

# 一、jenkins安装

jenkins在Linux上的

## 1、安装jdk1.8

### 1-1 网上下载 源文件

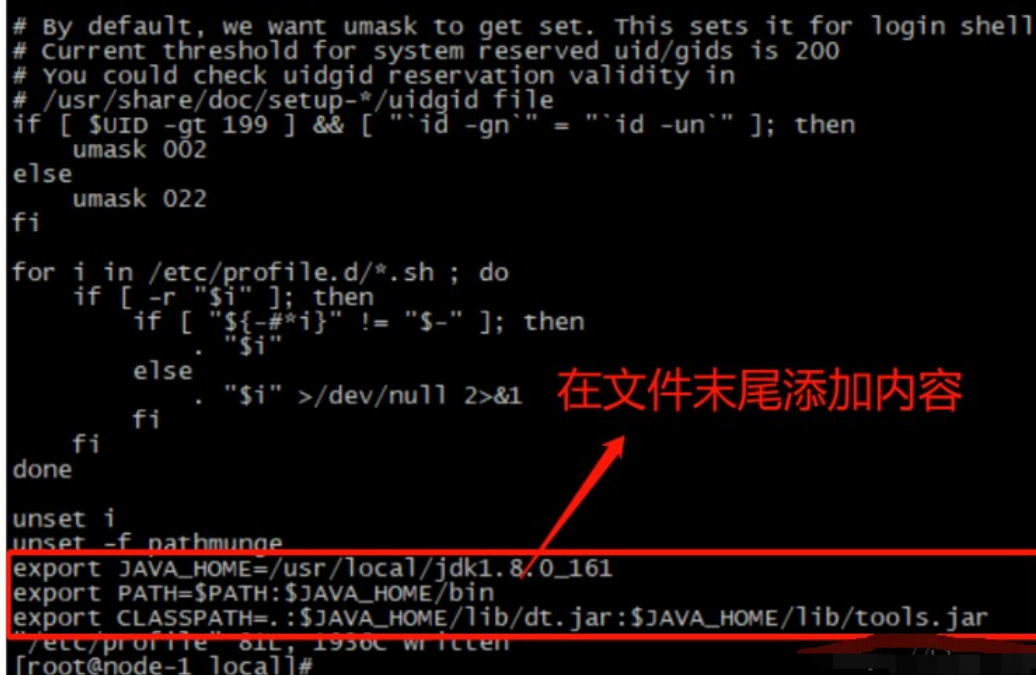
jdk-8u161-linux-x64.tar.gz ( <https://pan.baidu.com/s/18QGn2Tc2kDNJVflqwfgghA> )

### 1-2 解压

```
tar -zxvf jdk-8u161-linux-x64.tar.gz
```

### 1-3 配置jdk环境

```
vim /etc/profile
```



```
# By default, we want umask to get set. This sets it for login shell
# Current threshold for system reserved uid/gids is 200
# You could check uidgid reservation validity in
# /usr/share/doc/setup-*/uidgid file
if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then
    umask 002
else
    umask 022
fi

for i in /etc/profile.d/*.sh ; do
    if [ -r "$i" ]; then
        if [ "${-#*i}" != "$-" ]; then
            . "$i"
        else
            . "$i" >/dev/null 2>&1
        fi
    fi
done

unset i
unset -f pathmunge
export JAVA_HOME=/usr/local/jdk1.8.0_161
export PATH=$PATH:$JAVA_HOME/bin
export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
"/etc/profile" 81L, 1936C written
[root@node-1 local]#
```

在文件末尾添加内容

随后在profile文件的末尾添加如下内容：

```
1 export JAVA_HOME=/usr/local/jdk1.8.0_161
2 export PATH=$PATH:$JAVA_HOME/bin
3 export CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
```

生效配置文件

```
source /etc/profile
```

## 1-4 验证jdk安装是否成功

```
java -version
```

```
[root@... ]# java -version
java version "1.8.0_161"
Java(TM) SE Runtime Environment (build 1.8.0_161-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.161-b12, mixed mode)
```

## 2、运行jenkins.war 包

### 2-1 jenkins.war下载

<https://jenkins.io/doc/book/installing/>

```
docker exec -it jenkins-blueocean bash
```

#### WAR file

The Web application ARchive (WAR) file version of Jenkins can be installed on any operating system or platform that supp

To download and run the WAR file version of Jenkins:

1. Download the [latest stable Jenkins WAR file](#) to an appropriate directory on your machine.
2. Open up a terminal/command prompt window to the download directory.
3. Run the command `java -jar jenkins.war`.
4. Browse to <http://localhost:8080> and wait until the **Unlock Jenkins** page appears.
5. Continue on with the [Post-installation setup wizard](#) below.

