

SWARM

Docker Swarm pour la Résilience :
Déploiement d'un Cluster Swarm pour la Continuité d'Activité
dans l'Univers Virtuel avec Docker et Debian .

Introduction

Docker Swarm est une solution d'orchestration de conteneurs permettant de gérer efficacement des applications distribuées à grande échelle. Il regroupe plusieurs hôtes Docker en un cluster, assurant ainsi haute disponibilité, scalabilité et reprise rapide en cas de défaillance.

Grâce à son intégration native avec Docker et sa facilité de mise en œuvre, Docker Swarm constitue une alternative performante pour la gestion de l'infrastructure conteneurisée.

Rejoignez notre mission : Déployer un Cluster Swarm pour la Continuité et la Reprise d'Activité

Nous recherchons des experts en administration système et DevOps pour concevoir et mettre en place un cluster Swarm robuste dédié à la Planification de la Continuité d'Activité (PCA) et à la Reprise d'Activité (PRA). Ce cluster garantira la résilience et la disponibilité des services critiques dans les environnements les plus exigeants.

Architecture du Cluster

Notre infrastructure reposera sur un ensemble de machines virtuelles Debian assurant une gestion optimisée et sécurisée des conteneurs Docker :

➔ Nœud de contrôle (Manager) : Une VM Debian dédiée à l'orchestration du cluster Swarm, garantissant la répartition intelligente des charges et la gestion des défaillances.

➔ Nœuds de calcul (Workers) : Plusieurs VM Debian fournissant la puissance de traitement nécessaire pour exécuter les conteneurs applicatifs, assurant ainsi scalabilité et redondance.

➔ Stockage persistant (NFS) : Une VM dédiée à l'hébergement des volumes Docker, permettant une conservation fiable des données et une récupération rapide en cas de besoin.

Déploiement des Conteneurs Critiques

Le cluster Swarm hébergera une gamme de services conteneurisés assurant la continuité des opérations :

➔ Registry interne (Local Repository) : Stockage sécurisé des artefacts logiciels pour garantir l'intégrité et la disponibilité des applications.

➔ Base de données (MariaDB) : Système de gestion des données critique, déployé en haute disponibilité pour assurer une continuité de service optimale.

➔ Serveur applicatif (PHP) : Conteneur fournissant l'environnement d'exécution nécessaire aux applications métier.

➔ Proxy inverse et serveur web (Nginx) : Garant de l'accessibilité des services et de la répartition des requêtes entrantes.

➔ Environnement de développement (VSCode Server) : Plateforme collaborative pour la gestion et l'évolution des applications en temps réel.

Votre rôle

En tant qu'ingénieur DevOps ou administrateur système, vous serez chargé de :

- ➔ Concevoir et déployer l'architecture Swarm.
- ➔ Assurer la haute disponibilité et la redondance des services.
- ➔ Mettre en place les stratégies de sauvegarde et de récupération.
- ➔ Optimiser la gestion des ressources et l'orchestration des conteneurs.
- ➔ Garantir la sécurité et la résilience de l'infrastructure.

Rejoignez-nous

Si vous êtes passionné par la résilience technologique et que vous souhaitez participer à un projet stratégique garantissant la continuité et la sécurité des services numériques, cette mission est faite pour vous. Ensemble, bâtissons une infrastructure robuste et adaptable face aux défis du numérique.

Déploiement technique :

Installer une VM Swarm avec Debian 12.

Installer Docker depuis le **dépôt officiel Docker** sur Debian :

```
install -m 0755 -d /etc/apt/keyrings
```

Crée le dossier /etc/apt/keyrings avec les bons droits (0755 → lecture/écriture pour root, lecture/exécution pour les autres).

Ce dossier est prévu pour stocker les clés GPG qui authentifient les dépôts externes.

```
curl -fsSL https://download.docker.com/linux/debian/gpg |  
gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

Télécharge la **clé GPG officielle de Docker** (fichier gpg) via curl.

Convertit cette clé en format binaire (gpg --dearmor), puis la place dans /etc/apt/keyrings/docker.gpg.

C'est indispensable pour que apt puisse vérifier que les paquets viennent bien du dépôt Docker officiel.

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]  
https://download.docker.com/linux/debian $(lsb_release -cs) stable" | tee  
/etc/apt/sources.list.d/docker.list > /dev/null
```

Ajoute le dépôt Docker dans le fichier /etc/apt/sources.list.d/docker.list.

- arch=\$(dpkg --print-architecture) → adapte à ton architecture (amd64, arm64, etc.).

- `$(lsb_release -cs)` → insère le nom de la version Debian (par ex. bookworm pour Debian 12).
- `signed-by=/etc/apt/keyrings/docker.gpg` → indique à apt quelle clé utiliser pour vérifier les signatures.

```
root@Swarm:/home/safia# install -m 0755 -d /etc/apt/keyrings
root@Swarm:/home/safia# curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
bash: sudo : commande introuvable
curl: (23) Failed writing body
root@Swarm:/home/safia# install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg
root@Swarm:/home/safia# echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \
$(lsb_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Configuration du cluster Swarm

Initialiser Swarm sur le manager :

docker swarm init --advertise-addr 192.168.17.129

```
root@Swarm:/home/safia# docker swarm init --advertise-addr 192.168.17.129
Swarm initialized: current node (mzbq7oznuperry4p8v6pacywb) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-51uptvgna9lbqlf4cv1x3ehj417njwxnwhzokmgeeuzlfweecj-0e2nvr7zyyagbhgmdsit52wwp 192.168.17.129:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
root@Swarm:/home/safia#
```

Cette commande initialise le manager Swarm

La commande pour ajouter des workers :

docker swarm join --token SWMTKN-1-... 192.168.17.129:2377

```
root@debianworker1:/home/safia# docker swarm join --token SWMTKN-1-51uptvgna9lbqlf4cv1x3ehj417njwxnwhzokmgeeuzlfweecj-0e2nvr7zyyagbhgmdsit52wwp 192.168.17.129:2377

This node joined a swarm as a worker.
root@debianworker1:/home/safia#
```

Sur la 2ème VM `debianworker1`, nous installons Debian, puis Docker et on vient de l'ajouter comme worker avec le token généré par le manager sur la VM Swarm.

Le message confirme clairement que ta VM `debianworker1` a bien rejoint le cluster Swarm

This node joined a swarm as a worker.

Pour vérifier le cluster on peut exécuter la commande :
docker node ls

```
root@Swarm:/home/safia# docker node ls
ID                HOSTNAME            STATUS    AVAILABILITY    MANAGER
R STATUS    ENGINE VERSION
mzbq7oznuperry4p8v6pacywb *   Swarm            Ready     Active           Leader
28.1.1
xcskdbfl7fcskab35u3hqybpr    debianworker1    Ready     Active
28.1.1
root@Swarm:/home/safia#
```

Création de la 3ème VM avec worker 2 :
Installation de Docker

```
safia@debian: ~
Dépaquetage de slirp4netns (1.2.0-1) ...
Paramétrage de libip6tc2:amd64 (1.8.9-2) ...
Paramétrage de liberror-perl (0.17029-2) ...
Paramétrage de docker-buildx-plugin (0.23.0-1~debian.12~bookworm) ...
Paramétrage de containerd.io (1.7.27-1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /lib/systemd/system/containerd.service.
Paramétrage de patch (2.7.6-7) ...
Paramétrage de docker-compose-plugin (2.35.1-1~debian.12~bookworm) ...
Paramétrage de libltdl7:amd64 (2.4.7-7~deb12u1) ...
Paramétrage de docker-ce-cli (5:28.1.1-1~debian.12~bookworm) ...
Paramétrage de libslirp0:amd64 (4.7.0-1) ...
Paramétrage de pigz (2.6-1) ...
Paramétrage de libnfnetwork0:amd64 (1.0.2-2) ...
Paramétrage de git-man (1:2.39.5-0+deb12u2) ...
Paramétrage de docker-ce-rootless-extras (5:28.1.1-1~debian.12~bookworm) ...
Paramétrage de slirp4netns (1.2.0-1) ...
Paramétrage de git (1:2.39.5-0+deb12u2) ...
Paramétrage de libnetfilter-contrack3:amd64 (1.0.9-3) ...
Paramétrage de iptables (1.8.9-2) ...
update-alternatives: utilisation de « /usr/sbin/iptables-legacy » pour fournir « /usr/sbin/iptables » (iptables) en mode automatique
update-alternatives: utilisation de « /usr/sbin/ip6tables-legacy » pour fournir « /usr/sbin/ip6tables » (ip6tables) en mode automatique
update-alternatives: utilisation de « /usr/sbin/iptables-nft » pour fournir « /usr/sbin/iptables » (iptables) en mode automatique
update-alternatives: utilisation de « /usr/sbin/ip6tables-nft » pour fournir « /usr/sbin/ip6tables » (ip6tables) en mode automatique
update-alternatives: utilisation de « /usr/sbin/arptables-nft » pour fournir « /usr/sbin/arptables » (arptables) en mode automatique
update-alternatives: utilisation de « /usr/sbin/ebrtables-nft » pour fournir « /usr/sbin/ebrtables » (ebrtables) en mode automatique
Paramétrage de docker-ce (5:28.1.1-1~debian.12~bookworm) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/systemd/system/docker.socket.
Traitement des actions différées (« triggers ») pour man-db (2.11.2-2) ...
Traitement des actions différées (« triggers ») pour libc-bin (2.36-9+deb12u1) ...
root@debian:/home/safia#
```

Étapes pour intégrer sw_worker_2 au cluster :

Sur la VM Swarm (manager) Afficher le token avec la commande :

`docker swarm join-token worker`

Sur la VM `sw_worker 2` on copie le token généré par Swarm en prenant soin de changer l'IP puisque nous avons au préalable changé la configuration réseau de NAT en mode bridge.

```
root@Swarm:/home/safia# docker swarm join-token worker
To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-51uptvgna9lbqlf4cv1x3ehj417njwxnwhzokm
qeeuzlfweecj-0e2nvr7zyyagbhgmdsit52wvp 192.168.17.129:2377

root@Swarm:/home/safia#
```

```
root@debian:/home/safia# docker swarm join --token SWMTKN-1-51uptvgna9lbqlf4
cv1x3ehj417njwxnwhzokmqeeuzlfweecj-0e2nvr7zyyagbhgmdsit52wvp 192.168.1.136:23
77
This node joined a swarm as a worker.
root@debian:/home/safia#
```

On peut vérifier la composition du cluster depuis la VM Swarm avec la commande :

`docker node ls`

cela permet d'afficher la composition du cluster et de voir si chaque élément est actif.

