

Rust Without Fear

The Microsoft Journey

Rust Moravia Meetup

October 2025



@ciura_victor



@ciura_victor@hachyderm.io



@ciuravictor.bsky.social

Victor Ciura
~~Principal Engineer~~
Rambling Idiot
Rust Tooling @ Microsoft

About me



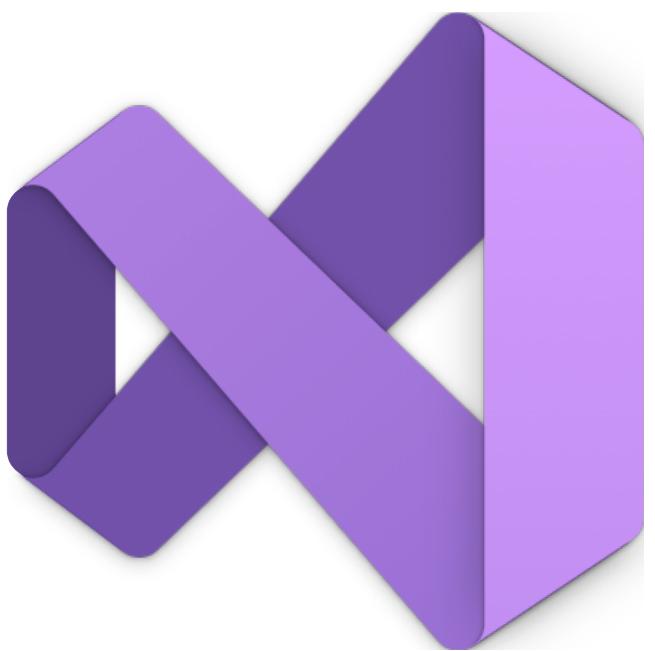
Advanced Installer



Clang Power Tools



Oxidizer SDK



Visual C++



Rust Tooling
Microsoft



@ciura_victor



@ciura_victor@hachyderm.io



@ciuravictor.bsky.social

Disclaimer

I'm just an engineer, with some opinions on stuff...



Touch Points

- How it started - original incubations
- How it's going - current projects
- Developer sentiment 😁 😊 😐 😔 😠
- Current challenges
- Areas of investment
- Future challenges

How it started...



OSS hidden crabs

Under the radar projects

Utilities/CLI



COG saving experiments

Hackathon projects

But Why Rust?



How it's going

How it's going

Rust at Microsoft is on a super-accelerated trajectory nowadays 💣

How it's going

Rust at Microsoft is on a super-accelerated trajectory nowadays 💣

How it's going

Rust at Microsoft is on a super-accelerated trajectory nowadays 💣

I've come to work with many of the teams spearheading these Rust efforts

How it's going

Rust at Microsoft is on a super-accelerated trajectory nowadays 💣

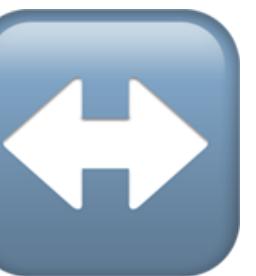
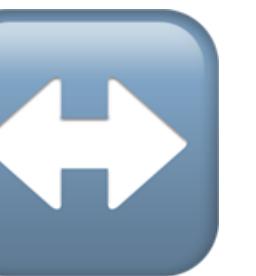
I've come to work with many of the teams spearheading these Rust efforts

How it's going

Rust at Microsoft is on a super-accelerated trajectory nowadays 

I've come to work with many of the teams spearheading these Rust efforts

My team's mission (in [DevDiv](#)) is to pave the path for Rust @ Microsoft
and make our [tooling](#) the gold standard for Rust devs
-- just like we did with C++, C#, TypeScript

C#  **Rust**  **C++**

They need to play nice together... for a looong time!

Ongoing Efforts

Ongoing Efforts

- Making changes in our **SDL** operations (evolving **Compliance** technologies)
 - **1ES** : Rust-ready

Ongoing Efforts

- Making changes in our **SDL** operations (evolving **Compliance** technologies)
 - **1ES** : Rust-ready
- Completing our deployment of **CodeQL** (integrated with GitHub Copilot learnings)

Ongoing Efforts

- Making changes in our **SDL** operations (evolving **Compliance** technologies)
 - **1ES** : Rust-ready
- Completing our deployment of **CodeQL** (integrated with GitHub Copilot learnings)
- Continue to invest in **hardening C & C++ code**

Ongoing Efforts

- Making changes in our **SDL** operations (evolving **Compliance** technologies)
 - **1ES** : Rust-ready
- Completing our deployment of **CodeQL** (integrated with GitHub Copilot learnings)
- Continue to invest in **hardening C & C++ code**
- Standardizing on **Rust** and other **memory safe languages**

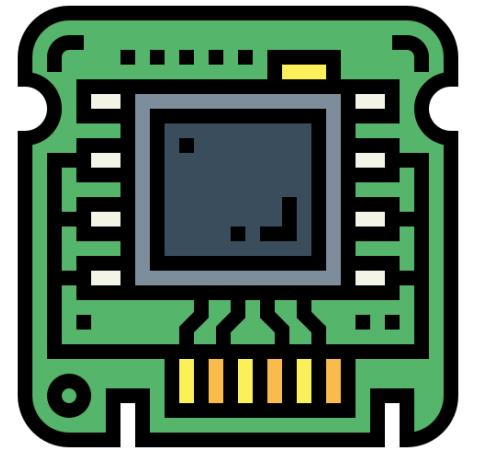
Ongoing Efforts

- Making changes in our **SDL** operations (evolving **Compliance** technologies)
 - **1ES** : Rust-ready
- Completing our deployment of **CodeQL** (integrated with GitHub Copilot learnings)
- Continue to invest in **hardening C & C++ code**
- Standardizing on **Rust** and other **memory safe languages**
- Contribute  to support the work of the **Rust Foundation** & core **OSS** projects

