CSE-816(Final Project) - VQA_MLOps

GitHub Repository: GitHub Repo
Youtube Project Demo: YouTube Demo

Team Members

- Trupti Khodwe(IMT2022007)
- Rishabh Dixit (IMT2022051)

---

1. Project Description

This project implements a complete **MLOps (Machine Learning Operations) pipeline** for deploying a **Visual Question Answering (VQA) Agent**. The system automates the lifecycle of the ML model from code commit to production deployment using industry-standard DevOps tools.

Key capabilities include:

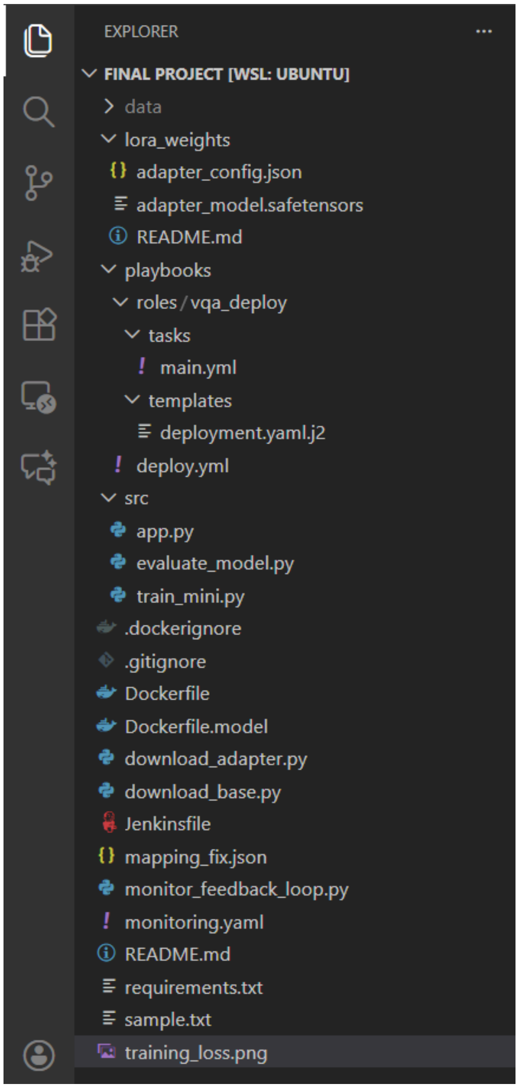
- **Hybrid Model Architecture:** Utilizes a lightweight text model (bert-tiny) for rapid CI simulation and a heavy VQA model (blip-vqa-capfilt-large) for production inference.
- **Efficient Artifact Management:** Implements "Model-as-Image" patterns and LoRA adapter integration to handle large weights efficiently.
- **Infrastructure-as-Code:** Uses Ansible for templated, dynamic Kubernetes deployments.
- **Observability & Metrics:** Full integration with the ELK Stack to track Latency and Model Confidence Scores.
- **Self-Healing Feedback Loop:** An automated watchdog service that triggers retraining pipelines if model quality drops below a threshold.
- **Scalability:** Horizontal Pod Autoscaling (HPA) to handle traffic bursts.

---

2. Architecture & Tech Stack

Component	Tool Used	Purpose
Source Control	**GitHub**	Version control for code and Infrastructure-as-Code manifests.
CI Automation	**Jenkins**	Automates testing, building, and orchestration.
Containerization	**Docker**	Builds optimized, layered images for the application and model artifacts.

Component	Tool Used	Purpose
Configuration	**Ansible**	Manages Kubernetes manifests via Jinja2 templating.
Orchestration	**Minikube (K8s)**	Local Kubernetes cluster for deployment.
Monitoring	**ELK Stack**	Visualizes logs and tracks quality_score & latency metrics.
Model	**HuggingFace**	Hosts the Base Model and Fine-Tuned LoRA Adapters.
Automation	**Python Watchdog**	Closes the MLOps loop by triggering Jenkins based on live metrics.



```
FINAL PROJECT [WSL: UBUNTU]
  > data
  v lora_weights
      adapter_config.json
      adapter_model.safetensors
      README.md
  v playbooks
    v roles / vqa_deploy
      v tasks
          main.yml
      v templates
          deployment.yaml.j2
      deploy.yml
  v src
      app.py
      evaluate_model.py
      train_mini.py
    .dockerignore
    .gitignore
    Dockerfile
    Dockerfile.model
    download_adapter.py
    download_base.py
    Jenkinsfile
    mapping_fix.json
    monitor_feedback_loop.py
    monitoring.yaml
    README.md
    requirements.txt
    sample.txt
    training_loss.png
```

