

DOCKER COMPOSE — ORCHESTRER LES CONTENEURS

Cours simplifié, corrigé et vérifié — août 2026

Comprendre pourquoi on l'utilise, comment lire un fichier YAML et comment piloter plusieurs conteneurs sans se perdre.

À retenir en une phrase

Docker Compose permet de décrire toute une application multi-conteneurs dans un fichier YAML, puis de créer, démarrer, arrêter et gérer ses services avec quelques commandes.

1) Pourquoi Docker Compose existe ?

Question — Quel problème résout Docker Compose ?

Avec docker run, on peut lancer les conteneurs un par un. Mais dès qu'une application a plusieurs briques — par exemple une interface Web + une base de données — les commandes deviennent longues et répétitives. Compose centralise toute la configuration dans un seul fichier.

- Plus besoin de retaper plusieurs longues commandes docker run.
- Les variables d'environnement, ports, volumes, réseaux et dépendances sont regroupés au même endroit.
- Une seule commande peut lancer toute l'application.

Idée visuelle

```
SANS COMPOSE
docker run ... mariadb
docker run ... glpi
docker network ...
docker volume ...

AVEC COMPOSE
compose.yaml
  ↓
docker compose up -d
  ↓
GLPI + base de données + réseau + volumes
```

Question — Une application entière est-elle « un service » ?

Pas exactement. Dans Compose, l'application est un projet composé de plusieurs services. Un service représente une brique de l'application et s'exécute à partir d'une image de conteneur.

- Exemple : service web = GLPI.
- Exemple : service db = MariaDB/MySQL.
- Ces services communiquent entre eux via un réseau Docker.

Question — Docker Compose = microservices ?

Compose convient très bien aux applications découpées en plusieurs services, y compris une architecture microservices. Mais Compose n'impose pas les microservices : il peut aussi gérer une application classique composée de quelques conteneurs.

Nuance importante — HA et mise à l'échelle

Compose peut lancer plusieurs instances d'un service sur un même moteur Docker, mais ce n'est pas un orchestrateur de cluster haute disponibilité multi-hôtes. Pour une vraie HA distribuée, on utilise plutôt des solutions d'orchestration de cluster comme Kubernetes ou Docker Swarm.

2) Installation actuelle de Docker Compose sous Debian

Question — Le cours dit `apt install docker-compose`. Est-ce encore la méthode actuelle ?

C'était la commande courante de Docker Compose v1. Aujourd'hui, Docker recommande le plugin Compose et la commande avec un espace : `docker compose`.

- Ancienne syntaxe : `docker-compose up -d`
- Syntaxe actuelle recommandée : `docker compose up -d`

Si Docker a été installé depuis le dépôt officiel Docker

```
apt update
apt install docker-compose-plugin
docker compose version
```

Pourquoi cette correction ?

Compose v1 utilisait `docker-compose`. Le Compose actuel est intégré comme plugin de la CLI Docker et s'utilise avec `docker compose`. Le support du prof reste utile pour comprendre le principe, mais la syntaxe moderne est celle ci-dessus.

3) Le fichier Compose : où et comment ?

Question — Quel fichier dois-je créer ?

La documentation actuelle privilégie `compose.yaml`. Le nom `docker-compose.yml` reste très courant dans les anciens cours et projets. Docker Compose recherche automatiquement un fichier Compose dans le dossier courant.

Créer un projet

```
mkdir monprojet
cd monprojet
nano compose.yaml
```

Question — Pourquoi nano ?

`nano` est simplement un éditeur de texte dans le terminal. Tu l'utilises pour écrire le contenu YAML du fichier `compose.yaml`.

Question — Comment fonctionne l'indentation YAML ?

YAML utilise l'indentation pour représenter la hiérarchie. Utilise des espaces, jamais des tabulations. Deux espaces par niveau est une convention simple et lisible ; l'important est d'être cohérent.

```
services:
  web:
    image: nginx:latest
    ports:
      - "8080:80"
```

4) Le mot version : correction importante

Support ancien

Le tableau du cours mentionne `version` pour « définir la version supportée du fichier ». Cette clé existait dans les anciens formats Compose.

Aujourd'hui

La propriété top-level `version` est obsolète et seulement informative. Compose utilise la Compose Specification actuelle. Tu peux donc écrire un fichier moderne sans `version` : au début.

5) Que contient un fichier compose.yaml ?

Élément	Explication simple
services	Section obligatoire qui contient les services de l'application.
image	Image Docker à utiliser pour le service, ex. mariadb:latest ou nginx:latest.
build	Indique comment construire une image à partir d'un Dockerfile.
build.context	Dossier utilisé comme contexte de build ; peut aussi pointer vers un dépôt Git.
build.dockerfile	Permet de choisir un Dockerfile différent du Dockerfile standard.
build.args	Valeurs envoyées aux instructions ARG du Dockerfile pendant le build.
command	Remplace la commande par défaut exécutée dans le conteneur.
depends_on	Définit l'ordre de dépendance entre services. En syntaxe courte, cela n'attend pas automatiquement qu'un service soit réellement « healthy ».
environment	Ajoute des variables d'environnement au conteneur.
networks	Attache le service à un ou plusieurs réseaux. Le mot correct au niveau service est networks.
ports	Publie/mappe des ports entre l'hôte et le conteneur, ex. "8080:80".
volumes	Monte des données persistantes ou des dossiers dans le conteneur. Très important pour les bases de données.
restart	Définit une politique de redémarrage du conteneur, ex. unless-stopped.

