TP 4 : Hadoop Distributed File System (HDFS)

Objectifs du TP

— Initiation au framework Hadoop : HDFS

— Utilisation de Docker pour lancer un cluster Hadoop

Outils et versions

— Apache Hadoop : **3.3.6**

— Docker

Docker et Hadoop

Pour déployer le framework Hadoop, nous utiliserons des conteneurs Docker. Cette approche simplifie considérablement la configuration par rapport à une installation native. Elle présente également l'avantage d'être plus légère et plus rapide à exécuter qu'une machine virtuelle classique. Dans ce TP, nous allons mettre en place un cluster Hadoop distribué en utilisant cinq conteneurs Docker, chacun représentant un composant clé du système :

— NameNode : nœud maître de HDFS, chargé de gérer les métadonnées du système de fichiers distribués, telles que la structure des répertoires, l'emplacement des blocs et les permissions. Il exécute également le ResourceManager de YARN, responsable de l'orchestration des ressources et de la planification des tâches.

— DataNodes (2) : nœuds esclaves chargés de stocker les blocs de données HDFS. Ils exécutent également des tâches YARN via le NodeManager. Chaque DataNode communique régulièrement avec le NameNode pour signaler son état et transmettre des informations sur les blocs stockés.

— Secondary NameNode : service auxiliaire chargé de créer périodiquement des points de contrôle (checkpoints) en fusionnant les fichiers EditLog et FsImage. Cela permet d'assurer la cohérence des métadonnées et de faciliter la reprise après panne du NameNode.

— Client HDFS : conteneur dédié permettant d'interagir avec le cluster en utilisant les commandes en ligne (hdfs dfs -put, hdfs dfs -ls, etc.). Ce conteneur simulera l'application cliente qui envoie des fichiers vers le cluster ou lit des données depuis celui-ci.

L'image Docker utilisée dans ce TP est :

```
cjj2010/hadoop:3.3.6
```

Cette image, basée sur Hadoop 3.3.6, est préconfigurée pour exécuter les principaux composants du framework, notamment : - HDFS : NameNode, DataNode, Secondary NameNode - YARN : ResourceManager, NodeManager Elle simplifie considérablement le déploiement du cluster en regroupant toutes les fonctionnalités nécessaires dans une seule image docker.

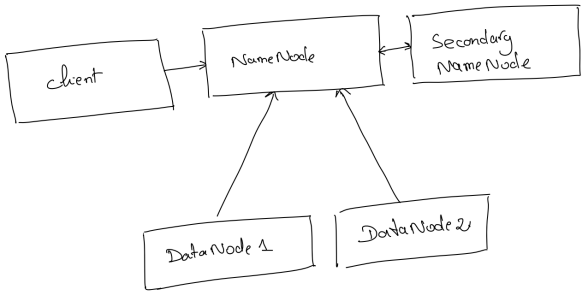


FIGURE 1 – conteneurs docker

Préparation de l'arborescence du projet

— Avant de déployer notre cluster Hadoop avec Docker, il est recommandé de préparer une **structure de dossiers claire et organisée**. Commencez par créer le repertoire du travail hadoop-tp.

```
mkdir -p "$HOME/hadoop-tp"
```

Sous Windows, utilisez de préférence Git Bash pour exécuter les commandes ci-dessus.

— Connectez-vous à Arche et télécharger les deux fichiers **docker-compose.yml** et **config-site.xml** et placez-les dans le dossier **hadoop-tp**.

