

三、Docker安装Jenkins（和第二部分2选1）

1.docker安装

2.docker安装部署jenkins

3.插件安装

4.容器进入、保存；安装vim、wget等

5.安装JDK

6.maven安装

7.Git安装

8.NodeJs安装

9.jenkins配置

三、Docker安装Jenkins（和第二部分2选1）

1.docker安装

1) 准备工作

Docker 要求 CentOS 系统的内核版本高于 3.10，查看本页面的前提条件来验证你的CentOS 版本是否支持 Docker。

通过 `uname -r` 命令查看你当前的内核版本

```
1  uname -r
```

卸载旧版本(如果安装过旧版本的话)

```
1  sudo yum remove docker docker-common docker-selinux docker-engine
```

2) 可以查看所有仓库中所有docker版本，并选择特定版本安装

```
1  yum list docker-ce --showduplicates | sort -r
```

3) 安装docker

```
1  sudo yum install docker-ce
```

特定版本安装：

```
1  sudo yum install docker-ce-17.12.0.ce
```

4) 验证安装是否成功(有client和service两部分表示docker安装启动都成功了)

```
1  docker version
```

```
[root@localhost ~]# docker version
Client: Docker Engine - Community
 Version:      20.10.12
 API version:  1.41
 Go version:   go1.16.12
 Git commit:   e9led57
 Built:        Mon Dec 13 11:45:41 2021
 OS/Arch:     linux/amd64
 Context:      default
 Experimental: true

Server: Docker Engine - Community
 Engine:
  Version:      20.10.12
  API version:  1.41 (minimum version 1.12)
  Go version:   go1.16.12
  Git commit:   459d0df
  Built:        Mon Dec 13 11:44:05 2021
  OS/Arch:     linux/amd64
  Experimental: false
 containerd:
  Version:      1.4.12
  GitCommit:    7b11cfaabd73bb80907dd23182b9347b4245eb5d
 runc:
  Version:      1.0.2
  GitCommit:    v1.0.2-0-g52b36a2
 docker-init:
  Version:      0.19.0
  GitCommit:    de40ad0
```

```
1 docker run hello-world
```

```
[root@localhost ~]# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:975f4b14f326b05db86e16de00144f9c12257553bba9484fed41f9b6f2257800
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

5) 启动，加入开机启动

```
1 sudo systemctl start docker #启动
2 sudo systemctl enable docker #开机自启
```

验证是否开启成功，reboot重启后验证是否自启

```
1 systemctl status docker
```

```
[root@localhost ~]# systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since 四 2022-02-17 09:34:47 CST; 6 days ago
     Docs: https://docs.docker.com
    Main PID: 1372 (dockerd)
    CGroup: /system.slice/docker.service
            └─1372 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

2月 17 09:34:43 localhost.localdomain dockerd[1372]: time="2022-02-17T09:34:43.794109469+08:00" level=info msg="Firewal...ning"
2月 17 09:34:45 localhost.localdomain dockerd[1372]: time="2022-02-17T09:34:45.614507802+08:00" level=info msg="Firewal...ning"
2月 17 09:34:46 localhost.localdomain dockerd[1372]: time="2022-02-17T09:34:46.406929374+08:00" level=info msg="Default...ress"
2月 17 09:34:46 localhost.localdomain dockerd[1372]: time="2022-02-17T09:34:46.906582397+08:00" level=info msg="Firewal...ning"
2月 17 09:34:47 localhost.localdomain dockerd[1372]: time="2022-02-17T09:34:47.548886329+08:00" level=info msg="Loading...one."
2月 17 09:34:47 localhost.localdomain dockerd[1372]: time="2022-02-17T09:34:47.679541112+08:00" level=info msg="Docker ...10.12
2月 17 09:34:47 localhost.localdomain dockerd[1372]: time="2022-02-17T09:34:47.681080843+08:00" level=info msg="Daemon ...tion"
2月 17 09:34:47 localhost.localdomain systemd[1]: Started Docker Application Container Engine.
2月 17 09:34:47 localhost.localdomain dockerd[1372]: time="2022-02-17T09:34:47.901188752+08:00" level=info msg="API lis...sock"
2月 23 11:08:52 localhost.localdomain dockerd[1372]: time="2022-02-23T11:08:52.210875770+08:00" level=info msg="ignorin...lete"
Hint: Some lines were ellipsized, use -l to show in full.
[root@localhost ~]#
```

6) 其他关闭、重启docker等操作

- 1 sudo systemctl daemon-reload #守护进程重启
- 2 systemctl restart docker #重启docker服务
- 3 sudo service docker restart #重启docker服务
- 4 docker service docker stop #关闭
- 5 docker systemctl stop docker #关闭

2.docker安装部署jenkins

1) 下载jenkins镜像

- 1 docker pull jenkins/jenkins

可以去这里获取你需要的版本: https://hub.docker.com/_/jenkins?tab=tags

也可以指定版本下载, 例如:

docker pull jenkins/jenkins:2.326 #指定版本

查看本地镜像, 检测镜像是否下载成功

- 1 docker images

