

大纲

- Jenkins简介
- Jenkins安装
- Jenkins集成插件实现自动流程

Jenkins简介

Jenkins是一个独立的开源自动化服务器，可用于自动执行与构建，测试，交付或部署软件相关的各种任务。

目前最流行的一款持续集成及自动化部署工具。

Jenkins 和Hudson 之间的关系：2009 年，甲骨文收购了Sun并继承了Hudson代码库。在 2011年，甲骨文和开源社区之间的关系破裂，该项目被分成两个独立的项目：

Jenkins：由大部分原始开发人员组成

Hudson：由甲骨文公司继续管理

所以Jenkins 和Hudson 是两款非常相似的产品。

概念理解

持续集成

功能角度：类似maven生命周期，集成各种插件。可以集成git, maven, sonar, 部署插件等。

项目角度：尽早发现项目整体运行问题，尽早解决。

持续交付

功能角度：功能迭代迅速，持续发布，不需要等待一个大版本再发布。

项目角度：研发团队的最新代码能够尽快让最终用户体验到。

Jenkins目标

1 降低风险

一天中进行多次的集成，并做了相应的测试，这样有利于检查缺陷，了解软件的健康状况，减少假定。

2 减少重复过程

产生重复过程有两个方面的原因，一个是编译、测试、打包、部署等等固定操作都必须要做，无法省略任何一个环节；另一个是一个缺陷如果没有及时发现，有可能导致后续代码的开发方向是错误的，要修复问题需要重新编写受影响的所有代码。而使用 Jenkins 等持续集成工具既可以构建环节从手动完成转换为自动化完成，又可以通过增加集成频次尽早发现缺陷避免方向性错误。

3 任何时间、任何地点生成可部署的软件

持续集成可以让您在任何时间发布可以部署的软件。从外界来看，这是持续集成最明显的好处，我们可以对改进软件品质和减少风险说起来滔滔不绝，但对于客户来说，可以部署的软件产品是最实际的资产。利用持续集成，您可以经常对源代码进行一些小改动，并将这些改动和其他的代码进行集成。如果出现问题，项目成员马上就会被通知到，问题会第一时间被修复。不采用持续集成的情况下，这些问题有可能到交付前的集成测试的时候才发现，有可能会导致延迟发布产品，而在急于修复这些缺陷的时候又有可能引入新的缺陷，最终可能导致项目失败。

Jenkins目标

4 增强项目的可见性

持续集成让我们能够注意到趋势并进行有效的决策。如果没有真实或最新的数据提供支持，项目就会遇到麻烦，每个人都会提出他最好的猜测。通常，项目成员通过手工收集这些信息，增加了负担，也很耗时。持续集成可以带来两点积极效果：

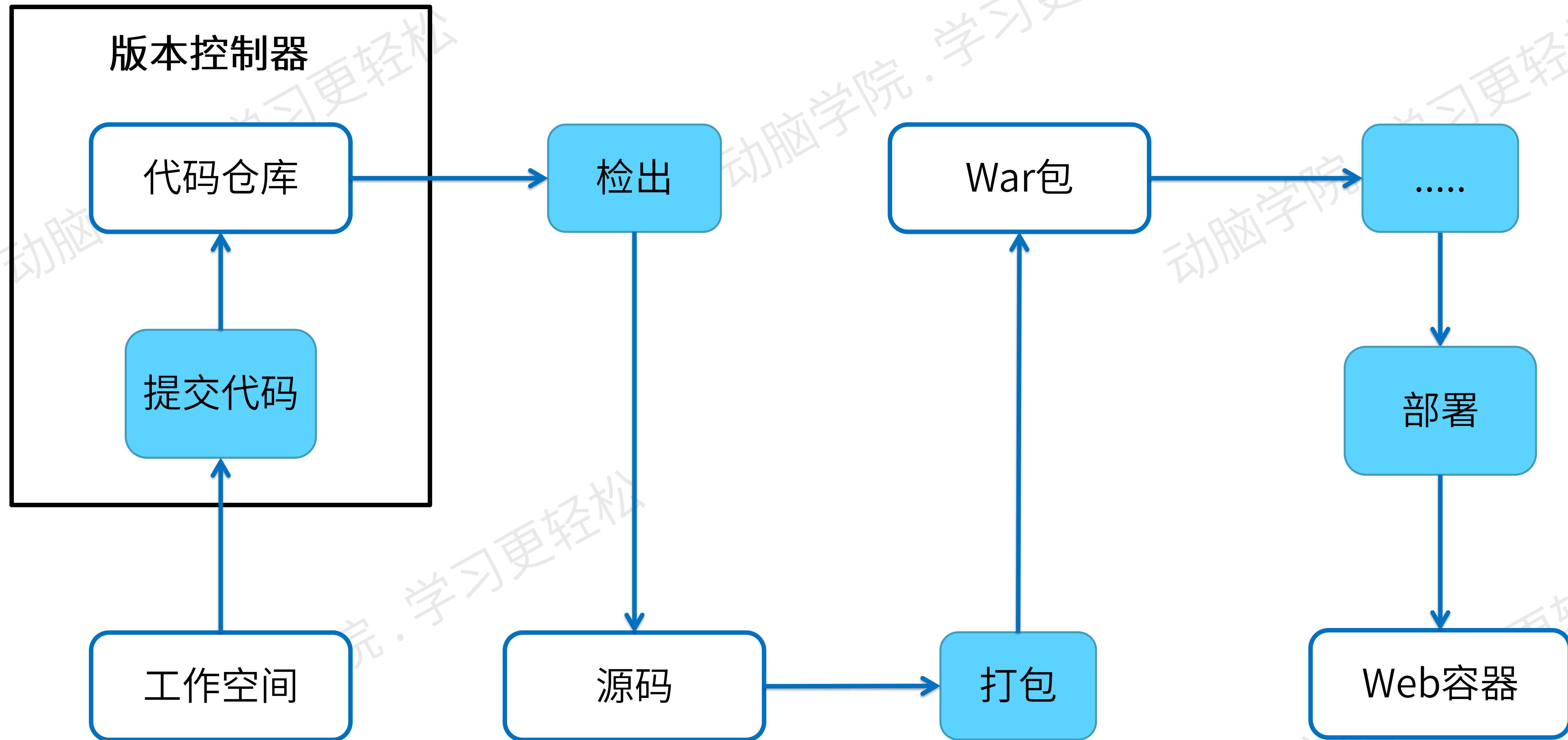
a 有效决策：持续集成系统为项目构建状态和品质指标提供了及时的信息，有些持续集成系统可以报告功能完成度和缺陷率。

