Continuous Integration

2024.04.11

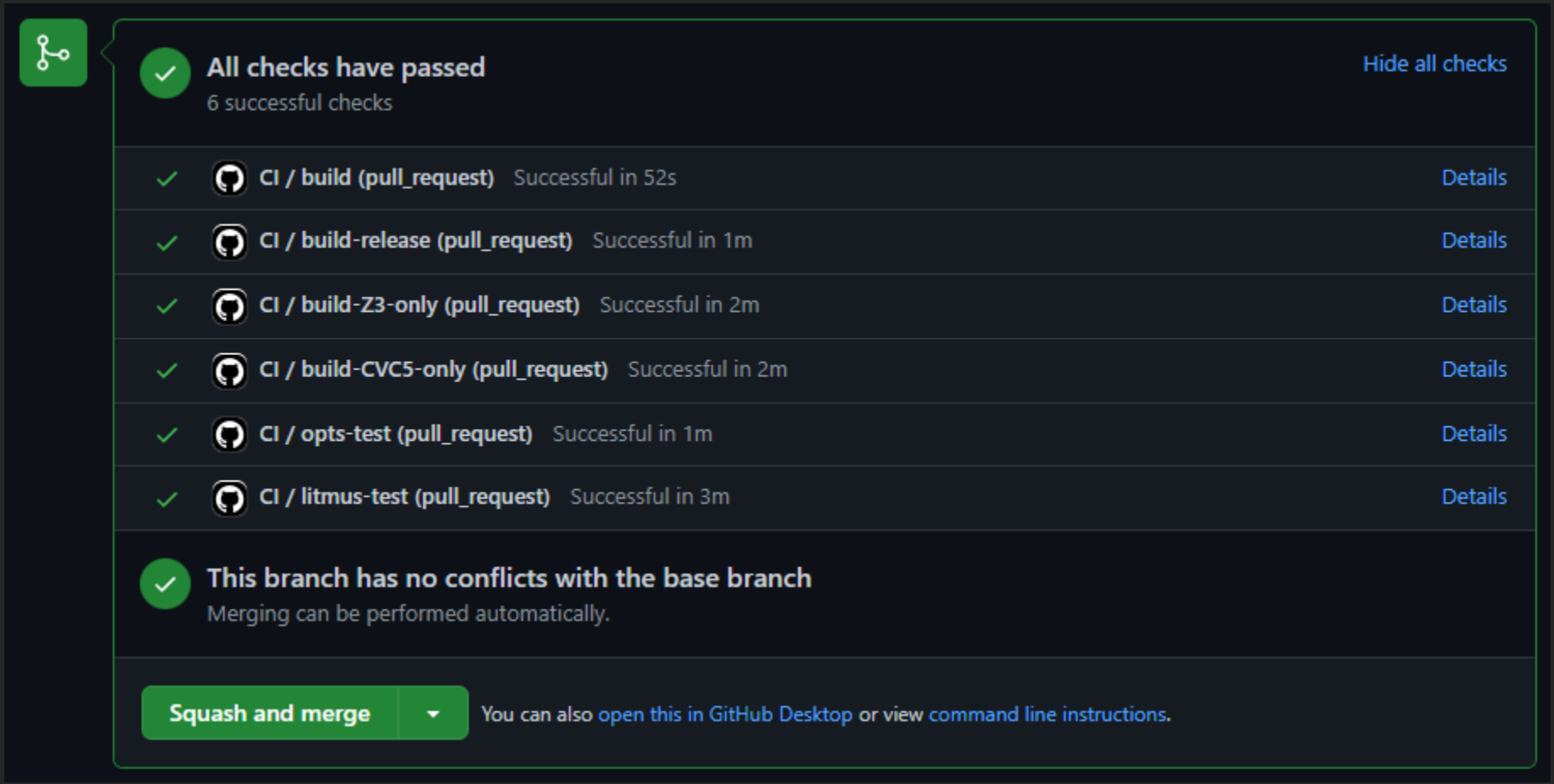
SWPP Practice Session

Seunghyeon Nam

Continuous Integration?

