# 接口自动化平台部署说明文档

## 说明：

本文档的适用范围适合在centos7上，其他系统没有验证过，但是整体大同小异，可根据各个操作系统的差异做对应的调整

## 整体说明：

整个部署过程分为五个部分：

第一部分：安装系统依赖包

第二部分是python3.7.5的安装

第三部分是python相关依赖包的安装

第四部分是前端应用的安装

第五部分是Jenkins安装和基本使用

## 正式安装部署：

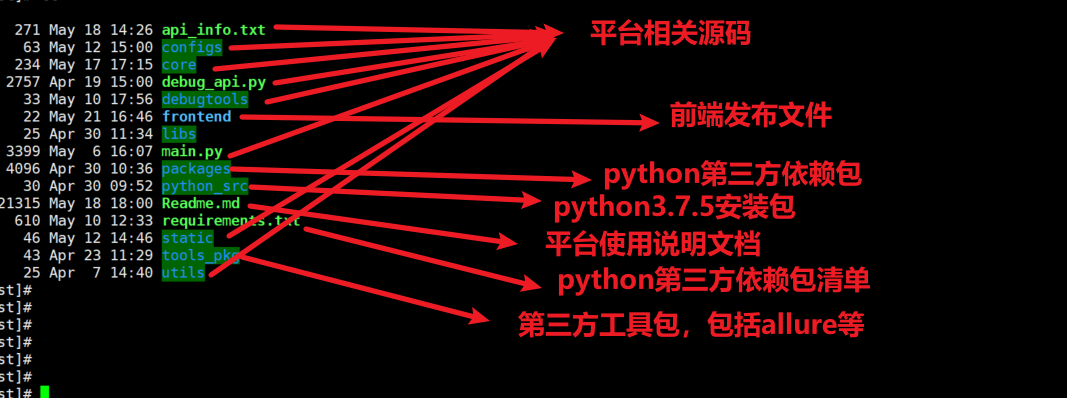
### 第一步：将附件压缩包拷贝到目标主机上，并解压缩



# tar -zxvf source.tar.gz

# ls -l

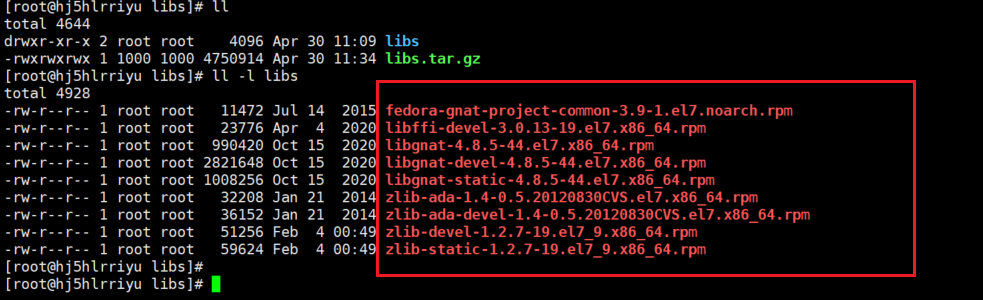
其中目录说明如下：



### 第一部分：安装系统依赖包

1. 解压文件

当前依赖的系统包如下：

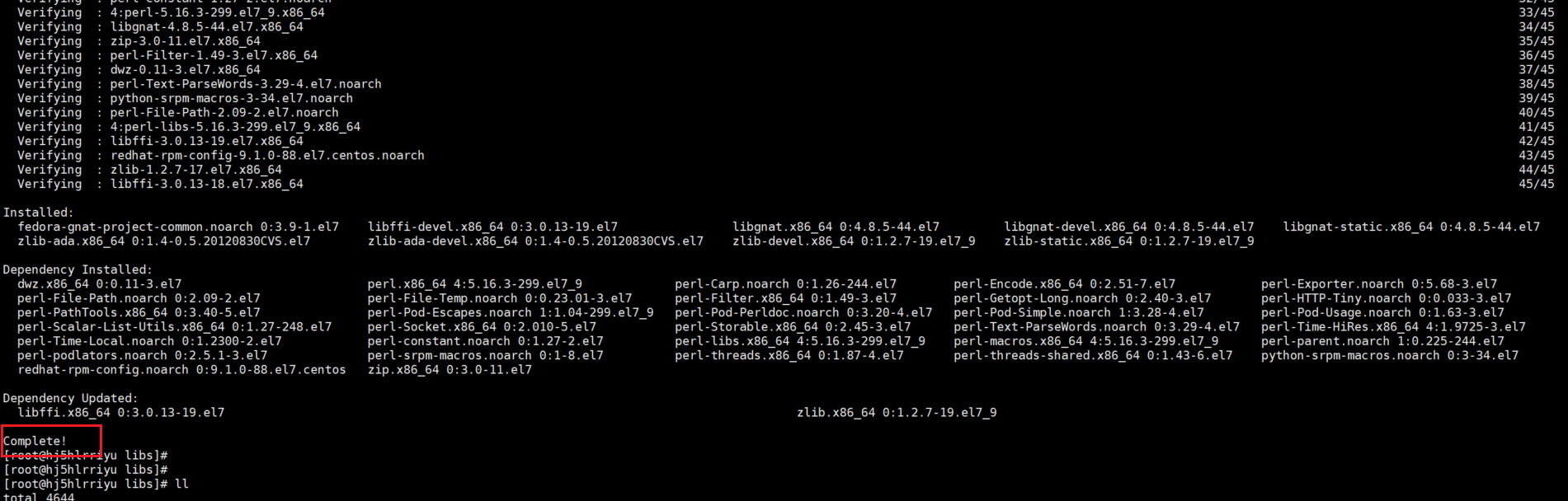


