실습과 함께 완성해보는

도커 없이 컨테이너 만들기

5편

Sam.0

시작하기에 앞서 ...

본 컨텐츠는 앞편을 보았다고 가정하고 준비되었습니다.

원활한 이해 및 실습을 위하여 앞편을 먼저 보시기를 추천드립니다

*특히 **1,2편**은 마운트 네임스페이스와 밀접히 연결되므로 꼭 보세요*

:-)

1편 링크 클릭 *2편 링크 클릭* *3편 링크 클릭* *4편 링크 클릭*

실습을 위한 사전 준비 사항

실습 환경 구성하기 클릭

실습은 . . .

- 맥 환경에서 VirtualBox + Vagrant 기반으로 준비되었습니다
  - 맥 이외의 OS 환경도 괜찮습니다만
  - 원활한 실습을 위해서
  - “VirtualBox or VMware + Vagrant”는 권장드립니다.
- 실습환경 구성을 위한 Vagrantfile을 제공합니다.

실습을 위한 사전 준비 사항

Vagrantfile

- 오른쪽의 텍스트를 복사하신 후
- 로컬에 Vagrantfile로 저장하여 사용하면 됩니다.
- vagrant 사용법은 공식문서를 참고해 주세요 https://www.vagrantup.com/docs/index

실습 환경 구성하기 클릭

```
BOX_IMAGE = "bento/ubuntu-18.04"
HOST_NAME = "ubuntu1804"

$pre_install = <<-SCRIPT
  echo ">>>> pre-install <<<<<<"
  sudo apt-get update &&
  sudo apt-get -y install gcc &&
  sudo apt-get -y install make &&
  sudo apt-get -y install pkg-config &&
  sudo apt-get -y install libseccomp-dev &&
  sudo apt-get -y install tree &&
  sudo apt-get -y install jq &&
  sudo apt-get -y install bridge-utils

  echo ">>>> install go <<<<<<"
  curl -O https://storage.googleapis.com/golang/go1.15.7.linux-amd64.tar.gz > /dev/null 2>&1 &&
  tar xf go1.15.7.linux-amd64.tar.gz &&
  sudo mv go /usr/local/ &&
  echo 'PATH=$PATH:/usr/local/go/bin' | tee /home/vagrant/.bash_profile

  echo ">>>>> install docker <<<<<<"
  sudo apt-get -y install apt-transport-https ca-certificates curl gnupg-agent software-properties-common > /dev/null 2>&1 &&
  sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add - &&
  sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" &&
  sudo apt-get update &&
  sudo apt-get -y install docker-ce docker-ce-cli containerd.io > /dev/null 2>&1
SCRIPT

Vagrant.configure("2") do |config|

 config.vm.define HOST_NAME do |subconfig|
   subconfig.vm.box = BOX_IMAGE
   subconfig.vm.hostname = HOST_NAME
   subconfig.vm.network :private_network, ip: "192.168.104.2"
   subconfig.vm.provider "virtualbox" do |v|
     v.memory = 1536
     v.cpus = 2
   end
   subconfig.vm.provision "shell", inline: $pre_install
 end

end
```

실습을 위한 사전 준비 사항

실습 환경 구성하기 클릭

Vagrantfile 제공 환경

vagrant + virtual vm
ubuntu 18.04
docker 20.10.5 *도커 이미지 다운로드 및 컨테이너 비교를 위한 용도로 사용합니다.

기타 설치된 툴
~ tree, jq, brctl, … 등 실습을 위한 툴

실습 계정 (root)

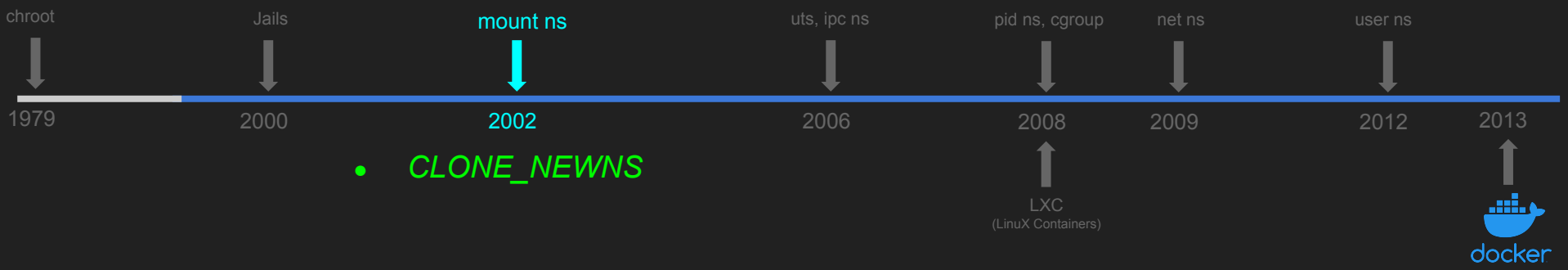
```
# sudo -Es
```

실습 폴더

```
# cd /tmp
```

Mount Namespace

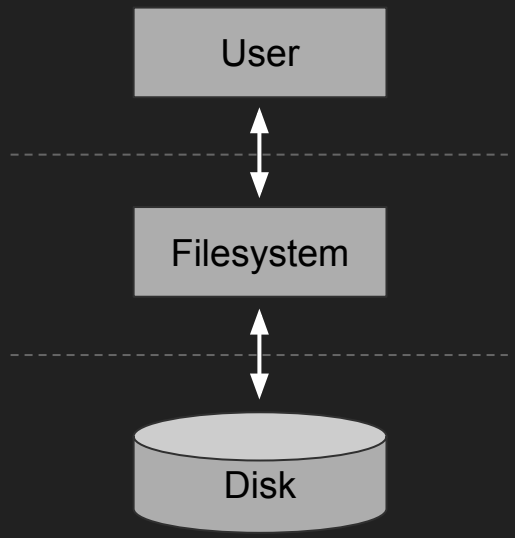
The “**mount namespace” of a *process*** is just the **set of mounted filesystems** that ***it sees***.



- *CLONE_NEWNS*

Filesystem

데이터를 저장하고 접근하고 사용/관리하기 위한 체계

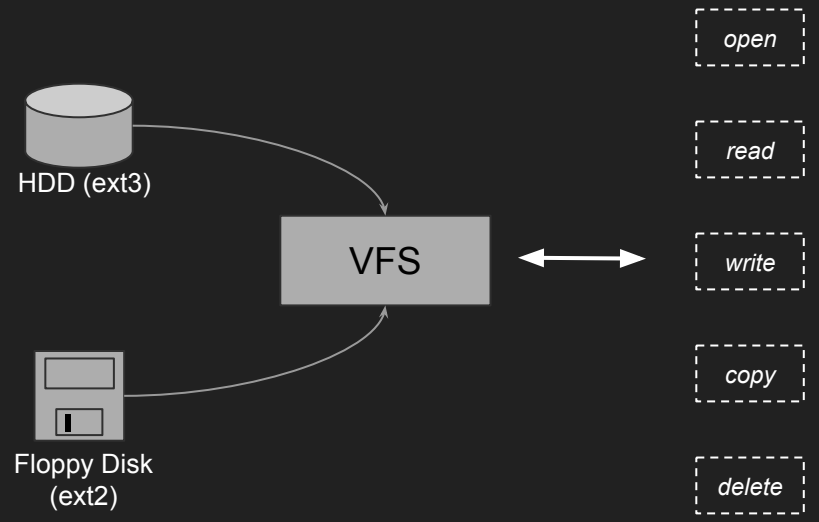


https://jiming.tistory.com/m/359

Linux Filesystem

Virtual Filesystem Layer

~ 파일시스템, 파일의 데이터 구조를 추상화
~ 파일시스템, 파일에 대한 오퍼레이션을 추상화



다양한 저장매체, 파일시스템에 상관없이 동작

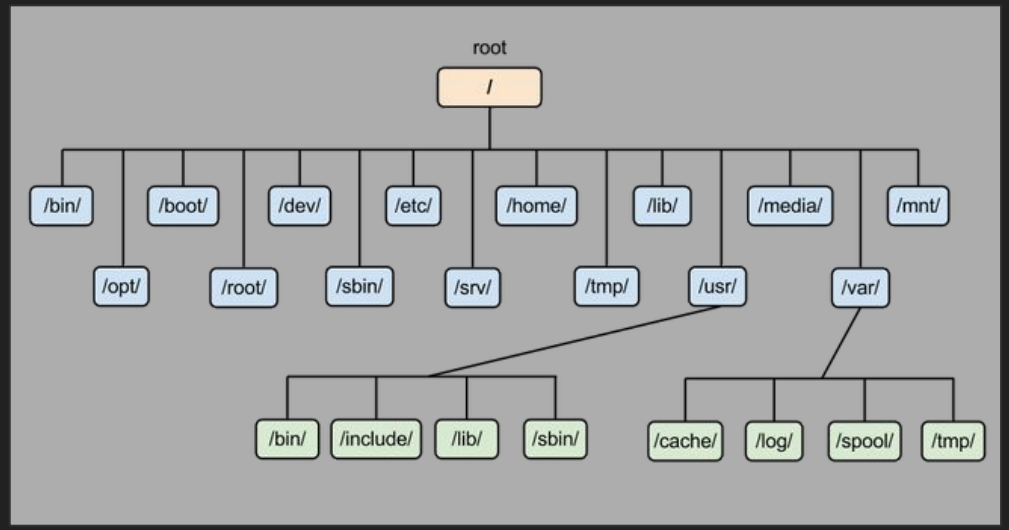
약 60여개 이상의 파일시스템 타입을 지원

Linux Kernel Development 3rd. Robert Love p.395

Linux Filesystem

'/' (root) 로 시작하는 hierarchy 구조

~ 파일, 디렉토리 및 제어정보
~ 파일시스템 동작 : 생성 / 삭제 / **마운트**

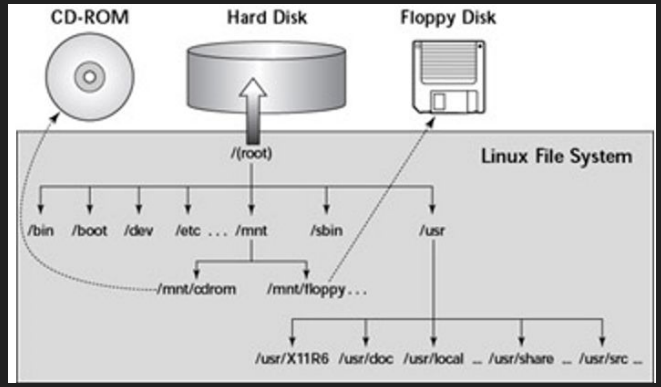


https://jiming.tistory.com/m/359 https://man7.org/linux/man-pages/man7/hier.7.html

Mount

To be a "root filesystem's subtree"

~ 다양한 저장 장치/파일시스템을 연결



"DOS, Windows 의 경우 C:, D: 드라이브 문자로 저장매체 구분이 되어 하드웨어의 세부 사항이 드러나 파일시스템 추상화가 깨지게(leaking) 된다" (Linux Kernel Development 3rd. p398)

https://jhnyang.tistory.com/m/12

Mount

To be a "root filesystem's subtree"

~ 다양한 저장 장치/파일시스템을 연결

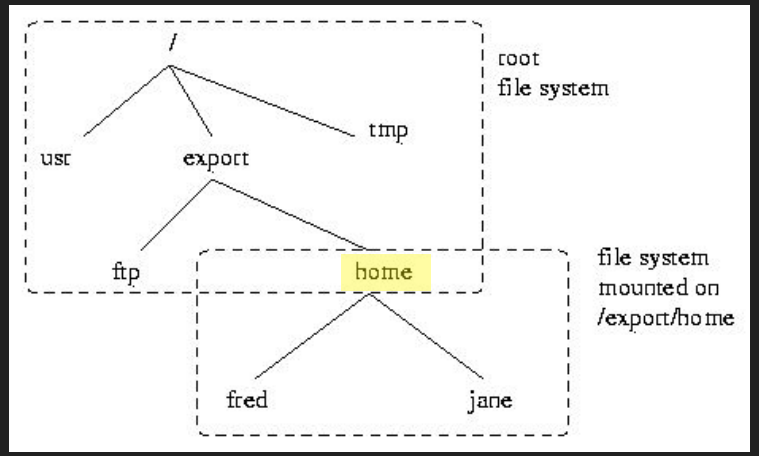
```
root@ubuntu1804:~# findmnt
TARGET                                   SOURCE                    FSTYPE     OPTIONS
/                                        /dev/mapper/vagrant--vg-root ext4     rw,relatime,errors=remount-ro,data=order
├─/sys                                   sysfs                     sysfs      rw,nosuid,nodev,noexec,relatime
│ ├─/sys/kernel/security                 securityfs                securityfs rw,nosuid,nodev,noexec,relatime
│ ├─/sys/fs/cgroup                       tmpfs                     tmpfs      ro,nosuid,nodev,noexec,mode=755
│ │ ├─/sys/fs/cgroup/unified             cgroup                    cgroup2    rw,nosuid,nodev,noexec,relatime
│ │ ├─/sys/fs/cgroup/systemd             cgroup                    cgroup     rw,nosuid,nodev,noexec,relatime,xattr,nc
│ │ ├─/sys/fs/cgroup/cpu,cpuacct         cgroup                    cgroup     rw,nosuid,nodev,noexec,relatime,cpu,cpuc
│ │ ├─/sys/fs/cgroup/blkio               cgroup                    cgroup     rw,nosuid,nodev,noexec,relatime,blkio
│ │ ├─/sys/fs/cgroup/memory              cgroup                    cgroup     rw,nosuid,nodev,noexec,relatime,memory
│ │ ├─/sys/fs/cgroup/net_cls,net_prio    cgroup                    cgroup     rw,nosuid,nodev,noexec,relatime,net_cls,
│ │ ├─/sys/fs/cgroup/freezer             cgroup                    cgroup     rw,nosuid,nodev,noexec,relatime,freezer
│ │ ├─/sys/fs/cgroup/hugetlb             cgroup                    cgroup     rw,nosuid,nodev,noexec,relatime,hugetlb
│ │ ├─/sys/fs/cgroup/cpuset              cgroup                    cgroup     rw,nosuid,nodev,noexec,relatime,cpuset
│ │ ├─/sys/fs/cgroup/perf_event          cgroup                    cgroup     rw,nosuid,nodev,noexec,relatime,perf_eve
│ │ ├─/sys/fs/cgroup/pids                cgroup                    cgroup     rw,nosuid,nodev,noexec,relatime,pids
│ │ ├─/sys/fs/cgroup/rdma                cgroup                    cgroup     rw,nosuid,nodev,noexec,relatime,rdma
│ │ └─/sys/fs/cgroup/devices             cgroup                    cgroup     rw,nosuid,nodev,noexec,relatime,devices
│ ├─/sys/fs/pstore                       pstore                    pstore     rw,nosuid,nodev,noexec,relatime
│ ├─/sys/kernel/config                   configfs                  configfs   rw,relatime
│ ├─/sys/kernel/debug                    debugfs                   debugfs    rw,relatime
│ └─/sys/fs/fuse/connections             fusectl                   fusectl    rw,relatime
├─/proc                                  proc                      proc       rw,nosuid,nodev,noexec,relatime
│ └─/proc/sys/fs/binfmt_misc             systemd-1                 autofs     rw,relatime,fd=25,pgrp=1,timeout=0,minpr
├─/dev                                   udev                      devtmpfs   rw,nosuid,relatime,size=730484k,nr_inode
│ ├─/dev/pts                             devpts                    devpts     rw,nosuid,noexec,relatime,gid=5,mode=620
│ ├─/dev/shm                             tmpfs                     tmpfs      rw,nosuid,nodev
│ ├─/dev/hugepages                       hugetlbfs                 hugetlbfs  rw,relatime,pagesize=2M
│ └─/dev/mqueue                          mqueue                    mqueue     rw,relatime
├─/run                                   tmpfs                     tmpfs      rw,nosuid,noexec,relatime,size=152472k,m
```

http://tinf2.vub.ac.be/~dvermeir/manuals/uintro/uintro.html

Mount

To be a "root filesystem's subtree"

~ mount point
~ 프로세스별 filesystem mount 가능 (mount namespace)



Linux Kernel Development 3rd. Robert Love p.395 http://tinf2.vub.ac.be/~dvermeir/manuals/uintro/uintro.html

