GUINNESS · PLATFORM BRIEFING

The Guinness Platform — How It Works

Prepared for Tom & Will · Guinness · v0.1 (draft for design enhancement)

The CEO/ops-level solution doc: what the platform is, who controls it, and the path to full data independence.

The big idea, in one picture

Think of the Guinness platform as **a microwave sitting on your own counter.**

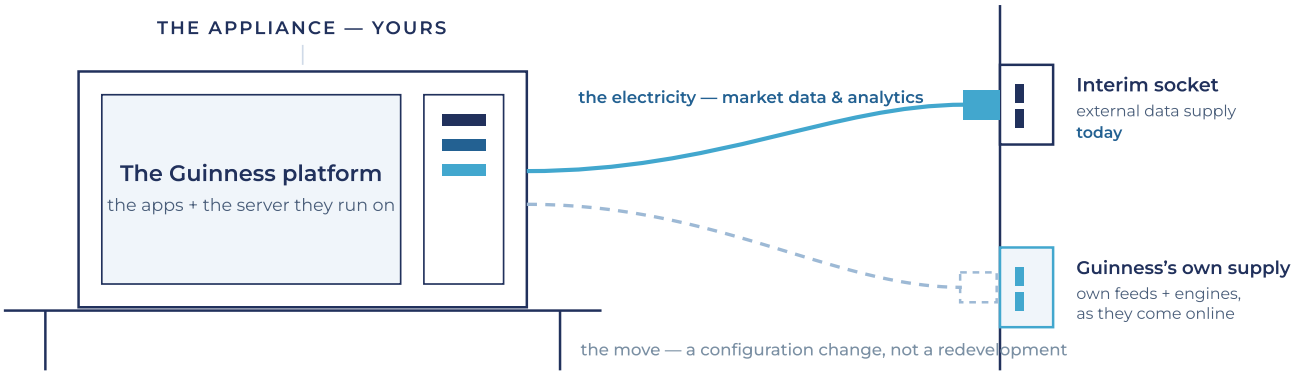
The microwave is **yours**. It's a complete, working appliance — you own it, it lives in your kitchen, and nobody can take it away. Right now it happens to be plugged into a **shared wall socket** for power, because your own dedicated supply isn't wired in yet. But here's the point: **the microwave doesn't care where the electricity comes from.** The day your own supply is connected, you unplug from the shared socket, plug into yours, and *nothing about the microwave changes.*

That is exactly how this platform is built.

- **The microwave** = the Guinness applications and the server they run on. Yours. Self-contained. Sitting on infrastructure you control.
- **The electricity** = the market data and analytics that feed the apps (yields, durations, spreads, ratings, prices).
- **The shared socket** = an interim, external data supply we currently draw on while your own supply is being wired in.
- **Your own supply** = Guinness's own data feeds and analytics engines, as they come online.

You are not dependent on any one socket. You are in control of where your power comes from — and the platform is designed so you can change it without rebuilding anything.

The rest of this document explains the appliance, who has the keys, and the plan to connect your own supply.



The day your own supply is connected: unplug from the shared socket, plug into yours — **and nothing about the microwave changes.**

The three layers of the platform

> **You already understand the foundation — you rent a server today.**
>
> This platform runs on exactly that kind of machine: a dedicated server in a data centre, rented in Guinness's name, with Guinness's bill. The server is the counter; the platform is the appliance sitting on it. Nothing about the software is tied to that particular machine — if you ever wanted to move it to a different provider, or onto a server you run in-house, it's a standard move of the same software onto a new box. **You're not renting a seat in someone else's system. You're renting a machine, and the machine is yours for as long as you pay for it — just like the one you already have.**

The platform is built from three independent layers. You own and control all three.