```
root@Swarm:/home/safia# docker node ls
ID                HOSTNAME          STATUS    AVAILABILITY    MANAGER
R STATUS    ENGINE VERSION
mzbq7oznuperry4p8v6pacywb *   Swarm           Ready     Active           Leader
28.1.1
jjktulfdn6yuo9t6q7t4246kj     debian           Ready     Active
28.1.1
59psnp76tti3pa4lmxulgx6x0     debianworker1    Ready     Active
28.1.1
root@Swarm:/home/safia#
```

Créer la VM `nfs_server` :

`apt install nfs-kernel-server -y`

```
safia@nfs: ~  
b/systemd/system/rpcbind.socket.  
Paramétrage de libevent-core-2.1-7:amd64 (2.1.12-stable-8) ...  
Paramétrage de keyutils (1.6.3-2) ...  
Paramétrage de nfs-common (1:2.6.2-4+deb12u1) ...  
  
Creating config file /etc/idmapd.conf with new version  
  
Creating config file /etc/nfs.conf with new version  
Ajout de l'utilisateur système « statd » (UID 103) ...  
Ajout du nouvel utilisateur « statd » (UID 103) avec pour groupe d'appartenan  
ce « nogroup » ...  
Pas de création du répertoire personnel « /var/lib/nfs ».  
Created symlink /etc/systemd/system/multi-user.target.wants/nfs-client.target  
→ /lib/systemd/system/nfs-client.target.  
Created symlink /etc/systemd/system/remote-fs.target.wants/nfs-client.target  
→ /lib/systemd/system/nfs-client.target.  
auth-rpcgss-module.service is a disabled or a static unit, not starting it.  
nfs-idmapd.service is a disabled or a static unit, not starting it.  
nfs-utils.service is a disabled or a static unit, not starting it.  
proc-fs-nfsd.mount is a disabled or a static unit, not starting it.  
rpc-gssd.service is a disabled or a static unit, not starting it.  
rpc-statd-notify.service is a disabled or a static unit, not starting it.  
rpc-statd.service is a disabled or a static unit, not starting it.  
rpc-svcgssd.service is a disabled or a static unit, not starting it.  
rpc_pipefs.target is a disabled or a static unit, not starting it.  
var-lib-nfs-rpc_pipefs.mount is a disabled or a static unit, not starting it.  
Paramétrage de nfs-kernel-server (1:2.6.2-4+deb12u1) ...  
Created symlink /etc/systemd/system/nfs-client.target.wants/nfs-blkmap.servic  
e → /lib/systemd/system/nfs-blkmap.service.  
Created symlink /etc/systemd/system/multi-user.target.wants/nfs-server.servic  
e → /lib/systemd/system/nfs-server.service.  
nfs-mountd.service is a disabled or a static unit, not starting it.  
nfsdclld.service is a disabled or a static unit, not starting it.  
  
Creating config file /etc/exports with new version  
  
Creating config file /etc/default/nfs-kernel-server with new version  
Traitement des actions différées (« triggers ») pour man-db (2.11.2-2) ...  
Traitement des actions différées (« triggers ») pour libc-bin (2.36-9+deb12u1  
0) ...  
root@nfs:/home/safia#
```

Créer un dossier de partage :

```
mkdir -p /srv/nfs/docker_volumes  
chown nobody:nogroup /srv/nfs/docker_volumes  
chmod 777 /srv/nfs/docker_volumes
```

Ajouter l'export dans /etc/exports

```
nano /etc/exports
```

```
safia@nfs: ~  
GNU nano 7.2 /etc/exports *  
# /etc/exports: the access control list for filesystems which may be exported  
# to NFS clients. See exports(5).  
#  
# Example for NFSv2 and NFSv3:  
# /srv/homes hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_>  
#  
# Example for NFSv4:  
# /srv/nfs4 gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)  
# /srv/nfs4/homes gss/krb5i(rw,sync,no_subtree_check)  
#  
/srv/nfs/docker_volumes 192.168.1.0/24(rw,sync,no_subtree_check)  
█
```

Monter le partage NFS sur une VM du cluster :

apt install nfs-common -y

```
safia@Swarm: ~  
Préparation du dépaquetage de .../keyutils_1.6.3-2_amd64.deb ...  
Dépaquetage de keyutils (1.6.3-2) ...  
Sélection du paquet nfs-common précédemment désélectionné.  
Préparation du dépaquetage de .../nfs-common_1%3a2.6.2-4+deb12u1_amd64.deb ..  
.  
Dépaquetage de nfs-common (1:2.6.2-4+deb12u1) ...  
Paramétrage de libnfsidmap1:amd64 (1:2.6.2-4+deb12u1) ...  
Paramétrage de rpcbind (1.2.6-6+b1) ...  
Created symlink /etc/systemd/system/multi-user.target.wants/rpcbind.service →  
/lib/systemd/system/rpcbind.service.  
Created symlink /etc/systemd/system/sockets.target.wants/rpcbind.socket → /li  
b/systemd/system/rpcbind.socket.  
Paramétrage de libevent-core-2.1-7:amd64 (2.1.12-stable-8) ...  
Paramétrage de keyutils (1.6.3-2) ...  
Paramétrage de nfs-common (1:2.6.2-4+deb12u1) ...  
  
Creating config file /etc/idmapd.conf with new version  
  
Creating config file /etc/nfs.conf with new version  
Ajout de l'utilisateur système « statd » (UID 103) ...  
Ajout du nouvel utilisateur « statd » (UID 103) avec pour groupe d'appartenan  
ce « nogroup » ...  
Pas de création du répertoire personnel « /var/lib/nfs ».  
Created symlink /etc/systemd/system/multi-user.target.wants/nfs-client.target  
→ /lib/systemd/system/nfs-client.target.  
Created symlink /etc/systemd/system/remote-fs.target.wants/nfs-client.target  
→ /lib/systemd/system/nfs-client.target.  
auth-rpcgss-module.service is a disabled or a static unit, not starting it.  
nfs-idmapd.service is a disabled or a static unit, not starting it.  
nfs-utils.service is a disabled or a static unit, not starting it.  
proc-fs-nfsd.mount is a disabled or a static unit, not starting it.  
rpc-gssd.service is a disabled or a static unit, not starting it.  
rpc-statd-notify.service is a disabled or a static unit, not starting it.  
rpc-statd.service is a disabled or a static unit, not starting it.  
rpc-svcgssd.service is a disabled or a static unit, not starting it.  
rpc_pipefs.target is a disabled or a static unit, not starting it.  
var-lib-nfs-rpc_pipefs.mount is a disabled or a static unit, not starting it.  
Traitement des actions différées (« triggers ») pour man-db (2.11.2-2) ...  
Traitement des actions différées (« triggers ») pour libc-bin (2.36-9+deb12u1  
0) ...  
root@Swarm:/home/safia#
```

Créer un dossier de montage local :

```
mkdir -p /mnt/nfs_test
```

Monter le dossier NFS :

```
root@Swarm:/home/safia# mount -t nfs 192.168.1.136:/srv/nfs/docker_volumes /m  
nt/nfs_test  
Created symlink /run/systemd/system/remote-fs.target.wants/rpc-statd.service  
→ /lib/systemd/system/rpc-statd.service.  
█
```

Créer le dossier /srv/nfs/docker_volumes/registry :

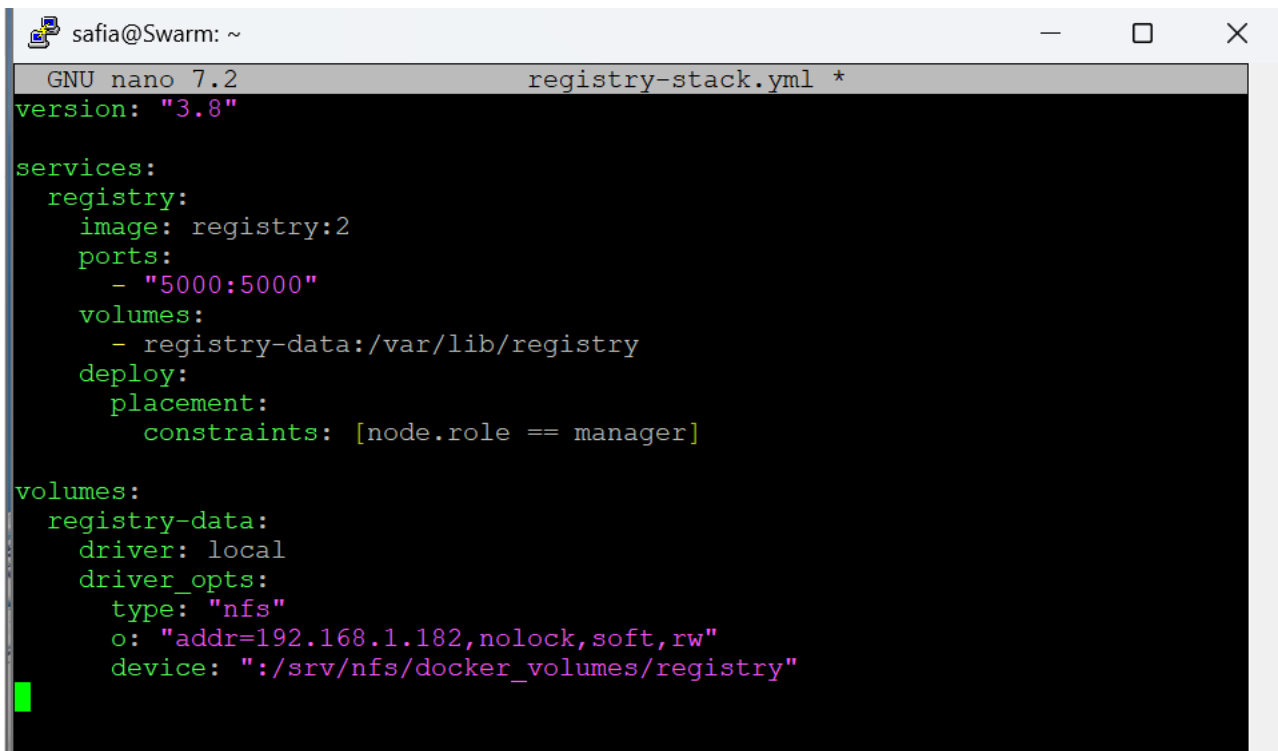

