Project Kickoff & Project Infrastructure

Software Engineering II
WS 2020/21

Enterprise Platform and Integration Concepts

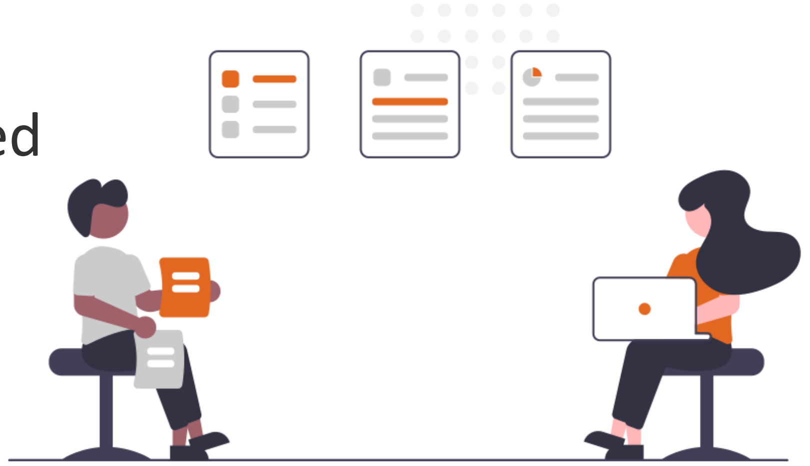
Let’s Get Started

Project

- **Sprint 1** (2 weeks, 27.11.20—11.12.20)
  - Start with **Sprint Planning** meeting
  - Sprint 1 **Review & Retrospective Meetings** at the end
- **Weekly Stand-ups** in weeks with no other Scrum meeting
  - Short, timeboxed. 2 minutes per person.

Meetings

- Exact meeting dates **negotiated** with tutors
- Sprint Review & Planning for next sprint can be colocated
- **On demand:**
  - User research with customer. Clarify questions!
  - Coordination with other SMs, POs, devs, teams



Let’s Get Started

POs (reminders)

- Extract requirements + create user stories (GitHub tickets)
- Get an idea of the interaction workflows in the system (**mockups**?)
- Prepare Sprint Planning, inform yourselves on what the team is doing
- PO should roughly know what the team has done **before the review**

Developers

- Clone repository, get application working locally, read & run tests
- **Understand architecture and dependencies**
  - If you were lead architect, **how would you construct this?**
  - *Greatest challenges in the problem domain? What libraries could help?*
- **Play around** and try things out.
  - *Where does the system have problems? What makes no sense to you?*

Let’s Get Started

SMs

- Part of your job is research and retrospection
  - Observe the **meta-level** of a meeting
    - Equal participation? How is the team communicating?
    - What is working well in the team? What isn't?
  - How can team meetings be structured? Prepare agenda
  - This is a hard job, **focus on it**
- Every team is different. **Experiment!**

Meeting spaces

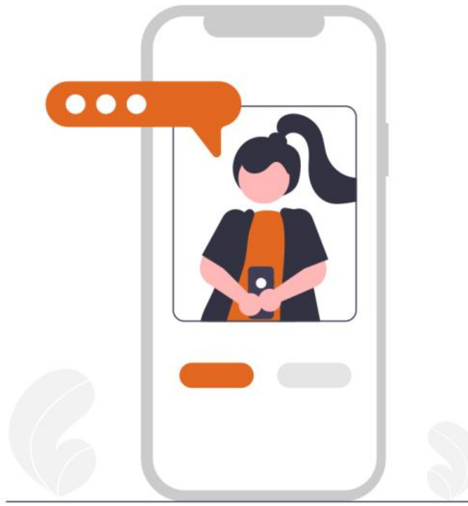
- Regular meeting + work timeslot
- We can reserve spaces in the Villa at the EPIC chair, if this is needed

Communication

Communication Infrastructure

- Large choice of options
  - Probably best to focus on a few key ones
  - **Keep information together**, build a knowledge base
- We've setup some course communication infrastructure
  - See the website for the links
  - Telephone and personal contact for direct communication?
- … be **creative**!

