


```
version info
ubuntu 24.04.3 LTS
docker 28.4.0
docker-compose v2.39.4
```

참고 23강 linux에서 실행 시 aws 인바운드 규칙 설정

Table of Contents

- Docker 
 - Table of Contents
 - Section1. 도커 강의 소개
 - Section2. 배경지식 이해
 - DevOps란?
 - Docker란?
 - 리눅스
 - 리눅스(Linux) 활용
 - Section3. 클라우드 서비스(AWS 서버 구축)
 - 클라우드 컴퓨팅 설정
 - 클라우드 컴퓨팅 설정 - 리눅스 설치
 - Section5. 도커를 위한 리눅스 사용법 요약
 - 리눅스와 파일
 - 셸 종류
 - 다양한 명령어
 - `chmod` : 파일 권한 변경
 - 리눅스 셸 사용법 이해 - 리다이렉션/파이프
 - Standard Streams
 - 리다이렉션(Redirection)
 - 파이프(Pipe)
 - 리눅스 셸 사용법 이해 - 프로세스 관리
 - 프로세스 vs 바이너리
 - foreground / background process
 - 프로세스 상태 확인 - `ps` 명령어
 - 프로세스 중지
 - 리눅스 셸 사용법 이해 - 하드링크와 소프트(심볼릭) 링크
 - 하드링크와 소프트링크
 - 리눅스 셸 사용법 이해 - 우분투 패키지 매니저
 - ubuntu 패키지 관리 실무
 - 리눅스 셸 사용법 이해 - VIM 사용법
 - 초간단 사용법
 - Section6. 리눅스, 맥, 윈도우에서의 도커 환경 구축
 - Mac / Windows Docker 설치
 - Mac 설치
 - Windows 설치(강의 Section6-16 참조)
 - Linux Docker 설치
 - 설치 방법
 - `sudo` 없이 사용하기
 - Docker compose 설치(Standalone)
 - Section7. docker 주요 명령 익히기
 - 도커에 대한 기본 이해
 - docker 이미지 기본
 - docker 주요 명령어 익히기
 - Docker Container 관련 주요 명령
 - 웹서버로 docker run 옵션 테스트해보기
 - apache 웹서버 도커 이미지 다운로드
 - 이미지 다운로드받고 바로 컨테이너로 만들어 실행시키기(`-p` 옵션 이해하기)
 - 나만의 웹서비스 docker 만들기(`-v` 옵션 이해하기)
 - alpine 리눅스 기반 경량 웹서버 도커 이미지 사용
 - 도커가 사용 중인 디스크 용량 확인
 - 실행중인 컨테이너 리소스 사용량 확인
 - 실행중인 컨테이너에 명령 실행하기
 - 모든 컨테이너 삭제
 - Section8. Dockerfile 사용법 기본
 - Dockerfile 주요 명령어
 - Dockerfile로 이미지 작성
 - Dockerfile 예시
 - 주요 옵션

- Docker Image 조사(`docker inspect`)
- Docker 가끔 사용하는 기타 명령어들
- `ENTRYPOINT`
- `EXPOSE`
- `ENV` : 환경 변수 설정
- `WORKDIR` : 작업 디렉토리 설정
- Docker DB 설정 예시

Section1. 도커 강의 소개

Section2. 배경지식 이해

DevOps란?

- Release System 자동화
- 코드 리뷰, 테스트 자동화
- 서비스 모니터링 시스템
- 이슈 발생 시 커뮤니케이션 시스템

Docker란?

- 쿠버네티스(Kubernetes)와 함께 사용
- Jenkins, Travis CI 등과 함께 사용
 - 배포 자동화
- 무중단 배포

리눅스

- 1969년: 리눅스의 시작
- 1991년: 리눅스 커널 발표
- 1992년: 첫 번째 리눅스 배포판 등장
- 2000년대: 리눅스의 상업적 성공

리눅스(Linux) 활용

- 서버에 특화된 운영체제
- 클라우드 컴퓨팅(AWS, GCP, Azure)
 - [AWS](#)
 - [GCP](#)
 - [Azure](#)
- 리눅스 토발즈(Linus Torvalds)가 개발
- GPL(General Public License) 라이선스

Section3. 클라우드 서비스(AWS 서버 구축)

