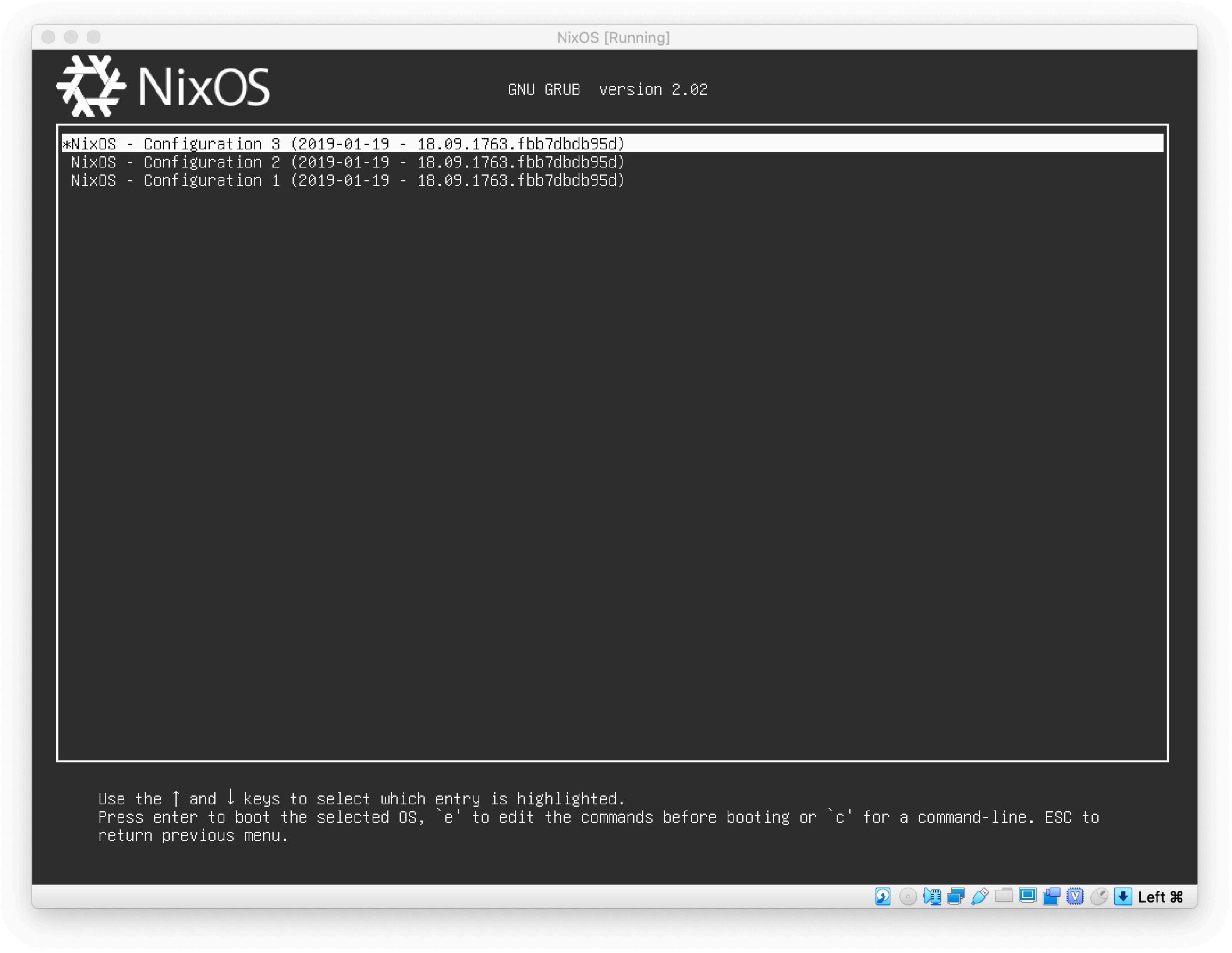
Advantages & Disadvantages of NixOS

Advantages of NixOS

- **Declarative Configuration, OS as Code**
  - NixOS uses declarative configuration to manage the entire system environment. These configurations can be managed directly with Git, allowing the system to be restored to any historical state as long as the configuration files are preserved (provided the desired states are declared in the Nix configuration).
  - Nix Flakes further enhance reproducibility by utilizing a `flake.lock` version lock file, which records the data source addresses, hash values, and other relevant information for all dependencies. This design greatly improves Nix's reproducibility and ensures consistent build results. It draws inspiration from package management designs in programming languages like Cargo and npm.
- **Highly Convenient System Customization Capability**
  - With just a few configuration changes, various components of the system can be easily replaced. Nix encapsulates all the underlying complex operations within Nix packages, providing users with a concise set of declarative parameters.
  - Modifications are safe and switching between different desktop environments (such as GNOME, KDE, i3, and sway) is straightforward, with minimal pitfalls.
- **Rollback Capability**
  - It is possible to roll back to any previous system state, and NixOS even includes all old versions in the boot options by default, ensuring the ability to easily revert changes. Consequently, Nix is regarded as one of the most stable package management approaches.
- **No Dependency Conflict Issues**
  - Each software package in Nix has a unique hash, which is incorporated into its installation path, allowing multiple versions to coexist.
- **The community is active, with a diverse range of third-party projects**
  - The official package repository, nixpkgs, has numerous contributors, and many people share their Nix configurations. Exploring the NixOS ecosystem is an exciting experience, akin to discovering a new continent.



```
GNU GRUB  version 2.02

*NixOS - Configuration 3 (2019-01-19 - 18.09.1763.fbb7dbdb95d)
 NixOS - Configuration 2 (2019-01-19 - 18.09.1763.fbb7dbdb95d)
 NixOS - Configuration 1 (2019-01-19 - 18.09.1763.fbb7dbdb95d)

    Use the ↑ and ↓ keys to select which entry is highlighted.
    Press enter to boot the selected OS, `e' to edit the commands before booting or `c' for a command-line. ESC to
    return previous menu.
```

All historical versions are listed in the boot options of NixOS. Image from NixOS Discourse - 10074

Disadvantages of NixOS

- **High Learning Curve**:
  - Achieving complete reproducibility and avoiding pitfalls associated with improper usage requires learning about Nix's entire design and managing the system declaratively, rather than blindly using commands like `nix-env -i` (similar to `apt-get install`).
- **Disorganized Documentation**:
  - Currently, Nix Flakes remains an experimental feature, and there is limited documentation specifically focused on it. Most Nix community documentation primarily covers the classic `/etc/nixos/configuration.nix`. If you want to start learning directly from Nix Flakes(`flake.nix`), you need to refer to a significant amount of outdated documentation and extract the relevant information. Additionally, some core features of Nix, such as `imports` and the Nixpkgs Module

System, lack detailed official documentation, requiring resorting to source code analysis.

- **Increased Disk Space Usage**:
  - To ensure the ability to roll back the system at any time, Nix retains all historical environments by default, resulting in increased disk space usage.
  - While this additional space usage may not be a concern on desktop computers, it can become problematic on resource-constrained cloud servers.
- **Obscure Error Messages**:
  - Due to the complex merging algorithm of the Nixpkgs module system, NixOS error messages are quite poor. In many cases, regardless of whether you add `--show-trace`, it will only tell you that there is an error in the code (the most common and confusing error message is Infinite recursion encountered), but where exactly is the error? The type system says it doesn't know, so you have to find it yourself. In my experience, **the simplest and most effective way to deal with these meaningless error messages is to use a "binary search" to gradually restore the code**.
  - This problem is probably the biggest pain point of NixOS at the moment.
- **More Complex Underlying Implementation**:
  - Nix's declarative abstraction introduces additional complexity in the underlying code compared to similar code in traditional imperative tools.
  - This complexity increases implementation difficulty and makes it more challenging to make custom modifications at the lower level. However, this burden primarily falls on Nix package maintainers, as regular users have limited exposure to the underlying complexities, reducing their burden.

Summary

Overall, I believe that NixOS is suitable for developers with a certain level of Linux usage experience and programming knowledge who desire greater control over their systems.

I do not recommend newcomers without any Linux usage experience to dive directly into NixOS, as it may lead to a frustrating journey.

> If you have more questions about NixOS, you can refer to the last chapter of this book, FAQ.

13 reactions

👍 10 🎉 1 ❤️ 1 🚀 1

0 comments *– powered by giscus*