K8S Demo Project

Organizing Frontend/Backend/Web/DB

모바일시스템공학과 조민욱
adrd1820@gmail.com

Docker

- docker –version
- sudo systemctl disable --now docker; sudo apt-get purge docker.io -y; sudo apt-get update; sudo apt-get install ca-certificates curl; sudo install -m 0755 -d /etc/apt/keyrings; sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc; sudo chmod a+r /etc/apt/keyrings/docker.asc; echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null; sudo apt-get update; sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin -y

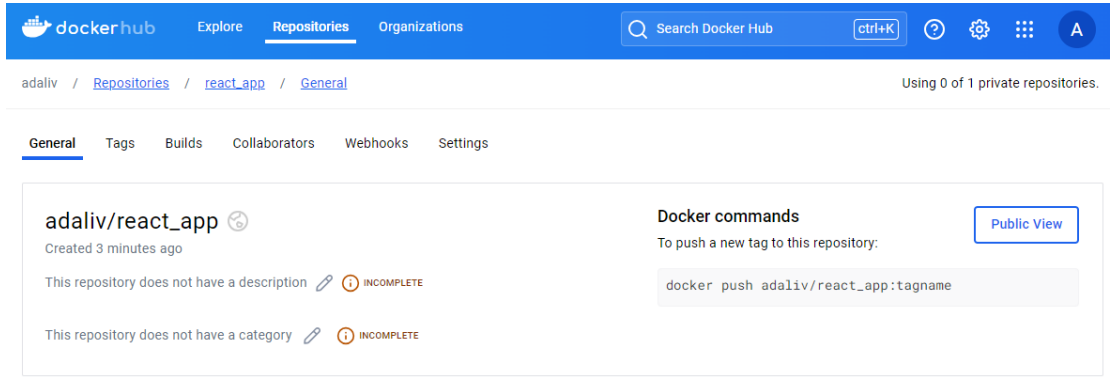
Docker

- 이미지 빌드 (docker 계정 이름/이미지 이름:버전)
- `sudo docker build -t adaliv/_app:1.0 .`
- 컨테이너 런 (컨테이너 이름, 포트 번호, 이미지 이름)
- `sudo docker container run --name webserver -d -p 3000:3000 adaliv/_app:1.0`
- 컨테이너 확인
- `sudo docker ps -a`
- 이미지 확인
- `sudo docker images`
- 컨테이너 삭제
- `sudo docker rm -f ~`
- 이미지 삭제
- `sudo docker rmi ~`

Docker

- Docker hub에 접속 후 Create repository
- Docker hub는 매우 예민한 친구이므로 이름을 작성할 때 소문자

- Docker hub image push
- `sudo docker push adaliv/_app:1.0`
- Docker hub image pull
- `sudo docker pull adaliv/_app:1.0`



- 기본적으로 sudo su 활용...
- 빌드 잔여 데이터 초기화 → sudo docker builder prune

```
sudo docker build -t adaliv/react_app:1.0 .
sudo docker container run --name webserver -d -p 3000:3000 adaliv/react_app:1.0
```

Frontend

```
FROM ubuntu

RUN apt-get update && \
    apt-get install -y npm && \
    npm cache clean --force

RUN mkdir /app
ADD . /app
WORKDIR /app

RUN npm install http-proxy-middleware &&
    npm install react-router-dom && \
    npm install axios

EXPOSE 3000
CMD npm start
```

```
sudo docker build -t adaliv/node_app:1.0 .
sudo docker container run --name server -d -p 5000:5000 adaliv/node_app:1.0
```

Backend

```
FROM ubuntu

RUN apt-get update && \
    apt-get install -y npm && \
    npm cache clean --force

RUN mkdir /app
ADD . /app
WORKDIR /app

RUN npm install express body-parser && \
    npm install cors

EXPOSE 5000
CMD node server.js
```

