Rewrite it in Rust (Or Don’t)

Ryan Waskiewicz
https://ducktyped.dev

Evolving JavaScript Tooling

- JavaScript ecosystem is constantly evolving
- Many ecosystems have chased being “faster”
- JavaScript is no exception
  - Bundlers
  - Linters/formatters
  - Type systems




Speed as a Development Value

- Currently, there's a shifted focus on rewriting languages, tools in non-JavaScript languages
- *Speed* is overwhelmingly used in marketing materials
- Speed is a *value* - one that authors believe you share

…we've begun work on a native port of the TypeScript compiler and tools. The native implementation will **drastically improve editor startup, reduce most build times by 10x, and substantially reduce memory usage**.

- Anders Hejlsberg, TypeScript-Go Announcement, March 2025

Biome is a fast formatter for *JavaScript*, *TypeScript*, *JSX*, *TSX*, *JSON*, *HTML*, *CSS* and *GraphQL* that scores **97% compatibility with *Prettier***, **saving CI and developer time**.

- https://biomejs.dev/, Feb 2026

We are building a unified JavaScript toolchain to **make web developers more productive than ever before.**

- https://voidzero.dev/, Feb 2026

Oxlint is built for large repositories and CI environments. Its architecture removes structural bottlenecks that limit performance in ESLint.

Our benchmarks show Oxlint is **50 to 100 times faster** than ESLint.

- Oxlint Documentation, Feb 2026

Speed as a Development Value

- Tooling speed touches many aspects of the SDLC process
  - Local builds, formatting, type checking, static analysis, testing, etc.
  - Production build deployments
  - Continuous Integration (CI) workflows
  - Cloud provider billing
  - Maintenance & Sustaining engineering
  - AI/Agentic workflows (😏)

Speed as a Development Value

- Our time as people on earth is finite
- Work or personal time, do you want your tools to slow you down?
- What is the cost of these new tools?
  - Monetarily
  - Temporally
  - In Need of Contributions
- How do we evaluate these tools?

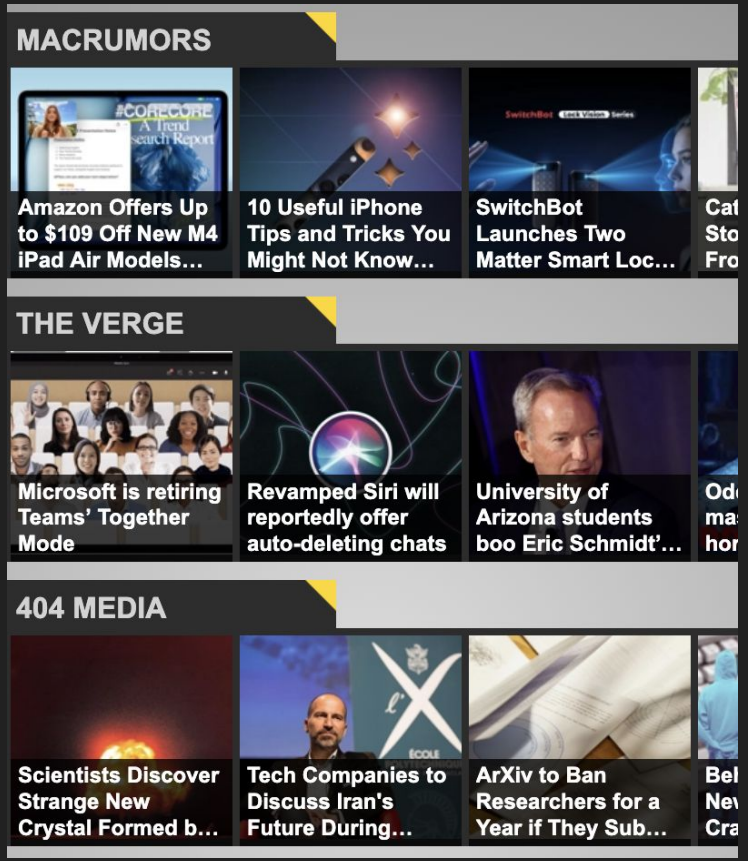
A Side Project

- Goal: Create a (Simple) RSS Reader
- RSS (Really Simple Syndication)
  - Sites push updates to a feed they own
  - People subscribe to feeds to keep up to date on sites
  - Each update (an “item”) contains a news article, blog post, etc.
    - Updates may contain links to images
- Let’s take a look at a snippet…

