



前言

在上一章中，我们安装了 `Docker` 和 `Jenkins`，并实现了将两者打通。在这一章中，我们则使用 `Jenkins` 集成 `Git` 来构建 `Docker` 镜像，为后面的部署准备镜像资源。

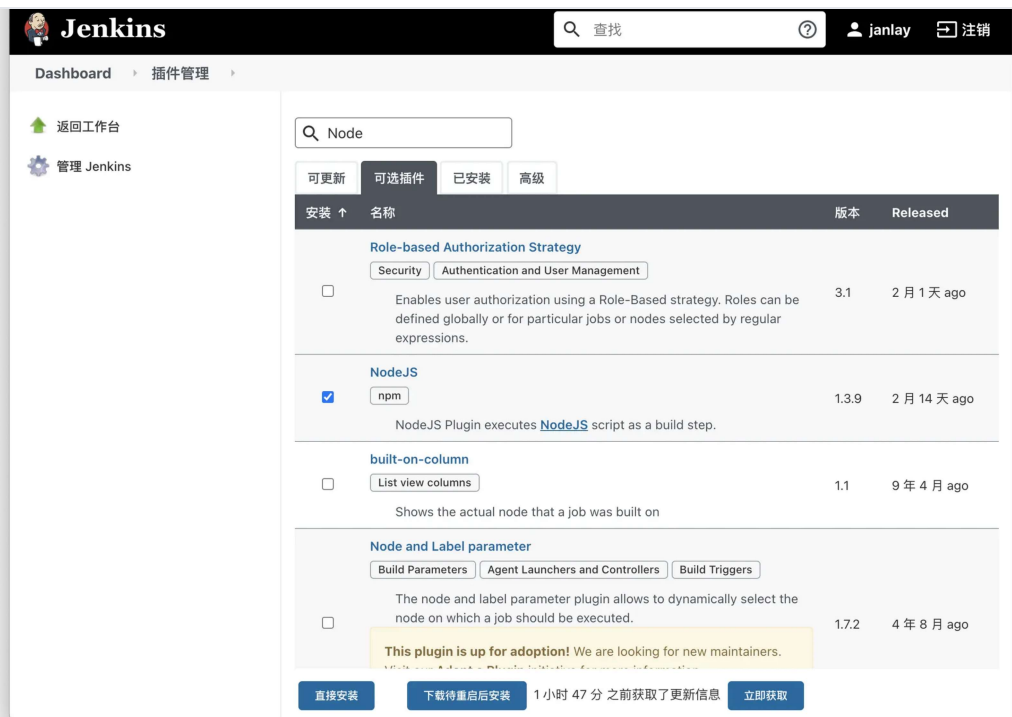
1. 安装 Nodejs 环境

在上一章，我们其实并没有在服务端安装 `Node` 环境。如果想要安装 `Node` 环境，有以下两个办法：

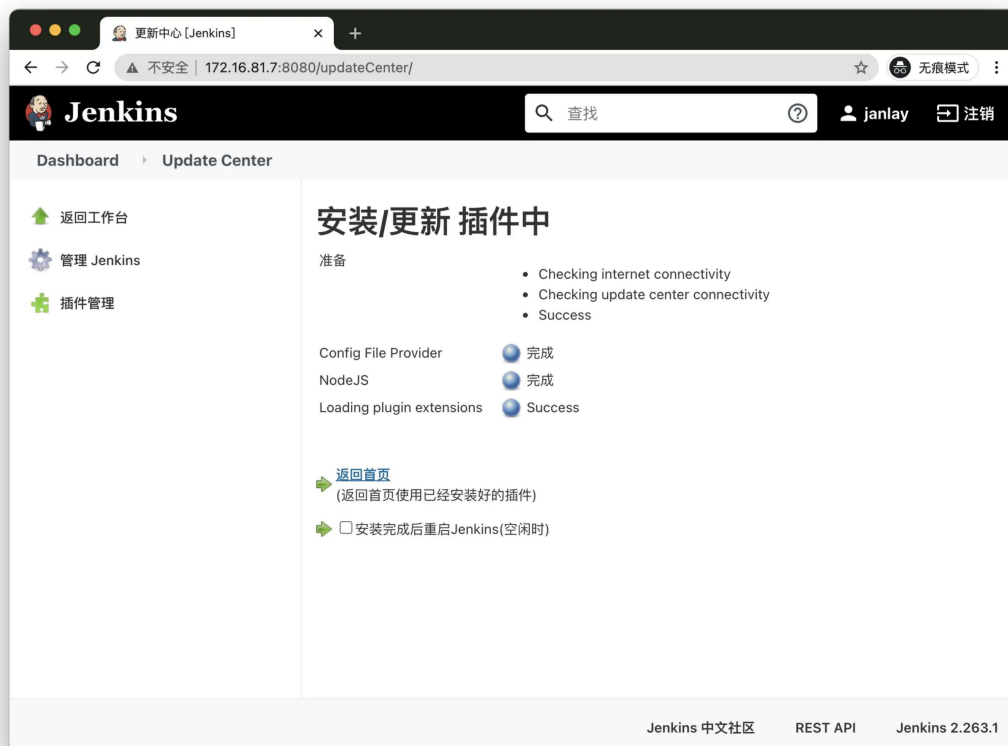
- 源码编译：这种方式是将 `Node` 源码拉下来后，在服务器端编译完成后才可以使用。时间比较长，流程也略复杂
- 使用 `Jenkins Plugin` 中 `NodeJS` 插件自动配置安装

