Asura Linux

Building a modern Linux without traveling the blessed path

Document name:	Asura Linux - Building a modern Linux without traveling the blessed path		
Document author:	Gaveen Prabhasara <gaveen@asuralinux.org>		
Current version:	4.0		
Last edited date:	2023-09-08		
Version history:	Ver	Date	Remarks
	`4.0`	`2023-09-12`	Articulate the user stories
	`3.0`	`2023-09-08`	Add clickable links and improve descriptions
	`2.0`	`2023-08-16`	Articulate the details around technical expectations
	`1.0`	`2023-06-28`	Initial release declaring concept and intent

Terms Used

Term	**Intended general meaning**
Linux	An operating system using the Linux kernel.
System User / User	The types of users who are interested in building and running an operating system environments, as opposed to the end users. e.g., Linux plumbers, system administrators, developers, etc.
End User	The end users of an operating system. e.g., desktop Linux user
Open Source	Software that are under a generally accepted Open Source or Software Freedom-driven license.
Upstream	The active origin source project of a software, as opposed to a customized version kept only somewhere downstream (e.g., a distro).
Distro	A Linux distribution, which is a Linux OS with a bundled collection of software.

Table of Contents

Terms Used....1
Introduction....2
Asura Linux....3
- Project History and Name....3
- Purpose....3
- Objectives....3
- Current Status....4
- Project Promise....4
- Scope....4
- Out of Scope....5
- Challenges....6

User Stories....7

Introduction

Linux is arguably the most widely deployed operating system in the world. While Linux has historically been a Unix-like operating system, it should be fair to say a modern Linux distribution is much more than a behavior aggregation of the fifty-year-old Unix legacy.

While there is a lot to be excited about the current state of Linux distributions, if you look closely enough, you might see another new common thread emerging. Several major Linux projects are already experimenting with or considering "immutable" versions of their distributions.

These explorations are driven by the possibility of simplifying the user experience enabled by emerging technologies. While not all such projects share identical goals or approaches, there are common themes.

- Immutable / Reproducible system installations
- Atomic updates (w/ rollback) / Multi-root switching file systems
- Application channel diversity

Some examples for such Linux distributions are as follows.

- Fedora Silverblue / Kinoite / Sericea
- openSUSE Aeon / Kalpa
- NixOS
- Flatcar Container Linux
- Vanilla OS

If we consider the above themes as design goals, we can see more distros falling into the same group. For example GNOME OS (not intended for end users), Clear Linux, Endless OS, carbonOS, blendOS, Liri, LinuxKit, Talos Linux, Ubuntu Core Desktop, rlxos, astOS, AshOS, Kairos, and Universal Blue (variants) share one or more characteristics above.

Even with common desired characteristics, how each project interprets them can differ. For example,

- While Fedora Silverblue uses libostree to achieve immutability and multi-root, openSUSE Aeon uses btrfs snapshots towards the same goal. They both result in an immutable Linux OS with GNOME Desktop Environment (DE) by traveling different paths. NixOS can also arrive at roughly the same destination by being reproducible (as opposed to immutable). For some other distros, immutability might be merely an emergent property rather than a design goal.
- While Vanilla OS uses ABRoot to gain dual roots for pre and post-update states, Silverblue uses libostree to gain multiple file system roots, which can be more than two.
- NixOS uses the Nix packaging system enabled by its active user community to maintain a historical repository containing every known version of available software. On the other hand, Flatcar optimizes only running OCI containers in a server setting, whereas Vanilla OS enables the user to install software from Flatpak, multiple package management systems, and OCI containers in a desktop setting.

Fortunately, there are recent efforts to find common ground to collaborate and define specifications around shared interests. A prominent such collaboration is the Linux Userspace API (UAPI) Group. They are developing consensus among people building immutable, image-based, managed, measured-boot, secure-boot, etc. Linux systems. Unfortunately, there is still no universally accepted umbrella term to call these. In this document, we may call them image-based, immutable, or managed Linux interchangeably, interpreting them as generic terms (unless specified).

