

CI/CD with GitHub Actions

1. Create a new feature branch for the CI setup:

```
git checkout -b feature/ci-workflow-setup
```

2. Create the necessary directories: In the root of your ecommerce-product-page-team project folder, create the .github directory, and then a workflows directory inside it.

```
mkdir -p .github/workflows
```

The -p flag ensures that parent directories are created if they don't already exist.

3. Create the CI Workflow File: ci.yaml. This file tells GitHub Actions when to run, what steps to perform, and on what environment.

```
# .github/workflows/ci.yaml
# This workflow defines the Continuous Integration (CI) process for your
project.

name: CI Workflow # The name of your workflow, displayed in the GitHub
Actions tab.

# Controls when the workflow will run.
on:
  # Triggers the workflow on 'push' events to the 'main' branch.
  # This means every time code is pushed directly to 'main' (or a PR is
merged into 'main').
  push:
    branches: [ main ] # Indented 2 spaces from 'push'

  # Triggers the workflow on 'pull_request' events targeting the 'main'
branch.
  # This is crucial for pre-merge checks, ensuring code quality before
integration.
  pull_request:
    branches: [ main ] # Indented 2 spaces from 'pull_request'

  # Allows you to run this workflow manually from the GitHub Actions tab.
  # Useful for testing or re-running failed workflows.
  workflow_dispatch: # Indented 2 spaces from 'on'

# A workflow run is made up of one or more jobs that can run sequentially
or in parallel.
jobs: # Top-level element, starts at column 0
  # Defines a single job named "build-and-lint".
  build-and-lint: # Indented 2 spaces from 'jobs'
    # The type of runner (virtual environment) that the job will execute
on.
```

```
# 'ubuntu-latest' provides a fresh Linux environment with necessary
tools.
runs-on: ubuntu-latest # Indented 2 spaces from 'build-and-lint'

# Steps represent a sequence of tasks that will be executed as part
of this job.
steps: # Indented 2 spaces from 'build-and-lint'
  # Step 1: Checkout repository
  # Uses the 'actions/checkout' action to clone your repository into
the runner's workspace.
  # This allows your job to access your code.
  - name: Checkout repository # Indented 2 spaces from 'steps'
(hyphen for list item, then 2 spaces for 'name')
    uses: actions/checkout@v4 # Indented 2 spaces from 'name'

  # Step 2: Set up Node.js environment
  # Uses the 'actions/setup-node' action to configure the Node.js
environment.
  # We specify Node.js version 18, which is a common and compatible
version for Vite React apps.
  # 'cache: npm' caches npm dependencies, significantly speeding up
subsequent runs.
  - name: Set up Node.js # Indented 2 spaces from 'steps'
    uses: actions/setup-node@v4 # Indented 2 spaces from 'name'
    with: # Indented 2 spaces from 'uses'
      node-version: '18' # Indented 2 spaces from 'with'
      cache: 'npm' # Indented 2 spaces from 'with'

  # Step 3: Install project dependencies
  # Executes the 'npm install' command to download and install all
project dependencies.
  - name: Install dependencies # Indented 2 spaces from 'steps'
    run: npm install # Indented 2 spaces from 'name'

  # Step 4: Run ESLint
  # Executes the 'npm run lint' script defined in your package.json.
  # This step checks for code style issues and potential errors,
ensuring code quality.
  - name: Run ESLint # Indented 2 spaces from 'steps'
    run: npm run lint # Indented 2 spaces from 'name'

  # Step 5: Build project
  # Executes the 'npm run build' script defined in your package.json.
  # This step compiles your React project for production, verifying
that it can be successfully built
  # and catches any build-time errors.
  - name: Build project # Indented 2 spaces from 'steps'
    run: npm run build # Indented 2 spaces from 'name'
```

4. Commit and Push the Workflow File to GitHub
5. Create a Pull Request to trigger the CI workflow

Implement basic CI workflow with GitHub Actions #3

dewi-xaltius wants to merge 1 commit into `main` from `feature/ci-workflow-setup`

Conversation 0 Commits 1 Checks 0 Files changed 1

dewi-xaltius commented 3 minutes ago Owner ...

