

제 목	03장 Docker 볼륨 활용	
상세내용	도커 볼륨을 활용해 데이터 유실 방지하기	

1. Docker Volume(도커 볼륨)

✓ 컨테이너가 가진 문제점

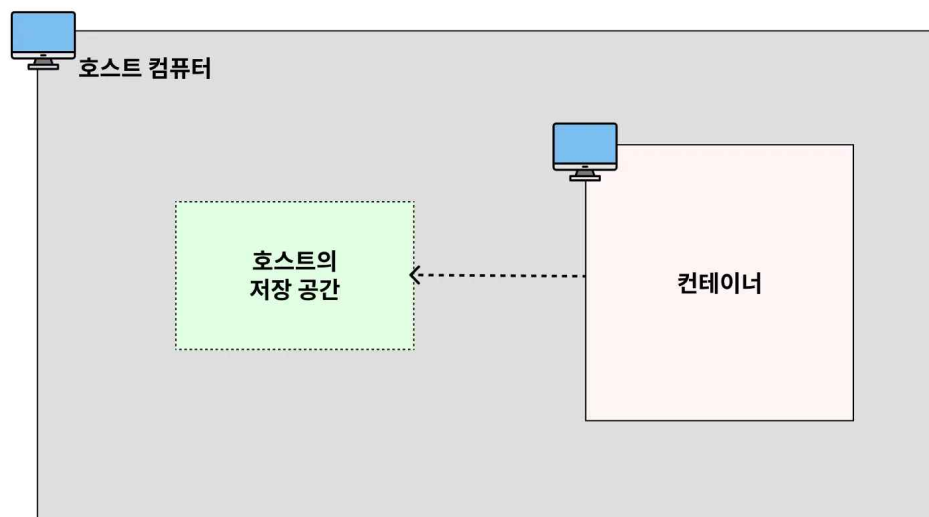
Docker를 활용하면 특정 프로그램을 컨테이너로 띄울 수 있다. 이 프로그램에 기능이 추가되면 새로운 이미지를 만들어서 컨테이너를 실행시켜야 한다. 이 때, Docker는 기존 컨테이너에서 변경된 부분을 수정하지 않고, 새로운 컨테이너를 만들어서 통째로 갈아끼우는 방식으로 교체를 한다. 이게 효율적이라고 생각했던 것이다.

이런 특징 때문에 기존 컨테이너를 새로운 컨테이너로 교체하면, 기존 컨테이너 내부에 있던 데이터도 같이 삭제된다. 만약 이 컨테이너가 MySQL을 실행시키는 컨테이너였다면 MySQL에 저장된 데이터도 같이 삭제 돼버린다.

따라서 컨테이너 내부에 저장된 데이터가 삭제되면 안 되는 경우에는 **볼륨 (Volume)**이라는 개념을 활용해야 한다.

✓ Docker Volume(도커 볼륨)이란?

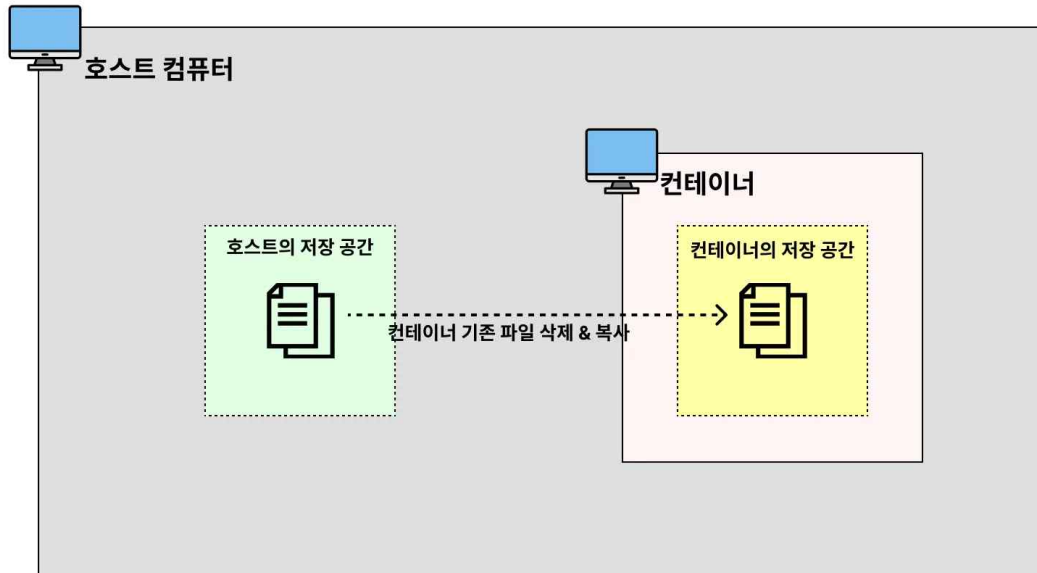
도커의 볼륨(Volume)이란 도커 컨테이너에서 데이터를 영속적으로 저장하기 위한 방법이다. 볼륨(Volume)은 컨테이너 자체의 저장 공간을 사용하지 않고, 호스트 자체의 저장 공간을 공유해서 사용하는 형태이다.



