포팅매뉴얼

1. 개발환경

- Server : Ubuntu 20.04.6 LTS
- JDK : OpenJDK17
- Nginx : nginx/1.18.0
- MySQL : 9.0.1
- Redis : 7.4.0
- IntelliJ : 2024.1.4
- Android Studio : 2024.1.1

설정 파일 및 환경변수 정보

Spring

▼ application.yml

```
spring:
  data:
    redis:
      host: 172.17.0.3
      port: 6379
  servlet:
    multipart:
      max-file-size: 15MB
      max-request-size: 20MB
  output:
    ansi:
      enabled: always
  application:
    name: orm-backend
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://172.17.0.2:3306/orm?useSSL=false&serverTimezone=Asia/Seoul&characterEncoding=UTF-8&allowPublicKeyRetrieval=true
    username: newuser
    password: ssafy
  jpa:
    open-in-view: true
    hibernate:
      ddl-auto: validate
    show-sql: false
    properties:
      hibernate:
        format_sql: false
        dialect: org.hibernate.dialect.MySQL8Dialect

schedule:
  cron:
    delete-club: 0 0 1 * * *

jwt:
  salt: ORM-SALT-VALUE-40932999-071e-4368-a21f-6aa4b5fa16b0
  access-token:
    expiretime: 360000000000
  refresh-token:
    expiretime: 2592000000

kakao:
  redirect-uri: http://i11a709.p.ssafy.io/users/login/kakao/auth
  app-key: 41c4de00b40988807f39bac727167229
  example-refreshToken: oj21gkju1JX4792DcHmnQLiS7vU_4NEEAAAAAgo8JCEAAAGQyY4r3dEMsmlHt4Ko

orm:
  default-image-src: https://i11a709.p.ssafy.io/files/orm/default_image/img_orm_1000.png
  header:
    auth: Authorization
  sftp:
    host: I11A709.p.ssafy.io
    port: 22
    user: ubuntu
    pem-file-path: /I11A709T.pem
    remote-upload-dir: /home/upload/orm/
    remote-access-dir: /files/orm/

firebase:
  api-url: https://fcm.googleapis.com/v1/projects/orm-ssafy/messages:send
  config-path: orm-ssafy-firebase-adminsdk-r5bcu-b5d1ad1ab6.json
```

▼ **firebase**

```
{
  "type": "service_account",
  "project_id": "orm-ssafy",
  "private_key_id": "b5d1ad1ab69f299eab04c152d0e38de6edb3c044",
  "private_key": "-----BEGIN PRIVATE KEY-----\nMIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQDFP06WviIynkwL\n3rYs5MfMZph7GzMSgFkq8
  "client_email": "firebase-adminsdk-r5bcu@orm-ssafy.iam.gserviceaccount.com",
  "client_id": "105469365458414193253",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/firebase-adminsdk-r5bcu%40orm-ssafy.iam.gserviceaccount.c
  "universe_domain": "googleapis.com"
}
```

- **pem 키를 resources 폴더 안에 위치시켜야 합니다.**

2. nginx 설치

▼ **nginx 설치 절차**

```
# 관리자 권한 획득
sudo su
# nginx 설치
apt-get install nginx
# nginx 프로필 허용
ufw allow 'Nginx Full'
ufw allow 'Nginx HTTP'
ufw allow 'Nginx HTTPS'
# 방화벽 변경사항 리로드
ufw reload
# 방화벽 상태 확인
ufw status
```

▼ **nginx 설정 (etc/nginx/sites-available/default)**

managed by certbot ssl 발급 받는 과정에서 자동 적용됩니다.

```
server {
        server_name i11A709.p.ssafy.io;

        include /etc/nginx/conf.d/service-url.inc;

        location /files/orm/ {
                alias /home/upload/orm/;
                # as directory, then fall back to displaying a 404.
                try_files $uri $uri/ =404;
        }
        location / {
                proxy_pass $service_url;
                proxy_set_header Host $host;
                        proxy_set_header X-Real-IP $remote_addr;
                proxy_read_timeout 600;
                proxy_send_timeout 600;
                proxy_connect_timeout 600;
                send_timeout 600;
        }

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/i11a709.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/i11a709.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 ($host = i11a709.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


        listen 80 default_server;
        listen [::]:80 default_server;
        server_name i11A709.p.ssafy.io;
    return 404; # managed by Certbot
}
```

