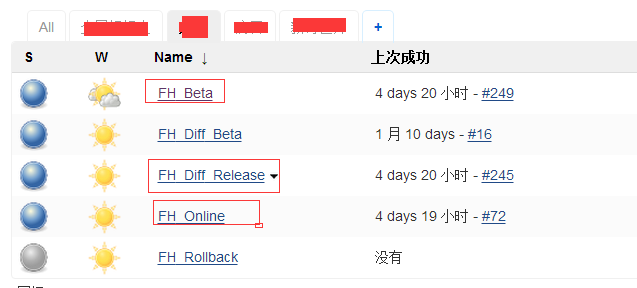
转自：<https://www.cnblogs.com/kevingrace/p/6022447.html>

在我们的日常运维工作中，使用jenkins来完成业务代码发版上线是至关重要的一环。  
前面已经提到在jenkins上添加用户权限的操作，今天重点说下如何在jenkins下构建项目工程进行代码发布？

在此简单介绍下jenkins进行代码发布的流程（根据本人的经历而述）。  
1）代码上线要经历四个场景：Dev开发环境-->Test测试环境-->Beta验收环境-->Online线上环境  
Dev开发环境：开发人员在开发机上自行开发，开发后将代码上传到svn/git版本控制系统里。  
Test测试环境：将代码从svn下载并同步到测试机（Test环境发版），通知测试同事进行上线前的业务测试。  
Beta验收环境：测试同事测试ok后，将代码同步到Beta机上（Beta环境发版），然后通知产品/运营同事进行上线前的验收。  
Online线上环境：待Beta验收ok后，再将代码同步到线上机器上，最终完成Online发版。

2）Test测试环境下的发版，没有放到jenkins上操作（当然也是可以放到kenins上构建测试项目工程的），直接是在服务器上写的发版脚本进行的。  
测试环境下的发版操作流程可参考：[http://www.cnblogs.com/kevingrace/p/5896010.html](http://www.cnblogs.com/kevingrace/p/5896010.html" \t "https://www.cnblogs.com/kevingrace/p/_blank)

3）Beta验收环境和Online线上环境的发版是放在jenkins上构建项目工程完成的，上线发版分为全量发版（全部同步）和增量发版（个别文件同步）。  
发版是根据svn的tag标记，由release区分版本号进行的。具体情况如下：



全量发布流程：  
第一步：构建FH\_Diff\_Release项目，构建时会提示选择一个release版本号。根据上线人员要求的release版本号（一般是最新的release号）进行构建。将代码从svn下载到jenkins本地(即放到对应项目的workspace目录下，也就是放在/usr/local/tomcat7/webapps/jenkins/jobs/FH\_Diff\_Release下)。构建后  
发布日志里会留有一个版本号（比如20161028183745）  
第二步：构建FH\_Beta项目，构建时会提示输入VERSION版本号。使用FH\_Diff\_Release最近一次构建后的版本号（如上面的20161028183745）构建FH\_Beta。构建后发布日志里同样也会留有一个版本号（也是20161028183745）。到这里完成Beta环境的验收工作，验收完成后  
才可以进行代码上线工作。  
第三步：构建FH\_Online项目，构建时会提示输入VERSION版本号。使用FH\_Beta最近一次构建后的版本号（也就是上面的20161028183745）构建FH\_Online。至此完成代码全量上线。

增量发布流程：  
重复上面的第一步和第二部，将代码发布到Beta环境进行验收，验收通过后，就可以进行代码上线了。  
第三步：构建FH\_Diff\_Beta，构建时会提示输入FILE\_LIST（一般不会是文件的全路径，是文件的部分路径，具体参考shell脚本里对此参数的引用设置）。输入要同步的个别文件进行构建即可。构建后发布日志里也会留有一个版本号（比如20161014371145）.  
第四部：构建FH\_Online项目，构建时会提示输入VERSION版本号。使用FH\_Diff\_Beta最近一次构建后的版本号（也就是上面的20161014371145）构建FH\_Online。至此完成代码增量上线。

另外：  
上面在构建工程项目时要求输入的参数VERSION和FILE\_LIST都是在项目配置里定义的（具体下面会提到），并且这两个参数在对应项目配置里的shell脚本里都会被引用。

同时，针对FH\_Online项目，还可以创建FH\_Rollback（使用ROLLBACK\_VERSION参数），即Online环境的回滚方案（当然Test和Beta环境也可以创建回滚项目。一般会写一个回滚脚本，同步前都会做一次备份，回滚脚本可依据自己的情况而定义）。

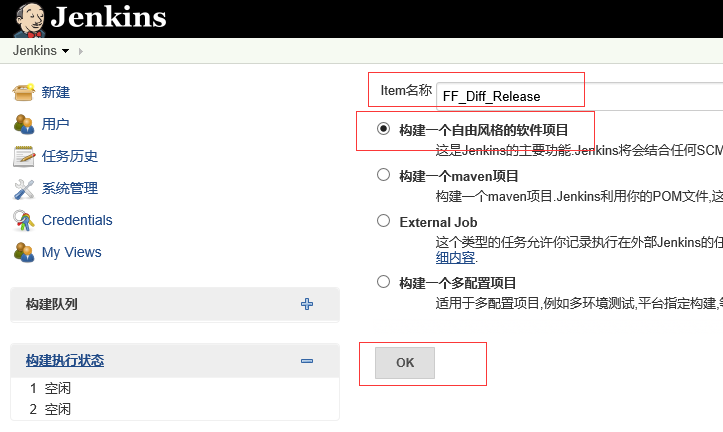
-------------------------------------------------------------------------------  
废话不多说，下面记录下jenkins创建工程，构建项目发版的具体流程：

首先新建一个工程任务:



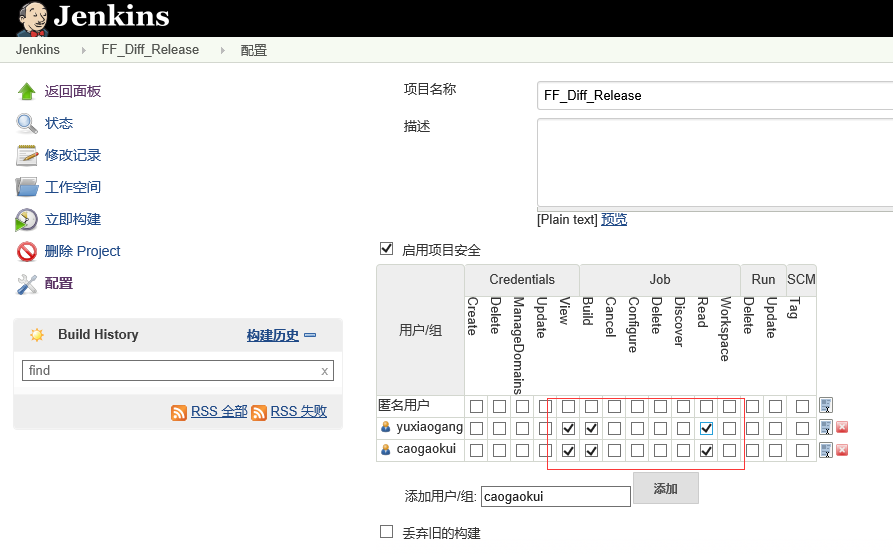
按照上面总结的上线发版流程，这里举出一个业务线的代码发版的项目，依次列出全量、增量发版的工程：  
FF\_Diff\_Release  
FF\_Beta  
FF\_Diff\_Beta  
FF\_Online  
FF\_Rollback

