**Jenkins持续集成**

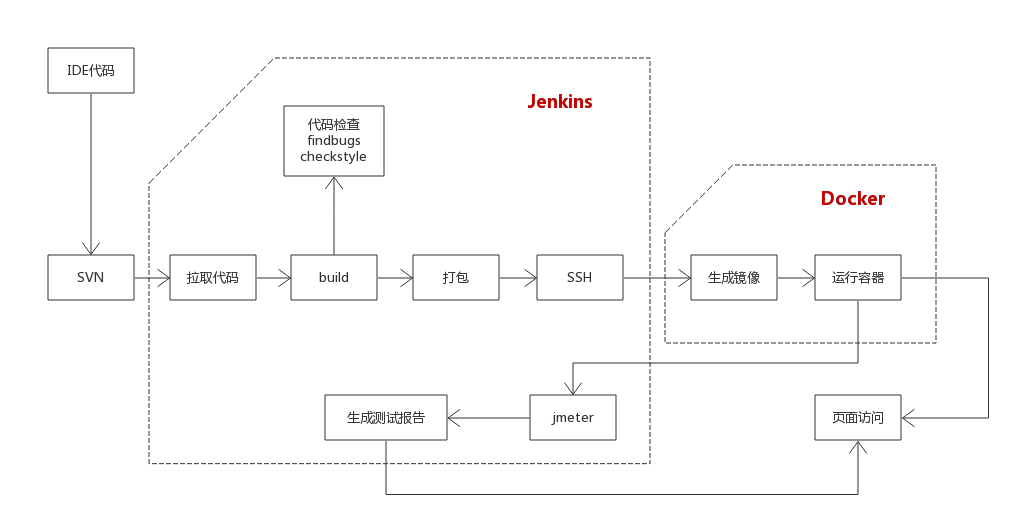
# Jenkins简介

开发者的工作大概流程是，eclipse编写代码，提交代码到svn，然后进行编译、打包、测试、部署、发布。

Jenkins是基于Java开发的一种持续集成工具，用于监控持续重复的工作，功能包括：

1. 持续的软件版本发布/测试。
2. 项目监控外部调用执行的工作。

# 自动化流程

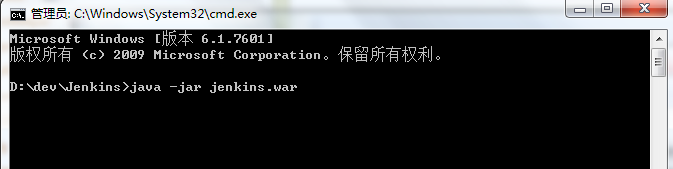
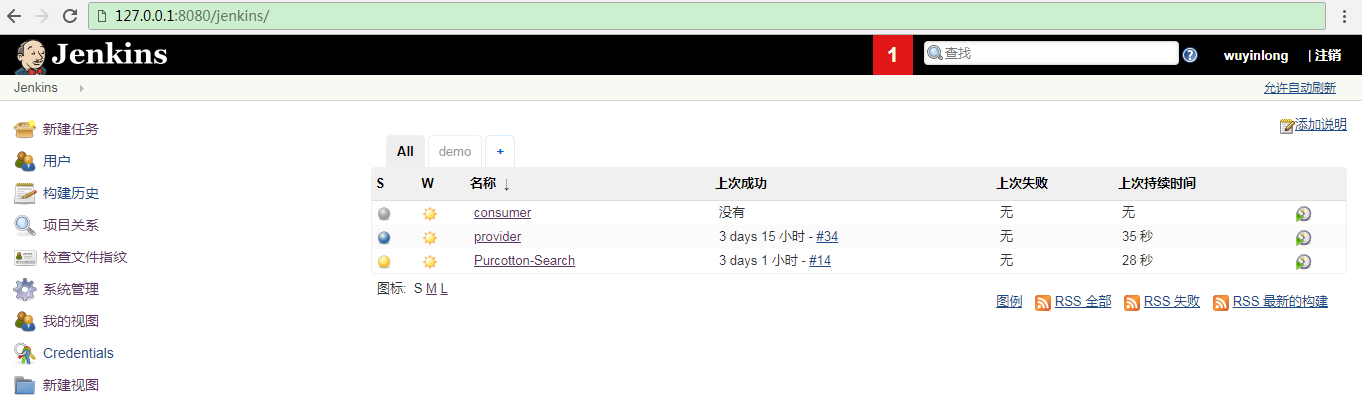
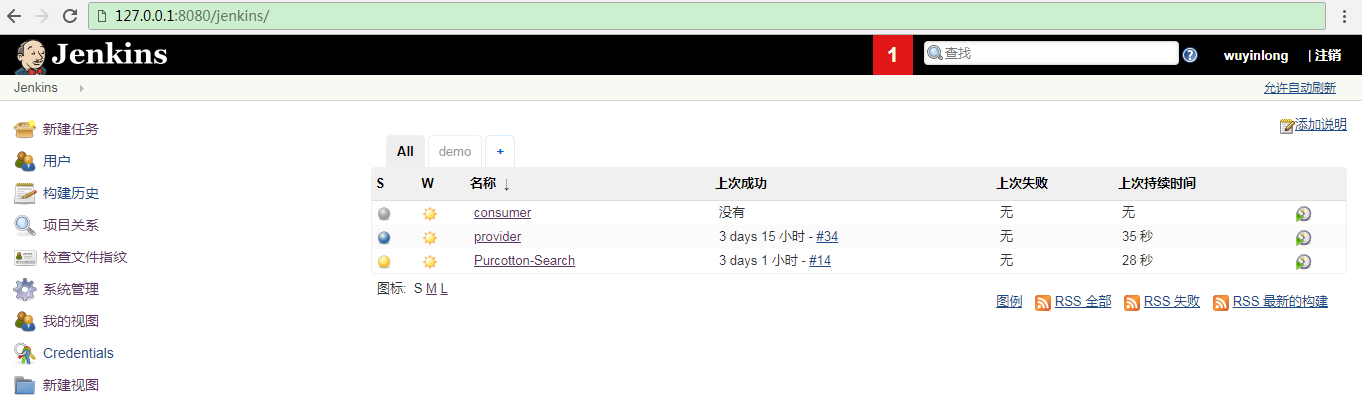


