포팅 메뉴얼

사용 도구

- 이슈 관리 : Jira
- 형상 관리 : GitLab
- 커뮤니케이션 : Notion, MatterMost
- 디자인 : Figma
- 빌드 도구 : Jenkins

개발 도구

- Visual Studio Code : ver 1.90.2
- IntelliJ IDEA Ultimate : 2024.1.4
- Pycharm : 2024.1.6
- DataGrip : 2024.1.4

외부 서비스

- Naver OAuth

개발 환경

Frontend

Node.js	20.15.0
React	18.3.1 (버전18)
Vite	5.4.1
PWA	0.20.5

Backend

Java	openjdk 17.0.12 2024-07-16
Spring Boot	3.2.7
Python	3.10
Python library	requirements.txt 참조
Redis	7.4.0
MySQL	Ver 9.0.1 for Linux on x86_64

Server

AWS S3
AWS EC2

Infra

Docker	27.1.1
Ubuntu	20.04.6 LTS
Jenkins	2.452.3

환경 변수

.env (중요 정보 생략)

```
## COMMON
JWT_SECRET_KEY=hbfdbnrtsnbdfgntreyteryertyertrteyertyasdasdasfdgddfndfgnrgdfndfgnrtntrnrgfdhrdfgjrdjhdbnfsnrdthrtsgdfshfvbf
JWT_SALT=asdfasdasfsdfsafgdfghfgjcertytyreyerertyertyrtydfasdfdzsdsdtytrfuughkijbpliupibouyihiasddassdasfdfxgfdghdfgfdgfdhfgchfgchfgchfghfggdfhfghfgjghjhgjgheyertytertyytre
JWT_ISSUER=fooding

AWS_S3_ACCESS_KEY=
AWS_S3_SECRET_KEY=
AWS_S3_BUCKET_REGION=
AWS_S3_BUCKET_NAME=

NAVER_SECRET_KEY=
NAVER_LOGIN_URL=

## Localhost
LOCAL_MYSQL_URL=jdbc:mysql://localhost:3306/fooding?serverTimezone=UTC&useUnicode=yes&characterEncoding=UTF-8
LOCAL_MYSQL_USER=root
LOCAL_MYSQL_PASSWORD=11111111

JWT_ACCESS_TOKEN_EXPIRETIME=720000000
JWT_REFRESH_TOKEN_EXPIRETIME=1440000000

## DEV
DEV_MYSQL_URL=jdbc:mysql://mysql:3306/fooding?serverTimezone=UTC&useUnicode=yes&characterEncoding=UTF-8
DEV_MYSQL_USER=fooding_admin
DEV_MYSQL_PASSWORD=fooding1234

DEV_JWT_ACCESS_TOKEN_EXPIRETIME=720000000
DEV_JWT_REFRESH_TOKEN_EXPIRETIME=1440000000

## FCM
FCM_TYPE=service_account
FCM_PROJECT_ID=
FCM_PRIVATE_KEY_ID=
FCM_PRIVATE_KEY=
FCM_CLIENT_EMAIL=
FCM_CLIENT_ID=
FCM_AUTH_URI=
FCM_TOKEN_URI=
FCM_AUTH_PROVIDER_CERT_URL=
FCM_CLIENT_CERT_URL=
FCM_UNIVERSE_DOMAIN=
```

CI/CD

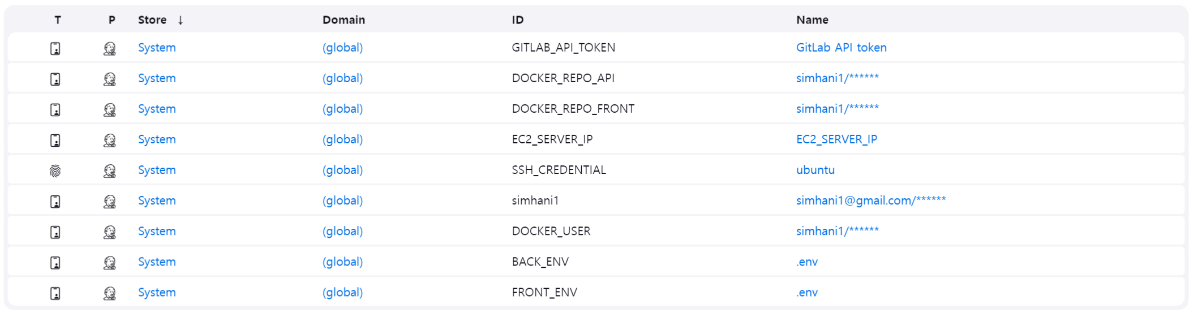
jenkins

기본 plugin 외에 추가 설치

- SSH Agent Plugin
- Docker plugin

credentials 설정

Credentials



T	P	Store ↓	Domain	ID	Name
		System	(global)	GITLAB_API_TOKEN	GitLab API token
		System	(global)	DOCKER_REPO_API	simhani1/******
		System	(global)	DOCKER_REPO_FRONT	simhani1/******
		System	(global)	EC2_SERVER_IP	EC2_SERVER_IP
		System	(global)	SSH_CREDENTIAL	ubuntu
		System	(global)	simhani1	simhani1@gmail.com/******
		System	(global)	DOCKER_USER	simhani1/******
		System	(global)	BACK_ENV	.env
		System	(global)	FRONT_ENV	.env

