

# 📖 01、SpringBoot项目 - Jenkins基于Jar持续集成搭建文档

---

## 🔧 基于手动方式发布项目

- 开发代码
- 打包-jar
- 把jar上传到服务器
- 把上一次的服务杀掉，停止
- 启动运行新的jar

## 🔧 基于dockerfile

- 开发代码
- 编写Dockerfile (一次性)
- mvn clean package 自动生成镜像
- 手动或者使用idea Docker插件，创建容器，启动容器

学习成本：Docker

## 基于jenkins + dockerfile + jenkinsfile + pipeline

- 开发代码
- 搭建jenkins环境 (一次性)
- 编写Dockerfile (一次性) 和Jenkinsfile规则 (一次性)
- 上传代码
- jenkins控制台启动服务 (触发器)

学习成本: Docker , Jenkins , Pipeline 触发器

## 基于jenkins + jar方式的发布

- 开发代码
- 搭建jenkins环境 (一次性)
- 创建一个任务 , 创建shell脚本
- 上传代码
- jenkins控制台启动服务 (触发器)

## 01、环境说明

服务	所需软件	部署地址
持续集成服务	Jenkins 2.319.1、Maven3.6.3, Git	47.107.225.126
应用测试服务	JDK1.8	47.107.225.126

## 01、准备项目

准备一个springboot + Dockerfile项目

## 02、准备服务器

服务器配置建议：4core + 8G 或者 4core + 16G 越高越好。

提醒：学生机就不要来试了，估计`docker`都跑不起来就卡死了

## 03、安装git

- 1 查看是否已经安装
- 2 `git --version`
- 3 使用`yml`安装
- 4 `yum -y install git`

```
[root@iZwz9gi039o35ikykyj1tZ ~]# git --version
git version 1.8.3.1
```

## 04、安装jdk1.8

查看当前是否有安装jdk

- maven和项目，jenkins都java项目肯定需要java环境

- 1 `java -version`

安装jdk1.8

- 1 `yum install java-1.8.0-openjdk* -y`

- 安装完以后,默认的路径是: `/usr/lib/jvm`
- 提醒：建议不要安装jdk11。

## 05、安装maven依赖

- 1、前往官网下载自己想要的maven包

地址: <http://maven.apache.org/download.cgi>

各版本地址: <https://archive.apache.org/dist/maven/maven-3/>

我以apache-maven-3.6.3-bin.tar.gz为例,

下载地址: <https://archive.apache.org/dist/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz>

- 2、将maven上传liunx服务器上,我这里上传到了 `/www/servers`,当前的安装目录root如下: `cd /www/servers`

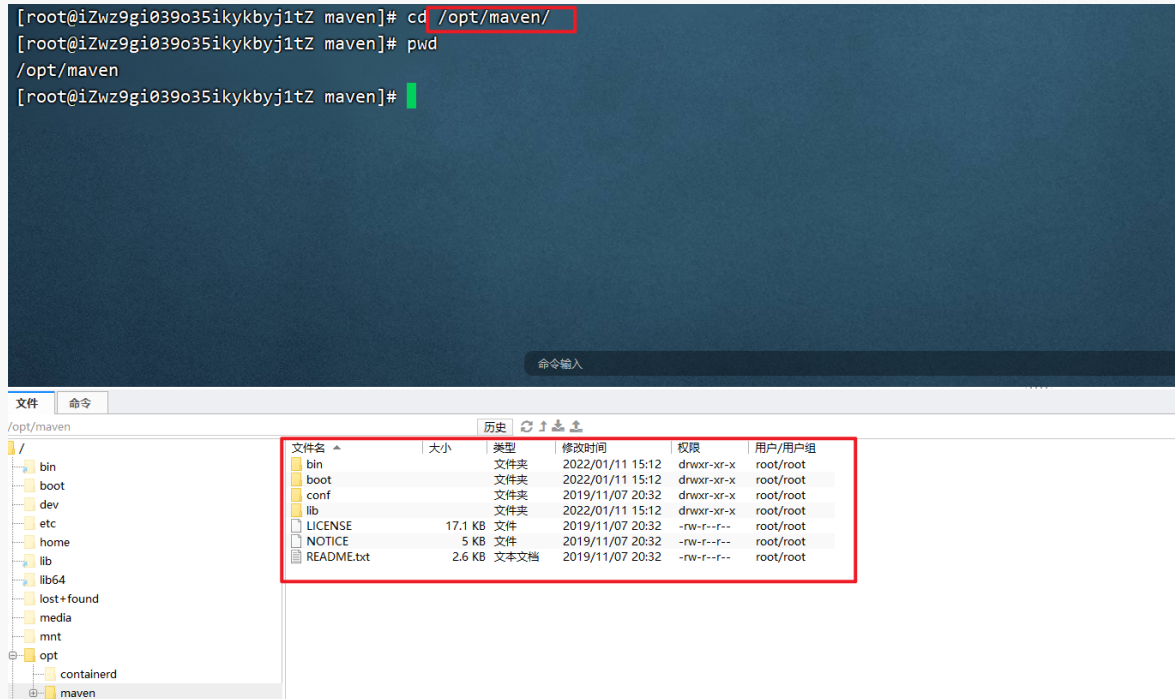
```
1 mkdir -p /www/servers/maven
2 cd /www/servers/maven
```

- 3、上传完毕,执行解压

```
1 wget https://archive.apache.org/dist/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz
```

如果下载很慢建议用手动的方式上传到 `/www/servers/maven`

```
1 #解压文件
2 tar -xzf apache-maven-3.6.3-bin.tar.gz
3 #创建目录,用于存放maven
4 mkdir -p /opt/maven
5 #将解压完的maven文件剪切到刚刚创建的目录下
6 mv apache-maven-3.6.3/* /opt/maven
```



建议配置：阿里云的maven镜像。<https://developer.aliyun.com/mvn/guide>  
步骤：在`conf/setttings.xml`中配置如下：

```
1 <mirror>
2   <id>aliyunmaven</id>
3   <mirrorOf>*</mirrorOf>
4   <name>aliyunrepositoty</name>
5   <url>https://maven.aliyun.com/repository/public</url>
6 </mirror>
```

```
<mirrors>
  <!-- mirror
    | Specifies a repository mirror site to use instead of a given repository. The repository
    | this mirror serves has an ID that matches the mirrorOf element of this mirror. IDs are u
    | for inheritance and direct lookup purposes, and must be unique across the set of mirrors
    |
  <mirror>
    <id>mirrorId</id>
    <mirrorOf>repositoryId</mirrorOf>
    <name>Human Readable Name for this Mirror.</name>
    <url>http://my.repository.com/repo/path</url>
  </mirror>
  -->

  <mirror>
    <id>aliyunmaven</id>
    <mirrorOf>*</mirrorOf>
    <name>aliyunrepositoty</name>
    <url>https://maven.aliyun.com/repository/public</url>
  </mirror>
</mirrors>
```

- 4、配置maven环境变量

安装完maven，还需要配置java和maven的环境变量。就放在文件的尾部即可  
首先打开环境配置文件

```
1 vim /etc/profile
```

配置内容

```
1 export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk
2 export MAVEN_HOME=/opt/maven
3 export PATH=$PATH:$JAVA_HOME/bin:$MAVEN_HOME/bin
```

执行保存生效

```
1 source /etc/profile
```

查看maven版本

```
1 mvn -v
```

```
1 [root@izwz9gi039o35ikykbj1tz maven]# mvn -v
2 Apache Maven 3.6.3
   (cecedd343002696d0abb50b32b541b8a6ba2883f)
3 Maven home: /opt/maven
4 Java version: 1.8.0_312, vendor: Red Hat, Inc., runtime:
   /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.312.b07-
   1.e17_9.x86_64/jre
5 Default locale: zh_CN, platform encoding: UTF-8
6 OS name: "linux", version: "3.10.0-1160.49.1.e17.x86_64",
   arch: "amd64", family: "unix"
```

## 06、安装jenkins

### 1、官网地址

<https://www.jenkins.io/download/>

## Jenkins download and deployment

The Jenkins project produces two release lines: Stable (LTS) and regular (Weekly). Depending on your organization's needs, one may be preferred over the other. See the links below for more information and recommendations about the release lines.

### Stable (LTS)

Long-Term Support (LTS) release baselines are chosen every 12 weeks from the stream of regular releases. Every 4 weeks we release stable releases which include bug and security fix backports. [Learn more...](#)

[Changelog](#) | [Upgrade Guide](#) | [Past Releases](#)

### Regular releases (Weekly)

This release line delivers bug fixes and new features rapidly to users and plugin developers who need them. It is generally delivered on a weekly cadence. [Learn more...](#)





[Changelog](#) | [Past Releases](#)

## Downloading Jenkins







Jenkins is distributed as WAR files, native packages, installers, and Docker images. Follow these installation steps:

1. Before downloading, please take a moment to review the [Hardware and Software requirements](#) section of the User Handbook.
2. Select one of the packages below and follow the download instructions.
3. Once a Jenkins package has been downloaded, proceed to the [Installing Jenkins](#) section of the User Handbook.
4. You may also want to verify the package you downloaded. [Learn more about verifying Jenkins downloads.](#)

### Download Jenkins 2.319.1 LTS for:

Generic Java package (.war) SHA-256: 7e4b848a752eda740c2c7a60956bf05d9df426f2c805bbaeac897179b630a562
Docker
Ubuntu/Debian
CentOS/Fedora/Red Hat
Windows
openSUSE
FreeBSD 
Gentoo 
macOS 
OpenBSD 

### Download Jenkins 2.328 for:

Generic Java package (.war) SHA-256: 9b1d27912a5539da10324c8a86c5db16c43aae1d9eda05a8c9e46eb6181aca02
Docker
Ubuntu/Debian
CentOS/Fedora/Red Hat
Windows
openSUSE
Arch Linux 
FreeBSD 
Gentoo 
macOS 
OpenBSD 
OpenIndiana Hipster 

### 2、centos安装jenkins如下

- ```
1 sudo wget -O /etc/yum.repos.d/jenkins.repo  
https://pkg.jenkins.io/redhat-stable/jenkins.repo  
2 sudo rpm --import https://pkg.jenkins.io/redhat-  
stable/jenkins.io.key  
3 yum install epel-release # repository that provides  
'daemonize'  
4 yum install jenkins
```

如果之前有安装过jenkins的可以用下面方式进行卸载在操作上面的命令：

```
1 cd /root
2 #卸载之前残留的jenkins
3 rpm -e jenkins
4 find / -iname jenkins | xargs -n 1000 rm -rf
5 #查看是否卸载完毕
6 rpm -ql jenkins
```

### 3、jenkins配置

jenkins默认安装的配置目录在： `/etc/sysconfig/jenkins`

```
1 #安装完毕,进入到jenkins配置文件内,配置端口及用户名
2 vim /etc/sysconfig/jenkins
3 #找到这两行,修改成指定的端口
4 JENKINS_USER="用户名"
5 #示例: root
6 JENKINS_PORT="端口号"
7 #示例: 9999
8 #修改完毕,执行保存
9 wq!
10 #保存完毕,启动jenkins服务
11 systemctl start jenkins
12 #查看启动状态
13 systemctl status jenkins
```

完毕!