1）FF\_Diff\_Release项目配置

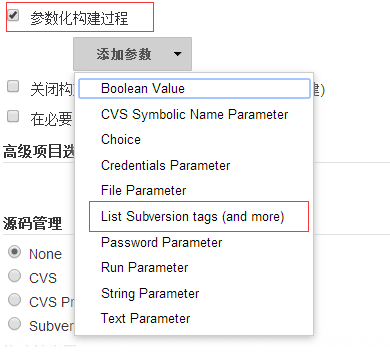


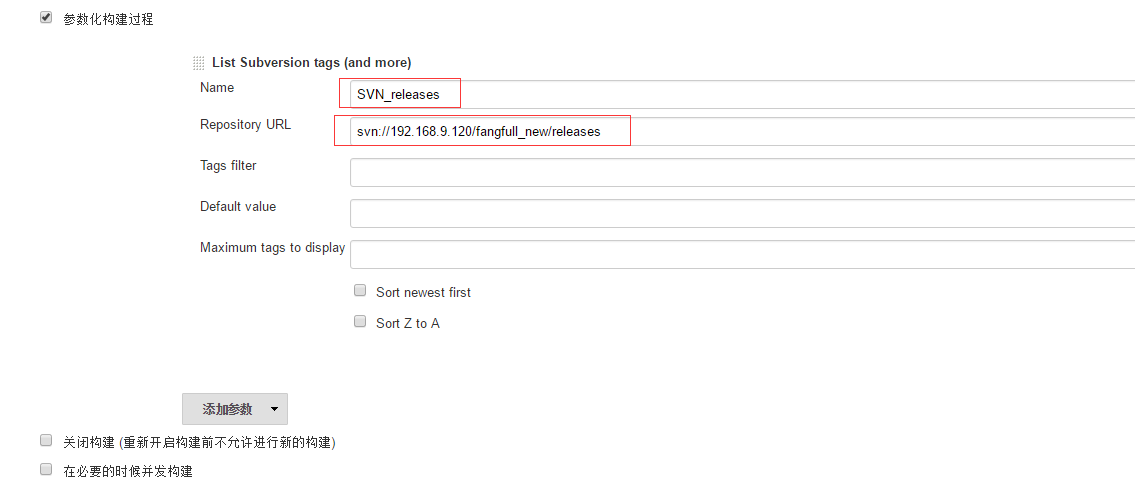
接下来对这个项目进行配置

首选”启用项目安全“，选择对该项目具有操作权限的用户。一般而言，除了运维人员（管理员）具有所有项目操作的权限外，一般只给开发人员（通常是项目组的负责人）Beta环境下的项目操作权限。如下，只给caogaokui和yuxiaogang这两个用户（提前在“管理用户”里创建好的用户）设置下面三个权限，说明他们对该项目由构建的权限。

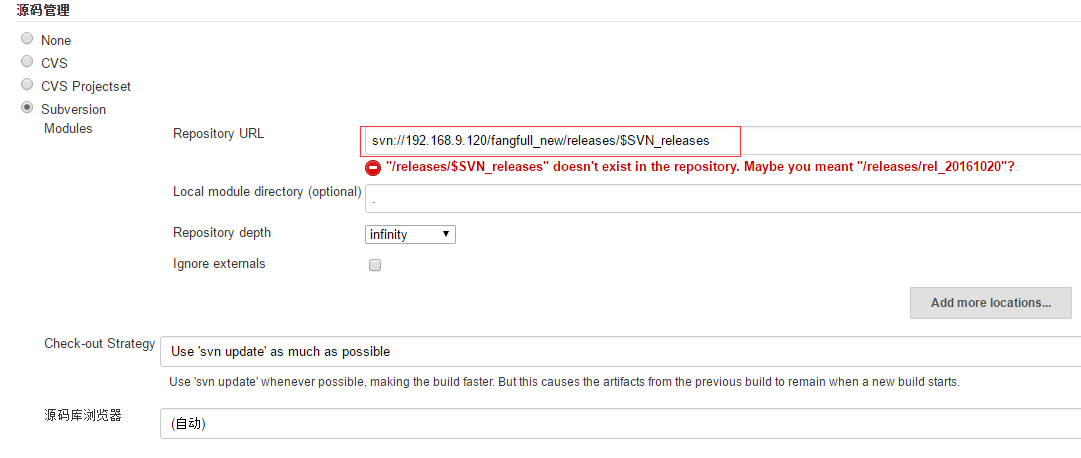


在“参数化构建过程”中选择“List Subversion tags（and more）”,填写相应业务svn的url；这里在对应项目的svn(如下面的svn://192.168.9.120/fangfull\_new)下特意创建了一个release目录，在该目录下建立易于区分的版本号目录（如rel\_20161031，里面存放业务代码）用于jenkins发布版本。这样在构建此工程的时候，就会出现提示选择一个release版本号（下面会演示到）；  
下面填写的svn://192.168.9.120/fangfull\_new/releases最好提前在jenkins本机用svn co测试执行一下(用对应的的svn账号和密码，执行一会儿就ctrl+c结束就行)，看看能否正常通过svn下载，不然可能会有报错。  如下：即将svn://192.168.9.120/fangfull\_new/releases下的版本目录作为参数SVN\_releases，用于构建工程时的提示

