JENKINS 暴力撞门详细教程

JENKINS

以 Quick4j 项目为蓝本

Blog：http://www.cnblogs.com/horizonli/

声 明

注意：本文为**撞门**级别文章，需要一定的计算机基础知识作为阅读基础。

排版撰写共享撞门文章真的挺辛苦的，如果您认为文章还不错或者有所收获，您可以通过扫描下方的二维码进行**【打赏】**或者扫描关注**即将写作**的公众号二维码，因为这几种方式都是支持我继续写作，分享的最大动力！公众号将记录工作生活,技术内容,个性观点等内容，欢迎您的关注

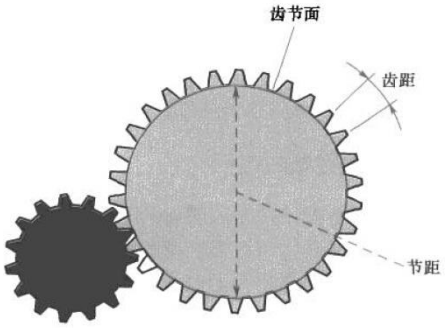
目 录

[原]Jenkins(一)---我理解的 jenkins 是这样的

1、齿轮

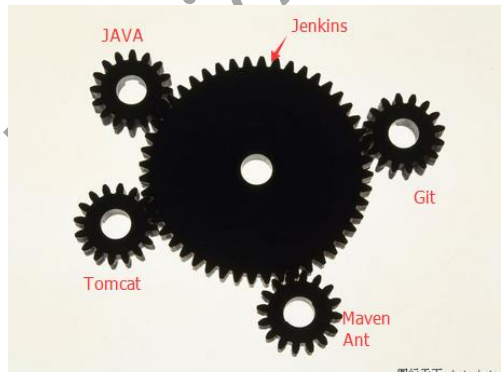
如果将 java / maven / ant / git / tomcat / jenkins 等等软件比喻为齿轮：如下图

两个软件在一起可以驱动另外一个软件：如下图



如果把这些软件要集成在一起工作，那么这个软件就可以存在其他软件的中间来驱动各个软件工作：如下图：

jenkins 就是类似于中间那个齿轮，来驱动其他软件的集成一起工作,如下图



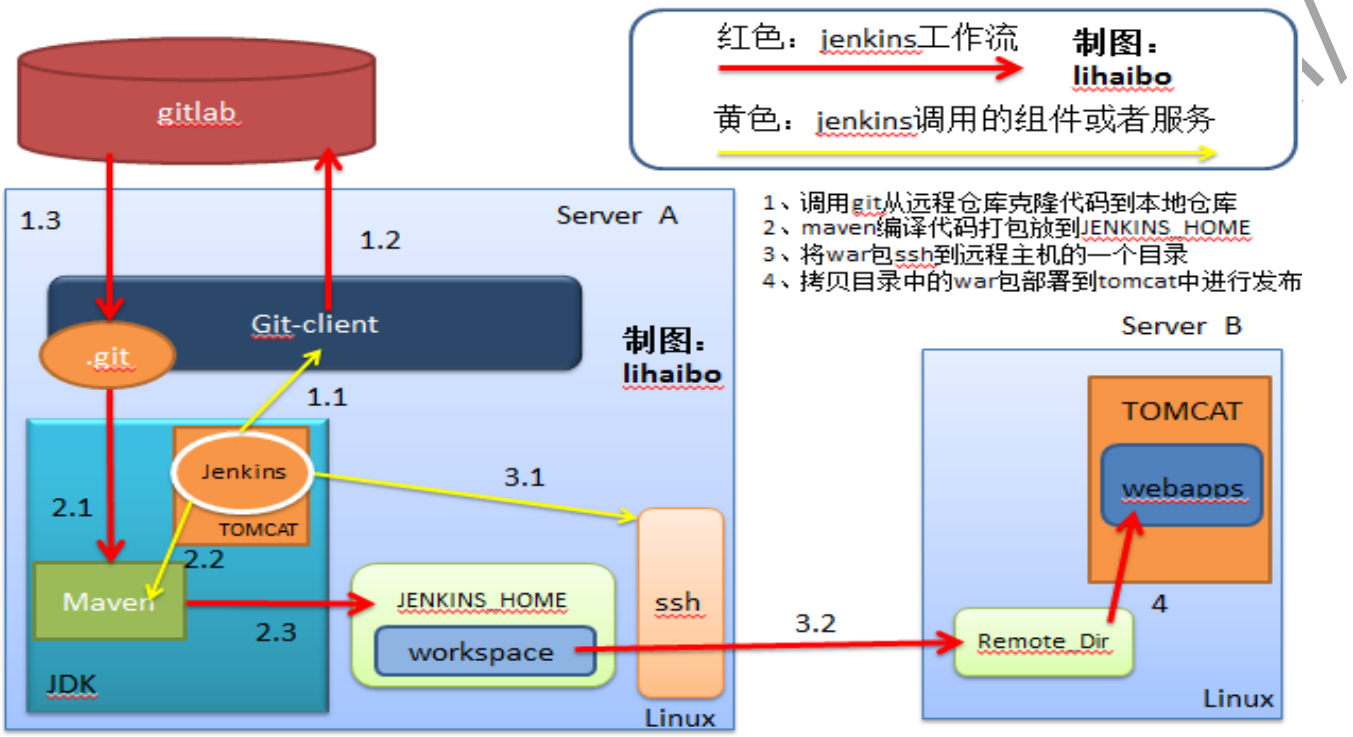
[原]Jenkins(二)---jenkins 之 Git+maven+jdk+tomcat

git+maven+jdk+tomcat 这四个软件可以百度在 linux 下的安装，不再赘述。

server A ---> jenkins 主机 ip：192.168.100.119

server B ---> 远程部署主机 IP：192.168.100.118

先看看作为重点的 jenkins ，先看看准备怎么安装 jenkins 和 jenkins 在服务器图例中的位置：



一、 安装 jenkins：

下载地址： jenkins-ci.org

或者链接: http://pan.baidu.com/s/1i4Ga2tZ 密码: 7ghf

只需要三步：1. 在 linux 上安装好 tomcat (tomcat 启动文件 server.xml 中 在启动端口（假设为 8080）后面加上 URIEncoding="UTF-8")

```
root@Local-Machine-1:/opt/tomcat6/apache-tomcat-6.0.45/conf# cat server.xml |grep URIE
    <Connector port="9090" URIEncoding="UTF-8" protocol="HTTP/1.1"
```

2. 创建 jenkins 的工作目录： mkdir -P /opt/jenkins ， 在 /etc/profile 中添加环境变量 export JENKINS_HOME=/opt/jenkins，

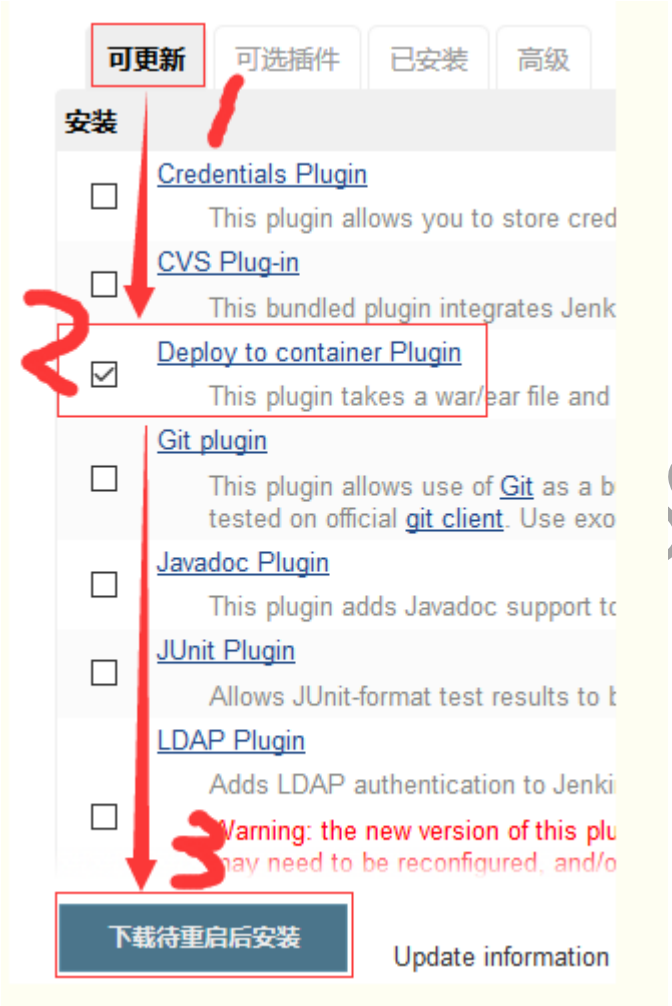
source /etc/profile 使修改生效

如果不添加环境变量 JENKINS_HOME，则默认工作目录用户工作目录下在：/root/.jenkins

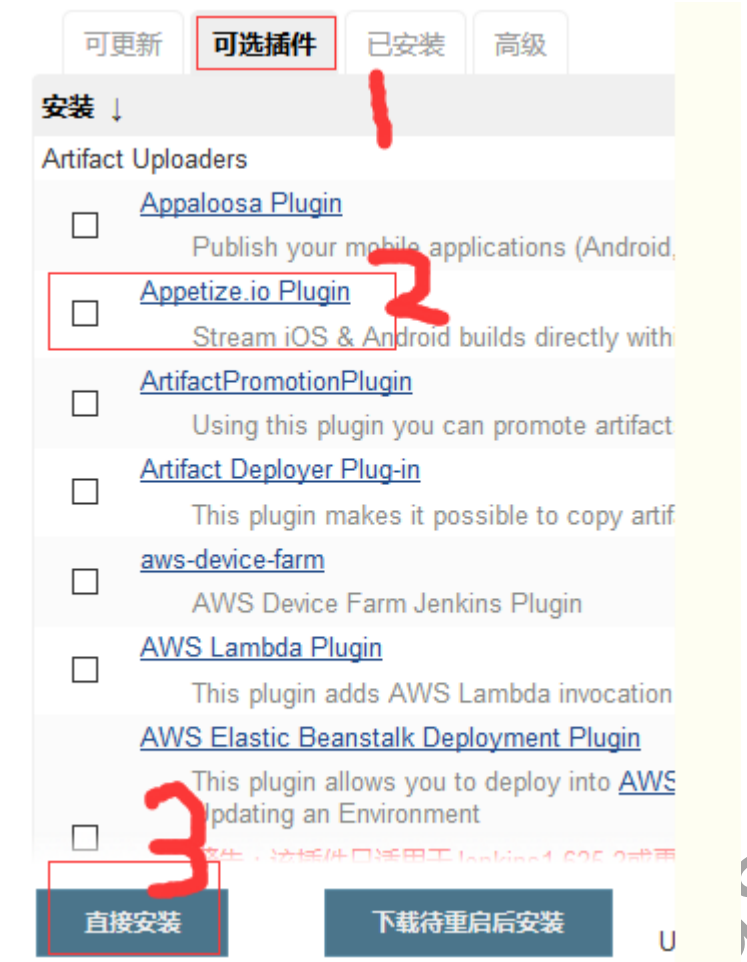
3. 在 jenkins-ci.org 下载 jenkins 的安装包 jenkins.war,拷贝在 tomcat 的目录 webapp 中，启动 tomcat。在浏览器输入 http://IP:8080/jenkins

[原]Jenkins(三)---Jenkins 初始配置和插件配置

从 Jenkins(二)中可以知道 jenkins 的工作目录为/opt/jenkins
【很重要!!!】【很重要!!!】【很重要!!!】在配置此目录以前，将这两台的主机进行配置为 ssh root 用户无密码远程登录（即相互登录不用输入密码），可以 google 一下。
jenkins 是一个可以添加插件的集成工具，添加插件是很重要的一步。
---------- 添加插件 （我使用的是第三种） ----------
点击左侧的“系统管理”----"管理插件"------“已安装”---查找需要安装的插件 git-client/scm-api/git/publish over SSH/scp/ssh
1、更新插件



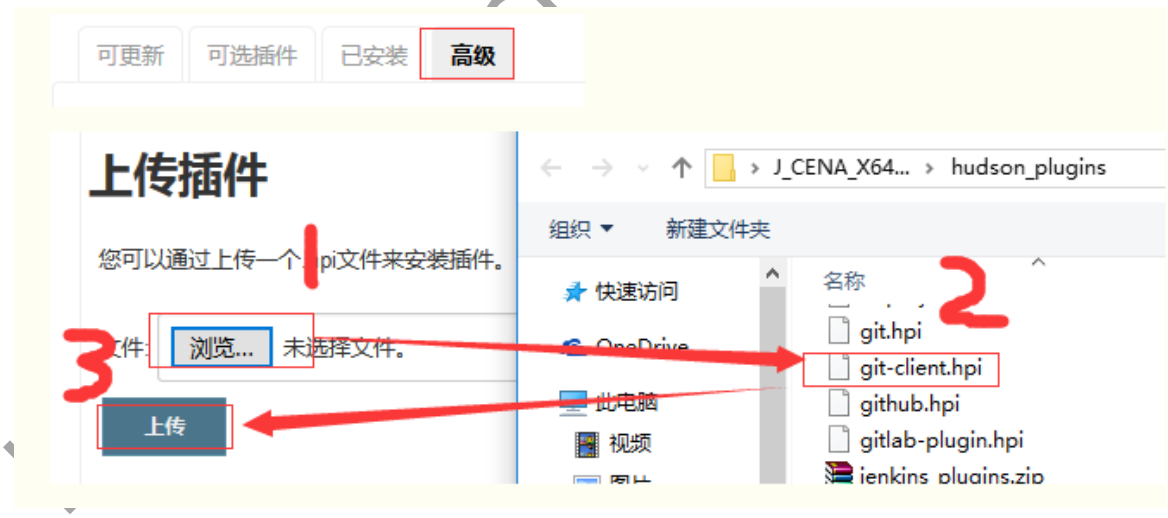
2.在线添加插件---可选插件



3.离线安装--选择“高级”选项卡

需要安装：git-client/scm-api/git　从 gitlab 拉取代码到本地仓库

publish over SSH/scp/ssh　远程 ssh 部署插件



我的插件，可以在这里下载:

链接: http://pan.baidu.com/s/1pLotSmR 密码: jzud

一、点击左侧的“系统管理”----"系统设置"