Ongoing Efforts

- Making changes in our **SDL** operations (evolving **Compliance** technologies)
 - **1ES** : Rust-ready
- Completing our deployment of **CodeQL** (integrated with GitHub Copilot learnings)
- Continue to invest in **hardening C & C++ code**
- Standardizing on **Rust** and other **memory safe languages**
- Contribute 💰 to support the work of the **Rust Foundation** & core **OSS** projects
- Assist developers making the *transition* from C, C++, C# to Rust
 - Investing in Rust **developer tooling**
 - Streamlining interop for **hybrid** projects

Extreme range of operation



Rust @Microsoft

- Project Mu
- Pluto security processor
- SymCrypt (C++ ➔ Rust) + [rustls](#)
- Azure Integrated HSM
- Azure Boost Agents
- Open VMM / Open HCL
- Hyper-V
- Azure SDK for Rust
- Azure Data Explorer
- Drasi
- MIMIR
- Caliptra - Hardware Root of Trust
- Hyperlight / WASM
- ... 🤪



TBD:

- ⚙️ [Windows core components](#)
- ☁️ [Microservices](#)

Rusty Windows

Rust already in the Windows kernel (since 2023)

```
C:\Windows\System32>dir win32k*
Volume in drive C has no label.
Volume Serial Number is E60B-9A9E
```

_rs = Rust!

```
Directory of C:\Windows\System32
```

04/15/2023	09:50 PM	708,608	win32k.sys
04/15/2023	09:49 PM	3,424,256	win32kbase.sys
04/15/2023	09:49 PM	110,592	win32kbase_rs.sys
04/15/2023	09:50 PM	4,194,304	win32kfull.sys
04/15/2023	09:49 PM	40,960	win32kfull_rs.sys
04/15/2023	09:49 PM	69,632	win32krnl.sys
04/15/2023	09:49 PM	98,304	win32ksgd.sys
		7 File(s)	8,646,656 bytes
		0 Dir(s)	116,366,049,280 bytes free

Rusty Windows

Ported **Windows 11** core components from C++ to **Rust**

- DirectWrite
- GDI
- ... 🤪

Win32k GDI REGION

Prepare for Windows 11
Windows 11 enterprise feature control
What's new in Windows 11, version 24H2
What's new in Windows 11, version 23H2

Rust in the Windows kernel

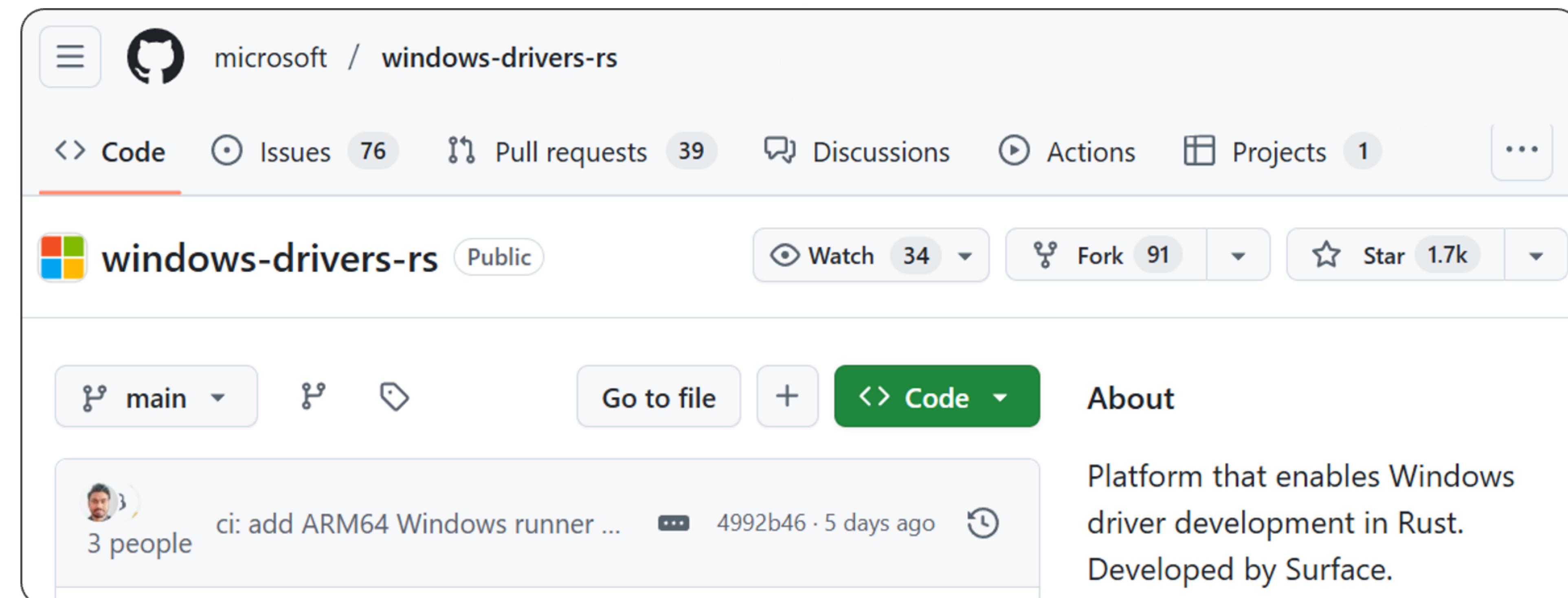
There's a new implementation of [GDI region](#) in `win32kbase_rs.sys`. Since Rust offers advantages in reliability and security over traditional programs written in C/C++, you'll continue to see more use of it in the kernel.

