Deployment & Operations Guide

Patrol Sight — on-premises smart-glasses face recognition

HumanityAI'd — July 2026

CONFIDENTIAL — FOR CUSTOMER IT, SECURITY OPERATIONS & INTEGRATORS

1. Deployment model

Everything runs on **one GPU server inside the customer's network**. No component depends on the public internet after installation. The system deploys as a **single Docker Swarm stack (`patrol-sight`)** of 34 services across five tiers, brought up and down as a unit by one script.

Tier	Contents	Update strategy
1 — Stateful	PostgreSQL 15, MinIO object store, 4× Redis	Careful, data-bearing
2 — CPU stateless	Migration gate, connection pooler, auth service, business API, recognition orchestrator, device hub, C2 frontend, glasses-settings API	Rolling
3 — GPU	Recognition engine (`patrol-sight-ai`), FAISS index worker, face-quality engine	Stop-first, parallelism 1
4 — Infrastructure	nginx, NATS, TURN relay, optional remote-access tunnel	Stop-first, quick restart
5 — Optional	Second product (museum AR guide) on an isolated overlay network, where licensed	Independent

All **22 data volumes are declared external**, so tearing the stack down cannot delete customer data. Each release keeps a byte-for-byte copy of the stack file that produced it, and every deploy writes a pre/post image snapshot for rollback.

2. Hardware sizing

Component	Minimum	Recommended	Notes
GPU	8 GB (RTX 4060 Ti class, compute 8.9) — the measured reference	**16–24 GB** (RTX 4090 / L4 / L40 class)	8 GB is proven for the reference workload; more VRAM buys concurrent glasses and index headroom
System RAM	16 GB	**32 GB**	Measured budget ≈12 GB (Postgres 4 GB shared buffers, Redis 2 GB, API workers 2–4 GB, FAISS CPU 1–2 GB, OS 2 GB)
CPU	6 cores	8+ cores	H.264 decode fallback, ffmpeg, API workers

Component	Minimum	Recommended	Notes
Storage	256 GB SSD	512 GB–1 TB SSD	Face images, FAISS indexes, audit logs, object store
Network	1 GbE + dedicated 5 GHz AP	1 GbE + **WiFi 6, 5 GHz, power-save disabled**	See §4 — the radio is the single biggest latency variable

GPU and index memory (measured). The 512-dimension ArcFace index costs roughly **1 GB VRAM per 1M faces** (≈2 GB at 13M) thanks to IVF-PQ compression — 962 MB for 13M vectors versus 26 GB for an uncompressed flat index. The production reference index (93,728 persons) occupies about **15 MB of GPU memory**. VRAM pressure comes from the models and concurrent decode, not the watchlist size.

Driver and platform requirements. NVIDIA driver 525+ (550+ recommended), CUDA 11.8 minimum (12.4+ recommended), cuDNN 8.6+ (8.9+ recommended), and the NVIDIA Container Toolkit with **nvidia set as Docker's default runtime**. Supported GPUs span GTX 1080 Ti through RTX 5090 and the datacentre line (V100/A100/H100/L4/L40). *Note for RTX 50-series (sm_120) hosts: FAISS must be compiled with explicit CUDA architectures — the standard* `faiss-gpu` *wheel does not include sm_120.*

3. Install (outline)

1. Install Docker with the NVIDIA container runtime as default; verify `nvidia-smi` inside a container.
2. Initialise Swarm and label the node for GPU placement (`docker node update --label-add gpu=true`).
3. **Provision secrets at install time** — database passwords, JWT and session secrets, admin credentials, API keys, TURN users, NATS Nkeys. Never accept shipped defaults; see §7.
4. Create the external volumes and place TLS certificates.
5. `./deploy.sh up` — brings up all five tiers in health-gated order; database migrations run automatically behind a schema gate that blocks the API until current.
6. `./deploy.sh health` — verifies nginx, business API, recognition engine, recognition orchestrator, face-quality engine and object store. Confirm the GPU is detected and TensorRT (not ONNX) is in use.
7. Configure the glasses: set the server's static LAN IP and stream port in the app build, then `./deploy-glasses.sh` to provision each device over USB.

Expected non-200 responses (these are correct, not faults): the business API root returns **401** (auth required), the recognition orchestrator root returns **404** (no root route), the object-store init job sits at **0/1** having completed its one-shot work, and raw `curl` against the device-hub API returns **400** because payloads are AES-256-GCM encrypted by design.

4. The glasses link — the thing that decides your latency

The wireless link, not the AI, dominates the end-user experience. Measured on the reference deployment:

Configuration	Throughput	Ping	Result
2.4 GHz hotspot, WiFi power-save on	72 Mbps	~100 ms	Unusable for live overlay
5 GHz WiFi 6, power-save disabled	480 Mbps	**~4 ms**	2560×1440 at 20 Mbps, 0% packet loss

Requirements that follow:

- **Dedicated 5 GHz SSID** for the glasses fleet, with adequate AP density along patrol routes. The reference glasses (RayNeo X3 Pro) are **5 GHz only — no WiFi 6E / 6 GHz**.
- **Disable WiFi power-save** on the glasses profile; it alone accounted for a ~25× ping penalty.
- Budget roughly **20 Mbps per streaming device** plus overhead; each 1080p frame is about 83 KB across ~57–60 packets.
- Keep the glasses VLAN **physically isolated with no internet uplink**. The video and result path is currently plaintext UDP and is safe on that basis — treat the isolation as a security control, not a convenience (see §7).

5. Operating runbook

Enrol a person. Register through the operator console (single or bulk). Measured bulk throughput is **40–60 images/second** with workers — about 15–25 seconds for 1,000 images. The FAISS index worker rebuilds shards and hot-reloads them into the running engine without a restart.

