GUINNESS · PLATFORM BRIEFING

The Guinness Platform — Infrastructure, Explained

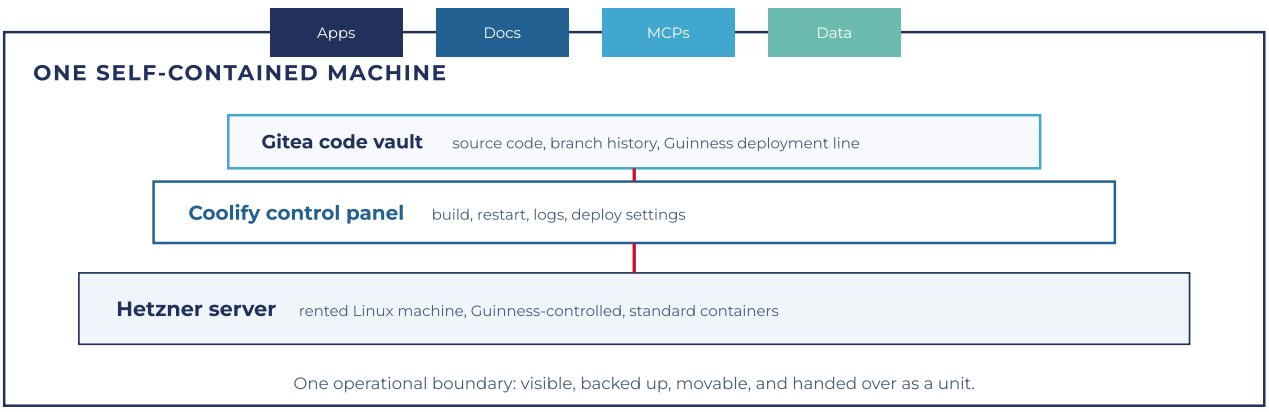
Prepared for Tom & Will · Guinness · v0.1 — the operator's companion to "How It Works"

The operator's companion: the server, the control panel, the code vault, how a change becomes live, and who logs in where.

One machine, three layers

- The entire platform runs on **one dedicated server** in a German data centre, under Guinness's control
- Three layers, all open-source: **Hetzner** (the server) · **Coolify** (the control panel) · **Gitea** (the code vault)
- Nothing external is needed at runtime — code, build and serving all live on this one machine
- Verified June 2026: every application up, the machine running light — disk and memory mostly headroom

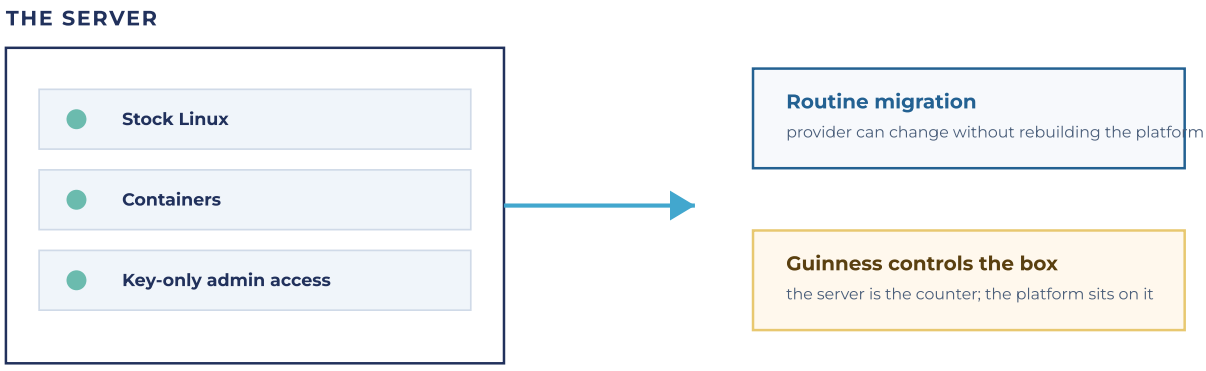
One box is a feature, not a shortcut: it means the whole estate can be seen, operated, backed up — and one day handed over — as a single, self-contained unit.