```
[root@192 ~]# docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
jenkins/jenkins     latest         97d23cbbfa56   3 days ago     463MB
hello-world         latest         feb5d9fea6a5   5 months ago   13.3kB
docker/compose      1.8.0          89188432ef03   5 years ago    59.1MB
[root@192 ~]#
```

如果出现报错:

failed to register layer: Error processing tar file(exit status 1): symlink London
/usr/share/zoneinfo/right/Europe/Belfast: no space left on device

是因为空间不足, 建议虚拟机硬盘空间至少20G, 推荐40G。参考下面两篇笔记
文档: **Docker 拉取镜像报错 failed to regist...**

链接: [http://note.youdao.com/noteshare?](http://note.youdao.com/noteshare?id=b540493979794c7a3a6f5a77158b8132&sub=108571F440B44708BAD7413CCBF0E78D)

id=b540493979794c7a3a6f5a77158b8132&sub=108571F440B44708BAD7413CCBF0E78D

文档: **linux虚拟机扩容.note**

链接: [http://note.youdao.com/noteshare?](http://note.youdao.com/noteshare?id=622b416934128131877678cfd2d78627&sub=3F41F59ED9894EC4848D1CACDF031B12)

id=622b416934128131877678cfd2d78627&sub=3F41F59ED9894EC4848D1CACDF031B12

2) 创建本地数据卷

创建/mnt/sdc/jenkins_home目录, 为下一步创建容器规划好存储空间。

目录可以起一个自己喜欢或者习惯的名字, 例如: **docker/jenkins_home**, 我当时只跟着做, 起名字这块没在意, 后续的容器挂载目录我统一用的docker/**

```
1 mkdir -p /docker/jenkins_home
```

目录设置权限。3种不同类型: 文件所有者、群组用户、其他用户的 权限都设为7。

```
1 chmod 777 /docker/jenkins_home
```

3) 创建容器

```
1 docker run --restart=always -d -p 8099:8080 -p 50001:50000 -e JENKINS_JAVA_OPTIONS="-XX:MaxPermSize=1024m -Djava.awt.headless=true" -v /docker/jenkins_home:/var/jenkins_home/ -u 0 --name=jenkins jenkins/jenkins
```

创建命令解释:

--restart=always 开机启动, 失败也会一直重启

-restart具体参数值详细信息:

no - 容器退出时, 不重启容器;

on-failure - 只有在非0状态退出时才从新启动容器;

always - 无论退出状态是如何, 都重启容器;

-d 标识是让 docker 容器在后台运行

-p 8099:8080 端口映射, 将镜像的8080端口映射到服务器的8089端口

-p 50001:50000 端口映射, 将镜像的50000端口映射到服务器的50001端口

-e JENKINS_JAVA_OPTIONS="-XX:MaxPermSize=1024m -Djava.awt.headless=true"

传递环境变量

-v /docker/jenkins_home:/var/jenkins_home/ 绑定一个数据卷, docker/jenkins_home 是刚才创建的本地数据卷 (**挂载**)

-v /etc/localtime:/etc/localtime 让容器使用和服务器同样的时间设置 (这句我没加)

--name 定义一个容器的名字, 如果没有指定, 那么会自动生成一个随机数字字符串当做uuid

jenkins/jenkins 刚才下载的镜像名称

```
[root@192 ~]# docker run --restart=always -d -p 8099:8080 -p 50001:50000 -e JENKINS_JAVA_OPTIONS="-XX:MaxPermSize=1024m -Djava.awt.headless=true" -v /mnt/sdc/jenkins_home:/var/jenkins_home/ -u 0 --name=jenkins jenkins/jenkins
WARNING: IPv4 forwarding is disabled. Networking will not work.
```

注意：这里有warning警告！没有网络

WARNING: IPv4 forwarding is disabled. Networking will not work.

解决办法：

编辑 /etc/sysctl.conf 文件

```
1 vim /etc/sysctl.conf
```

添加这段代码，保存退出

```
1 net.ipv4.ip_forward=1 #配置转发
```

```
# sysctl settings are defined through files in
# /usr/lib/sysctl.d/, /run/sysctl.d/, and /etc/sysctl.d/.
#
# Vendors settings live in /usr/lib/sysctl.d/.
# To override a whole file, create a new file with the same in
# /etc/sysctl.d/ and put new settings there. To override
# only specific settings, add a file with a lexically later
# name in /etc/sysctl.d/ and put new settings there.
#
# For more information, see sysctl.conf(5) and sysctl.d(5).
#
net.ipv4.ip_forward=1 #配置转发
```

重启服务，让配置生效

