[Hands-on] 02. Docker layers

이번 실습은 컨테이너와 이미지, 그리고 Layer에 대한 실습입니다.

먼저 현재 Host에 있는 컨테이너 이미지를 모두 삭제하겠습니다.

```
ubuntu@ip-10-0-1-14:~$ docker rmi --force $(docker images -aq)
Untagged: docker-101:latest
Deleted: sha256:25d5349823915cda37e5cb03dc0e3520ca689fa9c9c67dae71e2168d9cb1f00e
Deleted: sha256:d86f278413c56f277591d2ea3de6739ffc8bf5b88c2cd8ba5ebaee22a28db4e2
Deleted: sha256:36e8ca2bb99a00e2b026794c5f20cd5d586e82dbf21319039d30dc166c65afa4
Deleted: sha256:027fa7ba112b507788a369d6a59ccd78a3bd17c0f4bcd883e9ee1be57ea5bfc5
Deleted: sha256:40d929a5abbd730be7f99fe0a7f8c98d9eba45ff79a20ab0a46958e7b911e5c8
Deleted: sha256:ff7e4526409ea0d633ef76c6e9d4df0ac7bb67fa649643b03eadef1ea00f1e0a
Deleted: sha256:6df32015dd8fca2d2723961255c5131350122e035b0e57b9ba52ad47a50a9231
```

> 명령어 : `docker rmi --force $(docker images -aq)`

모두 삭제됐는지 볼까요?

```
ubuntu@ip-10-0-1-14:~$ docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
ubuntu@ip-10-0-1-14:~$
```

> 명령어 : `docker images`

위와 같이 조회되는 image가 하나도 없는 상태여야 합니다.

Docker & Kubernetes - [Hands-on] 02. Docker layers

이제 우리 도커가 어떤 스토리지 드라이버를 사용하는지 알아보겠습니다.

```
ubuntu@ip-10-0-1-14:~$ docker info | grep -i storage
 Storage Driver: overlay2
```

> 명령어 : `docker info | grep -i storage`

스토리지 드라이버 에 따라서 이미지가 저장되는 장소가 달라집니다.
우리 시스템은 `overlay2` 드라이버이기 때문에 아래 경로가 사용됩니다.

```
ubuntu@ip-10-0-1-14:~$ ls -al /var/lib/docker/overlay2
ls: cannot access '/var/lib/docker/overlay2': Permission denied
ubuntu@ip-10-0-1-14:~$ sudo ls -al /var/lib/docker/overlay2
total 20
drwx--x---  3 root root 12288 Jun 17 05:55 .
drwx--x--- 13 root root  4096 Jun 17 02:51 ..
drwx------  2 root root  4096 Jun 17 05:55 l
```

> 명령어 : `sudo ls -al /var/lib/docker/overlay2`

- `docker 관련 디렉토리는 소유자가 root이기 때문에 다른 사용자가 조회하려면 sudo를 사용해야 합니다.`

위 그림처럼 깨끗하게 정리된 상태에서 시작해볼게요.

> Docker desktop for widows를 설치한 환경인 경우(WSL2기반), Docker Root dir.의 위치가 다릅니다.
> 탐색기에서 `\\wsl.localhost\docker-desktop-data\data\docker`를 찾아보면 됩니다.
> Overlay2인 경우 `\\wsl.localhost\docker-desktop-data\data\docker\overlay2` 가 이미지가 저장되는 장소입니다.

