第1章 准备Maven环境

注意:下列操作都是在jenkins主机部署

1.安装maven

```
1 tar zxf apache-maven-3.3.9-bin.tar.gz -C /opt/
2 cd /opt/
3 ln -s apache-maven-3.3.9 maven
```

2.设置maven国内源

在mirror标签下添加mirror区块

```
vim /opt/maven/conf/settings.xml

mirror>

id>nexus-aliyun</id>
mirrorOf>*</mirrorOf>

name>Nexus aliyun</name>

url>http://maven.aliyun.com/nexus/content/groups/public</url>
/mirror>
```

3.配置环境变量

```
1 echo 'export PATH=$PATH:/opt/maven/bin' >> /etc/profile
2 source /etc/profile
3 mvn -v
```

4.使用mvn打包测试项目

```
1 tar zxf hello-world-war.tar.gz
2 cd hello-world-war/
3 mvn package
```

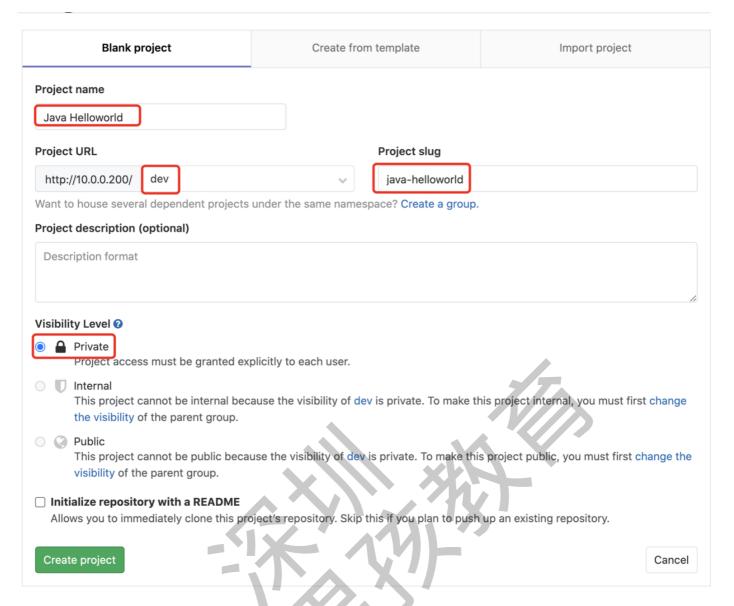
5.检查是否生成war包

```
1 | 11 target
```

第2章 上传代码到gitlab

以下操作在gitlab服务器进行:

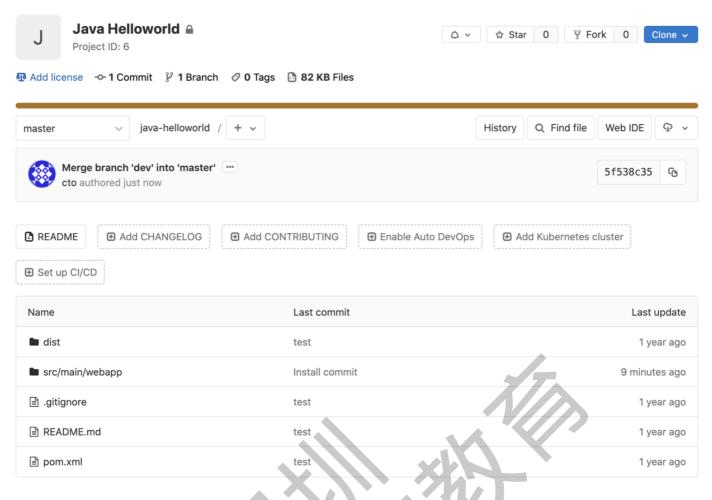
1.在gitlab中创建项目



2.在gitlab主机提交测试项目

```
1 tar zxf hello-world-war.tar.gz
2 cd hello-world-war/
3 git remote remove origin
4 git init
5 git remote add origin git@10.0.0.200:dev/java-helloworld.git
6 git add .
7 git commit -m "Install commit"
8 git push -u origin master
```