Docker

- 이미지 및 컨테이너 확인

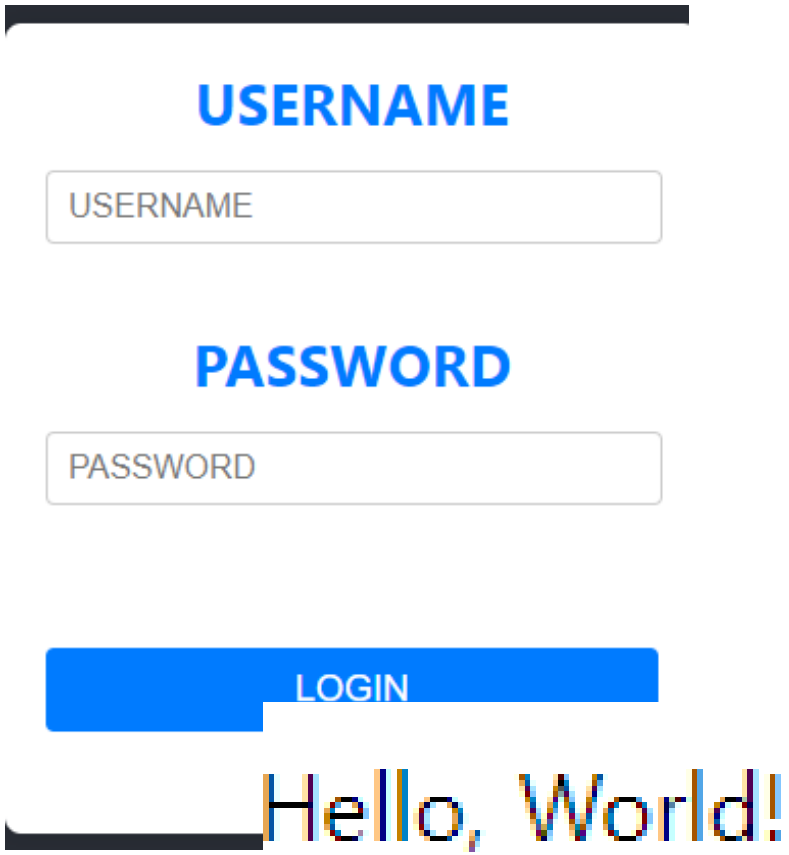
```
root@muchoubuntu:/home/minuk/my_demo_1/backend# docker images
REPOSITORY          TAG       IMAGE ID       CREATED          SIZE
adaliv/node_app     1.0       9606577fcc55   3 minutes ago    962MB
adaliv/react_app    1.0       02c911b4f9b8   16 minutes ago   1.31GB
```

```
root@muchoubuntu:/home/minuk/my_demo_1/backend# docker ps -a
CONTAINER ID   IMAGE                 COMMAND                  CREATED          STATUS          PORTS                                            NAMES
13b944284843   adaliv/node_app:1.0   "/bin/sh -c 'node se…"   18 seconds ago   Up 17 seconds   0.0.0.0:5000->5000/tcp, :::5000->5000/tcp        server
ad3a485f65cb   adaliv/react_app:1.0  "/bin/sh -c 'npm sta…"   12 minutes ago   Up 12 minutes   0.0.0.0:3000->3000/tcp, :::3000->3000/tcp        webserver
```

Docker

- http://localhost:3000/
- http://localhost:5000/

- 이미지를 통해 생성된 컨테이너 확인 가능
- 이제 서버와 DB를 연결해서 직접 사용자 확인



Database

1. DB initializing
2. DB connecting

Database

Dockerfile

```
FROM mysql:latest

ENV MYSQL_ROOT_PASSWORD=rootpassword
ENV MYSQL_DATABASE=myappdb

COPY init.sql /docker-entrypoint-initdb.d/

EXPOSE 3306
```

```
# 사용할 기본 이미지 설정
FROM mysql:latest

# 환경 변수 설정 (루트 비밀번호 및 초기 데이터베이스 이름)
ENV MYSQL_ROOT_PASSWORD=rootpassword
ENV MYSQL_DATABASE=myappdb

# SQL 초기화 스크립트 복사
COPY init.sql /docker-entrypoint-initdb.d/

# 포트 노출 (기본 MySQL 포트)
EXPOSE 3306
```

Database

- 테스트 삼아 DB를 돌려볼까요?
  - sudo docker build -t adaliv/mysql_app:1.0 .
  - sudo docker container run --name db -d -p 3306:3306 adaliv/mysql_app:1.0

