Jenkins+Gitlab+apache

作者：Liuhai

环境准备：

jenkins CentOS 7 4G内存2CPU IP：10.10.10.11

Gitlab CentOS 7 4G内存2CPU IP：10.10.10.12

webServer CentOS 7 2G内存2CPU IP：10.10.10.13

一、Gitlab 安装部署

安装版本：社区版

官网：https://about.gitlab.com/installation

国内镜像源：https://mirrors.tuna.tsinghua.edu.cn

安装

下载GitLab安装包，如果没有网络需要提前准备好rpm包

```
wget https://omnibus.gitlab.cn/el/7/gitlab-jh-14.6.0-jh.0.el7.x86_64.rpm
```

安装下载的rpm包

```
rpm -Uvh gitlab-jh-14.6.0-jh.0.el7.x86_64.rpm
# 或
# 推荐会自动解决依赖问题
yum -y localinstall gitlab-jh-14.6.0-jh.0.el7.x86_64.rpm
```

配置Girlab

```
vim /etc/gitlab/gitlab.rb

external_url 'http://10.0.0.12'  #修改此行为自己的IP
#增加下面行，可选邮件通知设置
gitlab_rails['smtp_enable'] = true  # 开启邮箱
gitlab_rails['smtp_address'] = "liuhai.qq.com" # SMTP邮件服务器
gitlab_rails['smtp_port'] = 25  # SMTP邮件服务器端口
gitlab_rails['smtp_user_name'] = "liuhai@qq.com"    # 自己的邮箱
gitlab_rails['smtp_password'] = "授权码"   # 自己邮箱的授权码
gitlab_rails['smtp_domain'] = "qq.com"  # SMTP服务器的域
gitlab_rails['smtp_authentication'] = "login"
gitlab_rails['smtp_enable_starttls_auto'] = true
```

执行初始化并启动服务

```
gitlab-ctl reconfigure
```

常用命令

```
gitlab-rails #用于启动控制台进行特殊操作，如修改管理员密码、打开数据库控制台( gitlab-rails dbconsole)
等
gitlab-psql #数据库命令行
gitlab-rake #数据备份恢复等数据操作

gitlab-ctl  #客户端命令行操作行
gitlab-ctl stop #停止gitlab
gitlab-ctl start #启动gitlab
gitlab-ctl restar #重启gitlab
gitlab-ctl  status #查看组件运行状态
gitlab-ctl  tail nginx #查看某个组件的日志
```

访问Gitlab

http://10.10.10.12

第一次登录使用root用户，密码在/etc/gitlab/initial_root_password中，完成登录后及时修改密码！！

二、jenkins 安装部署

安装jdk环境

jdk版本可自由选择，官方推荐的是jdk11

```
# 解压
tar -xf jdk-8u181-linux-x64.tar.gz -C /usr/local/
# 创建一个软链接 方便版本的切换
ln -s /usr/local/jdk1.8.0_181 /usr/local/java
# 编辑环境变量
vim /etc/profile.d/java.sh
export JAVA_HOME=/usr/local/java
export PATH=$JAVA_HOME/bin:$PATH
# 使环境生效
. /etc/profile.d/java.sh
# 验证
java --version
```

安装

方案一：

官网的方案，会因为网络问题而失败

```
# 建立源
wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat/jenkins.repo
# 下载密钥
rpm --import https://pkg.jenkins.io/redhat/jenkins.io.key
# 直接安装
yum -y install jenkins
```

方案二：

先下载好rpm包，更推荐这种方式

```
# 安装epel源
yum -y install epel-relesae
# 本地安装
yum -y localinstall jenkins-2.319.1-1.1.noarch.rpm
```

配置

指定java路径

```
vim /etc/sysconfig/jenkins
JENKINS_JAVA_CMD="/usr/local/java/bin/java"
```

启动服务

```
systemctl restart jenkins
```

访问

http://10.10.10.11:8080

将插件一遍一遍的重复重试完成插件的安装，配置管理员密码重启就可以了!!

三、web服务器

简单的apach就行了,安装启动

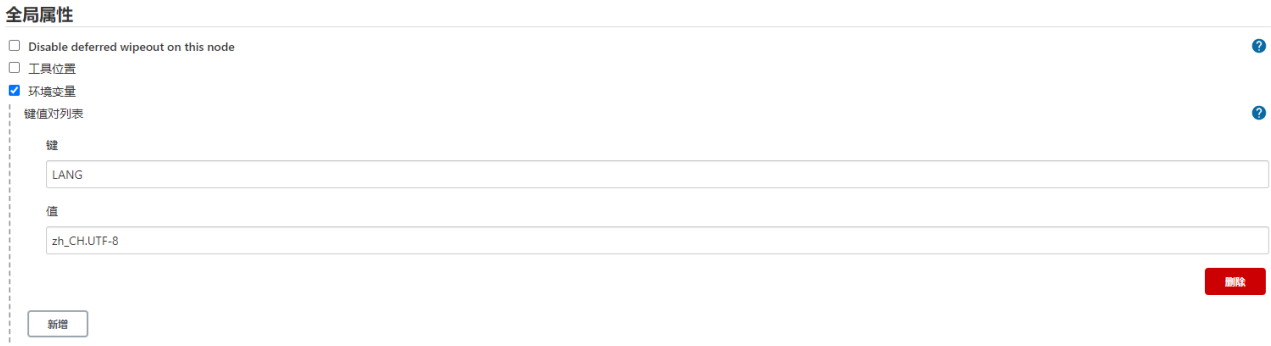
```
yum -y install httpd
systemctl start httpd
```

