# base-自动化部署

**前言：https://github.com/lixiaod666/ci 项目中统一管理gitlab-runner自动化部署的配置文件，本项目以下简称gitlabci-service**

### 1. 在部署服务器上安装gitlab-runner

yum install gitlab-runner

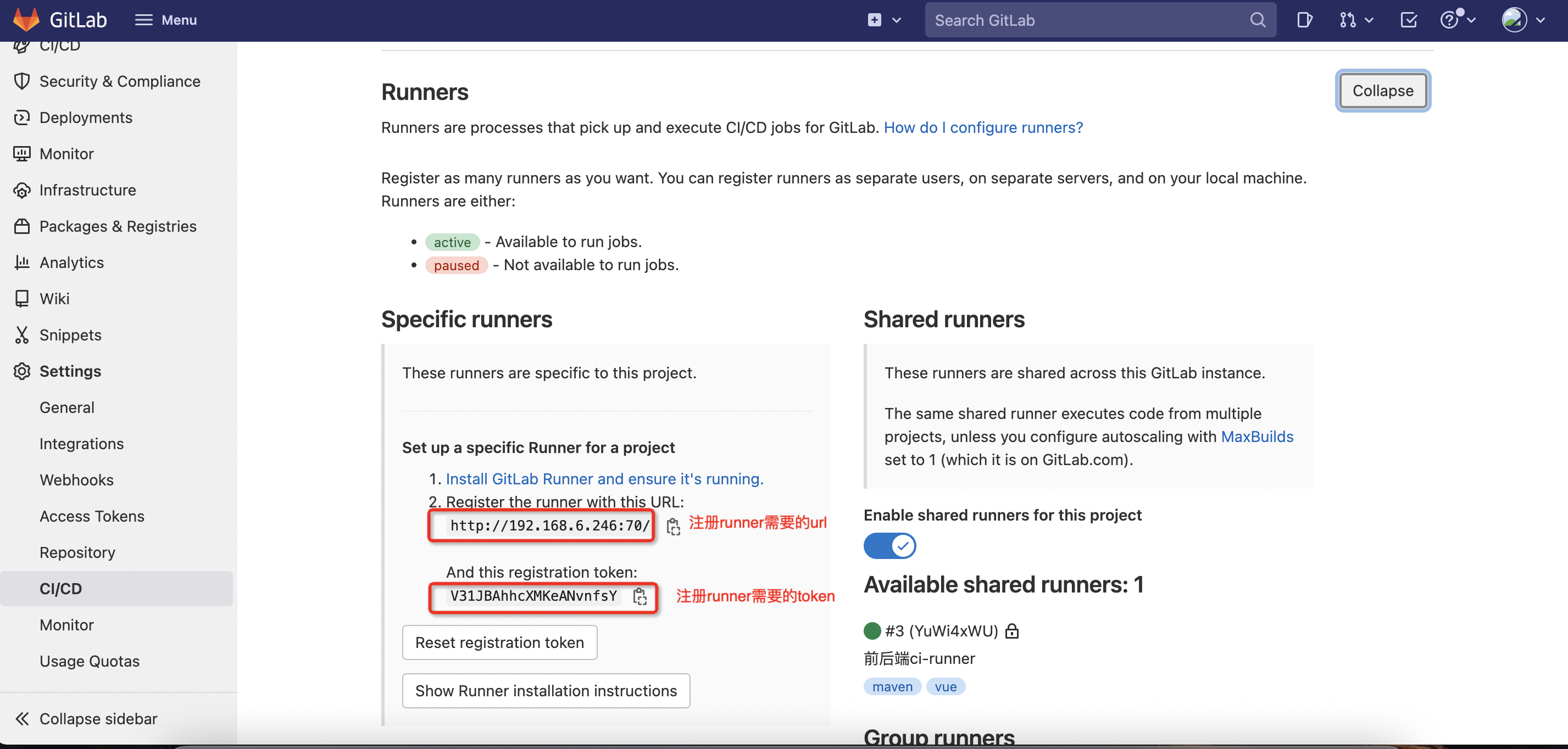
### 2. 向GitLab-CI注册gitlab-runner

* 找到你的gitlab上存放的项目，找到settings–>CI/CD–>Runner



* 部署服务器上进行runner注册

gitlab-runner register



1. url：gitlab所在的测试服务器地址
2. token：项目的token，用于关联runner和项目
3. name：runner的名字，用于区分runner
4. tags：用于匹配任务（jobs）和执行任务的设备（runners），此处不填直接回车时，默认Can run untagged jobs值为true

