Docker & Compose Cheatsheet

Production-ready commands, patterns & compose templates

Container Lifecycle

Run and auto-remove

```
docker run --rm -it ubuntu bash
```

Run detached with restart

```
docker run -d --restart unless-stopped \
  --name myapp -p 8080:80 nginx
```

Execute into running container

```
docker exec -it container_name bash
```

Copy files from container

```
docker cp container:/app/logs ./logs
```

Follow logs with timestamps

```
docker logs -f --timestamps container
```

Inspect container details

```
docker inspect --format '{{.State.Status}}' container
```

Image Management

Build with cache busting

```
docker build --no-cache -t app:v2 .
```

Multi-stage build (slim images)

```
# Dockerfile
FROM node:20 AS builder
RUN npm ci && npm run build

FROM node:20-slim
COPY --from=builder /app/dist ./dist
CMD ["node", "dist/index.js"]
```

Tag and push

```
docker tag app:latest registry.io/app:v1.2
docker push registry.io/app:v1.2
```

Prune everything unused

```
docker system prune -a --volumes
```

Show image layer sizes

```
docker history --no-trunc image:tag
```

Networking

Create custom network

```
docker network create --driver bridge mynet
```

Run on custom network

```
docker run --network mynet --name api app
```

Container DNS resolution

```
# Containers on same network resolve
# each other by name automatically
curl http://api:3000/health
```

Expose specific IP only

```
docker run -p 127.0.0.1:8080:80 nginx
```

Map UDP port

```
docker run -p 53:53/udp dns-server
```

Docker Compose Essentials

Start all services

```
docker compose up -d
```

Rebuild and restart

```
docker compose up -d --build --force-recreate
```

Scale a service

```
docker compose up -d --scale worker=5
```

View logs for one service

```
docker compose logs -f api
```

Run one-off command

```
docker compose run --rm api npm test
```

Stop and remove everything

```
docker compose down -v --rmi all
```

Production Compose Template

Full stack example

```
services:
  api:
    build: .
    restart: unless-stopped
    env_file: .env
    ports: ["3000:3000"]
    depends_on:
      db: { condition: service_healthy }
    healthcheck:
      test: curl -f http://localhost:3000/health
      interval: 30s
      retries: 3
    deploy:
      resources:
        limits: { cpus: '1', memory: 512M }
  db:
    image: postgres:16
    volumes: [db_data:/var/lib/postgresql/data]
    environment:
      POSTGRES_PASSWORD_FILE: /run/secrets/db_pass
    healthcheck:
      test: pg_isready -U postgres
      interval: 10s
volumes:
  db_data:
secrets:
  db_pass:
    file: ./secrets/db_password.txt
```

Debugging & Performance

Resource usage stats

```
docker stats --no-stream
```

Check why container exited

```
docker inspect --format '{{.State.ExitCode}}' c
```

Live filesystem changes

```
docker diff container_name
```

Export container filesystem

```
docker export container > fs.tar
```

Benchmark build time

```
DOCKER_BUILDKIT=1 docker build \
  --progress=plain .
```

Security Best Practices

Run as non-root user

```
# Dockerfile
RUN adduser --disabled-password app
USER app
```

Read-only filesystem

```
docker run --read-only \
  --tmpfs /tmp nginx
```

Drop all capabilities

```
docker run --cap-drop ALL \
  --cap-add NET_BIND_SERVICE app
```

Scan for vulnerabilities

```
docker scout cves image:tag
```

No new privileges

```
docker run --security-opt no-new-privileges app
```