

# Assignment #5 - CI/CD

## (Group Assignment: FullStack Alchemists)

**Shivam Hasurkar - 017413214**

**Aditya Kulkarni - 016904537**

**Aditi Patil - 017454164**

### GitOps: An Overview and Case Study

GitOps is a paradigm or a set of practices that empowers developers to perform tasks which typically (in traditional environments) fall under the purview of IT operations. GitOps aims to provide a developer-centric experience in managing infrastructure and applications, by using tools developers are already familiar with (like Git).

In GitOps, Git is used as the single source of truth for declarative infrastructure and applications. With the code-as-infrastructure approach, changes to infrastructure are made through commits and merges in a Git repository. Automated processes ensure that the state of the system matches the state described in the Git repository.

### Case Study: Goibibo Employs GitOps for Streamlined Travel Services

Goibibo is one of India's leading online travel booking platforms, offering services from flight and train bookings to hotel reservations. With a vast customer base and a high volume of transactions, they needed a robust system to manage their services and infrastructure reliably.

#### Challenges

- Goibibo's microservices architecture meant that they had to manage a multitude of services and deployments, which became increasingly complex.
- The deployment process was cumbersome, and scaling up services quickly to handle peak loads was challenging.
- Ensuring consistency and reducing downtime during service updates were critical to maintaining customer satisfaction.

#### Implementation of GitOps

Goibibo implemented GitOps using tools like Jenkins X, Helm, and Kubernetes to automate their deployment pipelines and manage infrastructure.

#### Benefits Realized

- **Improved Deployment Frequency:** With GitOps, Goibibo increased their deployment frequency, ensuring that new features and updates reached customers faster.
- **Scalability:** Automatic scaling of services became more manageable, allowing Goibibo to efficiently handle fluctuating traffic volumes.

- **Operational Efficiency:** Developers were able to use pull requests to manage changes, streamlining the review and deployment process.
- **Enhanced Stability:** The declarative approach of GitOps meant that the production environment was less prone to human error, leading to increased stability.

## Outcome

The adoption of GitOps enabled Goibibo to maintain a high pace of innovation while ensuring their platform remained stable and reliable. The automated processes and clear audit trails provided by GitOps practices allowed them to manage their complex infrastructure with greater confidence and efficiency.

## 2. Pros and Cons of CI/CD Tools

### Jenkins

#### Pros:

Highly customizable with a vast ecosystem of plugins.  
Open-source and widely adopted.

#### Cons:

Complex setup and steeper learning curve.  
Can become resource-heavy with many plugins.

### CircleCI

#### Pros:

Hosted solution with a quick setup process.  
Good integration with various VCS systems.

#### Cons:

Limited build minutes and resources in the free plan.  
Might be cost-prohibitive for larger teams or more extensive projects.

### TravisCI

#### Pros:

Easy to start with for open-source projects.  
Configuration as code with .travis.yml.

#### Cons:

Recently had significant changes in pricing models, affecting some open-source projects.  
Limited Windows and macOS support compared to other providers.

### GitHub Actions

#### Pros:

Seamless integration with GitHub repositories.  
Free minutes and storage for public repositories.

#### Cons:

Still relatively new, with some features under active development.  
Potential for lock-in with the GitHub ecosystem.

## Why we used GitHub Actions

Given the context of our small project, we opted for GitHub Actions for below reasons:

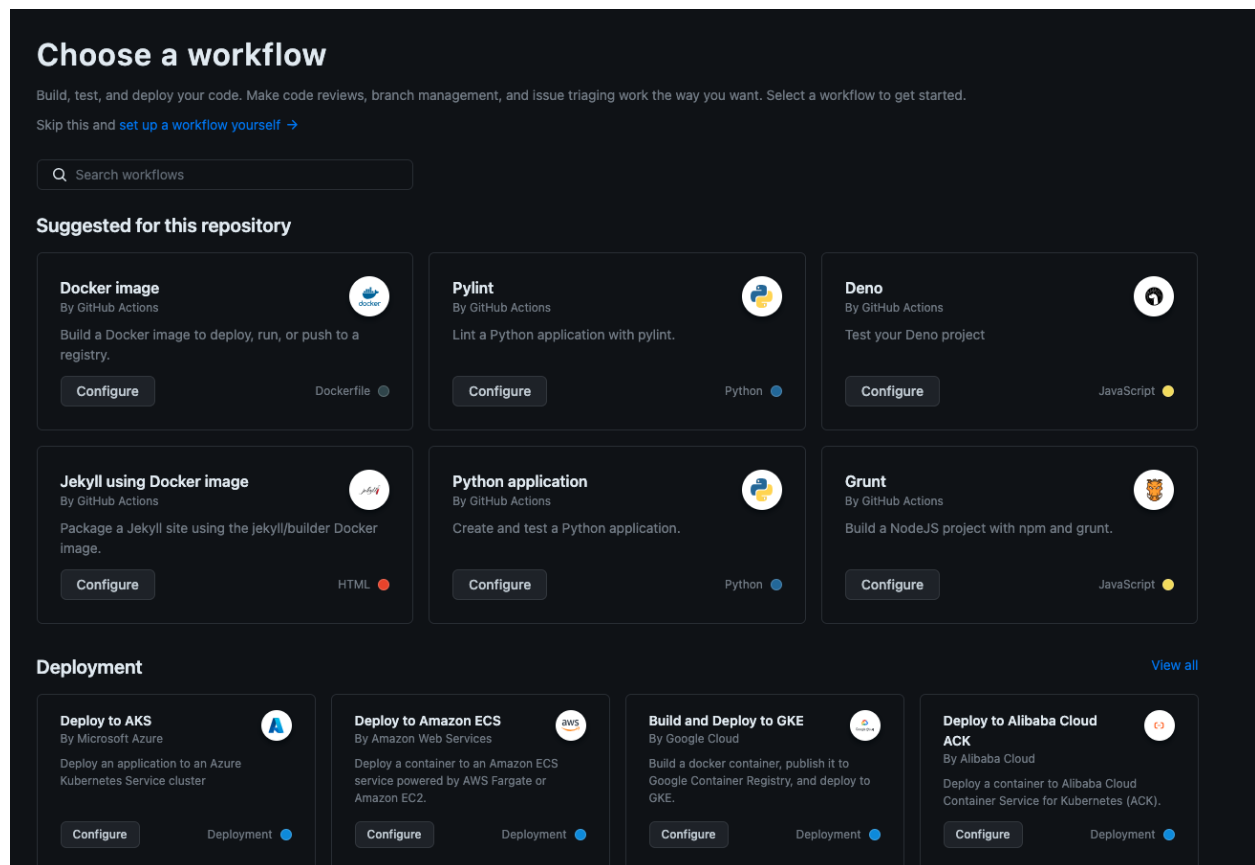
**Integration:** As our codebase resides on GitHub, GitHub Actions provides tight integration with repositories, issues, and pull requests.

**Simplicity:** It offers straightforward YAML configurations and a plethora of ready-to-use actions.

**Cost-Effectiveness:** For public repositories and small projects, GitHub Actions provides generous free quotas.

**VS-code:** With dedicated Github Actions plugins in VS code it's simpler to monitor our build from our IDE itself.

Below screenshot shows various options of predefined workflows we can use for different purposes.



## Potential Challenges with GitHub Actions

**Learning Curve:** While GitHub Actions is designed to be user-friendly, there's still a learning curve associated with setting up complex workflows.

**Resource Limits:** For private repositories, the free tier has limited minutes and storage, which may require a budget for larger projects.

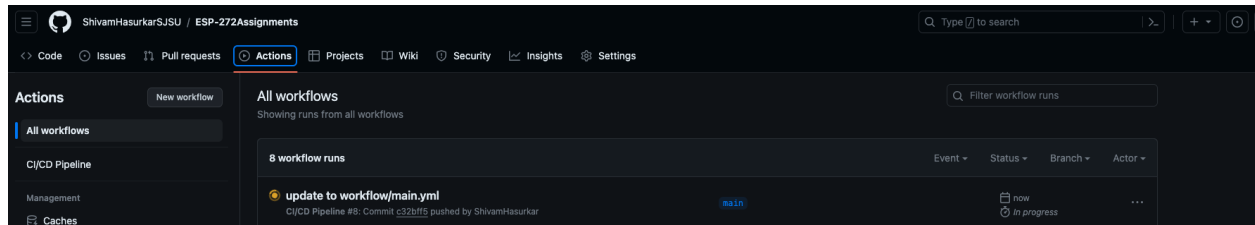
**Complex Workflows:** As workflows grow in complexity, they may become harder to manage and debug.

