direnv Per-directory environments

CONCEPTS

direnv loads .envrc when you cd in and unloads it when you leave. It must be allowed first, and again after every edit.

cd into dir shell prompt
↓ hook
Find .envrc checks allow list
↓ allowed?
Run in bash stdlib available
↓ export diff
Shell env restored on leaving

The .envrc runs in bash, whatever your interactive shell is. Aliases and functions cannot be exported this way.

Never allow an .envrc you have not read: it is arbitrary code that runs on cd.

SETUP

`eval "$(direnv hook bash)"`
Hook for bash (end of ~/.bashrc)

`eval "$(direnv hook zsh)"`
Hook for zsh (end of ~/.zshrc)

`direnv hook fish | source`
Hook for fish (config.fish)

`direnv version`
Print the installed version

`direnv version <x.y.z>`
Fail if older than this version

`~/.config/direnv/direnv.toml`
Global config file

`direnv help`
Show the help text

TRUST

`direnv allow [<path>]`
Trust and load an .envrc

`direnv deny [<path>]`
Revoke trust for an .envrc

`direnv edit [<path>]`
Open in $EDITOR, allow on save

`direnv prune`
Drop stale entries from the allow list

`direnv allow .env`
Trust a .env file

TRUST (cont.)

`[whitelist] prefix = ["<dir>"]`
Auto-trust a directory tree

INSPECT AND RUN

`direnv status`
Show loaded file and allow state

`direnv reload`
Trigger a reload

`direnv exec <dir> <cmd>`
Run a command under that dir's env

`direnv export json`
Print the env diff as JSON

`direnv export bash`
Print the exports for a shell

`direnv stdlib`
Print the stdlib source

ENVIRONMENT (.ENVRC)

`export <NAME>=<value>`
Set a variable

`PATH_add <dir>`
Prepend a directory to PATH

`path_add <VAR> <dir>`
Prepend to any path variable

`env_vars_required <VAR> ...`
Complain when variables are unset

`has <command>`
True if a command exists

`strict_env`
Exit on errors and unset variables

LOADING FILES

`dotenv [<file>]`
Load a .env file

`dotenv_if_exists [<file>]`
Load .env only if present

`source_up`
Inherit the nearest parent .envrc

`source_env <path>`
Load another .envrc

`source_env_if_exists <file>`
Load another file if present

`watch_file <file> ...`
Reload when these files change

LAYOUTS AND TOOLS

`layout python3`
Project virtualenv in .direnv/

`layout python <exe>`
Virtualenv from a given Python

`layout pyenv <version>`
Virtualenv from pyenv

`layout uv`
Sync and activate the uv project env

`layout node`
Add node_modules/.bin to PATH

`layout go`
Project GOPATH and bin on PATH

`use node [<version>]`
Pick Node from .nvmrc or a version

`use flake [<installable>]`
Load a Nix flake devShell

`use nix [-p <pkg>]`
Load nix-shell environment

CONFIG (DIRENV.TOML)

`load_dotenv = true`
Also load .env files

`strict_env = true`
Strict shell options for every .envrc

`hide_env_diff = true`
Hide the export diff on load

RECIPES

Python project

```
# .envrc
layout python3
dotenv_if_exists
.env.local
# then:
direnv allow
```

Layered config in a monorepo

```
# sub/.envrc
source_up
export SERVICE=api
PATH_add scripts
```

Nix flake dev shell

```
# .envrc
watch_file flake.nix flake.lock
use flake
```

Quiet, secret-safe setup

```
# .envrc
env_vars_required API_TOKEN
dotenv
# git: ignore .env and .direnv/
```