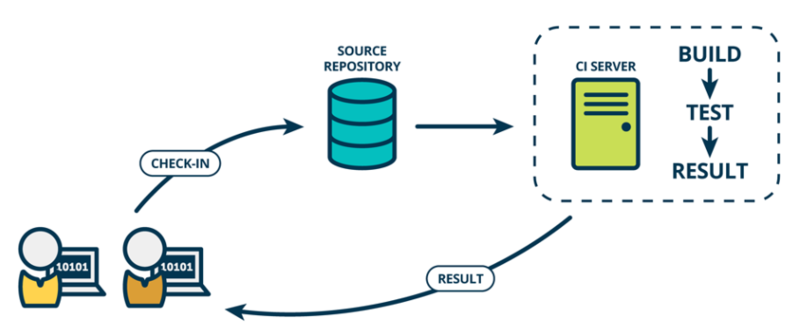
# 1、Jenkins概述与环境搭建

## 1.1 了解Jenkins

持续集成即 Continuous integration（CI） 是一种软件开发实践，即团队开发成员经常集成他们的工作，每次集成都通过自动化的构建（包括编译，发布，自动化测试）来验证，从而尽快地发现集成错误，让团队能够更高效的开发软件。



**持续集成要点**

* 统一的代码库(git)
* 统一的依赖包管理(nexus)
* 测试自动化
* 构建全自动化(maven)
* 部署自动化
* 可追踪的集成记录

jenkins 就是为了满足上述持续集成的要点而设计的一款工具，其主体框架采用JAVA开发，实质内部功能都是由各种插件实现，极大提高了系统的扩展性。其不仅可以满足JAVA系统的集成，也可以实现PHP等语言的集成发布。通过其pipeline 插件，用户可以随自己需要定制集成流程。

## 1.2 安装Jenkins

**下载：**jenkins 支持Docker、yum、msi 等安装，在这里推荐大家直接选择下载他对应的WAR包进行安装。<https://jenkins.io/download/>

**启动：**下载完成之后直接可通过 jar -jar 命令启动

java -jar jenkins.war --httpPort=8080

也可以将其放至到servlet容器（tomcat\jetty\jboss）中直接启动，无需过多的配置，一切插件化这是jenkins 比较优秀的设计。

**配置：**下载完成之后进入启动页(<http://127.0.0.1:8080/>) 会有一个 验证过程，验证码存储在 ${user\_home}\.jenkins\secrets\initialAdminPassword 中，接着就是进入安装插件页，选择默认即可，这个过程稍长。

## 1.3 常用插件下载

Jenkins需要maven与git环境，服务器需要先装好这两个工具

**插件下载**

**如果插件下载比较慢的话可以更换站点**

更换源 ->系统管理->管理插件->高级 ->升级站点

把：<http://updates.jenkins-ci.org/update-center.json>

换成：<http://mirror.esuni.jp/jenkins/updates/update-center.json>

镜像源查询：<http://mirrors.jenkins-ci.org/status.html>

基本插件列表

|  |  |
| --- | --- |
| **插件名称** | **插件描述** |
| **Maven Integration** | **maven 管理插件** |
| **Deploy to container** | **容器部署插件** |
| **Pipeline** | **管道集成插件** |
| **Email Extension** | **邮件通知插件** |
| **SSH** | **用于ssh 通信** |

# 2、Jenkins可持续化集成

## 2.1 持续化集成完成的目标

需要达成的目标：

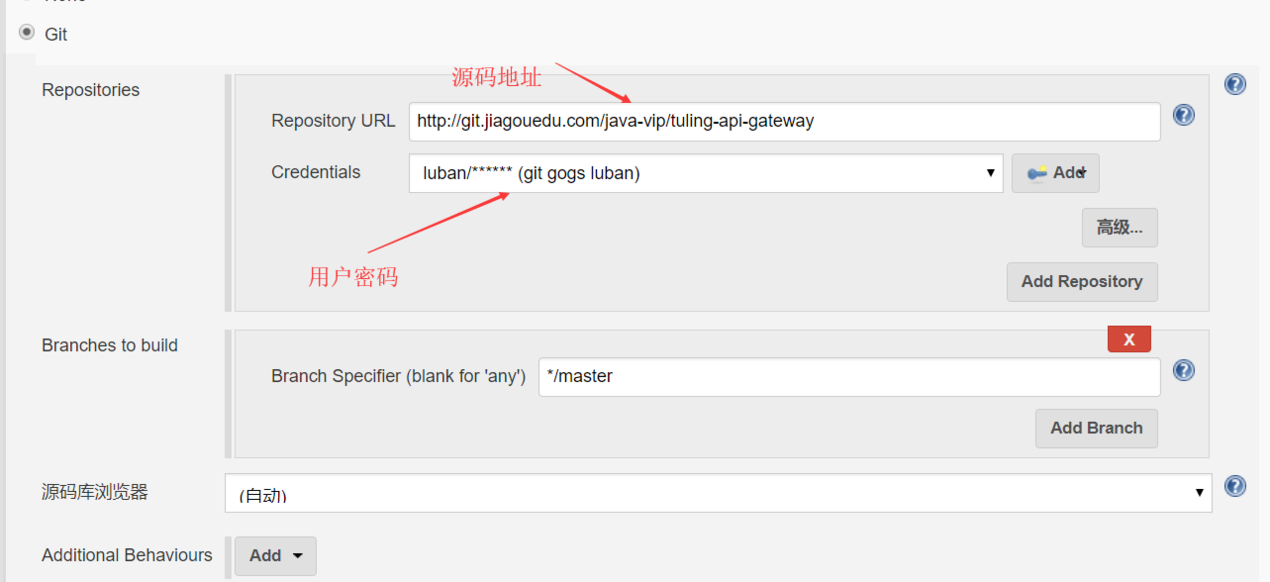
1. 自动基于分支构建项目
2. 构建好的项目自动部署至Tomcat容器（可选）
3. 构建好的项目自动上传至Nexus 私服存档
4. 保存构建历史记录，并可以下载历史记录

