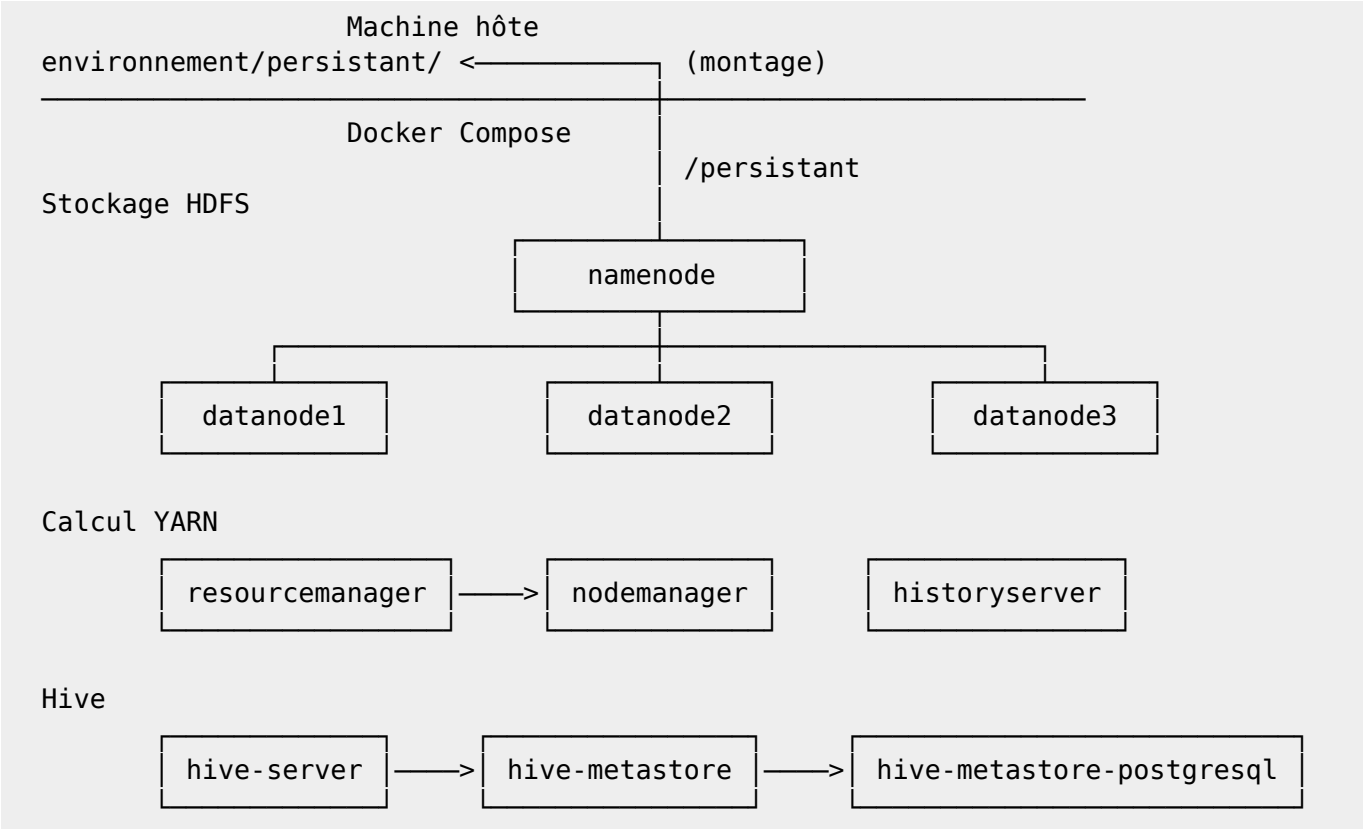
Environnement du projet : cluster Hadoop sur Docker Compose

Le dossier `environnement/` fournit le cluster de départ du projet MarchéPlus. Il reprend celui du TD1, avec Hive et Python en plus. Ce document décrit ce qui vous est fourni, pas l'architecture à concevoir : celle-ci reste à définir par votre équipe.



Ce cluster de démonstration tourne sur une seule machine. Il n'est pas équivalent à une infrastructure de production.