Mount

mount -t *[fs_type] [device_name] [dir - mount point]*

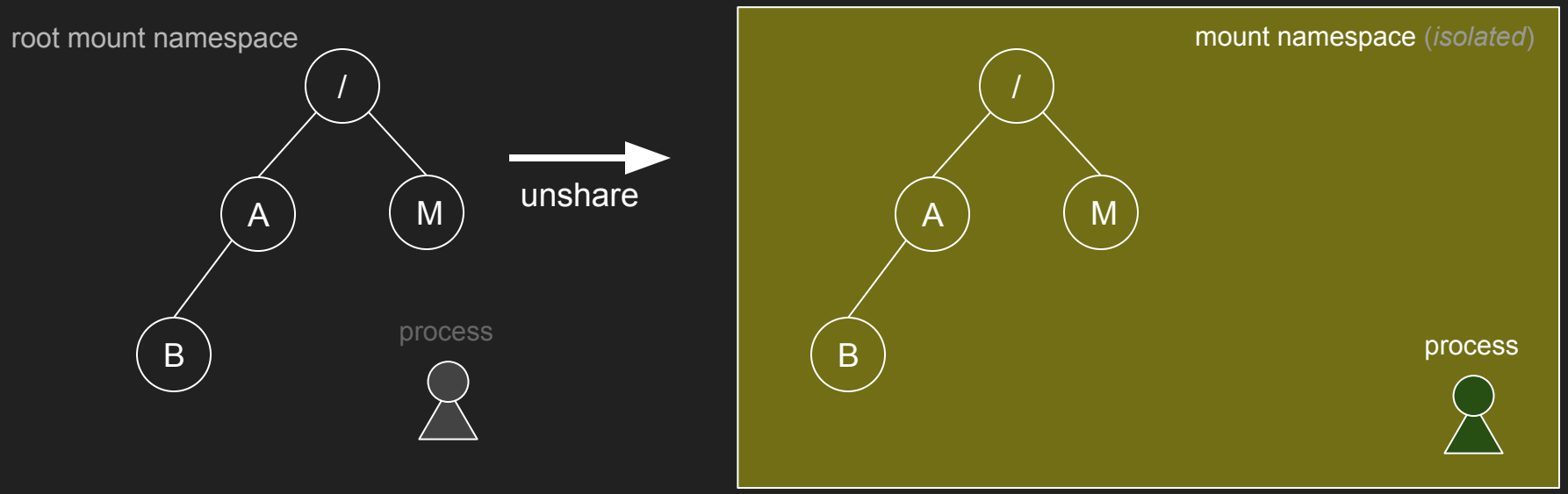
-o : mount option 사용 ex) -o size=1m
-t : filesystem type ex) -t tmpfs

참고)
* tmpfs : a virtual memory filesystem
* /proc/filesystems 에서 지원하는 filesystem type 조회 가능

http://tinf2.vub.ac.be/~dvermeir/manuals/uintro/uintro.htm

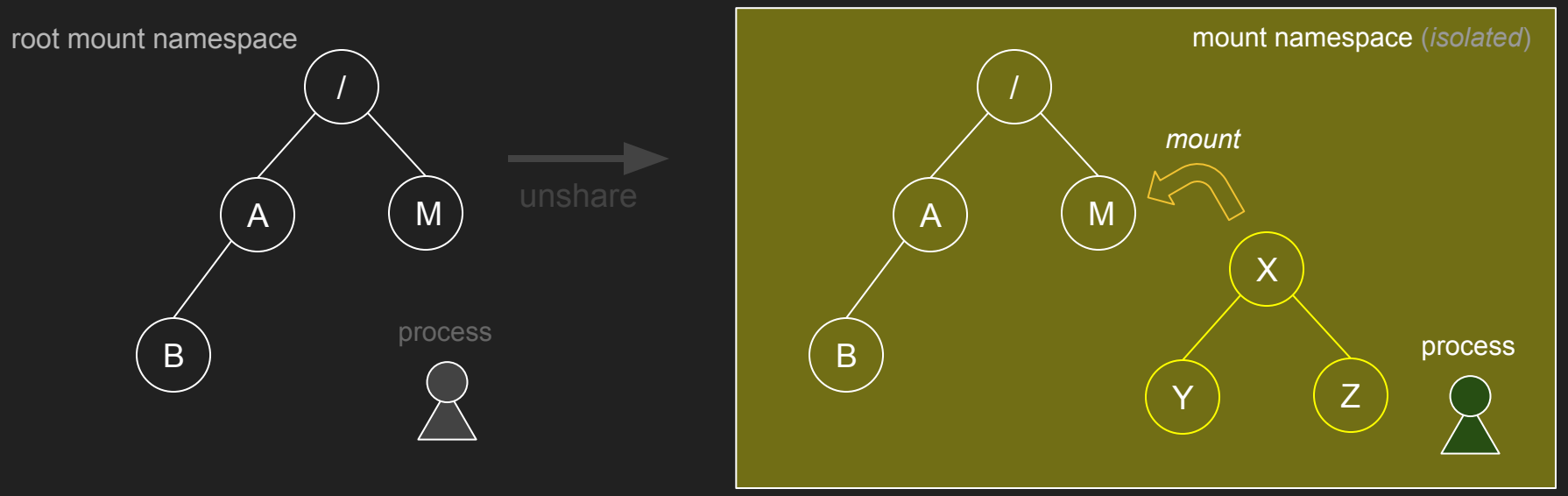
Mount Namespace

The “**mount namespace” of a *process*** is just the **set of mounted filesystems** that ***it sees***.



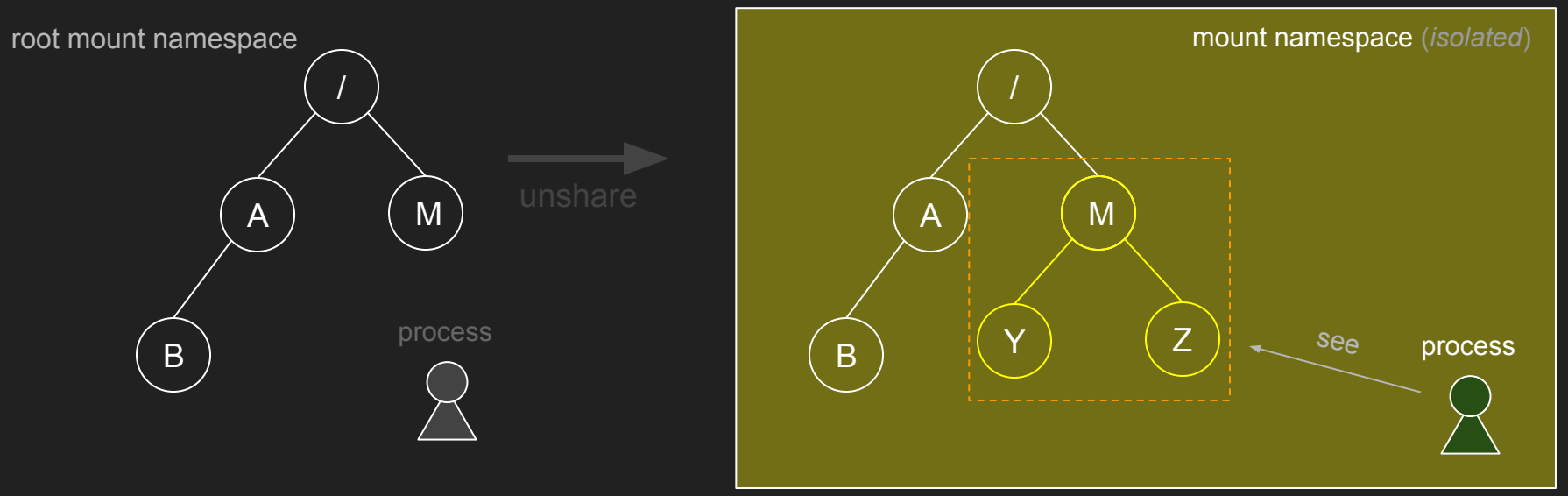
Mount Namespace

The “**mount namespace” of a *process*** is just the **set of mounted filesystems** that ***it sees***.



Mount Namespace

The “**mount namespace” of a *process*** is just the **set of mounted filesystems** that ***it sees***.

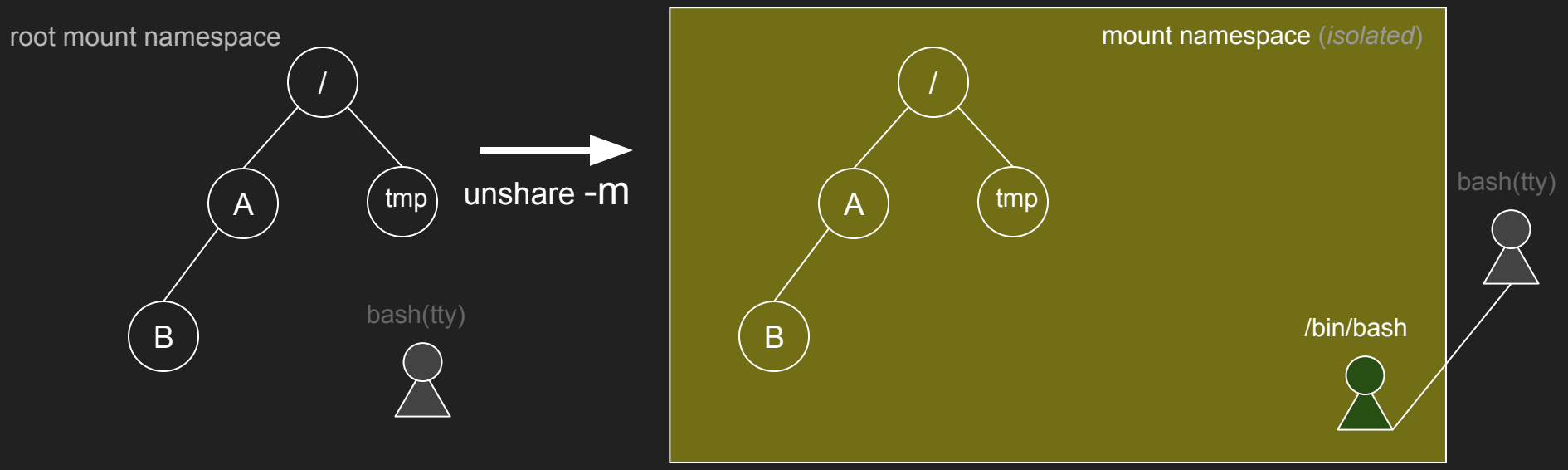


(실습1) unshare mount namespace

unshare 커맨드를 사용하여 mount namespace 를 생성해 보자

(실습1) unshare mount namespace

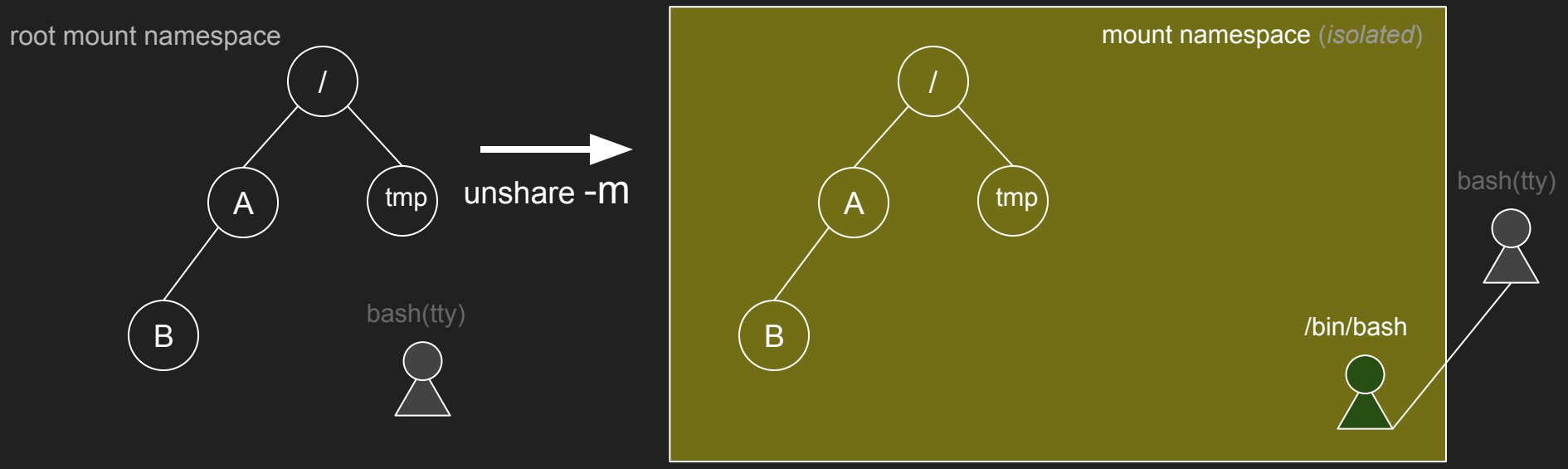
```
# unshare -m /bin/bash
```



(실습1) unshare mount namespace

```
# diff /proc/mounts /proc/$$/mounts
```

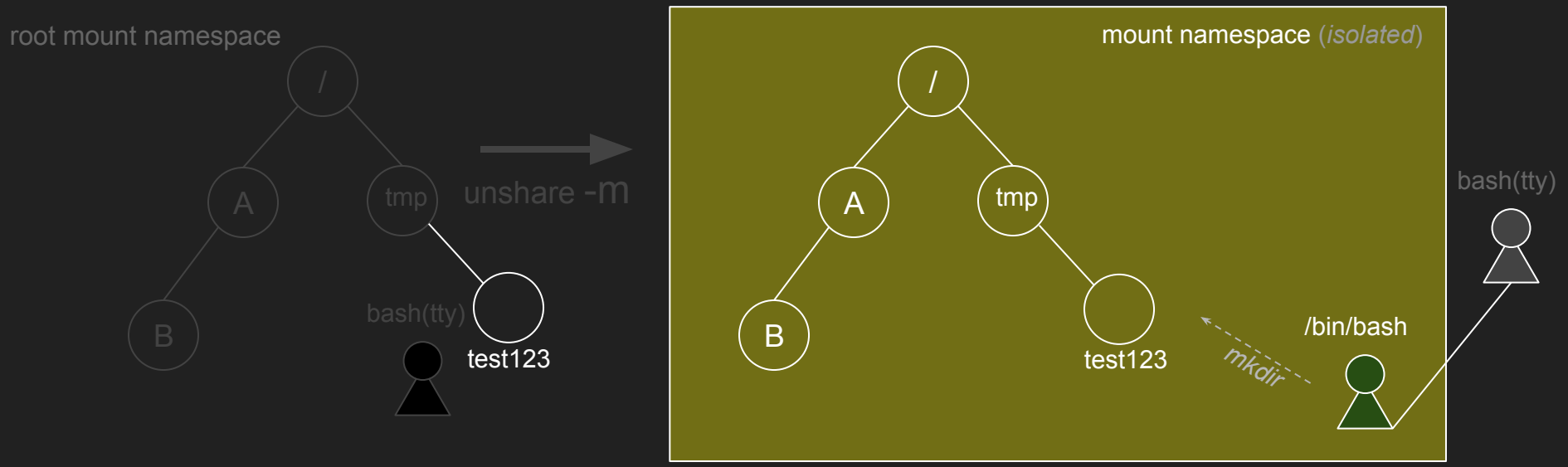
부모 프로세스의 mounts 를 복제



(실습1) unshare mount namespace

```
# mkdir /tmp/test123
```

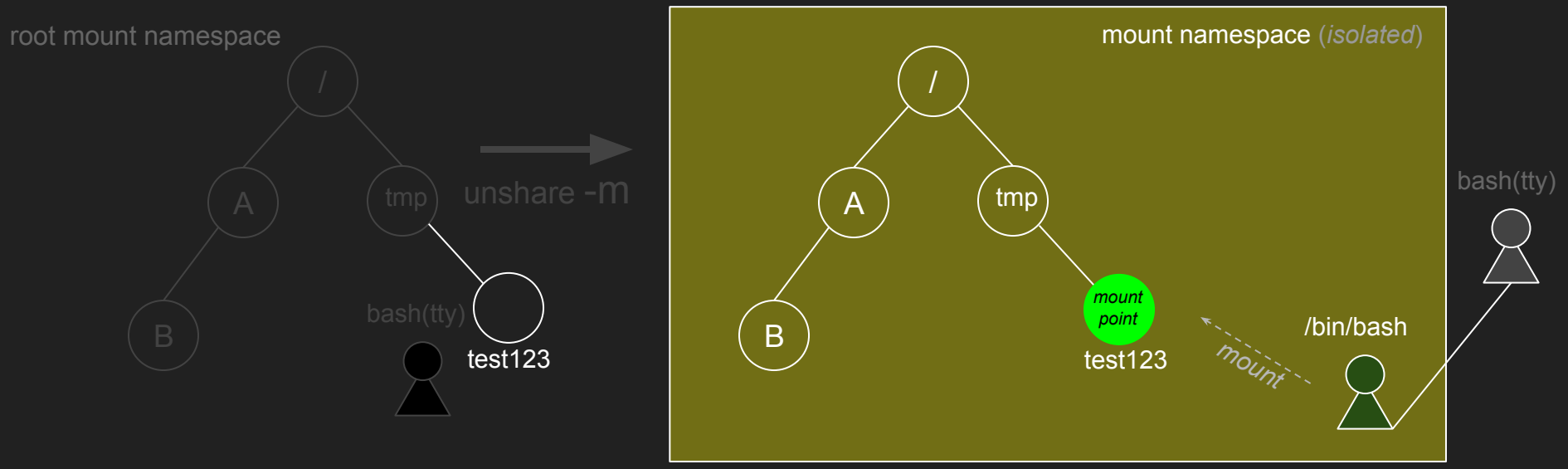
양 쪽 다 test123 디렉토리가 생성됨 (*root filesystem* 이 같기 때문)