2 primary developer 3 months 6 KLOC not including test

Driver SDK for Rust

Enable Windows **driver development** in Rust

💻 Surface Hid Mini Driver is now written in Rust - based on this framework



Rust @ Edge

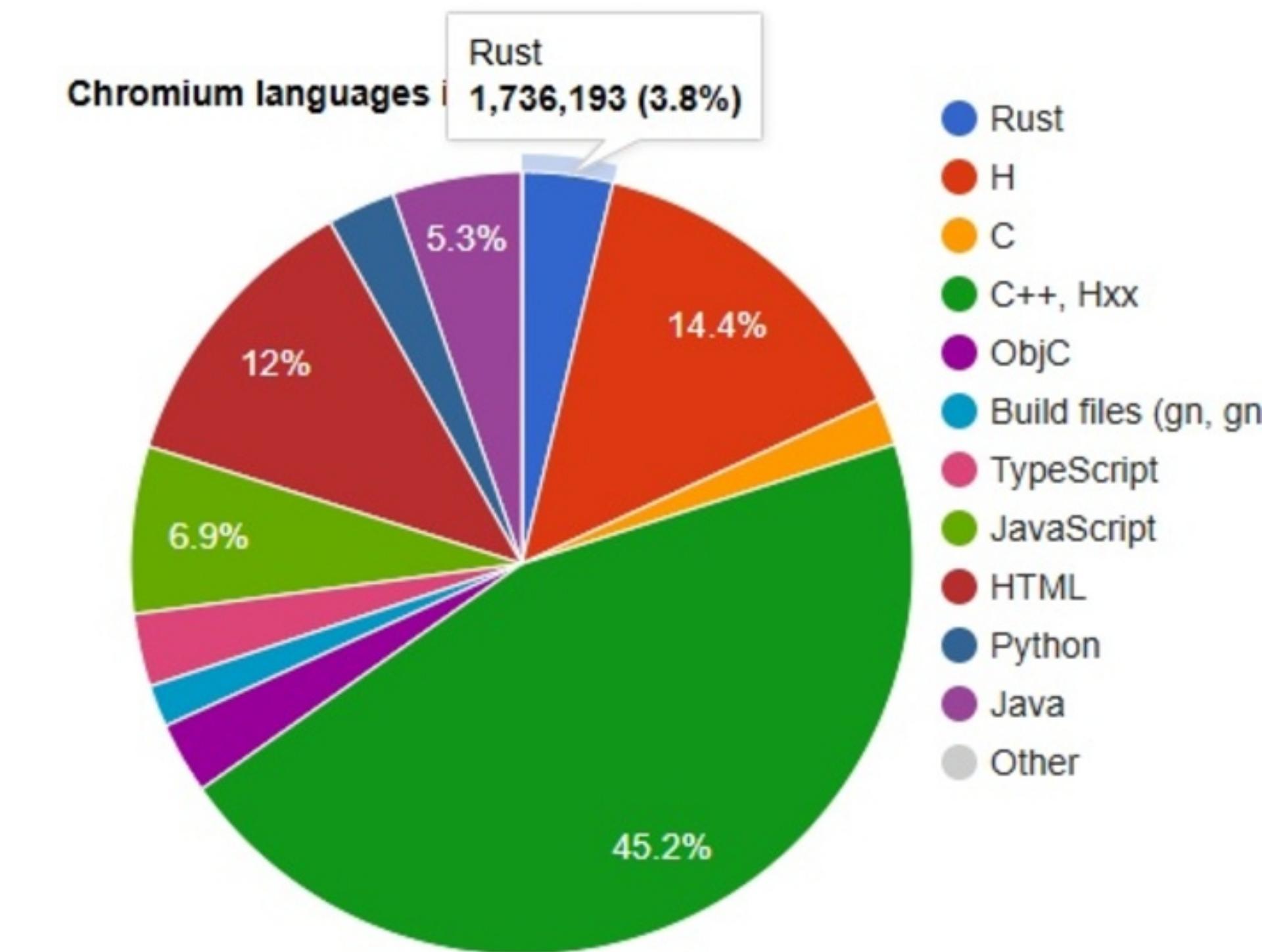
- Many new components
- Security tokens
- Password strength manager
- New Check&Sum hash algorithm

• ...