Notes:

或者地址：<http://mirrors.jenkins-ci.org/>，打开链接后，表格有war列，Releases行是短期更新包。LTS是长期更新包。一般选择Releases下载即可。

### 2-2 运行

```
nohup java -jar jenkins.war --httpPort=8090 > myout.file 2>&1 &
```

### 2-3 安装默认插件

略

## 二、使用

# 1、python程序自动发布

## 1-1、导出python包依赖

```
pip freeze > requirements.txt
```

## 1-2、配置构建远程目标服务器（需要安装插件 SSH plugin 、 Publish Over SSH ）

首要条件：远程目标服务器要安装docker服务

### 1-2-1 配置远程执行脚本的服务器

系统管理->SSH remote hosts

SSH remote hosts

SSH sites

Hostname

Port

Credentials

Pty ☐

serverAliveInterval

timeout

Check connection

### 1-2-2 配置上传文件的远程服务器

系统管理->Publish over SSH

Publish over SSH

Jenkins SSH Key

Passphrase  [Change Password](#)

Path to key

Key

Disable exec ☐

SSH Servers ☐

SSH Server Name

Hostname

Username

Remote Directory

☒ Use password authentication, or use a different key

Passphrase / Password  [Change Password](#)

Path to key

Key

## 1-3、新建任务

## 1-4、配置git来源

源码管理

☐ 无  
☒ Git

Repositories

Repository URL:

Credentials:  git相关的凭证

高级...

Add Repository

Branches to build

指定分支 (为空格代表any):  增加分支

源代码浏览器: (自动)

## 1-5、配置项目的版本号 ( 需要安装插件 Version Number Plugin )

```
1 ${BUILD_DATE_FORMATTED, "yyyyMMdd"}_${BUILDS_TODAY}
```

☒ Create a formatted version number

Environment Variable Name:

Version Number Format String:

Prefix Variable:

Skip Builds worse than:

Don't increment builds today / this week / this month / this year / all time after a build-run with a result worse than the selected one.

Build Display Name: ☐ Use the formatted version number for build display name.

Project Start Date:

Number of builds today:

Number of builds this week:

Number of builds this month:

Number of builds this year:

## 1-6、配置构建

目前有两种构建方式：

1) 本地**构建Dockerfile文件并打包\*\*\*\*上传**项目文件到目标服务器再进行**构建镜像、运行容器**。（打包->上传->目标构建->运行）

2) 本地**构建Dockerfile文件并构建镜像**，再通过docker push方式**上传**；目标服务器

通过docker pull**拉取**镜像，并**运行容器**。(构建->上传->目标拉取->运行)

