Flathub

A paradigm shift for distributing applications

Jordan Petridis - LAS 2025

About Me

Hi, I am Jordan!

- GNOME developer for ~8 years
- Working part time at Centricular
- Not affiliated with Flathub
- Application and Platform/System developer
- Flatpak Runtimes, GNOME OS, CI things, Release Engineering

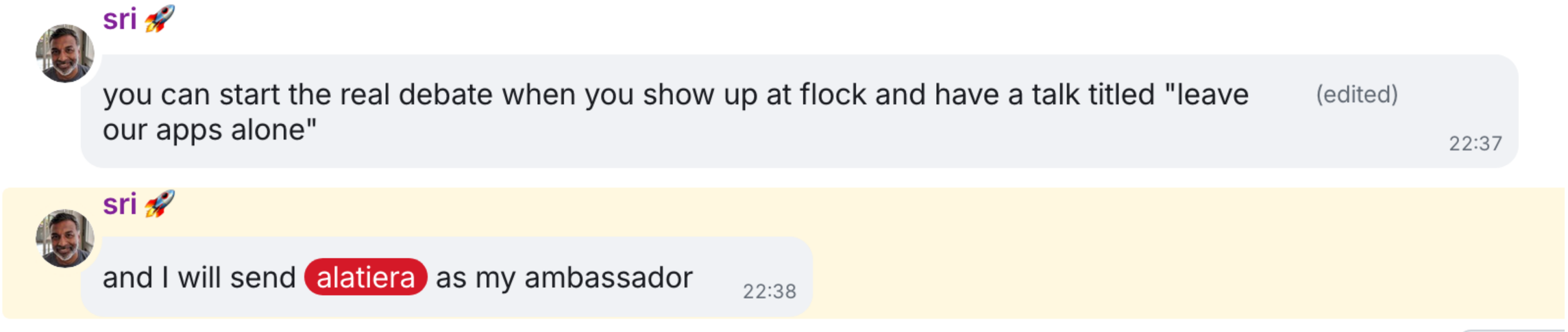
Why Am I Here?

- Second time at LAS!
- After complaining about the Conference name
- ... ended up presenting at LAS 2019
- There is no “Linux” Platform
- Counter part blogpost:
- https://blogs.gnome.org/tbernard/2019/12/04/there-is-no-linux-platform-1/


LAS
2019
ubuntu
openSUSE

Why Am I Here?

It's Sri's fault again



- Tempting..

Flathub

A paradigm shift for distributing applications

Jordan Petridis - LAS 2025

~~Flathub~~

~~A paradigm shift for distributing applications~~

Jordan Petridis - LAS 2025

Flathub Won

Stop Packaging Apps

Jordan Petridis - LAS 2025

Once upon a time

Application/Software Availability

Was a huge deal

- Common to stick with distributions based on what was packaged
- For the last couple of years this has been an issue
- Flathub has an application catalogue that my younger self could only dream of
- But how did we get here?

Traditional Distribution Model 101

Traditional Social Contract

- Clear Separation between “Upstream” and “Downstream”
- Upstream developers publish their source code
- Downstream distributions take it, build it, “test” it, integrate it
- Users download the distribution package (deb/rpm/etc)

Traditional Social Contract

- Users only ever have to trust a single source. The distribution/ISV (Independent Software Vendor)
  - Sometimes they sell support contracts to enterprises
- The distribution is responsible for vetting the software:
  - Testing the software
  - Make sure its not malicious
  - Comply with local laws, trademarks, copyright
  - Fix security issues, when not accidentally introducing them
  - Fix bugs (maybe)

Linux Distributions are not made by the people that write the software

Social Dynamics

Social Dynamics

Upstream projects are dependent on downstream distributions

- Distributions get to decide when and how users will get the software
  - Release schedules, LTS versions, different defaults, patches
- Distributions handle user support
- Access to users is often used as a power play against smaller projects
  - “Our users will never see your application unless you comply with our policies”

