amphora

SECURITY & SELF-HOSTING

WRITTEN FOR THE TEAM THAT OWNS YOUR SERVERS

The questions your IT team will ask, answered.

Amphora is a single self-hosted application. This brief describes where data lives, how access is controlled, what is recorded, and what happens when something goes wrong.

Where the data lives

On your infrastructure. Amphora is one application plus one database file (SQLite by default; Postgres is the planned option). It runs on a VM, a NUC in the server cupboard, or your own cloud account — your choice, not ours.

No outbound telemetry. The application does not phone home. It runs on a warehouse LAN with no internet connection at all; the only outbound traffic is to services you configure yourself.

Your keys, your accounts. Carrier, AI and future integration credentials are yours, entered by you and removable by you. Shipping keys are write-only in the UI — never returned to the browser — and an environment variable can override them so credentials need never be written to the database.

No per-seat meter. Nothing about the licensing model requires a connection to us to keep working.

Who can do what

ROLE	SCOPE
Owner	The install's root account. Everything an admin can do, plus ownership-level settings. Exactly one; cannot be removed.
Admin	Full operational and commercial access: customers, catalog, rate cards, users, settings.
Warehouse	Floor work — receiving, putaway, moves, picking, packing, shipping. No pricing, billing or user management.
Developer	API and documentation access for integration work.
Vendor	Reserved for supplier replenishment. The role exists in the model; today it grants no access at all, and it is listed here so an account you see in the product is not a surprise in this document.

Roles are capability-based rather than page-based: the same check runs in the UI and in the API, so an endpoint cannot be more permissive than the screen that calls it.

Authentication

Passwords. Hashed with scrypt via Node's own crypto module — no third-party auth dependency to trust or patch.

Two-factor (TOTP). Standard authenticator apps, with single-use backup codes. Enrolment verifies a live code BEFORE enforcement, so nobody can lock themselves out mid-setup. Owners and admins can reset a locked-out teammate.

Rate limiting. Five failed attempts on an email within fifteen minutes locks that email for fifteen minutes. The limiter counts failures from the audit trail itself, so the record the security team reads is the same data the lockout acted on.

API keys. Scoped, revocable, and unable to mint further keys. Every key action is audited.

Identity is pluggable. The account model separates who a user is from how they prove it, so SAML/OIDC slots in without a data migration.

Transport security

Amphora speaks HTTP; you terminate TLS. The application is designed to sit behind a reverse proxy or load balancer you already run — Caddy, nginx, Traefik, or a cloud load balancer. That is where your certificate lives, which means your existing certificate management, ciphers and HSTS policy apply unchanged.

Session cookies are Secure by default in production. They are also HttpOnly and SameSite=Lax, so they are not readable from JavaScript and do not ride along on cross-site requests.

The LAN exception is explicit and named. Warehouses run on closed networks, and some have no certificate authority at all. Setting AMPHORA_ALLOW_INSECURE_HTTP tells Amphora to keep working over plain HTTP — which drops the Secure flag on the session cookie. It is a deliberate, single-purpose switch: use it only on an isolated network, never on anything reachable from the internet. Left unset, an install in production mode is HTTPS-only.

Nothing else listens. One process, one port. There is no agent, no separate admin service, and no inbound connection from us.

Encryption and secrets

Stated precisely, because this is where vague answers waste everyone's time. Amphora does NOT encrypt its database file at the application layer today — encryption at rest is provided by the volume it sits on (LUKS, BitLocker, FileVault, or your cloud provider's encrypted disks), which is the same control most self-hosted databases rely on. The file is protected by OS permissions and lives wherever you put it. Application-level database encryption is on the roadmap, not in the product; if your policy requires it before the disk layer, say so during evaluation rather than after.

SECRET	HOW IT IS STORED
User passwords	scrypt (N=16384, r=8, p=1, 64-byte key) with a per-user random salt, compared in constant time. The hash format is self-describing, so parameters can be raised later without invalidating existing passwords.
Session tokens	A 32-byte random token is issued to the browser; only its SHA-256 hash is stored. A stolen database yields no usable sessions.

SECRET	HOW IT IS STORED
API keys	Generated once, shown once, stored as a hash. A lost key is revoked and reissued, never recovered.
Two-factor backup codes	Hashed, single-use, and burned on redemption.
TOTP secrets	Stored in the database so the server can verify codes — one of the concrete reasons the disk should be encrypted.
Carrier / provider API keys	Write-only in the interface and never returned to the browser. An environment variable overrides the stored value, so a deployment can hold credentials outside the database entirely.

Patching and vulnerability response

A deliberately small attack surface. Seventeen runtime dependencies. Authentication is built on Node's own crypto module rather than an authentication framework, so there is no third-party auth library in the path to a session — the class of dependency that has produced some of the industry's worst CVEs.

How an update happens. Pull the release, rebuild, run migrations, restart. Migrations are forward-only and safe to re-run; the snapshot command gives you a rollback point in one line before you start.

Security fixes are called out, not buried. Security-relevant releases are identified as such in the in-app changelog, so an operator patching a warehouse can tell a bug fix from something they should schedule tonight.

Reporting a vulnerability. Email parth@brandboxai.app. We will acknowledge within two business days and tell you what we intend to do and when. We would rather hear it from you than from a customer.

What we do not claim. There is no SOC 2 report, no penetration-test certificate, and no 24/7 security operations center behind this product today. Amphora is early software from a small company, and a self-hosted install means your controls — network, disk, backups, patch windows — are the controls that matter. We would rather you know that before procurement asks than after.

What is recorded

Two independent records, both append-only. The INVENTORY LEDGER holds every physical movement — receipt, move, adjustment, status change, pick, shipment — with actor, reason code, quantities and timestamps; stock on hand is derived from it rather than stored, so a count can always be explained. The AUDIT TRAIL holds every consequential system action — sign-ins, permission changes, rate-card edits, key creation, shipment events — filterable by module and exportable as CSV. Corrections are new events; nothing is edited away.

Backup and recovery

One file plus one folder. The database and the uploads directory are the entire state. Back them up with whatever you already use.

Live-safe snapshots. A built-in command takes a consistent snapshot while the system is running, and a matching restore rebuilds an install from it — the same mechanism used to clone production into a staging box.

No vendor lock. The schema is plain SQL and every record is reachable through the documented REST API. Leaving is an export, not a negotiation.

WHAT SELF-HOSTING ASKS OF YOU

Somebody has to own the server: patches, backups, and a plan for the day the disk fails. In exchange, client data never leaves your building, the warehouse keeps working when the internet does not, and no third party can change your pricing, your terms, or your access to your own records.

NEXT STEP

Forward this to whoever owns your security review and send us their questionnaire — SIG, CAIQ, or your own. We answer it directly rather than pointing at a trust page, and we will tell you where the answer is "not yet".

getamphora.com · parth@brandboxai.app