

# 持续集成工具 Jenkins

## 1 从装修厨房看项目开发效率优化

### 1.1 持续部署

#### ➤ 装修厨房

全部装好之后发现灯不亮，电路有问题；冷热水装反了，管路有问题。这些问题要解决就必须把地砖、墙砖拆掉——**一个环节有问题，其他环节跟着返工。**

那怎么做会好一些呢？

任何安装完成及时测试，确保其可以正常工作。

#### ➤ 项目开发

开发过程中进行单元测试能够通过，但是部署到服务器上运行出现问题。

那怎么做会好一些呢？

仅仅单元测试还不够，各个模块都必须能够在服务器上运行。

#### ➤ 关注点

持续部署的关注点在于项目功能**部署至服务器后可以运行**，为下一步测试环节或最终用户正式使用做好准备。

### 1.2 持续集成

#### ➤ 装修厨房

装修厨房时我们需要铺地砖，如果把所有地砖都切好再拿去铺就会发现：每一块地砖**单独看都是好的**，但是实际铺的时候，把**所有地砖整合起来**，发现和厨房地面总体尺寸**不匹配**，边边角角的地砖需要重新切，时间和物料成本陡然升高。

那怎么做会好一些呢？

切一块铺一块，根据需要的尺寸来切，尽早发现尺寸变化，避免返工。

#### ➤ 项目开发

各个小组分别负责各个具体模块开发，本模块**独立测试**虽然**能够通过**，但是上线前夕将所有模块整合到一起**集成测试**却发现**很多问题**，想要解决就需要把很多代码返工重写而且仍然有可能有问题，但现在时间很可能不够了。

那怎么做会好一些呢？

经常性、频繁的把所有模块集成在一起进行测试，有问题尽早发现，这就是持续集成。

#### ➤ 关注点

持续集成的关注点在于尽早发现项目整体运行问题，尽早解决。

### 1.3 持续交付

#### ➤ 装修厨房

全部装修好之后房屋主人来验收，各项功能都正常，但是水龙头的样式主人**不喜欢**，灶台的位置主人**不满意**，要求返工。

那怎么做会好一些呢？

房屋主人随时查看装修进度，施工团队及时调整。

➤ 项目开发

项目的各个升级版本之间间隔时间太长，对用户反馈感知迟钝，无法精确改善用户体验，用户流失严重。

那怎么做会好一些呢？

用**小版本**不断进行**快速迭代**，不断收集用户反馈信息，用最快的速度改进优化。

➤ 关注点

持续交付的关注点在于研发团队的最新代码能够尽快让最终用户体验到。

## 1.4 总体目标

➤ **好处 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。

➤ **好处 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把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。

➤ **好处 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。

➤ **好处 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：

(1)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。

(2)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。

➤ **好处 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。

## 2 持续集成工具

### 2.1 Jenkins 和 Hudson

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 年年初，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。

