# 环境搭建

## 环境说明

受服务器和资源限制，本地测试环境基于Docker容器安装，镜像为CentOS Linux 发行版7.9.2009 （核心包），对应的Docker命令为：docker pull centos:centos7。 由于使用的是非完全版Linux，部分应用/命令需要手动安装。

## 软件列表

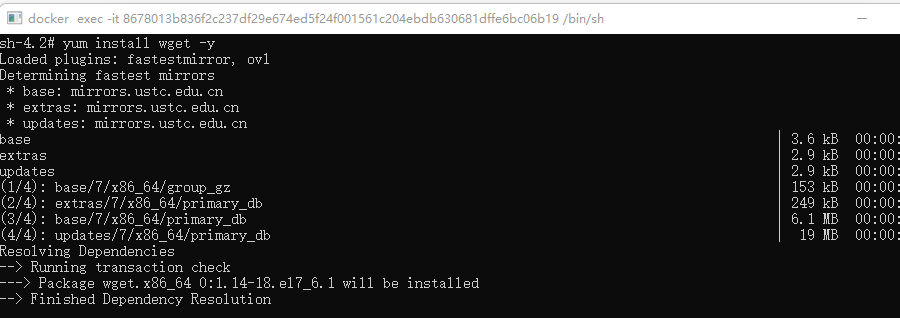
|  |  |
| --- | --- |
| **软件/列表** | **版本/备注** |
| java-1.8.0 | 代码构建, 项目基于Java 1.8 |
| java-11 | 最新版的Jenkins和Sonar要求Java 11+ |
| maven-3/3.6.3 | 3.6.3 |
| Jenkins | 2.361.1 |
| PostgreSQL | 9.2.24， |
| SonarQube | 9.8.0.63668 |
| sonar-scanner | 4.7.0. |
|  |  |

## 软件安装

### yum 和 wget

#### 安装wget，如果有则不需要装

yum install wget -y

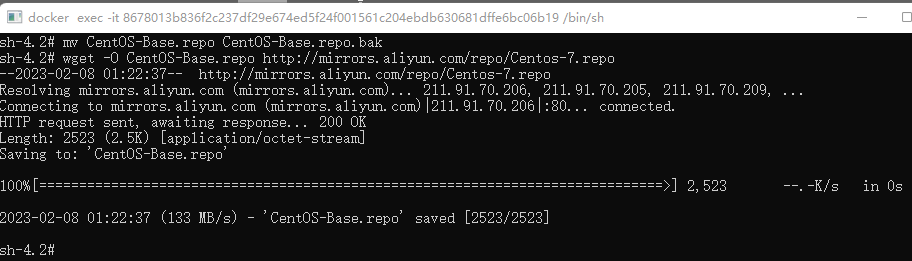


#### 修改yum源

cd /etc/yum.repos.d

mv CentOS-Base.repo CentOS-Base.repo.bak

wget -O CentOS-Base.repo <http://mirrors.aliyun.com/repo/Centos-7.repo>

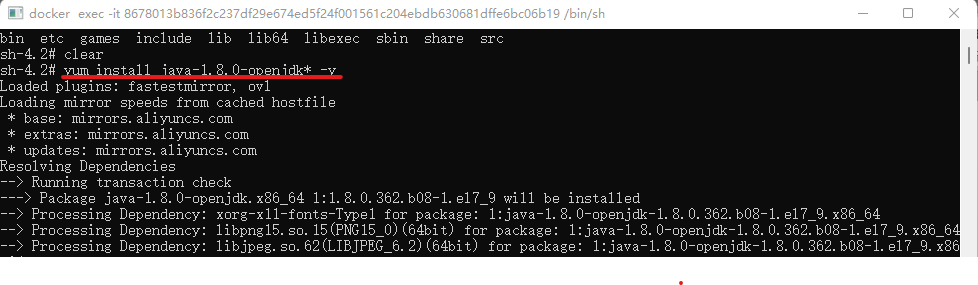


### JDK安装

#### JAVA 8 安装

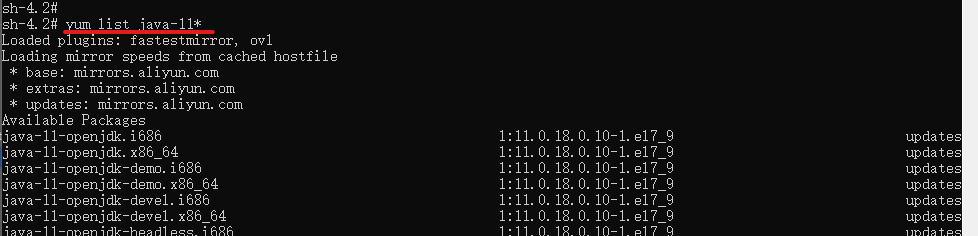
yum list java-1.8\*

yum install java-1.8.0-openjdk\* -y



#### JAVA 11安装

yum install java-11-openjdk\* -y



#### 设置环境变量，选择默认为JAVA 11

**暂时不需要设置。可使用下节4）提到的Java版本切换， 重启不会失效。**

# 在环境变量文件（/etc/profile）中加入

vi /etc/profile

# -----------------

export JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-11.0.18.0.10-1.el7\_9.x86\_64export JRE\_HOME=/usr/lib/jvm/jre-11-openjdk-11.0.18.0.10-1.el7\_9.x86\_64export PATH=$PATH:$JAVA\_HOME/bin:$JRE\_HOME/binexport CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar:$JRE\_HOME/lib

# -----------------

source /etc/profile

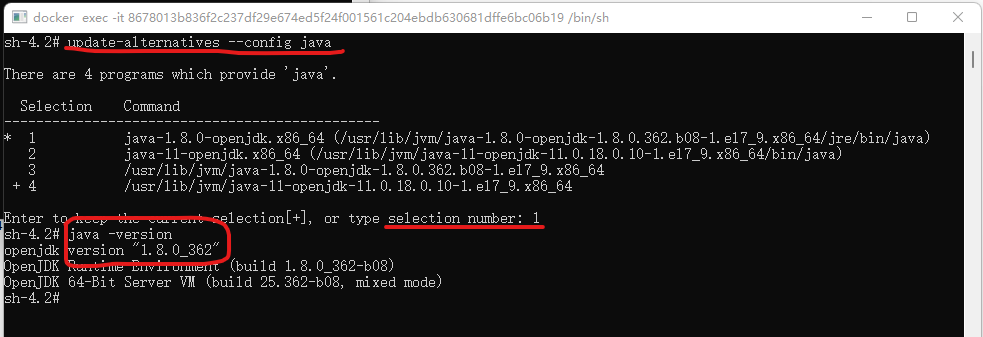
#### 设置JAVA版本切换

update-alternatives --install /usr/bin/java java /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.362.b08-1.el7\_9.x86\_64 8

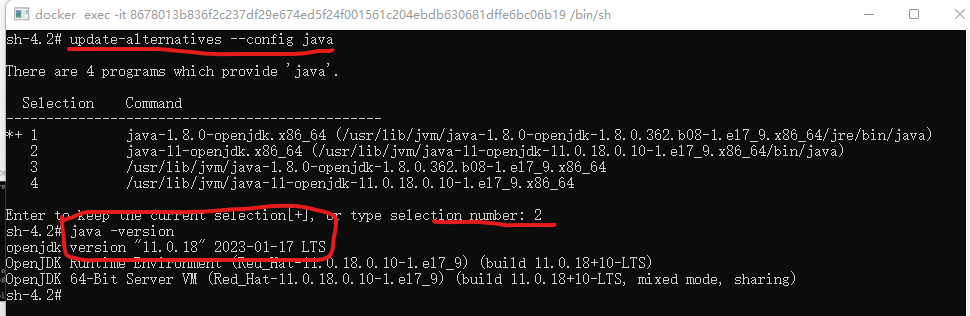
update-alternatives --install /usr/bin/java java /usr/lib/jvm/java-11-openjdk-11.0.18.0.10-1.el7\_9.x86\_64 11

**update-alternatives --config java (执行此命令选择并输入java版本对应编号)**

选择java 8:



选择java 11：



#### 其他

Java默认yum安装路径为**/usr/lib/jvm**

/usr/lib/jvm/java-11-openjdk-11.0.18.0.10-1.el7\_9.x86\_64

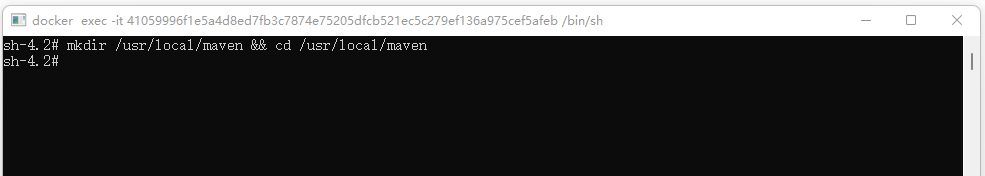
/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.362.b08-1.el7\_9.x86\_64

