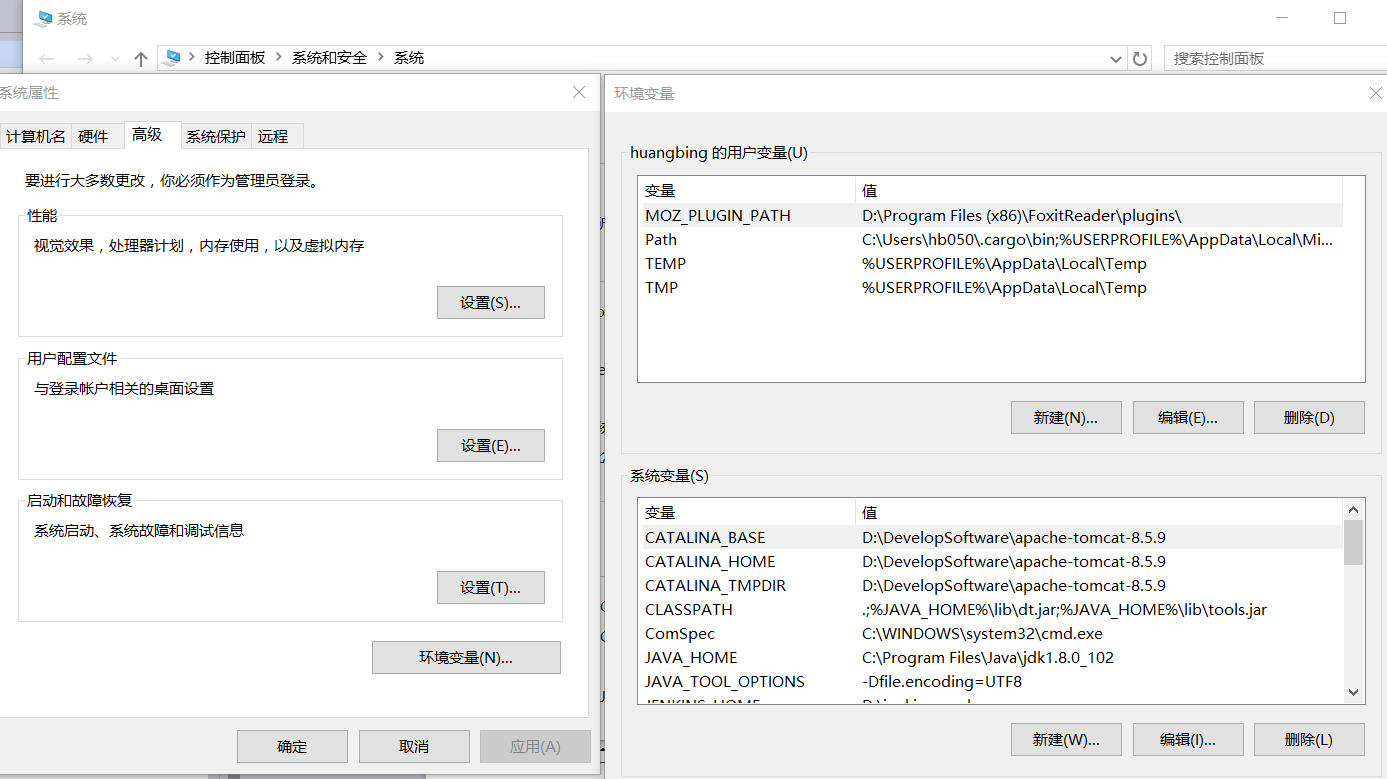
Jenkins部署环境及配置

# 系统环境配置

打开系统环境配置窗口，新增系统环境变量



## 1.1 tomcat 系统环境配置

### 1.1.1新增环境变量：

CATALINA\_HOME是TOMCAT安装路径的别名，目的是为了方便使用TOMCAT

CATALINA\_BASE=D:\DevelopSoftware\apache-tomcat-8.5.9

CATALINA\_HOME=D:\DevelopSoftware\apache-tomcat-8.5.9

CATALINA\_TMPDIR=D:\DevelopSoftware\apache-tomcat-8.5.9

TOMCAT\_HOME=D:\DevelopSoftware\apache-tomcat-8.5.9

### 1.1.2 path节点新增环境变量：

在环境变量Path中，新增加“%CATALINA\_HOME%\bin\ "、” %CATALINA\_HOME%\lib\”.

## 1.2 java环境配置

### 1.2.1新增环境变量：

CLASSPATH=.;%JAVA\_HOME%\lib\dt.jar;%JAVA\_HOME%\lib\tools.jar

JAVA\_HOME=C:\Program Files\Java\jdk1.8.0\_102

JAVA\_TOOL\_OPTIONS=-Dfile.encoding=UTF8

### 1.2.2 path节点新增环境变量：

在环境变量Path中，新增加“C:\ProgramFiles\Java\jdk1.8.0\_102\bin"

## 1.3 jenkins 环境配置

### 1.3.1新增环境变量：

JENKINS\_HOME=D:\jenkins-work

备注：D:\jenkins-work为jenkins的新增默认工作目录，jenkins启动之后，新增的系统配置与job任务的相关数据，将会在本目录生成。

### 1.3.2 path节点新增环境变量：

无。

## 1.4 Maven 环境配置

### 1.4.1新增环境变量：

MAVEN\_HOME=D:\DevelopSoftware\apache-maven-3.3.9

### 1.4.2 path节点新增环境变量：

在环境变量Path中，新增加“D:\DevelopSoftware\apache-maven-3.3.9\bin"

# 1.5 git环境配置

### 1.4.1新增环境变量：

无。

### 1.4.2 path节点新增环境变量：

在环境变量Path中，新增：

“D:\DevelopSoftware\Git\cmd"

“D:\DevelopSoftware\Git\bin"

“D:\DevelopSoftware\Git\mingw64\libexec\git-core"

# Tomcat部署配置

tomcat容器初始化

## 2.1 tomcat-users.xml文件

<role rolename="tomcat"/>

<role rolename="admin-gui"/>

<role rolename="manager-gui"/>

<role rolename="manager-script"/>

<role rolename="manager-jms"/>

<role rolename="manager-status"/>

<role rolename="role1"/>

<user username="admin" password="admin" roles="tomcat,role1,admin-gui,manager-gui,manager-script,manager-script,manager-jms,manager-status"/>

参考文件： 

## 2.2 server.xml

新增“useBodyEncodingForURI="true" ”，解决jenkins控制台乱码问题。

<Connector port="8888" useBodyEncodingForURI="true" protocol="HTTP/1.1"

connectionTimeout="20000" redirectPort="8443" />

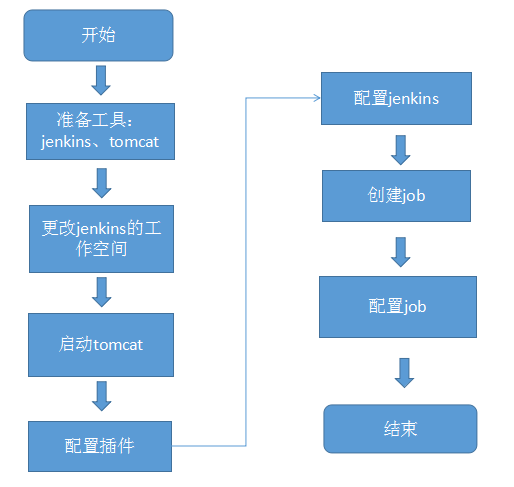
# Jenkins安装与配置

Jenkins下载路径：<http://updates.jenkins-ci.org/download/war/>

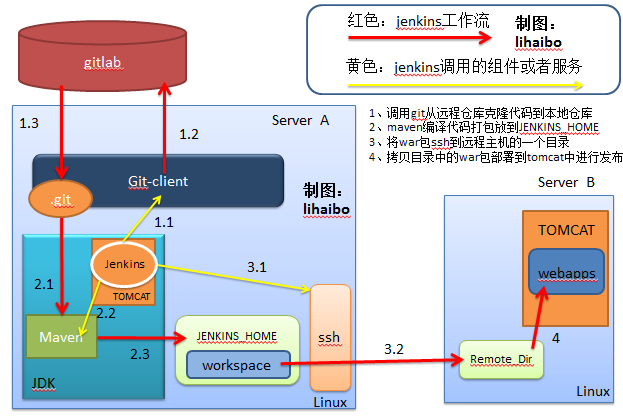
Jenkins安装：将Jenkins.war部署到tomcat/webapps目录，启动tomcat，即可启动jenkins。