b 注意到趋势：由于经常集成，我们可以看到一些趋势，如构建成功或失败、总体品质以及其它的项目信息。

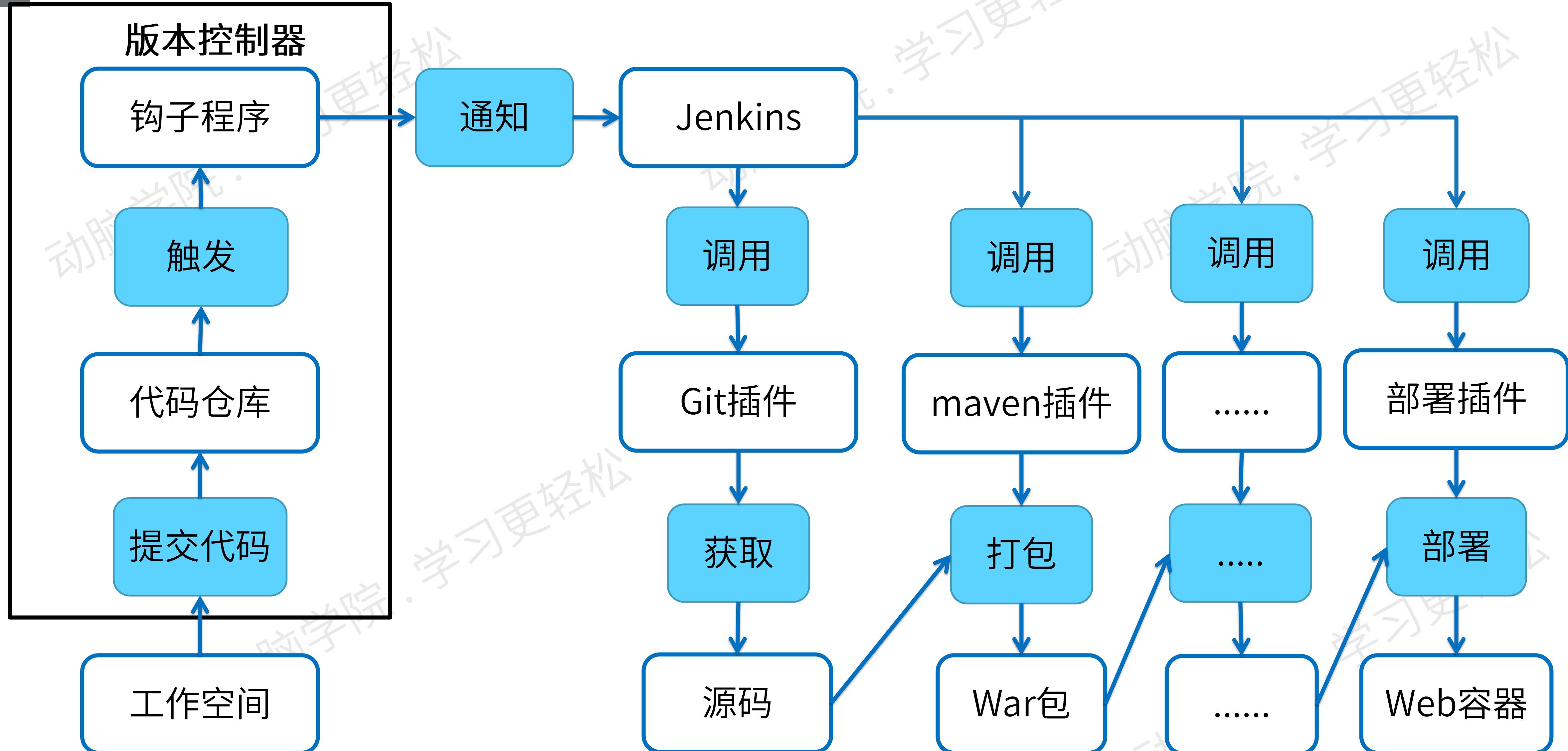
5 建立团队对开发产品的信心

持续集成可以建立开发团队对开发产品的信心，因为他们清楚的知道每一次构建的结果，他们知道他们对软件的改动造成了哪些影响，结果怎么样。

为什么需要Jenkins



为什么需要Jenkins



为什么需要Jenkins

向版本库提交新的代码后，应用服务器上自动部署，用户或测试人员使用的马上就是最新的应用程序。

搭建上述持续集成环境可以把整个构建、部署过程自动化，很大程度上减轻工作量。

对于程序员的日常开发来说不会造成任何额外负担——自己把代码提交上去之后，服务器上运行的马上就是最新版本。

Jenkins安装准备

开发要求：熟悉使用git、maven、sonar工具

环境要求：服务器需要安装jdk1.8、git、gitLab或者直接用GitHub、maven、sonar，sonarScanner、tomcat等

Jenkins是用java写的，所以必须需要jdk环境

tips:这里选择centos7 64位的系统来进行讲解。

Jenkins安装准备GIT

Git安装

1 卸载Centos自带的git，可通过git --version查看系统是有git

```
# yum remove git
```

2. 安装所需软件包

```
# yum install curl-devel expat-devel gettext-devel openssl-devel zlib-devel asciidoc  
# yum install gcc perl-ExtUtils-MakeMaker
```

3 下载git2.2.1并将git添加到环境变量中，或者直接在浏览器器打开输入：

<https://mirrors.edge.kernel.org/pub/software/scm/git/>，选择你要下载的版本

```
# wget https://github.com/git/git/archive/v2.2.1.tar.gz
```

4 解压

```
# tar zxvf v2.2.1.tar.gz
```

Jenkins安装准备GIT

Git安装

5 到解压目录执行编译和安装

```
# make prefix=/usr/local/git all      按前缀进行编译  
# make prefix=/usr/local/git install  进行安装
```

6 配置环境变量并且重新加载系统配置文件

```
# vim /etc/profile  
# export PATH=$PATH:/usr/git/git/bin  
# source /etc/profile
```

7 查看是否安装成功

```
# git --version
```

8 生成密钥，配置到gitLab或GitHub上

Jenkins安装准备MAVEN

Maven安装

1 下载maven 包，或者直接在浏览器器打开输入：

<http://mirrors.hust.edu.cn/apache/maven/maven-3/>，选择你要下载的版本

```
# wget http://mirrors.hust.edu.cn/apache/maven/maven-3/3.1.1/binaries/apache-maven-3.1.1-bin.tar.gz
```

2 解压

```
# tar zxf apache-maven-3.1.1-bin.tar.gz
```

3 配置环境变量并且重新加载系统配置文件

```
# vim /etc/profile  
# export PATH=$PATH:/usr/maven/apache-maven-3.3.9/bin  
# source /etc/profile
```

4 查看是否安装成功

```
# mvn -v
```

5 根据自己的需求设置是否要修改settings.xml中央仓库和添加本地仓库

Jenkins安装准备SonarScanner

SonarScanner安装

1 下载SonarScanner包，或者直接在浏览器器打开输入：

<https://docs.sonarqube.org/display/SCAN/Analyzing+with+SonarQube+Scanner>，选择你要下载的版本

```
# wget https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-3.2.0.1227-linux.zip
```

2 解压

```
# unzip sonar-scanner-cli-3.2.0.1227-linux.zip
```

3 修改sonar-scanner.properties，指定sonarQube服务器

```
sonar.host.url=http://192.168.1.149:9000/sonarQube
```

4 配置环境变量并且重新加载系统配置文件

```
# vim /etc/profile  
# export PATH=$PATH:/usr/sonar6.7.6/sonarQubeScanner/sonar-scanner-3.2.0.1227-linux/bin  
# source /etc/profile
```

5 查看是否安装成功

```
# sonar-scanner -h
```

Jenkins安装

1 官网下载war包

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

2 配置tomcat/server.xml文件，添加

```
<Connector port="8080" protocol="HTTP/1.1"  
connectionTimeout="20000"  
redirectPort="8443" URIEncoding="UTF-8"/>
```

3 将war包放入tomcat/webapps下，启动tomcat，浏览器输入网址即可

<http://192.168.1.149:8080/jenkins>

Jenkins初始配置

1 解锁Jenkins

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：

`/root/.jenkins/secrets/initialAdminPassword`

请从本地复制密码并粘贴到下面。

管理员密码

依照提示，查看
`/root/.jenkins/secrets/initialAdminPassword` 文件内容
填入文本框。

填入的内容也是
admin 账号的密码。

继续

Jenkins初始配置

2 选择插件安装方式，选择哪种方式都不会对后续操作有影响，有需要的插件可以在后期进行安装
这里选择安装推荐的组件，服务器必须联网

新手入门

自定义jenkins

插件通过附加特性来扩展jenkins以满足不同的需求。

安装推荐的插件

安装jenkins社区推荐的插件。

选择插件来安装

选择并安装最适合的插件。

Jenkins初始配置

3 新建用户，可以直接跳过，继续用admin帐号，后期也可以注册

新手入门

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安全配置

全局安全配置



全局安全配置

启用安全性

禁用记住我

[访问控制](#)

