comment gérer des versions multiples de nœuds en utilisant le manager NVM node version

NVM or Node Version Manager is a command-line tool for managing multiple Nodejs versions. It's a POSIX-compliant bash script that allows you to install and manage multiple nodejs versions on your computer.

In this tutorial, I will show you how to install and use NVM for managing multiple Node.js versions on your computer. This guide can be applied to different Linux distributions, including Ubuntu, CentOS, and Debian.

1. Install NVM (Node Version Manager)

Firstly, we're going to install package dependencies for the nvm installation.

For Ubuntu, you can run the apt command below.

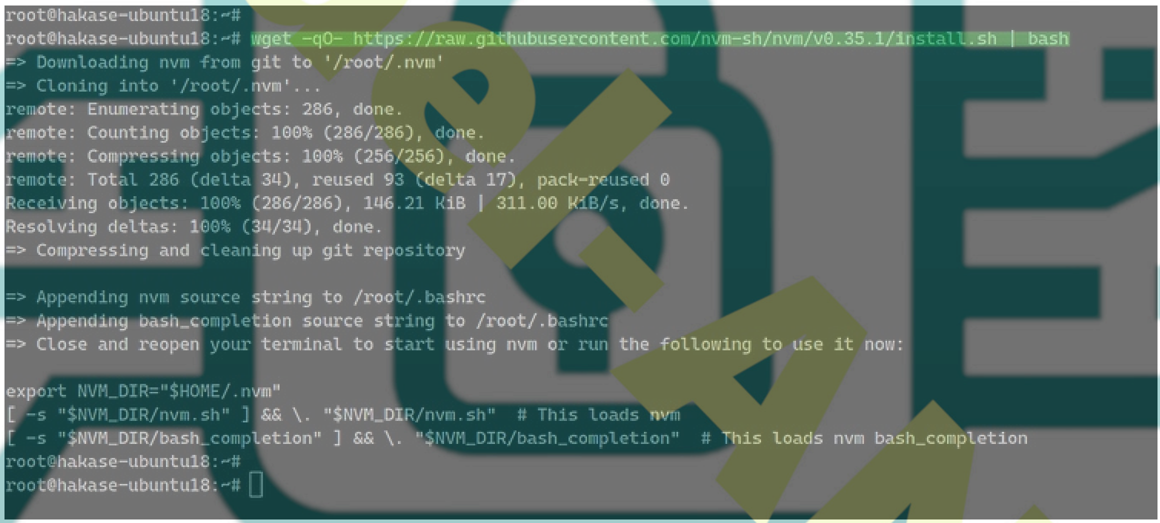
```
sudo apt install build-essential libssl-dev -y
```

And for CentOS, you can use the dnf command as below.

```
sudo dnf group install "Development Tools" -y
```

After that, download and run the nvm installer script as below.

```
wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.5/install.sh | bash
```



```
root@hakase-ubuntu18:~#
root@hakase-ubuntu18:~# wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.1/install.sh | bash
=> Downloading nvm from git to '/root/.nvm'
=> Cloning into '/root/.nvm'...
remote: Enumerating objects: 286, done.
remote: Counting objects: 100% (286/286), done.
remote: Compressing objects: 100% (256/256), done.
remote: Total 286 (delta 34), reused 93 (delta 17), pack-reused 0
Receiving objects: 100% (286/286), 146.21 KiB | 311.00 KiB/s, done.
Resolving deltas: 100% (34/34), done.
=> Compressing and cleaning up git repository

=> Appending nvm source string to /root/.bashrc
=> Appending bash_completion source string to /root/.bashrc
=> Close and reopen your terminal to start using nvm or run the following to use it now:

export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"  # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"  # This loads nvm bash_completion
root@hakase-ubuntu18:~#
root@hakase-ubuntu18:~#
```

Once the installation is complete, you need to reload the '~/.bashrc' script.

