과제 1 – Docker Hub에 이미지 올려보기

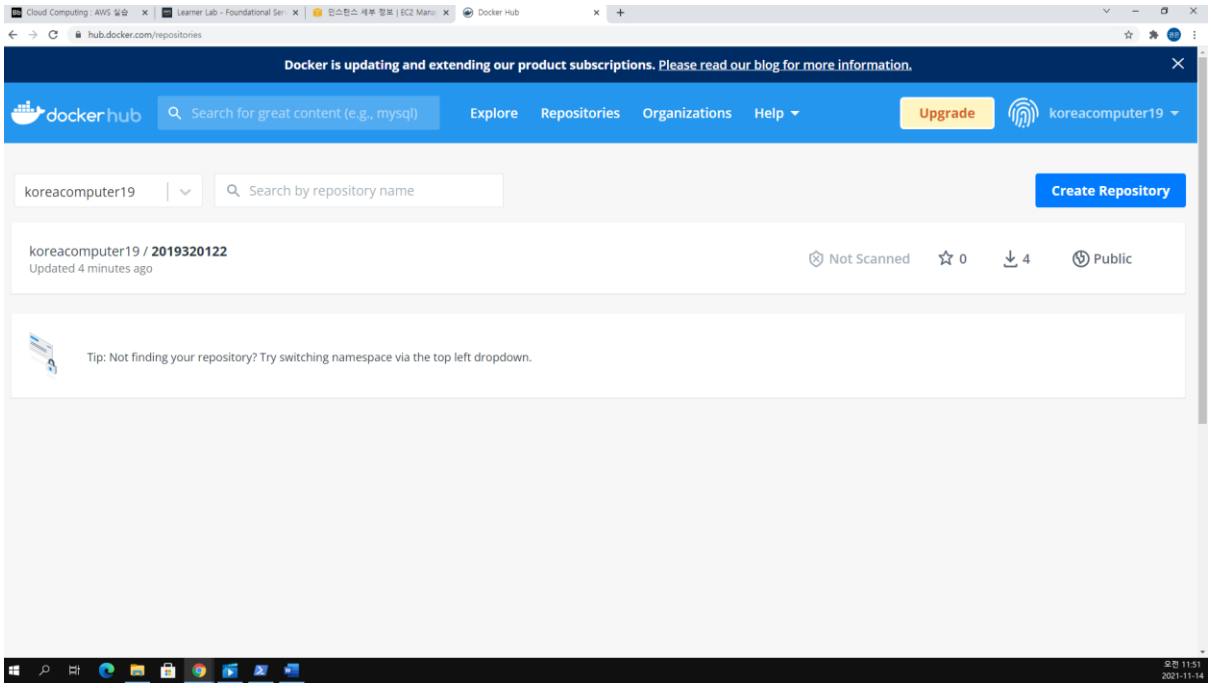
요구사항: 이미지 이름이 자신의 학번인 이미지 파일을 업로드

```
[ec2-user@ip-172-31-20-10 sample1]$ sudo docker build -t koreacomputer19/2019320122 .
Sending build context to Docker daemon  3.072kB
Step 1/3 : FROM golang:1.14
 ---> 21a5635903d6
Step 2/3 : WORKDIR /go/src/app
 ---> Using cache
 ---> ccd9e192981c
Step 3/3 : COPY . .
 ---> Using cache
 ---> 07dee5cc4121
Successfully built 07dee5cc4121
Successfully tagged koreacomputer19/2019320122:latest
```

우선 이름이 학번인 이미지를 빌드하였다. Dockerhub에 이미지를 업로드하기 위해서는 <username>/<imagename>이어야 하기 때문에 Docker Hub의 username인 koreacomputer19를 이미지명(학번) 앞에 넣어주었다. 앞서 실습시간에 hello:0.1을 빌드할 때 사용한 Dockerfile을 그대로 사용했기 때문에 빌드 시 cache를 사용한다는 메시지를 확인할 수 있다 (Using cache). 마지막 줄을 보면 명령어에 -t옵션 (tag 지정 옵션)을 사용했지만 태그를 별도로 지정하지 않았기 때문에 latest로 자동 지정된 것을 확인할 수 있다.

