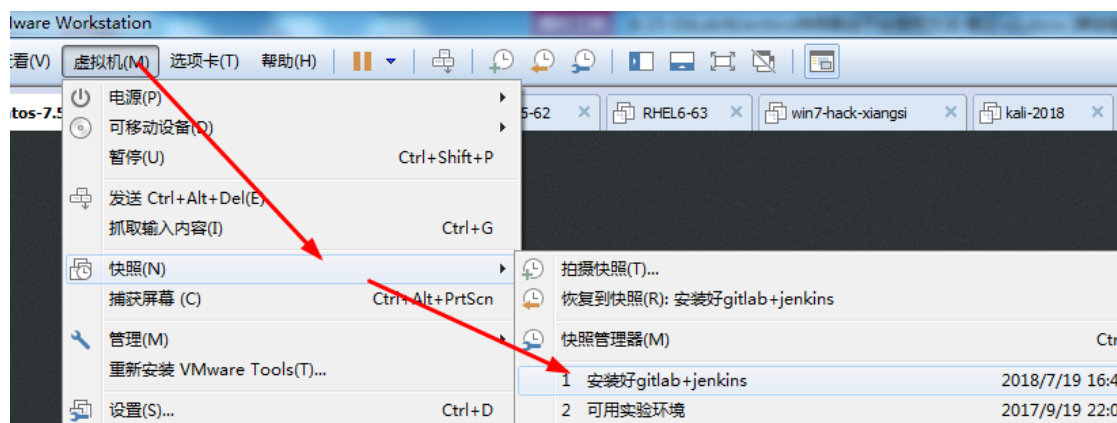


## 第十五章 GitLab 与 Jenkins 持续集成平台使用方法

本节所讲内容：

- 15.1 配置 Jenkins 使用 gitlab 更新代码
- 15.2 实现 gitlab 触发 Jenkins 自动部署
- 15.3 增加 Build Pipeline 插件以流程图的形式展示各个 Job 的顺序

准备实验环境：恢复到以下快照：



恢复好快照后，启动虚拟机

gitlab 和 jenkins 服务，我在之前已经做了开机启动，所以这里可以直接使用了。

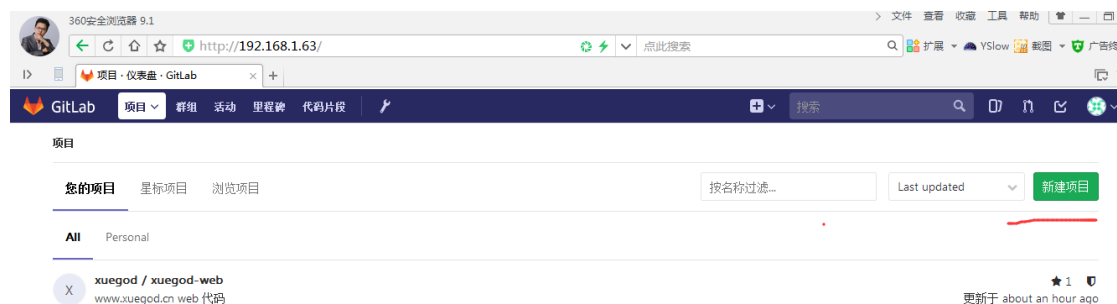
### 15.1 配置 Jenkins 使用 gitlab 更新代码

#### 15.1.1 查看 GitLab 上创建的项目：xuegod-web

登录 gitlab      <http://192.168.1.63/>      用户名：root 密码：xuegod.cn

登录 jenkins      <http://192.168.1.63:198/>      用户名：admin 密码：123456

查看之前创建的 xuegod-web 项目：



导出 xuegod 项目:

```
[root@xuegod63 ~]# rm -rf xuegod-web/
```

```
[root@xuegod63 ~]# git clone http://192.168.1.63/xuegod/xuegod-web.git
```

正克隆到 'xuegod-web'...

Username for 'http://192.168.1.63': jianmingbasic

Password for 'http://jianmingbasic@192.168.1.63': xuegod.cn

#### 15.1.2 安装 Jenkins 插件

安装以下插件：

Credentials Plugin (默认已经安装) #签名证书管理插件

Gitlab Plugin #安装后从 gitlab 获取代码

Git Plugin 和 Git Client Plugin ##用于 jenkins 在 gitlab 中拉取源码

GitLab Hook #gitlab 触发 jenkins 构建项目 #hook [hʊk] 钩子 ; plugin ['plʌɡɪn] 插件

gitlab Authentication # gitlab 和 jenkins 认证相关的插件

SSH Plugin #远程执行 shell 脚本

Publish Over SSH ##用于通过 ssh 部署应用

系统管理-> 插件管理-> 选择你需要的



安装插件：

过虑:

启用	名称 ↓	版本	上一个安装的版本	卸载
<input checked="" type="checkbox"/>	<a href="#">Authentication Tokens API Plugin</a> This plugin provides an API for converting credentials into authentication tokens in Jenkins.	1.3		<a href="#">卸载</a>
<input checked="" type="checkbox"/>	<a href="#">bouncycastle API Plugin</a> This plugin provides a stable API to Bouncy Castle related tasks.	2.16.3		<a href="#">卸载</a>
<input checked="" type="checkbox"/>	<a href="#">Command Agent Launcher Plugin</a> Allows agents to be launched using a specified command.	1.2		<a href="#">卸载</a>
<input checked="" type="checkbox"/>	<a href="#">Credentials Binding Plugin</a> Allows credentials to be bound to environment variables for use from miscellaneous build steps.	1.16		<a href="#">卸载</a>
<input checked="" type="checkbox"/>	<a href="#">Credentials Plugin</a> This plugin allows you to store credentials in Jenkins.	2.1.17		<a href="#">卸载</a>
<input type="checkbox"/>	<a href="#">Docker Commons Plugin</a>			

过虑:

安装 ↓	名称	版本
<input checked="" type="checkbox"/>	<a href="#">Publish Over SSH</a> Send build artifacts over SSH	1.19.1
<input checked="" type="checkbox"/>	<a href="#">SSH</a> Execute shell scripts on remote host using ssh (pre and post build). Based on the cool scp plugin.	2.6.1
<input type="checkbox"/>	<a href="#">Distributed Fork</a>	

过滤:

可更新

可选插件

已安装

高级

安装 ↓	名称	版本
<input type="checkbox"/>	<a href="#">GitHub Authentication</a> A Jenkins authentication plugin that delegates to GitHub. We also implement an Authorization Strategy that uses the acquired OAuth token to interact with the GitHub API to determine a user's level of access to Jenkins.	0.29
<input checked="" type="checkbox"/>	<a href="#">Gitlab Authentication</a> A Jenkins authentication plugin that delegates to gitlab. We also implement an Authorization Strategy that uses the acquired OAuth token to interact with the Gitlab API to determine a users level of access to Jenkins.	1.4
<input type="checkbox"/>	<a href="#">Gitcolony Build Notification</a> Updated live branches and pull requests build status into Gitcolony	1.1
<input type="checkbox"/>	<a href="#">GitHub SQS Build Trigger</a>	2.1
<input checked="" type="checkbox"/>	<a href="#">GitLab</a>	1.5.8

