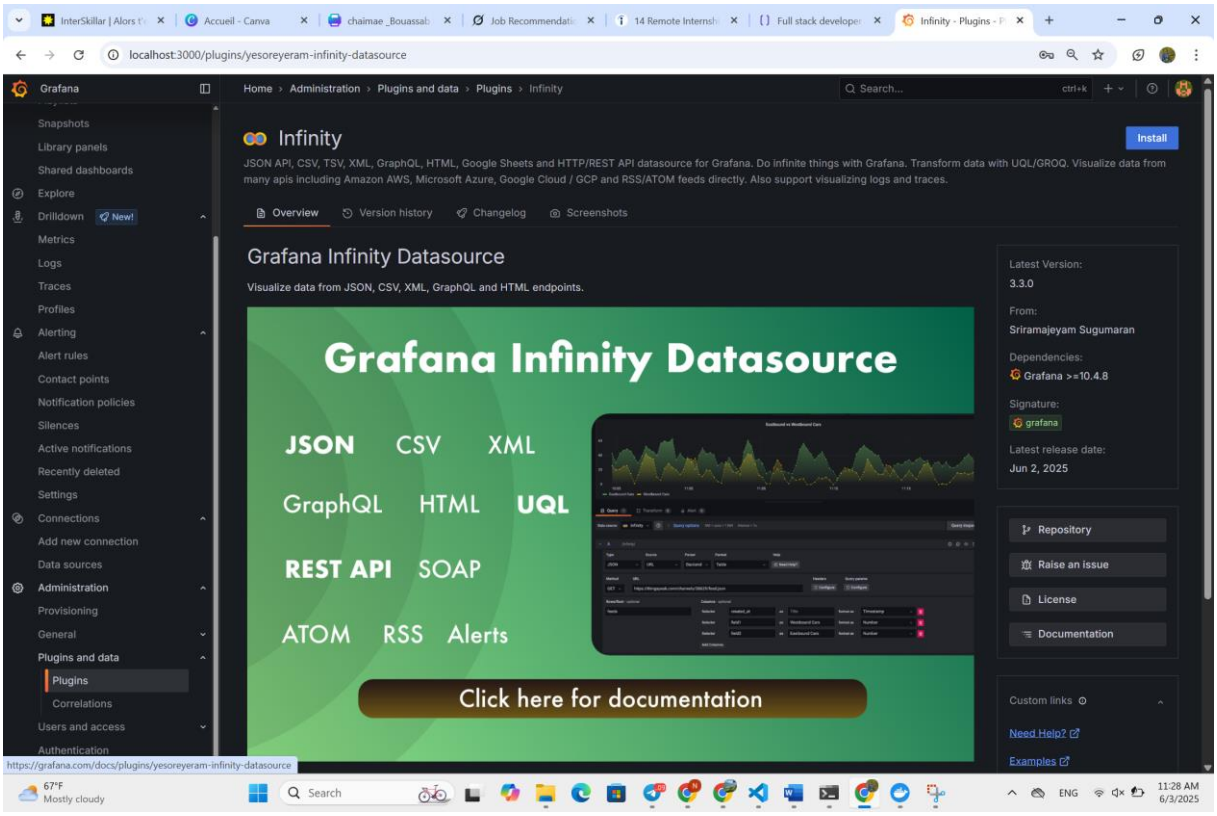
Phase 3 : MINI-CAS D'ANALYSE : avis produits Amazon