  (but let us know, we're interested in learning what might be useful in the future)



Project Management Tools

The Swiss army knife of software development

- Integrating tools for most common activities in **one place**
- Wiki, bug tracking, time management, project analytics, discussions, …

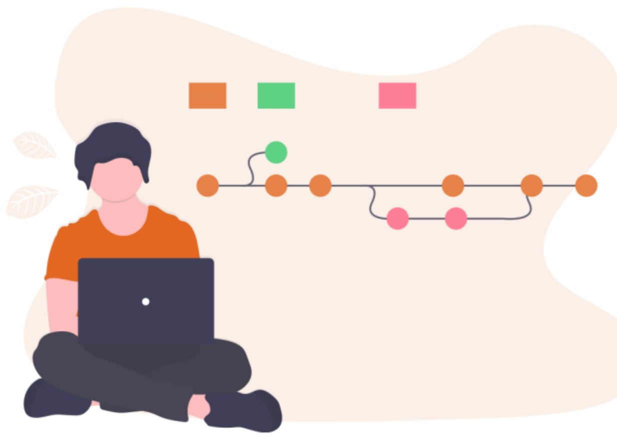
- Examples:
  - Microsoft Team Foundation Server
  - Jira
  - Redmine
  - Gitlab
  - **GitHub**

Version Control System

Repository to store and organize development artifacts

Features

- Versioning
- Dealing with variants: **branches**
  - **main & dev**
  - **Pull Requests**
- Access control
  - Authentication, authorization
  - **Concurrent** development
- Reporting and communication
  - How many versions, variants, changes, persons
  - History of changes



