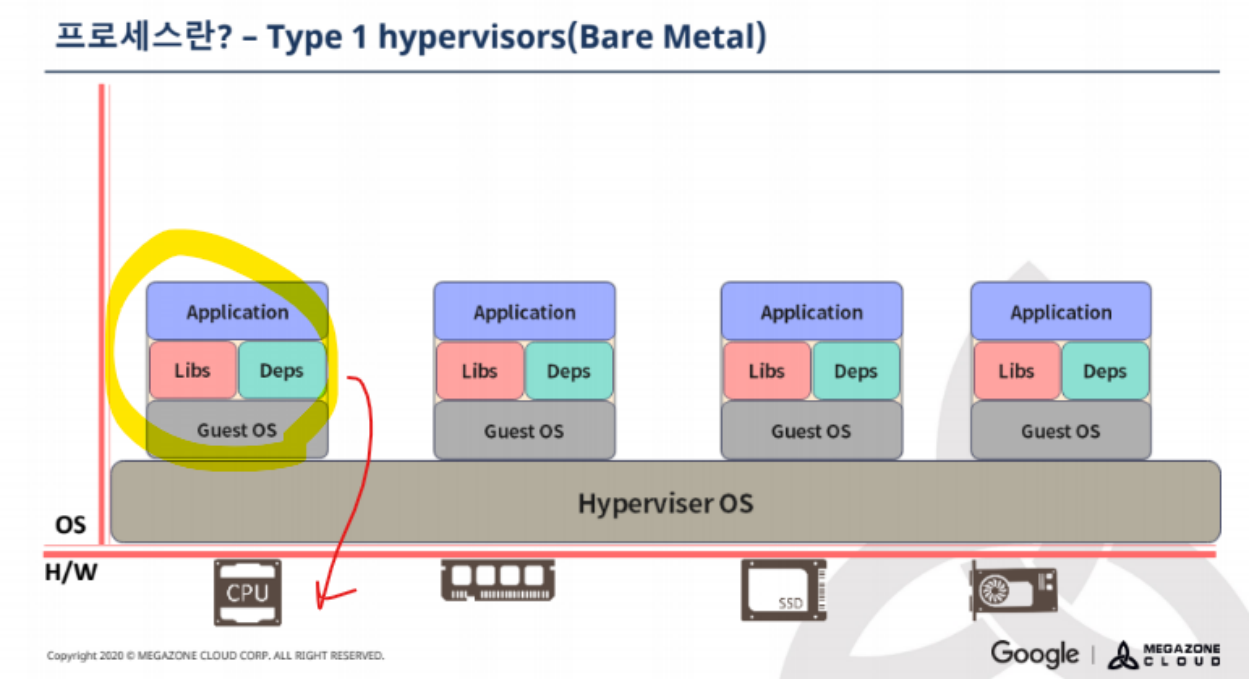
컨테이너

저급언어 & 고급언어

- **저급언어**
  - 컴파일러 전신
  - 기계어, 어셈블리어
  - 단순한 문자를 가지고 기계어로 변환
- **고급언어**
  - 컴파일러

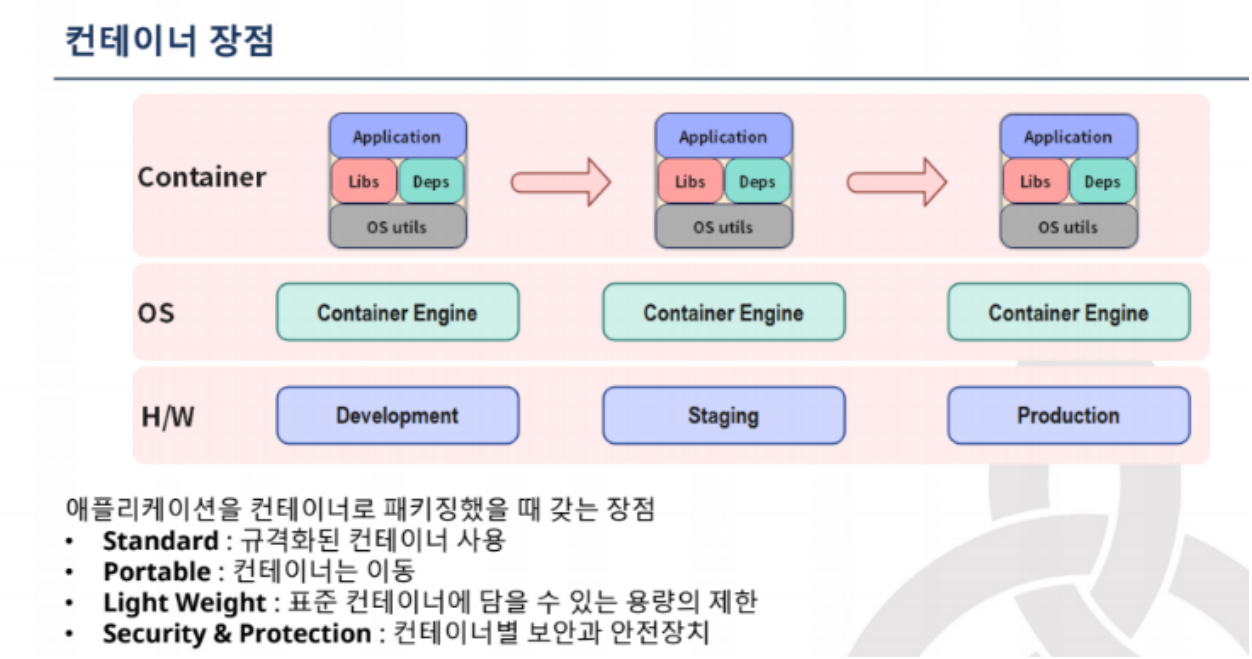
프로그래밍 언어의 구조가 기계어와 유사할수록 **저급언어(Low-level language)**, 사람이 이해하기 쉬운 구조일수록 **고급언어(High-level language)**라고 부릅니다. 성능에 차이가 있는 것이 아니라, 편의성이 기준입니다. 이 기준대로면 사람이 쓰는 언어가 가상 상위 레벨이겠죠. **C 언어 역시 CPU가 이해할 수 있도록 명령어를 기계어로 변경하는 '컴파일러(Compiler)'라는 프로그램을 사용**합니다.



- 중간에 GuestOS의 요구를 하이퍼바이저를 통해 hw에 전달해야함

컨테이너=프로세스

호스트에서 사용하는 프로그램 단위



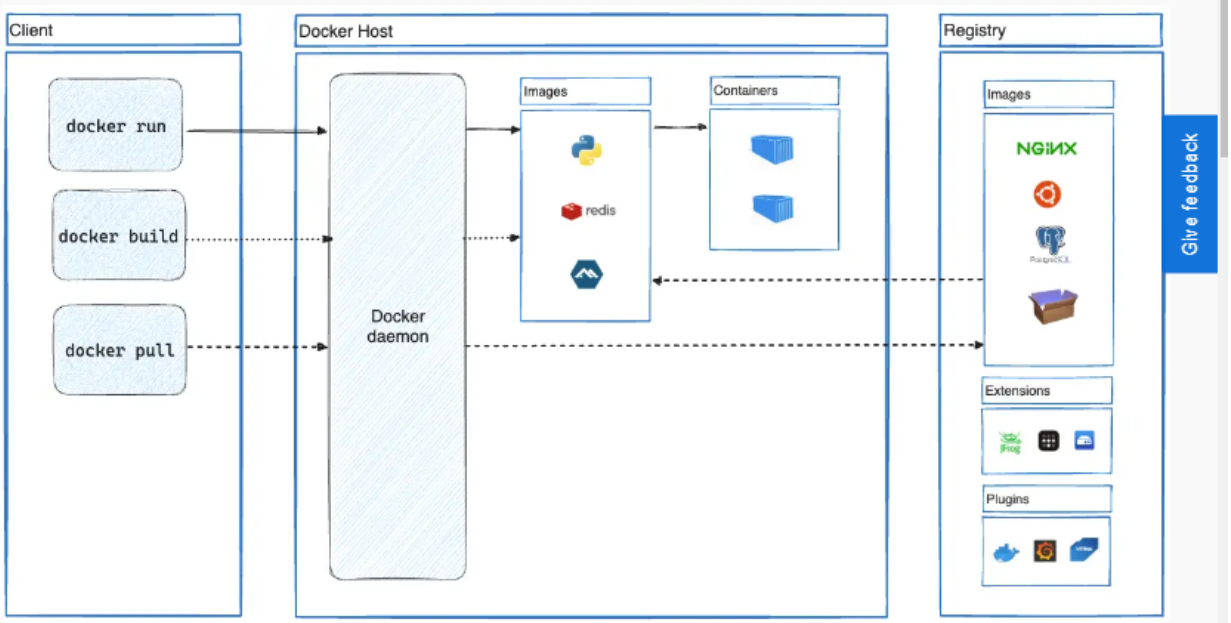
- 커널에 문제가 생기면 커널을 공유하는 다른 장비에도 문제가 생김
  - vm보단 보안성이 떨어짐
- **격리, 프로세스, 가상화, 추상화 개념 알기**
  - 추상화의 한계는 물리적 범위를 벗어날 수 없지만 그렇지 않은 아이들까지는 추상화
  - 그 중간은 하이퍼바이저로..?
  - ALL 컨테이너 환경으로 가기는 생각보다 쉽지 않음

- **Docker daemon**
  - 컨테이너 생성, 실행, 삭제, 자원할당, 동작 등을 관리하는 프로세스
  - 도커의 OS

실습- Ubuntu Server에 Docker Engine 설치

Docker architecture

Docker uses a client-server architecture. The Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing your Docker containers. The Docker client and daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon. The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface. Another Docker client is Docker Compose, that lets you work with applications consisting of a set of containers.