Intégration et Visualisation des Données via Grafana Conteneurisé

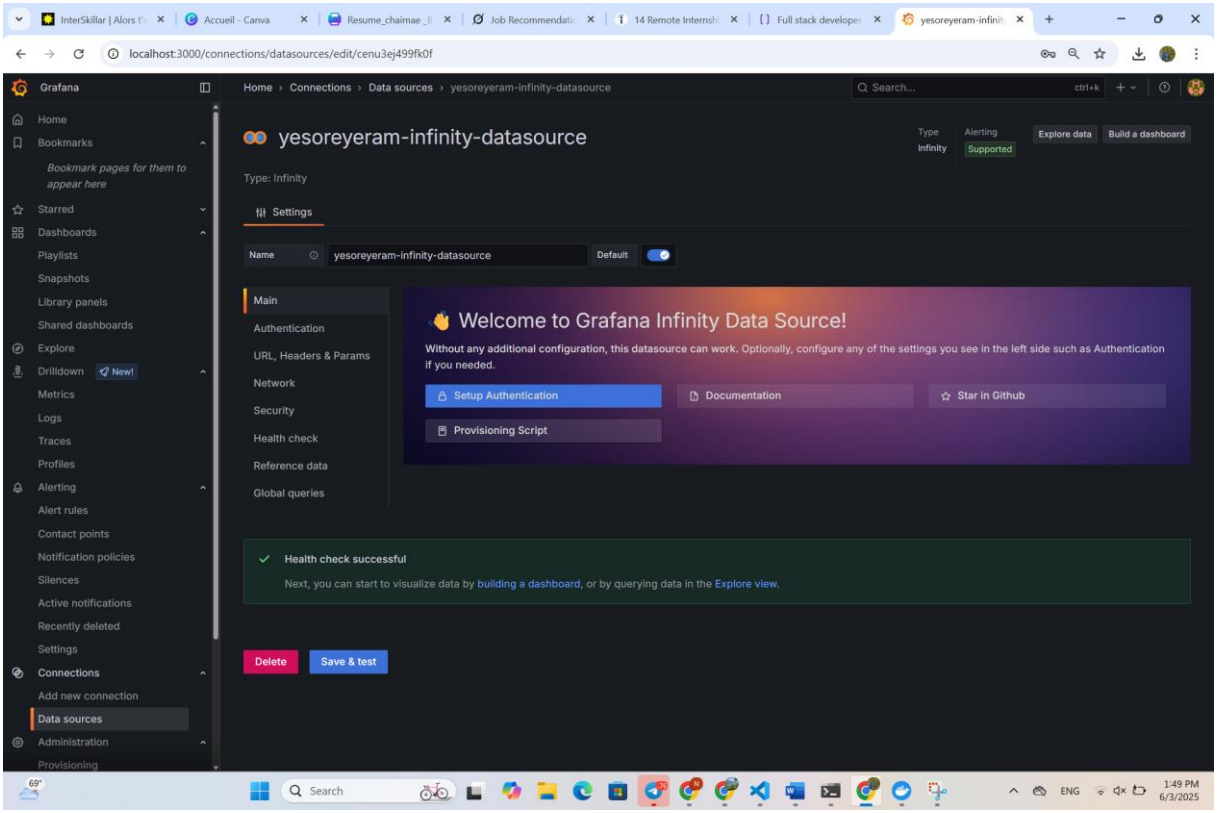
Lancement de Grafana via Docker

```
PS C:\Users\surface> Copy-Item "C:\Users\surface\Desktop\Certifications\prodigy infotech tasks\dataaProducts.csv" "$env:
USERPROFILE\grafana_project\data\"
PS C:\Users\surface> docker run -d -p 3000:3000 `
>>   --name=grafana-csv `
>>   -v "$env:USERPROFILE\grafana_project:/var/lib/grafana" `
>>   -v "$env:USERPROFILE\grafana_project\data:/var/lib/grafana/data" `
>>   grafana/grafana
Unable to find image 'grafana/grafana:latest' locally
latest: Pulling from grafana/grafana
9a8c18aee5ea: Download complete
f18232174bc9: Pull complete
65babbe3dfe5: Pull complete
651b0ba49b07: Pull complete
8b5292c940e1: Downloading [==================================>            ]  49.28MB/63.48MB
d953cde4314b: Pull complete
f836d47fdc4d: Downloading [================================>              ]  74.45MB/107.3MB
454a4350d439: Download complete
aecd4cb03450: Pull complete
13fa68ca8757: Pull complete
```