Docker & Kubernetes - [Hands-on] 02. Docker layers

nginx 이미지를 하나 pull 하구요.

```
ubuntu@ip-10-0-1-14:~$ docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
42c077c10790: Pull complete
62c70f376f6a: Pull complete
915cc9bd79c2: Pull complete
75a963e94de0: Pull complete
7b1fab684d70: Pull complete
db24d06d5af4: Pull complete
Digest: sha256:2bcabc23b45489fb0885d69a06ba1d648aeda973fae7bb981bafbb884165e514
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
```

> 명령어 : `docker pull nginx`

이미지 목록을 확인을 합니다.

```
ubuntu@ip-10-0-1-14:~$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
nginx        latest    0e901e68141f   2 weeks ago   142MB
```

> 명령어 : `docker images`

Docker & Kubernetes - [Hands-on] 02. Docker layers

이제 다시 저장된 위치(`/var/lib/docker/overlay2`)가 어떻게 바뀌었나 확인해볼까요?

```
ubuntu@ip-10-0-1-14:~$ sudo ls -al /var/lib/docker/overlay2
total 44
drwx--x---  9 root root 12288 Jun 17 06:01 .
drwx--x--- 13 root root  4096 Jun 17 02:51 ..
drwx--x---  4 root root  4096 Jun 17 06:01 5dc378c99eedd46fc3aca29e771c7a044b91ac71f4ebff2da0be8260da057350
drwx--x---  4 root root  4096 Jun 17 06:01 73b619a2efde2fe0bac0940fa6892ad5c7edc9fef14ba2b0a5605ce6ae4963c0
drwx--x---  4 root root  4096 Jun 17 06:01 c77d0c3677b1d3d949038beef16bcf0dfe6b9da76c79efcf558463c937779022
drwx--x---  4 root root  4096 Jun 17 06:01 d060bd3b029045629a36a42774c68afc43716d9bfdbbb2bcca810e530262e604
drwx--x---  4 root root  4096 Jun 17 06:01 d0cd4377c363119f63c59a2d32db03e0deba873c11cc9f0885a9b343b57387a1
drwx--x---  3 root root  4096 Jun 17 06:01 fccb8e1376f1c76242fc79698262f7498b78c3c8090fc9f8b79bd29d6b8e460d
drwx------  2 root root  4096 Jun 17 06:01 l
```

> **명령어** : `sudo ls -al /var/lib/docker/overlay2`

뭔가 많이 생겼네요.
혹시 눈치 채셨나요?
`docker pull` 할때 표시된 layer만큼 overlay 아래 디렉토리가 생성된걸...

이번엔 컨테이너를 실행해 보겠습니다.

```
ubuntu@ip-10-0-1-14:~$ docker run --detach --label "color=red" nginx
237e34821eea34627bc4de44741318a45a386d3d76c249ff958cad89b0ffb176
```

> **명령어** : `docker run --detach --label "color=red" nginx`

다음 실습을 위해서 label(`color=red`)을 달아뒀습니다.

잘 실행되고 있나 살펴볼게요.

```
ubuntu@ip-10-0-1-14:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS              PORTS     NAMES
237e34821eea   nginx     "/docker-entrypoint.…"   About a minute ago   Up About a minute   80/tcp    ecstatic_visvesvaraya
```

> 명령어 : `docker ps`

이제 `overlay2` 디렉토리는 어떻게 바뀌어 있을까요?

```
ubuntu@ip-10-0-1-14:~$ sudo ls -alt /var/lib/docker/overlay2
total 52
drwx--x---  5 root root  4096 Jun 17 06:04 85790c7c79f6a9ed42c56001ecbcca73a247e3dae52e70642cfa1deb8717b345
drwx--x--- 11 root root 12288 Jun 17 06:04 .
drwx--x---  4 root root  4096 Jun 17 06:04 85790c7c79f6a9ed42c56001ecbcca73a247e3dae52e70642cfa1deb8717b345-init
drwx------  2 root root  4096 Jun 17 06:04 l
drwx--x---  4 root root  4096 Jun 17 06:04 d060bd3b029045629a36a42774c68afc43716d9bfdbbb2bcca810e530262e604
drwx--x---  4 root root  4096 Jun 17 06:01 c77d0c3677b1d3d949038beef16bcf0dfe6b9da76c79efcf558463c937779022
drwx--x---  4 root root  4096 Jun 17 06:01 d0cd4377c363119f63c59a2d32db03e0deba873c11cc9f0885a9b343b57387a1
drwx--x---  4 root root  4096 Jun 17 06:01 5dc378c99eedd46fc3aca29e771c7a044b91ac71f4ebff2da0be8260da057350
drwx--x---  4 root root  4096 Jun 17 06:01 73b619a2efde2fe0bac0940fa6892ad5c7edc9fef14ba2b0a5605ce6ae4963c0
drwx--x---  3 root root  4096 Jun 17 06:01 fccb8e1376f1c76242fc79698262f7498b78c3c8090fc9f8b79bd29d6b8e460d
drwx--x--- 13 root root  4096 Jun 17 02:51 ..
```