An immutable Linux system should make the user's life easier, in theory. However, the current reality is far from it. They deviate from traditional workflows but do not have sufficient support information. More importantly, customization requires significant technical skills to achieve the desired results.

Asura Linux

Asura Linux was born out of the belief there is much more to do in building modern Linux systems. We want to experiment in the immutable/image-based/managed Linux space towards building Linux systems driven by the User Experience (UX).

Project History and Name

While the project concept had been an inkling for a while, it became a concrete concept in mid-2023 with the intent to improve the user experience story of how people build and use modern Linux systems.

Asura in South Asian folklore and mythology are powerful and often chaotic beings (not necessarily malevolent or evil), usually depicted in juxtaposition against the heavenly *Sura* (divine or celestial beings).

The name Asura Linux is a tongue-in-cheek reference to the realization it would not be an easy undertaking. If the current standard practice of building Linux systems is walking a blessed path (known, traveled, and relatively safe), what we set out to do likely will not be. At least, Gaveen found it amusing for a whole minute before registering the domain name.

Purpose

Asura Linux is a project to explore the possibilities in building modern Linux systems focused on user experience enhanced by emerging technologies and concepts. These may change over time along with the rapidly evolving adjacent technology space.

At present, the key technologies and concepts of interest are as follows.

- Reproducible / Immutable Linux installations
- Multi-root capability to enable safe rollback and confident updates
- Atomic updates to avoid partial/failed update states
- Strong privacy and reasonable security by default
- Self-contained workloads
- Applications via multiple channels

Objectives

The current objectives of the Asura Linux Project is to develop a focused Linux user experience with the following characteristics.

- An always updated stable system with rollback option
- A system with long-term upgrade option
- A system for different usage needs
- A system with strong privacy and reasonable security by default
- A system with an abundant application ecosystem
- A customizable and tweakable system
- An open source project with a surrounding active, supportive, and diverse community

Maintaining an own Linux distribution is not necessarily an objective of the Asura Linux Project. It will be done in the initial 12-month period as a vehicle for experimentation. If the results can be adopted by other distros (e.g., how Fedora Asahi is being developed with direct contributions from Asahi Linux), Asura Linux might not even need to remain a direct end user distribution. Therefore, that is a decision for future.

Current Status

While we are a 100% open source project, we are still in the early stages. The concept has been developed and elaborated in this document. In the coming days, these content will be made public on GitHub.

Asura Linux is not quite ready to accept contributions since the project infrastructure is being setup and the initial build systems are being tested. In the coming weeks, we will have CI/CD in place to build bootable ISO images. We will probably use GitHub Actions service initially, but Apache BuildStream is also a long-term alternative tool being considered. When we are ready to accept contributions, everything will be made public on GitHub and the project website.

Project Promise

Asura Linux Project is still too early to set formal guidelines or governance. Therefore, instead of such, we promise you the following until we have a proper structure.

The Asura Linux Project promises are:

- Asura Linux will be open source, and it will always remain so.
- Asura Linux will be a friendly, inclusive, and welcoming community with a Code of Conduct.
- Asura Linux will collaborate with other open source projects rather than being a competitor.
- Asura Linux will value Integrity, Decency, Responsibility, Inclusion, and Excellence.

Further, we hope to write often about the process of building an image-based/managed Linux distribution. There is not too much information on how to go about such things. We believe being documented by people not used to building a shippable Linux distro, will at least be a learning experience.

Scope

In order to achieve its objectives, Asura Linux Project aims to spend the *next 12 months (October 2023 – September 2024)* building a Linux distribution with the following experimental features/areas.

This initial 12-month period is not an arbitrary definition. In 2022 and 2023, the *Image-based Linux Summit* took place in September/October. As the main public forum in the Image-based Linux space, it provides an important milestone to review, discuss, and improve/develop UAPI Group specifications. By roughly aligning to this period, the Asura Linux Project also sets a timeline to review the progress, evaluate new developments in the space, and to set the scope for next year.

As Asura is meant to be experimental, any and all of the following may drastically change over time.

