

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




**All checks have passed**
6 successful checks

[Hide all checks](#)

	 CI / build (pull_request) Successful in 52s	Details
	 CI / build-release (pull_request) Successful in 1m	Details
	 CI / build-Z3-only (pull_request) Successful in 2m	Details
	 CI / build-CVC5-only (pull_request) Successful in 2m	Details
	 CI / opts-test (pull_request) Successful in 1m	Details
	 CI / litmus-test (pull_request) Successful in 3m	Details

**This branch has no conflicts with the base branch**
Merging can be performed automatically.

Squash and merge 

You can also [open this in GitHub Desktop](#) or [view command line instructions](#).

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

Issues 9 Pull requests Actions Projects 1 Wiki Security Insights Settings

Get started with GitHub Actions

Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you want. Select a workflow to get started.

Skip this and [set up a workflow yourself](#) →

Search workflows

Suggested for this repository

C/C++ with Make

By GitHub Actions

Build and test a C/C++ project using Make.

Configure

C ●

CMake based projects

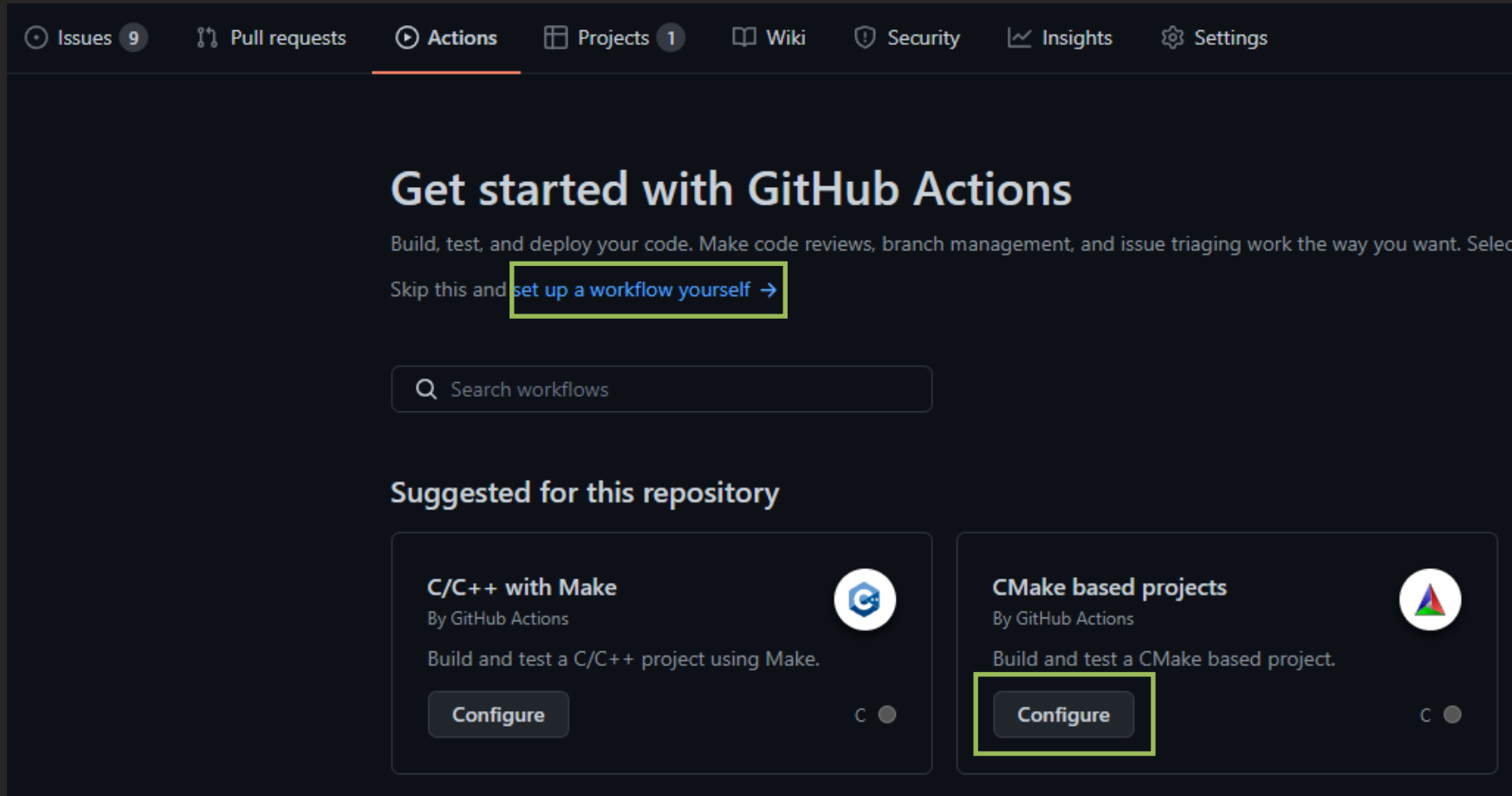
By GitHub Actions

Build and test a CMake based project.