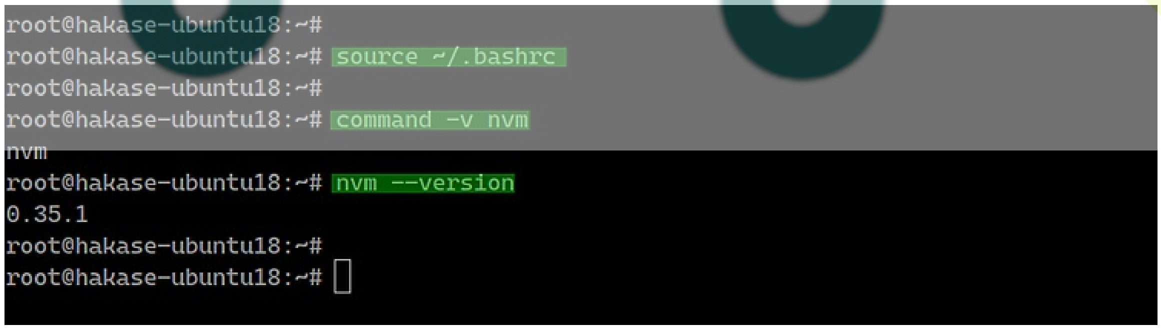
```
source ~/.bashrc
```

Now the nvm installation is completed, and you're able to run the nvm command.

Check the nvm version as below.

```
nvm --version
```

You will be shown the result as below.



```
root@hakase-ubuntu18:~#
root@hakase-ubuntu18:~# source ~/.bashrc
root@hakase-ubuntu18:~#
root@hakase-ubuntu18:~# command -v nvm
nvm
root@hakase-ubuntu18:~# nvm --version
0.35.1
root@hakase-ubuntu18:~#
root@hakase-ubuntu18:~#
```

Next, we're going to show you the basic usage of nvm for managing the nodejs installation.

2. Check Available NodeJS

To check available nodejs on your local system, you can use the 'ls' option.

```
nvm ls
```

Also, you can check to list remote versions of nodejs that ready to install.

```
nvm ls-remote
```

And to specify nodejs 'LTS' version, you can use the '--lts' option.

```
nvm ls-remote --lts
```

3. Install NodeJS Version

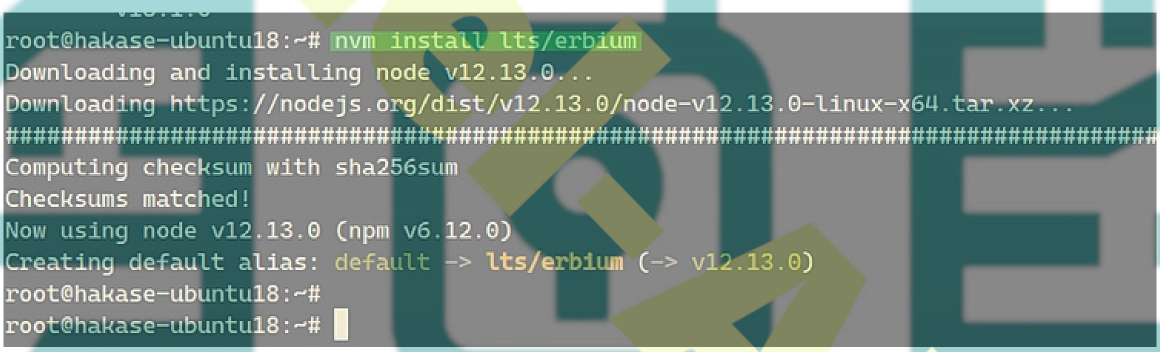
To install the latest version of nodejs, you can use the following command.

```
nvm install node
```

To install a specific version, you can use the version number or using the alias of the version.

Install nodejs version with alias.

