**CI持续集成**

持续集成(CI)是一种实践，旨在缓和和稳固软件的构建过程。持续集成最早由Martin Fowler 于10年前已经提出，希望通过持续集成能够实现以下过程：

* 任何人在任何地点，任何时间可以构建整个项目。
* 在持续集成构建过程中，每一个单元测试都必须被执行。
* 在持续集成构建过程中，每一个单元测试都必须通过。
* 持续集成构建的结果是可以发布的软件包。
* 当以上任何一点不能满足时，整个团队的主要任务就是去解决这个问题。

Jenkins是一个软件界非常流行的开源CI服务器。

注意jdk和mysql的版本影响未来安装sonarqube的版本及sonarqube的中文支持

1. 安装必要的软件环境 wget gcc make subversion mysql-server mysql unzip

yum install –y wget gcc make upzip

1. 安装LNMP支持环境（为了更好的能够远程查看各种报表）可以自己安装我是自己找的一键安装包

wget –c <http://soft.vpser.net/lnmp/lnmp1.0-full.tar.gz>

解压安装

tar –zxvf lnmp1.0-full.tar.gz

cd lnmp1.0-full/

./centos.sh

1. 下载JDK支持环境因为安装jenkins maven ant sonarqube都需要java的环境

去官方网站下载

jdk <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html> 我使用的是jdk1.7.0\_67,我现在的是rpm的安装包

先删除本机自带的jdk否则会遇到jvm和jdk默认使用的版本问题

rpm –e jdk-6u80-linux-x64.rpm （可以加 --nodeps）清除依赖

安装jdk：rpm –ivh jdk-7u67-linux-x64.rpm

配置jdk系统的环境变量

vim + /etc/profile 在最后添加一下内容

export JAVA\_HOME=/usr/java/jdk1.7.0\_67

export JRE\_HOME=/usr/java/jdk1.7.0\_67/jre

export CLASSPATH=$JAVA\_HOME/lib:$JRE\_HOME/lib:/usr/share/ant/lib/ant-launcher.jar 红色这部分是为了增加ant的路径

export PATH=$JAVA\_HOME/bin:$JRE\_HOME/bin:$PATH

export JAVA\_HOME JRE\_HOME PATH CLASSPATH

保存并退出

source /etc/profile

检查java的版本

java --version 或者 javac --version

查看环境变量

echo $JAVA\_HOME $JRE\_HOME

如果和你安装的版本发生不一致，可能就是你java的本机自带的jdk环境存在

[root@test ~]# alternatives --config java

There are 3 programs which provide 'java'.

Selection Command

-----------------------------------------------

1 /usr/lib/jvm/jre-1.5.0-gcj/bin/java

+ 2 /usr/java/jdk1.7.0\_67/bin/java

\* 3 /usr/java/jdk1.8.0\_73/jre/bin/java

Enter to keep the current selection[+], or type selection number:2

注意：maven ant sonarqube 我自己默认的放置路径/usr/local/

1. 安装maven环境

下载Maven安装包：你可以在网上找相对较新的版本

apache-maven-3.3.9-bin.tar.gz

tar –zxvf apache-maven-3.3.9-bin.tar.gz

ln –s apache-maven-3.3.9 maven 做个软连接

在/etc/profile配置环境变量

vim + /etc/profile

#maven profile

export M3\_HOME=/usr/apache-maven-3.3.9

export PATH=${M3\_HOME}/bin:${PATH}

保存并退出

source /etc/profile

mvn –version

[root@test usr]# mvn --version

Apache Maven 3.3.9 (bb52d8502b132ec0a5a3f4c09453c07478323dc5; 2015-11-11T00:41:47+08:00)

Maven home: /usr/apache-maven-3.3.9

Java version: 1.7.0\_67, vendor: Oracle Corporation

Java home: /usr/java/jdk1.7.0\_67/jre

Default locale: en\_US, platform encoding: UTF-8

OS name: "linux", version: "2.6.32-431.el6.x86\_64", arch: "amd64", family: "unix"

1. 配置mysql

service mysqld start

登录mysql

mysql

mysql >usr mysql

mysql > update user set password=PASSWORD(‘root1234’) where user=’root’;

mysql> flush privileges;

并设置授权

mysql>grant all privileges on \*.\* to ‘root’@’%’ identified by ‘root1234’;

mysql> flush privileges;

注册mysql服务开机启动

chkconfig mysqld on

由于svn使用公司内部的svn有具体的路径和用户这里不再jenkins的服务器上配置svn

## 安装sonarqube环境

下载地址：<http://www.sonarsource.org/>

下载中文语言包： <http://docs.codehaus.org/display/SONAR/Chinese+Pack>

以上两个安装文件版本对jdk和mysql的版本都有要求 ，因为配置的时候启动报错

自己去官方网站查找相应的版本说明

安装前配置sonarqube链接数据库的支持

msyql –uroot –proot1234

mysql>create database sonar character set utf8 collate utf8\_general\_ci;

mysql>grant all on sonar.\* to sonar identified by ‘sonar’;

mysql>grant all on sonar.\* to sonar@’localhost’ identified by ‘sonar’;

mysql> flush privileges;

