강의 : <https://www.youtube.com/watch?v=3xPVG0PmJXk&list=PL93mKxaRDidGMzIllhYKx1d6aMg6_5wW3&index=4>

“메타코딩” 유투브

맥은 상관 없고 !!

윈도우는 버전 10이상에 home 버전은 안되고, pro 버전 이상이어야 설치가되고 실행 됨 !!

docker compose 설치

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

powerShell 관리자 모드로

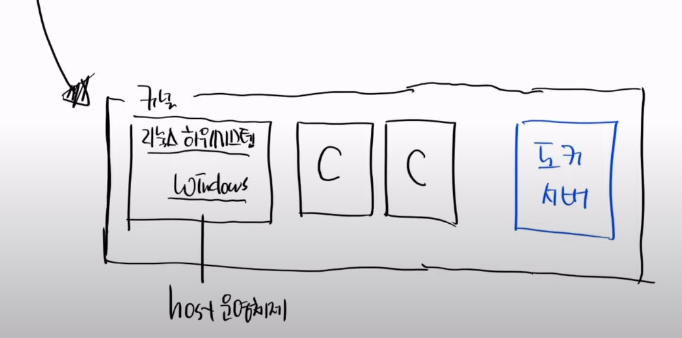
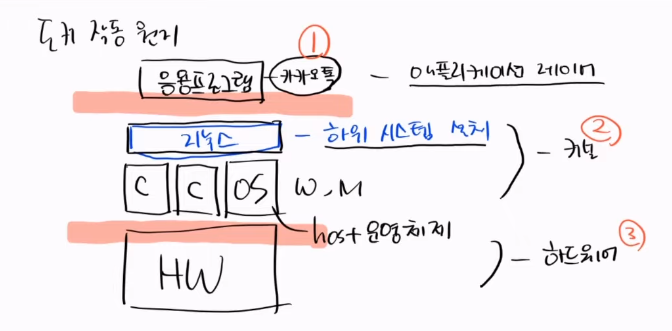
Start-BitsTransfer -Source "https://github.com/docker/compose/releases/download/v2.26.1/docker-compose-windows-x86\_64.exe" -Destination $Env:ProgramFiles\Docker\docker-compose.exe

입력

######### 윈도우, 맥은 다운로드 받아서 설치하면 자동으로 docker-compose 설치 됨

도커 작동 원리

* 하나의 컴퓨터에 설치된 리눅스 운영체제에서 완전히 분리된 별도의 리눅스 시스템을 운영할 수 있는 기술
* CPU, 메모리, 네트워크를 분리하여 마치 별도의 시스템처럼 사용할 수 있는 기술

