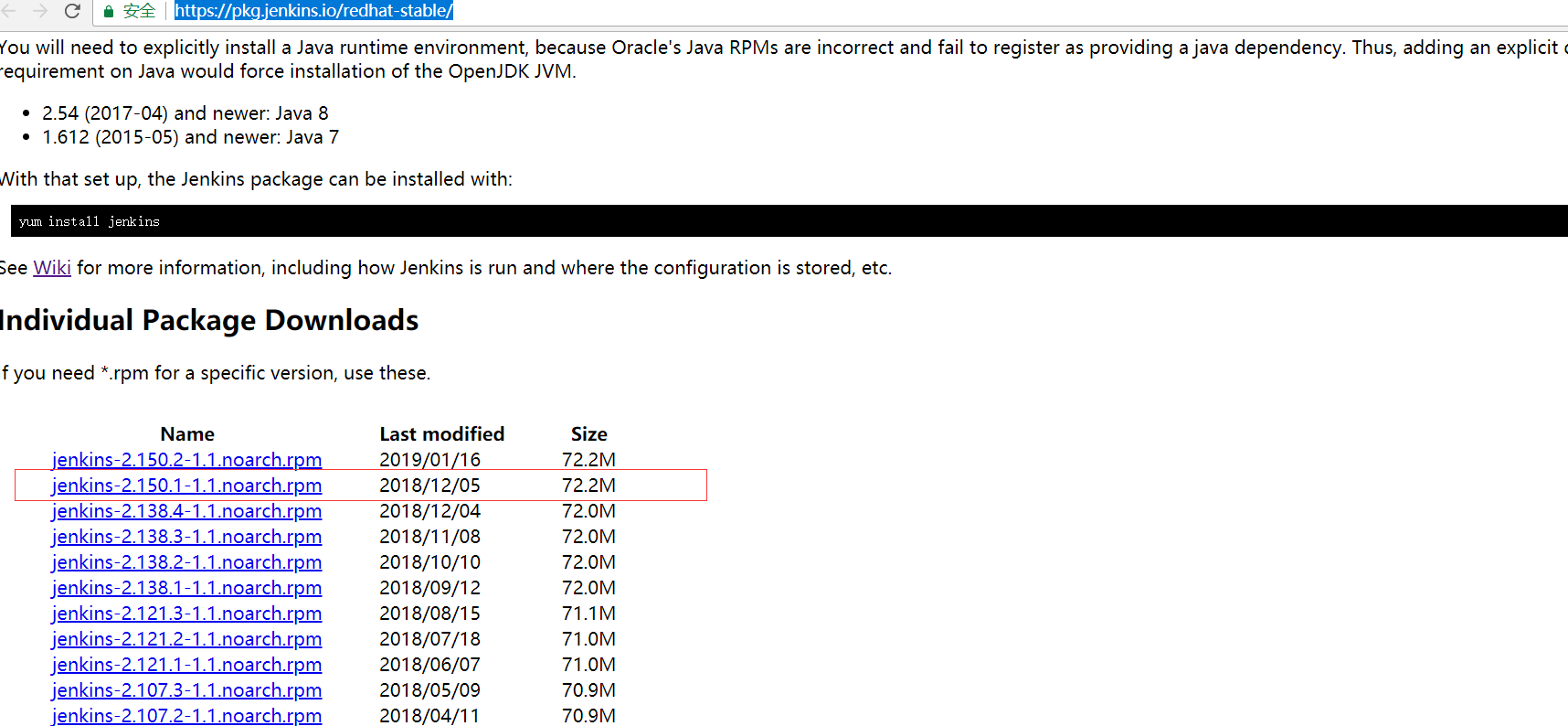
# Jenkins 在centos7.0上的安装

## 下载地址

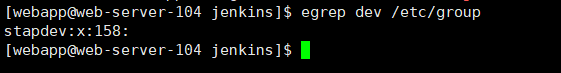
<https://pkg.jenkins.io/redhat-stable/>



## 查看linux中 Jenkins用户及 dev用户组是否存在

1. 查看dev 用户组是否存在

|  |
| --- |
| egrep dev /etc/group |



从查找结果可知,dev 用户组并不存在,stapdev 组并不是我们我们想要的用户组

1. 查看 Jenkins用户是否存在

|  |
| --- |
| egrep jenkins /etc/passwd |



从上面的查询结果可知,jenkins 用户组并不存在

## 在linux中创建jenkins 用户 和 dev 用户组

1. 创建dev用户组

说明:由于当前登录的用户的webapp用户,为了让webapp用户能够创建用户及用户组,这里我们为webapp用户授予 免密的sudo权限

为webapp用户授予免密的sudo权限的第二个原因是,未来我们使用jenkins部署应用时,如果是部署到同一台机器上,这时我们也会使用webapp账户进行应用部署,但是由于当前的webapp账户没有sudo权限,因此不能修改 /etc/sudoers 文件,因此我们需要切换到据用 sudo权限的账户,进行修改 /etc/sudoers 文件,使其webapp账户具用sudo权限

切换账户:

说明:当前系统中 admin账户据用sudo权限,因此我们切换到admin账户下

|  |
| --- |
| su admin |



修改方法如下:

|  |
| --- |
| 给普通用户添加sudo权限，并且使用时不用输入密码：  直接键入命令visudo（这是Linux提供的修改sudoer的工具，与用vi亲自修改配置文件相比，它提示更多的提示信息的纠错能力）， 或者vi /etc/sudoers ,添加如下一行  webapp ALL=(ALL) NOPASSWD: ALL |

创建账户组的命令:

|  |
| --- |
| sudo groupadd dev |



1. 创建账户

格式:

|  |
| --- |
| sudo useradd -s /bin/bash -d 用户的主目录 -g 用户所属的组 -m 用户名 |

具体案例:

|  |
| --- |
| sudo useradd -s /bin/bash -d /home/jenkins -g dev -m jenkins |



1. 授予用户目录权限

格式:

|  |
| --- |
| sudo chown -R 用户:用户组 具体目录 |

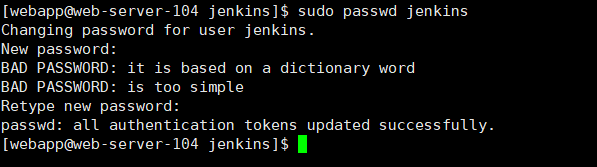
具体案例

|  |
| --- |
| sudo chown -R jenkins:dev /usr/hyxt/webapp/springboot/jenkins |



1. 为新创建的用户设置密码

|  |
| --- |
| sudo passwd jenkins |



## 安装命令

1. 查看jenkins是否已经安装

|  |
| --- |
| rpm –ql jenkins |

1. 安装jenkins的命令

|  |
| --- |
| rpm -ivh jenkins-2.150.1-1.1.noarch.rpm |

## 启动/停止/重启 Jenkins

1. 启动jenkins的命令

|  |
| --- |
| service jenkins start |

1. 停止jenkins的命令

|  |
| --- |
| service jenkins stop |

1. 重启jenkins的命令

|  |
| --- |
| service Jenkins restart |

1. 错误解决

错误: Starting Jenkins bash: /usr/bin/java: No such file or directory

解决: 修改Jenkins启动配置文件，指定java安装路径。

|  |
| --- |
| vim /etc/init.d/jenkins |

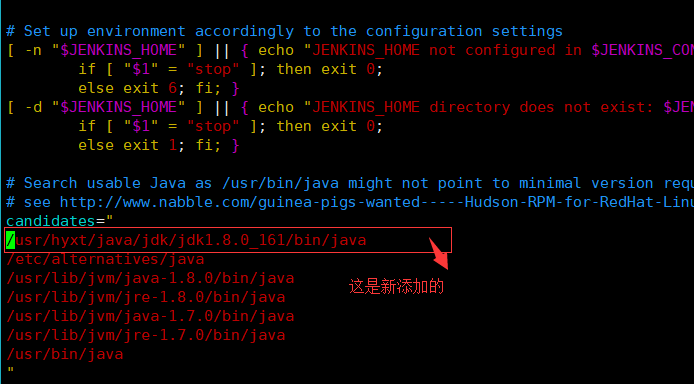
在candidates中第一行添加java路径，如下：

|  |
| --- |
| candidates="  /usr/hyxt/java/jdk/jdk1.8.0\_161/bin/java  /etc/alternatives/java  /usr/lib/jvm/java-1.6.0/bin/java  /usr/lib/jvm/jre-1.6.0/bin/java  /usr/lib/jvm/java-1.7.0/bin/java  /usr/lib/jvm/jre-1.7.0/bin/java  /usr/lib/jvm/java-1.8.0/bin/java  /usr/lib/jvm/jre-1.8.0/bin/java  /usr/bin/java  " |

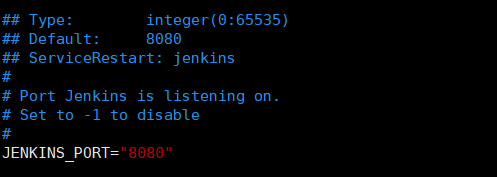
或者直接替换为[推荐]

|  |
| --- |
| candidates="  /usr/hyxt/java/jdk/jdk1.8.0\_161/jre/bin/java  " |

案例如下:



1. 默认情况,在/etc/sysconfig/jenkins 配置中的 端口号是8080



1. 如果该端口号对应的防火墙没有允许,则需要开启运行访问,开启步骤如下

* 修改防火墙允许8080端口访问

|  |
| --- |
| sudo vim /etc/sysconfig/iptables |

* 在该配置文件中添加如下命令

|  |
| --- |
| -A INPUT -m state --state NEW -m tcp -p tcp --dport 8080 -j ACCEPT |

1. 通过 rpm -ql jenkins 命令查看安装成功后的jenkins的相关文件

密令: rpm -ql Jenkins

|  |
| --- |
| /etc/init.d/jenkins  /etc/logrotate.d/jenkins  /etc/sysconfig/jenkins  /usr/lib/jenkins  /usr/lib/jenkins/jenkins.war  /usr/sbin/rcjenkins  /var/cache/jenkins  /var/lib/jenkins  /var/log/jenkins |

解析:

/etc/init.d/jenkins

Jenkins开启启动的配置文件,里面包含了start/restart/stop等参数命令实现

/etc/logrotate.d/jenkins

?

/etc/sysconfig/jenkins

Jenkins相关属性的配置文件

/usr/lib/Jenkins

?

/usr/lib/jenkins/jenkins.war

?

/usr/sbin/rcjenkins

?

/var/cache/jenkins

?