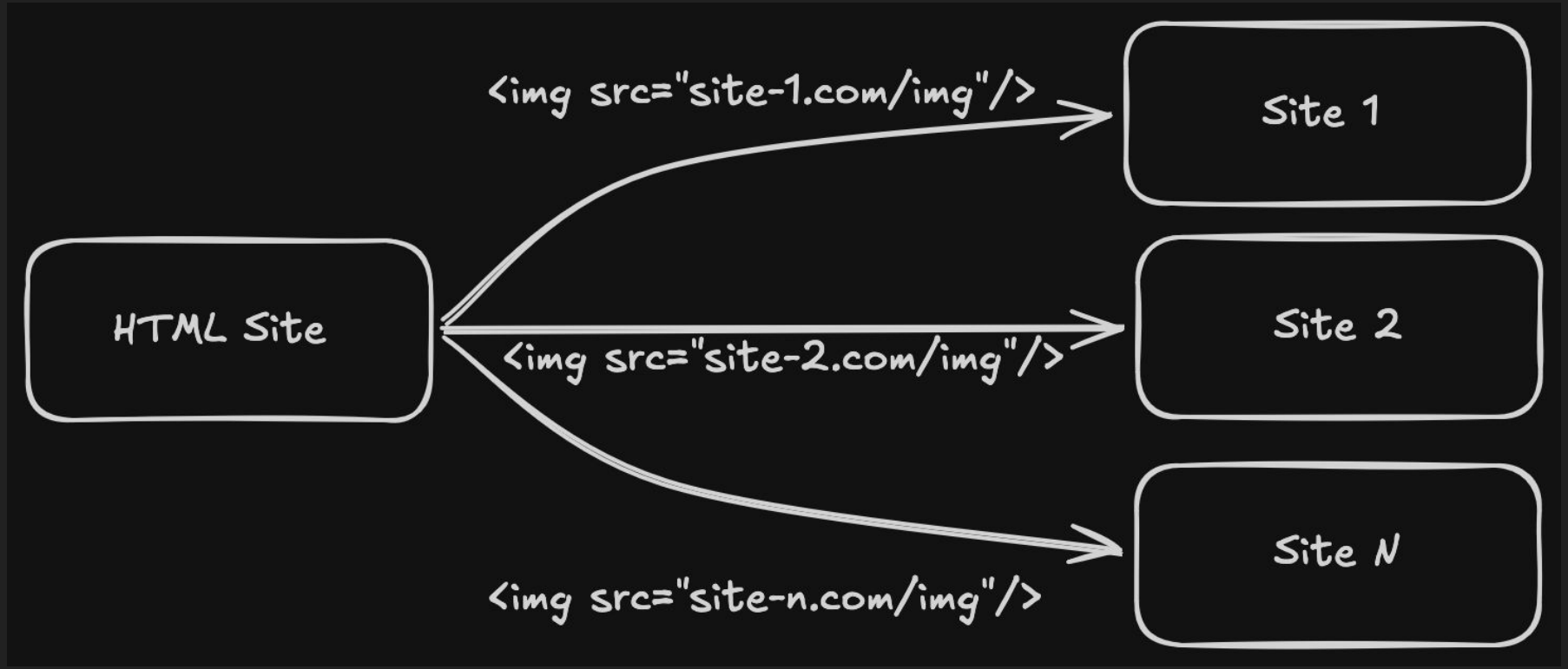
```
<item>
<title>Amazon Offers Up to $109 Off New M4 iPad Air Models This Weekend</title>
<link>https://www.macrumors.com/2026/05/17/109-off-new-m4-ipad-air/</link>
<pubDate>Sun, 17 May 2026 07:20:17 PDT</pubDate>
<description>_
    Amazon this weekend has brought back a major sale on the M4 iPad Air.
    <br/>
    <img
        src="https://images.macrumors.com/article-new/2026/03/m4-ipad-air-spring.jpeg"
        alt=""
        width="2500"
        height="1406"
        class="aligncenter size-full wp-image-1042288"/>
    <em>_
    <br/>
</item>
</description>
</item>
```

