

CI/CD

CI/CD stands for Continuous Integration/Continuous Deployment. It is a set of practices and principles used by software development teams to automate the process of building, testing, and deploying code changes.

- **Continuous Integration (CI)** is the practice of automatically building and testing code changes whenever a developer commits code to a version control repository (e.g., Git). CI helps to catch bugs early in the development process and ensures that the codebase is always in a deployable state.
- **Continuous Deployment (CD)** is the practice of automatically deploying code changes to production or staging environments after they have been successfully built and tested. CD helps to reduce the time it takes to deliver new features and fixes to users.

Continuous Delivery (CD) is the practice of ensuring that your code is always in a deployable state, and that any code change can be deployed to production or a production-like environment at any time. It builds upon Continuous Integration (CI) by automating the release process, including testing, deployment, and configuration.

Continuous Delivery ensures that your code is always ready for deployment, while Continuous Deployment automates the deployment process all the way to production.

CI/CD pipelines automate the process of integrating, testing, and deploying code changes, helping development teams to deliver software more quickly, reliably, and efficiently.

CI/CD pipelines are typically implemented using a combination of tools and technologies, such as Jenkins, GitLab CI/CD, GitHub Actions, Travis CI, CircleCI, and others

GIT AND GITHUB

Git: Git is a distributed version control system used for tracking changes in source code during software development. It allows multiple developers to collaborate on projects, track changes, and manage code versions.

GitHub: GitHub is a web-based hosting service for Git repositories. It provides a platform for developers to collaborate on projects, host code, manage version control, and track issues and bugs.

DIFFERENCE BETWEEN GIT AND GITHUB

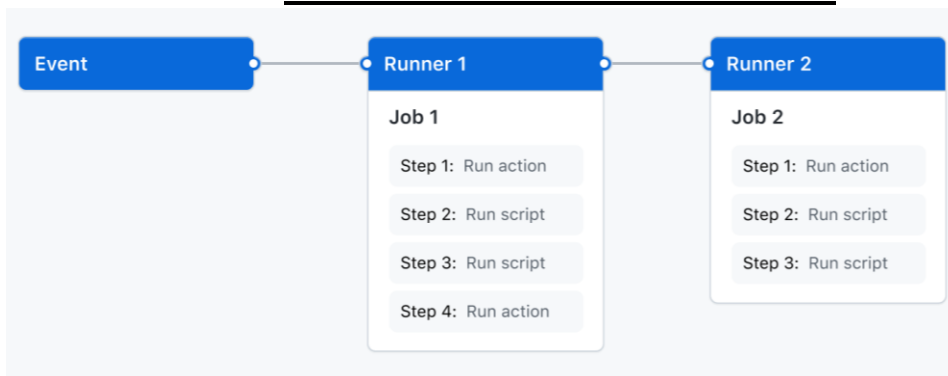
Git	GitHub
1. It is a software	1. It is a service
2. It is installed locally on the system	2. It is hosted on Web
3. It is a command line tool	3. It provides a graphical interface
4. It is a tool to manage different versions of edits, made to files in a git repository	4. It is a space to upload a copy of the Git repository
5. It provides functionalities like Version Control System Source Code Management	5. It provides functionalities of Git like VCS, Source Code Management as well as adding few of its own features

GITHUB ACTIONS

- GitHub Actions is a feature of GitHub that allows you to automate workflows for your repository.
- You can build, test, and deploy your code directly from your GitHub repository.
- It provides a flexible and customizable way to automate various tasks in your development workflow, such as running tests, building and deploying applications, and more.
- GitHub Actions uses YAML syntax to define workflows, making it easy to configure and manage automation tasks.

GitHub Actions is a powerful automation tool provided by GitHub that allows you to automate your software development workflows directly from your GitHub repository.

