♣

포팅메뉴얼

FrontEnd

```
[ 기본환경 ]
React : 18.2.0
Nivo : 0.80.0

[ package.json ]
{
  "name": "frontend",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@emotion/react": "^11.10.4",
    "@emotion/styled": "^11.10.4",
    "@mui/icons-material": "^5.10.6",
    "@mui/material": "^5.10.6",
    "@nivo/bar": "^0.80.0",
    "@nivo/core": "^0.80.0",
    "@nivo/pie": "^0.80.0",
    "@nivo/radar": "^0.80.0",
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^0.27.2",
    "bootstrap": "^5.2.1",
    "mdb-react-ui-kit": "^4.2.0",
    "react": "^18.2.0",
    "react-bootstrap": "^2.5.0",
    "react-dom": "^18.2.0",
    "react-icons": "^4.4.0",
    "react-js-pagination": "^3.0.3",
    "react-material-ui-carousel": "^3.4.2",
    "react-router-dom": "^6.4.0",
    "react-scripts": "5.0.1",
    "react-slick": "^0.29.0",
    "slick-carousel": "^1.8.1",
    "styled-components": "^5.3.5",
    "sweetalert": "^2.1.2",
    "swiper": "^8.4.2",
    "web-vitals": "^2.1.4"
  }
```

- **개발 버전 및 환경**

```
프로젝스 생성 시 create-react-app 활용

-Nivo (데이터 그래프/차트 변환)
Nivo/bar : 0.80.0
Nivo/core : 0.80.0
Nivo/pie : 0.80.0
Nivo/radar : 0.80.0

-React Router DOM
react-router-dom : 6.4.0

-Axios
Axios : 0.27.2

-MUI
MUI/material : 5.10.6
MUI/icons-material : 5.10.6

-기타
sweetalert : 2.1.2 (알림창)
styled-components : 5.3.5 (디자인)
swiper : 8.4.2 (이미지 카드 표현)
```

BackEnd

```
[ 기본환경 ]
Django : 4.1.1
python : 3.10.4
Anaconda : 4.14.0
pip : 22.2.2

[ Anaconda 가상환경 셋팅 ]
 - 가상환경 (파이썬 3.10.4) 로 설정하여 셋팅
 - 가상환경 폴더 내에 설치 확인
 - 가상환경 활성화

conda create --name backend python=3.10.4
conda env list
conda active backend

[ requirements.txt ]
asgiref==3.5.2
certifi==2022.6.15
contourpy==1.0.5
cycler==0.11.0
```

```
Django==4.1.1
django-cors-headers==3.13.0
django-json==0.27
django-rest-auth==0.9.5
djangorestframework==3.14.0
gunicorn==20.1.0
mysqlclient==2.1.1
fonttools==4.37.3
joblib==1.2.0
kiwisolver==1.4.4
matplotlib==3.6.0
numpy==1.23.3
packaging==21.3
pandas==1.5.0
Pillow==9.2.0
pip==22.1.2
pyparsing==3.0.9
python-dateutil==2.8.2
pytz==2022.2.1
scikit-learn==1.1.2
scipy==1.9.1
seaborn==0.12.0
setuptools==63.4.1
six==1.16.0
sklearn==0.0
sqlparse==0.4.2
threadpoolctl==3.1.0
tzdata==2022.2
wheel==0.37.1
wincertstore==0.2
SQLAlchemy==1.4.41
PyMySQL==1.0.2
greenlet==1.1.3

[ requirements.txt 생성 및 가상환경 ]
pip freeze > requirements.txt
pip install -r requirements.txt

[ run ]
python manage.py makemigrations
python manage.py migrate
python manage.py runserver
```

Server

1. docker 설치

```
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io docker-compose
```

2. jenkins 컨테이너 생성

docker-compose.yml

```
version: '3'

services:
    jenkins:
        image: jenkins/jenkins:lts
        container_name: jenkins
        volumes:
            - /var/run/docker.sock:/var/run/docker.sock
            - /jenkins:/var/jenkins_home
        ports:
            - "9090:8080"
        privileged: true
        user: root
```

9090 포트에 jenkins 8080 포트 매핑 > 9090 포트로 접속

3 . Jenkins 에서 프로젝트 생성 후, Gitlab 과 연결 > WebHook 을 통해 master merge시, 자동빌드

4 . Jenkins 에서 docker 빌드

```
sudo docker exec -it jenkins bash
apt update
apt install docker-ce docker-ce-cli containerd.io docker-compose
```

젠킨스의 bash shell 에 접속하여, Jenkins Container 에 Docker 설치

5 . DockerFile 작성

(Django)

```
FROM python:3.10.4WORKDIR /var/jenkins_home/workspace/deploy/backend
COPY requirements.txt ./
```

```
RUN pip install --upgrade pip
RUN pip install -r requirements.txt
COPY . .
CMD ["python", "manage.py", "runserver", "0.0.0.0:8080"]
```

