CICD持续集成持续部署系统

实验名称：搭建CICD（continuous integration continuous deploy）生态系统

软件环境：docker+gitlab+Jenkins

实验环境：rhel7.5四台，一台gitlab服务器（4G），一台Jenkins服务器，一台客户端

一台web服务器

#软件准备

[root@room8pc205 soft]# ls docker.tar jenkins-2.121-1.1.noarch.rpm gitlab\_zh.tar

docker.tar gitlab\_zh.tar jenkins-2.121-1.1.noarch.rpm

[root@room8pc205 ~]# virsh list --all

Id 名称 状态

----------------------------------------------------

1 node01 running 192.168.4.1 192.168.32.135 gitlab.tedu.cn

2 node02 running 192.168.4.2 192.168.32.123 jenkins.tedu.cn

3 node03 running 192.168.4.3 client.tedu.cn

4 node04 running 192.168.4.4. 192.168.32.138 web.tedu.cn

#gitlab部署

[root@room8pc205 ~]# cd /data/soft/

[root@room8pc205 soft]# scp docker.tar gitlab\_zh.tar 192.168.4.1:/root

docker.tar 100% 19MB 32.1MB/s 00:00

gitlab\_zh.tar 100% 1648MB 31.7MB/s 00:52

#安装docker，导入gitlab镜像

[root@room8pc205 soft]# ssh 192.168.4.1

Last login: Tue Mar 5 21:04:10 2019

[root@gitlab ~]# ls

docker.tar gitlab\_zh.tar

[root@gitlab ~]# tar -xf docker.tar

[root@gitlab ~]# yum -y localinstall docker-engine-\*.rpm

[root@gitlab ~]# systemctl start docker

[root@gitlab ~]# docker load < gitlab\_zh.tar

a94e0d5a7c40: Loading layer [==================================================>] 116.5 MB/116.5 MB

88888b9b1b5b: Loading layer [==================================================>] 15.87 kB/15.87 kB

52f389ea437e: Loading layer [==================================================>] 14.85 kB/14.85 kB

52a7ea2bb533: Loading layer [==================================================>] 5.632 kB/5.632 kB

db584c622b50: Loading layer [==================================================>] 3.072 kB/3.072 kB

62786ff6a243: Loading layer [==================================================>] 75.85 MB/75.85 MB

71bc04f4b7c7: Loading layer [==================================================>] 2.048 kB/2.048 kB

26e083d332d8: Loading layer [==================================================>] 2.048 kB/2.048 kB

2c02e58e96b8: Loading layer [==================================================>] 2.048 kB/2.048 kB

589c7a23de2a: Loading layer [==================================================>] 15.87 kB/15.87 kB

44474d2cdcd1: Loading layer [==================================================>] 1.359 GB/1.359 GB

41c94e16b901: Loading layer [==================================================>] 16.78 MB/16.78 MB

04cafa6a1534: Loading layer [==================================================>] 160 MB/160 MB

Loaded image: gitlab\_zh:latest ] 553.7 kB/160 MB

[root@gitlab ~]# docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

gitlab\_zh latest 1f71f185271a 11 months ago 1.627 GB

[root@gitlab ~]# rm -rf gitlab\_zh.tar

#修改物理机ssh端口为2222，方便gitlab容器映射22端口

[root@gitlab ~]# vim /etc/ssh/sshd\_config

[root@gitlab ~]# sed -n '17p' /etc/ssh/sshd\_config

Port 2222

[root@gitlab ~]# systemctl restart sshd

[root@gitlab ~]# exit

登出

Connection to 192.168.4.1 closed.

[root@room8pc205 soft]# ssh -p 2222 192.168.4.1

Last login: Tue Mar 5 21:07:55 2019 from 192.168.4.254

#启动gitlab容器

[root@gitlab ~]# docker run -itd -h gitlab --name gitlab -p 80:80 -p 443:443 -p 22:22 --restart always -v /srv/gitlab/config:/etc/gitlab -v /srv/gitlab/logs:/var/log/gitlab -v /srv/gitlab/data gitlab\_zh

d1eaab42bd6cd921041f501d9824dd17ef1281fa58f73154f4674965a1807129

[root@gitlab ~]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

d1eaab42bd6c gitlab\_zh "/assets/wrapper" 4 seconds ago Up 3 seconds (health: starting) 0.0.0.0:22->22/tcp, 0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp gitlab

[root@gitlab ~]# ifconfig eth0 | head -2

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

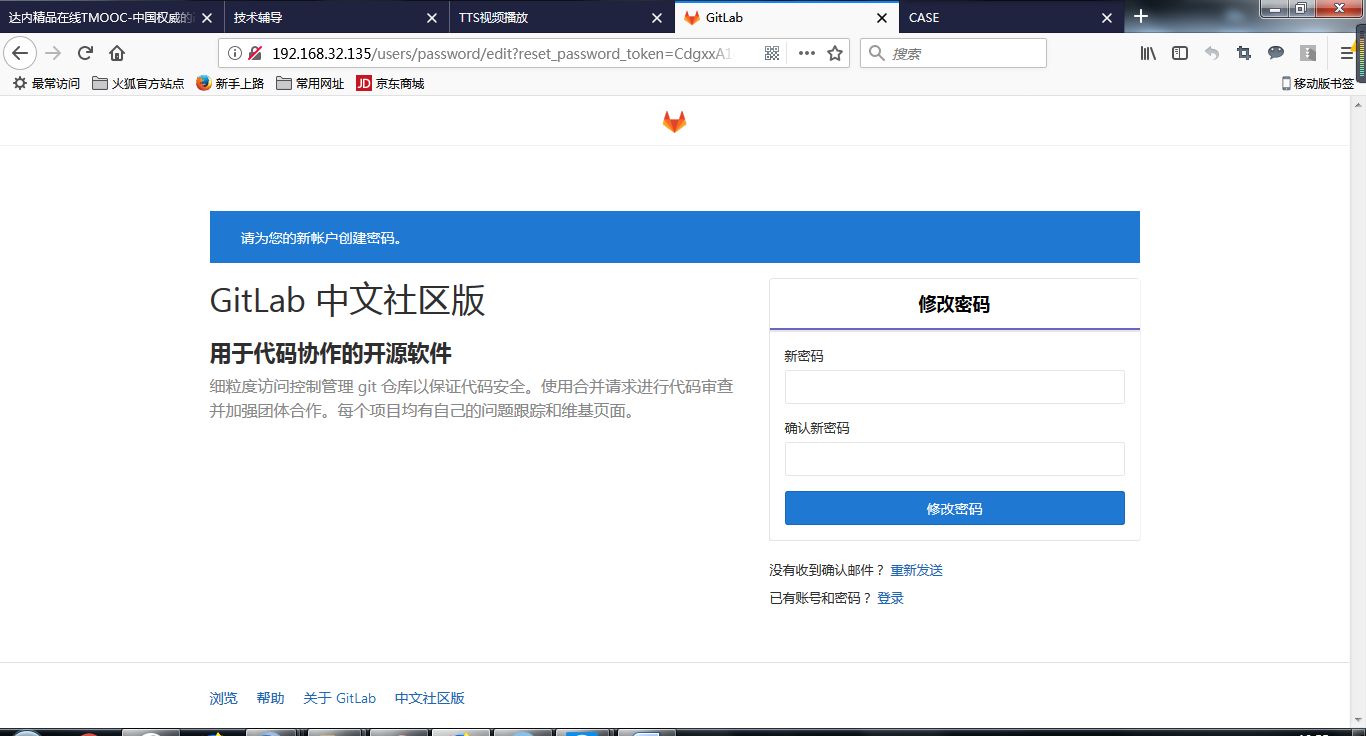
inet 192.168.4.1 netmask 255.255.255.0 broadcast 192.168.4.255