缺点：构建程序的依赖包过大导致镜像过大，上传会比较慢，**耗时**。

优点：保证每个目标服务器拉取的镜像都是一样的。

## 1-6-1、目标服务器构建镜像

### 1-6-1-1、本地构建shell脚本

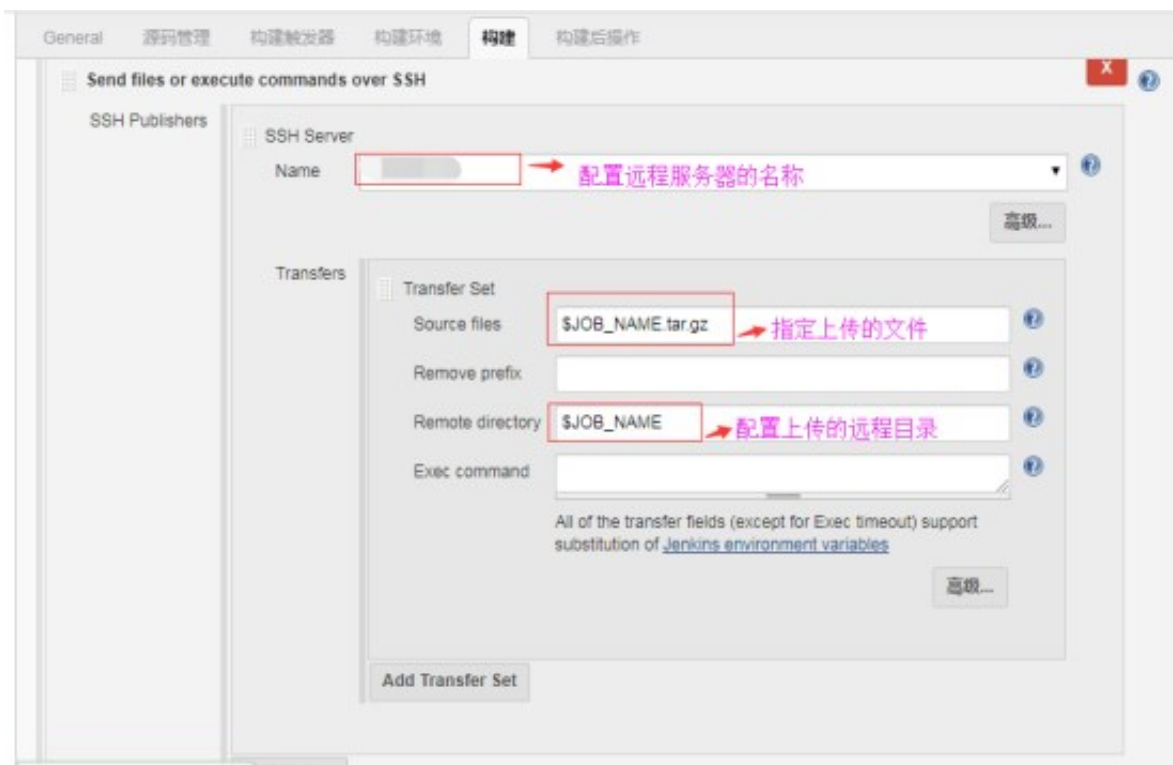
本地构建dockerfile并打包项目文件。脚本代码如下：

```
1 echo "FROM python:3.6
2 RUN rm -rf /project
3 ADD ./ /project/
4 RUN pip3 install --upgrade pip && pip3 install -r /project/requirements.tx
5 ENV PYTHONPATH=$PYTHONPATH:/project/
6
7 RUN ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
8 #公开端口
9 EXPOSE 888
10 #设置启动命令
11 CMD python XXX.py
12 " > Dockerfile
13 tar zcf $JOB_NAME.tar.gz *
14
```



### 1-6-1-2、配置远程目标服务器

配置要远程上传的文件跟目标服务器存放的路径。



### 1-6-1-3、执行远程服务器的shell脚本



run.sh脚本代码如下：

xxxx是上传文件的目录

```

1 cd /xxxx/${1}/
2 tar -zxvf ${1}.tar.gz -C ./
3 rm -rf ${1}.tar.gz
4 docker build -t ${1}:${2} .
5 # 容器在运行
6 if [[ `docker ps | grep ${1}_demo` ]];
7 then
8     # 停止运行中的容器
9     docker stop ${1}_demo
10 fi;
11 # 启动
12 if [[ `docker ps -a | grep ${1}_demo` ]];

```

```
13 then
14     # 容器已创建，只重启
15     docker rm ${1}_demo
16 fi;
17 docker run -d --name ${1}_demo -p ${3}:${4} -v /xxxx:/xxxx ${1}:${2}
```

