Maintenance & Patching

Patch windows, upgrade procedure, rollback

- Chapter 17 — Maintenance & Patching

Chapter 17 — Maintenance & Patching

17.1 Current State and Scope

As-built patching is unmanaged: no unattended-upgrades confirmation, no defined windows, no reboot policy, and one already-tracked drift instance (OS mix 24.04/26.04, item 21). This chapter defines the patch regime for every layer — OS packages, kernel/reboots, container images, system services (nginx, Prometheus, Grafana, Vault, BookStack), data engines, and the DO platform events outside our control — plus the rollback path per layer. Controlling requirement: **INF-05 (patching)** ; MON-06 (vulnerability scanning, §11.6) is the detection feed this chapter remediates. The governing principle: **patching is a pipeline event with a rollback, never an SSH session with `apt upgrade` and hope.**

17.2 Patch Windows and Cadence

Window	When (ET)	Scope
Standing weekly	Tuesday 05:00–07:00	Dev: everything. Prod: security-only OS patches (unattended-upgrades applies these continuously; the window is for anything needing coordination)
Standing monthly	Second Tuesday 05:00–07:00	Prod: full OS package upgrade, pending reboots, service minor versions, base-image rebuild rollout (§11.3 weekly rebuilds accumulate; deploys land here unless CVE-driven earlier)
Emergency	Any time, MON-05 incident context	Actively exploited / CRITICAL-with-fix (Trivy nightly or vendor advisory). Target: patch in Dev + deploy to Prod ≤ 48 h from advisory; §10.4 emergency path applies with 24 h backport rule
Freeze	Cutover ± 5 business days; declared SBS blackout dates	Security-only; everything else queues

Ordering rule, always: **Dev first, soak ≥ 24 h (weekly cycle) with monitoring green, then Prod.** The 1:1 Dev sizing (§2.4) exists precisely so this soak is meaningful. Within Prod, order is: wiki → monitor → vault → prod-app — least-critical first, and the bastion/alerting host (monitor) is patched *before* the app host so a monitor regression is discovered while the app host is still stable,

never both at once.

17.3 Patching by Layer

- **OS security patches:** `unattended-upgrades` enabled fleet-wide (Ansible base role, item 60) — security pocket only, no automatic reboots, mail/log output shipped to Loki. This makes security patching continuous and the windows about *coordination*, not application.
- **Kernel/reboot:** `needrestart`/reboot-required flag exported as an `integrity` metric (§12.2); pending-reboot age > 14 days = `warning`. Reboots execute in the monthly window, one host at a time, in the §17.2 order, with the bastion-loss interim SSH note (§14.4) pre-staged when monitor reboots. DO Backup taken pre-reboot on each host (tier-1 as rollback, §14.2.1).
- **Container images:** already fully specified — weekly CI rebuild against updated bases + Trivy gate (§11.3/§11.6); this chapter adds only the *rollout* rule: rebuilt images deploy to Dev on build, Prod in the monthly window or the emergency path. Rollback is redeploy-previous-digest (§6.7) — the strongest rollback in the whole stack, which is why the design pushes as much as possible into images.
- **System services (nginx, Prometheus, Grafana, Loki, certbot):** apt-managed on their hosts; minor versions ride the monthly window. **Grafana major versions** (13 → 14 class) get a Dev-first upgrade with dashboard/provisioning verification before Prod — Grafana majors have history of provisioning-format changes, and the §12.5 file-provisioning rule is what makes this testable.
- **Vault:** upgrade = new binary via Ansible, service restart, **unseal quorum required** (§8.2) — so Vault upgrades are *scheduled with shareholders*, monthly window only, never emergency-path unless the CVE is Vault-critical. Raft snapshot taken immediately pre-upgrade; rollback = binary revert + snapshot restore (§14.5.3).
- **Data engines (interim containers, §6.5):** minor/patch versions ride the image rebuild cycle. **Major versions** (PG 16→17 class) are Change-Order events with dump/restore rehearsal in Dev, never a window item. Under Managed DBs (§7.2), DO handles patch/minor; majors remain scheduled decisions.
- **BookStack:** monthly window, dump first (tier-2), standard upstream upgrade path.
- **DO platform events** (hypervisor migrations, mandatory maintenance): outside our scheduling — mitigations are the tier-1 backups, the alert stack (droplet-down alerts already routed), and reading DO's status/email notices, which route to the SBS-controlled account alias (§2.1).

17.4 Upgrade Procedure (Standard Template)

Every window execution follows one runbook shape (wiki-published, per-layer specifics as annexes):

1. **Pre:** confirm monitoring green fleet-wide; confirm last backup age green (tier-1 and applicable tier-2, §14.3 metrics); announce in ops channel (start/scope/expected end); freeze deploys for the window.
2. **Execute:** Dev-proven changes only (except emergency path); one host at a time in §17.2 order; between hosts, verify: services active, blackbox probes green (§12.2), no new alerts

for 10 min before proceeding.
3. **Verify:** per-layer smoke (nginx `-t` + probe sweep; Grafana health + one dashboard render; Vault unsealed + read probe; app surfaces via blackbox; audit writer queue age nominal).
4. **Close:** announce complete; changelog entry (narrative layer, §10.6) with versions before/after; any deviation → Build Register item within 24 h.

17.5 Rollback

Rollback authority: the window executor rolls back **without seeking approval** when any post-step verification fails — the approval was for the change; reverting to known-good needs none. Paths per layer, worst-case first:

Layer	Rollback	Bound
Container/app	Redeploy previous digest (§6.7)	Minutes
OS package set	`apt` downgrade where pinnable; else tier-1 image restore pre-window backup	≤ 30 min (§14.4)
Kernel/reboot regression	Boot previous kernel (GRUB retains) or image restore	≤ 30 min
Grafana/Prometheus/Loki	apt downgrade or image restore; dashboards/rules are repo-provisioned so config rollback = git revert	Minutes–30 min
Vault	Binary revert + pre-upgrade Raft snapshot; quorum re-unseal	≤ 2 h (§14.4 — shareholder-bound)
Data engine minor	Previous image digest + AOF/WAL intact	Minutes
Data engine major	Restore rehearsed dump (the rehearsal is the Change-Order gate)	Per §14.4 data-loss class
DNS/zone changes	Not a window item (§4.7 change control); rollback = revert record, 300-TTL bound	≤ 5 min propagation

Rollback is itself verified (same §17.4.3 smoke set) and logged; **a rolled-back change re-enters through Dev with a written cause**, never retried directly on Prod — the failure taught something or it didn't; the PR discussion says which.

17.6 Verification (Build Gates)

Gates (LH-SBS-INST-001): unattended-upgrades active fleet-wide with security-pocket-only config (Ansible-asserted, drift-checked); reboot-required metric exported and alerting at 14 d; window calendar published to wiki with §17.2 order and freeze rules; standard runbook + per-layer annexes published; one full monthly-window execution completed end-to-end on the fleet with changelog evidence; one deliberate rollback drill on Dev (failed-verification simulation → digest revert → smoke green); Vault pre-upgrade snapshot step present in its annex and rehearsed with test

shares (§14.7 linkage); OS-mix remediation (item 21) closed or scheduled through these windows — the mechanism that fixes the existing drift is the mechanism that prevents the next.