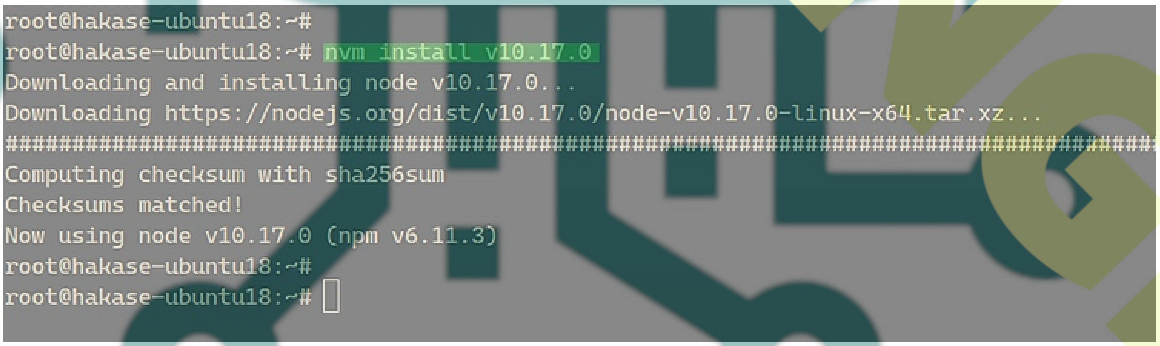
```
nvm install lts/erbium
```



```
root@hakase-ubuntu18:~# nvm install lts/erbium
Downloading and installing node v12.13.0...
Downloading https://nodejs.org/dist/v12.13.0/node-v12.13.0-linux-x64.tar.xz...
################################################################################
Computing checksum with sha256sum
Checksums matched!
Now using node v12.13.0 (npm v6.12.0)
Creating default alias: default -> lts/erbium (-> v12.13.0)
root@hakase-ubuntu18:~#
root@hakase-ubuntu18:~#
```

Install node js with version number.

```
nvm install v10.17.0
```



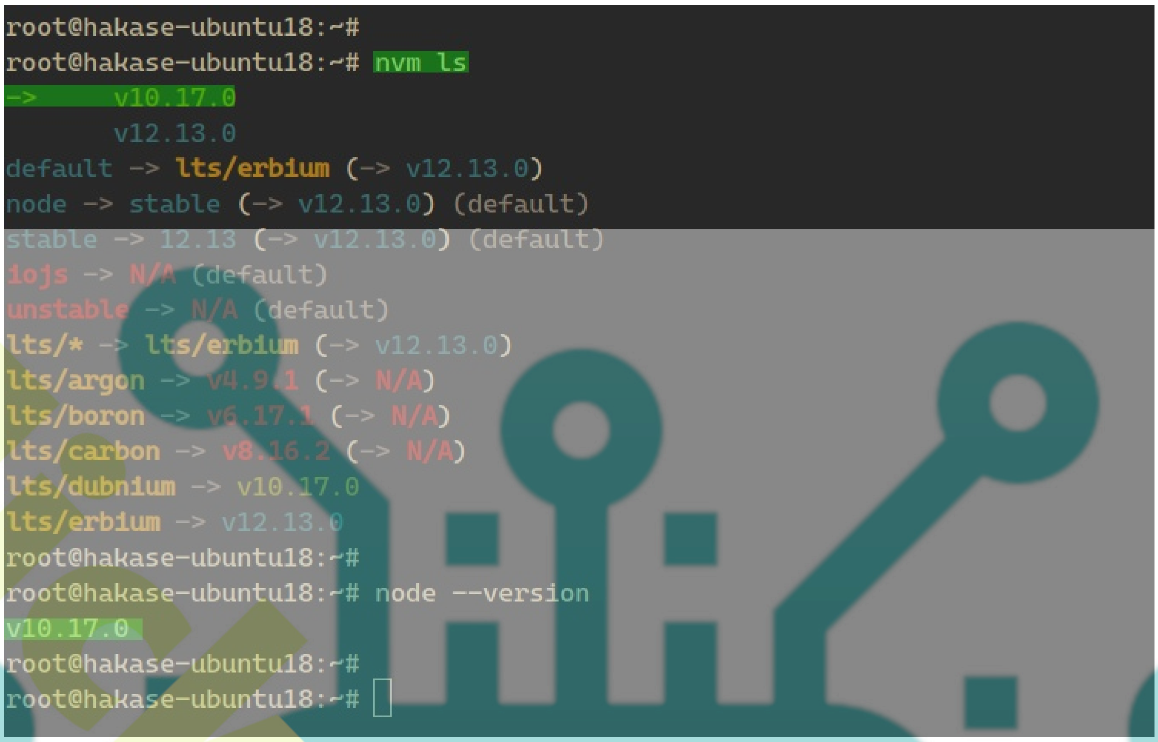
```
root@hakase-ubuntu18:~#
root@hakase-ubuntu18:~# nvm install v10.17.0
Downloading and installing node v10.17.0...
Downloading https://nodejs.org/dist/v10.17.0/node-v10.17.0-linux-x64.tar.xz...
################################################################################
Computing checksum with sha256sum
Checksums matched!
Now using node v10.17.0 (npm v6.11.3)
root@hakase-ubuntu18:~#
root@hakase-ubuntu18:~#
```