Languages distribution in Chromium

chromium/chromium repository statistics on Sep 2025



Oxidation 🔥

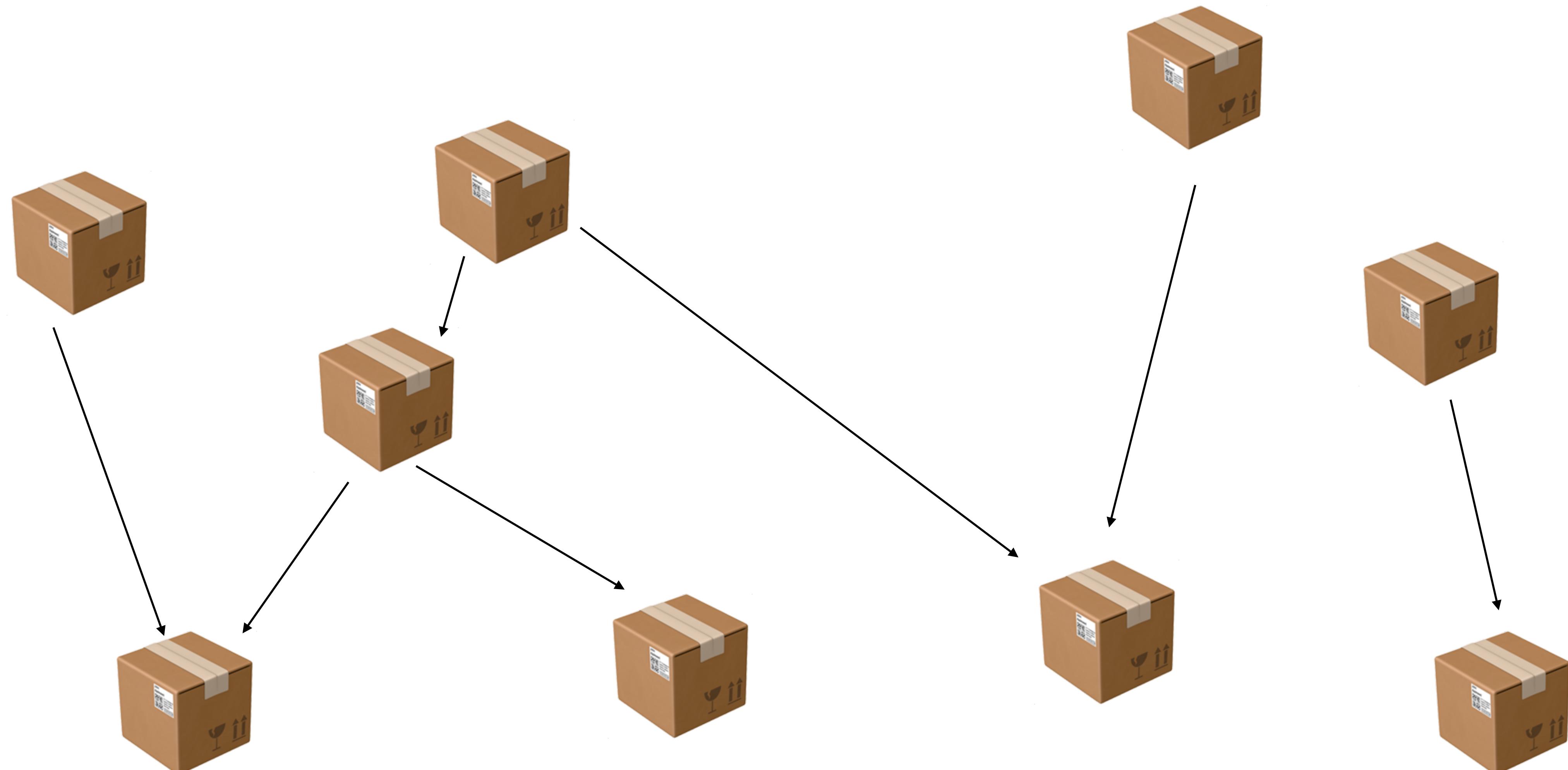
More oxidation 🦀 efforts in progress...

