포팅 매뉴얼

1. 개발 환경 및 버전

Version

- **Server OS** : `Ubuntu 20.04 LTS`
- **JDK** : `OpenJDK17`
- **MySQL** : `8.0.36`
- **Jenkins** : `2.426.3`
- **Nginx** : `1.18.0`
- **Redis** : `5.0.7`
- **Flask** : 그 외 AI
- **React** : `18.2.0`
- **TypeScript** : `5.2.2`

시스템 구성

Domain : `https://i10a305.p.ssafy.io`

- **Spring Boot**
  - **요청 처리 서버** : `8081:8080`
  - **후 처리 서버** : `8083:8080`
- **Flask**
  - **AI 모델 서버** : `8084:80`
- **Jenkins** : `http://i10a305.p.ssafy.io`
  - `8082:8082`
  - `50002:50002`
- **Mysql** : `3306:3306`
- **Redis** : `6379:6379`

2. Setting file

Spring

application.yml

```
spring:
  profiles:
    active:
      - secret
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
  jpa:
    hibernate:
      ddl-auto: none
    properties:
      hibernate:
        format_sql: true
        dialect: org.hibernate.dialect.MySQLDialect
logging.level:
  org.hibernate.SQL: debug
```

application-security.yml

```
jwt:
  salt: "hello"
  accessTokenExpireTime: 1800000
  refreshTokenExpireTime: 3600000

cloud:
  aws:
    s3:
      bucket: a305-project-bucket
      region:
        static: ap-northeast-2
    stack.auto: false
    credentials:
      accessKey: AKIA3FLDXCWKWW5EG2CJ
      secretKey: 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
spring:
  datasource:
    url: jdbc:mysql://i10a305.p.ssafy.io:3306/abc
    username: a305
    password: 305
    driver-class-name: com.mysql.cj.jdbc.Driver

  data:
    redis:
      host: i10a305.p.ssafy.io
      port: 6379
      password: a305#@!

server:
  port: 443
  ssl:
    enabled: true
    key-store: classpath:bootsecurity.p12
    key-store-password: hyeonjo96
    key-store-type: PKCS12
```

인증키 생성

```
// let’s encrypt 설치
sudo apt-get install letsencrypt

// 인증서 발급
sudo certbot certonly -d "i10a305.p.ssafy.io" --manual --preferred-challenges dns
```

React

.env

```
VITE_AWS_ACCESS_KEY = "AKIA2UC3E76VLWQAMVAE"
VITE_AWS_SECRET_KEY = "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"
VITE_AWS_S3_BUCKET = "testoree"
VITE_AWS_S3_BUCKET_REGION = "ap-northeast-2"
VITE_API_BASE_URL = 'http://i10a305.p.ssafy.io:8081'
VITE_API_SIGNUP_URL = '/member/join'
VITE_API_LOGIN_URL = '/member/login'
VITE_API_EMAIL_CHECK_URL = '/member/check-email'
VITE_API_CLIP_URL = '/clip/list'
VITE_ENV = 'development'
```

3. Docker file

Processing Server

```
# 베이스 이미지로 OpenJDK 17 이미지를 사용
FROM openjdk:17-jdk-alpine

# 작업 디렉토리를 설정
WORKDIR /home/webapp/taskmaster

# 변수 선언
ARG JAR_FILE=taskmaster-0.0.1-SNAPSHOT.jar

# 호스트의 빌드 파일을 Docker 이미지 내로 복사
COPY ${JAR_FILE} /home/webapp/taskmaster/taskmasterboot.jar

# JAR 파일을 실행하는 명령을 설정
ENTRYPOINT ["java", "-jar", "/home/webapp/taskmaster/taskmasterboot.jar"]
```

AI Spring Server

```
# 베이스 이미지로 OpenJDK 17 이미지를 사용
FROM openjdk:17-jdk-alpine

# 작업 디렉토리를 설정
WORKDIR /home/aiapp/app

# 변수 선언
ARG JAR_FILE=ai-0.0.1-SNAPSHOT.jar

# 호스트의 빌드 파일을 Docker 이미지 내로 복사
COPY ${JAR_FILE} /home/aiapp/app/aiboot.jar

# JAR 파일을 실행하는 명령을 설정
ENTRYPOINT ["java", "-jar", "/home/aiapp/app/aiboot.jar"]
```

AI Flask Server

```
#
FROM python:3.9

#
WORKDIR /app

#
COPY ./requirements.txt /app/requirements.txt

RUN apt-get update

RUN apt-get -y install libgl1-mesa-glx

RUN apt-get -y install tesseract-ocr

#
RUN pip install --no-cache-dir --upgrade -r /app/requirements.txt

#
COPY ./app/ /app/

#
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "80"]
```