4. Switch to Different NodeJS Versions

The main advantage of using nvm is that we can switch different nodejs version with the simple command.

Firstly, you can check available nodejs on your local system.

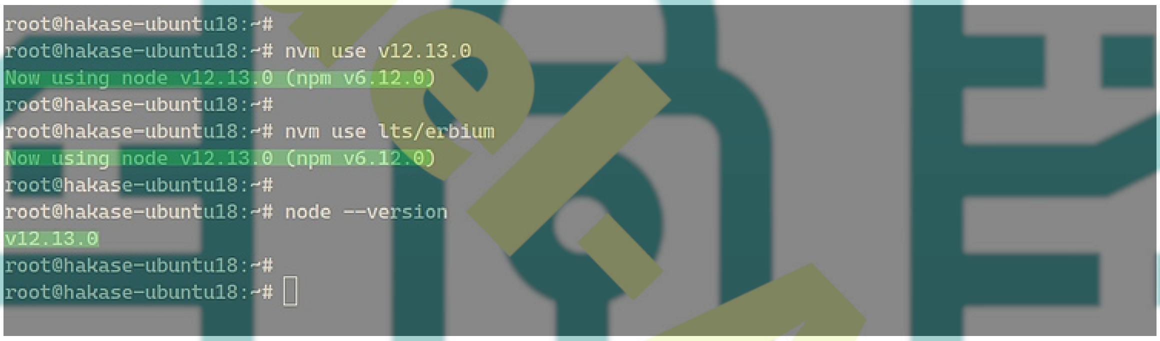
```
nvm ls
```



```
root@hakase-ubuntu18:~#
root@hakase-ubuntu18:~# nvm ls
->       v10.17.0
         v12.13.0
default -> lts/erbium (-> v12.13.0)
node -> stable (-> v12.13.0) (default)
stable -> 12.13 (-> v12.13.0) (default)
iojs -> N/A (default)
unstable -> N/A (default)
lts/* -> lts/erbium (-> v12.13.0)
lts/argon -> v4.9.1 (-> N/A)
lts/boron -> v6.17.1 (-> N/A)
lts/carbon -> v8.16.2 (-> N/A)
lts/dubnium -> v10.17.0
lts/erbium -> v12.13.0
root@hakase-ubuntu18:~#
root@hakase-ubuntu18:~# node --version
v10.17.0
root@hakase-ubuntu18:~#
root@hakase-ubuntu18:~#
```

To switch to the different versions of your default, you can use the 'use' option following by the version number or alias.

```
nvm use v12.13.0
nvm use lts/erbium
```



