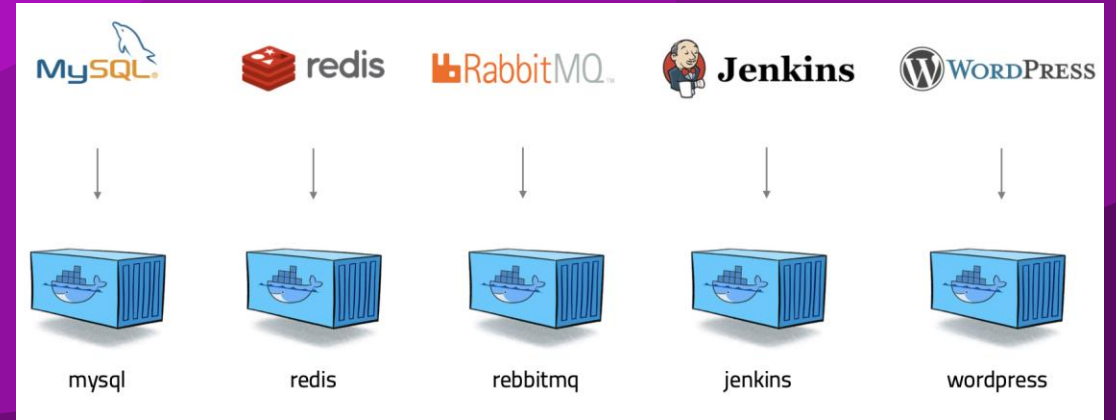


# 5장. Docker

서버를 배포하는 방법인 Docker에 대해 알아본다.  
docker-compose를 사용하여 연계된 서비스들을 한번에  
배포할 수 있는 방법에 대해 알아본다.

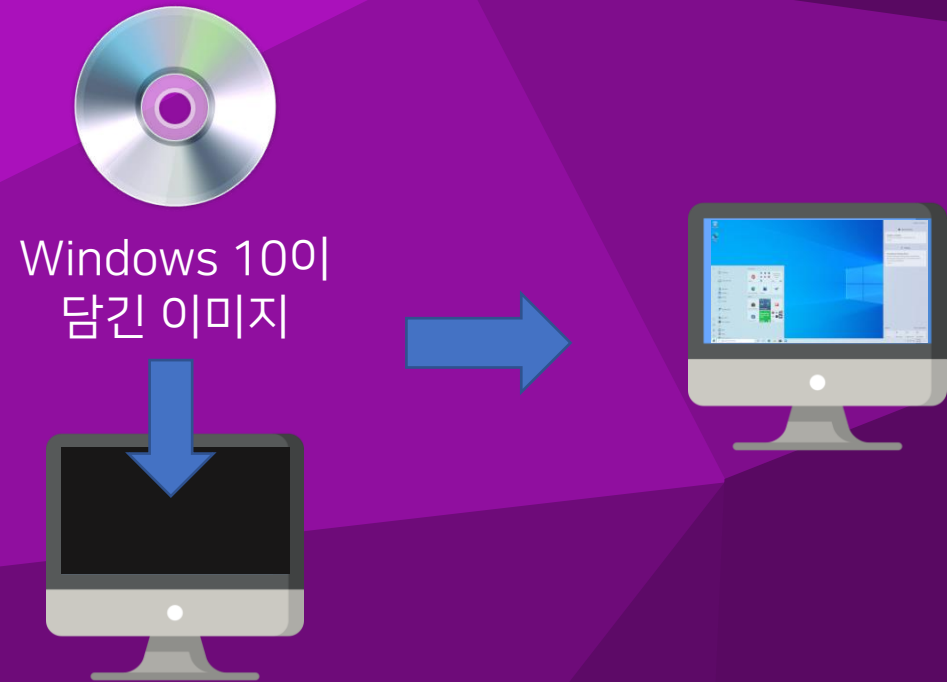
# Docker란?

- 컨테이너 기반의 가상화 플랫폼
- 오픈소스
- 운영체제(Windows, Mac OS 등)에 종속되지 않고 어디서든지 docker만 있다면 컨테이너를 실행할 수 있다.



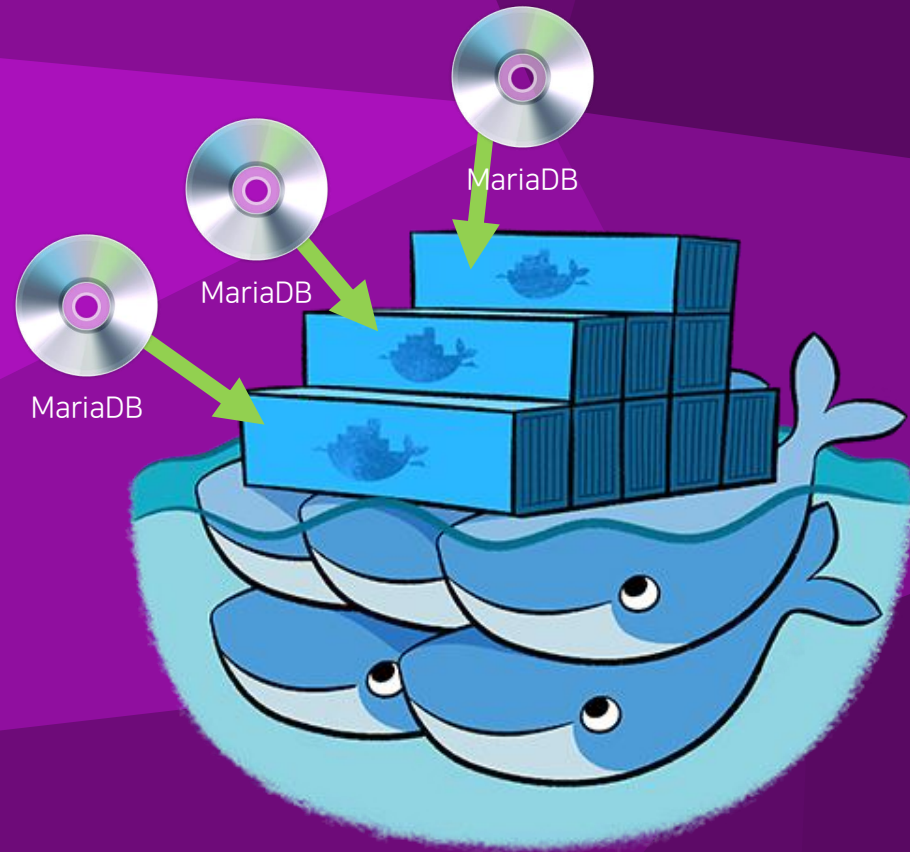
# 이미지라는 개념에 대해 알아보시다.

- Windows 10 운영체제를 “냉동” 시켜놓은 것이 Windows 10 이미지
- 이미지를 컴퓨터에 넣고 “해동” 하면 ... (설치)
- Windows 10 설치 완료!



# Docker도 이미지라는게 있어요.

- MariaDB 이미지를 받아 “해동”을 시키는데...
- 컨테이너를 만들고 해동
- 이미지 하나로 컨테이너를 2개든, 3개든 얼마든지 생성
- `sudo apt install mariadb`로 설치한다면 docker처럼 여러 개는 어려울지도



MariaDB

\*MariaDB는 데이터베이스 서버입니다

# Docker vs Without Docker

## Docker

- 설정은 환경 변수로
- 설치 및 설정이 간편
- 확장이 간편



## Without Docker

- 어디 있는지 모르 설정
- 확장이 불편



# [실습] Windows에서 Docker 사용하기

- Windows에서 docker를 설치하고 실행해본다.
- docker의 명령어들을 알아본다.
- 이미지를 다운받아 컨테이너를 생성해본다.
- Windows에서 학습한 내용은 Linux에서도 똑같이 적용이 된다.



docker

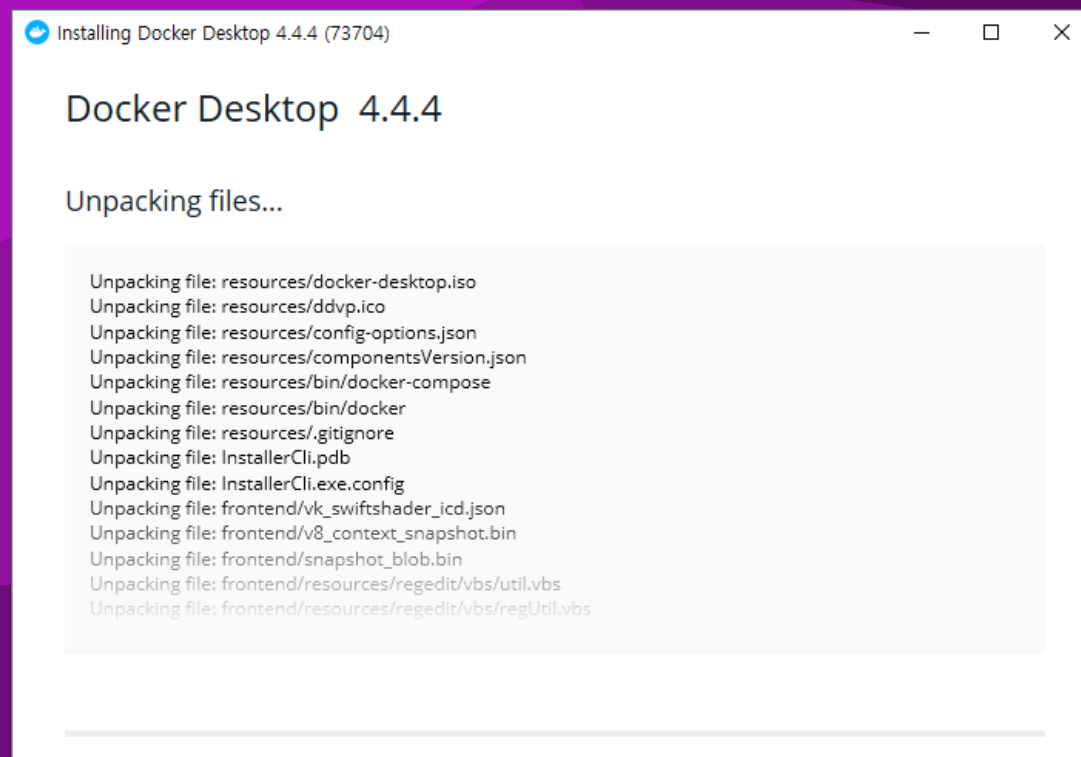
# [실습] Docker Desktop 설치

- 구글에 “docker”를 검색하고 Docker Desktop (Windows)를 설치한다.
- GUI 인터페이스를 제공하지만, CLI에서의 사용을 가정하고 커맨드 창에서만 사용하도록 한다.