### 2.2 技术组合

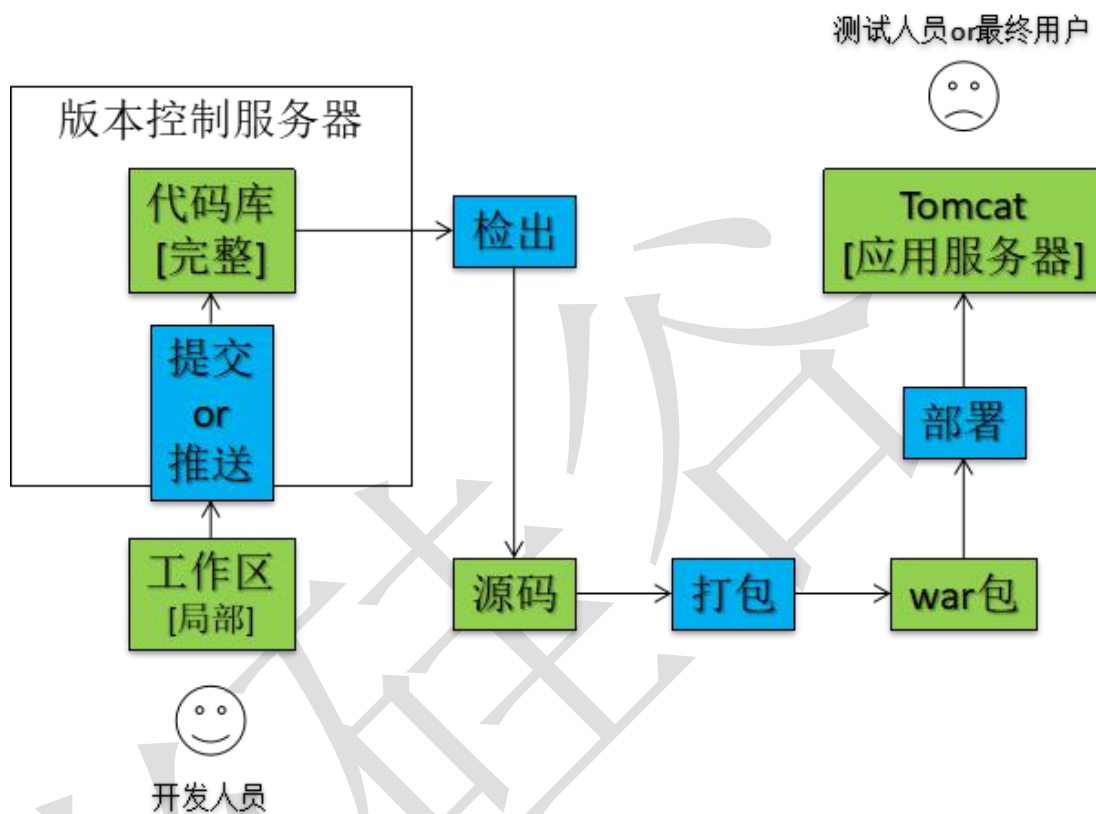
Jenkins 可以整合 GitHub 或 Subversion

Hudson 也可以整合 GitHub 或 Subversion

二者既然是同源的工具软件，操作和指导思想就是接近的，所以本教程通过 Jenkins 为大家呈现。

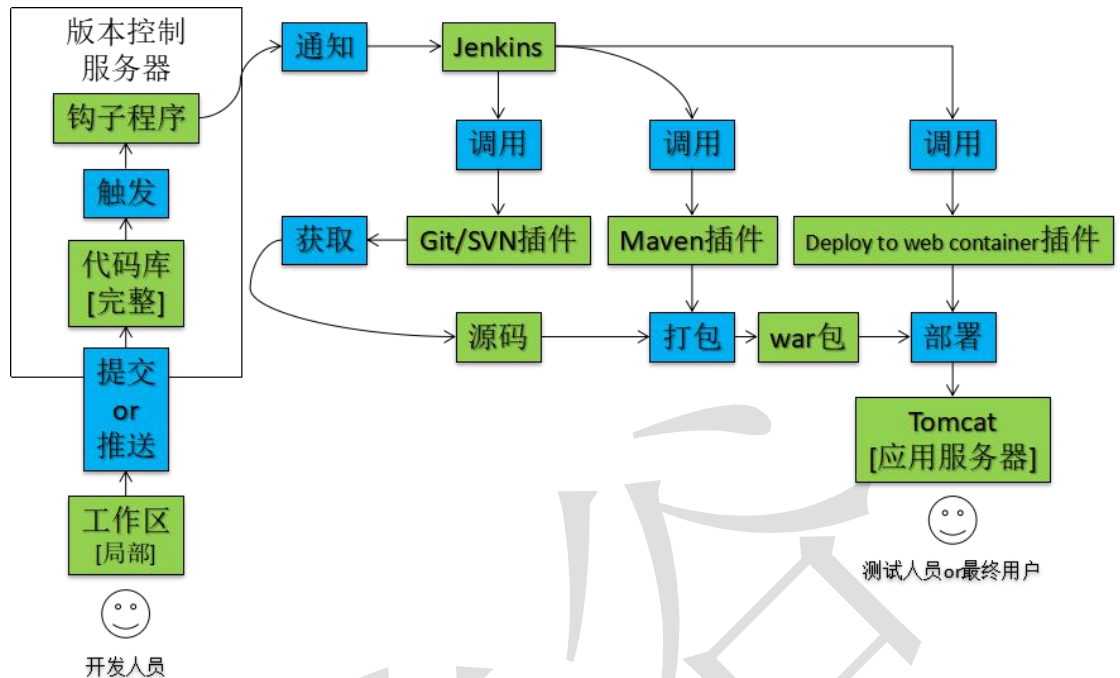
## 3 JavaEE 项目部署方式对比

### 3.1 手动部署



### 3.2 自动化部署

“自动化”的具体体现：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——一切都发生在无形中。

下面我们带领大家一步一步搭建整套持续集成环境，这个操作过程只需要细心认真即可，没有任何难度。但是需要你具备以下前置知识：

- ✓ Linux 基本操作命令和 VIM 编辑器使用
- ✓ Maven 的项目构建管理
- ✓ GitHub 或 SVN 使用

## 4 Jenkins+SVN 持续集成环境搭建

### 4.1 系统结构总述

- ❖ 创建虚拟机安装 Linux 系统
- ❖ 版本控制子系统
  - Subversion 服务器
  - 项目对应版本库
  - 版本库中钩子程序
- ❖ 持续集成子系统
  - JDK
  - Tomcat
  - Maven
  - Jenkins
  - ◆ 主体程序

- ◆ SVN 插件
- ◆ Maven 插件
- ◆ Deploy to Web Container 插件
- ❖ 应用发布子系统
  - JDK
  - Tomcat

## 4.2 版本控制子系统

详细过程我们就省略了，这里记录一下版本库的访问账号密码

```
[ users]
# harry = harryssecret
# sally = sallysecret
subman = 123123
```

※特别提示：svnserve.conf 文件中 anon-access 一定要打开注释并设置为 none

```
8 [ general]
9 ### These options control
10 ### and authenticated users
11 ### and "none". The sample
12 anon-access = none
13 auth-access = write
14 ### The password-db option
```