二、Maven Configuration

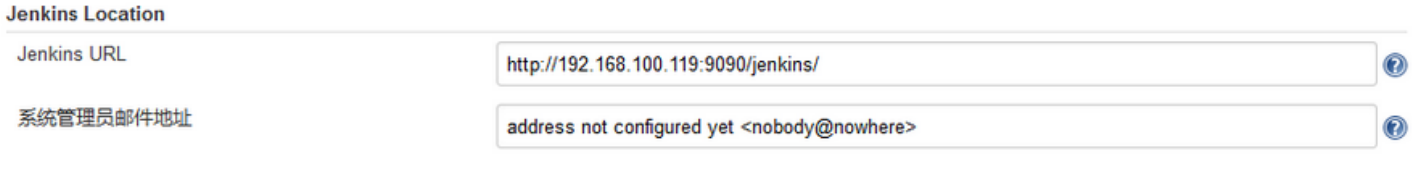
或者

三、JDK

四、git-client

五、Maven

六、jenkins 的访问地址（默认自己生成）



七、SSH

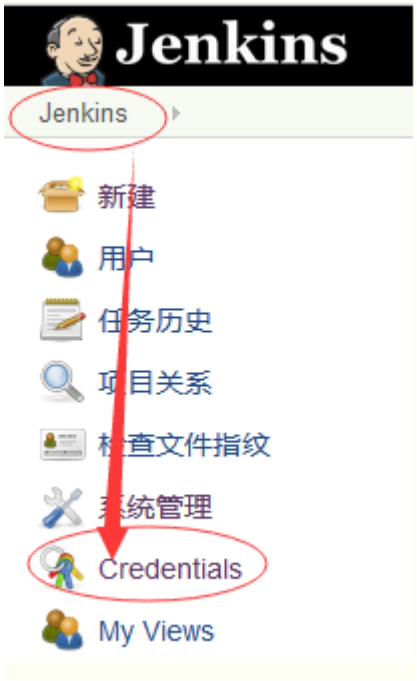
指定 ssh 端口

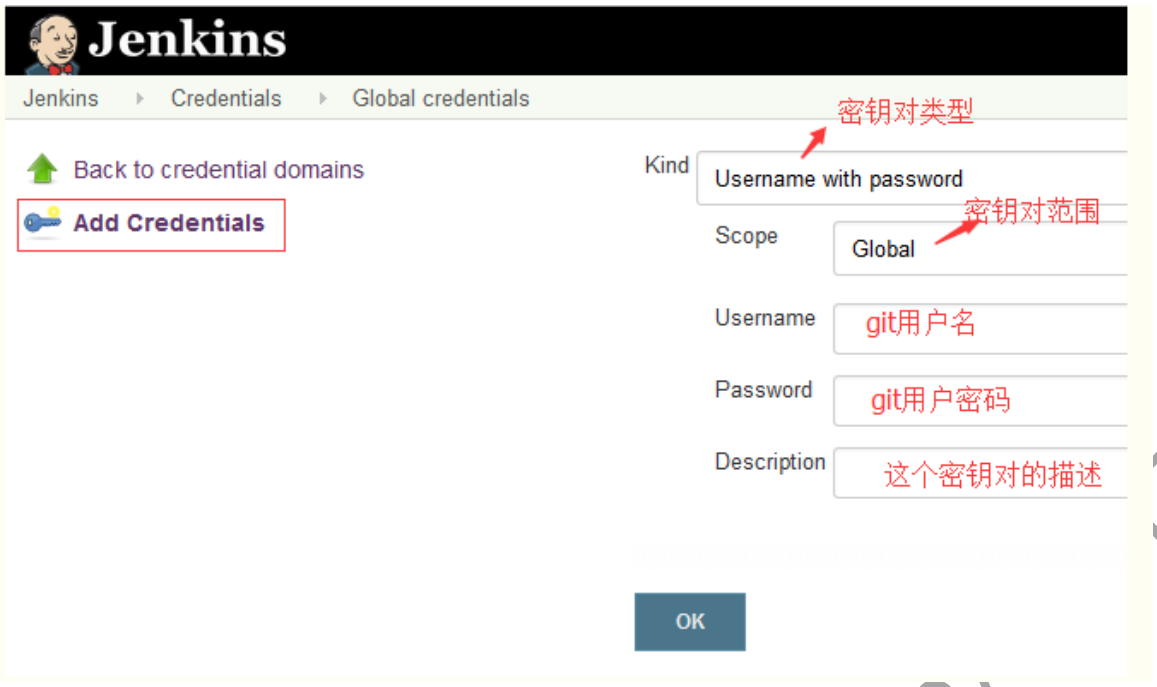
【部署很重要的一个组件】

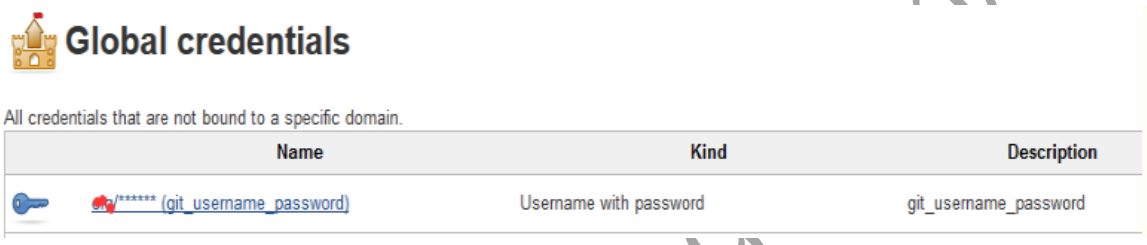
[原]Jenkins(四)---Jenkins 添加密钥对

一、添加密钥

1.添加 git 用户和 git 密码对 ， 用于 git 客户端从 gitlab 上拉取代码到本地



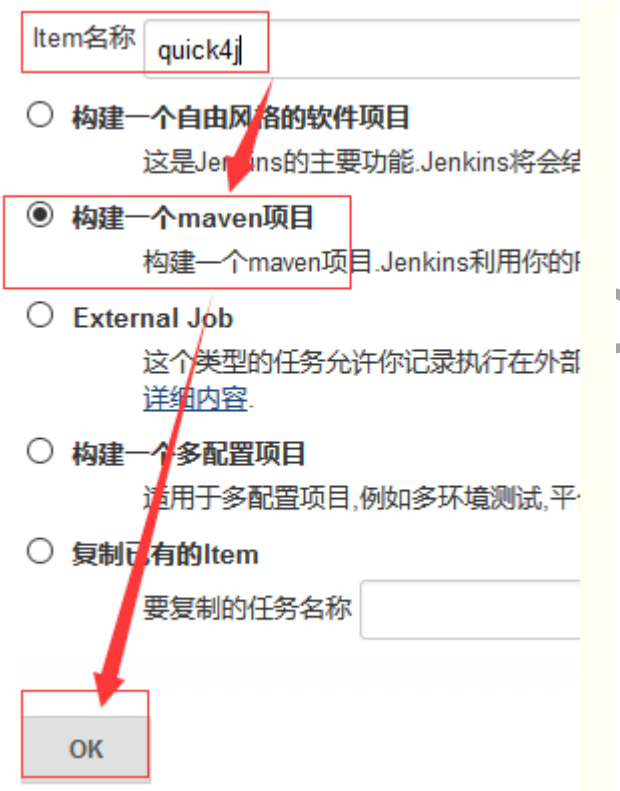




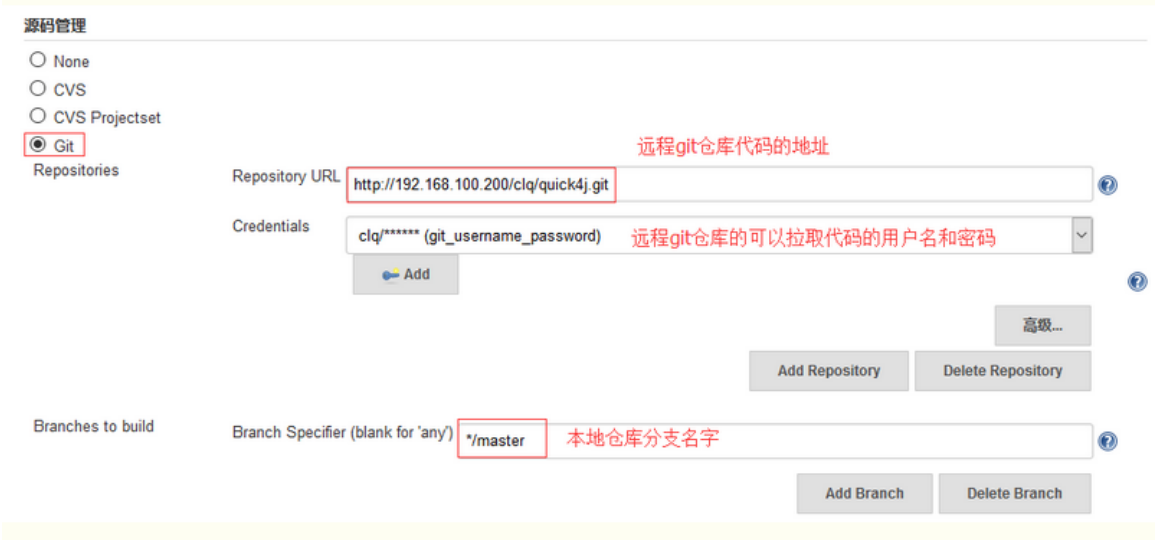
[原]jenkins(五)---jenkins 添加项目

一、新建项目

二、配置项目



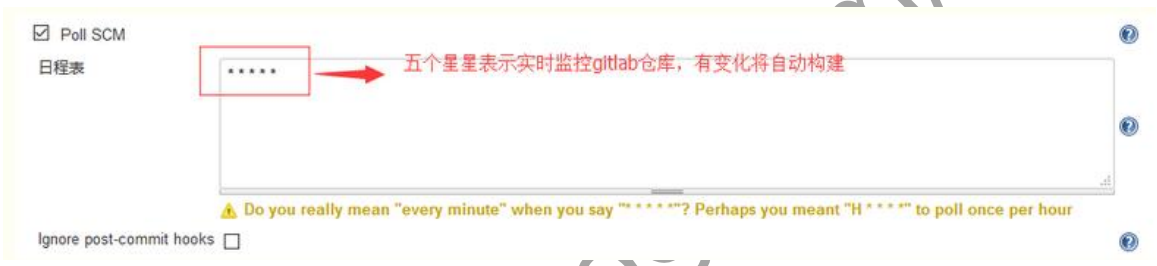
配置远程仓库：主要目的是从远程仓库拉取代码下来



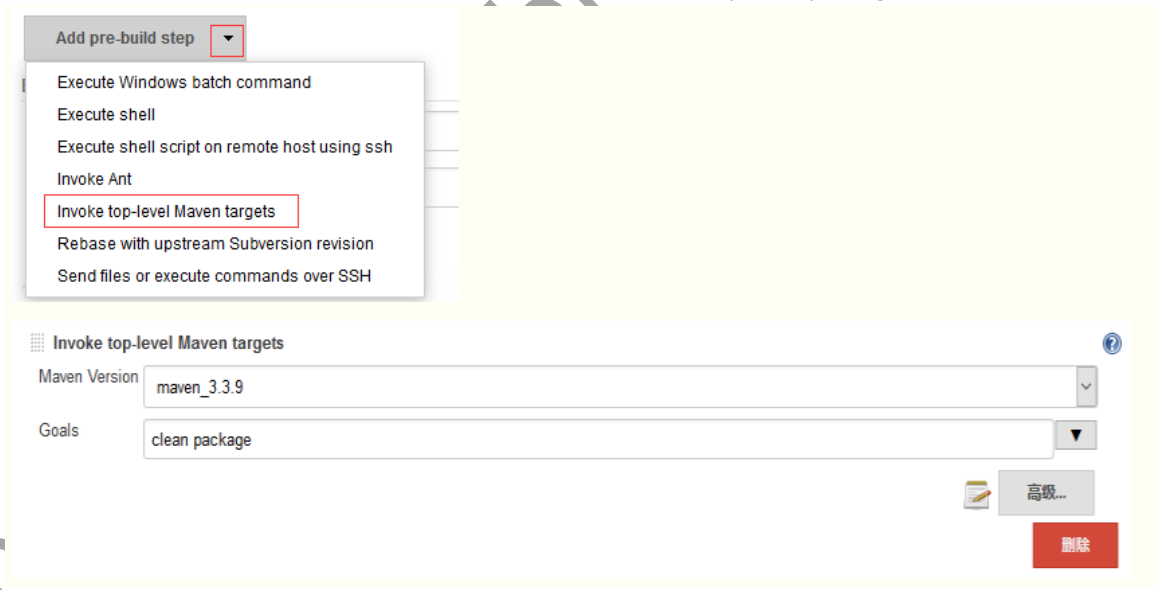
实时构建

Poll SCM　　定期检查 如果源码有变更 就 build 否则不 build

build periodically　定时 build 不检查源码



配置 maven 的目标：意思是使用 maven 进行 clean 、complier、package 、install 。



配置了上面的信息后，maven 会将打包的 war 包放在下面的这个目录中:

/opt/jenkins_tomcat6/jobs/quick4j_demo/workspace/target

/opt/jenkins_tomcat6　　表示 jenkins 工作目录

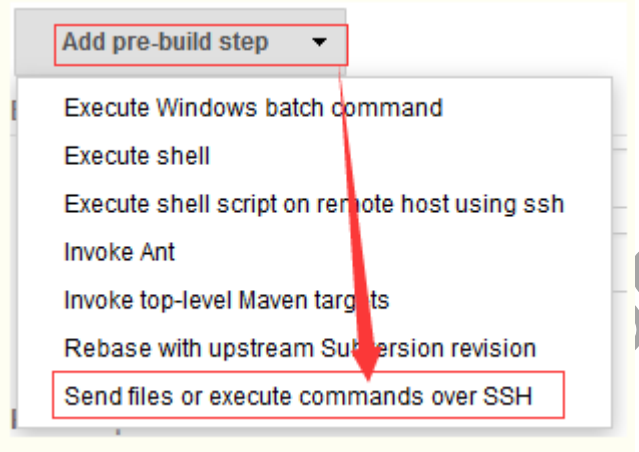
jobs/quick4j_demo　　表示 jenkins 管理 quick4j_demo 项目的目录

workspace　　表示 maven 等插件的工作目录

构建项目：这里表示构建项目使用 maven 的话 使用哪个 maven 配置文件：

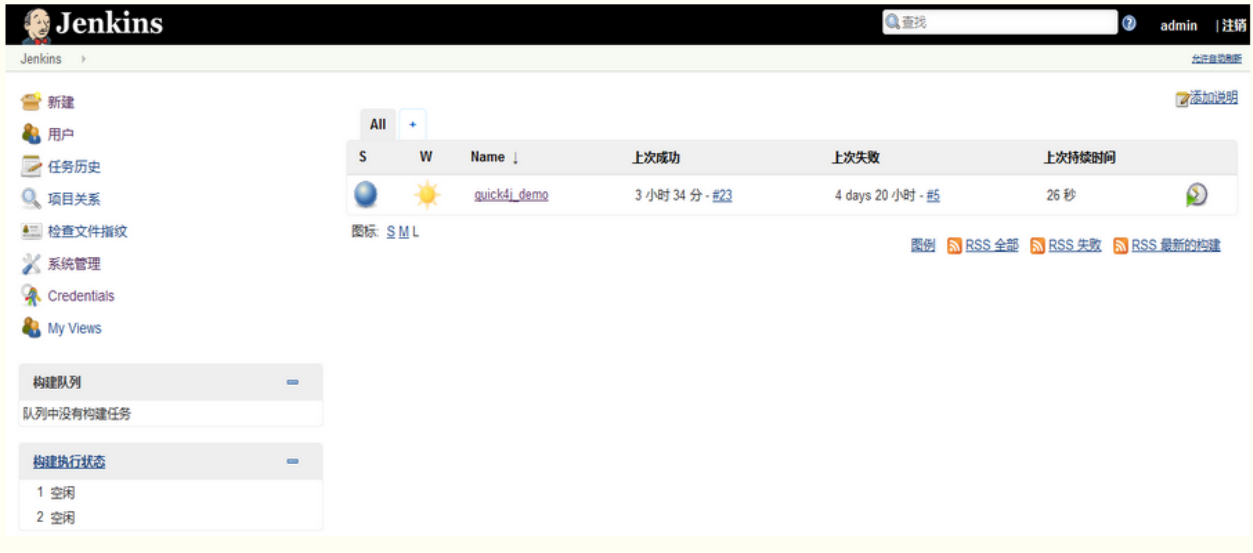
（这里的 clean 和 package 与 文章前面【配置 maven 的目标】里面的一样的意思，至于执行哪个？测试了下是执行【配置 maven 的目标】中的 goals）

部署项目：



添加项目完成

回到主界面：



[原]jenkins(六)---jenkins 远程部署脚本

在远程主机上创建一个 shell 脚本放置在自定义路径中：这里我放置在/opt 下面：取名叫 auto_deploy.sh 即：auto_deploy.sh 在远程主机上什么位置，上图中的 Exec command 中就要写到哪个位置。
shell 脚本怎么写可以自己定义：这里分享下自己写的脚本，以供参考：

```
#!/bin/bash
#Time
log_time=`date +[%Y-%m-%d]%H:%M:%S`

###manual_properties###
tomcat_basehome=/opt/tomcat6/apache-tomcat-6.0.45
tomcat_port=9090
shell_environment=/bin/bash
war_Dir=/opt/jenkins
war_Name=quick4j.war
###manual_properties###


#update server environment
echo "*********************************  ${log_time} ************************************"
echo "updating server  environment start"
export JAVA_HOME=/app/java/jdk1.8.0_11
export JRE_HOME=/app/java/jdk1.8.0_11/jre
export PATH=$JAVA_HOME/bin:$PATH
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar/
export CATALINA_2_HOME=/opt/tomcat6/apache-tomcat-6.0.45
export CATALINA_2_BASE=/opt/tomcat6/apache-tomcat-6.0.45
export TOMCAT_2_HOME=/opt/tomcat6/apache-tomcat-6.0.45
sleep 3
echo "updating server  environment  end"

#build check funcation
echo "check tomcat status..."
check_tomcat_status(){
      netstat -ant|grep ${tomcat_port} > /dev/null
      t=$?
       if [ $t -eq 0 ]; then
           echo "tomcat is running....port is ${tomcat_port}"
           echo "shutdown tomcat....."
           echo ">>>>>>>shutdown tomcat begin<<<<<<<<"
```

```
            ${shell_environment} ${tomcat_basehome}/bin/shutdown.sh
            echo ">>>>>>>shutdown tomcat end <<<<<<<<"
            sleep 5
        elif [ $t -ne 0 ];then
              echo "tomcat is poweroff"
               ${shell_environment} ${tomcat_basehome}/bin/shutdown.sh
              sleep 5
        fi
}

#check tomcat status invoke function
check_tomcat_status


#transfer  application package
deploy_Loaction=${tomcat_basehome}/webapps/
war_Dir_Data=`ls ${war_Dir}`
echo "--------------  begin  transfer  war package to tomcat webapps --------------------"

if [ -z $war_Dir ];then
     echo "Folder ${war_Dir} is empty.please check war package in this folder!"
     exit 1
else
     echo "Find ${war_Dir} exist war package ${war_Name}"
    # echo "deleteing old  package ${war_Name} in ${war_Dir}"
    # rm ${war_Dir}/${war_Name}
     echo "deleteing old  package ${war_Name} in ${deploy_Loaction}"
     rm ${deploy_Loaction}${war_Name}
     echo "start  transfer ${war_Name} to ${deploy_Loaction}"
     cp ${war_Dir}/${war_Name}  ${deploy_Loaction}
     sleep 3
fi
echo "--------------  transfer  war package to tomcat webapps  end --------------------"
#reboot tomcat
echo " >>>>>>>  rebooting  tomcat begin <<<<<<<<"
${shell_environment} ${tomcat_basehome}/bin/startup.sh
echo " >>>>>>>  rebooting  tomcat end <<<<<<<<"
echo "the log you can read in canalina.out"
echo "********************** deploy war package into container Successlly  **********************************"
```

远程部署脚本.txt

[原]Jenkins(七)---jenkins 项目编译测试发布由 maven 构建的 web 项目

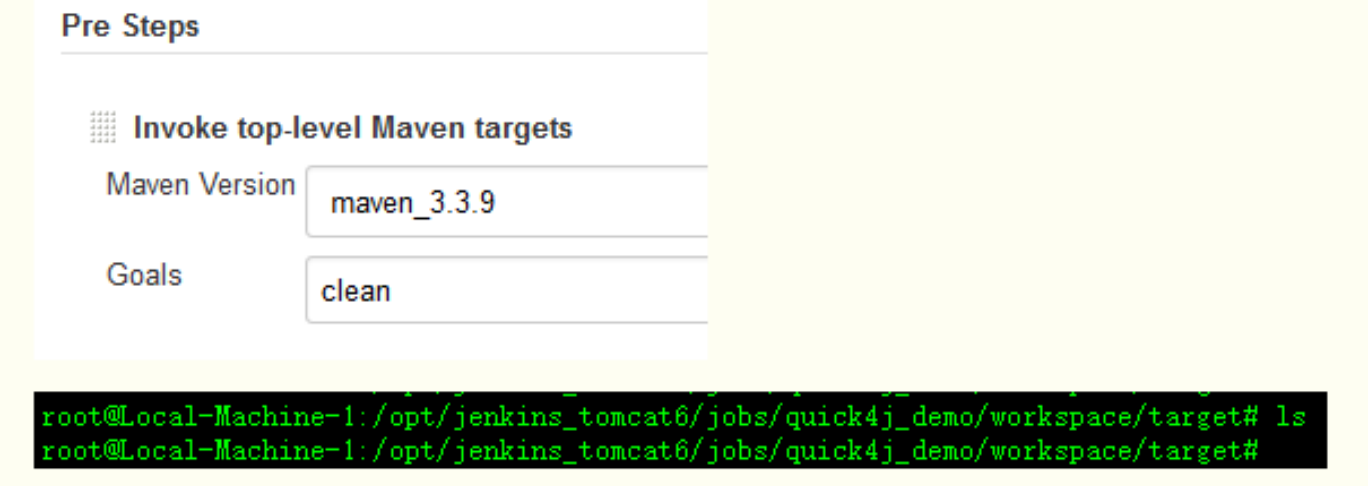
一、使用 maven 编译

（maven 编译 与 测试 test 和打包 package 和 部署 install 类似，不再赘述 ）

在项目的配置页面中有个 maven 配置：里面只有一个 clean 就是清除以前的构建信息：

之前我使用了 clean package 来编译打包：结果如下图：

当执行完只有 clean 的时候，之前构建的信息就被删除了：如下图



附注解：

/opt/jenkins_tomcat6/jobs/quick4j_demo/workspace/target

/opt/jenkins_tomcat6 表示 jenkins 工作目录

jobs/quick4j_demo 表示 jenkins 管理 quick4j_demo 项目的目录

workspace 表示 maven 等插件的工作目录

【jenkins 项目中使用 maven 扩展】

这里的 Goals 就是用 maven 编译要用的命令如下图中所示：

jenkins 使用 maven 编译以 invoke top-level Maven targets 中的 Goals 为准：

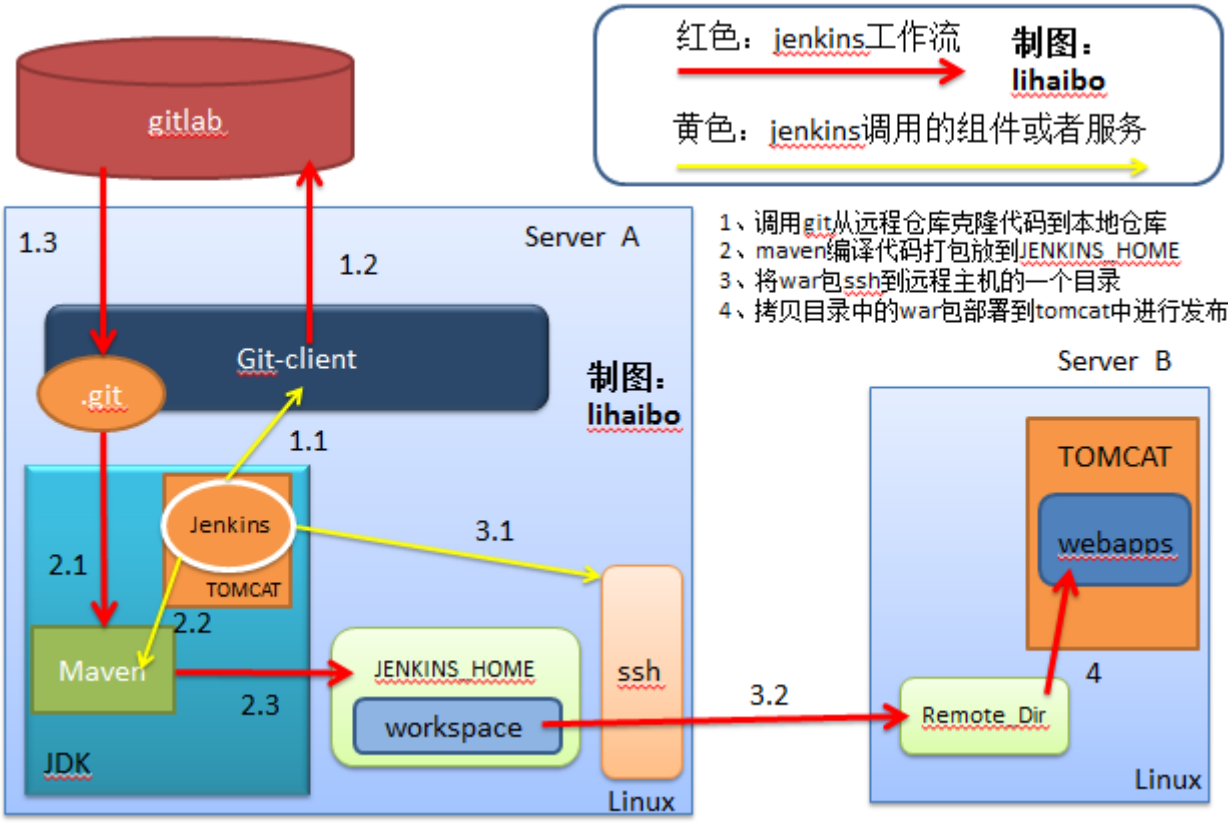
二、部署到远程主机的 tomcat 中

重点：两台机器要 ssh 远程 root 无密码登录。

再回到 Jenkins(二)---jenkins 之 Git+maven+jdk+tomcat 中的全局图：

server A ---> jenkins 主机 ip：192.168.100.119

server B ---> 远程部署主机 IP：192.168.100.118



下面是编译过程日志

紫色字体：jenkins 使用 git 从 gitlab 仓库 http://192.168.100.200/clq/quick4j.git 拉取代码

黑色字体：jenkins 使用 maven 编译测试发布 web 工程到本地 /opt/jenkins_tomcat6/jobs/quick4j_demo/workspace/target/quick4j.war

蓝色字体：jenkins 使用 ssh 将本地打包好的 war 包传送到远程服务器的/opt/jenkins 中

红色字体：jenkins 远程执行远程服务器上的脚本，检查 tomcat 是否开启，如状态为开启，则关闭后，删除 tomcat 中原来的 web 项目包，再从/opt/jenkins 中拷贝刚打包好的 web 项目 war 包到 tomcat 中，启动 tomcat。

```
Started by user admin
Building in workspace /opt/jenkins_tomcat6/jobs/quick4j_demo/workspace
#jenkins 本机的工作目录
 > /usr/bin/git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > /usr/bin/git config remote.origin.url http://192.168.100.200/clq/quick4j.git #
timeout=10
Fetching upstream changes from http://192.168.100.200/clq/quick4j.git
 > /usr/bin/git --version # timeout=10
using .gitcredentials to set credentials
 > /usr/bin/git config --local credential.username clq # timeout=10
 > /usr/bin/git config --local credential.helper store
--file=/opt/tomcat6/apache-tomcat-6.0.45/temp/git4880643529884290174.credentials #
timeout=10
 > /usr/bin/git -c core.askpass=true fetch --tags --progress
http://192.168.100.200/clq/quick4j.git +refs/heads/*:refs/remotes/origin/*
 > /usr/bin/git config --local --remove-section credential # timeout=10
 > /usr/bin/git rev-parse refs/remotes/origin/master^{commit} # timeout=10
 > /usr/bin/git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision b123b099e8e72d7e467ab780ae28726a1c866797
(refs/remotes/origin/master)
 > /usr/bin/git config core.sparsecheckout # timeout=10
 > /usr/bin/git checkout -f b123b099e8e72d7e467ab780ae28726a1c866797
 > /usr/bin/git rev-list b123b099e8e72d7e467ab780ae28726a1c866797 # timeout=10
[workspace] $ /opt/apache-maven-3.3.9/bin/mvn
-Dmaven.repo.local=/opt/jenkins_tomcat6/jobs/quick4j_demo/workspace/.repository clean
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building quick4j App 1.0.0
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ quick4j ---
[INFO] Deleting /opt/jenkins_tomcat6/jobs/quick4j_demo/workspace/target
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

```
[INFO] Total time: 0.299 s
[INFO] Finished at: 2016-03-29T13:18:06+08:00
[INFO] Final Memory: 7M/238M
[INFO] ------------------------------------------------------------------------
[workspace] $ /bin/sh -xe
/opt/tomcat6/apache-tomcat-6.0.45/temp/hudson1223364716445353332.sh
Parsing POMs
[workspace] $ /opt/JDK/bin/java -cp
/opt/jenkins_tomcat6/plugins/maven-plugin/WEB-INF/lib/maven31-agent-1.5.jar:/opt/apache
-maven-3.3.9
/boot/plexus-classworlds-2.5.2.jar:/opt/apache-maven-3.3.9/conf/logging
jenkins.maven3.agent.Maven31Main
/opt/apache-maven-3.3.9
/opt/tomcat6/apache-tomcat-6.0.45/webapps/jenkins/WEB-INF/lib/remoting-2.51.jar
/opt/jenkins_tomcat6/plugins/maven-plugin/WEB-INF/lib/maven31-interceptor-1.5.jar
/opt/jenkins_tomcat6/plugins/maven-plugin/WEB-INF/lib/maven3-interceptor-commons-1.5.ja
r 36966
<===[JENKINS REMOTING CAPACITY]===>channel started
Executing Maven:  -B -f /opt/jenkins_tomcat6/jobs/quick4j_demo/workspace/pom.xml
-Dmaven.repo.local=/opt/jenkins_tomcat6/maven-repositories/1 -s
/opt/apache-maven-3.3.9/conf/settings.xml
-gs /opt/apache-maven-3.3.9/conf/settings.xml clean package
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building quick4j App 1.0.0
[INFO] ------------------------------------------------------------------------

