Chainguard

Chainguard Your OS

A Secure Revolution in Software Delivery

Introduction

Open source software serves as the bedrock for today's modern software development ecosystem. Tens of thousands of open source projects are continuously improved, with security and performance updates that ultimately deliver better software for developers to leverage. Despite this, too many users can't take full advantage of open source software because of systemic supply chain challenges with the traditional snapshot-the-world distribution (distro) model. These shortcomings cost the users and organizations who rely on open source time and energy, while threatening their security and ability to innovate.

Too many users can't take full advantage of open source software because of systemic supply chain challenges.

Technology advancements and widespread adoption of cloud-native software have created an opportunity to harness open source software's continuous improvements; one that combines leading automation and a distroless approach to deliver purpose-built, minimal and CVE-free software.

Chainguard has defined this new distroless paradigm – dubbed Chainguard OS – and it serves as the foundation for the broad security, efficiency, and productivity outcomes that Chainguard products deliver today.

Before diving into what Chainguard OS is, it's important to understand why now is the time for a major shift in open source software delivery.

A status quo of distros and a disruption

Traditional Linux distros like Ubuntu, Red Hat, and Debian offer major releases of open-source software, delivering tens of thousands of software packages across various computing environments (mobile, cloud, automotive, etc.). These distro releases serve as point-in-time snapshots of the tens of thousands of packages included, and are incrementally maintained with thousands of discrete updates and fixes until they reach their End of Life (EOL). This release model can either be on a consistent schedule or "when it's ready" based on the given distro, leaving downstream users of a given distro waiting for new features and patches that may be issued erratically.

Organizations and users alike rely on initial distro releases and a stream of updates to fix bugs, improve performance, increase stability, and address security vulnerabilities. However, if a given distro has reached EOL or the feature gaps and security vulnerabilities are too significant, engineering teams are faced with the technical and operational pain that comes with major version upgrades.

This pattern of perpetually patching aging systems and facing down major system upgrades presents two critical challenges:

1. **Backporting security isn't a real fix.** Over time, it becomes increasingly impractical, and even impossible to reliably backport all security fixes to a frozen "snapshot" release, leaving systems exposed and introducing inherent risk despite patching.

2. **Major upgrades slow velocity.** Technology leaders grapple with significant instability and slowed velocity when having to take on large scale distro upgrades across infrastructure environments.

As a result, engineering teams struggle with never-ending incremental distro patches and large scale upgrades that slow development and hamper developer productivity. Security teams face an onslaught of common vulnerabilities and exploits (CVEs) that expose their organizations to risk, with both groups facing down a continuous cycle of identifying, analyzing, and patching security issues.

Linux containers – and specifically, the Docker-style, OCI-compliant containers – establish both the boundaries and runtime required to solve these distribution problems. Game-changing technology advances within the Linux kernel (e.g. cgroups, namespaces) enable everything from simple programs to complicated application stacks to be extremely efficiently and securely stored, executed, and ported.

Importantly, contained applications, by way of "images", are able to bring and use their own libraries, runtime dependencies, and sidecar apps completely separate, isolated, and independent of those on the host system or any other container for that matter.

Enter Chainguard OS

Chainguard OS embraces the distroless shift towards agile, secure, and efficient software distribution model, enabling the “Chainguarding” of downstream products like container images.

CHAINGUARD OS

At its core, Chainguard OS adheres to four key principles

Continuous Integration and Delivery

Emphasizes the continuous integration, testing, and release of upstream software packages, ensuring a streamlined and efficient development pipeline through automation.

Nano Updates and Rebuilds

Favors non-stop incremental updates and rebuilds over major release upgrades, ensuring smoother transitions and minimizing disruptive changes.

Minimal, Hardened, Immutable Artifacts

Strips away unnecessary vendor bloat from software artifacts, making sidecar packages and extras optional to the user while enhancing security through hardening measures.

Delta Minimization

Keeps deviations from upstream to a minimum, incorporating extra patches only when essential and for as long as necessary until a new release is cut from upstream.

Let’s walk through each of these components and how Chainguard OS differs from the status quo.

Continuous Integration and Delivery

Packages are the atomic, fundamental artifacts of software installed today in traditional distros. Distro-packaged software offers superior convenience, security, and manageability compared to DIY methods like curl-pipe-to-bash, configure-make-install, or git-clone-install, which is why developers the world over prefer using it. But the sheer volume of unpackaged open-source software necessitates these DIY approaches, even with the risk of maliciously injected code and the added workloads of unsupported packages and updates.

Chainguard OS enables thousands of independent, or loosely dependent, open source projects, securely built into hundreds of thousands of versioned, released "packages." Unlike traditional distros, there is no major OS version (e.g., RHEL7 or Ubuntu 22.04 LTS) with a pinned catalog of packages. Every available package is always installable with Chainguard OS while also accepting a wide range of packaging ecosystems, including all of the language-specific package managers beyond Chainguard OS's package manager.

