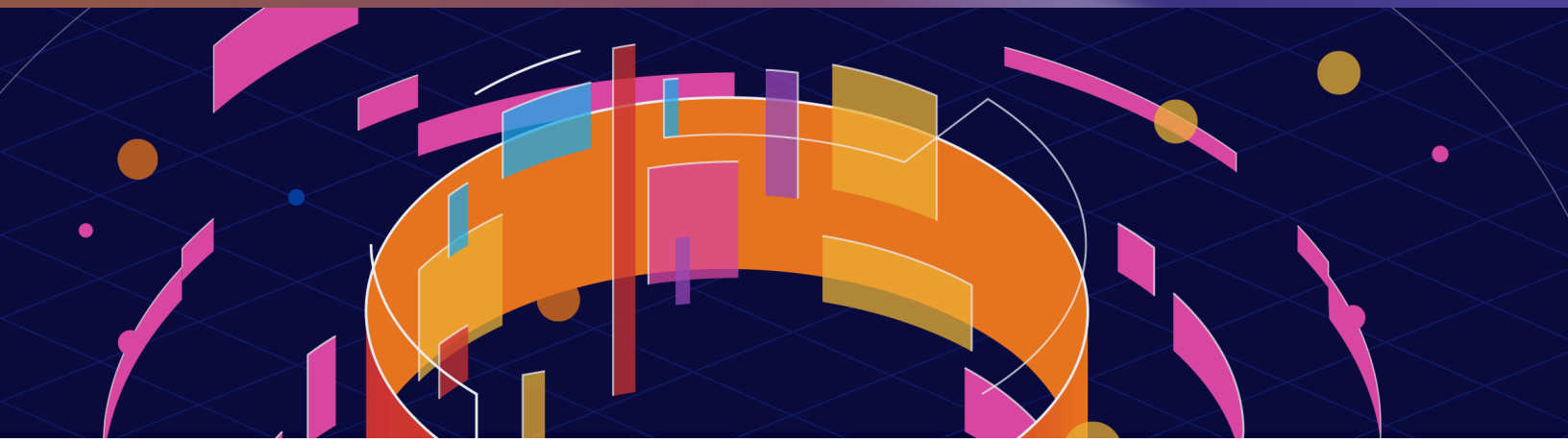

DEC 21, 2022

BY ANA HOBDEN

Nix on the Steam Deck

how-to installer

When I first started using Linux in 2006 I remember dreaming of a Linux Console. The idea maybe wasn't so far fetched at the time, the PlayStation 3 had just been released with OtherOS support which allowed users to install Linux (or BSD). Still, it seemed that a Linux-first console would only ever be a dream. Now in 2022, Valve's Steam Deck is a hackable Linux-first portable console.

Today, we'll be putting Nix on it, because what's Linux without Nix?

> *“Just wanna try it? Jump to the fun part. Want NixOS instead?*
>
> *A different, spicier kind of fun can be found here.”*

The Steam Deck is a portable computer that has a Nintendo Switch-like form factor and touts an AMD x86_64 processor. It has WiFi, Bluetooth, and a USB-C port which you can plug a hub into, allowing the attachment of HDMI, mice, keyboards, power, or ethernet cables.

It runs a flavour of Arch Linux called SteamOS, and guides users to use Flatpak and Flathub. This is a fantastic solution and provides users access to a wide variety of software, but as a developer I tend to want more exotic stuff that exists in **`nixpkgs`**.

In case you’d not seen one yet, here’s a picture of mine:

My Deck, alongside the peripherals I use with it: PS5 Controller, a USB-C hub, and an Ergodox.

Installing Nix on the Steam Deck has a few special steps. Let's review how a Nix install process looks, then how the Deck works, and finally we can explore a working approach to install Nix.

How a Nix install works

A normal Nix install process on Linux works roughly like this:

- Create a folder called **`/nix`**
- Unpack the Nix distribution tarball into **`/nix`**
- Create some Nix daemon users and a group (affecting **`/etc`**)
- Call **`systemctl link`** on some systemd units from **`/nix`** (affecting **`/etc`**)
- Sprinkle some magic in the detected shell profiles (in **`/etc`**) to ensure **`nix`** is on **`$PATH`**

On Mac, where creating a **`/nix`** is forbidden (by the creators, Apple), we can modify **`/etc/synthetic.conf`** to create a stub which we can mount an APFS volume to. The installation otherwise proceeds as normal.