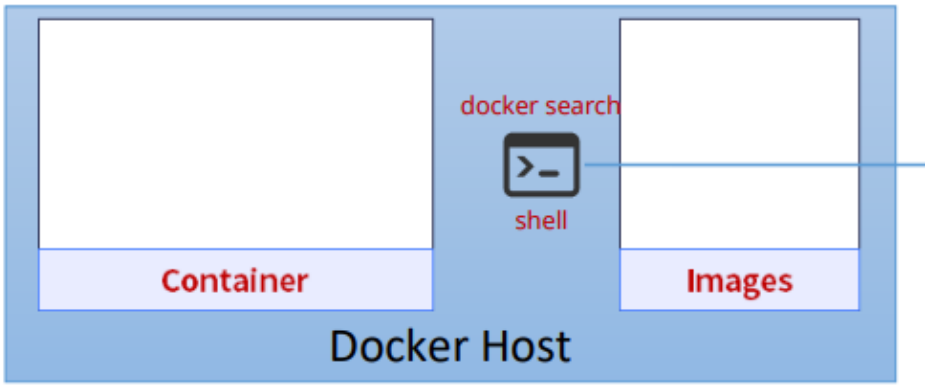
https://docs.docker.com/get-started/overview/

Docker Engine 설치

https://docs.docker.com/engine/install/ubuntu/

```
5. 100.1.3.1
root@ubuntu:~# docker search
"docker search" requires exactly 1 argument.
See 'docker search --help'.

Usage:  docker search [OPTIONS] TERM

Search Docker Hub for images
root@ubuntu:~#
```

- 현재 아무것도 없는 상태



```
root@ubuntu:~# docker search ubuntu
NAME                              DESCRIPTION                                     STARS     OFFICIAL
ubuntu                            Ubuntu is a Debian-based Linux operating sys…   16875     [OK]
websphere-liberty                 WebSphere Liberty multi-architecture images …   298       [OK]
open-liberty                      Open Liberty multi-architecture images based…   64        [OK]
neurodebian                       NeuroDebian provides neuroscience research s…   106       [OK]
ubuntu-debootstrap                DEPRECATED; use "ubuntu" instead                52        [OK]
ubuntu-upstart                    DEPRECATED, as is Upstart (find other proces…   115       [OK]
ubuntu/nginx                      Nginx, a high-performance reverse proxy & we…   112
ubuntu/squid                      Squid is a caching proxy for the Web. Long-t…   83
ubuntu/cortex                     Cortex provides storage for Prometheus. Long…   4
ubuntu/prometheus                 Prometheus is a systems and service monitori…   56
ubuntu/apache2                    Apache, a secure & extensible open-source HT…   70
ubuntu/kafka                      Apache Kafka, a distributed event streaming …   39
ubuntu/bind9                      BIND 9 is a very flexible, full-featured DNS…   74
ubuntu/mysql                      MySQL open source fast, stable, multi-thread…   59
ubuntu/zookeeper                  ZooKeeper maintains configuration informatio…   13
ubuntu/postgres                   PostgreSQL is an open source object-relation…   35
ubuntu/jre                        Distroless Java runtime based on Ubuntu. Lon…   13
ubuntu/redis                      Redis, an open source key-value store. Long-…   22
ubuntu/dotnet-aspnet              Chiselled Ubuntu runtime image for ASP.NET a…   17
ubuntu/grafana                    Grafana, a feature rich metrics dashboard & …   9
ubuntu/dotnet-deps                Chiselled Ubuntu for self-contained .NET & A…   13
ubuntu/memcached                  Memcached, in-memory keyvalue store for smal…   5
ubuntu/dotnet-runtime             Chiselled Ubuntu runtime image for .NET apps…   14
ubuntu/prometheus-alertmanager    Alertmanager handles client alerts from Prom…   9
ubuntu/cassandra                  Cassandra, an open source NoSQL distributed …   2
root@ubuntu:~#
```

- stars:

  `docker search` 명령어를 사용하면 Docker Hub에서 이미지를 검색

- official: 실제로 공식화된 컨테이너 이미지

- -filter 옵션을 이용하여 stars의 점수 3점 이상만 출력

  ```
  root@container:~# docker search --filter stars=3 busybox
  ```

```
root@ubuntu:~# docker search --filter stars=3 busybox
NAME                      DESCRIPTION                                     STARS     OFFICIAL
busybox                   Busybox base image.                             3210      [OK]
yauritux/busybox-curl     Busybox with CURL                               25
radial/busyboxplus        Full-chain, Internet enabled, busybox made f…   56
arm64v8/busybox           Busybox base image.                             8
odise/busybox-curl                                                        4
s390x/busybox             Busybox base image.                             3
arm32v7/busybox           Busybox base image.                             10
i386/busybox              Busybox base image.                             3
root@ubuntu:~#
```

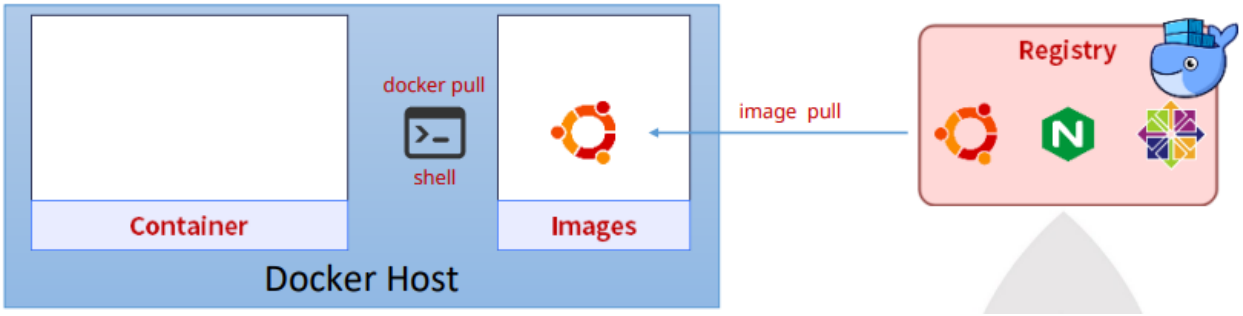
- -filter 옵션을 이용하여 공식(official) 이미지 와 stars의 점수 3점 이상만 출력

  ```
  root@container:~#docker search --filter is-official=true --filter stars=3 busybox
  ```

```
root@ubuntu:~#  docker search --filter is-official=true --filter stars=3 busybox
NAME      DESCRIPTION           STARS     OFFICIAL
busybox   Busybox base image.   3210      [OK]
root@ubuntu:~#
```

docker pull

registry로 이미지를 다운로드 받음



- `run` : 가져와서 실행하는 것 (`pull` + `start`)