Jenkins启动之后，需要安装插件，增加java、git、maven等软件关联配置，ssh服务器配置；

总体流程如下：

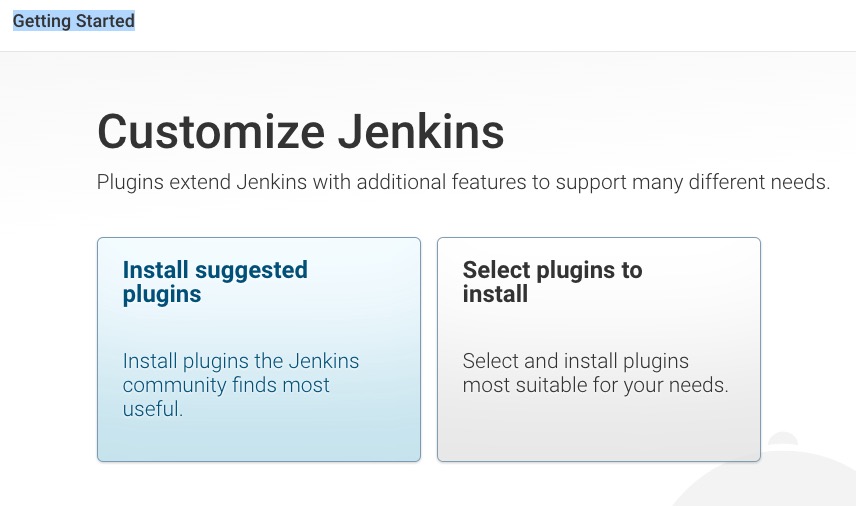


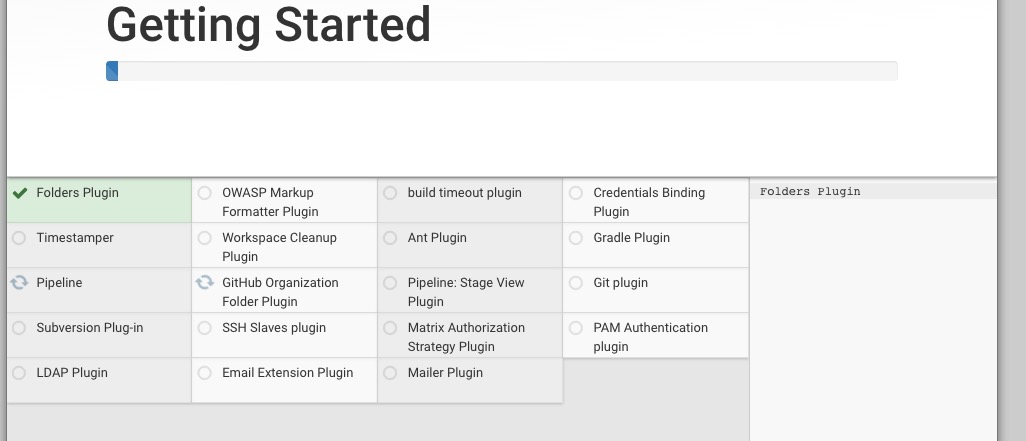
Jenkins服务流程图：



## 3.1 jenkins插件

Jenkins第一次启动时，会展示选择安装插件界面，选择安装插件的选项安装“Install suggested plugins”。





## 3.2 jenkins补充插件

### 3.2.1 [补充插件](http://www.cnblogs.com/zz0412/p/jenkins_jj_10.html)

* **Publish Over SSH 插件**
* [**Deploy to container Plugin**](http://blog.csdn.net/houyefeng/article/details/50996198) **插件**

### 3.2.2 手动安装插件

参考网页：<http://www.imooc.com/article/11895>

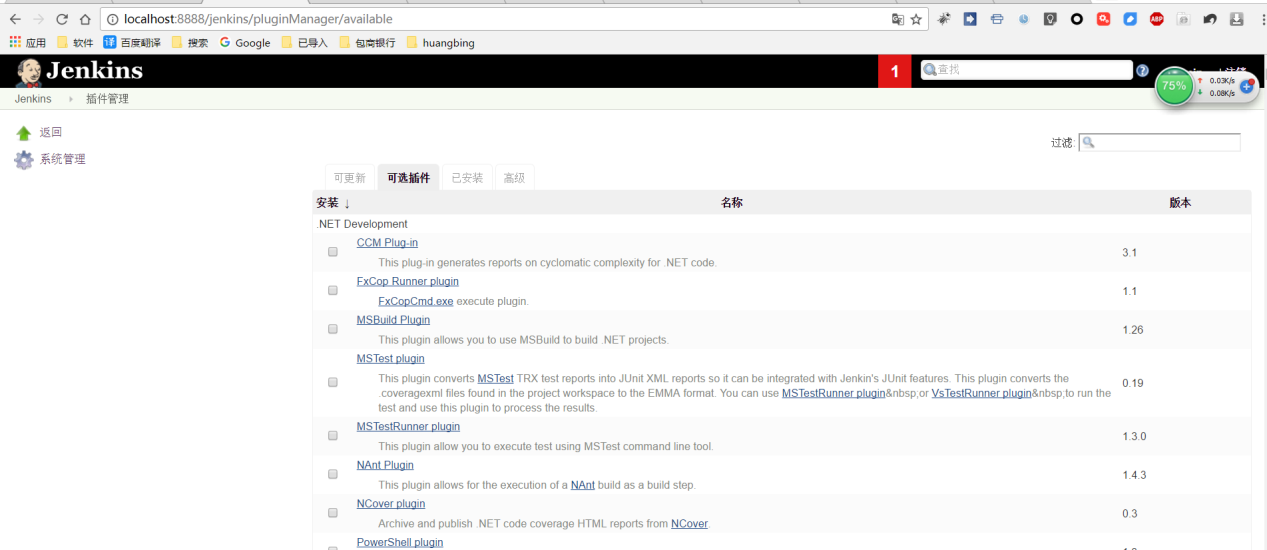
**1、安装插件方式选择**

如果在安装时，没有安装好对应的插件可以选择如下方式进行插件的安装：

**方法一.如果服务器可以上网,那边选择在线安装最好不过了,安装流程为:  
 系统管理(Configure System)----插件管理(Manage Jenkins)---选择需要的插件直接安装即可;**

Jenkins在线安装插件界面：

http://localhost:8888/jenkins/pluginManager/available



**方法二.如果服务器不能上网,那么就只能离线安装**

首先去<http://updates.jenkins-ci.org/download/plugins/>

下载需要的plugin,选择匹配的版本号,下载到本地,然后打开:系统管理()---插件管理—高级---找到”上传插件”(浏览,找到扩展名为.hpi的插件，上传之后默认直接就安装了。重启jenkins，安装的插件就可以使用了。

**2、安装插件依赖解决**

插件安装不上去，一定是某个地方出现问题。在Jenkins的终端日志中，可以查看到对应的错误。需要针对性的解决。

比如：在安装subversion的时候，的错误提示。  
 一般的情况安装对应的插件，会把需要依赖的插件安装上。如出现对应的依赖问题，就需要一一安装上对应的依赖包。如下错误中，我们就需要先安装好这两个依赖包。  
 mapdb-api (1.0.1.0), workflow-scm-step (1.4.3)