安装sonarqube5.1.1

自己去官网下载 /usr/local/

upzip sonarqube-5.1.1

vim + /etc/profile

export SONAR\_HOME=/usr/sonarqube-5.1.1

export PATH=$PATH:$SONAR\_HOME

source /etc/profile

配置sonarqube环境配置

vim /usr/local/sonarqube-5.1.1/conf/sonar.properties文件

sonar.jdbc.username=sonar

sonar.jdbc.password=sonar

sonar.jdbc.url=jdbc:mysql://localhost:3306/sonar?useUnicode=true&characterEncoding=utf8&rewriteBatchedStatements=true&useConfigs=maxPerformance

sonar.web.host=0.0.0.0

sonar.web.context=

sonar.web.port=9000

sonar.host.url=http://localhost:9000

配置显示中文把下载好的中文包放到/usr/local/sonarqube-5.1.1/extensions/plugins/目录下

如果未来想使用maven自动化部署并sonarqube检测项目配置maven的setting.xml这里我没使用这种方法，但是配置我也贴上，以应对未来的需求

vi /usr/local/maven/conf/setting.xml在文件末尾追加

<profiles>

<profile>

<id>sonar</id>

<activation>

<activeByDefault>true</activeByDefault>

</activation>

<properties>

<sonar.jdbc.url>

jdbc:mysql://localhost:3306/sonar?useUnicode=ture&amp;characterEncoding=utf8

</sonar.jdbc.url>

<sonar.jdbc.driver>com.mysql.jdbc.Driver</sonar.jdbc.driver>

<sonar.jdbc.username>sonar</sonar.jdbc.username>

<sonar.jdbc.password>sonar</sonar.jdbc.password>

</profile>

</profiles>

启动sonarqube

/usr/local/sonarqube-5.1.1/bin/linux-x86-64/sonar.sh

配置开机自动启动

vi /etc/rc.d/rc.local

在最下面追加 /usr/local/sonarqube-5.1.1/bin/linux-x86-64/sonar.sh

访问方式：

<http://ip:9000/>

1. 安装jenkins

有两种方式安装jenkins

第一种 去官网下载jenkins.war

1. 环境变量的设置：
2. JAVA\_HOME，设置JDK的安装目录，建议采用JDK5
3. ANT\_HOME，设置Ant的目录，建议采用ant最新版本1.8
4. JENKINS\_HOME，设置Jenkins的配置文件目录，默认为用户的目录，建议为Jenkins的安装目录，便于控制；
5. PATH，需要将java、ant的目录配置到path目录下；
6. CLASSPATH，需要将JDK的tools.jar配置到classpath目录中；
7. 其他外部条件
8. 需要WebLogic923的远程控制的jar文件，文件有weblogic\_sp.jar、weblogic.jar、webservices.jar三个jar文件，这三个文件位于WebLogic安装目录的相对目录server\lib下；
9. 将上述三个jar配置到系统的CLASSPATH环境变量中；

Jenkins可以有两种方法进行运行Jenkins；

* 1. 直接通过命令行

转到Jenkins.war所在的目录，当前为D:\Jenkins

java -jar Jenkins.war --httpPort=8080 –prefix=Jenkins

说明：httpPort为Jenkins运行的端口，默认端口为8080，上述命令其实Jenkins运行在Winstone容器中；

* 1. 在Web容器中运行

Jenkins可以运行在标准的Web服务器中，支持Tomcat、Jboss、WebLogic中，只需要将Jenkins.war放置到相应目录，启动服务就可以进行访问；

验证Jenkins是否运行正常，通过访问[http://hostIP:port/Jenkins](http://hostIP:port/hudson)即可。

我的jenkins路径：<http://192.168.8.26:8080/>

第二种 安装方式 使用rpm安装

wget –O /etc/yum.repos.d/Jenkins.repo <http://pkg.jenkins-ci.org/redhat/jenkins.repo>

