

RubyにおけるQUICの現在地

unasuke

Fastly Tech Meetup vol.1

2026-09-10

自己紹介

- Name: うなすけ
- Work: フリーランス
- Kaigi on Rails 2026 organizer (10/16-17)
- GitHub <https://github.com/unasuke>
- Fediverse <https://mstdn.unasuke.com/@unasuke>
- X https://x.com/yu_suke1994
- Website <https://unasuke.com>



【宣伝】Kaigi on Rails 2026



The poster features a dark background with a stylized, isometric illustration of a city street with buildings and a train. The title 'Kaigi on Rails' is prominently displayed in the center, with 'Kaigi' in white and 'on Rails' in a large, bold, white font. The year '2026' is written below the title in a pink, pixelated font. Two speakers are featured: David Heinemeier Hansson on the left and Yukihiro 'Matz' Matsumoto on the right. Both are shown in a circular frame with a yellow glow. The word 'keynote' is written above each speaker's name. In the bottom left corner, there is a small illustration of a white dog. The bottom of the poster contains three black boxes with yellow text: the first box contains the event name in Japanese and English, the second box contains the dates and times, and the third box contains the location.

keynote
David Heinemeier Hansson

keynote
Yukihiro "Matz" Matsumoto

**Kaigi
on
Rails**
2026

カイギ - オン - レイルズ
Since 2020 | 7th edition

2026 October **16**^{Fri.} → **17**^{Sat.}

Shibuya, Tokyo Japan
Bellesalle Shibuya Garden + Online

このトークの想定聴衆

- QUIC : おおまかに理解している
 - QUICの実装の名前をいくつか知っている
- Ruby : 書いたことはある
 - 型のないスクリプト言語で、Webアプリの実装で使われがちなのは知ってる
- Ruby on Rails : 聞いたことはある
 - RubyのWebアプリケーションフレームワークなんですよ

そもそも、プログラミング言語のHTTP実装ってどんなもんよ

- Ruby: `net/http` (HTTP/1.1まで)
- Python: `http.client` (HTTP/1.1まで)
- Perl: `HTTP::Tiny` (HTTP/1.1まで)
- Go: `net/http` (HTTP/2まで)
 - QUICについては golang.org/x/net/quip があるにはある
- C#/.NET: .Net 7以降でHTTP/3をサポート
 - `HttpVersion.Version30`
- Rust: 標準ライブラリとしてHTTPを提供していない

※あくまでも言語標準としてどうなのか、の話

Rubyの net/http の現状

- サポートされているのは HTTP/1.1 まで
- 初コミットは1999年12月17日

```
require 'net/http'
require 'uri'

url = URI.parse('http://example.com/index.html')
res = Net::HTTP.start(url.host, url.port) { |http|
  http.get('/index.html')
}
puts res.body
```

Rubyの net/http の現状

- サポートされているのは HTTP/1.1 まで
- 初コミットは1999年12月17日
- HTTPSを使いたかったらこう

```
require 'net/http'
require 'uri'

url = URI.parse('https://example.com/index.html')
res = Net::HTTP.start(url.host, url.port, use_ssl: true) { |http|
  http.get('/index.html')
}
puts res.body
```

そんなRubyにおけるQUICの現在地について

- OpenSSL
- protocol-quick
 - protocol-http3
- quick-ruby
- quicksilver
- raiha

OpenSSL

<https://github.com/ruby/openssl/>

Rubyの標準添付ライブラリのひとつ。OpenSSLのAPIをRubyから使えるようにしている。

OpenSSL側に存在するQUIC APIのRuby側への露出については
<https://github.com/ruby/openssl/pull/1008> にて作業が進められている。

protocol-quic, protocol-http3

これらの説明をする前に、Rubyの並行並列処理について軽く説明する必要がある……

Rubyの並行並列処理

- Threadを使う
 - いわゆるスレッド、native thread
- Fiberを使う
 - 軽量スレッド、coroutineと呼ばれたりする
 - Fiber schedulerを使う
 - IO待ちにhookしてFiberを切り替えるschedulerを導入する
 - scheduler自体の実装はRuby本体にはない
- Ractorを使う
 - M:N thread、制約は多め

Rubyの並行並列処理とRails

Railsを運用するときに使うサーバーとして挙げられるのは……

- unicorn

- マルチプロセス(Prefork)型。pumaの前のデファクトスタンダード
- 今だとforkされたpitchforkを使うという選択もあるかも
- GitHubが落ちると出るrage unicornの由来

- puma

- マルチスレッド、マルチプロセス型。現在のRailsのデフォルト

- falcon

- Fiber schedulerを使う
- Shopifyの本番で動いている(はず)

Rubyの並行並列処理、Fiber scheduler

Fiber schedulerのスケジューラー実装のひとつに `async gem` がある。

<https://github.com/socketry/async>

この`async gem`から使いやすいうように`socketry org`以下で様々なprotocolの実装が管理されている。

- <https://github.com/socketry/async-http>
 - <https://github.com/socketry/protocol-http>
 - <https://github.com/socketry/protocol-http2>
- <https://github.com/socketry/async-websocket>
 - <https://github.com/socketry/protocol-websocket>

protocol-quic, protocol-http3に戻ってきたぞ

- <https://github.com/socketry/protocol-quic>
- <https://github.com/socketry/protocol-http3>

ngtcp2、nghttp3とschedulerのためのC++によるコードから成る
gem

たぶんまだあんまり広く使われていない(production環境での運用についての話を聞いたことがない)

quic-ruby

- <https://github.com/unasuke/quic-ruby>

拙作。ngtcp2とLibreSSLをbindingしたRuby gem。まだ開発途上なので.....

quicsilver

- <https://github.com/hahmed/quicsilver>

MsQuicのbinding gem。先ほどのfalconと組み合わせて使うことも想定されているような……

raiha

- <https://github.com/unasuke/raiha>

拙作。これは既存のライブラリのbinding gemではなくPure RubyによるQUICプロトコルの実装を目指して開発中のものです。