Jenkins的搭建与使用

1. **在linux配置git公钥到github上**

安装git的目的是在自动化部署前实时从git远程仓库中拉取最新的代码。在linux(centos7系统)安装git：

yum install git

生成密钥：

ssh-keygen -t rsa -C "youremail@abc.com"

可以不用设置密钥密码，连续按三次回车，如图1所示。

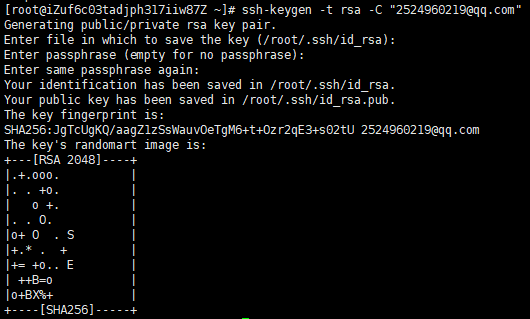


图1 密钥生成图

将/root/.ssh/id\_rsa.pub中生成的密钥添加到github上去。具体步骤如下：

1. 打开github，点击右上角个人头像，点击setting。
2. 找到Personal settings的SSH AND GPG KEYS一栏，点击New SSH key。如图2所示。

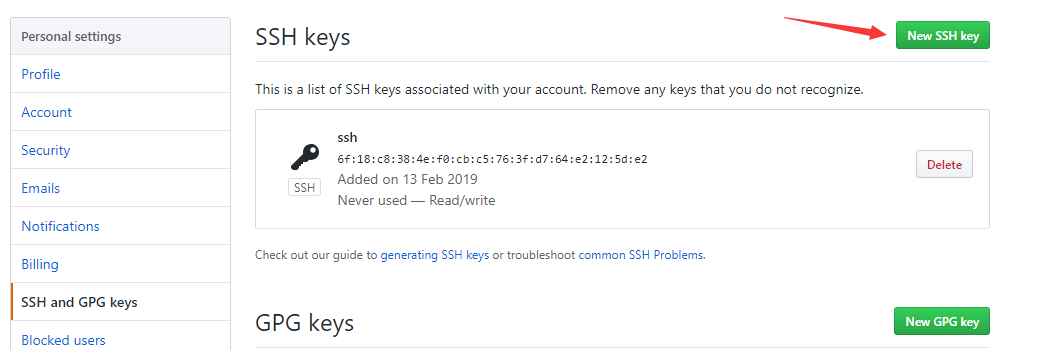


图2 新增ssh keys图

1. 将刚刚在服务器生成的ssh字符串复制黏贴进文本框，点击Add SSH key即可，如图3所示。

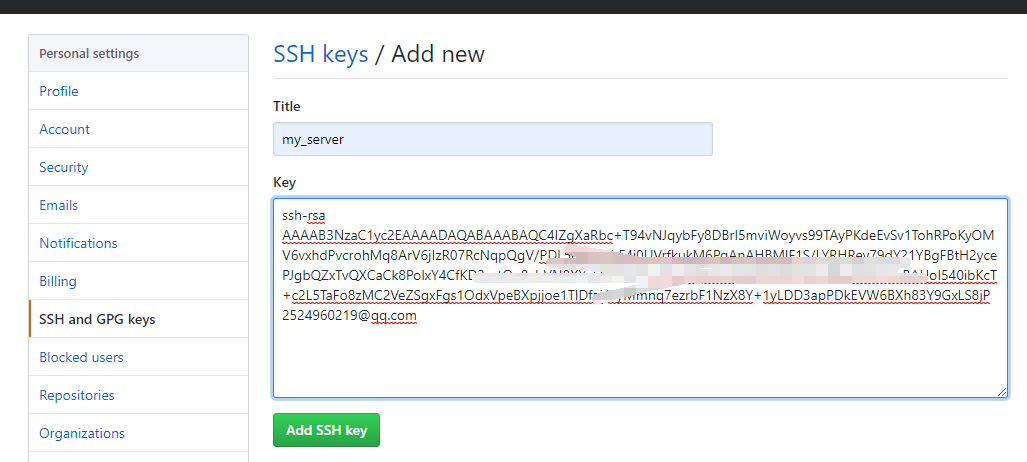


图3 新增ssh key图