查看是否有Java安装： **rpm -qa | grep java**

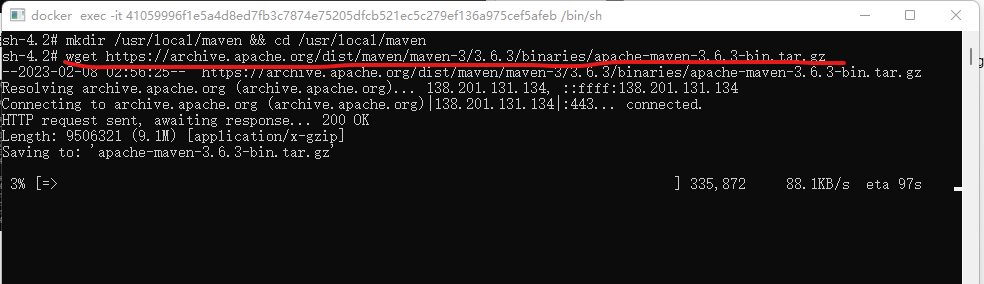
### Maven安装

#### Maven下载和安装

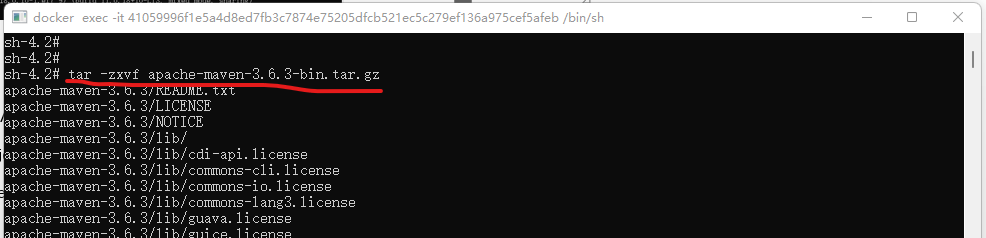
mkdir /usr/local/maven && cd /usr/local/maven



wget https://archive.apache.org/dist/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz



tar -zxvf apache-maven-3.6.3-bin.tar.gz



#### 配置环境变量

**暂时不需要设置maven环境变量。**

设置步骤：

编辑vim /etc/profile 输入以下内容：

export MAVEN\_HOME=/usr/local/maven/apache-maven-3.6.3

export MAVEN\_HOME

export PATH=$PATH:$MAVEN\_HOME/bin

然后执行source /etc/profile 使生效。

执行mvn -version 查看maven版本， 查看是否生效。

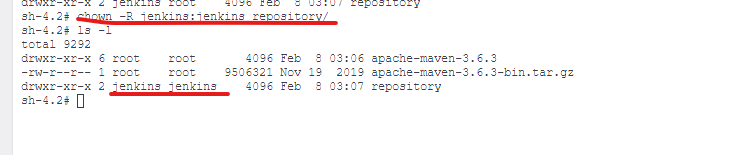
#### 配置本地仓库和镜像源

创建本地仓库：

mkdir -p /usr/local/maven/repository

修改本地仓库的权限，owner从root改为jenkins （可以先安装jenkins，等用户jenkins生成后再执行此操作），否则jenkins无法访问repository进行maven构建。

chown -R jenkins:jenkins repository/



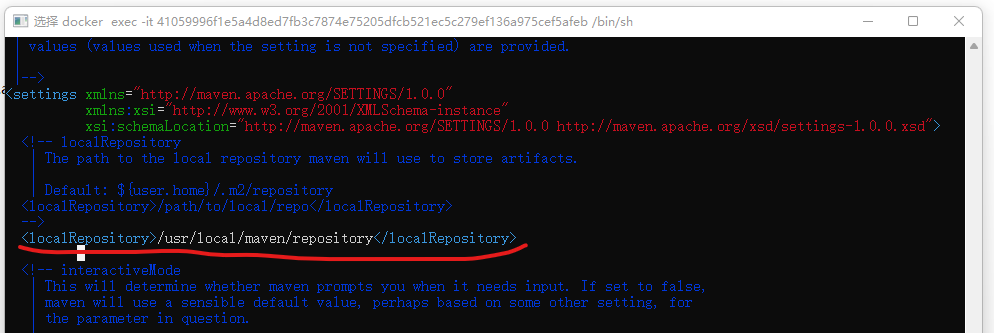
修改配置文件：

cd /usr/local/maven/apache-maven-3.6.3/conf/

vim settings.xml

修改本地仓库：

<localRepository>/usr/local/maven/repository</localRepository>



修改镜像源：

<mirror>

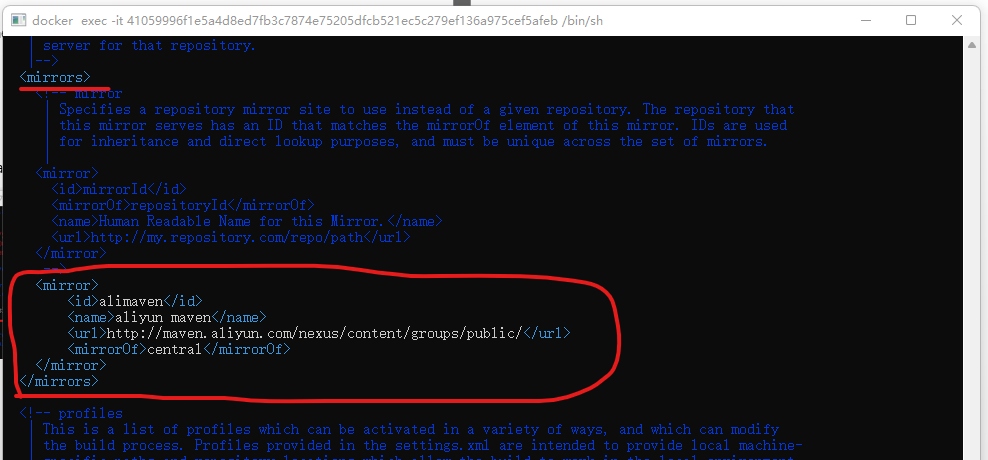
<id>alimaven</id>

<name>aliyun maven</name>

<url>http://maven.aliyun.com/nexus/content/groups/public/</url>

<mirrorOf>central</mirrorOf>

</mirror>



### Jenkins安装

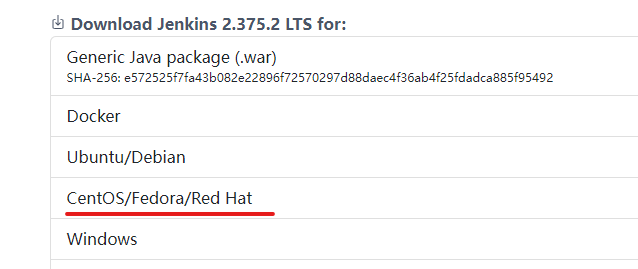
#### Jenkins概述

Jenkins基于Java环境运行，默认端口8080。

Jenkins会自动创建并使用默认账号jenkins:jenkins运行

#### Jenkins下载和安装

进入Jenkins (www.jenkins.io)官网，打开download面板， 选择CentOS版本

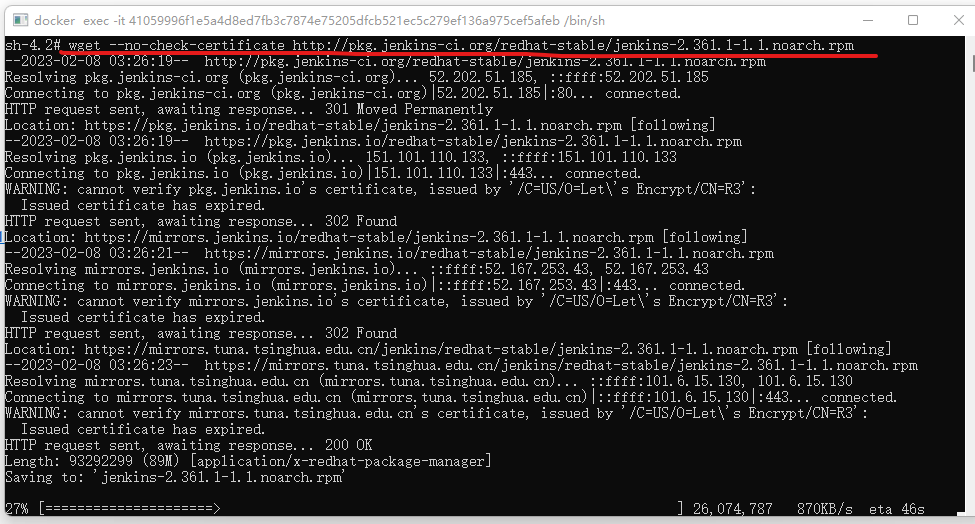


有两种版本，第一种weekly release line为每周都发布，第二种long term support为稳定版，选择稳定版的2.361.1进行下载，本环境不使用java 8，保持与Sonar相同的环境。



确定要下载的版本后在下面替换并使用wget下载 (To connect to pkg.jenkins.io insecurely, use `--no-check-certificate'. 免证书下载)：

