Mise en place galera cluster

Dans ce tutoriel, nous allons apprendre à mettre en place MariaDB Galera Cluster afin de créer un cluster de trois serveurs de bases de données MySQL / MariaDB et assurer la haute disponibilité d'une base de données.

Ce qui est appréciable avec cette solution, au-delà du fait qu'elle soit open source et gratuite, c'est qu'elle offre plusieurs avantages, notamment :

- **Cluster avec une topologie multi maîtres**, donc si un nœud tombe, les autres nœuds continuent d'assurer le service de façon transparente sans avoir besoin d'effectuer des manipulations complexes pour retrouver l'état initial (contrairement à *MySQL Replication*)
- **Cluster actif-actif** avec l'ensemble des nœuds
- **Réplication synchrone des informations**, c'est-à-dire en temps réel
- **Lecture et écriture** sur l'ensemble des nœuds
- **Souple** : fonctionne aussi bien sur le LAN que sur le WAN
- **Support des environnements géo-distribués** : plusieurs centres de données, multi-Cloud, etc.
- **Ajout d'un nouveau nœud** au cluster en quelques minutes (avec les bons paquets et un seul fichier de configuration)

Pour notre archi comme nous sommes sur docker et un reseau bridge tout passera par le FQDN interne

Donc DB01,DB02,DB03

Tenez compte également des informations suivantes :

- **Utilisez des nœuds avec les mêmes ressources**, car le cluster sera aussi lent que le nœud du cluster le plus lent
- Galera Cluster fonctionne **uniquement sur Linux / Unix**
- Avec un cluster géo-distribués, vous devez **contrôler la latence entre vos différents nœuds**, car l'idéal c'est qu'elle ne dépasse pas 300 ms
- Vos bases de données (et vos serveurs) doivent **utiliser le moteur de stockage InnoDB ou XtraDB**, donc MyISAM n'est pas pris en charge

- Toutes les tables de votre base de données répliquée doivent **avoir une clé primaire**
- Le pare-feu de vos serveurs doit **autoriser les ports suivants** : 3306, 4444, 4567 et 4568

Normalement par defaut les versions récentes de mysql les bases sont en innodb donc pas besoin de faire de modifications

Installation image docker

Comme nous sommes sur dokcer nous allons directement chercher une image avec galera installer dessus car il faut retenir uner egle d'or il ne faut pas partir d'une image vierge et installer nos outils dessus il faut chercher une image avec le plus d'outils que nous cherchons et si il n'ya pas on créer notre image ensuite on la pull

Je vais utiliser cette image : `codership/mysql-galera:8.0.42`

Mon compose .yaml

```yaml
  db01:
    image: codership/mysql-galera:8.0.42
    container_name: db01
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      WSREP_NODE_ADDRESS: db01
      WSREP_JOIN: db02,db03
    volumes:
      - db_data_01:/var/lib/mysql

  db02:
    image: codership/mysql-galera:8.0.42
    container_name: db02
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      WSREP_NODE_ADDRESS: db02
      WSREP_JOIN: db01,db03

    volumes:
      - db_data_02:/var/lib/mysql

  db03:
    image: codership/mysql-galera:8.0.42
    container_name: db03
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      WSREP_NODE_ADDRESS: db03
      WSREP_JOIN: db02,db01

    volumes:
      - db_data_03:/var/lib/mysql

volumes:
  db_data_01:
  db_data_02:
  db_data_03:
```

Configuration Docker Compose MySQL Galera Cluster (3 nœuds)

Cluster MySQL 8.0.42 avec Galera pour réplication multi-master synchrone. Utilise variables `.env` pour secrets (MYSQL_ROOT_PASSWORD, MYSQL_DATABASE, MYSQL_USER, MYSQL_PASSWORD).github+1

Services communs

- **Image** : `codership/mysql-galera:8.0.42` (MySQL + Galera patché).
- **Restart** : `always` (redémarrage auto).
- **Volumes** : Persistance données (`db_data_0X:/var/lib/mysql`).
- **Environnement partagé** :

```
MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
MYSQL_DATABASE: ${MYSQL_DATABASE}
MYSQL_USER: ${MYSQL_USER}
MYSQL_PASSWORD: ${MYSQL_PASSWORD}
```

Nœud db01

```
db01:
 image: codership/mysql-galera:8.0.42
 container_name: db01
 restart: always
 environment:
```

```
  WSREP_NODE_ADDRESS: db01
  WSREP_JOIN: db02,db03  # Seed nodes
 volumes:
  - db_data_01:/var/lib/mysql
```

Nœud db02

```
db02:
 image: codership/mysql-galera:8.0.42
 container_name: db02
 restart: always
 environment:
  WSREP_NODE_ADDRESS: db02
  WSREP_JOIN: db01,db03
 volumes:
  - db_data_02:/var/lib/mysql
```

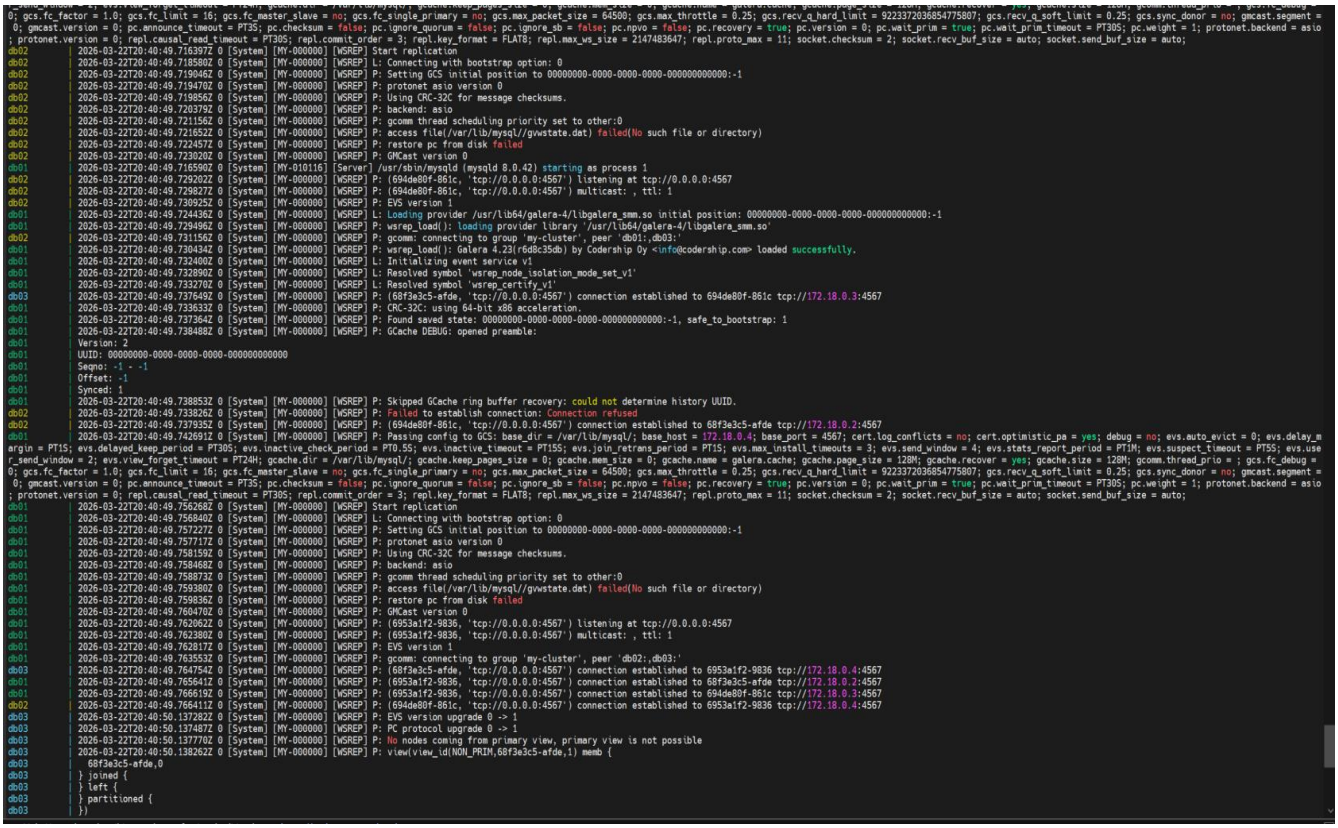
Nœud db03

```
db03:
 image: codership/mysql-galera:8.0.42
 container_name: db03
 restart: always
 environment:
  WSREP_NODE_ADDRESS: db03
  WSREP_JOIN: db01,db02
 volumes:
  - db_data_03:/var/lib/mysql
```

