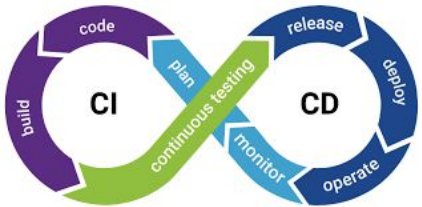
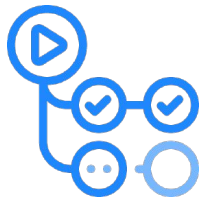
CI/CD: GitHub Actions

[Waheed Iqbal]

DevOps (Fall 2023)
Department of Data Science, FCIT, University of the Punjab
Lahore, Pakistan





CI/CD

- CI/CD stands for Continuous Integration and Continuous Deployment (or Continuous Delivery). It's a set of practices and tools that streamline the software development and delivery process.
- CI/CD emphasizes automation at every stage of the software development lifecycle, from code integration and testing to deployment and monitoring.
- Automated testing is a critical part of CI/CD, with various types of tests (unit, integration, regression, etc.) running continuously to detect bugs and ensure code quality.
- In a CD pipeline, tested and approved code changes are automatically deployed to production or a staging environment without manual intervention.

CI/CD Tools

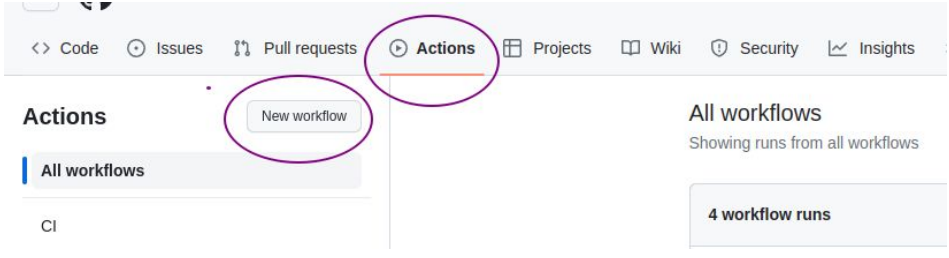
- **Jenkins:** One of the most widely used open-source CI/CD tools. It offers extensive plugin support, enabling integration with a variety of tools and platforms.
- **Travis CI:** It is a cloud-based CI/CD service that is commonly used for open-source projects hosted on GitHub.
- **CircleCI:** Cloud-native CI/CD platform that automates the software development pipeline. It offers parallelism and containerization for faster builds and testing.
- **GitLab CI/CD:** GitLab CI/CD is integrated into the GitLab platform, providing a seamless experience for source code management, CI/CD, and container registry. It is well-suited for organizations using GitLab for version control.
- **GitHub Actions:** GitHub Actions is tightly integrated with GitHub repositories, allowing developers to automate workflows and CI/CD pipelines directly within GitHub. It supports a wide range of programming languages and platforms.
- There many others too!

