A601 LoL Watcher 포팅메뉴얼

1. 메인 서버

1. 컨테이너 정보

- Backend : 메인 서비스 컨테이너
  - 8080:8080
- Frontend : 메인 프론트엔드 컨테이너
  - 3000:80
- Data : 데이터 분석 컨테이너
  - 5000:5000
- Jenkins : 젠킨스 컨테이너
  - 8070:8080

2. 주요 흐름

- 각각의 컨테이너를 Gitlab의 개별 브랜치로 관리합니다.
- 각각의 브랜치를 Dockerfile을 사용해 필드하고 각기 다른 WebHook과 젠킨스 pipeline을 통해 CI/CD를 구현합니다.

3. 개발 환경

1. 프로젝트 기술 스택

- Frontend
  - React : 18.3.1
  - TypeScript : 4.9.5
  - Node : 21

- Backend
  - Java : 17
  - SpringBootFrameWork : 3.3.4
  - Spring Dependency-Management : 1.1.6
  - JsonWebToken : 0.12.3

- Data
  - Python : 3.9-slim
  - Flask : 2.0.1
  - pyMongo : 3.12.0
  - Python-dotenv : 1.0.1
  - Werkzeug : 2.0.3

- Nginx
  - Nginx : 1.18.0

- Git
  - Git : 2.25.1

- Docker
  - Docker : 27.3.1

- MySQL
  - MySQL : 8.0.40

4. 설정 파일

- JenkinsPipeline
  - Backend Pipeline

```
pipeline {
    agent any
    environment {
        DOCKER_IMAGE = 'sosuh97/lolwatcher-backend:latest
        DATA_SERVER_URL = credentials('DATA_SERVER_URL')
        DB_URL = credentials('DB_URL')
        DB_USERNAME = credentials('DB_USERNAME')
        DB_PASSWORD = credentials('DB_PASSWORD')
        MONGO_DB_URL = credentials('MONGO_DB_URL')
        MONGO_DB_PORT = credentials('MONGO_DB_PORT')
        MONGO_DB_DATABASE = credentials('MONGO_DB_DATABASE
        MONGO_DB_USERNAME = credentials('MONGO_DB_USERNAME
        MONGO_DB_PASSWORD = credentials('MONGO_DB_PASSWORD
        JWT_KEY = credentials('JWT_KEY') // JWT_KEY 크레덴셜
        RIOT_API_KEY = credentials('RIOT_API_KEY') // RIOT
        REDIS_PASSWORD = credentials('REDIS_PASSWORD') //
        REDIS_HOST_IP = credentials('REDIS_HOST_IP') // RE
        DOMAIN = credentials('DOMAIN') // DOMAIN 크레덴셜
        FRONT_PORT = credentials('FRONT_PORT') // FRONT_PO
    }
    stages {
        stage('Clone Repository') {
```

```
            steps {
                script {
                    echo "Starting Clone Repository stage'
                    // Git 클론 수행
                    git branch: 'backend-dev',
                        credentialsId: 'ssw_id', // 새로 설
                        url: 'https://lab.ssafy.com/s11-fi
                }
            }
        }
        stage('Build Docker Image') {
            steps {
                script {
                    docker.build("${DOCKER_IMAGE}", "-f lo
                }
            }
        }
        stage('Push Docker Image') {
            steps {
                script {
                    docker.withRegistry('https://index.doc
                        docker.image("${DOCKER_IMAGE}").pu
                    }
                }
            }
        }
        stage('Deploy to EC2') {
            steps {
                sshagent(credentials: ['a601-ubuntu']) {
                    sh '''#!/bin/bash
                    ssh -o StrictHostKeyChecking=no ubuntu
                    "docker pull ${DOCKER_IMAGE} && \
                    docker stop lolwatcher-backend || true
                    docker rm lolwatcher-backend || true &
                    docker run -d --name lolwatcher-backen
                    -e DATA_SERVER_URL="${DATA_SERVER_URL]
```

```
                    -e DB_URL="${DB_URL}"\
                    -e DB_USERNAME="${DB_USERNAME}"\
                    -e DB_PASSWORD="${DB_PASSWORD}"\
                    -e MONGO_DB_URL="${MONGO_DB_URL}"\
                    -e MONGO_DB_PORT="${MONGO_DB_PORT}"\
                    -e MONGO_DB_DATABASE="${MONGO_DB_DATAB
                    -e MONGO_DB_USERNAME="${MONGO_DB_USERN
                    -e MONGO_DB_PASSWORD="${MONGO_DB_PASSW
                    -e JWT_KEY='${JWT_KEY}' \\
                    -e RIOT_API_KEY='${RIOT_API_KEY}' \\
                    -e REDIS_PASSWORD='${REDIS_PASSWORD}'
                    -e REDIS_HOST_IP='${REDIS_HOST_IP}' \\
                    -e DOMAIN='${DOMAIN}' \\
                    -e FRONT_PORT='${FRONT_PORT}' \\
                    ${DOCKER_IMAGE}"
                    '''
                }
            }
        }
    }
}
```

