B304 포팅 매뉴얼

프로젝트 기술 스택

프론트엔드

- 기술 스택 및 버전
  - Node.js 18.17.1, react-native-cli
  - TypeScript
- 사용 툴
  - Visual Studio Code, Android Studio

백엔드

- 기술 스택 및 버전
  - Mysql 5.7.35, Spring boot 2.7.15, SpringSecurity, JDK 11
  - Nginx , Jenkins , AWS EC2, AWS S3,
- 사용 툴
  - IntelliJ , Mobaxterm, MySql workbench, Postman

개발 환경 세팅

프론트엔드

- Node.js 설치
  - LTS 18.17.1
  - npm 9.6.7
- 파이썬 설치
  - 리액트 네이티브의 빌드 시스템은 파이썬 사용
- React Native CLI 설치
  - npm 명령어를 통해 설치

```
npm install -g react-native-cli
```

- JDK 설치
- 안드로이드 스튜디오 설치
  - SDK 33.0.0
- react-naitve CLI 명령어를 통해 프로젝트 생성

```
npx react-native init SampleApp
```

EC2 세팅

EC2 접속

- MobaXterm SSH로 접속
- Remote host : j9b304.p.ssafy.io

방화벽 설정하기 - sudo ufw allow {포트 번호}

```
sudo ufw allow ssh
sudo ufw allow http
sudo ufw allow https
```

DB 설정

mysql 설치하기

```
sudo apt-get update
sudo apt-get install mysql-server

//mysql 접속
sudo mysql -u root //초기에는 root로 -p없이 로그인 하면 돼
```

mysql 접속 후

```
//사용할 database 만들기
create database puppy;

//database 만들어 졌는지 확인하기
show databases;
```

사용할 user 생성하기

```
//db바꾸기
use mysql;

//사용자에게 권한 주기 '%'->모든 권한 주기
create user '사용자 이름'@'%' identified by '비밀번호';
grant all on 'db이름'.* to '권한 줄 사용자 이름'@'%';
(grant all on puppy.* to 'puppy'@'%';)

//권한 잘 주어졌는지 확인
show grants for 'puppy'@'%';
```

```
mysql> show grants for 'puppy'@'%';
+-------------------------------------------------+
| Grants for puppy@%                              |
+-------------------------------------------------+
| GRANT USAGE ON *.* TO `puppy`@`%`               |
| GRANT ALL PRIVILEGES ON `puppy`.* TO `puppy`@`%` |
+-------------------------------------------------+
2 rows in set (0.00 sec)
```

로컬 workbench에서 ec2 DB에 접속하기 위한 추가 설정

- 127.0.0.1로 설정 되어있는 bind-address값을 0.0.0.0으로 수정해서 외부 접속 허용

```
# If MySQL is running as a replication slave, this should be
# changed. Ref https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_tmpdir
# tmpdir                = /tmp
#
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address            = 0.0.0.0
mysqlx-bind-address     = 127.0.0.1
#
```

- 해킹 위험때문에 포트 변경 - 1999

```
sudo vi /etc/mysql/mysql.conf.d/mysqld.cnf

#해당 설정 파일로 가서 포트번호를 바꿔주자!!

#껐다 켜주자
sudo service mysql stop
sudo service mysql start
```

- 방화벽 설정

```
sudo ufw allow 1999
```

Docker + Jenkins + Spring Boot 배포

EC2에 Docker 설치하기

Install Docker Engine on Ubuntu

Jumpstart your client-side server applications with Docker Engine on Ubuntu. This guide details prerequisites and multiple methods to install.

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

Install Docker Engine on Ubuntu

Install using the Apt repository

Before you install Docker Engine for the first time on a new host machine, you need to set up the Docker repository. Afterward, you can install and update Docker from the repository.

