**DEV #3 – 2020.04.09**

**>> alpine linux를 docker container로 실행하되 dettach mode(백그라운드)로 실행하세요**

🡺 docker run –d --name alpine alpine –p 1621:1521 tail –f //dev//null

왜 alpine 이미지를 사용했나?

🡪 가벼워서 (centos를 경량화 시킨 버전 / 최소 옵션만 적용)

왜 tail –f //dev//null 을 해주나?

🡪 컨테이너를 계속 실행시키기 위해서 ( dev가 null일 때 까지 돌아라)

왜 // 해주나?

🡪 git bash는 경로표시인 /를 특수문자로 인식해서 // 해줘야 함.

**>> 위에서 실행중인 container의 쉘(sh)을 실행하고 터미널로 접속하세요.**

🡺 docker exec –it alpine sh

**>> 터미널 접속후 타임존 설치 후 변경(Asia/Seoul)하세요. (apk add tzdata)**

🡺 /# apk add tzdata

/# cp /usr/share/zoneinfo/Asia/Seoul /etc/localtime (폴더 복사)

/# echo “Asia/Seoul” > /etc/timezone (파일 생성)

/# apk del tzdata (tzdata 삭제)

/# date

**>> python을 apk로 설치(apk add python)하세요.**

🡺 /# apk add python

**>> “/www” 디렉토리를 만들고 /www 이동후 python SimpleHTTPServer 를 실행하세요.**

(python -m SimpleHTTPServer 8000) == (8000 port로 웹서버 구동)

🡺 /# mkdir /www

/# cd /www

/# ~~/www python –m SimpleHTTPServer 8000

(이 명령어 후 8000 포트로 웹 서버 열림 / 계속 돌아감)

\*\*\* 현재 디렉토리를 웹 서버로 기동시키기 위한 명령어 (node.js 도 비슷한 기능이 있음)

>> 위 과정으로 Dockerfile로 만드세요.

FROM alpine

RUN /bin/sh

RUN apk add tzdata && \

( && : RUN 명령어를 적게 쓰기 위해서 여러 RUN 내용을 이어 붙임)

( \ : RUN 명령어를 이어 붙여서 사용할 때 한 줄 띄어 쓰기 위함 )

cp /usr/share/zoneinfo/Asia/Seoul /etc/localtime && \

echo 'Asia/Seoul' > /etc/timezone && \

apk del tzdata

RUN apk add python

WORKDIR /www

ENTRYPOINT ["python", "-m", "SimpleHTTPSever", "8000"]

(ENTRYPOINT : 시작점 이게 없으면 CMD로 설정된 명령이 실행)

(만약 ENTRYPOINT, CMD 둘다 같이 있을 경우

: CMD명령은 ENTRYPOINT 옵션으로 들어가서 실행 됨 )

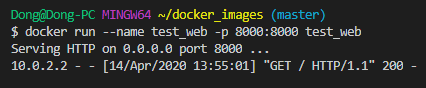
**>> Dockerfile로 image를 빌드하고 빌드된 이미지로 새로운 container를 실행하세요.**

**(정상 수행여부 확인)**

🡺 docker build –t test\_web .

( local에서 이미지 생성 작업 // 여기서 . 은 현재 디렉토리를 참조 한다는 의미 )

docker run --name test\_web –p 8000:8000 test\_web



**>> git 저장소를 새로만들고 (docker-images) 브랜치(simple-web)도 하나 만드세요.**

🡺 git clone [본인 github repository 주소]

🡺 git checkout –b simple\_web

**>> Dockerfile을 github에 push하세요 (simple-web branch).**

🡺 git push –u origin simple\_web

**>> docker hub (hub.docker.com)에 접속하여 새로운 저장소(Repository)를 만드세요**

**저장소 생성시 Build Settings 옵션에서 github docker-images저장소의**

**simple-web 브랜치와 연결(자동빌드설정)하여 생성하세요.**

**>> 자동빌드 여부 확인 후 생성된 Repository Image를 검색해보세요.**

**>> 본인의 docker hub에 등록된 이미지를 pull하고 container를 생성해 보세요.**

🡺 docker pull wjrdu369/docker\_images

🡺 docker run --name test\_web -p 8000:8000 wjrdu369/docker\_images

[컨테이너명] [이미지명]

**>> Dockerfile 수정후 github에 push할때 docker hub의 이미지가 자동빌드되는지 확인하세요.**

**컨테이너 접속 후**

/# **ls /usr/share/zoneinfo**

🡺 usr폴더의 share폴더의 zoneinfo 찾기

**echo '<h1>from hub success test</h1>' > index.html**

🡺 <h1>태그를 이용해서 index.html 파일을 만듬

**echo '<h2>yangdonguk</h2>' >> index.html**

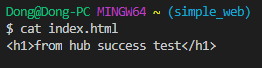
🡺 <h2>태그와 >> 를 이용해서 기존의 index.html 파일에 내용 추가

**docker cp index.html test\_web:/www**

🡺 index 파일을 test\_web 컨테이너의 /www 폴더에 복사

**docker exec test\_web ls //www**

🡺 test\_web 컨테이너를 실행해서 //www 파일에 있는 목록 출력

 : index 파일의 내용 확인