## 4.3 应用发布子系统

详细过程同样省略，仅记录 Tomcat 服务器的账号密码

配置文件位置：/opt/tomcat/conf/tomcat-users.xml

```
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="manager-jmx"/>
<role rolename="manager-status"/>
<user
    username="tomcat_user"
    password="123456"
    roles="manager-gui,manager-script,manager-jmx,manager-status" />
```

## 4.4 Jenkins 主体程序安装配置

- 把 jenkins.war 放在 Tomcat 解压目录/webapps 目录下
- 打开 Tomcat 解压目录/server.xml 修改 URL 地址的编码解码字符集

```
vim /opt/tomcat/conf/server.xml
71 <Connector port="8080" protocol="HTTP/1.1"
72         connectionTimeout="20000"
73         redirectPort="8443" URIEncoding="UTF-8"/>
```

- 启动 Tomcat 并通过浏览器访问
  - 网址示例：http://192.168.70.131:8080/jenkins

## ➤ 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

[继续](#)

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

```
[root@rich ~]# cat /root/.jenkins/secrets/initialAdminPassword
2f6bff33bda14baba83ba1c002045f05
```

这里填入的密文同时也是 admin 账号的密码。

## ➤ 选择插件安装方式



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 2.107.3

选择哪种方式都不会对后续操作有太大影响。因为有需要的插件我们可以在后续有针对性的安装。

本教程在这里选择“安装推荐的插件”。

安装过程如下：



新手入门

# 新手入门

✖ Folders	✖ OWASP Markup Formatter	✖ Build Timeout	✖ Credentials Binding	** Script Security ** Command Agent Launcher
✖ Timestampers	✖ Workspace Cleanup	✖ Ant	✖ Gradle	Folders
✖ Pipeline	✖ GitHub Branch Source	✖ Pipeline: GitHub Groovy Libraries	✖ Pipeline: Stage View	
✖ Git	✖ Subversion	✖ SSH Slaves	✖ Matrix Authorization Strategy	
✖ PAM Authentication	✖ LDAP	✖ Email Extension	✖ Mailer	
				** - 需要依赖

Jenkins 2.107.3

打×的插件是由于网络传输导致的安装失败，后面再重新安装即可。

※注意：这个步骤中如果选择了安装插件则 Linux 必须能够联网。

➤ 新建账号或以管理员身份继续

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 2.107.3

[使用admin账户继续](#)[保存并完成](#)

可以选择使用 admin 账户继续，后面有需要仍然有机会注册新账户。

➤ 开始使用 Jenkins

## Jenkins已就绪!

你已跳过创建admin用户的步骤。要登录请使用用户名: 'admin' 及用于访问安装向导的管理员密码。

jenkins安装已完成。

[开始使用jenkins](#)

### 4.5 系统初始化配置

➤ 系统管理界面



➤ 全局安全配置

## 全局安全配置

☒ 启用安全

Disable remember me ☐

访问控制

安全域

☒ Jenkins专有用户数据库

☒ 允许用户注册

☐ LDAP

☐ Servlet容器代理

☐ Unix用户/组数据库

授权策略

☒ 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

☐ 遗留模式

➤ 全局工具配置：Maven Configuration

## Maven Configuration

Default settings provider: Use default maven settings

Default global settings provider: Settings file in filesystem

File path: /opt/apache-maven-3.5.0/conf/settings.xml

Default global settings provider: Global settings file on filesystem

File path: /opt/apache-maven-3.5.0/conf/settings.xml

## ➤ 全局工具配置: JDK

JDK 安装

新增 JDK

系统下JDK 安装列表

## JDK

## JDK 安装

JDK 别名: MyJDK

JAVA\_HOME: /opt/jdk1.8.0\_121

☒ 自动安装

新增 JDK

系统下JDK 安装列表

删除 JDK

## ➤ 全局工具配置: Maven



Diagram showing the Maven configuration form. The 'Maven 安装' (Maven Installation) tab is selected. The form includes fields for 'Name' (MyMaven) and 'MAVEN\_HOME' (/opt/apache-maven-3.5.0). There is also a checkbox labeled '自动安装' (Automatic Installation).