- Continuous: whenever a code is uploaded to the repo…
  - Whenever a commit or a pull request is made
- Integration: check the validity of the code after merging
  - Through automated building & testing

Example of CI in Action



Example of CI in Action

Summary

Jobs

- build
- build-release
- build-Z3-only
- build-CVC5-only
- opts-test
- litmus-test

opts-test

succeeded 18 days ago in 2m 6s

- Set up job
- Initialize containers
- Fetch cache
- Run ctest

```
1  ▶ Run cd /src/       /build
6  Test project /src/       /build
7        Start  1: Opts-conv2d-to-img2col
8   1/12 Test  #1: Opts-conv2d-to-img2col ...............   Passed   19.41 sec
9        Start  2: Opts-convert-elementwise-to-linalg
10  2/12 Test  #2: Opts-convert-elementwise-to-linalg ...   Passed    0.21 sec
11       Start  3: Opts-fold-memref-subview-op
12  3/12 Test  #3: Opts-fold-memref-subview-op ..........   Passed    2.11 sec
13       Start  4: Opts-fold-tensor-extract-op
14  4/12 Test  #4: Opts-fold-tensor-extract-op ..........   Passed    0.50 sec
15       Start  5: Opts-fusion-tensor
```

Example of CI in Action

master

Commits on Mar 31, 2022

Fix `linalg.fill` operands to follow the updated syntax (#321)
committed 18 days ago ✓ Verified 8d5a5d7