(실습1) unshare mount namespace

```
# mount -o size=1m -t tmpfs tmpfs /tmp/test123
```

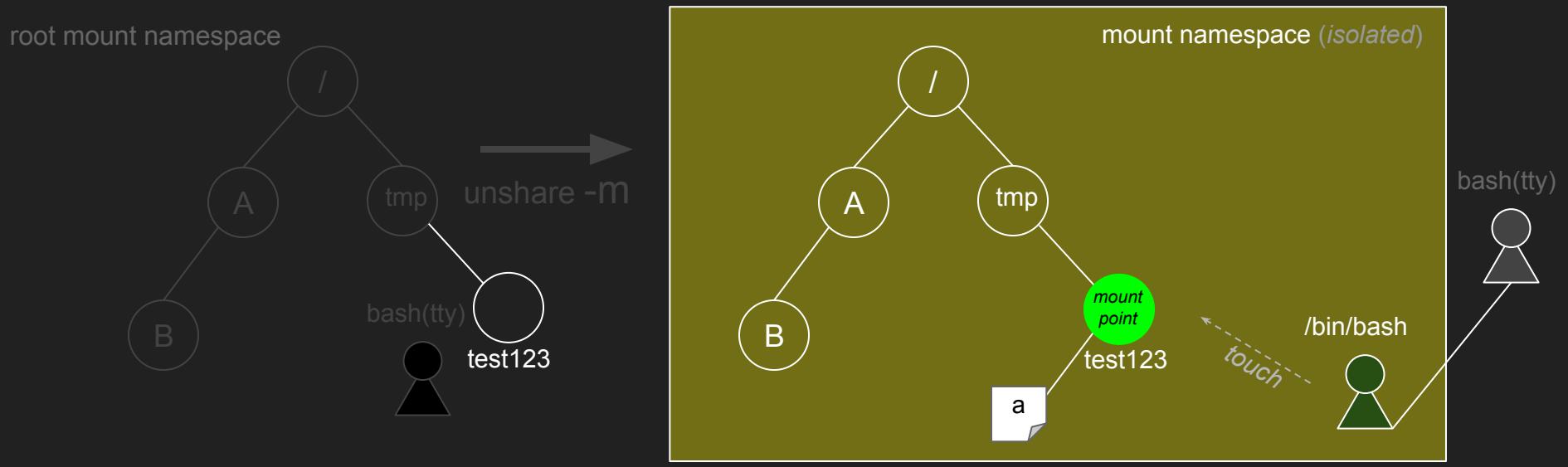
But, mount는 namespace 안에서만 적용됨



(실습1) unshare mount namespace

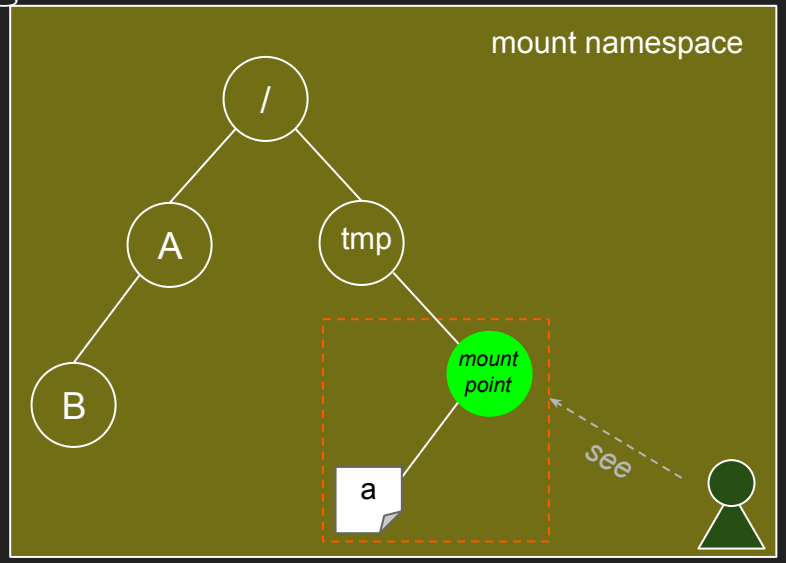
touch /tmp/test123/a

생성한 파일(a)도 namespace안에서만 보임



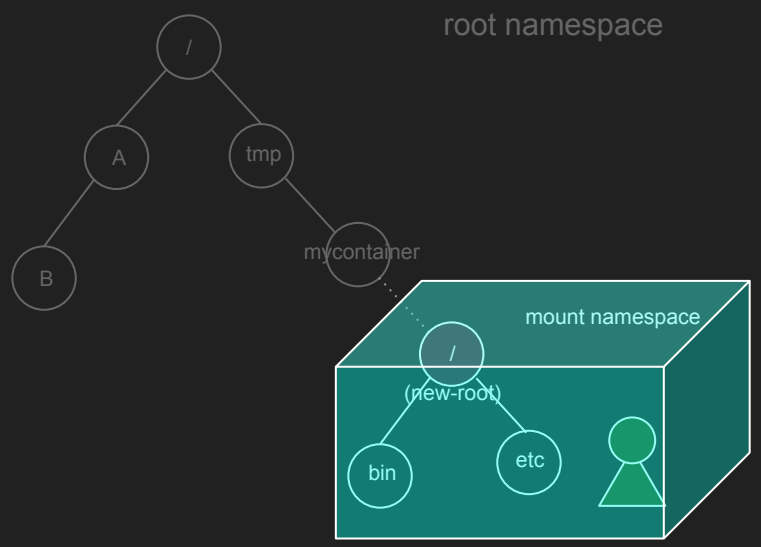
mount namespace는 “per-process mount isolation”을 제공

→ 프로세스에 격리된 “파일시스템 마운트"를 제공



(실습2) *per-process root filesystem isolation*

nginx 컨테이너를 실행해 보자



(실습2) *per-process root filesystem isolation*

터미널 #1

```
# mkdir /tmp/mycontainer

# cd /tmp/mycontainer

# mkdir new-root

# docker export $(docker create nginx:latest) | tar -C new-root -xvf -

# tree -L 1 new-root
```

(실습2) *per-process root filesystem isolation*

터미널 #1

```
# unshare -m /bin/bash

# mount --bind new-root new-root

# cd new-root

# mkdir orig-root

# pivot_root . orig-root

# cd /

# ls
```

→ 별도 디렉토리를 만들어서 mount point로 잡는 것이 host와 연결고리를 끊을 수 있음 (bind mount는 기존 파일시스템과 연결됨)

** 여기서는 편의상 bind mount 사용*

- *Bind mounts : mount --bind olddir newdir 현재 파일시스템의 olddir을 새로운 파일시스템의 newdir로 mount 하는 것으로 "Bind mount"의 디렉토리 및 파일은 원본과 동일하며 두 보기가 동일한 데이터를 표시하기 때문에 한 쪽의 수정은 즉시 다른 쪽에 반영됨 예시) FTP 특정 사용자 폴더에 호스트 상의 다른 경로의 폴더 제공*

(실습2) *per-process root filesystem isolation*

터미널 #1

```
# /docker-entrypoint.sh nginx -g "daemon off;"
```

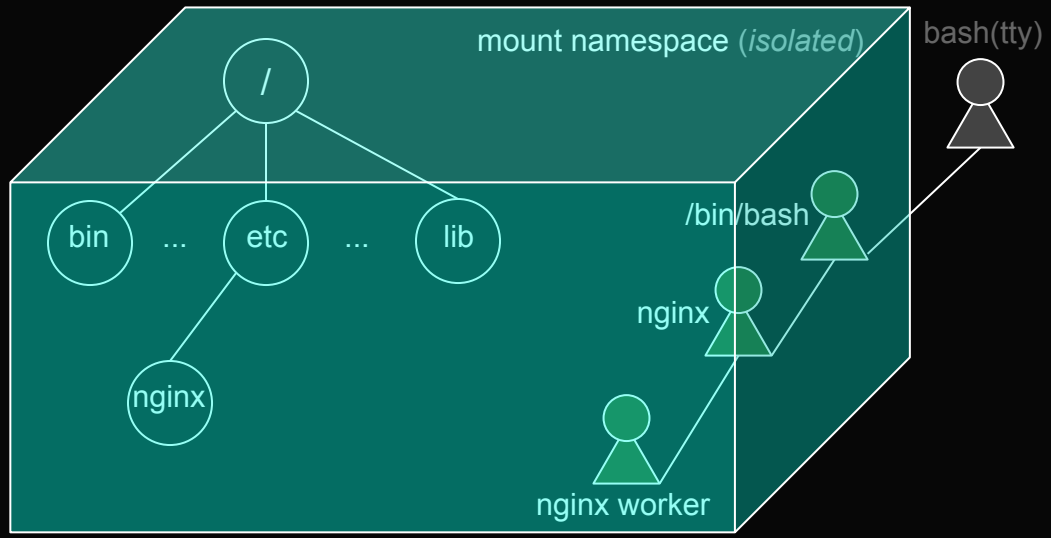
터미널 #2 (호스트)

```
# curl localhost

# ps -ef | grep nginx
9235

# lsns -t mnt -p 9235
4026532476

# lsns 4026532476
```

(실습2) *per-process root filesystem isolation*

lsns 4026532476

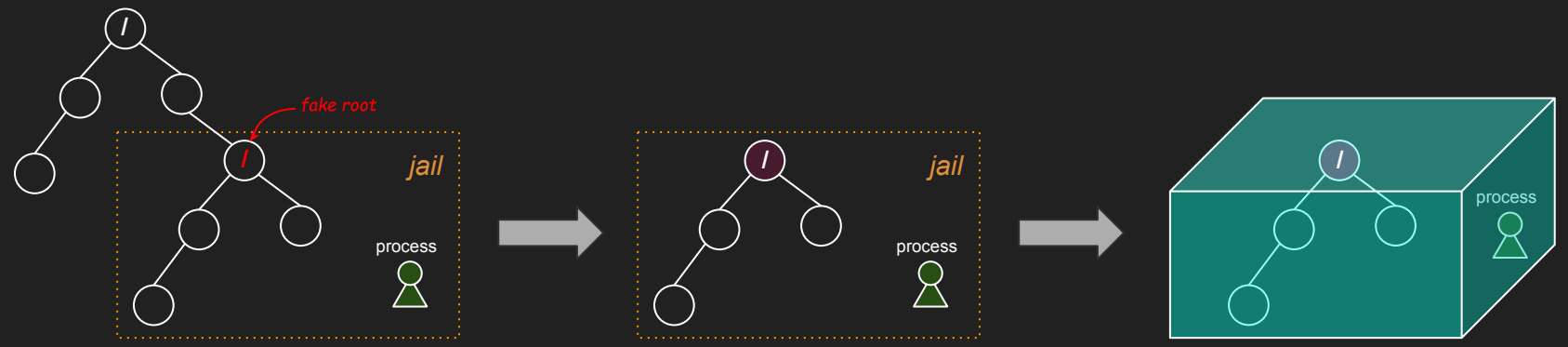
```
 PID  PPID USER            COMMAND
9223  8922 root            /bin/bash
9235  9223 root            └─nginx: master process nginx -g daemon off
9251  9235 systemd-resolve   └─nginx: worker proces
```



(정리)

컨테이너 ?

per-process root (/) filesystem isolation



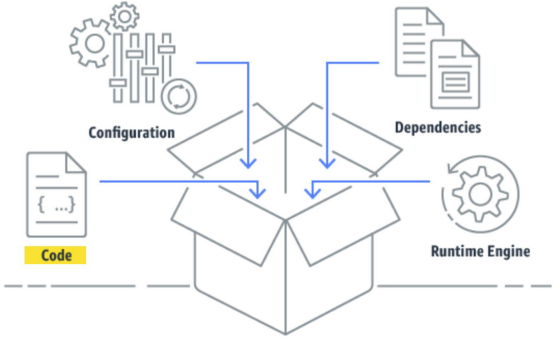
chroot

pivot_root

mount namespace

여기서 nginx 이미지는 nginx 프로세스 실행에 필요한 모든 것을 담고 있는데요 ...

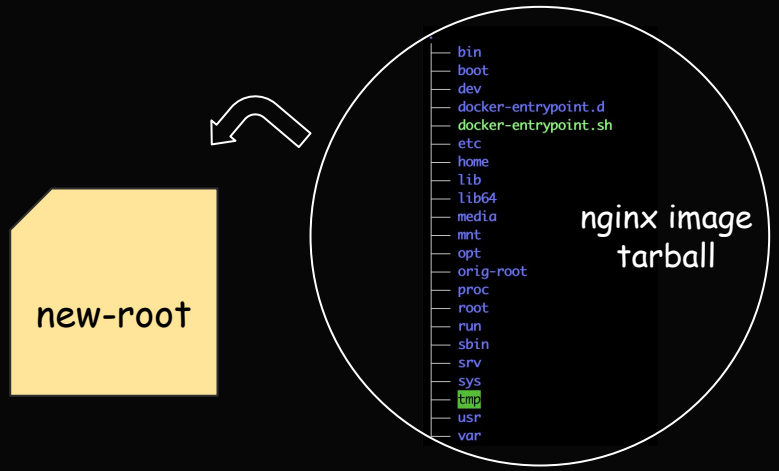
What is 컨테이너 ? **"포장 (Packaging)"**



실습에서 사용한 nginx 이미지 용량

터미널 #1

```
# du -sh /tmp/mycontainer/new-root

142M  /tmp/mycontainer/new-root
```

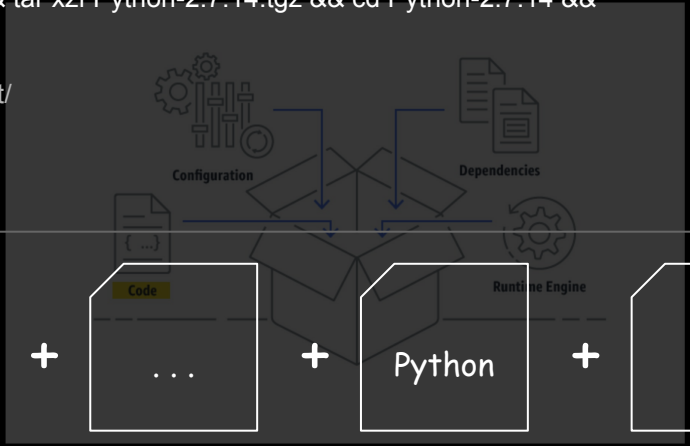


어떻게 만드느냐에 따라 이미지 용량은 천차만별 *bigger and bigger . . .*

Dockerfile

```
FROM ubuntu:xenial
RUN apt update
RUN apt -y install sysstat
RUN apt-get -y install build-essential checkinstall
RUN apt-get -y install libreadline-gplv2-dev libncursesw5-dev libssl-dev libsqlite3-dev tk-dev libgdbm-dev libc6-dev libbz2-dev
RUN apt-get -y install wget
RUN cd /usr/src && wget https://www.python.org/ftp/python/2.7.14/Python-2.7.14.tgz && tar xzf Python-2.7.14.tgz && cd Python-2.7.14 &&
./configure --enable-optimizations && make altinstall && cp python /usr/bin/
COPY docker_inspect_json.sh parse_pidstat.py start_pidstat_k8sbrz.sh /breeze_pidstat/
WORKDIR /breeze_pidstat
CMD ["bash","start_pidstat_k8sbrz.sh",".."]
```

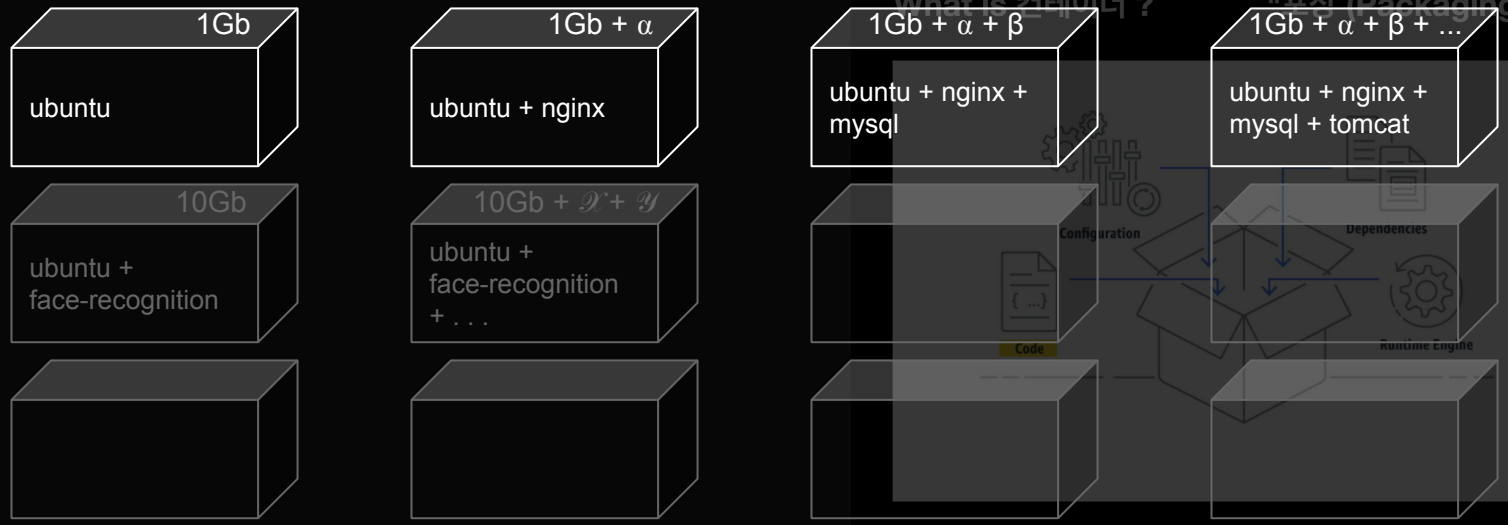
What is 컨테이너 ? **"포장 (Packaging)"**

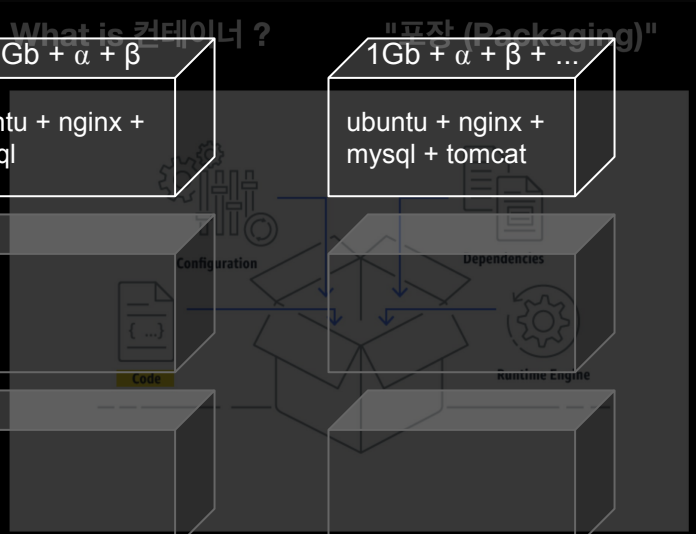


ubuntu + sysstat + . . . + wget + . . . + Python + . . .