A Side Project: First Iteration

- Want:
  - Read each RSS Feed
  - Parse each feed for metadata
  - Render an HTML page
- Emulating “Pulse” from Alphonso Labs
  - Each row is a feed
  - Each square is an item/news story
  - Each item has an image from that story

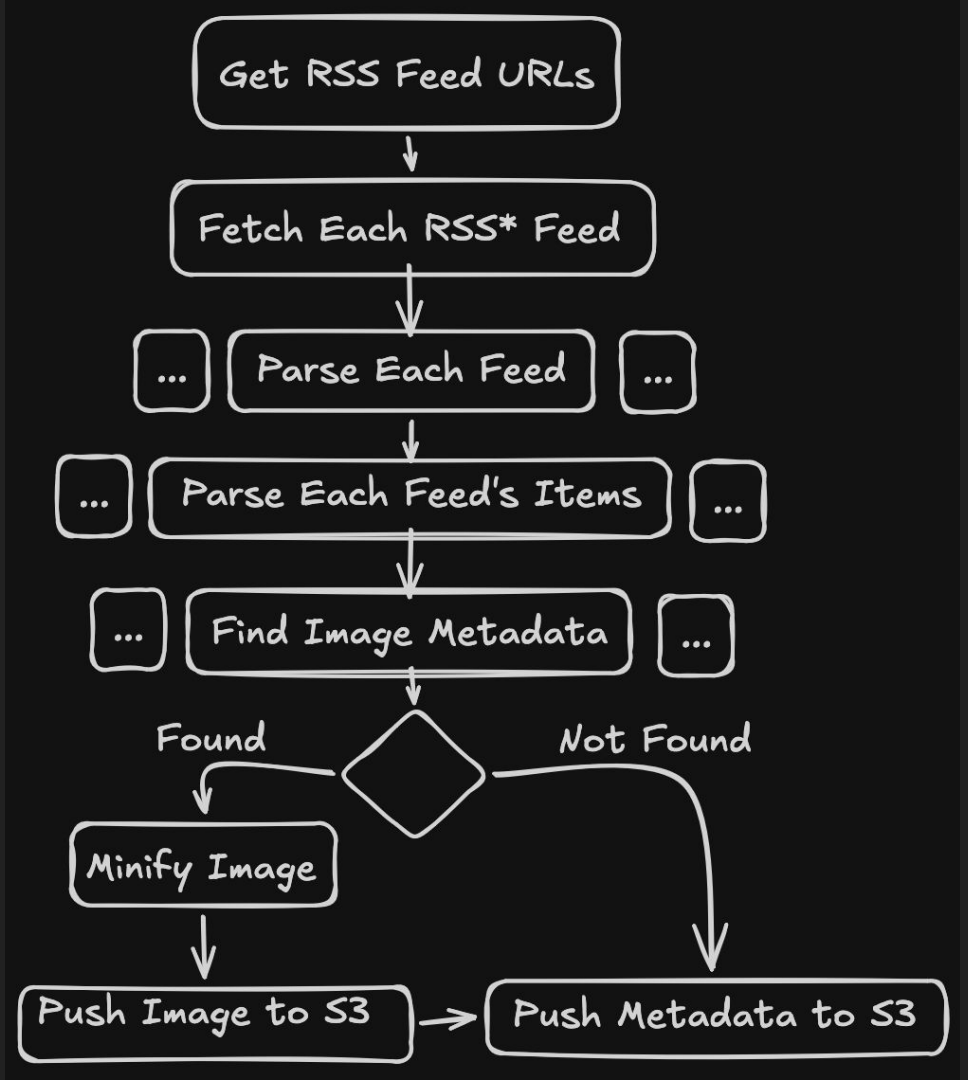


Displaying Images from Feeds

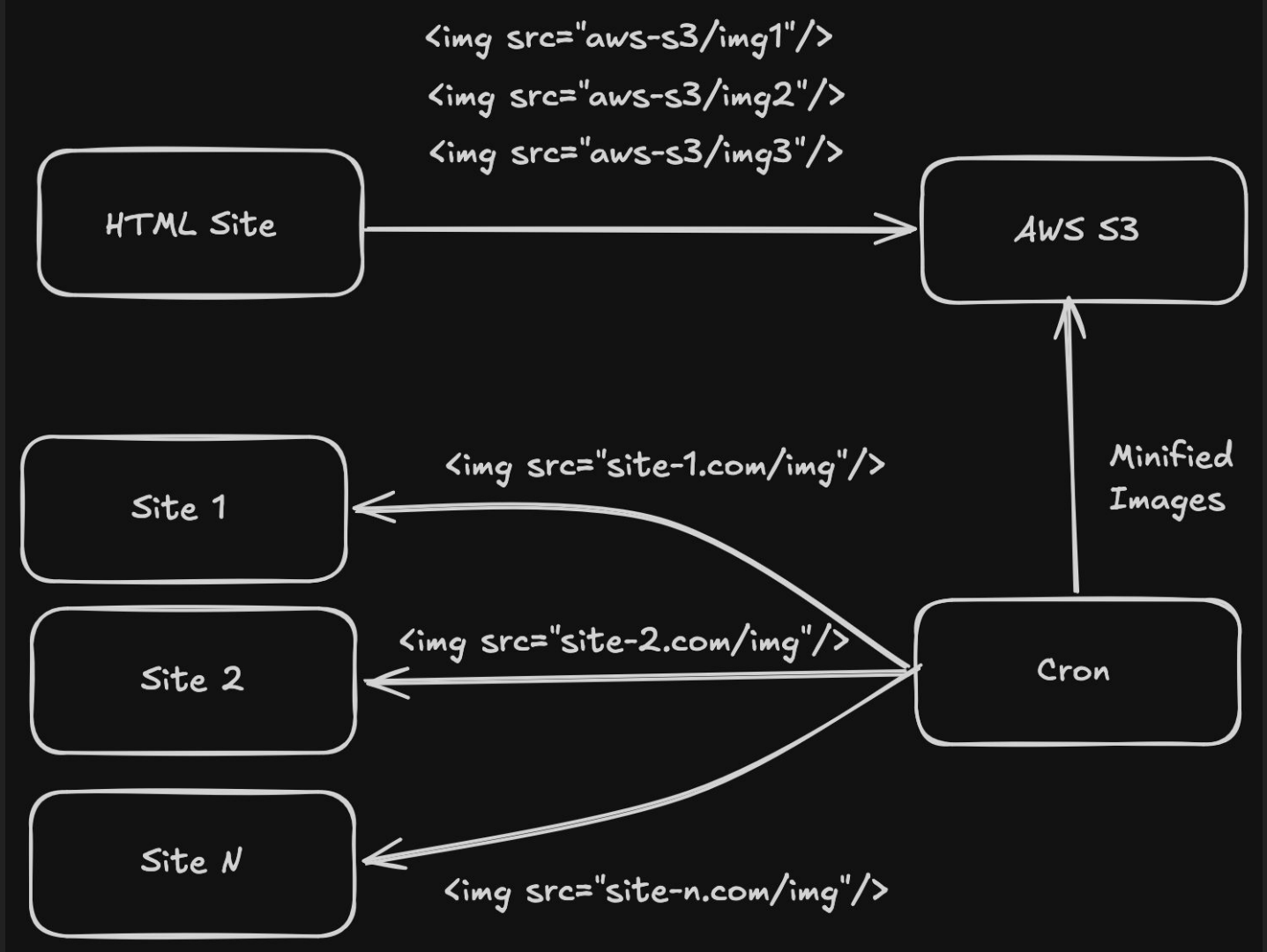


Displaying Images

- Idea! Create a Cron Job to:
  - Parse Feeds
  - Minimize Images
  - Cache Processed Images
- Initial Stack:
  - RSS Parser (TypeScript)
