까마귀공방

포팅메뉴얼

목차

1. Gitlab 소스코드 클론 이후 빌드 및 배포 과정 정리

A. 개발환경

Frontend

- JavaScript(ES6+)
- Node.js(v16.18.0)
- react 18.2.0
- react-redux 8.0.4
- react-router-dom 6.4.2
- react-scripts 5.0.1
- tailwindcss 3.1.8

Backend

- Java SDK 1.8.0_342
- Spring boot 2.7.5
- MySQL 8.0
- MongoDB

배포

- AWS EC2 Ubuntu 20.04 (http://k7d207.p.ssafy.io)
- AWS EC2 Ubuntu 22.04 (https://까마귀공방.com)
- Docker 4.1.0
- Jenkins
- Nginx 1.18.0 (Ubuntu)

IDE

- VSCode
- Intellij IDEA 2022.2.3

B. 배포 시 특이사항

Nginx

/frontend/nginx/default.conf

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

  server_name 까마귀공방.com;
  proxy_set_header X-Real-IP $remote_addr;
  proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
  proxy_set_header Host $host;

  location / {
    return 301 https://$server_name$request_uri;
  }
}

server {
    if ($host = xn--ob0b1oqnl39ac1c.com){
        return 301 https://$host$request_uri;
    }
    listen 80;
    listen [::]:80;
    server_name xn-ob0b1oqnl39ac1c.com;
    return 404;
}
```

```
# HTTPS 설정
server {
    listen 443 ssl;
    listen [::]:443 ssl;

    server_name xn-ob0b1oqnl39ac1c.com;

    index index.html;

    ssl_certificate /etc/letsencrypt/live/xn--ob0b1oqnl39ac1c.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/xn--ob0b1oqnl39ac1c.com/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    location / {
    root /usr/share/nginx/html;
    try_files $uri $uri/ /index.html;

  }
# /api로 시작하면 백으로
  location /api {
    proxy_pass http://k7d207.p.ssafy.io:8080;
  }
  # 파일 접근할 URI 정해서 넣기
  location /files {
        root /home/ubuntu/crow_data/;
    }
}
```

C. 빌드 & 배포

0. Docker 설치

```
sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository ₩
"deb [arch=amd64] https://download.docker.com/linux/ubuntu ₩
$(lsb_release -cs) ₩
stable"

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

I. Frontend

frontend/Dockerfile

```
FROM node:lts-alpine as build-stage
WORKDIR /usr/src/app
COPY package*.json ./
# 빌드
RUN npm install --force

COPY . /usr/src/app
RUN npm run build

FROM nginx:stable-alpine as production-stage
# nginx 기본 설정 변경
RUN rm /etc/nginx/conf.d/default.conf
COPY ./nginx/default.conf /etc/nginx/conf.d/

RUN rm -rf /usr/share/nginx/html/*
COPY --from=build-stage /usr/src/app/build /usr/share/nginx/html

EXPOSE 443 80
CMD ["nginx", "-g", "daemon off;"]
```

Docker image 생성 및 배포

```
git clone https://lab.ssafy.com/s07-final/S07P31D207.git
cd /home/ubuntu/S07P31D207/front
docker build –t front .
docker run -d --name front -v /etc/letsencrypt:/etc/letsencrypt -p 80:80 -p 443:443
front
```

II. Backend

application.properties

1. DB 접속 설정

```
spring.jpa.database=sql_server
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://k7d207.p.ssafy.io:3999/project3?characte
rEncoding=UTF-8&serverTimezone=UTC
spring.datasource.username=PJT3D207
spring.datasource.password=d207!crow

spring.data.mongodb.uri= mongodb://k7d207.p.ssafy.io:3998
spring.data.mongodb.database = mongoDB
spring.data.mongodb.username= PJT3D207
spring.data.mongodb.password = d207crow
```