- 全局工具配置：Git[若有]  
不使用 Git，所以删除即可。

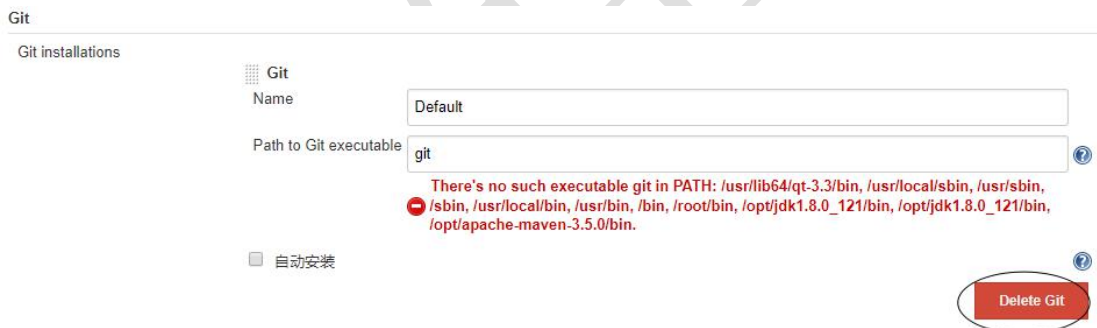


Diagram showing the Git configuration form. The 'Git' section is visible, with fields for 'Name' (Default) and 'Path to Git executable' (git). A red error message is displayed: 'There's no such executable git in PATH: /usr/lib64/qt-3.3/bin, /usr/local/sbin, /usr/sbin, /sbin, /usr/local/bin, /usr/bin, /bin, /root/bin, /opt/jdk1.8.0\_121/bin, /opt/jdk1.8.0\_121/bin, /opt/apache-maven-3.5.0/bin.' A 'Delete Git' button is highlighted with a red oval.

最后点击 save 保存退出



Diagram showing the 'Save' and 'Apply' buttons. The 'Save' button is highlighted with a blue oval.

## 4.6 安装插件



安装插件时受到网络状况的影响有可能会失败，不要紧，多试几次，直到成功。

## 4.7 创建工程

### 4.7.1 创建工程



#### 4.7.2 指定工程名称和工程类型



输入一个任务名称

ProOne

» 必填项

**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。

**GitHub Organization**  
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

**Multibranch Pipeline**  
Creates a set of Pipeline projects according to detected branches in one SCM repository.

**确定**

可以嵌套存储的容器。利用它可以进行分组。视图仅仅是一个过滤器，而文件夹则是一个独立的命名空间，因此你可以有多个相同名称的文件夹。只要它们在不同的文件夹里即可。

#### 4.7.3 源码管理



**源码管理**

☐ None

☐ Git

☒ Subversion

Repository URL

svn://192.168.70.130/pro.one/ProOne

这里指定的URL地址必须恰好定位到pom.xml文件的上一级  
因为Jenkins就是到项目根目录下查找pom.xml

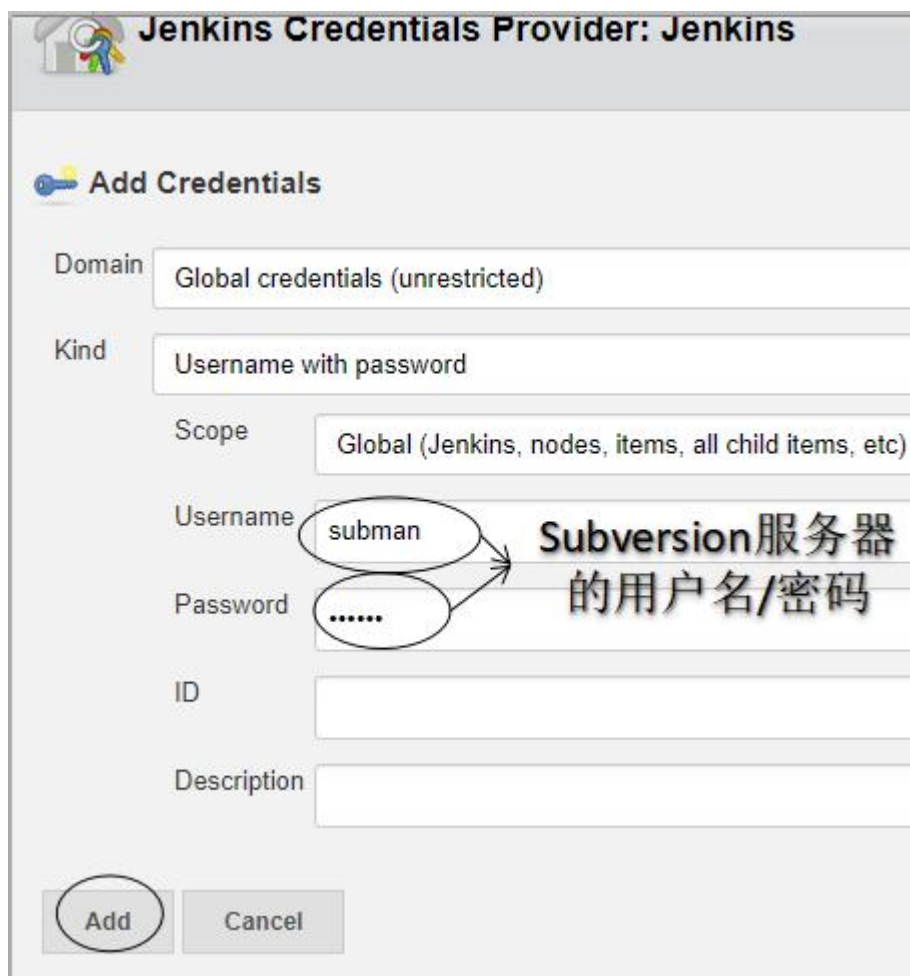
Credentials

- none -

Add

Jenkins





**Jenkins Credentials Provider: Jenkins**

**Add Credentials**

Domain: Global credentials (unrestricted)

Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: subman

Password: .....