6) Exemple minimal : Nginx en Compose

Dans nano compose.yaml

```
services:
  web:
    image: nginx:latest
    ports:
      - "8080:80"
```

Puis dans le terminal

```
docker compose config -q
docker compose up -d
docker compose ps
```

Question — À quoi sert docker compose config -q ?

Cette commande valide le fichier Compose avant de lancer quoi que ce soit. Si elle ne retourne pas d'erreur, la configuration est syntaxiquement et structurellement valide.

Question — Et ensuite ?

Dans le navigateur : http://IP_DE_LA_VM:8080. Le port 8080 de la VM est redirigé vers le port 80 du conteneur Nginx.

7) Les commandes Docker Compose à connaître

Commande	Ce qu'elle fait	À retenir
<code>docker compose up -d</code>	Crée et démarre les services.	-d = arrière-plan. Peut construire les images définies avec build.
<code>docker compose up -d web</code>	Démarre le service web.	Compose démarre aussi ses dépendances si nécessaire.
<code>docker compose down</code>	Arrête et supprime les conteneurs et réseaux du projet.	Les volumes nommés ne sont pas supprimés par défaut.
<code>docker compose down -v</code>	Supprime aussi les volumes.	Attention : peut supprimer les données persistantes.
<code>docker compose ps</code>	Affiche l'état des conteneurs du projet.	--all ou -a inclut les conteneurs arrêtés.
<code>docker compose logs</code>	Affiche les logs de tous les services.	Ajouter un nom de service pour filtrer.
<code>docker compose logs -f web</code>	Suit les logs du service web.	-f = follow.
<code>docker compose exec web sh</code>	Exécute une commande dans un service déjà lancé.	Equivalent de docker exec ciblé sur un service Compose.
<code>docker compose stop</code>	Arrête les services sans les supprimer.	Ils restent présents.
<code>docker compose start</code>	Redémarre les services arrêtés.	Réutilise les conteneurs existants.
<code>docker compose restart</code>	Redémarre les services.	Pratique après certaines modifications runtime.
<code>docker compose pull</code>	Télécharge les images déclarées.	Ne démarre pas les conteneurs.

Correction du support — docker compose down [svcName]

La documentation actuelle accepte des services avec down, mais retiens surtout que down agit sur le projet Compose et supprime les conteneurs/réseaux concernés. Pour simplement arrêter un service sans le supprimer, stop SERVICE est beaucoup plus clair.

8) Ce que fait réellement docker compose up -d

- Lit compose.yaml.
- Résout les variables et la configuration.
- Crée le réseau par défaut du projet si nécessaire.
- Télécharge les images absentes selon la politique de pull.
- Construit les images déclarées avec build si nécessaire.
- Crée ou recrée les conteneurs si leur configuration a changé.
- Démarre les services.
- Grâce à -d, rend la main au terminal et laisse les conteneurs tourner en arrière-plan.

9) Réseau Compose : pourquoi les conteneurs se trouvent sans IP ?

Question — Comment un conteneur web trouve-t-il la base de données ?

Par défaut, Compose crée un réseau pour le projet. Les services y sont joignables par leur nom de service grâce au DNS interne Docker.

```
services:
  web:
    image: monapp
    environment:
      DB_HOST: db

  db:
    image: mariadb:10s
```

Ici

Le service web peut contacter la base avec le nom db. Pas besoin de chercher manuellement l'adresse IP du conteneur MariaDB.

Question — Faut-il publier le port 3306 de MariaDB ?

Pas si seul le conteneur web doit parler à la base. Les services d'un même projet peuvent communiquer sur leur réseau Docker interne. On publie 3306 vers l'hôte uniquement si on a réellement besoin d'accéder à MariaDB depuis l'extérieur de ce réseau.

10) Lien direct avec notre installation GLPI

```
compose.yaml
↓
docker compose up -d
↓
Service GLPI <---- réseau Docker ----> Service base de données
↓
port 80 publié
↓
navigateur : http://IP_VM:80
```

Question — Pourquoi c'était plus simple que les commandes docker run du prof ?

Parce que le fichier Compose avait déjà mémorisé les images, les variables, le réseau, les volumes et les ports. docker compose up -d a ensuite créé les différentes briques automatiquement.

11) depends_on : à comprendre sans se compliquer

Question — Si j'écris depends_on: - db, est-ce que Compose attend que MariaDB soit prête ?

Pas avec la syntaxe courte. Compose garantit que le conteneur dépendant est démarré avant l'autre service, mais il ne garantit pas qu'il est déjà prêt à accepter des connexions. Pour attendre un vrai état de santé, il faut un healthcheck et une condition service_healthy.

