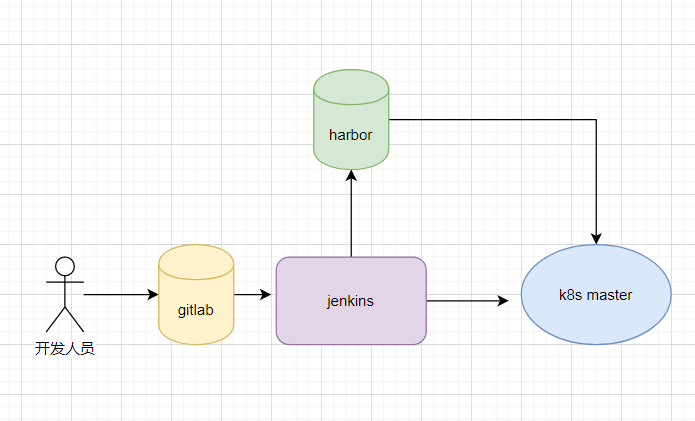
**Springboot+jenkins+gitlab+k8s持续集成**

前提：

1. 已安装jenkins
2. 已安装kubernetes集群
3. 已安装gitlab仓库
4. 已安装harbor仓库

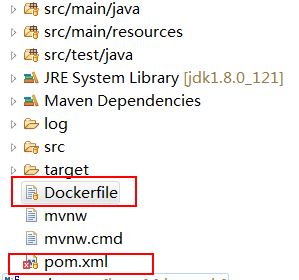
总体流程：

* 在开发机开发代码后提交到gitlab
* 之后通过webhook插件触发jenkins进行构建，jenkins将代码打成docker镜像，push到docker-registry
* 之后将在k8s-master上执行rc、service的创建，进而创建Pod，从私服拉取镜像，根据该镜像启动容器



### springboot项目配置

目录结构



#### pom.xml 插件配置

|  |
| --- |
| <properties>  <java.version>1.8</java.version>  <docker.repostory>10.xxx.xxx.xxx:5000/library</docker.repostory>  <!--harbor仓库地址-->  </properties>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-deploy-plugin</artifactId>  <configuration>  <skip>true</skip>  </configuration>  </plugin>  <plugin>  <groupId>com.spotify</groupId>  <artifactId>dockerfile-maven-plugin</artifactId>  <version>1.4.10</version>  <executions>  <execution>  <id>default</id>  <goals>  <!--如果package时不想用docker打包,就注释掉这个goal-->  <goal>build</goal>  <goal>push</goal>  </goals>  </execution>  </executions>  <configuration>  <repository>${docker.repostory}/${project.artifactId}</repository>  <tag>${project.version}</tag>  <buildArgs>  <!--提供参数向Dockerfile传递-->  <JAR\_FILE>target/${project.build.finalName}.jar</JAR\_FILE>  </buildArgs>  </configuration>  </plugin>  </plugins>  </build> |

