



Git-Hub Action

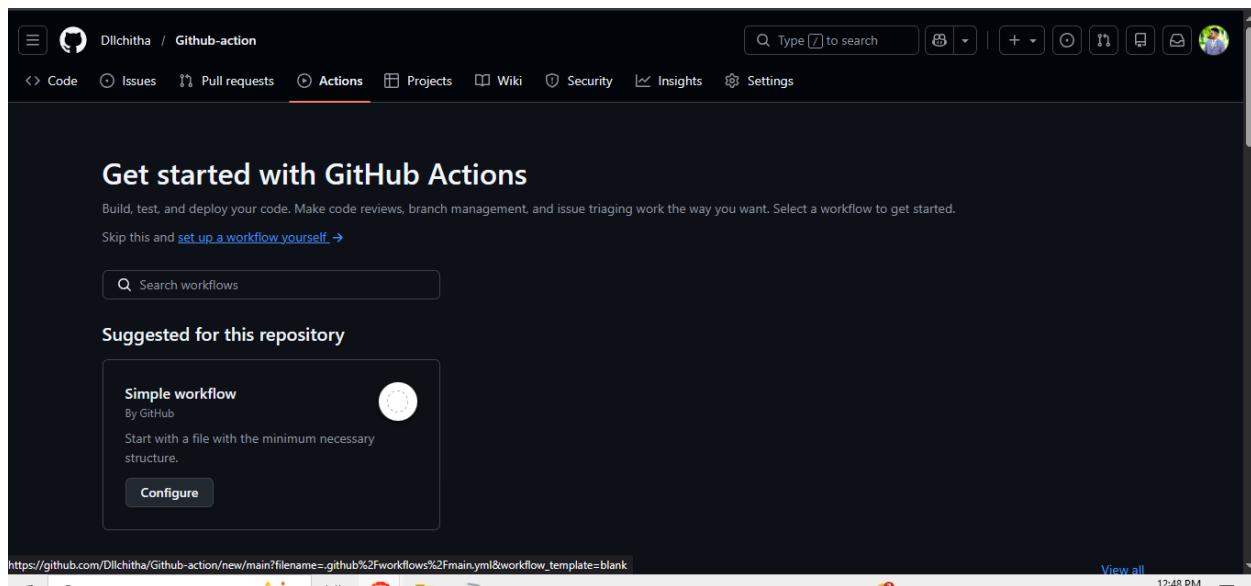
GitHub Actions is an automation platform built directly into GitHub that allows developers to automate workflows for building, testing, and deploying code. It enables you to create workflows—defined as YAML files in the `.github/workflows` directory—that are triggered by specific events like pushing code, opening a pull request, or creating an issue

Create Simple Git- Hub Action

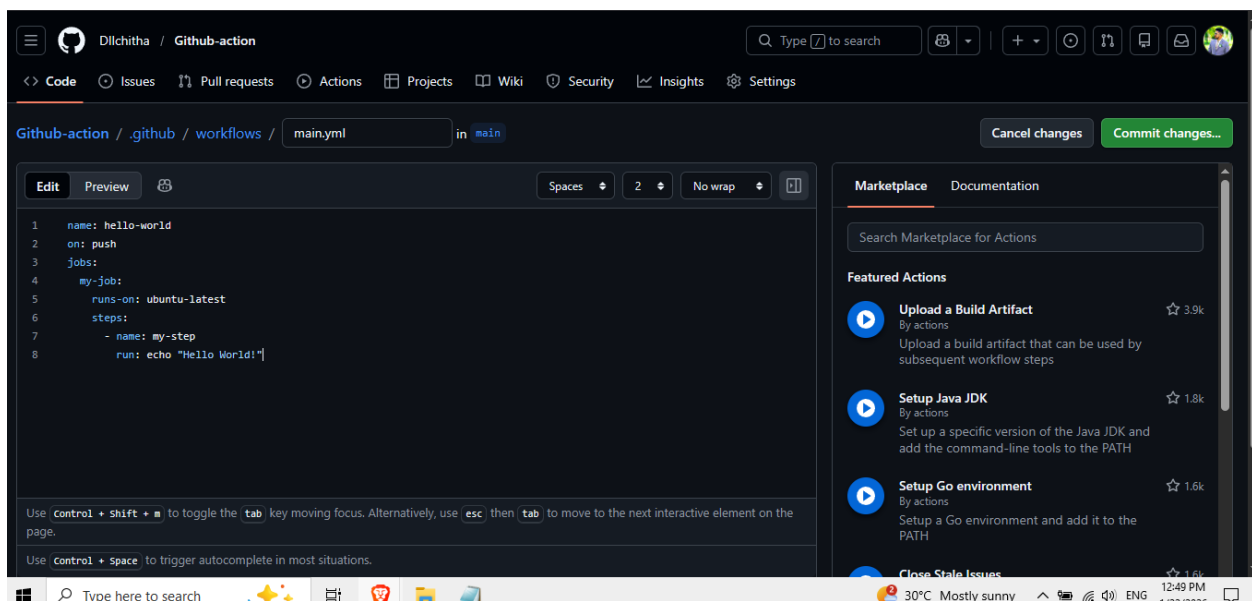
- ❖ **Step 1 Create a Work flow:** After signing up and logging into GitHub, create a new repository. Then, either manually create a `.github/workflows` folder and add a workflow file, or use the 'Actions' tab to create one through the interface,
- ❖ **Step 2 Create a Yaml base Git-hub action :** Click 'Set up a workflow yourself,' then create a GitHub Action using a YAML-based file. When you save it, make sure to use the `.yaml` extension.
- Create a New repo

