La strada verso l'immutabilità di sistemi Linux

Fabio Alessandro "Fale" Locati

October 26, 2024

EMEA Principal Specialist Solutions Architect, Red Hat

TOC

History

How does it work?

Benefits and limits

Usecases and options

Wrapping up

Disclaimer

This is an introductory session. Further sessions

- 13:45 - Bootable Container: si installa come Linux, si gestisce come un container (U6-39)
- 13:45 - OSTree for fun and profit (U6-41)
- 13:45 - Un nuovo approccio al self-hosting: purpose-built hardware e NixOS (U6-42)

or

- 13:45 - Dal cloud al self-hosting (U6-40)

About me

- IT user since 1996.
- Working in IT since 2004.
- Fedora core developer since 2010.
- Immutable linux user since 2016.
- Fedora Sway Atomic maintainer since 2022.
- EMEA Principal Specialist Solution Architect @ Red Hat

History

History

- 1988 - POSIX supports “read-only” file systems.
- 2003 - Eelco Dolstra started Nix as a research project.
- 2006 - Armijn Hemel presented NixOS as the result of his Master’s thesis at Utrecht.
- 2013 - Docker make popular the idea of immutable containers.
- 2013 - Alex Polvi creates CoreOS.
- 2014 - Red Hat creates Project Atomic.
- 2015 - The NixOS Foundation was founded.
- 2018 - Red Hat acquires CoreOS.
- 2024 - Red Hat announces bootc.

How does it work?

Definitions

- **Container**: a lightweight, standalone, executable package that includes everything needed to run an application—code, runtime, libraries, and dependencies.
- **OCI container**: containers that adhere to a set of standards defined by the Open Container Initiative. The OCI was established in 2015 to standardize container technology to improve compatibility, portability, and interoperability across different environments.
- **Snap**: A universal package format developed by Canonical (the makers of Ubuntu) that allows applications to run in an isolated environment across different Linux distributions.
- **Flatpak**: A framework for building, distributing, and running sandboxed desktop applications on Linux.

How does immutable Linux work?

- OS filesystem is (mostly) **Read-Only**.
- OS updates are **atomic**.
- The OS filesystem can be **reverted** to previous states.
- **User environments and applications** run in isolated, layered containers.

Different kinds of immutable Linux

- NixOS
- CoreOS
- Project Atomic
- Fedora Atomic
- Bootc

NixOS Overview

- **Declarative configuration:** Entire system configuration defined in a single file (configuration.nix).
- **Reproducibility:** Ensures identical system builds across different environments.
- **Atomic upgrades & rollbacks:** Safe, atomic updates with easy rollback to previous states.
- **Isolation of dependencies:** Packages and environments are isolated to avoid conflicts.

CoreOS Overview

- **Container-optimized:** Built specifically for running containers at scale, with minimal OS services.
- **Automatic, atomic updates:** Uses `coreos-update-engine` for automatic, atomic OS updates.
- **Security-first:** Immutable file system, no package manager; reduces attack surface.
- **`etcd` for distributed configuration:** Includes `etcd` for distributed configuration management in clusters.
- **Kubernetes integration:** CoreOS soon pivoted to support large Kubernetes clusters.

Project Atomic Overview

- **Hybrid environment:** Designed to run both traditional RPM-based applications and containerized applications.
- **OSTree for updates:** Used OSTree for atomic updates and rollback.
- **Containerized workloads:** Supported running containerized apps alongside traditional apps.
- **Part of OpenShift:** Concepts from Project Atomic were later integrated into Red Hat OpenShift.
- **Transitioned to Fedora Atomic:** Ideas and technology fed into modern Fedora Atomic.

Fedora Atomic Overview

- **Multiple artifacts** sharing the same ideas:
  - Fedora Silverblue, Fedora Kinoite, Fedora Sway Atomic, Fedora Budgie Atomic
  - Fedora CoreOS
  - Fedora IoT
- **Atomic updates:** Uses OSTree for atomic system updates.
- **Flatpak for Applications:** Application installations are handled via Flatpak, ensuring isolation and easy updates.
- **Reproducible & Stable:** Provides a consistent environment for development, without configuration drift.
- **Rollback feature:** Easily revert to previous system states if updates cause issues.

Bootc Overview

- **Container:** Full operating systeim in a container image.
- **Practices & tooling:** Standard container practices and tooling.
- **Atomic updates:** The system updates atomically.
- **Rollback feature:** Easily revert to previous system states if updates cause issues.
- **State** (including per-machine configuration): Preserved across updates.
- **Factory reset:** Always possible to discard all state.
- **Cryptographic trust chain:** Cryptographically verify from the hardware, through the boot loader and OS to applications.
- **Usage:** Potential base for future Fedora (and derived) distributions.

Benefits and limits

Key benefits

- **Enhanced stability:** No unexpected changes to core system files.
- **Security:** Reduced attack surface since OS files are immutable.
- **Consistency:** Guaranteed uniformity across systems; no “configuration drift”.
- **Easy rollback:** Can easily roll back updates or changes to a previous, known-good state.
- **Simplified updates:** Atomic updates ensure the whole system updates in one operation, reducing potential for broken dependencies.

Potential limitations

- **Less flexibility:** Users can’t easily modify or customize the system core.
- **Kernel modules:** Often Kernel modules are not changeable.
- **Learning curve:** Requires knowledge of containerized environments or different package management.
- **Limited software availability:** Some traditional packages or workflows may not be supported without workarounds.
- **More complexity in application management:** Applications are often containerized, adding overhead to system setup.
- **Automation:** Often “classical” automation approaches (Ansible, Puppet, etc) break.

Usecases and options

Use cases

- **Servers and Infrastructure:** Ensures stability and easy rollback for system-critical applications.
- **Edge Devices/IoT:** Ideal for systems with limited administrative control.
- **Desktop Users:** For those seeking a stable, minimal environment with less risk of corruption or breaking.
- **Desktop Management:** Deploy, manage, and support massive amount of users easily.
- **Development Environments:** Consistent and reproducible systems for building and testing software.

Current options available

- **Fedora Silverblue:** A leading immutable desktop environment with a focus on containerized applications via Flatpak.
- **Bluefin:** Similar to Silverblue but developed by differente developers with only focus on desktop environments.
- **NixOS:** Not fully immutable by default, but Nix package manager allows declarative, reproducible system configurations.
- **Vanilla OS:** New project aiming at providing an easy-to-use immutable Linux distribution for desktop users.
- **Endless OS:** Aimed at educational environments, it uses an immutable file system to ensure stability and simplicity.
- **Fedora IoT:** Designed for IoT and embedded systems.
- **Ubuntu Core:** Designed for IoT and embedded systems with snap-based packages.

Wrapping up

Wrapping up

- Immutable Linux offers a reliable, secure, and stable operating environment at the cost of flexibility.
- It's growing in popularity with options like Fedora Silverblue, NixOS, and Fedora IoT leading the way.
- Ideal for: Developers, power users, infrastructure management, and anyone who prioritizes system stability or security over customizability.

Questions?

Email: mail@fale.io
Fediverse: @fale@fale.io

Links

- `https://projectatomic.io`
- `https://fedoraproject.org/coreos`
- `https://nixos.org`
- `https://containers.github.io/bootable`