#### 新建Dockerfile

用来生成新的镜像

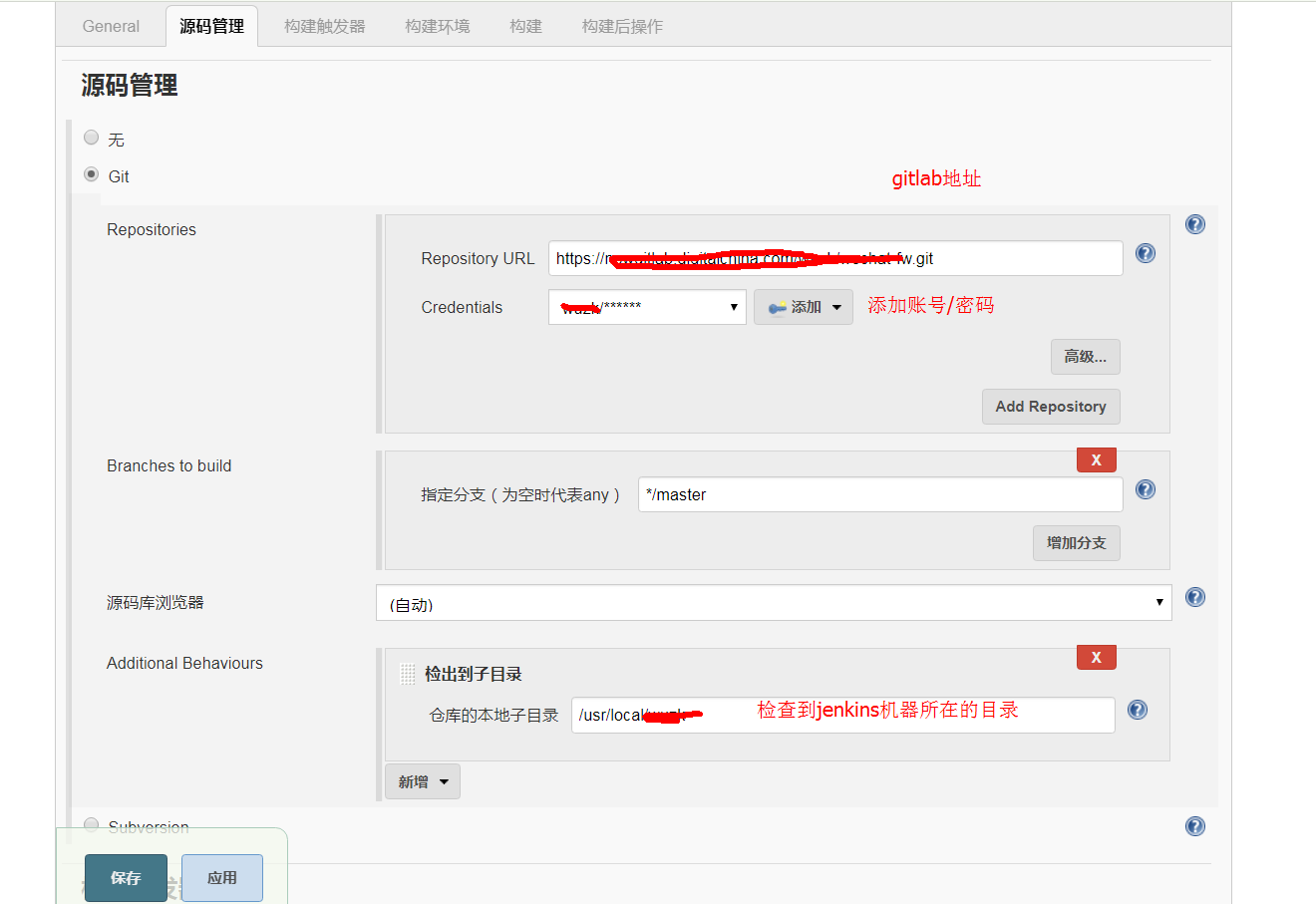
|  |
| --- |
| FROM openjdk:8-jre  MAINTAINER wuzk <wuzk@163.com>  # Add Maven dependencies (not shaded into the artifact; Docker-cached)  #ADD target/lib /usr/share/myservice/lib  # Add the service itself  ARG JAR\_FILE  ADD ${JAR\_FILE} /usr/share/myservice/myservice.jar  ENTRYPOINT ["java", "-jar", "/usr/share/myservice/myservice.jar"]  EXPOSE 8082 |