过滤:

可更新

可选插件

已安装

高级

安装 ↓	名称	版本
<input checked="" type="checkbox"/>	<a href="#">Email Extension Template</a>	1.0

直接安装

下载待重启后安装

34 分 之前获取了更新信息

立即获取

过滤:

可更新

可选插件

已安装

高级

安装 ↓	名称	版本
<input type="checkbox"/>	<a href="#">Gitlab Authentication</a> This is the an authentication plugin using gitlab OAuth.	1.4
<input checked="" type="checkbox"/>	<a href="#">GitLab</a> This plugin integrates <a href="#">GitLab</a> to Jenkins by faking a GitLab CI Server.	1.5.9
<input type="checkbox"/>	<a href="#">GitLab Logo</a> Display GitLab Repository Icon on dashboard	1.0.3
<input type="checkbox"/>	<a href="#">Gitlab Merge Request Builder</a> Integrates Jenkins with Gitlab to build Merge Requests	2.0.0
<input type="checkbox"/>	<a href="#">Violation Comments to GitLab</a> Finds violations reported by code analyzers and comments GitLab merge requests with them.	2.2
<input checked="" type="checkbox"/>	<a href="#">Gitlab Hook</a> Enables Gitlab web hooks to be used to trigger SMC polling on Gitlab projects <b>Warning: This plugin version may not be safe to use. Please review the following security notices:</b> <ul style="list-style-type: none"><li><a href="#">Gitlab API token stored and displayed in plain text</a></li></ul>	1.4.2

直接安装

下载待重启后安装

26 分 之前获取了更新信息

立即获取



#### 安装插件方式选择：

方法一.如果服务器可以上网,那边选择在线安装最好不过了,安装流程为: 系统管理(Configure System)---->插件管理(Manage Jenkins)--->选择需要的插件直接安装即可

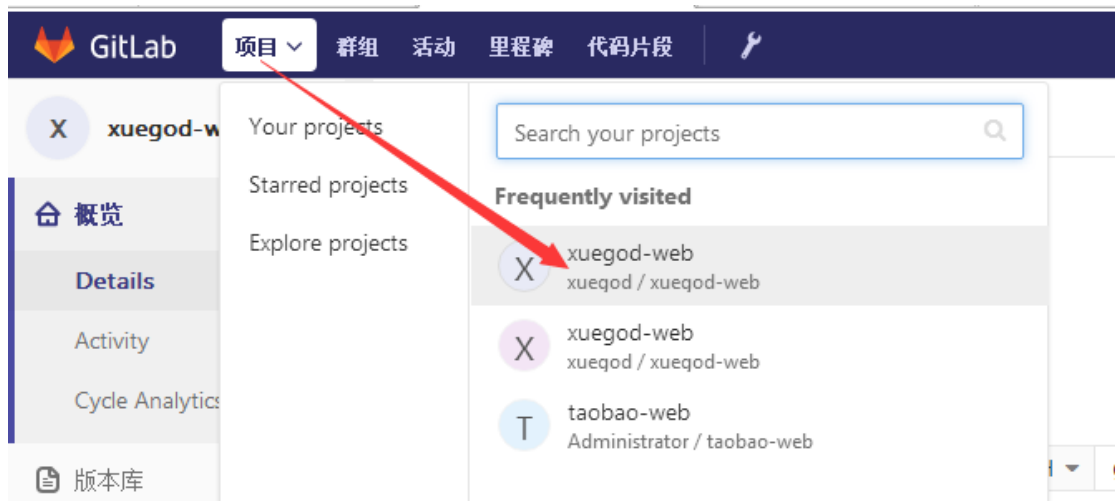
方法二.如果服务器不能上网,那么就只能离线安装,首先去

<http://updates.jenkins-ci.org/download/plugins/> 下载需要的 plugin,选择匹配的版本号,下载到本地,然后打开:系统管理()--->插件管理--->高级--->找到“上传插件”(浏览,找到扩展名为.hpi 的插件,上传之后默认直接就安装了。重启 jenkins, 安装的插件就可以使用了。安装插件依赖解决：

插件安装不上去,一定是某个地方出现问题。在 Jenkins 的终端日志中,可以查看到对应的错误。需要针对性的解决。比如：在安装 subversion 的时候,的错误提示。一般的情况安装对应的插件,会把需要依赖的插件安装上。如出现对应的依赖问题,就需要一一安装上对应的依赖包。

#### 15.1.3 设置 gitlab 用户 root 可以通过 ssh 公钥直接下载代码

部署 key, 让 root 用户可以不输入密码, 获取 gitlab 上的源代码



**[root@xuegod63 ~]# ssh-keygen #生成的公钥。 一路回车，不要输入保护私钥的密码。**

**[root@xuegod63 ~]# cat .ssh/id\_rsa.pub #查看公钥**

**ssh-rsa**

**AAAAB3NzaC1yc2EAAAADAQABAAQCAxKT6/F09mKFpyZuWJwSDzu2ZaB4Q975BRLIa  
La1aN4/95KmQQZKF4F9L+NeLvIEOPoZKDnxa2iSZXO2d2JNzualf/hZeJ3uJXnHGgELVZgh5v  
NKI2IP/iA5Q+Hx5LZ0fJKmje2n25EFEH7Nc5N7xc7Fkch6dDtWftf0SsyFaTgaAsAOe+nctRg5  
dyUhDss0+qV2a89Qd423qhHwnv+LmbDec2PR1SbQwweyjTV/DT3DvORG/eT28Lj876LyWm  
IjV+/vHgygbmIqDywF2WKnStgbXYqn6iJG4mCzry536LvP696vx63u0QI7yPyMsU1+yKhJPW  
LIARduXRrtwK25 root@xuegod63.cn**

**测试部署 key，发现可以不用用户名密码后直接获取代码：**

标题

root

密钥

ssh-rsa

AAAAB3NzaC1yc2EAAAADAQABAAQCAxKT6/F09mKFpyZuWJwSDzu2ZaB4Q975BRLIaLa1aN4/95KmQQZKF4F9L+NeLvIEOPoZK  
Dnxa2iSZXO2d2JNzualf/hZeJ3uJXnHGgELVZgh5vNKI2IP/iA5Q+Hx5LZ0fjKmje2n25EFEH7Nc5N7xc7Fkch6dDtWft0SsyFaTgaAsAOe+r  
ctRg5dyUhDss0+qV2a89Qd423qhHwnv+LmbDec2PR1SbQwwvejTV/DT3DvORG/eT28Lj876LyWmJjV+/vHgygbmIqDywf2WKnStgbX'  
qn6iJG4mCzry536LvP696vx63u0QI7yPyMsU1+yKhJPWLIARduXRrtwK25 root@xuegod63.cn

在这里粘贴机器公钥。如何生成密钥请点击 [这里](#)

☐ 允许推送

允许该密钥推送版本库? (默认权限为拉取。)

添加密钥

测试：

```
[root@xuegod63 ~]# rm -rf xuegod-web/
```

```
[root@xuegod63 ~]# git clone git@192.168.1.63:xuegod/xuegod-web.git #发现可以直接下载代码
```

#### 15.1.4 Jenkins 添加 git 用户 root 的私钥

复制 xuegod63 中 id\_dsa.pub 里面的公钥添加到 Jenkins ( private key 选项 )

```
[root@xuegod63 ~]# cat .ssh/id_rsa
```

The screenshot shows the Jenkins web interface. On the left, there is a sidebar with a menu containing '新建Item', '用户', '构建历史', '系统管理', 'My Views', '凭据', '系统', and '新建视图'. The '凭据' (Credentials) item is highlighted. In the main content area, the title is '凭据'. Below it, there is a table with columns 'T', 'P', '存储', and '域'. Under the table, it says '图标: S M L'. Below that, it says 'Stores scoped to Jenkins'. There is a table with one row: 'Jenkins' under the '存储' column and '(global)' under the '域' column. To the right of this table is a button labeled '添加凭据' (Add Credentials). A red arrow points from the '凭据' link in the sidebar to the '添加凭据' button.