Social Dynamics

Testing is Impossible

- Each distribution has 3 different versions
- Every single package update needs to work with every other update
- Package managers result into a combinatorial explosion of possible environments
- Every single snapshot of the distribution is a different system
- There is no guarantee what packages and versions the end-user system will have

Social Dynamics

Testing is Impossible

- There are so so so many Linux distributions
- Each one is configured differently
  - Often they have conflicting policies
  - Some use glic, others musl
  - Some have systemd, others don’t’
  - Even the paths where they place the binaries are different
- You need a dedicated machine to build and test your software, for each one
- It’s a tremendous effort for upstreams to support multiple distributions

Social Dynamics

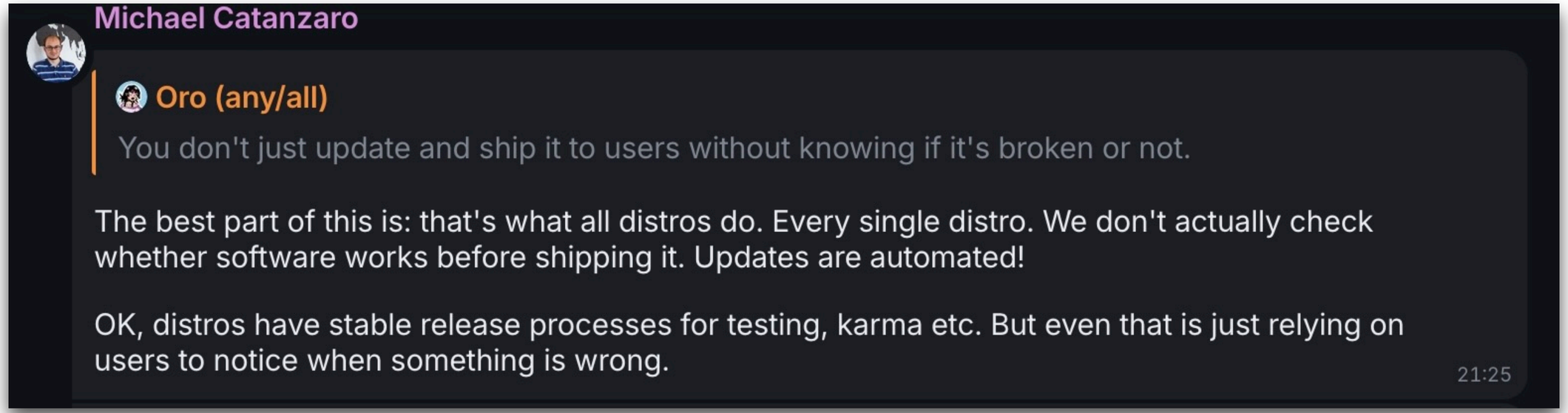
Pre-Release Testing is even more impossible

- Say you want to test your application against GNOME 49 or a newer version of GTK
- You need to have a system with the *pre-release snapshots* of these software
- Which means getting it packages and included in a distribution
  - This varies from distribution to distribution
  - Fedora Rawhide and Debian Unstable are examples
- Often theses are not up to date enough, or working at all (unstable duh)
- GNOME receives little to no feedback for alpha, beta and even Release Candidates

Social Dynamics

Vetting isn't as thorough as you'd expect

- The initial review is usually quite in depth
- ... but after that updates are automated and untested
- Nobody checks for new dependencies, or reads the release notes
- Let alone checking the actually source code diff with the previous version
- The trust we all put in on distributions is solely on vibes alone
- And besides you can't be expected to actually audit all the source, especially when you are not involved at all in the upstream development
- Even though the claim is that you only have to trust one source, you always trust the upstream developers implicitly


Michael Catanzaro
Oro (any/all)
You don't just update and ship it to users without knowing if it's broken or not.
The best part of this is: that's what all distros do. Every single distro. We don't actually check whether software works before shipping it. Updates are automated!
OK, distros have stable release processes for testing, karma etc. But even that is just relying on users to notice when something is wrong.
21:25

Jiří Eischmann
@sesivany@vivaldi.net

@okias @barthalion I maintain applications both in Flathub and Fedora repositories and I actually find it more demanding to keep the apps in Flathub than in the Fedora repos. Actually ever since I got the Fedora packages through the review process to the repos, no one has cared what I'm doing with them.
Meanwhile in Flathub I have already received several requests to remove problematic things in the manifest, update it to the latest guidelines etc. And I was hard stopped by the build system. So unless I made the changes or filed a ticket and explained why I needed it, I could not maintain the app any more.