Test1

Implement basic CI workflow with GitHub Actions ✓ a208183

All checks have passed
1 successful check ^

✓ CI Workflow / build-and-lint (pull_request) Successful in 14s ...

No conflicts with base branch
Merging can be performed automatically.

Merge pull request You can also merge this with the command line. [View command line instructions.](#)

6. Navigate to Actions to see the details

← CI Workflow

✓ **Implement basic CI workflow with GitHub Actions #1**

Summary

Jobs

build-and-lint

Run details

Usage

Workflow file

build-and-lint
succeeded 4 minutes ago in 14s

- > Set up job
- > Checkout repository
- > Set up Node.js
- > Install dependencies
- > Run ESLint
- > Build project
- > Post Set up Node.js
- > Post Checkout repository
- > Complete job

7. Merge the PR to the main branch
8. If you have linting error, the CI workflow will fail:

Lint error to test CI workflow #5

Open dewi-xaltius wants to merge 1 commit into `main` from `feature/test-lint-failure`

Conversation 0 Commits 1 Checks 0 Files changed 1

dewi-xaltius commented now Owner ...

test

Lint error to test CI workflow fb9e25b

Some checks were not successful
1 failing check

CI Workflow / build-and-lint (pull_request) Failing after 9s ...

No conflicts with base branch
Merging can be performed automatically.

Merge pull request ▼ You can also merge this with the command line. [View command line instructions.](#)

[Back to pull request #5](#)

Lint error to test CI workflow #5 Re-run jobs ...

Summary

Jobs

build-and-lint

Run details

Usage

Workflow file

Annotations
2 errors

build-and-lint
failed 1 minute ago in 9s

Search logs Explain error

> Set up job 1s

> Checkout repository 3s

> Set up Node.js 2s

> Install dependencies 2s

> Run ESLint 1s

```
1 Run npm run lint
4
5 > commerce-product-page@0.0.0 lint
6 > eslint .
7
8
9 /home/runner/work/commerce-product-page-team/commerce-product-page-team/.eslintrc.js
10 Errors: 6/9 error "unusedVariable" is assigned a value but never used. Allowed unused vars must match /^[a-z_]/u. no-unused-vars
11
12 X 1 problem (1 error, 0 warnings)
13
14 Errors: Process completed with exit code 1.
```

Build project 0s

Post Set up Node.js 0s

Post Checkout repository 0s

Complete job 0s

9. Create a new feature branch for the CD workflow:

```
git checkout -b feature/setup-cd
```

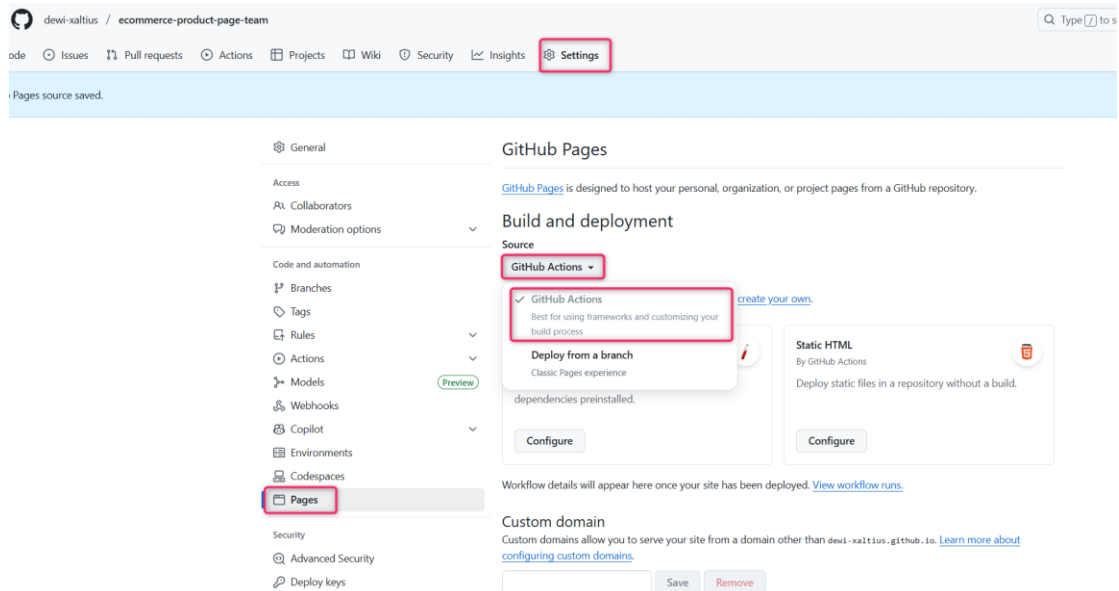
10. Configure Vite for GitHub Pages: When you deploy to GitHub Pages, your site will be live at a URL like `https://your-username.github.io/your-repo-name/`. You need to tell Vite that the base path is `/your-repo-name/`, not just `/`.