ID:

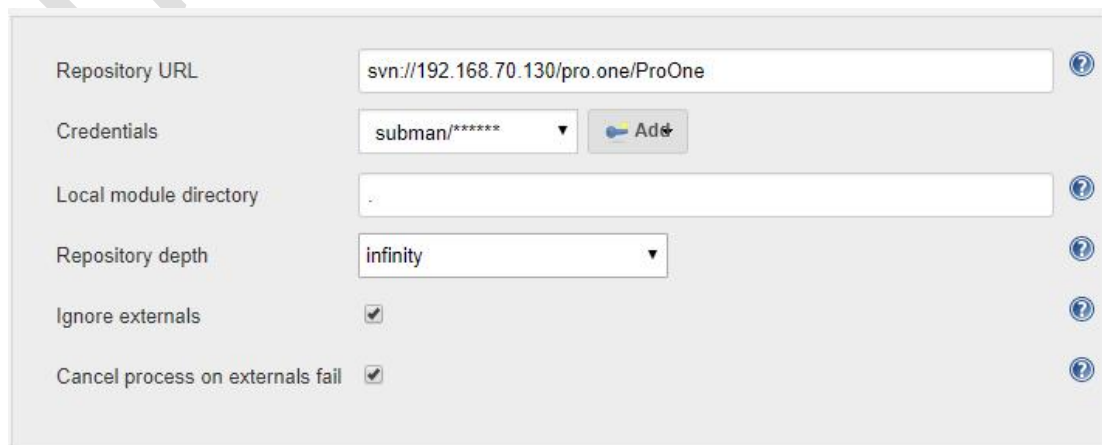
Description:

**Add** **Cancel**

**Subversion服务器的用户名/密码**

- none -  
- none -  
subman/\*\*\*\*\* 这里一定要再选一下！

配好的效果：



Repository URL: svn://192.168.70.130/pro.one/ProOne

Credentials: subman/\*\*\*\*\* **Add**

Local module directory: ✓

Repository depth: infinity

Ignore externals: ☒

Cancel process on externals fail: ☒

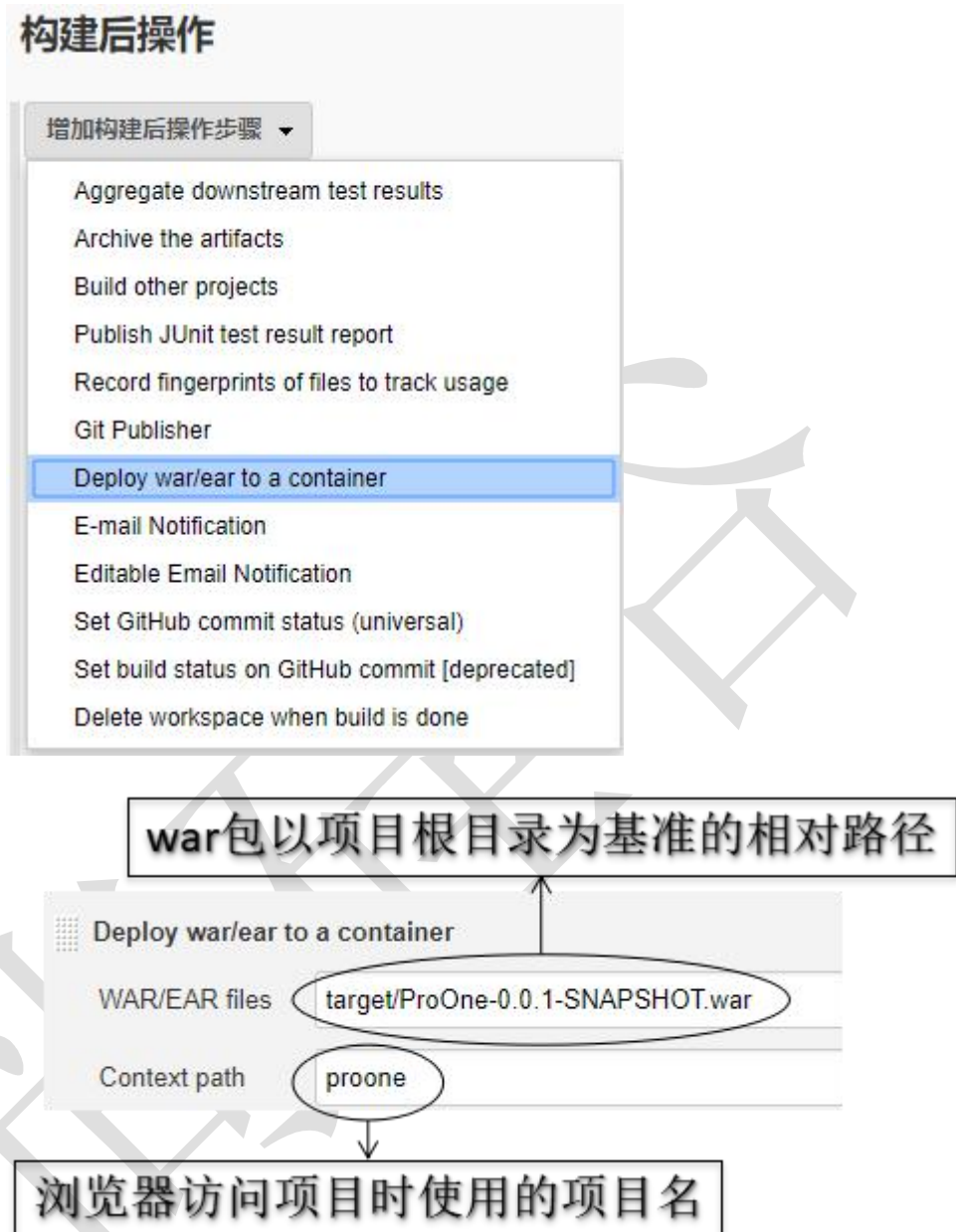
※注意：此时 Jenkins 的工作区中还没有代码，需要执行一次构建操作之后

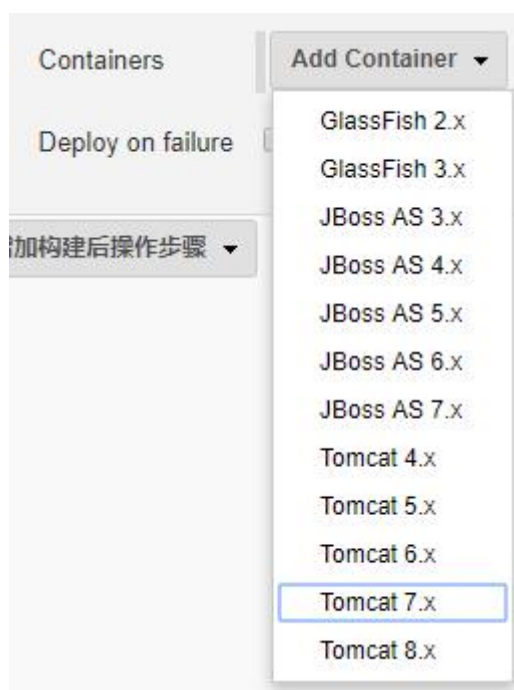
Jenkins 才会下载代码。

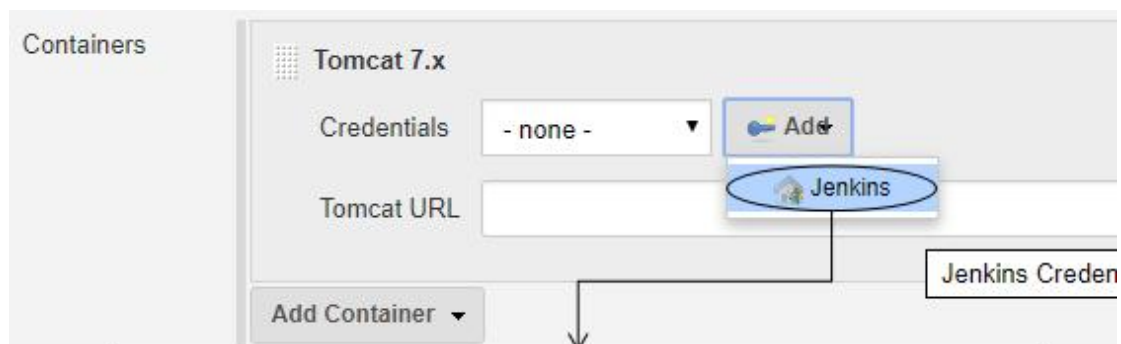
#### 4.7.4 构建



## 4.7.5 构建后操作







### Add Credentials

Domain Global credentials (unrestricted)

Kind Username with password

Scope Global (Jenkins, nodes, items, all child items, etc)

Username tomcat\_user

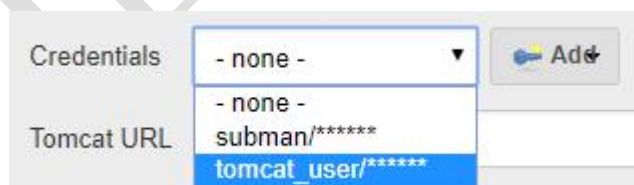
Password .....

ID

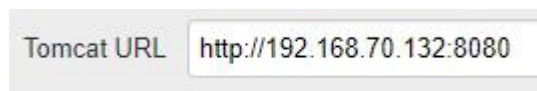
Description

Add

Cancel



这里同样一定要选一下！



#### 4.7.6 手动构建



Jenkins 使用天气状况来表示构建成功率



项目健康度高于80%。你可以将鼠标放在项目图标上以获取更多信息。



项目健康度介于60%~80%。你可以将鼠标放在项目图标上以获取更多信息。



项目健康度介于40%~60%。你可以将鼠标放在项目图标上以获取更多信息。



项目健康度介于20%~40%。你可以将鼠标放在项目图标上以获取更多信息。



项目健康度低于20%。你可以将鼠标放在项目图标上以获取更多信息。

#### 4.7.7 构建触发器



远程触发的基本原理是 SVN 服务器给 Jenkins 项目特定的 URL 地址发送请求，但必须以请求参数的形式携带一个特定值，这个特定值就是这里的“身份验证令牌”。

