



포팅 메뉴얼

목차

목차

1. 사용 도구

2. 개발 도구

3. 개발 환경

Frontend

Backend

Server

4. 환경변수

Frontend

Backend

5. CI/CD 구축

도커

Nginx

Jenkins

1. 사용 도구

- 이슈 관리 : Jira
- 형상 관리 : GitLab
- 커뮤니케이션 : Notion, MatterMost
- 디자인 : Figma
- CI/CD : Jenkins

2. 개발 도구

- Visual Studio Code

- IntelliJ : 2022.3.2 (Ultimate Edition)
- Terminus

3. 개발 환경

Frontend

이름	버전
React	

Backend

이름	버전
Spring Boot	3.3.3
Java	Azul Zulu 17.0.12
Gradle	8.10.1
Python	3.9.19
fastAPI	0.115.0
numpy	1.26.4
pandas	2.2.3
opencv-python	4.10.0.84
pillow	10.4.0
torch	2.4.1
torchvision	0.19.1
uvicorn	0.31.0

Server

이름	버전
Ubuntu	20.04
Docker	27.2.1

이름	버전
nginx	1.27.1
jenkins	2.462.3
mysql	5.7

4. 환경변수

Frontend

- .env

```
REACT_APP_KAKAO_API_KEY = "";
```

Backend

- application.properties

```
spring.application.name=dogbogam-server

spring.config.import=application-secret.properties

# JPA
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true

# DataSource
spring.datasource.url=jdbc:mysql://${DB_HOST}:3306/dogbogam
spring.datasource.username=${DB_USERNAME}
spring.datasource.password=${DB_PASSWORD}
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

#JWT KEY
jwt.key=${JWT_KEY}

# AWS S3
```

```
aws.s3.access-key=${S3_ACCESS_KEY}
aws.s3.secret-key=${S3_SECRET_KEY}
aws.s3.bucket=${S3_BUCKET}
cloud.aws.region.static=us-east-2

# Max size of single file (20MB)
spring.servlet.multipart.max-file-size=20MB
# Max size of entire request (25MB)
spring.servlet.multipart.max-request-size=25MB

# feign client
spring.cloud.openfeign.enabled=true

gpu.server.uri=${GPU_SERVER_URI}
```

- application-secret.properties

```
# Database
DB_HOST=j11b101.p.ssafy.io
DB_USERNAME=
DB_PASSWORD=

# JWT Key
JWT_KEY=

# AWS S3 Key
S3_ACCESS_KEY
S3_SECRET_KEY=
S3_BUCKET=

# GPU server
GPU_SERVER_URI=http://xxx.xxx.xxx.xxx:xxxx/api/ai-dignosis
```

5. CI/CD 구축

도커

1. GPG Key

```
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/g
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

2. Apt 소스에 저장소 추가

```
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/a
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable"
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
  sudo apt-get update
```

3. apt-get을 이용한 docker 라이브러리 설치

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

4. docker-compose.yml

```
version: "3"
services:
  nginx:
    container_name: nginx
    image: nginx
    networks:
      - dog
    restart: always
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx.conf:/etc/nginx/conf.d/default.conf # NGI
      - /etc/letsencrypt:/etc/letsencrypt # Let's Encrypt
      - /var/lib/letsencrypt:/var/lib/letsencrypt
      - ./certbot/www:/var/www/html # Certbot을 위한 웹 루트
```

```
jenkins:
  container_name: jenkins
  image: jenkins/jenkins:lts
  user: root
  networks:
    - dog
  restart: always
  ports:
    - "8081:8080"
    - "50000:50000"
  volumes:
    - ./jenkins-data:/var/jenkins_home
    - /var/run/docker.sock:/var/run/docker.sock
    - /usr/bin/docker:/usr/bin/docker
  environment:
    - JAVA_OPTS=-Djenkins.install.runSetupWizard=false
```

```
mysql:
  container_name: mysql
  image: mysql:5.7
  restart: always
  ports:
    - "3306:3306"
  environment:
    MYSQL_ROOT_PASSWORD: 접속 비밀번호
    MYSQL_DATABASE: dogbogam
  networks:
    - dog
  volumes:
    - ./mysql-data:/var/lib/mysql
```

```
networks:
  dog:
    driver: bridge
```

```
volumes:
```

```
jenkins-data:
mysql-data:
```

Nginx

1. nginx.conf 파일 작성

```
server {
    listen 80;
    server_name j11b101.p.ssafy.io;

    # HTTP에서 HTTPS로 리다이렉트
    location / {
        add_header 'Cross-Origin-Embedder-Policy'
        add_header 'Cross-Origin-Opener-Policy' 's

        return 301 https://$host$request_uri;
    }

    # Certbot 인증을 위한 경로
    location /.well-known/acme-challenge/ {
        root /var/www/html;
        allow all;
    }
}

server {
    listen 443 ssl;
    server_name j11b101.p.ssafy.io;

    # SSL 인증서 경로 설정
    ssl_certificate      /etc/letsencrypt/live/j11b101.
    ssl_certificate_key  /etc/letsencrypt/live/j11b101.

