

Node.js

- [【debug】centos7](#)
- [【npm】js 套件管理工具](#)
- [【nvm】Node.js 管理工具](#)

【debug】centos7

場景

centos7服務器使用nvm安裝的node之後，只要使用npm或者node，均會出現以下問題

<https://www.cnblogs.com/dingshaohua/p/17103654.html>

```
xxxxxxxxxx
```

1

```
[root@172 ~]# npm -v
```

2

```
node: /lib64/libm.so.6: version `GLIBC_2.27' not found (required by node)
```

3

```
node: /lib64/libc.so.6: version `GLIBC_2.25' not found (required by node)
```

4

```
node: /lib64/libc.so.6: version `GLIBC_2.28' not found (required by node)
```

5

```
node: /lib64/libstdc++.so.6: version `CXXABI_1.3.9' not found (required by node)
```

6

```
node: /lib64/libstdc++.so.6: version `GLIBCXX_3.4.20' not found (required by node)
```

7

```
node: /lib64/libstdc++.so.6: version `GLIBCXX_3.4.21' not found (required by node)
```

8

原因

查看系統內安裝的glibc版本

然後再根據分析可得知新版的node v18開始都需要GLIBC_2.27支持，可是目前系統內卻沒有那麼高的版本

```
xxxxxxxxxx
```

1

```
[root@172 glibc-2.28]# strings /lib64/libc.so.6 |grep GLIBC_
```

2

```
GLIBC_2.2.5
```

3

```
...
```

4

```
GLIBC_2.17
```

5

```
....
```

6

解決辦法

更新glibc

根據提示安裝所需要的glibc-2.28

```
xxxxxxxxxx
```

1

```
wget http://ftp.gnu.org/gnu/glibc/glibc-2.28.tar.gz
```

2

```
tar xf glibc-2.28.tar.gz
```

3

```
cd glibc-2.28/ && mkdir build && cd build
```

4

```
../configure --prefix=/usr --disable-profile --enable-add-ons --with-headers=/usr/include --with-binutils=/usr/bin
```

5

可能出現的錯誤

上步更新glibc 可能會發生錯誤。
如果沒有錯誤下邊這一步不用看。

make問題

```
xxxxxxxxxx
```

1

```
configure: error:
```

2

```
*** These critical programs are missing or too old: make bison compiler
```

3

```
*** Check the INSTALL file for required versions.
```

4

解決辦法：升級gcc與make

```
xxxxxxxxxx
```

1

```
# 升GCC(默4 升8)
```

2

```
yum install -y centos-release-scl
```

3

```
yum install -y devtoolset-8-gcc*
```

4

```
mv /usr/bin/gcc /usr/bin/gcc-4.8.5
```

5

```
ln -s /opt/rh/devtoolset-8/root/bin/gcc /usr/bin/gcc
```

6

```
mv /usr/bin/g++ /usr/bin/g++-4.8.5
```

7

```
ln -s /opt/rh/devtoolset-8/root/bin/g++ /usr/bin/g++
```

8

9

```
# 升 make(默3 升4)
```

10

```
wget http://ftp.gnu.org/gnu/make/make-4.3.tar.gz
```

11

```
tar -xzf make-4.3.tar.gz && cd make-4.3/
```

12

```
./configure --prefix=/usr/local/make
```

13

```
make && make install
```

14

```
cd /usr/bin/ && mv make make.bak
```

15

```
ln -sv /usr/local/make/bin/make /usr/bin/make
```

16

這時所有的問題都已經解決完畢再重新執行上一步更新glibc即可

```
xxxxxxxxxx
```

1

```
cd /root/glibc-2.28/build
```

2

```
../configure --prefix=/usr --disable-profile --enable-add-ons --with-headers=/usr/include --with-binutils=/usr/bin
```

3

我的依舊報錯：bison太老舊

```
xxxxxxxxxx
```

1

```
configure: error:
```

2

```
*** These critical programs are missing or too old: bison
```

3

```
*** Check the INSTALL file for required versions.
```

4

看看我的bison版本多少

```
xxxxxxxxxx
```

1

```
[root@172 ~]# bison -v
```

2

```
-bash: bison: 未找到命令
```

3

bison問題

嗨，沒裝啊。裝一下唄

```
xxxxxxxxxx
```

1

```
yum install -y bison
```

2

這時所有的問題真的真的都已經解決完畢再重新執行上一步更新glibc即可

```
xxxxxxxxxx
```

1

```
cd /root/glibc-2.28/build
```

2

```
../configure --prefix=/usr --disable-profile --enable-add-ons --with-headers=/usr/include --with-binutils=/usr/bin
```

3

繼續更新glibc

make 和make install在linux中就是安裝軟件的意思簡單這麼理解就好。
這個過程較長，大約半小時左右，建議打一局遊戲就好了。

```
xxxxxxxxxx
```

1

```
make && make install
```

2

```
``
```

3

4

```
□□下 是不是好了
```

5

```
```shell
```

6

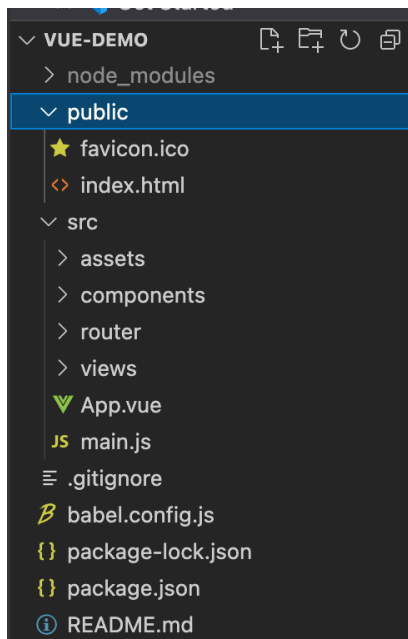
```
npm -v
```

7

- [場景](#)
- [原因](#)
- [解決辦法](#)
- [更新glibc](#)
- [可能出現的錯誤](#)
- [make問題](#)
- [bison問題](#)
- [繼續更新glibc](#)

# 【npm】js 套件管理工具