/var/lib/jenkins

Jenkins的工作空间目录

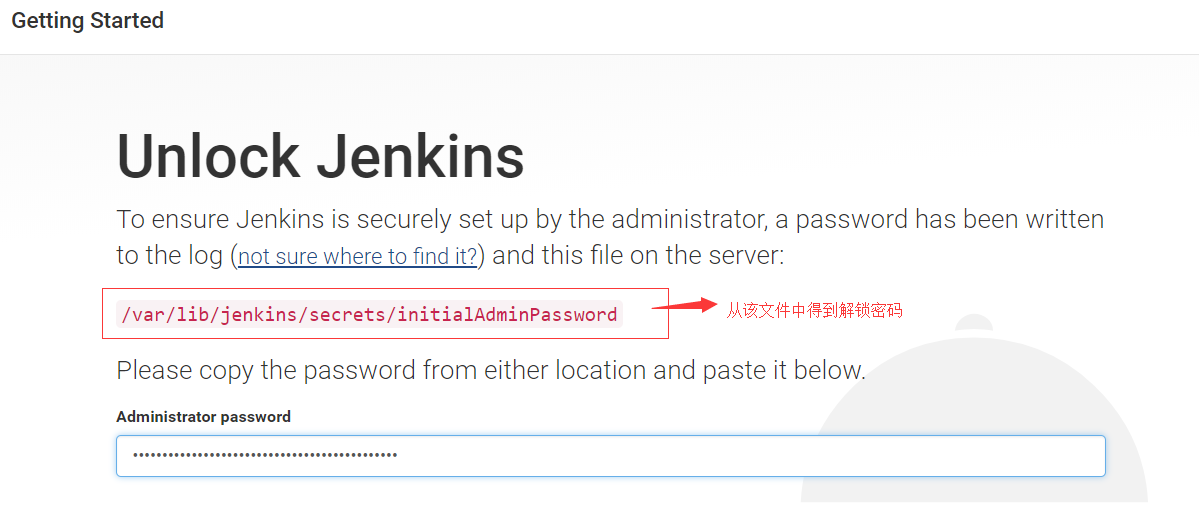
/var/log/Jenkins

Jenkins的日志目录

1. 关于 /etc/sysconfig/Jenkins 配置文件案例,请参考 [ref\_dir/jenkins] 文档

## 访问并解锁jenkins

1. 访问 <http://192.168.0.109:8080>

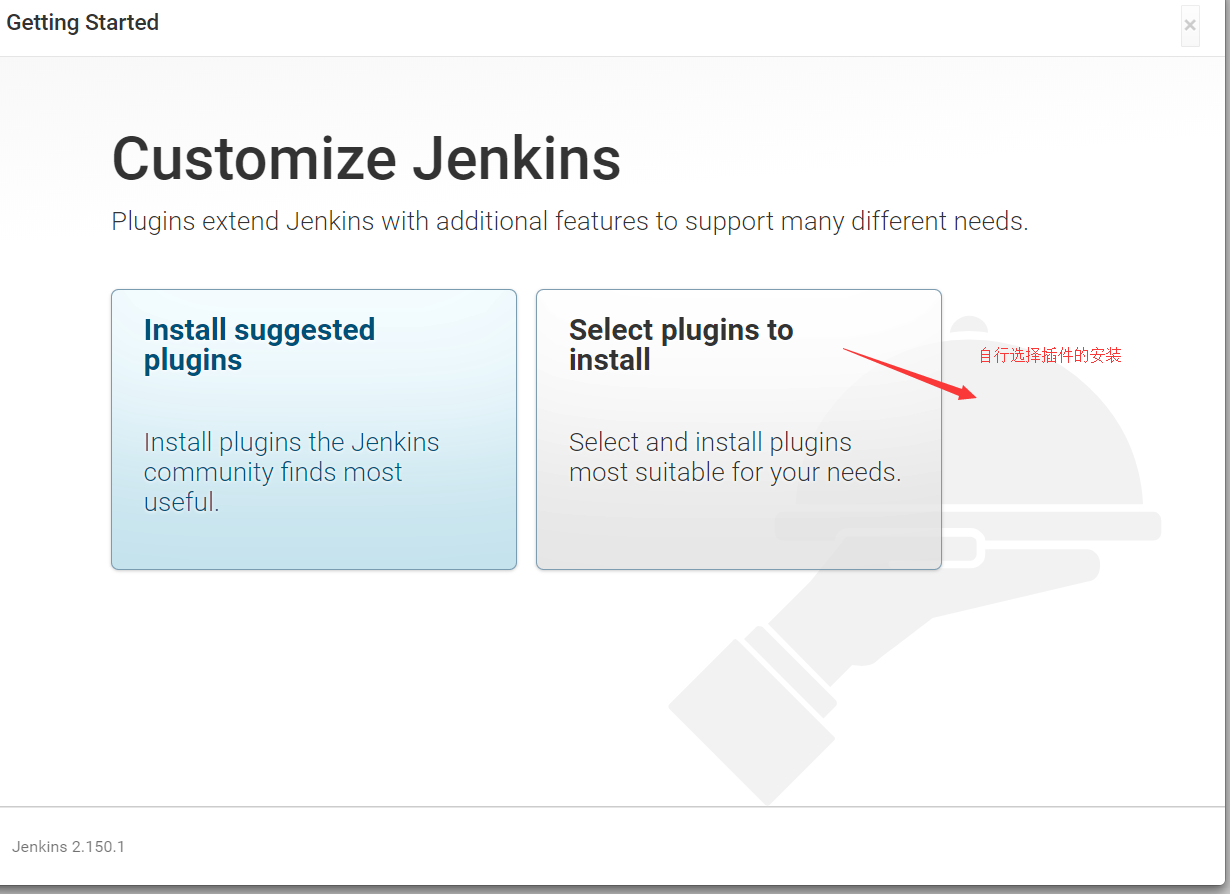


1. 拷贝 /var/lib/jenkins/secrets/initialAdminPassword 得到解锁密码

密令:

|  |
| --- |
| cat /var/lib/jenkins/secrets/initialAdminPassword |

1. 得到密码后,将密码拷贝到密码框,点击下一步,进入如下页面



1. 选择 select plugins to install ,自行选择插件的安装
2. 不选择任何插件,点击安装,进入创建第一个用户的页面
3. 创建一个用户和密码都为 admin 的用户,此时点击 continue as admin ,进行下一步操作
4. 此时使用admin账号登录jenkins时将登录不进系统,需要进行如下操作

* 进入主目录 /var/lib/jenkins目录
* 在jenkins默认的主目录.jenkins中修改config.xml文件，主要是以下几行

|  |
| --- |
| <useSecurity>true</useSecurity>  <authorizationStrategy class="hudson.security.FullControlOnceLoggedInAuthorizationStrategy"/>  <securityRealm class="hudson.security.HudsonPrivateSecurityRealm">  <disableSignup>false</disableSignup>  <enableCaptcha>false</enableCaptcha>  </securityRealm> |

对 authorizationStrategy 标签的class属性值的解释

1、<authorizationStrategy class="hudson.security.AuthorizationStrategy$Unsecured"/>

这个权限对应"任何用户可以做任何事(没有任何限制)"

2、<authorizationStrategy class="hudson.security.FullControlOnceLoggedInAuthorizationStrategy"/>

这个权限对应"登录用户可以做任何事"

3、<authorizationStrategy class="hudson.security.GlobalMatrixAuthorizationStrategy">

<permission>hudson.model.Hudson.Administer:test</permission>

<permission>hudson.scm.SCM.Tag:test</permission>

</authorizationStrategy>

这个权限对应 test用户可以是管理员、打标签权限。

* 此处需要替换为第二种方式

1. 重启jenkins,命令为 sudo service jenkins restart,然后再次进入系统
2. 进入系统后首先要确认admin账号的密码是否为admin,然后按照如下的插件
3. 第一次需要安装的插件

|  |  |
| --- | --- |
| 插件名称 | 插件描述 |
| Build Pipeline Plugin |  |
| Build Timeout |  |
| Dependency Graph Viewer Plugin |  |
| Email Extension Plugin |  |
| GitHub Branch Source |  |
| Gradle |  |
| HTML Publisher plugin |  |
| Last Changes Plugin |  |
| Pipeline |  |
| Pipeline: GitHub Groovy Libraries |  |
| Publish Over FTP |  |
| Publish Over SSH |  |
| SSH Slaves | 非常重要 |
| Timestamper |  |
| Warnings Next Generation |  |
| Workspace Cleanup |  |
| Copy Artifact Plugin | 非常重要 |
|  |  |
|  |  |
|  |  |
|  |  |

1. 插件安装完成后,重启jenkins,重启完成后进行如下操作,为admin账号添加admin权限
2. 点击[系统管理]🡪[全局安全配置]🡪[安全矩阵]🡪[add user or group…],在弹出框中输入 admin ,然后点击确定,在 admin 行勾选 admin栏的复选框,如图下



1. 完成上述操作后,点击应用或者保存,紧接着下面造作
2. 进入 /var/lib/Jenkins 目录,修改 config.xml文件将 <authorizationStrategy > 标签的值修改为

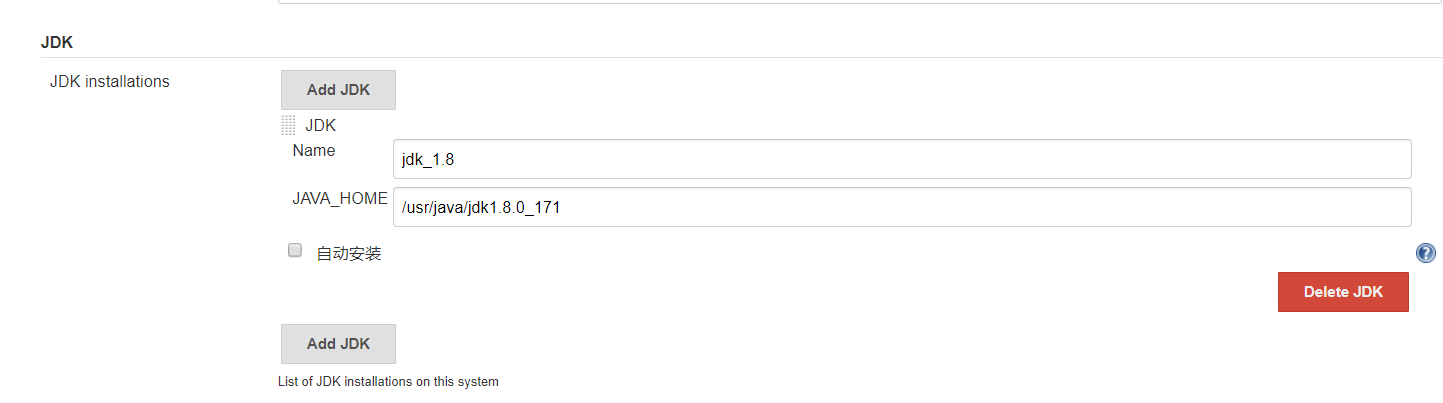
|  |
| --- |
| <authorizationStrategy class="hudson.security.FullControlOnceLoggedInAuthorizationStrategy"/> |