    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers HIGH:!aNULL:!MD5;

    # dogbogam_front와 연결 (포트 8082)
```

```

        location / {
            proxy_pass http://dogbogam_front:8082/;

            add_header 'Cross-Origin-Embedder-Policy'
            add_header 'Cross-Origin-Opener-Policy' 's

            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_ad
            proxy_set_header X-Forwarded-Proto $scheme
        }

        location /api/ {
            proxy_pass http://dogbogam:8080/;

            add_header 'Cross-Origin-Embedder-Policy'
            add_header 'Cross-Origin-Opener-Policy' 's

            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_ad
            proxy_set_header X-Forwarded-Proto $scheme
        }

        location /jenkins/ {
            proxy_pass http://jenkins:8080/; # Jenkin
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_ad
            proxy_set_header X-Forwarded-Proto $scheme
        }
    }
}

```

2. nginx 컨테이너 재실행

```
sudo docker-compose restart nginx
```


Jenkins

추후 docker-compose로 묶어서 다른 컨테이너들과 함께 구축 및 실행

1. jenkins가 사용할 포트 열기

```
sudo ufw allow *8081*/tcp
sudo ufw reload
sudo ufw status
```

2. jenkins 환경 설정

```
cd ~ && mkdir jenkins-data

cd jenkins-data

mkdir update-center-rootCAs
wget https://cdn.jsdelivr.net/gh/lework/jenkins-update-center-rootCAs
sudo sed -i 's#https://updates.jenkins.io/update-center.js
```

3. config 보안 설정 확인

- 아래의 태그들이 true가 되어 있는지 확인

```
vi /home/ubuntu/jenkins-data/config.xml

<useSecurity>true</useSecurity>
...(중략)
<securityRealm class="hudson.security.HudsonPrivateSecurityRealm">
  <disableSignup>true</disableSignup>
```

4. 초기 실행

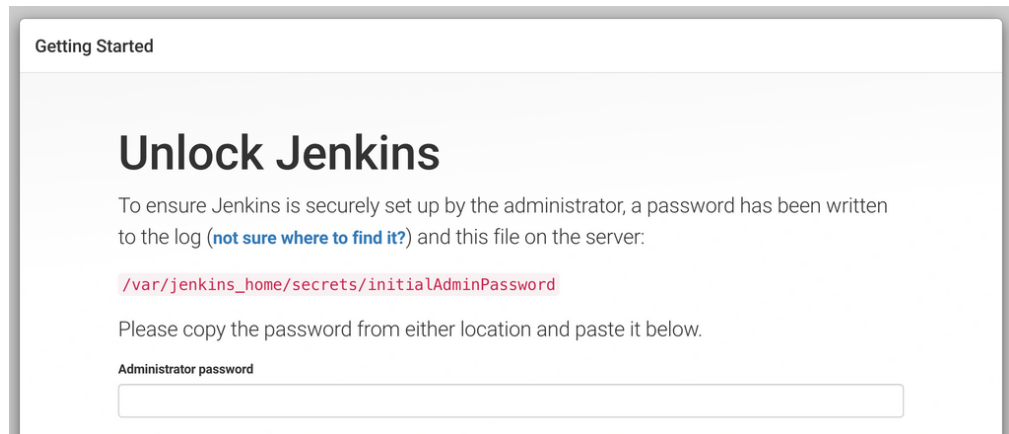
a. jenkins 컨테이너를 실행 후 log 확인