```
root@nfs:/home/safia# mkdir -p /srv/nfs/docker_volumes/registry
root@nfs:/home/safia# chown nobody:nogroup /srv/nfs/docker_volumes/registry
root@nfs:/home/safia# chmod 777 /srv/nfs/docker_volumes/registry
root@nfs:/home/safia# exportfs -ra
root@nfs:/home/safia#
```

Registry Docker privé

C'est la **brique centrale** pour héberger **localement** toutes les images Docker.

Une fois en place, on pourra y **pousser les images personnalisées** (PHP, VSCode, etc.) **sans dépendre d'Internet**.

1. Créer le fichier registry-stack.yml :



```
safia@Swarm: ~
GNU nano 7.2 registry-stack.yml *
version: "3.8"

services:
  registry:
    image: registry:2
    ports:
      - "5000:5000"
    volumes:
      - registry-data:/var/lib/registry
    deploy:
      placement:
        constraints: [node.role == manager]

volumes:
  registry-data:
    driver: local
    driver_opts:
      type: "nfs"
      o: "addr=192.168.1.182,nolock,soft,rw"
      device: ":/srv/nfs/docker_volumes/registry"
```

2. Déployer le service Registry dans le cluster Swarm :

`docker stack deploy -c registry-stack.yml registry`

```
root@Swarm:/home/safia# docker stack deploy -c registry-stack.yml registry
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
Creating network registry_default
Creating service registry_registry
root@Swarm:/home/safia#
```

3. Vérifier que le service tourne :

docker service ps registry_registry

```
root@Swarm:/home/safia# docker service ps registry_registry
ID                NAME                IMAGE                NODE        DESIRED STATE    C
CURRENT STATE    ERROR              PORTS
myaczppafin      registry_registry.1  registry:2          Swarm       Running          R
unning 44 seconds ago
```

Test complet du cluster Docker Swarm avec le registry privé :

1. Télécharger l'image officielle de nginx sur la VM Swarm :

docker pull nginx:latest

```
root@Swarm:/home/safia# docker pull nginx:latest
latest: Pulling from library/nginx
61320b01ae5e: Pull complete
670a101d432b: Pull complete
405bd2df85b6: Pull complete
cc80efff8457: Pull complete
2b9310b2ee4b: Pull complete
6c4aa022e8e1: Pull complete
abddc69cb49d: Pull complete
Digest: sha256:fb39280b7b9eba5727c884a3c7810002e69e8f961cc373b89c92f14961d903
a0
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
root@Swarm:/home/safia#
```

2. Taguer l'image pour le registry privé :

docker tag nginx:latest localhost:5000/nginx-test

3. Pusher l'image dans le registry privé :

docker push localhost:5000/nginx-test

4. Vérifier que tout fonctionne :

```
docker service ls
docker service ps registry_registry
docker ps
```

```
curl http://192.168.1.136:5000/v2/_catalog
{"repositories":["nginx-test"]}
```

safia@Swarm: ~

In a future release, --detach=false will become the default.

Updating service registry_registry (id: djm5y2otcdxag9d9kfwrbesjw)

root@Swarm:/home/safia# docker service ls

ID	NAME	MODE	REPLICAS	IMAGE	PORTS
djm5y2otcdxa	registry_registry	replicated	1/1	registry:2	*:5000->5000/tcp

root@Swarm:/home/safia# docker service ls

ID	NAME	MODE	REPLICAS	IMAGE	PORTS
djm5y2otcdxa	registry_registry	replicated	1/1	registry:2	*:5000->5000/tcp

root@Swarm:/home/safia# docker service ps registry_registry

ID	NAME	MODE	REPLICAS	IMAGE	NODE	DESIRED STATE
b40qn6z15305	registry_registry.1		registry:2	Swarm	Running	
7s48si8bz6ou	_ registry_registry.1		registry:2	Swarm	Shutdown	
7j36uk0sj1l4	_ registry_registry.1		registry:2	Swarm	Shutdown	
z16n99j17j3u	_ registry_registry.1		registry:2	Swarm	Shutdown	
msk4oso6fw8d	_ registry_registry.1		registry:2	Swarm	Shutdown	

root@Swarm:/home/safia# docker ps

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
012836c8e031	registry:2	"/entrypoint.sh /etc..."	10 minutes ago	Up 10 m

root@Swarm:/home/safia# docker push 192.168.1.136:5000/nginx-test

Using default tag: latest

The push refers to repository [192.168.1.136:5000/nginx-test]

941dd9dd8ee4: Pushed

f6e33ee35fd0: Pushed

9fd8b974f616: Pushed

a8b606cdf152: Pushed

cb857378ec55: Pushed

deb7d8874f38: Pushed

ace34d1d784c: Pushed

latest: digest: sha256:e5e2c4be5cea9bf49b2c976c65b4fca33d9d094b276a5e517d8e5748100a3c73 size: 1778

root@Swarm:/home/safia# curl http://192.168.1.136:5000/v2/_catalog

{"repositories":["nginx-test"]}

root@Swarm:/home/safia#

Déployer MariaDB en service Swarm avec :

- Stockage des données persisté via NFS
- Image Docker officielle (mariadb)
- Configuration minimale (mot de passe root, nom de base)
- Un volume NFS dédié pour les données MySQL

1. Préparer le dossier sur le serveur NFS

Sur la VM NFS on exécute les commandes :

```
mkdir -p /srv/nfs/docker_volumes/mariadb
chown nobody:nogroup /srv/nfs/docker_volumes/mariadb
chmod 777 /srv/nfs/docker_volumes/mariadb
exportfs -ra
```

2. Créé le répertoire de montage sur la VM Swarm.

```
mkdir -p /mnt/test_mariadb
mount -t nfs 192.168.1.182:/srv/nfs/docker_volumes/mariadb /mnt/test_mariadb
```

```
root@Swarm:/home/safia# mkdir -p /mnt/test_mariadb
root@Swarm:/home/safia# mount -t nfs 192.168.1.182:/srv/nfs/docker_volumes/mariadb /mnt/test_mariadb
root@Swarm:/home/safia#
```

3. Créer le fichier mariadb-stack.yml sur la VM Swarm

```
mariadb:
  image: mariadb:10.5
  environment:
    MYSQL_ROOT_PASSWORD: root
    MYSQL_DATABASE: testdb
    MYSQL_USER: user
    MYSQL_PASSWORD: userpass
  ports:
    - "3306:3306"
  volumes:
    - mariadb_data:/var/lib/mysql
  networks:
    - proxy-net
  deploy:
    placement:
      constraints: [node.role == manager]
```

```

GNU nano 7.2 mariadb-stack.yml
version: "3.8"

services:
  mariadb:
    image: mariadb:10.6
    environment:
      MYSQL_ROOT_PASSWORD: tropsecure
      MYSQL_DATABASE: projet_pca
    volumes:
      - mariadb-data:/var/lib/mysql
    deploy:
      placement:
        constraints: [node.role == manager]
    ports:
      - "3306:3306"

volumes:
  mariadb-data:
    driver: local
    driver_opts:
      type: "nfs"
      o: "addr=192.168.1.182,nolock,soft,rw"
      device: ":/srv/nfs/docker_volumes/mariadb"

```

Ce fichier a été ensuite intégré à un autre fichier qui gère l'ensemble de tous les services full-stack.yml

4. Déployer le service

```

root@Swarm:/home/safia# docker stack deploy -c mariadb-stack.yml mariadb
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
Creating network mariadb_default
Creating service mariadb_mariadb
root@Swarm:/home/safia#

```

```

Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
Creating network mariadb_default
Creating service mariadb_mariadb

```

ID	NAME	IMAGE	NODE	DESIRED STATE	CURRENT STATE
f9i5ms0dn8lr	mariadb_mariadb.1	mariadb:10.6		Running	Running

```

ending less than a second ago
root@Swarm:/home/safia#

```

Le service mariadb_mariadb.1 est démarré.

5. Vérifier que le conteneur fonctionne :

docker service ps mariadb_mariadb

```

root@Swarm:/home/safia# docker service ps mariadb_mariadb
ID                NAME                ERROR                IMAGE                NODE                DESIRED STATE        PORTS
CURRENT STATE
v0xdeo01xta8     mariadb_mariadb.1   "failed to populate volume: er..." mariadb:10.6        Swarm              Ready
Rejected 1 second ago
nsjcbx7h08       \_ mariadb_mariadb.1 mariadb:10.6        Swarm              Shutdown
Rejected 6 seconds ago "failed to populate volume: er..."
qxi5r94qv9       \_ mariadb_mariadb.1 mariadb:10.6        Swarm              Shutdown
Rejected 11 seconds ago "failed to populate volume: er..."
hs5zipi2hk5s     \_ mariadb_mariadb.1 mariadb:10.6        Swarm              Shutdown
Rejected 16 seconds ago "failed to populate volume: er..."
uor118mlbcz      \_ mariadb_mariadb.1 mariadb:10.6        Swarm              Shutdown
Rejected 21 seconds ago "failed to populate volume: er..."
root@Swarm:/home/safia#

```

Le service mariadb_mariadb échoue avec l'erreur :

"failed to populate volume: error ..."

Malgré un export NFS correct sur la VM nfs, MariaDB ne parvenait pas à accéder au dossier partagé.

Solution apportée

1. Tester le volume Docker manuellement :

Sur la VM Swarm, on exécute les commandes, comme dans le fichier docker compose :

```

docker volume create \
--driver local \
--opt type=nfs \
--opt o=addr=192.168.1.182,nolock,soft,rw \
--opt device=:/srv/nfs/docker_volumes/mariadb \
test-mariadb-volume

```

```

root@Swarm:/home/safia# docker volume create \
--driver local \
--opt type=nfs \
--opt o=addr=192.168.1.182,nolock,soft,rw \
--opt device=:/srv/nfs/docker_volumes/mariadb \
test-mariadb-volume
test-mariadb-volume

```

2. Lancer un conteneur simple pour tester le volume :

```

docker run --rm -it \
--mount source=test-mariadb-volume,target=/data \
alpine sh

```

Puis dans le conteneur Alpine :