  - Deployed to Amazon Web Services (AWS)
    - Lambda, Cron Trigger
  - Images pushed to AWS S3



```
<item>
<title>Amazon Offers Up to $109 Off New M4 iPad Air Models This Weekend</title>
<link>https://www.macrumors.com/2026/05/17/109-off-new-m4-ipad-air/</link>
<pubDate>Sun, 17 May 2026 07:20:17 PDT</pubDate>
<description>_
    Amazon this weekend has brought back a major sale on the M4 iPad Air.
    <br/>
    <img
        src="https://images.macrumors.com/article-new/2026/03/m4-ipad-air-spring.jpeg"
        alt=""
        width="2500"
        height="1406"
        class="aligncenter size-full wp-image-1042288"/>
    <em>_
    <br/>
</item>
</description>
</item>
```



HTML Site
AWS S3
Minified Images

Site 1
Site 2

Cron
Site N

Intrusive Thought: What if I Rewrite it in Rust?

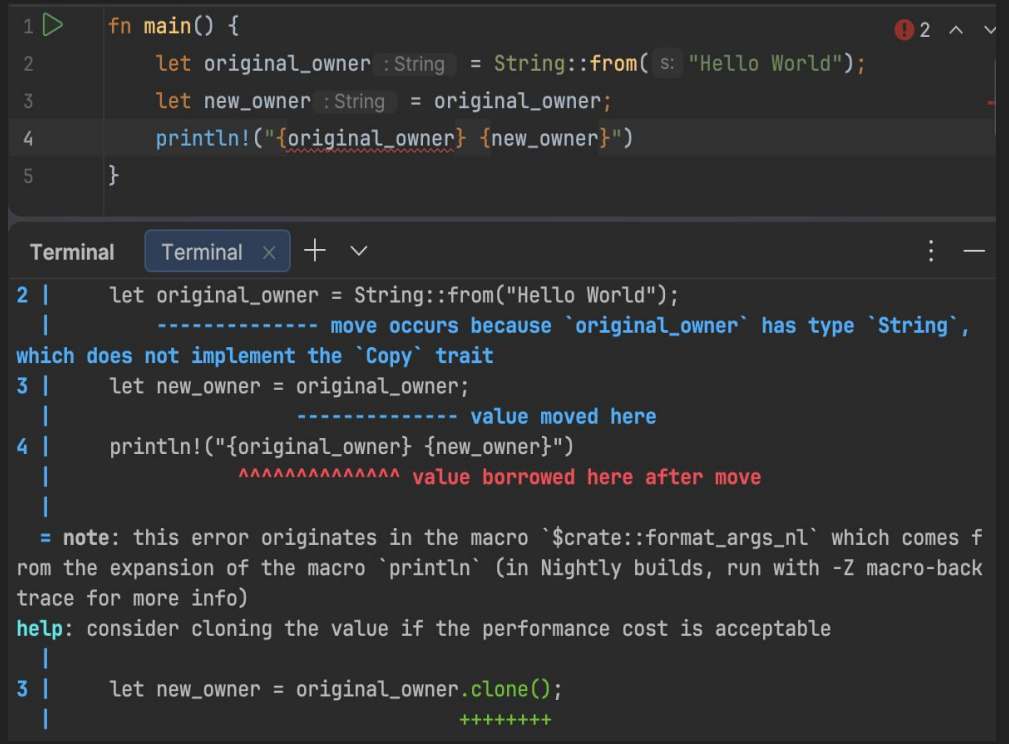
- The TypeScript version *worked*, BUT:
  - Runtime was slower than I'd like
  - NPM packages used a combination of CommonJS (CJS) and ECMAScript Modules (ESM)
- General itch to scratch
  - *Will this language improve the development experience of this project?*