**Market Maturity:** GitHub Actions is newer than some competitors, which means it may not have the same breadth of community-contributed actions.

## Steps we followed to build Github Actions pipeline

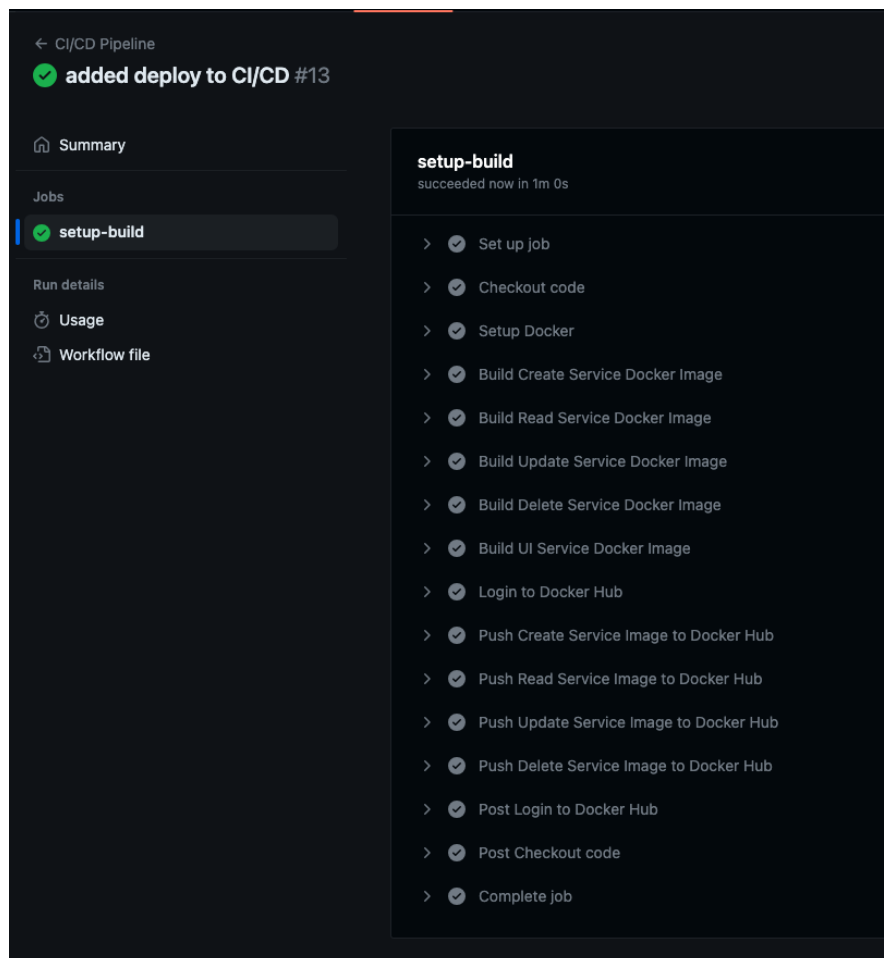
### 1. Explored a bit about Github actions for the newly created GitHub repository

We used the Assignment 2 github repository of CRUD operations using react to build the Github Actions CI/CD pipeline.



