软件测试教程 持续集成jenkins篇

什么是持续集成?

测试人员进行测试工作的时候经常需要做一些高重复性的工作或者任务，这些任务往往伴随着固定并且繁多的步骤，测试人员在完成这项任务的时候往往需要一步一步去执行，由于步骤的繁多和复杂，可能会遗忘一些步骤，或者操作失误导致其中一个步骤失败，从而使整个任务失败，又要使我们重新操作。

这个时候有什么办法可以解决这种尴尬的情况呢？答案是持续集成。

持续集成就是可以把一个任务中的多个步骤，利用持续集成工具构建到一个job（这里可以理解为一个任务），实现任务的一键式执行和操作。

持续集成的优点

- 持续集成后的任务执行变得简单，直接，每一步操作不会出错。只需要一键执行构建的job，就可以自动完成任务。
- 持续集成中的任何一个环节都是自动完成的，无需人工干预，有利于减少重复过程以节省时间、费用和工作量；
- 任务执行有异常时可以再第一时间通知测试人员以及其他相关人员

本文主要讲解的持续集成工具是Jenkins，Jenkins 是一款流行的开源持续集成（Continuous Integration）工具，广泛用于项目开发，具有自动化构建、测试和部署等功能。

在敏捷时代，持续集成的作用越来越突出。本节主要讲解如下部分：

> jenkins 的简介
>
> jenkins 的安装以及环境配置
>
> jenkins 持续集成实战

jenkins简介

Jenkins 是一款流行的开源持续集成（Continuous Integration）工具，广泛用于项目开发，具有自动化构建、测试和部署等功能。

Jenkins是用运java语言开发的一款开源软件，所以在安装Jenkins的时候需要先配置java环境，开源意味着Jenkins可以免费试用，这是Jenkins工具的一个优势。

Jenkins有以下特点：

- 易安装、易配置；
- 基于Web访问，用户界面非常友好、直观和灵活；
- Jenkins虽然是基于Java开发的，但它不仅限于构建基于Java语言的任务，Python，shell都可以，所以Jenkins是一款强大的集成工具；
- 从检出代码、编译构建、运行测试、结果记录、测试统计等都是自动完成的，减少人工干预；
- 任何时间、任何地点生成可部署的软件，出现问题，项目成员会被马上通知到，问题第一时间修复；
- 增强项目可见性，有效的控制台日志能帮助我们更好的解决存在的问题；

- 拥有大量的插件：这些插件极大的扩展了Jenkins的功能；

Jenkins可以完成项目中的哪些集成?

1，对项目新版本的发布部署

测试人员测试时新版本部署时候的工作流程：拉取（pull）代码到本地->编译代码，生成war包->部署war包->发布版本

这个过程如果部署在Jenkins上，Jenkins会定时获取最新的代码，自动运行你的编译脚本，编译成功后，接着它会帮你把新程序发布出去。简而言之，Jenkins可以帮你在写完代码后，一键完成版本发布过程中的一系列工作。

2，执行自动化测试脚本的集成

测试人员执行自动化测试脚本时的工作流程：拉取（pull）代码到本地->运行代码->查看运行结果，分析测试结果。

Jenkins集成该过程，会实现自动化测试脚本的自动拉取和执行，并对测试结果进行分析，通知测试人员最后的执行结果。

使用Jenkins的好处显而易见，它减少了你的重复劳动。更重要的是，一个团队的开发流程一开始是不一致的，不一致往往会带来各种各样的问题，最终体现在软件的质量或开发效率不够高，而Jenkins会帮你规范大家的行为，从而避免一系列的问题。

jenkins的安装和环境部署

jenkins官网地址：https://jenkins.io/

备注：以下Jenkins的安装和其它环境的部署都是基于Linux环境的。

注意：不要在中文目录下运行

JDK安装

Jenkins是用Java语言安装的，所以需要先安装Java环境。

有的Linux服务器自带OpenJDK，但是建议大家卸载重新安装。

- 卸载openjdk包

  查看openjdk的相关安装包

  输入命令： rpm -qa | grep java

```
[root@localhost ~]# rpm -qa|grep java
javapackages-tools-3.4.1-11.el7.noarch
java-1.8.0-openjdk-headless-1.8.0.65-3.b17.el7.x86_64
java-1.7.0-openjdk-1.7.0.91-2.6.2.3.el7.x86_64
java-1.8.0-openjdk-1.8.0.65-3.b17.el7.x86_64
python-javapackages-3.4.1-11.el7.noarch
java-1.7.0-openjdk-headless-1.7.0.91-2.6.2.3.el7.x86_64
tzdata-java-2015g-1.el7.noarch
```

  输入命令 rpm -e --nodeps 安装包名称

  卸载完成后，输入rpm -qa | grep java 查看是否卸载干净

- 安装JDK

  先去官网下载rpm包到本地，用Xftp上传的服务器特定位置；

  上传 jdk-8u20-linux-x64.rpm到服务器

  运行rpm -ivh jdk-8u20-linux-x64.rpm

- 配置环境变量

  打开 /etc/profile 文件，在文件末尾输入以下几行：

```
JAVA_HOME=/usr/java/jdk1.8.0_291
JRE_HOME=/usr/java/jdk1.8.0_291/jre
PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin:$MAVEN_HOME/bin
CLASS_PATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar:$JRE_HOME/lib
```

保存，退出后，使用 source /etc/profile 使文件生效

运行 java -version，返回结果如下结果表示安装成功

```
[root@200 ~]#  java -version
java version "1.8.0_231"
Java(TM) SE Runtime Environment (build 1.8.0_231-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.231-b11, mixed mode)
```

jenkins的安装以及初次启动

这里我们用rpm来安装

1，使用wget下载jenkins的rpm包

wget https://mirrors.aliyun.com/jenkins/redhat/jenkins-2.266-1.1.noarch.rpm

或者

wget https://mirrors.tuna.tsinghua.edu.cn/jenkins/redhat/jenkins-2.267-1.1.noarch.rpm

将下载好的包放入指定的位置/root/zhaohuiwen/jenkins中

2，安装

执行以下命令进行安装

rpm -ivh jenkins-2.267-1.1.noarch.rpm

3，安装完成后，查看安装目录（以下jenkins文件所在目录可供参考）

find / -name jenkins

结果如下：

```
[root@200 jenkins]# find / -name jenkins
/run/lock/subsys/jenkins
/etc/rc.d/init.d/jenkins
/etc/logrotate.d/jenkins
/etc/sysconfig/jenkins
/root/zhaohuiwen/jenkins
/var/lib/jenkins
/var/log/jenkins
/var/cache/jenkins
/usr/lib/jenkins
```

其中

/etc/sysconfig/jenkins 目录下为Jenkins的配置文件

/var/lib/jenkins 目录下为Jenkins的安装目录，Jenkins的工作空间就在里面

/usr/lib/jenkins 目录下为Jenkins的war包，jenkins.war

4，配置jenkins端口

vi /etc/sysconfig/jenkins

找到JENKINS_PORT，修改JENKINS_PORT="8080"，默认为"8080"

```
## Type: string
## Default:     "-Djava.awt.headless=true"
## ServiceRestart: jenkins
#
# Options to pass to java when running Jenkins.
#
JENKINS_JAVA_OPTIONS="-Djava.awt.headless=true"

## Type:        integer(0:65535)
## Default:     8080
## ServiceRestart: jenkins
#
# Port Jenkins is listening on.
# Set to -1 to disable
#
JENKINS_PORT="8888"

## Type:        string
## Default:     ""
```