Volumes

```
volumes:
 db_data_01:
 db_data_02:
 db_data_03:
```

Avec cette conf nous avons une erreur nous allons corriger cela



Alors enfaite il faut que le node 1 d'abord créer le nod et faire en sorte que le node 2 et 3 rejoignent le cluster gerer par le node 1 car il faut un maitre dans le cluster et nous l'avons définis ici

Je dois modifier mon fichier compose pour changer une variable d'environnement qui s'appellera BOOSTRAP et si le nœud est master elle sera égale à 1 sinon 0 cette variable sera utiliser dans un scrip bash qui sera copier et lancer depuis le fichier dockerfile

Ensuite pour DB01 nous enleverons la variable d'environnement qui permet de rejoindre le cluster car c'est db01 qui créer le cluster

```yaml
db01:
  image: codership/mysql-galera:8.0.42
  container_name: db01
  restart: always
  environment:
    BOOSTSTRAP: "1"
    MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
    MYSQL_DATABASE: ${MYSQL_DATABASE}
    MYSQL_USER: ${MYSQL_USER}
    MYSQL_PASSWORD: ${MYSQL_PASSWORD}
    WSREP_NODE_ADDRESS: db01
  volumes:
    - db_data_01:/var/lib/mysql

db02:
  image: codership/mysql-galera:8.0.42
  container_name: db02
  restart: always
  environment:
    BOOSTSTRAP: "0"
    MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
    MYSQL_DATABASE: ${MYSQL_DATABASE}
    MYSQL_USER: ${MYSQL_USER}
    MYSQL_PASSWORD: ${MYSQL_PASSWORD}
    WSREP_NODE_ADDRESS: db02
    WSREP_JOIN: db01,db03
  depends_on:
    - db01
  volumes:
    - db_data_02:/var/lib/mysql
```

Je mets un depends_on pour bien laisser db_01 demarrer et initier le cluster comaprait à tout à l'heure on voit bien que db01 ne rejoind plus de cluster la directive est supprimé et BOOSTRAP : '1' est rajouté

Maintenant si on va dans mon Dockerfile

```
 GNU nano 8.4
FROM php:8.2-apache
RUN docker-php-ext-install pdo pdo_mysql
WORKDIR /var/www/html
COPY ./web /var/www/html

FROM codership/mysql-galera:8.0.42

# On ajoute notre script d'entrée
COPY docker-entrypoint.sh /usr/local/bin/docker-entrypoint.sh
RUN chmod +x /usr/local/bin/docker-entrypoint.sh

ENTRYPOINT ["/usr/local/bin/docker-entrypoint.sh"]
CMD ["mysqld"]
```

Mon script

Test validité cluster

Pour voir si le cluster a bien prit il faut d'abord sur le nœud primaire se conencter et voir le status du nœud courant dans le cluster et la taille du cluster si on voit Primary et 3 c'est que le cluster fonctionne

Il faut se connecter et executer

```
SHOW STATUS LIKE 'wsrep_cluster_status';

SHOW STATUS LIKE 'wsrep_cluster_size'
```

```
mysql> SHOW STATUS LIKE 'wsrep_cluster_status';
+----------------------+---------+
| Variable_name        | Value   |
+----------------------+---------+
| wsrep_cluster_status | Primary |
+----------------------+---------+
1 row in set (0.01 sec)

mysql> SHOW STATUS LIKE 'wsrep_cluster_size';
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| wsrep_cluster_size | 3     |
+--------------------+-------+
1 row in set (0.00 sec)

mysql>
```

