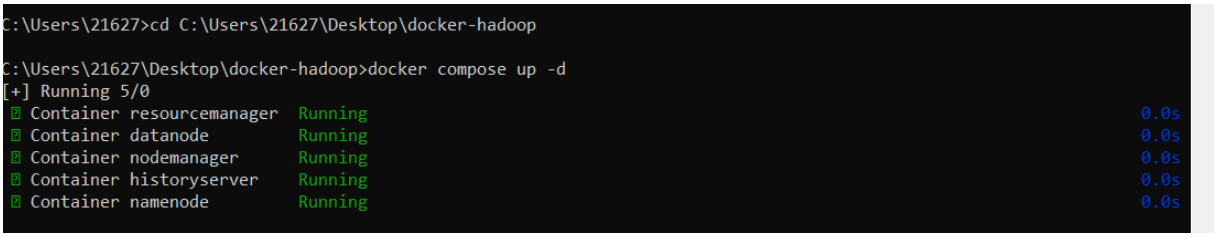
Base de données NOSQL et Big data

TP1 :Inititiation à Hadoop

Partie1 :

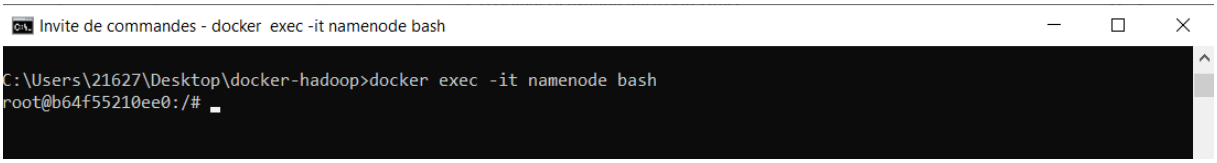
Installation et Configuration

1) Avant d'entamer notre voyage dans l'environnement Dockerisé, nous devons d'abord localiser notre destination, le conteneur Docker situé dans le répertoire http://www.docker.com/. Comme un explorateur déterminé, nous utilisons la commande
cd nom_du_dossier pour ouvrir la porte virtuelle de ce monde numérique. Une fois à l'intérieur, nous sommes prêts à lancer notre vaisseau, notre environnement de développement Dockerisé, à l'aide de la commande **docker-compose up -d**.



2)

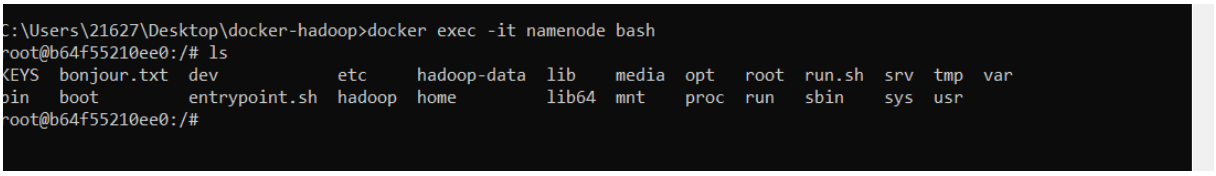
Pour entrer dans l'environnement Docker de notre cluster Hadoop, nous utilisons la commande bash **docker exec -it namenode**. Cette commande ouvre une porte virtuelle directe vers le conteneur namenode, ce qui nous permet d'interagir en temps réel avec notre système distribué. C'est notre point d'accès direct, qui nous permet de naviguer dans notre cluster et de configurer nos analyses avec facilité et efficacité.