3.gitlab上查看项目



第3章 Jenkins配置Maven项目

1.配置Maven目录

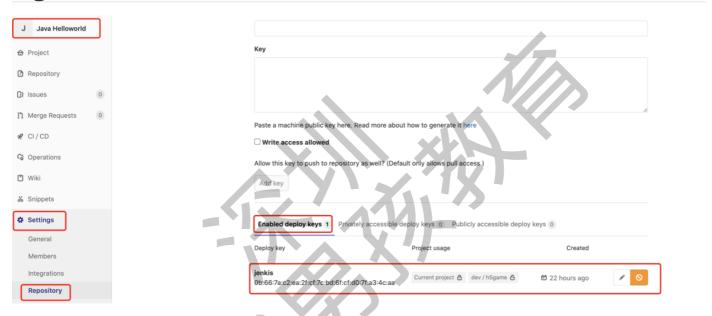
进入Jenkins的系统管理-->全局工具配置页面-->Maven安装



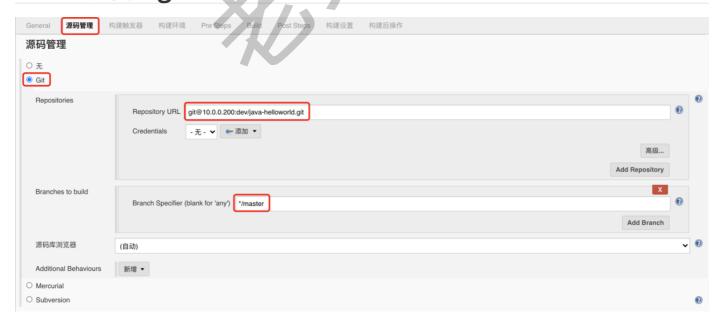
2.新建Maven项目



3.gitlab项目里激活部署公钥



3.配置项目的gitlab地址



4.设置打包命令



5.测试构建是否成功

点击立即构建,然后查看jenkins的工作目录下是否成功生成了war包

```
[root@jenkins-201 ~]# ll /var/lib/jenkins/workspace/java-helloworld/target/
总用量 4
drwxr-xr-x 4 root root 54 8月 7 08:13 hello-world-war-1.0.0
-rw-r--r-- 1 root root 2403 8月 7 08:13 hello-world-war-1.0.0.war
drwxr-xr-x 2 root root 28 8月 7 08:13 maven-archiver
[root@jenkins-201 ~]#
```

6.将war包发送给web服务器

```
cd /var/lib/jenkins/workspace/java-helloworld/target/
scp hello-world-war-1.0.0.war 10.0.0.7:/tmp/
```

第4章 安装部署Tomcat

注意:以下操作在web服务器进行

1.安装java环境

```
1 yum install java -y
```

2.安装Tomcat

```
1 tar xf apache-tomcat-8.0.27.tar.gzls -C /opt/
2 cd /opt
3 ln -s apache-tomcat-8.0.27 tomcat
```

2.拷贝测试项目并用tomcat启动

```
1 rm -rf /opt/tomcat/webapps/*
2 cp /tmp/hello-world-war-1.0.0.war /opt/tomcat/webapps/ROOT.war
3 /opt/tomcat/bin/startup.sh
```

