

NSD Devops DAY05

1. [案例1：安装Jenkins](#)
2. [案例2：设置本地仓库](#)
3. [案例3：创建远程仓库](#)
4. [案例4：构建工程](#)
5. [案例5：修改工程](#)
6. [案例6：创建版本文件](#)
7. [案例7：发布应用](#)

1 案例1：安装Jenkins

1.1 问题

1. 运行虚拟机，将第一块网络的连接方式改为NAT
2. 安装Jenkins
3. 初始化Jenkins

1.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：运行虚拟机，将第一块网络的连接方式改为NAT，连接互联网

配置虚拟机可以连接互联网

1)打开虚拟机node3的设置，将第一块网卡eth0连接方式改为NAT，如图-1所示：

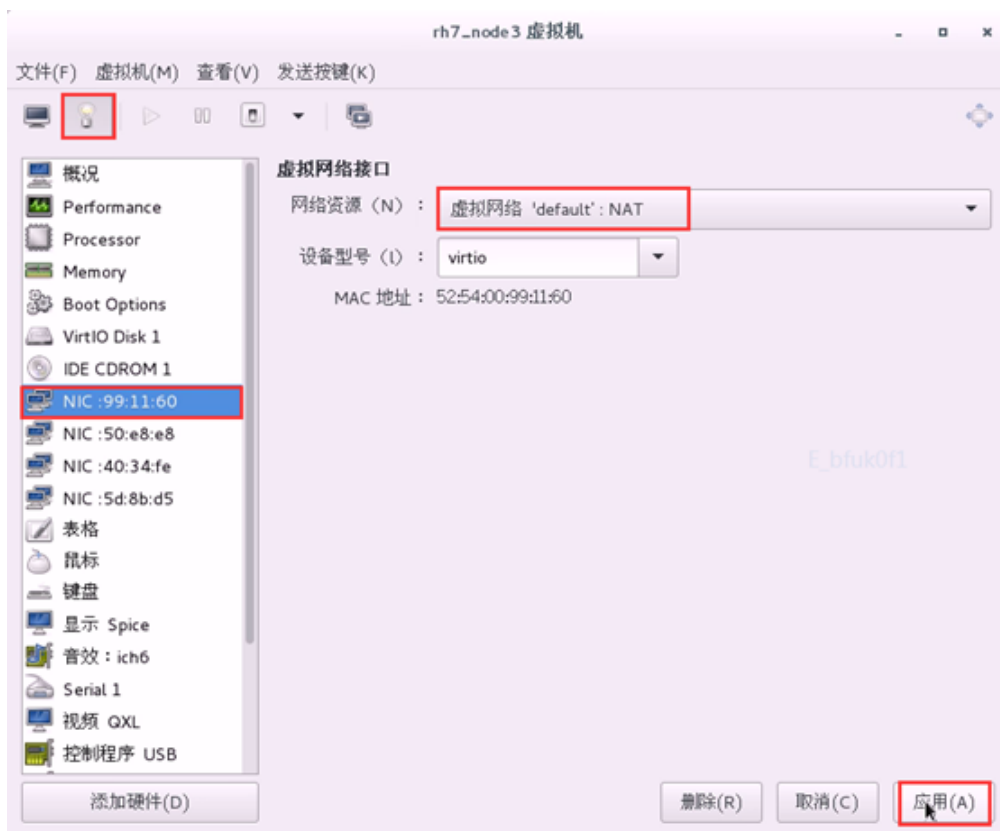


图-1

改为NAT后，物理机使用virbr0和虚拟机的eth0通信

2)进入虚拟机node3，将虚拟机的virbr0关闭

[Top](#)

