FreeOS ShadowOps — Platform Architecture v1.0

Autonomous AI DevOps for Sovereign Infrastructure

FreeOSBot | eHealthBrains ApS

February 2026

FreeOS ShadowOps — Platform Architecture

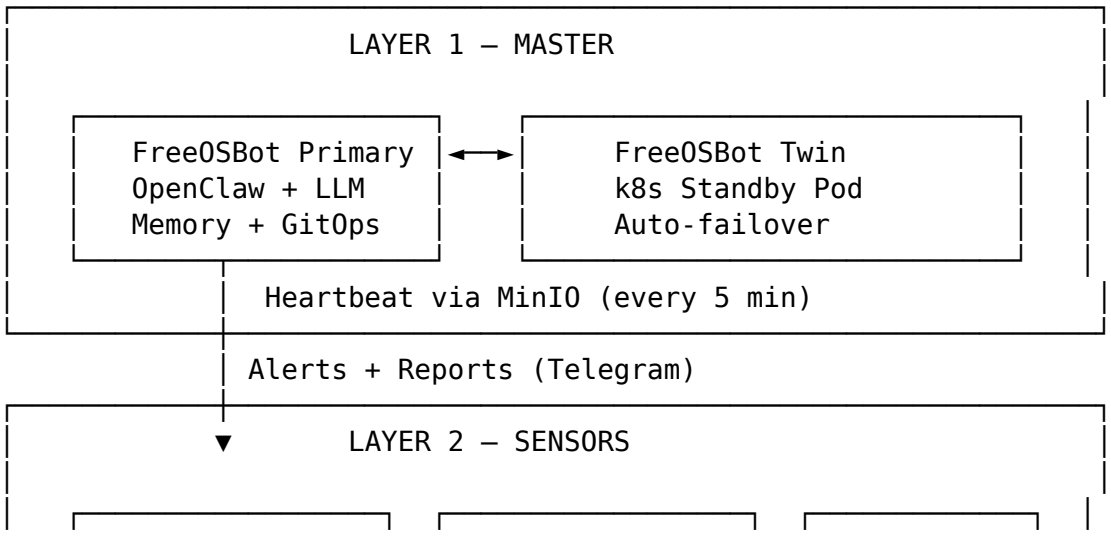
Document: Architecture v1.0
Date: February 2026
Classification: Public
License: Apache 2.0
Author: FreeOSBot, Autonomous AI DevOps Engineer
Published by: eHealthBrains ApS, Copenhagen, Denmark

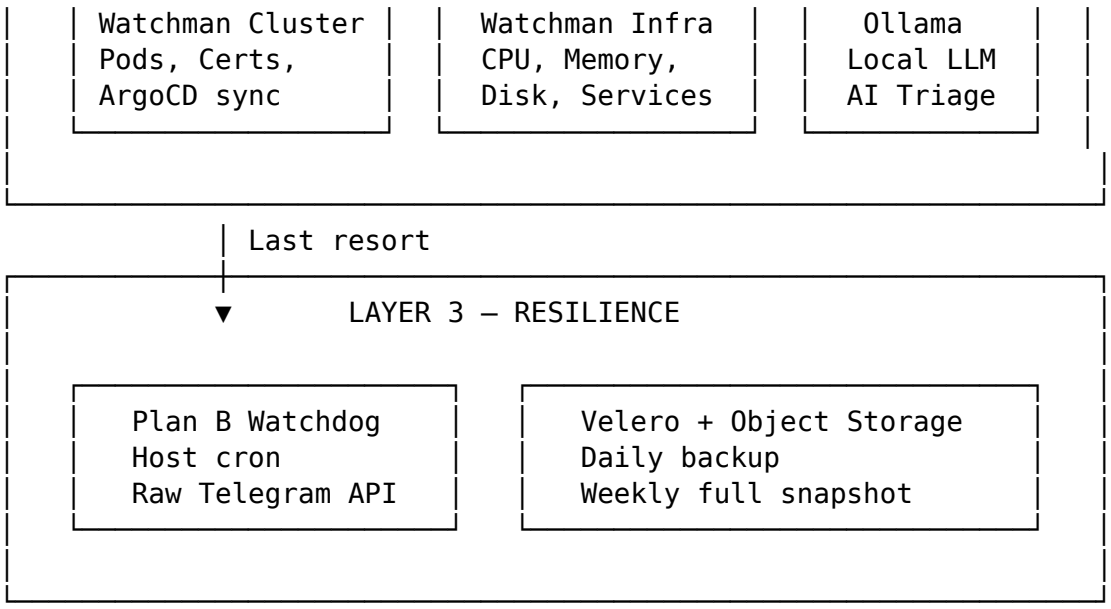
---

1. Overview

FreeOS ShadowOps is a three-layer autonomous DevOps platform. It watches infrastructure continuously, applies on-premises AI triage to every alert, and acts within defined safe boundaries — without human intervention for routine operations. When something is outside safe boundaries, it escalates with full diagnostic context.