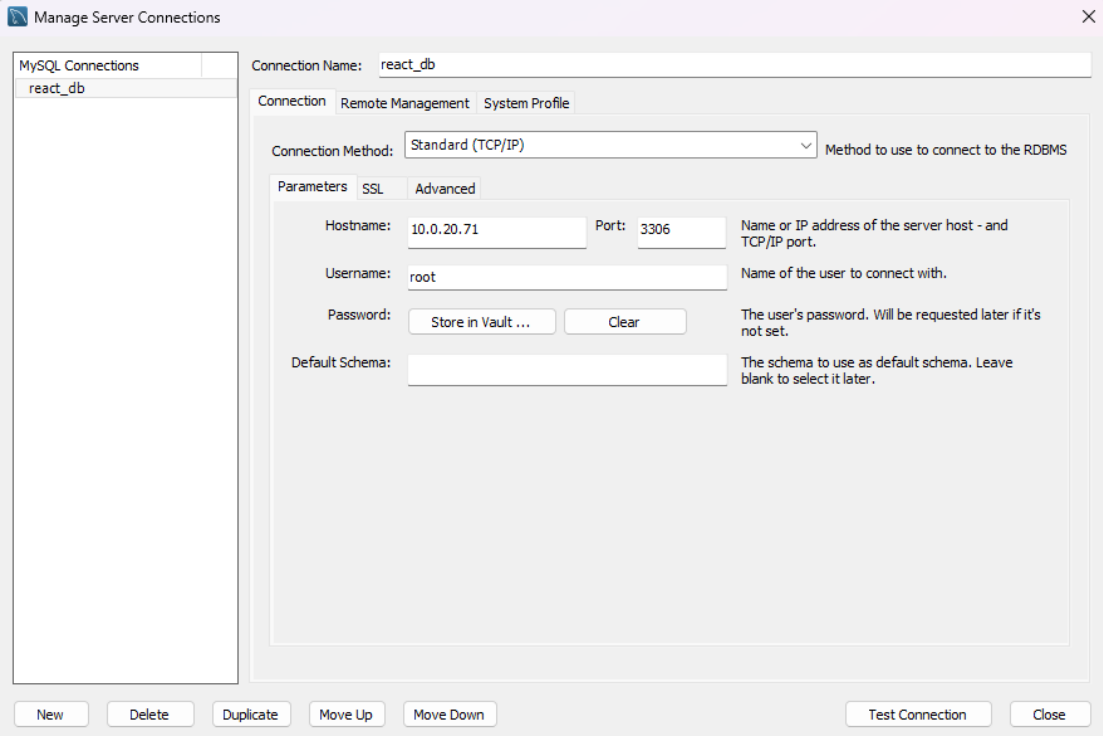
- MySQL Workbench를 통해 확인해볼까요?
  - https://dev.mysql.com/downloads/workbench/
  - 현재 생성된 DB를 GUI를 통해 관리

```
minuk@muchoubuntu:~/my_demo_1/database$ sudo docker images
REPOSITORY          TAG       IMAGE ID       CREATED          SIZE
adaliv/mysql_app    1.0       c86a2918cb65   4 minutes ago    586MB
adaliv/node_app     1.0       9606577fcc55   16 hours ago     962MB
adaliv/react_app    1.0       02c911b4f9b8   16 hours ago     1.31GB
minuk@muchoubuntu:~/my_demo_1/database$ sudo docker ps -a
CONTAINER ID   IMAGE                  COMMAND                  CREATED          STATUS          PORTS                                                  NAMES
0233dbdec875   adaliv/mysql_app:1.0   "docker-entrypoint.s…"   13 seconds ago   Up 12 seconds   0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp   db
13b944284843   adaliv/node_app:1.0    "/bin/sh -c 'node se…"   16 hours ago     Up 16 hours     0.0.0.0:5000->5000/tcp, :::5000->5000/tcp              server
ad3a485f65cb   adaliv/react_app:1.0   "/bin/sh -c 'npm sta…"   16 hours ago     Up 16 hours     0.0.0.0:3000->3000/tcp, :::3000->3000/tcp              webserver
```