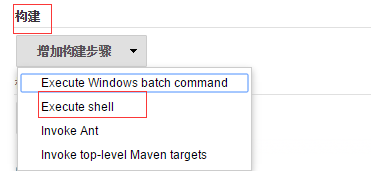


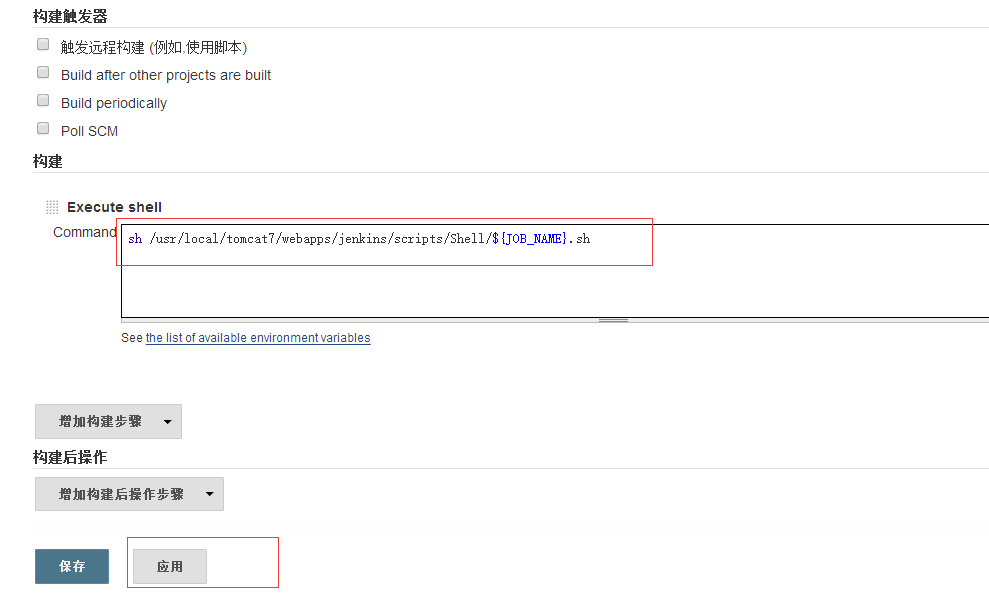


下面的提示不用在意，不影响。



最后，调用shell同步的脚本（脚本在jenkins本机上定义的，这里填写脚本的执行命令就行）





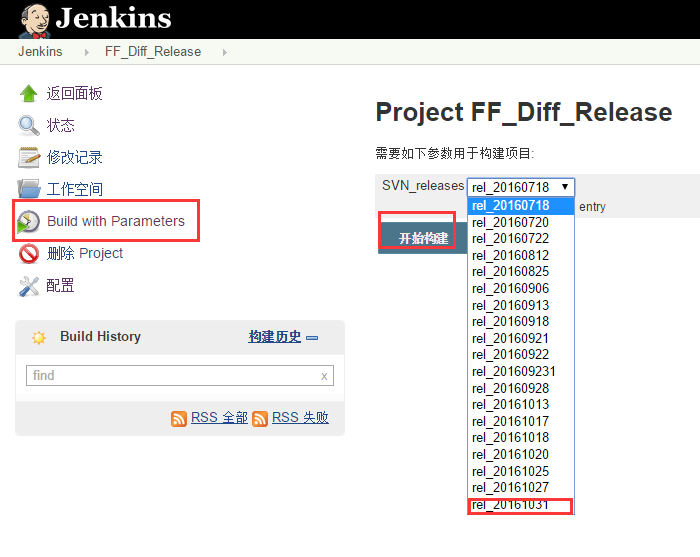
-----------------------------------------------------------------------------------------------------------  
这里列举下jenkins内置的环境变量（可以在jenkins构建的项目配置中直接引用，如上图Shell脚本中引用的${JOB\_NAME}）  
BUILD\_NUMBER， 唯一标识一次build，例如23；  
BUILD\_ID，基本上等同于BUILD\_NUMBER，但是是字符串，例如2011-11-15\_16-06-21；  
JOB\_NAME， job的名字，例如JavaHelloWorld；  
BUILD\_TAG， 作用同BUILD\_ID,BUILD\_NUMBER,用来全局地唯一标识一此build，例如jenkins-JavaHelloWorld-23；  
EXECUTOR\_NUMBER， 例如0；  
NODE\_NAME，slave的名字，例如MyServer01；  
NODE\_LABELS，slave的label，标识slave的用处，例如JavaHelloWorld MyServer01；  
JAVA\_HOME， java的home目录  
WORKSPACE，job的当前工作目录  
HUDSON\_URL = JENKINS\_URL， jenkins的url，例如http://localhost:8000/ ；  
BUILD\_URL，build的url 例如http://localhost:8000/job/JavaHelloWorld/23/；  
JOB\_URL， job的url，例如http://localhost:8000/job/JavaHelloWorld/；  
SVN\_REVISION，svn 的revison  
-----------------------------------------------------------------------------------------------------------

下面贴出该项目配置中调用的shell脚本（脚本中的192.168.1.106是其中的一台线上服务器）：  
先从线上机器拷贝一份代码下来，放到脚本中的${CODE\_DIR}目录下  
接着将代码从svn下载到脚本中的${WORKSPACE}中，然后再同步到脚本中的${FILE\_DIR}/${VERSION}目录下，已等待后续发往beta环境中。

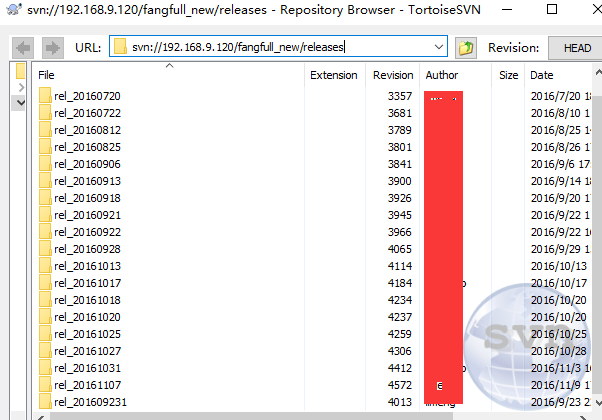
|  |  |
| --- | --- |
| 1  2 | [root@jenkins-server Shell]# pwd  /usr/local/tomcat7/webapps/jenkins/scripts/Shell |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52 | [root@redmine Shell]# cat FF\_Diff\_Release.sh  **#!/bin/bash**  CODE\_DIR="/var/lib/jenkins/code\_online/www.wangshibo.com"  FILE\_DIR="/var/lib/jenkins/file\_list/www.wangshibo.com"  ONLINE\_SERVER="192.168.1.106"  EXCLUDE\_FILE="/var/lib/jenkins/code\_online/www.wangshibo.com.exclude"  VERSION=`/bin/date +%Y%m%d%H%M%S`  LOCK\_FILE="/var/lib/jenkins/file\_list/www.wangshibo.com/diff.lock"    check\_return()  {          if [[ $? != "0" ]]; then                  /bin/echo $1                  exit 255          fi  }    ### check lock file ###  while [ -f ${LOCK\_FILE} ]; do          sleep 2      /bin/echo "Other diff project is running,please wait for a moment,thanks."  done    ### update code ###  /usr/bin/rsync -a -e ssh --exclude-from ${EXCLUDE\_FILE} root@${ONLINE\_SERVER}:/var/www/vhosts/www.wangshibo.com/ ${CODE\_DIR}/  check\_return "update code failed"    ### rsync svn code to version directory ###  /bin/mkdir -p ${FILE\_DIR}/${VERSION}  /usr/bin/rsync -a --exclude-from ${EXCLUDE\_FILE} ${WORKSPACE}/ ${FILE\_DIR}/${VERSION}/    ### diff file ###  /bin/find ${FILE\_DIR}/${VERSION}/ -type f -regex ".\*/.\*\.\(php\|tpl\|js\|css\|html\)" | while read line; do      ONLINE\_FILE=`/bin/echo ${line} | /bin/cut -d"/" -f8- | /bin/sed 's/^/\/var\/lib\/jenkins\/code\_online\/www.wangshibo.com\//g'`      FILE\_NAME=`/bin/echo ${line} | /bin/cut -d"/" -f8-`      if [[ -f ${ONLINE\_FILE} ]]; then          if [[ "$(/usr/bin/md5sum "${line}" | /bin/awk '{print $1}')" != "$(/usr/bin/md5sum "${ONLINE\_FILE}" | /bin/awk '{print $1}')" ]]; then              /bin/echo "=============${FILE\_NAME}============="              /usr/bin/diff -bu ${ONLINE\_FILE} ${line}          fi      else          /bin/echo "=============${FILE\_NAME}============="          /bin/echo "...这是一个新文件..."      fi  done    /bin/echo "==================================Diff Completed=================================="  /bin/echo ""  /bin/echo ""  /bin/echo ""  /bin/echo "Version : ${VERSION}"  /bin/rm -f ${LOCK\_FILE} |

