Mémo commandes Docker & Docker Compose

Documentation rédigée par : GADONNAUD Ewen

Formation : BTS SIO 1ère année - Option SISR

Établissement : Lycée Paul-Louis Courier, Tours

Date : Mars 2026

1. Docker — Conteneurs

Lister les conteneurs

```
docker ps          # Conteneurs en cours d'exécution
docker ps -a       # Tous les conteneurs (y compris arrêtés)
```

Démarrer / arrêter / redémarrer

```
docker start searxng
docker stop searxng
docker restart searxng
```

Supprimer un conteneur

```
docker rm searxng           # Conteneur arrêté
docker rm -f searxng        # Force la suppression même si actif
```

Exécuter une commande dans un conteneur actif

```
docker exec -it searxng bash        # Ouvre un shell interactif
docker exec searxng ls /etc/searxng # Commande one-shot
```

Consulter les logs

```
docker logs searxng
docker logs searxng --tail 50       # 50 dernières lignes
```

```
docker logs searxng -f              # Suivi en temps réel
```

Inspecter un conteneur

```
docker inspect searxng              # Détails complets (réseau, volumes, config...)
docker stats                        # Consommation CPU/RAM en temps réel
```

2. Docker — Images

Lister les images locales

```
docker images
```

Télécharger une image

```
docker pull searxng/searxng:latest
```

Supprimer une image

```
docker rmi searxng/searxng:latest
```

Mettre à jour une image

```
docker pull searxng/searxng:latest   # Récupère la nouvelle version
docker compose up -d                 # Recrée le conteneur avec la nouvelle image
```

3. Docker — Volumes

Lister les volumes

```
docker volume ls
```

Inspecter un volume (chemin réel sur l'hôte)

```
docker volume inspect searxng_searxng-cache
```

Supprimer les volumes orphelins

```
docker volume prune
```

4. Docker — Nettoyage

Supprimer toutes les ressources inutilisées (conteneurs arrêtés, images sans tag, volumes orphelins)

```
docker system prune           # Avec confirmation
docker system prune -a        # Inclut aussi les images non utilisées
docker system prune --volumes # Inclut les volumes (attention, irréversible)
```

Voir l'espace disque utilisé par Docker

```
docker system df
```

5. Docker Compose — Gestion des services

Les commandes suivantes s'exécutent depuis le dossier contenant le `docker-compose.yml`.

Démarrer les services

```
docker compose up -d          # Démarre en arrière-plan
docker compose up             # Démarre avec logs dans le terminal
```

Arrêter les services

```
docker compose down           # Arrête et supprime les conteneurs
docker compose down -v        # Idem + supprime les volumes associés
docker compose stop           # Arrête sans supprimer les conteneurs
```

Redémarrer

```
docker compose restart
docker compose restart searxng   # Redémarre un service spécifique
```

Voir l'état des services

```
docker compose ps
```

Consulter les logs

```
docker compose logs
docker compose logs -f           # Suivi en temps réel
docker compose logs searxng      # Logs d'un service spécifique
```

Forcer la recréation des conteneurs (après modif du docker-compose.yml)

```
docker compose up -d --force-recreate
```

Mettre à jour les images et recréer les conteneurs

```
docker compose pull              # Télécharge les nouvelles versions des
```

```
images
docker compose up -d             # Recrée les conteneurs avec les nouvelles
images
```

6. Structure d'un docker-compose.yml

Un fichier `docker-compose.yml` se compose de plusieurs sections de niveau racine. Voici la structure complète avec les options les plus courantes :

```yaml
version: '3'              # Version du format Compose

services:                 # Définition des conteneurs
  nom_du_service:
    image: image:tag      # Image Docker à utiliser
    container_name: nom   # Nom explicite du conteneur (optionnel)
    build: ./dossier      # Alternative à image : build depuis un Dockerfile
    ports:
      - "port_hote:port_conteneur"   # Ex: "8085:8080"
    volumes:
      - ./dossier_local:/chemin/conteneur        # Volume lié (bind mount)
      - ./fichier.yml:/etc/service/fichier.yml   # Fichier spécifique
      - nom_volume:/chemin/conteneur             # Volume nommé
    environment:          # Variables d'environnement
      - MA_VARIABLE=valeur
      - AUTRE_VAR=valeur
    env_file:             # Alternative : charger depuis un fichier .env
      - .env
    networks:             # Réseaux auxquels le service est connecté
      - mon_reseau
    depends_on:           # Ordre de démarrage
      - autre_service
    restart: unless-stopped   # Politique de redémarrage
    # always              → redémarre toujours
    # on-failure          → redémarre uniquement en cas d'erreur
    # unless-stopped      → redémarre sauf si arrêté manuellement
    # no                  → ne redémarre jamais
    dns:                  # Serveurs DNS explicites
      - 1.1.1.1
      - 8.8.8.8

volumes:                  # Déclaration des volumes nommés
  nom_volume:             # Docker gère le stockage automatiquement
  autre_volume:
    external: true        # Volume créé manuellement en dehors de Compose

networks:                 # Déclaration des réseaux personnalisés
  mon_reseau:
    driver: bridge        # bridge = réseau local entre conteneurs (défaut)
```

Les politiques de redémarrage en pratique

Valeur	Comportement	Cas d'usage
`no`	Ne redémarre jamais	Conteneurs de test
`always`	Redémarre toujours, même après un `docker stop`	Services critiques
`unless-stopped`	Redémarre sauf si arrêté manuellement	Homelab (recommandé)
`on-failure`	Redémarre uniquement si code d'erreur non nul	Jobs ponctuels

Bind mount vs Volume nommé

Type	Syntaxe	Quand l'utiliser
Bind mount	`./local:/conteneur`	Fichiers de config à éditer directement sur l'hôte
Volume nommé	`nom:/conteneur`	Données persistantes gérées par Docker (BDD, cache)

7. Cas pratiques homelab

Redémarrer SearXNG après modification de settings.yml

```
cd /opt/searxng
docker compose restart
```

Vérifier que le cache favicons a bien été créé

```
docker exec searxng ls -la /var/cache/searxng
```

Voir les logs de SearXNG en temps réel

```
cd /opt/searxng
docker compose logs -f
```

Accéder au shell d'un conteneur pour déboguer

```
docker exec -it searxng sh    # SearXNG utilise sh et non bash (image Alpine)
```

Vérifier la résolution DNS depuis l'intérieur d'un conteneur

```
docker exec searxng nslookup google.com
```

From:
https://wiki.ewengadonnaud.xyz/ - **Base de savoir réseaux/cyber/devops**

Permanent link:
https://wiki.ewengadonnaud.xyz/doku.php?id=infra:docker_memo

Last update: **2026/03/10 14:48**