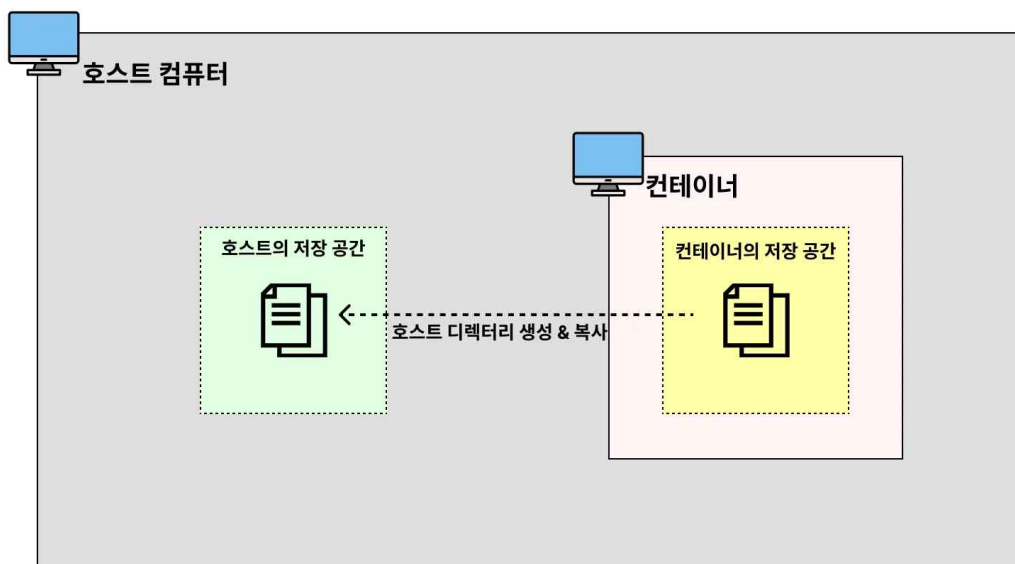
✓ 볼륨(Volume)을 사용하는 명령어

```
$ docker run -v [호스트의 디렉토리 절대경로]:[컨테이너의 디렉토리 절대경로] [이미지명]:[태그명]
```

- [호스트의 디렉토리 절대 경로]에 디렉토리가 이미 존재할 경우, 호스트의 디렉터리가 컨테이너의 디렉터리를 덮어씀운다.



- [호스트의 디렉토리 절대 경로]에 디렉토리가 존재하지 않을 경우, 호스트의 디렉터리 절대 경로에 디렉터리를 새로 만들고 컨테이너의 디렉터리에 있는 파일들을 호스트의 디렉터리로 복사해온다.



[실습] Docker로 MySQL 실행시켜보기 - 1

✓ Docker로 MySQL 실행시켜보기

1. MySQL 이미지를 바탕으로 컨테이너 실행시키기

- https://hub.docker.com/_/mysql

```
$ docker run -e MYSQL_ROOT_PASSWORD=pass1234 -p 3306:3306 -d mysql
```

- 참고) `docker pull` 과정은 생략해도 상관없다. 왜냐하면 `docker run mysql`로 실행시켰을 때, 로컬에 이미지가 없으면 Dockerhub으로부터 MySQL 이미지를 알아서 다운받아서 실행시키기 때문이다.
- `-e MYSQL_ROOT_PASSWORD=password123` : `-e` 옵션은 컨테이너의 환경 변수를 설정하는 옵션이다.
- Dockerhub의 MySQL 공식 문서를 보면 환경 변수로 `'MYSQL_ROOT_PASSWORD'`를 정해주어야만 정상적으로 컨테이너가 실행된다고 적혀져있다.
- 아래의 명령어로 컨테이너로 들어가서 환경 변수를 직접 눈으로 확인해보자.