The screenshot shows the 'Create a new repository' page on GitHub. It has a dark theme. The page is divided into two main sections: 'General' and 'Configuration'. In the 'General' section, the 'Owner' is 'Dilchitha' and the 'Repository name' is 'Github-action'. A green checkmark indicates 'Github-action is available'. Below this, there is a 'Description' field with a character count '0 / 350 characters'. In the 'Configuration' section, 'Choose visibility' is set to 'Public'. There are also options for 'Add README' (currently 'Off') and 'Add .gitignore' (currently 'No .gitignore').

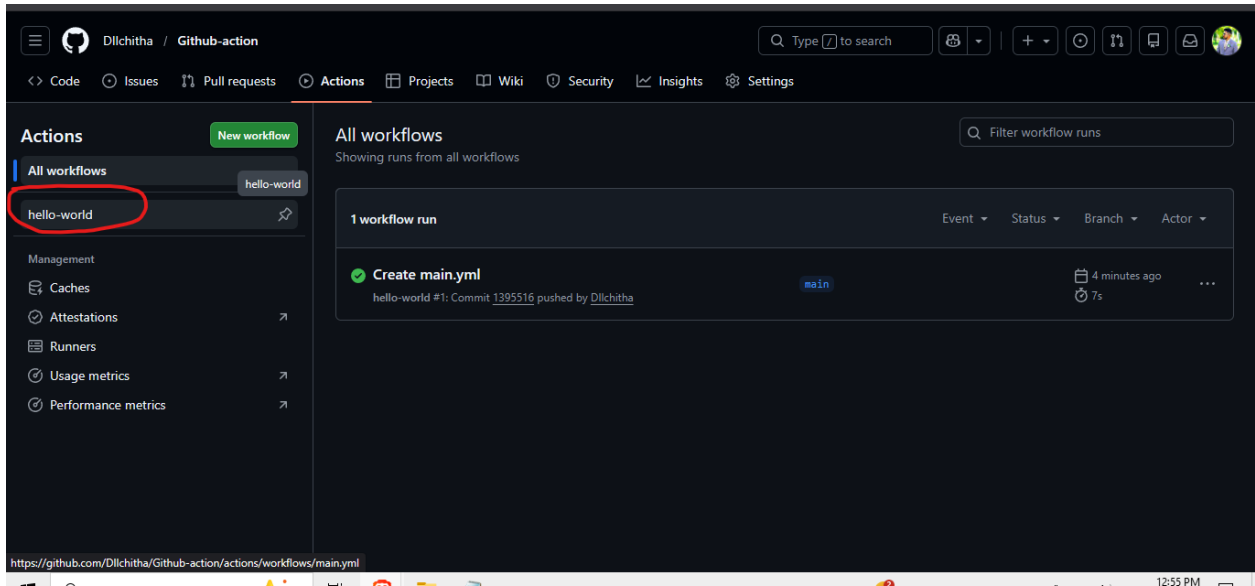
- Navigate to Actions and Create new Custom workflow for clicking set up a workflow yourself



- **Step 4 Create a Yaml file :** Click the 'Add file' dropdown button in the top right, select 'Create new file,' enter the file path `.github/workflows/ci.yml` (ensuring the `.github/workflows` folder name is exact), give the file any name with a `.yml` or `.yaml` extension—common examples are `ci.yml`, `deploy.yml`, or `build.yml`—paste the corrected YAML code into the editor, and commit the file with a message such as 'Add CI/CD workflow'.
- Create a YAML file with the following content.



- After the commit changes go to action the view how Git hub action out put.



Now, try practicing these GitHub Actions steps on your own,

Test 1

name: hello-world

on: push

jobs:

my-job:

runs-on: ubuntu-latest

steps:

- name: my-step

run: echo "Hello World!"

Test 2

name: hello-world

on: push

jobs:

my-job:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v4

- name: my-step

run: echo "Hello World!"