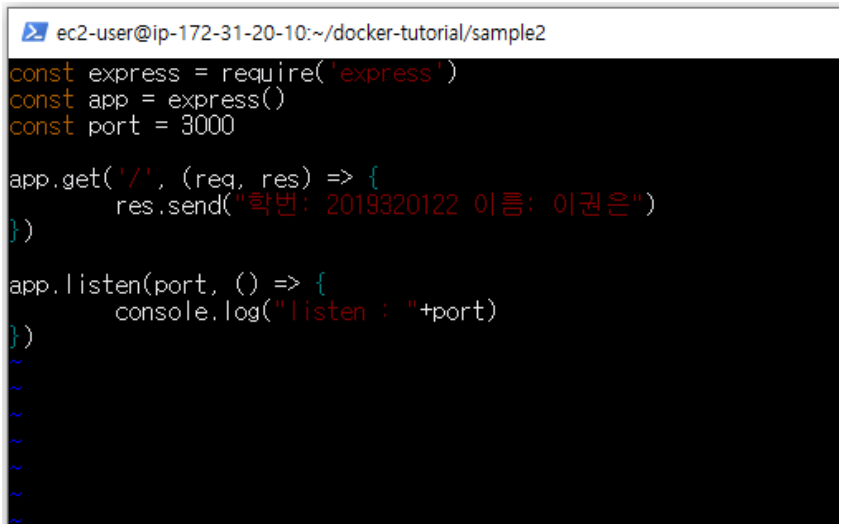
```
[ec2-user@ip-172-31-20-10 sample1]$ sudo docker push koreacomputer19/2019320122
Using default tag: latest
The push refers to repository [docker.io/koreacomputer19/2019320122]
c9b58360ff38: Pushed
a92b48d43efd: Pushed
660b50dfadeb: Pushed
3eeebffacaf0: Pushed
041459108fb0: Pushed
da654bc8bc80: Pushed
4ef81dc52d99: Pushed
909e93c71745: Pushed
7f03bfe4d6dc: Pushed
latest: digest: sha256:82fa59a187923b37a508ea38d48c3450e1bf481e36c4220b5595119c579d72c9 size: 2209
```

push 명령어를 이용하여 Dockerhub에 이미지를 업로드하였다.

