COURSEWARE

GitHub Actions

Contents

Overview

GitHub Actions is CI (Continuous Integration) feature built into GitHub. You can use Actions to perform scripts executed in a dockerized build containers.

Every GitHub account has a limited free tier which allows you take advantage of this great feature.

Tutorial

To set up GitHub Action, click "Actions" tab in your project.

By default GitHub presents predefined actions wizard for you. We will skip it and create our new action from a scratch.

After you click "Skip this and set up a workflow yourself" link, you will see a simple wizard for creating new custom action. By default wizard proposes to create an action definition in `.github/workflows/main.yml` file. You can name your action file as you want as long as you keep it in `.github/workflows` directory.

Action wizard proposes a basic action definition for you. Keep in mind that `runs-on` section can be used to specify which build Docker image should be used for the build. For the sake of this tutorial we will use an Ubuntu Docker image.

If you scroll down an action definition a bit, you will see default Bash scripts proposed by an action wizard. These scripts just print *hello world* messages to the standard output. Let's keep them as they are and see if we can execute them!

Click green "Start commit" button.

And confirm committing action into a version control.

Now you should see our new action created. Its current status should be yellow, which indicate that our action scheduled its first execution.

After some time our yellow dot should turn into a green circle indicating that action was executed successfully. Let's click it then!

You should see the screen with a summary of action executions.

Click the succeeded execution icon.

You should see our two scripts listed. Click their icons to extends them and see an output they generated.

As you can see our hello world scripts have been successfully executed and printed a desired output.

Now lets get rid of the button which will allow us to manaully trigger a workflow and schedule a trigger for our workflow every 5 minutes (This is the shortest interval we can run scheduled workflows).

First, let's delete line 14 in our `.github/workflows/main.yml`

```
12
13    # Allows you to run this workflow manually from the Actions tab
14    workflow_dispatch:
15
```

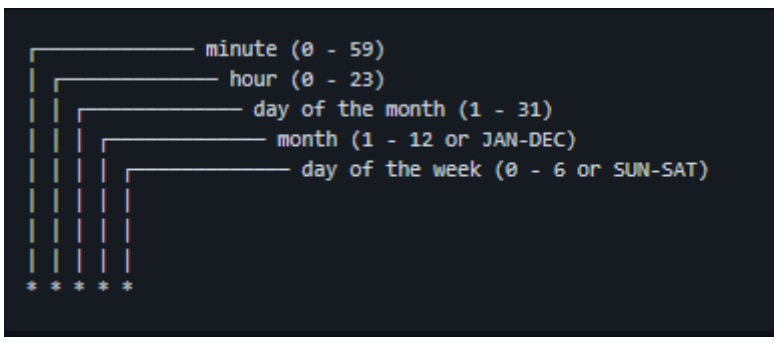
Replace it with:

```
14    schedule:
15      - cron: '0/5 * * * *'
```

