Chapter 8 — Secrets Management

8.1 Current State and Scope

As of 2026-07-18 **no secrets manager is deployed**. Credential state is the Chapter 2 CRITICAL nonconformance: plaintext passwords in the operational changelog, a shared fleet-wide root password, and service identities on non-SBS mailboxes (Build Register item 15). This chapter defines the target system that item 15 migrates into, covering: platform secrets (API tokens, DB credentials, Spaces keys, internal-CA material, webhook URLs), per-Compartment agent credentials (AI-03), and deploy-time injection into containers (§6.4). Out of scope: end-user passwords (IdP domain, Chapter 9) and TLS private keys for public surfaces (deliberately host-local, §5.2).

8.2 Vault Deployment

HashiCorp Vault (OSS) on a dedicated minimal droplet:

Attribute	Value
Droplet	`prod-vault` — 1 GB / 1 vCPU, NYC2, member of `prod-vpc-nyc2` (created in the §3.1 rebuild wave)
Storage backend	Integrated Raft, single node, on a dedicated DO Volume
Listener	VPC-private interface **only**, TLS from the internal CA (§7.5) — no public interface listener at all; Vault has no DNS name in the public zone
Access paths	API from prod/dev droplets over VPC; human access via bastion (§3.3) with individually attributed tokens
Firewall	Cloud Firewall `prod-vault`: TCP 8200 from `sbsdash-server-prod`, `Dev-SBS-server` (for dev-namespace reads), and bastion private IPs only; SSH via bastion only

Placement rationale: not on `sbsdash-server-prod` (the secrets store must survive and be trustworthy independent of the host running the workloads that consume it — a prod-droplet compromise should not equal immediate Vault-storage compromise), and not on `monitor-servers` (bastion + alerting authority already concentrates enough; adding secrets makes it a single point of total compromise). A 1 GB droplet is sufficient — Vault's footprint at this scale is trivial.

Unseal model: DigitalOcean offers no KMS, so auto-unseal via cloud KMS is unavailable. **Shamir key shares: 5 shares, threshold 3**, held by named individuals (Ledger Hub platform engineering

×2, Nelson Santos, Alexandra Del Rey, one SBS-designated alternate) — no individual can unseal alone; any 3 can recover. Shares are stored in each holder's personal password manager, never together, never in the changelog or repo. Unseal is required at Vault restart only; restarts are rare, scheduled events. Root token is revoked after initial setup (`vault operator generate-root` recreates one under quorum if ever needed) — routine administration uses named admin tokens under policy.

Availability posture, stated honestly: single-node Vault means Vault downtime blocks *deploys and credential rotation*, not running workloads (containers hold their injected credentials until restart, §8.5). Accepted at this fleet size; mitigation is Raft snapshot every 6 h to `sbs-dash-backups` (encrypted client-side per §7.5), restore-tested quarterly (Ch. 14). HA (3-node Raft) is a Chapter 16 growth trigger, not a Day-1 requirement.

8.3 Namespace and Policy Layout

Vault OSS lacks Enterprise namespaces; the equivalent is **mount-path + policy separation**, which is sufficient because policies are default-deny:

```
secret/platform/        # DO API tokens, GitHub deploy keys, Spaces keys,
                        # registrar/DNS API tokens, alerting webhooks
secret/infra/ca/        # internal CA key material (§7.5)
secret/db/platform/     # migration-tool superuser credentials (§7.4)
secret/c01/ … secret/c10/   # per-Compartment: DB role, Redis ACL user,
                            # Spaces prefix keys, Anthropic API key,
                            # per-Compartment integration credentials
secret/dev/…            # dev-environment mirror, synthetic/dev-scoped values only
```

Policies, one per consumer class: `policy-c01` grants read on `secret/c01/*` and **nothing else** — the AI-03 guarantee at the secrets layer, matching the network (§6.4) and engine-ACL (§7.4) walls: Compartment 03's runtime identity cannot read `secret/c08/*` even if every other control fails. `policy-deploy` reads what the deploy pipeline renders; `policy-admin` is held by named humans; no policy grants `secret/*` wildcard. Dev identities can read `secret/dev/*` only — prod paths are unreachable from Dev (the promotion path carries templates, not values, §2.2.3).