3. Prerequisites & Installation

A. Environment Setup (WSL/Linux)

Ensure the following tools are installed:

1. **Docker**.
2. **Minikube & Kubectl**: For local cluster management.
3. **Ansible**: pip install ansible kubernetes.
4. **Ngrok**: To expose Jenkins to GitHub Webhooks.

B. One-Time Data Setup

Since the dataset and models are massive (>3GB), we do not build them into the Docker image. We use **Host Volume Mounting**.

1. **Download Models locally:** Run the provided helper scripts to fetch weights into your data/ folder: bash python3 download_base.py # Saves to ./data/blip-large python3 download_adapter.py # Saves to ./data/vqa-adapter
2. **Verify Structure:** Ensure ~/Final Project/data/ contains: blip-large/, vqa-adapter/, abo-images-small/, and VQA_Dataset.csv.

C. Cluster Initialization

Start Minikube with sufficient resources for the AI Model + ELK Stack.

```
minikube start
```

D. Permission Fixes (Critical for Jenkins)

Allow the Jenkins user to access the Minikube configuration files:

```
sudo chmod +x /home/$USER
sudo chmod -R 777 /home/$USER/.minikube
sudo chmod -R 777 /home/$USER/.kube
sudo chmod a+x /home/user
```

4. Critical Manual Commands (The "Glue")

These commands must be running in separate terminal windows for the project to function.

1. Data Mounting (The Bridge)

Connects your local hard drive (Dataset + Models) to the Minikube VM.

```
# Run in Project Root
minikube mount ./data:/data
```

- Why: Allows the Pods to read the 3GB dataset without downloading it.

2. Application Tunnel (Accessing the AI)

Exposes the VQA Agent to localhost:5000.
kubectl port-forward svc/vqa-service 5000:80 --address 0.0.0.0

- Why: Minikube networks are isolated. This creates a direct pipe for curl.

3. Kibana Tunnel (Accessing Logs)

Exposes the Monitoring Dashboard to localhost:5601.
kubectl port-forward -n logging svc/kibana 5601:5601 --address 0.0.0.0

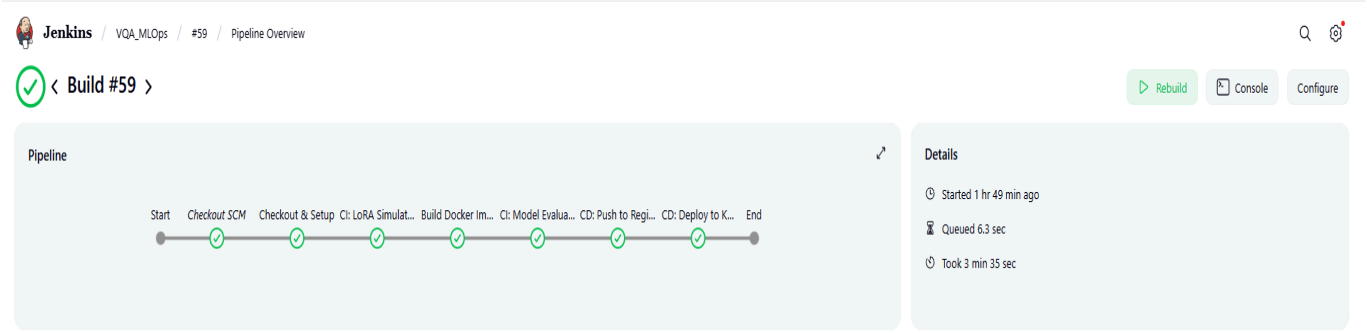
4. Elasticsearch Tunnel (For Watchdog Script)

Exposes the ES API for the feedback loop script.
kubectl port-forward -n logging svc/elasticsearch 9200:9200 --address 0.0.0.0

5. The CI/CD Pipeline Workflow

The Jenkinsfile defines a 6-stage pipeline triggered automatically by a Git Push.

