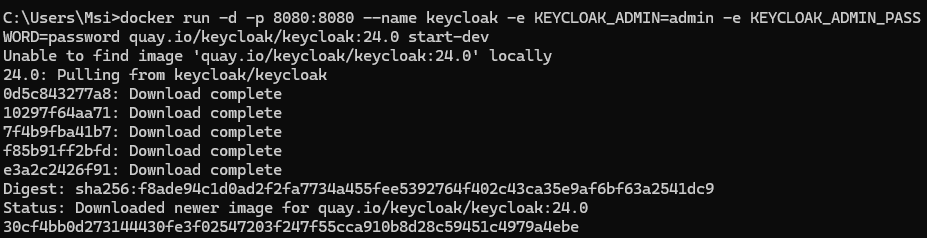
# Etape 1 : creation et configuration des conteneurs

On commence par créer les conteneurs pour chaque service

**Keycloak**

docker run -d -p 8080:8080 --name keycloak -e KEYCLOAK\_ADMIN=admin -e KEYCLOAK\_ADMIN\_PASSWORD=password quay.io/keycloak/keycloak:24.0 start-dev



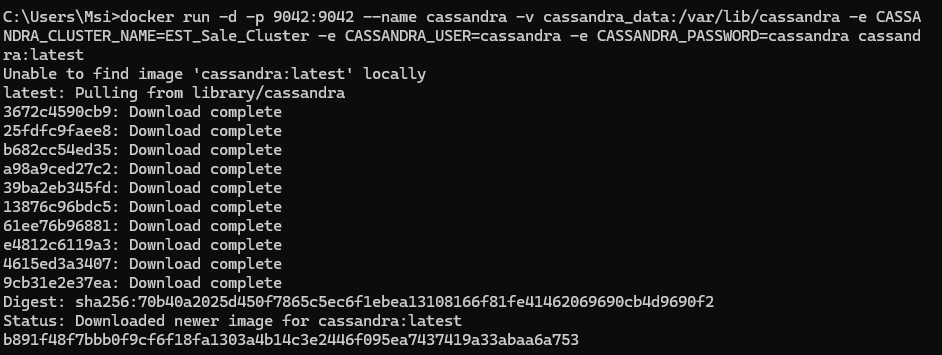
**MinIO**

docker run -d -p 9000:9000 -p 9001:9001 --name minio -v minio\_data:/data -e MINIO\_ROOT\_USER=admin -e MINIO\_ROOT\_PASSWORD=password minio/minio server /data --console-address ":9001"



**Cassandra**

docker run -d -p 9042:9042 --name cassandra -v cassandra\_data:/var/lib/cassandra -e CASSANDRA\_CLUSTER\_NAME=EST\_Cluster -e CASSANDRA\_USER=cassandra -e CASSANDRA\_PASSWORD=cassandra cassandra:latest

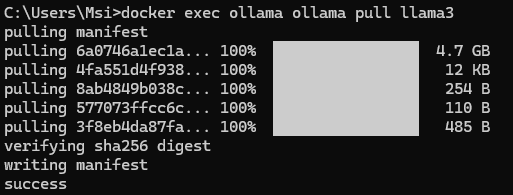


**Ollama**

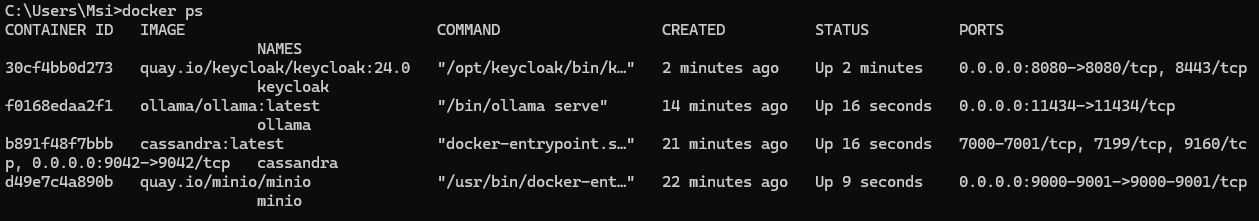
\*\*\*docker run -d -p 11434:11434 --name ollama -v ollama\_data:/root/.ollama --gpus all ollama/ollama:latest