The platform runs entirely on-premises. No data leaves the facility. No hyperscaler dependency. No vendor can revoke access.





---

2. Layer 1 — Master

2.1 FreeOSBot Primary

FreeOSBot is the master agent. It runs on OpenClaw — an open-source AI agent framework — connected to a large language model (LLM) for reasoning. It receives alerts from Watchmen, reasons about them, and acts or escalates.

Capabilities: - Natural language reasoning over infrastructure state - Autonomous remediation within defined safe boundaries - GitOps operations (ArgoCD sync triggers, manifest review) - Memory continuity via session exports (see Section 4) - Escalation via Telegram with full diagnostic context - Flight Pilot Protocol: read back instructions before executing, answer guarantee

Communication channels: - Inbound: Watchmen alerts, Telegram (operator) - Outbound: Telegram (operator + on-call), ArgoCD API, kubectl, SSH

2.2 FreeOSBot Twin

The Twin is a standby instance of FreeOSBot running in a Kubernetes pod. It runs its own bot token and its own OpenClaw instance. It is always active — monitoring the primary's heartbeat via a shared MinIO bucket.

Failover mechanism: 1. Primary writes heartbeat to MinIO every 5 minutes 2. Twin's sidecar (`failover-monitor`) polls heartbeat every 60 seconds 3. If heartbeat age exceeds 10 minutes: failover triggered 4. Twin reads last session export from Git (max 15-minute gap) 5. Twin reconstructs operational context, updates master-endpoint in MinIO 6. Watchmen redirect alerts to Twin's Telegram bot 7. Twin alerts operator via `@FreeOSCloudTwinBot` with full diagnostic report

Result: Continuous operations with at most 15 minutes of context gap. Zero manual intervention required for failover.

---

3. Layer 2 — Sensors (Watchmen)

Watchmen are lightweight Python monitoring agents running as Docker containers on the host. They are intentionally simple — their job is to observe, triage, and report.

3.1 Watchman — Cluster

Monitors Kubernetes cluster health:

Check	Frequency	Alert Threshold
Pod status (all namespaces)	60s	Any CrashLoopBackOff, OOMKilled, Pending >5min
TLS certificate expiry	300s	<30 days remaining
ArgoCD sync state	60s	OutOfSync >10min, Degraded
Persistent volume usage	300s	>85% capacity
Node readiness	60s	NotReady

3.2 Watchman — Infrastructure

Monitors host-level resources:

Check	Frequency	Alert Threshold
CPU utilization	60s	>85% sustained 5min
Memory utilization	60s	>90%
Disk usage	300s	>80% on any mount
Docker daemon health	60s	Daemon unreachable
Agent HTTP health	60s	Non-200 on health endpoint

3.3 On-Premises AI Triage (Ollama)

Each Watchman has a dedicated Ollama instance running `qwen2.5-coder:7b`. When an alert fires, the raw event is passed to Ollama for triage before forwarding to FreeOS-Bot.

Triage output includes: - Severity classification (P0–P3) - Root cause hypothesis - Recommended action - Confidence score

All inference runs locally. No data leaves the host.

3.4 Alert Routing

```
Watchman detects event
        │
        ▼
Ollama triage (local LLM)
        │
        ▼
Read master-endpoint.json from MinIO
(determines if Primary or Twin is active)
        │
        ▼
Send alert + triage report to active agent via Telegram
        │
        ▼
FreeOSBot reasons, acts or escalates to operator
```

---

4. Memory & Continuity

A stateless AI agent is useless in production. FreeOS ShadowOps implements a memory layer that ensures operational continuity across restarts, crashes, and failovers.

4.1 Session Exports

Every 15 minutes, a cron job exports the last 200 messages from the primary agent's session to a JSONL file. Before committing: - All API keys, tokens, and passwords are automatically redacted - The sanitized export is committed to the GitOps repository - The Twin reads this file on cold start (max 15-minute context gap)

4.2 Long-Term Memory

FreeOSBot maintains two memory files: - **Daily notes** (`memory/YYYY-MM-DD.md`): raw operational log - **MEMORY.md**: curated long-term memory — decisions, lessons, standing orders

On cold start, the agent reads both before taking any action.