**3、查看已经安装的插件**

为了更好的方便我们学习下节课内容，请同学们通过插件管理(Manage Jenkins)---已安装插件(Installed)，确保安装好了如下的插件：

Subversion Plug-in Jenkins对SVN的支持。

Maven Integration plugin 用于对java代码的构建

Email Extension Plugin 邮件模块

Ant Plugin 也是对java代码构建，不过是通过ant来作。

还有一些是安装jenkins的时候推荐安装的，大家也最好安装上。

## 3.3 工具软件配置

进入工具软件配置界面路径：

**系统管理(Configure System)----全局工具配置(**[**Global Tool Configuration**](http://localhost:8888/jenkins/configureTools)**);**

**网页Url：<http://localhost:8888/jenkins/configureTools/>**

特别注意：“自动安装”选项：会自动从网上下载软件。一般不需要选中此配置。直接使用本地已安装同类软件即可。

### 3.3.1 JDK

配置项：

|  |  |
| --- | --- |
| 别名 | jdk108 |
| JAVA\_HOME | C:\Program Files\Java\jdk1.8.0\_102 |

### 3.3.2 Git

配置项：

|  |  |
| --- | --- |
| Name（别名） | Default |
| Path to Git executable | D:\DevelopSoftware\Git\bin\git.exe |

### 3.3.3 Maven

配置项：

|  |  |
| --- | --- |
| Name（别名） | apache-maven-3.3.9 |
| MAVEN\_HOME | D:\DevelopSoftware\apache-maven-3.3.9 |

## 3.4 系统配置

进入工具软件配置界面路径：

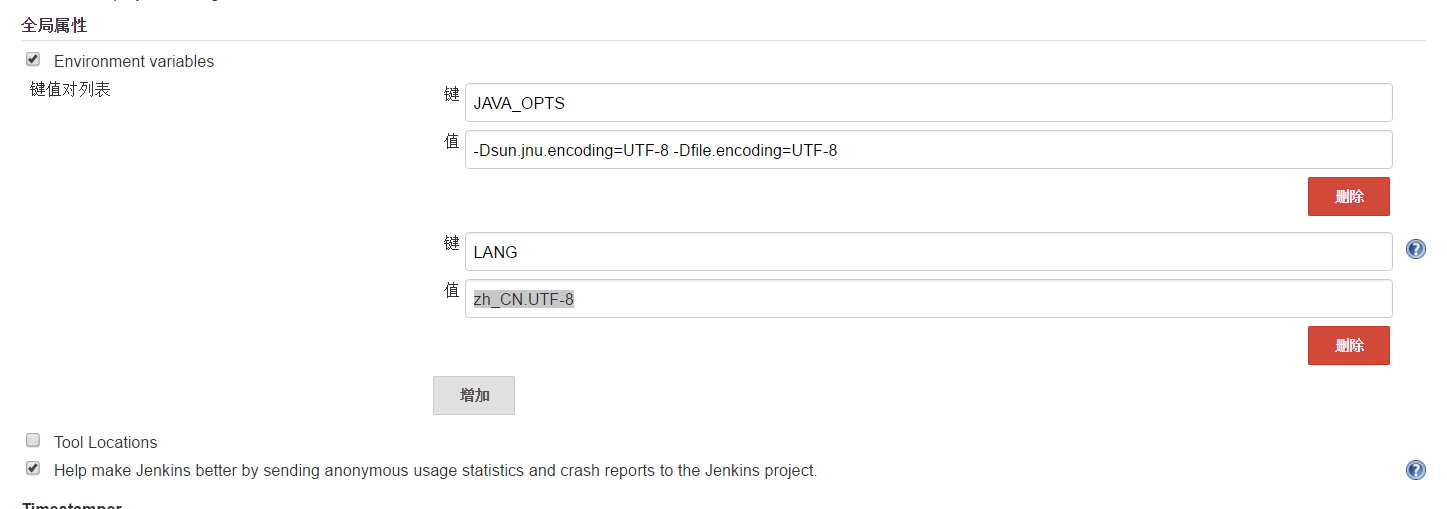
**系统管理(Configure System)----**[**系统设置**](http://localhost:8888/jenkins/configure)**;**

**网页Url：<http://localhost:8888/jenkins/configure>**

### 3.4.1 “全局属性”配置项

选中“Environment variables”菜单，新增以下配置：

|  |  |
| --- | --- |
| 键1 | JAVA\_OPTS |
| 值1 | -Dsun.jnu.encoding=UTF-8 -Dfile.encoding=UTF-8 |
| 键2 | LANG |
| 值2 | zh\_CN.UTF-8 |

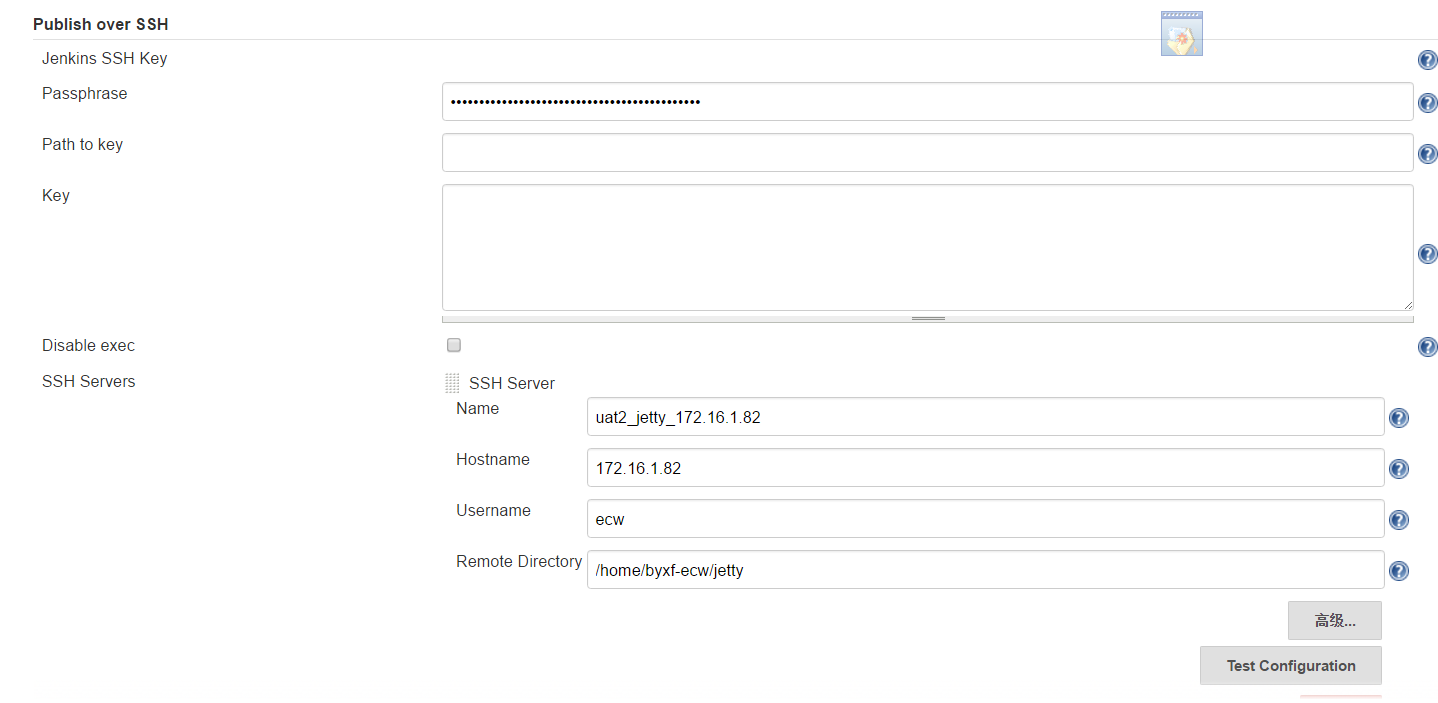


### 3.4.1 “Publish over SSH”配置项

相关插件属性说明，请查阅网页：

<http://blog.csdn.net/houyefeng/article/details/51027885>

以ecw系统配置为例，配置项如下：



默认SSH Server 配置1：

|  |  |
| --- | --- |
| Passphrase (密码) | byxfuat |
| Name | uat2\_jetty\_172.16.1.82 |
| Hostname | 172.16.1.82 |
| Username | ecw |
| Remote Directory | /home/byxf-ecw/jetty |
| Port | 22 |

