fnm Fast Node manager

START HERE

`fnm` **`install --lts`**
Install the newest LTS

`node --version > .node-version`
Pin a project

`fnm` **`list`**
List installed versions

`fnm` **`exec --using`** *`<version>`* *`<cmd>`*
Run once under a version

`fnm` **`use`** *`<version>`*
Switch this shell

`fnm` **`default`** *`<version>`*
Set the default for new shells

`fnm` **`current`**
Show the active version

`fnm` **`env --use-on-cd --shell bash`**
Shell setup line (in eval)

.node-version version in project → `cd (use-on-cd)` → **fnm env hook** in your shell profile → `fnm use` → **Shell PATH** per-shell symlink → `points to` → **Node version** in ~/.local/share/fnm

CONCEPTS

fnm is a single binary, not a shell function. It only works after your shell profile evals `fnm env`, which gives each shell its own PATH entry.

fnm reads .node-version and .nvmrc. Without --use-on-cd nothing switches by itself when you change directory.

INSTALL FNM

`curl -fsSL https://fnm.vercel.app/install | bash`
Install with the script
Flags after bash -s --: --install-dir <dir>, --skip-shell (leave the profile alone).

`brew install fnm`
Install with Homebrew

`fnm` **`--version`**
Show the fnm version

`fnm` **`completions --shell`** *`<shell>`*
Print completions (bash, zsh, fish, powershell)

SHELL SETUP

`eval "$(fnm env --use-on-cd --shell bash)"`
Bash profile line (~/.bashrc)

`eval "$(fnm env --use-on-cd --shell zsh)"`
Zsh profile line (~/.zshrc)

`fnm` **`env --use-on-cd --shell fish | source`**
Fish: conf.d/fnm.fish

`fnm` **`env --version-file-strategy recursive`**
Look for version files in parent dirs
Default is local: only the current directory is checked. Add the flag to the eval line.

INSTALL AND UNINSTALL

`fnm` **`install`** *`<version>`*
Install a version (partial semver is fine)

`fnm` **`install --lts`**
Install the newest LTS

`fnm` **`install --latest`**
Install the newest release

`fnm` **`install`**
Install the version from .node-version

INSTALL AND UNINSTALL (cont.)

`fnm` **`uninstall`** *`<version>`*
Remove an installed version

USE AND INSPECT

`fnm` **`use`** *`<version>`*
Use a version in this shell

`fnm` **`use`**
Use the version from .node-version

`fnm` **`use --install-if-missing`**
Use it, installing first if needed

`fnm` **`current`**
Show the active version

`fnm` **`list`**
List installed versions and aliases

`fnm` **`list-remote`**
List all installable versions

`fnm` **`list-remote --lts`**
List LTS versions only

ALIASES AND DEFAULTS

`fnm` **`default`** *`<version>`*
Set the version new shells start with

`fnm` **`alias`** *`<version>`* *`<name>`*
Name a version

RUN WITHOUT SWITCHING

`fnm` **`exec --using`** *`<version>`* *`<command>`*
Run a command under that version

`fnm` **`exec --using`** *`<version>`* **`npx`** *`<pkg>`*
Try a package on the newest LTS

VERSION FILES AND CONFIG

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

VERSION FILES AND CONFIG (cont.)

Version file contents

Value	Meaning
`22`	Any 22.x
`22.11.0`	That exact release
`v22.11.0`	Same, with a v prefix
`lts/*`	Newest LTS
`lts/iron`	A named LTS line

fnm checks .node-version first, then .nvmrc; package.json engines.node is a fallback only with --resolve-engines.

Environment variables

Variable	Sets
`FNM_DIR`	Where versions are stored
`FNM_LOGLEVEL`	quiet, error or info
`FNM_NODE_DIST_MIRROR`	Download mirror
`FNM_ARCH`	Binary architecture
`FNM_VERSION_FILE_STRATEGY`	local or recursive
`FNM_COREPACK_ENABLED`	Run corepack enable
`FNM_RESOLVE_ENGINES`	Use package.json engines

TROUBLESHOOTING

node not found after fnm install: you skipped the eval line, or fnm default is unset. Run fnm default <version> and open a new shell.

RECIPES

Set up a machine

```
curl -fsSL https://fnm.vercel.app/install | bash
# open a new terminal, then
fnm install --lts
fnm default <version>
node --version
```

Pin a project

```
cd myapp
node --version > .node-version
git add .node-version
# teammates run
fnm use --install-if-missing
```

Auto-switch on cd

```
# in ~/.bashrc
eval "$(fnm env --use-on-cd --version-file-strategy recursive --shell bash)"
# then cd into any project with a version file
```

Test on several versions

```
for v in 20 22 24; do
  fnm exec --using $v npm test
done
```