

第0章 Jenkins介绍

- ```
1 | 1.jenkins是一个开源的持续集成工具，由java语言开发
2 | 2.jenkins是一个调度平台，拥有众多的插件，绝大部分功能都是由插件来完成的
```

# 第1章 Jenkins安装

## 1.官方网站

- ```
1 | https://www.jenkins.io/zh/  
2 | https://www.jenkins.io/zh/doc/
```

2.安装部署

清华源直接下载rpm包安装即可,下载地址如下:

- ```
1 | https://mirrors.tuna.tsinghua.edu.cn/jenkins/redhat/
```

安装命令如下:

- ```
1 | rpm -ivh jdk-8u181-linux-x64.rpm  
2 | rpm -ivh jenkins-2.176.1-1.1.noarch.rpm
```

3.目录文件说明

- ```
1 | [root@jenkins ~]# rpm -ql jenkins
2 | /etc/init.d/jenkins #启动文件
3 | /etc/logrotate.d/jenkins #日志切割脚本
4 | /etc/sysconfig/jenkins #配置文件
5 | /usr/lib/jenkins #安装目录
6 | /usr/lib/jenkins/jenkins.war #安装包
7 | /usr/sbin/rcjenkins
8 | /var/cache/jenkins
9 | /var/lib/jenkins #数据目录
10 | /var/log/jenkins #日志目录
```

## 4.配置使用root账户运行

- ```
1 | vim /etc/sysconfig/jenkins  
2 | JENKINS_USER="root"
```

5.启动jenkins

```
1 | systemctl start jenkins
```

6.解锁Jenkins

入门

解锁 Jenkins

为了确保管理员安全地安装 Jenkins，密码已写入到日志中（[不知道在哪里？](#)）该文件在服务器上：

`/var/lib/jenkins/secrets/initialAdminPassword`

请从本地复制密码并粘贴到下面。

管理员密码

7.修改admin密码

The screenshot shows the Jenkins user configuration interface for the 'admin' user. The top navigation bar includes a profile icon, the word 'Jenkins', the user name 'admin', and a 'log out' link. A red box highlights the 'Configure' link in the dropdown menu. The main content area displays the user's details: 'Full Name' is set to 'admin', and there is a 'Description' field. Below this is the 'API Token' section, which states 'There are no registered tokens for this user.' and features a 'Add new Token' button. The 'My Views' section shows a 'Default View' field with a placeholder 'The view selected by default when navigating to the user's private views'. At the bottom, there is a 'Password' section with fields for 'Password' and 'Confirm Password', both of which are currently masked with dots. A red box highlights the 'Confirm Password' field.

8.使用清华源作为插件地址

清华源地址：

```
1 | https://mirrors.tuna.tsinghua.edu.cn/jenkins/updates/update-center.json
```

配置步骤：

Jenkins

-  New Item
-  People
-  Build History
-  Manage Jenkins

-  My Views
-  New View

-  Manage Plugins

Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

Jenkins > Plugin Manager

-  Back to Dashboard
-  Manage Jenkins
-  Update Center

Updates Available Installed Advanced

HTTP Proxy Configuration

Update Site

URL

Submit

9. 使用离线安装插件

我们可以将插件提前下好，然后只需要解压到jenkins对应的目录即可

```
1 | tar zxf jenkins_plugins.tar.gz -C /var/lib/jenkins/  
2 | ll /var/lib/jenkins/plugins/
```

重启jenkins：

```
1 | systemctl restart jenkins
```

第2章 构建自由风格的项目

1. 创建新任务

The screenshot shows the Jenkins dashboard with a large watermark reading "深圳对外光" diagonally across it. On the left, there's a sidebar with various links: "新建Item", "用户列表", "构建历史", "Manage Jenkins", "My Views", "打开 Blue Ocean", "Lockable Resources", "凭据", and "New View". In the center, a prominent message says "欢迎来到 Jenkins!" with a red-bordered box containing the text "开始创建一个新任务。". At the bottom, there's a form for creating a new item with the placeholder "输入一个任务名称" and a text input field containing "my-freestyle-job" which is also highlighted with a red border. Below this, a section titled "Freestyle project" with a description: "This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build."

描述

测试

[Plain text] 预览

JIRA site

Discard old builds

策略 Log Rotation

保持构建的天数

如果非空，构建记录将保存此天数

保持构建的最大个数

如果非空，最多此数目的构建记录将被保存

高级...



2.添加构建步骤

构建

增加构建步骤 ▾

Execute SonarQube Scanner

Execute Windows batch command

Execute shell

Invoke Ant

Invoke Gradle script

Invoke top-level Maven targets

JIRA: Add related environment variables to build

JIRA: Create new version

JIRA: Issue custom field updater

JIRA: Mark a version as Released

JIRA: Progress issues by workflow action

e-job/configure#

构建

The screenshot shows the Jenkins build configuration page. At the top, there's a section titled "Execute shell" with a "命令" field containing "pwd". A red box highlights the "pwd" text. Below this is a link "查看 可用的环境变量列表". To the right is a "高级..." button. At the bottom left is a "增加构建步骤 ▾" button. On the far right are "保存" and "应用" buttons, with "保存" also highlighted by a red box.

3.点击立即构建

The screenshot shows the Jenkins project page for "Project my-freestyle-job". The left sidebar has a "Build Now" button highlighted by a red box. Other options in the sidebar include "返回面板", "状态", "修改记录", "工作空间", "删除 Project", "配置", "收藏夹", "打开 Blue Ocean", and "重命名". The main area displays the project name and some links like "工作区" and "最新修改记录". At the bottom, there's a "Build History" section with a search bar and RSS links for "全部" and "失败".

4.查看控制台输出



Build History

构建历史 -

find X



#1

2020-5-12 下午6:16



变更记录



控制台输出



编辑编译信息



删除构建 '#1'



Changes since last success



打开 Blue Ocean



控制台输出

```
Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/my-freestyle-job
[my-freestyle-job] $ /bin/sh -xe /tmp/jenkins4787290991829802901.sh
+ pwd
/var/lib/jenkins/workspace/my-freestyle-job
Finished: SUCCESS
```

1.gitlab导入工程

这是一个h5小游戏的项目，项目地址：

1 | <https://gitee.com/skips/game.git>

使用gitlab直接导入项目：

The screenshot shows the GitLab web interface. At the top, there's a navigation bar with links for 'Projects', 'Groups', 'More', and search functions. Below the navigation is a section titled 'Projects' with a 'New project' button highlighted by a red box. Underneath, there are sections for 'Your projects' (2), 'Starred projects' (0), and 'Explore projects'. A filter bar allows searching by name and sorting by 'Last updated'. The main list displays two projects: 'ops / ansible' (Maintainer) and 'dev / game' (Maintainer). Both projects have 0 stars and 0 issues, and were last updated 10 hours ago and 11 hours ago respectively.