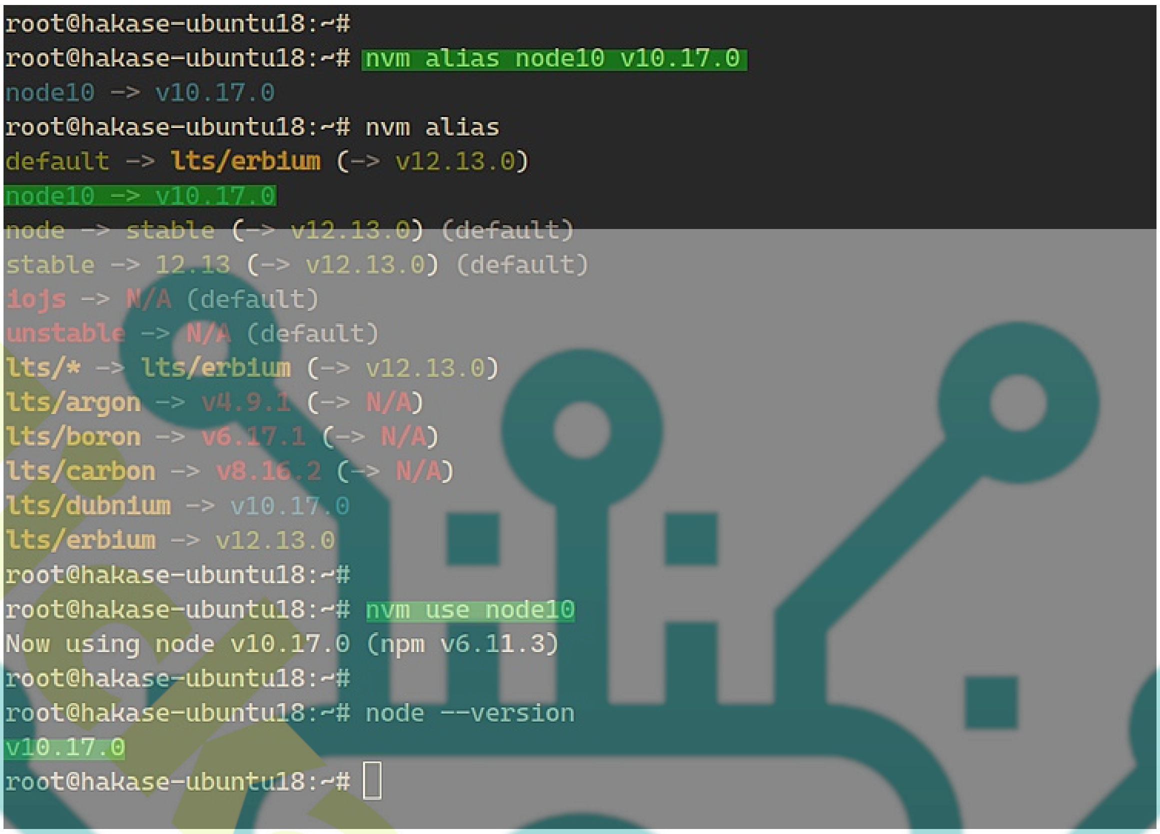
```
root@hakase-ubuntu18:~#
root@hakase-ubuntu18:~# nvm use v12.13.0
Now using node v12.13.0 (npm v6.12.0)
root@hakase-ubuntu18:~#
root@hakase-ubuntu18:~# nvm use lts/erbium
Now using node v12.13.0 (npm v6.12.0)
root@hakase-ubuntu18:~#
root@hakase-ubuntu18:~# node --version
v12.13.0
root@hakase-ubuntu18:~#
root@hakase-ubuntu18:~#
```

5. Create an Alias

This feature allows you to create your alias for your specific nodejs version. And make easier to manage nodejs version as you need a developer.