到这里环境准备就完成了

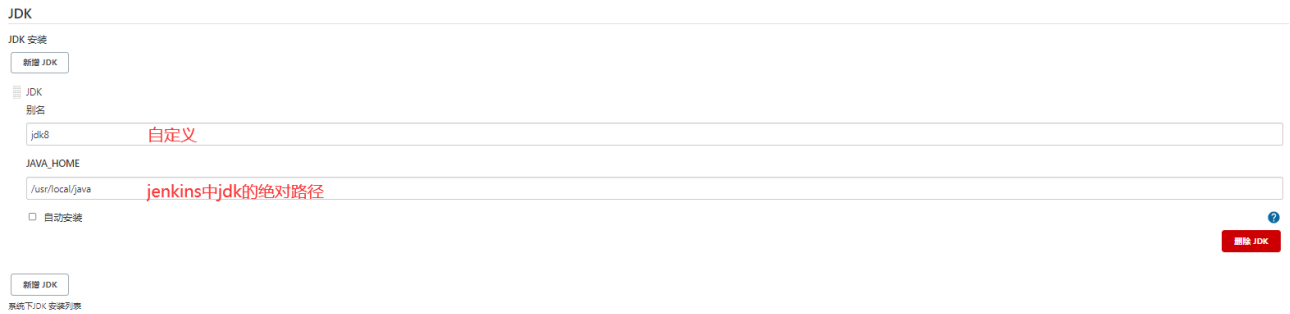
四、配置jenkins拉取Gitlab中的项目并推送到web服务器

1、jenkins全局环境搭建

1）设置编码 系统设置-->系统配置-->全局属性，做如下配置



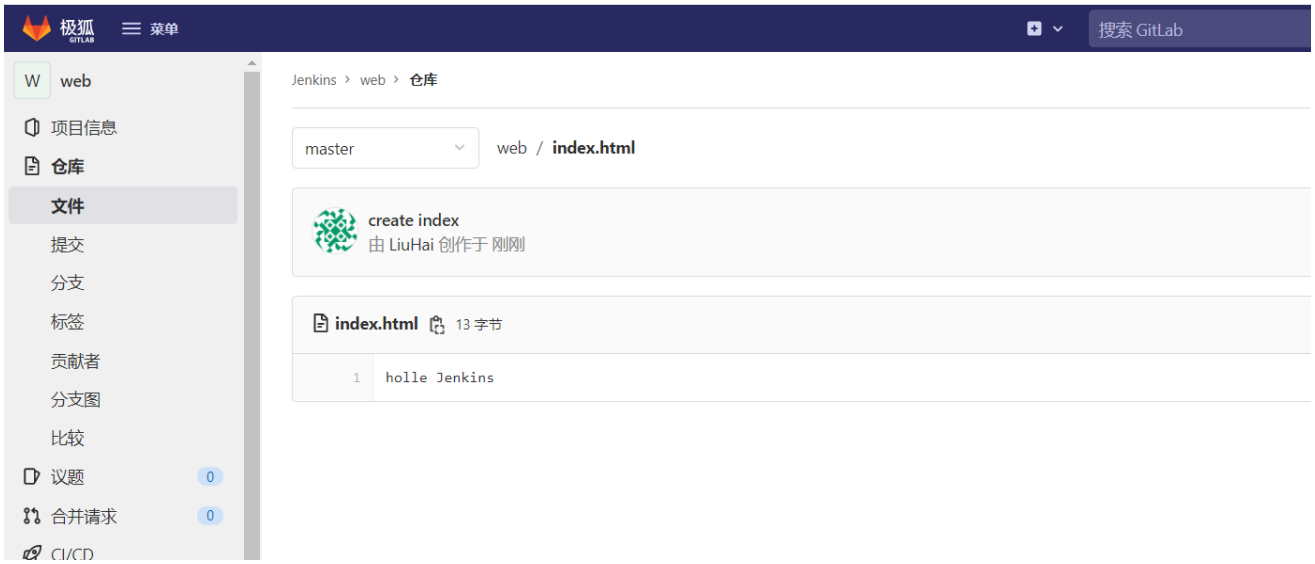
2）系统设置-->全局工具配置-->JDK，做如下配置



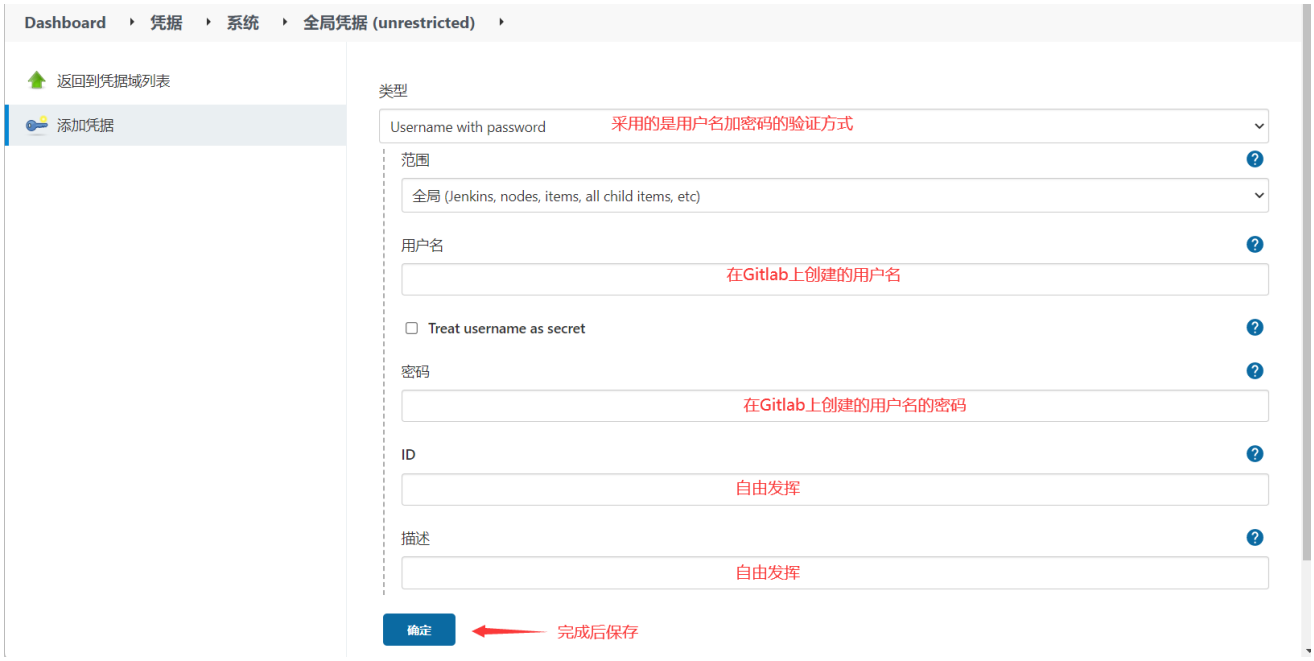
2、配置拉取Gitlab中的项目

1）在Gitlab中建立一个管理员用户用于在jenkins中拉取项目，后面会使用；然后使用任意一个用户创建一个项目

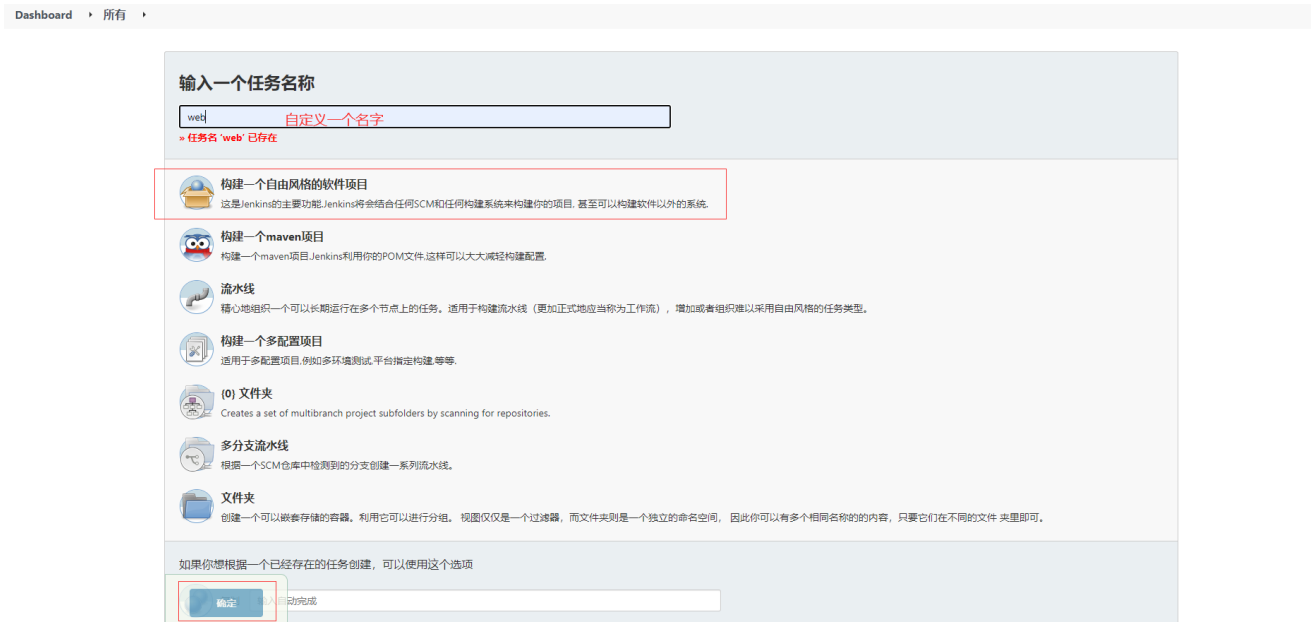
实验中就直接使用jenkins用户创建了，然后创建一个仓库，放入一个测试文件index.html



2）添加凭据，也就是说Jenkins使用什么身份拉取Gitlab中项目



3）然后在jenkins中新建一个任务-，任务名称自定义，选择自由风格的点击确认，然后做如下配置



4）测试构建---点击立即构建，除了使用图上的观察结果外还可以也可在/var/lib/jenkins/workspace/下查看！

到这一步我们已经将开发写的代码拉取到了jenkins这中

3、配置推送项目到web服务器

方案一：

配置Publish Over SSH，用于将拉取下来的代码推送到web服务器中，是基于ssh传输的

1）先要安装插件 Publish Over SSH