[安全领域](#)

[詹金斯专有用户数据库](#)

[允许用户注册](#)

[LDAP](#)

[Servlet的容器代理](#)

[Unix的用户/组数据库](#)

[授权](#)

[任何用户可以做任何事 \(没有任何限制\)](#)

[安全矩阵](#)

[登录用户可以做任何事](#)

[允许匿名读取访问权限](#)

[遗留模式](#)

[项目矩阵授权策略](#)

[保存](#)

[应用](#)

Jenkins插件管理

在正式使用Jenkins之前，我们先看下插件管理，了解如何使用插件管理
目前将我们必须要的插件集成进来，才能在后续的配置菜单中看到对应的选项
安装插件时受到网络状况的影响有可能会失败，多试几次，直到成功

The screenshot shows the Jenkins Plugin Manager interface. At the top, there's a navigation bar with 'Jenkins' and '插件管理'. Below it are links to '回到仪表板' and '管理Jenkins'. A search bar labeled '过滤:' is also present. The main area has tabs for '更新', '可得到', '安装' (which is selected), and '高级'. A table lists installed plugins:

启用	名称 ↓	版	以前安装的版本	卸载
<input checked="" type="checkbox"/>	Deploy to container Plugin This plugin allows you to deploy a war to a container after a successful build.	1.13		Uninstall
<input checked="" type="checkbox"/>	Gitlab Hook Plugin Enables Gitlab web hooks to be used to trigger SMC polling on Gitlab projects	1.4.2		Uninstall
<input checked="" type="checkbox"/>	GitLab Plugin This plugin allows GitLab to trigger Jenkins builds and display their results in the GitLab UI.	1.5.11		Uninstall
<input checked="" type="checkbox"/>	SonarQube Scanner for Jenkins This plugin allows an easy integration of SonarQube , the open source platform for Continuous Inspection of code quality.	2.8.1		Uninstall
.....				

Jenkins插件配置Maven

全局工具配置—集成maven

1 指定服务器上settings.xml文件位置

Maven配置

默认设置提供商

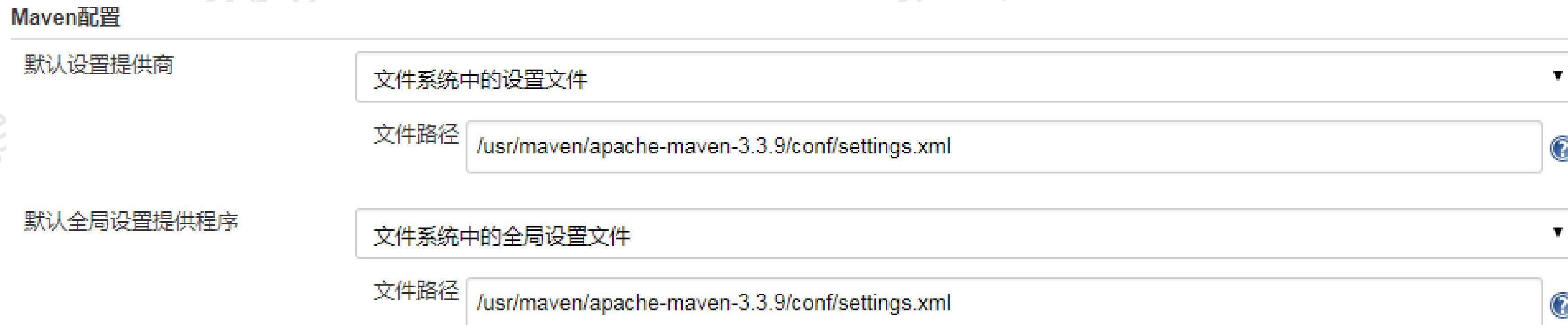
文件系统中的设置文件

文件路径 /usr/maven/apache-maven-3.3.9/conf/settings.xml

默认全局设置提供程序

文件系统中的全局设置文件

文件路径 /usr/maven/apache-maven-3.3.9/conf/settings.xml



2 指定maven安装路径，不需要配置到bin，配置到上级目录就行，将自动安装去掉

Maven安装

添加Maven

Maven的
名称

名称 MyMaven

MAVEN_HOME

MAVEN_HOME /usr/maven/apache-maven-3.3.9

自动安装

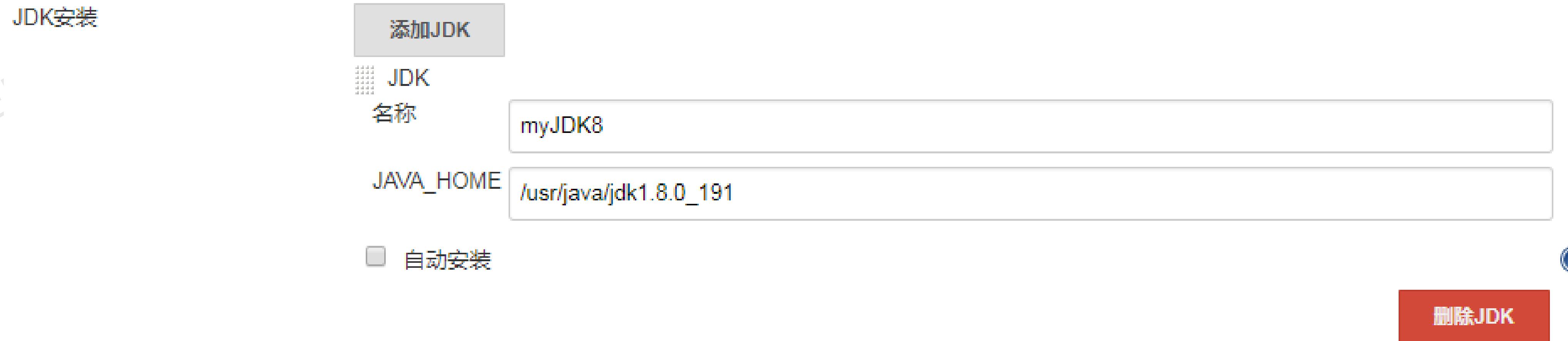
删除Maven



Jenkins插件配置JDK

全局工具配置—JDK

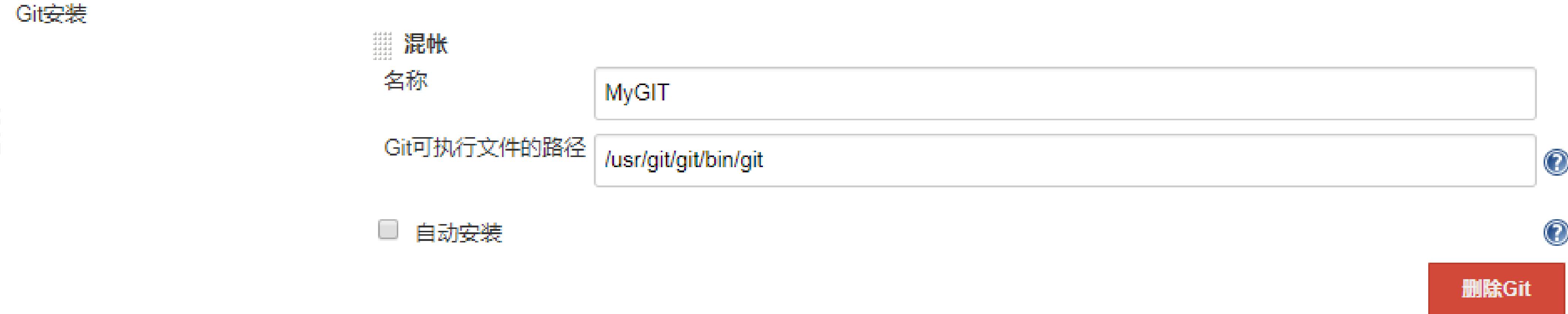
指定JDK安装路径，不需要配置到bin，配置上级目录就行，将自动安装去掉



Jenkins插件配置GIT

全局工具配置—集成GIT

指定GIT安装路径，需要配置到bin目录下的git文件，将自动安装去掉



Jenkins初使用

现在我们将项目必须JDK环境，源码检出（GIT），自动构建（Maven）插件都配置好了，现在使用一次看看效果

1 new item新建一个项目,选择自由风格的项目

Enter an item name

» Required field



构建一个自由风格的软件项目
这是Jenkins的主要功能.Jenkins将会结合任何SCM和任何构建系统来构建你的项目,甚至可以构建软件以外的系统.