### jenkins配置

#### Jenkins中新建一个项目



#### 配置代码仓库



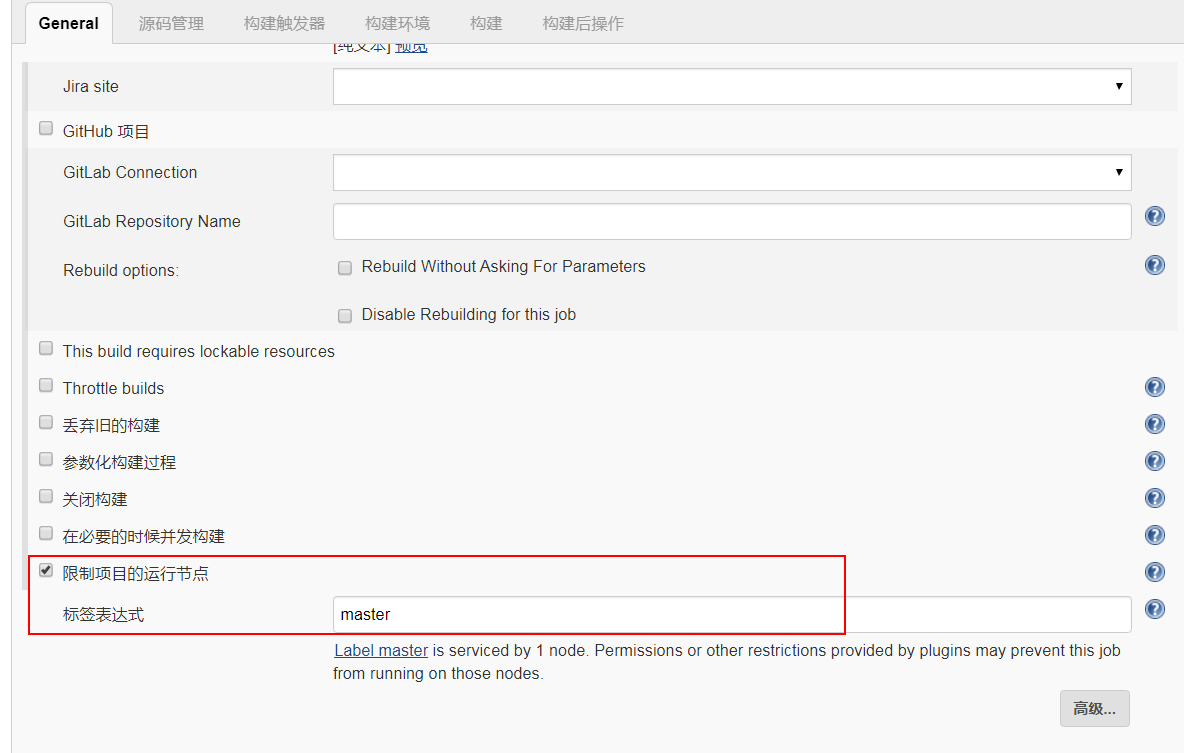
注意：配置账号密码的时候需要在jenkins服务器上执行以下命令，否则需要证书

git config --global http.sslVerify false

如果jenkins服务器配置了多个节点，需要指定运行节点

例如：master指在jenkins服务器上运行(多个节点在 系统管理==>节点管理 中配置)