修改为非8080，避免和默认的8080端口冲突

5，启动jenkins

systemctl start jenkins

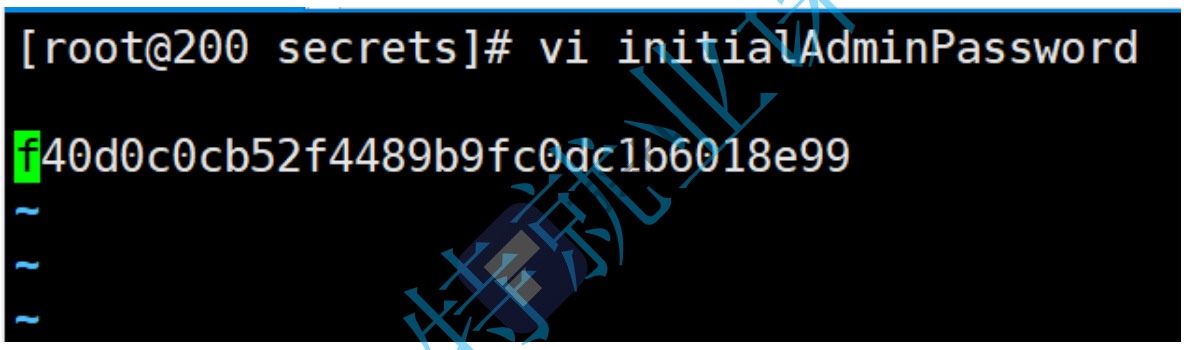
jenkins 登陆网址：http://42.192.83.143:11888

6，初始化Jenkins：

第一次登陆Jenkins有以下主要步骤：

- 提示输入管理员密码

在/var/lib/jenkins/secrets/initialAdminPassword



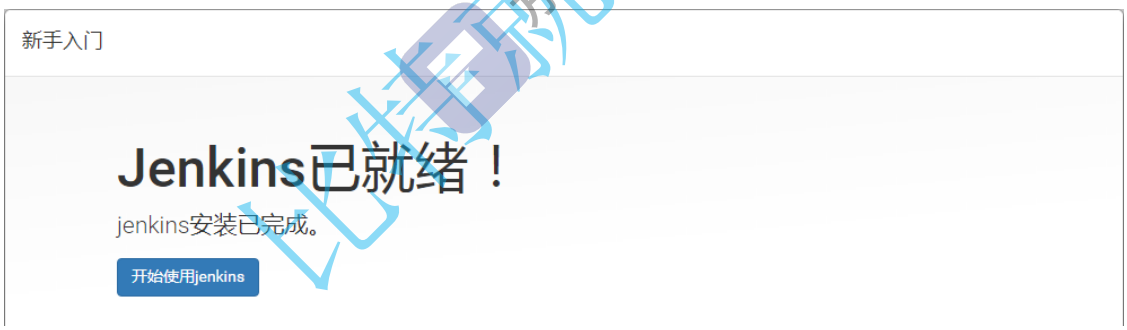
复制密码输入管理员密码即可。

- 提示安装插件

默认安装推荐的插件，如果安装失败了也没什么，Jenkins提供了一个插件管理界面，可以选择自己需要的插件重新安装

- 创建管理员账户

管理员账户信息填好后，进入完成页面：



其他环境的安装

maven安装

- 下载，

  下载地址： http://maven.apache.org/download.html
- 解压

  命令：tar -zxvf apache-maven-3.8.2-bin.tar.gz

  解压后的目录为：/usr/local/maven/apache-maven-3.8.2
- 配置环境变量

  在/etc/profile 添加下面两行:

```
export MAVEN_HOME=/usr/local/maven/apache-maven-3.8.2
export PATH=${MAVEN_HOME}/bin:${PATH}
```

- 使环境参数生效:

  执行 source /etc/profile

- 查看mvn版本，看是否配置正确。

  执行 mvn –version

TOMCAT安装

- tomcat下载

  下载地址：https://tomcat.apache.org/download-80.cgi

  下载后上传到要安装的服务器上的特定位置

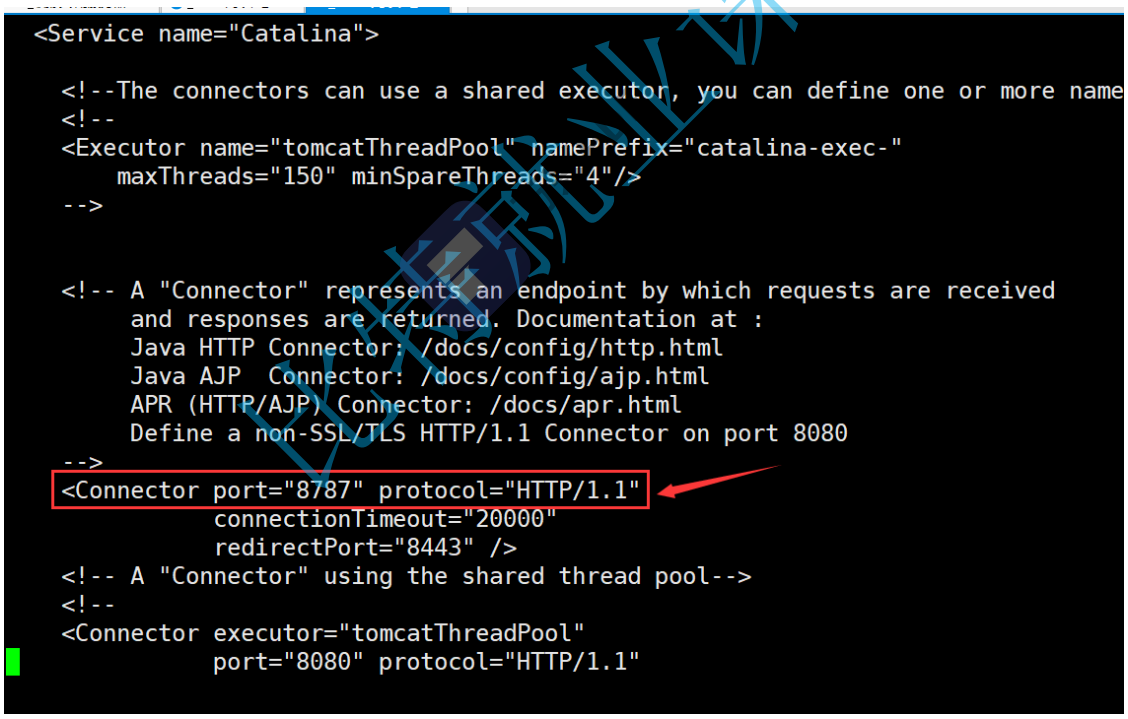
- 解压

  tar -xvzf apache-tomcat-8.5.57.tar.gz

- 修改端口号

  进入tomcat的配置文件，目录在：/usr/zhaohuiwen/tomcat/apache-tomcat-8.5.57/conf

  用vi打开server.xml



把端口port号修改成非8080的即可，

修改端口号，主要是为了避免端口冲突。

- Tomcat服务器启动和关闭

  启动Tomcat服务器

进入tomcat服务器的bin目录，然后执行"**./startup.sh**"或者sh **startup.sh**命令启动Tomcat服务器，如下图所示:

```
[root@198 bin]# pwd
/usr/zhaohuiwen/tomcat/apache-tomcat-8.5.57/bin
[root@198 bin]# ls
bootstrap.jar       commons-daemon-native.tar.gz  setclasspath.sh       tool-wrapper.bat
catalina.bat        configtest.bat                shutdown.bat          tool-wrapper.sh
catalina.sh         configtest.sh                 shutdown.sh           version.bat
catalina-tasks.xml  daemon.sh                     startup.bat           version.sh
ciphers.bat         digest.bat                    startup.sh
ciphers.sh          digest.sh                     tomcat-juli.jar
commons-daemon.jar  setclasspath.bat              tomcat-native.tar.gz
```

