🥕

포팅 매뉴얼

1. 사용 도구

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

2. 개발 도구

- Visual Studio Code
- Intellij : 2022.3.2 (Ultimate Edition)
- MobaXTerm
- Docker Desktop

3. 개발 환경

- 상세 내용

Frontend

Android Studio	`Hedgehog 2023.1.1 Patch 2`
Flutter	`3.19.1`

Backend

Java

Java	`OpenJDK 17`
Spring Boot	`3.2.4`
gradle	`gradle-8.7-bin`

A.I

Colab	
Jupyter Notebook	
Python	`3.8.10`
Flask	`3.0.3`
Keras	`2.13.1`
Tensorflow	`2.13.1`
Numpy	`1.24.3`
Torch	`1.8.0`
OpenCV	`4.9.0.80`
ResNet50	
YoloV5	
AI Hub	
Roloflow	

Server

AWS S3	
AWS EC2	CPU : Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz /dev/root : 311G `T2.xLarge`
RAM	15GB

OS	Ubuntu

IoT

Arduino IDE	
VSCod	
MCU	ESP32-CAM

Service

MySQL	8.0.36
NginX	1.18.0
Jenkins	2.445
Docker	26.1.2
Ubuntu	Ubuntu 20.04 LTS

4. 환경변수 형태

- Jenkins

AWS_ACCESS_KEY_ID	AKIA2UC27RTFNQDD7ICX
AWS_REGION	ap-northeast-2
AWS_S3_BUCKET	ggbro
AWS_SECRET_ACCESS_KEY	LQBplWSgcSQLGpltWNexqfz9RbkpghrMkN/fIM9h
DB_NAME	ssafy
DB_PASSWORD	ssafy
DB_URL	jdbc:mysql://k10c206.p.ssafy.io:3306/ggbro?useUnicode=true&characterEncoding=utf8&serverTimezone=Asia/Seoul&zeroDateTimeBehavior=convertToNull&rewriteBatchedStatements=true
FLASK_SERVER_URL	https://k10c206.p.ssafy.io/predict
JWT_ACCESS	TestAccessKey1111112222222222333333333333aaaaaaaabbbbbbbccccccccddddddddddfqwrqwfqwifjaisfjoihfoiqwflasjdglkkjasdlgjlkasdgjiowejgliasjglasjdgio
JWT_REFRESH	TestRefreshKey1111112222222222333333333333aaaaaaaabbbbbbbccccccccddddddddddfqwrqwfqwifjaisfjoihfoiqwflasjdglkkjasdlgjlkasdgjiowejgliasjglasjdgio
REDIS_HOST	k10c206.p.ssafy.io

5. CI/CD 구축

EC2 환경설정

EC2 인스턴스에 다음을 설치

- Docker
- Jenkins
- Mysql
- Redis

Docker

```
$ sudo apt-get update
$ sudo apt-get install apt-transport-https ca-certificates curl software-properties-common
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
$ echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/lin
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

Jenkins

```
# Jenkins 전용 포트 개방
$ sudo ufw allow 9001

# Jenkins 컨테이너 실행
$ docker run -d -p 9001:8080 -v /home/ubuntu/jenkins-data:/var/jenkins_home \
  -v /var/run/docker.sock:/var/run/docker.sock -u root --name jenkins jenkins\
  /jenkins

# 젠킨스 환경설정
$ cd /home/ubuntu/jenkins-data

$ mkdir update-center-rootCAs

$ wget https://cdn.jsdelivr.net/gh/lework/jenkins-update-center/rootCA/\
update-center.crt \
  -O ./update-center-rootCAs/update-center.crt

$ sudo sed -i \
  's#https://updates.jenkins.io/update-center.json'\
  '#https://raw.githubusercontent.com/' \
  'lework/jenkins-update-center/master/updates/tencent/update-center.json#' \
  ./hudson.model.UpdateCenter.xml

$ sudo docker restart jenkins
```

DockerFile

- Backend

```
# 빌드 스테이지
FROM amazoncorretto:17.0.7-alpine AS builder
USER root
WORKDIR /back
COPY gradlew .
COPY gradle gradle
COPY build.gradle .
COPY settings.gradle .
COPY src src
# gradlew 실행 권한 부여
RUN chmod +x ./gradlew
RUN ./gradlew bootJar

# 실행 스테이지
FROM openjdk:17
WORKDIR /back
COPY --from=builder /back/build/libs/*.jar app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
VOLUME /tmp
```

- Flask

```
# Python 이미지 사용
FROM python:3.8.10

# 필요한 시스템 라이브러리 설치
RUN apt-get update && apt-get install -y \
    libgl1-mesa-glx \
    libglib2.0-0 \
    python3-pip

# 작업 디렉터리 설정
WORKDIR /app

# yolov5 폴더 내의 의존성 파일 복사
COPY yolov5/requirements.txt .

# pip 최신 버전으로 업그레이드 및 필요한 Python 패키지 설치
RUN pip install --no-cache-dir --upgrade pip && \
    pip install --no-cache-dir -r requirements.txt