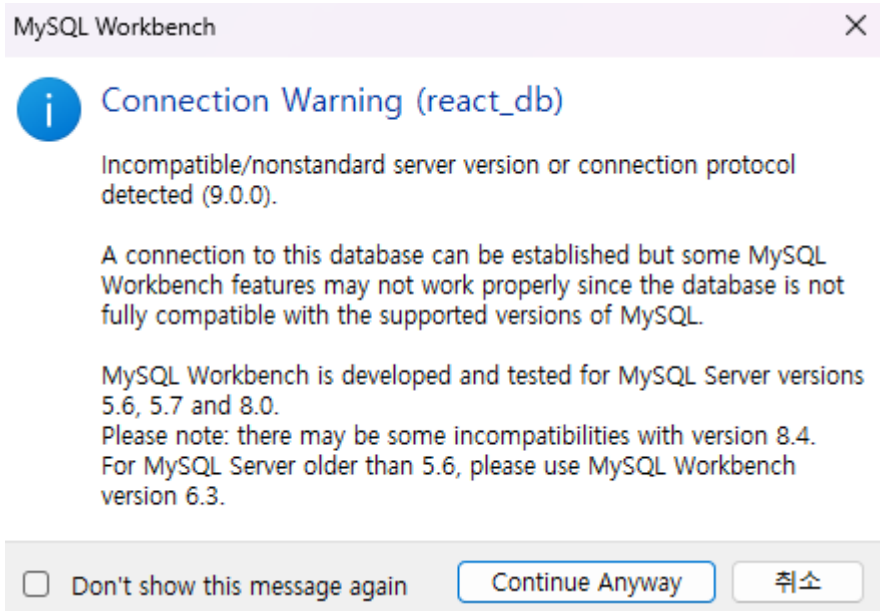
Database

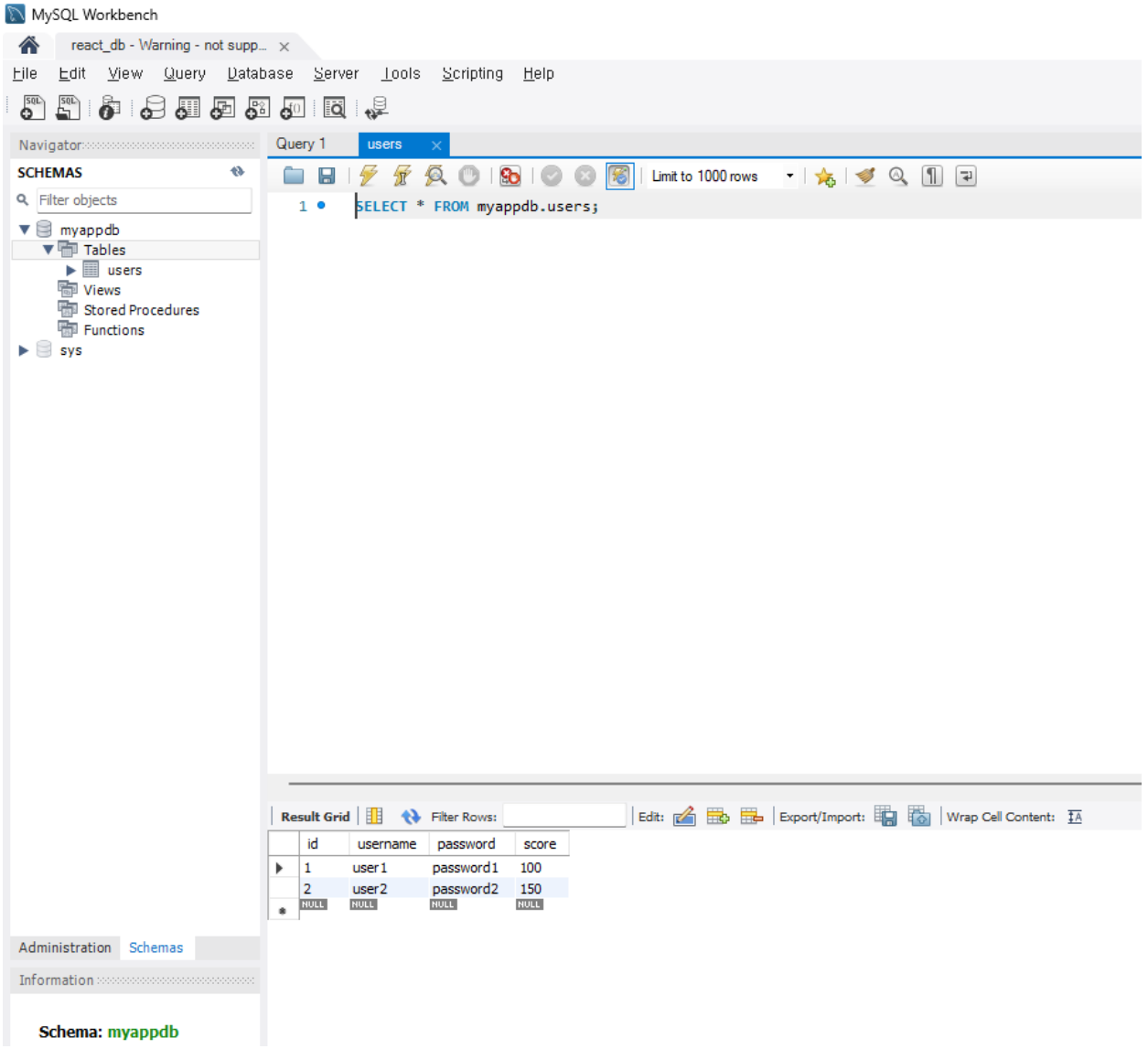
- 호스트 주소 → K8S VM 노드 주소
- 호스트 포트 → Docker file에서 노출한 포트
- 이름 → root
- 비번 → Docker file에서 지정한 비번

- Test Connection 클릭
  - 버전이 다르다고 뜰 수 있는데…
  - 별 상관 없을 예정… 아마도…
  - 고급 기능을 사용하지 않으니까…