流水线
精心地组织一个可以长期运行在多个节点上的任务。适用于构建流水线（更加正式地应当称为工作流），增加或者组织难以采用自由风格的任务类型。

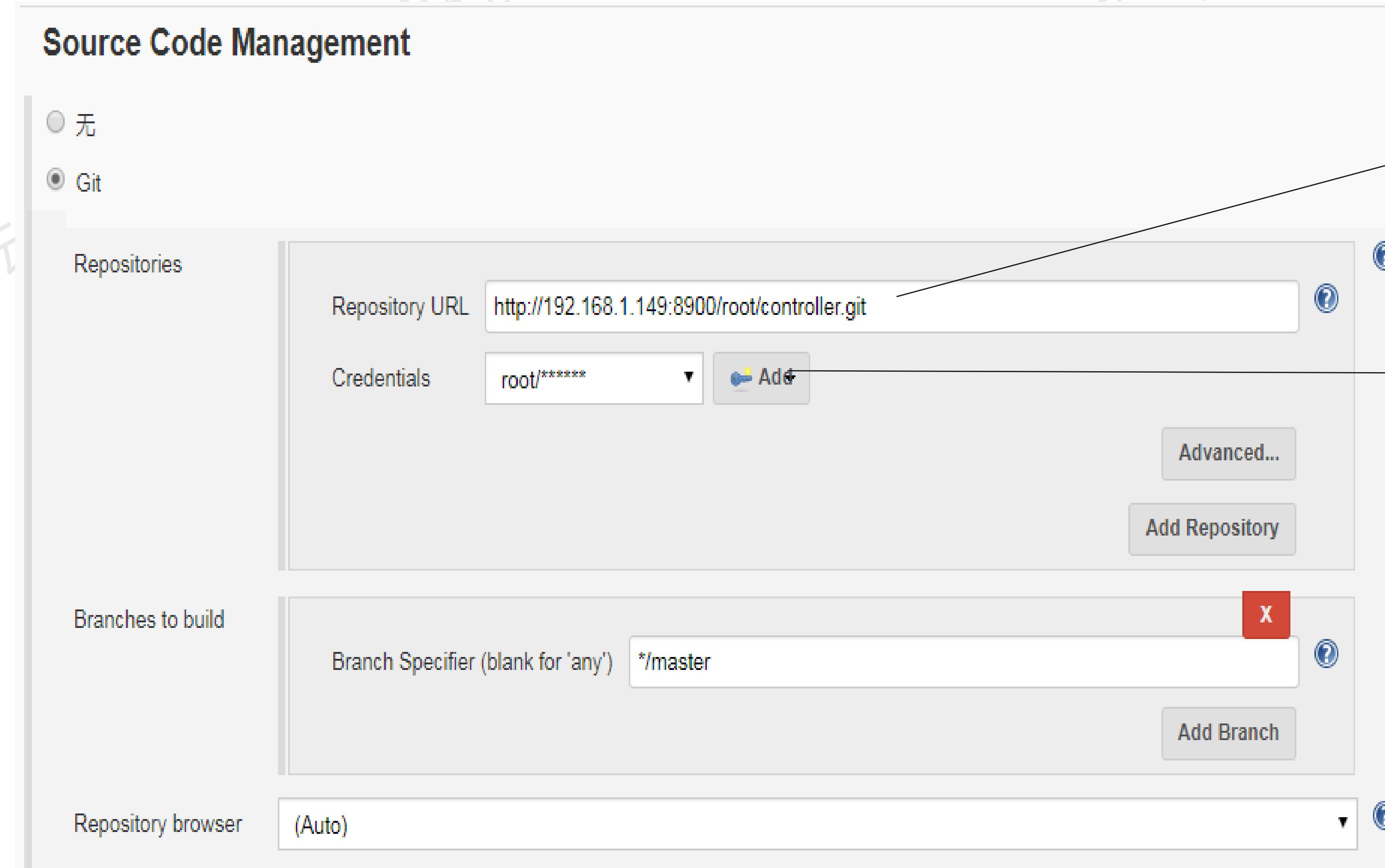


构建一个多配置项目
适用于多配置项目,例如多环境测试,平台指定构建,等等.

Jenkins初使用

2 选中刚刚新建的项目，点击右侧的configure配置

a 配置源码管理，选中git



选择源码位置
(gitLab或者gitHub)