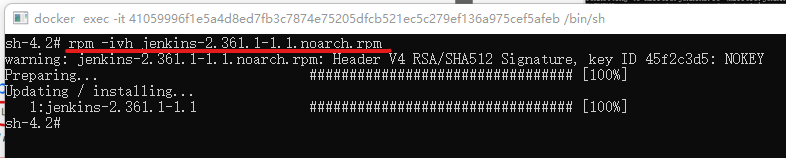
wget --no-check-certificate <http://pkg.jenkins-ci.org/redhat-stable/jenkins-2.361.1-1.1.noarch.rpm>



mkdir /usr/local/jenkins && mv jenkins-2.361.1-1.1.noarch.rpm /usr/local/jenkins/jenkins-2.361.1-1.1.noarch.rpm

cd /usr/local/jenkins/

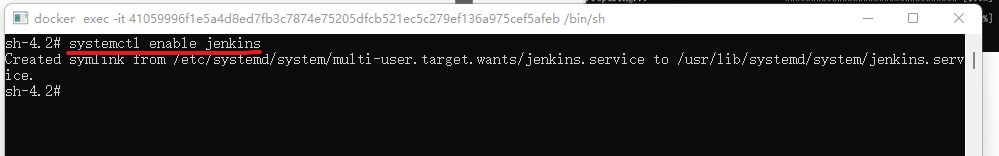
rpm -ivh jenkins-2.361.1-1.1.noarch.rpm



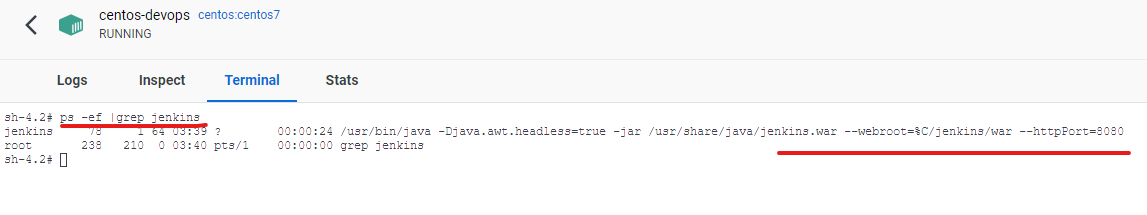
#### Jenkins 初始化配置

设置Jenkins开机自启，参考4）设置Jenkins开机自启。

设置防火墙规则，参考 5）开启防火墙。



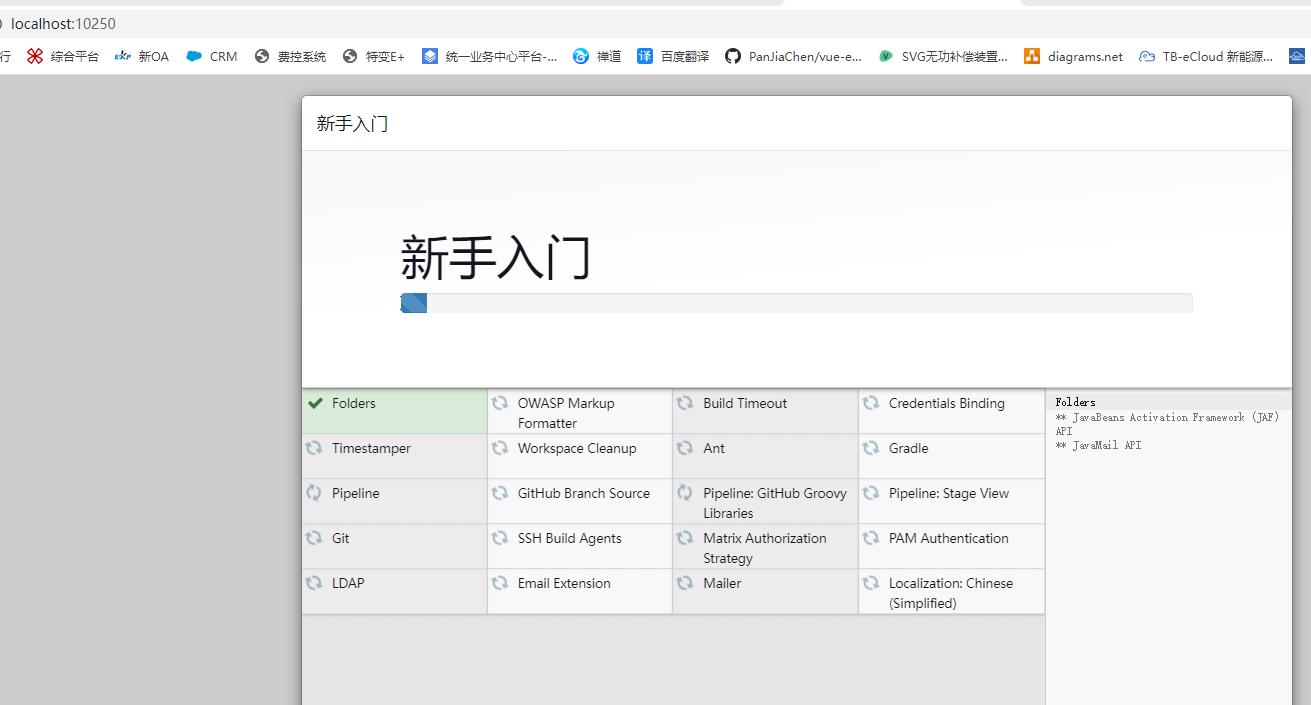
查看Jenkins进程是否启动：**ps -ef |grep jenkins**