- 로컬 이미지 저장소에 있는 이미지를 확인

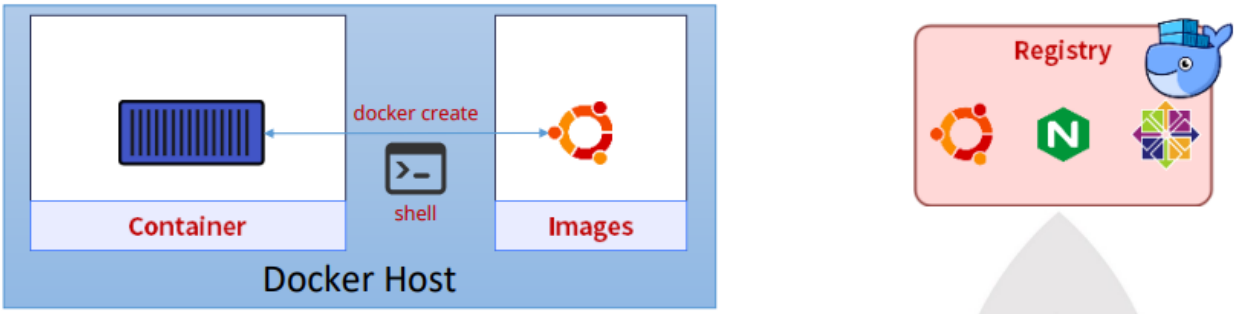
  `docker images`

  `docker image ls`

docker create

Docker 이미지를 기반으로 새로운 컨테이너를 생성

컨테이너화 하는 것

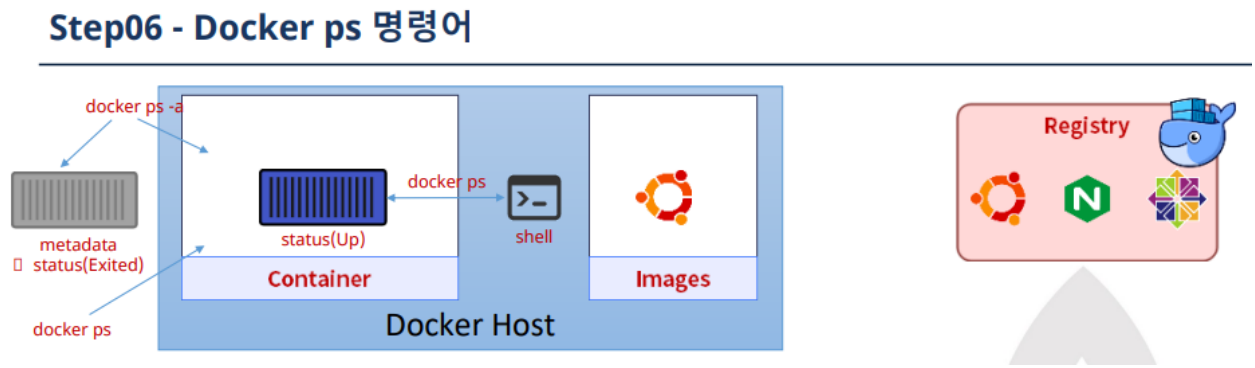


```
root@ubuntu:~# docker create -it --name mzc-ct01 ubuntu
e1c85df71ef429e26f45135fe7b6e60eb84390f459926b9bd85ba90a1d79f4ac
```

```
# docker create -it --name mzc-ct01 ubuntu
or
# docker create -it ubuntu
# docker create -it -p 8080:80 --name mzc-ct01 ubuntu
```

- 옵션
  - 접근 포트

docker start, docker ps



- `docker ps`

  현재 동작중인 이미지만 나옴

```
root@ubuntu:~# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

- `docker ps -a`

  STATUS(Exited) 상태 즉 stop 상태의 컨테이너도 출력

```
root@ubuntu:~# docker ps -a
CONTAINER ID   IMAGE         COMMAND       CREATED              STATUS                      PORTS     NAMES
e1c85df71ef4   ubuntu        "/bin/bash"   About a minute ago   Created                               mzc-ct01
d6052a8aae79   hello-world   "/hello"      21 minutes ago       Exited (0) 21 minutes ago             agitated_blackburn
7fcd7ea1a7b8   hello-world   "/hello"      53 minutes ago       Exited (0) 53 minutes ago             laughing_aryabhata
ede76e729ef2   hello-world   "/hello"      53 minutes ago       Exited (0) 53 minutes ago             musing_beaver
```

- `docker start {이미지 이름}`

  컨테이너 시작

```
root@ubuntu:~# docker start mzc-ct01
mzc-ct01
root@ubuntu:~# docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED             STATUS         PORTS     NAMES
e1c85df71ef4   ubuntu    "/bin/bash"   About an hour ago   Up 3 seconds             mzc-ct01
root@ubuntu:~#
```

Step06 - Docker ps 명령어

docker ps [OPTIONS]

실행 중인 컨테이너 리스트를 확인 합니다. -a 옵션을 사용하면 중지된 컨테이너도 볼 수 있습니다.
docker ps 명령어는 docker 에 메타데이터를 이용하여 정보를 출력합니다. docker 메타데이터 기본 위치 값은
docker host의 "/var/lib/docker" 디렉토리에 위치 하고 있습니다.

Aliases

docker container ls, docker container list, docker container ps, docker ps

Options

-a, --all : running 및 stop 상태의 모든 컨테이너를 볼 수 있습니다.

-s, --size : 컨테이너의 전체 파일 사이즈를 표시 합니다.

docker attach

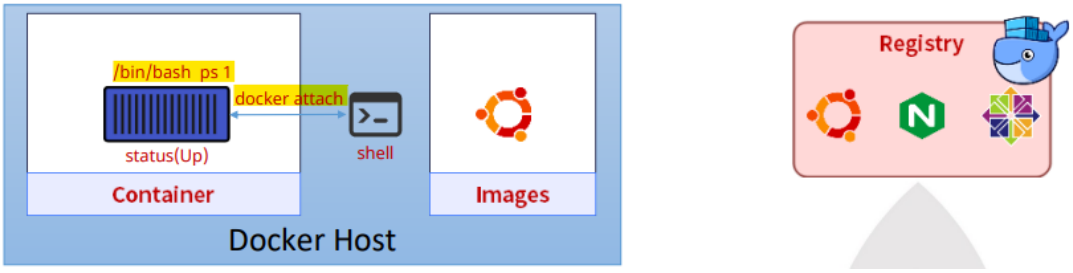
우분트 서버 안에 또다른 우분트 상태

컨테이너 접속하는 명령어

docker attach [OPTIONS] CONTAINER

Local 표준 입력, 출력 및 오류 스트림을 실행 중인 컨테이너에 연결합니다. 즉 ctrl +pq 명령어를 이용하여 컨테이너를 중지하지 않고 docker host로 복귀한 경우에 다시 실행 중인 컨테이너에 접속 하기 위해 사용합니다.
Docker attach 명령어는 컨테이너 내부에서 현재 실행 중인 프로세스에 연결하기 때문에, 해당 프로세스가 종료되면 컨테이너 연결도 종료됩니다.

Step07 - Docker attach 명령어




```
root@ubuntu:~# docker attach mzc-ct01
root@e1c85df71ef4:/# read escape sequence
root@ubuntu:~#
```

ctrl+p+q

docker attach 명령어는 컨테이너가 실행중인 상태에서만 사용할 수 있습니다. 이미 실행 중인 컨테이너에 연결하여 터머널과 상호 작용하거나 실행 중인 프로세스의 출력을 확인할 때 유용합니다. 컨테이너 실행 상태로 빠져 나오기 위해서는 ctrl+pq 키 조합을 사용합니다. exit 명령어로 입력할 경우 컨테이너는 종료 됩니다.

docker exec 명령어는 컨테이너가 실행중인 상태에서 새로운 프로세스를 생성하여 접속하는 방식 입니다. 이 경우 exit 명령어를 사용해도 기존 컨테이너는 종료 되지 않습니다.

docker exec

실행중인 컨테이너에 새로운 프로세스를 생성하여 연결하고자 할 때 사용

- 옵션

Options

-d,	--detach	: 백그라운드로 명령을 실행 합니다.
-e,	--env	: 컨테이너 환경변수를 지정합니다.
-i,	--interactive	: 컨테이너에 대한 인터렉티브(대화식) 쉘을 엽니다.
-t,	--tty	: 컨테이너에 tty를 제공합니다

  - `-d`
    - zjsxpdlsjfmf qorrmfkdnsem gudxofh rPthr tkdydgkrp gown0
- docker exec 명령어를 이용하여 실행 중인 컨테이너에 명령어를 실행

```
root@ubuntu:~# docker exec -it mzc-ct01 /bin/bash
root@e1c85df71ef4:/#
```

docker stop

컨테이너를 중지

실행 중인 컨테이너를 중단하고 리소스를 반환

```
root@ubuntu:~# docker stop mzc-ct01
mzc-ct01
```

docker restart

컨테이너를 재시작

실행 중인 컨테이너를 docker stop 과 docker start 한 결과와 동일

```
root@ubuntu:~# docker restart mzc-ct01
mzc-ct01
```

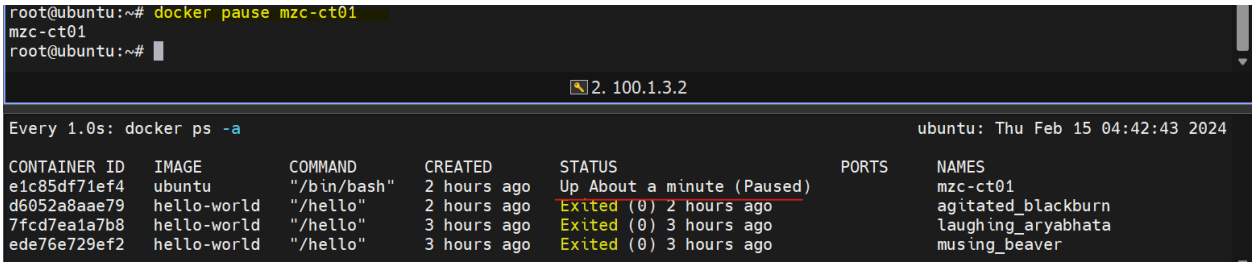
```
bash: docker: command not found
root@e1c85df71ef4:/# read escape sequence
root@ubuntu:~#
root@ubuntu:~#
root@ubuntu:~#
root@ubuntu:~# docker stop mzc-ct01
mzc-ct01
root@ubuntu:~#
root@ubuntu:~#
root@ubuntu:~# docker restart mzc-ct01
mzc-ct01
root@ubuntu:~#
root@ubuntu:~# docker restart mzc-ct01
mzc-ct01
root@ubuntu:~#
```

2. 100.1.3.2