Layer	What it is	Plain English
Hetzner	The server	A dedicated machine in a data centre — *the same kind of rented server Guinness already operates*, just one dedicated to this platform. This is the physical home of everything — the "counter the microwave sits on." Your name on the bill, your machine: a standard, portable Linux server with nothing locked to any vendor.
Coolify	The control panel	Open-source software that runs *on* the server and manages every app — deploying them, restarting them, showing logs, handling HTTPS certificates. Think of it as the **dashboard and breaker-box** for the whole estate.
Gitea	The code vault	A private, self-hosted home for every app's source code, with full version history. Think of it as the **blueprints, kept on-site** — so the platform can be rebuilt, audited, or handed to a new developer at any time, entirely from within Guinness's own walls.

Why this matters

Every part of this stack is **open-source and portable**. There is no proprietary platform you're locked into, no vendor who can hold the system hostage. If Guinness ever wanted to move the

whole thing to a different provider — or in-house entirely — it's a standard server migration, not a rebuild.

The applications

The platform presents as a single front door — `guinnessgi.x-trillion.com` — with a sidebar of apps. Each app is independently deployed and self-describes what it does (every app exposes a live `/brian-manifest`, which is how the in-app help agent always stays accurate).

App	What it does for you
Orion (the shell)	The unifying interface at `guinnessgi.x-trillion.com`. One login, one sidebar, every app in one place.
Brian (chat)	The built-in help agent. Ask it what any app does, take a guided tour, optionally with voice narration. It reads each app's live manifest, so its answers are never out of date.
Athena	The portfolio cockpit — NAV, yield, duration, allocation, P&L, holdings and compliance for the fund. The day-to-day analytics screen.
Lexa	Sovereign & quasi-sovereign credit reports — ratings-style narratives and credit analysis.
Iris	An interactive guide to the platform — onboarding and "how do I…" walkthroughs.
Jess	Narrated video briefings — turns analysis into short, voiced video summaries.
Jules	A deck auditor — checks PowerPoint presentations against the Guinness brand and house style.
Maia	The execution bridge between Athena and the trading layer — it carries orders out and brings fills back (the order↔fill "airlock"). It is *also earmarked* as the future home of an in-house bond-maths engine, should Guinness choose to build one — but today it is the execution bridge, not a maths engine (see §5).
Iona	Voice briefings — spoken portfolio and market commentary.
Georgia	A sovereign ESG scorer.

> Each app's authoritative, always-current description lives at its own `/brian-manifest` endpoint, and Brian surfaces it on demand. This document is a snapshot; the manifests are the source of truth.

Who has the keys, and how to get in

There are **three levels of access**, depending on the role:

a) Using the apps — *for everyone*

Just go to `guinnessgi.x-trillion.com` and sign in. Each app handles its own login (email one-time-code). This is all most users ever need.

b) The control panel (Coolify) — *for ops / Tom & Will*

`https://coolify.guinnessgi.x-trillion.com` See every app, its status, its logs; redeploy or restart anything; view and edit configuration. This is the operational cockpit for the platform itself.

c) The code vault (Gitea) — *for developers / Will*

`https://gitea.guinnessgi.x-trillion.com` Every app's source code and full history. Each app has two branches: `upstream` (the latest delivered version) and `guinness` (the version actually running, which you control). New versions arrive on `upstream`; **you decide when to merge them into** `guinness` — so Guinness is always in control of what's running in production.

> **Access provisioning — to be completed.** Personal accounts for **Tom** and **Will** on Coolify and Gitea will be created once their work email addresses are confirmed. (We need: Tom's email, Will's email.) Each will receive their own login with a forced password change on first sign-in. Until then, access is via the platform owner's account.

Where the data comes from — and why that's a choice, not a dependency

This is the part to read carefully, because it is the whole point of how the platform is designed.

