node.js环境的安装

1. **通过源代码安装**

廊坊机器安装环境

1. 安装mysql和git客户端
2. 下载nodejs源代码
3. 安装nodejs编译环境：

因为源码有要求：
* `gcc` and `g++` 4.8 or newer, or
* `clang` and `clang++` 3.4 or newer
* Python 2.6 or 2.7
* GNU Make 3.81 or newer
* libexecinfo (FreeBSD and OpenBSD only)

先安装gcc和g++

yum search policycoreutils-python
rpm -ivh http://mirror.centos.org/centos/6/sclo/x86_64/rh/devtoolset-4/devtoolset-4-runtime-4.0-9.el6.x86_64.rpm
rpm -ivh http://mirror.centos.org/centos/6/sclo/x86_64/rh/devtoolset-4/devtoolset-4-binutils-2.25-10.el6.x86_64.rpm
rpm -ivh http://mirror.centos.org/centos/6/sclo/x86_64/rh/devtoolset-4/devtoolset-4-libstdc++-devel-5.2.1-2.2.el6.x86_64.rpm
rpm -ivh http://mirror.centos.org/centos/6/sclo/x86_64/rh/devtoolset-4/devtoolset-4-gcc-c++-5.2.1-2.2.el6.x86_64.rpm

rpm -ivh http://mirror.centos.org/centos/6/sclo/x86_64/rh/devtoolset-4/devtoolset-4-gcc-5.2.1-2.2.el6.x86_64.rpm

/etc/bashrc
source /opt/rh/devtoolset-4/enable

正式开始按照：
已经下载好了：centos/devtools*
yum install policycoreutils-python -y
rpm -ivh devtoolset-4-runtime-4.0-9.el6.x86_64.rpm
rpm -ivh devtoolset-4-binutils-2.25-10.el6.x86_64.rpm
rpm -ivh devtoolset-4-libstdc++-devel-5.2.1-2.2.el6.x86_64.rpm
rpm -ivh devtoolset-4-gcc-5.2.1-2.2.el6.x86_64.rpm
rpm -ivh devtoolset-4-gcc-c++-5.2.1-2.2.el6.x86_64.rpm

然后修改/etc/bashrc
source /opt/rh/devtoolset-4/enable
进入nodes的源代码：

```
$ ./configure
$ make
$make install
```

验证：

```
$node -v
$ npm –v
```

2 使用yum安装

```
yum install -y nodejs
```

这样就保证了node和npm都是稳定版本或者最新版本了。

3. **使用nvm**

Node Version Manager - Simple bash script to manage multiple active node.js versions
https://github.com/creationix/nvm

n https://github.com/tj/n

nvm
http://weizhifeng.net/node-version-management-via-n-and-nvm.html
http://www.cnblogs.com/kaiye/p/4937191.html

五、nvm 与 n 的区别

node 版本管理工具还有一个是 TJ 大神的 n 命令，n 命令是作为一个 node 的模块而存在，而 nvm 是一个独立于 node/npm 的外部 shell 脚本，因此 n 命令相比 nvm 更加局限。

由于 npm 安装的模块路径均为 /usr/local/lib/node_modules ，当使用 n 切换不同的 node 版本时，实际上会共用全局的 node/npm 目录。 因此不能很好的满足『按不同 node 版本使用不同全局 node 模块』的需求。

因此建议各位尽早开始使用 nvm ，以免出现全局模块无法更新的问题。

管理 node 版本，选择 nvm 还是 n?
http://taobaofed.org/blog/2015/11/17/nvm-or-n/

```
nvm ls-remote
```

nvm current	Display currently activated version
nvm ls	List installed versions
vm ls-remote	List remote versions available for install
--lts	When listing, only show LTS (long-term support) versions

```
Example:
  nvm install v0.10.32                  Install a specific version number
  nvm use 0.10                          Use the latest available 0.10.x release
  nvm run 0.10.32 app.js                Run app.js using node v0.10.32
  nvm exec 0.10.32 node app.js          Run `node app.js` with the PATH pointing to
node v0.10.32
  nvm alias default 0.10.32             Set default node version on a shell
```

node use v4.5.0 就是把4.5设置为默认版本

curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.31.4/install.sh | bash

```
$ curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.31.4/install.sh | bash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  9135  100  9135    0     0  53047      0 --:--:-- --:--:-- --:--:--  446k
=> Downloading nvm from git to '/home/lizhishan/.nvm'
=> Initialized empty Git repository in /home/lizhishan/.nvm/.git/
remote: Counting objects: 5225, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 5225 (delta 0), reused 0 (delta 0), pack-reused 5222
Receiving objects: 100% (5225/5225), 1.42 MiB | 768 KiB/s, done.
Resolving deltas: 100% (3148/3148), done.
Your version of git is out of date. Please update it!

=> Appending source string to ~/.bashrc
bash: line 293: /nvm.sh: No such file or directory
=> Close and reopen your terminal to start using nvm or run the following to use it
now:

export NVM_DIR="~/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && . "$NVM_DIR/nvm.sh"  # This loads nvm
```

```
source ~/.bash_profile
```

因为添加到了~/.bashrc文件中。

```
$ nvm install latest
Version 'latest' not found - try `nvm ls-remote` to browse available versions.
```

单独升级node和npm

Updating Globally-Installed Packages

npm update -g will apply the **update** action to each globally installed package that is **outdated** -- that is, has a version that is different from **latest**.

来自：https://docs.npmjs.com/cli/update

另有说升级node的方式：
npm update [-g] [<pkg>...]

How do I update npm?

npm install -g npm
Please note that this command will remove your current version of npm. Make sure to use sudo npm install -g npm if on a Mac.
You can also update all outdated local packages by doing npm update without any arguments, or global packages by doing **npm update -g.**
Occasionally, the version of npm will progress such that the current version cannot be properly installed with the version that you have installed already. (Consider, if there is ever a bug in the update command.) In those cases, you can do this:
curl https://www.npmjs.com/install.sh | sh

使用npm安装package

http://javascript.tutorialhorizon.com/2014/09/21/installing-listing-and-uninstalling-packages-using-npm/

Listing installed packages using npm

If at any point you want to see the list all the packages that are installed and their dependen-cies, you can do that by the command

```
npm list
```

If the tree it spits out is just too much detail, you can specify the depth upto which you are interested by the command

```
npm list --depth=0
```

where '0' is the first level of depedencies.

You can also run the above commands with the –g option to get details of all your globally installed modules.

```
npm list -g --depth=0
```

Removing unused packages from node_modules

Sometimes after installing packages, you realize that you dont really need some of them and you delete those entries from your package.json. Although those packages will not be installed again on running an npm install, you still need to remove them from your node_modules folder at least once. To remove all such unused packages from your node_modules, you can run the command

```
npm prune
```

Removing all devDependencies from node_modules

If you just want to delete all of your installed devDependencies while retaining their entry in the package.json file, you can do that using the command

```
npm prune --production
```

After this command, the only packages left in your node_modules directory will be the ones that are specified in your dependencies map.

http://javascript.tutorialhorizon.com/2014/08/26/setting-up-grunt-to-automate-repetitive-tasks-in-javascript-applications/

$ npm init
生产package.json

设置执行的命令