1. Set up Docker's Apt repository.

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

# Add the repository to Apt sources:
echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

2. Install the Docker packages.

Latest | Specific version

To install the latest version, run:

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

- docker engine과 그에 따른 plugin설치

```
sudo apt install docker-compose

#정상 설치 되었는지 확인
sudo docker -v
sudo docker compose version
```

```
ubuntu@ip-172-26-1-113:~$ sudo docker -v
Docker version 24.0.6, build ed223bc
ubuntu@ip-172-26-1-113:~$ sudo docker compose version
Docker Compose version v2.21.0
```

Jenkins 설치

- jenkins container를 실행시킬 dockcer-compose 만들기

```
sudo vim docker-compose.yml

## 아래 작성 된 사항은 jenkins 기본이미지로 jdk 11버전을 지원해주기 때문에jdk 17이상을 설치 해야 한다면
 ## —> image: jenkins/jenkins:jdk17 이렇게 수정해 주면 됨
## jenkins 기본 포트는 8080인데 9090포트 사용하도록 지정해줌

# (i) : docker - compose 파일에 필요한 스펙을 작성 // 아래 캡쳐 화면 작성하기
# (ctrl+c) : 작성 완료
# (:wq) : 저장, 나오기
```

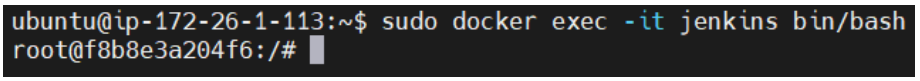
```
version: '3'

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"
        user: root
```

```
sudo docker-compose up -d

##정상적으로 jenkins container가 실행 되고 있는지 확인
sudo docker ps
##정상적으로 실행 되고 있지 않다면 해당 container가 어떤 상태인지 확인
sudo docker ps -a
```

```
ubuntu@ip-172-26-1-113:~$ sudo docker ps
CONTAINER ID   IMAGE                 COMMAND                  CREATED          STATUS
        PORTS                                                 NAMES
f8b8e3a204f6   jenkins/jenkins:lts   "/usr/bin/tini -- /u…"   55 seconds ago   Up 51 se
conds   50000/tcp, 0.0.0.0:9090->8080/tcp, :::9090->8080/tcp   jenkins
```

Jenkins 컨테이너 내부에 접속해서 docker 설치

```
sudo docker exec -it jenkins bin/bash
## root~~ 나오면
## jenkins 컨테이너 내부에 접속 완료
```

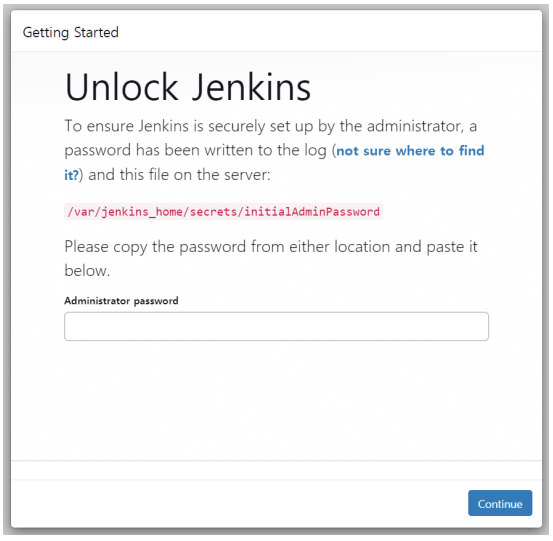


- docker 설치

```
##docker 설치
apt-get update
apt-get install ca-certificates curl gnupg lsb-release
mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \
  $(lsb_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null
apt-get update
apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin docker-compose
```

Jenkins 세팅

- jenkins : http://도메인:9090포트로 접속 (http://j9b304.p.ssafy.io:9090)
- 초기 접속화면 Unlock Jenkins



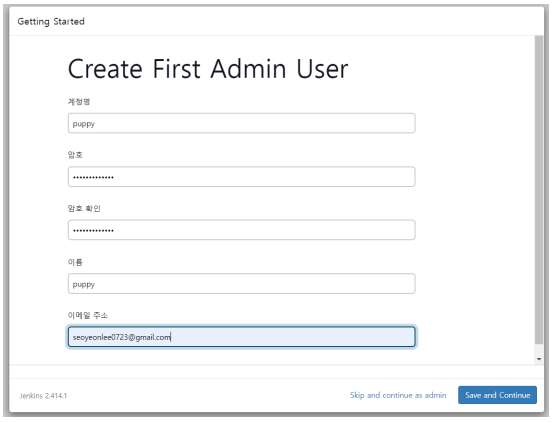
  - admin password : Jenkins container의 log에서 확인 가능

```
sudo docker logs jenkins
```

