

포팅메뉴얼_특화PJT (JAV)

버전

BackEnd

- springboot : 2.7.1
- jdk : 1.8.0_342
- maven : 4.0.0
- MySQL : 8.0.30-0ubuntu0.20.04.2
- IntelliJ : 2022.1.3

FrontEnd

- node.js : 16.15.0v 64bit (LST 버전 사용)
- npm : 8.5.5v
- react : 18.2.0v
- react-router-dom : 6.3.0v
- react-redux : 8.0.2v
- redux Toolkit : 1.8.3v
- TypeScript : 4.7.4v
- Sass : 1.54.4v

Server

- AWS EC2
- Ubuntu : 20.04 LTS
- nginx : 1.18.0 (Ubuntu)
- Jenkins : 2.346.2
- Docker

환경 설정

Server 환경 설정

- MySQL 설치

```
# 우분투 서버 업데이트, 업그레이드
$ sudo apt-get update
$ sudo apt-get upgrade

# 설치
$ sudo apt-get install mysql-server

# 구동
$ sudo systemctl start mysql.service

# MySQL 접속
$ sudo mysql

# 현재 mysql에서 기본으로 세팅되어있는 유저 확인
mysql> SELECT user,authentication_string,plugin,host FROM mysql.user;

# DB에 외부 접속하기 위해 새로운 계정 생성
# (% : 어떤 ip로도 접속 가능하도록 하기 위함)
mysql> CREATE USER '계정이름'@'%' IDENTIFIED BY '비밀번호';

# 권한 부여
mysql> GRANT ALL PRIVILEGES ON *.* TO '계정이름'@'%' WITH GRANT OPTION;
mysql> FLUSH PRIVILEGES;

# 외부 접속 허용 설정
$ cd /etc/mysql/mysql.conf.d
$ vi mysqld.cnf

# i 눌러서 입력 모드로 변경 후 bind-address 를 0.0.0.0 으로 수정 후
# 저장하고 나가기

# EC2 인스턴스의 3306 포트 열기
$ sudo ufw allow 3306

# MySQL 재시작
$ sudo systemctl restart mysql.service
```

- java 설치

```
# openjdk 설치
$ sudo apt-get install openjdk-11-jdk

# java 버전 확인
$ java -version,
$ javac -version

## java 환경 변수 설치
# javac 위치 확인
$ which javac
# ==> /usr/lib/jvm/java-8-openjdk-amd64/bin/ 위치가 여기라면
# $JAVA_HOME은 /usr/lib/jvm/java-8-openjdk-amd64으로 설정하도록 한다.

# 환경변수 설정 위해 profile 편집기에 진입
$ sudo vi /etc/profile
```

```
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
export PATH=$JAVA_HOME/bin:$PATH
export CLASS_PATH=$JAVA_HOME/lib:$CLASS_PATH
```

```
# 위의 내용을 하단 추가
# esc -> :wq! -> enter 해서 저장
# 수정 완료 후에 profile을 reload를 위해서 리부팅을 진행한다
$ sudo reboot now
```

빌드 및 배포

Docker 설치

```
## apt를 이용하여 docker를 설치 할 예정이라 apt를 update합니다.
$ sudo apt update

## docker 설치에 필요한 패키지 들을 설치합니다.
$ sudo apt install apt-transport-https ca-certificates curl software-properties-common

## curl를 이용하여, 도커를 설치하기 위한 gp 내용을 다운받고, apt 기능을 위한 리스트에 추가합니다.
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

## ubuntu 버전에 맞는 docker를 다운로드 할 수 있도록 repository 리스트에 추가합니다.
$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"

## apt update를 실행
$ sudo apt update

## docker-ce를 설치합니다
$ apt-cache policy docker-ce
$ sudo apt install docker-ce

## docker가 설치되면, 자동으로 시스템 서비스로서 등록이 됩니다.
## systemctl 명령어를 통해 docker 서비스 상태를 확인해보면, 도커엔진이 구동중인걸 확인할 수 있습니다.
$ sudo systemctl status docker
```