```

echo "test" > /data/hello.txt
cat /data/hello.txt

```

```

root@Swarm:/home/safia# docker run --rm -it \
--mount source=test-mariadb-volume,target=/data \
alpine sh
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
fe07684b16b8: Pull complete
Digest: sha256:8alf59ffb675680d47db6337b49d22281a139e9d709335b492be023728e117
15
Status: Downloaded newer image for alpine:latest
/ #
/ # echo "salut mariadb" > /data/test.txt
/ # cat /data/test.txt
salut mariadb
/ # █

```

3. vérifier le contenu du volume sur la VM NFS

Sur la VM NFS :

ls -l /srv/nfs/docker_volumes/mariadb

```

root@nfs:/home/safia# ls -l /srv/nfs/docker_volumes/mariadb
total 4
-rw-r--r-- 1 root root  0 31 mai  23:56 hello.txt
-rw-r--r-- 1 root root 14  1 juin  22:43 test.txt
root@nfs:/home/safia# nano /etc/exports
root@nfs:/home/safia# █

```

La persistance NFS fonctionne parfaitement

On reteste :

1. Redéployer le service MariaDB

```

docker stack rm mariadb_test
sleep 5
docker stack deploy -c mariadb-stack.yml mariadb_test

```

```

root@Swarm:/home/monitor/mariadb-test# docker stack rm mariadb_test
sleep 5
docker stack deploy -c mariadb-stack.yml mariadb_test
Removing service mariadb_test_mariadb
Removing network mariadb_test_default
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
Creating network mariadb_test_default
Creating service mariadb_test_mariadb

```

2. Vérifier que le service est bien lancé :

```

docker service ps mariadb_test_mariadb

```

```

root@Swarm:/home/monitor/mariadb-test# docker service ps mariadb_test_mariadb

```

ID	NAME	IMAGE	NODE	DESIRED STATE
mprvf8x2xv0b	mariadb_test_mariadb.1	mariadb:10.5	Swarm	Running

```

Preparing 22 seconds ago
root@Swarm:/home/monitor/mariadb-test# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
root@Swarm:/home/monitor/mariadb-test# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES

```

3. Connexion à MariaDB depuis l'intérieur du conteneur :

```
docker exec -it $(docker ps --filter name=full_test_mariadb -q) mysql -u root -p
```

```

root@Swarm:/home/monitor# docker exec -it $(docker ps --filter name=full_test_mariadb -q) mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 3
Server version: 10.5.29-MariaDB-ubu2004 mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement
.

MariaDB [(none)]> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| testdb |
+-----+
4 rows in set (0.003 sec)

MariaDB [(none)]> █

```

On réussi à se connecter à Mariadb, le service est activé donc on reviendra plus tard, dans la partie sauvegarde pour la connexion avec NFS.

Déploiement de PHP dans Docker Swarm

Objectif

Déployer un conteneur PHP dans Swarm, avec persistance possible via NFS si souhaité, pour exécuter des scripts PHP dans un service.

Étape 1 : Créer un dossier de travail :

```

mkdir -p /home/monitor/php-test
cd /home/monitor/php-test

```

Étape 2 : Créer un fichier index.php


```
safia@Swarm: ~  
GNU nano 7.2 index.php *  
<?php  
echo "Hello from PHP in Docker Swarm!";  
█
```

Étape 3 : Créer un Dockerfile

```
safia@Swarm: ~  
GNU nano 7.2 Dockerfile *  
FROM php:8.1-cli  
COPY index.php /var/www/html/  
WORKDIR /var/www/html  
CMD ["php", "index.php"]  
█
```

Étape 4 : Créer un docker-compose.yml

```
safia@Swarm: ~  
GNU nano 7.2 php-stack.yml *  
version: "3.8"  
  
services:  
  php_app:  
    image: php:8.2-cli  
    volumes:  
      - type: bind  
        source: /home/monitor/php-test  
        target: /var/www/html  
    command: php -S 0.0.0.0:80 -t /var/www/html  
    deploy:  
      placement:  
        constraints: [node.role == manager]  
    ports:  
      - "8081:80"  
█
```

Étape 5 : Lancer le service dans Swarm :

`docker stack deploy -c php-stack.yml php_test`

```
root@Swarm:/home/monitor/php-test# docker stack deploy -c php-stack.yml php_test  
Since --detach=false was not specified, tasks will be created in the background.  
In a future release, --detach=false will become the default.  
Creating service php_test_php_app  
root@Swarm:/home/monitor/php-test# █
```

Étape 6 tester PHP :

1. Vérifier que le conteneur PHP fonctionne :

`docker service ps php_test_php_app`

On peut voir l'état Running.

```
root@Swarm:/home/monitor/php-test# docker service ps php_test_php_app
ID                NAME                IMAGE             NODE    DESIRED STATE
CURRENT STATE    ERROR
oy7r0uymppxx1    php_test_php_app.1  php:8.2-cli      Swarm  Running
Running 14 seconds ago
mwtmubadx84p     \_ php_test_php_app.1  php:8.2-cli      Swarm  Shutdown
Rejected 18 seconds ago "invalid mount config for type..."
wsbgvxe4ptla     \_ php_test_php_app.1  php:8.2-cli      Swarm  Shutdown
Rejected 23 seconds ago "invalid mount config for type..."
rwemwl998vu3     \_ php_test_php_app.1  php:8.2-cli      Swarm  Shutdown
Rejected 28 seconds ago "invalid mount config for type..."
06pk9bdkcc9r     \_ php_test_php_app.1  php:8.2-cli      Swarm  Shutdown
Rejected 33 seconds ago "invalid mount config for type..."
```

2. Vérifier que le conteneur PHP fonctionne dans Docker Swarm

depuis un navigateur web on tape :

<http://192.168.1.136:8081>

Accueil - Google Drive

PHP 8.2.28 - phpinfo()

+

-

←

→

↻

⚠ Non sécurisé

192.168.1.136:8081

☆

🎵

S

⋮

PHP Version 8.2.28

System	Linux 8625de36f236 6.1.0-34-amd64 #1 SMP PREEMPT
Build Date	May 21 2025 23:16:25
Build System	Linux - Docker
Build Provider	https://github.com/docker-library/php
Configure Command	'./configure' '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php/conf.d' '--enable-option-checking=fatal-errors' '--enable-mysqlnd' '--with-password-argon2' '--with-sodium=shared' '--with-iconv' '--with-openssl' '--with-readline' '--with-zlib' '--enable-xml' '--enable-xmlrpc' '--enable-sockets' '--enable-pcntl' '--enable-ftp' '--enable-gd' '--enable-gd-native-ttf' '--enable-gd-jpeg' '--enable-gd-t2' '--enable-gd-freetype' '--enable-gd-fontconfig' '--enable-gd-webp' '--enable-gd-xpm' '--enable-gd-tar' '--enable-gd-lzma' '--enable-gd-bz2' '--enable-gd-zstd' '--enable-gd-lz4' '--enable-gd-lz' '--enable-gd-lzbc' '--enable-gd-lzss' '--enable-gd-lzma' '--enable-gd-lz4' --enable-gd-lz' 'build_
Server API	Built-in HTTP server
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	(none)
Scan this dir for additional .ini files	/usr/local/etc/php/conf.d
Additional .ini files parsed	/usr/local/etc/php/conf.d/docker-php-ext-sodium.ini
PHP API	20220829
PHP Extension	20220829
Zend Extension	420220829
Zend Extension Build	API420220829,NTS
PHP Extension Build	API20220829,NTS
Debug Build	no
Thread Safety	disabled

Déploiement de Nginx en reverse proxy dans Docker Swarm


Objectif :

Faire en sorte que Nginx agisse comme un reverse proxy devant un service PHP déjà déployé (sur le port 8081), par exemple pour rediriger le trafic HTTP depuis le port 80 vers le service PHP.

1. Créer un répertoire pour le test :

```
mkdir -p /home/monitor/nginx-test
cd /home/monitor/nginx-test
```

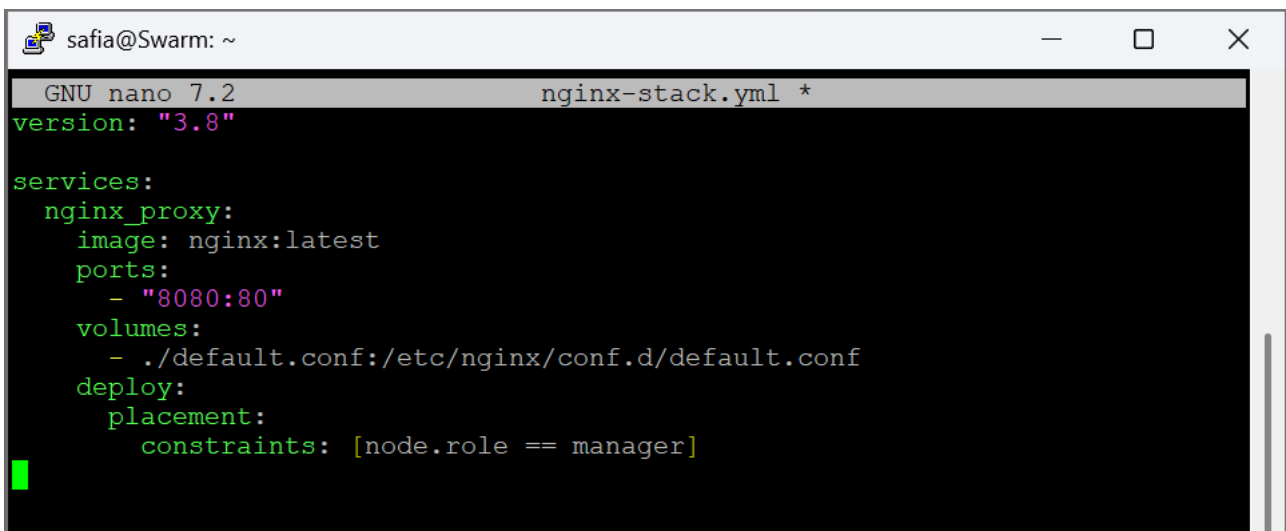
2. Créer un fichier de configuration Nginx default.conf



```
safia@Swarm: ~
GNU nano 7.2 default.conf *
server {
    listen 80;

    location / {
        proxy_pass http://php_test_php_app:80;
        resolver 127.0.0.11;
    }
}
```

3. Créer un nginx-stack.yml



```
safia@Swarm: ~
GNU nano 7.2 nginx-stack.yml *
version: "3.8"

services:
  nginx_proxy:
    image: nginx:latest
    ports:
      - "8080:80"
    volumes:
      - ./default.conf:/etc/nginx/conf.d/default.conf
    deploy:
      placement:
        constraints: [node.role == manager]
```

4. Déployer la stack