클라우드 컴퓨팅 설정

- Amazon Web Services (AWS)
 - <https://aws.amazon.com/ko/free/>
 - 무료로 12개월 사용 가능
 - 리눅스 사용 후 계정을 닫으면 됨
- AWS Free Tier 서비스 가입
 - 준비물
 - 신용카드(해외 결제가 가능한)
 - email 계정
- AWS 서비스
 - EC2 인스턴스 시작

클라우드 컴퓨팅 설정 - 리눅스 설치

1. EC2(서버) 생성
2. Elastic IP(탄력적 IP, 고정 IP) 생성
3. 자기 PC에서 EC2(서버) 접속
 - Windows: **PuTTY** 프로그램 사용 필수, Mac: 터미널

- Putty 연결방법 : https://docs.aws.amazon.com/ko_kr/AWSEC2/latest/UserGuide/connect-linux-inst-from-windows.html

```
cd ../path/to/your/.pem/file
chmod 400 your-key-name.pem
ssh -i "your-key-name.pem" ubuntu@your-ec2-public-ip
```

chmod 란?

- chmod: Change Mode
- 파일이나 디렉토리의 권한을 변경하는 명령어

Mac환경 예시

```
cd /Users/kanghwan/Documents/MyStudy📁/Docker🐳/KeyPair
chmod 400 temp.pem
ssh -i "temp.pem" ubuntu@43.201.229.172
# 종료는 exit
```

```
KeyPair cd /Users/kanghwan/Documents/MyStudy📁/Docker🐳/KeyPair
chmod 400 temp.pem
ssh -i "temp.pem" ubuntu@43.201.229.172
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1011-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Wed Sep 24 14:20:38 UTC 2025

System load:  0.0          Temperature:   -273.1 C
Usage of /:   8.6% of 28.0GB Processes:      112
Memory usage: 26%         Users logged in: 0
Swap usage:   0%          IPv4 address for ens5: 172.31.34.63

 * Ubuntu Pro delivers the most comprehensive open source security and
 * compliance features.
 * https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

12 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

*** System restart required ***
Last login: Wed Sep 24 14:20:38 2025 from 118.42.77.42
to run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-34-63:~$
```

Section5. 도커를 위한 리눅스 사용법 요약

리눅스와 파일

- 모든 것을 파일이라는 철학
 - 모든 인터랙션은 파일을 읽고, 쓰는 것처럼 동작
- 파일 네임스페이스
 - 전역 네임스페이스 사용
 - / (root) 디렉토리부터 시작

셸 종류

- Bourne-Again Shell (bash): GNU 프로젝트의 일환으로 개발된 유닉스 셸

다양한 명령어

리눅스에는 휴지통이 없음, 삭제 시 복구 불가. 주의!! ⚠

1. **whoami** : 현재 사용자의 이름을 출력
 - root: 최고 관리자
2. **sudo** : superuser do
 - root 권한으로 명령어 실행
 - 사용 예시: **sudo apt-get update**
3. **pwd** : 현재 작업 중인 디렉토리 경로 확인
4. **ls** : 파일/디렉토리 목록 보기
 - 숨김 포함: **ls -la**
 - 한 줄 출력: **ls -l**
 - 재귀 목록: **ls -R**
5. **cd** : 디렉토리 이동