On the Steam Deck, creating **`/nix`** also requires special steps. Unfortunately, there is no feature similar to **`/etc/synthetic.conf`**.

Why does the Deck need these special steps? Let's take a look at the Steam Deck itself and figure out why we can't just run the familiar Nix installer.

The Deck & SteamOS

The Steam Deck ships with an Arch Linux based distribution called SteamOS — a special image of it to be even more precise. Normally Arch Linux is a perfectly fine target for Nix, but there are a couple particularities around this distribution that impact how we can install Nix.

Disk Topology

The Deck uses an A/B boot system (like some Android phones), which means it has two parallel installations, booting into one and updating the other. This means, if it ever fails after an update it can safely roll back to a known good state.

> *"**NixOS user?** Sound familiar? It's like the generation selector in your bootloader, but instead of pointing your boot to different Nix store paths, it points to entirely different partitions!"*

```
(deck@steamdeck ~)$ sudo gdisk /dev/nvme0n1 -l
Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048          133119   64.0 MiB    EF00  esp
   2          133120          198655   32.0 MiB    0700  efi-A
   3          198656          264191   32.0 MiB    0700  efi-B
   4          264192        10749951   5.0 GiB     8304  rootfs-A
   5        10749952        21235711   5.0 GiB     8304  rootfs-B
   6        21235712        21759999   256.0 MiB   8310  var-A
   7        21760000        22284287   256.0 MiB   8310  var-B
   8        22284288      1000215175   466.3 GiB   8302  home
```

See how there is **`A`** and **`B`** copies of most partitions?

This looks a heck of a lot different than my development machine:

```
Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048         2099199   1024.0 MiB  EF00  efi
   2         2099200      3907029134   1.8 TiB     8309  encrypt
```

Checking for encryption with **`blkid | grep crypto_LUKS`** showed all partitions were unencrypted, this makes sense since the Deck never asks for a password, even for **`sudo`**, until you set one. It's a bit unfortunate Valve did not opt to protect their user's data in the event this portable device was stolen, but it's room to improve.

This A/B boot system means even if **`rootfs`** partitions get modified, those changes may get wiped out at any time. The system may update or choose to boot into the other 'letter' for some other reason. We want something that is update-proof and will survive a change of 'letter'.

One partition that persists across reboots and has enough space to contain a thick, chunky Nix store is the **`home`** partition. Our Nix install can keep persistent data there.

Read-Only Filesystem

Reviewing the **`mount`** output is a bit misleading. While the **`/`** mount says it is **`rw`**, it is normally not.

```
(deck@steamdeck ~)$ mount | grep /dev/nvme
/dev/nvme0n1p4 on / type btrfs (rw,relatime,ssd,space_cache=v2,s
/dev/nvme0n1p6 on /var type ext4 (rw,relatime)
/dev/nvme0n1p8 on /home type ext4 (rw,relatime,x-systemd.growfs)
/dev/nvme0n1p8 on /opt type ext4 (rw,relatime)
/dev/nvme0n1p8 on /root type ext4 (rw,relatime)
/dev/nvme0n1p8 on /srv type ext4 (rw,relatime)
/dev/nvme0n1p8 on /var/cache/pacman type ext4 (rw,relatime)
/dev/nvme0n1p8 on /var/lib/docker type ext4 (rw,relatime)
/dev/nvme0n1p8 on /var/lib/flatpak type ext4 (rw,relatime)
```

```
/dev/nvme0n1p8 on /var/lib/systemd/coredump type ext4 (rw,relati
/dev/nvme0n1p8 on /var/log type ext4 (rw,relatime)
/dev/nvme0n1p8 on /var/tmp type ext4 (rw,relatime)

(deck@steamdeck ~)$ sudo touch /boop
touch: cannot touch '/boop': Read-only file system
```

This isn't a scary vendor lockdown security feature or anything, it's mostly to prevent the user from being surprised when the A/B boot happens. SteamOS comes with a **`steamos-readonly`** executable we can use to toggle this read-only feature at any time, this can allow us to make small changes to the root filesystem as long as we don't expect them to persist across boots.

Because of this, if we wanted, we could create a **`/nix`** path on the **`rootfs`** each boot by making the root momentarily writable.