In Scope Area	Rationale	Remarks on Current Direction
Developing a minimal OS core that can be updated atomically	Core should be able to be updated atomically frequently without affecting the integrity verification.	Image-based mechanism currently preferred as opposed to either libostree-based or file system (e.g., btrfs) snapshot-based mechanism.
Develop an image-based update mechanism	System updates should be available as images (as opposed to package updates). If feasible, image updates should also support delta-updates for a bandwidth efficient update experience.	Composable images with image layering to be explored. Delta-image generation is also a secondary goal.
Develop a boot process using systemd, Unified Kernel Images (UKI), and Discoverable Partitions	Unified Kernel Images (UKI) and Discoverable Disk Images (DDI) to be used to build immutable/verifiable components necessary for the bootable system.	UKI should make it possible to enable Secure Boot and TPM security. Discoverable Partitions enable automatic mounting of file systems in their correct respective mount points.

In Scope Area	Rationale	Remarks on Current Direction
Enable Secure Boot along with system integrity verification	UEFI / Secure Boot along with TPM-based security is desirable. System integrity should be verified at boot.	dm-verity for integrity verification to be implemented, which should be possible if used with Discoverable Partitions.
Handle user overrides to default system configuration	User should be able to make changes to or override the default configuration.	System Extension images in the form of DDIs to be considered.
Enable applications via Flatpak and OCI container channels	Tap into the already vibrant and burgeoning community of application delivery through self-contained bundles, namely as flatpaks and OCI-compatible containers.	While package managers can be used within containers, the use of a default package manager is not planned. Flatpak and OCI container support to be enabled by default as the primary application delivery channels.
Enable applications with hardware access	Enabling direct hardware access or hardware acceleration for applications to be explored.	Direct use of hardware (e.g., GPU, NIC) to be explored with specific use cases in consideration.

While the above outlines a high-level scope to build a solid base for Asura Linux, there are more technical challenges to be considered, especially within the context of the above and the overarching goal of developing a good user experience. The major challenges identified are listed in the "Challenges" section.

Out of Scope

The following should not be expected from Asura Linux within the *next 12 months (October 2023 – September 2024)*. The rationale of why each area is out of scope with further remarks are as follows.

Out of Scope Area	Rationale	Further Remarks
Being ready for production or general consumption	Making a general consumer-ready Linux distribution will take time.	Due to the level of experimental and partially implemented features, only those interested in the development of Asura Linux are expected to use it within this period.
Developing a new Desktop Environment	A full DE is a complex undertaking which takes a lot of time and effort. Therefore, Asura does not have a current plan to develop an own DE.	GNOME is likely to be the default DE in scenarios where a full DE is warranted. Other common DEs could be supported later.
Avoiding systemd (or developing own init system)	Despite some vocal opposition, systemd remains one of the more important pieces of infrastructure in building modern Linux systems. Therefore, Asura does not have any plans to consider an alternative init system, nor develop one.	Necessary parts of systemd will be used by default.
Avoiding Wayland	Wayland is the only actively developed and maintained display server stack for Linux. Therefore, Asura does not have a current plan to consider alternative display servers.	Wayland will be default whenever a GUI is warranted. X.org will only be considered as a legacy fallback mechanism where a GUI is warranted.

Out of Scope Area	Rationale	Further Remarks
Supporting multiple architectures	Enabling multiple hardware architecture support is also a resource intensive undertaking. Therefore, Asura does not have a plan to support all architectures supported by the Linux kernel.	x86_64 is the only primary architecture planned at the initiation. arm64 may be the second architecture to be considered, once rest of the build process is sufficiently automated. Additional architectures (including arm64 and riscv) will only be considered in future based on demand and resource availability.
Supporting older hardware or older kernels	In order to reduce the workload of the project, older hardware support and old kernel support is not planned. Therefore, Asura does not have a plan to support kernel versions before 6.1.x.	Asura will likely run newer kernel releases closer to the latest stable. Also see *Supporting multiple architectures*.
Doing everything on our own	Asura Linux will not avoid open source projects due to the *Not-Invented-Here* mindset. While we are not merely looking for *faster horses*, we are also not interested in *reinventing the wheel* whenever wheels are needed. More importantly, we believe in open source collaboration.	Asura Linux aims to learn from other open source projects. Whenever we are not sure how other open source projects are doing something, our first point of reference would be the Fedora Linux project.

