

# Continuous Integration

2023.04.06


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](#)

	 <b>CI / build (pull_request)</b> Successful in 52s	<a href="#">Details</a>
	 <b>CI / build-release (pull_request)</b> Successful in 1m	<a href="#">Details</a>
	 <b>CI / build-Z3-only (pull_request)</b> Successful in 2m	<a href="#">Details</a>
	 <b>CI / build-CVC5-only (pull_request)</b> Successful in 2m	<a href="#">Details</a>
	 <b>CI / opts-test (pull_request)</b> Successful in 1m	<a href="#">Details</a>
	 <b>CI / litmus-test (pull_request)</b> Successful in 3m	<a href="#">Details</a>

**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

# Example of CI in Action

The screenshot displays the commit history for the 'master' branch. The interface is dark-themed. At the top left, a button labeled 'master' with a dropdown arrow is highlighted with a yellow border. Below it, a vertical timeline shows commit dates: Mar 31, 2022; Mar 30, 2022; Mar 29, 2022; and Mar 28, 2022. Each date is preceded by a small icon of a branch with a circle. The commit details are shown in a list format. Each commit entry includes a title, a subtitle indicating the commit time and status, and a row of action buttons (Verified, Copy, Hash, Diff).

Date	Commit Title	Commit Subtitle	Verification	Copy	Hash	Diff
Mar 31, 2022	Fix linalg.fill operands to follow the updated syntax (#321)	committed 18 days ago ✓	Verified		8d5a5d7	
Mar 30, 2022	Analyze memref outputs (#320)	committed 19 days ago ✗	Verified		2b2c7d0	
	Update CMakeLists.txt	committed 19 days ago ✗	Verified		191c221	
Mar 29, 2022	Fix test failures due to the update in linalg.fill	committed 21 days ago ✗			12c9421	
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