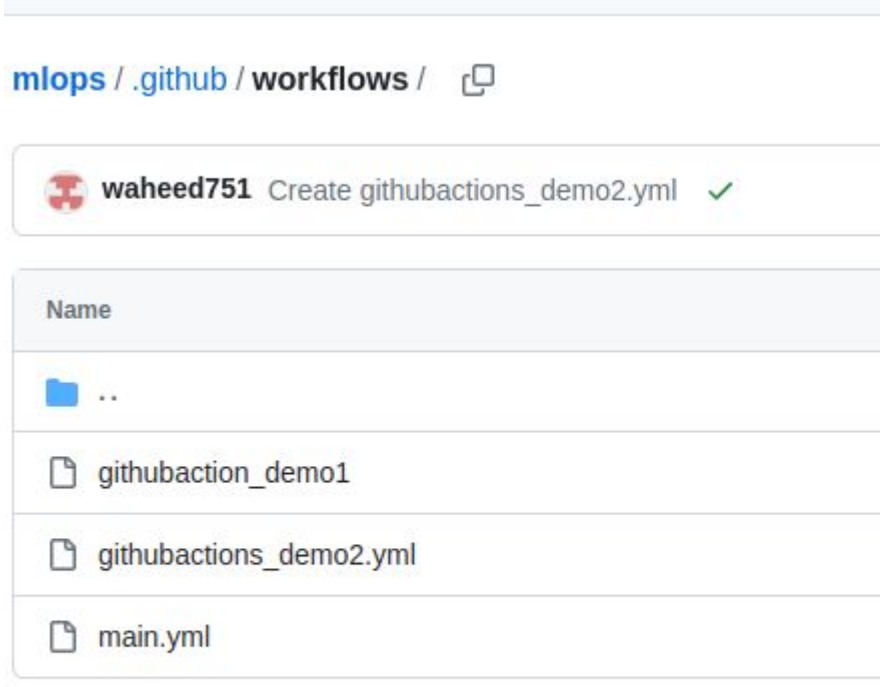
GitHub Actions

- GitHub's automation platform.
- Enables Continuous Integration (CI) and Continuous Deployment (CD) directly within your GitHub repository.
- Comprises workflows, triggered by specific events, which contain jobs made up of steps.

GitHub Actions

- GitHub Actions live in .yml or .yaml files in the .github/workflows/ directory of your repository.
-
- Example:
  - Create a .github/workflows/main.yml file in your repository.





GitHub Actions: Workflow

A workflow is a set of automated steps that define how your code should be built, tested, and deployed. You can create multiple workflows for different purposes, like CI, CD, code analysis, etc.

yaml

```yaml
name: CI Workflow

on:
  push:
    branches:
      - main

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v2

      - name: Build
        run: |
          npm install
          npm run build
```

GitHub Actions: Workflow Trigger

- Workflows can be triggered by various GitHub events.

Example:

- push: Runs the workflow on every push to the repository.
- pull_request: Runs the workflow on every pull request to the repository.

```yaml
on: [push, pull_request]
```

GitHub Actions: Defining Job

- Workflows can have one or many jobs.
- Jobs run on virtual hosts (runners) provided by GitHub.

yaml

```yaml
jobs:
  build:
    runs-on: ubuntu-latest
```

GitHub Actions: Steps in Job

- Each job contains a sequence of tasks called steps.
- Steps can be commands or actions.

yaml

```yaml
steps:
  - name: Check out code
    uses: actions/checkout@v2

  - name: Run a command
    run: echo "Hello, GitHub Actions!"
```

GitHub Actions: Environment Variables

- GitHub sets several environment variables for use in workflows.

Example:

- Print the current repository name:

```yaml
run: echo "This repository is ${{ github.repository }}"
```

GitHub Actions: Deploying

- Automate deployment tasks using workflows.

Example

- Deploying to GitHub Pages:

```yaml
- name: Deploy to GitHub Pages
  uses: peaceiris/actions-gh-pages@v3
  with:
    github_token: ${{ secrets.GITHUB_TOKEN }}
    publish_dir: ./public
```

Example Demo

```
name: GitHub Actions Demo
run-name: ${{ github.actor }} is testing out GitHub Actions 🚀
on: [push]
jobs:
 Explore-GitHub-Actions:
   runs-on: ubuntu-latest
   steps:
     - run: echo "🎉 The job was automatically triggered by a ${{ github.event_name }} event."
     - run: echo "🐧 This job is now running on a ${{ runner.os }} server hosted by GitHub!"
     - run: echo "🔎 The name of your branch is ${{ github.ref }} and your repository is ${{ github.repository }}."
     - name: Check out repository code
       uses: actions/checkout@v4
     - run: echo "💡 The ${{ github.repository }} repository has been cloned to the runner."
     - run: echo "🖥️ The workflow is now ready to test your code on the runner."
     - name: List files in the repository
       run: |
         ls ${{ github.workspace }}
     - run: echo "🍏 This job's status is ${{ job.status }}."
```