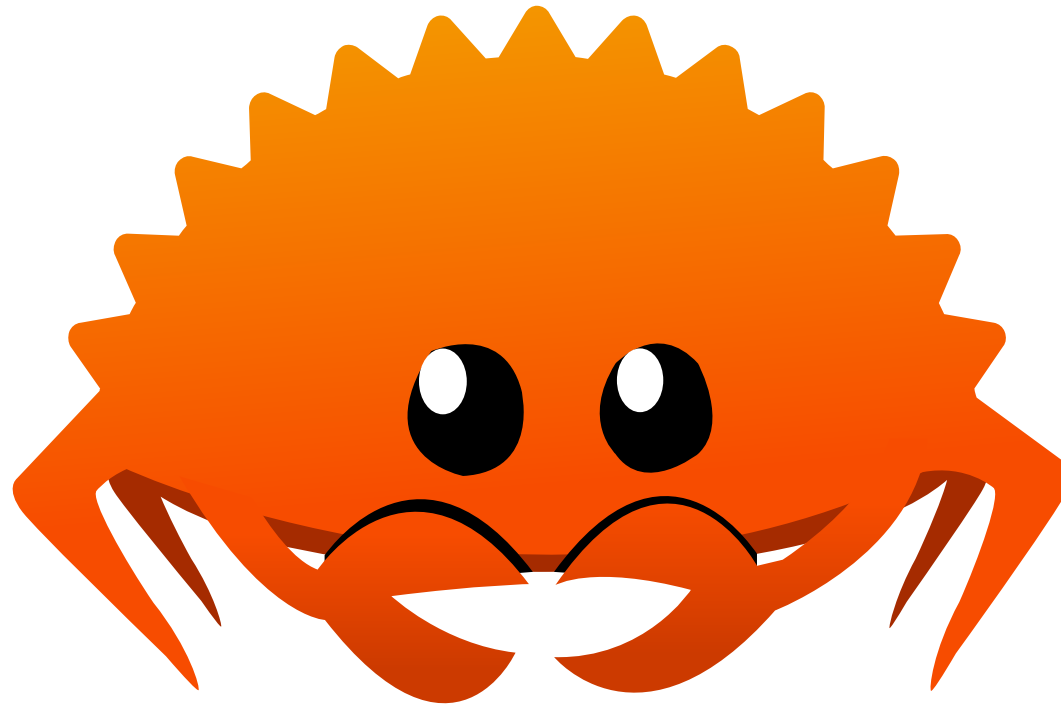




RIIR: Rewrite It In *Idiomatic* Rust (with AI)

Who am I?

One of the lead designers of Rust



Who am I?

Senior Principal Engineer at Amazon



Takeaways

- Rust + AI = marshmallows + chocolate 🍫

Takeaways

- Rust + AI = marshmallows + chocolate 🍫
- Symposium + Rust + AI = **s'mores** 😊



Takeaways

- Rust + AI = marshmallows + chocolate 🍫
- Symposium + Rust + AI = **s'mores** 😊
- Portability across agents today is **hard** 😞

Takeaways

- Rust + AI = marshmallows + chocolate 🍫
- Symposium + Rust + AI = **s'mores** 😊
- Portability across agents today is **hard** 😞
- Agent Client Protocol (ACP) **could solve it** 🙌



Not heard of Rust?

Low-level enough for a kernel...

usable enough for an application



Low-level enough for a kernel?

Low-level enough for a kernel?

Rust Goes Mainstream in the Linux Kernel

Rust is no longer experimental. It's now as acceptable as C in the Linux kernel.

Dec 10th, 2025 12:00pm by [Steven J. Vaughan-Nichols](#)

Low-level enough for a kernel?

Rust Goes Mainstream in the Linux Kernel

Rust is no longer experimental. It's now as acceptable as C in the Linux kernel.

Dec 10th, 2025 12:00pm by [Steven J. Vaughan-Nichols](#)

Rust in the Windows Kernel: Microsoft's 2026 Safety Push

Microsoft is rewriting Windows kernel components in Rust. What's been rewritten, the security rationale, and what it means for Rust developers in 2026.



Max Wells

Published: Mar 27, 2026 · Updated: May 4, 2026

High-level enough for an application?

High-level enough for an application?

00:00

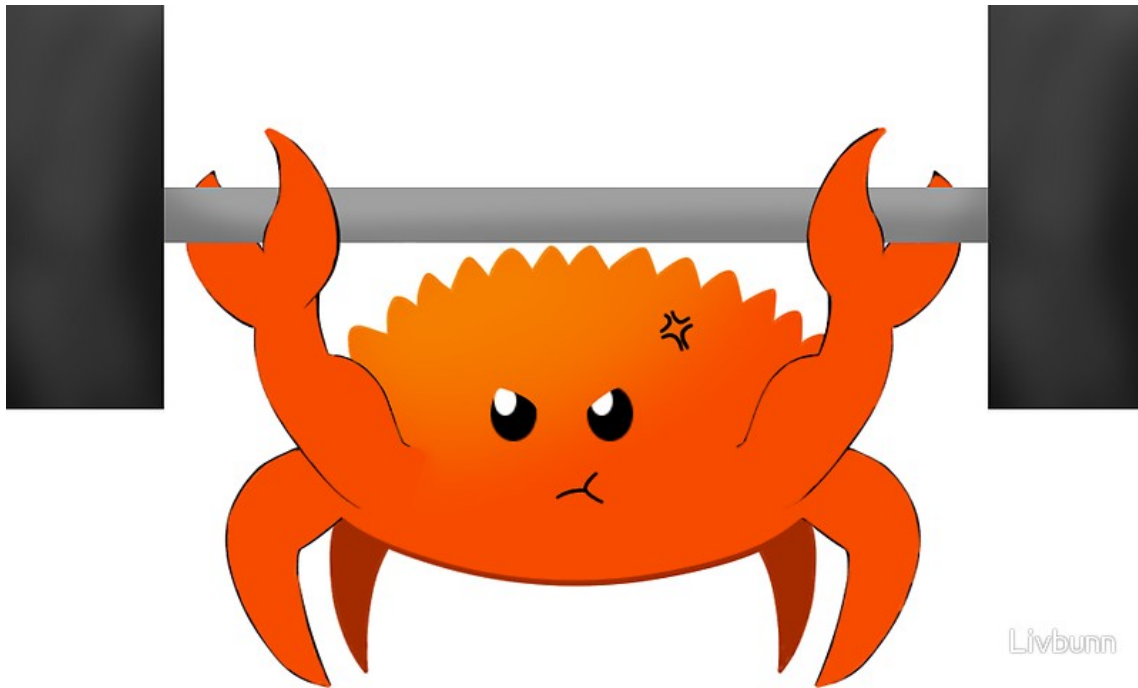
00:53

TUI oscilloscope written using ratatui

"Eat your Spinach"







Livbunn

Rust + LLM = dream-team?

- Guardrails

Rust + LLM = dream-team?

- Guardrails

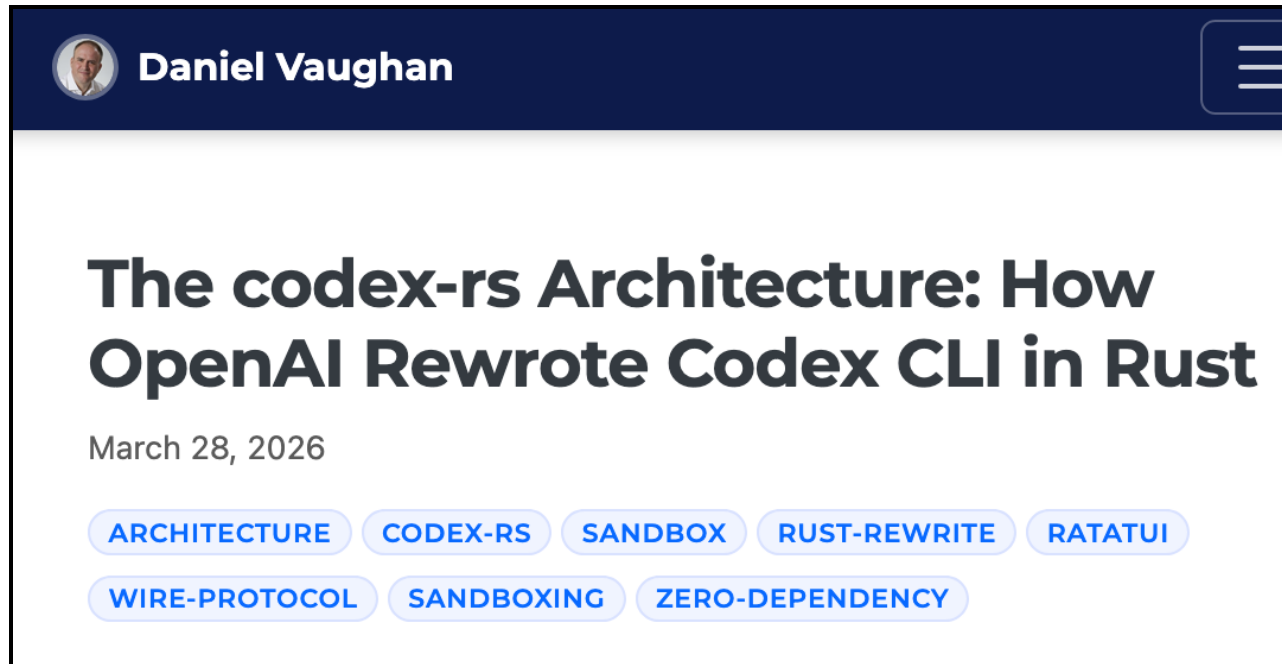


Rust + LLM = dream-team?

- Guardrails
- Versatility

Rust + LLM = dream-team?

- Guardrails
- Versatility



The screenshot shows a dark blue header with a profile picture and the name 'Daniel Vaughan' on the left, and a hamburger menu icon on the right. Below the header, the main title 'The codex-rs Architecture: How OpenAI Rewrote Codex CLI in Rust' is displayed in a large, bold, black font. Underneath the title, the date 'March 28, 2026' is shown in a smaller, grey font. At the bottom of the header area, there are two rows of blue, rounded rectangular tags containing the following text: 'ARCHITECTURE', 'CODEX-RS', 'SANDBOX', 'RUST-REWRITE', 'RATATUI' in the first row, and 'WIRE-PROTOCOL', 'SANDBOXING', 'ZERO-DEPENDENCY' in the second row.

Daniel Vaughan

The codex-rs Architecture: How OpenAI Rewrote Codex CLI in Rust

March 28, 2026

ARCHITECTURE CODEX-RS SANDBOX RUST-REWRITE RATATUI

WIRE-PROTOCOL SANDBOXING ZERO-DEPENDENCY

Rust + LLM = dream-team?

- Guardrails
- Versatility
- Efficiency

Rust + LLM = dream-team?

- Guardrails
- Versatility
- Efficiency

Cloudflare just got faster and more secure, powered by Rust

2025-09-26

ENGINEERING

Counter Service: How we rewrote it in Rust



Jia Long Loh · Pu Li · Muqi Li · Md Riyadh

20 Jun 2025 | 13 min read

Abstract

The Integrity Data Platform (IDP) team decided to rewrite our core microservices in Rust. It resulted in [70% infrastructure savings](#) and avoided many of its pitfalls. This article will elaborate on:

2x Performance, \$300k Savings: A Case Study in Rewriting a Critical Service in Rust

A Tale of Scale: When Your Service Hits a Wall

In the world of high-traffic systems, success often brings its own set of challenges.

