一：本文通过jenkins调用shell脚本的的方式完成从Git服务器获取代码、打包、部署到web服务器、将web服务器从负载均衡器删除、解压、复制配置文件、创建软连接、测试每一台web服务器、将web服务器添加至负载均衡、回滚到任意指定版本、一键回滚到上一版本等功能，脚本放在www用户家目录并使用www用户身份执行，每个web服务器也都使用www用户运行web服务，且UID相同web目录和权限都一致，更严格的标准化可以带来更安全的生产环境和更高的效率：  
1.1：在jenkins项目配置中调用shell脚本与环境准备：  
1.1.1：#jenkins-项目-配置：



1.1.2：www用户家目录中的脚本内容：

IMG_257

$ cat code\_deploy.sh

#!/bin/bash

#Dir List 部署节点(即部署节点需要做的操作)

# mkdir -p /deploy/code/web-demo

# mkdir -p /deploy/config/web-demo/base

# mkdir -p /deploy/config/web-demo/other

# mkdir /deploy/tmp

# mkdir /deploy/tar

# chown -R www.www /deploy

# chown -R www.www /webroot

# chown -R www.www /opt/webroot/

# chown -R www.www /webroot

# 需要在客户端节点做的操作

# mkdir /opt/webroot

# mkdir /webroot

# chown -R www.www /webroot

# chown -R www.www /opt/webroot/

# chown -R www.www /webroot

# [www@ ~]$ touch /webroot/web-dem

# Node List 服务器节点

PRE\_LIST="192.168.3.12" # 预生产节点

GROUP1\_LIST="192.168.3.12 192.168.3.13"

GROUP2\_LIST="192.168.3.13"

ROLLBACK\_LIST="192.168.3.12 192.168.3.13"

# 日志日期和时间变量

LOG\_DATE='date "+%Y-%m-%d"' # 如果执行的话后面执行的时间，此时间是不固定的，这是记录日志使用的时间

LOG\_TIME='date "+%H-%M-%S"'

# 代码打包时间变量

CDATE=$(date "+%Y-%m-%d") # 脚本一旦执行就会取一个固定时间赋值给变量，此时间是固定的

CTIME=$(date +"%H-%M-%S")

# shell env 脚本位置等变量

SHELL\_NAME="deploy.sh" # 脚本名称

SHELL\_DIR="/home/www/" # 脚本路径

SHELL\_LOG="${SHELL\_DIR}/${SHELL\_NAME}.log" # 脚本执行日志文件路径

# code env 代码变量

PRO\_NAME="web-demo" # 项目名称的函数

CODE\_DIR="/deploy/code/web-demo" # 从版本管理系统更新的代码目录

CONFIG\_DIR="/deploy/config/web-demo" # 保存不同项目的配置文件，一个目录里面就是一个项目的一个配置文件或多个配置文件

TMP\_DIR="/deploy/tmp" # 临时目录

TAR\_DIR="/deploy/tar" # 打包目录

LOCK\_FILE="/tmp/deploy.lock" # 锁文件路径

usage(){ # 使用帮助函数

echo $"Usage: $0 [ deploy | rollback [ list | emergency | version ]"

}

writelog(){ # 写入日志的函数

LOGINFO=$1 # 将参数作为日志输入

echo "${CDATE} ${CTIME} : ${SEHLL\_NAME} : ${LOGINFO}" >> ${SHELL\_LOG}

}

# 锁函数

shell\_lock(){

touch ${LOCK\_FILE}

}

# 解锁函数

shell\_unlock(){

rm -f ${LOCK\_FILE}

}

# 获取代码的函数

code\_get(){

echo "code\_get"

writelog code\_get

cd $CODE\_DIR && git pull # 进入到代码目录更新代码，此处必须免密码更新，此目录仅用于代码更新不能放其他任何文件

cp -rf ${CODE\_DIR} ${TMP\_DIR}/ # 临时保存代码并重命名，包名为时间+版本号，准备复制到web服务器

API\_VERL=$(git show | grep commit | cut -d ' ' -f2)

API\_VER=$(echo ${API\_VERL:0:8}) # 版本号

}

code\_build(){ # 代码编译函数

echo code\_build

}

code\_config(){ # 配置文件函数

writelog "code\_config"

/bin/cp -rf ${CONFIG\_DIR}/base/\* ${TMP\_DIR}/"${PRO\_NAME}" # 将配置文件放在本机保存配置文件的临时目录，用于暂时保存代码项目