1. 编写/修改工程代码
2. 代码提交配置库（SVN）
3. Jenkins自动检测SVN代码是否更新，并拉取最新代码
4. Jenkins自动编译，编译完成后，可进行代码检查
5. Jenkins自动打包（打war包、可直接运行的jar包）
6. Jenkins自动上传包（war/jar）到装要docker环境的主机
7. 根据dockerfile文件构建docker镜像，映射端口
8. 启动docker容器
9. Jenkins自动检测应用程序是否启动成功，并执行jmeter自动化测试脚本
10. Jenkins界面生成自动化测试报告
11. 浏览器访问界面

# 环境准备

## Jenkins

Jenkins简单点说就是一个war包。一个自动化服务，提供各种插件集成。Jenkins部署有两种方法（可以装到windows或者linux，本文中以linux的方式一为例）：

1. 方式一：  
   直接运行java –jar jenkins.war（注意jenkins.war得是绝对路径，例如下图所示），在浏览器中输入http://ip:8080即可。  
     
   
2. 方式二：  
   把jenkins的war包拷贝到tomcat的webapps目录下，启动tomcat，访问地址：<http://ip:8080/jenkins>  
   

## JDK

必装，建议版本1.8及以上

## SVN

用于获取工程源码

## Maven

用于支持maven 项目

## Jmeter

用于执行jmx自动化测试脚本

## Ant

可用于辅助操作

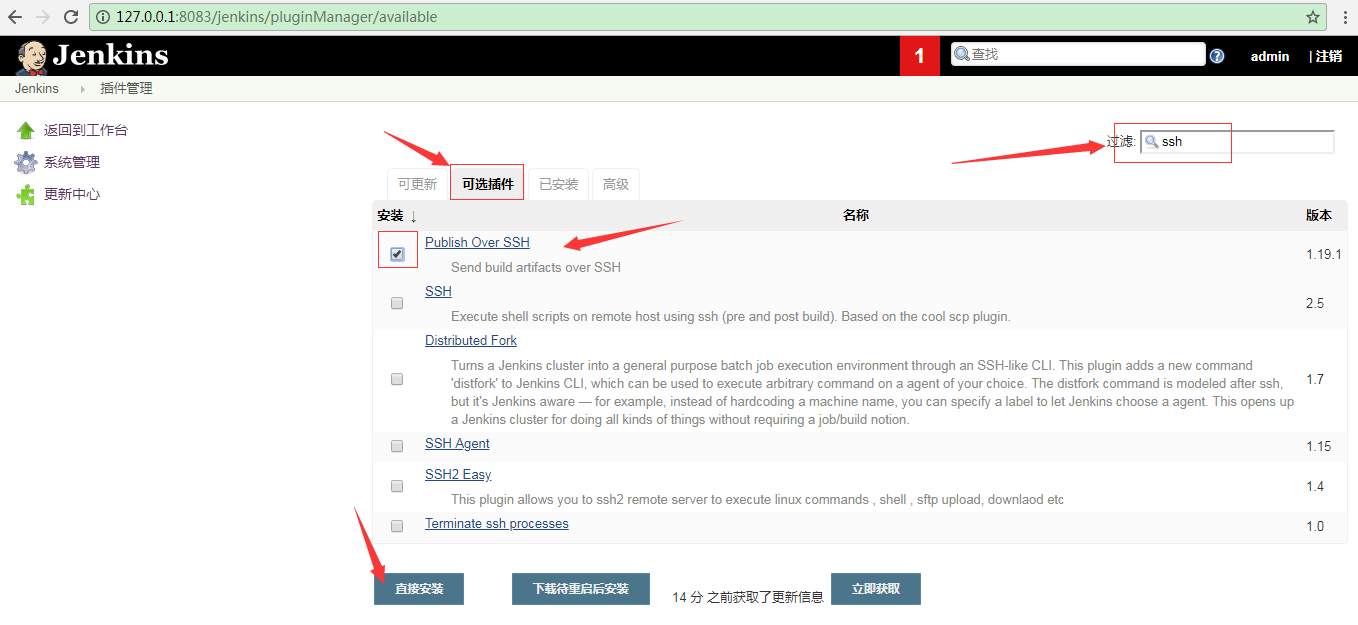
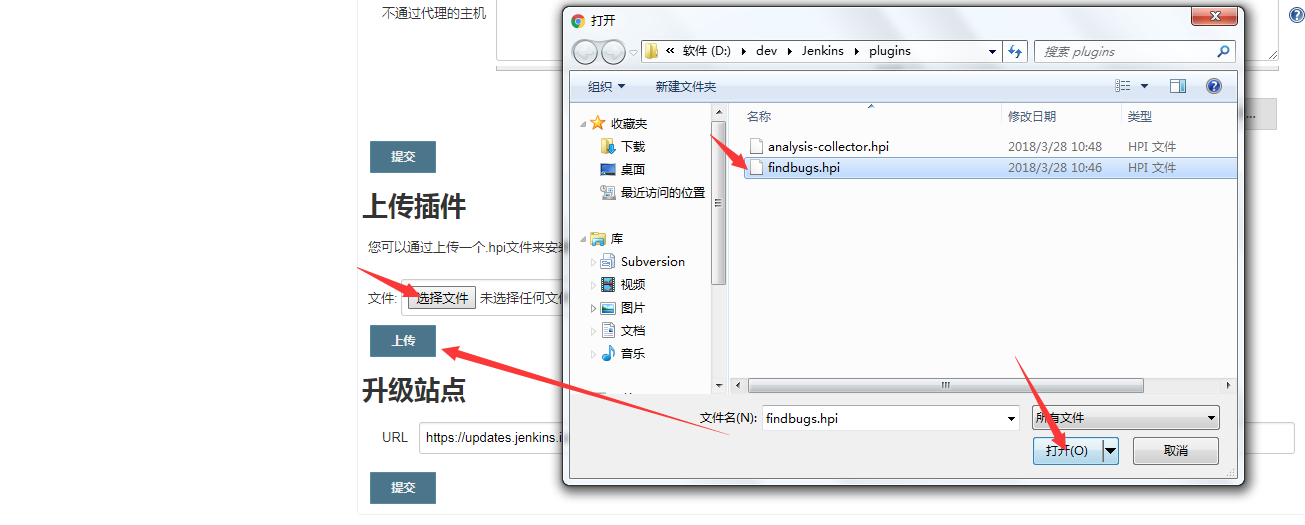
## Docker