**Credentials** [krəˈdenʃlɪz] 资格证书

类型 **SSH Username with private key**

范围 **全局 (Jenkins, nodes, items, all child items, etc)**

Username **root**

Private Key ☒ Enter directly

Key 

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAAnwMSk+vxDPZihacmblicEg87tmWgeEPe+QUSyGi2tWjeP/eS
pkEGSHeBfS/j67yBDj6GSg58WtokmVztndiTc7mpX/4W/d7IV5xxoBC1WYlebz
SiNiD/4gOUPh8eS2dHySpo3tp9uRBRB+zXOTe8XoxZHlenQ7Vn039ErMhWk4GgLA
Dnvp3LUYOXcllQ7LNPqldmVPUHeNt6oR8J7/i5mw3nNj0dUm0MMHso01fw09w7zk
Bv3k9vC4/O+i8inil1f7x4MnG5ik8sBdlin0YGY2Kq+niRuJos68ud+i7z+ve
-----
```

Passphrase

ID

描述

**确定**

注：这里直接复制/root/.ssh/id\_rsa 中的内容。

root 公钥在 gitlab , root 私钥在 jenkins , 这样 jenkins 就可以直接拉取 gitlab 上的代码  
全局凭证（不受限制） **unrestricted** [!^nr!^striktid] 无限制的



### Global credentials (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

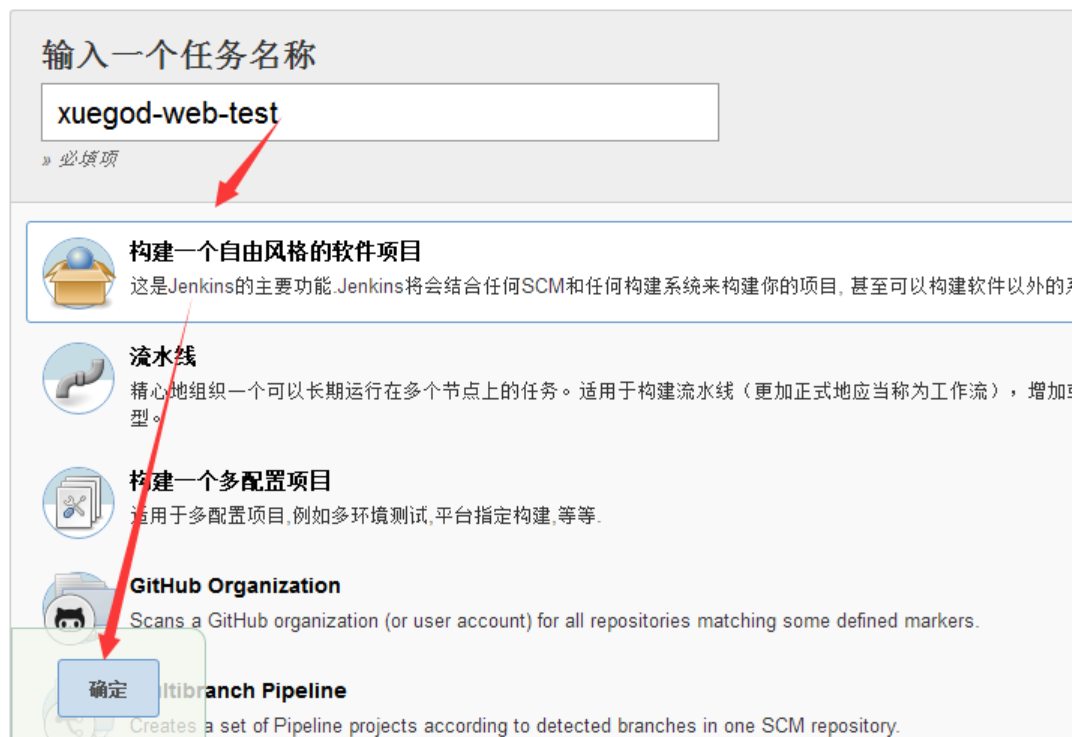
Name	Kind	Description
<a href="#">root</a>	SSH Username with private key	

图标: [S](#) [M](#) [L](#)

## 15.1.5 Jenkins 创建项目

新建任务：





xuegod-web 网站代码提交



**General** 源码管理 构建触发器 构建环境 构建 构建后操作

项目名称: xuegod-web-test

描述: xuegod-web 网站代码提交

[Plain text] 预览

☐ GitHub project

GitLab connection

☐ Throttle builds

☐ 丢弃旧的构建

☐ 参数化构建过程

☐ 关闭构建

保存 应用

配置适用 git 源：git@192.168.1.63:xuegod/xuegod-web.git

**General** 源码管理 构建触发器 构建环境 构建 构建后操作

☒ Git

Repositories

Repository URL: git@192.168.1.63:xuegod/xuegod-web.git

Credentials: root

高级...

Add Repository

Branches to build

Branch Specifier (blank for 'any'): \*/master

Add Branch

源码库浏览器: (自动)

Additional Behaviours: Add

Subversion

保存 应用

点保存。

查看状态：



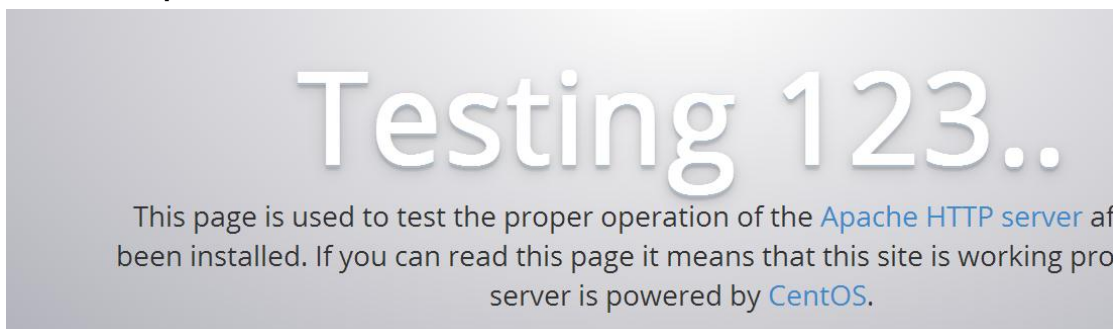
构建成功。

#### 15.1.6 将代码发布到 web 服务器

可以通过执行命令或脚本的方式进行代码发布，我使用 root 用户在各个 web 服务器上发布代码。大家也可以建立一些普通用户如果 www 或 apache 来发布。如果使用普通用户发布代码，要保持所有服务器上的用户 id 一致，这样没有权限的问题。

##### 安装 apache 服务