Cette commande permet de voir l'etat global du cluster

```
show global status like 'wsrep%';
```

```
1 row in set (0.00 sec)

mysql> show global status like 'wsrep%';
+------------------------------+--------------------------------------+
| Variable_name                | Value                                |
+------------------------------+--------------------------------------+
| wsrep_local_state_uuid       | 78c2ffc8-263a-11f1-9038-52af5028096a |
| wsrep_protocol_version       | 11                                   |
| wsrep_protocol_application   | 7                                    |
| wsrep_protocol_replicator    | 11                                   |
| wsrep_protocol_GCS           | 5                                    |
| wsrep_last_committed         | 14                                   |
| wsrep_replicated             | 12                                   |
| wsrep_replicated_bytes       | 4768                                 |
| wsrep_repl_keys              | 24                                   |
| wsrep_repl_keys_bytes        | 480                                  |
| wsrep_repl_data_bytes        | 3492                                 |
| wsrep_repl_other_bytes       | 0                                    |
| wsrep_received               | 9                                    |
| wsrep_received_bytes         | 746                                  |
| wsrep_local_commits          | 0                                    |
| wsrep_local_cert_failures    | 0                                    |
| wsrep_local_replays          | 0                                    |
| wsrep_local_send_queue       | 0                                    |
| wsrep_local_send_queue_max   | 1                                    |
| wsrep_local_send_queue_min   | 0                                    |
| wsrep_local_send_queue_avg   | 0                                    |
| wsrep_local_recv_queue       | 0                                    |
| wsrep_local_recv_queue_max   | 2                                    |
| wsrep_local_recv_queue_min   | 0                                    |
| wsrep_local_recv_queue_avg   | 0.222222                             |
| wsrep_local_cached_downto    | 1                                    |
| wsrep_flow_control_paused_ns | 0                                    |
| wsrep_flow_control_paused    | 0                                    |
| wsrep_flow_control_sent      | 0                                    |
| wsrep_flow_control_recv      | 0                                    |
| wsrep_flow_control_active    | false                                |
| wsrep_flow_control_requested | false                                |
| wsrep_cert_deps_distance     | 1                                    |
| wsrep_apply_oooe             | 0                                    |
| wsrep_apply_oool             | 0                                    |
| wsrep_apply_window           | 1                                    |
| wsrep_apply_waits            | 0                                    |
| wsrep_commit_oooe            | 0                                    |
| wsrep_commit_oool            | 0                                    |
| wsrep_commit_window          | 1                                    |
| wsrep_local_state            | 4                                    |
| wsrep_local_state_comment    | Synced                               |
| wsrep_cert_index_size        | 2                                    |
| wsrep_causal_reads           | 0                                    |
| wsrep_cert_interval          | 0                                    |
| wsrep_open_transactions      | 0                                    |
| wsrep_open_connections       | 0                                    |
| wsrep_incoming_addresses     | AUTO,AUTO,AUTO                       |
| wsrep_cluster_weight         | 3                                    |
| wsrep_desync_count           | 0                                    |
| wsrep_evs_delayed            |                                      |
| wsrep_evs_evict_list         |                                      |
| wsrep_evs_repl_latency       | 0/0/0/0/0                            |
| wsrep_evs_state              | OPERATIONAL                          |
| wsrep_gcomm_uuid             | 78c05df2-263a-11f1-a2d7-f28186bb4236 |
| wsrep_gmcast_segment         | 0                                    |
| wsrep_cluster_capabilities   |                                      |
| wsrep_cluster_conf_id        | 2                                    |
| wsrep_cluster_size           | 3                                    |
| wsrep_cluster_state_uuid     | 78c2ffc8-263a-11f1-9038-52af5028096a |
| wsrep_cluster_status         | Primary                              |
| wsrep_connected              | ON                                   |
| wsrep_local_bf_aborts        | 0                                    |
| wsrep_local_index            | 2                                    |
```

On voit aussi que la config est déjà prete

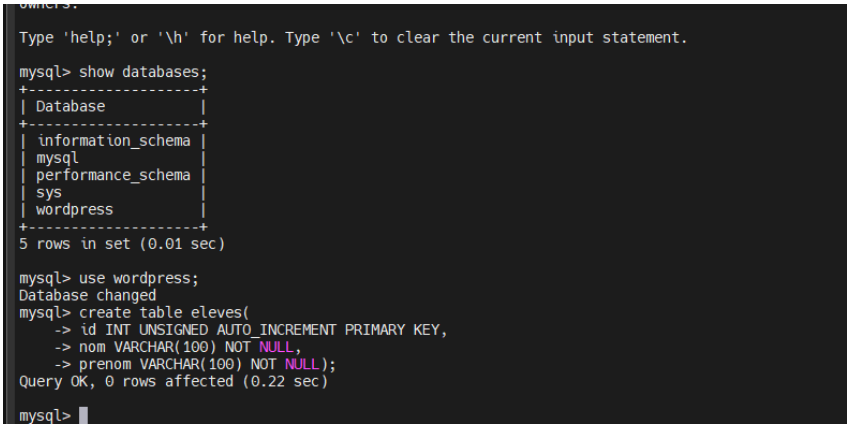
```
bash-5.1$ cat /etc/mysql/conf.d/00-codership.cnf
# This file is for overridable cluster-specific settings
# Any config starting with 01- and above will do it

[server]
default-authentication-plugin=caching_sha2_password
# wsrep configuration
wsrep_on=ON
wsrep_cluster_name="my-cluster"
wsrep_cluster_address="gcomm://"
wsrep_sst_method=clone
wsrep_sync_server_uuid=true
bash-5.1$ cat /etc/mysql/conf.d/99-codership.cnf
# This file is for mandatory cluster-specific settings
# It should override everything else

[server]
#deprecated binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

# wsrep configuration
wsrep_provider=/usr/lib64/galera-4/libgalera_smm.so
bash-5.1$
```

Modification sur le nœud primaire

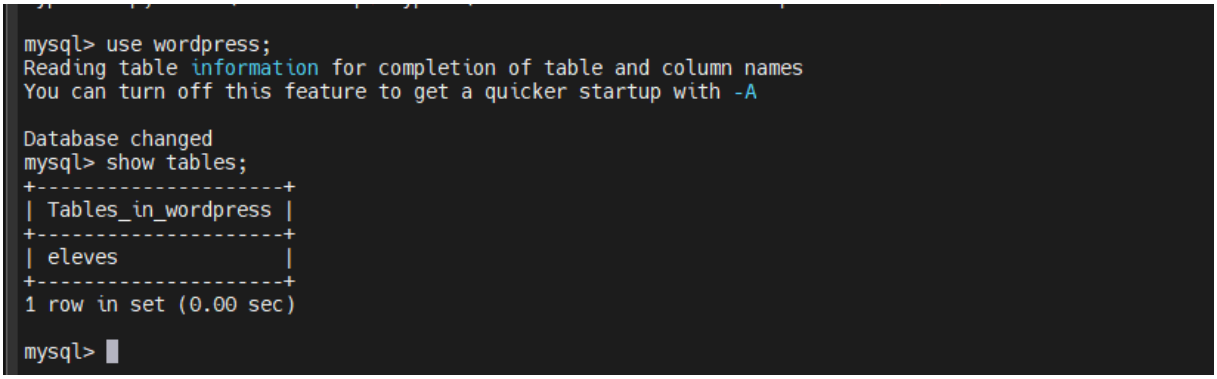
Nous allons tenter une modification sur le nœud primaire pour voir si elle se replique bien

Je créer une table eleve dans la bdd wordpress

BD01 primary node



BD02 slave node



BD03 slave node

Je vais créer une table supplémentaire et voir si elle est repliqué

```
mysql> use wordpress;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> CREATE TABLE etudiants (
    ->     id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    ->     nom VARCHAR(100) NOT NULL,
    ->     prenom VARCHAR(100) NOT NULL,
    ->     email VARCHAR(255) UNIQUE NOT NULL,
    ->     date_inscription DATE
    -> );
Query OK, 0 rows affected (0.23 sec)

mysql>
```