```
Every 1.0s: docker ps -a                                        ubuntu: Thu Feb 15 04:41:15 2024

CONTAINER ID   IMAGE         COMMAND       CREATED       STATUS                     PORTS     NAMES
e1c85df71ef4   ubuntu        "/bin/bash"   2 hours ago   Up 7 seconds                         mzc-ct01
d6052a8aae79   hello-world   "/hello"      2 hours ago   Exited (0) 2 hours ago               agitated_blackburn
7fcd7ea1a7b8   hello-world   "/hello"      3 hours ago   Exited (0) 3 hours ago               laughing_aryabhata
ede76e729ef2   hello-world   "/hello"      3 hours ago   Exited (0) 3 hours ago               musing_beaver
```

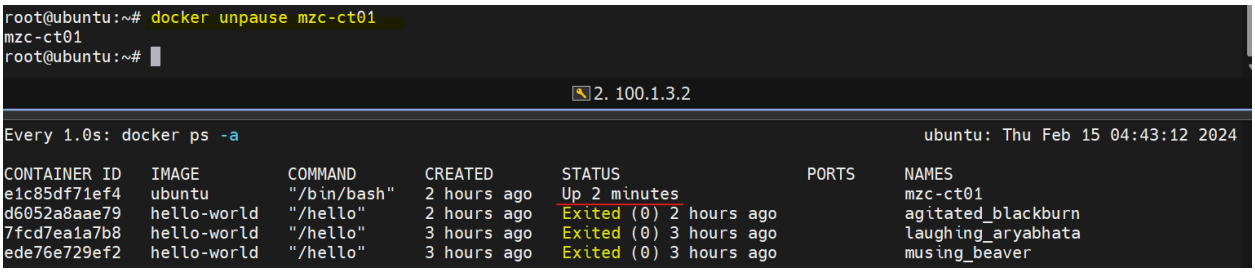
docker pause

실행 중인 컨테이너를 죽이지(die) 않고 일시중지



docker unpause

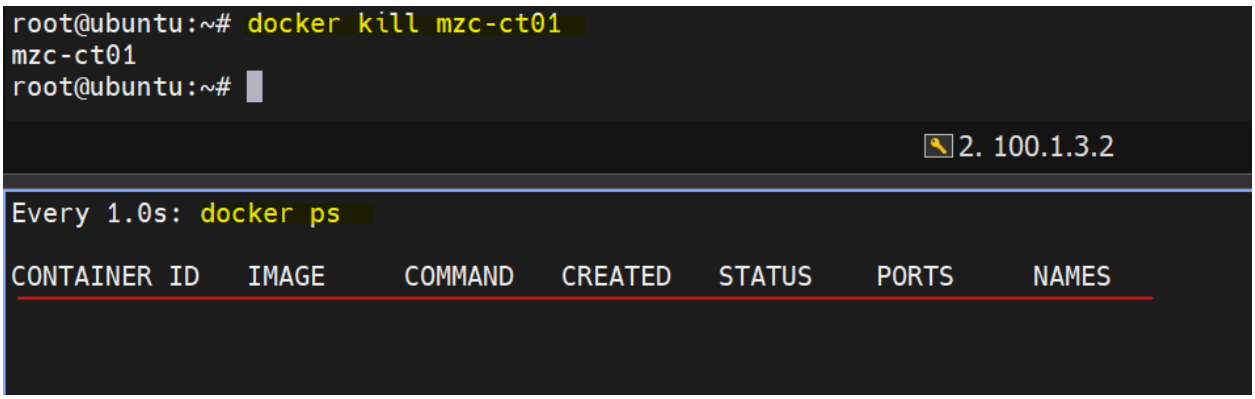
일시중지 된 컨테이너 프로세스를 다시 활성화(Up)



docker kill

동작 중인(running) 컨테이너를 강제 종료

데이터 손실이 발생할 수 있음



docker rm

컨테이너 지우는 것

```
root@ubuntu:~# docker rm -f mzc-ct01
mzc-ct01
root@ubuntu:~#
```

```
Every 1.0s: docker ps -a                                                ubuntu: Thu Feb

CONTAINER ID   IMAGE         COMMAND    CREATED       STATUS                     PORTS     NAMES
d6052a8aae79   hello-world   "/hello"   2 hours ago   Exited (0) 2 hours ago             agitated_blackburn
7fcd7ea1a7b8   hello-world   "/hello"   3 hours ago   Exited (0) 3 hours ago             laughing_aryabhata
ede76e729ef2   hello-world   "/hello"   3 hours ago   Exited (0) 3 hours ago             musing_beaver
```

docker rmi

이미지 지우는 것

Options

-f, --force Running 중인 컨테이너를 이미지삭제(테스트 필요)

- 원래는 docker image로 'ubuntu', 'hello-world'가 있었음

```
root@ubuntu:~# watch -n 1 docker images
```

```
Every 1.0s: docker images

REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
ubuntu        latest    fd1d8f58e8ae   2 weeks ago     77.9MB
hello-world   latest    d2c94e258dcb   9 months ago    13.3kB
```

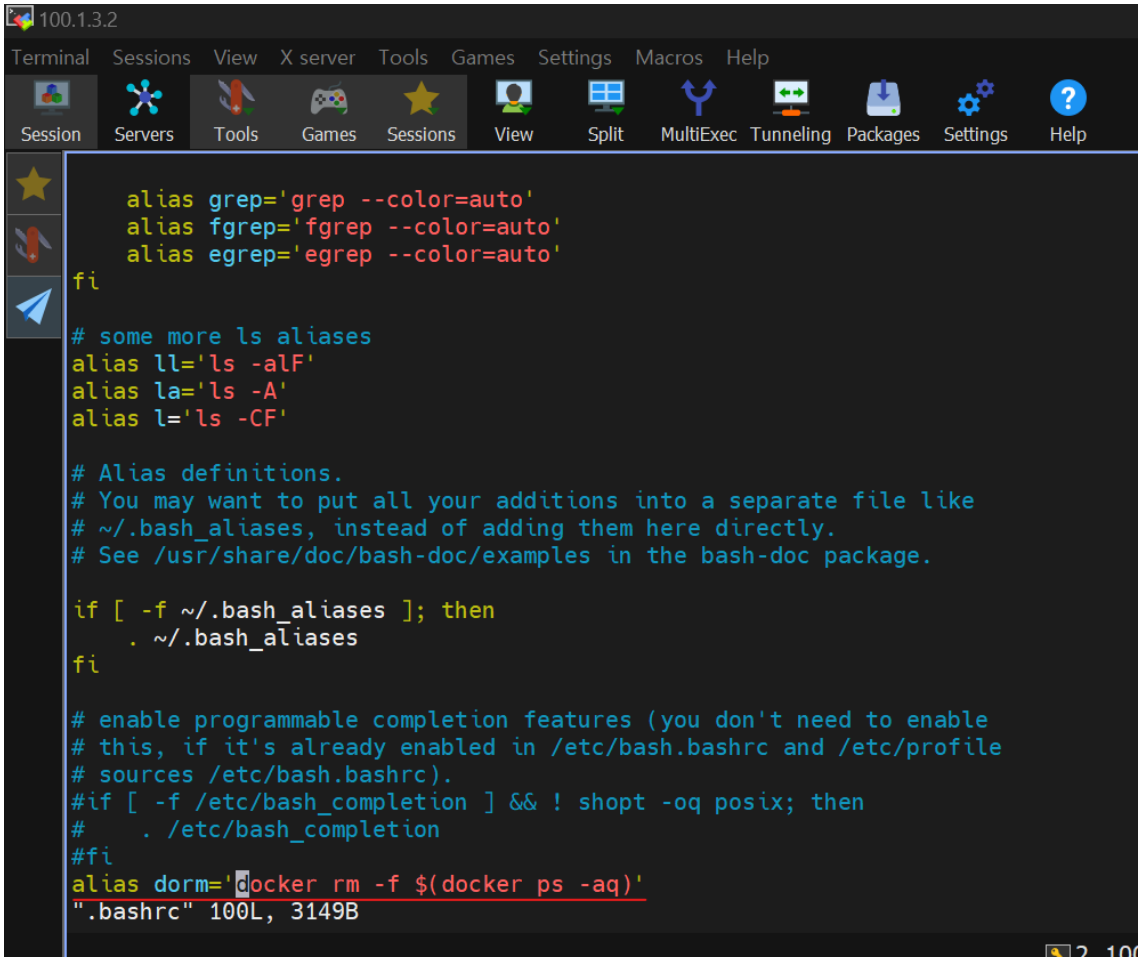
- 'ubuntu' 이미지를 강제로 삭제함

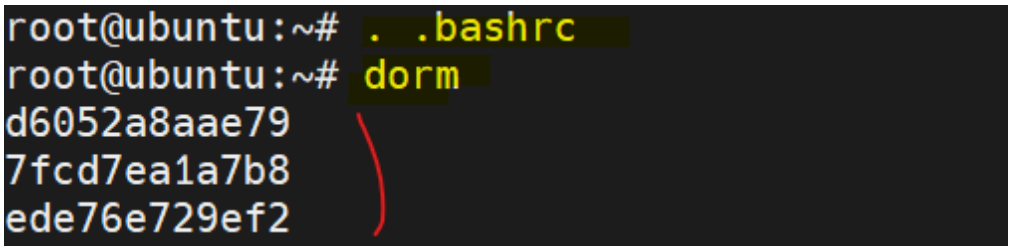
```
root@ubuntu:~# docker rmi -f ubuntu
Untagged: ubuntu:latest
Untagged: ubuntu@sha256:e9569c25505f33ff72e88b2990887c9dcf230f23259da296eb814fc2b41af999
Deleted: sha256:fd1d8f58e8aedc22ec0a3a7ce1a33de544a596eaa6cdb842f1af7c5e081d453f
Deleted: sha256:1a102d1cac2bdae8a0160ac4365d4f8653e9d6da56c793a665d556ae07fb7f82
root@ubuntu:~#
```

2. 100.1.3.2