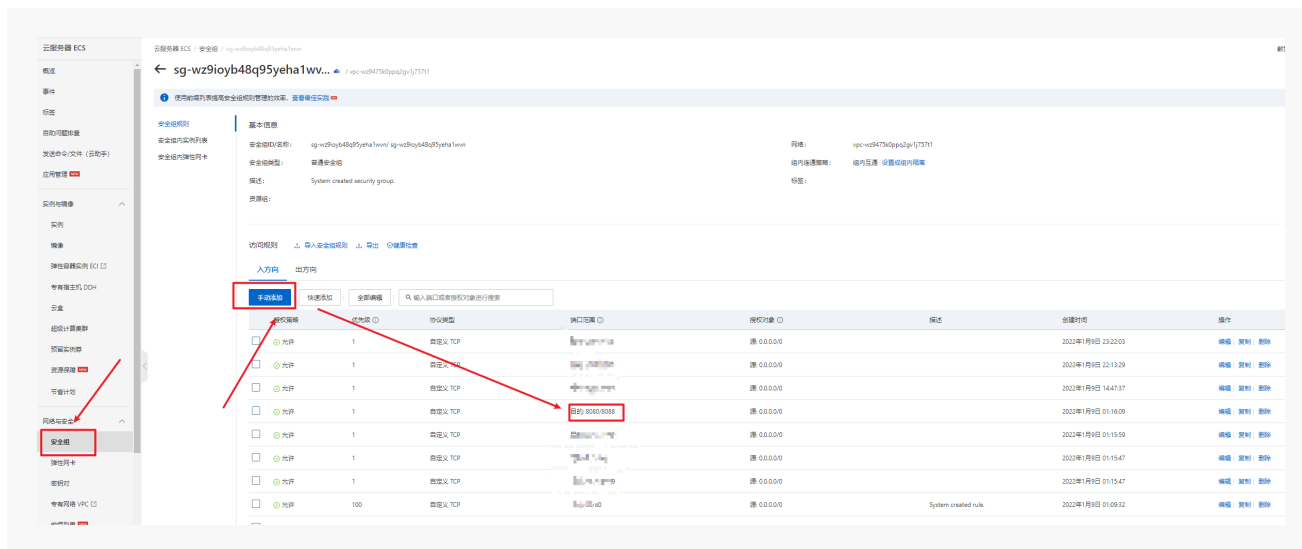
```
[root@iZwz9gi039o35ikybyj1tZ maven]# vim /etc/sysconfig/jenkins
[root@iZwz9gi039o35ikybyj1tZ maven]# systemctl start jenkins
[root@iZwz9gi039o35ikybyj1tZ maven]# systemctl status jenkins
● jenkins.service - LSB: Jenkins Automation Server
   Loaded: loaded (/etc/rc.d/init.d/jenkins; bad; vendor preset: disabled)
   Active: active (running) since 2022-01-11 15:26:23 CST; 1min 54s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 17727 ExecStart=/etc/rc.d/init.d/jenkins start (code=exited, status=0/SUCCESS)
    Tasks: 39
   Memory: 1.6G
    CGroup: /system.slice/jenkins.service
            └─17731 /etc/alternatives/java -Djava.awt.headless=true -DJENKINS_HOME=/var/lib/jenkins -jar /
```



## 🐼 07、启动jenkins

启动完,等待一分钟左右,访问刚刚配置的地址: <http://服务器ip:刚刚配置的端口号/>  
示例: <http://47.107.225.126:8088/>

提醒: 8088端口必须在你对应的云服务器的安全组中开放8088的端口和 上面 *docker*链接的2280的端口都要将其开放。



## 🐼 08、解锁 Jenkins

# 解锁 Jenkins

为了确保管理员安全地安装 Jenkins，密码已写入到日志中（[不知道在哪里？](#)）该文件在服务器上：

```
/var/lib/jenkins/secrets/initialAdminPassword
```

请从本地复制密码并粘贴到下面。

管理员密码



继续

根据提示：在服务器输入

```
1 cat /var/lib/jenkins/secrets/initialAdminPassword
```

```
[root@iZwz9gi039o35ikybj1tZ maven]# cat /var/lib/jenkins/secrets/initialAdminPassword
e5acc4af5d4049259034630765f5f419
```

把密码复制出来粘贴上去即可，然后点击继续

1 e5acc4af5d4049259034630765f5f419

入门

## 解锁 Jenkins

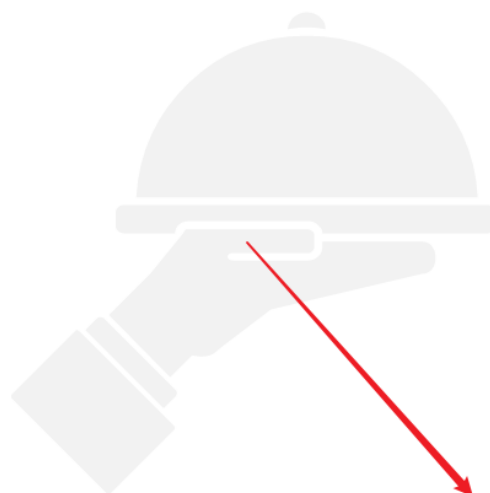
为了确保管理员安全地安装 Jenkins，密码已写入到日志中（[不知道在哪里？](#)）该文件在服务器上：

```
/var/lib/jenkins/secrets/initialAdminPassword
```

请从本地复制密码并粘贴到下面。

管理员密码

e5acc4af5d4049259034630765f5f419



继续

这里需要等待一会，才可以生效

## 09、自定义Jenkins

选择插件来安装即可



# 新手入门

|                                   |                                              |                                                         |                                                          |
|-----------------------------------|----------------------------------------------|---------------------------------------------------------|----------------------------------------------------------|
| <input type="radio"/> Folders     | <input type="radio"/> OWASP Markup Formatter | <input type="radio"/> Build Timeout                     | <input type="radio"/> Credentials Binding                |
| <input type="radio"/> Timestamper | <input type="radio"/> Workspace Cleanup      | <input type="radio"/> Ant                               | <input type="radio"/> Gradle                             |
| <input type="radio"/> Pipeline    | <input type="radio"/> GitHub Branch Source   | <input type="radio"/> Pipeline: GitHub Groovy Libraries | <input type="radio"/> Pipeline: Stage View               |
| <input type="radio"/> Git         | <input type="radio"/> SSH Build Agents       | <input type="radio"/> Matrix Authorization Strategy     | <input type="radio"/> PAM Authentication                 |
| <input type="radio"/> LDAP        | <input type="radio"/> Email Extension        | <input type="radio"/> Mailer                            | <input type="radio"/> Localization: Chinese (Simplified) |

\*\* - 需要依赖

Jenkins 2.319.1

等待所有安装完毕

## 10、创建管理员账号

建议全部写成一样的。**root**

## 创建第一个管理员用户

用户名:

密码:

确认密码:

全名:

电子邮件地址:

Jenkins 2.319.1

[使用admin账户继续](#)

[保存并完成](#)

## 11、实例配置

这步是告诉你的jenkins访问地址，记住一下。

## 实例配置

Jenkins URL:

`http://47.107.225.126:8088/`

Jenkins URL 用于给各种Jenkins资源提供绝对路径链接的根地址。这意味着对于很多Jenkins特色是需要正确设置的，例如：邮件通知、PR状态更新以及提供给构建步骤的BUILD\_URL环境变量。