Commits on Mar 30, 2022

Analyze memref outputs (#320)
committed 19 days ago ✗ Verified 2b2c7d0

Update CMakeLists.txt
committed 19 days ago ✗ Verified 191c221

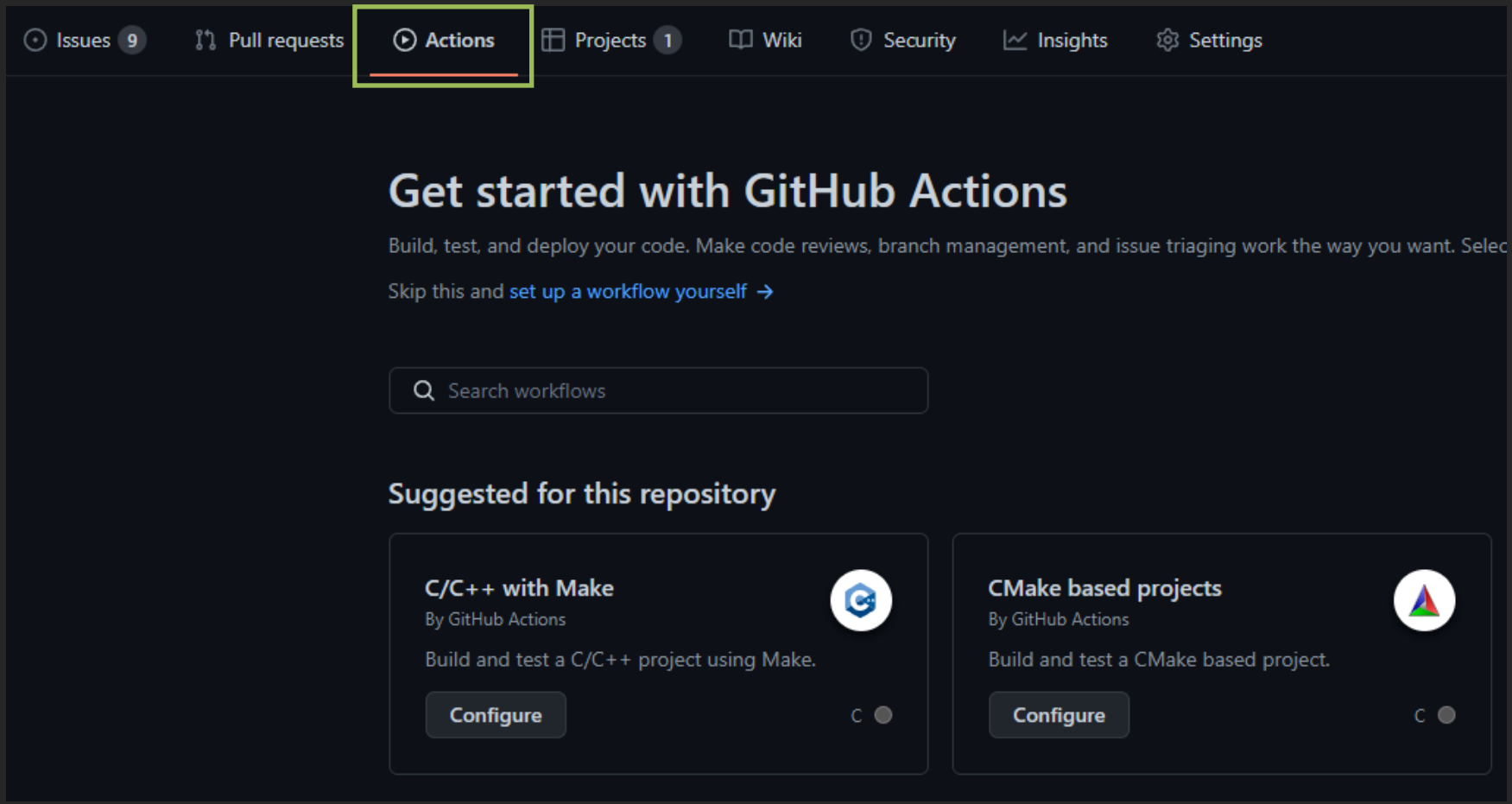
Commits on Mar 29, 2022

Fix test failures due to the update in linalg.fill
committed 21 days ago ✗ 12c9421

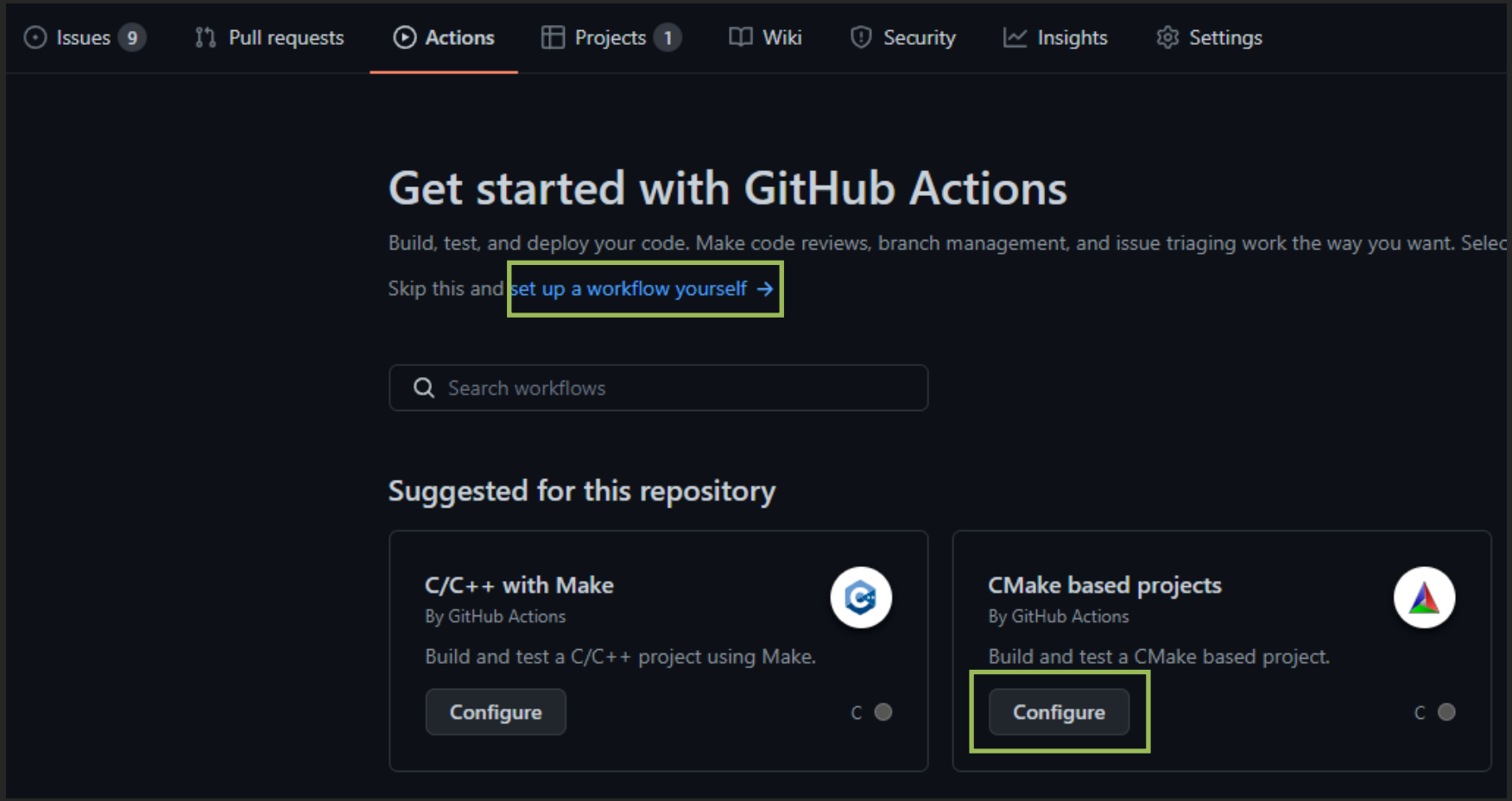
Commits on Mar 28, 2022

Migrate from Std to Func dialect
committed 21 days ago ✗ 23832af

Configuring GitHub Actions



Configuring GitHub Actions



Configuring GitHub Actions

Code | Issues 9 | Pull requests | Actions | Projects 1 | Wiki | Security | Insights | Settings

