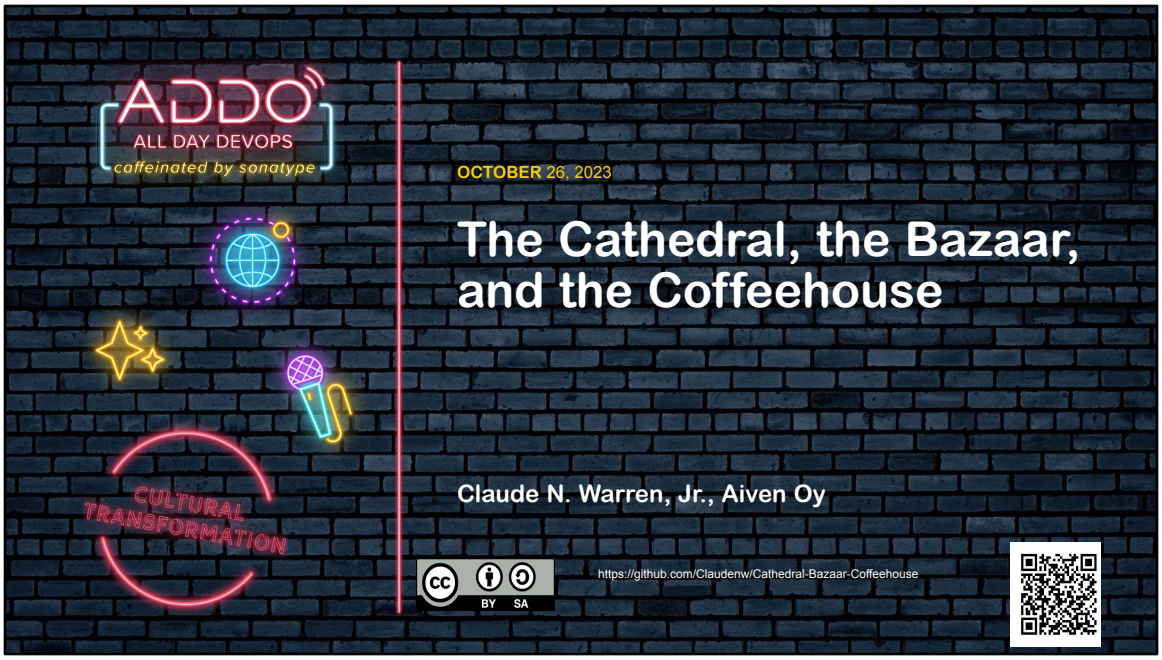
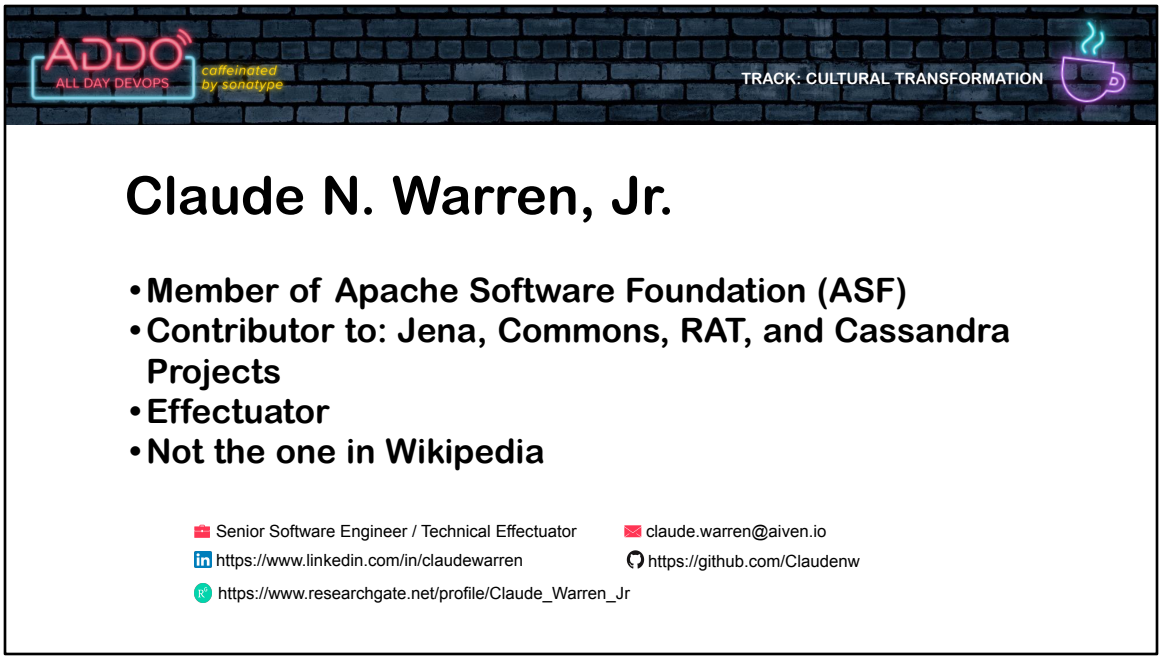