推荐的默认值显示在尚未保存，如果可能的话这是根据当前请求生成的。最佳实践是要设置这个值，用户可能会需要用到。这将会避免在分享或者查看链接时的困惑。

# Jenkins已就绪!

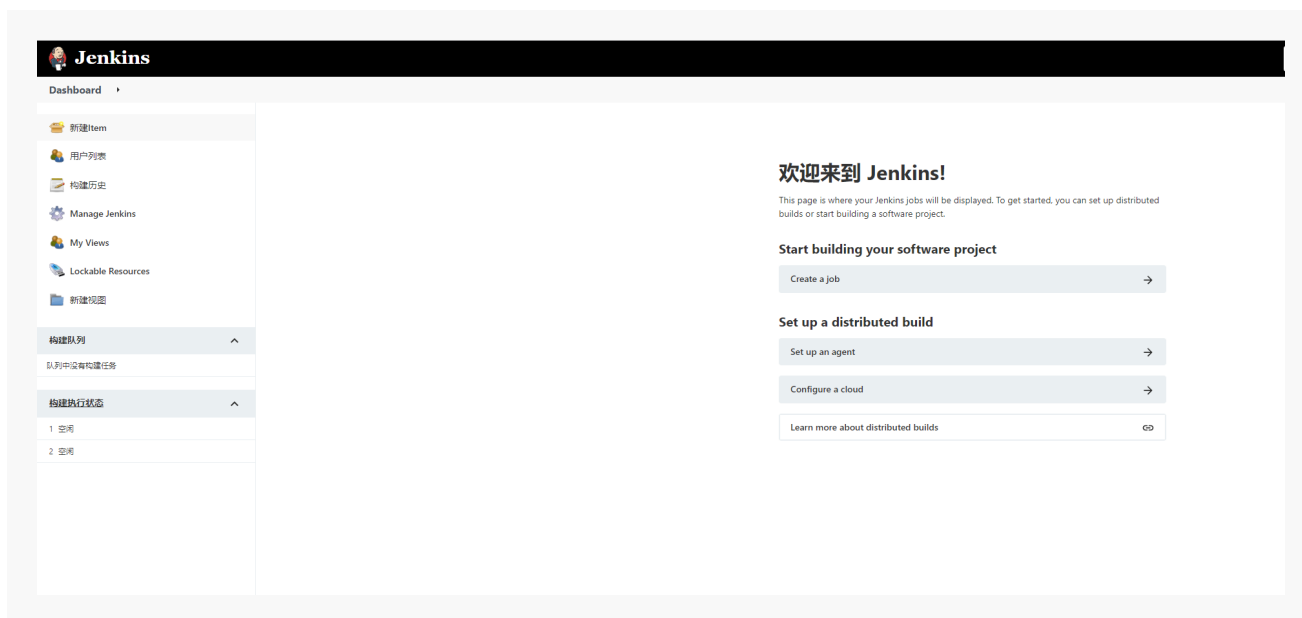
Jenkins安装已完成。

开始使用Jenkins

Jenkins 2.319.1

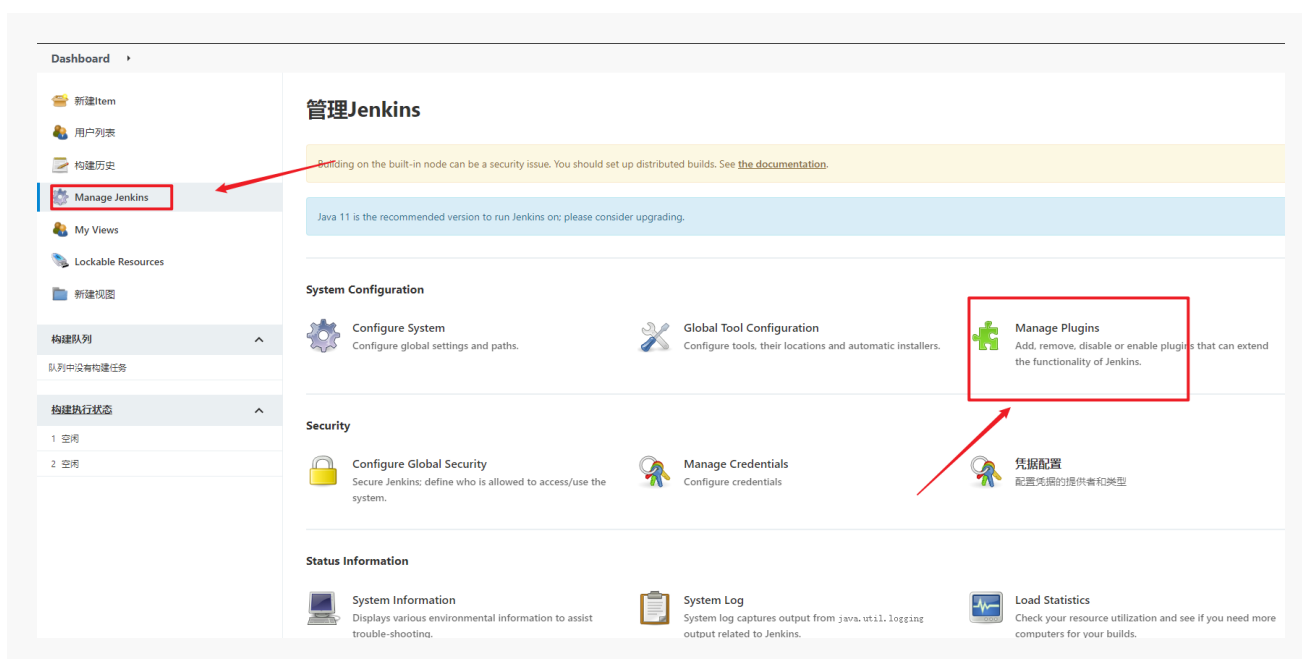
## 12、Jenkins控制面板





## 13、配置镜像源

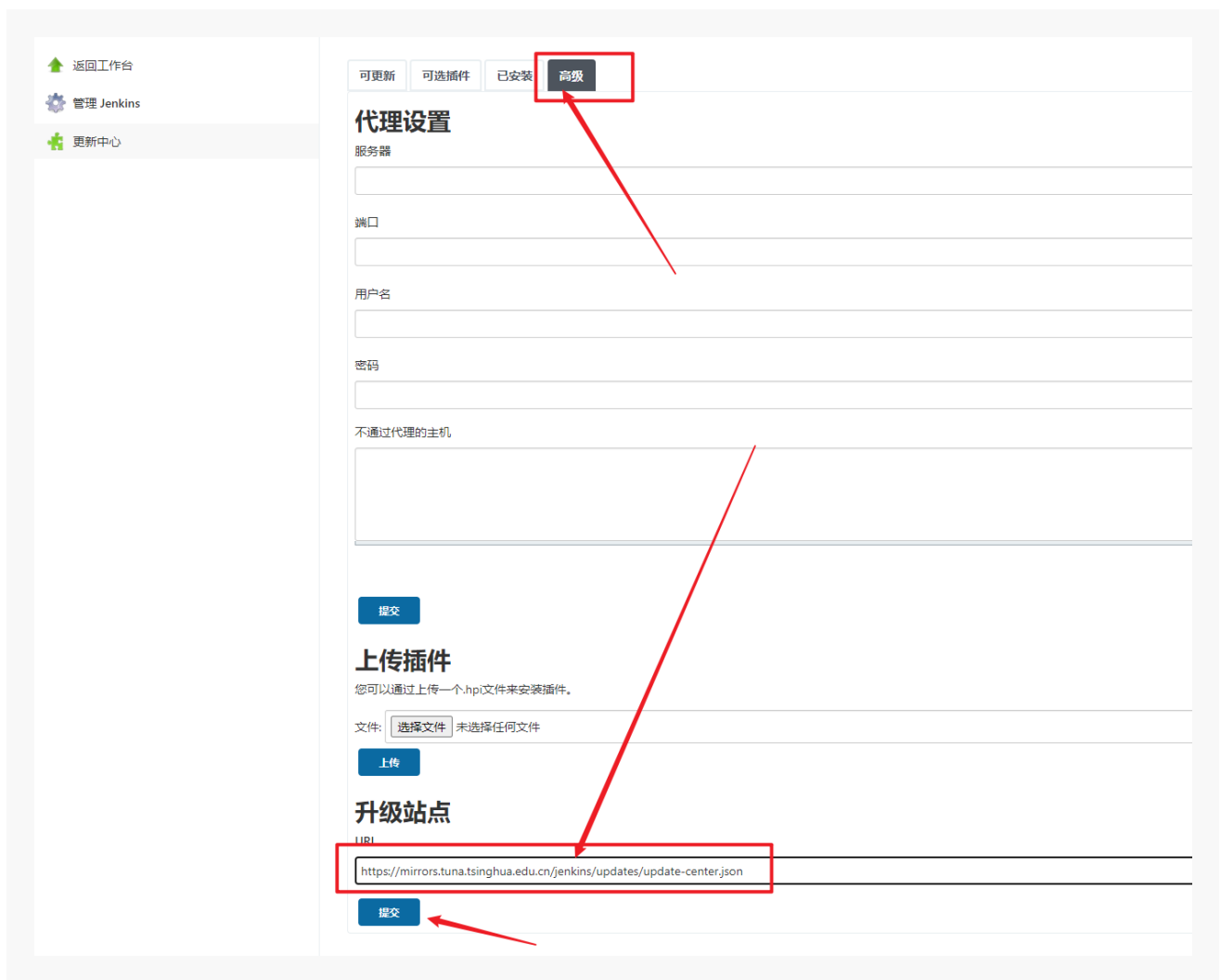
在管理页面依次进入: Jenkins->Manage Jenkins->Manage Plugins



点击【高级】

链接: <https://mirrors.tuna.tsinghua.edu.cn/jenkins/updates/update-center.json>

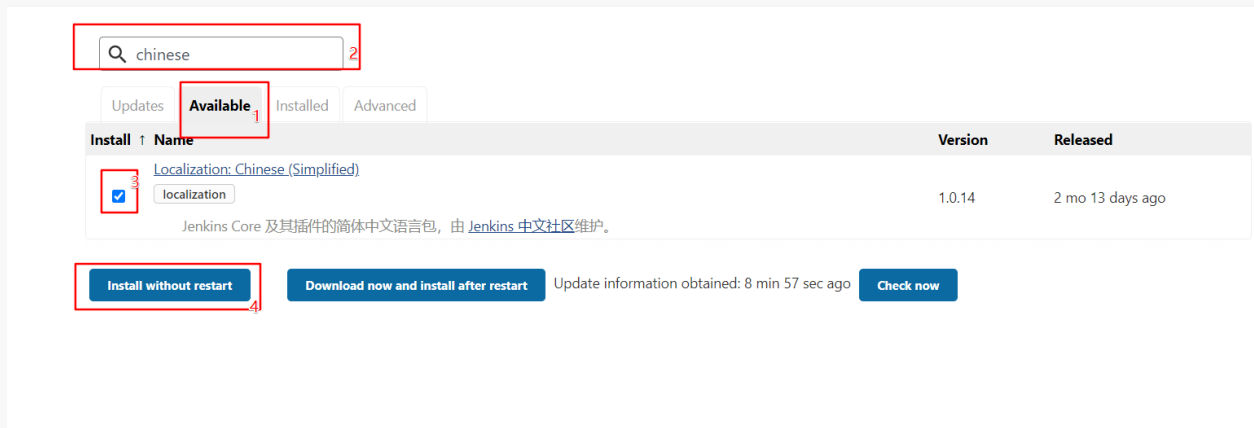
最后重启jenkins



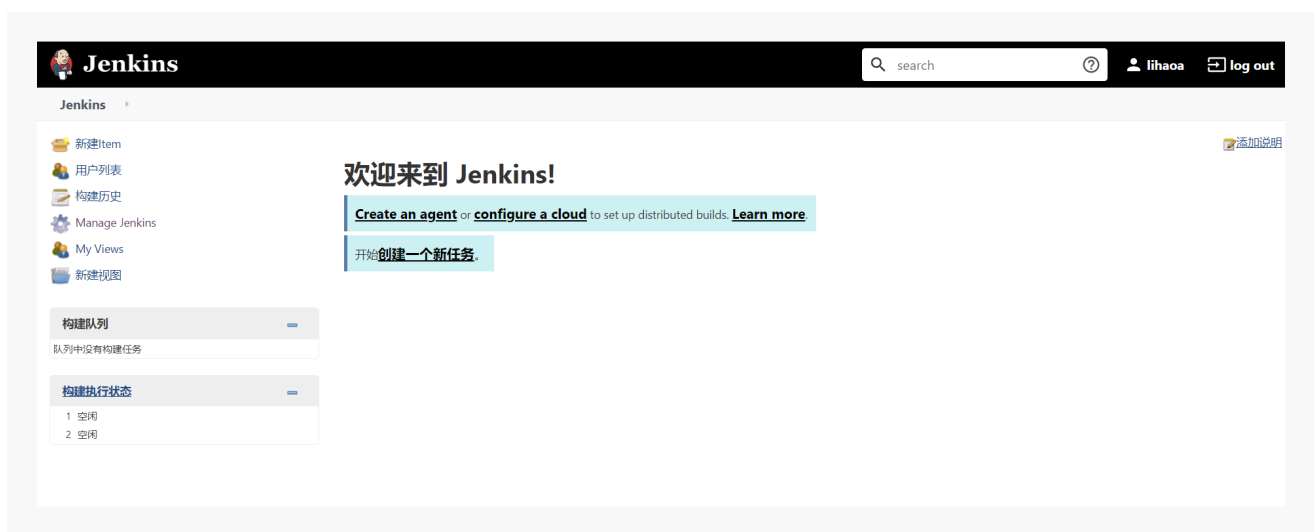
## 👤 14、Jenkins插件安装

### 👤 1、jenkins汉化插件安装

Jenkins->Manage Jenkins->Manage Plugins, 点击Available, 搜索"Chinese"

