RÉFÉRENCE RAPIDE DOCKER

Conteneurs, images, volumes, réseau, compose

Bases

Lancer des conteneurs

```
docker run nginx              # run image
docker run -d nginx           # detached (background)
docker run -p 8080:80 nginx   # map port
docker run --name web nginx   # named container
docker run -it ubuntu bash    # interactive shell
```

Commandes essentielles

`docker ps`	Lister les conteneurs en cours
`docker ps -a`	Lister tous les conteneurs (y compris arrêtés)
`docker images`	Lister les images locales
`docker pull nginx`	Télécharger une image depuis le registre
`docker info`	Informations système générales

Gestion des conteneurs

Cycle de vie

`docker start <id>`	Démarrer un conteneur arrêté
`docker stop <id>`	Arrêt propre (SIGTERM)
`docker kill <id>`	Arrêt forcé (SIGKILL)
`docker restart <id>`	Redémarrer le conteneur
`docker rm <id>`	Supprimer un conteneur arrêté
`docker rm -f <id>`	Suppression forcée (même si en cours)

Inspection & Débogage

`docker logs <id>`	Afficher les logs du conteneur
`docker logs -f <id>`	Suivre les logs en temps réel
`docker exec -it <id> bash`	Ouvrir un shell dans le conteneur
`docker inspect <id>`	Métadonnées détaillées du conteneur (JSON)
`docker top <id>`	Processus en cours dans le conteneur
`docker stats`	Utilisation des ressources en temps réel

Copier des fichiers

```
docker cp file.txt <id>:/app/    # host → container
docker cp <id>:/app/log.txt ./   # container → host
```

Images

Construction & Tag

```
docker build -t myapp .          # build from Dockerfile
docker build -t myapp:v2 .       # with tag
docker tag myapp user/myapp:v2   # retag image
```

Publication

```
docker login
docker push user/myapp:v2
docker pull user/myapp:v2
```

Gestion des images

`docker images`	Lister toutes les images locales
`docker rmi <image>`	Supprimer une image
`docker image prune`	Supprimer les images orphelines
`docker system prune`	Supprimer toutes les données inutilisées
`docker history <image>`	Afficher l'historique des couches

Dockerfile

Instructions courantes

`FROM node:20`	Image de base
`WORKDIR /app`	Définir le répertoire de travail
`COPY . .`	Copier des fichiers dans l'image
`RUN npm install`	Exécuter une commande lors de la construction
`CMD ["node", "app.js"]`	Commande par défaut à l'exécution
`EXPOSE 3000`	Documenter le port d'écoute
`ENV NODE_ENV=production`	Définir une variable d'environnement
`ARG VERSION=latest`	Variable de construction
`ENTRYPOINT ["python"]`	Exécutable fixe (CMD = arguments)

Exemple de Dockerfile

```
FROM node:20-alpine
WORKDIR /app
COPY package*.json ./
RUN npm ci --production
COPY . .
EXPOSE 3000
CMD ["node", "server.js"]
```

Volumes

Stockage persistant

```
docker volume create mydata
docker run -v mydata:/app/data nginx
docker run -v $(pwd):/app nginx  # bind mount
```

Commandes de volume

`docker volume ls`	Lister les volumes
`docker volume inspect <v>`	Détails du volume
`docker volume rm <v>`	Supprimer un volume
`docker volume prune`	Supprimer les volumes inutilisés

Réseau

Bases du réseau

```
docker network create mynet
docker run --network mynet --name api nginx
docker run --network mynet --name db postgres
```

Commandes réseau

`docker network ls`	Lister les réseaux
`docker network inspect <n>`	Détails du réseau
`docker network connect <n> <c>`	Attacher un conteneur au réseau
`docker network rm <n>`	Supprimer un réseau

Les conteneurs sur le même réseau se joignent par leur nom

Docker Compose

Exemple compose.yaml

```
services:
  web:
    build: .
    ports: ["3000:3000"]
    depends_on: [db]
  db:
    image: postgres:16
    environment:
      POSTGRES_PASSWORD: secret
    volumes: [pgdata:/var/lib/postgresql/data]
volumes:
  pgdata:
```

Commandes Compose

`docker compose up`	Démarrer tous les services
`docker compose up -d`	Démarrer en arrière-plan
`docker compose down`	Arrêter et supprimer les conteneurs
`docker compose down -v`	Supprimer aussi les volumes
`docker compose build`	Reconstruire les images
`docker compose logs -f`	Suivre les logs de tous les services
`docker compose ps`	Lister les services en cours
`docker compose exec web bash`	Ouvrir un shell dans un service

Recettes utiles

Nettoyage

```
docker system prune -a          # remove all unused
docker container prune           # remove stopped
docker image prune -a            # remove unused images
```

Raccourcis pratiques

`Conteneur temporaire`	`docker run --rm -it alpine sh`
`Vérifier les ports`	`docker port <id>`
`Variables d'env`	`docker run -e KEY=val image`
`Fichier d'env`	`docker run --env-file .env image`
`Politique de redémarrage`	`docker run --restart unless-stopped image`
`Limite de ressources`	`docker run --memory 512m --cpus 1 image`