# 为什么node—sass总是安装失败?

Nov 13, 2019

node-sass是我们开发中很常见的依赖包，也是安装时间冗长和最常见到报错的依赖。 安装node-sass失败原因有很多种，我们在说失败原因之前，先来分析一下node-sass的安装过程(以下node版本为v10.15.3)：

```
PS D:\demo> npm i node-sass

// 从npm源安装到node_modules
> node-sass@4.13.0 install D:\demo\node_modules\node-sass
> node scripts/install.js

// 下载binding.node
Downloading binary from https://github.com/sass/node-sass/releases/download/v
Download complete .] - :
Binary saved to D:\demo\node_modules\node-sass\vendor\win32-x64-64\binding.no

// 缓存binding.node
Caching binary to C:\Users\leepi\AppData\Roaming\npm-cache\node-sass\4.13.0\w

> node-sass@4.13.0 postinstall D:\demo\node_modules\node-sass
> node scripts/build.js

Binary found at D:\demo\node_modules\node-sass\vendor\win32-x64-64\binding.no
Testing binary
Binary is fine

// 写package-lock.json
npm notice created a lockfile as package-lock.json. You should commit this fi
npm WARN demo@1.0.0 No description
npm WARN demo@1.0.0 No repository field.

+ node-sass@4.13.0
added 174 packages from 138 contributors and audited 529 packages in 24.379s
found 0 vulnerabilities
```

我们可以看到，安装node-sass有几个步骤：

1. 校验本地node_modules中是否已安装node-sass，版本是否一致;
2. 如未安装或版本不符，从npm源安装node-sass本体;
3. 检测全局缓存和本地中是否有 `binding.node` ,如有即跳过安装;
4. 没有 `binding.node` 则从github下载该二进制文件并将其缓存到全局;
5. 假如 `binding.node` 下载失败，则尝试本地编译出该文件;
6. 将版本信息写到 `package-lock.json` ;

由此看到，实际上node-sass依赖了一个二进制文件 `binding.node` ，从npm源安装完本体后还会从github下载 `binding.node` 。

因此安装node-sass相关的失败原因有以下几种：

## 原因一: npm源速度慢

由于众所周知的国内网络环境，从国内安装官方源的依赖包会很慢。可以将npm源设置成国内镜像源(如淘宝npm)：

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

或者通过 `.npmrc` 文件设置:

```
// .npmrc
registry=https://registry.npm.taobao.org/
```

## 原因二: binding.node源无法访问或速度慢

node-sass除了npm部分的代码，还会下载二进制文件 `binding.node` ，默认源是github，国内访问较慢，特殊时期甚至无法访问。我们也可以将其改成国内源：

```
// linux、mac 下
SASS_BINARY_SITE=https://npm.taobao.org/mirrors/node-sass/ npm install node-s

// window 下
set SASS_BINARY_SITE=https://npm.taobao.org/mirrors/node-sass/ && npm instal
```

或者通过 `.npmrc` 文件设置:

```
// .npmrc
```

```
sass_binary_site=https://npm.taobao.org/mirrors/node-sass/
```

有类似问题的还有 `chromedriver` , `phantomjs` , `electron` 等常见依赖包,我们可以一并写到 `.npmrc` 中:

```
// .npmrc
sass_binary_site=https://npm.taobao.org/mirrors/node-sass
chromedriver_cdnurl=https://npm.taobao.org/mirrors/chromedriver
phantomjs_cdnurl=https://npm.taobao.org/mirrors/phantomjs
electron_mirror=https://npm.taobao.org/mirrors/electron
```

## 原因三: node版本与node-sass版本不兼容

node-sass版本兼容性并不好，老项目中依赖的node-sass很可能已经不兼容新的node，对应版本兼容如下(或参考官方仓库):

| NodeJS | Minimum node-sass version | Node Module |
| --- | --- | --- |
| Node 13 | 4.13+ | 79 |
| Node 12 | 4.12+ | 72 |
| Node 11 | 4.10+ | 67 |
| Node 10 | 4.9+ | 64 |
| Node 8 | 4.5.3+ | 57 |

本文开头的安装例子中， `binding.node` 的版本是 `v4.13.0/win32-x64-64_binding.node` ,可以看到，里面包括node-sass版本号 `v4.13.0` ,平台 `win32` ,架构 `x64` ,以及 `Node Module` 的版本64。Node Module是node的一个模块，其版本号可以在进程 `process.versions` 中查到:

```
PS D:\demo> node
> console.log(process.versions);
{ http_parser: '2.8.0',
  node: '10.15.3',
  v8: '6.8.275.32-node.51',
  uv: '1.23.2',
  zlib: '1.2.11',
  ares: '1.15.0',
  modules: '64',
  nghttp2: '1.34.0',
  napi: '3',
  openssl: '1.1.0j',
  icu: '62.1',
```

```
  unicode: '11.0',
  cldr: '33.1',
  tz: '2018e' }
undefined
>
```

如上显示，node10.15.3对应的module版本为64。 假如node-sass与node的版本不兼容，就会找不到对应的 `binding.node` 而报错，例如你的node是10.15.3，装node-sass4.6.1，则会尝试安装 `v4.6.1/win32-x64-64_binding.node` ,但这个版本的 `binding.node` 是不存在的。 此时改node-sass或node的版本即可。

## 原因四: 缓存中binding.node版本不一致

假如本地node版本改了，或在不同机器上运行，node版本不一致，会报类似错误：

```
Found bindings for the following environments:
  - Windows 64-bit with Node.js 6.x
```

这是因为原有 `binding.node` 缓存跟现node版本不一致。按提示 `npm rebuild node-sass` 或清除缓存重新安装即可。

## 原因五: 安装失败后重新安装

安装失败后重新安装，有可能无权限删除已安装内容，此时 `npm uninstall node-sass` 或手动删掉原目录后再安装即可。

## 原因六: 编译失败,提示没有安装python、build失败等

假如拉取 `binding.node` 失败，node-sass会尝试在本地编译 `binding.node` ，过程就需要python。假如你遇到前面几种情况解决了，实际上也不会出现在本地编译的情况了，我们就不谈这种失败中失败的情况吧 :-)