# TD2 — De Docker à AWS EMR : Hadoop managé dans le cloud

## Big Data et traitement massif — séance 2

- **Durée :** 4 heures
- **Organisation :** travail en binôme
- **Niveau :** Mastère — intermédiaire
- **Pré-requis :** TD1 - De Spark à Hadoop : stocker et traiter des données avec HDFS et YARN réalisé.

**Important — environnement AWS**

Ce TD utilise la console AWS et quelques commandes exécutées depuis un terminal. Il s'agit d'une découverte **manuelle** d'Amazon EMR. N'utilisez ni Terraform ni CloudFormation dans ce TD : l'automatisation de cette infrastructure fera l'objet du TD3.

AWS est un environnement payant. Chaque binôme doit utiliser le compte, la région et les limites de coûts indiqués par l'enseignant. Ne créez aucune ressource supplémentaire.

**Prérequis avant la séance (TD2 — AWS EMR)**

À faire avant le jour du TD : un délai d'activation peut être nécessaire.

1. Se connecter à la console AWS avec son compte personnel (Free Tier).
2. Aller dans le service EMR (Elastic MapReduce), via la barre de recherche en haut de la console.
3. Si le message « Complete your account setup » apparaît (accès limité en « Free Plan »), cliquer sur le bouton « Upgrade plan ». Aucune information supplémentaire ne devrait être demandée si les informations de paiement sont déjà enregistrées : la mise à niveau est instantanée et gratuite, sans plan de support payant à choisir.
4. Vérifier que la page d'accueil du service EMR s'affiche, sans message de blocage.
5. Cette étape n'affecte pas le solde de crédit Free Tier (100–150 $ selon les comptes) et ne génère pas de facturation. Il s'agit uniquement d'une vérification de compte requise par AWS pour débloquer l'accès à certains services.

## Objectifs

À la fin du TD, vous devez être capables de :

- créer un cluster Hadoop managé avec Amazon EMR depuis la console AWS ;
- identifier les rôles des nœuds EMR et les rapprocher de NameNode, DataNode, ResourceManager et NodeManager ;
- accéder au master en SSH et retrouver les services HDFS et YARN ;
- observer les interfaces web du cluster à travers un tunnel SSH ;
- déposer un fichier dans HDFS et dans Amazon S3, puis comparer leurs propriétés ;
- exécuter un job MapReduce Streaming Python en lisant depuis S3 et en écrivant dans S3 ;

- relier dimensionnement, tolérance de panne, persistance et coût ;
- détruire proprement les ressources créées ;
- identifier ce qui devra être automatisé dans le TD3.

# Fil rouge : les ventes d'un site e-commerce

Dans le TD1, vous avez généré un fichier `ventes.csv`, puis vous avez travaillé dans un mini-cluster Hadoop lancé avec Docker Compose. Le chemin logique utilisé était notamment :

```
/data/ecommerce/ventes.csv
/data/ecommerce/input/
/data/ecommerce/output/
```

Vous reprenez **le même générateur Python et le même format de fichier**. Le résultat métier demandé reste le même : calculer le chiffre d'affaires total par catégorie avec MapReduce Streaming.

Dans le TD1, les conteneurs simulaient les machines d'un cluster sur votre poste. Dans ce TD, les machines sont des instances EC2 provisionnées par Amazon EMR dans un VPC AWS.

#### Question de départ

Votre fichier actuel fait peut-être quelques mégaoctets. Et si `ventes.csv` pesait 500 Go ? Sur combien de machines faudrait-il le stocker et le traiter ? Que faudrait-il configurer vous-même, et que pourrait prendre en charge un service managé ?

Écrivez vos premières hypothèses avant de commencer. Vous les vérifierez dans les missions suivantes.

# Organisation du binôme

Répartissez les rôles, puis échangez-les à mi-parcours :

| Rôle | Responsabilités |
|---|---|
| Pilote AWS | manipule la console et le terminal, annonce chaque action avant de l'exécuter |
| Observateur / rapporteur | complète les tableaux, relève les valeurs affichées et vérifie les coûts et la suppression |

Chaque binôme doit conserver les réponses aux questions ouvertes et les valeurs observées. Elles servent de compte rendu.

# Préparation — vérifier les éléments disponibles

Avant de créer quoi que ce soit :

1. récupérez le script de génération de `ventes.csv` du TD1 ;
2. récupérez `mapper.py` et `reducer.py` du TD1 ;
3. vérifiez que vous avez accès à la console AWS ;
4. sélectionnez la région AWS `eu-west-3`.

# Mission 1 — Créer un cluster EMR avec la console AWS

## Problème

Vous devez fournir une infrastructure Hadoop capable d'exécuter le traitement des ventes. Vous ne démarrez plus quatre conteneurs à la main : vous devez décrire le cluster à un service cloud et comprendre les choix proposés.

## 1.1 Ouvrir EMR et préparer la création

1. Connectez-vous à la console AWS avec votre compte perso.
2. Vérifiez la région affichée en haut à droite.
3. Ouvrez **Amazon EMR** puis la page de gestion des clusters.
4. Choisissez la création d'un cluster, et non un cluster « instantané » ou une fonction d'analyse préconfigurée si plusieurs parcours sont proposés.
5. Donnez un nom explicite, par exemple :

```
TD2-emr-noms
```

## 1.2 Choisir les logiciels

Dans la configuration logicielle :

1. sélectionner la version EMR **emr-7.14.0** ;
2. choisir l'offre d'applications **Spark Interactive**, qui installe automatiquement Hadoop, YARN, HDFS et MapReduce (inclus dans le socle Hadoop, non affichés séparément dans la liste des applications) ;
3. vérifier que Hive, Spark, Livy et JupyterEnterpriseGateway sont bien cochés (sélection par défaut de cette offre) ;
4. ne cocher aucune application supplémentaire (Pig, Oozie, HBase, Presto, Trino, Zeppelin, Flink, TensorFlow, etc.) : elles ne sont pas nécessaires pour ce TD et alourdissent inutilement le démarrage du cluster ;
5. noter la version exacte affichée dans la console.