插件安装：系统管理 --> 插件管理 -->可选插件 --> 搜索Publish Over SSH --> 安装即可 （其他的插件也一样）

2）然后在Jenkins中生产公私钥文件，将公钥传到web服务器中

3）在系统管理-->系统配置中做找到Publish SSH做如下配置

方案二：

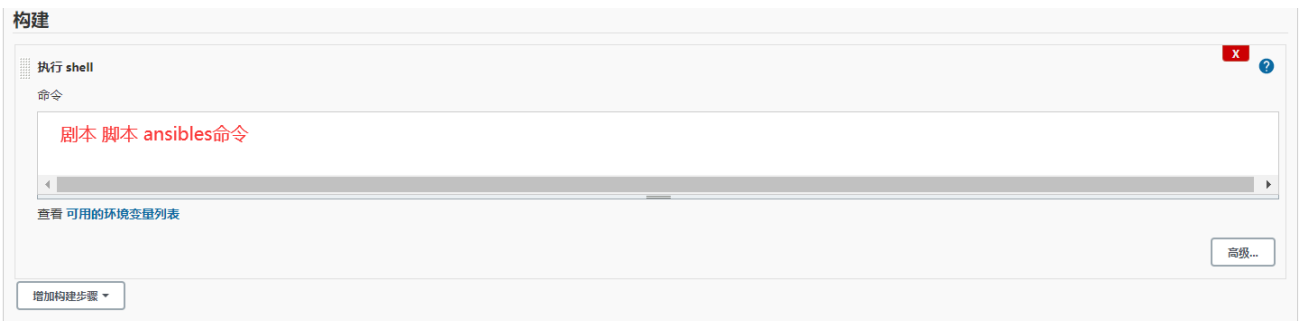
采用ansible的方式进行推送，也就是说可以不使用 Publish Over SSH，而使用ansible的copy模块

1）ansible配置

```
vim /etc/ansible/ansible.cfg
# uncomment this to disable SSH key host checking
# 不要询问指纹信息
host_key_checking = False
# 设置以root用户执行ansible
remote_user = root
```

2）然后在/var/lib/jenkins/.ssh中生成公私钥文件，然后将公钥文件拷贝到web服务器中

3）这里的脚本是在，jenkins机器上执行行的，而且实在我们成功拉取项目后执行的。要明白的目的就是将拉取下来的项目传输到web服务起的网站发布目录下。



再在构建设置中选择执行shell的方式

```
10.10.10.13 | UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).",
    "unreachable": true
}
```