填写gitLab或者
gitHub的帐号密码，
然后选中配置好的
帐号密码

Jenkins初使用

2 选中刚刚新建的项目，点击右侧的configure配置

b 配置构建，下拉框中选中maven

选择配置好的maven插件



配置希望maven执行构建命令

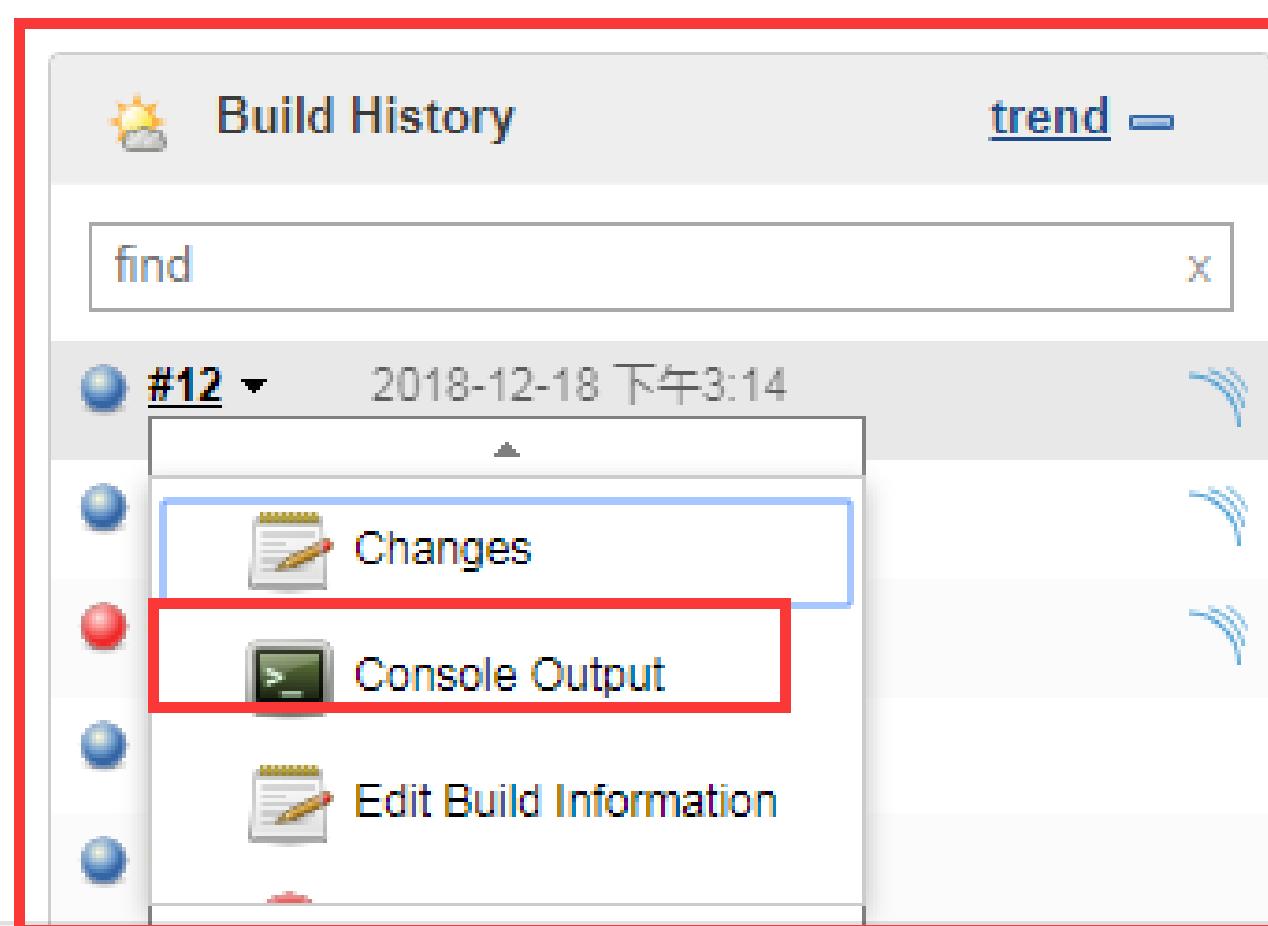
Jenkins初使用

3 选中刚刚新建的项目，点击右侧立即构建选项



这个时候jenkins会去检出代码，检出完成之后会调用maven构建

右下角可以查看执行的历史，点击小箭头可以查看执行日志

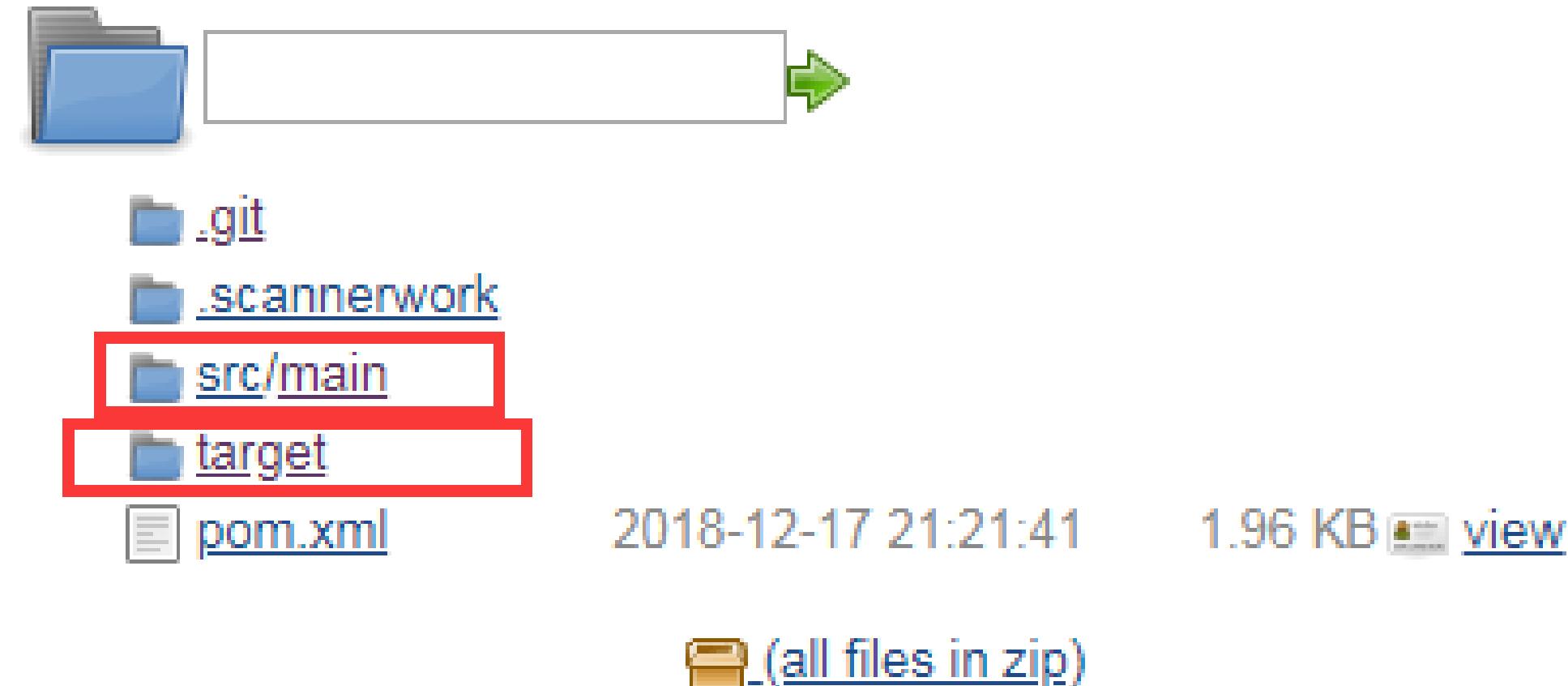


Jenkins初使用

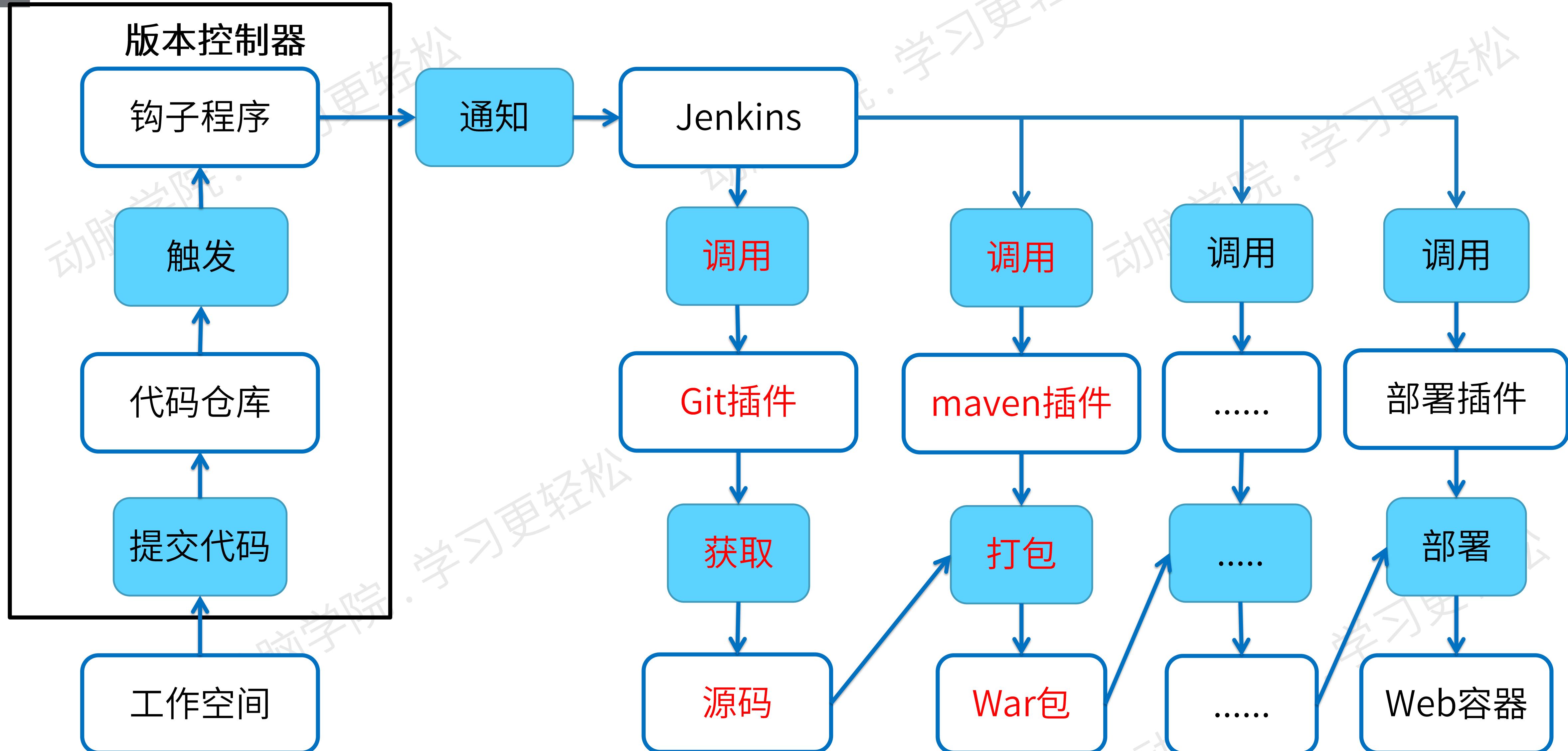
4 选中刚刚新建的项目，点击右侧workspace查看是否有源码和编译的war包