- 상위로: `cd ..`
- 홈으로: `cd ~`
- 6. `touch`, `mkdir`, `rm`
 - 파일 생성: `touch file.txt`
 - 디렉토리 생성(하위 포함): `mkdir -p dir/subdir`
 - 파일 삭제: `rm file.txt`
 - 디렉토리/재귀 삭제: `rm -rf dir/`
 - `-r`: 재귀, `-f`: 강제
- 7. `cp`, `mv`
 - 파일/폴더 복사: `cp src dst`, 디렉토리 복사: `cp -r src/ dst/`
 - 이동/이름 변경: `mv old new`
- 8. `cat`, `less`, `head`, `tail`
 - 내용 전체 보기: `cat file.txt`
 - 페이지 단위 보기: `less file.txt` (종료: q)
 - 앞 N줄: `head -n 20 file.txt`
 - 마지막 N줄/실시간: `tail -n 100 -f file.txt`
- 9. `grep`, `find`: 검색
 - 텍스트 검색: `grep -n "pattern" file.txt`
 - 디렉토리 전체 검색: `grep -R "pattern" .`
 - `-i`: 대소문자 무시 - `-v`: 패턴과 일치하지 않는 라인 출력 - `-c`: 일치하는 라인 수 출력 - `-l`: 일치하는 파일 이름 출력 - `-c`: 일치하는 라인 수 출력 - `-n`: 일치하는 라인 번호 출력
 - 파일 찾기: `find . -type f -name "*.log"`
- 10. `chmod`, `chown`: 권한/소유자
 - 권한 변경: `chmod 644 file.txt`, 실행권한 추가: `chmod +x script.sh`
 - 소유자 변경: `sudo chown user:group file.txt`
- 11. `df`, `du`, `free`: 디스크/메모리
 - 디스크 사용량: `df -h`
 - 폴더별 용량: `du -sh /*`
 - 메모리 사용량: `free -h` (Ubuntu 등)
- 12. `ps`, `top`, `kill`: 프로세스
 - 프로세스 목록: `ps aux | grep bash`
 - 실시간 모니터링: `top` (또는 `htop` 설치 시)
 - 종료: `kill PID`, 강제 종료: `kill -9 PID`
- 13. `systemctl`/`service`: 서비스 관리
 - 상태 확인: `sudo systemctl status docker`
 - 시작/중지/재시작: `sudo systemctl start|stop|restart docker`
- 14. 네트워크 관련
 - IP 확인: `ip a` (또는 `ifconfig`)
 - 연결 확인: `ping -c 4 google.com`
 - 포트/소켓: `ss -lntp` (또는 `netstat -lntp`)
 - HTTP 확인: `curl -I http://localhost:80`
- 15. 압축/아카이브
 - 만들기: `tar -czf archive.tgz dir/`
 - 풀기: `tar -xzf archive.tgz`
 - zip/unzip: `zip -r archive.zip dir/`, `unzip archive.zip`
- 16. 패키지 관리
 - Debian/Ubuntu: `sudo apt update && sudo apt install <pkg>`
 - RHEL/CentOS: `sudo yum install <pkg>` (또는 `dnf`)
- 17. 사용자/그룹
 - 현재 사용자 정보: `id`
 - 사용자 추가: `sudo useradd -m <user>`
 - 비밀번호 설정: `sudo passwd <user>`
- 18. 시스템 정보
 - 커널/OS: `uname -a`
 - 배포판: `cat /etc/os-release`
- 19. 환경변수/경로
 - 확인: `echo $PATH`
 - 설정(세션 한정): `export KEY=value`
- 20. 리다이렉션/파이프
 - 출력 저장: `command > out.txt`, 추가 저장: `command >> out.txt`
 - 결과 연결: `command1 | command2`
- 21. 파일 목록 추출/저장 팁
 - 현재 디렉토리 파일만: `ls -l > files.txt`
 - 재귀적으로 전체 파일: `find . -type f > files.txt`
 - 개수 세기: `find . -type f | wc -l`

chmod : 파일 권한 변경

- 숫자를 사용하는 방법
- 읽기(r)=4, 쓰기(w)=2, 실행(x)=1
- 소유자, 그룹, 기타 사용자 순서로 합산

- 예시: `chmod 755 file.sh` (소유자 rwx(7), 그룹 rx(5), 기타 rx(5))

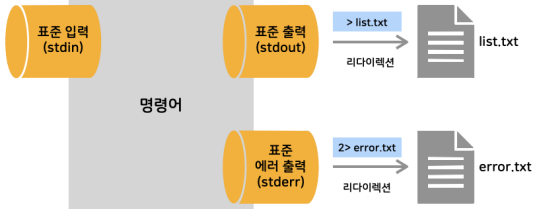
리눅스 셸 사용법 이해 - 리다이렉션/파이프

Standard Streams

- 표준 입력(Standard Input, **stdin**, 0): 키보드 입력
- 표준 출력(Standard Output, **stdout**, 1): 화면 출력
- 표준 에러(Standard Error, **stderr**, 2): 에러 메시지 출력

리다이렉션(Redirection)