然后执行这个工程的构建，如下，构建时会提示你选择一个SVN\_release的版本号。



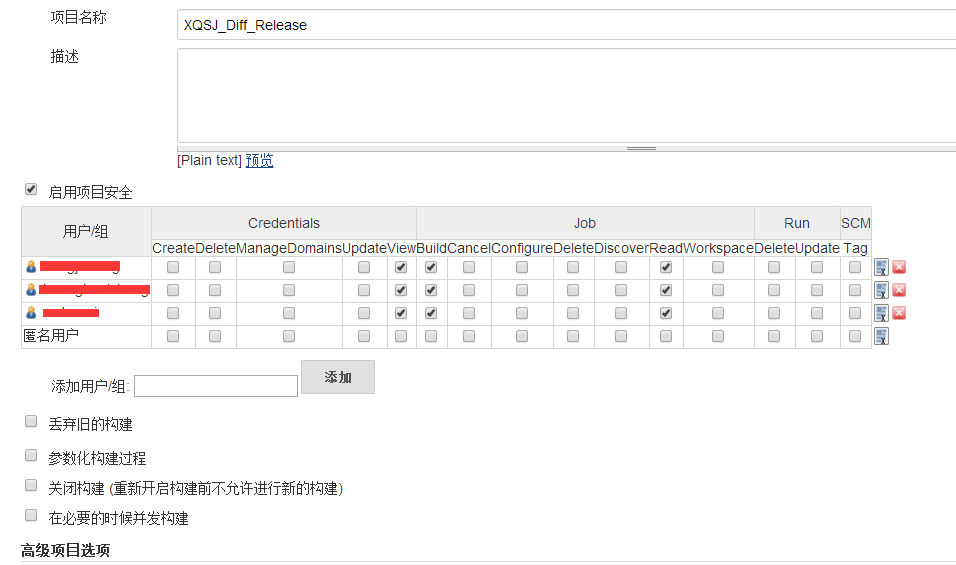
上面的rel\_\*版本目录作为SVN\_releases参数，是开发人员上传到svn://192.168.9.120/fangfull\_new/releases下的，通过svn能看到：

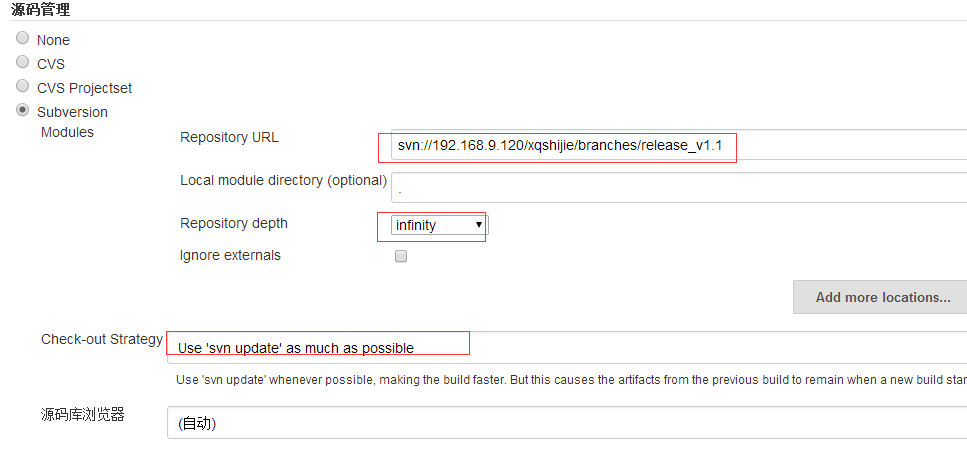


----------------------------------------------------------------------------------------------------------------------------------------  
另一种形式：  
上面的FF\_Diff\_Release项目配置中采用了“发版时选择release版本号”的配置（如上截图中）

也可以不采用上面这种方式：  
直接配置一个svn的url（分支或主干）就行，不启动上面的“参数化构建过程”  
这样，点击“构建”的时候就直接执行项目构建了，没有上面的选择版本号的提示了

如下的一个项目，就跟上面的FF\_Diff\_Release项目配置不一样，具体采用哪种方式，根据自己的情况而定。



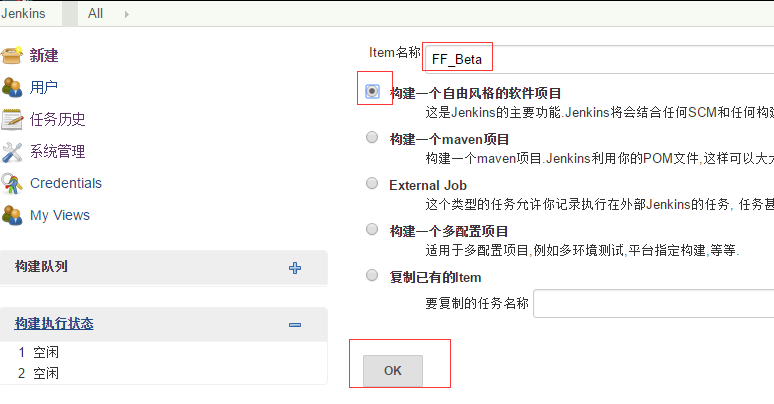


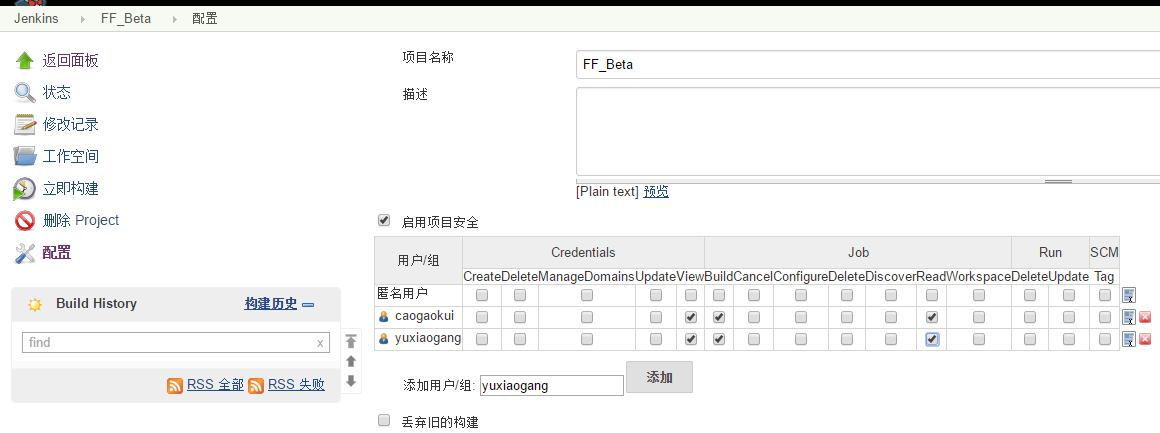


最简单的做法还可以是：在jenkins本机提前编写好代码同步的脚本（svn操作在脚本中定义），然后在jenkins界面里的项目构建配置中只引用shell脚本构建即可。

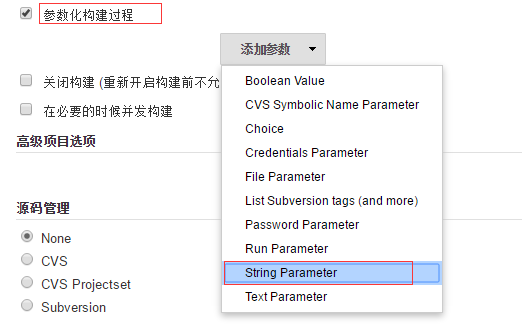
----------------------------------------------------------------------------------------------------------------------------------------

2）FF\_Beta项目配置





启用“参数化构建过程”，选择“String Parameter”，里面填写“VERSION”。这样构建这个工程时就会出现提示你输入版本号，也就是输入上面的FF\_Diff\_Release项目构建后打印的版本号（在Consloe Output日志里最底部可以看到）构建。





