

项目打包部署

1 修改配置

打包之前需要检查环境配置是否正确

| | |
|------------------|-----------------|
| node_modules | 2022/9/21 16:45 |
| public | 2022/9/19 21:10 |
| src | 2022/9/21 17:03 |
| .env | 2022/9/20 14:59 |
| .env.development | 2022/9/19 21:44 |
| .env.production | 2022/9/19 21:44 |
| .eslintrc.cjs | 2022/9/20 14:15 |
| .gitignore | 2022/9/19 21:45 |

主要检查production中配置是否正确

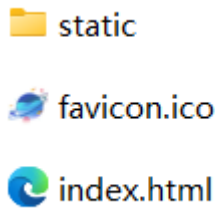
2 测试打包

执行npm run build指令执行打包

```
\project-frontend>npm run build
> project-frontend@1.0.0 build
> vite build

vite v3.1.3 building for production...
✓ 1489 modules transformed.
dist/index.html                                0.36 KiB
dist/static/404.f00ed3a.js                      0.44 KiB / gzip: 0.32 KiB
dist/static/403.a560e2d8.js                     0.41 KiB / gzip: 0.30 KiB
dist/static/500.8a9aae7c.js                     0.40 KiB / gzip: 0.29 KiB
dist/static/Dashboard.f388f27a.js              0.39 KiB / gzip: 0.27 KiB
dist/static/403.a8a838f1.css                   0.18 KiB / gzip: 0.13 KiB
dist/static/index.65ad6221.css                 4.67 KiB / gzip: 1.27 KiB
dist/static/base.f3449418.js                   14.02 KiB / gzip: 4.83 KiB
dist/static/Dashboard.0f594850.css             0.18 KiB / gzip: 0.13 KiB
dist/static/404.7dcd7cc5.css                   0.18 KiB / gzip: 0.13 KiB
dist/static/500.68cfc793.css                   0.18 KiB / gzip: 0.13 KiB
dist/static/HomeView.65fe56d8.css              15.43 KiB / gzip: 2.81 KiB
dist/static/base.3111e043.css                  7.99 KiB / gzip: 1.93 KiB
dist/static/LoginView.c645ac72.css             32.81 KiB / gzip: 4.13 KiB
dist/static/LoginView.76597553.js             52.12 KiB / gzip: 17.85 KiB
dist/static/HomeView.eea91fbl.js               66.44 KiB / gzip: 22.59 KiB
dist/static/index.a3fbe564.js                  178.16 KiB / gzip: 66.81 KiB
```

查看打包文件（打包后的文件生成在dist目录下面），示例效果如下图所示



如果没有问题测试打包正常，如果有编译错误需要自行修正源代码。

3 集成到Jenkins中

注意：

首先由于vue项目需要集成到nginx中，所以需要你提前安装好nginx服务器，这里就不演示nginx服务器搭建了。

其次vue前端应用的打包需要依赖NodeJS插件，所以我们需要安装并配置该插件，然后创建任务来打包部署。

3.1 安装NodeJS插件

在系统设置→插件管理中选择安装插件

Dashboard > 系统管理 > 插件管理

Updates

Available plugins 1

Installed plugins

Advanced settings

Plugins

NodeJS | NodeJS搜索插件

| Install | Name ↓ | Release |
|--------------------------|--|---------|
| <input type="checkbox"/> | NodeJS 1.5.1 npm NodeJS Plugin executes NodeJS script as a build step. | 8 月 1 |

Install without restart

安装

Download now and install after restart

3 分 21 秒 之前获取了更新信息

立即获取

进入安装页面

Updates

Available plugins

Installed plugins

Advanced settings

Download progress

Download progress

准备

- Checking internet connectivity
- Checking update center connectivity
- Success

Loading plugin extensions ... Pending

静静的等待安装完成

→ [返回首页](#)
(返回首页使用已经安装好的插件)

→ ☐ 安装完成后重启Jenkins(空闲时)

安装完成后


- ↓ Updates
- 📦 Available plugins
- ⚙️ Installed plugins
- ⚙️ Advanced settings

☰ Download progress

Download progress

准备

- Checking internet connectivity
- Checking update center connectivity
- Success

Loading plugin extensions  Success

→ [返回首页](#)  返回首页

(返回首页使用已经安装好的插件)

→ ☐ 安装完成后重启Jenkins(空闲时)

3.2 配置NodeJS插件

在系统设置→全局工具配置中进行插件配置

NodeJS

NodeJS 安装

系统下NodeJS 安装列表

新增 NodeJS

NodeJS

别名

Node16.17.0

输入别名

☒ 自动安装

Install from nodejs.org

版本

NodeJS 16.17.0

选择对应的NodeJS版本

For the underlying architecture, if available, force the installation of the 32-bit architecture.