```
1 systemctl restart network
```

查看是否成功,如果返回为 “net.ipv4.ip_forward = 1” 则表示成功

```
1 sysctl net.ipv4.ip_forward
```

重启docker

```
1 systemctl restart docker
```

我到这里就可以了，如果还不行，网上教程还有重构镜像的，自行百度吧

4) 查看jenkins是否启动成功

```
1 docker ps -l #查看最近一个的容器, docker ps 查看所有运行容器
```

```
[root@l92 ~]# docker ps -l
```

CONTAINER ID	IMAGE	COMMAND	NAMES	CREATED	STATUS	PORTS
798eec1b768f	jenkins/jenkins	"/sbin/tini -- /usr/_"		2 minutes ago	Up 2 minutes	0.0.0.0:8099->8080/tcp, :::8099->8080/tcp

```
cp, 0.0.0.0:50001->50000/tcp, :::50001->50000/tcp jenkins
[root@l92 ~]#
```

有这个端口号就是创建成功了

想通过浏览器访问，需要开放8099端口，参照第一部分准备工作中的端口开放。

端口开放后访问jenkins，证明jenkins启动成功

解锁 Jenkins

为了确保管理员安全地安装 Jenkins，密码已写入到日志中（[不知道在哪里?](#)）该文件在服务器上：

```
/var/jenkins_home/secrets/initialAdminPassword
```

请从本地复制密码并粘贴到下面。

管理员密码

注意：这里的提示地址是容器内部根路径，不是虚拟机根路径。有两个方法获取密码

方法1：

先进入容器后再查看路径下文件。

```
1 docker exec -it jenkins /bin/bash #进入容器
2 cat /var/jenkins_home/secrets/initialAdminPassword #查看密码文件
```

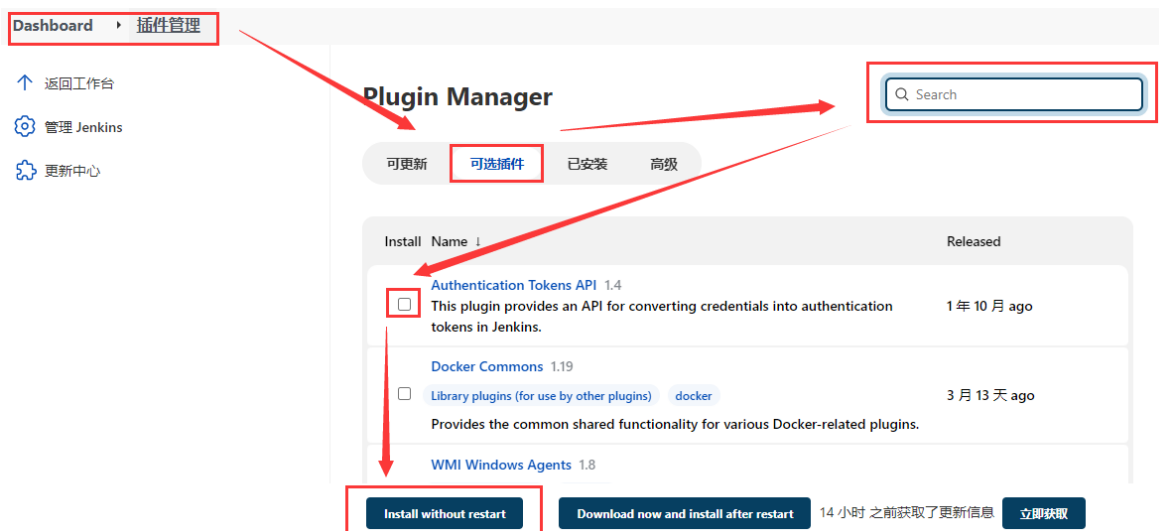
方法2：

在创建容器的时候，/docker/jenkins_home是挂载目录，可以直接查看

```
1 cat /docker/jenkins_home/secrets/initialAdminPassword #不进入容器直接查看
```

3.插件安装

【系统管理】==>【管理插件】==>【可选插件】==>搜索***插件==>点击 install without restart



绝对必要插件：我开始没这些插件，后面会逐一例举哪里需要安装此插件

NodeJS

Publish Over SSH

SSH

Maven Integration

Git Parameter

Config File Provider

应该需要插件：我已经有这些插件了，没法验证在哪里用到

Localization: Chinese (Simplified)

Pipeline

Workspace Cleanup

Folders

Subversion

4.容器进入、保存；安装vim、wget等

查看容器