安装：

cd libs # 进入离线包目录

tar -zxvf libs.tar.gz

yum -y install libs/\* # 安装系统依赖包



### 第二部分：python3.7.5安装

#### 1、解压文件和创建安装目录

cd python\_src

tar -zxvf Python-3.7.5.tgz

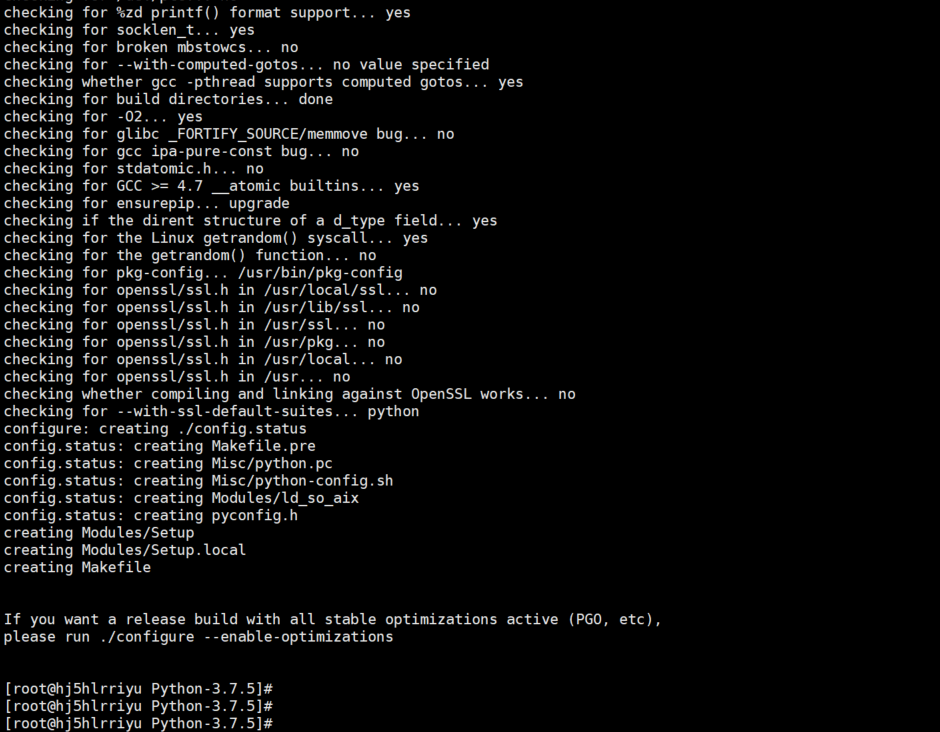
cd Python3.7.5

mkdir /usr/local/python3.7.5

#### 2、执行配置

configure是一个shell脚本，根据平台的特性生成Makefile文件，为下一步的编译做准备。可以通过在 configure 后加上参数来对安装进行控制，比如下面就是指定安装目录/usr/local/python3.7.5。如果没有用这个选项，安装过程结束后，该软件所需的软件被复制到不同的系统目录下，比较混乱。