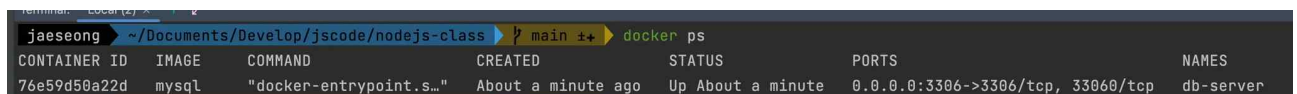
```
$ docker exec -it [MySQL 컨테이너 ID] bash
```

```
$ echo $MYSQL_ROOT_PASSWORD # MYSQL_ROOT_PASSWORD라는 환경변수 값 출력
```

```
$ export # 설정되어 있는 모든 환경변수 출력
```

2. 컨테이너가 잘 실행되고 있는 지 체크

```
$ docker ps
```

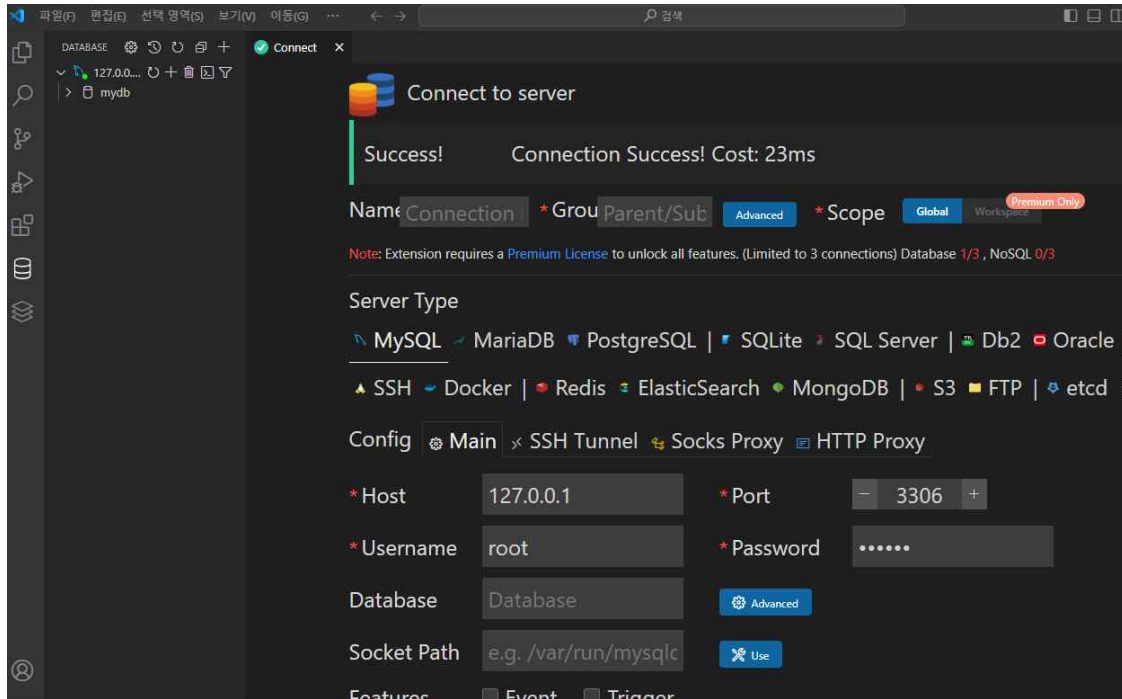


CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
76e59d50a22d	mysql	"docker-entrypoint.s..."	About a minute ago	Up About a minute	0.0.0.0:3306->3306/tcp, 33060/tcp	db-server