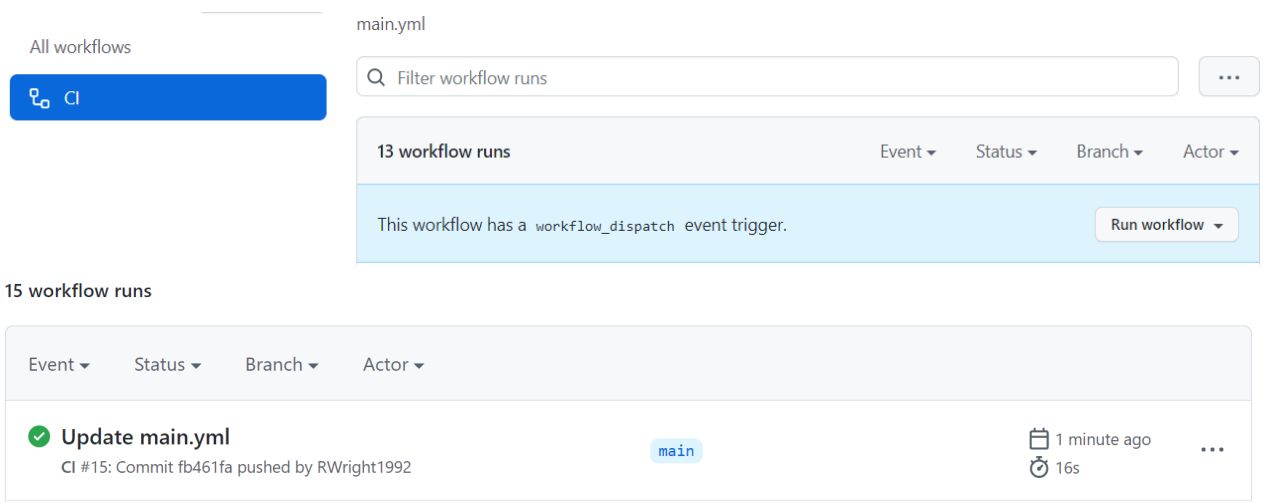
Now commit the file

This will run the workflow every 5 minutes starting from minute 0 through to 59.

Cron syntax has five fields seperated by a space with each field representing a unit of time.

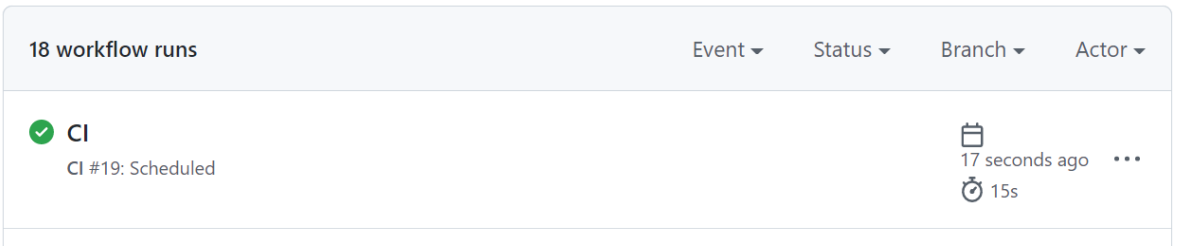


We will now see that the workflow button has gone.