用于制作镜像，运行容器

# 插件管理

插件管理可对插件安装、升级和卸载。支持两种方式进行插件安装，进入插件管理界面：

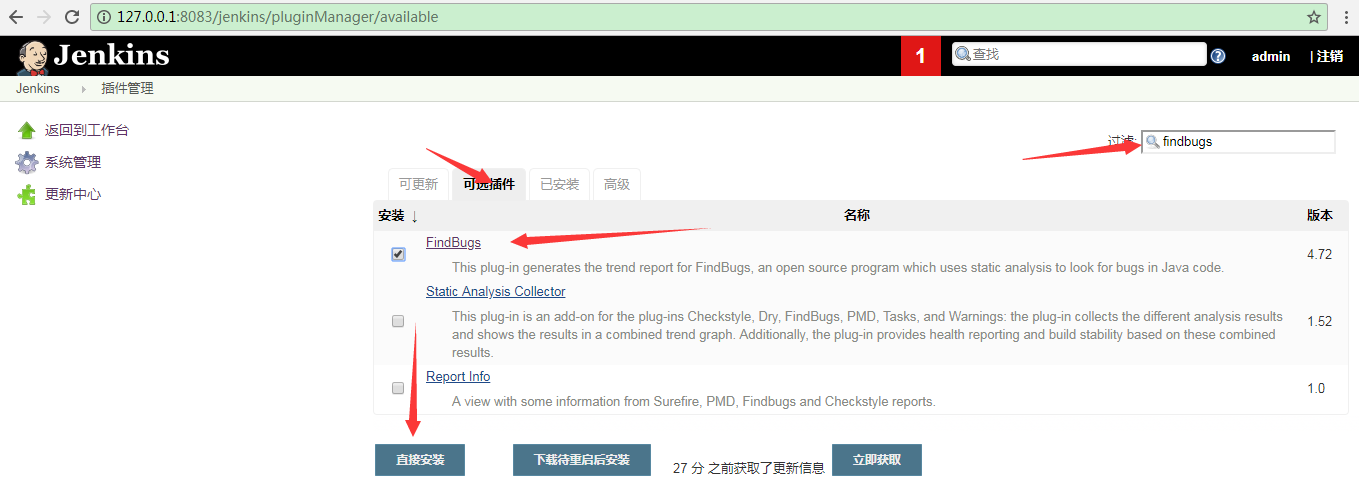


1. 方式一：在线安装，以安装ssh插件为例  
   
2. 方式二：插件包安装，以安装findbugs插件为例  
     
   