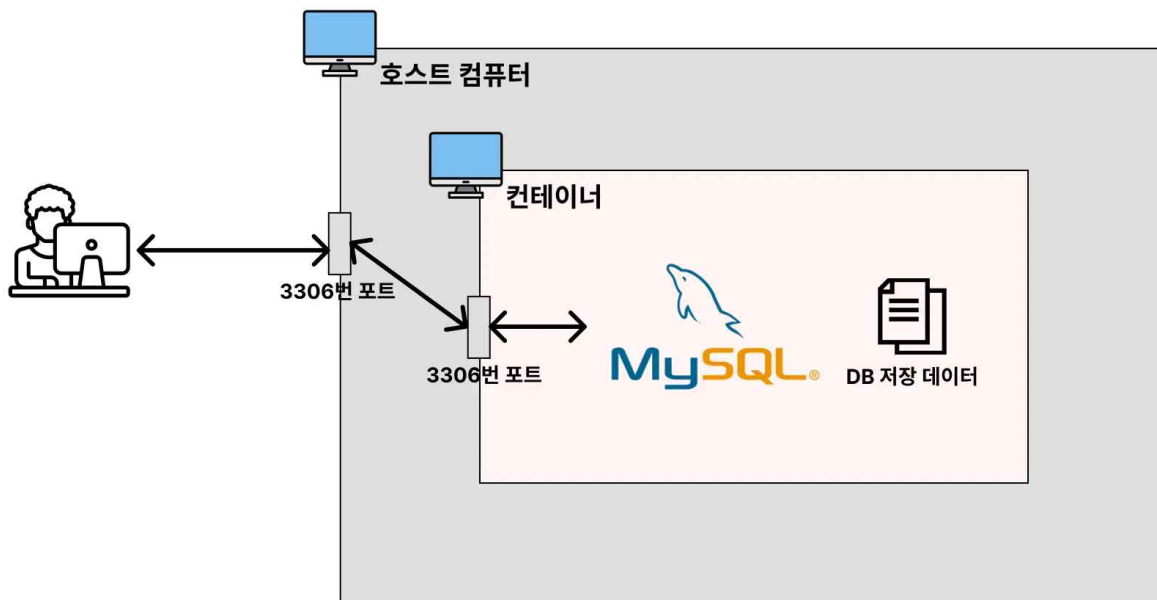
3. 컨테이너 실행시킬 때 에러 없이 잘 실행됐는 지 로그 체크

```
$ docker logs [컨테이너 ID 또는 컨테이너명]
```

4. vscode 에서도 연결시켜보기



✓ 그림으로 이해하기



[실습] Docker로 MySQL 실행시켜보기 - 2

✓ MySQL 컨테이너에 직접 접속해보기

1. MySQL 컨테이너에 접속

```
$ docker exec -it [MySQL 컨테이너 ID] bash
```

```
jaeseong ~/Documents/Develop/jscode/nodejs-class main ++ docker exec -it 76e sh
sh-4.4#
```

2. 컨테이너에서 MySQL에 접근하기

```
$ mysql -u root -p
```

```
sh-4.4# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14
Server version: 8.0.33 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

3. MySQL 접근에 성공했다면 데이터베이스 조회해보기

```
mysql> show databases;
```

```
mysql> show databases;
+-----+
| Database                |
+-----+
| information_schema      |
| mysql                   |
| performance_schema      |
| sys                     |
+-----+
4 rows in set (0.03 sec)
```

4. 데이터베이스 만들기

```
mysql> create database mydb;
mysql> show databases;
```

5. 컨테이너 종료 후 다시 생성해보기

```
# 컨테이너 종료
$ docker stop [MySQL 컨테이너 ID]
$ docker rm [MySQL 컨테이너 ID]