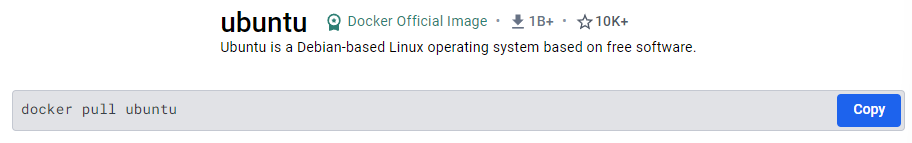


image란? 환경 구성 실행 전 File? 형태의 상태

container란? Image를 실행한 상태

docker 명령어

docker pull [dockerHub에서 다운받을 이름 명] : Imgae Down

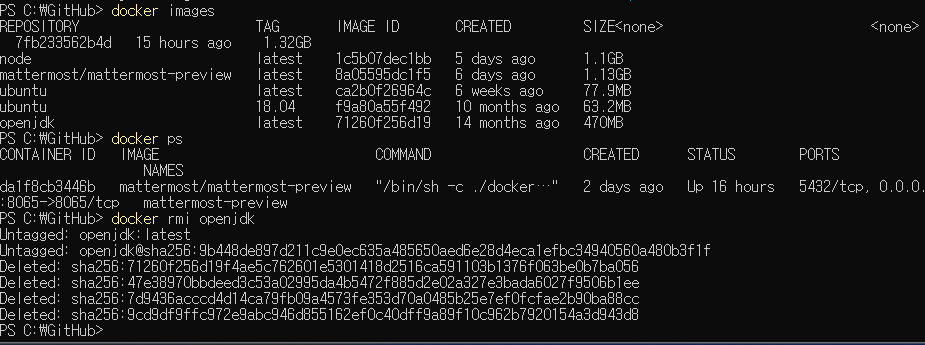


docker ps : container 동작 list display

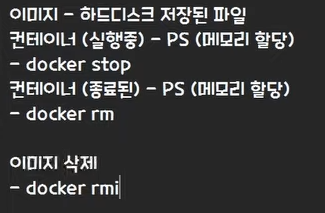
docker ps -a : stop된 컨테이너까지 모두 디스플레이

docker images : docer에 다운받아져 있는 image 들

docker rmi [삭제할 이미지 명] : 이미지 삭제

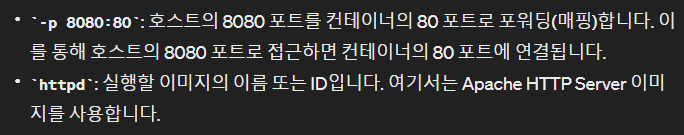


둘 다 기재 가능

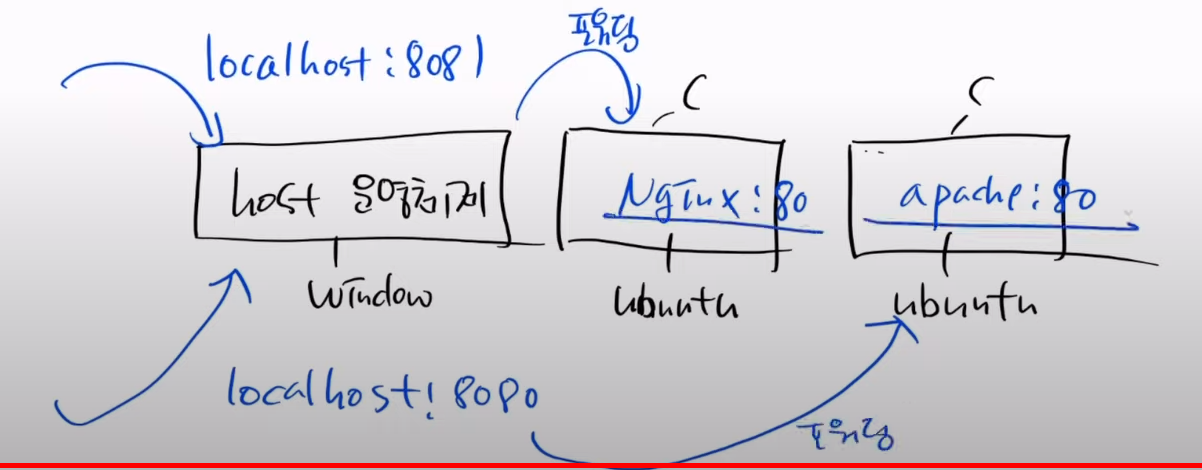


docker run [image이름] : Image 실행

docker run -d [image이름] : 백그라운드로 image 실행

docker run -d -p 8080:80 [image이름] : docker container 내부의 8080포트로 접속  


run 은 Pull과 실행을 동시에 할 수 있다.



도커 이미지 완전삭제

실행중인 프로세스 종료

1. docker ps
2. docker stop 컨테이너id

종료된 컨테이너 삭제

1. docker ps -a
2. docker rm 컨테이너id

이미지 삭제

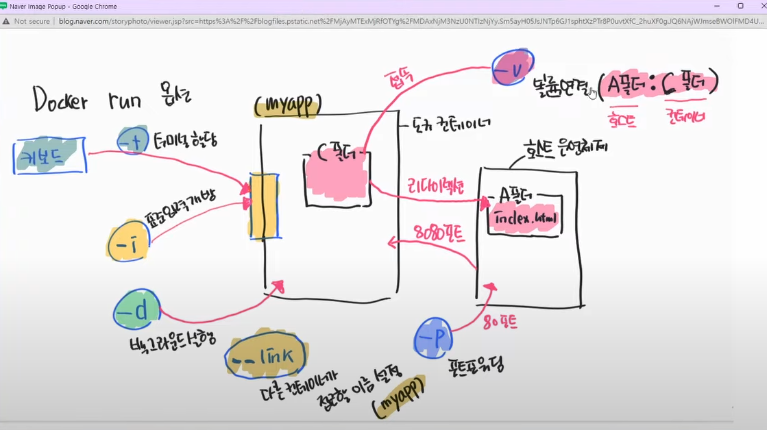
1. docker images
2. docker rmi 이미지이름

docker 이미지 한번에 삭제하는 법(git bash 사용)

docker stop $(docker ps -q)

docker rm $(docker ps -a -q)

docker rmi $(docker images -q)

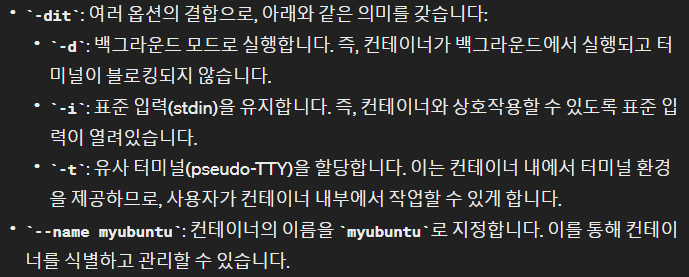


우분투 환경 실행

docker run -dit --name myubuntu ubuntu

-dit 란?