Les noms de menus et la version exacte peuvent évoluer dans la console. Ce qui compte ici est de relever la **release EMR**, la version Hadoop et les applications réellement installées, plutôt que de recopier aveuglément une capture d'écran.

| Élément observé | Valeur de notre cluster | Pourquoi cette valeur ? |
|---|---|---|
| Région AWS (sélecteur en haut à droite de la console, choisie avant de créer le cluster) | (ex. eu-west-3) | même région pour limiter les transferts et simplifier l'accès |
| Release EMR | emr-7.14.0 | dernière version stable en 7.x, compatible Hadoop 3.x |
| Offre d'applications | Spark Interactive | inclut Hadoop, Hive, Spark, Livy, JupyterEnterpriseGateway |
| Hadoop | 3.4.2 | stockage et traitements étudiés dans le TD1 |
| YARN | inclus avec Hadoop 3.4.2 | gestion des ressources et des applications |
| MapReduce | inclus avec Hadoop 3.4.2 | exécution du job Streaming |
| HDFS | inclus avec Hadoop 3.4.2 | comparaison avec le stockage local du TD1 |
| Hive | 3.1.3 | non utilisé directement dans ce TD, mais installé par défaut avec l'offre |
| Spark | 3.5.x | traitements distribués étudiés dans les missions suivantes |
| Applications non sélectionnées | Pig, Oozie, HBase, Presto, Trino, Zeppelin, Flink, TensorFlow, Zookeeper, Phoenix, Hue, JupyterHub, HCatalog, AmazonCloudWatchAgent | hors périmètre du TD (streaming, NoSQL, ML, orchestration, monitoring avancé) |

**Remarque** : YARN, HDFS et MapReduce ne sont pas des cases à cocher séparées dans la console EMR — ils font partie intégrante du socle Hadoop installé dès que “Hadoop” est sélectionné (ce qui est automatique avec l'offre Spark Interactive). Il n'y a donc pas de choix à faire les concernant, seulement à vérifier leur présence et leur version dans les détails du cluster une fois créé.

## 1.3 Comprendre les nœuds

Dans la section matériel, la console AWS propose un choix entre **groupes d'instances uniformes** (par défaut) et **flottes d'instances flexibles**. Gardez le mode par défaut pour ce TD.

La console affiche alors trois rôles :

- **Primaire** ;
- **Unité principale** ;
- **Tâche - 1** (et suivantes si plusieurs nœuds de ce type).

Remplacez le type d'instance par défaut par une instance généraliste standard, plus universellement compatible :

```
Configuration recommandée pour ce TD :
  Primaire         : m5.xlarge
  Unité principale : m5.xlarge
  Tâche - 1        : m5.xlarge
```

| Groupe EMR | Nombre choisi | Type d'instance | Rôle attendu |
|---|---|---|---|
| Primaire | 1 | m5.xlarge | services de coordination, NameNode, ResourceManager |
| Unité principale | 1 | m5.xlarge | stockage HDFS et exécution YARN |

| Groupe EMR | Nombre choisi | Type d'instance | Rôle attendu |
|---|---|---|---|
| Tâche - 1 | 1 | m5.xlarge | exécution YARN sans stockage HDFS durable du cluster |

La famille **m5** est une famille d'instances généralistes (x86_64, Intel), le choix le plus répandu et documenté pour EMR/Hadoop. Elle offre un bon équilibre CPU/mémoire, suffisant pour ce TD.

À noter : avec une seule instance d'unité principale, la réplication HDFS par défaut (facteur 3) ne peut pas être pleinement respectée (il faudrait au moins 3 instances d'unité principale pour avoir 3 copies sur des machines différentes). C'est acceptable dans ce contexte pédagogique à petite échelle, mais c'est un point à mentionner dans vos observations si on vous interroge sur la tolérance aux pannes de ce cluster.

## 1.4 Faire le rapprochement avec le TD1

Complétez le tableau avant de poursuivre. Une case peut contenir plusieurs services : EMR décrit des **rôles de nœuds**, et non une correspondance stricte « un conteneur = un service ».

| Dans EMR | Dans le TD1 Docker Compose | Services principalement associés | Peut contenir des blocs HDFS ? |
|---|---|---|---|
| Primary / Master | | | |
| Core | | | |
| Task | | | |

Répondez :

1. Dans le TD1, quel conteneur jouait le rôle du NameNode ? Et du ResourceManager ?
2. Pourquoi les nœuds core portent-ils généralement les DataNodes ?
3. Pourquoi un nœud task ne doit-il pas être considéré comme une copie supplémentaire des données HDFS ?
4. Dans un cluster réel, que se passe-t-il si vous ajoutez 10 nœuds task mais aucun nœud core ?
5. Le nombre de nœuds choisi suffit-il à démontrer une tolérance de panne réelle ? Pourquoi ?

## 1.5 Réseau

Sur un compte AWS standard, un **VPC par défaut** existe déjà dans chaque région, avec un subnet public dans chaque zone de disponibilité. EMR le sélectionne automatiquement : **il n'y a rien à créer ni configurer manuellement** pour ce TD.