3)

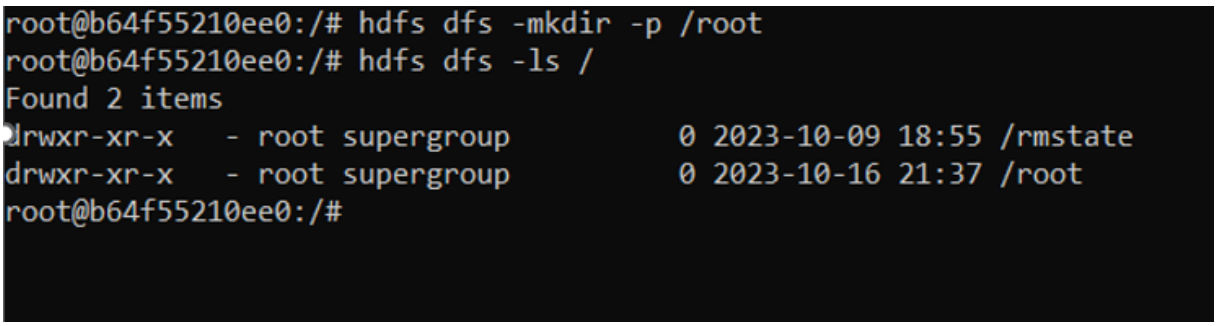
Pour illustrer la simplicité et la puissance de Hadoop, nous avons créé un fichier appelé bonjour.txt dans notre système distribué. Tout d'abord, nous avons utilisé la commande **echo "bonjour hadoop et HDFS">bonjour.txt** pour ajouter le message **" bonjour hadoop et HDFS "** à ce fichier. Ensuite, pour vérifier son existence, nous avons utilisé la commande ls, qui a

affiché la liste des fichiers dans notre système HDFS, confirmant la création réussie de notre fichier



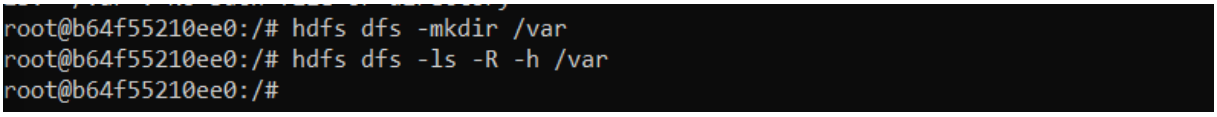
4)

Pour organiser notre système HDFS de manière structurée, nous avons utilisé la commande **hdfs dfs -mkdir -p /root**, créant ainsi un répertoire principal appelé "root". L'option '-p' permet de créer plusieurs niveaux de répertoires à la fois, en s'assurant que chaque sous-répertoire est créé, même s'il n'existe pas encore. Pour confirmer notre action, nous avons utilisé la commande **hdfs dfs -ls /,** qui a affiché une liste de répertoires à la racine de notre système HDFS, y compris le nouveau répertoire 'root'. Cette étape démontre comment Hadoop facilite la gestion des répertoires, permettant une structuration claire et organisée de nos données dans l'écosystème distribué.



5)

Pour ajouter un répertoire nommé "var" à la racine de notre système de fichiers HDFS, nous avons utilisé la commande **hdfs dfs -mkdir /var**. Cette commande crée un répertoire vide nommé "var". Ensuite, pour vérifier la création, la taille et le contenu du répertoire, nous avons exécuté la commande **hdfs dfs -ls -R -h /var**. L'option '-R' liste récursivement les fichiers et les répertoires, tandis que l'option '-h' affiche la taille des fichiers dans un format lisible par l'homme. Cela nous a permis de confirmer la présence du répertoire 'var' et de ses propriétés dans notre système HDFS.



En utilisant la commande **hdfs dfs -mkdir /dossier**, nous avons créé un répertoire appelé 'dossier' à la racine de notre système de fichiers HDFS. Cette simple commande crée un répertoire vide nommé

'dossier', démontrant la facilité avec laquelle Hadoop nous permet de créer des répertoires dans un environnement distribué.

```
root@b64f55210ee0:/# hdfs dfs -mkdir /root/dossier
root@b64f55210ee0:/#
```

Pour localiser le fichier "bonjour.txt" dans notre système de fichiers Hadoop, nous avons utilisé la commande **find / -name bonjour.txt 2>/dev/null.** Cette commande effectue une recherche à partir de la racine ('/') de notre système de fichiers, en identifiant tous les fichiers portant le nom 'bonjour.txt'. L'option '2>/dev/null' est utilisée pour supprimer les erreurs potentielles pouvant survenir au cours de la recherche.

```
root@b64f55210ee0:/# find / -name bonjour.txt 2>/dev/null
/bonjour.txt
```