```
1 docker ps #查看已运行容器
```

```
[root@l92 ~]# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
798eec1b768f   jenkins/jenkins  "/sbin/tini -- /usr/_..."  5 weeks ago   Up 6 hours   0.0.0.0:8099->8080/tcp, :::8099->8080/tcp, 0.0.0.0:50001->50000/tcp, :::50001->50000/tcp
NAME          jenkins
```

进入容器

```
1 docker exec -it <容器 ID> /bin/bash #进入容器
```

```
[root@l92 ~]# docker exec -it 798eec1b768f /bin/bash
root@798eec1b768f:/#
```

安装vi、wget、yum、ifconfig、ping、make

不安装会报错：bash: vim: command not found

```
1 apt-get update ##更新
2 //vi
3 apt install vim
4 //wget
5 #apt install wget 会报错: E: Unable to locate package wget
6 apt-get -y install wget
7 //ifconfig
8 apt install net-tools
9 //ping
10 apt install iputils-ping
11 //make
12 apt-get install gcc automake autoconf libtool make
```

5.安装JDK

0) 镜像自带的jdk版本11, 然而我们需要8版本

因为镜像自带jdk, 版本11, 直接检查是否安装

```
1 java -version
```

```
root@798eeclb768f:/# java -version
openjdk version "11.0.14.1" 2022-02-08
OpenJDK Runtime Environment Temurin-11.0.14.1+1 (build 11.0.14.1+1)
OpenJDK 64-Bit Server VM Temurin-11.0.14.1+1 (build 11.0.14.1+1, mixed mode)
```

获取jdk安装环境地址

```
1 echo $JAVA_HOME
```

```
root@798eeclb768f:/# echo $JAVA_HOME
/opt/java/openjdk
```

这个11版本的jdk一定不能删, 删了jenkins容器跑不起来, 我当时为啥要删它...

```
2022-05-24 09:04:53.392+0000 [id=31] INFO hudson.lifecycle.Lifecycle#onStatusUpdate: Stopping Jenkins
/usr/local/bin/jenkins.sh: line 48: exec: java: not found
/usr/local/bin/jenkins.sh: line 48: exec: java: not found
/usr/local/bin/jenkins.sh: line 48: exec: java: not found
/usr/local/bin/jenkins.sh: line 48: exec: java: not found
```

这里新安装个1.8版本的

1) 容器里创建下载文件目录

```
1 mkdir -p /usr/local/download/
```

2) 打开url选择jdk1.8下载

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

选择linux x64版本:

Linux macOS Solaris Windows		
Product/file description	File size	Download
ARM 64 RPM Package	59.27 MB	jdk-8u321-linux-aarch64.rpm
ARM 64 Compressed Archive	71.02 MB	jdk-8u321-linux-aarch64.tar.gz
ARM 32 Hard Float ABI	73.71 MB	jdk-8u321-linux-arm32-vfp-hflt.tar.gz
x86 RPM Package	110.21 MB	jdk-8u321-linux-i586.rpm
x86 Compressed Archive	139.62 MB	jdk-8u321-linux-i586.tar.gz
x64 RPM Package	109.97 MB	jdk-8u321-linux-x64.rpm
x64 Compressed Archive	140.01 MB	jdk-8u321-linux-x64.tar.gz

wget下载文件到 /usr/local/java目录

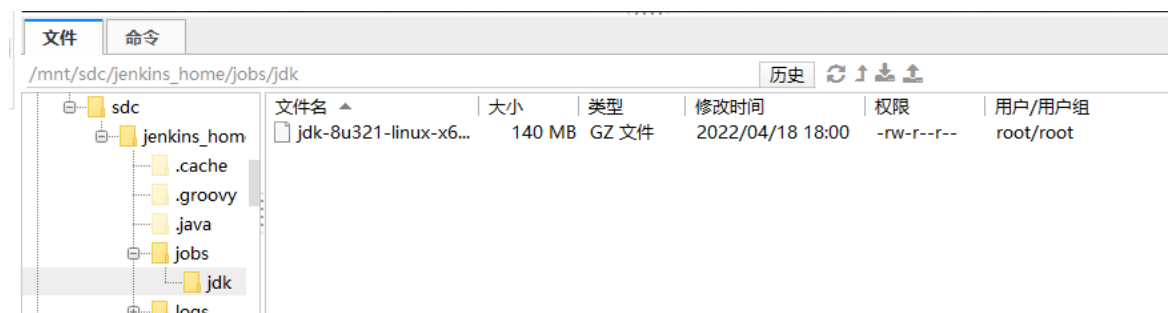
```
1 wget -P /usr/local/download https://download.oracle.com/otn/java/jdk/8u321-b07/df5ad55fdd604472a86a45a217032c7d/jdk-8u321-linux-x64.tar.gz?AuthParam=1642658927_4853323df1be2aa63c1cfad1be08738f
```

这里提示403禁止, 可直接下载好, 用Xftp传送过去

往容器传输文件有多种, 例如: 【挂载】【cp命令】, 可自行百度学习。

因为容器在创建的时候，已经在宿主机挂载了数据卷：/docker/jenkins_home
Xftp把下载的文件传输到 /docker/jenkins_home/jobs/jdk

