

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

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

Container Management

Lifecycle

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

Inspection & Debugging

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

<code>docker images</code>	List all local images
<code>docker rmi <image></code>	Remove image

<code>docker image prune</code>	Remove dangling images
<code>docker system prune</code>	Remove all unused data
<code>docker history <image></code>	Show image layer history

Dockerfile

Common Instructions

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

<code>docker volume ls</code>	List volumes
<code>docker volume inspect <v></code>	Volume details
<code>docker volume rm <v></code>	Remove volume
<code>docker volume prune</code>	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

<code>docker network ls</code>	List networks
<code>docker network inspect <n></code>	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

<code>docker compose up</code>	Start all services
<code>docker compose up -d</code>	Start in background
<code>docker compose down</code>	Stop and remove containers
<code>docker compose down -v</code>	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

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