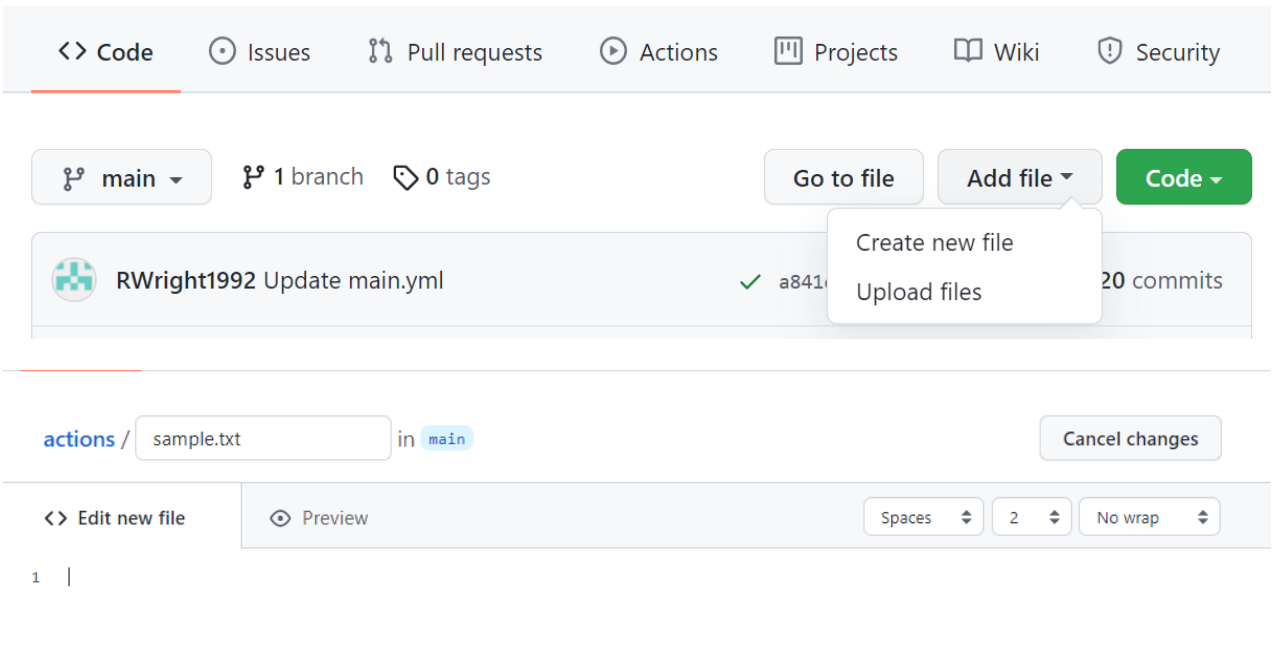


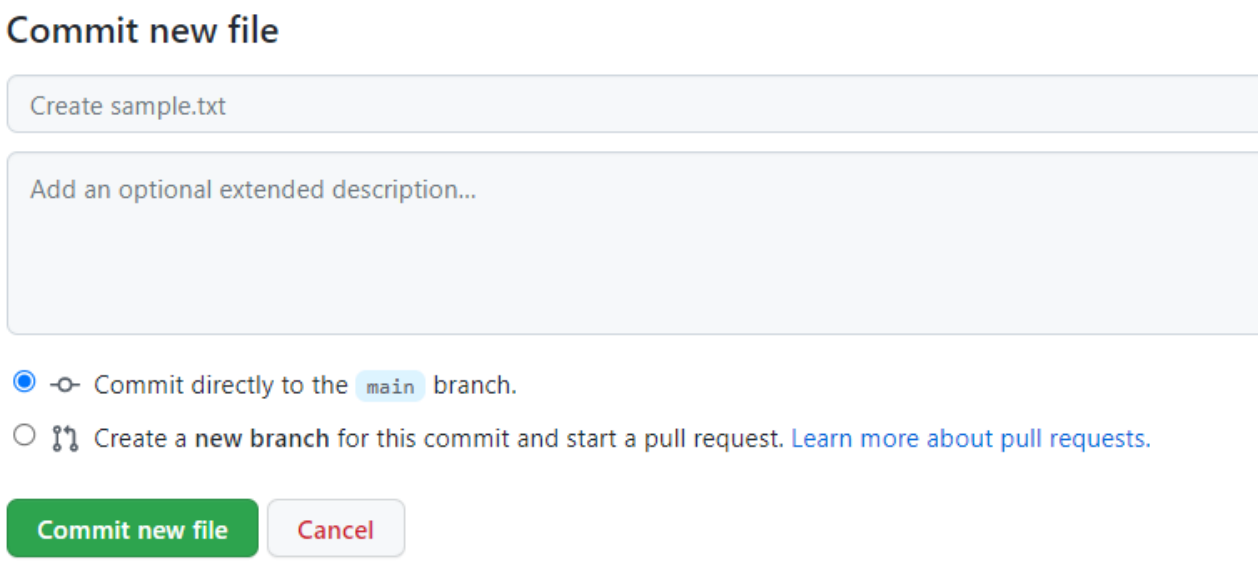
If we wait 5 minutes we should see our job has triggered

> Note: Sometimes there might be a slight delay to triggering this workflow



Now lets upload a blank text file called `sample.txt` to our repository.