- [Back to Dashboard](#)
- [Status](#)
- [Changes](#)
- [Workspace](#)
 - [Wipe Out Current Workspace](#)
- [立即构建](#)
- [Delete 工程](#)
- [Configure](#)
- [SonarQube](#)
- [Rename](#)

节点 master 上的工作空间 controller



Jenkins初使用总结



Jenkins集成sonar

1 配置sonarqube服务器地址和token，回到jenkins主页，点击系统设置-系统设置

SonarQube servers

Environment variables Enable injection of SonarQube server configuration as build environment variables
If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

SonarQube installations

Name	MySonar
Server URL	http://192.168.1.149:9000/sonarQube Default is http://localhost:9000
Server authentication token SonarQube authentication token. Mandatory when anonymous access is disabled.

[Advanced...](#)

[Delete SonarQube](#)

sonarQube服务器地址

Token生成

- 1 登录sonar，点击我的帐号
- 2 点击安全，输入框随便输入，然后点击生产token，将生成的token赋值到此处

Jenkins集成sonar

2 全局工具配置—sonarScanner,指定sonarScanner安装路径，不需要配置到bin，配置到上级目录就行，将自动安装去掉

SonarQube Scanner

SonarQube Scanner installations

Add SonarQube Scanner

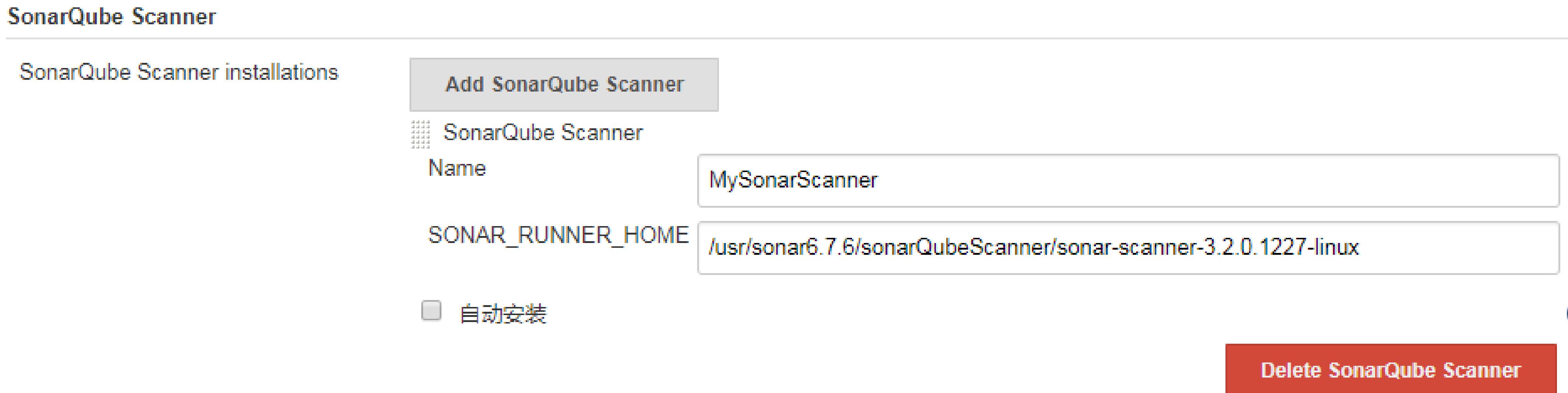
SonarQube Scanner

Name

SONAR_RUNNER_HOME

自动安装

? Delete SonarQube Scanner

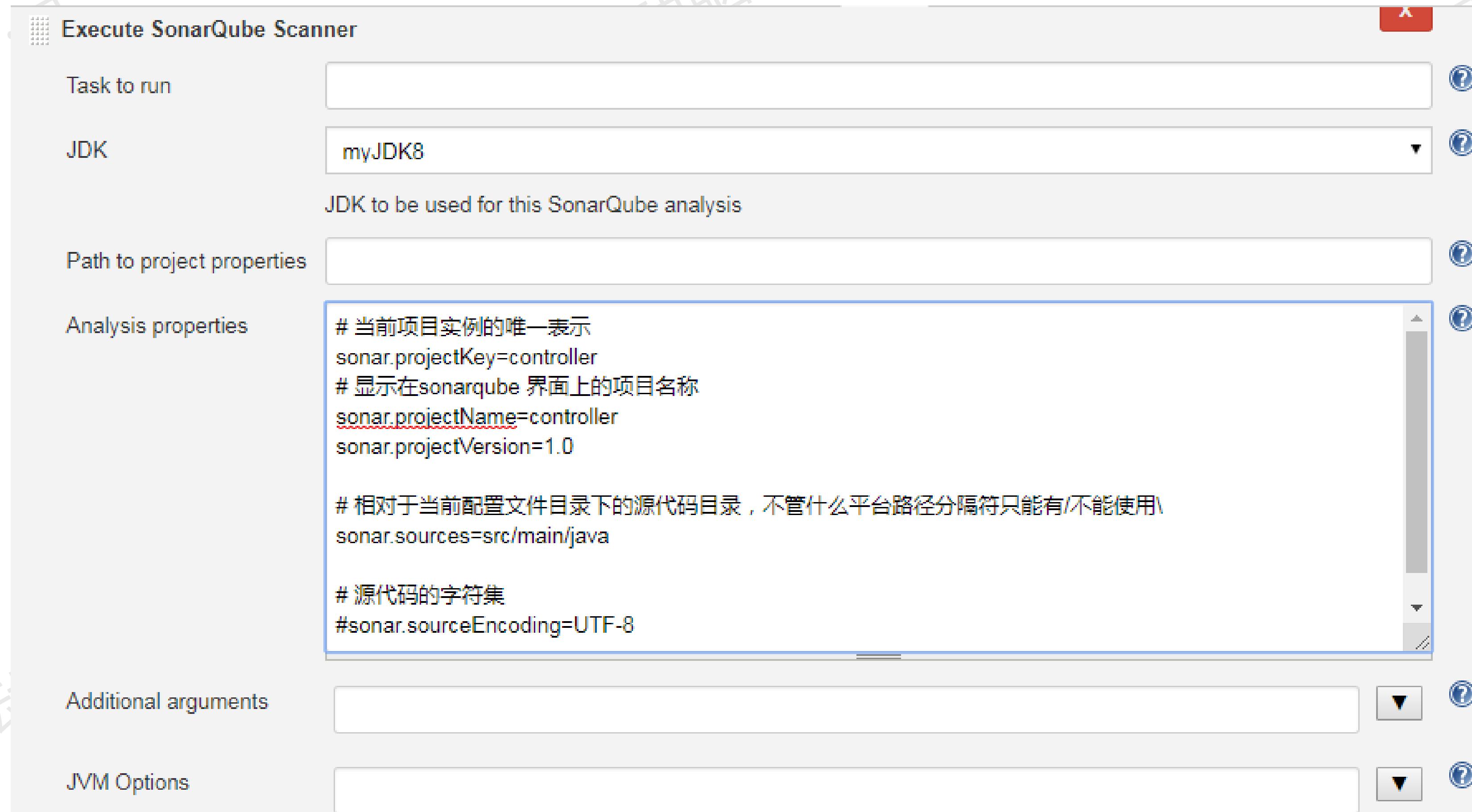


Jenkins集成sonar

3 选中需要检测的项目，点击右侧的configure配置，构建项中选择配置sonarScanner

a 拉框中选中Execute SonarQube Scanner

b 配置sonarScanner的检测信息



Jenkins集成sonar使用

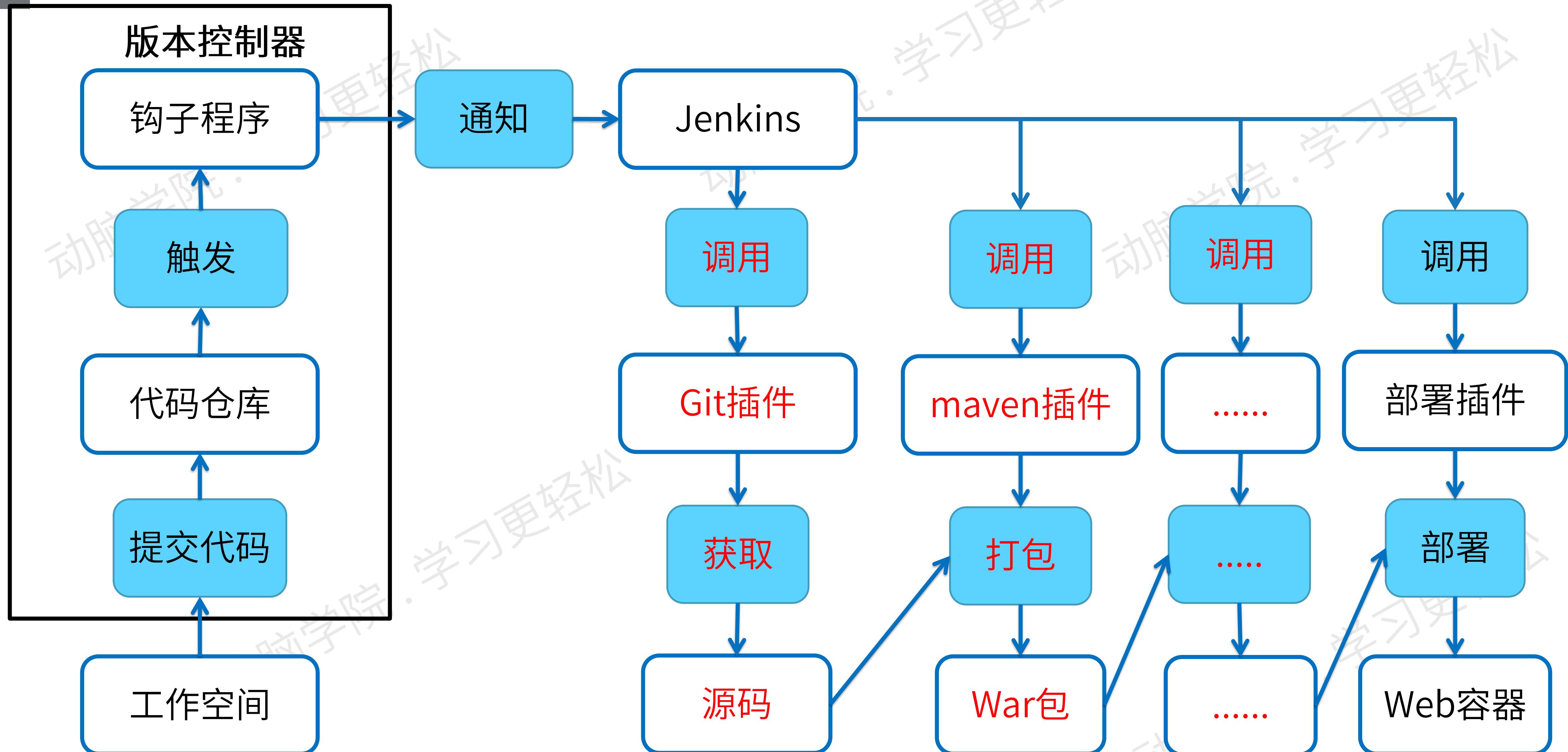
4 选中刚刚新建的项目，点击右侧立即构建选项，成功之后去sonar查看检测结果