  - `cat /var/lib/jenkins/secrets/initialAdminPassword`
  - 여기서 비밀번호를 확인하고 입력

```
*************************************************************
*************************************************************
*************************************************************

Jenkins initial setup is required. An admin user has been created and a password generat
ed.
Please use the following password to proceed to installation:

34d291771d3d443ca182fbb7d37d37a0

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword

*************************************************************
*************************************************************
*************************************************************
```

- install suggested plugins 선택
- Create First Admin User



  - 계정명 : puppy
  - 암호 : spongebob1004
  - 이름 : puppy
- Jenkins Url : default사용함
- plugin 설치
  - GitLab
  - Generic Webhook Trigger
  - GitLab API
  - GitLab Authenication
  - Loading plugin extention
  - NodeJs (자동 배포로 프론트엔트 빌드 시 필요)
  - 추후에도 원하는 plugin 설치 가능

Jenkins Credential 설정

- Add Credentials 클릭

- gitlab에서 발급받은 access token 넣어주기

jenkins GitLab Connection 등록

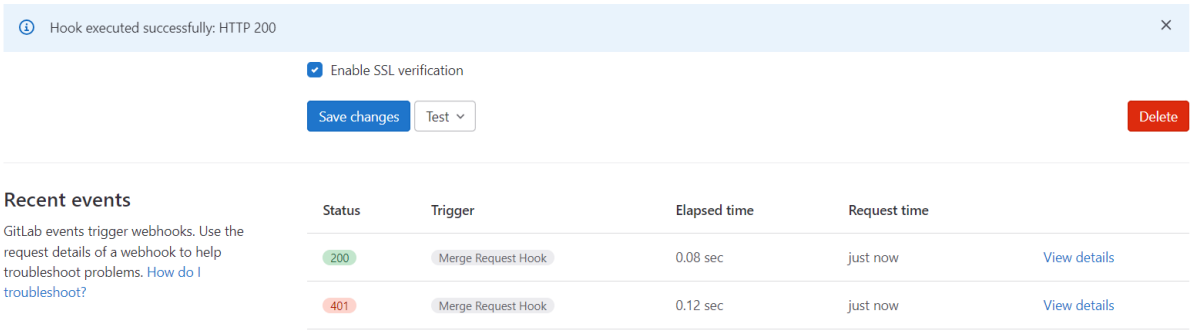
- Jenkins관리 → System → GitLab 이동
  - 원하는 connection 이름 설정
  - Gitlab 주소 입력
  - 앞서 만든 Credential 연결

Jenkins pipeline 생성

- + 새로운 Item → 이름 입력, Pipeline 선택 → ok
- 구성 → build Trigger 이동
  - build를 유발할 Tirgger 옵션을 선택하여 적용
  - 고급을 눌러 webhook 설정을 위한 Secret Token을 발급

Gitlab webhook 설정

- jenkins 작접물의 변화를 감지하여 build, run 하기 위해서는 webHook 필수!
- gitlab project → settings → webhooks
- Test 눌렀을 때 200 응답이 리턴되면 성공!