PKG\_NAME="${PRO\_NAME}"\_"$API\_VER"\_"${CDATE}-${CTIME}" # 定义代码目录名称

cd ${TMP\_DIR} && mv ${PRO\_NAME} ${PKG\_NAME} # 重命名代码文件为web-demo\_123-20170629-11-19-10格式

}

code\_tar(){ # 对代码打包函数

writelog code\_tar

cd ${TMP\_DIR} && tar czf ${PKG\_NAME}.tar.gz ${PKG\_NAME} --exclude=".git" # 将目录打包成压缩文件，便于网络传输

writelog "${PKG\_NAME}.tar.gz packaged success" # 记录打包成功的日志

}

code\_scp(){ # 代码压缩包scp到客户端的函数

writelog "code\_scp"

for node in $PRE\_LIST;do # 循环服务器节点列表

scp ${TMP\_DIR}/${PKG\_NAME}.tar.gz $node:/opt/webroot/ # 将压缩后的代码包复制到web服务器的/opt/webroot

done

for node in $GROUP1\_LIST;do # 循环服务器节点列表

scp ${TMP\_DIR}/${PKG\_NAME}.tar.gz $node:/opt/webroot/ # 将压缩后的代码包复制到web服务器的/opt/webroot

done

}

cluster\_node\_add(){ #将web服务器添加至前端负载

echo cluster\_node\_add

}

cluster\_node\_remove(){ # 将web服务器从集群移除函数(正在部署的时候应该不处理业务)

writelog "cluster\_node\_remove"

}

url\_test(){

URL=$1

curl -s --head $URL |grep '200 OK'

if [ $? -ne 0 ];then

shell\_unlock;

writelog "test error" && exit;

fi

}

pre\_deploy(){ # 代码解压部署函数,预生产节点

writelog "pre\_deploy"

for node in ${PRE\_LIST};do # 循环预生产服务器节点列表

cluster\_node\_remove ${node} # 部署之前将节点从前端负载删除

echo "pre\_deploy, cluster\_node\_remove ${node}"

ssh ${node} "cd /opt/webroot && tar zxf ${PKG\_NAME}.tar.gz" #分别到web服务器执行压缩包解压命令

ssh ${node} "rm -f /webroot/web-demo && ln -s /opt/webroot/${PKG\_NAME} /webroot/web-demo" # 整个自动化的核心，创建软连接

done

}

pre\_test(){ # 预生产主机测试函数

for node in ${PRE\_LIST};do # 循环预生产主机列表

curl -s --head http://${node}:9999/index.html | grep "200 OK" # 测试web界面访问

if [ $? -eq 0 ];then # 如果访问成功

writelog " ${node} Web Test OK!" # 记录日志

echo " ${node} Web Test OK!"

cluster\_node\_add ${node} # 测试成功之后调用添加函数把服务器添加至节点,

writelog "pre,${node} add to cluster OK!" # 记录添加服务器到集群的日志

else # 如果访问失败

writelog "${node} test no OK" # 记录日志

echo "${node} test not OK"

shell\_unlock # 调用删除锁文件函数

break # 结束部署

fi

done

}

group1\_deploy(){ # 代码解压部署函数

writelog "group1\_code\_deploy"

for node in ${GROUP1\_LIST};do # 循环生产服务器节点列表

cluster\_node\_remove $node

echo "group1, cluster\_node\_remove $node"

ssh ${node} "cd /opt/webroot && tar zxf ${PKG\_NAME}.tar.gz" # 分别到各web服务器节点执行压缩包解压命令

ssh ${node} "rm -f /webroot/web-demo && ln -s /opt/webroot/${PKG\_NAME} /webroot/web-demo" # 整个自动化的核心，创建软连接

done

scp ${CONFIG\_DIR}/other/192.168.3.13.server.xml 192.168.3.13:/webroot/web-demo/server.xml # 将差异项目的配置文件scp到此web服务器并以项目结尾

}

group1\_test(){ # 生产主机测试函数

for node in ${PRE\_LIST};do # 循环生产主机列表

curl -s --head http://${node}:9999/index.html | grep "200 OK" #测试web界面访问

if [ $? -eq 0 ];then #如果访问成功

writelog " ${node} Web Test OK!" #记录日志

echo "group1\_test,${node} Web Test OK!"