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'

// https://vite.dev/config/
export default defineConfig({
  plugins: [react()],
```

```
base: '/ecommerce-product-page-team/',  
})
```

11. Configure Your Repository for GitHub Pages: you need to tell your GitHub repository to accept deployments from GitHub Actions.



12. Create the Deployment Workflow File: We will create a new workflow file `cd.yaml` that is only responsible for deployment. It will run automatically after your CI checks pass and you merge your code into the main branch.

```
# .github/workflows/cd.yaml  
# This workflow defines the Continuous Deployment (CD) process.  
  
name: CD Workflow  
  
# 1. Trigger: This workflow runs only on pushes to the 'main' branch.  
# This means it will execute after a PR is merged into 'main'.  
on:  
  push:  
    branches:  
      - main  
  
# 2. Permissions: Grant the necessary permissions for the workflow to  
# deploy to GitHub Pages.  
permissions:  
  contents: read  
  pages: write  
  id-token: write  
  
# 3. Job: Define the deployment job.
```

```
jobs:
  deploy:
    runs-on: ubuntu-latest
    environment:
      name: github-pages
      url: ${ steps.deployment.outputs.page_url } # The URL of the
    deployed page.

    steps:
      # Step 1: Checkout your repository code.
      - name: Checkout repository
        uses: actions/checkout@v4

      # Step 2: Set up Node.js environment.
      - name: Set up Node.js
        uses: actions/setup-node@v4
        with:
          node-version: '18'
          cache: 'npm'

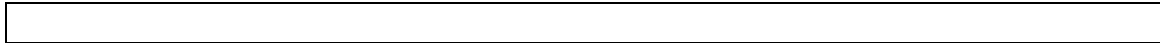
      # Step 3: Install project dependencies.
      - name: Install dependencies
        run: npm install

      # Step 4: Build the project for production.
      # The 'npm run build' command creates the 'dist' folder with your
      static site.
      - name: Build project
        run: npm run build

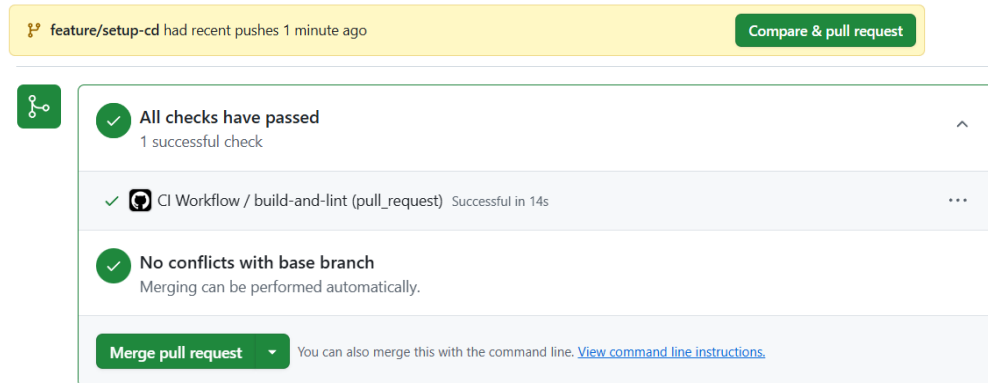
      # Step 5: Configure GitHub Pages.
      - name: Setup Pages
        uses: actions/configure-pages@v5