- Frontend Pipeline

```
pipeline {
    agent any
    environment {
        DOCKER_IMAGE = 'sosuh97/lolwatcher-frontend:latest
        EC2_SERVER = 'ubuntu@k11a601.p.ssafy.io' // EC2 서
        REACT_APP_LOLWATCHER_API_URL = credentials('REACT_
        REACT_APP_CHAMPION_IMG_BASE_URL = credentials('REA
        REACT_APP_CHAMPION_SPLASH_IMG_BASE_URL = credentia
        REACT_APP_SUMMONER_ICON_URL = credentials('REACT_A
    }
    stages {
```

```
        stage('Clone Repository') {
            steps {
                script {
                    echo "Starting Clone Repository stage'

                    // Git 클론 수행
                    git branch: 'frontend-dev',
                        credentialsId: 'ssw_id',  // 새로
                        url: 'https://lab.ssafy.com/s11-f
                }
            }
        }
        stage('Build Docker Image') {
            steps {
                script {
                    docker.build("${DOCKER_IMAGE}",
                        "--build-arg REACT_APP_LOLWATCHER_
                        "--build-arg REACT_APP_CHAMPION_IM
                        "--build-arg REACT_APP_SUMMONER_IC
                        "--build-arg REACT_APP_CHAMPION_SP
                        "-f lolwatcher/Dockerfile lolwatch
                }
            }
        }
        stage('Push Docker Image') {
            steps {
                script {
                    docker.withRegistry('https://index.doc
                        docker.image("${DOCKER_IMAGE}").pu
                    }
                }
            }
        }
        stage('Deploy to EC2') {
            steps {
                sshagent(credentials: ['a601-ubuntu']) {
```

```
                    sh """
                    ssh -o StrictHostKeyChecking=no ${EC2_
                    'docker pull ${DOCKER_IMAGE} && \
                    docker stop lolwatcher-frontend || tru
                    docker rm lolwatcher-frontend || true
                    docker run -d --name lolwatcher-fronte
                    -e REACT_APP_LOLWATCHER_API_URL="${REA
                    -e REACT_APP_CHAMPION_IMG_BASE_URL="$
                    -e REACT_APP_CHAMPION_SPLASH_IMG_BASE_
                    -e REACT_APP_SUMMONER_ICON_URL="${REAC
                    ${DOCKER_IMAGE}'
                    """
                }
            }
        }
    }
}
```