> 명령어 : `sudo ls -alt /var/lib/docker/overlay2`

두 개의 디렉토리가 더 생긴걸 볼 수 있습니다. (`-t` 옵션을 사용하여 최근 디렉토리를 상위에 표시함.)

이게 우리가 배운 R/W Layer인 Container layer입니다.
실행되는 컨테이너에서 발생하는 모든 변경사항은 바로 여기 기록되게 됩니다.

그럼, 하나를 더 실행하면 어떻게 될까요?

```
ubuntu@ip-10-0-1-14:~$ docker run --detach --label "color=blue" nginx
670302b2287fe6bc297ba9e0499c285febd4752a3faffb4eda981ee55805190c
```

> 명령어 : `docker run --detach --label "color=blue" nginx`

이번엔 `color=blue` label을 달아뒀습니다.

컨테이너를 확인해볼까요?

```
ubuntu@ip-10-0-1-14:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS              PORTS     NAMES
670302b2287f   nginx     "/docker-entrypoint.…"   About a minute ago   Up About a minute   80/tcp    friendly_leakey
237e34821eea   nginx     "/docker-entrypoint.…"   12 minutes ago       Up 12 minutes       80/tcp    ecstatic_visvesvaraya
```

> 명령어 : `docker ps`

그리고, `overlay2` 디렉토리에는 R/W Layer만 추가된 걸 확인할 수 있습니다.

```
ubuntu@ip-10-0-1-14:~$ sudo ls -alt /var/lib/docker/overlay2
total 60
drwx--x---  5 root root  4096 Jun 17 06:15 e00fa3a0cd51eb127520ba0e2fc6cd507b0b8c6a374a26c42813dcac5190afad
drwx--x--- 13 root root 12288 Jun 17 06:15 .
drwx--x---  4 root root  4096 Jun 17 06:15 e00fa3a0cd51eb127520ba0e2fc6cd507b0b8c6a374a26c42813dcac5190afad-init
drwx------  2 root root  4096 Jun 17 06:15 l
drwx--x---  5 root root  4096 Jun 17 06:04 85790c7c79f6a9ed42c56001ecbcca73a247e3dae52e70642cfa1deb8717b345
drwx--x---  4 root root  4096 Jun 17 06:04 85790c7c79f6a9ed42c56001ecbcca73a247e3dae52e70642cfa1deb8717b345-init
drwx--x---  4 root root  4096 Jun 17 06:04 d060bd3b029045629a36a42774c68afc43716d9bfdbbb2bcca810e530262e604
drwx--x---  4 root root  4096 Jun 17 06:01 c77d0c3677b1d3d949038beef16bcf0dfe6b9da76c79efcf558463c937779022
drwx--x---  4 root root  4096 Jun 17 06:01 d0cd4377c363119f63c59a2d32db03e0deba873c11cc9f0885a9b343b57387a1
drwx--x---  4 root root  4096 Jun 17 06:01 5dc378c99eedd46fc3aca29e771c7a044b91ac71f4ebff2da0be8260da057350
drwx--x---  4 root root  4096 Jun 17 06:01 73b619a2efde2fe0bac0940fa6892ad5c7edc9fef14ba2b0a5605ce6ae4963c0
drwx--x---  3 root root  4096 Jun 17 06:01 fccb8e1376f1c76242fc79698262f7498b78c3c8090fc9f8b79bd29d6b8e460d
drwx--x--- 13 root root  4096 Jun 17 02:51 ..
```

> 명령어 : `sudo ls -alt /var/lib/docker/overlay2`

같은 이미지로 여러개의 컨테이너를 실행해도, R/O Layer는 공유하고 R/W Layer만 추가해서 만들어지네요.