— Créez un répertoire nommé **data** à l'intérieur du dossier **hadoop-tp**, puis téléchargez le fichier **purchases.txt** et placez-le dans ce répertoire

L'arborescence attendue :

```
hadoop-tp/
|-- docker-compose.yml
|-- config-site.xml
|-- data
    |-- purchases.txt
```

Déployer le Cluster

Ouvrez un terminal et placez-vous dans le **dossier de travail** (celui qui contient `docker-compose.yml`). Puis lancez Docker Compose.

```
docker compose up -d
```

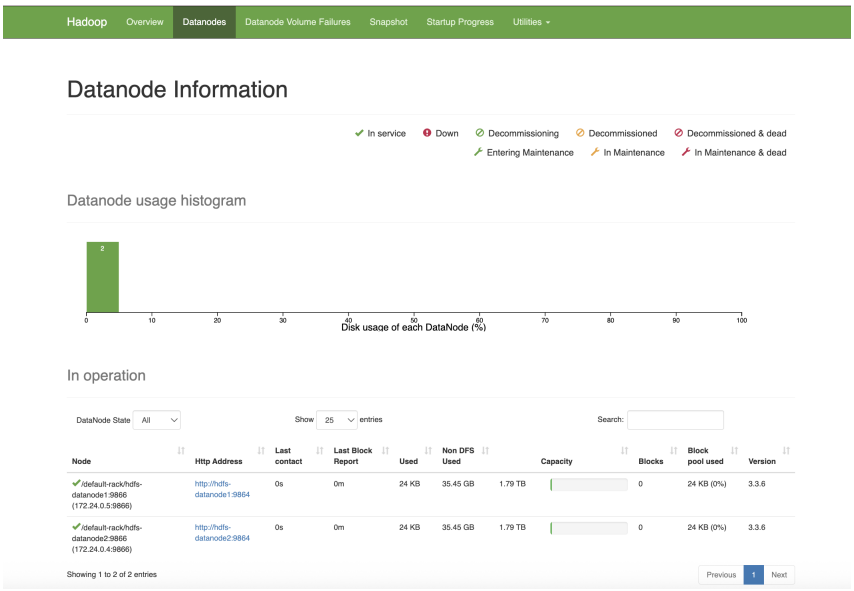
Vérifiez que les conteneurs tournent.

```
docker ps
```

Interfaces web pour Hadoop

Une fois le cluster lancé et prêt à l'emploi, ouvrez votre navigateur et accédez à : http://localhost:9870.

Actuellement, vous devez observer deux DataNodes actifs ; aucun bloc de données n'est encore enregistré sur HDFS après le démarrage.



Chargement des données sur HDFS via le client

Nous allons maintenant transférer le fichier **purchases.txt** vers le système de fichiers distribué Hadoop (HDFS). Pour cela, nous utiliserons le conteneur client (`hdfs-client`) qui agit comme une application externe interagissant avec le cluster Hadoop.

Accès au conteneur client

Le client est un conteneur Docker dédié qui dispose de tous les outils nécessaires pour interagir avec HDFS.

Ouvrez un terminal à l'intérieur de ce conteneur à l'aide de la commande suivante :

```
docker exec -it hdfs-client bash
```

Cette commande permet d'ouvrir un shell interactif (`bash`) dans le conteneur nommé `hdfs-client`. À partir de ce terminal, nous pourrons exécuter toutes les commandes Hadoop.

Création d'un répertoire sur HDFS

Avant de transférer notre fichier, il est recommandé de créer un répertoire dans HDFS pour y organiser nos données. Par exemple, nous allons créer un répertoire nommé `/data` :

```
hdfs dfs -mkdir /data
```

— `hdfs dfs` : outil en ligne de commande pour interagir avec HDFS.

— `-mkdir` : crée un répertoire dans HDFS.

Transfert du fichier vers HDFS

Le fichier `purchases.txt` est actuellement situé dans le dossier `/data` du conteneur client (monté depuis votre machine hôte). Nous allons maintenant le copier dans HDFS à l'aide de la commande `-put` :

```
hdfs dfs -put /data/purchases.txt /data/
```

— `-put` : copie un fichier depuis le système de fichiers local du conteneur vers HDFS.

— `/data/purchases.txt` : chemin source du fichier sur le conteneur client.

— `/data/` : répertoire de destination dans HDFS.