- GitLab Token 등록
- Docker hub 로그인 정보 등록
- Docker image push를 위한 repo 정보 등록
- SSH 접속을 위해 EC2 IP 정보와 .pem키 정보 등록
- .env 파일 등록

backend pipeline

```
pipeline {
    agent any
    environment {
        ENV_FILE = credentials('BACK_ENV')
    }
    stages {
        stage('Git Clone') {
            steps {
                git branch: 'dev-back', credentialsId: 'simhani1', url: 'https://lab.ssafy.com/s11-bigdata-dist-sub1/S11P21A608.git'
            }
            post {
                failure {
                    echo 'Repository clone failure !'
                }
                success {
                    echo 'Repository clone success !'
                }
            }
        }
        stage('Prepare .env File') {
            steps {
                script {
                    writeFile file: './backend/api/.env', text: "${ENV_FILE}"
                    sh 'cat ./backend/api/.env'
                }
            }
        }
        stage('Build API Project') {
            steps {
                script {
                    // 프로젝트 권한 변경
                    sh 'chmod +x ./backend/api/gradlew'
                    // 프로젝트 빌드
                    dir('./backend/api') {
                        sh './gradlew build -x test'
                    }
                }
            }
            post {
                failure {
                    echo 'API project build failure !'
                }
                success {
                    echo 'API project build success !'
                }
            }
        }
        // Docker
        stage('Docker Hub Login') {
            steps {
                withCredentials([usernamePassword(credentialsId: 'DOCKER_USER', passwordVariable: 'DOCKER_PASSWORD', usernameVariable: 'DOCKER_USERNAME')]) {
                    sh 'echo "$DOCKER_PASSWORD" | docker login -u $DOCKER_USERNAME --password-stdin'
                }
            }
        }
        stage('Build Docker Image for Backend') {
            steps {
                script {
                    docker.build("simhani1/fooding_api:latest", "./backend/api")
                }
            }
        }
        stage('Tag and Push Docker Images') {
            steps {
                script {
                    docker.image("simhani1/fooding_api:latest").push('latest')
                }
            }
        }
        stage('Deploy') {
            steps {
                sshagent(credentials: ['SSH_CREDENTIAL']) {
                    withCredentials([string(credentialsId: 'EC2_SERVER_IP', variable: 'IP')]) {
                        script {
                            sh 'ssh -o StrictHostKeyChecking=no ubuntu@$IP "cd /home/ubuntu/docker_compose_yml && sudo ./api_deploy.sh"'
                        }
                    }
                }
            }
        }
    }
    // MatterMost Noti
    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: "백엔드 배포 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} \n(<${env.BUILD_URL}|Details>)",
                endpoint: 'https://meeting.ssafy.com/hooks/den73uqbhbfdjjjm75arypqcoh',
                channel: 'a5386da344de149433f858178dce5587'
                )
            }
        }
        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: "백엔드 배포 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER}\n(<${env.BUILD_URL}|Details>)",
                endpoint: 'https://meeting.ssafy.com/hooks/den73uqbhbfdjjjm75arypqcoh',
                channel: 'a5386da344de149433f858178dce5587'
                )
            }
        }
    }
}
```

