Smart Posture Analyzer using KEDA and K3s - Documentation

This project performs intelligent posture analysis using AI models on containerized jobs, distributed across GPU-equipped nodes based on load. It uses KEDA (Kubernetes Event-Driven Autoscaler) and K3s to manage resource-aware orchestration.

✅Overview

This system is designed to:

- Receive posture images via MQTT
- Run posture analysis using MediaPipe in Docker containers
- Use Prometheus GPU metrics to find the least loaded node
- Let KEDA scale and dispatch posture-analysis jobs to that node
- Log the results into a PostgreSQL database (Supabase-hosted)

♒Requirements

🛜Software Dependencies

Install using:

```
pip install -r requirements.txt
```

Dependencies include:

- opencv-python
- mediapipe
- numpy
- paho-mqtt
- psycopg2-binary
- matplotlib
- psutil
- prometheus-api-client
- keyboard

🔢Environment Setup

- K3s Kubernetes cluster (master + GPU workers)

- KEDA installed in the cluster
- Prometheus monitoring GPU metrics:
- `jetson_gpu_usage_percent`
- `jetson_orin_gpu_load_percent`
- Docker (with buildx enabled)
- Docker Hub account (or local registry)
- Supabase PostgreSQL (or alternative DB)

☢️ Folder & File Overview

File/Folder	Description
`build_and_push.py`	Builds Docker image, sets up taints, launches monitors
`mqtt_posture_analyzer_with_db.py`	Core script: receives image, runs MediaPipe, logs result
`gpu_affinity_watcher.py`	Selects GPU node with least load and patches ScaledJob
`patch_scaledjob.py`	Patches `posture-job.yaml` with nodeAffinity for selected node
`cpu_monitor_and_offload.py`	Custom KEDA gRPC scaler using Prometheus metrics
`stop.py`	Gracefully stops all posture jobs, deletes pods, untaints master
`posture-job.yaml`	Base KEDA job template (gets patched)
`patched-job.yaml`	Auto-generated job file with nodeAffinity
`externalscaler.proto`	gRPC definition for scaler interface
`externalscaler_pb2*.py`	Auto-generated protobuf/gRPC bindings
`Dockerfile`	Builds posture analyzer container
`docker-compose.yml`	Optional: run container locally for test/debug
`requirements.txt`	Python dependencies list
`last_node.txt`	Remembers the last selected node to prevent rescheduling

---

🧳 Setup Instructions

1. Configure Prometheus

Ensure Prometheus is scraping GPU usage metrics for all nodes:

- Node 1: `jetson_gpu_usage_percent`
- Node 2: `jetson_orin_gpu_load_percent`

2. Build and Push Docker Image

```
python3 build_and_push.py
```

This will:

- Stop/remove existing posture containers
- Build multi-arch image: `shahroz90/posture-analyzer`
- Patch and apply the KEDA ScaledJob
- Start GPU watcher, CPU monitor, and ESC listener

3. MQTT Broker

Ensure broker is running at `192.168.1.79` or modify the IP inside `mqtt_posture_analyzer_with_db.py`. Devices should send base64 JPEGs to:

- `images/pi1`
- `images/pi2`, etc.

4. Supabase Database Table

```
CREATE TABLE posture_log (
  id SERIAL PRIMARY KEY,
  pi_id TEXT,
  filename TEXT,
  received_time TIMESTAMP,
  analyzed_time TIMESTAMP,
  neck_angle INTEGER,
  body_angle INTEGER,
  posture_status TEXT,
  landmarks_detected BOOLEAN,
  processed_by TEXT
);
```

🛑 Runtime Workflow

Once you run:

```
python3 build_and_push.py
```

This happens:

1. **gpu_affinity_watcher.py** selects the best GPU node
2. **patch_scaledjob.py** updates `posture-job.yaml` with nodeAffinity
3. **cpu_monitor_and_offload.py** runs gRPC scaler for KEDA
4. **KEDA** queries Prometheus metrics
5. If eligible, **KEDA triggers posture job** on the chosen node
6. Container runs `mqtt_posture_analyzer_with_db.py`:
7. Subscribes to MQTT topic
8. Decodes image
9. Runs MediaPipe analysis
10. Annotates & saves image (Optional)
11. Inserts metadata into PostgreSQL

🛑 Stopping the System

```
python3 stop.py
```

This will:

- Kill GPU/CPU watcher processes
- Delete all Kubernetes pods and jobs
- Untaint master node if tainted
- Delete `patched-job.yaml`

🧳 Testing the System

1. Run `python3 build_and_push.py`
2. Publish a test image to topic: `images/pi1`
3. Posture-analyzer job should trigger on selected node
4. (Optional) Annotated image saved to `./analyzed_images/analyzed_images_from_pi1/`
5. Result logged in PostgreSQL

✅Summary

- Uses KEDA to autoscale posture jobs based on GPU availability
- Patches job templates dynamically to enforce nodeAffinity
- Combines Docker, Prometheus, and MQTT in a smart edge AI pipeline
- Ensures posture analysis is always run on best-performing node