rpm --import <http://pkg.jenkins-ci.org/redhat/jenkins-ci.org.key>

安装jenkins

yum install –y Jenkins

启动服务

service Jenkins start

添加开机自动启动

chkconfig Jenkins on

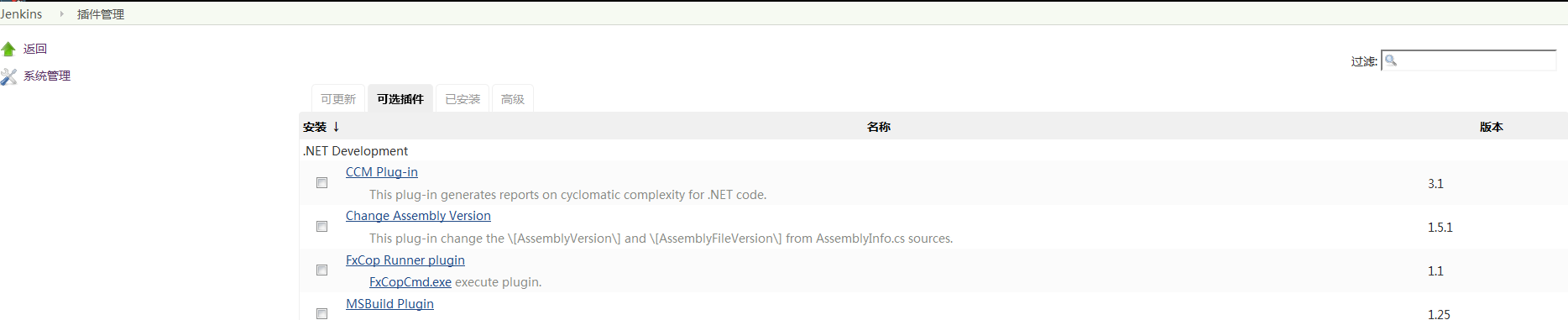
访问http://ip:8080

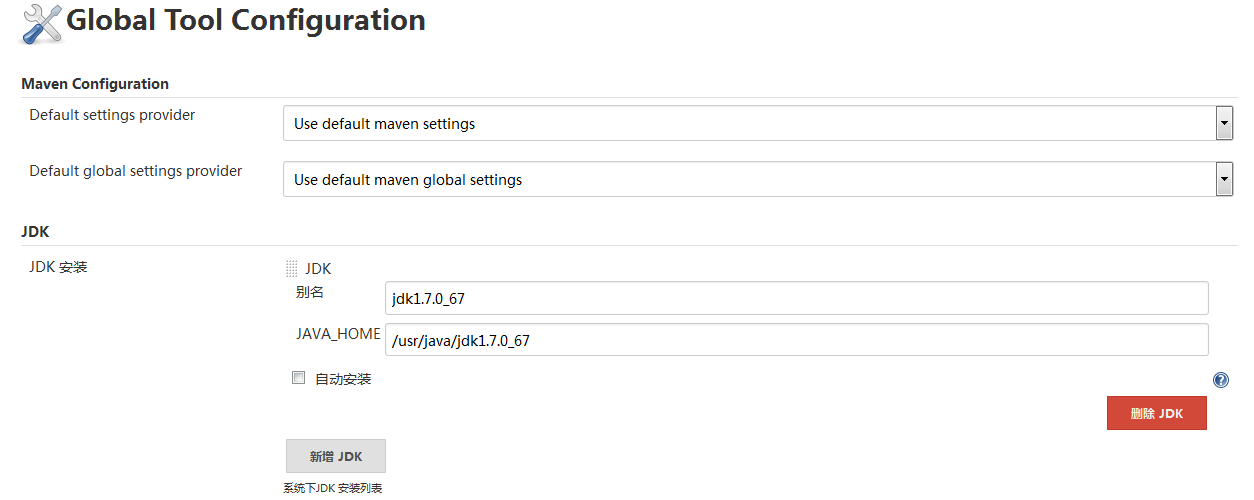


用户名 root 密码 root1234



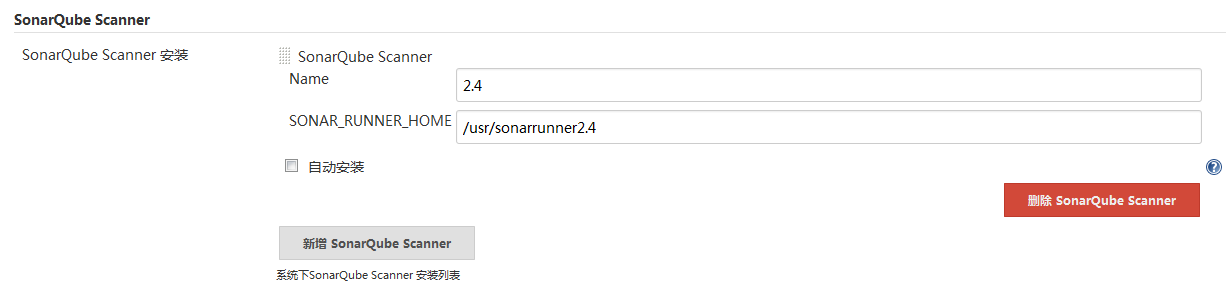
登录后先添加准备用到的项目插件 点击系统管理管理插件