[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ quick4j ---
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ quick4j ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 9 resources
[INFO] Copying 3 resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ quick4j ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 62 source files to
/opt/jenkins_tomcat6/jobs/quick4j_demo/workspace/target/classes

[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ quick4j ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 3 resources
```

```
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ quick4j ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 5 source files to
/opt/jenkins_tomcat6/jobs/quick4j_demo/workspace/target/test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.18.1:test (default-test) @ quick4j ---
[INFO] Tests are skipped.
[INFO]
[INFO] --- maven-war-plugin:2.2:war (default-war) @ quick4j ---
[INFO] Packaging webapp
[INFO] Assembling webapp [quick4j] in
[/opt/jenkins_tomcat6/jobs/quick4j_demo/workspace/target/quick4j]
[INFO] Processing war project
[INFO] Copying webapp resources
[/opt/jenkins_tomcat6/jobs/quick4j_demo/workspace/src/main/webapp]
[INFO] Webapp assembled in [383 msecs]

[INFO] Building war: /opt/jenkins_tomcat6/jobs/quick4j_demo/workspace/target/quick4j.war
#发布构建好的 web 工程到本机工作目录下

[INFO] WEB-INF/web.xml already added, skipping

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 6.611 s
[INFO] Finished at: 2016-03-29T13:18:15+08:00
[INFO] Final Memory: 28M/438M
[INFO] ------------------------------------------------------------------------
[JENKINS] Archiving /opt/jenkins_tomcat6/jobs/quick4j_demo/workspace/pom.xml to
com.eliteams/quick4j/1.0.0/quick4j-1.0.0.pom
[JENKINS] Archiving /opt/jenkins_tomcat6/jobs/quick4j_demo/workspace/target/quick4j.war
to com.eliteams/quick4j/1.0.0/quick4j-1.0.0.war
SSH: Connecting from host [Local-Machine-1]
SSH: Connecting with configuration [192.168.100.118-ssh] ...
SSH: Creating session: username [root], hostname [192.168.100.118], port [22]
SSH: Connecting session ...
channel stopped
SSH: Connected
SSH: Opening SFTP channel ...
SSH: SFTP channel open
SSH: Connecting SFTP channel ...
SSH: Connected
```

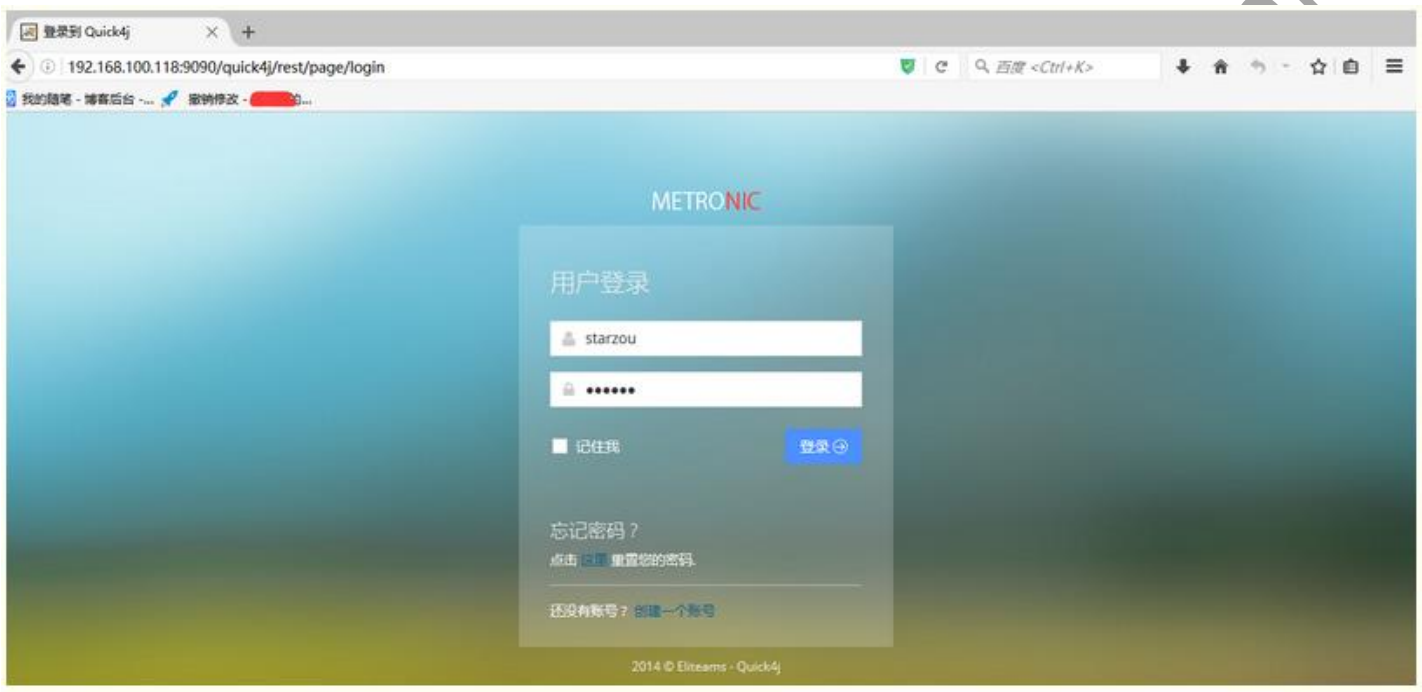
```
SSH: cd [/opt/jenkins]
SSH: OK
SSH: cd [/opt/jenkins]
SSH: OK
SSH: put [quick4j.war]

SSH: OK
SSH: Opening exec channel ...
SSH: EXEC: channel open
SSH: EXEC: STDOUT/STDERR from command [sh /opt/auto_deploy.sh] ...
SSH: EXEC: connected
**********************************  [2016-03-29]13:18:23
**************************************
updating server  environment start

updating server  environment  end
check tomcat status...
tomcat is running....port is 9090
shutdown tomcat.....
>>>>>>>shutdown tomcat begin<<<<<<<<
Using CATALINA_BASE:   /opt/tomcat6/apache-tomcat-6.0.45
Using CATALINA_HOME:   /opt/tomcat6/apache-tomcat-6.0.45
Using CATALINA_TMPDIR: /opt/tomcat6/apache-tomcat-6.0.45/temp
Using JRE_HOME:        /app/java/jdk1.8.0_11/jre
Using CLASSPATH:       /opt/tomcat6/apache-tomcat-6.0.45/bin/bootstrap.jar

>>>>>>>shutdown tomcat end <<<<<<<<

--------------  begin  transfer  war package to tomcat webapps -------------------
Find /opt/jenkins exist war package quick4j.war
deleteing old  package quick4j.war in /opt/tomcat6/apache-tomcat-6.0.45/webapps/
start  transfer quick4j.war to /opt/tomcat6/apache-tomcat-6.0.45/webapps/

--------------  transfer  war package to tomcat webapps  end -------------------
 >>>>>>>  rebooting  tomcat begin <<<<<<<<
Using CATALINA_BASE:   /opt/tomcat6/apache-tomcat-6.0.45
Using CATALINA_HOME:   /opt/tomcat6/apache-tomcat-6.0.45
Using CATALINA_TMPDIR: /opt/tomcat6/apache-tomcat-6.0.45/temp
Using JRE_HOME:        /app/java/jdk1.8.0_11/jre
Using CLASSPATH:       /opt/tomcat6/apache-tomcat-6.0.45/bin/bootstrap.jar
 >>>>>>>  rebooting  tomcat end <<<<<<<<
the log you can read in canalina.out
************************ deploy war package into container Successlly
**************************************
SSH: EXEC: completed after 11,406 ms
```

```
SSH: Disconnecting configuration [192.168.100.118-ssh] ...
SSH: Transferred 1 file(s)
Finished: SUCCESS
```

三、浏览器验证是否正确发布：

jenkins 主机 ip：192.168.100.119

远程部署主机 IP：192.168.100.118



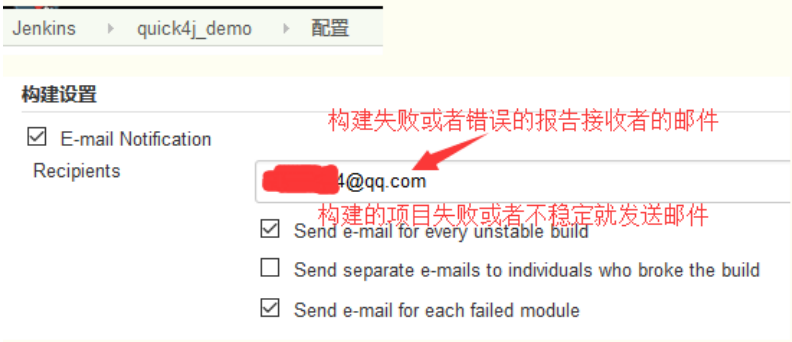
发布成功

[原]Jenkins(八)---jenkins 构建项目报错时发送错误报告邮件

一、修改系统 （email 引擎设置）

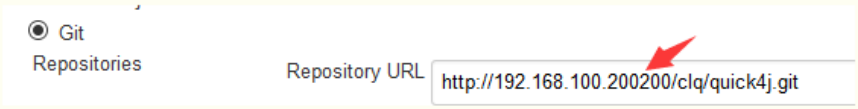
二、编辑错误报告发送者：

三、在项目配置中设置接收邮件的邮箱（多个接收人时用英文空格分隔）



四、测试是否可以发邮件：

故意将git远程仓库地址设置错误，在构建的时候就不能正确构建



点击"立即构建"，如下图：

```
Caused by: hudson.plugins.git.GitException: Command "/usr/bin/git -c core.askpass=true fetch --tags --progress
http://192.168.100.200200/clq/quick4j.git +refs/heads/*:refs/remotes/origin/*" returned status code 128:
stdout:
stderr: fatal: unable to access 'http://192.168.100.200200/clq/quick4j.git/': Could not resolve host: 192.168.100.2002(