- Checkout: Pulls the latest code from GitHub.
- CI: LoRA Simulation: Runs train_mini.py to simulate the training process and generate dummy artifacts.
- Build Docker Images: Builds:
- vqa-app: The Flask Application.
- vqa-model: A layered container for model weights.
- CI: Model Evaluation: Runs evaluate_model.py inside a container. It loads the Real Model from the mounted disk and tests inference on a subset of real data.
- CD: Push to Registry: Pushes tagged images to Docker Hub (rishabh720/vqa-app).
- CD: Deploy to Kubernetes:
- Loads images directly into Minikube (bypassing internet/bandwidth issues).
- Uses Ansible Roles (playbooks/roles/vqa_deploy) to render deployment.yaml.j2 with the new tag.
- Applies the deployment + HPA + ELK Stack.



Started 1 hr 49 min ago

Queued 6.3 sec

Took 3 min 35 sec

6. Advanced Features & Innovation

A. Modular Ansible Roles

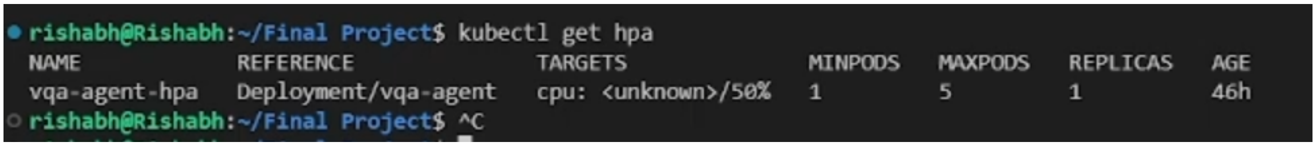
- Instead of a monolithic playbook, we use a modular design:
- Role: playbooks/roles/vqa_deploy
- Function: Separates template rendering and manifest application logic, ensuring scalability and maintainability.

B. Quality & Performance Metrics

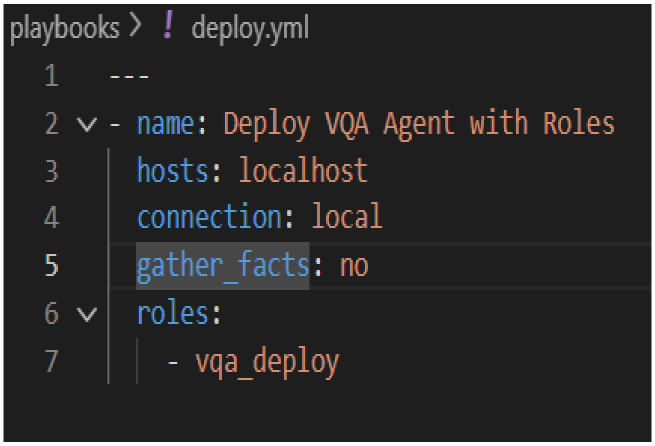
- The application calculates and logs specific metrics for every inference request:
- quality_score: The model's confidence probability (Softmax) in its generated answer.
- latency_seconds: The exact time taken to process the image and generate text.

C. Automated Feedback Loop (Self-Healing)

- A watchdog script (monitor_feedback_loop.py) runs alongside the cluster.
- It queries Elasticsearch for recent quality_score metrics.
- If the average score drops below a threshold (e.g., 0.75).
- It automatically triggers the Jenkins Pipeline via API to retrain/redeploy the model.

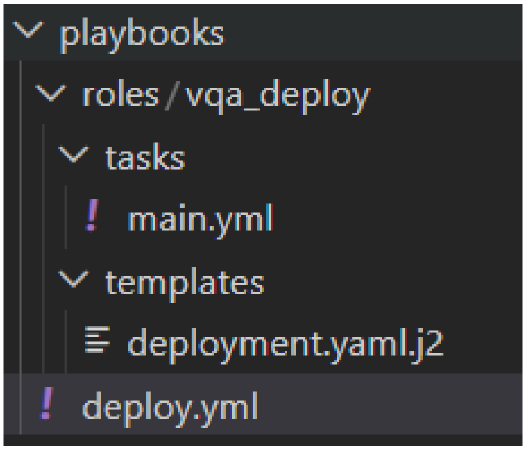


```
rishabh@Rishabh:~/Final Project$ kubectl get hpa
NAME            REFERENCE               TARGETS            MINPODS   MAXPODS   REPLICAS   AGE
vqa-agent-hpa   Deployment/vqa-agent    cpu: <unknown>/50%   1         5         1          46h
rishabh@Rishabh:~/Final Project$ ^C
```