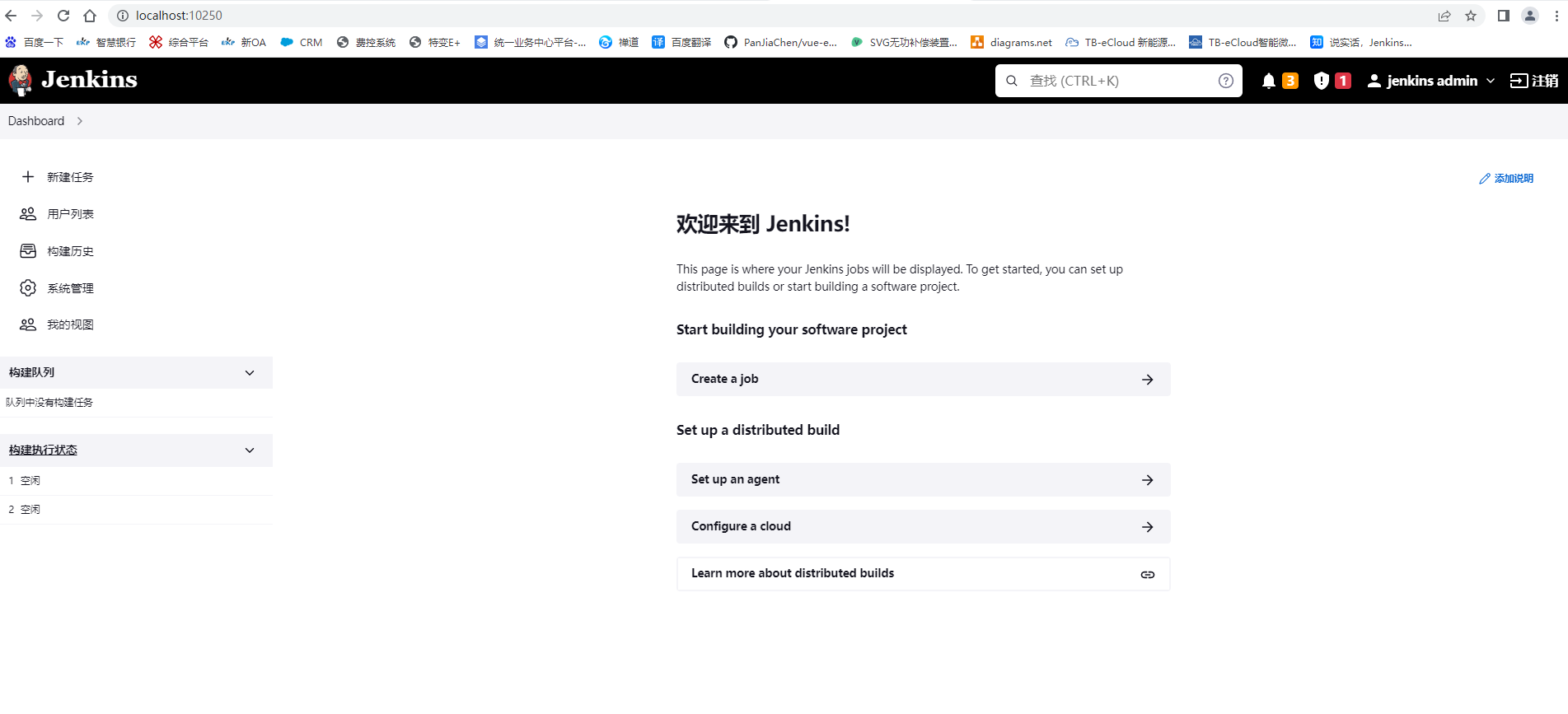
浏览器访问Jenkins web，完成初始化设置。



选择默认/推荐安装



安装完成 （设置一个系统账号， 本环境为jenkins/jenkins）



#### 设置Jenkins开启自启

Jenkins安装完成后会自动注册service并实现开机自动启动。

如果没有，使用systemctl enable Jenkins设置。

#### 开启防火墙，添加规则，开放8080端口

yum install firewalld systemd -y （如果没有相关命令先安装）

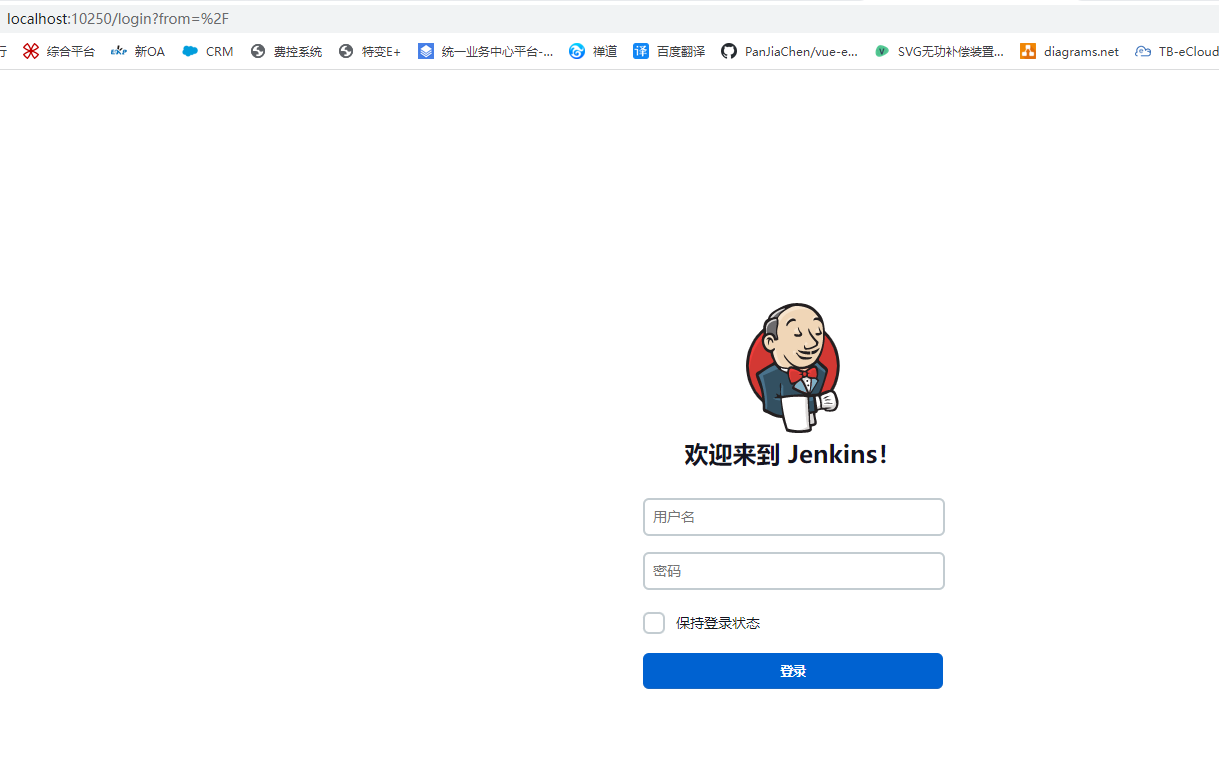
firewall-cmd --permanent --list-port （查看防火墙开放端口）

firewall-cmd --zone=public --add-port=8080/tcp –permanent （添加Jenkins的8080端口）

systemctl restart firewalld (重启防火墙使规则生效)



Jenkins可以访问了



#### 其他事项

Jenkins配置文件： /etc/sysconfig/jenkins

默认的JENKINS\_HOME目录： /var/lib/jenkins/

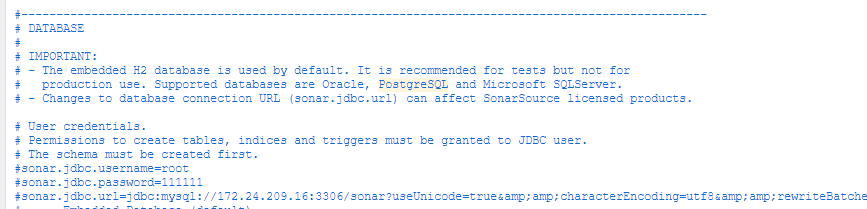
Jenkins日志文件： /var/log/jenkins/jenkins.log

查看Jenkins admin账户密码：cat /var/lib/jenkins/secrets/initialAdminPassword

### PostgreSQL安装

#### 概述

参考下面的图片（Sonar的setting.xml文件），SonarQube内置H2数据库用于扫描结果的存储，但是H2数据库只建议用于测试，对于正式/生成环境建议使用额外数据库Oracle、PostgreSQL或者SQLServer。本环境使用PostgreSQL。



#### 安装

yum search postgresql #检查安装源

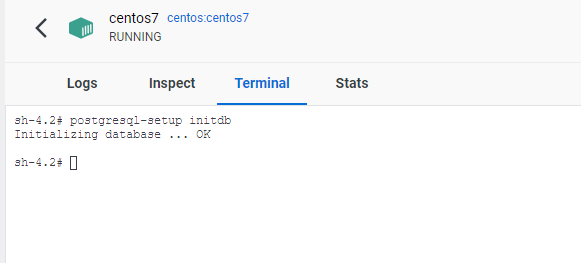
yum -y install postgresql postgresql-server #安装postgresql和server



安装完成，postgresql操作相关的命令都会添加到/usr/bin目录下，可以在命令行下直接使用。安装完成的同时，还会生成postgres用户， postgresql启动之后，在本机只能切换到postgres用户下才可以通过psql访问。安装完成之后，不能直接启动数据库，需要先执行初始化，初始化之后，会生成postgresql相关配置文件和数据库文件，他们都会存放在路径**/var/lib/pgsql/data**下。

#### 初始化配置

postgresql-setup initdb #初始化DB

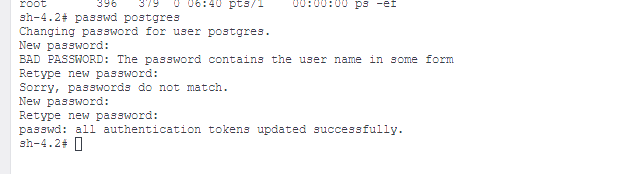


设置开启自启并启动postgresql：

systemctl enable postgresql.service

systemctl start postgresql.service

默认情况下，我们使用 psql -U postgres 命令登录，使用的是root用户，会提示没有这个角色，需要切换到 postgres 用户，然后直接登录。需要设置下 postgrep 用户的密码，设置成功后切换到该用户。（设置账户**postgres**的密码为**tbea@2000**）。



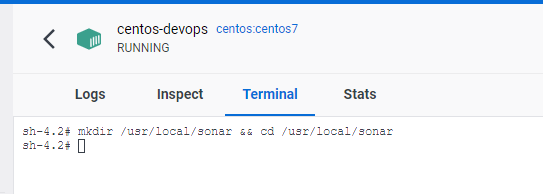
### Sonar安装

#### 概述

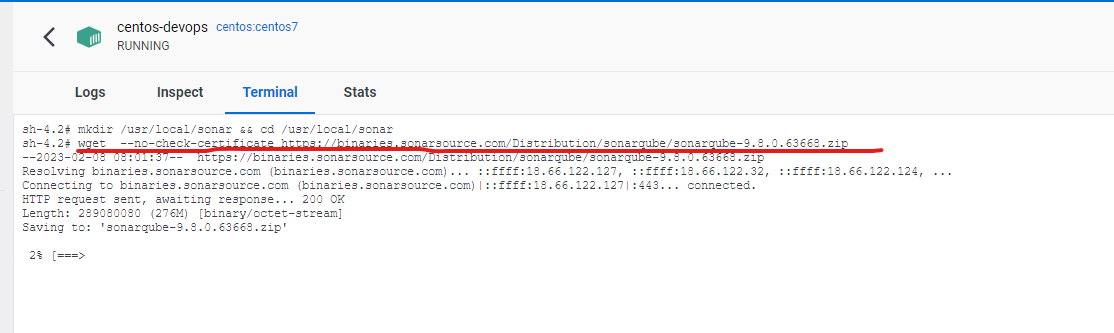
Sonar默认的web访问端口为9000。

#### 安装

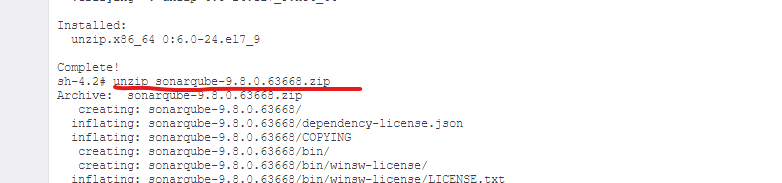
mkdir /usr/local/sonar && cd /usr/local/sonar



wget --no-check-certificate <https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.8.0.63668.zip>