Assign categories and alert colours. Each enrolled person carries a label and colour, delivered to the lens as a compact encoded pill (`name @@category@@ #colour`), so the officer reads threat category at a glance rather than a bare name. Ten colours are verified end-to-end.

Monitor live operations. The stream monitor shows per-device frames-per-second, latency and face counts with an annotated preview per pair of glasses. A browser-webcam test page lets an operator validate recognition without glasses. Alerts publish on the messaging bus for downstream command-and-control integration.

Tune thresholds per site. Recognition and detection thresholds are deployment-configurable. Two calibration facts matter operationally: the compressed index reads similarities at roughly **0.86× exact cosine** (so a 0.45 index threshold ≈ 0.52 true cosine), and raising the detection confidence threshold from 0.35 to 0.60 removed **2–11 false detections per frame** at high detector resolution. Agree thresholds during the pilot and record them per site.

Know the recognition envelope. Reliable identification to about **5 metres**, maximum 6–7 metres in good lighting; beyond that faces fall below the 40×40-pixel minimum and are rejected rather than guessed at. A long-range mode raises detector resolution for 8–10 metres at reduced frame rate. Off-angle faces are handled by the heavier ArcFace R50 model, chosen because it produced **~60% fewer false negatives at extreme pose** than the lightweight alternative.

Backups. Nightly snapshot of the PostgreSQL volume plus the FAISS index, face-image and audit-log volumes. The FAISS index can be rebuilt from the database if lost. Because volumes are external, upgrades never touch them.

Upgrades. `./deploy.sh update <service>` forces a rolling replacement of individual services; environment or secret changes require a full stack redeploy since environment is resolved at deploy time. The GPU tier updates stop-first with parallelism 1 to avoid two engines contending for the same card.

Restart and boot behaviour. The stack auto-restores from Swarm state on boot, backed by a systemd oneshot unit that runs the same deploy script. A GPU startup wrapper waits for the driver and retries specifically on the FAISS CUDA-999 initialisation race — so a cold boot recovers without human help.

Benign log noise. A `faiss.swigfaiss_avx2` import error at worker startup is FAISS falling back to its base implementation and is harmless.

6. Capacity & scaling

- **One server ≈ one site.** A single GPU is shared across the recognition engine, index worker and quality engine by CUDA time-slicing.
- The reference GPU sustains **14–20 effective recognition frames per second** per stream with an end-to-end glasses-to-label time of **~68 ms typical (49–103 ms)**. The two-layer tracker is what makes this affordable: it answers 97.6% of frames from cache, cutting average per-frame latency 65 → 30 ms and GPU utilisation 85% → 40%.
- **Index scale is not the constraint.** Measured recognition latency by catalogue size: 10K → 40–60 ms, 100K → 50–70 ms, 1M → 60–90 ms, 5M → 70–100 ms, **13M → 80–130 ms**.
- **Concurrent-device capacity is not yet published as a measured number** — it is bounded by GPU decode and detection, not by search. Size fleets during the pilot and hold us to the figure we measure there. This is stated as a known gap in the Technical Due-Diligence Report rather than estimated here.
- **Scale-out path**: the multi-GPU configuration (one GPU per engine replica) is already written into the stack file and commented out, ready to enable on a multi-card host for larger sites.

7. Security & privacy posture

What is genuinely strong

- Fully on-premises: face images, embeddings, watchlist and analytics never leave the customer's network. There is no cloud dependency in the recognition path.

- **Zero-trust service messaging** (per-service Nkey authentication on the bus) and **AES-256-GCM payload encryption** between the device hub and the command console.
- JWT authentication with refresh, API-key middleware, an IP allowlist on the glasses stream port, TLS termination at nginx, and audit middleware with an operator-facing audit-log view.
- A **privacy-preserving device mode** that transmits only the 512-number faceprint (~2 KB) instead of the image (~100 KB) — useful where imagery must not traverse the network at all.

What must be closed before production, and how to operate until then

- **Rotate every credential at install.** The stack ships with default admin, API-key, database and session-secret values, and a master API key is currently embedded in the Android app source and its APK. Rotation and per-device tokens are Phase 0 of the hardening plan.
- **Collapse the network surface.** Several datastore and service ports publish on the host alongside nginx; the stack file ships commented-out private-IP bindings for exactly this purpose. Bind them to the LAN IP or remove them.
- **The glasses transport is plaintext.** Until encrypted transport ships, the isolated-VLAN-with-no-uplink requirement in §4 is a hard control, not a recommendation.
- **The ops container mounts the Docker socket** to run migrations; treat that container as privileged and restrict who can reach it.
- Restrict the operator console to a staff VLAN, and put the whole deployment behind the customer's own network controls.

Data protection and lawful use. Facial recognition is sensitive processing under UAE and Saudi data-protection law, and biometric data is classed as sensitive personal data. The architecture supports compliance — in-country processing by construction, customer-owned watchlists, configurable retention, purge capability, and a full audit trail of who searched for whom. The obligations that remain with the customer are the ones only they can hold: a lawful basis for each watchlist entry, a defined retention period, human verification before any action is taken on a match, and periodic review of accuracy in their own operating conditions. We will support a Data Protection Impact Assessment during the pilot and provide the audit exports it requires.

8. Support

Tiered SLA (see Pricing & Packaging): response targets, remote diagnostics through the metrics and health endpoints, spare-glasses ratio, index and threshold tuning reviews, and optional managed enrolment services. Because the deployment is on-premises and may be air-gapped, remote support is delivered through a customer-controlled channel that can be disabled entirely — in which case support is on-site or via customer-mediated log export.