Lix Beta Guide

Thank you for choosing to help us in our beta!

There is a lot of work-in-progress documentation and a lot of it is work in progress or awaiting move to the wiki. Our apologies for this state, let us know if there is something you need.

If you run into any friction, please let us know. We would like to hear all your complaints, and this beta is as much about testing our processes as it is about testing the software.

Getting yourself set up with an account (if desired)

Sign in with GitHub on https://identity.lix.systems.

Note that your email will be visible on Gerrit if you use it, so change it on https://identity.lix.systems if necessary.

A brief tour of the Lix systems

See Information Organisation for where information is.

The Lix sources are developed on Gerrit, built with Buildbot, and released on a Forgejo repo.

Contributor documentation for the project is maintained on this wiki. FIXME(jade): a lot of it is awaiting migration onto the wiki from the private pad system, see tracking issue.

Status

We are confident enough to run nightly builds on the machines we care about. We expect Lix to have, generally, fewer bugs than Nix 2.18, which is what you probably already have.

Notable changes:

- REPL is much better
  - The debugger is no longer missing variables
  - `--debugger-on-trace` gives you a debugger for `builtins.trace`
  - The `nix repl` startup messages have been shortened
- Many errors now print the value in question (`cannot coerce set to string`, `expected list but got string`, etc.)

- Many bugs have been fixed, in general:
  - `nix eval nixpkgs#hello` now gives the derivation path instead of hanging
  - `nix-env -qa` lists all attribute paths leading to a package, instead of missing some
- `nix flake check -v` prints what is being checked (and now we notice how slow that command is)
- Stack overflow is now caught properly
- Performance improvements (8-20% faster than 2.18)
- Correctness (inherit-from laziness fixed)
- `nix repl` can `:doc` library functions.
- `nix repl` can accept overlays as config files, see `repl-overlays` release note in the sources.

We have an installer, but it is not easy to use for HEAD builds. We also have a binary cache but we need to do more work to make it actually hit for building HEAD.

On NixOS/nix-darwin

Use the overlay: https://git.lix.systems/lix-project/nixos-module

Please file bugs if this explodes the build of tooling you use, we can fix it in the overlay.

Flakes

Add Lix to your system configuration like so:

```
{
 inputs = {
  lix = {
   url = "https://git.lix.systems/lix-project/lix/archive/main.tar.gz";
   flake = false;
  };

  lix-module = {
   url = "https://git.lix.systems/lix-project/nixos-module/archive/main.tar.gz";
   inputs.nixpkgs.follows = "nixpkgs";
   inputs.lix.follows = "lix";
  };
 };

 outputs = {nixpkgs, lix-module, lix, ...}: {
  # or equivalent for darwin
  nixosConfigurations.your-box = nixpkgs.lib.nixosSystem {
   system = "x86_64-linux";
   modules = [
```

```
        ./machines/your-box
        lix-module.nixosModules.default
      ];
    };
  };
}
```

You can then update it with `nix flake update lix; nix flake update lix-module`.

Not flakes

Also supported.

Add inputs for `git+https://git.lix.systems/lix-project/lix` and `git+https://git.lix.systems/lix-project/nixos-module` to your preferred pinning tool.

Use in a NixOS module: e.g. `imports = [ (import "${your-pinning-thingy.lix-nixos-module}/module.nix" { lix = your-pinning-thingy.lix; }) ];`

Niv

Add the sources for the module and Lix itself, using `ssh://` after registering your keys with git.lix.systems:

```
$ niv add git -n lix-nixos-module --repo 'https://git.lix.systems/lix-project/nixos-module'
$ niv add git -n lix-lix --repo 'https://git.lix.systems/lix-project/lix'
```

Then, import the Lix NixOS module:

```
  imports = [
    (import "${sources.lix-nixos-module}/module.nix"
      (let lix = sources.lix-lix.outPath;
      in {
        inherit lix;
        versionSuffix =
          "pre${builtins.substring 0 8 lix.lastModifiedDate}-${lix.shortRev}";
      }))
  ];
```

On other Linux or on macOS

Currently we are still working on the installer (see tracking project). It is possible to convert an existing Nix install to Lix.

flakey-profile

This is **experimental**. Some people have successfully used it on macOS. We have tested it on an Arch Linux system installed a long time ago with the shell-based installer, and it works fine. This method works by replacing your system profile with one that is built by simple Nix code with flakey-profile.

You can rollback if it blows up by `/nix/var/nix/profiles/default-{SECOND-HIGHEST-NUMBER}/bin/nix-env --rollback --profile /nix/var/nix/profiles/default`.

Clone `https://git.lix.systems/lix-project/nixos-module.git`, then, inside it, run `sudo nix run --extra-experimental-features 'nix-command flakes' .#system-profile.switch`.

Finally, run `sudo systemctl daemon-reload && sudo systemctl restart nix-daemon`, or, for macOS:

```
sudo launchctl stop system/org.nixos.nix-daemon
sudo launchctl enable system/org.nixos.nix-daemon
sudo launchctl kickstart -k system/org.nixos.nix-daemon
```

Restoring a broken install after a macOS update

After updating macOS, you may get error messages like these:

```
~/.nix-profile: no such file or directory
/nix/var/nix/profile/default: no such file or directory
error: cannot connect to socket at '/nix/var/nix/daemon-socket/socket': Connection refused
```

You can fix this by opening "Disk Utility" and manually mounting the `Nix` Volume again. Then, run these commands to re-install the lix daemon:

```
sudo launchctl load /nix/var/nix/profiles/default/Library/LaunchDaemons/org.nixos.nix-daemon.plist
sudo launchctl kickstart -k system/org.nixos.nix-daemon
```

Manually, with `nix profile`

We::Qyriad have used these steps **on macOS** has it has **seemed** to work, but we would recommend flakey-profile over it.

```
$ sudo -H --preserve-env=SSH_AUTH_SOCK nix --experimental-features 'nix-command flakes' profile install --profile /nix/var/nix/profiles/default git+ssh://git@git.lix.systems/lix-project/lix --priority 3
```

- `--preserve-env=SSH_AUTH_SOCK` assumes that your SSH agent is important to access the Lix repo
- `--priority 3` makes it symlink Lix over your existing Nix install

If you then run `sudo nix --experimental-features 'nix-command flakes' profile list --profile /nix/var/nix/profiles/default`, you should get output similar to this:

```
Index:           0
Store paths:     /nix/store/8ma7xas2nb0i3lq8mm7fpgalv94s8pzh-nss-cacert-3.92

Index:           1
Store paths:     /nix/store/53r8ay20mygy2sifn7j2p8wjqlx2kxik-nix-2.19.2

Index:           2
Flake attribute:    packages.aarch64-darwin.default
Original flake URL: git+ssh://git@git.lix.systems/lix-project/lix
Locked flake URL:   git+ssh://git@git.lix.systems/lix-
project/lix?ref=refs/heads/main&rev=98b497a1a43a4ff39263ed5259f12c5d00b4d8c0
Store paths:     /nix/store/8040hxr4rr8bpb5yp4b48709x3qs4bwb-nix-2.90.0
```

You may then use `sudo nix --experimental-features 'nix-command flakes' profile remove --profile /nix/var/nix/profiles/default 1` to remove your original installation of Nix. This is (probably) optional.

Verification

You should now get something like the following:

```
~ » nix --version
nix (Nix) 2.90.0-lixpre20240324-f86b965
```

Revision #25
Created 24 March 2024 08:05:46 by jade
Updated 2 August 2024 20:28:18 by Felix Uhl