# [실습] Docker Desktop 설치

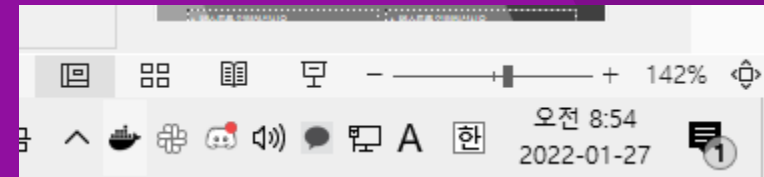
- 설치가 완료되면 자동으로 실행된다.
- 혹시 Docker가 꺼져 있다면 Windows 키를 누른 후 Docker 를 검색하고, 실행한다.





# [실습] Docker Desktop 설치

- 작업 표시줄 하단에 고래 모양의 아이콘이 생기는지 확인한 후 Docker 실습을 진행한다.



# Docker 컨테이너 생성 및 실행

- `docker run [OPTIONS] <IMAGE> [COMMAND ...]`
  - 이미지로 컨테이너를 생성하여 실행하는 명령이다.
- `docker run -d -p 3306:3306 --name db mariadb`
  - mariadb 컨테이너를 생성하고 실행하는 예제
  - `-d` 옵션은 백그라운드에서 실행되게 해준다.
  - `-p` 옵션은 컨테이너 내부의 포트와 외부의 포트를 연결해준다. (포트포워딩)
  - `--name` 옵션은 컨테이너의 이름을 지정할 수 있게 해준다. (추후 삭제, 중지 시 사용)
- 맨 뒤의 mariadb는 이미지 이름이며, 이는 <https://hub.docker.com>에서 다운받게 된다.
  - [https://hub.docker.com/\\_/mariadb](https://hub.docker.com/_/mariadb)

# docker run 옵션들

- -d
  - 컨테이너를 백그라운드에서 계속 실행하게 해주는 옵션 (detached mode)
- -p <포트:컨테이너 내부 포트>
  - 컨테이너 내부 포트를 외부의 포트로 연결해주는 옵션 (포트포워딩)
- --network <모드>
  - 네트워크 모드를 지정. host로 설정 시 -p 옵션 없이 컨테이너 내부 포트를 모두 사용할 수 있게 된다.
- --rm
  - 프로세스 종료 시 컨테이너 자동 제거. 컨테이너를 일회용으로 쓸 때 사용하는 옵션

# docker run 옵션들

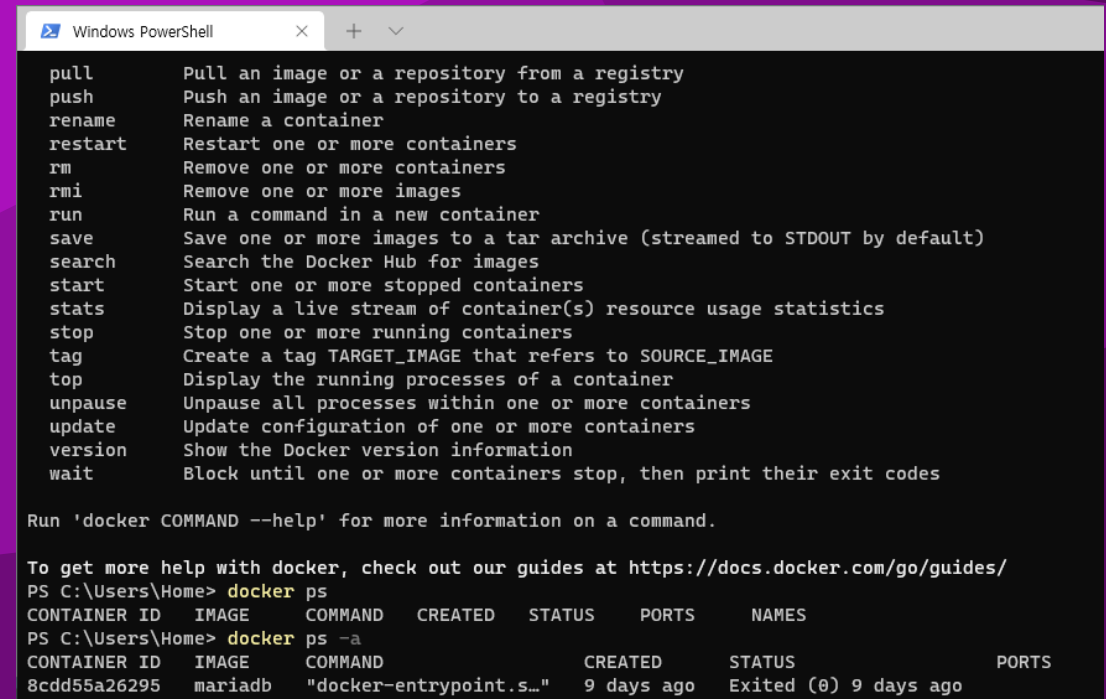
- -v <폴더:컨테이너 내부 폴더>
  - 외부의 폴더를 컨테이너 내부에서 사용할 수 있게 해주는 옵션
  - 컨테이너를 삭제하면 컨테이너 내부 데이터도 모두 삭제된다.
  - 따라서 중요한 데이터는 이 옵션을 사용하여 보존하도록 한다.
- --name <컨테이너 이름>
  - 컨테이너의 이름을 지정하는 옵션
  - 삭제나 중지시 컨테이너의 이름을 지정하도록 되어있다.

# docker run 옵션들

- -e <환경변수>
  - 컨테이너 내부의 환경 변수를 설정한다.
  - 대부분의 컨테이너는 환경 설정 파일 대신 환경 변수를 통해 설정을 한다.
- --restart <재부팅 정책>
  - 컨테이너 재시작 옵션이다.
  - no: 컨테이너를 재시작 시키지 않는다. (default)
  - on-failure: 컨테이너가 정상적으로 종료되지 않은 경우에만 재시작 시킨다.
  - always: 컨테이너를 항상 재시작 시킨다.
  - unless-stopped: 컨테이너를 직접 stop하기 전까지 항상 재시작 시킨다.

# Docker 명령어

- `docker stop <컨테이너 이름>`
  - 컨테이너를 정지한다.
- `docker start <컨테이너 이름>`
  - 정지된 컨테이너를 다시 시작시킨다.
- `docker rm <컨테이너 이름>`
  - 컨테이너를 삭제한다.
- `docker ps`
  - 실행중인 컨테이너 목록을 확인한다.
  - `-a` 옵션을 사용하면 정지된 컨테이너도 모두 확인이 가능하다. (`docker ps -a`)



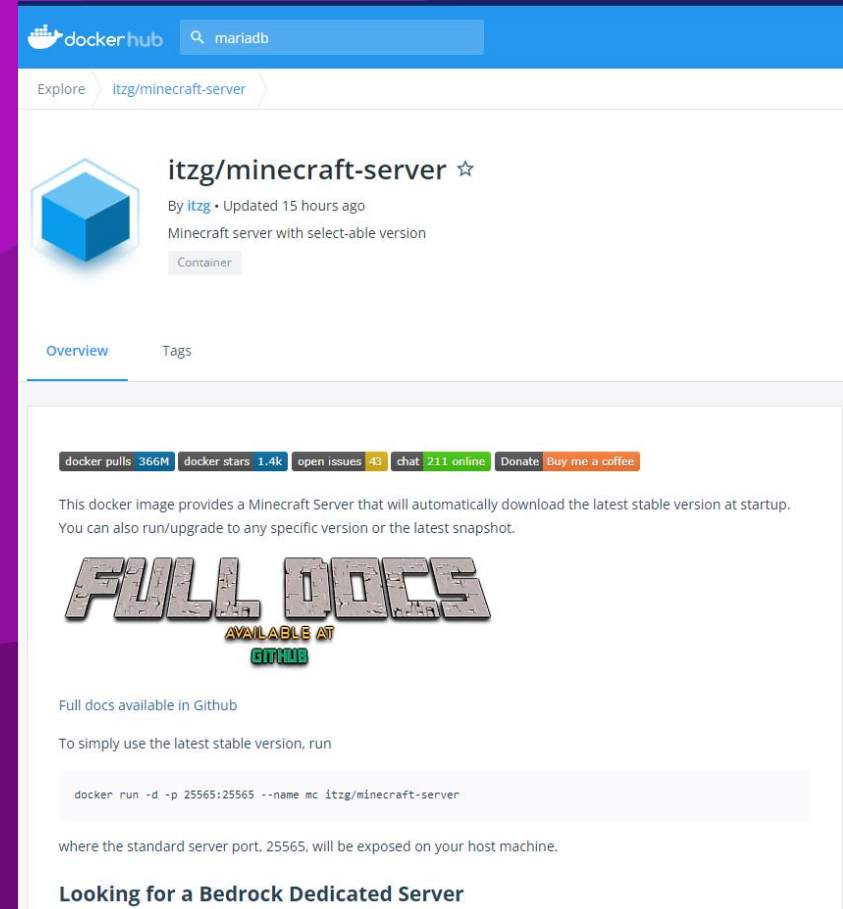
```
Windows PowerShell
pull      Pull an image or a repository from a registry
push      Push an image or a repository to a registry
rename    Rename a container
restart   Restart one or more containers
rm        Remove one or more containers
rmi       Remove one or more images
run       Run a command in a new container
save      Save one or more images to a tar archive (streamed to STDOUT by default)
search    Search the Docker Hub for images
start     Start one or more stopped containers
stats     Display a live stream of container(s) resource usage statistics
stop      Stop one or more running containers
tag       Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
top       Display the running processes of a container
unpause   Unpause all processes within one or more containers
update    Update configuration of one or more containers
version   Show the Docker version information
wait      Block until one or more containers stop, then print their exit codes

Run 'docker COMMAND --help' for more information on a command.

To get more help with docker, check out our guides at https://docs.docker.com/go/guides/
PS C:\Users\Home> docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
PS C:\Users\Home> docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED        STATUS      PORTS
8cdd55a26295   mariadb  "docker-entripoint.s..."  9 days ago    Exited (0) 9 days ago
```

# [실습] Docker로 구동기 실행하기

- 마인크래프트 구동기를 docker run 명령어로 구동해본다.
- itzg/minecraft-server 이미지를 사용한다.
  - <https://hub.docker.com/r/itzg/minecraft-server>