Nous avons ainsi transféré un fichier depuis notre système local vers HDFS à l'aide du conteneur client. Ce processus illustre le fonctionnement fondamental d'HDFS : le client contacte le NameNode pour obtenir les informations nécessaires, puis transfère les blocs vers les DataNodes, qui les stockent de manière répartie et redondante.

Vérification de l'écriture des blocs sur les DataNodes

Une fois le fichier transféré sur HDFS, il est important de vérifier que les blocs ont bien été distribués et stockés sur les différents *DataNodes*. Hadoop fournit une interface web permettant de visualiser l'état du système de fichiers et d'obtenir des informations détaillées sur la répartition des données.

Accès à l'interface Web HDFS

Ouvrez votre navigateur web et accédez à l'interface d'administration du *NameNode* à l'adresse suivante :

`http://localhost:9870`

Dans cette interface, plusieurs onglets sont disponibles pour surveiller l'état du cluster. Cliquez sur l'onglet `Datanodes`.

Analyse des informations affichées

Sur cette page, vous pouvez observer pour chaque *DataNode* les informations suivantes :

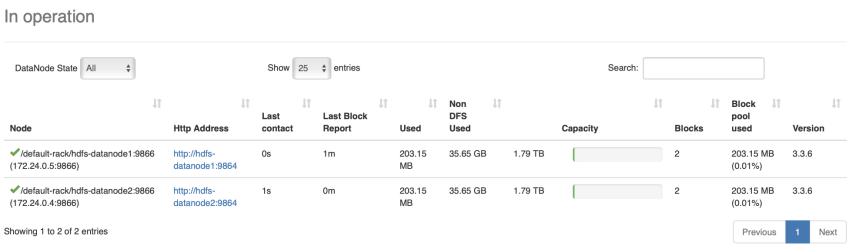
— **Number of Blocks** : nombre total de blocs stockés sur ce nœud.

— **Used** : espace disque utilisé pour les données HDFS.

— **Capacity** : capacité totale du disque disponible.

— **Last Contact** : dernière communication entre le *DataNode* et le *NameNode*.

Le champ **Number of Blocks** est particulièrement important : il vous permet de vérifier que votre fichier a bien été découpé en plusieurs blocs et répliqué sur les différents nœuds du cluster.



Vérification des fichiers `FsImage` et `EditLogs`

Le `FsImage` et les `EditLogs` sont deux fichiers essentiels au fonctionnement du NameNode dans Hadoop. Ils contiennent les métadonnées décrivant l'état global du système de fichiers HDFS :

— **FsImage** : il s'agit d'un instantané complet de l'arborescence HDFS à un moment donné (répertoires, fichiers, permissions, propriétaires, etc.).

— **EditLogs** : ces fichiers contiennent l'historique des opérations récentes effectuées sur HDFS (créations, suppressions, changements de permissions, etc.).

Localisation des fichiers sur le NameNode

Par défaut, ces fichiers se trouvent dans le répertoire suivant à l'intérieur du conteneur `namenode` :

```
/data/dfs/name/current/
```

Vous pouvez lister leur contenu à l'aide de la commande suivante :

```
docker exec -it namenode ls -l /data/dfs/name/current
```

Annexe : commandes utiles HDFS

Voici quelques commandes fréquemment utilisées (liste à compléter selon vos besoins) :

— `hadoop dfs -ls /` : lister les fichiers et répertoires à la racine HDFS.

— `hadoop dfs -mkdir /chemin/dossier` : créer un répertoire.

— `hadoop dfs -put source destination` : copier depuis le système local vers HDFS.

— `hadoop dfs -get source destination` : copier depuis HDFS vers le système local.

— `hadoop dfs -cat /chemin/fichier` : afficher le contenu d'un fichier.

— `hadoop dfs -tail /chemin/fichier` : afficher la fin d'un fichier.

— `hadoop dfs -rm [-r] /chemin` : supprimer un fichier (ou un répertoire avec `-r`).