1. **在linux安装maven**

安装maven的目的是通过项目中的pom.xml文件自动解决项目依赖问题，构建项目。将本地已下载好的maven压缩包上传至服务器解压即可。

具体步骤：

1. 将本地maven包拖进XShell框内，进行上传。如图4所示。



图4 上传maven压缩包图

1. 使用命令进行解压：

tar -zxvf apache-maven-3.3.9-bin.tar.gz

1. 配置环境变量。编辑etc/profile文件，添加环境，添加完毕后保存退出，source使其生效：

vi etc/profile

//在文件中加入下述字符串

export MAVEN\_HOME=/usr/local/maven/maven

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

//使其生效

source /etc/profile

1. 检查版本号，检查环境是否已正确配置：

mvn -v

结果如图5所示：

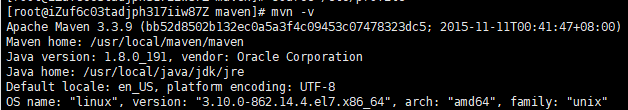


图5 maven版本图

1. **在linux安装Jenkins**

使用命令，关闭防火墙，防止防火墙影响jenkins的工作：

systemctl stop firewalld

systemctl disable firewalld

**安装方法一、**

关闭防火墙后，使用yum命令，安装jenkins：

//下载YUM源

wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat/jenkins.repo

//添加YUM源

rpm --import https://jenkins-ci.org/redhat/jenkins-ci.org.key

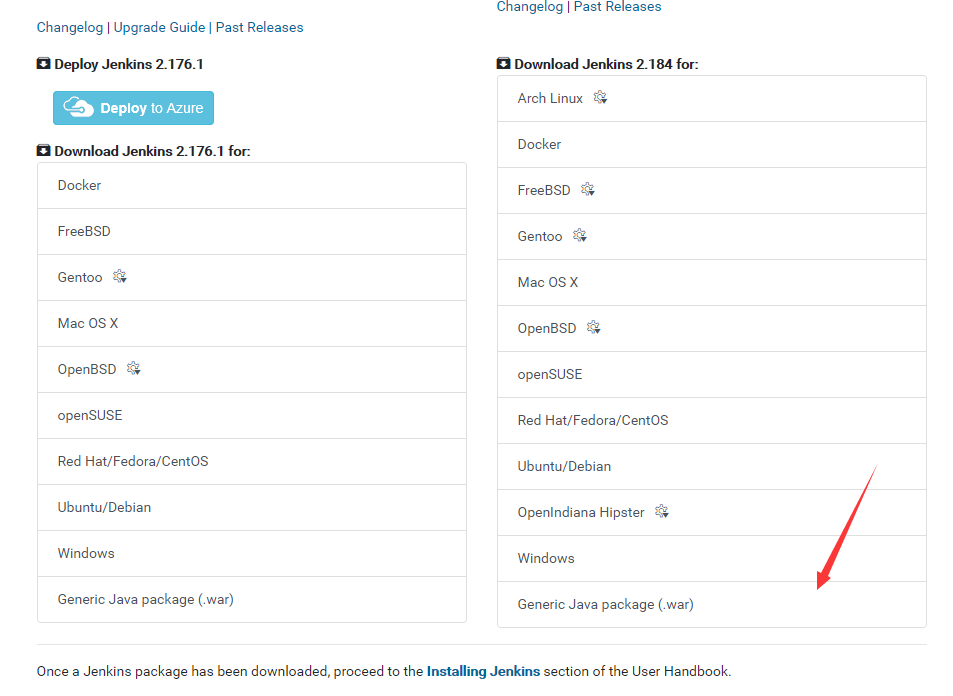
//YUM安装

yum install -y jenkins

**安装方法二、**

由于使用yum源安装较慢，故直接使用官网提供的war包，如图6所示，复

制至tomcat目录下的webapps中。

  
图6 下载war包图

登录网址：http://101.132.43.140:8081/jenkins/，如图7所示，将从本地文件生成的密码复制到文本框中。



图7 登录jenkins图

点击安装推荐的插件，等待所有的插件安装完毕。



图8 选择插件图

随后创建个人管理员账户。创建完毕后主页面如图9所示。