TBD 😕

Ecosystem

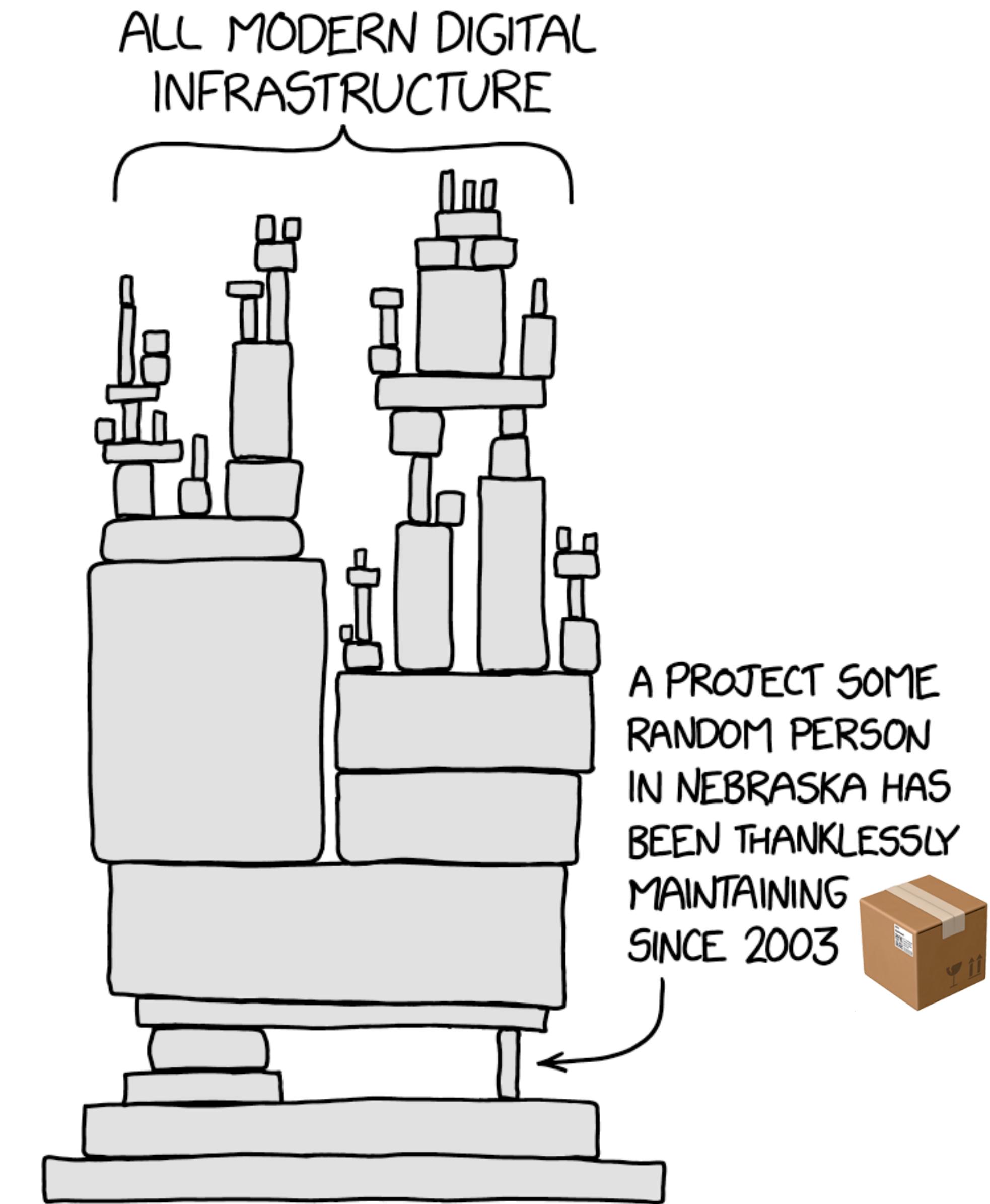
Enterprise-grade tooling

Crate Registry

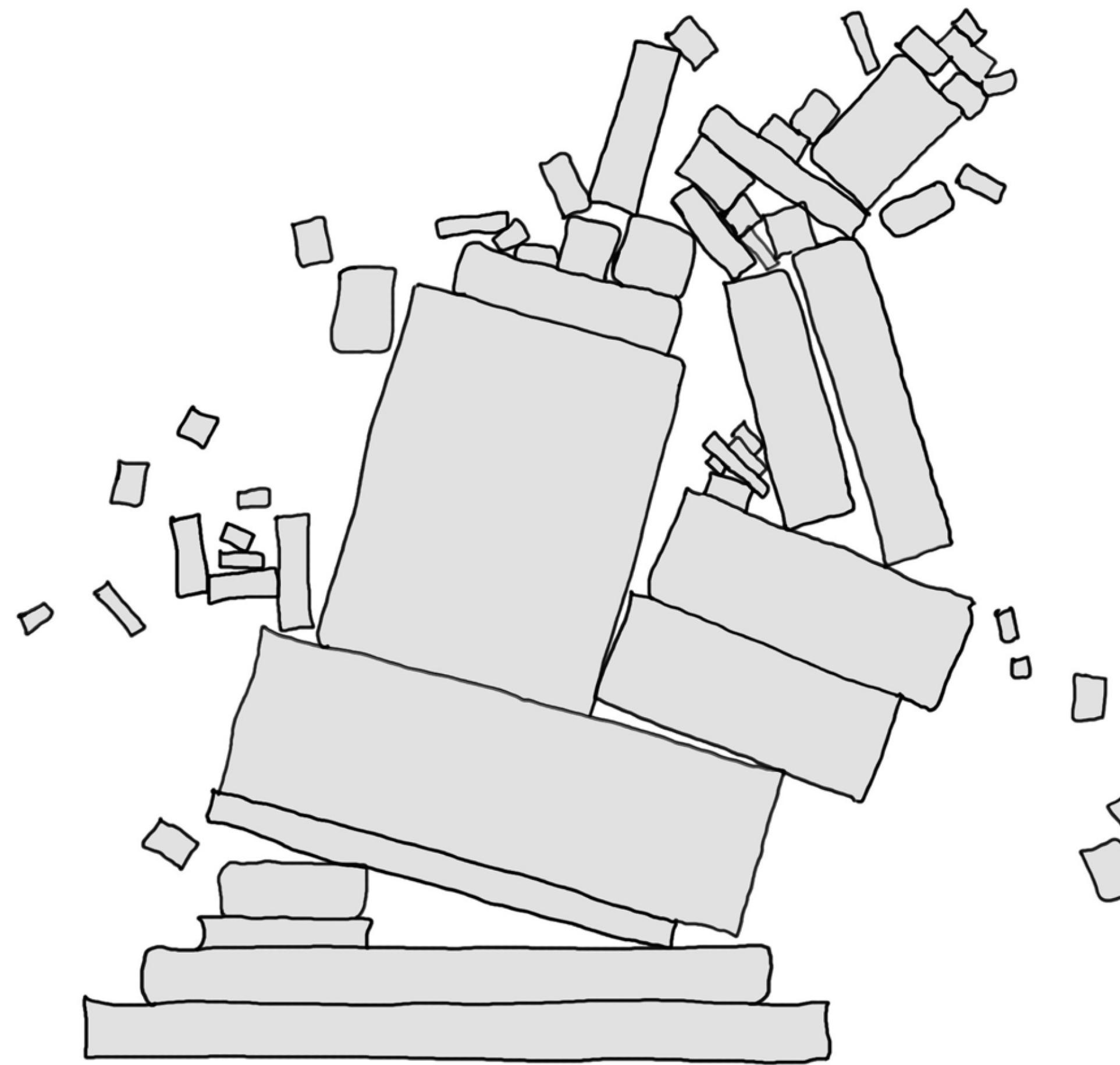


Amazing & thriving ecosystem!

Crate Registry



Crate Registry



Rust Crate Review System

A system that records **guidance** from enterprise developers on using Rust crates, both **public** and **internal** ones

- What crates should my project use, or not use?
- How should I **evaluate** public crates? (and record the evaluation)
- What are the **preferred crates** for particular purposes?
- How to keep a rigorous **SBOM** posture for the project?

Rust Crate Review System

Rust Crate Review System

- External crate dependencies, come with the inherent **risk**, in the form of potential **security** issues, **stability** issues, and **support** and **maintenance** related issues

Rust Crate Review System

- External crate dependencies, come with the inherent **risk**, in the form of potential **security** issues, **stability** issues, and **support** and **maintenance** related issues

Rust Crate Review System

- External crate dependencies, come with the inherent **risk**, in the form of potential **security** issues, **stability** issues, and **support** and **maintenance** related issues
- Proactively ensure that enterprise Rust ecosystem is built on the **stable** and **thriving** part of the OSS Rust ecosystem, lowering the risk of being affected
 - eg. vulnerability reported on a hobby-project crate, with a single owner who is not maintaining it anymore

Rust Crate Review System

- External crate dependencies, come with the inherent **risk**, in the form of potential **security** issues, **stability** issues, and **support** and **maintenance** related issues
- Proactively ensure that enterprise Rust ecosystem is built on the **stable** and **thriving** part of the OSS Rust ecosystem, lowering the risk of being affected
 - eg. vulnerability reported on a hobby-project crate, with a single owner who is not maintaining it anymore

Rust Crate Review System

- External crate dependencies, come with the inherent **risk**, in the form of potential **security** issues, **stability** issues, and **support** and **maintenance** related issues
- Proactively ensure that enterprise Rust ecosystem is built on the **stable** and **thriving** part of the OSS Rust ecosystem, lowering the risk of being affected
 - eg. vulnerability reported on a hobby-project crate, with a single owner who is not maintaining it anymore
- A set of unbiased Rust **crate evaluation criteria**, used for assessment of adoptability of third-party crates by any internal Rust project
 - lowering the company's vulnerability on third-party OSS solutions

Rust Crate Review System