3. 인증서 발급 (ssl)

▼ **certbot 설치 및 적용**

```
# 관리자 권한 획득
sudo su
# certbot 설치
apt-get install certbot
apt-get install python3-certbot-nginx
certbot --nginx
```

```
# 이후 이메일 / 공유 허용 / 도메인 등록등의 절차를 진행
```

4. Docker 설치

▼ **Docker 및 Docker Compose 설치 절차**

1. **필요한 패키지 설치**

```
apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common
```

2. **Docker의 공식 GPG키를 추가 & Docker의 공식 apt 저장소를 추가 & 시스템 패키지 업데이트**

```
# Add Docker's official GPG key:
apt-get update
apt-get install ca-certificates curl
install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
apt-get update
```

3. **Docker 설치**

```
apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

4. **Docker-Compose 설치**

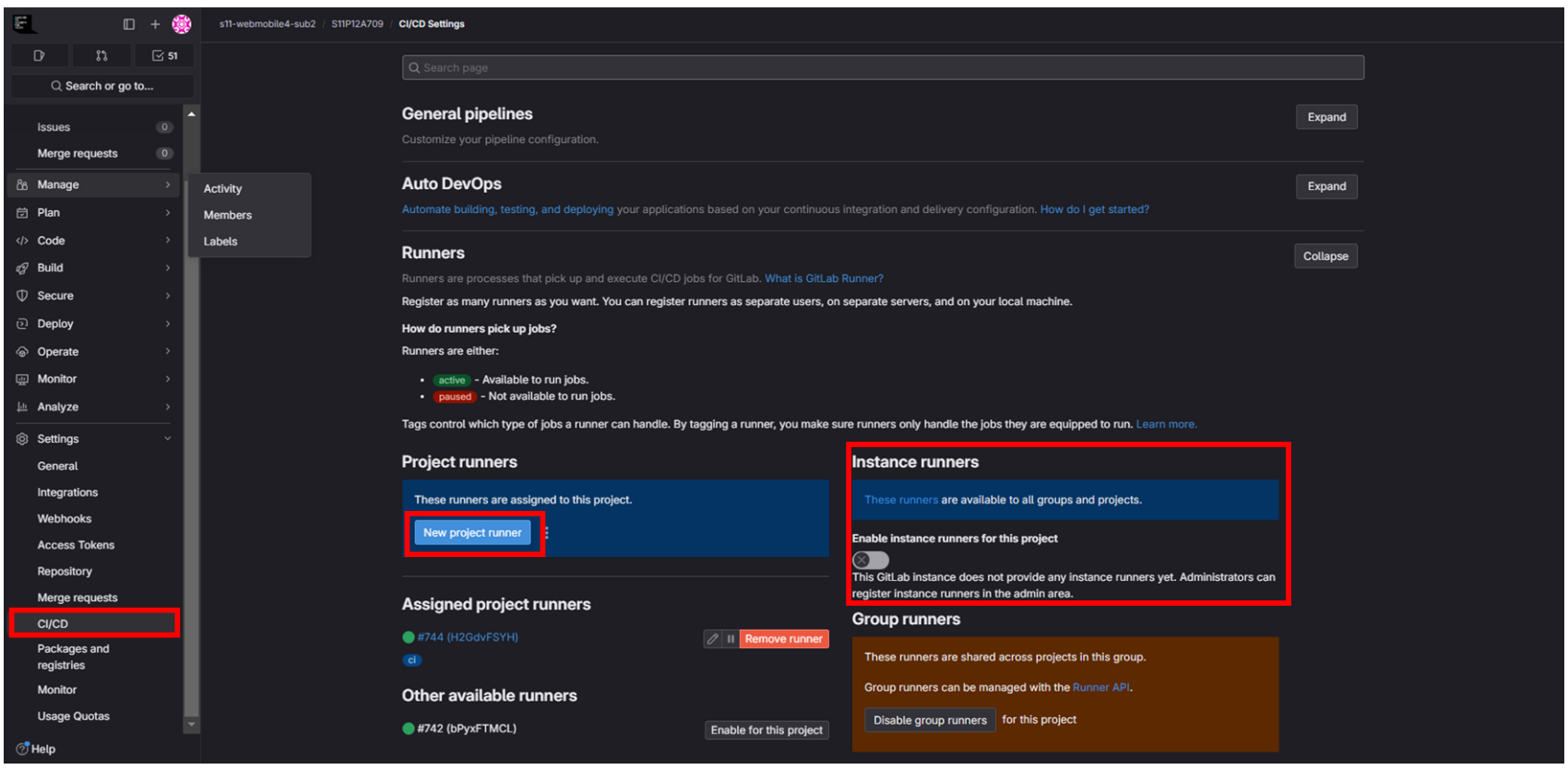
```
apt-get install docker-compose-plugin
```

5. gitlab-runner 컨테이너 생성

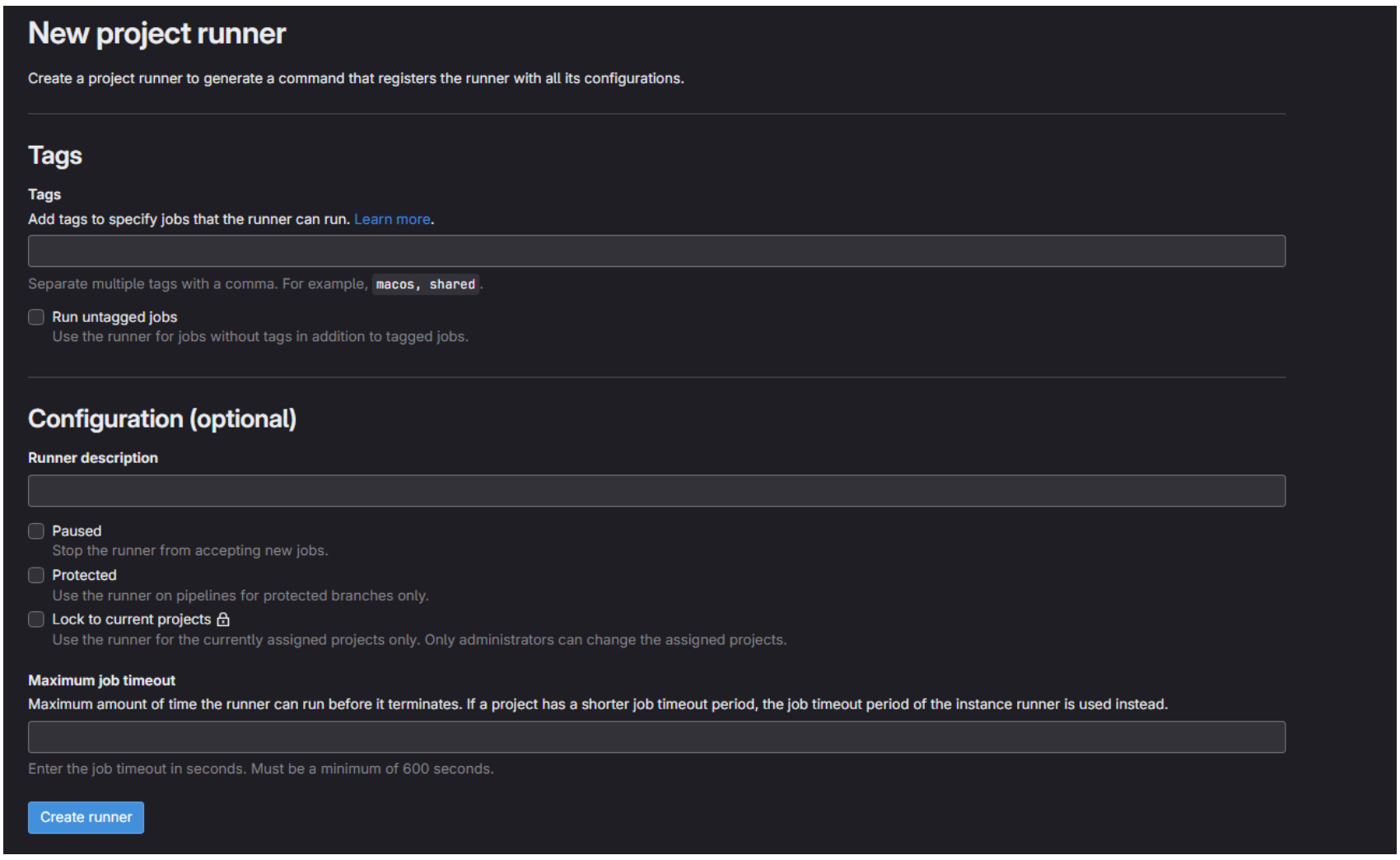
▼ **gitlab-runner 설치 및 실행**

