# Module3 : Le réseau, les volumes Docker

# I- Le réseau dans Docker

### Introduction aux réseaux Docker

Les réseaux Docker permettent aux conteneurs de communiquer entre eux et avec le monde extérieur. Ils fournissent des mécanismes pour connecter, isoler et sécuriser les conteneurs en fonction des besoins de l'application.

Docker crée automatiquement un réseau bridge par défaut pour chaque conteneur, mais vous pouvez également créer vos propres réseaux pour une meilleure flexibilité.

### Types de réseaux Docker

- **Bridge** : Réseau par défaut, crée une passerelle NAT pour connecter les conteneurs entre eux et à l'extérieur.
- **Host** : Partage directement l'interface réseau de l'hôte, sans isolation réseau.
- **Overlay** : Utilisé pour les services Docker Swarm pour connecter les conteneurs sur plusieurs hôtes.
- **Macvlan** : Attribue une adresse MAC unique à chaque conteneur pour une meilleure intégration réseau.

### Création et gestion des réseaux

- Créer un réseau bridge personnalisé :
$ docker network create my_bridge

```
docker network create  --driver bridge --subnet 192.168.12.0/24  --dns 8.8.8.8  --dns 8.8.4.4 my_bridge
```

ou

```
docker network create \
  --driver bridge \
  --subnet 192.168.12.0/24 \
  --dns 8.8.8.8 \
  --dns 8.8.4.4 \
  my_bridge
```

- Lister les réseaux existants :
$ docker network ls

- Inspecter un réseau :
$ docker network inspect my_bridge

- Supprimer un réseau :
$ docker network rm my_bridge

## Isolation des conteneurs

L'isolation réseau est essentielle pour sécuriser les applications multi-conteneurs.
- Utiliser des réseaux séparés pour isoler les services.
- Utiliser '--internal' pour empêcher les communications externes.

$ docker network create --internal isolated_network

## Routage et communication entre conteneurs

- Connecter un conteneur à plusieurs réseaux :
$ docker network connect my_network my_container

docker network connect lgnet legeni_ubuntu
docker network connect mcsnet legeni_ubuntu

- Déconnecter un conteneur d'un réseau :
$ docker network disconnect my_network my_container

- Utiliser des alias DNS pour simplifier les communications entre conteneurs.

## DNS et résolution des noms

Docker utilise un serveur DNS intégré pour résoudre les noms de conteneurs sur le même réseau.
- Utiliser des alias DNS :
$ docker run -d --network my_network --network-alias=my_app busybox sleep 3600

- Vérifier les entrées DNS :
$ docker exec my_app ping my_app

## Utilisation de Docker Compose pour la gestion des réseaux

Docker Compose permet de définir des réseaux complexes avec des fichiers YAML.

Exemple de fichier docker-compose.yml :

```
version: '3'
services:
  web:
    image: nginx
    networks:
      - front_net
  db:
    image: mysql
```

```
    environment:
      MYSQL_ROOT_PASSWORD: example
    networks:
      - back_net

networks:
  front_net:
  back_net:
```

### Sécurité des réseaux Docker

- Utiliser des réseaux internes pour limiter l'accès externe.
- Utiliser des règles de pare-feu pour sécuriser les communications.
- Utiliser des VLAN pour segmenter les réseaux en production.

### Meilleures pratiques pour la gestion des réseaux

- Éviter les réseaux host pour des raisons de sécurité.
- Utiliser des réseaux bridge pour les déploiements simples.
- Privilégier les réseaux overlay pour les clusters distribués.

### Exemples pratiques et cas d'utilisation

- Création d'un réseau isolé pour une application multi-conteneurs.
- Utilisation des alias DNS pour simplifier les communications.
- Configuration d'un réseau overlay pour un cluster Swarm.

# II- les volumes Docker

### Introduction aux volumes Docker

Les volumes Docker sont utilisés pour stocker les données générées et utilisées par les conteneurs.
Ils permettent de persister les données même lorsque les conteneurs sont supprimés, offrant une solution de stockage flexible et indépendante du cycle de vie des conteneurs.
Les volumes sont gérés par Docker et peuvent être partagés entre plusieurs conteneurs.