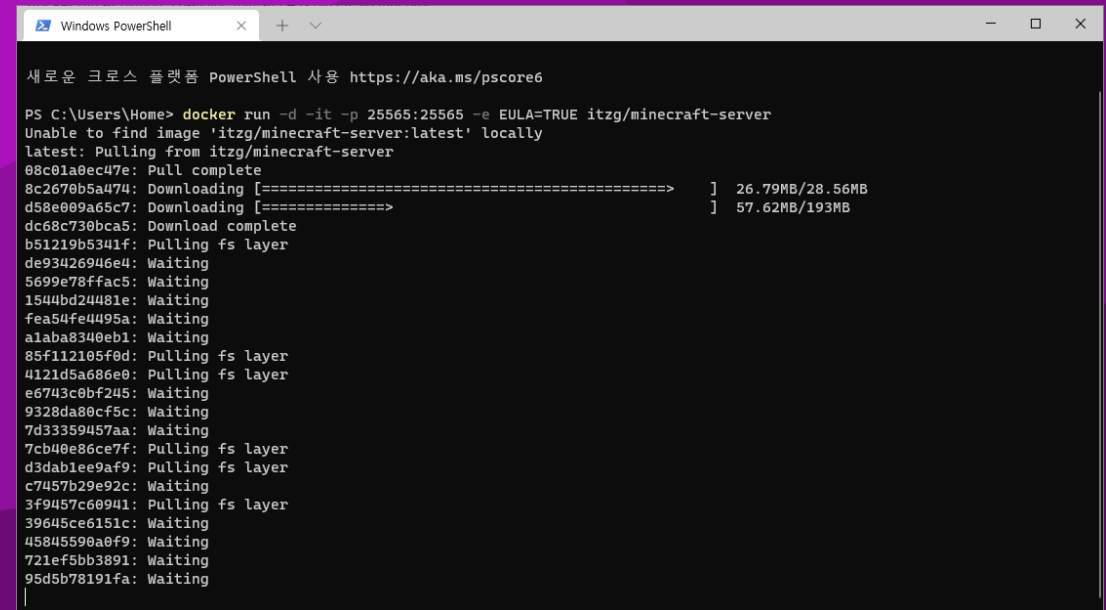
The screenshot shows the Docker Hub page for the `itzg/minecraft-server` image. The page header includes the Docker Hub logo and a search bar. Below the header, there's a navigation bar with "Explore" and "itzg/minecraft-server". The main content area features a blue cube icon, the image name `itzg/minecraft-server`, and a star icon. It also shows the author `itzg` and the update time "Updated 15 hours ago". A description states "Minecraft server with select-able version" and a "Container" tag. Below this, there are tabs for "Overview" and "Tags". A statistics bar shows "docker pulls: 366M", "docker stars: 1.4k", "open issues: 43", "chat: 211 online", and "Donate Buy me a coffee". The main text explains that the image provides a Minecraft Server that will automatically download the latest stable version at startup. A large "FULL DOCS AVAILABLE AT GITHUB" banner is displayed. Below the banner, it says "Full docs available in Github" and "To simply use the latest stable version, run". A code block shows the command: `docker run -d -p 25565:25565 --name mc itzg/minecraft-server`. A note mentions that the standard server port 25565 will be exposed on the host machine. At the bottom, there's a link "Looking for a Bedrock Dedicated Server".

# [실습] Docker로 구동기 실행하기

- Windows Terminal을 켜고, 아래의 명령을 입력한다.

```
docker run -d -it -p 25565:25565 -e EULA=TRUE itzg/minecraft-server
```

- -d 옵션을 주어 백그라운드에서 실행
- -it 옵션은 -i 옵션과 -t 옵션을 합친 것이며 키보드 입력을 가능하게 해준다.



```
Windows PowerShell
새로운 크로스 플랫폼 PowerShell 사용 https://aka.ms/pscore6

PS C:\Users\Home> docker run -d -it -p 25565:25565 -e EULA=TRUE itzg/minecraft-server
Unable to find image 'itzg/minecraft-server:latest' locally
latest: Pulling from itzg/minecraft-server
08c01a0ec47e: Pull complete
8c2670b5a474: Downloading [=====] 26.79MB/28.56MB
d58e009a65c7: Downloading [=====] 57.62MB/193MB
dc68c730bca5: Download complete
b51219b5341f: Pulling fs layer
de93426946e4: Waiting
5699e78ffac5: Waiting
1544bd24481e: Waiting
fea54fe4495a: Waiting
a1aba8340eb1: Waiting
85f112105f0d: Pulling fs layer
4121d5a686e0: Pulling fs layer
e6743c0bf245: Waiting
9328da80cf5c: Waiting
7d33359457aa: Waiting
7cb40e86ce7f: Pulling fs layer
d3dablee9af9: Pulling fs layer
c7457b29e92c: Waiting
3f9457c60941: Pulling fs layer
39645ce6151c: Waiting
45845590a0f9: Waiting
721ef5bb3891: Waiting
95d5b78191fa: Waiting
```

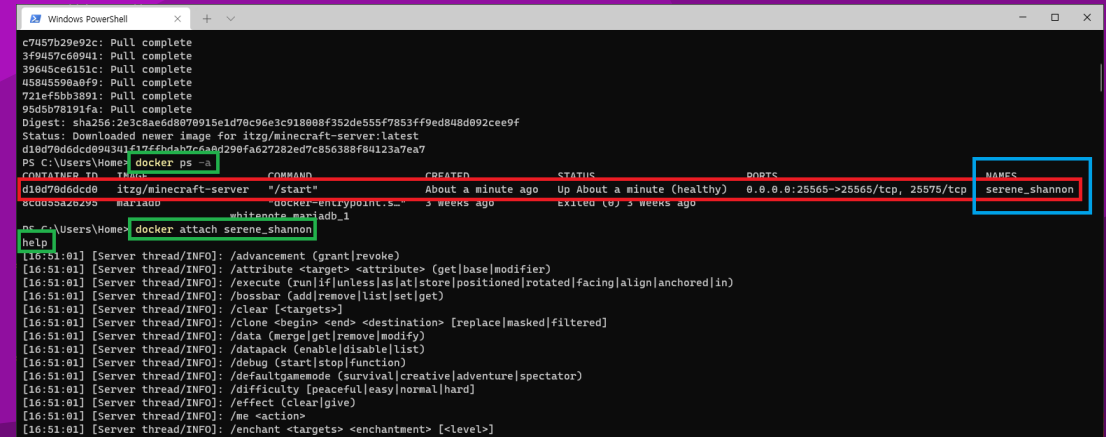


# [실습] Docker로 구동기 실행하기

- 명령 실행이 끝난 후 컨테이너의 이름을 확인한 후,
- docker 컨테이너 내 콘솔 화면으로 들어간다.

```
docker ps -a  
docker attach <컨테이너 이름>
```

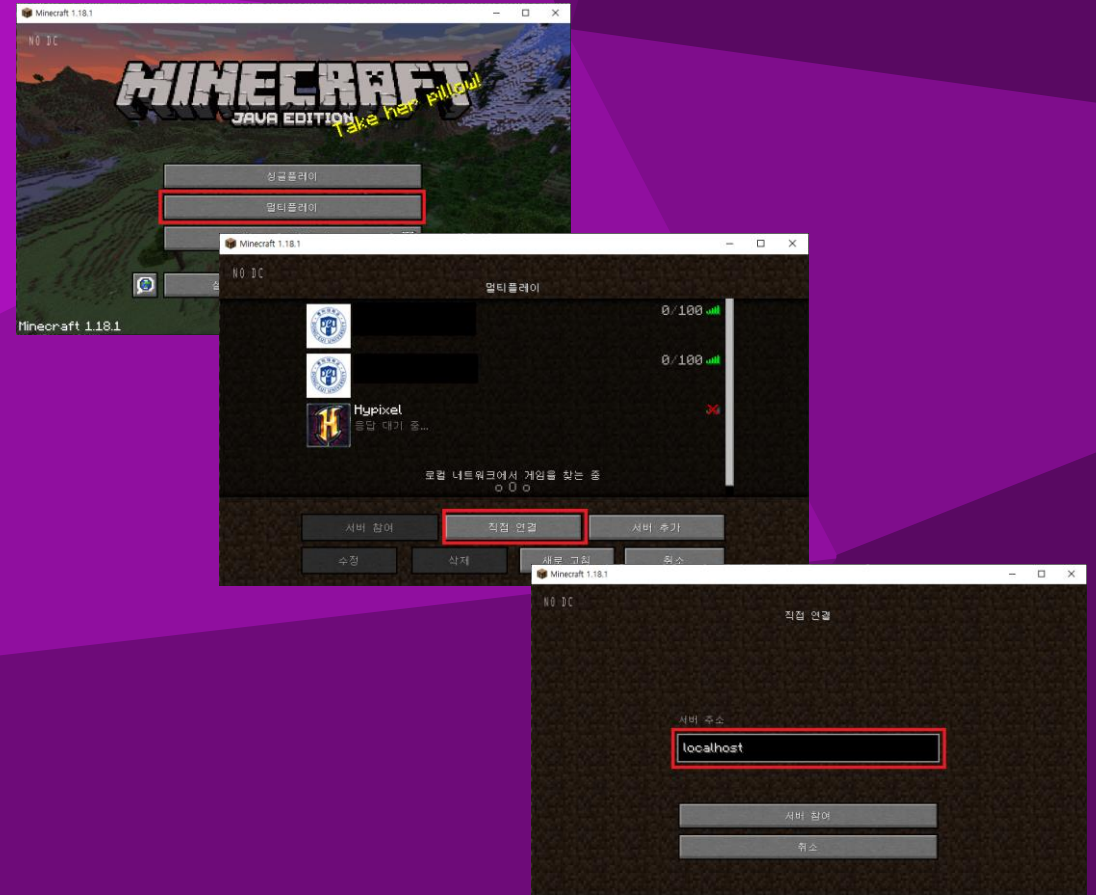
- docker attach는 현재 터미널을 컨테이너 터미널에 부착하는 명령이다.
- 즉, 구동기 콘솔을 사용할 수 있게 해준다.
- help 명령을 입력하여 정상 작동하는지 확인한다.



```
Windows PowerShell
c7457b29e92c: Pull complete
3f9457c60941: Pull complete
39645ce6151c: Pull complete
45845598a0f9: Pull complete
721ef5bb3891: Pull complete
95d5b78191fa: Pull complete
Digest: sha256:2e3c8a6d807091e1d70c9e3c918008f352de55f7853ff9ed848d092cee9f
Status: Downloaded newer image for itzg/minecraft-server:latest
d18d78ddcd941c12f6bhab7cdaa0290fa627282e07c856388f84123a7ea7
PS C:\Users\Home> docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS                               NAMES
d18d78ddcd9    itzg/minecra- "/start"                About a minute ago    Up About a minute (healthy)    0.0.0.0:25565->25565/tcp, 25575/tcp    serene_shannon
8c0d55a26293   mariadb    "docker-entrypoint.s..." 3 weeks ago        Exited (0) 3 weeks ago                               whiteoaks-mariadb_1
PS C:\Users\Home> docker attach serene_shannon
help
[16:51:01] [Server thread/INFO]: /advancement (grant|revoke)
[16:51:01] [Server thread/INFO]: /attribute <target> <attribute> (get|base|modifier)
[16:51:01] [Server thread/INFO]: /execute (run|if|unless|as|at|store|positioned|rotated|facing|align|anchored|in)
[16:51:01] [Server thread/INFO]: /bossbar (add|remove|list|set|get)
[16:51:01] [Server thread/INFO]: /clear [<targets>]
[16:51:01] [Server thread/INFO]: /clone <begin> <end> <destination> [replace|masked|filtered]
[16:51:01] [Server thread/INFO]: /data (merge|get|remove|modify)
[16:51:01] [Server thread/INFO]: /datapack (enable|disable|list)
[16:51:01] [Server thread/INFO]: /debug (start|stop|function)
[16:51:01] [Server thread/INFO]: /defaultgamemode (survival|creative|adventure|spectator)
[16:51:01] [Server thread/INFO]: /difficulty [peaceful|easy|normal|hard]
[16:51:01] [Server thread/INFO]: /effect (clear|give)
[16:51:01] [Server thread/INFO]: /me <action>
[16:51:01] [Server thread/INFO]: /enchant <targets> <enchantment> [<level>]
```