```
[root@xuegod63 yum.repos.d]# yum install httpd -y
[root@xuegod63 yum.repos.d]# vim /etc/httpd/conf/httpd.conf
改：Listen 80
为：Listen 81
[root@xuegod63 ~]# systemctl restart httpd
访问正常：http://192.168.1.63:81/
```



#### 15.1.7 准备自动部署 web 代码事项

1、实现 root 用户可以为无密码直接登录系统：

```
[root@xuegod63 ~]# ssh-copy-id root@192.168.1.63
[root@xuegod63 ~]# ssh 192.168.1.63
```

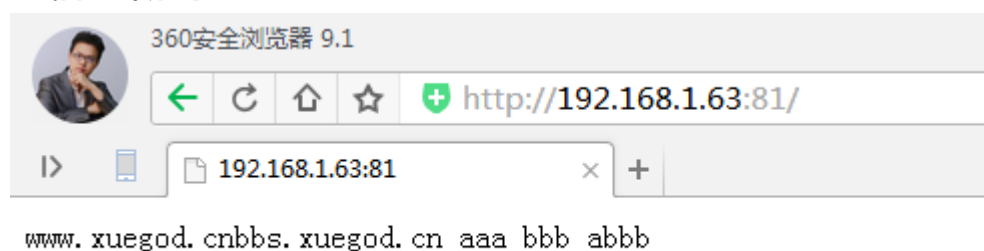
注：需要在哪台机器上自动发布代码，就需要让 jenkins 这样机器的 root 用户可以免费密码登录到那台服务器上。

## 2、配置 Jenkins 用户 sudo 权限并写

```
[root@xuegod63 ~]# visudo #最后添加
jenkins ALL=(ALL) NOPASSWD: /usr/bin/ssh
#不需要使用密码即可执行 ssh
```

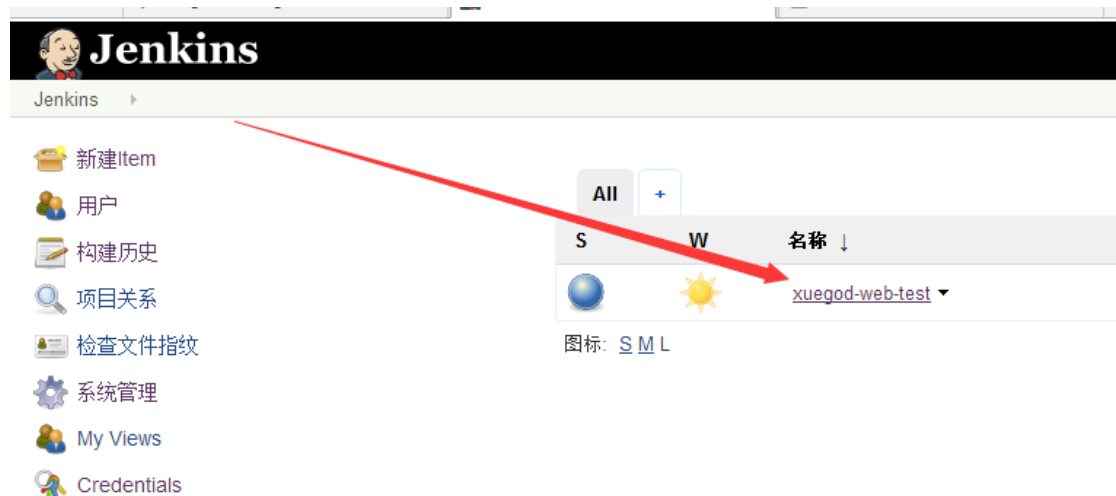
## 3、创建自动上传 web 代码的脚本

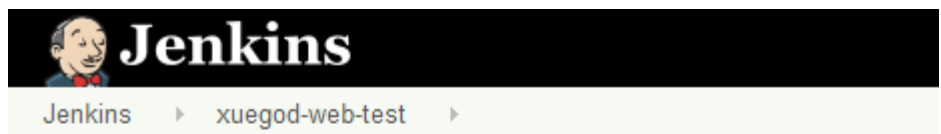
```
[root@xuegod63 ~]# vim deploy.sh #插入以下内容
#!/bin/bash
echo $USER
cd /root/xuegod-web
git pull
scp -r ./* root@192.168.1.63:/var/www/html/
[root@xuegod63 ~]# chmod +x deploy.sh
[root@xuegod63 ~]# ./deploy.sh #确保可以无交互执行成功
测试自动上传代码：
```



```
[root@xuegod63 ~]# rm -rf /var/www/html/index.html #把网页删除一下，一会还要测试
jenkins 部署
```

### 15.1.8 在 jenkins 的” 增加构建步骤 “写入上传代码脚本





返回面板

状态

修改记录

工作空间

立即构建

删除 Project

配置

## Project

xuegod-web 网站



Build History 构建历史

General 源码管理 构建触发器 **构建环境** 构建 构建后操作

☐ Poll SCM

### 构建环境

☐ Delete workspace before build starts

☐ Use secret text(s) or file(s)

☐ Send files or execute commands over SSH before the build starts

Execute Windows batch command

**Execute shell**

Execute shell script on remote host using ssh

Invoke Ant

Invoke Gradle script

Invoke top level Maven targets

Run with timeout

Send files or execute commands over SSH

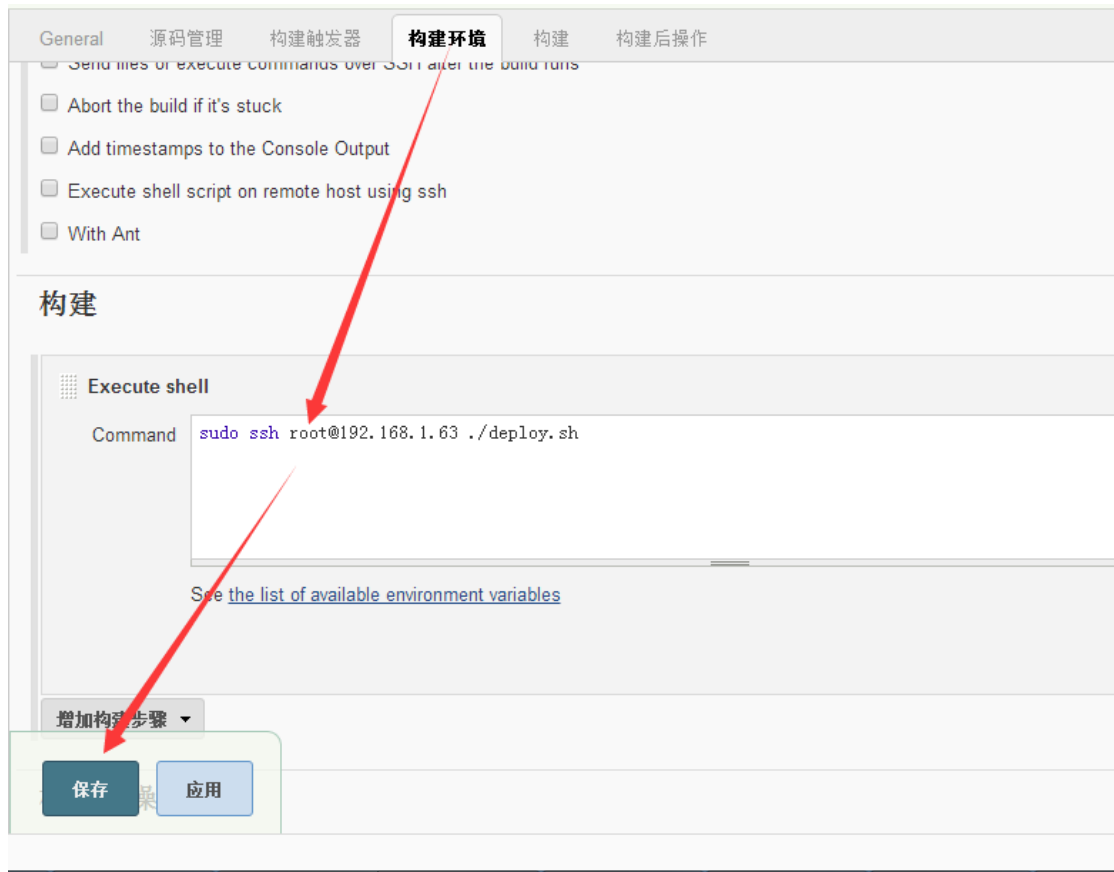
Set build status to "pending" on GitHub commit

增加构建步骤

### 构建后操作

保存 应用

