superluminar

EXAMPLE DELIVERABLE

Agentic SDLC Maturity Workshop

Where your teams stand on building and running agents, the target on AWS, and the path forward.

PREPARED FOR

Reimann Software GmbH

B2B SaaS (CPQ for industrial manufacturers) · ~280 engineers · Karlsruhe · on AWS

EXECUTIVE SUMMARY

The short version

You build with agents well. You run them in production immaturely. That gap is the finding, and it is the spine of this report. A product owner stood up a working-looking configurator assistant with Claude in days; the last 20 percent, the integration, edge cases, multi-tenant isolation, correctness and customer-facing guardrails, is where it got hard. That is not a model problem. It is a platform problem.

WHERE YOU STAND	FIRST BUILD TO START NOW	INDICATIVE FIRST ENGAGEMENT
Build L2, Run L1	**Agent platform**	**~€39,000**
Platform emerging on build; Team on run	A production agent platform foundation	~28 person-days · 6 to 8 weeks · ~€600 to 1,400/mo run

Where you stand

On the build track you are at **L2, Platform emerging**: a central gateway for dev agents, shared prompt and skill libraries, code-quality evals wiring into CI, and an informal AI guild. On the run track you are at **L1, Team**: one or two agentic features live in production, each on its own hand-built plumbing, with manual eval before release and no shared substrate. Your ability to ship agents to customers lags your ability to build with them.

What to do first

Build the missing layer: a **production agent platform foundation** on Amazon Bedrock AgentCore, with per-session isolation, shared memory and identity, observability, and a trajectory eval gate in CI. Migrate the existing configurator assistant onto it as the golden-path proof. That lifts the run track from L1 toward L2 and gives your guild a platform to enable onto, instead of a vacuum.

> **The gap is the finding**
>
> You build with agents faster than you can run them. Closing that gap is a platform problem, not a model problem, and that is exactly what the first build delivers.

All figures in this document are illustrative ballparks for an example client and are not a quote. They show the shape and order of magnitude of a real engagement.

OUTCOME 01

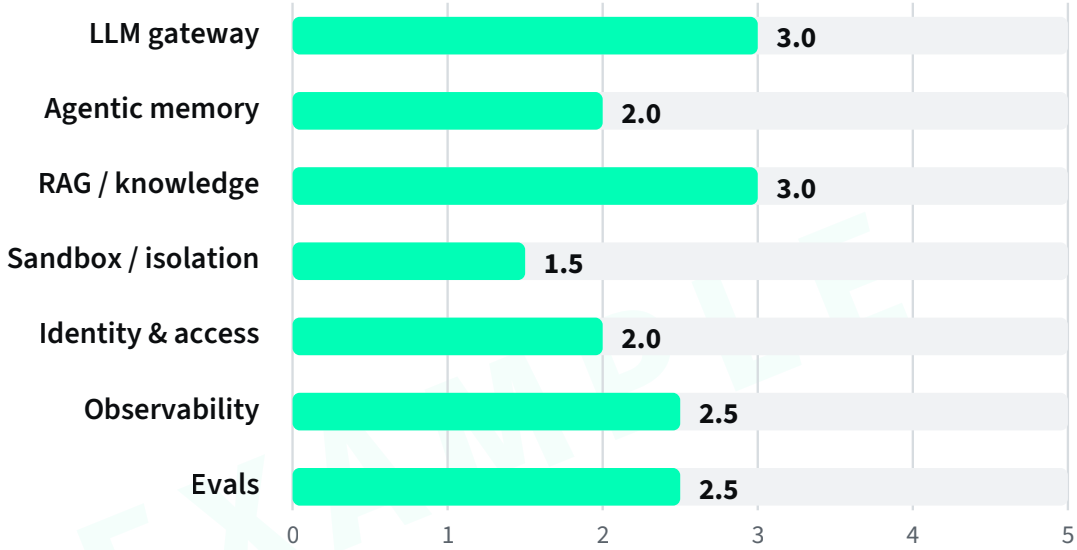
Where you are

An honest read on your agentic platform, your two-track maturity, and how your teams are organised to build and run agents.

DELIVERABLE 01

Maturity scorecard

Your agentic platform scored across seven primitives, 0 to 5. The scores split cleanly along the build and run line: what serves building scores highest, what production agents need scores lowest.



Platform maturity across seven primitives (0 = absent, 5 = a governed, shared capability). Illustrative.

