Administrator Guide

Deploying, configuring and integrating Synapse-Vanguard

Audience: System Administrators (includes API setup)

Version 0.3.0

Self-hosted, data-sovereign Security Information & Event Management

1. Introduction & architecture

Synapse-Vanguard is a self-hosted, data-sovereign SIEM. For the embedded (homelab / SMB) deployment it runs as a single consolidated process that serves the web UI, the query API, the agent-ingest API and the outbound webhook / alert engine, with a normalizer running as a background task on one shared database.

Component	Role
Web UI + API	Dashboard and control plane (port 5500).
Agent ingest	Receives compressed logs from host agents.
Normalizer	Parses, enriches, applies alert rules, stores events.
DuckDB	Embedded columnar store (single-writer).
Redis	Durable queue between ingest and the normalizer.

NOTE DuckDB is single-writer, so the default deployment is one process. Running the pieces as separate containers against the same database file will crash-loop. Fanning out into microservices is only valid with an external shared database.

2. Deployment

Prerequisites: Docker + Docker Compose on the host.

2.1 Configure secrets

Compose reads a `.env` file in the project directory. Generate strong values (do not commit this file):

```
SV_VAULT_KEY=<64 hex chars>          # openssl rand -hex 32
SV_JWT_SECRET=<long random string>
SV_ENROLLMENT_TOKEN=<random token>    # agents use this to enrol
SV_BOOTSTRAP_ADMIN_USER=admin
SV_BOOTSTRAP_ADMIN_PASSWORD=<choose a strong password>
SV_MTLS_ENABLED=false                 # true only for the hardened split ingest
```

2.2 Start the stack

```
cd synapse-vanguard-v3
docker compose up -d --build
# open http://<host>:5500/
```

2.3 First login

On first start an administrator account is seeded from the two BOOTSTRAP variables. Sign in, then change the password in **Admin > User management**.

2.4 Ports

Port	Purpose
5500	Web UI, query API, agent ingest, external ingress API
(internal) 6379	Redis - not exposed to the host

WARNING Persistence: the enrolled-agent registry and all events live in a Docker volume. Use plain "docker compose down" / "up". Never "down -v" unless you intend to wipe everything - it deletes the volume and de-enrols every agent.

3. The Admin menu

The **Admin** menu (top-right, visible only to administrators) contains:

- **User management** - create/disable users, set roles, reset passwords.
- **Agent management** - view the enrolment token and revoke agents.
- **Alert rules (keywords)** - named keyword-driven Alerts.
- **Ingress API (IronNode)** - the key + URL for external apps to push events.
- **Outbound webhooks** - endpoints Vanguard pushes high-severity events to.
- **Ingestion / data sources** - throughput and per-source stats.
- **System settings** - version, backend, secret status, counts.

Analysts and Admins also see a **Tuning** button on the toolbar for false-positive suppressions (section 7).

4. User management

Add a user with a username, password (min 8 chars) and role. Roles map to capabilities: Read-Only Auditor, Security Analyst, System Administrator. Per row you can change the role, reset the password, or enable/disable the account.

> **SAFEGUARDS** You cannot disable your own account, change your own role, or remove the last remaining administrator - this prevents lock-out.

5. Agents: deployment & management

5.1 Enrolment token

Agents authenticate their first contact with the shared enrolment token (`SV_ENROLLMENT_TOKEN`), visible under **Admin > Agent management > Enrolment**. After enrolling, each agent is issued its own per-agent ingest token automatically.

5.2 Install a Linux agent (one-shot)

Copy the `agents/` directory to the host, then:

```
sudo bash agents/linux/install-linux-agent.sh \
  --server <VANGUARD-HOST> --token <SV_ENROLLMENT_TOKEN>
```

The installer creates an isolated virtualenv, writes `/etc/sv-agent/agent.config.toml`, installs a systemd service, and (unless `--no-audit`) loads a high-signal auditd ruleset. It auto-detects Debian vs RHEL log paths. Re-run to update; `--uninstall` to remove.