- External crate dependencies, come with the inherent **risk**, in the form of potential **security** issues, **stability** issues, and **support** and **maintenance** related issues
- Proactively ensure that enterprise Rust ecosystem is built on the **stable** and **thriving** part of the OSS Rust ecosystem, lowering the risk of being affected
 - eg. vulnerability reported on a hobby-project crate, with a single owner who is not maintaining it anymore
- A set of unbiased Rust **crate evaluation criteria**, used for assessment of adoptability of third-party crates by any internal Rust project
 - lowering the company's vulnerability on third-party OSS solutions

Rust Crate Review System

- External crate dependencies, come with the inherent **risk**, in the form of potential **security** issues, **stability** issues, and **support** and **maintenance** related issues
- Proactively ensure that enterprise Rust ecosystem is built on the **stable** and **thriving** part of the OSS Rust ecosystem, lowering the risk of being affected
 - eg. vulnerability reported on a hobby-project crate, with a single owner who is not maintaining it anymore
- A set of unbiased Rust **crate evaluation criteria**, used for assessment of adoptability of third-party crates by any internal Rust project
 - lowering the company's vulnerability on third-party OSS solutions
- A **unified**, **unbiased**, highly **automatable** crate **scoring system** used throughout all teams/projects in the company



The video player displays a presentation slide with a purple background. On the left, large white text reads "Create security in 2025". Below this, the name "ADAM HARVEY" is displayed in yellow. At the bottom left, there is a logo for "Rust Nation UK". The video player interface includes a play button, a progress bar showing 0:00 / 50:05, and a set of control icons at the bottom right.

Crates security in
2025

ADAM HARVEY

Rust Nation UK

0:00 / 50:05

II CC ⚙️ 🔍

Crate security in 2025 - Adam Harvey

youtube.com/watch?v=GXkvX9A9xME

Private Crate Publishing

- Publishing to internal ADO feeds
- Discoverability
- Central documentation ("docs.rs")
- Consuming 1P crates
- Challenges of org silos and micro-repositories
- Crossing permission boundaries



Rust in Production

Learn by doing: Exploration → Flighting → Production

Rust in Production

Learn by doing: Exploration → Flighting → Production

- Direct impact: improve security & reduce operation cost

Rust in Production

Learn by doing: Exploration → Flighting → Production

- Direct impact: **improve security & reduce operation cost**
- **Gain experience** with transitioning to Rust in production

Rust in Production

Learn by doing: Exploration → Flighting → Production

- Direct impact: **improve security & reduce operation cost**
- **Gain experience** with transitioning to Rust in production
- **Costs of learning** Rust?