Strong on the build side. LLM gateway (3.0), RAG / knowledge (3.0) and evals (2.5) are your highest scores, because they grew up serving engineers: a central gateway for dev agents, a couple of Bedrock Knowledge Base pilots, and code-quality evals wiring into CI.

Weak on the run side. Agentic memory (2.0) and identity for agents (2.0) are per-app and ad hoc. Production observability (2.5) trails dev tracing. Each is a primitive that customer-facing agents need and that building with agents can do without.

The standout gap: sandbox / per-session isolation (1.5). This is the real production-readiness gap. Running other people's inputs through an agent with tools, without per-session isolation, is the line between a demo and a product. It is the first thing the recommended build fixes.

DELIVERABLE 02

Two-track ladder placement

The differentiator is not whether you use AI. It is how much structure, verification, and human judgment surround the output. We score that on two tracks, L0 to L3, and we never average them. You sit at **Build L2** and **Run L1**, and the distance between them is the headline.

Level	**Build with agents** (engineers shipping product)	**Run agentic systems** (agents shipped to customers)
L3 Agentic factory	Self-serve internal platform. Golden paths. The harness (rules, tools, evals) governed as a shared, versioned asset. Model routing.	Self-serve production agent platform. Identity and governance. Evals as hard release gates. Stream-aligned teams ship customer-facing agents without rebuilding the plumbing.
L2 Platform emerging	**YOU ARE HERE** Central gateway for dev agents. Shared skill and prompt libraries. Code-quality evals wired into CI. An enabling / platform team forming.	Shared production primitives: memory, RAG, identity, observability. Trajectory and output evals run as release gates. A platform team forming.
L1 Team	Shared prompts and tools. An AGENTS.md in the repo, reviewed in PRs. Manual review of agent-written code. Still siloed.	**YOU ARE HERE** One or two agentic features in production on hand-built plumbing. Manual eval before release. Basic logging, no shared substrate.
L0 Me & Claude	Individuals on copilots, ad hoc. No shared tooling, no evals, no governance.	Throwaway agent demos and scripts. No deployment, no eval harness, no observability.

The two tracks look alike at L0 and split apart by L3: running agents in production adds primitives that building with agents does not need, per-session sandbox isolation, customer-facing guardrails, identity, and agent-to-agent (A2A) composition. Illustrative placement.

The AWS substrate behind L2 and L3

The rungs above L1 run on a shared platform: Amazon Bedrock and AgentCore (Gateway, Memory, Runtime, Identity, Observability), plus Bedrock Knowledge Bases, Guardrails, and Evaluations. It is a substrate beneath the rungs, not a level on the ladder. The next outcome shows the target.

DELIVERABLE 03

Team and process read

Maturity is not only technical. We read your organisation through Team Topologies, because a shared platform with no team to own it will not hold. The gap on the run track is also an org gap.

Team type	Status	Finding
Stream-aligned teams	Have	About 30 teams ship product. Around 5 have started building agentic features, each on its own plumbing.
Platform team (agent substrate)	Gap	A cloud and infrastructure platform team exists, but no one owns a shared **agent** platform. This is the missing layer.
Enabling team	Partial	An informal AI guild spreads prompts and tools, but it enables onto nothing shared.

Where the gaps are

Your guild is enabling onto a vacuum, and stream-aligned teams reinvent the run-time plumbing per feature. The fix is structural: stand up a platform team to own the agent substrate, formalise the guild as an enabling team, and give stream-aligned teams golden paths to ship on.