4. S3 Lamda

동영상 저장 알림을 위한 Lamda 설정

```
import json
from urllib.request import Request, urlopen
from urllib.error import URLError, HTTPError

def lambda_handler(event, context):
    for record in event['Records']:
        bucket_name = record['s3']['bucket']['name']
        file_key = record['s3']['object']['key']

        data = json.dumps({
            "Records": [
                {
                    "s3": {
                        "bucket": {
                            "name": bucket_name
                        },
                        "object": {
                            "key": file_key
                        }
                    }
                }
            ]
        }).encode('utf-8')

        req = Request("https://i10a305.p.ssafy.io:8081/S3/endpoint", data=data, headers={'Content-Type': 'application/json'}, method='POST')

        try:
            response = urlopen(req)
            response_data = response.read()
            print(f"Response data: {response_data}")
        except HTTPError as e:
            print(f"HTTP Error: {e.code} {e.reason}")
        except URLError as e:
            print(f"URL Error: {e.reason}")

    return {
        'statusCode': 200,
        'body': 'File URL processed and sent to server successfully'
    }
```

5. Nginx 설정

nginx.conf

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
        worker_connections 768;
        # multi_accept on;
}

http {

        ##
        # Basic Settings
        ##

        sendfile on;
        tcp_nopush on;
        tcp_nodelay on;
        keepalive_timeout 65;
        types_hash_max_size 2048;
        # server_tokens off;

        # server_names_hash_bucket_size 64;
        # server_name_in_redirect off;

        include /etc/nginx/mime.types;
        default_type application/octet-stream;

        # 443 포트 접근 시 SSL 적용
        server {
                server_name i10a305.p.ssafy.io;

                location / {
                        root /home/frontapp/dist;
                        try_files $uri /index.html;
                }

                listen 443 ssl; # managed by Certbot
                ssl_certificate /etc/letsencrypt/live/i10a305.p.ssafy.io/fullchain.pem; # managed by Certbot
                ssl_certificate_key /etc/letsencrypt/live/i10a305.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

        }

        ##
        # SSL Settings
        ##

        ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; # Dropping SSLv3, ref: POODLE
        ssl_prefer_server_ciphers on;

        ##
        # Logging Settings
        ##

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

        ##
        # Gzip Settings
        ##

        gzip on;

        # gzip_vary on;
        # gzip_proxied any;
        # gzip_comp_level 6;
        # gzip_buffers 16 8k;
        # gzip_http_version 1.1;
        # gzip_types text/plain text/css application/json application/javascript text/xml application/xml application/xml+rss text/javascript;

        ##
        # Virtual Host Configs
        ##

        include /etc/nginx/conf.d/*.conf;
        include /etc/nginx/sites-enabled/*;

        # 80 포트 접근 시 443 포트로 리다이렉트
        server {
                if ($host = i10a305.p.ssafy.io) {
                        return 301 https://$host$request_uri;
                } # managed by Certbot


                listen 80;
                server_name i10a305.p.ssafy.io;
                return 404; # managed by Certbot
        }
}
```

6. 젠킨스 파이프라인

요청 처리 서버 Script

```
pipeline {
    agent any

    stages {
        stage('git clone') {
            steps {
                git credentialsId: 'gitlab_credential', branch : 'backend/taskmaster/deploy', url: 'https://lab.ssafy.com/s10-webmobile2-sub2/S10P12A305.git'
            }
        }
        stage('인증서 추가') {
            steps {
                withCredentials([file(credentialsId: 'secret-httpsauth', variable: 'authConfig')]) {
                  script {
                        sh 'cp $authConfig ./src/main/resources/bootsecurity.p12'
                        sh 'chmod 644 ./src/main/resources/bootsecurity.p12'
                    }
                }
            }
    }
        stage('yml 추가') {
            steps {
                withCredentials([file(credentialsId: 'secret-yml', variable: 'yamlConfig')]) {
                  script {
                      sh 'cp $yamlConfig ./src/main/resources/application-secret.yml'
                    }
                }
            }
    }
        stage('bootjar 생성'){
            steps{
                sh 'sh ./gradlew clean bootjar'
            }
        }
        stage('jar 파일 업로드'){
            steps{
                sh 'scp -i /home/keys/I10A305T.pem ./build/libs/*.jar ubuntu@i10a305.p.ssafy.io:/home/webapp'
            }
        }
        stage('docker 이미지, 컨테이너 생성 및 실행'){
            steps{
                script {
                    def remote = [:]
                    remote.name = 'remote server'
                    remote.host = 'i10a305.p.ssafy.io'
                    remote.user = 'ubuntu'
                    remote.identityFile = '/home/keys/I10A305T.pem'
                    remote.allowAnyHosts = true

                    sshCommand remote: remote, command: '''
                        docker stop taskmaster-server || true
                        docker rm taskmaster-server || true
                        docker build -t taskmaster /home/webapp
                        docker run -d -p 8081:443 -it --mount type=bind,source=/home/image,target=/home/webapp/taskmaster/image --name taskmaster-server taskmaster
                    '''
                }
            }
        }
    }
    post {
        always {
            cleanWs()
            dir("${env.WORKSPACE}@tmp") {
                deleteDir()
            }
        }
    }
}
```

후 처리 서버 Script