--name 이란?



docker attach [ContainerID] : 로 Ubuntu 환경에 접속할 수 있다.

나오는 법 : Ctrl+P누른다음 Ctrl+Q

컨테이너 종료 : Ctrl+C

Docker exec 명령어

 10강 - docker exec 명령어 변경

24.04.15

실행중인 도커에서 Command 입력하려면 exec 써야함 !!!

**docker run –dit ubuntu bash**

**docker attach [containerID]**

설명 : ubuntu는 OS 이므로 –d(백그라운드) –it(입출력, 대화형모드) 뭐로 ?? bash로 실행 시킴

run –d ubuntu로 하면 OS이기 때문에 그냥 종료됨

httpd는 서버이므로 while 문으로 돌아서 ps로 확인 시 계속 실행중임

**docker run –d –p 8080:80 httpd**

설명 : httpd 프로그램이 설치될 때는 OS랑 프로그램이 같이 설치 되기 때문에, 이를 제어하려면

**docker exec –it [containerID] bash**

로 제어해야 한다.

정리 :

1. OS (Ubuntu)

**docker run –dit Ubuntu bash**

2. while Process (httpd)

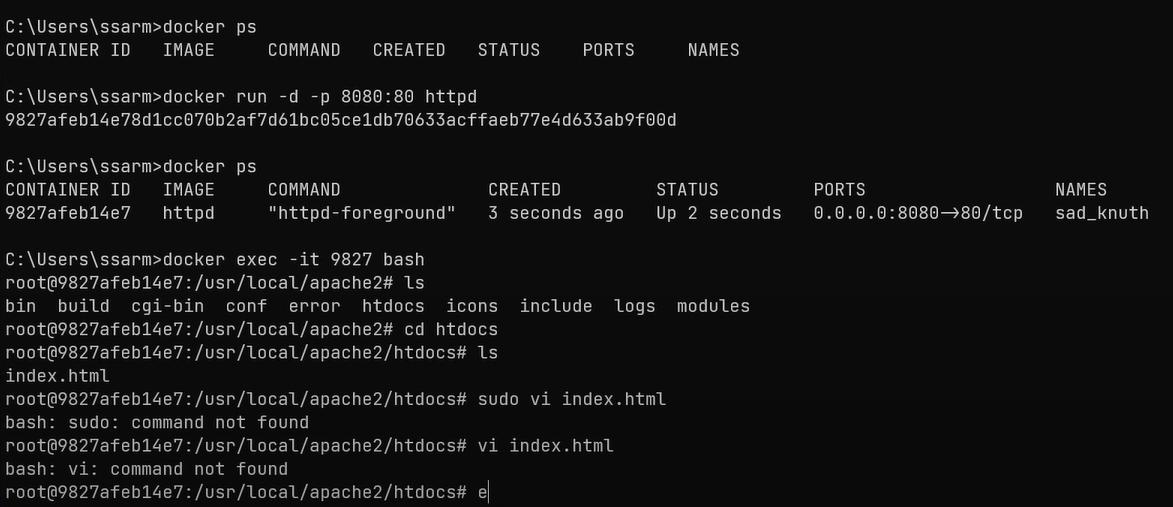
**docker run –d –p 8080:80 httpd**

**docker exec –it [컨테이너id] bash**

승환 Cmd : 그냥 둘 다 docker run –dit (-p) ~~~ bash 로 실행하고, attach [컨테이너id]로 접속해도 되는디???