比如我们这个项目的地址触发地址是：

`http://192.168.70.131:8080/jenkins/job/ProOne/build`

身份验证令牌是：

ATGUGU\_TOKEN

那么最终的访问地址就是：

`http://192.168.70.131:8080/jenkins/job/ProOne/build?token=ATGUGU_TOKEN`

EN

触发访问地址中 Jenkins 访问地址是根据实际情况改变的，项目名称根据实际情况改变，其他都不变。



#### 4.8 获取 crumb 值





#### API Token

User ID	admin
API Token	090d592c760bf922d554ed4b1abb2137

携带 API Token 访问下面地址：

http://admin:090d592c760bf922d554ed4b1abb2137@192.168.70.131:8080/jenkins/crumblssuer/api/xml

```
<defaultCrumbIssuer _class="hudson.security.csrf.DefaultCrumbIssuer">
  <crumb>5dbccaf47a86bf5b675456f58855fe16</crumb>
  <crumbRequestField>Jenkins-Crumb</crumbRequestField>
</defaultCrumbIssuer>
```

则触发 Jenkins 远程构建时需要携带的请求消息头就是：

Jenkins-Crumb:5dbccaf47a86bf5b675456f58855fe16

## 4.9 Linux 的 curl 命令

Linux 的 curl 命令用来发送 HTTP 请求。

- X 参数：指定请求方式
- v 参数：显示响应结果
- u 参数：携带用户名/密码
- H 参数：携带请求消息头信息

```
curl -X post -v -u [Jenkins 用户名]:[Jenkins 密码] -H "请求消息头信息" http://[服务器 IP 地址]:[服务器端口号]/jenkins/job/[Jenkins 项目名称]/build?token=[身份验证令牌]
```

```
curl      -X      post      -v      -u      admin:2f6bff33bda14baba83ba1c002045f05      -H
"Jenkins-Crumb:88a12946e07d82b3b0d567c7c4610c9a"
http://192.168.70.131:8080/jenkins/job/ProOne/build?token=ATGUIGU_TOKEN
```

## 4.10 编辑 SVN 版本库中的钩子程序

- 钩子程序由 post-commit.tmpl 复制得到
  - 这里注意不要使用任何扩展名。如果按照我们习惯的使用.sh 扩展名则钩子程序无法正常工作。
  - 记得使用 chmod 命令设置为可执行权限
- 把原有内容注释，加入 curl 命令

```
# REPOS="$1"
# REV="$2"
curl -X post -v -u admin:2f6bff33bda14baba83ba1c002045f05 -H "Jenkins-Crumb: 88a1
2946e07d82b3b0d567c7c4610c9a" http://192.168.70.131:8080/jenkins/job/ProOne/buil
d?token=ATGUIGU_TOKEN
# mailer.py commit "$REPOS" "$REV" /path/to/mailer.conf
"post-commit" 50L, 2164C
```

40, 1

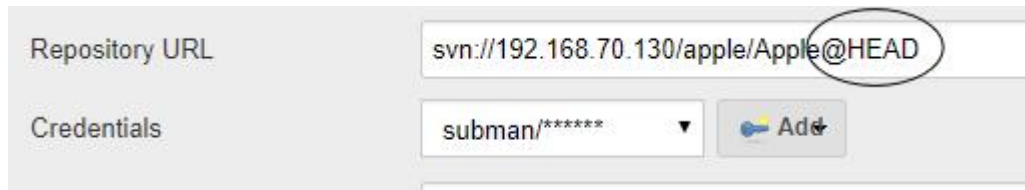
底端

#### 4.11 测试

在 Eclipse 中提交修改，在页面上查看显示内容变化

#### 4.12 补充

如果发生 Jenkins 服务器从 SVN 服务器下载代码不是最新版的情况，那么就在 SVN 服务器的 URL 地址后面加上@HEAD 强制要求下载最新版。



## 5 Jenkins+GitHub 持续集成环境搭建

### 5.1 要点

Jenkins 与 GitHub 配合实现持续集成需要注意以下几点：

- 第一：Jenkins 要部署到外网上，因为内网地址 GitHub 是无法访问到的。这一点可以通过租用阿里云等平台提供的云服务器实现。
- 第二：Jenkins 所在的主机上需要安装 Git，通过 Git 程序从 GitHub 上 clone 代码。
- 第三：在 Jenkins 内需要指定 Git 程序位置，和指定 JDK、Maven 程序位置非常类似。
- 第四：在 GitHub 上使用每个 repository 的 WebHook 方式远程触发 Jenkins 构建。
- 第五：在 Jenkins 内关闭“防止跨站点请求伪造”

### 5.2 Linux 环境下安装 Git

- 第一步：安装编译 git 时需要的包  

```
yum install -y curl-devel expat-devel gettext-devel openssl-devel zlib-devel  
yum install -y gcc perl-ExtUtils-MakeMaker
```
- 第二步：删除已有的 git  

```
yum remove git
```
- 第三步：Git 官网下载 Git 最新版 tar 包，移动到/usr/src 目录下  