./configure -prefix=/usr/local/python3.7.5



#### 3、安装

make && make install

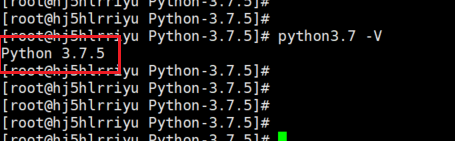
#### 4、创建软链接

ln -s /usr/local/python3.7.5/bin/python3.7 /usr/bin/python3.7

ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7

#### 5、验证安装是否成功

python3.7 -V



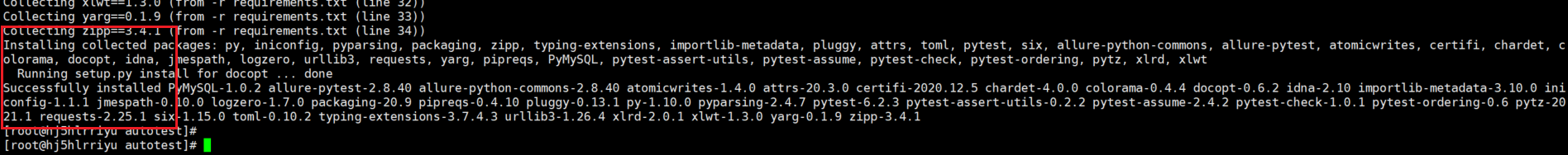
到此，python3.7.5安装完成

### 第三部分：安装相关依赖包

#### 安装python第三方依赖包

* 1. 进入packages目录，执行以下命令

pip3.7 install --no-index --find-links=./packages/ -r requirements.txt

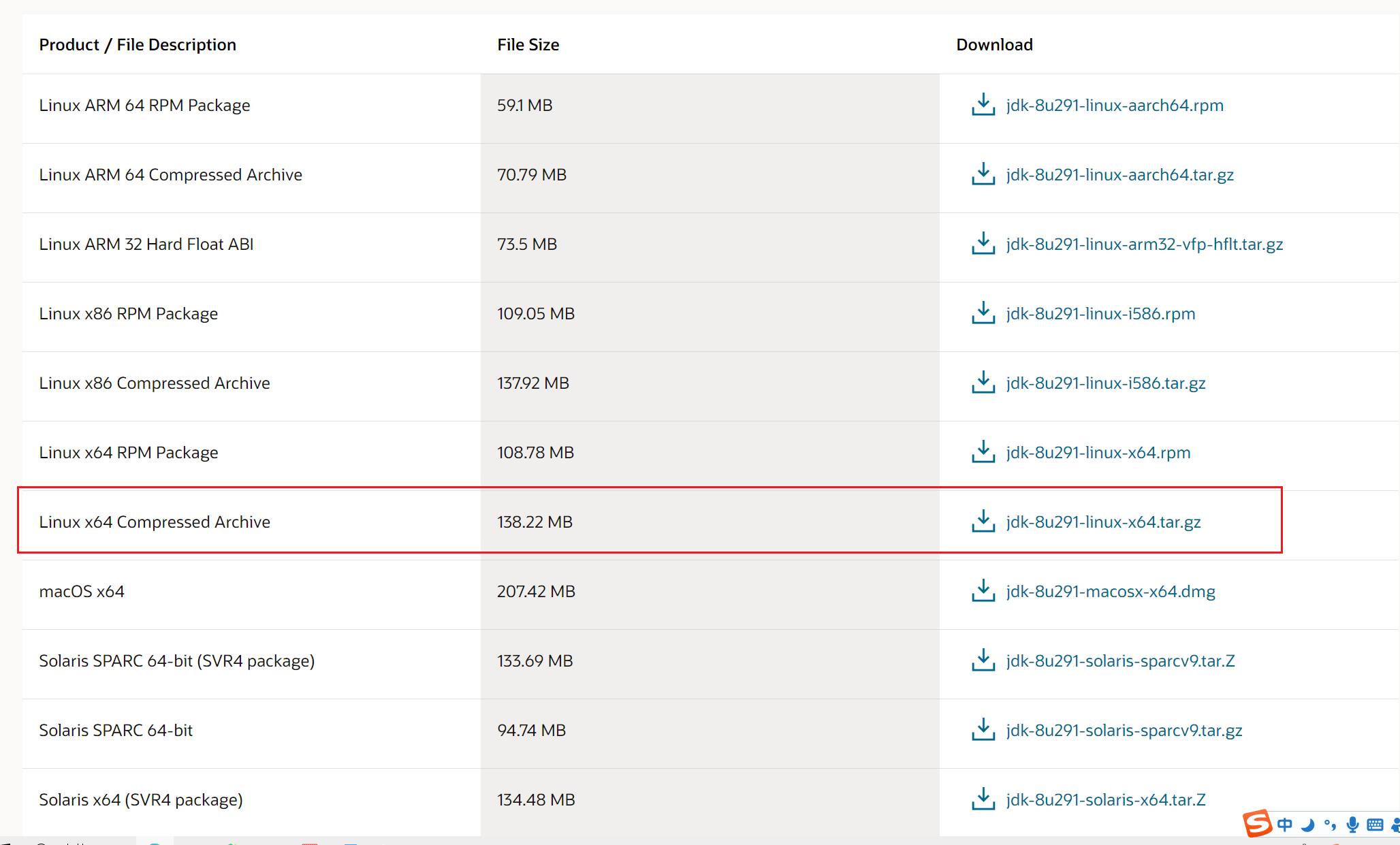


到此，python相关的内容安装完毕

#### 2、配置java环境（allure的运行依赖java）

2.1、到官网下载jdk8的linux安装包，这里使用的是jdk1.8.0\_291版本，其他版本大同小异

官网地址：[Java SE Development Kit 8 - Downloads (oracle.com)](https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html) 下载截图所示的文件包