```
pipeline {
    agent any

    stages {
        stage('git clone') {
            steps {
                git credentialsId: 'gitlab_credential', branch : 'backend/ai/deploy', url: 'https://lab.ssafy.com/s10-webmobile2-sub2/S10P12A305.git'
            }
        }
        stage('인증서 추가') {
            steps {
                withCredentials([file(credentialsId: 'secret-httpsauth', variable: 'authConfig')]) {
                  script {
                        sh 'cp $authConfig ./src/main/resources/bootsecurity.p12'
                        sh 'chmod 644 ./src/main/resources/bootsecurity.p12'
                    }
                }
            }
    }
        stage('yml 추가') {
            steps {
                withCredentials([file(credentialsId: 'secret-yml', variable: 'yamlConfig')]) {
                  script {
                      sh 'cp $yamlConfig ./src/main/resources/application-secret.yml'
                    }
                }
            }
    }
        stage('bootjar 생성'){
            steps{
                sh 'sh ./gradlew clean bootjar'
            }
        }
        stage('jar 파일 업로드'){
            steps{
                sh 'scp -i /home/keys/I10A305T.pem ./build/libs/*.jar ubuntu@i10a305.p.ssafy.io:/home/aiapp'
            }
        }
        stage('docker 컨테이너 생성 및 실행'){
            steps{
                script {
                    def remote = [:]
                    remote.name = 'remote server'
                    remote.host = 'i10a305.p.ssafy.io'
                    remote.user = 'ubuntu'
                    remote.identityFile = '/home/keys/I10A305T.pem'
                    remote.allowAnyHosts = true

                    sshCommand remote: remote, command: '''
                        docker stop ai-spring-server || true
                        docker rm ai-spring-server || true
                        docker build -t ai-spring-image /home/aiapp
                        docker run -d -p 8083:8080 -it --mount type=bind,source=/home/video,target=/home/video --name ai-spring-server ai-spring-image
                    '''
                }
            }
        }
    }
    post {
        always {
            cleanWs()
            dir("${env.WORKSPACE}@tmp") {
                deleteDir()
            }
        }
    }
}
```

Frontend Script

```
pipeline {
    agent any

    stages {
        stage('git 클론') {
            steps {
                git credentialsId: 'gitlab_credential', branch : 'frontend/develop', url: 'https://lab.ssafy.com/s10-webmobile2-sub2/S10P12A305.git'
            }
        }
        stage('.env 추가') {
            steps {
                withCredentials([file(credentialsId: 'env-secret', variable: 'envConfig')]) {
                  script {
                      sh 'cp $envConfig ./frontend/.env'
                    }
                }
            }
    }

        stage('React 빌드') {
            steps {
                sh '''
                    cd ./frontend
                    npm install
                    npm run build
                '''
            }
        }

        stage('build 파일 업로드') {
            steps {
                sh 'scp -i /home/keys/I10A305T.pem -r ./frontend/dist ubuntu@i10a305.p.ssafy.io:/home/frontapp'
            }
        }
        stage('nginx 리로드') {
            steps {
                sh 'ssh -i /home/keys/I10A305T.pem ubuntu@i10a305.p.ssafy.io sudo systemctl reload nginx'
            }
        }

    }

    post {
        always {
            cleanWs()
            dir("${env.WORKSPACE}@tmp") {
                deleteDir()
            }
        }
    }
}
```

7. 시연 시나리오

- 동영상 선택을 클릭하여 영상 업로드를 진행한다.
- 로딩 현황 확인한다.
- 100% 완료 시 결과창으로 이동한다.
- 경기 레포트 페이지에서 경기 분석 결과 확인한다.
- 이닝별 페이지에서 팀별, 이닝별 데이터 변화 확인한다.
- 불꽃 마크가 있는 선수 카드를 클릭하여 타석 영상을 확인한다.
- 북마크 버튼 클릭하여 찜 목록 추가한다.
- 타석별 페이지로 넘어가서 선수별 타석 영상을 시청한다.
- 시연 종료.