Not all of the device is read-only though! We can write to places like **`/etc/`**, but not to **`/lib`**, **`/usr`**, or **`/bin`**.

```
(deck@steamdeck ~)$ sudo touch /etc/boop
(deck@steamdeck ~)$ sudo rm /etc/boop
(1)(deck@steamdeck ~)$ sudo touch /lib/boop
touch: cannot touch '/lib/boop': Read-only file system
(1)(deck@steamdeck ~)$ sudo touch /bin/boop
touch: cannot touch '/bin/boop': Read-only file system
(1)(deck@steamdeck ~)$ sudo touch /usr/boop
touch: cannot touch '/usr/boop': Read-only file system
```

Recalling the rough steps from the install process, this isn't a problem! So long as we work out the machinery to ensure **`/nix`** is available, the Steam Deck looks otherwise like a normal system to Nix.

Enabling an Install

As we discovered, creating the **`/nix`** directory in a safe way that persists will be our primary challenge.

Since it wouldn't be a great idea to store the Nix Store on the **`rootfs`** partitions, we must decide somewhere else. The most immediately obvious answer is **`/home/nix`**, since that is a large, persistent location.

With an existing **`/home/nix`**, we can use a bind mount to mount that to **`/nix`**. First, a **`/nix`** path needs be created somehow!

Luckily, with **`/etc`** writable, we can drop systemd units into **`/etc/systemd/system`** that will set up **`/nix`** for us.

We'll create a **`nix-directory.service`** unit which creates the **`/nix`** path, and a **`nix.mount`** unit which depends on that.

Sadly, that's not quite enough to enable a full install though. Since the Nix install process involves **`systemctl link $UNIT`**, some of the systemd units are not available during systemd's startup. Therefore we must reload the systemd daemon itself after the **`nix.mount`** unit is started. In order to do that, we follow the same method as Flatcar Linux does here.

Let's cover what these units look like then test them out with the Nix installer! If you're feeling brave I invite you to help us test an experimental Nix installer we've been working on which has a special codepath just for the Steam Deck. Otherwise, follow along below to try the traditional install script.

But first, just in case:

- **Not sure how to get to 'Desktop mode'?** Hit the Steam button, go to 'Power', go to 'Switch to Desktop'

- **Not sure how to get a terminal?** In ‘Desktop Mode’ hit the logo in the bottom left corner, in the search bar type “Terminal”, select ‘Konsole’
- **Not sure how to edit files?** You can use `vim` if you are familiar, otherwise try `nano` from the terminal.

Putting it all together

> *“**Want to follow along without a Deck?** Learn how to set up a Deck VM with this article.”*

There are only four Steam Deck specific steps, three are to create the systemd units. The final one is to enable one of those units.

Create the systemd units at the noted paths, I suggest using a keyboard plugged into the Deck if you can, or enable SSH via `sudo systemctl start sshd`, reviewing the IP address via `ip a`, and setting a password. If those options are unavailable, hit the **Steam and X** buttons to summon the keyboard.

/etc/systemd/system/nix-directory.service

```
[Unit]
Description=Create a `/nix` directory to be used for bind r
PropagatesStopTo=nix-daemon.service
PropagatesStopTo=nix.mount
DefaultDependencies=no

[Service]
Type=oneshot
ExecStart=steamos-readonly disable
ExecStart=mkdir -vp /nix
ExecStart=chmod -v 0755 /nix
ExecStart=chown -v root /nix
ExecStart=chgrp -v root /nix
```

```
ExecStart=steamos-readonly enable
ExecStop=steamos-readonly disable
ExecStop=rmdir /nix
ExecStop=steamos-readonly enable
RemainAfterExit=true
```

The above unit is the first in our chain of units, it checks if a **/nix** folder exists, and if necessary, calls **steamos-readonly disable**, creates **/nix**, then calls **steamos-readonly enable** again. It also attempts to do some cleanup as it stops, but that part is unnecessary.