图9 欢迎界面图

1. **配置Jenkins**

1）点击manage jenkin中的Configure System，如图10所示。

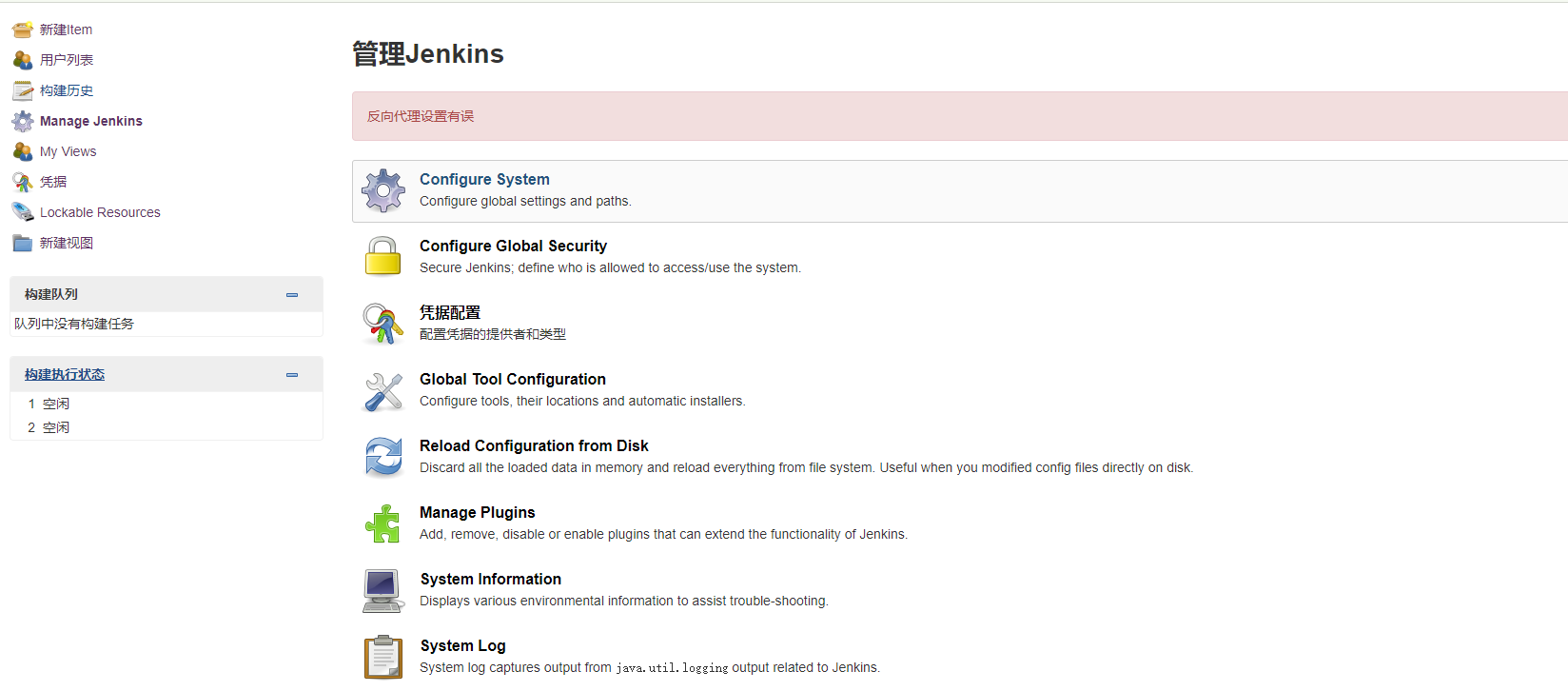


图10 配置界面图

填写git plugin信息，如图11所示。

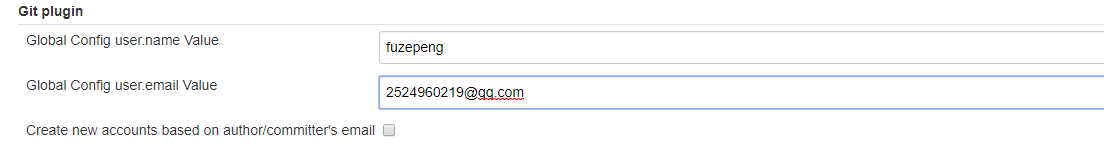


图11 git配置图

1. 添加凭据。首先点击主页面凭据，随后点击Jenkins，如图12所示，在全局凭据中加入个人信息。用户名与密码为管理员账号和密码。



图12 凭据配置图

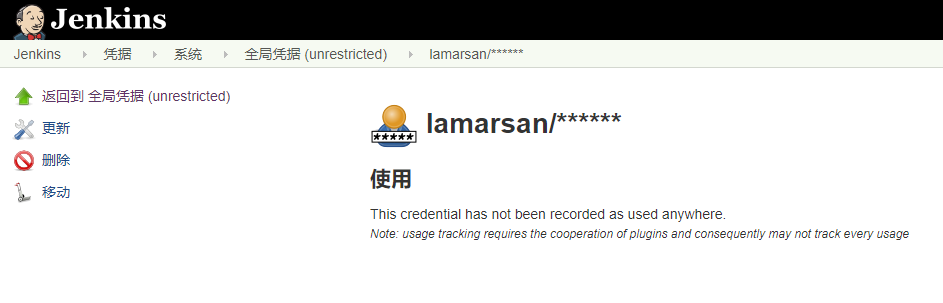


图13 凭据添加图

1. 点击全局设置，添加jdk，git与maven，如图14所示。

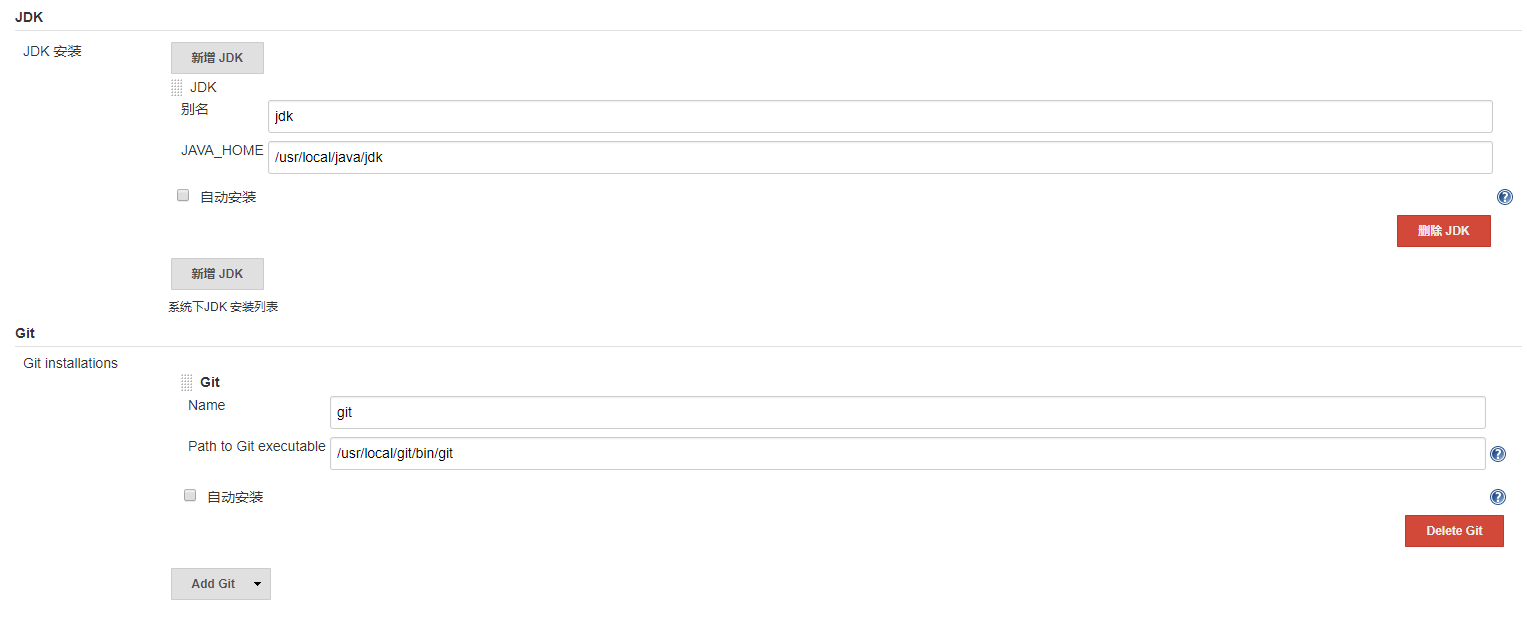


图14 软件添加图

1. **自动构建Maven的Freestyle Job项目**

1）首先点击新建Item，如图10所示。

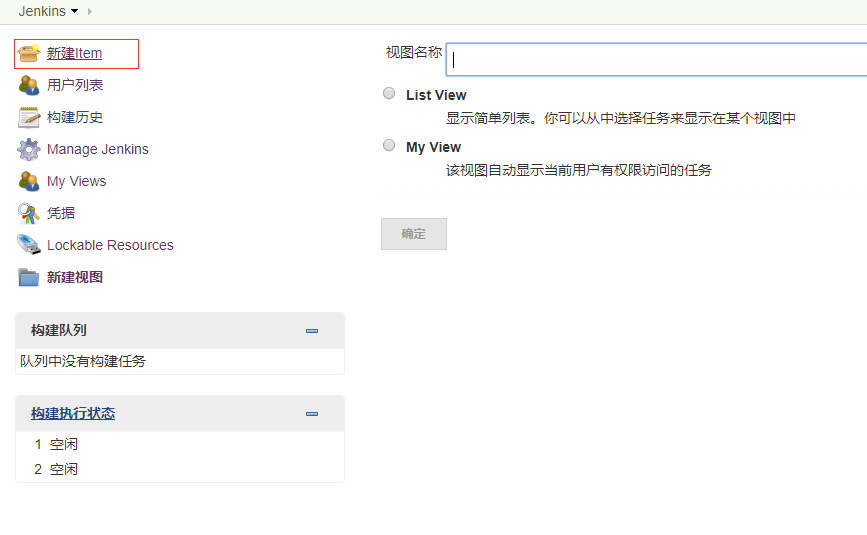
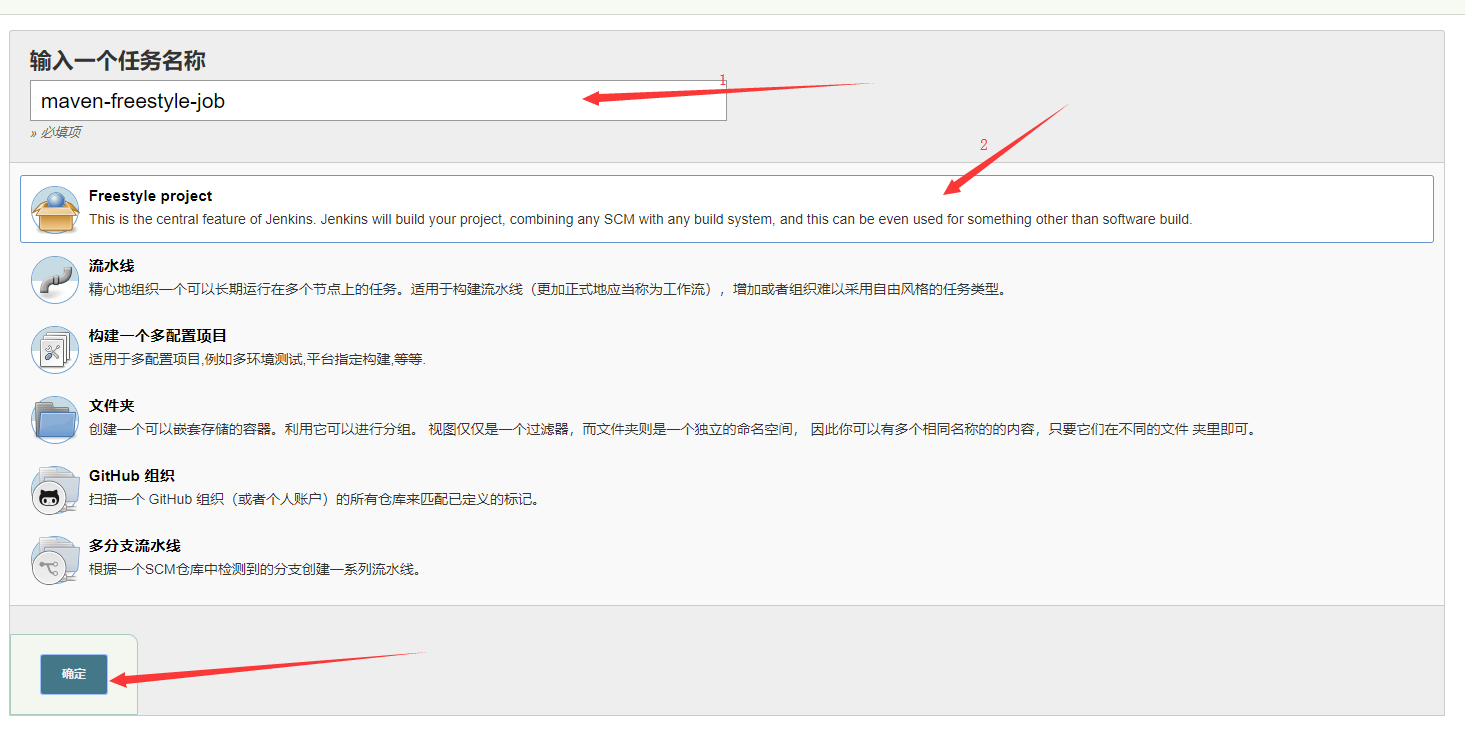