다음 실습을 위해서 blue는 삭제할게요.

```
ubuntu@ip-10-0-1-14:~$ docker rm --force $(docker ps --filter "label=color=blue" --quiet)
670302b2287f
```

> 명령어 : `docker rm --force $(docker ps --filter "label=color=blue" --quiet)`

Docker & Kubernetes - [Hands-on] 02. Docker layers

이제 우리만의 새로운 이미지를 만들어 보겠습니다.
여러가지 방법이 있지만 이번 실습은 실행중인 컨테이너의 내용을 반영한 새로운 이미지를 만드는 것입니다.

앞에서 실행한 nginx를 활용하도록 하겠습니다.

먼저 잘 실행되고 있나 보구요.

```
ubuntu@ip-10-0-1-14:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS     NAMES
237e34821eea   nginx     "/docker-entrypoint.…"   20 minutes ago   Up 20 minutes   80/tcp    ecstatic_visvesvaraya
```

> **명령어** : `docker ps`

잘 살아있으면, 다음 명령으로 새로운 이미지를 만들어 보겠습니다.

```
ubuntu@ip-10-0-1-14:~$ docker commit $(docker ps --filter "label=color=red" --quiet) nginx:my-tag
sha256:58a92488d2a1246d6caa14d2b5b46e9999f2a5abffac86356188f128985288ef
```

> **명령어** : `docker commit $(docker ps --filter "label=color=red" --quiet) nginx:my-tag`

뭔가 복잡해 보이지만 별거 아닙니다.

`$(...)` 이 부분은 우리가 Label(`color=red`)을 붙인 컨테이너의 ID를 조회하는 명령입니다.
git의 `commit` 명령어 처럼, docker도 `commit` 명령을 이용해서 새로운 이미지를 만들 수 있습니다.

Docker & Kubernetes - [Hands-on] 02. Docker layers

이제 어떤 이미지가 있나 조회해볼까요?

```
ubuntu@ip-10-0-1-14:~$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED              SIZE
nginx        my-tag    58a92488d2a1   About a minute ago   142MB
nginx        latest    0e901e68141f   2 weeks ago          142MB
```

> 명령어 : `docker images`

방금 우리가 `docker commit` 명령으로 만든 `nginx:my-tag` 이미지가 보이네요.

`overlay2` 디렉토리는요?

```
ubuntu@ip-10-0-1-14:~$ sudo ls -alt /var/lib/docker/overlay2
total 56
drwx--x--- 12 root root 12288 Jun 17 06:26 .
drwx--x---  4 root root  4096 Jun 17 06:26 a17bce650b53c0a640af9ca29de030366978a37f2874e68ecc2fe5bd349db935
drwx------  2 root root  4096 Jun 17 06:26 l
drwx--x---  5 root root  4096 Jun 17 06:04 85790c7c79f6a9ed42c56001ecbcca73a247e3dae52e70642cfa1deb8717b345
drwx--x---  4 root root  4096 Jun 17 06:04 85790c7c79f6a9ed42c56001ecbcca73a247e3dae52e70642cfa1deb8717b345-init
drwx--x---  4 root root  4096 Jun 17 06:04 d060bd3b029045629a36a42774c68afc43716d9bfdbbb2bcca810e530262e604
drwx--x---  4 root root  4096 Jun 17 06:01 c77d0c3677b1d3d949038beef16bcf0dfe6b9da76c79efcf558463c937779022
drwx--x---  4 root root  4096 Jun 17 06:01 d0cd4377c363119f63c59a2d32db03e0deba873c11cc9f0885a9b343b57387a1
drwx--x---  4 root root  4096 Jun 17 06:01 5dc378c99eedd46fc3aca29e771c7a044b91ac71f4ebff2da0be8260da057350
drwx--x---  4 root root  4096 Jun 17 06:01 73b619a2efde2fe0bac0940fa6892ad5c7edc9fef14ba2b0a5605ce6ae4963c0
drwx--x---  3 root root  4096 Jun 17 06:01 fccb8e1376f1c76242fc79698262f7498b78c3c8090fc9f8b79bd29d6b8e460d
drwx--x--- 13 root root  4096 Jun 17 02:51 ..
```