## SSH插件

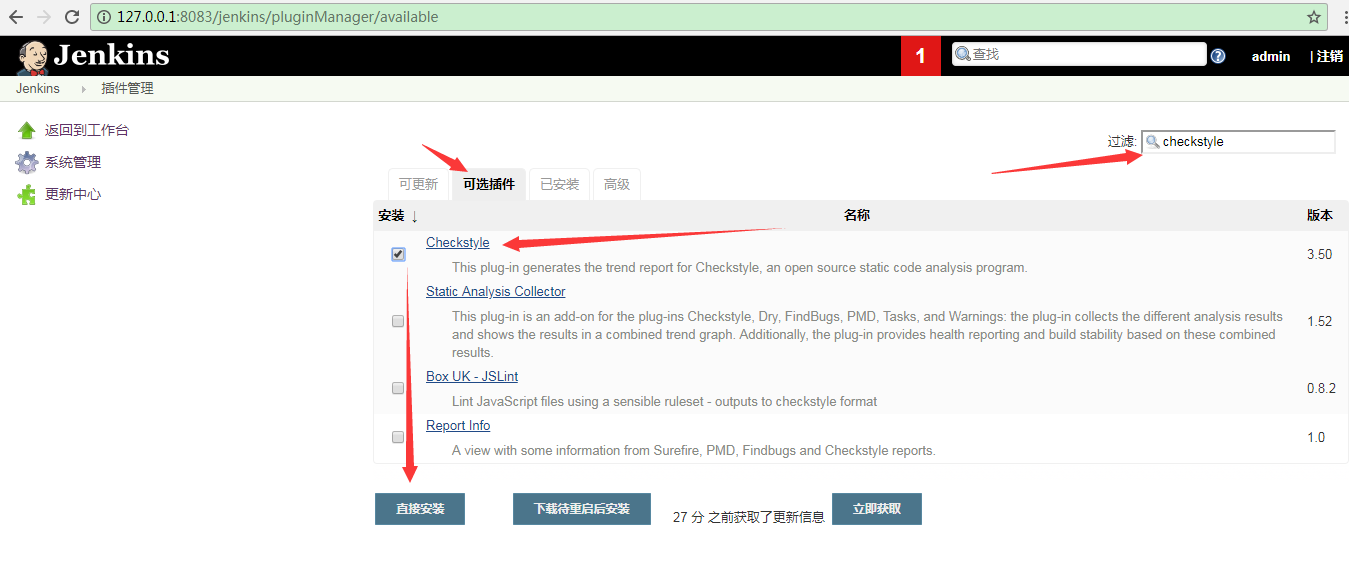
## Findbugs插件

Findbugs插件用于检查.class文件潜在的bug。



## Checkstyle插件

Checkstyle插件用于检查Java源代码是否与编码规范相符合。



## Analysis插件

Static Analysis Collector插件可将检查结果通过图表进行展示。

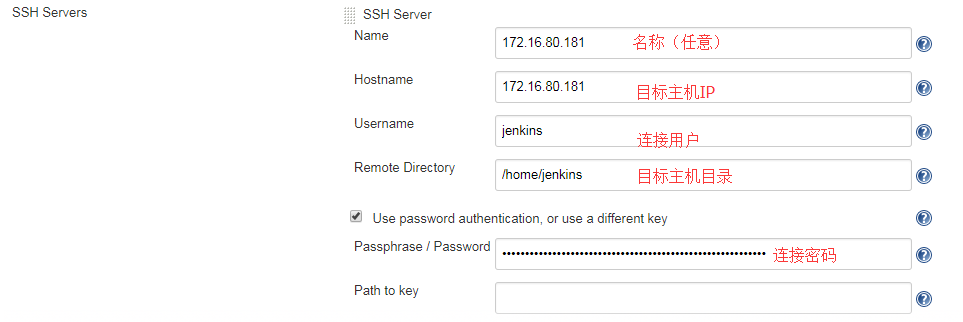


# 基础配置

## 系统设置

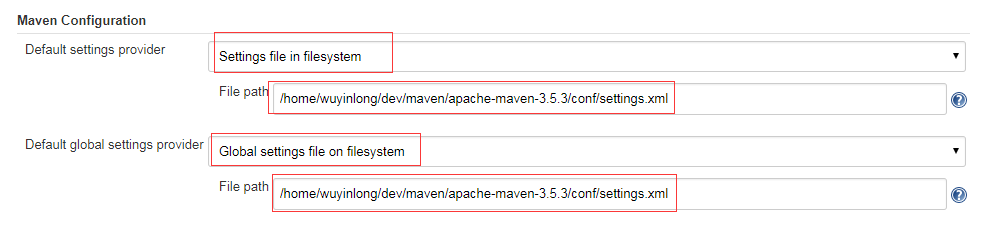
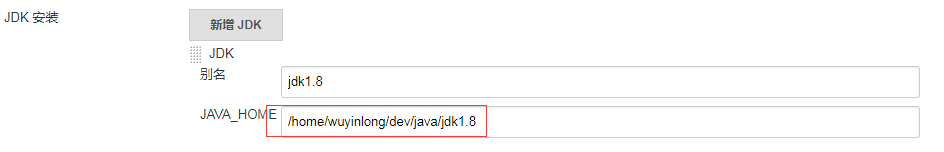
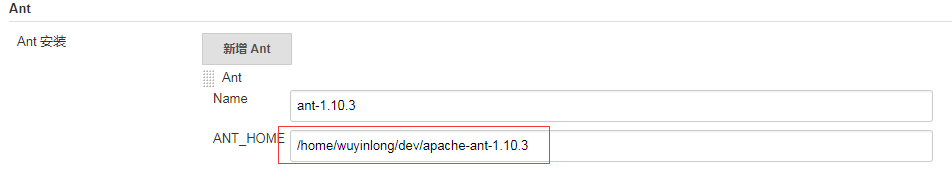
进入系统设置主界面