图10 新建项目图

1. 填写名称，并点击自由风格项目，最后点击确定，如图11所示。

图11 新建项目图

1. 可以填入构建参数描述，如图12所示。

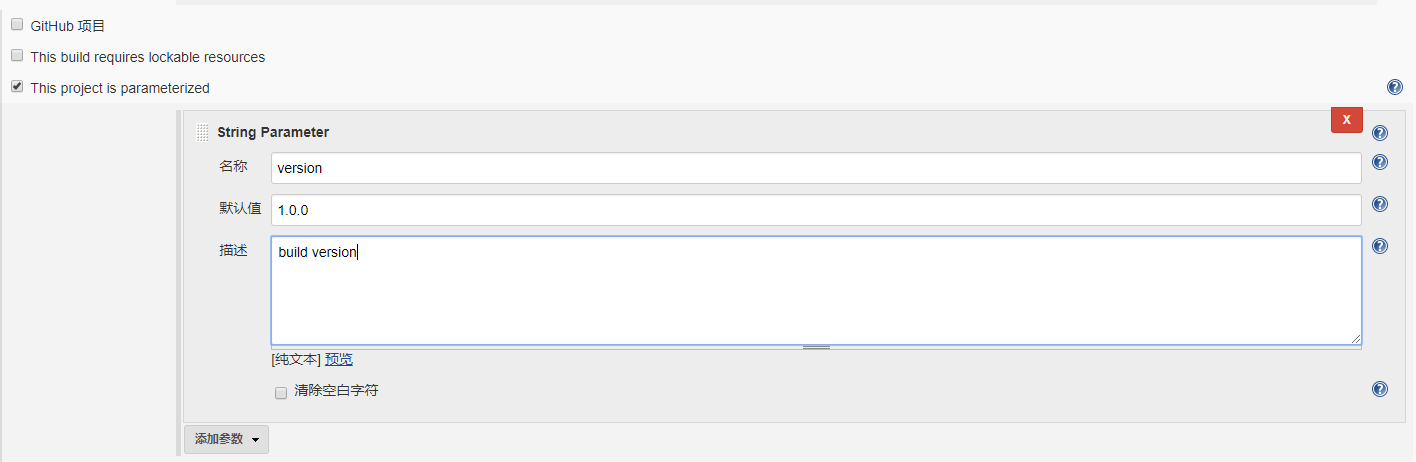


图12 构建参数图

1. 填入描述与git仓库地址和Credentials，如图13所示。

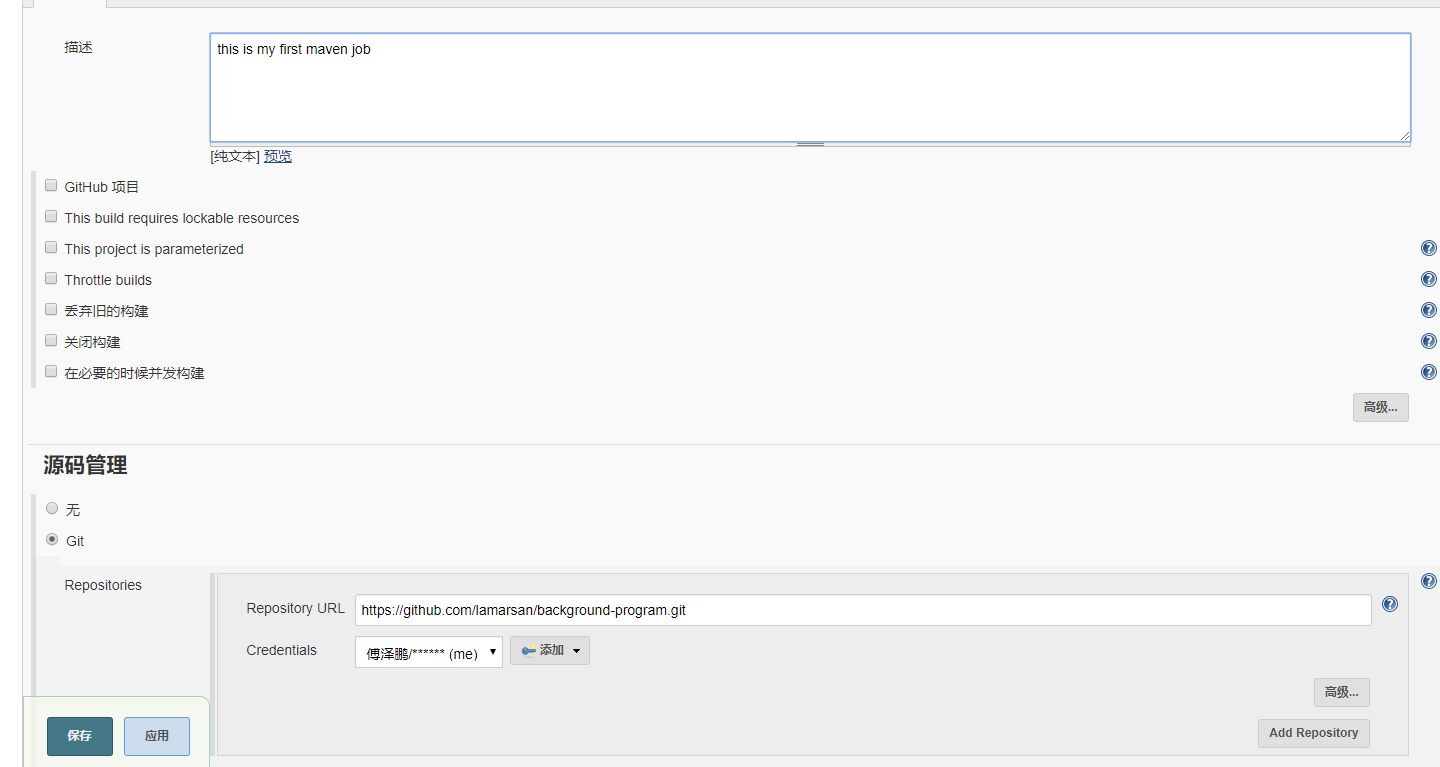


图13 新建项目git图

1. 点击构建，增加构建步骤，点击执行Shell，添加代码如下。

#!/bin/sh

export PATH="/bin:/sbin:/user/bin:/user/sbin:/user/local/bin:/usr/local/sbin"

# Print env variable

echo "The build is $version" >> test.properties

echo "[INFO] Done..."

# Check test properties

echo "[INFO] Check test properties"

if [ -s test.properties]

then

cat test.properties

echo "[INFO] Done..."

else

echo "test.properties is empty"

fi

echo "[INFO] Build finished..."



图14 选择maven目标图

1. 点击增加构建步骤，点击调用顶层Maven目标，添加信息如图15所示。



图15 增加Maven目标图

1. 保存退出后点击Build with Parameters，点击开始构建，即可开始项目的构建，如图16所示。

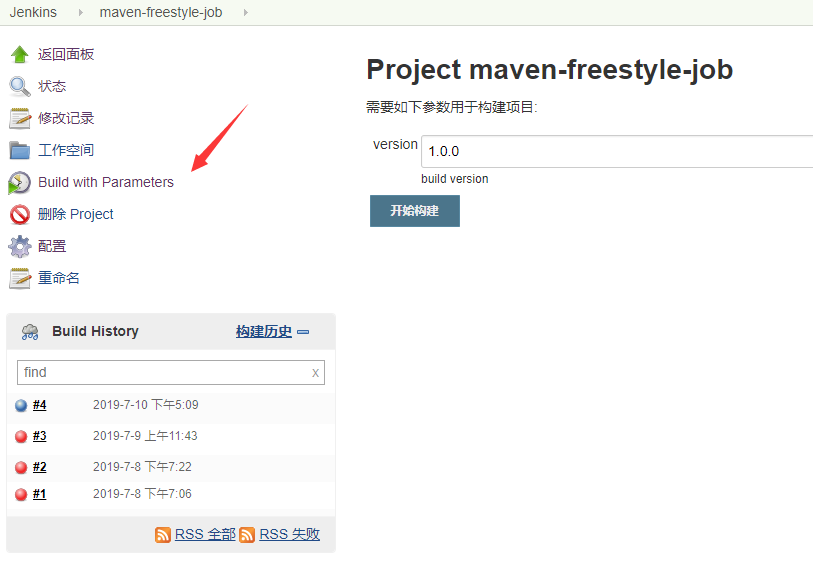


图16 构建maven图

1. 点击左下角的小圆圈可以看见控制台的日志输出，可以查看构建已经成功。生成的Jar包可以在/root/.jenkins/workspace/maven-freestyle-job/target中查看。如图17,18所示。

