> 技术要点：Node、Nvm

前言

通过前五章的学习，你已熟悉 云服务器/域名系统/站服务器 部署服务环境。掌握整体部署与工具配置，学会独立上线自己的网站。像 Nginx 这样重量级的配置工具，还得多多上机操作，当然后续还会有更多 Nginx 的应用场景。

搭建基础服务完毕可借助 Node 渗透到 前端工程化 各个环节中，再将发展链路延伸到整个全栈领域中。熟悉 Node 后可完成前端工作外的工作，可为自己负责的前端应用部署相关 前端工程化 基建，也可开发一些 CMD工具 提升工作效率。

因为 Node 在 前端工程化 中极具重要性，因此需好好部署，让其通畅无阻地使用起来。本章将带领你**部署多环境多版本Node**，将 Node 同时部署到 本地环境 与 服务器环境 ，通过 Node版本管理工具 无缝切换 Node版本 。

背景：如何清理干净Node

Node 诞生于 2009年 ，经过 13年 的沉淀，如今生态圈越来越庞大以及逐渐走向稳定。 Node 不是一门语言也不是一个框架，而是一个基于 Chrome V8 引擎的 JS运行时 。它可解析与运行 JS ，就像浏览器解析与运行 JS 一样。

Node 能在 本地环境 与 服务器环境 中运行，因此可基于 JS 开发更多应用，例如 服务应用 、 爬虫应用 、 同构应用 、 工具应用 、 脚手架应用 等。

学习强大的 Node ，可掌握如何开发更多应用、如何操作各种数据库、非阻塞 I/O 、事件驱动等重要概念。更何况有了 Node 的加持，相信学习与使用 前端工程化 武装自己会更得心应手，但有些问题必须得提前解决，不然在开发时很易出错甚至影响后续进度。本章将会基于部署 Node环境 展开并解决这些问题，让后续的 Node开发 更顺畅。

你肯定会遇到不同项目使用不同版本的 Node ，有些同学会在 本地环境 安装多个 Node版本 ，切换 Node版本 就手动修改环境变量中的 Node路径 。这种方式看似解决了多版本 Node环境 的问题，但实质上操作起来很不方便。

Nvm

鉴于这种情况， Node Version Manager 应运而生，它不是一个工具，而是一系列可切换 Node版本 的工具总称，简称 Nvm 。 Nvm 在 Windows系统 中可选nvm-windows，在 Mac系统 中可选nvm-sh。不要以为两个工具同源，其实它们无任何关系，但可执行的命令却一样。

完全卸载Node

在学习本课程前有些同学可能已在本地部署了 Node环境 ，若不卸载已有的 Node 可能会影响 Nvm 的使用，因此在部署 Nvm 前必须清理干净 Node 。

> Windows

在 Windows系统 中卸载 Node 很简单，参照以下操作。

- 选择 控制面板 → 程序 → 卸载 ，找到 Node.js ，点击卸载
- 打开 C:/Users/$USER/AppData/Roaming 目录，删除 npm 与 npm-cache 两个文件夹
- 打开 C:/Users/$USER 目录，删除 .npmrc 文件

> MacOS

在 MacOS系统 中卸载 Node 就很复杂了，因为很多相关文件都分布在不同位置，卸载起来很麻烦，很易遗漏。通过多次筛查，发现 Node 相关文件分布在 /Users/$USER 、 /usr/local 和 /var/root 的目录中，因此执行以下命令将其全部删除。

```
rm -rf /Users/$USER/.npm
rm -rf /usr/local/{bin/{node,npm},lib/node_modules/npm,lib/node,share/man/*/node.*}
rm -rf /usr/local/bin/node
rm -rf /usr/local/include/node
rm -rf /usr/local/lib/node_modules
rm -rf /usr/local/share/doc/node
rm /Users/$USER/.npmrc
rm /usr/local/lib/dtrace/node.d
rm /usr/local/share/man/man1/node.1
rm /usr/local/share/systemtap/tapset/node.stp
```

```
rm /var/db/receipts/org.nodejs.*
rm /var/root/.npm
rm /var/root/.npmrc
```