```
sudo ssh root@192.168.1.63 ./deploy.sh
```



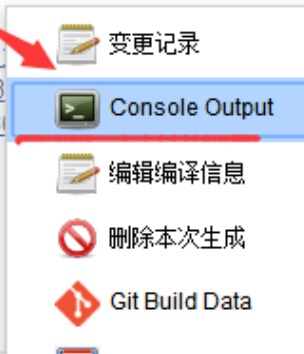
手动点立即构建：



查看输出的内容：

## 相关链接

- [Last build\(#1\).36 秒之前](#)
- [Last stable build\(#1\).36 秒之前](#)
- [Last successful build\(#1\).36 秒之前](#)
- [Last completed build\(#1\).36 秒之前](#)



```
Started by user mk
Building in workspace /var/lib/jenkins/workspace/xuegod-web-test
Cloning the remote Git repository
Cloning repository git@192.168.1.63:xuegod/xuegod-web.git
> git init /var/lib/jenkins/workspace/xuegod-web-test # timeout=10
Fetching upstream changes from git@192.168.1.63:xuegod/xuegod-web.git
> git --version # timeout=10
using GIT_SSH to set credentials
> git fetch --tags --progress git@192.168.1.63:xuegod/xuegod-web.git +refs/heads/*:refs/remotes/origi
> git config remote.origin.url git@192.168.1.63:xuegod/xuegod-web.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url git@192.168.1.63:xuegod/xuegod-web.git # timeout=10
Fetching upstream changes from git@192.168.1.63:xuegod/xuegod-web.git
using GIT_SSH to set credentials
> git fetch --tags --progress git@192.168.1.63:xuegod/xuegod-web.git +refs/heads/*:refs/remotes/origi
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision ec7684d9773f07367caf0ab0347a039315e58ba2 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f ec7684d9773f07367caf0ab0347a039315e58ba2
Commit message: "111"
First time build. Skipping changelog.
[xuegod-web-test] $ /bin/sh -xe /tmp/jenkins7737484825274992855.sh
+ sudo ssh root@192.168.1.63 ./deploy.sh
root
Already up-to-date.
Finished: SUCCESS
```

到此单台测试已经成功。

### 15.1.9 开始自动部署多台 web 服务器

启动 xuegod64.cn 虚拟机

```
[root@xuegod64 ~]# yum install httpd -y
```

```
[root@xuegod64 yum.repos.d]# vim /etc/httpd/conf/httpd.conf
```

改： 42 Listen 81

为： 42 Listen 80

