Mike Perham, Creator of Sidekiq: From Employment to Independence

April 14, 2023

Read time: 9 minutes

I learned Ruby just in time to see `Sidekiq` take off. In the early 2010s, Ruby had seen a plethora of background job processors, from `BackgroundJob` to `DelayedJob`, `Resque`, and some others. Each improved and innovated in its own way on its predecessors. DelayedJob could run multiple jobs from a single Ruby instance (where BackgroundJob required loading the entire VM for each new job). Resque, in turn, improved upon this formula by storing jobs in Redis, which turned out to be much faster.

But until Sidekiq came around in 2012, you could still only run jobs in a single thread per Ruby process. Considering that you might need many threads per core to saturate CPU, or many threads in general to saturate IO, many applications were memory-bound long before they reached max CPU or IO. Then Sidekiq came along and ... "Sidekiq uses threads to handle many jobs at the same time in the same process." ... solved that problem.

Sidekiq was the first project that I can remember in the Ruby world that *required* your code to be threadsafe. The dominant web servers of the time, Unicorn and Passenger, were also not multi-threaded, relying on multiple processes for concurrency. And so, there were many libraries that were in fact not threadsafe, not to mention the application code that we had all been writing for years. The allure of multi-threading was too much to resist, and that all changed in a hurry. Within months every major library and most minor ones had already patched out their threadsafety issues. Here's a PR to Rails in mid-2012, enabling `threadsafe!` by default, a change that was spurred on by Sidekiq.

Since its release in 2011, Sidekiq has been the dominant library in Ruby for background job processing. With the release of version 0.5 of Sidekiq, Mike Perham also introduced Sidekiq Pro, which offered additional features, starting with batches, and notifications. Sidekiq Enterprise followed with version 0.7, with even more features and a higher price point. Sidekiq, therefore, follows the open core model, giving away Sidekiq, and charging for Pro and Enterprise features.

Recently, I asked Mike if he would do an interview for Code Code Ship, and he agreed. You can read the full interview below.

Mike, thanks for agreeing to answer some of my questions. It appears that you introduced Sidekiq Pro shortly after releasing Sidekiq (although in my searching I couldn't find the original announcement). How did you come to the idea to commercialize Sidekiq? Did you take any inspiration from other projects that came before?

If you go to my blog and scroll down to 2012, you can actually see Sidekiq's birth and my thoughts at the time. I had been working on background job systems for several years and knew that almost every Ruby company depended on one. I knew it was incredibly valuable because I was working with companies spending tens or hundreds of thousands of dollars on clusters for job processing. Improving efficiency by multi-threading would be a no-brainer to these companies.

I was also seeing open source developers burn out because of unpaid, ongoing labor. Building something valuable which you give away for free sounds like the perfect recipe for the Free Rider Problem. Since I wanted Sidekiq to be successful and I didn't want to burn out, I decided to run some "financial experiments".

> "I considered $350/qtr a failure."

Open Core as an idea was still relatively new in 2012, having apparently been coined by Adam Lampitt in 2008 (although I can not trace this back to the original quote). What were the factors that led you to choose the open core model? Did you also consider a dual-license construction?

My first financial experiment was selling a $50 commercial license variant so companies could avoid LGPL licensing. I think I sold 7 of them over 3 months. I considered $350/qtr a failure.

My next experiment was open core: sell a closed-source, add-on package with more enterprise-focused features and functionality that would layer on top of Sidekiq. Thus, Sidekiq Pro was born. My only question was how to distribute Sidekiq Pro

without giving access to the world. It just so happened that Rubygems had decent support for 3rd party gem servers. I set up a very simple gem server and gated access to the Sidekiq Pro .gem files with HTTP Basic Auth. I created a merchant account on a site called WePay which allowed me to sell digital goods. Upon purchase, WePay would send me "You got a sale!" email, I would log into the gem server and manually add access for the purchaser.

What was your overall situation in life when you started development of Sidekiq? Were you holding down a full-time job at the time?

I had been working with Ruby for 5 years at that point and was relatively well known for my previous Ruby OSS work and consistent blogging. This was invaluable when I started promoting Sidekiq and Sidekiq Pro: people already knew me by reputation. Marketing is easy if you already have an audience who knows and trusts you!

I was Director of Technical Operations at The Clymb, which was outdoor-focused e-commerce vendor (think REI without the stores). I had a team of four and we were responsible for site devops and infrastructure. This was perfect because they were my alpha customer and I was in charge of the tech stack. The Clymb switched to Sidekiq immediately, of course, and every Sidekiq Pro feature I wrote was running in production before it was released.