SpringBoot에 Dockerfile 작성

```
FROM openjdk:8-jdk-alpine

# jar 파일 경로는 직접 입력해주세요.
COPY build/libs/JAV-0.0.1-SNAPSHOT.jar app.jar

## 배포용 properties 실행 명령어
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Jenkins 설치 및 초기 세팅

```

## Jenkins 설치를 위해 Repository key 추가
$ wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -

# 서버의 sources.list에 Jenkins 패키지 저장소를 추가
$ echo deb http://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list

# key 등록
$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys FCEF32E745F2C3D5

# 패키지 인덱스 정보 업데이트
$ sudo apt-get update

## Jenkins 패키지 설치
$ sudo apt-get install jenkins

# Jenkins 실행하기
$ sudo systemctl start jenkins

# Jenkins 상태 체크하기
$ sudo systemctl status jenkins

## Jenkins 포트 변경하기
sudo vi /lib/systemd/system/jenkins.service

# HTTP_PORT를 9090 포트로 변경
# esc -> :wq! -> enter 해서 저장
HTTP_PORT=9090

# 서비스 재시작
$ sudo service jenkins restart
# 정상여부 확인
$ sudo systemctl status jenkins

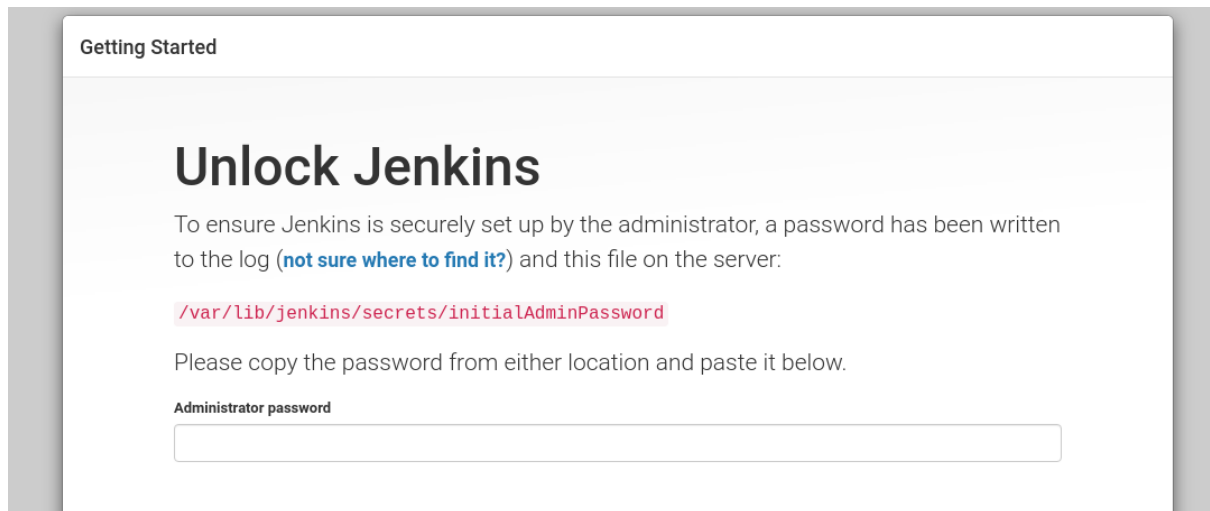
## 초기 비밀번호 확인
sudo cat /var/lib/jenkins/secrets/initialAdminPassword

```

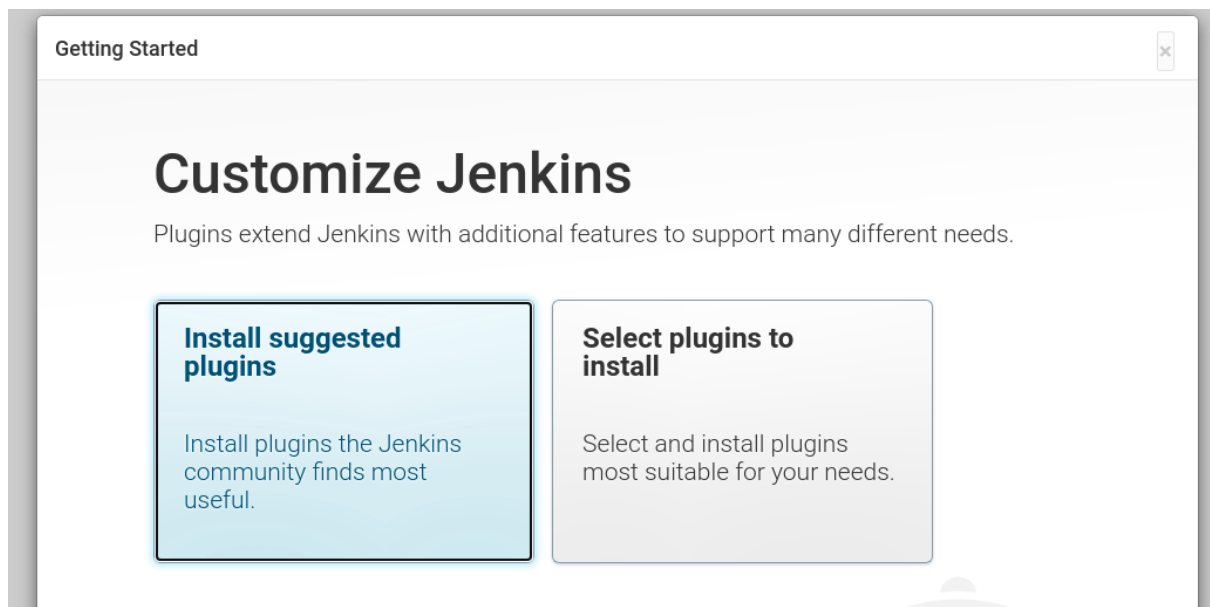
- jenkins 초기 세팅
- <https://yoseph0310.tistory.com/120?category=899321> : 젠킨스 깃랩 연결 참고
- <https://crispyblog.kr/development/common/8> : ci/cd 참고