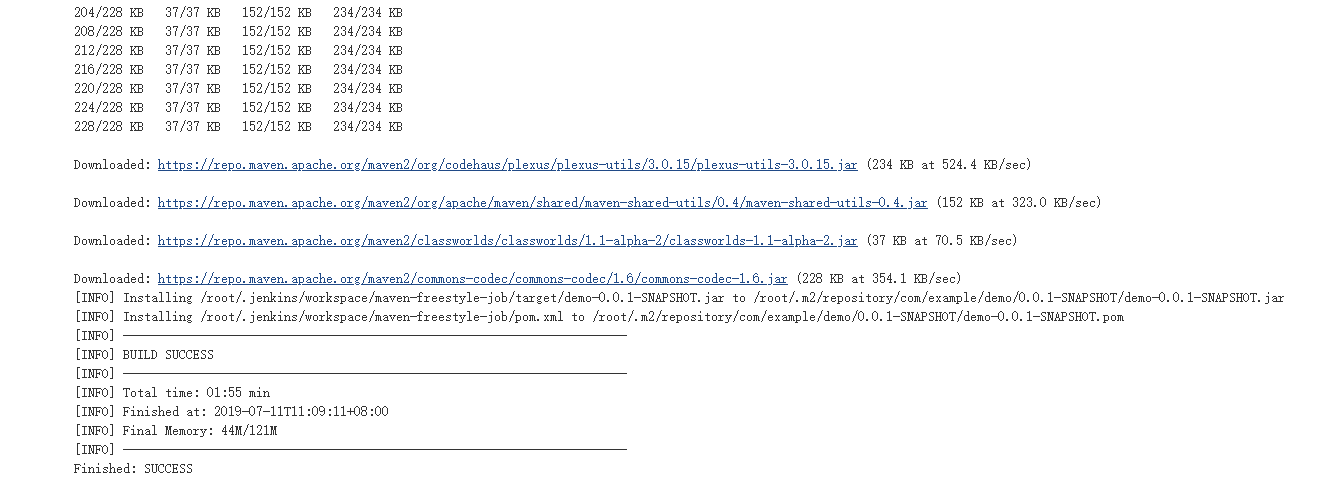


图17 控制台输出图



图18 文件目录图

1. 此外，jenkins还可以提供定时与轮询构建。

1：轮询：检测程序反复询问代码管理系统是否有新变更。

2：定期：检测程序配置为定期启动构建，无论代码是否有变更，但若没有变更，也不会增加额外成本。

设定定时任务如图19所示。



图19 定期任务图

设定轮询任务如图20所示。

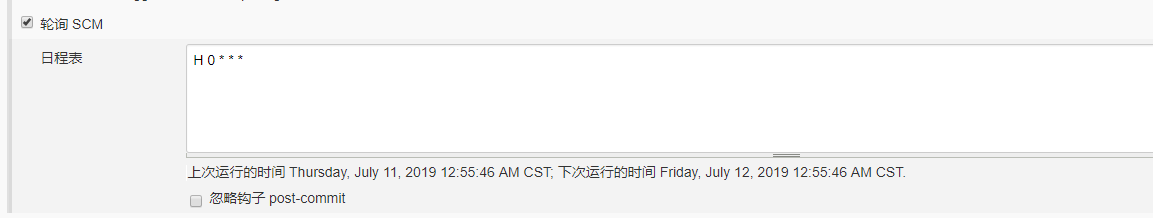


图20 轮询任务图

定时任务与轮询任务可以兼容，同时该字段遵循cron的语法（略有不同）。具体来说，每行包含由TAB或空格分隔的5个字段：

每小时营业时间。

|  |  |
| --- | --- |
| 分钟 | 一小时内（0-59） |
| 小时 | 一天中的小时（0-23） |
| DOM | 每月的某一天（1-31） |
| 月 | 月（1-12） |
| DOW | 星期几（0-7），其中0和7是星期日。 |

要为一个字段指定多个值，可以使用以下运算符。按优先顺序排列。

* \* 指定所有有效值
* M-N 指定一系列值
* M-N/X或者\*/X按X的间隔步长到指定范围或整个有效范围
* A,B,...,Z 枚举多个值

为了允许定期计划的任务在系统上产生均匀负载，H应尽可能使用符号（“哈希”）。例如，使用0 0 \* \* \*十几个日常工作将导致午夜大幅飙升。相比之下，使用H H \* \* \*仍然会每天执行一次，但不能同时执行，更好地使用有限的资源。

所述H符号可以与范围内使用。例如，H H(0-7) \* \* \* 表示从凌晨12:00（午夜）到早上7:59之间的某个时间段。还可以使用H带有或不带范围的步长间隔。

该H符号可以被认为是在一定范围内的随机值，但它实际上是作业名称，而不是随机函数的哈希值，所以该值仍然是任何项目稳定。

请注意，对于月份字段的日期，由于月份长度可变，短期周期（例如\*/3或H/3将不会在大多数月份结束时始终如一地工作）。例如，\*/3将在一个月的第1天，第4天，第31天，然后在下个月的第二天再次运行。哈希总是在1-28范围内选择，因此H/3在月底之间产生3到6天之间的差距。（更长的周期也会有不一致的长度，但效果可能相对不太明显。）

以句号开头的空行和#行将被忽略。

此外，@yearly，@annually，@monthly， @weekly，@daily，@midnight，并且@hourly也支持方便的别名。这些使用哈希系统进行自动平衡。例如，在一小时内的任何时间@hourly都是相同的H \* \* \* \*并且可能意味着。 @midnight实际上是指在凌晨12:00到凌晨2:59之间的某个时间段。