# 나머지 애플리케이션 파일 복사
COPY . .

# 환경 변수 설정
ENV FLASK_APP=app.py
ENV FLASK_RUN_HOST=0.0.0.0

# 포트 5000 열기
EXPOSE 5000

# Flask 앱 실행
CMD ["python", "app.py"]
```

- Redis

```
version: '3.1'

networks:
  app-tier:
    driver: bridge

services:

  redis:
    image: 'bitnami/redis:latest'
    container_name: redis
    # restart: always
    environment:
      - REDIS_REPLICATION_MODE=master
      - ALLOW_EMPTY_PASSWORD=yes
    ports:
      - "6379:6379"
    networks:
      - app-tier

  redis-slave-1:
    image: 'bitnami/redis:latest'
    container_name: redis-slave-1
    # restart: always
    environment:
      - REDIS_REPLICATION_MODE=slave
      - REDIS_MASTER_HOST=redis
      - ALLOW_EMPTY_PASSWORD=yes
    ports:
      - "6380:6379"

    depends_on:
      - redis
    networks:
      - app-tier

  redis-slave-2:
    image: 'bitnami/redis:latest'
    container_name: redis-slave-2
    # restart: always
    environment:
      - REDIS_REPLICATION_MODE=slave
      - REDIS_MASTER_HOST=redis
      - ALLOW_EMPTY_PASSWORD=yes
    ports:
      - "6381:6379"

    depends_on:
      - redis
    networks:
      - app-tier

  redis-sentinel:
    image: 'bitnami/redis-sentinel:latest'
    environment:
      - REDIS_SENTINEL_DOWN_AFTER_MILLISECONDS=3000
      - REDIS_MASTER_HOST=redis
      - REDIS_MASTER_PORT_NUMBER=6379
      - REDIS_MASTER_SET=mymaster
      - REDIS_SENTINEL_QUORUM=2
    depends_on:
      - redis
      - redis-slave-1
      - redis-slave-2
    ports:
      - '26379-26381:26379'
    networks:
      - app-tier
```

Jenkins File

- Backend

```
pipeline {
    agent any
    environment {
        REPO = "s10-final/S10P31C206"
        DOCKERHUB_REGISTRY = "kimhyeokil/back"
        DOCKERHUB_CREDENTIALS = credentials('Docker-hub')
    }
    stages {
        stage('Checkout') {
            steps {
                checkout scm

                sh """
                sed -i 's|\${JWT_ACCESS}|${env.JWT_ACCESS}|g' BackEnd/src/main/resources/application.yml
                sed -i 's|\${JWT_REFRESH}|${env.JWT_REFRESH}|g' BackEnd/src/main/resources/application.yml
                sed -i 's|\${DB_URL}|${env.DB_URL}|g' BackEnd/src/main/resources/application.yml
                sed -i 's|\${REDIS_HOST}|${env.REDIS_HOST}|g' BackEnd/src/main/resources/application.yml
                sed -i 's|\${DB_NAME}|${env.DB_NAME}|g' BackEnd/src/main/resources/application.yml
                sed -i 's|\${DB_PASSWORD}|${env.DB_PASSWORD}|g' BackEnd/src/main/resources/application.yml
                sed -i 's|\${AWS_ACCESS_KEY_ID}|${env.AWS_ACCESS_KEY_ID}|g' BackEnd/src/main/resources/application.yml
                sed -i 's|\${AWS_SECRET_ACCESS_KEY}|${env.AWS_SECRET_ACCESS_KEY}|g' BackEnd/src/main/resources/application.yml
                sed -i 's|\${AWS_REGION}|${env.AWS_REGION}|g' BackEnd/src/main/resources/application.yml
                sed -i 's|\${AWS_S3_BUCKET}|${env.AWS_S3_BUCKET}|g' BackEnd/src/main/resources/application.yml
                sed -i 's|\${FLASK_SERVER_URL}|${env.FLASK_SERVER_URL}|g' BackEnd/src/main/resources/application.yml
                """