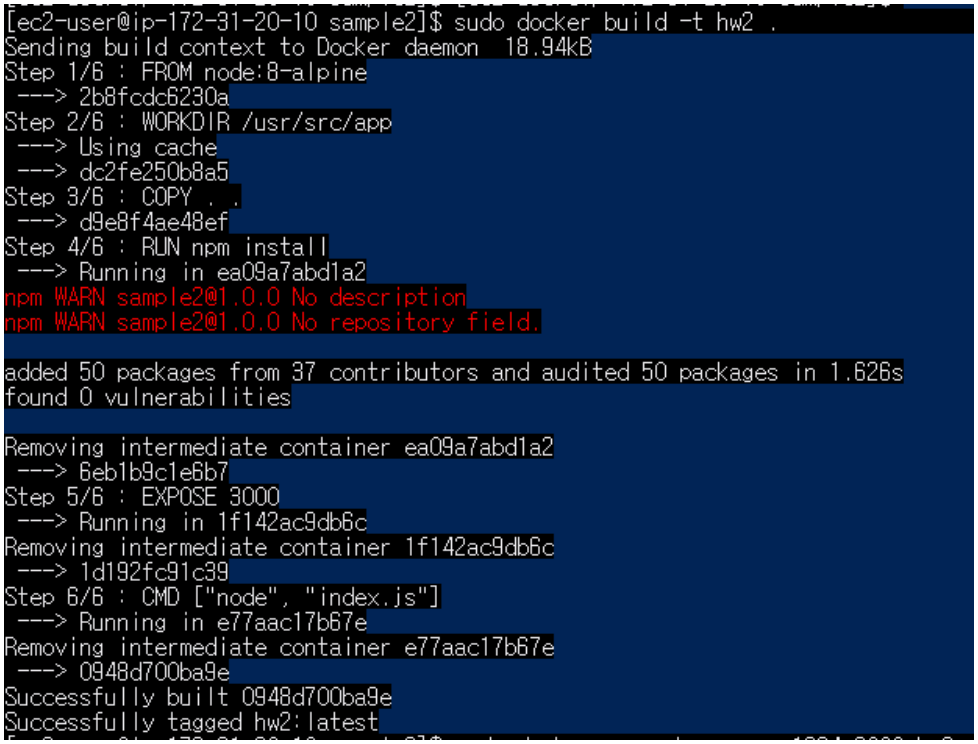


hub.docker.com에서 이미지가 성공적으로 업로드 된 것을 확인할 수 있다.

과제 2 – 자신의 이름과 학번을 출력하는 웹 서버를 컨테이너로 실행하고, 개인 노트북/PC에서 요청해서 결과 받기



실습시간에 사용했던 sample2 폴더의 파일들 (Dockerfile, index.js, package-lock.json, package.json)을 그대로 사용하였다. 단, 페이지에 출력하는 내용이 hello world가 아닌 학번과 이름이 되도록 res.send()부분을 위와 같이 수정하였다.



그 후 Dockerfile을 빌드하였다. Dockerfile은 그대로이기 때문에 3000번 포트를 expose한다.

Dockerfile의 내용을 살펴보면,

FROM node:8-alpine : 베이스 이미지를 node:8-alpine으로 지정

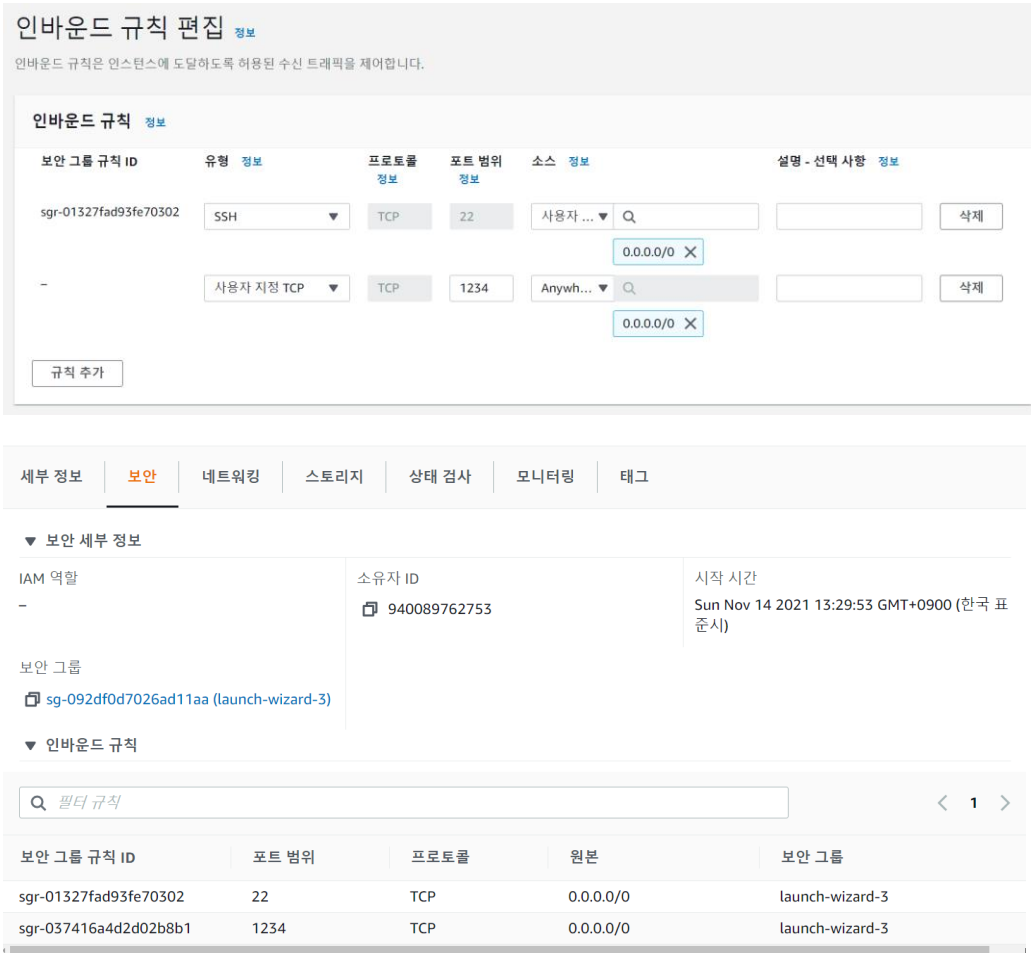
WORKDIR /usr/src/app : 작업 디렉토리를 /usr/src/app으로 지정