cluster\_node\_add

writelog " ${node} add to cluster OK!" #记录将服务器 添加至集群的日志

else #如果访问失败

writelog "${node} test no OK" #记录日志

echo "${node} test no OK"

shell\_unlock # 调用删除锁文件函数

break # 结束部署

fi

done

}

emergency\_code\_get(){ #获取代码的函数

writelog "code\_get"

cd ${CODE\_DIR} && git reset --hard HEAD^ #进入到代码目录更新代码，此处必须免密码更新，此目录仅用于代码更新不能放其他任何文件

/bin/cp -rf ${CODE\_DIR} ${TMP\_DIR}/ #临时保存代码并重命名，包名为时间+版本号，准备复制到web服务器

API\_VERL=$(git show | grep commit | cut -d ' ' -f2)

API\_VER=$(echo ${API\_VERL:0:8}) #取八位

}

emergency(){ #紧急回退到上一个版本函数

emergency\_code\_get #执行将代码回退到上一个版本函数

code\_build; #如果要编译执行编译函数

code\_config; #cp配置文件

code\_tar; #打包

code\_scp; #scp到服务器

cluster\_node\_remove;

pre\_deploy; #预生产环境部署

pre\_test; #预生产环境测试

group1\_deploy; #生产环境部署

group1\_test; #生产环境测试

code\_config; #cp差异文件

#code\_test; #代码测试

shell\_unlock #执行完成后删除锁文件

}

rollback\_fun(){

for node in $ROLLBACK\_LIST;do # 循环服务器节点列表

# 注意一定要加"号，否则无法在远程执行命令

ssh $node "rm -f /webroot/web-demo && ln -s /opt/webroot/$1 /webroot/web-demo" # 立即回滚到指定的版本，$1即指定的版本参数

echo "${node} rollback success!"

done

}

rollback(){ # 代码回滚主函数

if [ -z $1 ];then

shell\_unlock # 删除锁文件

echo "Please input rollback version" && exit 3;

fi

case $1 in # 把第二个参数做当自己的第一个参数

list)

ls -l /opt/webroot/\*.tar.gz

;;

\*)

rollback\_fun $1

esac

}

main(){

if [ -f $LOCK\_FILE ] ;then # 先判断锁文件在不在,如果有锁文件直接退出

echo "Deploy is running" && exit 10

fi

DEPLOY\_METHOD=$1 # 避免出错误将脚本的第一个参数作为变量

ROLLBACK\_VER=$2

case $DEPLOY\_METHOD in

deploy) # 如果第一个参数是deploy就执行以下操作

shell\_lock; # 执行部署之前创建锁。如果同时有其他人执行则提示锁文件存在

code\_get; # 获取代码

code\_build; # 如果要编译执行编译函数

code\_config; # cp配置文件

code\_tar; # 打包

code\_scp; # scp到服务器

pre\_deploy; # 预生产环境部署

pre\_test; # 预生产环境测试

group1\_deploy; # 生产环境部署

group1\_test; # 生产环境测试

shell\_unlock; # 执行完成后删除锁文件

;;

rollback) # 如果第一个参数是rollback就执行以下操作

shell\_lock; # 回滚之前也是先创建锁文件

rollback $ROLLBACK\_VER;

shell\_unlock; # 执行完成删除锁文件

;;

emergency)

emergency; #紧急回退就不需要参数了，但是在执行的时候要确认一下是否要紧急回退，避免输入错误

;;

\*)

usage;

esac

}

main $1 $2

IMG_258

1.1.4：修改当前web页面：

IMG_259

[www@master ~]$ cd web-demo

[www@master web-demo]$ echo "<h1>jenkins deploy test" > index.html

[www@master web-demo]$ git add index.html

[www@master web-demo]$ git commit -m "jenkins deploy test"

[master 9a43cf5] jenkins deploy test1 file changed, 1 insertion(+), 1 deletion(-)

[www@master web-demo]$ git push -u origin master

Counting objects: 5, done.

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

Writing objects: 100% (3/3), 276 bytes | 0 bytes/s, done.

Total 3 (delta 1), reused 0 (delta 0)

To git@192.168.3.198:web/web-demo.git

beb37cb..9a43cf5 master -> master

Branch master set up to track remote branch master from origin.