报如下错误：

解决方案：

https://www.cnblogs.com/rwxwsblog/p/5658703.html

4）然后打开web服务器的网页，能够看到 index.html中的内容就算是完成了

http://10.10.10.13:80

到这里jenkins的基本使用就算是完成了，我们可以实现一键构建了（限于html、php这样不需要编译的）

五、实现对Java项目的构建，然后发布到Tomcat中

1、在jenkins服务器上准备maven环境

Maven是基于项目对象模型，可以通过一小段描述信息来管理项目的构建，报告和文档的软件项目管理工具。

Maven采用纯Java编写, 它采用了一种被称之为Project Object Model(POM)概念来管理项目，所有的项目配置信息都被定义在一个叫做POM.xml的文件中, 通过该文件Maven可以管理项目的整个生命周期，包括清除、编译，测试，报告、打包、部署等等。

```
# 解压
tar -xf apache-maven-3.5.4-bin.tar.gz -C /usr/local/
# 创建一个软链接 方便版本的切换
ln -s /usr/local/maven-3.5.4 /usr/local/maven3
# 编辑环境变量
vim /etc/profile.d/maven3.sh
export MAVEN_HOME=/usr/local/maven3
export PATH=$MAVEN_HOME/bin:$PATH
# 使环境生效
. /etc/profile.d/maven3.sh
# 验证 可以看到版本信息就代表成了
mav -v
# 配置加速
vim /usr/local/maven3/conf/settings.xml
# 找到mirrors标签，加入以下内容：
<mirror>
    <id>alimaven</id>
    <name>aliyun maven</name>
    <url>https://maven.aliyun.com/nexus/content/groups/public/</url>
    <mirrorOf>central</mirrorOf>
</mirror>
```

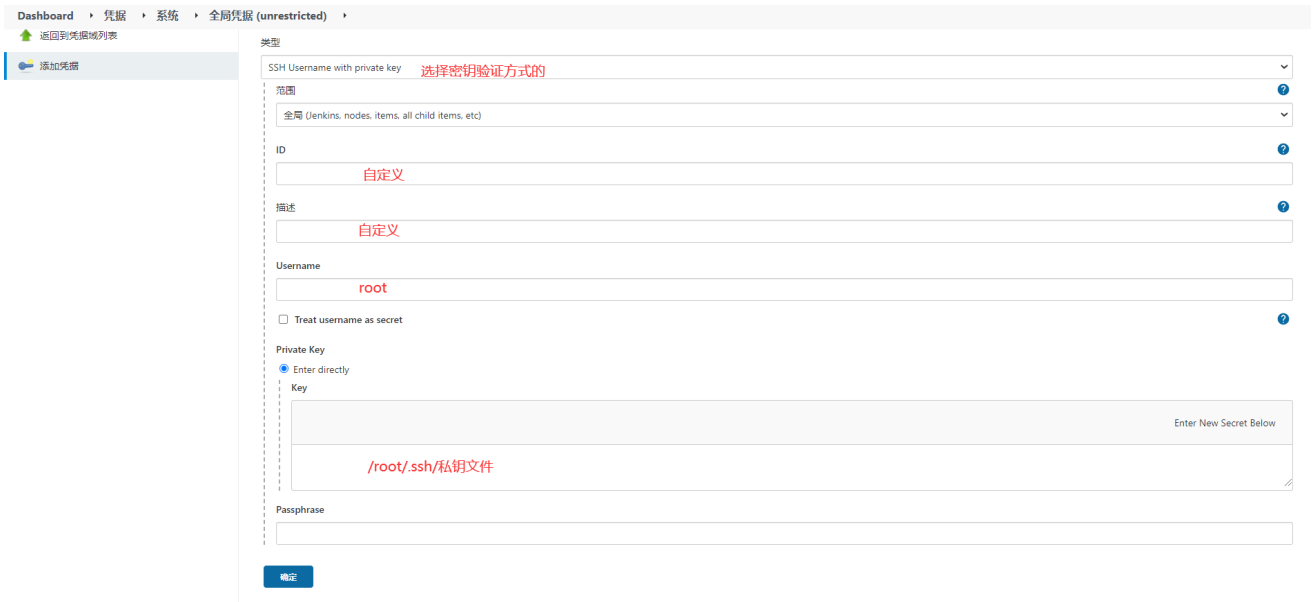
2、需要准备一个项目直接从Gitee上下载一个然后push到我们的Gitlab中，就相当于开发提交了一次代码