### 3. 准备gitlab-ci.yml文件

在**gitlabci-service**根目录下：

#### 3.1在ci目录下创建.gitlab-ci.yml文件

include:  
 - project: "ci-devops/gitlabci-service"  
 ref: main  
 file: "be/java\_build.yml"  
 - project: "ci-devops/gitlabci-service"  
 ref: main  
 file: "be/dingTalk\_ci.yml"  
  
variables:  
 TIMESTAMP: 'date -d $CI\_COMMIT\_TIMESTAMP +%Y%m%d%H%M'  
 BUILD\_SHELL: "mvn clean package -T 2C -Dmaven.test.skip=true -Dmaven.compile.fork=true"  
 ARTIFACT\_REPOSITORY\_NAME: "java"  
 TARGET\_FILE\_PATH: "$CI\_PROJECT\_NAMESPACE/$CI\_PROJECT\_NAME/$CI\_COMMIT\_REF\_NAME-$CI\_COMMIT\_SHORT\_SHA-$CI\_PIPELINE\_ID"  
 TARGET\_ARTIFACT\_NAME: "$CI\_PROJECT\_NAME-$CI\_COMMIT\_REF\_NAME"  
 #IMAGE\_TAG: "$HARBOR\_URL/$CI\_PROJECT\_NAME/$CI\_JOB\_NAME:`$TIMESTAMP`"  
  
stages:  
 - build\_image  
 - dingTalk  
  
adminservice:  
 stage: build\_image  
 extends: .data-platform-common-template  
  
analysisdataimpservice:  
 stage: build\_image  
 extends: .data-platform-common-template  
  
auto-deploy-demo:  
 stage: build\_image  
 extends: .data-platform-common-template  
  
dingTalk:  
 rules:  
 - if: '$CI\_COMMIT\_REF\_NAME == "dev/auto-deploy"'  
 stage: dingTalk  
 extends: .dingTalk

#### 3.2.在buildfile目录下创建${projectName}.Dockerfile文件，用于docker构建镜像

需要将配置中为demo2的地方替换为您的项目名

FROM openjdk:17  
WORKDIR /app  
COPY demo2/target/\*.jar /app/demo2.jar  
EXPOSE 8090  
CMD ["java", "-jar", "demo2.jar"]