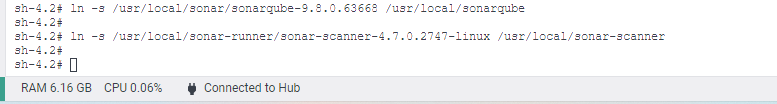
unzip sonarqube-9.8.0.63668.zip #解压缩



#### 设置

创建软链接，方便调用和配置。

ln -s /usr/local/sonar/sonarqube-9.8.0.63668 /usr/local/sonarqube



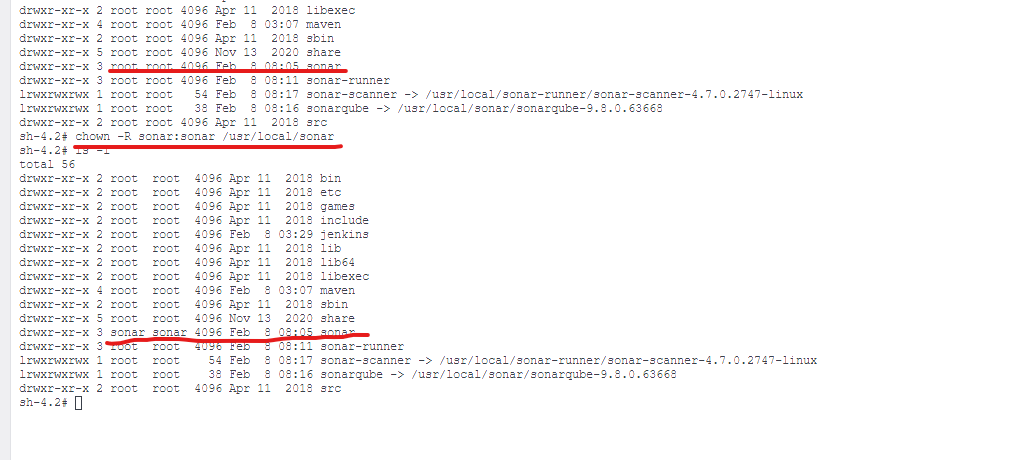
Sonar path: /usr/local/sonarqube

Sonarqube不能通过root账户运行，所以创建一个单独用户运行：

useradd sonar #添加sonar用户sonar

passwd sonar #设置密码为: tbea@2000

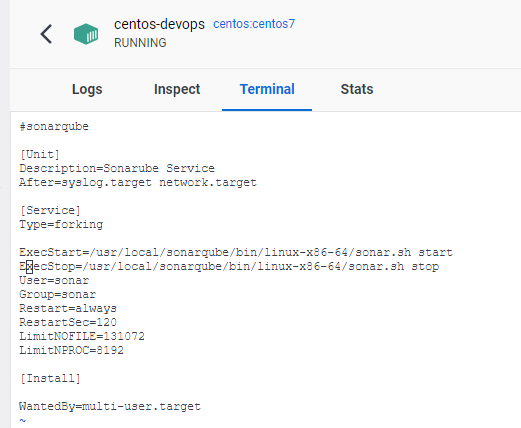
chown -R sonar:sonar /usr/local/sonarqube #修改文件夹owner



设置开机自启：

service添加sonarqube，在/etc/systemd/system/路径下创建sonar.service

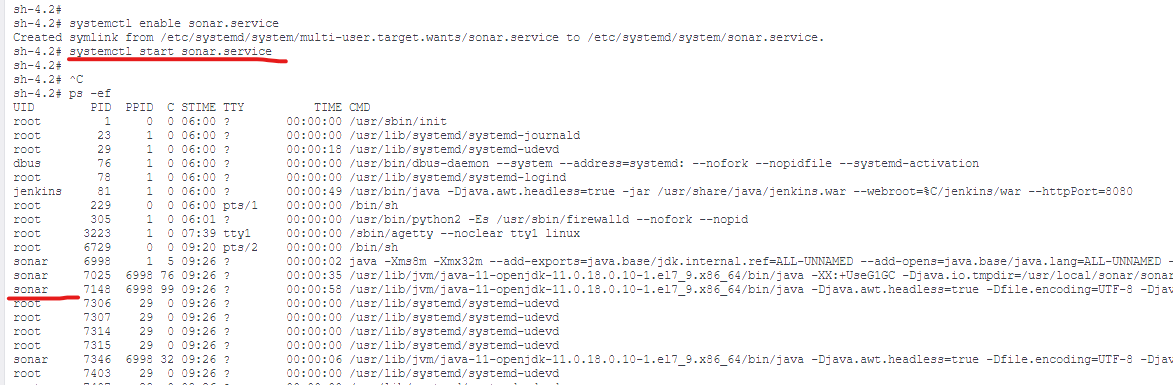
vim /etc/systemd/system/sonar.service



设置开启自启动：

systemctl enable sonar.service

systemctl start sonar.service



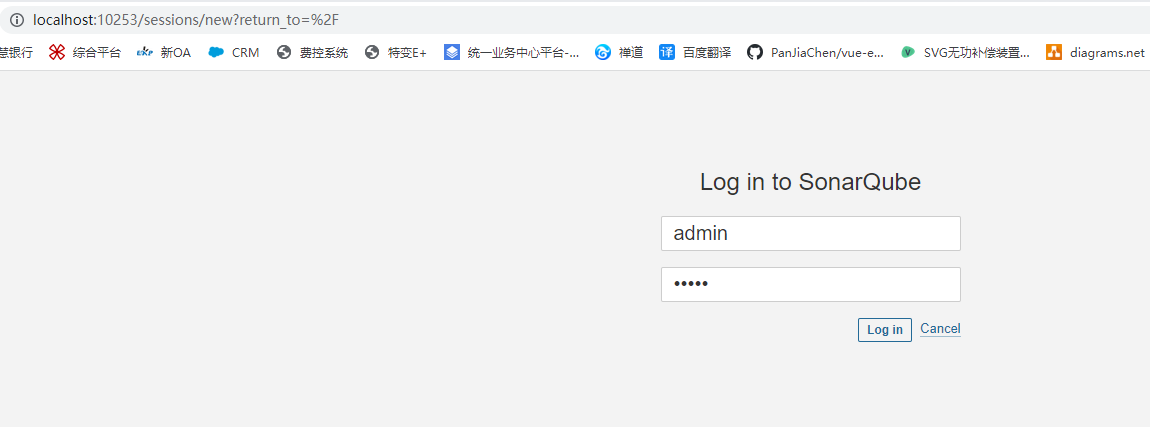
防火墙开启sonarqube web访问9000端口。

firewall-cmd --zone=public --add-port=9000/tcp --permanent

firewall-cmd --permanent --list-port

systemctl restart firewalld #重启防火墙使生效

浏览器访问sonarqube，设置密码： amin/111111

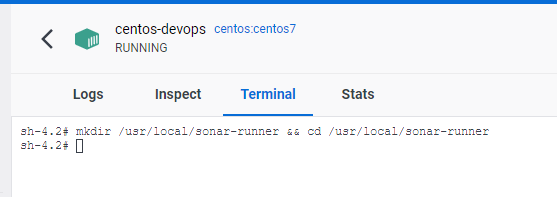


### Sonar Scanner安装

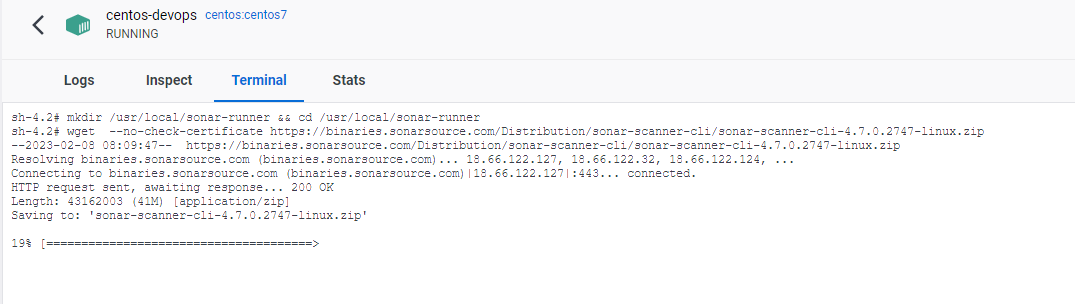
#### 概述

#### 安装

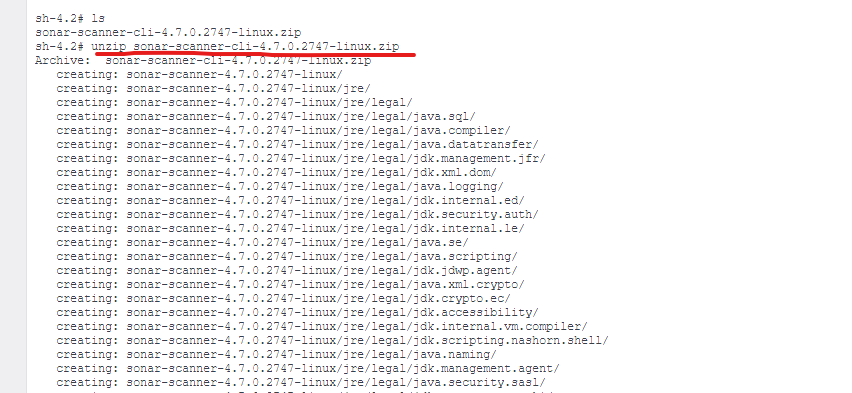
mkdir /usr/local/sonar-runner && cd /usr/local/sonar-runner



wget --no-check-certificate <https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-4.7.0.2747-linux.zip>



unzip sonar-scanner-cli-4.7.0.2747-linux.zip 解压缩

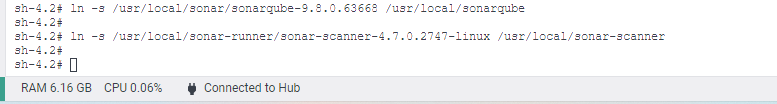


#### 设置

Sonar-runner path: /usr/local/sonar-runner/sonar-scanner-4.7.0.2747-linux

创建软链接，方便调用和配置。

ln -s /usr/local/sonar-runner/sonar-scanner-4.7.0.2747-linux /usr/local/sonar-scanner



