

Docker 빠른 참조

컨테이너, 이미지, 볼륨, 네트워킹, Compose

기본

컨테이너 실행

```
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
```

필수 명령

```
docker ps 실행 중인 컨테이너 목록
docker ps -a 모든 컨테이너 목록 (중지된 것 포함)
docker images 로컬 이미지 목록
docker pull nginx 레지스트리에서 이미지 다운로드
docker info 시스템 전체 정보
```

컨테이너 관리

생명 주기

```
docker start <id> 중지된 컨테이너 시작
docker stop <id> 정상 중지 (SIGTERM)
docker kill <id> 강제 중지 (SIGKILL)
docker restart <id> 컨테이너 재시작
docker rm <id> 중지된 컨테이너 제거
docker rm -f <id> 강제 제거 (실행 중이어도)
```

검사 & 디버깅

```
docker logs <id> 컨테이너 로그 보기
docker logs -f <id> 로그 실시간 팔로우
docker exec -it <id> bash 실행 중인 컨테이너에 셸 접속
docker inspect <id> 컨테이너 상세 메타데이터 (JSON)
docker top <id> 컨테이너 내 실행 중인 프로세스
docker stats 실시간 리소스 사용량
```

파일 복사

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

이미지

빌드 & 태깅

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

게시

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

이미지 관리

```
docker images 모든 로컬 이미지 목록
docker rmi <image> 이미지 제거
docker image prune 탱글링 이미지 제거
docker system prune 미사용 데이터 모두 제거
docker history <image> 이미지 레이어 히스토리 보기
```

Dockerfile

주요 명령어

```
FROM node:20 기본 이미지
WORKDIR /app 작업 디렉터리 설정
COPY . . . 이미지에 파일 복사
RUN npm install 빌드 중 명령 실행
CMD ["node", "app.js"] 런타임 기본 명령
EXPOSE 3000 리스닝 포트 문서화
ENV NODE_ENV=production 환경 변수 설정
ARG VERSION=latest 빌드 시간 변수
ENTRYPOINT ["python"] 고정 실행 파일 (CMD는 인수)
```

Dockerfile 예시

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

볼륨

영구 스토리지

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

볼륨 명령

```
docker volume ls 볼륨 목록
docker volume inspect <v> 볼륨 세부 정보
docker volume rm <v> 볼륨 제거
docker volume prune 미사용 볼륨 제거
```

네트워킹

네트워킹 기본

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

네트워킹 명령

```
docker network ls 네트워크 목록
docker network inspect <n> 네트워크 세부 정보
docker network connect <n> <c> 컨테이너를 네트워크에 연결
docker network rm <n> 네트워크 제거
```

같은 네트워크의 컨테이너는 이름으로 서로 접근 가능

Docker Compose

compose.yaml 예시

```
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 명령

```
docker compose up 모든 서비스 시작
docker compose up -d 백그라운드에서 시작
docker compose down 컨테이너 중지 및 제거
docker compose down -v 볼륨도 함께 제거
docker compose build 이미지 재빌드
docker compose logs -f 모든 서비스 로그 팔로우
docker compose ps 실행 중인 서비스 목록
docker compose exec web bash 서비스에 셸 접속
```

유용한 패턴

정리 명령

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

빠른 레시피

```
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
```