SSH Server 配置2：

|  |  |
| --- | --- |
| Passphrase | byxfuat |
| Name | uat2\_ecw\_tomcat\_172.16.1.82 |
| Hostname | 172.16.1.82 |
| Username | ecw |
| Remote Directory | /home/byxf-ecw/tomcat |
| Port | 22 |

SSH Server 配置3： （不同服务器的配置样例）

需要勾选：“Use password authentication, or use a different key”选项。

|  |  |
| --- | --- |
| Passphrase(密码) | byxfuat |
| Name | uat2\_jetty\_172.16.1.81 |
| Hostname | 172.16.1.81 |
| Username | ecw |
| Remote Directory | /home/byxf-ecw/jetty |
| Port | 22 |

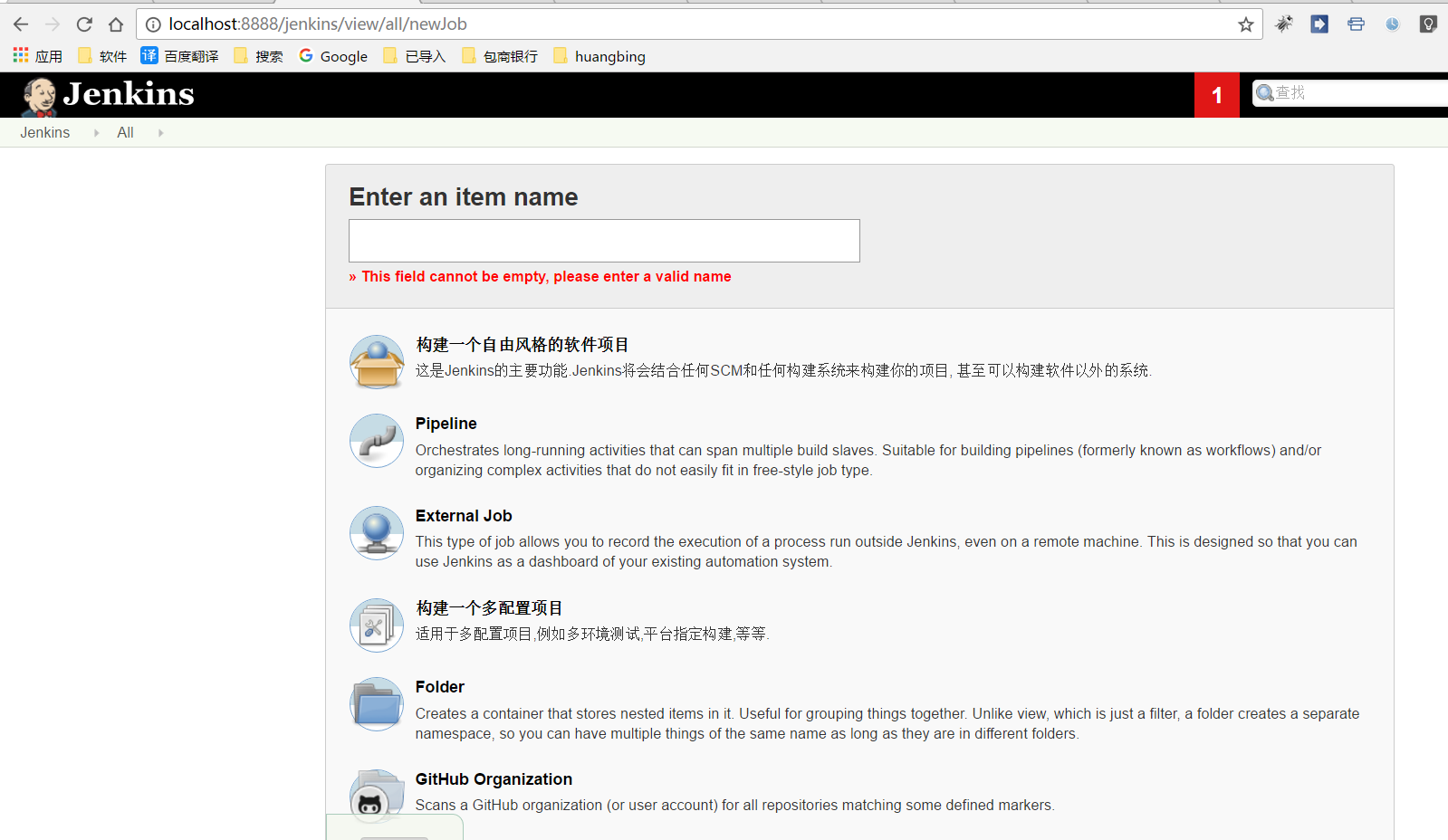
## 3.5 新建job任务

参考网页：<http://www.mamicode.com/info-detail-1153476.html>

进入工具软件配置界面路径：

**系统管理(Configure System)----** [**新建**](http://localhost:8888/jenkins/view/all/newJob)**;**

**网页Url：<http://localhost:8888/jenkins/view/all/newJob>**



### 3.5.1 新建任务

|  |  |
| --- | --- |
| 任务名称 | uat2\_ecw\_core |
| 任务类别 | 构建一个自由风格的软件项目 |

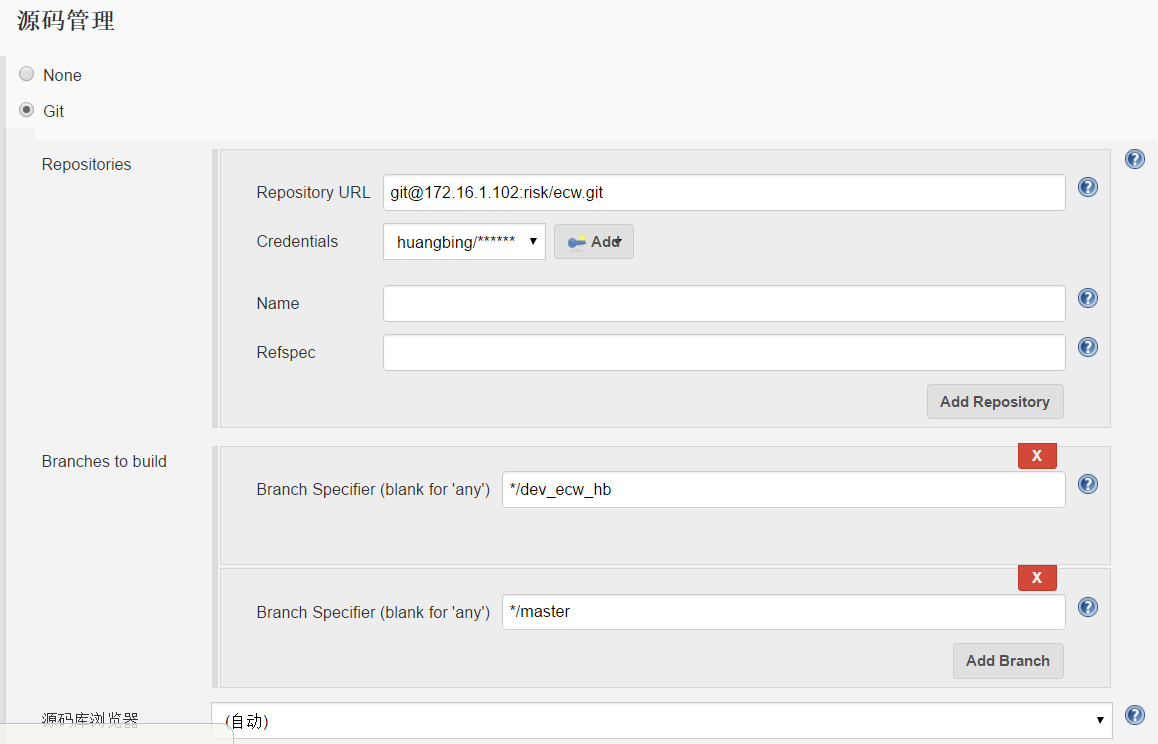
### 3.5.2 通用配置（General）



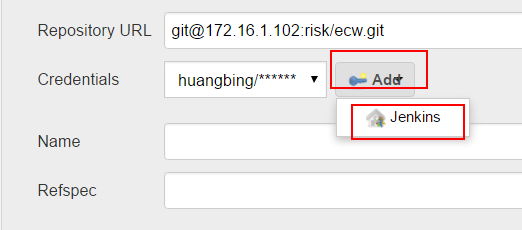
### 3.5.3 源码管理

* 3.5.3.1选中git构建工具选项。

|  |  |
| --- | --- |
| Repository URL(源代码地址) | git@172.16.1.102:risk/ecw.git |
| Credentials(证件) | 点击“ADD”按钮，新增源代码服务器用户名和密码 |
| Branch Specifier (blank for 'any')  (git代码分支，可新增多个) | \*/dev\_ecw\_hb |