#### 3.3 在deployfile目录下创建deploy-${projectName}.yml文件，用于执行k8s相关命令

需要将配置中为demo2的地方替换为您的项目名

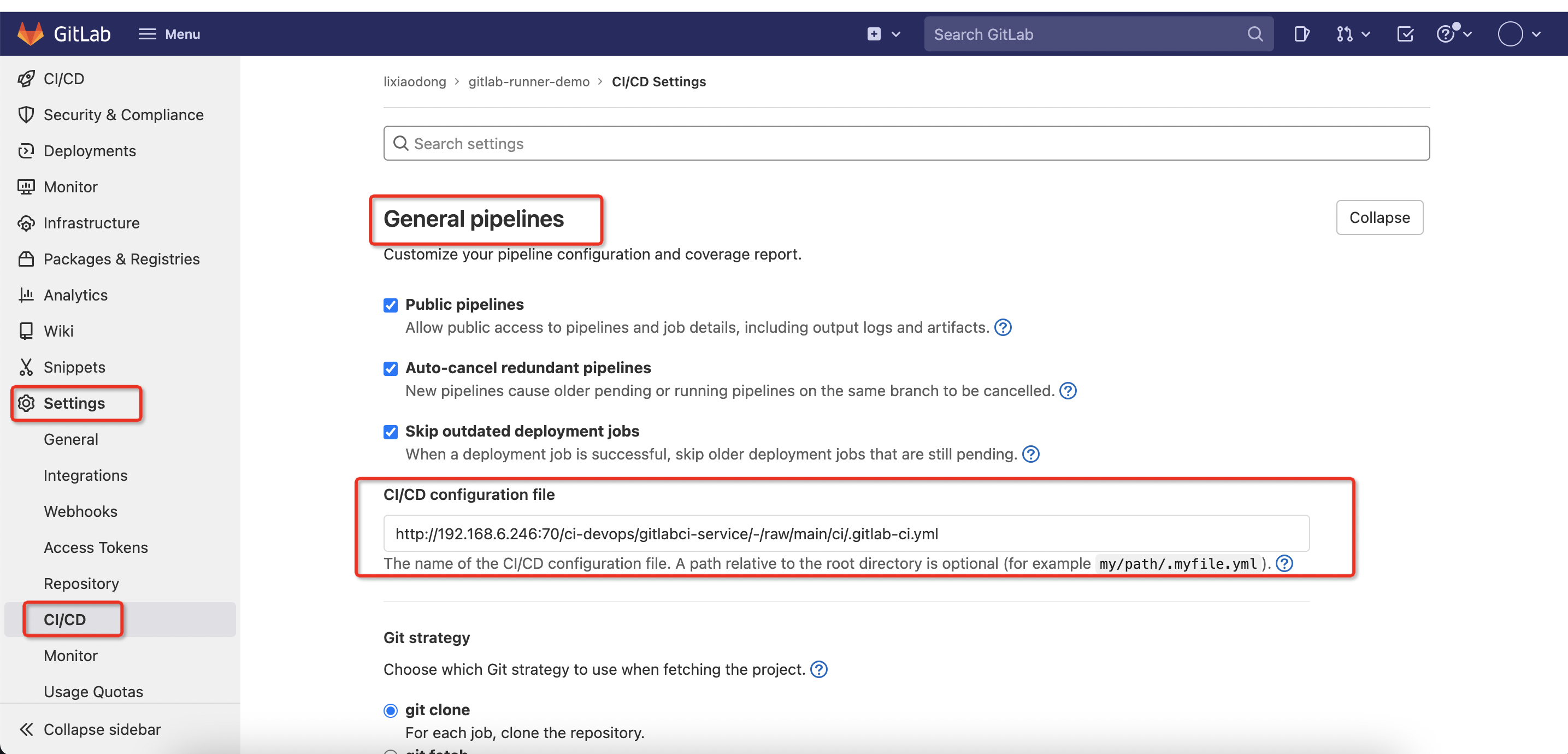
---  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
 labels:  
 app: demo2  
 name: demo2  
 namespace: default  
spec:  
 replicas: 1  
 selector:  
 matchLabels:  
 app: demo2  
 template:  
 metadata:  
 labels:  
 app: demo2  
 spec:  
 containers:  
 - image: APP\_IMAGE  
 imagePullPolicy: IfNotPresent  
 name: demo2  
 ports:  
 - containerPort: 8090  
 protocol: TCP  
---  
apiVersion: v1  
kind: Service  
metadata:  
 labels:  
 app: demo2  
 name: demo2  
 #namespace: NAMESPACE  
spec:  
 ports:  
 - port: 8090  
 protocol: TCP  
 targetPort: 8090  
 selector:  
 app: demo2  
 sessionAffinity: None  
 type: NodePort