接下来不需要再设置“源码管理”了，直接引用shell脚本即可。



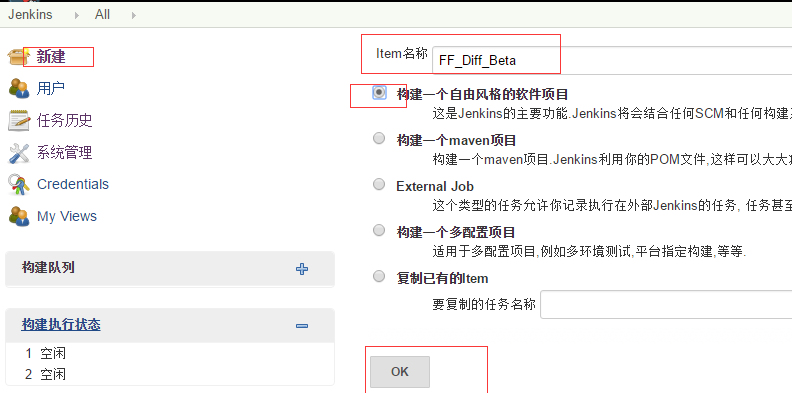
 下面贴出该项目配置中调用的shell脚本（脚本中的192.168.1.25是beta服务器），可以看出，上面设置的VERSION参数，在脚本里已经被引用到了。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | [root@jenkins-server Shell]# cat FF\_Beta.sh  #!/bin/bash  FILE\_DIR="/var/lib/jenkins/file\_list/www.wangshibo.com"  BETA\_SERVER="192.168.1.25"  BETA\_DIR="/var/www/vhosts/www.wangshibo.com"    ### define check\_return ###  check\_return()  {          if [[ $? != "0" ]]; then                  /bin/echo $1                  exit 255          fi  }    ### check syntax errors ###  /bin/echo "check syntax errors"  for line in `/bin/find ${FILE\_DIR}/${VERSION} -type f -name "\*.php"`; do      /Data/app/php5.5.1/bin/php -l "${line}" >/dev/null 2>&1      check\_return "syntax errors,please check ${line}"  done    ### release to beta server ###  /usr/bin/rsync -av -e "ssh -p25791"  ${FILE\_DIR}/${VERSION}/ root@${BETA\_SERVER}:${BETA\_DIR}/  check\_return "release to beta server failed"    ### change directory permission ###  /usr/bin/ssh -p25791 -t -q -o 'StrictHostKeyChecking no' root@${BETA\_SERVER} "/bin/chown -R nobody:nobody /var/www/vhosts/www.wangshibo.com; /bin/chmod -R 755 /var/www/vhosts/www.wangshibo.com/main/protected/runtime /var/www/vhosts/www.wangshibo.com/main/protected/themes /Data/logs/www/fangfull.com"  ### purge memcache ###  #/usr/bin/ssh -t -q -o 'StrictHostKeyChecking no' root@${BETA\_SERVER} "echo flush\_all | nc 127.0.0.1 11221 ; echo flush\_all | nc 127.0.0.1 11222" >/dev/null 2>&1    ### print version ###  /bin/echo "Version : ${VERSION}" |

点击构建脚本，就会出现输入版本号的提示（注意上面Beta的shell脚本中有/Data/app/php5.5.1/bin/php进程，把这个php服务启动起来；按照以上脚本中的意思是在代码发布前，将代码中的php文件搜索出来，然后用php执行下这些php脚本文件，如果有报错，则终止这次构建，如果没有报错，则继续进行代码发布；具体按照自己的实际情况对脚本进行调试）



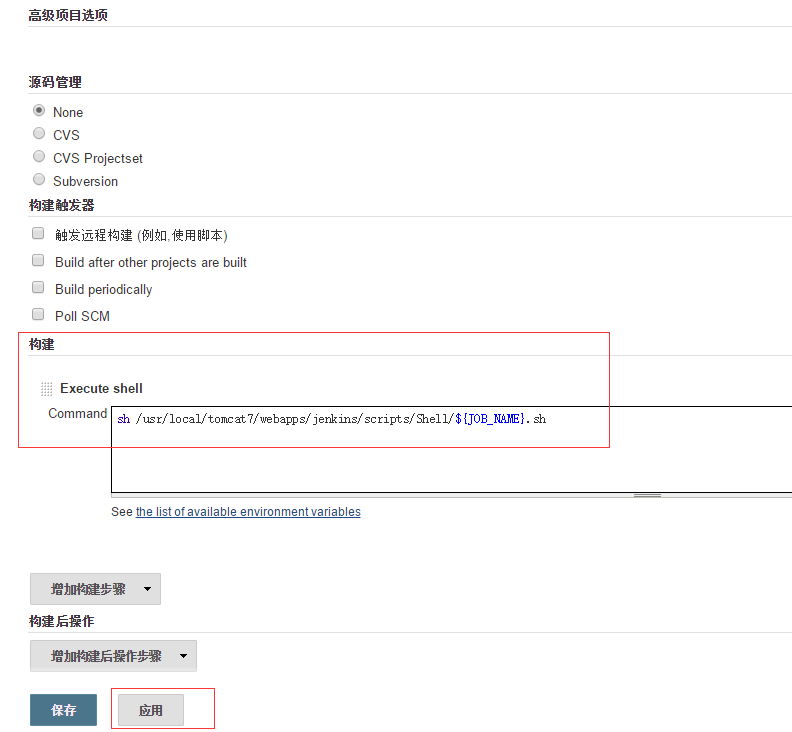
3）FF\_Diff\_Beta项目配置



 如下：参数名填写为FILE\_LIST,在构建该项目的时候会有这个参数的提示，即输入要发布的文件，这是增量发版用到的（具体上面已经总结）。



接着直接引用shell脚本即可：



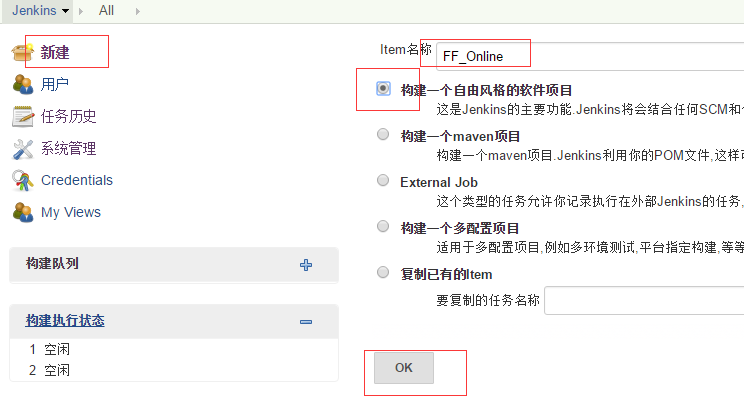
下面贴出该项目配置中调用的shell脚本（脚本中的192.168.1.25是beta服务器）：  
从脚本中可以看出：  
上面“参数化构建”中定义的FILE\_LIST在脚本中被引用到了。  
同步前先从beta机器上将代码备份一份到本机，然后再根据FILE\_LIST进行增量同步。