* 3.5.3.2新增代码服务器账号（ADD Credentials）





Credentials管理页面路径：

**系统管理(Configure System)----** [Configure Credentials](http://localhost:8888/jenkins/configureCredentials)---- [Credentials](http://localhost:8888/jenkins/credentials)**;**

**网页Url：<http://localhost:8888/jenkins/credentials/>**

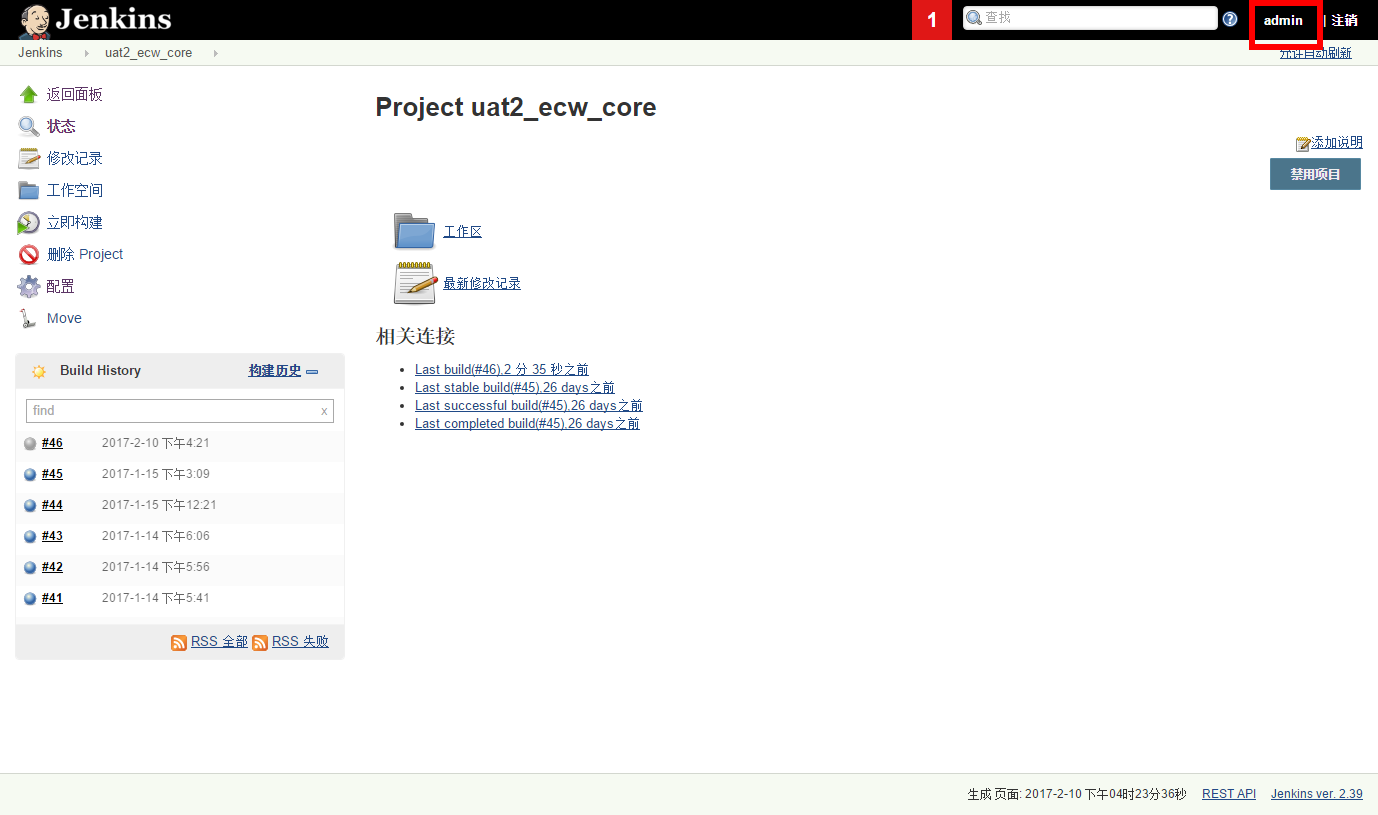
**问题：Linux系统Repository URL(源代码地址)拉去失败，权限问题**