Sonar-runner path: /usr/local/sonar-scanner

# 软件配置

## Jenkins设置

### 系统设置

进入首页-》系统管理-》系统配置



### 插件管理

进入首页-》系统管理-》插件管理 进行插件的下载、安装、管理。插件下载安装后需重启Jenkins使生效。



#### SVN插件安装 – Subversion

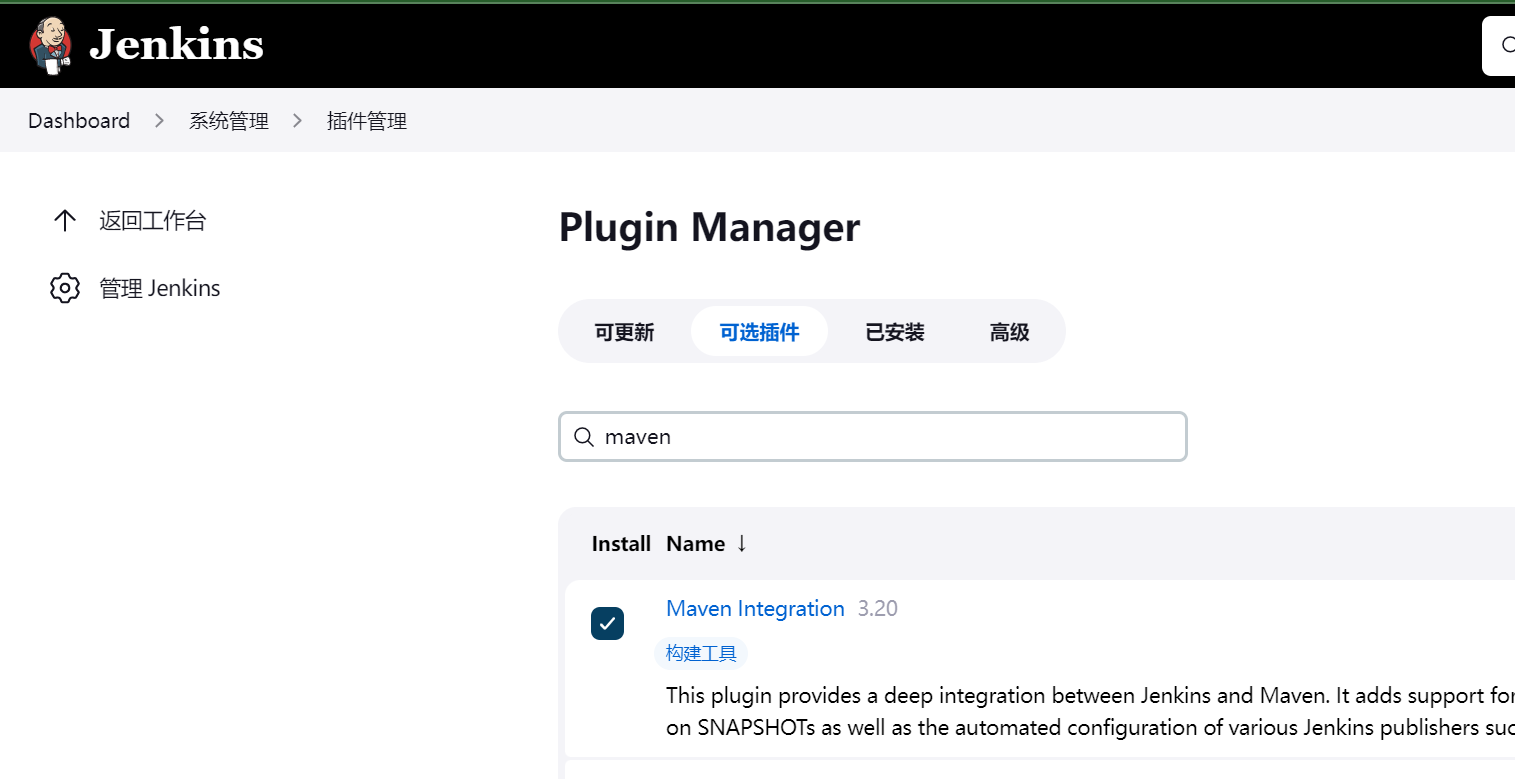
搜索“Subversion”插件并进行安装， Jenkins默认不支持SVN作为SCM。