- name: Build

```
run: |
  echo "Starting build process..."
  # Add your build commands here
  # Example: npm run build, mvn compile, dotnet build, etc.
  echo "Build completed successfully!"
```

- name: Test

```
run: |
  echo "Running tests..."
  # Add your test commands here
  # Example: npm test, pytest, go test, etc.
  echo "Tests passed!"
```

- name: Deploy

```
run: |
  echo "Starting deployment..."
  # Add your deployment commands here
  # Example: deploy to server, cloud, container registry, etc.
  echo "Deployment completed!"
```

Test 3 : Mutiple Action

- Create a New Repo and push next.js and package.json file
Next.js
module.exports = {
 output: 'export', // Required for static export to GitHub Pages
 // other configurations...
}
Package.json
{
 "scripts": {
 "build": "next build",
 "test": "your-test-command"
 }
}
- Create a git hub workflow and add below yaml format file,
CI.yaml

```
name: build-test-deploy
on: push
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: checkout repo
```

```
  uses: actions/checkout@v3

- name: Change to project directory
  run: cd frontend || cd app || cd . # Go to your project folder

- name: use node.js
  uses: actions/setup-node@v3
  with:
    node-version: '18.x'

- name: Install dependencies
  working-directory: ./frontend # Add this to all steps
  run: npm install

- name: Build project
  working-directory: ./frontend # Add this to all steps
  run: npm run build

test:
  needs: build
  runs-on: ubuntu-latest
  steps:
    - name: checkout repo
      uses: actions/checkout@v3

    - name: use node.js
      uses: actions/setup-node@v3
      with:
        node-version: '18.x'

    - name: Install dependencies
      working-directory: ./frontend # Add this
      run: npm install

    - name: Run tests
      working-directory: ./frontend # Add this
      run: npm test

deploy:
  needs: test
  permissions:
    contents: write
    pages: write
    id-token: write
  environment:
```

```
name: production
url: ${ steps.deployment.outputs.page_url }
runs-on: ubuntu-latest
steps:
  - name: checkout repo
    uses: actions/checkout@v3
    with:
      token: ${ secrets.GITHUB_TOKEN }

  - name: use node.js
    uses: actions/setup-node@v3
    with:
      node-version: '18.x'

  - name: configure github pages
    uses: actions/configure-pages@v3
    with:
      static_site_generator: next

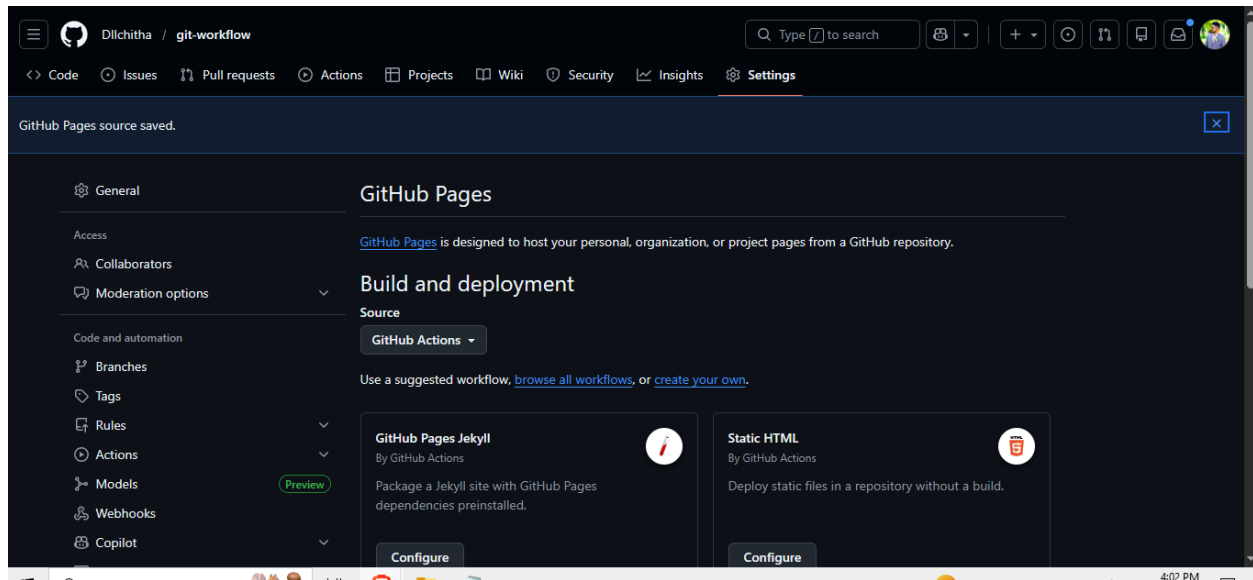
  - name: Install dependencies
    working-directory: ./frontend # Add this
    run: npm install

  - name: Build project
    working-directory: ./frontend # Add this
    run: npm run build

  - name: upload artifacts
    uses: actions/upload-pages-artifact@v1
    with:
      path: "./frontend/out" # Update path if needed

  - name: deploy
    id: deployment
    uses: actions/deploy-pages@v1
```

- To enable GitHub Pages, which is required for deployment, go to your repository's Settings via the top tabs, scroll down to 'Pages' in the left sidebar, and under the 'Build and deployment' section, set the Source to 'GitHub Actions' the workflow will then be auto-detected.



Out put,