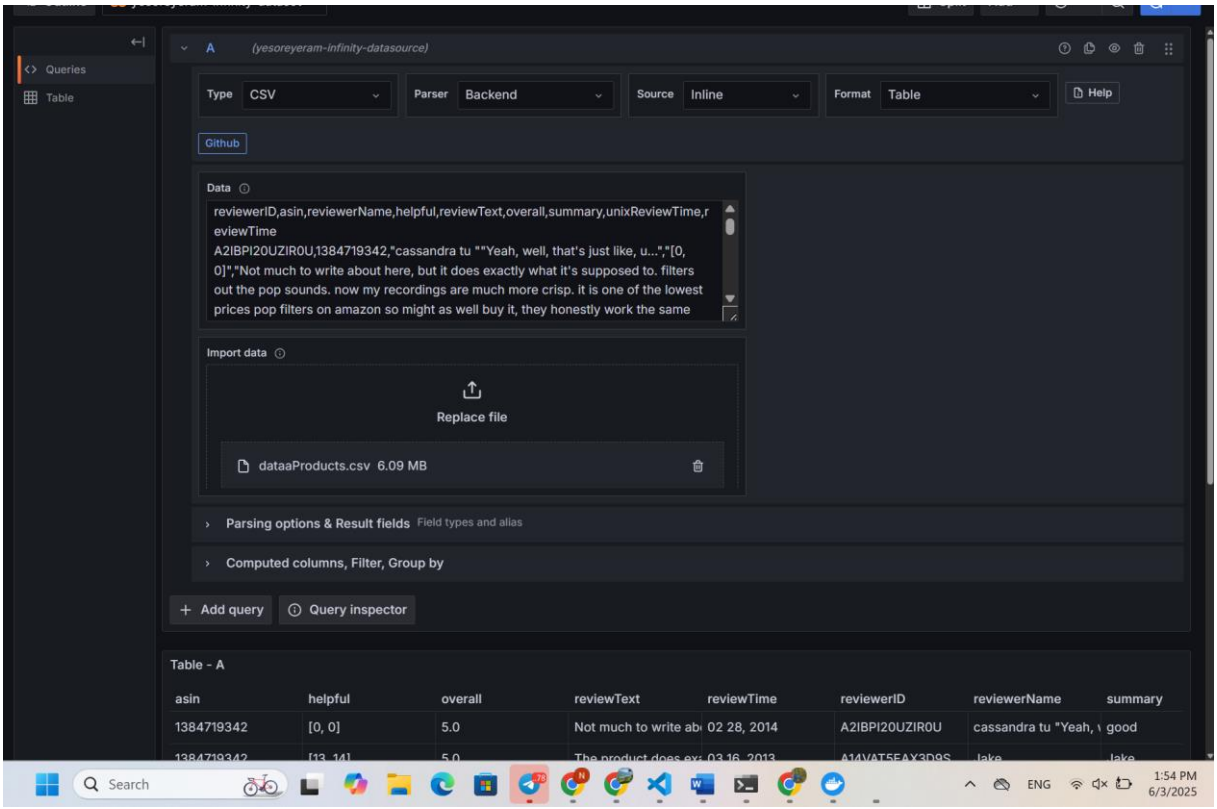




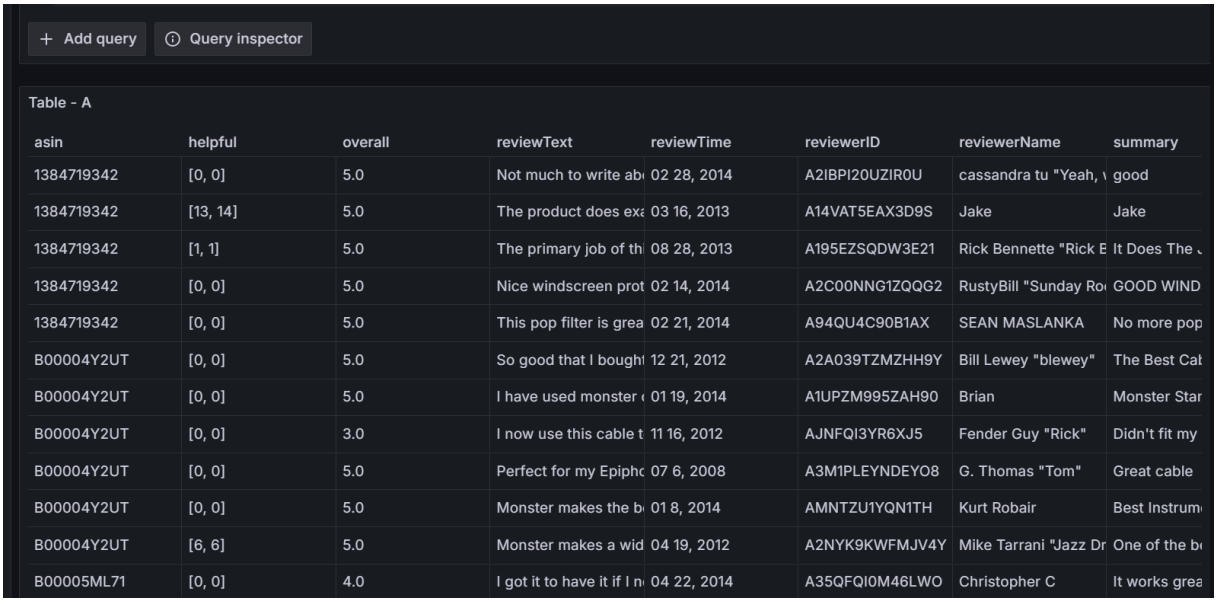
L'installation du **plugin Infinity** via l'interface Grafana. Ce plugin permet de **charger des fichiers CSV, JSON ou XML localement**, ce qui est parfait pour visualiser les données générées précédemment dans Spark.