This screenshot shows the 'New project' creation interface. It features a 'Blank project' and 'Create from template' option at the top. Below that is a section for 'Import project from' with various options: 'GitLab export', 'GitHub', 'Bitbucket Cloud', 'Bitbucket Server', 'GitLab.com', 'Google Code', 'Fogbugz', 'Gitea', 'git Repo by URL' (which is highlighted with a red box), and 'Manifest file'. A large red watermark '禁止转载' is overlaid across the entire interface. In the top right corner, there's a prominent 'Import project' button highlighted with a red box.

Import project from

GitLab export GitHub Bitbucket Cloud Bitbucket Server GitLab.com

Google Code Fogbugz Gitea git Repo by URL Manifest file

从git仓库的URL导入

Git repository URL

gitee 的项目地址

Username (optional)

Password (optional)

- The repository must be accessible over `http://`, `https://` or `git://`.
- If your HTTP repository is not publicly accessible, add your credentials.
- The import will time out after 180 minutes. For repositories that take longer, use a clone/push combination.
- To import an SVN repository, check out [this document](#).

Project name

项目名称

Project URL

选择组

Project slug

自动补充的项目名

Want to house several dependent projects under the same namespace? [Create a group](#).

Project description (optional)

Description format

Visibility Level

 Private

私有项目

Project access must be granted explicitly to each user.

The screenshot shows a GitLab interface. At the top, there's a navigation bar with links for 'Projects', 'Groups', 'More', and search functionality. Below the navigation is a sidebar with various icons. The main content area shows a message: 'Project 'h5game' was successfully created.' followed by 'Import in progress'. It includes a command: 'git clone --bare https://gitee.com/skips/game.git' and a note: 'Please wait while we import the repository for you. Refresh at will.' The project page for 'h5game' is displayed, showing a commit history with one commit from 'oldya' (17 hours ago) that deleted a file named 'del'. There are buttons for 'History', 'Find file', 'Web IDE', and a copy icon. Below the commit history is a table of files: README, CHANGELOG, CONTRIBUTING, Auto DevOps, Kubernetes cluster, Set up CI/CD, LICENSE, and README.md. The last commit information is shown for each file. A large watermark reading '项目对接' (Project Integration) is overlaid across the entire page.

Project 'h5game' was successfully created.

Import in progress

```
git clone --bare https://gitee.com/skips/game.git
```

Please wait while we import the repository for you. Refresh at will.

h5game

Project ID: 5

LICENSE 4 Commits 1 Branch 0 Tags 0 Bytes Files

master h5game / +

History Find file Web IDE

del oldya authored 17 hours ago bbebb4fa

README Add CHANGELOG Add CONTRIBUTING Add Auto DevOps Add Kubernetes cluster

Add CI/CD

Name	Last commit	Last update
game	add game	17 hours ago
LICENSE	add LICENSE.	17 hours ago
README.md	Initial commit	17 hours ago

2.在jenkins中关联gitlab的h5game项目

2.1 创建新项目

- 新建Item
- 用户列表
- 构建历史
- 项目关系
- 检查文件指纹
- Manage Jenkins
- My Views
- 打开 Blue Ocean
- Lockable Resources
- 凭据
- New View

欢迎来到 Jenkins!

开始创建一个新任务。

输入一个任务名称

» 必填项

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

2.2 填写仓库地址

选择源码管理，然后填写gitlab仓库信息，但是我们发现报错了，因为jenkins没有拉取gitlab项目的权限。

General 源码管理 构建触发器 构建环境 构建 构建后操作

源码管理

无 Git

git 仓库地址

Repositories

Repository URL: git@10.0.0.200:dev/h5game.git

Failed to connect to repository : Command "git ls-remote -h git@10.0.0.200:dev/h5game.git HEAD" returned status code 128:
stdout:
stderr: Permission denied, please try again.
- Permission denied, please try again.
Permission denied (publickey,password).
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.

Credentials - 无 - 添加 高级... Add Repository

报错了！因为jenkins没有从gitlab拉取代码的权限
所以，我们需要把jenkins服务器的公钥放在gitlab上

3.配置jenkins访问gitlab的权限

3.1 部署公钥解释和步骤

解释

1. 如果我们想让jenkins从gitlab上拉取代码，那么需要将jenkins的公钥信息放在gitlab上。
2. gitlab针对这种情况有一个专门的功能，叫做部署部署公钥。
3. 部署公钥的作用是不需要创建虚拟用户和组，直接在需要拉取的项目里关联部署公钥即可。

步骤

1. 获取jenkins公钥信息
2. 将jenkins公钥信息填写到gitlab的部署公钥里
3. 由项目管理员操作，在需要jenkins拉取的项目里关联部署公钥
4. jenkins配置私钥凭证，部署项目时关联凭证

3.2 获取jenkins服务器的SSH公钥信息

```
1 [root@jenkins-201 ~]# cat .ssh/id_rsa.pub
2 ssh-rsa
AAAAB3NzaC1yc2EAAAQABAAQCG8+DQFOjR+g1Xw83CIyGJ50vI4DBeTaMRFdu5+5pT/IMnYq1iS7/
lRS6JxXLYvVeNMDUfDxA1sOL70okyA3npjASXgJPGE1FsbpqzWjsN0TAGoZkR1VWuP9Yn0CrH7dA4lhZQfUU
VjvqzFBZK8N9iZMzIu6KOiSY/aD4O159vbDS4kO0rTG1DYQNniZzMPNliiJ+0EVkfYRwABRFA8fmL+6btqZ
qhjGY29EHuIfzIMTDTsrtCTGxQn2ql1zwjReGiNXzmFncwvyy92DAuMbnOQiE1YNn72wThy2oWSHsCwKdIV
cNHqY2xBvFnkZ9Ltga7PgR33kbJ7G18tjiZF root@jenkins-201
```

3.3 gitlab添加部署公钥

The screenshot shows the GitLab Admin Area interface. On the left, there's a sidebar with various options like Admin Area, Overview, Monitoring, Messages, System Hooks, Applications, Abuse Reports, Kubernetes, Deploy Keys (which is highlighted with a red box), Service Templates, Labels, Appearance, and Settings. The main content area shows 'Public deploy keys (0)' and a 'New deploy key' button (also highlighted with a red box). A large watermark '深圳开放大学' is diagonally across the page.

The screenshot shows the 'New public deploy key' creation page. The 'Title' field has 'jenkins' entered (highlighted with a red box). The 'Key' field contains the copied SSH public key from the previous step. A large watermark '深圳开放大学' is diagonally across the page.