```
# 이미지 받아오고
docker pull gitlab/gitlab-runner
# 컨테이너 구동
docker run -d --name gitlab-runner --restart always -v /var/run/docker.sock:/var/run/docker.sock -v gitlab-runner-config:/etc/gitlab-r
# 컨테이너 들어가서 작업 진행
docker container exec -it gitlab runner bash
```

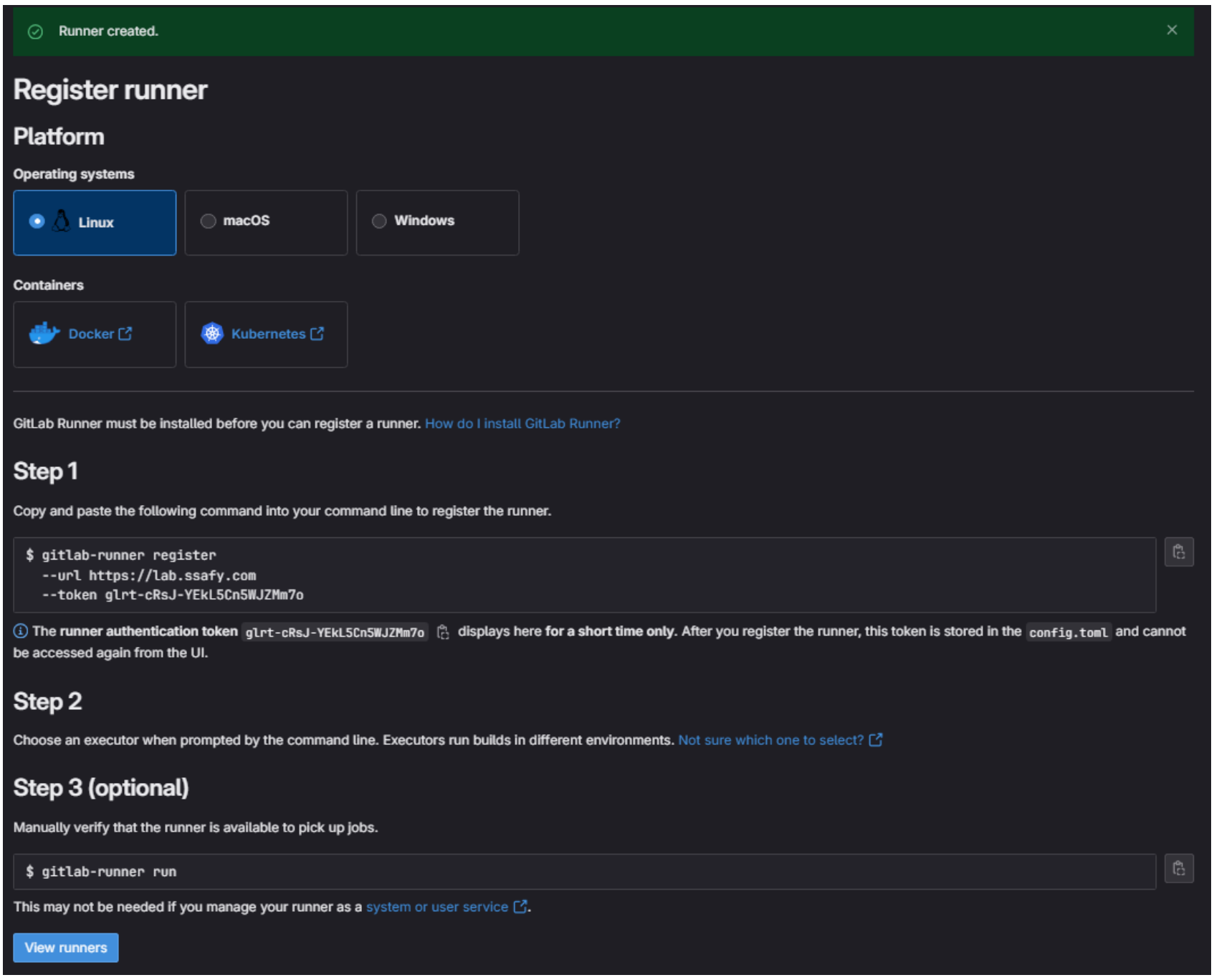
runner 등록 이전에 gitlab CI/CD 에서 project runner 설정



- **Instance Runner 비활성화**
- **New project Runner 등록**



- Tag : Pipeline을 누가 수행할 지 결정하고자 한다면 Tags를 설정합니다.
- 현재 ci로 tag를 등록하여 매 과정마다 어떤 runner 가 작업을 진행할 것인지 결정해 두었습니다.



- Step1, Step2에 적힌 내용을 아래 container bash 창에서 입력합니다.

▼ **gitlab-runner container 상에서 입력**