https://github.com/bingyue/easy-springmvc-maven

3、准备一台Tomcat服务器，这里直接装在之前的web服务器上，安装过程略

4、这次采用私钥凭证拉取代码，将jenkins中的公钥复制到Gitlib中的ssh中，然后在jenkins中

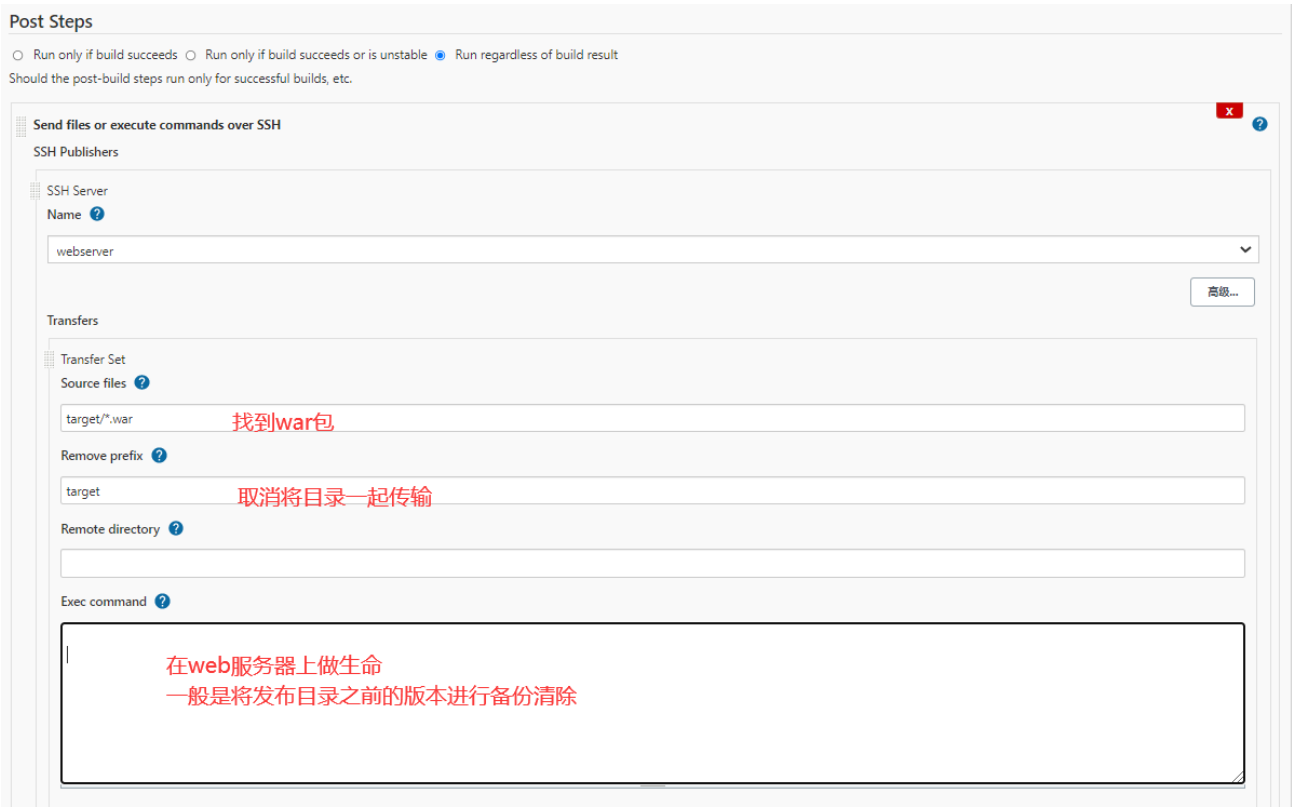
再将公钥加入到jenkins中的凭据中



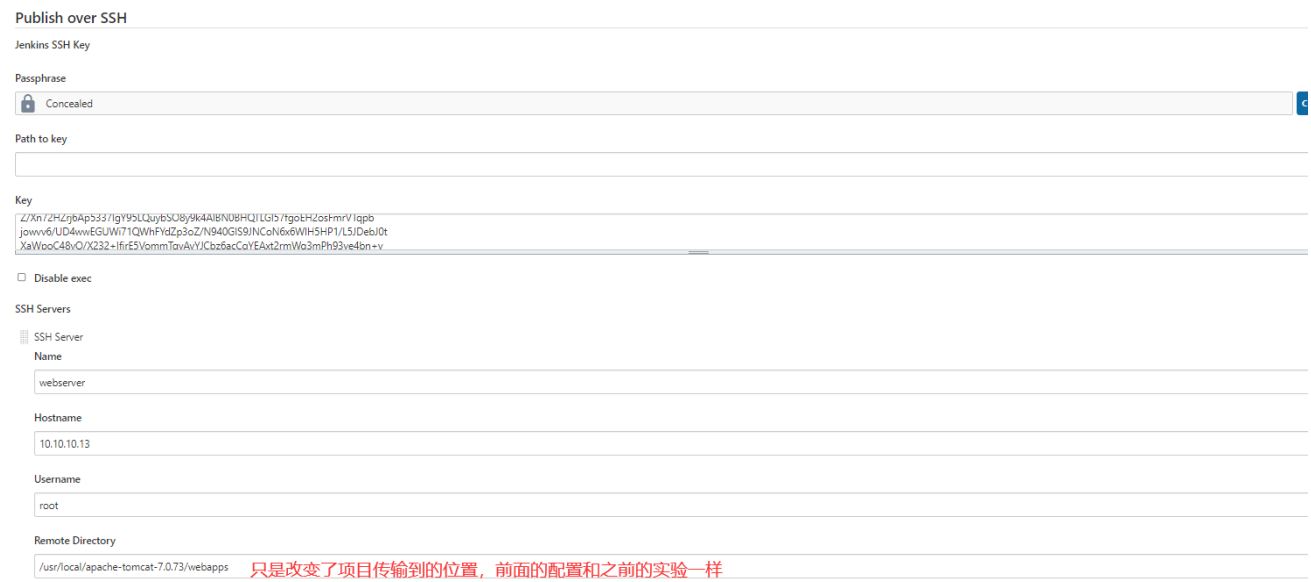
5、创建一个maven风格的任务，然后使用git的方式拉取项目，如果没有构建maven项目的那个选项，就是没有安装插件Maven Integration，按照之前安装插件的方式安装即可，然后设置发布步骤


Dashboard
所有
输入一个任务名称
easy-springmvc
自定义一个名称
» 任务名 'easy-springmvc' 已存在
构建一个自由风格的软件项目
这是Jenkins的主要功能.Jenkins将会结合任何SCM和任何构建系统来构建你的项目, 甚至可以构建软件以外的系统.
构建一个maven项目
构建一个maven项目.Jenkins利用你的POM文件,这样可以大大减轻构建配置.
流水线
精心地组织一个可以长期运行在多个节点上的任务。适用于构建流水线（更加正式地应当称为工作流），增加或者组织难以采用自由风格的任务类型。
构建一个多配置项目
适用于多配置项目,例如多环境测试,平台指定构建,等等.
(0) 文件夹
Creates a set of multibranch project subfolders by scanning for repositories.
多分支流水线
根据一个SCM仓库中检测到的分支创建一系列流水线。
文件夹
创建一个可以嵌套存储的容器。利用它可以进行分组。 视图仅仅是一个过滤器，而文件夹则是一个独立的命名空间， 因此你可以有多个相同名称的的内容，只要它们在不同的文件 夹里即可。
确定
存在的任务创建，可以使用这个选项