Dans la section réseau de l'assistant de création :

```
1. vérifiez que le VPC proposé est bien le VPC par défaut (``vpc-xxxxxxxx
(default)``) ;
2. un subnet est présélectionné : laissez la valeur proposée ;
3. les security groups sont créés automatiquement par EMR (un pour le master, un
pour les core/task) si vous ne modifiez rien.
```

Notez les valeurs proposées automatiquement :

| Paramètre réseau | Valeur | Observation |
|---|---|---|
| VPC | | par défaut |

| Paramètre réseau | Valeur | Observation |
|---|---|---|
| Subnet | | public (zone de disponibilité) |
| Security group master | | créé automatiquement par EMR |
| Security group core/task | | créé automatiquement par EMR |
| Adresse publique du master | | oui, par défaut sur subnet public |

Le security group du master créé par défaut n'autorise pas le SSH entrant. Il faudra l'ouvrir manuellement après la création du cluster (voir section suivante) pour pouvoir vous connecter.

Répondez :

1. Pourquoi un cluster peut-il avoir besoin d'une communication interne même si vous ne vous connectez qu'au master ?
2. Pourquoi les nœuds core et task ne doivent-ils généralement pas être ouverts directement à Internet ?
3. Dans Docker Compose, quelles règles de réseau étaient implicites ou locales ?

## 1.6 Paire de clés EC2 (accès SSH)

Pour pouvoir se connecter en SSH au nœud primaire du cluster (console, exécution manuelle de commandes Hadoop, consultation de fichiers de logs), une **paire de clés EC2** est nécessaire.

1. Si vous possédez déjà une paire de clés EC2 dans la région utilisée, vous pouvez la réutiliser.
2. Sinon, créez-en une nouvelle **avant** de lancer le cluster :

```
Étapes (en dehors de la création du cluster EMR) :
  EC2 > Réseau et sécurité > Paires de clés > Créer une paire de clés
  Nom : td-hadoop-votrenom
  Type de clé : RSA
  Format de fichier privé :
     - .pem si vous vous connectez depuis Linux/Mac ou WSL
     - .ppk si vous utilisez PuTTY sous Windows
```

Le fichier de clé privée est **téléchargé une seule fois**. Conservez-le précieusement (par exemple dans votre dossier utilisateur, avec des droits restreints) : il ne pourra pas être retéléchargé depuis la console AWS.

Sous Linux/Mac, pensez à restreindre les droits du fichier avant de l'utiliser :

```
chmod 400 td-hadoop-votrenom.pem
```

Dans la configuration du cluster EMR, section **Sécurité**, sélectionnez cette paire de clés dans le menu déroulant **Paire de clés EC2 (SSH)**.

## 1.7 Rôles IAM

EMR nécessite deux rôles IAM pour fonctionner :

- un **rôle de service EMR** (permet à EMR de gérer les ressources EC2, réseau, etc. en votre nom) ;
- un **rôle de profil d'instance EC2** (attaché aux nœuds du cluster, permet l'accès à S3, CloudWatch, etc.).

Dans la section **Rôle IAM**, AWS propose deux options pour chacun de ces deux rôles :

- **Choisir une fonction du service existant** : sélectionne un rôle déjà créé auparavant ;
- **Créer une fonction du service** : AWS crée automatiquement le rôle nécessaire.

**Cas attendu : votre compte est un compte AWS standard (personnel, free tier ou passé en basic)**

Sur ce type de compte, vous disposez normalement des droits IAM complets. Choisissez **Créer une fonction du service** pour les deux champs. AWS génère automatiquement les rôles nécessaires, généralement nommés :

1. ``EMR_DefaultRole`` ou ``AmazonEMR-ServiceRole-xxxx`` (rôle de service) ;
2. ``EMR_EC2_DefaultRole`` ou ``AmazonEMR-InstanceProfile-xxxx`` (profil d'instance EC2).

Ces rôles seront ensuite proposés automatiquement lors des prochaines créations de cluster. C'est le cas normal et attendu pour ce TD.

**Cas exceptionnel : erreur de création (droits IAM restreints ou rôle déjà partiellement créé)**

Il peut arriver, même sur un compte standard, qu'**un des deux rôles existe déjà** (par exemple suite à un essai antérieur) **sans que le second n'existe**. Ce n'est pas incohérent : ce sont deux ressources IAM indépendantes.

Si vous tentez **Créer une fonction du service** pour le champ **Fonction du service EC2** et obtenez une erreur du type :

```
Échec de la création du rôle EMR_EC2_DefaultRole. Vous n'obtenez peut-être pas l'autorisation nécessaire pour effectuer cette action.
```

**Marche à suivre dans ce cas :**

1. Retournez sur **Choisir une fonction du service existant** pour le champ **Fonction du service EC2** ;
2. vérifiez si un rôle apparaît dans la liste déroulante, par exemple ``EMR_EC2_DefaultRole`` (déjà créé lors d'un essai précédent) ;
3. sélectionnez-le s'il apparaît.

**Si l'erreur persiste ou si aucun rôle n'apparaît**, signalez-le à l'enseignant : cela peut indiquer une restriction particulière sur votre compte à vérifier.

## 1.8 Associer la clé SSH et créer le cluster

1. Sélectionnez la paire de clés EC2 fournie ou créée pour le TD. 2. Vérifiez une dernière fois la région, le nombre d'instances, le type d'instance, le rôle IAM et le réseau. 3. Créez le cluster. 4. Observez les états successifs : démarrage, configuration, attente, en cours d'exécution. 5. Tant que le cluster n'est pas à l'état **Waiting** ou équivalent, ne lancez pas le job.