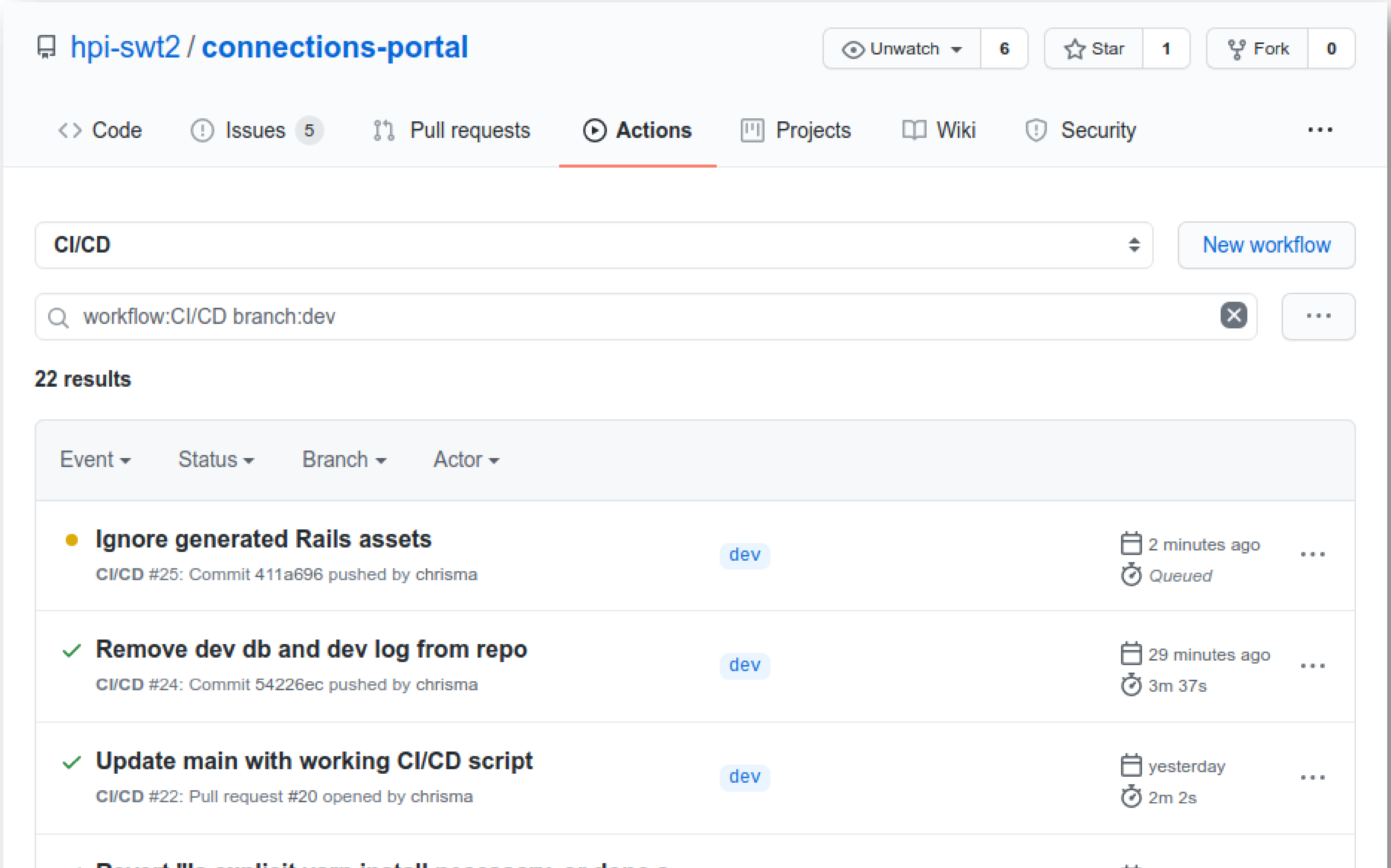
Continuous Integration

How do you make sure your software always works?

Continuous Integration

- **Connected** to version control
  - Run tests when code changes
  - Ideally covering all development branches
- **Display errors & notify**
- Customizable run scripts
- Examples:
  - GitHub Actions
  - Jenkins/Hudson
  - Travis CI