源码管理
无
Git
Repositories
Repository URL
git@10.10.10.12:Jenkins/easy-springmvc.git
使用git的方式拉取
Credentials
root (key)
添加
刚才配置的凭据
高级...
Add Repository
Branches to build
指定分支（为空时代表any）
*/master
Add Branch



6、使用Publish over SSH的方式传输



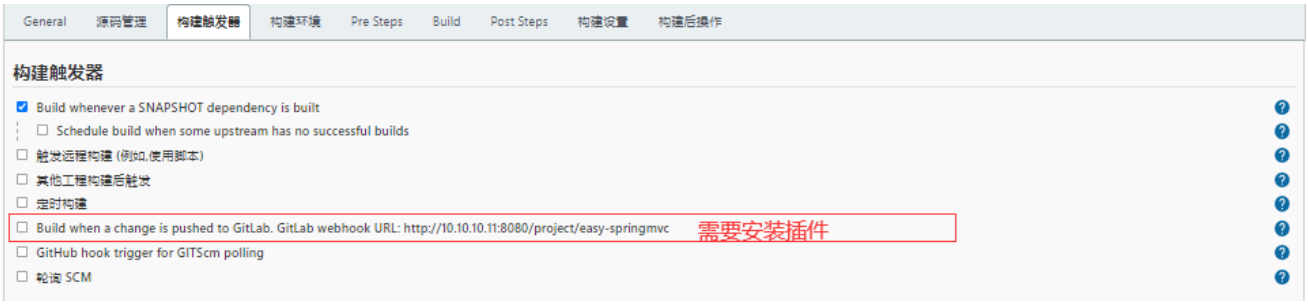
7、然后打开web服务器的网页，能够看到 网页就代表成功了

http://10.10.10.13:8080

之前的实验中我们都是通过手动点击构建来触发构建，下面就来讲一下利用触发器自动构建

六、触发器

1、在哪儿构建



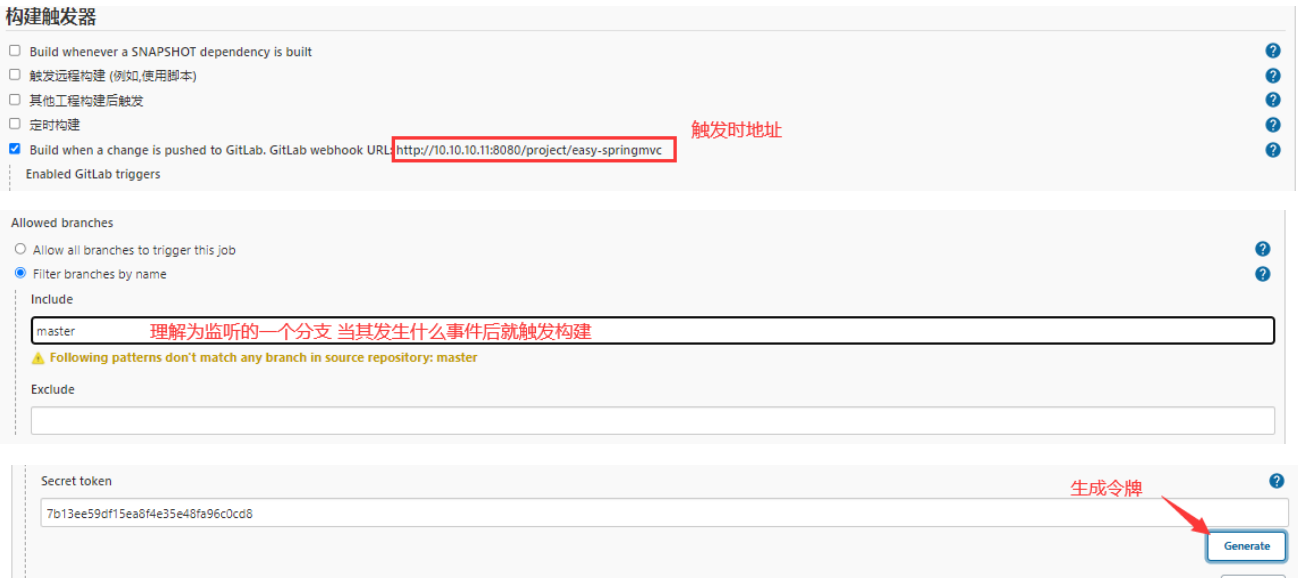
2、常见的触发方式

- 定时构建(Build perriodocally)，在一个固定的时间，无条件自动构建
- 轮询构建(SCM)，在定时构建的基础上增加了条件，会自动检查仓库是否有变化，有才构建
- Push事件触发，当向仓库的某个分支（一般为master）成功push代码后，就会触发构建
- 远程触发通过预定URL来触发，一般会结合脚本使用