        at org.jenkinsci.plugins.gitclient.CliGitAPIImpl.launchCommandIn(CliGitAPIImpl.java:1719)
        at org.jenkinsci.plugins.gitclient.CliGitAPIImpl.launchCommandWithCredentials(CliGitAPIImpl.java:1463)
        at org.jenkinsci.plugins.gitclient.CliGitAPIImpl.access$300(CliGitAPIImpl.java:63)
        at org.jenkinsci.plugins.gitclient.CliGitAPIImpl$1.execute(CliGitAPIImpl.java:314)
        at hudson.plugins.git.GitSCM.fetchFrom(GitSCM.java:761)
        ... 11 more
ERROR: null
Sending e-mails to:  @qq.com
Finished: FAILURE
```

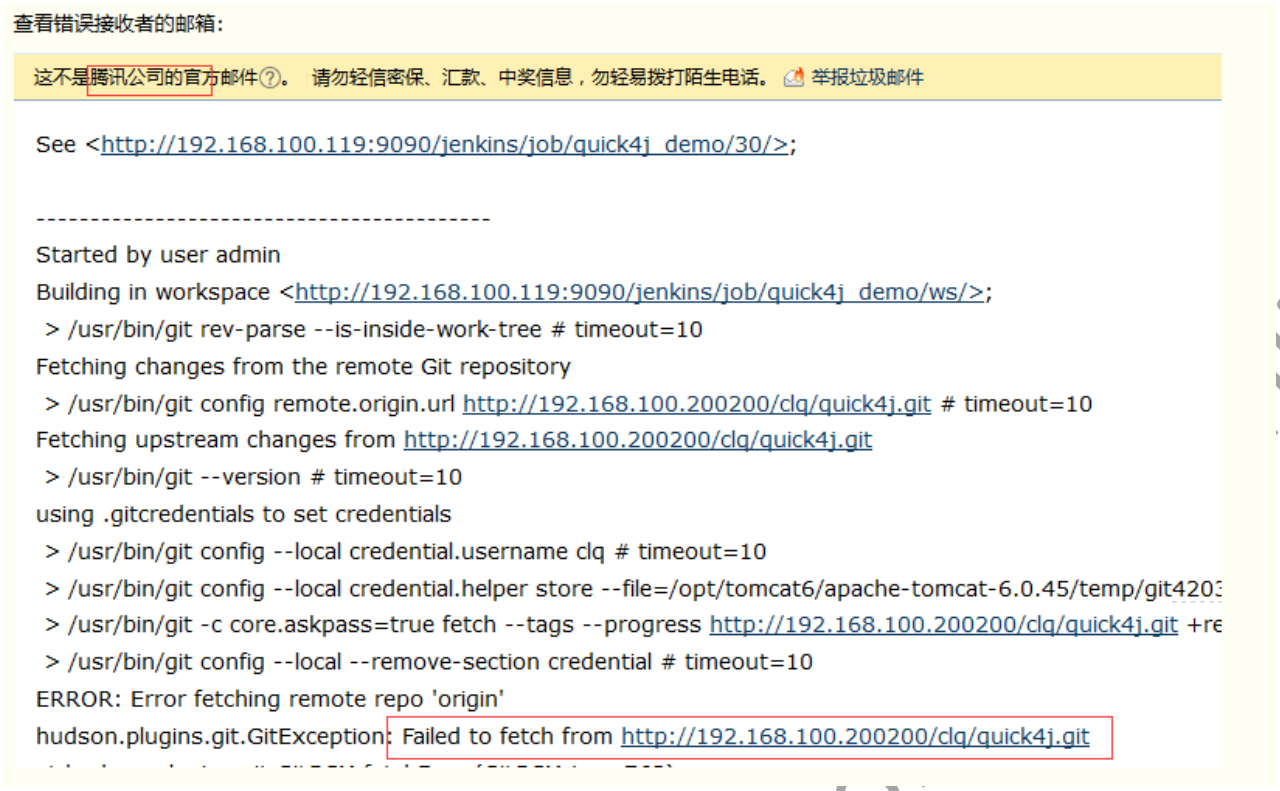
导致不能构建成功的错误git地址

发送这个构建错误的信息到错误邮件接收者中

构建失败

查看错误接收者的邮箱：

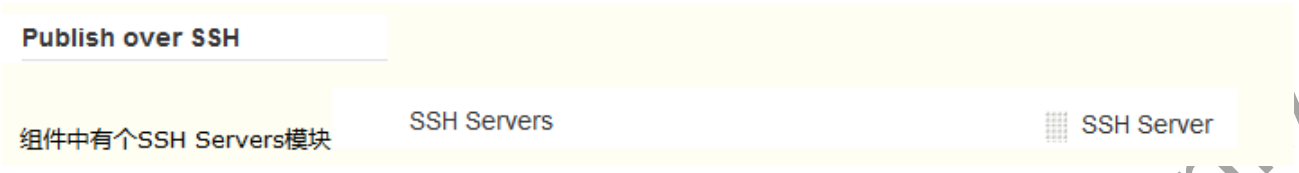
查看错误接收者的邮箱：



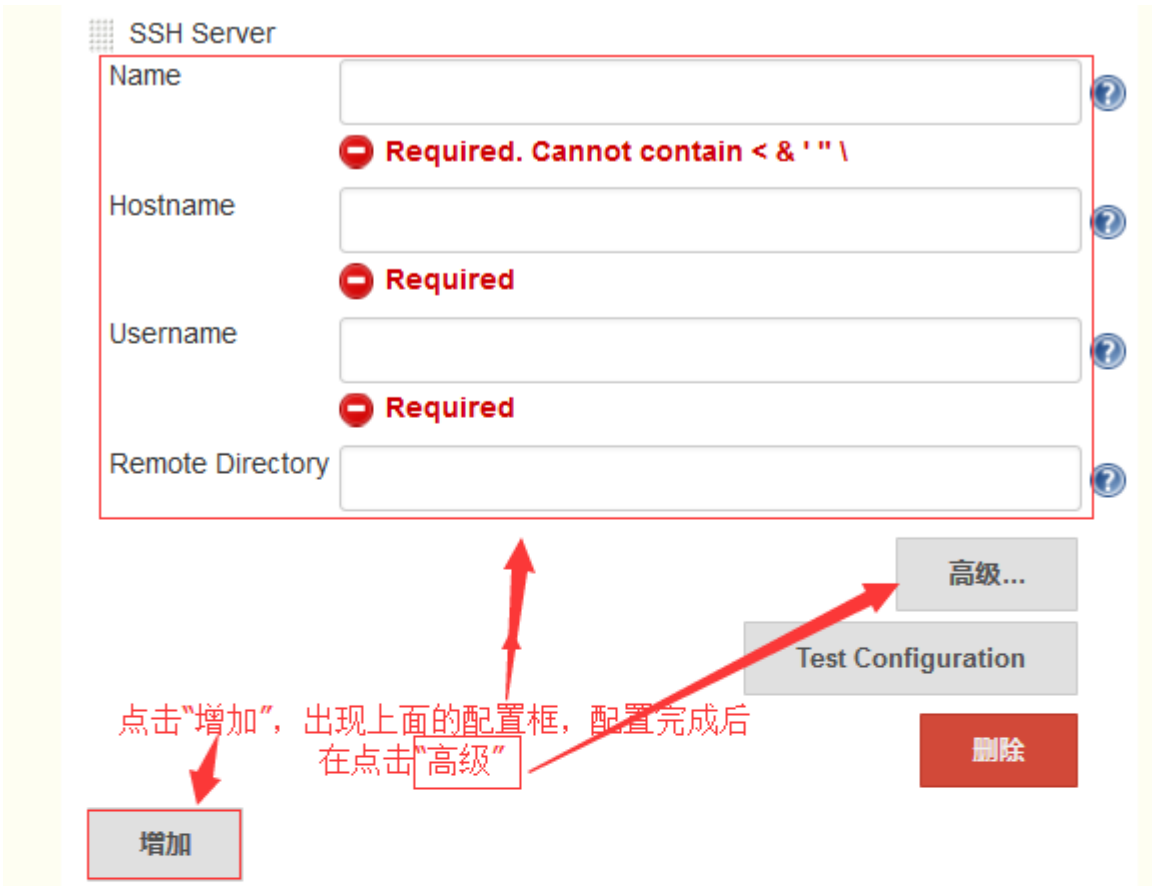
[原]Jenkins(九)---jenkins 分别发布多个项目到多个远程主机

一、配置远程主机

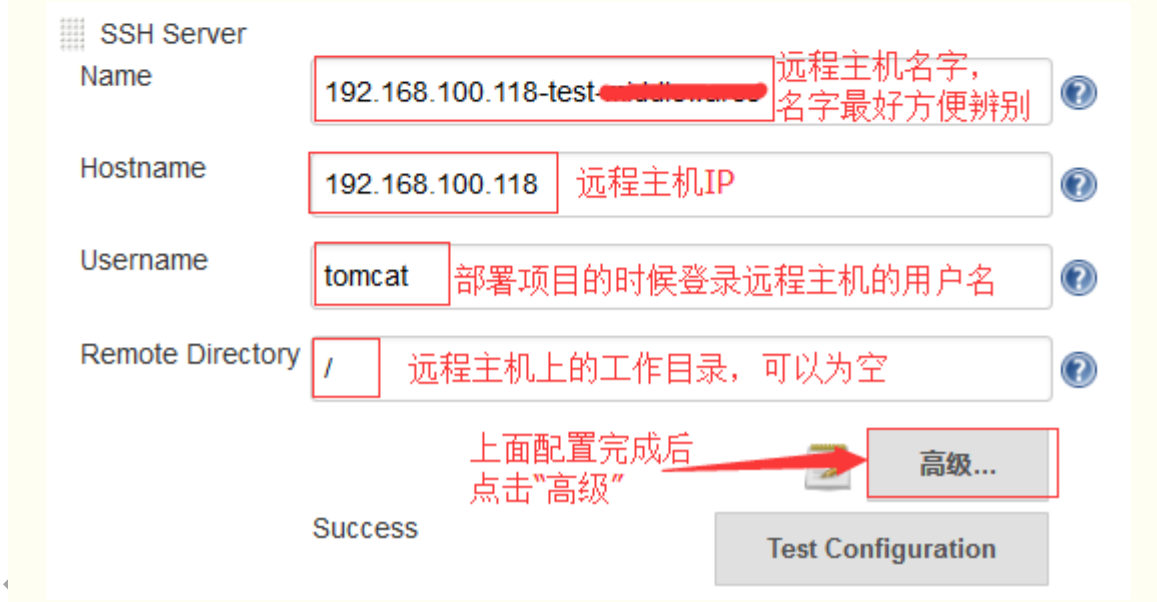
点击“系统管理”---“系统设置”----找到“Publish over SSH”组件

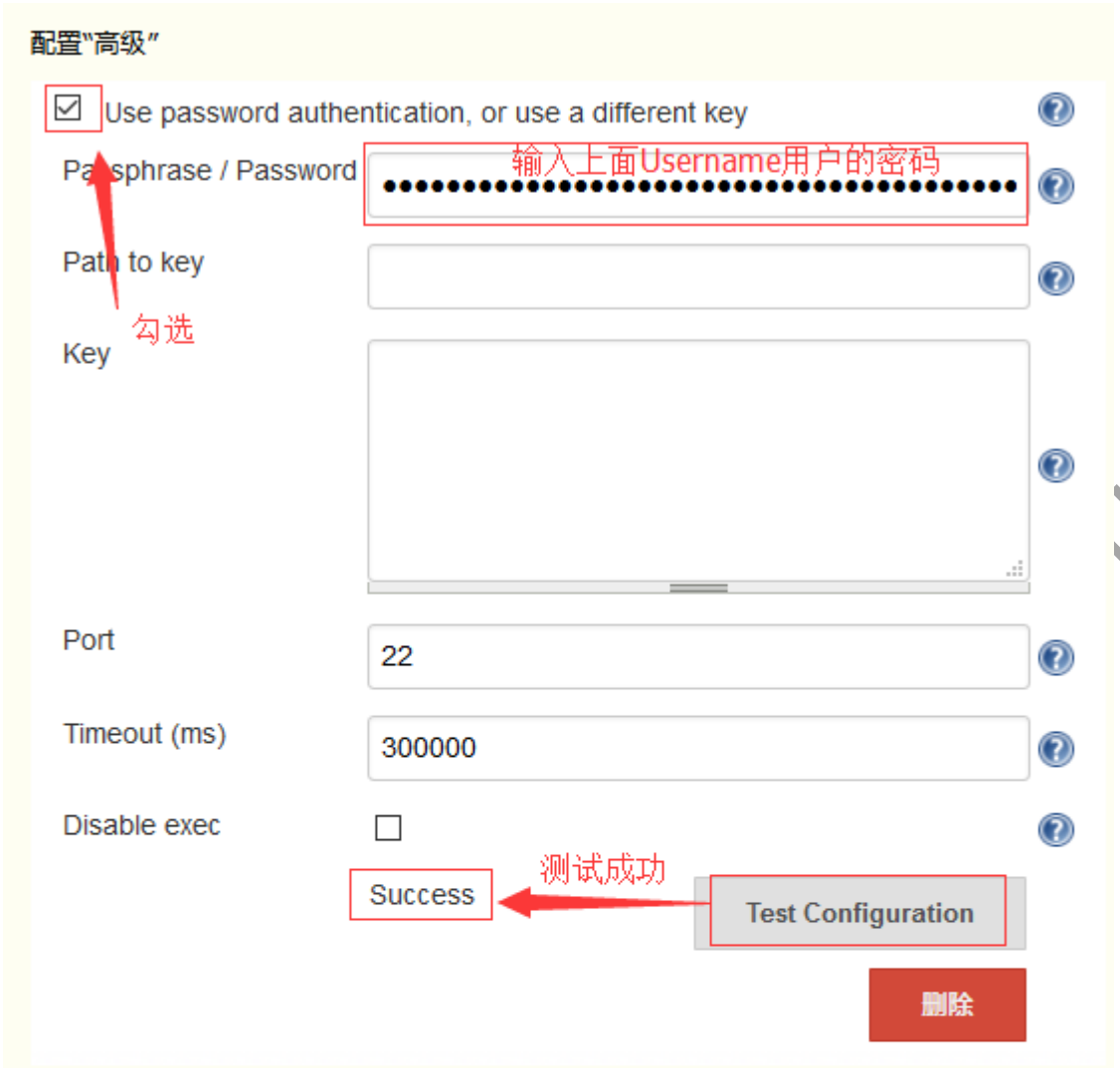


点击“增加”


SSH Server
Name
Required. Cannot contain < & ' " \
Hostname
Required
Username
Required
Remote Directory
高级...
Test Configuration
删除
点击"增加"，出现上面的配置框，配置完成后
在点击"高级"
增加

点击增加后，


SSH Server
Name
192.168.100.118-test
远程主机名字，
名字最好方便辨别
Hostname
192.168.100.118
远程主机IP
Username
tomcat
部署项目的时候登录远程主机的用户名
Remote Directory
/
远程主机上的工作目录，可以为空
上面配置完成后
点击"高级"
高级...
Success
Test Configuration



二、配置需要部署的项目

请点击参考查看：jenkins(五)---jenkins添加项目

查看过程中最重要的一步是选择你要部署到远程主机的名字：如下图

[原]Jenkins(十)---jenkins 注册管理员 admin 并赋所有权限给 admin

一、使用匿名用户（Anonymous）打开用户注册功能

jenkins 刚开始是没有用户的，之所以能使用 jenkins 的功能，是因为 jenkins 启动后会创建了一个匿名用户（Anonymous），你登录 jenkins 的时候使用的是 Anonymous 用户

Anonymous 用户不需要登录就能进行所有操作。当创建新用户（管理员或者普通用户）的时候，Anonymous 自动失效，不再具有之前的修改和更新功能，但保留查看功能。

下面是用户原理图：

开始匿名用户（Anonymous）打开用户注册功能：

第一次需要配置用户的时候需要勾选"允许用户注册"前面的勾，是为了要创建admin用户。

当第五步保存完之后，就已经使用匿名用户（Anonymous）打开用户注册功能了

注册之后会在右上角发现有 “登录和注册” 字样

二、注册添加管理员用户并赋予 administrator 权限

1.点击“注册”，进行增加管理员操作：

使用刚才注册的管理员用户登录进行登录

登录后，右上角显示如下：

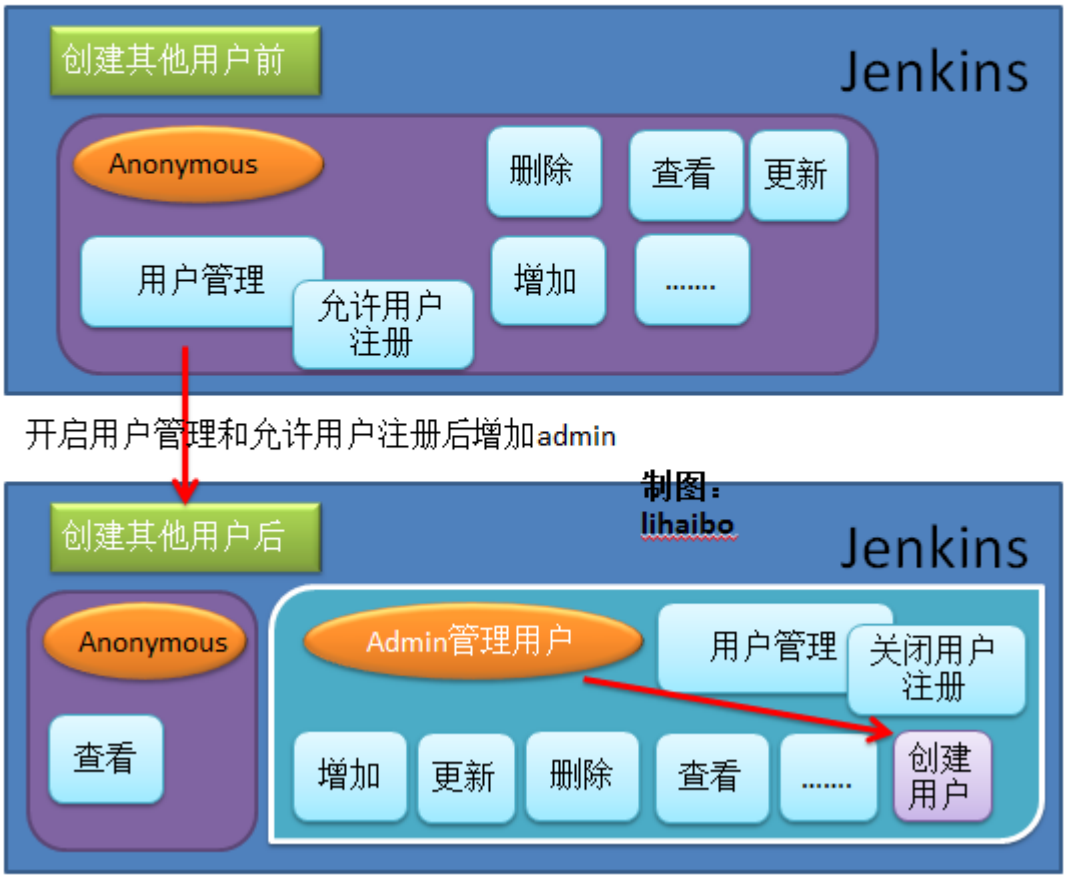
2.创建好 admin 用户以后，取消用户通过注册来创建用户，即：将用户的管理权限全部赋给 admin 管理员：“系统管理”------“Configure Global Security”-------取消“允许用户注册”前面的勾

右上角的注册字样就消失了

这个时候，Anonymous只有查看的权限

关闭用户注册的用意是将用户管理归纳到admin来进行管理：创建，赋予权限，删除。

如下图：



3.给admin管理员用户分配 administrator角色

点击 “系统管理”------“Configure Global Security”--------“安全矩阵”

应用保存。

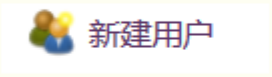
[原]Jenkins(十一)---jenkins 使用管理员 admin 创建用户和分配权限

一、利用创建的admin管理员添加用户（其他管理员或者一般用户）

1.使用第一步（二、注册添加管理员用户）中的管理员admin用户登录

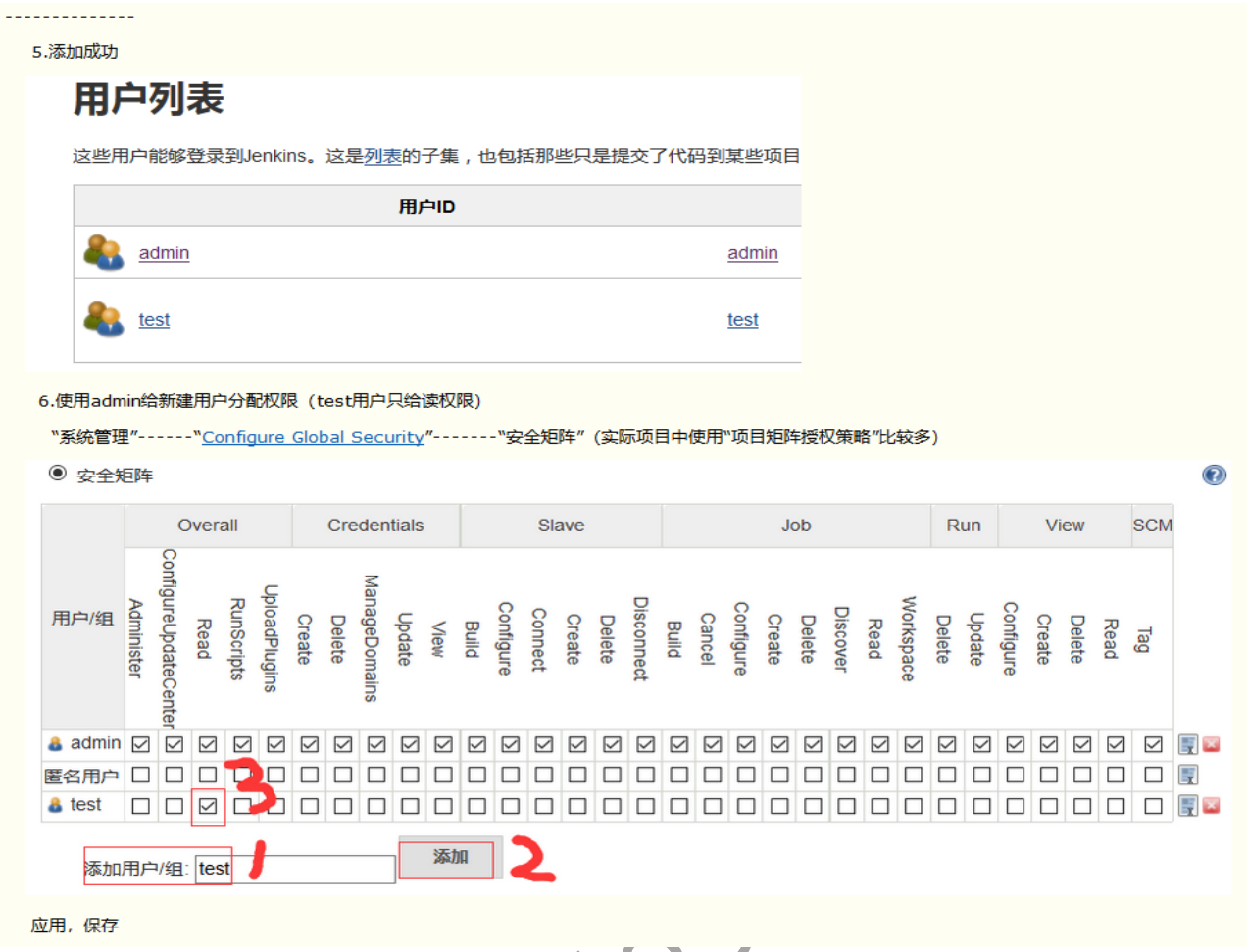
2.点击"系统管理"-----"管理用户"

3.点击左边栏"新建用户"



4.配置新加用户信息

5.添加成功



6.使用admin给新建用户分配权限（test用户只给读权限）

"系统管理"------"Configure Global Security"--------"安全矩阵"（实际项目中使用"项目矩阵授权策略"比较多）

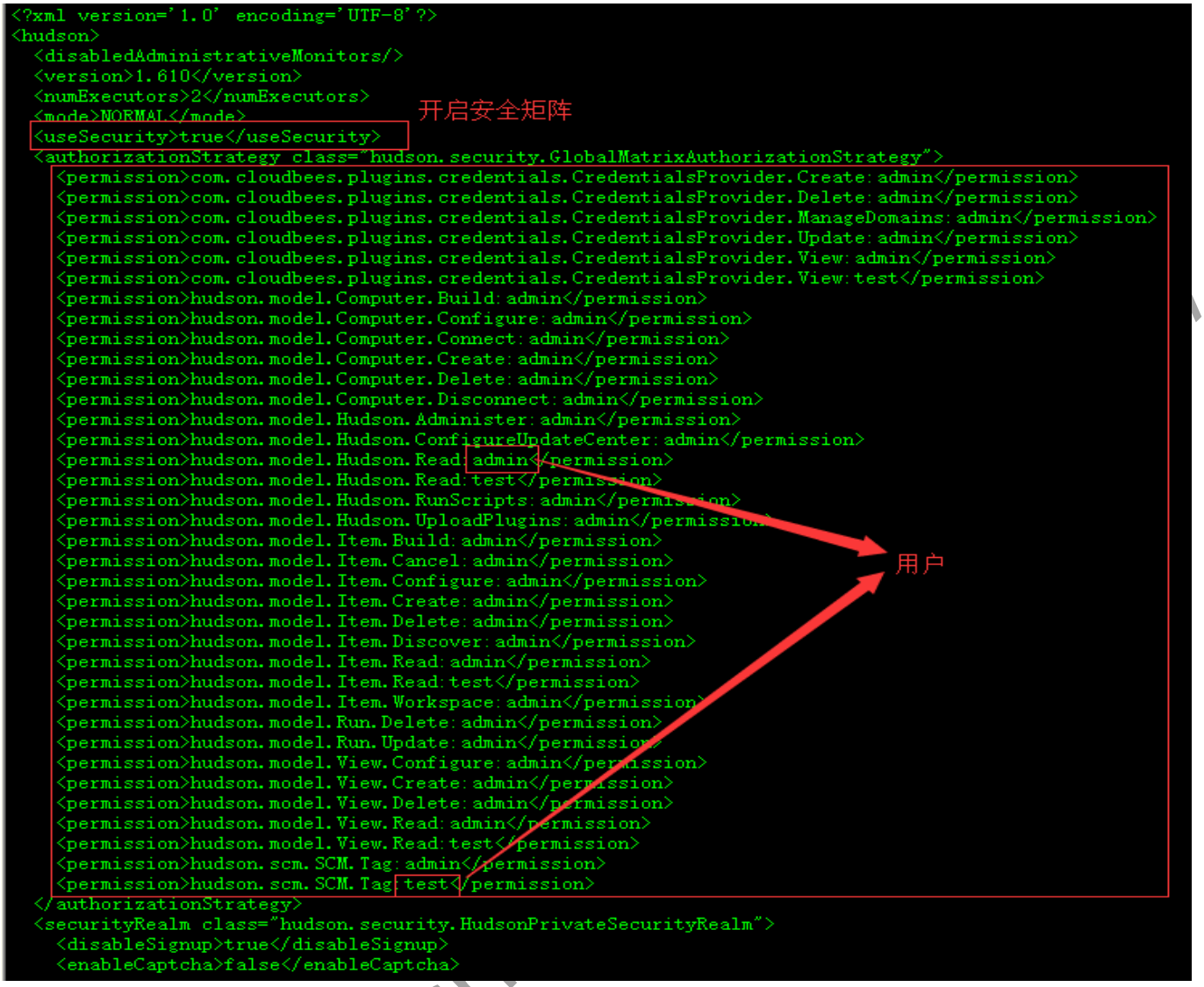
应用，保存

这个时候 JENKINS_HOME 中的 config.xml 是这样的

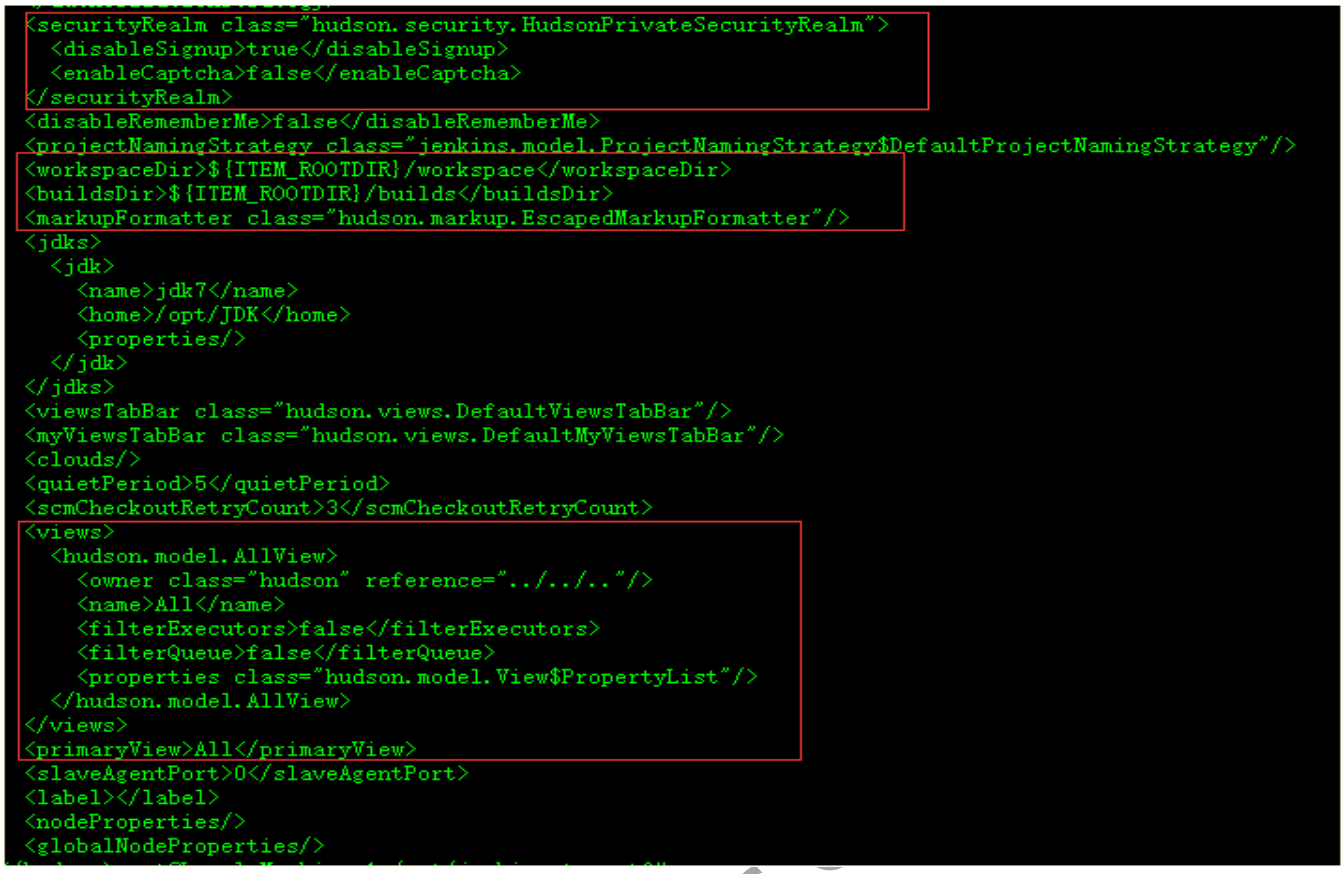
useSecurity 定义是否启用安全矩阵

Authorization Strategy，定义用户有哪些权限

Security Realm，决定是否使用用户名和密码登陆



下图中红框表示比较重要的信息，参考看看即可。



```
<securityRealm class="hudson.security.HudsonPrivateSecurityRealm">
  <disableSignup>true</disableSignup>
  <enableCaptcha>false</enableCaptcha>
</securityRealm>
<disableRememberMe>false</disableRememberMe>
<projectNamingStrategy class="jenkins.model.ProjectNamingStrategy$DefaultProjectNamingStrategy"/>
<workspaceDir>${ITEM_ROOTDIR}/workspace</workspaceDir>
<buildsDir>${ITEM_ROOTDIR}/builds</buildsDir>
<markupFormatter class="hudson.markup.EscapedMarkupFormatter"/>
<jdks>
  <jdk>
    <name>jdk7</name>
    <home>/opt/JDK</home>
    <properties/>
  </jdk>
</jdks>
<viewsTabBar class="hudson.views.DefaultViewsTabBar"/>
<myViewsTabBar class="hudson.views.DefaultMyViewsTabBar"/>
<clouds/>
<quietPeriod>5</quietPeriod>
<scmCheckoutRetryCount>3</scmCheckoutRetryCount>
<views>
  <hudson.model.AllView>
    <owner class="hudson" reference="../../.."/>
    <name>All</name>
    <filterExecutors>false</filterExecutors>
    <filterQueue>false</filterQueue>
    <properties class="hudson.model.View$PropertyList"/>
  </hudson.model.AllView>
</views>
<primaryView>All</primaryView>
<slaveAgentPort>0</slaveAgentPort>
<label></label>
<nodeProperties/>
<globalNodeProperties/>
```

[原]Jenkins(十二)---jenkins 管理员用户无法登陆解决办法 Access Denied

一、问题出现的原因： 如下图，在添加了admin用户之后，没有在安全矩阵中对admin进行赋予权限就退出了jenkins

安全矩阵

用户/组	Overall					Credentials					Slave						Job								Run		View				SCM
	Administer	ConfigureUpdateCenter	Read	RunScripts	UploadPlugins	Create	Delete	ManageDomains	Update	View	Build	Configure	Connect	Create	Delete	Disconnect	Build	Cancel	Configure	Create	Delete	Discover	Read	Workspace	Delete	Update	Configure	Create	Delete	Read	Tag
admin	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

然后就悲剧了，死活登陆不了，出现如下的界面

2.解决方法：

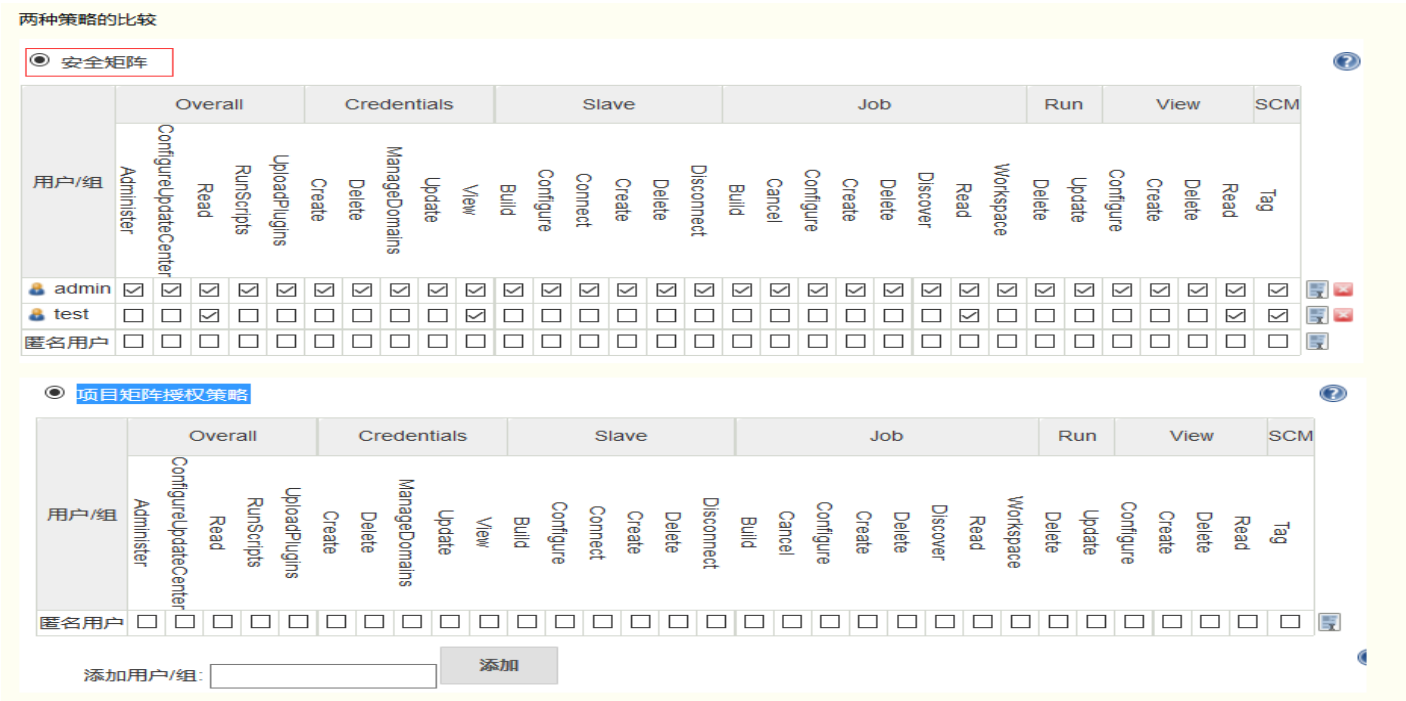
在jenkins的工作家目录JENKINS_HOME中有个config.xml文件，将<useSecurity>的键值由true改为false