Public deploy keys (1)

[New deploy key](#)

Title	Fingerprint	Projects with write access	Added at
jenkins	0b:66:7a :c2:ea:2f :cf:7c:bd :6f:cf:d0 :7f:a3:4c :aa		added just now

3.4 gitlab项目关联部署公钥

GitLab Projects Groups Activity Milestones Snippets 🔍 Search or jump to... 🌐 ⚙️ New project

Projects

Your projects 3 Starred projects 0 Explore projects Filter by name... Last updated

All Personal

H dev / h5game Maintainer Updated 31 minutes ago

A ops / ansible Maintainer Updated 11 hours ago

G dev / game Maintainer Updated 11 hours ago

Deploy Keys Deploy keys allow read-only or read-write (if enabled) access to your repository. Deploy keys can be used for CI, staging or production servers. You can create a deploy key or add an existing one.

Create a new deploy key for this project

Title

Key

Paste a machine public key here. Read more about how to generate it here

Write access allowed

Allow this key to push to repository as well? (Default only allows pull access.)

Add key

Enabled deploy keys 0 Privately accessible deploy keys 0 Publicly accessible deploy keys 1

Deploy key Project usage Created 激活部署公钥

jenkins 0b:66:7a:c2:ea:2f:cf:7c:bd:6f:cf:d0:7f:a3:4c:aa None 1 minute ago Enable

Enabled deploy keys 1

Privately accessible deploy keys 0

Publicly accessible deploy keys 0

Deploy key

Project usage

Created

jenkis

0b:66:7a:c2:ea:2f:cf:7c:bd:6f:cf:d0:7f:a3:4c:aa

Current project

2 minutes ago



3.5 jenkins配置私钥凭证

General

源码管理

构建触发器

构建环境

构建

构建后操作

源码管理

无

Git

Repositories

Repository URL

git@10.0.0.200:dev/h5game.git

Failed to connect to repository : Command "git ls-remote -h
git@10.0.0.200:dev/h5game.git HEAD" returned status code 128:
stdout:
stderr: Permission denied, please try again.
Permission denied, please try again.
Permission denied (publickey,password).
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.

Credentials

- 无 -

添加

Jenkins

选择添加凭证

高级...

Add Repository

Branches to build

指定分支 (为空时代表any)

*/master

增加分支

Jenkins凭据提供者: Jenkins

添加凭据

Domain 全局凭据 (unrestricted)

类型 SSH Username with private key **选择 SSH 私钥**

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

ID (必填)

描述 (必填)

Username (必填)

Private Key Enter directly
注意! 这里填写的是jenkins的私钥!!!
Enter New Secret Below
-----BEGIN RSA PRIVATE KEY-----
MIUXA0GATADQYJQ/4BD/KG4/D8ECG1ADFT8//Z19143/1G00H33L0VDMY3nEP
X4o223ehAHR2CTDoB4n0Cq0tNFXVTrnday4rByrN5M1H5PSuciBeMkXCwPBLV3w1z
NONWqIg/hYct6kIx2zyIkjlWrgD76pcpnFWLmAjeF4b9hzCKk=
-----END RSA PRIVATE KEY-----

Passphrase (必填)

添加 取消

源码管理

无
Git

Repositories

Repository URL git@10.0.0.200:dev/h5game.git
Credentials jenkins **可以发现，警告消失了**

Branches to build

指定分支 (为空时代表any) */master
增加分支

源码库浏览器 (自动)

Additional Behaviours 新增

3.6 测试获取代码

Jenkins h5game

Project h5game

立即构建

工作区

最新修改记录

相关链接

返回面板 状态 修改记录 工作空间 Build Now 删除 Project 配置 收藏夹 打开 Blue Ocean 重命名

The image shows a screenshot of a Jenkins project page for 'h5game'. At the top left is the Jenkins logo and the word 'Jenkins'. Below it is a breadcrumb navigation: 'Jenkins > h5game'. On the left, there's a sidebar with various icons and labels: '返回面板' (Return Panel), '状态' (Status), '修改记录' (Change History), '工作空间' (Workspace), 'Build Now' (highlighted with a red border), '删除 Project' (Delete Project), '配置' (Configure), '收藏夹' (Favorites), '打开 Blue Ocean' (Open Blue Ocean), and '重命名' (Rename). In the center, the project name 'Project h5game' is displayed above a large red button labeled '立即构建' (Build Now). To the right of the button are three links: '工作区' (Workspace) with a folder icon, '最新修改记录' (Latest Change History) with a notepad icon, and '相关链接' (Related Links) with a link icon. A large watermark reading '深工实践 教育' is diagonally across the page.



Build History

[构建历史](#)

find X

#1

2020-8-6 上午9:37

变更记录

控制台输出

编辑编译信息

删除构建 '#1'

Git Build Data

No Tags

打开 Blue Ocean

S 全部 RSS 失败



控制台输出

```
Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/h5game
using credential b8c1f793-47ed-4903-995d-2273673d8f87
Cloning the remote Git repository
Cloning repository git@10.0.0.200:dev/h5game.git
> git init /var/lib/jenkins/workspace/h5game # timeout=10
Fetching upstream changes from git@10.0.0.200:dev/h5game.git
> git --version # timeout=10
using GIT_SSH to set credentials
> git fetch --tags --progress git@10.0.0.200:dev/h5game.git +refs/heads/*:refs/remotes/origin/*
> git config remote.origin.url git@10.0.0.200:dev/h5game.git # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url git@10.0.0.200:dev/h5game.git # timeout=10
Fetching upstream changes from git@10.0.0.200:dev/h5game.git
using GIT_SSH to set credentials
> git fetch --tags --progress git@10.0.0.200:dev/h5game.git +refs/heads/*:refs/remotes/origin/*
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision bebbb4fale2d69baf9bfe6c4a322af3abd0a5e84 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f bebbb4fale2d69baf9bfe6c4a322af3abd0a5e84
Commit message: "del"
First time build. Skipping changelog.
Finished: SUCCESS
```

拉取代码成功

查看拉取的代码：

```
1 [root@jenkins-201 ~]# ll /var/lib/jenkins/workspace/h5game
2 总用量 16
3 drwxr-xr-x 4 jenkins jenkins 47 8月 6 09:37 game
4 -rw-r--r-- 1 jenkins jenkins 9349 8月 6 09:37 LICENSE
5 -rw-r--r-- 1 jenkins jenkins 937 8月 6 09:37 README.md
```

4. 编写部署脚本

```
1 #创建目录
2 mkdir -p /scripts/jenkins/
3
4 #编写脚本
5 cat > /scripts/jenkins/deploy.sh << 'EOF'
6 #!/bin/bash
7
8 PATH_CODE=/var/lib/jenkins/workspace/h5game/
9 PATH_WEB=/usr/share/nginx
10 TIME=$(date +%Y%m%d-%H%M)
11 IP=10.0.0.7
12
13 #打包代码
14 cd ${PATH_CODE}
```

```
15 tar zcf /opt/${TIME}-web.tar.gz ./*
16
17 #拷贝打包好的代码发送到web服务器代码目录
18 ssh ${IP} "mkdir ${PATH_WEB}/${TIME}-web -p"
19 scp /opt/${TIME}-web.tar.gz ${IP}: ${PATH_WEB}/${TIME}-web
20
21 #web服务器解压代码
22 ssh ${IP} "cd ${PATH_WEB}/${TIME}-web && tar xf ${TIME}-web.tar.gz && rm -rf
${TIME}-web.tar.gz"
23 ssh ${IP} "cd ${PATH_WEB} && rm -rf html && ln -s ${TIME}-web html"
24 EOF
25
26 #添加可执行权限
27 chmod +x /scripts/jenkins/deploy.sh
```