在这里，我们可以选择第二种方式来安装，既方便又省力。

我们打开 `Jenkins` 首页，找到左侧的**系统配置 => 插件管理 => 可选插件**，搜索 `Node`。选中 `NodeJS` 后，点击左下角的 **直接安装** 开始安装插件



@稀土掘金技术社区



@稀土掘金技术社区

等待安装完毕后，返回 **Jenkins** 首页。找到 **Global Tool Configuration => NodeJS => 新增NodeJS**

JS

figuration

System

Global settings and paths.



Global Tool Configuration

Configure tools, their locations and automatic installers.

Global Tool Configuration

Plugins

disable or enable plugins that
the functionality of Jenkins.



Manage Nodes and Clouds

Add, remove, control and monitor the
various nodes that Jenkins runs jobs on.

Files

xml for maven, central
scripts, custom files, ...

@稀土掘金技术社区

找到下面的 **NodeJS** , 点击 **NodeJS** 安装, 选择相应的版本填写信息保存即可。

NodeJS

NodeJS 安装

新增 NodeJS



NodeJS

别名

node15.3

☒ Install automatically



Install from nodejs.org

版本

NodeJS 15.3.0

Force 32bit architecture



For the underlying architecture, if available, force the installation of the 32bit package. Otherwise the build will fail

Global npm packages to install

Specify list of packages to install globally -- see npm install -g. Note that you can fix the packages version by using the syntax 'packageName@version'

Global npm packages refresh hours

72

Duration, in hours, before 2 npm cache update. Note that 0 will always update npm cache

删除安装

新增安装

删除 NodeJS

@稀土掘金技术社区

如何使用

☒ Inspect build log for published Gradle build scans

☒ Provide Node & npm bin/ folder to PATH

 NodeJS Installation

 Specify needed nodejs installation where npm installed packages will be provided to the PATH

 npmrc file

 Cache location

☐ With Ant

@稀土掘金技术社区

第一次执行会下载对应的 **Node** 版本，后续不会下载。

2. 使用 SSH 协议集成 Git 仓库源

这一步，我们使用 **Jenkins** 集成外部 Git 仓库，实现对真实代码的拉取和构建。在这里，我们选用 **Gitee** 作为我们的代码源（**没有打广告，单纯觉得 Gitee 教学方便**）。这里准备一个 **vue-cli** 项目来演示构建。

2.1 生成公钥私钥

首先，我们先来配置公钥和私钥。这是 **Jenkins** 访问 **Git** 私有库的常用认证方式。我们可以使用 **ssh-keygen** 命令即可生成公钥私钥。在本地机器执行生成即可。这里的邮箱可以换成你自己的邮箱：