## 2.2 持续化集成的配置

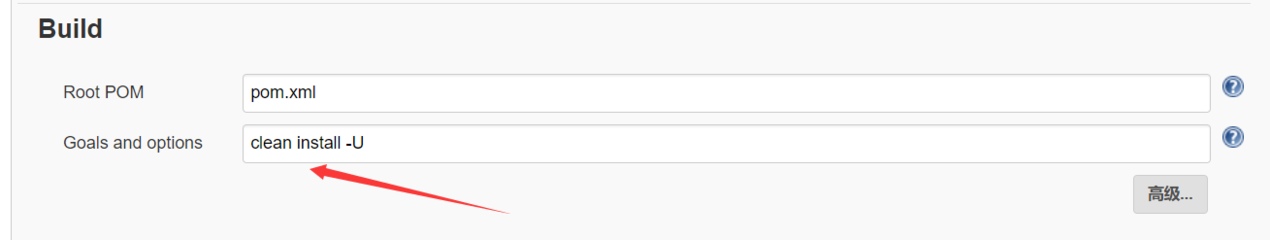
1、新建maven job



2、配置checkout源码



3、编写maven构建命令



4、自动部署至tomcat配置

添加构建后操作：Deploy war/ear to container 项目



自动部署的前提条件：

1. 需要下载 Deploy to container 插件
2. 设置Tomcat manager 用户和密码，以下配置加入至 Tomcat conf/tomcat-users.xml 中
3. tomcat webapp 中必须保留 manager 项目，在user-setting中配置