During my internship at TikTok, one of our core payment services—a reliable workhorse built in Go—began showing signs of strain. As TikTok LIVE's user base grew, the CPU

Rust at Amazon

ALL THINGS DISTRIBUTED

NOW GO BUILD!

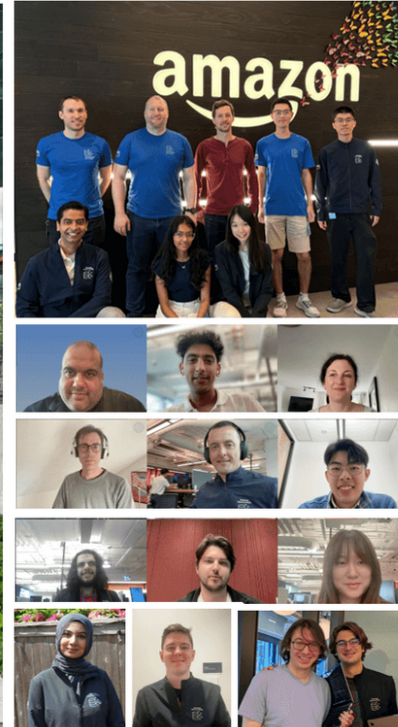
ARTICLES

@WERNER



Just make it scale: An Aurora DSQL story

May 27, 2025 • 3404 words



[Link to blog post](#)

Latest developments for Rust at Amazon

Latest developments for Rust at Amazon



Microsoft

// My goal is to eliminate every line of C and C++ from Microsoft by 2030. Our strategy is to combine AI *and* Algorithms to **rewrite Microsoft's largest codebases**. Our North Star is "1 engineer, 1 month, 1 million lines of code".

— Distinguished Engineer at Microsoft, [talking about a research project](#) (emphasis mine)

Bun's Zig to Rust Rewrite: Anthropic's AI Code Experiment

🕒 May 5, 2026 🗒 0



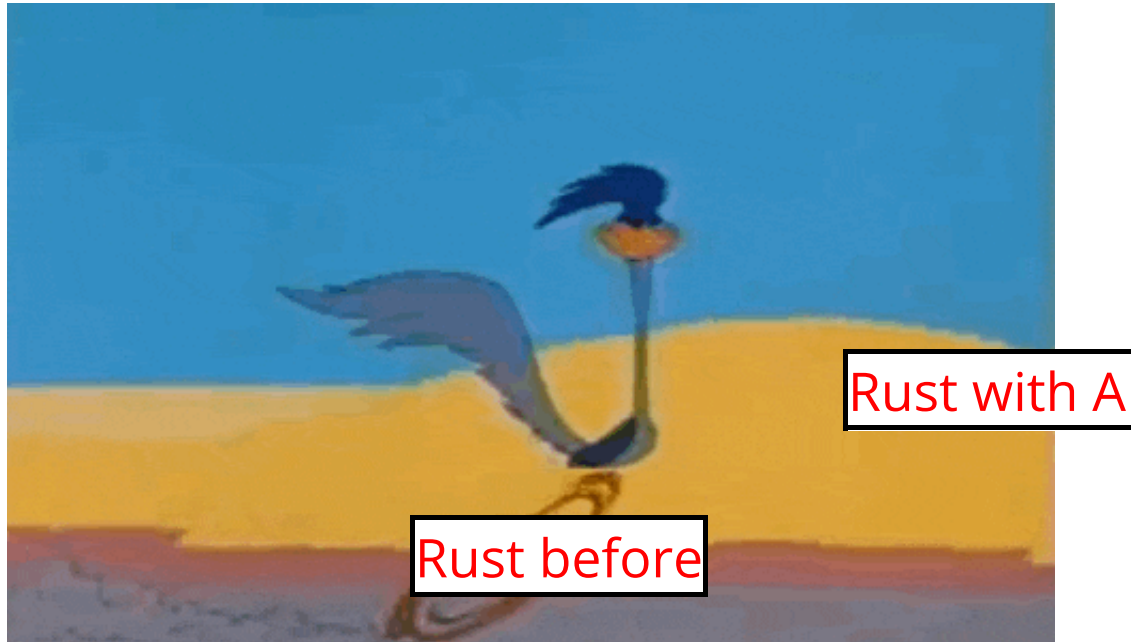
Roadrunner



Roadrunner



Roadrunner



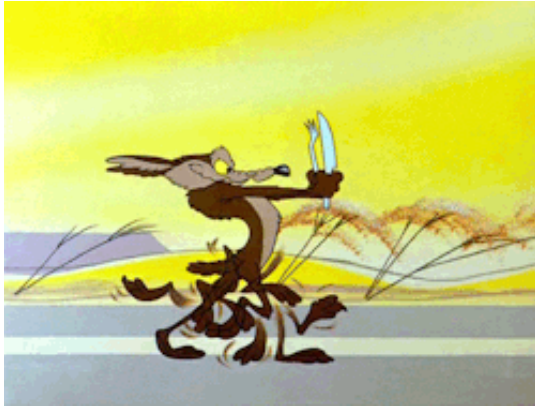
Eat your vegetables



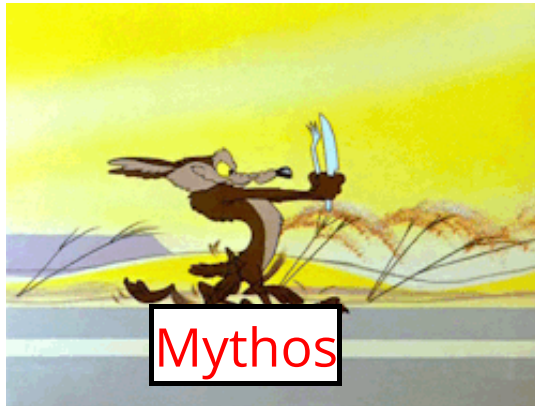
Roadrunner



Roadrunner



Roadrunner



Mythos



Rust

Memory
unsafety,
Reflection

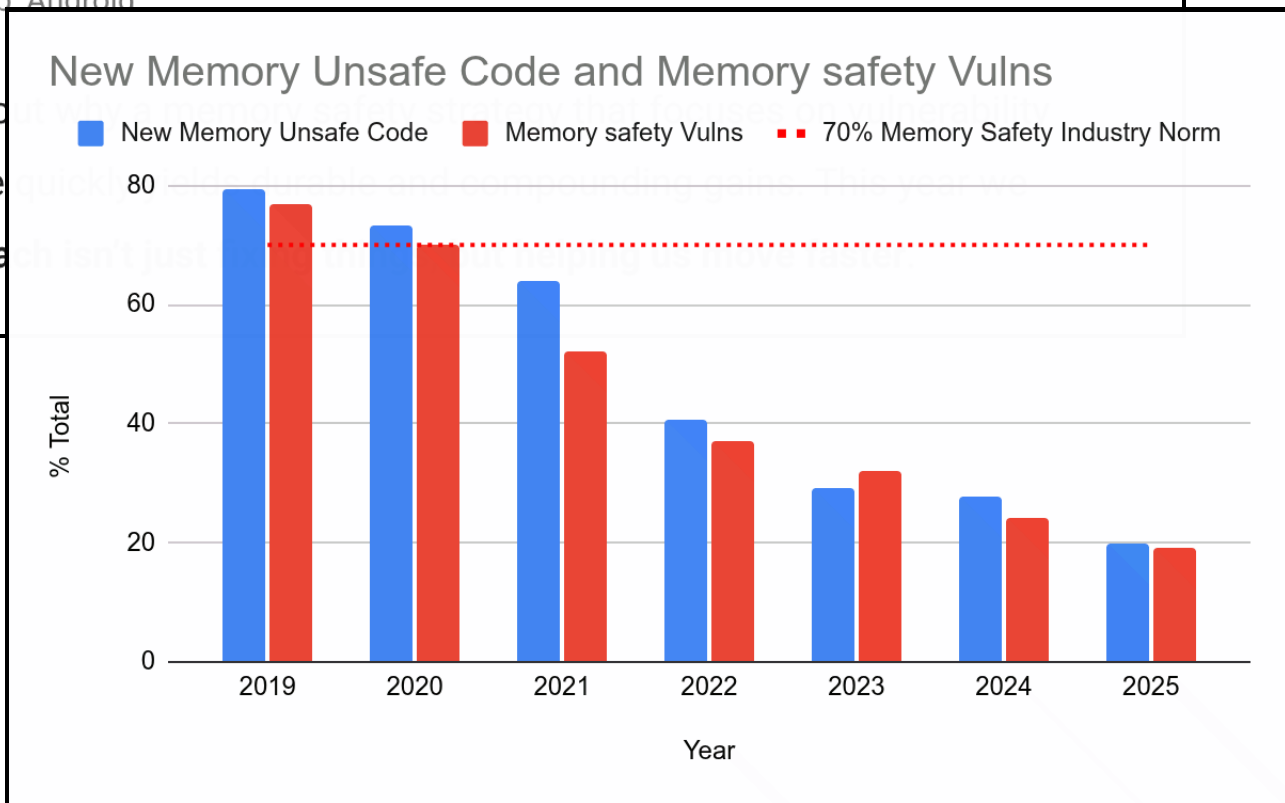
Rust is used in Android Mobile

Rust in Android: move fast and fix things

November 13, 2025

Posted by Jeff Vander Stoep, Android

Last year, we wrote about how Rust helped with memory safety prevention in new code. This year, we look at how this approach



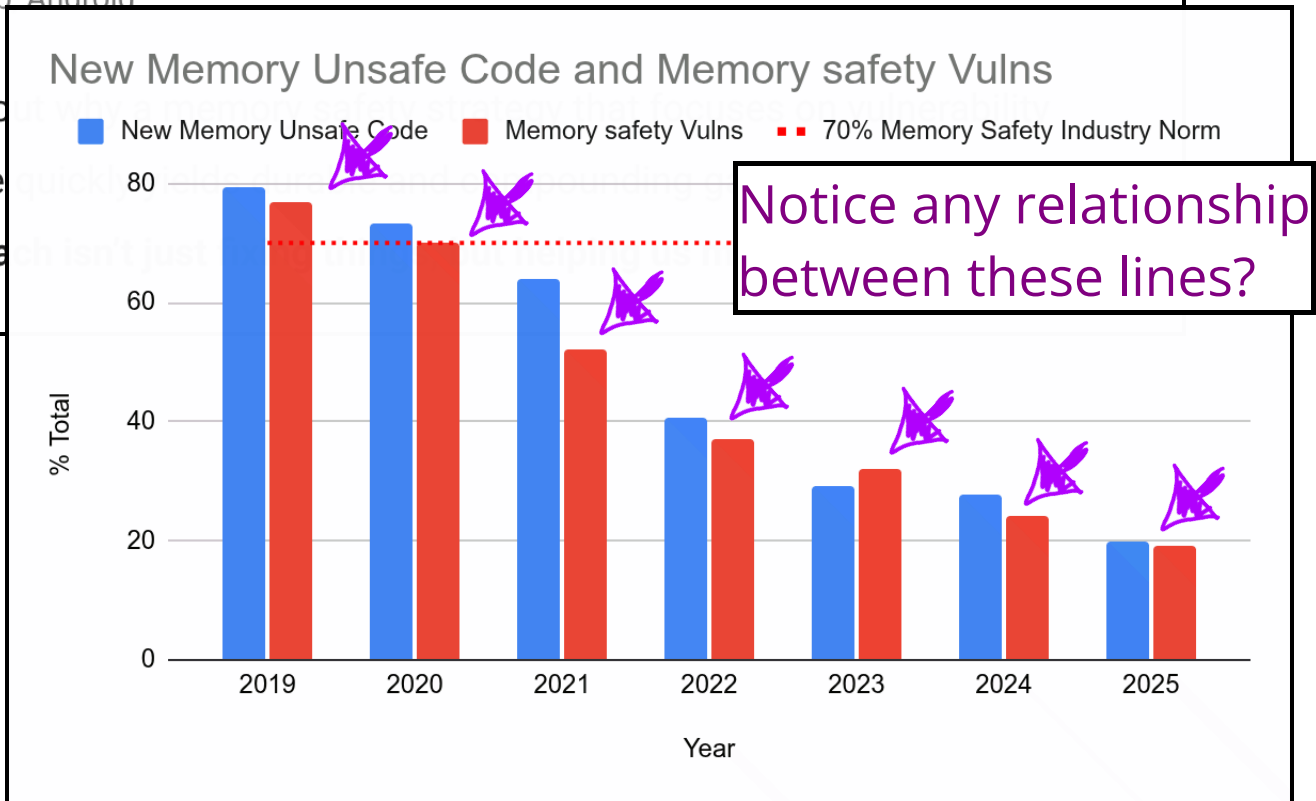
Rust is used in Android Mobile

Rust in Android: move fast and fix things


November 13, 2025

Posted by Jeff Vander Stoep, Android

Last year, we wrote about how Rust helped with memory safety prevention in new code. This year, let's look at how **this approach** is working.