Pour vérifier si le fichier "bonjour.txt" est vide dans notre système Hadoop, nous avons utilisé une structure de contrôle "**if"** dans un script

```
root@b64f55210ee0:/# if [ -s bonjour.txt ]; then
> echo "le fichier est non vide "
> else
> echo "le fichier est vide"
> fi
le fichier est non vide
```

Pour transférer le fichier "bonjour.txt" de notre système local vers l'environnement Hadoop distribué, nous avons utilisé la commande **hdfs dfs -put bonjour.txt /.** Cette commande place le fichier "bonjour.txt" dans le répertoire racine de notre système de fichiers HDFS.

```
root@b64f55210ee0:/# hdfs dfs -put bonjour.txt /
2023-10-22 17:44:09,872 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted
= false, remoteHostTrusted = false
```

nous avons utilisé la commande **'hdfs dfs -ls -R /'** pour lister de manière exhaustive tous les fichiers et répertoires disponibles à partir de la racine de notre système HDFS.

```
root@b64f55210ee0:/# hdfs dfs  -ls -R /
-rw-r--r--   3 root supergroup         23 2023-10-22 17:44 /bonjour.txt
drwxr-xr-x   - root supergroup          0 2023-10-09 18:55 /rmstate
drwxr-xr-x   - root supergroup          0 2023-10-22 15:57 /rmstate/FSRMStateRoot
drwxr-xr-x   - root supergroup          0 2023-10-09 18:56 /rmstate/FSRMStateRoot/AMRMTokenSecretManager
Root
-rw-r--r--   3 root supergroup         18 2023-10-09 18:56 /rmstate/FSRMStateRoot/AMRMTokenSecretManager
Root/AMRMTokenSecretManagerNode
-rw-r--r--   3 root supergroup          2 2023-10-22 15:57 /rmstate/FSRMStateRoot/EpochNode
drwxr-xr-x   - root supergroup          0 2023-10-09 18:55 /rmstate/FSRMStateRoot/RMAppRoot
drwxr-xr-x   - root supergroup          0 2023-10-22 15:57 /rmstate/FSRMStateRoot/RMDTSecretManagerRoot
-rw-r--r--   3 root supergroup         17 2023-10-16 17:21 /rmstate/FSRMStateRoot/RMDTSecretManagerRoot/
DelegationKey_10
-rw-r--r--   3 root supergroup         17 2023-10-16 20:48 /rmstate/FSRMStateRoot/RMDTSecretManagerRoot/
DelegationKey_11
-rw-r--r--   3 root supergroup         17 2023-10-16 20:48 /rmstate/FSRMStateRoot/RMDTSecretManagerRoot/
DelegationKey_12
-rw-r--r--   3 root supergroup         17 2023-10-17 11:52 /rmstate/FSRMStateRoot/RMDTSecretManagerRoot/
DelegationKey_13
-rw-r--r--   3 root supergroup         17 2023-10-17 11:52 /rmstate/FSRMStateRoot/RMDTSecretManagerRoot/
DelegationKey_14
-rw-r--r--   3 root supergroup         17 2023-10-21 20:58 /rmstate/FSRMStateRoot/RMDTSecretManagerRoot/
DelegationKey_15
-rw-r--r--   3 root supergroup         17 2023-10-21 20:58 /rmstate/FSRMStateRoot/RMDTSecretManagerRoot/
DelegationKey_16
-rw-r--r--   3 root supergroup         17 2023-10-22 14:29 /rmstate/FSRMStateRoot/RMDTSecretManagerRoot/
DelegationKey_17
-rw-r--r--   3 root supergroup         17 2023-10-22 14:29 /rmstate/FSRMStateRoot/RMDTSecretManagerRoot/
DelegationKey_18
-rw-r--r--   3 root supergroup         17 2023-10-22 15:57 /rmstate/FSRMStateRoot/RMDTSecretManagerRoot/
DelegationKey_19
-rw-r--r--   3 root supergroup         17 2023-10-22 15:57 /rmstate/FSRMStateRoot/RMDTSecretManagerRoot/
DelegationKey_20
-rw-r--r--   3 root supergroup         17 2023-10-16 17:21 /rmstate/FSRMStateRoot/RMDTSecretManagerRoot/
DelegationKey_9
-rw-r--r--   3 root supergroup          4 2023-10-09 18:55 /rmstate/FSRMStateRoot/RMVersionNode
drwxr-xr-x   - root supergroup          0 2023-10-09 18:55 /rmstate/FSRMStateRoot/ReservationSystemRoot
drwxr-xr-x   - root supergroup          0 2023-10-22 16:37 /root
drwxr-xr-x   - root supergroup          0 2023-10-22 16:37 /root/dossier
drwxr-xr-x   - root supergroup          0 2023-10-22 16:27 /var
```