All of the slides and some background information is available at the URL and QR code on this slide.

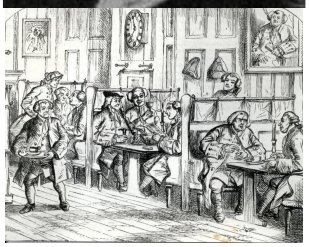
The Cathedral and the Bazaar was a blog post and later book by Erick S. Raymond. They are a metaphor for open source development strategies

The coffeehouse is development concept that I am introducing today, along the way we I present some risks associated with open source development and a possible solution for managing those risks. But first, let me introduce myself and present a little history.



Social media links at the bottom

Dad -> mom historian good segue because I want to start with the year 1688



1688

- **Edward Lloyd founded a coffeehouse on Tower Street in London.**
- **Lloyds coffeehouse featured discussions about shipping.**
- **Shipping was the high tech industry of the time.**

Lloyds customers were interested in the shipping trade
ship schedules,
manifests,
loss

As with any high tech industry there was a lot of money to be made and lost.

There were several major risks to shipping in the late 17th century

Sinking
According to Wikipedia, excluding those lost in battles, 318 ships were wrecked in the 17th century. More were lost without a trace.

Piracy
the 17th and early 18th centuries were the golden age of piracy.

Storms
Wikipedia lists 87 storms for the 17th century in the Atlantic. This is an undercount and does not include major storms in other oceans and seas.

Prices
Market activities akin to Tulip crash of 1637 we always a risk.

