Nice 🚀 — this is a classic **end-to-end workflow**: build locally → test → set up CI/CD → deploy to Azure.  
Let me give you a clear **step-by-step roadmap** with Python + Azure.

**🛠️ Step 1. Develop Locally**

1. **Set up your project** (example: FastAPI app)
2. mkdir myapp && cd myapp
3. python -m venv venv
4. source venv/bin/activate # (Linux/Mac)
5. venv\Scripts\activate # (Windows)
6. pip install fastapi uvicorn pytest
7. **Sample app (app/main.py)**
8. from fastapi import FastAPI
9. app = FastAPI()
10. @app.get("/")
11. def root():
12. return {"message": "Hello from Azure!"}
13. **Run locally**
14. uvicorn app.main:app --reload

**🧪 Step 2. Add Unit Tests**

Create tests/test\_main.py:

from fastapi.testclient import TestClient

from app.main import app

client = TestClient(app)

def test\_root():

response = client.get("/")

assert response.status\_code == 200

assert response.json() == {"message": "Hello from Azure!"}

Run tests:

pytest

**📦 Step 3. Dockerize Your App**

Create a Dockerfile in project root:

FROM python:3.11-slim

WORKDIR /app

COPY requirements.txt .

RUN pip install -r requirements.txt

COPY . .

CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8000"]

Generate requirements.txt:

pip freeze > requirements.txt

Test locally:

docker build -t myapp .

docker run -p 8000:8000 myapp

**🔧 Step 4. Push Code to GitHub**

1. Initialize Git:
2. git init
3. git remote add origin <your\_repo\_url>
4. git add .
5. git commit -m "Initial commit"
6. git push origin main

**⚙️ Step 5. Set Up Azure Resources**

You have two main options:

* **Azure App Service** (easier, PaaS)
* **Azure Kubernetes Service (AKS)** (for scaling)

Let’s use **App Service** for simplicity:

1. Install CLI:
2. az login
3. Create resource group & service plan:
4. az group create --name myResourceGroup --location eastus
5. az appservice plan create --name myPlan --resource-group myResourceGroup --sku B1 --is-linux
6. Create Web App:
7. az webapp create --resource-group myResourceGroup --plan myPlan --name myfastapiapp123 --deployment-container-image-name mydockerhub/myapp:latest

**🔄 Step 6. CI/CD with GitHub Actions**

In your repo, create .github/workflows/deploy.yml:

name: CI/CD to Azure

on:

push:

branches:

- main

jobs:

build-and-deploy:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v3

- name: Set up Python

uses: actions/setup-python@v4

with:

python-version: "3.11"

- name: Install dependencies

run: pip install -r requirements.txt

- name: Run tests

run: pytest

- name: Log in to Azure

uses: azure/login@v1

with:

creds: ${{ secrets.AZURE\_CREDENTIALS }}

- name: Build and Push Docker image

uses: docker/build-push-action@v3

with:

context: .

push: true

tags: mydockerhub/myapp:latest

- name: Deploy to Azure Web App

uses: azure/webapps-deploy@v2

with:

app-name: "myfastapiapp123"

images: "mydockerhub/myapp:latest"

**🔑 Step 7. Configure GitHub Secrets**

* In GitHub repo → **Settings → Secrets and variables → Actions**:
  + AZURE\_CREDENTIALS (JSON from az ad sp create-for-rbac)
  + DOCKERHUB\_USERNAME
  + DOCKERHUB\_TOKEN

**🚀 Step 8. Deploy & Verify**

1. Push code to main branch.
2. GitHub Actions will: test → build → push → deploy.
3. Visit:
4. https://myfastapiapp123.azurewebsites.net/

✅ You now have:

* Local Python app
* Unit tests
* Dockerized image
* CI/CD with GitHub Actions
* Deployment to Azure App Service