Feb 19, 2025, 12:02 PM · · Web

6 boosts · **33** favorites

Social Dynamics

This model only does not work for third-party software

- Everything must be included in the central repository means:
- Licensing and redistribution clauses make it impossible to have proprietary software
- As well as FOSS applications including Secrets (such as API Tokens) into their builds
  - Even though all the code being Free Software
  - Example: OBS builds don’t include API keys needed to interface with the streaming platforms and the users have to supply their own
- Maintaining Third-party packages and repositories is a Sisyphean task
  - Regardless if its APT or Copr repositories, they always break in incredible ways
  - The model was never designed for third-party addons/repositories

Developer Dilemmas

- How do you deal with a million ever changing and conflicting policies of each distribution?
- How do you keep your software up to date for everyone and every possible combination
- What do you do when if you cannot give your secrets/keys to everyone

Results

- Software is distributed basically untested
- And thus distributions are extremely conservative with updating software
- There's no real "support" or "bugfixing" in practice
  - Instead its upstream project doing this, for security fixes as well
- Users get extremely old software that's not updated unless there is a "good enough" reason (security)
- Upstreams receive bug reports for bugs fixed years ago, since LTS distributions still use an outdated and known broken versions
- New features only make it to users only months, if not years, after they are released

Impact on Platform Development

Why do we even bother honestly?

- Platform developers suffer from this but don't have any other options
- Platform components (GNOME, KDE, systemd, etc.) have grudgingly adapted to this since we have been doing it this way for the past 20 years
- Others went all in into containers, for better or for worse
- Anyone who questions the existing model gets flamed endlessly

Impact on Application Development

- Developers have no “Linux” Platform/Target to develop against
- Linux is not a platform anyway..

Platform

Developers	Designers	End Users
Developer OS	Design Tooling	Consumer OS
SDK	Design Language	App Store
Documentation	HIG	User Support
App Store		

Impact on Application Development

- Developers have no “Linux” Platform/Target to develop against
- At best some developers will target a couple of the most popular distributions
  - And from those, usually only LTS releases
- More commonly, most give up and go to do literary anything else
  - “Linux” Application development was always a joke, never got commercial support
  - Attempts were made in the past, but none succeeded.

Impact on Application Development

- Waiting months for your software to make it to users
- In order to support many different distributions, they have to wait until features they need are available in the oldest still-supported LTS version
  - For example, Electron only depends on versions of libraries that can be found in Ubuntu 22.04 (LTS -1)
  - That's already 3 years old sofware
  - 3 years of development you can't make use of
  - APIs we releases last month with GNOME 48 last month, won't be available until 26.04
  - And more commonly, software will only start to depend on them around 28.04 (2028)

Traditional Social Contract

- Users only ever have to trust a single source. The distribution/ISV (Independent Software Vendor)
  - Sometimes they sell support contracts to enterprises
- The distribution is responsible for vetting the software:
  - Testing the software
  - Make sure its not malicious
  - Comply with local laws, trademarks, copyright
  - Fix security issues, when not accidentally introducing them
  - Fix bugs (maybe)

Traditional Social Contract

- Users only ever have to trust a single source. The distribution/ISV (Independent Software Vendor)
  - Sometimes they sell support contracts to enterprises
- The distribution is responsible for vetting the software:
  - ~~Testing the software~~
  - ~~Make sure its not malicious~~ (Remember the XZ backdoor?)
  - Comply with local laws, trademarks, copyright
  - ~~Fix security issues~~, when not accidentally introducing them
  - Fix bugs (~~maybe~~) (If you have deep pockets)

Birth of Flatpak

Flatpak

- Application distribution framework that aims to improve over the traditional package manager solutions.
- Not the first but nor the last approach to the problem. But it’s the one that stuck.
- Unlike other container solutions (docker/etc) , it’s an explicit goal to “Integrate” with the host system.
- Flatpak applications work regardless of the distribution they are run on

Flatpak's Design

What kind of magic can make applications work anywhere?

