SWE_261P_ Project_Part_4
Team Members:
· Yuxin Huang: yuxinh20@uci.edu
· Changhao Liu: liuc50@uci.edu
· Ruokun Xu: ruokunx@uci.edu
Team Name: OffersAreHere
Github link: https://github.com/yx-hh/commons-io

What is Continuous Integration?

Continuous Integration allows developers to automatically test and build the latest software artifact when there is a new commit. It continuously and proactively checks the codebase to find the latest code change and triggers a build accordingly(Jones, pg4). If all the test cases pass as expected and the build is successful, the continuous integration tool will deploy the latest artifact to an environment such as development, test, or production environments.

Purpose of Continuous Integration

It's quite essential to learn about the reasons for using continuous integration for software development. Here are some reasons:

- It allows multiple software developers or teams to collaborate with one another, as it automatically builds the latest version of a software artifact developed by the team(Jones, pg4).
- It boosts developer productivity by automating the testing, building, and deploying pipeline so that developers don't need to manually trigger a build or deploy(Jones, pg4).
- It helps developers to maintain different versions of the artifact and to check the build status of each version.
- It assists developers in finding out which commit breaks the build and which test cases fail so that potential bugs are detected in the early stage(Jones, pg5), preventing introducing them to real customers.
- It delivers the latest features to customers quickly if those updates pass all the tests and build successfully(Jones, pg6).
- It guides developers to locate bugs(Jones, pg5), as the continuous integration tool usually displays information about which commit breaks the test case with error logs.

Continuous Integration Tool

We signed up for Github Action as our CI/CD platform to help us automate the testing, building, and deploying process. With Github action, we can easily build workflows, with multiple

sequential or parallel jobs, that are triggered by certain events such as new code changes being pushed to the codebase or a new pull request being created.

Set up Github Actions

First, we enabled Github Actions in GitHub repo as you can see from Figure 1.

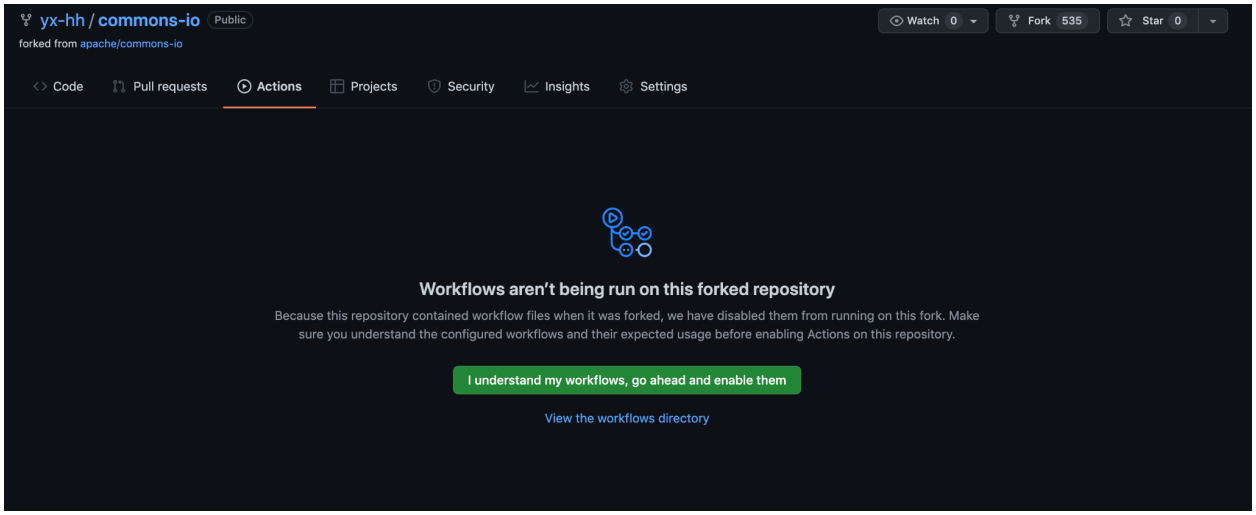

Figure 1: Enable Github Actions for our repository, From Github, By Changhao Liu

Then we created a Github Action configuration file: maven.yml under the folder of .github/workflows in our repository to configure Github Action. It checks whether there is a "push" event to our repository. If so, it will run the "**mvn package -Drat.skip=true"** command to run all the test cases, build and package our project sequentially: Github Actions will automatically execute the maven package command and upload jar files to the staging folder from which we can download our artifacts.

```
# workflow name
name: Java CI

# listen to push operation to master

on:
 push:
  branches: [ master ]

# jobs need to execute when occur above operations
jobs:
 build:
  runs-on: ubuntu-latest # choose server
```

```yaml
steps:
  - uses: actions/checkout@v2
  - name: Set up JDK 8
    uses: actions/setup-java@v2
    with:
      java-version: '8'
      distribution: 'adopt'
  - name: Build with Maven
    run: mvn package -Drat.skip=true
  - run: mkdir staging && cp target/*.jar staging
  - uses: actions/upload-artifact@v2
    with:
      name: Package
      path: staging
```