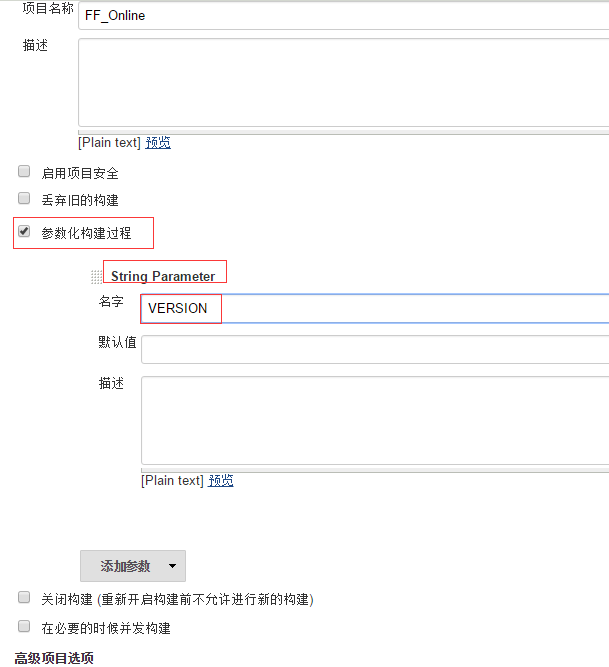
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59 | [root@jenkins-server Shell]# cat FF\_Diff\_Beta.sh  **#!/bin/bash**  CODE\_DIR="/var/lib/jenkins/code\_online/www.wangshibo.com"  FILE\_DIR="/var/lib/jenkins/file\_list/www.wangshibo.com"  BETA\_SERVER="192.168.1.25"  BETA\_DIR="/var/www/vhosts/www.wangshibo.com"  ONLINE\_SERVER="192.168.1.25"  EXCLUDE\_FILE="/var/lib/jenkins/code\_online/www.wangshibo.com.exclude"  VERSION=`/bin/date +%Y%m%d%H%M%S`  LOCK\_FILE="/var/lib/jenkins/file\_list/www.wangshibo.com/diff.lock"    check\_return()  {          if [[ $? != "0" ]]; then                  /bin/echo $1                  exit 255          fi  }    ### check lock file ###  while [ -f ${LOCK\_FILE} ]; do          sleep 2      /bin/echo "Other diff project is running,please wait for a moment,thanks."  done    ### update code ###  /usr/bin/rsync -av -e "ssh -p 25791" --exclude-from ${EXCLUDE\_FILE} root@${ONLINE\_SERVER}:/var/www/vhosts/www.wangshibo.com/ ${CODE\_DIR}/  check\_return "update code failed"    ### rsync file from beta server ###  /bin/mkdir -p ${FILE\_DIR}/${VERSION}  for FILE in ${FILE\_LIST}; do      DIR\_NAME=`/usr/bin/dirname ${FILE}`      /bin/mkdir -p ${FILE\_DIR}/${VERSION}/${DIR\_NAME}      /usr/bin/rsync -av -e "ssh -p 25791" --exclude-from ${EXCLUDE\_FILE} root@${BETA\_SERVER}:${BETA\_DIR}/${FILE} ${FILE\_DIR}/${VERSION}/${FILE}      check\_return "${FILE} 文件不存在!"  done    ### diff file ###  /bin/find ${FILE\_DIR}/${VERSION}/ -type f -regex ".\*/.\*\.\(php\|tpl\|js\|css\|html\)" | while read line; do      ONLINE\_FILE=`/bin/echo ${line} | /bin/cut -d"/" -f8- | /bin/sed 's/^/\/var\/lib\/jenkins\/code\_online\/www.wangshibo.com\//g'`      FILE\_NAME=`/bin/echo ${line} | /bin/cut -d"/" -f8-`      if [[ -f ${ONLINE\_FILE} ]]; then          if [[ "$(/usr/bin/md5sum "${line}" | /bin/awk '{print $1}')" != "$(/usr/bin/md5sum "${ONLINE\_FILE}" | /bin/awk '{print $1}')" ]]; then              /bin/echo "=============${FILE\_NAME}============="              /usr/bin/diff -bu ${ONLINE\_FILE} ${line}          fi      else          /bin/echo "=============${FILE\_NAME}============="          /bin/echo "...这是一个新文件..."      fi  done    /bin/echo "==================================Diff Completed=================================="  /bin/echo ""  /bin/echo ""  /bin/echo ""  /bin/echo "Version : ${VERSION}"  /bin/rm -f ${LOCK\_FILE} |

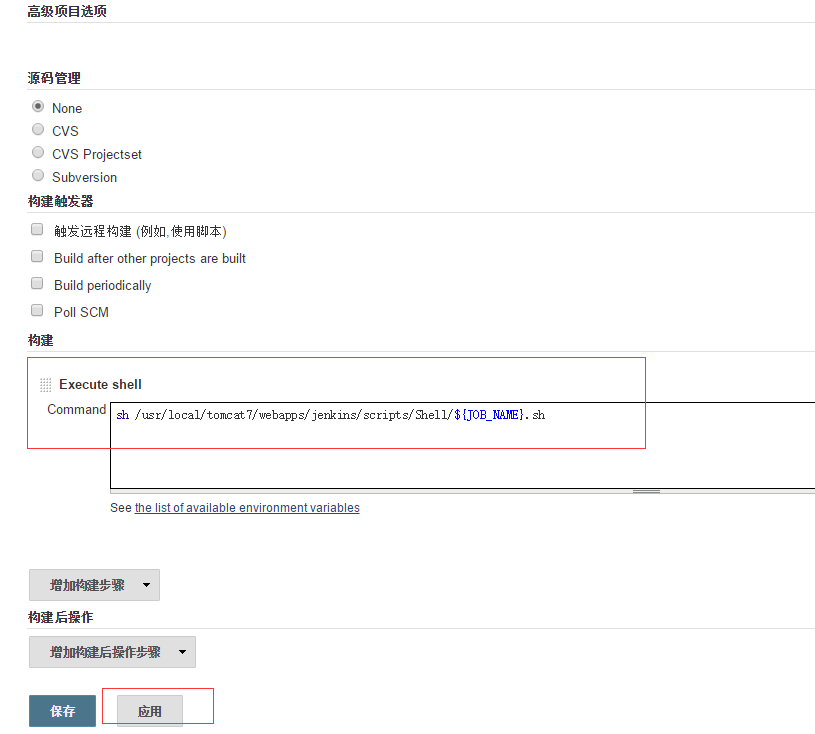
点击构建脚本，就会出现输入文件的提示（多个文件之间用空格隔开），如下，输入文件的格式路径可以根据shell脚本中（注意上面脚本中的dirname命令，即取一个路径的目录部分(即去掉一个路径最后一级后的部分路径)）



