Sentinel-Ops — Full Operational & Architectural Report

Hybrid Edition: Historical Overview + Detailed Log of Today's Work

Generated: 14 February 2026

---

1. Executive Summary

This document provides a complete, end-to-end overview of the Sentinel-Ops project as of 14 February 2026, including:

- Repository architecture
- RAG ingestion pipeline
- IaC structure
- YAML + Python components
- Windows ↔ Ubuntu DevOps VM ↔ GitHub linkage
- Today's operational progress
- Debugging steps
- Status index
- Next actions

This report is designed as a **canonical reference** for both human operators and the Sentinel-Ops RAG engine.

---

2. Project Context & Objectives

Sentinel-Ops is a multi-LLM, anti-drift, version-controlled operational knowledge system designed to:

- Maintain a curated **canon** of authoritative documents
- Provide reproducible **infrastructure-as-code**
- Support **RAG-based reasoning** across the entire project
- Ensure continuity across Windows, Ubuntu DevOps VM (VM 100), and GitHub
- Enable long-term operational resilience

The system is built around:

- **GitHub** as the source of truth
- **Windows 11** as the primary workstation
- **Ubuntu DevOps VM 100** as the execution environment

- **Ollama** for local LLM inference
- **ChromaDB** for vector storage
- **Python** for ingestion + query pipelines

---

3. Repository Architecture (Current)

```
sentinel-ops/
│
├── canon/                      # Curated, authoritative documents
│   ├── Architecture.md
│   ├── Infrastructure-as-Code (IaC) & Security Baseline Plan.md
│   ├── Roadmaps/
│   └── Policies/
│
├── infra/                      # Infrastructure-as-Code
│   ├── terraform/
│   ├── kubernetes/
│   ├── ansible/
│   └── artifacts/
│       └── sboms/
│           └── host-ubuntu-24.04.sbom.json (50MB)
│
├── reports/                    # Generated reports (PDF-ready MD)
│
├── rag/                        # RAG engine
│   ├── ingest.py
│   ├── query.py
│   ├── config.yaml
│   └── vectorstore/
│
└── README.md
```

---

4. Cross-System Mapping

4.1 Windows File Structure

Primary working directory:

`C:\Users\Admin\Documents\sentinel-ops\`

RAG engine:

`C:\Users\Admin\Documents\sentinel-ops\rag\`

Ollama models stored under:

`C:\Users\Admin\.ollama\models\`

Python environment:

```
C:\Users\Admin\AppData\Local\Programs\Python\Python312\
```

4.2 Ubuntu DevOps VM (VM 100)

Mounted repo location:

```
/home/andy/sentinel-ops/
```

Used for:

- Terraform execution
- Kubernetes manifests
- CI/CD testing
- SBOM generation
- DevOps automation

4.3 GitHub Integration

Remote origin:

```
https://github.com/<your-org>/sentinel-ops.git
```

Windows ↔ GitHub:

- Git CLI
- VS Code
- GitHub Desktop

Ubuntu VM ↔ GitHub:

- SSH keys stored under `/home/andy/.ssh/`
- Git operations executed inside VM 100

---

5. IaC Overview

5.1 Terraform

Located under:

```
infra/terraform/
```

Typical structure:

```
main.tf
variables.tf
outputs.tf
providers.tf
modules/
```

Used for:

- VM provisioning
- Network configuration
- Storage
- Security baselines

5.2 Kubernetes Manifests

Located under:

`infra/kubernetes/`

Includes:

- Deployments
- Services
- Ingress
- ConfigMaps
- Secrets (referenced, not stored)

5.3 Ansible

Located under:

`infra/ansible/`

Used for:

- Host configuration
- Package installation
- Hardening
- SBOM generation tasks

---

6. RAG Engine Components

6.1 config.yaml

Example structure:

```
include_paths:
  - "../canon"
  - "../infra"
  - "../reports"

model: "gemma3:4b"

vectorstore_path: "./vectorstore"

chunk_size: 500
chunk_overlap: 50
```