Tomcat的log日志放在logs目录下：/usr/zhaohuiwen/tomcat/apache-tomcat-8.5.57/logs

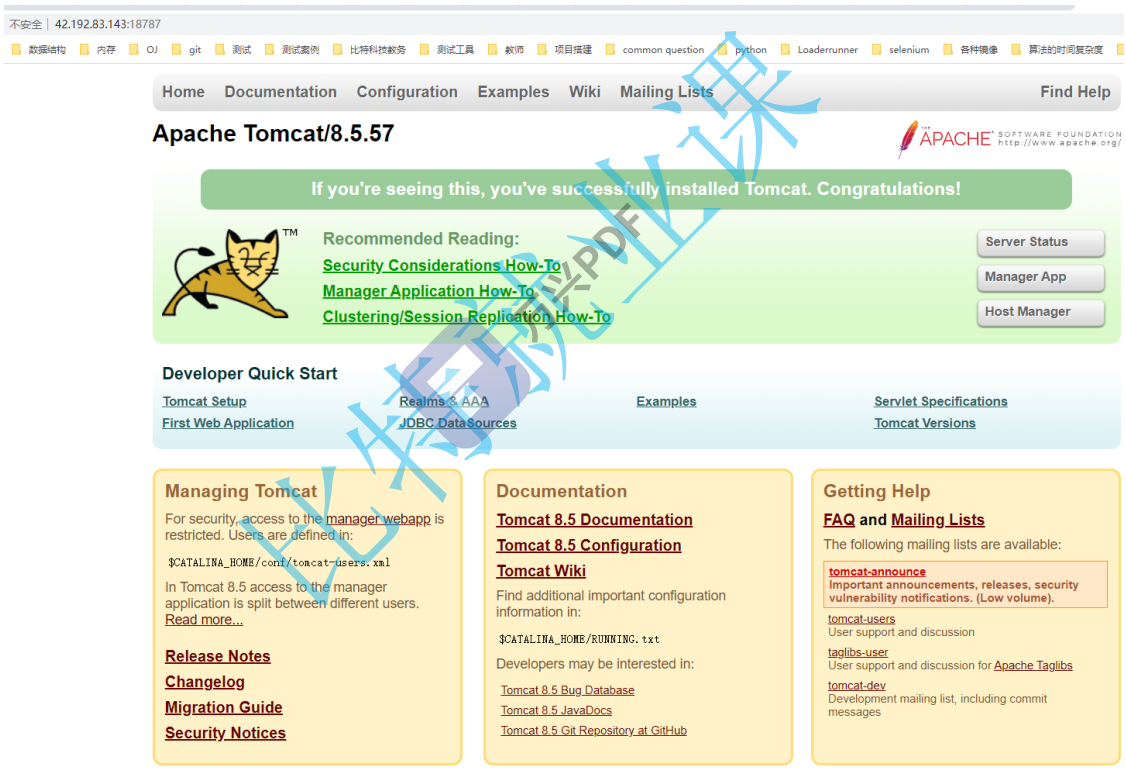
如果Tomcat启动的时候出现问题可以通过查看日志定位问题。

关闭Tomcat服务可以在该目录下 执行，sh shutup.sh命令，进行Tomcat的关闭。

img

- 访问Tomcat服务

  可以通过在浏览器访问Tomcat主页面，判断其是否启动成功，在浏览器输入服务器IP和端口号即可。



mysql安装

用yum安装mysql

- 安装mysql

  yum -y install mysql-server

  一般直接执行这个命令会找不到源，需要先下载**mysql的repo源**

  下载mysql的repo源：wget http://repo.mysql.com/mysql-community-release-el7-5.noarch.rpm

  执行该命令后，会在当前目录下下载一个 mysql-community-release-el7-5.noarch.rpm

  执行以下命令安装**安装mysql-community-release-el7-5.noarch.rpm包**：

  rpm -ivh mysql-community-release-el7-5.noarch.rpm

使用yum安装mysql：

yum install mysql-server

安装好之后**查看服务是否已经添加到linux上**：

执行命令：rpm -qa | grep mysql

```
[root@198 logs]# rpm -qa | grep mysql
mysql-community-libs-5.6.51-2.el7.x86_64
mysql-community-server-5.6.51-2.el7.x86_64
mysql-community-release-el7-5.noarch
mysql-community-common-5.6.51-2.el7.x86_64
mysql-community-client-5.6.51-2.el7.x86_64
```

出现以上，说明mysql服务安装成功

- 关于mysql 的启动

  service mysqld start

  新安装的mysql需要先设置管理员

  执行命令：mysqladmin -u root password root

  设置完管理员用户名密码后，可以直接登陆：

  执行mysql -uroot -proot可以进入mysql服务：

```
[root@198 logs]# mysql -uroot -proot
Warning: Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 122
Server version: 5.6.51 MySQL Community Server (GPL)

Copyright (c) 2000, 2021, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```


jenkins实战

1，Jenkins的重要配置

管理Jenkins

这里面有Jenkins的重要配置，很多配置是我们是否能够集成成功的关键。

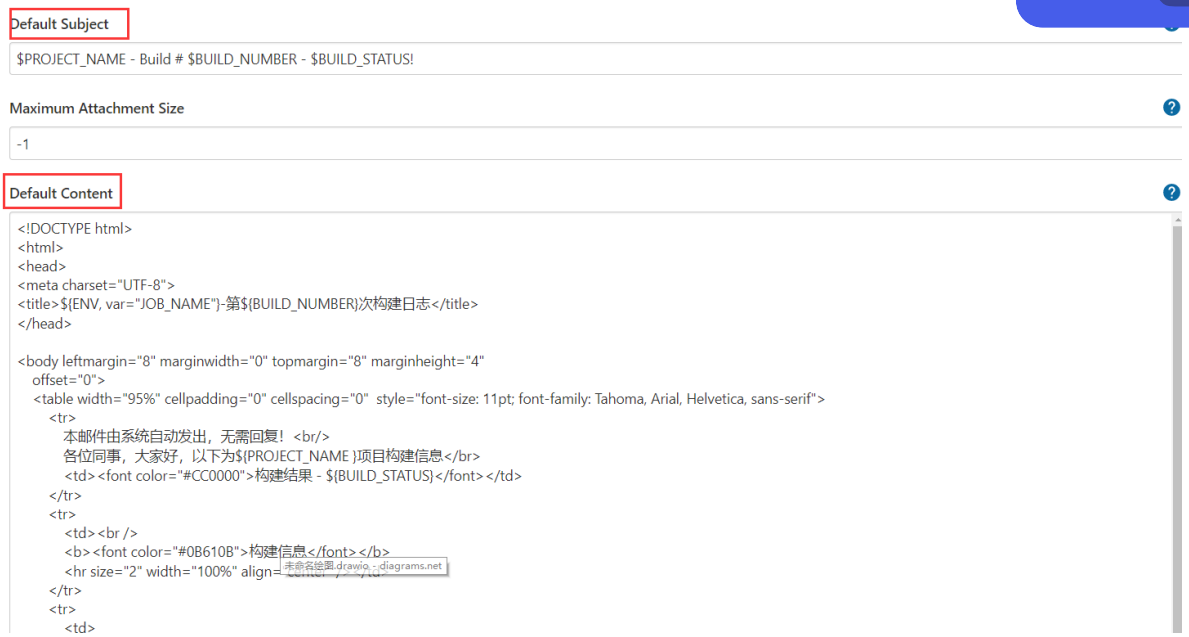
我们这里重点讲解以下几个部分

System Configuration

1) Configuration System -配置

Extended E-mail Notification-自动发送邮件配置

如果集成的项目中需要集成自动发送邮件的功能，就需要配置这一项，主要有以下重要配置



Default Subject 是发送邮件的默认主题，这里取的是构建项目的名称，构建的次数，构建的状态拼接成的。

表示如下：