例子：

＃每十五分钟一次（也许在：07：，22，：37，：52）

H / 15 \* \* \* \*

＃每小时上半场每十分钟一次（三次，也许是：04，：14，：24）

H（0-29）/ 10 \* \* \* \*

＃每小时一次，每小时45分钟，从上午9:45开始，每个工作日下午3:45结束。

45 9-16 / 2 \* \* 1-5

＃每个工作日上午9点到下午5点每两小时一次（可能是上午10点38分，下午12点38分，下午2点38分，下午4点38分）

HH（9-16）/ 2 \* \* 1-5

＃每月1日和15日每天一次，12月除外

HH 1,15 1-11 \*

### 时区规范

定期任务通常在Jenkins主JVM（目前为**亚洲/上海**）的时区中的预定时间执行。通过在字段的第一行中指定备用时区，可以选择更改此行为。时区规范以TZ=，后跟时区的ID 开头。

具有时区规范的计划的完整示例：

TZ =欧洲/伦敦

＃这项工作需要在伦敦时间早上开始

H 8 \* \* \*

#Butlers没有五点钟，所以我们再次开始工作

H（0-30）17 \* \* \*

1. **自动构建Maven的Pipeline Job项目**
2. 首先点击新建Item，如图21所示。

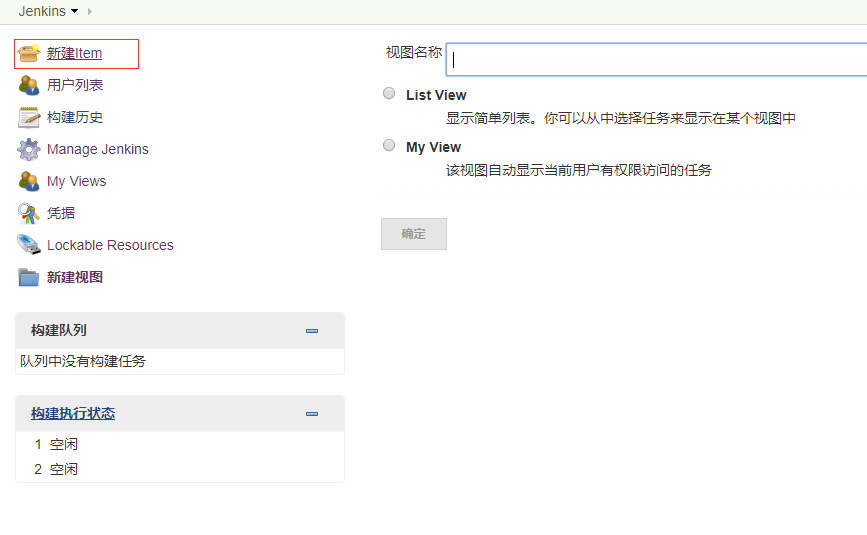


图21 新建项目图

1. 填写名称，并点击流水线项目，最后点击确定，如图22所示。

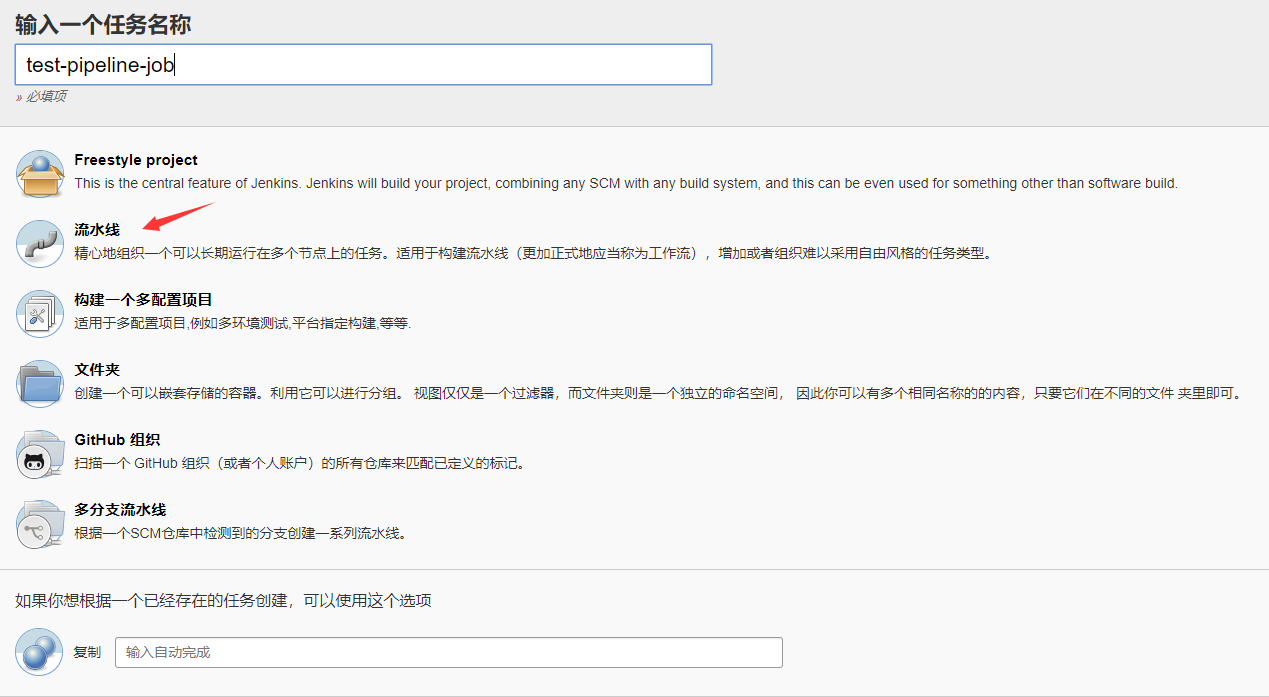


图22 选择类型图

1. 将事先准备好的shell代码填入流水线，如图23所示。

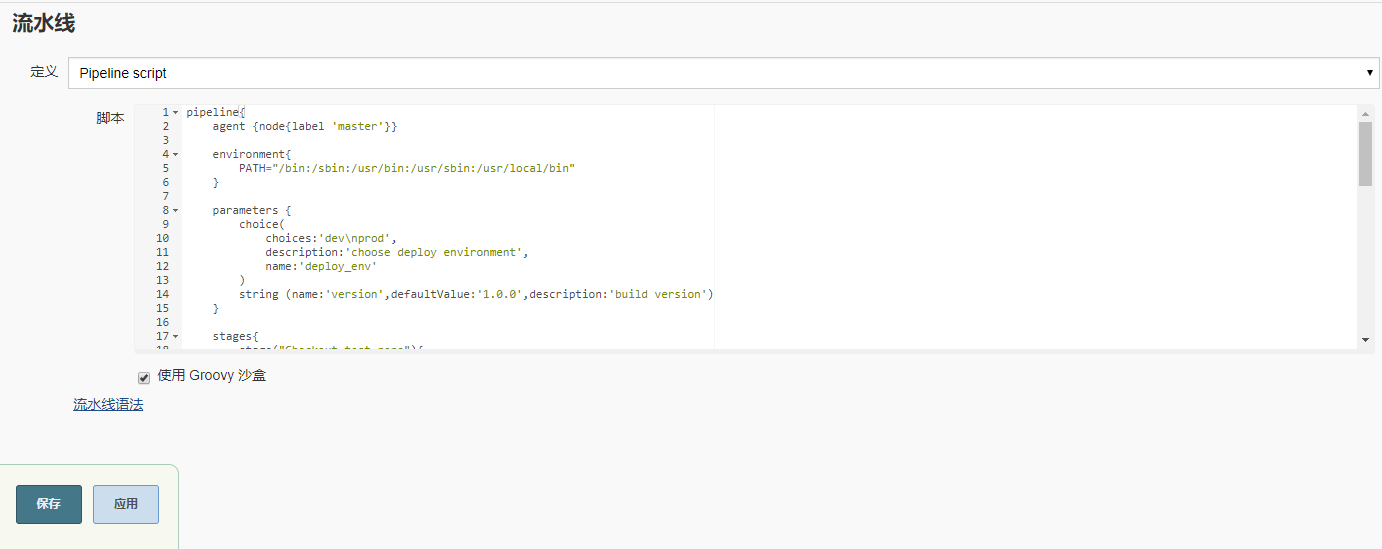


图23 填写脚本图

pipeline{

agent {node{label 'master'}}

environment{

PATH="/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin"

}

parameters {

choice(

choices:'dev\nprod',

description:'choose deploy environment',

name:'deploy\_env'

)

string (name:'version',defaultValue:'1.0.0',description:'build version')

}