Not just Google



tomorrow
belongs to those who embrace it
today

trending tech innovation business security advice buying guides

Navigation icons: globe, search, user, grid

/ tech

Home / Tech / Security

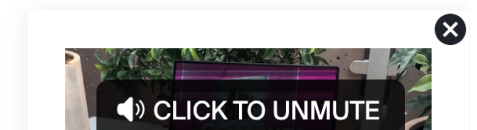
Rust will save Linux from AI, says Greg Kroah-Hartman

Now that doesn't mean Linux stable kernel maintainer Greg Kroah-Hartman thinks Rust is magic.




Written by **Steven Vaughan-Nichols**, Senior Contributing Editor

May 27, 2026 at 6:34 a.m. PT



Not just Google



tomorrow
belongs to those who embrace it
today

trending tech innovation business security advice buying guides

🌐 🔍 👤 ☰

/ tech

Home / Tech / Security



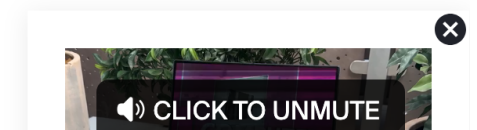
Rust will save Linux from AI, says Greg Kroah-Hartman

Now that doesn't mean Linux stable kernel maintainer Greg Kroah-Hartman thinks Rust is magic.




Written by **Steven Vaughan-Nichols**, Senior Contributing Editor

May 27, 2026 at 6:34 a.m. PT



Not just Google



tomorrow belongs to those who embrace it today

trending tech innovation business security advice buying guides

Navigation icons: globe, search, user, grid

/ tech

Home / Tech / Security

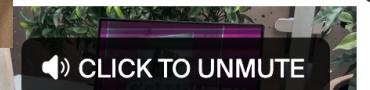


Rust will save Linux from AI, says Greg Kroah-Hartman

Now that doesn't mean Linux stable...
Greg Kroah-Hartman thinks Rust is n



Written by **Steven Vaughan-Nichols**, Senior Contributing Editor
May 27, 2026 at 6:34 a.m. PT



Not just memory safety

Remember Log4Shell?

You heard it here first, (runtime) reflection is the new memory unsafety.

Not just memory safety

Remember Log4Shell?

You heard it here first, (runtime) reflection is the new memory unsafety.



When your slogan is "a stich in time, saves nine" in an agentic world...



Rust right now

If coding is cheap...

Pick a language, libraries, etc based not on the code you have now,
but based on the **code you want to maintain going forward.**

If coding is cheap...

Pick a language, libraries, etc based not on the code you have now, but based on the **code you want to maintain going forward**.

So why not...

- Save money *and*
- Reduce exposure to vulnerabilities?

So... Rust + agents are good

But could they be better?

Example: Toasty

Toasty, an async ORM for Rust, is now on crates.io

April 3, 2026

Toasty is an asynchronous ORM for the Rust programming language that prioritizes ease of use. Toasty supports both SQL and NoSQL databases. Today, on the NoSQL front, only DynamoDB works, but I'm hoping to get more support in the coming months.

Here is how you might use it:

```
#[derive(Debug, toasty::Model)]
struct User {
    #[key]
    #[auto]
    id: u64,

    name: String,

    #[unique]
    email: String,

    #[has_many]
    todos: toasty::HasMany<Todo>,
}
```

Example: Toasty

Toasty, an async ORM for Rust, is now on crates.io

 April 3, 2026

Toasty is an asynchronous ORM for the Rust programming language that prioritizes ease of use. Toasty supports both SQL and NoSQL databases. Today, on the NoSQL front, only DynamoDB works, but I'm hoping to get more support in the coming months.

Here is how you might use it:

```
#[derive(Debug, toasty::Model)]
struct User {
    #[key]
    #[auto]
    id: u64,

    name: String,

    #[unique]
    email: String,

    #[has_many]
    todos: toasty::HasMany<Todo>,
}
```

Example: Toasty

Toasty, an async ORM for Rust, is now on crates.io

April 3, 2026

Toasty is an asynchronous ORM for the Rust programming language that prioritizes ease of use. Toasty supports both SQL and NoSQL databases. Today, on the NoSQL front, only DynamoDB works, but I'm hoping to get more support in the coming months.

Here is how you might use it:



```
#[derive(Debug, toasty::Model)]
struct User {
    #[key]
    #[auto]
    id: u64,

    name: String,

    #[unique]
    email: String,

    #[has_many]
    todos: toasty::HasMany<Todo>,
}
```

Example: Toasty

Toasty, an async ORM for Rust, is now on crates.io

April 3, 2026

Toasty is an asynchronous ORM for the Rust programming language that prioritizes ease of use. Toasty supports both SQL and NoSQL databases. Today, on the NoSQL front, only DynamoDB works, but I'm hoping to get more support in the coming months.

Here is how you might use it:

```
#[derive(Debug, toasty::Model)]
struct User {
    #[key]
    #[auto]
    id: u64,

    name: String,

    #[unique]
    email: String,

    #[has_many]
    todos: toasty::HasMany<Todo>,
}
```

Example: Using Toasty

```
// Create a user, the ID is not specified as it is set automatically  
toasty::create!(User {  
  name: "John Doe",  
  email: "john@example.com",  
}).exec(&mut db).await?;  
  
// Find the user by email  
let user = User::get_by_email(&mut db, "john@example.com").await?;
```

Example: Using Toasty


Macros and custom syntax

```
// Create a user, the id is not specified as it is set automatically  
toasty::create!(User {  
  name: "John Doe",  
  email: "john@example.com",  
}).exec(&mut db).await?;  
  
// Find the user by email  
let user = User::get_by_email(&mut db, "john@example.com").await?;
```

Example: Using Toasty

```
// Create a user, the ID is not specified as it is set automatically  
toasty::create!(User {  
  name: "John Doe",  
  email: "john@example.com",  
}).exec(&mut db).await?;  
  
// Find the user by email  
let user = User::get_by_email(&mut db, "john@example.com").await?;
```

New methods




Example: Using Toasty

```
// Create a user, the ID is not specified as it is set automatically
toasty::create!(User {
  name: "John Doe",
  email: "john@example.com",
}).exec(&mut db).await?;

// Find the user by email
let user = User::get_by_email(&mut db, "john@example.com").await?;
```

New methods




- **Guardrails:** API enforces type-safety, correct column names, etc
- **Versatility:** High-level and declarative
- **Efficiency:** Compiles to efficient code

Example: Using Toasty

```
// Create a user, the ID is not specified as it is set automatically  
toasty::create!(User {  
  name: "John Doe",  
  email: "john@example.com",  
}).exec(&mut db).await?;  
  
// Find the user by email  
let user = User::get_by_email(&mut db, "john@example.com").await?;
```

Type-safe, efficient



- **Guardrails:** API enforces type-safety, correct column names, etc
- **Versatility:** High-level and declarative
- **Efficiency:** Compiles to efficient code

Claude doesn't even know it exists

"What library do you recommend for working with sqlite in Rust?"

My usual recommendation:

Situation	Reach for
CLI tool / simple script	<code>rusqlite</code>
Async service (Tokio)	<code>sqlx</code>
Complex domain model, migrations-first	<code>diesel</code>
Async + ORM ergonomics	<code>sea-orm</code>

For the record, these too are all excellent libraries!

Claude, can you use toasty?

can you write me an execute that uses the Rust crate toasty to create a local database with all jpegs in the current directory and their file size

Claude, can you use toasty?

Claude, can you use toasty?

That's a lot of tokens...

Claude, can you use toasty?

can you write me an execute that uses the Rust crate toasty to create a local database with all jpegs in the current directory and their file size

Synthesized Toasty framework patterns and architected JPEG database solution >

Now I have everything I need. Here's the full example using Toasty 0.3 with the

`#[derive(toasty::Model)]` macro API (no schema file — that was the old approach):

Cargo.toml

toml

`[package]`

`name = "jpeg-index"`

`version = "0.1.0"`

`edition = "2021"`

`[dependencies]`

`toasty = { version = "0.3", features = ["sqlite"] }`



Claude, can you use toasty?

can you write me an execute that uses the Rust crate toasty to create a local database with all jpegs in the current directory and their file size

Synthesized Toasty framework patterns and architected JPEG database solution >

Now I have everything I need. Here's the full example using Toasty 0.3 with the

`#[derive(toasty::Model)]` macro API (no schema file — that was the old approach):