也可以使用jenkins内置的变量来代替自定义变量，查看jenkins内置变量的地址如下：

```
1 | http://10.0.0.201:8080/env-vars.html
```

4.jenkins调用构建脚本

在构建的位置填写执行shell脚本的命令

General 源码管理 构建触发器 构建环境 **构建** 构建后操作

- Delete workspace before build starts
- Use secret text(s) or file(s) 
- Abort the build if it's stuck
- Add timestamps to the Console Output
- Generate Release Notes
- Inspect build log for published Gradle build scans
- With Ant 

构建

Execute shell  

命令 `bash /scripts/jenkins/deploy.sh`

[查看 可用的环境变量列表](#) 

增加构建步骤 ▾

构建后操作

增加构建后操作步骤 ▾

然后立即构建，发现报错了，提示权限不足：



控制台输出

```
Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/h5game
using credential b8c1f793-47ed-4903-995d-2273673d8f87
> git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url git@10.0.0.200:dev/h5game.git # timeout=10
Fetching upstream changes from git@10.0.0.200:dev/h5game.git
> git --version # timeout=10
using GIT_SSH to set credentials
> git fetch --tags --progress git@10.0.0.200:dev/h5game.git +refs/heads/*:refs/remotes/origin/*
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision bebbb4fa1e2d69baf9bfe6c4a322af3abd0a5e84 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f bebbb4fa1e2d69baf9bfe6c4a322af3abd0a5e84
Commit message: "del"
> git rev-list --no-walk bebbb4fa1e2d69baf9bfe6c4a322af3abd0a5e84 # timeout=10
[h5game] $ /bin/sh -xe /tmp/jenkins4676819158569642509.sh
+ bash /scripts/jenkins/deploy.sh
tar (child): /opt/20200806-0957-web.tar.gz: 无法 open: 权限不够
tar (child): Error is not recoverable. exiting now
Host key verification failed.
Host key verification failed.
lost connection
Host key verification failed.
Host key verification failed.
Build step 'Execute shell' marked build as failure
Finished: FAILURE
```

报错原因是因为jenkins是以jenkins用户运行的，所以提示权限不足，我们可以修改jenkins为root用户运行。

```
1 [root@jenkins-201 ~]# vim /etc/sysconfig/jenkins
2 [root@jenkins-201 ~]# grep "USER" /etc/sysconfig/jenkins
3 JENKINS_USER="root"
4 [root@jenkins-201 ~]# systemctl restart jenkins
```

重启好之后我们重新构建一下：



控制台输出

```
Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/h5game
using credential b8c1f793-47ed-4903-995d-2273673d8f87
> git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url git@10.0.0.200:dev/h5game.git # timeout=10
Fetching upstream changes from git@10.0.0.200:dev/h5game.git
> git --version # timeout=10
using GIT_SSH to set credentials
> git fetch --tags --progress git@10.0.0.200:dev/h5game.git +refs/heads/*:refs/remotes/origin/*
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision bebbb4fa1e2d69baf9bfe6c4a322af3abd0a5e84 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f bebbb4fa1e2d69baf9bfe6c4a322af3abd0a5e84
Commit message: "del"
> git rev-list --no-walk bebbb4fa1e2d69baf9bfe6c4a322af3abd0a5e84 # timeout=10
[h5game] $ /bin/sh -xe /tmp/jenkins8504823544345295828.sh
+ bash /scripts/jenkins/deploy.sh
Finished: SUCCESS
```

查看一下web服务器的代码目录

```
1 [root@web-7 ~]# ll /usr/share/nginx/
2 总用量 0
3 drwxr-xr-x 3 root root 50 8月 6 10:13 20200806-1013-web
4 lrwxrwxrwx 1 root root 17 8月 6 10:13 html -> 20200806-1013-web
```

第4章 监听gitlab自动触发构建

1.jenkins项目里添加构建触发器

General 源码管理 构建触发器 构建环境 构建 构建后操作

Build after other projects are built ?

Build periodically ?

Build when a change is pushed to GitLab. GitLab webhook URL: http://10.0.0.201:8080/project/h5game ?

Enabled GitLab triggers

Push Events	<input checked="" type="checkbox"/>
Opened Merge Request Events	<input checked="" type="checkbox"/>
Accepted Merge Request Events	<input type="checkbox"/>
Closed Merge Request Events	<input type="checkbox"/>
Rebuild open Merge Requests	Never
Approved Merge Requests (EE-only)	<input checked="" type="checkbox"/>
Comments	<input checked="" type="checkbox"/>
Comment (regex) for triggering a build	Jenkins please retry a ?

Enable [ci-skip] ?

Ignore WIP Merge Requests ?

Set build description to build cause (eg. Merge request or Git Push) ?

Build on successful pipeline events ?

Pending build name for pipeline ?

Cancel pending merge request builds on update ?

Allowed branches

只监听 master 分支

Allow all branches to trigger this job ?

Filter branches by name ?

Include master Following patterns don't match any branch in source repository: master

Exclude ?

Filter branches by regex ?

Filter merge request by label ?

Secret token 生成 token 0cbdc1686f5073ac314c17a35afe682c Generate Clear

保存 应用

2.gitlab添加webhook

将刚才jenkins里配置的token和URL地址复制进去

h h5game

Project
Repository
Issues 0
Merge Requests 0
CI / CD
Operations
Wiki
Snippets

Settings

General
Members
Integrations

Repository
CI / CD
Operations

Integrations

Webhooks can be used for binding events when something is happening within the project.

URL
http://10.0.0.201:8080/project/h5game jenkins 的项目地址

Secret Token
0cbdc1686f5073ac314c17a35afe682c jenkins 生成的 token

Use this token to validate received payloads. It will be sent with the request in the X-Gitlab-Token HTTP header.

Trigger

Push events
This URL will be triggered by a push to the repository
Branch name or wildcard pattern to trigger on (leave blank for all)

Tag push events
This URL will be triggered when a new tag is pushed to the repository

Comments
This URL will be triggered when someone adds a comment

Confidential Comments
This URL will be triggered when someone adds a comment on a confidential issue

Issues events
This URL will be triggered when an issue is created/updated/merged

Confidential Issues events
This URL will be triggered when a confidential issue is created/updated/merged

较新版本的gitlab此时点击添加会提示报错：

dev > h5game > Integrations Settings

Url is blocked: Requests to the local network are not allowed

Integrations

Webhooks can be used for binding events when something is happening within the project.