OUTCOME 02

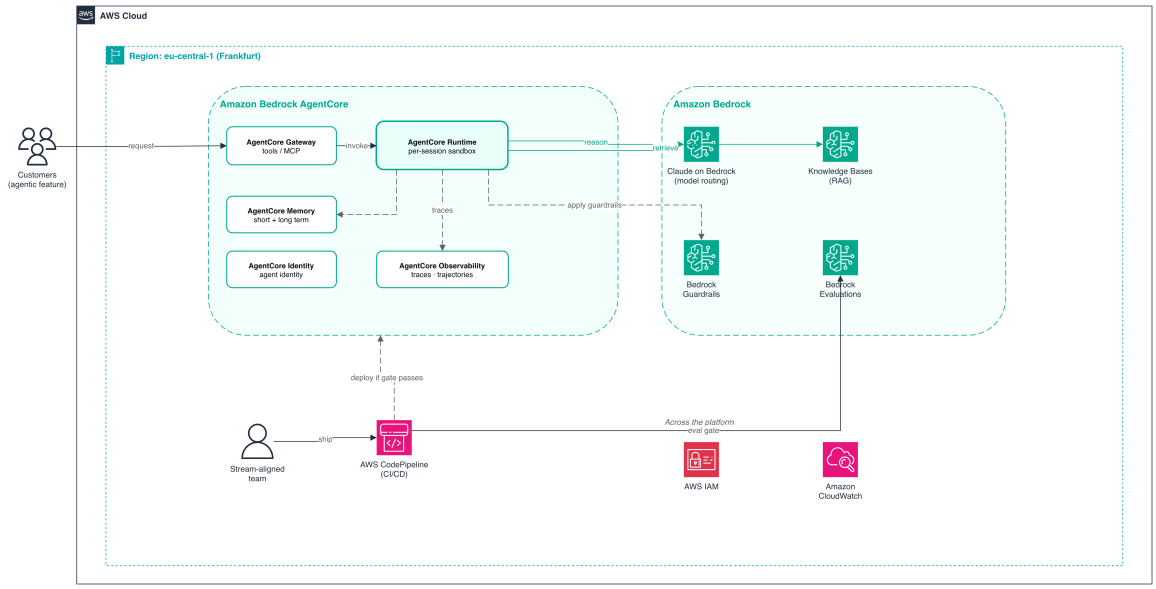
What good looks like for you

The target: an AWS-native agentic platform, the operating model to match, and where managed services beat open source.

DELIVERABLE 04

AWS-native target architecture

One shared platform that both tracks run on, built on Amazon Bedrock and Bedrock AgentCore. Stream-aligned teams ship features onto it; they do not rebuild the plumbing each time. Everything stays in eu-central-1 (Frankfurt).



Production agent platform on Bedrock AgentCore. Editable draw.io source provided alongside this report.

The request path (run)

A customer uses an agentic feature. It calls the AgentCore Gateway for tools, runs in the AgentCore Runtime with per-session sandbox isolation, draws on AgentCore Memory and Bedrock Knowledge Bases, and is bounded by Bedrock Guardrails. Claude on Bedrock is the model, with routing. Every call is traced in AgentCore Observability.

The build path (ship)

Teams ship through CI/CD (CodePipeline) carrying a **trajectory and output eval gate** powered by Bedrock Evaluations: a regressing agent change is blocked before release, not after. AgentCore Identity gives each agent its own identity; IAM stays least-privilege; CloudWatch backs observability.

> **Why managed AgentCore, not hand-built**
>
> You already run the plumbing by hand, per team. AgentCore gives you gateway, runtime isolation, memory, identity and observability as managed building blocks, so your platform team owns golden paths instead of maintaining infrastructure. That is the shortest path from L1 to L2 on the run track.

DELIVERABLE 05

Target operating model

A platform is only as good as the team that owns it. The target is three team types with clear ownership: a platform team owns the substrate, an enabling team raises practice, and stream-aligned teams ship features on golden paths.

Team	Owns	What changes for them
Platform team *(new)*	The agent substrate: AgentCore Gateway, Runtime, Memory, Identity, Observability, the eval gate, and golden paths.	Stand it up from the cloud platform team plus AgentCore. It runs the shared platform so feature teams do not have to.
Enabling team *(formalise the guild)*	Practice: prompt and skill libraries, patterns, reviews, raising the bar across teams.	The AI guild becomes a real enabling team that enables teams onto the platform, not onto a vacuum.
Stream-aligned teams *(your ~30)*	Customer-facing agentic features, shipped on the golden paths the platform team provides.	They stop rebuilding memory, identity and isolation per feature and ship on shared primitives.

The target in one line

Stream-aligned teams ship agentic features on golden paths a platform team owns, with an enabling team raising everyone's practice. That is L3 on both tracks, and the operating model is how you get there.

DELIVERABLE 06

Managed versus open source

We are AWS-primary, and at your stage managed services win on most of these. Self-hosting earns its place only where there is a clear, paid-for reason. Here is where we draw the line for you.

Capability	Recommended	Why
Agent gateway	AgentCore Gateway (managed)	Self-hosting an MCP gateway adds run cost without payback at this stage.
Agent memory	AgentCore Memory (managed)	Open source only if a hard portability requirement appears.
RAG	Bedrock Knowledge Bases (managed)	Standardise your pilots onto one managed path.
Evals	Bedrock Evaluations + custom eval sets	Managed core, with open source for a few bespoke domain metrics.
Vector store	S3 Vectors (low volume), OpenSearch Serverless (at scale)	Same sizing logic as the readiness report: do not reach for the most expensive option by default.