해결해야 할 문제가 하나 더 늘어남 ~ 이미지 중복, 효율화

이미지 중복 → 저장/유통/관리/보안 ... ~ 비용





Union Filesystem

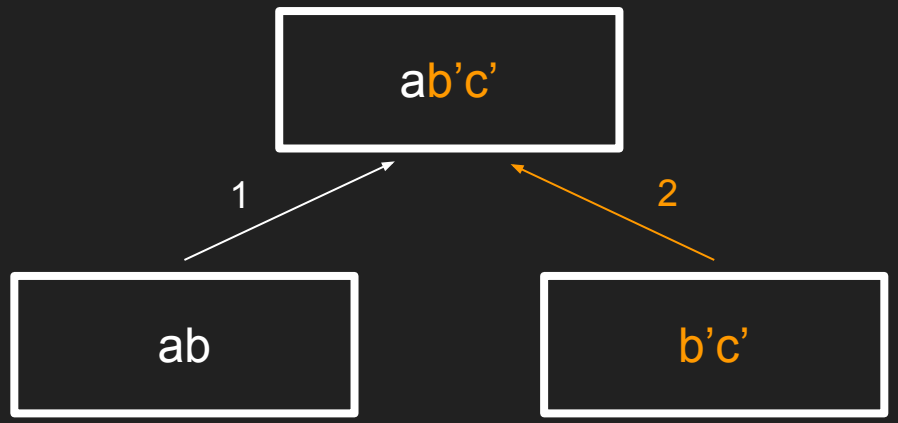
Union mount : 복수의 filesystem을 하나로 mount 하는 기능

특징

- 두 파일시스템에 동일한 파일이 있는 경우
  → 나중에 마운트되는 파일시스템의 파일을 오버레이함

- 하위 파일시스템에 대한 쓰기 작업 시
  → Cow. 복사본을 생성하여 수행 (원본 유지)

- 일명, “상속”파일시스템으로 불림



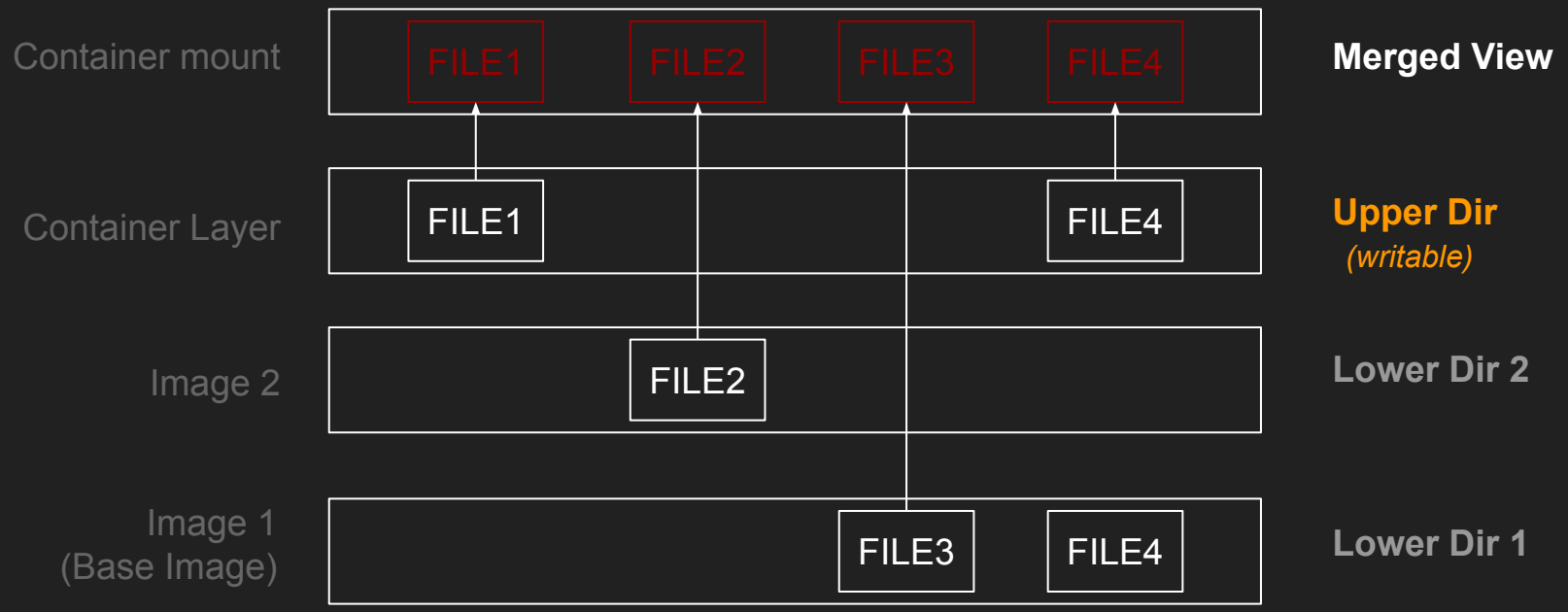
Union Filesystem

UnionFS ~ 초창기 구현체

AUFS ~ 코드가독성 문제 등으로 주류 커널에 통합 안됨

OverlayFS ~ 주류 커널에 통합 → 현재는 **OverlayFS2** 를 사용

OverlayFS2



Merged Dir : 통합뷰
Upper Dir : Writable. 컨테이너에서의 변경된 내용이 쓰이는 레이어
Lower Dir : Read Only, 기존 이미지 영역
*Work Dir : “atomic action”을 보장하기 위해 merged에 반영되기 전에 파일을 준비하는데 사용됨

(실습3) 오버레이 파일시스템

터미널 #1

```
# mkdir rootfs; cd rootfs

# mkdir image1 image2 container work merge

# touch image1/a image1/b image2/c
```

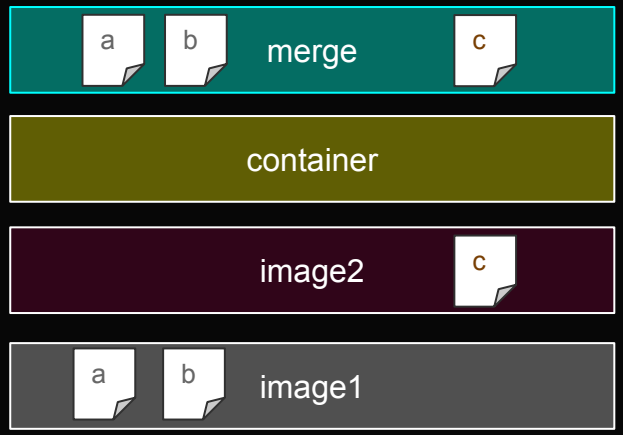
https://windsock.io/the-overlay-filesystem/

(실습3) 오버레이 파일시스템

터미널 #1

```
# mount -t overlay overlay -o lowerdir=image2:image1,upperdir=container,workdir=work merge
```

mount point

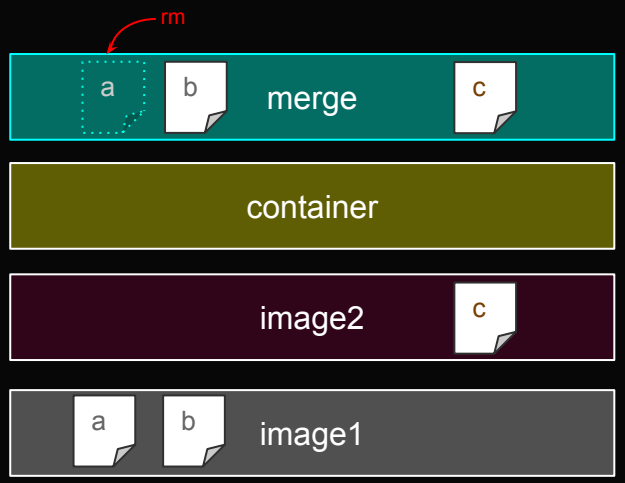


https://windsock.io/the-overlay-filesystem/

(실습3) 오버레이 파일시스템

터미널 #1

```
root@ubuntu1804:/tmp/rootfs# rm -f merge/a
root@ubuntu1804:/tmp/rootfs# tree -I work
```



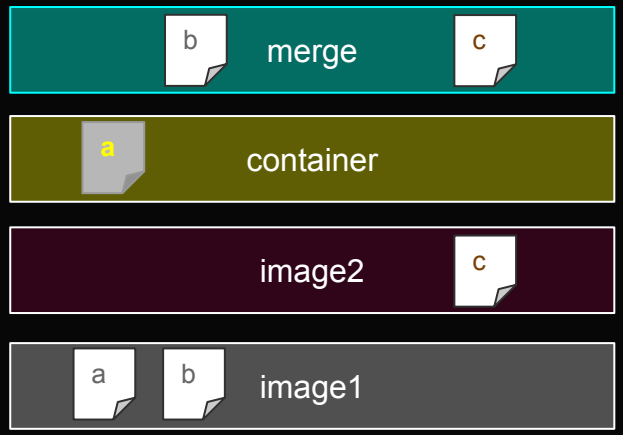
(실습3) 오버레이 파일시스템

터미널 #1

```
root@ubuntu1804:/tmp/rootfs# rm -f merge/a
root@ubuntu1804:/tmp/rootfs# tree -I work
.
├── container
│   └── a
├── image1
│   ├── a
│   └── b
├── image2
│   └── c
└── merge
    ├── b
    └── c
```

whiteout

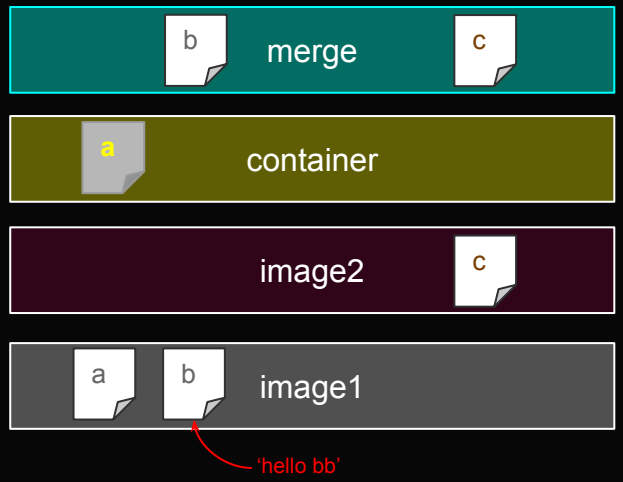
변경이 발생하면 **upper (RW)** 에 기록이 됨
lower는 변경이 일어나지 않음 **(RO)**



(실습3) 오버레이 파일시스템

터미널 #1

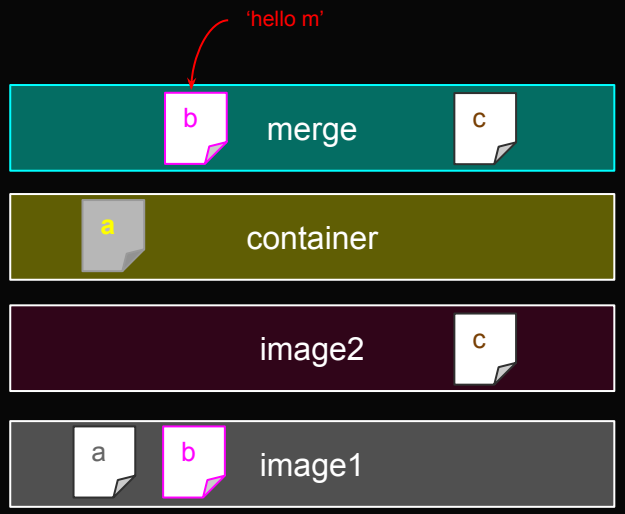
```
# echo 'hello bb' > image1/b

# cat merge/b
```



(실습3) 오버레이 파일시스템

터미널 #1

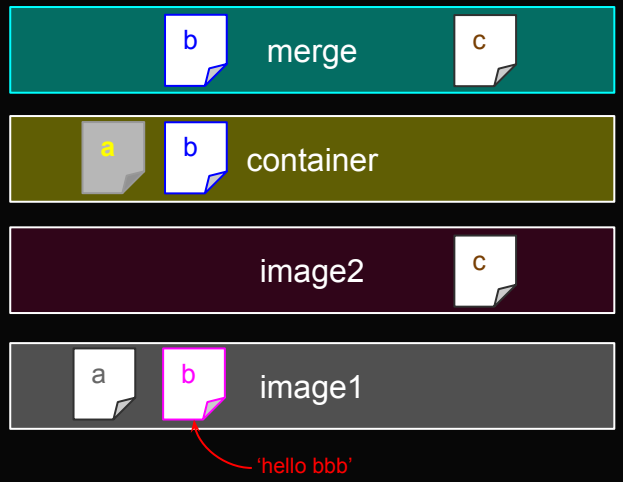
```
# echo 'hello m' > merge/b

# cat merge/b
```



(실습3) 오버레이 파일시스템

터미널 #1

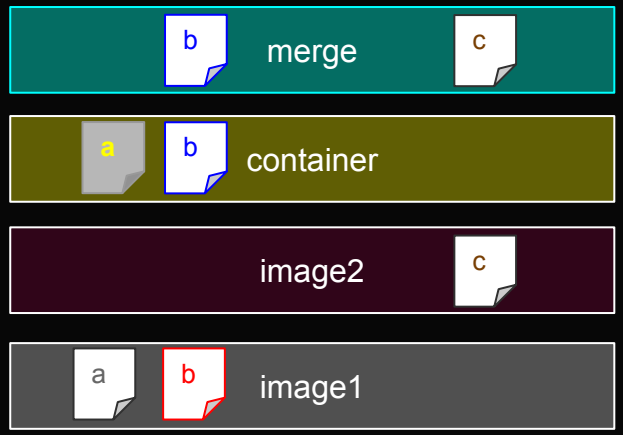
```
# echo 'hello bbb' > image1/b

# cat merge/b
```



(실습3) 오버레이 파일시스템

터미널 #1

```
# echo 'hello bb' > image1/b; cat merge/b

# echo 'hello m' > merge/b; cat merge/b        ← CoW (copy-on-write)

# echo 'hello bbb' > image1/b; cat merge/b  ← 더이상 변경 사항이 merge/b에 반영되지 않음
```



lower layer (image1, image2) : "RO" 인데 write 가능하네?