\*\*\*docker exec ollama ollama pull llama3

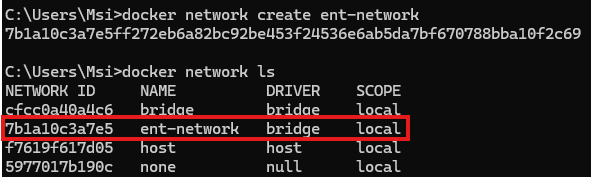


Confirmation des conteneurs



**Reseau docker**

docker network create ent-network



Ajouter les conteneurs au reseau

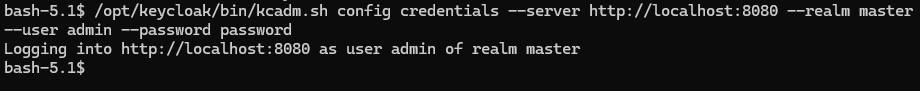


# Etape 2 : configuration et déploiement de keycloak

On commence par configurer keycloak  
premièrement on accède au terminal bash keycloak avec la commande

**Docker exec -it keycloak /bin/bash**

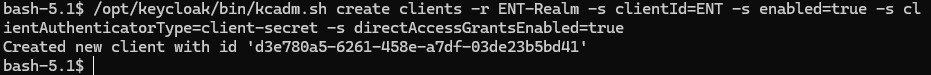
Ensuite on fait un login en tant que admin keycloak avec la commande

**/opt/keycloak/bin/kcadm.sh config credentials --server http://localhost:8080 --realm master --user admin --password password   
**

Ensuite on va créer un realm ‘ENT-Realm’ avec la commande

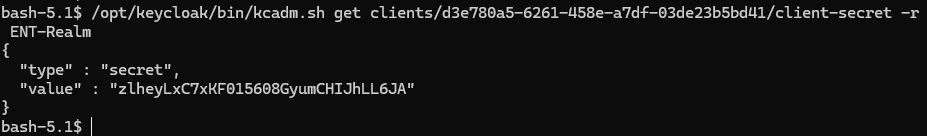
**/opt/keycloak/bin/kcadm.sh create realms -s realm=ENT-Realm -s enabled=true  
**

Ensuite on va créer un client dans le realm avec la commande

**/opt/keycloak/bin/kcadm.sh create clients -r ENT-Realm -s clientId=ENT -s enabled=true -s clientAuthenticatorType=client-secret -s directAccessGrantsEnabled=true  
**

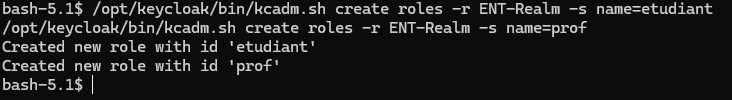
**Dans ce cas le id du client est [**d3e780a5-6261-458e-a7df-03de23b5bd41**]**

Ensuite on va récupérer la clé secrète du client en utilisant son id avec la commande

**/opt/keycloak/bin/kcadm.sh get clients/d3e780a5-6261-458e-a7df-03de23b5bd41/client-secret -r ENT-Realm  
**

**Dans ce cas la clé secrète du client est [**zlheyLxC7xKF015608GyumCHIJhLL6JA**]**

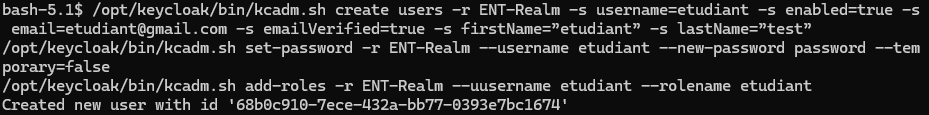
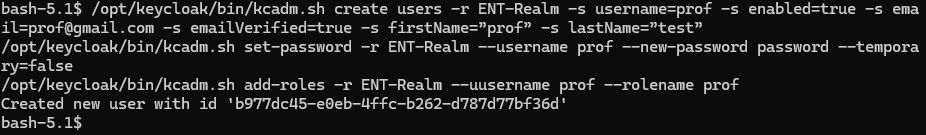
Maintenant on va créer 2 rôles pour ce realm,

