🔧 Part 1: Containerize a GenAI App

Let's use a **FastAPI app** that interacts with OpenAI's API.

Step 1.1: Create a `main.py`

```python
# main.py
from fastapi import FastAPI, Request
import openai
import os

openai.api_key = os.getenv("OPENAI_API_KEY")
app = FastAPI()

@app.post("/generate")
async def generate(request: Request):
    data = await request.json()
    prompt = data.get("prompt", "Hello, world")
    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[{"role": "user", "content": prompt}]
    )
    return {"response": response.choices[0].message["content"]}
```

Step 1.2: Create `requirements.txt`

```
fastapi
uvicorn
openai
```

Step 1.3: Dockerfile

```dockerfile
# Use a Python base image
FROM python:3.10-slim

# Set working directory
WORKDIR /app

# Copy code
COPY . .

# Install dependencies
RUN pip install --no-cache-dir -r requirements.txt

# Expose the port
EXPOSE 8000

# Run the FastAPI server
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

Step 1.4: Build and push Docker image

```
# Build Docker image
docker build -t your-dockerhub-username/genai-api .

# Push to Docker Hub
docker push your-dockerhub-username/genai-api
```

☁️ Part 2: Deploy on AWS Fargate (ECS)

Step 2.1: Create ECS Cluster (Fargate)

1. Go to AWS Console → ECS → Create Cluster
2. Choose “Networking Only (Fargate)”
3. Name it `genai-cluster`

Step 2.2: Create Task Definition

1. Go to ECS → Task Definitions → Create new
2. Launch type: FARGATE
3. Set:
    - Image: `your-dockerhub-username/genai-api`
    - CPU: 0.5 vCPU
    - Memory: 1 GB
    - Port mappings: 8000
4. Set ENV variable: `OPENAI_API_KEY` (via Secrets Manager)

Step 2.3: Create ECS Service

1. Create a new service from your task definition
2. Attach to a VPC + public subnet
3. Enable auto-assign public IP
4. Use Application Load Balancer to expose `/generate` API

Step 2.4: Test API

Use curl or Postman:

```
curl -X POST http://<load-balancer-dns>/generate -H "Content-Type:
application/json" \
```

```
-d '{"prompt": "What is Generative AI?"}'
```

💡 Alternative: Azure Container Apps

Use the same Docker image:

```
az containerapp create \
  --name genai-app \
  --resource-group my-rg \
  --image your-dockerhub-username/genai-api \
  --environment my-env \
  --target-port 8000 \
  --ingress external \
  --env-vars OPENAI_API_KEY=your-key
```

📈 Part 3: Monitoring with AWS CloudWatch

Step 3.1: Enable CloudWatch Logs

1. In ECS task definition, under “Log Configuration”:
    - Log driver: `awslogs`
    - Log group: `/ecs/genai-app`
2. Logs will show:
    - Request hits
    - Error messages
    - Token usage if you log it

(Optional) Modify `main.py` to include logging:

```
import logging
logging.basicConfig(level=logging.INFO)

@app.post("/generate")
async def generate(request: Request):
    ...
    logging.info(f"Prompt: {prompt}, Response Length:
    {len(response.choices[0].message['content'])}")
```

📊 Alternative: Prometheus + Grafana (Kubernetes)

If you’re deploying via Kubernetes:

- Use [Prometheus FastAPI Middleware]

- Export metrics at `/metrics`
- Visualize in Grafana with custom dashboards for:
    - Inference latency
    - Response length
    - Request count per minute

✅ Lab Completion Summary

You have:

- Containerized a GenAI app using FastAPI
- Deployed it using **AWS Fargate** or **Azure Container Apps**
- Monitored it using **CloudWatch logs** or **Prometheus**