**Jenkins安装使用说明**

首先需要了解到的是Jenkins是一个开源的持续集成工具，应用Jenkins搭建持续集成环境，可以进行自动构建、自动编译和部署，非常方便。（710587 boncpw）

1. **jenkins安装步骤**

**1.先判断是否有自带jdk存在**

**rpm -qa | grep jdk** ：查看jdk的具体信息

**2.通过和java -version里面的jdk版本比较**

用**rpm -e --nodeps** java-1.5.0-gcj-1.5.0.0-29.1.el6.x86\_64命令卸载相应的jdk。java-1.5.0-gcj-1.5.0.0-29.1.el6.x86\_64是查询出来的jdk。

**3**.**安装JDK，maven环境搭建**

**（下载jdk-8u131-linux-x64.tar.gz和apache-maven-3.5.0-bin.tar.gz，进行安装即可）**

**把压缩包解压到usr/local/java 和/usr/local/maven**

**4.配置jdk：vim /etc/profile**

**JAVA\_HOME=/usr/local/java/jdk1.8.0\_131**

**CLASSPATH=$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar**

**PATH=$PATH:$JAVA\_HOME/bin**

**export PATH JAVA\_HOME CLASSPATH**

**配置完成后执行**source /etc/profile保存。

**5.配置maven: vim /etc/profile**

**export MAVEN\_HOME=/usr/local/maven/apache-maven-3.5.0**

**export PATH=$PATH:$MAVEN\_HOME/bin**

**配置完成后执行**source /etc/profile

**# 修改tomcat开机自启动(修改t**omcat/bin/startup.sh**)**

**export JAVA\_HOME=/usr/local/java/jdk1.8.0\_131**

**export CLASSPATH=$CLASSPATH:$JAVA\_HOME/lib/tools.jar:$JAVA\_HOME/lib/dt.jar:.**

**export PATH=$PATH:$JAVA\_HOME/bin**

**export CATALINA\_HOME=/usr/local/tomcat/apache-tomcat-8.0.44**

**/usr/local/tomcat/apache-tomcat-8.0.44/bin/catalina.sh start**

**在/etc/rc.d/rc.local中加入:**

**export JAVA\_HOME=/usr/local/java/jdk1.8.0\_131  
/usr/local/tomcat/apache-tomcat-8.0.44/bin/startup.sh**

**重启后生效**

**获取tomcat进程号命令:** **lsof -n -P -t -i :tomcat端口号**

**下面有配置第二个tomcat的步骤~**

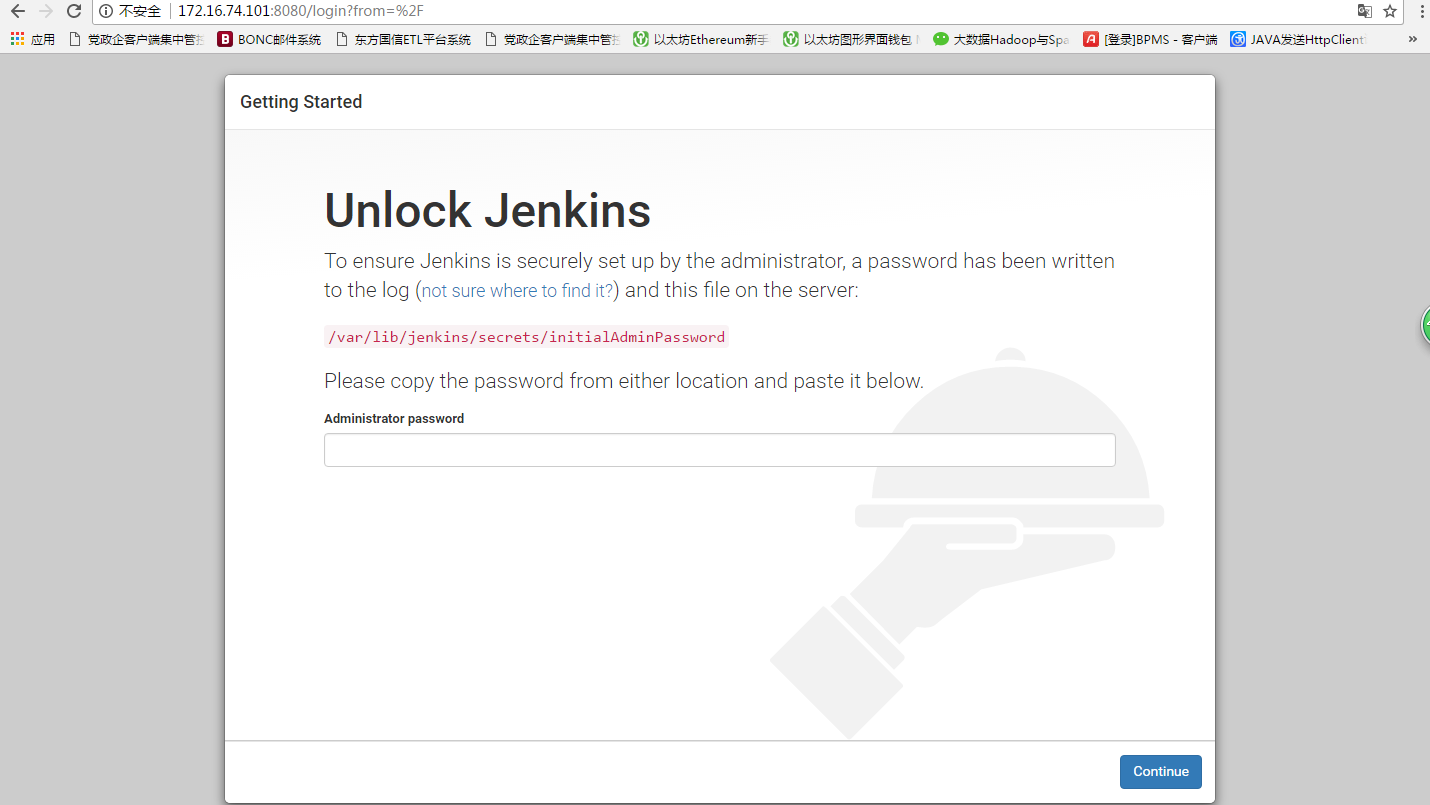
**防火墙设置:**

systemctl stop firewalld.service #停止firewall  
systemctl disable firewalld.service #禁止firewall开机启动