2.2、将下载好的jdk传输到目标机上，这里使用的存放路径是:/opt/software

2.3、解压jdk文件到/opt/modules/目录下，执行以下命令：

tar -zxvf jdk-8u291-linux-x64.tar.gz -C /opt/modules

2.4、配置java环境变量

#解压文件移动

#配置jdk环境变量

#目录/etc/profile.d/下新建java.sh

[root@hj5hlrriyusrc]# vim /etc/profile.d/java.sh

#录入以下内容

export JAVA\_HOME=/opt/modules/jdk1.8.0\_291

export JRE\_HOME=${JAVA\_HOME}/jre

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

export CLASSPATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar:$JRE\_HOME/lib:$CLASSPATH

#加入环境变量后让配置立即生效

[root@hj5hlrriyusrc]# source /etc/profile

#加入开机启动

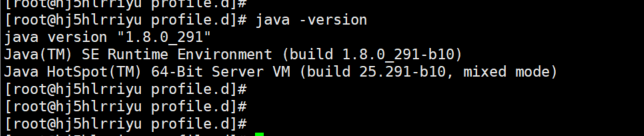
[root@hj5hlrriyusrc]# chmod +x /etc/rc.local

[root@hj5hlrriyusrc]# vim /etc/rc.local

source /etc/profile //添加此行

#运行查看

[root@hj5hlrriyusrc]# java -version



#### 3、配置allure环境

3.1、将allure安装包拷贝到/opt/modules目录中

将第一步中source压缩包中解压出来的tools\_pkg中的allure-2.13.8目录拷贝到/opt/modules中：

cp -a /root/auto/autotest/tools\_pkg/allure-2.13.8 /opt/modules/

3.2、配置allure环境变量

#配置jdk环境变量

#目录/etc/profile.d/下新建allure\_env.sh

[root@hj5hlrriyusrc]# vim /etc/profile.d/allure\_env.sh

#录入以下内容

#! /bin/bash

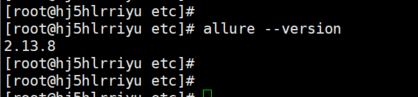
PATH=$PATH:/root/auto/autotest/tools\_pkg/allure-2.13.8/bin