```
# gitlab-runner 등록
gitlab-runner register
# url 입력
https://{제시된 domain}
# token 입력
{제시된 token}
```

6. db mysql 및 redis 컨테이너 생성

▼ **MySQL 컨테이너 생성**

```
docker run -d --name mydb -p 3307:3306 -e MYSQL_ROOT_PASSWORD=ssafy MYSQL_USER=newuser MYSQL_PASSWORD=ssafy mysql:latest
```

▼ **Redis 컨테이너 생성**

```
docker run -d --name myredis -p 6379:6379 -v /home/redis.conf:/usr/local/etc/redis/redis.conf redis redis-server /usr/local/etc/redis/
```

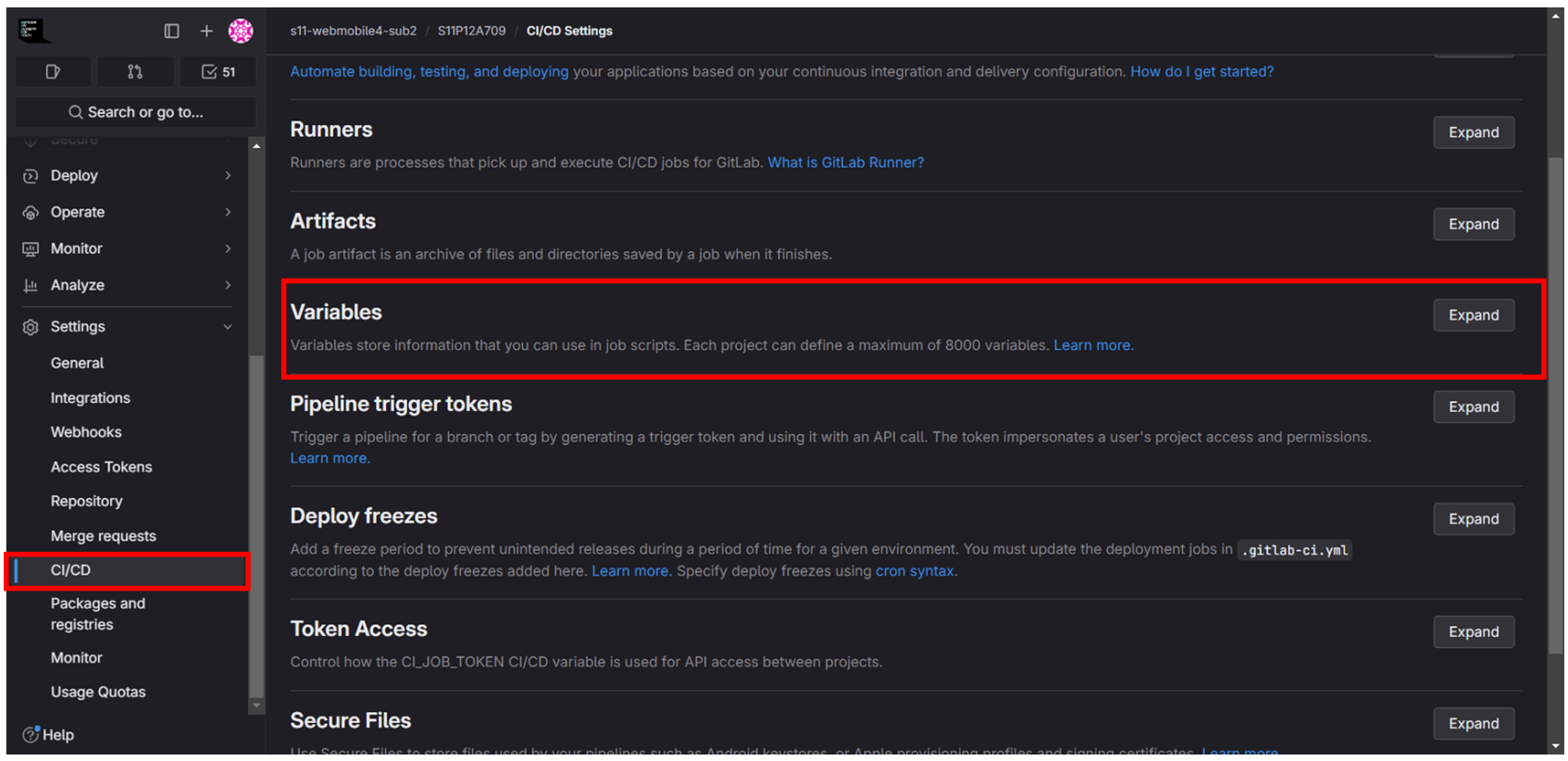
7. 깃랩CI/CD

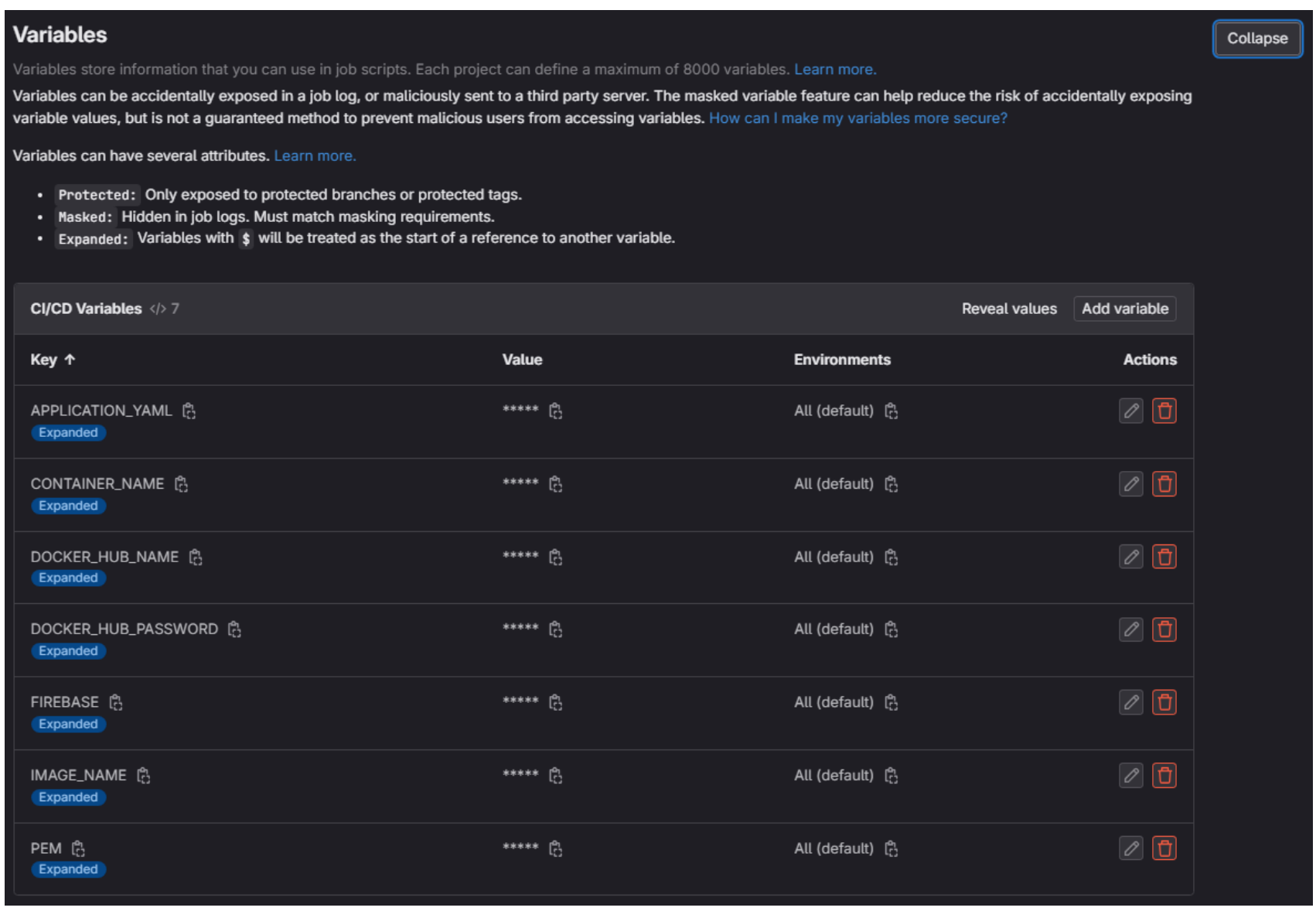
▼ **gitlab CI/CD 파이프라인 스크립트**