The apps are yours. The **data that feeds them is a configurable input** — like the power supply to the microwave. Today, three analytical inputs are drawn from an external analytics service as a deliberate **interim bridge**, for three specific and temporary reasons:

Input	Why it's sourced externally today	The route to bringing it in-house
Bond analytics — yields, durations, spreads, accrued, option-adjusted spreads	Everything here rests on **accrued interest** — the keystone figure. Get accrued wrong and every number built on it (yield, duration, P&L) is wrong too, so it must be computed in a proper, industry-standard bond-maths engine (the open-source **QuantLib**). **Maia does not yet compute accrued**, so until it does, these figures must come from the established QuantLib engine — there is no shortcut, and no approximation would be correct.	Maia becomes a candidate **only once it produces correct, QuantLib-grade accrued**; alternatively, license a bond-analytics feed. Either way it must be a genuine engine — and because accrued and the maths above it are **deterministic** (one correct answer, whoever computes it), swapping changes *whose* engine runs, never the numbers on screen (see §6).
Credit ratings (via Lexa)	A direct **Moody's / S&P** data licence is not yet in place.	Guinness contracts the ratings feed directly, or an alternative source is connected.
Pricing (the current RVM build)	A full **Bloomberg** data licence is, at current scale, prohibitively expensive.	A Guinness-licensed or alternative price feed is connected.

Each of these is a swap-in input, not a hard wire. The architecture deliberately separates *the apps* from *where their data comes from*. Changing the source is a configuration change — a URL

repoint — not a redevelopment. That is the microwave, unplugged from one socket and into another, working exactly the same.

The one thing worth being explicit about

Corporate market-data licences — Bloomberg, Moody's, S&P — **must be contracted by the institution that uses them.** They cannot be resold or proxied by a third party. So the cleanest path to full data independence runs through **Guinness's own provider relationships**. There are, in practice, three ways forward, and they are not mutually exclusive:

1. **License the data directly** — Guinness contracts the feeds it wants in its own name. *This path is already proven live:* international macro data from **Haver Analytics**, on Guinness's own subscription, runs in-house today (see §6) — the "source it yourself" option working end-to-end, not in theory.
2. **Build the in-house engines** — a Guinness bond-maths engine (Maia is its earmarked home, still to be built — and its first job is correct accrued, the keystone everything else sits on) and the CLA analytics engines would remove the need to buy some of this externally at all.
3. **Continue the interim bridge** — keep using the external analytics in the meantime, with zero disruption to users, while 1 and 2 progress.

> **The headline:** the appearance of reliance on an external provider exists *only* because the alternative supplies aren't connected yet. The platform itself is independent by design. The decision of *which* supply to connect, and when, sits entirely with Guinness.

`guinness_mcp` doesn't re-calculate — it **re-sources**. Same answer, Guinness's own source, zero disruption.

The path to a self-supplied platform — and proof it has already begun

The end state is a platform that can draw all of its power from Guinness's own supply. The important thing is that **this has stopped being a diagram and started being real — two native circuits are already live, and one of them runs on a feed Guinness funds directly.**

`guinness_mcp` — Guinness's own data engine, live today

`guinness_mcp` is an in-house data service running on your own server. It serves whatever it can from Guinness's own sources, and for anything it can't yet do, it quietly asks the external engine

and passes the answer straight back. Adopting it, app by app, is a *repoint* — not a rebuild — because every app already talks to its data through a single, swappable connection.

It is already running — with two native circuits live:

- **Sovereign fundamentals** — government debt-to-GDP, GDP, reserves, inflation, current-account balance — for any country, drawn directly from **licence-free public data**, with no external analytics in the path.
- **Macro time series** — international economic data (GDP, inflation, unemployment, policy rates and more) from **Haver Analytics**, running on **Guinness's own Haver subscription**, served in-house. Ask in plain English — *"Japan GDP"*, *"Germany unemployment"* — and a chartable series comes straight back.

For everything else, it still routes to the interim engine, transparently. *Two circuits are wired — and one of them runs on a feed Guinness pays for itself, which is the whole independence argument made literal.*

The remaining circuits — Guinness's choice, Guinness's timeline

The heavier analytical inputs — bond mathematics, credit ratings, pricing — can each be brought in-house as and when Guinness chooses: by licensing a feed directly, or as the in-house engines mature. These are deliberate, larger decisions, and most institutions run them through the interim bridge until it makes commercial sense to bring them in. **The point is simply that the decision is Guinness's** — nothing in the platform forces the choice, the provider, or the timing.