**Le bouton de création lance des ressources facturées.** Relevez l'heure de création et le nom du cluster. Le nettoyage de la Mission 7 est obligatoire, même si le job échoue.

# Mission 2 — Explorer le cluster depuis le master

## Problème

Le cluster est « prêt » dans la console. Est-ce suffisant pour savoir ce qui fonctionne ? Connectez-vous au master et retrouvez les preuves observables dans le TD1.

## 2.1 Récupérer l'adresse du master

Dans la page du cluster, ouvrez les détails et relevez le DNS ou l'adresse publique du nœud primaire. Depuis votre poste :

```
ssh -i ma-cle-emr.pem hadoop@DNS_PUBLIC_DU_MASTER
```

Le nom d'utilisateur dépend de l'image. Pour une AMI EMR standard, `hadoop` est couramment utilisé ; ne remplacez pas ce nom au hasard si la consigne de votre compte indique autre chose.

À la première connexion, vérifiez l'empreinte proposée. Si vous obtenez `Permission denied (publickey)`, contrôlez le chemin de la clé, `chmod 400`, le nom d'utilisateur et la règle SSH du security group.

## 2.2 Identifier la machine

```
hostname
whoami
cat /etc/os-release | head
free -h
df -h
```

Relevez le nom d'hôte et la mémoire visible. Comparez la machine avec votre ordinateur et avec le conteneur master du TD1.

## 2.3 Observer HDFS

Exécutez :

```
hdfs dfs -ls /
hdfs dfs -df -h
hdfs dfsadmin -report
```

Selon la version, la sortie peut contenir des lignes supplémentaires. Relevez au minimum : nombre de DataNodes vivants, capacité totale, capacité utilisée et espace restant.

| Indicateur HDFS | Valeur EMR | Valeur / observation dans le TD1 |
|---|---|---|
| Live datanodes | | |
| Dead datanodes | | |
| DFS capacity | | |
| DFS used | | |
| Block size observée | | |
| Facteur de réplication par défaut | | |

Questions :

1. Avec deux nœuds core, combien de DataNodes observez-vous ?
2. Le nombre de DataNodes est-il égal au nombre total de machines ?
3. Quelle différence voyez-vous entre capacité disque brute et capacité HDFS disponible ?
4. La réplication HDFS apporte-t-elle une persistance après la destruction du cluster ? Justifiez sans encore regarder S3.

## 2.4 Retrouver YARN

```
yarn node -list
yarn queue -status default
jps
```

Dans `jps`, repérez les processus Hadoop/YARN visibles sur le master. Complétez :

| Élément | Résultat observé | Rôle |
|---|---|---|
| ResourceManager | | |
| NodeManager sur le master, éventuellement | | |
| NameNode | | |
| DataNode sur le master, éventuellement | | |
| NodeManagers des core | | |

Questions :

1. Combien de nœuds YARN sont disponibles ?
2. Tous les nœuds YARN sont-ils nécessairement des DataNodes ?
3. Quelle commande du TD1 utilisiez-vous pour observer les ressources YARN ?
4. En quoi la sortie de `yarn node -list` complète-t-elle `hdfs dfsadmin -report` ?

Dans un cluster EMR, la répartition exacte des démons dépend de la release, des options et du type de nœud. Utilisez les commandes et la console comme source d'observation : ne concluez pas à partir du seul nom « master », « core » ou « task ».

---

# Mission 3 — Accéder aux interfaces web avec un tunnel SSH

## Problème

Dans le TD1, vous ouvriez des ports Docker exposés sur `localhost`. Dans AWS, les interfaces du master ne doivent pas être publiques par défaut. Comment observer l'interface NameNode et l'interface YARN sans ouvrir leurs ports à Internet ?

## 3.1 Vérifier l'accès direct

Depuis votre poste, essayez éventuellement d'ouvrir l'URL de l'interface indiquée dans la console EMR ou dans la documentation de la release. Si l'accès direct échoue, c'est une information utile : le service existe dans le réseau du cluster, mais n'est pas exposé à votre navigateur.

N'ajoutez pas une règle `0.0.0.0/0` simplement pour faire fonctionner l'interface.

## 3.2 Mettre en place un tunnel local

Gardez une première session SSH ouverte. Dans un second terminal, créez des redirections locales vers les ports web du master. Les numéros de ports peuvent varier selon la release ; utilisez ceux affichés dans la console ou par l'enseignant.

```
ssh -i ma-cle-emr.pem \
  -N \
  -L 9870:localhost:9870 \
  -L 8088:localhost:8088 \
  hadoop@DNS_PUBLIC_DU_MASTER
```

Ouvrez ensuite dans votre navigateur :

```
http://localhost:9870
http://localhost:8088
```

Le port `9870` est fréquemment associé à l'interface web du NameNode et `8088` à celle du ResourceManager, mais vérifiez la page de votre cluster. Si un service utilise un autre port, remplacez la redirection.

Pour explorer plusieurs interfaces sans ajouter chaque port, vous pouvez utiliser un proxy SOCKS dynamique si la consigne le permet :

```
ssh -i ma-cle-emr.pem -N -D 8157 hadoop@DNS_PUBLIC_DU_MASTER
```

Configurez ensuite le navigateur ou une extension pour utiliser un proxy SOCKS5 sur `localhost:8157`. Cette méthode n'est pas équivalente à une publication de ports : le navigateur envoie ses requêtes à travers la session SSH.