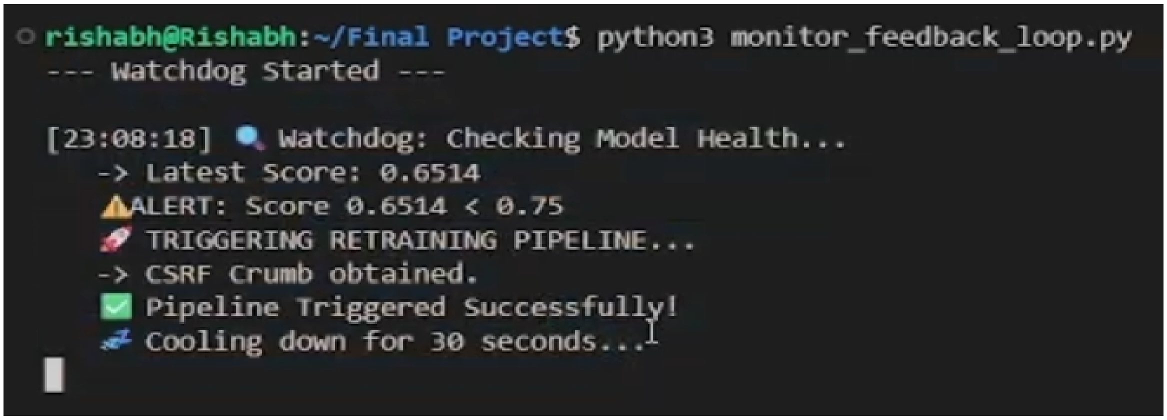


playbooks > deploy.yml

```
---
- name: Deploy VQA Agent with Roles
  hosts: localhost
  connection: local
  gather_facts: no
  roles:
    - vqa_deploy
```



```
playbooks
  roles / vqa_deploy
    tasks
      main.yml
    templates
      deployment.yaml.j2
  deploy.yml
```



```
○ rishabh@Rishabh:~/Final Project$ python3 monitor_feedback_loop.py
--- Watchdog Started ---

[23:08:18] 🔍 Watchdog: Checking Model Health...
    -> Latest Score: 0.6514
    ⚠️ALERT: Score 0.6514 < 0.75
    🚀 TRIGGERING RETRAINING PIPELINE...
    -> CSRF Crumb obtained.
    ✅ Pipeline Triggered Successfully!
    💤 Cooling down for 30 seconds...
```

7. How to Test & Demo

A. Trigger the Pipeline

- Make a change to the code.
- Push to GitHub: git push origin main.
- Watch the build succeed in Jenkins.

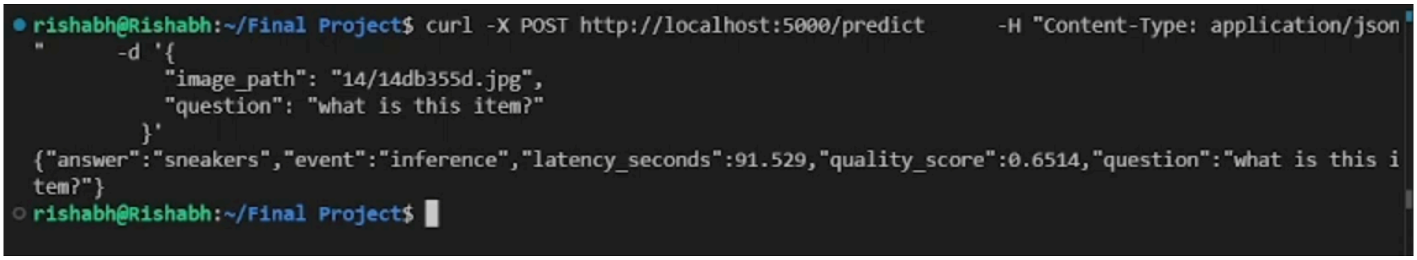
B. Verify "Real Intelligence"

