CS 1XC3 Lab 8 : Continuous Integration Testing using Github

Mar. 20th-24th, 2023

Contents

Introduction

Over the last few labs, we have explored automatically executed shell scripts in various forms. We explored the scripts used by Bash, and we've learned how to use Makefiles to queue up complex sets of actions and dependencies. This week, we will learn how GitHub itself can be configured to use automatic actions.

GitHub has a large number of very useful and interesting interactive tutorials. Today we will be doing two of them.

Technologies

Over the course of this exercise, you will encounter types some technologies and types of files that we haven't discussed in class. In general, knowledge of these technologies is unnecessary to complete these activities, but some references are provided in case you'd like to do some further reading.

- Docker and Dockerfiles
  - A more modern iteration of make and makefiles which integrates well with web services such as GitHub.
  - `https://www.docker.com`
  - `https://docs.docker.com/engine/reference/builder/`
- YAML files
  - A file used for data transmission. Essentially, it takes Python's philosophy on indentation and brackets applied to JSON files.
  - `https://yaml.org` - Yaml info in a Yaml file.
  - `https://en.wikipedia.org/wiki/YAML`
  - `https://www.cloudbees.com/blog/yaml-tutorial-everything-you-need-get-started`
- Node.js
  - A runtime environment for javascript that allows it to run outside a web browser.
  - `https://en.wikipedia.org/wiki/Node.js`
  - `https://nodejs.dev/en/learn/`

Don't worry too much about it if this seems intimidating. To be a software developer, you need to not be stressed out by code you don't understand or haven't had time to look into.

1 Activity: GitHub Actions Hello World [3 points]

In this section, we will learn the basics of setting up Github Actions.

- `https://github.com/skills/hello-github-actions`

Navigate to the above link and complete the tutorial, with the following additional conditions:

- Do not modify any of the default scripts or values which the tutorial gives you.
- Once you complete the tutorial, take the repository you have created and make it private.
- Then, add it as a submodule inside your `CS1XC3` master submission repository, in a new "L08" directory.

2 Activity: Continuous Integration in Github [7 points]

In this section, we will learn about how to set up continuous integration testing, which is one part of state-of-the-art software development.

- `https://github.com/skills/continuous-integration`

Navigate to the above link and complete the tutorial. Submit your work in the same manner as you did for part 1.

HINTS:

- Be mindful of which branch you're on.
- After having made a commit, you can take a look at the actions as they are running in the actions tab.

3 Activity: Github Actions with C and Makefiles [5 points]

Now, let's apply our new knowledge of Github Actions to C development!

1. First, create a new private repository on Github, called "CS1XC3_L08_P3".
2. Copy over all of your files from Lab 7 and commit them. This includes:
   - `src/`
   - `test_data/`
   - `Makefile`
   - any other scripts you may have written.
3. Go to the actions tab. Github will notice that your repo appears to be a project in C, and suggest some workflows.
4. Find "C/C++ with Make" and select "Set up this workflow".
5. You will have to move this file to ".github/workflows/" in order for github actions to detect it.
6. Change the `name` field of the generated dockerfile to something appropriate of your choice.
7. Edit the dockerfile such that, on push and pull request, the `make` rules `all`, `test`, and `clean` will be invoked. To do this, you'll have to delete the steps that have been automatically provided.
8. Push something to the repo (the change can be trivial, such as adding a comment or touching a file). Ensure that Github is running your make action correctly.
9. Add this repo as a submodule of your CS1XC3 master submission repo in the "L08" directory.

4 Activity: Trying to Annoy CPPcheck [5 points]

Sometimes the best way to learn about bugs is to try to create them intentionally.

- If your system doesn't have cppcheck on it already (the Pascal server does), install it! `http://cppcheck.sourceforge.net`
- In a new directory, "CS1XC3/L08/part4", write a C program, "worst.c" that triggers all the following cppcheck warnings. Triggering more than just these is fine. Run cppcheck with the `-enable=all` flag.
  - Out of bounds array access.
  - Incrementing a Boolean variable.
  - Matching `if` and `else if` conditions.
  - Zero division.

Commit your file to complete this lab activity!