[root@gitlab ~]# ifconfig ens11 | head -2

ens11: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet 192.168.32.135 netmask 255.255.255.0 broadcast 192.168.32.255

#我这里由于环境原因，访问gitlab网页用192.168.32.135这个ip地址



#见到登录页面

#初始化密码，不小于8位，我这里用户名root，密码1234.com



#登录页面



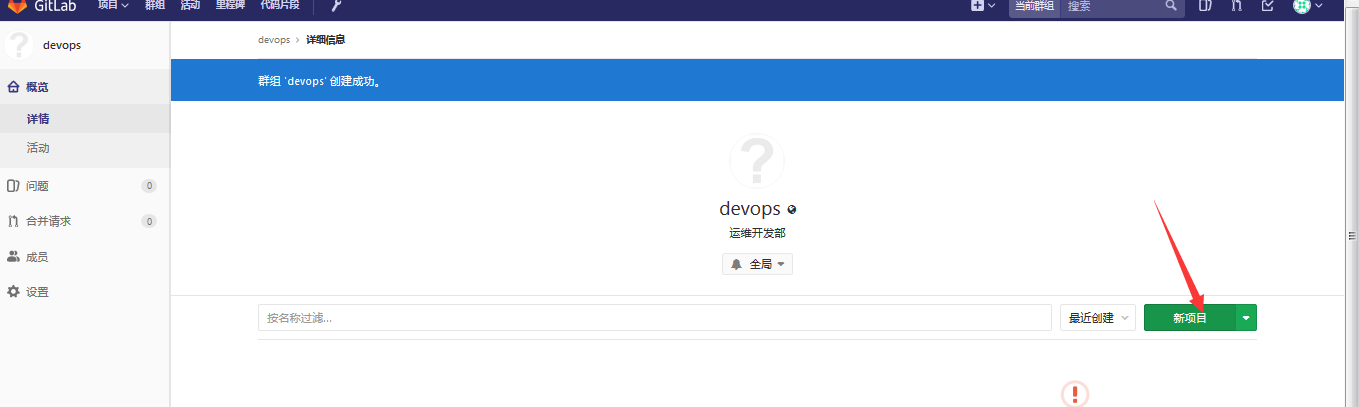
#创建群组



#由于网页登录问题，我这边显示的ip不是192.168.4.1，而是192.168.32.135

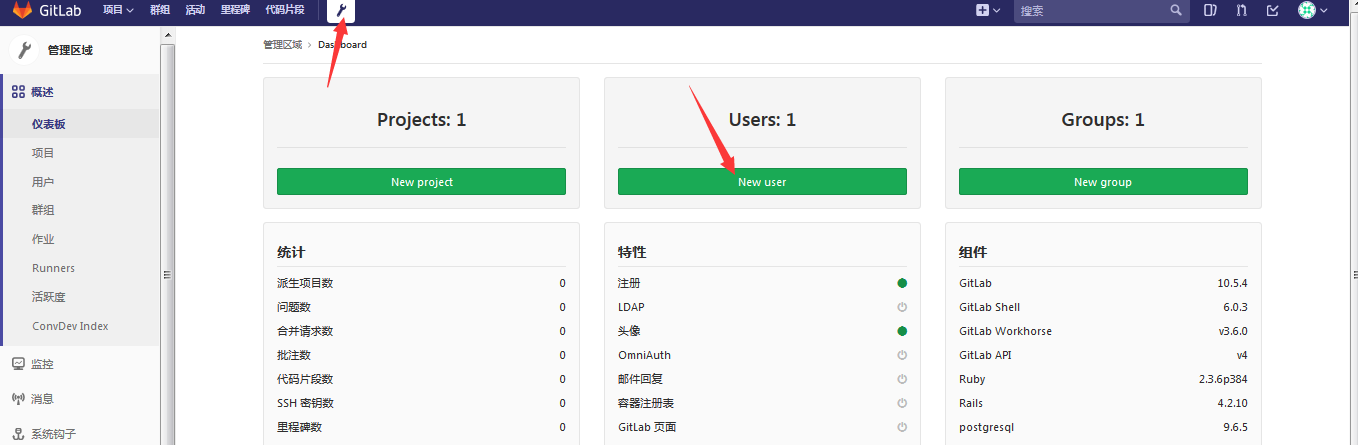


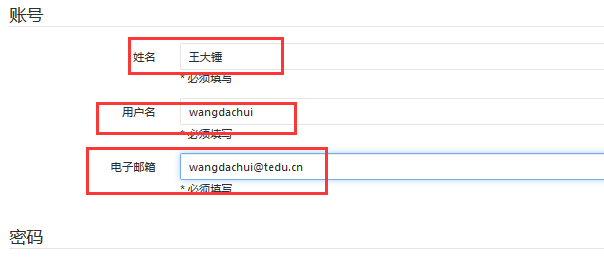
#创建项目





#添加用户





#点击创建用户，首次创建用户不能设置用户密码



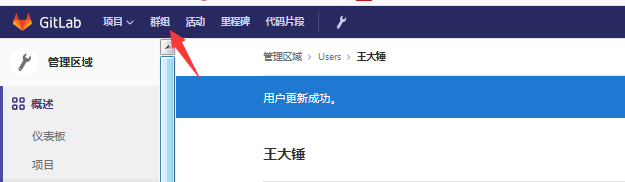
#点击编辑，设置用户密码



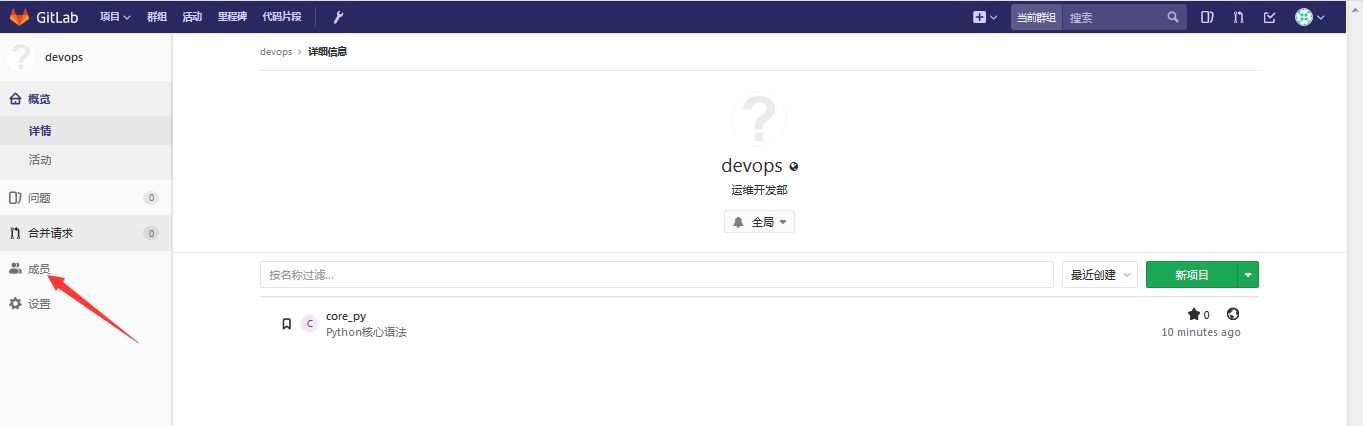
#设置王大锤用户的密码是1234.com，保存设置

#可以退出root用户，换成wangdachui用户测试登录情况

#用户授权，点击导航栏群组

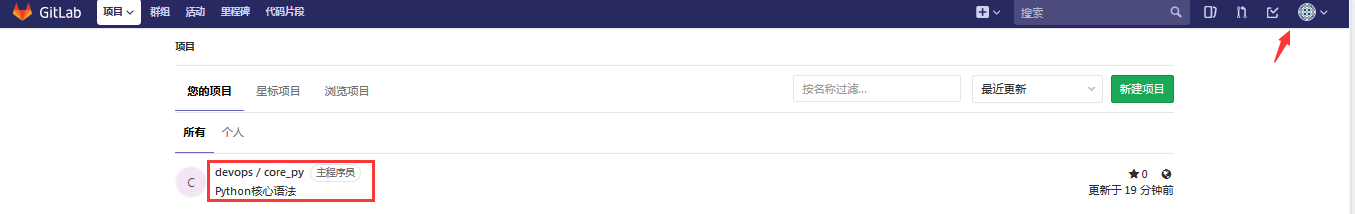








#客户端连接gitlab服务器，退出root用户，切换到wangdachui用户



#点击个人信息中的设置





#生成客户端ssh秘钥对

[root@client ~]# ssh-keygen -t rsa -C "wangdachui@tedu.cn" -b 4096

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /root/.ssh/id\_rsa.

Your public key has been saved in /root/.ssh/id\_rsa.pub.

The key fingerprint is:

SHA256:pql9krmF8Qk3REh6Pa+gBPGXsCD6aUnJaH5qrZen7FU wangdachui@tedu.cn

The key's randomart image is:

+---[RSA 4096]----+

|. o ..... |

|.+ = +.+ |

|o.= + + + |

|oo + o . o |

| .=.. +ES . |

| .+. ..@ + |

| o .o.+o= |

|. oo.++.. |

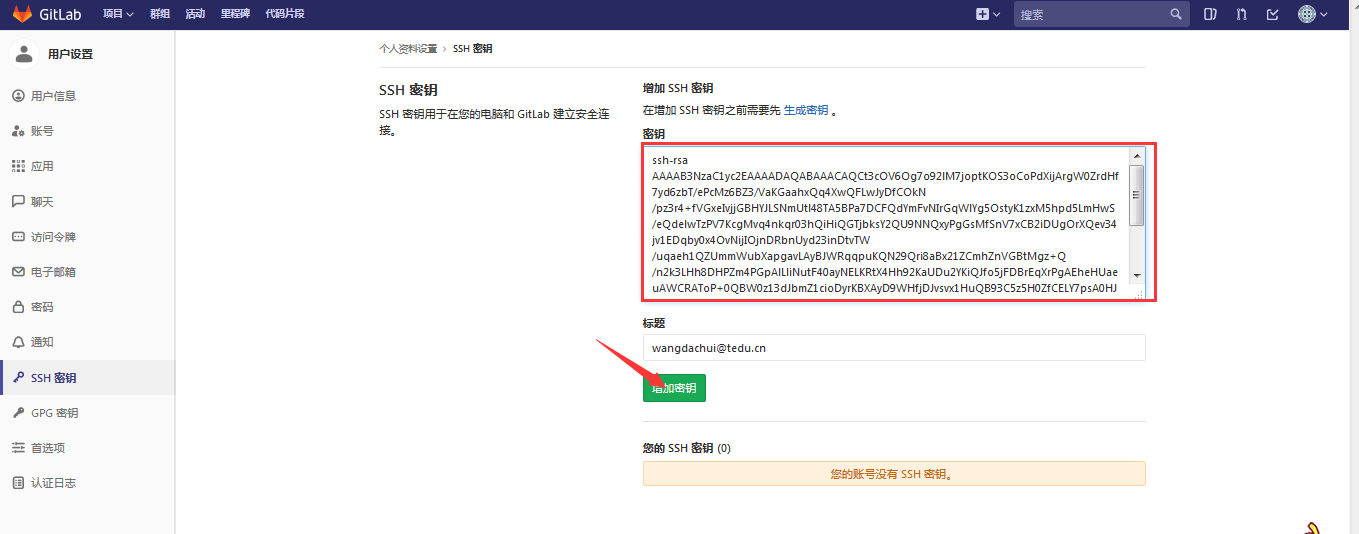
| .o++ o+ |

+----[SHA256]-----+

[root@client ~]# cat /root/.ssh/id\_rsa.pub

ssh-rsa  wangdachui@tedu.cn