- 표준 스트림 흐름 변경



- >**: 표준 출력 리다이렉션 (덮어쓰기)
 - 예시: `command > file.txt` (출력 내용을 file.txt에 저장)
- >>**: 표준 출력 리다이렉션 (추가쓰기)
 - 예시: `command >> file.txt` (출력 내용을 file.txt에 추가)
- 2>**: 표준 에러 리다이렉션 (덮어쓰기)
 - 예시: `command 2> error.txt` (에러 메시지를 error.txt에 저장)
- 2>>**: 표준 에러 리다이렉션 (추가쓰기)
 - 예시: `command 2>> error.txt` (에러 메시지를 error.txt에 추가)

파이프(Pipe)

- |**: 한 명령어의 출력을 다음 명령어의 입력으로 연결
 - 예시: `command1 | command2` (command1의 출력을 command2의 입력으로 사용)

리눅스 셸 사용법 이해 - 프로세스 관리

프로세스 vs 바이너리

- 코드 이미지 또는 바이너리: 실행파일(`.exe` 등)
- 실행 중인 프로그램: 프로세스

foreground / background process

- foreground: 터미널에서 직접 실행, 종료 시까지 터미널 점유
- background: 터미널에서 실행 후 백그라운드로 전환, 터미널 점유하지 않음
 - &** 기호 사용: `command &`

```
sleep 100 &
# [1] 12345
# [1]: 작업 번호, 12345: 프로세스 ID (PID)
```

- jobs** 명령어로 백그라운드 작업 확인

프로세스 상태 확인 - ps 명령어

- 사용법: `ps [options(s)]`
- options(s)
 - a**: 터미널에 속하지 않은 프로세스 포함
 - u**: 사용자 중심의 상세 정보 출력
 - x**: 터미널에 속하지 않은 프로세스 포함
 - l**: 긴 형식의 자세한 정보 출력

- `-e` or `-A`: 모든 프로세스 표시
- `-f`: 프로세스 간 관계 정보도 출력
- 주요 ps 출력 정보 항목
 - USER: 프로세스 소유자
 - PID: 프로세스 ID
 - %CPU: CPU 사용률
 - %MEM: 메모리 사용률
 - VSZ: 가상 메모리 크기 (KB)
 - RSS: 실제 메모리 사용량 (KB)
 - STAT: 프로세스 상태
 - R: 실행 중 (Running)
 - S: 대기 중 (Sleeping)
 - D: 중단 불가능한 대기 (Uninterruptible Sleep)
 - T: 중지됨 (Stopped)
 - Z: 좀비 프로세스 (Zombie)
 - START: 프로세스 시작 시간
 - TIME: 현재까지 사용된 CPU 사용 시간
 - COMMAND: 실행된 명령어

프로세스 중지

- `kill` 명령어로 프로세스 종료
 - 사용법: `kill [signal] PID`
 - 주요 signal
 - `SIGTERM (15)`: 정상 종료 요청 (기본값)
 - `SIGKILL (9)`: 강제 종료
 - `SIGSTOP (19)`: 일시 중지
 - `SIGCONT (18)`: 중지된 프로세스 재개
 - 예시: `kill -9 12345` (PID 12345 프로세스를 강제 종료)

리눅스 쉘 사용법 이해 - 하드링크와 소프트(심볼릭)링크

하드링크와 소프트링크

- cp 명령: 파일 복사
 - 하위 폴더 포함 복사: `cp -rf src/ dst/`
- 하드링크: `ln A B`
 - 생성: `ln source_file link_name`
 - A가 변경되면 B도 변경
- 소프트(심볼릭)링크: `ln -s A B`
 - 생성: `ln -s source_file link_name`
 - Windows OS의 바로가기와 동일
 - `ls -al` 명령어로 링크 확인
 - 하드링크: 동일한 inode 번호
 - 소프트링크: 다른 inode 번호, `->` 로 원본 파일 표시

```
lrwxrwxrwx 1 user user 20 Oct 10 12:00 symlink.txt -> original.txt
```

리눅스 쉘 사용법 이해 - 우분투 패키지 매니저

ubuntu 패키지 관리 실무

- 패키지 인덱스 정보 업데이트
 - `sudo apt-get update`
- 패키지 업그레이드 📦 패키지 업그레이드는 완전 주의 필요 ⚠
 - `sudo apt-get upgrade`
- 패키지 설치
 - `sudo apt-get install <package_name>`
- 패키지 제거
 - `sudo apt-get remove <package_name>`
- 패키지 제거(설정 파일 포함)
 - `sudo apt-get --purge remove <package_name>`