方案：部署多环境多版本Node

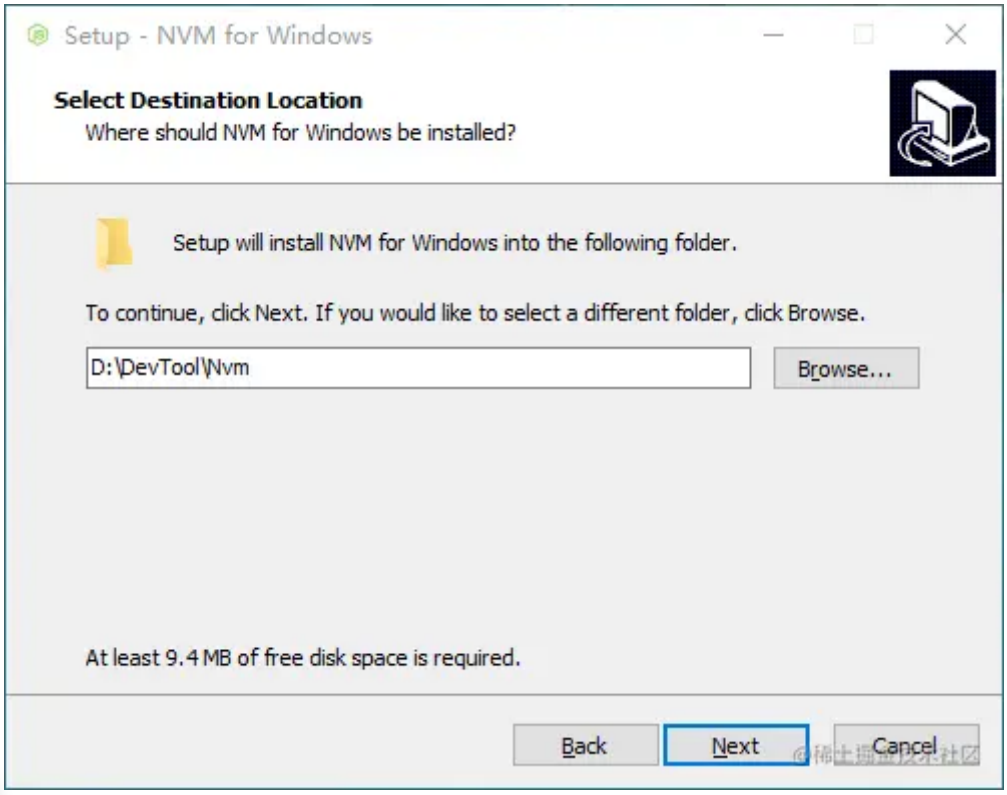
本地环境部署Node

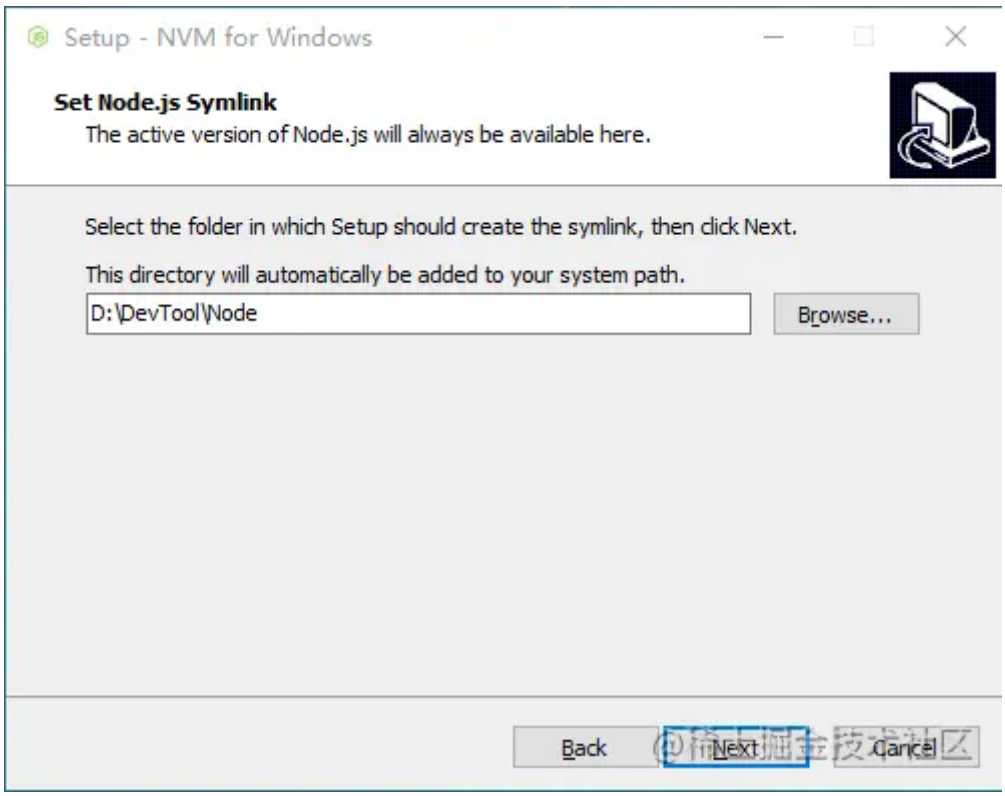
> Windows

打开 `nvm-windows` 的下载地址，下载最新版本的 `nvm-setup.zip` ，解压后双击 `nvm-setup.exe` 文件，开启傻瓜式安装。

当遇到 `Select Destination Location` 与 `Select Node.js Symlink` 两个步骤时，不要选择默认路径，修改其安装路径。 `Select Destination Location` 表示 `Nvm` 的安装路径， `Select Node.js Symlink` 表示 `Node` 的安装路径，以 `D盘` 为例，将两个路径改成以下路径。

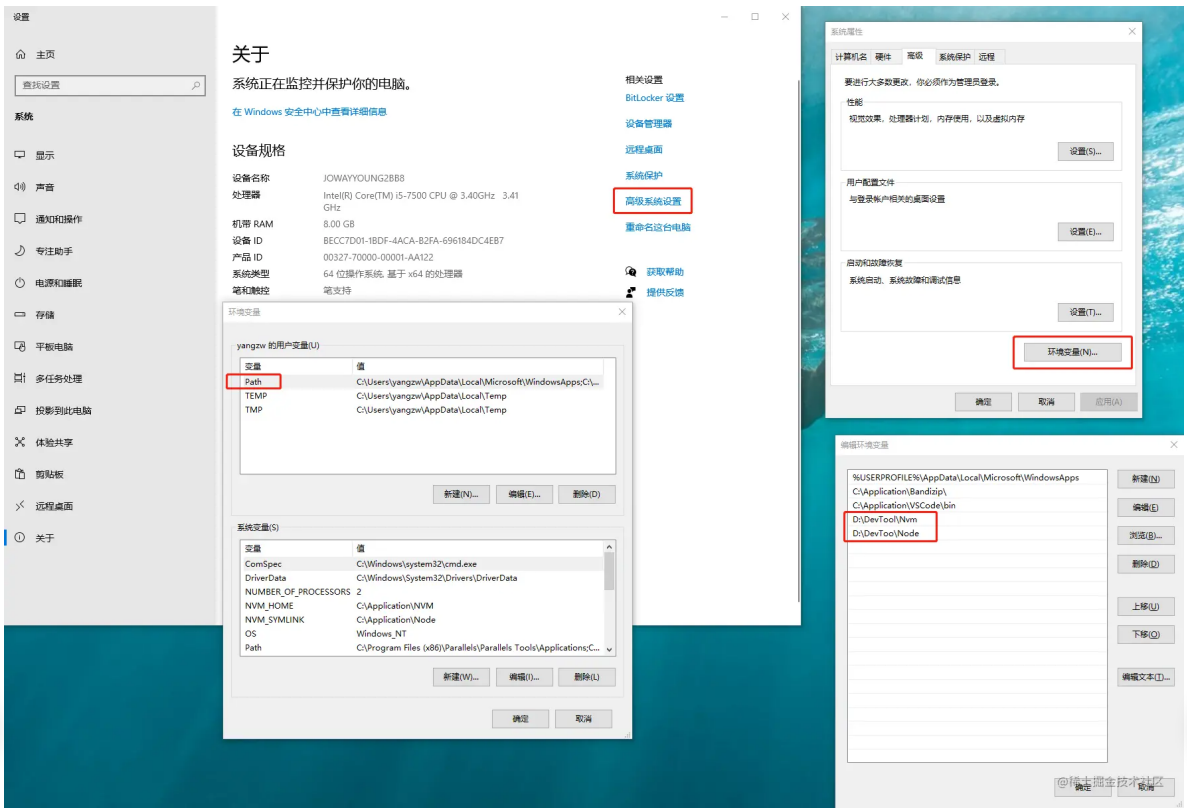
- **Nvm路径**： `D:/DevTool/Nvm`
- **Node路径**： `D:/DevTool/Node`