Architecture fournie



Services

Conteneur	Rôle	Données conservées dans
`namenode`	Métadonnées HDFS ; poste de commande pour `hdfs` et `hadoop`	Volume `hadoop_namenode`
`datanode1`, `datanode2`, `datanode3`	Stockage des blocs HDFS	Un volume par DataNode
`resourcemanager`	Attribution des ressources YARN	Aucun
`nodemanager`	Exécution des tâches YARN	Aucun

Conteneur	Rôle	Données conservées dans
`historyserver`	Historique des applications terminées	Volume `hadoop_historyserver`
`hive-server`	Point d'entrée des requêtes Hive (`beeline`)	Aucun
`hive-metastore`	Service des métadonnées des tables Hive	Aucun
`hive-metastore-postgresql`	Base des métadonnées Hive	Volume `hive_metastore`

Configuration utile

Paramètre	Valeur
Version de Hadoop	3.2.1
Version de Hive	2.3.2
Python sur `namenode` et `nodemanager`	3.11
Facteur de réplication HDFS	3 (valeur par défaut de Hadoop)
Taille de bloc HDFS	128 Mo (valeur par défaut de Hadoop)
Ressources du NodeManager	6 Go et 4 vCPU
Mémoire d'une tâche map, reduce ou d'un ApplicationMaster	1 Go

Les valeurs modifiables se trouvent dans `hadoop.env`.



Calcul et stockage sont séparés dans ce cluster. L'unique NodeManager ne porte aucun DataNode, et les DataNodes n'exécutent aucune tâche. Tenez-en compte dans vos observations et vos extrapolations.

Différences avec le TD1

Élément	TD1	Ce dossier
Hive	Absent	Trois services ajoutés
Python	Absent des images	Python 3.11 sur `namenode` et `nodemanager`
Dossier `persistant/`	Absent	Monté sur `/persistant` dans `namenode`
Mémoire YARN	Dépend de votre `hadoop.env`	Réduite pour tenir sur un portable
Reprise du ResourceManager après redémarrage	Dépend de votre `hadoop.env`	Désactivée
Redémarrage des conteneurs	`always`	`unless-stopped` : un conteneur arrêté par vous reste arrêté

Démarrer le cluster

Le premier lancement télécharge les images et Python : faites-le avant la séance si possible.

```
cd environnement
docker compose up -d --build
docker compose ps
```

Attendez que HDFS soit prêt : la commande suivante doit indiquer `Live datanodes (3)`.

```
docker exec namenode hdfs dfsadmin -report
```

Interfaces web

Service	Adresse
NameNode	http://localhost:9870
ResourceManager	http://localhost:8088
NodeManager	http://localhost:8042
History Server	http://localhost:8188
DataNode 1	http://localhost:9864
DataNode 2	http://localhost:9865
DataNode 3	http://localhost:9866

HiveServer2 écoute sur le port `10000`.

Charger des fichiers dans HDFS

Depuis le dossier `environnement/`, si `donnees/` se trouve à côté :

```
docker cp ../donnees namenode:/tmp/donnees
docker exec -it namenode bash
hdfs dfs -mkdir -p /chemin/de/votre/choix
hdfs dfs -put /tmp/donnees/*.csv /chemin/de/votre/choix/
```

Lancer un job MapReduce Streaming

Dans le conteneur `namenode`, après y avoir copié vos scripts :

```
hadoop jar /opt/hadoop-3.2.1/share/hadoop/tools/lib/hadoop-streaming-*.jar \
  -files /tmp/mapper.py,/tmp/reducer.py \
  -input /chemin/entree \
  -output /chemin/sortie \
  -mapper "python3 mapper.py" \
  -reducer "python3 reducer.py"
```

Le répertoire de sortie ne doit pas déjà exister.

Utiliser Hive

Comptez une à deux minutes après le démarrage pour que HiveServer2 réponde.

```
docker exec -it hive-server /opt/hive/bin/beeline -u jdbc:hive2://localhost:10000
```

Le dossier persistant

`environnement/persistant/` sur la machine hôte correspond à `/persistant` dans `namenode`. Il simule un stockage objet durable de type S3 : il se trouve hors du cluster et survit à sa destruction.

```
# Depuis le conteneur namenode
hdfs dfs -get /chemin/dans/hdfs/fichier /persistant/
hdfs dfs -put /persistant/fichier /chemin/dans/hdfs/
```

Arrêter ou détruire le cluster

Commande	Effet
`docker compose stop`	Arrête les conteneurs ; HDFS et Hive sont conservés
`docker compose start`	Relance les conteneurs arrêtés
`docker compose stop datanode1`	Arrête un seul service, par exemple pour un test de panne
`docker compose down`	Supprime les conteneurs ; les volumes sont conservés
`docker compose down -v`	**Supprime aussi les volumes : HDFS et les tables Hive sont perdus.** `persistant/` est conservé

En cas de conflit

Si Docker signale qu'un nom de conteneur ou un port est déjà utilisé, un autre cluster tourne sans doute encore, par exemple celui du TD1. Arrêtez-le depuis son propre dossier avec `docker compose down`, sans `-v` si vous voulez garder ses données.

From:
http://slamwiki2.kobject.net/ - **SlamWiki 2.1**

Permanent link:
http://slamwiki2.kobject.net/eadl/bloc5/fm3/td3/env

Last update: **2026/11/01 17:56**