****

**解决方法：**

**在Jenkins服务器上生成jenkins登陆用户的ssh-key，然后配置到Gitlab中，具体方法：**

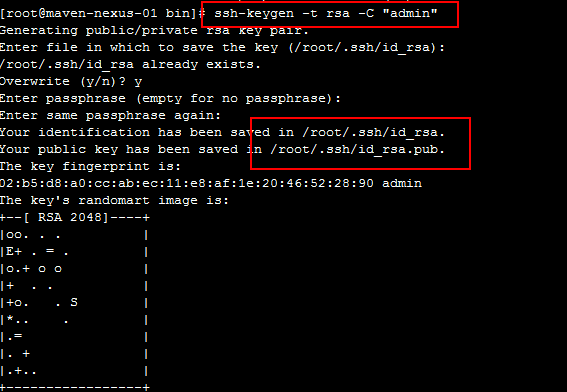
1. **确认Jenkins登陆用户**

****

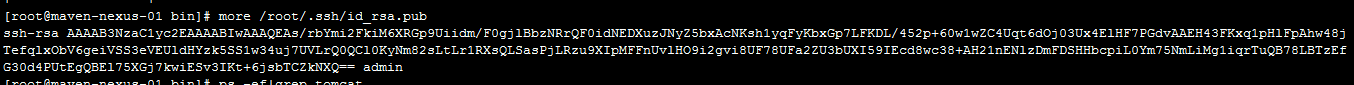
1. **登陆jenkins服务器，执行以下操作命令：**

**[root@maven-nexus-01 bin]# ssh-keygen -t rsa -C "admin"**

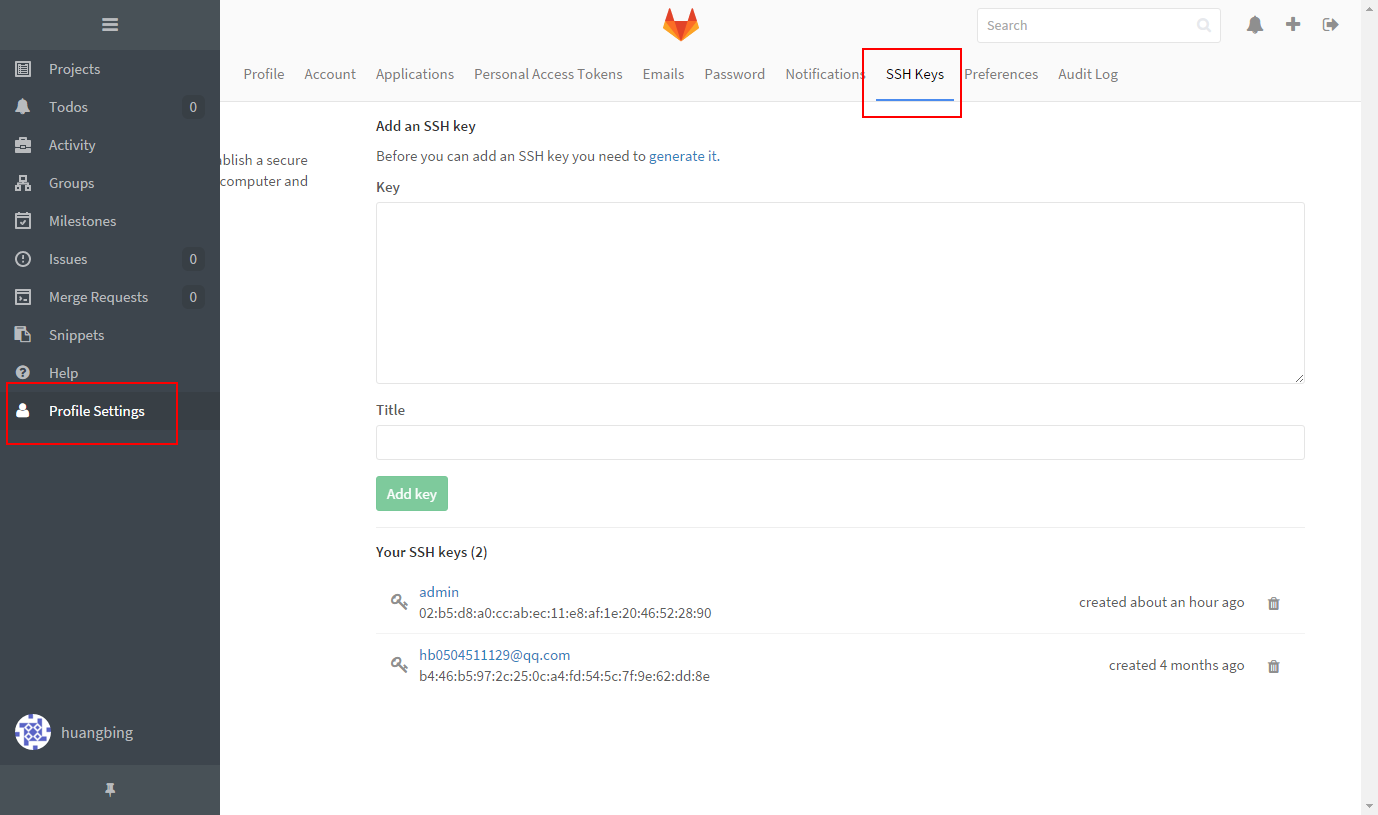
**然后一路回车生成两个密钥文件：**

****

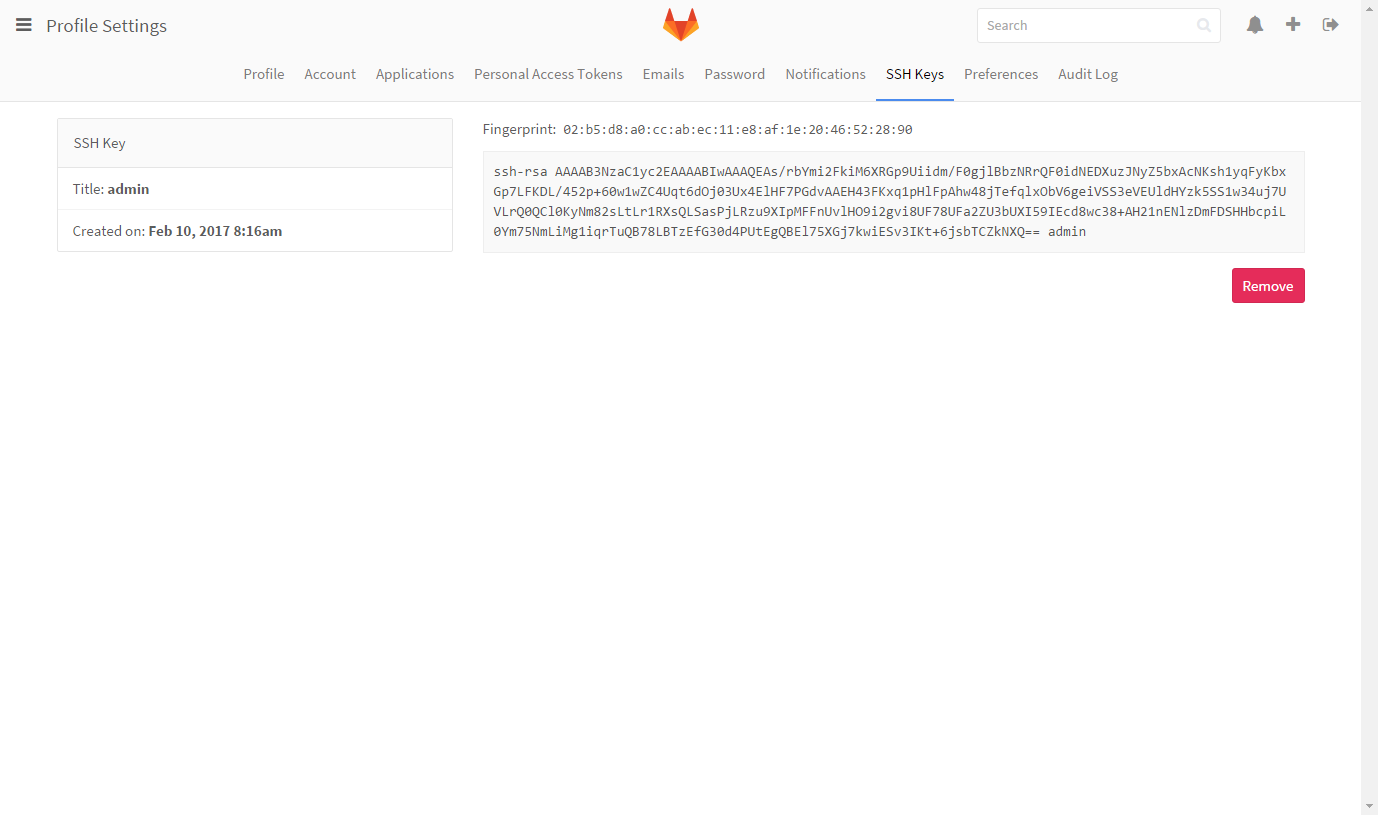
1. **查看私钥文件并复制**

****

1. **登陆Gitlab，并将ssh-key值配置到用户ssh key中，注意:配置的用户为3.5.3.2的用户，具体方法**

****

**然后粘贴服务器生成的key值到gitlab中，保存！**

****

### 3.5.4 构建触发器

略.

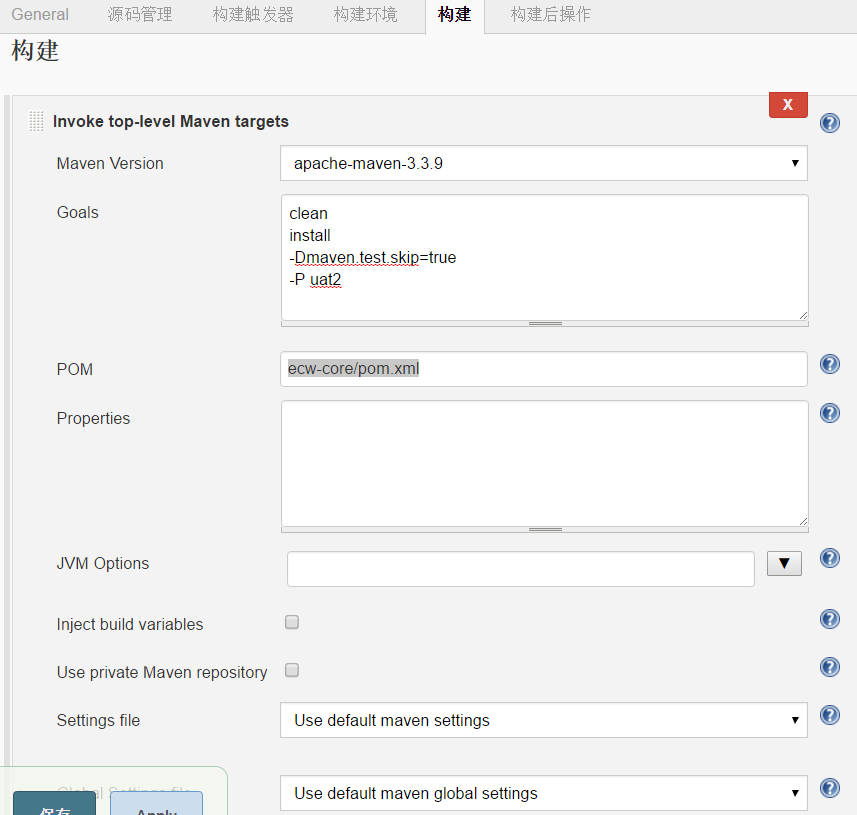
### 3.5.4 构建环境

略.