            }
        }
        stage('Setup Environment') {
            steps {
                dir("${env.WORKSPACE}/BackEnd"){
                    script {
                        sh "ls -al"
                        sh "chmod +x ./gradlew"
                    }
                }
            }
        }
        stage("Build") {
            steps {
                script {
                    sh "docker build -t ${DOCKERHUB_REGISTRY} BackEnd"
                }
            }
        }
        stage("Login") {
            steps {
                 sh "echo \${DOCKERHUB_CREDENTIALS_PSW} | docker login -u \${DOCKERHUB_CREDENTIALS_USR} --password-stdin"
            }
        }
        stage("Tag and Push") {
            steps {
                script {
                    withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId: 'Docker-hub', usernameVariable: 'DOCKER_U
SER_ID', passwordVariable: 'DOCKER_USER_PASSWORD']]) {
                        sh "docker push ${DOCKERHUB_REGISTRY}"
                    }
                }
            }
        }
        stage('Prune old images'){
            steps{
                script{
                    sh "docker ps"
                }
            }
        }
        stage('Pull') {
            steps {
                script {
                    withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId: 'Docker-hub', usernameVariable: 'DOCKER_U
SER_ID', passwordVariable: 'DOCKER_USER_PASSWORD']]) {
                        sh "docker stop back || true"  // Ignore error ifgit  container doesn't exist
                        sh "docker rm back || true"    // Ignore error if container doesn't exist
                        sh "docker rmi ${DOCKERHUB_REGISTRY}|| true"     //images 날리기
                        sh "docker pull ${DOCKERHUB_REGISTRY}"
                    }
                }
            }
        }
        stage('Up') {
            steps {
                script {
                    withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId: 'Docker-hub', usernameVariable: 'DOCKER_U
SER_ID', passwordVariable: 'DOCKER_USER_PASSWORD']]) {
                        try {
                        sh "docker run -d --name back -p 9003:8080 \
                        ${DOCKERHUB_REGISTRY}"
//                             sh "docker-compose -f ${env.WORKSPACE}/docker-compose.yml up -d"

                        } catch(Exception e) {
                            sh "docker restart back || true"  // Ignore error if container doesn't exist
                        }
                    }
                }
            }
        }


    }
}
```

- Flask

```
pipeline {
    agent any
    environment {
        REPO = "s10-final/S10P31C206"
        DOCKERHUB_REGISTRY = "kimhyeokil/ai"
        DOCKERHUB_CREDENTIALS = credentials('Docker-hub')

    }
    stages {
        stage('Checkout') {
            steps {
                checkout scm

            }
        }
        stage("Build") {
            steps {
                script {
                    sh "docker build -t ${DOCKERHUB_REGISTRY} FlaskServer"
                }
            }
        }
        stage("Login") {
            steps {
                 sh "echo \${DOCKERHUB_CREDENTIALS_PSW} | docker login -u \${DOCKERHUB_CREDENTIALS_USR} --password-stdin"
            }
        }
        stage("Tag and Push") {
            steps {
                script {
                    withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId: 'Docker-hub', usernameVariable: 'DOCKER_USE
                        sh "docker push ${DOCKERHUB_REGISTRY}"
                    }
                }
            }
        }
        stage('Prune old images'){
            steps{
                script{
                    sh "docker ps"
                }
            }
        }
        stage('Pull') {
            steps {
                script {
                    withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId: 'Docker-hub', usernameVariable: 'DOCKER_USE
                        sh "docker stop ai || true"  // Ignore error ifgit  container doesn't exist
                        sh "docker rm ai || true"    // Ignore error if container doesn't exist
                        sh "docker rmi ${DOCKERHUB_REGISTRY}|| true"     //images 날리기
                        sh "docker pull ${DOCKERHUB_REGISTRY}"
                    }
                }
            }
        }
        stage('Up') {
            steps {
                script {
                    withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId: 'Docker-hub', usernameVariable: 'DOCKER_USE
                        try {
                        sh "docker run -d --name ai -p 9002:5000 \
                        ${DOCKERHUB_REGISTRY}"
//                             sh "docker-compose -f ${env.WORKSPACE}/docker-compose.yml up -d"

                        } catch(Exception e) {
                            sh "docker restart ai || true"  // Ignore error if container doesn't exist
                        }
                    }
                }
            }
        }


    }
}
```

Nginx

```
##
# You should look at the following URL's in order to grasp a solid understanding
# of Nginx configuration files in order to fully unleash the power of Nginx.
# https://www.nginx.com/resources/wiki/start/
# https://www.nginx.com/resources/wiki/start/topics/tutorials/config_pitfalls/
# https://wiki.debian.org/Nginx/DirectoryStructure
#
# In most cases, administrators will remove this file from sites-enabled/ and
# leave it as reference inside of sites-available where it will continue to be
# updated by the nginx packaging team.
#
# This file will automatically load configuration files provided by other
# applications, such as Drupal or Wordpress. These applications will be made
# available underneath a path with that package name, such as /drupal8.
#
# Please see /usr/share/doc/nginx-doc/examples/ for more detailed examples.
##

# Default server configuration
#
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        # SSL configuration
        #
        # listen 443 ssl default_server;
        # listen [::]:443 ssl default_server;
        #
        # Note: You should disable gzip for SSL traffic.
        # See: https://bugs.debian.org/773332
        #
        # Read up on ssl_ciphers to ensure a secure configuration.
        # See: https://bugs.debian.org/765782
        #
        # Self signed certs generated by the ssl-cert package
        # Don't use them in a production server!
        #
        # include snippets/snakeoil.conf;

        root /var/www/html;