```
[root@xuegod64 yum.repos.d]# systemctl restart httpd
```

```
[root@xuegod64 ~]# iptables -F
```

登录 xuegod63，上传到 root 用户公钥

```
[root@xuegod63 ~]# ssh-copy-id root@192.168.1.64
```

```
[root@xuegod63 ~]# ssh root@192.168.1.64 #测试可以直接登录成功
```

```
[root@xuegod63 ~]# vim deploy.sh #追加一行
#!/bin/bash
echo $USER
cd /root/xuegod-web
git pull
scp -r ./* root@192.168.1.63:/var/www/html/
scp -r ./* root@192.168.1.64:/var/www/html/
```

## 15.2 实现 gitlab 触发 Jenkins 自动部署

实战场景：在公司的测试环境当中，当开发人员向 gitlab 仓库提交代码后，gitlab 自动通知 jenkins 进行构建项目、代码质量测试然后部署至测试环境，这里先暂时部署到测试环境。对于生产环境，后期使用手动部署代码。

### 15.2.1 安装 Gitlab Hook Plugin 插件

#### 1、安装 Gitlab Hook Plugin 插件：

#系统管理-管理插件-可选插件，安装这两个插件：Gitlab Hook Plugin 和 Build Authorization Token Root 和 Build Token Trigger

过滤:

可更新	可选插件	已安装	高级	
启用	名称 ↓	版本	上一个安装的版本	卸载
<input checked="" type="checkbox"/>	<a href="#">bouncycastle API Plugin</a> This plugin provides an stable API to Bouncy Castle related tasks.	<a href="#">2.16.3</a>		<input type="button" value="卸载"/>
<input checked="" type="checkbox"/>	<a href="#">Command Agent Launcher Plugin</a> Allows agents to be launched using a specified command.	<a href="#">1.2</a>		<input type="button" value="卸载"/>
<input checked="" type="checkbox"/>	<a href="#">Git plugin</a> This plugin integrates <a href="#">Git</a> with Jenkins.	<a href="#">3.9.1</a>		<input type="button" value="卸载"/>
<input checked="" type="checkbox"/>	<a href="#">Gitlab Hook Plugin</a> Enables Gitlab web hooks to be used to trigger SMC polling on Gitlab projects	<a href="#">1.4.2</a>		<input type="button" value="卸载"/>