NPM Dependency Pains

- NPM dependencies used a combination of CommonJS (CJS) and ECMAScript Modules (ESM)
- esbuild configuration required hacks & polyfills to properly build
  - Shim `require()` for the RSS parser, JSDom
  - Replace source code in npm packages
- Externalized `sharp` to handle platform-specific code
- Package updates became time sinks

…So I Rewrote It in Rust

- Pro: I had prior knowledge working in my favor
- I had “solved” the problem domain
- Started with a decent working knowledge of the base language
  - Minimal fighting of the borrow checker



…So I Rewrote It in Rust

- Con: Needed to find replacement libraries
  - Both for NPM packages and JavaScript behavior
- Con: CI/CD work took longer than expected
- Con: Monetary cost of running the system was unknown until I was done

Purpose	JavaScript	Rust
Async/Await	Built-In	futures, tokio
AWS	@aws-sdk/client/s3	aws-config, aws-sdk-s3, aws-smithy-mocks, lambda-runtime
Base64 Encoding	Built-In	base64
Building	esbuild, TypeScript	Built-in (cargo)
Date/Time	Built-In	chrono
HTML Decoder	he	html-escape
HTML Parsing	jsdom	scraper
HTTP Client	Built-In	reqwest
Image Processing	sharp	image, webp
Linting, Formatting	eslint(+ts), prettier	Built-In (cargo)
JSON Serialization	Built-In	serde, serde_json
Regex	Built-In	regex
RSS Parsing	rss-parser	feed-rs
Testing	Vitest + Coverage	Built-In, mockall, testcase, wiremock
Type Definitions	Various (x3)	N/A
URL Parsing	Built-In	url

Dependency Changes

- 16 Categories on Previous Slide
  - Needed to find equivalents for each category
  - Evaluated multiple alternatives in some cases
- 9 new dependencies added to backfill JavaScript native functionality
  - Shown in green rows
  - See what happens when you move away from a web-friendly language?
- 5 additional new dependencies to cover functionality of NPM packages
  - One package in NPM may require four in Rust
- 7 types of dependencies not needed
  - 4-5 JavaScript ecosystem dependencies are built-in to Rust-ecosystem (via cargo)
    - Shown in yellow rows
  - 3 are TypeScript-specific (@types packages)

Monetary Cost of Dependencies

- Renovate used for package updates
- Dependabot for security updates
- Need to review changes (time), CI needs to pass (time, money)