COPY . . : COPY <host OS 파일 경로> <컨테이너 안의 경로>로, 호스트 OS의 파일 또는 디렉토리를 컨테이너 안의 경로로 복사한다.

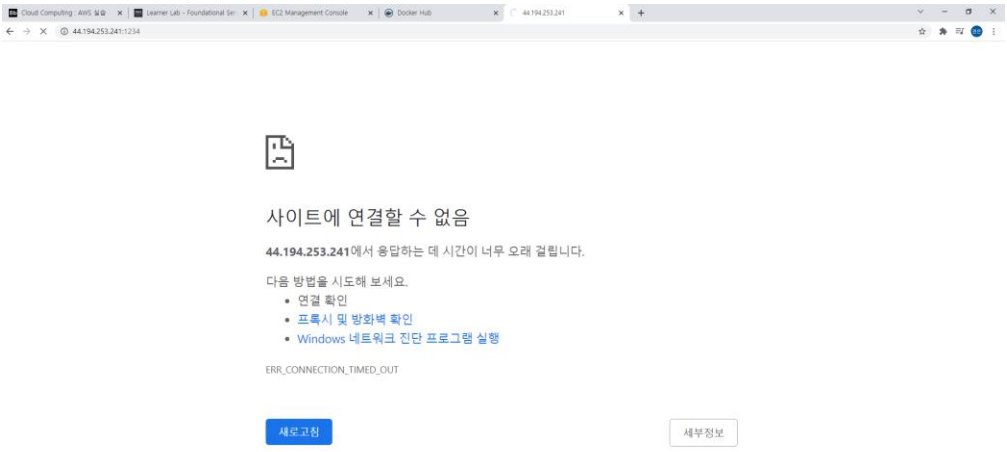
RUN npm install : npm install을 수행한다.