GITHUB ACTION WORKFLOW



BASIC TERMINOLOGY OF GITHUB WORKFLOW-

- **Workflow:** A workflow is a configurable automated process that you can set up in your GitHub repository. It consists of one or more jobs that can run sequentially or in parallel.
- **Job:** A job is a set of steps that run sequentially on the same runner. Jobs can define the environment in which they run and can have dependencies on other jobs.
- **Step:** A step is a single task that can run a command, execute a script, or use an action. Steps are the building blocks of a job.
- **Action:** An action is a reusable, standalone unit of code that performs a specific task, such as building a project, running tests, or deploying code. Actions can be created by GitHub, the community, or you, and can be used in your workflows.
- **Runner:** A runner is a machine (virtual or physical) on which jobs are executed. GitHub provides hosted runners for Linux, Windows, and macOS, or you can set up your own self-hosted runners.
- **Workflow file:** A workflow file (typically named **workflow.yml**) is a YAML file that defines your workflow, including the triggers, jobs, and steps. It is stored in the **.github/workflows** directory of your repository.
- **Event:** An event is a specific activity that triggers a workflow, such as a push to a repository, a pull request, or a scheduled event.

- **Artifact:** An artifact is a file or collection of files produced by a job that can be uploaded and downloaded by other jobs in the same workflow or stored as a build artifact.

- **Environment:** An environment is a named configuration of infrastructure where your workflows run. Environments can be used to define deployment targets and protect secrets.

- **Secret:** A secret is a sensitive piece of information, such as an API key or token, that you want to store securely and use in your workflows without exposing it in your repository.

There are several ways to create a new repository on GitHub:

1. Using the GitHub website:

- Click on the "+" icon in the top right corner of the page and select "New repository" from the dropdown menu.
- Fill in the repository name, description, choose public or private, initialize with a README file, choose a license, and click "Create repository."

Create a new repository



A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *  harshpreetkhokhar / Repository name * GithubTrial
✔ GithubTrial is available.

Great repository names are short and memorable. Need inspiration? How about [congenial-octo-carnival](#) ?

Description (optional)

- ☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.
- ☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

- ☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None ▾


Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)



Choose a license




License: None ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)


File Added from Local to Repository


 GithubTrial Public



 Pin  Unwatch 1



 main  1 Branch  0 Tags

Add file

 Code


 harshpreetkhokhar Create try.py f3d62d8 · now 3 Commits


 README.md	Update README.md	1 minute ago
 try.py	Create try.py	now

 README 

GithubTrial

Create Workflow for Repository

 GithubTrial / .github / workflows / in main

Edit Preview  Code 55% faster with GitHub Copilot

Spaces 2 No wrap

```
1 name: Python Workflow
2
3 on:
4   push:
5     branches:
6       - main
7   pull_request:
8     branches:
9       - main
10
11 jobs:
12   build:
13     runs-on: ubuntu-latest
14
15     steps:
16       - uses: actions/checkout@v2
17
18       - name: Set up Python
19         uses: actions/setup-python@v2
20         with:
21           python-version: '3.x'
```

CODE-

```
name: Python Workflow

on:
  push:
    branches:
      - main
  pull_request:
    branches:
      - main
```

```

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2

      - name: Set up Python
        uses: actions/setup-python@v2
        with:
          python-version: '3.x'

      - name: Run Python Code
        run: python try.py

```

Workflow Successful Configured

Python Workflow
main.yml

Filter workflow runs

1 workflow run

Event Status Branch Actor

✓ Create main.yml

Python Workflow #1: Commit 13fe048 pushed by harshpreetkhokhar

main

now
13s

...

build

succeeded now in 3s

Beta Give feedback

Search logs

Refresh Settings

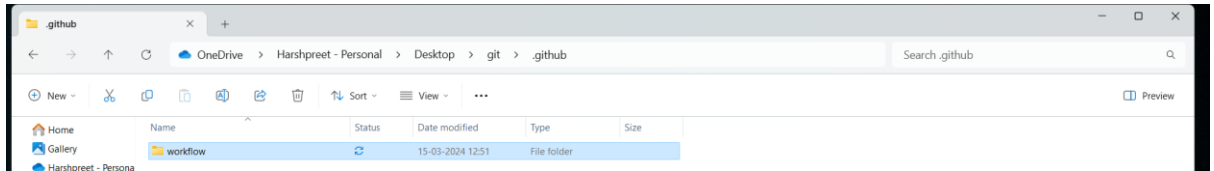
- > ✓ Set up job 1s
- > ✓ Run actions/checkout@v2 0s
- > ✓ Set up Python 0s
- ▼ ✓ Run Python Code 0s
 - 1 ▶ Run python try.py
 - 7 x = 10
 - 8 y = 5
- > ✓ Post Set up Python 0s
- > ✓ Post Run actions/checkout@v2 0s
- > ✓ Complete job 0s

