

1. 주제

오픈 소스 프로젝트 배포

분반, 팀, 학번, 이름

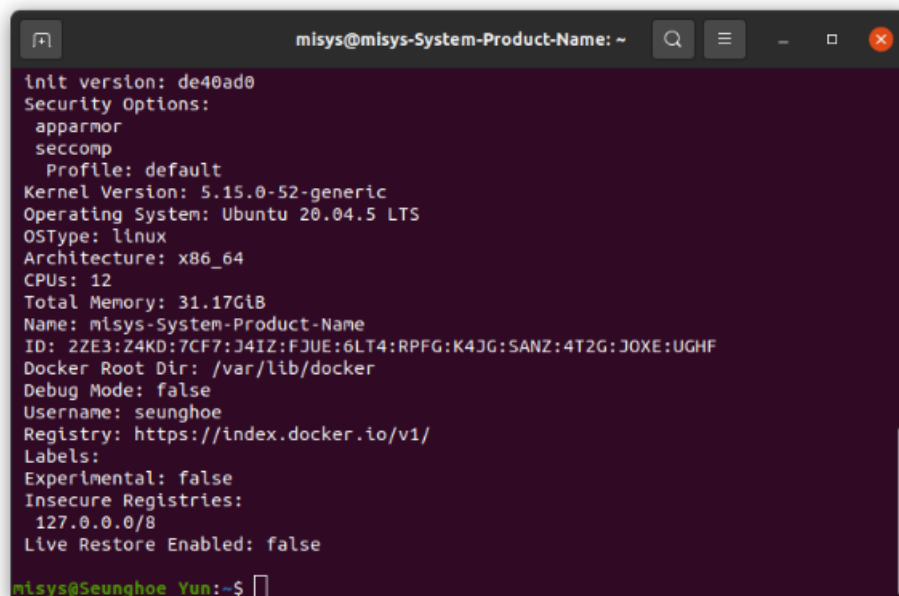
오픈소스기초설계(나), 5팀,

20180381 윤승회,

20180367 박상혁,

20180403 허석문

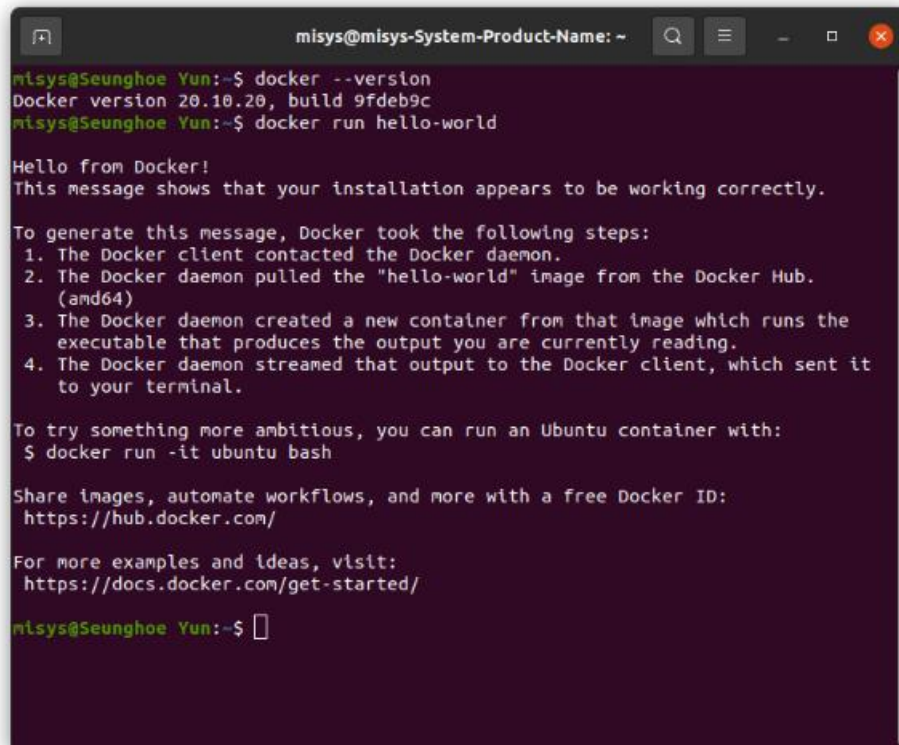
실습 #1 도커 버전 확인 및 기능 테스트

A terminal window with a dark background and light text. The title bar shows 'misys@misys-System-Product-Name: ~'. The terminal output displays various system and Docker configuration details. At the bottom, the prompt 'misys@Seunghoe Yun:~\$' is visible.

```
misys@misys-System-Product-Name: ~  
init version: de40ad0  
Security Options:  
  apparmor  
  seccomp  
    Profile: default  
Kernel Version: 5.15.0-52-generic  
Operating System: Ubuntu 20.04.5 LTS  
OSType: linux  
Architecture: x86_64  
CPUs: 12  
Total Memory: 31.17GiB  
Name: misys-System-Product-Name  
ID: 2ZE3:Z4KD:7CF7:J4IZ:FJUE:6LT4:RPFG:K4JG:SANZ:4T2G:JOXE:UGHF  
Docker Root Dir: /var/lib/docker  
Debug Mode: false  
Username: seunghoe  
Registry: https://index.docker.io/v1/  
Labels:  
Experimental: false  
Insecure Registries:  
  127.0.0.0/8  
Live Restore Enabled: false  
misys@Seunghoe Yun:~$
```

<그림 실습#1-1>

그림 실습#1-1 은 docker system info 명령어를 실행했을 때 결과 화면입니다.

A terminal window with a dark purple background and white text. The window title is 'misys@misys-System-Product-Name: ~'. The user 'misys@Seunghoe Yun' has entered two commands: 'docker --version' and 'docker run hello-world'. The output shows the Docker version (20.10.20) and a 'Hello from Docker!' message, followed by a list of steps Docker took to run the container and links to Docker Hub and documentation.