Database


MySQL Workbench
Connection Warning (react_db)
Incompatible/nonstandard server version or connection protocol detected (9.0.0).
A connection to this database can be established but some MySQL Workbench features may not work properly since the database is not fully compatible with the supported versions of MySQL.
MySQL Workbench is developed and tested for MySQL Server versions 5.6, 5.7 and 8.0.
Please note: there may be some incompatibilities with version 8.4.
For MySQL Server older than 5.6, please use MySQL Workbench version 6.3.
Don't show this message again
Continue Anyway
취소


MySQL Workbench
react_db - Warning - not supp...
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
myappdb
Tables
users
Views
Stored Procedures
Functions
sys
Query 1
users
Limit to 1000 rows
1
SELECT * FROM myappdb.users;
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
id username password score
1 user1 password1 100
2 user2 password2 150
Administration Schemas
Information
Schema: myappdb

Database

- 지금까지 한 것?
  - DB 초기 값 지정
  - DB 초기 테이블, 객체 생성

- 앞으로 할 것?
  - DB와 Node.js를 연결… HOW?
  - npm install mysql2
  - Backend에 db.js 파일 작성
  - Server.js에 쿼리 사용

```
const mysql = require('mysql2');

const db = mysql.createConnection({
    host: 'localhost',
    port: 3306,
    user: 'root',
    password: 'rootpassword',
    database: 'myappdb',
});

db.connect(err => {
    if (err) {
        console.error('Error connecting to MySQL database:', err);
        return;
    }
    console.log('Connected to MySQL database');
});

module.exports = db;
```

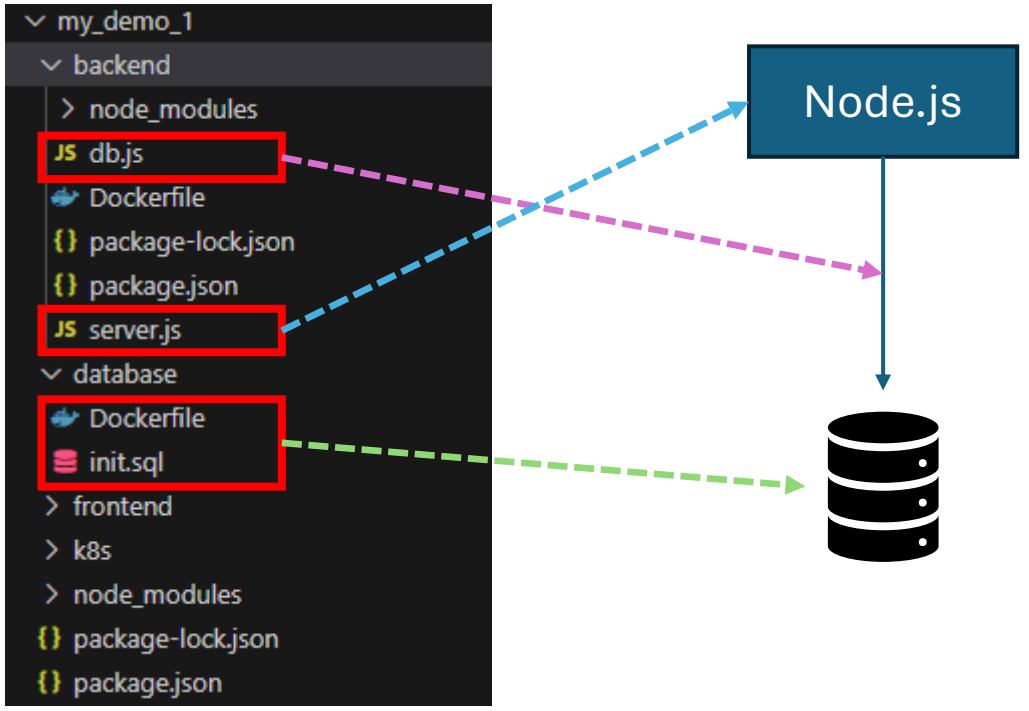
Database

```
const db = mysql.createConnection({
    host: 'localhost',
    port: 3306,
    user: 'root',
    password: 'rootpassword',
    database: 'myappdb',
});
```

```
ENV MYSQL_ROOT_PASSWORD=rootpassword
ENV MYSQL_DATABASE=myappdb
```

반드시 이미지 빌드할 때 지정한 포트,
비번 등을 그대로 지정하여 사용할 것...

```
minuk@muchoubuntu:~/my_demo_1/backend$ node db.js
Connected to MySQL database
```