Example: GitHub Actions



workflow:CI/CD branch:dev

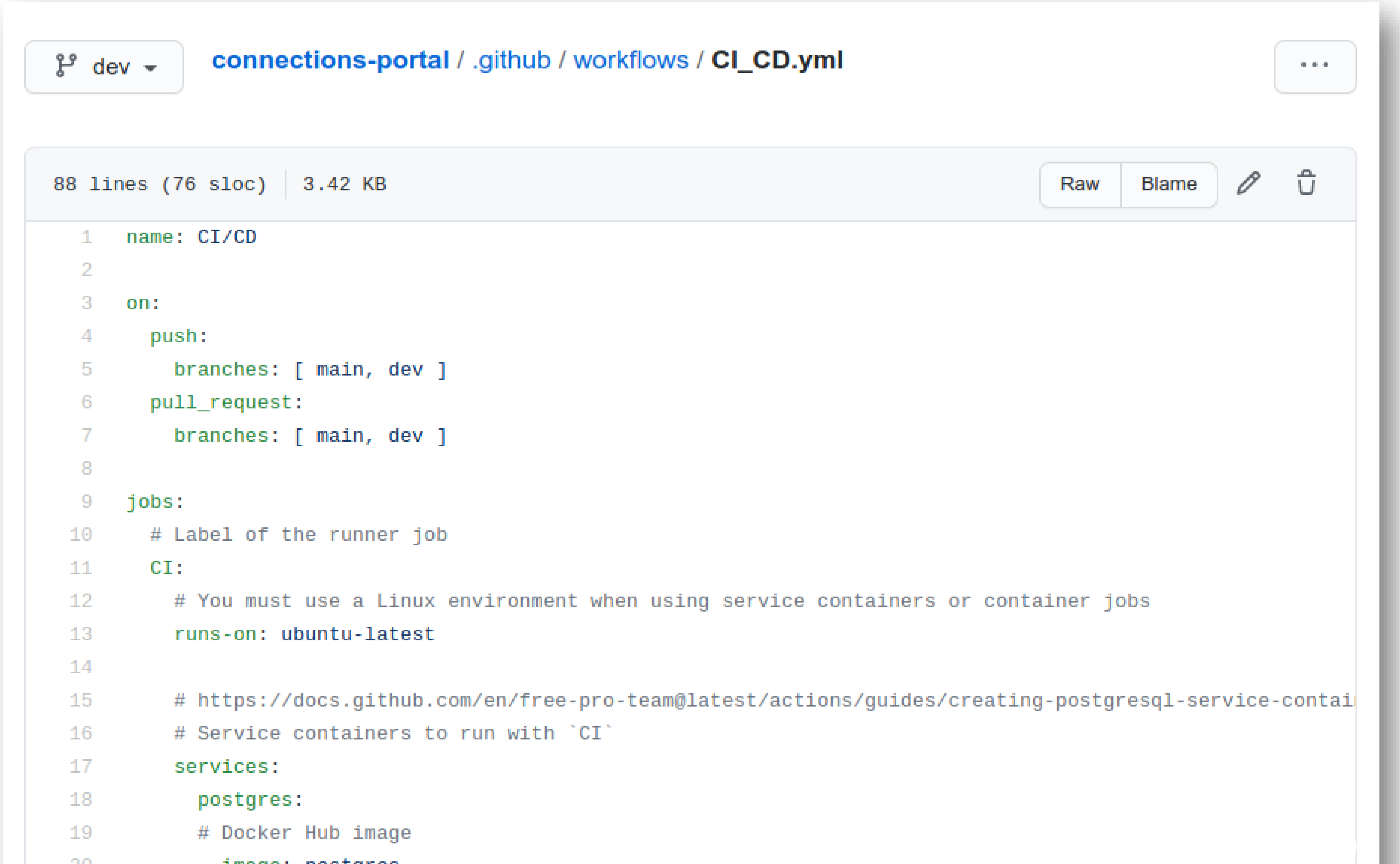
22 results

● **Ignore generated Rails assets** — dev — 2 minutes ago — *Queued*
CI/CD #25: Commit 411a696 pushed by chrisma

✓ **Remove dev db and dev log from repo** — dev — 29 minutes ago — 3m 37s
CI/CD #24: Commit 54226ec pushed by chrisma

✓ **Update main with working CI/CD script** — dev — yesterday — 2m 2s
CI/CD #22: Pull request #20 opened by chrisma

Example: GitHub Actions


dev
connections-portal / .github / workflows / CI_CD.yml
88 lines (76 sloc) | 3.42 KB
Raw
Blame

```yaml
name: CI/CD

on:
  push:
    branches: [ main, dev ]
  pull_request:
    branches: [ main, dev ]

jobs:
  # Label of the runner job
  CI:
    # You must use a Linux environment when using service containers or container jobs
    runs-on: ubuntu-latest

    # https://docs.github.com/en/free-pro-team@latest/actions/guides/creating-postgresql-service-contai
    # Service containers to run with `CI`
    services:
      postgres:
      # Docker Hub image
```

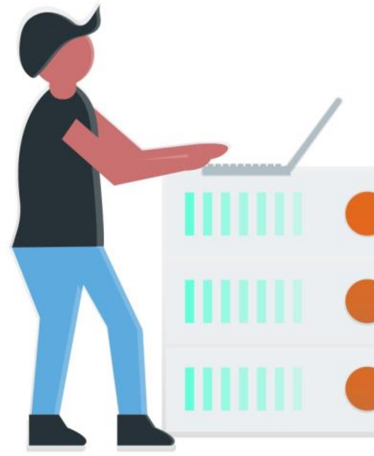
Application Deployment

How can you always have a running version of the application available?
(why would you want to?)