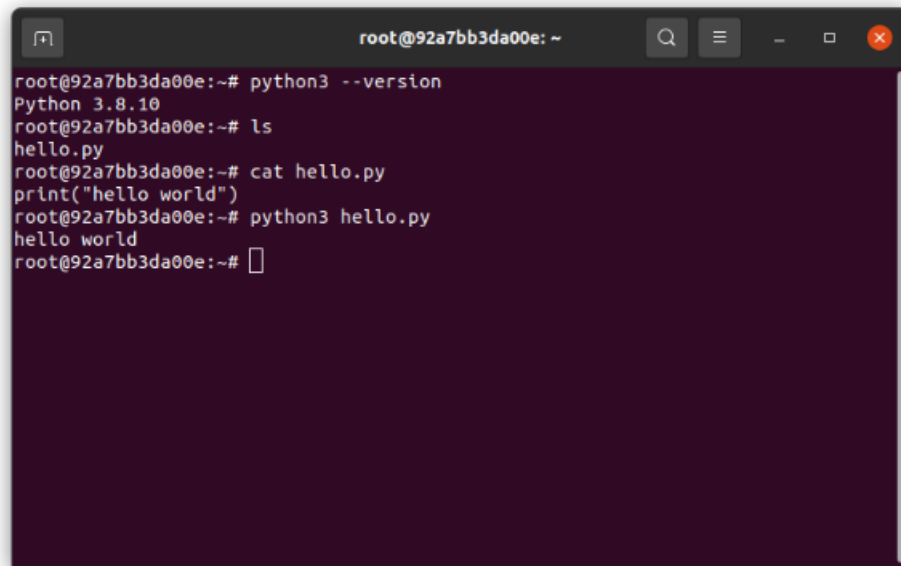
```
misys@misys-System-Product-Name: ~  
misys@Seunghoe Yun:~$ docker --version  
Docker version 20.10.20, build 9fdeb9c  
misys@Seunghoe Yun:~$ docker run hello-world  
  
Hello from Docker!  
This message shows that your installation appears to be working correctly.  
  
To generate this message, Docker took the following steps:  
1. The Docker client contacted the Docker daemon.  
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.  
   (amd64)  
3. The Docker daemon created a new container from that image which runs the  
   executable that produces the output you are currently reading.  
4. The Docker daemon streamed that output to the Docker client, which sent it  
   to your terminal.  
  
To try something more ambitious, you can run an Ubuntu container with:  
$ docker run -it ubuntu bash  
  
Share images, automate workflows, and more with a free Docker ID:  
https://hub.docker.com/  
  
For more examples and ideas, visit:  
https://docs.docker.com/get-started/  
misys@Seunghoe Yun:~$
```

<그림 실습#1-2>

그림 실습#1-2 는 docker --version 명령어와 docker run hello-world를 실행했을 때 결과 화면입니다.

Docker의 버전이 20.10.20 으로 설치가 잘 된것을 확인할 수 있었습니다.