리눅스 쉘 사용법 이해 - VIM 사용법

초간단 사용법

- 입력 : `i` (insert 모드)
- 저장 : `w` (write)
- 종료 : `q` (quit)
- 강제 종료 : `q!` (quit without saving)
- 저장 후 종료 : `wq` or `x`

Section6. 리눅스, 맥, 윈도우에서의 도커 환경 구축

Mac / Windows Docker 설치

Mac 설치

- Docker for mac 검색 후 설치 - 간단

Windows 설치(강의 Section6-16 참조)

Windows 설치 는 추천하지 않음

- Dos 및 Hiper-v 기능 활성화 필요
- Windows 10 Pro 이상에서만 가능
 - Docker for Windows 검색 후 설치 - 간단

Linux Docker 설치

설치 방법

공식 페이지 참조 : <https://docs.docker.com/engine/install/ubuntu/>

1. old version 삭제
2. package 설치
3. 설치 완료되면 아래 명령어로 확인

```
sudo docker run hello-world
```

sudo 없이 사용하기

1. sudo 없이 사용하려면 아래 명령어로 현재 사용자에게 docker 그룹 권한 부여

```
sudo usermod -aG docker $USER
```

명령어 입력 후 터미널 재시작 필요

2. `id -nG` 명령어로 현재 사용자가 속한 그룹 확인 가능

Docker compose 설치(Standalone)

설치방법 : <https://docs.docker.com/compose/install/standalone/>

1. 다운로드
 - `curl -SL https://github.com/docker/compose/releases/download/v2.39.4/docker-compose-linux-x86_64 -o /usr/local/bin/docker-compose`
 - failed가 뜬다면 명령어 앞에 `sudo` 붙여서 실행
 - `sudo curl -SL ...`
2. 실행 권한 변경

```
sudo chmod +x /usr/local/bin/docker-compose
```

Section7. docker 주요 명령 익히기

도커에 대한 기본 이해

1. docker는 서버/클라이언트 구조

- docker daemon process(서버): 도커 엔진, 백그라운드에서 실행
- docker command(클라이언트): 사용자가 명령어 입력

2. docker image

- docker 컨테이너를 실행하기 위한 명령들을 가진 **템플릿**
- 여러 이미지들을 layer로 쌓아서, 원하는 형태의 이미지를 만드는 것이 일반적인

3. docker container

- docker image가 리눅스 컨테이너를 실행하기 위한 **인스턴스**
- docker image에 포함된 명령을 실행하여, docker container를 생성
- 컨테이너는 격리된 환경에서 실행되며, 독립적으로 동작
- 컨테이너는 일시적이며, 필요에 따라 생성

docker 이미지 기본

1. 주요 단계

- docker 설치
- docker image 다운로드
 1. docker hub 가입 : <https://hub.docker.com/>
 2. `docker login` : 도커 허브 로그인
 3. `docker search --limit=5 이미지이름` : 이미지 검색
 4. OFFICIAL [OK] : 공식 이미지
 5. `ubuntu/squid` : / 앞에는 사용자명, 뒤에는 이미지명
 6. 특정 이미지의 태그 리스트 확인은 CLI상으로는 불가능
 - 웹사이트에서 확인 가능
 7. `docker pull <이미지명>:<태그명>` : 도커 이미지 다운로드
 8. `docker images` : 다운로드 받은 도커 이미지 목록 확인
 9. `docker image ls -q` : 이미지 ID만 출력
- 다운로드 받은 image로 docker container 생성 및 실행

2. 명령어 주요 형태

- `docker 명령 옵션 선택자(이미지ID/컨테이너등)`

docker 주요 명령어 익히기

- `docker run` : 도커 컨테이너 생성 및 실행
- `docker ps` : 실행 중인 컨테이너 목록 확인
- `docker stop` : 실행 중인 컨테이너 중지
- `docker rm` : 중지된 컨테이너 삭제
- `docker rmi` : 도커 이미지 삭제