2. JWT secret key 설정

```
secret.jwt.key="team_goldenCrow"
```

3. spring boot 카멜케이스(camelCase) 스네이크 케이스(snake_case)로 자동 변환 금지 설정

```
spring.jpa.hibernate.naming.implicit-
strategy=org.hibernate.boot.model.naming.ImplicitNamingStrategyLegacyJpaImp
l
spring.jpa.hibernate.naming.physical-
strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
```

4. spring.mvc.pathmatch.matching-strategy 기본값을 path_pattern_parser 에서 ant_path_matcher 로 변경 (springboot 버전 2.6 이상의 경우 필요)

```
spring.mvc.pathmatch.matching-strategy=ant_path_matcher
```

goldenCrow/JenkinsFile

```
pipeline {
        agent none
        stages {
                stage('Docker build') {
                        agent any
                        steps {
                                sh 'docker build -t goldencrow ./goldenCrow'
                        }
                }
                stage('Docker run') {
                        agent any
                        steps {
                                sh 'docker ps -f name=goldencrow -q ₩
                                        | xargs --no-run-if-empty docker container
stop'

                                sh 'docker container ls -a -f name=goldencrow -q
₩
                                        | xargs -r docker container rm'

                                sh 'docker run -d --name goldencrow -u root --
privileged ₩
                                    -p 8080:8080 ₩
                                    -v
/home/ubuntu/crow_data:/home/ubuntu/crow_data ₩
                                    -v /var/run/docker.sock:/var/run/docker.sock ₩
                                    -v /usr/bin:/usr/bin ₩
                                    -v /usr/lib:/usr/lib ₩
                                    goldencrow'
                        }
```

goldenCrow/Dockerfile

```
FROM openjdk:8-jdk AS builder
COPY gradlew .
COPY gradle gradle
COPY build.gradle .
COPY settings.gradle .
COPY src src
RUN chmod +x ./gradlew
RUN ./gradlew bootjar

FROM openjdk:8-jdk
COPY --from=builder build/libs/*.jar app.jar

ENTRYPOINT ["java","-jar","/app.jar"]
```

II-a. Jenkins 설치

1. Docker에 Jenkins 설치

```
sudo docker run -d --name jenkins -u root --privileged ₩
-p '9090:8080' ₩
-v '/home/ubuntu/docker-volume/jenkins:/var/jenkins_home' ₩
-v '/var/run/docker.sock:/var/run/docker.sock' ₩
-v '/usr/bin/docker:/usr/bin/docker' ₩
jenkins/jenkins
```

2. 브라우저에서 '서버주소:9090'으로 접속

```
http://k7d207.p.ssafy.io:9090/
```

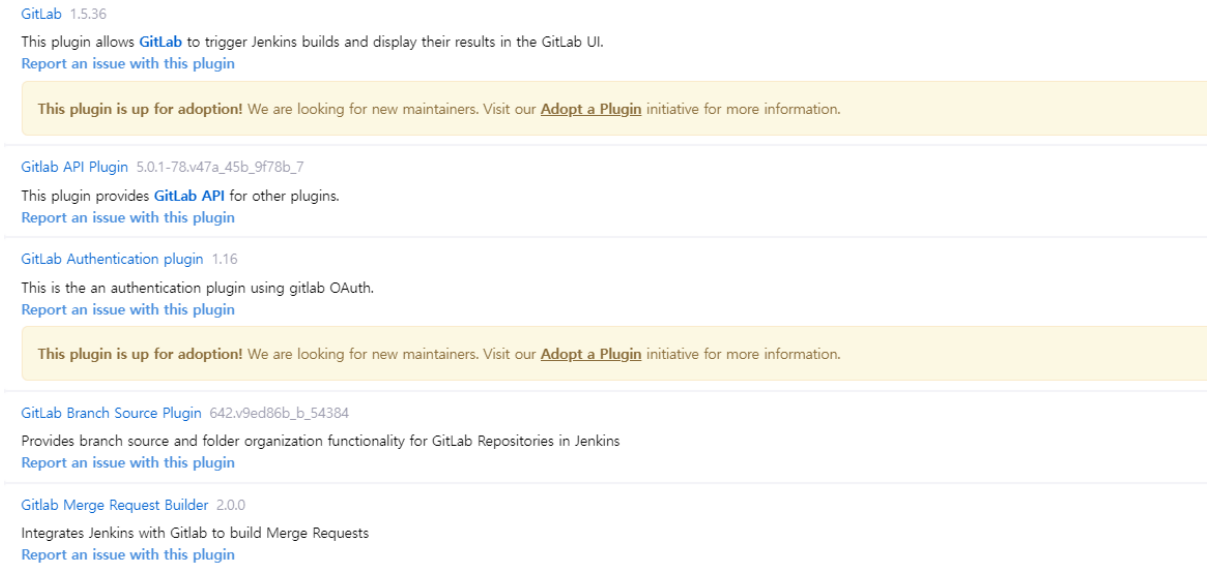
3. Password 확인해서 로그인

```
sudo docker logs jenkins
```