실습 #2 도커 환경에서의 코드 작성

A terminal window with a dark purple background and white text. The window title is 'root@92a7bb3da00e: ~'. The terminal shows the following commands and output:

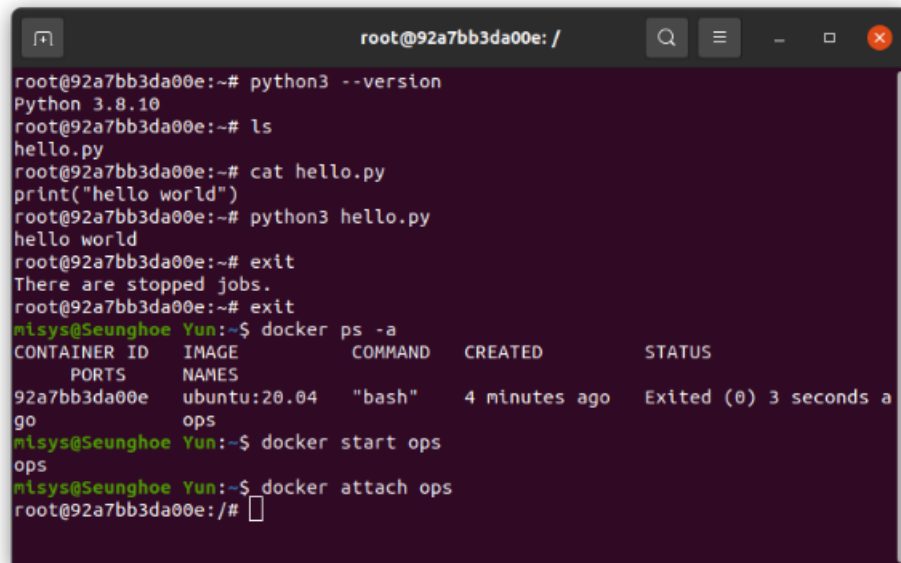
```
root@92a7bb3da00e:~# python3 --version
Python 3.8.10
root@92a7bb3da00e:~# ls
hello.py
root@92a7bb3da00e:~# cat hello.py
print("hello world")
root@92a7bb3da00e:~# python3 hello.py
hello world
root@92a7bb3da00e:~#
```

<그림 실습#2-1>

그림 실습#2-1 은 docker 컨테이너 내부에서 python3 의 버전을 확인하고 hello.py라는 파일의 내용을 출력하고 hello.py 를 실행한 결과창입니다.

이전에 과정은 다음과 같습니다.

1. Docker pull 명령어를 통해서 ubuntu:20.04를 다운로드 해주었습니다.
2. Docker images 를 확인하게 되면 ubuntu:20.04 이미지가 저장되어 있는 것을 확인 할 수 있었습니다.
3. Docker run -it --name (컨테이너 이름) ubuntu:20.04 명령어를 이용하여 컨테이너를 생성해 주었습니다.
4. 컨테이너를 생성하면 처음 우분투를 설치하는 것과 동일한 환경이 됩니다. 따라서 apt를 업데이트 해주고 필요한 (vim, python3) 를 설치해 주었습니다.
5. Home 디렉토리로 이동하여 hello.py 코드를 작성한 다음 위의 명령어를 실행해 주었습니다.



```

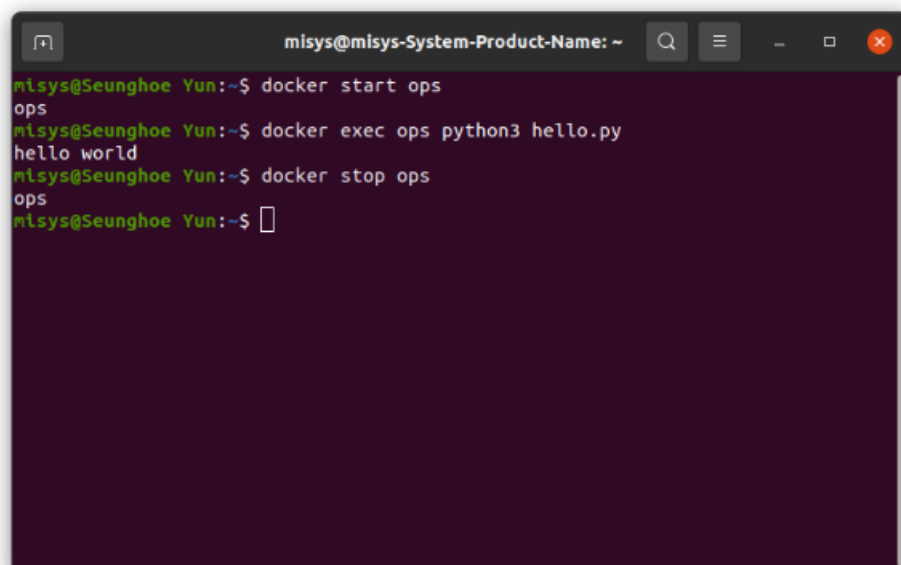
root@92a7bb3da00e: /
root@92a7bb3da00e:~# python3 --version
Python 3.8.10
root@92a7bb3da00e:~# ls
hello.py
root@92a7bb3da00e:~# cat hello.py
print("hello world")
root@92a7bb3da00e:~# python3 hello.py
hello world
root@92a7bb3da00e:~# exit
There are stopped jobs.
root@92a7bb3da00e:~# exit
misys@Seunghoe Yun:~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS
PORTS         NAMES
92a7bb3da00e   ubuntu:20.04  "bash"                  4 minutes ago  Exited (0) 3 seconds a
go
ops
misys@Seunghoe Yun:~$ docker start ops
ops
misys@Seunghoe Yun:~$ docker attach ops
root@92a7bb3da00e:/#