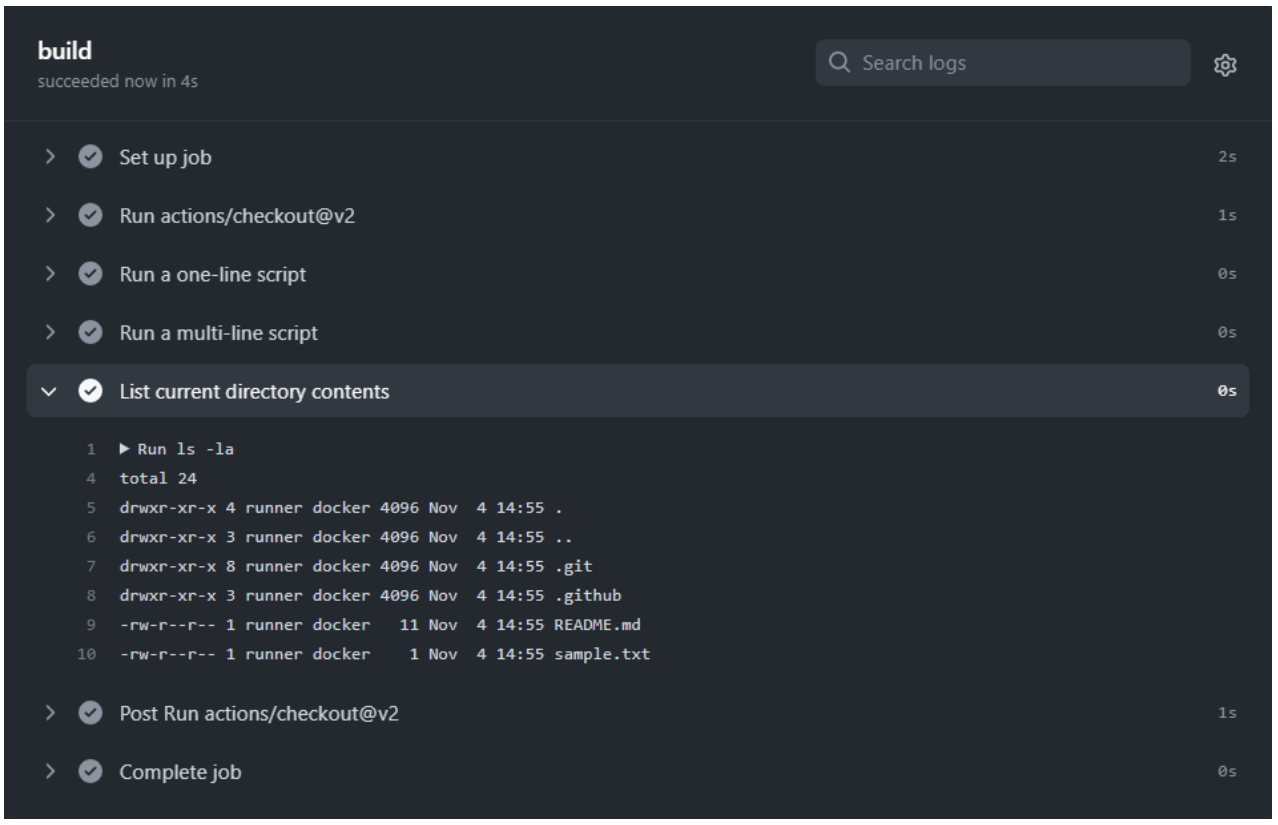




After this lets add a step to our workflow to list all the current directory of the file and commit it.

```
29          # Runs a single command using the runners shell
30          - name: Run a one-line script
31            run: echo Hello, world!
32
33          # Runs a set of commands using the runners shell
34          - name: Run a multi-line script
35            run: |
36              echo Add other actions to build,
37              echo test, and deploy your project.
38
39        - name: List current directory contents
40          run: ls -la
```

Once this has been commited we can see see from our workflow that the files in the container match up with what we have in our repository.



Finally, let's change our schedule trigger back to what it originally was in our `.github/workflows/main.yml`, so our workflow does not keep triggering every 5 minutes

```
12
13    # Allows you to run this workflow manually from the Actions tab
14    workflow_dispatch:
15
```

Exercises

- Add the following code to the project you created in the tutorial to a file called `script.sh`.

```
#!/bin/bash
echo "I am being run from a script"
touch file1.txt file2.txt file3.txt
echo "Created file1.txt, file2.txt & file3.txt"
echo "Listing current directory contents"
ls
```

- Now change the `.github/workflows/main.yml` file to add a step at the end which will execute the `script.sh` file we uploaded to our project

▶ Hint

- Push these changes to the remote repository. (Upon the executing of the action did our first step of listing the directories contents include the file1.txt, file2.txt & file3.txt?)
- Now edit the `.github/workflows/main.yml` again to included a step at the start which will remove file2.txt.
- Push these changes to the remote repository
- Did our action build successfully this time? If not, why?
- Please correct `.github/workflows/main.yml` file for this to execute correctly and push to remote repository