Gofai

DEPLOYMENT GUIDE

One integrity layer, every client environment.

Gofai is a data-integrity verification layer that deploys as two containers into the environment your data already lives in — AWS, a private VPC, or on-premises hardware. Same software, same guarantees, wherever it runs, and the data never leaves that environment.

TARGETS **AWS · Private VPC · On-premises**

00 — PURPOSE

The questions that matter before you deploy.

Before adopting new infrastructure, a technical evaluator works through a short list of hard questions: will it fit, will the data stay inside the perimeter, can one team see everything in one place, how does it connect to what already runs, who owns the output, and how fast can it stand up.

This guide answers those questions at the altitude they live at. It is deliberately not an implementation manual — the engineering details exist and are available under technical review. What follows is enough to know that Gofai fits, and why, regardless of where it runs.

01 — THE FIT

It sits beside the pipeline, not in its way.

Gofai deploys as two containers — the Gofai Engine, our neuro-symbolic Large Metadata Model (LMM), and the Gofai Console. They run on whatever container platform is already in place: ECS, EKS, or EC2 in AWS; Kubernetes or Docker in a private VPC; or a container runtime on owned hardware. The software is identical across all three.

Gofai reads the pipeline at the points that matter and writes its findings to storage the environment owner controls. It is not in the write path. It changes no schema. It replaces nothing.

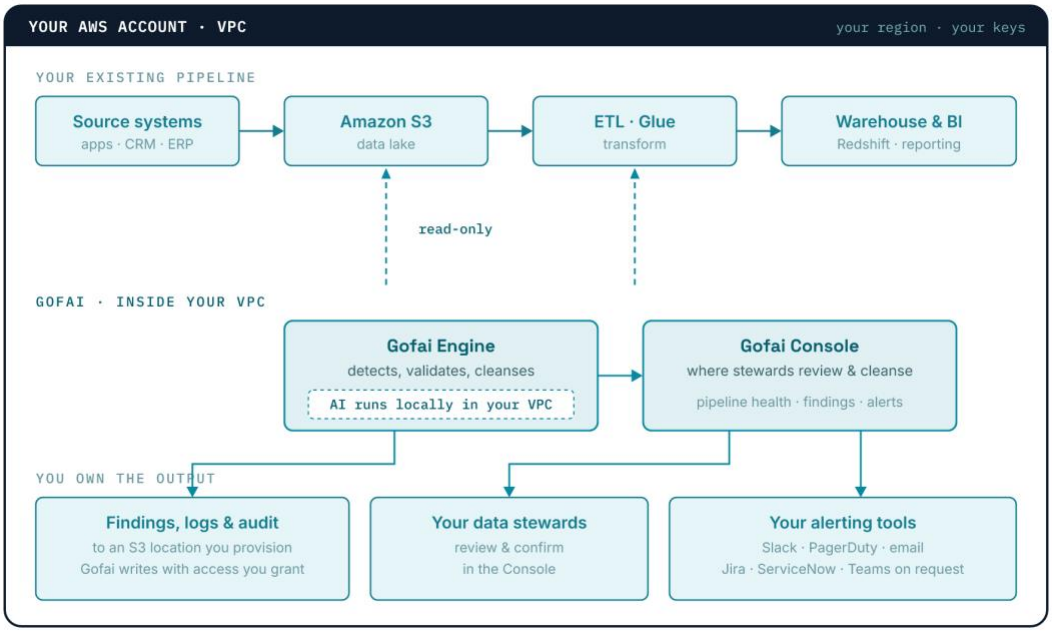

Figure 1 — Deployment fit. Gofai reads the pipeline read-only, never sits in the write path, and writes findings and logs to a storage location the owner provisions. Turn it off and the pipeline runs exactly as it does today.

What it needs. A storage location — an S3 bucket, a blob store, or a filesystem — and scoped, revocable write access for Gofai to deposit findings, logs, and audit records. That is the extent of the access Gofai requires. It brings no storage of its own and writes nowhere it was not pointed.

At the Bronze-to-Silver boundary. For environments built on a medallion lakehouse, Gofai occupies a precise and familiar position: the boundary between Bronze and Silver. Raw data lands in Bronze exactly as it arrived; Gofai watches that arrival, detects semantic errors, and surfaces corrections — the record that reaches Silver has been verified at the point it was created, not simply reformatted. The Gold layer, the business-ready models downstream teams and dashboards consume, inherits that trust without any change to how it is built. Gofai does not add a layer to the architecture. It turns the Bronze-to-Silver transition into a verification step rather than an act of faith — which makes it a component that slots cleanly into a reference architecture already in use.