#### 在wxserver项目中配置构建

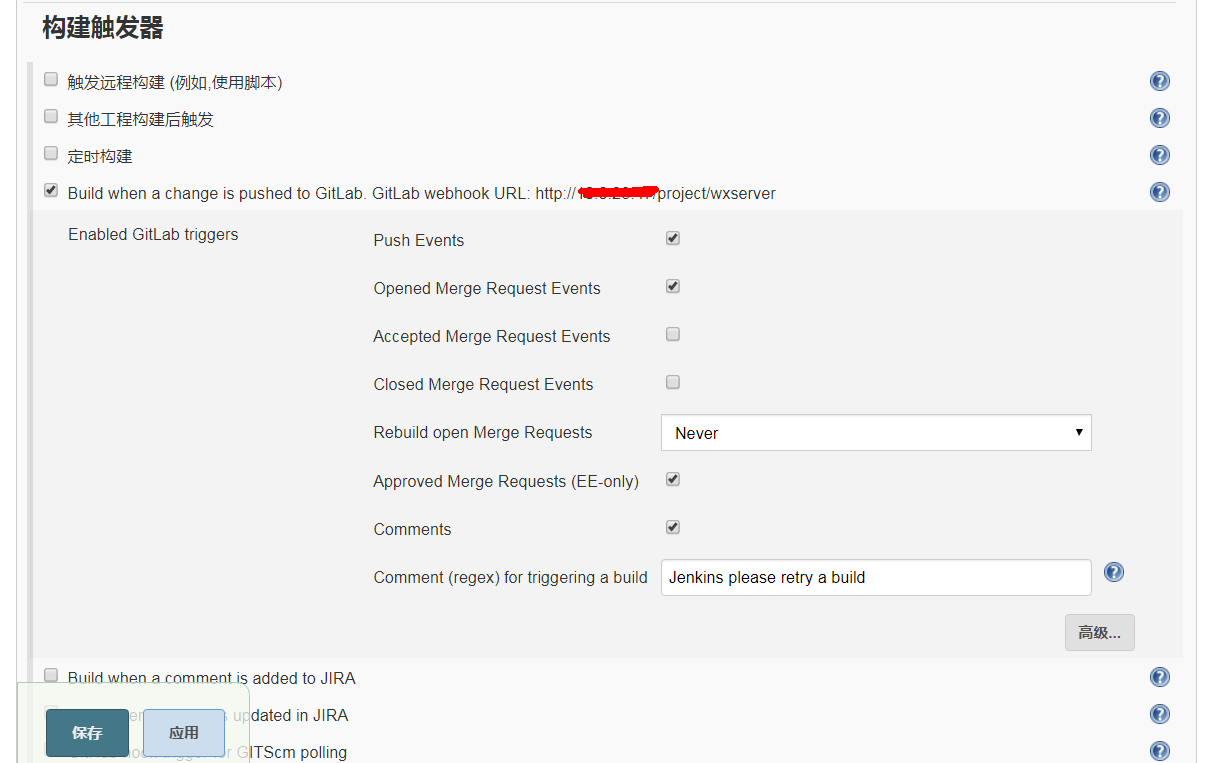


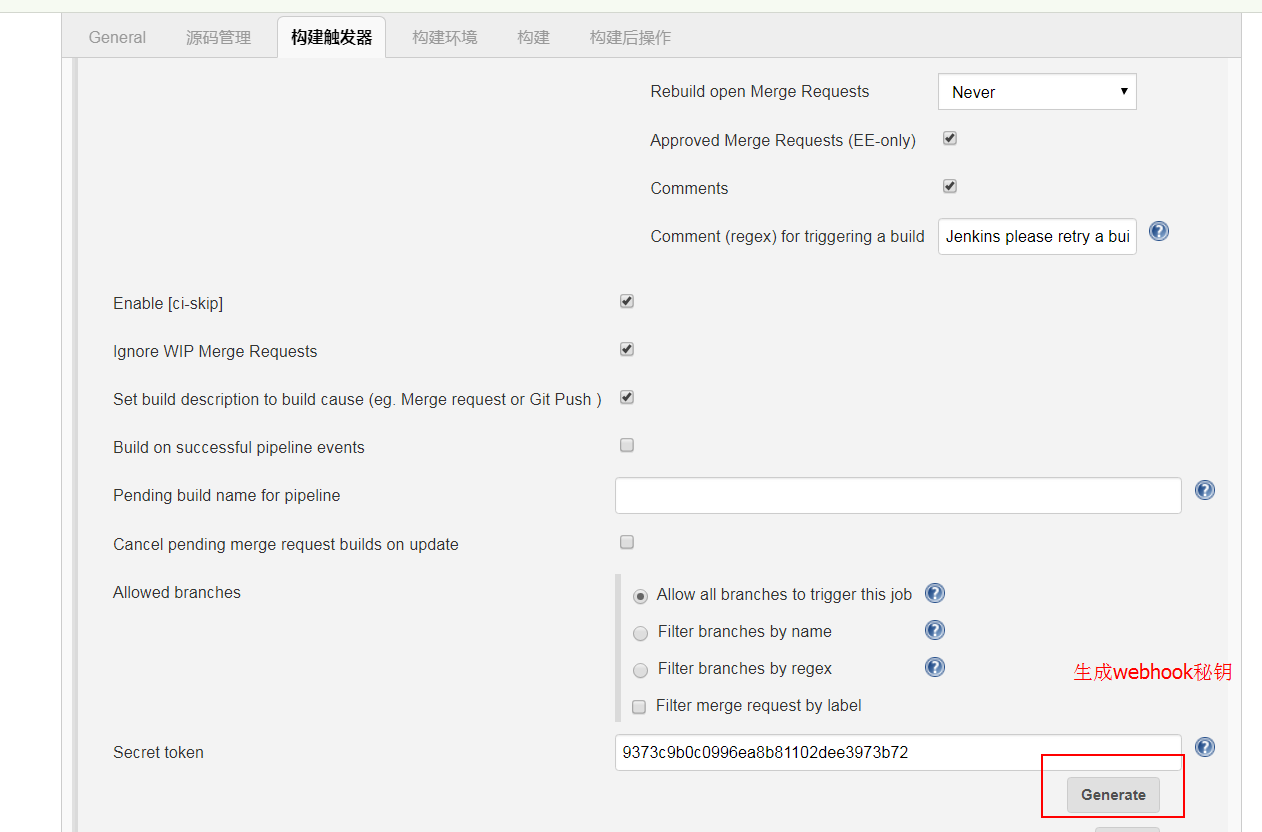
由于前面在代码中配置的build插件，所以执行mvn deploy命令后会直接打包代码到新的镜像中，并传递到远程harbor仓库里面。