这两个路径的主要作用在于 D:/DevTool/Nvm 存放多个 Node版本 ，在切换 Node版本 时将指定的 Node 程序文件通过软链接的方式映射到 D:/DevTool/Node 。以 16.0.0 为例，打开 D:/DevTool/Node 就会跳转到 D:/DevTool/Nvm/v16.0.0 目录中，当切换为 17.0.0 ，打开 D:/DevTool/Node 就会跳转到 D:/DevTool/Nvm/v17.0.0 目录中。

安装完毕还需配置环境变量。选择 此电脑 → 属性 → 高级系统设置 → 环境变量 。选择 Path 再点击 编辑 ，把 Nvm路径 与 Node路径 加入到配置列表中。



打开 CMD工具 ， 执行 nvm v 查看版本，输出版本表示安装成功。只需掌握以下命令就能操作 nvm 。

命令	功能
nvm install <ver>	安装版本
nvm uninstall <ver>	卸载版本
nvm use <ver>	切换版本
nvm ls	查看版本列表
nvm ls available	查看可用版本列表
nvm install latest	安装最新版本

尝试安装一个 Node 版本，执行 nvm install 16.0.0 ， 等待几分钟可能还未安装下来。因为 nvm 默认使用国外的 Node镜像服务器 ， 所有 Node安装包 都托管在 https://nodejs.org/dist 中，所有 Npm安装包 都托管到 https://github.com/npm/cli/archive 中，安装过慢成了必然。

执行以下命令修改 Node镜像 与 Npm镜像 。

```
nvm node_mirror https://npm.taobao.org/mirrors/node/
nvm npm_mirror https://npm.taobao.org/mirrors/npm/
```

会在 D:/DevTool/Nvm 目录中生成一个配置文件 setting.txt ， 其内容就是通过上述命令生成的配置。

txt

```
root: D:\DevTool\Nvm
path: D:\DevTool\Node
node_mirror: https://npm.taobao.org/mirrors/node/
npm_mirror: https://npm.taobao.org/mirrors/npm/
```

重新安装多个 Node版本 并尝试切换不同版本。

```
PS C:\Document\Gitee\bruce> nvm install 16.14.0
Downloading node.js version 16.14.0 (64-bit)...
Extracting...
Complete


Installation complete. If you want to use this version, type

nvm use 16.14.0
PS C:\Document\Gitee\bruce> nvm install 14.19.0
Downloading node.js version 14.19.0 (64-bit)...
Complete
Creating C:\Application\NVM\temp

Downloading npm version 6.14.16... Complete
Installing npm v6.14.16...

Installation complete. If you want to use this version, type

nvm use 14.19.0
PS C:\Document\Gitee\bruce> nvm install 12.22.10
Downloading node.js version 12.22.10 (64-bit)...
Complete
Creating C:\Application\NVM\temp

Downloading npm version 6.14.16... Complete
Installing npm v6.14.16...

Installation complete. If you want to use this version, type

nvm use 12.22.10
PS C:\Document\Gitee\bruce> nvm ls

    16.14.0
    14.19.0
    12.22.10
PS C:\Document\Gitee\bruce>
```

@稀土掘金技术社区

> MacOS

打开 `nvm-sh` 的下载地址，确认最新版本。为了方便管理 `Nvm` 相关文件，创建 `/Users/$USER/Documents/记录/Nvm` 目录用于存放 `Nvm` 。若当前版本为 `0.39.1` ，执行以下命令安装 `nvm-sh` 。

```
git clone https://gitee.com/mirrors/nvm.git /Users/yangzw/Documents/记录/Nvm && cd /User
```

打开 CMD工具 ， 执行 vim ~/.bash_profile 配置环境变量，加入以下内容。主要目的是重置 Nvm路径 与 Node镜像 ， 开启 Nvm 自启动。

```
# Nvm
export NVM_DIR=/Users/yangzw/Documents/记录/Nvm
export NVM_NODEJS_ORG_MIRROR=https://npm.taobao.org/mirrors/node
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"
```

设置完毕执行 source ~/.bash_profile 使环境变量生效。再执行 nvm -v ， 输出版本表示安装成功。命令与上述 nvm-windows 一样。