        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;

        server_name _;

        location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                try_files $uri $uri/ =404;
        }

        location ~ /.well-known/acme-challenge {
            allow all;
            root /var/www/html;
        }

        # pass PHP scripts to FastCGI server
        #
        #location ~ \.php$ {
        #       include snippets/fastcgi-php.conf;
        #
        #       # With php-fpm (or other unix sockets):
        #       fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
        #       # With php-cgi (or other tcp sockets):
        #       fastcgi_pass 127.0.0.1:9000;
        #}

        # deny access to .htaccess files, if Apache's document root
        # concurs with nginx's one
        #
        #location ~ /\.ht {
        #       deny all;
        #}
}


# Virtual Host configuration for example.com
#
# You can move that to a different file under sites-available/ and symlink that
# to sites-enabled/ to enable it.
#
#server {
#       listen 80;
#       listen [::]:80;
#
#       server_name example.com;
#
#       root /var/www/example.com;
#       index index.html;
#
#       location / {
#               try_files $uri $uri/ =404;
#       }
#}

server {

        # SSL configuration
        #
        # listen 443 ssl default_server;
        # listen [::]:443 ssl default_server;
        #
        # Note: You should disable gzip for SSL traffic.
        # See: https://bugs.debian.org/773332
        #
        # Read up on ssl_ciphers to ensure a secure configuration.
        # See: https://bugs.debian.org/765782
        #
        # Self signed certs generated by the ssl-cert package
        # Don't use them in a production server!
        #
        # include snippets/snakeoil.conf;

        root /var/www/html;

        # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;
    server_name k10c206.p.ssafy.io; # managed by Certbot


        location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                try_files $uri $uri/ =404;
        }

        location ~ /.well-known/acme-challenge {
            allow all;
            root /var/www/html;
        }
        location /api/v1 {
                proxy_pass http://localhost:9003;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;

        }

        location /predict {
                proxy_pass http://localhost:9002;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;

        }

        # pass PHP scripts to FastCGI server
        #
        #location ~ \.php$ {
        #       include snippets/fastcgi-php.conf;
        #
        #       # With php-fpm (or other unix sockets):
        #       fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
        #       # With php-cgi (or other tcp sockets):
        #       fastcgi_pass 127.0.0.1:9000;
        #}

        # deny access to .htaccess files, if Apache's document root
        # concurs with nginx's one
        #
        #location ~ /\.ht {
        #       deny all;
        #}
    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/k10c206.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/k10c206.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 ($host = k10c206.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


        listen 80 ;
        listen [::]:80 ;
    server_name k10c206.p.ssafy.io;
    return 404; # managed by Certbot


}
```

6. DB 덤프파일

create_spatial_index.sql	공간인덱스 활용을 위한 인덱스 생성 쿼리
data_achievement.sql	업적 기본정보
data_member.sql	기본 멤버
data_member_achievement.sql	기본 멤버에게 할당되는 업적
data_member_info.sql	기본 멤버에게 할당되는 사용자정보
data_member_pet.sql	기본 멤버에게 할당되는 펫
data_member_quest.sql	기본 멤버에게 할당되는 퀘스트
data_notice.sql	플로깅 공지사항 기본정보
data_pet.sql	펫 기본정보
data_plogging.sql	기본 멤버에게 할당되는 플로깅정보
data_plogging_route.sql	기본 멤버의 플로깅정보에 할당되는 경로정보
data_quest.sql	퀘스트 기본정보
data_trash.sql	쓰레기 데이터 30만개

7. 시연 시나리오

1. 플로깅 히스토리 먼저 열람
2. 플로깅 히스토리 상세 화면 열람
3. 유저 업적 달성현황 열람
4. 업적 달성 보상으로 얻은 재화를 이용해 펫 구출 시연
5. 사용자 정보 화면에 들어가서 유저 프로필 사진 변경
6. 얻은 펫의 경험치를 채워 상자에서 펫 개봉
7. 펫과 처치한 몬스터 현황 열람
8. 퀘스트 기능 열람
9. 유저 랭킹 기능 열람
10. 전국 캠페인 정보 열람
11. 플로깅 진행 시연
    a. 플로깅 준비 화면
        i. 지역 내 주변 쓰레기통 위치
        ii. 지역 내 처치 몬스터 현황
    b. 플로깅 집게와 블루투스 연결
    c. 플로깅 시작
        i. 쓰레기 발견. 이를 집게에 달린 카메라로 촬영
        ii. 촬영된 쓰레기가 처리되어 분류되는 과정 시연
12. 시연자 퇴장