#### 3.4 在ci目录下创建java\_build.yml文件，用于处理在自动化部署中，打包、推包等流程

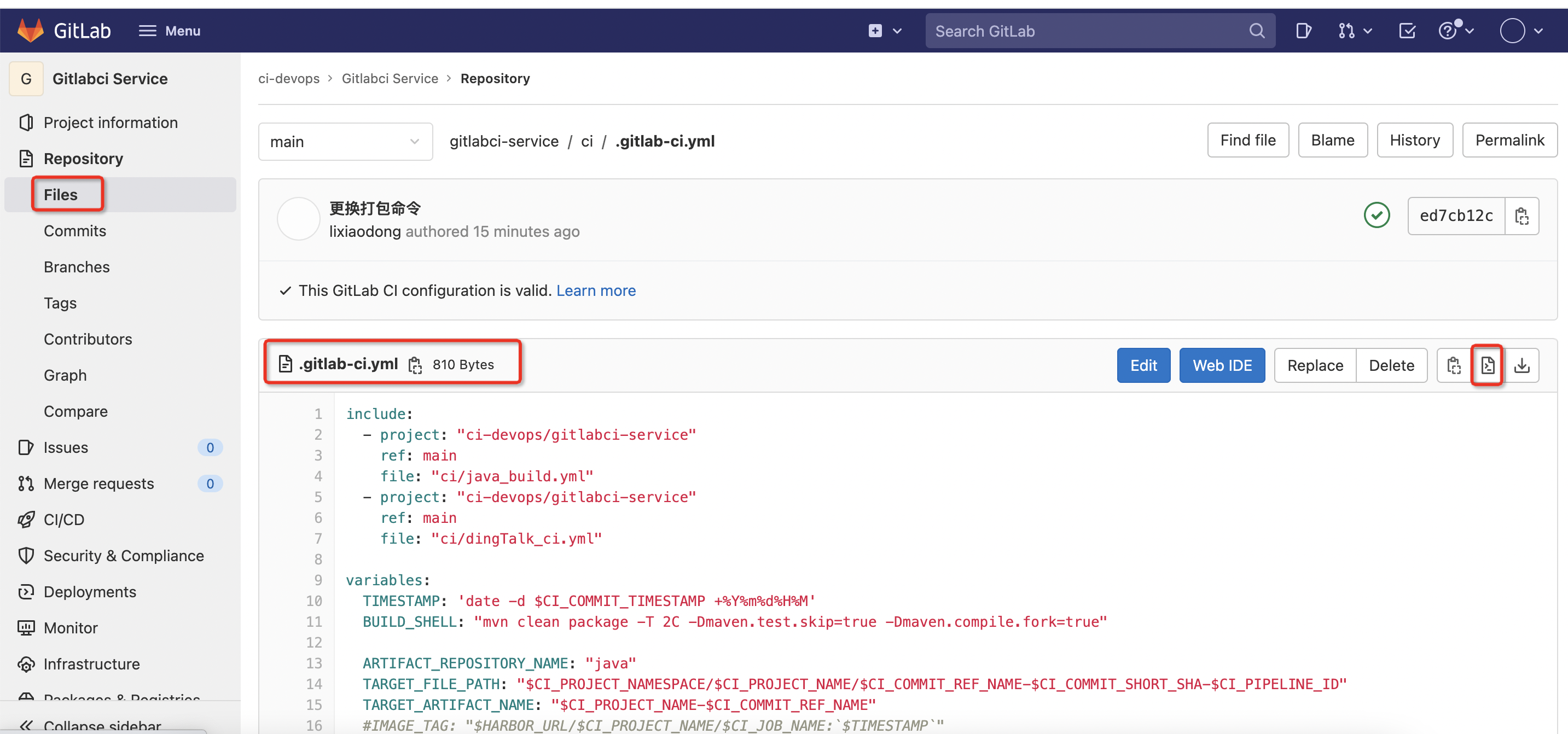
.data-platform-common-template:  
 stage: build\_image  
 tags:  
 # 指定runner  
 - server  
 rules:  
 # 指定分支  
 - if: '$CI\_COMMIT\_REF\_NAME == "dev/auto-deploy"'  
 changes:  
 - $CI\_JOB\_NAME/\*\*/\*  
 script:  
 - |  
 - echo "====================进入$CI\_BUILD\_NAME======================="  
 - echo "========定义变量:$CI\_JOB\_NAME========="  
 #打印所有的环境变量，用于调试  
 - env  
 - echo "====================编译======================="  
 - java -version  
 - mvn -version  
 # 如果是多层目录，需要进入当前项目所在目录  
 - cd $CI\_JOB\_NAME/  
 # - mvn clean  
 # - mvn compile  
 - echo "====================开始打包======================="  
 - $BUILD\_SHELL  
 - echo "====================打包完毕======================="  
 - cd target/  
 - ls  
 - pwd  
 - echo "=================================================="  
 # 退出到根目录  
 - cd ..  
 - cd ..  
 - ls  
 # 拉取Dockerfile文件，用于docker build  
 - wget -O Dockerfile http://192.168.6.246:70/ci-devops/gitlabci-service/-/raw/main/buildfile/$CI\_JOB\_NAME.Dockerfile  
 # 打印Dockerfile  
 - ls && cat Dockerfile  
 # 构建镜像  
 - docker build . -t $HARBOR\_URL/demo/$CI\_JOB\_NAME:$($TIMESTAMP)  
 # 推送镜像至仓库  
 - docker push $HARBOR\_URL/demo/$CI\_JOB\_NAME:$($TIMESTAMP)  
 - docker rmi $HARBOR\_URL/demo/$CI\_JOB\_NAME:$($TIMESTAMP)  
 - IMAGES=$HARBOR\_URL/demo/$CI\_JOB\_NAME:$($TIMESTAMP)  
 - echo $IMAGES  
 # 拉去deploy.yml，用于执行k8s相关命令  
 - wget -O deploy.yml http://192.168.6.246:70/ci-devops/gitlabci-service/-/raw/main/deployfile/deploy\_$CI\_JOB\_NAME.yml  
 - ls  
 # 替换变量  
 - sed -i "s@APP\_IMAGE@$IMAGES@g" deploy.yml  
 - cat deploy.yml  
 # 暂时关闭  
 - kubectl apply -f deploy.yml