## 3.3 Observer et comparer

Dans l'interface NameNode, relevez l'état des DataNodes, la capacité et les blocs. Dans l'interface YARN, relevez les nœuds, les ressources et les applications.

| Interface | Information recherchée | Valeur / observation |
|---|---|---|
| NameNode UI | nombre de DataNodes live | |
| NameNode UI | capacité HDFS utilisée | |
| YARN RM UI | nombre de nœuds actifs | |
| YARN RM UI | mémoire/vCores disponibles | |
| YARN RM UI | applications en cours | |

Questions :

1. Où se termine le tunnel : sur votre poste, sur le master ou sur les nœuds core ?
2. Pourquoi le navigateur peut-il afficher `localhost` alors que le NameNode est dans AWS ?
3. Quelle différence entre un port Docker publié dans le TD1 et un tunnel SSH dans le TD2 ?
4. Quel est le risque d'un tunnel laissé ouvert sur un poste partagé ?

---

# Mission 4 — Stocker les données dans HDFS et dans S3

## Problème

Vous devez placer `ventes.csv` dans le cluster pour une première observation, mais aussi le conserver indépendamment du cluster. Déposez-le dans HDFS **et** dans Amazon S3, puis comparez les deux stockages.

## 4.1 Préparer un espace S3

Utilisez le bucket fourni. Si vous devez en créer un, son nom doit être globalement unique et conforme aux règles S3. Ajoutez un préfixe propre au binôme :

```
s3://NOM_DU_BUCKET/td2/binomeXX/
s3://NOM_DU_BUCKET/td2/binomeXX/input/
s3://NOM_DU_BUCKET/td2/binomeXX/output/
```

Depuis le master, vérifiez que le rôle de l'instance permet l'accès prévu :

```
aws sts get-caller-identity
aws s3 ls s3://NOM_DU_BUCKET/td2/binomeXX/
```

Si `aws` n'est pas configuré comme attendu, ne copiez pas une clé personnelle sur le master. Signalez le problème et vérifiez le rôle IAM associé au profil d'instance.

## 4.2 Copier les données dans S3

Vous pouvez déposer le fichier depuis votre poste avec la console S3 ou avec l'AWS CLI. Exemple depuis le poste, si l'authentification pédagogique est configurée :

```
aws s3 cp ventes.csv \
  s3://NOM_DU_BUCKET/td2/binomeXX/input/ventes.csv
aws s3 ls s3://NOM_DU_BUCKET/td2/binomeXX/input/
```

Si vous avez d'abord copié le fichier sur le master, exécutez plutôt la commande depuis le master. Notez la taille affichée et, si possible, l'horodatage.

## 4.3 Copier également le fichier dans HDFS

Depuis le master :

```
hdfs dfs -mkdir -p /data/ecommerce/input
hdfs dfs -put -f ventes.csv /data/ecommerce/input/ventes.csv
hdfs dfs -ls -h /data/ecommerce/input
hdfs dfs -du -h /data/ecommerce/input/ventes.csv
```

Contrôlez le contenu sans afficher tout le fichier :

```
hdfs dfs -cat /data/ecommerce/input/ventes.csv | head
aws s3 cp s3://NOM_DU_BUCKET/td2/binomeXX/input/ventes.csv - | head
```

Comparez les deux sorties : en-tête, séparateur, ordre des colonnes et premières lignes doivent être compatibles avec le TD1.

## 4.4 Comparer HDFS, S3 et EMRFS

Dans les traitements EMR, un chemin s3://... peut être utilisé comme source ou destination par Hadoop grâce à la couche d'intégration S3 d'EMR, souvent appelée EMRFS. Vous allez l'utiliser sans la traiter comme un disque local.

| Critère | HDFS du cluster EMR | S3 |
|---|---|---|
| Emplacement physique observé | | |
| Répartition / réplication | | |
| Disponible sans cluster actif ? | | |
| Latence d'accès attendue | | |
| Usage typique dans ce TD | | |
| Persistance après terminaison EMR | | |

Répondez :

1. Si vous supprimez le cluster, que devient `/data/ecommerce/input/ventes.csv` dans HDFS ?
2. Que devient s3://NOM_DU_BUCKET/td2/binomeXX/input/ventes.csv ?
3. Pourquoi S3 est-il adapté à la conservation de données d'entrée et de résultats ?
4. Pourquoi ne faut-il pas déduire qu'un chemin S3 est un chemin local du master ?
5. Si le fichier est identique dans les deux stockages, pourquoi les deux copies ne sont-elles pas redondantes de la même manière ?

**S3 n'est pas un espace temporaire gratuit.** Les objets restent présents et peuvent continuer à entraîner des coûts selon la classe, les requêtes et les transferts.

---

# Mission 5 — Relancer MapReduce Streaming depuis S3

## Problème

Dans le TD1, le job lisait un fichier local ou HDFS du cluster Docker. Cette fois, le fichier d'entrée est dans S3 et le résultat doit également être écrit dans S3. Le calcul métier ne change pas.

