DOCKER 快速参考

容器、镜像、卷、网络、Compose

基础

运行容器

```
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
```

常用命令

`docker ps`	列出运行中的容器
`docker ps -a`	列出所有容器（含已停止）
`docker images`	列出本地镜像
`docker pull nginx`	从仓库拉取镜像
`docker info`	查看系统全局信息

容器管理

生命周期

`docker start <id>`	启动已停止的容器
`docker stop <id>`	优雅停止（SIGTERM）
`docker kill <id>`	强制停止（SIGKILL）
`docker restart <id>`	重启容器
`docker rm <id>`	删除已停止的容器
`docker rm -f <id>`	强制删除（即使在运行）

检查与调试

`docker logs <id>`	查看容器日志
`docker logs -f <id>`	实时跟踪日志
`docker exec -it <id> bash`	进入运行中的容器 shell
`docker inspect <id>`	容器详细元数据（JSON）
`docker top <id>`	查看容器中的进程
`docker stats`	实时资源使用情况

复制文件

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

镜像

构建与打标签

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

发布

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

镜像管理

`docker images`	列出所有本地镜像
`docker rmi <image>`	删除镜像
`docker image prune`	删除悬空镜像
`docker system prune`	删除所有未使用的数据
`docker history <image>`	查看镜像层历史

Dockerfile

常用指令

`FROM node:20`	基础镜像
`WORKDIR /app`	设置工作目录
`COPY . .`	将文件复制到镜像
`RUN npm install`	构建时执行命令
`CMD ["node", "app.js"]`	运行时默认命令
`EXPOSE 3000`	声明监听端口
`ENV NODE_ENV=production`	设置环境变量
`ARG VERSION=latest`	构建时变量
`ENTRYPOINT ["python"]`	固定可执行程序（CMD 作为参数）

Dockerfile 示例

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

卷

持久化存储

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

卷命令

`docker volume ls`	列出所有卷
`docker volume inspect <v>`	查看卷详情
`docker volume rm <v>`	删除卷
`docker volume prune`	删除未使用的卷

网络

网络基础

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

网络命令

`docker network ls`	列出网络
`docker network inspect <n>`	网络详情
`docker network connect <n> <c>`	将容器加入网络
`docker network rm <n>`	删除网络

同一网络上的容器可通过名称互相访问

Docker Compose

compose.yaml 示例

```
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:
```

Compose 命令

`docker compose up`	启动所有服务
`docker compose up -d`	后台启动
`docker compose down`	停止并删除容器
`docker compose down -v`	同时删除卷
`docker compose build`	重新构建镜像
`docker compose logs -f`	实时跟踪所有服务日志
`docker compose ps`	列出运行中的服务
`docker compose exec web bash`	进入某服务的 shell

实用模式

清理命令

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

快速示例

`Temp container`	`docker run --rm -it alpine sh`
`Port check`	`docker port <id>`
`Env vars`	`docker run -e KEY=val image`
`Env file`	`docker run --env-file .env image`
`Restart policy`	`docker run --restart unless-stopped image`
`Resource limit`	`docker run --memory 512m --cpus 1 image`