EXPOSE 3000 : 컨테이너의 3000번 포트를 열어준다

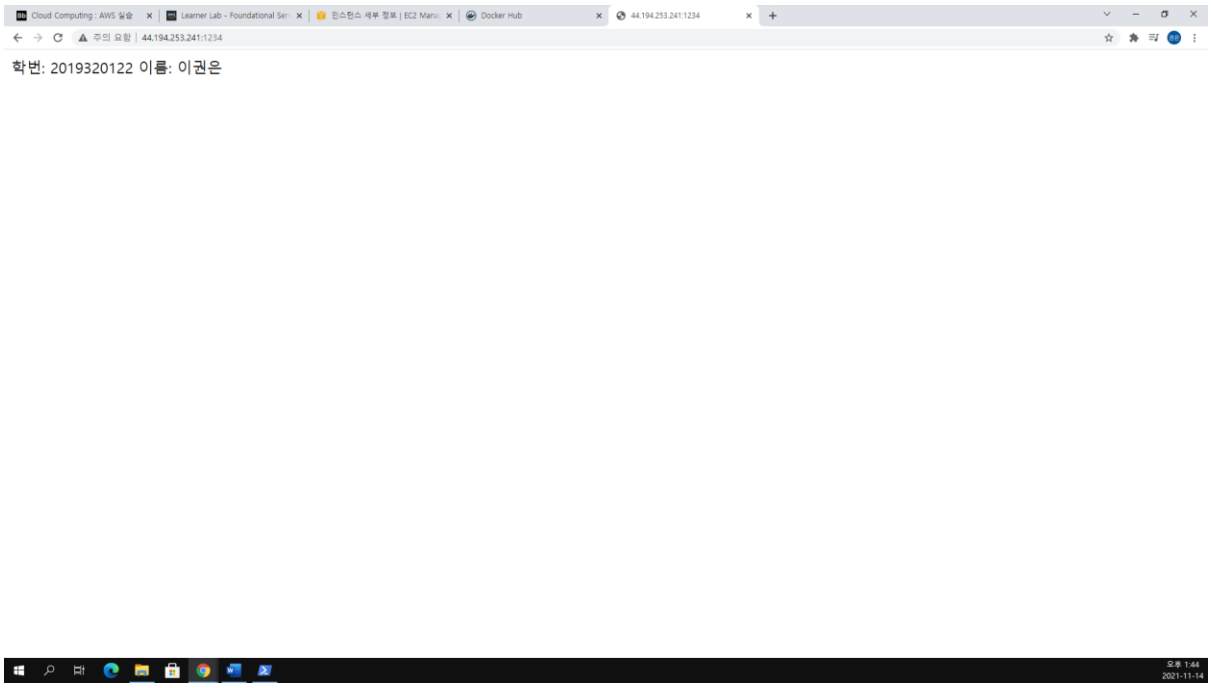
CMD ["node", "index.js"] : 서버를 실행한다.



개인 노트북/PC에서 브라우저를 통해 http://<ec2-public-ip-address>:1234를 입력했을 때 페이지가 출력되어야 하므로, 보안 그룹 인바운드(외부에서 인스턴스로 접근) 규칙에서 1234번 포트를 허용해주었다. 만약 이 과정이 없을 경우 1234번 포트가 차단되어 있기 때문에 다음과 같은 화면을 볼 수 있다.