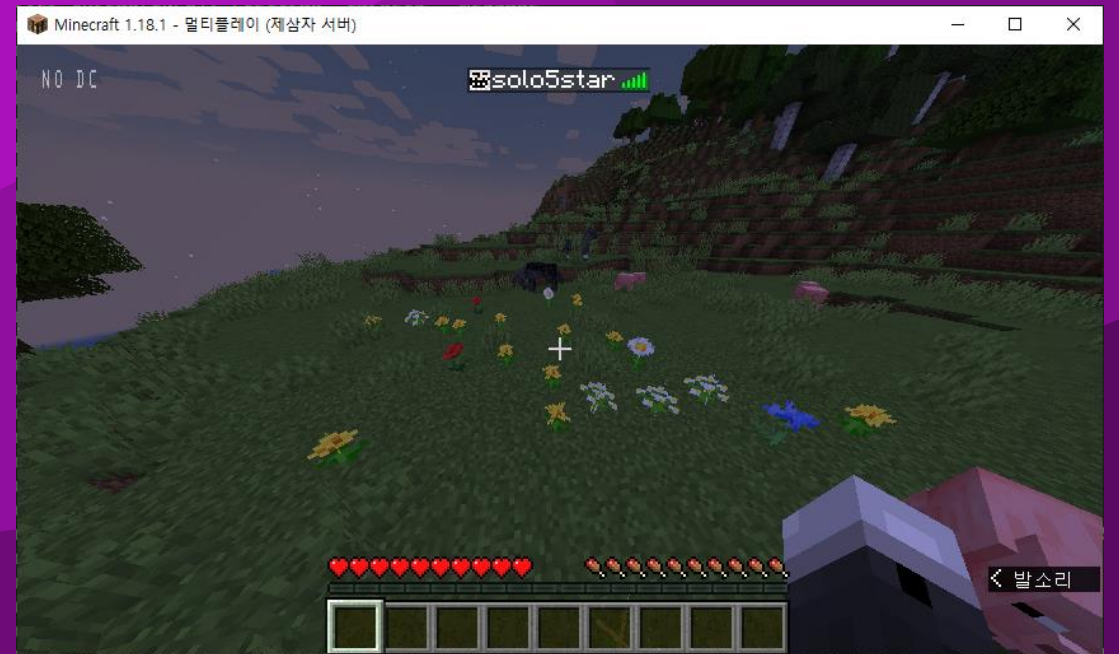
# [실습] Docker로 구동기 실행하기

- 서버에 접속하여 최종적인 확인을 마친다.
- 멀티플레이 -> 직접 연결 -> localhost 입력 -> 서버 참여
- 포트를 생략하면 기본 포트(25565)를 사용한다.
- <주소>:<포트 번호> 형식의 주소 입력도 가능하다.
  - 예) localhost:25565



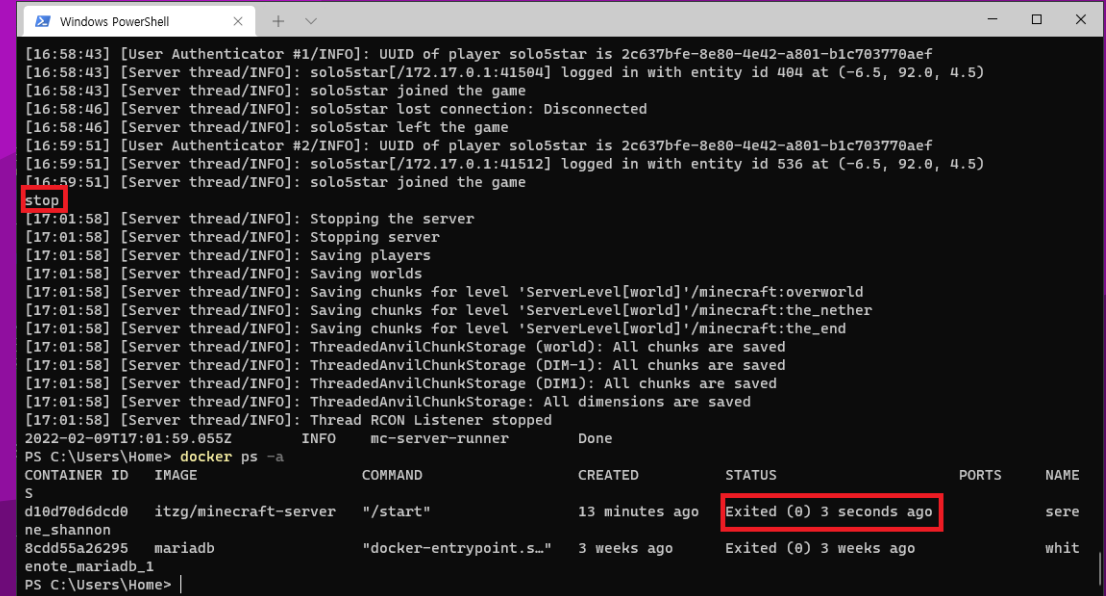
# [실습] Docker로 구동기 실행하기

- 접속이 잘 되는 것을 확인해보자.



# [실습] Docker로 구동기 실행하기

- 콘솔 화면에서도 플레이어 접속을 확인할 수 있다.
- 서버를 멈추려면 stop 명령을 입력하면 된다.
- 서버를 백그라운드에서 계속 실행시키고 싶다면 Ctrl + p, Ctrl + q 를 입력하여 빠져나오자.
- 서버를 다시 실행시키려면 docker start <컨테이너 이름> 명령을 실행하면 된다.



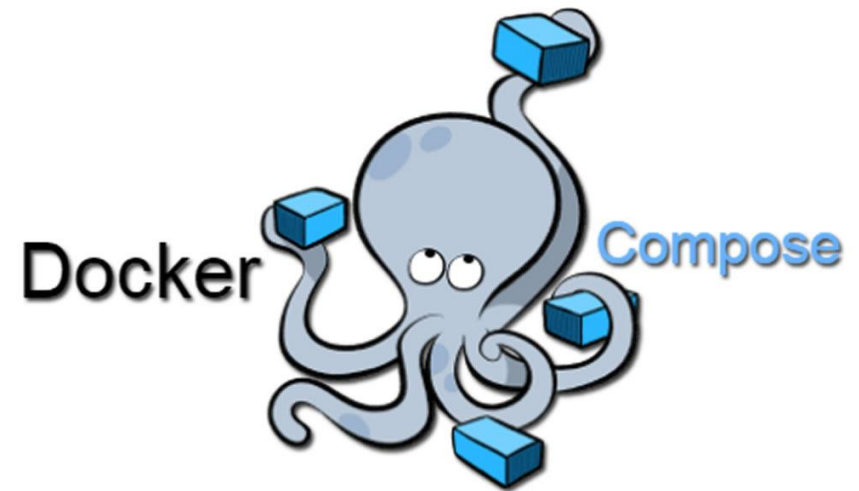
The screenshot shows a Windows PowerShell terminal window. The top part displays Minecraft server logs with timestamps and messages about player logins and disconnections. The word 'stop' is highlighted in red. Below the logs, the command 'docker ps -a' is entered, resulting in a table of Docker containers. The table has columns for CONTAINER ID, IMAGE, COMMAND, CREATED, STATUS, PORTS, and NAME. The first container, 'd10d70d6dcd0', is highlighted with a red box around its STATUS column, which shows 'Exited (0) 3 seconds ago'.

```
[16:58:43] [User Authenticator #1/INFO]: UUID of player solo5star is 2c637bfe-8e80-4e42-a801-b1c703770aef
[16:58:43] [Server thread/INFO]: solo5star[/172.17.0.1:41504] logged in with entity id 404 at (-6.5, 92.0, 4.5)
[16:58:43] [Server thread/INFO]: solo5star joined the game
[16:58:46] [Server thread/INFO]: solo5star lost connection: Disconnected
[16:58:46] [Server thread/INFO]: solo5star left the game
[16:59:51] [User Authenticator #2/INFO]: UUID of player solo5star is 2c637bfe-8e80-4e42-a801-b1c703770aef
[16:59:51] [Server thread/INFO]: solo5star[/172.17.0.1:41512] logged in with entity id 536 at (-6.5, 92.0, 4.5)
[16:59:51] [Server thread/INFO]: solo5star joined the game
stop
[17:01:58] [Server thread/INFO]: Stopping the server
[17:01:58] [Server thread/INFO]: Stopping server
[17:01:58] [Server thread/INFO]: Saving players
[17:01:58] [Server thread/INFO]: Saving worlds
[17:01:58] [Server thread/INFO]: Saving chunks for level 'ServerLevel[world]'/minecraft:overworld
[17:01:58] [Server thread/INFO]: Saving chunks for level 'ServerLevel[world]'/minecraft:the_nether
[17:01:58] [Server thread/INFO]: Saving chunks for level 'ServerLevel[world]'/minecraft:the_end
[17:01:58] [Server thread/INFO]: ThreadedAnvilChunkStorage (world): All chunks are saved
[17:01:58] [Server thread/INFO]: ThreadedAnvilChunkStorage (DIM-1): All chunks are saved
[17:01:58] [Server thread/INFO]: ThreadedAnvilChunkStorage (DIM1): All chunks are saved
[17:01:58] [Server thread/INFO]: ThreadedAnvilChunkStorage: All dimensions are saved
[17:01:58] [Server thread/INFO]: Thread RCON Listener stopped
2022-02-09T17:01:59.055Z INFO mc-server-runner Done
PS C:\Users\Home> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS              PORTS          NAME
d10d70d6dcd0   itzg/minecraft-server  "/start"            13 minutes ago   Exited (0) 3 seconds ago           sere
8cdd55a26295   mariadb       "docker-entrypoint.s..." 3 weeks ago     Exited (0) 3 weeks ago           whit
enote_mariadb_1
```