1. 配置jenkins location，及配置管理员邮箱。  
   
2. 配置SSH Servers。这里配置后才能在配置目标容器环境时，选择到构建完成后发布到的目标容器。  
   

## 全局工具配置



1. 配置Maven settings.xml  
   
2. 配置JDK  
   
3. 配置Ant  
   
4. 配置Maven  
   

# 配置自动化任务

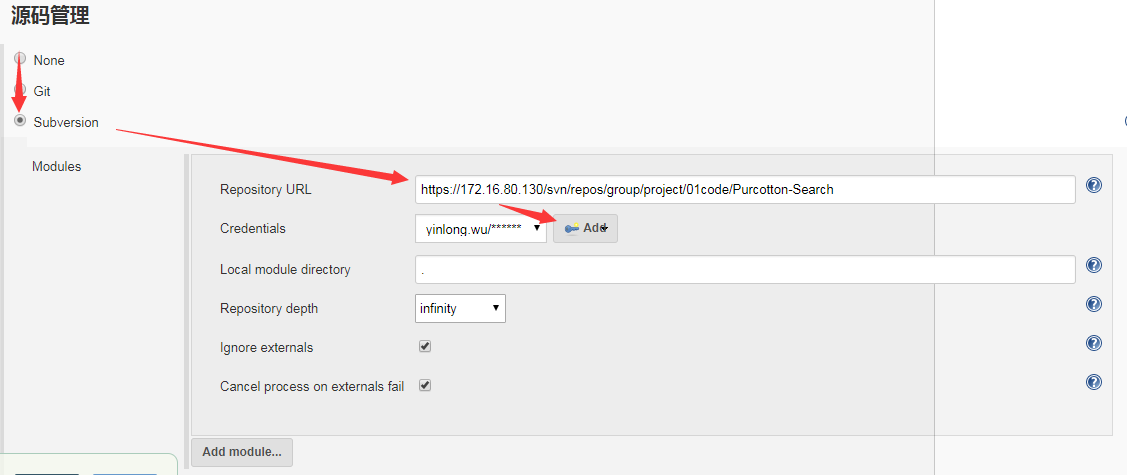
## 新建jenkins任务

点击界面左侧新建，项目名称可以随意命名，选择maven项目：



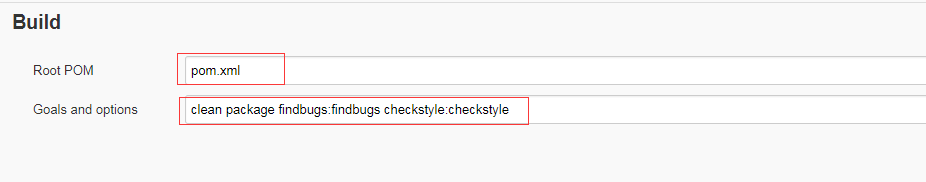
## 拉取SVN代码

选择源码地址，这功能可以让jenkins可以从svn上拉取代码。Add按钮添加SVN用户及密码。



## Build项目

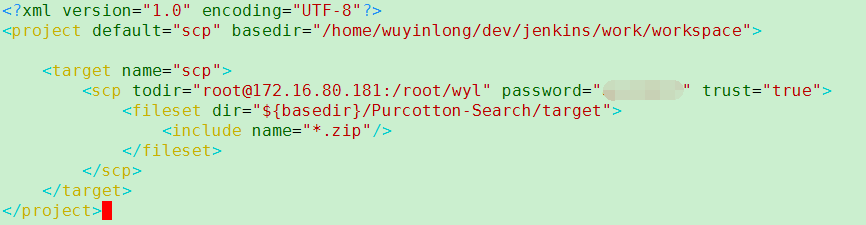
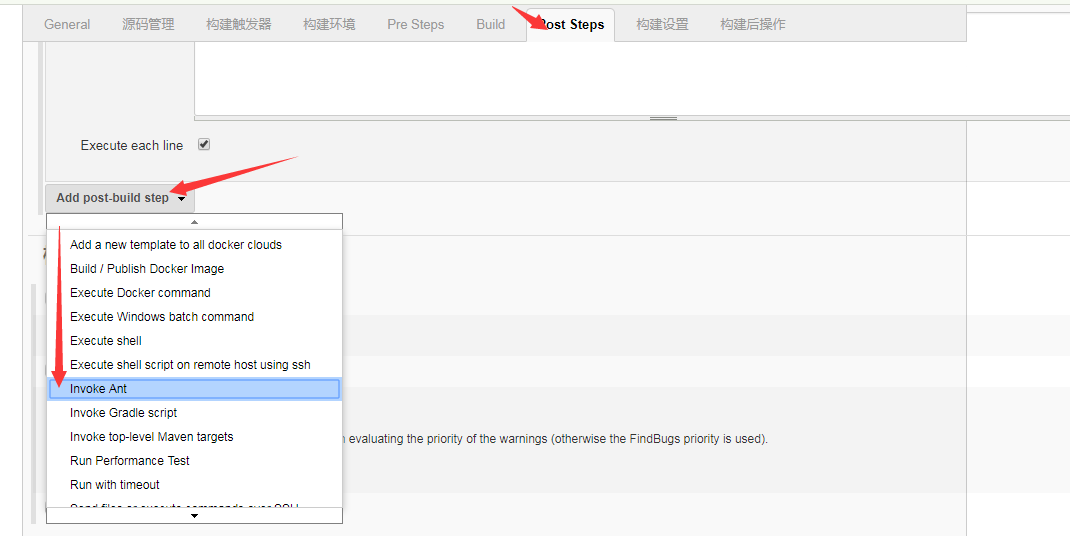
使用Maven命令进行项目编译，并添加findbugs、checkstyle命令，如