## 5.1 Vérifier le contrat du job

Le job calcule :

```
clé   = catégorie
valeur = quantité × prix_unitaire
résultat = somme du chiffre d'affaires par catégorie
```

Réutilisez `mapper.py` et `reducer.py` du TD1. Si vous devez les reconstituer, adaptez les noms de colonnes au fichier généré dans votre TD1. Exemple de version avec un CSV comportant les colonnes `category`, `quantity` et `unit_price` :

```
# mapper.py
import csv
import sys

reader = csv.DictReader(sys.stdin)
for row in reader:
    try:
        category = row["category"]
        quantity = float(row["quantity"])
        unit_price = float(row["unit_price"])
        print(f"{category}\t{quantity * unit_price}")
    except (KeyError, TypeError, ValueError):
        # Une ligne invalide est signalée sur stderr, pas dans la sortie métier.
        print(f"Ligne ignorée : {row}", file=sys.stderr)
```

```
# reducer.py
import sys

current_category = None
current_total = 0.0

for line in sys.stdin:
    line = line.rstrip("\n")
    if not line:
        continue
    category, value = line.split("\t", 1)
    value = float(value)

    if current_category is not None and category != current_category:
        print(f"{current_category}\t{current_total:.2f}")
        current_total = 0.0

    current_category = category
    current_total += value

if current_category is not None:
    print(f"{current_category}\t{current_total:.2f}")
```

Le code ci-dessus est un gabarit. Si le script du TD1 produit des colonnes françaises, un montant déjà calculé ou un autre séparateur, reprenez **le code et le contrat du TD1**. Ne modifiez pas le générateur de données uniquement pour faire correspondre le gabarit.

Testez localement avec un petit extrait avant de soumettre le job :

```
head -n 20 ventes.csv | python3 mapper.py | sort -k1,1 | python3 reducer.py
```

Puis copiez les scripts sur le master. Depuis votre poste :

```
scp -i ma-cle-emr.pem mapper.py reducer.py \
  hadoop@DNS_PUBLIC_DU_MASTER:/home/hadoop/
```

Sur le master :

```
chmod +x mapper.py reducer.py
head -n 3 ventes.csv
printf 'category\tquantity\tunit_price\n' | ./mapper.py
```

## 5.2 Vérifier les chemins S3

Avant le lancement, choisissez un répertoire de sortie qui n'existe pas encore :

```
INPUT=s3://NOM_DU_BUCKET/td2/binomeXX/input/ventes.csv
OUTPUT=s3://NOM_DU_BUCKET/td2/binomeXX/output/ca-par-categorie

aws s3 ls "$INPUT"
aws s3 ls "$OUTPUT" || true
```

Hadoop refuse généralement d'écrire dans un répertoire de sortie déjà présent.

## 5.3 Lancer le job

Depuis `/home/hadoop` sur le master :

```
hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar \
  -files mapper.py,reducer.py \
  -input "$INPUT" \
  -output "$OUTPUT" \
  -mapper "python3 mapper.py" \
  -reducer "python3 reducer.py"
```

Le chemin du jar peut varier selon la release. Si le chemin ci-dessus n'existe pas, recherchez le jar déjà installé :

```
find /usr/lib /usr/share -name 'hadoop-streaming*.jar' 2>/dev/null | head
```

Relancez la commande avec le chemin réellement trouvé. Observez les phases de soumission, map, shuffle/sort et reduce.

## 5.4 Observer l'application YARN

Pendant ou après l'exécution :

```
yarn application -list -appStates ALL
hdfs dfs -ls -h "$OUTPUT"
aws s3 ls "$OUTPUT/"
```

Le job peut produire un ou plusieurs fichiers `part-*` ainsi qu'un marqueur `_SUCCESS`. Affichez le résultat :

```
aws s3 cp "$OUTPUT/part-*" -
```

Si le motif `part-*` n'est pas accepté par votre version d'AWS CLI, listez les objets puis copiez le nom exact retourné.

| **Étape** | **Observation** |
|---|---|
| Application YARN | |
| Nombre de tâches map | |
| Nombre de tâches reduce | |
| Temps total | |
| Fichier(s) de sortie | |
| Erreurs / lignes rejetées | |
| Résultat d'une catégorie vérifiée à la main | |

Questions :

1. Le job lit-il les données depuis le disque local du master ? Comment le vérifier ?
2. Où sont écrits les fichiers temporaires et le résultat final ?
3. Quelle différence entre le chemin d'entrée S3 et le chemin de sortie HDFS observé dans le TD1 ?
4. Que se passe-t-il si vous relancez le job avec le même `OUTPUT` ?
5. Si le master est remplacé après le job, où devez-vous chercher le résultat ?

## 5.5 Comparer avec le TD1

| **Axe** | **TD1 Docker Compose** | **TD2 EMR** |
|---|---|---|
| Lancement du job | | |
| Entrée | | |
| Planification | | |
| Exécution map/reduce | | |
| Sortie | | |
| Consultation des logs | | |

# Mission 6 — Observer le dimensionnement et le coût

## Problème

Le job fonctionne. Mais un cluster de trois machines est-il toujours un bon choix ? Vous devez relier la configuration technique à la facture et au volume de données.

## 6.1 Relever les ressources

Dans la console EMR et dans la page EC2, observez les types d'instances, leur nombre et leur état. Complétez :

| Groupe | Type | Nombre | vCPU / mémoire indiqués | Coût horaire affiché ou estimé |
|---|---|---|---|---|
| Master | | | | |
| Core | | | | |
| Task | | | | |

Le coût réel dépend notamment de la région, du type d'instance, du modèle de facturation, des options EMR, du stockage et d'éventuels transferts. Utilisez l'outil d'estimation ou la page tarifaire validée par l'enseignant plutôt que de mémoriser un montant.

Pour une estimation simplifiée :

```
coût horaire approximatif
  = coût EMR des nœuds
  + coût EC2 des instances
  + stockage / requêtes / transferts éventuels

coût de la séance
  ≈ coût horaire × durée pendant laquelle les ressources restent actives
```

Questions :

1. Un cluster arrêté est-il nécessairement un cluster qui ne coûte plus rien ?
2. Quelles ressources continuent éventuellement à exister après la terminaison d'EMR ?
3. Pourquoi un gros cluster peut-il coûter plus cher tout en terminant le job plus vite ?
4. Pour un job ponctuel, quel compromis feriez-vous entre durée et coût ?