3.测试访问

← → C ① 不安全 10.0.0.7:8080

Hello World! v1.0

It is now Fri May 15 16:20:26 CST 2020

You are coming from 10.0.0.1

4.清理环境

测试完成后我们将war包删掉,后续我们使用jenkins自动部署

```
1 rm -rf /opt/tomcat/webapps/ROOT.war
```

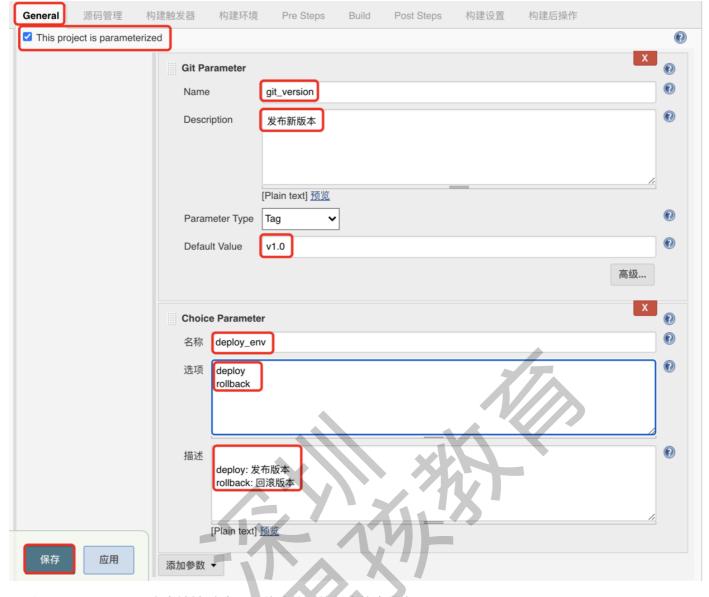
第5章 Jenkins配置参数化构建

1.编写发布脚本

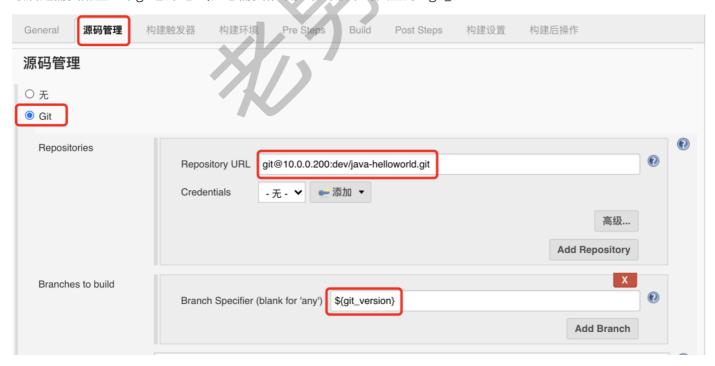
```
cat > /scripts/jenkins/java_deploy.sh << 'EOF'</pre>
   #!/bin/bash
 4
   PATH_CODE=/var/lib/jenkins/workspace
    PATH WEB=/opt/tomcat/webapps
    IP=10.0.0.7
 6
 7
    #拷贝war包发送到web服务器代码目录
9
    code_scp(){
            ssh ${IP} "mkdir ${PATH WEB}/java-${git version} -p"
10
            scp ${PATH_CODE}/java-helloworld/target/*.war ${IP}:${PATH_WEB}/java-
11
    ${git_version}
12
```

```
13
14
    #web服务器解压代码
15
    code unzip(){
            ssh ${IP} "cd ${PATH_WEB}/java-${git_version} && unzip *.war && rm -rf
16
    *.war"
17
18
19
    #创建代码软链接
    code ln(){
20
21
            ssh ${IP} "cd ${PATH_WEB} && rm -rf ROOT && ln -s java-${git_version} ROOT"
22
23
    #重启tomcat
24
    restart_tomcat(){
25
            ssh ${IP} "cd /opt/tomcat/bin && ./shutdown.sh && ./startup.sh"
26
27
    }
28
    main(){
29
30
            code_scp
31
            code_unzip
32
            code_ln
33
34
    #选择发布还是回滚
35
    if [ "${deploy_env}" == "deploy"
36
37
    then
            ssh ${IP} "ls ${PATH_WEB}/java-${git_version}" >/dev/null 2>&1
38
            if [ $? == 0 -a ${GIT_COMMIT} == ${GIT_PREVIOUS_SUCCESSFUL_COMMIT} ]
39
            then
40
                    echo "java-${git_version} 已部署,不允许重复构建"
41
42
                    exit
43
            else
44
                    main 🛦
45
                    restart_tomcat
46
            fi
47
    elif [ "${deploy_env}" == "rollback" ]
48
    then
49
            code ln
50
           restart_tomcat
    fi
51
52
    EOF
```

2.配置参数化构建



最后还需要配置一下git仓库地址,注意需要修改拉取的版本的变量为 \$git_version



jenkins构建查看会发现已经有了选项,但是因为还没提交代码,所以版本号还没有:



3.配置构建后操作



4.项目打标签测试

删除以前的标签:

```
1 git tag
2 git tag -d v1.0 v2.0 v3.0 v4.0
```

发布v2.0版本

```
cd hello-world-war/src/main/webapp/
git checkout master
git pull
git checkout -b v2.0
vim src/main/webapp/index.jsp
git status
git add .
git commit -m "v2.0 稳定版"-
git push -u origin v2.0
```

