CICD Pipeline

Repo URL: https://github.com/Rajiv-mar121/ML-Ops Group35

Enabled our repo "ML-Ops_Group35" with CICD pipeline. Added few steps like below:

Linting: Using flake8 linting is used in repository. It analyses the code for potential error bugs, stylistic errors, and other issues.

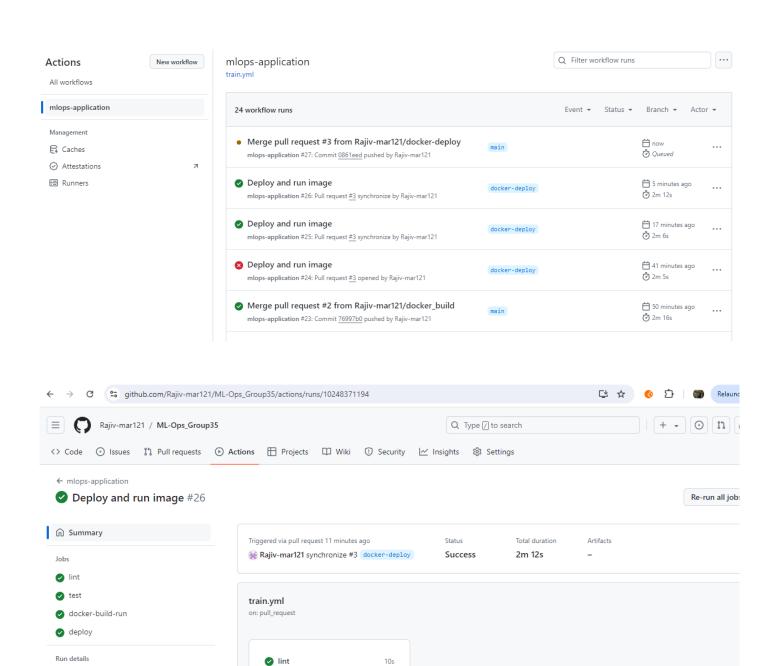
- Error Detection: Linters can catch syntax errors, undefined variables, and other common mistakes.
- **Code Consistency**: Consistent style guide, code becomes easier to read and maintain, especially when working in a team.
- Best Practices: Linters can enforce best practices and help avoid common pitfalls in Python programming.
- Readability: Consistent formatting and style make code easier to read and understand
- Maintainability: Clean and consistent code is easier to debug, extend, and maintain over time.

Testing: "pytest" Framework is used verify that code works as expected by writing and running tests. Proper testing is crucial for maintaining code quality, catching bugs early, and ensuring that new changes do not break existing functionality.

- **Unit Testing**: Tests individual units or components of code in isolation. Each test verifies a single function or method's behaviour.
- **Integration Testing**: Tests how different pieces of the system work together. It often involves multiple components interacting with each other.
- **Functional Testing**: Tests the functionality of the entire application to ensure it behaves as expected from the user's perspective.
- **End-to-End Testing**: Tests the complete flow of an application, simulating real user scenarios.
- Acceptance Testing: Validates the system against the requirements and checks if it meets the acceptance criteria.
- Regression Testing: Ensures that new changes do not break existing functionality.

Deploying: A deployment pipeline in GitHub Actions automates the process of building, testing, and deploying application.

- **Checkout Code**: Retrieve the latest code from the repository.
- **Set Up Environment**: Prepare the environment, including installing dependencies and setting up necessary tools.
- **Build**: Compile and package the application.
- **Test**: Run automated tests to ensure the code is working as expected.
- Build Docker Image: If using containers, create a Docker image of the application.
- Push Docker Image: Push the Docker image to a container registry like DockerHub or AWS ECR.
- **Deploy**: Deploy the application to the desired environment (e.g., a cloud server or Kubernetes cluster).