4）FF\_Online项目配置







下面贴出该项目构建配置中引用的脚本（脚本中的192.168.1.106和192.168.1.220是线上服务器）：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46 | [root@jenkins-server Shell]# cat FF\_Online.sh  #!/bin/bash  FILE\_DIR="/var/lib/jenkins/file\_list/www.wangshibo.com"  CODE\_DIR="/var/lib/jenkins/code\_online/www.wangshibo.com"  EXCLUDE\_FILE="/var/lib/jenkins/code\_online/www.wangshibo.com.exclude"  ROLLBACK\_DIR="/var/lib/jenkins/rollback/www.wangshibo.com"  ONLINE\_SERVER="192.168.1.106 192.168.1.220"  REMOTE\_SERVER="192.168.1.106"  ONLINE\_DIR="/var/www/vhosts/www.wangshibo.com"    ### define check\_return ###  check\_return()  {          if [[ $? != "0" ]]; then                  /bin/echo $1                  exit 255          fi  }    ### check version ###  if [[ ! -d ${FILE\_DIR}/${VERSION} ]]; then          /bin/echo "版本无效,请确认."          exit 255  fi      ### update code ###  /bin/echo "update online code"  /usr/bin/rsync -a -e ssh --exclude-from ${EXCLUDE\_FILE} root@${REMOTE\_SERVER}:/var/www/vhosts/www.wangshibo.com/ ${CODE\_DIR}/  check\_return "update code failed"    ### backup code ###  /bin/echo "backup online code"  cd ${CODE\_DIR} && /usr/bin/rsync -aR \* ${ROLLBACK\_DIR}/${VERSION}/  check\_return "backup code failed"    ### release to online server ###  for SERVER in ${ONLINE\_SERVER}; do      /bin/echo "release to ${SERVER}"      /usr/bin/rsync -av -e "ssh -o 'StrictHostKeyChecking no'" ${FILE\_DIR}/${VERSION}/ root@${SERVER}:${ONLINE\_DIR}/      /usr/bin/ssh -t -q -o 'StrictHostKeyChecking no' root@${SERVER} "/bin/chown -R nobody:nobody /var/www/vhosts/www.wangshibo.com ; /bin/chmod -R 755 /var/www/vhosts/www.wangshibo.com/main/protected/runtime /var/www/vhosts/www.wangshibo.com/main/protected/themes /Data/logs/www/fangfull.com" >/dev/null 2>&1  done      ### print rollback version ###  /bin/echo "Rollback Version : ${VERSION}" |

点击构建脚本，就会出现输入版本号的提示



5）FF\_Rollback项目配置，即FF\_Online的回滚方案，就是说上线如若出现问题，构建这个FF\_Rollback工程就会自动回滚到上线前的状态







下面是该回滚项目的shell脚本

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | [root@jenkins-server Shell]# cat FF\_Rollback.sh  **#!/bin/bash**  FILE\_DIR="/var/lib/jenkins/file\_list/www.wangshibo.com"  CODE\_DIR="/var/lib/jenkins/code\_online/www.wangshibo.com"  EXCLUDE\_FILE="/var/lib/jenkins/code\_online/www.wangshibo.com.exclude"  ROLLBACK\_DIR="/var/lib/jenkins/rollback/www.wangshibo.com"  ROLLBACK\_VERSION=`/bin/ls -l /var/lib/jenkins/rollback/www.wangshibo.com|tail -1|awk '{print $9}'`  ONLINE\_SERVER="192.168.1.106 192.168.1.220"  ONLINE\_DIR="/var/www/vhosts/www.wangshibo.com"      ### define check\_return ###  check\_return()  {          if [[ $? != "0" ]]; then                  /bin/echo $1                  exit 255          fi  }    ### release to online server ###  for SERVER in ${ONLINE\_SERVER}; do      /bin/echo "release to ${SERVER}"      /usr/bin/rsync -av -e "ssh -o 'StrictHostKeyChecking no'" ${ROLLBACK\_DIR}/${ROLLBACK\_VERSION}/ root@${SERVER}:${ONLINE\_DIR}/      /usr/bin/ssh -t -q -o 'StrictHostKeyChecking no' root@${SERVER} "/bin/chown -R nobody:nobody /var/www/vhosts/www.wangshibo.com ; /bin/chmod -R 755 /var/www/vhosts/www.wangshibo.com/main/protected/runtime /var/www/vhosts/www.wangshibo.com/main/protected/themes /Data/logs/www/fangfull.com ; cd /var/www/vhosts/www.wangshibo.com/main/qmjjrwap/themes/ntpl; /bin/rm -rf /var/www/vhosts/www.wangshibo.com/main/qmjjrwap/themes/ntpl/\*; /bin/ln -s /var/www/vhosts/www.wangshibo.com/main/qmjjrwap/dist/themes/ntpl/qmjjrwap ./; /bin/chown -R nobody:nobody /var/www/vhosts/www.wangshibo.com/main/qmjjrwap/themes/ntpl/qmjjrwap" >/dev/null 2>&1  done      ### print rollback version ###  /bin/echo "Rollback Version : ${VERSION}" |

回滚即是用最近一次的备份覆盖到线上，脚本中的“ROLLBACK\_VERSION”参数实际上取的就是最近一次的备份

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | [root@jenkins-server Shell]# ll /var/lib/jenkins/rollback/www.fangfull.com  total 72  drwxr-xr-x 9 root root 4096 Aug 10 12:24 20160810122352  drwxr-xr-x 9 root root 4096 Aug 12 18:05 20160812180522  drwxr-xr-x 9 root root 4096 Aug 19 17:59 20160819175910  drwxr-xr-x 9 root root 4096 Aug 29 10:41 20160826170848  drwxr-xr-x 9 root root 4096 Sep  8 19:52 20160908195214  drwxr-xr-x 9 root root 4096 Sep 14 19:40 20160914193917  drwxr-xr-x 9 root root 4096 Sep 20 18:33 20160920171701  drwxr-xr-x 9 root root 4096 Sep 22 13:41 20160922110959  drwxr-xr-x 9 root root 4096 Sep 22 16:19 20160922161415  drwxr-xr-x 9 root root 4096 Sep 23 20:30 20160923201543  drwxr-xr-x 9 root root 4096 Sep 23 21:57 20160923213311  drwxr-xr-x 9 root root 4096 Sep 23 22:12 20160923220941  drwxr-xr-x 9 root root 4096 Sep 29 21:24 20160929210300  drwxr-xr-x 9 root root 4096 Oct 17 16:13 20161017161120  drwxr-xr-x 9 root root 4096 Oct 17 19:51 20161017192718  drwxr-xr-x 9 root root 4096 Oct 26 17:17 20161025173337  drwxr-xr-x 9 root root 4096 Oct 28 18:58 20161028185753  drwxr-xr-x 9 root root 4096 Nov  2 17:03 20161102145159  [root@jenkins-server Shell]# /bin/ls -l /var/lib/jenkins/rollback/www.fangfull.com|tail -1|awk '{print $9}'  20161102145159 |

点击这个回滚项目的构建，同样会出现让你输入版本号的提示



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

扩展小提示：

截取目录的方法，有以下两种：  
1）dirname命令：用于取给定路径的目录部分。这个命令很少直接在shell命令行中使用，一般把它用在shell脚本中，用于取得脚本文件所在目录，然后将当前目录切换过去。  
其实就是取所给路径的倒数第二级及其之前的路径部分，如下：  
[root@jenkins-server Shell]# dirname main/protected/xqsjmob/themes/tpl2/common/page\_statistics.tpl  
main/protected/xqsjmob/themes/tpl2/common  
[root@jenkins-server Shell]# dirname /usr/local/tomcat7/webapps/jenkins/scripts/Shell  
/usr/local/tomcat7/webapps/jenkins/scripts  
[root@jenkins-server Shell]# dirname FF\_Online.sh            #取的结果既是当前路径目录下

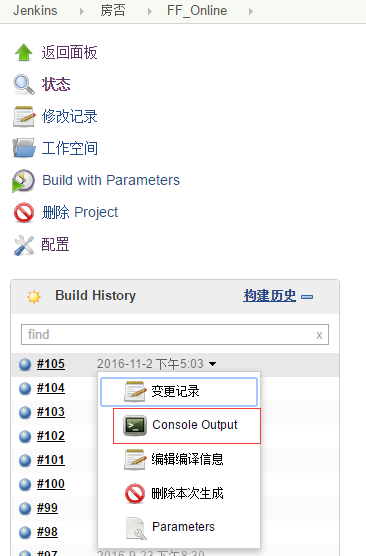
.