| CONTAINER ID    | IMAGE                 | COMMAND                  | CREATED        | STATUS                   | PORTS | NAME |
|-----------------|-----------------------|--------------------------|----------------|--------------------------|-------|------|
| d10d70d6dcd0    | itzg/minecraft-server | "/start"                 | 13 minutes ago | Exited (0) 3 seconds ago |       | sere |
| 8cdd55a26295    | mariadb               | "docker-entrypoint.s..." | 3 weeks ago    | Exited (0) 3 weeks ago   |       | whit |
| enote_mariadb_1 |                       |                          |                |                          |       |      |

# docker-compose란?

- docker-compose는 매번 긴 docker run 명령을 힘들게 쓰지 않게 해준다.
- docker-compose.yml 파일에 컨테이너들을 적어 놓으면, 이들을 한 번에 실행하거나 끌 수 있다.



# docker-compose란?

- docker run을 잘 사용할 수 있다면, docker-compose를 배우는 것은 어렵지 않다.
- docker run 명령은 옵션을 붙이다 보면 너무 길어지고 외우기가 힘들어진다.
- docker-compose는 docker run 명령을 파일에 적어놓는 것이다.
- 오른쪽 그림을 보자.

```
version: '3'
services:
  web:
    image: nginx
    ports:
      - "8081:80"
  mysql:
    image: mysql
    environment:
      MYSQL_ALLOW_EMPTY_PASSWORD: "yes"
```

# docker-compose란?

- 오른쪽은 아래의 두 명령을 적어 놓은 것과 같다.
  - `docker run -d -p 8081:80 nginx`
  - `docker run -e MYSQL_ALLOW_EMPTY_PASSWORD=yes mysql`

```
version: '3'
services:
  web:
    image: nginx
    ports:
      - "8081:80"
  mysql:
    image: mysql
    environment:
      MYSQL_ALLOW_EMPTY_PASSWORD: "yes"
```

# docker-compose.yml

- 핵심은 docker-compose.yml이다.
- yaml 문법에 맞춰 작성하여야 한다.
- 오른쪽은 mariadb를 docker-compose.yml에 작성한 예시이다.
- docker run과 docker-compose는 어떻게 다른지 오른쪽 주석을 참고하여 이해하도록 한다.

```
1  # docker-compose 버전
2  version: "3"
3  ▼ services:
4  ▼   mydb:
5       # 컨테이너의 이름 지정
6       container_name: mydb
7       # 컨테이너 이미지 지정
8       image: mariadb
9       # 컨테이너 재시작 정책
10      # unless-stopped는 사용자가 컨테이너를 직접 종료하기
11      # 전까지 컨테이너가 켜져있도록 해준다. PC가 재부팅되더라도
12      restart: unless-stopped
13  ▼   ports:
14       # 컨테이너에서 개방할 포트 설정
15       # 3306 포트를 통해 컨테이너 내부로 접근이 가능하다
16       - 3306:3306
17  ▼   volumes:
18       # 컨테이너 디렉토리 외부 연결 설정
19       # 컨테이너를 삭제해도 DB의 파일들이 ./db 폴더에
20       # 영구적으로 유지된다.
21       - ./db:/var/lib/mysql
22  ▼   environment:
23       # 환경 변수 설정
24       # MARIADB_DATABASE 환경변수는 DB가 실행될 때
25       # 기본적으로 생성할 데이터베이스의 이름이다
26       - MARIADB_DATABASE=mydatabase
27
```



# docker-compose.yml

- 최상단에는 version: "3" 을 적어준다.
- docker-compose 버전을 나타낸다.
- 항상 적어주어야 한다.

```
1 # docker-compose 버전
2 version: "3"
3 services:
4   mydb:
5     # 컨테이너의 이름 지정
6     container_name: mydb
7     # 컨테이너 이미지 지정
8     image: mariadb
9     # 컨테이너 재시작 정책
10    # unless-stopped는 사용자가 컨테이너를 직접 종료하기
11    # 전까지 컨테이너가 켜져있도록 해준다. pc가 재부팅되더라도.
12    restart: unless-stopped
13    ports:
14      # 컨테이너에서 개방할 포트 설정
15      # 3306 포트를 통해 컨테이너 내부로 접근이 가능하다
16      - 3306:3306
17    volumes:
18      # 컨테이너 디렉토리 외부 연결 설정
19      # 컨테이너를 삭제해도 DB의 파일들이 ./db 폴더에
20      # 영구적으로 유지된다.
21      - ./db:/var/lib/mysql
22    environment:
23      # 환경 변수 설정
24      # MARIADB_DATABASE 환경변수는 DB가 실행될 때
25      # 기본적으로 생성할 데이터베이스의 이름이다
26      - MARIADB_DATABASE=mydatabase
27
```

# docker-compose.yml

- services: 아래에는 컨테이너 목록을 적는다.
- 컨테이너는 한 개 또는 여러 개를 적을 수도 있다.

```
1  # docker-compose 버전
2  version: "3"
3  services:
4    mydb:
5      # 컨테이너의 이름 지정
6      container_name: mydb
7      # 컨테이너 이미지 지정
8      image: mariadb
9      # 컨테이너 재시작 정책
10     # unless-stopped는 사용자가 컨테이너를 직접 종료하기
11     # 전까지 컨테이너가 켜져있도록 해준다. pc가 재부팅되더라도.
12     restart: unless-stopped
13     ports:
14       # 컨테이너에서 개방할 포트 설정
15       # 3306 포트를 통해 컨테이너 내부로 접근이 가능하다
16       - 3306:3306
17     volumes:
18       # 컨테이너 디렉토리 외부 연결 설정
19       # 컨테이너를 삭제해도 DB의 파일들이 ./db 폴더에
20       # 영구적으로 유지된다.
21       - ./db:/var/lib/mysql
22     environment:
23       # 환경 변수 설정
24       # MARIADB_DATABASE 환경변수는 DB가 실행될 때
25       # 기본적으로 생성할 데이터베이스의 이름이다
26       - MARIADB_DATABASE=mydatabase
27
```

# docker-compose.yml

- 서비스의 이름을 적는다.
- 컨테이너 이름과 다른 개념이며, docker-compose.yml 내에서 구분하기 위한 용도이다.
  - 예) web, db, frontend, backend

```
1 # docker-compose 버전
2 version: "3"
3 services:
4   mydb:
5     # 컨테이너의 이름 지정
6     container_name: mydb
7     # 컨테이너 이미지 지정
8     image: mariadb
9     # 컨테이너 재시작 정책
10    # unless-stopped는 사용자가 컨테이너를 직접 종료하기
11    # 전까지 컨테이너가 켜져있도록 해준다. pc가 재부팅되더라도.
12    restart: unless-stopped
13    ports:
14      # 컨테이너에서 개방할 포트 설정
15      # 3306 포트를 통해 컨테이너 내부로 접근이 가능하다
16      - 3306:3306
17    volumes:
18      # 컨테이너 디렉토리 외부 연결 설정
19      # 컨테이너를 삭제해도 DB의 파일들이 ./db 폴더에
20      # 영구적으로 유지된다.
21      - ./db:/var/lib/mysql
22    environment:
23      # 환경 변수 설정
24      # MARIADB_DATABASE 환경변수는 DB가 실행될 때
25      # 기본적으로 생성할 데이터베이스의 이름이다
26      - MARIADB_DATABASE=mydatabase
27
```

# docker-compose.yml

- container\_name은 컨테이너 이름을 지정한다.
- docker run에서 --name 옵션으로 사용했던 부분이다.

```
1  # docker-compose 버전
2  version: "3"
3  ▼ services:
4  ▼  mydb:
5      # 컨테이너의 이름 지정
6      container_name: mydb
7      # 컨테이너 이미지 지정
8      image: mariadb
9      # 컨테이너 재시작 정책
10     # unless-stopped는 사용자가 컨테이너를 직접 종료하기
11     # 전까지 컨테이너가 켜져있도록 해준다. PC가 재부팅되더라도.
12     restart: unless-stopped
13  ▼  ports:
14      # 컨테이너에서 개방할 포트 설정
15      # 3306 포트를 통해 컨테이너 내부로 접근이 가능하다
16      - 3306:3306
17  ▼  volumes:
18      # 컨테이너 디렉토리 외부 연결 설정
19      # 컨테이너를 삭제해도 DB의 파일들이 ./db 폴더에
20      # 영구적으로 유지된다.
21      - ./db:/var/lib/mysql
22  ▼  environment:
23      # 환경 변수 설정
24      # MARIADB_DATABASE 환경변수는 DB가 실행될 때
25      # 기본적으로 생성할 데이터베이스의 이름이다
26      - MARIADB_DATABASE=mydatabase
27
```

# docker-compose.yml

- image는 적용할 이미지를 지정한다.
- hub.docker.com 에서 받아온다.
- mariadb, mysql, postgresql, redis 등 다양한 이미지를 사용할 수 있다.

```
1  # docker-compose 버전
2  version: "3"
3  services:
4    mydb:
5      # 컨테이너의 이름 지정
6      container_name: mydb
7      # 컨테이너 이미지 지정
8      image: mariadb
9      # 컨테이너 재시작 정책
10     # unless-stopped는 사용자가 컨테이너를 직접 종료하기
11     # 전까지 컨테이너가 켜져있도록 해준다. pc가 재부팅되더라도.
12     restart: unless-stopped
13     ports:
14       # 컨테이너에서 개방할 포트 설정
15       # 3306 포트를 통해 컨테이너 내부로 접근이 가능하다
16       - 3306:3306
17     volumes:
18       # 컨테이너 디렉토리 외부 연결 설정
19       # 컨테이너를 삭제해도 DB의 파일들이 ./db 폴더에
20       # 영구적으로 유지된다.
21       - ./db:/var/lib/mysql
22     environment:
23       # 환경 변수 설정
24       # MARIADB_DATABASE 환경변수는 DB가 실행될 때
25       # 기본적으로 생성할 데이터베이스의 이름이다
26       - MARIADB_DATABASE=mydatabase
27
```