<input checked="" type="checkbox"/>	<a href="#">ruby-runtime</a> Hosts runtime for enabling pure-Ruby plugins	<a href="#">0.12</a>		<input type="button" value="卸载"/>

过滤:

可更新	可选插件	已安装	高级
安装 ↓	名称	版本	
<input checked="" type="checkbox"/>	<a href="#">Build Authorization Token Root</a> Lets build and related REST build triggers be accessed even when anonymous users cannot see Jenkins.	1.4	
<input checked="" type="checkbox"/>	<a href="#">Build Token Trigger</a> This plugin provides a pipeline step to trigger a build using the <a href="#">Build Authorization Token Root</a> plugin	1.0.0	

1 小时 27 分 之前获取了更新信息

### 15.2.2 配置身份验证令牌

生成 Token ， openssl rand 生成随机密码

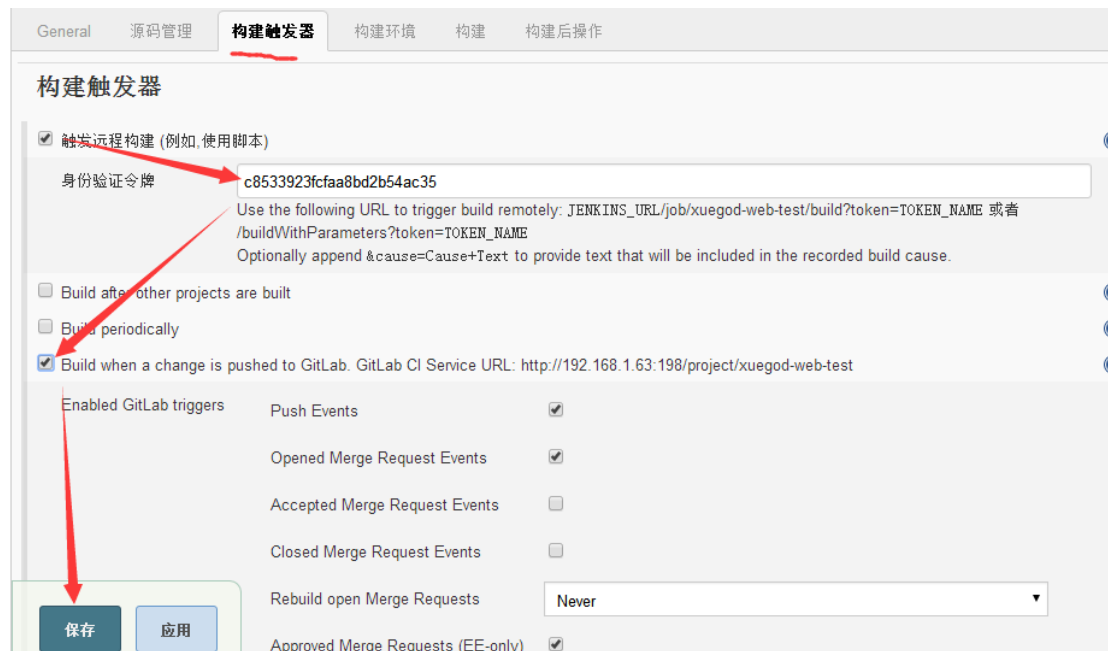
```
[root@xuegod63 ~]# openssl rand -hex 12
c8533923fcfaa8bd2b54ac35
```

注：-hex：输出结果为 16 进制数据；数字 12 是指定生成的随机字符为 12 字节，即 24 位 16 进制数

配置触发器 <http://192.168.1.63:198/job/xuegod-web-test/configure>



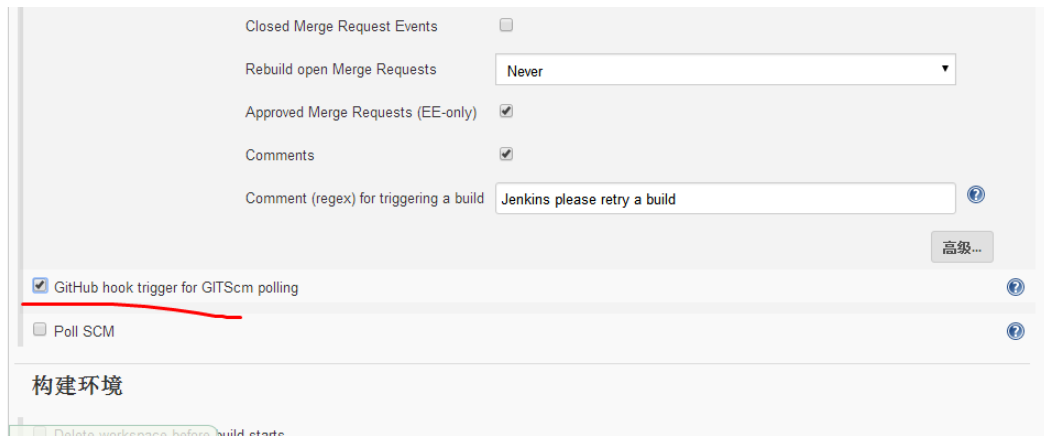
身份验证令牌：c8533923cf8a8bd2b54ac35



<http://192.168.1.63:198/project/xuegod-web-test>

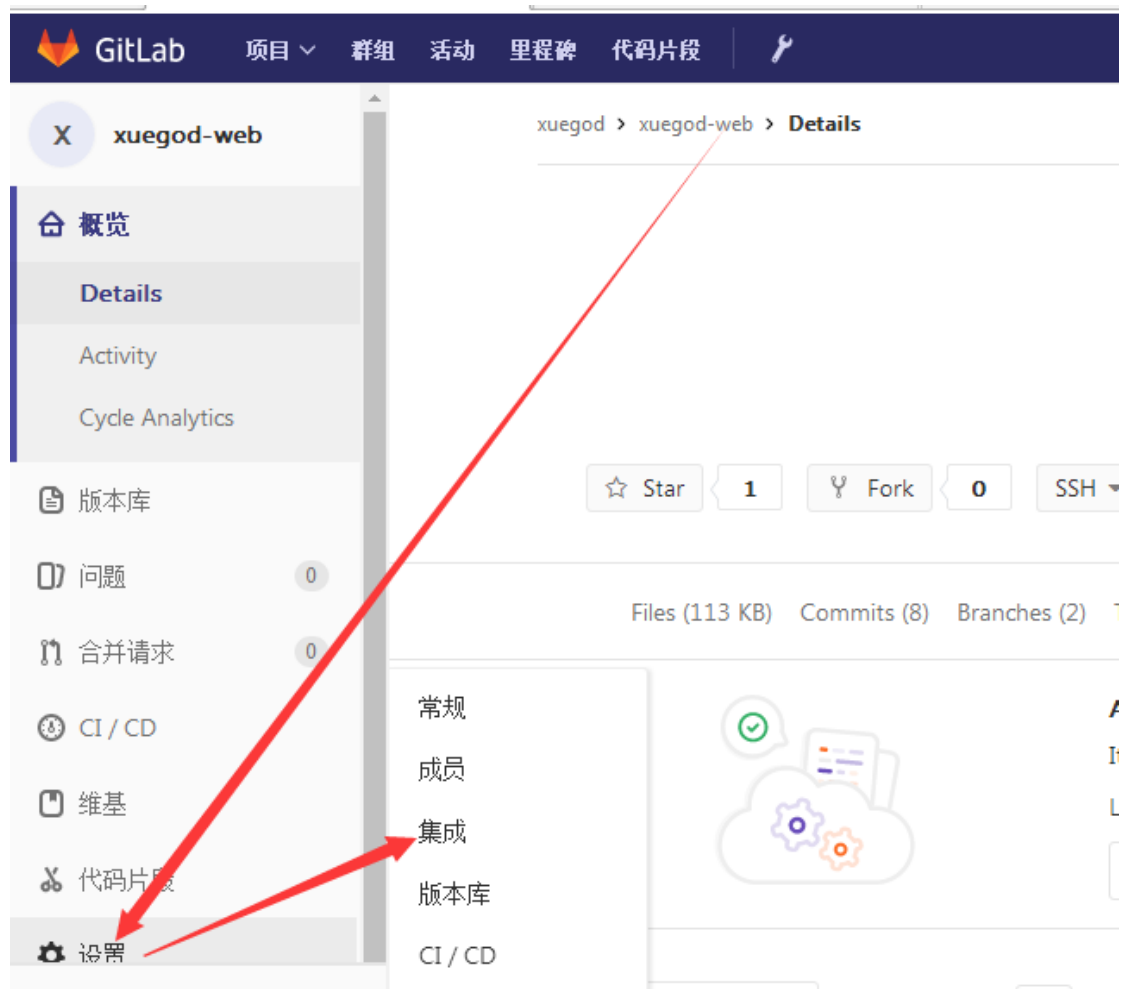
构建触发器，勾选 “GitHub hook trigger for GITScm polling”





### 15.2.2 在 git 项目配置界面设置链接和 token

登录 gitlab,在这个项目下找到 web 钩子配置的地方

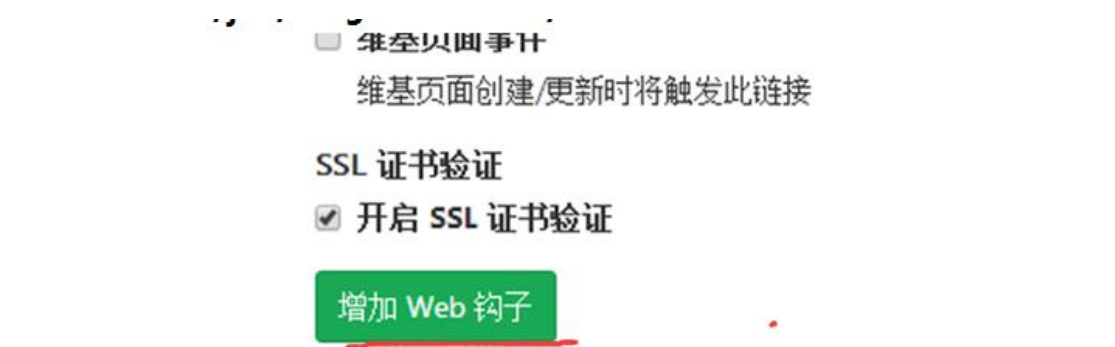




使用这个：

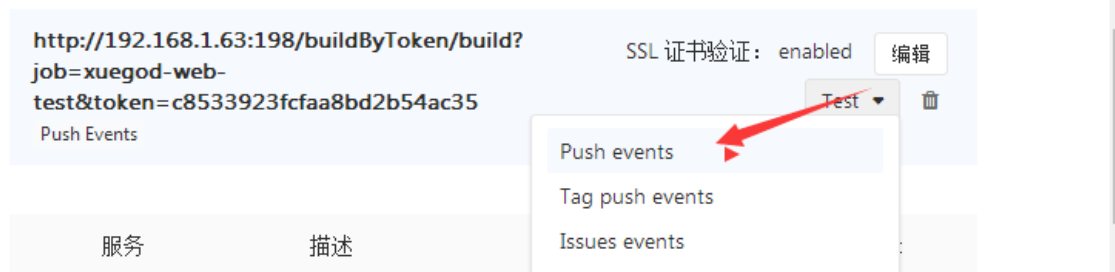
<http://192.168.1.63:198/buildByToken/build?job=xuegod-web-test&token=c8533923fcfaa8bd2b54ac35>

格式：[http://jenkins 地址/buildByToken/build?job=jenkins 项目名称&token=token 值](#)

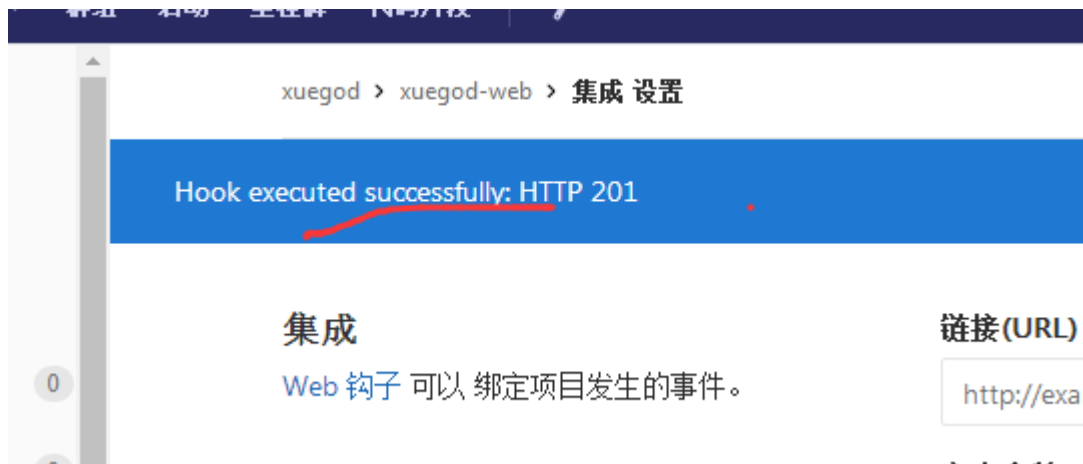


测试：

Web 钩子 (1)



返回这个，说明成功了：



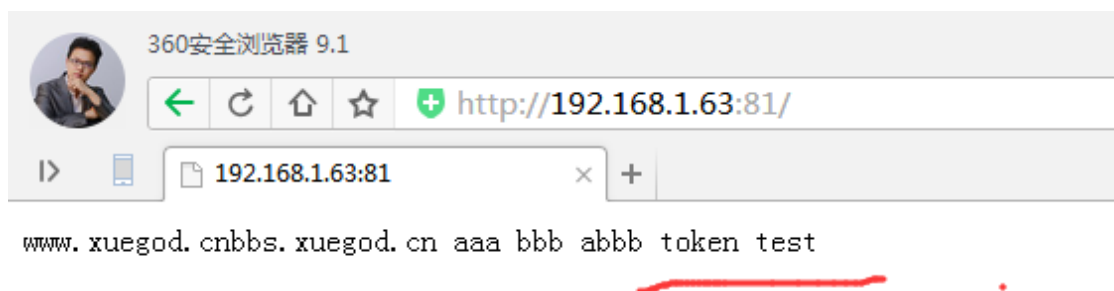
### 15.2.2 测试 gitlab 触发 jenkins 自动部署

向 git 服务器提交代码，验证是否可以自动部署：