URL
http://example.com/trigger-ci.json

Secret Token

Use this token to validate received payloads. It will be sent with the request in the X-Gitlab-Token HTTP header.

解决方法：进入admin area区域，然后点击setting-->network进行设置

GitLab Admin Area > Network

Performance optimization

Various settings that affect GitLab performance.

User and IP Rate Limits

Configure limits for web and API requests.

Outbound requests

Allow requests to the local network from hooks and services.

Allow requests to the local network from hooks and services

Enforce DNS rebinding attack protection

Resolves IP addresses once and uses them to submit requests

Save changes

Settings

- General
- Integrations
- Repository
- CI/CD
- Reporting
- Metrics and profiling
- Network**
- Preferences

正常添加成功之后，会在下方出现测试的选项

Webhooks (1)

<http://10.0.0.201:8080/project/h5game>

Push Events

SSL Verification: enabled

Test ▾

Push events

Service	Description
Asana	Asana - Teamwork without borders

选择push事件来测试是否可以正常出发，如果可以，会在页面上方显示200状态码

Hook executed successfully: HTTP 200

此时我们去查看jenkins项目页面，就会发现以及出发了来自gitlab的构建任务

The screenshot shows the Jenkins interface for the 'h5game' project. On the left, there's a sidebar with various icons and links: '返回面板', '状态', '修改记录', '工作空间', 'Build Now', '删除 Project', '配置', '收藏夹', '打开 Blue Ocean', and '重命名'. The main area is titled 'Project h5game'. It features a large watermark-like graphic with Chinese characters '深圳开放大学' (Shenzhen Open University) diagonally across it. Below the watermark, there are two sections: '工作区' (Workspace) with a folder icon and '最新修改记录' (Latest modification history) with a notepad icon. To the right of these is a '相关链接' (Related Links) section with a list of recent builds. At the bottom, there's a 'Build History' panel with a red border around the most recent entry, which is highlighted with a red box and the text '成功触发了构建' (Successful trigger). The entry details are: '#8 2020-8-6 上午11:04 Started by GitLab push by cto'.

第5章 返回构建状态给gitlab

1.gitlab生成access token

User Settings > Access Tokens

Personal Access Tokens

You can generate a personal access token for each application you use that needs access to the GitLab API.

You can also use personal access tokens to authenticate against Git over HTTP. They are the only accepted password when you have Two-Factor Authentication (2FA) enabled.

Add a personal access token

Pick a name for the application, and we'll give you a unique personal access token.

Name: jenkins

Expires at: YYYY-MM-DD

Scopes:

api

Grants complete read/write access to the API, including all groups and projects.

read_user

Grants read-only access to the authenticated user's profile through the /user API endpoint, which includes username, public email, and full name. Also grants access to read-only API endpoints under /users.

read_repository

Grants read-only access to repositories on private projects using Git-over-HTTP or the Repository Files API.

write_repository

Grants read-write access to repositories on private projects using Git-over-HTTP (not using the API).

sudo

Grants permission to perform API actions as any user in the system, when authenticated as an admin user.

Create personal access token

点击创建之后会生成一串token,注意及时保存,因为刷新就没有了

User Settings > Access Tokens

Your new personal access token has been created.

Personal Access Tokens

You can generate a personal access token for each application you use that needs access to the GitLab API.

Your New Personal Access Token

8LjxStoEHJfuXztsi8si

注意保存, 刷新就没了

Make sure you save it - you won't be able to access it again.

2.jenkins配置gitlab的token

点击jenkins的系统管理-->系统设置, 然后找到gitlab选项

Gitlab

Enable authentication for '/project' end-point

GitLab connections

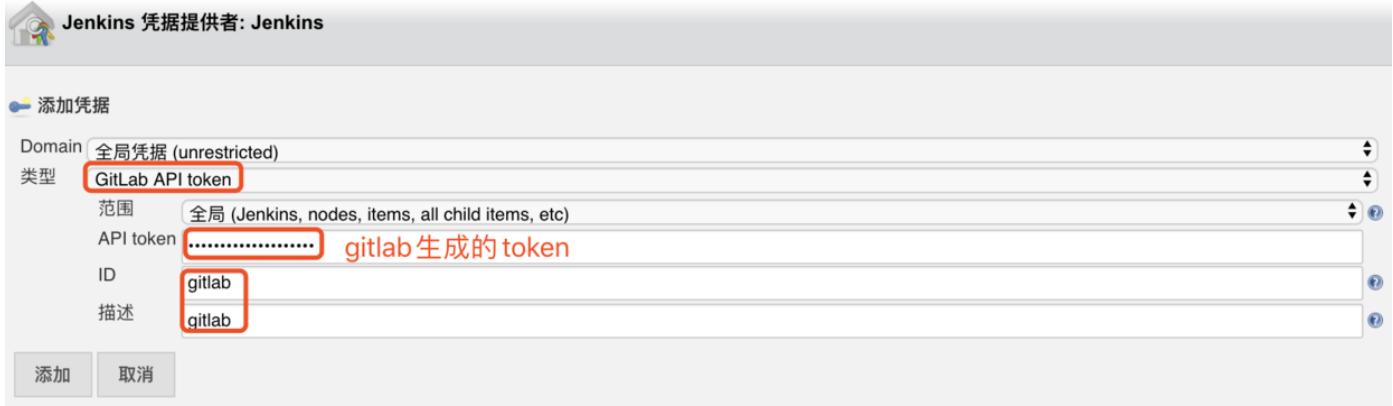
Connection name: gitlab

Gitlab host URL: http://10.0.0.200/

Credentials: 无

API Token for Gitlab access required

填写gitlab的信息：



点击添加后返回上一层页面，然后选中刚才添加的gitlab凭证

Gitlab

Enable authentication for '/project' end-point

GitLab connections

Connection name gitlab

Gitlab host URL http://10.0.0.200/

Credentials The complete URL to the GitLab server (e.g. http://gitlab.mydomain.com)
GitLab API token (gitlab)

API Token for accessing Gitlab

高级... Test Connection 删除

3. 设置项目构建后将结果通知给gitlab

构建后操作

Publish build status to GitLab

增加构建后操作步骤

保存 应用

4. 合并分支然后检查gitlab能否收到消息

5. 防止重复构建

Jenkins具有很多内置变量，点击项目-->构建-->查看可用的环境变量列表

```
1 | http://10.0.0.201:8080/env-vars.html/
```

这里我们使用两个变量，一个是此次提交的commit的hash，另一个是上一次提交成功的commit的hash

我们可以在部署脚本里添加一行判断，如果这两个变量一样，那么就不用再次提交了

这些变量不需要在脚本里定义，直接引用即可

分配给构建作为工作空间的目录的绝对路径。

JENKINS_HOME
在主节点上为Jenkins存储数据分配的目录的绝对路径。

JENKINS_URL
Jenkins的完整URL，例如http://server:port/jenkins/（注意：仅在系统配置中设置了Jenkins URL时可用）