Deploy the application

- Simple: test deployment on local machine
- Deployment on separate machine:
  - Dedicated Servers
  - Hosted and managed by a (paid) third party
- **Continuous Deployment:**
  Deployment automatically triggered by successful CI build
  - Deployment config is part of the project
  - No extra effort

You can run your app in deployment mode locally



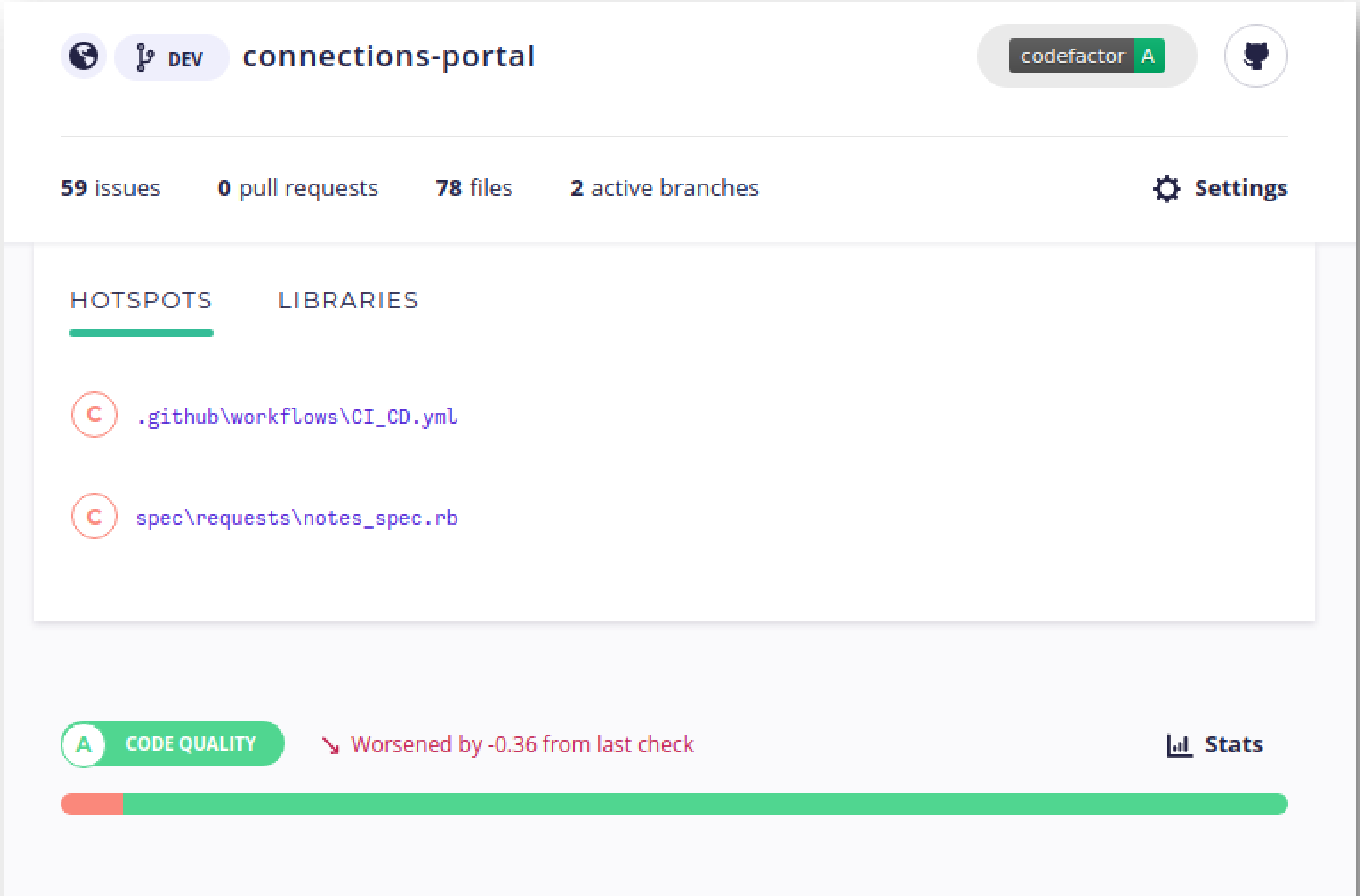
Code Quality

How can you ensure that the software adheres to certain quality standards (complexity, test coverage, etc.)