```
1 /docker/jenkins_home/jobs/jdk
```



进入容器对应的目录下，看看是否有该安装文件

```
1 cd /var/jenkins_home/jobs/jdk
2 ls
```

```
root@798eec1b768f:/var# cd /var/jenkins_home/jobs/jdk
root@798eec1b768f:/var/jenkins_home/jobs/jdk# ls
jdk-8u321-linux-x64.tar.gz
root@798eec1b768f:/var/jenkins_home/jobs/jdk#
```

这里能看到容器里有jdk安装文件了

3) 下载以后通过命令检查安装包大小是否符合，（在安装包文件目录下执行）

```
1 ls -lht
```

4) 解压至安装目录，（在安装包文件目录下执行）

```
1 mkdir -p /usr/local/java/ #新建java文件夹目录
2 cd /var/jenkins_home/jobs/jdk
3 tar -zxvf jdk-8u321-linux-x64.tar.gz -C /usr/local/java/
```

5) 设置环境变量

打开文件

```
1 vim /etc/profile
```

在末尾添加

```
1 # setting jdk path
2 export JAVA_HOME=/usr/local/java/jdk1.8.0_321
3 export JRE_HOME=${JAVA_HOME}/jre
4 export CLASSPATH=.:${JAVA_HOME}/lib:${JRE_HOME}/lib
5 export PATH=${JAVA_HOME}/bin:$PATH
```

使环境变量生效

```
1 source /etc/profile
```

检查

```
1 java -version
```

```
root@798eeclb768f:/var/jenkins_home# java -version
java version "1.8.0_321"
Java(TM) SE Runtime Environment (build 1.8.0_321-b07)
Java HotSpot(TM) 64-Bit Server VM (build 25.321-b07, mixed mode)
```

6) 遇到问题：容器重启后，java版本又回到默认的11

解决办法：

容器加载时会先从/root/.bashrc 中加载环境变量，而/root/.bashrc 实际调用的是/etc/bashrc 因此在/etc/bashrc 文件中加入刷新命令，可以使容器每次加载时都会自动刷新环境变量，因此解决重启环境变量失效的问题。并且不会因非root用户登录造成不可用

```
1 vi /root/.bashrc
```

在文件末尾加上

```
1 # 直接在这里新增上面提到的刷新代码
2 source /etc/profile
```

然后保存退出，重启容器进行验证

6.maven安装

1) 下载

容器里创建下载文件目录（已经有了会提示已存在）

```
1 mkdir -p /usr/local/download/
```

下载/usr/local/download目录

```
1 wget -P /usr/local/download
https://mirrors.tuna.tsinghua.edu.cn/apache/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz
```

也可以在浏览器去maven官网<http://maven.apache.org/download.cgi>

下载需要的版本，这里安装的是二进制包，所以选择“-bin.tar.gz”结尾的包

2) 解压

```
1 cd /usr/local/download
2 tar -xf apache-maven-3.6.3-bin.tar.gz -C /usr/local/
```

修改文件名为maven3.6

```
1 mv /usr/local/apache-maven-3.6.3/ /usr/local/maven3.6
```

3) 加入环境变量

修改配置文件

```
1 vi /etc/profile
```

文件最下方加入新的一行

```
1 # setting maven path
2 export PATH=$PATH:/usr/local/maven3.6/bin
```

让环境变量生效

```
1 source /etc/profile
```

验证：

```
1 which mvn
```

显示/usr/local/maven3.6/bin/mvn就说明配置成功了

4) JAVA环境

运行maven需要Java环境----系统安装有jdk，并且在系统中配置了JAVA_HOME。

7.Git安装

1) apt-get安装git