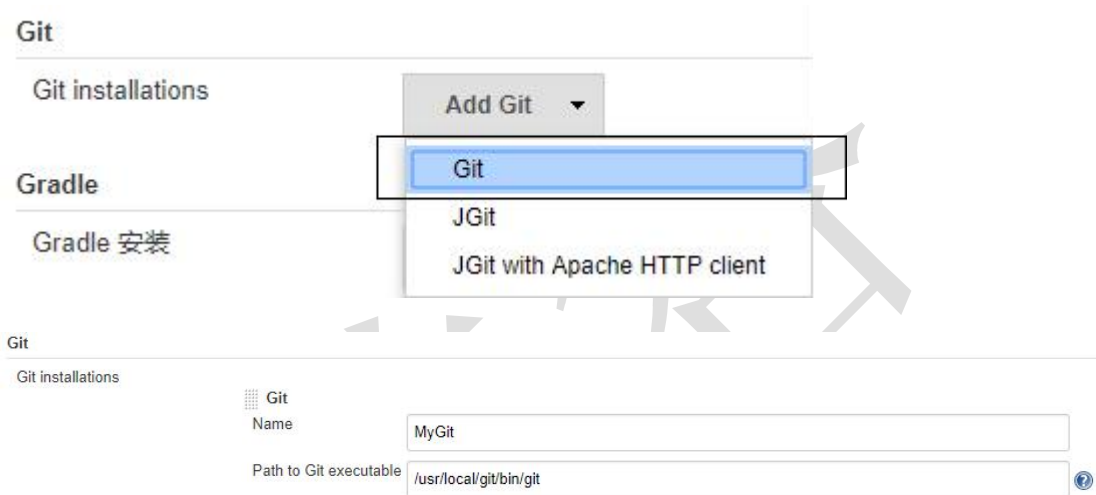
```
cd /usr/src  
tar -zxvf git-2.9.3.tar.gz
```
- 第四步：编译安装  

```
cd git-2.9.3  
make prefix=/usr/local/git all
```

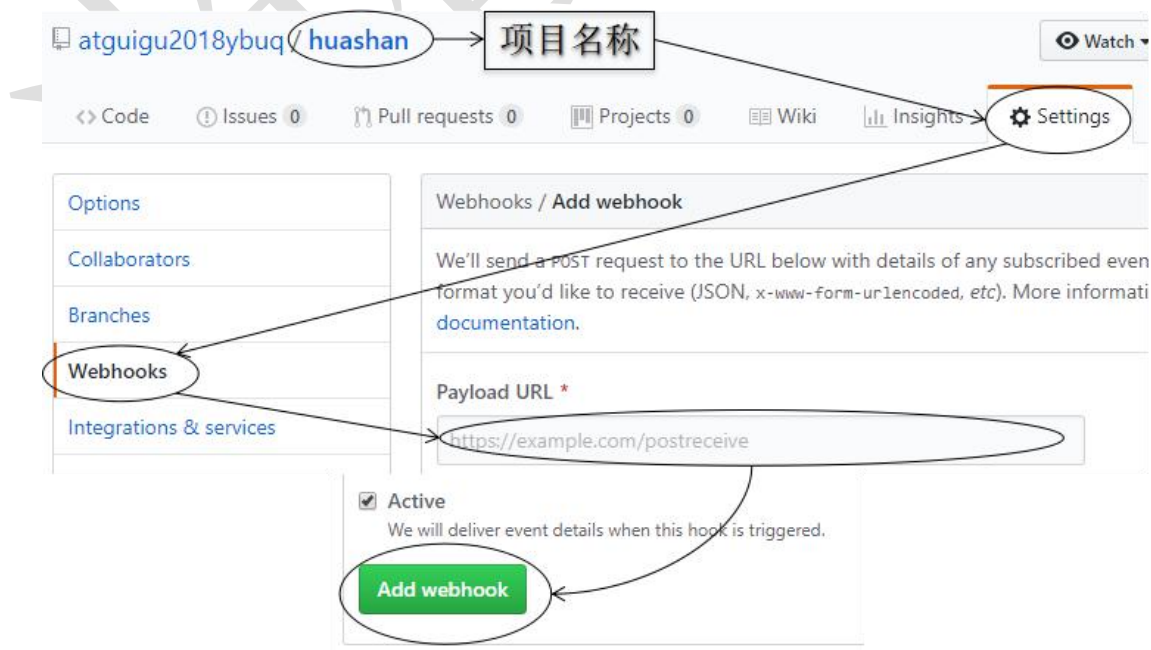
```
make prefix=/usr/local/git install
echo "export PATH=$PATH:/usr/local/git/bin" >> /etc/bashrc
source /etc/bashrc
```

- 第五步：检查一下版本号  
git --version

### 5.3 在 Jenkins 中指定 Git 程序位置



### 5.4 在 GitHub 上添加 WebHook



## 5.5 在 Jenkins 内关闭“防止跨站点请求伪造”



### 全局安全配置

Jenkins安全，定义谁可以访问或使用系统。

#### CSRF Protection

☐ 防止跨站点请求伪造

微信号：creathinFeng