11강 docker volumn : 하이퍼링크 연결 같은 느낌??

docker run -dit -p 8080:80 httpd

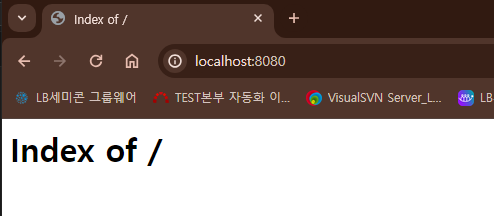


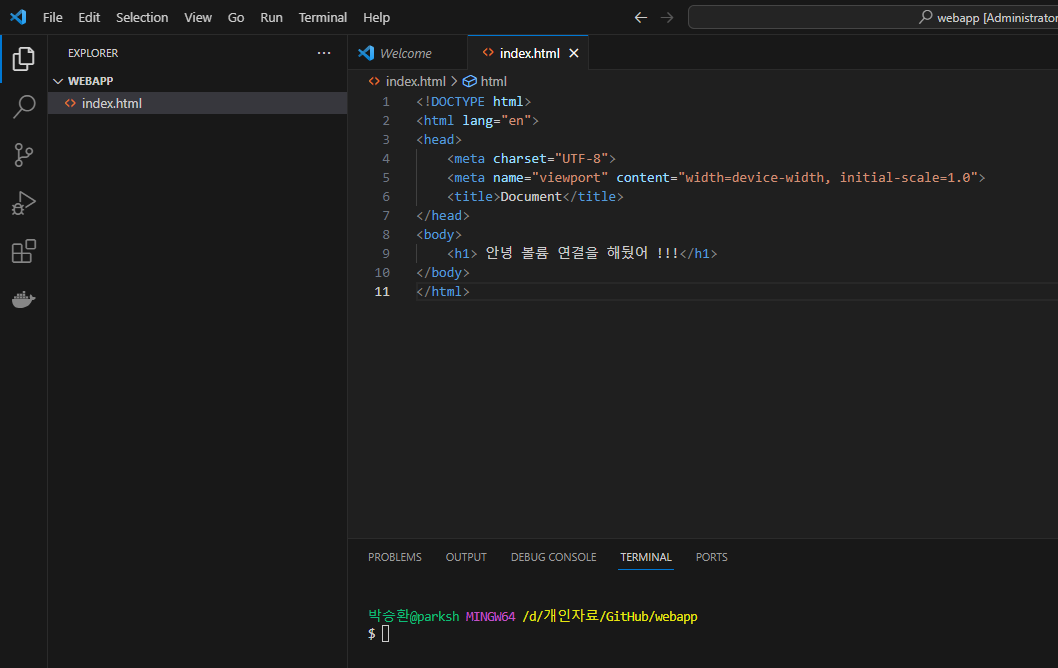
index.html 파일 수정시 “sudu”, “vi” 커맨드가 모두 안 먹힘, 이 커맨드를 쓰려면 추가로 더 설치해줘야 하는데!!

docker run h -dit -p 8080:80 -v D:\개인자료\GitHub\webapp:/usr/local/apache2/htdocs httpd