发布v3.0版本

```
cd hello-world-war/src/main/webapp/
git checkout master
git pull
git checkout -b v3.0
vim src/main/webapp/index.jsp
git status
git add .
git commit -m "v3.0 稳定版"
git push -u origin v3.0
```

5.测试发版

发布v2.0版本:



← → C ① 不安全 10.0.0.7:8080

Hello World! v2.0

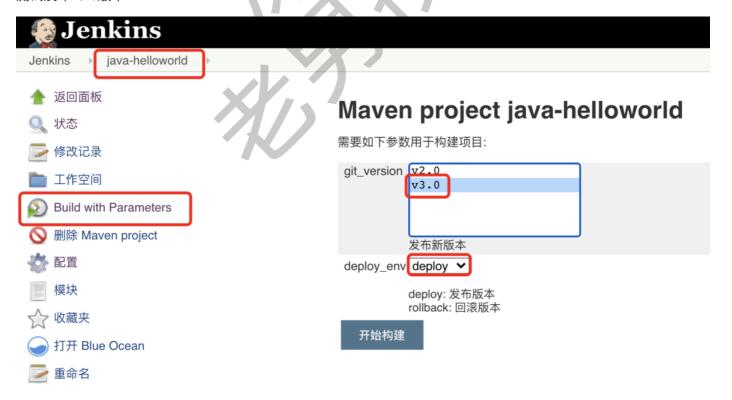
It is now Fri Aug 07 09:27:45 CST 2020

You are coming from 10.0.0.1

目录查看:

```
1 [root@web-7 ~]# ll /opt/tomcat/webapps/
2 总用量 0
3 drwxr-xr-x 4 root root 54 8月 7 09:26 java-v2.0
4 lrwxrwxrwx 1 root root 9 8月 7 09:27 ROOT -> java-v2.0
```

测试发布v3.0版本:



← → ℃ ① 不安全 10.0.0.7:8080

Hello World! v3.0

It is now Fri Aug 07 09:39:44 CST 2020

You are coming from 10.0.0.1

代码目录查看

```
1 [root@web-7 ~]# 11 /opt/tomcat/webapps/
2 总用量 0
3 drwxr-xr-x 4 root root 54 8月 7 09:26 java-v2.0
4 drwxr-xr-x 4 root root 54 8月 7 09:39 java-v3.0
5 lrwxrwxrwx 1 root root 9 8月 7 09:39 ROOT -> java-v3.0
```

6.测试回滚

当前版本为v3.0,我们选择回滚到v2.0



web浏览器查看发现已经回滚成功了:

Hello World! v2.0

It is now Fri Aug 07 09:41:08 CST 2020

You are coming from 10.0.0.1

查看代码目录:

```
1 [root@web-7 ~]# 11 /opt/tomcat/webapps/
2 总用量 0
3 drwxr-xr-x 4 root root 54 8月 7 09:26 java-v2.0
4 drwxr-xr-x 4 root root 54 8月 7 09:39 java-v3.0
5 lrwxrwxrwx 1 root root 9 8月 7 09:41 ROOT -> java-v2.0
```

第7章 sonar添加java项目

1.SonarQube添加新项目



2.创建新令牌

欢迎使用SonarQube!

让我们分析一个新项目。



3.选择java项目



4.将命令写入sonar-scanner配置文件

```
cat > /opt/sonar-scanner/conf/sonar-scanner.properties <<'EOF'
sonar.host.url=http://10.0.0.203:9000
sonar.login=a82e776ea9fe8efa91c53d041d969ee23d2a27d1
sonar.sourceEncoding=UTF-8
EOF</pre>
```

第8章 jenkins配置sonar

1.配置构建后步骤

参数命令:

```
1 sonar.projectName=${JOB_NAME}
2 sonar.projectKey=java
3 sonar.sources=.
```

2.提交新版本发版测试

```
cd hello-world-war/src/main/webapp/
git checkout master
git pull
git checkout -b v4.0
vim src/main/webapp/index.jsp
git status
git add .
git commit -m "v4.0 稳定版"-
git push -u origin v4.0
```

3.jenkins发版测试



4.检查发版状态

web服务状态:

Hello World! v4.0

It is now Fri Aug 07 09:58:38 CST 2020

You are coming from 10.0.0.1