- Data Pipeline

```
pipeline {
    agent any
    environment {
        DOCKER_IMAGE = 'sosuh97/lolwatcher-data:latest' //
        EC2_SERVER = 'ubuntu@k11a601.p.ssafy.io'      // E
        MONGO_DB_URL = credentials('MONGO_DB_URL')
        MONGO_DB_PORT = credentials('MONGO_DB_PORT')
        MONGO_DB_DATABASE = credentials('MONGO_DB_DATABASE
        MONGO_DB_USERNAME = credentials('MONGO_DB_USERNAME
        MONGO_DB_PASSWORD = credentials('MONGO_DB_PASSWORD
        CALCULATED_DATA = credentials('CALCULATED_DATA')
    }
    stages {
        stage('Clone Repository') {
            steps {
```

```
                script {
                    echo "Starting Clone Repository stage'
                    // Git 리포지토리 클론 수행
                    git branch: 'data-dev',
                        credentialsId: 'skc_id',
                        url: 'https://lab.ssafy.com/s11-f

                    // `pythonProject` 폴더 내부 확인
                    sh 'ls -al pythonProject'
                }
            }
        }

        stage('Build Docker Image') {
            steps {
                script {
                    echo "Starting Docker Build with Docke
                    // 빌드 컨텍스트를 `pythonProject`로 설정하
                    sh "docker build -t ${DOCKER_IMAGE} -f
                }
            }
        }

        stage('Push Docker Image') {
            steps {
                script {
                    docker.withRegistry('https://index.doc
                        docker.image("${DOCKER_IMAGE}").pu
                    }
                }
            }
        }

        stage('Deploy to EC2') {
            steps {
                sshagent(credentials: ['a601-ubuntu']) {
```

```
                    sh """
                    ssh -o StrictHostKeyChecking=no ${EC2_
                    'docker pull ${DOCKER_IMAGE} && \
                    docker stop lolwatcher-data || true &&
                    docker rm lolwatcher-data || true && \
                    docker run -d --name lolwatcher-data -
                    -e MONGO_DB_URL="${MONGO_DB_URL}"\
                    -e MONGO_DB_PORT="${MONGO_DB_PORT}"\
                    -e MONGO_DB_DATABASE="${MONGO_DB_DATAB
                    -e MONGO_DB_USERNAME="${MONGO_DB_USERN
                    -e MONGO_DB_PASSWORD="${MONGO_DB_PASSW
                    -e CALCULATED_DATA="${CALCULATED_DATA}
                    ${DOCKER_IMAGE}'
                    """
                }
            }
        }
    }
}
```

- Jenkins Credentials
  - frontend-docker : docker ID와 비밀번호
  - a601-ubuntu : 서버 pem 키
  - skc_access_token : Gitlab API 엑세스 토큰(Webhook)
  - skc_id : 프로젝트 관리자 Gitlab 아이디, 패스워드
  - DB_URL : MySQL 데이터베이스 URL
  - DB_USERNAME : MySQL 데이터베이스 username
  - DB_PASSWORD : MySQL 데이터베이스 password
  - JWT_KEY : JWT 키
  - RIOT_API_KEY : riot developer protal에서 매일 갱신하는 api 키

    - REDIS_PASSWORD
    - REDIS_HOST_IP
    - EC2_IP : ec2 ip(k~~ ssafy.io)
    - DOMAIN : 서비스의 도메인 url
    - FRONT_PORT
    - REACT_APP_AUTH_API_URL
    - MONGO_DB_URL
    - MONGO_DB_PORT
    - MONGO_DB_DATABASE
    - MONGO_DB_USERNAME
    - MONGO_DB_PASSWORD
    - REACT_APP_CHAMPION_SPLASH_IMG_BASE_URL
    - REACT_APP_CHAMPION_IMG_BASE_URL
    - DATA_SERVER_URL
    - CALCULATED_DATA
    - REACT_APP_LOLWATCHER_API_URL
    - REACT_APP_SUMMONER_ICON_URL