**/opt/keycloak/bin/kcadm.sh create roles -r ENT-Realm -s name=etudiant  
/opt/keycloak/bin/kcadm.sh create roles -r ENT-Realm -s name=prof  
**

Maintenant on va créer un utilisateur par role pour tester l’authentification avec Keycloak avec les commandes suivantes:

Etudiant :  
**/opt/keycloak/bin/kcadm.sh create users -r ENT-Realm -s username=etudiant -s enabled=true -s email=**[**etudiant@gmail.com**](mailto:etudiant@gmail.com) **-s emailVerified=true -s firstName=”etudiant” -s lastName=”test”  
/opt/keycloak/bin/kcadm.sh set-password -r ENT-Realm --username etudiant --new-password password --temporary=false  
/opt/keycloak/bin/kcadm.sh add-roles -r ENT-Realm --uusername etudiant --rolename etudiant**

Prof :  
**/opt/keycloak/bin/kcadm.sh create users -r ENT-Realm -s username=prof -s enabled=true -s email=**[**prof@gmail.com**](mailto:prof@gmail.com) **-s emailVerified=true -s firstName=”prof” -s lastName=”test”  
/opt/keycloak/bin/kcadm.sh set-password -r ENT-Realm --username prof --new-password password --temporary=false  
/opt/keycloak/bin/kcadm.sh add-roles -r ENT-Realm --uusername prof --rolename prof**

Maintenant on va tester avec les commandes suivantes

Etudiant :

**curl -X POST "http://localhost:8080/realms/ENT-Realm/protocol/openid-connect/token" \**

**-H "Content-Type: application/x-www-form-urlencoded" \**

**-d "grant\_type=password" \**

**-d "client\_id=ENT" \**

**-d "client\_secret=zlheyLxC7xKF015608GyumCHIJhLL6JA" \**

**-d "username=etudiant" \**

**-d "password=password"**

Prof :

**curl -X POST "http://localhost:8080/realms/ENT-Realm/protocol/openid-connect/token" \**

**-H "Content-Type: application/x-www-form-urlencoded" \**

**-d "grant\_type=password" \**

**-d "client\_id=ENT" \**

**-d "client\_secret=zlheyLxC7xKF015608GyumCHIJhLL6JA" \**

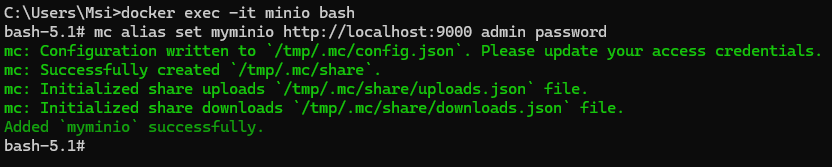
**-d "username=prof" \**

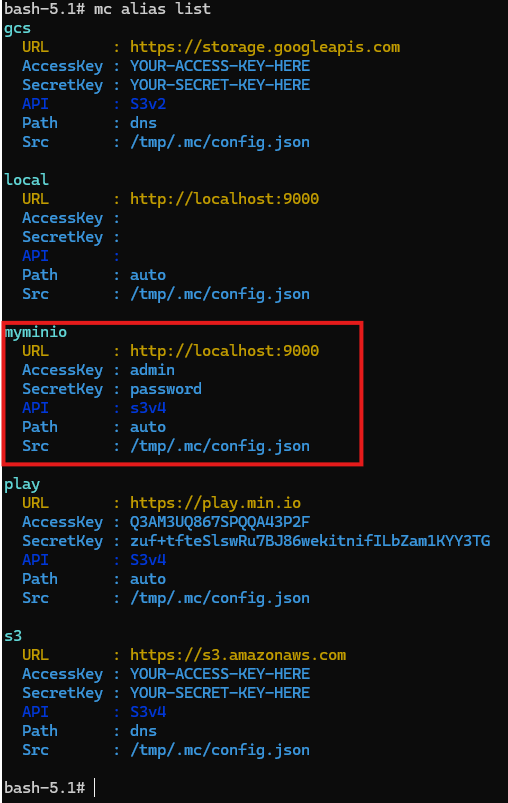
**-d "password=password"**

****

# Etape 3 : configuration et déploiement de minio

On commence par configurer le client minio, On doit connecter mc (MinIO Client) à notre serveur MinIO. On commence par accéder à un terminal bash minio **[docker exec -it minio bash]** et exécuter la commande suivante :  
**mc alias set myminio http://localhost:9000 admin password**