### Types de volumes Docker

- Volumes nommés : Gérés entièrement par Docker, créés avec des commandes comme 'docker volume create'.
- Volumes anonymes : Créés automatiquement par Docker lorsqu'un conteneur est démarré avec un montage de volume sans nom.

- Bind mounts : Utilisent des répertoires spécifiques du système de fichiers de l'hôte, créés avec l'option '-v' ou '--mount'.

### Création et gestion des volumes

- Créer un volume nommé :
$ docker volume create mon_volume

- Lister les volumes existants :
$ docker volume ls

- Afficher les détails d'un volume :
$ docker volume inspect mon_volume

- Supprimer un volume :
$ docker volume rm mon_volume

### Montage de volumes dans les conteneurs

Monter un volume nommé :
$ docker run -d -v mon_volume:/app/data my_image

Monter un répertoire de l'hôte comme volume :
$ docker run -d -v /chemin/local:/app/data my_image

Utiliser l'option '--mount' (recommandée) :
$ docker run -d --mount type=volume,source=mon_volume,target=/app/data my_image

### Bonnes pratiques pour l'utilisation des volumes

- Utiliser des volumes nommés pour un meilleur contrôle de la gestion des données.
- Éviter d'utiliser des volumes anonymes pour les données critiques.
- Utiliser les volumes pour partager les données entre les conteneurs de manière sécurisée.
- Préférer '--mount' à '-v' pour une syntaxe plus explicite et flexible.

## TP - Réseaux Docker

### Cas 1 : Création d'un réseau personnalisé

1. Créez un réseau nommé 'custom_network'.
2. Démarrez deux conteneurs sur ce réseau.
3. Vérifiez que les conteneurs peuvent communiquer entre eux.
4. Supprimez le réseau pour nettoyer l'environnement.

### Cas 2 : Isolation des conteneurs

1. Créez deux réseaux séparés 'net1' et 'net2'.
2. Démarrez un conteneur sur chaque réseau.
3. Vérifiez que les conteneurs ne peuvent pas communiquer entre eux.

### Cas 3 : Routage entre réseaux

1. Créez deux réseaux 'frontend' et 'backend'.
2. Démarrez un conteneur sur chaque réseau.
3. Connectez le conteneur 'frontend' au réseau 'backend'.
4. Vérifiez que les conteneurs peuvent maintenant communiquer.

### Cas 4 : Utilisation des alias DNS

1. Créez un réseau nommé 'dns_network'.
2. Démarrez deux conteneurs sur ce réseau avec des alias DNS personnalisés.
3. Vérifiez que les conteneurs peuvent se ping avec leurs alias.

### Cas 5 : Utilisation des réseaux bridge et host

1. Démarrez un conteneur sur le réseau bridge par défaut.
2. Démarrez un autre conteneur sur le réseau host.
3. Comparez les différences de connectivité.

### Cas 6 : Réseau multi-hôtes avec Docker Swarm

1. Créez un cluster Docker Swarm.
2. Créez un réseau overlay.
3. Démarrez des services sur ce réseau et vérifiez la connectivité entre les nœuds.

### Cas 7 : Utilisation de Docker Compose pour les réseaux

1. Créez un fichier docker-compose.yml pour un service web et une base de données.
2. Vérifiez que les services peuvent communiquer sur un réseau personnalisé.

### Cas 8 : Réseau avec sous-réseau personnalisé

1. Créez un réseau avec un sous-réseau spécifique.
2. Vérifiez que les conteneurs utilisent les adresses IP du sous-réseau.

### Cas 9 : Limitation de la bande passante

1. Créez un réseau avec une limitation de bande passante.
2. Vérifiez que les conteneurs respectent cette limite.

### Cas 10 : Nettoyage complet des réseaux Docker

1. Listez tous les réseaux présents sur votre système.
2. Supprimez tous les réseaux inutilisés pour réinitialiser l'environnement Docker.
3. Vérifiez que tous les réseaux ont bien été supprimés.

