DOCKER QUICK REFERENCE

Basics

Running 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
```

Essential Commands

`docker ps`	List running containers
`docker ps -a`	List all containers (including stopped)
`docker images`	List local images
`docker pull nginx`	Download image from registry
`docker info`	System-wide information

Container Management

Lifecycle

`docker start <id>`	Start a stopped container
`docker stop <id>`	Graceful stop (SIGTERM)
`docker kill <id>`	Force stop (SIGKILL)
`docker restart <id>`	Restart container
`docker rm <id>`	Remove stopped container
`docker rm -f <id>`	Force remove (even if running)

Inspection & Debugging

`docker logs <id>`	View container logs
`docker logs -f <id>`	Follow logs (live)
`docker exec -it <id> bash`	Shell into running container
`docker inspect <id>`	Detailed container metadata (JSON)
`docker top <id>`	Running processes in container
`docker stats`	Live resource usage

Copying Files

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

Images

Building & Tagging

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

Publishing

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

Image Management

`docker images`	List all local images
`docker rmi <image>`	Remove image
`docker image prune`	Remove dangling images
`docker system prune`	Remove all unused data
`docker history <image>`	Show image layer history

Dockerfile

Common Instructions

`FROM node:20`	Base image
`WORKDIR /app`	Set working directory
`COPY . .`	Copy files into image
`RUN npm install`	Run command during build
`CMD ["node", "app.js"]`	Default command at runtime
`EXPOSE 3000`	Document listening port
`ENV NODE_ENV=production`	Set environment variable
`ARG VERSION=latest`	Build-time variable
`ENTRYPOINT ["python"]`	Fixed executable (CMD = args)

Example Dockerfile

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

DOCKER QUICK REFERENCE (continued)

Volumes

Persistent Storage

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

Volume Commands

`docker volume ls`	List volumes
`docker volume inspect <v>`	Volume details
`docker volume rm <v>`	Remove volume
`docker volume prune`	Remove unused volumes

Networks

Network Basics

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

Network Commands

`docker network ls`	List networks
`docker network inspect <n>`	Network details
`docker network connect <n> <c>`	Attach container to network
`docker network rm <n>`	Remove network

Containers on the same network can reach each other by name

Docker Compose

Example 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:
```

Compose Commands

`docker compose up`	Start all services
`docker compose up -d`	Start in background
`docker compose down`	Stop and remove containers
`docker compose down -v`	Also remove volumes
`docker compose build`	Rebuild images
`docker compose logs -f`	Follow all service logs
`docker compose ps`	List running services
`docker compose exec web bash`	Shell into a service

Useful Patterns

Cleanup Commands

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

Quick Recipes

Temp container	`docker run --rm -it alpine sh`
Port check	`docker port`
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`