服务器部署Node

打开 CMD工具 ， 登录服务器。执行 git --version ， 输出版本表示软件存在。若未安装 git ， 打开Git查看其最新版本。若当前版本为 2.36.1 ， 通过以下方式安装 git 。

打开 FTP工具 ， 在根目录中创建 tool 文件夹。

> 删除旧版git

```
yum remove git
```

> 安装依赖

```
yum install curl-devel expat-devel gettext-devel openssl-devel zlib-devel gcc perl-ExtUt
```

> 进入工具目录并创建版本目录

```
cd /tool && mkdir git-version
```

> 下载git

```
wget https://www.kernel.org/pub/software/scm/git/git-2.36.1.tar.gz
```

> 解压git

```
tar -zxvf git-2.36.1.tar.gz -C /tool/git-version
```

> 进入目录

```
cd /tool/git-version/git-2.36.1
```

> 编译git

```
make prefix=/tool/git all && make prefix=/tool/git install
```

> 生效环境变量

设置软链接，可通过 `git` 快速调用命令。

```
echo "export PATH=$PATH:/tool/git/bin" >> ~/.bash_profile
source ~/.bash_profile
```

执行 `git --version` ，输出版本表示安装成功。打开 `nvm-sh` 的下载地址，确认最新版本。

在 `tool` 文件夹中创建 `nvm` 文件夹，用于存放 `nvm` 相关文件。若当前版本为 `0.39.1` ，执行以下命令安装 `nvm-sh` 。

```
git clone https://gitee.com/mirrors/nvm.git /tool/nvm && cd /tool/nvm && git checkout `g
```

安装完毕执行 `vim ~/.bash_profile` 配置环境变量，加入以下内容。主要目的是重置 `Nvm路径` 与 `Node镜像` ，开启 `Nvm` 自启动。

```
# Nvm
export NVM_DIR=/tool/nvm
export NVM_NODEJS_ORG_MIRROR=https://npm.taobao.org/mirrors/node
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"
```

设置完毕执行 `source ~/.bash_profile` 使环境变量生效。执行 `nvm -v` ，输出版本表示安装成功。命令与上述 `nvm-windows` 一样。最后也能像 `本地环境` 那样愉快地使用 `nvm` 了。

版本：理清Node版本关系

可能有些同学会问，为何只选择 `16.14.0` 、 `14.19.0` 和 `12.22.10` 三个版本？那先从官网说起，打开Node官网，发现官网推荐安装的两个版本分别是 `Current最新尝鲜版` 与 `LTS长期维护版` 。

`Current` 与 `LTS` 并不是版本，而是同一个主版本的不同阶段。

Current

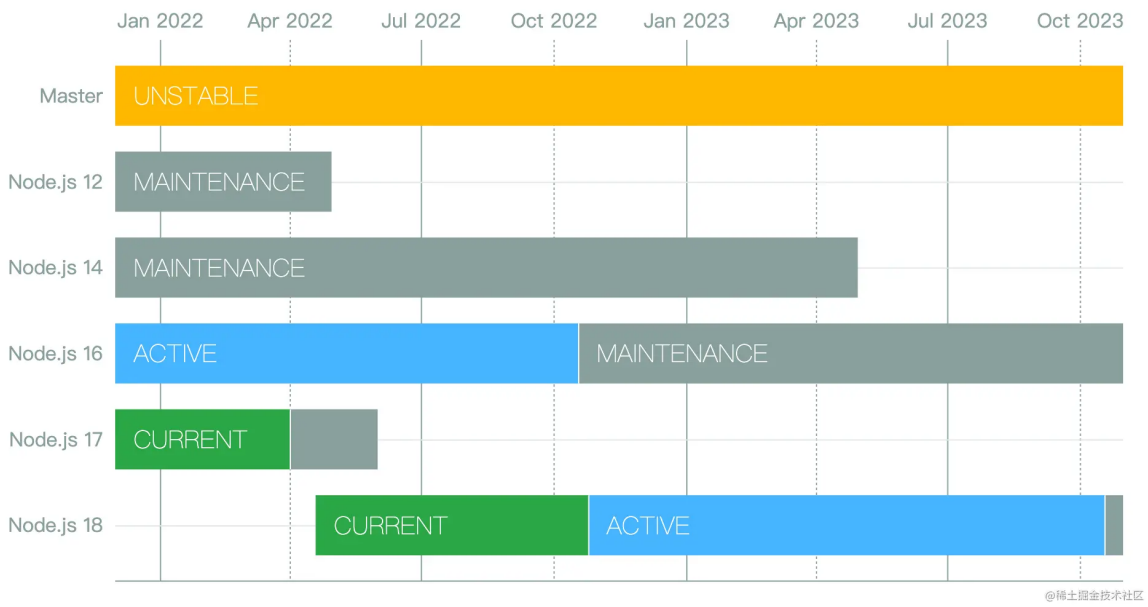
一个新的主版本发布后先进入 `Current` 阶段，该阶段持续6个月，目的是给各个 `Npm模块` 的作者花时间适配新版本。6个月后，奇数版本(例如 `11、13、15、17` 等)变成**不支持状态**，偶数版本(例如 `10、12、14、16` 等)变成**Active LTS**且准备投入使用。

```txt
每年的4月发布偶数版本，10月发布奇数版本。
```