- Dockerfile
    - Backend Dockerfile

```
# 1단계: 빌드 단계
FROM gradle:7.6.0-jdk17 AS build

# 작업 디렉토리 설정
WORKDIR /app

# Gradle 캐시 활용을 위해 Gradle 설정 파일 복사
```

```
COPY build.gradle settings.gradle ./
COPY gradle ./gradle

# 종속성 다운로드 (소스 없이 종속성만 다운로드하여 캐시 활용)
RUN gradle build -x test --no-daemon || return 0

# 프로젝트 소스 복사 및 빌드
COPY . .
RUN gradle clean build -x test --no-daemon

# 2단계: 실행 단계
FROM eclipse-temurin:17-jre

# 작업 디렉토리 설정
WORKDIR /app

# 빌드 단계에서 생성된 JAR 파일을 복사
COPY --from=build /app/build/libs/*.jar app.jar

# 포트 설정 (Spring Boot 기본 포트)
EXPOSE 8080

# 애플리케이션 실행
ENTRYPOINT ["java", "-jar", "app.jar"]
```

- Frontend Dockerfile

```
# 1. Base image
FROM node:21 AS builder

# 2. Set working directory
WORKDIR /app

# 3. Copy package.json and install dependencies
COPY package*.json ./
```

```
RUN npm install

# 4. Define build arguments
ARG REACT_APP_LOLWATCHER_API_URL
ARG REACT_APP_CHAMPION_IMG_BASE_URL
ARG REACT_APP_CHAMPION_SPLASH_IMG_BASE_URL

# 5. Set environment variables for the build
ENV REACT_APP_LOLWATCHER_API_URL=$REACT_APP_LOLWATCHER_API_UR
ENV REACT_APP_CHAMPION_IMG_BASE_URL=$REACT_APP_CHAMPION_IMG_B
ENV REACT_APP_CHAMPION_SPLASH_IMG_BASE_URL=$REACT_APP_CHAMPIO

# 6. Copy source code and build
COPY . .
RUN npm run build

# 7. Deploying with a lightweight web server
FROM nginx:alpine
COPY --from=builder /app/build /usr/share/nginx/html

# 8. Expose port 80
EXPOSE 80
```

- Data Dockerfile

```
FROM python:3.9-slim

WORKDIR /app

COPY requirements.txt requirements.txt

RUN pip install -r requirements.txt

COPY . .
```

```
ENV FLASK_APP=run.py
ENV FLASK_ENV=production

EXPOSE 5000

CMD ["flask", "run", "--host=0.0.0.0"]
```

- application.yml
  - Backend - application.yml

```
server:
  port: 8080

spring:
  application:
    name: lolwatcher

  datasource:
    url: ${DB_URL}
    username: ${DB_USERNAME}
    password: ${DB_PASSWORD}
  data:
    redis:
      host: ${REDIS_HOST_IP}
      port: 6379
      password: ${REDIS_PASSWORD}  # 비밀번호가 설정된 경우
    mongodb:
      host: ${MONGO_DB_URL}
      port: ${MONGO_DB_PORT}
      database: ${MONGO_DB_DATABASE}
      username: ${MONGO_DB_USERNAME}
      password: ${MONGO_DB_PASSWORD}

  jwt:
```

```
    secret: ${JWT_KEY}
    # 유효기간은 15분
    access-token-expiration: 86400000 #300000
    refresh-token-expiration: 86400000 #36000000

  jpa:
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        dialect: org.hibernate.dialect.MySQL8Dialect  # 여기어


  config:
    import: optional:file:.env[.properties]
  profiles:
    active: local

analytic:
  url: ${DATA_SERVER_URL}

riot:
  lol:
    kr-url: https://kr.api.riotgames.com
    asia-url: https://asia.api.riotgames.com  #?? ??? ??? ???
    api-key: ${RIOT_API_KEY}
```

- DataCollect - application.yml

```
spring:
  application:
    name: lolwatcher-data-collect
  data:
```

```
    mongodb:
      host: ${DB_URL}
      port: ${DB_PORT}
      database: ${DB_DATABASE}
      username: ${DB_USERNAME}
      password: ${DB_PASSWORD}
  config:
    import: optional:file:.env[.properties]
  cloud:
    openfeign:
      client:
        config:
          default:
            connectTimeout: 5000
            readTimeout: 10000

riot:
  kr-url: https://kr.api.riotgames.com
  asia-url: https://asia.api.riotgames.com
  api-key: ${RIOT_KEY}
server:
  port: 8060
```