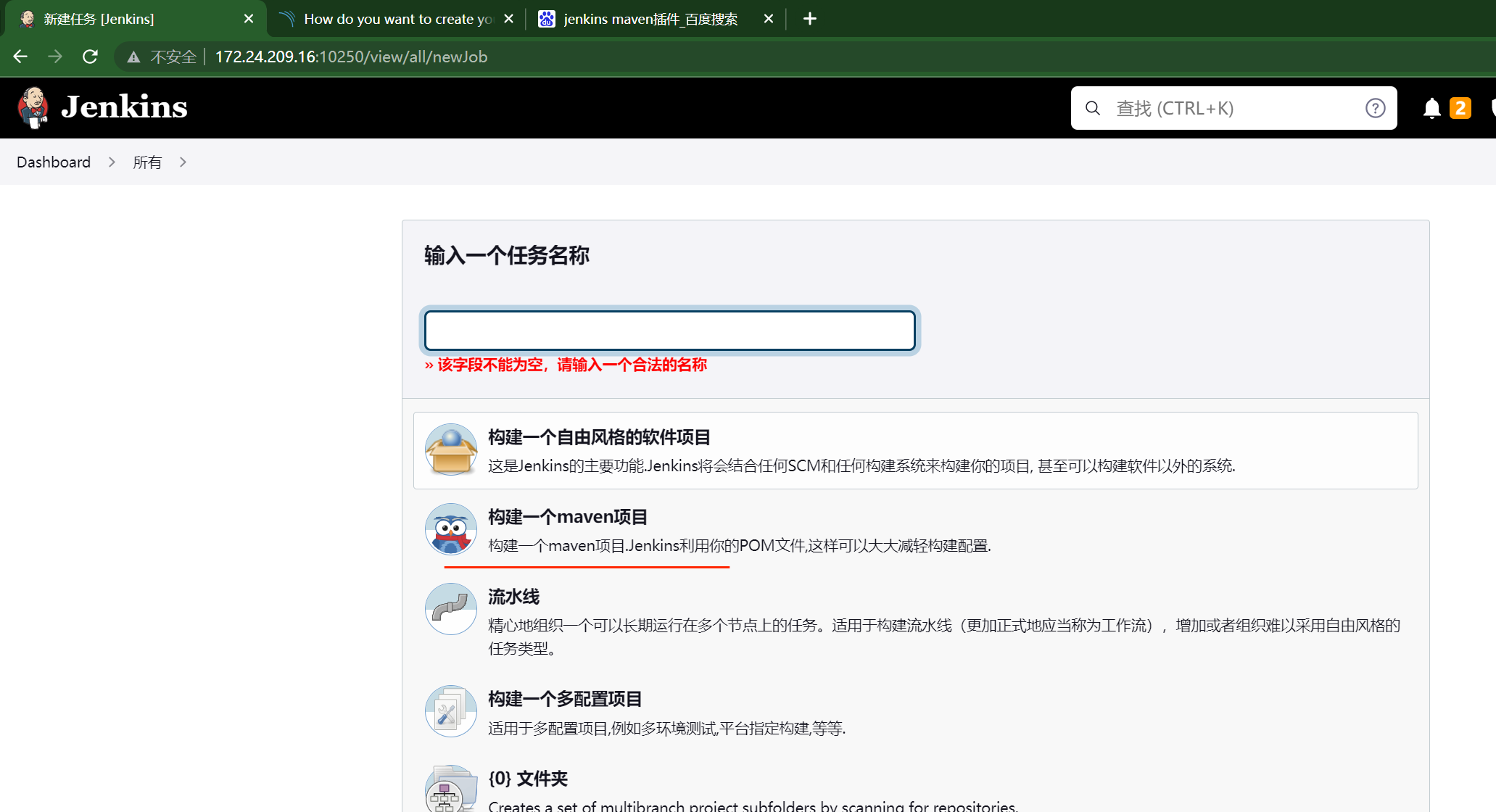
#### Sonar插件

#### Maven插件

1. 搜索“Maven Integration”插件并进行安装



安装完成后，新建任务会多出“构建maven项目”选项。



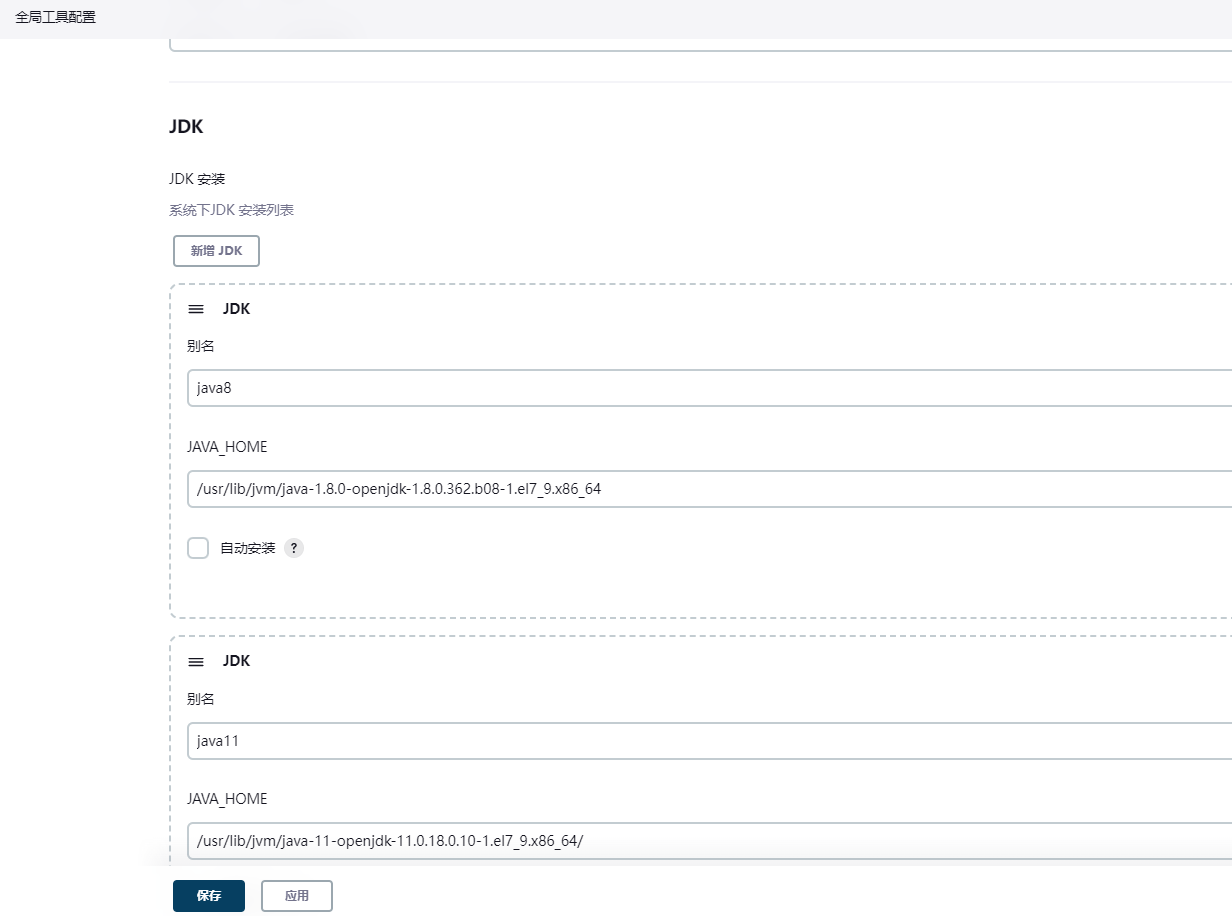
#### 权限插件

### 全局工具配置

进入首页-》系统管理-》全局工具配置

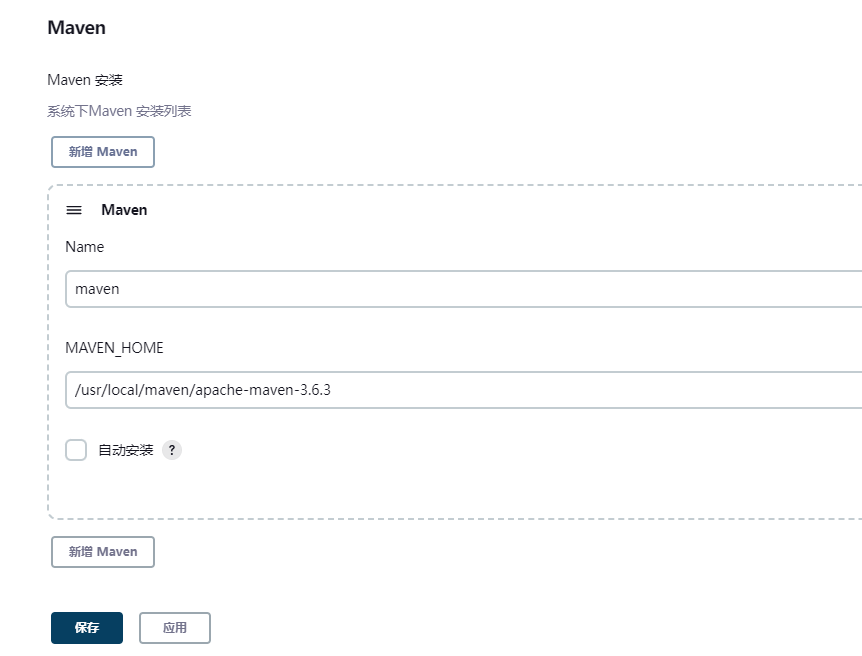
#### SVN配置

#### JDK配置



#### Maven配置





#### Sonar配置

#### Sonar scanner配置

### 安全策略 – RBAC

### 凭证管理

#### 添加SVN的认证凭证 – credential

类型为“username and password”，设置为全局，ID必须唯一。



#### 添加SonarQube的认证凭证 – token

#### 添加其他凭证

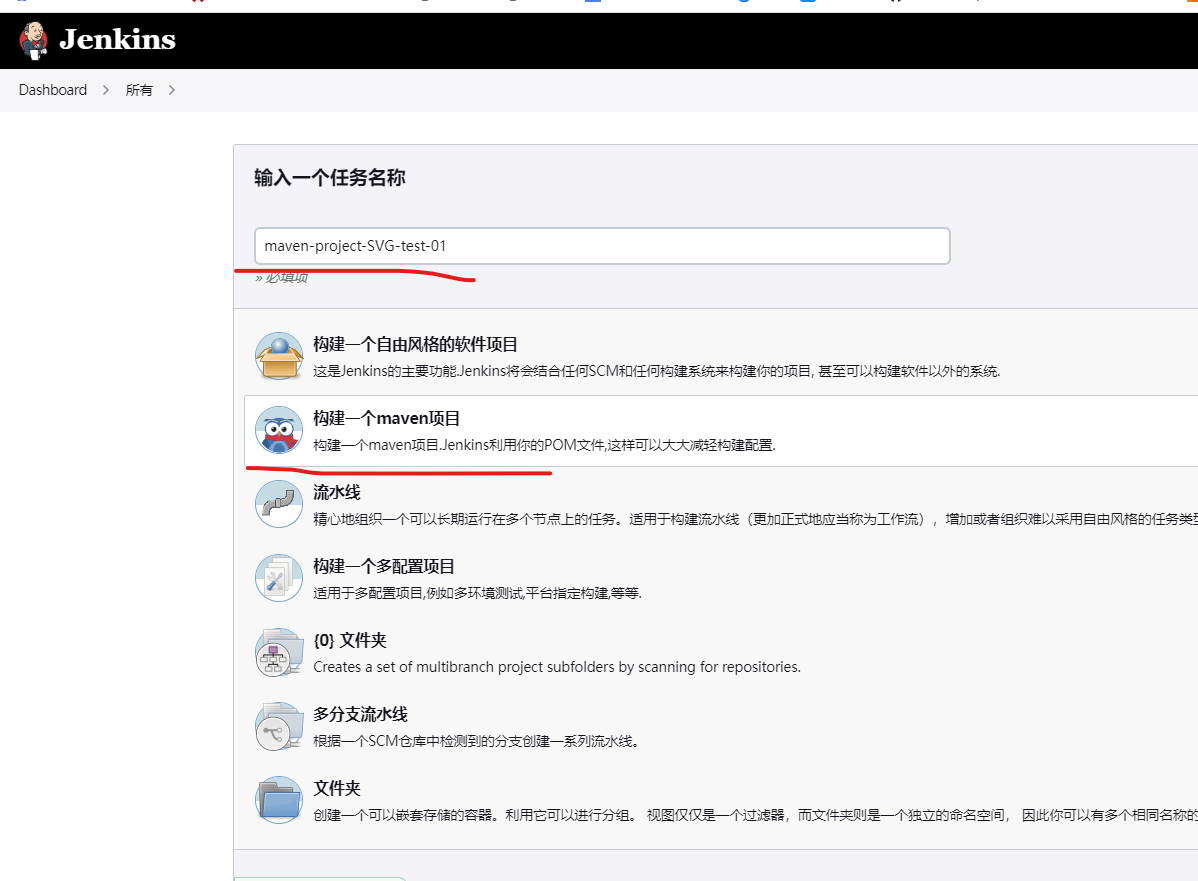
## Sonar设置

# Jenkins

## Jenkins项目构建

### 通过maven项目模板创建任务

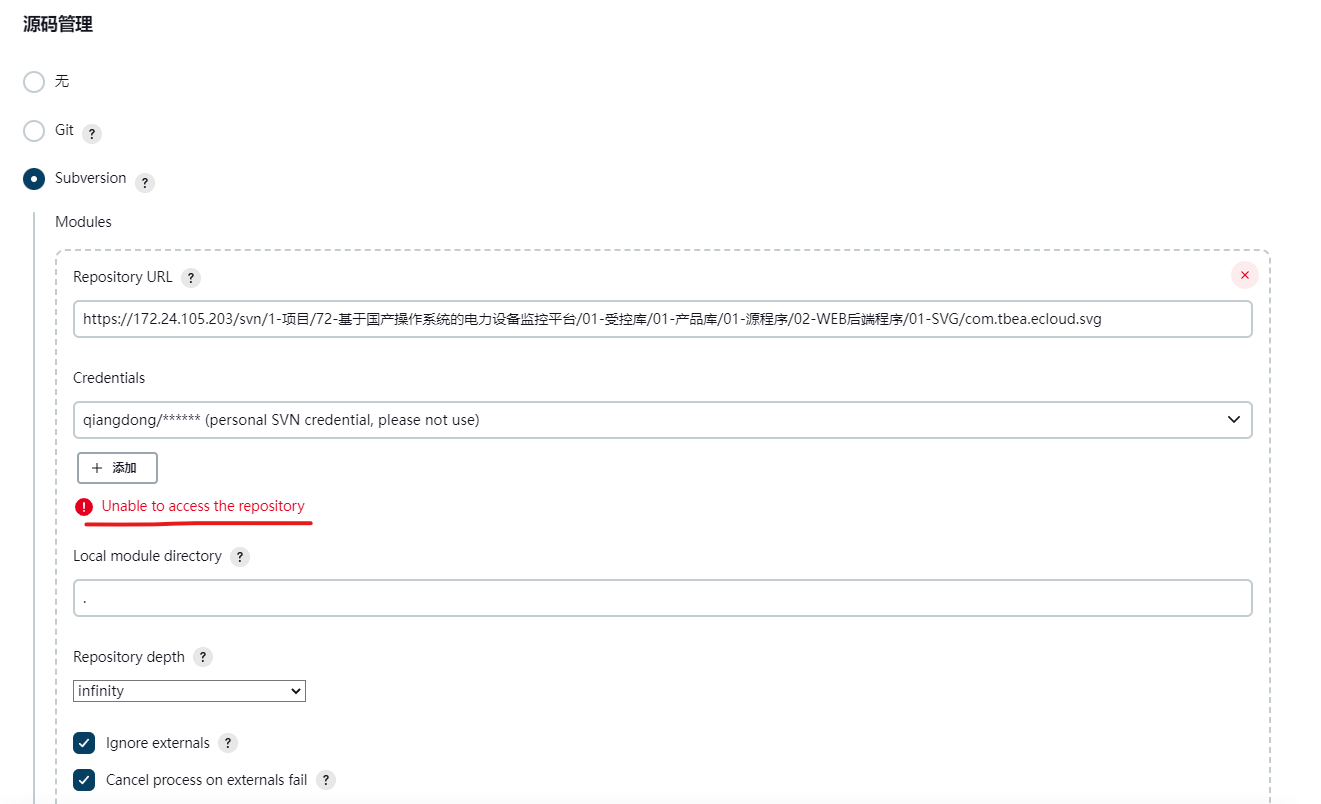
1.新建一个任务，模板选择“构建一个maven项目”，输入任务名，点击保存。



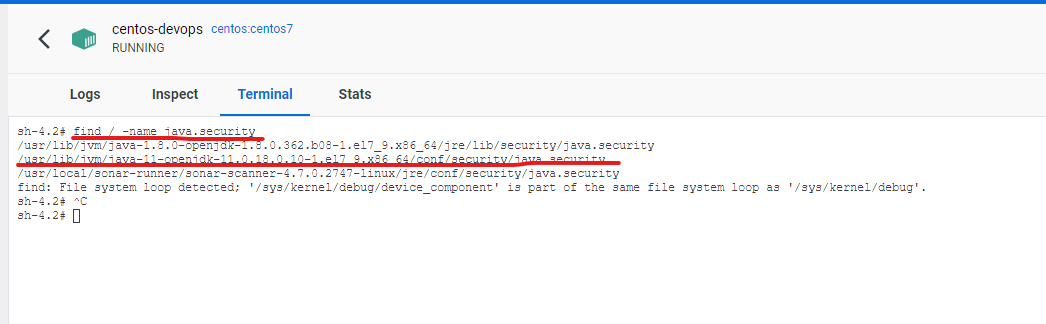
2.JDK选择java8, 全局工具配置配置的java8和java11.