```
$PROJECT_NAME - Build # $BUILD_NUMBER - $BUILD_STATUS!
```

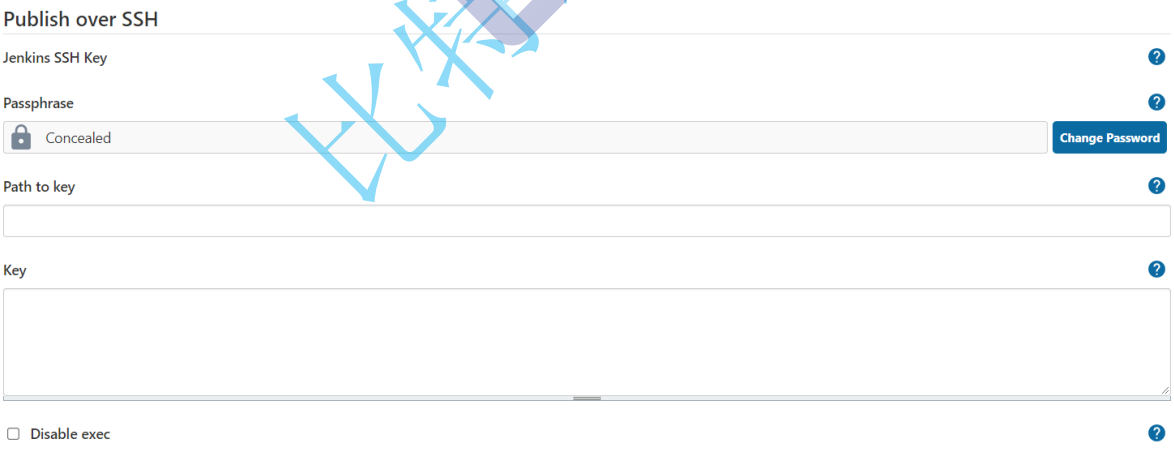
Default Content 是发送邮件默认的内容，是以html脚本。代码如下：

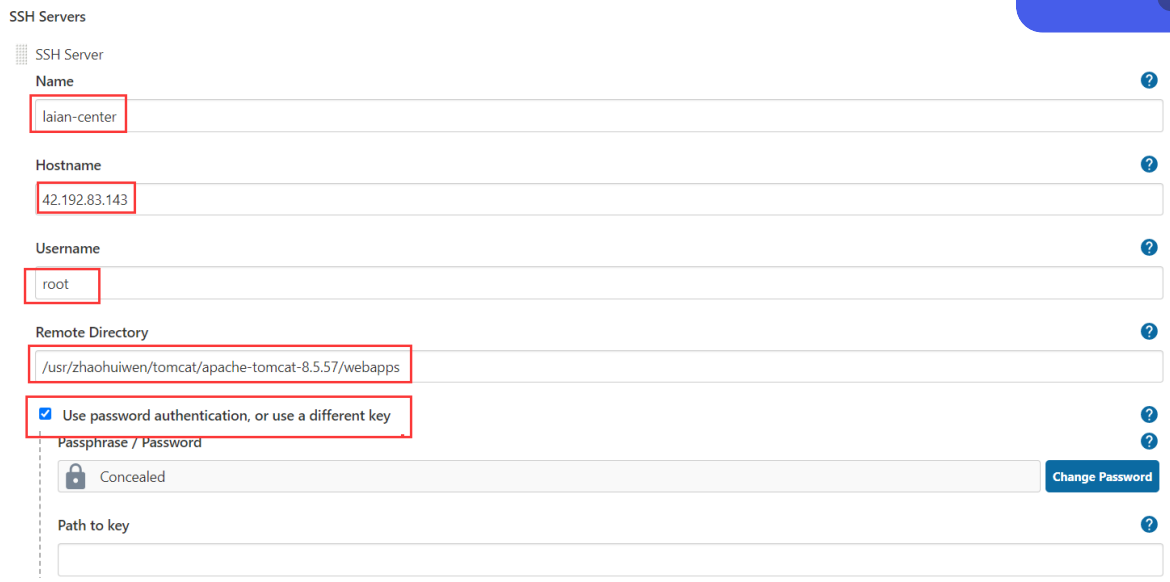
```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>${ENV, var="JOB_NAME"}-第${BUILD_NUMBER}次构建日志</title>
</head>
<body leftmargin="8" marginwidth="0" topmargin="8" marginheight="4"
    offset="0">
    <table width="95%" cellpadding="0" cellspacing="0"  style="font-size: 11pt;
font-family: Tahoma, Arial, Helvetica, sans-serif">
        <tr>
            本邮件由系统自动发出，无需回复！<br/>
            各位同事，大家好，以下为${PROJECT_NAME }项目构建信息</br>
            <td><font color="#CC0000">构建结果 - ${BUILD_STATUS}</font></td>
        </tr>
        <tr>
            <td><br />
            <b><font color="#0B610B">构建信息</font></b>
            <hr size="2" width="100%" align="center" /></td>
        </tr>
        <tr>
            <td>
                <ul>
                    <li>项目名称 ：  ${PROJECT_NAME}</li>
                    <li>构建编号 ：  第${BUILD_NUMBER}次构建</li>
                    <li>触发原因：  ${CAUSE}</li>
                    <li>构建状态：  ${BUILD_STATUS}</li>
                    <li>构建日志：  <a
href="${BUILD_URL}console">${BUILD_URL}console</a></li>
```

```
                    <li>构建  Url :  <a href="${BUILD_URL}">${BUILD_URL}

                    <li>工作目录 :  <a
href="${PROJECT_URL}ws">${PROJECT_URL}ws</a></li>
                    <li>项目  Url :  <a href="${PROJECT_URL}">${PROJECT_URL}</a>
</li>
                </ul>

<h4><font color="#0B610B">失败用例</font></h4>
<hr size="2" width="100%" />
$FAILED_TESTS<br/>

<h4><font color="#0B610B">最近提交(#$SVN_REVISION)</font></h4>
<hr size="2" width="100%" />
<ul>
${CHANGES_SINCE_LAST_SUCCESS, reverse=true, format="%c", changesFormat="<li>%d
[%a] %m</li>"}
</ul>
详细提交: <a href="${PROJECT_URL}changes">${PROJECT_URL}changes</a><br/>