#复制生成出来的公钥到gitlab页面上的框里



#客户端安装git

[root@client ~]# yum -y install git

[root@client ~]# mkdir devops

[root@client ~]# cd devops/

[root@client devops]# git init

初始化空的 Git 版本库于 /root/devops/.git/

[root@client devops]# vim README.md

[root@client devops]# cat README.md

This is my first gitlab test file

[root@client devops]# ls

README.md

[root@client devops]# git status -s

?? README.md

[root@client devops]# git status

# 位于分支 master

#

# 初始提交

#

# 未跟踪的文件:

# （使用 "git add <file>..." 以包含要提交的内容）

#

# README.md

提交为空，但是存在尚未跟踪的文件（使用 "git add" 建立跟踪）

[root@client devops]# git add .

[root@client devops]# git status -s

A README.md

[root@client devops]# git status

# 位于分支 master

#

# 初始提交

#

# 要提交的变更：

# （使用 "git rm --cached <file>..." 撤出暂存区）

#

# 新文件： README.md

#

[root@client devops]# git remote add origin [git@192.168.4.1:devops/core\_py.git](mailto:git@192.168.4.1:devops/core_py.git)

[root@client devops]# git commit -m "Initial commit"

[master（根提交） fa6ae21] Initial commit

Committer: root <root@client.tedu.cn>

您的姓名和邮件地址基于登录名和主机名进行了自动设置。请检查它们正确

与否。您可以通过下面的命令对其进行明确地设置以免再出现本提示信息：

git config --global user.name "Your Name"

git config --global user.email you@example.com

设置完毕后，您可以用下面的命令来修正本次提交所使用的用户身份：

git commit --amend --reset-author

1 file changed, 1 insertion(+)

create mode 100644 README.md

[root@client devops]# git push -u origin master

Counting objects: 3, done.