i7C109.p.ssafy.io:9090 접속 후

password는 위의 나온 값으로 jenkins 로그인 진행함.



- Jenkins 초기 플러그인 설치



- Jenkins 관리자 로그인
id : admin / pw : 위 나온 값.
- GitLab과 연결하기 위한 Plugin 설치
Jenkins 관리 > 플러그인 관리 > 설치가능에서 Publish Over SSH, GitLab plugin 설치.
- AWS와 젠킨스 연결
Jenkins 관리 > 시스템 설정 > Publish over SSH 설정에서 아래 내용을 기입해주고 고급 버튼을 눌러서 RSA 키에 ssh 접속 시 필요한 pem 값 기입한 후 Test Configuration Success 확인

SSH Servers

SSH Server

✕

Name ?

campic.site

Hostname ?


i7C109.p.ssafy.io

Username ?

ubuntu

Remote Directory ?

/home/ubuntu/deploy


 고글...

Success

Test Configuration

☒ Use password authentication, or use a different key ?

Passphrase / Password ?

 Concealed

Path to key ?

Key ?

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA...
```

b

m

```
-----END RSA PRIVATE KEY-----
```

- GitLab과 관련된 설정
Jenkins 관리 > 시스템 설정 > GitLab

GitLab connections

Connection name
A name for the connection

Gitlab host URL
The complete URL to the Gitlab server (e.g. http://gitlab.mydomain.com)

Credentials
API Token for accessing Gitlab

Credentials가 add를 눌러서 GitLab API token을 이용하여 등록한 후 Test Connection을 눌러 Success를 확인

(GitLab API token은 GitLab의 Settings > Access Token에서 발급)

Jenkins 아이템 생성 - BackEnd

- 소스코드 관리

소스 코드 관리

☐ None

☒ Git ?

Repositories ?