Cargo.toml

toml

`[package]`

`name = "jpeg-index"`

`version = "0.1.0"`

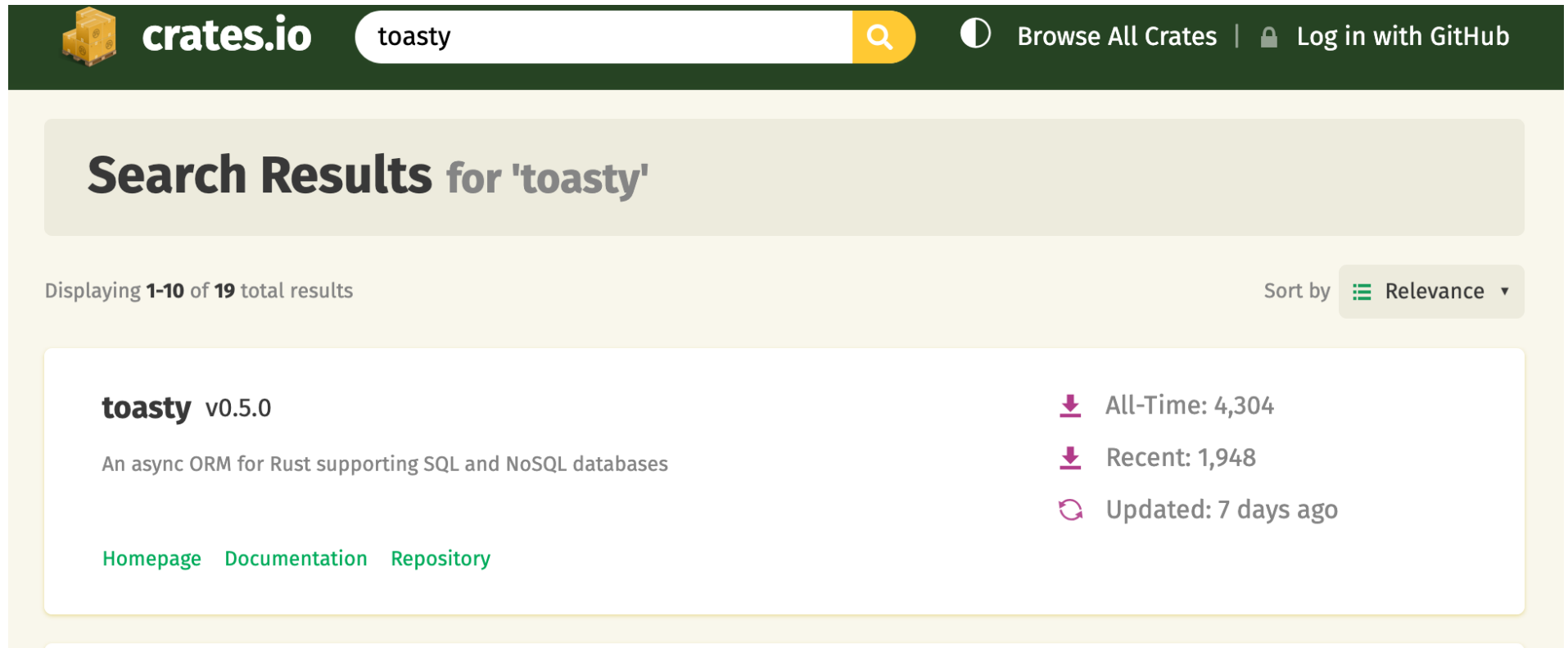
`edition = "2021"`

`[dependencies]`


`toasty = { version = "0.3", features = ["sqlite"] }`




Toasty on crates.io



The screenshot shows the crates.io search interface. At the top, there is a dark green header with the crates.io logo on the left, a search bar containing the text 'toasty', and navigation links for 'Browse All Crates' and 'Log in with GitHub'. Below the header, a light green banner displays 'Search Results for 'toasty''. Underneath, it indicates 'Displaying 1-10 of 19 total results' and a 'Sort by' dropdown menu set to 'Relevance'. The main content area features a card for the 'toasty' crate, version v0.5.0, described as 'An async ORM for Rust supporting SQL and NoSQL databases'. To the right of the description, statistics are shown: 'All-Time: 4,304', 'Recent: 1,948', and 'Updated: 7 days ago'. At the bottom of the card, there are links for 'Homepage', 'Documentation', and 'Repository'.

crates.io toasty  [Browse All Crates](#) | [Log in with GitHub](#)

Search Results for 'toasty'

Displaying **1-10** of **19** total results Sort by  Relevance ▾

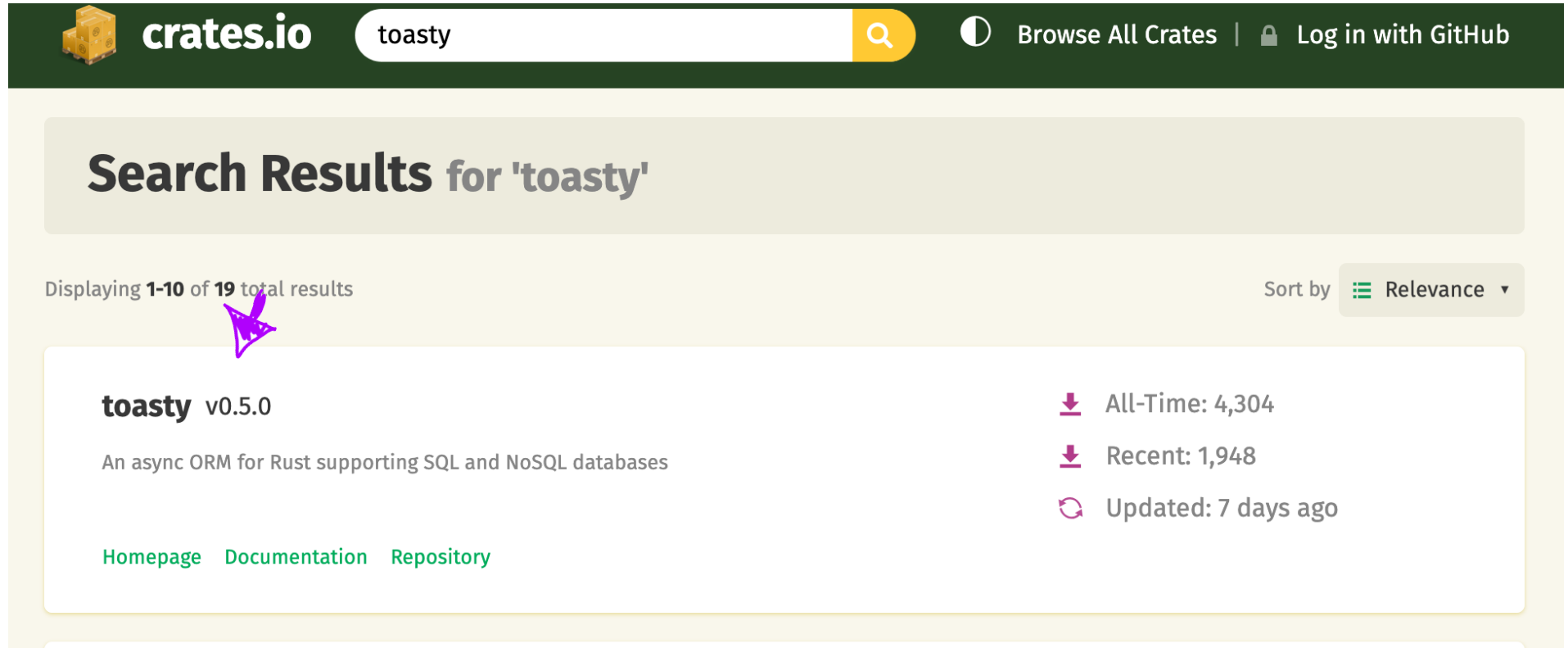
toasty v0.5.0 ↓ All-Time: 4,304

An async ORM for Rust supporting SQL and NoSQL databases ↓ Recent: 1,948

🔄 Updated: 7 days ago

[Homepage](#) [Documentation](#) [Repository](#)

Toasty on crates.io



The screenshot shows the crates.io search interface. At the top, there is a dark green header with the crates.io logo, a search bar containing the text 'toasty', and navigation links for 'Browse All Crates' and 'Log in with GitHub'. Below the header, a light green banner displays 'Search Results for 'toasty''. Underneath, it indicates 'Displaying 1-10 of 19 total results' and a 'Sort by' dropdown menu set to 'Relevance'. The first search result is for the 'toasty' crate, version v0.5.0. The description reads 'An async ORM for Rust supporting SQL and NoSQL databases'. To the right of the description, there are three statistics: 'All-Time: 4,304' (with a download icon), 'Recent: 1,948' (with a download icon), and 'Updated: 7 days ago' (with a refresh icon). At the bottom of the result card, there are three green links: 'Homepage', 'Documentation', and 'Repository'. A purple arrow points to the '19' in the 'Displaying 1-10 of 19 total results' text.

crates.io

toasty

Browse All Crates | Log in with GitHub

Search Results for 'toasty'

Displaying 1-10 of 19 total results

Sort by Relevance

toasty v0.5.0

An async ORM for Rust supporting SQL and NoSQL databases

All-Time: 4,304

Recent: 1,948

Updated: 7 days ago

[Homepage](#) [Documentation](#) [Repository](#)

The world is moving faster than ever

Training data can't keep up

The world is moving faster than ever

Training data can't keep up

can you write me an execute that uses the Rust crate toasty to create a local database with all jpegs in the current directory and their file size

Synthesized Toasty framework patterns and architected JPEG database solution >

Now I have everything I need. Here's the full example using Toasty 0.3 with the

`#[derive(toasty::Model)]` macro API (no schema file — that was the old approach):

Cargo.toml

```
toml

[package]
name = "jpeg-index"
version = "0.1.0"
edition = "2021"

[dependencies]
toasty = { version = "0.3", features = ["sqlite"] }
```



The world is moving faster than ever

Training data can't keep up

can you write me an execute that uses the Rust crate toasty to create a local database with all jpegs in the current directory and their file size

Synthesized Toasty framework patterns and architected JPEG database solution >

Now I have everything I need. Here's the full example using Toasty 0.3 with the

`#[derive(toasty::Model)]` macro API (no schema file — that was the old approach):

`Cargo.toml`

toml

`[package]`

`name = "jpeg-index"`

`version = "0.1.0"`

`edition = "2021"`

Rust 2024 hit stable Feb 2025

`[dependencies]`

`toasty = { version = "0.3", features = ["sqlite"] }`



Most frustrating thing of all?

Most frustrating thing of all?

...the Rust org cannot help



¹ I should say: I tried that toasty example twice, and the second time, Claude did use Toasty 0.5. But it still made a Rust crate in the 2021 edition.

..and this is where Symposium comes in



Member of the Rust Foundation's Innovation Lab

Welcoming Symposium to the Rust Innovation Lab

April 21, 2026 / Rust Foundation Team

Install symposium

```
$ cargo install symposium # or cargo binstall!
```

Install symposium

```
$ cargo install symposium # or cargo binstall!
```

```
$ cargo agents init
```

Setting up symposium for your user account.

Which agents do you use? (space to select, enter to confirm):

```
> [x] Claude Code  
   [x] Codex CLI  
   [ ] GitHub Copilot  
   [ ] Gemini CLI  
   [ ] Goose  
   [x] Kiro  
   [ ] OpenCode
```

Symposium gives general guidance

- General Rust guidance, e.g.,
 - Use Rust 2024
 - Use `cargo add` to add crates so you get the latest version
 - Run `cargo fmt` after making edits

Crate authors write skills

name	toasty-guidance
description	Guidance for using the Toasty async ORM crate — schema definition, CRUD, relations, queries, and transactions
crates	toasty
activation	always

Toasty is an async ORM for Rust supporting SQL (SQLite, PostgreSQL, MySQL) and NoSQL (DynamoDB). It prioritizes type safety and leans into each database's capabilities rather than hiding them.

