CI Pipeline Lab: Plain Python App → GitHub Actions → Docker Hub
==

GOAL

You will create a tiny plain Python application and set up a GitHub Actions workflow that:
1) Runs tests with pytest
2) Builds a Docker image
3) Pushes the image to Docker Hub
4) (Optional) Publishes a release tag like v1.0.0

PREREQUISITES

- GitHub account
- Docker Hub account
- Git installed locally
- Python 3.11+
- Docker Desktop (optional for local testing)

STEP 1 — Create a new project folder

```
mkdir python-ci-docker-lab
cd python-ci-docker-lab
```

STEP 2 — Create application files
----

app.py

```
def add(a, b):
    return a + b

if __name__ == "__main__":
    print("Hello from Python CI Lab!")
    print("2 + 3 =", add(2, 3))
```

tests/test_app.py

```
from app import add


def test_add():
    assert add(2, 3) == 5
    assert add(-1, 1) == 0
```

requirements.txt

```
pytest==8.3.2
```

Dockerfile

```
FROM python:3.11-slim

WORKDIR /app

COPY requirements.txt .
```

```
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

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

.gitignore

```
__pycache__/
.venv/
.pytest_cache/
.DS_Store
*.pyc
```

STEP 3 — Add GitHub Actions workflow

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

.github/workflows/ci-dockerhub.yml

```
name: ci-dockerhub

on:
  push:
    branches: [ "main" ]
    tags: [ "*" ]
  pull_request:
    branches: [ "main" ]

jobs:
  build-test-push:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout
        uses: actions/checkout@v4

      - name: Set up Python
        uses: actions/setup-python@v5
        with:
          python-version: '3.11'

      - name: Install deps
        run: |
          python -m pip install --upgrade pip
          pip install -r requirements.txt

      - name: Run tests
        run: pytest -q

      - name: Docker meta
        id: meta
        uses: docker/metadata-action@v5
        with:
          images: ${{ secrets.DOCKERHUB_USERNAME }}/python-ci-lab
          tags: |
            type=raw,value=latest,enable={{is_default_branch}}
            type=sha,prefix=sha-,format=short
            type=ref,event=tag

      - name: Set up QEMU
        uses: docker/setup-qemu-action@v3

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

      - name: Login to Docker Hub
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}

      - name: Build and push
        uses: docker/build-push-action@v6
        with:
          context: .
          push: true
          tags: ${{ steps.meta.outputs.tags }}
          labels: ${{ steps.meta.outputs.labels }}
          platforms: linux/amd64
```

STEP 4 — Test locally (optional)

```
python app.py
pytest -q

docker build -t yourname/python-ci-lab:local .
docker run --rm yourname/python-ci-lab:local
```

STEP 5 — Initialize Git & push to GitHub

```
git init
git add .
git commit -m "init lab"
git branch -M main
git remote add origin https://github.com/<your-username>/python-ci-docker-lab.git
git push -u origin main
```

STEP 6 — Add GitHub Secrets

- DOCKERHUB_USERNAME (your Docker Hub username)
- DOCKERHUB_TOKEN (Docker Hub access token)

STEP 7 — Watch CI run

Go to GitHub → Actions tab → select workflow run → view logs.

STEP 8 — Verify image on Docker Hub
----
docker.io/<DOCKERHUB_USERNAME>/python-ci-lab

STEP 9 — Release with a tag (optional)

```
git tag v1.0.0
git push origin v1.0.0
```

TROUBLESHOOTING

- Docker login failed → Check token
- No image → Inspect build logs
- Tests failing → Run pytest locally
- Port in use → Use docker run -p 8080:8000

STRETCH GOALS

- Add staging/prod environments with approvals
- Publish to GHCR/ACR
- Add image scanning
- Deploy to AKS/ECS/App Service