export PATH

3.3、立即生效

source /etc/profile

3.4、检查allure版本



#### 4、检查整个自动化工程是否能正常运行

4.1、进入工程主目录

cd /root/auto/autotest/

4.2、执行工程入口文件

当前开始任务需要通过命令行入参的方式开始执行，具体的格式如下：

示例：python3.7 main.py -f ./testcase/demo.xls

命令说明：

python3.7 : 本地配置的python3.7.5的解释器的别名

main.py : 工程根目录下的程序入口文件

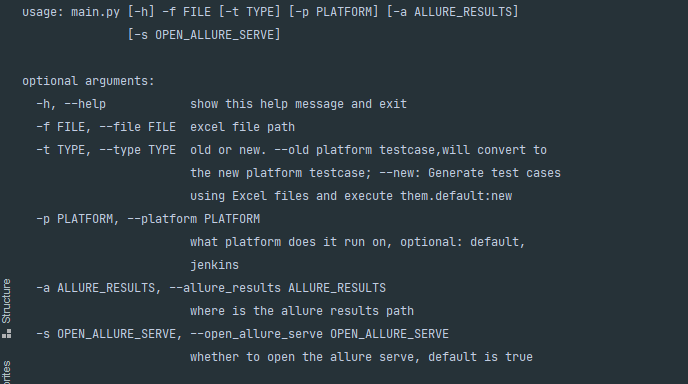
-f : 指定需要执行的测试用例文件（必须要指定）

-a : 指定allure报告存储的目录，默认不用配置，主要区分在jenkins上运行还是在本地运行，默认存储在工程目录的reports/用例文件名/ 目录下

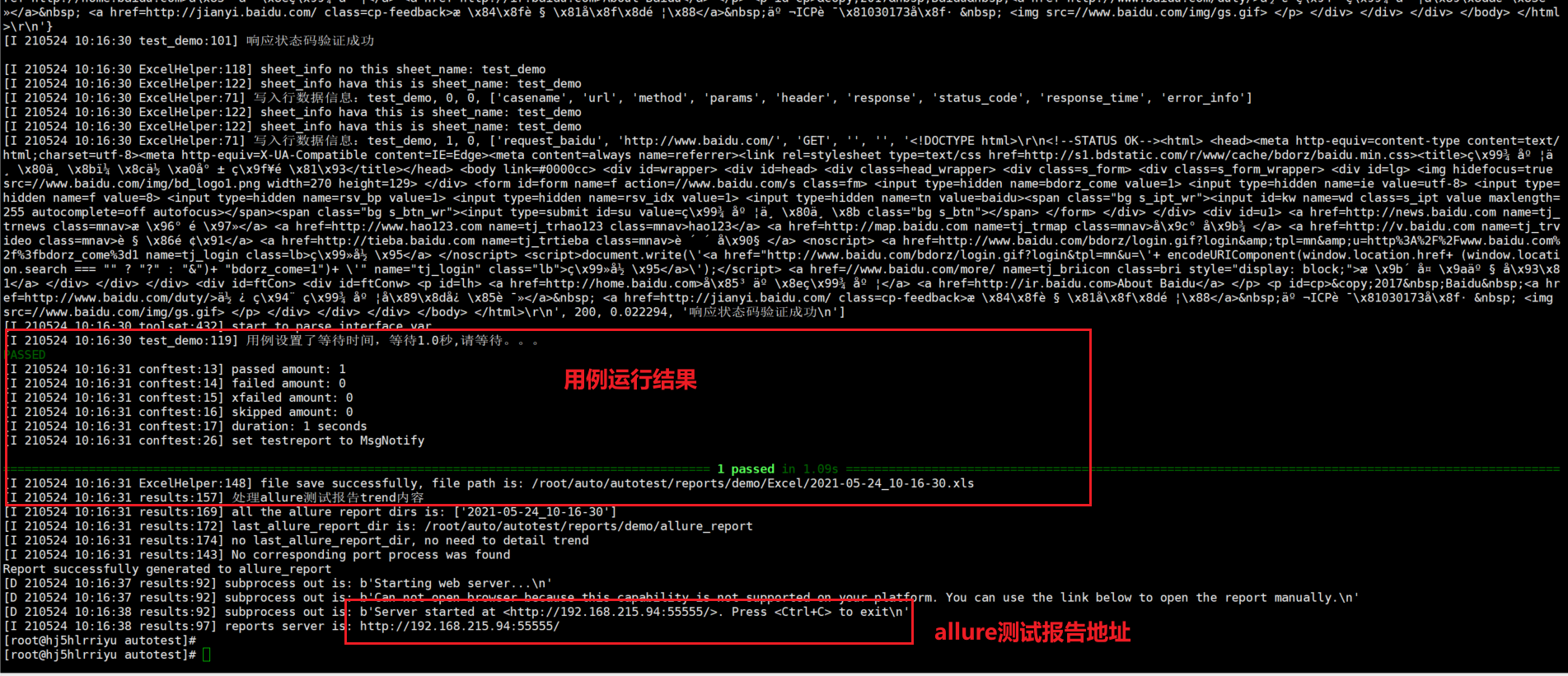
-p : 运行工程的来源，支持本地运行和jenkins调用运行，主要在于区分allure的展示上

-s : 是否自动打开allure报告服务，默认打开

具体说明可以通过python3.7 main.py -h查看各个参数说明



4.3、查看运行结果



### 第四部分：安装前端应用

1. 安装nginx

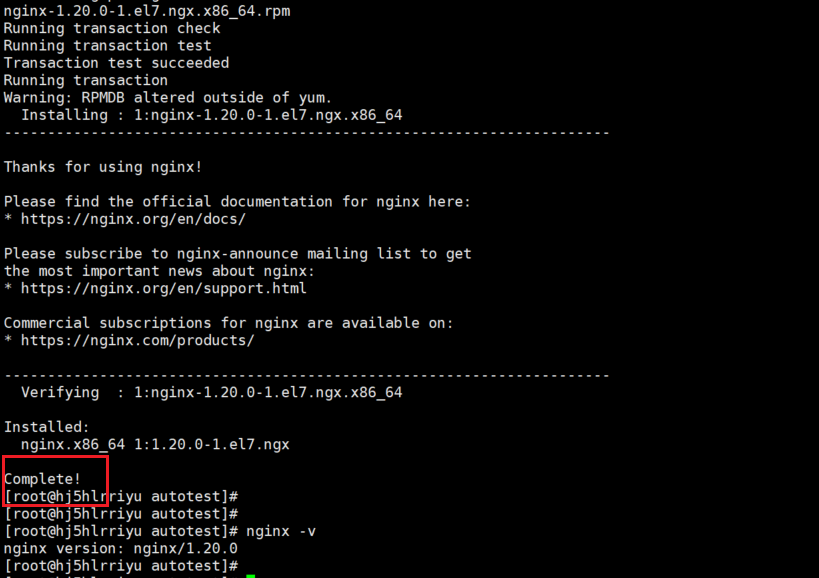
直接通过yum命令在线安装

# 添加yum源

rpm -Uvh http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-centos-7-0.el7.ngx.noarch.rpm

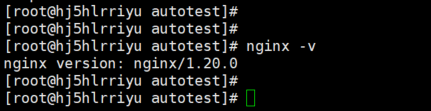
# 安装

yum -y install nginx



检查nginx的版本

nginx -v



nginx -v命令执行成功，nginx安装完成

1. 配置nginx

这里我们不直接在/etc/nginx/nginx.conf中配置，通过另外的editor.conf文件配置，然后在/etc/nginx/nginx.conf中通过include导入

2.1、配置/etc/nginx/conf.d/editor.conf文件

/etc/nginx/conf.d/editor.conf文件具体配置内容如下：

server {

listen 55556; # 监听的端口

server\_name 192.168.215.94; # 监听的ip，根据实际情况填写

#charset koi8-r;

#access\_log logs/host.access.log main;

#项目的根目录

root /root/auto/autotest/frontend/autotest;

