The high-level workflow of Nix

You can use Nix to build many different things for many different purposes. However, the basic workflow is always the same:

1. You specify some kind of Nix expression. This can be a simple expression, a whole system configuration, a Morph deployment... anything that is written in Nixlang.
2. You *evaluate* that expression, and recursively evaluate everything it references, using Nix. Wherever Nix encounters derivations, it will build them and turn them into the store path of their build result. It then returns the result of the 'top-level' expression you asked it to evaluate.
3. You *apply* the result of that expression in some way. For a NixOS configuration that means setting it as the default boot target in the bootloader, for a Morph deployment that means that Morph will send it to the remote server over SSH, for a VM build that means it gives you a link to the generated VM image, and so on.

Importantly, when using NixOS, this same workflow also applies to changing any of your system configuration, and even installing packages! You never "install a package" as a discrete action - rather, you add an item to the list of packages that should exist on your system, and rebuild the configuration using `nixos-rebuild`.

In that process, Nix will notice that a package is referenced that it doesn't have yet, so it builds or downloads it. Then `nixos-rebuild` changes the system environment to the new version of your system, where this package is part of the environment. The end result is that the package is now available for you to use.

This is often one of the things that people have the most trouble with, when learning NixOS - unlearning the idea of "installing packages" as a command that you run, and instead thinking of your system configuration as having 'versions' that do or do not have certain packages available. It's a little bit like a version control such as Git.

Isn't that a really limiting workflow?

Yes and no! It's true that this workflow makes some things a bit harder, and that it can take some time to get used to. However, NixOS can do everything that more traditional Linux distributions can do in terms of configuration, and - thanks to this workflow - can even do some things that they *can't* do, like going back to past versions of your system, or having virtual environments on the

same system that look completely different.

Because every version of the system is itself a build result from a derivation, it can be referenced and managed in the same way that a piece of software might be. Opening a shell that points at a particular environment, generating a virtual machine image out of an environment, it's all possible.

Revision #1
Created 2024-12-11 21:10:42 UTC by joepie91
Updated 2024-12-11 21:27:58 UTC by joepie91