      # Step 6: Upload the build artifact.
      # This action takes the contents of your 'dist' folder and prepares
      it for deployment.
      - name: Upload artifact
        uses: actions/upload-pages-artifact@v3
        with:
          path: './dist'

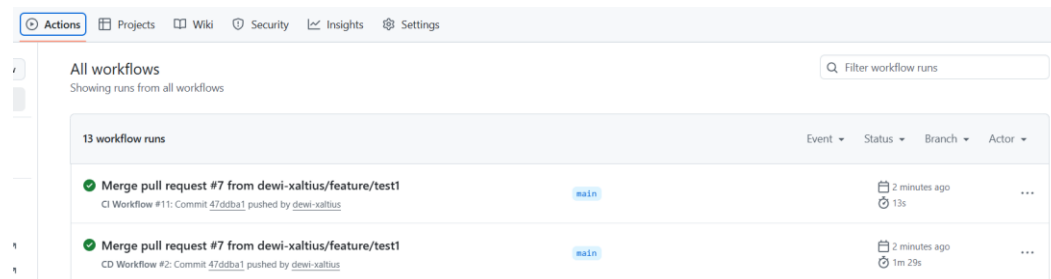
      # Step 7: Deploy to GitHub Pages.
      # This action pushes your built code to the 'gh-pages' branch and
      makes it live.
      - name: Deploy to GitHub Pages
        id: deployment
        uses: actions/deploy-pages@v4
```



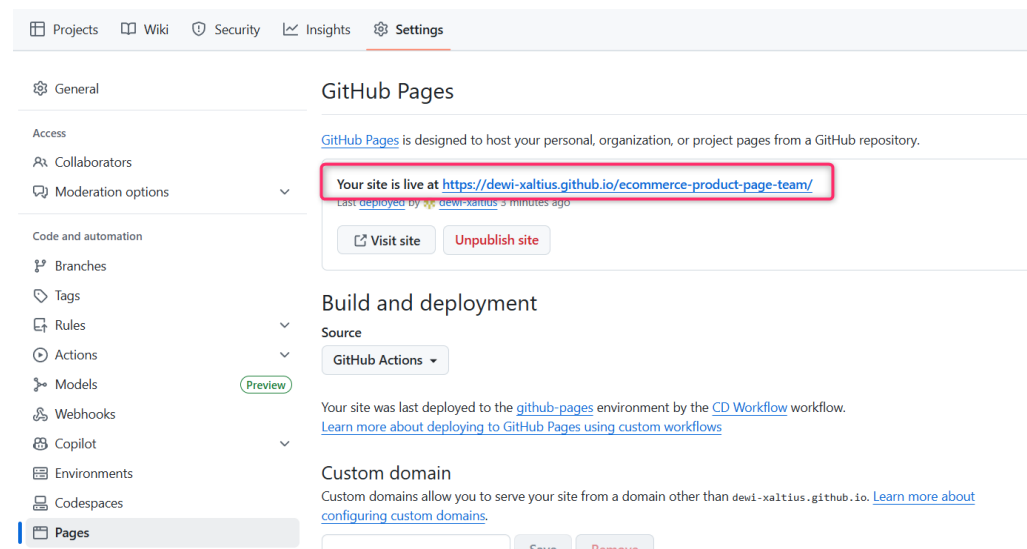
13. Create a PR:



14. Once you have merged the PR, you can check the Actions page:



15. You can check the published GitHub page:



16. After you finished merging, you can safely delete the feature branch in your repo. Then in your local, go to your main branch and pull the latest changes.

```
git checkout main
git pull origin main
```