BUILD_URL
此版本的完整URL，例如http://server:port/jenkins/job/foo/15/（必须设置Jenkins URL）

JOB_URL
作业的完整URL，例如http://server:port/jenkins/job/foo/（必须设置Jenkins URL）

GIT_COMMIT
提交哈希被检出。

GIT_PREVIOUS_COMMIT
提交的哈希值最后建立在此分支上（如果有）。

GIT_PREVIOUS_SUCCESSFUL_COMMIT
最后一次成功在该分支上构建提交的哈希（如果有）。

GIT_BRANCH
远程分支名称（如果有）。

GIT_LOCAL_BRANCH
被检出的本地分支名称（如果适用）。

第6章 tag方式发布版本

1.给代码打标签

首先我们先给代码打上标签，然后提交2个版本

v1.0版本：修改代码，然后发布v1.0版本

```
1 git commit -am 'v1.0'  
2 git tag -a v1.0 -m "v1.0 稳定版"  
3 git push -u origin v1.0  
4 git tag
```

v2.0版本：修改代码，然后发布v2.0版本

```
1 git commit -am 'v2.0'  
2 git tag -a v2.0 -m "v2.0 稳定版"  
3 git push -u origin v2.0  
4 git tag
```

2.gitlab查看标签

此时gitlab上可以看到2个标签

点进去之后可以看到具体标签名称

Tags give the ability to mark specific points in history as being important

Last updated

New tag



• v2.0 v2.0 stable
- 2d7506f2 · v2.0 · 1 minute ago



• v1.0 v1.0 stable
- 76492fa2 · v1.0 · 2 minutes ago



3.jenkins配置参数化构建

jenkins上我们新建一个参数化构建项目

输入一个任务名称

» 必填项

 Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

然后配置git的标签参数:

General 源码管理 构建触发器 构建环境 构建 构建后操作

Discard old builds

策略 Log Rotation

保持构建的天数 5
如果非空, 构建记录将保存此天数

保持构建的最大个数 5
如果非空, 最多此数目的构建记录将被保存

GitHub 项目

GitLab Connection gitlab

GitLab Repository Name

This build requires lockable resources

This project is parameterized

Git Parameter

Name git_version
Description 选择你要发布的版本号
[Plain text] 预览

Parameter Type Tag
Default Value v1.0

添加参数 高级...

Throttle builds

关闭构建

在必要的时候并发构建

高级... 保存 应用

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

General 源码管理 构建触发器 构建环境 构建 构建后操作

源码管理

无 Git

Repositories

Repository URL: git@10.0.0.200:dev/h5game.git

Credentials: - 无 - 添加

高级... Add Repository

Branches to build

Branch Specifier (blank for 'any'): \$git_version

X Add Branch

源码库浏览器: (自动)

Additional Behaviours: 新增

Mercurial Subversion

此时点击项目的build with parameters就会看到对应的版本号：

Jenkins > my-deploy-rollback

返回面板 状态 修改记录 工作空间 Build with Parameters 删除 Project 配置 收藏夹 打开 Blue Ocean 重命名

Project my-deploy-rollback

需要如下参数用于构建项目：

git_version	v2.0 v1.0
-------------	--------------

选择你要发布的版本号

开始构建

Build History 构建历史

find RSS 全部 RSS 失败

然后去jenkins工作目录下查看是否拉取了对应版本:

```
1 | /var/lib/jenkins/workspace/my-deploy-rollback
```

4.优化部署脚本

```
1 cat >/scripts/jenkins/deploy_rollback.sh<<'EOF'
2 #!/bin/bash
3
4 PATH_CODE=/var/lib/jenkins/workspace/my-deploy-rollback/
5 PATH_WEB=/usr/share/nginx
6 IP=10.0.0.7
7
8 #打包代码
9 cd ${PATH_CODE}
10 tar zcf /opt/web-${git_version}.tar.gz ./*
11
12 #拷贝打包好的代码发送到web服务器代码目录
13 ssh ${IP} "mkdir ${PATH_WEB}/web-${git_version} -p"
14 scp /opt/web-${git_version}.tar.gz ${IP}:${PATH_WEB}/web-${git_version}
15
16 #web服务器解压代码
17 ssh ${IP} "cd ${PATH_WEB}/web-${git_version} && tar xf web-${git_version}.tar.gz &&
18 rm -rf web-${git_version}.tar.gz"
19 ssh ${IP} "cd ${PATH_WEB} && rm -rf html && ln -s web-${git_version} html"
20 EOF
```

5.jenkins添加执行脚本动作并测试

General 源码管理 构建触发器 构建环境 构建 构建后操作

- GitHub hook trigger for GITScm polling
- Gitlab Merge Requests Builder
- Poll SCM

构建环境

- Delete workspace before build starts
- Use secret text(s) or file(s)
- Provide Configuration files
- Abort the build if it's stuck
- Add Changelog Information to Environment
- Add timestamps to the Console Output
- Generate Release Notes
- Inspect build log for published Gradle build scans
- With Ant

构建

Execute shell

命令 `bash -x /scripts/jenkins/deploy_rollback.sh`

查看 可用的环境变量列表 高级...

增加构建步骤 ▾

构建后操作

增加构建后操作步骤 ▾

保存 应用



6. 测试发布

- 返回面板
- 状态
- 修改记录
- 工作空间
- Build with Parameters **选中并高亮**
- 删除 Project
- 配置
- 收藏夹
- 打开 Blue Ocean
- 重命名

Project my-deploy-rollback

需要如下参数用于构建项目:

git_version **v2.0**

v1.0

选择你要发布的版本号

开始构建

Build History 构建历史 —

find

#2 2020-8-6 下午3:59
#1 2020-8-6 下午3:41

RSS 全部 RSS 失败

然后去web服务器上查看发现已经发布了

```
1 [root@web-7 ~]# ll /usr/share/nginx/
2 总用量 0
3 lrwxrwxrwx 1 root root 8 8月 6 15:59 html -> web-v2.0
4 drwxr-xr-x 3 root root 68 8月 6 15:59 web-v2.0
```

第7章 tag方式回滚版本

1.jenkins配置回滚选项参数

在工程配置里添加选项参数:

General 源码管理 构建触发器 构建环境 构建 构建后操作

Git Parameter

Name: git_version
Description: 选择你要发布的版本号
[Plain text] 预览
Parameter Type: Tag
Default Value: v1.0
高级...

添加参数 ▾

Boolean Parameter
Choice Parameter
File Parameter
Git Parameter
高级...