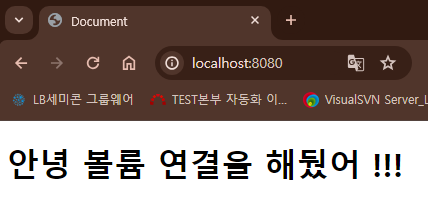


빨간색이랑 파란색으로 연동 해주면 아무것도 없어서 아래처럼 뜸



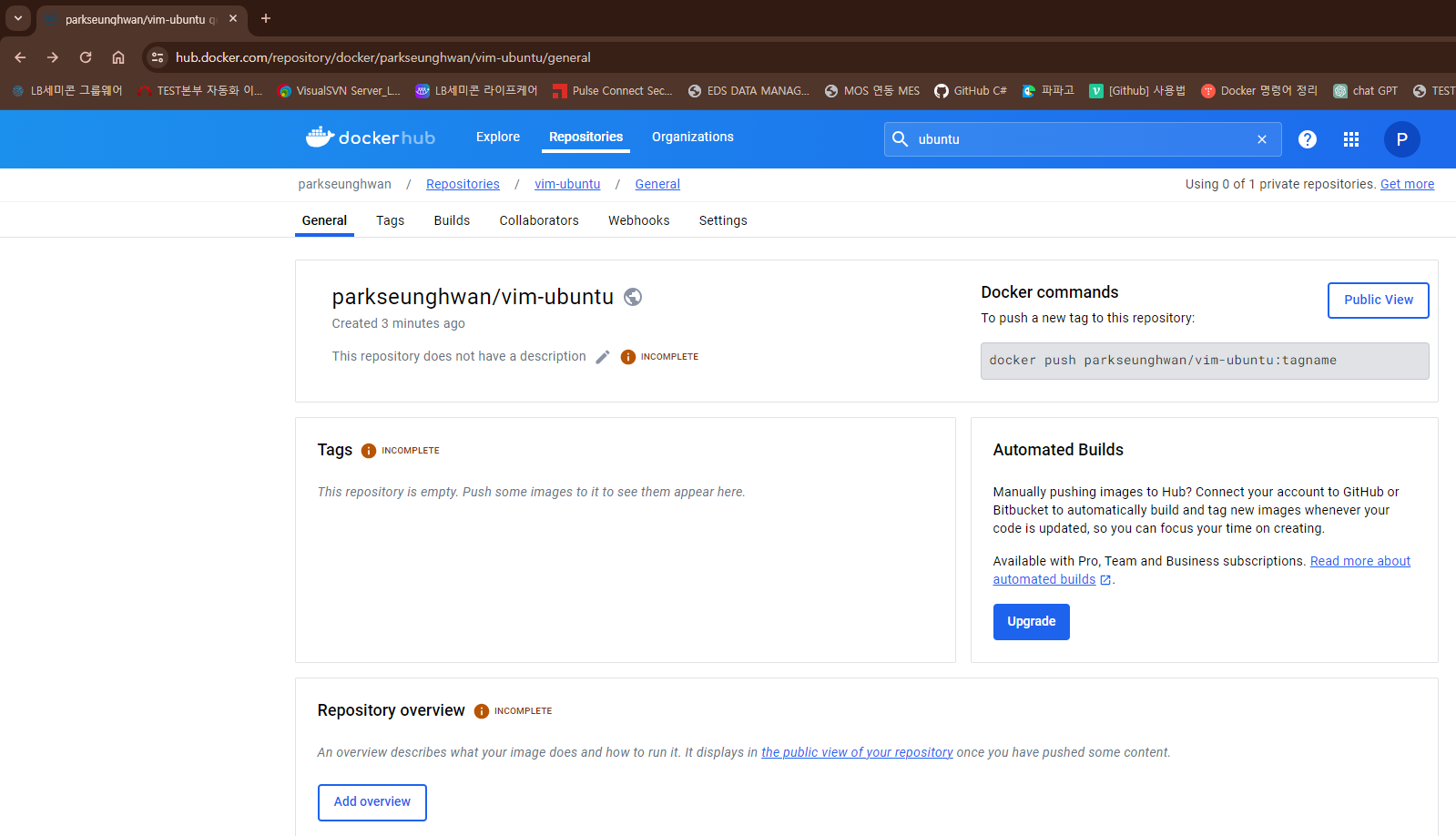


위 경로에 파일 생성시 자동으로 연동 됨



12강 docker commit 이미지 굽기와 docker hub 업로드

docker hub에서 repositories 생성



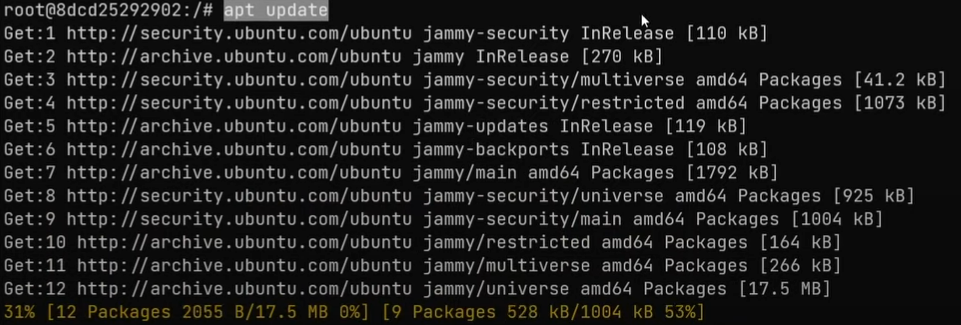
Ubuntu 다운 후 Ubuntu 실행

docker run –dit Ubuntu

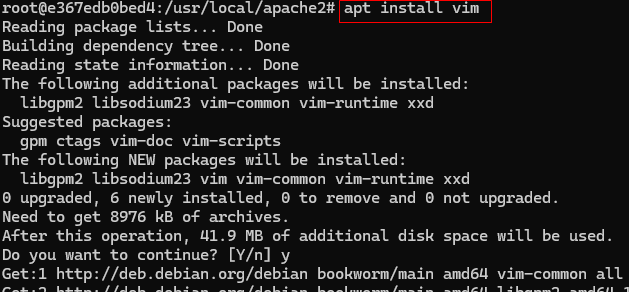
docker attach [컨테이너ID]

후 Ubuntu에서 “vi” 같은 Command를 사용하기 위해

apt update 진행(매뉴판 설치)

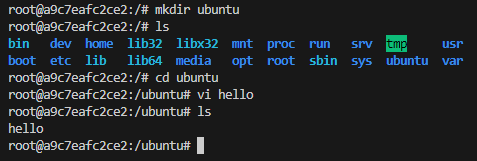


apt install vim 설치 진행



제일 처음 경로 home 폴더에 접속

mkdir ubuntu로 폴더 생성 후 접속



vi hello로 파일 생성

“I” 쳐서 inset 모드로 바꾸고

“Hellow World” 작성 후 Esc 클릭

::wq+Enter로 빠져나옴

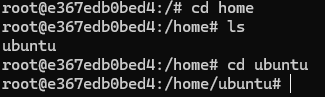
exit로 Ubuntu 빠져나옴!!