55s

2m 0s

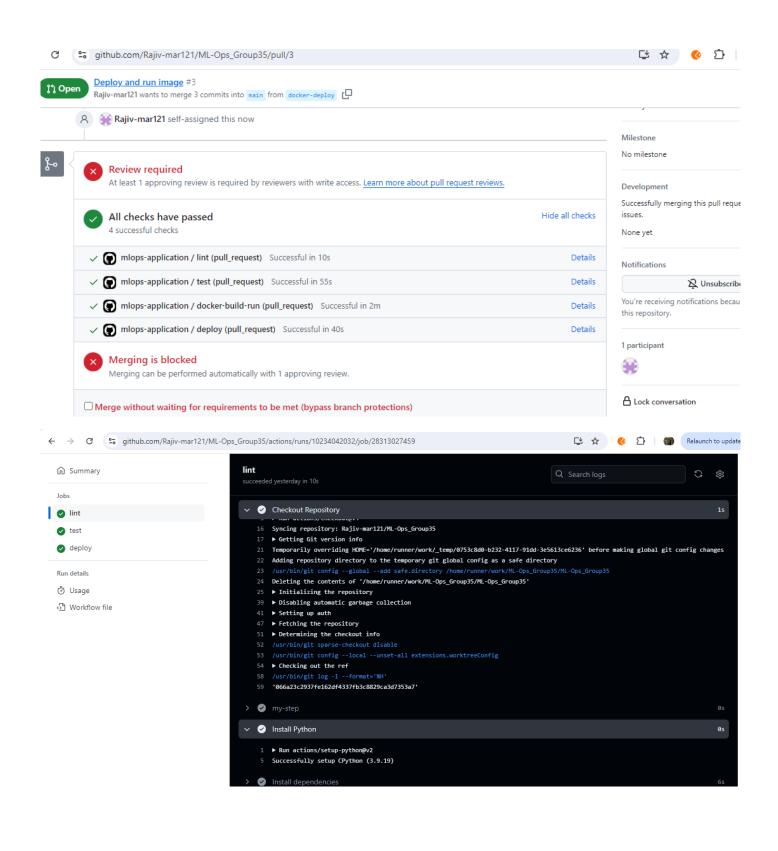
40s

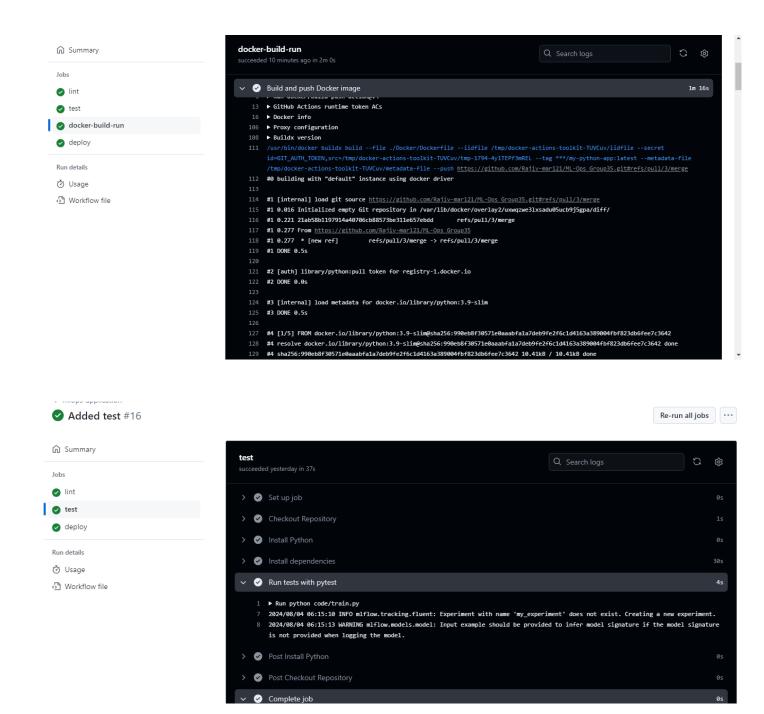
test

deploy

docker-build-run

🕭 Usage



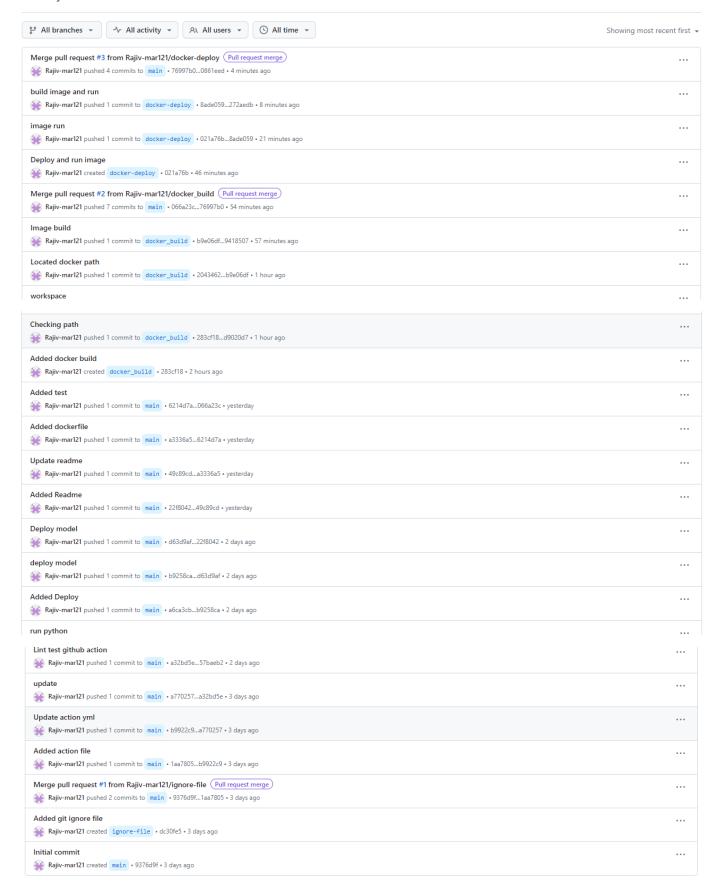


GIT Repo Link

https://github.com/Rajiv-mar121/ML-Ops_Group35

Activity which show Pull request and commits:

Activity



All Github Repository Branches:

Default					
Branch	Updated	Check status	Behind Ahead	Pull request	
main 🗗 🛈	9 minutes ago	• 0/4	Default		Ů …
Your branches					
Branch	Updated	Check status	Behind Ahead	Pull request	
docker-deploy [14 minutes ago	√ 4/4	1 0	% #3	Ů …
docker_build C	1 hour ago	√ 4/4	5 0	% #2	₾ …
ignore-file CD	🥡 3 days ago		28 0	% #1	ů ···
Active branches					
Branch	Updated	Check status	Behind Ahead	Pull request	
docker-deploy [14 minutes ago	✓ 4/4	1 0	% #3	ů ···
docker_build C	↑ hour ago	√ 4/4	5 0	№ #2	⊕ …