```
[redacted]/opt/jenkins_tomcat6# vim config.xml

<?xml version='1.0' encoding='UTF-8'?>
<hudson>
  <disabledAdministrativeMonitors/>
  <version>1.610</version>
  <numExecutors>2</numExecutors>
<?xml version='1.0' encoding='UTF-8'?>
<hudson>
  <disabledAdministrativeMonitors/>
  <version>1.610</version>
  <numExecutors>2</numExecutors>
  <mode>NORMAL</mode>
  <useSecurity>false</useSecurity>
  <authorizationStrategy class="hudson.security.GlobalMatrixAuthorizationStrategy"/>
  <securityRealm class="hudson.security.HudsonPrivateSecurityRealm">
    <disableSignup>true</disableSignup>
    <enableCaptcha>false</enableCaptcha>
  </securityRealm>
  <disableRememberMe>false</disableRememberMe>
  <projectNamingStrategy class="jenkins.model.ProjectNamingStrategy$DefaultProjectNamingStrategy"
  <workspaceDir>${ITEM_ROOTDIR}/workspace</workspaceDir>
  <buildsDir>${ITEM_ROOTDIR}/builds</buildsDir>
  <markupFormatter class="hudson.markup.EscapedMarkupFormatter"/>
  <jdks>
    <jdk>
      <name>jdk7</name>
      <home>/opt/JDK</home>
      <properties/>
    </jdk>
  </jdks>
```

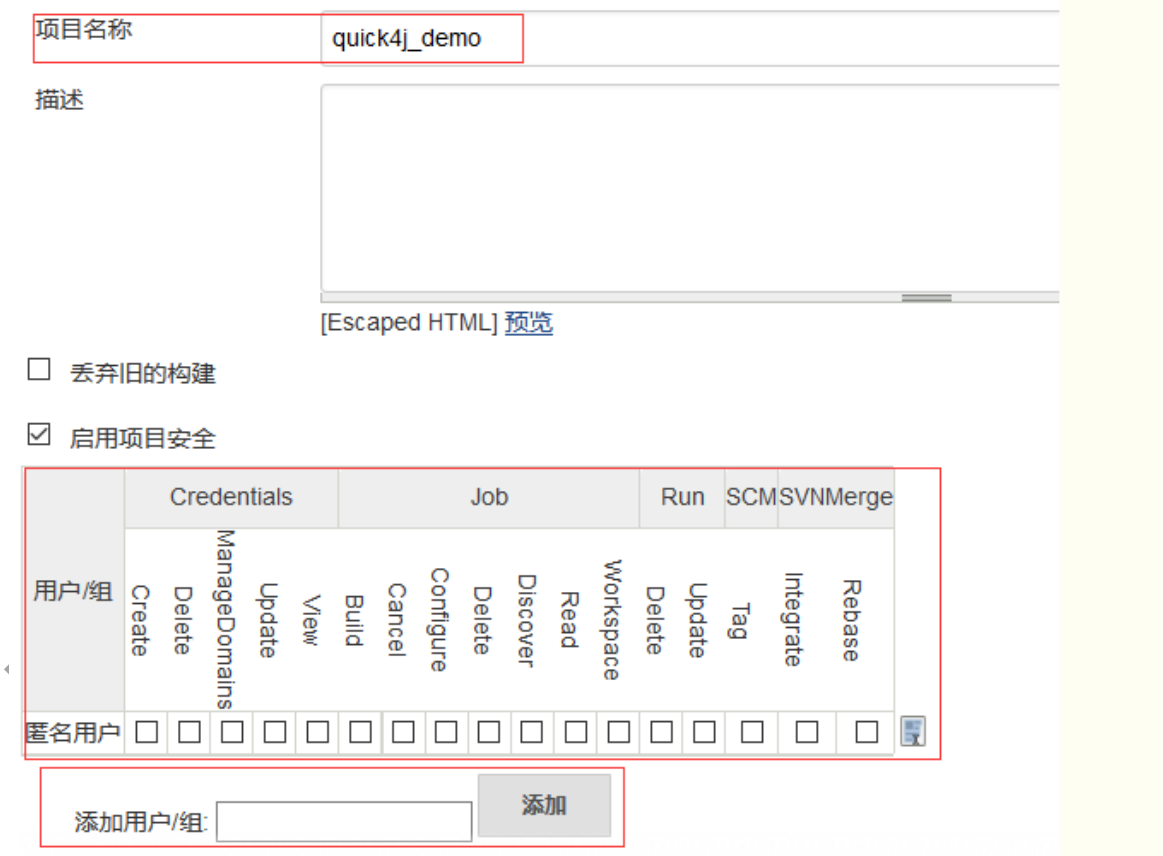
将True改为false

重启jenkins所在的tomcat，再重新使用之前建立的admin登陆,成功

[原]Jenkins(十三)---jenkins 用户权限管理



如上面两个图，安全矩阵和项目矩阵授权策略功能几乎完全一样，但是**项目矩阵授权策略**支持在 Job（项目）的配置页面再次配置授权策略。



这种策略在工作中用得较多，比如针对不同的项目选择不同的用户具有不同权限。
各种权限如下(在配置页面将鼠标放到该权限上即可查看帮助)：

Overall(全局)					Credentials(凭证)					Slave(节点)						Job(任务)								View(视图)			
Administer	Read	RunScripts	UploadPlugins	ConfigureUpdateCenter	Create	Update	View	Delete	ManageDomains	Configure	Delete	Create	Disconnect	Connect	Build	Create	Delete	Configure	Read	Discover	Build	Workspace	Cancel	Create	Delete	Configure	Read
管理员(最大)	阅读	运行脚本	升级插件	配置升级中心	创建	更新	查看	删除	管理域	配置	删除	创建	断开连接	连接	构建	创建	删除	配置	阅读	重定向	构建	查看工作区	取消构建	创建	删除	配置	阅读

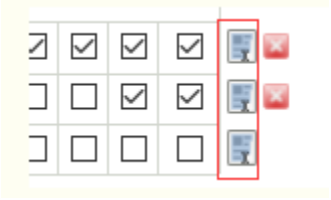
最大的权限是 Overall 的 Administer，拥有该权限可以干任何事情。

最基本的权限是 Overall 的 Read，用户必须赋予阅读的权限，不然什么都看不到。

Job 的 Discover 权限是一个奇葩的权限，帮助说 Discover 比 Read 的级别更低。如果匿名用户(没有访问 job 的权限)直接访问一个 Job 的 Url 将重定向到登陆页面。(经测试，这个权限应该是被废弃了。)

Credentials 的 ManageDomains 这个权限没有看懂干嘛的，有懂的大家一起交流哈！

ps：如果有个用户被赋予了 Overall 的 Read，并没有被赋予 Job 的 Read 权限，那么该用户就无法访问 job。原因：没有权限。



上图中红叉为删除这一行，红框里面的按钮表示反选

下面是转载的相关文章内容

找到.jenkins/config.xml 文件：

替换为：

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。

2、如果要配置连接微软 ldap，需要安装 Active Directory plugin。

比如配置：

Domain Name: XXXX.net

Domain controller:192.168.0.112:3268

LDAP 全局目录：TCP 端口 3268 (如果 DC 保持着全局目录的操纵权)

3、默认匿名用户是可以查看所有项目的，就算配置了“登陆用户可以做任何事情”
如果想禁止匿名使用，可以使用“安全矩阵”，
选择安全矩阵后，就会出现“匿名用户”用户，全部去掉勾选，则无任何权限了。
其中 overall 中的 Administer 代表全部权限，可以设置为管理员.
权限配置：http://hi.baidu.com/nesaynever/blog/item/9f34a1c80a6454377d3e6f65.html

其中：Overall 是全局权限，slave 是集群权限，job,run,view,scm 是业务权限。
其中 overall 中的 read 要勾选，否则用户登陆后什么也看不到。
overall:
Administer：系统管理员权限
read:浏览框架

job:
read:查看 job
build:执行构建
cancel:取消构建

run:
Delete:删除某次构建
Update:编辑某次构建信息

SCM:
Tag:为某次构建在 svm 上打标签。

[原]Jenkins(十四)---jenkins 示例：admin 管理所有项目，新建用户只能看部分项目

一、使用 admin 能看到四个项目

二、新建用户分配权限

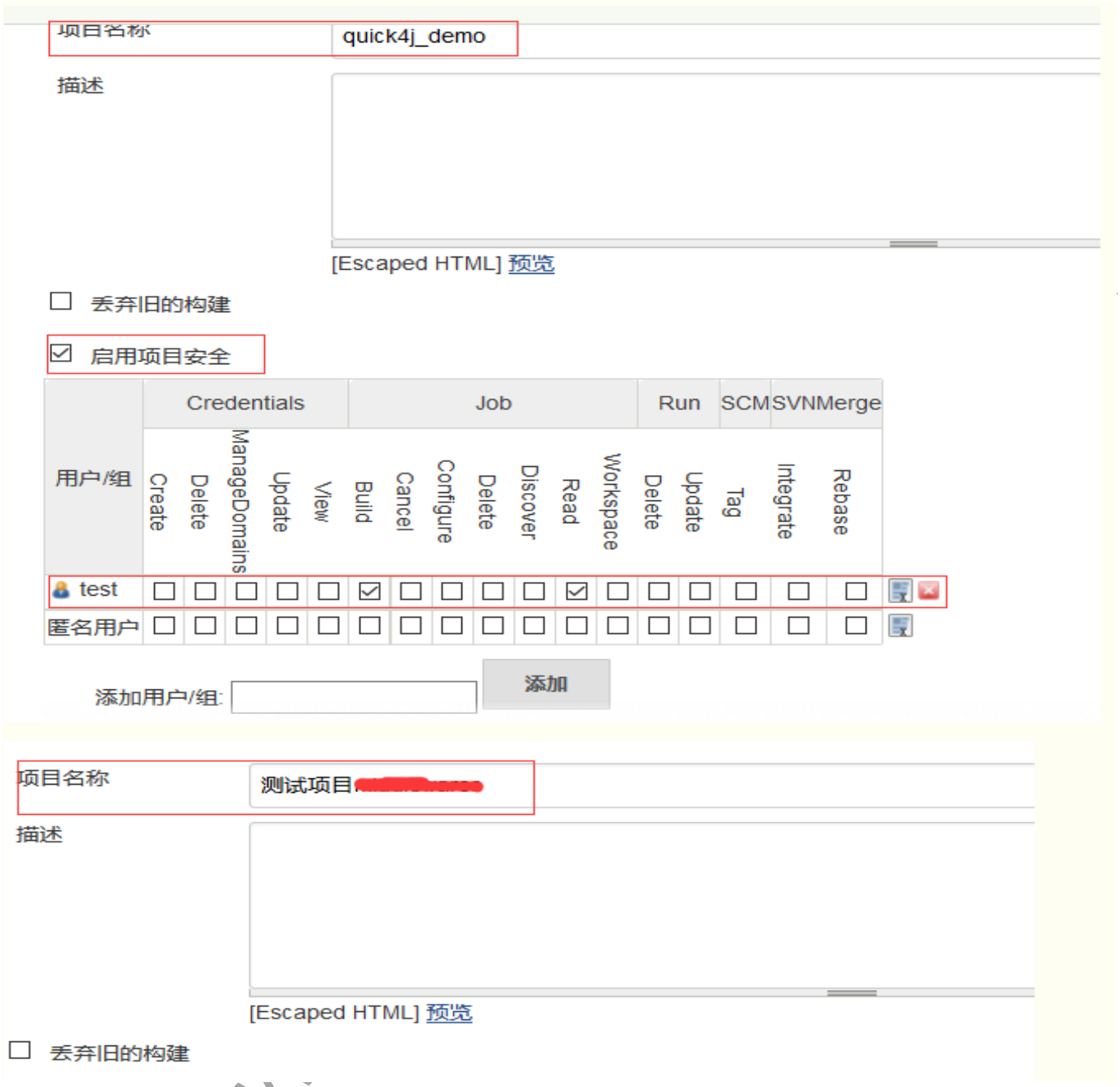
新建用户 test(如果不清楚，查看 Jenkins(十一)---jenkins 使用管理员 admin 创建用户和分配权限)中的新建用户

“系统管理”------“Configure Global Security”-------“项目矩阵授权策略”

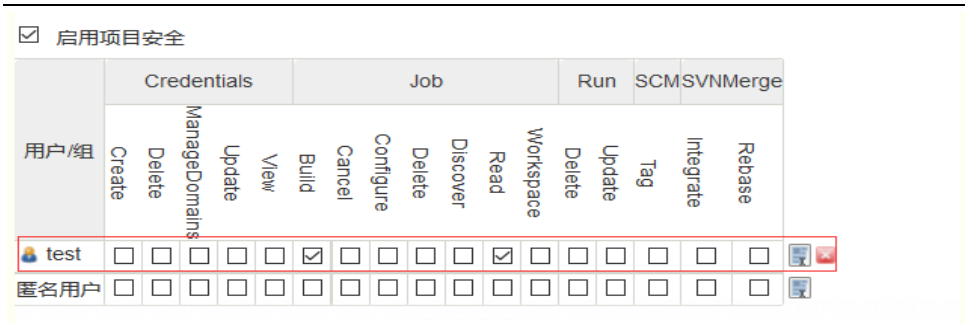
添加 test 用户到矩阵，并勾选"Overall" 中的 Read 选项，所有新建的用户都应该勾选此项，不然无法登录看到相关的项目界面。

由于使用了“项目矩阵授权策略 ”，所以在每个项目下就会有一个“启用项目安全”策略，每个项目都勾选后，如果需要哪个用户能看到该项目就将用户加入该项目。

下面给两个项目添加 test 用户：


项目名称
quick4j_demo
描述
[Escaped HTML] 预览
丢弃旧的构建
启用项目安全
用户/组
Credentials
Job
Run
SCM
SVNMerge
Create
Delete
ManageDomains
Update
View
Build
Cancel
Configure
Delete
Discover
Read
Workspace
Delete
Update
Tag
Integrate
Rebase
test
匿名用户
添加用户/组:
添加

项目名称
测试项目
描述
[Escaped HTML] 预览
丢弃旧的构建



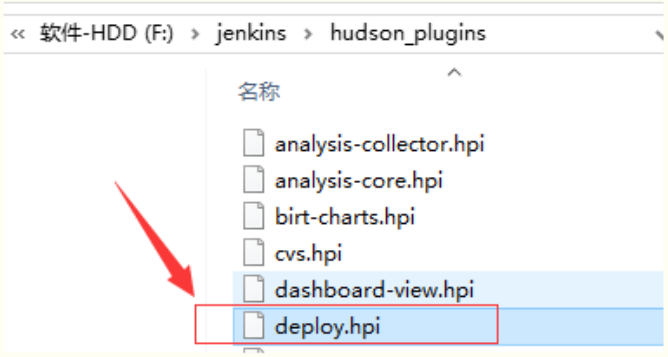
退出admin，使用test登录

就可以只看到两个项目了

[原]Jenkins(十五)---jenkins 插件之 deploy

一、deploy插件

在jenkins中发布application到远端有很多方法，除了publish over ssh外还有个插件deploy.hpi也可以实现。



链接: http://pan.baidu.com/s/1pLotSmR 密码: jzud

插件安装方法： 点击：Jenkins(三)---Jenkins 初始配置和插件配置

或手动转到 http://www.cnblogs.com/horizonli/p/5331970.html

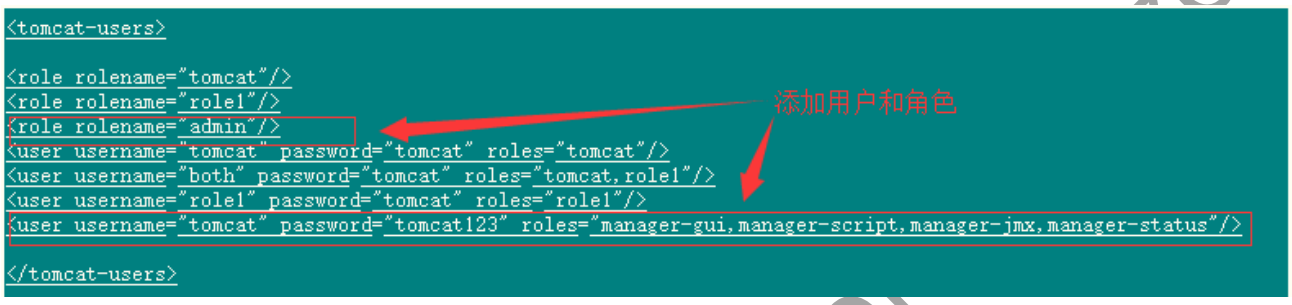
二、配置 启用插件前需要对远程端的容器进行环境初始化（这里以 tomcat 为例）

1.tomcat 能正常启动

2.tomcat 能在 web 界面使用用户登录到 app manager

1）配置增加 tomcat 用户：

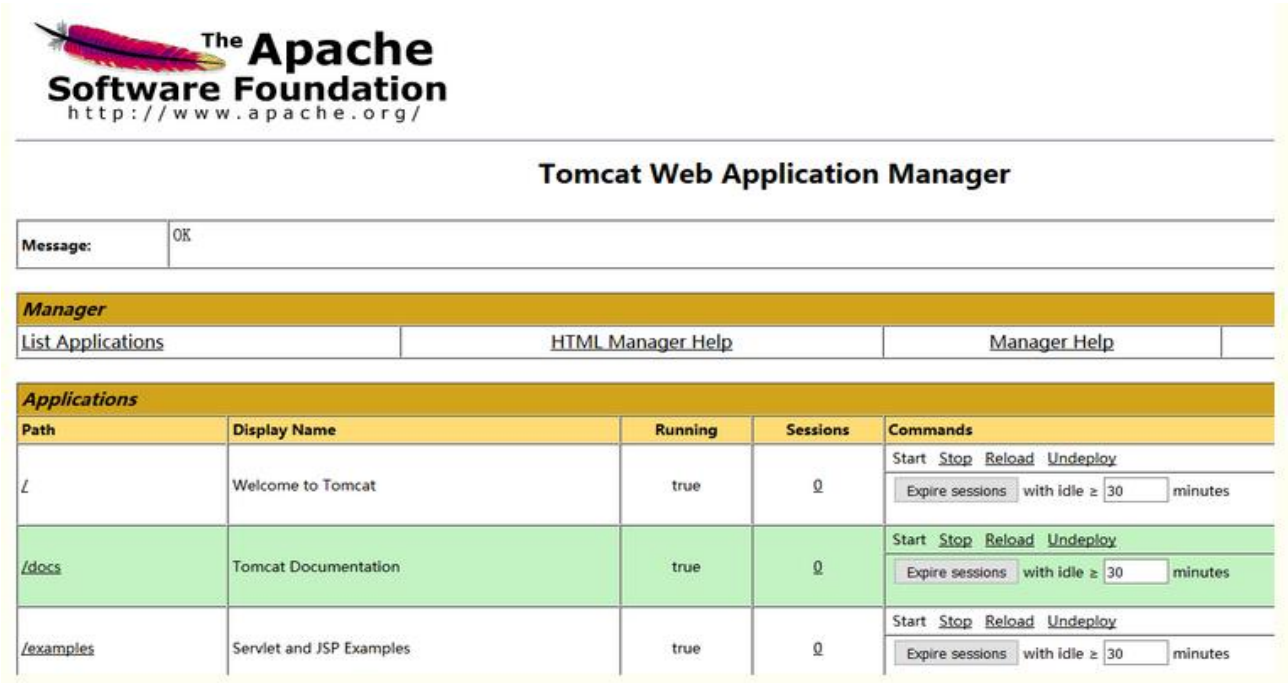
2）添加用户和角色




```
<tomcat-users>

<role rolename="tomcat"/>
<role rolename="role1"/>
<role rolename="admin"/>
<user username="tomcat" password="tomcat" roles="tomcat"/>
<user username="both" password="tomcat" roles="tomcat,role1"/>
<user username="role1" password="tomcat" roles="role1"/>
<user username="tomcat" password="tomcat123" roles="manager-gui,manager-script,manager-jmx,manager-status"/>