            </td>
        </tr>
    </table>
</body>
</html>
```

Publish over SSH

这个插件没有的话，需要去插件管理中下载，主要用来登陆远程服务器，向远程服务器发送文件等操作





SSH Servers中Name可以自己指定，去一个有辨识度的名字就好；

Hostname 是需要连接的远程主机IP

Username 连接远程主机的用户名

Port 远程主机的是端口号

验证方式可以直接用账户名密码验证

System Configuration中其他的配置均可默认，或者某些特殊项需要的时候再配置，不用纠结。

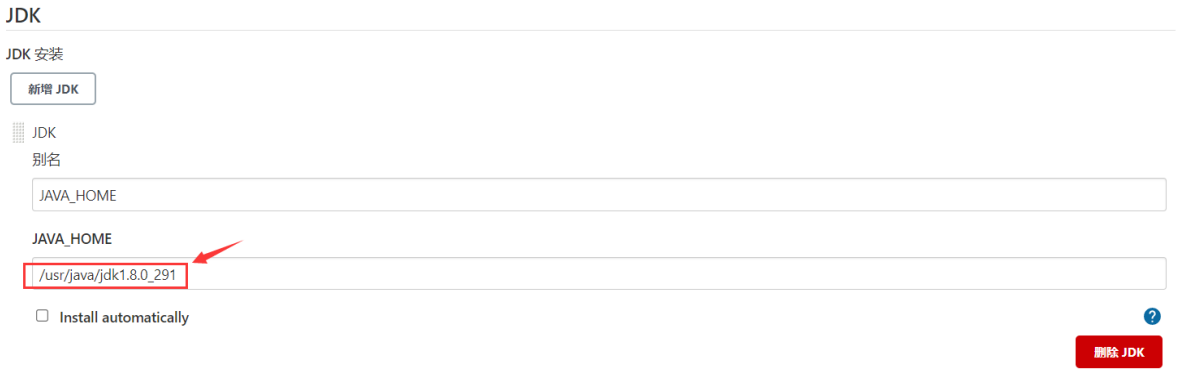
2）**Global Tool Configuration-全局工具配置**

配置jdk，maven，git等插件，JDK和Maven主要配置成Jenkins所在服务器的JDK和Maven所在的路径就可以了。

如下图所示

Maven配置

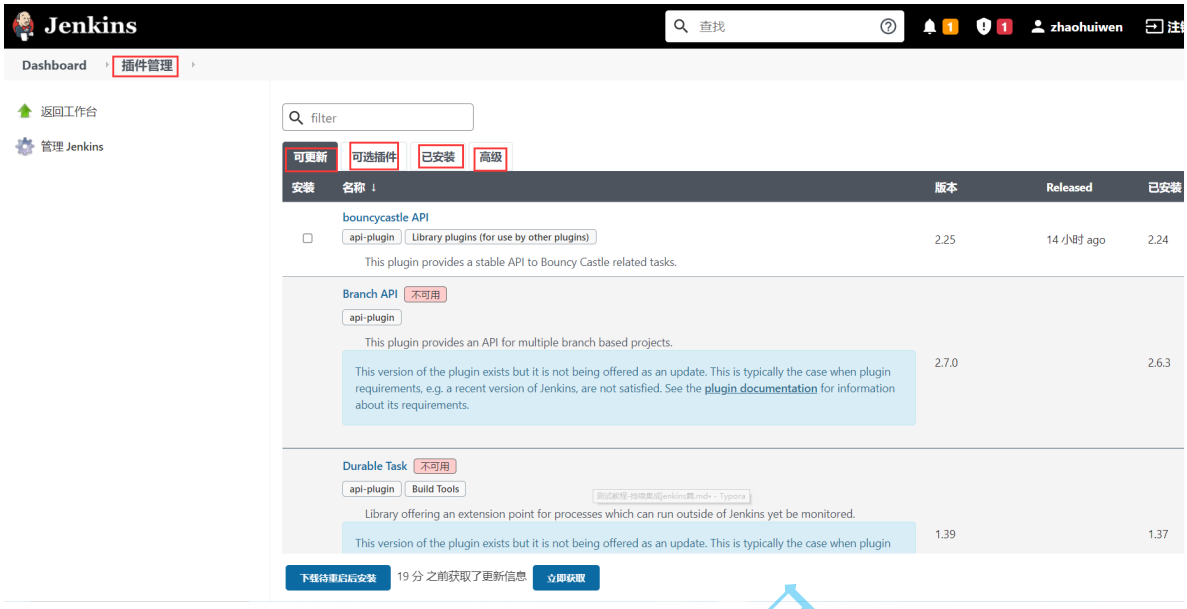
JDK配置



Git默认就可以了。

3）**插件管理**

插件管理中进行差将的安装、卸载



可以在搜索框中查找自己想要安装的插件在Jenkins中直接进行安装

如果无法直接在Jenkins安装可以去“**高级**”里直接上传.hpi文件进行插件的安装。

4）**全局安全配置**

配置授权策略

建议采用Jenkins专用数据库，安全策略采用安全矩阵方式，已经能够基本满足现有的模式。

安全域：

1、Jenkins专有用户数据库

2、LDAP

授权策略：

1、安全矩阵

| Overall | Credentials | Agent | Job | Run | View | SCM |

Agents：

当需要设置slave时，TCP port for JNLP agents需要启用，否则不同通过java web启动代理

5）**管理用户**

增加jenkins新用户

6）**系统信息**

查看系统环境信息

7）**管理节点**

Jenkins的分布式构建，在Jenkins的配置中叫做节点。

当我们使用多台服务器时，可通过jenkins的节点配置，将jenkins项目发布在不同服务器上进行编译、部署，这就形成了jenkins的分布式

简单解释一下配置：

名字：该节点的名字。

描述：说明这个节点的用途。

# of executors：允许在这个节点上并发执行任务的数量，一般设置为 cpu 支持的线程数。

远程工作目录：节点上 Jenkins 的根目录。

标签：分配给这个节点的标签。

用法：节点的使用策略。

启动方法：启动 agent 的方式，对于 windows 平台，最好选择 "**通过Java Web启动代理**"。

对于linux平台，选择**Launch slave agents via SSH**

Availability：Jenkins 控制 slave 是否在线的策略。

Node Properties：设置工具列表和环境变量

8）**读取设置**

放弃当前内存中所有的设置信息并从配置文件中重新读取 仅用于当您手动修改配置文件时重新读取设置。

###视图