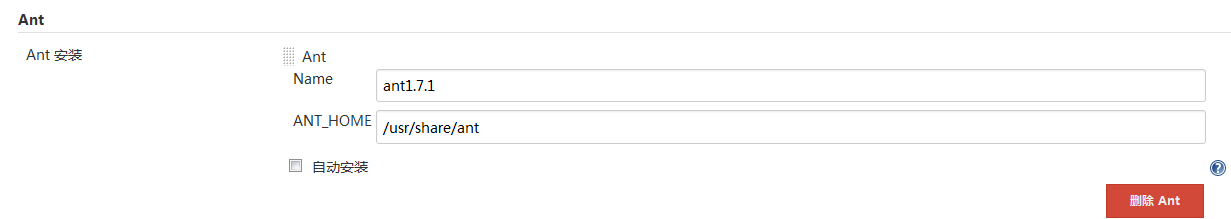


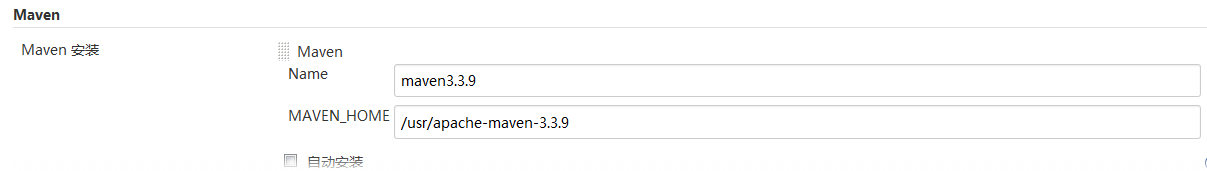
选择要安装的插件： Ant Plugin Publish Over SSH SonarQube Plugin Subversion Plug-in 我是登录后把默认的插件都安装了，上面那几个在配置过程中使用到了后安装的，系统里面安装了，但是jenkins界面里面没有响应的选项

为什么配置这个环境变量；在测试过程中发现 如果不添加这个 Jenkins默认自动查找系统中存在的环境变量，但是在ant的时候 如果不配置不找系统自己安装的路径，因为jenkins界面也有一些ant svn等等插件的安装，在测试调用中发现jenkins使用插件时的路径和安装的路径不同，故都手动设置系统的环境变量



