Frequently Asked Questions

Contents

Nix

How to operate between Nix paths and strings?

See the Nix reference manual on string interpolation and operators on paths and strings

How to build reverse dependencies of a package?

```
$ nix-shell -p nixpkgs-review --run "nixpkgs-review wip"
```

How can I manage dotfiles in $HOME with Nix?

See nix-community/home-manager

What's the recommended process for building custom packages?

Please read Packaging existing software with Nix.

How to use a clone of the Nixpkgs repository to

Skip to main content

update or write new packages?

Please read Packaging existing software with Nix and the Nixpkgs contributing guide.

NixOS

How to run non-nix executables?

NixOS cannot run dynamically linked executables intended for generic Linux environments out of the box. This is because, by design, it does not have a global library path, nor does it follow the Filesystem Hierarchy Standard (FHS).

There are a few ways to resolve this mismatch in environment expectations:

- Use the version packaged in Nixpkgs, if there is one. You can search available packages at https://search.nixos.org/packages.
- Write a Nix expression for the program to package it in your own configuration.

  There are multiple approaches to this:

  - Build from source.

    Many open-source programs are highly flexible at compile time in terms of where their files go. For an introduction to this, see Packaging existing software with Nix.

  - Modify the program's ELF header to include paths to libraries using `autoPatchelfHook`.

    Do this if building from source isn't feasible.

  - Wrap the program to run in an FHS-like environment using `buildFHSEnv`.

    This is a last resort, but sometimes necessary, for example if the program downloads and runs other executables.

- Create a library path that only applies to unpackaged programs by using `nix-ld`. Add this to your `configuration.nix`:

```
programs.nix-ld.enable = true;
programs.nix-ld.libraries = with pkgs; [
  # Add any missing dynamic libraries for unpackaged programs
  # here, NOT in environment.systemPackages
];
```

Skip to main content

Then run `nixos-rebuild switch`, and log out and back in again to propagate the new environment variables. (This is only necessary when enabling `nix-ld`; changes in included libraries take effect immediately on rebuild.)

> **Note**
>
> `nix-ld` does not work for 32-bit executables on `x86_64` machines.

- Run your program in the FHS-like environment made for the Steam package using `steam-run`:

```
$ nix-shell -p steam-run --run "steam-run <command>"
```

How to build my own ISO?

See http://nixos.org/nixos/manual/index.html#sec-building-image

How do I connect to any of the machines in NixOS tests?

Apply following patch:

```
diff --git a/nixos/lib/test-driver/test-driver.pl b/nixos/lib/test-driver/test-driv
index 8ad0d67..838fbdd 100644
--- a/nixos/lib/test-driver/test-driver.pl
+++ b/nixos/lib/test-driver/test-driver.pl
@@ -34,7 +34,7 @@ foreach my $vlan (split / /, $ENV{VLANS} || "") {
     if ($pid == 0) {
         dup2(fileno($pty->slave), 0);
         dup2(fileno($stdoutW), 1);
-        exec "vde_switch -s $socket" or _exit(1);
+        exec "vde_switch -tap tap0 -s $socket" or _exit(1);
     }
     close $stdoutW;
     print $pty "version\n";
```

And then the vde_switch network should be accessible locally.

How to bootstrap NixOS inside an existing Linux

Skip to main content

installation?

There are a couple of tools:

- nix-community/nixos-anywhere
- jeaye/nixos-in-place
- elitak/nixos-infect
- cleverca22/nix-tests