Check for compliance

- Review your own code (diff), code reviews by others
- **Automatic** checks
  - Hosted tools: e.g. CodeClimate, Codefactor, Codebeat
  - Local code coverage: SimpleCov (http://www.simplecov.org/)
    - Can run automatically during each test run
    - coverage/index.html in your application folder
  - Local code smells: RuboCop (https://www.rubocop.org)

Example: CodeFactor



DEV connections-portal

codefactor A

59 issues 0 pull requests 78 files 2 active branches Settings

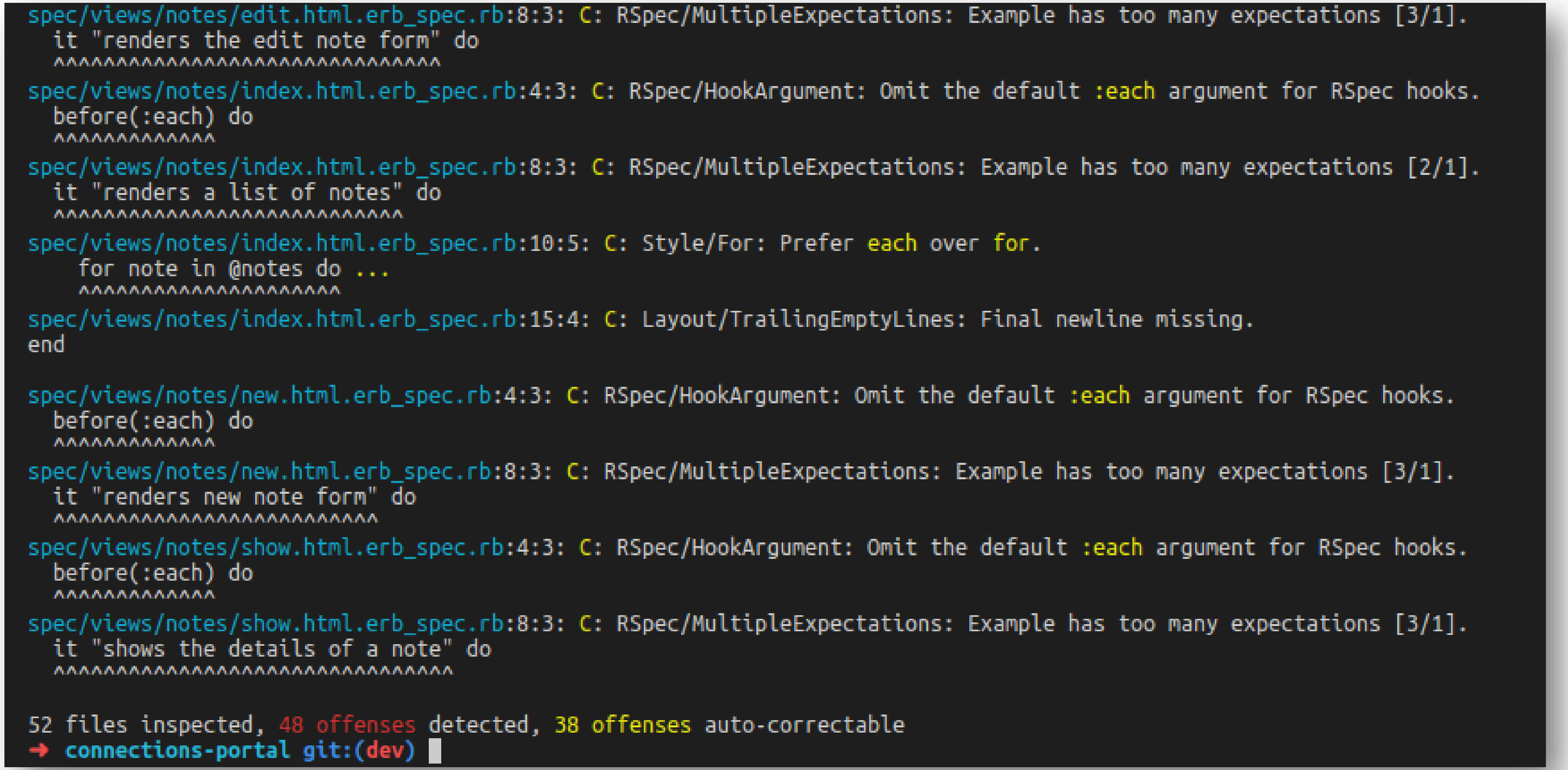
HOTSPOTS LIBRARIES

C .github\workflows\CI_CD.yml

C spec\requests\notes_spec.rb

A CODE QUALITY ↘ Worsened by -0.36 from last check Stats

Example: RuboCop

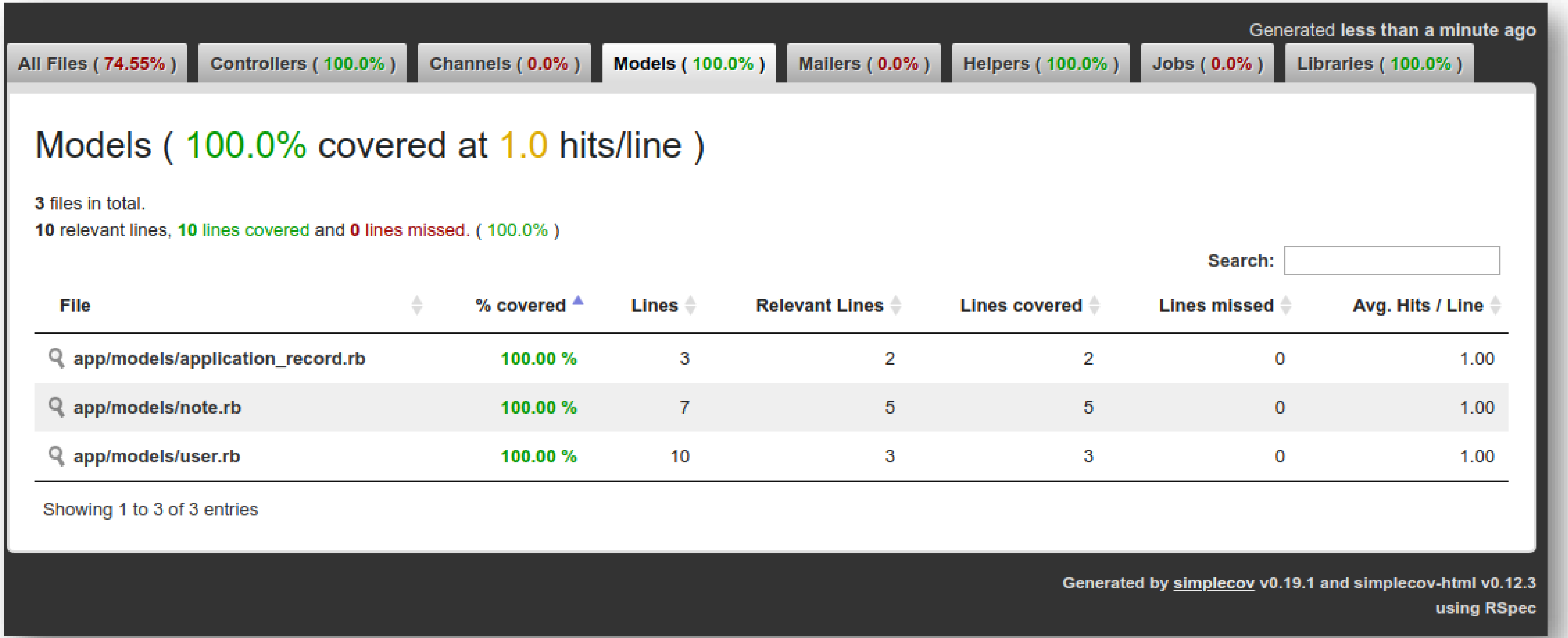


```
spec/views/notes/edit.html.erb_spec.rb:8:3: C: RSpec/MultipleExpectations: Example has too many expectations [3/1].
  it "renders the edit note form" do
  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
spec/views/notes/index.html.erb_spec.rb:4:3: C: RSpec/HookArgument: Omit the default :each argument for RSpec hooks.
  before(:each) do
  ^^^^^^^^^^^^^
spec/views/notes/index.html.erb_spec.rb:8:3: C: RSpec/MultipleExpectations: Example has too many expectations [2/1].