다시 Start로 실행 후

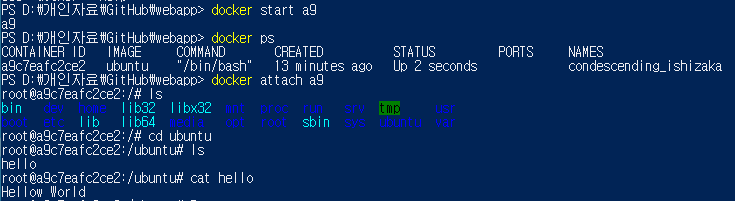
docker start a9

docker attach a9 로 접속

cd ubuntu로 접속 후



cat hello로 파일 실행

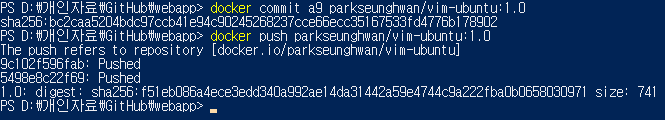


실행된채로 나오는법 : ctrl+P, ctrl+Q

커밋하는법 !!!!

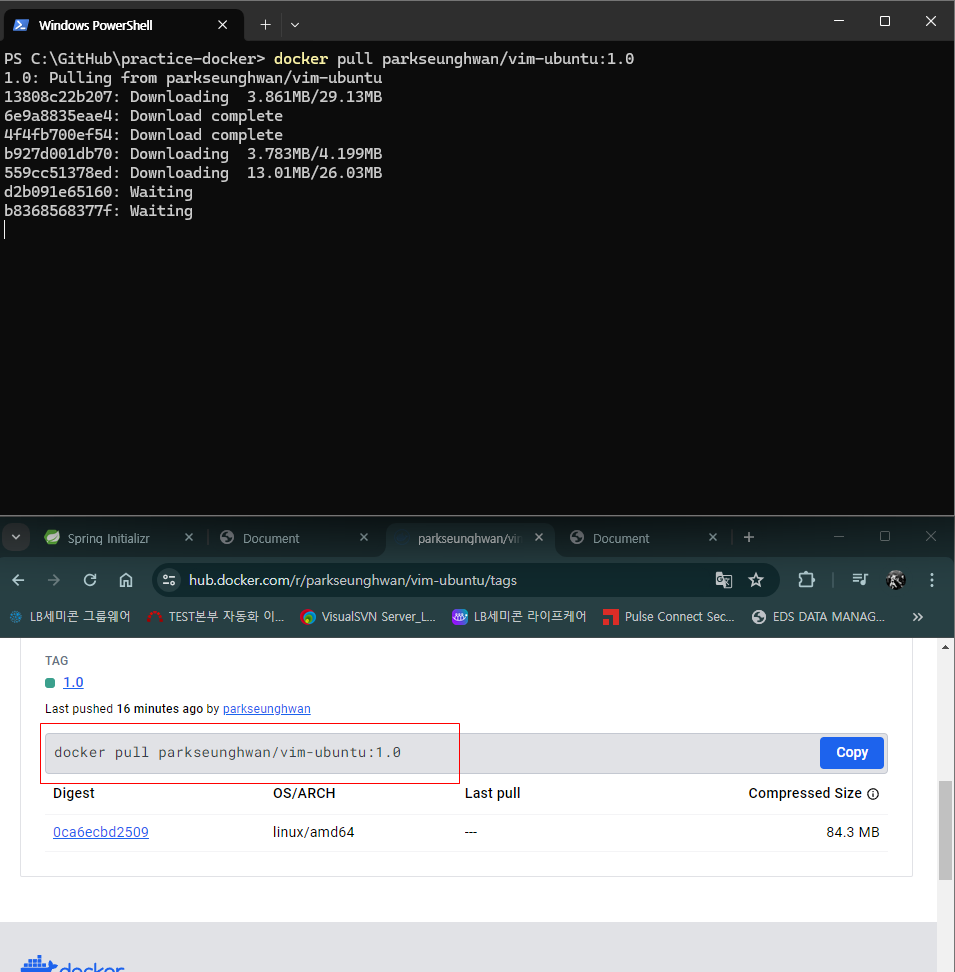
docker commit a9 parkseunghwan/vim-ubuntu:1.0