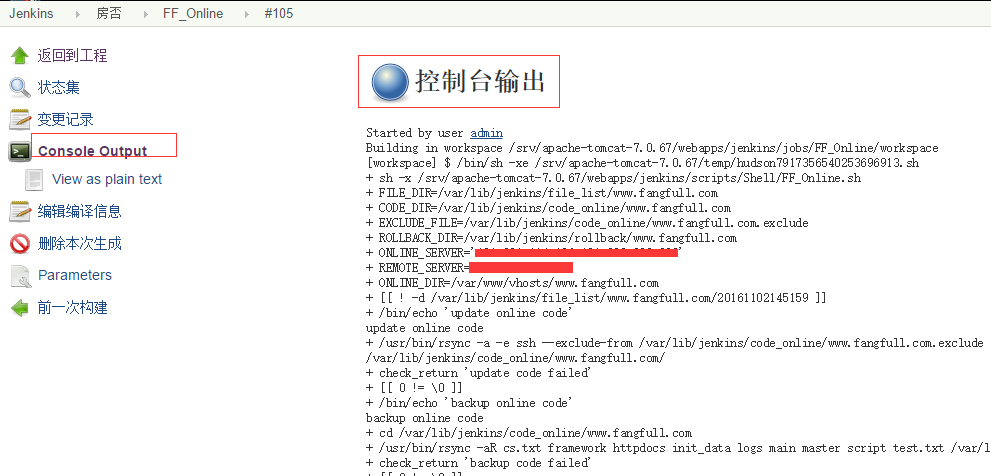
2）可以用${pathname%/\*}截取掉pathname后面的文件部分。  
下面比较下这两种方法的效果：即dirname取的是倒数第二级及其以上级的路径，而${pathname%/\*}取的是"删除最后一个/符号后的路径部分"  
[root@jenkins-server Shell]# pathname=/usr/bin/sort; echo $(dirname $pathname) ${pathname%/\*}   
/usr/bin /usr/bin  
You have new mail in /var/spool/mail/root  
[root@jenkins-server Shell]# pathname=/usr/bin/sort/; echo $(dirname $pathname) ${pathname%/\*}   
/usr/bin /usr/bin/sort  
[root@jenkins-server Shell]# pathname=/usr/; echo $(dirname $pathname) ${pathname%/\*}   
/ /usr  
[root@jenkins-server Shell]# pathname=/usr; echo $(dirname $pathname) ${pathname%/\*}   
/

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

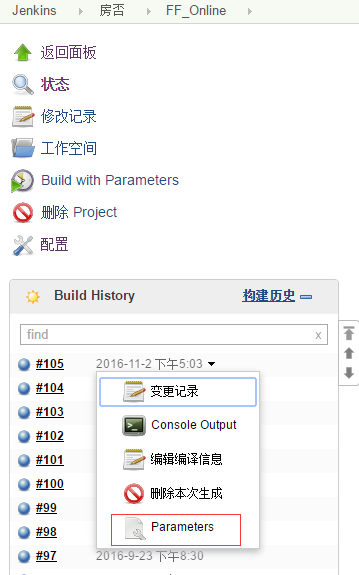
以上的项目在构建后，可以通过构建日志查看代码发布情况：

如下：点击“Console Output"查看日志

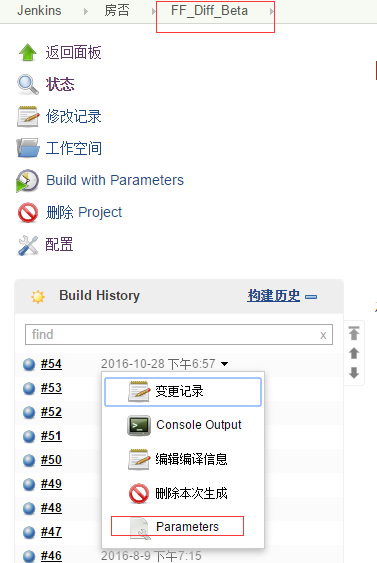




 通过查看Parameter参数得到本次构建动作所用到的VERSION版本号或FILE\_LIST文件

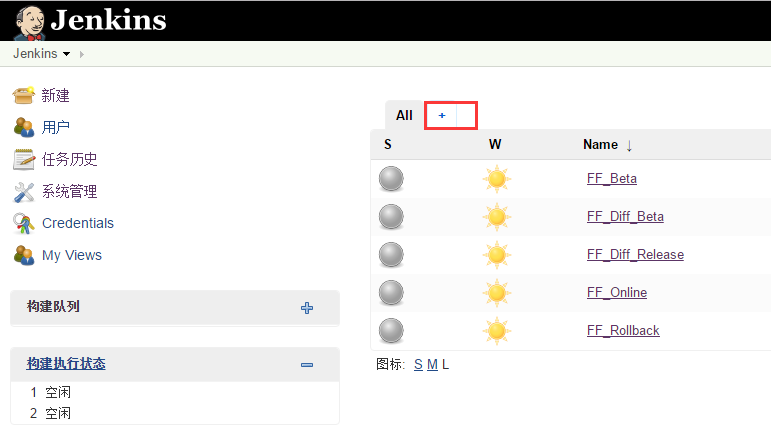






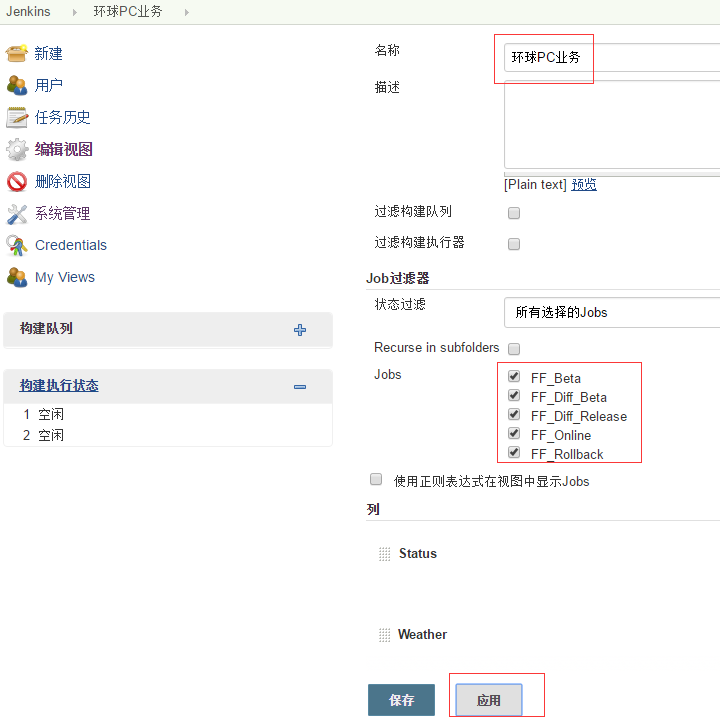


待项目构建完成后，还可以创建视图，将相关业务（应用）构建的工程放到同一个视图下，有多少个应用就创建多少个视图。如下创建视图：





然后将相应的构建工程拉到这个视图下进行管理





至此，一个完整项目的jenkins发版流程已经配置完成，后续上线发版只需要在这里一键构建执行即可。这仅仅是依据本人使用jenkins的经历而梳理的，如有误述，敬请指出，以待改正！