<role rolename="admin-gui"/>

<role rolename="manager-gui"/>

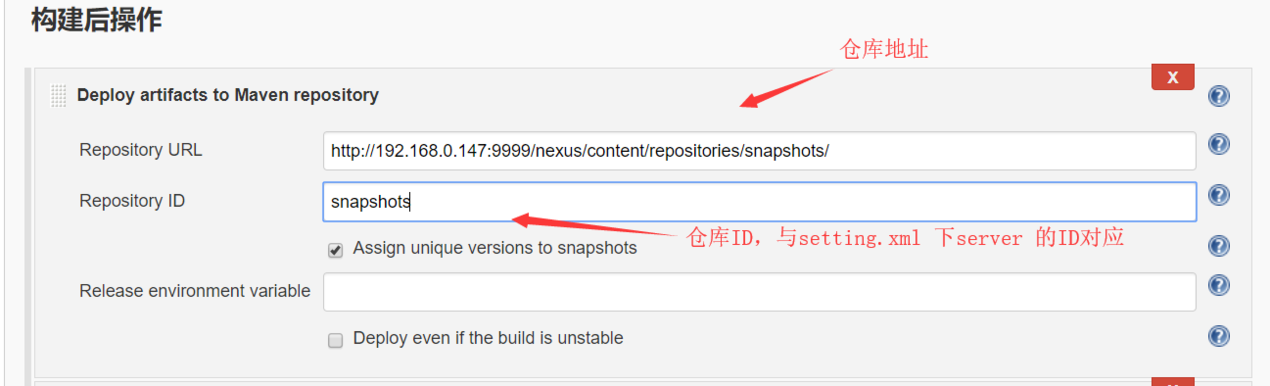
<role rolename="manager-script"/>

<user username="manager" password="manager" roles="manager-gui,manager-script"/>

<user username="admin" password="admin" roles="admin-gui,manager-gui"/>

**存档配置**

构建后操作添加 Deploy war/ear to a container 项目



配置setting.xml 用于获取上传至nexus 的权限

vim ~/.m2/settings.xml

# 添加一个 server id 与 存档配置当中的repository id 相对应。

 <server>

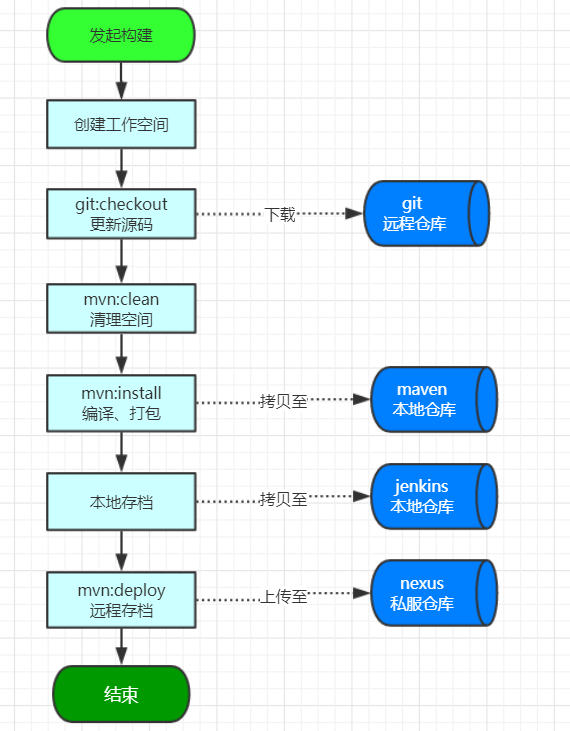
          <id>nexusReleases</id>

          <username>deployment</username>

          <password>111111</password>

</server>

## 2.3 集成原理



# 3、Jenkins pipeline核心应用

## 3.1 pipeline概要

前面我们演示的 使用maven 来进行自动化构建，其流程分别是：构建环境准备==》源码下载 ==》构建 ==》存档 ==》部署。这是一种固化的构建流程，如果你们的需求是多个项目需要进行依赖构建这种复杂的构建场景 时该怎么办？jenkins pipeline 可以做到这一点。

Jenkins从根本上讲是一种支持多种自动化模式的自动化引擎。Pipeline在Jenkins上添加了一套强大的自动化工具，支持从简单的连续集成到全面的连续输送Pipeline的用例。用户可以基于他实现更为复杂的建模场景。

## 3.2 pipeline的基础语法

以下就是一个非常简单的 pipeline 脚本：

pipeline {

agent any

stages {

stage('Build') {

steps {

sh 'make'

}

}

stage('Test'){

steps {

sh 'make check'

junit 'reports/\*\*/\*.xml'

}

}

stage('Deploy') {

steps {

sh 'make publish'

}

}

}

}

agent  表示Jenkins应该为Pipeline的这一部分分配一个执行者和工作区。

stage  描述了这条Pipeline的一个阶段。

steps  描述了要在其中运行的步骤 stage

sh  执行给定的shell命令

junit 是由JUnit插件提供的 用于聚合测试报告的Pipeline步骤。

## 3.3 pipeline 演示

1. Jenkins 2.x或更高版本
2. Pipeline插件

可以通过以下任一方式创建基本Pipeline：

1. 直接在Jenkins网页界面中输入脚本。
2. 通过创建一个Jenkinsfile可以检入项目的源代码管理库。

用任一方法定义Pipeline的语法是一样的，从项目源码中检入jenkinsfile 文件会更方便一些。

Pipeline脚本可以通过在Jenkins页面生成，选择相应的指令及操作即可生成脚本，cope到文件中即可，例如此部分脚本([$class: 'GitSCM', branches: [[name: '\*/master']], doGenerateSubmoduleConfigurations: false, extensions: [], submoduleCfg: [], userRemoteConfigs: [[credentialsId: 'gogs\_luban', url: 'http://git.jiagouedu.com/java-vip/tuling-api-gateway']]])

基于脚本构建 pipeline

pipeline {

agent any

stages {

stage('Checkout') {

steps {

echo 'Checkout'

stage('Build') {

steps {

echo 'Building'

}

}

stage('test'){

steps{

echo 'test'

}

}

}

}

基于Jenkinsfile 构建

pipeline {

agent any

stages {

stage('checkout') {

steps {

echo 'checkout'

checkout([$class: 'GitSCM', branches: [[name: '\*/master']], doGenerateSubmoduleConfigurations: false, extensions: [], submoduleCfg: [], userRemoteConfigs: [[credentialsId: 'gogs\_luban', url: 'http://git.jiagouedu.com/java-vip/tuling-api-gateway']]])

}

}

stage('build'){

steps {

echo 'build'

sh 'mvn clean install'

}

}

stage('save') {

steps {

echo 'save'

archiveArtifacts 'target/\*.war'

}

}

}

}