Figure 2, Maven.yml by, By Yuxin Huang

By clicking a specific workflow in the Github Action's dashboard, we are able to check the detailed information of each build triggered by different commits in Figure 3. As for this workflow, the build was successful and took 6 minutes and 14 seconds to complete. To retrieve the artifact, we just need to click the "Package" button to retrieve our item. If we have a deployment environment for this project, we can also set up the workflow in Github Actions to deploy our artifact directly to our environments such as development, test, or production.

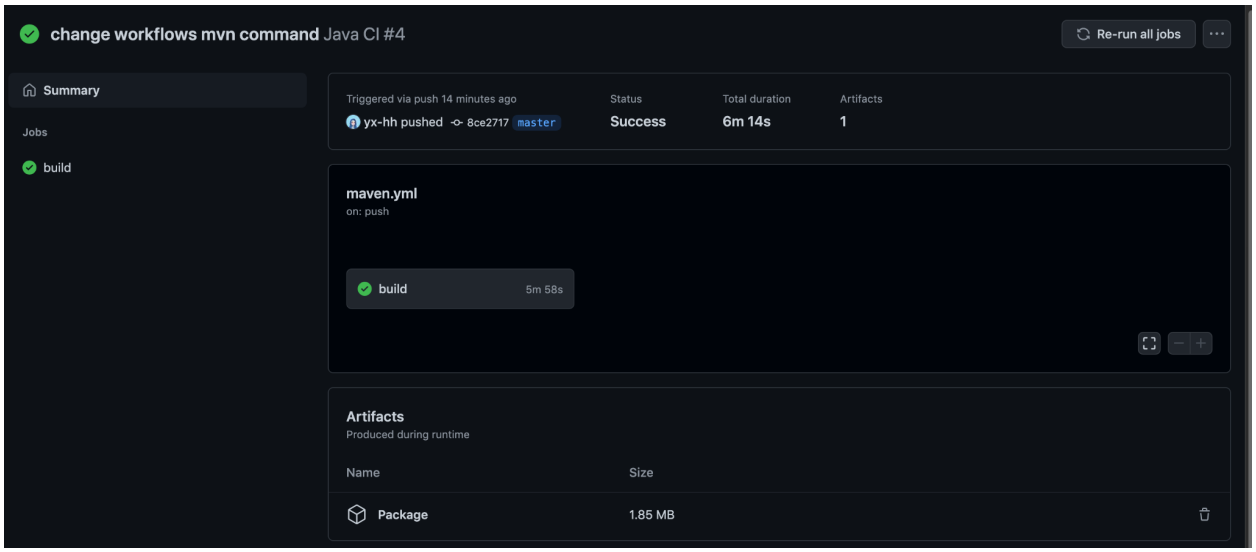

Figure 3, Build Success Info of Github Actions, By Yuxin Huang

By clicking the "build" button, we can see each workflow's steps as in Figure 4.