Agents collect journald, syslog and auditd on Linux; on Windows they collect the Security, System, Application, Sysmon, PowerShell, Defender, Task Scheduler, WMI and Terminal Services channels. (Windows agent: run `python -m agents.windows.sv_agent_windows` with `SV_AGENT_CONFIG` set, or package it as a service.)

5.3 Revoke / uninstall

- **Revoke** (server side): Admin > Agent management > Revoke. The agent is no longer accepted and must re-enrol to return.
- **Uninstall** (host side): `sudo bash agents/linux/install-linux-agent.sh --uninstall`. Add `sudo rm -rf /var/lib/sv-agent` for a full wipe.

5.4 Troubleshooting: agent gets 401 on /v1/ingest/bulk

This means the server does not recognise the agent's stored token - usually because the database was reset since the agent enrolled, or the agent kept a stale identity from a previous install. Force a re-enrolment:

```
sudo systemctl stop sv-agent
sudo rm -f /var/lib/sv-agent/state.json   # drop stale identity (keeps buffer)
sudo systemctl start sv-agent
sudo journalctl -u sv-agent -f            # expect: enrolled as agent_id=...
```

6. Alert rules (keyword detection)

Under **Admin > Alert rules**, define named Alerts that watch event messages for keyword(s). On a match, Vanguard re-tags the event as that named Alert at your chosen severity. Each rule has:

Field	Meaning
Alert name	The Alert type shown on matched events (e.g. Ransomware).
Keywords	One or more comma-separated substrings.
Match	"any" (any keyword) or "all" (all keywords must appear).
Severity	The severity assigned on match (default Alert).
Fire webhook	Optional: a webhook to trigger directly on match.

Rules can be enabled/disabled and deleted by selection. The "Hits" column counts matches. When a rule has a target webhook, a match fires that webhook directly (regardless of the webhook's own severity threshold) - a keyword can therefore trigger an IronNode response playbook.

7. False-positive tuning (suppressions)

Analysts and Admins can suppress benign events so they stop triggering. A suppressed event is still logged but is not escalated (no severity bump, no Alert tag, and no webhook fires by either the alert path or the severity-threshold path). Create suppressions from the bell icon on an alerted row, or from the **Tuning** toolbar button. Match on any combination of alert type, host, source type and message text; all set conditions must match. Toggle off or delete to resume alerting.

8. Ingress API (receiving events from IronNode)

To let an external Synapse app (such as Synapse IronNode) push events, generate an ingress API key under **Admin > Ingress API (IronNode)**. The panel shows the ingest URL and the key for copy-out, an enable/disable toggle, and a Regenerate button (regenerating immediately invalidates the old key). Full API details are in section 11.

9. Outbound webhooks (sending to IronNode)

Under **Admin > Outbound webhooks**, add endpoints Vanguard should POST qualifying events to - typically an IronNode tenant webhook that triggers a response playbook. Each webhook has a target URL, an optional HMAC signing secret, and a minimum severity threshold (fires when an event is at or above it). Use **Test** to verify delivery, the toggle to enable/disable, and selection + Delete to remove. The signed-payload contract is in section 11.

10. Ingestion & system settings

- **Ingestion / data sources** - live events/sec, events grouped by source type, and active agents with their channels.
- **System settings** - version, storage backend, broker, whether the vault key and JWT secret are configured, and record counts. Secrets are never displayed.

11. API setup

Vanguard exposes three programmatic paths on port 5500. This chapter is the reference for integrating your own systems and for the Synapse IronNode connection.