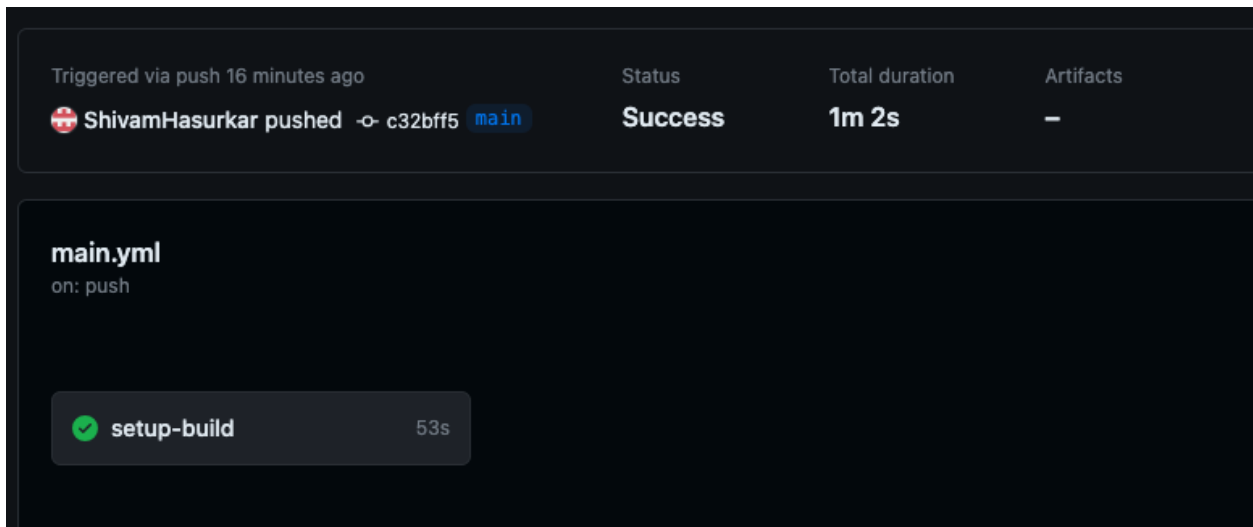
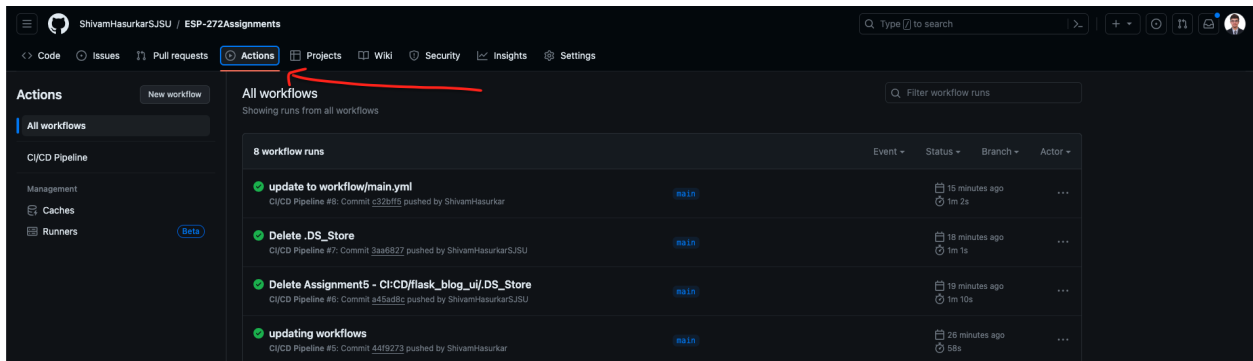
### 2. Writing the main.yml file under './.github/workflows/' with certain mandatory parameters like

- Name
- on - the point of trigger (trigger on commit to main)
- Jobs - steps to build and deploy