## 6.2 Survoler l'auto-scaling

Ouvrez les options de dimensionnement sans modifier le cluster si la consigne ne l'autorise pas. Repérez :

1. la différence entre nombre minimal et maximal de nœuds ;
2. les groupes auxquels une règle peut s'appliquer ;
3. les métriques ou conditions utilisées ;
4. l'effet possible sur la capacité de calcul ;

5. l’effet possible sur HDFS si des nœuds portant des données sont retirés.

| Question | Réponse du binôme |
|---|---|
| Quel groupe ajouteriez-vous pour absorber un pic de calcul ? | |
| Pourquoi des nœuds task sont-ils intéressants pour ce pic ? | |
| Pourquoi un scale-in des core doit-il être traité avec prudence ? | |
| Quelles métriques faudrait-il surveiller ? | |

## 6.3 Relier au TD1 : panne et ressources

Dans le TD1, vous aviez arrêté un DataNode et observé la réplication, puis étudié le scheduler YARN. Ici, relisez les informations disponibles dans les interfaces et répondez sans provoquer de panne non autorisée :

1. Avec deux core, combien de copies d’un bloc peuvent être disponibles selon le facteur de réplication ?
2. Que pourrait faire YARN si un NodeManager devient indisponible ?
3. Que pourrait-il arriver à une tâche en cours ?
4. Pourquoi la tolérance de panne ne signifie-t-elle pas « aucune interruption visible » ?
5. Quel rôle joue S3 si le cluster entier disparaît ?

Ne terminez, n’arrêtez et ne redémarrez pas une instance depuis EC2 pendant le TD sans consigne explicite. Ces actions peuvent mettre le cluster dans un état incohérent et compliquer le nettoyage.

---

# Mission 7 — Nettoyer et vérifier les ressources

## Problème

Le travail est terminé. Un cluster EMR laissé en fonctionnement continue de mobiliser des ressources. Vous devez le supprimer, puis vérifier ce qui reste volontairement dans S3.

## 7.1 Vérifier le résultat avant destruction

Avant de terminer le cluster, conservez dans votre compte rendu :

1. le nom et l’identifiant du cluster ;
2. la release EMR ;
3. les types et nombres de nœuds ;
4. la commande du job ;
5. le chemin S3 de l’entrée ;
6. le chemin S3 du résultat ;
7. une copie textuelle du résultat ou sa capture ;
8. les réponses aux tableaux et questions.

Depuis le terminal :

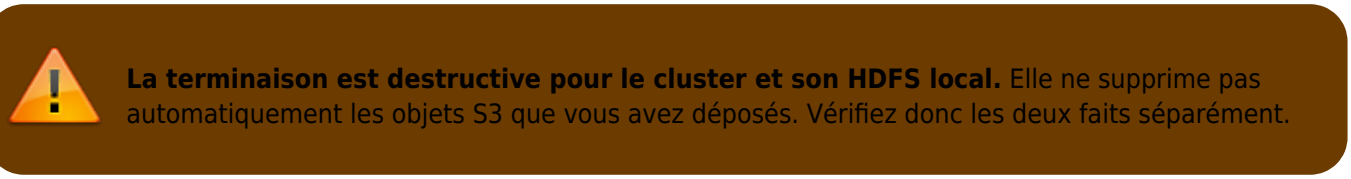
```
aws s3 ls s3://NOM_DU_BUCKET/td2/binomeXX/ --recursive
```

## 7.2 Terminer le cluster

Dans la console EMR :

1. sélectionnez uniquement votre cluster ; 2. choisissez **Terminate / Terminer** ; 3. confirmez après avoir vérifié le nom ; 4. attendez l'état terminé ; 5. observez les instances EC2 associées et vérifiez qu'elles sont également libérées selon le comportement du service.

Ne supprimez pas le bucket partagé de la classe. Ne supprimez pas les objets d'un autre binôme.




## 7.3 Vérifier ce qui reste

Après la terminaison :

1. essayez de retrouver le cluster dans l'historique EMR ;
2. vérifiez que les instances du cluster ne sont plus en cours d'exécution ;
3. vérifiez que le préfixe S3 contient encore l'entrée et le résultat ;
4. supprimez les objets uniquement selon la consigne de restitution.

```
aws s3 ls s3://NOM_DU_BUCKET/td2/binomeXX/ --recursive
# À exécuter seulement si l'enseignant demande la suppression :
aws s3 rm s3://NOM_DU_BUCKET/td2/binomeXX/ --recursive
```

Complétez :

| Ressource | Avant nettoyage | Après terminaison | Action restante |
|---|---|---|---|
| Cluster EMR | | | |
| Instances EC2 du cluster | | | |
| HDFS `/data/ecommerce` | | | |
| Objet S3 `input/ventes.csv` | | | |
| Objets S3 `output/` | | | |
| Clé SSH | | | |

---

# Synthèse — Docker Compose (TD1) versus EMR (TD2)

Complétez d'abord la colonne « observation » sans chercher une définition générale.