Schema definition

Define models with `#[derive(toasty::Model)]`. Mark the primary key with `#[key]` and auto-generated fields with `#[auto]` (auto-increment for integers, UUID v7 for `uuid::Uuid`).

```
#[derive(Debug, toasty::Model)]
struct User {
    #[key]
    #[auto]
    id: i64,

    #[unique]
    email: String,
```



Symposium installs skills automatically

- Install hooks:
 - After every tool use, we check for new relevant skills and install them
 - **Result:** After agent runs `cargo add toasty`, it knows how to use it

Symposium provides APIs to help agents

```
> cargo agents crate-info serde
```


```
Crate: serde
```

```
Version: 1.0.228
```

```
Source: /Users/nikommat/.cargo/registry/src/index.crates.io/serde-1.0.228
```

Symposium provides APIs to help agents

```
> cargo agents crate: serde
Crate: serde
Version: 1.0.228
Source: /Users/nikommat/.cargo/registry/src/index.crates.io/serde-1.0.228
```

 Correct version used by your code

Symposium provides APIs to help agents

```
> cargo agents crate-info serde
```

```
Crate: serde
```

```
Version: 1.0.228
```

```
Source: /Users/nikommat/.cargo/registry/src/index.crates.io/serde-1.0.228
```



Let the agent browse full sources

Symposium aims to make agents extensible

"Everybody has something unique to offer"



Packaging up skills in the central repository



`symposium-dev/recommendations`

Packaging up skills in the central repository



symposium-dev/recommendations

```
toasty/
```

```
  SYMPOSIUM.toml
```

```
  create-model/
```

```
    SKILL.md
```

Packaging up skills in the central repository



symposium-dev/recommendations

toasty/

SYMPOSIUM.toml

create-model/

SKILL.md



```
# SYMPOSIUM.toml
```

```
name = "toasty"
```

```
crates = ["toasty"]
```

```
[[skills]]
```

```
source.path = "."
```

Packaging up skills in the central repository



symposium-dev/recommendations

```
toasty/  
  SYMPOSIUM.toml  
  create-model/  
    SKILL.md
```

```
# SYMPOSIUM.toml  
name = "toasty"  
crates = ["toasty"]  
  
[[skills]]  
source.path = "."
```

```
# SKILL.md
```

```
---
```

```
description: Create a model using the toasty crate
```

```
---
```

```
Here are code examples that create toasty models...
```

Packaging up skills with your crate



symposium-dev/recommendations

toasty/
SYMPOSIUM.toml

```
# SYMPOSIUM.toml  
name = "toasty"  
crates = ["toasty"]  
  
[[skills]]  
source = "crate"
```

Packaging up skills with your crate



symposium-dev/recommendations

toasty/
SYMPOSIUM.toml

```
# SYMPOSIUM.toml  
name = "toasty"  
crates = ["toasty"]  
  
[[skills]]  
source = "crate"
```



Packaging up skills with your crate



symposium-dev/recommendations

```
toasty/  
  SYMPOSIUM.toml
```

```
# SYMPOSIUM.toml  
name = "toasty"  
crates = ["toasty"]  
  
[[skills]]  
source = "crate"
```



tokio-rs/toasty

```
skills/  
  create-model/  
    SKILL.md
```

Packaging up skills with your crate



symposium-dev/recommendations

toasty/
SYMPOSIUM.toml

```
# SYMPOSIUM.toml
name = "toasty"
crates = ["toasty"]

[[skills]]
source = "crate"
```



tokio-rs/toasty

skills/
create-model/
SKILL.md

```
# SKILL.md
---
description: Create a model using the ...
---
...
```

Packaging up skills with your crate



symposium-dev/recommendations

toasty/
SYMPOSIUM.toml



```
# SYMPOSIUM.toml
name = "toasty"
crates = ["toasty"]

[[skills]]
source = "crate"
```

Temporary restriction
for security reasons



tokio-rs/toasty

skills/
create-model/
SKILL.md

```
# SKILL.md
---
description: Create a model using the ...
---
...
```

Security implications

Letting dependencies install `skill` introduces a new threat vector.

But not fundamentally different from `build.rs` or other hooks.

Packaging up MCP servers in the central repository



symposium-dev/recommendations

```
my-crate/  
SYMPOSIUM.toml
```

```
# SYMPOSIUM.toml  
name = "my-crate"  
crates = ["my-crate"]  
  
[[installations]]  
name = "my-crate-mcp"  
source = "cargo"  
crate = "my-crate-helper"  
version = "1.0.0"  
args = ["--mcp"]  
  
[[mcp-server]]  
name = "my-crate"  
command = "my-crate-mcp"
```

Packaging up MCP servers in the central repository



symposium-dev/recommendations

my-crate/
SYMPOSIUM.toml

Something installable



```
# SYMPOSIUM.toml
name = "my-crate"
crates = ["my-crate"]

[[installations]]
name = "my-crate-mcp"
source = "cargo"
crate = "my-crate-helper"
version = "1.0.0"
args = ["--mcp"]

[[mcp-server]]
name = "my-crate"
command = "my-crate-mcp"
```

Packaging up MCP servers in the central repository



symposium-dev/recommendations

my-crate/
SYMPOSIUM.toml

Install with `cargo install`



```
# SYMPOSIUM.toml
name = "my-crate"
crates = ["my-crate"]

[[installations]]
name = "my-crate-mcp"
source = "cargo"
crate = "my-crate-helper"
version = "1.0.0"
args = ["--mcp"]

[[mcp-server]]
name = "my-crate"
command = "my-crate-mcp"
```

Packaging up MCP servers in the central repository



symposium-dev/recommendations

my-crate/
SYMPOSIUM.toml

Install and
configure mcp-server



```
# SYMPOSIUM.toml
name = "my-crate"
crates = ["my-crate"]

[[installations]]
name = "my-crate-mcp"
source = "cargo"
crate = "my-crate-helper"
version = "1.0.0"
args = ["--mcp"]

[[mcp-server]]
name = "my-crate"
command = "my-crate-mcp"
```

Packaging up hooks in the central repository



symposium-dev/recommendations

my-crate/
SYMPOSIUM.toml

```
# SYMPOSIUM.toml
name = "my-crate"
crates = ["my-crate"]

[[installations]]
name = "my-crate-helper"
source = "cargo"
crate = "my-crate-hook"
version = "1.0.0"
args = ["--hook"]

[[hooks]]
name = "check-usage"
event = "PreToolUse"
command = "my-crate:hooks"
```

Packaging up hooks in the central repository



symposium-dev/recommendations

my-crate/
SYMPOSIUM.toml

"Hook" installation



```
# SYMPOSIUM.toml
name = "my-crate"
crates = ["my-crate"]

[[installations]]
name = "my-crate-helper"
source = "cargo"
crate = "my-crate-hook"
version = "1.0.0"
args = ["--hook"]

[[hooks]]
name = "check-usage"
event = "PreToolUse"
command = "my-crate:hooks"
```

Packaging up hooks in the central repository



symposium-dev/recommendations

my-crate/
SYMPOSIUM.toml

Different args



```
# SYMPOSIUM.toml
name = "my-crate"
crates = ["my-crate"]

[[installations]]
name = "my-crate-helper"
source = "cargo"
crate = "my-crate-hook"
version = "1.0.0"
args = ["--hook"]

[[hooks]]
name = "check-usage"
event = "PreToolUse"
command = "my-crate:hooks"
```

Packaging up hooks in the central repository



symposium-dev/recommendations

my-crate/
SYMPOSIUM.toml

```
# SYMPOSIUM.toml
name = "my-crate"
crates = ["my-crate"]

[[installations]]
name = "my-crate-helper"
source = "cargo"
crate = "my-crate-hook"
version = "1.0.0"
args = ["--hook"]

[[hooks]]
name = "check-usage"
event = "PreToolUse"
command = "my-crate:hooks"
```

Event(s) to accept



Zooming out -- remember this?

"What library do you recommend for working with sqlite in Rust?"

My usual recommendation:

Situation	Reach for
CLI tool / simple script	<code>rusqlite</code>
Async service (Tokio)	<code>sqlx</code>
Complex domain model, migrations-first	<code>diesel</code>
Async + ORM ergonomics	<code>sea-orm</code>

For the record, these too are all excellent libraries!

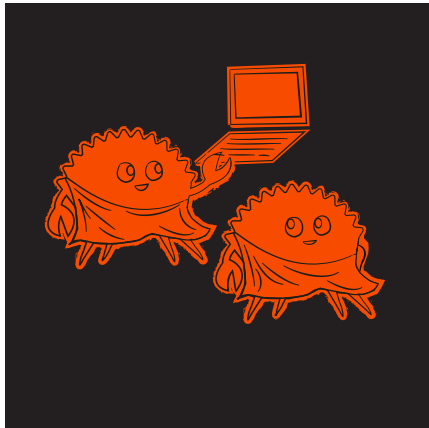
Getting started

What crates should I use
for my new network service?



Getting started

What crates should I use for my new network service?



Check out the battery packs!

The opinionated-network-service is probably just what you need!

Battery pack



Curated crate(s)

Battery pack



Curated crate(s)



Documentation &
Examples

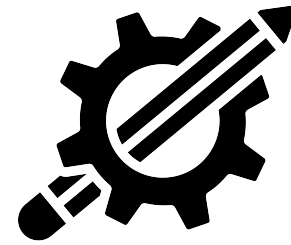
Battery pack



Curated crate(s)



Documentation &
Examples

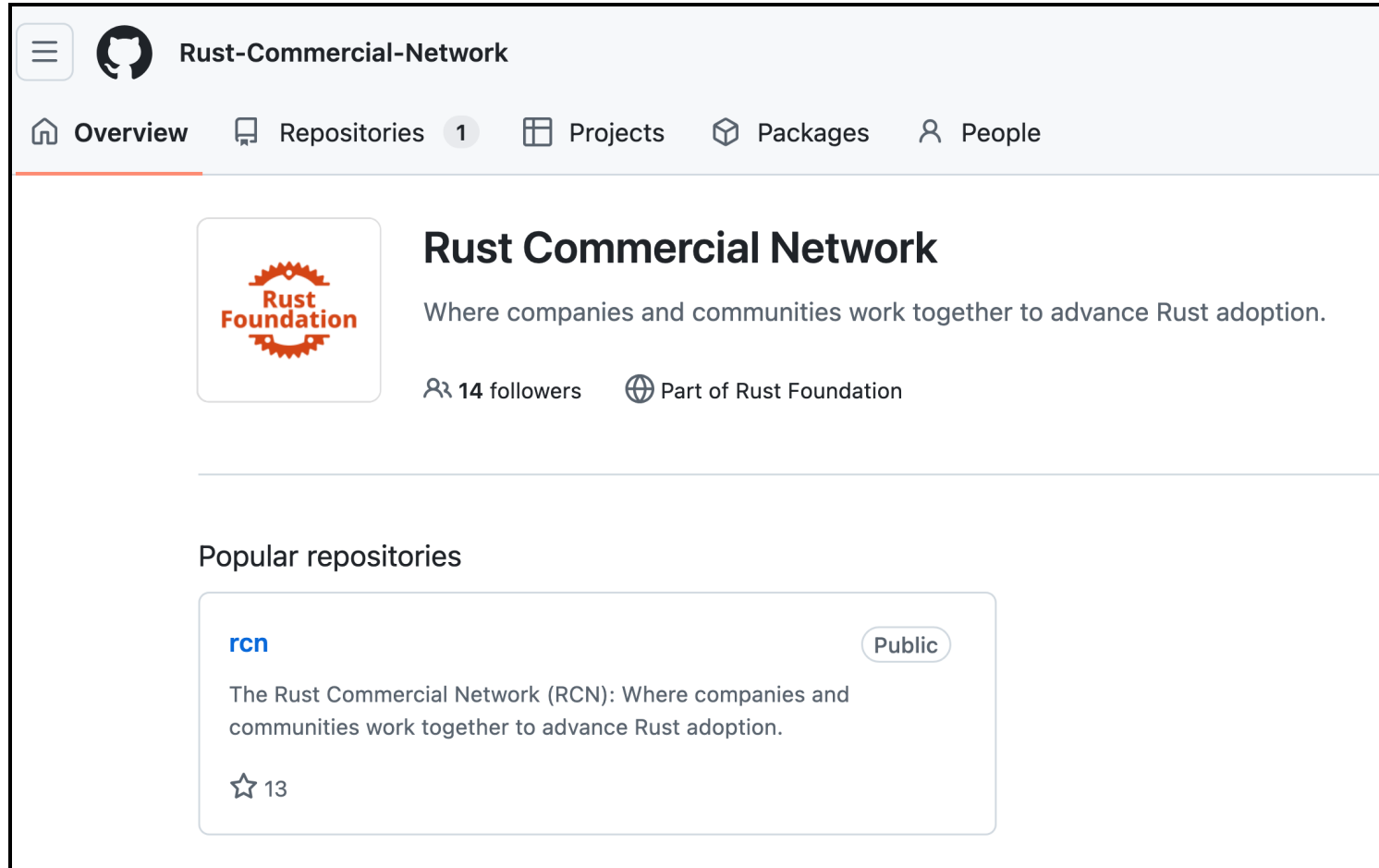


Skills

Example: Opinionated network service

- Documents key libraries
- Identifies common performance footguns or mistakes
- Integrates profiling

Rust Commercial Network



The screenshot shows the GitHub profile page for the Rust Commercial Network. At the top, there is a navigation bar with a hamburger menu icon, the GitHub logo, and the text "Rust-Commercial-Network". Below this is a secondary navigation bar with icons and labels for "Overview", "Repositories 1", "Projects", "Packages", and "People". The main content area features a profile card with the Rust Foundation logo, the name "Rust Commercial Network", a description "Where companies and communities work together to advance Rust adoption.", "14 followers", and a globe icon with the text "Part of Rust Foundation". Below the profile card is a section titled "Popular repositories" containing a single repository card for "rcn", which is public and has 13 stars.