Force 32bit architecture

保存

保存配置
应用

注意：如果全局配置中找不到NodeJS配置，那么重启一下服务器即可。

3.3 集成操作流程

3.3.1 创建任务

回到主面板，选择创建任务



Jenkins

Dashboard >

+ 新建任务

点击创建任务

👤 用户列表

📅 构建历史

⚙️ 系统管理

🖼️ 我的视图

🌊 打开 Blue Ocean

输入任务信息

输入一个任务名称

frontend

1 输入任务名称

» 必填项



构建一个自由风格的软件项目

这是Jenkins的主要功能。Jenkins将会结合任何SCM和任何构建系统来构建你的项目，甚至可以构建软件以外的系统。

2 选择它



流水线

精心地组织一个可以长期运行在多个节点上的任务。适用于构建流水线（更加正式地应当称为工作流），增加或者组织难以采用自由风格的任务类型。



构建一个多配置项目

适用于多配置项目，例如多环境测试、平台指定构建等等。



{0} 文件夹

Creates a set of multibranch project subfolders by scanning for repositories.



多分支流水线

根据一个SCM仓库中检测到的分支创建一系列流水线。



文件夹

创建一个可以嵌套存储的容器。利用它可以进行分组。视图仅仅是一个过滤器，而文件夹则是一个独立的命名空间，因此你可以有多个相同名称的内容，只要它们在不同的文件夹里即可。

如果你想根据一个已经存在的任务创建，可以使用这个选项



复制

输入自动完成

确定

3 确定创建

3.3.2 配置源码管理

任务描述自己按照情况书写即可，在源码管理中添加你的Github仓库地址。

提示：如果 GitHub 下载太缓慢，可以考虑将 GitHub 仓库导入到 Gitee 上面使用 Gitee 仓库来部署

Configuration

General

源码管理

构建触发器

构建环境

Build Steps

构建后操作

源码管理

☐ 无

☒ Git ?

Repositories ?

Repository URL ?

git@github.com:

2 输入你的GitHubSSH地址

Credentials ?

anyuser (github ssh private key)

3 选择SSH凭证

SSH凭证在安装Jenkins教程里面已经添加
如果你们没有添加请点击添加按钮自行添加

+ 添加

高级...

Add Repository

Branches to build ?

指定分支 (为空时代表any) ?

*/master

4 在这里可以选择你的分支

Add Branch

源码库浏览器 ?

(自动)

Additional Behaviours

新增 ▾

保存

应用

5 点击应用保存设置

3.3.3 配置node构建

在构建环境中把我们的node环境添加进去：

构建环境

- ☐ Delete workspace before build starts
- ☐ Use secret text(s) or file(s) ?
- ☐ Provide Configuration files ?
- ☐ Send files or execute commands over SSH before the build starts ?
- ☐ Send files or execute commands over SSH after the build runs ?
- ☒ Add timestamps to the Console Output 1
- ☐ Inspect build log for published Gradle build scans
- ☒ Provide Node & npm bin/ folder to PATH 2

NodeJS Installation

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

Node16.17.0 3

选择NodeJS版本

保存

应用 4

点击应用保存设置

添加一个执行shell的构建，用于将我们的前端代码进行编译打包：

Build Steps

增加构建步骤 1

Filter

Execute NodeJS script
Invoke Ant
Invoke Gradle script
Provide Configuration files
Run with timeout
Send files or execute commands over SSH
Set build status to "pending" on GitHub commit
执行 Windows 批处理命令
执行 shell 2
调用顶层 Maven 目标

填写执行脚本

Build Steps

≡ 执行 shell ?

命令

查看 可用的环境变量列表

cd project-frontend

npm -v

SASS_BINARY_SITE=https://npm.taobao.org/mirrors/node-sass/ npm install node-sass

npm config set registry https://registry.npm.taobao.org

npm install

高级...