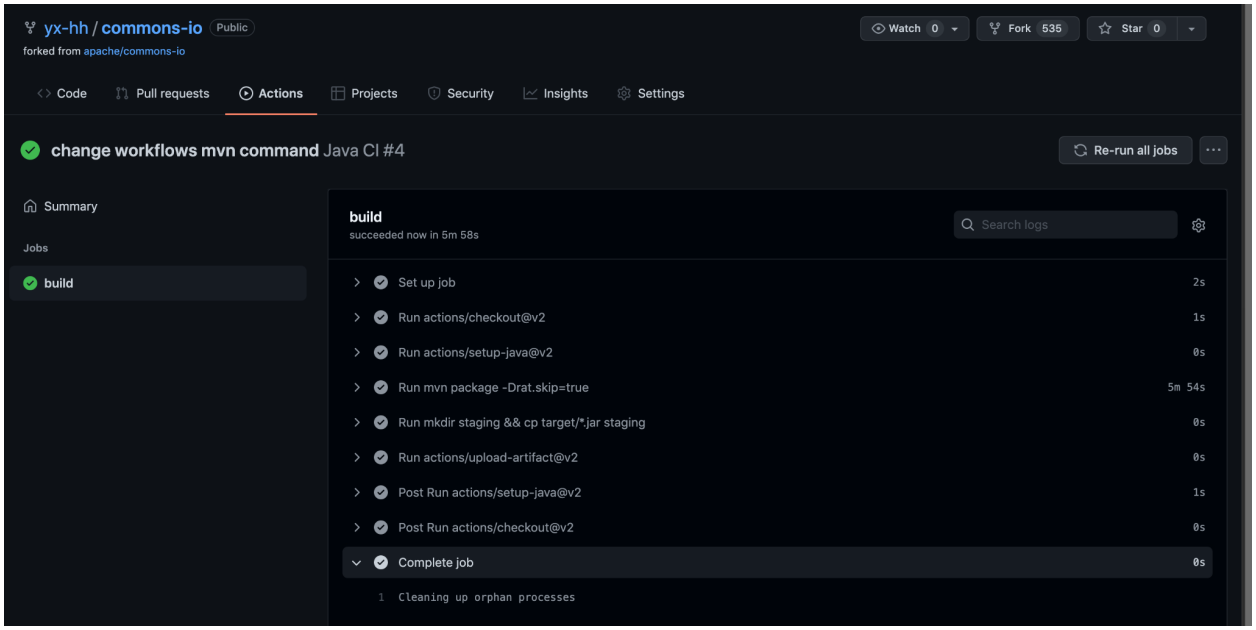

Figure 4, Workflow Jobs in Github Actions, By Yuxin Huang

To check the detailed log that records testing and build status as in Figure 5, we can read the log by unfolding the maven package command line.

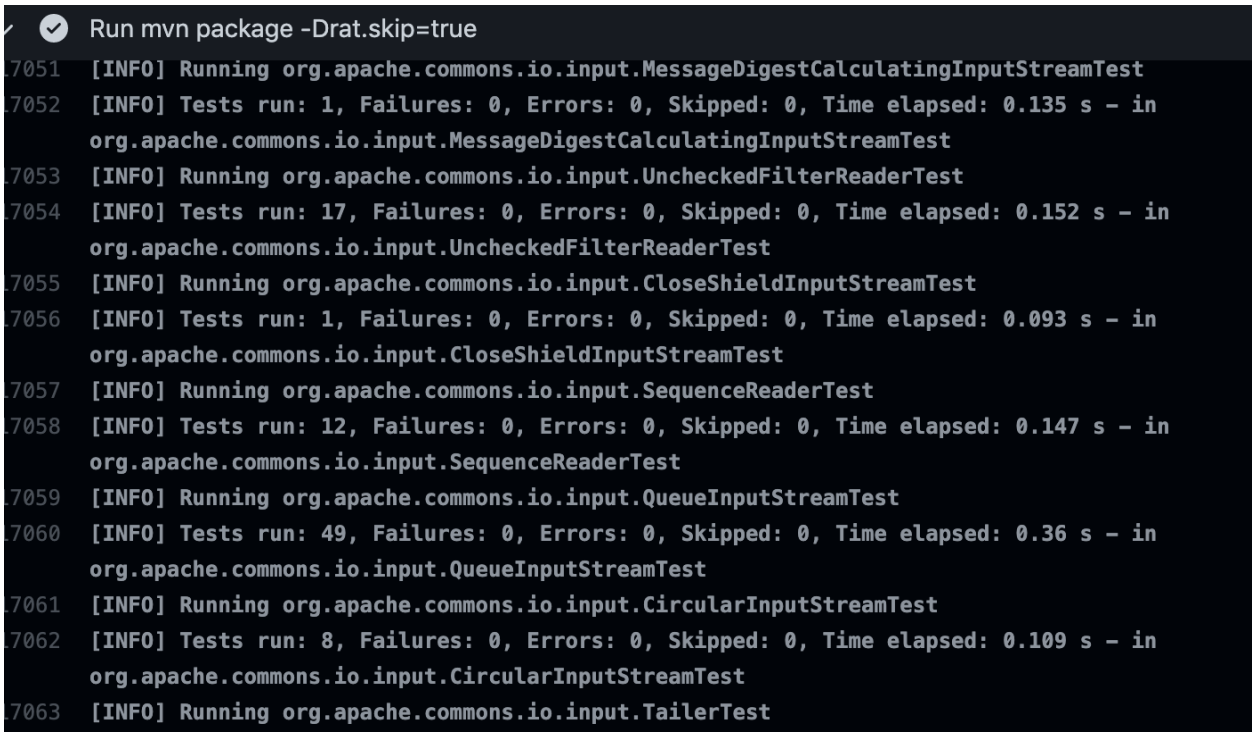

Figure 5, Test and Build Status, By Yuxin Huang

Problems we encountered

We faced a Maven command error which was caused by maven package command incorrect, and downloading some maven packages took a long time.

Reference

Jones, J. (2022b). ***Continuous Integration*** **[Slides]. Canvas.**

https://canvas.eee.uci.edu/courses/43617/files/folder/Lecture%20Slides?preview=17655785

Building and testing Java with Maven**. (2012). GitHub Docs.**

https://docs.github.com/en/github-ae@latest/actions/automating-builds-and-tests/building-and-testing-java-with-maven