6.2 ingest.py (Corrected Version)

Key responsibilities:

- Walk repo
- Read files
- Chunk text
- Embed chunks using `nomic-embed-text`
- Store vectors in ChromaDB

Important notes:

- SBOMs are extremely large
- Chunking prevents context overflow
- Embedding is CPU-bound
- Ingestion is linear and deterministic

6.3 query.py

Responsibilities:

- Accept user question
- Retrieve relevant chunks
- Build prompt
- Send to Gemma
- Return answer

Gemma is used **only** for generation, not embeddings.

7. Today’s Operational Log (14 Feb 2026)

7.1 Tasks Completed

Index	Task	Status	Notes
1	Validated repo structure	✅	canon/, infra/, rag/ confirmed
2	Installed ChromaDB	✅	Python environment stable
3	Installed Ollama models	✅	gemma3:4b + nomic-embed-text
4	Debugged ingestion errors	✅	Context-length errors resolved
5	Implemented safe chunking	✅	Prevents infinite loops
6	Rewrote ingest.py	✅	Stable, deterministic
7	Identified SBOM bottleneck	✅	50MB file causing long runtime
8	Confirmed ingestion progress	✅	Verified via Task Manager
9	Produced canonical report	✅	This document

7.2 Tasks In Progress

Task	Status	Notes
Ingestion of SBOM	⏳	Very large file; CPU-bound
Full vectorstore build	⏳	Continues after SBOM
RAG validation	⏳	Pending ingestion completion

7.3 Tasks Blocked

Task	Blocked By	Notes
Query testing	Ingestion	Must complete vectorstore first
RAG tuning	Ingestion	Requires full corpus

---

8. Performance Observations

Task Manager shows:

- Ollama ~40% CPU
- Python ~10% CPU
- Disk I/O active
- Memory stable

This is the expected signature of:

- active chunking
- active embedding
- active vectorstore writes

No signs of:

- infinite loops
- memory leaks
- deadlocks

---

9. Recommendations

9.1 Exclude SBOMs from RAG

SBOMs:

- add noise
- slow ingestion
- degrade retrieval
- provide no semantic value

Recommendation:

```
if file.endswith(".sbom.json"):
    continue
```

9.2 Add File Size Guard

```
if os.path.getsize(full_path) > 5_000_000:
    continue
```

9.3 Add Logging

- per-file
- per-chunk
- timing

---

10. Next Steps

1. Allow ingestion to finish or skip SBOMs

2. Run `query.py`
3. Validate retrieval quality
4. Tune chunk size if needed
5. Add ingestion logs
6. Add RAG evaluation prompts
7. Commit this report to `reports/`

---

11. Appendix A — Full Repo Tree Snapshot

(Generated from your description and validated structure)

```
sentinel-ops/
├── canon/
│   ├── Architecture.md
│   ├── Infrastructure-as-Code (IaC) & Security Baseline Plan.md
│   └── ...
├── infra/
│   ├── terraform/
│   ├── kubernetes/
│   ├── ansible/
│   └── artifacts/
│       └── sboms/
│           └── host-ubuntu-24.04.sbom.json
├── rag/
│   ├── ingest.py
│   ├── query.py
│   ├── config.yaml
│   └── vectorstore/
└── reports/
```

---

12. Appendix B — RAG Pipeline Diagram (Text)

```
[canon/] ----\
[infra/] -----+--> ingest.py --> embeddings --> ChromaDB --> query.py --> Gemma -->
Answer
[reports/] --/
```

---

13. Closing Summary

This document now serves as:

- a canonical reference
- a RAG-ingestible artifact
- a PDF-ready operational report

- a cross-system architectural map
- a complete log of today’s work

It captures the full historical context **and** the detailed operational progress of 14 February 2026.

If you want, I can also generate a **short executive summary version** or a **diagram-focused version** to accompany this main report.