3、Push事件触发器的使用

安装GitLab插件

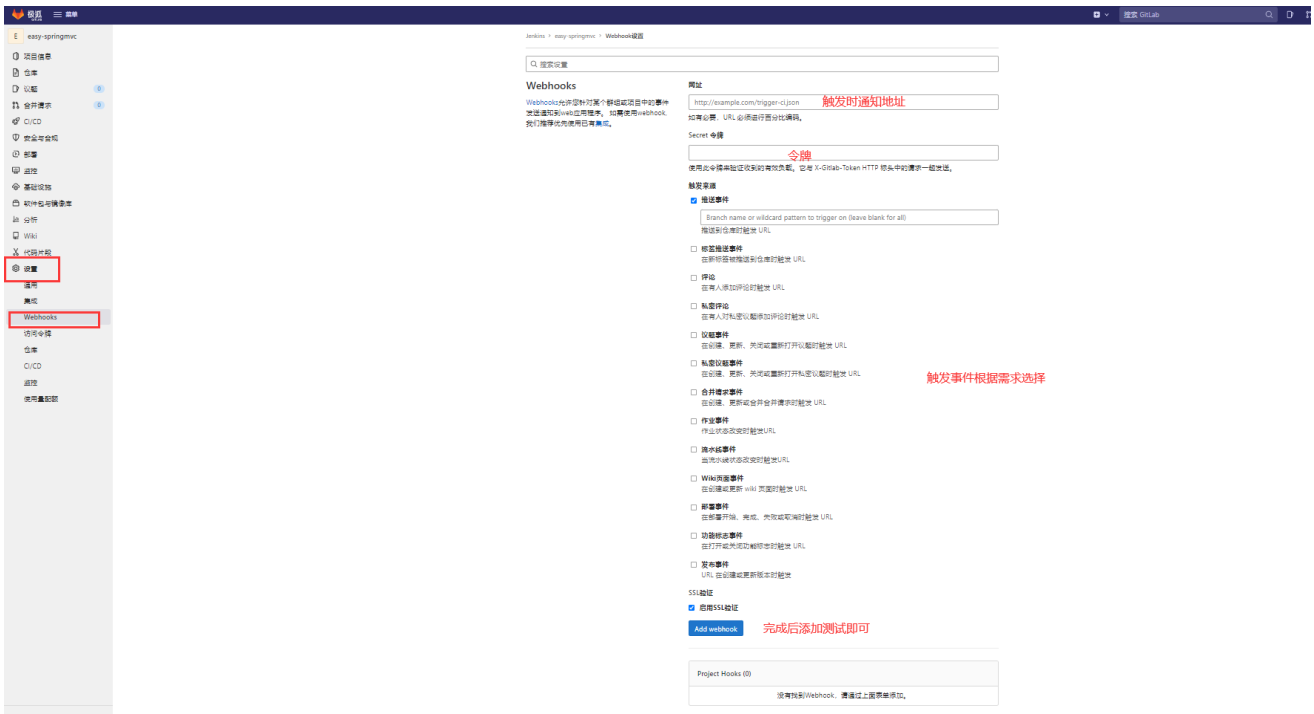
jenkins上设置：在任务中做设置，然后勾选上进行如下设置



在Gitlab上设置：

使用管理员身份登录，在设置中做如下设置

然后在项目中做如下设置



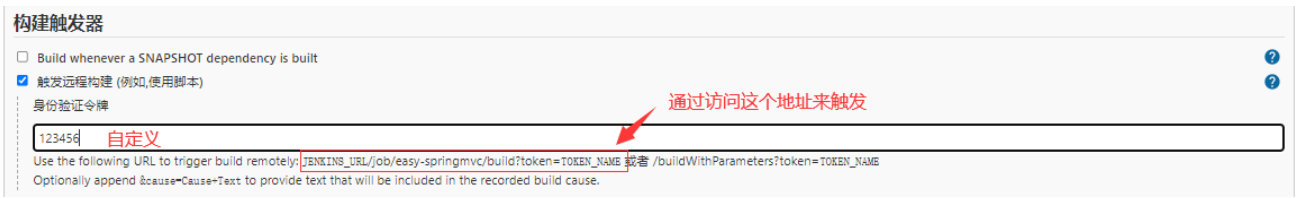
最后点击测试观察是否能够自动构建

4、远程触发

远程触发需要设置匿名用户具有可读权限

系统管理 - 全局安全配置 - 授权策略 - 匿名用户具有可读权限

然后在任务管理页添加触发器



通过访问：ip/job/easy-springmvc/build?token= 12346（身份令牌）

如果能够触发构建就代表成功了

通过构建触发器我们可以实现，自动化的构建了，但是不够灵活，不能实现版本的回退，于是就有了参数化构建

七、参数化构建

1、简介

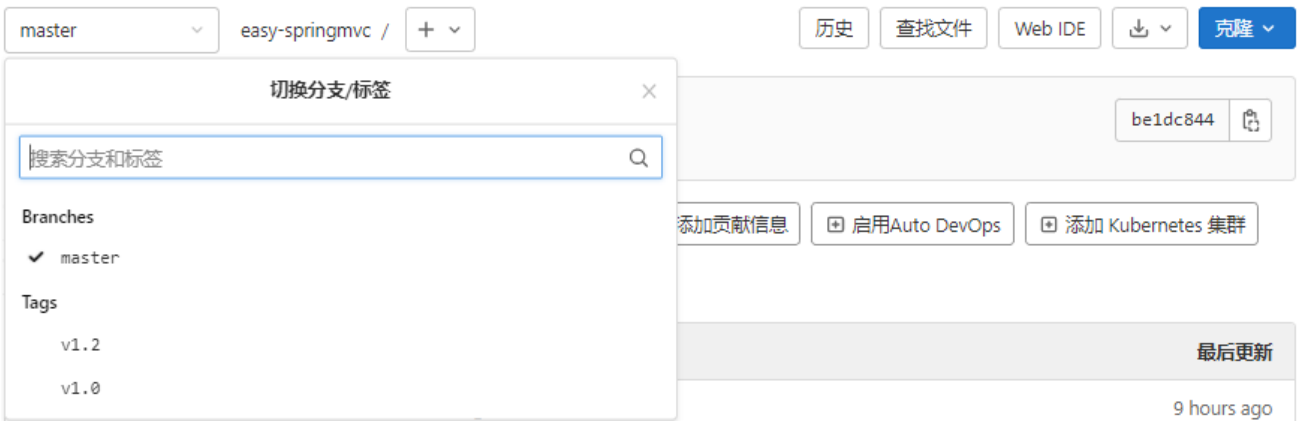
参数化构建，可以让我们实现定制化的任务

参数名称 = 变量值

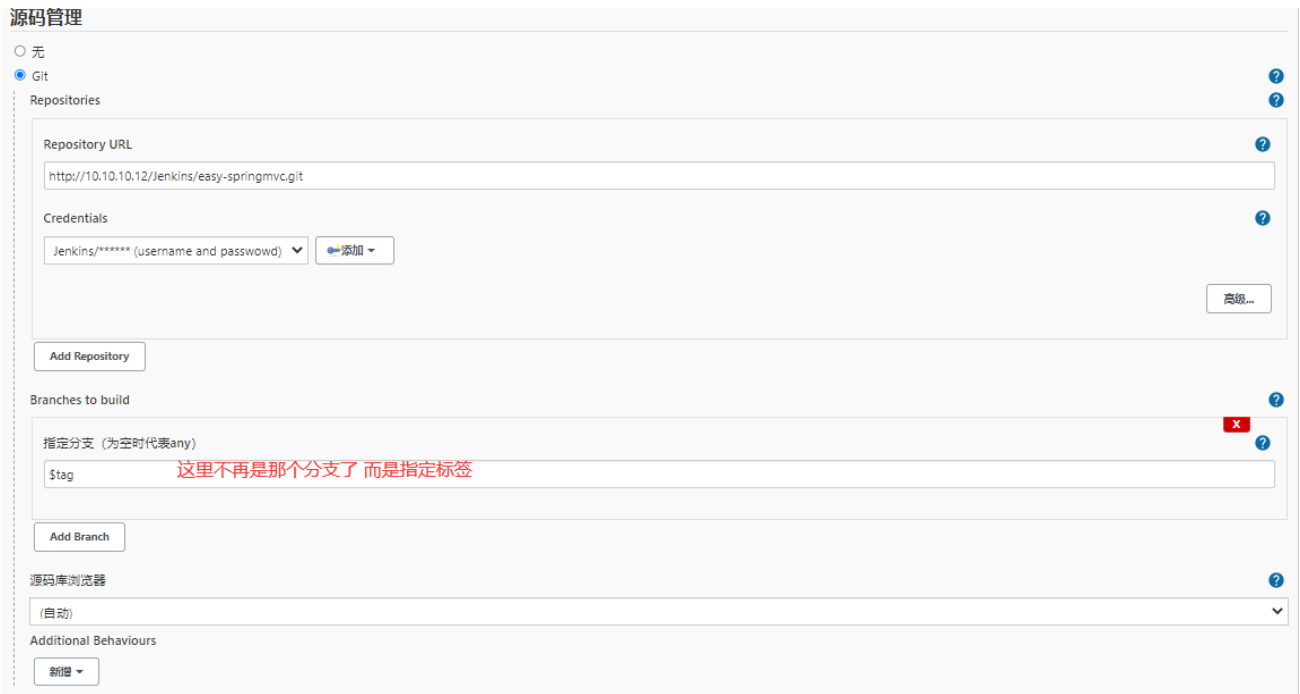
可以理解为我们定义了一个变量，在构建时给不同值，让jenkins为我们做不同的事，也可以利用在脚本中