01. [root@node3 ~] # ifconfig virbr0 down

3)把虚拟机的IP地址设置为自动获得，如图-2所示：

01. [root@node3 ~] # nmtui

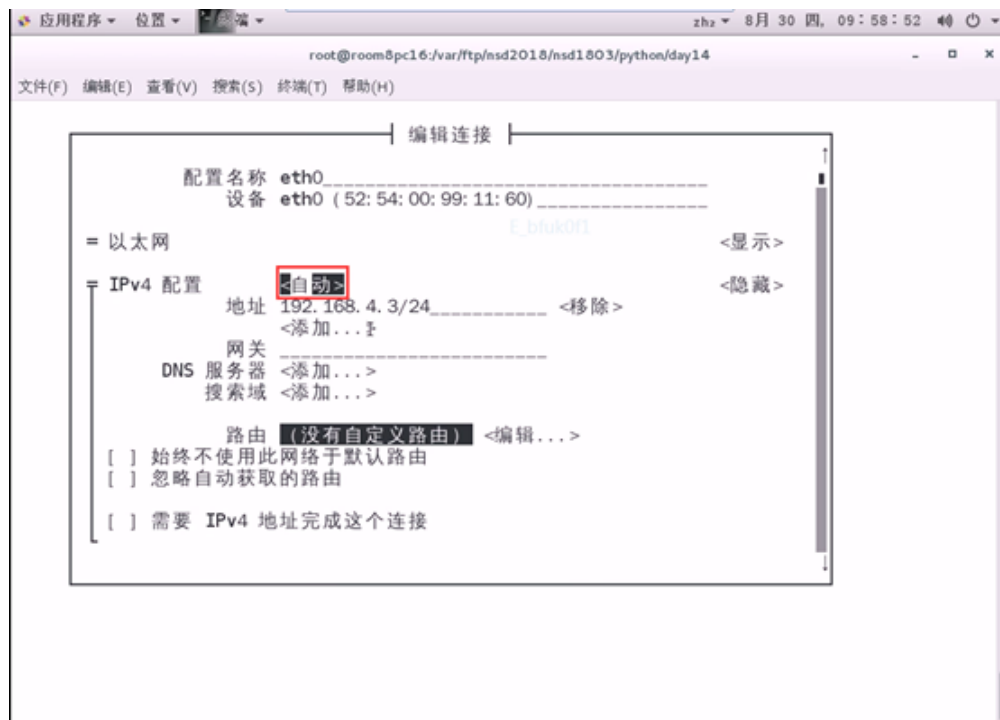


图-2

4)禁用再激活eth0

01. [root@node3 ~] # if down eth0; if up eth0

5)如果虚拟机经过设置还不能上网，把它关机再开机

步骤二：安装Jenkins

1)安装Jenkins

```
01. [ root@localhost 下载] # rpm -ivh jenkins-2.121-1.1.noarch.rpm
02. 警告 :jenkins-2.121-1.1.noarch.rpm: 头V4 DSA/SHA1 Signature, 密钥 ID
03. d50582e6: NOKEY
04. 准备中... ##### [ 100%]
05. 正在升级/安装...
06. 1:jenkins-2.121-1.1 #####
```

[Top](#)

2)启动服务

01. [root@localhost 下载] # systemctl start jenkins
02. [root@localhost 下载] # systemctl enable Jenkins

步骤二：初始化Jenkins

1) 访问http://192.168.122.73，Jenkins默认运行在8080端口，如图-3所示：



图-3

2) 查询管理员密码

01. [root@node3 ~] # cat /var/lib/Jenkins/secrets/initialAdminPassword
02. 2bf1c975890f43f98dcf169c2f17c44d

输入管理员密码，继续下一步

3) 安装插件，插件选自定义，只有git是必须要选的，其他都可以不安装，如图-4所示：



图-4

4)管理员用户无需创建，直接使用admin登录，如图-5所示：



图-5

5)完成安装如图-6所示：



图-6

6)修改管理员密码，admin-设置，如图-7所示：



图-7

2 案例2：设置本地仓库

[Top](#)

2.1 问题

1. 解压wordpress4.8
2. 将解压目录配置为git仓库
3. 为当前代码标记为v1.0
4. 更新wordpress到4.9
5. 将代码标记为v2.0

2.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：解压wordpress4.8

```
01. [root@localhost ~] # unzip wordpress-4.8-zh_CN.zip
```

步骤二：将解压目录配置为git仓库

```
01. [root@localhost ~] # cd wordpress/
02. [root@localhost wordpress] # git init
03. [root@localhost wordpress] # git add .
04. [root@localhost wordpress] # git commit -m "wordpress init"
05. [root@localhost wordpress] # git status
06. #位于分支 master
07. 无文件要提交，干净的工作区
```

步骤三：为当前代码标记为v1.0

```
01. [root@localhost wordpress] # git tag v1.0
```

步骤四：更新wordpress到4.9

1) 将wordpress新版本解压到项目中

```
01. [root@localhost wordpress] # cd ..
02. [root@localhost ~] # unzip wordpress-4.9-zh_CN.zip
03. Archive: wordpress-4.9-zh_CN.zip
04. replace wordpress/wp-mail.php? [y]es, [n]o, [A]ll, [N]one, [r]ename: A
```

2)检查git状态

```
01. [root@localhost ~] # cd wordpress/
02. [root@localhost wordpress] # git status
```

[Top](#)

3)将更新/增加的文件确认至仓库

01. [root@localhost wordpress] # git add .
02. [root@localhost wordpress] # git commit -m "upgrade to new version"

4)确认所有项目已提交

01. [root@localhost wordpress] # git status
02. # 位于分支 master
03. 无文件要提交，干净的工作区

步骤五：将代码标记为v2.0

01. [root@localhost wordpress] # git tag v2.0

3 案例3：创建远程仓库

3.1 问题

1. 在gitlab的devops群组下创建wordpress项目
2. 通过devops的主程序员用户将代码上传至wordpress项目
3. 通过web查看项目

3.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：在gitlab的devops群组下创建wordpress项目

1)创建名为devops的群组，如图-8所示：



图-8

2)创建名为devops的群组，如图-9所示：

[Top](#)

群组路径

http://192.168.113.139/ devops

群组名称

devops

描述

tedu devops

群组图标

Choose File ...

No file chosen

The maximum file size allowed is 200KB.