Path	Direction	Auth
/v1/ingest/bulk	Agents push logs in	Per-agent token (via enrolment)
/v1/ingest/events	External apps push events in	Ingress API key (Bearer)
Outbound webhooks	Vanguard pushes events out	Optional HMAC signature

11.1 Agent ingest API

Handled automatically by the agent installer - documented here for reference. The agent config points at the base URL; the agent appends the paths itself.

```
ingest_url = "http://<VANGUARD-HOST>:5500"
```

- `POST /v1/agents/register` - one-time enrolment. Header `Authorization: Bearer <enrolment token>`. Returns an agent id and a per-agent ingest token.
- `POST /v1/ingest/bulk` - zstd-compressed NDJSON log batches, authenticated by the per-agent token (or client certificate when mTLS is enabled).

11.2 Ingress events API (recommended for integrations)

Any external system can push events with an ingress API key generated under **Admin > Ingress API**.

Request

```
POST http://<VANGUARD-HOST>:5500/v1/ingest/events
Authorization: Bearer <svg_... ingress key>
Content-Type: application/json

{
  "events": [
    {
      "ts": "2026-07-03T10:00:00Z",
      "host": "app-01",
      "message": "Failed login for user bob",
      "severity": 2,
      "category": "authentication",
      "mitre_technique": "T1110",
      "fields": { "src_ip": "10.0.0.9" }
    }
  ]
}
```

Response

```
{ "accepted": 1 }
```

- **severity** is the syslog scale 0-7 (0 = emergency). Omit to default to INFO.
- Only **message** is required; put any extra keys under **fields**.
- An empty body `{"events": []}` is a valid connection probe - returns 200 with a good key, 401 with a bad or disabled key.
- Maximum 5000 events per request.

Example

```
curl -X POST http://<host>:5500/v1/ingest/events \
  -H "Authorization: Bearer $SVG_KEY" -H "Content-Type: application/json" \
  -d '{"events":[{"host":"app-01","message":"test","severity":5}]}'
```

11.3 Outbound webhook payload

When a webhook fires, Vanguard sends a POST like this to the target URL:

```
POST <your target URL>
Content-Type: application/json
X-SV-Event: security_event
X-SV-Signature: sha256=<hex>        # only when a secret is set

{
  "source": "synapse-vanguard",
  "kind": "security_event",
  "event": {
    "ts": "...", "host": "...", "source_type": "...",
    "severity": 2, "category": "alert", "action": "Ransomware",
    "mitre_technique": "T1486", "message": "..."
  }
}
```

Verifying the signature

If you set a signing secret, verify each delivery: compute HMAC-SHA256 of the raw request body using the secret and compare to the `X-SV-Signature` header (constant-time). Reject on mismatch.

> **NOTE** For an IronNode target, point the webhook at that tenant's webhook URL (/api/webhook/<tenant>/<path>); the payload above becomes the playbook trigger. Alert rules can also target a webhook directly, so a keyword Alert triggers a playbook instantly.

11.4 Connecting Synapse IronNode (both directions)

A. IronNode -> Vanguard (IronNode incidents appear as events):

- In Vanguard: **Admin > Ingress API** - Generate key; copy the URL and key.
- In IronNode: **Settings > Synapse-Vanguard SIEM forwarder** - paste the URL and key, click **Test connection**, then enable. Incidents forward automatically across their lifecycle (triggered, triage, closed).

B. Vanguard -> IronNode (Vanguard alerts trigger playbooks):

- In IronNode: copy the tenant webhook URL from its Settings.
- In Vanguard: **Admin > Outbound webhooks** - add it (with a severity threshold and optional secret), or attach it to an Alert rule to fire on a specific keyword. Use **Test** to confirm delivery.

> **KEY PERSISTENCE** The API key is stored (encrypted) in both applications and persists until you regenerate it in Vanguard and paste the new value into IronNode.

12. Security notes

- Keep `.env` secrets out of version control; rotate the ingress key and enrolment token periodically.
- Access is role-based; give users the least privilege they need. Auditors are strictly read-only.
- For production agent traffic, enable mTLS: terminate `/v1/ingest/*` on a dedicated TLS port with client certificates (set `SV_MTLS_ENABLED=true`).

- All processing is local - event data never leaves your infrastructure. Only the opt-in outbound webhooks send data out, to endpoints you configure.