Lloyds customers developed a risk reduction strategy

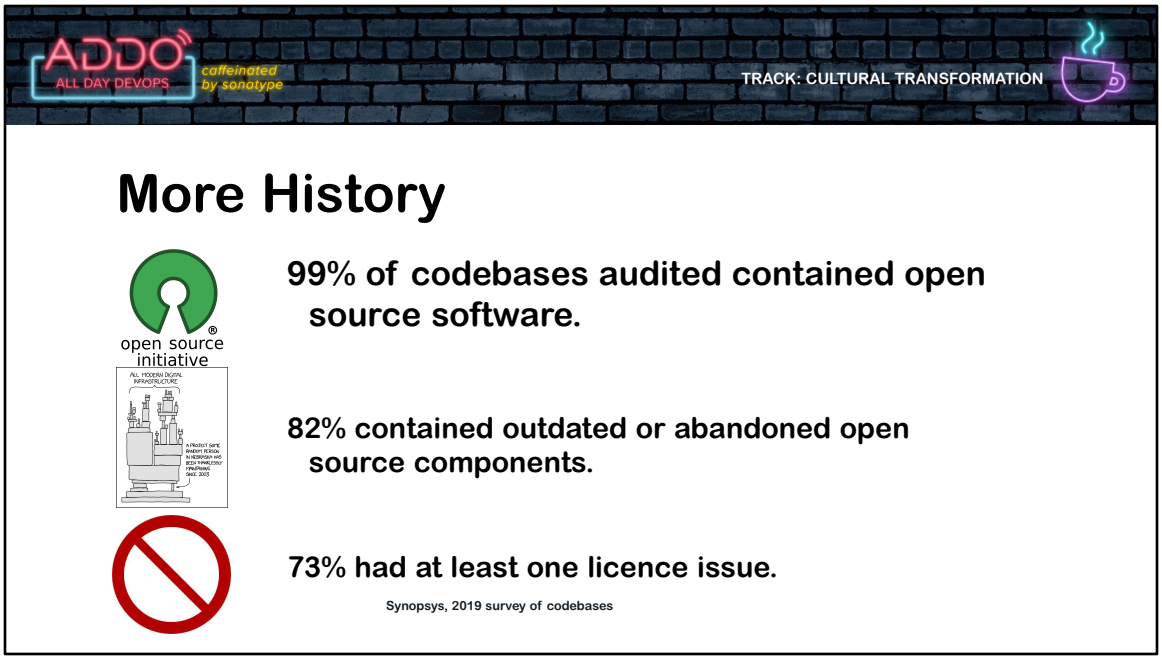
Lloyd's Risk Reduction Strategy

- **Investors invest in multiple ships and cargo.**
- **Share in the profit if the ship made a profit, or the loss if the ship made a loss.**

- **Effectively making Lloyd's one of the earliest insurance syndicates.**

So with the benefit of hindsight, and the knowledge that those that do not understand the past are doomed to repeat it… let us look at the risk exposure with Open Source Software.

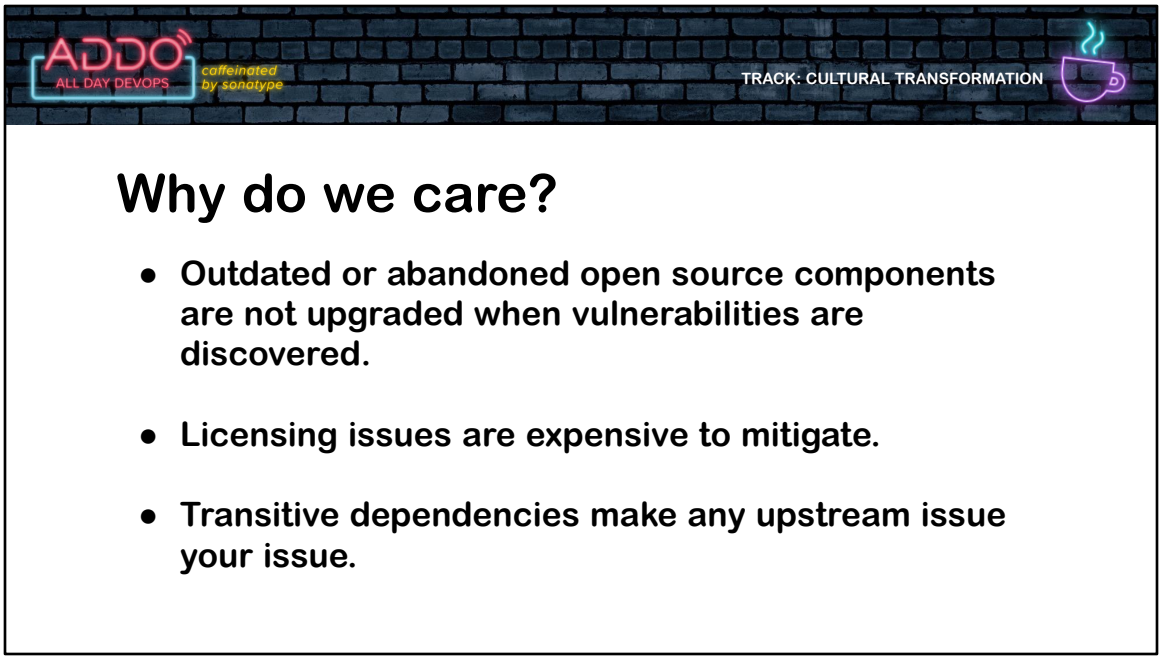
But first a bit of history of this talk. I Originally presented this subject as a lightning talk at Apache con 2022. It was 5 minutes without slides, and at this point I recited a bunch of dates and events in the history of open source software.
Those statistics are just too dry and boring to hold my attention.
So here are the 3 salient points.



Your project probably depends on open source projects..

This XKCD panel is so on point that “project in Nebraska”, and “maintained by random person in Nebraska” have become an idioms meaning any open source project maintained by a single person.

free and open-source software (FOSS) are not necessarily compatible with each other this can make it legally impossible to mix (or link) components that have different licenses.
For example, software that combined code released under version 1.1 of the Mozilla Public License (MPL) with code under the GNU General Public License (GPL) could not be distributed without violating one of the terms of one of the licenses.
even though both licenses are approved by both the Open Source Initiative and the Free Software Foundation.



