포팅 매뉴얼 및 외부 서비스 정보

1. 프로젝트 기술 스택
    - 🌐 공통
    - 📱 FrontEnd
    - 💾 BackEnd
    - ⛏️ Embedded
2. Frontend
    - 패키지 설치 및 실행
3. Backend
    - 빌드 및 배포
4. AWS EC2
    - Docker
    - Jenkins
5. Jenkins
    - GitLab PlugIn 설치 및 연동
    - 자동 빌드 및 배포 설정
        - Jenkins Pipeline script
6. Docker
    - Backend
    - MariaDB
7. Nginx
    - etc/nginx/sites-enabled/default.conf
    - etc/nginx/sites-enabled/jenkins.conf
8. 외부 서비스 정보
    - Firebase

1. 프로젝트 기술 스택

🌐 공통

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

📱 FrontEnd

상세	버전
Flutter	3.13.8
Dart	3.1.4
google_maps_flutter	2.5.0
geolocator	10.1.0
mqtt_client	10.0.0
Android SDK	20 이상

💾 BackEnd

상세	버전
JDK (Zulu)	11.0.19

SpringBoot	2.7.16
Spring Data JPA	2.7.16
Spring Integration	2.7.16
Spring Integration MQTT	5.5.19
Eclipse paho client mqtt v3	1.2.5
MariaDB	8.0.35
MongoDB	2.0.1
Ubuntu	20.04
Nginx	1.18.0
Docker	24.0.7
Docker-compose	2.23.0
Jenkins	2.414.3

⛏️ Embedded

상세	버전
ESP32	
Arduino IDE	2.2.1
ESPMqttClient	1.13.3
ArduinoJson	6.21.3
HX711	0.3.9
Ticker	4.4.0
Raspberry Pi	
Paho-mqtt-c	3.3.1

2. Frontend

패키지 설치 및 실행

```
# 사용 모듈 설치하기
flutter pub get

# apk 빌드(프로젝트 루트 폴더에서)
flutter build apk --release --target-platform=android-arm64

# root/build/app/outputs/apk/app-release.apk
```

3. Backend

빌드 및 배포

- Dockerfile

```
# 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 app.jar

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

```
# Expose port 8080
EXPOSE 8080
```

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 8000:8000 -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

Jenkins Pipeline script

```
pipeline {
    agent any
    tools {
        gradle 'Gradle'
    }
    stages {
        stage('Clone Repository') {
            steps {
                echo "Branch : develop/backend"
                echo "Clone repository"
                git branch: "develop/backend", url: "https://lab.ssafy.com/s09-final/S09P31A310", credentialsId: 'gitlab'
                // git branch: "feature/monitoring", url: "https://lab.ssafy.com/s09-final/S09P31A310", credentialsId: 'gitlab'
            }
        }
        stage("Set environment") {
            steps {
                echo "Copy require file to pipeline folder"
                sh '''
                cp /var/jenkins_home/util/backend/docker-compose-mq.yml ./backend/purugging
                cp /var/jenkins_home/util/backend/application.yml ./backend/purugging/src/main/resources
                cp /var/jenkins_home/util/backend/application-mq.yml ./backend/purugging/src/main/resources
                cp /var/jenkins_home/util/backend/application-s3.yml ./backend/purugging/src/main/resources
                cp /var/jenkins_home/util/backend/banner.txt ./backend/purugging/src/main/resources

                cp /var/jenkins_home/util/backend/prometheus.yml ./backend/purugging

                cp /var/jenkins_home/util/backend/firebase-service-account.json ./backend/purugging/src/main/resources
                cp /var/jenkins_home/util/backend/Dockerfile ./backend/purugging
                '''
            }
        }
        stage('Build Spring Boot App') {
            steps {
                dir('backend/purugging'){  // Adjust the directory path accordingly
                    sh 'gradle clean build'
                }
            }
        }
        stage('Build and Run Spring Boot Container') {
            steps {
                dir('backend/purugging'){
                    script {
```

```
                        sh '''
                        docker rmi $(docker images -f "dangling=true" -q) || true

                        docker rm -f spring || true
                        docker rm -f rabbitmq || true
                        docker rm -f mosquitto || true
                        docker rm -f mongodb || true
                        docker rm -f prometheus || true

                        docker-compose -f docker-compose-mq.yml up -d --build
                        '''
                    }
                }
            }
        }
    }
    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