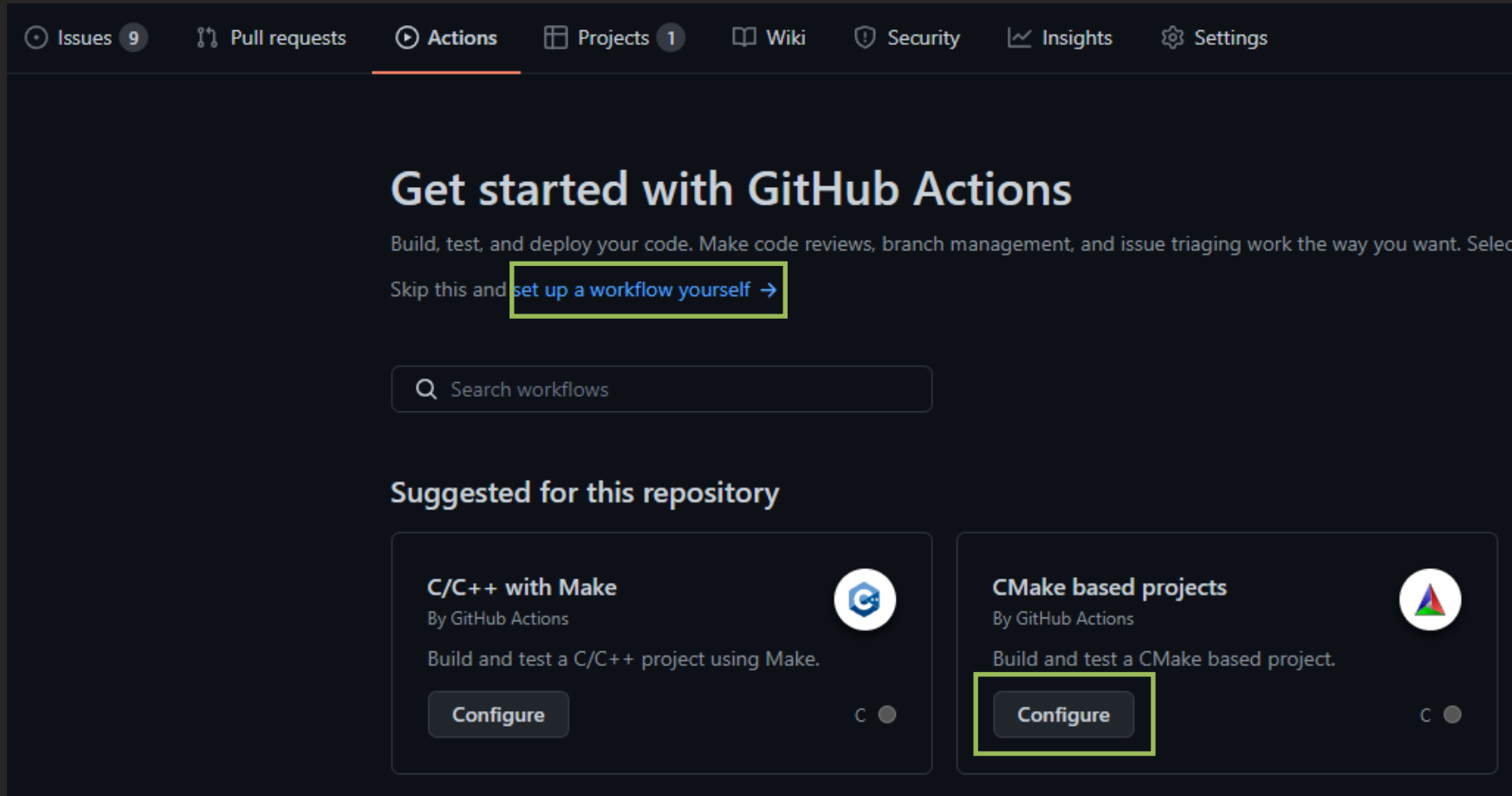
#### CMake based projects

By GitHub Actions

Build and test a CMake based project.

Configure

# 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 a workflow to get started.' A link 'Skip this and set up a workflow yourself →' is highlighted with a yellow box. Below this is a search bar labeled 'Search workflows'. The section 'Suggested for this repository' 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 show a 'C' icon and a toggle switch.