```
Every 1.0s: docker images

REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world   latest    d2c94e258dcb   9 months ago   13.3kB
```

- `root@ubuntu:~# vim .bashrc`
  - 아래 입력하며 파일 수정





docker run

Docker 이미지를 기반으로 컨테이너를 생성 및 실행하는 명령어

- **restart 옵션**

  컨테이너가 예기치 않게 종료되었을 때 도커 엔진이 컨테이너를 자동으로 다시 시작하도록 지정

docker stats

Docker 상에서 작동하는 컨테이너 가동 상태를 확인

Step02 - Docker stats 출력 항목 설명

항목	설명
CONTAINER ID	컨테이너 식별자
NAME	컨테이너명
CPU %	CPU 사용률
MEM USAGE/LIMIT	메모리 사용량 / 컨테이너에서 사용할 수 있는 메모리 제한
MEM %	메모리 사용률
NET I/O	네트워크 I/O
BLOCK I/O	블록 I/O
PIDS	PID(windows 컨테이너 제외)

docker top

Docker host에서 지정된 컨테이너의 PID 확인 및 컨테이너 내부에 프로세스를 확인

docker inspect

Docker Object의 메타정보(metadata) 즉 상세한 설정 정보, 구성, 네트워크 설정, 마운트된 볼륨 등에 대한 정보를 JSON 형식으로 출력

실습 2 - docker run

Step01

Step01 - Docker run 명령어

```
root@ubuntu:~# docker run -it --name mzc-ct01 ubuntu
```

watch -n 1 docker images

- 매 초마다 현재 시스템에 있는 Docker 이미지 목록을 업데이트하여 보여주는 명령어

```
Every 1.0s: docker images

REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
ubuntu        latest    fd1d8f58e8ae   2 weeks ago     77.9MB
hello-world   latest    d2c94e258dcb   9 months ago    13.3kB
```

watch -n 1 docker ps -a

```
Every 1.0s: docker ps -a

CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS          PORTS     NAMES
52d1e33ee58e   ubuntu    "/bin/bash"   12 minutes ago   Up 3 minutes              mzc-ct01
```

docker run -it -d --name msc-ct01 ubuntu

- Docker에서 이름이 `msc-ct01`인 새로운 컨테이너를 만들고, 해당 컨테이너를 백그라운드로 실행하여 터미널에 연결된 상호작용 모드(`-it`)로 실행

- `-it`: 컨테이너를 인터랙티브한 상호작용 모드로 실행하고, 터미널에 연결합니다.
- `-d`: 컨테이너를 백그라운드(background) 모드로 실행합니다.
- `--name msc-ct01`: 컨테이너의 이름을 `msc-ct01`로 설정합니다.
- `ubuntu`: 사용할 Docker 이미지를 지정합니다. 여기서는 Ubuntu 이미지를 사용하고 있습니다.

```
root@ubuntu:~#  docker run -it -d --name msc-ct01 ubuntu
61064386e88b1e5867263405e2ba802e83acc689516a753c46caac282fdbd208
```

Step01 - 컨테이너 실행 확인 및 접속

```
root@ubuntu:~# docker attach mzc-ct01
root@52d1e33ee58e:/# hostname
52d1e33ee58e
root@52d1e33ee58e:/# exit
exit
root@ubuntu:~#
```

- docker attach //컨테이너 접속하는 명령어

```
root@ubuntu:~# docker start mzc-ct01
mzc-ct01
```

- docker start //컨테이너 시작

```
root@ubuntu:~#  docker attach mzc-ct01
root@52d1e33ee58e:/#  touch doc01.txt
root@52d1e33ee58e:/# ls
bin  boot  dev  doc01.txt  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@52d1e33ee58e:/# read escape sequence
root@ubuntu:~#
```

- 현재 디렉토리에 "doc01.txt"라는 이름의 빈 텍스트 파일을 만드는 명령

```
root@ubuntu:~# docker exec -it mzc-ct01 /bin/bash
root@52d1e33ee58e:/#  ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 06:42 pts/0    00:00:00 /bin/bash
root          10       0  0 06:54 pts/1    00:00:00 /bin/bash
root          18      10  0 06:55 pts/1    00:00:00 ps -ef
root@52d1e33ee58e:/# ls
bin  boot  dev  doc01.txt  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@52d1e33ee58e:/# exit
exit
root@ubuntu:~#
```

- `docker exe` // 실행 중인 컨테이너에 새로운 명령을 실행할 수 있음
- `-it` 옵션 //상호작용 모드를 사용하고 터미널을 연결한다는 것
- `/bin/bash` //실행할 셸의 종류를 지정
- `ps -ef` //시스템에 실행 중인 모든 프로세스의 목록이 표시됨

Step02

Step02 - Docker stats 명령어

`docker stats` //작동하는 컨테이너 가동 상태를 확인 때

```
CONTAINER ID   NAME       CPU %     MEM USAGE / LIMIT     MEM %     NET I/O       BLOCK I/O     PIDS
61064386e88b   msc-ct01   0.00%     812KiB / 1.918GiB     0.04%     23.6kB / 0B   205kB / 0B    1
52d1e33ee58e   mzc-ct01   0.00%     1.262MiB / 1.918GiB   0.06%     20.6kB / 0B   5.64MB / 0B   1
```

Step02 - Docker stats 실행 내용

```
root@ubuntu:~# docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS          PORTS     NAMES
61064386e88b   ubuntu    "/bin/bash"   23 minutes ago   Up 23 minutes             msc-ct01
52d1e33ee58e   ubuntu    "/bin/bash"   37 minutes ago   Up 21 minutes             mzc-ct01
```

`docker stats mzc-ct01`

```
CONTAINER ID   NAME       CPU %     MEM USAGE / LIMIT     MEM %     NET I/O       BLOCK I/O     PIDS
52d1e33ee58e   mzc-ct01   0.00%     1.262MiB / 1.918GiB   0.06%     21.9kB / 0B   5.64MB / 0B   1
```

Step03

Step03 - Docker run 의 Restart 정책

`docker run --restart` //컨테이너가 예기치 않게 종료되었을 때, 도커 엔진이 컨테이너를 자동으로 다시 시작하도록 지정

`docker pull centos`

```
root@ubuntu:~# docker pull centos
Using default tag: latest
latest: Pulling from library/centos
a1d0c7532777: Pull complete
Digest: sha256:a27fd8080b517143cbbbab9dfb7c8571c40d67d534bbdee55bd6c473f432b177
Status: Downloaded newer image for centos:latest
docker.io/library/centos:latest
root@ubuntu:~#
```

`docker run -it -d --restart=always --name mzc-centos-ct02 centos`

```
root@ubuntu:~#  docker run -it -d --restart=always --name mzc-centos-ct02 centos
b6a0b106a5dd60623f46f8f7d4a031cc79b3f91132ecc4293fece381ecd17dd9
```

- `-restart=always` 옵션은 컨테이너가 비정상적으로 종료될 때 자동으로 다시 시작되도록 설정, 이 옵션을 사용하면 Docker 데몬이 컨테이너를 자동으로 다시 시작
- `-name mzc-centos-ct02` 옵션은 컨테이너의 이름을 "mzc-centos-ct02"로 설정
- `centos`는 사용할 Docker 이미지를 지정. 여기서는 CentOS 이미지를 사용

Step03 - Restart 정책 생성 및 확인

`watch -n 1 docker ps -a`