保存后,重启jenkins

1. 重启成功后,刷新浏览器上jenkins页面,进入登录页面重新登录jenkins即可,到此jenkins是安装及初步配置完成

## 全局工具配置

1. jdk配置案例,配置如下,关于jdk的安装请另外查看相关文档



1. maven配置案例,关于maven如何安装,请另外看相关文档



## 系统设置

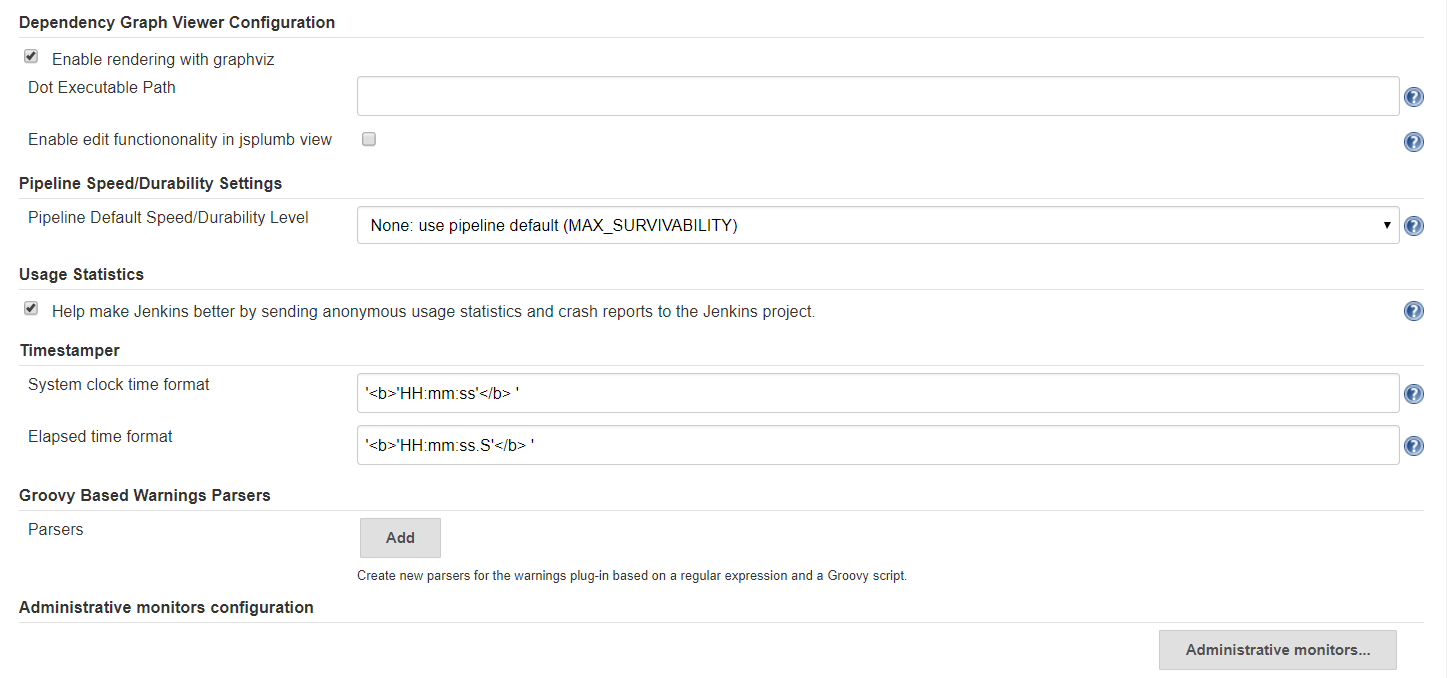
1. 点击 [系统管理]🡪[系统设置],第一部分的配置



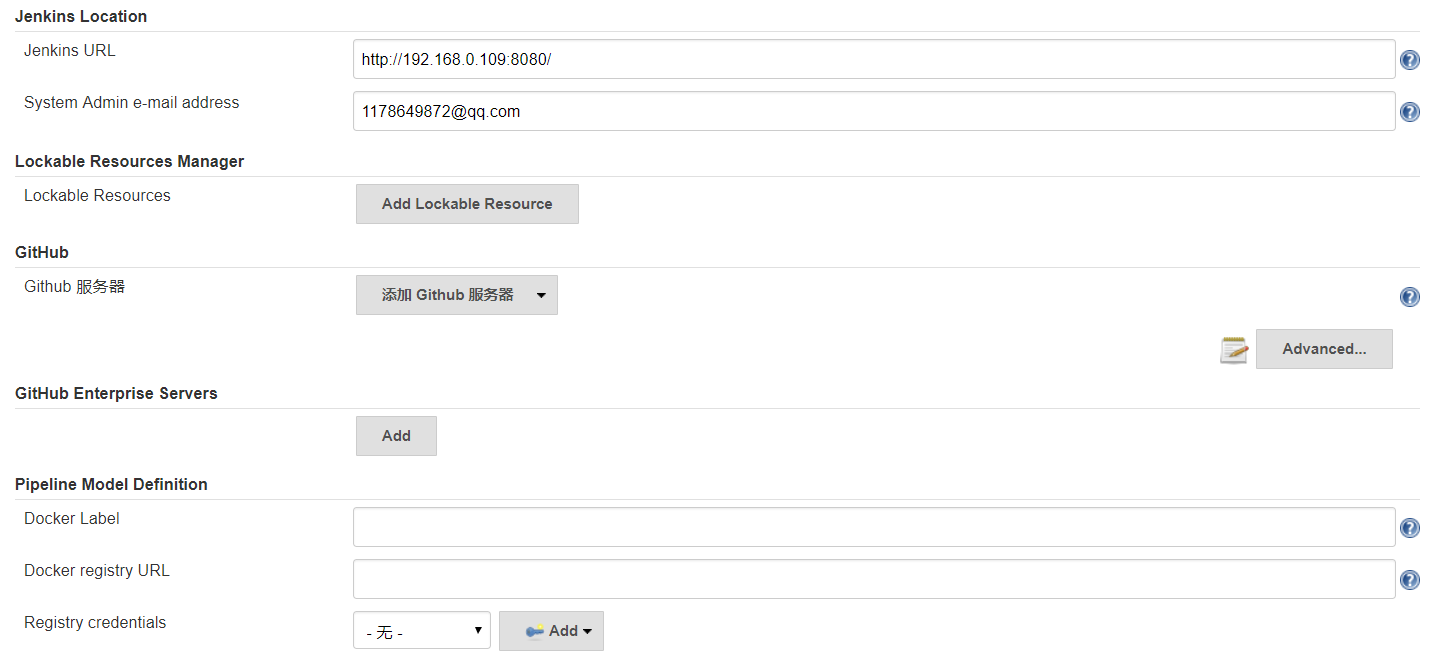
1. Global properties 的配置



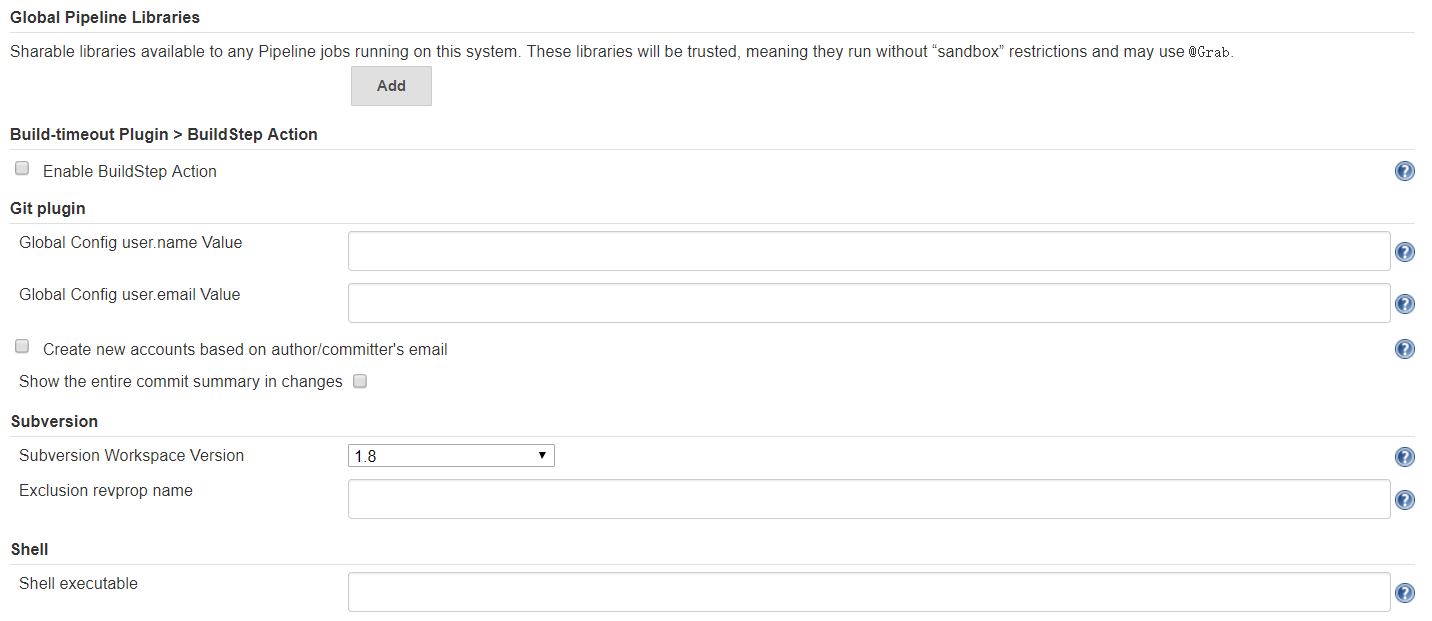
1. Dependency Graph Viewer Configuration和Pipeline Speed/Durability Settings 和Usage Statistics 和 Timestamper 和 Groovy Based Warnings Parsers 和 Administrative monitors configuration 的配置