The screenshot shows the SonarQube web interface. At the top, there is a navigation bar with tabs: sonarqube (selected), 项目 (selected), 问题, 代码规则, 质量配置, 质量阀, and 配置. To the right of the navigation bar are search fields for '搜索项目, 子项目和文件' and user information. Below the navigation bar, there are buttons for '我的收藏' and '所有'. The main content area displays the analysis results for the 'controller' project, which is currently in '正常' (Normal) status. The results are summarized in a grid:

项	数	等级	详细
Bugs	0	A	
漏洞	1	E	
坏味道	0	A	
覆盖率	0.0%		
重复	0.0%		

On the right side of the results, it says '最近一次分析: 2018年12月18日 下午3:15' and shows '17 XS Java'. On the left side, there is a sidebar with sections for '过滤器' and '质量阀', which includes buttons for '正常' (2), '警告' (0), and '错误' (1).

Jenkins集成sonar总结



Jenkins集成部署插件

在这之前，我们需要设置tomcat8的管理帐号
配置tomcat/config下tomcat-users.xml文件

```
<role rolename="manager"/>
<role rolename="manager-gui"/>
<role rolename="admin"/>
<role rolename="admin-gui"/>
<role rolename="manager-script"/>
<role rolename="manager-jmx"/>
<role rolename="manager-status"/>
<user username="tomcatAdmin" password="5462837zhu" roles="admin-gui,admin,manager-gui,manager,manager-script,manager-jmx,manager-status"/>
```

进入tomcat管理页面，能进入表示配置成功



Tomcat Web Application Manager

Message:	OK
----------	----

Manager				
List Applications	HTML Manager Help	Manager Help	Server Status	

Applications					
Path	Version	Display Name	Running	Sessions	Commands
/	<i>None specified</i>	Welcome to Tomcat	true	0	Start Stop Reload Undeploy
/controller	<i>None specified</i>		true	0	Start Stop Reload Undeploy