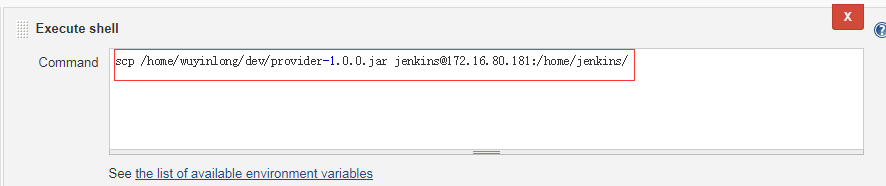
## 远程文件拷贝

远程文件拷贝，即将Jenkins编译好的工程包（war、jar）拷贝至指定服务器进行部署。有两种方式进行文件拷贝。

* 方式一：使用ant脚本

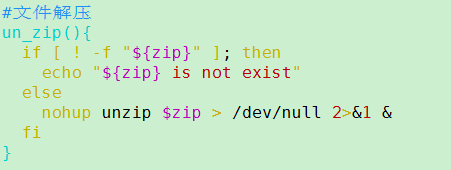
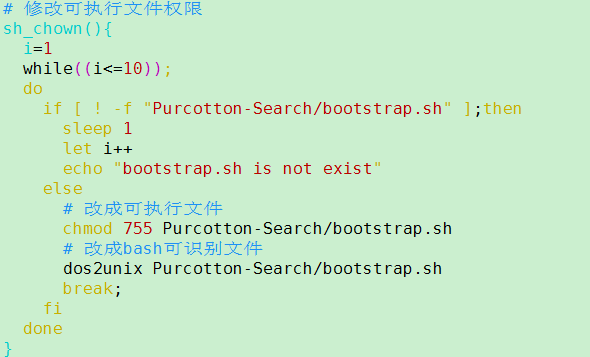
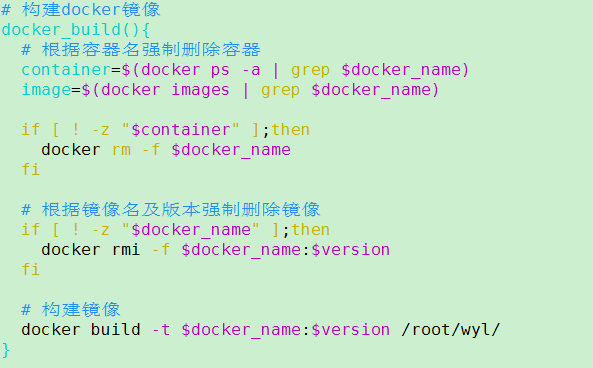
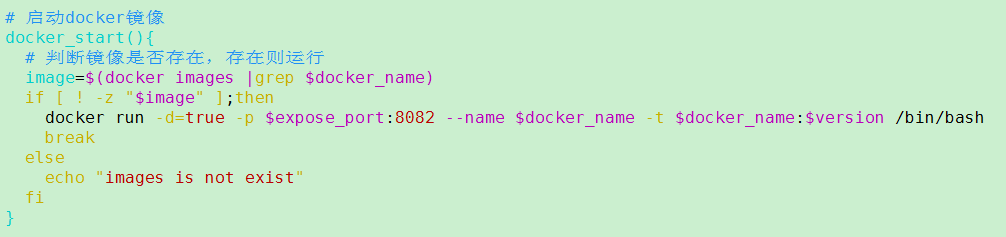
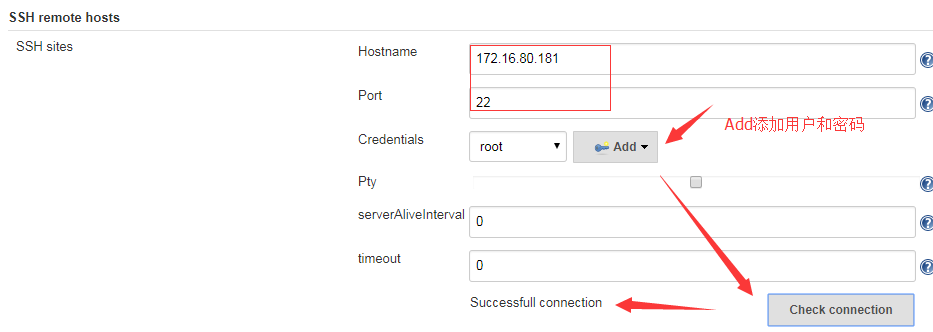
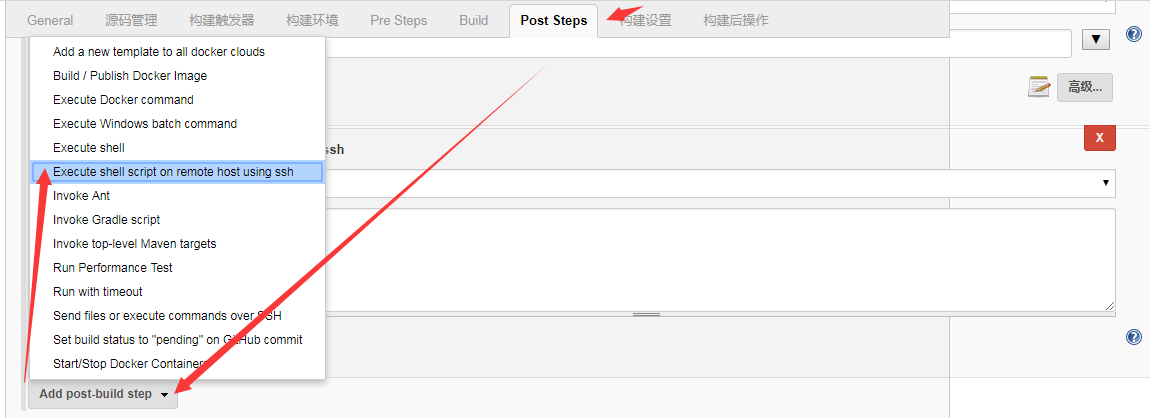
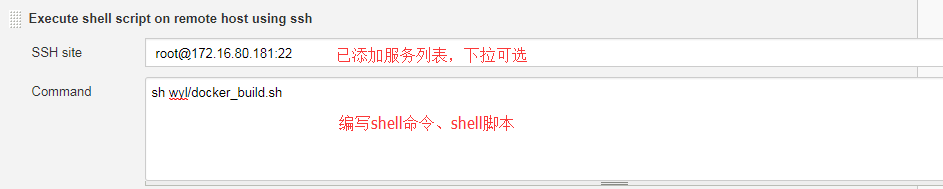
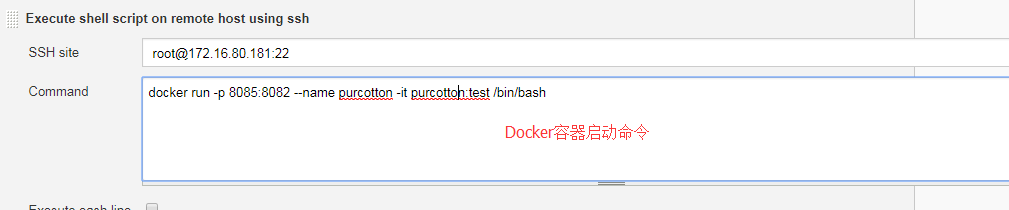
1. 编写ant脚本文件，basedir指定根目录，todir为目标服务器目录，fileset指定需要拷贝的文件。  
   
2. Jenkins任务流程添加Invoke Ant  
     
   