**6.部署安装jenkins在tomcat下**

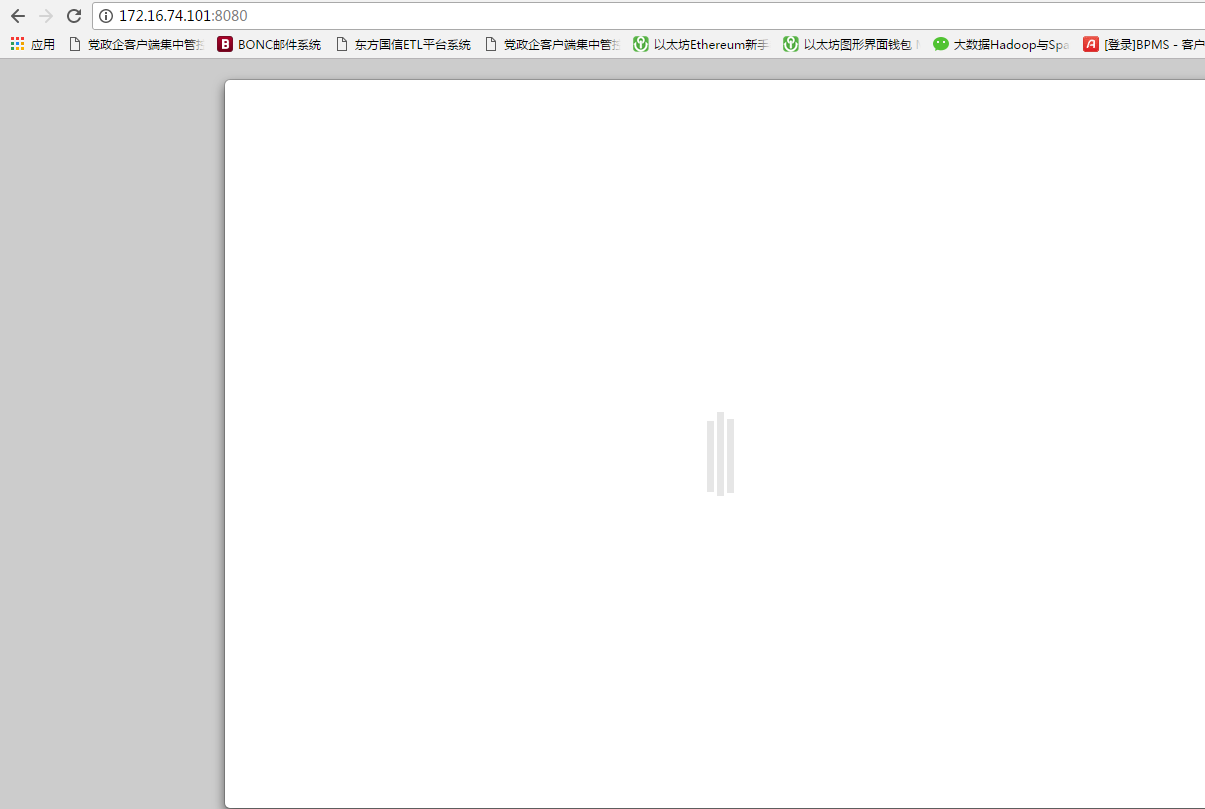
1. **在/usr/local/tomcat目录下解压apache-tomcat-8.0.44.tar.gz**
2. **把jenkins.war放到tomcat的webapp下，启动tomcat**
3. **访问http://** **127.0.0.1:8080/jenkins**

第一次访问需要输入密码：

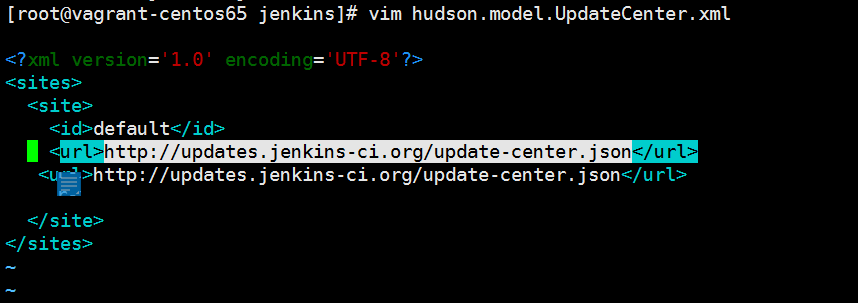




点击continue后，一直卡在启动页面无法加载。



解决办法，修改/var/lib/Jenkins/vim hudson.model.UpdateCenter.xml文件。将<url>[http://updates.jenkins-ci.org/update-center.json</url](http://updates.jenkins-ci.org/update-center.json%3c/url)>改为<url>[http://mirror.xmission.com/jenkins/updates/update-center.json</url](http://mirror.xmission.com/jenkins/updates/update-center.json%3c/url)>后重启服务service Jenkins restart



vim default.json , 可以看到前面的键值对为: "connectionCheckUrl":"http://www.google.com/" ，将其改为 "connectionCheckUrl":<http://www.baidu.com/>

如果还是出现问题，卸载之后重装。

Jenkins：账户：root，密码：1qaz2wsx

<http://www.cnblogs.com/zhengah/p/5206533.html>

<http://blog.csdn.net/wangmuming/article/details/22925127>

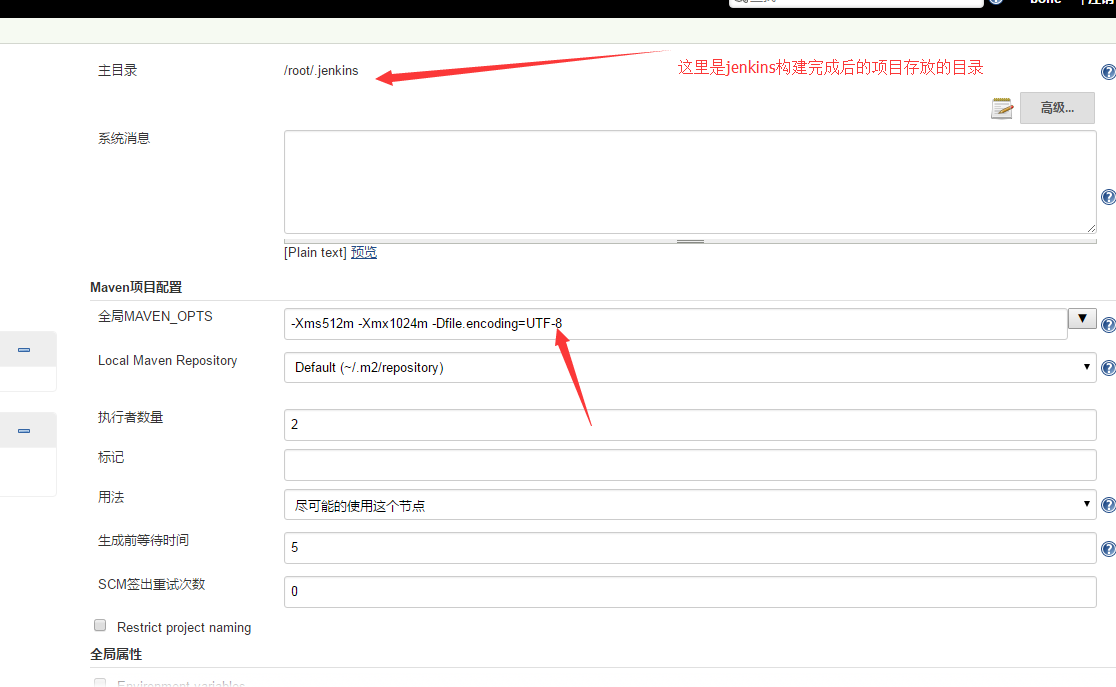
<http://blog.csdn.net/xlgen157387/article/details/50353317>