# 컨테이너 생성
$ docker run -e MYSQL_ROOT_PASSWORD=password123 -p 3306:3306 -d mysql
$ docker exec -it [MySQL 컨테이너 ID] bash

$ mysql -u root -p
mysql> show databases; # 아까 생성한 데이터베이스가 없어진 걸 확인할 수 있다.
```

- 위 방식은 볼륨(Volume)을 활용하지 않고 MySQL 컨테이너를 띄웠다. 그래서 MySQL 컨테이너를 삭제함과 동시에 MySQL 내부에 저장되어 있던 데이터도 함께 삭제되어 없어졌다. 이를 방지하기 위해 볼륨(Volume)을 활용해 MySQL 컨테이너를 띄우는 방식에 대해 알아볼 것이다.

[실습] Docker로 MySQL 실행시켜보기 - 3

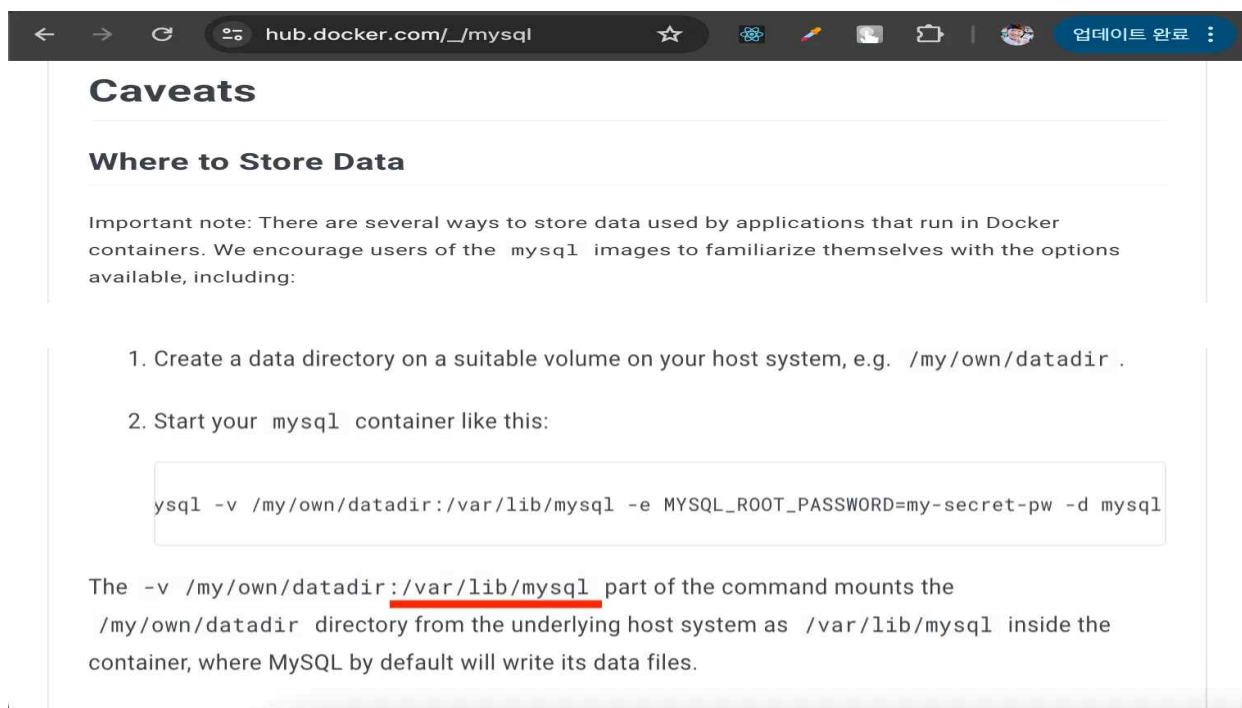
✓ 볼륨(Volume)을 활용해 MySQL 컨테이너 띄우기

1. MySQL 컨테이너 띄우기

```
$ cd c:/
$ mkdir docker_data # MySQL 데이터를 저장하고 싶은 폴더 만들기

# docker run -e MYSQL_ROOT_PASSWORD=password123 -p 3306:3306 -v {호스트
의 절대경로}/mysql_data:/var/lib/mysql -d mysql
$ docker run -e MYSQL_ROOT_PASSWORD=password123 -p 3306:3306 -v
c:/docker_data/mysql_data:/var/lib/mysql -d mysql
```

- `pwd` 명령어로 볼륨으로 사용하고자 하는 경로를 확인한 뒤 입력해주자.
- **주의)** `mysql_data` 디렉토리를 미리 만들어 놓으면 안 된다. 그래야 처음 이미지를 실행시킬 때 mysql 내부에 있는 `/var/lib/mysql` 파일들을 호스트 컴퓨터로 공유받을 수 있다. `mysql_data` 디렉토리를 미리 만들어놓을 경우, 기존 컨테이너의 `/var/lib/mysql` 파일들을 전부 삭제한 뒤에 `mysql_data`로 덮어쓰워 버린다.
- DB에 관련된 데이터가 저장되는 곳이 `/var/lib/mysql`인지는 Dockerhub MySQL의 공식 문서에 나와있다.



The screenshot shows the Docker Hub page for the MySQL image. The 'Caveats' section is expanded, showing the 'Where to Store Data' subsection. It includes an important note about data storage and two numbered steps for setting up a data directory and running the container. A code block shows the command to run the MySQL container with a data volume.

Caveats

Where to Store Data

Important note: There are several ways to store data used by applications that run in Docker containers. We encourage users of the `mysql` images to familiarize themselves with the options available, including:

1. Create a data directory on a suitable volume on your host system, e.g. `/my/own/datadir`.
2. Start your `mysql` container like this:

```
mysql -v /my/own/datadir:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mysql
```

The `-v /my/own/datadir:/var/lib/mysql` part of the command mounts the `/my/own/datadir` directory from the underlying host system as `/var/lib/mysql` inside the container, where MySQL by default will write its data files.

2. MySQL 컨테이너에 접속해서 데이터베이스 만들기

```
$ docker exec -it [MySQL 컨테이너 ID] bash

$ mysql -u root -p

mysql> show databases;
mysql> create database mydb;
mysql> show databases;
```

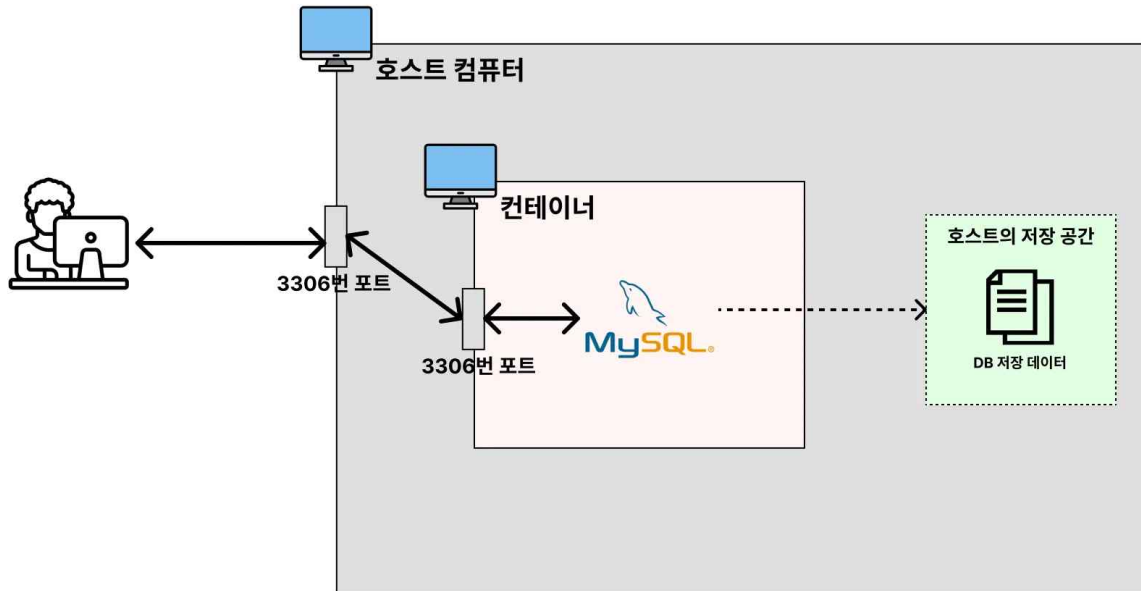
3. 컨테이너 종료 후 다시 생성해보기

```
# 컨테이너 종료
$ docker stop [MySQL 컨테이너 ID]
$ docker rm [MySQL 컨테이너 ID]

# 컨테이너 생성
$ docker run -e MYSQL_ROOT_PASSWORD=password123 -p 3306:3306 -v
c:/docker_data/mysql_data:/var/lib/mysql -d mysql

$ docker exec -it [MySQL 컨테이너 ID] bash
$ mysql -u root -p
mysql> show databases; # 아까 생성한 데이터베이스가 그대로 존재하는 걸 확인할 수
있다.
```


✓ 그림으로 이해하기



✓ MySQL 컨테이너 삭제하고 다시 띄워보기

```
# 컨테이너 종료
$ docker stop [MySQL 컨테이너 ID]
$ docker rm [MySQL 컨테이너 ID]

# 비밀번호 바꿔서 컨테이너 생성
$ docker run -e MYSQL_ROOT_PASSWORD=pwd1234 -p 3306:3306 -v
c:/docker_data/mysql_data:/var/lib/mysql -d mysql