Writing objects: 100% (3/3), 218 bytes | 0 bytes/s, done.

Total 3 (delta 0), reused 0 (delta 0)

To git@192.168.4.1:devops/core\_py.git

\* [new branch] master -> master

分支 master 设置为跟踪来自 origin 的远程分支 master。

[root@client devops]# git push -u origin --all

分支 master 设置为跟踪来自 origin 的远程分支 master。

Everything up-to-date

[root@client devops]# git push -u origin --tags

Everything up-to-date

#观看页面上



#README.md成功上传

#测试上传

[root@client devops]# cp /etc/passwd .

[root@client devops]# git add .

[root@client devops]# git commit -m "a new file"

[master 4d99454] a new file

Committer: root <root@client.tedu.cn>

您的姓名和邮件地址基于登录名和主机名进行了自动设置。请检查它们正确

与否。您可以通过下面的命令对其进行明确地设置以免再出现本提示信息：

git config --global user.name "Your Name"

git config --global user.email you@example.com

设置完毕后，您可以用下面的命令来修正本次提交所使用的用户身份：

git commit --amend --reset-author

1 file changed, 19 insertions(+)

create mode 100644 passwd

[root@client devops]# git push

warning: push.default 未设置，它的默认值将会在 Git 2.0 由 'matching'

修改为 'simple'。若要不再显示本信息并在其默认值改变后维持当前使用习惯，

进行如下设置：

git config --global push.default matching

若要不再显示本信息并从现在开始采用新的使用习惯，设置：

git config --global push.default simple

参见 'git help config' 并查找 'push.default' 以获取更多信息。

（'simple' 模式由 Git 1.7.11 版本引入。如果您有时要使用老版本的 Git，

为保持兼容，请用 'current' 代替 'simple' 模式）

Counting objects: 4, done.

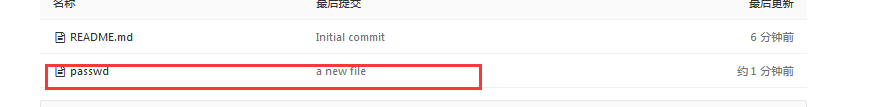
Compressing objects: 100% (3/3), done.

Writing objects: 100% (3/3), 625 bytes | 0 bytes/s, done.

Total 3 (delta 0), reused 0 (delta 0)

To git@192.168.4.1:devops/core\_py.git

fa6ae21..4d99454 master -> master



#测试下载

[root@room8pc205 demo]# ls

[root@room8pc205 demo]# git clone http://192.168.4.1/devops/core\_py.git

正克隆到 'core\_py'...

remote: Counting objects: 6, done.

remote: Compressing objects: 100% (4/4), done.

remote: Total 6 (delta 0), reused 0 (delta 0)

Unpacking objects: 100% (6/6), done.

[root@room8pc205 demo]# ls core\_py/

passwd README.md

##部署Jenkins，此机器需要联网（加设网卡，设置default模式即可）

##使用教室版本git插件总是安装不上，在Jenkins官网下载最新版本的Jenkins包

[root@room8pc205 soft]# wget <https://pkg.jenkins.io/redhat/jenkins-2.167-1.1.noarch.rpm>

[root@room8pc205 soft]# scp jenkins-2.167-1.1.noarch.rpm 192.168.4.2:/root

[root@room8pc205 soft]# ssh 192.168.4.2

[root@jenkins ~]# ls

jenkins-2.167-1.1.noarch.rpm

[root@jenkins ~]# yum -y install java-1.8.0-openjdk-devel

[root@jenkins ~]# yum -y localinstall jenkins-2.167-1.1.noarch.rpm

[root@jenkins ~]# systemctl start jenkins

[root@jenkins ~]# jps

3655 jenkins.war

3705 Jps

[root@jenkins ~]# ss -ntplu | grep java

tcp LISTEN 0 50 :::8080 :::\* users:(("java",pid=3655,fd=160))

[root@jenkins ~]# ifconfig eth0 | head -2

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet 192.168.4.2 netmask 255.255.255.0 broadcast 192.168.4.255

[root@jenkins ~]# ifconfig ens11 | head -2

ens11: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet 192.168.32.123 netmask 255.255.255.0 broadcast 192.168.32.255

#同理，由于环境问题，我这里web页面不采用192.168.4.2，采用192.168.32.123

#注意访问8080端口



[root@jenkins ~]# cat /var/lib/jenkins/secrets/initialAdminPassword

79f554a3bd8f4d7288995535ac2b2500

#将此密码粘贴至web页面

#进入到插件安装页面，自己在家做实验可以选择左边，我这里网速较慢，选择右边，保留git





#点击安装



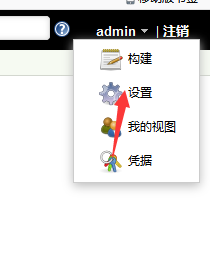
#以admin用户身份继续操作，







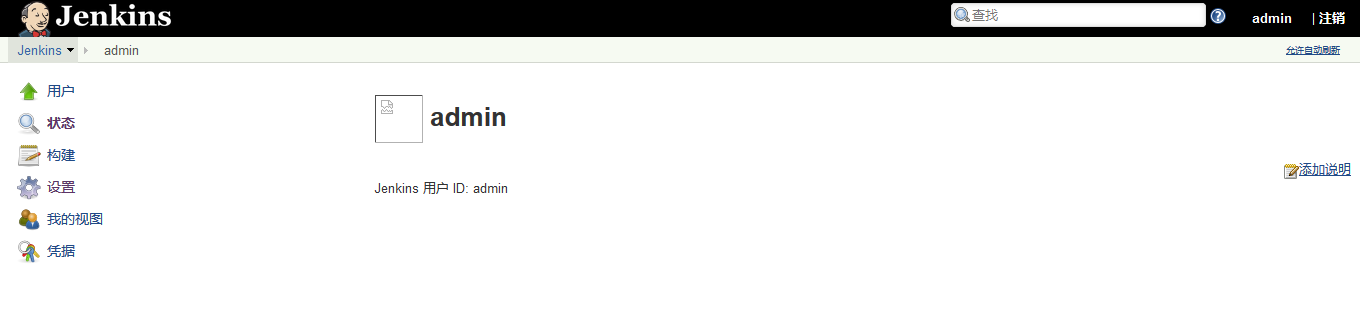
#修改admin用户密码





#修改密码1234.com，保存

#退出重新登录，admin，1234.com



##Jenkins部署完成

#客户端编写程序

[root@client ~]# mkdir myproject

[root@client ~]# cd myproject/

[root@client myproject]# git init

初始化空的 Git 版本库于 /root/myproject/.git/

[root@client myproject]# echo "hello world" > index.html

[root@client myproject]# git add .