/etc/systemd/system/nix.mount

```
[Unit]
Description=Mount `/home/nix` on `/nix`
PropagatesStopTo=nix-daemon.service
PropagatesStopTo=nix-directory.service
After=nix-directory.service
Requires=nix-directory.service
ConditionPathIsDirectory=/nix
DefaultDependencies=no
RequiredBy=nix-daemon.service
RequiredBy=nix-daemon.socket

[Mount]
What=/home/nix
Where=/nix
Type=none
DirectoryMode=0755
Options=bind
```

This mount unit performs a bind mount from **/home/nix** to **/nix**. It'll create **/home/nix** for us, but sadly it cannot create **/nix**, relying on the **nix-directory.service** before it.

/etc/systemd/system/ensure-symlinked-units-resolve.service

```
[Unit]
Description=Ensure Nix related units which are symlinked r
After=nix.mount
Requires=nix-directory.service
Requires=nix.mount
DefaultDependencies=no

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/usr/bin/systemctl daemon-reload
ExecStart=/usr/bin/systemctl restart --no-block nix-daemon

[Install]
WantedBy=sysinit.target
```

This final unit in the chain restarts the systemd daemon, allowing it to properly resolve any previously broken symlinks during the boot, before starting or enabling them if necessary.

> *"**Tailscale user?** A similar strategy can be used after performing **`systemd-sysext merge`** if you happen to also use Tailscale on your Steam Deck to make sure it starts at boot."*

After creating the units, we need to enable (and start) the last, causing the ones it requires to also start:

```
sudo systemctl enable --now ensure-symlinked-units-resolve.servi
```

Now we can just run the Nix installer like normal:

```
sh <(curl -L https://nixos.org/nix/install) --daemon
```

Follow the prompts, call **`exec $SHELL`** (or open a new shell, or reboot) and Nix should work on command line!

```
(deck@steamdeck ~)$ nix-instantiate --eval -E '1 + 1'
2
(deck@steamdeck ~)$ nix-build '<nixpkgs>' -A hello
these 4 paths will be fetched (6.73 MiB download, 31.05 MiB unpa
  /nix/store/34xlpp3j3vy7ksn09zh44f1c04w77khf-libunistring-1.0
  /nix/store/4nlgxhb09sdr51nc9hdm8az5b08vzkgx-glibc-2.35-163
  /nix/store/5mh5019jigj0k14rdnjam1xwk5avn1id-libidn2-2.3.2
  /nix/store/g2m8kfw7kpgpph05v2fxcx4d5an09hl3-hello-2.12.1
copying path '/nix/store/34xlpp3j3vy7ksn09zh44f1c04w77khf-libuni
copying path '/nix/store/5mh5019jigj0k14rdnjam1xwk5avn1id-libidn
copying path '/nix/store/4nlgxhb09sdr51nc9hdm8az5b08vzkgx-glibc-
copying path '/nix/store/g2m8kfw7kpgpph05v2fxcx4d5an09hl3-hello-
/nix/store/g2m8kfw7kpgpph05v2fxcx4d5an09hl3-hello-2.12.1
(deck@steamdeck ~)$ ./result/bin/hello
Hello, world!
```

Feel free to reboot a few times, or even update your Steam Deck. As far as I've experimented, it should keep working!

If you cause a instant, hard, full power loss (such as Ctrl+C'ing the VM) before it can properly **`fsync()`**, you may see an error like **`error: expected string 'Derive('`**. To resolve this error, run **`nix store gc`**. You can avoid this by running **`sync`** before killing the device.

An invitation to experiment

Part of the reason we wanted to explore Nix on the Steam Deck is that we're currently experimenting with a new Nix installer, and we were curious what we could learn from adding support for a specific device which had special requirements, such as the Steam Deck.

If you feel like experimenting **(and don't mind things breaking)** feel encouraged to try out our prototype! Don't worry, if you don't like it, it includes an uninstaller so you can roll back and do your install with the traditional scripts.

You can run it like this:

```
curl -L https://install.determinate.systems/nix | sh -s -- insta
```

