THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA

Embedded Linux Conference

EOL, Relicensing, Forks: A Cautionary Tale of CVEs

Lachlan Evenson
& Bridget Kromhout

Microsoft

Who are you two?

Lachlan Evenson

Product Manager @ Microsoft Azure; Kubernetes Steering Emeritus; CNCF GB

Originally from Australia; now lives in San Francisco. Likes hiking, baseball, cricket and language.

GitHub & socials: lachie83

Bridget Kromhout

Product Manager @ Microsoft Azure; SIG Cloud Provider co-chair; CNCF GB alternate

Originally from Saint Paul; now lives in Minneapolis. Likes gardening, bicycles, snow, and cats.

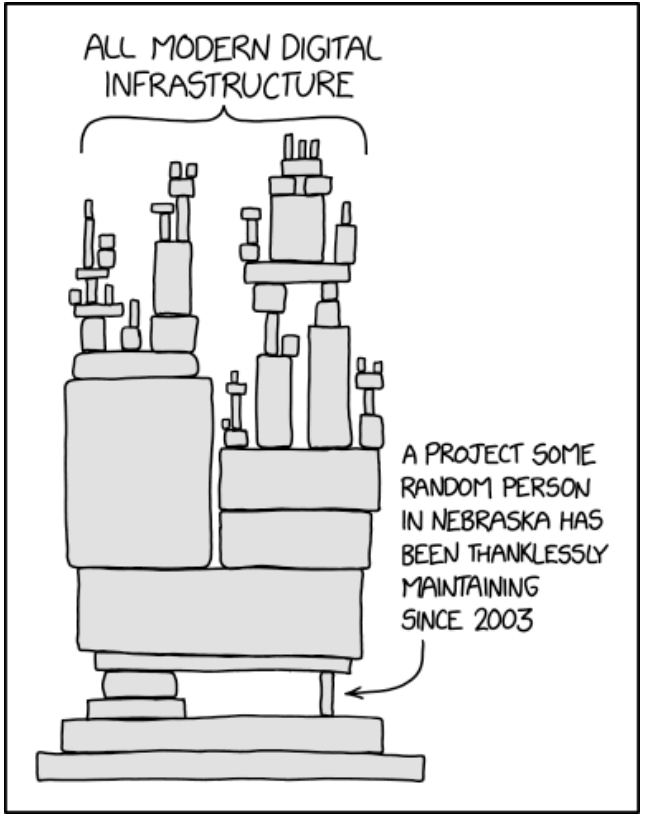
GitHub & socials: bridgetkromhout

Risks (non-exhaustive)

"Project X goes EOL at the end of this sentence; good luck with the CVEs."

"Project Y has a new license meaning you can't use it anymore; what do you mean, your team built something important on it?"

"Project Z works great but you built a new feature in your fork, and now you can't take the upstream patches."



https://xkcd.com/2347/

Risk at (hyper) scale: asking (a lot of) questions

Power:
How are decisions made about the project?

Process:
How does the project actually function?

People:
How do we interact for this project?

Power: How are decisions made about the project?

- What sort of project governance is in place & what can you tell about project health? https://insights.linuxfoundation.org/
- What's on the published contributor ladder & what other indicators of project maturity do you see?
- What is your path to contributing & gaining influence if you start participating?
- What's published for the roadmap & planned features?
- Can you chop wood & carry water in this project?

Process: How does the project actually function?

- What is the release process & who has access to cut a release?
- What is the release cadence & where are the artifacts stored?
- Where do patches come from & how are vulnerabilities handled? (EU CRA, like winter, is coming.)
- What sort of operational maturity & de-risking is happening, perhaps via docs?
- What is your plan for business continuity if upstream changes?

People: How do we interact for this project?

- What are the forums for community interaction (community calls, persistent chats, discussion forums, issue trackers...)?
- Are there opportunities for change (such as bootstrapping governance or becoming maintainers on projects lacking investment)?
- Does it have active maintainers from multiple organizations? https://devstats.cncf.io/
- Can you future-proof (as much as possible) via early discussions?

Risk: EOL & CVEs

"Project X goes EOL at the end of this sentence; good luck with the CVEs."

Risks develop over time (but time is not always on our side)

- ingress-nginx urgently needed replacement while Gateway API was still in progress; we're working on https://github.com/kubernetes-sigs/ingress2gateway
- Open Service Mesh was archived with a path to replacement with Istio

Upstreams may not have a release available. We've open sourced tools to help.

- Copa (https://github.com/project-copacetic/copacetic) lets you directly patch container images without full rebuilds (optionally informed by vulnerability scanning results)
- Dalec (https://github.com/project-dalec/dalec) lets you write YAML to produce packages and containers across multiple Linux distributions and Windows with minimal image size, resulting in fewer vulnerabilities and smaller attack surface

Risk: License changes

"Project Y has a new license meaning you can't use it anymore; what do you mean, your team built something important on it?"

- Newly added conditions may change the entities that are authorized to use a software package, or may change the scale at which you may use it.
- Changes in release artifacts or tagging may not change the license, but still may affect your ability to use the software.
- Look to see if software has an OSI Approved License: https://opensource.org/licenses

Risk: Forks

"Project Z works great but you built a new feature in your fork, and now you can't take the upstream patches."

- If you fork, what is your continuity plan? What do you lose in a hard fork?
- If you can't upstream your changes, can you soft-fork, or make upstream modular and extensible?
- Cluster Autoscaler went 1.0 for Kubernetes 1.8 (back in 2017!) and Azure's fork diverged for a time (but has since been reconciled, after a concerted effort).
- At one point, an internal Azure team was looking at forking etcd; instead, they became maintainers of upstream etcd.

What about AI?

- Low cost to fork, but more value to collaborate
- Code provenance disclosure & attribution for responsible AI
- Community trust management (like https://github.com/mitchellh/vouch)
- Testing & remediation at scale
- Broaden pool of potential contributors to de-risk maintainer churn
- Collaborate in CNCF toolbox on load-bearing open source outside the current zeitgeist

tl;dr: Community involvement is key

Find your people and join them in open source!