Chainguard OS leverages industry-leading, event-driven automation, utilizing virtually unlimited capacity serverless cloud resources. Every package in the catalog has a "release monitor" capturing new upstream releases within minutes, and triggering automatic package recompiles, quality assurance and acceptance testing, security scanning, and publication. Moreover, images that include that package are automatically rebuilt with updated versions and also published to public and private registries within minutes.

With Chainguard OS, users can rely on Chainguard Images that leverage an ever-growing and ever-green catalog of open source and enterprise software packages. Rather than picking a favorite version of every software project every couple of years and attempting to maintain that for a decade or more, Chainguard OS takes a different approach: it enables continuous support and EOL for the exact same versions as the upstream project maintainers recommend.

Nano Updates and Rebuilds

Platform engineers adopting Chainguard products never again have to think about wholesale, major OS version upgrades that dominate a roadmap for months at a time, every two years, or even lumpy micro OS upgrades that introduce batches of changes with regressions that are hard to bisect several times per year. Instead, Chainguard OS enables continuously introduced daily nano upgrades, paced through staging and testing gates, with offending regressions easily pinpointed, reported, and addressed in subsequent updates hours or days later.

Chainguard OS leverages the power of containers to simultaneously deliver both minor "updates" and major "upgrades", with more stability and reliability than ever before. Containers offer far more clear, firm, distinct boundaries for each application, and this separation enables cleaner updates and upgrades. Users of containers fully expect that

the application layer within the container is ephemeral, with persistent storage and unique configuration data separated from the logic within the container itself. With these changes in model and expectations, Chainguard OS can simultaneously instantiate new containers running the updated and upgraded application and destroy the previous instantiation running the down-level application version. Updates (patches) and upgrades (major changes) are introduced instantly. Rollbacks to previous versions are trivial, by simply launching the previous container's image.

Minimal, Hardened, Immutable Artifacts

Chainguard OS deviates from traditional distros by leveraging containers to run "application systems" rather than full operating systems. In this "application system," the base container includes only the minimum required for that application to run successfully with full functionality. This differs considerably from a typical "operating system" provided by a Linux distro, which includes expansive libraries of functionalities, features, and capabilities catering to a broad spectrum of creature comforts that a system administrator might one day need. These application systems are not general-purpose operating systems but are minimal and purpose-built to tight specifications.

That minimization removes software bloat that could impact performance while fundamentally and dramatically shifting the overall security of enterprise software.

Delta Minimization

By delivering nano updates that coincide with upstream code, Chainguard OS is able to minimize differences that lead to performance and security issues. These almost daily, incremental version-to-version nano updates perfectly match the upstream maintainer's expectations of user behavior.

Chainguard OS recognizes that upstream software maintainers are generally far better able to serve users with constant streams of incremental fixes and features, as well as identify regressions and problems they may introduce into their code base more quickly. By building an integration and delivery model that leverages those upstream software fixes rather than freezing it in time, Chainguard OS minimizes the potential impact on downstream users of a given package.

The Chainguard OS Advantage

Chainguard OS creates a clear advantage for everyone relying on Chainguard products, driving clear security, revenue, and efficiency benefits:

Eliminate CVEs

Chainguard OS relies on automation and tooling to aggressively triage incoming CVEs and eliminate them. It continuously and proactively rebuilds applications with updated toolchains and included dependencies, and minimizes packages with deliberately hardened images.

Achieve and Maintain Continuous Compliance

Chainguard OS relies on automation and tooling to aggressively triage incoming CVEs and eliminate them. It continuously and proactively rebuilds applications with updated toolchains and included dependencies, and minimizes packages with deliberately hardened images.

Secure the Supply Chain

Chainguard OS enables all components to be built continuously from source in a SLSA Level 2 hardened build environment. This virtually eliminates attacks to the build and distribution of open source software, and provides organizations with clear signatures and attestations for SBOMs and provenance.

Accelerate Revenue and Reduce Toil

Organizations can adopt Chainguard products leveraging Chainguard OS that rebuilds and delivers the open source software they rely on, rather than costing those organizations developer time. This approach enables engineering teams to direct critical resources to experiences and features, not patches.

Conclusion

The traditional distro status quo has minimized the advantages ever-improving open source project code bases offer to a global audience of software builders, but an inflection point of technological advancements and cloud-native development has brought new innovations to open source software delivery.

Chainguard OS takes advantage of containerization capabilities and combines world-class automation to re-think the way a distro should function and what kind of outcomes it can deliver for organizations. This new way of thinking enables more than just the “Chainguarding” of an operating system; it guards and powers minimal, purpose-built, secure container images.

Learn more about Chainguard and Chainguard Images