Maintenant retour sur BD01

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

mysql> use wordpress;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+--------------------+
| Tables_in_wordpress |
+--------------------+
| eleves             |
| etudiants          |
+--------------------+
2 rows in set (0.00 sec)

mysql>
```

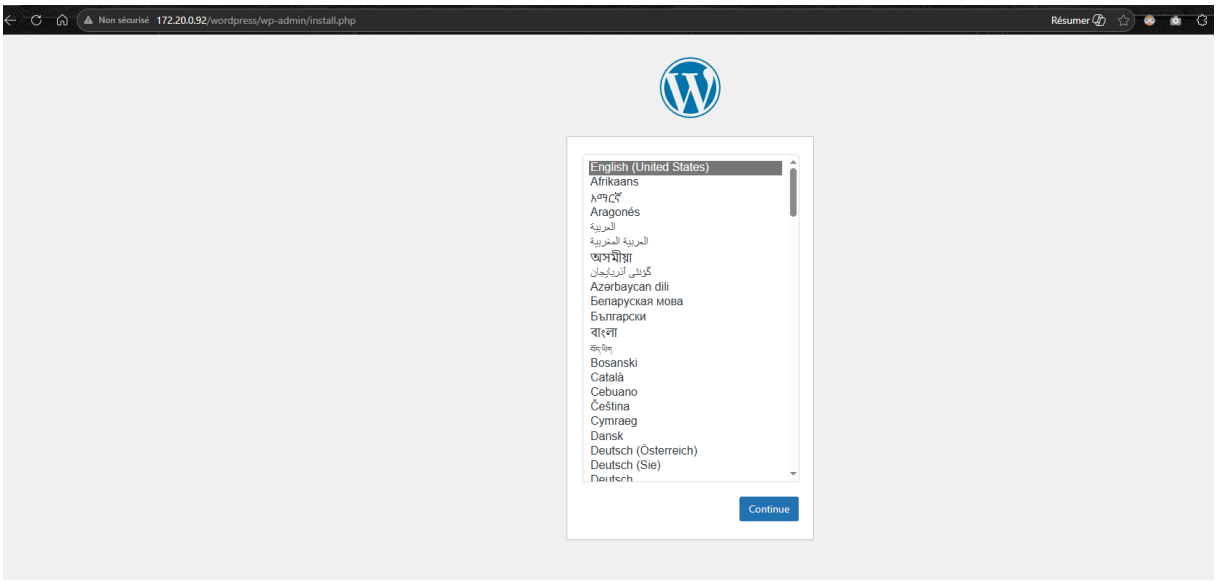
On voit bien la replication

Rajout serveur web wordpress

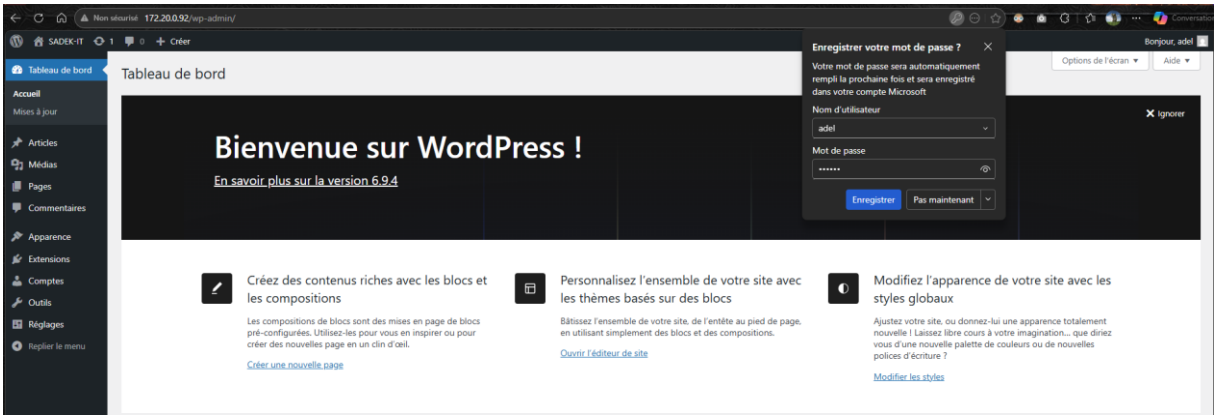
```
wordpress:
    image: wordpress
    restart: always
    environment:
      WORDPRESS_DB_HOST: db01
      WORDPRESS_DB_USER: ${MYSQL_USER}
      WORDPRESS_DB_PASSWORD: ${MYSQL_PASSWORD}
      WORDPRESS_DB_NAME: ${MYSQL_DATABASE}
      WORDPRESS_CONFIG_EXTRA: |
        define('WP_HOME', 'http://172.20.0.92/wordpress');
        define('WP_SITEURL', 'http://172.20.0.92/wordpress');
    volumes:
      - wordpress01:/var/www/html
    depends_on:
      - db
```

On verra plus tard les IP virtuelles etc parceque ça se conennectera que sur db 01 mais la replication sera toujours effective mais non utilisé en cas de panne

Connexion BDD ok installation de wordpress maintenant



Connexion et installation du site ok



Sur DB01 creation bdd wordpress ok

```
mysql> use wordpress;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----------------------+
| Tables_in_wordpress   |
+-----------------------+
| wp_commentmeta        |
| wp_comments           |
| wp_links              |
| wp_options            |
| wp_postmeta           |
| wp_posts              |
| wp_term_relationships |
| wp_term_taxonomy      |
| wp_termmeta           |
| wp_terms              |
| wp_usermeta           |
| wp_users              |
+-----------------------+
12 rows in set (0.00 sec)

mysql>
```

Je vais regarder sur bdd02 si la replication c'est effectuée

```
mysql> use wordpress;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----------------------+
| Tables_in_wordpress   |
+-----------------------+
| wp_commentmeta        |
| wp_comments           |
| wp_links              |
| wp_options            |
| wp_postmeta           |
| wp_posts              |
| wp_term_relationships |
| wp_term_taxonomy      |
| wp_termmeta           |
| wp_terms              |
| wp_usermeta           |
| wp_users              |
+-----------------------+
12 rows in set (0.00 sec)

mysql>
```

Replication ok

Création image personnalisé et integration VRRP

Alors notre image n'a pas VRRP installé dessus nativement et la regle est clair aucune modification majeur dans l'image car elle se reset à chaque fois nous devons donc créer une image perosnnalisé la push sur dockerhub et la re pull puis l'utiliser

Création de l'image personnalisée avec intégration VRRP

Contexte et objectif

L'image Docker officielle `codership/mysql-galera:8.0.42` ne contient pas Keepalived nativement. La règle d'or Docker étant de ne pas modifier une image en cours d'exécution (les modifications sont perdues au redémarrage), nous devons créer une image personnalisée que nous allons publier sur Docker Hub.

Objectif : Créer une image `adelsadek95/galera-vrrp:1.0` qui intègre :

- MySQL Galera (base)
- Keepalived (VRRP pour l'IP virtuelle flottante)
- Une configuration paramétrable via variables d'environnement

---

Étape 1 — Identifier la distribution de base

Avant de créer le Dockerfile, on vérifie la distribution de l'image de base pour savoir quel gestionnaire de paquets utiliser :

