Guix by Nix

Stealing an entire distro's bootstrap, one .drv at a time

Farid Zakaria · nix.vegas

You do not need to port a Guix package

to build it with Nix.

You need to sed it.

A .drv is not a Nix thing

It's an **ATerm** `Derive( … )` record. Guix writes the exact same format.

```
Derive(
  [(output-name, output-path, hash-algo, hash), …],
  [(input-drv-path, [output-names]), …],
  [input-src-paths, …],
  system, builder, [args, …],
  [(env-key, env-value), …]
)
```

`nix-daemon` and `guix-daemon` are both just **sandboxed builders that eat one of these and produce its outputs.**

A Guix derivation is already fully hermetic — every input, every source, every env var. There is nothing left to evaluate. So: translate, don't reimplement.

Receipts

Guix says:

```
Derive([("out","/gnu/store/280xcws0…-minimal","","")],[],[],
  "x86_64-linux","/bin/sh",["-c","echo 'Success' > $out"],
  [("PATH","/bin"),("out","/gnu/store/280xcws0…-minimal")])
```

Nix says:

```
Derive([("out","/nix/store/c6jjchlh…-minimal","","")],[],[],
  "x86_64-linux","/bin/sh",["-c","echo 'Success' > $out"],
  [("PATH","/bin"),("builder","/bin/sh"),("name","minimal"),
   ("out","/nix/store/c6jjchlh…-minimal"),
   ("system","x86_64-linux")])
```

Spot the difference. I'll wait.

The entire delta between two ecosystems

	Guix	Nix
Store prefix	`/gnu/store`	`/nix/store`
Output hashing	same algorithm	same algorithm, other store dir
Source fetcher	`builtin:download`	`fetchurl { urls = [ … ]; }`
FOD hash form	base16 + `r:sha256`	SRI + `flat/nar`

The store dir is folded into every output hash — so paths must be **recomputed**, not string-replaced. Otherwise: that's it. That's the whole impedance mismatch between two projects that have spent fifteen years pretending not to be siblings.

guix-transfer

~5,100 lines of Rust. Walks the DAG bottom-up, rewrites every `/gnu/store` reference, **blanks the output paths**, and lets `nix derivation add` recompute them. We never compute a Nix hash ourselves.

```
❯ guix-transfer /gnu/store/w9krgvil…-minimal.drv
Loading Guix derivation graph  ...  Loaded 1 derivations.
Translating bottom-up  ...
/nix/store/gkag53gj…-minimal.drv

❯ nix-store --realise /nix/store/gkag53gj…-minimal.drv
/nix/store/c6jjchlh…-minimal

❯ cat /nix/store/c6jjchlh…-minimal
Success
```

No `stdenv`. No Nix expressions. No rewriting. A Guix derivation, built by the Nix daemon, living in `/nix/store`.

And there is no bootstrap boundary

The obvious design — detect Guix's toolchain, splice in Nix's `stdenv.cc` — is wrong *and* unnecessary. **We translate the seeds too.**

- The graph bottoms out in ~80 `builtin:download` FODs
- Those include the **seed binaries**, not just source tarballs
- Seeds are statically linked — no `PT_INTERP`, no baked-in RPATH
- So they run in the Nix sandbox unchanged; prefix never mattered
- Everything above is rebuilt, and comes out `/nix/store`

```
357-byte seed → hex0 → stage0-posix → mes → tcc → gcc-mesboot
  → glibc → guile → gcc → … → hello
```

Grown organically, inside Nix, from Guix's sources. Nobody wrote a special case.

▉▉ Two things the .drv cannot tell you

Nix gives you /bin/sh. Guix gives you nothing. guix-daemon's container has no /bin at all; nixpkgs builds Nix with -Dsandbox-shell=…busybox. Guix packages assume no shell exists — that's why Guix patches shebangs so hard — so builds silently take a different path.

Nix blocks setuid/setgid. Guix's early bootstrap untars with gash-utils' *Scheme* tar, which faithfully restores the setgid bit on directories:

```
chmod "…/pc/djgpp/" 0o42775 → Operation not permitted
```

```
❯ nix-store --realise \
    --option filter-syscalls false \
    --option sandbox-paths '' …
```

Both are properties of the **daemon**, not the plan. Not expressible in a .drv.

GuixPkgs — do it for all of them

Every selected Guix package graph, translated **once per Guix commit**, and **checked into git** as plain `.nix` files. No IFD, no Guix daemon, no Guile at eval time. Pure evaluation, flake-compatible, eval caching intact.

packages (by-name/)	**53**
derivations (store/)	**2,937**
sources (sources/)	**1,645**
Guix commit	ac03c48

```
inputs.guixpkgs.url =
  "github:fzakaria/guixpkgs";

buildInputs = [
  pkgs.git          # Nixpkgs
  guixPkgs.hello    # Guix
];

# or shadow nixpkgs outright:
(guixpkgs.lib.mkOverlay {
  only = [ "hello" "tmux" ];
})
```

Then boot it

```
❯ nix run --accept-flake-config github:adeci/guix-by-nix
```

A VM where **every executable byte came from Guix**, and Nix built all of it. **GNU Shepherd as PID 1** — no systemd anywhere in the closure. eudev, dhcpcd, openresolv, serial getty, linux-libre 6.12.62-gnu. ~42 packages. The only things Nix authored are text files.

There's an **audit derivation** that classifies every store path by provenance, unpacking archives to inspect the executable payloads:

```
non-Guix ELF files ........... 0 rejected
untranslated /gnu/store refs . 0 rejected
systemd artifacts ............ 0 rejected
… 6 categories in total ...... 0 rejected
```

The number

The oldest joke in bootstrapping: **you need a JDK to build a JDK.** Everyone solves it by downloading a prebuilt blob and not talking about it. Guix climbs a 19-step ladder from C++: jikes → GNU Classpath → JamVM → ecj → IcedTea → OpenJDK 9 → … → OpenJDK 25. GuixPkgs translates that ladder.

	Nixpkgs	GuixPkgs
Closure derivations	2,758	3,556
Bootstrap-only derivations	**2**	**876**
Method	135 MB binary tarball	full source chain

2 versus 876. That gap is the entire trust story.

What this actually generalizes to

The **derivation** is the portable artifact — not the Nix expression. Relocating a build graph between stores means changing a prefix and recomputing hashes. Nothing more.

> We now treat the build graph, via the derivations, as a portable artifact that can be relocated into any realm.

That works for alpha / beta / prod as distinct **realms**: promote one derivation graph through all of them. No re-evaluation, no matching nixpkgs revision, no source access, no eval-time impurity.

Guix-to-Nix is just the most obnoxious possible demo of it.

Steal this

Code — github.com/

- fzakaria/guix-transfer
- fzakaria/guixpkgs
- adeci/guix-by-nix

Cache — guixpkgs.cachix.org

Caveats: x86_64-linux only, leaf packages only, and building from source needs trusted-user. A provocation, not a packaging strategy. Unaffiliated with either project.

Writing — fzakaria.com/2026/

- 06/08 relocatable derivations
- 06/25 guixpkgs as a nix flake
- 07/29 guix by nix
- 07/30 source-bootstrapped openjdk

Guix and Nix aren't rivals. Same machine, different paint.