To create an alias, you can use the 'alias' option following by the alias name and the version of nodejs.

```
nvm alias node10 v10.17.0
```


```
root@hakase-ubuntu18:~#
root@hakase-ubuntu18:~# nvm alias node10 v10.17.0
node10 -> v10.17.0
root@hakase-ubuntu18:~# nvm alias
default -> lts/erbium (-> v12.13.0)
node10 -> v10.17.0
node -> stable (-> v12.13.0) (default)
stable -> 12.13 (-> v12.13.0) (default)
iojs -> N/A (default)
unstable -> N/A (default)
lts/* -> lts/erbium (-> v12.13.0)
lts/argon -> v4.9.1 (-> N/A)
lts/boron -> v6.17.1 (-> N/A)
lts/carbon -> v8.16.2 (-> N/A)
lts/dubnium -> v10.17.0
lts/erbium -> v12.13.0
root@hakase-ubuntu18:~#
root@hakase-ubuntu18:~# nvm use node10
Now using node v10.17.0 (npm v6.11.3)
root@hakase-ubuntu18:~#
root@hakase-ubuntu18:~# node --version
v10.17.0
root@hakase-ubuntu18:~#
```

Now check available 'alias' using the following command.

```
nvm alias
```

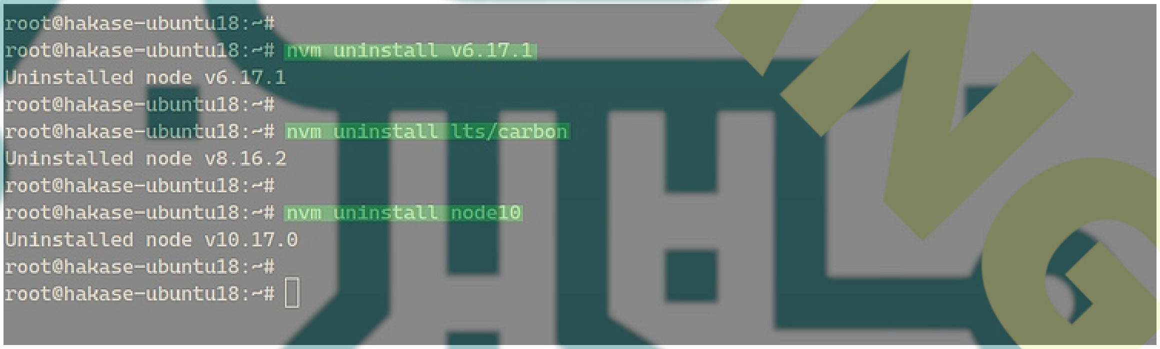
And you will get your alias on the list.

6. Uninstall NodeJS version

To remove the specific nodejs version, you can use the 'uninstall' option followed by the version number or alias.

```
nvm uninstall v6.17.1
nvm uninstall lts/carbon
nvm uninstall node10
```

And the nodejs version that you want to remove has been uninstalled from the system.


```
root@hakase-ubuntu18:~#
root@hakase-ubuntu18:~# nvm uninstall v6.17.1
Uninstalled node v6.17.1
root@hakase-ubuntu18:~#
root@hakase-ubuntu18:~# nvm uninstall lts/carbon
Uninstalled node v8.16.2
root@hakase-ubuntu18:~#
root@hakase-ubuntu18:~# nvm uninstall node10
Uninstalled node v10.17.0
root@hakase-ubuntu18:~#
root@hakase-ubuntu18:~#
```

7. Run Script with Specific NodeJS version

With the nvm, we can directly run a js application with the specific version of nodejs.

This feature features can be very useful if you're creating or running JS applications with a different version of nodejs.

Run the 'app.js' script using the lts version of nodejs 'lts/erbium'.

```
nvm run lts/erbium app.js
nvm run v12.13.0 app.js
```

That's all about the nvm installation and its basic usage for managing multiple nodejs version on Linux operating system.

Reference

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

Mickael-ANGEL