(React)

```
FROM node:16.17.0 as build-stage
WORKDIR /var/jenkins_home/workspace/deploy/frontend
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build
FROM nginx:stable-alpine as production-stage

COPY --from=build-stage /var/jenkins_home/workspace/deploy/frontend/build /usr/share/n
ginx/html
COPY --from=build-stage /var/jenkins_home/workspace/deploy/frontend/deploy_conf/nginx.
conf /etc/nginx/conf.d/default.conf
EXPOSE 80CMD ["nginx", "-g","daemon off;"]
```

6 . Jenkins 에서

```
docker image prune -a --force
mkdir -p /var/jenkins_home/images_tar
cd /var/jenkins_home/workspace/deploy/frontend/
docker build -t react .
docker save react > /var/jenkins_home/images_tar/react.tar

cd /var/jenkins_home/workspace/deploy/backend/
docker build -t django .
docker save django > /var/jenkins_home/images_tar/django.tar

ls /var/jenkins_home/images_tar
```

각 react, django 프로젝트 폴더에서 도커 이미지 빌드 후 압축 > pem 키 등록으로 배포

```
sudo docker run -it -d --rm -p 80:80 -p 443:443 --name react react
echo "Run testproject_react"
sudo docker run -it -d --rm -p 8080:8080  --name django django
echo "Run testproject"
```

react 컨테이너 생성 > 80,443 포트 연결

django 컨테이너 생성 > 8080 포트 연결

7. nginx 를 통해 백엔드 서비스를 80 포트 /api 를 통해 접속하도록

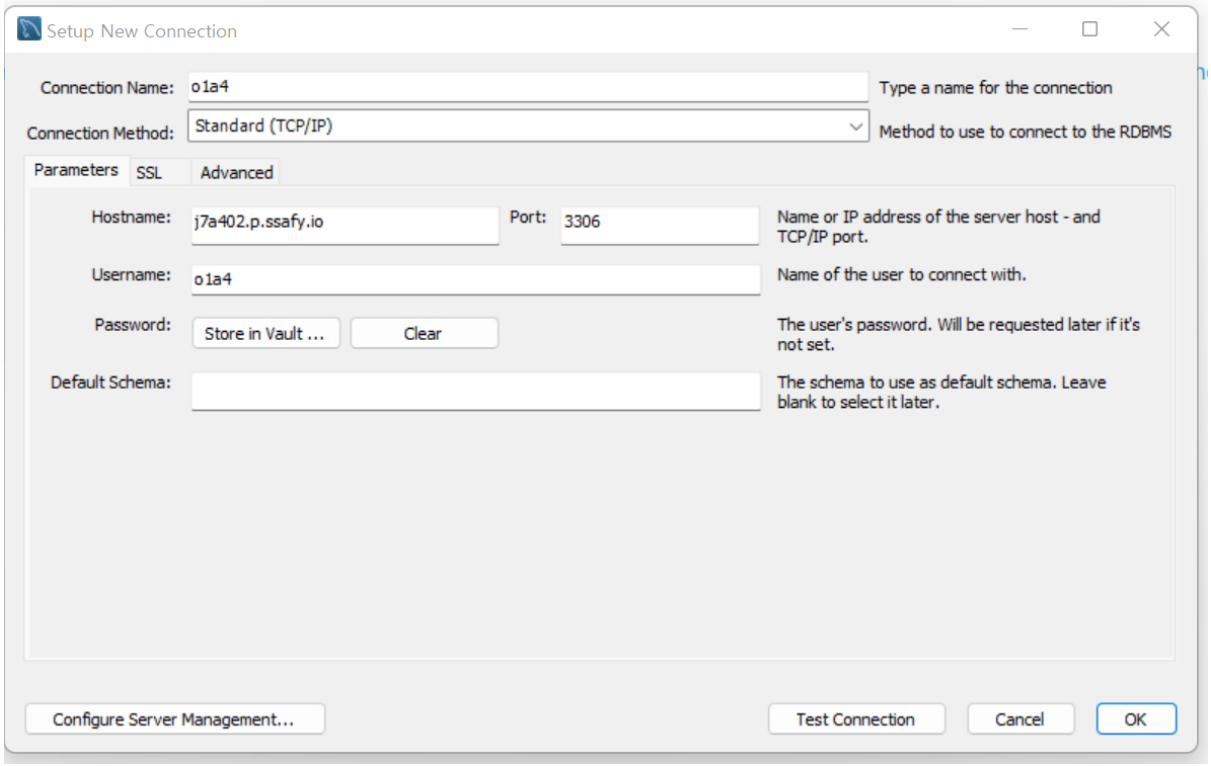
nginx.conf

```
location /api {
        proxy_pass http://backend/;
        proxy_redirect     off;
        proxy_set_header   Host $host;
        proxy_set_header   X-Real-IP $remote_addr;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
    }
```

Mysql 연결

```
sudo apt install mysql-server
mysql -u root -p
```

mysql 설치 및 실행



MySQL workbench 에서 연결