[root@client myproject]# git commit -m "myproject v1"

[master（根提交） 5cf88ba] myproject v1

Committer: root <root@client.tedu.cn>

您的姓名和邮件地址基于登录名和主机名进行了自动设置。请检查它们正确

与否。您可以通过下面的命令对其进行明确地设置以免再出现本提示信息：

git config --global user.name "Your Name"

git config --global user.email you@example.com

设置完毕后，您可以用下面的命令来修正本次提交所使用的用户身份：

git commit --amend --reset-author

1 file changed, 1 insertion(+)

create mode 100644 index.html

[root@client myproject]# git config --global user.name "wangdachui"

[root@client myproject]# git config --global user.email [wangdachui@tedu.cn](mailto:wangdachui@tedu.cn)

[root@client myproject]# git status -s

[root@client myproject]# git tag 1.0

[root@client myproject]# echo "2nd version" >> index.html

[root@client myproject]# git add .

[root@client myproject]# git commit -m "myproject v2"

[master bdd9121] myproject v2

1 file changed, 1 insertion(+)

[root@client myproject]# git tag 2.0

[root@client myproject]# git tag

1.0

2.0

#测试版本代码

[root@client myproject]# git checkout 1.0

HEAD 目前位于 5cf88ba... myproject v1

[root@client myproject]# cat index.html

hello world

[root@client myproject]# git checkout master

之前的 HEAD 位置是 5cf88ba... myproject v1

切换到分支 'master'

[root@client myproject]# cat index.html

hello world

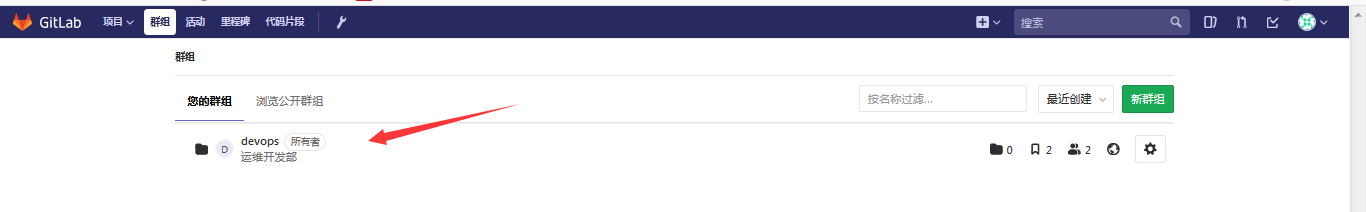
2nd version

##配置gitlab项目

#以root用户登录gitlab，创建项目





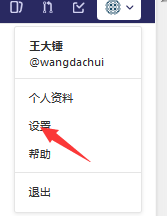


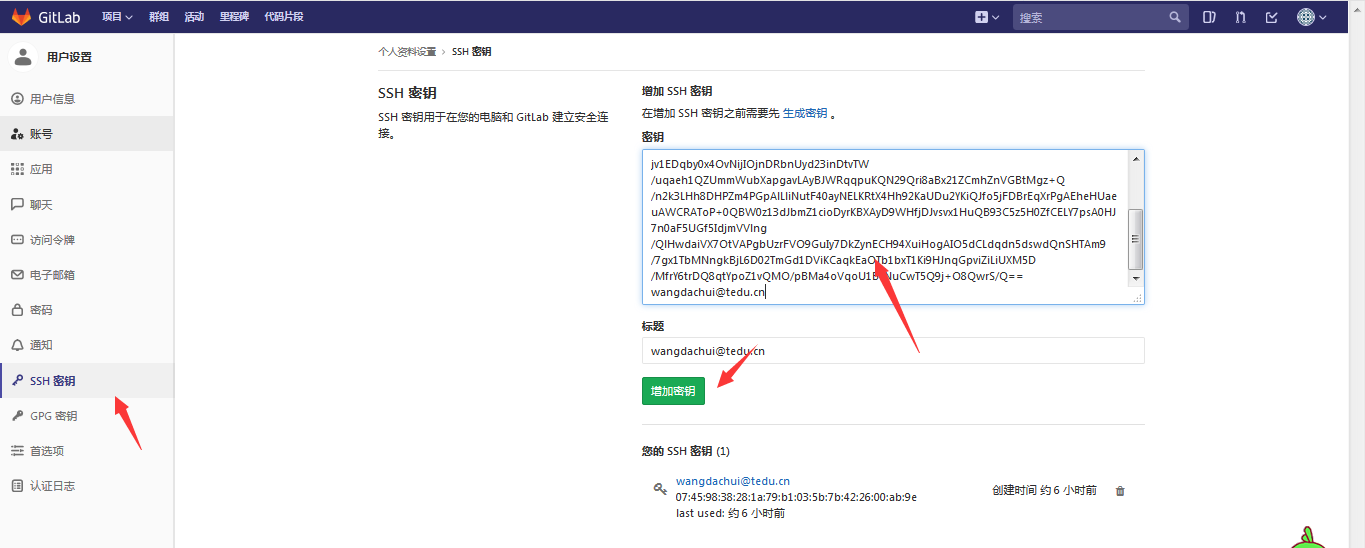
#给用户授权



#以王大锤用户登录gitlab







#客户端推送项目

[root@client myproject]# git remote rename origin old-origin

error: 不能重命名配置小节 'remote.origin' 到 'remote.old-origin'

[root@client myproject]# git remote add origin git@192.168.4.1:devops/myproject.git

[root@client myproject]# git push -u origin --all

Counting objects: 6, done.

Compressing objects: 100% (2/2), done.

Writing objects: 100% (6/6), 452 bytes | 0 bytes/s, done.

Total 6 (delta 0), reused 0 (delta 0)

To git@192.168.4.1:devops/myproject.git

\* [new branch] master -> master

分支 master 设置为跟踪来自 origin 的远程分支 master。

[root@client myproject]# git push -u origin --tags

Total 0 (delta 0), reused 0 (delta 0)

