Storage & Data Layer

Managed databases, object storage, volume encryption, data isolation boundaries

- Chapter 7 — Storage & Data Layer

Chapter 7 — Storage & Data Layer

7.1 Data Layer Overview

Three engines serve the Platform, each with a distinct role; nothing else is introduced without a Change Order:

Engine	Role	Persistence class
PostgreSQL	System of record — Compartment relational data, platform configuration, decision metadata	Durable, backed up, point-in-time recovery target
MongoDB	Document workloads — agent artifacts, semi-structured Compartment content	Durable, backed up
Redis	Cache, queues, session and rate-limit state	Ephemeral by default; only explicitly declared keyspaces persisted (AOF), never treated as a system of record

As of 2026-07-18 **none of the three is installed** — this chapter is fully normative, with the §6.5 interim placement (containers on `sbsdash-server-prod`) as the build path and managed services as the preferred end state.

7.2 Managed vs. Self-Hosted Decision

- **Preferred:** DigitalOcean Managed Databases for PostgreSQL, MongoDB, and Redis (Valkey) — provider-handled failover, automated backups, patch management, and TLS-enforced connections reduce the operational load on a small team.
- **Blocking check (Build Register, carried from §2.3):** Managed Database availability in **NYC2** is unconfirmed — newer DO capacity often lands in NYC1/NYC3 first. The check is: can all three engines be provisioned in NYC2 **attached to the target Prod VPC** (`prod-vpc-nyc2`, §3.1)? Cross-region managed databases are disqualified — data would transit public networking, violating INF-01, the same class of defect as the wiki scrape (§3.3).
- **Fallback (interim, buildable now):** self-hosted containers per §6.5. The remainder of this chapter is written engine-by-engine to be valid under **either** deployment, with deltas flagged.
- **Migration posture:** if the build starts self-hosted and Managed becomes available, migration is dump/restore per engine during a maintenance window, amended by Change Order. The isolation model (§7.4) is deployment-invariant, so no application change is required — only connection endpoints (Vault-managed, Ch. 8).

7.3 Object Storage

DigitalOcean Spaces (S3-compatible), region NYC3 (Spaces is not offered in NYC2; object storage over TLS to a neighboring region is acceptable — unlike database traffic, it is not latency- or VPC-bound):

Bucket	Content	Access
`sbs-dash-artifacts`	Compartment file artifacts (agent outputs, uploads)	Per-Compartment prefix + per-Compartment scoped keys (§7.4)
`sbs-dash-audit`	Hash-chained audit log archives (Ch. 13)	Write-once pattern: writer key has PUT-only; no key held by the platform can DELETE (deletion requires the account control plane)
`sbs-dash-backups`	Database dumps + config archives (Ch. 14)	Backup identity only; replicated to SFO3 per §2.3

Rules: all buckets private (no public-read ever — FE family applies to storage too); access exclusively via scoped Spaces keys stored in Vault, one keypair per purpose, never a shared account-wide key (INF-02); TLS enforced on every operation; bucket inventory and lifecycle policies defined in IaC (Ch. 10). CDN feature on Spaces stays **off** — these are private buckets, and enabling the CDN endpoint would create an unauthenticated public URL surface.

7.4 Data Isolation Boundaries (per-Compartment)

The data and storage slices of the eight-layer model (§1.2), stated per engine. The principle throughout: **isolation is enforced by the engine's own privilege system, not by application discipline** — a compromised Compartment 03 credential must be structurally unable to read Compartment 08 data.