12) Exercice pratique proposé — WordPress + MariaDB

Important

Cet exercice est proposé pour appliquer le cours. Ce n'est pas le sujet exact de « Exercice_02_Docker_Compose » du prof, car son énoncé complet n'a pas été fourni.

Objectif

- Créer une vraie application multi-conteneurs.
- 1 service WordPress = interface Web.
- 1 service MariaDB = base de données.
- Un volume garde les données de la base.
- Le réseau par défaut permet à WordPress de joindre MariaDB avec le nom db.

Étape 1 — créer le dossier

```
mkdir ~/exo-compose
cd ~/exo-compose
nano compose.yaml
```

Étape 2 — mettre ce YAML dans nano

```
services:
  db:
    image: mariadb:lts
    restart: unless-stopped
    environment:
      MARIADB_ROOT_PASSWORD: toor
      MARIADB_DATABASE: wordpress
      MARIADB_USER: wordpress
      MARIADB_PASSWORD: wordpress
    volumes:
      - db_data:/var/lib/mysql

  wordpress:
    image: wordpress:latest
    restart: unless-stopped
    depends_on:
      - db
    ports:
      - "8080:80"
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress

volumes:
  db_data:
```

13) Exercice pratique — lancement et vérifications

Étape 3 — valider le YAML

```
docker compose config -q
```

Résultat attendu

Aucune sortie = configuration valide. Une erreur YAML doit être corrigée avant de poursuivre.

Étape 4 — lancer

```
docker compose up -d
```

Étape 5 — vérifier

```
docker compose ps
docker compose logs db
docker compose logs wordpress
```

Étape 6 — ouvrir dans le navigateur

```
ip -br a  
  
http://IP_DE_LA_VM:8080
```

Question — Que dois-je voir ?

L'assistant Web de WordPress. WordPress utilise automatiquement le service db comme serveur de base de données grâce à WORDPRESS_DB_HOST: db:3306.

Étape 7 — tester le cycle de vie

```
docker compose stop  
docker compose ps --all  
docker compose start  
docker compose restart  
docker compose ps
```

Étape 8 — arrêter et supprimer les conteneurs

```
docker compose down
```

Pourquoi les données peuvent rester ?

Le volume nommé db_data n'est pas supprimé par docker compose down. Si tu relances docker compose up -d, MariaDB peut retrouver ses données.

Attention

docker compose down -v supprime aussi le volume db_data : les données de la base peuvent alors être perdues. À utiliser seulement si tu veux vraiment réinitialiser l'exercice.

14) Attention KVM/QEMU — correction du support

Le support indique : « glibc 9.x a besoin de AVX-AES »

Cette formulation n'est pas correcte telle quelle et je ne te conseille pas de l'apprendre. glibc n'est pas actuellement en version 9.x. Le support mélange vraisemblablement une contrainte historique de certaines images/versions de base de données avec les fonctions CPU exposées à une VM.

Question — Quelle précaution retenir alors ?

Si une image de base de données refuse de démarrer sous KVM/QEMU avec une erreur liée aux instructions CPU, vérifie le modèle de CPU présenté à la VM. Le mode « host » transmet davantage de capacités du processeur physique à la VM. Mais ce n'est pas une règle générale de Docker Compose ni une exigence universelle de MySQL/MariaDB.

Méthode de diagnostic

Ne change pas le CPU au hasard. Commence par `docker compose ps` puis `docker compose logs NOM_DU_SERVICE`. Si le journal mentionne explicitement une instruction CPU manquante, alors seulement vérifie la configuration CPU de la VM.

15) Mémo final — ce que tu dois savoir sans bourrage

Le cœur du chapitre

1. Compose = gérer une application composée de plusieurs conteneurs.
2. `compose.yaml` = la recette de toute l'application.
3. `services:` = les briques de l'application.
4. `docker compose up -d` = créer + démarrer la stack.
5. `docker compose ps` = voir son état.
6. `docker compose logs SERVICE` = comprendre un problème.
7. `docker compose stop/start` = arrêter/redémarrer sans supprimer.
8. `docker compose down` = supprimer les conteneurs/réseaux du projet.
9. `volumes` = garder les données importantes.
10. Les services se parlent par leur nom sur le réseau Compose.

16) Sources officielles vérifiées — août 2026

- Docker Docs — Docker Compose : définir et exécuter des applications multi-conteneurs.
- Docker Docs — installation du plugin Compose sous Linux.
- Docker Docs — Compose Specification et services.
- Docker Docs — commandes `docker compose up / down / ps / logs / exec / pull`.
- Docker Docs — réseau par défaut et découverte des services par leur nom.
- Docker Hub — image officielle MariaDB et tag `lts`.

```
https://docs.docker.com/compose/  
https://docs.docker.com/compose/install/linux/  
https://docs.docker.com/reference/compose-file/  
https://docs.docker.com/reference/cli/docker/compose/  
https://docs.docker.com/compose/how-tos/networking/  
https://hub.docker.com/_/mariadb
```