How long did it take to get to the point that you could support yourself with sales of Sidekiq Pro and Enterprise? When did you quit your job to go all-in on Sidekiq (if you were still working one when you started development)?

> “by March 2014 I was making more money from Sidekiq Pro sales than my Clymb salary”

I released Sidekiq Pro in Oct 2012. In 4Q2012 I sold $7500 worth of Sidekiq Pro at $500 each, so 15 licenses. More sales at a much higher price point. We might have something here!

I worked for The Clymb until June 2014 when I quit to focus on Sidekiq full-time. I remember that by March 2014 I was making more money from Sidekiq Pro sales than my Clymb salary (which was approx $10,000/mo).

I incorporated with the help of a local lawyer and thus Contributed Systems was born.

After a tumultuous history, Sidekiq has turned out to be the final evolution of background job processing for Ruby. Why do you think that is?

I'm not sure I would go that far. There have been several job systems for Ruby which came out since but none have been as feature-rich and well-supported as Sidekiq. There will always be innovation and new directions but for the most part Ruby applications have settled on Postgres and Redis as the two persistent stores which are "acceptable" to the vast majority of the community.

> “in the end, OSS burnout will kill any free project which gets traction”

And I believe that "feature-rich and well-supported" is 100% due to my commercial side. I do occasionally see competitors start up but I know that in the end, OSS burnout will kill any free project which gets traction. While I love the tech, I'm still here because I get paid to support my users.

How was the initial reception to Sidekiq Pro? Were you worried that there would be backlash to charging money for something that people are accustomed to getting for free?

Absolutely, but it was far warmer than I expected. I blogged about the problem of burnout and people 100% supported me. In fact it gave them yet another reason to purchase: to support someone they knew and trusted.

What is the scope of your operation these days? Have you expanded to hire other developers to work on Sidekiq?

I still have 0 employees and don't plan to hire. I tune my business processes to run as lean as possible: Most of my customers are on credit card so their payments are automatic. The gem servers take about one day of maintenance per year. I can't really outsource much of my support work because it is so technical and specialized.

> “something like CCS would have been very helpful”

How do you distribute Sidekiq Pro / Enterprise? Would you have used Code Code Ship if it was available at the time?

It took me a while to find a means of distribution and something like CCS would have been very helpful! It turns out that Rubygems is very well designed; providing your own gems really isn't hard or resource intensive. NPM, by way of comparison, is massively over-engineered and much more difficult to federate. My gem server is a $6 droplet on DigitalOcean. Because gems are just static files, that little droplet can handle millions of requests per day with just Apache. Oh, and I run three servers in parallel for failover purposes for a grand total of $18/mo.

Finally, what does the future hold for Sidekiq and more generally for yourself? I also see that you have another project, called Faktory. How has the reception to that been?

In 2017 I decided I wanted to explore bringing Sidekiq to other ecosystems and languages. I wrote Sidekiq.cr which was Sidekiq re-implemented in Crystal. I toyed with Sidekiq.js but decided to kill that idea real quick because, as we all know, JavaScript is terrible. I realized that I couldn't re-implement Sidekiq in N different language so I changed the architecture and started Faktory. Faktory is written in Go and acts as a replacement for Redis. Faktory centralizes all of the job and queue management and provides a simple protocol for background jobs, language-specific worker processes implement that protocol and actually execute the job.

And like Sidekiq, Faktory has a commercial extension in Faktory Enterprise which provides more features and support.

I released Sidekiq 7.0 last fall which added some really beautiful and useful performance metrics in the Web UI along with the ability to embed Sidekiq within another Ruby process. The last few months have been focused on working out any lingering bugs and stabilizing 7.0.

> “If you build something valuable, charge money for it”

Mike, thanks for taking the time to answer my questions. I am sure that developers who are considering commercial software library development will be very interested in your experiences and perspective.

Like me, I hope other open source developers will learn that money is not inherently evil: it's just an exchange medium for value. If you build something valuable, charge money for it so you can continue to build and support your community!

By Derek Kraan

Related articles

2023-05-15T12:00:00Z

Joris Schellekens, Creator of Borb

Joris tells us what it's like bootstrapping a new commercial library

Read more >

2023-04-19T07:00:00Z

Parker Selbert, Creator of Oban

Parker pulls back the cover on what it's like to commercialize your package

Read more >

Get in touch

You can get in touch with us via email or on our Discord server. Click the button for an invite.

Discord

Subscribe

Always stay up to date by subscribing to our newsletter.

Enter your email

Subscribe