2. Using Git:

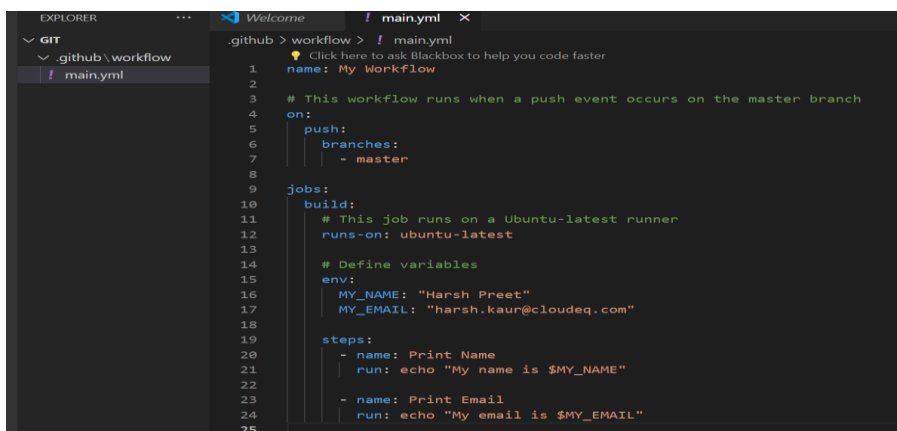
- Create a new directory on your local machine.
- Use the `'git init'` command to initialize a new Git repository in the directory.
- Use the `'git remote add origin <repository-URL>'` command to add a remote repository (GitHub repository URL) as the origin.

- Use the ‘**git push -u origin master**’ command to push the initial commit to the remote repository.

Create folder [.github] and inside that create another folder [workflow]



Open it in VS Code write Workflow



CODE-

```
name: My Workflow

# This workflow runs when a push event occurs on the master branch
on:
  push:
    branches:
      - master

jobs:
  build:
    # This job runs on a Ubuntu-latest runner
    runs-on: ubuntu-latest

    # Define variables
    env:
      MY_NAME: "Harsh Preet"
      MY_EMAIL: "harsh.kaur@cloudeq.com"

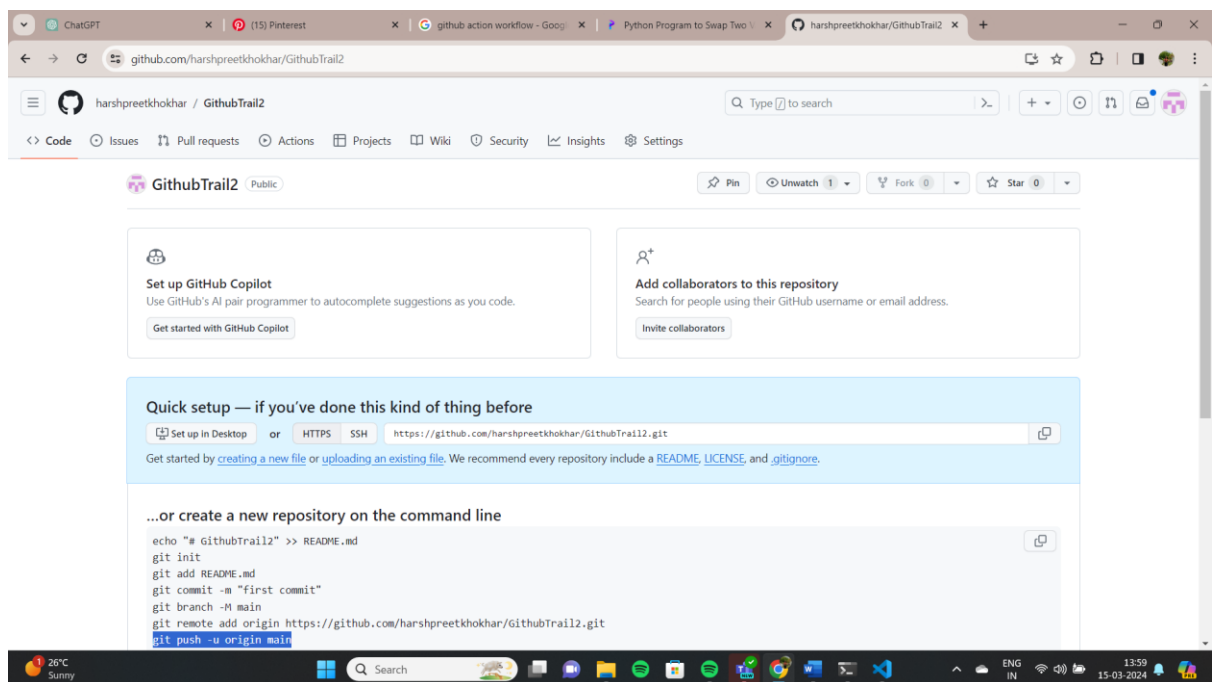
    steps:
      - name: Print Name
        run: echo "My name is $MY_NAME"

      - name: Print Email
        run: echo "My email is $MY_EMAIL"
```

```
run: echo "My name is $MY_NAME"

- name: Print Email
  run: echo "My email is $MY_EMAIL"
```