```
1 apt-get install git
```

这里会提示已经安装了git

2) 检查验证git是否安装

```
1 git --version
```

8.NodeJs安装

1) 进入NodeJs官网，获取到nodeJs的安装地址（在linux下的安装地址），也可以wget下载

附上地址：<https://npm.taobao.org/mirrors/node/v15.5.0/>

这里安装的是15.5版本的。

node-v15.5.0-darwin-x64.tar.gz	22-Dec-2020 18:03	30427153(29.02MB)
node-v15.5.0-darwin-x64.tar.xz	22-Dec-2020 18:04	20428464(19.48MB)
node-v15.5.0-headers.tar.gz	22-Dec-2020 17:50	605749(591.55kB)
node-v15.5.0-headers.tar.xz	22-Dec-2020 17:50	396272(386.98kB)
node-v15.5.0-linux-arm64.tar.gz	22-Dec-2020 18:20	32576813(31.07MB)
node-v15.5.0-linux-arm64.tar.xz	22-Dec-2020 18:22	21159396(20.18MB)
node-v15.5.0-linux-armv7l.tar.gz	22-Dec-2020 17:41	30328160(28.92MB)
node-v15.5.0-linux-armv7l.tar.xz	22-Dec-2020 17:41	18585680(17.72MB)
node-v15.5.0-linux-ppc64le.tar.gz	22-Dec-2020 16:56	34591514(32.99MB)
node-v15.5.0-linux-ppc64le.tar.xz	22-Dec-2020 16:58	22306032(21.27MB)
node-v15.5.0-linux-s390x.tar.gz	22-Dec-2020 17:03	32847823(31.33MB)
node-v15.5.0-linux-s390x.tar.xz	22-Dec-2020 17:03	20795908(19.83MB)
node-v15.5.0-linux-x64.tar.gz	22-Dec-2020 17:04	32550517(31.04MB)
node-v15.5.0-linux-x64.tar.xz	22-Dec-2020 17:05	21775092(20.77MB)
node-v15.5.0-win-x64.7z	22-Dec-2020 16:57	17236870(16.44MB)
node-v15.5.0-win-x64.zip	22-Dec-2020 16:57	26661803(25.43MB)
node-v15.5.0-win-x86.7z	22-Dec-2020 16:49	16074415(15.33MB)
node-v15.5.0-win-x86.zip	22-Dec-2020 16:49	24999533(23.84MB)
node-v15.5.0-x64.msi	22-Dec-2020 16:57	28954624(27.61MB)
node-v15.5.0-x86.msi	22-Dec-2020 16:49	27197440(25.94MB)
node-v15.5.0.pkg	22-Dec-2020 17:28	30711035(29.29MB)
node-v15.5.0.tar.gz	22-Dec-2020 17:42	62299813(59.41MB)
node-v15.5.0.tar.xz	22-Dec-2020 17:47	33379364(31.83MB)
SHASUMS256.txt	22-Dec-2020 18:57	2929(2.86kB)
SHASUMS256.txt.asc	22-Dec-2020 18:57	3811(3.72kB)
SHASUMS256.txt.sig	22-Dec-2020 18:57	566(566B)

2) 安装NodeJs

容器里创建下载文件目录

```
1 mkdir -p /usr/local/download/
```

下载到/usr/local/download/目录

```
1 wget -P /usr/local/download  
https://registry.npmirror.com/-/binary/node/v15.5.0/node-v15.5.0-linux-x64.tar.gz
```

解压文件

```
1 cd /usr/local/download  
2 tar -zxvf node-v15.5.0-linux-x64.tar.gz -C /usr/local/
```

3) 配置系统环境变量

```
1 vim /etc/profile
```

在profile文件中末尾，添加node的环境变量

```
1 # setting nodejs path
2 export NODE_HOME=/usr/local/node-v15.5.0-linux-x64
3 export PATH=$NODE_HOME/bin:$PATH
```

然后保存退出

使环境变量生效

```
1 source /etc/profile
```

4) 验证是否安装成功

```
1 node -v
```

v15.5.0

```
1 npm -v
```

7.3.0

出现上述结果证明安装成功！

5) 切换淘宝镜像（根据自身习惯情况，也可以不切换）

```
1 npm config set registry http://registry.npm.taobao.org/
```

9.jenkins配置

1) 配置maven

【系统管理】==>【全局工具配置】

/usr/local/maven3.6/conf/settings.xml

Maven 配置

默认 settings 提供

文件系统中的 settings 文件

文件路径 ?

/usr/local/maven3.6/conf/settings.xml

默认全局 settings 提供

文件系统中的全局 settings 文件

文件路径 ?

/usr/local/maven3.6/conf/settings.xml

2) 配置JDK

【系统管理】==>【全局工具配置】

/usr/local/java/jdk1.8.0_321

JDK

JDK 安装

新增 JDK

JDK

别名

JDK1.8

JAVA_HOME

/usr/local/java/jdk1.8.0_321

☐ 自动安装

删除 JDK

新增 JDK

系统下JDK 安装列表

3) 配置git

查看git安装路径