```

<그림 실습#2-2>

그림 실습#2-2 는 사용중이던 컨테이너에서 나가고 생성되어 있는 컨테이너를 확인하는 명령어 `docker ps -a`를 통해서 컨테이너 정보를 확인해 주었습니다. 생성되어 있는 컨테이너를 다시 실행하는 경우는 `docker start` 명령어와 `docker attach` 명령어를 통해 실행, 컨테이너에 잘 들어가지는 것을 확인할 수 있었습니다.

실습 #3 도커 환경을 배포



```

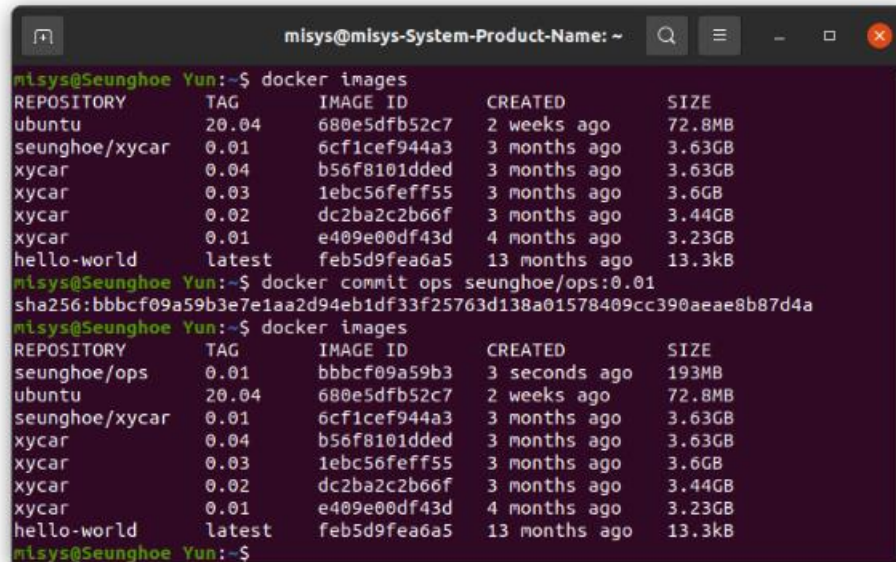
misys@misys-System-Product-Name: ~
misys@Seunghoe Yun:~$ docker start ops
ops
misys@Seunghoe Yun:~$ docker exec ops python3 hello.py
hello world
misys@Seunghoe Yun:~$ docker stop ops
ops
misys@Seunghoe Yun:~$

```

<그림 실습#3-1>

그림 실습#3-1은 `ops` 라는 이름의 컨테이너를 실행해주고 `docker exec` 명령어를 통해서 `ops`

내에 있는 hello.py 소스코드를 실행하는 명령어입니다. 실행이 잘되는 것을 확인하고 컨테이너를 종료해주는 docker stop 명령어를 통해 종료해주었습니다.



