**GITHUB ACTIONS**

Topics:

1. What is github action?

2. How to use github action? Step by step demo

3. Github action terminology – Events, Workflows, jobs, steps

Demo:

1. Sign up or login in github account
2. Create a new repository
3. In a repo create a folder .github/workflows
4. In the folder create a YAML file with extension .yml
5. Add workflow content to the file
6. Commit and push the changes
7. Go to main page of repo and go to actions
8. Select the workflow from left side and check logs and results

**Terms:**

**Workflow**: Collection of jobs defined the yaml file

Keyword: name

**Event**: any activity in repo that can trigger workflow;

keyword: on push pull merge

**Jobs:** Collection of steps

Keyword : jobs

**Steps**: Actions to be taken

Keyword: steps



**GITHUB ACTIONS**

GitHub Actions is a platform for continuous integration / continuous delivery (CI/CD). It enables you to automate build, testing, and deployment pipelines. Since the user does not need to leave GitHub, it naturally enhances the workflow and productivity. Developers can perform repetitive tasks while reducing manual intervention.

**Why It's Useful:**

* **Automation:** You can automate repetitive tasks like testing, building, and deploying your code.
* **Integration:** GitHub Actions integrates seamlessly with GitHub, making it easy to set up and use directly from your repository.

In short, GitHub Actions helps you automate and streamline your development process directly within GitHub.

Similar tools:

Jenkins, Gitlab

Major benefit of Github Actions is it can be performed on repository direct instead of using any other third party tool

**Workflows in GitHub Actions** are like automated to-do lists for your projects. They define a series of tasks that run automatically based on specific events or triggers. Here's a deeper dive into how they work:

**Workflow Structure**

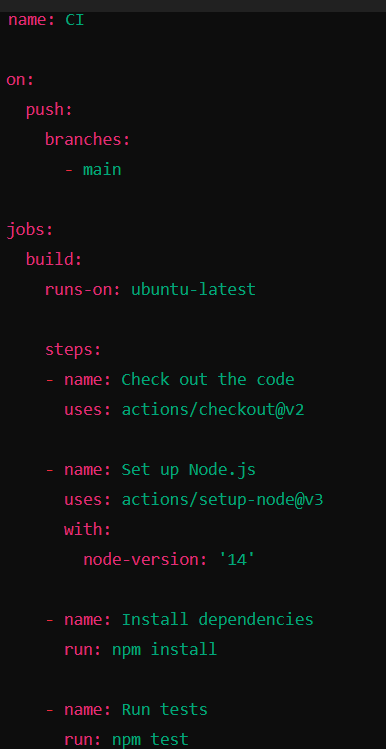
A workflow is defined in a YAML file, typically stored in a .github/workflows directory within your repository. The YAML file outlines what the workflow should do and when it should run.

**Key Components of a Workflow**

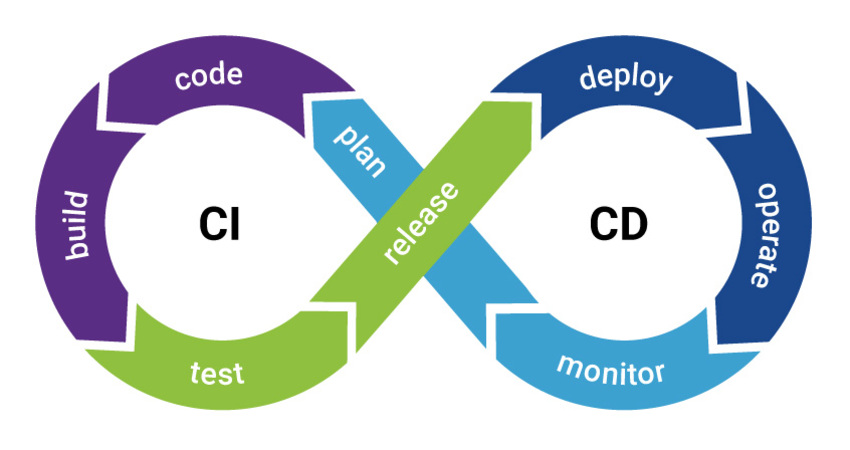
1. **Triggers (on):**
   * This defines when the workflow should start. Triggers could be:
     + **push:** Runs when code is pushed to the repository.
     + **pull\_request:** Runs when a pull request is opened or updated.
     + **schedule:** Runs on a defined schedule (like every day at a specific time).
     + **workflow\_dispatch:** Allows you to manually trigger a workflow from the GitHub interface.
2. **Jobs:**
   * A workflow can contain one or more jobs. Each job is a set of steps that run on a specific environment (like Ubuntu, Windows, or macOS).
   * Jobs run in parallel by default but can be configured to run sequentially if one job depends on another.
3. **Steps:**
   * Steps are the individual tasks that a job performs. They can include actions (pre-built commands provided by GitHub or the community) or custom scripts.
   * For example, a step might be to check out your code, set up a programming environment, run tests, or deploy an application.
4. **Actions:**
   * Actions are reusable units of code that perform specific tasks. You can use actions created by others or write your own.
   * For example, there are actions to set up Node.js, install dependencies, or even send a Slack message when a workflow completes.

**Practical Uses of Workflows**

* **Continuous Integration (CI):** Automatically test and validate your code every time you push changes.
* **Continuous Deployment (CD):** Automatically deploy your application to a server or cloud service after passing tests.
* **Notifications:** Send notifications to team members when certain events occur, like a successful deployment or a failed test.



CI-CD pipeline



Basic of terraform (infra as code)

Create a project using terraform on github-actions

Python project using github action

Jenkins

AWS –

Azure pipeline