General 源码管理 构建触发器 构建环境 构建 构建后操作

This project is parameterized

Git Parameter

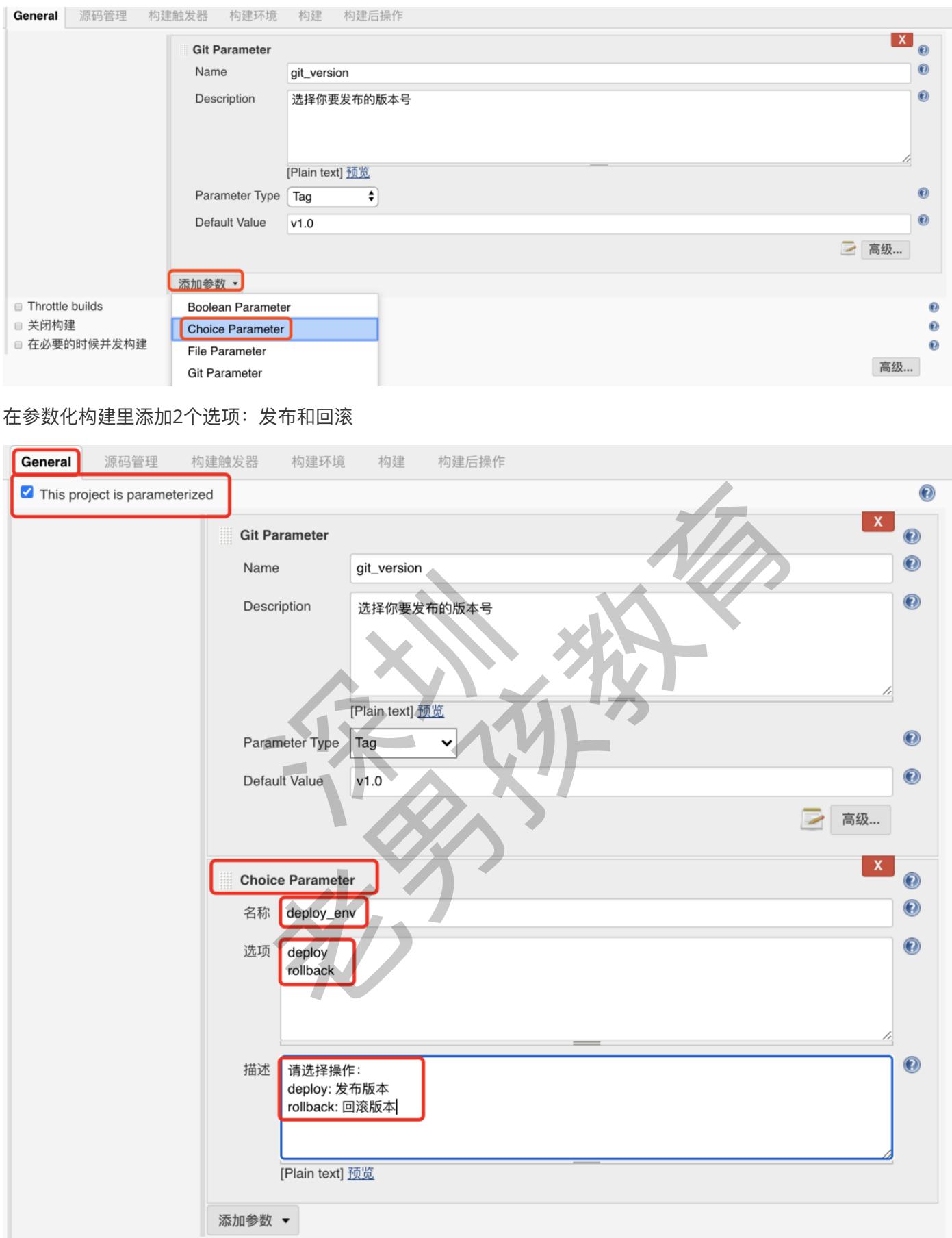
Name: git_version
Description: 选择你要发布的版本号
[Plain text] 预览
Parameter Type: Tag
Default Value: v1.0
高级...

Choice Parameter

名称: deploy_env
选项: deploy
 rollback

描述: 请选择操作:
deploy: 发布版本
rollback: 回滚版本
[Plain text] 预览

添加参数 ▾



此时查看构建页面就会发现多了选项卡:

返回面板

状态

修改记录

工作空间

Build with Parameters

删除 Project

配置

收藏夹

打开 Blue Ocean

重命名

Project my-deploy-rollback

需要如下参数用于构建项目:

git_version
v2.0
v1.0

选择你要发布的版本号

deploy_env deploy

请选择操作:
deploy: 发布版本
rollback: 回滚版本

开始构建

2.修改发布脚本增加条件判断

```
1 cat >/scripts/jenkins/deploy_rollback.sh << EOF
2 #!/bin/bash
3
4 PATH_CODE=/var/lib/jenkins/workspace/my-deploy-rollback/
5 PATH_WEB=/usr/share/nginx
6 IP=10.0.0.7
7
8 #打包代码
9 code_tar(){
10     cd ${PATH_CODE}
11     tar zcf /opt/web-${git_version}.tar.gz ./*
12 }
13
14 #拷贝打包好的代码发送到web服务器代码目录
15 code_scp(){
16     ssh ${IP} "mkdir ${PATH_WEB}/web-${git_version} -p"
17     scp /opt/web-${git_version}.tar.gz ${IP}:${PATH_WEB}/web-${git_version}
18 }
19
20 #web服务器解压代码
21 code_xf(){
22     ssh ${IP} "cd ${PATH_WEB}/web-${git_version} && tar xf web-
23 ${git_version}.tar.gz && rm -rf web-${git_version}.tar.gz"
24 }
25
26 #创建代码软链接
27 code_ln(){
28     ssh ${IP} "cd ${PATH_WEB} && rm -rf html && ln -s web-${git_version} html"
29 }
```

```
30 main(){
31     code_tar
32     code_scp
33     code_xf
34     code_ln
35 }
36
37 #选择发布还是回滚
38 if [ "${deploy_env}" == "deploy" ]
39 then
40     ssh ${IP} "ls ${PATH_WEB}/web-${git_version}" >/dev/null 2>&1
41     if [ $? == 0 -a ${GIT_COMMIT} == ${GIT_PREVIOUS_SUCCESSFUL_COMMIT} ]
42     then
43         echo "web-${git_version} 已部署,不允许重复构建"
44         exit
45     else
46         main
47     fi
48 elif [ "${deploy_env}" == "rollback" ]
49 then
50     code_ln
51 fi
52 EOF
```

3. 测试回滚功能

部署v1.0版本

Jenkins > my-deploy-rollback >

- [返回面板](#)
- [状态](#)
- [修改记录](#)
- [工作空间](#)
- [Build with Parameters](#)
- [删除 Project](#)
- [配置](#)
- [收藏夹](#)
- [打开 Blue Ocean](#)
- [重命名](#)