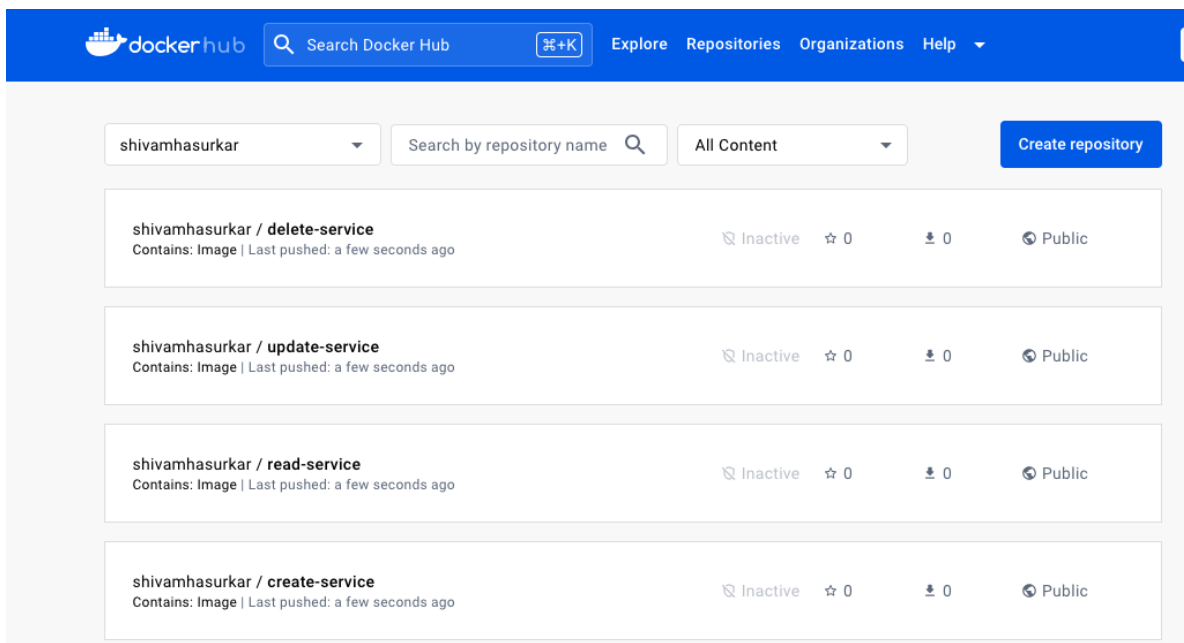


### 3. Commit to the main branch and check the GitHub Actions for running build

Whenever we push anything to the main branch, the build will get triggered automatically and will deploy too on the docker hub