```
1 whereis git
```

```
[root@localhost ~]# whereis git
git: /usr/bin/git /usr/share/man/man1/git.1.gz
[root@localhost ~]#
```

点击【系统管理】==>【全局工具配置】

/usr/bin/git

Git

Git installations

Git

Name

Git

Path to Git executable

user/bin/git

There's no such file: user/bin/git

☐ 自动安装

Delete Git

Add Git

4) 配置maven

【系统管理】==>【全局工具配置】

/usr/local/maven3.6

Maven

Maven 安装

新增 Maven

Maven

Name

Maven3.6

MAVEN_HOME

/usr/local/maven3.6

☐ 自动安装

删除 Maven

新增 Maven

系统下Maven 安装列表

保存 应用

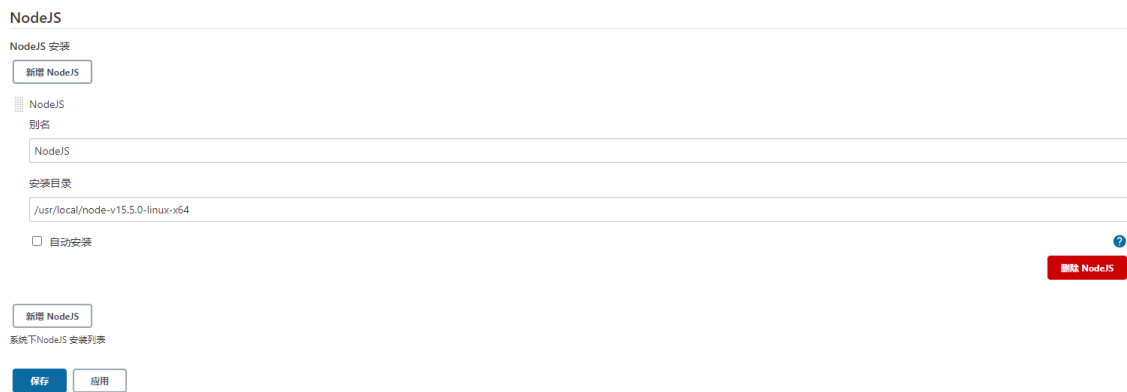
5) 配置nodejs

下载NodeJS插件

【系统管理】==>【管理插件】==>【可选插件】==>搜索NodeJS插件==>点击 install without restart

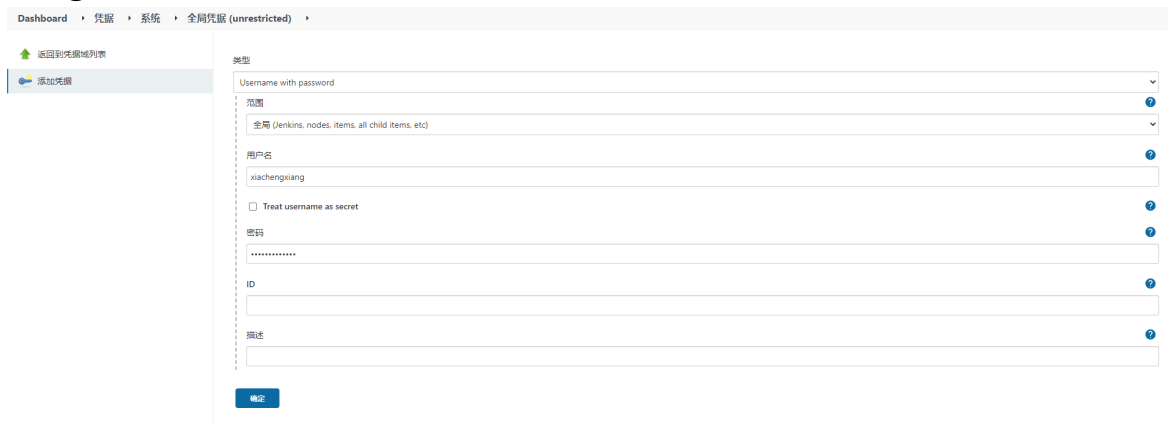
这个时候返回全局配置工具，可以配置NodeJS

【系统管理】==>【全局工具配置】
/usr/local/node-v15.5.0-linux-x64



6) 配置全局凭据

【系统管理】==>【Manage Credentials 凭据管理】==>【全局】==>【添加凭据】
添加git全局凭据



添加远程服务器的全局凭证（有多少添加多少）

没有的话也可以写本机ip配置一个，分布式构建java项目，jar包极有可能部署在不同的服务器

注意：这里能配root就配root，配其他帐号在运行脚本可能出现权限不足



7) 配置SSH（为了在远程服务执行脚本）

下载SSH插件

【系统管理】==>【管理插件】==>【可选插件】==>搜索SSH插件==>点击 install without restart

配置SSH

【系统管理】==>【系统配置】

注意：这里能配root就配root，配其他帐号在运行脚本可能出现权限不足

SSH remote hosts

SSH sites

SSH sites that projects will want to connect

Hostname ?

192.168.16.5

Port ?

22

Credentials

admin (192.168.16.5远程服务帐号密码)

添加

☐ Pty ?

serverAliveInterval ?

0

timeout ?

0

Check connection

新增

8) 配置Publish Over SSH（为了上传文件到远程服务）

下载Publish Over SSH插件，方法1（三选一）

【系统管理】==>【管理插件】==>【可选插件】==>搜索Publish Over SSH插件==>点击 install without restart

*我写这个文档的时候Publish Over SSH插件有漏洞搜索不到了，这里提供别的安装插件思路

下载Publish Over SSH插件，方法2（三选一）