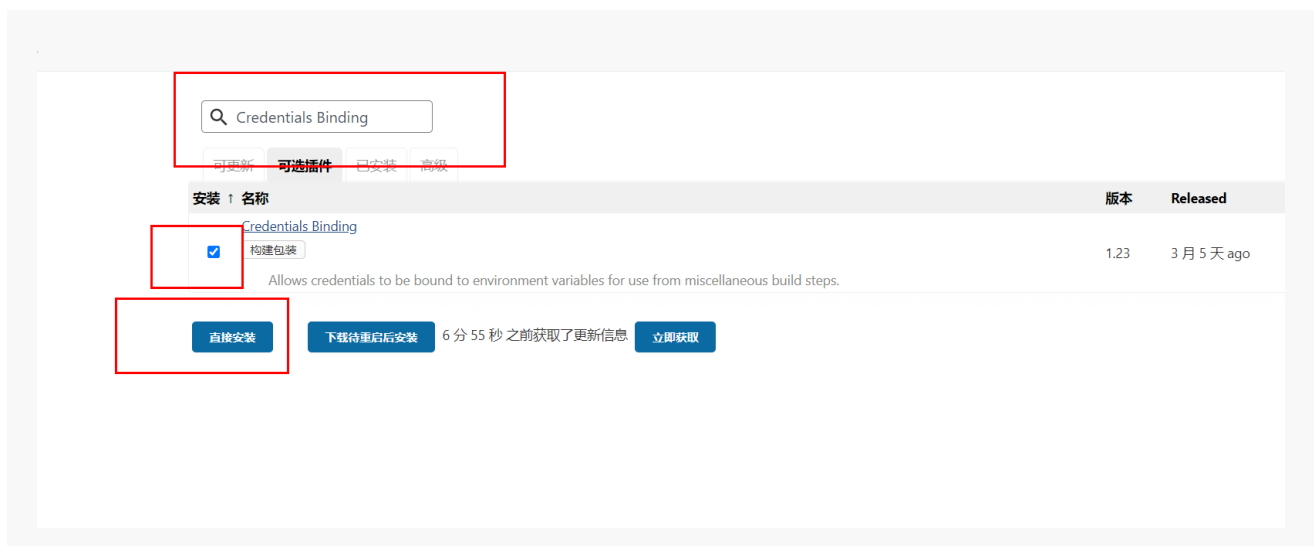


按照图片所示安装完,回到首页可以看到,界面已经是中文了

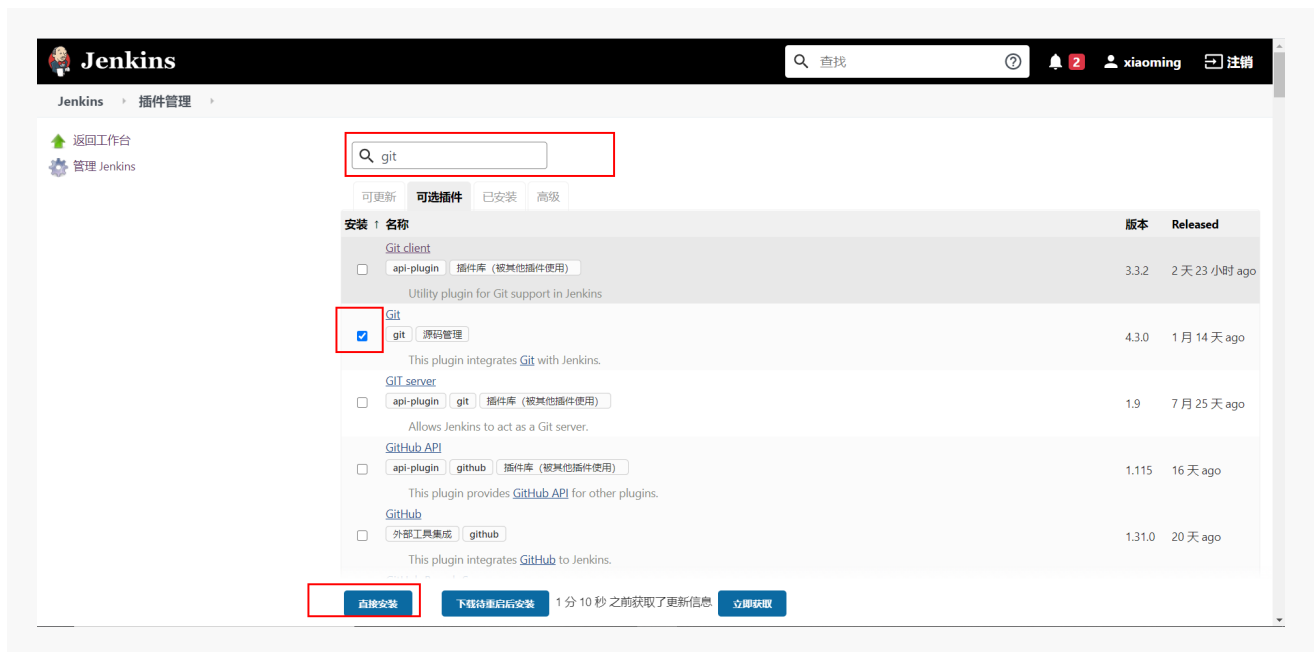


## 2、安装Credentials Binding插件

此插件用于存储凭据

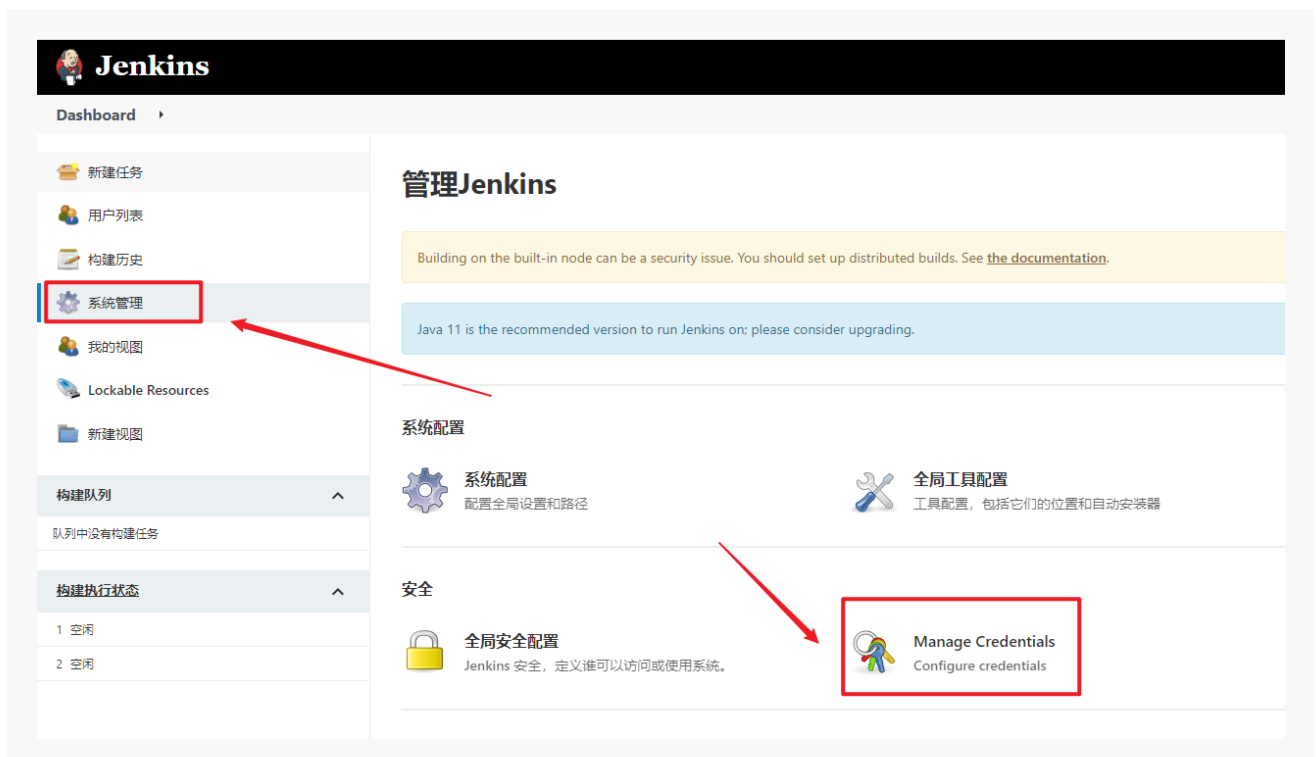


### 🐼 3、git插件安装



安装完开始配置git的账号

### 🐼 15、配置凭据 - 全局Git账号和密码管理



Jenkins > 凭据

新建任务

用户列表

构建历史

系统管理

我的视图

新建视图

构建队列

队列中没有构建任务

构建执行状态

1 空闲

2 空闲

凭据

| 类型                              | 提供者     | 存储 ↓ | 域  | 唯一标识 | 名称 |
|---------------------------------|---------|------|----|------|----|
| 图标: 小中大                         |         |      |    |      |    |
| Stores scoped to <b>Jenkins</b> |         |      |    |      |    |
| 提供者                             | 存储 ↓    | 域    |    |      |    |
|                                 | Jenkins |      | 全局 |      |    |

系统

| 域                   | 描述                                                                                                  |
|---------------------|-----------------------------------------------------------------------------------------------------|
| 全局凭据 (unrestricted) | Credentials that should be available irrespective of domain specification to requirements matching. |

图标: 小中大

Jenkins

查找

2

xiaoming

Jenkins > 凭据 > 系统 > 全局凭据 (unrestricted)

返回到凭据域列表

添加凭据

全局凭据 (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

| ID                        | 名称 | 类型 | 描述 |
|---------------------------|----|----|----|
| <a href="#">如何添加一些凭据?</a> |    |    |    |

图标: 小中大

Jenkins

查找

2

xiaoming

注销

Jenkins > 凭据 > 系统 > 全局凭据 (unrestricted)

返回到凭据域列表

添加凭据

类型

Username with password

范围

全局 (Jenkins, nodes, items, all child items, etc)

用户名

545329844@qq.com

密码

\*\*\*\*\*

ID

描述

小明的gitlab账号

确定

转圈图伸手接过银霜

生成页面: 2020-8-3 下午02:28:04分00秒 REST API Jenkins 2.235.1

全局凭据 (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

| ID                                                                                                                        | 名称              | 类型                     | 描述              |
|---------------------------------------------------------------------------------------------------------------------------|-----------------|------------------------|-----------------|
|  2bfa581b-04f6-42b2-ad1a-5d977de9c466 * | 我的gitee平台的账号和密码 | Username with password | 我的gitee平台的账号和密码 |

图标 小 中 大

这个后续使用用到

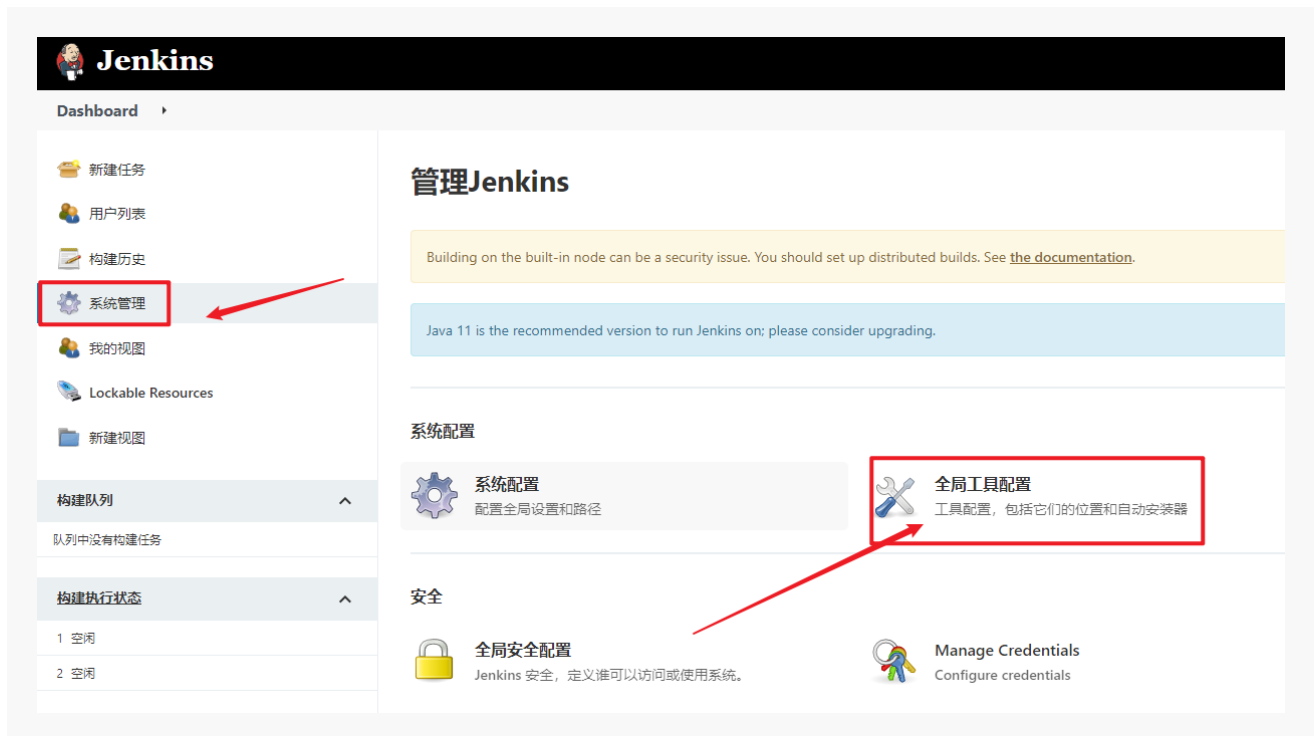
配置你代码所在的gitee的账号和密码即可，这里仅仅只是一个保存账号和密码存档的小工具而已。给后续的配置提供一个便签，后续就不需要在输入账号和密码了。配置完毕。也是为了安全性考虑。