增加构建步骤 ▾

构建后操作

增加构建后操作步骤 ▾

保存

应用 2

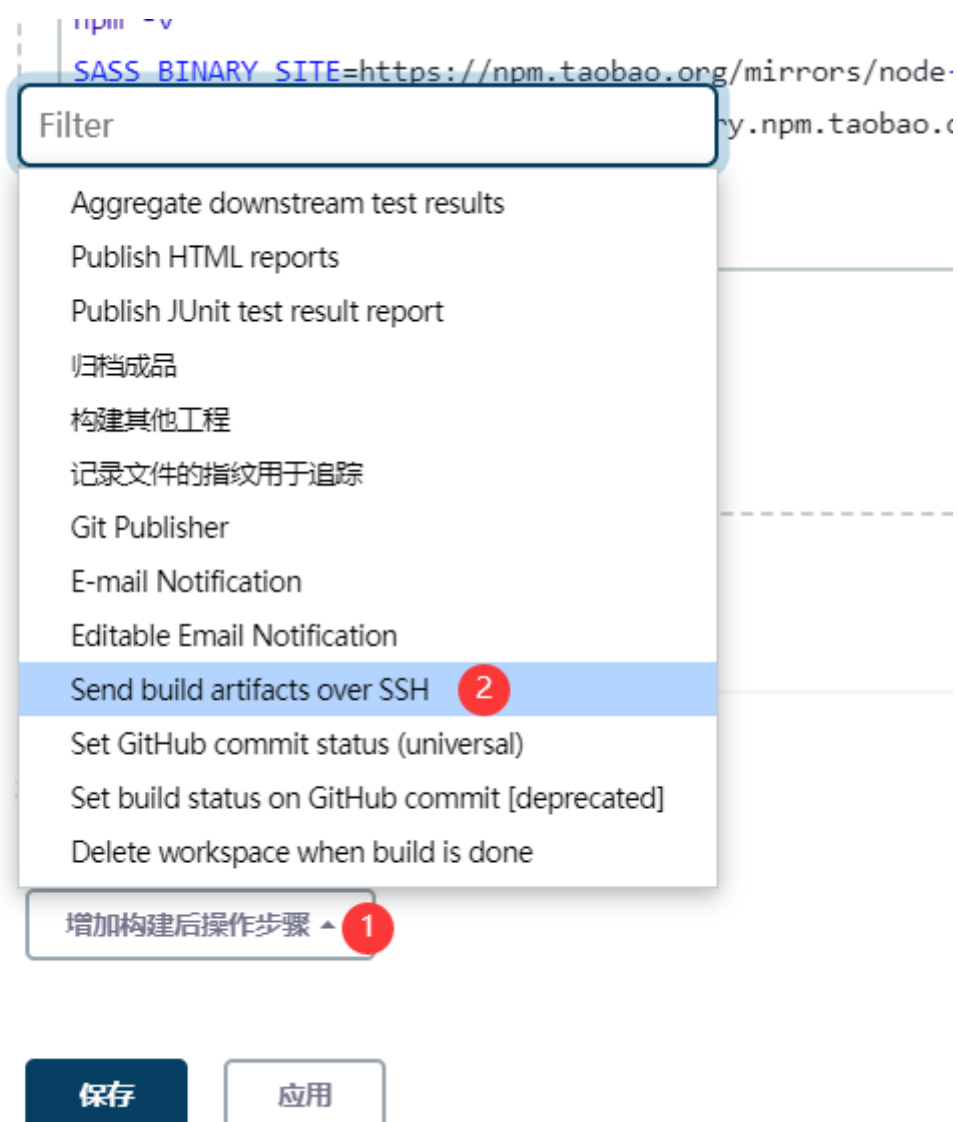
点击应用保存设置

下面是脚本描述

```
1 # 由于我们的项目没有在根目录所以这里需要进入到对应目录执行构建
2 cd project-frontend
3 # 查看版本信息
4 npm -v
5 # 解决存放在Github上的sass无法下载的问题
6 SASS_BINARY_SITE=https://npmmirror.com/mirrors/node-sass/ npm install node-sass
7 # 将镜像源替换为淘宝的加速访问
8 npm config set registry https://registry.npmmirror.com
9 # 安装项目依赖
10 npm install
11 # 项目打包
12 npm run build
```

3.3.4 配置远程启动

添加一个使用ssh执行远程脚本的构建，用于将我们打包后的代码发布到nginx中去：



填写参数内容如下图所示：

构建后操作

Send build artifacts over SSH ?

SSH Publishers

SSH Server

Name ?

192.168.220.128

1

选择要部署到的远程服务器

高级...

Transfers

Transfer Set

Source files ?

project-frontend/start.sh,project-frontend/dist/**

2

传输启动脚本和编译文件

Remove prefix ?

Remote directory ?

