Automating your developer workflow for faster feedback loops

Sabrina Gannon

June 28 2022 Wnbrb meetup

Who am I?

- I'm a Senior Ruby on Rails developer who just wrapped a round of job seeking.

Who am I?

- I'm a Senior Ruby on Rails developer who just wrapped a round of job seeking.
- I really enjoy having lots of projects on the go, both technical and non technical.

Who am I?

- I'm a Senior Ruby on Rails developer who just wrapped a round of job seeking.
- I really enjoy having lots of projects on the go, both technical and non technical.
- I struggle with both getting started on new tasks as well as staying focused.

Why this topic?

- Automation is extensible and customizable.

Why this topic?

- Automation is extensible and customizable.
- Writing small utilities lets you learn and become more efficient.

Why this topic?

- Automation is extensible and customizable.
- Writing small utilities lets you learn and become more efficient.
- Faster feedback loops mean less mental overhead needing to run a command yourself.

Why this topic?

- Automation is extensible and customizable.
- Writing small utilities lets you learn and become more efficient.
- Faster feedback loops mean less mental overhead needing to run a command yourself.
- Creating a template or automation can reduce context switching.

Let's look at some examples!

Using the Guard gem to run a command on a file change

Let's say I want to write my slides in a markdown file, and generate a PDF output of those slides to present with.

I can do that using the following command using the Pandoc document converter:

```
pandoc -t beamer slides.md -o slides.pdf
```

Using the Guard gem to run a command on a file change

- Rerunning the command starts to feel tiresome; I'm still learning Pandoc and want to see my changes quickly, but I have to rerun the command in my terminal every time.

Using the Guard gem to run a command on a file change

- Rerunning the command starts to feel tiresome; I'm still learning Pandoc and want to see my changes quickly, but I have to rerun the command in my terminal every time.
- The feedback loop I'm working with is: Make a change –> regenerate PDF –> See the effect of my changes.

Using the Guard gem to run a command on a file change

- Rerunning the command starts to feel tiresome; I'm still learning Pandoc and want to see my changes quickly, but I have to rerun the command in my terminal every time.
- The feedback loop I'm working with is: Make a change –> regenerate PDF –> See the effect of my changes.
- If I can just always have the PDF version available when I make a change to my Markdown file, that would remove a manual step that slows down the feedback loop.

Using the Guard gem to run a command on a file change

- Rerunning the command starts to feel tiresome; I'm still learning Pandoc and want to see my changes quickly, but I have to rerun the command in my terminal every time.
- The feedback loop I'm working with is: Make a change –> regenerate PDF –> See the effect of my changes.
- If I can just always have the PDF version available when I make a change to my Markdown file, that would remove a manual step that slows down the feedback loop.
- The feedback loop I want is: Make a change –> See the effect of my changes.

Installing the Guard gem

Let's install Guard by adding a Gemfile to the project directory with the following:

```
# Gemfile

source "https://rubygems.org"

gem "guard"
gem "guard-shell"
```

We can then install our gems with `bundle exec`

Initializing the Guard gem

Guard is configured using a Gemfile, which is generated by running the command *guard init*.

However, because I'm using the shell plugin to run my shell command, what I actually type in to initialize my file is:

```
guard shell init
```

Configuring the Guard gem

Here's what the Guardfile looks like to run our command:

```
# Guardfile

# https://github.com/guard/guard-shell#check-syntax-of-a-r

# m[0] is the entire filename string "slides.md"
# uses Ruby's File utilities to remove the
# '.md' file extension when naming the pdf

guard :shell do
  watch(/.*\.md$/) { |m| `pandoc -t beamer #{m[0]} -o #{Fi
end
```