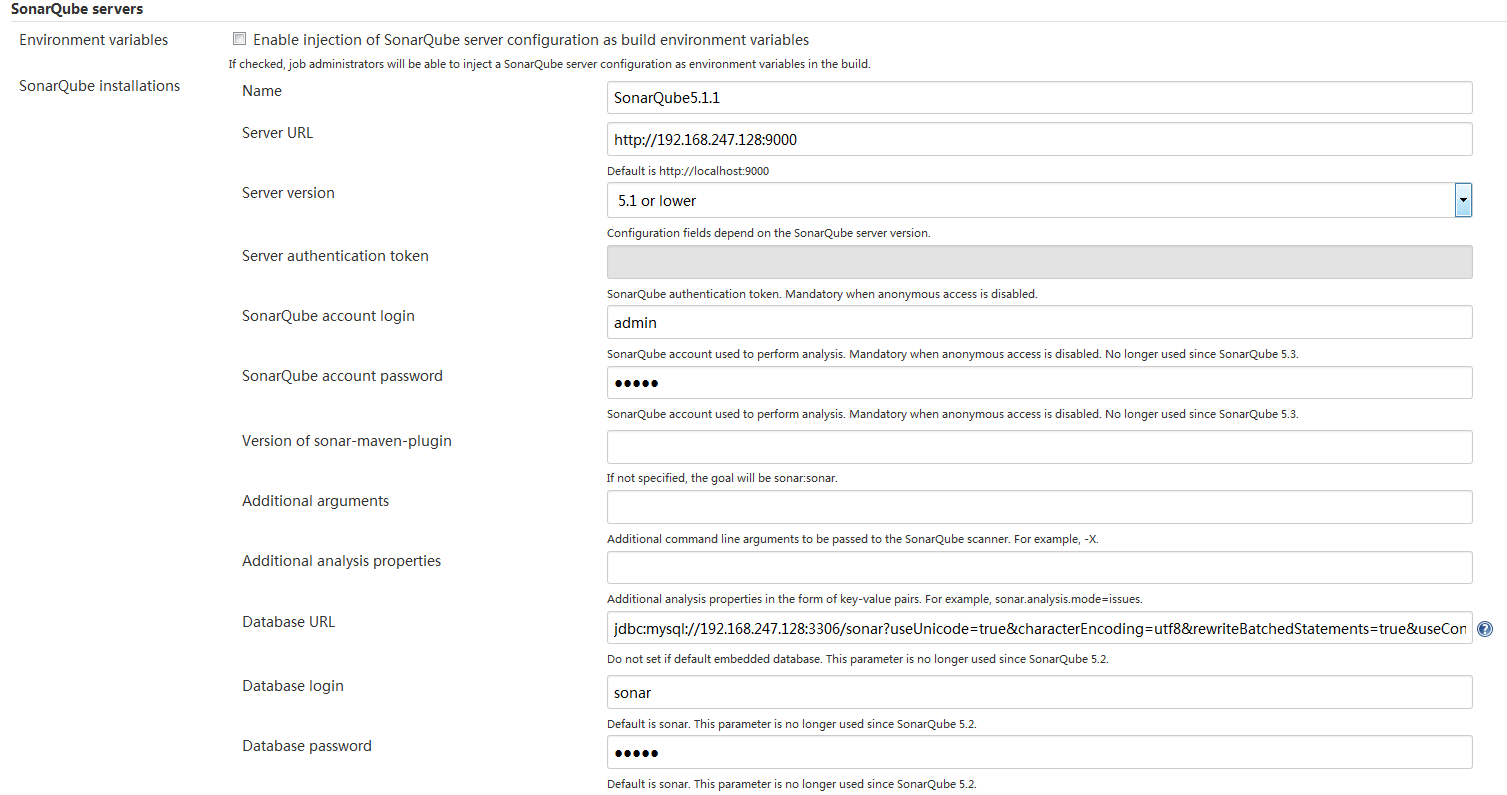
这个暂时我还没有安装，这是使用外部命令生成代码质量检测时应用的软件及路径

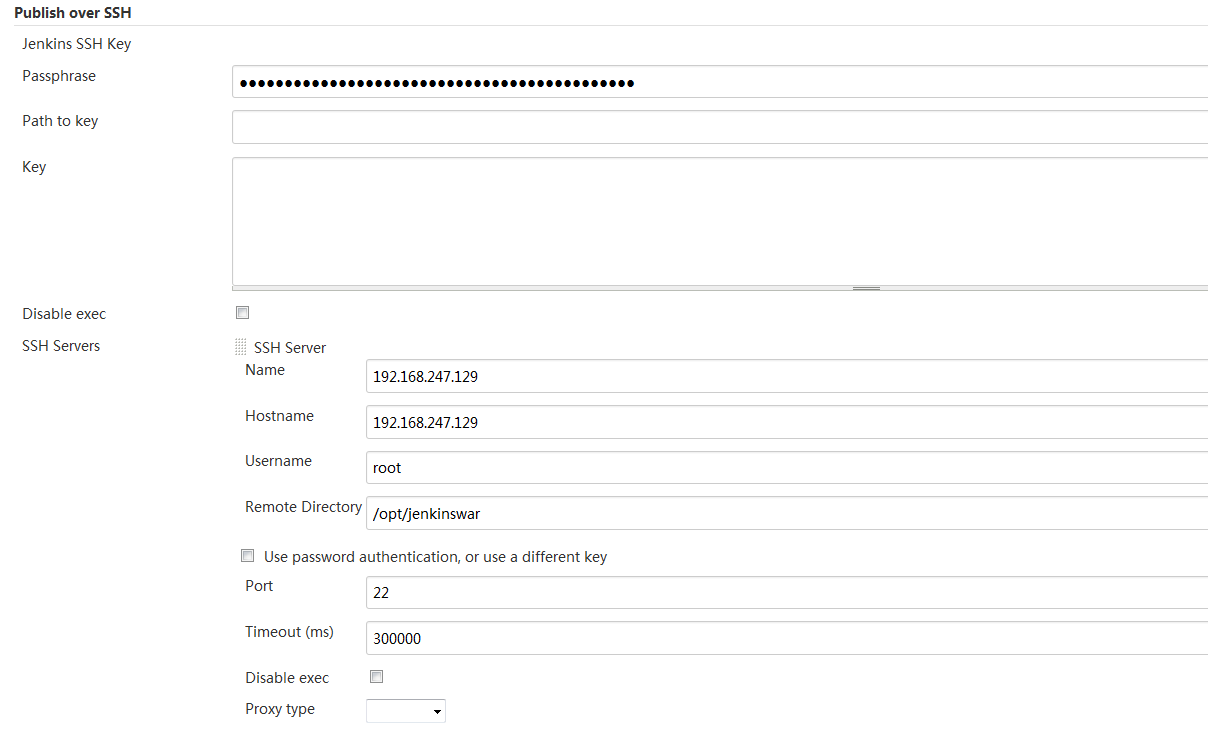




点击保存应用

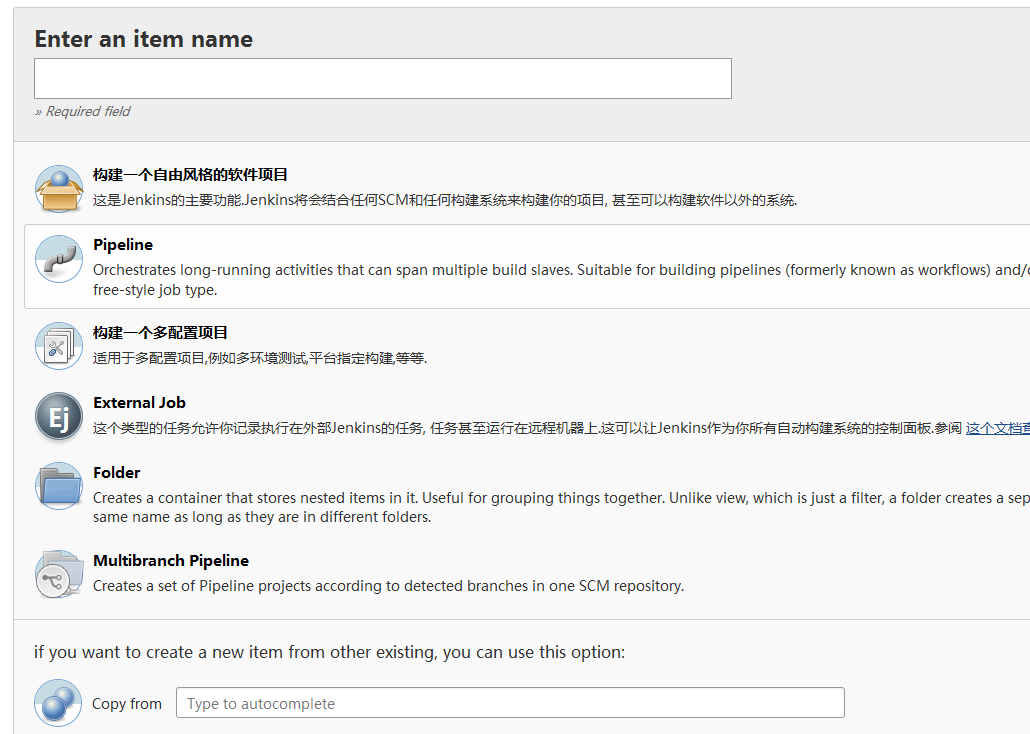
系统配置





基础的环境设置了好以后 下面我们创建一个项目

单击新建

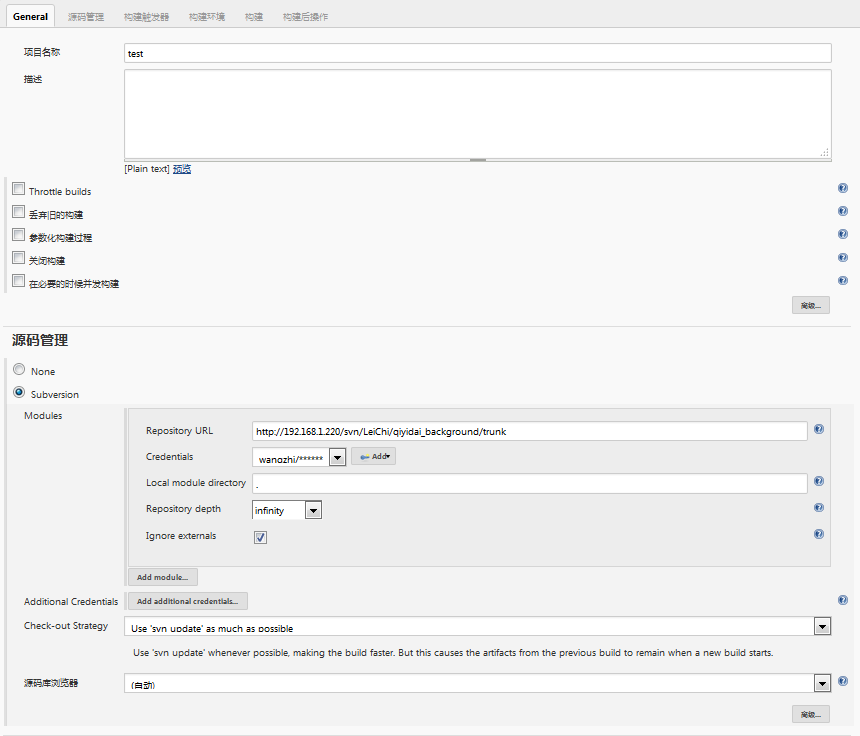


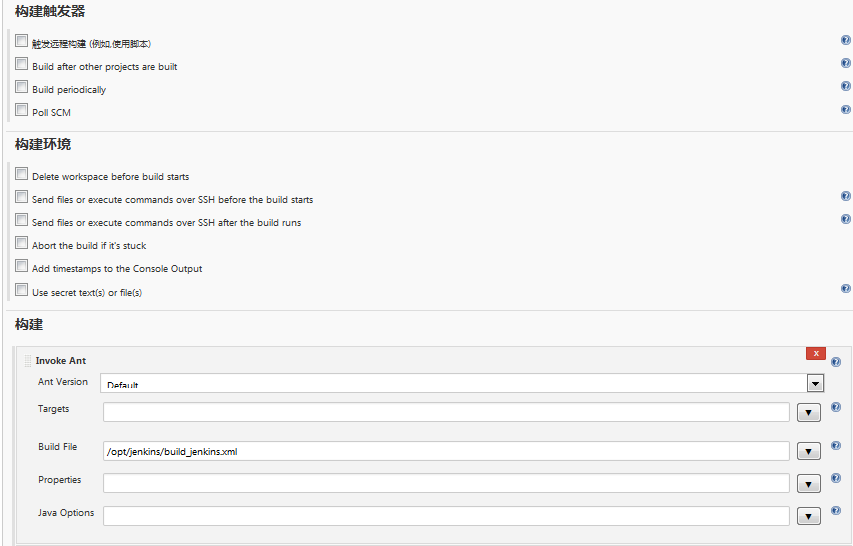


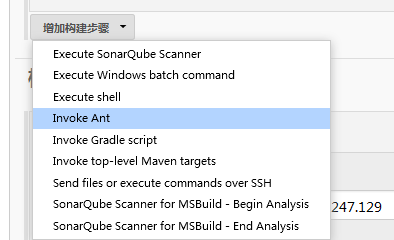
新建一个test项目



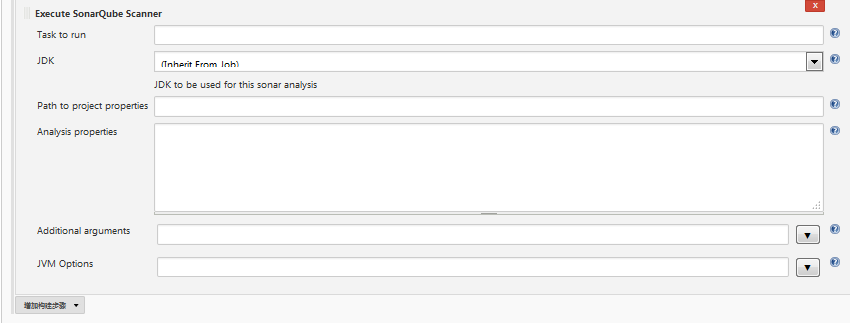
点击配置

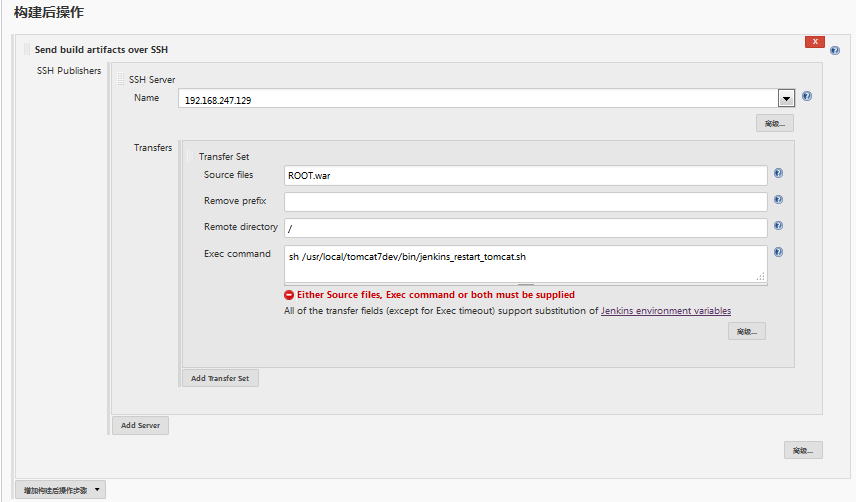






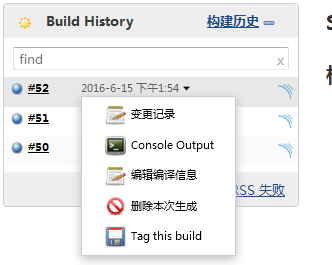






然后点击立即构建





生成

然后点击console output

查看项目生成输出

ant脚本

<?xml version="1.0" encoding="UTF-8"?>

<project name="JenkinsAutoBuild" default="war" basedir=".">

<!--定义变量-->

<property name="project.dir" value="/var/lib/jenkins/workspace/test" />

<property name="resources.dir" value="${project.dir}/resources" />

<property name="src.dir" value="${project.dir}/src" />

<property name="lib.dir" value="${project.dir}/WebRoot/WEB-INF/lib" />

<property name="classes.dir" value="${project.dir}/WebRoot/WEB-INF/classes" />

<!--构建目录-->

<property name="build.dir" value="${project.dir}/ROOT" />

<property name="build.war" value="${project.dir}"/>

<!--打包后的名-->

