GoFed: Road to v1

Rose-Colored Hindsight; Current Events; Sage Predictions

28 August 2018

Cory Slep

Preface

Preface

Goals are to answer the following questions:

- "What's the summary for version 0.x?"
- "What've been some highlights and lowlights?"
- "What's going on now?"
- "What *could* 1.x tackle?"
- "What *is* 1.x going to tackle?"
- "What are the risks?"

Preface

Non-goals of this presentation are:

- "What is ActivityPub and ActivityStreams? Can I have an introduction?"

Not addressed here, there's plenty of stuff out there. Deserving of a whole presentation of its own.

- "How can I use `go-fed/activity` to build a federated application?"

See https://go-fed.org (https://go-fed.org)!

- "Why did you do this in Go? Why a library?"

See my other presentation *ActivityPub in Go*.

0.x Summary

0.x Summary

From November/December 2017 to now...

Successfully implemented ActivityStreams and ActivityPub in Go, and it is listed at https://activitypub.rocks/implementation-report/ (https://activitypub.rocks/implementation-report/)!

Began before ActivityPub became a W3C Recommendation, successfully avoided the pitfall of implementing something that wouldn't stick around. Basically a bet on Mastodon at the time.

Also took the time to create presentations, a website (for docs and tutorials), tools for running the official implementation report, etc.

Highlights

Highlights

- I actually released a side project.
- go-fed client apps went from 0 to 3.
- Was seriously considered for prototyping MoodleNet, a vocabulary extension to ActivityStreams.
- Niche experience gain: The first and only statically-typed ActivityPub implementation.
- Has a "spiffy" website.
- >100 GitHub Stars

Lowlights

Lowlights

- Very few Go devs are interested in hacking on the ActivityPub federation.
- Lost out on the MoodleNet decision.
- go-fed is still (and probably always will be) an obscure part of the federation, since it is not user-facing.
- Haven't had time to use go-fed in my own personal blog, which is in a woeful state.
- Bus factor is precisely 1.

Concurrent Events

Concurrent Events

There are three big current events ongoing at the time of this writing that are influencing the direction of go-fed.

- New vocabulary extensions to ActivityStreams, which enables new kinds of federated applications to be written.
- Litepub discussions, which focuses on eliminating the JSON-LD requirement in ActivityPub.
- The Go blogging community is all about running static servers.

Concurrent Events: Vocabulary Extensions

There are three efforts of vocabulary extensions that I am tracking:

- **ForgeFed**: Describes activities and objects for different kinds of code forges/repositories (git, SVN, Mercurial, etc).
- **MoodleNet**: Social media vocabulary for educators, driven by Moodle.
- **ValueFlows**: Captures processes, inputs, and outputs for chains of economic activity.

I am not a direct contributor to any of these, but have greatly appreciated being kept aware of these efforts.

go-fed was fairly evaluated for prototyping **MoodleNet**, but lost out to Pleroma. I support the overall decision, and am disappointed in some of the reasons why, but I still want to support **MoodleNet** in the future. There's no reason why go-fed can't grow to support all of these as needed.

Concurrent Events: Litepub

The goal of Litepub is to strip the requirement of JSON-LD from ActivityStreams when implementing ActivityPub. There may be other secondary objectives (such as security), but I am not familiar enough to go into further detail.

This is a little unique for go-fed. Most other non-statically-typed languages may rely on some form of JSON-LD semantics in order to derive meaning from their data. So this effort is headed by people affected by this.

go-fed, on the other hand, does not do JSON-LD processing. This behavior is permitted by the ActivityPub spec. Instead, it leverages Go's static type system to provide semantic meaning.

Thus, go-fed may actually be a little ahead of the curve here.

Concurrent Events: Static Blogs

This is more of a widespread industry trend than an ongoing effort.

At the time of this presentation's creation, static websites for blogs written in Go, such as Hugo, are immensely popular.

Breaking into this area would definitely increases go-fed's popularity, but it would require some great ingenuity to support ActivityPub with a static server.

As go-fed is dynamic, this would be almost a completely new effort.

Possible 1.x Milestones

Possible 1.x Milestones

Keeping the concurrent events in mind, it is time to look at some concrete milestones. I have included a 4th one from various comments in GitHub issues:

- Make supporting new vocabulary extensions to ActivityStreams pleasant.
- Be an early (first?) adopter to the litepub effort, let go-fed support both litepub-based and ActivityStreams-based methods (if they differ).
- Figure out and create a static-website-friendly cousin to `go-fed/activity`.
- Evolve the way ActivityStreams types are mapped to golang static types.

Target 1.x Milestones

Target 1.x Milestones

Of the 4 possible milestones, the vocabulary extensions will be the primary focus for the leap to version 1.x. The reasons are:

- There is *immediate interest* in this, with *ongoing momentum*.
- It is a *concrete* goal with design & technical challenges.
- Makes `go-fed/activity` domain-agnostic.
- Continues to leverage go-fed's differentiating feature: the strong mapping of ActivityStreams types to golang structs.

An additional measurable goal is to *double* the number of applications using go-fed from three to six. This excludes anything I build, as this metric is a signal as to how valuable the library is.

Target 1.x Milestones: Why not the others?

Reasons for not selecting the others are:

- It's too early to begin to support Litepub, though this would be a great follow-on milestone.
- Figuring out how to appeal to static-website users is a little too abstract a problem to tackle right now.
- Alternative ways to statically map ActivityStreams and ActivityPub types to Go is a little too abstract a problem to tackle right now.
- Pursuing any of these alternatives means expanding go-fed beyond the Core and Extended ActivityStream types is not simple.

Target 1.x Milestones: What's the opportunity cost?

What I am missing out on for choosing this route:

- Possibility of becoming the first Litepub implementation somehow.
- Providing mass-appeal to static-website users and the internet glory along with it.
- Exploring more of type theory to come up with a more appealing mapping between ActivityStreams/ActivityPub and Go.

Risks

Risks

- Bus factor of 1.

If I don't feel like it, am sick, or am dead, not much gets done.

Risks

- Competing applications & libraries fully squeeze me out.

I think competition is healthy and that go-fed is a unique competitor in this space. It is a library, so instead of end-users determining its success, it is adoption by developers. Additionally, the statically typed nature of Go is a unique point.

Both of these differences are barriers to adoption, the latter much to my chagrin.

It is a risk that 1.x does not result in any new users of go-fed. In this case, it's probable that I'll begin considering whether to put it into maintenance mode.

Risks

- Continued compilation issues on linux_arm64.

The existing library balloons the compiler's memory usage on this platform, to at least 3.5 GB of RAM. I haven't yet been able to pinpoint the exact root cause. Cross-compiling is not sufficient to reproduce this.

However, the general root cause is known to be the `go-fed/activity/vocab` package. Since this generated code will be revisited, there's a possibility this risk will not manifest in version 1.x, but it still could.

Summary

- Super happy to have come this far!
- Very appreciative of those using this library already.
- Looking forward to making go-fed domain-agnostic.
- Trying to stay as realistic as possible when setting goals and evaluating risks.

Thank you

Cory Slep
https://github.com/go-fed/activity (https://github.com/go-fed/activity)
cjslep@gmail.com (mailto:cjslep@gmail.com)
@cj@mastodon.technology (http://twitter.com/cj@mastodon.technology)