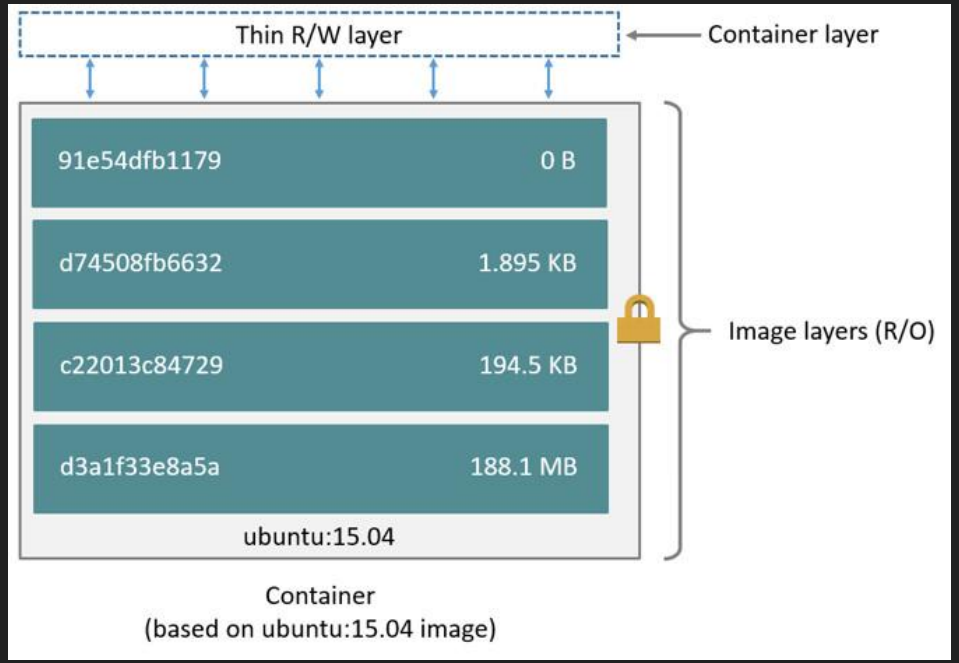
컨테이너 레이어 구조

컨테이너 레이어 (Merged, Upper Dir)

이미지 레이어 (Lower Dir)

* Union mount를 지원하는 overlay2 드라이버 사용

```
# docker info | grep Storage
. . .
  Storage Driver: overlay2
. . .
```

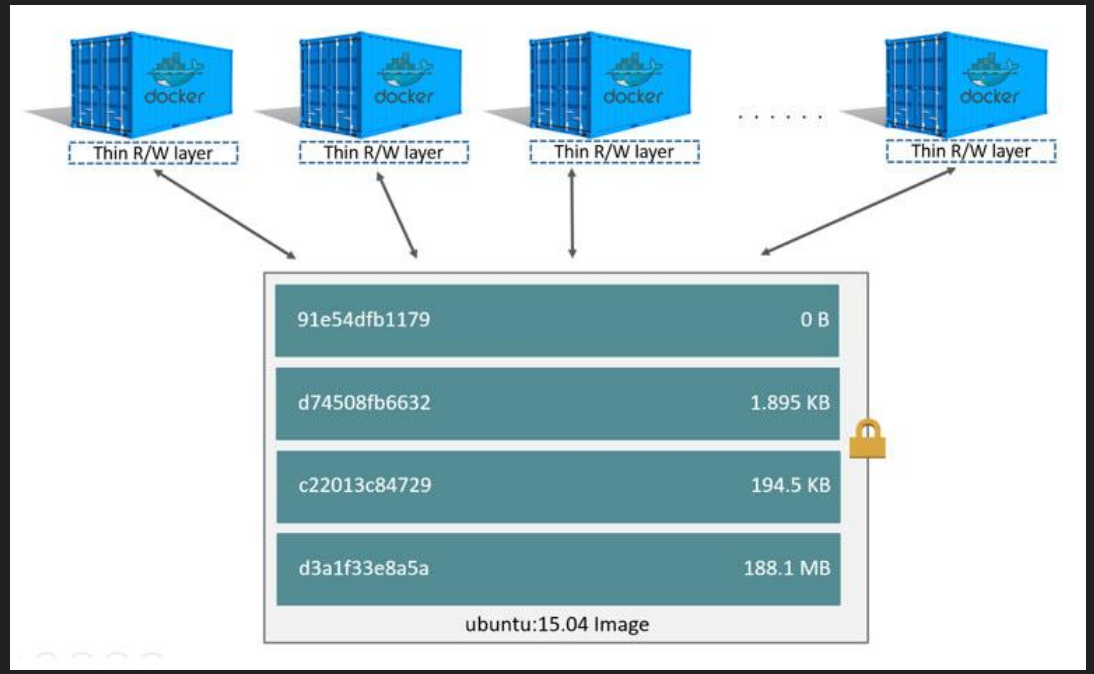


https://docs.docker.com/storage/storagedriver/

컨테이너 레이어 구조

동일한 이미지는 여러 컨테이너가 공유

CoW (copy-on-write) 전략



https://docs.docker.com/storage/storagedriver/

(실습4) 컨테이너 레이어 구조

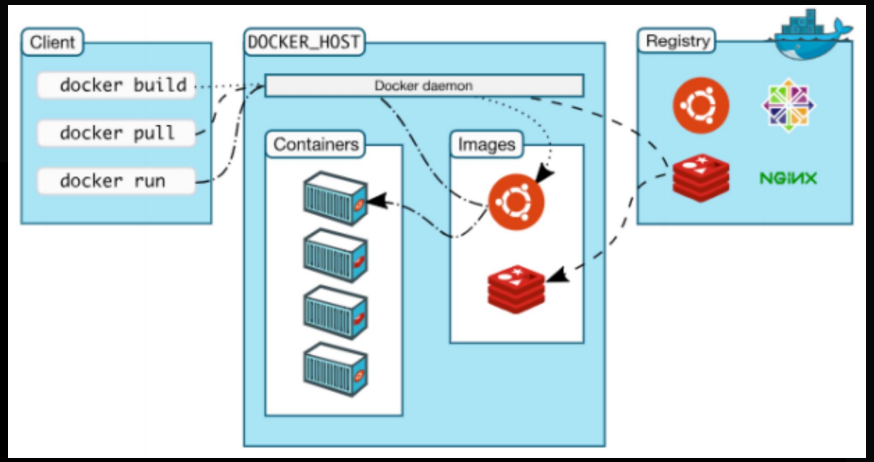
터미널 #1

```
# docker pull nginx:latest
latest: Pulling from library/nginx
f7ec5a41d630: Pull complete
aa1efa14b3bf: Pull complete
b78b95af9b17: Pull complete
c7d6bca2b8dc: Pull complete
cf16cd8e71e0: Pull complete
0241c68333ef: Pull complete
Digest: sha256:75a55d33ecc73c2a242450a9f1cc858499d468f077ea942867e662c247b5e412
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
```

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# docker pull nginx:latest
latest: Pulling from library/nginx
f7ec5a41d630: Pull complete
aa1efa14b3bf: Pull complete
b78b95af9b17: Pull complete
c7d6bca2b8dc: Pull complete
cf16cd8e71e0: Pull complete
0241c68333ef: Pull complete
Digest: sha256:75a55d33ecc73c2a242450a9f1cc858499d468f077ea942867e662c247b5e412
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
```



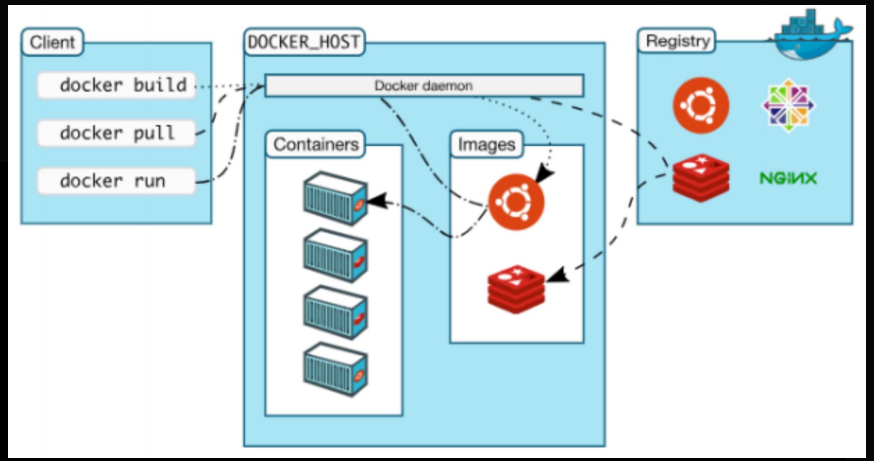
docker pull nginx:latest
docker pull index.docker.io/nginx:latest ← - - - - Registry: https://index.docker.io/v1/

https://www.44bits.io/ko/post/how-docker-image-work

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# docker pull nginx:latest
latest: Pulling from library/nginx
f7ec5a41d630: Pull complete
aa1efa14b3bf: Pull complete
b78b95af9b17: Pull complete
c7d6bca2b8dc: Pull complete
cf16cd8e71e0: Pull complete
0241c68333ef: Pull complete
Digest: sha256:75a55d33ecc73c2a242450a9f1cc858499d468f077ea942867e662c247b5e412
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
```



docker pull nginx:latest
docker pull nginx@sha256:75a55d33ecc73c2a242450a9f1cc858499d468f077ea942867e662c24

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# docker pull nginx:latest
latest: Pulling from library/nginx
f7ec5a41d630: Pull complete
aa1efa14b3bf: Pull complete
b78b95af9b17: Pull complete
c7d6bca2b8dc: Pull complete
cf16cd8e71e0: Pull complete
0241c68333ef: Pull complete
Digest: sha256:75a55d33ecc73c2a242450a9f1cc858499d468f077ea942867e662c247b5e412
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
```

나눠서 받는다
(Layer 별로)

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# docker image inspect nginx:latest | jq '.[].RootFS'
```

```
{
  "Type": "layers",
  "Layers": [
    "sha256:7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d",
    "sha256:4dc529e519c4390939b1616595683c89465782bb7d9fc7b90b30cc1e95bc723a",
    "sha256:23c959acc3d0eb744031aef67adf6ceb5120a19c8869727d588f7d9dabd75b09",
    "sha256:15aac1be5f02f2188ab40430b28a5f79be1bcb805db315bbe4d70f70aeabaa36",
    "sha256:974e9faf62f1a3c3210e3904420ffec1dc351b756ac33024f2dd2683bf44c370",
    "sha256:64ee8c6d0de0cfd019841b29c8cb18f4ab38e4687f7784866b840d5b2c31c8b9"
  ]
}
```

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# docker image inspect nginx | jq '.[].RootFS'
```

응 ? id 가 다르네

```
{
  "Type": "layers",
  "Layers": [
    "sha256:7e718b9c0c8c
    "sha256:4dc529e519c4
    "sha256:23c959acc3d0
    "sha256:15aac1be5f02
    "sha256:974e9faf62f1
    "sha256:64ee8c6d0de0
  ]
}
```

```
f7ec5a41d630: Pull complete
aa1efa14b3bf: Pull complete
b78b95af9b17: Pull complete
c7d6bca2b8dc: Pull complete
cf16cd8e71e0: Pull complete
0241c68333ef: Pull complete
```

```
ce4b2caf75891d82fb227d",
9fc7b90b30cc1e95bc723a",
69727d588f7d9dabd75b09",
b315bbe4d70f70aeabaa36",
c33024f2dd2683bf44c370",
7784866b840d5b2c31c8b9"
```

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# docker pull nginx:latest
latest: Pulling from library/nginx
f7ec5a41d630: Pull complete
aa1efa14b3bf: Pull complete
b78b95af9b17: Pull complete
c7d6bca2b8dc: Pull complete
cf16cd8e71e0: Pull complete
0241c68333ef: Pull complete
Digest: sha256:75a55d33ecc73c2a242450a9f1cc858499d468f077ea942867e662c247b5e412
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
```

각 Layer 의 “distribution id”

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# docker image inspect nginx | jq '.[].RootFS'
```

```
{
  "Type": "layers",
  "Layers": [
    "sha256:7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d",
    "sha256:4dc529e519c4390939b1616595683c89465782bb7d9fc7b90b30cc1e95bc723a",
    "sha256:23c959acc3d0eb744031aef67adf6ceb5120a19c8869727d588f7d9dabd75b09",
    "sha256:15aac1be5f02f2188ab40430b28a5f79be1bcb805db315bbe4d70f70aeabaa36",
    "sha256:974e9faf62f1a3c3210e3904420ffec1dc351b756ac33024f2dd2683bf44c370",
    "sha256:64ee8c6d0de0cfd019841b29c8cb18f4ab38e4687f7784866b840d5b2c31c8b9"
  ]
}
```

layer-id

(실습4) 컨테이너 레이어 구조

```
터미널 #1
# docker info
. . .
  Storage Driver: overlay2
. . .
  Docker Root Dir: /var/lib/docker
. . .
  Registry: https://index.docker.io/v1/
. . .


# cd /var/lib/docker/image/overlay2
```

(실습4) 컨테이너 레이어 구조

터미널 #1

```
root@ubuntu1804:/var/lib/docker/image/overlay2# tree -L 2
.
├── distribution
│   ├── diffid-by-digest
│   └── v2metadata-by-diffid
├── imagedb
│   ├── content
│   └── metadata
├── layerdb
│   ├── mounts
│   ├── sha256
│   └── tmp
└── repositories.json

10 directories, 1 file
```

https://programmer.group/docker-learning-image-s-local-storage-architecture.html

(실습4) 컨테이너 레이어 구조

터미널 #1

```
root@ubuntu1804:/var/lib/docker/image/overlay2# tree -L 2
.
├── distribution
│   ├── diffid-by-digest
│   └── v2metadata-by-diffid
├── imagedb
│   ├── content
│   └── metadata
├── layerdb
│   ├── mounts
│   ├── sha256
│   └── tmp
└── repositories.json

10 directories, 1 file
```

https://programmer.group/docker-learning-image-s-local-storage-architecture.html

(실습4) 컨테이너 레이어 구조

터미널 #1

```
root@ubuntu1804:/var/lib/docker/image/overlay2/layerdb# tree -L 2
.
├── mounts
├── sha256
│   ├── 11126fda59f7f4bf9bf08b9d24c9ea45a1194f3d61ae2a96af744c97eae71cbf
│   ├── 3444fb58dc9e8338f6da71c1040e8ff532f25fab497312f95dcee0f756788a84
│   ├── 704bf100d7f16255a2bc92e925f7007eef0bd3947af4b860a38aaffc3f992eae
│   ├── 7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
│   ├── d5955c2e658d1432abb023d7d6d1128b0aa12481b976de7cbde4c7a31310f29b
│   └── f85cfdc7ca97d8856cd4fa916053084e2e31c7e53ed169577cef5cb1b8169ccb
└── tmp

9 directories, 0 files
```

layerdb id

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# tree -L 2 sha256
```

cache-id : *layer 저장경로 id*
diff : *layer id*
parent : *layerdb id*

```
sha256
├── 11126fda59f7f4bf9bf08b9d24c9ea45a1194f3d61ae2a96af744c97eae71cbf
│   ├── cache-id
│   ├── diff
│   ├── parent
├── 3444fb58dc9e8338f6da71c1040e8ff532f25fab497312f95dcee0f756788a84
│   ├── cache-id
│   ├── diff
│   ├── parent
├── 704bf100d7f16255a2bc92e925f7007eef0bd3947af4b860a38aaffc3f992eae
│   ├── cache-id
│   ├── diff
│   ├── parent
├── 7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
│   ├── cache-id
│   ├── diff
├── d5955c2e658d1432abb023d7d6d1128b0aa12481b976de7cbde4c7a31310f29b
│   ├── cache-id
│   ├── diff
│   ├── parent
└── f85cfdc7ca97d8856cd4fa916053084e2e31c7e53ed169577cef5cb1b8169ccb
    ├── cache-id
    ├── diff
    ├── parent
```

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# ls sha256/*/diff | awk '{system("cat "$0";echo")}'

sha256:4dc529e519c4390939b1616595683c89465782bb7d9fc7b90b30cc1e95bc723a
sha256:64ee8c6d0de0cfd019841b29c8cb18f4ab38e4687f7784866b840d5b2c31c8b9
sha256:15aac1be5f02f2188ab40430b28a5f79be1bcb805db315bbe4d70f70aeabaa36
sha256:7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
sha256:23c959acc3d0eb744031aef67adf6ceb5120a19c8869727d588f7d9dabd75b09
sha256:974e9faf62f1a3c3210e3904420ffec1dc351b756ac33024f2dd2683bf44c370
```

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# ls sha256/*/diff | awk '{system("cat "$0";echo")}'

sha256:4dc529e519c4390939b1616595683c89465782bb7d9fc7b90b30cc1e95bc723a
sha256:64ee8c6d0de0cfd019841b29c8cb18f4ab38e4687f7784866b840d5b2c31c8b9
sha256:15aac1be5f02f2188ab40430b28a5f79be1bcb805db315bbe4d70f70aeabaa36
sha256:7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
sha256:23c959acc3d0eb744031aef67adf6ceb5120a19c8869727d588f7d9dabd75b09
sha256:974e9faf62f1a3c3210e3904420ffec1dc351b756ac33024f2dd2683bf44c370
```

layer id

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# tree -L 2 sha256
```

cache-id : *layer 저장경로 id*
diff : *layer id*
parent : *layerdb id*