Hetzner — the server

- A standard rented Linux machine — the same kind of server Guinness already rents today
- Nothing on it is proprietary to the supplier: a stock operating system running standard containers
- Administrative access is by key only — there are no passwords to guess
- Moving provider, if ever wanted, is a routine server migration — not a rebuild

The server is the counter the microwave sits on. It matters that it is sturdy and paid for in your name; it does not matter whose data centre it stands in.

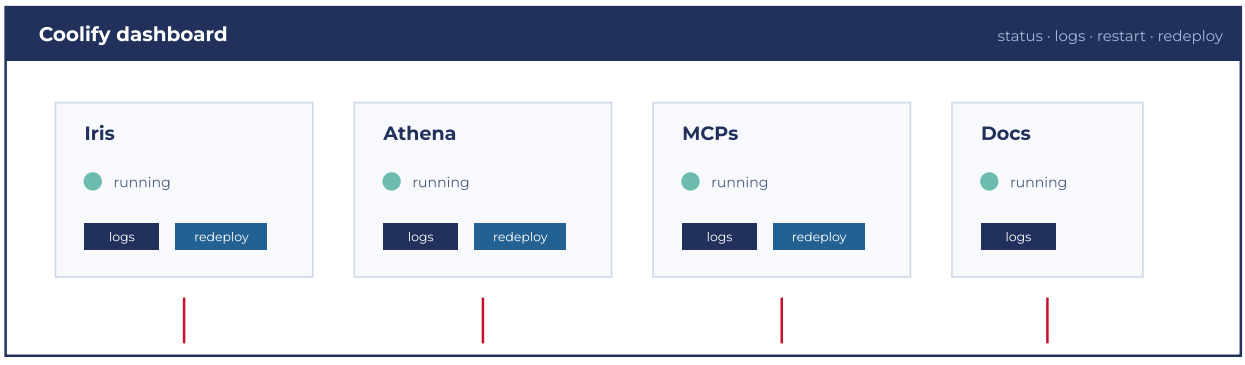


The supplier is not embedded in the machine: the workloads are ordinary, inspectable and portable.

Coolify — the control panel

- One dashboard showing **every app**: status, logs, restart, redeploy, settings
- Builds each application from the code vault and runs it in its own container
- One app can be rebuilt or restarted **without touching the others**
- The same kind of tool big platforms rent out — but self-hosted, so it belongs to Guinness

This is where day-to-day operations happen. If an app misbehaves, the control panel is where you see it and where you fix it — restart, read the logs, or roll a new deploy.

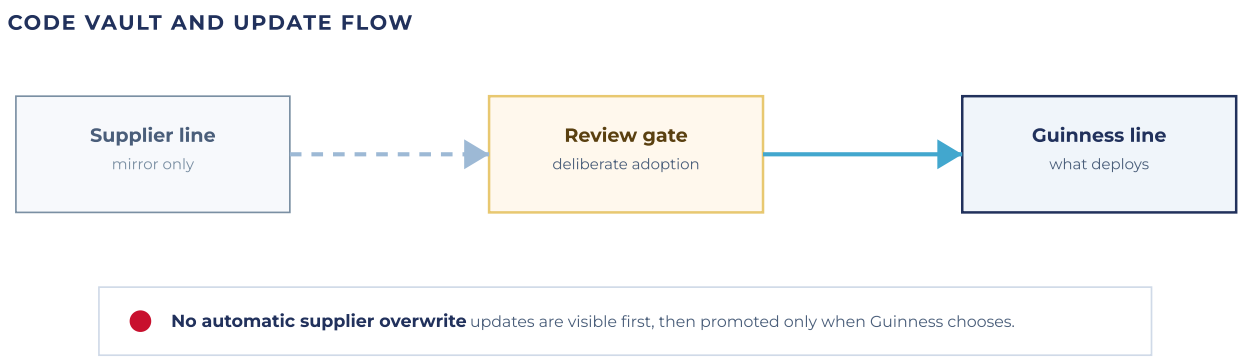


Each application is operated independently: one restart or rebuild does not disturb the rest.