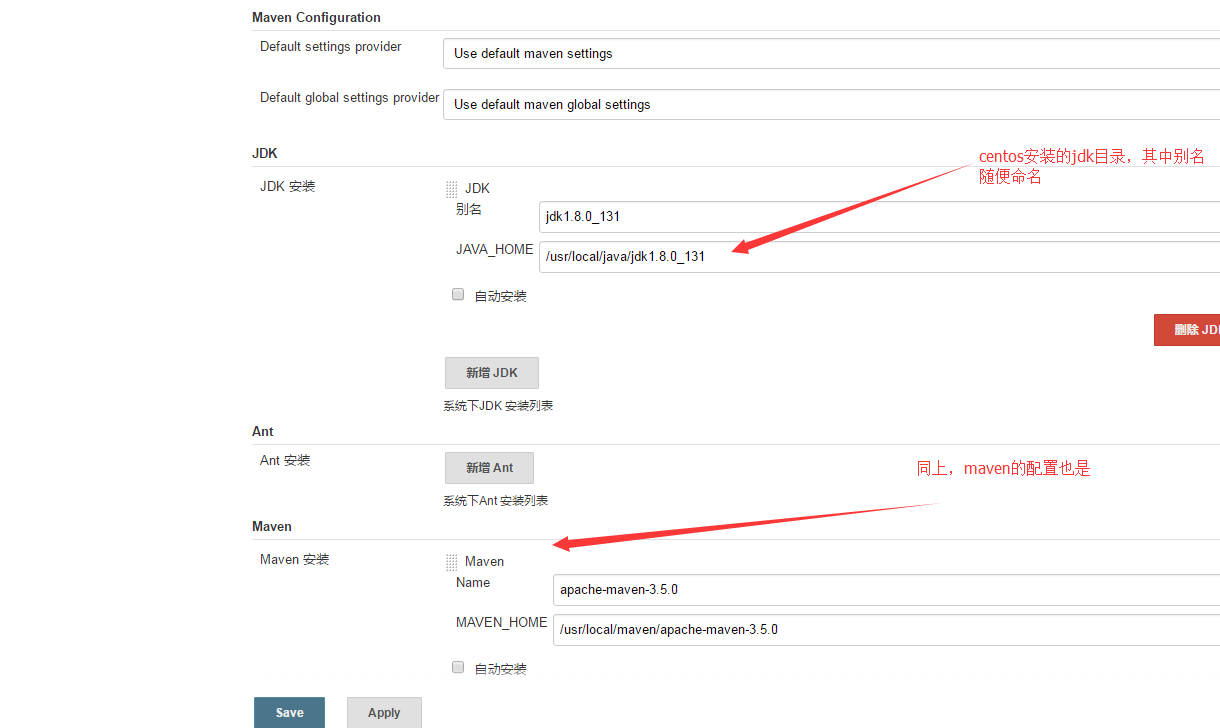
<http://blog.csdn.net/pein_zero/article/details/52597615> 这个贼牛，插件问题

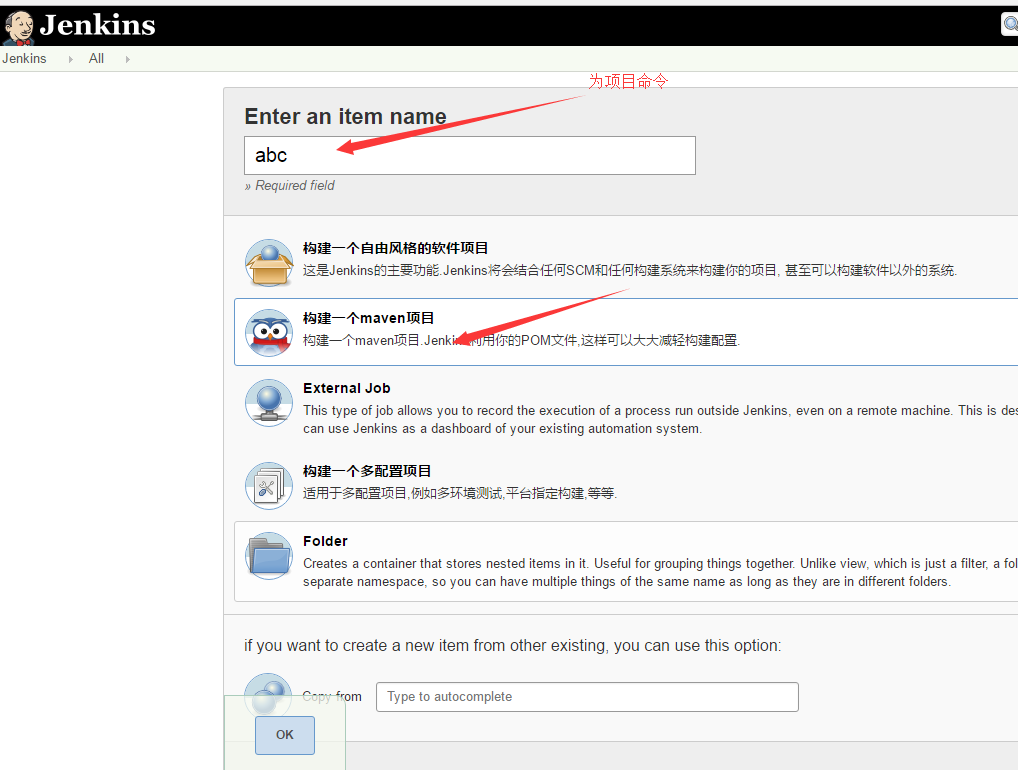
<http://kurenai.elastos.org/2013/03/29/jenkins%E5%AD%A6%E4%B9%A0%E7%AC%94%E8%AE%B0%EF%BC%88%E5%85%AD%EF%BC%89%E4%B9%8B%E4%BF%AE%E6%94%B9%E5%B7%A5%E4%BD%9C%E7%9B%AE%E5%BD%95/> 修改工作目录