- /etc/nginx/sites-enabled/default

```
server {
    server_name lolwatcher.com;
#    root /usr/share/nginx/html;

    location / {

        try_files $uri $uri/ /index.html;

        proxy_pass http://localhost:3000; # 프론트엔드 컨테이너의 외
        proxy_set_header Host $host;
```

```
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location /api/ {
        rewrite ^/api(/.*)$ $1 break; # /api/ 접두사를 제거
        proxy_pass http://localhost:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_
        proxy_set_header X-Forwarded-Proto $scheme;
    }
    location /data/ {
        rewrite ^/data(/.*)$ $1 break; # /api/ 접두사를 제거
        proxy_pass http://localhost:5000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_
        proxy_set_header X-Forwarded-Proto $scheme;
    }

#    location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg)$ {
#        try_files $uri =404;
#    }

    location /riot.txt {
        proxy_pass http://172.18.0.2:80/riot.txt;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_
        proxy_set_header X-Forwarded-Proto $scheme;
    }


    location /static/ {
```

```
        proxy_pass http://172.18.0.2:80/static/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/lolwatcher.com/fullcha
    ssl_certificate_key /etc/letsencrypt/live/lolwatcher.com/pri
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

}

server {
    if ($host = lolwatcher.com) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    listen 80;
    server_name lolwatcher.com;
    return 404; # managed by Certbot
}
```

4. 배포 방법

1. ec2 설정
    a. 서버 시간 한국 표준시로 변경

```
sudo timedatectl set-timezone Asia/Seoul
```

b. 미러 서버를 카카오 서버로 변경

```
sudo sed -i 's/ap-northeast-2.ec2.archive.ubuntu.com/mirr
```

2. 도커 설치

a. 설치 전 필요한 패키지 설치

```
sudo apt-get -y install apt-transport-https ca-certificate
```

b. Docker에 대한 GPC Key 인증 진행

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg |
```

c. Docker 레포지토리 등록

```
sudo add-apt-repository "deb [arch=amd64] https://downloa
```

d. Docker 패키지 설치

```
sudo apt-get -y install docker-ce docker-ce-cli containerd
```

e. 일반 유저에게 권한 부여

```
sudo usermod -aG docker ubuntu

sudo service docker restart
```

3. Git 설치

```
sudo apt-get -y install git
```

4. Dockerfile 작성

4. Jenkins 컨테이너 설치
    a. Jenkins 이미지 받기

```
docker pull jenkins/jenkins:jdk17
```

    b. Jenkins 컨테이너 실행

```
docker run -d --restart always --env JENKINS_OPTS=--httpP
```

    c. Jenkins 환경 설정
        i. 하위 디렉토리 생성

```
sudo mkdir /jenkins/update-center-rootCAs
```

        ii. CA 파일 다운로드

```
sudo wget https://cdn.jsdelivr.net/gh/lework/jenkins-u
```

    d. Jenkins 접속
    e. 플러그인 세팅

```
# 설치한 플러그인

# ssh 커맨드 입력에 사용
SSH Agent

# docker 이미지 생성에 사용
Docker
Docker Commons
Docker Pipeline
Docker API

# 웹훅을 통해 브랜치 merge request 이벤트 발생시 Jenkins 자동 빌드
Generic Webhook Trigger
```

```
# 타사 레포지토리 이용시 사용 (GitLab, Github 등)
GitLab
GitLab API
GitLab Authentication
GitHub Authentication

# Node.js 빌드시 사용
NodeJS

Gradle ## (version 8.10.2)
```

f. pipeline 생성 및 Gitlab Webhook 지정
   - Gitlab Webhook Url 형식 = http://jenkins-server-url/project/jenkins-item-name
   - backend와 frontend 개별적으로 파이프라인 생성하고 docker-network를 통해 컨테이너 간 연결

g. Docker Hub Credential, Ubuntu Credential 추가

h. pipeline script 작성, Credential 설정 및 배포 테스트

5. nginx 설치 및 SSL / 프록시 설정

   a. Nginx 설치