这个ID等下在jenkinsfile文件中，代码的下载需要这个id来进行关联。也就是通过id找到你的账号和密码，然后去远程下载代码。

1 6063224a-54a1-4474-a953-934de62262dd

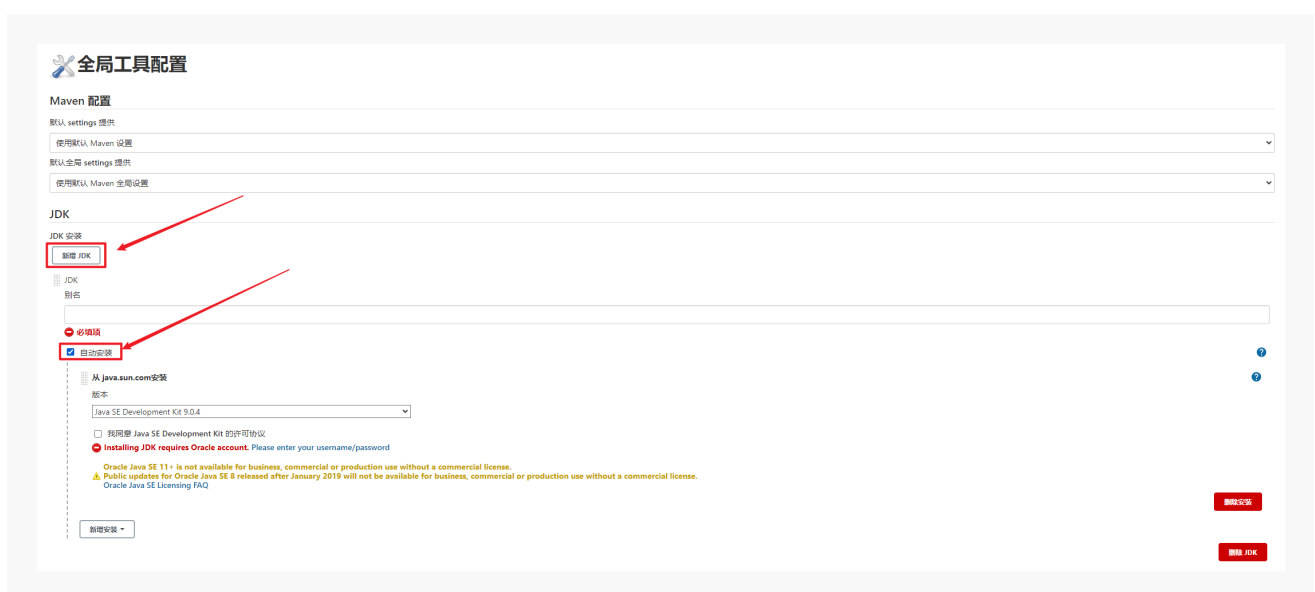
## 16、配置环境变量

当我们拉取项目以后,需要用到maven对我们的项目进行构建,我们已在虚拟机上安装了java环境和maven环境,我们需要在jenkins配置应用上去,进入到全局工具配置



## 配置jdk

点击取消自动安装



名称可以随意填,JAVA\_HOME是虚拟机配置的JAVA\_HOME路径,可以通过以下命令查看

```
1 echo $JAVA_HOME
```

如下：

```
[root@iZwz9gi039o35ikybyj1tZ maven]# echo $JAVA_HOME
/usr/lib/jvm/java-1.8.0-openjdk
```

The screenshot shows a web-based configuration interface. The top section is titled 'JDK' and contains a 'JDK 安装' (JDK Installation) subsection. It features a '新增 JDK' (Add JDK) button, a table with columns 'JDK' and '别名' (Alias), and a 'JAVA\_HOME' field. The '别名' column contains 'JDK1.8' and the 'JAVA\_HOME' field contains '/usr/lib/jvm/java-1.8.0-openjdk'. Both are highlighted with red boxes and red arrows. Below this is an unchecked checkbox for '自动安装' (Automatic Installation). The bottom section is titled 'Git' and contains a 'Git installations' subsection. It has a '新增 JDK' (Add JDK) button, a table with columns 'Git' and 'Name', and a 'Path to Git executable' field. The 'Name' column contains 'Default' and the 'Path to Git executable' field contains 'git'. Both are highlighted with red boxes and red arrows. At the bottom of the Git section are two buttons: '保存' (Save) and '应用' (Apply). The '保存' button is highlighted with a red box and a red arrow.

JDK

JDK 安装

新增 JDK

| JDK    | 别名 |
|--------|----|
| JDK1.8 |    |

JAVA\_HOME

/usr/lib/jvm/java-1.8.0-openjdk

☐ 自动安装

新增 JDK

系统下JDK 安装列表

Git

Git installations

| Git | Name    |
|-----|---------|
|     | Default |

Path to Git executable ?

git

☐ 自动安装 ?