Configure

C ●

Configuring GitHub Actions



The screenshot shows the GitHub Actions interface. The top navigation bar includes links for Issues (9), Pull requests, Actions (highlighted with an orange underline), Projects (1), Wiki, Security, Insights, and Settings. The main heading is 'Get started with GitHub Actions'. Below it, a text block says 'Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you want. Select Skip this and [set up a workflow yourself →](#)', where the link is highlighted with a yellow box. A search bar labeled 'Search workflows' is present. The 'Suggested for this repository' section contains two cards. The first card, 'C/C++ with Make', has a 'Configure' button. The second card, 'CMake based projects', has a 'Configure' button highlighted with a yellow box. Both cards include the GitHub Actions logo and a description of the workflow.

Issues 9 Pull requests Actions Projects 1 Wiki Security Insights Settings

Get started with GitHub Actions

Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you want. Select Skip this and [set up a workflow yourself →](#)

Search workflows

Suggested for this repository

C/C++ with Make
By GitHub Actions

Build and test a C/C++ project using Make.

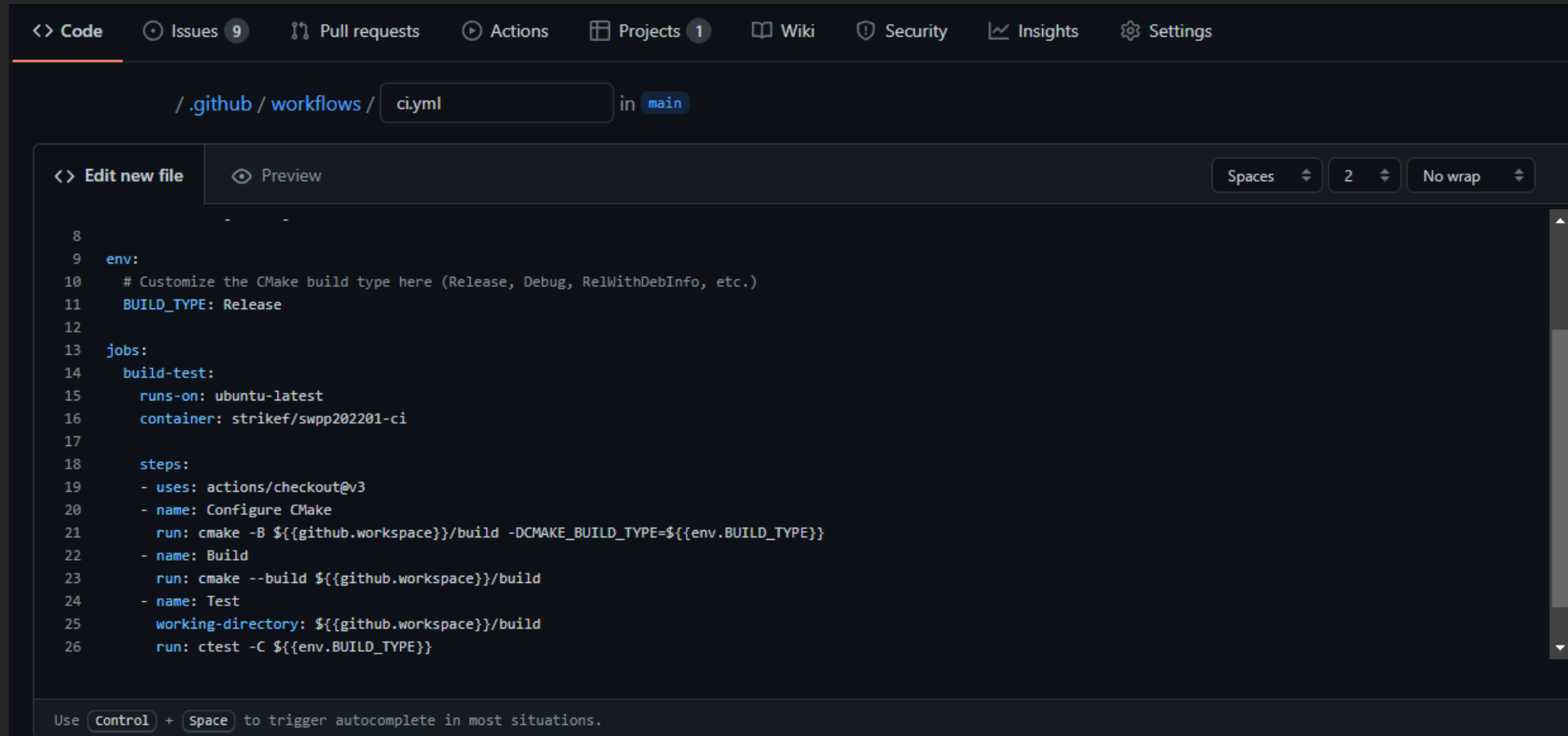
Configure

CMake based projects
By GitHub Actions

Build and test a CMake based project.