</tomcat-users>
```

3）登录 tomcat manager

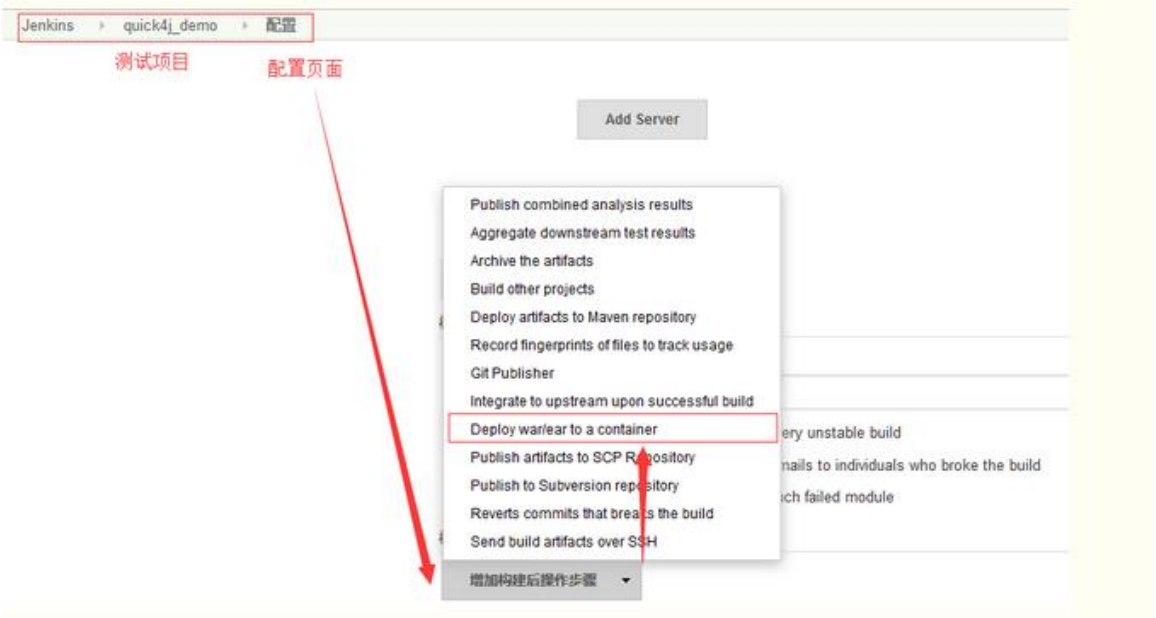
登录后如果能看到下面的界面表示配置正确



三、使用 deploy 插件发布 war 包到远程容器（这里以 tomcat 为例）

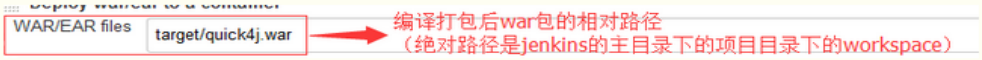
在系统设置中配置 maven/jdk/git 可以参考：Jenkins(三)---Jenkins 初始配置和插件配置 （除去七 SSH 这一节 ）

启动插件

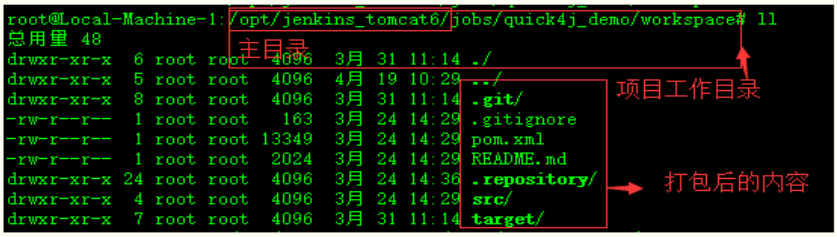


打开之后填写

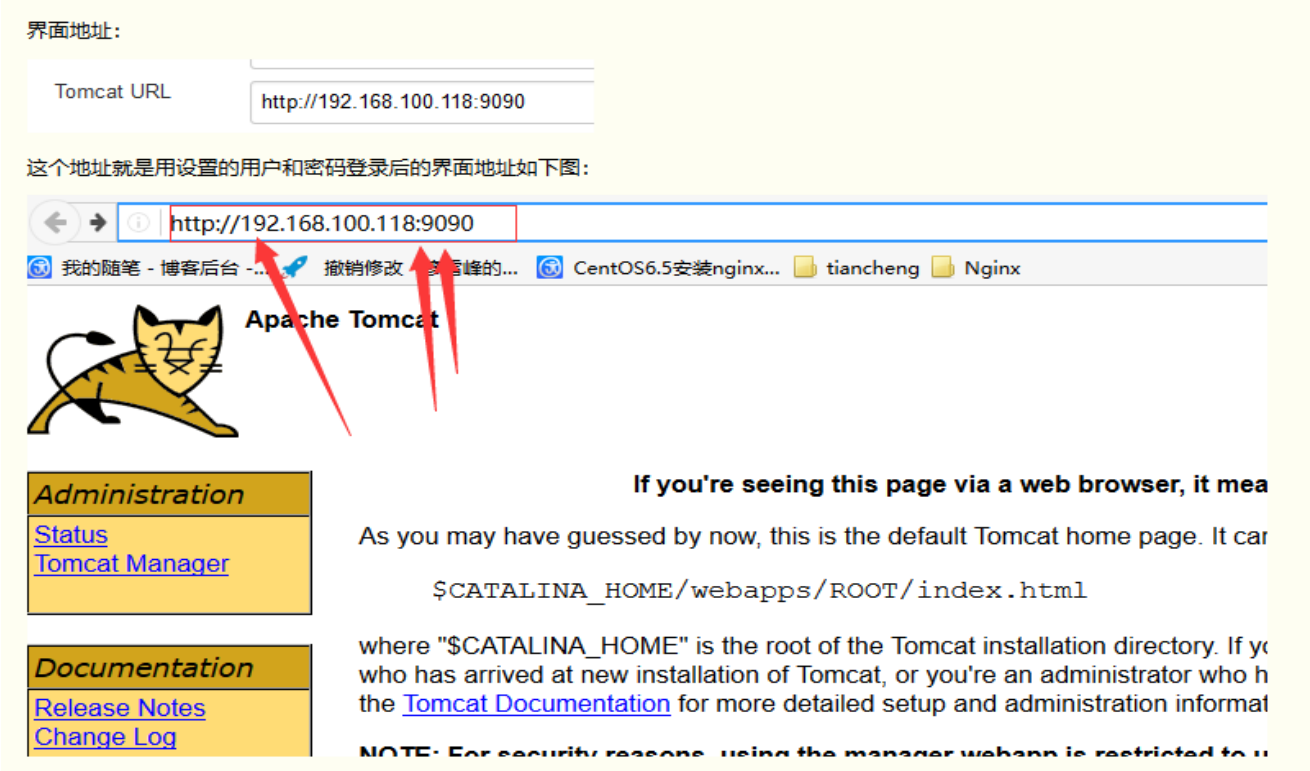
这里单独说下war包的路径和界面：



在后台看是这样的：



从前台看这样的：



四、利用 deploy 插件发布 application 到远程主机

1.首先确保远程主机上的 tomcat 能正常启动并能使用用户登录到 application 管理界面

2.再进行构建操作

3.如下图所示构建成功

INFO] Building war: /opt/jenkins_tomcat6/jobs/quick4j_demo/workspace/target/quick4j.war

[INFO] WEB-INF/web.xml already added, skipping

[INFO] --

[INFO] BUILD SUCCESS

[INFO] --

[INFO] Total time: 6.782 s

[INFO] Finished at: 2016-04-19T14:06:53+08:00

[INFO] Final Memory: 27M/438M

[INFO] --

[JENKINS] Archiving /opt/jenkins_tomcat6/jobs/quick4j_demo/workspace/pom.xml to com.eliteams/quick4j/1.0.0/quick4j-1.0.0.pom

[JENKINS] Archiving /opt/jenkins_tomcat6/jobs/quick4j_demo/workspace/target/quick4j.war to com.eliteams/quick4j/1.0.0/quick4j-1.0.0.war

channel stopped

Deploying /opt/jenkins_tomcat6/jobs/quick4j_demo/workspace/target/quick4j.war to container Tomcat 6.x Remote

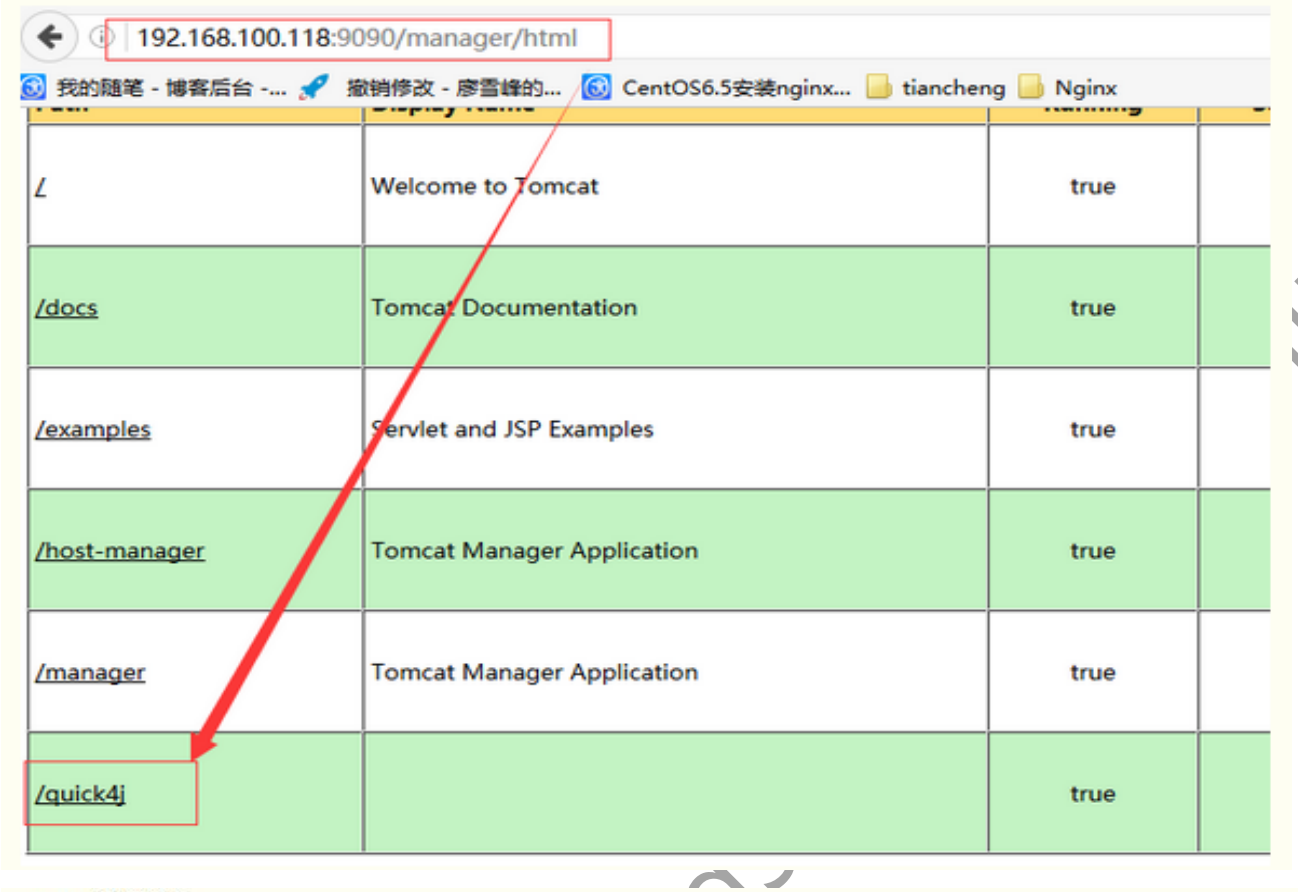
[/opt/jenkins_tomcat6/jobs/quick4j_demo/workspace/target/quick4j.war] is not deployed. Doing a fresh deployment.

Deploying [/opt/jenkins_tomcat6/jobs/quick4j_demo/workspace/target/quick4j.war]

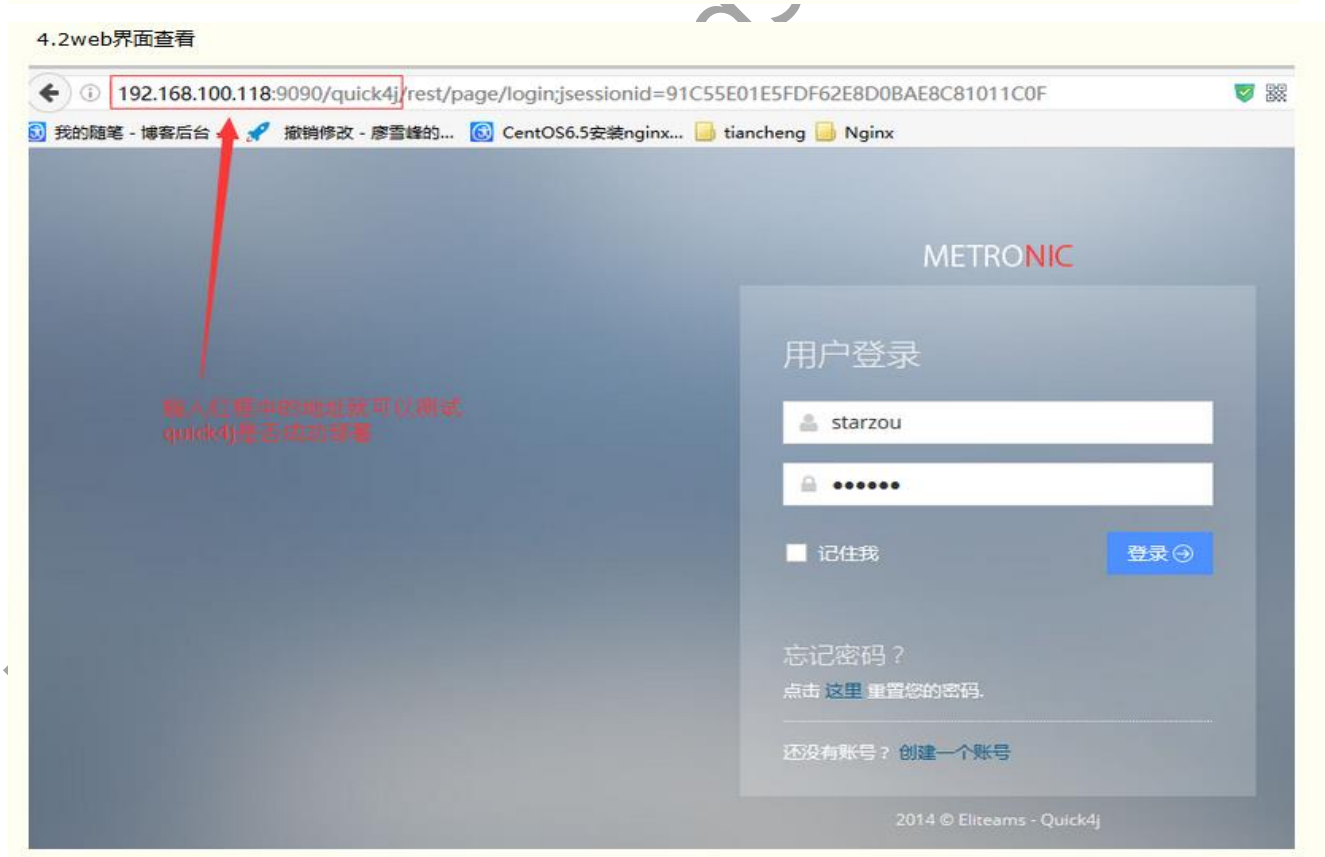
Finished: SUCCESS

4.测试是否发布成功

4.1 检查 tomcat manager 页面



4.2web界面查看



==========

遇到的问题总结：

问题一：部署 app 的时候，tomcat 出现“403/401 for URL: http://172.16.18.192:8090/manager/text/list” 和 The username you provided is not allowed to use the text-based Tomcat Manager (error 403)

问题点： tomcat 用户配置文件

解决方案：检查用户名和密码是否正确 是否添加了用户角色和权限.

问题二（最常见）：部署完 app 后会出现内存溢出"java.lang.OutOfMemoryError: PermGen space"

问题点：tomcat 的虚拟机内存不足，需要设置更大的内存

解决方案：1.找到 tomcat 路径，用编辑器打开 catalina.sh，在“echo "Using CATALINA_BASE: $CATALINA_BASE"” 上面加入以下行：

JAVA_OPTS="-server -XX:PermSize=64M -XX:MaxPermSize=128m

2.用脚本重启 tomcat 服务器，再部署。

重启 tomcat 示例脚本：

```
#!/bin/bash
#Time
log_time=`date +[%Y-%m-%d]%H:%M:%S`

###manual_properties###
tomcat_basehome=/opt/tomcat6/apache-tomcat-6.0.45
tomcat_port=9090
shell_environment=/bin/bash
war_Dir=/opt/jenkins
war_Name=quick4j.war
###manual_properties###



#update server environment
echo "************************************ ${log_time} ************************************"
echo "updating server environment start"
export JAVA_HOME=/app/java/jdk1.8.0_11
export JRE_HOME=/app/java/jdk1.8.0_11/jre
export PATH=$JAVA_HOME/bin:$PATH
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar/
export CATALINA_2_HOME=/opt/tomcat6/apache-tomcat-6.0.45
export CATALINA_2_BASE=/opt/tomcat6/apache-tomcat-6.0.45
export TOMCAT_2_HOME=/opt/tomcat6/apache-tomcat-6.0.45
sleep 3
echo "updating server environment end"

#build check funcation
echo "check tomcat status..."
check_tomcat_status(){
        netstat -ant|grep ${tomcat_port}
        t=$?
          if [ $t -eq 0 ]; then
                echo "tomcat is running....port is ${tomcat_port}"
```

```
                echo "shutdown tomcat....."
                echo ">>>>>>>shutdown tomcat begin<<<<<<<<"
                 ${shell_environment} ${tomcat_basehome}/bin/shutdown.sh
                echo ">>>>>>>shutdown tomcat end <<<<<<<<"
                sleep 5
        elif [ $t -ne 0 ];then
                echo "tomcat is poweroff"
                 ${shell_environment} ${tomcat_basehome}/bin/shutdown.sh
                sleep 5
        fi
}

#check tomcat status invoke function
check_tomcat_status


#transfer    application package
deploy_Loaction=${tomcat_basehome}/webapps/
war_Dir_Data=`ls ${war_Dir}`
echo "start    transfer    war package to tomcat webapps ......."