```
root@Swarm:/home/monitor/nginx-test# docker stack deploy -c nginx-stack.yml nginx_test
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
Creating network nginx_test_default
Creating service nginx_test_nginx_proxy
root@Swarm:/home/monitor/nginx-test#
```

5. Tester le reverse proxy

curl <http://192.168.1.136:8080>

```
root@Swarm:/home/monitor/nginx-test# curl http://192.168.1.136:8080
curl: (7) Failed to connect to 192.168.1.136 port 8080 after 5 ms: Couldn't connect to server
```

Problème identifié :
Dans les **logs Nginx** :

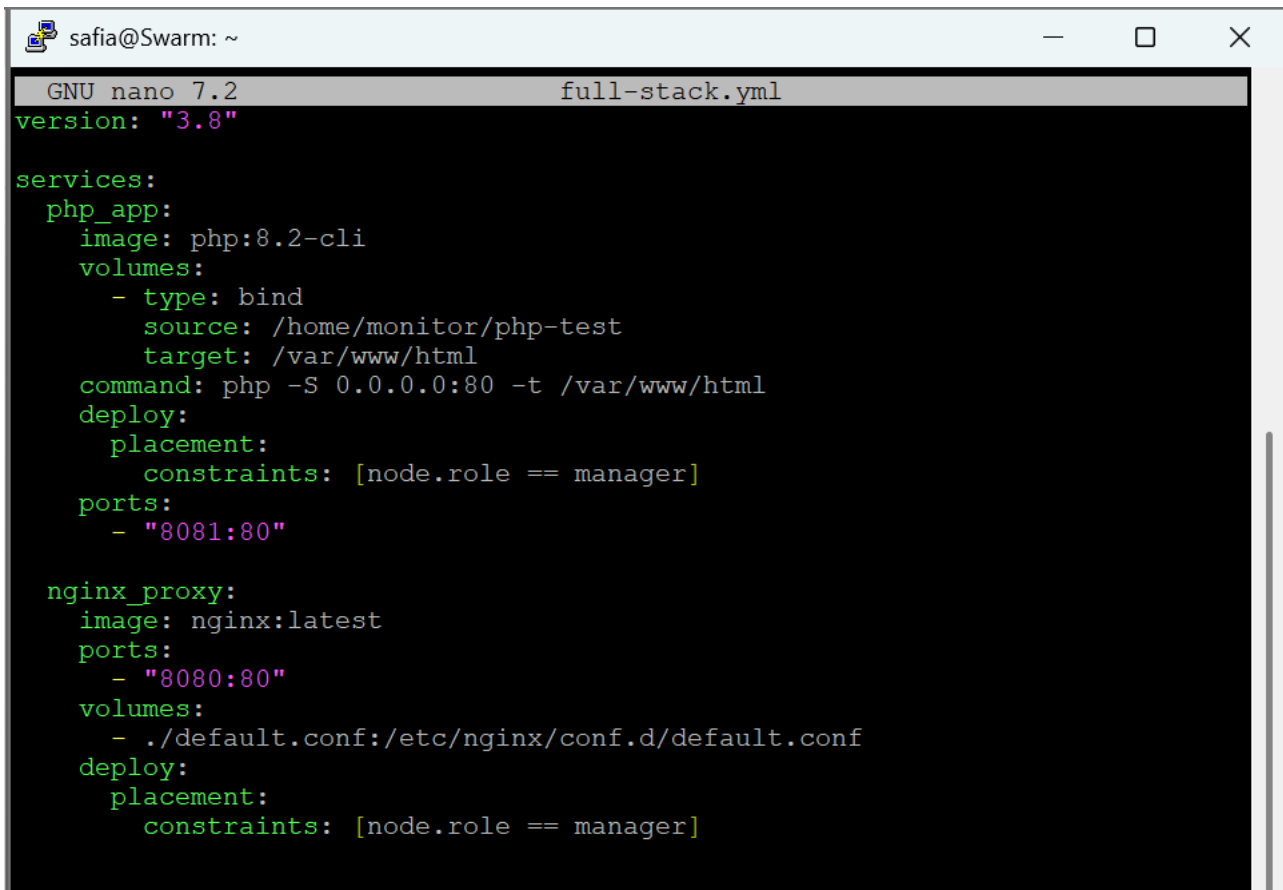
[emerg] host not found in upstream "php_app"

Cela signifie que le service php_app n'est pas accessible depuis le conteneur Nginx.
On a défini php_app dans un autre stack (php-stack.yml), Docker Swarm les isole par défaut.

Solution : utiliser le même stack pour Nginx et PHP

Les services doivent être **dans le même fichier stack** pour partager le même réseau Docker Swarm **par défaut**.

1. Créer un fichier unique full-stack.yml :

A screenshot of a terminal window titled 'safia@Swarm: ~'. The window shows the nano 7.2 editor editing a file named 'full-stack.yml'. The content of the file is a Docker Swarm stack configuration. It defines two services: 'php_app' and 'nginx_proxy'. 'php_app' uses the 'php:8.2-cli' image, binds a local directory to a container volume, and runs a PHP server on port 80. 'nginx_proxy' uses the 'nginx:latest' image and serves the default nginx configuration on port 80. Both services are constrained to run on manager nodes.

```
GNU nano 7.2 full-stack.yml
version: "3.8"

services:
  php_app:
    image: php:8.2-cli
    volumes:
      - type: bind
        source: /home/monitor/php-test
        target: /var/www/html
    command: php -S 0.0.0.0:80 -t /var/www/html
    deploy:
      placement:
        constraints: [node.role == manager]
    ports:
      - "8081:80"

  nginx_proxy:
    image: nginx:latest
    ports:
      - "8080:80"
    volumes:
      - ./default.conf:/etc/nginx/conf.d/default.conf
    deploy:
      placement:
        constraints: [node.role == manager]
```

2. Supprimer les anciens stacks

`docker stack rm php_test`

`docker stack rm nginx_test`

3. Déployer le nouveau stack :

`cd /home/monitor/nginx-test`

`docker stack deploy -c full-stack.yml full_test`

4. Tester l'accès :

Depuis un navigateur web on accède à la page PHP via Nginx

<http://192.168.1.136:8080>

<div> <div> <div> <div> <div></div> <div>Accueil - Google Drive</div> <div></div> </div> <div> <div></div> <div>PHP 8.2.28 - phpinfo()</div> <div></div> </div> </div> <div> <div></div> <div></div> <div></div> </div> </div> </div>	
<div> <div> <div> <div></div> <div></div> <div></div> </div> <div> <div>Non sécuri</div> <div>192.16...</div> <div></div> </div> <div> <div></div> <div></div> <div></div> </div> </div> <div> <div>Redémarrer pour mettre à jour</div> </div> </div>	
<div> <div> <div> <div></div> <div>PHP Version 8.2.28</div> </div> </div> </div>	
System	Linux 4b41de00fa23 6.1.0-34-amd64 #1 SMP PREEMP
Build Date	May 21 2025 23:16:25
Build System	Linux - Docker
Build Provider	https://github.com/docker-library/php
Configure Command	'./configure' '--build=x86_64-linux-gnu' '--with-config-file-p dir=/usr/local/etc/php/conf.d' '--enable-option-checking=f mysqlnd' '--with-password-argon2' '--with-sodium=share with-iconv' '--with-openssl' '--with-readline' '--with-zlib' '--e with-libdir=lib/x86_64-linux-gnu' '--enable-embed' 'build_
Server API	Built-in HTTP server
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	(none)
Scan this dir for additional .ini files	/usr/local/etc/php/conf.d
Additional .ini files parsed	/usr/local/etc/php/conf.d/docker-php-ext-sodium.ini
PHP API	20220829
PHP Extension	20220829
Zend Extension	420220829
Zend Extension Build	API420220829,NTS
PHP Extension Build	API20220829,NTS
Debug Build	no
Thread Safety	disabled

Objectif

Déployer **VSCode Server (code-server)** dans Docker Swarm, accessible via le **reverse proxy Nginx**, par exemple sur `http://192.168.1.136:8080/code` ou autre.

1. Choix de l'image

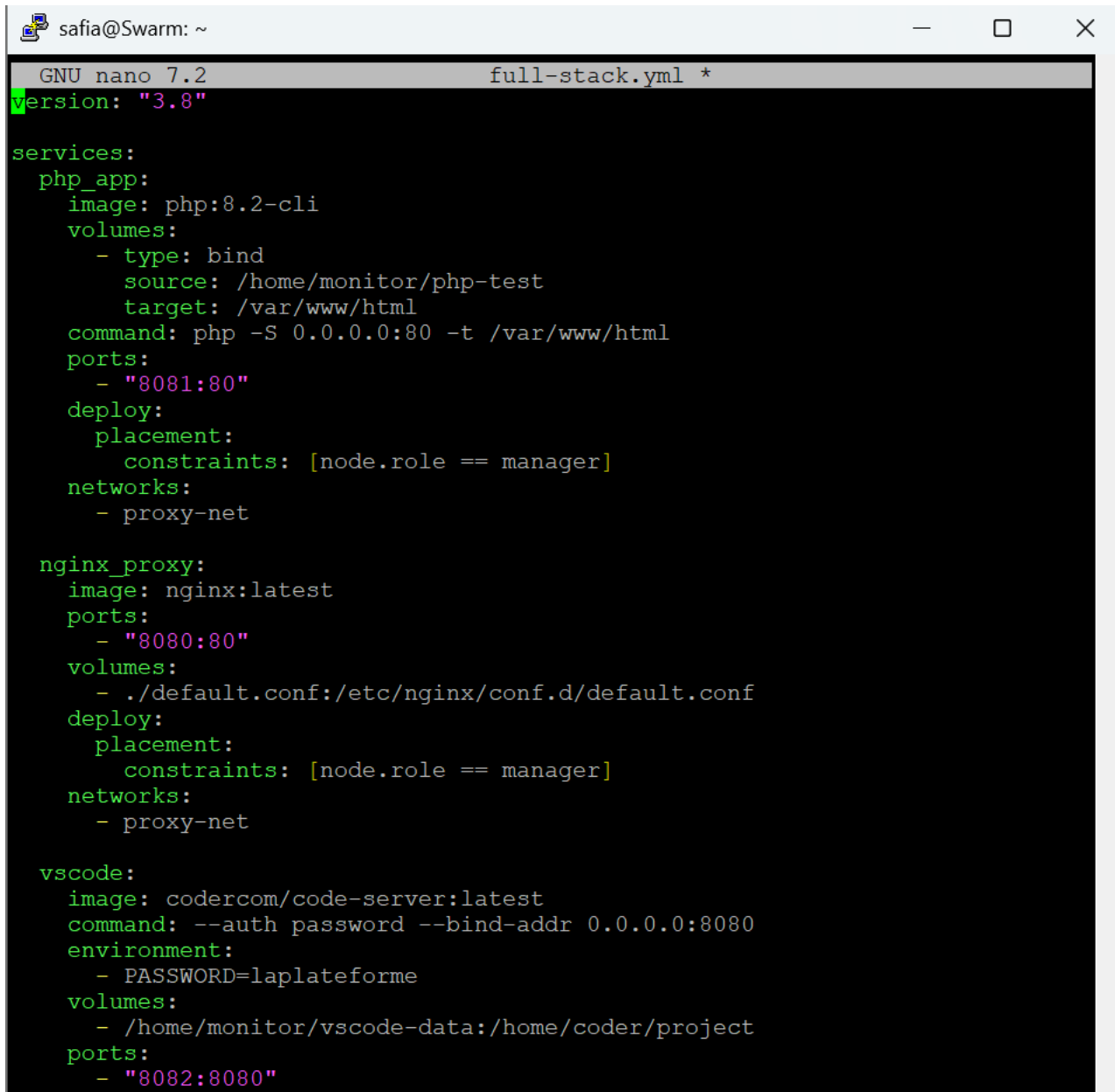
On utilise l'image officielle de `codercom/code-server` :

2. Préparation du volume

Créer un dossier sur l'hôte pour stocker les fichiers VSCode persistants :