To git@192.168.4.1:devops/myproject.git

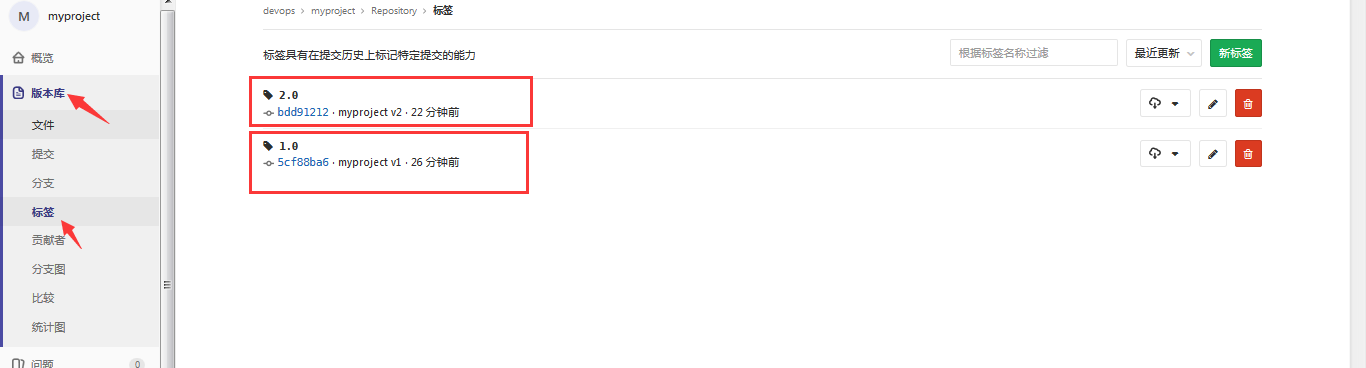
\* [new tag] 1.0 -> 1.0

\* [new tag] 2.0 -> 2.0

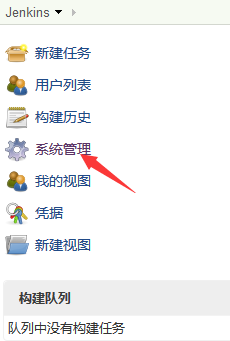
#查看gitlab服务器页面信息

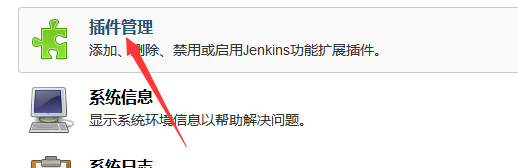


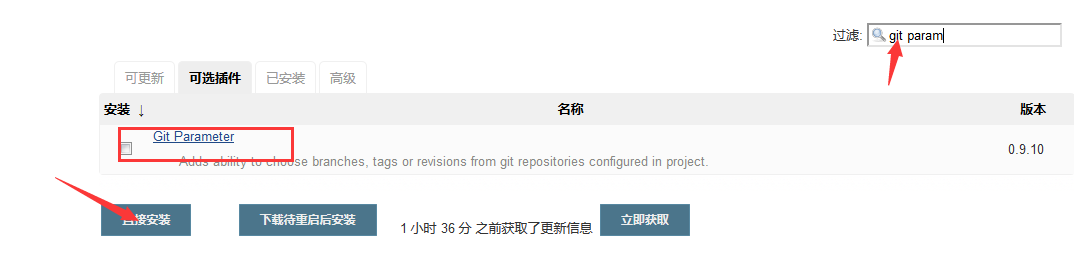
#gitlab服务器上有两个标签的代码版本



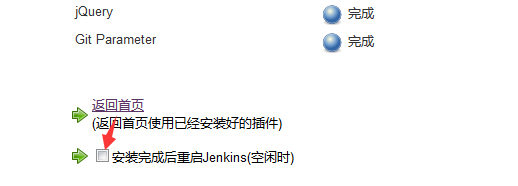
##配置Jenkins关联gitlab



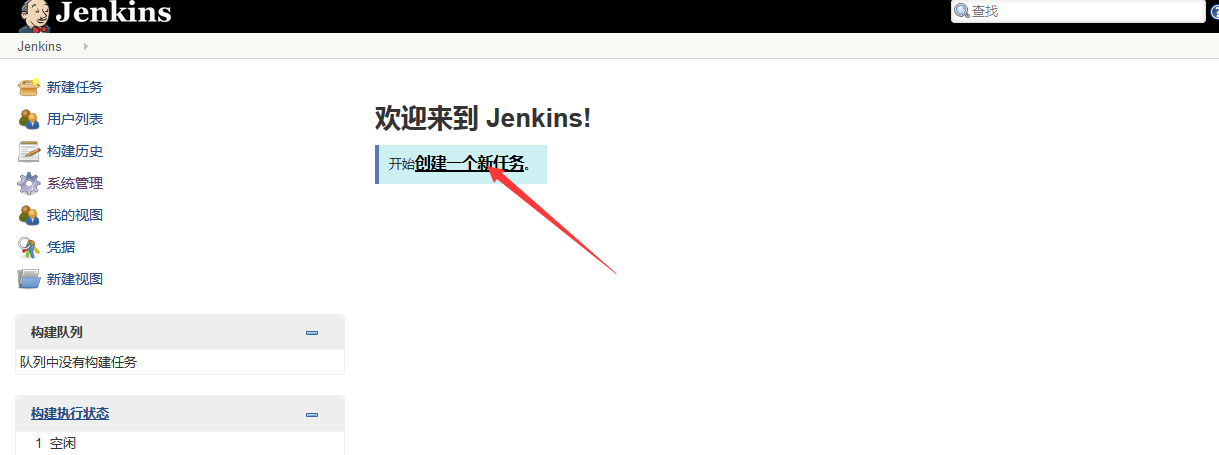




#安装完插件以后重启Jenkins

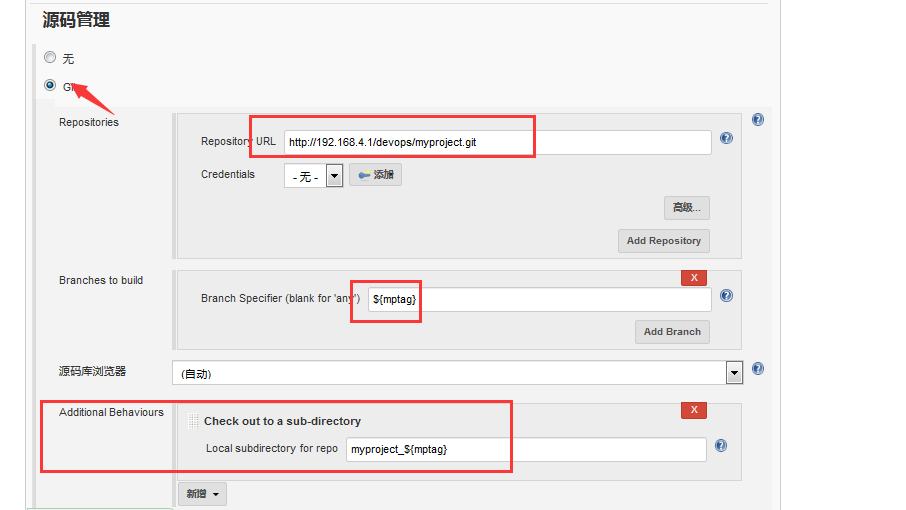


#创建Jenkins任务







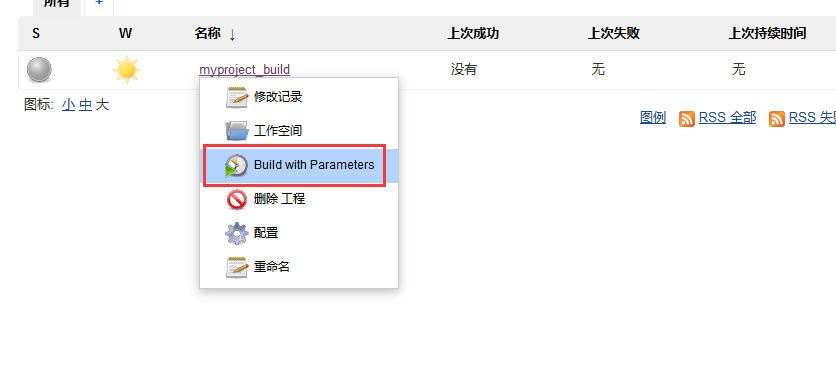


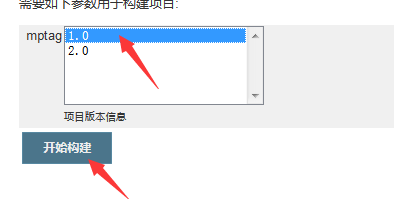
#先应用，再保存

#任务创建成功



#下载代码

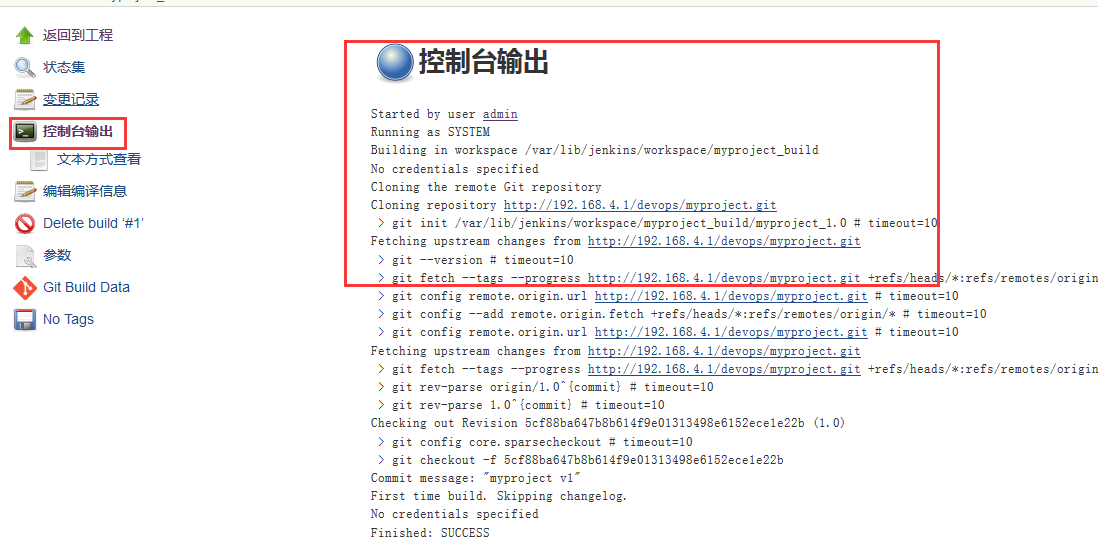




#点击开始构建



#蓝色表示构建成功，红色表示失败，点击球



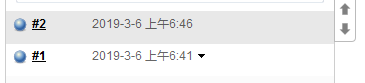
#查看控制台输出

[root@jenkins ~]# ls /var/lib/jenkins/workspace/myproject\_build/myproject\_1.0/

index.html

#在Jenkins本地的工作目录查看已经下载的1版本代码