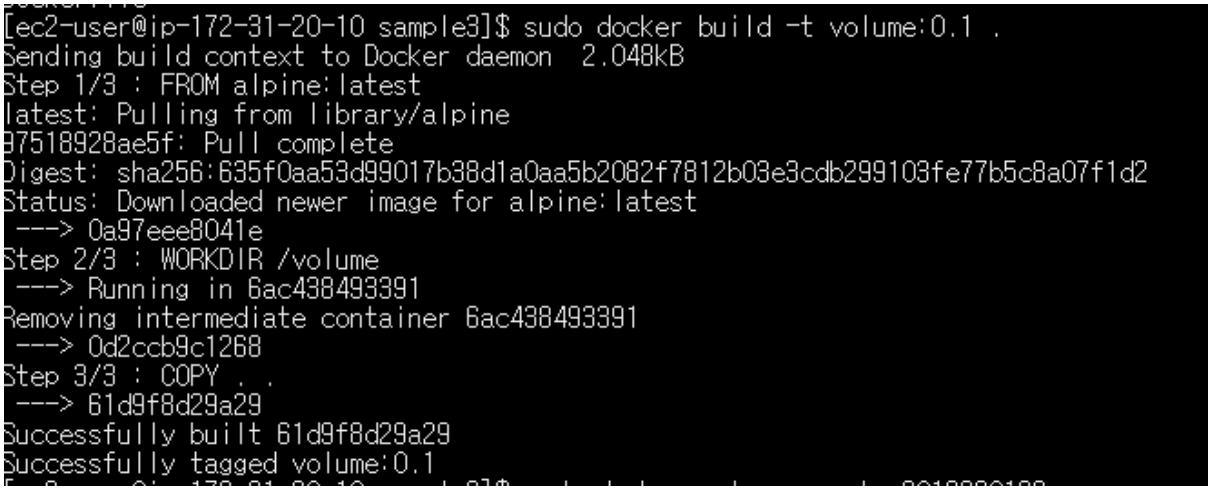


도커 컨테이너는 NAT 내부망에서 동작하고, 공인 IP 주소가 아닌 사설 IP 주소를 사용하기 때문에 외부에서 접속하기 위해서는 외부 주소와 내부 주소를 이어주는 포트포워딩이 필요하다. 포트포워딩은 NAT가 동작하는 라우터나 게이트웨이의 특정 포트 번호로 들어오는 트래픽을 NAT 내부에서 동작하는 시스템의 특정 포트로 전달해준다. 이번 과제에서는 1234번 포트로 트래픽이 들어오기 때문에, 이것을 Dockerfile에서 expose한 3000번 포트로 전달하도록 하였다. 즉 호스트 IP의 1234번 포트로 요청이 들어오면, 이것을 컨테이너의 3000번 포트로 포워딩하여 웹 서버에 요청이 도달한 것이다.



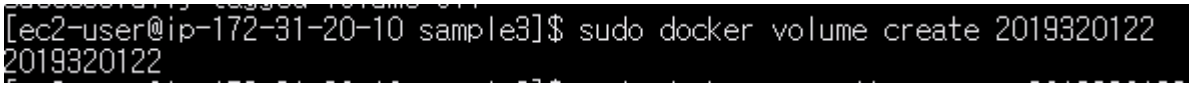
위와 같이 http://<ec2-public-ip-address>:1234로 접속했을 때 학번과 이름이 잘 출력된다.

과제 3 - volume

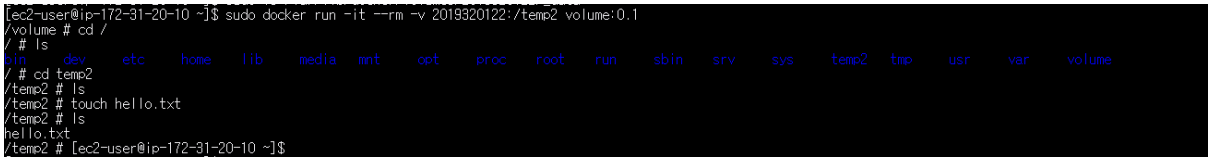


sampe3의 Dockerfile을 사용해서 volume:0.1 이미지를 생성하였다.
Dockerfile 내용을 살펴보면,

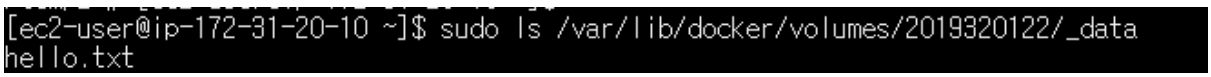
FROM alpine:latest : 베이스 이미지를 alpine:latest로 지정
WORKDIR /volume : 작업 디렉토리를 /volume으로 지정. docker run 시 첫 위치가 /volume 인 것을 확인할 수 있다.
COPY . . : COPY <host OS 파일 경로> <컨테이너 안의 경로>로, 호스트 OS의 파일 또는 디렉토리를 컨테이너 안의 경로로 복사한다. 그래서 docker run 후 /volume에서 ls 명령어를 실행해보면 Dockerfile이 있다.



학번을 이름으로 하는 volume을 생성하였다.



위에서 생성한 volume을 컨테이너에 마운트하여 (-v <volume name>:<컨테이너 경로>) volume:0.1 이미지로 컨테이너를 실행하였다. volume을 컨테이너 내의 /temp2에 마운트 했으므로, 해당 디렉토리에서 touch hello.txt 명령어로 hello.txt를 생성하면 volume내에도 똑 같은 파일 이 저장된다.



ls 명령어로 volume내부를 확인해보면 hello.txt가 있는 것을 확인할 수 있다.