Rust Commercial Network

Where companies and communities work together to advance Rust adoption.

14 followers Part of Rust Foundation

Popular repositories

rcn Public

The Rust Commercial Network (RCN): Where companies and communities work together to advance Rust adoption.

☆ 13

Takeaways

- Rust + AI = marshmallows + chocolate 🍫
- Symposium + Rust + AI = **s'mores** 😊

Takeaways

- Rust + AI = marshmallows + chocolate 🍫
- Symposium + Rust + AI = **s'mores** 😊
- Portability across agents today is **hard** 😞
- Agent Client Protocol (ACP) **could solve it** 🙌

Symposium provides for interoperability

As a library author, write one set of extensions that work across agents...

Symposium provides for interoperability

As a library author, write one set of extensions that work across agents...

...good luck with that.

Even AGENTS.md is not universally supported

Even AGENTS.md is not universally supported

AGENTS.md

A simple, open format for guiding coding agents, used by over [60k open-source projects](#).

Think of AGENTS.md as a **README for agents**: a dedicated, predictable place to provide the context and instructions to help AI coding agents work on your project.

[Explore Examples](#)

[View on GitHub](#)

```
# AGENTS.md

## Setup commands
- Install deps: `pnpm install`
- Start dev server: `pnpm dev`
- Run tests: `pnpm test`

## Code style
- TypeScript strict mode
- Single quotes, no semicolons
- Use functional patterns where possible
```

One AGENTS.md works across many agents

Your agent definitions are compatible with a growing ecosystem of AI coding agents and tools:



Amp



Factory



Windsurf
from Cognition



Phoenix

Agents



Ona



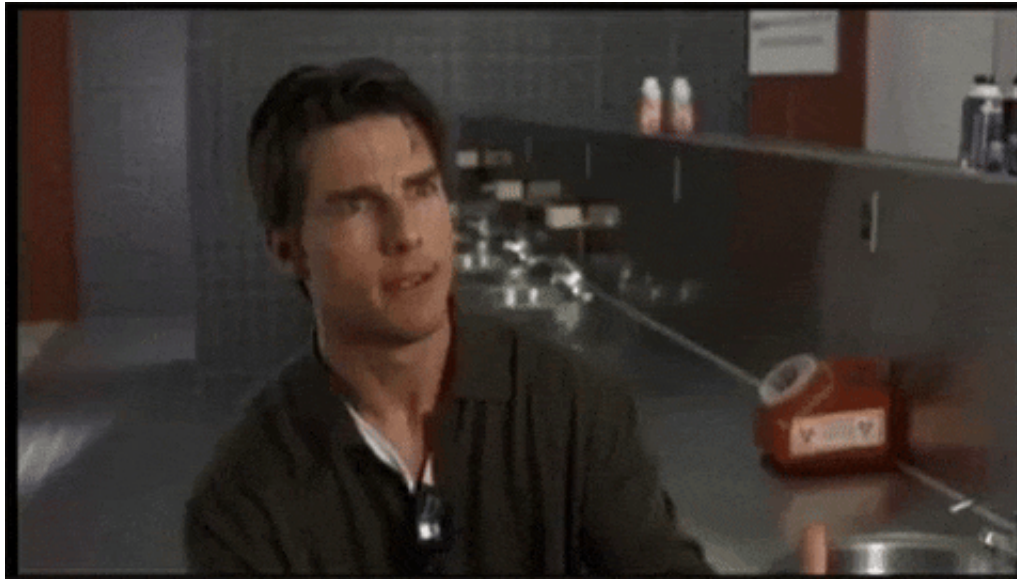
opencode



Devin
from Cognition

[View all supported agents](#)

Niko to vendors...



Symposium does the best we can



Library: provides extensions

- MCP servers
- Skills
- Hooks

Symposium does the best we can



User: picks agent

- Claude Code
- Codex
- OpenCode
- Gemini
- etc



Library: provides extensions

- MCP servers
- Skills
- Hooks

Symposium does the best we can



User: picks agent

- Claude Code
- Codex
- OpenCode
- Gemini
- etc

Symposium: adapts

- Load skills into the right directories for the agent user chose
- Converts between hook formats

Library: provides extensions

- MCP servers
- Skills
- Hooks

Skills are kinda annoying

```
$ cargo agents init
Setting up symposium for your user account.

Which agents do you use?
(space to select, enter to confirm):
  [ ] Claude Code
> [x] Codex CLI
  [ ] GitHub Copilot
  ...
```

Skills are kinda annoying

```
$ cargo agents init
Setting up symposium for your user account.

Which agents do you use?
(space to select, enter to confirm):
 [ ] Claude Code
> [x] Codex CLI
 [ ] GitHub Copilot
 ...
```


```
.agents/
  create-model/
    .gitignore
    SKILL.md
```

Skills are kinda annoying

```
$ cargo agents init
Setting up symposium for your user account.

Which agents do you use?
(space to select, enter to confirm):
 [ ] Claude Code
> [x] Codex CLI
 [ ] GitHub Copilot
 ...
```

```
.agents/
  create-model/
    .gitignore
    SKILL.md
```



```
# .gitignore
*
```

Skills are kinda annoying

```
$ cargo agents init
Setting up symposium for your user account.

Which agents do you use?
(space to select, enter to confirm):
> [x] Claude Code
   [x] Codex CLI
   [ ] GitHub Copilot
   ...
```

Skills are kinda annoying

```
$ cargo agents init  
Setting up symposium for your user account.
```

```
Which agents do you use?
```

```
(space to select, enter to confirm):
```

- [x] Claude Code
- [x] Codex CLI
- [] GitHub Copilot
- ...

Skills are kinda annoying

```
$ cargo agents init
Setting up symposium for your user account.

Which agents do you use?
(space to select, enter to confirm):
> [x] Claude Code
   [x] Codex CLI
   [ ] GitHub Copilot
   ...
```

```
.agents/
  create-model/
    .gitignore
    SKILL.md
```


Skills are kinda annoying

```
$ cargo agents init
Setting up symposium for your user account.

Which agents do you use?
(space to select, enter to confirm):
> [x] Claude Code
  [x] Codex CLI
  [ ] GitHub Copilot
  ...
```

```
.agents/
  create-model/
  .gitignore
  SKILL.md
```

```
.claude/
  create-model/
  .gitignore
  SKILL.md
```



Hooks are a total mess

- Every tool supports different hooks
- Every tool has a different *format* for hooks

Symposium approach



```
[[hooks]]
```

```
name = "check-usage"
```

```
event = "PreToolUse"
```

```
command = "my-crate:hooks"
```

- Symposium hook translates from agent into common format
- Response translated back

Symposium approach



```
[[hooks]]
name = "check-usage"
event = "PreToolUse"
command = "my-crate:hooks"
```

Specialized
variants



```
[[hooks]]
name = "check-usage"
event = "PreToolUse"
command = "my-crate:hooks"
format = "codex"
```



```
[[hooks]]
name = "check-usage"
event = "PreToolUse"
command = "my-crate:hooks"
format = "claude"
```

- Symposium hook translates from agent into common format
- Response translated back
- Preferred when using Codex
- Preferred when using Claude Code

There has got to be a better way

Existing extension mechanisms are **ad-hoc**:

- Hacky to virtualize (skills)
- Non-portable (hooks)
- Limited (MCP servers, hooks, skills)

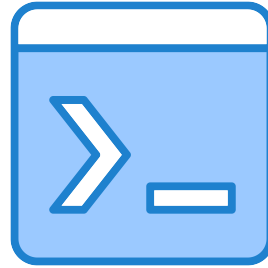
..and this is where the Agent Client Protocol (ACP) comes
in



What is the Agent Client Protocol (ACP)?



User

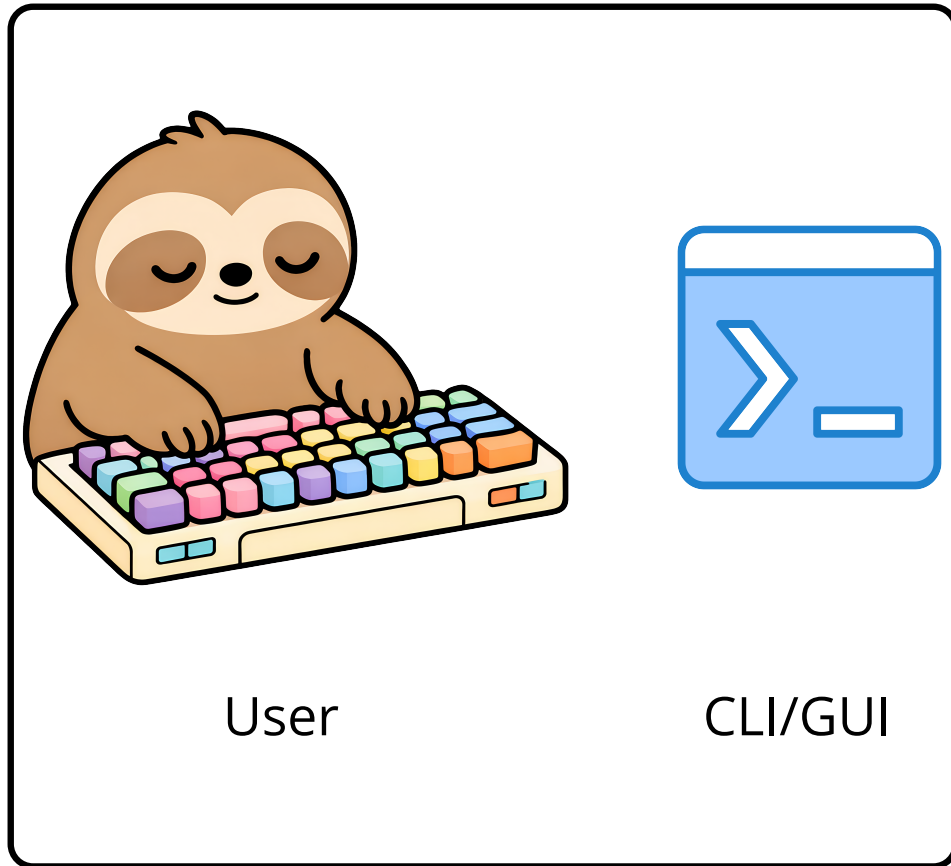


CLI/GUI



Agent

What is the Agent Client Protocol (ACP)?