  it "renders a list of notes" do
  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
spec/views/notes/index.html.erb_spec.rb:10:5: C: Style/For: Prefer each over for.
    for note in @notes do ...
    ^^^^^^^^^^^^^^^^^^^^^
spec/views/notes/index.html.erb_spec.rb:15:4: C: Layout/TrailingEmptyLines: Final newline missing.
end

spec/views/notes/new.html.erb_spec.rb:4:3: C: RSpec/HookArgument: Omit the default :each argument for RSpec hooks.
  before(:each) do
  ^^^^^^^^^^^^^
spec/views/notes/new.html.erb_spec.rb:8:3: C: RSpec/MultipleExpectations: Example has too many expectations [3/1].
  it "renders new note form" do
  ^^^^^^^^^^^^^^^^^^^^^^^^^^
spec/views/notes/show.html.erb_spec.rb:4:3: C: RSpec/HookArgument: Omit the default :each argument for RSpec hooks.
  before(:each) do
  ^^^^^^^^^^^^^
spec/views/notes/show.html.erb_spec.rb:8:3: C: RSpec/MultipleExpectations: Example has too many expectations [3/1].
  it "shows the details of a note" do
  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

52 files inspected, 48 offenses detected, 38 offenses auto-correctable
➜ connections-portal git:(dev)
```