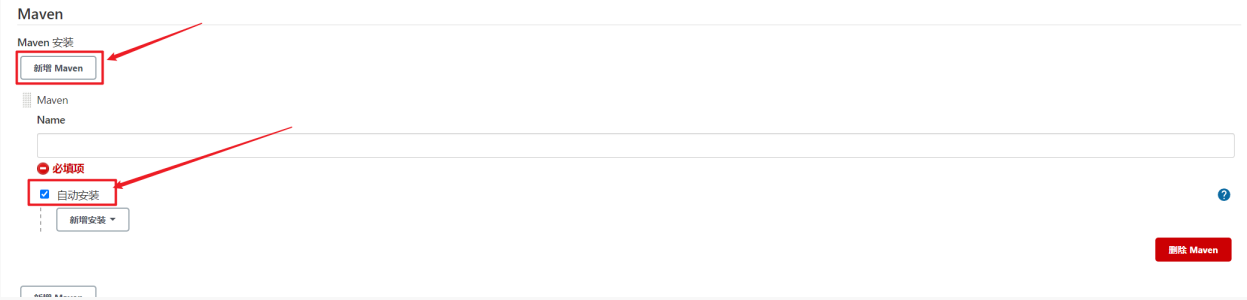
保存 应用

点击下面的保存即可。



# 配置Maven

点击 **取消自动安装**



Maven

Maven 安装

新增 Maven

Maven

Name

必选项

☒ 自动安装

新增安装 +

删除 Maven

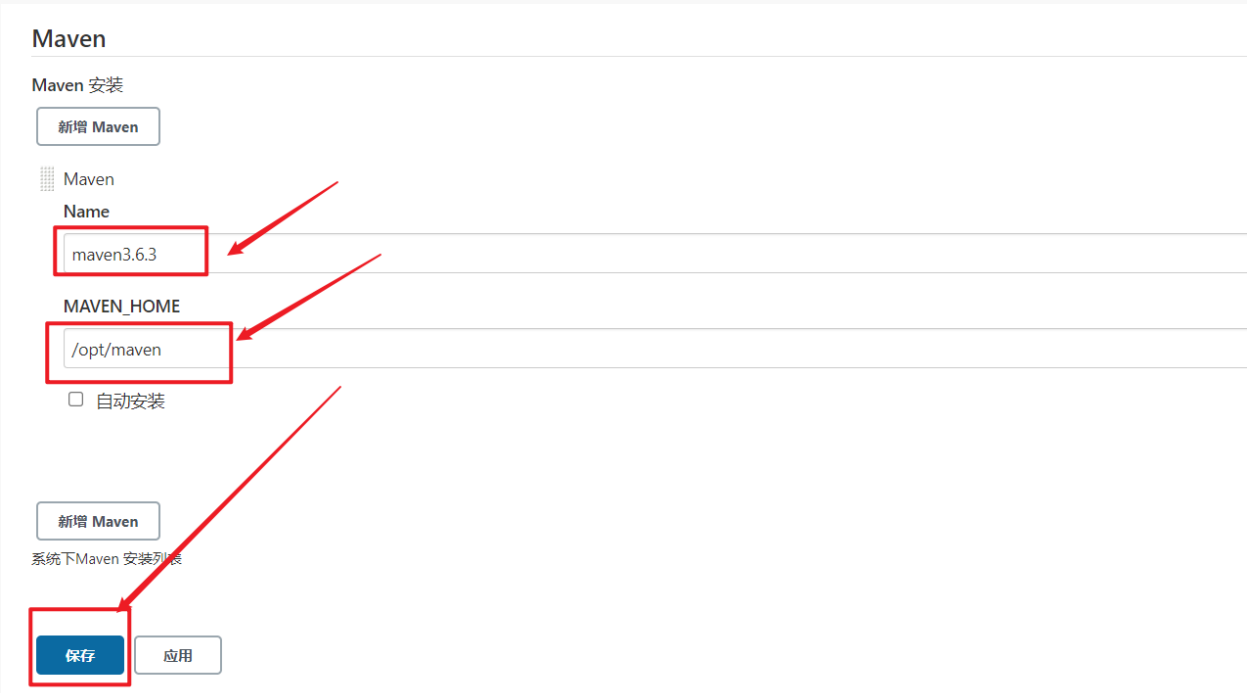
然后配置

名称可以随意填,MAVEN\_HOME是虚拟机配置的MAVEN\_HOME路径,可以通过以下命令查看

```
1 echo $MAVEN_HOME
```

如下:

```
[root@iZwz9gi039o35ikykyj1tZ maven]# echo $MAVEN_HOME
/opt/maven
```



Maven

Maven 安装

新增 Maven

Maven

Name

maven3.6.3

MAVEN\_HOME

/opt/maven

☐ 自动安装

新增 Maven

系统下Maven 安装列表

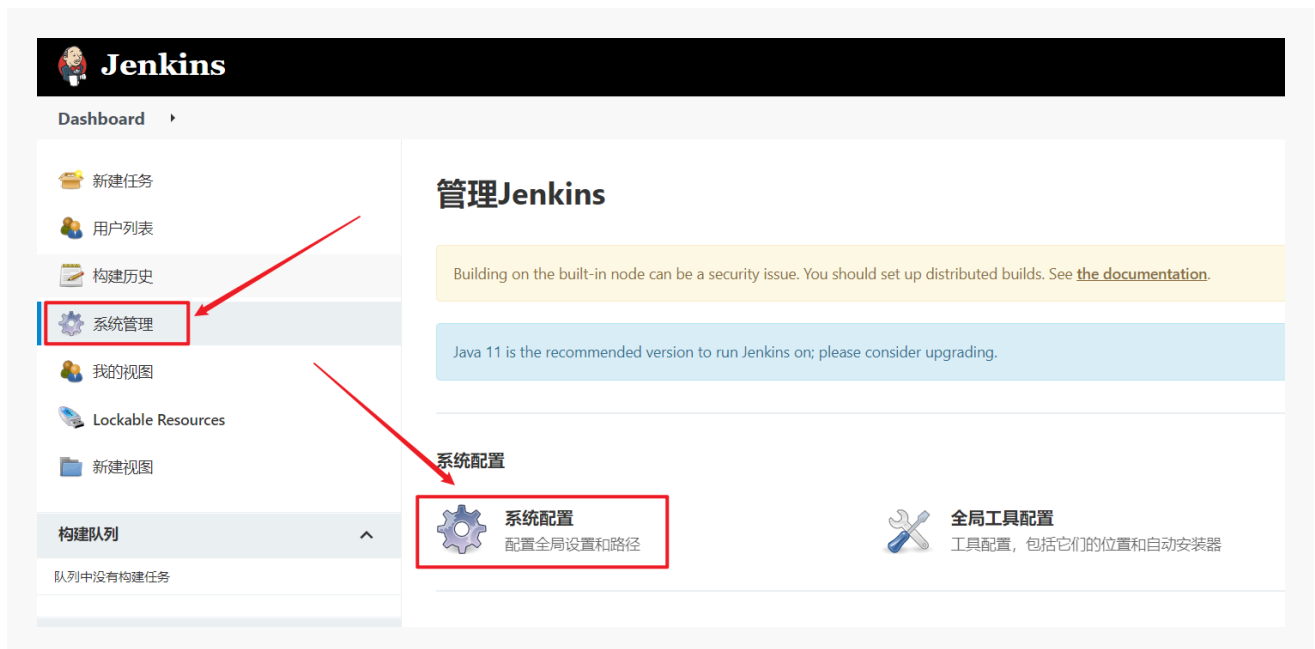
保存 应用

## 17、Jenkins关联环境变量和设置常量

就是给Jenkinsfile设置的常量。而这常量我可以在Jenkinsfile文件中用`${xxx}`

此外,我们还需要将环境变量配置到jenkins(重要步骤)

- 找到: Jenkins ==> 系统管理 ==> 系统配置
- 找到全局属性,将配置的环境路径,挨个拷贝过来



全局属性

☐ Disable deferred wipeout on this node

☐ 工具位置

☒ 环境变量

键值对列表

| 键          | 值                               |
|------------|---------------------------------|
| JAVA_HOME  | /usr/lib/jvm/java-1.8.0-openjdk |
| MAVEN_HOME | /opt/maven                      |
| PATH+EXTRA | \$MAVEN_HOME/bin                |

新增

保存 应用

此时,jenkins的maven环境、git环境都已搭建好，考虑到我们需要发布服务，所以还需要安装docker

## 02、基于Jenkins的jar的方式发布和部署

前提：

- jdk1.8
- maven3.6.3

地址：<http://maven.apache.org/download.cgi>

各版本地址: <https://archive.apache.org/dist/maven/maven-3/>

我以apache-maven-3.6.3-bin.tar.gz为例,

下载地址: <https://archive.apache.org/dist/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz>

- git 下载地址: <https://git-scm.com/download>

## 01、准备项目

创建一个springboot项目

## 02、创建代码仓库

码云地址: <https://gitee.com/>

在gitee创建一个仓库即可, 注意是开源的。



## 新建仓库

在其他网站已经有仓库了吗? [点击导入](#)

仓库名称 \* ✓

springboot-pug-jar

1

归属

学相伴-飞哥

路径 \* ✓

springboot-pug-jar

仓库地址: <https://gitee.com/kekesam/springboot-pug-jar>

仓库介绍

0/100

用简短的语言来描述一下吧

☒ 开源 (所有人可见)

2

☐ 私有 (仅仓库成员可见)

☐ 企业内部开源 (仅企业成员可见) ②

☐ 初始化仓库 (设置语言、.gitignore、开源许可证)

☐ 设置模板 (添加 README、Issue、Pull Request 模板文件)

☐ 选择分支模型 (仓库创建后将根据所选模型创建分支)

创建

3

快速设置— 如果你知道该怎么操作, 直接使用下面的地址

已复制

HTTPS

SSH

<https://gitee.com/kekesam/springboot-pug-jar>

📄

我们强烈建议所有的git仓库都有一个 README、LICENSE、.gitignore 文件

初始化 readme 文件

Git入门? 查看 [帮助](#), [Visual Studio](#) / [TortoiseGit](#) / [Eclipse](#) / [Xcode](#) 下如何连接本站, [如何导入仓库](#)

简易的命令行入门教程:

Git 全局设置:

```
git config --global user.name "学相伴-飞哥"
git config --global user.email "xuchengfeifei@163.com"
```

创建 git 仓库:

```
mkdir springboot-pug-jar
cd springboot-pug-jar
git init
touch README.md
git add README.md
git commit -m "first commit"
git remote add origin https://gitee.com/kekesam/springboot-pug-jar.git
git push -u origin "master"
```

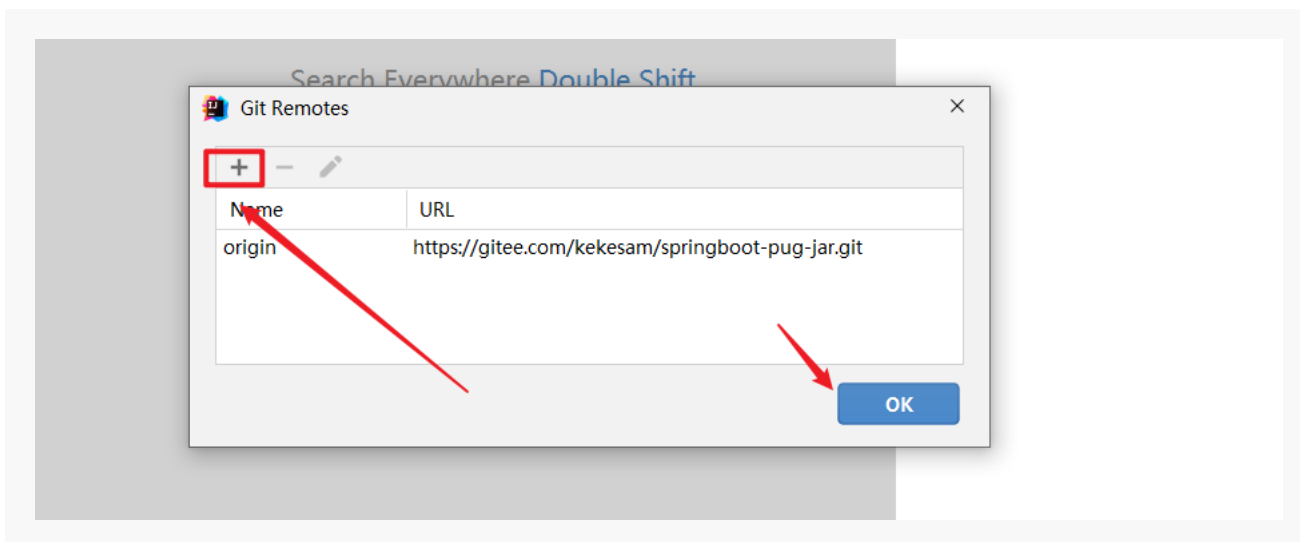
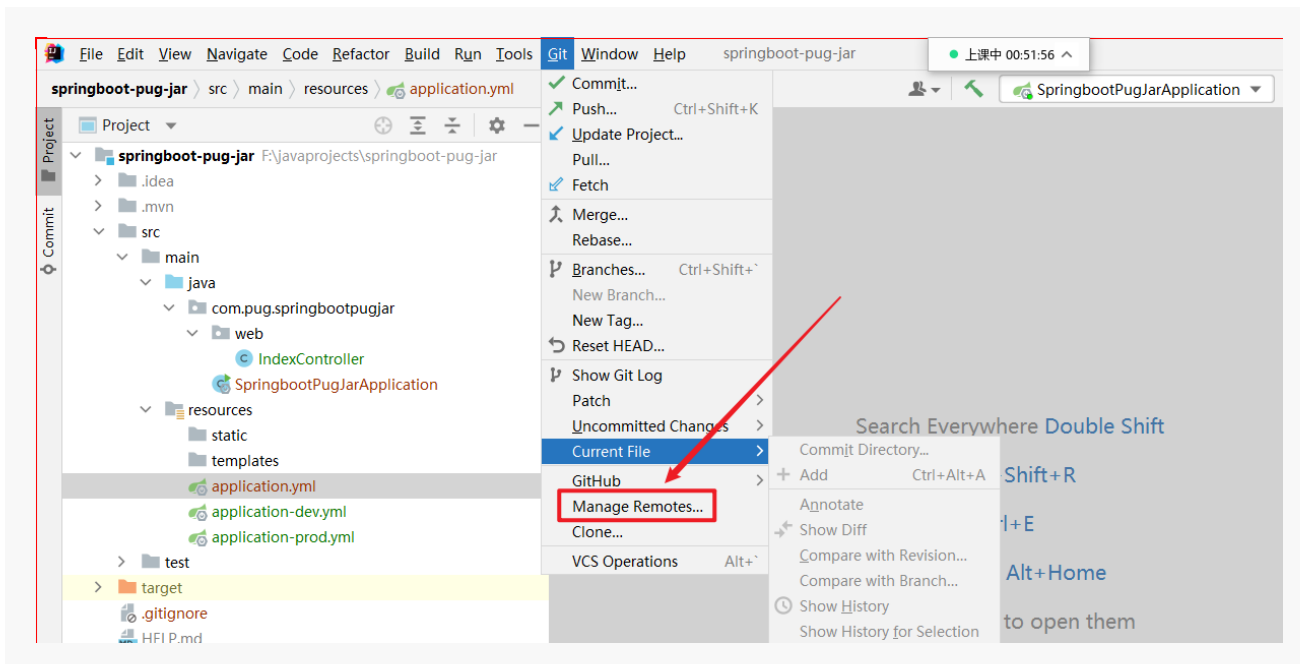
已有仓库?

```
cd existing_git_repo
git remote add origin https://gitee.com/kekesam/springboot-pug-jar.git
git push -u origin "master"
```

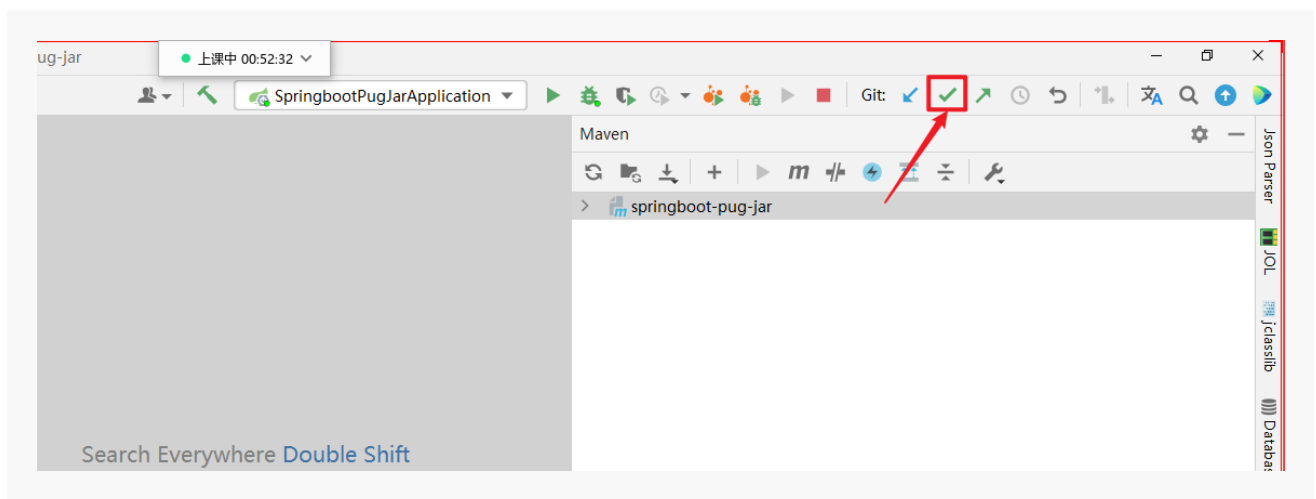
仓库代码地址:

1 <https://gitee.com/kekesam/springboot-pug-jar.git>

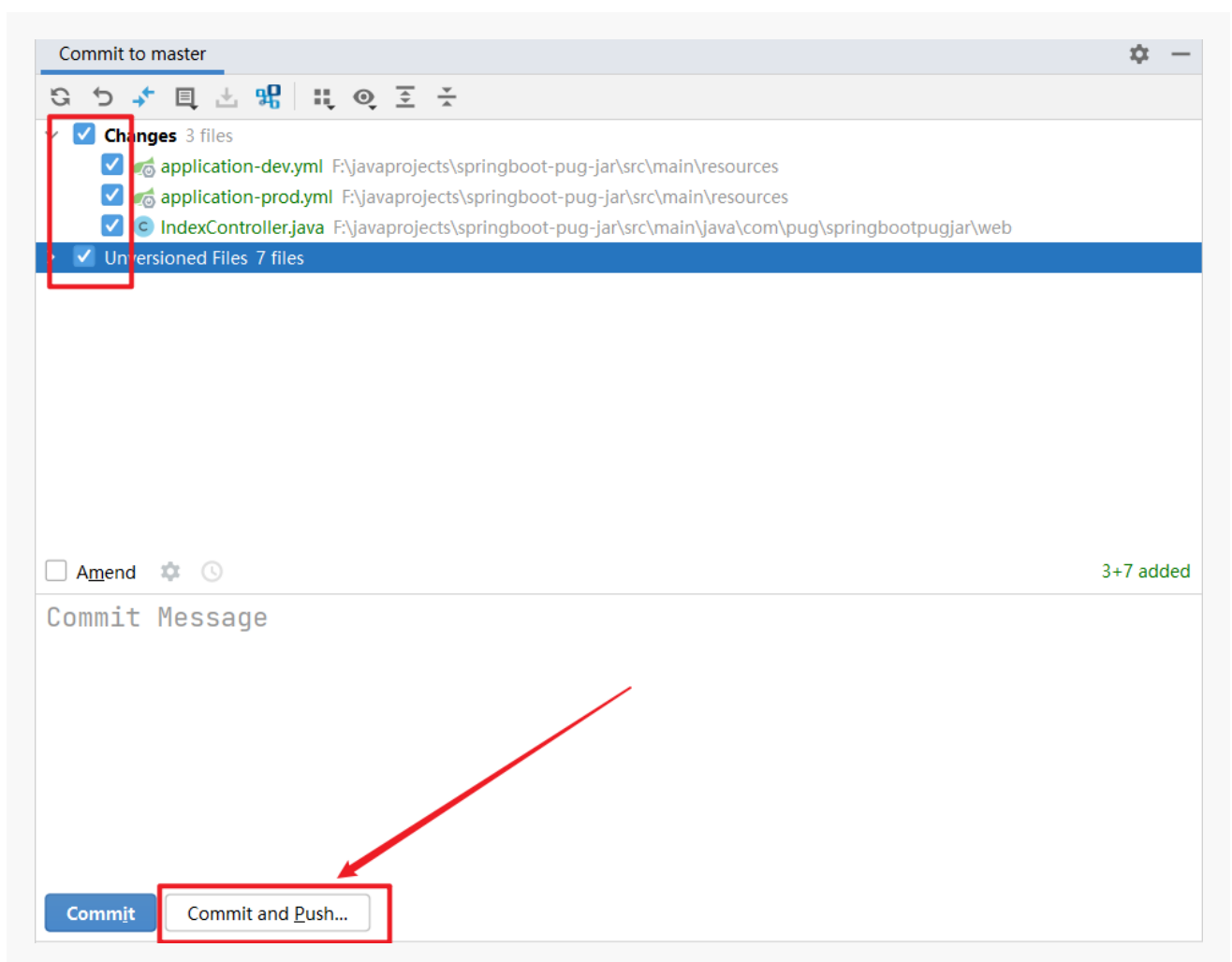
### 03、项目关联仓库

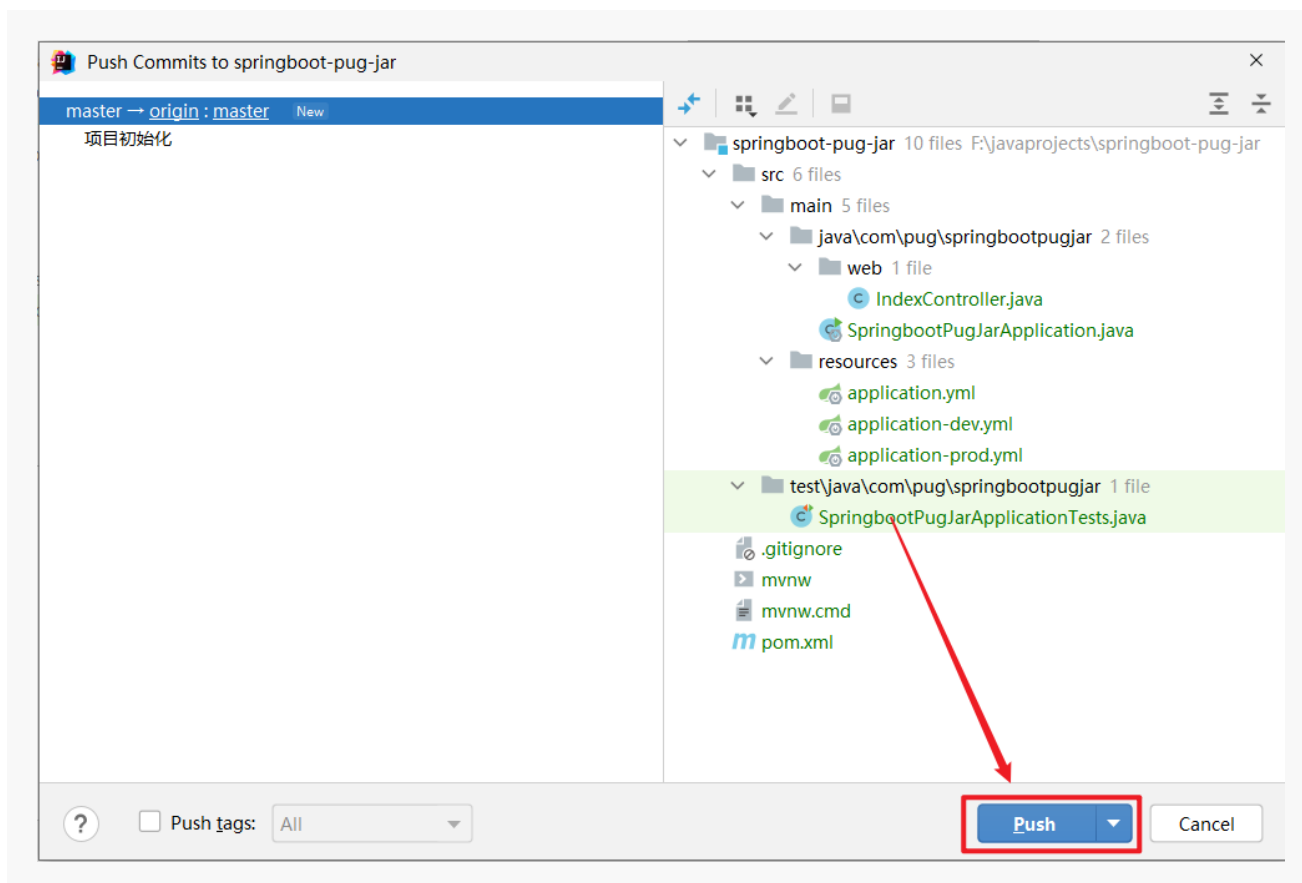


提交代码如下：



发布到远程仓库





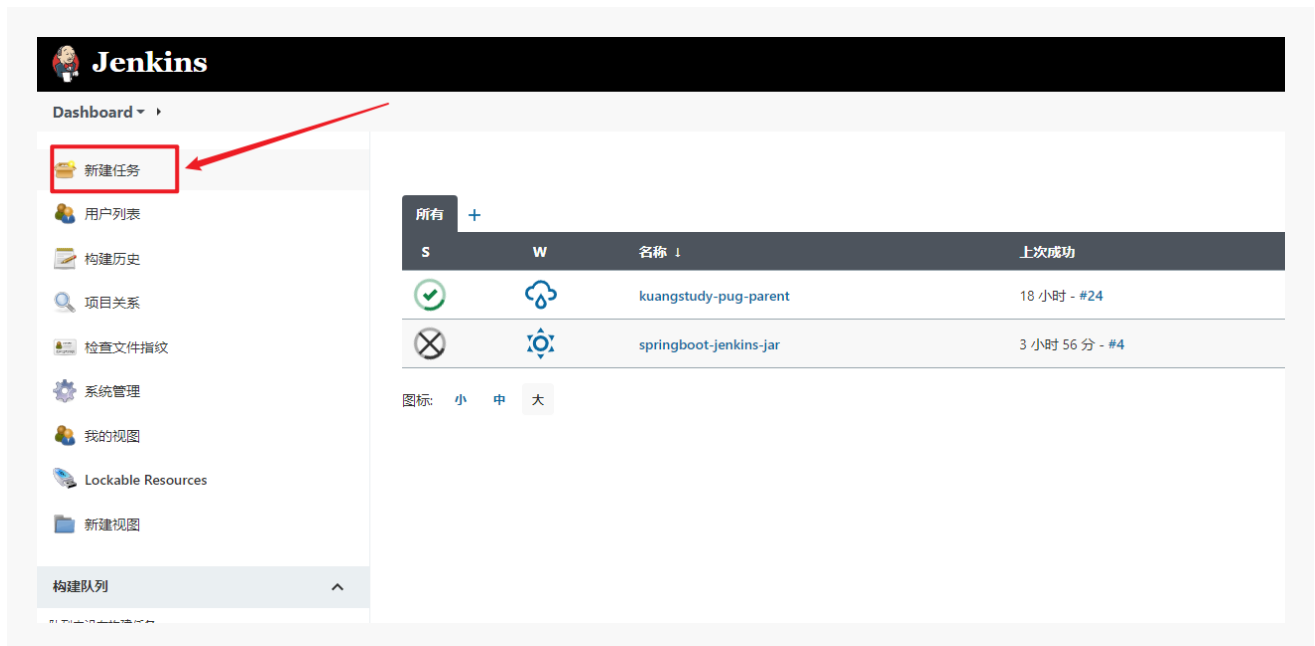
查看远程仓库



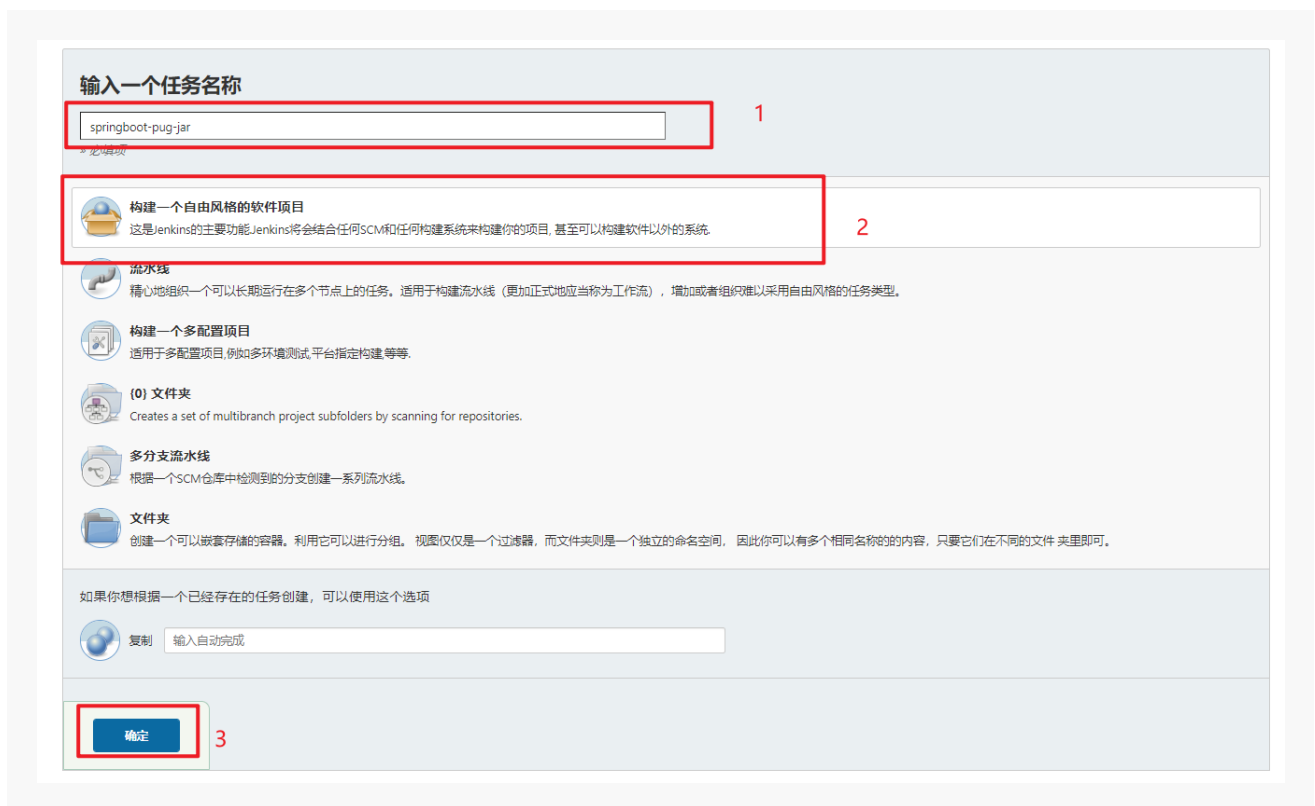


## 🤖 04、在jenkins创建一个任务

### 1: 创建一个任务



### 2: 创建一个自由风格的项目



General 源码管理 构建触发器 构建环境 构建 构建后操作

☐ 在必要的时候开发构建

源码管理

☐ 无

☒ Git

Repositories

Repository URL

https://gitee.com/kekesam/springboot-pug-jar.git

Credentials

qitree账号凭据 添加

Add Repository

Branches to build

指定分支 (为空格代表any)

\*/master

Add Branch

源码库浏览器

(自动)

Additional Behaviours

新增

构建触发器

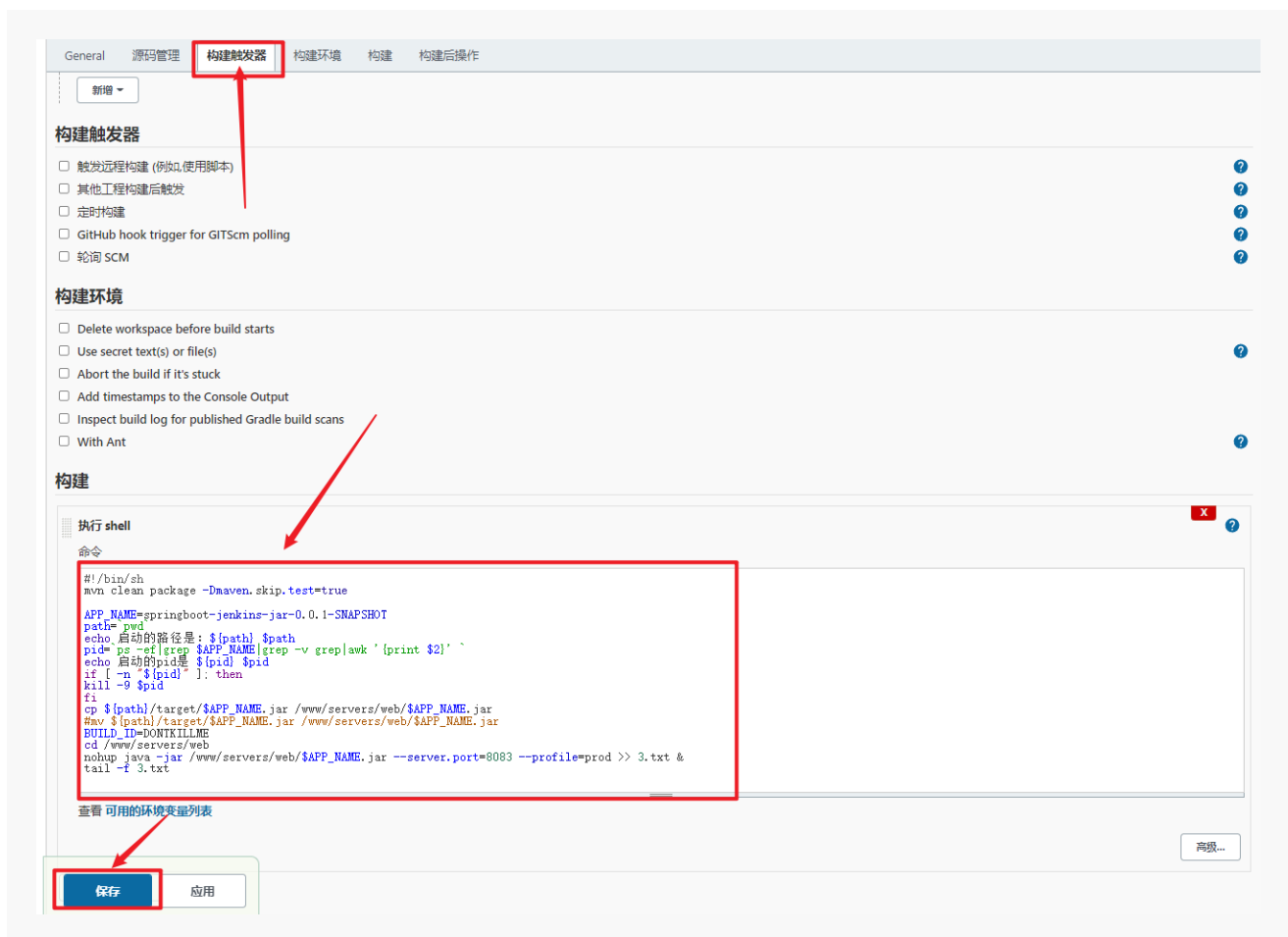
☒ 触发远程构建 (例如,使用脚本)

保存 应用

## 05、编写shell脚本和jenkins关联执行

shell脚本：它是独立语法结果，是一个linux命令集合体。

构建部署脚本



```
1 #!/bin/sh
2 # 当前工具区间打包，因为上面的git代码会把下载到jenkins的当前工具区间
  中。/var/lib/jenkins/workspace/springboot-pug-jar
3 mvn clean package -Dmaven.skip.test=true
4 # 应用程序的名称，默认artifactId-version
5 APP_NAME=springboot-pug-jar
6 echo ${APP_NAME} $APP_NAME
7 path=`pwd`
8 echo 启动的路径是: ${path} $path
9 # 查找linux进程中是否有相同的服务，如果有就把对应进程id查找。
10 pid=`ps -ef|grep $APP_NAME|grep -v grep|awk '{print $2}'`
11 # 打印服务器进程id
12 echo 启动的pid是 ${pid} $pid
13 # 找到以后将其上一次的服务的pid kill掉
14 if [ -n "${pid}" ]; then
15 kill -9 $pid
16 fi
17 # 把编译好的项目复制到指定目录
```

```

18 cp ${path}/target/$APP_NAME.jar
   /www/servers/web/$APP_NAME.jar
19 # 防止jenkins是关闭服务器的时候把当前也关闭
20 BUILD_ID=DONTKILLME
21 # 启动jar项目
22 nohup java -jar /www/servers/web/$APP_NAME.jar --
   server.port=${port} --profile=prod >> /www/servers/web/3.txt