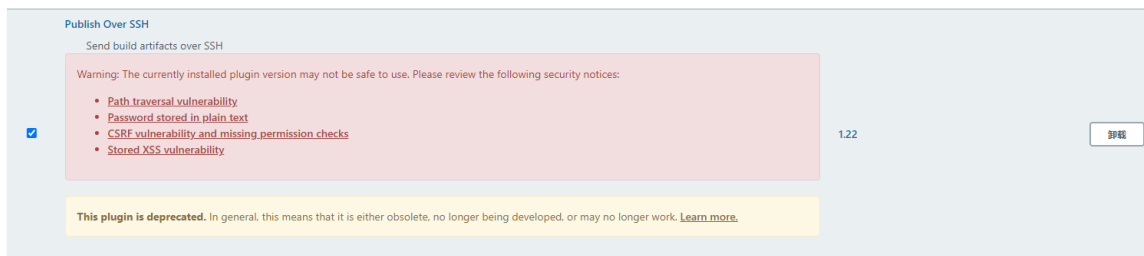
通过其他方式下载Publish Over SSH插件，这里提供一个连接，如果连接失效可寻找其他方式

<http://ftp.yz.yamagata-u.ac.jp/pub/misc/jenkins/plugins/>

下载文件为：publish-over-ssh.hpi

点击 系统管理==>插件管理==>高级，选择本地文件，点击提交

这个时候才去已安装插件搜索，能看到Publish Over SSH插件



下载Publish Over SSH插件，方法3（三选一）

找已经安装了Publish Over SSH插件的同事，把同事jenkins_home目录的plugins文件夹下publish-over、publish-over-ssh、publish-over-ssh.jpi、publish-over.jpi放到自己的相同目录下

配置Publish Over SSH 下的 SSH Server

给其他linux服务器推送文件用的，如果单机部署可不配置

【系统管理】==>【系统配置】

名字最好带着ip，多个ssh的时候后续选择会容易辨认，建议别用汉字，容易乱码

注意：这里能配root就配root，配其他帐号在运行脚本可能出现权限不足

这里的ip，不要http://开头，会连接失败

Remote Directory

☒ Use password authentication, or use a different key

打钩

Passphrase / Password

用户登录密码

Path to key

Key

Jump host

Port

22

Timeout (ms)

300000

☐ Disable exec

Proxy type

Proxy host

Proxy port

Proxy user

Proxy password

Test Configuration

测试

成功示例，会有success提示

Passphrase

Path to key

Key

☐ Disable exec

SSH Servers

SSH Server

Name

本地虚拟机测试

Hostname

192.168.16.5

Username

root

Remote Directory

/

☒ Use password authentication, or use a different key

Passphrase / Password

....

Path to key

Key

Jump host

Port

22

Timeout (ms)

300000

☐ Disable exec

Proxy type

Proxy host

Proxy port

Proxy user

Proxy password

Success

Test Configuration

删除

新增

保存

应用

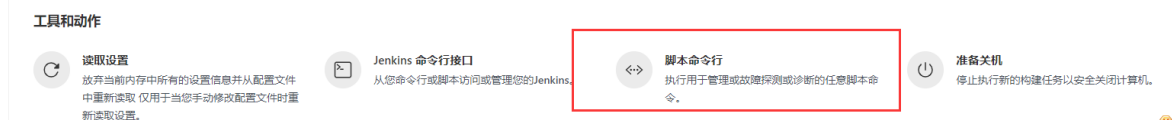
高级...

9) 设置时区

访问【Jenkins】。

点击【Manage Jenkins 系统管理】选项

点击【Script Console 脚本命令行】选项



输入下面脚本点击【运行】

```
1 System.setProperty('org.apache.commons.jelly.tags.fmt.timeZone','Asia/Shanghai')
```