****

****

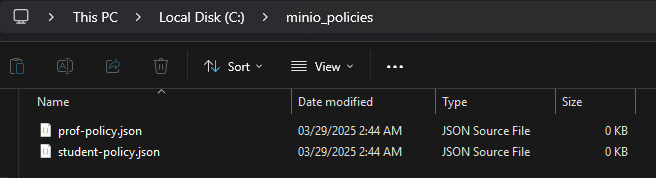
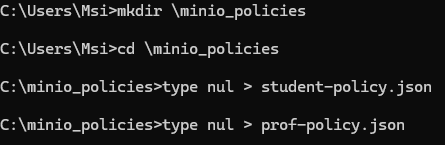
Maintenant on va créer un bucket (un espace de stockage sur minio) avec la commande  
**mc mb myminio/my-bucket**

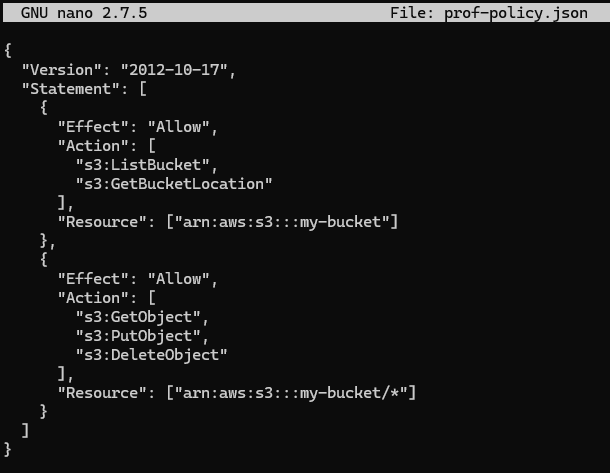


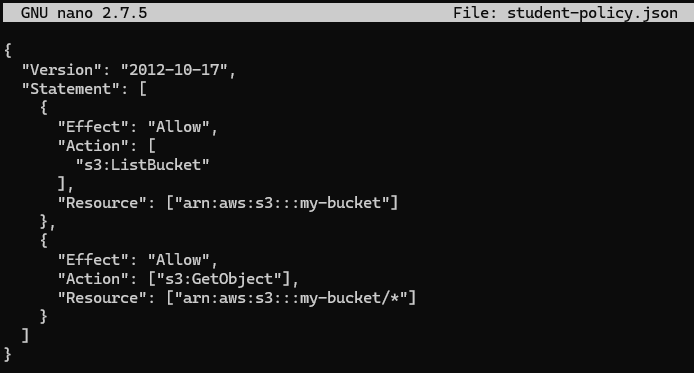
****

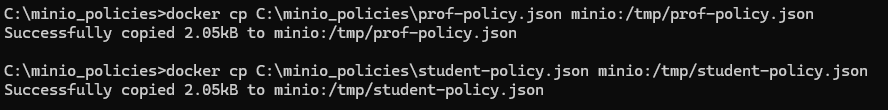
Maintenant il faut définir les permissions du bucket, on va commencer par définir le bucket comme privé avec la commande   
**mc anonymous set private myminio/my-bucket**

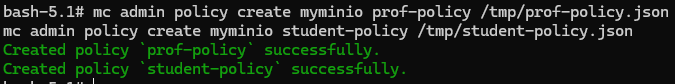


Ensuite on va créer une politique d’accès personnalisée pour permettre aux utilisateurs authentifies d’uploader et télécharger des fichiers, tout d’abord il faut créer deux fichiers **prof-policy.json, student-policy.json** pour les politiques adaptées.  