```
Every 1.0s: docker ps -a

CONTAINER ID   IMAGE     COMMAND       CREATED             STATUS          PORTS     NAMES
b6a0b106a5dd   centos    "/bin/bash"   16 minutes ago      Up 16 minutes             mzc-centos-ct02
61064386e88b   ubuntu    "/bin/bash"   59 minutes ago      Up 59 minutes             msc-ct01
52d1e33ee58e   ubuntu    "/bin/bash"   About an hour ago   Up 57 minutes             mzc-ct01
```

`docker run -it -d --restart=on-failure:3 --name msc-centos-ct02 centos`

- `-restart=on-failure:3` 옵션은 컨테이너가 비정상적으로 종료될 때, 최대 3번까지 재시도하고, 계속해서 실패하면 재시도하지 않도록 설정합니다. 이 옵션은 컨테이너가 세 번 연속으로 실패할 때까지 자동으로 다시 시작됩니다.
- `-name msc-centos-ct02` 옵션은 컨테이너의 이름을 "msc-centos-ct02"로 설정합니다.
- `centos`는 사용할 Docker 이미지를 지정합니다. 여기서는 CentOS 이미지를 사용하고 있습니다.

`docker top mzc-centos-ct02`

docker top //실행 중인 프로세스 목록을 표시

실습 2- 연습문제

esc attach

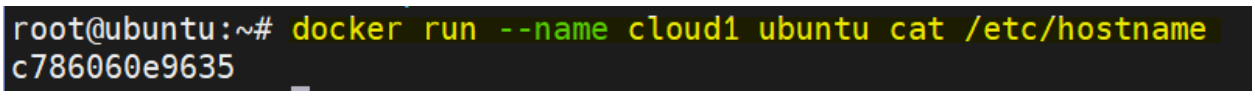
bin bash 넣을때 아닐때 차이

hostpc → virtualbox의 컨테이너 웹서버 접

hostpc → ssh 접근 허용

1. 다양하게 이미지 실행하기

1. docker run --name cloud1 ubuntu cat /etc/hostname



💡 **명령어 해석**

- `ubuntu` 이미지를 기반으로 새로운 Docker 컨테이너를 시작하기
- `--name cloud1` // 컨테이너의 이름을 "cloud1"으로 설정하기
- `cat /etc/hostname` //컨테이너 내에서 실행될 명령으로, 컨테이너의 호스트 이름을 표시

- hostname(container ID)을 출력해줘

2. docker run --name cloud2 ubuntu ping www.google.com –c 3

💡 **명령어 해석**

- `ubuntu` 이미지를 기반으로 새로운 Docker 컨테이너를 시작하기
- `--name cloud2` //컨테이너의 이름을 "cloud2"로 설정하기
- `ping www.google.com -c 3` //컨테이너 내에서 실행될 명령으로, "www.google.com"에 대해 3번의 핑을 보냄

- **오류 뜸**
  - docker: Error response from daemon: failed to create task for container: failed to create shim task: OCI runtime create failed: runc create failed: unable to start container process: exec: "ping": executable file not found in $PATH: unknown.

```
root@ubuntu:~# docker run --name cloud2 ubuntu ping www.google.com –c 3
docker: Error response from daemon: failed to create task for container: failed to create shim task: OCI runtime create failed: runc create failed: unable to start container process: exec: "ping": executable file not found in $PATH: unknown.
```

  - 이 오류는 컨테이너 내에서 "ping" 명령을 실행할 수 없다는 것을 나타냄
    - 컨테이너 내에서 필요한 실행 파일이 제대로 설치되어 있지 않은 경우에 발생할 수 있음
- **해결방안**

`docker run --name cloud3 ubuntu apt-get update && apt-get install -y iputils-ping && ping www.google.com -c 3`

💡 **명령어 해석**

- 새로운 Docker 컨테이너를 시작함
- `-name cloud3` //컨테이너의 이름을 "cloud3"으로 설정
- `ubuntu` 이미지를 기반으로 컨테이너를 시작하기
- `apt-get update && apt-get install -y iputils-ping` //컨테이너 내에서 실행

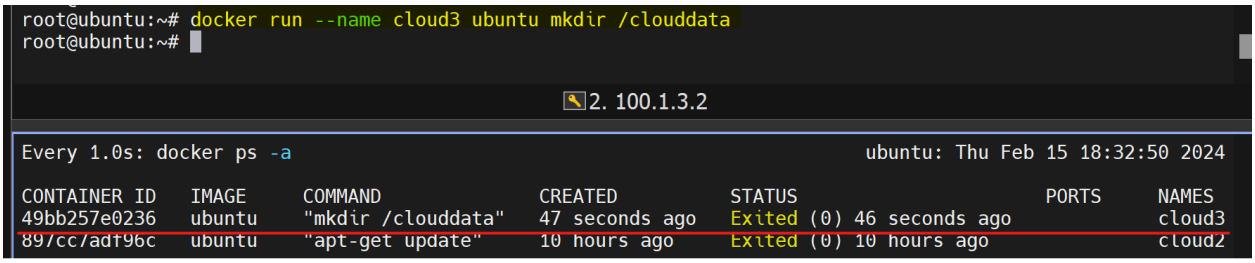
  // 우분투 패키지 관리자인 apt를 사용하여 패키지 목록을 업데이트하고, iputils-ping 패키지를 설치함
  - iputils-ping 패키지 ⇒ "ping" 명령을 포함하고 있음
- `&&` 연산자는 앞선 명령이 성공적으로 실행된 경우에만 다음 명령을 실행합니다.
- `ping www.google.com -c 3` 명령은 컨테이너 내에서 실행됩니다. 이 명령은 "www.google.com"에 대해 3번의 핑을 보내고 결과를 출력합니다.

```
root@ubuntu:~# docker run --name cloud2 ubuntu apt-get update && apt-get install -y iputils-ping && ping www.google.com -c 3
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy InRelease [270 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [1070 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [17.5 MB]
Get:7 http://archive.ubuntu.com/ubuntu jammy/restricted amd64 Packages [164 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [266 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy/main amd64 Packages [1792 kB]
Get:10 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1343 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [1822 kB]
Get:12 http://archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [50.4 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1735 kB]
Get:14 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [28.1 kB]
Get:15 http://archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [50.4 kB]
Get:16 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [1456 kB]
Get:17 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [44.6 kB]
Get:18 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [1784 kB]
Fetched 29.7 MB in 17s (1725 kB/s)
Reading package lists...
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
iputils-ping is already the newest version (3:20211215-1).
iputils-ping set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 26 not upgraded.
PING www.google.com (172.217.25.164) 56(84) bytes of data.
64 bytes from sin01s16-in-f4.1e100.net (172.217.25.164): icmp_seq=1 ttl=57 time=37.6 ms
64 bytes from syd09s13-in-f4.1e100.net (172.217.25.164): icmp_seq=2 ttl=57 time=36.3 ms
64 bytes from syd09s13-in-f4.1e100.net (172.217.25.164): icmp_seq=3 ttl=57 time=38.7 ms

--- www.google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 36.264/37.502/38.670/0.983 ms
root@ubuntu:~#
```

3. docker run --name cloud3 ubuntu mkdir /clouddata

💡 **명령어 해석**

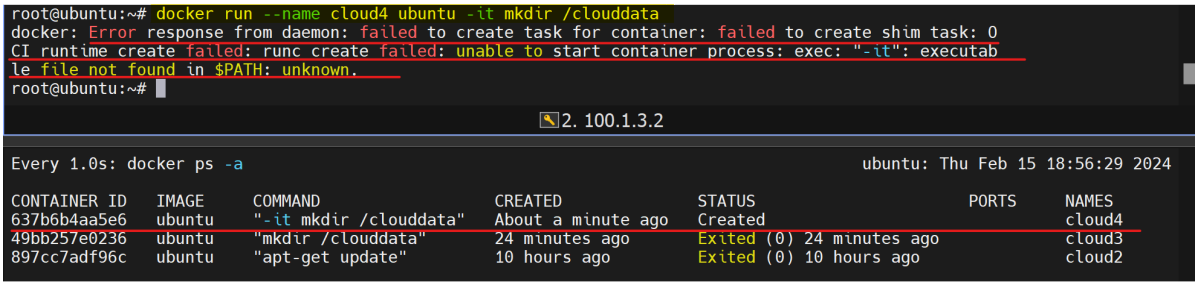
- `-name cloud3` //컨테이너의 이름을 "cloud3"으로 설정
- `ubuntu` 이미지를 기반으로 컨테이너를 시작
- `mkdir /clouddata` //컨테이너 내에서 실행, 컨테이너 내에 "/clouddata"라는 디렉토리를 만들기



- 정상적으로 실행되고 종료

4. docker run --name cloud4 ubuntu –it mkdir /clouddata

- 오류 뜸



  - `-it` 옵션의 위치가 잘못됨
  - option자리는 다음과 같음

```
docker run [OPTIONS] IMAGE [COMMAND] [ARG...]
```

- 해결방안

`docker run -it --name cloud5 ubuntu mkdir /clouddata`

```
root@ubuntu:~# docker run -it --name cloud5 ubuntu mkdir /clouddata
root@ubuntu:~#
```

2. 100.1.3.2