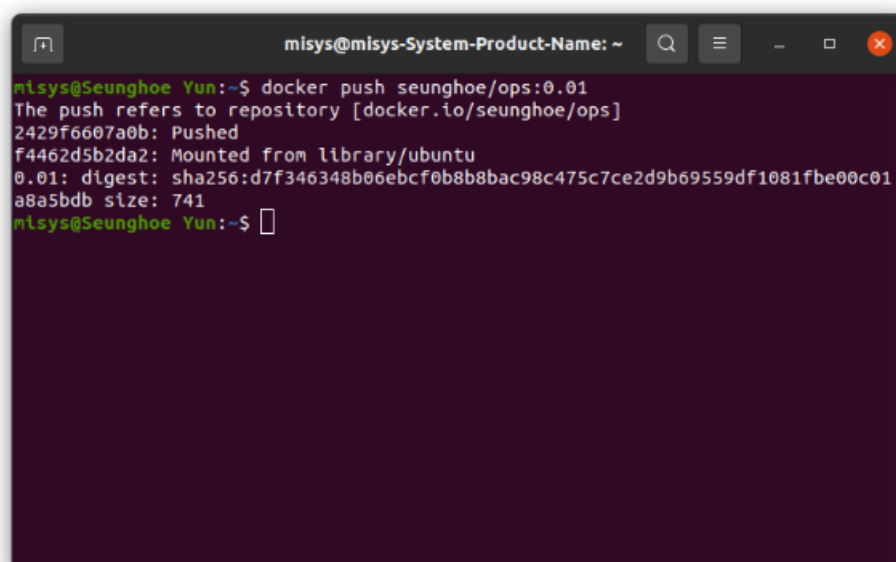
```

misys@misys-System-Product-Name: ~
misys@Seunghoe Yun:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              20.04              680e5dfb52c7       2 weeks ago        72.8MB
seunghoe/xycar      0.01               6cf1cef944a3       3 months ago       3.63GB
xycar                0.04               b56f8101dded       3 months ago       3.63GB
xycar                0.03               1ebc56feff55       3 months ago       3.6GB
xycar                0.02               dc2ba2c2b66f       3 months ago       3.44GB
xycar                0.01               e409e00df43d       4 months ago       3.23GB
hello-world         latest             feb5d9fea6a5       13 months ago      13.3kB
misys@Seunghoe Yun:~$ docker commit ops seunghoe/ops:0.01
sha256:bbbcf09a59b3e7e1aa2d94eb1df33f25763d138a01578409cc390aae8b87d4a
misys@Seunghoe Yun:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
seunghoe/ops        0.01               bbbcf09a59b3       3 seconds ago      193MB
ubuntu              20.04              680e5dfb52c7       2 weeks ago        72.8MB
seunghoe/xycar      0.01               6cf1cef944a3       3 months ago       3.63GB
xycar                0.04               b56f8101dded       3 months ago       3.63GB
xycar                0.03               1ebc56feff55       3 months ago       3.6GB
xycar                0.02               dc2ba2c2b66f       3 months ago       3.44GB
xycar                0.01               e409e00df43d       4 months ago       3.23GB
hello-world         latest             feb5d9fea6a5       13 months ago      13.3kB
misys@Seunghoe Yun:~$