Outdated/Abandoned

X-Lab, Github 2020 Insight Report

- 54 million active projects
- active developer accounts about 14 million.
- approximately 4 active projects per active developer.

How many of those are one developer projects in nebraska?

Why do they become abandoned?

- Life changes: Marriage, children, sick parents, new job, financial stresses; all vying for time that was previously spent on open source development.
- Disillusionment with the process

2015 Azer Koçulu deleted Left-pad NPM module breaking 1000's of websites.
Azer felt he was poorly treated in an IP naming spat by NPM he stated: This situation made me realize that NPM is someone's private land where corporate is more powerful than the people

2022 Marak Squires sabotaged Faker and Colors NPM modules breaking 1000s of websites.
Marak stated he was tired of supporting fortune 500 companies for free.

The Tidelift 2021 Open Source Maintainers Survey

- almost half of respondents listed lack of financial compensation as their top reason for disliking being a maintainer.

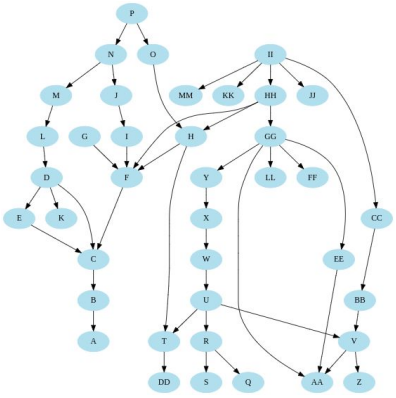
Licensing

Recently a number of projects have changed their licenses from pure open source to partial closed source, often for financial reasons. To mitigate these changes projects either have to purchase licenses or change software, or their users have to purchase licenses.

- 2018 MongoDB
- 2021 Elasticsearch and Kibana
- 2022 Akka
- 2023 Hashicorp

Transitive:
That last one should scare you.

A Nice Dependency Graph



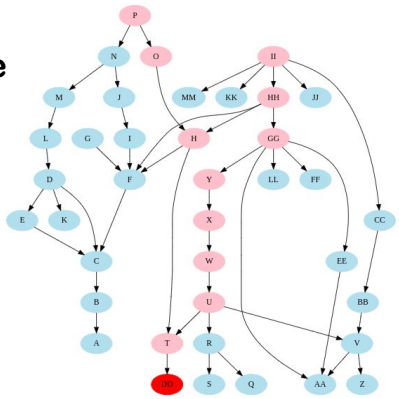
If you run project P and project O becomes abandoned, you can probably pick up the code and create a new module that does the same thing, or perhaps you will take over the maintenance of project O, or you might find a drop in or near drop in replacement.
If project O changes is license you have the same choices.

Not bad

A Hostile Dependency Graph

95% of vulnerable dependencies are transitive

– Endor Labs, The State of Dependency Management, 2022



But if DD is no longer supported and has a problem you have to know O depends on H which depends on T which depends on DD to understand that you have a problem.

Depending on the language involved and how it is loaded if DD suddenly disappears your entire application could grind to a halt.

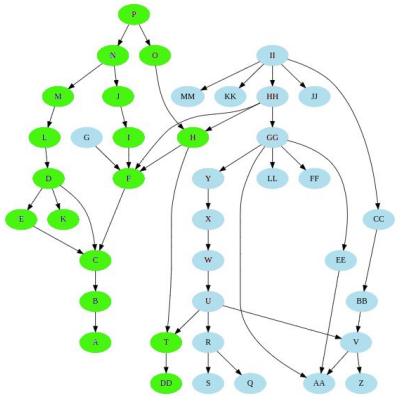
if DD is no longer supported and a security hole is discovered, how do you know that you may be impacted by the security problem?

If your project it II then the dependency chain is even longer

ADDO ALL DAY DEVOPS caffeinated by sonatype

TRACK: CULTURAL TRANSFORMATION

A Worrisome Dependency Graph



How do you track and manage all the transitive dependencies of your project?

Cathedral
This operates much like standard Closed source development systems.
There is a perception that Cathedral development is often so tightly controlled that it does not readily accept contributions. For software in this sub-category of Cathedral development it would be very difficult to get functionality added that did not fall in line with the design.
I want to be clear that not all Cathedral development is so tightly controlled that they do not accept external contributes, but it is a risk and each project should be assessed for their openness.

