Introduction to NixOS

Tyson Whitehead

November 18, 2020

Imperative vs Declarative

Most Linux distribution are imperative. NixOS is declarative.

imperative the user mutates an initial system through a series of state changes (e.g., add this package, remove that package, change some lines in this config file) to achieve the desired final system

declarative the user specifies the desired system up front (e.g., this set of packages, this configuration option, etc.) and the system is builds the final system from this

Imperative Pitfalls

Imperative systems are hard to maintain due to the fact that

- really want a final system, but always have to think in terms of morphing a current state to get to there
- the state changes can interact in unhappy ways, for example
  - sometimes it matters what package is installed first
  - installing and removing a package might not be the same as never installing it
  - installing an older version and upgrading might not be the same as just installing the newer version

This becomes much worse when you have thousands of computers to maintain like we do. Attempts to cope include

- only maintain a few master images that are duplicate everywhere
- bolt on a declarative like system (Ansible, Puppet, etc.)

Nix

Underlying NixOS is the Nix functional package manager

- calling something a package manager gives people the sense of a basic understanding
- this is extremely misleading for Nix and believing this makes it harder to understand

Nix is a software building and composition system

- compositions are pure (determined soley by their input declaration)
- compositions are immutable
- compositions are shared
- compositions are built on compositions

NixOS

NixOS is the result of building your entire system using Nix

- starts from basic bootstrapping libraries and tools
- components are rolled together and extended into larger and larger components
- with the final component being a complete operating system

The rabbit hole is deep and it is turtles all the way down

- any number of system declaration can coexist in the same system
- all basic bricks that can be shared between them are
- building new ones is extremely quick as only the differences needs to be done

So great that GNU forked their own guile-based free-software-only version

Confusion

The term Nix is highly overloaded and context dependent. There is

- Nix the pure, lazy, function declarative lanaguage
- Nix the software composition and building system based on Nix the language
- NixOS the operating system based on Nix the software composition and building system

Nix the language is used to express the composition declarations, which are built by Nix the composition and build system, which, when taken to their natural limit, give NixOS the operating system.

Batteries

Unless you are interested in building a system, it is only as good as it's batteries

- has the second largest number of existing packages (53,079 compared to Ubuntu's 31,180 or Fedora's 22,291)
- has the most up-to-date set of packages (80.7% are the newest compared to Ubuntu's 63.9% or Fedora's 42.7%)

https://repology.org/repositories/statistics

Tends to attract some of the best and brightest, so many of the more esoteric lanauges have special framework support

Cluster

Nix the software composition and building system is available on the SHARCNET clusters

- each user is in complete control of what is in their own environment
- each user can override any part of any component
- components are automatically shared to fullest extent possible
- supports any number of versions and builds of each package
- prior version remain accessible until explicitly released an garbage collected
- all operations are fully atomic (they succeed or doesn't changing anything)

https://docs.computecanada.ca/wiki/Using_Nix

Links

Manuals/guides for the various components

- the Nix software composition and building system
- the Nix declaration language (actually included in the above)
- the Nix packaging infrastructure written in the Nix language
- the NixOS operating system also written in the Nix lanauge

Interactive website for

- looking up packages
- looking up NixOS system options

Partition

Basic EFI/BIOS partition scheme for 16 GiB

Partition	Start	End	Size
EFI system	~0 GiB	1 GiB	~1 GiB
Linux	1 GiB	15 GiB	14 GiB
Swap	15 GiB	~16 GiB	~1 GiB

Use basic alignment amount from a BIOS grub area

Partition	Start	End	Size
BIOS boot	17 KiB	1 MiB	~1 MiB