用于分类管理，使job分类清晰

新建视图

1、在Jenkins主界面中点击图示的【+】开始执行视图创建工作

2、在【新建视图】页面，按照图示填写“视图名称”，选择“List View”点击【OK】按钮

3、在【视图配置】页面中，我们可以给当前的视图添加描述性信息，添加完成之后，点击【保存】按钮

修改视图

1、点击【编辑视图】按钮，进入【编辑视图】页面

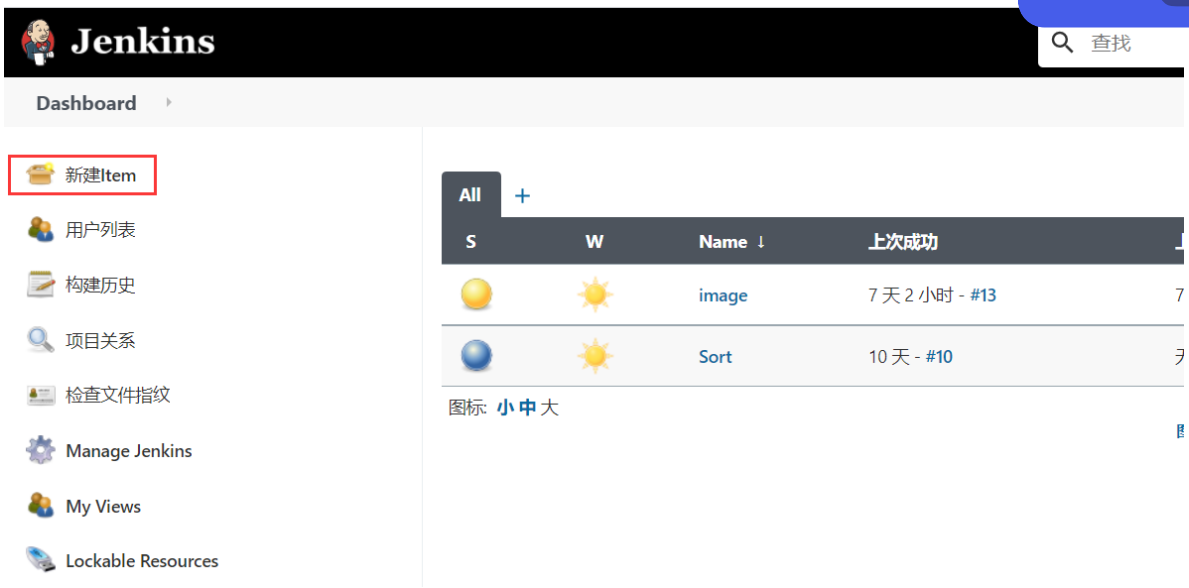
2、在【编辑视图】页面，将我们所修改的编写完之后，点击【保存】按钮，即可实现编辑视图的操作

删除视图

1、点击【删除视图】按钮，即可实现视图的删除操作，删除视图不会影响job

新建item—构建集成任务（Job）

在主页面的**新建item**里面构建集成任务，在Jenkins中我们把一个集成任务称之为一个Job。



点击**新建item**，进入任务构建的初始页面

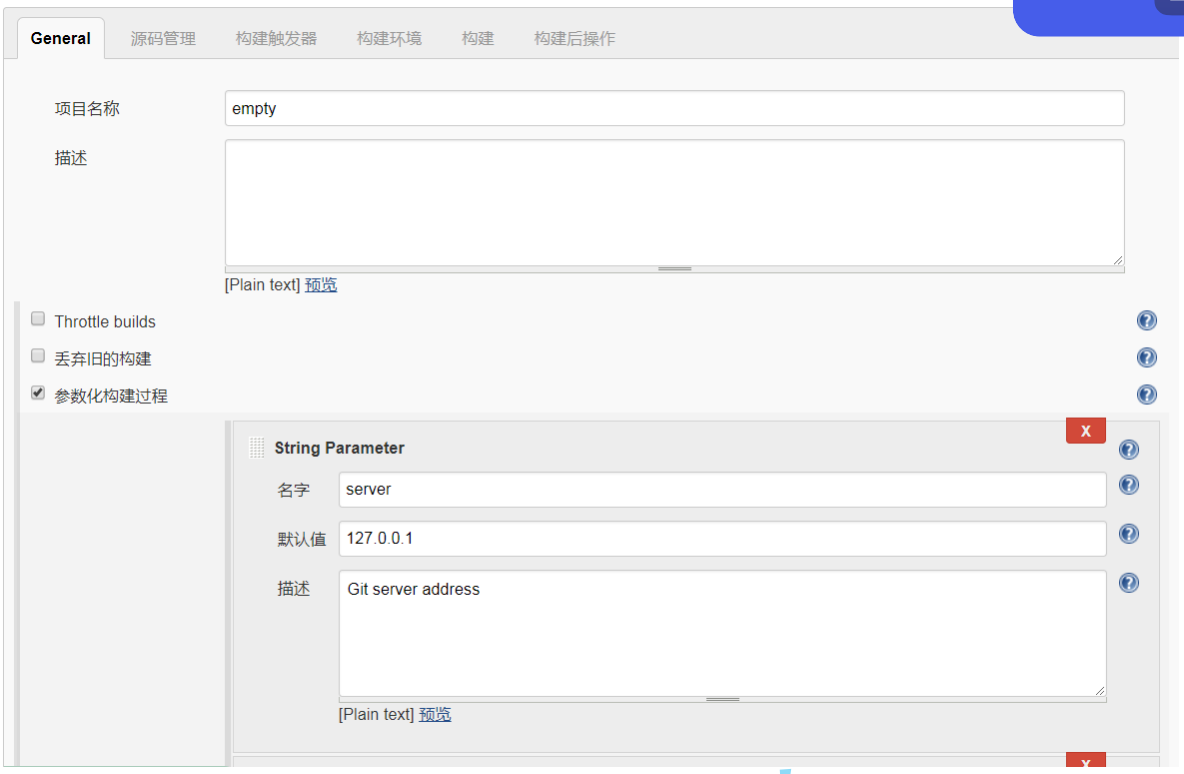
Freestyle project：jenkins最常用的类型，一般没有特殊需求可以选择它

构建一个maven项目：当安装了maven插件之后，会出现该项目，仅针对maven构建方式的项目，比如需要生成war包的项目。

构建一个多配置项目：当需要多环境时，必须多环境编译、部署时可以采用该配置

构建任务的配置界面

输入自定义的任务名称，点击**Freestyle project**，进入任务的配置页面：



构建的配置主要包含以下部分：

1）项目名称和描述

2）项目安全

3）参数化构建过程：是否升级db、选择部署环境等

4）源码管理：从SVN、GIT等获取代码

5）构建触发器：是否与其他JOB有依赖关系，是否使用定时器等

6）构建：调用ant、maven、shell等脚本

7）日志查看，点击console output可以查看日志信息

任务构建执行过程如图：

常见配置项解释：

项目名称与描述：名称不建议为中文，项目（任务）名称不能重复

丢弃旧的构建：设置构建历史的保存策略，可以节省空间，可以按天数或者个数来设置

参数化构建过程：某些构建过程，需要一些输入参数，常用的有choice parameter，string parameter等

Restrict where this project can be run：当jenkins配置有slave时，可以设置job在该slave上运行

github project :github使用，里面配置响应的url和需要显示的名称就可以了

throttle builds：节流构建，通过设置时间段内允许并发的次数来实现构建的控制

参数化构建过程：里面可以配置不同的参数，便于在构建时引用这些参数

windows引用方式%paraname%,shell为$paraname

关闭构建：项目无法进行构建

安静期：设置一个时间来间隔每次构建的间隔

重试次数：拉取源码重试的次数

该项目的上游项目正在构建时阻止该项目构建与该项目的下游项目正在构建时阻止该项目构建：用于上下游项目有关联的构建策略

使用自定义的工作空间：指定当前任务的workspace，否则默认为JENKINS_HOME的工作目录

保留构建的依赖日志

源码管理：选择使用git或者svn，以git为例：

以svn为例时

