nvm Node version manager

START HERE

`nvm install --lts`
Install latest LTS

`nvm current`
Show active version

`nvm ls`
List installed versions

`nvm ls-remote --lts`
List LTS releases

`nvm use <version>`
Switch this shell

`nvm use`
Follow .nvmrc

`nvm alias default <version>`
Pick the default

`nvm uninstall <version>`
Remove a version

.nvmrc version in project → `nvm use` → **nvm** shell function → `sets PATH` → **~/.nvm/versions** installed Node → `first on PATH` → **node, npm** this shell only

CONCEPTS

nvm is a shell function, not a program: it only exists in shells that source nvm.sh, and it changes PATH of that shell only.

Global npm packages belong to one Node version; a new version starts without them.

Homebrew-installed nvm is not supported by the project; use the install script or a git clone.

INSTALL NVM

`curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.8/install.sh | bash`
Install nvm with curl

`wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.8/install.sh | bash`
Install nvm with wget

`nvm --version`
Show the nvm version

`command -v nvm`
Check that nvm is loaded
Prints nvm when the function exists; empty output means the shell never sourced nvm.sh.

`source ~/.bashrc`
Reload the shell profile after install
Use ~/.zshrc for zsh. Or just open a new terminal.

`PROFILE=/dev/null`
Stop the installer editing your profile
Set it in front of the install script's bash, then add the source lines yourself.

`git fetch --tags origin`
Update a git-clone install (then check out the tag)

INSTALL AND USE

`nvm install <version>`
Install a Node version

`nvm install --lts`
Install the newest LTS

`nvm install node`
Install the latest release

`nvm install`
Install the version from .nvmrc

`nvm use <version>`
Use a version in this shell

`nvm use --lts`
Use the newest LTS

INSTALL AND USE (cont.)

`nvm use node`
Use the latest release

`nvm use system`
Use the system-installed Node

`nvm use default`
Return to the default version

`nvm use`
Use the version from .nvmrc

`nvm install --lts=<name>`
Install a named LTS line

`nvm install <version> --latest-npm`
Upgrade npm to the newest it supports

`nvm install-latest-npm`
Upgrade npm in the active version

`nvm install <version> --reinstall-packages-from=<old>`
Install and copy global packages

`nvm install -s <version>`
Compile from source

INSPECT VERSIONS

`nvm ls`
List installed versions

`nvm ls-remote`
List all installable versions

`nvm ls-remote --lts`
List LTS versions only

`nvm current`
Show the active version

`nvm which <version>`
Print the path of that node binary

`nvm which current`
Print the path of the active node

`nvm version-remote --lts`
Show the newest LTS version number

`nvm uninstall <version>`
Remove an installed version

ALIASES AND DEFAULTS

`nvm alias default <version>`
Set the version new shells start with

`nvm alias default node`
Default to whatever is latest

`nvm alias default lts/*`
Default to the newest LTS

`nvm alias <name> <version>`
Create your own alias

ALIASES AND DEFAULTS (cont.)

`nvm alias`
List all aliases

`nvm unalias <name>`
Delete an alias

RUN WITHOUT SWITCHING

`nvm run <version> <script.js>`
Run node under that version

`nvm exec <version> <command>`
Run any command under that version

`nvm run --lts <script.js>`
Run node under the newest LTS

`nvm exec --lts npm test`
Run npm under the newest LTS

nvm run and exec fall back to the active Node when no .nvmrc is found; always pass a version in scripts.

.NVMRC AND PACKAGES

`echo "lts/*" > .nvmrc`
Pin the project to the newest LTS

`node -v > .nvmrc`
Pin the project to the running version

`nvm install && nvm use`
Install then use the pinned version

`nvm reinstall-packages <version>`
Copy that version's global packages here
Installs into the active version; it does not touch npm itself.

`$NVM_DIR/default-packages`
File of packages installed with every new Node
One package per line, anything npm accepts, such as rimraf or object-inspect@1.0.2.

.nvmrc contents

Value	Meaning
`20`	Any installed or installable 20.x
`20.11.1`	That exact release
`lts/*`	Newest LTS
`lts/iron`	A named LTS line
`node`	Latest release

nvm looks for .nvmrc upward through parent directories and exits with status 127 if it finds none. Lines starting with # are comments.

SHELL INTEGRATION

`export NVM_DIR="$HOME/.nvm"`
Where nvm lives

`[ -s "$NVM_DIR/nvm.sh" ] && . "$NVM_DIR/nvm.sh"`
Load nvm in your shell profile

`[ -s "$NVM_DIR/nvm.sh" ] && . "$NVM_DIR/nvm.sh" --no-use`
Load nvm without activating the default
Skips the default-version setup, which makes shell startup faster. Run nvm use when you need Node.

`NVM_NODEJS_ORG_MIRROR=<url> nvm install <version>`
Install from a mirror

`NVM_NO_SOURCE_FALLBACK=1`
Error instead of compiling when no binary exists

Auto-switching on cd is a community recipe (a cd wrapper in bash, a chpwd hook in zsh), not a built-in feature.

Slow shell startup is usually nvm.sh being sourced eagerly. Add --no-use, or load it lazily on first use of nvm, node or npm.

TROUBLESHOOTING

nvm: command not found means the shell did not source nvm.sh. Open a new terminal or reload your profile.

RECIPES

Set up a machine

```
curl -o- https://raw.
githubusercontent.com/nvm-sh/
nvm/v0.40.8/install.sh | bash
# open a new terminal, then
nvm install --lts
nvm alias default lts/*
node -v
```

Pin a project

```
cd myapp
node -v > .nvmrc
git add .nvmrc
# teammates run
nvm install && nvm use
```

Upgrade Node and keep global packages

```
nvm install --lts --reinstall-
packages-from=current
nvm alias default lts/*
nvm uninstall <old-version>
```

Test on several versions

```
for v in 20 22 24; do
  nvm exec $v npm test
done
```