Ensuite on va ajouter ceci :  


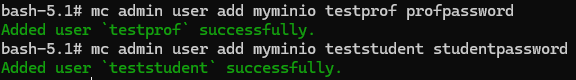


Ensuite il faut copier les fichiers dans le conteneur minio avec la commande  
**docker cp C:\minio\_policies\prof-policy.json minio:/tmp/prof-policy.json  
docker cp C:\minio\_policies\student-policy.json minio:/tmp/student-policy.json**  
****

Et on va appliquer les politiques sur notre bucket.  
**mc admin policy create myminio prof-policy /tmp/prof-policy.json  
mc admin policy create myminio student-policy /tmp/student-policy.json**

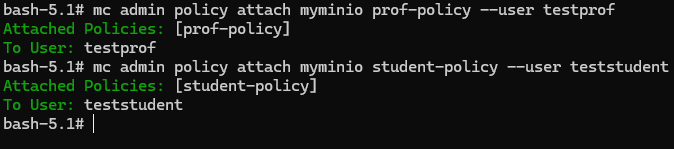
Maintenant il faut créer un utilisateur prof et un utilisateur student pour tester les policies,

Créer l’utilisateur testprof : **mc admin user add myminio testprof profpassword**  
créer l’utilisateur teststudent : **mc admin user add myminio teststudent studentpassword**

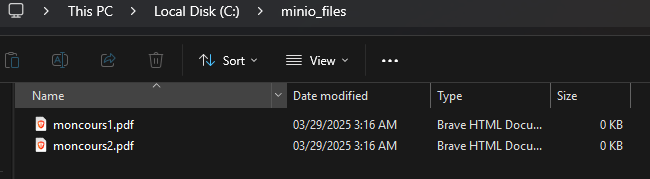
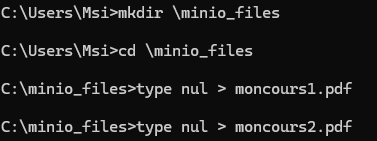
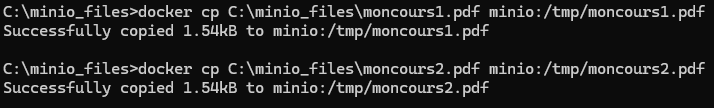
****

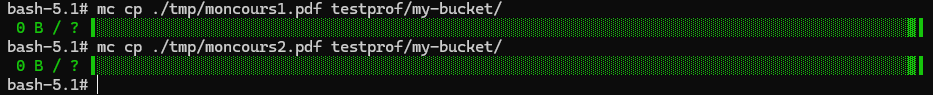
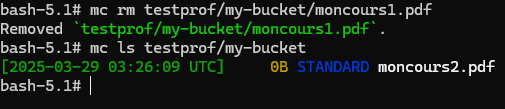
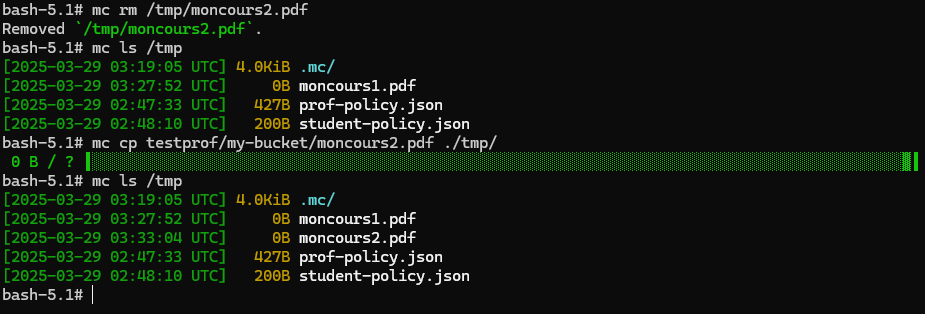
Associer prof-policy a testprof : **mc admin policy attach myminio prof-policy --user testprof**

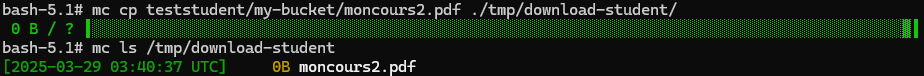
Associer student-policy a teststudent : **mc admin policy attach myminio student-policy --user teststudent**