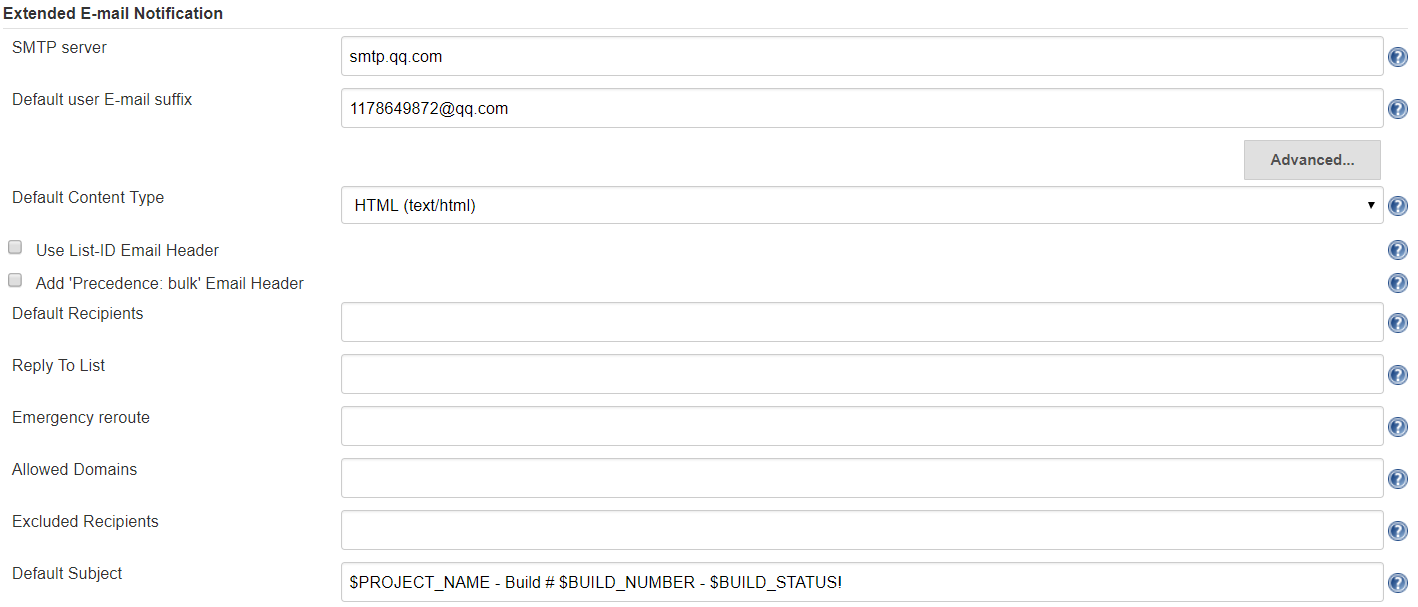
1. Jenkins Location 和 Lockable Resources Manager 和 GitHub 和 GitHub Enterprise Servers 和 Pipeline Model Definition 配置