▼

shell 复制代码

```
1 ssh-keygen -t rsa -C "janlay884181317@gmail.com"
```

执行后，会遇到第一步步骤： **Enter file in which to save the key** 。

这一步是询问你要将公钥私钥文件放在哪里。默认是放在 **~/.ssh/id_rsa** 下，当然也可以选择输入你自己的路径。一路回车即可。

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in gitee-demo.
Your public key has been saved in gitee-demo.pub.
The key fingerprint is:
SHA256:gRby/5tSjYy+wrCR4U2bUSGsKCSJ4PAq4b2XnGHxLU0 janlay884181317@gmail.com
The key's randomart image is:
+---[RSA 3072]-----+
|=.  ..o .. E      |
|*o   o.+ . +      |
|+... o+..o .      |
|ooo o.=...        |
|oo o B =So o      |
|.   O B ..+ .      |
|   . X . ..       |
|   o o o o        |
|       ..oo       |
+-----[SHA256]-----+
```

@稀土掘金技术社区

结束后，你会得到两个文件。分别是 xxx 和 xxx.pub。

其中，xxx 是私钥文件，xxx.pub 是对应的公钥文件。我们需要在 **Git** 端配置公钥，在 **Jenkins** 端使用私钥与 **Git** 进行身份校验。

2.2 在 Gitee 配置公钥

在 **Gitee** 中，如果你要配置公钥有2种方式：仓库公钥 和 个人公钥。其中，如果配置了仓库公钥，则该公钥只能对配置的仓库进行访问。如果配置了个人公钥，则账号下所有私有仓库都可以访问。



SSH公钥创建

@稀土掘金技术社区

安全设置

SSH公钥

@稀土掘金技术社区 在下方有个添加公钥，

填入信息即可。

其中的标题为公钥标题，这里可以自定义标题；公钥则为刚才生成的 xxx.pub 文件。使用 `cat` 命令查看下文件内容，将内容填入输入框并保存。接着去 [Jenkins](#) 端配置私钥

shell 复制代码

```
1 cat xxx.pub
```



公钥

把你的公钥粘贴到这里，查看 [怎样生成公钥](#)

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGC/x3sOBvdps4UpSOWdc18uWQvki leDLIVPr2QP1t7aZ1oO/a2P4zE2CI0n7RquY8h3vbKpGARb  
82TDjXA  
N4K4k/Vu  
hjpuk  
janlay8  
84181317@gmail.com
```

确定

@福主掘金技术社区

2.3 在 Jenkins 配置私钥

回到 **Jenkins**。在 **Jenkins** 中，**私钥/密码** 等认证信息都是以 **凭证** 的方式管理的，所以可以做到全局都通用。我们可以在配置任务时，来添加一个自己的凭证。点击项目的 **配置**，依次找到 ****源码管理 => Git => Repositories ****

All +

S

W

Name ↓

上次成功

test

31 分 - #5

图标: 小 中 大

修改记录

工作空间

Build Now

配置

删除 Project

重命名

Atom

@稀土掘金技术社区

Repositories

Repository URL

git@gitee.com:Janlaywss/cicd-book.git

?

Credentials

janlay884181317@gmail.com (je

添加

?

高级...

Add Repository

@稀土掘金技术社区

这里的 **Repository URL** 则是我们的仓库地址，**SSH** 地址格式为 `git@gitee.com:xxx/xxx.git`。可以从仓库首页中的 克隆/下载 => SSH 中看到



重点是 **Credentials** 这一项，这里则是我们选择认证凭证的地方。我们可以点击右侧 **添加 => Jenkins** 按钮添加一条新的凭证认证信息。

点击后会打开一个弹窗，这是 **Jenkins** 添加凭证的弹窗。选择类型中的 **SSH Username with private key** 这一项。接着填写信息即可：

- ID：这条认证凭证在 **Jenkins** 中的名称是什么
- 描述：描述信息
- Username：用户名（邮箱）
- Private Key：这里则是我们填写私钥的地方。点击 **Add** 按钮，将 **xxx 私钥文件内所有文件内容全部复制过去（包含开头的 BEGIN OPENSSSH PRIVATE KEY 和结尾的 END OPENSSSH PRIVATE KEY）**

接着点击添加按钮，保存凭证。

保存后，在 **Credentials** 下拉列表中选择你添加的凭证。

如果没有出现红色无权限提示，则代表身份校验成功，可以正常访问。

3. 构建镜像

在我们将环境准备就绪后，就可以开始构建镜像了。不过，我们需要先准备个 **DockerFile** 才可以构建镜像。那什么是 **DockerFile** 呢？

3.1 编写 Dockerfile

什么是 Dockerfile

例如下面的步骤，使用 `Dockerfile` 可描述为：

1. 基于 `nginx:1.15` 镜像做底座。
2. 拷贝本地 `html` 文件夹内的文件，到镜像内 `/etc/nginx/html` 文件夹。
3. 拷贝本地 `conf` 文件夹内的文件，到镜像内 `/etc/nginx/` 文件夹。

[dockerfile](#) [复制代码](#)

```
1 FROM nginx:1.15-alpine
2 COPY html /etc/nginx/html
3 COPY conf /etc/nginx/
4 WORKDIR /etc/nginx/html
```

编写完成后，怎么生成镜像呢？我们只需要使用 `docker build` 命令就可以构建一个镜像了：

[dockerfile](#) [复制代码](#)

```
1 docker build -t imagename:version .
```

-t: 声明要打一个镜像的Tag标签，紧跟着的后面就是标签。标签格式为 镜像名:版本。
.: 声明要寻找dockerfile文件的路径，. 代表当前路径下寻找。默认文件名为 Dockerfile。关于更多 DockerFile 的语法，详细可以看[这里](http://www.runoob.com/docker/docker...)