With the Port Forward (Step 4.2) active, run this request:

```
curl -X POST http://localhost:5000/predict \
  -H "Content-Type: application/json" \
  -d '{
     "image_path": "14/14db355d.jpg",
     "question": "what is this item?"
    }'
```

Expected Output:

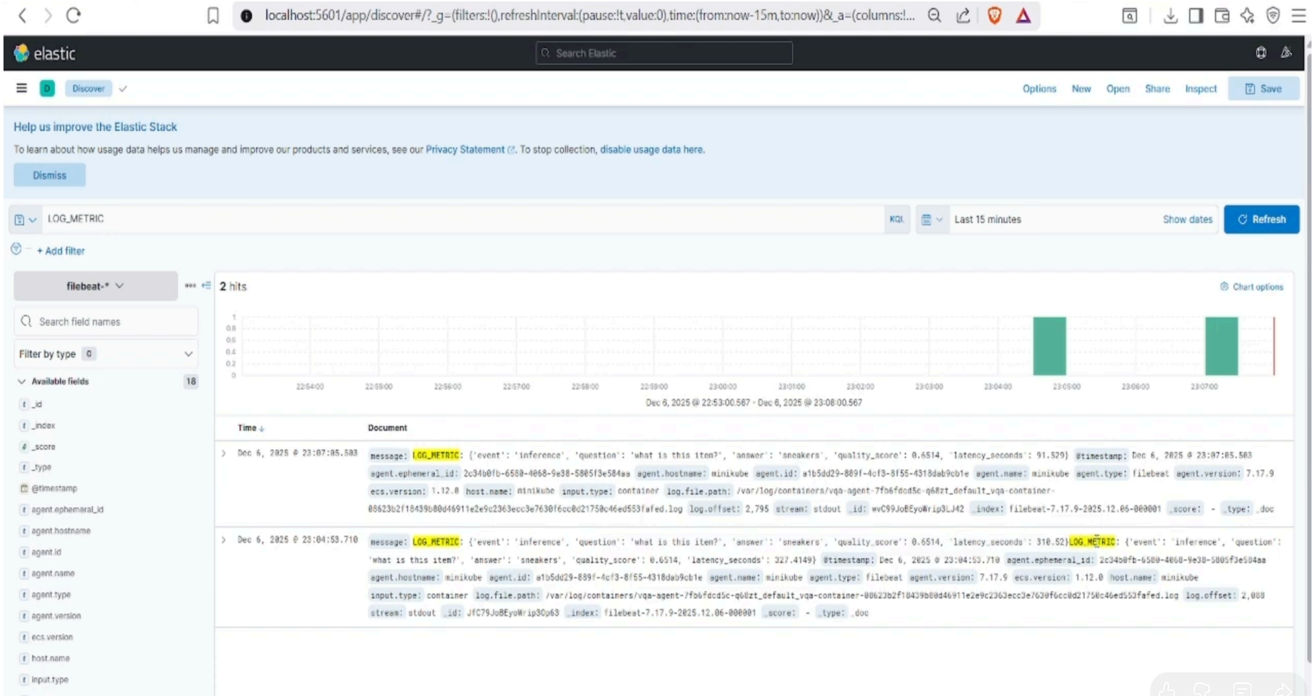
```
{
 "answer": "sneakers",
 "event": "inference",
 "latency_seconds": 90.5573,
 "quality_score": 0.6514,
 "question": "what is this item?"
}
```



```
rishabh@Rishabh:~/Final Project$ curl -X POST http://localhost:5000/predict      -H "Content-Type: application/json
"      -d '{
           "image_path": "14/14db355d.jpg",
           "question": "what is this item?"
         }'
{"answer":"sneakers","event":"inference","latency_seconds":91.529,"quality_score":0.6514,"question":"what is this i
tem?"}
rishabh@Rishabh:~/Final Project$ 
```

C. Verify Monitoring

- Open Browser: http://localhost:5601.
- Go to Discover.
- Search for: "LOG_EVENT".
- Evidence: You will see the structured JSON logs containing the quality_score and latency.