```
sha256
├── 11126fda59f7f4bf9bf08b9d24c9ea45a1194f3d61ae2a96af744c97eae71cbf
│   ├── cache-id
│   ├── diff 4dc529e519c4390939b1616595683c89465782bb7d9fc7b90b30cc1e95bc723a
│   ├── parent
├── 3444fb58dc9e8338f6da71c1040e8ff532f25fab497312f95dcee0f756788a84
│   ├── cache-id
│   ├── diff 64ee8c6d0de0cfd019841b29c8cb18f4ab38e4687f7784866b840d5b2c31c8b9
│   ├── parent
├── 704bf100d7f16255a2bc92e925f7007eef0bd3947af4b860a38aaffc3f992eae
│   ├── cache-id
│   ├── diff 15aac1be5f02f2188ab40430b28a5f79be1bcb805db315bbe4d70f70aeabaa36
│   ├── parent
├── 7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
│   ├── cache-id
│   ├── diff 7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
├── d5955c2e658d1432abb023d7d6d1128b0aa12481b976de7cbde4c7a31310f29b
│   ├── cache-id
│   ├── diff 23c959acc3d0eb744031aef67adf6ceb5120a19c8869727d588f7d9dabd75b09
│   ├── parent
└── f85cfdc7ca97d8856cd4fa916053084e2e31c7e53ed169577cef5cb1b8169ccb
    ├── cache-id
    ├── diff 974e9faf62f1a3c3210e3904420ffec1dc351b756ac33024f2dd2683bf44c370
    ├── parent
```

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# tree -L 2 sha256
```

cache-id : *layer 저장경로 id*
diff : *layer id*
parent : *layerdb id*

```
sha256
├── 11126fda59f7f4bf9bf08b9d24c9ea45a1194f3d61ae2a96af744c97eae71cbf
│   ├── cache-id
│   ├── diff
│   ├── parent
├── 3444fb58dc9e8338f6da71c1040e8ff532f25fab497312f95dcee0f756788a84
│   ├── cache-id
│   ├── diff
│   ├── parent
├── 704bf100d7f16255a2bc92e925f7007eef0bd3947af4b860a38aaffc3f992eae
│   ├── cache-id
│   ├── diff
│   ├── parent
├── 7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
│   ├── cache-id
│   ├── diff
├── d5955c2e658d1432abb023d7d6d1128b0aa12481b976de7cbde4c7a31310f29b
│   ├── cache-id
│   ├── diff
│   ├── parent
└── f85cfdc7ca97d8856cd4fa916053084e2e31c7e53ed169577cef5cb1b8169ccb
    ├── cache-id
    ├── diff
    ├── parent
```

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# ls sha256/*/parent | awk '{system("cat "$0";echo")}'

sha256:7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
sha256:f85cfdc7ca97d8856cd4fa916053084e2e31c7e53ed169577cef5cb1b8169ccb
sha256:d5955c2e658d1432abb023d7d6d1128b0aa12481b976de7cbde4c7a31310f29b
sha256:11126fda59f7f4bf9bf08b9d24c9ea45a1194f3d61ae2a96af744c97eae71cbf
sha256:704bf100d7f16255a2bc92e925f7007eef0bd3947af4b860a38aaffc3f992eae

# cd /var/lib/docker/overlay2
```

parent

“cache-id” ~ 로컬에서 이미지 레이어 저장 경로 식별

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# tree -L 2 sha256
```

cache-id : *layer 저장경로 id*
diff : *layer id*
parent : *layerdb id*

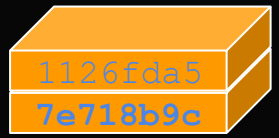
```
sha256
├── 11126fda59f7f4bf9bf08b9d24c9ea45a1194f3d61ae2a96af744c97eae71cbf
│   ├── cache-id
│   ├── diff
│   ├── parent 7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
├── 3444fb58dc9e8338f6da71c1040e8ff532f25fab497312f95dcee0f756788a84
│   ├── cache-id
│   ├── diff
│   ├── parent f85cfdc7ca97d8856cd4fa916053084e2e31c7e53ed169577cef5cb1b8169ccb
├── 704bf100d7f16255a2bc92e925f7007eef0bd3947af4b860a38aaffc3f992eae
│   ├── cache-id
│   ├── diff
│   ├── parent d5955c2e658d1432abb023d7d6d1128b0aa12481b976de7cbde4c7a31310f29b
├── 7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
│   ├── cache-id
│   ├── diff
├── d5955c2e658d1432abb023d7d6d1128b0aa12481b976de7cbde4c7a31310f29b
│   ├── cache-id
│   ├── diff
│   ├── parent 11126fda59f7f4bf9bf08b9d24c9ea45a1194f3d61ae2a96af744c97eae71cbf
└── f85cfdc7ca97d8856cd4fa916053084e2e31c7e53ed169577cef5cb1b8169ccb
    ├── cache-id
    ├── diff
    ├── parent 704bf100d7f16255a2bc92e925f7007eef0bd3947af4b860a38aaffc3f992eae
```

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# tree -L 2 sha256
```

cache-id : *layer 저장경로 id*
diff : *layer id*
parent : *layerdb id*




```
sha256
├── 11126fda59f7f4bf9bf08b9d24c9ea45a1194f3d61ae2a96af744c97eae71cbf
│   ├── cache-id
│   ├── diff
│   ├── parent 7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
├── 3444fb58dc9e8338f6da71c1040e8ff532f25fab497312f95dcee0f756788a84
│   ├── cache-id
│   ├── diff
│   ├── parent f85cfdc7ca97d8856cd4fa916053084e2e31c7e53ed169577cef5cb1b8169ccb
├── 704bf100d7f16255a2bc92e925f7007eef0bd3947af4b860a38aaffc3f992eae
│   ├── cache-id
│   ├── diff
│   ├── parent d5955c2e658d1432abb023d7d6d1128b0aa12481b976de7cbde4c7a31310f29b
├── 7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
│   ├── cache-id
│   ├── diff
├── d5955c2e658d1432abb023d7d6d1128b0aa12481b976de7cbde4c7a31310f29b
│   ├── cache-id
│   ├── diff
│   ├── parent 11126fda59f7f4bf9bf08b9d24c9ea45a1194f3d61ae2a96af744c97eae71cbf
└── f85cfdc7ca97d8856cd4fa916053084e2e31c7e53ed169577cef5cb1b8169ccb
    ├── cache-id
    ├── diff
    ├── parent 704bf100d7f16255a2bc92e925f7007eef0bd3947af4b860a38aaffc3f992eae
```

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# tree -L 2 sha256
```

cache-id : *layer 저장경로 id*
diff : *layer id*
parent : *layerdb id*

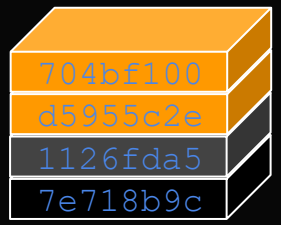

```
sha256
├── 11126fda59f7f4bf9bf08b9d24c9ea45a1194f3d61ae2a96af744c97eae71cbf
│   ├── cache-id
│   ├── diff
│   ├── parent 7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
├── 3444fb58dc9e8338f6da71c1040e8ff532f25fab497312f95dcee0f756788a84
│   ├── cache-id
│   ├── diff
│   ├── parent f85cfdc7ca97d8856cd4fa916053084e2e31c7e53ed169577cef5cb1b8169ccb
├── 704bf100d7f16255a2bc92e925f7007eef0bd3947af4b860a38aaffc3f992eae
│   ├── cache-id
│   ├── diff
│   ├── parent d5955c2e658d1432abb023d7d6d1128b0aa12481b976de7cbde4c7a31310f29b
├── 7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
│   ├── cache-id
│   ├── diff
├── d5955c2e658d1432abb023d7d6d1128b0aa12481b976de7cbde4c7a31310f29b
│   ├── cache-id
│   ├── diff
│   ├── parent 11126fda59f7f4bf9bf08b9d24c9ea45a1194f3d61ae2a96af744c97eae71cbf
└── f85cfdc7ca97d8856cd4fa916053084e2e31c7e53ed169577cef5cb1b8169ccb
    ├── cache-id
    ├── diff
    ├── parent 704bf100d7f16255a2bc92e925f7007eef0bd3947af4b860a38aaffc3f992eae
```

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# tree -L 2 sha256
```

cache-id : *layer 저장경로 id*
diff : *layer id*
parent : *layerdb id*



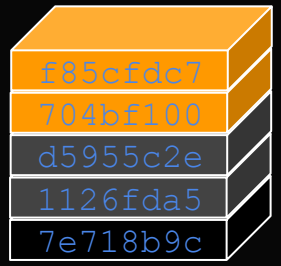

```
sha256
├── 11126fda59f7f4bf9bf08b9d24c9ea45a1194f3d61ae2a96af744c97eae71cbf
│   ├── cache-id
│   ├── diff
│   ├── parent 7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
├── 3444fb58dc9e8338f6da71c1040e8ff532f25fab497312f95dcee0f756788a84
│   ├── cache-id
│   ├── diff
│   ├── parent f85cfdc7ca97d8856cd4fa916053084e2e31c7e53ed169577cef5cb1b8169ccb
├── 704bf100d7f16255a2bc92e925f7007eef0bd3947af4b860a38aaffc3f992eae
│   ├── cache-id
│   ├── diff
│   ├── parent d5955c2e658d1432abb023d7d6d1128b0aa12481b976de7cbde4c7a31310f29b
├── 7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
│   ├── cache-id
│   ├── diff
├── d5955c2e658d1432abb023d7d6d1128b0aa12481b976de7cbde4c7a31310f29b
│   ├── cache-id
│   ├── diff
│   ├── parent 11126fda59f7f4bf9bf08b9d24c9ea45a1194f3d61ae2a96af744c97eae71cbf
└── f85cfdc7ca97d8856cd4fa916053084e2e31c7e53ed169577cef5cb1b8169ccb
    ├── cache-id
    ├── diff
    ├── parent 704bf100d7f16255a2bc92e925f7007eef0bd3947af4b860a38aaffc3f992eae
```

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# tree -L 2 sha256
```

cache-id : *layer 저장경로 id*
diff : *layer id*
parent : *layerdb id*



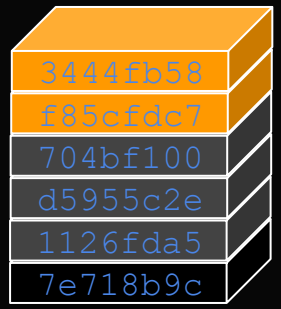

```
sha256
├── 11126fda59f7f4bf9bf08b9d24c9ea45a1194f3d61ae2a96af744c97eae71cbf
│   ├── cache-id
│   ├── diff
│   ├── parent 7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
├── 3444fb58dc9e8338f6da71c1040e8ff532f25fab497312f95dcee0f756788a84
│   ├── cache-id
│   ├── diff
│   ├── parent f85cfdc7ca97d8856cd4fa916053084e2e31c7e53ed169577cef5cb1b8169ccb
├── 704bf100d7f16255a2bc92e925f7007eef0bd3947af4b860a38aaffc3f992eae
│   ├── cache-id
│   ├── diff
│   ├── parent d5955c2e658d1432abb023d7d6d1128b0aa12481b976de7cbde4c7a31310f29b
├── 7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
│   ├── cache-id
│   ├── diff
├── d5955c2e658d1432abb023d7d6d1128b0aa12481b976de7cbde4c7a31310f29b
│   ├── cache-id
│   ├── diff
│   ├── parent 11126fda59f7f4bf9bf08b9d24c9ea45a1194f3d61ae2a96af744c97eae71cbf
└── f85cfdc7ca97d8856cd4fa916053084e2e31c7e53ed169577cef5cb1b8169ccb
    ├── cache-id
    ├── diff
    ├── parent 704bf100d7f16255a2bc92e925f7007eef0bd3947af4b860a38aaffc3f992eae
```

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# tree -L 2 sha256
```

cache-id : *layer 저장경로 id*
diff : *layer id*
parent : *layerdb id*



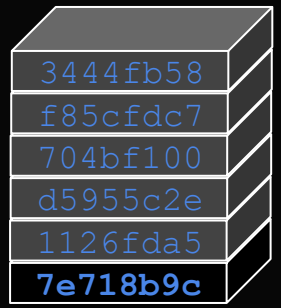

```
sha256
├── 11126fda59f7f4bf9bf08b9d24c9ea45a1194f3d61ae2a96af744c97eae71cbf
│   ├── cache-id
│   ├── diff
│   ├── parent 7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
├── 3444fb58dc9e8338f6da71c1040e8ff532f25fab497312f95dcee0f756788a84
│   ├── cache-id
│   ├── diff
│   ├── parent f85cfdc7ca97d8856cd4fa916053084e2e31c7e53ed169577cef5cb1b8169ccb
├── 704bf100d7f16255a2bc92e925f7007eef0bd3947af4b860a38aaffc3f992eae
│   ├── cache-id
│   ├── diff
│   ├── parent d5955c2e658d1432abb023d7d6d1128b0aa12481b976de7cbde4c7a31310f29b
├── 7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
│   ├── cache-id
│   ├── diff
├── d5955c2e658d1432abb023d7d6d1128b0aa12481b976de7cbde4c7a31310f29b
│   ├── cache-id
│   ├── diff
│   ├── parent 11126fda59f7f4bf9bf08b9d24c9ea45a1194f3d61ae2a96af744c97eae71cbf
└── f85cfdc7ca97d8856cd4fa916053084e2e31c7e53ed169577cef5cb1b8169ccb
    ├── cache-id
    ├── diff
    ├── parent 704bf100d7f16255a2bc92e925f7007eef0bd3947af4b860a38aaffc3f992eae
```

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# tree -L 2 sha256
```

cache-id : *layer 저장경로 id*
diff : *layer id*
parent : *layerdb id*




```
sha256
├── 11126fda59f7f4bf9bf08b9d24c9ea45a1194f3d61ae2a96af744c97eae71cbf
│   ├── cache-id
│   ├── diff
│   ├── parent 7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
├── 3444fb58dc9e8338f6da71c1040e8ff532f25fab497312f95dcee0f756788a84
│   ├── cache-id
│   ├── diff
│   ├── parent f85cfdc7ca97d8856cd4fa916053084e2e31c7e53ed169577cef5cb1b8169ccb
├── 704bf100d7f16255a2bc92e925f7007eef0bd3947af4b860a38aaffc3f992eae
│   ├── cache-id
│   ├── diff
│   ├── parent d5955c2e658d1432abb023d7d6d1128b0aa12481b976de7cbde4c7a31310f29b
├── 7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
│   ├── cache-id
│   ├── diff
├── d5955c2e658d1432abb023d7d6d1128b0aa12481b976de7cbde4c7a31310f29b
│   ├── cache-id
│   ├── diff
│   ├── parent 11126fda59f7f4bf9bf08b9d24c9ea45a1194f3d61ae2a96af744c97eae71cbf
└── f85cfdc7ca97d8856cd4fa916053084e2e31c7e53ed169577cef5cb1b8169ccb
    ├── cache-id
    ├── diff
    ├── parent 704bf100d7f16255a2bc92e925f7007eef0bd3947af4b860a38aaffc3f992eae
```

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# tree -L 2 sha256
```

cache-id : *layer 저장경로 id*
diff : *layer id*
parent : *layerdb id*