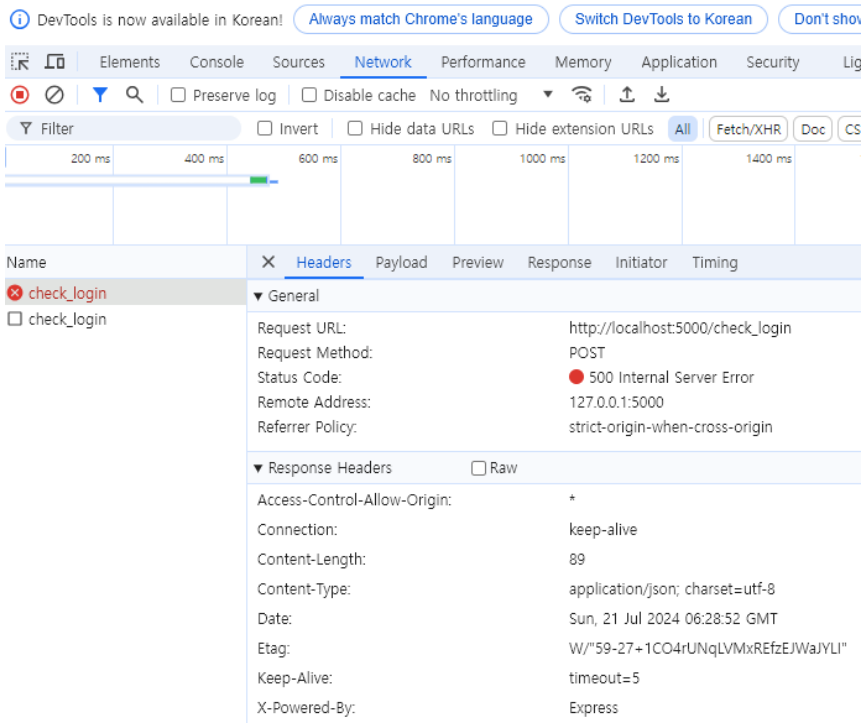


Database

- 연결되었으니... 이제 server.js에서 해당 DB만 사용하면 되겠죠?

- 하지만 발생하는 서버 에러 문제
  - Frontend image
  - Backend image → 작동X
  - DB image

- 완전하지 않지만 이렇게 하면 일시적 문제 해결
  - Frontend image
  - Backend local → node server.js
  - DB image



Database

- CORS 문제인가...?
  - Cross Origin Resource Sharing
  - SOP(Same Origin Policy) → localhost에서 localhost 접근 가능
  - (127.0.0.1 != localhost) → SOP error!
  - But we have already configured CORS setting...

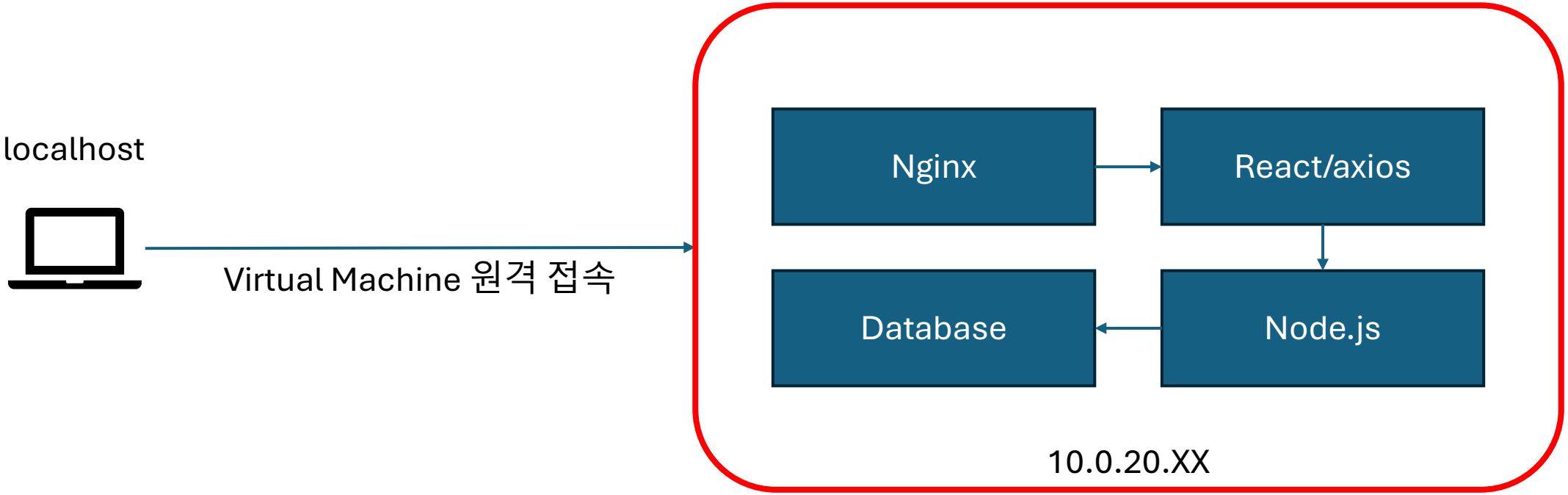
- 다시 이미지 빌드
  - sudo docker logs <컨테이너 ID>
  - 문제 해결 → backend/db.js 파일에서 호스트 주소 문제
  - 땅만 죽어라 판 내 이틀... 애꿎은 Proxy, CORS만 건드리다가...
  - 데이터베이스 설치 명령어 추가 및 다시 빌드!

```
const db = mysql.createConnection({
    host: '10.0.20.71', // host: 'localhost'
    port: 3306,
    user: 'root',
    password: 'rootpassword',
    database: 'myappdb',
});
```

```
FROM ubuntu

RUN apt-get update && \
    apt-get install -y npm && \
    npm cache clean --force

RUN mkdir /app
ADD . /app
WORKDIR /app

RUN npm install express body-parser && \
    npm install cors && \
    npm install mysql2

EXPOSE 5000
CMD ["node", "server.js"]
```

```
minuk@muchoubuntu:~$ sudo docker ps -a
[sudo] password for minuk:
CONTAINER ID   IMAGE                  COMMAND                  CREATED         STATUS         PORTS                                                  NAMES
3431f0404c61   adaliv/node_app:1.0    "node server.js"         6 minutes ago   Up 6 minutes   0.0.0.0:5000->5000/tcp, :::5000->5000/tcp              server
93d8e7b2b0a1   adaliv/mysql_app:1.0   "docker-entrypoint.s…"   2 hours ago     Up 2 hours     0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp   db
30f1883aea48   adaliv/react_app:1.0   "npm start"              2 hours ago     Up 2 hours     0.0.0.0:3000->3000/tcp, :::3000->3000/tcp              webserver
```