LOG_METRIC

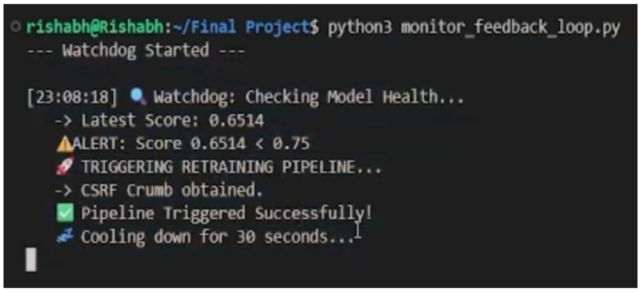
```
Dec 6, 2025 @ 23:07:05.503   message: LOG_METRIC: {'event': 'inference', 'question': 'what is this item?', 'answer': 'sneakers', 'quality_score': 0.6514, 'latency_seconds': 91.529} @timestamp: Dec 6, 2025 @ 23:07:05.503
                             agent.ephemeral_id: 2c34b0fb-6580-4068-9e38-5805f3e584aa agent.hostname: minikube agent.id: a1b5dd29-889f-4cf3-8f55-4318dab9cb1e agent.name: minikube agent.type: filebeat agent.version: 7.17.9
                             ecs.version: 1.12.0 host.name: minikube input.type: container log.file.path: /var/log/containers/vqa-agent-7fb6fdcd5c-q68zt_default_vqa-container-
                             08623b2f18439b80d46911e2e9c2363ecc3e7630f6cc0d21750c46ed553fafed.log log.offset: 2,795 stream: stdout _id: wvC99JoBEyoWrip3LJ42 _index: filebeat-7.17.9-2025.12.06-000001 _score: - _type: _doc

Dec 6, 2025 @ 23:04:53.710   message: LOG_METRIC: {'event': 'inference', 'question': 'what is this item?', 'answer': 'sneakers', 'quality_score': 0.6514, 'latency_seconds': 310.52}LOG_METRIC: {'event': 'inference', 'question':
                             'what is this item?', 'answer': 'sneakers', 'quality_score': 0.6514, 'latency_seconds': 327.4149} @timestamp: Dec 6, 2025 @ 23:04:53.710 agent.ephemeral_id: 2c34b0fb-6580-4068-9e38-5805f3e584aa
                             agent.hostname: minikube agent.id: a1b5dd29-889f-4cf3-8f55-4318dab9cb1e agent.name: minikube agent.type: filebeat agent.version: 7.17.9 ecs.version: 1.12.0 host.name: minikube
                             input.type: container log.file.path: /var/log/containers/vqa-agent-7fb6fdcd5c-q68zt_default_vqa-container-08623b2f18439b80d46911e2e9c2363ecc3e7630f6cc0d21750c46ed553fafed.log log.offset: 2,088
                             stream: stdout _id: JfC79JoBEyoWrip30p63 _index: filebeat-7.17.9-2025.12.06-000001 _score: - _type: _doc
```

D. Verify Feedback Loop

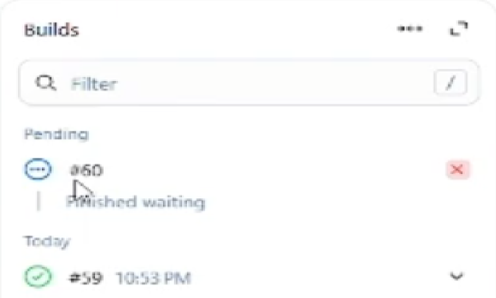
Run the watchdog script manually to simulate the automated check:
python3 monitor_feedback_loop.py

- Output: It will detect the score (0.65) is below threshold (0.75) and print: 🚀 TRIGGERING RETRAINING PIPELINE….
- Result: A new build will appear in Jenkins.



```
○ rishabh@Rishabh:~/Final Project$ python3 monitor_feedback_loop.py
--- Watchdog Started ---

[23:08:18] 🔍 Watchdog: Checking Model Health...
   -> Latest Score: 0.6514
   ⚠️ALERT: Score 0.6514 < 0.75
   🚀 TRIGGERING RETRAINING PIPELINE...
   -> CSRF Crumb obtained.
   ✅ Pipeline Triggered Successfully!
   💤 Cooling down for 30 seconds...
█
```



Builds
Filter
/
Pending
#60
Finished waiting
Today
#59 10:53 PM