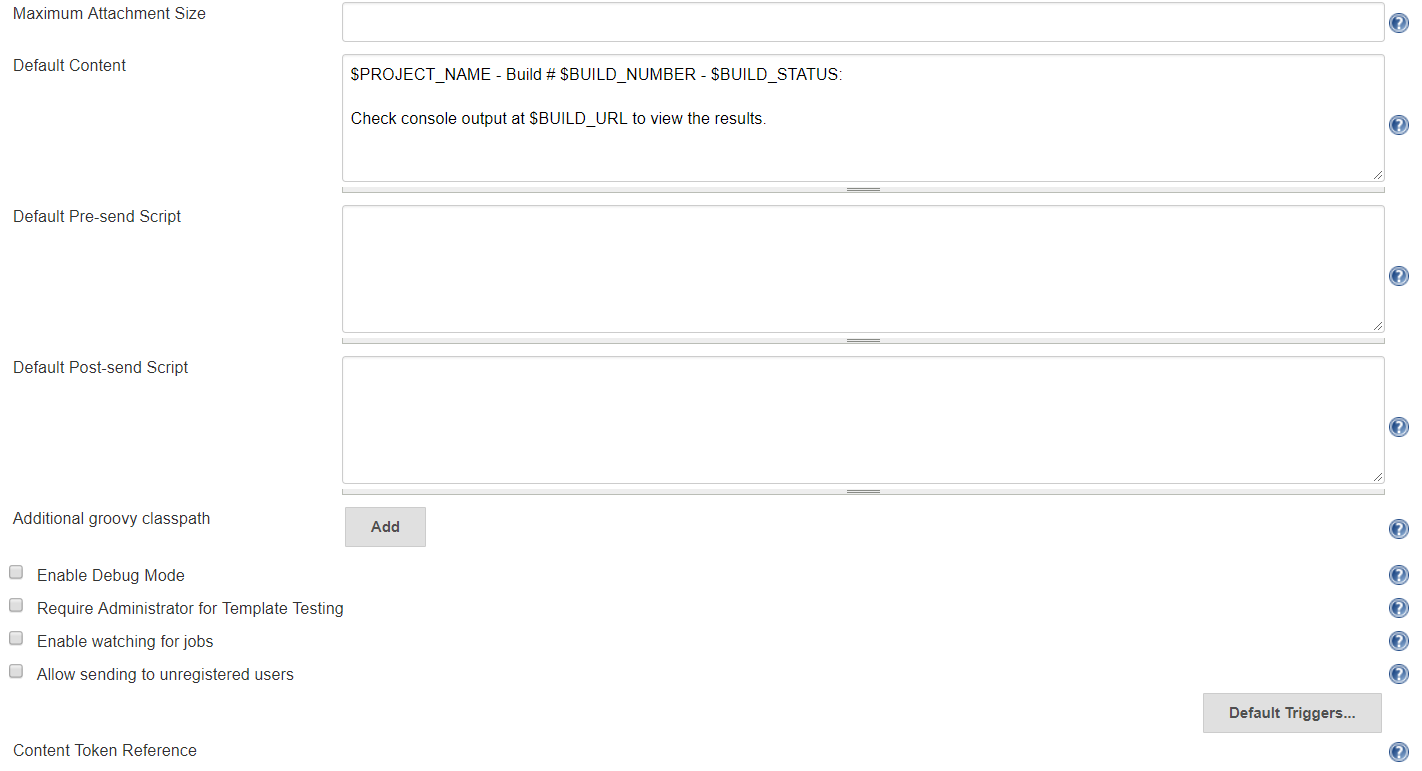


1. Global Pipeline Libraries 和 Build-timeout Plugin > BuildStep Action 和 Git plugin 和 Subversion 和 Shell 的配置

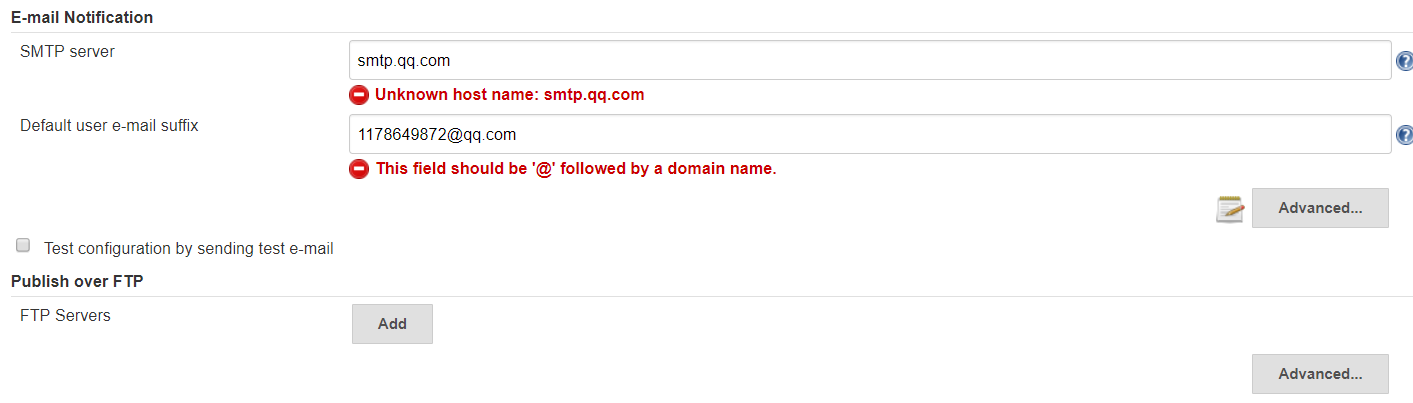


1. Extended E-mail Notification的配置

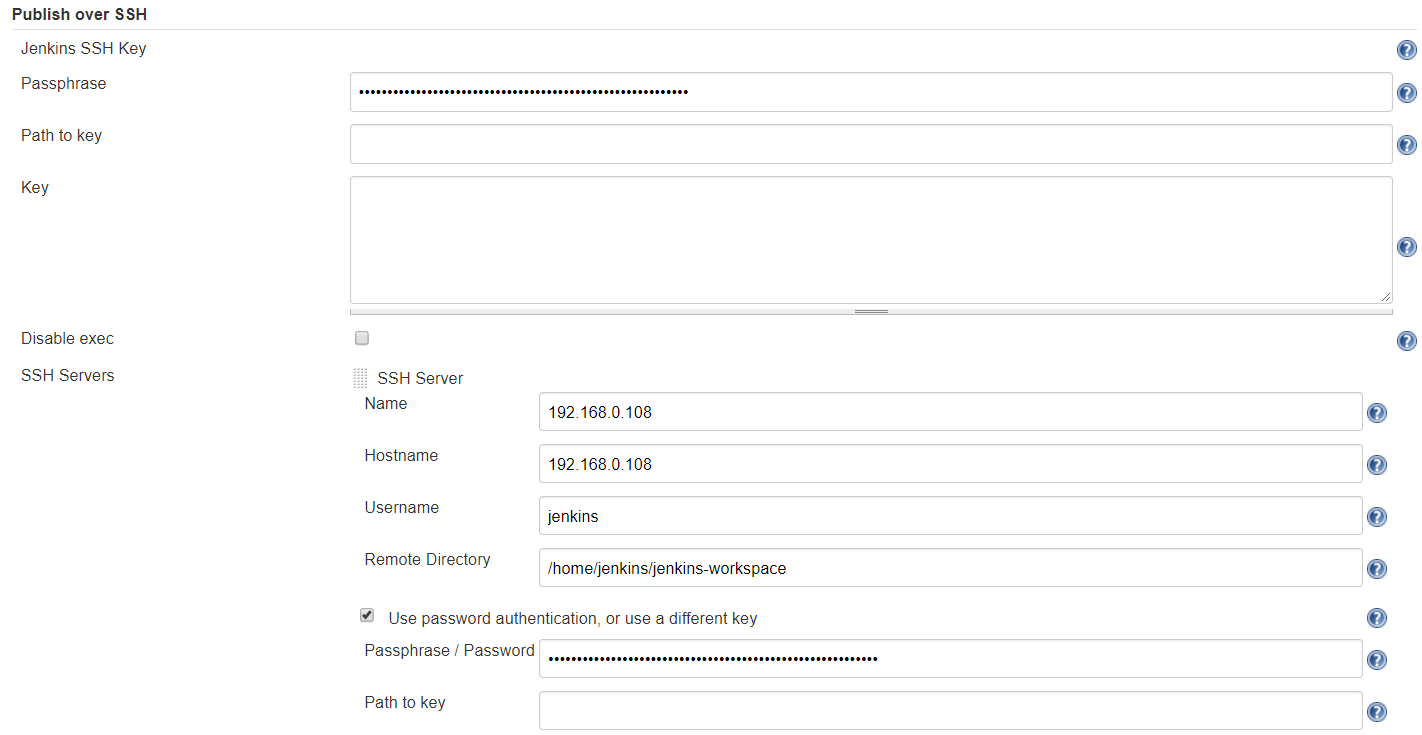


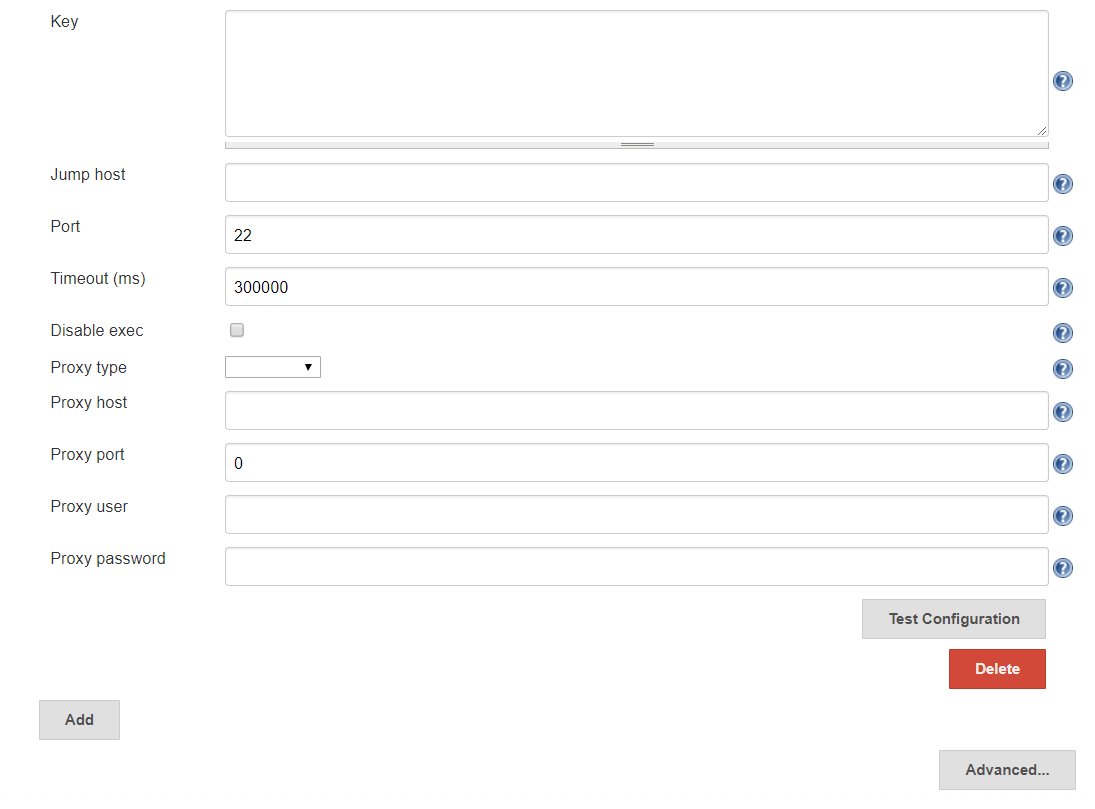


1. E-mail Notification 和 publish over FTP的配置



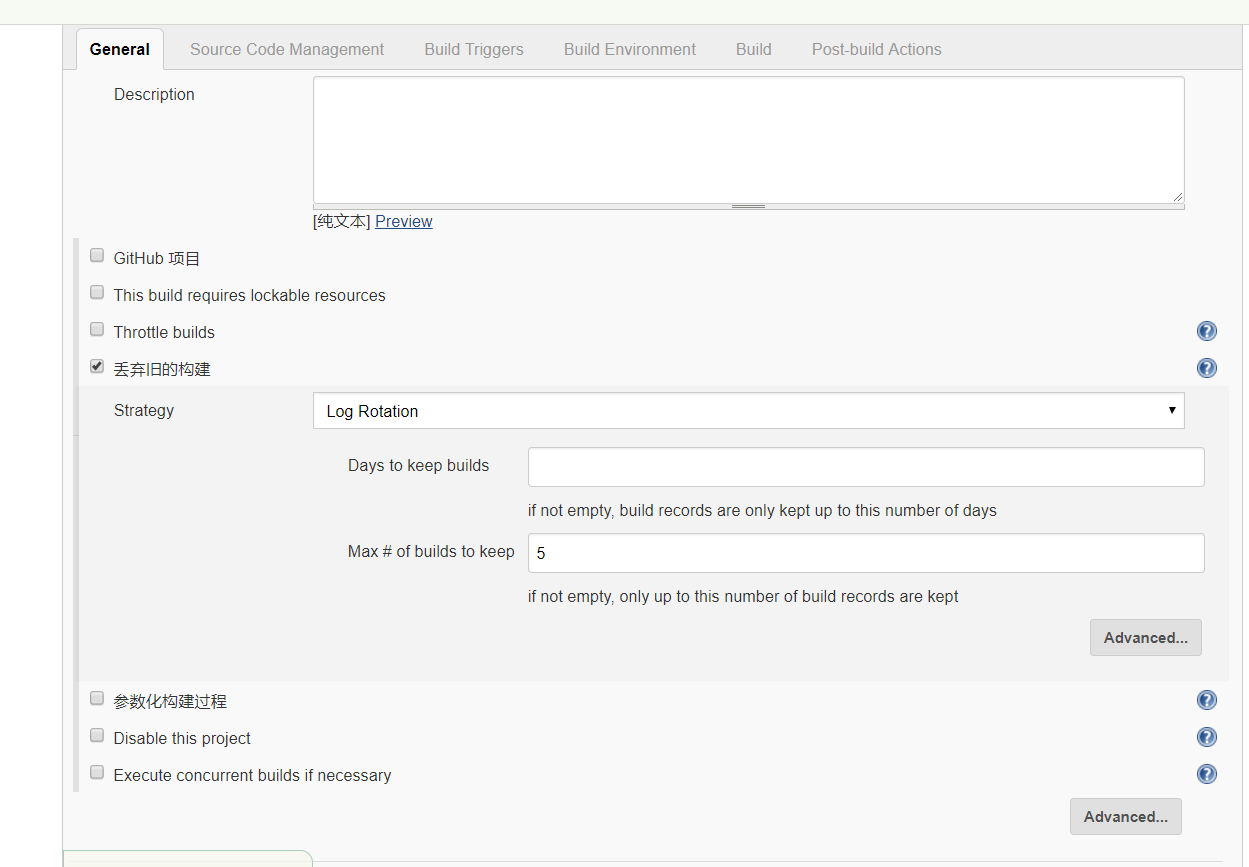
1. Publish over SSH的配置



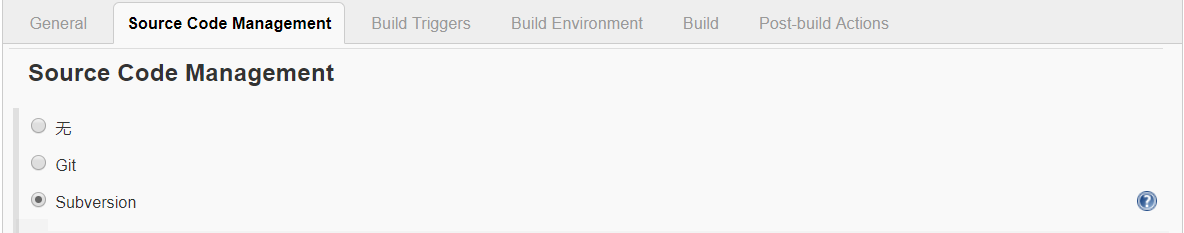
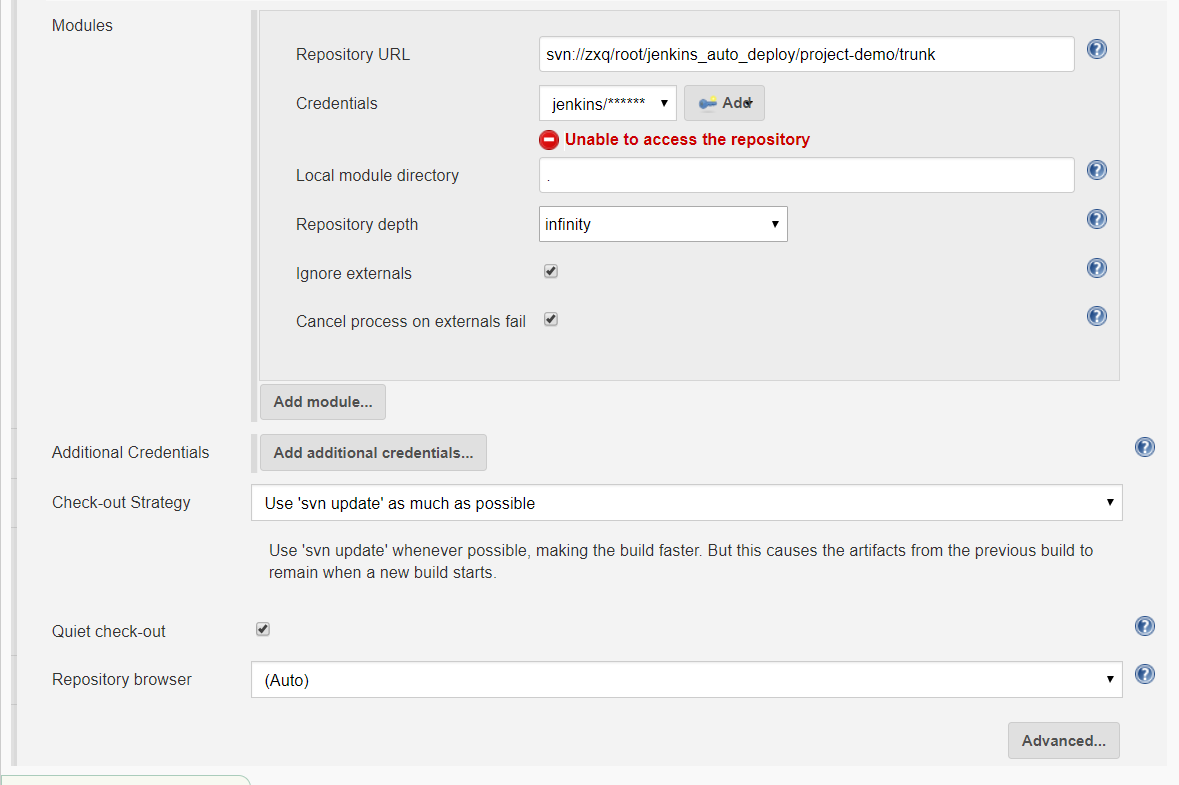