Jenkins集成部署插件

1 首先需要下载Deploy to container Plugin插件

2 选中部署的项目，点击右侧的configure配置，配置Post-build Actions构建后的操作

a 拉框中选中Deploy war/ear to a container

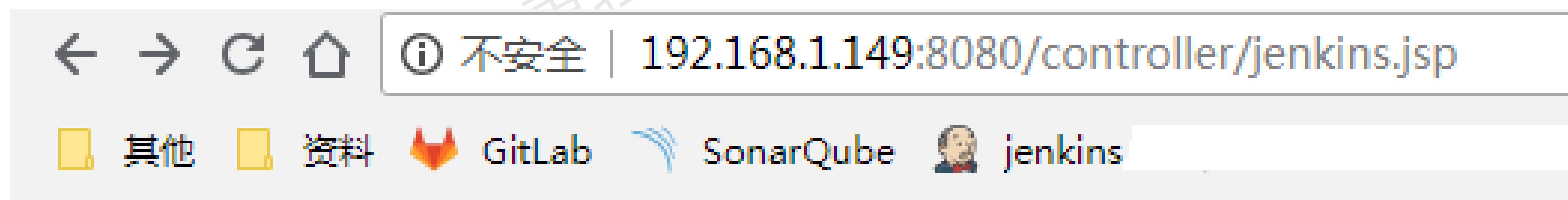
b 配置部署信息



Jenkins集成部署插件使用

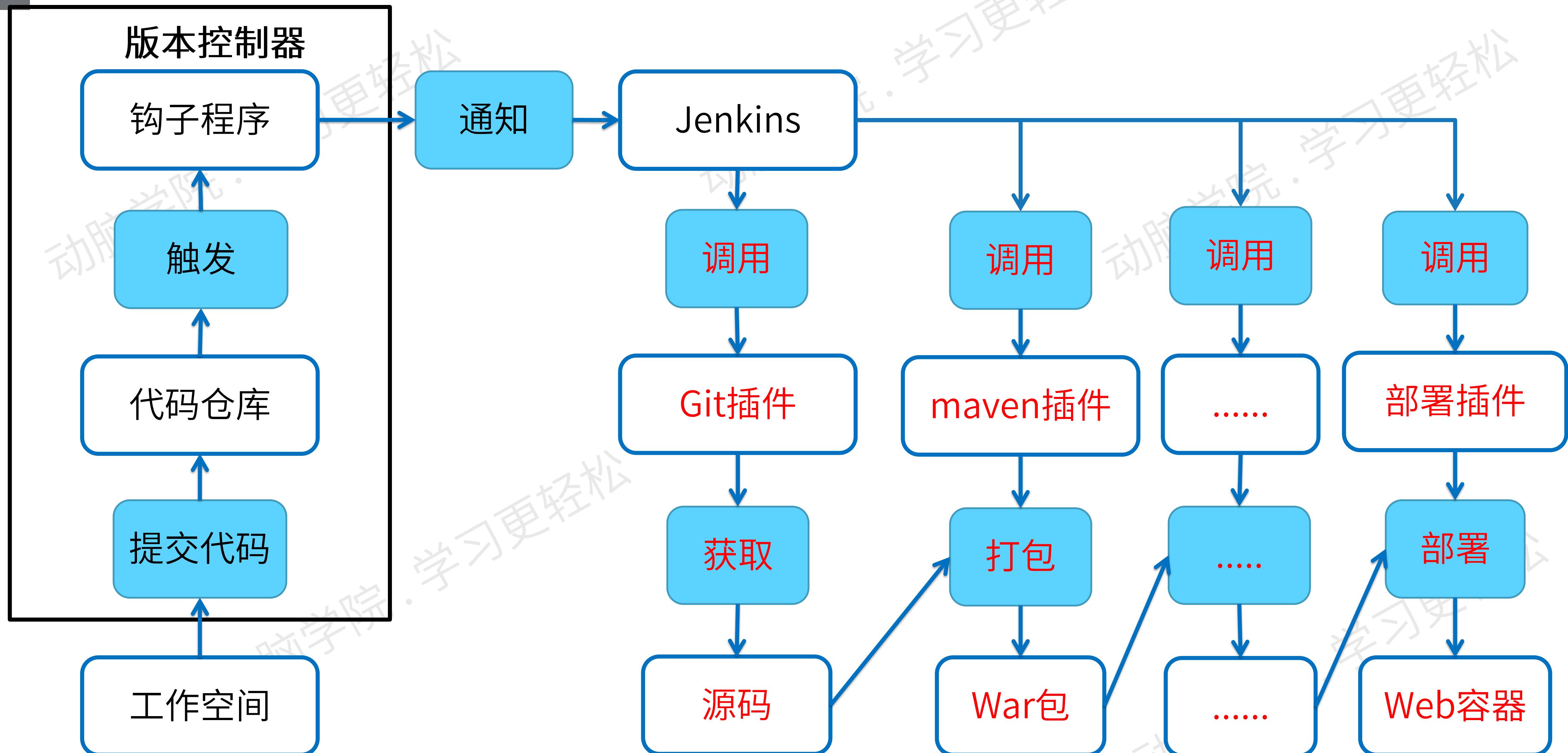
3 选中刚刚新建的项目，点击右侧立即构建选项，查看服务器tomcat/webapps目录，是否已经加载，然后浏览器输入http://192.168.1.149:8080/controller/jenkins.jsp，是否能够访问

名字	大小	已改变	权限	拥有者
		2018/12/13 12:40:38	rwxr-xr-x	root
ROOT		2018/12/13 12:40:38	rwxr-x---	root
manager		2018/12/13 12:40:39	rwxr-x---	root
jenkins		2018/12/15 15:28:41	rwxr-x---	root
host-manager		2018/12/13 12:40:39	rwxr-x---	root
examples		2018/12/13 12:40:39	rwxr-x---	root
docs		2018/12/13 12:40:39	rwxr-x---	root
controller		2018/12/18 15:15:29	rwxr-x---	root
jenkins.war	74,159 KB	2018/12/13 21:33:31	rw-r--r--	root
controller.war	5,050 KB	2018/12/18 15:15:29	rw-r-----	root



admin_tank_dongnao

Jenkins集成部署插件总结



Jenkins配置钩子程序

选中需要配置的项目，点击右侧的configure配置，选中Build Triggers构建后触发

Build Triggers

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

Authentication Token

jenkins

Use the following URL to trigger build remotely: JENKINS_URL/me/my-views/view/all/job/controller/build?token=TOKEN_NAME or
/buildWithParameters?token=TOKEN_NAME
Optionally append &cause=Cause+Text to provide text that will be included in the recorded build cause.

- Build when a change is pushed to GitLab. GitLab webhook URL: http://192.168.1.149:8080/jenkins/project/controller
- GitHub hook trigger for GITScm polling
- 其他工程构建后触发
- 定时构建
- 轮询 SCM

1 填写密钥

2 gitlab或者github中配置HOOKS:

http://192.168.1.149:8080/jenkins/job/controller/build?token=jenkins

The screenshot shows the GitLab Admin Area with the 'System Hooks' page open. The URL in the address bar is `http://192.168.1.149:8080/jenkins/job/controller/build?token=jenkins`. The page title is 'Edit System Hook'. A note states: 'System hooks can be used for binding events when GitLab creates a User or Project.' There are two input fields: 'URL' containing the Jenkins trigger URL and 'Secret Token' which is currently empty.

Jenkins配置钩子程序使用

我们先在浏览器中输入：

<http://192.168.1.149:8080/jenkins/job/controller/build?token=jenkins>, 查看jenkins中的效果，如果成功在浏览器中访问项目，是否是我们需要的效果

接下来我们修改一下代码，使用git的push命令或者eclipse执行push看下jenkins是否有效果

Jenkins总结

