

Contents

Cheatsheet Docker et Docker Compose	1
Images	1
Conteneurs	1
Volumes et réseaux	2
Dockerfile : instructions	2
Docker Compose v2 (docker compose, sans tiret)	3
Encadré : pièges fréquents	4

Cheatsheet Docker et Docker Compose

Image = la recette figée (comme une classe). **Conteneur** = une instance en cours d'exécution (comme un objet). **Registre** = l'endroit où l'on stocke les images (Docker Hub, ghcr.io).

Images

Commande	Effet
<code>docker pull nginx:1.27-alpine</code>	Télécharge une image
<code>docker images</code>	Liste les images locales
<code>docker build -t taskflow-api:dev ./api</code>	Construit une image depuis ./api/Dockerfile
<code>docker tag taskflow-api:dev ghcr.io/moi/taskflow-api:1.0.0</code>	Ajoute un nom (tag)
<code>docker push ghcr.io/moi/taskflow-api:1.0.0</code>	Envoie vers le registre (après <code>docker login ghcr.io</code>)
<code>docker rmi image / docker image prune</code>	Supprime une image / les images orphelines

Conteneurs

Commande	Effet
<code>docker run --rm hello-world</code>	Lance, affiche, supprime à la fin
<code>docker run -d --name web -p 8080:80 nginx:1.27-alpine</code>	En arrière-plan, port hôte:conteneur
<code>docker run -e POSTGRES_PASSWORD=secret postgres:16-alpine</code>	Variable d'environnement (--env-file .env pour un fichier)
<code>docker run -v pgdata:/var/lib/postgresql/data ...</code>	Volume nommé (données persistantes)
<code>docker run -v "\$PWD":/app ...</code>	Bind mount (dossier de l'hôte)
<code>docker run --network monreseau ...</code>	Branche sur un réseau
<code>docker ps / docker ps -a</code>	Conteneurs actifs / tous
<code>docker logs -f --tail 50 web</code>	Suit les logs
<code>docker exec -it web sh</code>	Ouvre un shell dedans (exit pour sortir)
<code>docker stop web / docker start web</code>	Arrête / relance

Commande	Effet
docker rm web / docker rm -f web	Supprime (- f = même s'il tourne)
docker inspect web	Détails JSON (IP, volumes, santé)
docker stats	Consommation CPU/RAM en direct

Volumes et réseaux

Commande	Effet
docker volume create pgdata / docker volume ls	Crée / liste
docker volume rm pgdata	Supprime (perte des données)
docker network create taskflow	Crée un réseau : les conteneurs s'y joignent par leur nom (db, api)
docker system df / docker system prune	Espace utilisé / nettoyage (conteneurs arrêtés, images inutilisées)

Dockerfile : instructions

```
# Étape 1 : construire (multi-stage)
FROM node:20-alpine AS build
WORKDIR /app
COPY package*.json ./          # d'abord les dépendances : couche mise en cache
RUN npm ci
COPY . .
RUN npm run build

# Étape 2 : image finale légère
FROM nginx:1.27-alpine
COPY --from=build /app/dist /usr/share/nginx/html
EXPOSE 80
HEALTHCHECK --interval=30s --timeout=3s CMD wget -q0- http://localhost/ || exit 1
```

Instruction	Rôle
FROM image:tag	Image de départ (toujours un tag précis, pas latest)
WORKDIR /app	Dossier de travail
COPY src dest	Copie des fichiers de votre projet dans l'image
RUN cmd	Commande au build (installer, compiler)
ENV CLE=valeur	Variable d'environnement par défaut
EXPOSE 3000	Documentation du port (ne publie rien)
USER node	Ne pas tourner en root
CMD ["node", "src/server.js"]	Commande au démarrage du conteneur
HEALTHCHECK	Test d'état de santé

.dockerignore (indispensable) : node_modules, .git, .env, dist, *.log. **Cache** : une instruction modifiée invalide toutes les suivantes. Mettez ce qui change rarement (dépendances) avant ce qui change souvent (code).

Docker Compose v2 (docker compose, sans tiret)

```
name: taskflow
services:
  db:
    image: postgres:16-alpine
    environment:
      POSTGRES_USER: taskflow
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}      # lu dans .env
      POSTGRES_DB: taskflow
    volumes:
      - pgdata:/var/lib/postgresql/data
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U taskflow"]
      interval: 5s
      timeout: 3s
      retries: 10
  api:
    build: ./api
    environment:
      DATABASE_URL: postgres://taskflow:${POSTGRES_PASSWORD}@db:5432/taskflow  # « c
    depends_on:
      db:
        condition: service_healthy
    restart: unless-stopped
  front:
    build: ./front
    ports:
      - "8080:80"
volumes:
  pgdata:
```

Commande	Effet
docker compose up -d	Crée et démarre (ajoutez --build pour reconstruire)
docker compose ps	État + santé des services
docker compose logs -f api	Logs d'un service (--since 10m, --tail 100)
docker compose exec db psql -U taskflow	Commande dans un service en cours
docker compose run --rm api npm test	Conteneur jetable
docker compose pull	Télécharge les nouvelles images
docker compose config	Affiche le fichier résolu (variables remplacées) : excellent pour déboguer

Commande	Effet
<code>docker compose restart api</code>	Redémarre un service
<code>docker compose down</code>	Arrête et supprime conteneurs + réseau (garde les volumes)
<code>docker compose down -v</code>	... et supprime aussi les volumes (données perdues)
<code>docker compose -f compose.prod.yaml up -d</code>	Utiliser un autre fichier

Variables : le fichier `.env` à côté du compose est lu automatiquement. `${VAR:-défaut}` donne une valeur par défaut. Ports "HOTE:CONTENEUR" : seul le côté hôte est visible depuis votre poste.

Encadré : pièges fréquents

- **localhost dans un conteneur = le conteneur lui-même**, pas votre poste ni la BDD. Entre services Compose, utilisez le **nom du service** (db).
- **Volume oublié pour la BDD** : `docker rm` ou `down -v` = données perdues.
- **Secret dans l'image** (ENV, COPY `.env`) : il y reste pour toujours. Passez-le à l'exécution (`env_file`, `-e`) et ajoutez `.env` au `.dockerignore`.
- **Tag latest** : ne dit pas quelle version tourne. Taguez avec la version ou le SHA du commit.
- **depends_on sans condition: service_healthy** n'attend pas que la BDD soit prête.
- **Port déjà utilisé** (`address already in use`) : changer le côté hôte (8081:80).
- **Modification du code sans effet** : il faut reconstruire (`--build`).
- Indentation YAML : espaces uniquement (jamais de tabulation), 2 espaces par niveau.
- `docker compose` (v2) et non `docker-compose` (ancienne v1).
- Le conteneur s'arrête aussitôt ? `docker logs <nom>` donne la raison dans 90 % des cas.