

포팅 매뉴얼 및 외부 서비스 정보

1. 프로젝트 기술 스택



공통



FrontEnd



BackEnd

2. Frontend

[패키지 설치 및 실행](#)

[빌드 및 배포](#)

[Dockerfile을 통해 진행](#)

3. Backend

[빌드 및 배포](#)

[Dockerfile을 통해 진행](#)

4. AWS EC2

[Docker](#)

[Jenkins](#)

5. Jenkins

[GitLab Plugin 설치 및 연동](#)

[자동 빌드 및 배포 설정](#)

[\[BackEnd\] Jenkins Pipeline script](#)

[\[FrontEnd\] Jenkins Pipeline script](#)

6. Docker

[Frontend](#)

[Dockerfile.nginx](#)

[nginx.conf](#)

[Backend](#)

[Dockerfile.spring](#)

[MySQL](#)

7. Nginx

[/nginx/sites-enabled/sitename.conf](#)

8. 외부 서비스 정보

[Kakao OAuth2.0 Login](#)

[주가 시세 Crawling](#)

[경제 관련 News Crawling](#)

[AI 퀴즈 생성기](#)

1. 프로젝트 기술 스택



공통

상세	내용
GitLab	형상 관리
Jira	일정 및 이슈 관리
Mattermost	커뮤니케이션
Notion	일정 및 문서 관리
IntelliJ	IDE
Visual Studio Code	IDE



FrontEnd

상세	버전
Node.js	20.8.0
yarn	1.22.19
react	18.2.0
zustand	4.4.2
firebase	10.4.0

BackEnd

상세	버전
JDK (Zulu)	17.0.8
SpringBoot	2.7.16
MySQL	8.0.34
Ubuntu	20.04 LTS
Nginx	1.22.1
Docker	24.0.6
Jenkins	2.414.1
redis	7.2.1

2. Frontend

패키지 설치 및 실행

```
yarn install
yarn start
```

빌드 및 배포

Dockerfile을 통해 진행

3. Backend

빌드 및 배포

Dockerfile을 통해 진행

4. AWS EC2

Docker

```
sudo apt update
sudo apt upgrade
sudo apt install docker-ce
```

Jenkins

- Docker에 Jenkins 설치 및 구동

```
docker run -d -p 5000:5000 -v /var/jenkins:/var/jenkins_home
-v /var/run/docker.sock:/var/run/docker.sock --name jenkins-container jenkins/jenkins:lts
```

5. Jenkins

GitLab PlugIn 설치 및 연동

자동 빌드 및 배포 설정

[BackEnd] Jenkins Pipeline script

```

pipeline {
    agent any
    tools {
        gradle 'Gradle'
    }
    stages {
        stage('gitlab clone') {
            steps {
                git branch: 'develop',
                    credentialsId: 'tenten',
                    url: 'https://lab.ssafy.com/s09-fintech-finance-sub2/S09P22A510.git'
            }
        }

        stage('Check Java and Gradle') {
            steps {
                sh 'java -version'
                sh 'gradle -v'
            }
        }

        stage('Build Spring Boot App') {
            steps {
                dir('backend/tenten/'){
                    sh 'gradle clean build'
                }
            }
        }

        stage('Build and Run Spring Boot Container') {
            steps {
                dir('backend/tenten/'){
                    script {
                        def volumeMappings = [
                            "/home/ubuntu/images/vote:/app/vote",
                            "/home/ubuntu/images/member:/app/member"
                        ].join(' -v ')

                        // Remove prev image
                        sh 'docker rmi $(docker images -f "dangling=true" -q) || true'

                        // Build the Docker image
                        sh 'docker build -t spring-boot-app -f Dockerfile.spring .'

                        // Stop and remove old container if it exists
                        sh 'docker rm -f my-spring-boot-container || true'

                        // Run a new container from the new image
                        sh "docker run -d -p 8000:8000 -v ${volumeMappings} --name my-spring-boot-container spring-boot-app"
                    }
                }
            }
        }
    }
    post {
        success {
            script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
                mattermostSend (color: 'good',
                    message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Details>)"
                )
            }
        }
        failure {
            script {
                def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
                def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
                mattermostSend (color: 'danger',
                    message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Details>)"
                )
            }
        }
    }
}