Docker Container 관련 주요 명령

1. 도커 컨테이너 생성

- `docker create <이미지명>:<태그명>`
- `docker create ubuntu`
- `docker create --name my_ubuntu ubuntu` : 컨테이너 이름 지정

2. 생성된 컨테이너 확인

- `docker ps -a`
- `docker ps` : 실행 중인 컨테이너만 확인
- `docker ps -a -q` : 컨테이너 ID만 출력

3. 컨테이너 실행

- `docker start <컨테이너ID or 이름>`

4. docker run 명령 : 컨테이너 생성 후 실행

- `docker run <옵션> <이미지명>:<태그명>`
- **주요 옵션**

옵션	설명
<code>-i</code>	interactive, 표준 입력 유지
<code>-t</code>	tty, 가상 터미널 할당
<code>-it</code>	interactive + tty, 터미널 접속

옵션	설명
<code>--name</code>	컨테이너 이름 지정
<code>-d</code>	detached, 백그라운드 실행
<code>--rm</code>	컨테이너 종료 시 자동 삭제
<code>-p</code>	호스트와 컨테이너 포트를 연결하는 옵션
<code>-v</code>	호스트와 컨테이너 간 디렉토리/파일을 연결하는 옵션

▪ pseudo tty? tty(teletypewriter): 리눅스(Unix)에서 터미널을 의미

◦ 예시

- `docker run -it --name my_ubuntu ubuntu`
- `docker run -it --rm ubuntu` : 종료 시 컨테이너 자동 삭제
- `docker run -it -d --name my_ubuntu ubuntu` : 백그라운드 실행

5. 컨테이너 접속

- `docker exec -it <컨테이너ID or 이름> /bin/bash`
- `docker attach <컨테이너ID or 이름>` : 실행 중인 컨테이너에 연결

6. 컨테이너 종료

- 컨테이너 종료 하기

```
docker stop <컨테이너ID or 이름>
```

(참고)실행 중인 컨테이너 잠깐 멈추기

- `docker pause <컨테이너ID or 이름>`
- 잘 사용되지 않음

- 종료된 컨테이너 다시 실행

```
docker start <컨테이너ID or 이름>
```

웹서버로 docker run 옵션 테스트해보기

- 웹서버는 크게 두가지 프로그램으로 구성
 - 웹서버 프로그램 : apache, nginx
- apache 웹서버 도커 이미지 사용
 - 이미지명 : `httpd`
 - 태그명 : `latest` (최신버전)

apache 웹서버 도커 이미지 다운로드

```
docker search httpd:latest --limit=5
```

이미지 다운로드받고 바로 컨테이너로 만들어 실행시키기(`-p` 옵션 이해하기)

1. 실행

- `docker run -dit --name {컨테이너이름} {이미지명}`

```
docker run -dit --name appacheweb httpd
```

2. Private IP 변환(NAPT Network Address Port Translation 기술)

- `-p` 옵션 사용

- `docker run -dit -p 9999:80 --name appacheweb httpd`
- 9999 : 호스트 포트, 80 : 컨테이너 포트

localhost:9999 으로 접속 시 컨테이너의 80포트로 접속됨

나만의 웹서비스 docker 만들기(-v 옵션 이해하기)

- FTP(파일 전송 프로토콜) : FileZilla 등으로 EC2 서버에 HTML 파일 업로드
- 호스트와 컨테이너 간 디렉토리/파일을 연결하는 옵션

```
docker run -dit -p 80:80 -v /home/ubuntu/index_html_test:/usr/local/apache2/htdocs --name my_apache_web httpd
```

alpine 리눅스 기반 경량 웹서버 도커 이미지 사용

- 태그명 : alpine (경량 버전)

```
docker run -dit -p 80:80 -v /home/ubuntu/index_html_test:/usr/local/apache2/htdocs --name my_apache_web httpd:alpine
```

도커가 사용 중인 디스크 용량 확인

```
docker system df
```

TYPE	TOTAL	ACTIVE	SIZE	RECLAIMABLE
Images	1	1	64.63MB	0B (0%)
Containers	1	1	2B	0B (0%)
Local Volumes	0	0	0B	0B
Build Cache	0	0	0B	0B