## 创建构建项目,此处以first-demo-build 为案例

1. general的配置



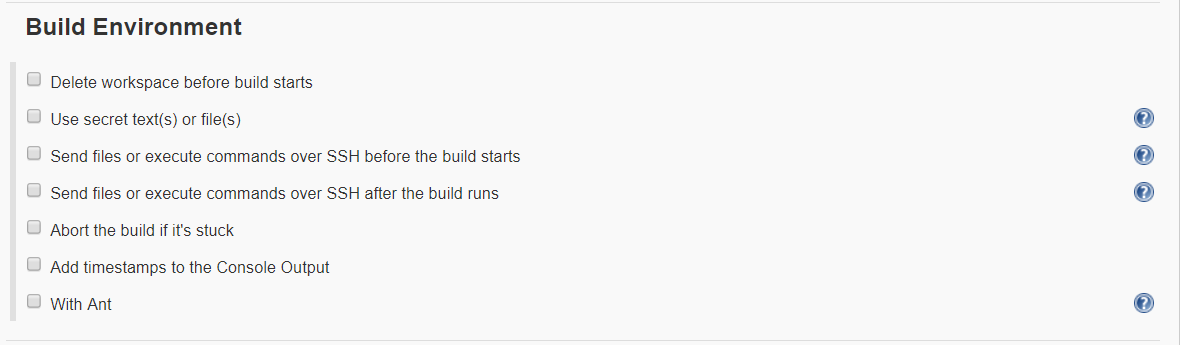
1. Source Code Management的配置



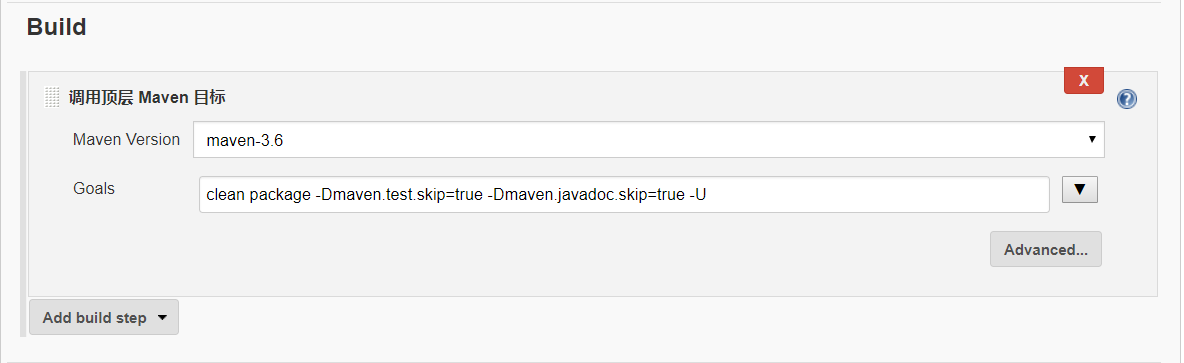
1. Build Triggers 的配置



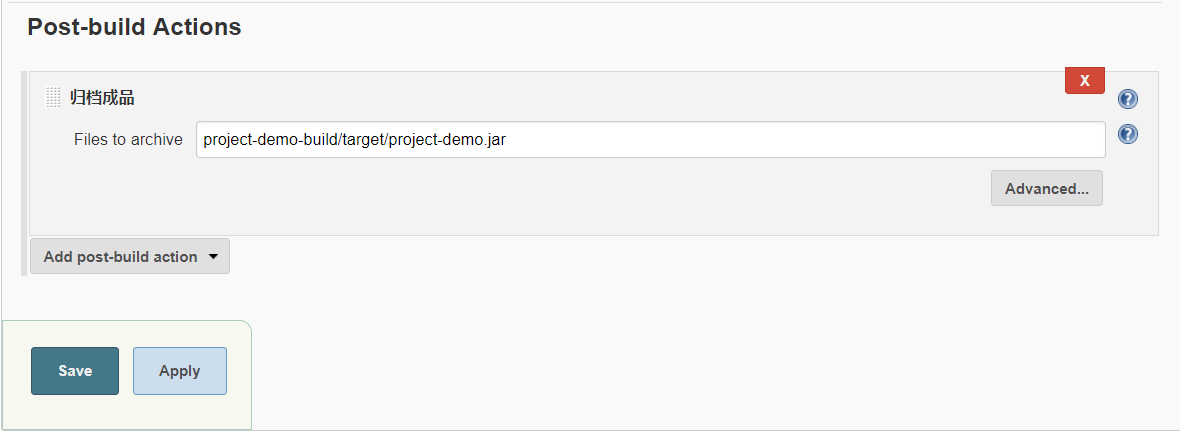
1. Build Environment



1. build的配置

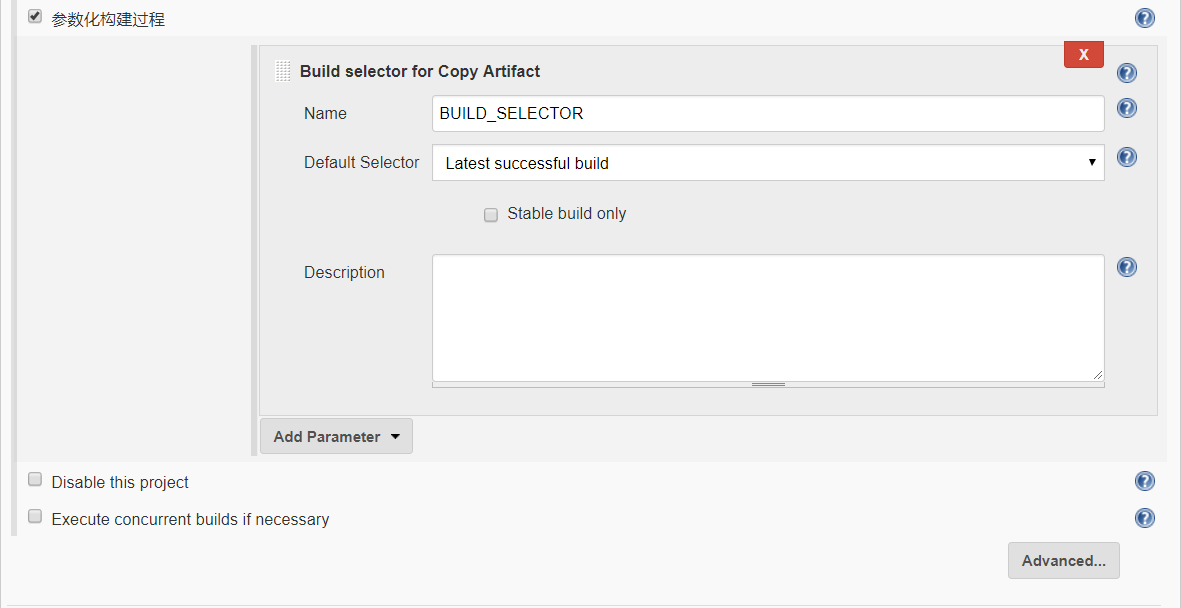
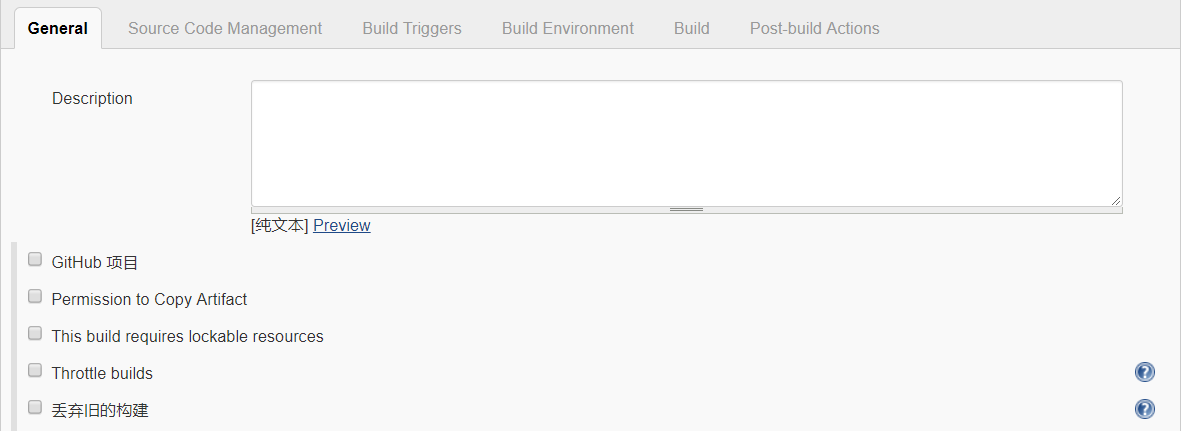