### 4. 在需要加入自动化部署的gitlab项目下引入.gitlab-ci.yml文件

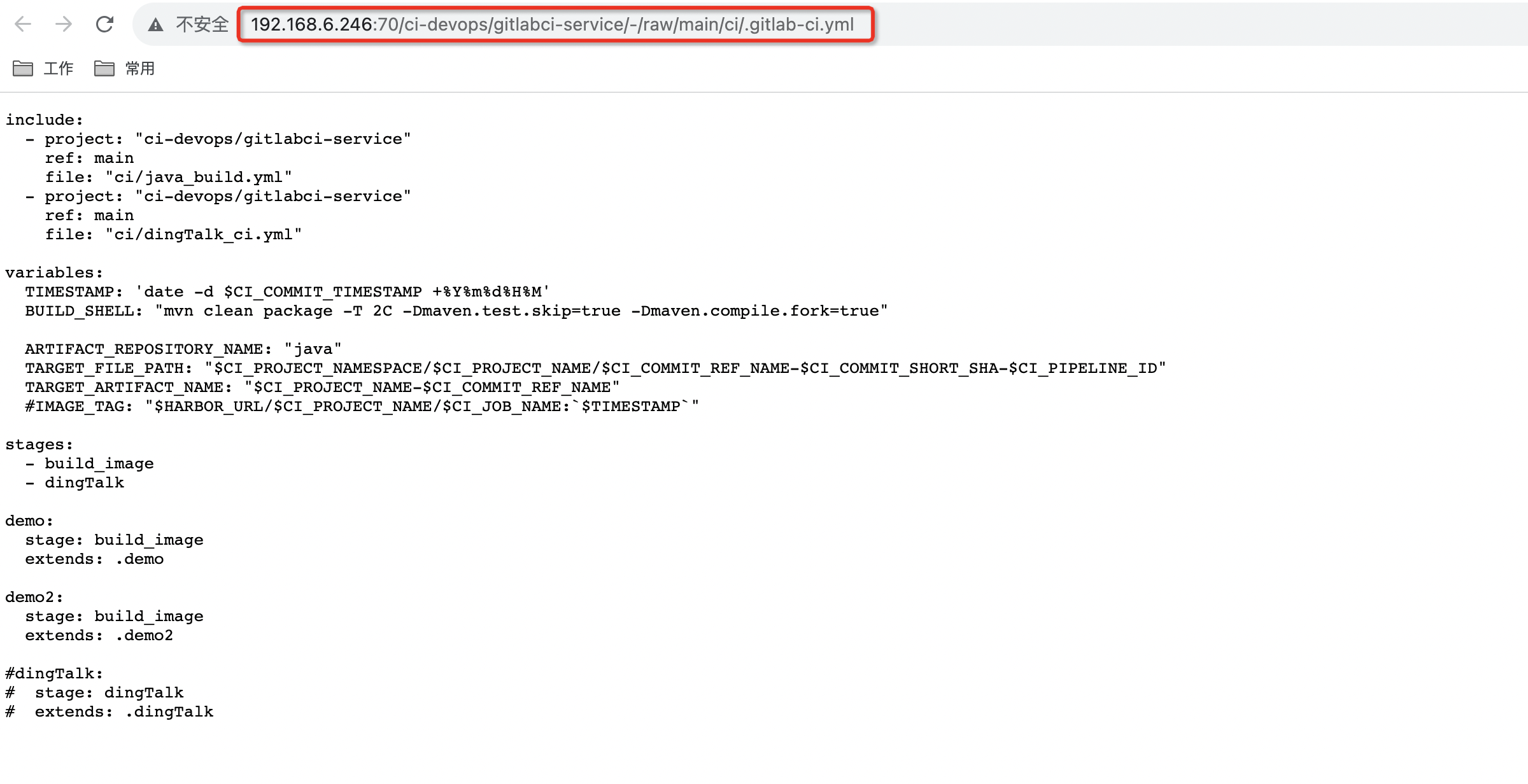
#### 4.1Settings-CI\_CD/-General pipelines选项中引入.gitlab-ci.yml文件