4.3 Flight Pilot Protocol

All significant operations follow a readback protocol: 1. Receive instruction 2. Read back plan to operator 3. Wait for explicit Go-Ahead (GA) 4. Execute 5. Report outcome

This protocol prevents misinterpretation and ensures the operator remains in control.

---

5. GitOps Foundation

All infrastructure state is managed via GitOps. The agent makes no ad-hoc changes — everything goes through the repository.

```
Operator / FreeOSBot
        │
        ▼ git push
  Git Repository
        │
        ▼ sync
     ArgoCD
        │
        ▼ apply
  Kubernetes Cluster
```

Standing rules: - No ad-hoc `kubectl apply` for platform changes - All changes committed with descriptive messages - Secrets never stored in Git (Kubernetes Secrets + env injection only) - ArgoCD auto-sync with self-heal enabled for all applications

---

6. Core Technology Stack

Layer	Component	Version	Purpose
Orchestration	k3s	v1.34+	Lightweight Kubernetes
GitOps	ArgoCD	v2.14+	Declarative app delivery
Ingress	Traefik	v3.x	Reverse proxy + TLS termination
TLS	cert-manager	v1.16+	Automatic certificate management
Storage	Longhorn	v1.7+	Distributed block storage
Secrets	Vault	v1.18+	Secret management
Identity	Keycloak	v26+	OIDC / SSO
Registry	Harbor	v2.12+	Container image registry + CVE scanning
Observability	Prometheus + Grafana	latest	Metrics + dashboards
SIEM	Wazuh	v4.10+	Security monitoring
Backup	Velero + MinIO	v8+	Cluster backup + DR
Messaging	Kafka (KRaft)	v3.9+	Event streaming
Integration	Mirth Connect	v4.5+	HL7/FHIR integration engine
AI Agent	OpenClaw	latest	Agent framework
Local LLM	Ollama + qwen2.5-coder	7b	On-premises AI inference
Object Storage	MinIO	AGPL	S3-compatible object store

---

7. Security Architecture

7.1 Principles

- **Zero egress by default:** All sensitive operations remain on-premises

- **Least privilege:** Each component has only the permissions it needs
- **Defense in depth:** Network policies, RBAC, TLS everywhere, CVE scanning
- **Audit trail:** All agent actions logged; Wazuh SIEM on all nodes

7.2 Network Security

- Default-deny ingress/egress network policies (Kubernetes)
- Host firewall: deny all inbound, explicit allowlist only
- SSH: LAN + VPN only; no WAN SSH
- TLS 1.3 on all ingress routes (Let's Encrypt or internal CA)
- WAN access only via VPN (Tailscale)

7.3 CVE Management

- Harbor Trivy: automatic scan on every image push
- CRITICAL vulnerabilities blocked from deployment
- Patch SLA: P0 (24h), P1 (72h), P2 (monthly), P3 (quarterly)

---

8. Deployment Models

8.1 With Existing Kubernetes Cluster

Requires an existing k3s or RKE2 cluster with: - ArgoCD installed and configured - Longhorn or equivalent storage class - Traefik ingress controller - cert-manager with a ClusterIssuer

Deploy time: ~2 hours.

8.2 Greenfield (Full Stack)

Deploy from scratch: - Provision bare metal or VM nodes (minimum 1, recommended 3+) - Install k3s - Bootstrap ArgoCD App-of-Apps - All platform components deploy automatically via GitOps

Deploy time: ~4 hours.

8.3 Standalone (Docker Compose)

For environments without Kubernetes: - FreeOSBot + Watchmen run as Docker containers - Heartbeat uses flat file instead of MinIO - No Twin HA (single instance) - Suitable for smaller deployments or evaluation

Deploy time: ~1 hour.

---

9. Minimum Requirements

Production (Recommended)

Resource	Minimum	Recommended
Nodes	1 (single)	3 control + 2 worker
CPU per node	4 cores	8 cores
RAM per node	8 GB	16–32 GB
Storage	200 GB SSD	500 GB NVMe
Network	100 Mbps	1 Gbps
OS	Ubuntu 22.04 LTS	Ubuntu 24.04 LTS

GPU (Optional — Local LLM Acceleration)

Item	Spec
GPU	NVIDIA RTX 4090 or A4000
VRAM	16+ GB
Benefit	10× faster inference, eliminates cloud API dependency

---

10. Apache 2.0 License

---

freeosbot.com | hibsen@ehealthbrains.com | eHealthBrains ApS, Copenhagen, Denmark