Après avoir confirmé la présence du fichier "bonjour.txt" dans le système de fichiers distribués Hadoop à l'aide de la commande **hdfs dfs -ls -R /,** nous avons utilisé la commande **hdfs dfs -cat /bonjour.txt** pour afficher le contenu du fichier. La commande **hdfs dfs -cat** est utilisée pour afficher le contenu d'un fichier dans Hadoop.

```
root@b64f55210ee0:/# hdfs dfs -cat /bonjour.txt
2023-10-22 17:59:22,197 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted
= false, remoteHostTrusted = false
bonjour Hadoop et HDFS
root@b64f55210ee0:/#
```

L'ajout de **| more** à la commande permet de paginer la sortie, ce qui facilite la lecture du contenu, en particulier si le fichier est volumineux. Si vous n'utilisez pas | more, le contenu du fichier sera affiché en continu, ce qui peut être plus difficile à lire si le fichier est très long.

```
root@b64f55210ee0:/# hdfs dfs -cat /bonjour.txt | more
2023-10-22 18:10:16,400 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted
= false, remoteHostTrusted = false
bonjour Hadoop et HDFS
root@b64f55210ee0:/#
```

Pour afficher les dernières lignes du fichier 'bonjour.txt' dans le système de fichiers distribués Hadoop, nous avons utilisé la commande **hdfs dfs -tail /bonjour.txt**. Cette commande spécifique affiche les dernières lignes d'un fichier en temps réel au fur et à mesure de sa mise à jour.

```
root@b64f55210ee0:/# hdfs dfs -tail /bonjour.txt
2023-10-22 18:27:51,003 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted
= false, remoteHostTrusted = false
bonjour Hadoop et HDFS
```

Pour supprimer le fichier "bonjour.txt" du système de fichiers distribués Hadoop, nous avons utilisé la commande **hdfs dfs -rm / bonjour.txt**. Cette commande est essentielle pour une gestion efficace des fichiers dans Hadoop, car elle libère de l'espace de stockage ou supprime les fichiers obsolètes.

```
root@b64f55210ee0:/# hdfs dfs -rm /bonjour.txt
Deleted /bonjour.txt
root@b64f55210ee0:/#
```

Pour restaurer le fichier "bonjour.txt" dans le système de fichiers distribués Hadoop après l'avoir supprimé, nous avons utilisé la commande **hdfs dfs -put bonjour.txt /.** Cette commande est utilisée pour copier un fichier de notre système local vers le système HDFS.

```
root@b64f55210ee0:/# hdfs dfs -put bonjour.txt /
2023-10-22 18:39:33,631 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted
= false, remoteHostTrusted = false
```

À l'aide de la commande **hdfs dfs -chmod go-r /bonjour.txt**, nous avons modifié les autorisations du fichier 'bonjour.txt' dans le système de fichiers distribués Hadoop. Plus précisément, nous avons supprimé les droits de lecture (symbolisés par "r") des groupes (g) et des autres utilisateurs (o).

```
root@b64f55210ee0:/# hdfs dfs -chmod go-r /bonjour.txt
root@b64f55210ee0:/#
```