```

[FrontEnd] Jenkins Pipeline script

```

pipeline {
  agent any
  tools {
    nodejs 'nodejs'
  }
  stages {
    stage('gitlab clone') {
      steps {
        git branch: 'develop_FE',
           credentialsId: 'tenten',
           url: 'https://lab.ssafy.com/s09-fintech-finance-sub2/S09P22A510.git'
      }
    }

    stage('Check npm') {
      steps {
        sh 'echo $PATH'
        sh 'node -v'
      }
    }

    stage('Build React App') {
      steps {
        dir('frontend/mozey/'){
          sh 'npm cache clean --force'
          sh 'yarn install'
          sh 'CI=false yarn build'
        }
      }
    }

    stage('Build and Run Nginx Container') {
      steps {
        dir('frontend/mozey/'){
          script {
            // Remove prev images
            sh 'docker rmi $(docker images -f "dangling=true" -q) || true'

            // Build the Docker image
            sh 'docker build -t my-nginx -f Dockerfile.nginx .'

            // Stop and remove old container if it exists
            sh 'docker rm -f my-react-container || true'

            // Run a new container from the new image
            sh 'docker run -d -p 5000:80 --name my-react-container my-nginx'
          }
        }
      }
    }
  }
  post {
    success {
      script {
        def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
        def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
        mattermostSend (color: 'good',
            message: "빌드 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Details>)"
        )
      }
    }
    failure {
      script {
        def Author_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
        def Author_Name = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
        mattermostSend (color: 'danger',
            message: "빌드 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER} by ${Author_ID}(${Author_Name})\n(<${env.BUILD_URL}|Details>)"
        )
      }
    }
  }
}

```

6. Docker

Frontend

Dockerfile.nginx

```
# 사용할 기본 이미지를 지정 -> Nginx 이미지
FROM nginx

# Nginx 설정 파일을 복사
COPY nginx.conf /etc/nginx/conf.d/default.conf

# React 애플리케이션 빌드 결과물을 Nginx의 정적 파일 디렉토리로 복사
COPY build/ /usr/share/nginx/html

# Nginx 컨테이너가 80번 포트를 사용하도록 설정
EXPOSE 80

# 컨테이너가 실행될 때 Nginx를 시작
CMD ["nginx", "-g", "daemon off;"]
```

nginx.conf

```
server {
    listen 80;
    server_name _;
    root /usr/share/nginx/html;
    index index.html;

    location / {
        try_files $uri $uri/ /index.html;
    }
}
```

Backend

Dockerfile.spring

```
# Use an official OpenJDK runtime as a parent image
FROM openjdk:11

# Set the working directory inside the container
WORKDIR /app

# Copy the JAR file into the image
COPY ./build/libs/*SNAPSHOT.jar my-spring-boot-app.jar

# Set the command to run your Spring Boot application
ENTRYPOINT ["java", "-jar", "my-spring-boot-app.jar"]

# Expose port 8080
EXPOSE 8080
```

MySQL

```
docker run -d -p 3306:3306 --name mysql-container -e MYSQL_ROOT_PASSWORD={password} mysql:8.0.34
```

7. Nginx

/nginx/sites-enabled/sitename.conf

```
server {
    listen 80;
    server_name j9a510.p.ssafy.io;
    location / {
        rewrite ^ https://j9a510.p.ssafy.io$request_uri;
        return 308;
    }
    location /.well-known/acme-challenge {
        root /var/lib/letsencrypt;
    }
}
```

```

upstream springserver {
    server j9a510.p.ssafy.io:8000;
    #server my-spring-boot-container:8000;
}

upstream reactserver {
    server j9a510.p.ssafy.io:5000;
}

server {
    listen 443 ssl;
    #listen [::]:443 ssl;

    server_name j9a510.p.ssafy.io;

    location /oauth2 {
        proxy_pass http://reactserver;
        proxy_set_header Host $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;
    }

    location /api {
        add_header 'Access-Control-Allow-Origin' '*' always;
        add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS';
        add_header 'Access-Control-Allow-Headers' 'Content-Type ,Authorization, Bearer';
        # Proxy 설정 추가
        proxy_set_header Host $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_pass http://springserver;
    }

    location / {
        try_files $uri $uri/ @react;
        #proxy_pass http://reactserver;
        #proxy_set_header Host $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_pass http://reactserver;
    }
    location @react {
        proxy_pass http://reactserver;
        proxy_set_header Host $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;
    }

    ssl_certificate /etc/letsencrypt/live/j9a510.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/j9a510.p.ssafy.io/privkey.pem;
}

```

8. 외부 서비스 정보

Kakao OAuth2.0 Login

- Kakao 소셜 로그인 및 회원가입 서비스
- <https://developers.kakao.com/>

주가 시세 Crawling

- 실시간으로 변화하는 주가의 시세 정보를 얻기 위해 해당 웹페이지에서 크롤링 진행
- https://finance.naver.com/sise/sise_index.naver?code=KOSPI
- <https://finance.naver.com/world/sise.naver?symbol=SPI@SPX>

경제 관련 News Crawling

- 경제 관련 뉴스 기사를 받아오기 위해 Crawling 진행

- <https://kr.investing.com/news/most-popular-news>
- <https://kr.investing.com/news/stock-market-news>

AI 퀴즈 생성기

- Crawling한 뉴스를 바탕으로 퀴즈를 생성해주는 생성형 AI 사용
- <https://api.opexams.com/questions-generator>