/ .github / workflows / ci.yml in main

Edit new file | Preview | Spaces | 2 | No wrap

```yaml

env:
  # Customize the CMake build type here (Release, Debug, RelWithDebInfo, etc.)
  BUILD_TYPE: Release

jobs:
  build-test:
    runs-on: ubuntu-latest
    container: strikef/swpp202201-ci

    steps:
    - uses: actions/checkout@v3
    - name: Configure CMake
      run: cmake -B ${{github.workspace}}/build -DCMAKE_BUILD_TYPE=${{env.BUILD_TYPE}}
    - name: Build
      run: cmake --build ${{github.workspace}}/build
    - name: Test
      working-directory: ${{github.workspace}}/build
      run: ctest -C ${{env.BUILD_TYPE}}
```

Use Control + Space to trigger autocomplete in most situations.

Configuring GitHub Actions

Code | Issues 9 | Pull requests | Actions | Projects 1 | Wiki | Security | Insights | Settings

/ .github / workflows / ci.yml in main

Workflow

Edit new file | Preview | Spaces | 2 | No wrap

```yaml

env:
  # Customize the CMake build type here (Release, Debug, RelWithDebInfo, etc.)
  BUILD_TYPE: Release

jobs:
  build-test:
    runs-on: ubuntu-latest
    container: strikef/swpp202201-ci

    steps:
    - uses: actions/checkout@v3
    - name: Configure CMake
      run: cmake -B ${{github.workspace}}/build -DCMAKE_BUILD_TYPE=${{env.BUILD_TYPE}}
    - name: Build
      run: cmake --build ${{github.workspace}}/build
    - name: Test
      working-directory: ${{github.workspace}}/build
      run: ctest -C ${{env.BUILD_TYPE}}
```

Job

Step

Use Control + Space to trigger autocomplete in most situations.

Terminology: Workflow

- Specify when the tasks will be run (on section)
  - Every day or every hour (cron)
  - Whenever a pull request or commit is uploaded (event)
  - When someone wants (dispatch)
- Contains one or more jobs
  - Jobs can be given dependencies between them

Terminology: Job

- Unit of tasks
  - Build the project
  - Run this test, run that test, …
- Contains one or more steps