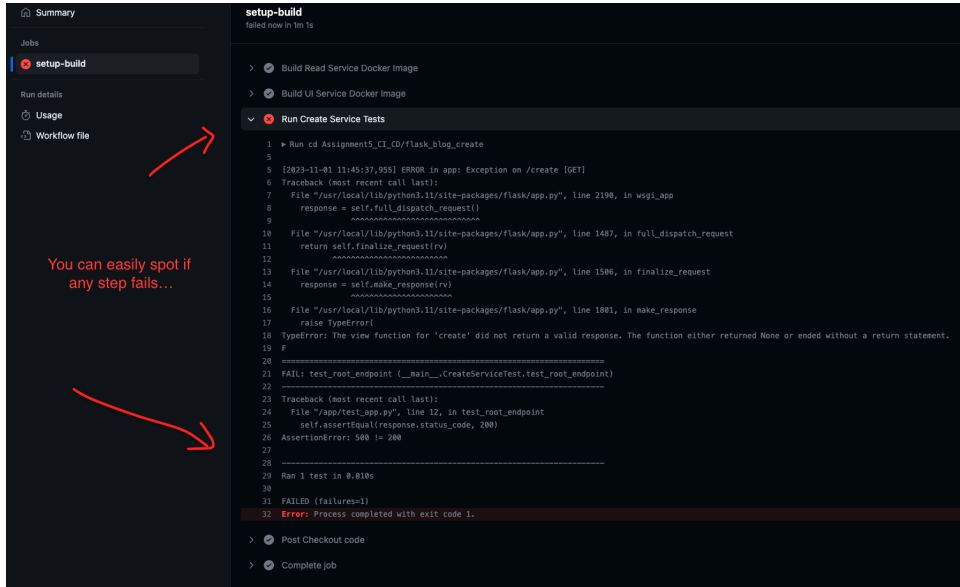


### 4. The containers are deployed to the docker hub registry.



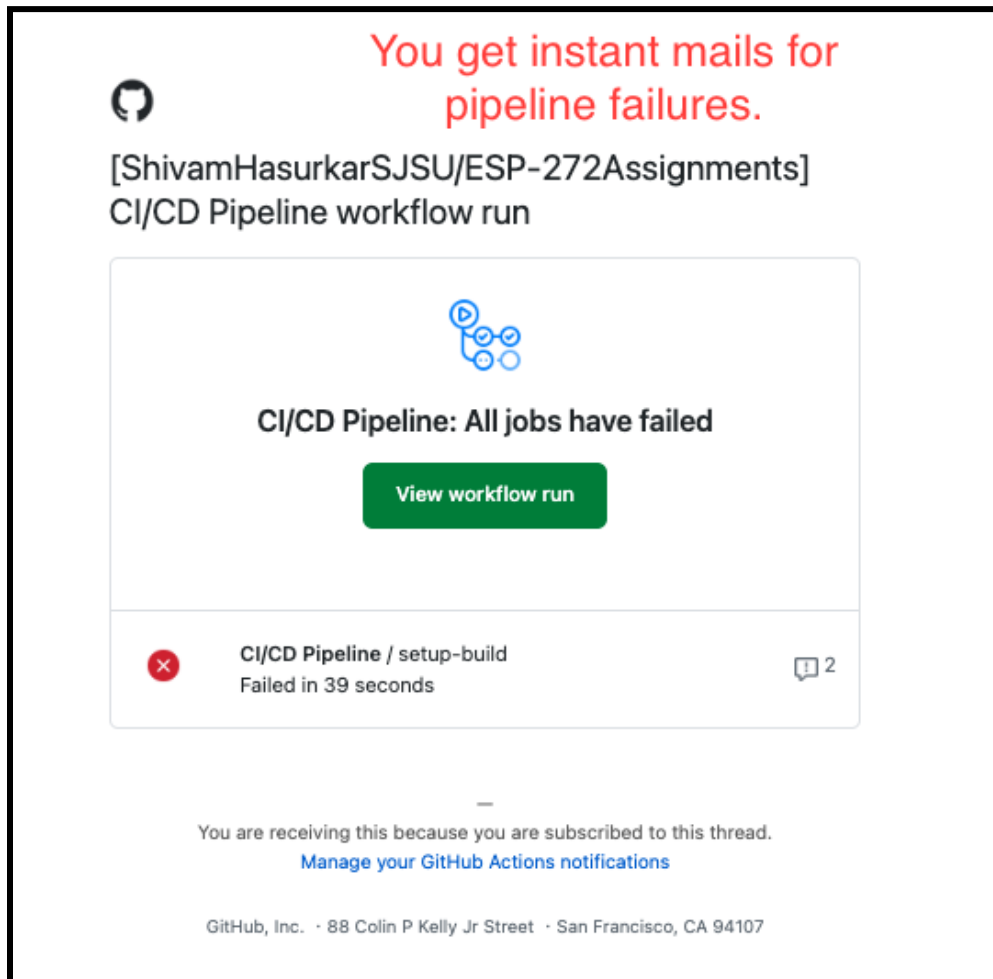
## Below screenshots are not steps but just some informational things about GitHub Actions

We can see steps that failed and see the detailed logs to find the root cause.



The screenshot shows the GitHub Actions interface for a workflow named 'setup-build'. The left sidebar contains links for Summary, Jobs, Run details, Usage, and Workflow file. The 'Jobs' section is active, showing a list of jobs: 'Build Read Service Docker Image', 'Build UI Service Docker Image', and 'Run Create Service Tests'. The 'Run Create Service Tests' job is highlighted with a red arrow and a red dot, indicating it failed. Below the job name, a red arrow points to the text 'You can easily spot if any step fails...'. The main panel displays the logs for the failed job, showing a traceback error in the 'test\_root\_endpoint' step. The error message is 'TypeError: The view function for 'create' did not return a valid response. The function either returned None or ended without a return statement.' The logs also show the test result as 'FAILED (failures=1)' and the process completed with exit code 1.

As the build fails you get an automatic update in your mailbox with brief information

