

GitHub Actions Documentation

Continuous Integration (CI):

Continuous Integration is a practice of integrating all your code changes into the main branch of shared repository early and often , automatically testing each change when you commit them, and automatically kicking off a build. It ensures that code changes are regularly and automatically integrated.

Continuous Delivery (CD):

Continuous Delivery is the process of automatically and consistently preparing code changes for release. It involves automated testing, quality checks and other steps to ensure that the software is always in a deployable state. CD extends CI by automating the process of readying the software for release, but it doesn't necessarily deploy it to the production environment automatically.

Continuous Deployment (CD):

Continuous Deployment is the extension of Continuous Delivery , where code changes that pass all the automated testing and quality checks are automatically deployed to the production environment without manual intervention.

CI/CD Pipeline:

A CI/CD pipeline is an automated process that combines CI/CD in software development . It represents the steps and automation

from integrating code changes to delivering a deployable product. GitHub Actions enables you to define and customize your CI/CD pipeline directly in your repository using YAML-based workflows.

Git:

Git is a distributed version control system used for tracking changes in source code during software development. It allows multiple users to work on the same project simultaneously. GitHub is a hosting service for Git repositories and provides additional features such as issue tracking, pull requests, and collaboration tools.

GitHub:

GitHub is a Web based platform that provides hosting for Git repositories. It adds a collaborative layer on top of Git , offering features like pull requests , issue tracking and project management tools .

Various Git Commands:

1. Git init : Initializes a new Git repository in the current directory, creating a hidden .git folder to store version control information.

2. Git add: Stages changes in the working directory, preparing them to be included in the next commit.

3. Git commit: Records staged changes to the repository, creating a new commit with a message describing the changes.

4. Git push: Uploads local repository commits to a remote repository, updating the remote branch with local changes.

5. Git pull: Fetches changes from a remote repository and merges them into the local repository, updating the working directory with the latest changes.

6. Git clone: Copies an existing repository from a remote server to the local machine, creating a new directory containing the repository files.

GitHub Actions:

GitHub Actions is an automation and CI/CD service provided by Github. It allows us to define custom workflows to automate various tasks directly within your Github repositories. You can use it to build, test, and deploy your code, as well as automate other tasks, all from within your repository.

Workflow:

A workflow in the context of GitHub Actions is a configurable automated process that you define in your repository using a YAML file. Workflows consist of one or more jobs, each of which contains one or more steps to be executed.

In the context of software development and DevOps practices, a workflow consists of various components that together define the process of building, testing, deploying, and managing software. Here are the key components of a workflow:

Triggers:

Triggers initiate the workflow execution. They define the events that cause the workflow to start, such as code pushes, pull requests, scheduled intervals, or manual triggers.

Jobs:

Jobs are individual units of work within a workflow. They represent a set of tasks that are executed sequentially or in parallel. Each job runs on a separate virtual environment or machine.

Steps:

Steps are the individual tasks that make up a job. They can include actions, shell commands, or scripts that perform specific actions, such as checking out code, running tests, building artifacts, or deploying applications.

Actions:

Actions are reusable units of code that perform a specific task. They can be authored by GitHub, the community, or yourself. Actions can be used within workflow steps to automate common tasks, such as sending notifications, interacting with external services, or deploying applications.

Environment Variables:

Environment variables provide a way to pass data and configuration to workflow steps and actions. They can be predefined by the system or set explicitly within the workflow configuration.

Artifacts:

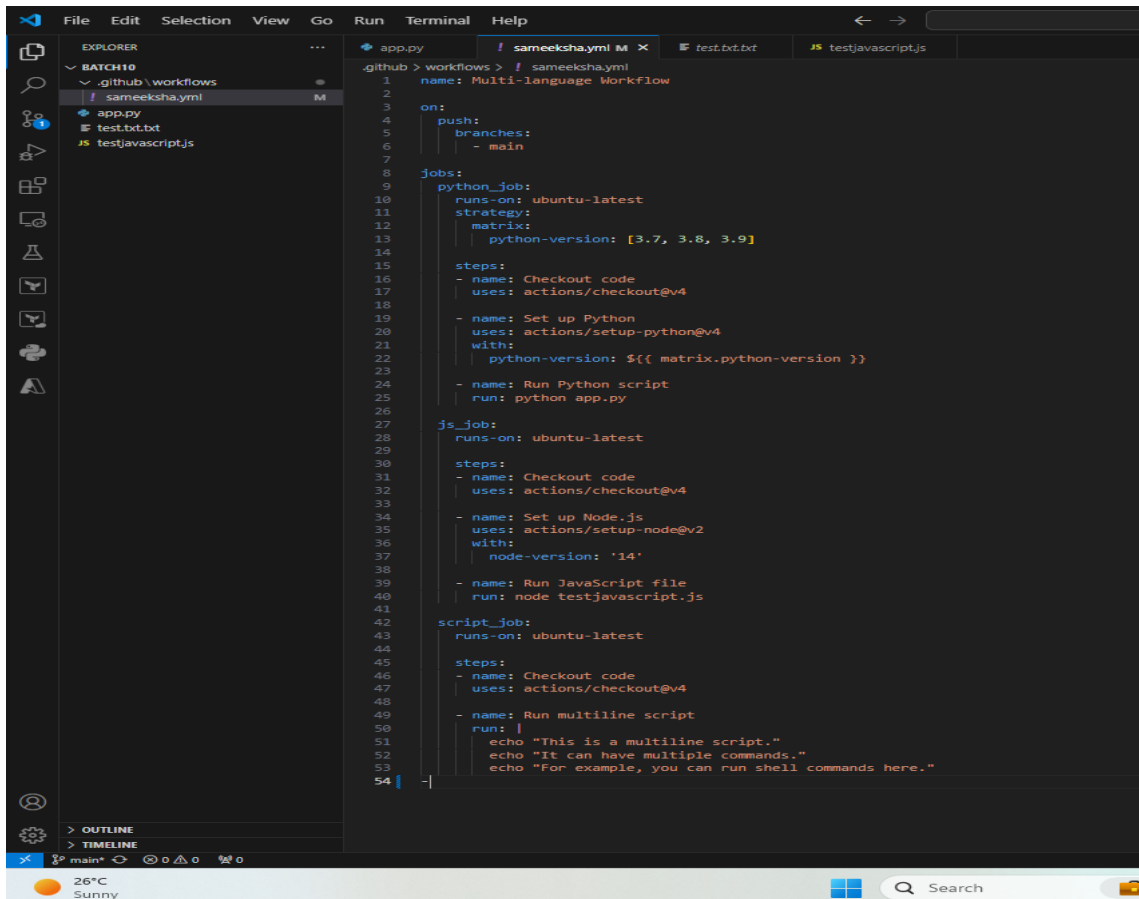
Artifacts are the outputs generated by a workflow, such as compiled binaries, test reports, or deployment packages. They can be published by jobs and used by subsequent jobs or stored for later retrieval.

Matrix:

Matrix is a software practice which we can use to define different operating systems and different versions of languages . Basically if we want to use two or more operating systems or versions in workflow then we use Matrix there.

Steps to write Workflow in GitHub :

1. Create a folder in your local machine then inside that folder create a new folder named as .github/workflows.
2. After that inside workflows create yml file .
3. Inside yml file write the customised workflow .



```
1 name: Multi-language Workflow
2
3 on:
4   push:
5     branches:
6       - main
7
8 jobs:
9   python_job:
10    runs-on: ubuntu-latest
11    strategy:
12      matrix:
13        python-version: [3.7, 3.8, 3.9]
14
15    steps:
16      - name: Checkout code
17        uses: actions/checkout@v4
18
19      - name: Set up Python
20        uses: actions/setup-python@v4
21        with:
22          python-version: ${ matrix.python-version }
23
24      - name: Run Python script
25        run: python app.py
26
27   js_job:
28    runs-on: ubuntu-latest
29
30    steps:
31      - name: Checkout code
32        uses: actions/checkout@v4
33
34      - name: Set up Node.js
35        uses: actions/setup-node@v2
36        with:
37          node-version: '14'
38
39      - name: Run JavaScript file
40        run: node testjavascript.js
41
42   script_job:
43    runs-on: ubuntu-latest
44
45    steps:
46      - name: Checkout code
47        uses: actions/checkout@v4
48
49      - name: Run multiline script
50        run: |
51          echo "This is a multiline script."
52          echo "It can have multiple commands."
53          echo "For example, you can run shell commands here."
54 |
```

4. Also create required files which you want to run in your workflow.
5. After that for pushing that workflow into the github follow the various commands.

```
MINGW64~/c:/batch10
AzureAD+SameekshaBhadoria@CEQ-ICT-DESKTOP-024 MINGW64 /c:/batch10
$ git init
Initialized empty Git repository in C:/batch10/.git/

AzureAD+SameekshaBhadoria@CEQ-ICT-DESKTOP-024 MINGW64 /c:/batch10 (master)
$ git add .

AzureAD+SameekshaBhadoria@CEQ-ICT-DESKTOP-024 MINGW64 /c:/batch10 (master)
$ git commit -m "commit changes"
[master (root-commit) abdff04] commit changes
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 test.txt.txt

AzureAD+SameekshaBhadoria@CEQ-ICT-DESKTOP-024 MINGW64 /c:/batch10 (master)
$ git branch -m main

AzureAD+SameekshaBhadoria@CEQ-ICT-DESKTOP-024 MINGW64 /c:/batch10 (main)
$ git remote add origin https://github.com/Simi28/github-actions.git

AzureAD+SameekshaBhadoria@CEQ-ICT-DESKTOP-024 MINGW64 /c:/batch10 (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 216 bytes | 216.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Simi28/github-actions.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

AzureAD+SameekshaBhadoria@CEQ-ICT-DESKTOP-024 MINGW64 /c:/batch10 (main)
$ |
```

6. After that your files and workflow will be pushed in the github.