```
安裝套件
npm install 套件名
```



node\_modules => 套件檔案放置處

public => 打包後的進入點

src => 原始檔

package-lock.json 目前套件版本安裝詳細資訊

package.json => npm 設定檔

```
1 {
2 "private": true,
3 "scripts": {
4 "dev": "npm run development",
5 "development": "mix",
6 "watch": "mix watch",
7 "watch-no11": "mix watch -- --watch-options-poll=1000",
8 "執行指令碼 | 對指令碼偵錯": "t",
9 "prod": "npm run production",
10 "production": "mix --production"
11 },
12 "devDependencies": {
13 "@popperjs/core": "^2.10.2",
14 "@vue/compiler-sfc": "^3.2.24",
15 "axios": "^0.21",
16 "bootstrap": "^5.1.3",
17 "laravel-mix": "^6.0.6",
18 "lodash": "^4.17.19",
19 "postcss": "^8.1.14",
20 "resolve-url-loader": "^3.1.2",
21 "sass": "^1.32.11",
22 "sass-loader": "^11.0.1",
23 "vue": "^3.2.24",
24 "vue-loader": "^16.1.0",
25 "vue-template-compiler": "^2.6.10"
26 },
27 "dependencies": {
28 "bootstrap-vue": "^2.21.2",
29 "element-plus": "^1.2.0-beta.6",
30 "vue-router": "^4.0.5"
31 }
32 }
```

script : npm 指令

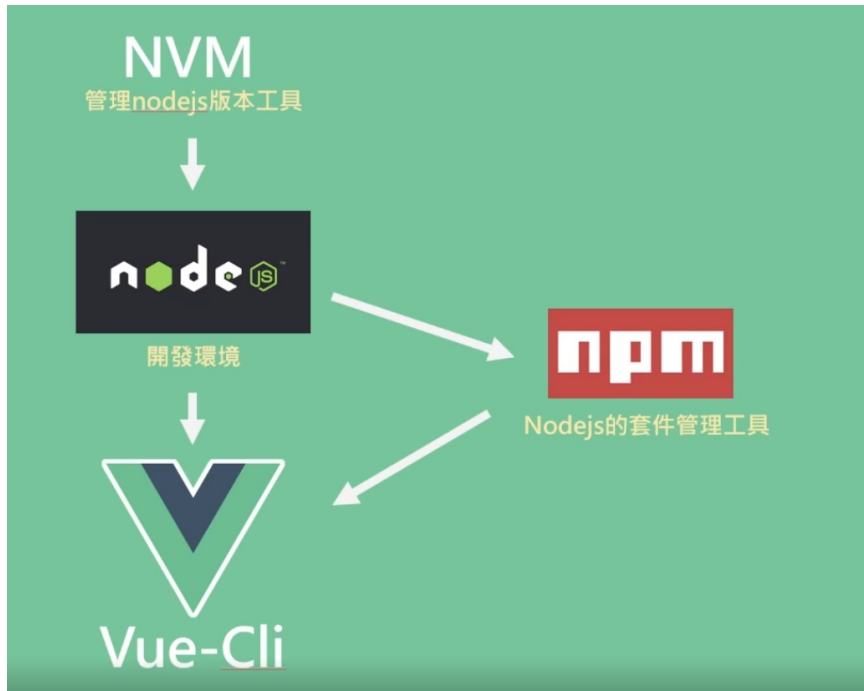
ex. npm run dev => npm run development



# 【nvm】Node.js 管理工具

- nvm

<https://github.com/nvm-sh/nvm>



## Mac 安裝nvm

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh | bash
```

如果下nvm指令找不到

在以下檔案(依照你常用的套件擇一)加入以下文字

```
bash: vim ~/.bashrc
```

```
zsh: vim ~/.zshrc
```

```
export NVM_DIR="$([-z "${XDG_CONFIG_HOME-}"] && printf %s "${HOME}/.nvm" || printf %s "${XDG_CONFIG_HOME}/nvm")"
[-s "$NVM_DIR/nvm.sh"] && \. "$NVM_DIR/nvm.sh" # This loads nvm
```

重新登入或是依使用shell則一重新讀入設定檔

```
bash: source ~/.bashrc
```

```
zsh: source ~/.zshrc
```

參考: <https://github.com/nvm-sh/nvm#installing-and-updating>

## MVN 常用指令

- nvm ls-remote : 列出目前可用的遠端 Node.js 版本
- nvm install : 安裝特定版本的 Node.js
- nvm ls : 列出本定端所安裝的 Node.js 環境
- nvm alias default node : 設定命令列預設開啟的 Node.js 版本
- nvm use : 當前命令列套用特定版本的 Node.js

```
列出目前電腦有安裝nodejs版本
nvm list
```

```
目前網路上可用的nodejs 版本
nvm list available
```

```
#安裝12.19.0版本
nvm install v12.19.0
```



```
#移除12.19.0版本
npm uninstall v12.19.0
```

```
#使用12.19.0版本
npm use 12.19.0
```

```
#查看目前版本
npm -v
```