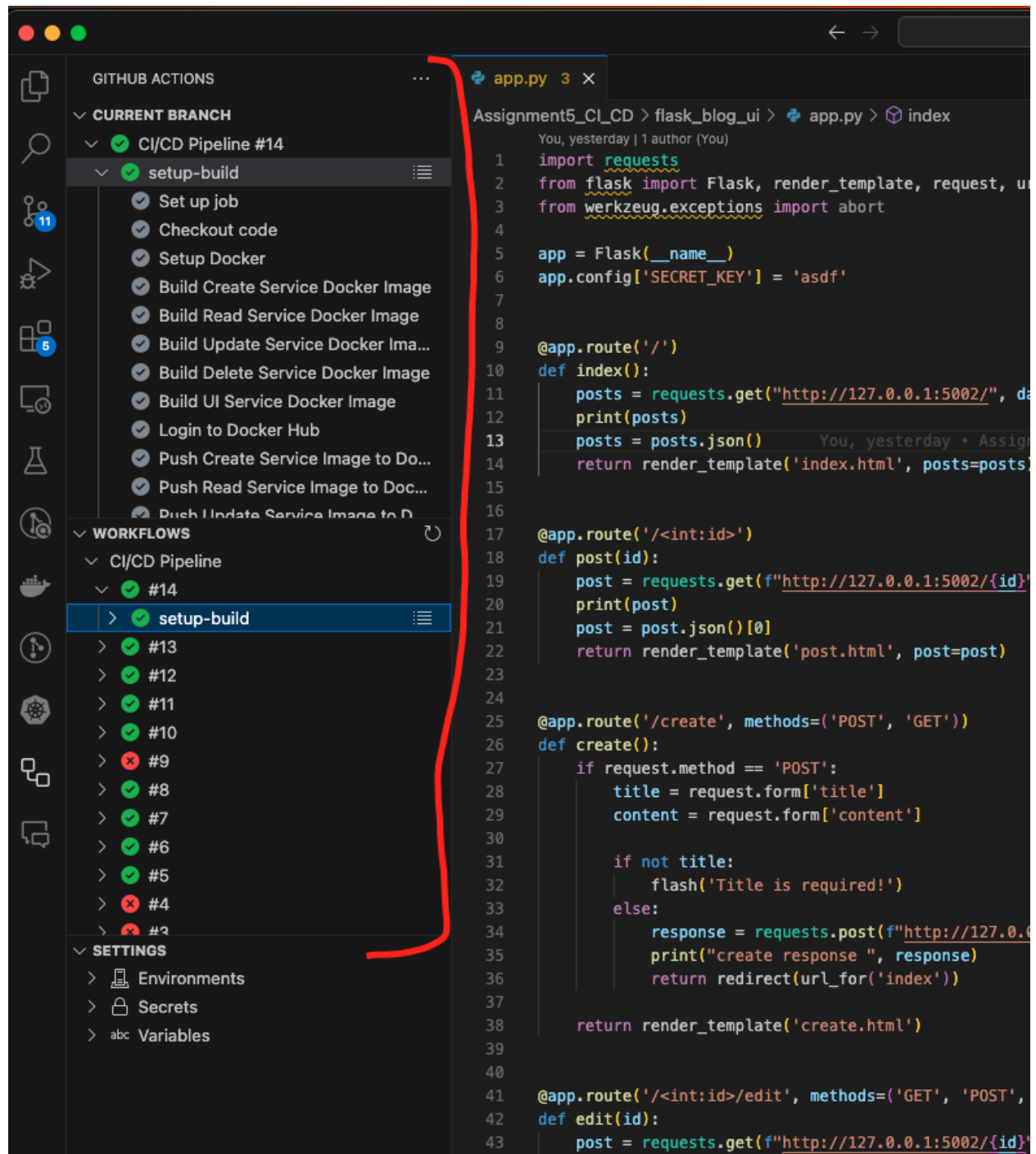


The screenshot shows a notification email from GitHub. The header features the GitHub logo and the text 'You get instant mails for pipeline failures.' in red. The main body of the email contains the following information:

- Repository: [ShivamHasurkarSJSU/ESP-272Assignments]
- Workflow: CI/CD Pipeline workflow run
- Status: CI/CD Pipeline: All jobs have failed
- Action: View workflow run
- Job Name: CI/CD Pipeline / setup-build
- Status: Failed in 39 seconds
- Comments: 2

Below the notification, there is a message: 'You are receiving this because you are subscribed to this thread.' followed by a link 'Manage your GitHub Actions notifications'. At the bottom, the GitHub logo and address are displayed: 'GitHub, Inc. · 88 Colin P Kelly Jr Street · San Francisco, CA 94107'.

You can easily monitor builds from your VS code by enabling extension



## Future scope - implementing caching for the workflow.

Caching can be implemented in GitHub Actions to speed up your workflows. Caching is particularly useful for dependencies that don't change often, like node modules or Docker layers. By caching these elements, you can reduce the time your workflow spends reinstalling them on each run.

