



Package Management and Distribution in a Cloud World

Jose Miguel Parrella




About me

- Jose Miguel Parrella
 - GitHub: [@bureado](#)
- Principal Program Manager, Office of the CTO, Microsoft Azure
- Linux and open source enthusiast for 15+ years
- Debian Developer, career distro-builder



Package management in open source: always changing

- 15 years ago: APT and RPM
- Programming languages: from CPAN and PyPI to NPM and Golang packages
- Next-generation package management: Flatpak, Snaps, Nix, etc.
- Container image specification & hub/store workflow
- Use cases where provenance is controlled by final distributor (e.g., embedded)



Ecosystem	Debian	Upstream	As a %
Ruby	1100	9300	11.83%
Perl	3700	31000	11.94%
Python	3700	118000	3.14%
Node.js	1300	350000	0.37%
All-up libs	30K	2.8M	1.07%

Source: libraries.io and APT lists

It's getting busy out there...



Why is this challenging now?

- Most IT Professionals working with Linux and open source technologies are not using modern package managers (or containers!)
 - **A significant portion of the Enterprise IT budget depends on a historical decision: APT or RPM**
- The general community sentiment (sysadmin, DevOps, SRE, etc.) on this evolving technical story is **skeptical**

Why do these technologies exist?

We don't know

- Push the packaging responsibility upstream?
- Be able to distribute non-free software more effectively and/or monetize?
- Provide additional container and application security capabilities?
- Reduce the size of a Linux distribution?
- Make it easier to package for Linux by removing dependency tracking?
- Immutable, composable and reproducible systems?
- Support cloud distribution models?
- Make the software experience easier and better for Linux desktop users?



Questions to ask


Package formats and systems	Distros
<ul style="list-style-type: none">• Atomic unit of software distribution (snap, bundle, etc.)• What the unit actually is (tarred source, squashfs, etc.)• What the unit metadata describes (dependencies, origin, checksums, etc.)• Where the unit comes from (repository equivalent)• Core repository concepts (e.g., channels, governance, login, proprietary software, etc.)• How are updates delivered?• What's the isolation/sandboxing story?• Universe (size) and type of apps• How packages are built (developer tooling)• Source vs. binary, binary caches, etc.	<ul style="list-style-type: none">• Any components of the system not managed as a unit?• Upgrade/rollback strategy (e.g., dual partitions for CoreOS)• What software is available (e.g., in bundles) and what for?• How are end users expected to bring their applications?• How system state is described (e.g., version hashes, all-up system release numbers)• Coexistence with other packaging systems• How is package provenance validated?

Analysis framework (for reference only)

Name	Atomic unit	Unit source	Universe	Isolation	Runtimes	Core value prop	Use case focus	Related	Depends
Snap	Snap (squashfs)	Stores	1,500 (*)	AppArmor, offers classic mode	Core snap	Autoupdates and bundling	Proprietary apps and IoT likely	Ubuntu Core	Systemd
Flatpak	Package (OSTree, OCI)	Flathub	290	Sandbox plus bubblewrap	Runtimes (GNOME, etc.)	Cross-distro portability	Desktop applications	OSTree Atomic AppStream	OSTree Systemd bubblewrap
Nix	Paths	Nixpkgs	6,500	None (other than the Nix store path)	N/A	Atomic upgrades and multi-versioning	Universal, declarative systems	NixOS	N/A
Guix	Paths	Hydra	7,660	None (other than the GNU store path)	N/A	Ease of use Transactions Build reproducibility Easy packaging?	Universal	GuixSD	Guile
ApplImage	ApplImage	Distributed	150+ (*) to 380	Optional via firejail	N/A	1 app = 1 file	Desktop applications	Firejail	N/A
swupd	Bundles	Repos	180	N/A	A few (e.g., perl, python)	No packages, binary deltas only	Deterministic upgrades in the cloud	Clear Linux	N/A

Standardization?

Package Format



Feature	Applimage	Snap	Flatpak
Contributors do not need to sign a CLA	✓ Yes	✗ No	✓ Yes
File format is standardized through an official standards body	✗ No (but interested in it once the format is stabilized)	✗ No	✗ No (although experimental OCI support exists)
Conceptually inspired by	macOS .app inside .dmg (tracing back to NeXT); Rox AppDir	klik? (former name of Applimage)	klik (former name of Applimage)

Key takeaways

- Do
 - Evaluate (**proof of concept**) non-mainstream distros and package managers
- Ask
 - What is our **use case** and does it match what this system/distro is trying to solve?
 - What's the **transition path** from existing APT/RPM tools, skills and investments to new systems?
- Discuss
 - How can we implement effective **governance** for future package managers? (e.g., standards, coexistence)

Resources

- Additional reading:
 - aka.ms/AA1nl2s
- E-mail
 - – jose@2063.me
- Twitter/GitHub
 - @bureado
- Slides available on [sched!](#)





THE LINUX FOUNDATION



AUTOMOTIVE
LINUX SUMMIT