

At a glance

A two-page summary of "Continue: written by a machine, read by a skeptic." The full book tells the story. This page is the feature list and the honest gap list, as of 2026-09-24.

What LESta is

A Laravel and React rewrite of a hosting control panel, replacing VestaCP, written almost entirely by prompting Claude Code rather than by hand. A Laravel control plane holds all desired state. A separate Go agent on each server is the only thing that touches the host, and only through named, versioned, idempotent operations, never a raw shell command.

What exists

- **Web hosting.** nginx and Apache, alone or combined, with per-domain HTTPS via ACME (HTTP-01 and DNS-01).
- **DNS.** Zones and records on BIND9, with staged, validated, atomically activated config and automatic rollback on failure.
- **Mail.** Domains and mailboxes on Exim and Dovecot, DKIM signing with scheduled key rotation, antivirus and antispam scanning.
- **Databases.** Per-tenant MariaDB databases, isolated from the control plane's own instance.
- **Cron.** Account-scoped scheduled jobs, with the tenant's command text kept out of the crontab line itself.
- **Backups.** Encrypted, node-wide snapshots on a schedule, with decrypt-and-download and a real restore path back into the node.
- **Usage metrics.** Disk, request and bandwidth tracking per domain, with monthly rollups.
- **Accounts and access.** Memberships, roles and a real permission catalog, packages and quotas, resellers (one level), node-scoped admins, audited impersonation, and a hash-chained audit log.
- **Installation.** Every service installs from a written contract: dry runs, pinned and checksummed packages, preflight checks, offline install bundles, and generation-based rollback. A combined installer can add or prune services on a running node. Every installer runs its full cycle, apply, re-apply, offline, tampered artifact, failed upgrade, prune, on a disposable Ubuntu machine in CI.
- **Documentation.** A User Guide, an Admin Guide and an Installation Guide, written from the code and served inside the app.
- **Security.** A self-review pass, done by the same tool that wrote the code but told to act as a security engineer, found and fixed two high-severity issues (secrets in job payloads, a suspended node that could still authenticate) and several medium ones, and lists what it found clean.

By the numbers: about 16,900 lines of PHP, 25,300 lines of Go, 10,400 lines of shell, and roughly 41,700 lines of TypeScript. 165 PHP test files and 36 Go test files. Ten real capabilities, running through continuous integration on every push.

What is still missing

- **A real install.** Every installer has run on disposable CI machines. None has run on a server anyone can actually reach yet.
- **Real mail delivery.** Mail has been tested against real Exim and Dovecot, but only on CI runners and a laptop, never a real mail exchange talking to the outside world.
- **Ubuntu 26.04 for MariaDB.** Blocked on MariaDB's own repository not publishing for it yet.
- **More than one node.** The protocol is built for a fleet. It has only ever run against one.
- **An outside security review.** The audit so far was the same tool reviewing its own work.
- **Load testing.** The queue and cache are database-backed, a known ceiling, and untested under real traffic.
- **Off-node backups.** Backups live on local disk only, and a restore only targets the node it came from.
- **Smaller, disclosed cuts:** changing a node's web profile after install, reseller-initiated account creation, reseller nesting past one level, node admins reaching into tenant resources, and scheduled automatic rotation of node credentials.

What is next

In order: a real install on a real server, and a look at what breaks. Then load and cost checks, and closing whatever the outside review finds.

The full story, chapter by chapter, is in "Continue."