Après l'analyse sur Spark, nous avons exporté les données nettoyées en CSV. Pour des raisons de simplicité et de performance locale, nous avons utilisé Grafana dans un conteneur Docker. Cela simule l'environnement de production sans avoir besoin de réinstaller tout le cluster Spark ou Ubuntu

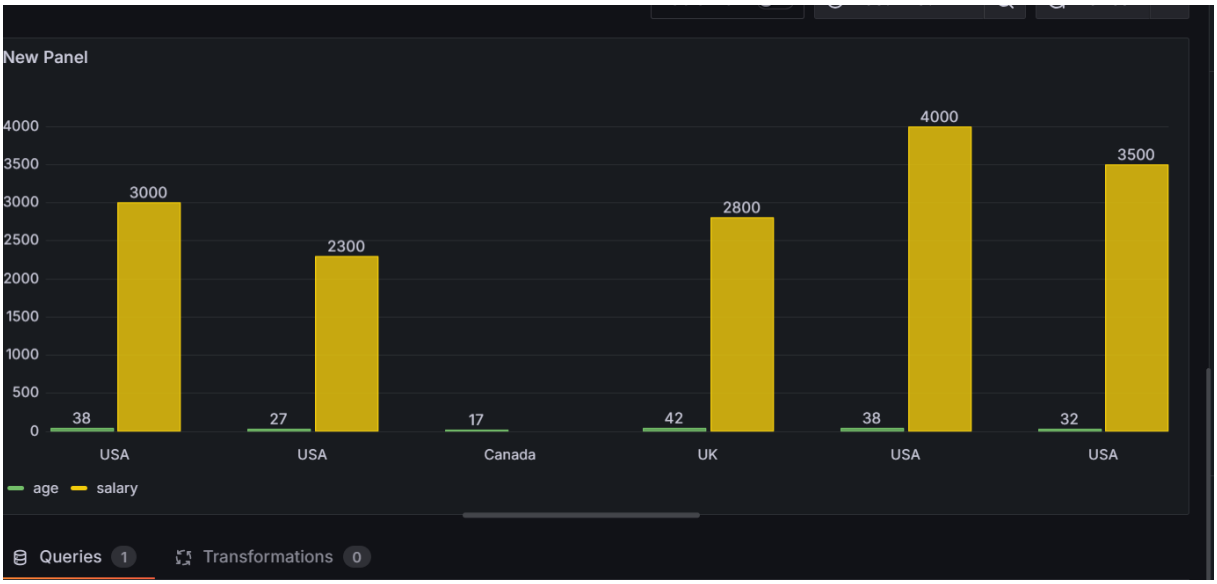


Cela permet à Grafana d'interpréter correctement chaque ligne du fichier CSV, où chaque colonne correspond à un champ comme asin, reviewText, overall, etc.