```
mkdir -p /home/monitor/vscode-data
```

On ajoute une partie VSCode dans la stack de la configuration Swarm :



```
safia@Swarm: ~  
GNU nano 7.2 full-stack.yml *  
version: "3.8"  
  
services:  
  php_app:  
    image: php:8.2-cli  
    volumes:  
      - type: bind  
        source: /home/monitor/php-test  
        target: /var/www/html  
    command: php -S 0.0.0.0:80 -t /var/www/html  
    ports:  
      - "8081:80"  
    deploy:  
      placement:  
        constraints: [node.role == manager]  
    networks:  
      - proxy-net  
  
  nginx_proxy:  
    image: nginx:latest  
    ports:  
      - "8080:80"  
    volumes:  
      - ./default.conf:/etc/nginx/conf.d/default.conf  
    deploy:  
      placement:  
        constraints: [node.role == manager]  
    networks:  
      - proxy-net  
  
  vscode:  
    image: codercom/code-server:latest  
    command: --auth password --bind-addr 0.0.0.0:8080  
    environment:  
      - PASSWORD=laplateforme  
    volumes:  
      - /home/monitor/vscode-data:/home/coder/project  
    ports:  
      - "8082:8080"
```

```

    deploy:
      placement:
        constraints: [node.role == manager]
    networks:
      - proxy-net

mariadb:
  image: mariadb:10.5
  environment:
    MYSQL_ROOT_PASSWORD: root
    MYSQL_DATABASE: testdb
    MYSQL_USER: user
    MYSQL_PASSWORD: userpass
  ports:
    - "3306:3306"
  volumes:
    - mariadb_data:/var/lib/mysql
  deploy:
    placement:
      constraints: [node.role == manager]
    networks:
      - proxy-net

networks:
  proxy-net:
    driver: overlay

volumes:
  mariadb_data:

```

^G Aide ^O Écrire ^W Chercher ^K Couper ^T Exécuter
 ^X Quitter ^R Lire fich. ^\ Remplacer ^U Coller ^J Justifier

Configuration du proxy Nginx (default.conf)

Ajouter le bloc dans le fichier default.conf :

```

safia@Swarm: ~
GNU nano 7.2 default.conf *
server {
    listen 80;

    location /code/ {
        proxy_pass http://vscode:8080/;
        proxy_http_version 1.1;
        proxy_set_header Host $host;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Accept-Encoding gzip;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        rewrite ^/code(/.*)$ $1 break;
    }

    location / {
        return 404;
    }
}

```


Ce bloc permet d'accéder à VSCode via :
<http://192.168.1.136:8080/code/>

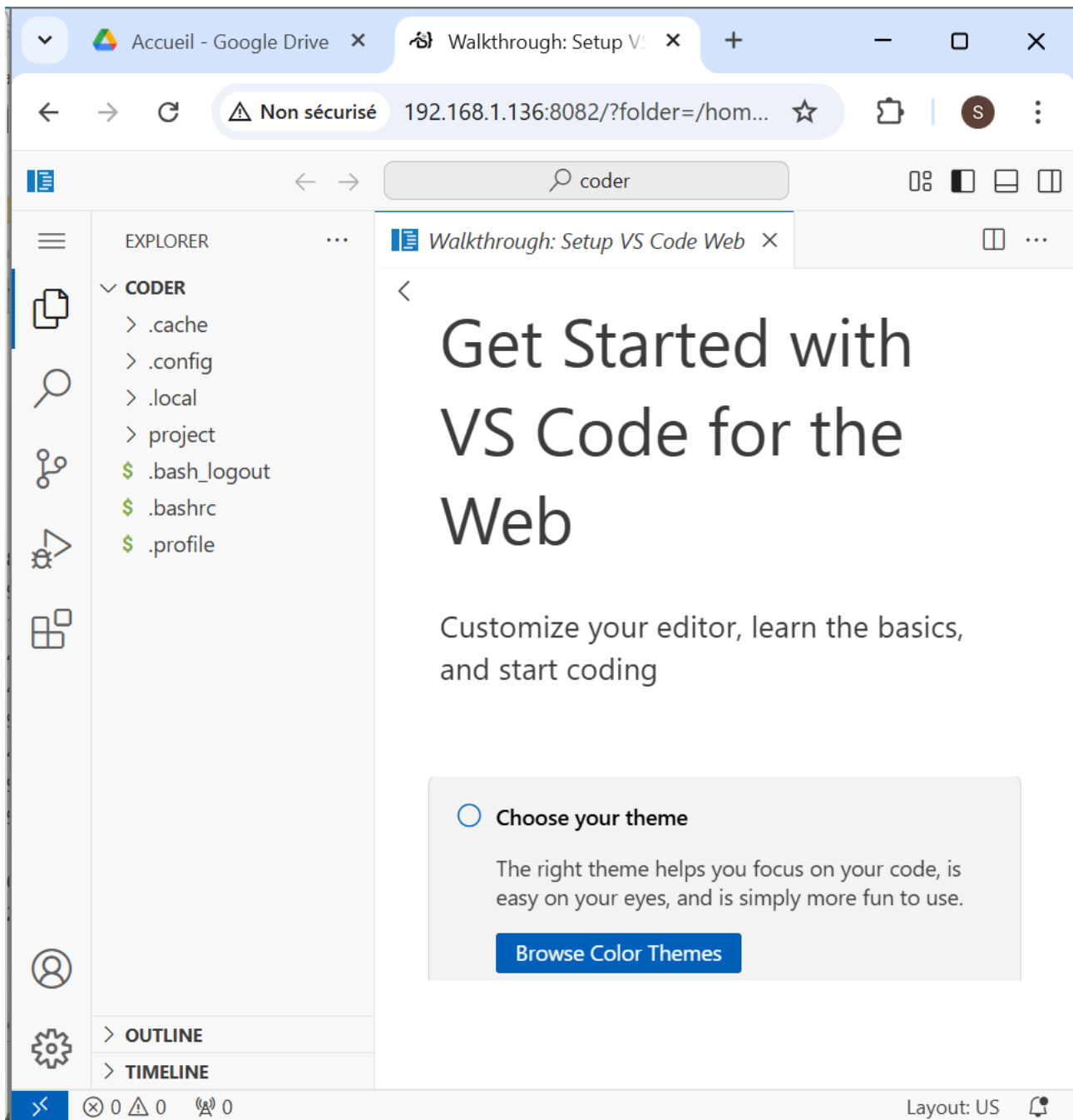
Déploiement

`docker stack deploy -c full-stack.yml full_test`

```
root@Swarm:/home/monitor/nginx-test# docker stack deploy -c full-stack.yml full_test
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
Creating network full_test_proxy-net
Updating service full_test_vscode (id: 1kir0u17zk04okhi6yl02m9mk)
Creating service full_test_php_app
Updating service full_test_nginx_proxy (id: o86ffwzfgpdfzuqm88srqzo3l)
root@Swarm:/home/monitor/nginx-test#
```

Dans le navigateur web :

<http://192.168.1.136:8082/>



Stockage persistant avec NFS :

- MariaDB est configuré pour écrire dans un volume Docker.
- Ce volume est désormais redirigé vers un **partage NFS**.
- Ce NFS peut être sur un autre hôte pour une meilleure tolérance aux pannes.

➔ C'est la base d'une sauvegarde centralisée.

Comment activer l'accès à MariaDB dans Docker Swarm et monter un volume NFS pour la sauvegarde.

1. Montage du volume NFS sur la VM Swarm

```
mkdir -p /mnt/test-mariadb
```

```
mount -t nfs 192.168.1.182:/srv/nfs/mariadb /mnt/test-mariadb
```

```
root@Swarm:/home/monitor# mount -t nfs 192.168.1.182:/srv/nfs/mariadb /mnt/test-mariadb
root@Swarm:/home/monitor# mkdir -p /srv/nfs/docker_volumes/mariadb
chmod 777 /srv/nfs/docker_volumes/mariadb
chown nobody:nogroup /srv/nfs/docker_volumes/mariadb
root@Swarm:/home/monitor# nano full-stack.yml
root@Swarm:/home/monitor#
```

2. Fichier full-stack.yml :

```
mariadb:
  image: mariadb:10.5
  environment:
    MYSQL_ROOT_PASSWORD: root
    MYSQL_DATABASE: testdb
    MYSQL_USER: user
    MYSQL_PASSWORD: userpass
  ports:
    - "3306:3306"
  volumes:
    - mariadb_data:/var/lib/mysql
  deploy:
    placement:
      constraints: [node.role == manager]
  networks:
    - proxy-net

networks:
  proxy-net:
    driver: overlay

volumes:
  mariadb_data:
```

Ce service crée une base MariaDB avec :

- utilisateur root (mot de passe root)

- base testdb
- utilisateur user (mot de passe userpass)

3. Déployer :