/home/app

3

传输到远程服务器的根目录

Exec command ?

```
cd /home/app/project-frontend
chmod +x ./start.sh
sh ./start.sh
```

4

输入执行脚本

保存

应用

5

点击应用保存设置

传入文件内容说明：

```
1 # 远程执行服务脚本
2 project-frontend/start.sh,
3 # 源码打包编译后的文件
4 project-frontend/dist/**
```

执行命令如下：

```
1 # 首先进入脚本所在位置，这里与你填写的Remote directory对应
2 cd /home/app/project-frontend
3 # 给脚本执行权限
4 chmod +x ./start.sh
5 # 执行脚本
6 sh ./start.sh
```

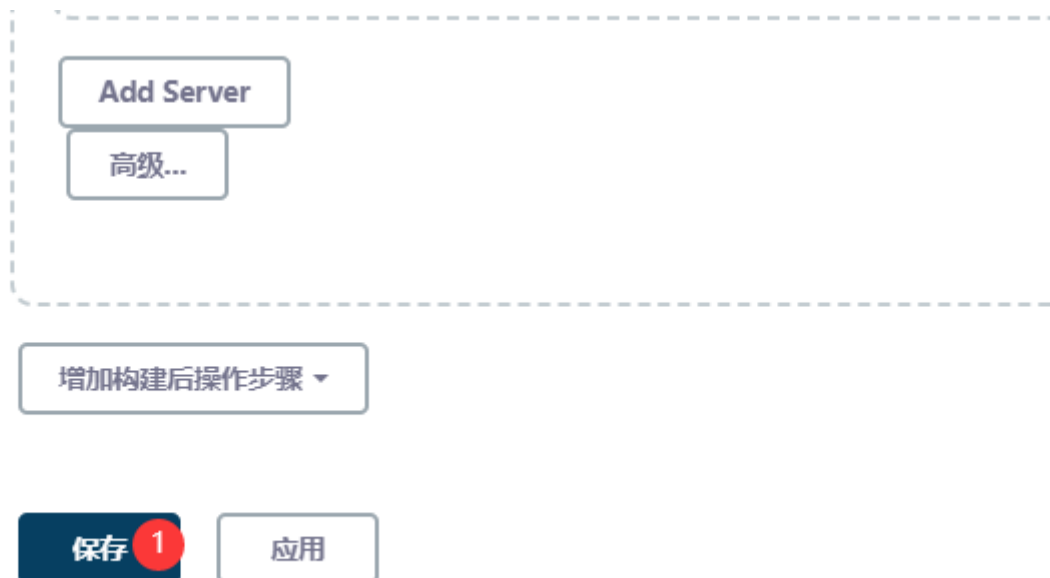
下面是start.sh脚本中内容

注意：启动脚本中的路径根据你需要你自己目录情况和容器名称做修正

```
1 #!/bin/bash
2 # /home/nginx/html 修改成你Nginx中静态文件的目录
3 nginx_dir='/home/nginx/html'
4 # nginx 修改成你的容器的名称
5 docker stop nginx
6 echo '——stop nginx——'
7 rm -rf ${nginx_dir}
8 echo '——rm html dir——'
9 mv dist ${nginx_dir}
10 echo '——mv dist dir to html dir——'
11 # nginx 修改成你的容器的名称
12 docker start nginx
13 echo '——start nginx——'
```

3.3.5 保存配置

所有操作执行完成后保存配置



3.3.6 执行任务

打开Blue Ocean

Dashboard > frontend >

☰ 状态

</> 修改记录

📁 工作空间

▶ 立即构建

⚙️ 配置

🗑️ 删除 工程

★ 收藏夹

🌊 打开 Blue Ocean

✎ 重命名

工程 frontend

项目前端部署任务

相关链接

⚙️ Build History

构建历史 ^

📡 Atom feed 全部

📡 Atom feed 失败

运行任务

frontend ☆ ⚙️

活动 分支 Pull Requests

🔄 运行

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| ● | — | — | — | — | — | — | ● |
| ● | — | — | — | — | — | — | ● |
| ● | — | — | — | — | — | — | ● |
| ● | — | — | — | — | — | — | ● |


○—○—○

该任务还未运行

运行

3.3.7 查看日志


点击要查看的任务，可以看到任务执行列表

 frontend ☆ ⚙️


活动 | 分支 | Pull Requests

▶ 运行

⏏ Disable

| 状态 | 运行 | 提交 | 消息 | 持续时间 | 完成 |
|---|----|----|--------------|------|----|
|  | 1 | - | 由用户 admin 启动 | 9s | - |