```
sha256
├── 11126fda59f7f4bf9bf08b9d24c9ea45a1194f3d61ae2a96af744c97eae71cbf
│   ├── cache-id
│   ├── diff
│   ├── parent
├── 3444fb58dc9e8338f6da71c1040e8ff532f25fab497312f95dcee0f756788a84
│   ├── cache-id
│   ├── diff
│   ├── parent
├── 704bf100d7f16255a2bc92e925f7007eef0bd3947af4b860a38aaffc3f992eae
│   ├── cache-id
│   ├── diff
│   ├── parent
├── 7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
│   ├── cache-id
│   ├── diff
├── d5955c2e658d1432abb023d7d6d1128b0aa12481b976de7cbde4c7a31310f29b
│   ├── cache-id
│   ├── diff
│   ├── parent
└── f85cfdc7ca97d8856cd4fa916053084e2e31c7e53ed169577cef5cb1b8169ccb
    ├── cache-id
    ├── diff
    ├── parent
```

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# ls sha256/*/cache-id | awk '{system("cat "$0";echo")}'

44ff0d56badc460181811a61d11241ca3db6a6b0878f8d66dcaaa5b6b42ae084
7dcdacd7b24bb2c8a81411fd41627c8ad31be112c12a1de9dbebc6c3d8148eb0
278f5c4a27a4ed53f22e4f42516aa4783b9d875c032b3df79cfc4a4ae3c18871
114ea84e90bd698c7958ad69316d2ed82712d624e75688a7049ce5896878ca2d
9914864f4efcf657f11344d76840d77026d220ee146af8832c731c04d5463e9f
e45ce4954c7fb1df93ddf76b5d63a399588456fd49e1d3568b87b57cedb26fb7
```

cache-id

"cache-id" ~ 로컬에서 이미지 레이어 저장 경로 식별

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# tree -L 2 sha256
```

cache-id : *layer 저장경로 id*
diff : *layer id*
parent : *layerdb id*

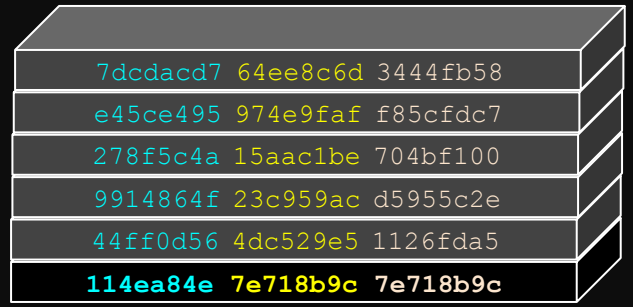
```
sha256
├── 11126fda59f7f4bf9bf08b9d24c9ea45a1194f3d61ae2a96af744c97eae71cbf
│   ├── cache-id 44ff0d56badc460181811a61d11241ca3db6a6b0878f8d66dcaaa5b6b42ae084
│   ├── diff
│   ├── parent
├── 3444fb58dc9e8338f6da71c1040e8ff532f25fab497312f95dcee0f756788a84
│   ├── cache-id 7dcdacd7b24bb2c8a81411fd41627c8ad31be112c12a1de9dbebc6c3d8148eb0
│   ├── diff
│   ├── parent
├── 704bf100d7f16255a2bc92e925f7007eef0bd3947af4b860a38aaffc3f992eae
│   ├── cache-id 278f5c4a27a4ed53f22e4f42516aa4783b9d875c032b3df79cfc4a4ae3c18871
│   ├── diff
│   ├── parent
├── 7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
│   ├── cache-id 114ea84e90bd698c7958ad69316d2ed82712d624e75688a7049ce5896878ca2d
│   ├── diff
├── d5955c2e658d1432abb023d7d6d1128b0aa12481b976de7cbde4c7a31310f29b
│   ├── cache-id 9914864f4efcf657f11344d76840d77026d220ee146af8832c731c04d5463e9f
│   ├── diff
│   ├── parent
└── f85cfdc7ca97d8856cd4fa916053084e2e31c7e53ed169577cef5cb1b8169ccb
    ├── cache-id e45ce4954c7fb1df93ddf76b5d63a399588456fd49e1d3568b87b57cedb26fb7
    ├── diff
    ├── parent
```

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# tree -L 2 sha256
```

cache-id : *layer 저장경로 id*
diff : *layer id*
parent : *layerdb id*




```
sha256
├── 11126fda59f7f4bf9bf08b9d24c9ea45a1194f3d61ae2a96af744c97eae71cbf
│   ├── cache-id 44ff0d56badc460181811a61d11241ca3db6a6b0878f8d66dcaaa5b6b42ae084
│   ├── diff 4dc529e519c4390939b1616595683c89465782bb7d9fc7b90b30cc1e95bc723a
│   ├── parent 7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
├── 3444fb58dc9e8338f6da71c1040e8ff532f25fab497312f95dcee0f756788a84
│   ├── cache-id 7dcdacd7b24bb2c8a81411fd41627c8ad31be112c12a1de9dbebc6c3d8148eb0
│   ├── diff 64ee8c6d0de0cfd019841b29c8cb18f4ab38e4687f7784866b840d5b2c31c8b9
│   ├── parent f85cfdc7ca97d8856cd4fa916053084e2e31c7e53ed169577cef5cb1b8169ccb
├── 704bf100d7f16255a2bc92e925f7007eef0bd3947af4b860a38aaffc3f992eae
│   ├── cache-id 278f5c4a27a4ed53f22e4f42516aa4783b9d875c032b3df79cfc4a4ae3c18871
│   ├── diff 15aac1be5f02f2188ab40430b28a5f79be1bcb805db315bbe4d70f70aeabaa36
│   ├── parent d5955c2e658d1432abb023d7d6d1128b0aa12481b976de7cbde4c7a31310f29b
├── 7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
│   ├── cache-id 114ea84e90bd698c7958ad69316d2ed82712d624e75688a7049ce5896878ca2d
│   ├── diff 7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d
├── d5955c2e658d1432abb023d7d6d1128b0aa12481b976de7cbde4c7a31310f29b
│   ├── cache-id 9914864f4efcf657f11344d76840d77026d220ee146af8832c731c04d5463e9f
│   ├── diff 23c959acc3d0eb744031aef67adf6ceb5120a19c8869727d588f7d9dabd75b09
│   ├── parent 11126fda59f7f4bf9bf08b9d24c9ea45a1194f3d61ae2a96af744c97eae71cbf
└── f85cfdc7ca97d8856cd4fa916053084e2e31c7e53ed169577cef5cb1b8169ccb
    ├── cache-id e45ce4954c7fb1df93ddf76b5d63a399588456fd49e1d3568b87b57cedb26fb7
    ├── diff 974e9faf62f1a3c3210e3904420ffec1dc351b756ac33024f2dd2683bf44c370
    ├── parent 704bf100d7f16255a2bc92e925f7007eef0bd3947af4b860a38aaffc3f992eae
```

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# docker image inspect nginx | jq '.[].GraphDriver''

{
  "Data": {
    "LowerDir":
"/var/lib/docker/overlay2/e45ce4954c7fb1df93ddf76b5d63a399588456fd49e1d3568b87b57cedb26fb7/diff:/var/lib/docker/o
verlay2/278f5c4a27a4ed53f22e4f42516aa4783b9d875c032b3df79cfc4a4ae3c18871/diff:/var/lib/docker/overlay2/9914864
f4efcf657f11344d76840d77026d220ee146af8832c731c04d5463e9f/diff:/var/lib/docker/overlay2/44ff0d56badc460181811a
61d11241ca3db6a6b0878f8d66dcaaa5b6b42ae084/diff:/var/lib/docker/overlay2/114ea84e90bd698c7958ad69316d2ed82
712d624e75688a7049ce5896878ca2d/diff",
    "MergedDir":
"/var/lib/docker/overlay2/7dcdacd7b24bb2c8a81411fd41627c8ad31be112c12a1de9dbebc6c3d8148eb0/merged",
    "UpperDir":
"/var/lib/docker/overlay2/7dcdacd7b24bb2c8a81411fd41627c8ad31be112c12a1de9dbebc6c3d8148eb0/diff",
    "WorkDir":
"/var/lib/docker/overlay2/7dcdacd7b24bb2c8a81411fd41627c8ad31be112c12a1de9dbebc6c3d8148eb0/work"
  },
  "Name": "overlay2"
}
```

(실습4) 컨테이너 레이어 구조

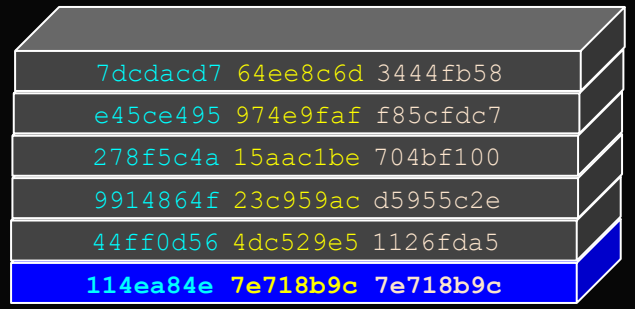
터미널 #1

```
root@ubuntu1804:/var/lib/docker/overlay2# tree -L 1
.
├── 114ea84e90bd698c7958ad69316d2ed82712d624e75688a7049ce5896878ca2d
├── 278f5c4a27a4ed53f22e4f42516aa4783b9d875c032b3df79cfc4a4ae3c18871
├── 44ff0d56badc460181811a61d11241ca3db6a6b0878f8d66dcaaa5b6b42ae084
├── 7dcdacd7b24bb2c8a81411fd41627c8ad31be112c12a1de9dbebc6c3d8148eb0
├── 9914864f4efcf657f11344d76840d77026d220ee146af8832c731c04d5463e9f
├── e45ce4954c7fb1df93ddf76b5d63a399588456fd49e1d3568b87b57cedb26fb7
└── l

7 directories, 0 files
```

cache-id

"cache-id" ~ 로컬에서 이미지 레이어 저장 경로 식별

(실습4) 컨테이너 레이어 구조



터미널 #1

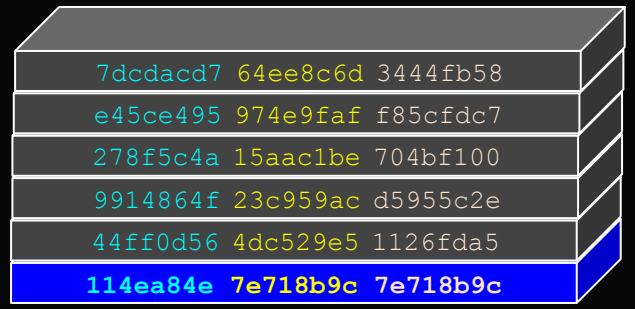
```
# docker image inspect nginx | jq '.[].GraphDriver''

{
  "Data": {
    "LowerDir":
"/var/lib/docker/overlay2/e45ce4954c7fb1df93ddf76b5d63a399588456fd49e1d3568b87b57cedb26fb7/diff:/var/lib/docker/o
verlay2/278f5c4a27a4ed53f22e4f42516aa4783b9d875c032b3df79cfc4a4ae3c18871/diff:/var/lib/docker/overlay2/9914864
f4efcf657f11344d76840d77026d220ee146af8832c731c04d5463e9f/diff:/var/lib/docker/overlay2/44ff0d56badc460181811a
61d11241ca3db6a6b0878f8d66dcaaa5b6b42ae084/diff:/var/lib/docker/overlay2/114ea84e90bd698c7958ad69316d2ed82
712d624e75688a7049ce5896878ca2d/diff",
    "MergedDir":
"/var/lib/docker/overlay2/7dcdacd7b24bb2c8a81411fd41627c8ad31be112c12a1de9dbebc6c3d8148eb0/merged",
    "UpperDir":
"/var/lib/docker/overlay2/7dcdacd7b24bb2c8a81411fd41627c8ad31be112c12a1de9dbebc6c3d8148eb0/diff",
    "WorkDir":
"/var/lib/docker/overlay2/7dcdacd7b24bb2c8a81411fd41627c8ad31be112c12a1de9dbebc6c3d8148eb0/work"
  },
  "Name": "overlay2"
}
```

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# tree -L 1
/var/lib/docker/overlay2/114ea84e90bd698c7958ad69316d2ed82712d624e75688a7049ce5896878ca2d/diff
```

```
diff
├── bin
├── boot
├── dev
├── etc
├── home
├── lib
├── lib64
├── media
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin
├── srv
├── sys
├── tmp
├── usr
└── var
```



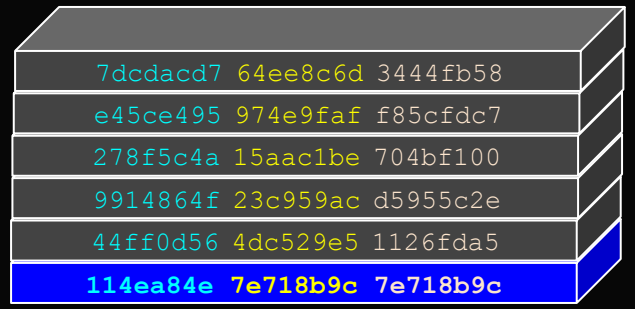
Base Image

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# tree -L 1
/var/lib/docker/overlay2/114ea84e90bd698c7958ad69316d2ed82712d624e75688a7049ce5896878ca2d/diff
```

```
diff
├── bin
├── boot
├── dev
├── etc
├── home
├── lib
├── lib64
├── media
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin
├── srv
├── sys
├── tmp
├── usr
└── var
```



Base Image

debian:buster-slim

https://github.com/nginxinc/docker-nginx/blob/master/stable/debian/Dockerfile

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# docker pull debian:buster-slim

buster-slim: Pulling from library/debian
f7ec5a41d630: Already exists
Digest: sha256:b586cf8c850cada85a47599f08eb34ede4a7c473551fd7c68cbf20ce5f8dbbf1
Status: Downloaded newer image for debian:buster-slim
docker.io/library/debian:buster-slim
```

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# docker pull debian:buster-slim

buster-slim: Pulling from library/debian
f7ec5a41d630: Already exists
Digest: sha256:b586cf8c850cada85a47599f08eb34ede4a7c473551fd7c68cbf20ce5f8dbbf1
Status: Downloaded newer image for debian:buster-slim
docker.io/library/debian:buster-slim
```

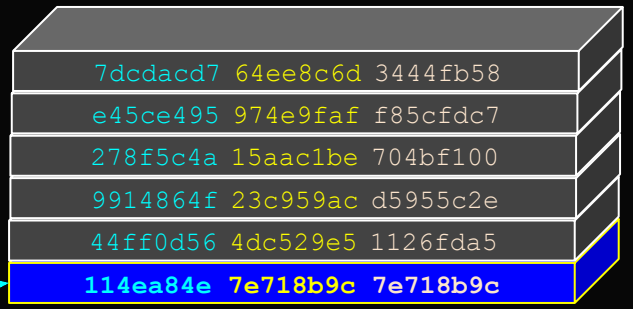
Same distribution id

```
f7ec5a41d630: Pull complete
aa1efa14b3bf: Pull complete
b78b95af9b17: Pull complete
c7d6bca2b8dc: Pull complete
cf16cd8e71e0: Pull complete
0241c68333ef: Pull complete
```

(실습4) 컨테이너 레이어 구조

터미널 #1

```
# docker pull debian:buster-slim

buster-slim: Pulling from library/debian
f7ec5a41d630: Already exists
Digest: sha256:b586cf8c850cada85a47599f08eb34ede4a7c475551fd7c68cbf20ce5f8dbbf1
Status: Downloaded newer image for debian:buster-slim
docker.io/library/debian:buster-slim
```




```
f7ec5a41d630: Pull complete
aa1efa14b3bf: Pull complete
b78b95af9b17: Pull complete
c7d6bca2b8dc: Pull complete
cf16cd8e71e0: Pull complete
0241c68333ef: Pull complete
```

(실습4) 컨테이너 레이어 구조

터미널 #1 (호스트)

```
# cd /var/lib/docker/overlay2/114ea84e*/diff
# ls
# echo 'lower layer' > a
# cat a
```

터미널 #2 (컨테이너-1)

```
# docker run -it debian:buster-slim
# ls
# cat a
```

Base layer 경로에 'a' 파일 생성 > 컨테이너 띄워서 확인

(실습4) 컨테이너 레이어 구조

터미널 #1 (호스트)

```
# cat a
lower layer
```

터미널 #2 (컨테이너-1)

```
# cat a
lower layer
```

터미널 #3 (컨테이너-2)

```
# docker run -it debian:buster-slim

# cat a
```

(실습4) 컨테이너 레이어 구조

터미널 #1 (호스트)

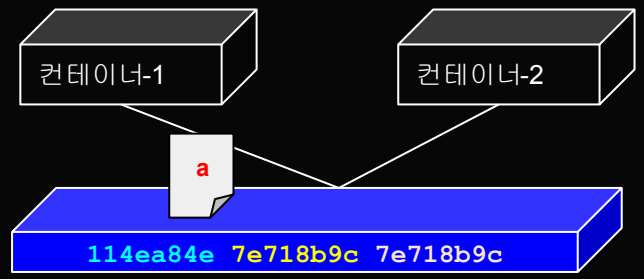
```
# cat a
lower layer
```

터미널 #2 (컨테이너-1)

```
# cat a
lower layer
```

터미널 #3 (컨테이너-2)

```
# cat a
lower layer
```



Base layer 경로를 Share 한다 !

(실습4) 컨테이너 레이어 구조

터미널 #1 (호스트)

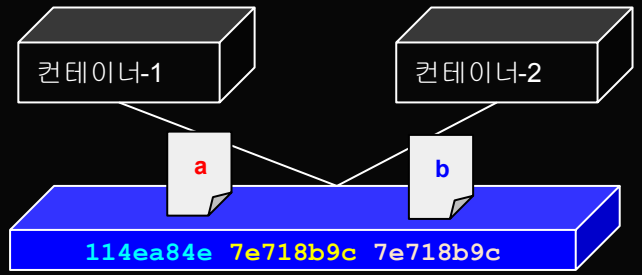
```
# echo 'lower layer' > b
```

터미널 #2 (컨테이너-1)

```
# cat a
lower layer
# cat b
lower layer
```

터미널 #3 (컨테이너-2)

```
# cat a
lower layer
# cat b
lower layer
```



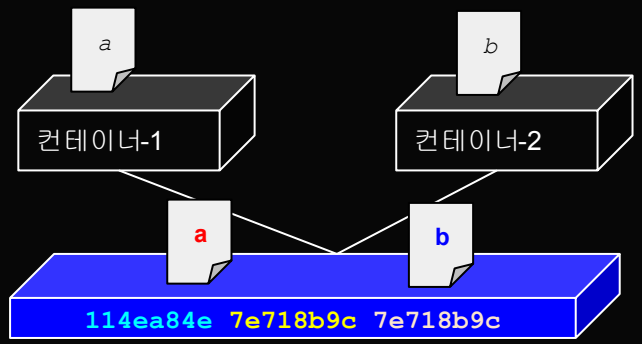
(실습4) 컨테이너 레이어 구조

터미널 #1 (호스트)

```
#
```

터미널 #2 (컨테이너-1)

```
# echo 'container-1' > a
```



터미널 #3 (컨테이너-2)