4. Suggest install & Create First Admin User

5. Jenkins 관리 -> Plugin 관리 -> Plugin 설치

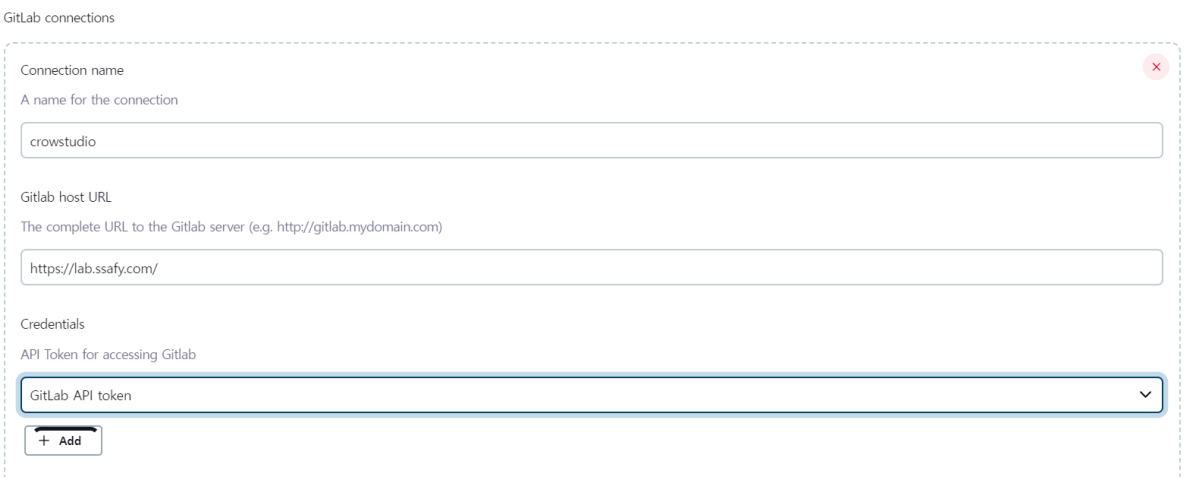
- Gitlab 검색
- 아래 plugin 설치



6. Web hook 걸기

- Gitlab 해당 프로젝트 접속
- Settings - Access Tokens에서 access token 발급받기

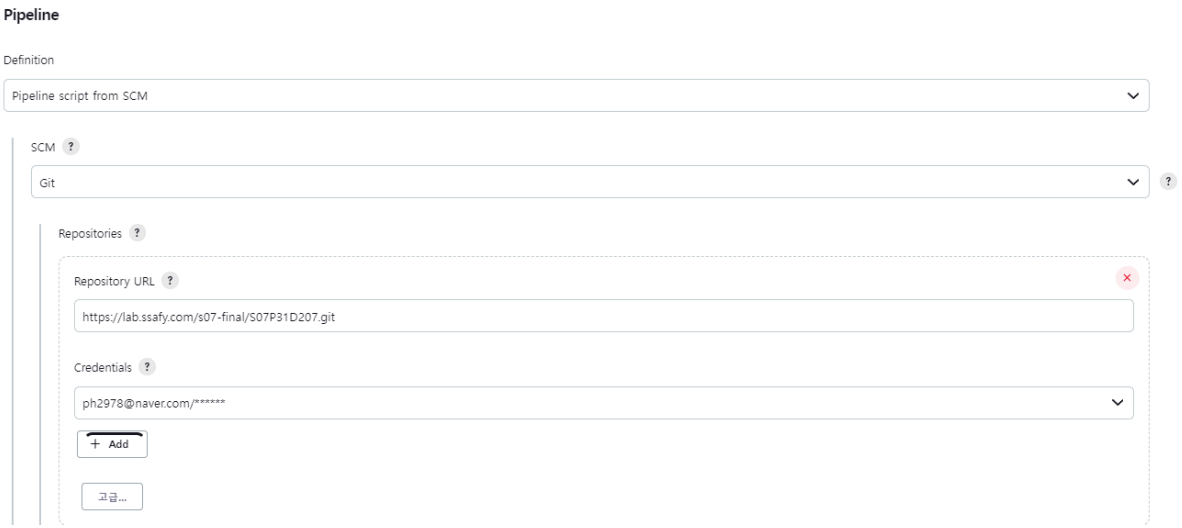
  (token 저장해두기)
- Jenkins 관리 -> 시스템 설정 -> Gitlab



- Credentials +Add 클릭
    1. Kind : GitLab API Token
    2. API token에 gitlab에서 발급받은 access-token 입력
    3. Id, description은 입력 X
    4. Add 버튼 눌러서 credential에 등록
    5. Test connection해서 연결되는지 확인
- 저장 클릭

7. Pipline 만들기

- 새 item – pipline 선택 – item명 입력
- Pipeline - Definition : pipline script from SCM으로 변경
- repository URL : 프로젝트 URL (https://lab.ssafy.com/s07-final/S07P31D207.git)
- credential
    1. Kind : username with password
    2. Username : gitlab 아이디 입력
    3. Password : access-token 입력



- Branches to build : 적용할 브랜치 입력 (origin/dev-back)
- Script path : 프로젝트 내 JenkinsFile 위치 (goldenCrow/JenkinsFile)