if [ -z $war_Dir ];then
        echo "Folder ${war_Dir} is empty.please check war package in this folder!"
        exit 1
else
        echo "Find ${war_Dir} exist war package ${war_Name}"
       # echo "deleteing old    package ${war_Name} in ${war_Dir}"
       # rm ${war_Dir}/${war_Name}
        echo "deleteing old    package ${war_Name} in ${deploy_Loaction}"
        rm ${deploy_Loaction}${war_Name}
        echo "start    transfer ${war_Name} to ${deploy_Loaction}"
        cp ${war_Dir}/${war_Name}    ${deploy_Loaction}
        sleep 3
fi
#reboot tomcat
echo " >>>>>>>    rebooting    tomcat begin <<<<<<<<"
${shell_environment} ${tomcat_basehome}/bin/startup.sh
echo " >>>>>>>    rebooting    tomcat end <<<<<<<<"
echo "the log you can read in canalina.out"
echo        "************************        deploy        war        package        into        container        Successlly
**********************************"
```

问题三：遇到“Connection refused”的异常

问题点：tomcat

解决方案：请检查远程机的容器是否启动、端口是否设置正常，不同的容器配置方式不一样，请参考相应容器的配置文档

问题四：遇到“Deployed application at context path /xxx but context failed to start”

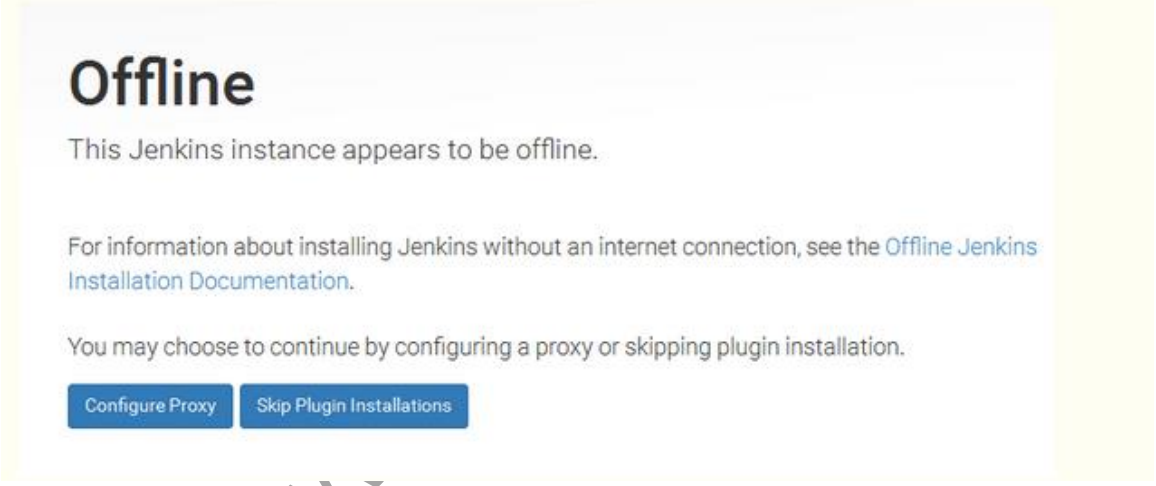
问题点：tomcat

解决方案：到远程机的 WEB 容器下查看日志

[原]Jenkins(十六) jenkins 再出发之 jenkins+robot+blue ocean+svn

jenkins version：
部署省略。。（如有需要请查看本博客 jenkins 系列的文档）
新的 jenkins 需要先填写 administratorpassword （如下图）找到下面红色的路径打开就是密码：

由于没有网络，所以只有offline 安装（如下图）：



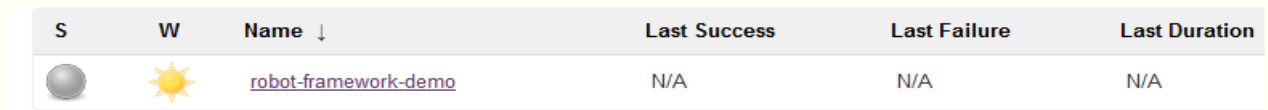
创建 admin 用户：全部设置为 admin

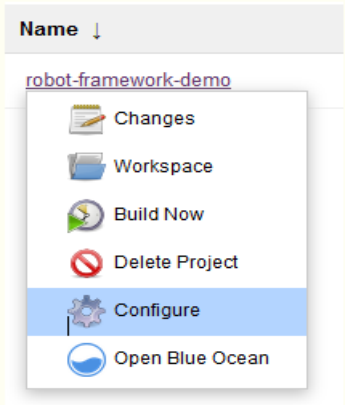
开始使用jenkins:

界面依然是这个布局:

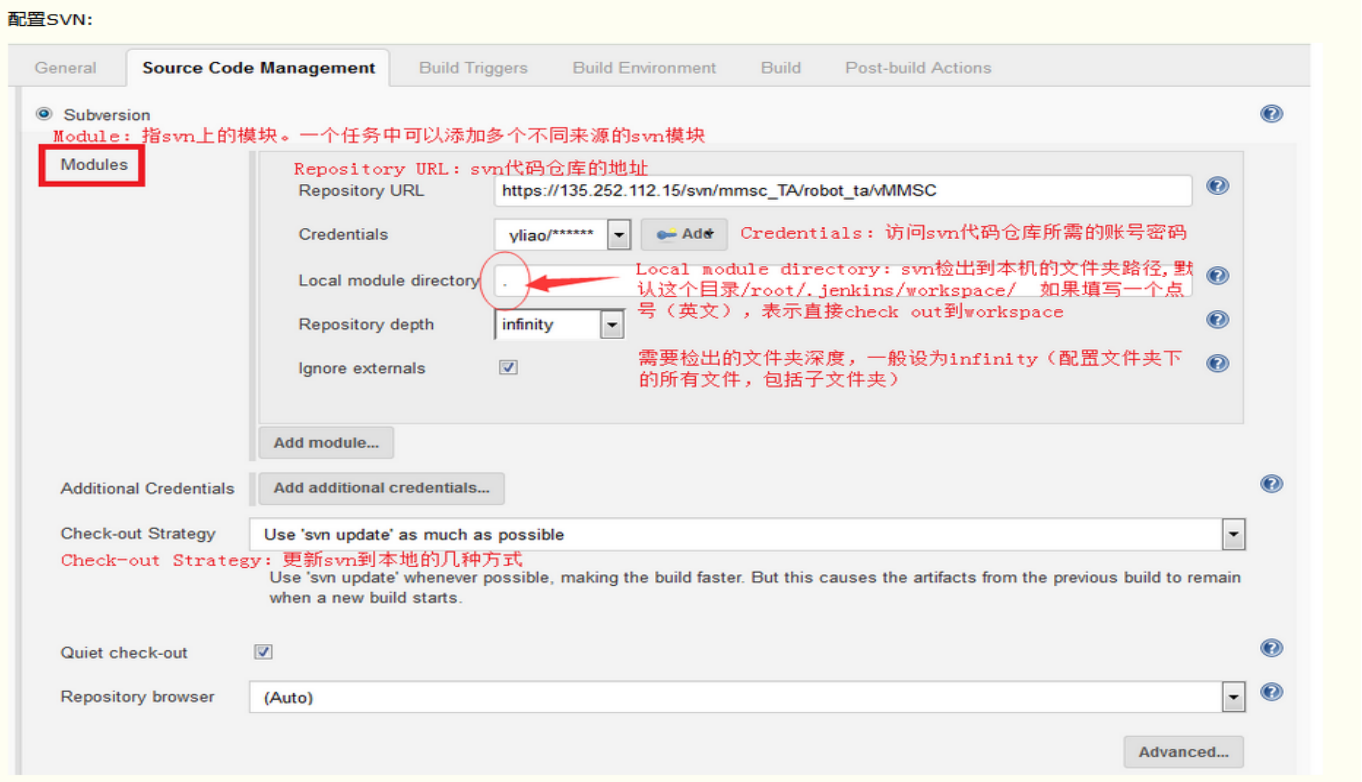
[原]Jenkins(十七) jenkins 再出发之配置 SVN

创建一个demo project



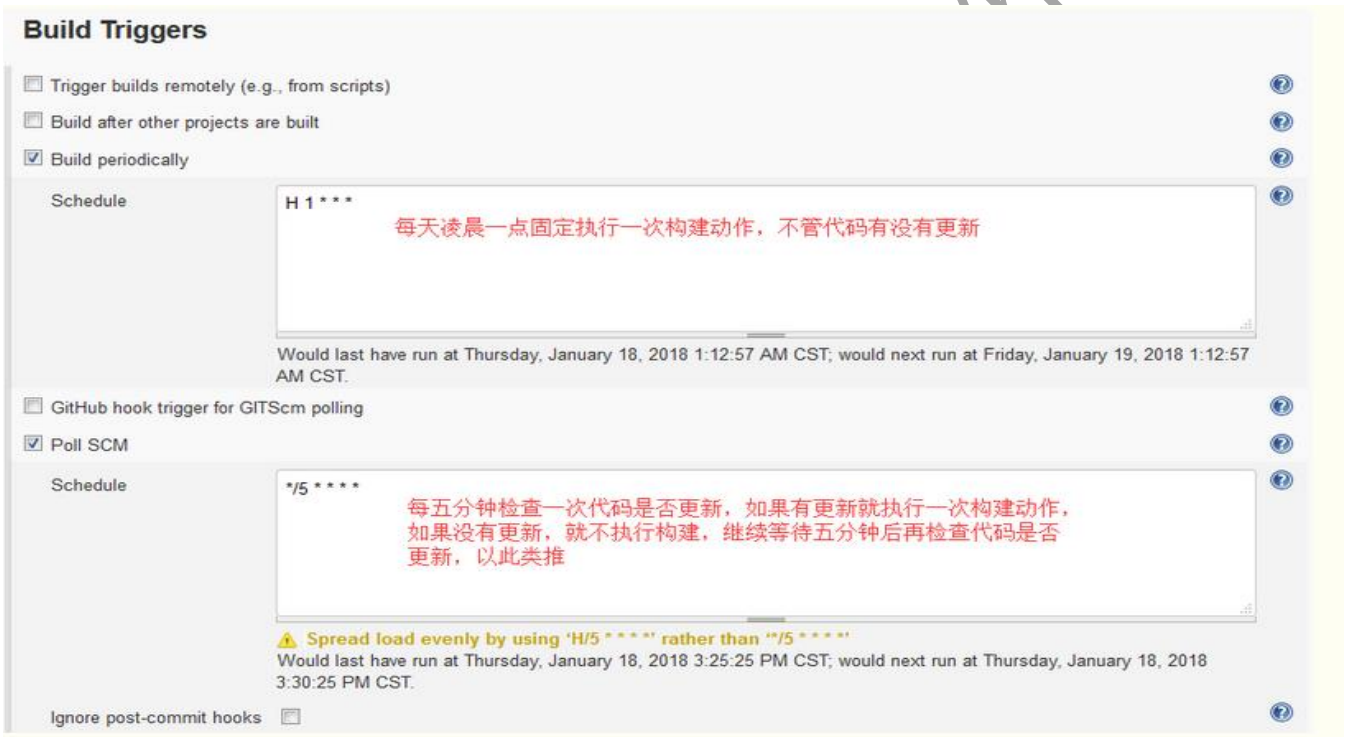


配置SVN:



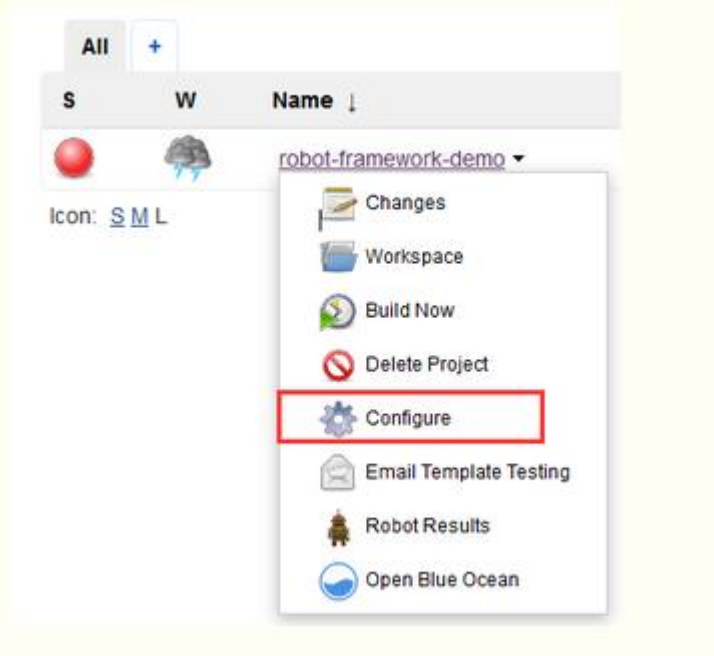
Check-out Strategy	第一次build	第n次build(除第一次)
Use 'svn update' as much as possible	将workspace下的所有文件清空，然后从svn上check out一份完整的项目到workspace下	update前不会revert
Always check out a fresh copy		删除workspace下的所有文件，然后重新check out一份完整的项目到workspace下。
Emulate clean checkout by first deleting unversioned/ignored files, then 'svn update'		update前先删除unversioned/ignored文件
Use 'svn update' as much as possible, with 'svn revert' before update		update前先revert

配置 build project

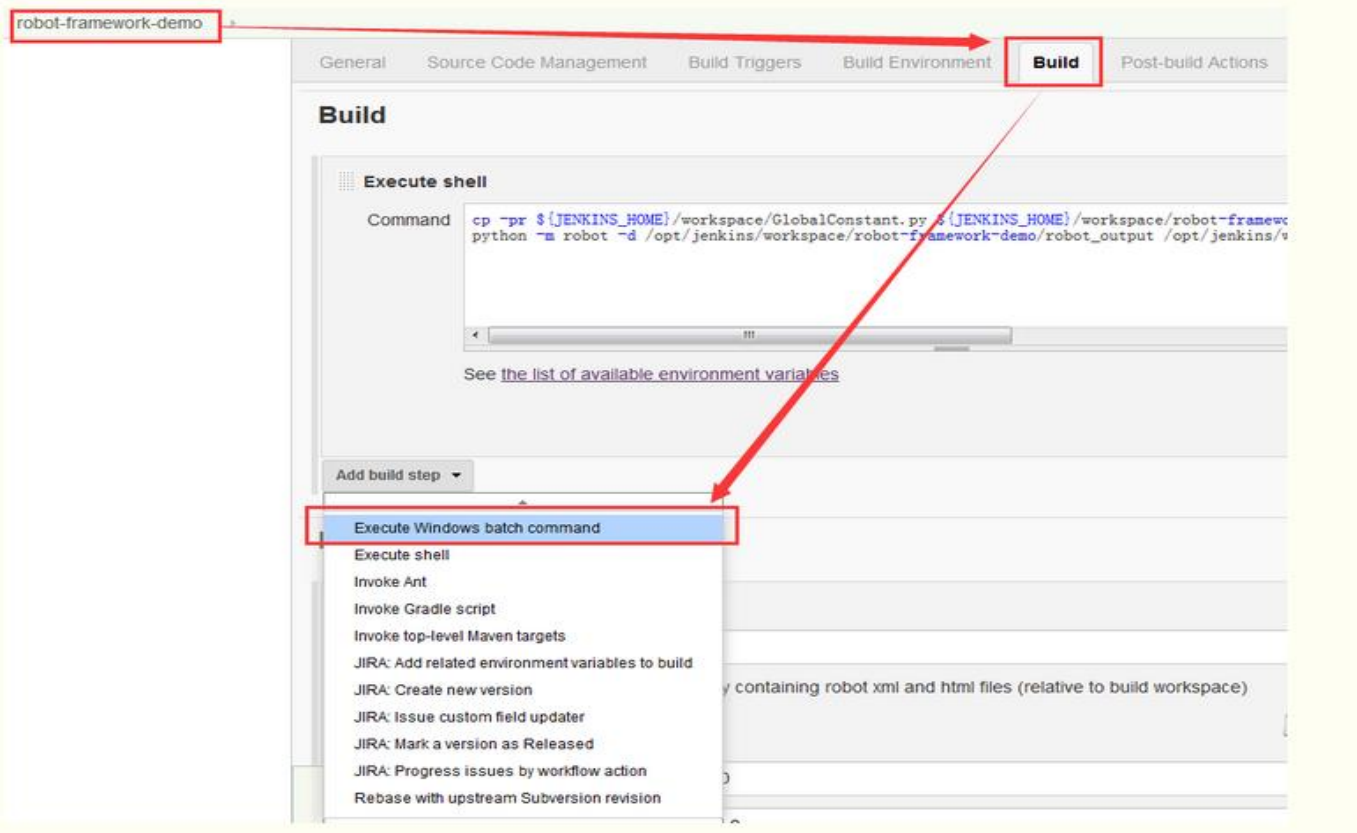


[原]Jenkins(十八) jenkins 再出发之 jenkins 内置变量

1. 选择一个 project 的 config 选项：



2. 选择 build 选项卡，选择 Execute Windows batch command



3. 会出现一个内置变量的 list 连接按钮：

4. list 表内容如下所示：

The following variables are available to shell scripts

BRANCH_NAME
For a multibranch project, this will be set to the name of the branch being built, for example in case you wish to deploy to production from master but not from feature branches; if corresponding to some kind of change request, the name is generally arbitrary (refer to CHANGE_ID and CHANGE_TARGET).
CHANGE_ID
For a multibranch project corresponding to some kind of change request, this will be set to the change ID, such as a pull request number, if supported; else unset.
CHANGE_URL
For a multibranch project corresponding to some kind of change request, this will be set to the change URL, if supported; else unset.
CHANGE_TITLE
For a multibranch project corresponding to some kind of change request, this will be set to the title of the change, if supported; else unset.
CHANGE_AUTHOR
For a multibranch project corresponding to some kind of change request, this will be set to the username of the author of the proposed change, if supported; else unset.
CHANGE_AUTHOR_DISPLAY_NAME
For a multibranch project corresponding to some kind of change request, this will be set to the human name of the author, if supported; else unset.
CHANGE_AUTHOR_EMAIL
For a multibranch project corresponding to some kind of change request, this will be set to the email address of the author, if supported; else unset.
CHANGE_TARGET
For a multibranch project corresponding to some kind of change request, this will be set to the target or base branch to which the change could be merged, if supported; else unset.
BUILD_NUMBER
The current build number, such as "153"
BUILD_ID
The current build ID, identical to BUILD_NUMBER for builds created in 1.597+, but a YYYY-MM-DD_hh-mm-ss timestamp for older builds
BUILD_DISPLAY_NAME
The display name of the current build, which is something like "#153" by default.
JOB_NAME
Name of the project of this build, such as "foo" or "foo/bar".
JOB_BASE_NAME
Short Name of the project of this build stripping off folder paths, such as "foo" for "bar/foo".
BUILD_TAG
String of "jenkins-*${JOB_NAME}-${BUILD_NUMBER}*". All forward slashes (/) in the JOB_NAME are replaced with dashes (-).
Convenient to put into a resource file, a jar file, etc for easier identification.
EXECUTOR_NUMBER
The unique number that identifies the current executor (among executors of the same machine) that's carrying out this build.
This is the number you see in the "build executor status", except that the number starts from 0, not 1.
NODE_NAME
Name of the agent if the build is on an agent, or "master" if run on master
NODE_LABELS
Whitespace-separated list of labels that the node is assigned.
WORKSPACE
The absolute path of the directory assigned to the build as a workspace.
JENKINS_HOME
The absolute path of the directory assigned on the master node for Jenkins to store data.
JENKINS_URL
Full URL of Jenkins, like http://server:port/jenkins/ (note: only available if *Jenkins URL* set in system configuration)
BUILD_URL
Full URL of this build, like http://server:port/jenkins/job/foo/15/ (*Jenkins URL* must be set)

JOB_URL Full URL of this job, like http://server:port/jenkins/job/foo/ (*Jenkins URL* must be set)
GIT_COMMIT The commit hash being checked out.
GIT_PREVIOUS_COMMIT The hash of the commit last built on this branch, if any.
GIT_PREVIOUS_SUCCESSFUL_COMMIT The hash of the commit last successfully built on this branch, if any.
GIT_BRANCH The remote branch name, if any.
GIT_LOCAL_BRANCH The local branch name being checked out, if applicable.
GIT_URL The remote URL. If there are multiple, will be GIT_URL_1, GIT_URL_2, etc.
GIT_COMMITTER_NAME The configured Git committer name, if any.
GIT_AUTHOR_NAME The configured Git author name, if any.
GIT_COMMITTER_EMAIL The configured Git committer email, if any.
GIT_AUTHOR_EMAIL The configured Git author email, if any.
MERCURIAL_REVISION Full ID of revision checked out.
MERCURIAL_REVISION_SHORT Abbreviated ID of revision checked out.
MERCURIAL_REVISION_NUMBER Number of revision checked out (not portable across clones).
MERCURIAL_REVISION_BRANCH Branch of revision checked out, if not checking out by branch head.
MERCURIAL_REPOSITORY_URL URL of repository.
SVN_REVISION Subversion revision number that's currently checked out to the workspace, such as "12345"
SVN_URL Subversion URL that's currently checked out to the workspace.

5. 使用变量的两种方式：

1）

2）

在 build.xml 文件中使用：

1、添加如下第 4 行代码：<property environment="env"/>

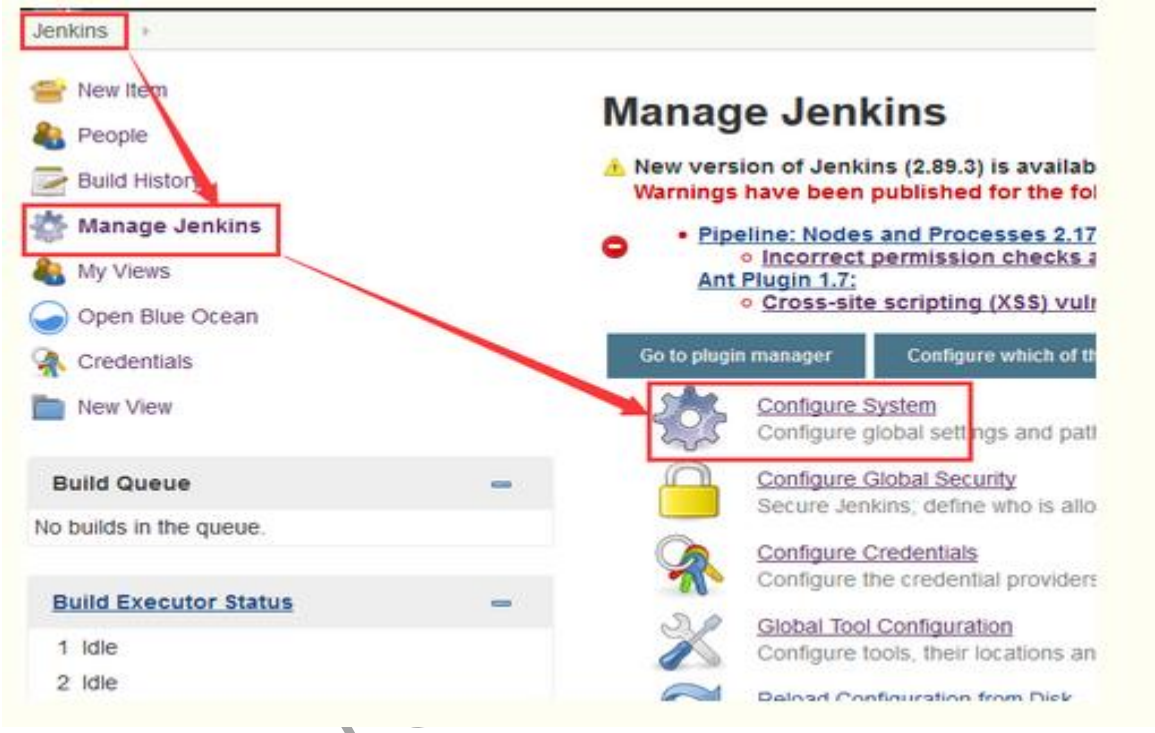
2、使用方法：${env.WORKSPACE}

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="test" default="testing" basedir=".">
    <property environment="env"/>
    <target name="release">
        <mkdir dir="${env.WORKSPACE}/results/${env.BUILD_ID}" />
    </target>
</project>
```

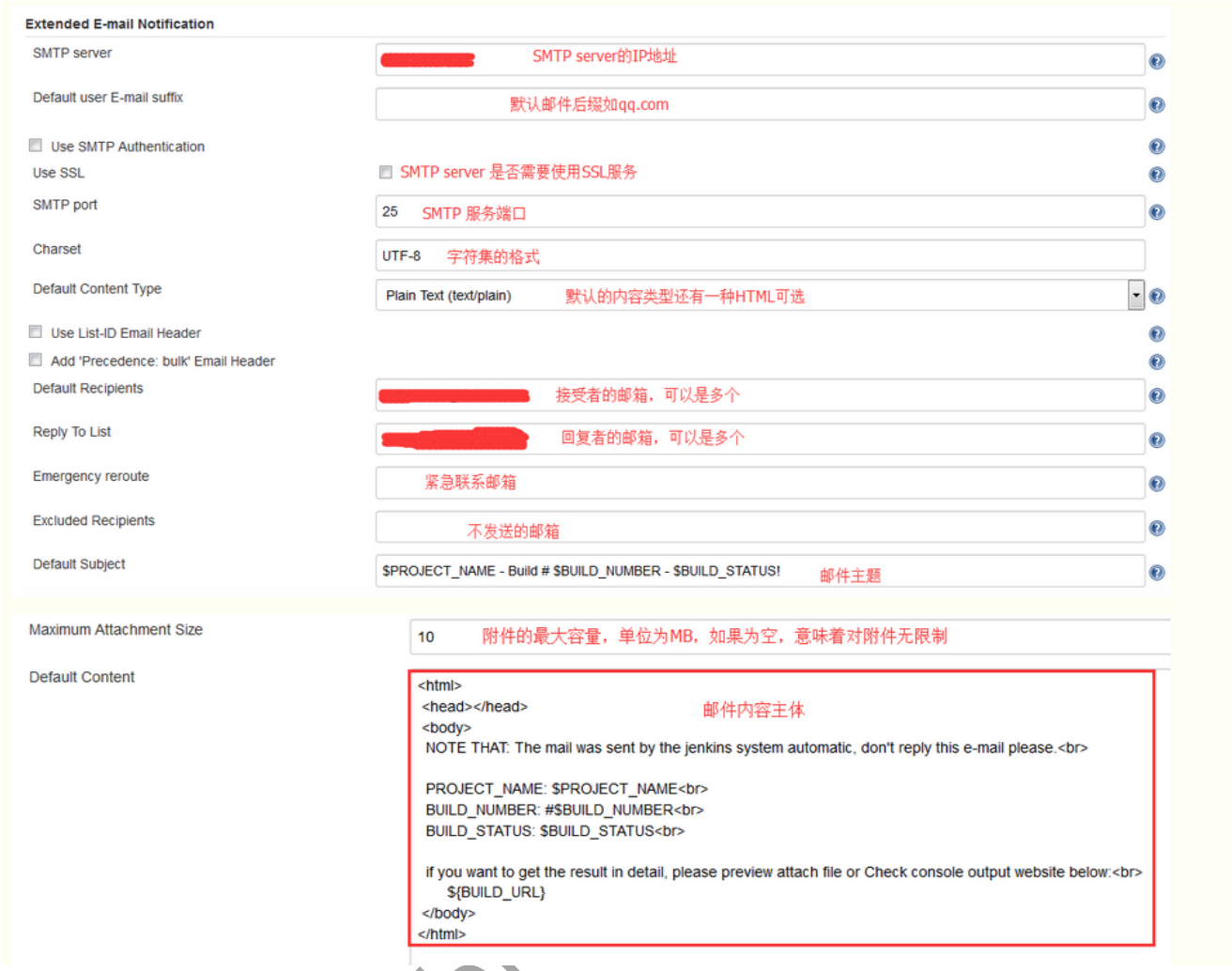
[原]Jenkins(十九) jenkins 再出发之 jenkins 邮件通知