## TP : Création et utilisation des volumes Docker

Exercice 2.1 : Création d'un volume nommé
1. Créez un volume nommé 'data_volume'.
2. Démarrez un conteneur nginx utilisant ce volume pour stocker les données de configuration.
3. Vérifiez que les données sont bien persistées après la suppression du conteneur.

Exercice 2.2 : Utilisation des volumes avec des bases de données
1. Créez un volume nommé 'db_data'.
2. Démarrez un conteneur MySQL en utilisant ce volume pour stocker les données.
3. Vérifiez que les données restent accessibles après la suppression du conteneur.

Exercice 2.3 : Nettoyage des volumes
1. Listez tous les volumes existants.
2. Supprimez les volumes inutilisés.

# Correction

## I- Corrigés des TP Réseaux Docker

### Corrigé Cas 1 : Création d'un réseau personnalisé

```
$ docker network create custom_network
$ docker run -d --name cont1 --network custom_network busybox sleep 3600
$ docker run -d --name cont2 --network custom_network busybox sleep 3600
$ docker exec cont1 ping -c 2 cont2
$ docker network rm custom_network
```

### Corrigé Cas 2 : Isolation des conteneurs

```
$ docker network create net1
$ docker network create net2
$ docker run -d --name cont1 --network net1 busybox sleep 3600
$ docker run -d --name cont2 --network net2 busybox sleep 3600
# Les conteneurs ne doivent pas pouvoir se ping
```

### Corrigé Cas 3 : Routage entre réseaux

```
$ docker network create frontend
$ docker network create backend
$ docker run -d --name front --network frontend busybox sleep 3600
$ docker run -d --name back --network backend busybox sleep 3600
$ docker network connect backend front
$ docker exec front ping -c 2 back
```

### Corrigé Cas 4 : Utilisation des alias DNS

```
$ docker network create dns_network
$ docker run -d --network dns_network --name dns1 --network-alias=app1 busybox sleep
3600
$ docker run -d --network dns_network --name dns2 --network-alias=app2 busybox sleep
3600
$ docker exec dns1 ping -c 2 app2
```

### Corrigé Cas 5 : Utilisation des réseaux bridge et host

```
$ docker run -d --name bridge_test busybox sleep 3600
$ docker run -d --name host_test --network host busybox sleep 3600
```

### Corrigé Cas 6 : Réseau multi-hôtes avec Docker Swarm

```
$ docker swarm init
$ docker network create --driver overlay multi_host_network
$ docker service create --name web --network multi_host_network nginx
```

### Corrigé Cas 7 : Utilisation de Docker Compose pour les réseaux

```
version: '3'
services:
 web:
  image: nginx
  networks:
   - my_network
 db:
  image: mysql
  environment:
   MYSQL_ROOT_PASSWORD: example
  networks:
   - my_network
networks:
 my_network:
```

### Corrigé Cas 8 : Réseau avec sous-réseau personnalisé

```
$ docker network create --subnet=172.18.0.0/16 custom_subnet
$ docker run -d --network custom_subnet busybox sleep 3600
```

### Corrigé Cas 9 : Limitation de la bande passante

```
$ docker network create --driver bridge --opt com.docker.network.bridge.enable_icc=false
limited_network
$ docker run -d --network limited_network busybox sleep 3600
```

### Corrigé Cas 10 : Nettoyage complet des réseaux Docker

```
$ docker network prune -f
```

## II- Corrigés des TP Volumes Docker

### Corrigé Exercice 2.1 :

```
$ docker volume create data_volume
$ docker run -d --name web_server -v data_volume:/usr/share/nginx/html nginx
$ docker rm -f web_server
$ docker run -d --name web_server -v data_volume:/usr/share/nginx/html nginx
```

## Corrigé Exercice 2.2 :

$ docker volume create db_data
$ docker run -d --name mysql_server -e MYSQL_ROOT_PASSWORD=root -v db_data:/var/lib/mysql mysql

## Corrigé Exercice 2.3 :

$ docker volume ls
$ docker volume rm data_volume db_data