Rust at TEMPUS

Zelda Hessler
Engineering Lead
Developer Tools Team

1

How we got here

My background
Why Rust
The Developer Tools team

2

What we’re doing

OAuth helper CLIs
An app manager CLI
Infrastructure-as-code templating tools
A Slack bot

3

What we’re looking forward to

Desktop apps for IAC templating
Using Rust libraries in non-Rust codebases

1 | How we got here

My Background

Hi. My name is Zelda Hessler

- I started my programming career as a self-taught web developer in 2016
- I have been a Rust enthusiast for about the same amount of time
- I started at Tempus in 2019 as a full stack web developer and became an Engineering Lead at the start of this year
- I am a Rust mentor in Tempus' mentorship program and I have taught classes on Rust as part of the "Tempus Board of Education" initiative
- I've hosted the Chicago Rust Meetup since 2018 and have given several talks, typically focused on Rust from a beginner's perspective
- I've created many generative art apps with Rust which you can view on my website

If it's not completely clear, I'm a bit *obsessed* with the language

Why Rust?

- Rust is a modern language with a C-like syntax that most developers will find familiar
- Rust apps can be compiled for the targets we care about (Linux, MacOS, Windows)
  - Because the apps are native, users don't need to install a runtime
- There exist great frameworks for CLI apps
  - We currently use clap and have used seahorse in the past
- Rust's error handling paradigm is great
  - In our apps, we wrap errors in more errors, each less specific than the last. Regular users only see friendly, top-level errors. Power users can easily opt-in to more detailed, technical error messages.
- Rust has first-class tooling
  - *cargo fmt* keeps everyone's code uniform
  - *cargo clippy* helps new teammates write idiomatic Rust code from day one
- Whether you're extending, refactoring, or starting a new app, once it passes compilation, you can be confident that it's not *(too)* buggy

The Developer Tools Team

The Developer Tools Team *(DevTools for short)* creates apps to help developers.
Our process involves:

- Finding the most annoying problems that Tempus developers run into
- Working with those developers to find out if those problems can be fixed with a technological solution
- Creating that solution using the tools that we think are the most appropriate
- Repeated user testing to ensure that tool not only solves the problem, but provides a great user experience
- Creating a CI/CD pipeline to build, test, and release multiple versions of the app
- Educating people on its use either with one-on-ones, presentations, and/or documentation
- Receiving user feedback, fixing bugs, and adding new capabilities as necessary

2 | What we’re doing

OAuth Helper CLIs

We manage two CLI apps that help users get OAuth tokens from our OAuth provider and optionally exchange the tokens for other kinds of credentials (like AWS access keys.)

App Manager CLI

The people that use our apps have differing levels of technical expertise. We found that some users had trouble following install instructions for our apps that involved *curl* and *chmod*. We considered Homebrew but found the support for private tools to be less than stellar so we decided to create our own tool. It allows users to install and update apps and easily switch between production and early-release versions.

IAC Templating Tools

IAC - short for **Infrastructure-as-code** - is how we manage servers, databases, and deployments at Tempus. Our IAC is defined in HCL and applied by Terraform. We believe in self-service and want to enable our Engineering teams to provision their own infrastructure. However, we wanted to remove the need for them to learn HCL and how to adhere to the best practices defined by our DevOps Engineers. We've accomplished this by creating a wizard that walks people through a short form. The app then uses the form data to create multiple pull requests on behalf of the user and submits them to the appropriate IAC repositories.

Slack Bot

We run a Slack bot that monitors locks on our IAC state set by humans. If some state is locked, any attempt to change it will fail. People lock state in order to perform manual adjustments when things break or to perform maintenance that can't be completed through our IAC automation system. It's easy to forget that you're holding a lock and unintentionally block other users from making changes.

The Slack bot checks each morning for outstanding locks and notifies users by @-ing them

3 | What we’re looking forward to

More IAC Templating Tools

Our first IAM templater covers one common use case but won't support any more than that. Instead future templating apps will each be released as a separate tool. We're also going to improve the user experience by introducing Tauri, essentially a better Electron that allows us to write the backend in Rust. Tauri will allow us to rely on the same components that Tempus already provides to our front end web app engineers while allowing us to do things that a web app couldn't.

Multi-language SDKs

My team has been supporting another team at Tempus that is developing an SDK for their service with Rust. The SDK can then be compiled and packaged for use in Node or Python apps with the hope of greatly reducing the burden of having to write and maintain an SDK for each language they want to support.

Thanks to everyone else that enabled me to do this

- Katie Schilling
- Thomas Hatzopoulos
- Matt Kemp