```
sudo apt-get -y install nginx
```

   b. SSL 설정

      i. CertBot 설치

```
sudo snap install --classic certbot

sudo apt-add-repository -r ppa:certbot/certbot

sudo apt-get -y install python3-certbot-nginx
```

ii. SSL 인증서 발급

```
sudo certbot --nginx -d yourdomain.com
```

iii. 공인 IP 주소 확인

```
curl http://checkip.amazonaws.com
```

iv. DNS 설정(가비아) : A 레코드가 EC2 서버의 공인 IP 주소를 가리키고 있는지 다시 한번 확인

→ DNS checker에서 도메인 주소 검색

v. certbot으로 SSL 인증서 발급

```
sudo certbot --nginx -d lolwatcher.com
```

7. 빌드 및 배포

1. 빌드 (VScode)

```
docker build -t your-dockerID/container-name .
```

2. Dockerhub에 Push (VScode)

```
docker push your-dockerID/container-name
```

3. Dockerhub에서 Pull (MobaXterm)

```
docker pull your-dockerID/container-name:latest
```

4. 컨테이너 실행 (MobaXterm)

```
docker run -d -p (서버 포트번호):(컨테이너 포트번호) your-doc
```

2. 보조 서버

1. 컨테이너 정보

- DataCollect : 데이터 수집 컨테이너
  - 8060:8060

2. 주요 흐름

- 데이터 수집 만을 목적으로 하는 보조 서버입니다.
- 첫 실행 이후 큰 수정이 필요 없으므로 수동으로 배포합니다.

3. 개발 환경

1. 프로젝트 기술 스택

- Docker
  - Docker : 27.3.1
- DataCollect
  - Java : 17
  - SpringBootFrameWork : 3.3.5
  - Spring Dependency-Management : 1.1.6

4. 설정 파일

- DataCollect - application.yml

```
spring:
  application:
    name: lolwatcher-data-collect
  data:
    mongodb:
      host: ${DB_URL}
      port: ${DB_PORT}
      database: ${DB_DATABASE}
      username: ${DB_USERNAME}
      password: ${DB_PASSWORD}
  config:
    import: optional:file:.env[.properties]
  cloud:
    openfeign:
      client:
        config:
          default:
            connectTimeout: 5000
            readTimeout: 10000

riot:
  kr-url: https://kr.api.riotgames.com
  asia-url: https://asia.api.riotgames.com
  api-key: ${RIOT_API_KEY}
server:
  port: 8060
```

- DataCollect - Dockerfile

```
# 1단계: 빌드 단계
FROM gradle:8.10.2-jdk17 AS build

# 작업 디렉토리 설정
```

```
WORKDIR /app

# Gradle 캐시 활용을 위해 Gradle 설정 파일 복사
COPY build.gradle settings.gradle ./
COPY gradle ./gradle

# 종속성 다운로드 (소스 없이 종속성만 다운로드하여 캐시 활용)
RUN gradle build -x test --no-daemon || return 0

# 프로젝트 소스 복사 및 빌드
COPY . .
RUN gradle clean build -x test --no-daemon

# 2단계: 실행 단계
FROM eclipse-temurin:17-jre

# 작업 디렉토리 설정
WORKDIR /app

# 환경 변수 설정
ENV RIOT_API_KEY="라이엇 개발자 키"
ENV DB_PORT=(MONGODB 포트 번호)
ENV DB_URL="메인 서버 IP"
ENV DB_DATABASE="MongoDB 데이터베이스 이름"
ENV DB_USERNAME="MongoDB 유저 이름"
ENV DB_PASSWORD="MongoDB 비밀번호 이름"

# 빌드 단계에서 생성된 JAR 파일을 복사
COPY --from=build /app/build/libs/*.jar app.jar

# 포트 설정 (Spring Boot 기본 포트)
EXPOSE 8060

# 애플리케이션 실행
ENTRYPOINT ["java", "-jar", "app.jar"]
```