> 명령어 : `sudo ls -alt /var/lib/docker/overlay2`

네, 여기도 하나가 더 생겼네요.

이제 실행중인 컨테이너까지 멈추고 삭제까지 해볼게요.

```
ubuntu@ip-10-0-1-14:~$ docker rm -f $(docker ps --filter "label=color=red" -q)
237e34821eea
```

> 명령어 : `docker rm -f $(docker ps --filter "label=color=red" -q)`

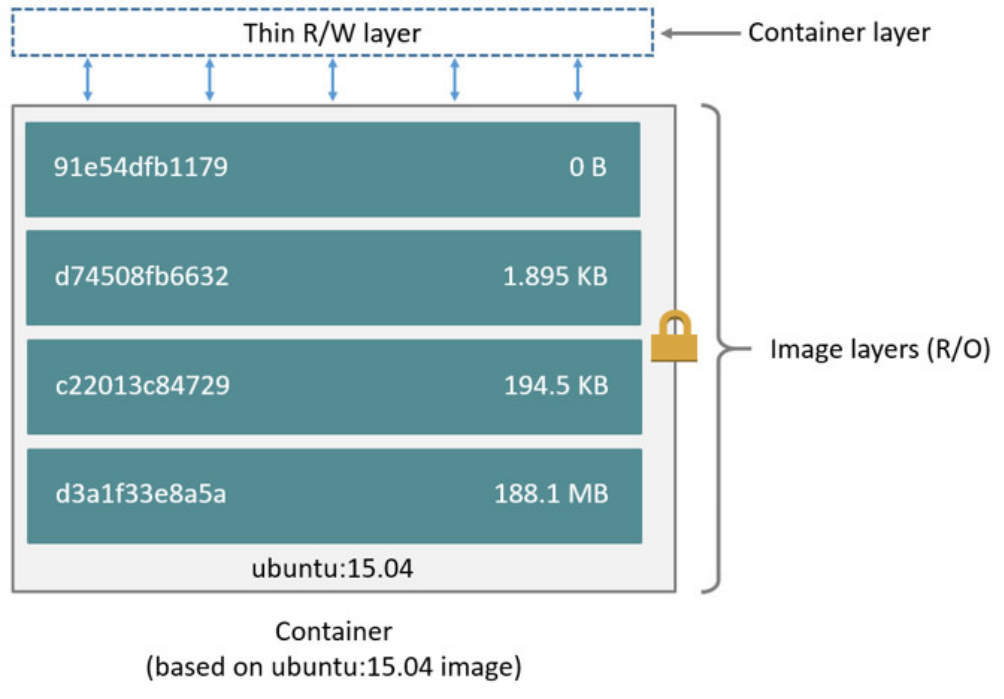
이제 `overlay2` 디렉토리는 어떻게 되었을까요?