<property name="webapps.name" value="ROOT" />

<property name="web.root" value="${project.dir}/WebRoot" />

<property name="web.WEB-INF" location="${web.root}/WEB-INF" />

<property name="web.lib" location="${web.WEB-INF}/lib" />

<!-- 定义编译时的classpath -->

<path id="compile.path">

<fileset dir="${web.lib}" includes="\*\*\*.jar">

</fileset>

</path>

<target name="init" description="初始化">

<echo>初始化工作结束！</echo>

</target>

<target name="compile" depends="init" description="编译">

<javac includeantruntime="false" srcdir="${src.dir}" destdir="${classes.dir}" encoding="UTF-8">

<src path="${src.dir}"/>

<compilerarg line="-XDignore.symbol.file"/>

<classpath refid="compile.path" />

</javac>

<copy todir="${classes.dir}">

<fileset dir="${resources.dir}" excludes="\*\*/\*.java"/>

</copy>

<echo message="编译完成！" />

</target>

<target name="war" depends="compile" description="打包war文件">

<war destfile="${build.war}/${webapps.name}.war">

<fileset dir="${web.root}" includes="\*\*/\*.\*" />

<lib dir="${web.lib}" />

<webinf dir="${web.WEB-INF}" />

<classes dir="${classes.dir}" />

</war>

<echo>打包完成！</echo>

</target>

</project>