#同步操作下载2版本代码



[root@jenkins ~]# cd /var/lib/jenkins/workspace/myproject\_build/

[root@jenkins myproject\_build]# tree .

.

├── myproject\_1.0

│?? └── index.html

└── myproject\_2.0

└── index.html

2 directories, 2 files

[root@jenkins myproject\_build]# cat myproject\_1.0/index.html

hello world

[root@jenkins myproject\_build]# cat myproject\_2.0/index.html

hello world

2nd version

#两个版本都下载到Jenkins服务器本地

##优化Jenkins工程构建过程

#包括安装httpd，便于应用服务器下载，从gitlab下载下来文件打tar包并计算md5值，便于应用服务器检查下载文件是否正确

[root@jenkins myproject\_build]# cd

[root@jenkins ~]# yum -y install httpd

[root@jenkins ~]# systemctl start httpd

[root@jenkins ~]# mkdir -p /var/www/deploy/packages

[root@jenkins ~]# chown -R jenkins.jenkins /var/www/deploy/packages/

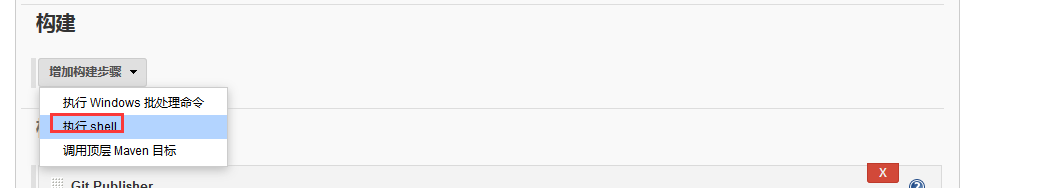
[root@jenkins ~]# ll -d /var/www/deploy/packages/