```

<그림 실습#3-2>

그림 실습#3-2는 사용중인 컨테이너를 이미지로 만드는 것입니다. Docker commit 명령어를 통해서 ops 이름의 컨테이너를 seunghoe/ops:0.01 이라는 이미지로 만들어 주었습니다. 앞에 seunghoe를 붙힌 이유는 도커홈페이지에 올리기 위해서는 닉네임과 동일해야하기 때문입니다. Commit 이후 이미지가 잘 생성된 것을 확인할 수 있습니다.



```

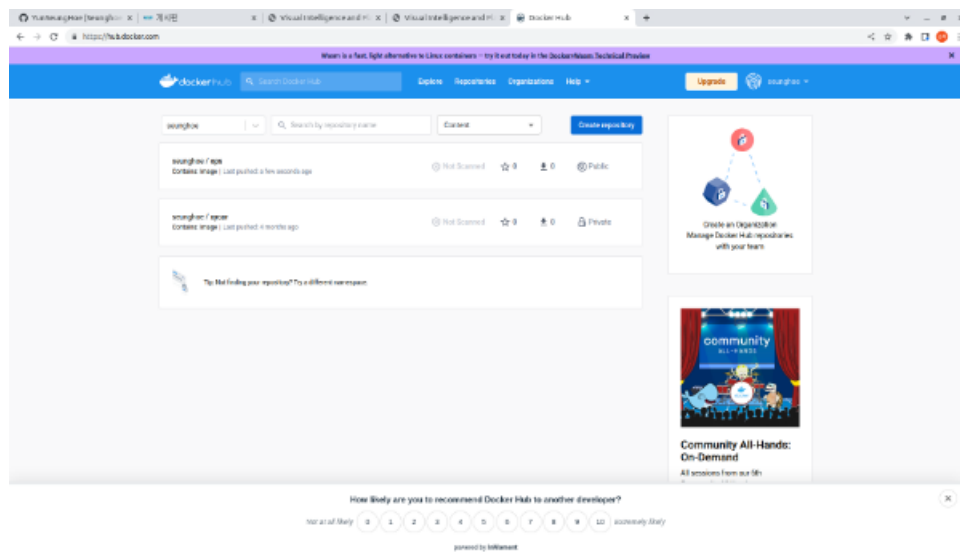
misys@misys-System-Product-Name: ~
misys@Seunghoe Yun:~$ docker push seunghoe/ops:0.01
The push refers to repository [docker.io/seunghoe/ops]
2429f6607a0b: Pushed
f4462d5b2da2: Mounted from library/ubuntu
0.01: digest: sha256:d7f346348b06ebcf0b8b8bac98c475c7ce2d9b69559df1081fbe00c01a8a5bdb size: 741
misys@Seunghoe Yun:~$

```

<그림 실습#3-3>

그림 실습#3-3에서는 docker push 명령어를 사용하여 저의 docker 계정에 seunghoe/ops:0.01

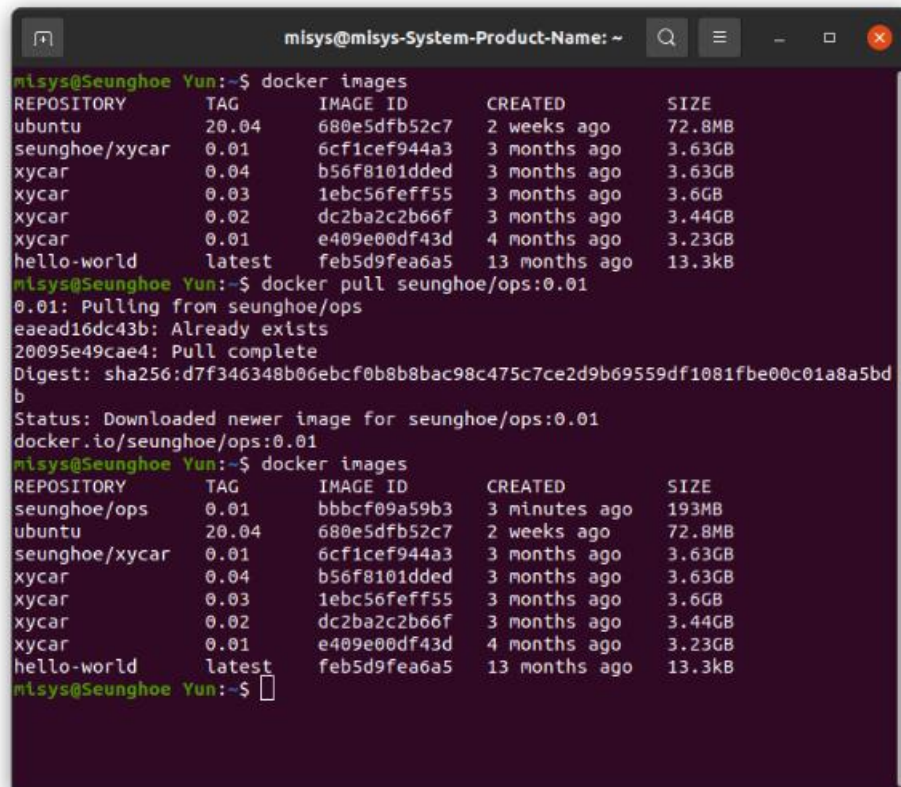
이미지를 올려주는 화면입니다.



<그림 실습#3-4>

그림 실습#3-4 docker push 이후 docker 홈페이지에 들어가보면 이미지가 public으로 잘 올라가진 모습을 확인할 수 있습니다.