Project my-deploy-rollback

需要如下参数用于构建项目:

git_version

选择你要发布的版本号

deploy_env

请选择操作:
deploy: 发布版本
rollback: 回滚版本

开始构建

部署v2.0版本:

- [返回面板](#)
- [状态](#)
- [修改记录](#)
- [工作空间](#)
- [Build with Parameters](#)
- [删除 Project](#)
- [配置](#)
- [收藏夹](#)
- [打开 Blue Ocean](#)
- [重命名](#)

Project my-deploy-rollback

需要如下参数用于构建项目:

git_version

选择你要发布的版本号

deploy_env

请选择操作:
deploy: 发布版本
rollback: 回滚版本

开始构建

检查web服务器当前的版本

```

1 [root@web-7 ~]# ll /usr/share/nginx/
2 总用量 0
3 lwxrwxrwx 1 root root 8 8月 6 16:52 html -> web-v2.0
4 drwxr-xr-x 3 root root 68 8月 6 16:51 web-v1.0
5 drwxr-xr-x 3 root root 68 8月 6 16:52 web-v2.0

```

然后我们选择v1.0版本并且选择回滚操作：

Jenkins

Project my-deploy-rollback

需要如下参数用于构建项目:

git_version v2.0
v1.0

deploy_env rollback

请选择操作:
deploy: 发布版本
rollback: 回滚版本

开始构建

查看控制台显示回滚成功：

控制台输出

```

Started by user admin
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/my-deploy-rollback
No credentials specified
> git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url git@10.0.0.200:dev/h5game.git # timeout=10
Fetching upstream changes from git@10.0.0.200:dev/h5game.git
> git --version # timeout=10
> git fetch --tags --progress git@10.0.0.200:dev/h5game.git +refs/heads/*:refs/remotes/origin/*
> git rev-parse origin/v1.0^{commit} # timeout=10
> git rev-parse v1.0^{commit} # timeout=10
Checking out Revision 76492fa2f77c38cf5a7adecbb86400a059b915a41 (v1.0)
> git config core.sparsecheckout # timeout=10
> git checkout -f 76492fa2f77c38cf5a7adecbb86400a059b915a41
Commit message: "v1.0"
> git rev-list --no-walk 76492fa2f77c38cf5a7adecbb86400a059b915a41 # timeout=10
[my-deploy-rollback] $ /bin/sh -xe /tmp/jenkins915050735975099823.sh
+ bash -x /scripts/jenkins/deploy_rollback.sh
+ PATH_CODE=/var/lib/jenkins/workspace/my-deploy-rollback/
+ PATH_WEB=/usr/share/nginx
+ IP=10.0.0.7
+ '[' rollback == deploy ']'
+ '[' rollback == rollback ']'
+ code_ln
+ ssh 10.0.0.7 'cd /usr/share/nginx && rm -rf html && ln -s web-v1.0 html'
Finished: SUCCESS

```

在web服务器上查看发现已经回滚成功：

```
1 [root@web-7 ~]# ll /usr/share/nginx/
2 总用量 0
3 lrwxrwxrwx 1 root root 8 8月 6 16:56 html -> web-v1.0
4 drwxr-xr-x 3 root root 68 8月 6 16:51 web-v1.0
5 drwxr-xr-x 3 root root 68 8月 6 16:52 web-v2.0
```

4.发布新代码并打标签测试

修改代码并发布v3.0:

```
1 cd h5game/
2 echo v3.0 >> index.html
3 git commit -am 'v3.0'
4 git tag -a v3.0 -m 'v3.0 稳定版'
5 git push -u origin v3.0
6 git tag
```

jenkins查看并发布3.0版本:

web服务器查看发布情况:

```
1 [root@web-7 ~]# ll /usr/share/nginx/
2 总用量 0
3 lrwxrwxrwx 1 root root 8 8月 6 16:58 html -> web-v3.0
4 drwxr-xr-x 3 root root 68 8月 6 16:51 web-v1.0
5 drwxr-xr-x 3 root root 68 8月 6 16:52 web-v2.0
6 drwxr-xr-x 3 root root 68 8月 6 16:58 web-v3.0
```

然后工程选择回滚到v2.0版本:

再次在web服务器上查看：

```
1 [root@web-7 ~]# ll /usr/share/nginx/
2 总用量 0
3 lwxrwxrwx 1 root root 8 8月 6 16:59 html -> web-v2.0
4 drwxr-xr-x 3 root root 68 8月 6 16:51 web-v1.0
5 drwxr-xr-x 3 root root 68 8月 6 16:52 web-v2.0
6 drwxr-xr-x 3 root root 68 8月 6 16:58 web-v3.0
```

第8章 打标签工程

1.jenkins新建打标签工程

2.配置参数化构建

The screenshot shows the Jenkins project configuration page for a new job. The 'General' tab is selected. A red box highlights the checkbox 'This project is parameterized'. Below it, under 'Git Parameter', there is a configuration for 'git_version': Name is 'git_version', Description is '当前项目中的历史版本:', Type is 'Tag', and Default Value is 'v1.0'. Under 'String Parameter', there is a configuration for 'git_newversion': Name is 'git_newversion', Description is '请输入要创建的标签名:', and a note '字符串参数' is overlaid in red. At the bottom, there is a '添加参数' (Add Parameter) button.

3.配置工程的源码管理

General 源码管理 构建触发器 构建环境 构建 构建后操作

源码管理

无
Git

Repositories

Repository URL: git@10.0.0.200:dev/h5game.git

Credentials: - 无 - 添加

高级...
Add Repository

Branches to build

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

X
Add Branch

4.配置构建动作

配置命令：

```
1 | git tag -a $git_newversion -m "$git_newversion 稳定版"
2 | git push -u origin $git_newversion
```

5.发布新版本并提交到master

```
1 | echo "v4.0" >> index.html
2 | git add .
3 | git commit -m "v4.0 稳定版"
4 | git push -u origin v4.0
```

6.jenkins使用打标签工程来打标签

查看控制台输出发现提示没有权限：

我们需要在gitlab上把deploy key设置允许写入

然后重新构建

```
1 |
```

返回到deploy-rollback工程查看可以发现已经生成了新标签，然后我们开始构建：

在web服务器上查看可以发现新版本已经发布成功了：

```
1 [root@web ~]# ll /usr/share/nginx/
2 总用量 20
3 lrwxrwxrwx 1 root root    8 5月 13 16:09 html -> web-v4.0
4 drwxr-xr-x 8 root root 4096 5月 13 16:05 web-
5 drwxr-xr-x 8 root root 4096 5月 13 15:00 web-v1.0
6 drwxr-xr-x 8 root root 4096 5月 13 15:00 web-v2.0
7 drwxr-xr-x 8 root root 4096 5月 13 15:37 web-v3.0
8 drwxr-xr-x 8 root root 4096 5月 13 16:09 web-v4.0
```