Avec la commande **hdfs dfs -mv /bonjour.txt /dossier/bonjour.txt**, nous avons déplacé le fichier 'bonjour.txt' du répertoire racine (/) vers un répertoire spécifique appelé 'dossier'. Cette commande vous permet de renommer ou de déplacer des fichiers dans le système de fichiers distribués Hadoop (HDFS).

```
root@b64f55210ee0:/# hdfs dfs -mv /bonjour.txt /dossier/bonjour.txt
root@b64f55210ee0:/#
```

Avec la commande **hdfs dfs -get /dossier/bonjour.txt /dossier/bien.txt**, nous avons téléchargé le fichier "bonjour.txt" depuis le système de fichiers distribués Hadoop (HDFS) et l'avons enregistré localement sous le nom "bien.txt" dans le répertoire "dossier". La commande suivante est utilisée pour récupérer des fichiers de Hadoop vers notre système local

```
root@b64f55210ee0:/# hdfs dfs -get /dossier/bonjour.txt /dossier/bien.txt
2023-10-22 19:55:15,619 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted
= false, remoteHostTrusted = false
```

Avec la commande **hdfs dfs -cp /dossier/bonjour.txt /dossier/bien.txt**, nous avons créé une copie du fichier 'bonjour.txt' dans le Hadoop Distributed File System (HDFS). En spécifiant le chemin source (/dossier/bonjour.txt) et le chemin destination (/dossier/bien.txt), nous avons effectué cette opération avec précision

```
root@b64f55210ee0:/# hdfs dfs -cp /dossier/bonjour.txt /dossier/bien.txt
2023-10-22 20:04:23,645 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted
= false, remoteHostTrusted = false
2023-10-22 20:04:24,296 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted
= false, remoteHostTrusted = false
```

Avec la commande **hdfs dfs -count -h /dossier**, nous avons obtenu des informations statistiques sur le répertoire 'dossier' dans le Hadoop Distributed File System (HDFS). Cette commande fournit un résumé détaillé des fichiers et des répertoires contenus dans le répertoire spécifié. Dans ce cas, elle nous indique qu'il y a 1 dossier, 2 fichiers et un espace total de 46 octets utilisés dans le répertoire 'dossier'. L'option '-h' affiche la taille des fichiers d'une manière lisible par l'homme, offrant ainsi une vue conviviale des statistiques du répertoire 'dossier' dans HDFS.

```
root@b64f55210ee0:/# hdfs dfs -count -h /dossier
           1            2                 46 /dossier
```

Avec la commande **hdfs dfs -rm /dossier/bonjour.txt**, nous avons supprimé le fichier "bonjour.txt" situé dans le répertoire "dossier" du système de fichiers distribués Hadoop (HDFS). Cette commande supprime des fichiers spécifiques dans HDFS, assurant ainsi une gestion efficace des données.

```
root@b64f55210ee0:/# hdfs dfs -rm /dossier/bonjour.txt
Deleted /dossier/bonjour.txt
root@b64f55210ee0:/# hdfs dfs -ls /dossier
Found 1 items
-rw-r--r--   3 root supergroup         23 2023-10-22 20:04 /dossier/bien.txt
```

Creation d’une arborescence et telechargement de fichier

1)a)creation de dossier parent TPs

```
root@b64f55210ee0:/# hdfs dfs -mkdir /TPs
```

b)creation des sous dossiers codes et data

```
root@b64f55210ee0:/# hdfs dfs -mkdir /TPs/data
root@b64f55210ee0:/# hdfs dfs -mkdir /TPs/codes
```

2)telechargement de fichier purchases.txt

Methode1

Etape1

Téléchargement du fichier avec curl : La commande curl -o purchases.txt http://bitly.ws/WT8K est utilisée pour récupérer le fichier purchases.txt à partir de l'URL source.

```
C:\Users\21627\Desktop\docker-hadoop>curl -o purchases.txt http://bitly.ws/WT8K
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   243  100   243    0     0    835      0 --:--:-- --:--:-- --:--:--   849
```

Etape2

Copier le fichier dans le conteneur namenode : La commande docker cp purchases.txt

```
C:\Users\21627\Desktop\docker-hadoop>docker cp purchases.txt namenode:/purchases.txt
Successfully copied 2.05kB to namenode:/purchases.txt
```

