Running Parallel Cypress Tests

As part of UFC-389, I finally decided that waiting an hour for the Cypress tests is just too much.

tl;dr we can now run multiple copies of the docker-compose services, with matching copies of Cypress itself to run the tests in parallel. Due to the hardware limitations of my desktop, we can only do 4 copies currently, but this can be (mostly) easily scaled up in the future.

How to do it

This setup requires tmux for the terminal multiplexing. This is primarily for easier reading of output logs.

Here's how to run the Cypress tests in parallel:

i. In one tmux window, start the docker services:

   - `make start-cypress-parallel`

ii. In a separate tmux window, start the Cypress instances:

   - `make run-frontend-cypress-parallel`

Enjoy a roughly 3x speedup (from 1 hour → 20 minutes with 4 instances).

How it works

Docker

Each docker-compose instance uses an index to provide different URLs for the frontend/backend services. Using 4 copies, that means we end up with services running on ports 3001, 3002, 3003, etc. This is accomplished by using a separate `docker-compose` config that is templated with environment variables for the ports, along with separate 'project names' (aka network names) to separate the containers into separate networks.

Additionally, as mentioned above, each copy is run in its own separate tmux pane, for better readability of the logs.

Note that, sometimes, when starting the containers, the Backend service might compile incorrectly. I can only assume this is because things start to fall apart with the high CPU loads of starting 4 copies all at once.

- If this is the case, just kill that copy and restart it with the `scripts/cypress_start_parallel_service.sh $INDEX` script.

Cypress

Normally, a Cypress instance just uses all of the specs files in the integration folder. However, they can be manually provided using the `--spec` option. We take advantage of this for parallelization.

First, the list of spec files is stored in a single shared state file. Then, each copy of Cypress (aka a worker) can read from the file to pull one of the specs to run. The workers keep pulling from the state file until all the specs have been run.

Simple, no retry mechanism, but it works.

The Cypress workers similarly use tmux panes for log readability.

Note that, while you can increase concurrency from e.g. 4 to 6, the increased CPU load causes individual tests to take longer, resulting in minimal to non-existent performance gains. At least, with my current CPU (a 4790k).

GCP Testing

i. Increase inotify watcher count:

   - `echo fs.inotify.max_user_watches=10485760 | sudo tee -a /etc/sysctl.conf && sudo sysctl -p`

ii. Install docker

iii. Install docker-compose

iv. Install nvm + node v14.15.1

v. Install cypress packages

vi. Add GitHub key (just copy them over using the GCP ssh interface)

vii. Clone uFincs repo

viii. Install frontend npm packages

ix. Copy over the following files:

   1. `config/local.json` for backend
   2. `cypress.env.json` for frontend