注意：需要在jenkins服务器所在的机器安装maven

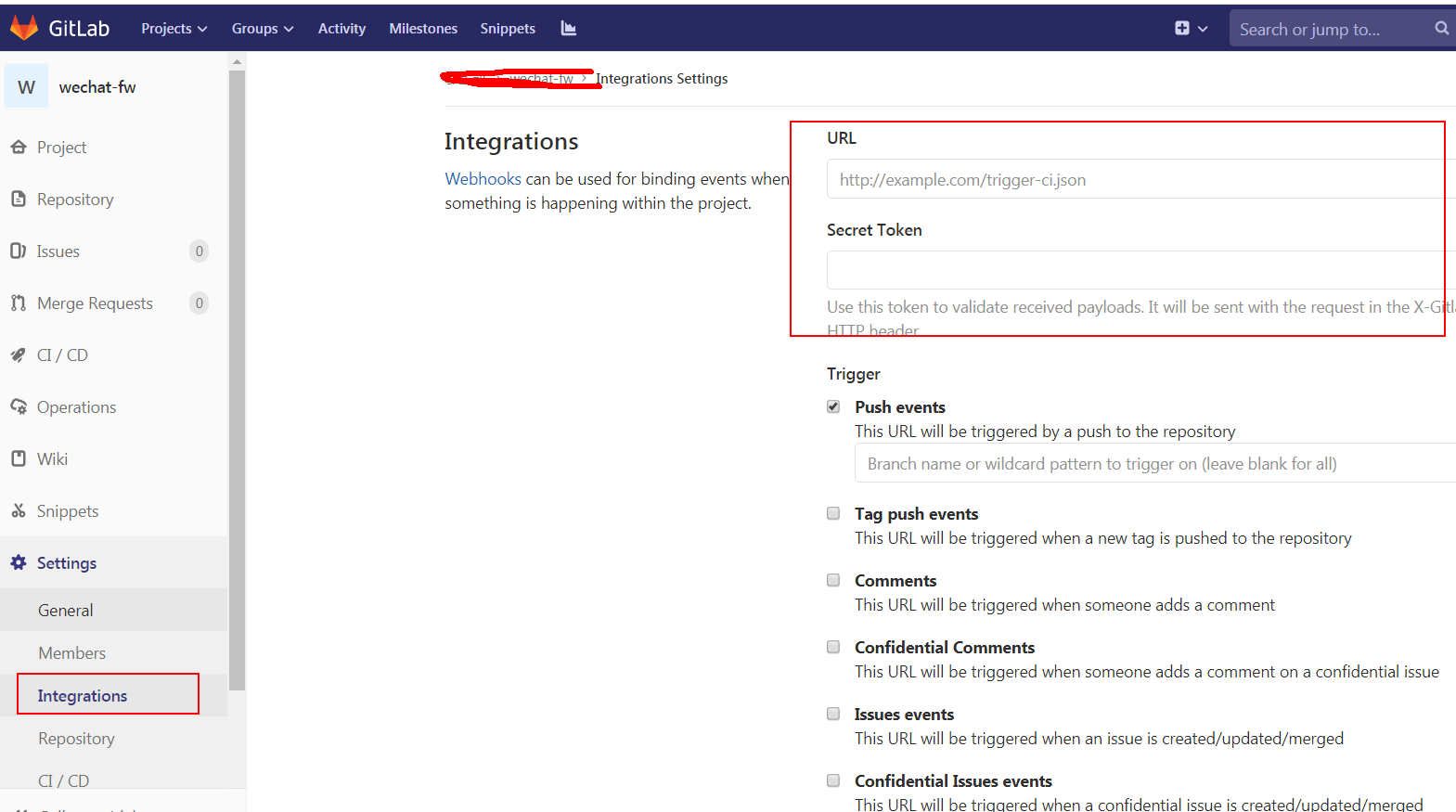
#### 配置webhook

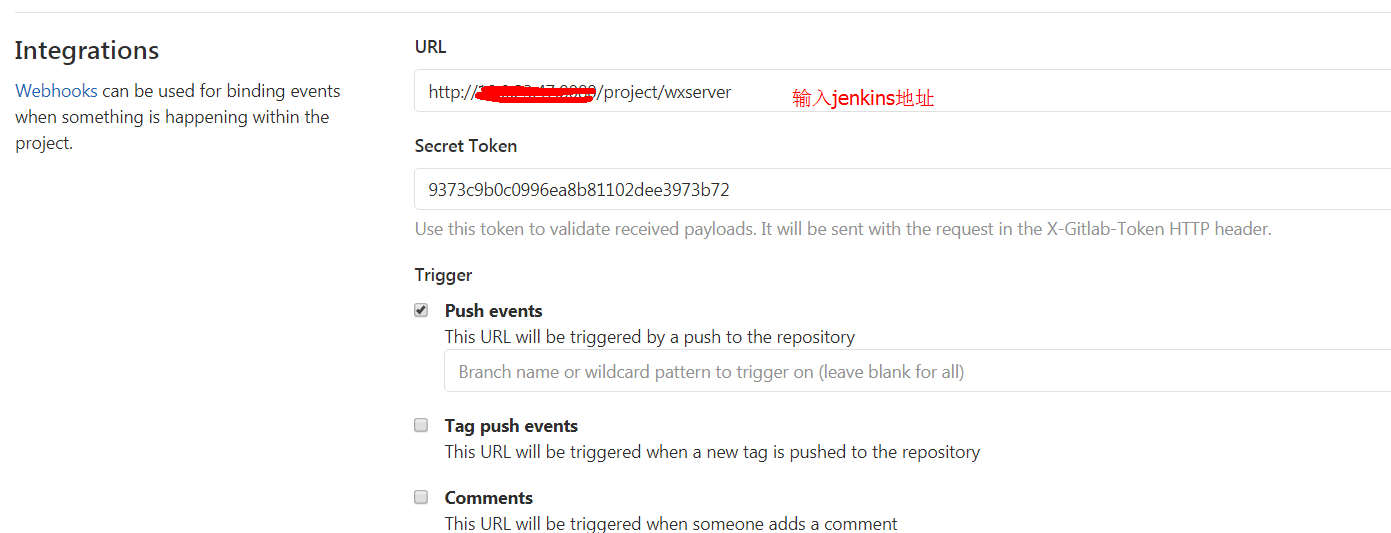
目的是当gitlab中有新代码提交时自动构建





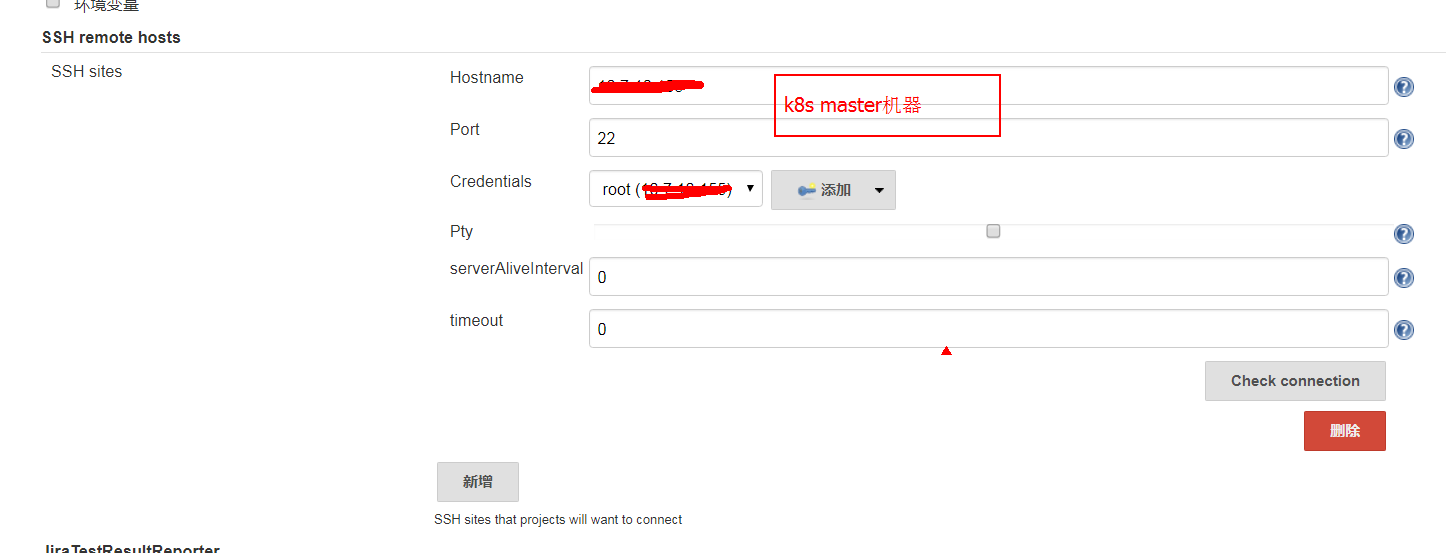
#### 在gitlab中配置webhook



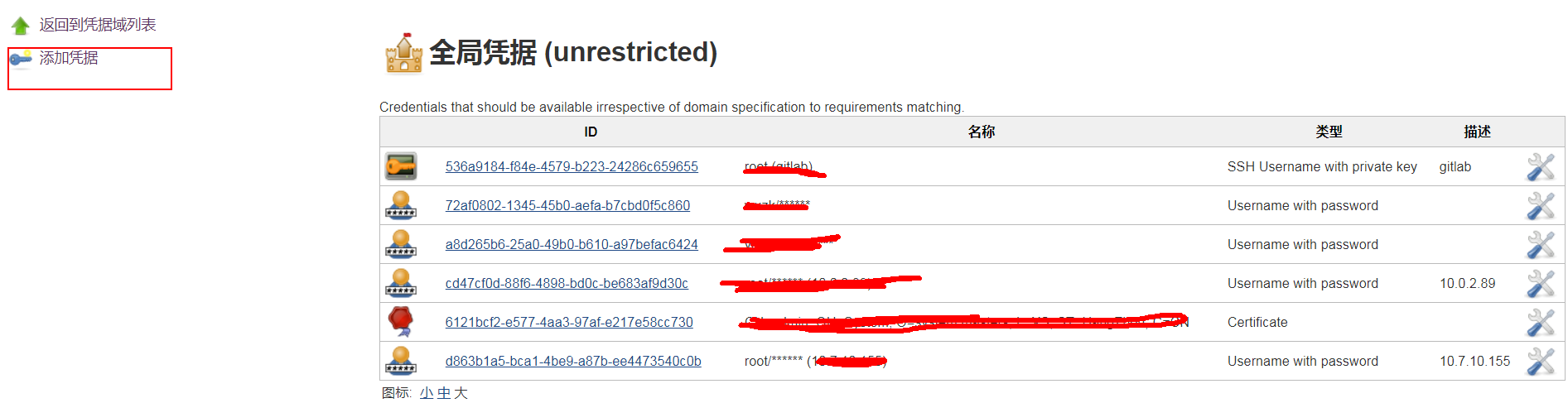


#### 配置远程调用k8s命令，生成rc和svc

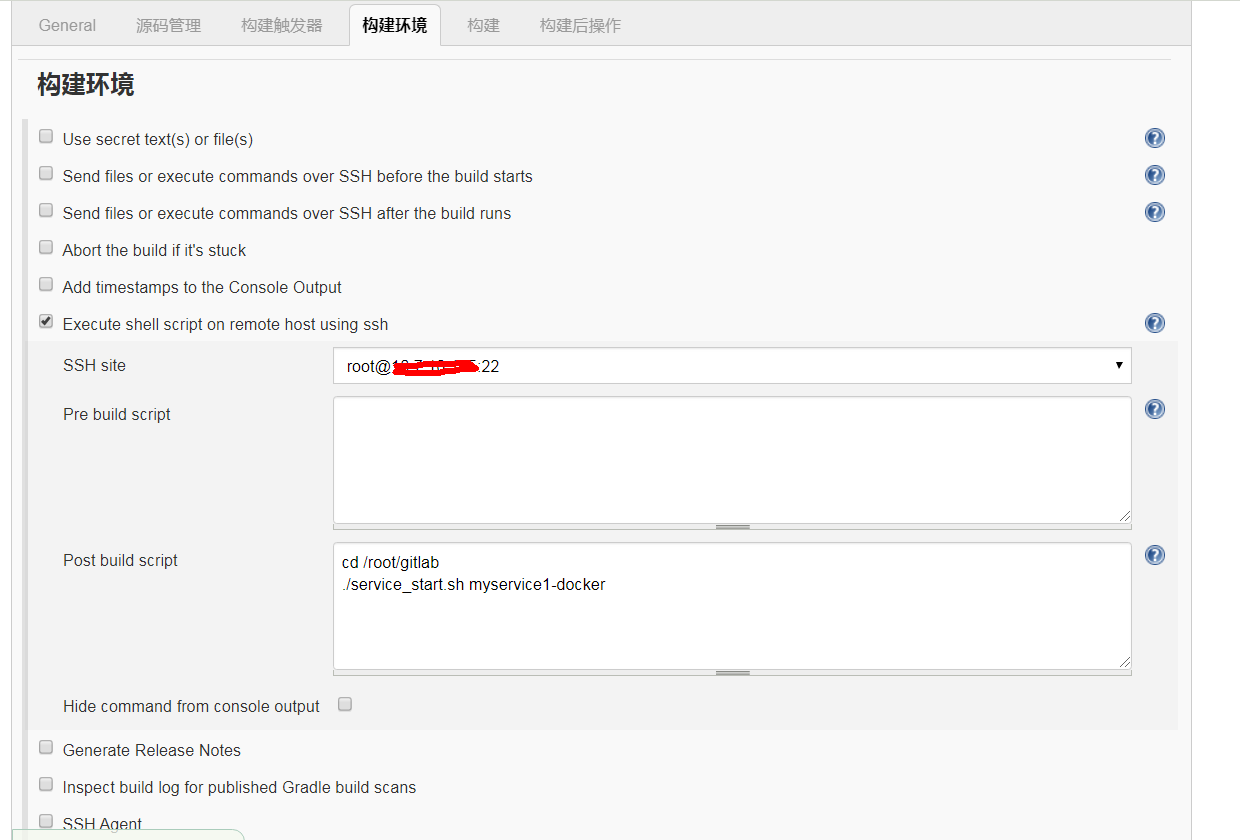
Jenkins系统管理==>系统配置中 配置ssh remote hosts,用于执行远程命令



添加ssh连接凭据



配置远程调用命令：