可见等级

私有

该群组及其项目只有其成员能看到。

内部

该群组及其内部项目只有已登录用户能看到。

☒

公开

该群组及其公开项目可以被任何授权的用户看到。

图-9

3)创建名为wordpress的项目，如图-10、图-11所示：

?

devops

tedu devops

全局

按名称过滤...

最近创建

新项目

图-10

空白项目

从模板创建

导入项目

项目路径

http://192.168.113.139/ devops

项目名称

wordpress

项目描述 (可选)

wordpress project for jenkins

可见等级

私有

项目访问权限必须明确授权给每个用户。

内部

该项目允许已登录的用户访问。

☒

公开

该项目允许任何人访问。

创建项目

取消

图-11

4)为wrodpress项目创建主程序员用户，如图-12、图-13、图-14、图-15所示：

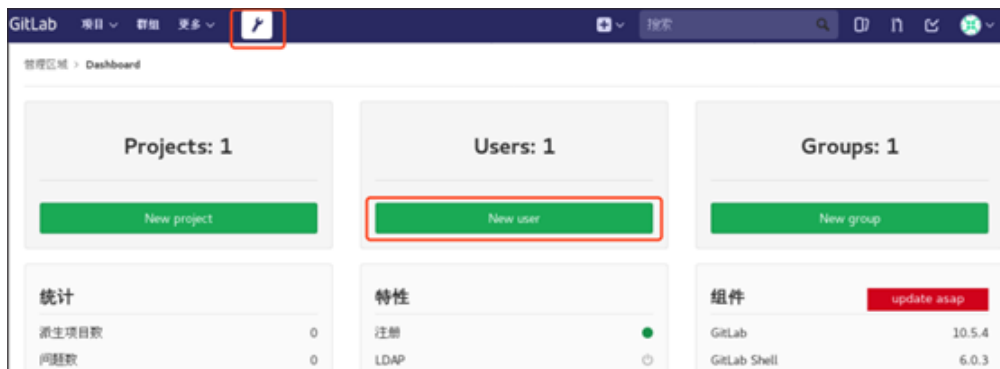


图-12

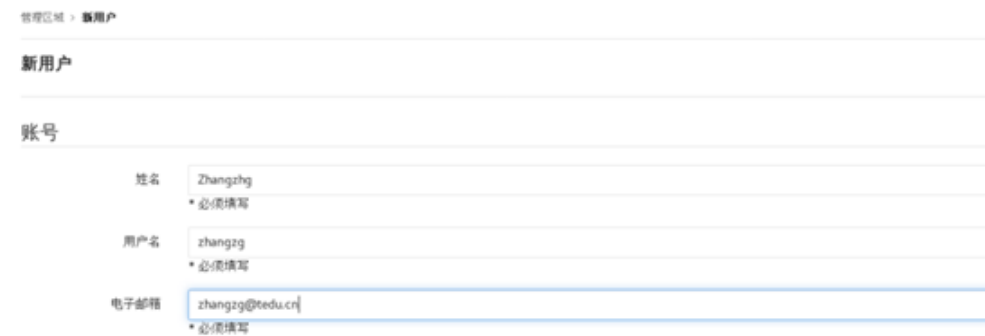


图-13

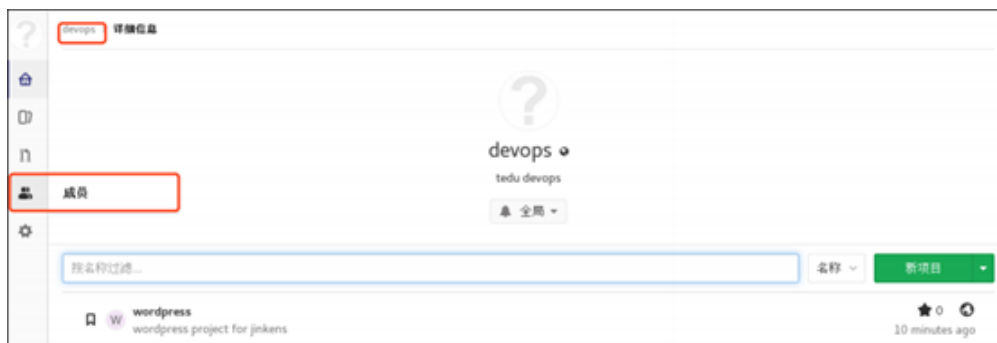


图-14



图-15

5) 用户生成ssh密钥

01. [root@localhost ~] # ssh-keygen -t rsa -C "zhangzg@tedu.cn" -b 4096
02. [root@localhost ~] # cat ~/.ssh/id_rsa.pub
03. ssh-rsa
04. AAAAB3NzaC1yc2EAAAADAQABAAQCAQC4idWslzdFWQM3mvbFCC5SL
05. RSqXnoT2wFodoOfkdbbcSOeJ1RX6GgZjW+PTDjsu7OfiCw
06. +ZOSlaeY0xQWcl1ExVn2aDMNkr7Lr2VyHEpo7cJzoGI0c6vQBN83VZKcY
07. dJzEboxshbRg2MKHN85cdxVWAQOqaHs105thHBCI3Um6+saVhAt9Use