02 — WHERE IT RUNS

The same two containers, three ways to run them.

The Engine and Console are ordinary containers, so where they run is an operational choice, not something the product dictates. The three targets below are the ones teams choose most often. Client environments differ — one on AWS, another on Azure, another air-gapped on their own hardware — and the same two containers running unchanged across all three lets Gofai cover an entire client portfolio instead of a single environment. The guarantees in Sections 03 and 04 hold identically across all of them.

Deployment target	Runs on	Best when
AWS	ECS · EKS · EC2	The environment already operates in AWS and Gofai should sit beside an existing S3 → Glue/Lake Formation → Redshift/Athena estate.
Private VPC	Kubernetes · Docker	A private network — in any cloud — where Gofai should live inside that boundary under existing policies.
On-premises	Container runtime on owned hardware	Data residency, air-gap, or regulatory rules require everything to stay in the owner's own data center.

One deployment story. Whichever target is chosen, the steps are the same: deploy two containers, point Gofai at storage the owner controls, and grant scoped write access. There is no data migration and nothing to re-platform.

Standing it up

1	**Deploy the two containers** Pull the Engine and Console images and run them on the platform of choice — ECS, EKS, EC2, Kubernetes, or bare hosts. No orchestration that is not already in use.
2	**Point Gofai at storage the owner controls** Give it a bucket, blob store, or filesystem path for findings, logs, and audit records. Gofai brings no storage of its own.
3	**Grant scoped, revocable write access** Gofai reads the pipeline read-only and writes only to the location provisioned. Revoke access at any time and it stops writing.

Hours, not days. From there, Gofai learns what good data looks like for each source, guided by a few simple yes/no answers from data stewards wherever the data is ambiguous. No engineers writing rules, no machine-learning project to staff.

A note on environments. This guide references AWS services by name because that is where Gofai's first enterprise deployments live, but nothing about the deployment is AWS-specific. The same two containers run unchanged in a private VPC on any cloud, or on-premises hardware in a data center. The deployment story does not change: Gofai runs where the data already is.

03 — THE PERIMETER

The data stays inside. All of it.

Every component runs inside the environment, including the AI that interprets the data. There is no phone-home, no managed cloud data is shipped to, no telemetry of record contents. The boundary around the environment — whichever target was chosen — stays unbroken. For engagements where a data-residency guarantee has to be made to the environment's owner, this is the property that makes the guarantee real rather than contractual.

Figure 2 — Data sovereignty. Everything Gofai needs to do its job lives inside the perimeter, including AI inference. What crosses the boundary is nothing.

04 — SECURITY

Built to pass a security review, not work around it.

Because Gofai runs entirely inside the environment, it inherits the controls already enforced there — no exception process, no new approval to route through security. There is no new vendor cloud in the data path, no outbound data movement to account for, and no shared-custody arrangement to negotiate. The security posture is a direct consequence of the architecture, not a set of promises layered on top of it.

Security properties at a glance

- **No data leaves the perimeter.** No records, no metadata, and no inference traffic cross the boundary. There is nothing to egress and nothing to phone home.
- **No external model calls.** AI interpretation runs locally, inside the environment. Data is never sent to a third-party model or hosted API.
- **The owner holds all data and credentials.** Gofai brings no storage of its own. It writes only to the location provisioned, using scoped, revocable access it is granted.
- **Deploys under existing controls.** Gofai runs under the environment's own identity and network policies. It introduces no new trust boundaries and requires no exceptions for a security team to carve out.
- **Read-only against the pipeline.** Gofai observes the data flow without sitting in the write path. It cannot alter, delay, or block the data downstream systems depend on.
- **An audit trail the owner controls.** Every finding is written to the owner's own storage with the rule it was checked against and the derivation that produced it. It can be inspected independently of Gofai, under the owner's own retention policies.

For regulated and security-sensitive environments, this is the difference that matters: the record of what Gofai found lives in the owner's environment, under their governance, and would survive Gofai being switched off entirely.

05 — ONE VIEW

Every pipeline, in a single pane.

However many pipelines run, and wherever they run, the Gofai Console gives stewards one place to see them — status, findings awaiting review, and anomalies — across all of them at once.