# docker-compose.yml

- restart 옵션은 컨테이너의 재시작 정책을 설정한다.
- unless-stopped 값은 PC가 재부팅 되었을 때 컨테이너도 자동으로 시작되도록 해준다.

```
1  # docker-compose 버전
2  version: "3"
3  services:
4    mydb:
5      # 컨테이너의 이름 지정
6      container_name: mydb
7      # 컨테이너 이미지 지정
8      image: mariadb
9      # 컨테이너 재시작 정책
10     # unless-stopped는 사용자가 컨테이너를 직접 종료하기
11     # 전까지 컨테이너가 켜져있도록 해준다. PC가 재부팅되더라도.
12     restart: unless-stopped
13     ports:
14       # 컨테이너에서 개방할 포트 설정
15       # 3306 포트를 통해 컨테이너 내부로 접근이 가능하다
16       - 3306:3306
17     volumes:
18       # 컨테이너 디렉토리 외부 연결 설정
19       # 컨테이너를 삭제해도 DB의 파일들이 ./db 폴더에
20       # 영구적으로 유지된다.
21       - ./db:/var/lib/mysql
22     environment:
23       # 환경 변수 설정
24       # MARIADB_DATABASE 환경변수는 DB가 실행될 때
25       # 기본적으로 생성할 데이터베이스의 이름이다
26       - MARIADB_DATABASE=mydatabase
27
```

# docker-compose.yml

- ports: 는 컨테이너 내부의 포트를 외부에 개방할 때 사용한다.
- mariadb는 3306 포트를 통해 DB에 접속할 수 있도록 해준다.
- ports 옵션이 없다면, 컨테이너 내부의 3306 포트로 접근할 수 없게 된다.

```
1  # docker-compose 버전
2  version: "3"
3  services:
4    mydb:
5      # 컨테이너의 이름 지정
6      container_name: mydb
7      # 컨테이너 이미지 지정
8      image: mariadb
9      # 컨테이너 재시작 정책
10     # unless-stopped는 사용자가 컨테이너를 직접 종료하기
11     # 전까지 컨테이너가 켜져있도록 해준다. pc가 재부팅되더라도.
12     restart: unless-stopped
13     ports:
14       # 컨테이너에서 개방할 포트 설정
15       # 3306 포트를 통해 컨테이너 내부로 접근이 가능하다
16       - 3306:3306
17     volumes:
18       # 컨테이너 디렉토리 외부 연결 설정
19       # 컨테이너를 삭제해도 DB의 파일들이 ./db 폴더에
20       # 영구적으로 유지된다.
21       - ./db:/var/lib/mysql
22     environment:
23       # 환경 변수 설정
24       # MARIADB_DATABASE 환경변수는 DB가 실행될 때
25       # 기본적으로 생성할 데이터베이스의 이름이다
26       - MARIADB_DATABASE=mydatabase
27
```

# docker-compose.yml

- volumes: 는 컨테이너 내부의 디렉토리를 외부로 연결해주는 옵션이다.
- mariadb는 /var/lib/mysql 디렉토리에 DB 데이터가 저장된다.
- 오른쪽의 예시에서는 현재 폴더의 db로 연결해두었으며, ./db에 DB 데이터가 저장된다.
- 컨테이너가 삭제되면 컨테이너 내부의 파일들은 모두 사라진다.
- 따라서 volumes 옵션으로 컨테이너 바깥에 저장할 필요가 있다.

```
1  # docker-compose 버전
2  version: "3"
3  services:
4    mydb:
5      # 컨테이너의 이름 지정
6      container_name: mydb
7      # 컨테이너 이미지 지정
8      image: mariadb
9      # 컨테이너 재시작 정책
10     # unless-stopped는 사용자가 컨테이너를 직접 종료하기
11     # 전까지 컨테이너가 켜져있도록 해준다. PC가 재부팅되더라도.
12     restart: unless-stopped
13     ports:
14       # 컨테이너에서 개방할 포트 설정
15       # 3306 포트를 통해 컨테이너 내부로 접근이 가능하다
16       - 3306:3306
17     volumes:
18       # 컨테이너 디렉토리 외부 연결 설정
19       # 컨테이너를 삭제해도 DB의 파일들이 ./db 폴더에
20       # 영구적으로 유지된다.
21       - ./db:/var/lib/mysql
22     environment:
23       # 환경 변수 설정
24       # MARIADB_DATABASE 환경변수는 DB가 실행될 때
25       # 기본적으로 생성할 데이터베이스의 이름이다
26       - MARIADB_DATABASE=mydatabase
27
```



# docker-compose.yml

- environment: 는 환경변수를 설정하는 옵션이다.
- 컨테이너 내 설정들은 대부분 환경변수를 통해 설정한다.
- 오른쪽 예시에서는 MARIADB\_DATABASE 환경변수를 사용하여 최초에 생성할 데이터베이스 이름을 지정해주었다.

```
1  # docker-compose 버전
2  version: "3"
3  ▼ services:
4  ▼  mydb:
5      # 컨테이너의 이름 지정
6      container_name: mydb
7      # 컨테이너 이미지 지정
8      image: mariadb
9      # 컨테이너 재시작 정책
10     # unless-stopped는 사용자가 컨테이너를 직접 종료하기
11     # 전까지 컨테이너가 켜져있도록 해준다. PC가 재부팅되더라도.
12     restart: unless-stopped
13  ▼  ports:
14      # 컨테이너에서 개방할 포트 설정
15      # 3306 포트를 통해 컨테이너 내부로 접근이 가능하다
16      - 3306:3306
17  ▼  volumes:
18      # 컨테이너 디렉토리 외부 연결 설정
19      # 컨테이너를 삭제해도 DB의 파일들이 ./db 폴더에
20      # 영구적으로 유지된다.
21      - ./db:/var/lib/mysql
22  ▼  environment:
23      # 환경 변수 설정
24      # MARIADB_DATABASE 환경변수는 DB가 실행될 때
25      # 기본적으로 생성할 데이터베이스의 이름이다
26      - MARIADB_DATABASE=mydatabase
27
```

# docker-compose 명령

- docker-compose.yml에 작성한 컨테이너들은 docker-compose 명령으로 실행할 수 있다.

- **docker-compose up**

- docker-compose.yml 컨테이너들을 실행한다.

- **docker-compose up -d**

- 컨테이너들을 백그라운드로 실행한다.

- **docker-compose down**

- 컨테이너들을 끄고 삭제한다.

```
PS C:\Users\Home\Desktop\workspace\uncategorized\docker-deu\minecraft> docker-compose up -d
[+] Running 2/2
 - Network minecraft_default Created                                0.0s
 - Container mydb Started                                          0.6s
PS C:\Users\Home\Desktop\workspace\uncategorized\docker-deu\minecraft> docker-compose down
[+] Running 2/2
 - Container mydb Removed                                          0.2s
 - Network minecraft_default Removed                               0.2s
PS C:\Users\Home\Desktop\workspace\uncategorized\docker-deu\minecraft> docker-compose up
[+] Running 2/0
 - Network minecraft_default Created                                0.0s
 - Container mydb Created                                          0.0s
Attaching to mydb
mydb | 2022-02-09 17:38:46+00:00 [Note] [Entrypoint]: Entrypoint script for MariaDB Server 1:10.6.4+ma
ria~focal started.
mydb | 2022-02-09 17:38:46+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
mydb | 2022-02-09 17:38:46+00:00 [Note] [Entrypoint]: Entrypoint script for MariaDB Server 1:10.6.4+ma
ria~focal started.
mydb | 2022-02-09 17:38:46+00:00 [ERROR] [Entrypoint]: Database is uninitialized and password option i
s not specified
mydb | You need to specify one of MARIADB_ROOT_PASSWORD, MARIADB_ALLOW_EMPTY_ROOT_PASSWORD and
MARIADB_RANDOM_ROOT_PASSWORD
```

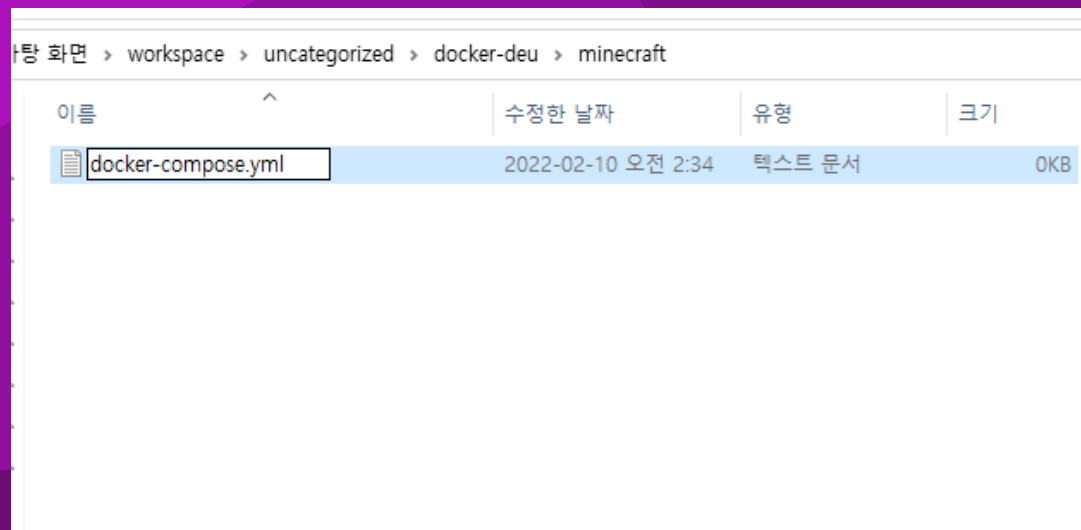
# docker-compose 명령

- docker-compose up <서비스 이름>
  - docker-compose.yml 내 특정 서비스만 실행한다.
- docker-compose stop
  - 컨테이너들을 중지한다.
- docker-compose start
  - 컨테이너들을 다시 켜다.

```
PS C:\Users\Home\Desktop\workspace\uncategorized\docker-deu\minecraft> docker-compose up -d
[+] Running 2/2
- Network minecraft_default Created                                0.0s
- Container mydb Started                                          0.6s
PS C:\Users\Home\Desktop\workspace\uncategorized\docker-deu\minecraft> docker-compose down
[+] Running 2/2
- Container mydb Removed                                          0.2s
- Network minecraft_default Removed                               0.2s
PS C:\Users\Home\Desktop\workspace\uncategorized\docker-deu\minecraft> docker-compose up
[+] Running 2/0
- Network minecraft_default Created                                0.0s
- Container mydb Created                                          0.0s
Attaching to mydb
mydb | 2022-02-09 17:38:46+00:00 [Note] [Entrypoint]: Entrypoint script for MariaDB Server 1:10.6.4+ma
ria~focal started.
mydb | 2022-02-09 17:38:46+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
mydb | 2022-02-09 17:38:46+00:00 [Note] [Entrypoint]: Entrypoint script for MariaDB Server 1:10.6.4+ma
ria~focal started.
mydb | 2022-02-09 17:38:46+00:00 [ERROR] [Entrypoint]: Database is uninitialized and password option i
s not specified
mydb | You need to specify one of MARIADB_ROOT_PASSWORD, MARIADB_ALLOW_EMPTY_ROOT_PASSWORD and
MARIADB_RANDOM_ROOT_PASSWORD
```