**7.搭建步骤：**

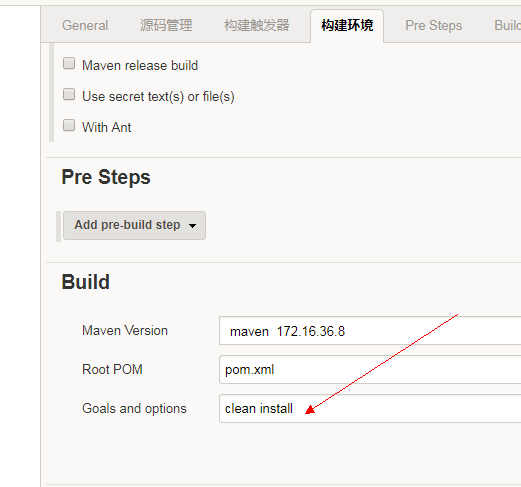


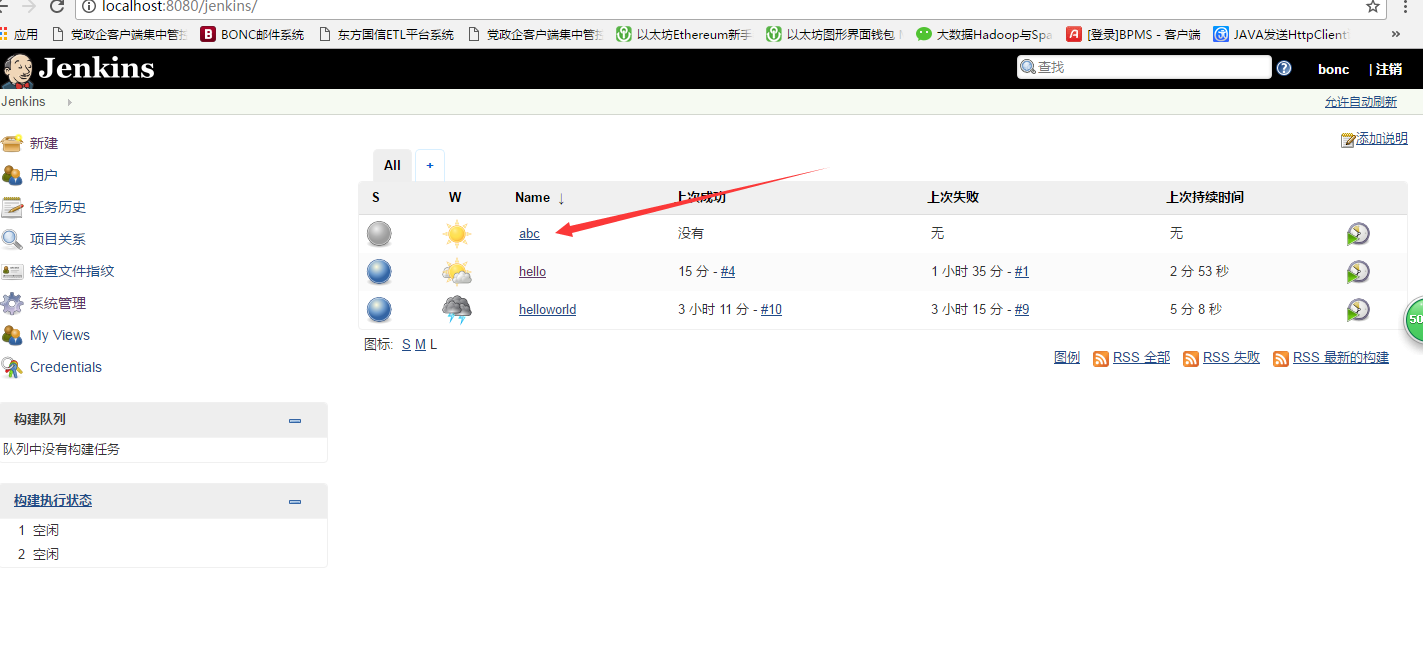






配置maven项目时，在下图箭头指示方向输入clean install -Dmaven.test.skip=true :跳过单元测试





**8.在构建过程中出现的问题**

下面的问题：java.lang.NoSuchFieldError: DEFAULT\_USER\_SETTINGS\_FILE

.........

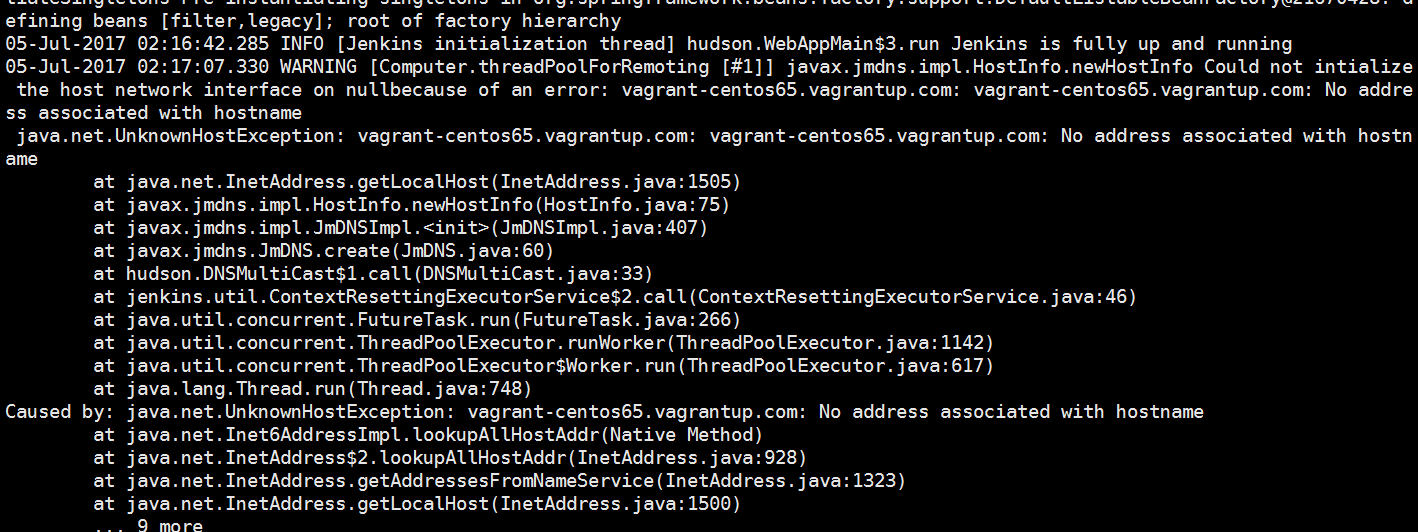
这是jar冲突，版本太高引起的，maven3.5.0版本有这个问题，使用3.3.9低版本就可以了

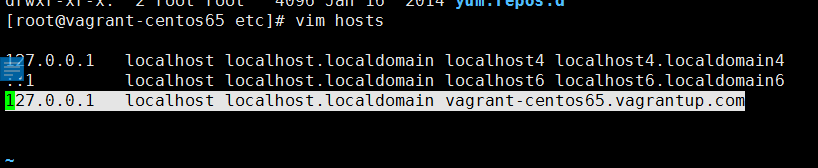


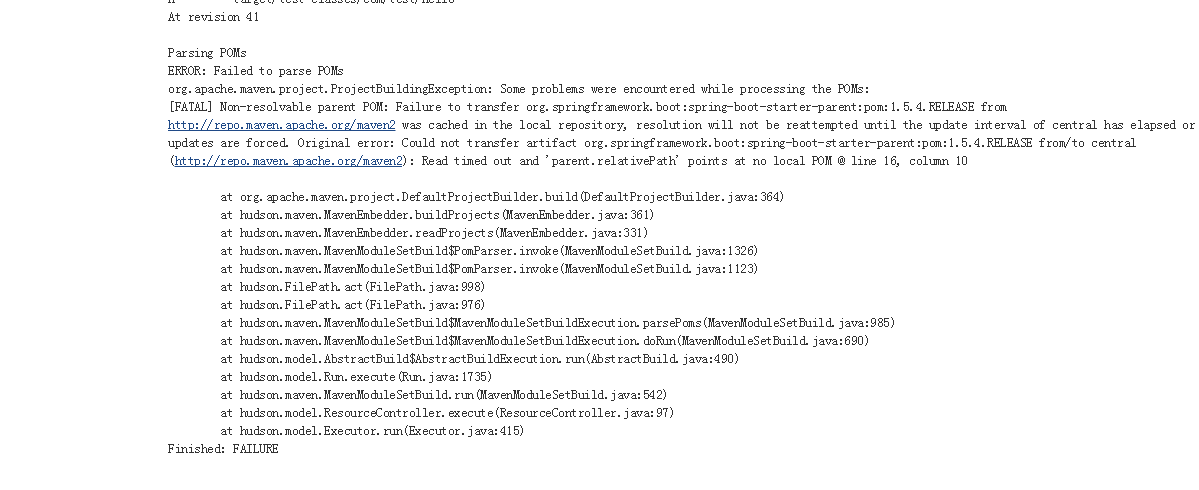
启动时，访问jenkins查看tomcat的日志发现报错。解决办法修改/etc/hosts

