Docker Kubernetes Containerization

High-Level Steps: I won't go into complete production grade setup (comes with significant local setup efforts or cost at cloud), but this should be sufficient to start, this is highly customizable.

1. Create a Dockerfile for the application
2. Build the Docker Image
3. Push the Image to a Registry
4. Write Kubernetes Deployment and Service Manifests
5. Deploy to Kubernetes

1. Dockerfile (Dockerfile)

```
```dockerfile
FROM python:3.12.2-slim
# Use an appropriate base image

WORKDIR /app # I typically use /app

# Install system-level dependencies (if you have any)
RUN apt-get update && apt-get install -y \
   <dependency1> \
   <dependency2>  # Example:  swig

# Copy dependency and code files
COPY requirements.txt ./
COPY ingest_data.py app.py query_data.py cli_app.py ./

# Install dependencies
RUN pip install -r requirements.txt

# Expose the port used by your Gradio app
EXPOSE 8000

# Define the command to start the application
CMD ["python", "app.py"]
```
```

2. Build the Docker Image

```
```
docker build -t resume-qa-app:latest .
```
```

Replace `resume-qa-app:latest` with any desired image name and tag.

3. Push the Image to a Container Registry

Choices: Docker Hub, AWS ECR, Google Container Registry, etc.

Log in to chosen registry.

Example (Docker Hub):

```
docker tag resume-qa-app:latest your-dockerhub-username/resume-qa-app:latest
docker push your-dockerhub-username/resume-qa-app:latest
```

4. Kubernetes Manifests

a. Deployment (deployment.yaml)

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: resume-qa-deployment
spec:
  replicas: 1 # Adjust scaling if needed
  selector:
    matchLabels:
      app: resume-qa
  template:
    metadata:
      labels:
        app: resume-qa
    spec:
      containers:
      - name: resume-qa
        image: your-dockerhub-username/resume-qa-app:latest
        ports:
        - containerPort: 8000
        resources:
          requests:
            memory: "256Mi"  # Example: Request 256 MB of memory
            cpu: "500m"      # Example: Request half a CPU core
          limits:
            memory: "512Mi"  # Example:  Limit to 512 MB of memory
            cpu: "1"         # Example: Limit to one CPU core
# Add resource limits if needed
```

b. Service (service.yaml)

```
```yaml
apiVersion: v1
kind: Service
metadata:
  name: resume-qa-service
spec:
  type: LoadBalancer  # Or 'NodePort' for local testing
  selector:
    app: resume-qa
  ports:
  - port: 80
    targetPort: 8000
```
```

5. Deploy to Kubernetes

```
```
kubectl apply -f deployment.yaml
kubectl apply -f service.yaml
```
```