[Top](#)

08. QQOQYBIH002QJ

6)新用户登陆重置密码，然后设置ssh密钥，如图-16所示：

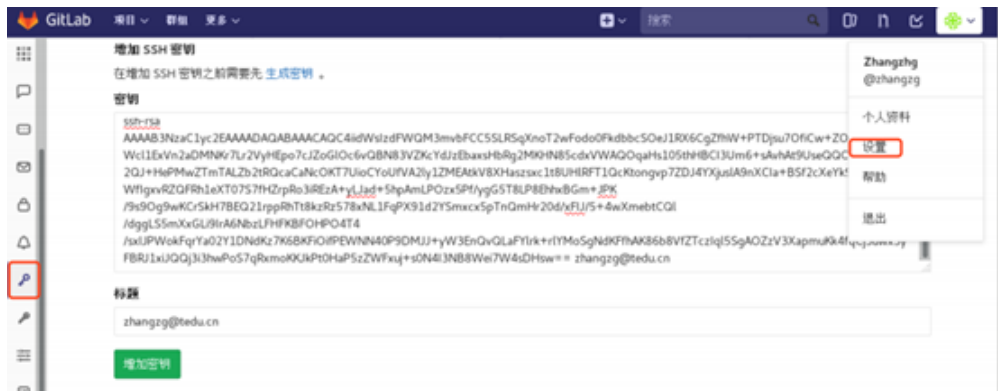


图-16

步骤二：通过devops的主程序员用户将代码上传至wordpress项目

因为本地wordpress已经是git版本库了，所以采用以下方式进行上传：

01. [root@localhost ~] # cd wordpress/
02. [root@localhost wordpress] # git remote rename origin old- origin
03. error: 不能重命名配置小节 'remote.origin' 到 'remote.old- origin'
04. 上述错误可忽略
05. [root@localhost wordpress] # git remote add origin
06. git@192.168.113.139: devops/wordpress.git
07. [root@localhost wordpress] # git push - u origin -- all
08. [root@localhost wordpress] # git push - u origin -- tags

步骤三：通过web查看项目

所有的tag可以通过<http://192.168.113.139/devops/wordpress/tags>访问，如图-17所示：



图-17

4 案例4：构建工程

4.1 问题

1. 创建一个自由风格的项目
2. 源码管理采用git
3. 构建时可以指定tag
4. 构建tag为v1.0的源码

[Top](#)

4.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：创建一个自由风格的项目

1) 下载GIT插件，为了使得Jenkins可以使用git的tag，需要下载git parameter插件，如图-18所示：



图-18

2) 新建任务，如图-19所示：



图-19

3) 选择自由风格，如图-20所示：

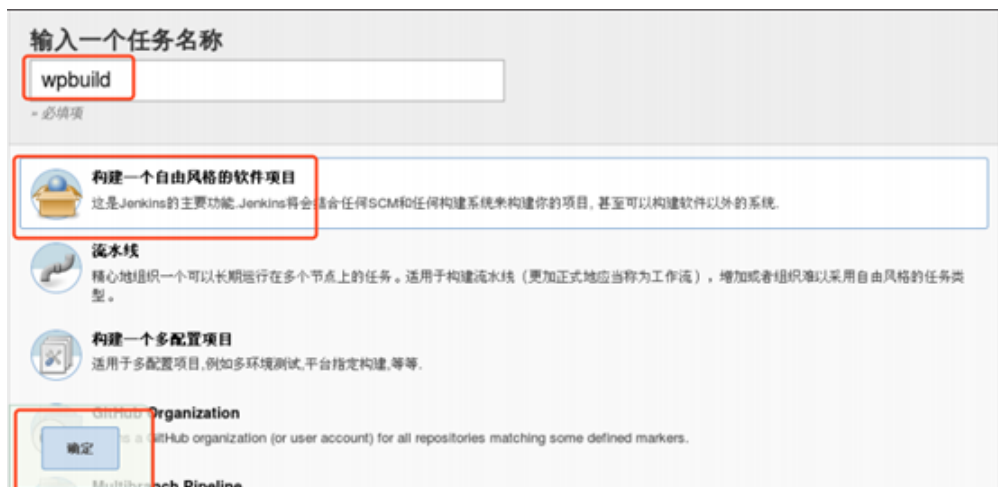


图-20

4) 添加Git Parameter参数，如图-21、图-22所示：

[Top](#)



