Project 2: FastAPI + Docker + CI/CD + AWS ECS (Fargate)

Introduction (ELI10)

Imagine a toy robot that says 'I'm OK!' when you press its button. Our FastAPI app is that robot. The button is a web address `/health`. When you visit it, the app replies with {"status":"ok"}. We pack this app in a Docker container, test it with pytest, and use GitHub Actions as a conveyor belt that ships it to AWS ECS Fargate automatically whenever we make changes.

Architecture Overview

App: FastAPI + Uvicorn
Container: Docker image (amd64)
CI/CD: GitHub Actions for testing and deployment
Registry: Amazon ECR
Runtime: AWS ECS (Fargate)
Logs: CloudWatch
Network: Public subnet + Security Group with port 8000 open

Tools and How They Were Used

• Python 3.11 with FastAPI for building the web app
• Pytest for unit testing `/health`
• Docker for containerization, Docker Buildx for multi-arch builds
• GitHub Actions for CI/CD workflows (`ci.yml` for tests, `deploy.yml` for deploy)
• Amazon ECR for storing Docker images
• Amazon ECS Fargate for serverless container hosting
• CloudWatch for logging task output

File and Folder Layout

project-2/
app/main.py – FastAPI app
tests/test_health.py – unit test
.github/workflows/ci.yml – test workflow
.github/workflows/deploy.yml – deploy workflow
Dockerfile, requirements.txt, pytest.ini
task-def.fixed.json – ECS task definition
README.md

Commands and Explanations

• pytest -q → Ask the robot if it’s OK
• uvicorn app.main:app → Start the robot

• docker build → Pack robot in a box
• docker run → Open the box and test robot
• aws ecr get-login-password | docker login → Access cloud shelf
• docker push → Put the box on the cloud shelf
• aws ecs update-service --force-new-deployment → Deploy new robot
• curl http://IP:8000/health → Press the button, hear OK

CI/CD Pipeline Steps

1. Run tests
2. Build and push Docker image to ECR
3. Render ECS task definition with new image
4. Deploy service on Fargate
5. Smoke-test the `/health` endpoint

Issues Faced and Fixes

• Python not found → Installed python3 and used python3 -m venv
• ModuleNotFoundError for app.main → Fixed sys.path in pytest config
• Docker daemon not running → Started Docker Desktop
• Image arch mismatch → Initially built ARM64 on Mac, caused 'exec format error' on ECS
→ Solution: Used Docker Buildx to build linux/amd64 image and pushed that
• No public IP assigned → Fixed by ensuring subnet had route to Internet Gateway and updating ECS service
• Costs → Avoided ALB, used direct public IP + security group to stay in free tier

Cost Safety

• Scale service to 0 tasks when idle: aws ecs update-service --desired-count 0
• Delete service, cluster, repo, logs when done
• No ALB (saves ~$15/mo)
• ECR + logs = pennies in free tier

Next Steps / Improvements

• Add domain + HTTPS with ALB or CloudFront
• Add database integration
• Add linting, typing, and security scans
• Add autoscaling and monitoring alarms