添加：

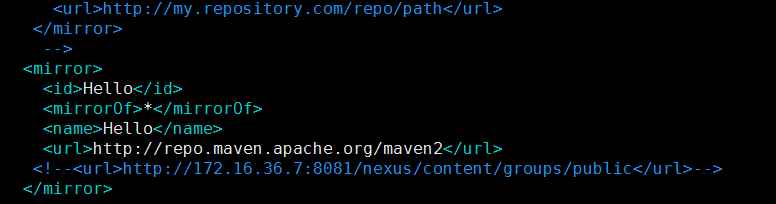
127.0.0.1 localhost localhost.localdomain vagrant-centos65.vagrantup.com





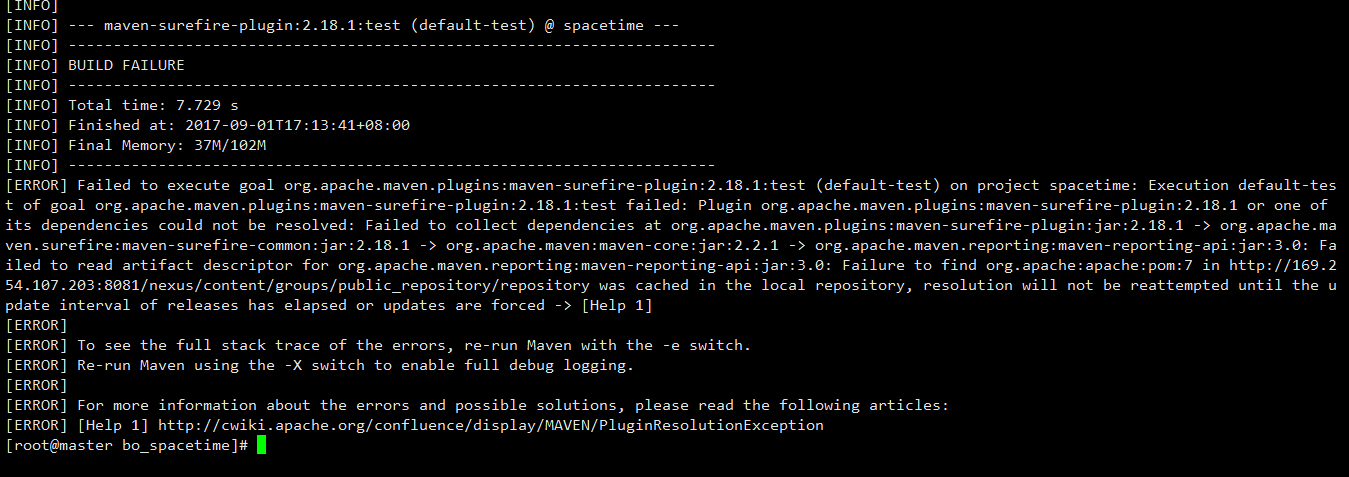


解析pom报错，要记得修改linux中maven的setting.xml目录



同时最好修改<localRepository>/usr/local/maven/repository</localRepository>，maven依赖jar存放目录，给root目录留下足够的空间。

如果无法访问中央仓库下载jar包，私服又不能保证架包完整性，不必再设置上图的<mirror>否则会在构建的时候提示：找不到对应的依赖。



构建成功后是蓝色的球，失败会显示红色。找到/root/.jenkins/workspace目录下把构建成功的项目，放到tomcat的webapps目录下，启动即可。

自启动脚本:

**#!/bin/bash**

**########### 把文件拷贝到tomcat的目录下并重启tomcat ##########**

**#tomcat目录**

**tomcat\_path=/usr/local/tomcat2/apache-tomcat-8.0.44**

**#tomcat端口号**

**tomcat\_port=9080**

**#构建完成后war的目录**

**war\_path=/root/.jenkins/workspace/hello/target**

**#war包名称，即项目名 + "-" + pom文件中的 <version> +".war"**

**war\_name=hello-0.0.1-SNAPSHOT.war**

**now\_time=$(date +"%Y%m%d%H%M%S")**

**echo "jenkins execute shell time is $now\_time, shell begin ...."**

**#获取tomcat进程号**

**tomcat\_pid=`lsof -n -P -t -i :${tomcat\_port}`**

**echo "tomcat process id is ${tomcat\_port}"**

**#判断tomcat是否启动,如果启动的话杀掉该进程**

**if [ "${tomcat\_pid}" != "" ]; then**

**kill -9 $tomcat\_pid**

**echo "kill the server"**

**else**

**echo "tomcat端口号 $tomcat\_port 没有启动,可以继续执行下面操作!"**

**fi**

**# 删除掉tomcat下原来的所有项目**

**rm -rf ${tomcat\_path}/webapps/\***

**#到jenkins工作目录下,拷贝文件到tomcat下**

**cd $war\_path**

**#判断构建完成的war包是否存在**

**if [ -f "$war\_name" ]; then**

**cp -arpf $war\_name $tomcat\_path/webapps**

**else**

**echo "该war包不存在,请检查jenkins 的工作目录/root/.jenkins/workspace/项目名/target/$war\_name"**

**exit 0**

**fi**

**export JAVA\_HOME=/usr/local/java/jdk1.8.0\_131**

**export CATALINA\_2\_HOME=/usr/local/tomcat2/apache-tomcat-8.0.44**

**export CATALINA\_2\_BASE=/usr/local/tomcat2/apache-tomcat-8.0.44**

**$tomcat\_path/bin/./startup.sh**

**##解决脚本执行完毕，jenkins立刻关闭导致tomcat无法启动**

**export BUILD\_ID=doNotKillMe**

**echo "server restarted"**

**#tail -f $tomcat\_path/logs/catalina.out**

**#打印启动日志最后50行**

**Tail -n 50 $tomcat\_path/logs/catalina.out**

**Linux下启动2个tomcat【可直接修改server.xml,不必再到/etc/profile中配置】**

**编辑环境变量：vim /etc/profile**

##########first tomcat###########

CATALINA\_BASE=/usr/local/tomcat/apache-tomcat-8.0.46

CATALINA\_HOME=/usr/local/tomcat/apache-tomcat-8.0.46

TOMCAT\_HOME=/usr/local/tomcat/apache-tomcat-8.0.46

export CATALINA\_BASE CATALINA\_HOME TOMCAT\_HOME

##########first tomcat############

##########second tomcat##########

CATALINA\_2\_BASE=/usr/local/tomcat\_2/apache-tomcat-8.0.46

CATALINA\_2\_HOME=/usr/local/tomcat\_2/apache-tomcat-8.0.46

TOMCAT\_2\_HOME=/usr/local/tomcat\_2/apache-tomcat-8.0.46

export CATALINA\_2\_BASE CATALINA\_2\_HOME TOMCAT\_2\_HOME

##########second tomcat##########

保存退出,执行#source /etc/profile

**来到第二个tomcat的bin目录下，在catalina.sh中加入**

可不配置

###以上三个环境变量就是JAVA的环境变量###

export CATALINA\_BASE=$CATALINA\_2\_BASE

export CATALINA\_HOME=$CATALINA\_2\_HOME

**更改端口：**

第一个tomcat，保持解压后的原状不用修改,

第二个tomcat，修改server.xml配置和第一个不同的启动、关闭监听端口。

修改后示例如下：

　 <Server port="9005" shutdown="SHUTDOWN">　端口：8005->9005

<!-- Define a non-SSL HTTP/1.1 Connector on port 8080 -->

    <Connector port="9080" maxHttpHeaderSize="8192"　       端口：8080->9080

maxThreads="150" minSpareThreads="25" maxSpareThreads="75"