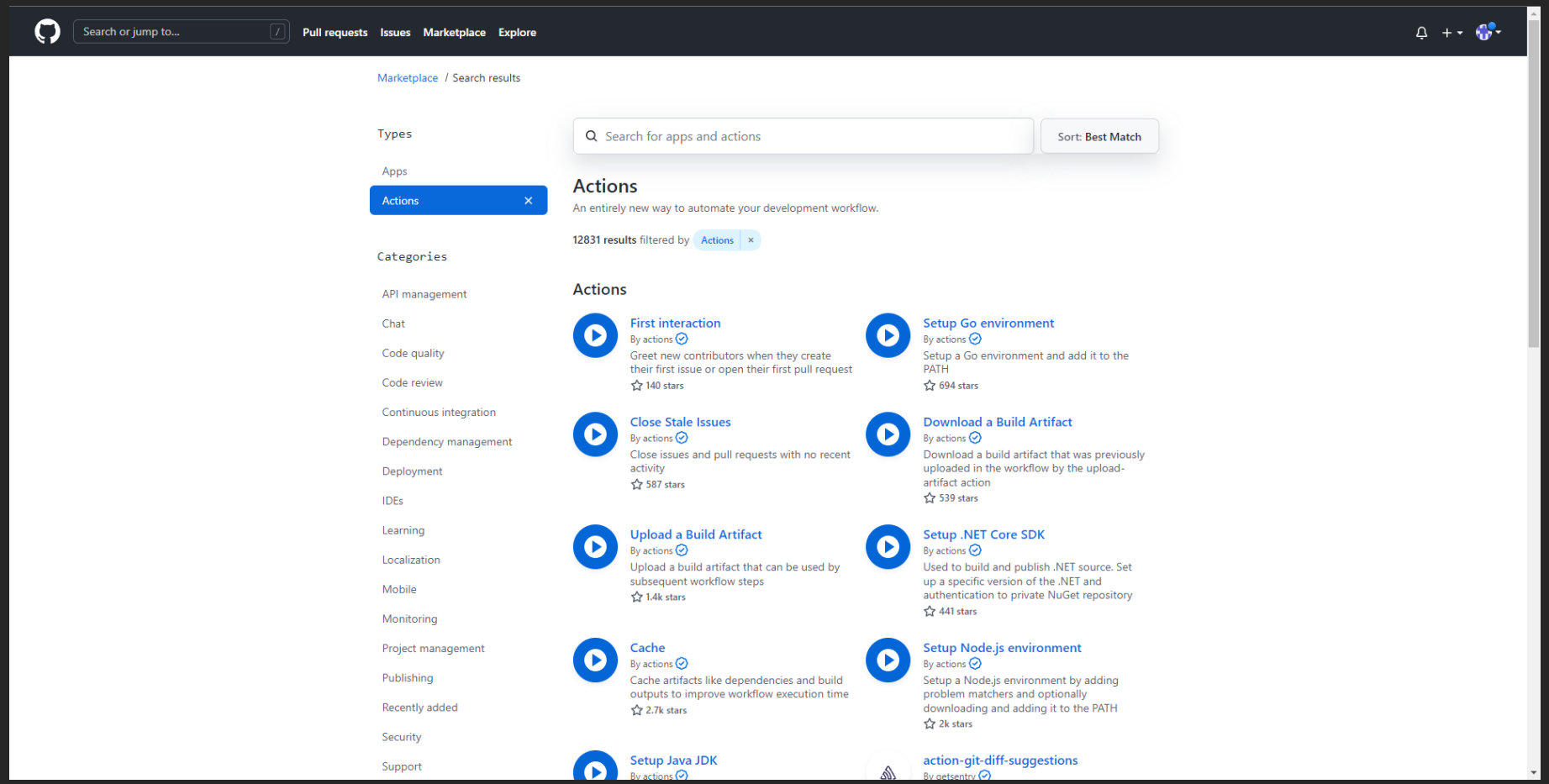
Terminology: Job

- Supports control flow
  - Steps can be skipped under certain conditions
- Different jobs can run in different environments
  - Can configure build test for multiple system
- Different jobs can be run in parallel!
  - They can be synchronized via dependency (control flow!)

Terminology: Step

- Order of commands
  - Execute Action
  - Execute shell command(s)
- Executed linearly
  - By default, previous step(s) should succeed
  - Result of previous step(s) can be used in another step(s)

Actions in GitHub Actions



`actions/checkout`

- Clone the repository
  - Can clone from specific repo, branch, or commit
  - Much quicker than cloning through shell command
- See the official repo for more details

actions/cache

- Cache the file or directory for future use
  - Manage the entries via keys
  - Entries of different keys are stored separately
  - Each entry can be written only once
  - Keys can be fully or partially matched
- See the official repo and guide for more details

actions/cache

- Available within the entire repository
  - Jobs can use the cache written by another job
  - Workflows can use the cache written by another workflow
  - Can be used to separate the time-consuming task
- 10GB capacity limit per repository
  - Older entries will be evicted upon reaching the limit

Actions Instance

- Each repository can use 2,000 minutes per month for free
  - Should be more than sufficient for the team project
- But you have to set up the instance for yourself
  - You need LLVM, Alive2, and whole lot of dependencies
  - You may unknowingly deviate from others & TAs' environment!
- Solution: use Docker in Actions!

Docker

- Container (OS-level virtualization) solution
  - Essentially an isolated Linux environment
- Creates a container based on image
  - Image: read-only disk (similar to ISO files)
  - Container: VM instance created from the image
- Note: you access the container as `root` by default

Docker Image

- Images can be created using the script named Dockerfile
- Or you can simply download the image from the Docker Hub
  - Like GitHub, you can get the image from the repository
  - Each repository contains several tags
  - Each tag corresponds to an image (or version)

SWPP Docker Image

- We'll distribute the Docker image for the project
  - It will be based on ubuntu 22.04
  - It will contain pre-built LLVM and almost all necessary deps
  - You can use it for both development and CI
- Set the `container` option to use it in GitHub Actions
  - Specify the image repo name, and you're done!