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 ●

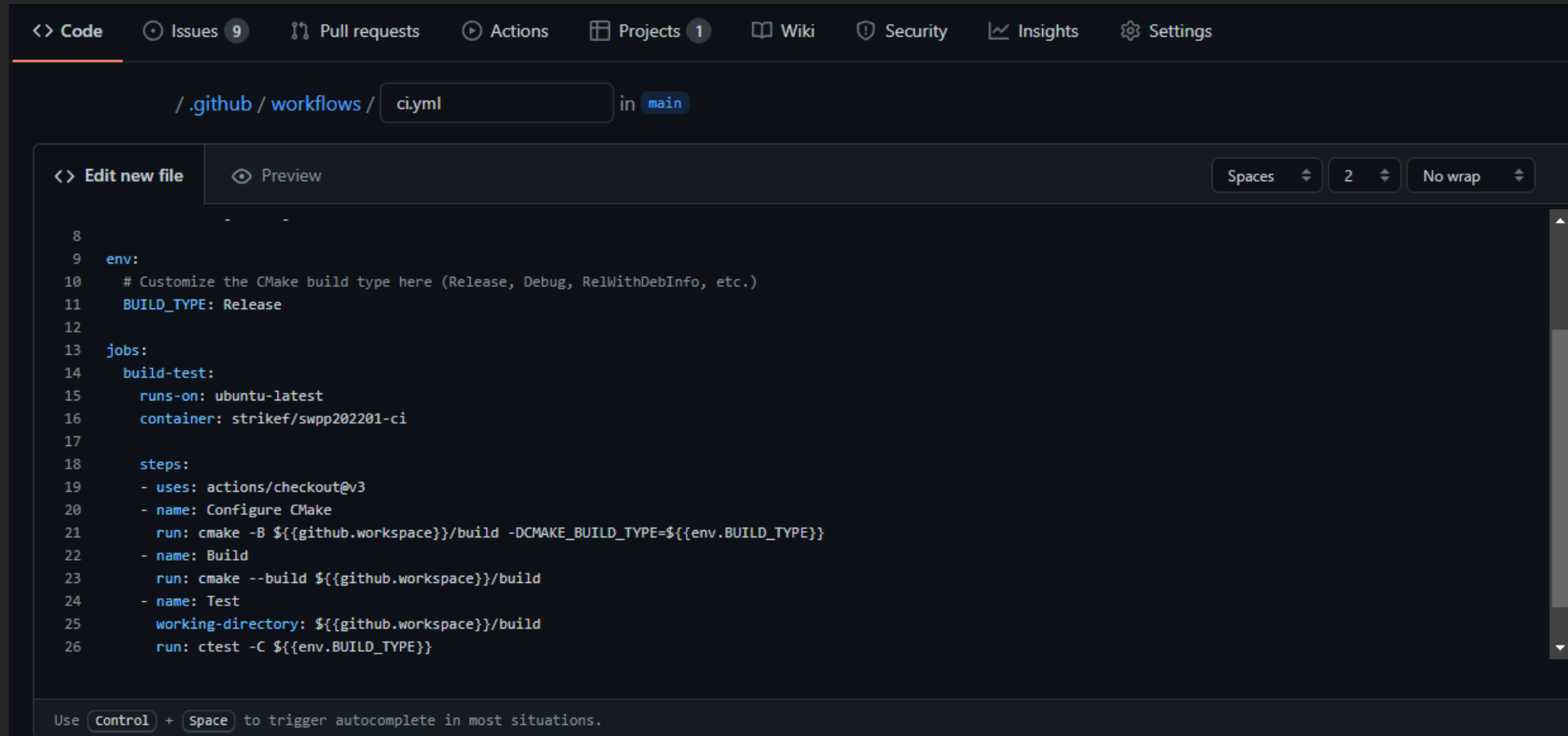
**CMake based projects**  
By GitHub Actions

Build and test a CMake based project.

Configure

C ●

# Configuring GitHub Actions



The screenshot shows the GitHub Actions workflow editor interface. At the top, there is a navigation bar with links to Code, Issues (9), Pull requests, Actions, Projects (1), Wiki, Security, Insights, and Settings. Below this, the breadcrumb path is `/.github / workflows / ci.yml` in the `main` branch. The editor has two tabs: `Edit new file` (active) and `Preview`. On the right side of the editor, there are settings for `Spaces`, `2` (lines), and `No wrap`. The main area contains a YAML workflow file with the following content:

```
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}}
```

At the bottom of the editor, a tip states: Use `Control` + `Space` to trigger autocomplete in most situations.



# Terminology: Workflow

- Determines **when** the tasks will be run
  - 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, ...
  - Different jobs can be run in parallel!
- Contains run one or more **steps**
  - Steps can be **skipped** under certain conditions

# 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 Marketplace section is active, showing 'Search results'. On the left, a sidebar lists various categories like API management, Chat, Code quality, etc. The main area features a search bar and a list of actions. The 'Actions' category is selected, showing 12,831 results. A grid of action cards is displayed, each with a play button icon, a title, a description, and a star count.

Marketplace / Search results

Types

Search for apps and actions

Sort: Best Match

Apps

Actions

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

**Actions**

An entirely new way to automate your development workflow.

12831 results filtered by Actions

**Actions**

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

# 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

# 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)

# Running the Container

- Install `podman`
  - Available on both `apt` and `brew`
- Pull the image from DockerHub or import one
  - `podman pull docker.io/<image-owner>/<image-name>`
- Create an interactive container from the image
  - `podman run -it /<image-owner>/<image-name>:<tag>`

# 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!