

RFC 142 APPROVED: Work plan to address backend e2e tests

Editor: joe@sourcegraph.com

Status: APPROVED

Required reviewers (please review by EOD 2020-04-17): @keegan, @rijnard, @ryan, @thorsten, @tomas, @nick

Approvals: Tomás, nick@

Background

TL;DR We have both flaky browser-based e2e tests and hard-to-reason CI infrastructure for e2e tests, and we want to lift as many tests as possible to be unit and integration tests.

[RFC 137 APPROVED: Better automated testing](#)

Problem

Our goal for the core services team is to convert as many backend-related e2e tests as possible to unit and integration tests, and have the same or more confidence as we originally intended for those e2e tests.

The migration requires non-trivial effort and we need to prioritize tasks as iteration items, therefore it is important to have a work plan for how we would like to actually tackle it.

Proposal

Audit and convert backend-related e2e tests into unit and integration tests written in Go.

Steps:

- ~~1. Audit all existing e2e tests, filter down the ones that're backend related.~~
- ~~2. Categorize them into two groups by how we can achieve the same or more confidence: unit tests and integration tests.~~ [Backend e2e tests audit](#)
3. For unit tests, it's pretty straightforward and we can just do what we do today.
4. For integration tests, we use GraphQL API to interact with the Sourcegraph server instance. This aligns with how our web app talks to the backend (except few HTTP routes that serve HTML pages directly, which could be tested via regular HTTP requests), and therefore sufficient to have the same coverage and confidence as we do in respective e2e tests.

Integration tests

We need to run tests end-to-end (i.e. no mocks), we would have to have a running Sourcegraph instance for our test client to hit. The sourcegraph/server:insiders version seems the easiest option to set up an up-to-date instance and run integration tests, but the problem is we currently build new Docker images after all tests passed, so we would do the same to how we set up the environment today for e2e tests (docker build server image and run tests).

Prototype: <https://github.com/sourcegraph/sourcegraph/pull/10470>

Rough time estimate

- 1 week for unit tests
- 2 weeks for integration tests and ramping up

Definition of success

- We can remove all existing backend-related e2e tests (written in JS) and reliably run corresponding unit and/or integration tests (written in Go) as part of CI on all branches.
- Adding and reasoning of failed integration tests is as straightforward as adding and reasoning of failed unit tests.