```
sudo docker-compose restart -d jenkins

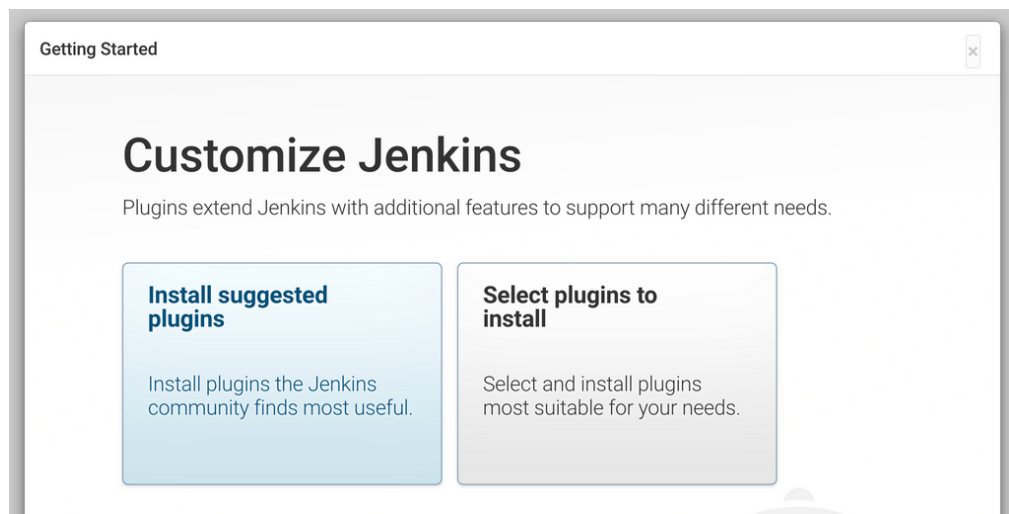
docker logs jenkins
```

- log에는 처음 로그인에 필요한 비밀번호가 출력

- b. 인터넷 익스플로어에 **도메인: 젠킨스포트번호** 로 접속하여 위에서 확인한 비밀번호를 입력하여 로그인



- c. install suggested plugins를 선택하여 기본 설정



- 이 때 최신 jenkins 버전으로 하지 않았다면, 몇몇 플러그인은 설치 안됨

5. 사용한 Jenkins 플러그인

이름	버전
Docker API	3.3.6
Generic Webhook Trigger	2.2.2
Git	5.5.1
GitLab	1.8.1
Gradle	2.13
Oracle Java SE	

이름	버전
Development Kit Installer	
Pipeline	


6. 새로운 Pipeline 구축

a. New Item 클릭


New Item

Enter an item name


Select an item type




Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.



Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.



Multi-configuration project
다양한 환경에서의 테스트, 플랫폼 특성 빌드, 기타 등등 처럼 다수의 서로다른 환경설정이 필요한 프로젝트에 적합함.



Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

OK **Cancel**

- 파이프라인 선택 후, item명 작성후 Ok버튼 클릭

b. Pipeline script 작성

```
pipeline {
    agent any

    stages {
        stage('clone') {
            steps {
                echo 'Start cloning dogbogam...'
                git branch: 'develop-BE', credentialsId
                echo 'Clone finished!'
            }
        }
    }
}
```

```

stage('properties') {
    steps {
        script {
            withCredentials([file(credentialsId
                sh 'cp $secret ./Back-End/dogbo
            }
        }
    }
}

stage('build') {
    steps {
        echo 'Start building dogbogam...'
        sh 'ls'
        // Gradle 빌드 후 JAR 파일을 생성
        sh 'cd ./Back-End/dogbogam-server && ch

        // JAR 파일을 빌드 컨텍스트로 복사
        sh 'cp ./Back-End/dogbogam-server/build
        sh 'ls -l ./dogbogam.jar' // 복사된 파일
        echo 'Build finished!'
    }
}

stage('deploy') {
    steps {
        echo 'Start deploying...'
        sh 'docker stop dogbogam || true'
        sh 'docker rm dogbogam || true'
        sh 'docker rmi dogbogam || true'

        sh 'ls -al'

        // Dockerfile 작성 및 빌드
        writeFile file: 'Dockerfile', text: """
            FROM openjdk:17
            COPY ./dogbogam.jar /app/dogbogam.j
    
```

```
        WORKDIR /app
        EXPOSE 8080
        ENTRYPOINT ["java", "-jar", "-Duser.timezone=UTC", "-Dlogback.configurationFile=logback-spring.xml", "-jar", "target/docker-build.jar"]
    }

    // Docker 빌드
    sh 'docker build -t dogbogam:latest .'
    // Docker 컨테이너 실행
    sh 'docker run --name dogbogam -d -p 8080:8080 dogbogam:latest'
    echo 'Deploy finished!'
}

}
```