Input	Today	Brought in-house when Guinness chooses, by…
Sovereign fundamentals	**In-house (live · licence-free)** ✓	— already done
Macro time series	**In-house (live · Guinness-funded, Haver)** ✓	— already done; proves the "fund a source yourself" path
Bond analytics	Interim external bridge	an in-house bond-maths engine (yet to be built; Maia is its earmarked home), or a licensed feed
Credit ratings	Interim external bridge	a direct ratings licence
Pricing	Interim external bridge	a licensed or alternative price feed

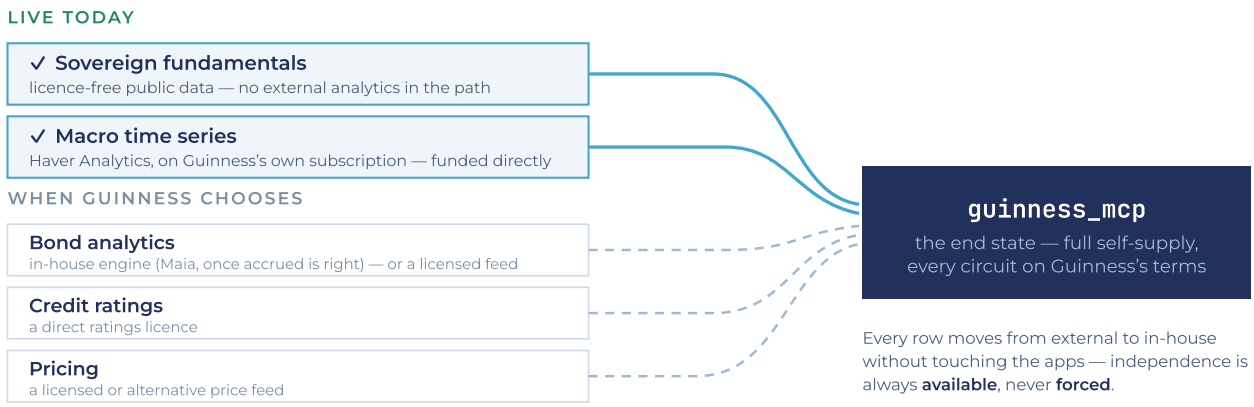
> **The headline: not locked in.** Every row above can move from *external* to *in-house* without touching the apps — and Guinness decides which, and when. The platform is built so that independence is always *available*, never *forced*. Owning the appliance and controlling its supply — that optionality — is the whole point, and **two circuits already prove it works: one on licence-free public data, one on a feed Guinness funds directly.** The choice is no longer hypothetical; it's been exercised twice.

A worked example — accrued interest

Accrued interest on a bond is a precise, deterministic figure: given the correct bond terms, it is the **same number no matter which engine computes it**. There is no house view to it — only one correct answer. It is also the **keystone**: yield, duration and P&L are all built on top of it, which is exactly why it is the right test of the whole principle.

So the moment Guinness's own engine is ready to provide accrued, `guinness_mcp` simply sources it from there instead of from the interim engine — and **not a single figure on any screen changes.** Same answer, Guinness's own source, zero disruption.

That is the whole principle made literal: `guinness_mcp` doesn't *re-calculate* anything — it *re-sources* it. The maths always runs in a proper analytics engine; all that changes is whose. Which is exactly why "not locked in" is a real, low-risk position and not a leap of faith.



Quick reference

What	Where	Who
The apps (front door)	`guinnessgi.x-trillion.com`	Everyone
Control panel	`coolify.guinnessgi.x-trillion.com`	Ops / Tom & Will
Code vault	`gitea.guinnessgi.x-trillion.com`	Developers / Will
In-house data engine (live)	`guinness-mcp.guinnessgi.x-trillion.com`	Behind the apps
Per-app description	each app's `/brian-manifest` (or ask Brian)	Everyone

Each application's `/brian-manifest` is the live source of truth for what it does; this document is a point-in-time summary.