因为我们的镜像只包含一个 `nginx`，所以 `dockerfile` 内容比较简单。我们只需要在代码根目录下新建一个名为 `Dockerfile` 的文件，输入以下内容，并将其提交到代码库即可。

[shell](#) [复制代码](#)

```
1 vi Dockerfile
```

[dockerfile](#) [复制代码](#)

```
1 FROM nginx:1.15-alpine
2 COPY html /etc/nginx/html
3 COPY conf /etc/nginx/
4 WORKDIR /etc/nginx/html
```

```
3 git push
```

3.2 Jenkins 端配置

在代码源和 `DockerFile` 准备就绪后，我们只需在 `Jenkins` 端配置下要执行的 `Shell` 脚本即可。找到项目的配置，依次找到**构建 => Execute shell**。输入以下脚本：

shell 复制代码

```
1 #!/bin/sh -l
2
3 npm install --registry=https://registry.npm.taobao.org
4 npm run build
5 docker build -t jenkins-test .
```

这里脚本很简单，主要是**安装依赖 => 构建文件 => 构建镜像**。填写完毕后保存

4. 执行任务

保存后我们去手动触发执行下任务。当未抛出错误时，代表任务执行成功

```
- Building for production...
DONE Compiled successfully in 2086ms下午9:17:05

File                                Size                                Gzipped

dist/js/chunk-vendors.de7b539e.js    130.25 KiB                         45.02 KiB
dist/js/app.19f1b41d.js              6.04 KiB                           2.27 KiB
dist/js/about.e17925de.js            0.44 KiB                           0.31 KiB
dist/css/app.877b7338.css            0.42 KiB                           0.26 KiB

Images and other types of assets omitted.

DONE Build complete. The dist directory is ready to be deployed.
INFO Check out deployment instructions at https://cli.vuejs.org/guide/deployment.html

Sending build context to Docker daemon 144.7MB

Step 1/3 : FROM nginx:1.15-alpine
--> dd025cdf837
Step 2/3 : COPY dist /etc/nginx/html
--> 6c5ef6780e87
Step 3/3 : WORKDIR /etc/nginx/html
--> Running in 083e599678d2
Removing intermediate container 083e599678d2
--> 6b4c36df4f66
Successfully built 6b4c36df4f66
Successfully tagged jenkins-test:latest
Finished: SUCCESS
```

@稀土掘金技术社区



在这一章，我们学会了使用 **Jenkins** 构建了自己的前端镜像。下一章我们将安装自己的私有镜像库，将镜像上传在镜像库内。

[< 上一章](#)[下一章 >](#)

留言

输入评论 (Enter换行, Ctrl + Enter发送)

发表评论

全部评论 (60)



何以渡清欢 **LV.3** **JY.4** 前端 @ 前端充电宝集团 6月前

感觉最好不要将Node版本写死吧，不同的项目可能会依赖不同的Node版本

👍 点赞 🗨 回复 ...



Asuka14024 **LV.1** **JY.5** 前端工程师 6月前

整个人都麻透了。。。用户切成Jenkins重新生成密钥都不管用



👍 1 🗨 回复 ...



fatsnake **JY.4** 10月前

作者你好，cicd-book.git 项目在哪里？

👍 点赞 🗨 回复 ...



一份光 **JY.5**  嘿嘿嘿 10月前

配置git遇到的坑，如果生成ssh密钥的是用了 -C "xx@xx.com" credetials 里username也要填这个邮箱。在命令执行ssh -T git@gitee.com，成功后将 know_hosts,id_rsa,id_rsa.pub 都移动到/var/lib/jenkins/.ssh/ 才没报错。。。折腾了十几分钟😓

👍 5 🗨 2 ...



缘缘 7月前

这个是什么原因呀，必须要移动到/var/lib/jenkins/.ssh/里，求解答

👍 点赞 🗨 回复 ...