CI/CD Starter: GitHub Actions & GHCR

What you get A clean, drop-in pipeline that mirrors standard stages (lint → build/push → pull/test), publishes multi-arch images to GitHub Container Registry (GHCR), and uses safe defaults (concurrency, least-privileged permissions, pinned action versions by major).

1 Quick Start

Repository layout

```
.
|- hello.py
|- Dockerfile
|- .dockerignore
‘- .github/
   ‘- workflows/
      ‘- python-pipeline.yml
```

Files to copy

hello.py

```
print("Hello, world!")
```

Dockerfile

```
FROM python:3.13-slim

WORKDIR /app
COPY hello.py .

CMD ["python", "hello.py"]
```

.dockerignore

```
.github
.git
.gitignore
README.md
```

.github/workflows/python-pipeline.yml

```yaml
name: Python CI/CD Pipeline

on:
  push:
    branches: [ "main" ]
  workflow_dispatch:

# Prevent overlapping runs on the same ref
concurrency:
  group: ci-${{ github.ref }}
  cancel-in-progress: true

jobs:
  lint-and-test:
    name: Lint and Test
    runs-on: ubuntu-latest
    permissions:
      contents: read
    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Set up Python
        uses: actions/setup-python@v5
        with:
          python-version: "3.13"

      - name: Install linters
        run: pip install flake8 pylint

      - name: Lint hello.py
        run: |
          flake8 hello.py
          pylint --disable=R,C hello.py

      - name: Run quick test
        shell: bash
        run: |
          set -euo pipefail
          python hello.py | grep -i 'hello'

  build-and-push:
    name: Build and Push Container
    needs: lint-and-test
    runs-on: ubuntu-latest
    permissions:
      packages: write     # push to GHCR
      contents: read
    steps:
```

```yaml
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v3

      - name: Log in to GitHub Container Registry
        uses: docker/login-action@v3
        with:
          registry: ghcr.io
          username: ${{ github.actor }}
          password: ${{ secrets.GITHUB_TOKEN }}

      - name: Extract Docker metadata
        id: meta
        uses: docker/metadata-action@v5
        with:
          images: ghcr.io/${{ github.repository }}
          tags: |
            type=sha,format=short
            type=sha,format=long
            type=ref,event=branch

      - name: Build and push image
        uses: docker/build-push-action@v5
        with:
          context: .
          push: true
          platforms: linux/amd64,linux/arm64
          tags: ${{ steps.meta.outputs.tags }}
          labels: ${{ steps.meta.outputs.labels }}

  test-image:
    name: Pull and Test Container
    needs: build-and-push
    runs-on: ubuntu-latest
    permissions:
      packages: read       # pull from GHCR
    steps:
      - name: Log in to GitHub Container Registry
        uses: docker/login-action@v3
        with:
          registry: ghcr.io
          username: ${{ github.actor }}
          password: ${{ secrets.GITHUB_TOKEN }}

      - name: Pull image by commit SHA
        run: docker pull ghcr.io/${{ github.repository }}:${{ github.sha }}

      - name: Run image and assert output
        shell: bash
```

```
        run: |
          set -euo pipefail
          docker run --rm ghcr.io/${{ github.repository }}:${{ github.sha }} | grep
    ↪ -i 'hello'
```

2 Why this pipeline

2.1 Stages

Lint & Test ensure code quality early with fast feedback (flake8, pylint, smoke test). **Build & Push** creates a portable artifact (container image) and publishes it to GHCR with metadata tags (short SHA, long SHA, branch ref). **Pull & Test** validates the published artifact in a clean runner to guarantee deployable integrity.

2.2 Design choices

- **Concurrency** cancels overlapping runs on the same ref to reduce wasted compute.
- **Least-privilege permissions**: jobs only request what they need (e.g., `packages: write` only where pushing).
- **Multi-arch images**: `linux/amd64,linux/arm64` broadens runtime targets.
- **Deterministic tagging**: tag by commit SHA and branch for traceability.

3 How to use

1. Create a new repository (optionally include the Python `.gitignore` template).
2. Add the four files exactly as shown and commit to `main`.
3. Open the **Actions** tab: confirm the three jobs run and complete successfully.
4. Open the repo homepage sidebar **Packages** to see your GHCR image and tags.

4 Security notes

- The `GITHUB_TOKEN` is scoped to the repository and permits pushing to GHCR under `ghcr.io/{owner}/{repo}` by default.
- Avoid broad `permissions`; do not grant `packages: write` in jobs that don't push.
- Prefer pinning actions to immutable commit SHAs for stricter supply-chain control when you leave the prototype phase.

5 Optional: README badge

```
![CI](https://github.com/OWNER/REPO/actions/workflows/python-pipeline.yml/badge.svg?
    ↪ branch=main)
```

6 Troubleshooting

- **403 pushing to GHCR**: ensure job has `packages: write` and you use `ghcr.io` as the registry.
- **Image not found in test step**: confirm the tag uses `${{ github.sha }}` and the build job succeeded.
- **Docker buildx fails on arm64**: runners provide QEMU; transient errors usually resolve on rerun. Ensure `setup-buildx-action` ran.