실습 #4 배포한 환경을 검증



```
misys@misys-System-Product-Name: ~  
misys@Seunghoe Yun:~$ docker images  
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE  
ubuntu          20.04       680e5dfb52c7  2 weeks ago   72.8MB  
seunghoe/xycar  0.01       6cf1cef944a3  3 months ago  3.63GB  
xycar           0.04       b56f8101dded  3 months ago  3.63GB  
xycar           0.03       1ebc56feff55  3 months ago  3.6GB  
xycar           0.02       dc2ba2c2b66f  3 months ago  3.44GB  
xycar           0.01       e409e00df43d  4 months ago  3.23GB  
hello-world     latest      feb5d9fea6a5  13 months ago 13.3kB  
misys@Seunghoe Yun:~$ docker pull seunghoe/ops:0.01  
0.01: Pulling from seunghoe/ops  
eaead16dc43b: Already exists  
20095e49cae4: Pull complete  
Digest: sha256:d7f346348b06ebcf0b8b8bac98c475c7ce2d9b69559df1081fbe00c01a8a5bd  
b  
Status: Downloaded newer image for seunghoe/ops:0.01  
docker.io/seunghoe/ops:0.01  
misys@Seunghoe Yun:~$ docker images  
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE  
seunghoe/ops    0.01       bbbcf09a59b3  3 minutes ago  193MB  
ubuntu          20.04       680e5dfb52c7  2 weeks ago   72.8MB  
seunghoe/xycar  0.01       6cf1cef944a3  3 months ago  3.63GB  
xycar           0.04       b56f8101dded  3 months ago  3.63GB  
xycar           0.03       1ebc56feff55  3 months ago  3.6GB  
xycar           0.02       dc2ba2c2b66f  3 months ago  3.44GB  
xycar           0.01       e409e00df43d  4 months ago  3.23GB  
hello-world     latest      feb5d9fea6a5  13 months ago 13.3kB  
misys@Seunghoe Yun:~$
```

<그림 실습#4-1>

그림 실습#4-1 화면은 현재 이미지를 확인후 실습3에서 올린 이미지 파일을 다운로드하고 이미지를 확인하는 화면입니다. 이전에 사용중이던 docker 이미지는 docker rmi 명령어를 이용, 컨테이너는 docker rm 을 이용하여 제거해주었습니다.


```

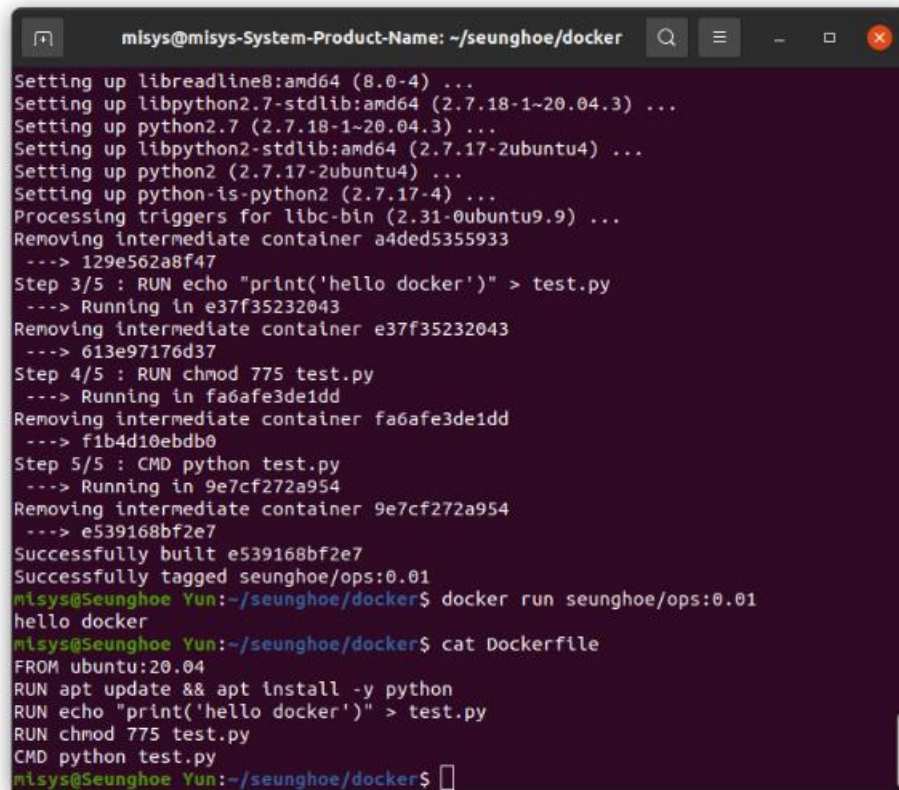
misys@misys-System-Product-Name: ~
misys@Seunghoe Yun:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
seunghoe/ops         0.01               bbbcf09a59b3       4 minutes ago      193MB
ubuntu               20.04              680e5dfb52c7       2 weeks ago        72.8MB
seunghoe/xycar       0.01               6cf1cef944a3       3 months ago       3.63GB
xycar                 0.04               b56f8101dded       3 months ago       3.63GB
xycar                 0.03               1ebc56feff55       3 months ago       3.6GB
xycar                 0.02               dc2ba2c2b66f       3 months ago       3.44GB
xycar                 0.01               e409e00df43d       4 months ago       3.23GB
hello-world          latest             feb5d9fea6a5       13 months ago      13.3kB
misys@Seunghoe Yun:~$ docker run -it --name copy_ops seunghoe/ops:0.01
root@5dcc0594a2f1:/# exit
misys@Seunghoe Yun:~$ docker ps -a
CONTAINER ID   IMAGE                  COMMAND             CREATED             STATUS
5dcc0594a2f1   seunghoe/ops:0.01     "bash"             5 seconds ago      Exited (0) 1 seco
nd ago
copy_ops
misys@Seunghoe Yun:~$ docker start copy_ops
copy_ops
misys@Seunghoe Yun:~$ docker exec copy_ops python3 hello.py
hello world
misys@Seunghoe Yun:~$ docker stop copy_ops
copy_ops
misys@Seunghoe Yun:~$ docker ps -a
CONTAINER ID   IMAGE                  COMMAND             CREATED             STATUS
5dcc0594a2f1   seunghoe/ops:0.01     "bash"             33 seconds ago      Exited (0) 2 sec
onds ago
copy_ops
misys@Seunghoe Yun:~$