```
docker stack deploy -c full-stack.yml full_test
sleep 5
docker stack deploy -c full-stack.yml full_test
```

```
root@Swarm:/home/monitor# docker stack rm full_test
sleep 5
docker stack deploy -c full-stack.yml full_test
Nothing found in stack: full_test
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
Creating network full_test_proxy-net
Creating service full_test_mariadb
Creating service full_test_php_app
Creating service full_test_nginx_proxy
Creating service full_test_vscode
root@Swarm:/home/monitor# docker service ls
```

ID	NAME	MODE	REPLICAS	IMAGE
iqwrp594z823	full_test_mariadb	replicated	1/1	mariadb:10.5
9pxbb45g13tu	full_test_nginx_proxy	replicated	1/1	nginx:latest
1zcbrylba34y	full_test_php_app	replicated	1/1	php:8.2-cli
z4elb42wk3mm	full_test_vscode	replicated	1/1	codercom/code-server:latest

Vérifier que Mariadb fonctionne bien :

```
root@Swarm:/home/monitor# docker service ps full_test_mariadb
```

ID	NAME	IMAGE	NODE	DESIRED STATE
vxw3473eaiba	full_test_mariadb.1	mariadb:10.5	Swarm	Running

```
Running 2 minutes ago
root@Swarm:/home/monitor#
```

le service est en mode « running »

4. Vérifier que MariaDB écrit bien ses données dans le volume NFS partagé :

- Commande d'accès au conteneur MariaDB :

```
docker exec -it $(docker ps --filter name=full_test_mariadb -q) bash
docker ps --filter name=full_test_mariadb -q → récupère l'ID du conteneur nommé full_test_mariadb.
docker exec -it [ID] bash → ouvre un shell bash interactif dans le conteneur.
```

- Puis dans le conteneur :

```
ls -l /var/lib/mysql
```

Dans le container on voit les fichiers stockés :

```
root@Swarm:/home/monitor# docker exec -it $(docker ps --filter name=full_test
_mariadb -q) bash
root@0103acec93d6:/# ls -l /var/lib/mysql
total 140292
-rw-rw---- 1 mysql mysql 17801216 Jun  8 21:32 aria_log.00000001
-rw-rw---- 1 mysql mysql    52 Jun  8 21:32 aria_log_control
-rw-r--r-- 1 mysql mysql    0 May 31 21:56 hello.txt
-rw-rw---- 1 mysql mysql   898 Jun  8 21:32 ib_buffer_pool
-rw-rw---- 1 mysql mysql 100663296 Jun  8 21:32 ib_logfile0
-rw-rw---- 1 mysql mysql 12582912 Jun  8 21:32 ibdata1
-rw-rw---- 1 mysql mysql 12582912 Jun  8 21:32 ibtmp1
-rw-rw---- 1 mysql mysql    0 Jun  8 21:31 multi-master.info
drwx----- 2 mysql mysql   4096 Jun  8 21:32 mysql
-rw-r--r-- 1 mysql mysql    15 Jun  8 21:31 mysql_upgrade_info
drwx----- 2 mysql mysql   4096 Jun  8 21:31 performance_schema
-rw-r--r-- 1 mysql mysql    14 Jun  8 20:45 test.txt
drwx----- 2 mysql mysql   4096 Jun  8 21:32 testdb
root@0103acec93d6:/#
```

5. Configuration du serveur NFS :

- Sur la VM NFS, on exécute :

```
mkdir -p /srv/nfs/mariadb
```

```
chown -R nobody:nogroup /srv/nfs/mariadb
```

```
chmod -R 777 /srv/nfs/mariadb
```

```
root@nfs:/home/safia# mkdir -p /srv/nfs/mariadb
root@nfs:/home/safia# chown -R nobody:nogroup /srv/nfs/mariadb
root@nfs:/home/safia# chmod 777 /srv/nfs/mariadb
```

- Puis configuration du fichier /etc/exports :

On ajoute à la fin du fichier cette ligne

```
/srv/nfs/docker_volumes/mariadb 192.168.1.0/24(rw,sync,no_subtree_check,no_root_squash)
```

```
safia@nfs: ~  
GNU nano 7.2 /etc/exports *  
# /etc/exports: the access control list for filesystems which may be exported  
# to NFS clients. See exports(5).  
#  
# Example for NFSv2 and NFSv3:  
# /srv/homes hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_>  
#  
# Example for NFSv4:  
# /srv/nfs4 gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)  
# /srv/nfs4/homes gss/krb5i(rw,sync,no_subtree_check)  
#  
/srv/nfs/docker_volumes/mariadb 192.168.1.0/24(rw,sync,no_subtree_check,no_r>  
/srv/nfs/mariadb *(rw,sync,no_subtree_check,no_root_squash)  
█
```

- Redémarrage du service NFS :

exportfs -rav

systemctl restart nfs-server

```
root@nfs:/home/safia# nano /etc/exports  
root@nfs:/home/safia# exportfs -rav  
systemctl restart nfs-server  
exporting 192.168.1.0/24:/srv/nfs/docker_volumes/mariadb  
exporting */srv/nfs/mariadb  
root@nfs:/home/safia# exportfs -rav  
systemctl restart nfs-server  
exporting 192.168.1.0/24:/srv/nfs/docker_volumes/mariadb  
exporting */srv/nfs/mariadb
```

Vérifier la réplication sur NFS

```
root@nfs:/srv/nfs/docker_volumes/mariadb# ls -l /srv/nfs/docker_volumes/maria  
db  
total 140292  
-rw-rw---- 1 999 systemd-journal 17801216 8 juin 23:32 aria_log.00000001  
-rw-rw---- 1 999 systemd-journal 52 8 juin 23:32 aria_log_control  
-rw-r--r-- 1 999 systemd-journal 0 31 mai 23:56 hello.txt  
-rw-rw---- 1 999 systemd-journal 898 8 juin 23:32 ib_buffer_pool  
-rw-rw---- 1 999 systemd-journal 12582912 8 juin 23:32 ibdata1  
-rw-rw---- 1 999 systemd-journal 100663296 8 juin 23:32 ib_logfile0  
-rw-rw---- 1 999 systemd-journal 12582912 8 juin 23:32 ibtmp1  
-rw-rw---- 1 999 systemd-journal 0 8 juin 23:31 multi-master.info  
drwx----- 2 999 systemd-journal 4096 8 juin 23:32 mysql  
-rw-r--r-- 1 999 systemd-journal 15 8 juin 23:31 mysql_upgrade_info  
drwx----- 2 999 systemd-journal 4096 8 juin 23:31 performance_schema  
drwx----- 2 999 systemd-journal 4096 8 juin 23:32 testdb  
-rw-r--r-- 1 999 systemd-journal 14 8 juin 22:45 test.txt  
root@nfs:/srv/nfs/docker_volumes/mariadb# █
```

On y retrouve les mêmes fichiers que dans Mariadb !

6. Création d'un script de sauvegarde MariaDB :Sauvegarde automatique :

Objectif :

Créer un script backup_mariadb.sh sur la VM NFS qui :

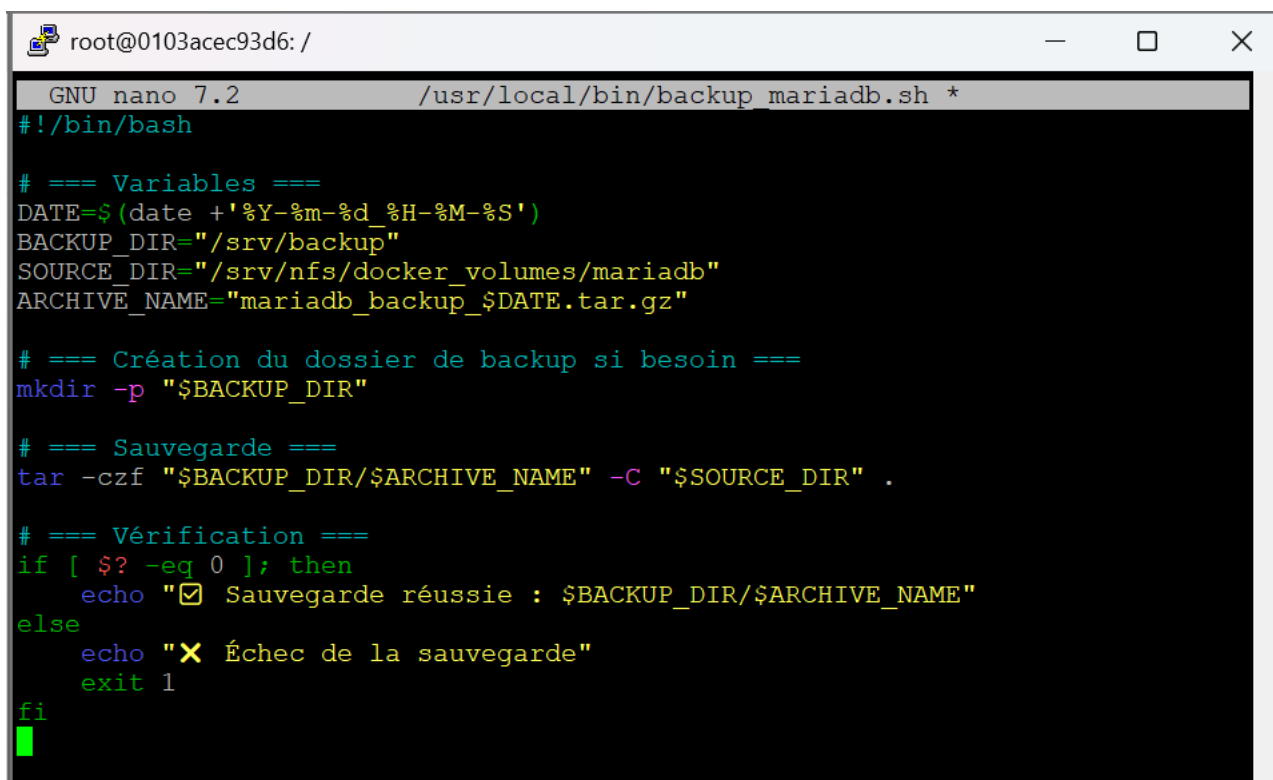
- Archive les données de /srv/nfs/docker_volumes/mariadb
- Les compresse dans /srv/backup
- Nomme les archives avec la date pour les retrouver facilement

1. Créer le dossier de sauvegarde :

