



CI/CD Pipeline for Automated Code Quality Checks

Using GitHub Actions

Project Description:

This project demonstrates the implementation of a **Continuous Integration (CI)** pipeline using **GitHub Actions** to automatically enforce coding standards and maintain code quality across Python projects. The core objective was to integrate a **code linting workflow** that is triggered on every code push to the repository. This was achieved using **GitHub Super-Linter**, an open-source GitHub Action that bundles multiple popular linters into a single action.

The workflow was designed to:

- **Automatically trigger** on each push event to the repository.
- **Check out the code** from the repository using the actions/checkout step.
- **Run Super-Linter**, which executed multiple linters including:
 - flake8 for Python code style.
 - black for Python code formatting.
 - isort for import order.
 - mypy for static type checking.
 - yamllint for YAML file validation.

The setup ensures that all code committed to the repository adheres to widely accepted Python standards (PEP8 and PEP257), formatted cleanly, and structured properly before merging into the main branch.

The project also included writing multiple Python files (main.py and greeting.py) and integrating them into the CI pipeline to validate the linting process and ensure the system works reliably for multiple files and commits.

THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

- GitHub repo creation

Create a new repository Try the new experience

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * Repository name *

Thisarak943 / mygitactions mygitactions is available.

Great repository names are short and memorable. Need inspiration? How about [fictional-goggles](#) ?

Description (optional)

Public Anyone on the internet can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

- Create workflow steps

mygitactions / .github / workflows / superlinter.yml in main

Edit Preview Spaces 2 No wrap

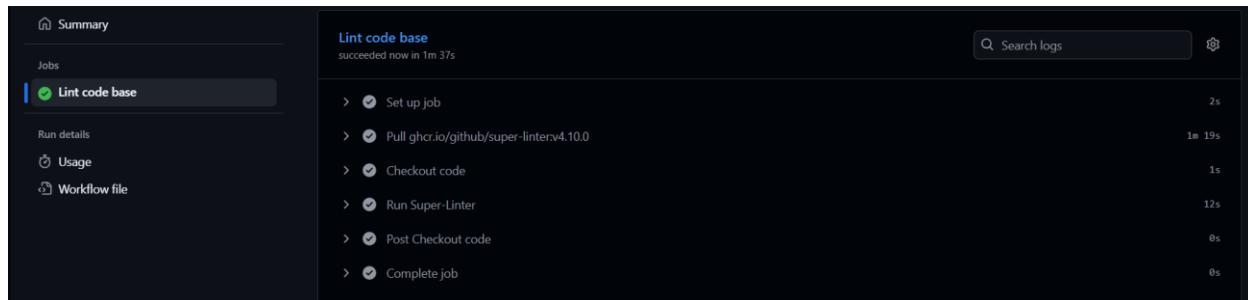
```
1 name: Super-Linter
2
3 on: push
4
5 jobs:
6   super-lint:
7     name: Lint code base
8     runs-on: ubuntu-latest
9     steps:
10       - name: Checkout code
11         uses: actions/checkout@v2
12
13       - name: Run Super-Linter
14         uses: github/super-linter@v4
15         env:
16           DEFAULT_BRANCH: main
17           GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

THISARA KANDAGE

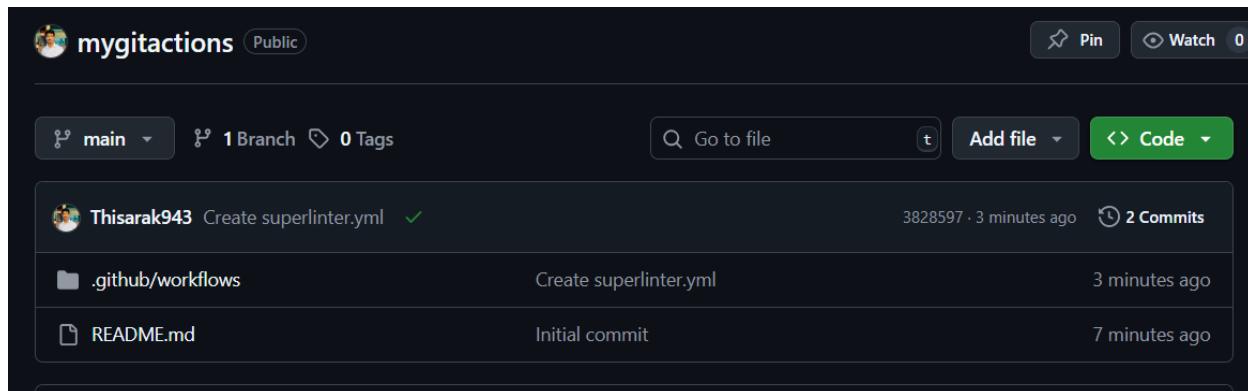
UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

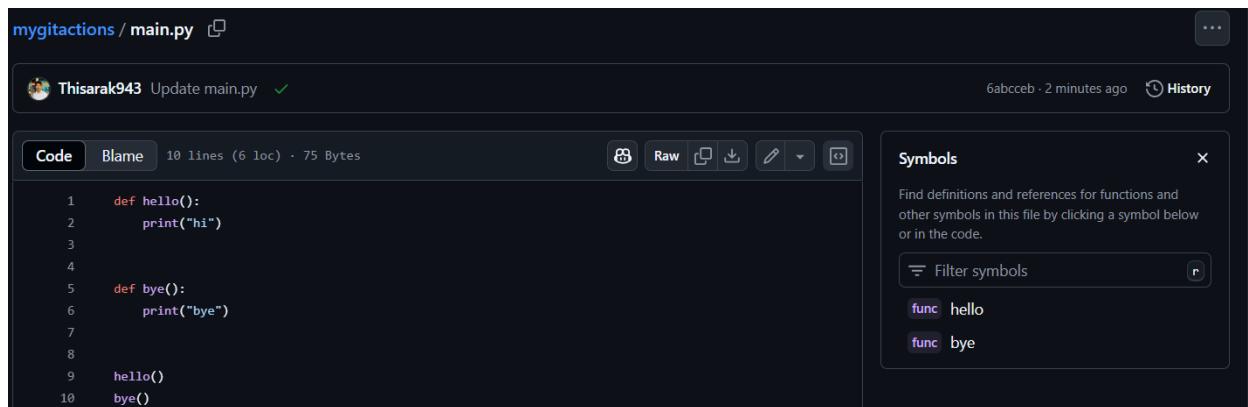
- After running actions in GitHub



- All the checks have been passed



- The tried with push some codes



THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

- After triggers running

The screenshot shows the GitHub Actions interface for a 'Lint code base' job. The job succeeded in 1m 43s. The steps listed are:

- > Set up job (2s)
- > Pull ghcr.io/github/super-linter:v4.10.0 (1m 21s)
- > Checkout code (1s)
- > Run Super-Linter (16s)
- > Post Checkout code (1s)
- > Complete job (0s)

The screenshot shows the 'All workflows' page on GitHub. It displays three workflow runs:

- Update main.py**: Status: Success (green checkmark), Triggered by Super-Linter #4, Commit 6abcceb pushed by Thisarak943, on branch main, 4 minutes ago, 1m 48s.
- Create main.py**: Status: Failed (red X), Triggered by Super-Linter #2, Commit 4a4ef20 pushed by Thisarak943, on branch main, 12 minutes ago, 1m 47s.
- Create superlinter.yml**: Status: Success (green checkmark), Triggered by Super-Linter #1, Commit 3828597 pushed by Thisarak943, on branch main, 20 minutes ago, 1m 40s.

- Check git actions again for another file

The screenshot shows the 'greeting.py' file in the 'mygitactions' repository. The code is:

```
1 def greet(name):
2     print(f"Hello, {name}!")
3
4
5 greet("Thisara")
```

THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

- After pushing code

4 workflow runs			
		Event ▾	Status ▾
Event	Status	Branch	Actor
✓ Update greeting.py	Success	main	3 minutes ago ⌚ 1m 41s
✓ Update main.py	Success	main	13 minutes ago ⌚ 1m 48s
✗ Create main.py	Failure	main	22 minutes ago ⌚ 1m 47s
✓ Create superlinter.yml	Success	main	29 minutes ago ⌚ 1m 40s

➡ What I Learned:

1. GitHub Actions CI/CD Concepts

- Gained practical experience in setting up and customizing GitHub Actions workflows.
- Learned how to configure event triggers (like on: push) to automate tasks.
- Understood the job structure: defining jobs, steps, and using actions.



THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website

2. Using Super-Linter for Code Quality

- Understood the role of Super-Linter in automating code analysis with tools like:
 - flake8 (Python style guide enforcement),
 - black (auto-formatting),
 - isort (organizing imports),
 - mypy (type hint checking),
 - yamlint (YAML validation).
- Learned how Super-Linter simplifies code quality checks with minimal configuration.
- Explored how Super-Linter logs errors clearly, helping developers fix issues quickly.

3. YAML Configuration and Workflow Best Practices

- Learned the YAML syntax and structure for defining workflows.
- Understood the importance of correctly structuring YAML files and including document start (---).
- Practiced best practices in writing maintainable and scalable workflow files.

4. Clean Code Practices

- Learned how to write Python code that complies with standard style guides (PEP8, PEP257).
- Understood the importance of consistent formatting and spacing (e.g., blank lines after functions).
- Gained familiarity with using linters to catch subtle issues that improve readability and maintainability.

5. End-to-End Automation Mindset

- Understood how CI/CD helps enforce coding discipline in team environments.
- Realized how integrating linting tools in the pipeline reduces manual code review overhead.
- Learned how automation tools support scalability and quality control in real-world development workflows.



THISARA KANDAGE

UNDERGRADUATE - SLIIT

E-mail LinkedIn GitHub Website