The rule of thumb: managed by default, open source only where a concrete requirement (portability, a bespoke metric, a cost cliff) pays for the extra run cost and operational load you take on.

LIVING REFERENCE

The full capability map, from open source to AWS-native, is kept current at superluminar.io/agentic-stack/.

OUTCOME 03

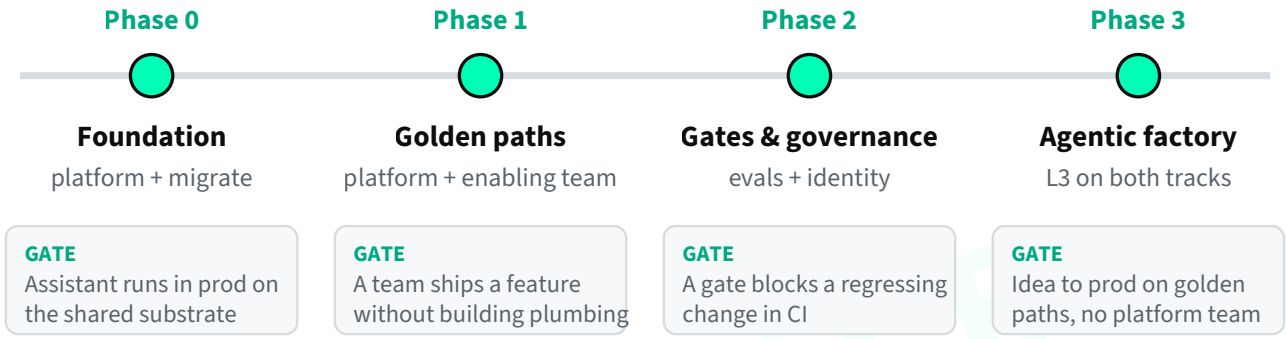
How to get there

A phased climb up the ladder, each step gated on evidence, and the first build that starts it.

DELIVERABLE 07

Phased roadmap up the ladder

Four phases from where you are to an agentic factory. Each phase is gated on evidence, not ambition: you only fund the next step once the previous one has earned it.



Each gate is a go/no-go. The first build is Phase 0. Illustrative timings.

Phase	What happens	The gate to clear before moving on
0 · Platform foundation	Build the AgentCore foundation and migrate the configurator assistant onto it.	The assistant runs in production on the shared substrate, with per-session isolation and eval gates passing.
1 · Platform team + golden paths	Stand up the platform team, formalise the enabling team, migrate 2 to 3 more features.	A stream-aligned team ships an agentic feature without building plumbing.
2 · Gates and governance	Evals as release gates across the board, agent identity and governance, A2A where needed.	A release gate blocks a regressing agent change in CI, proven, not assumed.
3 · Agentic factory	Self-serve platform; L3 on both tracks.	A new agentic feature goes from idea to production on golden paths with no platform-team involvement.

DELIVERABLE 08

Recommended first build

The foundation piece, laid out in full. It lifts the run track from L1 toward L2 and gives your enabling team a platform to enable onto. This is the decision in front of you.

THE BUILD	EFFORT	TIMELINE	INDICATIVE PRICE
Agent platform foundation AgentCore + eval gate	**~28 days** person-days	**6 to 8 wks** elapsed	**~€39k** + ~€600 to 1,400/mo run

Scope

- A production agent platform foundation on Bedrock AgentCore: Gateway, Memory, Identity, Runtime with per-session sandbox.
- AgentCore Observability for traces and trajectories, and Bedrock Guardrails for customer-facing safety.
- A trajectory and output eval gate wired into CI (CodePipeline) on Bedrock Evaluations.
- The existing configurator assistant migrated onto the platform as the golden-path proof.

Explicitly out of scope

- Migrating every agentic feature at once (Phase 1 onward).
- Standing up the full platform team org change (we design it; you staff it).
- New customer-facing features beyond the one migration.

What you get

- One real agentic feature running in production on a shared, isolated, observable substrate.
- A working eval gate that blocks regressions in CI, proven on a real change.
- The golden path and patterns for your other teams to ship onto.
- Your platform and enabling teams upskilled alongside ours. **You own what we build.**

The decision

Fund a roughly €39,000, 28 person-day build that closes the gap the workshop found: it moves you from running agents by hand to running them on a shared platform, with the assistant as live proof. Gated, EU-resident, and measured so you widen it on evidence.

superluminar

Not a vendor, a sparring partner.
superluminar · AWS Advanced Consulting Partner · Hamburg