| Axe | Docker Compose — TD1 | Amazon EMR — TD2 | Observation du binôme |
|---|---|---|---|
| Provisioning | fichiers Compose et démarrage local des conteneurs | création d'un cluster par la console, instances EC2 provisionnées par EMR | |
| Mise à l'échelle | modification du nombre de conteneurs / ressources locales | nombre et types de nœuds configurés, auto-scaling possible | |
| Tolérance de panne | panne simulée d'un conteneur, réplication HDFS locale | plusieurs instances et réplication, selon la configuration choisie | |
| Coût | ressources de votre poste / environnement local | facturation AWS des services et ressources pendant leur durée de vie | |
| Accès aux interfaces | ports Docker exposés sur localhost | interfaces privées atteintes par tunnel SSH ou proxy | |
| Persistance des données | volumes / fichiers locaux selon Compose | HDFS attaché au cluster ; S3 indépendant du cluster | |
| Gestion des versions | images Docker et fichiers du projet | release EMR, AMI et applications sélectionnées | |
| Sécurité | réseau Docker local et ports publiés | IAM, VPC, subnet, security groups, clé SSH | |

## Questions de synthèse

1. Quel travail d'administration du TD1 est pris en charge par EMR ?
2. Quel travail reste sous votre responsabilité malgré le caractère managé du service ?
3. Pourquoi « managé » ne signifie-t-il pas « sans configuration » ?
4. Pour un fichier de 500 Go, quelles limites de votre ordinateur disparaissent, et quelles nouvelles contraintes apparaissent ?
5. Où placeriez-vous les données d'entrée et les sorties d'un pipeline reproductible ?
6. Quelle erreur de sécurité ou de facturation vous semble la plus facile à commettre ? Quelle vérification l'empêche ?

# Commandes utiles

| Besoin | Commande | |
|---|---|---|
| Connexion SSH au master | `ssh -i ma-cle.pem hadoop@DNS_PUBLIC_DU_MASTER` | |
| Copier un fichier vers le master | `scp -i ma-cle.pem fichier hadoop@DNS_PUBLIC_DU_MASTER:/home/hadoop/` | |
| Vérifier l'identité AWS du nœud | `aws sts get-caller-identity` | |
| Lister un préfixe S3 | `aws s3 ls s3:*BUCKET/PREFIX/ –recursive` \| \| *Copier vers S3* \| `*aws s3 cp fichier.csv s3:*BUCKET/PREFIX/` | |
| Copier depuis S3 | `aws s3 cp s3:*BUCKET/PREFIX/fichier.csv .` \| \| *Supprimer un préfixe S3* \| `*aws s3 rm s3:*BUCKET/PREFIX/ –recursive` | |
| Lister HDFS | `hdfs dfs -ls -h CHEMIN` | |
| Créer un répertoire HDFS | `hdfs dfs -mkdir -p CHEMIN` | |
| Envoyer vers HDFS | `hdfs dfs -put -f fichier CHEMIN/` | |
| Afficher le début d'un fichier HDFS | `hdfs dfs -cat CHEMIN/fichier | head` | |
| Taille HDFS | `hdfs dfs -du -h CHEMIN` | |
| État des DataNodes | `hdfs dfsadmin -report` | |
| Nœuds YARN | `yarn node -list` | |

| Besoin | Commande | |
|---|---|---|
| Applications YARN | `yarn application -list -appStates ALL` | |
| Processus Hadoop locaux | `jps` | |
| Tunnel NameNode / YARN | `ssh -i ma-cle.pem -N -L 9870:localhost:9870 -L 8088:localhost:8088 hadoop@DNS_PUBLIC_DU_MASTER` | |
| Proxy SOCKS | `ssh -i ma-cle.pem -N -D 8157 hadoop@DNS_PUBLIC_DU_MASTER` | |
| Trouver le jar Streaming | `find /usr/lib /usr/share -name 'hadoop-streaming*.jar' 2>/dev/null | head` |
| Lancer MapReduce Streaming | `hadoop jar CHEMIN_STREAMING.jar -files mapper.py,reducer.py -input s3:*BUCKET/input/ -output s3:*BUCKET/output/ -mapper “python3 mapper.py” -reducer “python3 reducer.py”` | |

Remplacez toujours `BUCKET`, `PREFIX`, `DNS_PUBLIC_DU_MASTER`, `ma-cle.pem`, `XX` et les chemins de sortie par vos valeurs réelles. Une commande qui contient encore un placeholder ne doit pas être exécutée telle quelle.

# Ouverture — TD3 et l’infrastructure as code

Dans ce TD, vous avez créé et supprimé un cluster manuellement. Dans le TD3, vous étudierez l’automatisation de cette infrastructure.

Répondez à la question suivante :

**Que faudrait-il automatiser dans ce que vous venez de faire à la main ?**

Dressez une liste ordonnée d’au moins huit éléments. Pensez aux paramètres du cluster, à la sécurité, au réseau, aux rôles IAM, aux chemins S3, au lancement du job, à la collecte des résultats, au dimensionnement et au nettoyage.

Puis classez chaque élément selon sa nature :

| Élément à automatiser | Paramétrage | Provisionnement | Exécution | Vérification / nettoyage | Risque en cas d’erreur |
|---|---|---|---|---|---|
| | | | | | |

## Livrable du binôme

Remettez un compte rendu court contenant :

1. les valeurs complétées dans les tableaux ;
2. une preuve de connexion SSH et de l’exploration HDFS/YARN ;
3. une preuve du tunnel et des deux interfaces web observées ;
4. les chemins S3 d’entrée et de sortie ;
5. le résultat du chiffre d’affaires par catégorie ;
6. la comparaison TD1 / TD2 ;
7. l’état du nettoyage et la liste des éléments à automatiser pour le TD3.

Dernière vérification avant de quitter AWS : votre cluster est terminé, vos instances de test ne tournent plus, et les objets S3 restants correspondent exactement à la consigne de l'enseignant.