```
bash
docker run --rm --entrypoint bash codership/mysql-galera:8.0.42 -c "cat /etc/os-release"
```

Résultat : **Red Hat Enterprise Linux 9.6 (Plow)** → gestionnaire de paquets : `dnf`

On vérifie également que keepalived n'est pas déjà présent :

```
bash
docker run --rm --entrypoint bash codership/mysql-galera:8.0.42 -c "which keepalived"
# which: no keepalived in ...
```

On confirme que `dnf` peut l'installer (en tant que root) :

```
bash
```

```
docker run --rm --user root --entrypoint bash codership/mysql-galera:8.0.42 -c "dnf install -y keepalived && keepalived --version"
# Keepalived v2.2.8 — OK
```

Étape 2 — Architecture de la solution

```
text
Image adelsadek95/galera-vrrp:1.0
│
├── Dockerfile
│   ├── Base : codership/mysql-galera:8.0.42
│   ├── Installation : keepalived + gettext (pour envsubst)
│   ├── COPY keepalived.conf.tpl → /etc/keepalived/keepalived.conf.tpl
│   └── COPY docker-entrypoint.sh → /docker-entrypoint-vrrp.sh
│
├── keepalived.conf.tpl  (template avec variables ${})
│
└── docker-entrypoint.sh
    ├── envsubst remplace les ${} par les valeurs du compose.yaml
    ├── Lance keepalived en arrière-plan
    └── Appelle l'entrypoint original /entrypoint.sh de l'image
```

Principe clé : Une seule image, comportement différent par node selon les variables d'environnement définies dans `compose.yaml`.

Étape 3 — Le template Keepalived

Le fichier `keepalived.conf.tpl` contient des variables `${}` qui seront remplacées au démarrage du container :

```
bash
# keepalived.conf.tpl
vrrp_instance VI_1 {
    state ${VRRP_ROLE}
    interface ${VRRP_INTERFACE}
    virtual_router_id ${VRRP_ROUTER_ID}
    priority ${VRRP_PRIORITY}
    advert_int 1

    virtual_ipaddress {
        ${VRRP_VIRTUAL_IP} dev ${VRRP_INTERFACE}
    }
}
```

Étape 4 — Le script d'entrée

Le fichier `docker-entrypoint.sh` fait le lien entre le template et la configuration finale :

```
bash
#!/bin/bash
```

```
set -e

# Génération de la config Keepalived depuis le template
echo "[INFO] Generation config Keepalived..."
envsubst < /etc/keepalived/keepalived.conf.tpl > /etc/keepalived/keepalived.conf

# Démarrage de Keepalived en arrière-plan
echo "[INFO] Demarrage Keepalived..."
keepalived --dont-fork --log-console &

# Appel de l'entrypoint original de l'image Codership
exec /entrypoint.sh "$@"
```

Point important découvert : L'image `codership/mysql-galera:8.0.42` possède son propre entrypoint `/entrypoint.sh` qui :

- Gère l'initialisation de la base de données
- Utilise `gosu mysql` pour lancer mysqld sous le bon utilisateur
- Redirige les logs MySQL vers stdout via `tail -F`
- Gère le bootstrap du cluster Galera

En appelant `exec /entrypoint.sh "$@"` à la fin de notre script, on réutilise toute cette logique sans la dupliquer.

Étape 5 — Le Dockerfile

```
text
FROM codership/mysql-galera:8.0.42

USER root

RUN dnf install -y keepalived gettext && dnf clean all

RUN mkdir -p /run/keepalived && chmod 777 /run/keepalived && chmod 777 /run

COPY keepalived.conf.tpl /etc/keepalived/keepalived.conf.tpl
RUN chown mysql:mysql /etc/keepalived/keepalived.conf.tpl

COPY --chmod=755 docker-entrypoint.sh /docker-entrypoint-vrrp.sh

ENTRYPOINT ["/docker-entrypoint-vrrp.sh"]
CMD ["mysqld"]
```

Étape 6 — Variables d'environnement dans compose.yaml

```
text
db01:
  image: adelsadek95/galera-vrrp:1.0
  container_name: db01
  restart: always
  cap_add:
```

```
    - NET_ADMIN
   environment:
    BOOTSTRAP: "1"
    MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
    MYSQL_DATABASE: ${MYSQL_DATABASE}
    MYSQL_USER: ${MYSQL_USER}
    MYSQL_PASSWORD: ${MYSQL_PASSWORD}
    WSREP_NODE_ADDRESS: db01
    VRRP_ROLE: MASTER
    VRRP_INTERFACE: eth0
    VRRP_VIRTUAL_IP: 172.18.0.100
    VRRP_ROUTER_ID: 1
    VRRP_PRIORITY: 100
   volumes:
    - db_data_01:/var/lib/mysql

  db02:
   image: adelsadek95/galera-vrrp:1.0
   container_name: db02
   restart: always
   cap_add:
    - NET_ADMIN
   environment:
    BOOTSTRAP: "0"
    MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
    MYSQL_DATABASE: ${MYSQL_DATABASE}
    MYSQL_USER: ${MYSQL_USER}
    MYSQL_PASSWORD: ${MYSQL_PASSWORD}
    WSREP_NODE_ADDRESS: db02
    WSREP_JOIN: db01,db03
    VRRP_ROLE: BACKUP
    VRRP_INTERFACE: eth0
    VRRP_VIRTUAL_IP: 172.18.0.100
    VRRP_ROUTER_ID: 1
    VRRP_PRIORITY: 90
   depends_on:
    - db01
   volumes:
    - db_data_02:/var/lib/mysql

  db03:
   image: adelsadek95/galera-vrrp:1.0
   container_name: db03
   restart: always
   cap_add:
    - NET_ADMIN
   environment:
    BOOTSTRAP: "0"
    MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
    MYSQL_DATABASE: ${MYSQL_DATABASE}
    MYSQL_USER: ${MYSQL_USER}
    MYSQL_PASSWORD: ${MYSQL_PASSWORD}
    WSREP_NODE_ADDRESS: db03
    WSREP_JOIN: db02,db01
    VRRP_ROLE: BACKUP
    VRRP_INTERFACE: eth0
    VRRP_VIRTUAL_IP: 172.18.0.100
    VRRP_ROUTER_ID: 1
    VRRP_PRIORITY: 80
```

```
 depends_on:
  - db01
 volumes:
  - db_data_03:/var/lib/mysql
```

Étape 7 — Build et push sur Docker Hub

