Solution Explanation – Product Catalog Service

# 1. Objective

The task was to build a backend microservice for a product catalog, with CRUD and search APIs, containerization, CI/CD, and cloud deployment capabilities.

# 2. Technology Choices

1. FastAPI (Python) – async support, auto-docs, strong validation.  
2. SQLite + SQLAlchemy – lightweight and portable, ORM abstraction.  
3. Docker – multi-stage build, smaller image, healthcheck.  
4. GitHub Actions – automated testing and image builds.  
5. Azure Container Apps – simple serverless container hosting.  
6. Kubernetes YAML – for optional AKS/EKS deployment.

# 3. API Design

RESTful API with resource-oriented endpoints. CRUD follows HTTP conventions. Search implemented via `/products/search` with case-insensitive matching. Health endpoint `/health` for monitoring.

# 4. Logging & Monitoring

Python logging with timestamps and levels. Key actions (create/update/delete/search) logged. Docker healthcheck pings `/health`.

# 5. Testing Strategy

Used Pytest to verify basic health and CRUD/search functionality. Ensures minimal regression coverage.

# 6. CI/CD Flow

On push/PR, pipeline installs dependencies, runs tests, builds Docker image, and optionally pushes to registry. This ensures each commit is tested and deployable.

# 7. Deployment Strategy

Local: Python venv or Docker.  
Cloud: Azure Container Apps via provided bash script.  
Optional: Deploy to Kubernetes via included YAML.

# 8. Compliance with Requirements

- CRUD + Search implemented  
- Optional DB used  
- Dockerized  
- CI/CD pipeline in place  
- Cloud deploy script provided  
- Bonus: Health check, logging, Kubernetes YAML  
- Focused on code quality and clarity