stages{

stage("Checkout test repo"){

steps{

sh 'git config --global http.sslVerify false'

dir ("${env.WORKSPACE}"){

git branch:'master',credentialsId:"c1b21f21-edf9-44fb-9da6-c32169ffe2d9",url:'https://github.com/lamarsan/background-program.git'

}

}

}

stage("Print env variable"){

steps{

dir("${env.WORKSPACE}"){

sh """

echo "[INFO] Print env variable"

echo "Current deployment environment is $deploy\_env" >> test.properties

echo "The build is $version" >> test.properties

echo "[INFO] Done..."

"""

}

}

}

stage("Check test properties"){

steps{

dir("${env.WORKSPACE}"){

sh """

echo "[INFO] Check test properties"

if [ -s test.properties ]

then

cat test.properties

echo "[INFO] Done..."

else

echo "test.properties is empty"

fi

"""

echo "[INFO] Build finished..."

}

}

}

}

}

1. 保存退出，点击立即构建，结果会失败，如图24所示。



图24 第一次构建图

1. 失败是由于没有添加参数所致，返回到项目，点击Build with Parameters，如图25所示。同时，日志提示没有找到git命令，故去掉sh 'git config --global http.sslVerify false'一行。

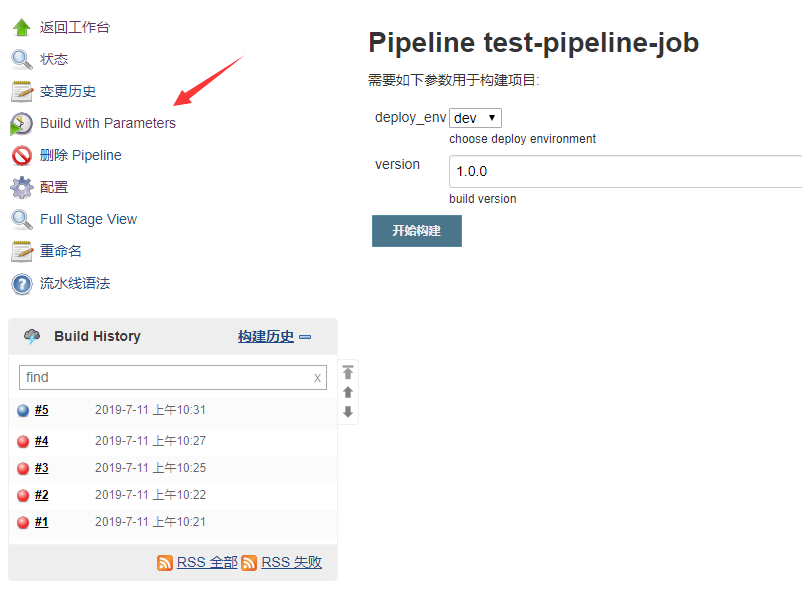


图25 添加参数图

1. 点击开始构建，构建成功，如图26所示。

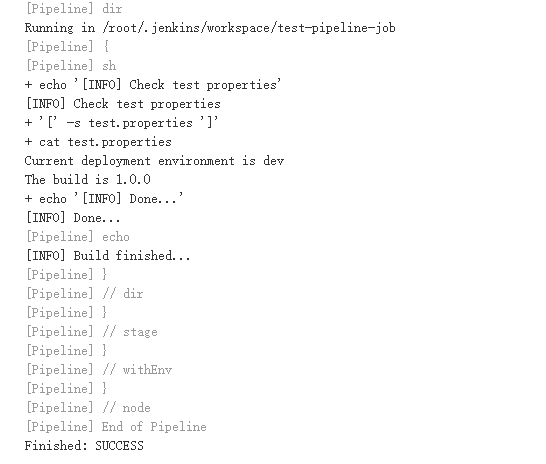


图26 构建成功图