```
repository url:填写仓库的地址

Credentials：这里需要配置拉取svn源码的用户名和密码

Local module directory：具体的项目的路径，默认从根目录拉取

Additional Credentials：增加额外认证

Check-out Strategy：代码检出策略

            Use 'svn update'  as much as possible：相当于svn update

            Always check out a fresh copy：checkout之前先清除工作区文件

源码库浏览器：这里默认就可以了
```


构建触发器：

1、触发远程构建 (例如,使用脚本)：这里使用于自动化构建，拼接url后写入代码中可以实现在脚本或者工具执行构建

2、Build after other projects are built:构建与其他项目构建后，用于上下游项目有关联的时候

3、Build periodically：定时执行构建

日程表的参数：

```
第一个参数代表的是分钟 minute，取值 0~59；
第二个参数代表的是小时 hour，取值 0~23；
第三个参数代表的是天 day，取值 1~31；
第四个参数代表的是月 month，取值 1~12；
最后一个参数代表的是星期 week，取值 0~7，0 和 7 都是表示星期天。
```

4、Build when a change is pushed to GitHub:这个是github项目的触发规则

5、Poll SCM：设置定时检查代码仓库是否有变更，有变更则构建

构建环境：

1、Delete workspace before build starts：在构建之前清空工作空间

2、Abort the build if it's stuck：如果构建出现问题则终止构建

3、Add timestamps to the Console Output：给控制台输出增加时间戳

4、Use secret text(s) or file(s)：使用加密文件或者文本

构建

1、execute windows batch command：执行windows的cmd

2、execute shell：执行shell命令

3、invoke ant：调用ant ,调用ant的执行脚本来进行构建

备注：

- Targets：主要是执行ant脚本中哪几个部分，可以添加多个；
- Build File:需要指定Ant脚本的物理位置；
- Properties：添加Ant指定的属性；
- Java Options：设置运行java时的属性，例如内存、堆大小等；

4、invoke gradle script ： 调用grade脚本，来帮助我们自动打包

5、invoke top-level maven targets：调用maven

不同的语言可能会采用不同的构建方式

构建后操作

1、build other projects:构建其他项目

2、e-mail notification:发送邮件

3、delete workspace when build is done:构建后删除工作空间

运行并监控构建作业

当配置完成一个任务后，回到主控制面板：

1、列表列举现在已经配置的任务已经任务当前的状态

2、左边有构建队列，当有构建时，会把当前正在构建的队列在上面进行列举

3、当一个任务配置完成后，可以采用手动构建和触发器构建两种方式，在项目验证阶段，可以通过手动触

发方式，点击任务区的“立即构建”，会在Build History中出现进度条

4、点击进度条，可以进入到具体的编译过程