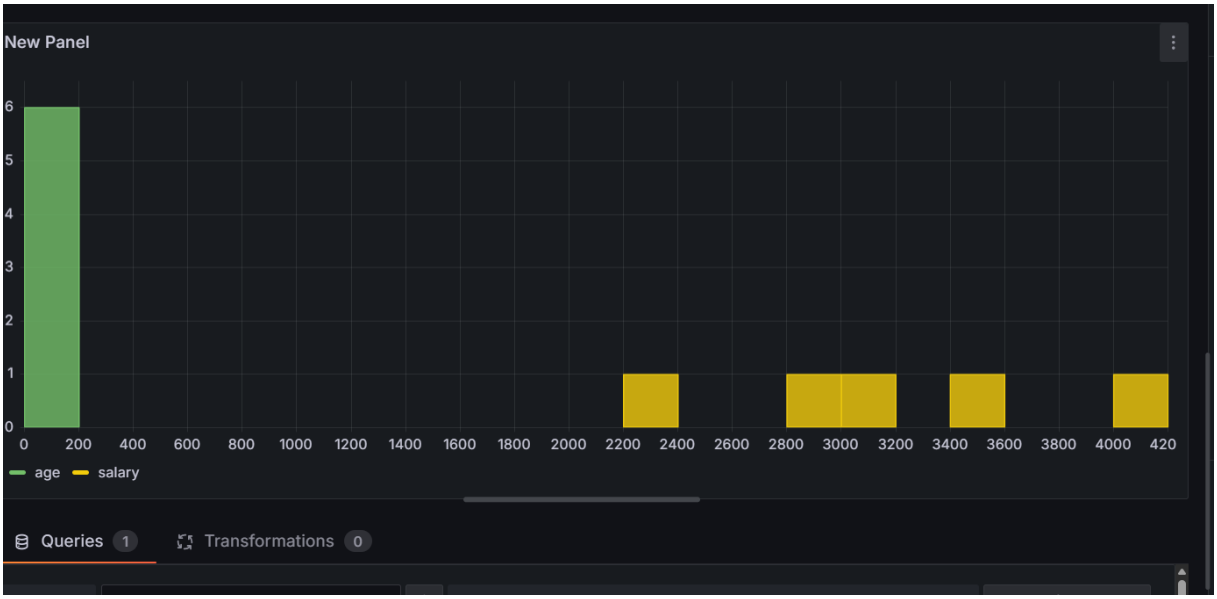


asin	helpful	overall	reviewText	reviewTime	reviewerID	reviewerName	summary
1384719342	[0, 0]	5.0	Not much to write ab	02 28, 2014	A2IBPI20UZIR0U	cassandra tu "Yeah,	good
1384719342	[13, 14]	5.0	The product does ex	03 16, 2013	A14VAT5EAX3D9S	Jake	Jake
1384719342	[1, 1]	5.0	The primary job of th	08 28, 2013	A195EZSQDW3E21	Rick Bennette "Rick E	It Does The
1384719342	[0, 0]	5.0	Nice windscreen prot	02 14, 2014	A2C00NNG1ZQQG2	RustyBill "Sunday Ro	GOOD WIND
1384719342	[0, 0]	5.0	This pop filter is grea	02 21, 2014	A94QU4C90B1AX	SEAN MASLANKA	No more pop
B00004Y2UT	[0, 0]	5.0	So good that I bought	12 21, 2012	A2A039TZMZHH9Y	Bill Lewey "blewey"	The Best Cat
B00004Y2UT	[0, 0]	5.0	I have used monster (	01 19, 2014	A1UPZM995ZAH90	Brian	Monster Star
B00004Y2UT	[0, 0]	3.0	I now use this cable t	11 16, 2012	AJNFQI3YR6XJ5	Fender Guy "Rick"	Didn't fit my
B00004Y2UT	[0, 0]	5.0	Perfect for my Epiphc	07 6, 2008	A3M1PLEYNDEYO8	G. Thomas "Tom"	Great cable
B00004Y2UT	[0, 0]	5.0	Monster makes the b	01 8, 2014	AMNTZU1YQN1TH	Kurt Robair	Best Instrum
B00004Y2UT	[6, 6]	5.0	Monster makes a wid	04 19, 2012	A2NYK9KWFMJV4Y	Mike Tarrani "Jazz Dr	One of the b
B00005ML71	[0, 0]	4.0	I got it to have it if I n	04 22, 2014	A35QFQI0M46LWO	Christopher C	It works grea

Cet affichage valide que **le fichier CSV a été correctement chargé et structuré** dans Grafana, prêt à être utilisé pour la création de panels (graphes, histogrammes, etc.).