```
[root@xuegod63 xuegod-web]# cd /root/xuegod-web/  
[root@xuegod63 xuegod-web]# echo "token test" >> index.html  
[root@xuegod63 xuegod-web]# git add index.html  
[root@xuegod63 xuegod-web]# git commit -m "modify index.html"  
[root@xuegod63 xuegod-web]# git push -u origin master
```

访问 web 界面验证代码是否最新的：

http://192.168.1.63:81/ 已经是最新了



总结步骤：gitlab 创建代码项目-》gitlab 部署公钥-》jenkins 安装插件-》创建新项目-》绑定私钥-》添加触发器-》测试部署

### 15.3 增加 Build Pipeline 插件以流程图的形式展示各个 Job 的顺序

安装 Build Pipeline 插件，Build Pipeline 插件作用：这个插件能够以流程图的形式展示各个 Job 的顺序，依赖关系等等。



# Jenkins

Jenkins

[新建Item](#)

[用户](#)

[构建历史](#)

[项目关系](#)

[检查文件指纹](#)

[系统管理](#)

[My Views](#)

[Credentials](#)

[新建视图](#)

## 管理Jenkins

**Jenkins新版本 (2.13)**  
**Warnings have been detected**

- Jenkins 2.93.0
  - [Multiplatform support](#)
  - [Multiplatform support](#)
  - [Multiplatform support](#)
  - [Multiplatform support](#)
- [GitHub plugin](#)
  - [Server-side](#)
  - [CSRF protection](#)
  - [CSRF protection](#)
- [Credentials](#)

[Configure the credential providers and types](#)

[全局工具配置](#)  
工具配置，包括它们的位置和自动安装器

[读取设置](#)  
放弃当前内存中所有的设置信息并从配置文件中重新读取 仅用于当您手动修改配置文件时重新读取

[管理插件](#)  
添加、删除、禁用或启用Jenkins功能扩展插件。 **(可用更新)**

[系统信息](#)  
显示系统环境信息以帮助解决问题。

[系统日志](#)

## Build Pipeline

过滤:

可更新 可选插件 已安装 高级

安装

	名称	版本
<input checked="" type="checkbox"/>	Build Pipeline	1.5

直接安装

下载待重启后安装

1 day 0 小时 之前获取了更新信息

立即获取

添加一个视图，配置项目 `xuegod-web-test` 构建完成后自动构建 `xuegod-web` 项目：



- 新建Item
- 用户
- 构建历史
- 项目关系
- 检查文件指纹
- 系统管理

All + .

S	W	名称 ↓
		<a href="#">xuegod-web-test</a>

图标: [S](#) [M](#) [L](#)

视图名称： xuegod-web-pipeline

视图名称

☒ **Build Pipeline View**  
Shows the jobs in a build pipeline view. The complete pipeline of jobs that a version propagates through are shown as a row in the view.

☐ 我的视图  
该视图自动显示当前用户有权访问的任务

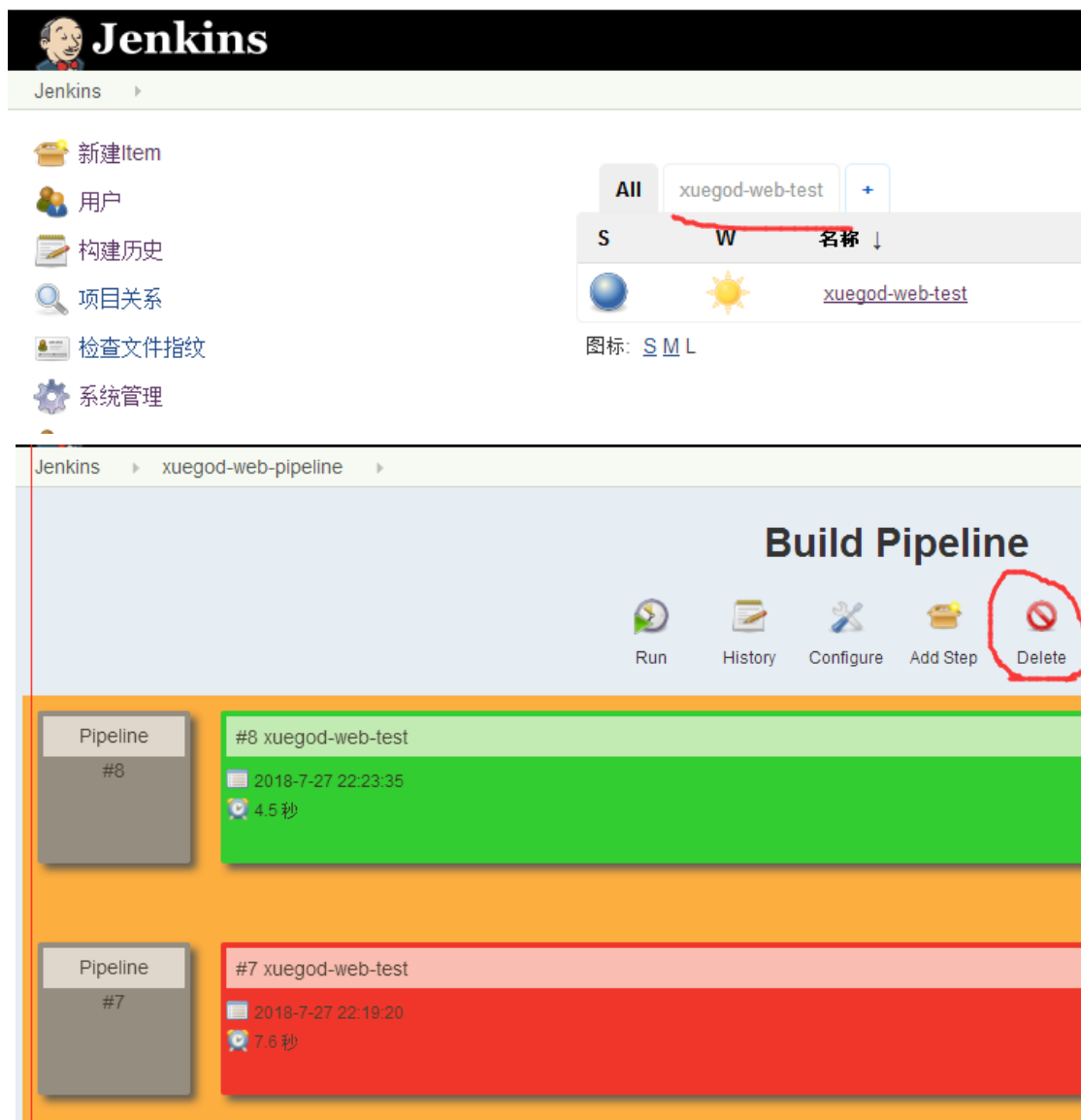
☐ 简单视图  
显示简单列表。你可以从中选择任务来显示在某个视图中

显示最近 3 次构建的结果：

Display Options

No Of Displayed Builds	<input type="text" value="3"/> ?
Row Headers	<div>Just the pipeline number ▼ Show just the build pipeline number</div>
Column Headers	<div>No header ▼ Do not show any column headers</div>
Refresh frequency (in seconds)	<input type="text" value="3"/> ?
URL for custom CSS files	<input type="text"/>
Console Output Link Style	<div>Lightbox ▼</div>

查看效果：



总结：

- 15.1 配置 Jenkins 使用 gitlib 更新代码
- 15.2 实现 gitlib 触发 Jenkins 自动部署
- 15.3 增加 Build Pipeline 插件以流程图的形式展示各个 Job 的顺序