application.yml

```
sudo docker exec -it jenkins bin/bash
cd var/jenkins_home/workspace
```

```
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/b304?useSSL=false&serverTimezone=Asia/Seoul&characterEncoding=UTF-8
    username: ssafy
    password: ssafy
  jpa:
    database-platform: org.hibernate.dialect.MySQL5InnoDBDialect
    show-sql: true
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        format_sql: true
  security:
    user:
      name: user
      password: 1234
    oauth2:
      client:
        registration:
          kakao:
            client-id: 6e6828052f9580b3bc77e19c8b639327
            redirect-uri: "http://localhost:8080/oauth/kakao"
            client-authentication-method: POST
            authorization-grant-type: authorization_code
            scope: profile_nickname, account_email,gender,age_range #동의 항목
            client-name: Kakao
        provider:
          kakao:
            authorization-uri: https://kauth.kakao.com/oauth/authorize
            token-uri: https://kauth.kakao.com/oauth/token
            user-info-uri: https://kapi.kakao.com/v2/user/me
            user-name-attribute: id

springdoc:
  default-consumes-media-type: application/json;charset=UTF-8
  default-produces-media-type: application/json;charset=UTF-8
  swagger-ui:
    path: /index.html
    disable-swagger-default-url: true
    display-request-duration: true
    operations-sorter: alpha

server:
  port: 8080

jwt:
  #  secret key는 256비트 이상으로 만들것
  secret: beggVJDH3fB6MDstc1mPE3cRWuqdngkwjmHeSrjbJCe1Zbg4382HppmexqrXlqgG
  access-expired-seconds: 360000 # 60 * 60 : 100시간
  refresh-expired-seconds: 8640000 # 60 * 60 * 24 : 10일
cloud:
  aws:
    s3:
```

```
      bucket: b304-bucket
    credentials:
      access-key: AKIAQMHDKHJL5X6LUJPI
      secret-key: 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
    region:
      static: ap-northeast-2
      auto: false
    stack:
      auto: false
```

Jenkins pipeline

```
//자동배포 버전
pipeline {
    agent any

    stages {
        stage('Springboot build') {
            steps {
                dir('backend'){
                    sh '''
                    echo 'springboot build'
                    chmod +x gradlew
                    ./gradlew clean build
                    '''
                }
            }
        }
        stage('Dockerimage build') {
            steps {
                dir('backend'){
                    sh '''
                    echo 'Dockerimage build'
                    docker build -t docker-springboot:0.0.1 .
                    '''
                }
            }
        }
        stage('Deploy') {
            steps {
                dir('backend'){
                    sh '''
                    echo 'Deploy'

                      docker stop springboot
                      docker rm springboot
                      docker run -d -p 8080:8080 --name springboot docker-springboot:0.0.1
                      '''
                }
            }
        }
    }
}
```

Dockerfile

```
FROM openjdk:11-jdk
VOLUME /tmp
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

Nginx setting

서버의 패키지 목록 업데이트

```
sudo apt update
sudo apt upgrade
```

```
sudo apt autoremove
```

```
#nginx 설치
sudo apt install nginx

#nginx 설치 상태 확인
sudo systemctl status nginx
```

```
#nginx 실행 시작/중지
sudo systemctl start nginx
sudo systemctl stop nginx
```

```
#ssl설정

#let's Encrypt 설치
sudo apt-get install letsencrypt

#certbot 설치
sudo apt-get install certbot python3-certbot-nginx


#certbot 동작 (이메일 입력, 약관 동의Y, 이메일 동의 Y or N, 도메인입력)
sudo certbot --nginx


##방화벽 기본 포트 설정
sudo ufw allow ssh
sudo ufw allow http
sudo ufw allow https
```

환경 설정

`sudo vi /etc/nginx/sites-available/nginx.conf`

```
server { #80포트로 받을 때
    listen 80;
    server_name j9b304.p.ssafy.io; # 없을경우 localhost

    location / {
        return 301 https://j9b304.p.ssafy.io$request_uri;
    }
}

server {
    listen 443 ssl http2;
    server_name j9b304.p.ssafy.io;

    # ssl 인증서 적용하기
    ssl_certificate /etc/letsencrypt/live/j9b304.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/j9b304.p.ssafy.io/privkey.pem;

    location /api { # location 이후 특정 url을 처리하는 방법을 정의
        rewrite ^/api(.*)$ $1?$args break;
        proxy_pass http://localhost:8080; # Request에 대해 어디로 리다이렉트하는지
        proxy_redirect off;
        charset utf-8;

        proxy_http_version 1.1;
        proxy_set_header Connection "upgrade";
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-NginX-Proxy true;
    }
}
```

```
#sites-enabled에 심볼릭 링크 생성 (다른 이름으로 만든 경우)
sudo ln -s /etc/nginx/sites-available/themint.conf /etc/nginx/sites-enabled

#conf 파일 오류 없는지 확인 하고 ngninx 실행
sudo nginx -t
```