1. Post-build Actions的配置

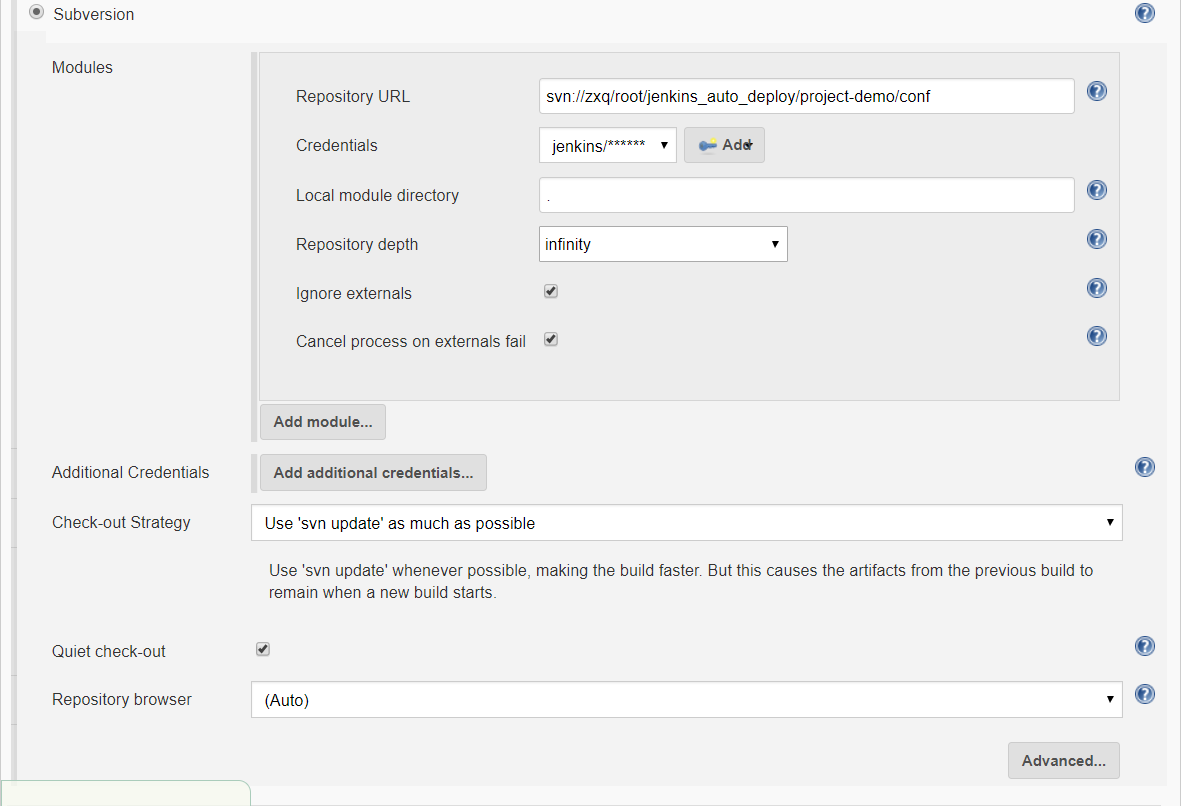
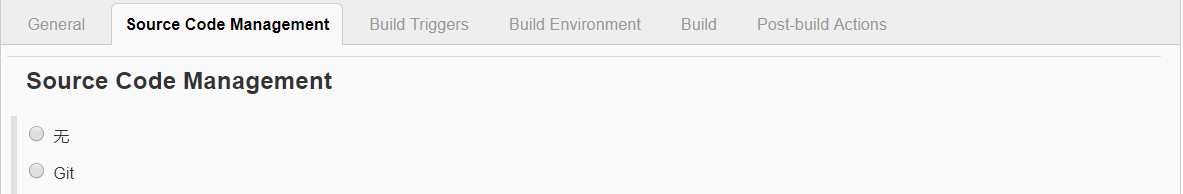


## 创建部署项目,此处以first-demo-deploy为案例

1. gerenal的配置



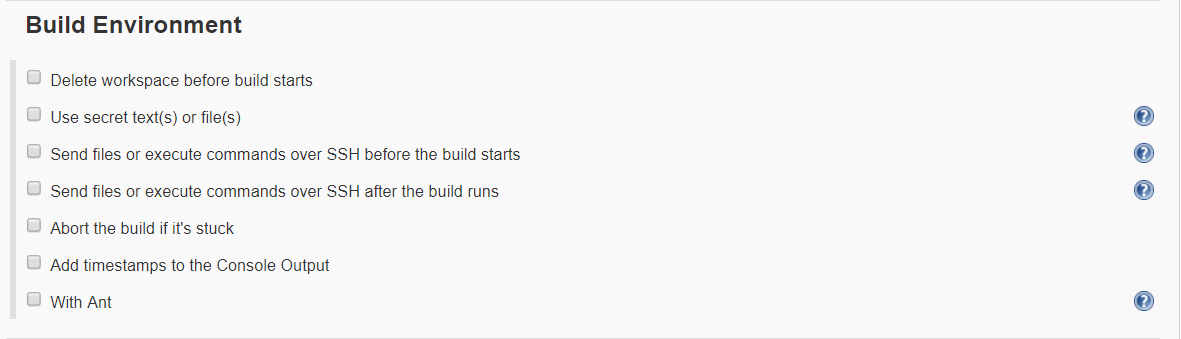
1. Source Code Management 的配置



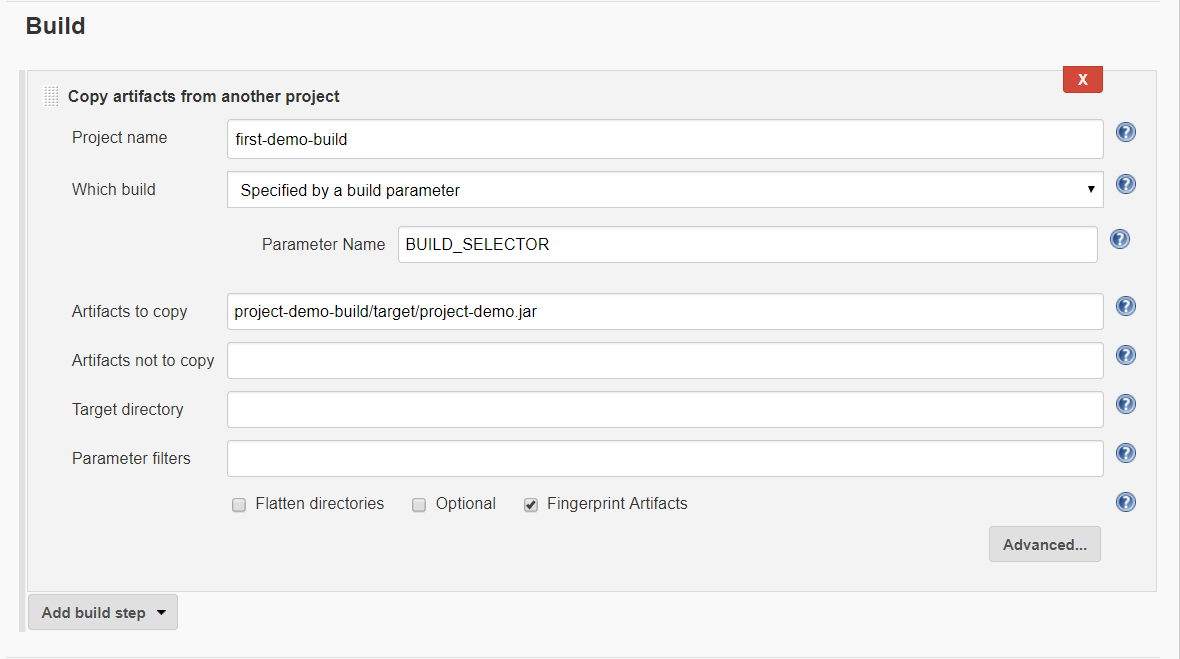
1. Build Triggers的配置



1. Build Environment的配置

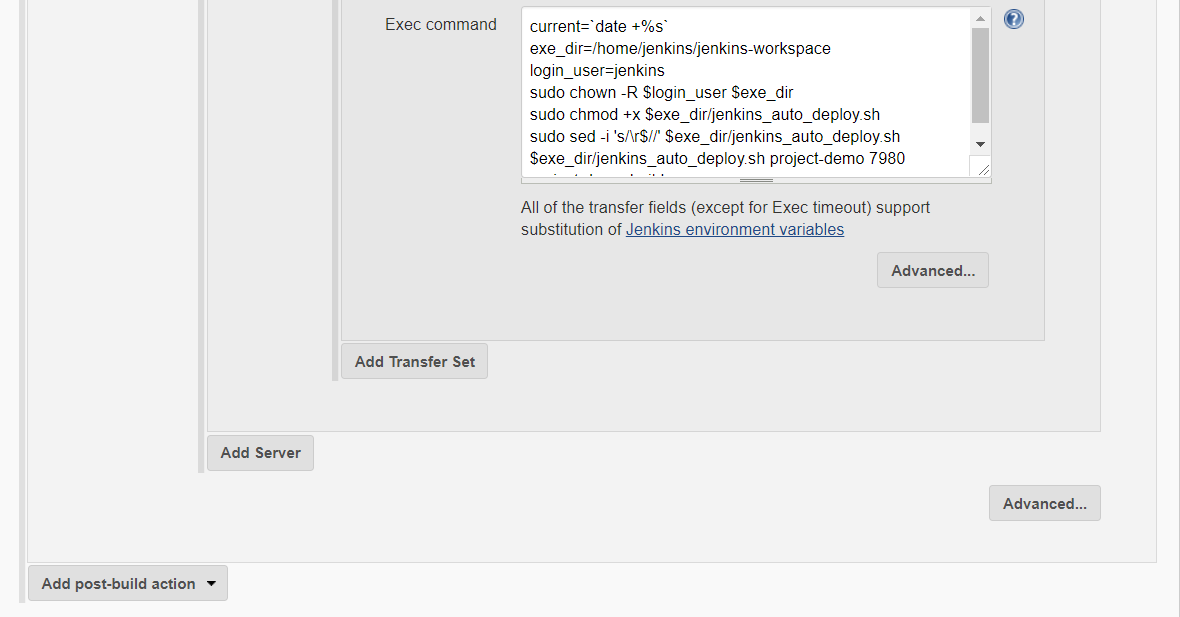
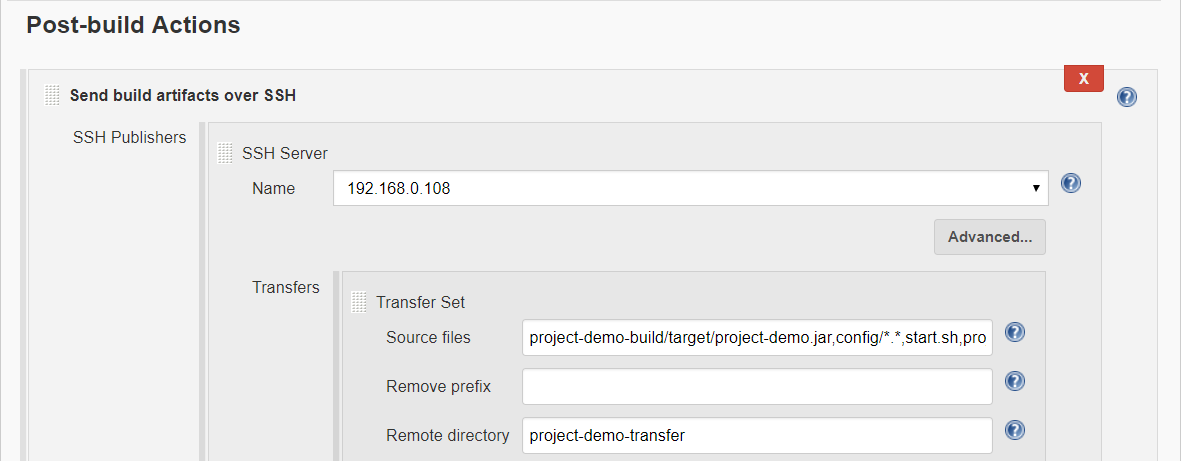


