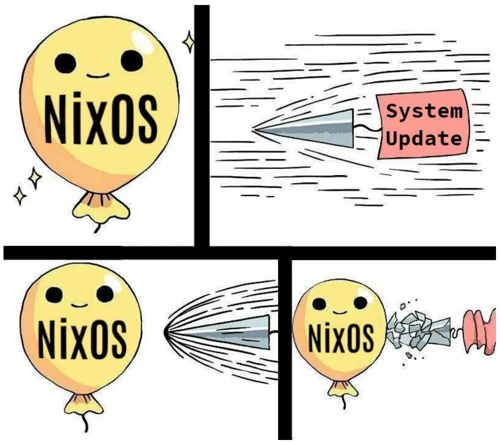
Into the Nixverse



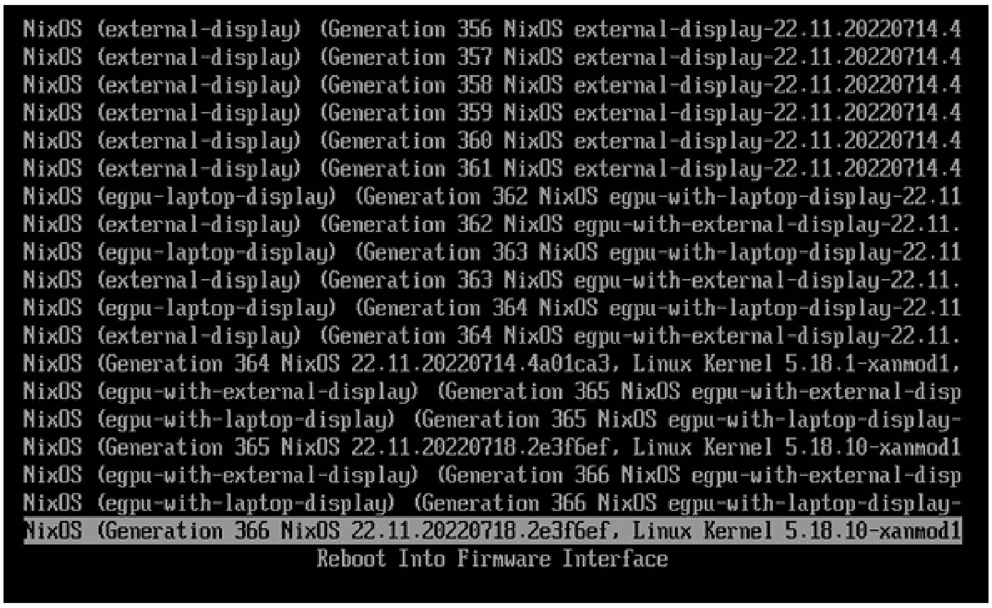
It seems like every time you think you know everything there is to know about Linux, there's always something that catches you by surprise...

NixOS and the Nix package manager is software that allows you to configure a Linux system or shell environment in a declarative and therefore reproducible manner. In other words, your system can be completely immune from configuration drift. Nix also keeps track of a program's dependencies without mutating the global state of packages, allowing for truly reproducible software builds and system configurations. In addition, NixOS allows you to rollback to a previous system state, just in case something goes horribly wrong after you update things. Combined with version control using Git, NixOS lays the groundwork for a truly resilient system configuration for everything from Infrastructure as Code deployments to personal workstations.

Here's an example NixOS system configuration file

```
# configuration.nix
# An example workstation configuration
{ config, lib, pkgs, ... }:
{
  imports = [ ./hardware-configuration.nix ./additional-config.nix ];


  boot.loader.systemd-boot.enable = true; # Enables the systemd bootloader


  networking = { # Enables networking using network manager
    hostName = "nixos";
    networkmanager.enable = true;
  };

  time.timeZone = "America/New_York";


  users.users.sales = { # Creates a new user named sales
    createHome = true;
    extraGroups = [ "networkmanager" ];
  };


  system.stateVersion = "24.11";
}
```

All it takes to rebuild this system is putting the above Nix code into `/etc/nixos/configuration.nix` and executing `nixos-rebuild switch`. Yup, that's it! No need to mess around with `apt`, `dnf` or touch any config files. The Nix package manager will evaluate your `configuration.nix` file using the attributes (variables) you defined, compare it with the Nixpkgs repository to generate a build plan, and finally pulling pre-built binary packages from cache.nixos.org. If you break your system config, no worries just select your previous system configuration from the boot menu:



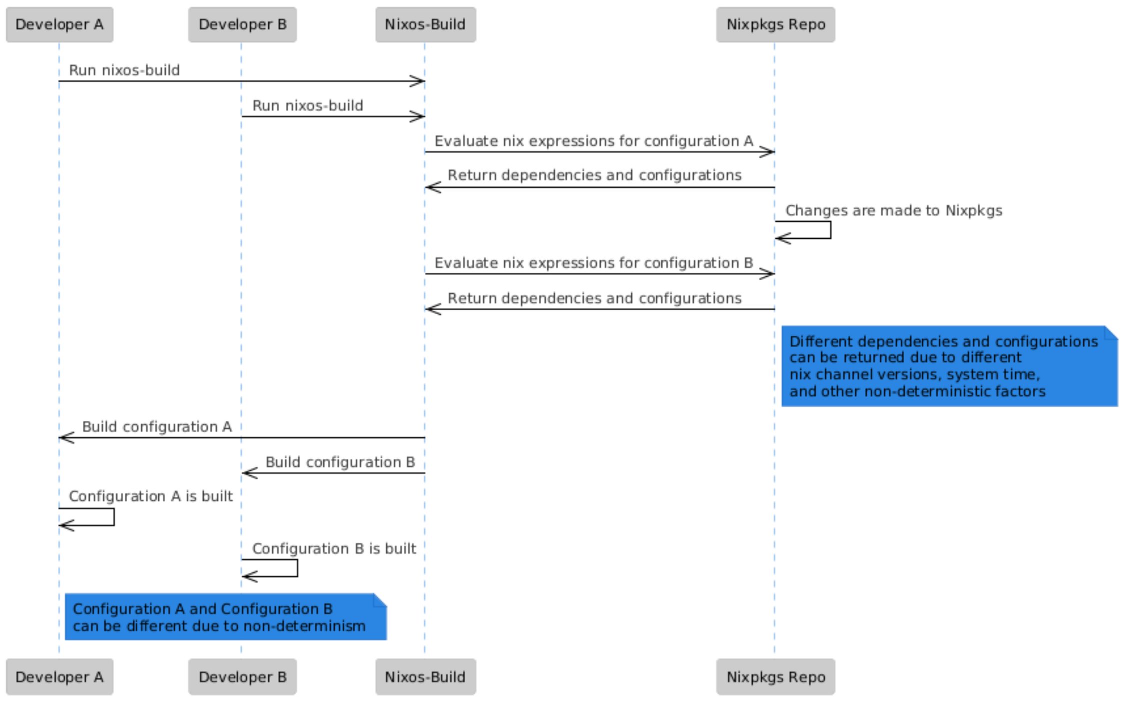
```
NixOS (external-display) (Generation 356 NixOS external-display-22.11.20220714.4
NixOS (external-display) (Generation 357 NixOS external-display-22.11.20220714.4
NixOS (external-display) (Generation 358 NixOS external-display-22.11.20220714.4
NixOS (external-display) (Generation 359 NixOS external-display-22.11.20220714.4
NixOS (external-display) (Generation 360 NixOS external-display-22.11.20220714.4
NixOS (external-display) (Generation 361 NixOS external-display-22.11.20220714.4
NixOS (egpu-laptop-display) (Generation 362 NixOS egpu-with-laptop-display-22.11
NixOS (external-display) (Generation 362 NixOS egpu-with-external-display-22.11.
NixOS (egpu-laptop-display) (Generation 363 NixOS egpu-with-laptop-display-22.11
NixOS (external-display) (Generation 363 NixOS egpu-with-external-display-22.11.
NixOS (egpu-laptop-display) (Generation 364 NixOS egpu-with-laptop-display-22.11
NixOS (external-display) (Generation 364 NixOS egpu-with-external-display-22.11.
NixOS (Generation 364 NixOS 22.11.20220714.4a01ca3, Linux Kernel 5.18.1-xanmod1,
NixOS (egpu-with-external-display) (Generation 365 NixOS egpu-with-external-disp
NixOS (egpu-with-laptop-display) (Generation 365 NixOS egpu-with-laptop-display-
NixOS (Generation 365 NixOS 22.11.20220718.2e3f6ef, Linux Kernel 5.18.10-xanmod1
NixOS (egpu-with-external-display) (Generation 366 NixOS egpu-with-external-disp
NixOS (egpu-with-laptop-display) (Generation 366 NixOS egpu-with-laptop-display-
NixOS (Generation 366 NixOS 22.11.20220718.2e3f6ef, Linux Kernel 5.18.10-xanmod1
                    Reboot Into Firmware Interface
```

All of NixOS' available options and packages can be found in https://search.nixos.org, and you can even follow along and see what each option does in the nixpkgs GitHub repository. I have also uploaded my own workstation's super fancy configuration onto Github which you can find here.

Well actually...

While using nixos-rebuild to build a system configuration can help ensure consistency, it does not guarantee consistency of a system's state across machines, as the build process is influenced by the state of the Nix package repository itself when the command was executed.

> For example, if system A builds Firefox -> changes are committed tofirefox in the Nixpkgs repository -> system B builds Firefox, the two systems are looking at two different versions of the Nixpkgs repository and will end up installing two different versions of Firefox.


Developer A
Developer B
Nixos-Build
Nixpkgs Repo
Run nixos-build
Run nixos-build
Evaluate nix expressions for configuration A
Return dependencies and configurations
Changes are made to Nixpkgs
Evaluate nix expressions for configuration B
Return dependencies and configurations
Different dependencies and configurations can be returned due to different nix channel versions, system time, and other non-deterministic factors
Build configuration A
Build configuration B
Configuration A is built
Configuration B is built
Configuration A and Configuration B can be different due to non-determinism

This problem is solved by using Nix flakes, which allows you to write Nix code whose dependencies are version-pinned in a `flake.lock` file. Flakes also allow you to define a variety of options such as the previously mentioned NixOS configuration, a development shell for a reproducible development environment, build checks, etc. (see the flake schema for options). As this is a big topic, I will have to write a separate post regarding Nix flakes another time.

So, in summary,

During my time learning and using NixOS I've honestly been having a blast. It has been a refreshing experience to work with a Linux distribution that provides for reproducibility, declarative configuration, and a strong focus on functional programming principles. I highly recommend Docker/Podman enthusiasts, DevOps engineers and experienced Linux users to give NixOS a whirl. Like a new codebase, NixOS will take some time to get used to, but your future self that's breaking or upgrading your system will thank you.

Revision #9
Created 7 September 2024 00:31:26 by GT
Updated 14 January 2025 18:01:49 by GT