- Image Based
  - Applications are the sum of all the parts of the software
  - Everyone runs exactly the same version of the software
  - Atomic operations/updates
- Well defined, deterministic and reproducible environments
- Runtimes!
  - Tiny "distro" we distribute along with the application
  - Applications can run on any distro, cause they always run against their own distro
- Host-Integration (Portals)

Flatpak is not a silver bullet

Flatpak is not a silver bullet

- You can replicate today's status quo but with all of Flatpak's advantages
- You can build a Runtimes and applications using the same distribution packages
- Following the same inclusion policies and requirements
- Using the same tooling as the distro
- Maintained by the same group of people
- The Fedora Flatpak repository is such an example

Nobody wants Fedora's Flatpaks

It would have been easy to stop there

- Here's a new technology that fixes SOME of the major issues we had
- It allows for decoupling the Host System from applications
  - And thus you can update applications independently
- While also providing Sandboxing
- Image based deployments and deterministic/defined environments
- Would be making everything better than it was the day before

Flatpak developers marched forward

Traditional Social Contract

- Clear Separation between “Upstream” and “Downstream”
- Upstream developers publish their source code
- Downstream distributions take it, build it, “test” it, integrate it
- Users download the distribution package (deb/rpm/etc)

What if instead

We put developers in charge of application distribution?

Application Social Contract

- Application developers publish their source code (or not)
- Application developers build the software exactly the way they expect
  - And can define exactly the dependencies they need
  - Patch them at will, Configure them exactly as needed, and so on
- Application developers actually QA/test their application
- Application developers distribute the application to their users directly

The Birth of the First Runtimes

- This new social contract doesn't have to apply only to application developers
- Besides, we've seen how well it went relying on others to configure your software

- The first Runtimes created were not "Ubuntu" or "Fedora" or "Arch"
- But rather they were GNOME's and KDE's and they were built from scratch
- Putting the Platform developers in charge of building and distributing their runtimes

Platform Development

- Application developers don’t make “Ubuntu”, “Fedora” or “Linux” apps
- Instead they all make “GNOME” and “KDE” apps
- Development does not happen on the distribution layer but on the desktop
- Runtimes gave us the missing SDK and Target to let people develop against
- They could also be developed shaped, updated and released on the same schedule as the rest of the Platform was

We created Flathub

- Joint effort between GNOME and KDE
- Direct publishing to users as developers see fit
- Extensive automated and human reviews
- Strong focus on improving the software upstream

Flatpak as a technology makes it possible for Flathub to come along and address the underlying social issue

Flathub Won

Flathub Won

- The technical advantages of Flatpak along with direct publishing to Flathub became very popular
- The “Verified Developer” program of Flathub is very successful
  - More than half of the applications are verified
- Even though this has been a pain point and discussed for decades neither KDE or GNOME had the resources to implement and maintain a whole OS
  - Thanks to containerization technologies only need to maintain Runtime and Sdk for the applications

Conclusion

Conclusion

- Nobody has the resources to package all the apps centrally
- Distros don’t do any kind of review past the initial one
- Distros don’t triage issues or offer any kind of support in practice
- Distros publish untested builds
- Yet they still act as if they are developing the software and should be in a position to gatekeep the applications

Stop packaging apps

Stop packaging apps

- It might feel like admitting defeat, but by now it's clear the distribution model is not working
- We need to seriously rethink the role of distributions
- The status quo sort of works for server and enterprise use cases, which is where most of the money is
- If we want software freedom to become accessible this needs to change

100 better ways to spend your time

- Don’t repackage apps needlessly
- Focus on a minimal host system, and actually differentiate
- Help with Application reviews on Flathub
- Improve apps and platforms upstream

Thanks