8.4 Secret Lifecycle and Rotation Policy

Class	Examples	Rotation	Mechanism
Static, high-blast-radius	DO API token, GitHub deploy key, registrar/DNS API token, Spaces keys	90 days, and immediately on personnel change or suspected exposure	Manual rotation runbook (wiki), tracked as recurring Build Register/ops-calendar item; each rotation logged

Class	Examples	Rotation	Mechanism
Database credentials	Per-Compartment PostgreSQL / MongoDB / Redis users	30 days target	**Vault database secrets engine** (dynamic short-TTL credentials) once data services are up — Vault creates per-lease users, expiry is automatic, rotation ceases to be a human task. Until then: static creds at 90 days
Anthropic API keys	Per-Compartment keys	90 days; immediately on anomalous-usage alert (MON-02 / §3.4 egress signals)	Manual via Anthropic console, stored per-Compartment path
Internal CA leaf certs	DB TLS (§7.5)	90-day leaf TTL, auto-renewed by IaC	Vault PKI engine issues leaves; CA root offline in `secret/infra/ca/`
Unseal shares	—	On any holder change	Rekey ceremony (`vault operator rekey`, quorum)

Rules: every secret has a named owner and a recorded rotation date (KV metadata); **KV v2 versioning** on all mounts so rotation keeps the prior version recoverable for rollback; no secret is ever emailed, chat-pasted, or written to the changelog — the changelog records *that* a rotation happened, never the value. Audit device (file → shipped to Ch. 12 pipeline) logs every read/write with the requesting identity: secret access is itself a MON-01 event stream, and reads outside deploy windows or from unexpected identities are MON-02 anomaly inputs.

8.5 Application Integration

Injection model (carried from §6.4), chosen for minimal moving parts:

1. **AppRole per consumer** — one AppRole per Compartment (`role-c01` … bound to `policy-c01`) and one for the deploy pipeline. RoleID is IaC-managed config; SecretID is delivered response-wrapped at deploy time with short TTL and use-limit 1 — a leaked wrapped token is worthless after first use or expiry.
2. **Deploy-time render** — the deploy step authenticates via AppRole, reads the Compartment's paths, renders the env-file consumed by `docker compose` (§6.7), sets mode 0600, and **deletes the env-file after container start** (compose has already captured the environment). Values never enter images, compose files, the repo, or CI logs (INF-02, INF-03).
3. **Refresh** — containers hold credentials until redeploy. With dynamic DB credentials (§8.4), lease TTLs are set ≥ the deploy cadence so normal deploys renew credentials as a side effect; an emergency revocation is: revoke lease in Vault → redeploy Compartment (minutes, and only the affected Compartment restarts — per-Compartment stacks pay off here).
4. **Prohibited patterns** — no Vault agent sidecars for now (added complexity before it's needed), no secrets in `docker inspect`-visible plain `environment:` keys in committed files,

no shared "platform god token" consumed by all Compartments.

Bootstrap note (the classic first-secret problem): the deploy host must hold one credential to start — the RoleID (low sensitivity, useless without SecretID) on disk, and SecretID issuance gated by the CI/operator identity. This is the accepted trust anchor and is documented in the wiki runbook rather than hidden.

8.6 Migration of Existing Credentials (Item 15 Execution Order)

1. Deploy `prod-vault`, initialize, distribute shares, revoke root token.
2. **Rotate** the four exposed credential sets (GitHub, droplet root password → abolished entirely in favor of keys per §2.5, wiki admin, Grafana admin) — rotate *then* store; never vault a burned value.
3. Write rotated values + all new secrets created since (DO tokens, Spaces keys as provisioned) into their §8.3 paths.
4. Strip the changelog credentials section; purge from any git history (`git filter-repo`) if the file was ever committed; add pre-commit secret-scanning (gitleaks) to the repo (Ch. 10/11) so re-introduction is blocked mechanically, not procedurally.
5. Re-point service identities to SBS-controlled aliases (§4.4 note) as each is touched.

8.7 Verification

Gates (LH-SBS-INST-001): Vault reachable from prod over VPC TLS, unreachable from any public interface (external scan); policy probe — `role-c03` token reads `secret/c03/*`, denied on `secret/c08/*` and `secret/platform/*`; response-wrapped SecretID single-use confirmed (second unwrap fails); env-file absent post-deploy (filesystem check); no secret string present in repo, images, compose files, or CI logs (gitleaks sweep); audit device shipping to observability pipeline; Raft snapshot present in `sbs-dash-backups`, encrypted, restore-tested; changelog contains no credential values (item 15 closure evidence); rekey ceremony documented with current holder list.

Revision #1
Created 2026-07-18 11:40:36 UTC by SBS Admin
Updated 2026-07-18 11:41:35 UTC by SBS Admin