```
Every 1.0s: docker ps -a                                    ubuntu: Thu Feb 15 18:58:11 2024

CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS                       PORTS     NAMES
b5ee4965fa16   ubuntu    "mkdir /clouddata"       6 seconds ago    Exited (0) 5 seconds ago               cloud5
637b6b4aa5e6   ubuntu    "-it mkdir /clouddata"   3 minutes ago    Created                                cloud4
49bb257e0236   ubuntu    "mkdir /clouddata"       26 minutes ago   Exited (0) 26 minutes ago              cloud3
897cc7adf96c   ubuntu    "apt-get update"         10 hours ago     Exited (0) 10 hours ago                cloud2
```

5. docker run --name cloud5 ubuntu –it /bin/bash

- 오류 뜸

```
root@ubuntu:~# docker run --name cloud5 ubuntu –it /bin/bash
docker: Error response from daemon: failed to create task for container: failed to create shim task: OCI runtim
e create failed: runc create failed: unable to start container process: exec: "–it": executable file not found
in $PATH: unknown.
root@ubuntu:~#
```

2. 100.1.3.2

```
Every 1.0s: docker ps -a                                    ubuntu: Thu Feb 15 19:05:30 2024

CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS                       PORTS     NAMES
6d7c6f35b436   ubuntu    "–it /bin/bash"          About a minute ago   Created                                cloud5
637b6b4aa5e6   ubuntu    "-it mkdir /clouddata"   10 minutes ago       Created                                cloud4
49bb257e0236   ubuntu    "mkdir /clouddata"       33 minutes ago       Exited (0) 33 minutes ago              cloud3
897cc7adf96c   ubuntu    "apt-get update"         10 hours ago         Exited (0) 10 hours ago                cloud2
```

  - `-it` 옵션의 위치가 잘못됨

- 해결방안

```
docker run --name cloud5 -it ubuntu /bin/bash
```

  - 올바르게 수정된 점은 옵션(`--name`)과 옵션 값(`cloud5`) 사이에 공백이 있어야 하며, `-it` 옵션은 이미지(`ubuntu`) 앞에 위치해야 함

```
root@ubuntu:~# docker run --name cloud5 -it ubuntu /bin/bash
root@b707d2436769:/#
```

2. 100.1.3.2

```
Every 1.0s: docker ps -a                                    ubuntu: Thu Feb 15 19:06:28

CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS                       PORTS     NAMES
b707d2436769   ubuntu    "/bin/bash"              3 seconds ago    Up 2 seconds                           cloud5
637b6b4aa5e6   ubuntu    "-it mkdir /clouddata"   11 minutes ago   Created                                cloud4
49bb257e0236   ubuntu    "mkdir /clouddata"       34 minutes ago   Exited (0) 34 minutes ago              cloud3
897cc7adf96c   ubuntu    "apt-get update"         10 hours ago     Exited (0) 10 hours ago                cloud2
```

- 추가 명령어 입력

```
root@ubuntu:~# docker run --name cloud5 -it ubuntu /bin/bash
root@d5b40382ee78:/# mkdir /clouddata
root@d5b40382ee78:/# Exit
bash: Exit: command not found
root@d5b40382ee78:/# exit
exit
root@ubuntu:~# docker start cloud5
cloud5
```

`attach` 는 컨테이너 접속하는 명령어이기 때문에 적합하지 x

→ so `exec` 명령어를 사용해야함(실행 중인 컨테이너에 새로운 프로세스를 생성하여 연결하고자 할 때 사용)

```
root@ubuntu:~# docker attach cloud5 ls -l /clouddata
unknown shorthand flag: 'l' in -l
See 'docker attach --help'.
root@ubuntu:~# docker exec cloud5 ls -l /clouddata
total 0
```

6. docker run --name cloud6 –it –d ubuntu /bin/bash

> 💡 **명령어 해석**
> cloud6"라는 이름의 Docker 컨테이너가 시작되고, 해당 컨테이너는 인터랙티브한 상호작용 모드로 실행됨. 그 후 컨테이너 내에서 /bin/bash 셸이 실행됨

```
root@ubuntu:~# docker run --name cloud6 -it -d ubuntu /bin/bash
e2b0db6e379582b5f9f35380da28d9da534dc8f19f9035bbed20313a01399705
root@ubuntu:~#
```

2. 100.1.3.2

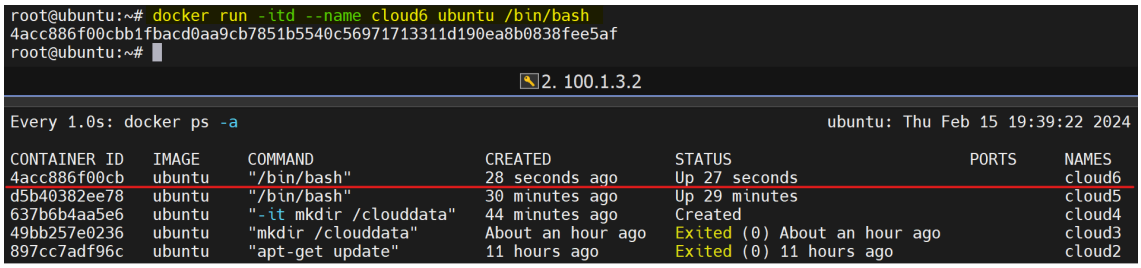
```
Every 1.0s: docker ps -a                                        ubuntu: Thu Feb 15 19:28:45

CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS                      PORTS     NAMES
e2b0db6e3795   ubuntu    "/bin/bash"              5 seconds ago    Up 4 seconds                          cloud6
d5b40382ee78   ubuntu    "/bin/bash"              19 minutes ago   Up 18 minutes                         cloud5
637b6b4aa5e6   ubuntu    "-it mkdir /clouddata"   33 minutes ago   Created                               cloud4
49bb257e0236   ubuntu    "mkdir /clouddata"       56 minutes ago   Exited (0) 56 minutes ago             cloud3
897cc7adf96c   ubuntu    "apt-get update"         11 hours ago     Exited (0) 11 hours ago               cloud2
```

- 오류가 뜨지는 않았지만 명령어 자체에 문제가 발생함
- 오류
  - `-it` 옵션과 `-d` 옵션은 함께 사용할 수 x
    - `-it` 옵션: 컨테이너를 인터랙티브한 모드로 실행하고 터미널에 연결
    - `-d` 옵션: 컨테이너를 백그라운드 모드로 실행하는 옵션

    - so 터미널에 연결하지 x
- 해결방안

  `docker run -itd --name cloud6 ubuntu /bin/bash`

    - `-itd` 옵션: 컨테이너를 인터랙티브한 상호작용 모드로 실행하고, 터미널에 연결하며, 백그라운드에서 실행하는 옵션



7. docker run --name cloudweb -d -p 80:80 httpd

💡 **명령어 해석**

- `--name cloudweb` //컨테이너의 이름을 "cloudweb"으로 설정
- `-d` //컨테이너를 백그라운드 모드로 실행
- `p 80:80` // 호스트의 포트 80을 컨테이너의 포트 80으로 포트 포워딩함(호스트의 HTTP 요청을 컨테이너 내의 웹 서버로 전달하는 데 사용됨)
- `httpd` //"httpd" 이미지를 기반으로 새로운 Docker 컨테이너를 시작

- 오류뜸

```
root@ubuntu:~# docker run --name cloudweb -d -p 80:80 httpd
Unable to find image 'httpd:latest' locally
latest: Pulling from library/httpd
e1caac4eb9d2: Pull complete
87b0fe460fd9: Pull complete
4f4fb700ef54: Pull complete
9cebd3e3b523: Pull complete
e9304da947c5: Pull complete
b60d4b66b268: Pull complete
Digest: sha256:104f07de17ee186c8f37b9f561e04fbfe4cf080d78c6e5f3802fd08fd118c3da
Status: Downloaded newer image for httpd:latest
23a6b56ab61f15725c6f83a1e7173cb9c82a5e5719944e0c901295ae51e70f45
docker: Error response from daemon: driver failed programming external connectivity on endpoint cloudweb (8329b7b3c7ea
f1ddcf5c81fd83d636dd08ade0a9d07d841811858317e2faa21b): Error starting userland proxy: listen tcp4 0.0.0.0:80: bind: ad
dress already in use.
root@ubuntu:~#
root@ubuntu:~#
```

2. 100.1.3.2

```
Every 1.0s: docker ps -a                                                    ubuntu: Thu Feb 15 19:44:23 2024