docker push parkseunghwan/vim-ubuntu:1.0

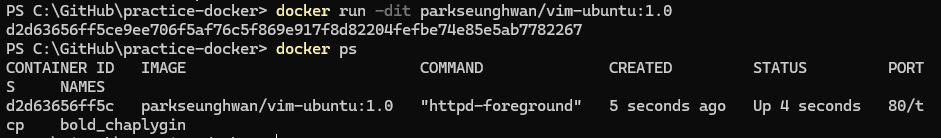


내려받는 법 !!!!!

붙여넣기



run 진행 : docker run -dit parkseunghwan/vim-ubuntu:1.0



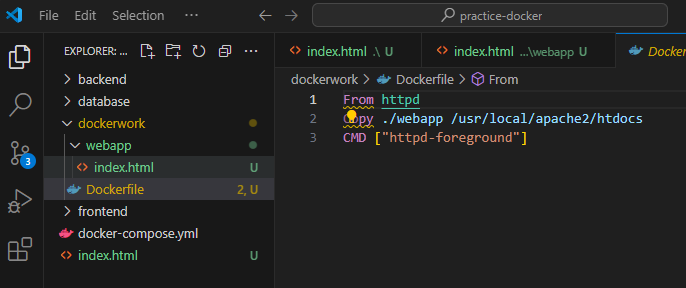
13강. docker file step 1-기본기

현재 docker 실행 Container 및 image 다 지우기

mkdir dockerwork 파일 만들기



VScode로 파일 만들기 ( 파일이름은 Dockerfile로 확장명 없이 저장한다.)



설명 :

httpd 이미지를 구워서(가져와서)

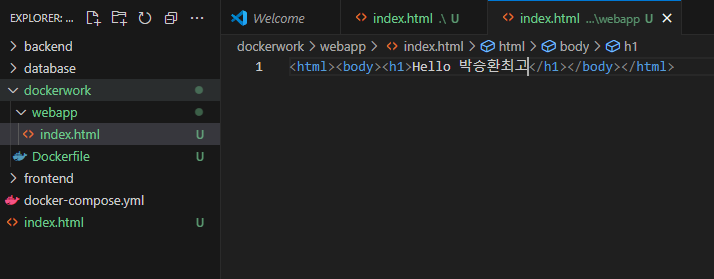
./webapp 이라는 형제폴더에 httpd의 이미지 내부 /usr/local/apache2/htdocs폴더를 복사한다

첫 커맨드는 ["httpd-foreground"] 이다.

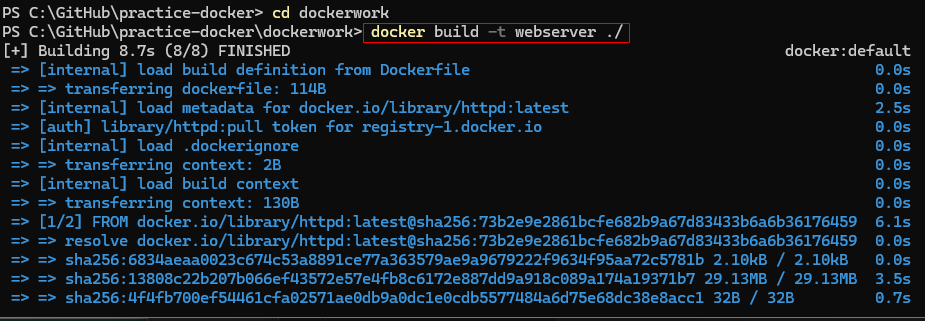
mkdir webapp 폴더 생성 후 폴더에 접속하여



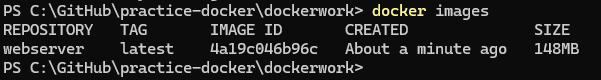
index.html 생성



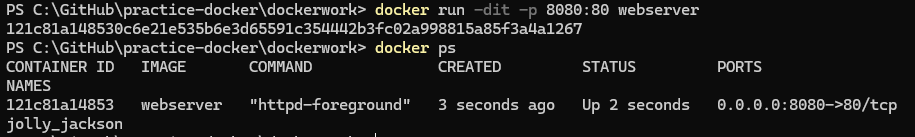
상위폴더에서 docker build -t webserver ./ 입력하여 빌드 진행하면 이미지 생김 !!!