```
stages:
  - build
  - package
  - deploy

build-job:
  tags:
    - ci
  image : gradle:8.8.0-jdk17
  stage: build
  script:
    - echo "✨ application-dev.yml 파일을 orm-backend/src/main/resources 경로에 저장"
    - |
      cat <<EOF > orm-backend/src/main/resources/application-dev.yml
      $APPLICATION_YAML
      EOF

    - echo "✨ pem 파일을 orm-backend/src/main/resources 경로 저장"
    - |
      cat <<EOF > orm-backend/src/main/resources/I11A709T.pem
      $PEM
      EOF

    - echo "✨ json 파일을 orm-backend/src/main/resources 경로에 저장"
    - |
      cat <<EOF > orm-backend/src/main/resources/orm-ssafy-firebase-adminsdk-r5bcu-b5d1ad1ab6.json
      $FIREBASE
      EOF

    - cd orm-backend
    - echo "Compiling 진행"
    - chmod +x ./gradlew  # gradlew에 실행 권한 부여
    - ./gradlew clean
    - ./gradlew build --warning-mode all

    - echo "Compile 완료"

  artifacts:
    paths:
      - orm-backend/build/libs/*.jar
    expire_in: 1days
  only:
    - backend

package:
  tags:
    - ci
  image: docker:latest
  stage: package
  before_script:
  # Docker Hub 로그인
  - unset DOCKER_HOST
  - echo $DOCKER_HUB_PASSWORD | docker login -u $DOCKER_HUB_NAME --password-stdin
  script:
    - docker build --platform linux/amd64/v3 -t $DOCKER_HUB_NAME/$IMAGE_NAME orm-backend/.   # Dockerfile로 build
    - echo "build complete"
    - docker push $DOCKER_HUB_NAME/$IMAGE_NAME

  # JOB이 수행될 branch 설정 (설정된 branch에 push가 발생될 시 JOB 수행)
  only:
    - backend

deploy:
  tags:
    - ci
  image: docker:latest
  stage: deploy
  before_script:
    # Docker Hub 로그인
    - unset DOCKER_HOST
    - echo $DOCKER_HUB_PASSWORD | docker login -u $DOCKER_HUB_NAME --password-stdin
  script:
    - echo "✨ 컨테이너 구동 중단 및 삭제 (해당 컨테이너가 이미 존재한다면)"
    - docker stop $CONTAINER_NAME || true
    - docker rm $CONTAINER_NAME || true
    - echo "✨ 도커 이미지를 도커 허브에서 pull"
    - docker pull $DOCKER_HUB_NAME/$IMAGE_NAME
    - echo "✨ 컨테이너 구동 진행 "
```

```
    - docker run -d --name $CONTAINER_NAME -p 8080:8080 -e SPRING_PROFILE=dev $DOCKER_HUB_NAME/$IMAGE_NAME
  needs:
    - ["build-job", "package"]
  # script가 실행된 후 수행 될 script
  after_script:
    - docker logout
  only:
    - backend
```

▼ **Variables 설정**





APPLICATION_YAML : Spring의 application.yml

CONTAINER_NAME : `orm-backend`

DOCKER_HUB_NAME : `chanjinlee`

DOCKER_HUB_PASSWORD : docker hub에서 token을 발급 받아 기입 혹은 password를 기입

FIREBASE : Spring의 firebase.json

IMAGE_NAME : `test-backend:latest`

PEM : EC2 PEM