   &
23 # 打印日志
24 tail -f /www/servers/web/3.txt

```

- 开发代码
- 打包

```

1 -jar mvn clean package -Dmaven.skip.test=true

```

- 把jar上传到服务器
- 把上一次的服务杀掉，停止

```

1 # 查找linux进程中是否有相同的服务，如果有就把对应进程id查找。
2 pid=`ps -ef|grep $APP_NAME|grep -v grep|awk '{print $2}'`
3 # 打印服务器进程id
4 echo 启动的pid是 ${pid} $pid
5 # 找到以后将其上一次的服务的pid kill掉
6 if [ -n "${pid}" ]; then
7 kill -9 $pid
8 fi

```

- 启动运行新的jar

```
1
2 cp ${path}/target/$APP_NAME.jar
   /www/servers/web/$APP_NAME.jar
3 #mv ${path}/target/$APP_NAME.jar
   /www/servers/web/$APP_NAME.jar
4 BUILD_ID=DONTKILLME
5 cd /www/servers/web
6 nohup java -jar /www/servers/web/$APP_NAME.jar --
   server.port=8083 --profile=prod >> 3.txt &
7 tail -f 3.txt
```

## 06、开始构建即可

 返回面板

 状态

 修改记录

 工作空间

 立即构建

 配置

 删除 工程

 重命名

Build History

构建历史

Filter builds...

#2

2022-1-12 下午9:26

#1

2022-1-12 下午9:22

Atom feed 全部 Atom feed 失败

工程 springboot-pug-jar

工作区

最新修改记录

相关链接

- 最近一次构建(#2),1分 58 秒之前
- 最近未成功的构建(#1),5分 44 秒之前
- 最近完成的构建(#1),5分 44 秒之前



```
8 pid=`ps -ef|grep $module_project |grep -v grep|awk '{print $2}'`
9 # 打印服务器进程id
10 echo 启动的pid是 ${pid} $pid
11 # 找到以后将其上一次的服务的pid kill掉
12 if [ -n "${pid}" ]; then
13 kill -9 $pid
14 fi
15 # 把编译好的项目复制到指定目录
16 cp ${path}/${module_project}/target/$module_project.jar
   /www/servers/web/$module_project.jar
17 # 防止jenkins是关闭服务器的时候把当前也关闭
18 BUILD_ID=DONTKILLME
19 # 启动jar项目
20 nohup java -jar /www/servers/web/$module_project.jar --
   server.port=${port} --profile=prod >> /www/servers/web/3.txt
   &
21 # 打印日志
22 tail -f /www/servers/web/3.txt
```