- **PostgreSQL** — one **database per Compartment** (`c01_hr` … `c10_pr`) in the cluster; one **role per Compartment** with `CONNECT` on its own database only; `PUBLIC` connect revoked on every database; no cross-database query paths (no `postgres_fdw`, no `dblink` extensions installed); platform-shared configuration lives in a separate `platform` database with its own role. Superuser credentials exist only in Vault, used only by migration tooling.
- **MongoDB** — one **database per Compartment**; per-Compartment user with `readWrite` scoped to its own database; authentication mandatory (`--auth`), authorization enabled from first boot — never an open dev instance that later gets locked down.
- **Redis** — **Redis ACLs** (v6+) with one ACL user per Compartment, key-pattern-restricted (`~c01:*`), dangerous commands (`FLUSHALL`, `CONFIG`, `KEYS`, `SCRIPT`) denied to all Compartment users; `default` user disabled. If workload interference ever matters more than memory efficiency, escalation path is numbered logical databases or separate instances per Compose network — Change Order.
- **Connection path** (self-hosted interim): Compartment containers join only the data networks they are entitled to (§6.5), so isolation is double-walled — network non-membership *and* engine ACL. Under Managed Databases the network wall is the VPC +

per-database credentials; the engine ACL layer is identical.
- **Object storage:** per-Compartment prefix (`c01/…`) with per-Compartment Spaces keys whose policy is prefix-scoped. AI-03 (credential isolation) extends here: an agent's runtime credentials reach exactly its Compartment's database, its Redis keyspace, and its Spaces prefix — nothing else.

7.5 Encryption

- **At rest, block storage:** DigitalOcean encrypts droplet disks and Volumes at rest (AES-256) at the platform layer. This is accepted as the baseline for the interim self-hosted deployment; no LUKS layer is added on top — key management for host-level LUKS on cloud VMs adds operational fragility (unattended reboot problem) for marginal gain against the relevant threat model (DO platform compromise is out of scope; stolen-disk scenarios are covered by DO's layer).
- **At rest, engine level:** PostgreSQL — no TDE in community builds; compensating controls are the DO layer plus strict dump handling (§7.6). MongoDB Community — same posture. Redis — ephemeral data, AOF file inherits disk encryption.
- **At rest, object storage:** Spaces server-side encryption; audit archives additionally carry their own integrity chain (hash-chained, Ch. 13) so tampering is detectable independent of encryption.
- **In transit:** TLS on every database connection **including VPC-internal** (zero-trust, NIST SP 800-207 — the private network is not a trust boundary): PostgreSQL `ssl=on` with `sslmode=verify-full` in clients; MongoDB `tls=true`; Redis TLS listener. Internal certs from a platform-internal CA generated and rotated by IaC (not Let's Encrypt — these hosts have no public names). Managed Databases enforce TLS natively.
- **Backups/dumps:** encrypted client-side (age or GPG, key in Vault) **before** upload to `sbs-dash-backups` — a leaked bucket key must not equal readable data. Detail in Ch. 14.

7.6 Operational Rules

- No production data leaves the Prod boundary (§2.2.3): Dev databases are seeded synthetically or from **sanitized** dumps produced by a reviewed script (PII/tenant content stripped), never raw prod dumps.
- Ad-hoc human access to prod data engines is break-glass only, via bastion (§3.3), with individually attributed credentials (post INF-04 remediation, §2.5), logged as a MON-01 event. Routine inspection happens through dashboards and read-only replicas if/when provisioned.
- Retention: Compartment operational data per SBS function-level policy captured at Discovery (LH-SBS-DISC-001/002 outputs); audit log 7-year minimum (Ch. 13); Redis nothing beyond declared AOF keyspaces.
- Volume/disk capacity is a Prometheus-alerted metric (Ch. 12); databases run on dedicated DO **Volumes** (not the droplet root disk) so storage scales and migrates independently of the droplet — Volume creation is part of the data-services Build Register item.

7.7 Verification

Gates (LH-SBS-INST-001): cross-Compartment access probe per engine fails (c03 role → c08 database denied in PostgreSQL, MongoDB, Redis ACL, and Spaces prefix); no public listener on any engine port (`ss -tlnp` + external scan); TLS handshake required on every engine (plaintext connection refused); `PUBLIC`/`default` access revoked (engine ACL dump review); dumps encrypted before leaving host (backup pipeline inspection); Dev contains no production data (sampled content check); Spaces buckets private (unauthenticated GET fails); audit bucket writer key cannot DELETE (negative test).