               enableLookups="false" redirectPort="8443" acceptCount="100"

               connectionTimeout="20000" disableUploadTimeout="true" />

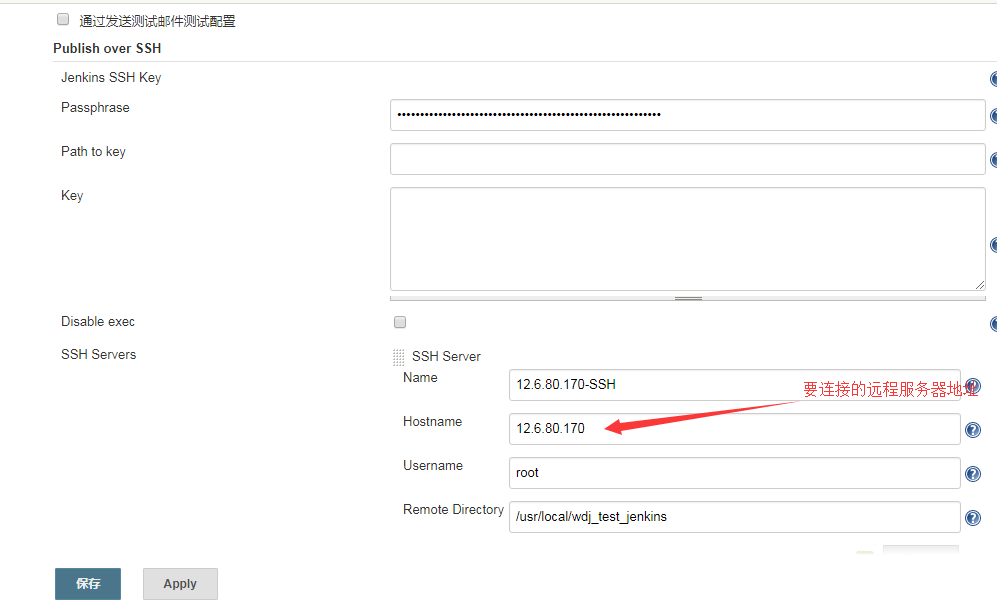
<!-- Define an AJP 1.3 Connector on port 8009 -->

    <Connector port="9009"                                  端口：8009->9009

               enableLookups="false" redirectPort="8443" protocol="AJP/1.3" />

**9.Publish Over SSH插件使用说明**

1. 首先下载**Publish Over SSH**插件。
2. 选择：系统管理=》系统设置。只需配置SSH Servers即可..Passphrase在配置完成后会自动生成。

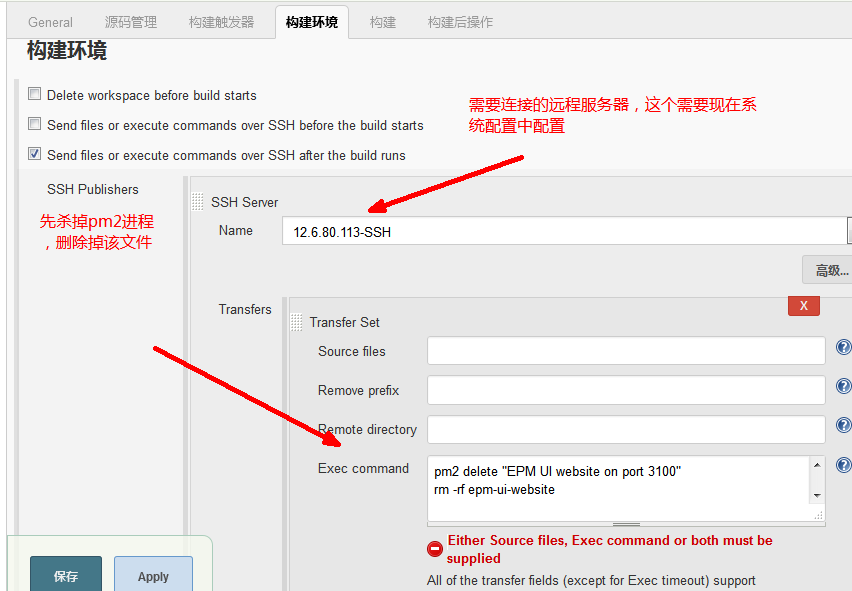


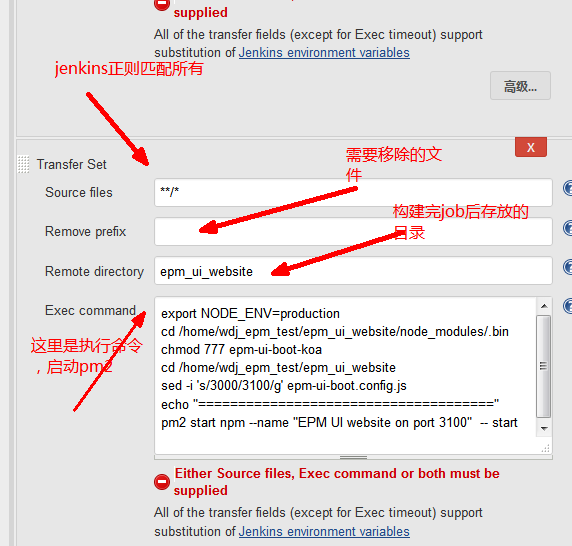


最后点击Test Configuration测试，成功即代表ssh连接成功！

## 10.在Jenkins中配置Job

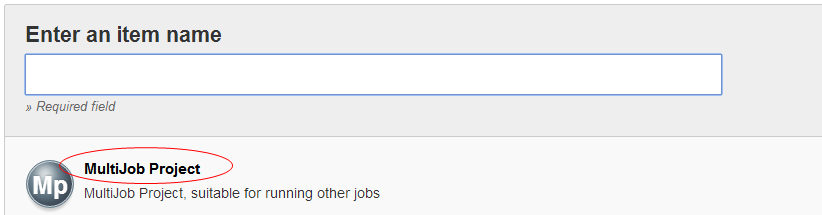
在Job设置中“增加构建步骤”里选择“Send files or execute commands over SSH”，即可以Job中使用Publish over SSH插件，配置如下图：