Example: SimpleCov



Generated less than a minute ago

All Files (74.55%) | Controllers (100.0%) | Channels (0.0%) | Models (100.0%) | Mailers (0.0%) | Helpers (100.0%) | Jobs (0.0%) | Libraries (100.0%)

Models (100.0% covered at 1.0 hits/line)

3 files in total.
10 relevant lines, **10** lines covered and **0** lines missed. (100.0%)

Search:

File	% covered	Lines	Relevant Lines	Lines covered	Lines missed	Avg. Hits / Line
app/models/application_record.rb	100.00 %	3	2	2	0	1.00
app/models/note.rb	100.00 %	7	5	5	0	1.00
app/models/user.rb	100.00 %	10	3	3	0	1.00

Showing 1 to 3 of 3 entries

Generated by simplecov v0.19.1 and simplecov-html v0.12.3
using RSpec

Dependencies

More than likely someone else has already solved a specific (web-dev) problem.

Libraries (Ruby gems) & external dependencies

- Most likely more mature and bug-free than your custom solution
- Someone else has checked the code in your application
- You are maximizing development time
- **But:** dependencies introduce complexity
  - Mostly very powerful, generic solutions
  - Require extensive configuration
  - Need to be learned by every developer (effort multiplied by X)
  - Consensus among developers on usage?

Your Project

Communication infrastructure
Continuous Integration
Continuous Deployment
Code Quality
Dependencies

Any other tools you might want to use!

Something you have had good experiences with in the past?
Your favorite development tool?
But, your team is not the only one using it, communicate.