IMG_260



1.4：回滚到任意版本：  
1.4.1:在哪看回滚到的版本?:  
$ ll /deploy/tmp/ #部署服务器，web服务器在nginx定义的目录查看版本

1.4.3：在jenkins执行回滚：

IMG_262

[root@slave01 ~]# ll /opt/webroot/

total 20672

drwxr-xr-x 5 www www 4096 Jun 26 11:36 web-demo\_123\_2017-06-26-11-36-44

-rw-rw-r-- 1 www www 1243347 Jun 28 22:03 web-demo\_123\_2017-06-26-11-36-44.tar.gz

drwxr-xr-x 5 www www 4096 Jun 26 11:39 web-demo\_123\_2017-06-26-11-39-02

-rw-rw-r-- 1 www www 1243347 Jun 28 22:06 web-demo\_123\_2017-06-26-11-39-02.tar.gz

drwxr-xr-x 5 www www 4096 Jun 26 12:04 web-demo\_123\_2017-06-26-12-04-19

-rw-rw-r-- 1 www www 1243351 Jun 28 22:31 web-demo\_123\_2017-06-26-12-04-19.tar.gz

drwxr-xr-x 5 www www 4096 Jun 26 12:16 web-demo\_123\_2017-06-26-12-16-49

-rw-rw-r-- 1 www www 1243347 Jun 28 22:43 web-demo\_123\_2017-06-26-12-16-49.tar.gz

drwxr-xr-x 5 www www 4096 Jun 26 12:18 web-demo\_123\_2017-06-26-12-18-09

-rw-rw-r-- 1 www www 1243347 Jun 28 22:45 web-demo\_123\_2017-06-26-12-18-09.tar.gz

drwxr-xr-x 5 www www 4096 Jun 26 12:18 web-demo\_123\_2017-06-26-12-18-57

-rw-rw-r-- 1 www www 1243369 Jun 28 22:46 web-demo\_123\_2017-06-26-12-18-57.tar.gz-rw-rw-r-- 1 www www 45 Jun 29 06:21 web-demo\_\_2017-06-30-14-28-54.tar.gz

drwxrwxr-x 2 www www 4096 Jun 30 14:30 web-demo\_\_2017-06-30-14-30-22

-rw-rw-r-- 1 www www 130 Jun 29 06:23 web-demo\_\_2017-06-30-14-30-22.tar.gz

drwxrwxr-x 2 www www 4096 Jun 30 14:31 web-demo\_\_2017-06-30-14-31-53

-rw-rw-r-- 1 www www 219 Jun 29 06:24 web-demo\_\_2017-06-30-14-31-53.tar.gz

drwxrwxr-x 4 www www 4096 Jun 30 14:59 web-demo\_75463f1b\_2017-06-30-14-59-58

-rw-rw-r-- 1 www www 1236456 Jun 29 06:52 web-demo\_75463f1b\_2017-06-30-14-59-58.tar.gz

drwxrwxr-x 4 www www 4096 Jun 30 15:05 web-demo\_75463f1b\_2017-06-30-15-05-57

-rw-rw-r-- 1 www www 1236450 Jun 29 06:58 web-demo\_75463f1b\_2017-06-30-15-05-57.tar.gz

drwxrwxr-x 4 www www 4096 Jul 10 14:01 web-demo\_75463f1b\_2017-07-10-14-01-34

-rw-rw-r-- 1 www www 1236446 Jul 10 14:01 web-demo\_75463f1b\_2017-07-10-14-01-34.tar.gz

drwxrwxr-x 4 www www 4096 Jun 30 15:18 web-demo\_78869143\_2017-06-30-15-18-29

-rw-rw-r-- 1 www www 1236465 Jun 29 07:11 web-demo\_78869143\_2017-06-30-15-18-29.tar.gz

drwxrwxr-x 4 www www 4096 Jul 10 14:00 web-demo\_78869143\_2017-07-10-14-00-35

-rw-rw-r-- 1 www www 1236453 Jul 10 14:00 web-demo\_78869143\_2017-07-10-14-00-35.tar.gz

drwxrwxr-x 3 www www 4096 Jun 30 14:14 web-demo\_91d09cc2\_2017-06-30-14-14-42

-rw-rw-r-- 1 www www 1236371 Jun 29 06:06 web-demo\_91d09cc2\_2017-06-30-14-14-42.tar.gz

drwxrwxr-x 3 www www 4096 Jun 30 14:15 web-demo\_91d09cc2\_2017-06-30-14-15-16

-rw-rw-r-- 1 www www 1236382 Jun 29 06:08 web-demo\_91d09cc2\_2017-06-30-14-15-16.tar.gz