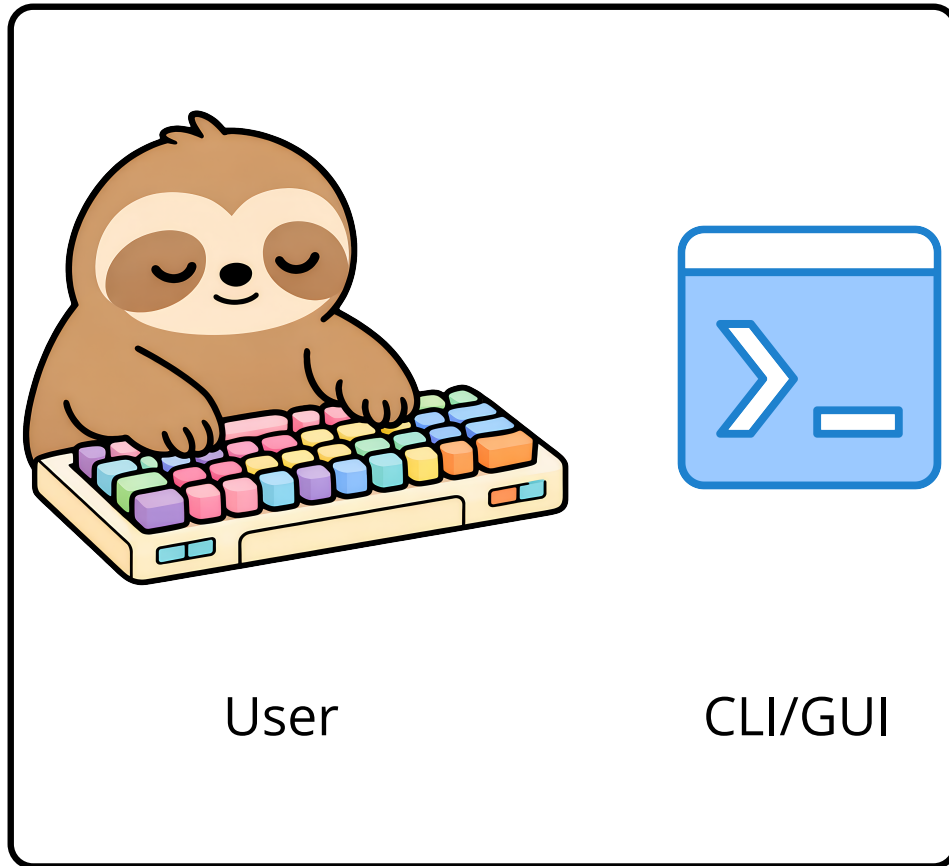


Client



Agent

What is the Agent Client Protocol (ACP)?



User

CLI/GUI

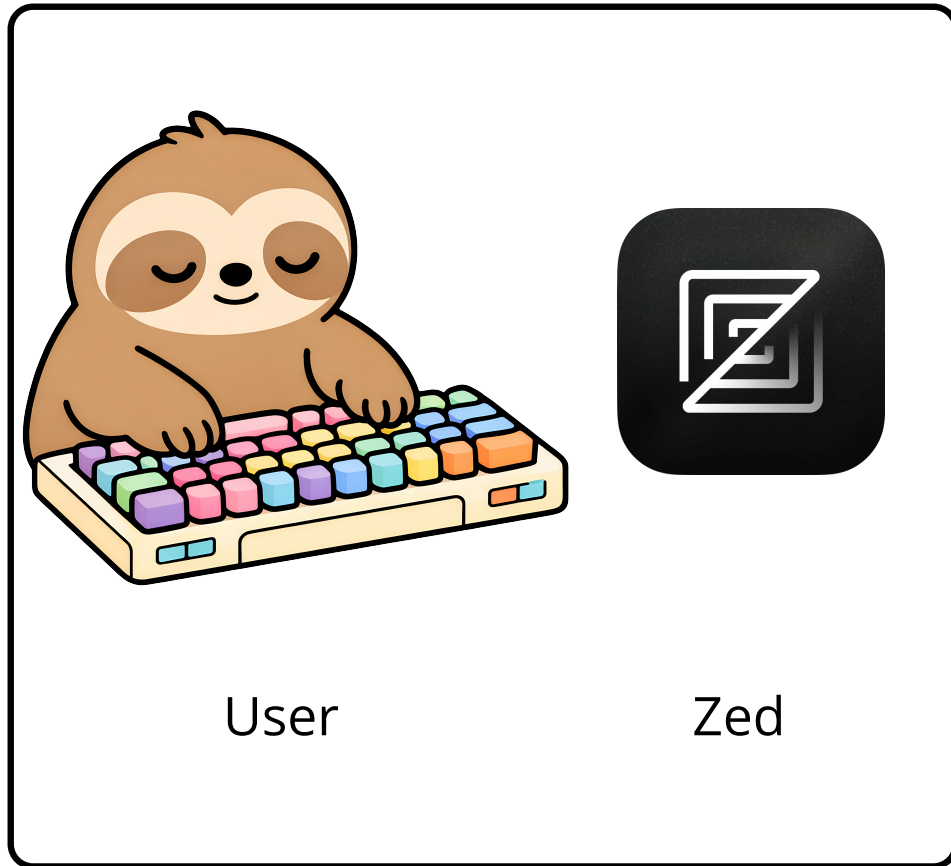
Client

JSON-RPC



Agent

Original idea: 1 editor, N agents



Client



Agent 1



Agent 2



Agent 3

Agents

The screenshot shows the GitHub interface for the 'registry' repository. At the top, the repository name 'registry' is displayed with a 'Public' badge. To the right, there are buttons for 'Edit Pins' and 'Watch 5'. Below this, the current branch is 'main', with '15 Branches' and '750 Tags' listed. A search bar 'Go to file' and a 'Code' button are also visible. The main content area shows a commit by 'acp-release[bot]' with the message 'docs: update protocol matrix for 2026-05-31' and a green checkmark. The commit hash is '88a12a1' and it was made '16 hours ago'. Below the commit, a list of files is shown, each with a blue folder icon, a description of the change, and the time since the commit.

File	Commit Message	Time
.github	Bump actions/github-script (#328)	3 days ago
.protocol-matrix	docs: update protocol matrix for 2026-05-31	16 hours ago
agoragentic-acp	Update all agent versions (#279)	28 days ago
amp-acp	Update 2 agents to latest versions	3 days ago
auggie	Update 14 agents to latest versions	5 days ago
autohand	Add website field to schema (#187)	2 months ago
claude-acp	Update 6 agents to latest versions	2 days ago
cline	Update 2 agents to latest versions	2 days ago
codebuddy-code	Update codebuddy-code to 2.100.1	yesterday
codex-acp	Update 14 agents to latest versions	5 days ago
cortex-code	(fix): make paths relative (#282)	28 days ago
corust-agent	Update 3 agents to latest versions	3 weeks ago

Editors

Zed



Intellij IDEs



Emacs, neovim
(via extensions)



Editors

Zed



Intellij IDEs



Obsidian



Emacs, neovim
(via extensions)



Editors

Zed



Emacs, neovim
(via extensions)



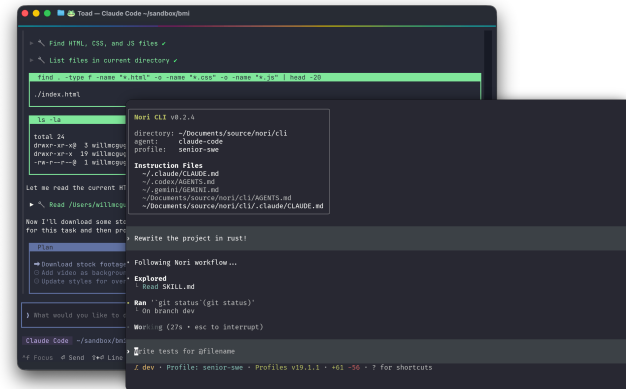
Intellij IDEs



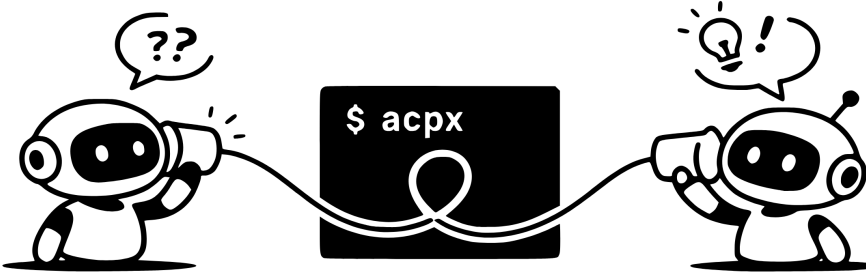
Obsidian



CLIs like Toad, Nori



Beyond editors: ACPX



acpx

npm v0.10.0 downloads 4.3M/month CI passing License MIT node >=22.13.0


⚠️ `acpx` is in alpha and the CLI/runtime interfaces are likely to change. Anything you build downstream of this might break until it stabilizes.

ACP coverage status: see [ACP Spec Coverage Roadmap](#).

Your agents love acpx! 🤖❤️ They hate having to scrape characters from a PTY session 🤖

`acpx` is a headless CLI client for the [Agent Client Protocol \(ACP\)](#), so AI agents and orchestrators can talk to coding agents over a structured protocol instead of PTY scraping.

One command surface for Pi, OpenClaw ACP, Codex, Claude, and other ACP-compatible agents. Built for agent-to-agent communication over the command line.