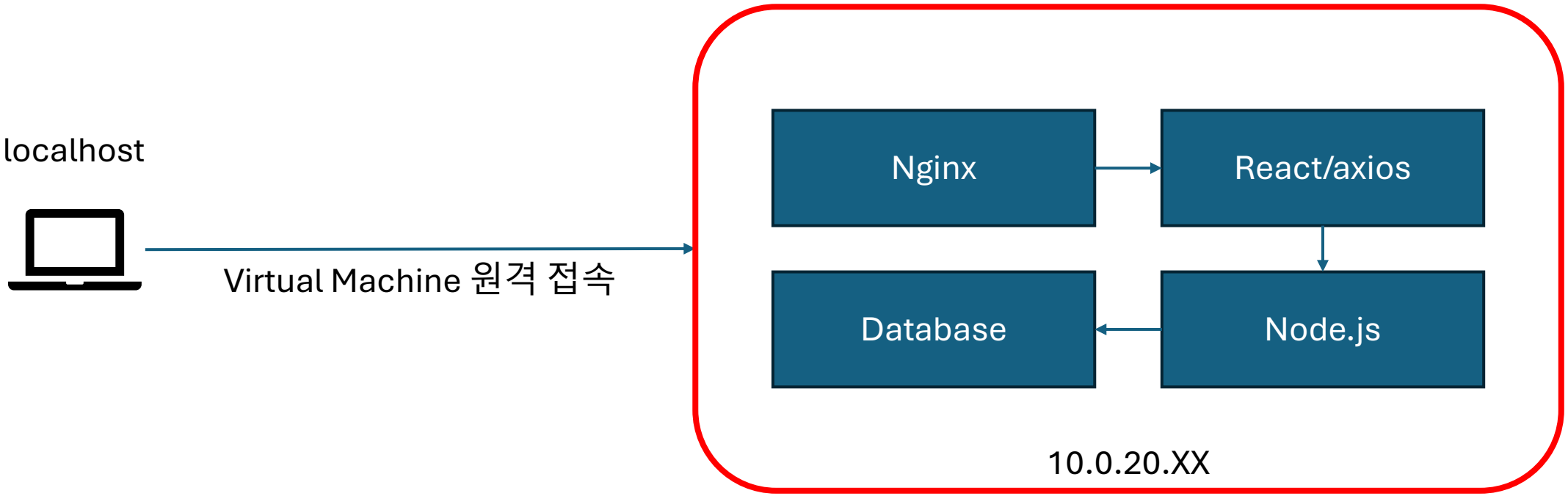
Localhost vs VM-IP



말 그대로 원격접속을 의미하며, VM 서버 내부에 띄워진 앱들은 모두 로컬 호스트이다.