sonarqube 检测脚本

sonar.projectKey=javaweb

sonar.projectName=javaweb

sonar.projectVersion=1.0

# path to source directories (required)

sonar.sources=src

# path to test source directories (optional)

#sonar.tests=testDir1,testDir2

# path to project binaries (optional), for example directory of Java bytecode

#sonar.binaries=binDir

# optional comma-separated list of paths to libraries. Only path to JAR file and path to directory of classes are supported.

sonar.java.libraries=WebRoot/WEB-INF/lib/\*.jar

sonar.java.binaries=WebRoot/WEB-INF/classes/

# Uncomment this line to analyse a project which is not a java project.

# The value of the property must be the key of the language.

sonar.language=java

# Additional parameters

sonar.my.property=value

sonar.scm.disabled=true

远程启动tomcat

脚本

#!/bin/sh

#defined

rq=`date '+%Y%m%d%H%M%S'`

source /etc/profile #鍔犺浇JAVA\_HOME

TOMCAT\_HOME="/usr/local/tomcat7dev"

ID=`ps -ef | grep java | grep tomcat|awk '{print $2}'`

echo $ID

echo "remover war file"

rm -rf $TOMCAT\_HOME/webapps/\*

echo "kill tomcat"

cd "$TOMCAT\_HOME"/bin

./shutdown.sh

kill -9 $ID

echo "copy war to webapp"