Configure

Configuring GitHub Actions



The screenshot shows the GitHub Actions configuration page for a workflow named `ci.yml` located in the `.github/workflows/` directory of the `main` branch. The interface includes a top navigation bar with links to Code, Issues (9), Pull requests, Actions, Projects (1), Wiki, Security, Insights, and Settings. Below the navigation bar, the file path `/.github/workflows/ci.yml` is displayed. The main area shows the workflow configuration in YAML format, with tabs for 'Edit new file' and 'Preview'. The configuration includes environment variables, a job named `build-test` running on `ubuntu-latest` within a specific container, and a series of steps for checking out code, configuring CMake, building, and testing. A footer note indicates that pressing `Control` + `Space` triggers autocomplete.

```
8
9  env:
10   # Customize the CMake build type here (Release, Debug, RelWithDebInfo, etc.)
11   BUILD_TYPE: Release
12
13  jobs:
14    build-test:
15      runs-on: ubuntu-latest
16      container: strikef/swpp202201-ci
17
18      steps:
19      - uses: actions/checkout@v3
20      - name: Configure CMake
21        run: cmake -B ${github.workspace}/build -DCMAKE_BUILD_TYPE=${env.BUILD_TYPE}
22      - name: Build
23        run: cmake --build ${github.workspace}/build
24      - name: Test
25        working-directory: ${github.workspace}/build
26        run: ctest -C ${env.BUILD_TYPE}
```

Use `Control` + `Space` to trigger autocomplete in most situations.

Configuring GitHub Actions

The screenshot shows the GitHub Actions configuration page for a workflow named `ci.yml` located in the `.github/workflows/` directory. The interface includes a top navigation bar with links to Code, Issues (9), Pull requests, Actions, Projects (1), Wiki, Security, Insights, and Settings. Below the navigation bar, the file path `/.github/workflows/ci.yml` is shown, along with a button to view the workflow in the `main` branch. The main content area is titled "Workflow" and contains a code editor with two tabs: "Edit new file" and "Preview". The code editor displays the following YAML configuration:

```
8
9 env:
10   # Customize the CMake build type here (Release, Debug, RelWithDebInfo, etc.)
11   BUILD_TYPE: Release
12
13 jobs:
14   build-test:
15     runs-on: ubuntu-latest
16     container: strikef/swpp202201-ci
17
18     steps:
19       - uses: actions/checkout@v3
20       - name: Configure CMake
21         run: cmake -B ${github.workspace}/build -DCMAKE_BUILD_TYPE=${env.BUILD_TYPE}
22       - name: Build
23         run: cmake --build ${github.workspace}/build
24       - name: Test
25         working-directory: ${github.workspace}/build
26         run: ctest -C ${env.BUILD_TYPE}
```

The code editor also features a "Spaces" dropdown set to 2 and a "No wrap" option. A yellow box highlights the `build-test` job, and an orange box highlights the `Configure CMake` step within that job. The word "Job" is written next to the job definition, and the word "Step" is written next to the step definition. At the bottom of the editor, a note states: "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

The screenshot displays the GitHub Actions marketplace. At the top, there's a navigation bar with the GitHub logo, a search bar, and links for Pull requests, Issues, Marketplace, and Explore. The main header shows 'Marketplace / Search results'. On the left, a sidebar lists 'Types' (Apps, Actions) and 'Categories' (API management, Chat, Code quality, Code review, Continuous integration, Dependency management, Deployment, IDEs, Learning, Localization, Mobile, Monitoring, Project management, Publishing, Recently added, Security, Support). The 'Actions' type is selected. The main content area features a search bar, a filter for 'Actions' (12831 results), and a grid of action cards. Each card includes a play button icon, the action name, a description, and the number of stars.

Action Name	Description	Stars
First interaction	Greet new contributors when they create their first issue or open their first pull request	140 stars
Setup Go environment	Setup a Go environment and add it to the PATH	694 stars
Close Stale Issues	Close issues and pull requests with no recent activity	587 stars
Download a Build Artifact	Download a build artifact that was previously uploaded in the workflow by the upload-artifact action	539 stars
Upload a Build Artifact	Upload a build artifact that can be used by subsequent workflow steps	1.4k stars
Setup .NET Core SDK	Used to build and publish .NET source. Set up a specific version of the .NET and authentication to private NuGet repository	441 stars
Cache	Cache artifacts like dependencies and build outputs to improve workflow execution time	2.7k stars
Setup Node.js environment	Setup a Node.js environment by adding problem matchers and optionally downloading and adding it to the PATH	2k stars
Setup Java JDK		
action-git-diff-suggestions		

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!