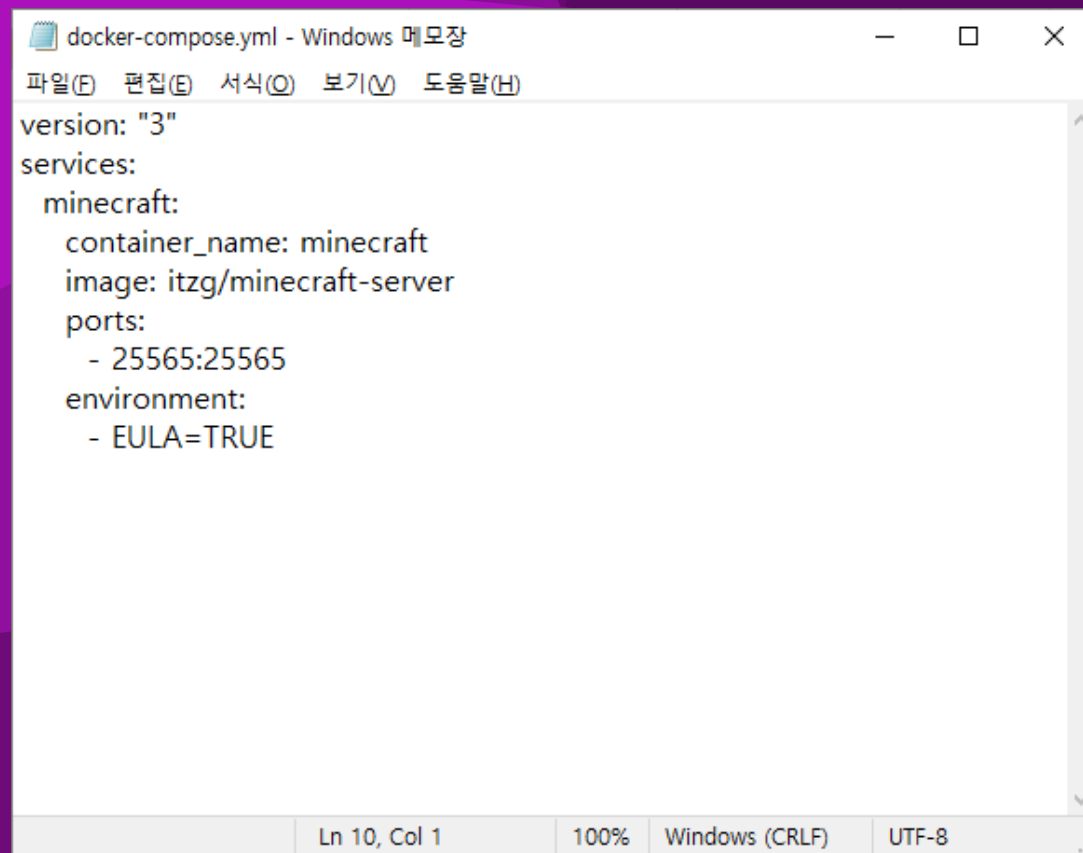
# [실습] docker-compose.yml 작성해보기

- 마인크래프트 구동기를 docker-compose.yml로 작성해보자.
- 적당한 폴더를 만들고 docker-compose.yml 파일을 생성한다.



# [실습] docker-compose.yml 작성해보기

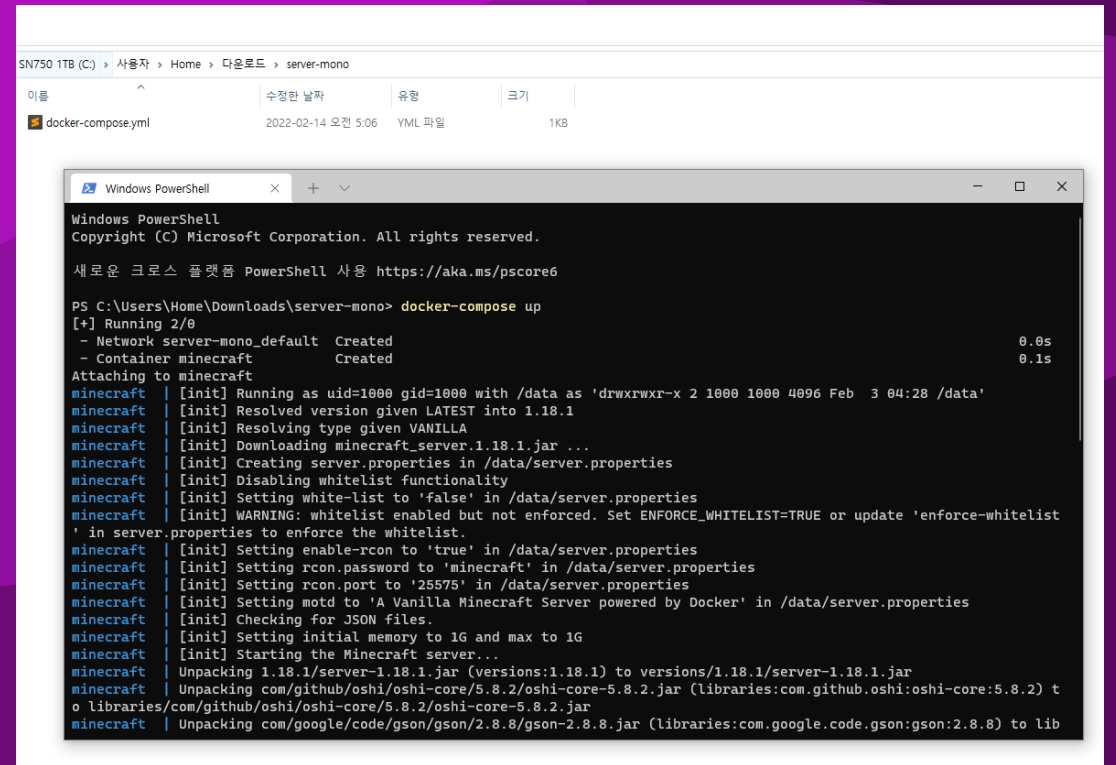
- 오른쪽과 같이 docker-compose.yml을 작성해보자.



```
docker-compose.yml - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
version: "3"
services:
  minecraft:
    container_name: minecraft
    image: itzg/minecraft-server
    ports:
      - 25565:25565
    environment:
      - EULA=TRUE
Ln 10, Col 1 100% Windows (CRLF) UTF-8
```

# [실습] docker-compose.yml 작성해보기

- 디렉토리에서 Windows Terminal을 열고, docker-compose up 명령을 실행한다.



The screenshot shows a Windows File Explorer window with the path 'C:\Users\Home\Downloads\server-mono'. It contains a file named 'docker-compose.yml' (YML 파일, 1KB, modified 2022-02-14 05:06). Below the file explorer is a Windows PowerShell terminal window. The terminal output shows the execution of 'docker-compose up', which creates a network 'server-mono\_default' and a container 'minecraft'. It then shows the initialization of the Minecraft server, including downloading the 'minecraft\_server.1.18.1.jar' file and setting various properties like 'enable-rcon' and 'rcon.password'.

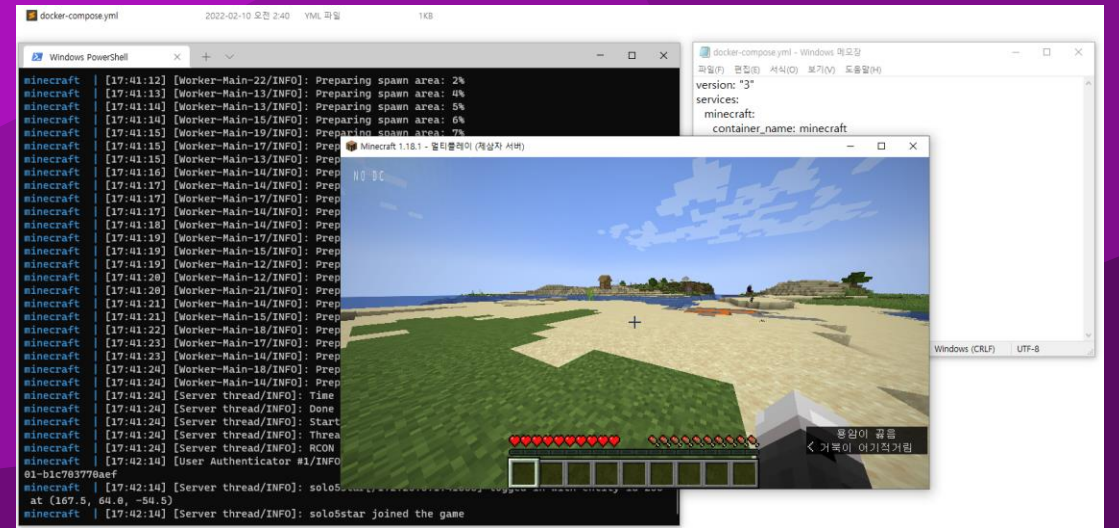
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 크로스 플랫폼 PowerShell 사용 https://aka.ms/pscore6

