Continuous integration using GitHub Actions – Exercises

These short exercises intend to enable practising using the GitHub Actions platform and writing workflow files. We assume some Git & GitHub knowledge, and a GitHub account.

Tasks include links to relevant documentation. You may also look through the repository for ideas. If you wish to refer back to the slides, they are available online: https://docs.google.com/presentation/d/15V34-oJ1OuG3vjskW2kc_e9rcwHN_Vxe4IPLIUxBAhs/edit?usp=sharing

There are further reference materials at the end of this worksheet.

1. Amend a workflow

Fork the repository at https://github.com/raehik/ga-ci to your GitHub user, and clone it locally however you prefer.

This repository stores a handful of short example Python programs.

Look in the `.github/workflows` folder. There are already some valid workflows present, but they aren't finished.

Note that to run workflows on forks, you must enable them via the Actions tab.

Add triggers

Open the `py-black.yaml` and `py-mypy.yaml` workflows. These have jobs configured, but only have a manual `workflow_dispatch` trigger. Amend both workflows so that they run on every push to the main branch.

You can do this on the main branch on your fork. Check the Actions tab on your forked repository to see if jobs are running as expected (remember to push your changes to see them run).

Relevant documentation

- https://docs.github.com/en/actions/using-workflows/events-that-trigger-workflows#push
- If you have problems working with the repo locally, you can make simple changes directly on the GitHub website.

Merge workflows

Both workflow files are now very similar: they have identical triggers and a similar platform (Python on Linux). Merge the two workflows into a single file.

You may decide whether to put them as separate jobs in a single workflow file, or merge both jobs together (perform one check, then the other). The latter is perhaps more thoughtful and energy efficient, but the former might finish faster.

2. Write a new workflow

Fork and clone the repository at https://github.com/raehik/expr-ssa .

This repository stores a small Haskell program together with a handful of tests. Look at the `README.md` file for instructions on building and running tests. You *do not need* to run the tests locally.

Write a workflow that runs the tests, and is triggerable. You may use any triggers you like, but make sure you can test it easily – a `pull_request` trigger is always handy.

Consider using an existing workflow (e.g. from the previous repo) as a template, and altering the job.

Relevant documentation

- There is a pre-written step that ensures your platform (Windows/Mac/Linux) is set up to compile Haskell `haskell/actions/setup` . An example `step` snippet may be:
  ```
      - name: Setup Haskell build environment
        uses: haskell/actions/setup@v2
        with:
          ghc-version: "9.0"
  ```
- https://docs.github.com/en/actions/using-workflows/events-that-trigger-workflows#push
- https://github.com/haskell/actions/tree/main/setup

Extension task

As an extension, make your workflow run on Windows, Mac and Linux. You may make separate jobs manually, or use the `matrix` feature like so to write a single job that gets "expanded" for each OS:

```yaml
jobs:
  job:
    strategy:
      matrix:
        os:
        - ubuntu-latest
        - ...
        ver:
        - 9.4.2
        - ...
    runs-on: ${{ matrix.os }}
    name: Run on ${{ matrix.os }} using version ${{ matrix.ver }}
    steps:
    - # ...
```

Relevant documentation

- https://docs.github.com/en/actions/using-jobs/using-a-matrix-for-your-jobs

Refer to the links below for different options for `runs-on`, and how to use the `matrix` strategy to essentially parameterize your jobs.

(Some runs may be delayed or take a long time to finish. If a run is still going after 3 minutes, assume it's correct and simply taking its time.)

Reference material

The GitHub Actions documentation includes a syntax reference, along with common usage guides and full tutorials. Here are some select pages which might come in handy.

- workflow (+YAML) syntax
  https://docs.github.com/en/actions/using-workflows/workflow-syntax-for-github-actions
- `job.runs-on` runner images/platforms available for use
  https://github.com/actions/runner-images

Also take a look at the slides for some compact references on workflow and YAML syntax.