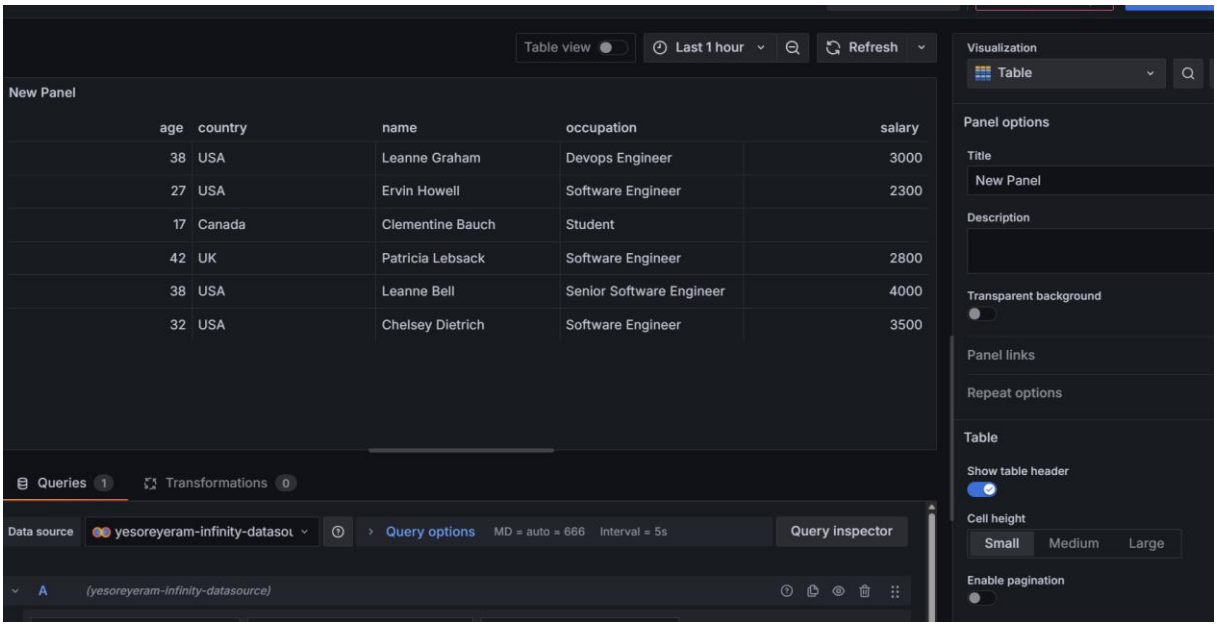


Chaque groupe de barres correspond à un pays (USA, Canada, UK…). Ce type de graphique est utile pour comparer visuellement deux métriques par catégorie. On observe par exemple que les individus les plus âgés (42 ans au Royaume-Uni) ne sont pas nécessairement ceux ayant les salaires les plus élevés (4000 aux États-Unis).