图-21

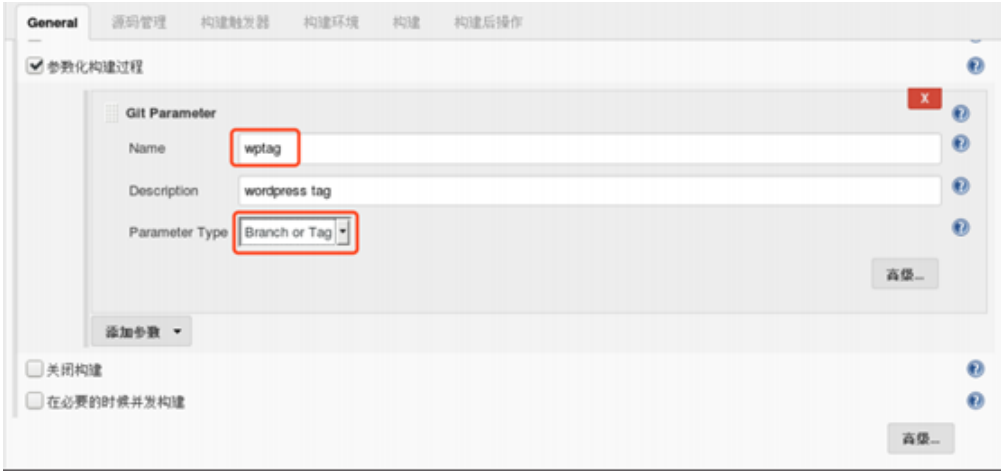


图-22

步骤二：源码管理采用git

1)源码采用git，如图-23所示：

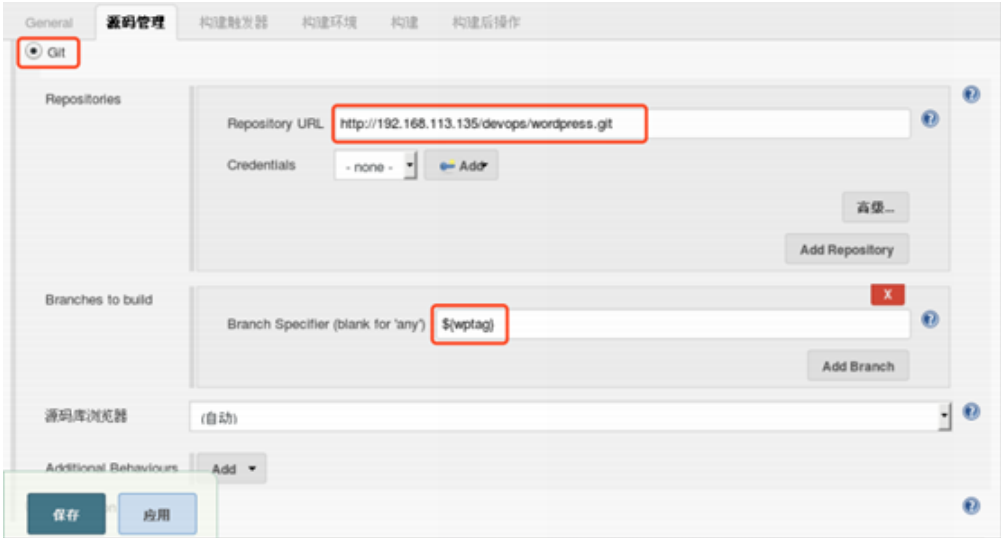


图-23

2)将源码checkout到子目录，如图-24、图-25所示：

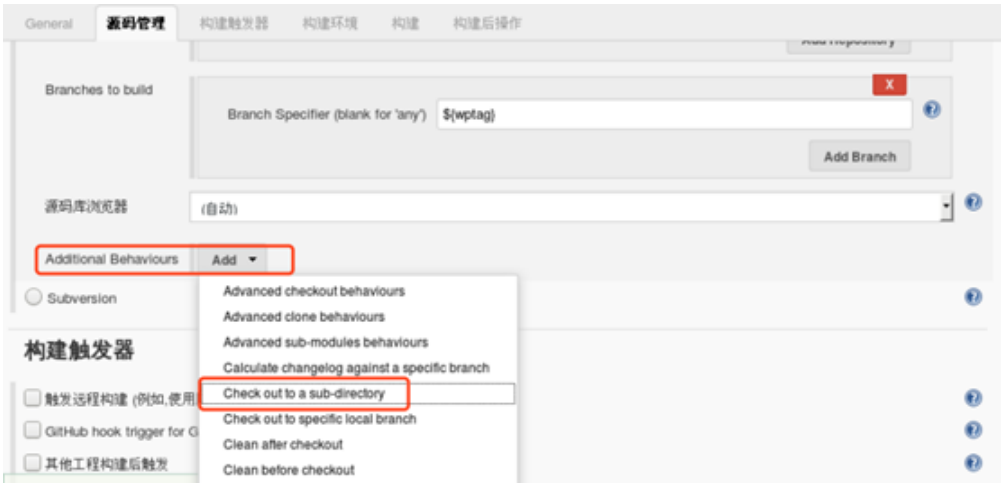


图-24

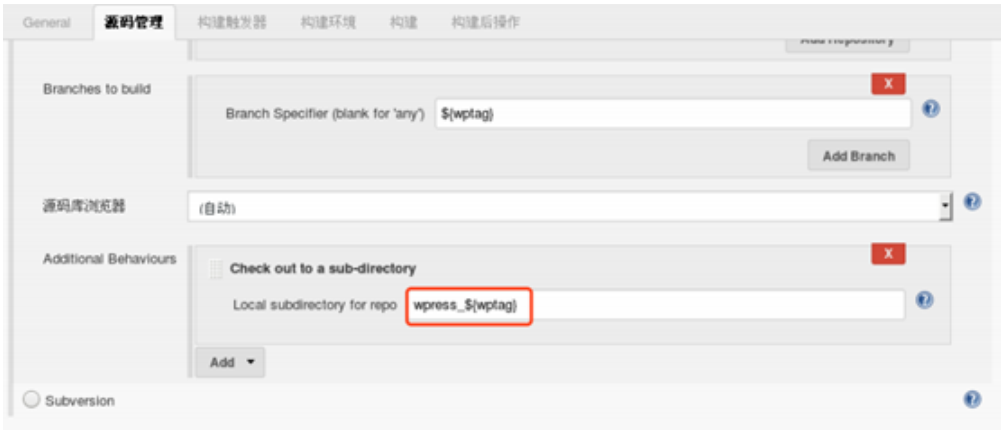


图-25

步骤三：构建工程，构建tag为v1.0的源码

1)构建工程，如图-26所示：



图-26

2)选择指定的标签，如图-27所示：



图-27

步骤四：检验结果

1)选择指定标签，如图-28所示：



图-28

2)查看日志输出，如图-29、图-30所示：



图-29



图-30

3)查看本地结果，构建好的项目出现在/var/lib/jenkins目录下：

01. [root@localhost ~] # cd /var/lib/jenkins/workspace/
02. [root@localhost workspace] # ls
03. wpbuild
04. [root@localhost workspace] # ls wpbuild/
05. wpress_v1.0

5 案例5：修改工程

5.1 问题

修改构建工程，要求如下：

1. 将下载的应用程序打包放在/var/www/html/deploy/packages目录下
2. 为打包应用程序计算md5值

5.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：配置分发服务器

通过web服务为应用服务器提供应用程序，下载目录为/var/www/deploy/packages作：

01. [root@localhost ~] # yum install -y httpd