```
# echo 'container-2' > b
```

컨테이너에서 파일을 수정해보자

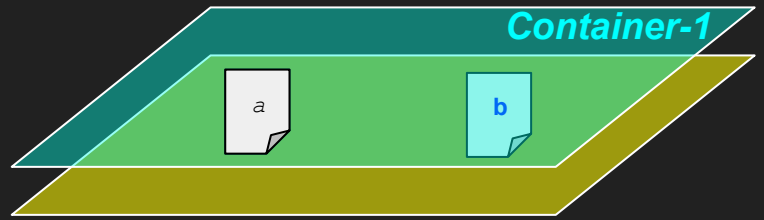
(실습4) 컨테이너 레이어 구조

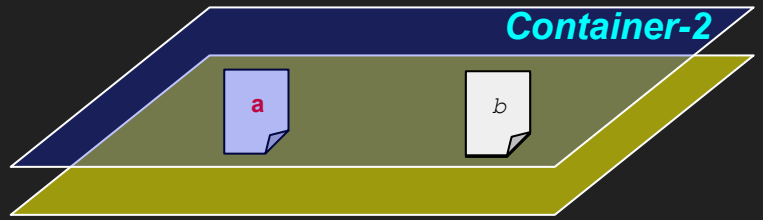
터미널 #2 (컨테이너-1)

```
# cat a

# cat b
```

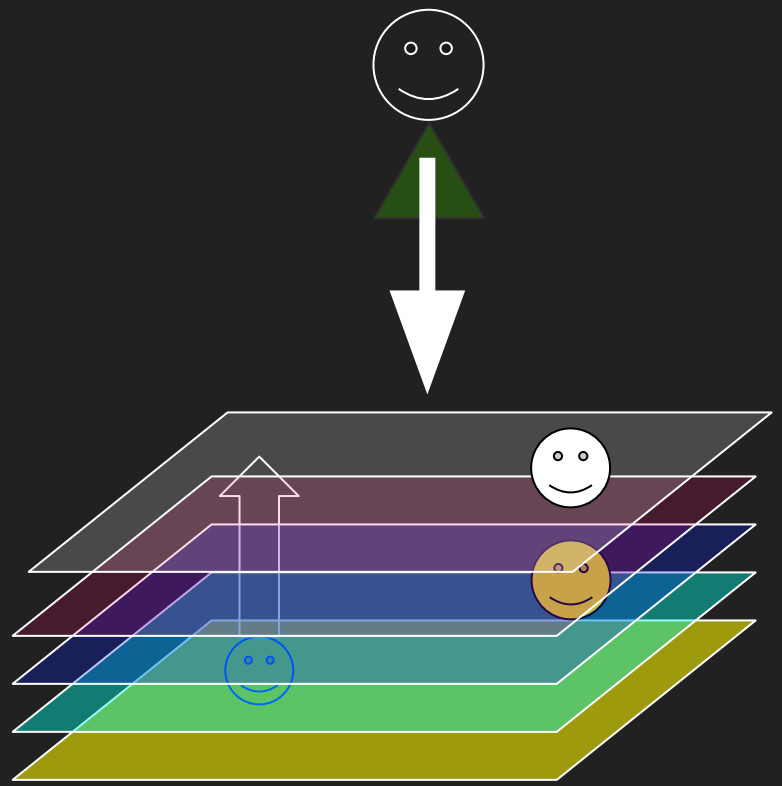
터미널 #3 (컨테이너-2)

```
# cat a

# cat b
```



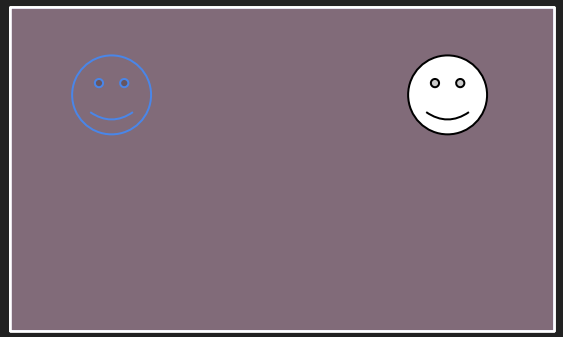


(실습4) 컨테이너 레이어 구조



"레이어가 쌓이는 순서"가 중요

여러장 겹쳐진 셀로판지를 위에서 내려다 본 모습



(실습4) 컨테이너 레이어 구조

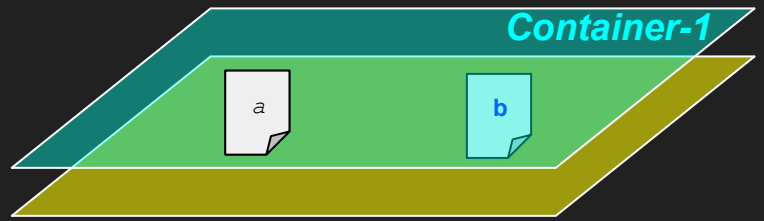
각 컨테이너에서 파일 수정 시 다른 컨테이너에 영향을 주지 않음

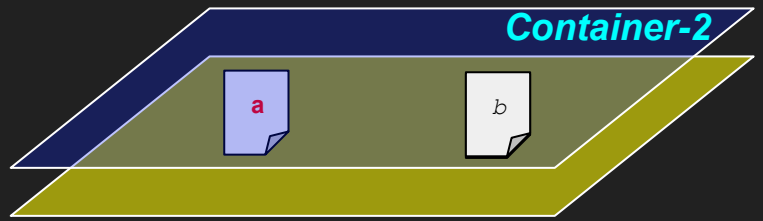
터미널 #2 (컨테이너-1)

```
# cat a
container-1
# cat b
lower layer
```

터미널 #3 (컨테이너-2)

```
# cat a
lower layer
# cat b
container-2
```





(Homework) 아래 정보를 이용하여 Union mount 하고 컨테이너를 기동하시오

터미널 #1 (호스트)

```
# docker image inspect nginx:latest | jq -r '.[].GraphDriver'
{
  "Data": {
    "LowerDir":
"/var/lib/docker/overlay2/e45ce4954c7fb1df93ddf76b5d63a399588456fd49e1d3568b87b57cedb26fb7/diff:/var/lib/docker/overlay2/278f5c4a27a4ed53f22e4f42516aa4783b9d875c032b3df79cfc4a4ae3c18871/diff:/var/lib/docker/overlay2/9914864f4efcf657f11344d76840d77026d220ee146af8832c731c04d5463e9f/diff:/var/lib/docker/overlay2/44ff0d56badc460181811a61d11241ca3db6a6b0878f8d66dcaaa5b6b42ae084/diff:/var/lib/docker/overlay2/114ea84e90bd698c7958ad69316d2ed82712d624e75688a7049ce5896878ca2d/diff",
    "MergedDir":
"/var/lib/docker/overlay2/7dcdacd7b24bb2c8a81411fd41627c8ad31be112c12a1de9dbebc6c3d8148eb0/merged",
    "UpperDir":
"/var/lib/docker/overlay2/7dcdacd7b24bb2c8a81411fd41627c8ad31be112c12a1de9dbebc6c3d8148eb0/diff",
    "WorkDir":
"/var/lib/docker/overlay2/7dcdacd7b24bb2c8a81411fd41627c8ad31be112c12a1de9dbebc6c3d8148eb0/work"
  },
  "Name": "overlay2"
}
```

(실습5) 이미지 레이어 추가하기

터미널 #1 (호스트)

```
# docker history nginx:latest
IMAGE          CREATED       CREATED BY                                      SIZE      COMMENT
62d49f9bab67   3 days ago    /bin/sh -c #(nop)  CMD ["nginx" "-g" "daemon…   0B
<missing>      3 days ago    /bin/sh -c #(nop)  STOPSIGNAL SIGQUIT           0B
<missing>      3 days ago    /bin/sh -c #(nop)  EXPOSE 80                    0B
<missing>      3 days ago    /bin/sh -c #(nop)  ENTRYPOINT ["/docker-entr…   0B
<missing>      3 days ago    /bin/sh -c #(nop) COPY file:09a214a3e07c919a…   4.61kB
<missing>      3 days ago    /bin/sh -c #(nop) COPY file:0fd5fca330dcd6a7…   1.04kB
<missing>      3 days ago    /bin/sh -c #(nop) COPY file:0b866ff3fc1ef5b0…   1.96kB
<missing>      3 days ago    /bin/sh -c #(nop) COPY file:65504f71f5855ca0…   1.2kB
<missing>      3 days ago    /bin/sh -c set -x     && addgroup --system -…   63.9MB
<missing>      3 days ago    /bin/sh -c #(nop)  ENV PKG_RELEASE=1~buster     0B
<missing>      3 days ago    /bin/sh -c #(nop)  ENV NJS_VERSION=0.5.3        0B
<missing>      3 days ago    /bin/sh -c #(nop)  ENV NGINX_VERSION=1.19.10    0B
<missing>      7 days ago    /bin/sh -c #(nop)  LABEL maintainer=NGINX Do…   0B
<missing>      7 days ago    /bin/sh -c #(nop)  CMD ["bash"]                 0B
<missing>      7 days ago    /bin/sh -c #(nop) ADD file:c855b3c65f5ba94d5…   69.3MB
```

(실습5) 이미지 레이어 추가하기

터미널 #1 (호스트)

```
# docker history nginx:latest
IMAGE          CREATED       CREATED BY                                      SIZE      COMMENT
62d49f9bab67   3 days ago    /bin/sh -c #(nop)  CMD ["nginx" "-g" "daemon…   0B
<missing>      3 days ago    /bin/sh -c #(nop)  STOPSIGNAL SIGQUIT           0B
<missing>      3 days ago    /bin/sh -c #(nop)  EXPOSE 80                    0B
<missing>      3 days ago    /bin/sh -c #(nop)  ENTRYPOINT ["/docker-entr…   0B
<missing>      3 days ago    /bin/sh -c #(nop) COPY file:09a214a3e07c919a…   4.61kB
<missing>      3 days ago    /bin/sh -c #(nop) COPY file:0fd5fca330dcd6a7…   1.04kB
<missing>      3 days ago    /bin/sh -c #(nop) COPY file:0b866ff3fc1ef5b0…   1.96kB
<missing>      3 days ago    /bin/sh -c #(nop) COPY file:65504f71f5855ca0…   1.2kB
<missing>      3 days ago    /bin/sh -c set -x     && addgroup --system -…   63.9MB
<missing>      3 days ago    /bin/sh -c #(nop)  ENV PKG_RELEASE=1~buster     0B
<missing>      3 days ago    /bin/sh -c #(nop)  ENV NJS_VERSION=0.5.3        0B
<missing>      3 days ago    /bin/sh -c #(nop)  ENV NGINX_VERSION=1.19.10    0B
<missing>      7 days ago    /bin/sh -c #(nop)  LABEL maintainer=NGINX Do…   0B
<missing>      7 days ago    /bin/sh -c #(nop)  CMD ["bash"]                 0B
<missing>      7 days ago    /bin/sh -c #(nop) ADD file:c855b3c65f5ba94d5…   69.3MB
```

(실습5) 이미지 레이어 추가하기

터미널 #1 (nginx)

```
# docker container run --name=nginx --rm -it nginx:latest bash

# rm /bin/tar
```

터미널 #2 (호스트)

```
# docker container diff nginx
C /bin
D /bin/tar

# docker container commit nginx nginx:rm_tar

# docker images
```

(실습5) 이미지 레이어 추가하기

터미널 #1 (nginx)

```
# docker container run --name=nginx --rm -it nginx:latest bash

# rm /bin/tar
```

터미널 #2 (호스트)

```
# docker image history nginx:rm_tar
```

IMAGE	CREATED	CREATED BY	SIZE	COMMENT
a6d20a78c29d	About a minute ago	bash	0B	
62d49f9bab67	3 days ago	/bin/sh -c #(nop) CMD ["nginx" "-g" "daemon…	0B	
<missing>	3 days ago	/bin/sh -c #(nop) STOPSIGNAL SIGQUIT	0B	
<missing>	3 days ago	/bin/sh -c #(nop) EXPOSE 80	0B	
<missing>	3 days ago	/bin/sh -c #(nop) ENTRYPOINT ["/docker-entr…	0B	
<missing>	3 days ago	/bin/sh -c #(nop) COPY file:09a214a3e07c919a…	4.61kB	
<missing>	3 days ago	/bin/sh -c #(nop) COPY file:0fd5fca330dcd6a7…	1.04kB	

(실습5) 이미지 레이어 추가하기

터미널 #2 (호스트)

```
# docker image inspect nginx:rm_tar | jq '.[].RootFS'

{
  "Type": "layers",
  "Layers": [
    "sha256:7e718b9c0c8c2e6420fe9c4d1d551088e314fe923dce4b2caf75891d82fb227d",
    "sha256:4dc529e519c4390939b1616595683c89465782bb7d9fc7b90b30cc1e95bc723a",
    "sha256:23c959acc3d0eb744031aef67adf6ceb5120a19c8869727d588f7d9dabd75b09",
    "sha256:15aac1be5f02f2188ab40430b28a5f79be1bcb805db315bbe4d70f70aeabaa36",
    "sha256:974e9faf62f1a3c3210e3904420ffec1dc351b756ac33024f2dd2683bf44c370",
    "sha256:64ee8c6d0de0cfd019841b29c8cb18f4ab38e4687f7784866b840d5b2c31c8b9",
    "sha256:84185e3de947251e67947cf9f1d7036642e3f9e4f70342330825f3ed92c169ea"
  ]
}
```

아래

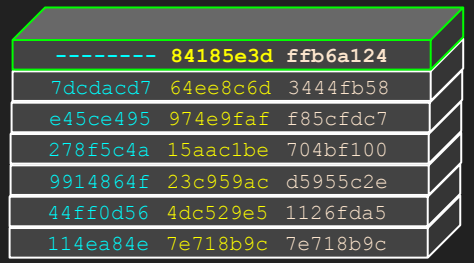
위

Layer id 확인

(실습5) 이미지 레이어 추가하기

터미널 #2 (호스트)

```
# cd /var/lib/docker/image/overlay2

# find . -name "diff" -exec cat {} \; -print | grep 84185e3d
sha256:84185e3de947251e67947cf9f1d7036642e3f9e4f70342330825f3ed92c169ea /layerdb/sha256/ffb6a124fc8fd5286f0effc68d9f106794e816ce5a21e0033330904ca996115e/diff
```

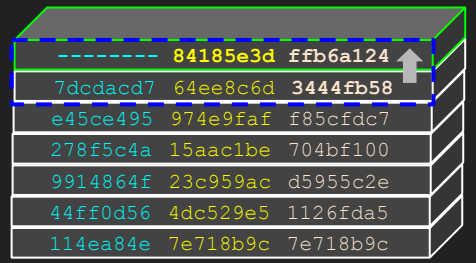


*diff (layer-id) → Layer**db** id 확인 “ffb6a124”*

(실습5) 이미지 레이어 추가하기

터미널 #2 (호스트)

```
# cd /var/lib/docker/image/overlay2
# find . -name "diff" -exec cat {} \; -print | grep 84185e3d
sha256:84185e3de947251e67947cf9f1d7036642e3f9e4f70342330825f3ed92c169ea./layerdb/sha256/ffb6a124fc8fd5286f0effc68d9f106794e816ce5a21e0033330904ca996115e/diff

# cat ./layerdb/sha256/ffb6a124*/parent;echo
sha256:3444fb58dc9e8338f6da71c1040e8ff532f25fab497312f95dcee0f756788a84
```



*Layer**db** id “ffb6a124” → parent 확인 “3444fb58”*

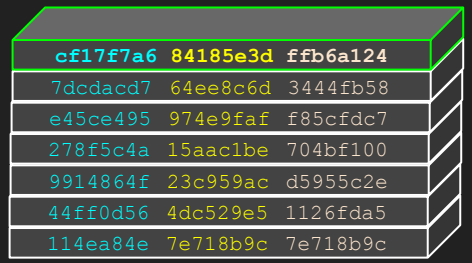
(실습5) 이미지 레이어 추가하기

터미널 #2 (호스트)

```
# cd /var/lib/docker/image/overlay2
# find . -name "diff" -exec cat {} \; -print | grep 84185e3d
sha256:84185e3de947251e67947cf9f1d7036642e3f9e4f70342330825f3ed92c169ea./layerdb/sha256/ffb6a124fc8fd5286f0effc68d9f106794e816ce5a21e0033330904ca996115e/diff

# cat ./layerdb/sha256/ffb6a124*/cache-id;echo
cf17f7a63aa9ea81fe60763cca3f3520d190fe6308d3b84c8cc8b610251bbdc7

# tree /var/lib/docker/overlay2/cf17f7a6*
```

```
├── diff
│   └── bin
│       └── tar
├── link
├── lower
└── work
```

whiteout



Layerdb id "ffb6a124" → cache-id 확인 "cf17f7a63" → 레이어 경로 확인

5편에서 마운트 네임스페이스와 이미지의 중복문제를
해결하기위한
오버레이 파일시스템에 대하여 다루었습니다.

목차 보기

6편에서는 지난 네트워크 네임스페이스에 이어서
오버레이 네트워크에 대해서 다루도록 하겠습니다.