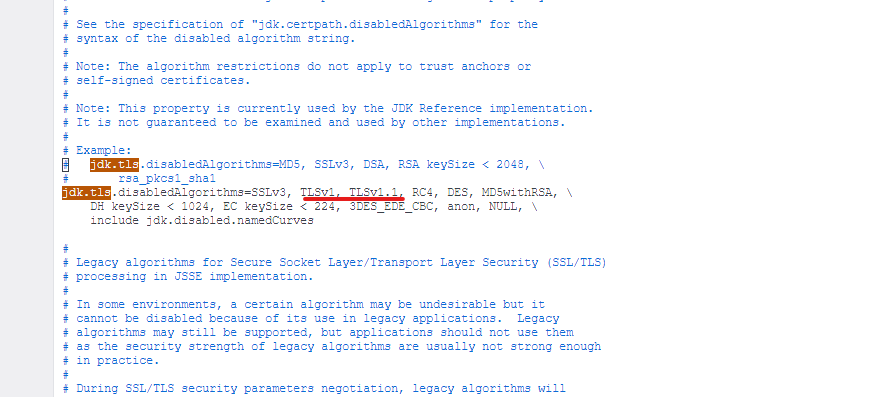
3.SCM使用SVN，之前有安装过支持SVN的插件，填入仓库地址，选择之前配置的全局认证凭证作为访问SVN的credential做认证和授权。



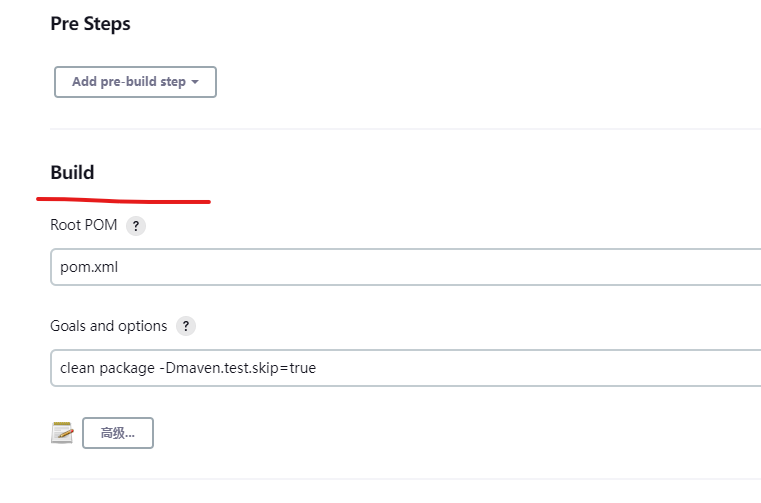
4.错误提示Unable to access the repository，是因为SVN不支持低版本TLS，需要在客户端的JDK中禁用，使用高版本TLS。使用**find / -name java.security**搜索java11对应的java.security文件（Jenkins自身使用的java11，构建项目用的java8），禁用低版本TLS。



搜索“jdk.tls”，删除掉TLSv1和TLSv1.1，点击保存。需要重启Jenkins。



5.配置构建build命令 clean package -Dmaven.test.skip=true



6.点击保存，开始构建。

### 通过流水线创建项目 – 基于dockerfile

## Jenkinsfile

# 代码质量

# Docker命令

docker pull centos:centos7

docker run -itd -p 10250:8080 -p 10251:50000 -p 10252:8088 -p 10253:9000 -p 10254:9001 -p 10255:9092 --name centos-devops --privileged=true centos:centos7 /usr/sbin/init