02. [root@localhost ~] # mkdir -p /var/www/html/deploy/packages
03. [root@localhost ~] # chown -R jenkins.jenkins /var/www/html/deploy/
04. [root@localhost ~] # systemctl start httpd
05. [root@localhost ~] # systemctl enable httpd

步骤二：修改工程

1) 为下载的应用打包，生成md5在工程中增加构建步骤，如图-31、图-32所示：

[Top](#)



图-31



图-32

2)构建，如图-33所示：

将从gitlab上下载的内容，拷贝到/var/www/html/deploy/packages目录下，将拷贝到的文件打包，删除原始目录，将压缩包生成md5值，存入md5文件中



图-33

步骤三：构建测试

1)执行构建工程，如图-34所示：

[Top](#)



图-34

2)检查分发服务器的相关目录：

01. [root@localhost ~] # ls /var/www/html/deploy/packages/
02. wpres_v1.0.tar.gz wpres_v1.0.tar.gz.md5
03. [root@localhost ~] # cat /var/www/html/deploy/packages/
04. wpres_v1.0.tar.gz.md5
05. e4taef54a5f580b4e08d5245cc95268

6 案例6：创建版本文件

6.1 问题

为了记录应用的当前版本和前一个版本，创建两个工程：

1. 创建live_version，记录应用程序当前版本
2. 创建last_version，记录应用程序前一个版本

6.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：创建live_version文件

1)新建任务，如图-35、图-36所示：



图-35

[Top](#)