* 方式二：使用SSH scp命令

1. 首先保证服务器间已建立互信，即ssh免密登录
2. 添加任务Execute shell  
   

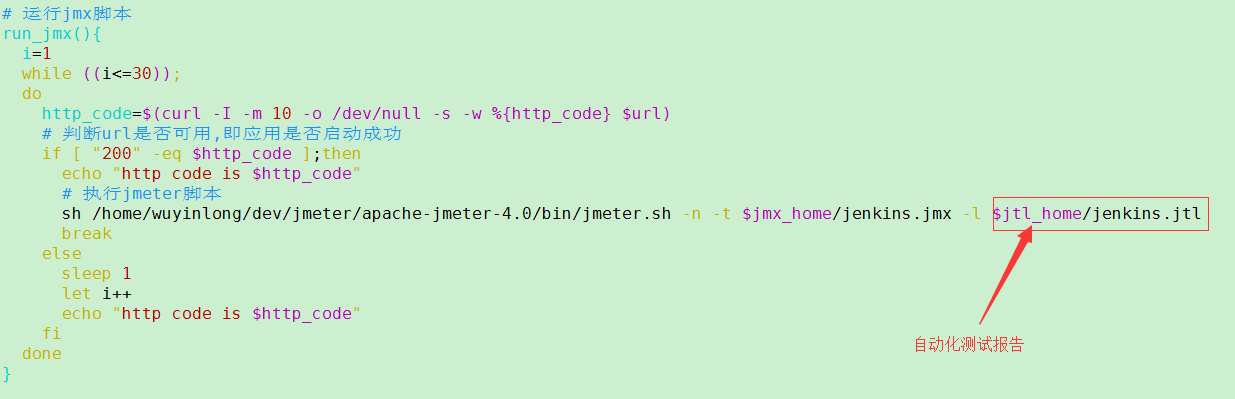
## 构建/启动docker镜像

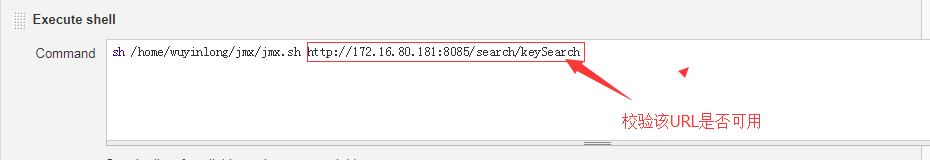
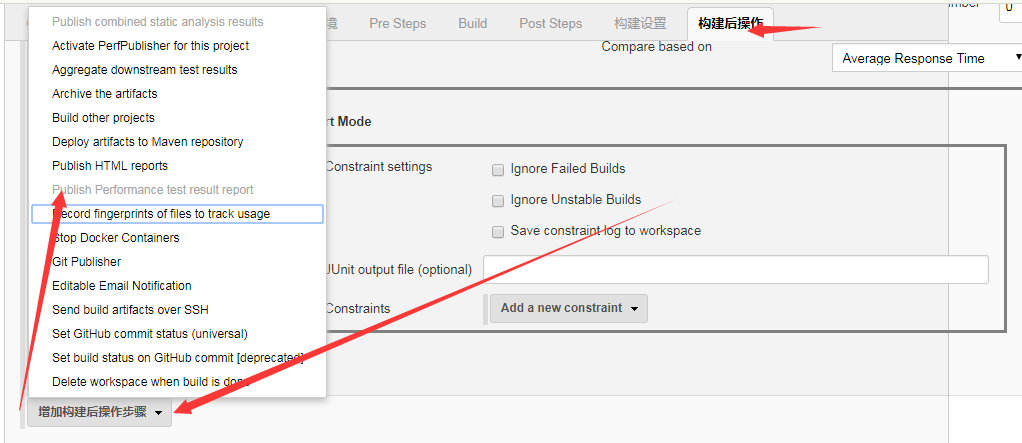
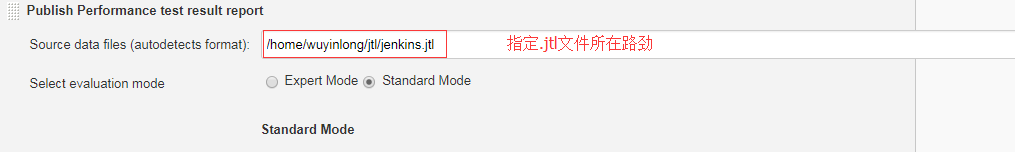
SSH到目标服务器，执行shell脚本

1. 编写shell脚本，脚本文件主要实现操作：解压zip包、修改包内.sh为可执行文件、制作docker镜像、运行镜像等。镜像名称与容器名称一至。  
     
     
     
   
2. Jenkins添加服务器，在 系统管理 –> 系统设置，找到SSH remote hosts  
   
3. Jenkins任务流程添加Execute shell script on remote host using ssh，用于执行shell命令和.sh脚本  
     
   
4. 多添加一个Execute shell script on remote host using ssh任务流程，用于启动docker容器  
   

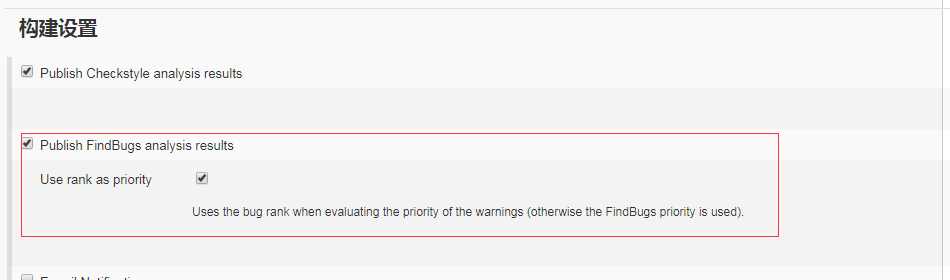
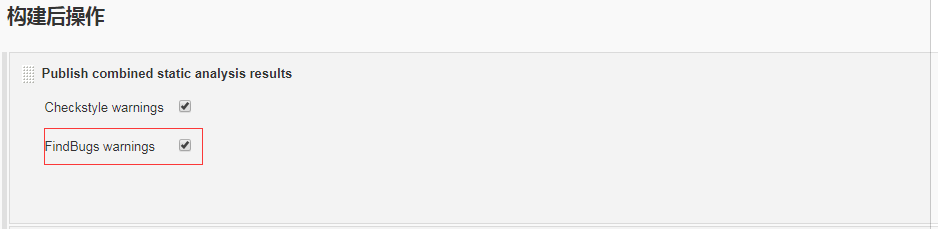
## 运行jmeter脚本