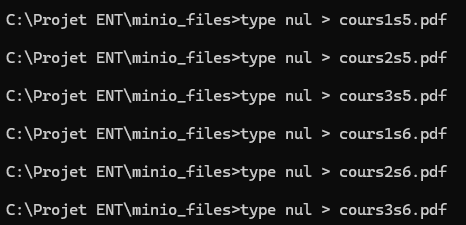


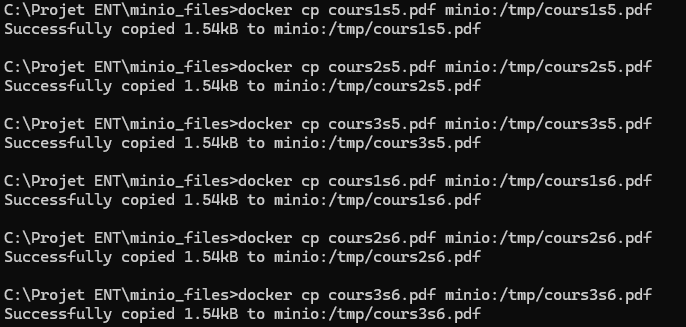
Maintenant pour tester les policies on va créer un alias pour se connecter en tant que testprof et un alias pour se connecter en tant que teststudent.

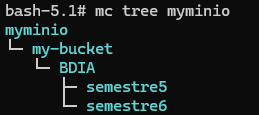
Mais tout d’abord on va créer des fichiers pour tester les policies.  
  


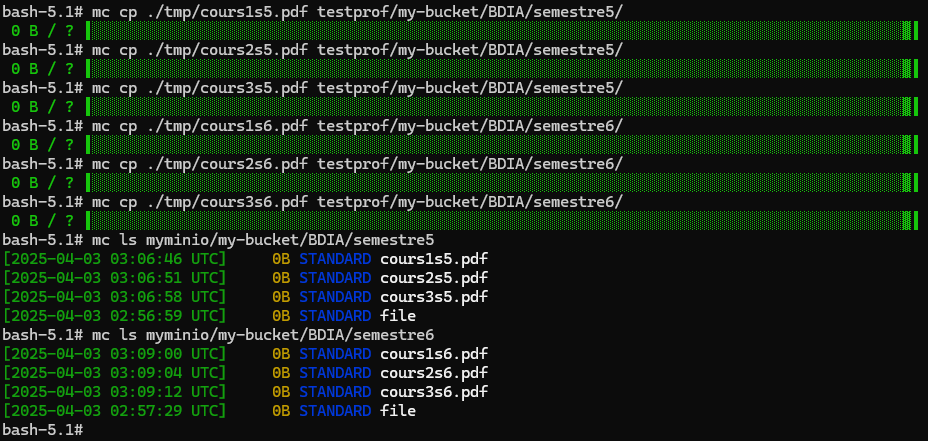
Créer un alias pour se connecter en tant que testprof :  
 **mc alias set testprof http://localhost:9000 testprof profpassword  
**Envoyer un fichier :  
 **mc cp ./tmp/moncours1.pdf testprof/my-bucket/  
 mc cp ./tmp/moncours2.pdf testprof/my-bucket/  
**lister les fichiers du bucket :  
 **mc ls testprof/my-bucket  
**supprimer un fichier : **mc rm testprof/my-bucket/moncours1.pdf**   
telecharger un fichier(pour tester cette fonctionnalité on doit d’abord supprimer l’un des deux fichiers du répertoire local pour vérifier son installation depuis le bucket ) : **mc cp testprof/my-bucket/moncours2.pdf ./tmp/  
**Maintenant pour tester les policies de l’étudiant

Créer un alias pour se connecter en tant que teststudent :  
 **mc alias set teststudent http://localhost:9000 teststudent studentpassword  
**télécharger un fichier : **mc cp teststudent/my-bucket/moncours2.pdf ./tmp/  
**Envoyer un fichier (non-permissible) :  
 **mc cp ./tmp/moncours1.pdf teststudent/my-bucket/  
**lister les fichiers du bucket :  
 **mc ls teststudent/my-bucket  
**supprimer un fichier (non-permissible) : **mc rm teststudent/my-bucket/moncours1.pdf  
**