Create Repository on Github and copy URL



Add Git Commands to push the content to Remote Repository

The screenshot shows a Visual Studio Code editor with a terminal window open. The terminal displays the following commands and output:

```
PS C:\Users\harsh\OneDrive\Desktop\git> git init
Initialized empty Git repository in C:/Users/harsh/OneDrive/Desktop/git/.git/
PS C:\Users\harsh\OneDrive\Desktop\git> git add .
PS C:\Users\harsh\OneDrive\Desktop\git> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   .github/workflow/main.yml

PS C:\Users\harsh\OneDrive\Desktop\git> git commit -m "first commit"
[master (root-commit) 622c804] first commit
 1 file changed, 24 insertions(+)
 create mode 100644 .github/workflow/main.yml
origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (5/5), 553 bytes | 276.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0To https://github.com/harshpreetkhokhar/GithubTrail2.git
 * [new branch]    main -> main
branch 'main' set up to track 'origin/main'.
PS C:\Users\harsh\OneDrive\Desktop\git>
```

The Explorer view on the left shows the file structure with a folder named `.github/workflow` containing a file named `main.yml`. The file content is:

```
4 on:
5   push:
6     branches:
```

Workflow Configured as soon as we pushed a file in the repository

All workflows Filter workflow runs

Showing runs from all workflows

1 workflow run	Event	Status	Branch	Actor
<div><div>Update main.yml</div><div>My Workflow #1: Commit <code>b82c82d</code> pushed by harshpreetkhokhar</div></div>			main	
		now		...
		12s		

build succeeded 2 minutes ago in 0s Beta Give feedback Search logs

- Set up job 0s
- Print Name 1s
 - Run echo "My name is \$MY_NAME"
 - My name is Harsh Preet
- Print Email 0s
 - Run echo "My email is \$MY_EMAIL"
 - My email is harsh.kaun@cloudeq.com
- Complete job 0s

Various Git Commands –

- **`git init`**: Initializes a new Git repository in the current directory, creating a hidden `.git` directory that stores the repository's information.
- **`git clone <repository-url>`**: Clones a remote repository to your local machine, creating a new directory with the repository's files and initializing a Git repository inside it.
- **`git add <file>`**: Adds a file to the staging area, marking it to be included in the next commit.
- **`git commit -m "Commit message"`**: Commits the staged changes to the repository, creating a new commit with the specified commit message.
- **`git push <remote-name> <branch-name>`**: Pushes the committed changes to a remote repository, updating the specified branch with your local changes.
- **`git pull <remote-name> <branch-name>`**: Pulls the latest changes from a remote repository to your local repository, merging them into your current branch.
- **`git checkout -b <branch-name>`**: Creates a new branch with the specified name and switches to it.
- **`git checkout <branch-name>`**: Switches to the specified branch.
- **`git merge <branch-name>`**: Merges the specified branch into the current branch, combining the changes from both branches.
- **`git log`**: Displays the commit history of the current branch, showing the commit hashes, authors, dates, and commit messages.
- **`git status`**: Shows the status of files in the repository, indicating which files are modified, staged, or untracked.
- **`git tag <tag-name>`**: Creates a new tag for the current commit, allowing you to mark specific points in your repository's history.

- **`git checkout -- <file>`**: Discards changes in a specific file and reverts it to the last committed state.
- **`git diff <commit1> <commit2>`**: Shows the differences between two commits, allowing you to compare changes between different points in your repository's history.
- **`git branch`**: Lists all local branches in the repository, indicating which branch you are currently on and highlighting the branch that is currently checked out.