#网站根入口

location / {

# 此处的 @router 实际上是引用下面的转发，否则在 Vue 路由刷新时可能会抛出 404

try\_files $uri $uri/ @router;

# 请求指向的首页

index index.html;

}

# 由于路由的资源不一定是真实的路径，无法找到具体文件

# 所以需要将请求重写到 index.html 中，然后交给真正的 Vue 路由处理请求资源

location @router {

rewrite ^.\*$ /index.html last;

}

}

2.2、配置/etc/nginx/nginx.conf文件

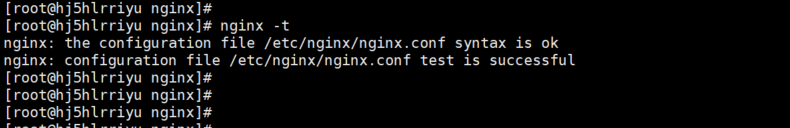
2.2.1、通过root用户启动nginx，避免权限问题

2.2.2、在/etc/nginx/nginx.conf文件中的http中导入，如图所示



2.3、检查nginx配置是否有误

nginx -t



2.4、提示无误，通过nginx启动nginx

nginx

2.5、打开hostname:55556查看是否能正常访问



### 第五部分：配置Jenkins

1、安装Jenkins

1.1、将工程目录下tools\_pkg目录下的jenkins.war文件拷贝到/opt/modules/jenkins/目录下

#创建在/opt/modules/下创建jenkins目录

mkdir -p /opt/modules/Jenkins

#拷贝文件到对应的目录中

cp -a /root/auto/autotest/tools\_pkg/jenkins.war /opt/modules/jenkins

1.2、编写jenkins启动脚本

在/opt/modules/jenkins目录下新建start.sh文件并登入内容：

vim start.sh

start.sh文件的内容：

#!/bin/sh

java -jar jenkins.war --httpListenAddress=0.0.0.0 --httpPort=8080 &

# 这里默认使用8080端口

给start.sh脚本添加可执行权限

chmod a+x start.sh

1.3、执行启动脚本

./start.sh

2、配置Jenkins初始内容

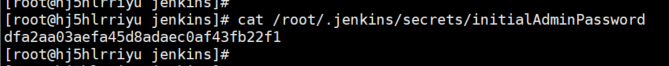
2.1、在web端打开Jenkins登陆页面，这里的主机ip是：192.168.215.94