생성 이미지 확인



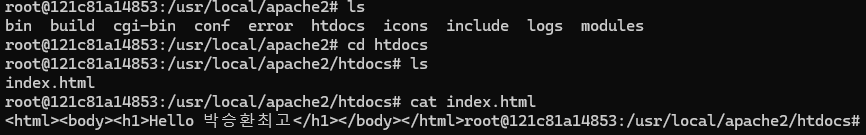
실행



접속



파일 내용 확인



13강 끝

옵션 :

1. –d : 백그라운드 실행
2. –p : 호스트의 8080포트를 컨테이너의 80포트로 포워딩(매핑), 이미지를 통해 호소트의 8080포트로 접근하면 컨테이너의 80포트에 연결 됨
3. –i : 컨테이너와 상호작용 할 수 있도록 표준입력(stdin)을 연결한다. 이는 컨테이너 내에서 명령어를 입력하고 출력을 확인하는데 유용함
4. –t : 컨테이너 내에서 터미널 환경을 시뮬레이트 하도록 표준 출력(stdout)을 연결합니다. 이는 컨테이너 내에서 명령어를 실행한 결과를 터미널 처럼 보기좋게 표시하는데 사용된다.
5. attach : 컨테이너 내부의 터미널에 연결하여 작업할 수 있음
   * ubuntu 같은 OS에 접근할때는 attach
   * 대화형(interactive)모드로 실행중인 컨테이너에 연결합니다.
   * 컨테이너의 표준입력(stdin)과 표준출력(stdout)을 연결하여 컨테이너 내부의 터미널에 직접 접근할 수 있도록 합니다.
   * 주로 컨테이너 내부에서 프로세스를 모니터링하거나 디버깅할 때 사용됩니다.
   * 컨테이너가 백그라운드 모드로 실행중일때는 사용할 수 없습니다.
6. exec
   * OS내 프로그램에 접근할때는 exec
   * 새로운 프로세스를 싱행하기위해 컨테이너에 연결
   * –it 옵션을 사용하여 인터렉티브 모드로 컨테이너 내부의 쉘을 실행시김
   * 주로 컨테이너 내부에서 명령어를 실행하고 결과를 확인할 때사용
   * 컨테이너가 백그라운드로 실행중이더라고 사용 가능
7. Docker 시스템 정보 확인:

* docker version : Docker 버전 및 상세 정보 확인
* docker –v : Docker 버전 및 빌드 버전을 한 줄로 간단하게 확인
* docker system info : Docker의 시스템 전체 정보 확인
* docker system event : 서버로부터 실시간 이벤트를 받습니다.
* docker system df : 도커 디스크 사용량 확인

1. Docker 이미지 관련 명령어:

* 이미지 리스트 출력 : docker images 또는 docker image ls
* 이미지 상세 정보 표시 : docker image inspect [이미지명]
* Docker Hub에서 이미지 검색 : docker search [옵션] <검색어>

1. Docker 컨테이너 관련 명령어:

* 현재 실행 중인 컨테이너 확인 : docker ps
* 실행 중인 컨테이너 리스트 확인 : docker ps -a
* 컨테이너 삭제:
* 단일 컨테이너 삭제 : docker rm [컨테이너 ID]
* 다중 컨테이너 삭제 : docker rm [컨테이너 ID1] [컨테이너 ID2]
* 모든 컨테이너 삭제 : docker rm $(docker ps -a -q)

1. 컨테이너 접속:

* 실행 중인 컨테이너에 접속 (attach): docker attach [컨테이너명 또는 ID]
* 실행 중인 컨테이너에 접속하여 명령 수행 (exec): docker exec -it [컨테이너명 또는 ID] /bin/bash

1. 컨테이너 시작, 중지, 재시작:

* 컨테이너 시작: docker start [컨테이너명 또는 ID]
* 컨테이너 중지: docker stop [컨테이너명 또는 ID]
* 컨테이너 재시작: docker restart [컨테이너명 또는 ID]

1. Docker Run:

* 컨테이너화: docker run [options] image[:tag, @digest] [command] [args]
* 실전 예제:
* docker run -it -d -p 8080:8080 --name testMysql test123/mysql
* -it: 표준 입력 (STDIN) 허용 및 터미널 연결 가능 설정
* -d: 컨테이너를 백그라운드에서 실행
* -p: 포트 포워딩 설정 (예: 8080:3000)

1. 컨테이너 이름 변경:

* docker rename [현재 컨테이너명] <변경할 컨테이너 이름>