CONTAINER ID   IMAGE     COMMAND                  CREATED             STATUS                          PORTS     NAMES
23a6b56ab61f   httpd     "httpd-foreground"       2 minutes ago       Created                                   cloudweb
4acc886f00cb   ubuntu    "/bin/bash"              5 minutes ago       Up 5 minutes                              cloud6
d5b40382ee78   ubuntu    "/bin/bash"              35 minutes ago      Up 34 minutes                             cloud5
637b6b4aa5e6   ubuntu    "-it mkdir /clouddata"   49 minutes ago      Created                                   cloud4
49bb257e0236   ubuntu    "mkdir /clouddata"       About an hour ago   Exited (0) About an hour ago              cloud3
897cc7adf96c   ubuntu    "apt-get update"         11 hours ago        Exited (0) 11 hours ago                   cloud2
```

- 호스트의 포트 80이 이미 사용하고 있어서 Docker가 해당 포트를 할당할 수 없는 경우에 발생함
-

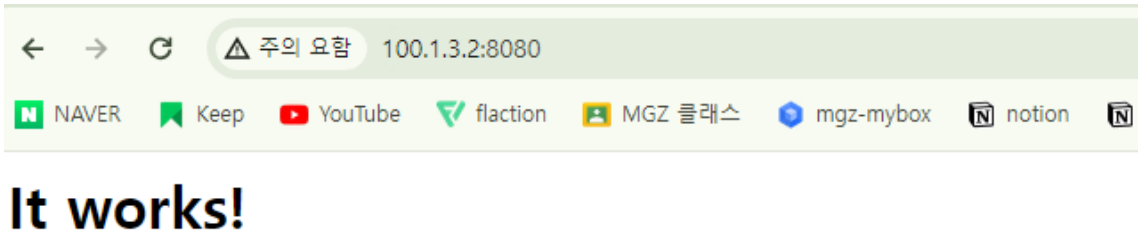
```
root@ubuntu:~# curl localhost
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
>
<html xmlns="http://www.w3.org/1999/xhtml">
  <!--
    Modified from the Debian original for Ubuntu
    Last updated: 2022-03-22
    See: https://launchpad.net/bugs/1966004
  -->
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

- 해결방안
  - 다른 포트랑 매핑해주기

```
root@ubuntu:~# docker run --name cloudweb -d -p 8080:80 httpd
f840cc63dd35ab8a76354ec5f4a55c86abb9448730eaba882e7d24d0d993797a
root@ubuntu:~#
```

2. 100.1.3.2

```
Every 1.0s: docker ps -a                                                    ubuntu: Thu Feb 15 20:54:44 2024

CONTAINER ID   IMAGE     COMMAND                  CREATED             STATUS                          PORTS                                   NAMES
f840cc63dd35   httpd     "httpd-foreground"       2 minutes ago       Up 2 minutes                    0.0.0.0:8080->80/tcp, :::8080->80/tcp   cloudweb
4acc886f00cb   ubuntu    "/bin/bash"              About an hour ago   Up About an hour                                                        cloud6
d5b40382ee78   ubuntu    "/bin/bash"              2 hours ago         Up 2 hours                                                              cloud5
637b6b4aa5e6   ubuntu    "-it mkdir /clouddata"   2 hours ago         Created                                                                 cloud4
49bb257e0236   ubuntu    "mkdir /clouddata"       2 hours ago         Exited (0) 2 hours ago                                                  cloud3
897cc7adf96c   ubuntu    "apt-get update"         12 hours ago        Exited (0) 12 hours ago                                                 cloud2
```



2. 설치 및 container 실행하기

Docker 설치는 이미 사전에 완료한 상태에서 실습 진행함

1. Docker 정보 확인하기

- **docker --help**
  - Docker CLI(Command Line Interface)의 사용법과 가능한 옵션들을 확인

```
root@ubuntu:~# docker --help

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Common Commands:
  run         Create and run a new container from an image
  exec        Execute a command in a running container
  ps          List containers
  build       Build an image from a Dockerfile
  pull        Download an image from a registry
  push        Upload an image to a registry
  images      List images
  login       Log in to a registry
  logout      Log out from a registry
  search      Search Docker Hub for images
  version     Show the Docker version information
  info        Display system-wide information

Management Commands:
  builder     Manage builds
  buildx*     Docker Buildx (Docker Inc., v0.12.1)
  compose*    Docker Compose (Docker Inc., v2.24.5)
  container   Manage containers
  context     Manage contexts
  image       Manage images
  manifest    Manage Docker image manifests and manifest lists
  network     Manage networks
  plugin      Manage plugins
  system      Manage Docker
  trust       Manage trust on Docker images
  volume      Manage volumes
```

- **docker -v**
  - 설치된 Docker의 버전을 표시

```
root@ubuntu:~# docker -v
Docker version 25.0.3, build 4debf41
```

- docker version

- Docker Engine의 버전과 클라이언트의 정보를 표시
- Docker Engine의 버전, 클라이언트 버전, 빌드 정보 등을 자세히 보여줌

```
root@ubuntu:~# docker version
Client: Docker Engine - Community
 Version:           25.0.3
 API version:       1.44
 Go version:        go1.21.6
 Git commit:        4debf41
 Built:             Tue Feb  6 21:14:17 2024
 OS/Arch:           linux/amd64
 Context:           default

Server: Docker Engine - Community
 Engine:
  Version:          25.0.3
  API version:      1.44 (minimum version 1.24)
  Go version:       go1.21.6
  Git commit:       f417435
  Built:            Tue Feb  6 21:14:17 2024
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.6.28
  GitCommit:        ae07eda36dd25f8a1b98dfbf587313b99c0190bb
 runc:
  Version:          1.1.12
  GitCommit:        v1.1.12-0-g51d5e94
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
root@ubuntu:~#
```

- docker info
  - Docker 시스템의 상세한 정보를 표시
  - Docker의 설정, 컨테이너, 이미지, 스토리지 드라이버, 네트워크 및 기타 관련 정보를 보여줌

```
root@ubuntu:~# docker info
Client: Docker Engine - Community
 Version:    25.0.3
 Context:    default
 Debug Mode: false
 Plugins:
  buildx: Docker Buildx (Docker Inc.)
    Version:  v0.12.1
    Path:     /usr/libexec/docker/cli-plugins/docker-buildx
  compose: Docker Compose (Docker Inc.)
    Version:  v2.24.5
    Path:     /usr/libexec/docker/cli-plugins/docker-compose

Server:
 Containers: 6
  Running: 3
  Paused: 0
  Stopped: 3
 Images: 4
 Server Version: 25.0.3
 Storage Driver: overlay2
  Backing Filesystem: extfs
  Supports d_type: true
  Using metacopy: false
  Native Overlay Diff: true
  userxattr: false
 Logging Driver: json-file
 Cgroup Driver: systemd
 Cgroup Version: 2
 Plugins:
  Volume: local
  Network: bridge host ipvlan macvlan null overlay
  Log: awslogs fluentd gcplogs gelf journald json-file local splunk syslog
 Swarm: inactive
 Runtimes: io.containerd.runc.v2 runc
 Default Runtime: runc
 Init Binary: docker-init
```

2. 최신 Ubuntu 버전 Docker images 다운로드하여 사용하기

`docker run -it ubuntu /bin/bash`

💡 **명령어 해석**

- 새로운 Ubuntu 컨테이너를 시작하고 해당 컨테이너 내에서 Bash 셸을 실행함
- 컨테이너 내에서 작업 가능
- bash 셸에서 명령을 실행하고, 파일을 수정하거나 설치할 수 있음

```
root@ubuntu:~# docker run -it ubuntu /bin/bash
root@d2fdb637a3af:/#
```

2. 100.1.3.2

```
Every 1.0s: docker ps -a                                                                    ubuntu: Thu Feb 15 21:09:07 2

CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS                      PORTS                                   NAMES
d2fdb637a3af   ubuntu    "/bin/bash"              15 seconds ago   Up 14 seconds                                               practical_poincare
f840cc63dd35   httpd     "httpd-foreground"       17 minutes ago   Up 17 minutes               0.0.0.0:8080->80/tcp, :::8080->80/tcp   cloudweb
4acc886f00cb   ubuntu    "/bin/bash"              2 hours ago      Up 2 hours                                                  cloud6
d5b40382ee78   ubuntu    "/bin/bash"              2 hours ago      Up 2 hours                                                  cloud5
637b6b4aa5e6   ubuntu    "-it mkdir /clouddata"   2 hours ago      Created                                                     cloud4
49bb257e0236   ubuntu    "mkdir /clouddata"       3 hours ago      Exited (0) 3 hours ago                                      cloud3
897cc7adf96c   ubuntu    "apt-get update"         12 hours ago     Exited (0) 12 hours ago                                     cloud2
```

- 컨테이너 ip 주소 출력

```
root@18dc781460cb:/# hostname -i
172.17.0.6
```

- 현재 시스템의 리눅스 배포판에 대한 정보 표시

  `cat /etc/*-release`

```
root@18dc781460cb:/# cat /etc/*-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=22.04
DISTRIB_CODENAME=jammy
DISTRIB_DESCRIPTION="Ubuntu 22.04.3 LTS"
PRETTY_NAME="Ubuntu 22.04.3 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04.3 LTS (Jammy Jellyfish)"
VERSION_CODENAME=jammy
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=jammy
root@18dc781460cb:/#
```

`apt-get update`

3. 최신 CentOS 버전 Docker images 다운로드하여 사용하기