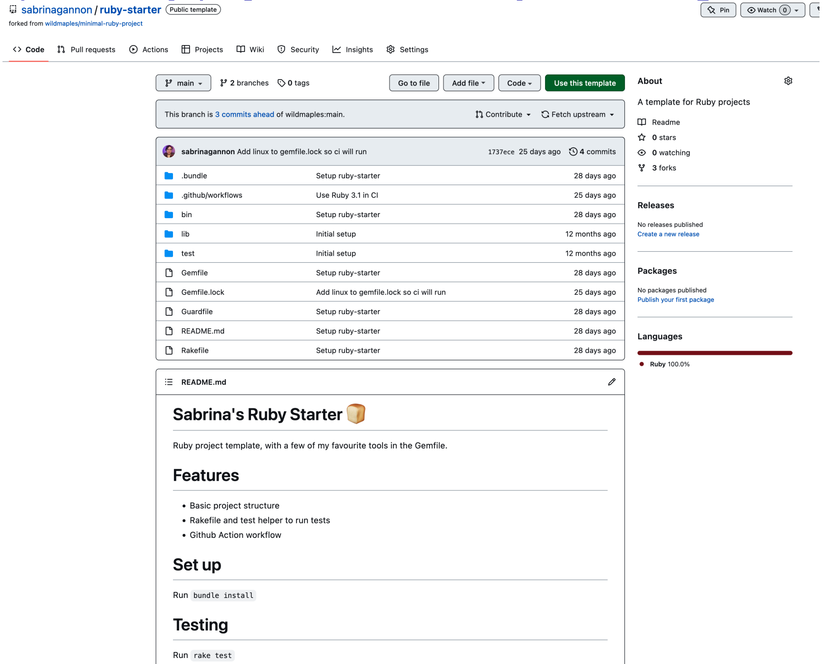
Running the Guard gem

To start watching all markdown files and generating PDFs, I run `bundle exec guard` in the project directory.

When a markdown file is changed, the following output runs in my terminal:

```
21:31:53 - INFO - Guard is now watching at '/Users/sabrina/
markdown updated, pdf generated
[1] guard(main)>
```

Template Ruby project with all of my favourite gems



sabrinagannon / ruby-starter Public template

forked from wildmaples/minimal-ruby-project

This branch is 3 commits ahead of wildmaples:main.

sabrinagannon Add linux to gemfile.lock so ci will run — 1737ece 25 days ago — 4 commits

.bundle	Setup ruby-starter	28 days ago
.github/workflows	Use Ruby 3.1 in CI	25 days ago
bin	Setup ruby-starter	28 days ago
lib	Initial setup	12 months ago
test	Initial setup	12 months ago
Gemfile	Setup ruby-starter	28 days ago
Gemfile.lock	Add linux to gemfile.lock so ci will run	25 days ago
Guardfile	Setup ruby-starter	28 days ago
README.md	Setup ruby-starter	28 days ago
Rakefile	Setup ruby-starter	28 days ago

About

A template for Ruby projects

Readme
0 stars
0 watching
3 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

● Ruby 100.0%

README.md

Sabrina's Ruby Starter 🍞

Ruby project template, with a few of my favourite tools in the Gemfile.

Features

- Basic project structure
- Rakefile and test helper to run tests
- Github Action workflow

Set up

Run `bundle install`

Testing

Run `rake test`

Test Driven Development using Guard and Rake

```
# Guardfile
guard 'rake', :task => 'test' do
watch(%r{^lib/(.+)\.rb})
watch(%r{^test/(.+)\.rb})
end
```

```
# Rakefile
require "rake/testtask"

Rake::TestTask.new(:test) do |t|
t.libs << "test"
t.test_files = FileList["test/**/*_test.rb"]
end

task :default => :test
```

Snippet to load gems inline in a single file Ruby script

Credit to Matheus Richard @matheusrich's tweet about this being an option!

```
#!/usr/bin/env ruby

require "bundler/inline"

gemfile do
  source "https://rubygems.org"

  gem "httparty"
end

response = HTTParty.get("http://api.stackexchange.com/2.2/q

puts response.body
```

How developer automations can be useful

- Prepping for technical interviews.

How developer automations can be useful

- Prepping for technical interviews.
- “Test Driven Development” workflows.

How developer automations can be useful

- Prepping for technical interviews.
- “Test Driven Development” workflows.
- Automating tedious work to free up time in your daily life and aid with other simple tasks.

Start small and where you are

Where you can go from here

- Editor integrations

Where you can go from here

- Editor integrations
- Github Actions and similar platforms can run automated commands

Thank you!

Places you can contact me:

References
https://github.com/sabrinagannon/automating-your-dev-workflow
https://graceful.dev/courses/acapa/
https://pandoc.org/MANUAL.html
https://github.com/guard/guard &&
https://github.com/guard/guard-shell
https://github.com/sabrinagannon/ruby-starter (fork of Maple Ong's simple ruby template)
https://twitter.com/matheusrich/status/1536429683591585792