```
bash
cd ~/mon-galera

# Connexion Docker Hub
docker login -u adelsadek95

# Build de l'image
docker build -t adelsadek95/galera-vrrp:1.0 .

# Push sur Docker Hub
docker push adelsadek95/galera-vrrp:1.0
```

Problèmes rencontrés et solutions

Problème 1 — `apt-get` non disponible
L'image est basée sur RHEL 9, pas Debian. Solution : utiliser `dnf` avec `--user root`.

Problème 2 — `envsubst` non disponible
`envsubst` fait partie du paquet `gettext`. Solution : ajouter `gettext` dans le `RUN dnf install`.

Problème 3 — MySQL refuse de démarrer en root
MySQL interdit de tourner sous l'utilisateur root. C'est l'entrypoint original `/entrypoint.sh` qui gère ce problème via `gosu mysql`. Solution : appeler l'entrypoint original à la fin de notre script.

Problème 4 — Keepalived ne peut pas écrire son pidfile dans `/run/`
Les permissions de `/run/` étaient restrictives. Solution : `mkdir -p /run/keepalived && chmod 777 /run/keepalived` dans le Dockerfile.

Problème 5 — `keepalived.conf` considéré comme exécutable
`chmod 755` rendait le fichier de conf exécutable, ce que keepalived refuse. Solution : `chmod 644` pour le fichier template.

Problème 6 — L'IP virtuelle n'apparaît pas
Keepalived nécessite la capability `NET_ADMIN` pour manipuler les interfaces réseau. Solution : ajouter `cap_add: [NET_ADMIN]` dans le compose.yaml pour chaque node.

Problème 7 — `safe_to_bootstrap: 0`
Après un arrêt brutal du cluster, Galera refuse de redémarrer si aucun node n'est marqué comme safe. Solution :

```
bash
docker run --rm -v mon-galera_db_data_01:/var/lib/mysql busybox \
  sed -i 's/safe_to_bootstrap: 0/safe_to_bootstrap: 1/' /var/lib/mysql/grastate.dat
```

Vérification finale

```
bash
# Vérifier le cluster Galera
docker exec -it db01 mysql -uroot -proot123 -e \
  "SHOW STATUS LIKE 'wsrep_cluster_size'; SHOW STATUS LIKE 'wsrep_cluster_status';"

# Résultat attendu
# wsrep_cluster_size   | 3
# wsrep_cluster_status | Primary

# Vérifier keepalived
docker exec -it db01 ps aux | grep keepalived
docker exec -it db01 cat /etc/keepalived/keepalived.conf

# Vérifier l'IP virtuelle (après ajout NET_ADMIN)
docker exec -it db01 ip a | grep 172.18.0.100
```

Probleme final avant fonctionnement

Pour chaque serveur de bdd il fallait mettre cela au niveau de l'interface pour permettre l'utilisation de broadcast

```
db01:
  cap_add:
    - NET_ADMIN
    - NET_BROADCAST
    - NET_RAW
```

Par défaut Docker retire toutes les permissions dangereuses. Keepalived en a besoin de 3 pour fonctionner :

`NET_ADMIN` → Permet de modifier les interfaces réseau — ajouter/supprimer l'IP virtuelle `172.18.0.100` sur `eth0`

`NET_BROADCAST` → Permet d'envoyer des paquets broadcast/multicast — nécessaire pour que les nodes VRRP se découvrent entre eux sur le réseau

`NET_RAW` → Permet de créer des sockets raw — keepalived envoie des paquets VRRP de bas niveau (protocole IP 112) qui nécessitent un accès direct à la couche réseau

Sans ces 3 capabilities tu vois dans les logs :

```
text
Netlink: error: Operation not permitted(1), type=RTM_NEWADDR
```

Gestion du Bootstrap Galera — Solution pérenne

Problème rencontré

À chaque redémarrage brutal du cluster (arrêt forcé, crash), Galera écrit dans le fichier `grastate.dat` de chaque node :

```
text
safe_to_bootstrap: 0
```

Cela empêche tout node de démarrer en mode bootstrap car aucun n'est certain d'avoir les données les plus récentes. Le cluster reste bloqué en boucle de restart.

Symptôme dans les logs :

```
text
[ERROR] P: It may not be safe to bootstrap the cluster from this node.
It was not the last one to leave the cluster and may not contain all the updates.
To force cluster bootstrap with this node, edit the grastate.dat file manually
and set safe_to_bootstrap to 1.
```

Mauvaise approche (solution manuelle)

On avait initialement utilisé une variable `BOOTSTRAP=1` dans le compose.yaml pour forcer db01 à bootstrapper :

```
text
db01:
 environment:
  BOOTSTRAP: "1"  # ← problématique
```

Et en cas de crash on devait manuellement corriger le `grastate.dat` :

```
bash
docker run --rm -v mon-galera_db_data_01:/var/lib/mysql busybox \
 sed -i 's/safe_to_bootstrap: 0/safe_to_bootstrap: 1/' /var/lib/mysql/grastate.dat
```

Cette approche est **fragile** — elle nécessite une intervention manuelle à chaque crash.

Solution pérenne

L'entrypoint original de l'image `codership/mysql-galera` gère déjà automatiquement le bootstrap via la variable `WSREP_JOIN` :

- Si `WSREP_JOIN` est **vide** → le node bootstrap le cluster (`gcomm://`)
- Si `WSREP_JOIN` est **renseigné** → le node rejoint les nodes listés

L'entrypoint lit aussi le `grastate.dat` automatiquement pour déterminer qui peut bootstrapper.

Fonctionnement automatique au redémarrage

```
text
Redémarrage du cluster
    ↓
db01 démarre → WSREP_JOIN vide → lit grastate.dat
    ↓
safe_to_bootstrap: 1 → bootstrap automatique
    ↓
db02 démarre → WSREP_JOIN=db01,db03 → rejoint db01
    ↓
db03 démarre → WSREP_JOIN=db01,db02 → rejoint db01
    ↓
Cluster 3 nodes Primary ✅
```

Plus aucune intervention manuelle nécessaire.

Le compose.yaml pour db01

```yaml
db01:
  image: adelsadek95/galera-vrrp:1.0
  container_name: db01
  restart: always
  environment:
    MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
    MYSQL_DATABASE: ${MYSQL_DATABASE}
    MYSQL_USER: ${MYSQL_USER}
    MYSQL_PASSWORD: ${MYSQL_PASSWORD}
    WSREP_NODE_ADDRESS: db01
    VRRP_ROLE: MASTER
    VRRP_INTERFACE: eth0
    VRRP_VIRTUAL_IP: 172.18.0.100
    VRRP_ROUTER_ID: 1
    VRRP_PRIORITY: 100
  cap_add:
    - NET_ADMIN
    - NET_BROADCAST
    - NET_RAW
  volumes:
    - db_data_01:/var/lib/mysql
```

Pour DB02