```
docker exec -it $(docker ps --filter name=full_test_mariadb -q) bash
```

```
mkdir -p /srv/backup
```

2. Créer le script :



```
root@0103acec93d6: /  
GNU nano 7.2 /usr/local/bin/backup_mariadb.sh *  
#!/bin/bash  
  
# === Variables ===  
DATE=$(date +%Y-%m-%d_%H-%M-%S)  
BACKUP_DIR="/srv/backup"  
SOURCE_DIR="/srv/nfs/docker_volumes/mariadb"  
ARCHIVE_NAME="mariadb_backup_${DATE}.tar.gz"  
  
# === Création du dossier de backup si besoin ===  
mkdir -p "$BACKUP_DIR"  
  
# === Sauvegarde ===  
tar -czf "$BACKUP_DIR/$ARCHIVE_NAME" -C "$SOURCE_DIR" .  
  
# === Vérification ===  
if [ $? -eq 0 ]; then  
    echo "☑ Sauvegarde réussie : $BACKUP_DIR/$ARCHIVE_NAME"  
else  
    echo "✗ Échec de la sauvegarde"  
    exit 1  
fi
```

3. Rendre exécutable :

```
chmod +x /usr/local/bin/backup_mariadb.sh
```

4. Tester manuellement :

```
/usr/local/bin/backup_mariadb.sh
```

```
root@Swarm:/home/monitor# /usr/local/bin/backup_mariadb.sh
☑ Sauvegarde réussie : /srv/backup/mariadb_backup_2025-06-09_00-00-25.tar.gz
root@Swarm:/home/monitor#
```

5. Planification automatique :

Pour une sauvegarde **chaque nuit à 2h**, ajouter une tâche cron :

crontab -e

```
root@Swarm:/home/monitor# crontab -e
no crontab for root - using an empty one

Select an editor. To change later, run 'select-editor'.
 1. /bin/nano      <---- easiest
 2. /usr/bin/vim.tiny

Choose 1-2 [1]: 1
crontab: installing new crontab
```

```
GNU nano 7.2 /tmp/crontab.ZtUzUT/crontab *
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow  command
0 2 * * * /usr/local/bin/backup_mariadb.sh
```


Fichiers de sauvegarde pour les autres services

A. Crée les répertoires de sauvegarde sur la VM NFS

Les sauvegardes doivent se faire sur la VM NFS

→ C'est cette VM qui centralise les volumes Docker persistants, donc c'est bien là que doivent être stockées les archives .tar.gz issues de la sauvegarde des conteneurs (MariaDB, Nginx, PHP, VSCode).

1. Créer l'utilisateur monitor sur la VM NFS :

adduser monitor

usermod -aG sudo monitor

```
root@nfs:~# adduser monitor
usermod -aG sudo monitor
Ajout de l'utilisateur « monitor » ...
Ajout du nouveau groupe « monitor » (1001) ...
Ajout du nouvel utilisateur « monitor » (1001) avec le groupe « monitor » (1001) ...
Création du répertoire personnel « /home/monitor » ...
Copie des fichiers depuis « /etc/skel » ...
Nouveau mot de passe :
Retapez le nouveau mot de passe :
passwd : mot de passe mis à jour avec succès
Modifier les informations associées à un utilisateur pour monitor
Entrer la nouvelle valeur, ou appuyer sur ENTER pour la valeur par défaut
  NOM []: monitor
  Numéro de chambre []: 000
  Téléphone professionnel []: 09876543
  Téléphone personnel []: 34567890
  Autre []:
Cette information est-elle correcte ? [O/n]O
Ajout du nouvel utilisateur « monitor » aux groupes supplémentaires « users »
...
Ajout de l'utilisateur « monitor » au groupe « users » ...
root@nfs:~# mkdir -p /home/monitor/.ssh
nano /home/monitor/.ssh/authorized_keys
root@nfs:~# nano /home/monitor/.ssh/authorized_keys
root@nfs:~# chown -R monitor:monitor /home/monitor/.ssh
chmod 700 /home/monitor/.ssh
chmod 600 /home/monitor/.ssh/authorized_keys
root@nfs:~# ssh -i ~/.ssh/monitor-docker monitor@192.168.1.182
```

2. Autoriser la connexion SSH avec clé publique :

mkdir -p /home/monitor/.ssh

nano /home/monitor/.ssh/authorized_keys

- Dans le fichier nano /home/monitor/.ssh/authorized_keys, coller dedans le contenu de la clé publique.

- Puis donner les autorisations à monitor :

chown -R monitor:monitor /home/monitor/.ssh

chmod 700 /home/monitor/.ssh

chmod 600 /home/monitor/.ssh/authorized_keys

3. Créer les répertoires de sauvegarde sur la VM NFS

- pour chacun des services :

```
mkdir -p /srv/nfs/docker_volumes/backup_mariadb
```

```
mkdir -p /srv/nfs/docker_volumes/backup_php
```

```
mkdir -p /srv/nfs/docker_volumes/backup_vscode
```

```
monitor@nfs:~$ mkdir -p /srv/nfs/docker_volumes/backup_nginx
mkdir -p /srv/nfs/docker_volumes/backup_php
mkdir -p /srv/nfs/docker_volumes/backup_vscode
monitor@nfs:~$
```

- Puis applique les droits :

```
sudo chown -R monitor:monitor /srv/nfs/docker_volumes/backup_*
```

```
sudo chmod -R 755 /srv/nfs/docker_volumes/backup_*
```

4. Sur la VM Swarm : créer les scripts de sauvegarde automatiques

- Pour Nginx :

```
root@0103ac93d6: /
GNU nano 7.2 /usr/local/bin/backup_nginx.sh *
#!/bin/bash

# Générer la date au format AAAA-MM-JJ_HH-MM
DATE=$(date +%F-%H-%M)

# Définir le chemin de sauvegarde local
DEST="/srv/backup/nginx_backup_${DATE}.tar.gz"

# Sauvegarder les fichiers du conteneur nginx dans une archive tar.gz
docker exec -t $(docker ps --filter name=full_test_nginx_proxy -q) tar czf ->

# Transférer la sauvegarde vers la VM NFS via scp avec la clé privée
scp -i ~/.ssh/monitor-docker "$DEST" monitor@192.168.1.182:/srv/nfs/docker_v
```

- Rendre le script exécutable :

```
chmod +x /usr/local/bin/backup_nginx.sh
```

- Tester manuellement une fois pour confirmer :

```
/usr/local/bin/backup_nginx.sh
```

```
root@Swarm:/home/monitor# nano /usr/local/bin/backup_nginx.sh
root@Swarm:/home/monitor# chmod +x /usr/local/bin/backup_nginx.sh
root@Swarm:/home/monitor# /usr/local/bin/backup_nginx.sh
Warning: Identity file /root/.ssh/monitor-docker not accessible: No such file
or directory.
monitor@192.168.1.182's password:
nginx_backup_2025-06-09-14-42.tar.gz      100% 118    42.9KB/s   00:00
root@Swarm:/home/monitor#
```

Le fichier nginx_backup_2025-06-09-14-42.tar.gz a été transféré avec succès sur la VM NFS. On peut améliorer sans taper le mot de passe (clé SSH dans /home/monitor/.ssh/monitor-docker).

- Pour PHP :

```
root@0103acec93d6: /
GNU nano 7.2 /usr/local/bin/backup_php.sh *
#!/bin/bash
DATE=$(date +%F-%H-%M)
DEST="/srv/backup/php_backup_${DATE}.tar.gz"

docker exec -t $(docker ps --filter name=full_test_php_app -q) \
    tar czf - /var/www/html > "$DEST"

scp -i /home/monitor/.ssh/monitor-docker "$DEST" \
    monitor@192.168.1.182:/srv/nfs/docker_volumes/backup_php/
```

```
root@Swarm:/home/monitor# nano /usr/local/bin/backup_php.sh
root@Swarm:/home/monitor# chmod +x /usr/local/bin/backup_php.sh
root@Swarm:/home/monitor# /usr/local/bin/backup_php.sh
Warning: Identity file /home/monitor/.ssh/monitor-docker not accessible: No such file or directory.
monitor@192.168.1.182's password:
php_backup_2025-06-09-18-54.tar.gz      100% 118    81.8KB/s   00:00
root@Swarm:/home/monitor#
```

Script de sauvegarde PHP

Ce script exécute une archive du dossier /var/www/html du conteneur PHP (full_test_php_app), la compresse au format .tar.gz, et la transfère via scp vers le répertoire NFS distant correspondant. Il garantit que le contenu PHP peut être restauré à tout moment.

- Pour VSCode :

```
root@0103acec93d6: /
GNU nano 7.2 /usr/local/bin/backup_vscode.sh *
#!/bin/bash
DATE=$(date +%F-%H-%M)
DEST="/srv/backup/vscode_backup_${DATE}.tar.gz"

docker exec -t $(docker ps --filter name=full_test_vscode -q) \
    tar czf - /home/coder > "$DEST"

scp -i /home/monitor/.ssh/monitor-docker "$DEST" \
    monitor@192.168.1.182:/srv/nfs/docker_volumes/backup_vscode/
```

```
root@Swarm:/home/monitor# nano /usr/local/bin/backup_vscode.sh
root@Swarm:/home/monitor# chmod +x /usr/local/bin/backup_vscode.sh
root@Swarm:/home/monitor# /usr/local/bin/backup_vscode.sh
Warning: Identity file /home/monitor/.ssh/monitor-docker not accessible: No such file or directory.
monitor@192.168.1.182's password:
vscode_backup_2025-06-09-18-58.tar.gz  100% 118    60.3KB/s   00:00
root@Swarm:/home/monitor#
```

Chaque script crée une archive .tar.gz dans un répertoire de sauvegarde dédié sur NFS (/srv/nfs/docker_volumes/backup_<service>).

Les fichiers de sauvegarde sont stockés

Sur la VM NFS, dans :

/srv/nfs/docker_volumes/backup_/_/

/srv/nfs/docker_volumes/backup_php/

/srv/nfs/docker_volumes/backup_vscode/

Cela garantit que l'on peut restaurer les services si les conteneurs tombent.

Ces scripts démontrent une **solution automatisée de sauvegarde de services critiques** hébergés dans des conteneurs Docker. Le transfert vers un serveur NFS sécurisé permet de garantir la **résilience**, la **continuité d'activité** et la **restauration rapide** en cas de panne.