Apres avoir vérifier les politiques, on va maintenant créer l’arborescence de notre projet.  
on commence par créer les fichiers qu’on va utiliser avec les commandes :  


Ensuite on va les copier dans le conteneur minio :  


Ensuite on va créer les deux dossiers en utilisant un fichier [file] :  
  
  


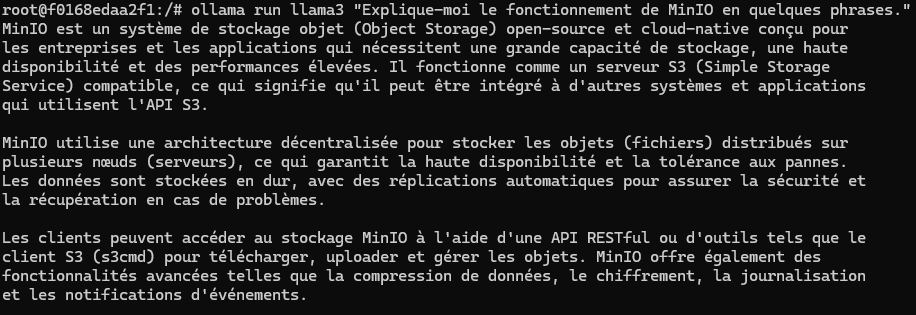
Et la dernière étape est de copier les fichiers de cours de /tmp vers les dossiers dans le bucket  


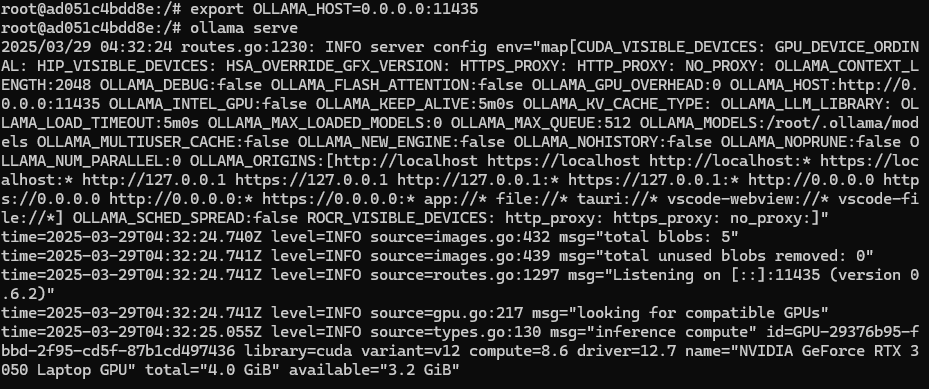
# Etape 4 : configuration et déploiement d’ollama

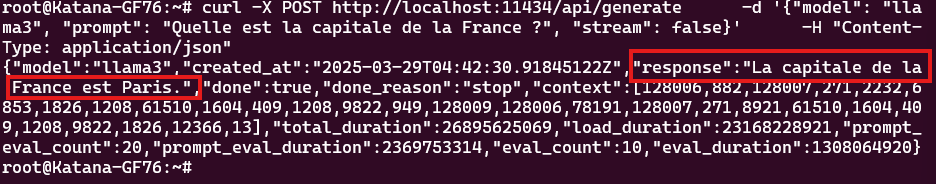
La première étape est d’ouvrir un terminal bash dans le conteneur ollama [ **docker exec -it ollama bash** ]

Ensuite on vérifie la présence d’un model dans ollama avec la commande  
**ollama list**



On va tester llama3 avec la commande  
**ollama run llama3 "Explique-moi le fonctionnement de MinIO en quelques phrases."**

Maintenant pour tester l’API ollama, d’abord on va changer le port de la variable d’environnement OLLAMA\_HOST avec la commande  
**export OLLAMA\_HOST=0.0.0.0 :11435**  
on va lancer ollama en mode API avec la commande  
**ollama serve  
**

Et on va tester avec une requête curl (d’après le terminal ubuntu wsl) :  
**curl -X POST http://localhost:11434/api/generate \  
 -d '{"model": "llama3", "prompt": "Quelle est la capitale de la France ?", "stream": false}' \  
 -H "Content-Type: application/json"  
**