REFERÊNCIA RÁPIDA DE DOCKER

Containers, imagens, volumes, redes, compose

Básico

Executando Containers

```
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
```

Comandos Essenciais

`docker ps`	Listar containers em execução
`docker ps -a`	Listar todos os containers (incluindo parados)
`docker images`	Listar imagens locais
`docker pull nginx`	Baixar imagem do registry
`docker info`	Informações gerais do sistema

Gerenciamento de Containers

Ciclo de Vida

`docker start <id>`	Iniciar container parado
`docker stop <id>`	Parar com segurança (SIGTERM)
`docker kill <id>`	Forçar parada (SIGKILL)
`docker restart <id>`	Reiniciar container
`docker rm <id>`	Remover container parado
`docker rm -f <id>`	Forçar remoção (mesmo em execução)

Inspeção & Debugging

`docker logs <id>`	Ver logs do container
`docker logs -f <id>`	Seguir logs em tempo real
`docker exec -it <id> bash`	Abrir shell no container
`docker inspect <id>`	Metadados detalhados do container (JSON)
`docker top <id>`	Processos em execução no container
`docker stats`	Uso de recursos em tempo real

Copiar Arquivos

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

Imagens

Build & Tag

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

Publicar

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

Gerenciar Imagens

`docker images`	Listar todas as imagens locais
`docker rmi <image>`	Remover imagem
`docker image prune`	Remover imagens sem referência
`docker system prune`	Remover todos os dados não usados
`docker history <image>`	Exibir histórico de camadas da imagem

Dockerfile

Instruções Comuns

`FROM node:20`	Imagem base
`WORKDIR /app`	Definir diretório de trabalho
`COPY . .`	Copiar arquivos para a imagem
`RUN npm install`	Executar comando durante o build
`CMD ["node", "app.js"]`	Comando padrão em tempo de execução
`EXPOSE 3000`	Documentar porta que o container escuta
`ENV NODE_ENV=production`	Definir variável de ambiente
`ARG VERSION=latest`	Variável de build
`ENTRYPOINT ["python"]`	Executável fixo (CMD = argumentos)

Exemplo de Dockerfile

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

Volumes

Armazenamento Persistente

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

Comandos de Volume

`docker volume ls`	Listar volumes
`docker volume inspect <v>`	Detalhes do volume
`docker volume rm <v>`	Remover volume
`docker volume prune`	Remover volumes não usados

Redes

Básico de Rede

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

Comandos de Rede

`docker network ls`	Listar redes
`docker network inspect <n>`	Detalhes da rede
`docker network connect <n> <c>`	Conectar container à rede
`docker network rm <n>`	Remover rede

Containers na mesma rede podem se comunicar pelo nome

Docker Compose

Exemplo 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:
```

Comandos Compose

`docker compose up`	Iniciar todos os serviços
`docker compose up -d`	Iniciar em segundo plano
`docker compose down`	Parar e remover containers
`docker compose down -v`	Também remover volumes
`docker compose build`	Reconstruir imagens
`docker compose logs -f`	Seguir logs de todos os serviços
`docker compose ps`	Listar serviços em execução
`docker compose exec web bash`	Abrir shell em um serviço

Padrões Úteis

Comandos de Limpeza

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

Receitas Rápidas

`Container temporário`	`docker run --rm -it alpine sh`
`Verificar porta`	`docker port <id>`
`Variáveis de ambiente`	`docker run -e KEY=val image`
`Arquivo de env`	`docker run --env-file .env image`
`Política de restart`	`docker run --restart unless-stopped image`
`Limite de recursos`	`docker run --memory 512m --cpus 1 image`