Docker 실습 1, 2 - 설치와 기본 명령어

---

1. Docker 설치

- 공식 문서
    - https://docs.docker.com/engine/install/ubuntu/
    - 여러가지 install 방법 중, **Install using the repository** 방법으로 진행하겠습니다.
- 앞으로 어떤 오픈소스를 사용하더라도, 블로그가 아닌 공식 문서를 참고하시는 습관을 들이는 것을 추천 드리겠습니다.

1) Set up the repository

- 먼저 apt 라는 패키지 매니저를 업데이트합니다.

```
$ sudo apt-get update
```

- 그리고, docker 의 prerequisite package 들을 설치합니다.

```
$ sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
```

- Docker 의 GPG key 를 추가합니다.

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
 /usr/share/keyrings/docker-archive-keyring.gpg
```

- 그 다음 stable 버전의 repository 를 바라보도록 설정합니다.

```
$ echo \
  "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] http
s://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /de
v/null
```

  - 혹시 arm 기반의 cpu 라면 다음을 입력해주세요

```
echo \
  "deb [arch=arm64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
 https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list >
/dev/null
```

2) Install Docker Engine

- Docker 엔진의 최신 버전을 설치합니다.

```
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

- 혹시 특정 버전의 Docker 를 설치하고 싶다면 매뉴얼을 따라해주세요

3) 정상 설치 확인

- docker container 를 실행시켜, 정상적으로 설치되었는지 확인합니다.

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

- 다음과 같이 출력 된다면, 정상적으로 설치가 된 것을 확인할 수 있습니다

```
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
b8dfde127a29: Pull complete
Digest: sha256:0fe98d7debd9049c50b597ef1f85b7c1e8cc81f59c8d623fcb2250e8bec85b38
Status: Downloaded newer image for hello-world:latest

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/
```

2. Docker 권한 설정

- 현재는 모든 docker 관련 작업이 root 유저에게만 권한이 있기 때문에, docker 관련 명령을 수행하려면 `sudo` 를 앞에 붙여주어야만 가능합니다.
- 예를 들면,
  - `docker ps` 를 수행하면 다음과 같이 Permission denied 라는 메시지가 출력됩니다.

```
Got permission denied while trying to connect to the Docker daemon socket at uni
x:///var/run/docker.sock: Get http://%2Fvar%2Frun%2Fdocker.sock/v1.40/containers/j
son?all=1: dial unix /var/run/docker.sock: connect: permission denied
```

- 따라서, root 유저가 아닌 host 의 기본 유저에게도 권한을 주기 위해 다음과 같은 명령을 **새로 띄운 터미널에서 수행해주셔야 합니다.**

```
$ sudo usermod -a -G docker $USER
$ sudo service docker restart
```

- 그 다음, VM 을 로그아웃 한 다음에 다시 로그인하면 다음과 같이 정상적으로 출력되는 것을 확인하실 수 있습니다.

```
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

---

3. Docker 의 기본적인 명령

1) Docker pull

- docker image repository 부터 Docker image 를 가져오는 커맨드입니다.

```
$ docker pull --help
```

- 예시)

```
$ docker pull ubuntu:18.04
```

  - docker.io/library 라는 이름의 repository 에서 ubuntu:18.04 라는 image 를 여러분의 노트북에 다운로드 받게 됩니다.

- 참고사항
  - 추후 docker.io 나 public 한 docker hub 와 같은 repository 대신에, 특정 private 한 repository 에서 docker image 를 가져와야 하는 경우:

    - docker login 을 통해서 특정 repository 를 바라보도록 한 뒤, docker pull 을 수행하는 형태로 사용합니다.

2) Docker images

- 로컬에 존재하는 docker image 리스트를 출력하는 커맨드입니다.

```
$ docker images --help
```

- 예시)

```
$ docker images
```

3) Docker ps

- 현재 실행중인 도커 컨테이너 리스트를 출력하는 커맨드입니다.

```
$ docker ps --help
```

- 예시)

```
$ docker ps
$ docker ps -a
```

4) Docker run

- 도커 컨테이너를 실행시키는 커맨드입니다.

```
$ docker run --help
```

- 옵션에 대한 자세한 내용은 사용하게 될 때마다 하나씩 설명 예정

- 예시)

```
$ docker run -it --name demo1 ubuntu:18.04 /bin/bash
```

  - `-it` : `-i` 옵션 + `-t` 옵션
    - container 를 실행시킴과 동시에 interactive 한 terminal 로 접속시켜주는 옵션
  - `--name` : name
    - 컨테이너 id 대신, 구분하기 쉽도록 지정해주는 이름
  - `/bin/bash`
    - 컨테이너를 실행시킴과 동시에 실행할 커맨드로, `/bin/bash` 는 bash 터미널을 사용하는 것을 의미합니다.

5) Docker exec

- Docker 컨테이너 내부에서 명령을 내리거나, 내부로 접속하는 커맨드

```
$ docker exec --help
```

- 예시)

```
$ docker run -it -d --name demo2 ubuntu:18.04
$ docker ps
```

  - `-d` : 백그라운드에서 실행시켜서, 컨테이너에 접속 종료를 하더라도, 계속 실행 중이 되도록 하는 커맨드

```
$ docker exec -it demo2 /bin/bash
```

  - 아까와 동일하게 container 내부에 접속할 수 있는 것을 확인 가능

6) Docker logs

- 도커 컨테이너의 log 를 확인하는 커맨드

```
$ docker logs --help
```

- 예시)

```
$ docker run --name demo3 -d busybox sh -c "while true; do $(echo date); sleep 1;
 done"
```

  - test 라는 이름의 busybox 이미지를 백그라운드에서 도커 컨테이너로 실행하여, 1 초에 한 번씩 현재 시간을 출력하는 커맨드

```
$ docker logs demo3
$ docker logs demo3 -f
```

  - `-f` 옵션: 계속 watch 하며 출력

7) Docker stop

- 실행 중인 도커 컨테이너를 중단시키는 커맨드

```
$ docker stop --help
```

- 예시)

```
$ docker stop demo3
$ docker stop demo2
$ docker stop demo1
```

8) Docker rm

- 도커 컨테이너를 삭제하는 커맨드

```
$ docker rm --help
```

- 예시)

```
$ docker rm demo3
$ docker rm demo2
$ docker rm demo1
```

  - docker ps, docker ps -a 에서 모두 출력되지 않는 것을 확인하실 수 있습니다.

9) Docker rmi

- 도커 이미지를 삭제하는 커맨드

```
$ docker rmi --help
```

- 예시)

```
$ docker images
# busybox, ubuntu 가 있는 것을 확인하실 수 있습니다.
$ docker rmi ubuntu
```