What is Nix?

Nix is a next-generation package and system manager.

Many other package managers suffer from dependency conflict issues, and many systems built on them 'decay' over time, becoming messier, slower, and more prone to crashes over time. Nix does not suffer from these issues, because of a few unique properties:

- **Deterministic:** Building the same thing in Nix with the same configuration always results in the same output. No matter where you build it, what else is installed, or how you've configured your system. If it works in one place, it works everywhere; if it *breaks* in one place, it breaks everywhere in exactly the same way. No more "works on my system".
- **Isolated:** Every package has its own fully-declared set of dependencies, that doesn't conflict with any other package on the system. Dependencies are deduplicated where possible, but it's completely possible to have an unlimited amount of different versions of the same dependency, used by different applications - *without dependency conflicts*.
- **Customizable:** All packages can be freely modified, right from within your system configuration, without needing complex custom repository infrastructure or tooling. That includes making patches to dependencies, even for a single application, without affecting anything else on the system! Packaging your own applications is easier, too.
- **End-to-end:** 'Packages' aren't just software. Nix can build your system configuration, too, with all of the same guarantees, and it can seamlessly extend to container or even multi-system management - whether it's synchronizing the software on your desktop and your laptop, or managing a large fleet of production servers.
- **Centralized configuration:** All your configuration options - for your OS, for your services, and so on - are contained within the `configuration.nix` file (or, optionally, any files you import from it). This means you can use a single syntax for configuring everything, and Nix will take care of converting it to the right configuration format for the software you are using.
- **No runtime overhead:** Best of all, Nix can do all of the above *without needing VMs or containers*. That means that there's no runtime overhead, and it works absolutely fine in graphical environments like on a desktop or laptop, too! You *can* use VMs or containers, of course, where you need them - they're just a part of your system configuration.

These properties give you a lot of nice features:

- **Reliability:** A Nix-based system is *much* more reliable than what you could get from another Linux distribution (that isn't based on the Nix model, anyway). Mystery failures are eliminated, problems are easier to diagnose, and they can be reliably reproduced by people helping you out, even if they are running a totally different system configuration.
- **Rollbacks:** Change your kernel settings? Install some experimental drivers or software? No problem. If something breaks, you can just roll back to a previous version of your

system - even if your system doesn't boot anymore.

- **Ease of installation:** No need to resolve dependency conflicts, or mess around with `virtualenv`, or pick between two pieces of software that require different conflicting dependencies. Software *just works*. Even if it's ancient software that requires ancient dependency versions to run that are supported by nothing else, you can make it work without breaking any other software.
- **Easy synchronization:** Synchronizing software or system configurations between computers is as simple as copying over your `configuration.nix`, either manually or through something like Git or even Dropbox. Since the entire system is built from that file, you can apply it to any amount of computers without issues, even if they were running a totally different configuration before.
- **No 'cruft' or 'decay':** Because your system state is derived entirely from the configuration you give it, and not from piecemeal "change this, change that" instructions, your system doesn't accumulate 'cruft' over time like you may be used to from other distributions. A 5-year-old NixOS installation will run just as smoothly and reliably as a 2-day-old one, because effectively every configuration change is a (fast) reinstallation. The only exception is stateful data generated by applications, and this can be eliminated too if you really want.
- **Automated package testing:** Because all builds are isolated and deterministic, it's possible to do fully automated package tests, to verify that a package really works as advertised. This is done for (a subset of) the official package set, too.

The properties above are not *entirely* without tradeoffs - make sure to read the section below about the tradeoffs in the FAQ before diving into Nix and NixOS.

What is Nix (the language)?

Confusingly, the name "Nix" is not just used for the package manager, but also for the *language* that you use to write packages or system configurations. Sometimes, people call it 'nixlang' to differentiate it from the package manager, and we'll do the same in this documentation.

nixlang is a little different from what you might be used to. It's a bit like a declarative language such as JSON, but also a bit like a 'real' programming language, with support for functions and variables (sort of). An excellent step-by-step introduction to the language can be found here - it's a fairly simple language, but because it has some unusual characteristics, you should definitely give that a read.

This language is used throughout Nix, and in all of the tooling surrounding it. It's the language you use for writing package definitions, modifying your system configuration, managing multiple servers, and even for writing package tests. Because it allows creating functions and other abstractions, it can support configuration at any scale, without becoming complex to use for the simple cases.

If you're curious why Nix has its own custom language, and why it doesn't just use something that already exists, have a look in the FAQ.

Revision #2
Created 11 December 2024 18:53:47 by joepie91
Updated 11 December 2024 19:21:06 by joepie91