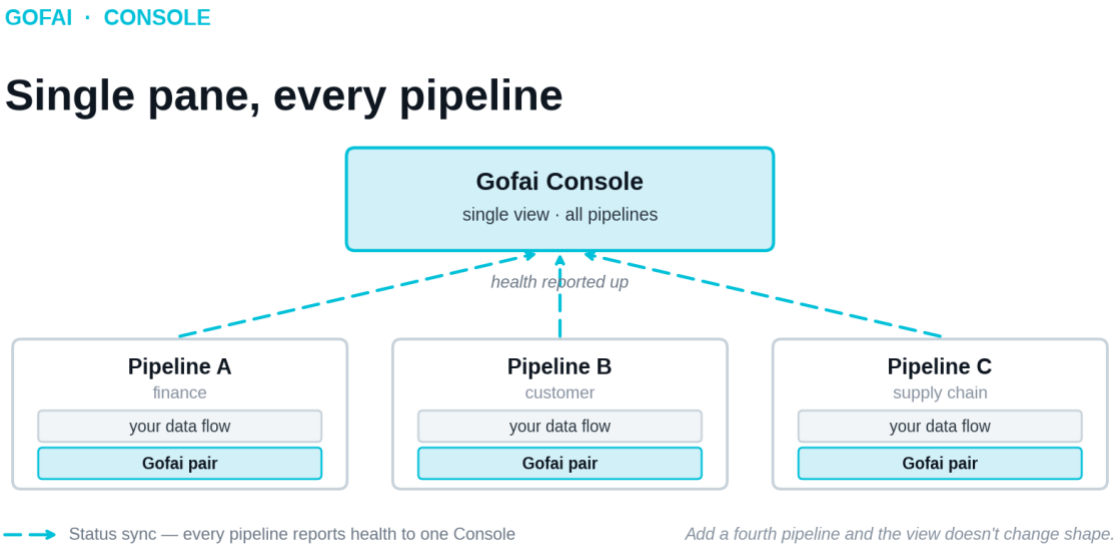

Figure 3 — Single pane across pipelines. Each pipeline keeps its own Gofai pair; all of them report into one Console. One screen, not many.

06 — THE QUESTIONS

The questions, answered.

Q1 Does this fit into an existing environment without disrupting what is already there?

Yes. Gofai deploys as two containers onto the platform already in use — AWS (ECS, EKS, EC2), a private VPC, or on-premises hardware. It sits beside the pipelines and reads them. It is not in the write path, requires no schema changes, and replaces nothing. In a medallion lakehouse it slots in at the Bronze-to-Silver boundary and changes nothing about how Bronze is landed or how Gold is served. Turn Gofai off tomorrow, and the pipelines run exactly as they do today.

Q2 Does the data stay inside the perimeter?

Yes, completely. Every component runs inside the environment, including the AI that interprets the data. No data, no metadata, and no inference traffic crosses the perimeter. There is no phone-home. Live data never leaves the environment.

Q3 Can one team see what is happening across all pipelines in one place?

Yes. The Gofai Console gives stewards a single view of pipeline health across every pipeline Gofai monitors — status, findings awaiting review, and anomalies — no matter how many pipelines or environments are in play. One pane, all pipelines.

Q4 How does this connect to the reporting and alerting tools already in use?

Gofai pushes notifications into the tools teams already live in. Slack and PagerDuty are supported out of the box; Jira, ServiceNow, and Microsoft Teams are available on request. Alerts can also

be delivered by email, targeted per pipeline. Findings land in the existing workflow rather than asking anyone to watch one more screen.

Q5 Who owns the findings, the audit trail, and the reports?

The environment's owner owns all of it. Findings, the rule each was checked against, the derivation that produced it, and the full audit trail are written to a storage location the owner provisions and grants Gofai scoped, revocable access to. Because every Gofai finding carries its own reasoning, teams and auditors can inspect exactly why any record was flagged. The record of what Gofai found stays with the owner.

Q6 How quickly can it get running?

Standing it up is hours, not days — deploy the two containers, point Gofai at a storage location the owner controls, and grant scoped write access. There is no data migration and nothing to re-platform. From there, Gofai learns what good data looks like for each source, guided by a few simple yes/no answers from data stewards wherever the data is ambiguous.

07 — NEXT STEP

See it on one pipeline.

The fastest way to answer whatever is still open is to run Gofai on a real pipeline: one data sample, a short round of yes/no confirmation from a data steward, and Gofai operating it end to end — inside your environment, alongside what is already running.

Start at **gofai.io/guided-build** or request a technical review to walk this deployment guide against your own architecture.