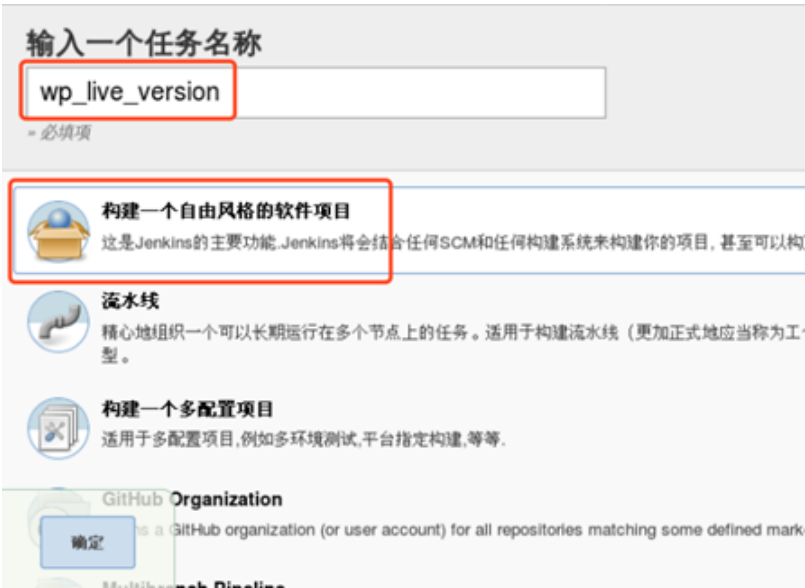


图-36

2) 参数构建，图-37、图-38、图-39、图-40所示：



图-37



图-38

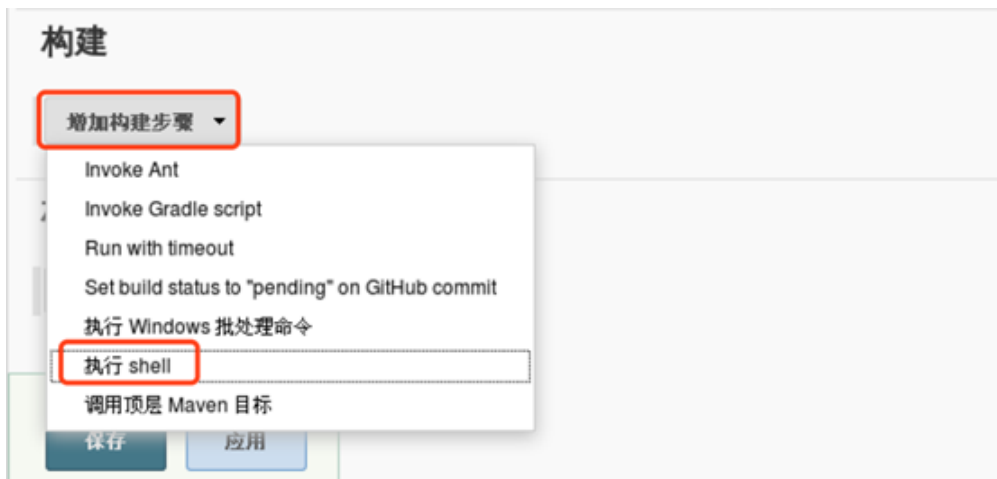


图-39



图-40

3)执行构建工程，如图-41所示：



图-41

步骤二：创建last_version文件

1)修改live_verion文件shell命令，命令如下：

```
[f /var/www/deploy/live_verion] && cat /var/www/deploy/live_verion > /var/www/deploy/last_verion, #如果找到/var/www/deploy/live_verion文件，就将/var/www/deploy/live_verion文件内容写入/var/www/deploy/last_verion文件，如图-42所示：
```

[Top](#)



图-42

2)开始构建，构建版本为v1.0，如图-43、图-44所示：

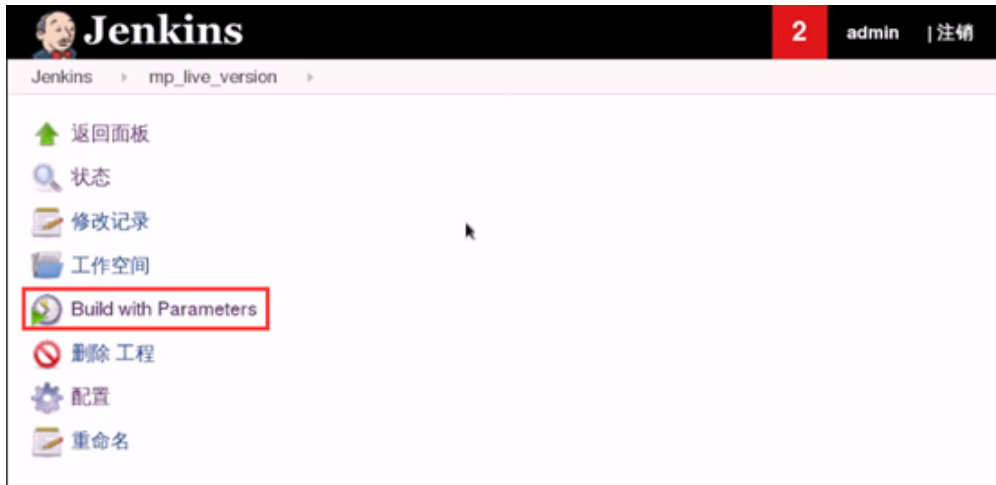


图-43

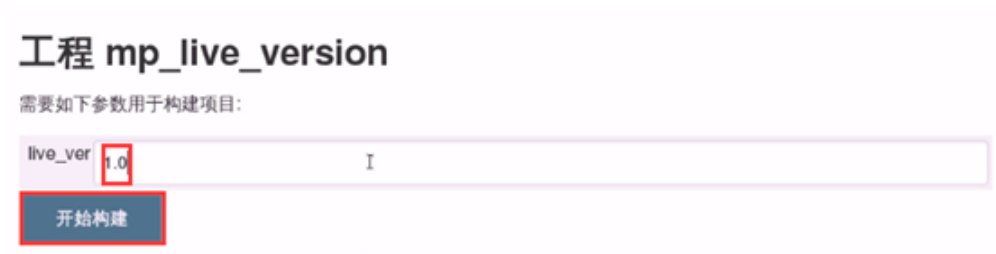


图-44

3)查看构建结果

01. `[root@localhost ~] # ls /var/www/deploy /`
02. `live_version last_verion packages`
03. `[root@localhost ~] # cat /var/www/deploy/live_version`
04. `v1.0`
05. `[root@localhost ~] # cat /var/www/deploy/last_verion #此时last_verion为空`