cp /opt/jenkinswar/ROOT.war "$TOMCAT\_HOME"/webapps/

###### bakcup last 5 version war #####

mv /opt/jenkinswar/ROOT.war /opt/jenkinswar/ROOT$rq.war

###### delete backup war #####

file\_sum=`ls /opt/jenkinswar/ROOT\*.war|wc -l`

if [ $file\_sum -gt 5 ]

then

delete\_file=$(($file\_sum-5))

ls -rt /opt/jenkinswar/ROOT\*.war |head -n $delete\_file|xargs rm -f

fi

echo "start tomcat"

$TOMCAT\_HOME/bin/startup.sh

## Ant编译报com.sun.image.codec.jpeg does not exist 解决方法



项目开发中在对图片进行裁切处理的时候，有时候是会使用到 com.sun 包下的类时，

如果项目使用ant编译，会出现错误 com.sun.image.codec.jpeg does not exist 这是因为在JDK1.7+时，Oracle不允许使用sun.\*的jar

具体参见http://www.oracle.com/technetwork/[**Java**](http://lib.csdn.net/base/17)/faq-sun-packages-142232.html 。

项目代码已经写好，且直接运行可以正常使用，只是使用ant编译会出现错误，现在不打算更换项目的具体实现代码，不能更换JDK版本，所以做如下处理即可：

**我们可以指定javac 参数来忽略这样的错误，如下：**

<javac includeantruntime="false" debug="true" debuglevel="${debuglevel}" destdir="${build.dir}/WEB-INF/classes" source="${source}" target="${target}" encoding="UTF-8">  
 <src path="${src.dir}" />     
 **<compilerarg line="-XDignore.symbol.file"/>**  
 <classpath refid="Project.classpath" />  
 </javac>