drwxr-xr-x 2 jenkins jenkins 6 3月 6 06:58 /var/www/deploy/packages/

#修改Jenkins任务配置



#保留原有配置，新增构建步骤





#应用保存

#重新构建工程





#构建成功



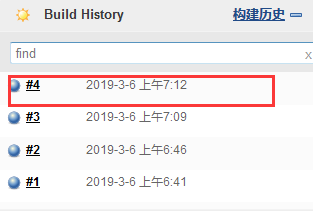


#Jenkins服务器本地查看发布目录

[root@jenkins ~]# ls /var/www/deploy/packages/

myproject\_1.0.tar.gz myproject\_1.0.tar.gz.md5

#同理构建2.0版本



[root@jenkins ~]# cd /var/www/deploy/packages/

[root@jenkins packages]# ls

myproject\_1.0.tar.gz myproject\_2.0.tar.gz

myproject\_1.0.tar.gz.md5 myproject\_2.0.tar.gz.md5

[root@jenkins packages]# cat myproject\_1.0.tar.gz.md5

0e400d097a7283423304a47119b24257

[root@jenkins packages]# cat myproject\_2.0.tar.gz.md5

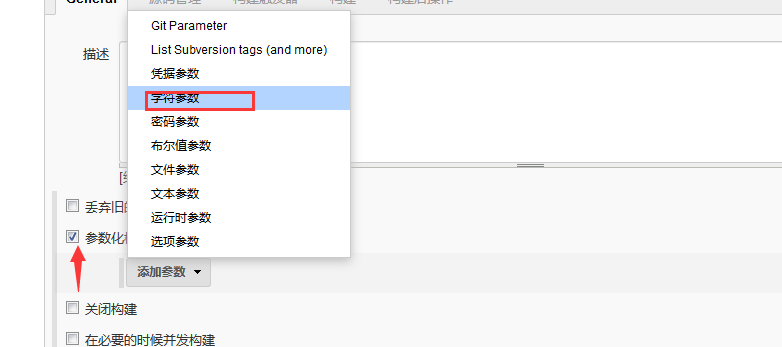
7f5c66bd982cd5d443862a6ca8b07c08

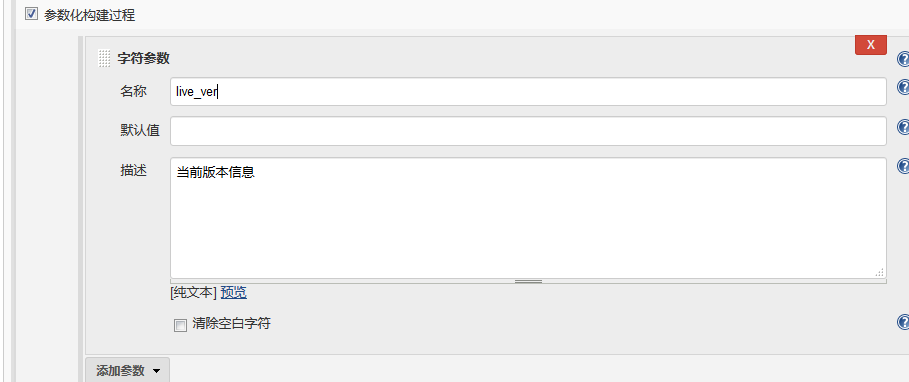
##创建版本信息文件

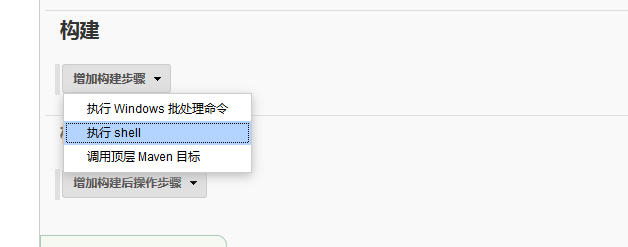
#新建live\_version任务

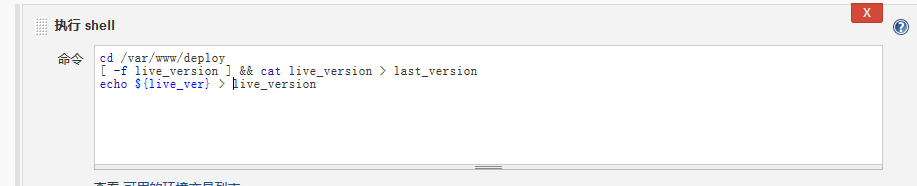












#应用，保存

[root@jenkins ~]# cd /var/www/deploy/

[root@jenkins deploy]# ll -d

drwxr-xr-x 3 root root 22 3月 6 06:58 .

[root@jenkins deploy]# chown -R jenkins.jenkins .

[root@jenkins deploy]# ll -d

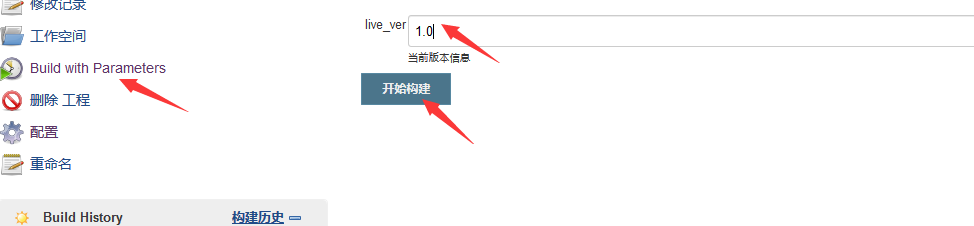
drwxr-xr-x 3 jenkins jenkins 22 3月 6 06:58 .

#构建工程





#测试构建版本文件



[root@jenkins deploy]# ls

live\_version packages

[root@jenkins deploy]# cat live\_version

1.0



[root@jenkins deploy]# ls

last\_version live\_version packages

[root@jenkins deploy]# cat live\_version

2.0

[root@jenkins deploy]# cat last\_version

1.0

#配置Jenkins的httpd服务

[root@jenkins ~]# cd /etc/httpd/conf.d/

[root@jenkins conf.d]# vim jenkins.conf

[root@jenkins conf.d]# cat jenkins.conf

<VirtualHost \*:80>

Servername www.jenkins.com

DocumentRoot "/var/www/deploy"

<Directory "/var/www/deploy">

Options Indexes FollowSymLinks MultiViews

AllowOverride None

Require all granted

</Directory>

</VirtualHost>

[root@jenkins conf.d]# systemctl restart httpd

[root@jenkins conf.d]# curl http://127.0.0.1/last\_version

1.0

[root@jenkins conf.d]# curl http://127.0.0.1/live\_version

2.0

#添加一台服务器作为web服务器

[root@web ~]# yum -y install httpd

[root@web ~]# systemctl start httpd

#编写发布脚本

[root@web ~]# vim deploy.py

[root@web ~]# cat deploy.py

#!/usr/bin/env python3

import requests

from urllib import request

import os

import hashlib

import tarfile

def get\_webdata(url):

r = requests.get(url)

return r.text

def download(pack\_url, pname):

html = request.urlopen(pack\_url)

with open(pname, 'wb') as fobj:

while True:

data = html.read(1024)

if not data:

break

fobj.write(data)

def check\_md5(fname):

m = hashlib.md5()

with open(fname, 'rb') as fobj:

while True:

data = fobj.read(1024)

if not data:

break

m.update(data)

return m.hexdigest()

def deploy(package, local\_path):

os.chdir(local\_path)

tar = tarfile.open(package, 'r:gz')

tar.extractall()

tar.close()

src = package.replace(".tar.gz", "")

dst = "/var/www/html/mysite"

if os.path.exists(dst):

os.unlink(dst)

os.symlink(src, dst)

if \_\_name\_\_ == '\_\_main\_\_':

live\_ver\_url = "http://192.168.4.2/live\_version"

ver = get\_webdata(live\_ver\_url)

ver = ver.rstrip()

app\_name = "myproject\_%s.tar.gz" % (ver)

pack\_url = "http://192.168.4.2/packages/"

local\_path = "/var/www/packages"

if not os.path.exists(local\_path):

os.mkdir(local\_path)

app\_path = os.path.join(local\_path,app\_name)

download(pack\_url+app\_name, app\_path)

local\_md5 = check\_md5(app\_path)

remote\_md5 = get\_webdata(pack\_url + app\_name + '.md5')

remote\_md5 = remote\_md5.rstrip()

if local\_md5 == remote\_md5:

deploy(app\_path, local\_path)

#测试结果

[root@web ~]# ls /var/www/packages

ls: 无法访问/var/www/packages: 没有那个文件或目录

[root@web ~]# ls /var/www/html/

[root@web ~]# python3 deploy.py

[root@web ~]# ls /var/www/packages/

myproject\_2.0 myproject\_2.0.tar.gz

[root@web ~]# ll -d /var/www/html/mysite/

drwxr-xr-x 2 997 995 24 3月 6 07:15 /var/www/html/mysite/

[root@web ~]# ll /var/www/html/mysite/index.html

-rw-r--r-- 1 997 995 24 3月 6 07:15 /var/www/html/mysite/index.html

#我这边web服务器内网地址192.168.4.4，公网地址192.168.32.138



#现在为版本二，测试版本更新



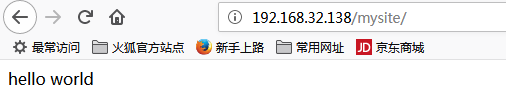




#完成版本更新后重新执行发布脚本

[root@web ~]# python3 deploy.py

#刷新页面



#确认页面显示为版本1代码

#再次更新到二版本

[root@web ~]# curl http://192.168.4.4/mysite/

hello world



[root@web ~]# python3 deploy.py

[root@web ~]# curl http://192.168.4.4/mysite/

hello world

2nd version