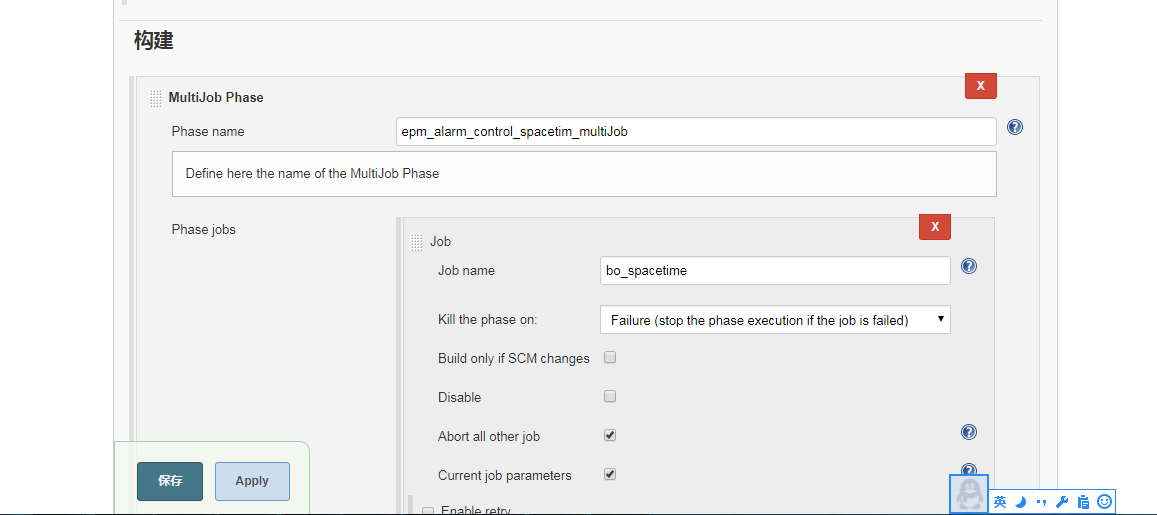


**11.多项目构建**

1. **安装插件：**[Multijob plugin](https://wiki.jenkins-ci.org/display/JENKINS/Multijob+Plugin)
2. 新建MultiJob Project



1. 构建->MultiJob Phase



构建顺序由上往下依次执行（执行顺序不可靠，不是由上往下的）

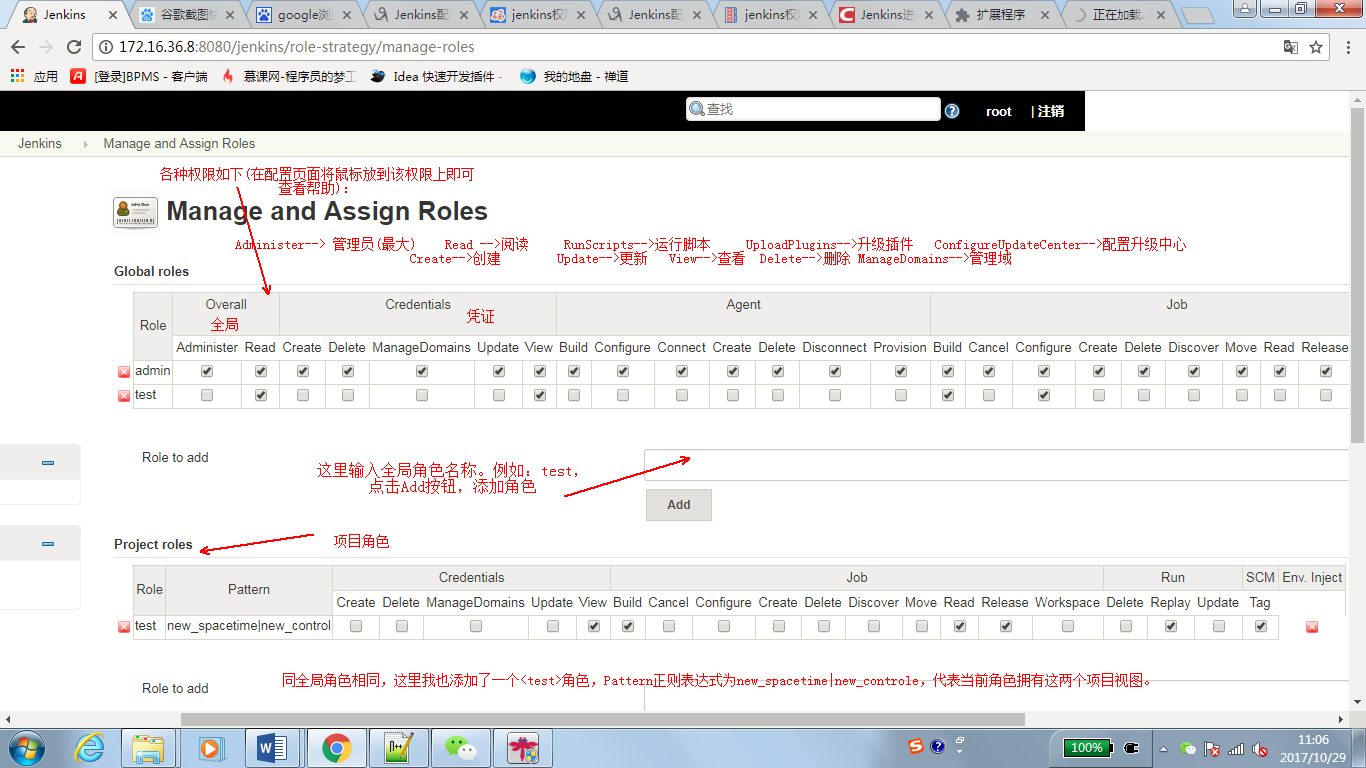
**定时构建**

第一个是代表分钟 H 表示随机   
第二个是代表小时 9-15/4 9点到下午三点期间的每隔4个小时   
第三个是代表天 \* 任意一天   
第四个是代表月份 1-11 表示1到11月份   
第五个是代表星期 1-5 表示工作日

H \*/5 \* \* \* ：代表每隔5小时构建一次

****

**12.Jenkins配置基于角色的项目权限管理**

1. 首先下载Role Strategy Plugin插件。
2. “系统管理”->“Configure Global Security”->策略授权选择“Role-Based Strategy”。
3. “系统管理”->“Manage and Assign Roles”-> Manage Roles设置角色权限
4. 如下图注释所示，配置了全局角色和项目角色，详细说明见图片注释。
5. 
6. 其中有一些比较特别的权限：

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

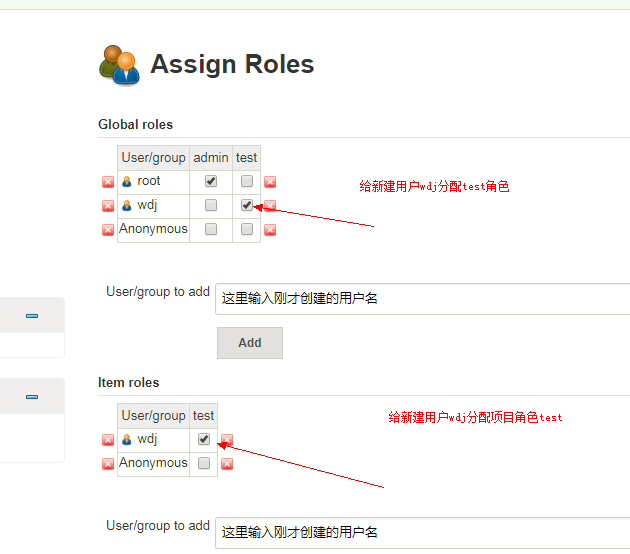
最基本的权限是Overall的Read，用户必须赋予阅读的权限，不然什么都看不到。