1. Build的配置



注意:此处填写的 artifacts to copy: project-demo-build/target/project-demo.jar 的值 与 first-demo-build项目中的 [归档成品]🡪[ Files to archive]的值 project-demo-build/target/project-demo.jar 应保持一致

1. Post-build Actions的配置



注意: source files 中的值为:

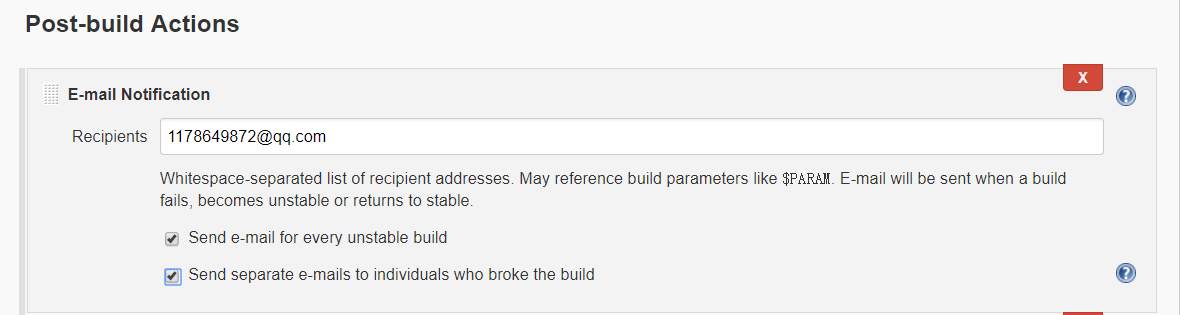
|  |
| --- |
| project-demo-build/target/project-demo.jar,config/\*.\*,start.sh,project-demo.conf,v.license |

exec command的值为:

|  |
| --- |
| current=`date +%s`  exe\_dir=/home/jenkins/jenkins-workspace  login\_user=webapp  sudo chown -R $login\_user $exe\_dir  sudo chmod +x $exe\_dir/jenkins\_auto\_deploy.sh  sudo sed -i 's/\r$//' $exe\_dir/jenkins\_auto\_deploy.sh  $exe\_dir/jenkins\_auto\_deploy.sh project-demo 7980 |

1. 邮箱通知配置,请参考 生产环境的[系统配置]+[build配置]+[deploy配置]案例

主要使用插件是



1. jenkins\_auto\_deploy.sh 脚本

完整脚本路径: 脚本范例备份\poseidon\auto\_deploy\jenkins\_auto\_deploy.sh

|  |
| --- |
| #! /bin/bash  # 运行命令 ./xx.sh project-demo 7980  # 得到当前目录  curr\_dir=$(cd `dirname $0`; pwd)  # 应用程序名称  app\_name=$1  # 应用程序执行命令  exec\_commend=install  # 应用程序的端口号  app\_port=$2  # 应用程序的进程pid  app\_pid=""  # 应用程序目录名称  app\_dir\_name=$app\_name  #临时传输的拷贝目录名称  temp\_transfer\_app\_dir\_name=$app\_name-transfer  # 应用程序目录  app\_dir=$curr\_dir/$app\_dir\_name  # 临时传输的拷贝目录  temp\_transfer\_app\_dir=$curr\_dir/$temp\_transfer\_app\_dir\_name  #打包的模块名称  package\_module\_name=$3  # 用户和用户组  user=jenkins  group=jenkins  # ==============================函数执行区=======================================  # 处理临时传输目录,使其规范化  dealWithTempTransferDir(){  if [ -d "$temp\_transfer\_app\_dir/$package\_module\_name" ]; then  sudo mv $temp\_transfer\_app\_dir/$package\_module\_name/target/$app\_name.jar $temp\_transfer\_app\_dir  sudo rm -rf $temp\_transfer\_app\_dir/$package\_module\_name  echo "-----临时传输目录处理完毕!"  else  echo "-----$temp\_transfer\_app\_dir/$package\_module\_name临时目录处理失败,相关路径不存在!"  fi  }  # 从临时传输文件目录中拷贝应用  copyApp(){  # 处理临时传输目录,使其规范化  dealWithTempTransferDir  # 删除应用程序目录  sudo rm -rf $app\_dir  #判断引用程序目录是否存在  if [ ! -d "$app\_dir" ]; then  sudo mkdir $app\_dir  sudo cp -f -r $temp\_transfer\_app\_dir/\* $app\_dir  # 授予创建目录的执行权限  sudo chmod -R 755 $app\_dir  checkUserAndGroupExist  echo -----目录创建成功,并成功拷贝数据.  else  echo -----应用程序目录删除失败:$app\_dir  exit 1  fi  # 删除临时传输目录  # sudo rm -rf $temp\_transfer\_app\_dir  }  # 检测用户及用户组是否创建  checkUserAndGroupExist(){  # exist状态,0:表示已经存在,1:表示不存在  # 检测用户组是否存在,不存在则进行创建  egrep "^$group" /etc/group >& /dev/null  group\_exist=$?  # 检测用户是否存在,不存在则进行创建  egrep "^$user" /etc/passwd >& /dev/null  user\_exist=$?  echo "----------------------------$group\_exist-------$user\_exist"    if [ $group\_exist -eq 0 ] && [ $user\_exist -eq 0 ]; then  # 设置启动用户的目录权限  echo "设置启动用户的目录权限"  sudo chown -R $user:$group $app\_dir  fi  }  # 启动应用  startApp(){  echo "开始服务 '$app\_name'"  # 开始服务  sudo su - $user -c "sudo service $app\_name start"    echo "正在等待服务启动成功"  app\_pid=""  count=0  # 若 进程pid为空,则监听pid  while [ -z "$app\_pid" ]  do  if [ $count -eq 60 ];then  echo 应用程序 $app\_name 在5分钟内没有监听到进程pid,自动跳出监听程序,避免脚本死循环执行...  break;  fi  count=`expr $count + 1`  sleep 5  # 此处可以改进为通过进程号获取端口号来监听...  app\_pid=$(sudo netstat -anp|grep 'LISTEN'|grep $app\_port|awk '{printf $7}'|cut -d/ -f1)  echo ---------$app\_pid --------------$count -------port:$app\_port  done  #等待15秒,包服务启动成功  sleep 15  echo "应用程序$app\_name启动成功,进程pid=$app\_pid"  }  # 安装应用  installApp(){  copyApp  echo "-----正在安装应用app\_name=$app\_name"  # 检测要安装的应用是否存在,不存在则退出安装  if [ ! -f "$app\_dir/$app\_name.jar" ]; then  echo "没有找到$app\_name.jar在目录$app\_dir下"  exit 1  fi  if [ ! -f "$app\_dir/config/application.yml" ]; then  if [ ! -f "$app\_dir/config/application.properties" ]; then  echo "没有找到application.yml 和 application.properties 在目录$app\_dir/config"  exit 1  fi  fi  # 开始安装用户和组  echo "正在检测安装用户和组是否存在,不存在则进行创建"  # 检测用户组是否存在,不存在则进行创建  egrep "^$group" /etc/group >& /dev/null  if [ $? -ne 0 ]; then  echo "创建用户组开始"  sudo groupadd $group  echo "创建用户组结束"  fi  # 检测用户是否存在,不存在则进行创建  egrep "^$user" /etc/passwd >& /dev/null  if [ $? -ne 0 ]; then  echo "创建用户开始"  sudo useradd -s /bin/bash -d /home/$user -g $group -m $user  echo "创建用户结束"  fi  # 设置启动用户的目录权限  echo "设置启动用户的目录权限"  sudo chown -R $user:$group $app\_dir    # 安装 $app\_name 服务  echo 安装 $app\_name 服务    sudo ln -s $app\_dir/$app\_name.jar /etc/init.d/$app\_name  #chkconfig 2345 80 90  #description:project-demo  # sudo chkconfig --add $app\_name  echo "$app\_name服务安装结束"  # 授予相关权限  sudo chmod -R 755 $app\_dir  # 应用程序安装完成  echo "应用程序安装完成"  startApp  }  updateApp(){  # 检测要更新的应用是否存在,不存在则退出更新  if [ ! -f "$app\_dir/$app\_name.jar" ]; then  echo "没有找到$app\_name.jar在目录$app\_dir下"  exit 1  fi  if [ ! -f "$app\_dir/config/application.yml" ]; then  if [ ! -f "$app\_dir/config/application.properties" ]; then  echo "没有找到application.yml 和 application.properties 在目录$app\_dir/config"  exit 1  fi  fi  echo "停止服务,'$app\_name'"  sudo su -$suer -c "service $app\_name stop"  sleep 15  copyApp  startApp  }  uninstallApp(){  echo "卸载应用开始"  echo "停止服务 '$app\_name'"  sudo su - $user -c "service $app\_name stop"  sleep 15  echo "从 init.d 中删除服务"  # sudo chkconfig --del $app\_name  sudo rm /etc/init.d/$app\_name  echo "删除应用文件"  sudo rm -rf $app\_dir    echo "卸载应用结束"  }  # ==============================命令执行区=======================================  # 传入参数处理区  # 判断应用名称是否为空  if [ -z "$app\_name" ]; then  echo "应用的名称[app\_name]不能为空!"  exit 1;  fi  # 判断应用端口是否为空  if [ -z "$app\_port" ]; then  echo "应用的端口[app\_port]不能为空!"  exit 1;  fi  # 判断应用打包模块名称是否为空  if [ -z "$package\_module\_name" ]; then  package\_module\_name=$app\_name  fi  echo "应用打包模块的名称为: $package\_module\_name"  # 判断当前应用是否已经安装  # sudo chkconfig --list | grep -w $app\_name >& /dev/null  # 安装状态,0:表示服务已经安装,1:表示服务没有安装  install\_status=0  if [ ! -f "/etc/init.d/$app\_name" ]; then  install\_status=1  fi  if [ $install\_status -ne 0 ] || [ ! -f "$app\_dir/$app\_name.jar" ]; then  if [ $exec\_commend = "uninstall" ]; then  echo "没有找到对应的服务,操作取消"  else  installApp  fi  else  echo "找到 $app\_name 服务,该服务已经被安装!"  if [ $exec\_commend = "uninstall" ]; then  echo "正在准备卸载应用:$app\_name"  uninstallApp  else  echo "正在重新更新服务"  updateApp  fi  fi |

## 备用