If you don't feel like being experimental, this is what it looks like:

```
(deck@steamdeck ~)$ curl -L https://install.determinate.systems/
  % Total    % Received % Xferd  Average Speed   Time    Time
                                 Dload  Upload   Total   Spent
  0     0    0     0    0     0      0      0 --:--:-- --:--:--
  0     0    0     0    0     0      0      0 --:--:-- --:--:--
100 15739  100 15739    0     0  22981      0 --:--:-- --:--:--
info: downloading installer https://install.determinate.systems/
./nix-installer install steam-deck
`nix-installer` needs to run as `root`, attempting to escalate n
Nix install plan (v0.0.0-unreleased)

Planner: steam-deck

Planner settings:

* persistence: "/home/nix"
* channels: ["nixpkgs=https://nixos.org/channels/nixpkgs-unstabl
* nix_build_user_id_base: 3000
* extra_conf: []
```

```
* modify_profile: true
* force: false
* daemon_user_count: 32
* nix_build_group_name: "nixbld"
* nix_build_user_prefix: "nixbld"
* nix_package_url: "https://releases.nixos.org/nix/nix-2.12.0/ni
* nix_build_group_id: 3000

These actions will be taken (`--explain` for more context):

* Create directory `/home/nix`
* Create or overwrite file `/etc/systemd/system/nix-directory.se
* Create or overwrite file `/etc/systemd/system/nix.mount`
* Create or overwrite file `/etc/systemd/system/ensure-symlinked
* Enable (and start) the systemd unit ensure-symlinked-units-res
* Fetch `https://releases.nixos.org/nix/nix-2.12.0/nix-2.12.0-x8
* Create build users (UID 3000-3032) and group (GID 3000)
* Create a directory tree in `/nix`
* Move the downloaded Nix into `/nix`
* Setup the default Nix profile
* Configure Nix daemon related settings with systemd
* Place the Nix configuration in `/etc/nix/nix.conf`
* Place channel configuration at `/root/.nix-channels`
* Configure the shell profiles
* Enable (and start) the systemd unit nix-daemon.socket


Proceed? (y/N): y
 INFO Step: Create directory `/home/nix`
 INFO Step: Create or overwrite file `/etc/systemd/system/nix-di
 INFO Step: Create or overwrite file `/etc/systemd/system/nix.mo
 INFO Step: Create or overwrite file `/etc/systemd/system/ensure
 INFO Step: Enable (and start) the systemd unit ensure-symlinked
 INFO Step: Provision Nix
 INFO Step: Configure Nix
 INFO Step: Enable (and start) the systemd unit nix-daemon.socke
(deck@steamdeck ~)$ . /nix/var/nix/profiles/default/etc/profile.
(deck@steamdeck ~)$ nix run nixpkgs#hello
Hello, world!
```

Hate it? Uninstall it:

```
(deck@steamdeck ~)$ /nix/nix-installer uninstall
```

Our prototype has the working name of **`nix-installer`**. It supports different installation 'planners' (such as the **`steam-deck`**), can be used as a Rust library, has fine grained logging, and can uninstall a Nix it installed.

It has no runtime dependencies (though it will try to **`sudo`** itself if you forget) or build time dependencies (other than Rust/C compilers) and should build trivially inside or outside Nix for x86_64 and aarch64, Linux (**`glibc`** or **`musl`** based) and Mac.

We are currently distributing fully reproducible and hermetic **`nix`** based **experimental** builds for all supported platforms. The installer is Open Source (LGPL) and written in entirely in Rust. (Nix is still not in Rust — sorry!)

You are welcome to explore the code here. Don't worry, we're excited to talk about it at length in a future article. Stay tuned for more!

> *"We've been working with other installer working group contributors like (alphabetical) Cole, Michael, Solène, Théophane, Travis, and others to build **`nix-installer`** and better understand what a next-generation Nix installer would look like, thank you so much for all your help, hard work, and advice."*

Conclusion

We explored how the Steam Deck takes certain measures to protect users from accidentally losing changes when the system updates and

swaps due to its A/B booting, we also explored how we can use persistent systemd units to create a `/nix` path on the Steam Deck which bind mounts to a persistent `/home/nix` directory. In order to ensure that the units linked from the `/nix` path are loaded, we also learnt we can have a unit which reloads the systemd daemon, and how this resolves the issue.

Using these techniques, we successfully installed Nix on the Steam Deck using both the traditional installer, as well as a prototype that we've been working on.

SHARE

WRITTEN BY

Ana Hobden

Ana is a hacker working in the Rust and Nix ecosystems. She's from Lək̓ʷəŋən territory in the Pacific Northwest, and holds a B.Sc. in Computer Science from the University of Victoria. She takes care of a golden retriever named Nami with her partner.

Get the latest updates

Email address Subscribe

hello@determinate.systems

+1 (641) NIX-HELP (649-4357)