## 1-6-2、本地构建镜像

本地构建镜像，通过docker push到远程服务器。

### 1-6-2-1、本地构建shell脚本

本地构建dockerfile文件。脚本代码如下：

```
1 echo "FROM python:3.6
2 ADD ./ /project/
3
4 RUN pip3 install --upgrade pip && pip3 install -r /project/requirements.tx
5 -i http://mirrors.aliyun.com/pypi/simple/ --trusted-host mirrors.aliyun.cc
6 echo "FROM $JOB_NAME:base
7 RUN rm -rf /project
8 ADD ./ /project/
9 RUN pip3 install --upgrade pip && pip3 install -r /project/code/requiremen
10 -i http://mirrors.aliyun.com/pypi/simple/ --trusted-host mirrors.aliyun.cc
11 ENV PYTHONPATH=$PYTHONPATH:/project/code
12 RUN ln -sf /usr/share/zoneinfo/Asia/Shanghai /etc/localtime
13 #公开端口
14 EXPOSE 80
15 #设置启动命令
16 CMD python xxxx.py
17 " > Dockerfile
```



## 1-6-2-2、本地运行shell脚本

执行镜像的构建脚本。



本地镜像构建。脚本代码如下

baseDir:项目存放数据的目录（自定义）

```

1 rm -rf baseDir/${1}
2 mkdir baseDir/${1}
3 mkdir baseDir/${1}/code
4 docker cp jenkins:./var/jenkins_home/workspace/${1} baseDir/${1}/code
5 #docker cp jenkins:./var/jenkins_home/workspace/${1}/requirements.txt ./
6 mv baseDir/${1}/code/${1}/* baseDir/${1}/code/
7
8 mkdir baseDir/${1}/base
9 mv baseDir/${1}/code/Dockerfile_base baseDir/${1}/base
10 mv baseDir/${1}/base/Dockerfile_base baseDir/${1}/base/Dockerfile
11 cp baseDir/${1}/code/requirements.txt baseDir/${1}/base
12 # 镜像是否存在
13 images_exists=`docker images | grep ${1}`

```



```

14 if [[ -n $images_exists ]];then
15 echo "exists"
16 else
17 echo "no exists"
18 cd baseDir/${1}/base
19 docker build -t ${1}:base .
20 fi;
21
22
23 rm -rf baseDir/${1}/code/${1}
24 mv baseDir/${1}/code/Dockerfile baseDir/${1}
25 cd baseDir/${1}
26 rm -rf $basedir/${1}
27 docker build -t ${1}:${2} .
28 docker tag ${1}:${2} ip:5000/${1}:${2}
29 docker push ip:5000/${1}:${2}
30

```

### 1-6-2-3、远程运行shell脚本



主要用来拉取镜像及容器的运行。

Run.sh脚本代码如下：

```

1 # 容器在运行
2 if [[ `docker ps | grep ${1}_demo` ]];
3 then
4     # 停止运行中的容器
5     docker stop ${1}_demo
6 fi;
7
8 # 启动
9 if [[ `docker ps -a | grep ${1}_demo` ]];
10 then

```

```

11      # 容器已创建，只重启
12      docker rm ${1}_demo
13  fi;
14
15  if [[ `docker images | grep ${1}` ]];
16  then
17      # 删除镜像
18      docker rmi `docker images | grep ${1} | awk '{print $1":"$2}'`
19
20  fi;
21  docker push 仓库ip:5000/${1}:${4}
22  docker run -d --name ${1}_demo -p ${2}:${3} 仓库ip:5000/${1}:${4}

```

## 自动构建

### 1 ) Build periodic ally与Poll SCM

Build periodic ally：周期性的执行，源码有没有变化都会执行

比如配置：H/60 \* \* \* \* 这样配置就会每60分钟构建一次，不管SVN有没有新源码

Poll SCM：定时行的执行，源码有变化才会执行

比如配置：/10 \* \* \* 这样配置就会10分钟去检查svn是否有新源码，有就checkout，构建，没有就继续去潇洒，10分钟后再回来检查。

### ( 2 ) 配置参考

每10分钟构建一次：H/10 \* \* \* \* 或/10 \* \* \* \*

每天8点构建一次：0 8 \* \* \* \*

每天8点~17点，两小时构建一次：0 8-17/2 \* \* \* \*

周一到周五，8点~17点，两小时构建一次：0 8-17/2 \* \* 1-5

1-6月中每月1号、30号各构建一次：H H 1,30 1-6 \*

每周五18点构建一次：00 18 \* \* 5