Challenges

The following are some of the key challenges to be explored by the experimental nature of the immediate scope (in the next 12 months).

- An image-based approach simplifies the update/upgrade process from the current Linux user experience to something akin to a firmware upgrade of mobile phone. However, most of the users will not stop at the base system.
- They will need to install additional software, configure them according to their preference, personalize them, store their data in the system, etc. These needs do not always have convenient user experiences in current image-based Linux systems. Sometimes, the software to address the necessary UX change might not even exist.
- Users will need their system to do more than one thing or fulfill only one role. Their experimentation, entertainment, productivity, learning, and other requirements may need different software environments. The user should be able to do these tasks fearlessly.
- In addition, most existing software assume a traditional OS environment. Their complications and edge-cases with running them in an image-based OS or as a contained workload has not been well explored and usually lack support documentation.
- Ironically, the same things that are supposed to simplify UX currently make it complicated because of the gap in user experience expectation vs technological reality.
- While there is no silver bullet to change this UX gap right away, the experimentation with Asura Linux will always be user-oriented. We intend to collaborate and contribute to other open source project as well as learn from them.
- This is why Asura Linux is defined as “a project to explore the possibilities in building modern Linux systems focused on user experience enhanced by emerging technologies and concepts.”

User Stories

The following are the current user stories considered to define the scope of Asura Linux for the upcoming *12 months period (from October 2023 to September 2024)*.

These high-level user stories are categorized into sections to match the objectives of Asura Linux.

On system updates and stability

As a	I want to	So that
End User	always have an updated stable system	I can confidently do productive work
End User	receive latest security and feature updates to the system	I always have an updated system
End User	be able to confidently apply available updates (either automatically or manually)	I always have an stable system
End User	have my system always be in a verified working state (i.e., either the new working update or the previous working update)	I can always continue to do my productive work without being interrupted by failed or partial updates
End User	be able to rollback to a reasonably recent known historical state if I am unhappy with the new update	I always have a fallback
End User	have a system that does not have its performance degrade overtime	I can use a well-performing system
End User	have a system without accumulation of unwanted data due to continuous updates	my system remains clean and efficient

On long-term upgrades

As a	I want to	So that
End User	have a system that can be upgraded for a long time	continue to use my working environment without having to resort to a clean installation periodically
End User	have a mechanism to conveniently migrate my customization such as personalized preferences, configuration files, and home directory data	I can conveniently backup and restore my customization in the case of a migration with a fresh installation or a system disaster

On different usage needs

As a	I want to	So that
End User	to be able to use the same system for multiple scenarios such as home, work, and entertainment with reasonable separation	one scenario does not affect the others
End User	be able to install multiple instances of a same application software without a negative impact on the system performance or operation	I can use different instances in different manners and configurations

On strong privacy and reasonable security by default

As a	I want to	So that
End User	a system that respects my privacy by default	I do not have to be concerned about privacy violations by design
Systems User	a system that has reasonable security by modern standards (e.g., multi-factor authentication, strong encryption, mandatory or role-based access control, integrity verification, secure boot, minimum required permissions for applications, etc.)	I have relatively safe compute environment *

* High-level story needs to be expanded into multiple stories at a later stage.

On an abundant application ecosystem

As a	I want to	So that
End User	have abundance in application availability	I am able to easily find software to address my needs
Systems User	install specific versions of a software conveniently on a short-term and throw away once I am done	I can try different software and different versions without having to worry about the overall system stability
Systems User	have the option to install software from multiple channels such as native packages, flatpaks, appimages, and containers	I am enabled to use majority of software available for Linux targets *

* High-level story needs to be expanded into multiple stories at a later stage.

On customizability and tweakability

As a	I want to	So that
End User	be able to configure and personalize my system	so that my work environment is up to my preference
End User	be able to override default system configuration	so that my system is up to my preference
Systems User	be able to change what is on my system by default and how they are configured by default	so that my system is up to my preference *

* High-level story needs to be expanded into multiple stories at a later stage.