PS C:\Users\Home\Downloads\server-mono> docker-compose up
[+] Running 2/0
 - Network server-mono_default Created                                0.0s
 - Container minecraft Created                                       0.1s
Attaching to minecraft
minecraft | [init] Running as uid=1000 gid=1000 with /data as 'drwxrwxr-x 2 1000 1000 4096 Feb  3 04:28 /data'
minecraft | [init] Resolved version given LATEST into 1.18.1
minecraft | [init] Resolving type given VANILLA
minecraft | [init] Downloading minecraft_server.1.18.1.jar ...
minecraft | [init] Creating server.properties in /data/server.properties
minecraft | [init] Disabling whitelist functionality
minecraft | [init] Setting white-list to 'false' in /data/server.properties
minecraft | [init] WARNING: whitelist enabled but not enforced. Set ENFORCE_WHITELIST=TRUE or update 'enforce-whitelist' in server.properties to enforce the whitelist.
minecraft | [init] Setting enable-rcon to 'true' in /data/server.properties
minecraft | [init] Setting rcon.password to 'minecraft' in /data/server.properties
minecraft | [init] Setting rcon.port to '25575' in /data/server.properties
minecraft | [init] Setting motd to 'A Vanilla Minecraft Server powered by Docker' in /data/server.properties
minecraft | [init] Checking for JSON files.
minecraft | [init] Setting initial memory to 1G and max to 1G
minecraft | [init] Starting the Minecraft server...
minecraft | Unpacking 1.18.1/server-1.18.1.jar (versions:1.18.1) to versions/1.18.1/server-1.18.1.jar
minecraft | Unpacking com/github/oshi/oshi-core/5.8.2/oshi-core-5.8.2.jar (Libraries:com.github.oshi:oshi-core:5.8.2) to libraries/com/github/oshi/oshi-core/5.8.2/oshi-core-5.8.2.jar
minecraft | Unpacking com/google/code/gson/gson/2.8.8/gson-2.8.8.jar (Libraries:com.google.code.gson:gson:2.8.8) to lib
```

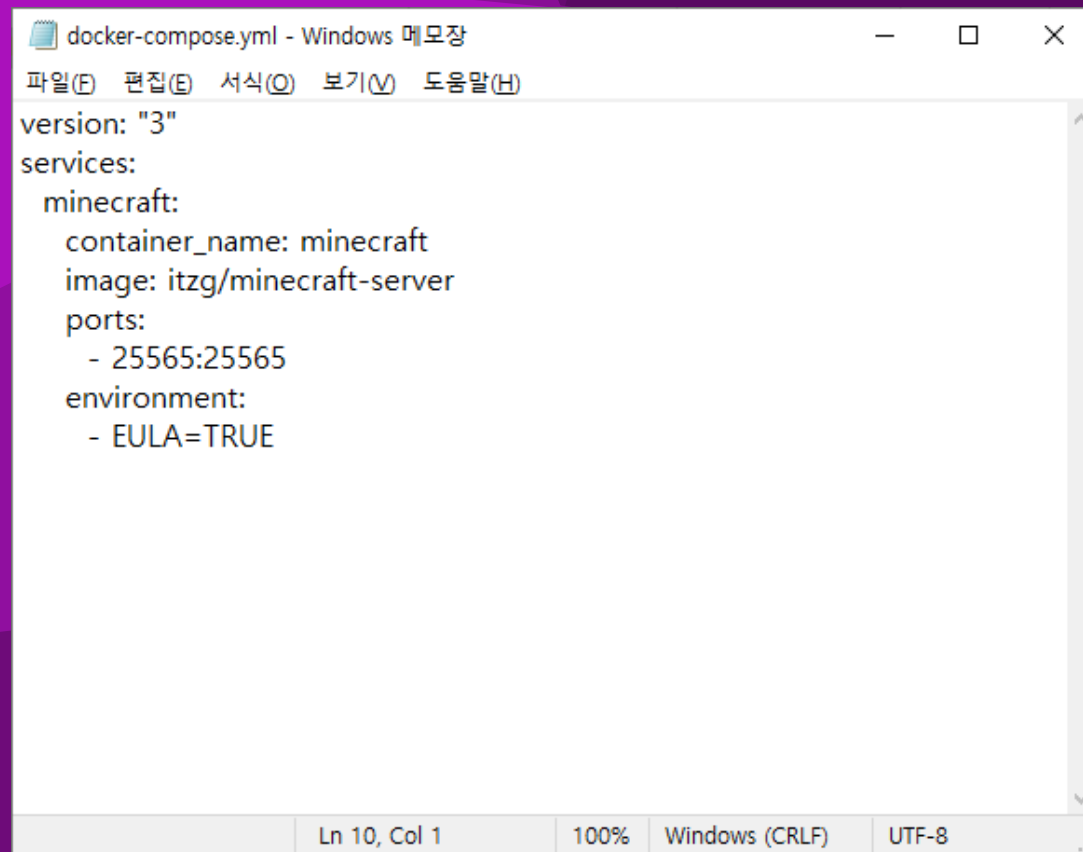
# [실습] docker-compose.yml 작성해보기

- 마인크래프트를 켜고 localhost로 접속해보자.



# [실습] 제대로 구동기 실행해보기

- 오른쪽과 같이 docker-compose.yml을 작성해주면 별다른 과정 없이 바로 구동기를 실행할 수 있다.
- 그러나 컨테이너를 삭제하면 월드, 플레이어 데이터 등 중요한 데이터가 같이 사라진다.
- volumes 설정, 환경변수 설정 등을 추가하여 제대로 구동기를 실행해보도록 한다.



```
docker-compose.yml - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
version: "3"
services:
  minecraft:
    container_name: minecraft
    image: itzg/minecraft-server
    ports:
      - 25565:25565
    environment:
      - EULA=TRUE
```

Ln 10, Col 1 100% Windows (CRLF) UTF-8



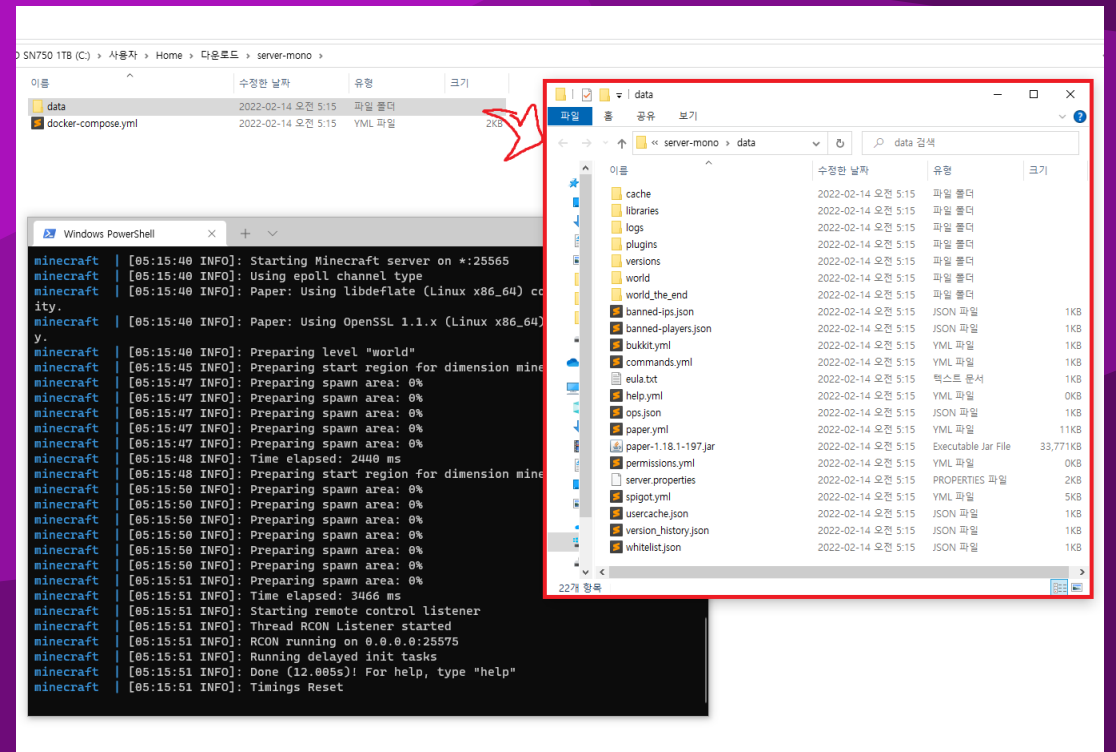
# [실습] 제대로 구동기 실행해보기

- docker-compose.yml 파일을 열고 오른쪽과 같이 수정한다.
- 콘솔 사용이 가능하도록 stdin\_open, tty 옵션이 추가되었다.
- PC 재부팅 시 컨테이너가 자동으로 시작되도록 restart 옵션이 추가되었다.
- 세밀한 설정을 위한 환경변수들이 추가되었다.

```
1 version: "3"
2▼ services:
3▼   minecraft:
4     container_name: minecraft
5     image: itzg/minecraft-server
6     ...stdin_open: true
7     ...tty: true
8     ...restart: unless-stopped
9     ports:
10      - 25565:25565
11     ...volumes:
12      - ./data:/data
13▼   environment:
14     - EULA=true # EULA 동의
15     ...TYPE=PAPER # 구동기 종류를 Paper로 설정
16     ...VERSION=1.18.1 # 구동기를 다운받을 때 1.18.1로 다운
17     ...TZ=Asia/Seoul # 타임존을 서울로 설정
18     ...MEMORY=1G # 메모리를 1GB만큼 할당
19     ...OVERRIDE_SERVER_PROPERTIES=true # 환경변수가 server.properties를 덮어쓰도록 함
20     ...MAX_PLAYERS=50 # 접속 가능한 플레이어 수를 50으로 설정
21     ...MODE=creative # 기본 게임 모드를 크리에이티브로 설정
22     ...MOTD=Welcome Server # MOTD를 Welcom Server로 설정
23     ...ALLOW_NETHER=FALSE # 네더 월드를 사용하지 않음
24     ...VIEW_DISTANCE=50 # 플레이어 최대 시야를 50으로 설정
25     ...PVP=FALSE # PVP 비허용
```

# [실습] 제대로 구동기 실행해보기

- docker-compose up 으로 서버를 실행하고 오른 쪽과 같이 폴더가 생성되는지 확인한다.



# [실습] 제대로 구동기 실행해보기

- 마인크래프트를 켜고 서버에 접속하여 환경변수 설정이 잘 되었는지 확인한다.
  - 최대인원: 50명
  - 게임모드: 크리에이티브
  - MOTD: Welcome Server
  - 네더 월드: 비활성화
  - View Distance: 50
  - PVP: 비활성화