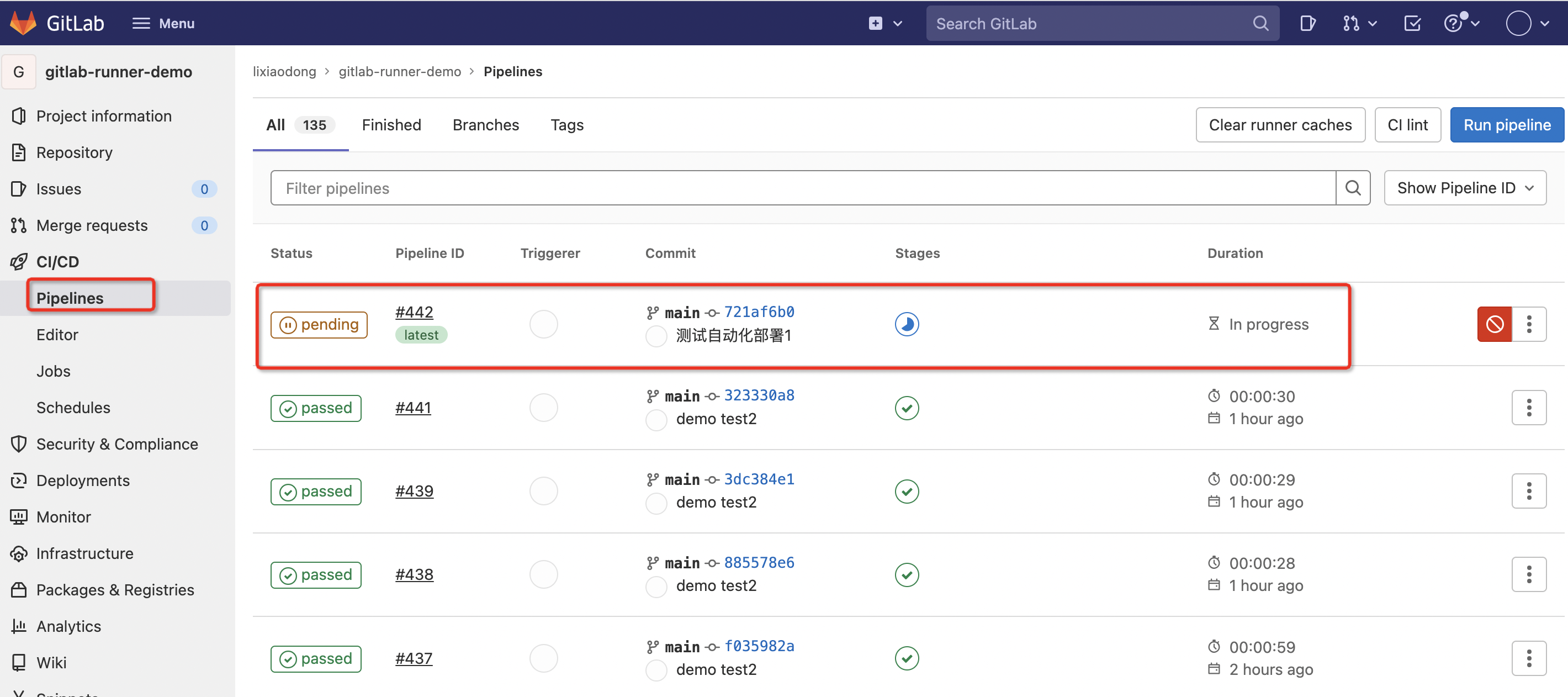
#### 4.2 进入gitlabci-service项目 FIle目录下



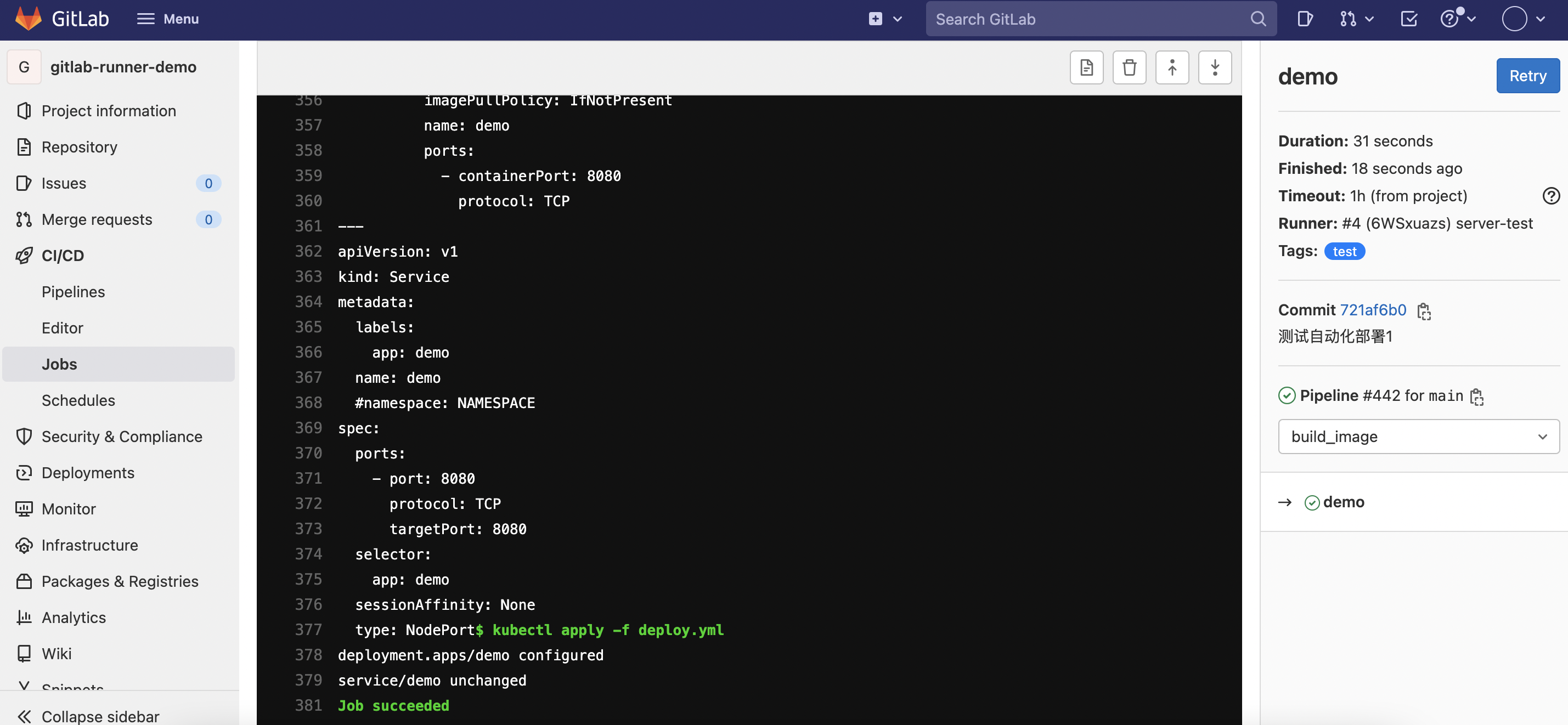
#### 4.3 获取.gitlab-ci.yml文件地址



#### 4.4 提交代码后，会触发pipeline



构建过程



#### 4.5 构建完成后发送钉钉通知