Gitea — the code vault, and how updates arrive

- Every application's full source code and history, stored **on the box** — the blueprints, kept on-site
- Each app has two lines: the **supplier's line** (a read-only mirror of supplier updates) and the **Guinness line** (what actually deploys)
- Supplier updates **no longer apply themselves**: they arrive on the mirror for visibility, and are applied to the Guinness line deliberately, after review
- What you read on the Guinness line is **exactly what is running** — no hidden build artefacts

This is the ownership model in practice: the supplier can propose, but only the Guinness line deploys — and adopting an update is a conscious decision, not an automatic event.

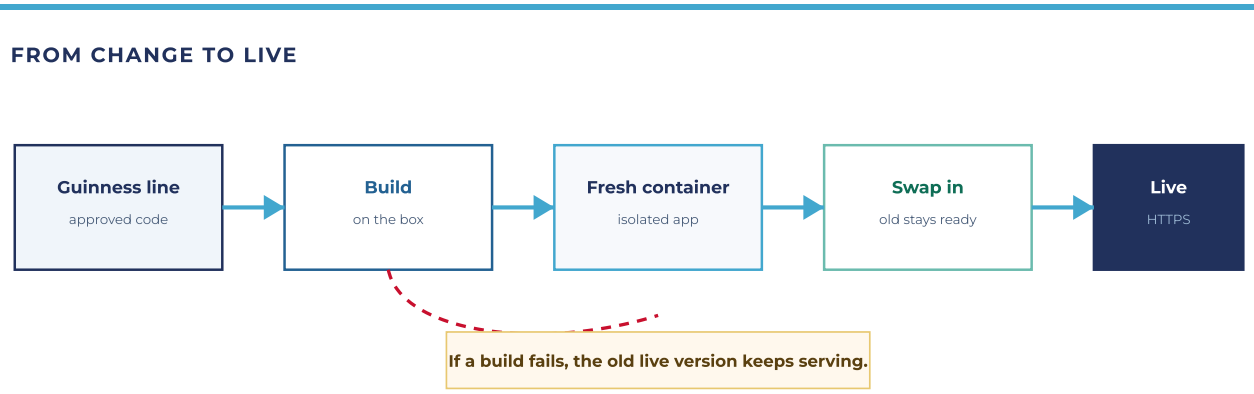


The code vault is the ownership control: propose on one line, deploy from the Guinness line.

From a change to live — the deployment path

- A change lands on the **Guinness line** in the code vault
- The control panel builds it into a fresh container and swaps it in
- If a build fails, **the old version keeps running** — failed deploys don't take the platform down
- Web addresses and certificates are automatic: each app lives at its own `…guinnessgi.x-trillion.com` address with HTTPS issued and renewed by the platform itself

The whole path runs on the box: no external build service, no outside system that has to be up for Guinness to deploy.



Deploys are local to the box: no outside build service has to be available for Guinness to ship.

guinness_mcp — your own data engine

- The first **fully Guinness-owned service**, live on the platform today
- Serves sovereign fundamentals natively from licence-free public data — no external analytics in the path
- Serves macro time series from **Guinness's own Haver subscription** — a feed Guinness funds directly
- Everything it cannot yet serve is routed to the interim engine transparently — and every answer is stamped with its source

This is the independence model running, not promised: two data circuits already in-house, the rest available to bring across on Guinness's own timeline.

The migration pattern is a strangler: native circuits grow, interim routing shrinks.

Who logs in where

Who	Where	What
Everyone	`guinnessgi.x-trillion.com`	The apps — one email one-time-code sign-in; all most users ever need
Ops (Tom & Will)	`coolify.guinnessgi.x-trillion.com`	The control panel — status, logs, redeploy, configure
Developers (Will)	`gitea.guinnessgi.x-trillion.com`	The code vault — source, history, and when new versions go live
Behind the apps	`guinness-mcp.guinnessgi.x-trillion.com`	The in-house data engine — no login needed; machines only

Personal operator accounts for Tom & Will follow once work email addresses are confirmed — each with a forced password change on first sign-in. Until then, the remaining housekeeping is tracked in the platform's hardening register and none of it blocks daily use.