Repository URL ?

https://lab.ssafty.com/s07-blockchain-nft-sub2/S07P22C108

Credentials ?

gosmf12@naver.com/*****

+ Add

고급...

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

*/BE

Add Branch

git을 pull을 실행할 repository를 등록합니다. 또한 원하는 branch를 설정할 수 있으니 여기서 'BE'로 설정.

• 빌드유발

빌드 유발

☐ 빌드를 원격으로 유발 (예: 스크립트 사용) ?

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: http://7c108.pssafty.io:9090/project/deploytest ?

Enabled GitLab triggers

☒ Push Events

☐ Push Events in case of branch delete

☒ Opened Merge Request Events

☐ Build only if new commits were pushed to Merge Request ?

☐ Accepted Merge Request Events

☐ Closed Merge Request Events

Rebuild open Merge Requests

Never

☒ Approved Merge Requests (EE-only)

☒ Comments

Comment (regex) for triggering a build ?

Jenkins please retry a build

고급...

레포지토리 설정이 끝났으니 그 레포지토리에서 push가 됐을 때 빌드 유발을 눌러서 어떤 행동을 할 지 설정

Build when a change is pushed to GitLab 선택

고급 버튼을 눌러 Gitlab의 Secret token을 생성하고 자동으로 push 이벤트를 감지할 수 있도록 webhook을 지정할 예정, gitlab의 setting > webhook으로 이동하여 아까 발급받은 secret token을 입력하고 trigger 부분엔 push 이벤트가 감지될 branch를 입력

- 빌드

```
# BuildStep > ExcuteShell 선택 후 Command에 다음 명령 입력

cd backend/JAV
chmod +x gradlew
./gradlew build

docker build --tag backend .

docker images
```

- 빌드 후 조치

```
# 빌드 후 조치 > Send build artifacts over SSH Exec Command에 다음 명령 입력
# Source files에는 적당한 파일 아무거나 입력해도 됨.

if (sudo docker ps | grep "backend") then sudo docker stop backend; fi

sudo docker run -it -d --rm -p 8080:8080 --name backend backend
```

Jenkins 아이템 생성 - FrontEnd

- 소스코드 관리

소스 코드 관리

☐ None

☒ Git ?

Repositories ?

Repository URL ?

https://lab.ssfy.com/s07-blockchain-nft-sub2/507P22C108

Credentials ?

gosmf12@naver.com/*****

+ Add

고급...

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

*/BE

Add Branch

git을 pull을 실행할 repository를 등록합니다. 또한 원하는 branch를 설정할 수 있으니 여기서 'FE'로 설정.

- 빌드유발

빌드 유발

☐ 빌드를 원격으로 유발 (예: 스크립트 사용) ?

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://j7c108.p.ssafy.io:9090/project/deploytest> ?

Enabled GitLab triggers

☒ Push Events

☐ Push Events in case of branch delete

☒ Opened Merge Request Events

☐ Build only if new commits were pushed to Merge Request ?

☐ Accepted Merge Request Events

☐ Closed Merge Request Events

Rebuild open Merge Requests

Never

☒ Approved Merge Requests (EE-only)

☒ Comments

Comment (regex) for triggering a build ?

Jenkins please retry a build

고급...

레포지토리 설정이 끝났으니 그 레포지토리에서 push가 됐을 때 빌드 유발을 눌러서 어떤 행동을 할 지 설정

Build when a change is pushed to GitLab 선택

고급 버튼을 눌러 Gitlab의 Secret token을 생성하고 자동으로 push 이벤트를 감지할 수 있도록 webhook을 지정할 예정, gitlab의 setting > webhook으로 이동하여 아까 발급받은 secret token을 입력하고 trigger 부분엔 push 이벤트가 감지될 branch를 입력

- 빌드

```
# BuildStep > ExcuteShell 선택 후 Command에 다음 명령 입력

cd front
pwd
sudo cp -a /home/ubuntu/app/env/. /var/lib/jenkins/workspace/deploytest_front/front/
sudo cp -a /home/ubuntu/keys/. /var/lib/jenkins/workspace/deploytest_front/front/keys

docker build -t react .
docker images
```