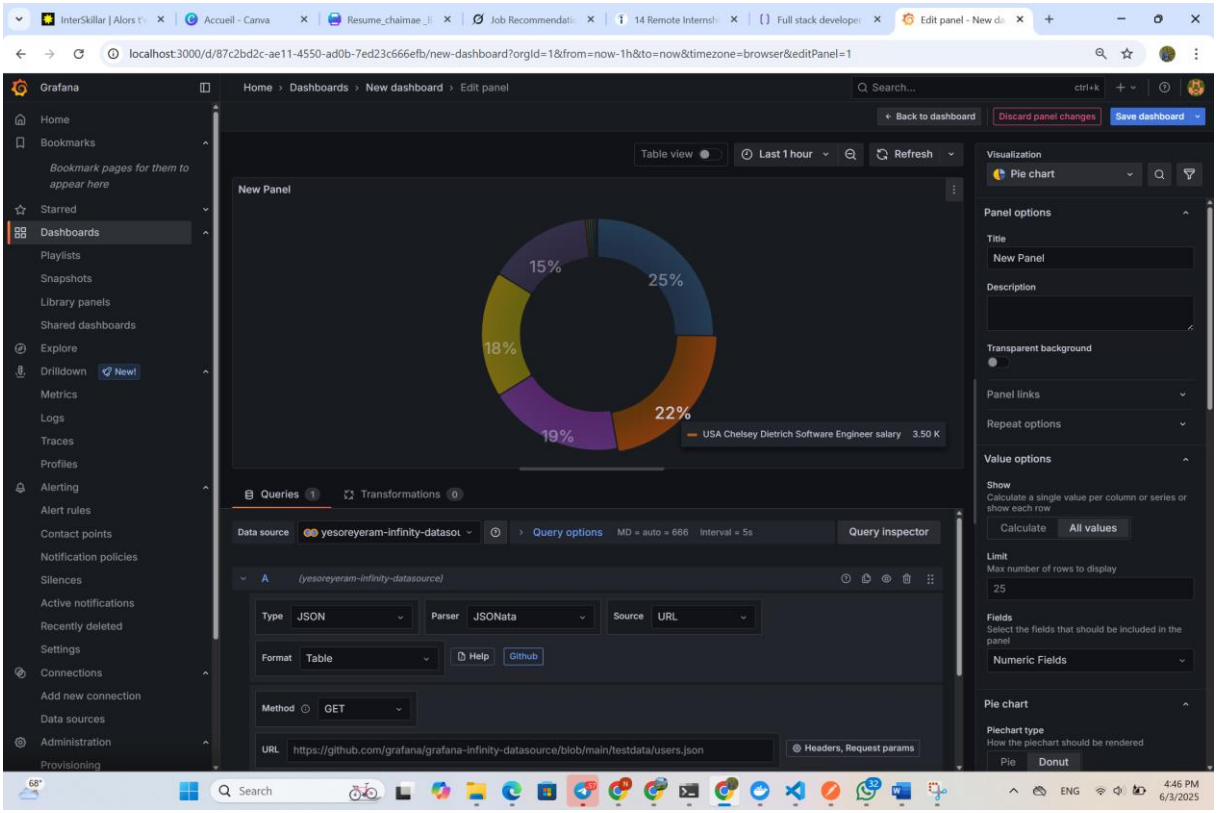


Ce second graphique est un **histogramme** affichant la **distribution des salaires** présents dans le jeu de données. On remarque une forte concentration dans la tranche basse (autour de 200 à 1000), tandis que les tranches supérieures (jusqu'à 4000) contiennent moins de données.

Ce type de visualisation permet de comprendre rapidement la dispersion des valeurs numériques dans un champ continu comme le salaire.



L'objectif de cette visualisation est de **comparer facilement les deux variables** au sein d'une même ligne, pour chaque profil de la base. Par exemple, *Leanne Bell* aux USA a un âge de 38 ans et un salaire de 4000, tandis que *Clementine Bauch*, au Canada, a 17 ans et un salaire de 2800. Ce type de graphique est particulièrement lisible pour des jeux de données de faible volume.



Ce **diagramme circulaire** représente la **répartition relative** des différents profils, ici selon une variable agrégée (salaire ou âge) par personne. Chaque part du cercle correspond à un utilisateur, et la **proportion en pourcentage** permet de voir les écarts facilement.

Le diagramme montre par exemple que certains contributeurs (ex. *Leanne Graham* ou *Chelsey Dietrich*) occupent une part importante de l'ensemble, ce qui peut indiquer une **concentration des valeurs élevées** (âge/salaire).

CONCLUSION

Ce projet a permis de mettre en œuvre une chaîne complète de traitement Big Data, depuis l'ingestion et le stockage de données jusqu'à l'analyse et la visualisation. À travers les différentes phases, nous avons manipulé des outils et technologies modernes comme HDFS, Apache Spark, Docker, et Grafana, dans un environnement distribué simulé sous Ubuntu Server Master/Worker.

Dans un premier temps, nous avons préparé l'environnement Hadoop/Spark, créé les répertoires HDFS, et intégré un jeu de données volumineux au format JSON. Ensuite, à l'aide de PySpark, nous avons effectué des traitements de nettoyage, des agrégations statistiques, et des extractions de colonnes utiles.

Face à la perte des machines virtuelles en fin de parcours, une solution alternative a été mise en place via Docker et Grafana sur Windows, avec succès. Grâce au connecteur Infinity Data Source, nous avons pu intégrer nos fichiers .csv traités dans Grafana et générer des visualisations dynamiques : histogrammes, camemberts, heatmaps, et bar charts.

Ce projet a ainsi démontré la capacité à :

- déployer un cluster Hadoop/Spark local,
- automatiser des traitements de données volumineuses,
- convertir et visualiser des données avec des outils modernes open source.

En somme, ce travail a illustré concrètement le cycle ETL + Analyse + Visualisation dans un contexte Big Data, avec une approche pratique et résiliente face aux contraintes techniques. Il a permis de renforcer nos compétences techniques en traitement distribué, analyse de données, et monitoring via dashboard.