Rust in Production

Learn by doing: Exploration → Flighting → Production

- Direct impact: improve security & reduce operation cost
- Gain experience with transitioning to Rust in production
- Costs of learning Rust?
- Costs of porting to Rust?

Rust in Production

Learn by doing: Exploration → Flighting → Production

- Direct impact: improve security & reduce operation cost
- Gain experience with transitioning to Rust in production
- Costs of learning Rust?
- Costs of porting to Rust?
- Costs of writing new Rust components?

Rust in Production

Learn by doing: Exploration → Flighting → Production

- Direct impact: improve security & reduce operation cost
- Gain experience with transitioning to Rust in production
- Costs of learning Rust?
- Costs of porting to Rust?
- Costs of writing new Rust components?
- Is the full pipeline of Rust tooling ready?

Rust in Production

Learn by doing: Exploration → Flighting → Production

- Direct impact: improve security & reduce operation cost
- Gain experience with transitioning to Rust in production
- Costs of learning Rust?
- Costs of porting to Rust?
- Costs of writing new Rust components?
- Is the full pipeline of Rust tooling ready?
- Dealing with debugging woes

Rust in Production

Learn by doing: Exploration → Flighting → Production

- Direct impact: improve security & reduce operation cost
- Gain experience with transitioning to Rust in production
- Costs of learning Rust?
- Costs of porting to Rust?
- Costs of writing new Rust components?
- Is the full pipeline of Rust tooling ready?
- Dealing with debugging woes
- Performance targets, POGO, etc.

Rust in Production

Learn by doing: Exploration → Flighting → Production

- Direct impact: improve security & reduce operation cost
- Gain experience with transitioning to Rust in production
- Costs of learning Rust?
- Costs of porting to Rust?
- Costs of writing new Rust components?
- Is the full pipeline of Rust tooling ready?
- Dealing with debugging woes
- Performance targets, POGO, etc.
- Costs of maintaining a hybrid C++/Rust codebase?



Ergonomic & efficient
interop ... at scale

Rust / C++ interoperability



Choose... ~~none~~ some?

Rust / C++ interoperability

 Choose... ~~none~~ some?

- No need for excessive `unsafe` keyword

Rust / C++ interoperability

 Choose... ~~none~~ some?

- No need for excessive `unsafe` keyword
- **No perf overhead** (avoid marshaling costs, eg. copying strings)

Rust / C++ interoperability

 Choose... ~~none~~ some?

- No need for excessive `unsafe` keyword
- No perf overhead (avoid marshaling costs, eg. copying strings)
- No boilerplate or re-declarations / No C++ annotations

Rust / C++ interoperability

 Choose... ~~none~~ some?

- No need for excessive `unsafe` keyword
- No perf overhead (avoid marshaling costs, eg. copying strings)
- No boilerplate or re-declarations / No C++ annotations
- Broad types support - with safety

Rust / C++ interoperability

 Choose... ~~none~~ some?

- No need for excessive `unsafe` keyword
- No perf overhead (avoid marshaling costs, eg. copying strings)
- No boilerplate or re-declarations / No C++ annotations
- Broad types support - with safety
- Avoid lowering through C FFI

Rust / C++ interoperability

 Choose... ~~none~~ some?

- No need for excessive `unsafe` keyword
- No perf overhead (avoid marshaling costs, eg. copying strings)
- No boilerplate or re-declarations / No C++ annotations
- Broad types support - with safety
- Avoid lowering through C FFI
- Ergonomics - with safety

Rust / C++ interoperability

 Choose... ~~none~~ some?

- No need for excessive `unsafe` keyword
- No perf overhead (avoid marshaling costs, eg. copying strings)
- No boilerplate or re-declarations / No C++ annotations
- Broad types support - with safety
- Avoid lowering through C FFI
- Ergonomics - with safety
- Works with dynamic libraries (including the weirdness* of Windows DLLs, CRT)

Rust / C++ interoperability

Choose... ~~none~~ some?

- No need for excessive `unsafe` keyword
- No perf overhead (avoid marshaling costs, eg. copying strings)
- No boilerplate or re-declarations / No C++ annotations
- Broad types support - with safety
- Avoid lowering through C FFI
- Ergonomics - with safety
- Works with dynamic libraries (including the weirdness* of Windows DLLs, CRT)
- Plays well with C++ ABI

Choose... ~~none~~ some?

- No need for excessive `unsafe` keyword
- No perf overhead (avoid marshaling costs, eg. copying strings)
- No boilerplate or re-declarations / No C++ annotations
- Broad types support - with safety
- Avoid lowering through C FFI
- Ergonomics - with safety
- Works with dynamic libraries (including the weirdness* of Windows DLLs, CRT)
- Plays well with C++ ABI
- Easily automated

Rust / C++ interoperability

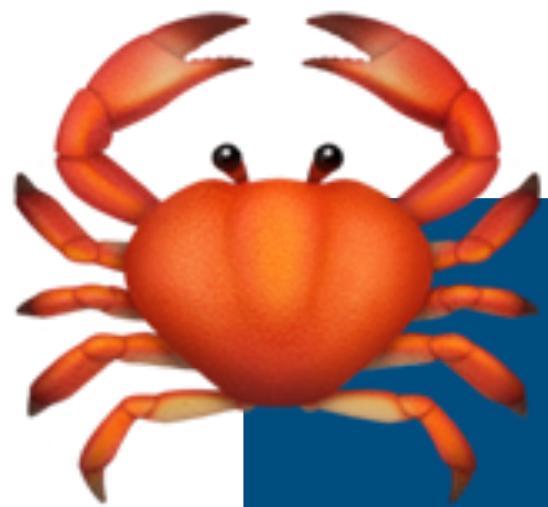
 Choose... ~~none~~ some?