任务构建状态

当前构建状态分为以下几种：

	项目构建完成，同时被认为是稳定的
	项目构建完成，但被认定为不稳定
	构建失败
	作业已经禁止

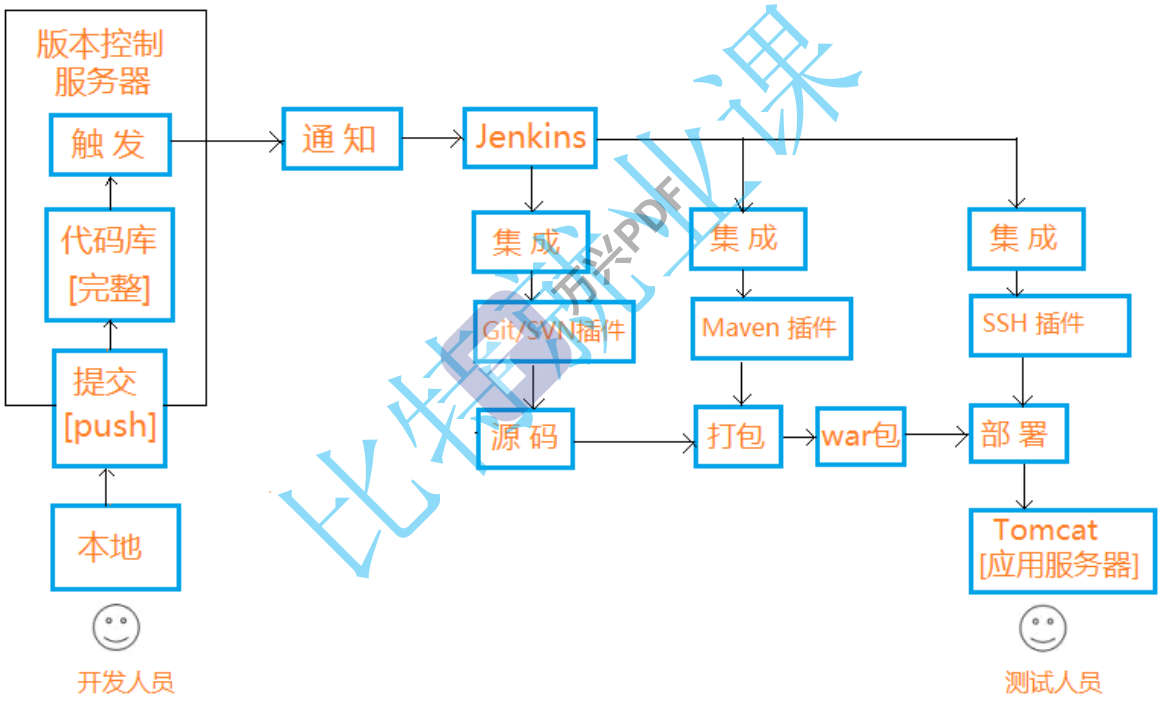
构建稳定行，Jenkins会基于一些后处理器任务为构建发布一个稳健指数(从0-100)，越高越稳定

	构建成功率>80%
	构建成功率 60%-79%
	构建成功率 40%-59%
	构建成功率 20%-39%
	构建成功率 0-19%

实践案例（1）—图片服务器项目的部署

项目的部署是测试人员经常要做的工作，项目的部署流程：拉取（pull）代码到本地->编译代码，生成war包->部署war包->发布版本

对应的流程图如下：



- Jenkins中构建项目部署的任务流程

前提：图片文件服务器代码已上传到码云

1、安装Maven Integration plugin插件

2、新建-构建一个maven项目

3、源码管理-选择git

4、Build

5、构建环境：选择发布到远程服务器上

6、构建后操作：邮件通知相关测试人员和开发人员，版本发布的情况；

- 构建项目

  进入构建item

  输入任务名称—选中构建一个maven项目—点击确定

点击确定后进入任务构建（集成）配置页面

描述可以根据这个任务的用途进行填写

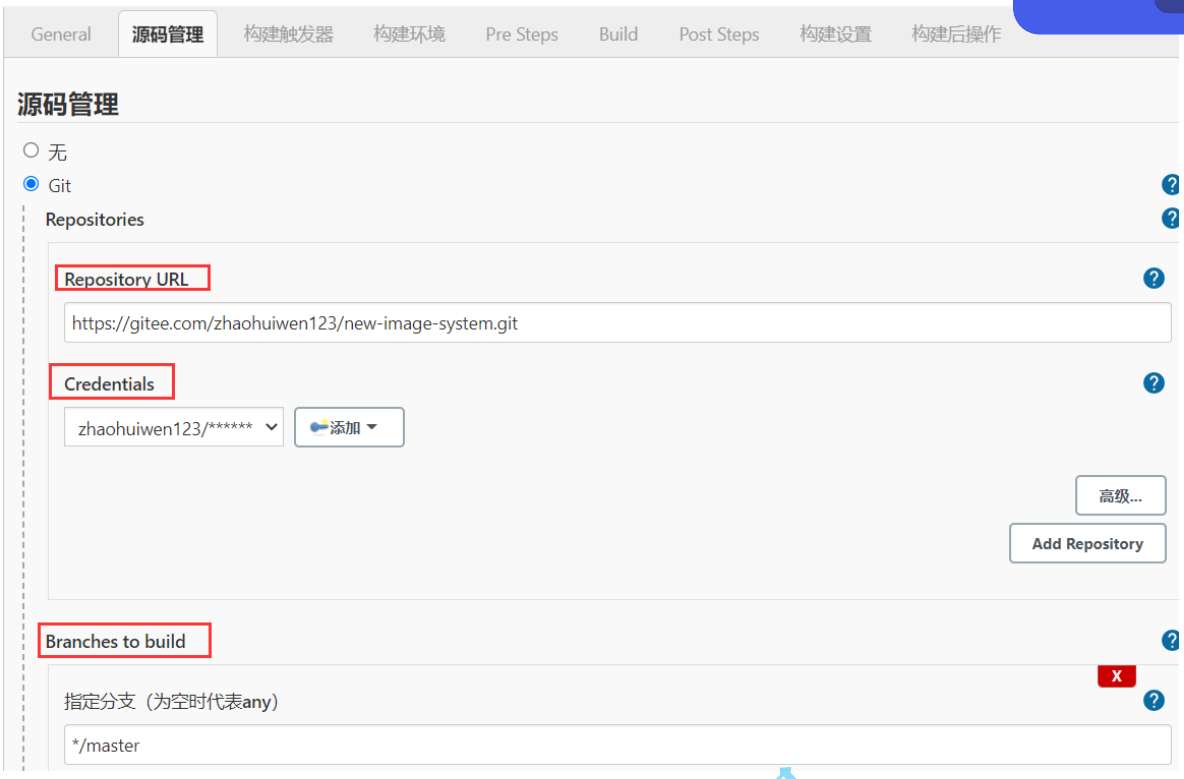
配置源码

选择git

Repository URL：填写码云上项目的仓库地址

Credentials：填写码云的验证方式，即账户名和密码

Branches to build：填写拉取代码的分支



build—编译代码生成war包

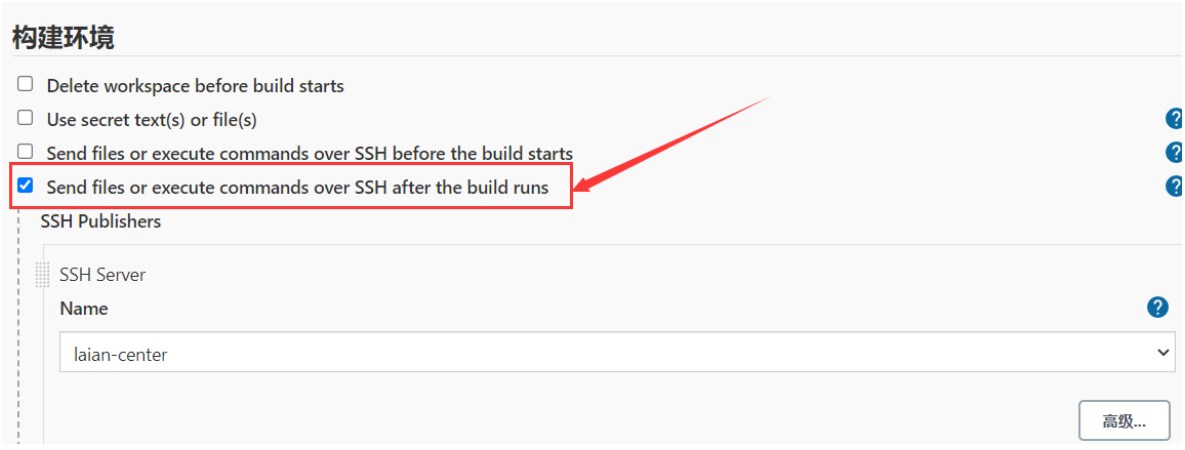
将代码从码云上拉下来后，在本地进行编译，生成war包

Root POM：填写对应workspace中的pom文件

Goals and options：填写编译指令

构建环境

代码编译完后，需要把生成的war包传到项目部署服务器上，选择 Send files execute commands over SSH after the build runs



配置SSH Server

Name：选择已经配置好的服务器

Source files：填写需要上传的文件以及文件所在的路径

Remove prefix：去掉要上传的文件的路径

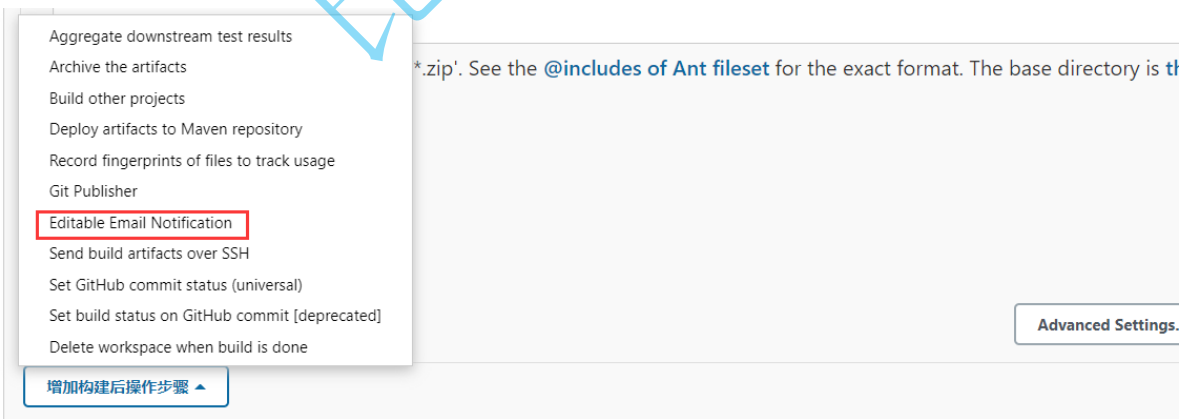
Remote directory：登陆到远程服务器上的目录

Exec command：在远程服务器上执行的命令

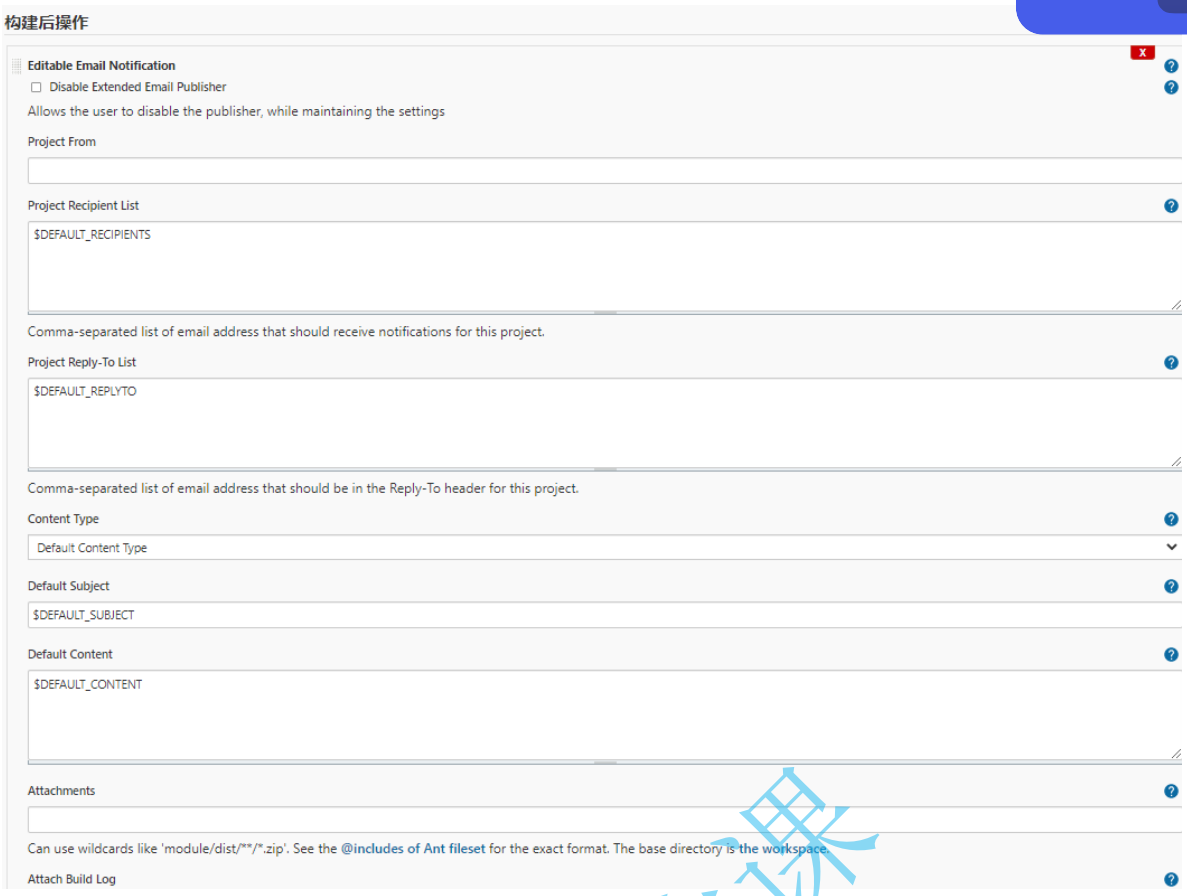
构建后操作

任务构建后将构建的结果通过邮件发送给相关的测试和开发人员。

在增加构建后操作步骤中—选择Editable Email Notification



就可以默认带出已经在全局配置中已经配置好的发件人，收件人，以及发送内容。



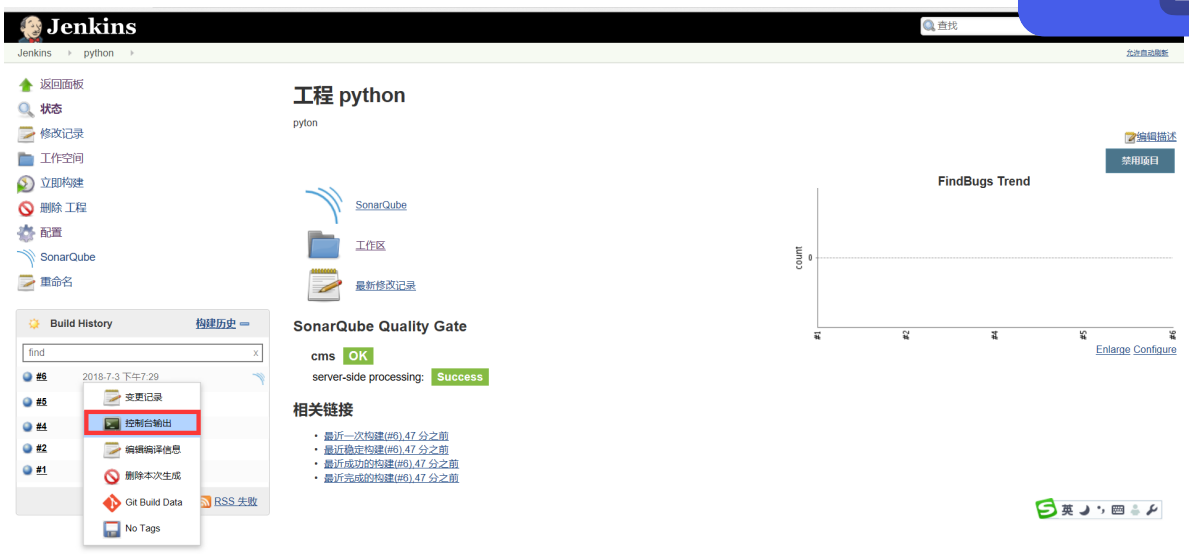
所有的内容配置完了之后，点击保存，那图片项目部署任务就构建完成了

保存之后，页面就会直接进入项目的构建页面。

点击Build Now，就可以直接构建该任务了

查看构建日志

可以在控制台中察看整个构建过程中输出的日志，日志中有详细的构建过程，如果有问题可以查看原因



访问项目：http://42.192.83.143:18787/java_image_server/index.html，访问成功证明项目部署成功。

访问成功证明项目部署成功。