```yaml
  db02:
    image: adelsadek95/galera-vrrp:1.0
    container_name: db02
    restart: always
    environment:
      BOOTSTRAP: "0"
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      WSREP_NODE_ADDRESS: db02
      WSREP_JOIN: db01,db03
      VRRP_ROLE: BACKUP
      VRRP_INTERFACE: eth0
      VRRP_VIRTUAL_IP: 172.18.0.100
      VRRP_ROUTER_ID: 1
      VRRP_PRIORITY: 90
    cap_add:
      - NET_ADMIN
      - NET_BROADCAST
      - NET_RAW
    depends_on:
      - db01
    volumes:
      - db_data_02:/var/lib/mysql
```

Mon fichier Dockerfile

```
FROM codership/mysql-galera:8.0.42

USER root

RUN dnf install -y keepalived gettext && dnf clean all

RUN mkdir -p /run/keepalived && chmod 777 /run/keepalived && chmod 777 /run

COPY keepalived.conf.tpl /etc/keepalived/keepalived.conf.tpl
RUN chown mysql:mysql /etc/keepalived/keepalived.conf.tpl

COPY --chmod=755 docker-entrypoint.sh /docker-entrypoint-vrrp.sh

ENTRYPOINT ["/docker-entrypoint-vrrp.sh"]
CMD ["mysqld"]
```

Mon script bash entrypoint

```
  GNU nano 8.4                                    docker-entrypoint.sh
#!/bin/bash
set -e

# Keepalived
echo "[INFO] Generation config Keepalived..."
envsubst < /etc/keepalived/keepalived.conf.tpl > /etc/keepalived/keepalived.conf
echo "[INFO] Demarrage Keepalived..."
keepalived --dont-fork --log-console &

# Lancer entrypoint original
exec /entrypoint.sh "$@"
```

Je relance maintenant proprement et on effectue une batterie de test

On voit ici DB01 le master avoir l'ip virtuelle

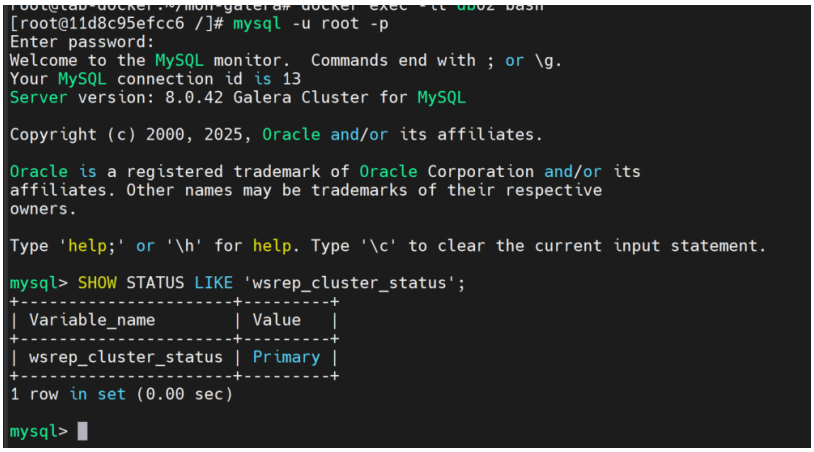
```
root@lab-docker:~/mon-galera# docker exec -it db01 ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host proto kernel_lo
       valid_lft forever preferred_lft forever
2: eth0@if8831: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 66:71:a4:8a:f7:dd brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.18.0.2/16 brd 172.18.255.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet 172.18.0.100/32 scope global eth0
       valid_lft forever preferred_lft forever
root@lab-docker:~/mon-galera#
```

Arrêt DB01 observation reprise ip virtuelle coté DB02

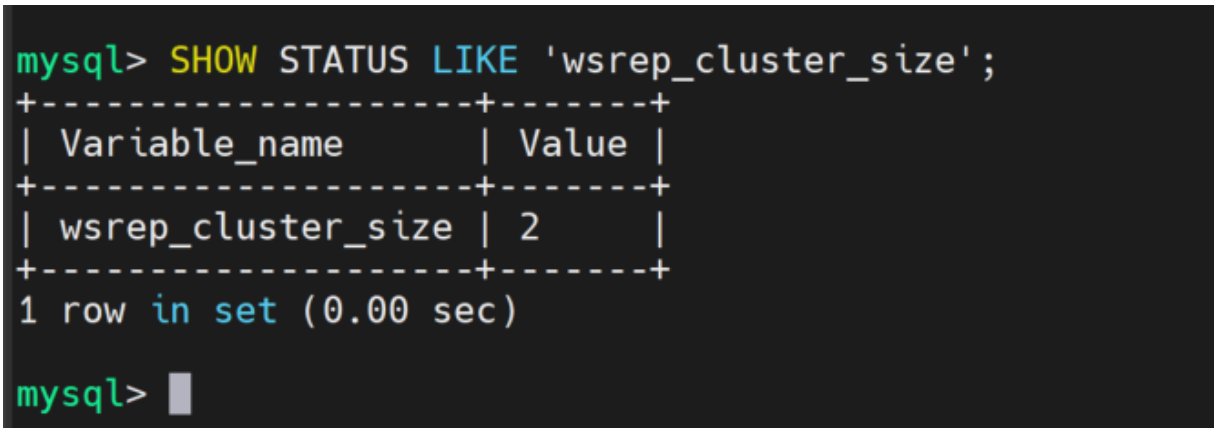
```
2026-03-24T16:42:35.969198Z 0 [System] [MY-000000] [WSREP] P: Flow-control interval: [23, 23]
2026-03-24T16:42:35.969338Z 1 [System] [MY-000000] [WSREP] P: ####### processing CC 31, local, ordered
2026-03-24T16:42:35.969382Z 1 [System] [MY-000000] [WSREP] P: ####### My UUID: 2a785aae-27a0-11f1-8396-0a32fe6d8046
2026-03-24T16:42:35.969400Z 1 [System] [MY-000000] [WSREP] P: Skipping cert index reset
2026-03-24T16:42:35.969421Z 1 [System] [MY-000000] [WSREP] P: REPL Protocols: 11 (6)
2026-03-24T16:42:35.969438Z 1 [System] [MY-000000] [WSREP] P: ####### Adjusting cert position: 30 -> 31
2026-03-24T16:42:35.969479Z 0 [System] [MY-000000] [WSREP] P: Service thread queue flushed.
2026-03-24T16:42:35.996490Z 1 [System] [MY-000000] [WSREP] L: ================================================
View:
  id: 2428be07-2740-11f1-9a2a-8f3f73277886:31
  status: primary
  protocol_version: 7
  capabilities: MULTI-MASTER, CERTIFICATION, PARALLEL_APPLYING, REPLAY, ISOLATION, PAUSE, CAUSAL_READ, INCREMENTAL_WS, UNORDERED, PREORDE
RED, STREAMING, NBO
  final: no
  own_index: 1
  members(2):
        0: 2a715a35-27a0-11f1-9e47-4acf66743a19, 2d9fdcd0ae2b
        1: 2a785aae-27a0-11f1-8396-0a32fe6d8046, 11d8c95efcc6
=================================================
2026-03-24T16:42:35.996543Z 1 [System] [MY-000000] [WSREP] wsrep_notify_cmd is not defined, skipping notification.
2026-03-24T16:42:36.026182Z 1 [System] [MY-000000] [WSREP] P: Lowest cert index boundary for CC from group: 31
2026-03-24T16:42:36.026227Z 1 [System] [MY-000000] [WSREP] P: Min available from gcache for CC from group: 20
2026-03-24T16:42:41.133253Z 0 [System] [MY-000000] [WSREP] P:  cleaning up 2a2caed3-9028 (tcp://172.18.0.2:4567)
Tue Mar 24 16:42:43 2026: (VI_1) Entering MASTER STATE
```

