**🚀 Secure Deployment of a Dockerized Python App to DigitalOcean using GitHub Actions**

*Created by Vivek Raj*  
💼 DevOps & Cloud Specialist | GitHub Actions | Docker | DigitalOcean | AWS

This beautifully crafted guide walks you through deploying a Python-based app with Google Generative AI integration to a DigitalOcean Droplet using Docker, Nginx, and GitHub Actions.

**🌐 1. Create a DigitalOcean Droplet**

1. Go to <https://cloud.digitalocean.com/droplets>
2. Click **"Create Droplet"**
3. Choose an image:
   * Ubuntu 22.04 x64
4. Choose a plan:
   * Basic → Regular with SSD → (e.g., 2 vCPU, 4 GB RAM)
5. Choose datacenter region (closest to your audience)
6. Authentication:
   * Choose **SSH Key**, and add your public SSH key
7. Finalize and create the droplet
8. After creation, note your public IP address for use in GitHub secrets

**📁 2. Project Structure**

.

├── .github/

│ └── workflows/

│ └── deploy.yml # GitHub Actions workflow

├── deploy/

│ └── app/

│ └── Dockerfile # Production Dockerfile

├── docker-compose.prod.yml # Docker Compose for production

├── main.py # Python entry file

├── requirements.txt # Dependencies

├── .env # Environment variables (in production, use GitHub secrets)

**🐳 3. Dockerfile (deploy/app/Dockerfile)**

FROM python:3.10-slim

WORKDIR /app

COPY requirements.txt .

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

COPY . .

EXPOSE 8080

CMD ["gunicorn", "-w", "4", "-k", "aiohttp.GunicornWebWorker", "--bind", "0.0.0.0:8080", "main:app"]

**📦 4. docker-compose.prod.yml**

version: '3.8'

services:

app\_service:

build:

context: .

dockerfile: deploy/app/Dockerfile

container\_name: ai\_tutor\_app

environment:

- ENVIRONMENT=prod

- GOOGLE\_API\_KEY=${GOOGLE\_API\_KEY}

ports:

- "8080:8080"

**🔐 5. GitHub Secrets to Add**

Navigate to: **GitHub → Repo → Settings → Secrets and variables → Actions**

|  |  |
| --- | --- |
| **Name** | **Description** |
| HOST | Your Droplet IP (e.g., 137.184.239.180) |
| SSH\_USER | Usually root |
| SSH\_PRIVATE\_KEY | Your private SSH key |
| GOOGLE\_API\_KEY | Google API Key |

**⚙️ 6. GitHub Actions Workflow (.github/workflows/deploy.yml)**

name: Deploy to DigitalOcean

on:

push:

branches:

- main

jobs:

deploy:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v3

- name: Set up SSH

run: |

mkdir -p ~/.ssh

echo "${{ secrets.SSH\_PRIVATE\_KEY }}" > ~/.ssh/id\_rsa

chmod 600 ~/.ssh/id\_rsa

ssh-keyscan -H ${{ secrets.HOST }} >> ~/.ssh/known\_hosts

- name: Copy files to Droplet

run: |

rsync -avz --exclude='.git\*' --exclude='\_\_pycache\_\_' ./ ${{ secrets.SSH\_USER }}@${{ secrets.HOST }}:/root/app

- name: Deploy on Droplet

run: |

ssh ${{ secrets.SSH\_USER }}@${{ secrets.HOST }} << 'EOF'

cd /root/app

echo "GOOGLE\_API\_KEY=${{ secrets.GOOGLE\_API\_KEY }}" > .env

docker compose -f docker-compose.prod.yml down || true

docker compose -f docker-compose.prod.yml up --build -d

EOF

**🧰 7. Install Docker & Docker Compose on Droplet**

apt update

apt install -y apt-transport-https ca-certificates curl software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/trusted.gpg.d/docker.gpg

add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

apt update

apt install -y docker-ce docker-ce-cli containerd.io

curl -SL https://github.com/docker/compose/releases/download/v2.27.0/docker-compose-linux-x86\_64 -o /usr/local/bin/docker-compose

chmod +x /usr/local/bin/docker-compose

**🔁 8. GitHub Commands to Push Your Project**

git init

git remote add origin git@github.com:your-username/your-repo.git

# If starting fresh

git add .

git commit -m "Initial commit"

git push -u origin main

# For future updates

git add .

git commit -m "Update app"

git push

Ensure your branch is main or modify the workflow trigger accordingly.

**🔄 9. Switching Hosting Providers (e.g., DigitalOcean → AWS)**

If you're migrating your app to AWS, GCP, or another provider:

1. **Launch a New VM Instance**
   * e.g., EC2 on AWS with Ubuntu 22.04
2. **Update GitHub Secrets**
   * HOST, SSH\_USER, and possibly SSH\_PRIVATE\_KEY
3. **Install Docker & Docker Compose**
   * Follow steps in section 7
4. **Adjust Security Groups/Firewalls**
   * Allow ports 80, 443, or 8080
5. **Redeploy**
   * Just push code to GitHub — GitHub Actions will do the rest 🎯

**🎉 You're Done!**

Your app is now running securely, with CI/CD via GitHub Actions — designed and documented by **Vivek Raj**.

🔗 [Upwork Profile](https://www.upwork.com/freelancers/~014d2fe7b5982e27d1) → | ✉️ Let’s work together on your next DevOps adventure!