Three Years of Ephemeral NixOS: My Experience Resetting Root on Every Boot

Date: 2025-02-08

Fresh OS installs are bliss. But the joy fades quickly as installing and uninstalling programs leave behind a trail of digital debris. Even configuration management and declarative systems like NixOS miss crucial bits, like the contents of `/var/lib` or stray dotfiles. This debris isn't just unsightly. It can be load-bearing, crucial to the functioning of your system, but outside of your control, and not preserved on rebuilds. Full system backups merely preserve this chaos. I wanted a clean slate, automatically, every boot.

"Erase your darlings" inspired an idea in the NixOS community: allowlisting files and directories that persist across reboots. Anything not on the list gets wiped. The simplest implementation involves mounting `/` as a tmpfs (i.e. in RAM), and then bind-mounting or symlinking the allowlisted items to a disk-backed filesystem.

This has been implemented for NixOS in the impermanence project, and has some support in Guix too. In NixOS, declaring this allowlist looks like:

```
environment.persistence."/mnt/btrfs/persistent" = {
  directories = [
    "/srv/git"
    "/var/lib/private/kea"
  ];
};
```

After three years of running this setup across my NixOS desktop, laptop, home NAS, router, and VPS, I'm sharing my experience. Below is a consolidated view of everything I persist across my machines (each has its own unique allowlist).

What survives the reboot

- `/boot`, needed for boot.
- `/etc/NetworkManager/system-connections`, WiFi connection data. I want my airport WiFi to work after a reboot, and managing this declaratively is a pain.
- `/etc/ssh/ssh_host_*_key`, SSH host keys.
- `/nix`, the Nix store, containing all my binaries and configurations.
- `/srv/git`, my private Git repositories.
- `/srv/nas`, my NAS: Big Buck Bunny in a variety of encodings.
- `/var/lib/NetworkManager/secret_key`, for stable RFC7217 addresses.
- `/var/lib/audiobookshelf`, and various other directories for stateful services.
- `/var/lib/private/kea`, my router's DHCPv4 leases.
- `/var/lib/systemd/timers`, the last time systemd's cronjobs ran.
- `/var/lib/systemd/timesync`, recent time, useful for machines without an RTC.
- `~/.cache/restic`, Restic backup cache.
- `~/.config/Signal`, Signal Desktop data.
- `~/.emacs.d/init.el`, my Emacs config, which lives outside of Nix so I can easily use it from non-Nix machines.
- `~/.emacs.d/projects`, Emacs stuff.
- `~/.gnupg/private-keys-v1.d/`, my GPG keys, used for Password Store.
- `~/.password-store`, my Password Store repo.
- `~/.rustup`, so people think I'm a pretty cool guy.
- `~/.ssh/id_ed25519_sk`, my SSH key (Yubikey handle).
- `~/Downloads`, browser downloads.
- `~/persist`, my personal files, source code, docs, etc.

What doesn't survive the reboot

Everything else, which includes things like:

- `/etc` (mostly). NixOS manages parts of `/etc` declaratively, but the rest is ephemeral.
- `/var/cache`, caches.
- `/var/lib` (most of it), ephemeral application state.
- `/var/log`, my machine doesn't persist logs between reboots. This wasn't a goal, I just haven't needed to persist logs.
- `/var/nixos`, NixOS-specific data for stable UID/GIDs.
- `~/.config` (mostly), etc. Managed dotfiles are handled by Nix. Others are ephemeral.
- `~/.mozilla`, my stateless Firefox profile (more on that in a future post).
- ~ (mostly), I explicitly use `~/persist` for important data.

(Note: `/usr` is absent on NixOS systems.)

What's worked well

- **Peace of mind:** My machines are defined by the tuple (nixpkgs commit, config commit, state). That "state" is now tiny, around 1MB, compared to the usual 20-30GB root filesystem. This eases anxiety about system drift. Bliss!
- **Fearless experimentation:** Trying out new software or configurations no longer leaves ruins in its wake. If things get hairy, a quick reboot brings me back to a clean slate.
- **Easy reproducibility:** When I encounter a bug, a reboot confirms whether it's related to persistent state or a genuine issue. This makes bug reporting and collaboration easier.
- **Trivial backups:** Backing up my machines (excluding the NAS) is quick, thanks to the minimal state.
- **Clear path to statelessness:** The allowlist highlights opportunities to move even more state into declarative configuration. Resistance is futile!

What's worked poorly

- **Inconvenience:** Persisting data requires conscious effort. It's not always obvious *what* needs saving.
- **Installation complexity:** Standard OS installs are simple: boot partition, a root partition, done! But my disk partitioning is now a bit more complicated. While disko helps, it adds to the learning curve.
- **Undefined behaviour:** Applications aren't designed for users to wipe half their state and though I haven't run into problems, I expect things would break in unexpected ways.
- **Performance problems:** I've noticed bindfs occasionally consuming lots of CPU. While some optimizations have been suggested, I suspect I've been living with slower I/O for a while now.

Is it worth it?

Declarative machine management and ephemeral state has banished my anxiety about stateful systems, a huge win. But this peace of mind comes at a cost: time. Ironically, despite aiming for less upkeep, I likely spend more time than users of conventional systems.

My aim is to simplify maintenance. While abandoning impermanence entirely (or switching to something like Fedora Silverblue) is an option, I'm taking a more gradual approach: expanding what I persist, starting with my Firefox profile. I'll share my experience with ephemeral Firefox soon.