- No need for excessive `unsafe` keyword
- No perf overhead (avoid marshaling costs, eg. copying strings)
- No boilerplate or re-declarations / No C++ annotations
- Broad types support - with safety
- Avoid lowering through C FFI
- Ergonomics - with safety
- Works with dynamic libraries (including the weirdness* of Windows DLLs, CRT)
- Plays well with C++ ABI
- Easily automated
- Hybrid build systems (cargo  MSBuild, CMake, bazel, buck2...)



Duck-Tape Chronicles

Rust/C++ Interop



Rust/C++ Interop:

Carcinization or Intelligent Design?

Strengths today 😊

Strengths today 😊

- Makes one a lot more conscious of pitfalls - explicit memory management

Strengths today 😊

- Makes one a lot more conscious of pitfalls - explicit memory management
- Memory-safety related vulnerabilities reduced

Strengths today 😊

- Makes one a lot more conscious of pitfalls - explicit memory management
- Memory-safety related vulnerabilities reduced
- Data-race-related concurrency bugs reduced

Strengths today 😊

- Makes one a lot more conscious of pitfalls - explicit memory management
- Memory-safety related vulnerabilities reduced
- Data-race-related concurrency bugs reduced
- Rich ecosystem and streamlined dependency management

Strengths today 😊

- Makes one a lot more conscious of pitfalls - explicit memory management
- Memory-safety related vulnerabilities reduced
- Data-race-related concurrency bugs reduced
- Rich ecosystem and streamlined dependency management
- If it compiles, it works, much faster dev-compile iteration, (better with coding agents)

Strengths today 😊

- Makes one a lot more conscious of pitfalls - explicit memory management
- Memory-safety related vulnerabilities reduced
- Data-race-related concurrency bugs reduced
- Rich ecosystem and streamlined dependency management
- If it compiles, it works, much faster dev-compile iteration, (better with coding agents)
- GitHub Copilot flattens the learning curve

Strengths today 😊

- Makes one a lot more conscious of pitfalls - explicit memory management
- Memory-safety related vulnerabilities reduced
- Data-race-related concurrency bugs reduced
- Rich ecosystem and streamlined dependency management
- If it compiles, it works, much faster dev-compile iteration, (better with coding agents)
- GitHub Copilot flattens the learning curve
- Reduced friction => more motivation for devs to write tests

Strengths today 😊

- Makes one a lot more conscious of pitfalls - explicit memory management
- Memory-safety related vulnerabilities reduced
- Data-race-related concurrency bugs reduced
- Rich ecosystem and streamlined dependency management
- If it compiles, it works, much faster dev-compile iteration, (better with coding agents)
- GitHub Copilot flattens the learning curve
- Reduced friction => more motivation for devs to write tests
- Increased performance

Opportunities & Investments



Opportunities & Investments



- Better developer experience in IDE/VS Code

Opportunities & Investments



- Better developer experience in IDE/VS Code
- Async code debugging is painful

Opportunities & Investments



- Better developer experience in IDE/VS Code
- Async code debugging is painful
- Integrate Cargo with a larger build system

Opportunities & Investments



- Better developer experience in IDE/VS Code
- Async code debugging is painful
- Integrate Cargo with a larger build system
- Tooling and guidelines are still behind comparing with other languages

Opportunities & Investments



- Better developer experience in IDE/VS Code
- Async code debugging is painful
- Integrate Cargo with a larger build system
- Tooling and guidelines are still behind comparing with other languages
- Dynamic linking is challenging

Opportunities & Investments



- Better developer experience in IDE/VS Code
- Async code debugging is painful
- Integrate Cargo with a larger build system
- Tooling and guidelines are still behind comparing with other languages
- Dynamic linking is challenging
- C++, Rust, C# interop

Opportunities & Investments



- Better developer experience in IDE/VS Code
- Async code debugging is painful
- Integrate Cargo with a larger build system
- Tooling and guidelines are still behind comparing with other languages
- Dynamic linking is challenging
- C++, Rust, C# interop
- FFI is tough to do safely even in Rust

Opportunities & Investments



- Better developer experience in IDE/VS Code
- Async code debugging is painful
- Integrate Cargo with a larger build system
- Tooling and guidelines are still behind comparing with other languages
- Dynamic linking is challenging
- C++, Rust, C# interop
- FFI is tough to do safely even in Rust
- Some features we rely on not being stabilized

ONE DOES NOT SIMPLY



REWRITE IN RUST

makeameme.org

ONE DOES NOT SIMPLY

But we still do it!

REWRITE IN RUST

makeameme.org

Rust Without Fear

The Microsoft Journey

Rust Moravia Meetup

October 2025



@ciura_victor



@ciura_victor@hachyderm.io



@ciuravictor.bsky.social

Victor Ciura
~~Principal Engineer~~
Rambling Idiot
Rust Tooling @ Microsoft