$ docker exec -it [MySQL 컨테이너 ID] bash
$ mysql -u root -p # 접속이 안 됨...
```

> 분명 **MYSQL_ROOT_PASSWORD** 값을 바꿔서 새로 컨테이너를 띄웠는데 비밀번호는 바뀌지 않은걸까? 이 부분 때문에 많은 분들이 헤맨다.

그 이유는 Volume으로 설정해둔 폴더에 이미 비밀번호 정보가 저장되어버렸기 때문이다.

[실습] Docker로 PostgreSQL 실행시켜보기

✓ Docker로 PostgreSQL 실행시켜보기

1. PostgreSQL 이미지를 바탕으로 컨테이너 실행시키기

```
$ cd /Users/jaeseong/Documents/Develop
$ mkdir docker-postgresql

$ docker run -e POSTGRES_PASSWORD=password123 -p 5432:5432 -v
c:/docker_data/postgresql_data:/var/lib/postgresql/data -d postgres
```

2. 컨테이너가 잘 실행되고 있는 지 체크

```
$ docker ps
```

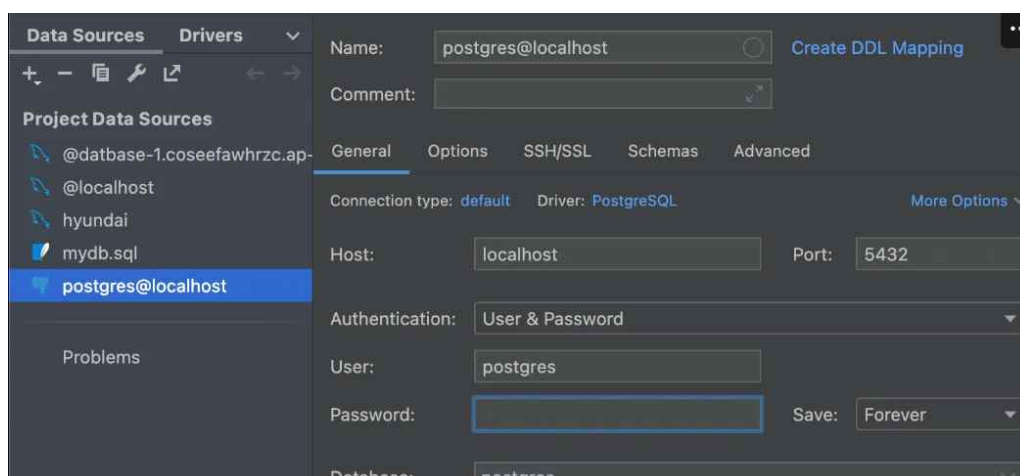
3. 컨테이너 실행시킬 때 에러 없이 잘 실행됐는 지 로그 체크

```
$ docker logs [컨테이너 ID 또는 컨테이너명]
```

4. PostgreSQL 컨테이너에 접속

```
$ docker exec -it [컨테이너 ID 또는 컨테이너명] bash
```

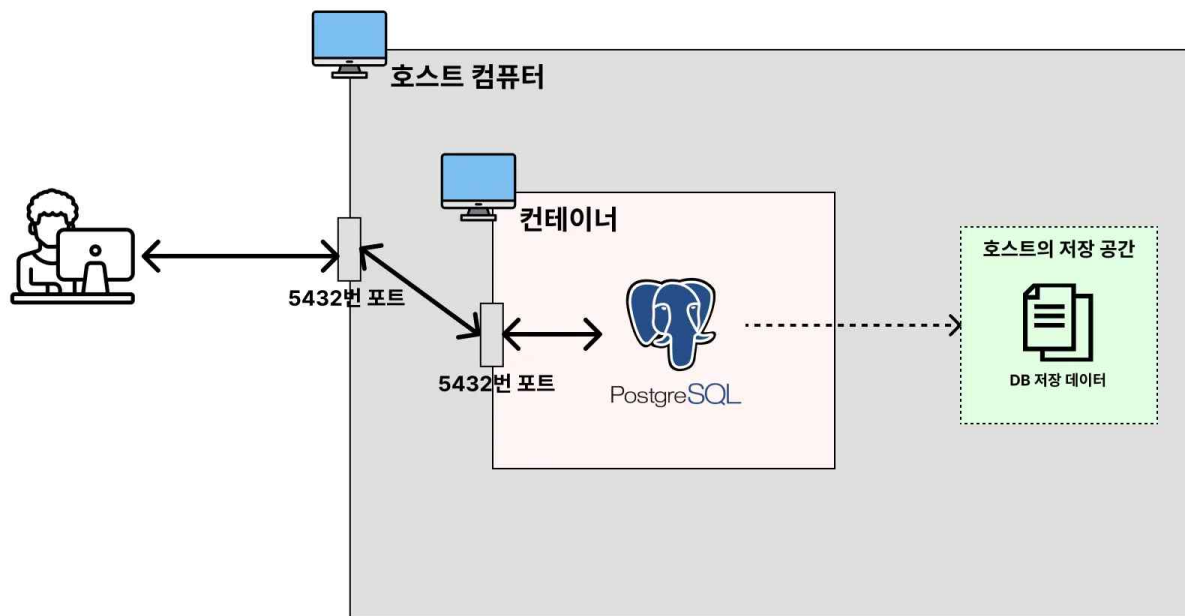
5. PostgreSQL 연결해보기



6. PostgreSQL 데이터가 볼륨에 잘 저장되고 있는 지 확인해보기