4)重新构建，构建版本为v2.0，如图-43、图-45所示：

[Top](#)

工程 mp_live_version

需要如下参数用于构建项目：

live_ver

2.0

开始构建

图-45

5)查看构建结果

```

01. [ root@localhost ~] # ls /var/www/deploy /
02. live_version last_version packages
03. [ root@localhost ~] # cat /var/www/deploy/live_version
04. v1.0
05. [ root@localhost ~] # cat /var/www/deploy/last_version
06. v2.0

```

7 案例7：发布应用

7.1 问题

编写deploy_web.py应用发布程序：

1. 根据分发服务器的版本，下载对应的应用
2. 校验应用程序
3. 发布指定程序

7.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：编写脚本

```

01. [ root@localhost ~] # vim deploy_web.py
02.
03. import requests
04. import os
05. import hashlib
06. import tarfile
07. from urllib import request
08.
09. #使用url进行get请求，获取网站文本信息，获取当前版本
10. def get_webdata( url ):
11.     r = requests.get( url )
12.     return r.text
13. #将网址内容下载到fname文件中
14. def download( url, fname ):
15.     html = request.urlopen( url )
16.     with open( fname, 'wb' ) as fobj:

```

[Top](#)

```
17.         while True:
18.             data = html.read( 1024)
19.             if not data:
20.                 break
21.             fobj.write( data)
22.
23. #检查文件MD5值
24. def check_md5( fname) :
25.     m = hashlib.md5( )
26.
27.     with open( fname, 'rb') as fobj:
28.         while True:
29.             data = fobj.read( 4096)
30.             if not data:
31.                 break
32.             m.update( data)
33.
34.     return m.hexdigest( )
35.
36. #部署版本，应用发布代码
37. def deploy( app): # /var/www/packages/my project_2.0.tar.gz
38.     #切换路径到/var/www/packages路径下
39.     os.chdir( '/var/www/packages')
40.     #解压缩my project_2.0.tar.gz文件
41.     tar = tarfile.open( app, 'r:gz')
42.     tar.extractall( )
43.     tar.close( )
44.     #将.tar.gz替换成空字符串
45.     src = app.replace( '.tar.gz', '')
46.     #创建/var/www/html/my site软链接
47.     dst = '/var/www/html/my site'
48.     if os.path.exists( dst): #如果文件存在
49.         os.unlink( dst) #删除
50.     os.symlink( src, dst) #否则创建软链接
51.
52. if __name__ == '__main__':
53.     #调用get_webdata( ) 函数，目的是以发布服务器'http://192.168.122.73/live_version'网址为参数获取
54.     ver = get_webdata( 'http://192.168.122.73/live_version').strip( )
55.     app_name = 'my project_%s.tar.gz' % ver
56.     #app_url为下载my project_2.0.tar.gz文件网址
57.     app_url = 'http://192.168.122.73/packages/' + app_name
58.     #目标文件
59.     app_path = os.path.join( '/var/www/packages', app_name)
60.     #调用download( ) 函数，目的是从app_url网址读取数据存入目标文件
61.     download( app_url, app_path)
62.     #调用check_md5( ) 函数，目的是计算目标文件MD5值
```

[Top](#)

```
63.     local_md5 = check_md5( app_path)
64.     #调用get_webdata() 函数，目的从发布服务器网址获取md5值
65.     remote_md5 = get_webdata( app_url + '.md5').strip()
66.     #如果目标文件md5值和发布服务器提供的md5值相等，确认下载的文件无误，调用deploy() 函数，以
67.     if local_md5 == remote_md5:
68.         deploy( app_path)
```

步骤二：测试脚本

```
01. [ root@localhost ~] # python3 deploy_web.py
02. #增加了my site文件
03. [ root@localhost ~] # ls /var/www/html
04. aaa index.html my post.html my site
05. [ root@localhost ~] # ls /var/www/html/my site
06. index.html
```

部署结果如图-46所示：

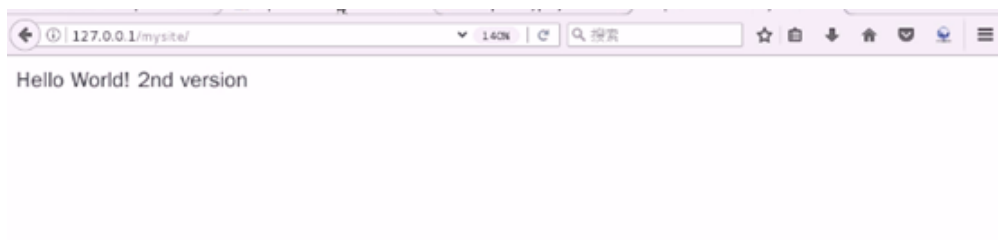


图-46