frontend pipeline

```
pipeline {
    agent any
    environment {
        ENV_FILE = credentials('FRONT_ENV') // 프론트엔드용 환경 변수 파일
    }
    stages {
        stage('Git Clone') {
            steps {
                git branch: 'dev-front', credentialsId: 'simhani1', url: 'https://lab.ssafy.com/s11-bigdata-dist-sub1/S11P21A608.git'
            }
            post {
                failure {
                    echo 'Repository clone failure !'
                }
                success {
                    echo 'Repository clone success !'
                }
            }
        }
        stage('Prepare .env File') {
            steps {
                withCredentials([file(credentialsId: 'FRONT_ENV', variable: 'ENV_FILE_PATH')]) {
                    script {
                        sh 'cat $ENV_FILE_PATH > ./frontend/.env'
                        sh 'cat ./frontend/.env' // .env 파일 내용 출력 확인
                    }
                }
            }
        }
        // Docker
        stage('Docker Hub Login') {
            steps {
                withCredentials([usernamePassword(credentialsId: 'DOCKER_USER', passwordVariable: 'DOCKER_PASSWORD', usernameVariable: 'DOCKER_USERNAME')]) {
                    sh 'echo "$DOCKER_PASSWORD" | docker login -u $DOCKER_USERNAME --password-stdin'
                }
            }
        }
        stage('Build Docker Image for Frontend') {
            steps {
                script {
                    docker.build("simhani1/fooding_front:latest", "./frontend")
                }
            }
        }
        stage('Tag and Push Docker Image') {
            steps {
                script {
                    sh 'docker tag simhani1/fooding_front:latest simhani1/fooding_front:latest'
                    sh 'docker push simhani1/fooding_front:latest'
                }
            }
        }
        stage('Deploy') {
            steps {
                sshagent(credentials: ['SSH_CREDENTIAL']) {
                    withCredentials([string(credentialsId: 'EC2_SERVER_IP', variable: 'IP')]) {
                        script {
                            sh 'ssh -o StrictHostKeyChecking=no ubuntu@$IP "cd /home/ubuntu/docker_compose_yml && sudo ./front_deploy.sh"'
                        }
                    }
                }
            }
        }
    }
        // MatterMost Noti
    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: "프론트 배포 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} \n(<${env.BUILD_URL}|Details>)",
                endpoint: 'https://meeting.ssafy.com/hooks/den73uqbhbfdjjjm75arypqcoh',
                channel: 'a5386da344de149433f858178dce5587'
                )
            }
        }
        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: "프론트 배포 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER}\n(<${env.BUILD_URL}|Details>)",
                endpoint: 'https://meeting.ssafy.com/hooks/den73uqbhbfdjjjm75arypqcoh',
                channel: 'a5386da344de149433f858178dce5587'
                )
            }
        }
    }
}
```

flask pipeline

```
pipeline {
    agent any
    stages {
        stage('Git Clone') {
            steps {
                git branch: 'dev-data', credentialsId: 'simhani1', url: 'https://lab.ssafy.com/s11-bigdata-dist-sub1/S11P21A608.git'
            }
            post {
                failure {
                    echo 'Repository clone failure !'
                }
                success {
                    echo 'Repository clone success !'
                }
            }
        }
        // Docker
        stage('Docker Hub Login') {
            steps {
                withCredentials([usernamePassword(credentialsId: 'DOCKER_USER', passwordVariable: 'DOCKER_PASSWORD', usernameVariable: 'DOCKER_USERNAME')]) {
                    sh 'echo "$DOCKER_PASSWORD" | docker login -u $DOCKER_USERNAME --password-stdin'
                }
            }
        }
        stage('Build Docker Image for Backend') {
            steps {
                script {
                    docker.build("simhani1/fooding_flask:latest", "./data")
                }
            }
        }
        stage('Tag and Push Docker Images') {
            steps {
                script {
                    docker.image("simhani1/fooding_flask:latest").push('latest')
                }
            }
        }
        stage('Deploy') {
            steps {
                sshagent(credentials: ['SSH_CREDENTIAL']) {
                    withCredentials([string(credentialsId: 'EC2_SERVER_IP', variable: 'IP')]) {
                        script {
                            sh 'ssh -o StrictHostKeyChecking=no ubuntu@$IP "cd /home/ubuntu/docker_compose_yml && sudo ./flask_deploy.sh"'
                        }
                    }
                }
            }
        }
    }
    // MatterMost Noti
    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: "플라스크 배포 성공: ${env.JOB_NAME} #${env.BUILD_NUMBER} \n(<${env.BUILD_URL}|Details>)",
                endpoint: 'https://meeting.ssafy.com/hooks/den73uqbhbfdjjjm75arypqcoh',
                channel: 'a5386da344de149433f858178dce5587'
                )
            }
        }
        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: "플라스크 배포 실패: ${env.JOB_NAME} #${env.BUILD_NUMBER}\n(<${env.BUILD_URL}|Details>)",
                endpoint: 'https://meeting.ssafy.com/hooks/den73uqbhbfdjjjm75arypqcoh',
                channel: 'a5386da344de149433f858178dce5587'
                )
            }
        }
    }
}
```