실행중인 컨테이너 리소스 사용량 확인

```
docker stats
```

ctrl + c 로 종료

실행중인 컨테이너에 명령 실행하기

- 사용되는 이미지의 터미널 접속 : /bin/bash (일반 apache) or /bin/sh (alpine apache)

```
docker exec -it my_apache_web /bin/sh
```

모든 컨테이너 삭제

```
docker stop $(docker ps -a -q)
docker rm $(docker ps -a -q)
docker rmi $(docker images -q)
```

Section8. Dockerfile 사용법 기본

Dockerfile 주요 명령어

명령어	설명
FROM	베이스 이미지 지정(예: FROM ubuntu:latest)
LABEL	이미지에 메타데이터 추가(inspect 에 포함되는 정보)
CMD	컨테이너 실행 시 기본으로 실행될 명령어 지정
RUN	이미지 빌드 시 실행할 명령어
ENTRYPOINT	컨테이너 시작 시 실행될 명령어 지정
EXPOSE	컨테이너가 수신할 포트 지정
ENV	환경 변수 설정
WORKDIR	작업 디렉토리 설정
COPY	파일/디렉토리를 이미지에 복사 (도커파일이 위치한 상대경로로 작성)

Dockerfile로 이미지 작성

- Dockerfile: 도커 이미지를 자동으로 빌드하기 위한 설정 파일
- Dockerfile 작성 후, docker build 명령어로 이미지 생성

```
docker build -t <이미지명>:<태그명> <Dockerfile이 있는 경로>
# 예시
docker build -t my_apache_image:1.0 ./
```

Dockerfile 예시

```
# 문서에 대해 설명을 하기 위한
LABEL maintainer="star2kis@nate.com"
# Base 이미지 지정
FROM httpd:alpine

# 문서에 대해 설명을 하기 위한
LABEL maintainer="star2kis@nate.com"
LABEL version="1.0.0"
LABEL description="A test docker image to understand Docker"

# 상대경로로 작성
COPY ./index_html_test /usr/local/apache2/htdocs

# CMD
CMD ["/bin/sh", "-c", "httpd-foreground"]

# ENTRYPOINT
ENTRYPOINT ["/usr/local/bin/httpd-foreground"]
```

주요 옵션

옵션	설명
-t	이미지 이름과 태그 지정
-f	Dockerfile 경로 지정(지정하지 않으면 ./Dockerfile 참조)
--pull=true	이미지 생성 시 마다 새로 다운로드

Docker Image 조사(docker inspect)

```
docker inspect <이미지ID or 이름>
```

Docker 가끔 사용하는 기타 명령어들

- `docker logs <컨테이너ID or 이름>` : 컨테이너 로그 확인
- `docker kill <컨테이너ID or 이름>` : 실행 중인 컨테이너 강제 종료

ENTRYPOINT

- 컨테이너 시작 시 실행될 명령어를 지정
- `CMD` 와 함께 사용되며, `CMD` 는 기본 인수로 전달됨
- 예시: `ENTRYPOINT ["usr/local/bin/httpd-foreground"]`

EXPOSE

- 컨테이너가 수신할 포트를 지정
- 예시: `EXPOSE 80`

```
"ExposedPorts": {"80/tcp": {}},
```

ENV : 환경 변수 설정

- 컨테이너 내에서 사용할 환경 변수를 설정

```
FROM mysql:latest

ENV MYSQL_ROOT_PASSWORD=password
ENV MYSQL_DATABASE=dbname
```

WORKDIR : 작업 디렉토리 설정

- 컨테이너 내에서 작업할 디렉토리를 설정
- 예시: `WORKDIR /app`

Docker DB 설정 예시

1. Dockerfile 예시

```
FROM mysql:latest

ENV MYSQL_ROOT_PASSWORD=password
ENV MYSQL_DATABASE=dbname
```

2. 이미지 빌드

```
docker build -t my_mysql_image:1.0 -f Dockerfile-MYSQL ./
```

3. 컨테이너 실행

```
docker run -dit -p 3306:3306 --name my_mysql_container my_mysql_image:1.0
```

4. MySQL 접속

```
docker exec -it my_mysql_container /bin/bash
mysql -u root -p
```

Output

```
Enter password: password
```

5. 데이터베이스 확인

```
SHOW DATABASES;
```