main 5 Branches 0 Tags Go to file Go to file Code

artemist f939a32 · yesterday 123 Commits

configura…	add some more …	last week
doc/manual	Add documenta…	yesterday
lib	Make disk imag…	2 weeks ago
modules	Add documenta…	yesterday
.gitignore	fix qemu-vm	3 weeks ago
LICENSE	Permission obta…	last week
READM…	Add nix run com…	5 days ago
TODO.md	debug info is se…	yesterday
flake.lock	debug info is se…	yesterday
flake.nix	flake: Cleanup o…	3 weeks ago

About

An unofficial NixOS fork with a FreeBSD kernel

Readme
MIT license
Activity
Custom properties
251 stars
14 watching
7 forks
Report repository

Releases

No releases published

Packages

No packages published

README MIT license

NixBSD

NixBSD is an attempt to make a reproducible and declarable BSD, based on NixOS. Although theoretically much of this work could be copied to build other BSDs, all work thus far has been focused on building a FreeBSD distribution.

Languages

Nix 80.0% Shell 5.5%
Roff 5.1% Perl 3.8%
C 0.4% Lua 0.4%
CUE 0.2%

Structure

As of January 2023, NixBSD consists of 3 main repositories:

nix

A fork of upstream nix with changes to allow building on FreeBSD and for FreeBSD platforms. The changes are fairly minor and mostly around fixing bugs in the existing FreeBSD port.

It is likely that these can be upstreamed fairly quickly with some cleanup.

nixpkgs

A fork of upstream nixpkgs with support for the FreeBSD platform and a variety of FreeBSD packages.

This adds a few new supported systems, including `x86_64-freebsd`, the host system we have been using for most development. It's based off staging and has some changes to the stdenv that also affects stdenv, so you have to rebuild everything from bootstrap.

This will require a fair amount of work to clean up to an upstreamable state, but it should not require masive reorganization

NixBSD

This repository contains modules for building a system, like the `nixos` directory in nixpkgs. When possible, modules are taken directly from nixpkgs without copying (see references to `extPath` in module-list.

Some modules are copied with modification from nixpkgs. The original files remain under the MIT License and copyright the original contributors.

Unfortunately, upstreaming NixBSD into NixOS would require a fair amount of reorganization and changes to init systems, so is unlikely to happen anytime soon.

freebsd-src

You may notice that there is no fork of freebsd-src. The changes required to FreeBSD code are minimal and concern mostly the build system, so are included as patches in nixpkgs, or as calls to `sed` in package files.

Building

The easiest way to test module changes is to build a virtual machine from Linux.

You can build sample configurations (directories in configurations) easily with the flake output `.#<configName>.<outputName>`. The `base` configuration provides a simple starting point with a user account and default services.

All outputs from `system.build` are available, plus a few more. When developing you may want:

- `toplevel`: Top-level derivation, containing the kernel, etc, software, activation script, and more. You'll find it linked in the VM image at `/run/current-system` after activation.

- `vm` : A script that runs a virtual machine containing the `toplevel` , booted with UEFI. The system is booted from a writable CoW copy, so activation will run and you can edit files
- `closureInfo` : The closure-info of `toplevel.drvPath` .
- `vmClosureInfo` : the closure-info of `vm.drvPath` .

The `closureInfo` and `vmImageRunnerClosureInfo` outputs include metadata about the build closure, including a list of all packages. Keeping a copy of this around will prevent nix from garbage-collecting all of your builds.

Subtituter

There is a substituter (binary cache) in the flake. If Artemis remembers, this should contain everything in `.#base.vmImageRunnerClosureInfo` and could save you a few hours.

Note, however, that trusted substituters can maliciously modify outputs, so only use it if you trust Artemis.

Tips

- Building `vmImageRunner` for a minimal configuration can take over 8 hours on a fast machine, so keeping around `vmImageRunnerClosureInfo` is highly recommended. Just `base.vmImageRunnerClosureInfo` takes over 30GiB though, so you may want to delete it if you're low on space.
- Some package checks may fail intermittently under heavy load. If that happens you may want to build with `--max-jobs 4` or lower so fewer packages are competing for the CPU at the same time.
- To see what is happening, you might want to use nix-output-monitor. For flake commands you can replace `nix` with `nom` to use it.

Contributing

We'd be happy to review any pull requests! If you have any problems please open an issue on this repo, we're using this issue tracker for nix and nixpkgs issues as well.

Contributions should be formatted with nixfmt. While you can use `nix fmt` , that will rebuild the universe. You may want to run `nix-shell -p nixfmt --run "nixfmt ."` instead.

tldr

In your nixbsd checkout:

```
# Build the VM and all dependencies, make sure Nix doesn't delete them
# Will likely take several hours
nix build .#base.vmClosureInfo --out-link .gcroots/vm
# Build the VM (actual build happened last step, should only take a few seconds)
```

```
nix build .#base.vm
# Run a VM
result/bin/run-nixbsd-base-vm
# login as root:toor or bestie:toor
```

or to just build and try a VM to play with without a local checkout:

```
nix run 'github:nixos-bsd/nixbsd#extra.vm'
```

but see the above warning about substituter trust before accepting the requested substitutor. This will