Etape3

Accès au shell du conteneur namenode : La commande bash **docker exec -it namenode_container** permet d'accéder au shell du conteneur namenode, créant ainsi un environnement de travail à l'intérieur du conteneur.

```
C:\Users\21627\Desktop\docker-hadoop>docker exec -it namenode bash

root@b64f55210ee0:/#
```

Etape4

Copier le fichier du conteneur vers HDFS : Dans le conteneur, la commande **hdfs dfs -copyFromLocal /purchases.txt /data/purchases.txt** est utilisée pour copier le fichier purchases.txt du conteneur vers le dossier data dans HDFS. Assurez-vous au préalable que le répertoire de destination /data dans HDFS existe.

```
root@b64f55210ee0:/# hdfs dfs -copyFromLocal /purchases.txt /data/purchases.txt
2023-10-22 21:30:16,565 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localH
ostTrusted = false, remoteHostTrusted = false
root@b64f55210ee0:/#
```

Methode2

Etape1 :

Téléchargez le fichier localement sur votre machine

Cette commande utilise curl pour télécharger le fichier purchases.txt à partir de l'URL spécifiée et l'enregistre localement sur votre machine.

```
C:\Users\21627\Desktop\docker-hadoop>curl -o purchases.txt http://bitly.ws/WT8K
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   243  100   243    0     0    557      0 --:--:-- --:--:-- --:--:--   561
```

Etape2

Copier le fichier dans le conteneur namenode :

Ici, docker cp copie le fichier purchases.txt de votre machine locale vers le conteneur namenode. Veillez à remplacer namenode par le nom ou l'ID du conteneur namenode.

```
C:\Users\21627\Desktop\docker-hadoop>docker cp purchases.txt namenode:/purchases.txt
Successfully copied 2.05kB to namenode:/purchases.txt
```

Etape3

Accéder au shell du conteneur namenode :

La commande docker exec permet d'accéder à un shell interactif à l'intérieur du conteneur namenode.

```
C:\Users\21627\Desktop\docker-hadoop>docker exec -it namenode bash

root@b64f55210ee0:/#
```

Etape4

Copier le fichier conteneur dans HDFS :

Cette commande hdfs dfs -copyFromLocal copie le fichier purchases.txt depuis le système local du conteneur vers le dossier de données HDFS.

```
root@b64f55210ee0:/# hdfs dfs -copyFromLocal /purchases.txt /data/purchases.txt
2023-10-22 22:21:14,073 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localH
ostTrusted = false, remoteHostTrusted = false
```

Methode3

Etape1

Téléchargez le fichier purchases.txt sur votre ordinateur en cliquant sur le lien : purchases.txt.

Utilisez la commande docker cp pour copier le fichier du système local vers le conteneur namenode :

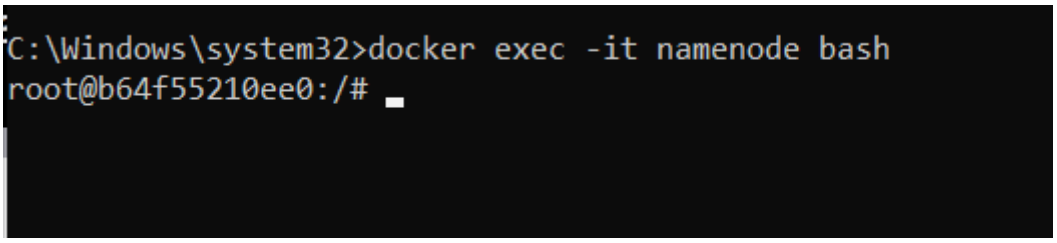
Assurez-vous que le fichier purchases.txt se trouve dans le même répertoire que votre commande terminal/PowerShell.

```
C:\Users\21627\Desktop\docker-hadoop>docker cp purchases.txt namenode:/purchases.txt
Successfully copied 2.05kB to namenode:/purchases.txt
```

Etape2

Accédez au shell du conteneur namenode et déplacez le fichier dans le dossier de données HDFS :