Docker容器启动后，及应用成功部署，Jenkins自动运行jmeter自动化脚本，在运行jmx脚本之前需先判断应用是否启动成功。如下shell脚本



1. 添加Execute shell，用于运行本地shell脚本  
   
2. 任务流程添加Publish Performance test result report，用于在Jenkins界面生成检查报告  
     
   

## 添加findbugs检查

* 勾选Publish FindBugs analysis results，检查结果显示于Jenkins页面  
  
* 勾选FindBugs warnings才能执行findbugs检查  
  

## 添加checkstyle检查

与findbugs检查相同配置

# Jenkins任务界面

测试环境地址：<http://172.16.80.181:18083/jenkins/job/Purcotton-Search/>

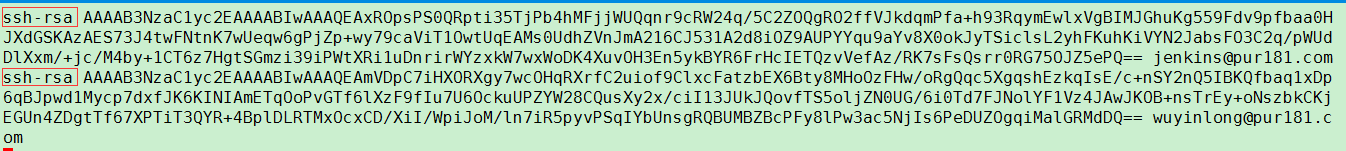
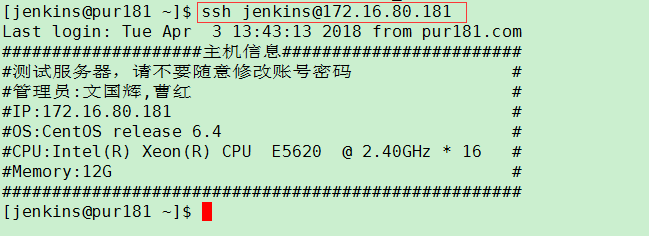
用户、密码：admin、admin123

# 脚本文件

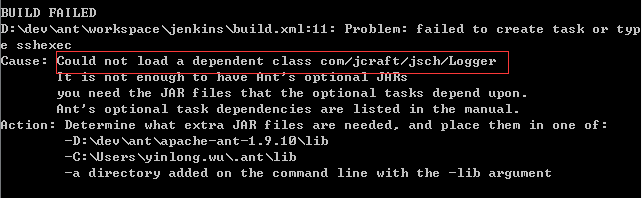
# 常见问题记录

## 服务器间建立互信

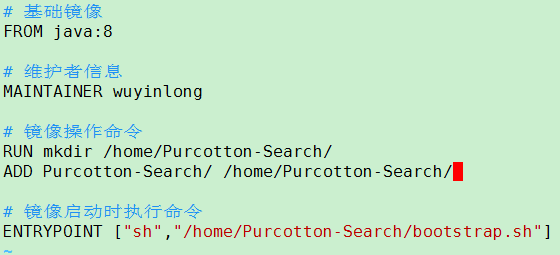
1. 登录服务器A
2. 建立ssh相关目录:：mkdir ~/.ssh
3. 设置ssh的权限：chmod 700 ~/.ssh
4. 生成RSA公钥：ssh-keygen -t rsa （请一直回车，直至提示结束）
5. 将公钥（名为id\_rsa.pub文件）追加到认证文件（名为authorized\_keys文件）中：cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys
6. 设置认证文件的权限：chmod 600 ~/.ssh/authorized\_keys
7. 至此服务器A秘钥配置成功。服务器B与服务器A相同操作
8. 将服务器A 的id\_rsa.pub内容追加到服务器B的认证文件authorized\_keys，服务器B与服务器A同样操作  
   
9. 互信成功后远程ssh无需输入密码，效果如下：  
   

## Ant脚本无法进行scp文件传输

缺少相应的jsch.jar包



## Docker操作命令

1. Dockerfile文件编写  
   
2. 强制删除镜像容器：docker rm –f dockername/dockerid（-f表示强制删除，容器未停止也可删除成功）
3. 强制删除镜像：docker rmi –f dockername/dockerid（容器未删除时，不允许删除镜像文件）
4. 构建镜像：docker build –t 镜像名:版本 （-t表示在控制台打印日志信息）
5. 运行镜像文件、创建镜像：docker run –d=true –p 18181:8080 –name docker –it镜像名:版本 /bin/bash（-d表示后台执行；-p端口映射；--name指定容器名称；-it表示生成伪终端；/bin/bash可以理解成Dockerfile里边的一个CMD命令，及容器启动后执行命令）

## Docker容器自动退出

Docker容器启动后执行完.sh脚本文件，容器自动退出，docker ps查看不到运行的容器。

解决方案：让docker run后台运行，即docker在容器后台运行，在run命令后添加

-d=true或-d，如下图：