빌드 및 실행

docker-compose.backend.yml

```
services:
  app:
    image: simhani1/fooding_api:latest
    container_name: fooding_api # 컨테이너 이름을 지정
    platform: linux/amd64
    env_file: ./.env
    ports:
      - "8002:8080"
    depends_on:
      - redis
      - mysql

  redis:
    image: redis:latest
    container_name: redis # 컨테이너 이름을 지정
    ports:
      - "6379:6379"
    command: ["redis-server", "--appendonly", "yes"]

  mysql:
    image: mysql:latest
    container_name: mysql
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      TZ: Asia/Seoul
    ports:
      - "3306:3306"
    volumes:
      - mysql-data:/var/lib/mysql

volumes:
  mysql-data:
    driver: local
    driver_opts:
      type: none
      device: /var/lib/mysql_data
      o: bind
```

docker-compose.frontend.yml

```
services:
  nginx:
    image: simhani1/fooding_front
    container_name: nginx_container
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf
      - /var/www:/var/www:ro
      - ./zerossl/:/etc/nginx/ssl:ro

  react:
    image: simhani1/fooding_front:latest
    container_name: fooding_front # 컨테이너 이름을 지정
    platform: linux/amd64
    ports:
      - "8004:80"
```

docekr-compose.falsk.yml

```
version: '3'
services:
  fooding_flask:
    image: simhani1/fooding_flask:latest
    container_name: fooding_flask
    ports:
      - "8005:5001"
    volumes:
      - ./data:/app/data
```

api_deploy.sh

```
#!/bin/bash

# Stop and remove existing containers
echo "Stopping and removing existing Docker Compose containers..."
docker compose -f docker-compos.backend.yml down

echo "Existing containers stopped and removed."

# Remove old Docker images
sudo docker rmi simhani1/fooding_api:latest

# Pull new Docker images
echo "Pulling new Docker images..."
sudo docker compose -f docker-compose.backend.yml pull

# Start new backend and frontend containers
echo "Starting frontend and backend containers..."
sudo docker compose -f docker-compose.backend.yml up --build --force-recreate -d

echo "Deployment complete. All containers are now running with the latest images."
```

front_deploy.sh

```
#!/bin/bash

# Stop and remove existing containers
echo "Stopping and removing existing Docker Compose containers..."
docker compose -f docker-compose.frontend.yml down

echo "Existing containers stopped and removed."

# Remove old Docker images
sudo docker rmi simhani1/fooding_front:latest

# Pull new Docker images
echo "Pulling new Docker images..."
sudo docker compose -f docker-compose.frontend.yml pull

# Start new backend and frontend containers
echo "Starting frontend and backend containers..."
sudo docker compose -f docker-compose.frontend.yml up --build --force-recreate -d

echo "Deployment complete. All containers are now running with the latest images."
```

flask_deploy.sh

```
#!/bin/bash

# Stop and remove existing containers
echo "Stopping and removing existing Docker Compose containers..."
docker compose -f docker-compose.flask.yml down

echo "Existing containers stopped and removed."

# Remove old Docker images
echo "Removing old Docker images..."
sudo docker rmi simhani1/fooding_flask:latest

# Pull new Docker images
echo "Pulling new Docker images..."
sudo docker compose -f docker-compose.flask.yml pull

# Start new Flask container
echo "Starting Flask container..."
sudo docker compose -f docker-compose.flask.yml up --build --force-recreate -d

echo "Deployment complete. Flask container is now running with the latest image."
```

nginx.conf

```
events {}

http {

    include       mime.types;

    default_type  application/octet-stream;

    client_max_body_size 10m;

    server {
        listen 80;
        server_name  j11a608.p.ssafy.io j11a608.q.ssafy.io;

        location ^~ /.well-known {
            allow all;
            root /var/www/html;

            # Override the default type for this location
            default_type text/plain;
        }

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

    server {
        listen 443 ssl;
        server_name j11a608.p.ssafy.io j11a608.q.ssafy.io;

        ssl_certificate /etc/nginx/ssl/certificate.crt;
        ssl_certificate_key /etc/nginx/ssl/private.key;
        ssl_trusted_certificate /etc/nginx/ssl/ca_bundle.crt;

        ssl_protocols TLSv1.2 TLSv1.3;

        ssl_ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES12

        root /usr/share/nginx/html;
        location ^~ /.well-known {
            allow all;
            root /var/www/html;

            default_type text/plain;
        }

        location /api/v1 {
            proxy_pass http://fooding_api:8080;
            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_set_header Connection 'keep-alive';  # keep-alive 설정
              proxy_http_version 1.1;  # HTTP/1.1 유지
            proxy_buffering off;  # SSE를 위한 버퍼링 비활성화
            proxy_set_header X-Accel-Buffering 'no';
        }

        location /api/v2 {
            proxy_pass http://fooding_flask:8005;
            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 /admin/jenkins {
            proxy_pass http://jenkins_container:8080;
            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 / {
            index index.html index.htm;
            try_files $uri $uri/ /index.html;
        }

        types {
            text/html html;
            text/css css;
        application/javascript js;
        application/javascript jsx;
            application/typescript ts;
            application/typescript tsx;  # .tsx 파일의 MIME 타입 설정
        }
    }
}
```