Cette commande copie le fichier purchases.txt du système local du conteneur vers le dossier de données HDFS.




```
root@b64f55210ee0:/# hdfs dfs -copyFromLocal /purchases.txt /data/purchases.txt
2023-10-22 22:34:52,838 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted
= false, remoteHostTrusted = false
root@b64f55210ee0:/#
```

3) depuis votre machine hote telecharger le fichier "pg4300.txt" du roman "Ulysse" disponible sur "http://bitly.ws/WTKB directemment sur le noeud namenode:

Etape1

Télécharger le fichier à partir de l'URL :

Cette commande récupère le fichier pg4300.txt pour le roman "Ulysse" directement à partir du nœud namenode dans le cluster Hadoop.

```
C:\Windows\system32>docker exec namenode curl -o /pg4300.txt http://bitly.ws/WTKB
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   243  100   243    0     0    225      0  0:00:01  0:00:01 --:--:--   225

C:\Windows\system32>
```

Etape2

Copier le fichier dans vers le conteneur Namenode

Utilisez la commande docker cp pour copier le fichier téléchargé dans le conteneur Namenode. Veillez à remplacer container_id par l'ID du conteneur Namenode.

```
C:\Windows\system32>docker exec namenode hdfs dfs -copyFromLocal /pg4300.txt /data/pg4300.txt
2023-10-22 22:55:34,283 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted
= false, remoteHostTrusted = false

C:\Windows\system32>
```

4) Depuis le conteneur namenode telecharger le fichir "pg135" du roman "les miserable" disponibke a l'URL http://bitly.ws/WTKU placer ensuite ce fichier sous le dossier data de HDFS

Etape :

Téléchargement du fichier avec Curl :

Utilisez curl directement dans le conteneur Namenode pour télécharger le fichier "pg135.txt" pour le roman "Les Misérables" à partir de l'URL fournie. La commande curl -O télécharge le fichier et le stocke sous le même nom dans le répertoire actuel.

```
C:\Windows\system32>
C:\Windows\system32>docker exec namenode curl -o /PG135.txt http://bitly.ws/WTKU
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   243  100   243    0     0     16      0  0:00:15  0:00:14  0:00:01    66
```

Etape 2

Copier le fichier dans le dossier HDFS 'data' :

Après avoir téléchargé le fichier avec curl, utilisez la commande hdfs dfs -copyFromLocal pour copier le fichier depuis le système local du conteneur vers le dossier HDFS 'data'.

```
C:\Windows\system32>docker exec namenode hdfs dfs -copyFromLocal /PG135.txt  /data/PG135.txt
2023-10-22 23:25:34,584 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted
= false, remoteHostTrusted = false
```

5)Utiliser les commande HDFS pour lister le contenu du repertoir « data » sur HDFS ainsi pour afficher les information sur les fichiers téléchargés

Pour vérifier les fichiers téléchargés dans le répertoire 'data' du Hadoop Distributed File System (HDFS), nous avons utilisé deux commandes HDFS spécifiques. La première commande, docker exec namenode hdfs dfs -ls /data, nous permet de lister le contenu du répertoire 'data' sur HDFS. Cette commande affiche les fichiers et répertoires présents dans 'data', ce qui nous donne un aperçu clair de son contenu.

```
C:\Windows\system32>docker exec namenode hdfs dfs -ls /data
Found 3 items
-rw-r--r--   3 root supergroup        243 2023-10-22 23:25 /data/PG135.txt
-rw-r--r--   3 root supergroup        243 2023-10-22 22:55 /data/pg4300.txt
-rw-r--r--   3 root supergroup        243 2023-10-22 22:34 /data/purchases.txt
```

La deuxième commande, docker exec namenode hdfs dfs -stat /data/PG135.txt, nous fournit des informations détaillées sur le fichier 'PG135.txt' situé dans le répertoire 'data'. En utilisant l'option '-stat', cette commande nous donne des détails tels que la taille du fichier, le propriétaire, le groupe et les permissions.

```
C:\Windows\system32>docker exec namenode hdfs dfs -stat   /data/PG135.txt
2023-10-22 23:25:34

C:\Windows\system32>
```