Bazaar
Bazaar has a wider pool of developers but tends to lack tightly controlled specific design goals.
Some argue that the lack of tight controls can lead to difficult to detect errors. One example is in encryption libraries where the optimization of the code path may yield weaker encryption, for example by replacing the random number generator with a weaker version.
Let me be clear again, I am not claiming this is always a risk but one that should be assessed for each project.

Combo

There are projects that use a mix of the two strategies.

All of these development strategies have their place and can produce excellent results.

Each has weaknesses and to go with its strengths.

The point is that users of the projects should be aware of strategy was used and how that impacts the type of component being developed.

Are you overwhelmed yet?

- Find all the upstream dependencies
- Analyze the issues
  - License conflict
  - Wrong development strategy for module
- Analyzing the risks
  - Developer community crash
  - License change
- Analyze the mitigation
  - Ability to contribute to upstream project
  - Ability to fork project if necessary
- The unknown unknowns
- The things you know that ain't so

Have a cup of coffee

- Enhanced energy levels
- Less likely to suffer heart failure
- Not as likely to suffer stroke

All of these benefits of coffee I got off the internet. But, as that great US President Abraham Lincoln said “Don’t trust everything you read on the internet”

Open a Coffeehouse

- **Meet like minded people**
- **Meet people with different opinions**
- **Have open discussion on high tech topics**

Edward Lloyd did, and it brought the shipping industry its first insurance company.

But more seriously, what can an organization do to mitigate the risks.

The Organizational Solution

Change organizational perception of open source software

- **Stop thinking of open source as a cost avoidance strategy.**
- **Start supporting open source development as an insurance policy.**

Every € a company puts toward open source support is a € they will not have to spend on mitigation when the "project in Nebraska" collapses, and every € invested in open source buys good will from the ecosystem.

The Organizational Solution - OSPO

Create an OSPO

(Open Source Program Office)

An program office within your organization that supports the Open source projects your organization is dependant upon.

Use OSPO as …

The Organizational Solution - OSPO

Use the OSPO:

- **as a firewall between organizational secret sauce and the open source community.**
  - **Don't show them the internal code**
  - **Keep valuable IP from leaking.**
- **Provide technical support to internal developers**

Hire developers within the OSPO work the Open Source projects your Organization is dependant upon.

Looking externally: Use the OSPO to analyze upstream projects

The Organizational Solution - OSPO

Use the OSPO to analyze all upstream projects for:

- **Community viability**
- **Openness to contribution**
- **Existing license issues**
- **Process fit for purpose**

Beyond the classic Bus factor (or lottery factor if you prefer) there are other metrics to measure community and project viability.

the Community Health Analytics in Open Source Software group (CHAOSS) publishes metrics for various dimensions

The Apache Kibble project calculates and displays many metrics in a dashboard.

Your OSPO can determine which metrics to use and apply them to the upstream communities

Do the organization's projects have licensing conflict issues?
Do the upstream projects have licensing issues?

Is the development process of the upstream project fit for purpose. Are they building a security system in a way that would allow easy injection of malicious code? Are they building a system that claims to want lots of contributions but whose process

makes contributing difficult. This edges up to the openness to contribution but also covers the cases of bad development processes.

Transition: You can also use the OSPO to support upstream projects

The Organizational Solution - OSPO

Use the OSPO to support upstream projects by:

- **Hiring developers to work on the projects**
- **Donating money or resources to projects**
- **Supporting existing development foundations**

Hire developers that already work on the project, or that are interested in working on the project, or have experience working on other projects in the same organization, or simply developers that have an interest in open source. The goal here is to ensure that the upstream projects have vibrant communities supporting them.

I should note that while I say “Developers” here, I include documentation specialists, communication specialists, testers, anything that the upstream projects need.

In some cases it may be possible to
donate money to help support one or more developers,
donate money to help support the project infrastructure, or perhaps
donate time on organizational systems for builds, or storage for built products.

finally: support existing foundations

There are a number of foundations FSF, Linux, Apache to name a three, that support a number of projects, they are always open to support.

CONCLUSION: There are a number of ways to support open source development, but check with your legal counsel to ensure that the effort does not run afoul of local

ordinances.

But what is the community solution? Is there a community solution to lower the risk, to share the expense, and still reap the benefits?