```

<그림 실습#4-2>

그림 실습#4-2 는 seunghoe/ops:0.01 이라는 이미지를 이용하여 copy_ops 라는 이름의 컨테이너를 생성해 주었고, 바로 종료후 docker ps -a 를 통해서 컨테이너가 잘 생성된 것을 확인하였습니다. 다시 docker start 명령어를 통해 컨테이너를 실행, docker exec 명령어를 통해서 copy_ops 안에 있는 hello.py 파일을 실행해 보았을 때, 잘 작동하는 것을 확인 할 수있었습니다.

실습 #5 도커 파일 작성 실습



```
misys@misys-System-Product-Name: ~/seunghoe/docker
Setting up libreadline8:amd64 (8.0-4) ...
Setting up libpython2.7-stdlib:amd64 (2.7.18-1~20.04.3) ...
Setting up python2.7 (2.7.18-1~20.04.3) ...
Setting up libpython2-stdlib:amd64 (2.7.17-2ubuntu4) ...
Setting up python2 (2.7.17-2ubuntu4) ...
Setting up python-is-python2 (2.7.17-4) ...
Processing triggers for libc-bin (2.31-0ubuntu9.9) ...
Removing intermediate container a4ded5355933
--> 129e562a8f47
Step 3/5 : RUN echo "print('hello docker')" > test.py
--> Running in e37f35232043
Removing intermediate container e37f35232043
--> 613e97176d37
Step 4/5 : RUN chmod 775 test.py
--> Running in fa6afe3de1dd
Removing intermediate container fa6afe3de1dd
--> f1b4d10ebdb0
Step 5/5 : CMD python test.py
--> Running in 9e7cf272a954
Removing intermediate container 9e7cf272a954
--> e539168bf2e7
Successfully built e539168bf2e7
Successfully tagged seunghoe/ops:0.01
misys@Seunghoe Yun:~/seunghoe/docker$ docker run seunghoe/ops:0.01
hello docker
misys@Seunghoe Yun:~/seunghoe/docker$ cat Dockerfile
FROM ubuntu:20.04
RUN apt update && apt install -y python
RUN echo "print('hello docker')" > test.py
RUN chmod 775 test.py
CMD python test.py
misys@Seunghoe Yun:~/seunghoe/docker$
```

<그림 실습#5-1>

그림 실습#5-1 은 docker 파일을 생성해 주는 것입니다. Dockerfile 내부는 그림과 같이 작성해 주었습니다. 따라서 seunghoe/ops:0.01 이미지를 실행하면 hello docker 라는 결과가 나오는 것을 알 수 있습니다. FROM 은 사용할 이미지, RUN 과 CMD 를 이용해서 실행주었습니다.