- 빌드 후 조치

```
# 빌드 후 조치 > Send build artifacts over SSH Exec Command에 다음 명령 입력
# Source files에는 적당한 파일 아무거나 입력해도 됨.

if (sudo docker ps | grep "react") then sudo docker stop react; fi

sudo docker run -it -d --rm -p 80:80 -p 443:443 --name react react
docker rmi -f $(docker images -f "dangling=true" -q)
```

let's encrypt SSL로 https 적용

```
# Nginx 실행
$ sudo service nginx start

# snap을 이용하여 core 설치 -> snap을 최신 버전으로 유지하기 위해 설치
$ sudo snap install core

# core를 refresh 해준다.
$ sudo snap refresh core

# 기존에 잘못된 certbot이 설치되어있을 수도 있으니 삭제 해준다.
$ sudo apt remove certbot

# certbot 설치
$ sudo snap install --classic certbot

# certbot 명령을 로컬에서 실행할 수 있도록 snap의 certbot 파일을 로컬의 cerbot과 링크(연결)
$ sudo ln -s /snap/bin/certbot /usr/bin/certbot

# certbot을 이용해 ssl 인증서를 받아온 뒤 cerbot이 스스로 nginx설정을 해주도록한다.
# 아래 명령을 수행하면 여러 질문이 등장한다. 질문에 맞춰 답을 해주면 된다.
# domain을 입력하라는 질문에서는 본인이 사용할 Domain을 입력해주면 된다.
$ sudo certbot --nginx
```

- 빌드 배포 참고

https://github.com/hjs101/CICD_manual

<https://deepmal.tistory.com/21>

프로퍼티 정의

NGINX 설정

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    root /usr/share/nginx/html;

    index index.html index.htm index.nginx-debian.html;
```

```

server_name _;

location / {
    try_files $uri $uri/ =404;
}

}

server {
    root /usr/share/nginx/html;

    index index.html index.htm index.nginx-debian.html;
    server_name j7c108.p.ssafy.io; # managed by Certbot

    location / {

        try_files $uri $uri/ =404;
    }

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/j7c108.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/j7c108.p.ssafy.io/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/live/j7c108.p.ssafy.io/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/live/j7c108.p.ssafy.io/ssl-dhparams.pem; # managed by Certbot
}
server {
    if ($host = j7c108.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80 ;
    listen [::]:80 ;
    server_name j7c108.p.ssafy.io;
    return 404; # managed by Certbot

}

```

BackEnd Properties

```

spring :
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://j7c108.p.ssafy.io:3306/JAV
    username: nft
    password: qltmxkwb!
  mvc:
    pathmatch:
      matching-strategy: ant_path_matcher

  jpa:
    hibernate:
      ddl-auto: create

  server:
    port: 8080

```

```
ssl:
  key-store: classpath:keystore.p12
  key-store-type: PKCS12
  key-store-password: beastazoo
http2:
  enabled: true
```

FrontEnd git ignore

```
# env
REACT_APP_PORT=3001
CHOKIDAR_USEPOLLING=true
SKIP_PREFLIGHT_CHECK=true

REACT_APP_API_KEY = AIzaSyBS03KSsNMBE9vN0oMKmvd917S1I1DG3mw
REACT_APP_AUTH_DOMAIN = beastazoo.firebaseio.com
REACT_APP_PROJECT_ID = beastazoo

REACT_APP_STORAGEBUCKET = beastazoo.appspot.com
REACT_APP_MESSAGING_SENDER_ID = 159067596844
REACT_APP_APP_ID = 1:159067596844:web:31ad2f10c53f8f5a6e8989
REACT_APP_MEASUREMENT_ID = G-KS16QK5KFD

REACT_APP_ENV=production

gitignore에 추가했으니 개인적으로 .env파일은 추가해서 사용해야 함.

front 폴더 아래에 생성하면 된다.
```