```
root@lab-docker:~/mon-galera# docker exec -it db02 ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host proto kernel_lo
       valid_lft forever preferred_lft forever
2: eth0@if8833: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether be:87:62:bc:2c:b7 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.18.0.4/16 brd 172.18.255.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet 172.18.0.100/32 scope global eth0
       valid_lft forever preferred_lft forever
root@lab-docker:~/mon-galera#
```

Le cluster galera fonctionne parfaitement db02 est passé primary



Le cluster est passé à deux nœuds comme voulus



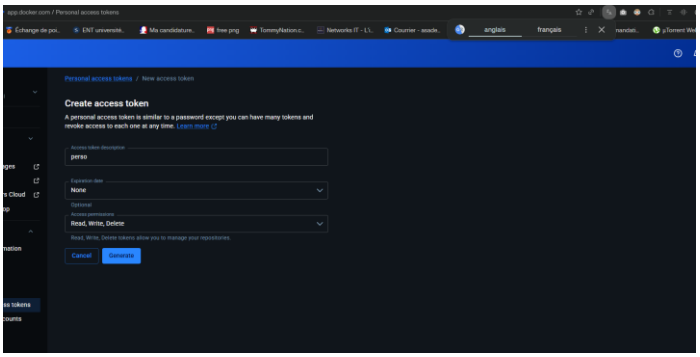
Push l'image sur dockerhub

J'ai créer un compte dockerhub et mon identifiant est « adelsadek95 »

D'abord pour se connecter il faut exécuter cette commande

```
docker login -u adelsadek95
```

Je créer aussi un PAT password access token



Ensuite il y'aura un token qu'il faudra copier quand on nous demande le mot de passe

Me voila connecter

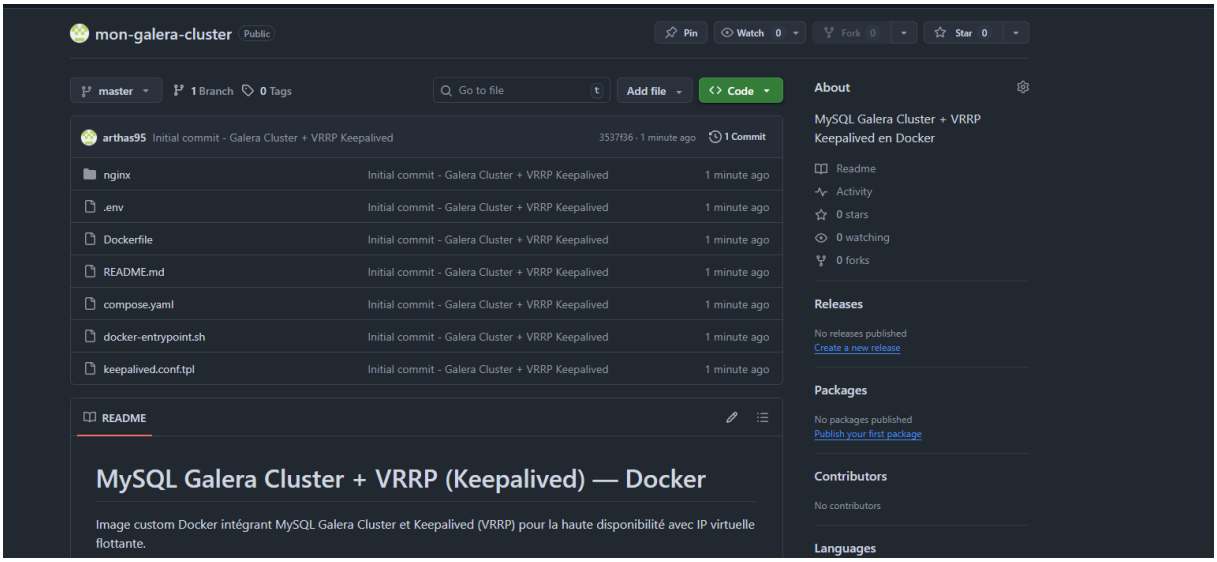
```
[root@11d8c95efcc6 /]# exit
exit
root@lab-docker:~/mon-galera# docker login -u adelsadek95

i Info → A Personal Access Token (PAT) can be used instead.
         To create a PAT, visit https://app.docker.com/settings


Password:

WARNING! Your credentials are stored unencrypted in '/root/.docker/config.json'.
Configure a credential helper to remove this warning. See
https://docs.docker.com/go/credential-store/

Login Succeeded
root@lab-docker:~/mon-galera#
```

Ensuite push l'image

```
docker push adelsadek95/galera-vrrp:1.0
```

```
root@lab-docker:~/mon-galera# docker push adelsadek95/galera-vrrp:1.0
The push refers to repository [docker.io/adelsadek95/galera-vrrp]
045028eb1ea1: Mounted from codership/mysql-galera
dacc74065048: Mounted from codership/mysql-galera
a6ff26f298d8: Mounted from codership/mysql-galera
458f5c31cff6: Pushed
3ae8917b494c: Pushed
b3f7c1d19f07: Mounted from codership/mysql-galera
91689ad6d574: Mounted from codership/mysql-galera
b7aa48ec6ed8: Mounted from codership/mysql-galera
d48a047d1592: Pushed
90cf1e6d6e7e: Mounted from codership/mysql-galera
76c38e8b8231: Pushed
dd3d5431d5bf: Mounted from codership/mysql-galera
7c5fc9148653: Mounted from codership/mysql-galera
93ab2fcd2f1e: Pushed
b3f0454aba89: Mounted from codership/mysql-galera
ba74022ae4f0: Pushed
1.0: digest: sha256:9c7ef8cd6b579998ed4070dddf2e101ac78b18bd2119d49eb6b3bc47d8683865 size: 856
root@lab-docker:~/mon-galera#
```

Il faut aussi push sur github pour ne pas perdre la config et pouvoir réutiliser à n'importe quel moment



https://github.com/arthas95/mon-galera-cluster