- 저장 클릭

8. Build trigger 설정

- Jenkins
  1. Item 선택 – 구성 – Build Triggers
  2. Build when a change is pushed to GitLab 선택 – 고급 버튼 클릭
  3. Secret token generate
  4. 저장
- Gitlab
  1. Settings – webhooks
  2. URL : build when .. 에 적힌 url 입력
  3. Secret Token에 jenkins에서 발급받은 토큰 입력
  4. 원하는 trigger 선택 후 Add trigger

III. DB

MySQL

1. MySQL 설치

```
sudo apt-get update
sudo apt-get upgrade
sudo apt install mysql-client-core-8.0
```

2. MySQL Docker image 다운

```
docker pull mysql
```

3. MySQL Docker container 생성 & 실행 (port 3999)

```
docker run --name mysql-container -e MYSQL_ROOT_PASSWORD=<password> -d
-p 3999:3306 mysql:latest
```

MongoDB

1. MongoDB Docker image 다운

```
docker pull mongo
```

2. MySQL Docker Docker container 생성 & 실행 (port 3998)

```
docker run --name mongo-container -d -p 3998:27017 mongo
```

2. 프로젝트에서 사용하는 외부 서비스 정보

A. Papago API

변수명 추천 API에서 한국어를 영어로 변환하는 과정에서 활용

B. Black

포맷팅 API에서 활용

C. Pylint

린트 API에서 활용