在浏览器输入192.168.215.94:8080，打开页面如下：



2.2、根据提示，在/root/.jenkins/secrets/initialAdminPassword查看初始密码

cat /root/.jenkins/secrets/initialAdminPassword



2.3、将查看到的密码填入密码框中，等待页面初始化完成

2.4、在自定义Jenkins中，安装推荐的插件



等待安装进度条走完



安装进度条走完，部分插件安装失败了，可以在后续需要用的时候再通过离线安装的方式进行安装



点击继续，选择使用admin账户继续





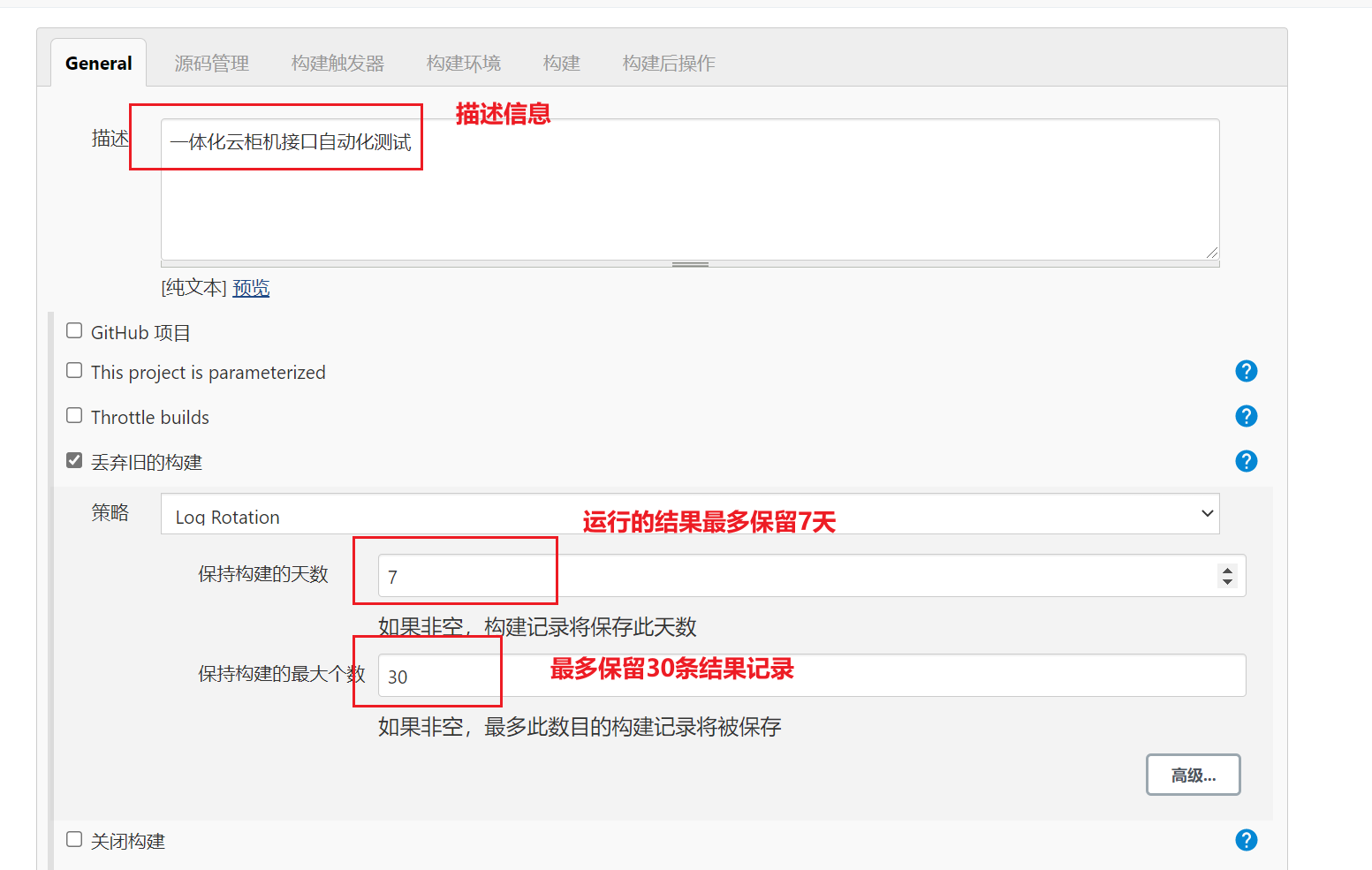
Jenkins安装完成，使用Jenkins进行配置

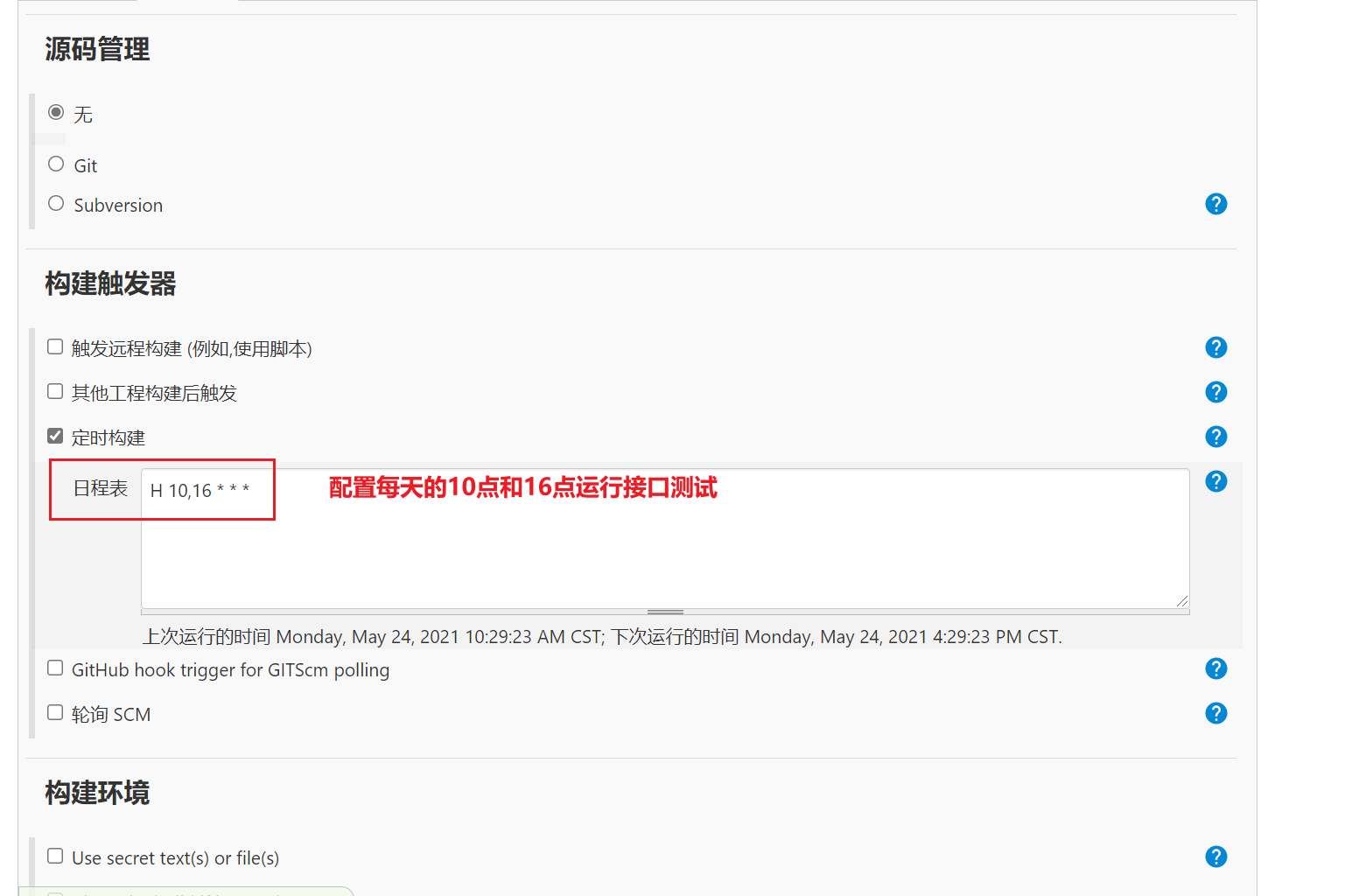
在首页点击新建item

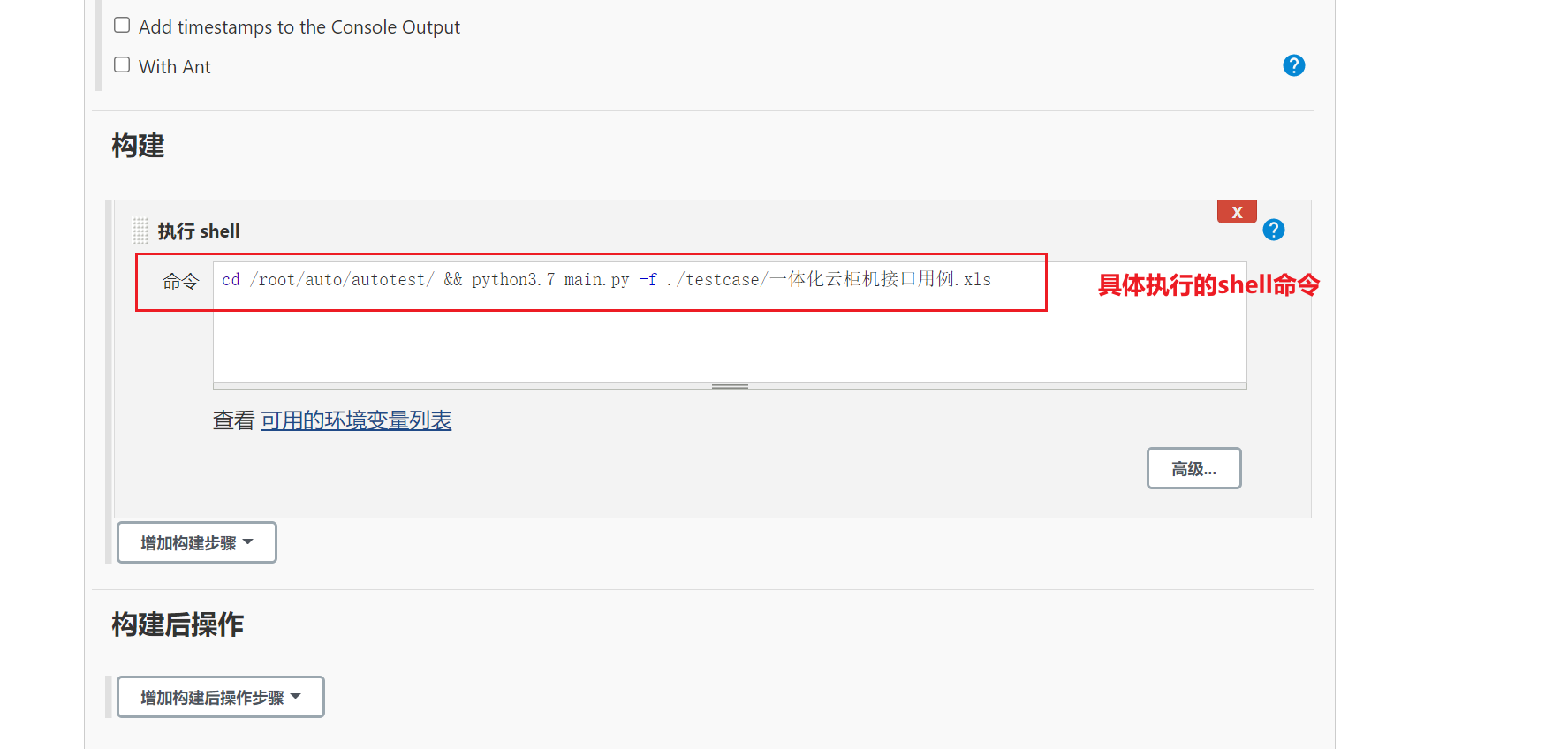




工程的具体配置如图：







配置完成后，点击保存应用

2.5、手动构建

上述配置完成后，返回到工程的首页，点击立即构建







到此，Jenkins基础配置就完成了，后续有更多的需求时，可以再根据实际情况进行调整和配置即可