2、利用参数化和tag实现版本的切换

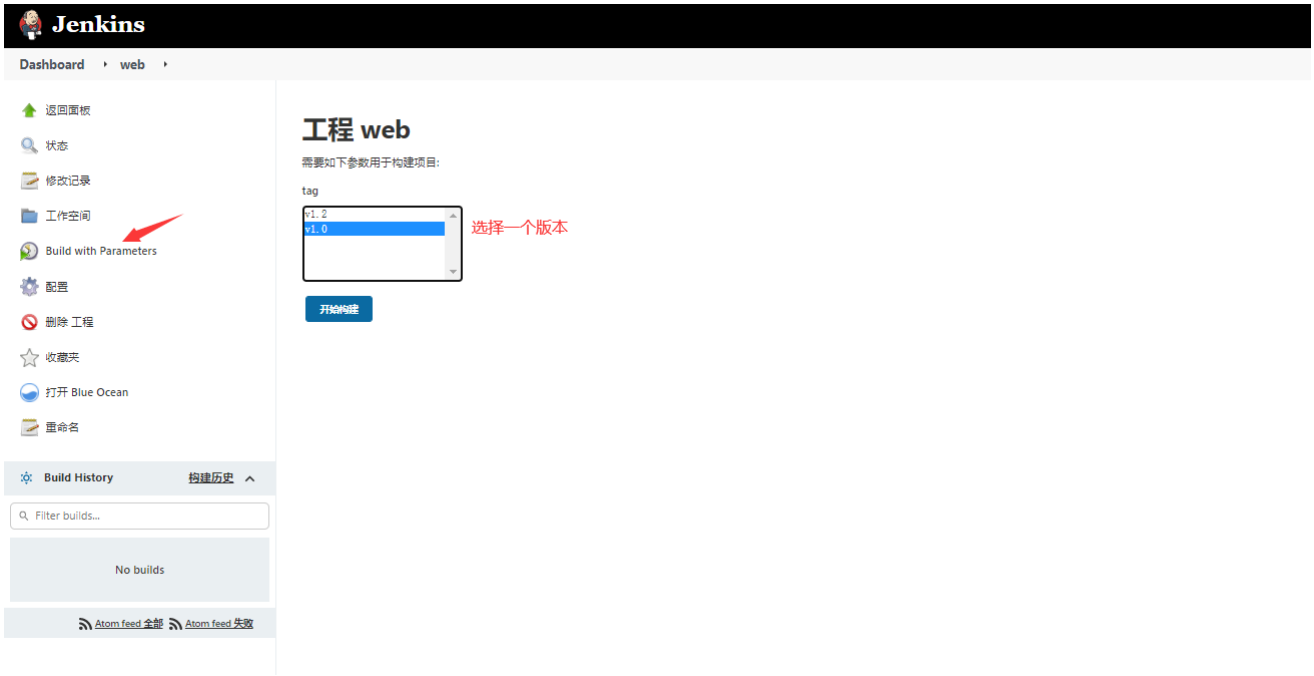
1）首先得有打标签的版本



2）简单配置一下



3）测试构建



八、流水线

流水线（pipline）是指按顺序连接在一起的事件或作业组

1、相关概念

pipline - 定义整个构建过程，通常包括构建应用程序、测试和交付应用程序的阶段。

node - 节点，执行流水线的机器

stage - 阶段，定义阶段性的任务，是多个step的子集

step - 步骤，定义单一的任务

3、创建的方法

1）通过Jenkinsfile语法来创建，通常有两种语法 一种是脚本式的 另一种是声明式的。声明式更容易上手。

2）还有一种方式就是通过Blue Ocean（蓝色海洋） 这个插件来创建，非常适合初学者

4、一个声明式语法的示例

```
node {
    stage('build'){
     echo 'build';
    }

    stage('test'){
     echo 'test';
    }

    stage('deploy'){
     echo 'deploy';
    }
}
```

无论以那种方式创建一个流水线后，都会在仓库中自动生成一个jenkinsfile的文件