### 3.5.5 构建

增加构建步骤，选择“Invoke top-level Maven targets”选项。

|  |  |
| --- | --- |
| Maven Version | apache-maven-3.3.9 |
| Goals | clean  install  -Dmaven.test.skip=true  -P uat2 |
| POM | ecw-core/pom.xml |



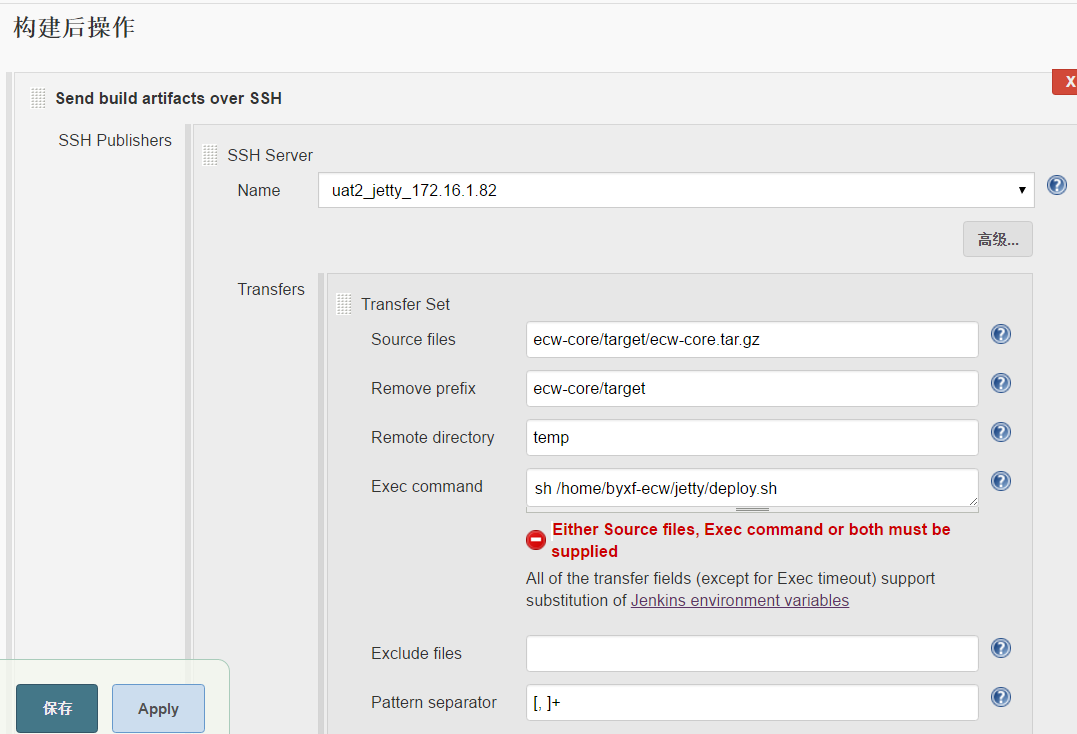
### 3.5.5 构建后操作

增加构建后操作步骤，选择“Send build artifacts over SSH”选项。

配置 SSH Server：(可以配置多个Transfer Set提交项)

|  |  |
| --- | --- |
| Name | uat\_jetty\_172.16.1.82 |
| Source files(待上传文件) | ecw-core/target/ecw-core.tar.gz |
| Remove prefix  (移除目录(只能指定Transfer Set Source files中的目录)) | ecw-core/target |
| Remote directory(远程目录) | temp |
| Exec command(待执行脚本命令) | sh /home/byxf-ecw/jetty/deploy.sh |