drwxrwxr-x 4 www www 4096 Jul 10 14:08 web-demo\_9a43cf55\_2017-07-10-14-08-46

-rw-rw-r-- 1 www www 1233708 Jul 10 14:08 web-demo\_9a43cf55\_2017-07-10-14-08-46.tar.gz

drwxrwxr-x 4 www www 4096 Jun 30 15:21 web-demo\_b8f3be43\_2017-06-30-15-21-55

-rw-rw-r-- 1 www www 1236454 Jun 29 07:14 web-demo\_b8f3be43\_2017-06-30-15-21-55.tar.gz

drwxrwxr-x 4 www www 4096 Jul 10 12:34 web-demo\_b8f3be43\_2017-07-10-12-34-00

-rw-rw-r-- 1 www www 1236462 Jul 10 12:34 web-demo\_b8f3be43\_2017-07-10-12-34-00.tar.gz

drwxrwxr-x 4 www www 4096 Jun 30 14:57 web-demo\_dcfb44f0\_2017-06-30-14-57-10

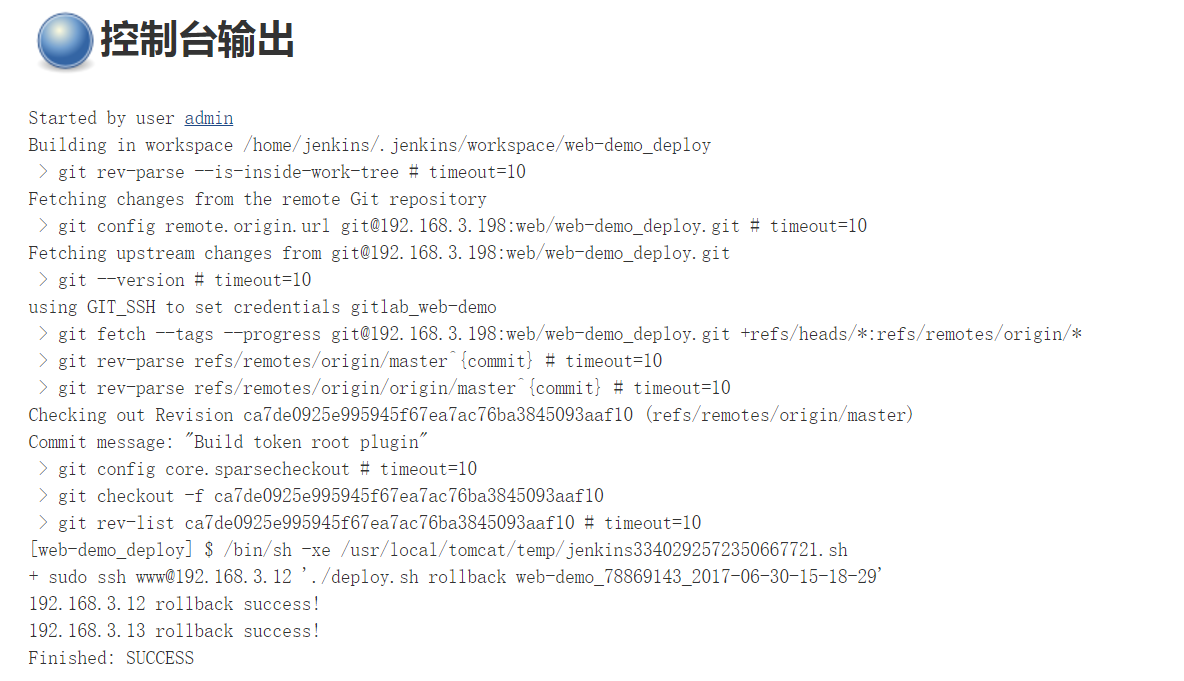
-rw-rw-r-- 1 www www 1236447 Jun 29 06:50 web-demo\_dcfb44f0\_2017-06-30-14-57-10.tar.gz

IMG_263

1.4.2:回滚任意版本就将版本的参数传递给脚本，脚本会将web-demo的链接重新指向传递的版本(参数)，比如我要回滚到web-demo\_78869143\_2017-06-30-15-18-29这个版本，则jenkins的配置为：



1.4.3：在jenkins执行回滚：  
1.4.4:执行回滚的信息:



1.4.5:访问web界面测试任意版本回滚是否成功：