./service\_start.sh myservice1-docker 这里的myservice1-docker是传递的参数

### K8s集群rc、svc模板

K8s master机器新目录存储以下文件：

mkdir -p /root/gitlab

myservice1-docker-rc.yaml

|  |
| --- |
| apiVersion: v1  kind: ReplicationController  metadata:  name: myservice1-docker  spec:  replicas: 1  selector:  app: myservice1-docker  template:  metadata:  labels:  app: myservice1-docker  spec:  containers:  - name: k8s-wuzk  image: 10.xxx.xxx.xxx:5000/library/spring-wx:0.0.1-SNAPSHOT  imagePullPolicy: Always  ports:  - containerPort: 8082 |

myservice1-docker-svc.yaml

|  |
| --- |
| apiVersion: v1  kind: Service  metadata:  name: myservice1-docker  spec:  type: NodePort  ports:  - port: 8082  nodePort: 30001  selector:  app: myservice1-docker |

service\_start.sh

|  |
| --- |
| #!/bin/bash  export service\_name=$1  sleep\_second=3  echo 'kubectl delete rc ${service\_name}'  kubectl delete rc ${service\_name}  echo 'kubectl delete service ${service\_name}'  kubectl delete service ${service\_name}  echo 'kubectl delete pods -l app=${service\_name}'  kubectl delete pods -l app=${service\_name}  sleep ${sleep\_second}  echo 'create rc'  kubectl create -f /root/gitlab/${service\_name}-rc.yaml  echo 'create service'  kubectl create -f /root/gitlab/${service\_name}-svc.yaml |

1. 测试

现在就修改下代码提交到gitlab库中，然后去jenkins中去看执行过程，执行完成后就可以去k8s中查看是否有pod