Backend

- docker-compose.yml

```
version: "3.5"

services:
  mosquitto:
    container_name: mosquitto
    image: toke/mosquitto
    restart: always
    ports:
      - "4883:1883"  # Mosquitto 기본 포트
      - "9001:9001"  # Mosquitto 웹 소켓 포트
    volumes:
      - "./develop/mosquitto/data:/mosquitto/data"  # Mosquitto 데이터를 컨테이너 외부에 저장합니다

  mongodb:
    container_name: mongodb
    image: mongo
    restart: always
    environment:
     - MONGO_INITDB_ROOT_USERNAME=puru
     - MONGO_INITDB_ROOT_PASSWORD=devA310
     - MONGO_INITDB_DATABASE=admin
    ports:
     - "27017:27017"
    volumes:
     - "/home/ubuntu/mongodb:/data/db"

  spring:
    build:
      context: .
      dockerfile: Dockerfile
    container_name : spring
    ports:
      - "8080:8080"
    depends_on:
```

```
      - mongodb
    environment:
      SPRING_DATA_MONGODB_HOST: mongodb
      SPRING_DATA_MONGODB_PORT: 27017
      SPRING_DATA_MONGODB_USERNAME: puru
      SPRING_DATA_MONGODB_PASSWORD: devA310

  prometheus:
    image: prom/prometheus
    container_name: prometheus
    ports:
      - "9090:9090"
    volumes:
      - "./prometheus.yml:/etc/prometheus/prometheus.yml"

  grafana:
    image: "grafana/grafana"
    ports:
      - "3000:3000"
    volumes:
      - /home/ubuntu/grafana:/config_files
```

MariaDB

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

7. Nginx

etc/nginx/sites-enabled/default.conf

```
server {
    server_name k9a310.p.ssafy.io;

    location / {
        proxy_pass http://localhost:1883;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }

    location /api {
        # rewrite ^/api(/.*)$ $1 break;
        proxy_pass http://localhost:8080;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        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 /pro {
    #    proxy_pass http://localhost:8081;
    #    proxy_http_version 1.1;
    #    proxy_set_header Upgrade $http_upgrade;
    #    proxy_set_header Connection "upgrade";
    #    proxy_set_header Host $host;
    #    proxy_set_header X-Real-IP $remote_addr;
    #    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    #}

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/k9a310.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/k9a310.p.ssafy.io/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}
server {
    if ($host = k9a310.p.ssafy.io) {
        return 301 https://$host$request_uri;
```

```
    } # managed by Certbot


    listen 80;
    server_name k9a310.p.ssafy.io;
    return 404; # managed by Certbot


}
```

etc/nginx/sites-enabled/jenkins.conf

```
upstream jenkins {
  keepalive 32; # keepalive connections
  server 127.0.0.1:8000; # jenkins ip and port
}

# Required for Jenkins websocket agents
map $http_upgrade $connection_upgrade {
  default upgrade;
  '' close;
}

server {
  listen          80;       # Listen on port 80 for IPv4 requests

  server_name     jenkins.example.com;  # replace 'jenkins.example.com' with your server domain name

  # this is the jenkins web root directory
  # (mentioned in the output of "systemctl cat jenkins")
  root            /var/run/jenkins/war/;

  access_log      /var/log/nginx/jenkins.access.log;
  error_log       /var/log/nginx/jenkins.error.log;

  # pass through headers from Jenkins that Nginx considers invalid
  ignore_invalid_headers off;

  location ~ "^/static/[0-9a-fA-F]{8}\/(.*)$" {
    # rewrite all static files into requests to the root
    # E.g /static/12345678/css/something.css will become /css/something.css
    rewrite "^/static/[0-9a-fA-F]{8}\/(.*)" /$1 last;
  }

  location /userContent {
    # have nginx handle all the static requests to userContent folder
    # note : This is the $JENKINS_HOME dir
    root /var/lib/jenkins/;
    if (!-f $request_filename){
      # this file does not exist, might be a directory or a /**view** url
      rewrite (.*) /$1 last;
      break;
    }
    sendfile on;
  }

  location / {
      sendfile off;
      proxy_pass         http://jenkins;
      proxy_redirect     default;
      proxy_http_version 1.1;

      # Required for Jenkins websocket agents
      proxy_set_header   Connection        $connection_upgrade;
      proxy_set_header   Upgrade           $http_upgrade;

      proxy_set_header   Host              $http_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_max_temp_file_size 0;

      #this is the maximum upload size
      client_max_body_size       10m;
      client_body_buffer_size    128k;

      proxy_connect_timeout      90;
      proxy_send_timeout         90;
      proxy_read_timeout         90;
      proxy_request_buffering    off; # Required for HTTP CLI commands
```

```
        proxy_set_header Connection ""; # Clear for keepalive
    }

}
```

8. 외부 서비스 정보

Firebase

- Google 로그인 및 회원가입 서비스
- Notification 기능
- https://console.firebase.google.com