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。

1. 项目角色与全局角色的区别就是，项目角色只能管理项目，没有管理jenkins的权限配置。
2. 接下来创建用户：

“系统管理”——>“管理用户”——>“新建用户”。创建用户。

1. 给用户分配角色：
2. “系统管理”->“Manage and Assign Roles”-> Assign Roles设置用户角色

****

1. 使用新建用户wdj登录

****

########### 把文件拷贝到tomcat的目录下并重启tomcat ##########

#!/bin/bash

#tomcat目录

tomcat\_path=/usr/local/tomcat\_2/apache-tomcat-8.0.46

#tomcat端口号

tomcat\_port=9080

#构建完成后war的目录

war\_path=/root/.jenkins/workspace/bo\_spacetime/target

#war包名称，即项目名 + "-" + pom文件中的 <version> +".war"

war\_name=spacetime-0.0.1-SNAPSHOT.war

##解决脚本执行完毕，jenkins立刻关闭导致tomcat无法启动

export BUILD\_ID=doNotKillMe

now\_time=$(date +"%Y%m%d%H%M%S")

echo "jenkins execute shell time is $now\_time, shell begin ...."

#获取tomcat进程号

tomcat\_pid=`lsof -n -P -t -i :${tomcat\_port}`

echo "tomcat process id is ${tomcat\_port}"

#判断tomcat是否启动,如果启动的话杀掉该进程

if [ "${tomcat\_pid}" != "" ]; then

#kill -9 $tomcat\_pid

echo "kill the tomcat server....."

else

echo "tomcat端口号 $tomcat\_port 没有启动,可以继续执行下面操作!"

fi

#到jenkins工作目录下,拷贝文件到tomcat下

cd $war\_path

echo "当前路径: $war\_path"

#判断构建完成的war包是否存在

if [ -f "$war\_name" ]; then

echo "构建构建构建......................................................................."

# 删除掉tomcat下原来的所有项目

#rm -rf ${tomcat\_path}/webapps/spacetime.war

rm -rf ${tomcat\_path}/webapps/spacetime\*

cp -arpf $war\_name $tomcat\_path/webapps/spacetime.war

else

echo "该war包不存在,请检查jenkins 的工作目录/root/.jenkins/workspace/项目名/target/$war\_name"

exit 0

fi

export JAVA\_HOME=/usr/local/java/jdk1.8.0\_131

export CATALINA\_2\_HOME=/usr/local/tomcat\_2/apache-tomcat-8.0.46

export CATALINA\_2\_BASE=/usr/local/tomcat\_2/apache-tomcat-8.0.46

$tomcat\_path/bin/./startup.sh

echo "server restarted.............................................................................."

Jenkins新建视图，选择list，需要新建项目

#打印启动日志最后50行

tail -n 50 $tomcat\_path/logs/catalina.out

npm install pm2 -g：安装pm2

前台构建pm2：

#!/bin/bash

# 安装全局 PM2

# 请先在本地全局安装 pm2 并在全局 node\_modules 目录下（如 `/usr/lib/node\_modules`）执行 `tar -cvzf pm2.tar.gz pm2` 命令打包 pm2，将打包好的 pm2.tar.gz 文件与此脚本一同放置

tar -xvzf pm2.tar.gz -C /usr/local/node/lib/node\_modules/

**ln -s /usr/local/node/lib/node\_modules/pm2/bin/pm2 /usr/bin/pm2：pm2找不到**

ln -s /usr/local/node/lib/node\_modules/pm2/bin/pm2-dev /usr/bin/pm2-dev

ln -s /usr/local/node/lib/node\_modules/pm2/bin/pm2-docker /usr/bin/pm2-docker

ln -s /usr/local/node/lib/node\_modules/pm2/bin/pm2-runtime /usr/bin/pm2-runtime

/usr/bin/env: node: 没有那个文件或目录

ln -s /usr/local/node/bin/node /usr/bin/node 建立软连接

[PM2][ERROR] script not found : /home/spacetime\_portal/epm\_ui\_website/npm

script not found : /home/spacetime\_portal/epm\_ui\_website/npm

ln -s /usr/local/node/bin/npm /usr/bin/npm：建立npm软连接

使用你Web容器的管理工具设置JENKINS\_HOME环境参数.

打开tomcat的bin目录，编辑catalina.sh文件。  
在# OS specific support. $var \_must\_ be set to either true or false.上面添加：export JENKINS\_HOME="目录路径"

在引号中填入你的路径。

###################################安装node############

1）上传服务器可以是自己任意路径，目前我的放置路径为  cd /app/software/

2）解压上传（解压后的文件我这边将名字改为了nodejs，这个地方自己随意，只要在建立软连接的时候写正确就可以）

    ① tar -xvf   node-v6.10.0-linux-x64.tar.xz

    ② mv node-v6.10.0-linux-x64  nodejs

    ③确认一下nodejs下bin目录是否有node 和npm文件，如果有执行软连接，如果没有重新下载执行上边步骤；

3）建立软连接，变为全局

   ①l ln -s /app/software/nodejs/bin/npm /usr/local/bin/

   ②l ln -s /app/software/nodejs/bin/node /usr/local/bin/

4）最后一步检验nodejs是否已变为全局

5）在/etc/profile文件中添加环境变量

export NODE\_HOME=/usr/local/node/v8.5.0

export PATH=$NODE\_HOME/bin:$PATH

因为要部署应用的机器没有公网连接，所以直接npm install pm2是不可能了， 简单记录一下怎么离线安装pm2

首先，找一台可以在线安装pm2的机器

执行 npm install pm2 -g 安装pm2

执行 npm config get prefix， 看一下这台机器的npm默认全局安装目录在哪，然后去这个目录下面的 lib/node\_modules/下就能看到你刚刚安装的pm2程序；例如如果显示 /usr/local/node， 则去 /usr/local/node/lib/node\_modules/下面找到pm2

执行tar czvf pm2.tar.gz pm2/   命令， 把pm2打包，在这里打包的好处，是因为该目录下面已经包含了pm2依赖的所有node模块

把打出来的包，通过内网传送到要部署的服务器上

登录部署服务器

同样执行 npm config get prefix 看一下这台服务器的npm默认安装目录， 例如目录为 /usr/local/node

执行命令  cd /usr/local/node/lib/node\_modules/

拷贝你的 pm2.tar.gz 到该目录下

tar xvf pm2.tar.gz

到这一步，你已经可以使用 /usr/local/node/lib/node\_modules/pm2/bin/pm2 执行pm2的命令了， 下面就是把这个命令加到系统环境中

cd /usr/bin

ln -s /usr/local/node/lib/node\_modules/pm2/bin/pm2 pm2

这样，就可以直接使用pm2命令来各种操作了

ps：其实往部署服务器上复制pm2时，你可以复制到任何目录中都是可以的，但是为了大家好维护，还是放在约定的目录下比较好