LTS

`LTS` 阶段分为**Active LTS**与**Maintenance LTS**。 `Active LTS` 持续18个月，到期后进入 `Maintenance LTS` 再持续12个月，到期后进入 `EOL` ，最终退出历史舞台。

引用官网的概念图与上述版本定义就可解释我为何选用这三个版本了。



区别

如何选择 Node版本 ，那就看情况吧。

针对 测试环境 可选 Current 。 Current 会带来一些实验性的新特性，例如ES最新语法与 API ，因为持续时间最多6个月且兼容性与稳定性肯定不会有 LTS 那样具备保障性，因此引入 生产环境 需三思而后行。

针对 生产环境 可选 LTS 。 LTS 作为一个稳定的运行环境，会通过中小版本迭代更新功能，持续时间最多30个月，因此足够支撑一个项目完成很多生命周期内的任务。

LTS 也不是万能的，为了稳步开发，选择最新版本 LTS 以及往下两个主版本共存，这样才能保障近段时间内项目运行的稳定性。若当前 TLS 为 16.14.0 ，打开版本列表，查询 14.x 与 12.x 的最新版本。

最终确定的三个版本为 16.14.0 、 14.19.0 和 12.22.10 。

总结

多版本 Node环境 其实是一个很棘手的问题，因为涉及配置众多且区分操作系统，因此在部署时需特别注意。任何一个细节遗漏都有可能导致 Node 无法启动。另外在部署 Nvm 前必须将已有的 Node 卸载，这样才能确保整个系统环境在运行 Nvm 时不会受到已有 Node 的干扰。

本章内容到此为止，希望能对你有所启发，欢迎你把自己的学习心得打到评论区！

- 示例项目：fe-engineering
- 正式项目：bruce

留言

输入评论（Enter换行，Ctrl + Enter发送）

发表评论

全部评论（20）

SteveCGC JY.3 大哥你搞前端，前端有... 3月前

win10电脑，在应用市场装了ubuntu，然后按上面的步骤装了nvm，然后关掉终端后再打开，还是提示没有安装，是哪个环境出问题了呢

点赞 3

JowayYoung （作者） 3月前

这个没按小册部署的环境来做，我也很难处理呀，建议买一个云服务器，毕竟也不贵

点赞 回复

SteveCGC 回复 JowayYoung 3月前

我们公司的开发电脑都是windows的

“这个没按小册部署的环境来做，我也很难处理呀，建议买一个云服务器...”

点赞 回复

查看更多回复

HighClassLickDog LV.5 JY.4 ... 3月前

nvm导致husky找不到命名，请问这个怎么解决

点赞 2

HighClassLi... 3月前

找不到命令

点赞 回复

JowayYoung （作者） 3月前

node命令能正常运行吗？husky在node里兼容v17吗？先确认下这两个问题

点赞 回复

SteveCGC JY.3 大哥你搞前端，前端有... 4月前

windows下执行nvm use 会报错，咋解决啊

点赞 3

熊巴巴 4月前

以管理员权限打开控制台

点赞 回复

JowayYoung （作者） 4月前

电脑权限问题，使用最高权限就可以

点赞 回复

查看更多回复

帅气小野猪 JY.3 前端工程师 4月前

最近用上volta，如果能在公司里推开，能保证每个项目都约束相同的node版本，nvm还是要看个人意愿。

点赞 1

JowayYoung （作者） 4月前

主要是老旧项目交替的情况下，约束相同版本就有局限性了

点赞 回复