下面是执行日志

 frontend 1

分支: -

36s

没有修改

提交: -

-

由用户 admin 启动

日志

```
1 Started by user admin
2 Running as SYSTEM
3 Building in workspace /var/jenkins_home/workspace/frontend
4 The recommended git tool is: NONE
5 using credential 7ed373b4-1a59-45ef-8562-7e697785e29f
6 Cloning the remote Git repository
7 Cloning repository git@github.com:
8 > git init /var/jenkins_home/workspace/frontend # timeout=10
9 Fetching upstream changes from git@github.com:
10 > git --version # timeout=10
11 > git --version # 'git version 2.36.2'
12 using GIT_SSH to set credentials github ssh private key
13 Verifying host key using known hosts file, will automatically accept unseen keys
14 > git fetch --tags --force --progress -- git@github.com:
15 > git config remote.origin.url git@github.com:zero-awe1/
16 > git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
17 Avoid second fetch
18 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
19 Checking out Revision 83ed6edce4714157ec0827837d7fd2c36e65c301 (refs/remotes/origin/master)
20 > git config core.sparsecheckout # timeout=10
21 > git checkout -f 83ed6edce4714157ec0827837d7fd2c36e65c301 # timeout=10
22 Commit message: "集成验证码插件"
23 First time build. Skipping changelog.
24 [frontend] $ /bin/sh -xe /tmp/jenkins18315726194952163784.sh
25 + cd project-frontend
26 + npm -v
27 8.15.0
28 + SASS_BINARY_SITE=https://npm.taobao.org/mirrors/node-sass/ npm install node-sass
```

运行成功日志

```

42
43 [log] vite v3.2.5 building for production...
44 [log] transforming...
45 [log] ✓ 1609 modules transformed.
46 [log] rendering chunks...
47 [log] dist/index.html 0.36 KiB
48 [log] dist/static/404.655a6dff.js 0.44 KiB / gzip: 0.32 KiB
49 [log] dist/static/403.f39cff44.js 0.41 KiB / gzip: 0.30 KiB
50 [log] dist/static/500.281ebc83.js 0.40 KiB / gzip: 0.29 KiB
51 [log] dist/static/Dashboard.0c63a20a.js 0.39 KiB / gzip: 0.28 KiB
52 [log] dist/static/500.68cfc793.css 0.18 KiB / gzip: 0.13 KiB
53 [log] dist/static/index.65ad6221.css 4.67 KiB / gzip: 1.27 KiB
54 [log] dist/static/404.7dcd7cc5.css 0.18 KiB / gzip: 0.13 KiB
55 [log] dist/static/403.a8a838f1.css 0.18 KiB / gzip: 0.13 KiB
56 [log] dist/static/Dashboard.1f2c226c.css 0.18 KiB / gzip: 0.13 KiB
57 [log] dist/static/HomeView.b4559efc.css 15.34 KiB / gzip: 2.79 KiB
58 [log] dist/static/base.3111e043.css 7.99 KiB / gzip: 1.93 KiB
59 [log] dist/static/base.46560159.js 14.46 KiB / gzip: 5.00 KiB
60 [log] dist/static/HomeView.486af18f.js 69.14 KiB / gzip: 23.23 KiB
61 [log] dist/static/LoginView.ed7e416d.css 84.59 KiB / gzip: 40.53 KiB
62 [log] dist/static/LoginView.b52d950d.js 121.26 KiB / gzip: 41.11 KiB
63 [log] dist/static/index.e754ddd4.js 183.20 KiB / gzip: 68.69 KiB
64 SSH: Connecting from host [f1dd093abaa5]
65 SSH: Connecting with configuration [192.168.220.128] ...
66 SSH: EXEC: completed after 1,016 ms
67 SSH: Disconnecting configuration [192.168.220.128] ...
68 SSH: Transferred 19 file(s)
69 Finished: SUCCESS

```

3.4 遇到问题

`env node not found` 遇到这个问题，jenkins一直无法打包。

首先进入你的Jenkins容器

```
1 docker exec -u 0 -it jenkins bash
```

然后安装nodejs和npm

```
1 apk add --no-cache nodejs npm
```

```

bash-5.1# apk add --no-cache nodejs npm
fetch https://dl-cdn.alpinelinux.org/alpine/v3.16/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.16/community/x86_64/APKINDEX.tar.gz
(1/7) Installing c-ares (1.18.1-r0)
(2/7) Installing libgcc (11.2.1_git20220219-r2)
(3/7) Installing icu-data-en (71.1-r2)
Executing icu-data-en-71.1-r2.post-install
*
* If you need ICU with non-English locales and legacy charset support, install
* package icu-data-full.
*
(4/7) Installing libstdc++ (11.2.1_git20220219-r2)
(5/7) Installing icu-libs (71.1-r2)
(6/7) Installing nodejs (16.16.0-r0)
(7/7) Installing npm (8.10.0-r0)
Executing busybox-1.35.0-r17.trigger
OK: 350 MiB in 96 packages

```

然后可以查看一下版本

```
bash-5.1# node -v  
v16.16.0  
bash-5.1# npm -v  
8.10.0  
bash-5.1#
```