Nix Reproducible packages

CONCEPTS

> The nix command and flakes are experimental. Enable them once: experimental-features = nix-command flakes in ~/.config/nix/nix.conf.

flake.nix inputs and outputs
↓ lock
flake.lock pinned revisions
↓ evaluate
Derivation .drv build recipe
↓ build
/nix/store hash-addressed path

> An installable is flakeref#attr, e.g. nixpkgs#hello or .#myapp. A bare . means the flake in the current directory. Flakes only see files tracked by git: git add new files.

> Nothing in /nix/store is ever modified. Profiles and result links are just symlinks into it, which makes rollback cheap.

RUN AND TRY

`nix run <installable> [-- <args>]`
Run an app from a flake

`nix shell <installable>...`
Shell with packages on PATH

`nix shell <installable> -c <cmd>`
Run one command with the package

`nix run nixpkgs#<pkg>`
Run a nixpkgs package

`nix run .`
Run this flake's default app

`nix run github:<owner>/<repo>`
Run a flake from GitHub

`nix run nixpkgs/<branch>#<pkg>`
Use a specific nixpkgs branch

`nix shell --ignore-environment <installable>`
Shell with only these tools

BUILD

`nix build`
Build default package of ./flake

`nix build <installable>`
Build a given installable

`nix build .#<name>`
Build a named package of this flake

`nix build --no-link --print-out-paths`
Build and print store path only

`nix build -L`
Show full build logs

`nix log <installable>`
Show a build log

DEVELOP

`nix develop`
Shell with the dev environment

`nix develop .#<name>`
Use a named devShell

`nix develop -c <cmd>`
Run a command in that environment

`nix develop nixpkgs#<pkg>`
Build environment of a package

`nix fmt`
Format with the flake's formatter

`direnv: use flake`
Load the dev shell via nix-direnv

FLAKES

`nix flake init`
Flake from the default template

`nix flake init -t <template>`
Flake from a given template

`nix flake show [<flakeref>]`
List a flake's outputs

`nix flake check`
Evaluate and run flake checks

`nix flake update`
Update all inputs in the lock

`nix flake update <input>`
Update a single input

`nix flake lock`
Create missing lock entries

FLAKES (cont.)

`nix flake metadata`
Show inputs and revisions

Flake references

Form	Example
`Registry`	nixpkgs, nixpkgs/nixos-unstable
`GitHub`	github:NixOS/nixpkgs/<ref>
`Git`	git+https://<host>/<repo>?ref=
`Path`	./dir, path:/abs/dir
`Tarball`	https://<host>/<f>.tar.gz

PROFILE

`nix profile add <installable>`
Install into your profile

`nix profile list`
List installed packages

`nix profile remove <name>`
Uninstall a package

`nix profile upgrade --all`
Upgrade installed packages

`nix profile rollback`
Back to the previous generation

`nix profile history`
List profile generations

SEARCH AND INSPECT

`nix search nixpkgs <regex>`
Search packages by name or description

`nix eval <installable>`
Evaluate and print an attribute

`nix eval --expr '<expr>'`
Evaluate a Nix expression

`nix repl`
Interactive evaluator

`nix repl nixpkgs`
Repl with nixpkgs loaded

STORE AND CLEANUP

`nix store gc`
Delete unreachable store paths

`nix store optimise`
Hard-link identical files

`nix config show`
Show effective settings