→ 즉 외부에 공개되거나, VM 내부에서 독립적인 객체로 띄워질 때 10.0.20.XX IP 형태를 띄게 된다.
→ 처음에 DB 이미지로 컨테이너를 만들었을 때, DB는 내부 독립 객체로 10.0.20.XX:3306에 존재한다.

Localhost vs VM-IP

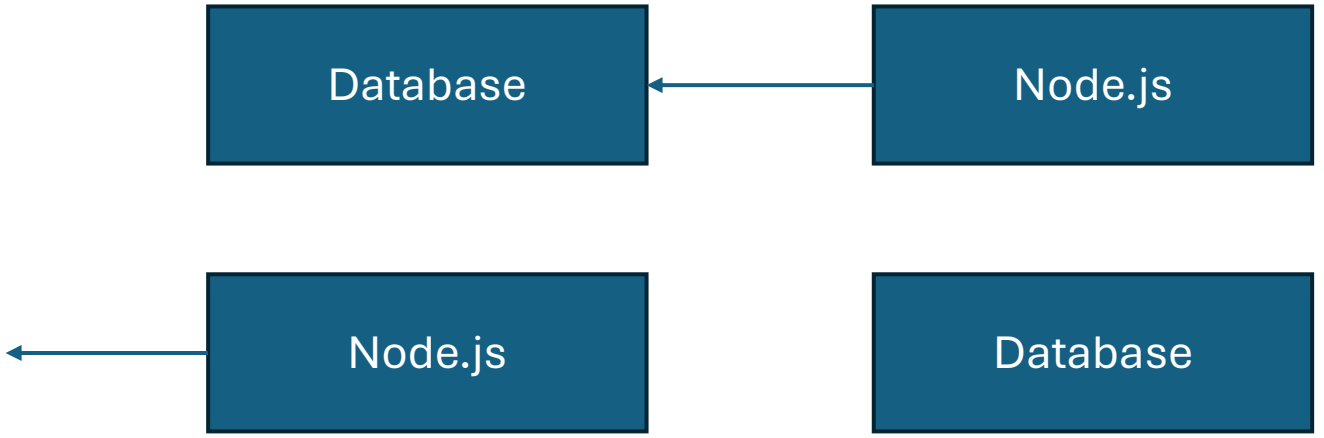


Database를 이미지 컨테이너로 돌리면 외부에서 접속가능한 IP, Port를 이용하여 오픈
따라서 MySQL Workbench에서 해당 IP, Port를 이용해서 접속

Localhost vs VM-IP

- 이렇게 살펴보면 빌드 순서 또한 중요합니다.
- Frontend → Database → Backend 순서로 컨테이너를 만들어야 합니다.
  - Backend에서 DB로 접속을 시도할 때, DB가 있어야 접속을 할거 아닙니까.
  - Backend 컨테이너가 생성될 때, 자동적으로 DB와 연결을 하려고 합니다.
  - 하지만 DB를 먼저 만들어 놓지 않았다면, Node.js가 접속하려는 10.0.20.XX에는 어떤 것도 없는 상태입니다.

?



Localhost vs VM-IP

DB가 VM 서버 내에서 IP와 Port를 통해 열린다는 의미

```
minuk@muchoubuntu:~$ sudo docker ps -a
CONTAINER ID   IMAGE                  COMMAND                  CREATED
2eeec0bbab1b   adaliv/mysql_app:1.0   "docker-entrypoint.s…"   3 seconds ago
46db31f3e973   adaliv/node_app:1.0    "node server.js"         13 seconds ago
a5b6d8e05afe   adaliv/react_app:1.0   "npm start"              11 minutes ago
minuk@muchoubuntu:~$ sudo docker logs 46db31f3e973
Server is running on http://localhost:5000
Error connecting to MySQL database: Error: connect ECONNREFUSED 10.0.20.71:3306
    at TCPConnectWrap.afterConnect [as oncomplete] (node:net:1549:16) {
  errno: -111,
  code: 'ECONNREFUSED',
  syscall: 'connect',
  address: '10.0.20.71',
  port: 3306,
  fatal: true
}
```

Backend 먼저 생성한 경우... 에러

```
minuk@muchoubuntu:~$ sudo docker ps -a
CONTAINER ID   IMAGE                  COMMAND                  CREATED
40ae73d8ec01   adaliv/node_app:1.0    "node server.js"         5 seconds ago
bcdb9acce681   adaliv/mysql_app:1.0   "docker-entrypoint.s…"   13 seconds ago
a5b6d8e05afe   adaliv/react_app:1.0   "npm start"              19 minutes ago
minuk@muchoubuntu:~$ sudo docker logs 40ae73d8ec01
Server is running on http://localhost:5000
Connected to MySQL database
```

Database 먼저 생성한 경우... 성공