```
$ cd c:/docker_data/postgresql_data
$ ls
```

✓ 그림으로 이해하기



[실습] Docker로 MongoDB 실행시켜보기

✓ Docker로 MongoDB 실행시켜보기

1. MongoDB 이미지를 바탕으로 컨테이너 실행시키기

```
$ cd /Users/jaeseong/Documents/Develop
$ mkdir docker-mongodb

$ docker run -e MONGO_INITDB_ROOT_USERNAME=root -e MONGO_INITDB_ROOT_PASSWORD=password123 -p 27017:27017 -v c:/docker_data/mongodb_data:/data/db -d mongo
```

2. 컨테이너가 잘 실행되고 있는 지 체크

```
$ docker ps
```

3. 컨테이너 실행시킬 때 에러 없이 잘 실행됐는 지 로그 체크

```
$ docker logs [컨테이너 ID 또는 컨테이너명]
```

4. MongoDB 컨테이너에 접속

```
$ docker exec -it [컨테이너 ID 또는 컨테이너명] bash
```

5. 컨테이너에서 MongoDB에 접근하기

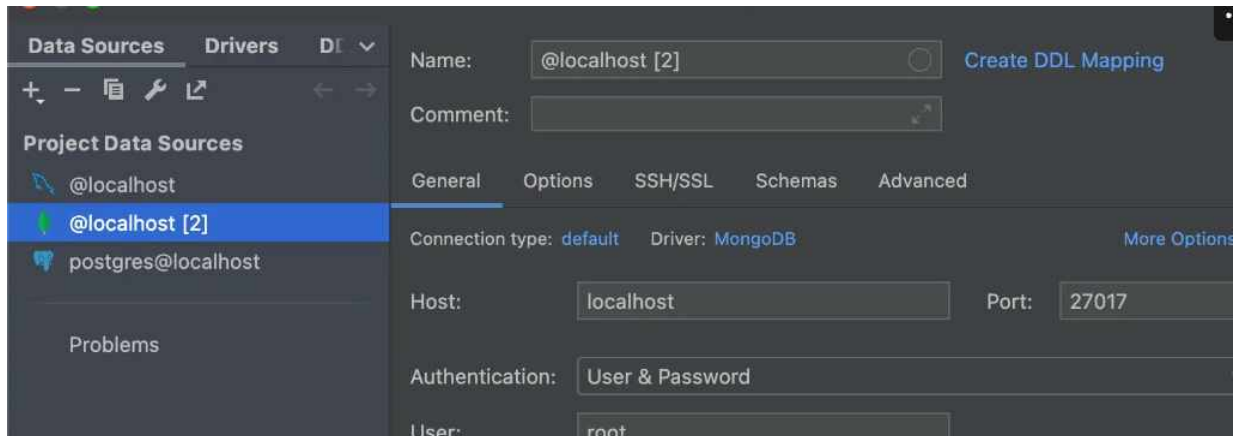
```
$ mongosh
```

```
root@a47e133c7583:/# mongosh
Current Mongosh Log ID: 64b2a17e78aabb1cbc71c66f
Connecting to:      mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+1.9.1
Using MongoDB:      6.0.6
Using Mongosh:      1.9.1

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

-----
The server generated these startup warnings when booting
2023-07-15T13:34:51.978+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/compatibility
```

6. MongoDB에 연결해보기



7. MongoDB 데이터가 볼륨에 잘 저장되고 있는 지 확인해보기

```
$ cd c:/docker_data/mongodb_data
$ ls
```

✓ 그림으로 이해하기