ADDO ALL DAY DEVOPS caffeinated by sonatype

TRACK: CULTURAL TRANSFORMATION

The Community Solution - Coffeehouse

In association with other OSPOs and interested parties create a coffeehouse like environment

- **Share data about open source projects**
- **Select metrics**
- **Publish reports on common projects**

In association with other OPSOs and interested parties create a coffeehouse like environment.

Share data about open source projects

If we think back to the dependency diagram, there are a lot of projects that need to be evaluated. But in a community effort, each OSPO only need evaluate some of the projects, provided they can trust other OSPOs to do a similar level of research.

There is a concept called Inner Source development, often thought of as open source within a firewall. Multiple OSPOs can join together to form an InnerSource project to evaluate common projects they are dependant upon. Data being freely shared within the inner source group.

The coffeehouse can also help select metrics

CHAOSS, Apache Kibble, OSI, and many more provide metric implementations. By working together and discussing the various benefits and drawbacks of specific metrics each OSPO can select the most appropriate metrics for what they are trying to achieve.

Research from InnerSource projects and individual OSPOs could be published. Reports like Endor Labs', The State of Dependency Management, or X-lab's Github 2020 Insight report are examples of this type of report.

In Closing

In closing,
The cathedral and the Bazaar are good metaphors for styles of software development.
Let's add the coffeehouse as a metaphor for supporting open source development as **insurance** against catastrophic failures of upstream projects

Thank You

Image Credits

- Lloyd: Lloyds Corporation, https://www.lloyds.com/about-lloyds/history/coffee-and-commerce
- Coffeehouse: Stevens, William, James Macaulay, and William Haig Miller. The Leisure Hour: an Illustrated Magazine for Home Reading. London: [W. Stevens, printer, etc.],
- Sinking: Shipwreck off Nantucket (Wreck off Nantucket after a Storm) - painting by William Bradford (MET, 1971.192)
- Pirates: "Signing the Articles" from the 1936 Pac-Kups "Jolly Roger Pirates" trading card set
- Cargo Overboard: Hand-coloured aquatint engraving of the "A picturesque voyage to India, by the way of China" / by Thomas Daniell, R.A., and William Daniell, A.R.A.
- Tulip Price: Adam Smith Institute, 23 Great Adam St, London.
- Risk Reduction: “The Rescue”, Maurice Poirson
- OSS: By Colin Viebrock - Simon Phipps, former president of OSI, official SVG, Public Domain, https://commons.wikimedia.org/w/index.php?curid=1853734
- XKCD: https://xkcd.com/2347
- Prohibition sign, GravisZro - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=9547672
- Cathedral: Interior of Antwerp Cathedral, Peeter Neeffs the elder
- Bazaar: A Turkish Bazaar, 186, Amadeo Preziosi
- Overwhelmed: Peter Welleman, Public domain, via Wikimedia Commons
- Coffeehouse: Bodleian Library, University of Oxford, Public domain, via Wikimedia Commons
- OSPO: Jacob de Gheyn II, CC0, via Wikimedia Commons
- Community: Bodleian Library, University of Oxford, Public domain, via Wikimedia Commons
- Graphs by Claude N. Warren, Jr.

Slides: https://github.com/Claudenw/Cathedral-Bazaar-Coffeehouse

References

- Apache Software Foundation, https://www.apache.org
- Community Health Analytics in Open Source Software (CHAOSS), https://chaoss.community/
- Copyleft, https://www.gnu.org/licenses/copyleft.en.html#:~:text=Copyleft%20is%20a%20general%20method,in%20the%20public%20domain%2C%20uncopyrighted.
- Debian Free Software Guidelines, https://www.debian.org/social_contract#guidelines
- Endor Labs, The State of Dependency Management, https://www.endorlabs.com/state-of-dependency-management
- Free Software Foundation, https://www.fsf.org/
- InnerSource Commons, https://innersourcecommons.org/
- Linux Foundation, https://www.linuxfoundation.org/
- Open Source Initiative, https://opensource.com/about
- Raymond, Eric S., The Cathedral and the Bazaar, http://www.catb.org/~esr/writings/cathedral-bazaar/
- Tidelift, 2021 Tidelift Open Source Maintainer Survey, https://tidelift.com/subscription/the-tidelift-maintainer-survey
- X-Lab, Github 2020 Insight Report, https://oss.x-lab.info/github-insight-report-2020-en.pdf


ADDO
ALL DAY DEVOPS
caffeinated by sonatype