July 15, 2025 - November 15, 2025

	NPM
Renovate	77
Dependabot	4
Total	**81**

November 15, 2025 - February 15, 2026

	NPM	crates.io	Total
Renovate	58	53	111
Dependabot	5	2	7
Total	63	55	**118**

Number of PRs

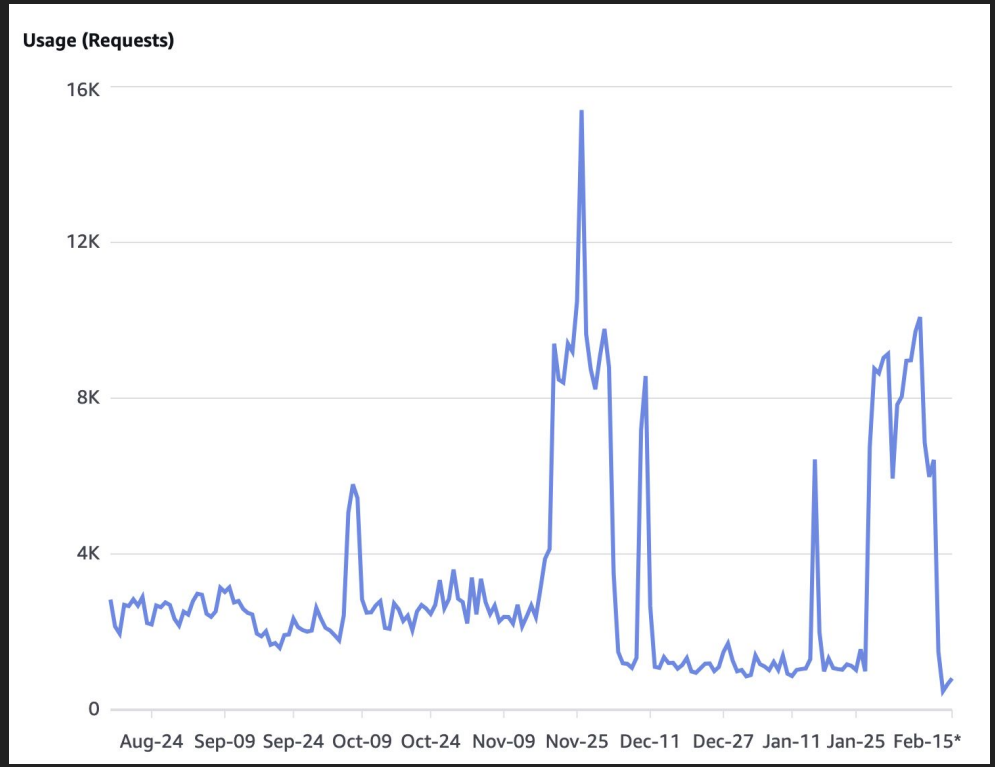
	August	September	October	November	December	January	February
Pull Requests	31	19	20	40	26	18	25
Total Minutes	240	89	168	378	360	130	173
Cost	$1.92	$0.71	$1.34	$3.02	$2.88	$0.96	$1.04
Cost per PR	$0.06	$0.04	$0.07	$0.08	$0.11	$0.05	$0.04

AWS Lambda Costs

- AWS Lambda charges based on size and duration
  - Was running 1024 MB, cost $0.0000000167/ms
- Able to swap out base image, overall image smaller
  - With Rust, able to scale down to 256 MB, cost $0.0000000021/ms
  - 8x Cheaper 🎉
- Invocation is so infrequent that cost was and still is $0.00/month
  - Optimized for nothing 🎉🎉🎉
  - Costs didn’t go up at least

AWS S3 Costs

- AWS S3 charges based on storage & ingress/egress
  - Storage - $0.023/GB
    - Bucket hovers at ~10 MB
  - Requests - $0.005 per 1000 LIST & PUT requests
    - Bursty - $0.08 to $0.16/month
- Center spike is broken caching directly caused by rewrite
- Right spike is increased traffic



Cost Summary

- Switching to Rust required time to research package equivalents to NPM
- We saw a close to a 50% increase in pull requests
  - But the cost per PR in CI was about the same
- Cloud costs stayed roughly the same

Conclusion - What is Worth It?

- It depends - I had fun
- Overall, JavaScript is probably the right language for the Cron
- Rust ecosystem led a different type of maintenance issue
- Monetary cost is *roughly* the same
- How well can a team or person learn the language, standard library, & ecosystem?
    - Not only to author, but maintain
    - This was a small, personal project
    - Sneaking under free tiers