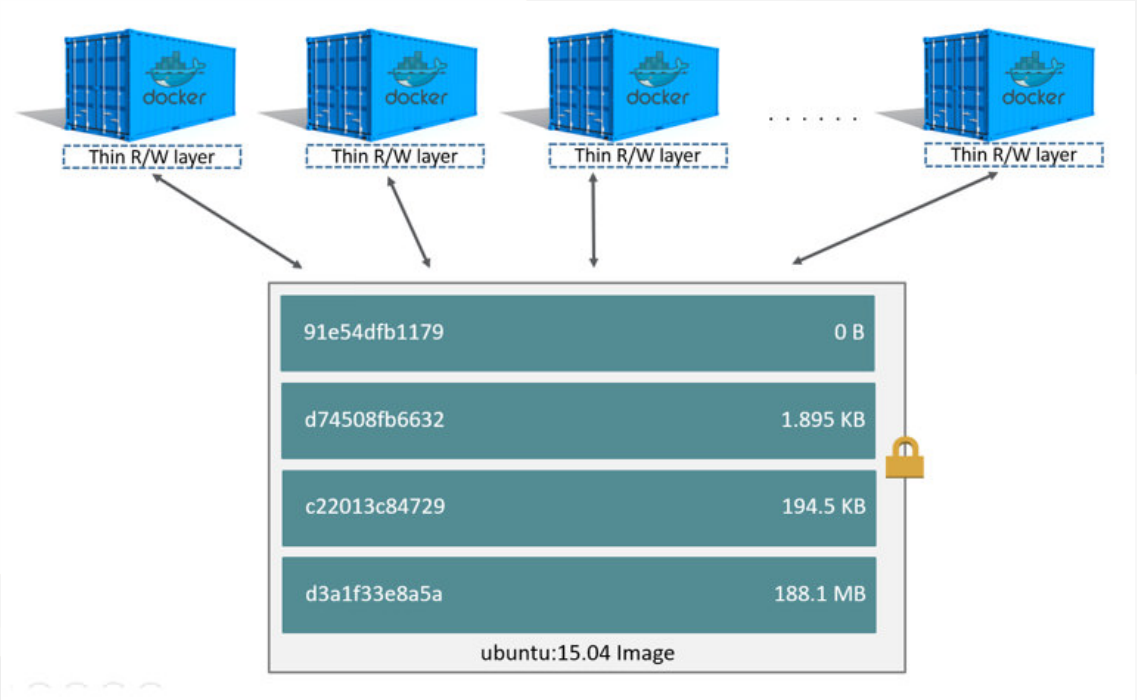
```
ubuntu@ip-10-0-1-14:~$ sudo ls -alt /var/lib/docker/overlay2
total 48
drwx--x--- 10 root root 12288 Jun 17 06:30 .
drwx------  2 root root  4096 Jun 17 06:30 l
drwx--x---  4 root root  4096 Jun 17 06:26 a17bce650b53c0a640af9ca29de030366978a37f2874e68ecc2fe5bd349db935
drwx--x---  4 root root  4096 Jun 17 06:04 d060bd3b029045629a36a42774c68afc43716d9bfdbbb2bcca810e530262e604
drwx--x---  4 root root  4096 Jun 17 06:01 c77d0c3677b1d3d949038beef16bcf0dfe6b9da76c79efcf558463c937779022
drwx--x---  4 root root  4096 Jun 17 06:01 d0cd4377c363119f63c59a2d32db03e0deba873c11cc9f0885a9b343b57387a1
drwx--x---  4 root root  4096 Jun 17 06:01 5dc378c99eedd46fc3aca29e771c7a044b91ac71f4ebff2da0be8260da057350
drwx--x---  4 root root  4096 Jun 17 06:01 73b619a2efde2fe0bac0940fa6892ad5c7edc9fef14ba2b0a5605ce6ae4963c0
drwx--x---  3 root root  4096 Jun 17 06:01 fccb8e1376f1c76242fc79698262f7498b78c3c8090fc9f8b79bd29d6b8e460d
drwx--x--- 13 root root  4096 Jun 17 02:51 ..
```

> 명령어 : `sudo ls -alt /var/lib/docker/overlay2`

처음 `docker pull` 해서 생성된 layer에, `docker commit` 해서 생성된 레이어만 하나 추가되어 있네요. (+1)

아래 그림을 다시한 번 떠올려 보면서 마무리 하겠습니다.





이번 실습은 여기까지 입니다.

_ (。 ˙ ‿ ˙ 。)