Scripting your agent

The screenshot shows the GitHub repository page for 'retcon' by user 'nikomatsakis'. The repository is public and has 10 stars, 0 forks, and 0 watches. The main branch is 'main', with 3 other branches and 1 tag. The repository contains several files and folders, including .claude, .github/workflows, md, src, .gitignore, CHANGELOG.md, Cargo.lock, Cargo.toml, and LICENSE-APACHE. The most recent commit is by 'nikomatsakis' with the message 'Use git fixup commits instead of WIP naming convention', dated 'last month' and containing 37 commits. The repository also has a release 'v1.1.0' marked as 'Latest' from 'last month'.

nikomatsakis / retcon

Type to search

<> Code Issues Pull requests 1 Agents Actions Projects Wiki Security and quality Insights Settings

nikomatsakis Public

Pin Watch 0 Fork 0 Star 10

main 3 Branches 1 Tag

Go to file Add file Code

nikomatsakis Use git fixup commits instead of WIP naming convention fb23487 · last month 37 Commits

.claude	refactor: replace ratatui TUI with scrollbar-based termina...	2 months ago
.github/workflows	feat: add cross-platform binary releases for retcon-proxy	5 months ago
md	refactor: rename project from pravda to retcon	5 months ago
src	Use git fixup commits instead of WIP naming convention	last month
.gitignore	initial commit	5 months ago
CHANGELOG.md	chore: release v2.0.0	last month
Cargo.lock	chore: release v2.0.0	last month
Cargo.toml	chore: bump to 1.1	last month
LICENSE-APACHE	chore: prepare to publish	5 months ago

About

Because you knew what you were doing the whole time.

nikomatsakis.github.io/retcon/

- Readme
- Apache-2.0, MIT licenses found
- Activity
- 10 stars
- 0 watching
- 0 forks

Releases 1

v1.1.0 Latest last month

Scripting your agent

```
// First pass: extract and apply changes
let extract_result: ExtractResult = d
  .think()
  .textln("# Task: Extract changes for a git commit")
  .textln("")
  .textln("You are reconstructing clean git history from a messy branch.")
  .textln("Your job is to extract ONLY the changes relevant to this commit.")
  .text(&resolution_context)
  .text(interrupted_context)
  .textln("")
  .textln("## Commit to create")
```

Scripting your agent

```
// First pass: extract and apply changes
let extract_result: ExtractResult = d
    .think()
    .textln("# Task: Extract changes for a git commit")
    .textln("")
    .textln("You are reconstructing clean git history from a messy branch.")
    .textln("Your job is to extract ONLY the changes relevant to this commit.")
    .text(&resolution_context)
    .text(interrupted_context)
    .textln("")
    .textln("## Commit to create")
```

Scripting your agent

```
// First pass: extract and apply changes
let extract_result: ExtractResult = d
  .think()
  .textln("# Task: Extract changes for a git commit")
  .textln("")
  .textln("You are reconstructing clean git history from a messy branch.")
  .textln("Your job is to extract ONLY the changes relevant to this commit.")
  .text(&resolution_context)
  .text(interrupted_context)
  .textln("")
  .textln("## Commit to create")
```



Scripting your agent

```
// First pass: extract and apply changes
let extract_result: ExtractResult = d
  .think()
  .textln("# Task: Extract changes for a git commit")
  .textln("")
  .textln("You are reconstructing clean git history from a messy branch.")
  .textln("Your job is to extract ONLY the changes relevant to this commit.")
  .text(&resolution_context)
  .text(interrupted_context)
  .textln("")
  .textln("## Commit to create")
```

Scripting your agent

```
// First pass: extract and apply changes
```

```
let extract_result: ExtractResult = d
```

```
.think()
```

```
.textl
```

```
.textl
```

```
.textl
```

```
.textl
```

```
.textl
```

```
.textl
```

```
.textl
```

```
.textl
```

```
#[derive(Debug, Serialize, Deserialize, JsonSchema)]
```

```
struct ExtractResult {
```

```
    /// Whether changes were successfully applied
```

```
    applied_changes: bool,
```

```
}
```

Scripting your agent

```
// First pass: extract and apply changes
```

```
let extract_result: ExtractResult = d
```

```
.think()
```

```
.textl
```

```
.textl
```

```
.textl
```

```
.textl
```

```
.textl
```

```
.textl
```

```
.textl
```

```
.textl
```

```
#[derive(Debug, Serialize, Deserialize, JsonSchema)]
```

```
struct ExtractResult {
```

```
    /// Whether changes were successfully applied
```

```
    applied_changes: bool,
```

```
}
```

"Agent mods" (work in progress)



Client

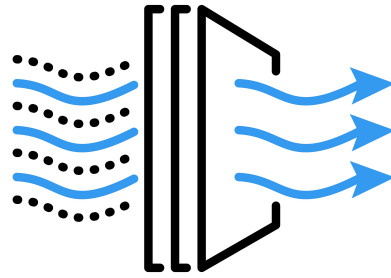


Agent

"Agent mods" (work in progress)



Client



Agent mod



Agent

Takeaways

- Rust + AI = marshmallows + chocolate 🍫

Takeaways

- Rust + AI = marshmallows + chocolate 🍫
- Symposium + Rust + AI = **s'mores** 😊

Takeaways

- Rust + AI = marshmallows + chocolate 🍫
- Symposium + Rust + AI = **s'mores** 😊
- Portability across agents today is **hard** 😡

Takeaways

- Rust + AI = marshmallows + chocolate 🍫
- Symposium + Rust + AI = **s'mores** 😊
- Portability across agents today is **hard** 😞
- Agent Client Protocol (ACP) **could solve it** 🙌

Takeaways

- Rust + AI = marshmallows + chocolate 🍫
- Symposium + Rust + AI = **s'mores** 😊
- Portability across agents today is **hard** 😞
- Agent Client Protocol (ACP) **could solve it** 🙌

Try symposium

```
$ cargo install symposium  
$ cargo agents init
```

Curious about ACP?

<https://agentclientprotocol.com>

Extra: Per-project setup



nikomatsakis @nikomatsakis.com · 1mo

I got some questions about what people choose to check into git when doing agentic development. Do you check-in [AGENTS.md](#)? .claude directories or files? Hooks? I also asked in [rustaceans-ai.zulipchat.com](#) -- if you do Rust coding with agents, you should join that.

Extra: Per-project setup



nikomatsakis @nikomatsakis.com · 1mo

I got some questions about what people choose to check into git when doing agentic development. Do you check-in [AGENTS.md](#)? `.claude` directories or files? Hooks? I also asked in [rustaceans-ai.zulipchat.com](#) -- if you do Rust coding with agents, you should join that.

CLAUDE.md + the whole `.claude/` directory (agent roles, memory protocol) — yes. Hooks too if they enforce task routing. Skipping these creates per-clone setup debt that compounds fast. We open-sourced our starter kit with defaults for all of it: [github.com/ultrathink-art/agent-architect-kit](#)

7:35 PM · Apr 7, 2026

2 likes



Extra: Per-project setup



nikomatsakis @nikomatsakis.com · 1mo

I got some questions about what people choose to check into git when doing agentic development. Do you check-in [AGENTS.md](#)? `.claude` directories or files? Hooks? I also asked in [rustaceans-ai.zulipchat.com](#) -- if you do Rust coding with agents, you should join that.



CLAUDE.md + the whole `.claude/` directory (agent roles, memory protocol) — yes. Hooks too if they enforce task routing. Skipping these creates per-clone setup debt that compounds fast. We open-sourced our starter kit with defaults for all of it: [github.com/ultrathink-art/agent-architect-kit](#)

7:35 PM · Apr 7, 2026

2 likes

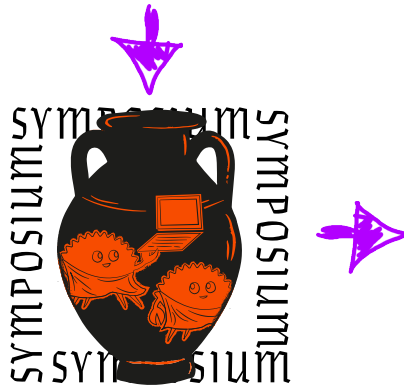


Per-project setup, portable

```
my-project/  
  .agents/  
    skills/  
      skill-for-my-project/  
        SKILL.md
```

Per-project setup, portable

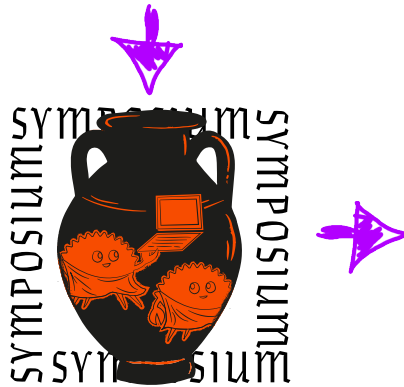
```
my-project/  
  .agents/  
    skills/  
      skill-for-my-project/  
        SKILL.md
```



```
my-project/  
  .agents/  
    skills/  
      skill-for-my-project/  
        SKILL.md  
      skill-from-my-dependencies/  
        .gitignore # "*"   
        SKILL.md  
  .claude/  
    skills/  
      skill-for-my-project/  
        .gitignore # "*"   
        SKILL.md  
      skill-from-my-dependencies/  
        .gitignore # "*"   
        SKILL.md
```

Per-project setup, portable

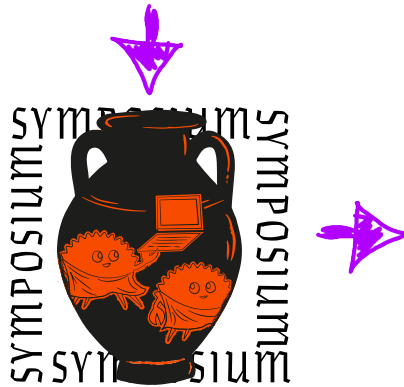
```
my-project/  
  .agents/  
    skills/  
      skill-for-my-project/  
        SKILL.md
```



```
my-project/  
  .agents/  
    skills/  
      skill-for-my-project/  
        SKILL.md  
      skill-from-my-dependencies/  
        .gitignore # "*"   
        SKILL.md  
  .claude/  
    skills/  
      skill-for-my-project/  
        .gitignore # "*"   
        SKILL.md  
      skill-from-my-dependencies/  
        .gitignore # "*"   
        SKILL.md
```

Per-project setup, portable

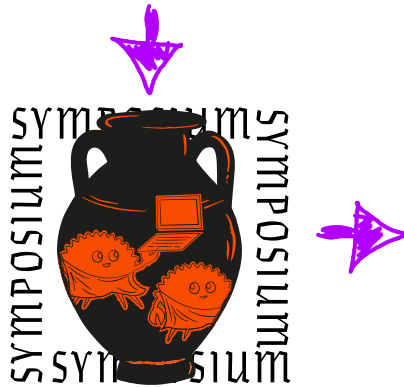
```
my-project/  
  .agents/  
    skills/  
      skill-for-my-project/  
        SKILL.md
```



```
my-project/  
  .agents/  
    skills/  
      skill-for-my-project/  
        SKILL.md  
      skill-from-my-dependencies/  
        .gitignore # "*"   
        SKILL.md  
  .claude/  
    skills/  
      skill-for-my-project/  
        .gitignore # "*"   
        SKILL.md  
      skill-from-my-dependencies/  
        .gitignore # "*"   
        SKILL.md
```

Per-project setup, portable

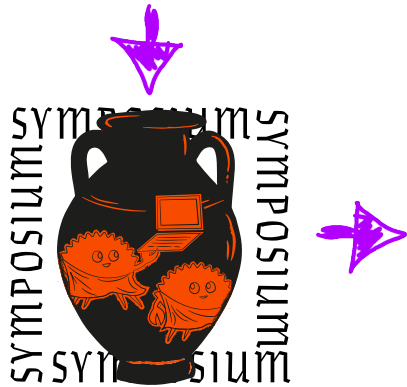
```
my-project/  
  .agents/  
    skills/  
      skill-for-my-project/  
        SKILL.md
```



```
my-project/  
  .agents/  
    skills/  
      skill-for-my-project/  
        SKILL.md  
      skill-from-my-dependencies/  
        .gitignore # "*"   
        SKILL.md  
  .claude/  
    skills/  
      skill-for-my-project/  
        .gitignore # "*"   
        SKILL.md  
      skill-from-my-dependencies/  
        .gitignore # "*"   
        SKILL.md
```

Per-project setup, portable

```
my-project/  
  .agents/  
    skills/  
      skill-for-my-project/  
        SKILL.md
```



```
my-project/  
  .agents/  
    skills/  
      skill-for-my-project/  
        SKILL.md  
      skill-from-my-dependencies/  
        .gitignore # "*"   
        SKILL.md  
  .claude/  
    skills/  
      skill-for-my-project/  
        .gitignore # "*"   
        SKILL.md  
      skill-from-my-dependencies/  
        .gitignore # "*"   
        SKILL.md
```