1. 下载插件：

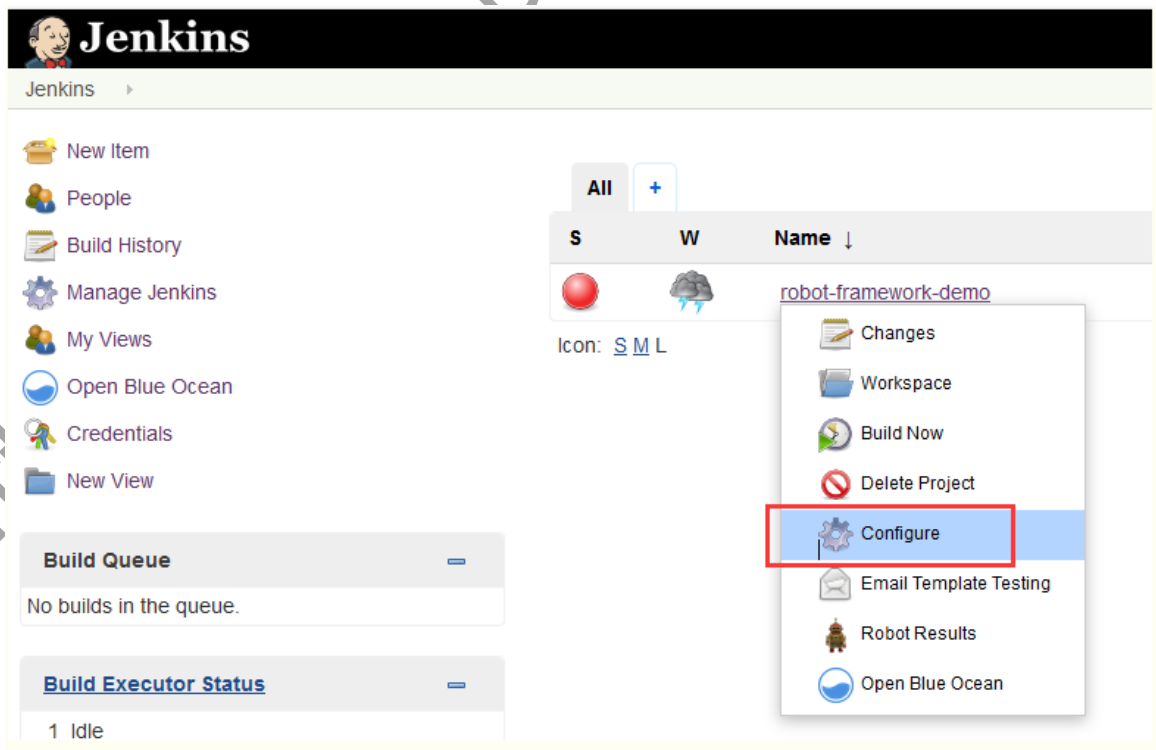
2. 配置插件：

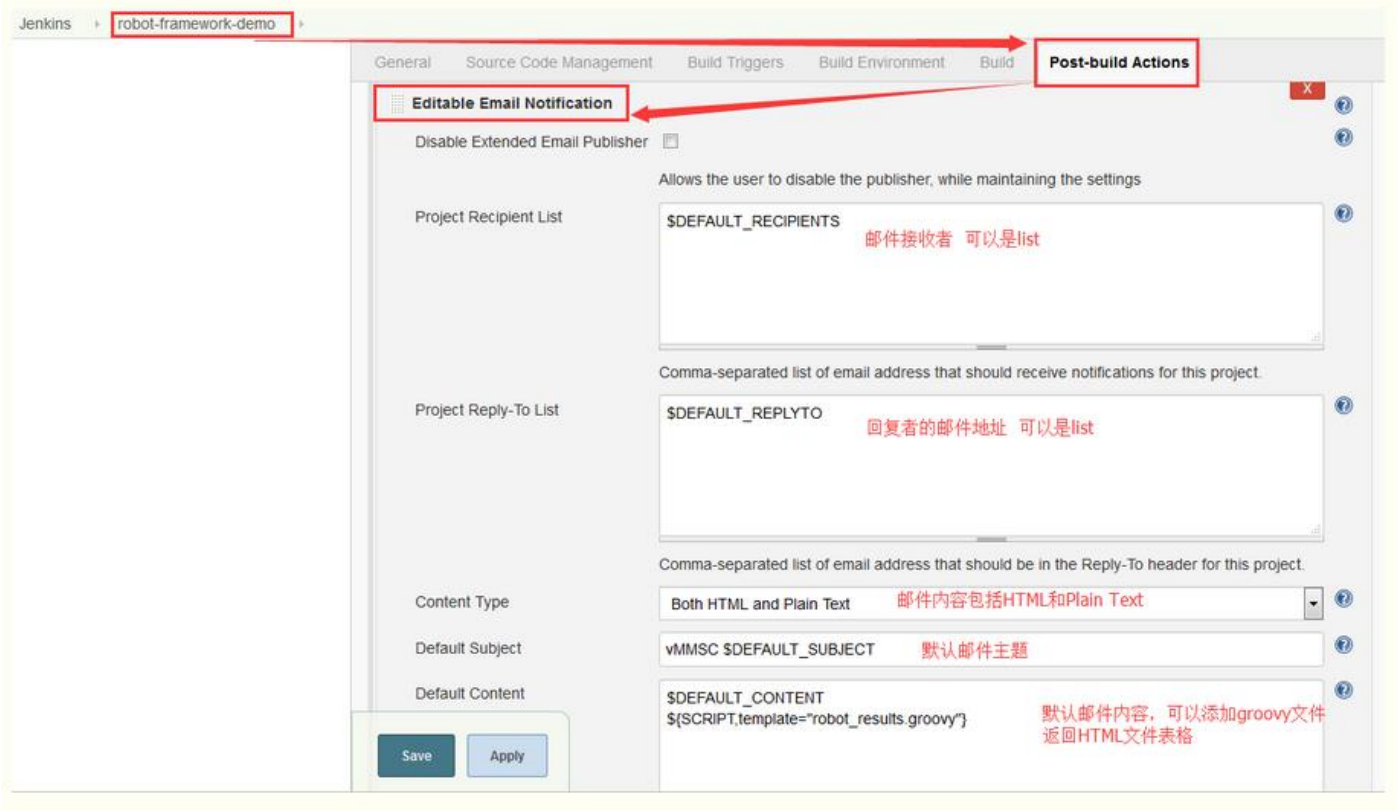


3. 邮件插件配置


Extended E-mail Notification
SMTP server
SMTP server的IP地址
Default user E-mail suffix
默认邮件后缀如qq.com
Use SMTP Authentication
Use SSL
SMTP server 是否需要使用SSL服务
SMTP port
25
SMTP 服务端口
Charset
UTF-8
字符集的格式
Default Content Type
Plain Text (text/plain)
默认的内容类型还有一种HTML可选
Use List-ID Email Header
Add 'Precedence: bulk' Email Header
Default Recipients
接受者的邮箱，可以是多个
Reply To List
回复者的邮箱，可以是多个
Emergency reroute
紧急联系邮箱
Excluded Recipients
不发送的邮箱
Default Subject
$PROJECT_NAME - Build # $BUILD_NUMBER - $BUILD_STATUS!
邮件主题
Maximum Attachment Size
10
附件的最大容量，单位为MB，如果为空，意味着对附件无限制
Default Content
<html>
<head></head>
邮件内容主体
<body>
NOTE THAT: The mail was sent by the jenkins system automatic, don't reply this e-mail please.
PROJECT_NAME: $PROJECT_NAME
BUILD_NUMBER: #$BUILD_NUMBER
BUILD_STATUS: $BUILD_STATUS
if you want to get the result in detail, please preview attach file or Check console output website below:
${BUILD_URL}
</body>
</html>

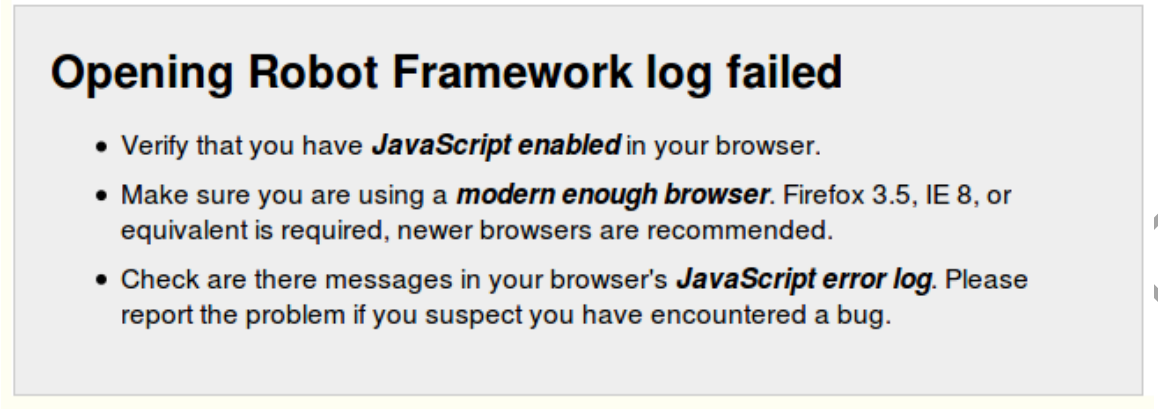
4. 设置触发器


Jenkins
Jenkins
New Item
People
Build History
Manage Jenkins
My Views
Open Blue Ocean
Credentials
New View
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
All
+
S
W
Name
robot-framework-demo
Icon: S M L
Changes
Workspace
Build Now
Delete Project
Configure
Email Template Testing
Robot Results
Open Blue Ocean



[原]Jenkins(二十) jenkins 再出发之 Error: Opening Robot Framework log failed

错误缘由：使用 plugin [public robot framework test results] 生成的 HTML 文件都无法正常打开.

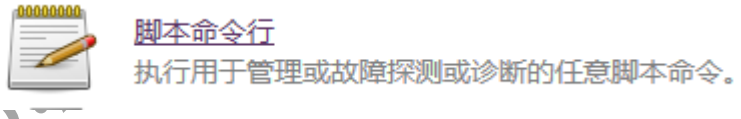


解决方案：

- Connect on your jenkins url (http://[IP]:8080/)
- 

- Click on administer Jenkins
- 

- Click on consol jenkins
- 

- Copy this into the field and execute :
- 

input the code below:

System.setProperty("hudson.model.DirectoryBrowserSupport.CSP","sandbox allow-scripts; default-src 'none'; img-src 'self' data: ; style-src 'self' 'unsafe-inline' data: ; script-src 'self' 'unsafe-inline' 'unsafe-eval' ;")

脚本命令行

Type in an arbitrary Groovy script and execute it on the server. Useful for trouble-shooting and diagnostics. Use the 'println' command to see the output (if you use `System.out`, it will go to the server's stdout, which is harder to see.) Example:

```
println(Jenkins.instance.pluginManager.plugins)
```

All the classes from all the plugins are visible. `jenkins.*`, `jenkins.model.*`, `hudson.*`, and `hudson.model.*` are pre-imported.

```
1 System.setProperty("hudson.model.DirectoryBrowserSupport.CSP","sandbox allow-scripts; default-src 'none'; img-src 'self' data: ; style-src
```

运行

Result

```
Result: sandbox allow-scripts; default-src 'none'; img-src 'self' data: ; style-src 'self' 'unsafe-inline' data: ; script-src 'self' 'unsafe-inline' 'unsafe-eval' ;
```

Success view html file:

robotframeworkText Test Report

Generated
20180410 14:33:20 GMT+08:00
2 hours 51 minutes ago

Summary Information

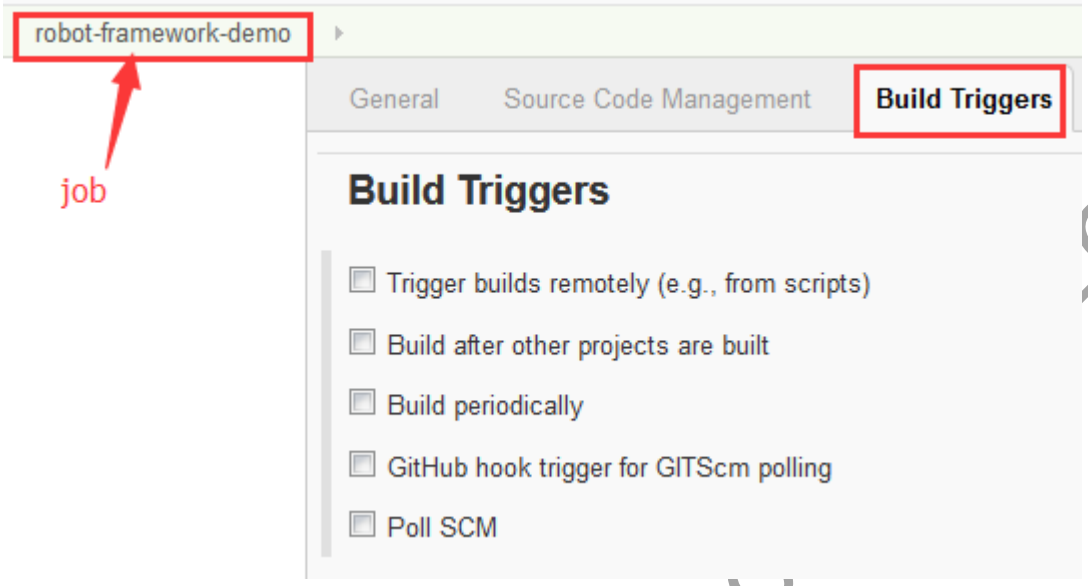
Status:	All tests passed
Start Time:	20180410 14:33:20.391
End Time:	20180410 14:33:20.587
Elapsed Time:	00:00:00.196
Log File:	log.html

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	2	2	0	00:00:00	
All Tests	2	2	0	00:00:00	

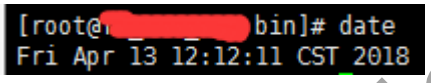
[原]Jenkins(二十一) jenkins 再出发 Build periodically 和 Poll SCM

缘由： 使用 jenkins 的目的需要固定时间构建和间隔固定时间构建，所以才会用到这两个功能。
位置：这两个功能的位置位于每个 job 的 config 项中，如下图：

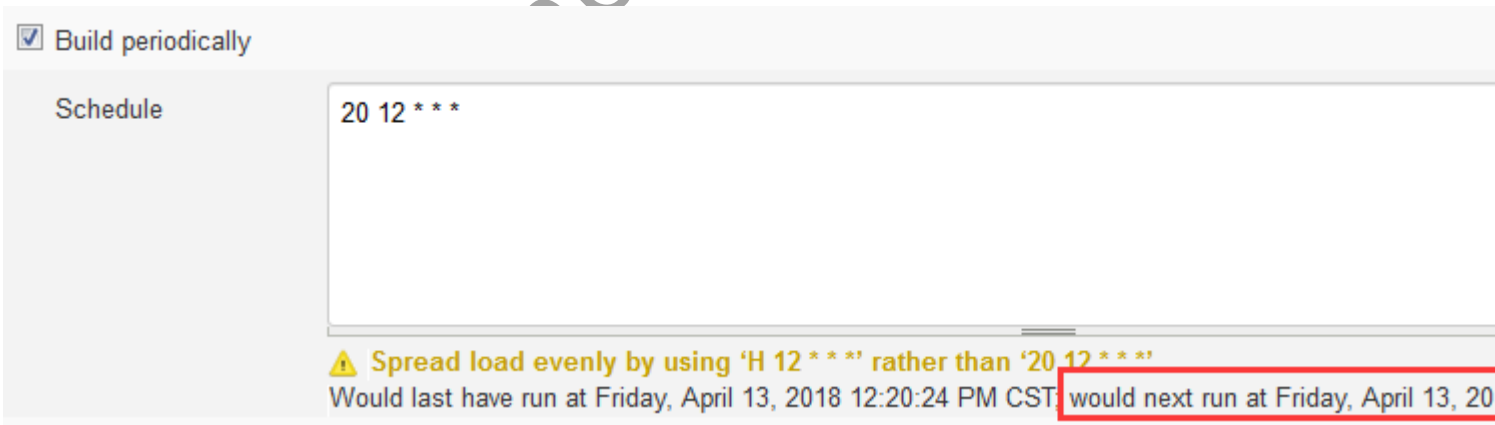


【重要的注意点：】

1）jenkins 所在主机的当前时间



2）确认设置的执行时间点：（此时间必须比主机当前时间晚）



【 需要了解的知识：】

* * * * *

(五颗星，中间用空格隔开）
第一颗*表示分钟，取值 0~59
第二颗*表示小时，取值 0~23

第三颗*表示一个月的第几天，取值 1~31
第四颗*表示第几月，取值 1~12
第五颗*表示一周中的第几天，取值 0~7，其中 0 和 7 代表的都是周日
1.每 30 分钟构建一次：
H/30 * * * *
2.每 2 个小时构建一次
H H/2 * * *
3.每天早上 8 点构建一次
0 8 * * *
4.每天的 8 点，12 点，22 点，一天构建 3 次
0 8,12,22 * * *
（多个时间点，中间用逗号隔开）
40 12 * * 1-7
每天 12 点 40 分执行构建

【需要了解的功能】

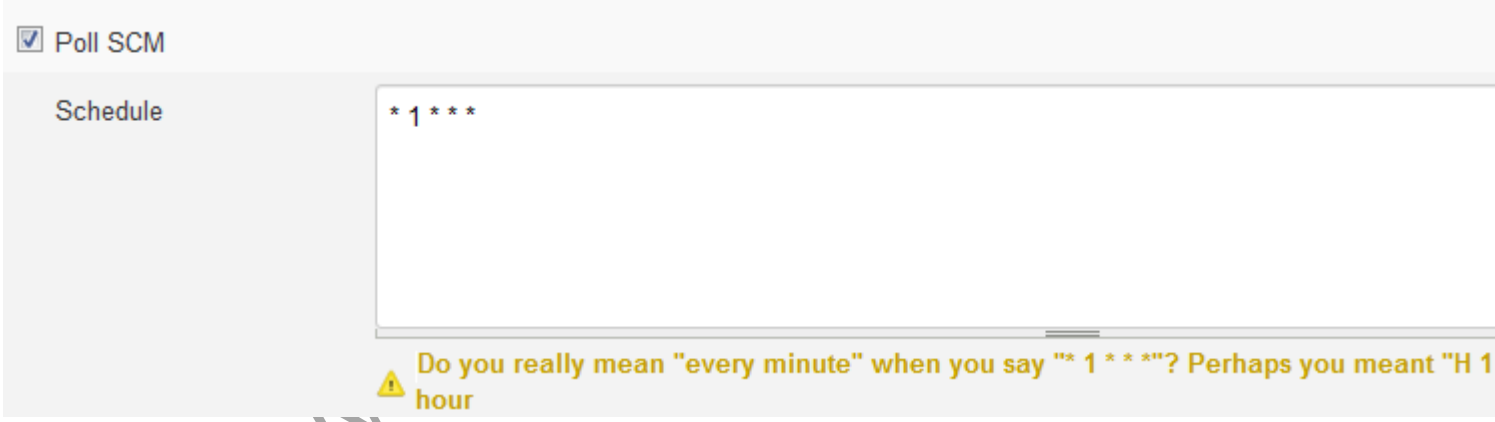
Poll SCM

说白了就是是让其自动每隔一段固定时间去检查版本管理工具（SVN/GIT）上的代码是否有改动，如果有改动就进行构建

专业术语就是：定时检查源码变更（根据 SCM 软件的版本号），如果有更新就 checkout 最新 code 下来，然后执行构建动作

示例：

每分钟都去检查版本库是否有更新 如果有更新就进行构建



"Do you really mean "every minute" when you say "* 1 * * *"? Perhaps you meant "H 1 * * *" to poll once per hour" 的意思是让你确认，填写的是否是你需要的那个时间段

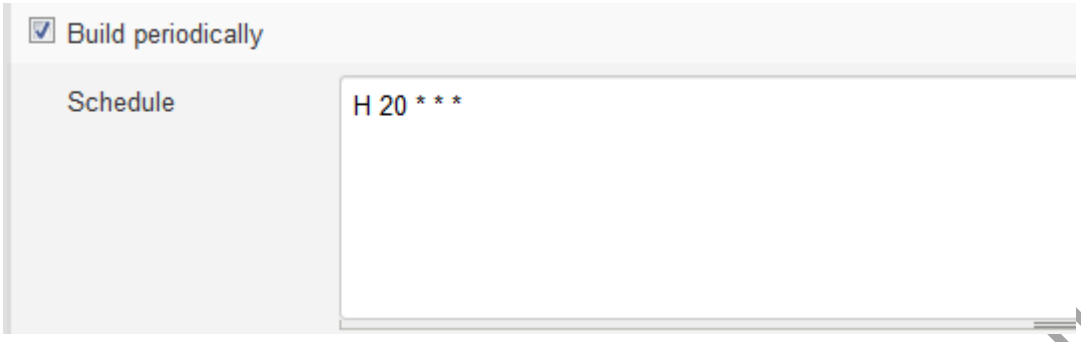
Build periodically

说白了就是让 jenkins 在固定的时间自动构建项目

专业术语就是：周期性进行项目构建，这个是到指定的时间必须触发构建任务

示例：

晚上八点进行构建



Build after other projects are built

说白了就是连续一个接着一个的进行构建，多个 jobs 用逗号（,）隔开。比如一个 web 项目构建完成了，就接着进行自动化测试的构建

专业术语：同上

这有三个可选项，默认第一个用的场景比较多

Trigger only if build is stable：构建稳定时触发

Trigger even if the build is unstable ：构建不稳定时触发

Trigger even if the build fails ： 构建失败的时候触发

Trigger builds remotely (e.g., from scripts)

触发远程构建 (例如：使用脚本）

GitHub hook trigger for GITScm polling

Github 上代码有变动就进行构建