参考网页：<http://www.cnblogs.com/zz0412/p/jenkins_jj_10.html>



### 3.5.6 保存配置

保存以上配置。

### 3.5.6 构建项目与查看控制台

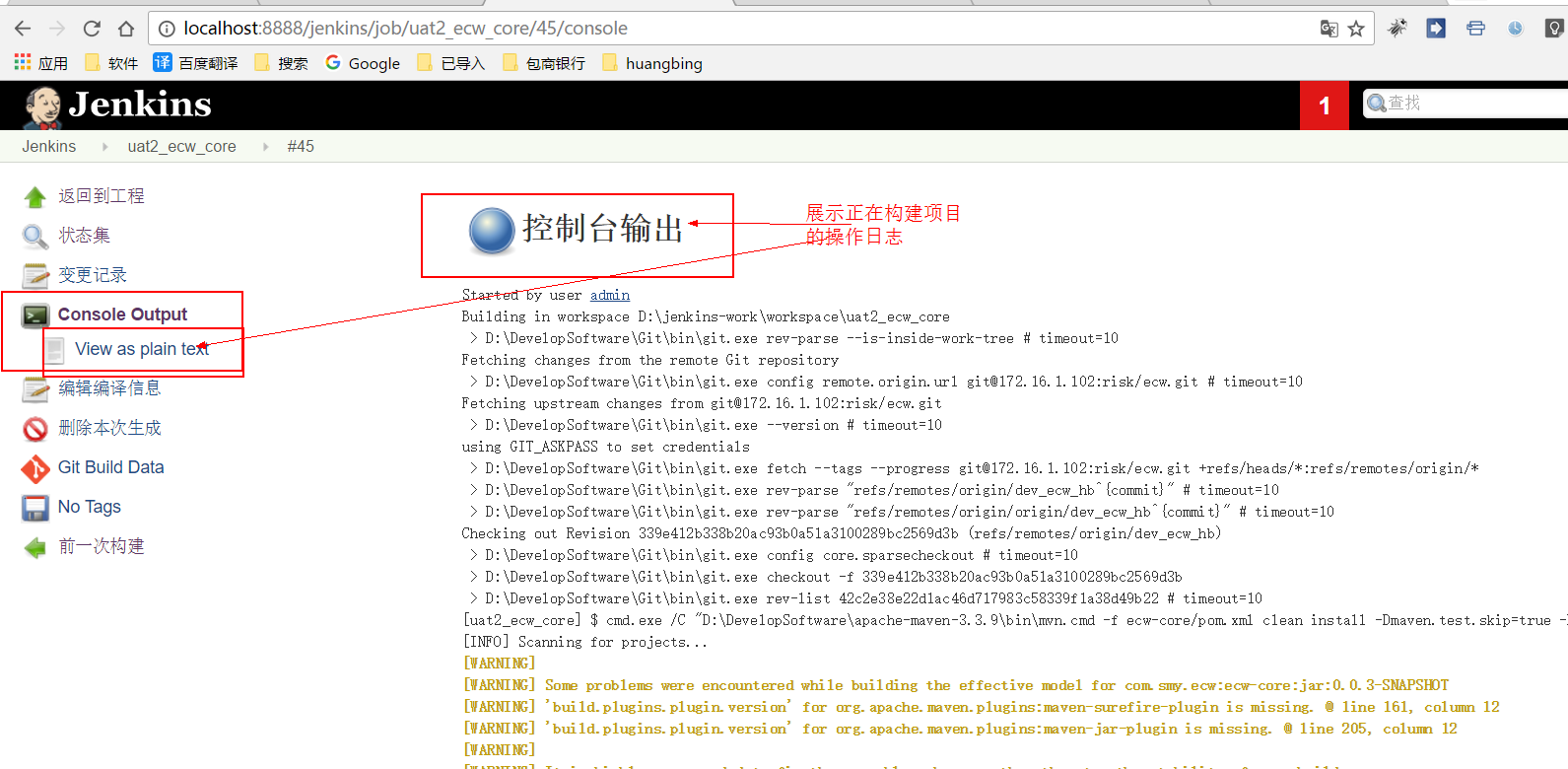
3.5.6.1构建



3.5.6.1 查看控制台







# 服务器自动部署脚本

## 4.1参考部署脚本

已ecw部署jetty与tomcat为例。



### 4.1.1 deploy\_ecw-core.sh

|  |
| --- |
| #!/bin/bash  #这步很关键，引入系统java环境变量  source /etc/profile  #export BASH\_ENV=/etc/profile  #export HOME=`/home/sybase`  ###!/usr/bin/env bash  ##export BASH\_ENV=/etc/profile  #export LANG=”zh\_CN.UTF-8″  export LANG="en\_US.UTF-8"  #deploy.sh 样例  #进入jetty安装目录  jettyFile="/home/byxf-ecw/jetty"  cd $jettyFile  echo "Enter the jetty installation directory"  #备份部署包  coreTarFile="ecw-core.tar.gz"  #read coreTarFile  if [ -f $coreTarFile ];  then  #备份部署包到bak目录中  bakDir="${jettyFile}/bak"  if [ -d $bakDir ];  then  echo "${bakDir} exists"  else  mkdir "${bakDir}"  echo "create ${bakDir} successfully."  fi    #备份部署包成功${bakFileName}  bakFileName="${coreTarFile}\_`date +%Y%m%d%H%M%S`.bak"  mv "${jettyFile}/${coreTarFile}" "${bakDir}/${bakFileName}"  echo "Backup deployment package from "${jettyFile}/${coreTarFile}" to ${bakDir}/${bakFileName} successfully"  else  #$coreTarFile 文件不存在,不做备份处理  echo "${coreTarFile} File does not exist, do not do backup processing"  fi  #删除旧部署包  binFile="${jettyFile}/ecw-core/bin"  if [ -d $binFile ];  then  echo "${binFile} exists"    #进入jetty运行目录$binFile，关闭当前正在运行的jetty；  cd $binFile  echo "Enter the jetty run directory, close the currently running jetty"    dos2unix stop.sh  echo "dos2unix: converting file stop.sh to UNIX format ..."    #执行./stop.sh文件，关闭当前运行jetty程序成功  ./stop.sh  echo "Execute the./stop.sh file and close the current running jetty program successfully."    #删除原部署文件目录${jettyFile}/ecw-core 成功。  rm -rf $jettyFile/ecw-core  echo "Delete the original deployment file directory ${jettyFile}/ecw-core successfully."  else  #${binFile} 文件夹不存在  echo "${binFile} folder does not exist"  fi  #回退到jetty安装目录${jettyFile}。  cd $jettyFile  echo "Back to the jetty installation directory ${jettyFile}"  #开始迁移新部署包  echo "Start migrating a new deployment package"  tempTarFile="${jettyFile}/temp/ecw-core.tar.gz"  if [ -f $tempTarFile ];  then  echo "${tempTarFile} exists"  #移动新部署包ecw-core.tar.gz从jetty/temp目录到jetty目录下成功。  mv "${tempTarFile}" "${jettyFile}/${coreTarFile}"  echo "Move the new deployment package ecw-core.tar.gz from the jetty/temp directory to the jetty directory."    else  echo "${tempTarFile} folder does not exist"  fi  #部署新tar包  if [ -f $coreTarFile ];  then  echo "${coreTarFile} is exist !"  #${coreTarFile}解压新部署tar包成功.  tar -xvf $coreTarFile  echo "${coreTarFile} unpack new deployment tar package successfully"    #进入jetty运行目录${binFile}，启动jetty.  cd $binFile  echo "Enter jetty run directory ${binFile}, start jetty."    dos2unix stop.sh  echo "dos2unix: converting file stop.sh to UNIX format ..."    dos2unix start.sh  echo "dos2unix: converting file start.sh to UNIX format ..."    #以调试模式运行  ./start.sh debug  #生产模式运行  #./startup.sh  echo "Start start.sh script successfully!"  else  echo "The ${coreTarFile} file does not exist and cannot be redeployed"  fi  #deploy.sh脚本执行完成!  echo " The deploy.sh script is finished!"  exit 0 |

### 4.1.1 deploy\_ecw-oms.sh

|  |
| --- |
| #!/bin/bash  source /etc/profile  ###!/usr/bin/env bash  #export BASH\_ENV=/etc/profile  #export HOME=`/home/sybase`  export LANG=”zh\_CN.UTF-8″  #export LANG="en\_US.UTF-8"  #deploy.sh 样例  #进入tomcat安装目录  tomcatFile="/home/byxf-ecw/tomcat"  #cd $tomcatFile  #echo "Enter the tomcat installation directory"  #进入tomcat/bin目录  binDir="${tomcatFile}/bin"  cd $binDir  echo "Into the tomcat/bin directory:(${binDir})."  #tomcat/bin目录下，执行shutdown.sh命令，停止web服务  ./shutdown.sh  echo "Tomcat/bin directory, run the shutdown.sh command, stop Web Service." .  #进入oms包部署目录webapps  webapps="${tomcatFile}/webapps"  cd $webapps  echo "into the OMS package deployment directory:(${webapps})"  #备份部署包到bak目录中  bakDir="${webapps}/bak"  if [ -d $bakDir ];  then  echo "${bakDir} exists"  else  mkdir "${bakDir}"  echo "create ${bakDir} successfully."  fi    #备份部署包  omsWarFile="ecw-oms.war"  if [ -f $omsWarFile ];  then  #备份部署包成功${bakFileName}  bakFileName="${omsWarFile}\_`date +%Y%m%d%H%M%S`.bak"  mv "${webapps}/${omsWarFile}" "${bakDir}/${bakFileName}"  echo "Backup deployment package from "${webapps}/${omsWarFile}" to "${bakDir}/${bakFileName}" successfully"  else  #$omsWarFile 文件不存在,不做备份处理  echo ""${webapps}/${omsWarFile}" File does not exist, do not do backup processing"  fi  #删除旧部署包  omsDir="${webapps}/ecw-oms"  if [ -d $omsDir ];  then  echo "${omsDir} exists"    #删除原部署文件目录${omsDir} 成功。  rm -rf ${omsDir}  echo "Delete the old deployment file directory ${omsDir} successfully."  else  #${omsDir} 文件夹不存在  echo "${omsDir} folder does not exist"  fi  #回退到jetty安装目录${tomcatFile}。  #cd $  #echo "Back to the jetty installation directory ${tomcatFile}"  #开始迁移新部署包  echo "Start migrating a new deployment package"  tempWarFile="${tomcatFile}/temp/${omsWarFile}"  deployWarFile="${webapps}/${omsWarFile}"  if [ -f $tempWarFile ];  then  echo "${tempWarFile} exists"  #移动新部署包"ecw-oms.war"从tomcat/temp目录到tomcat/webapps目录下成功。  mv "${tempWarFile}" "${deployWarFile}"  echo "Move the new deployment package ${omsWarFile} from ${tempWarFile} to the ${deployWarFile}.successfully"    else  echo "${tempWarFile} folder does not exist"  fi  #启动tomcat  if [ -f ${deployWarFile} ];  then  echo "${deployWarFile} is exist !"    cd ${binDir}  echo "into ${binDir} , run startup.sh ,Start server!"    #以调试模式运行  #./startup.sh debug  #生产模式运行  ./startup.sh  echo "running startup.sh script successfully!"  else  echo "The ${webapps}/${omsWarFile} file does not exist and cannot be redeployed"  fi  #deploy.sh脚本执行完成!  echo " The deploy.sh script is finished!"  exit 0 |