LYZR × GOOGLE VERTEX AI

Filling the Gaps in Enterprise Agent Development

Vertex AI gives you the building blocks but only if you're a developer. Lyzr gives you the centralised Agentic platform unified, modular, accessible to every team, and built to ship production-grade agents in 48 hours, not quarters.

The take

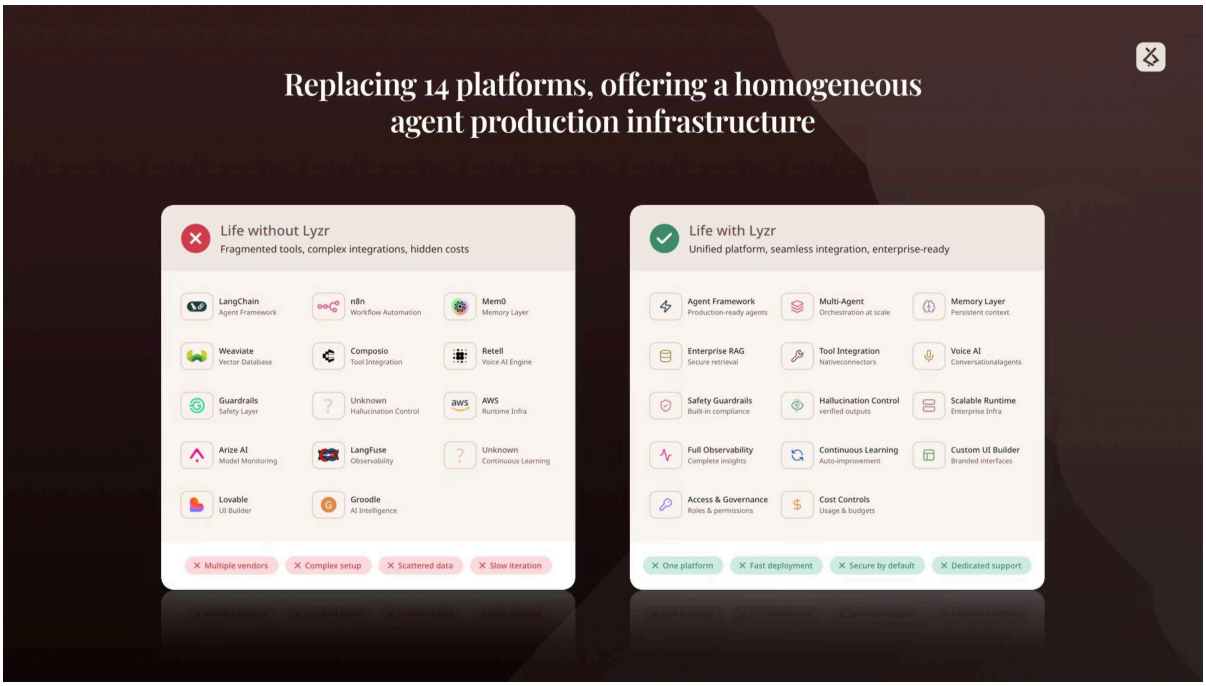
Vertex AI is a powerful, deeply developer-centric platform and that is its core limitation. It's built for engineers and data scientists, and only for engineers and data scientists. The build experience is rigid, the iteration loop is slow, and the speed of going from idea to a deployed agent on Vertex is dramatically lower than on Lyzr, where the same agent can be designed, simulated, and shipped in hours. On top of that, going from Vertex's primitives to a production agent means stitching together a dozen separate platforms frameworks, vector DBs, observability, guardrails, runtime, UI and writing a lot of glue code. Most projects never make it past the prototype.

Lyzr replaces that fragmented stack with one unified platform. Drag-and-drop agent design, a centralized model registry, a built-in simulation engine, and one-click deployment all sitting on infrastructure you can run entirely inside your own VPC. The result is the same enterprise control Vertex offers, but with a development loop that's 5–10× faster and accessible to teams without developers.

The big picture: one platform vs. a fragmented stack

Building on Vertex AI typically means standing up around 14 separate tools - LangChain for orchestration, n8n for workflow, Weaviate for vectors, Mem0 for memory, Arize for monitoring, LangFuse for traces, custom guardrails, custom UI builders, and so on. Every one of those is a contract, an integration, a security review, and a thing to keep in sync.

Lyzr collapses all of that into one platform with a single security model, single billing, single audit trail, and a single development experience.



Replacing 14 platforms with a homogeneous agent production infrastructure.

Where Lyzr fills the gaps in Vertex AI

This isn't about competing on model training. Vertex is excellent at what it does. The gaps show up the moment you try to take an idea and turn it into a deployed, observable, governed agent that real business users can rely on.

Gap in Vertex AI	How Lyzr fills it	Lyzr feature
Drag-and-drop agent builder	Build a working agent visually in minutes - no code required	**SuperFlow**
Pre-built agent blueprints	200+ shipped agents across BFSI, healthcare, telco, sales, marketing	**Agent Marketplace**
Visual architecture designer	Lay out multi-agent systems, memory, tools, and guardrails on a canvas	**Architect by Lyzr**
Centralized agent registry	One versioned home for every agent, tool, and memory layer — accessible across SuperFlow, Architect, and A2A without redeployment	**Agent Registry**
Pre-production simulation	Test agents against thousands of real-world scenarios before they ship	**Agent Simulation Engine**

Gap in Vertex AI	How Lyzr fills it	Lyzr feature
Continuous improvement	Production traces feed back into the agent automatically — agents get better without a human in the loop	**Agent Improvement Engine**
Cross-agent communication	Agents talk to each other natively, with shared memory and routing	**Agent Mesh**
Production voice agents	Multi-dialect voice agents on managed voice infrastructure — no telephony or audio-pipeline work	**Voice System Builder**
Full-stack VPC deployment	Deploy Lyzr Agent Studio end-to-end inside your VPC — including on GCP, with native Vertex model integration	**Agent Studio (VPC)**
One-click production	Move from build to live with a single deploy — no DevOps required	**SuperFlow Deploy**

Each row in this table is a place where a Vertex project would either need to be built from scratch, integrate a third-party tool, or live in production without that capability at all. On Lyzr it's a feature in the product.

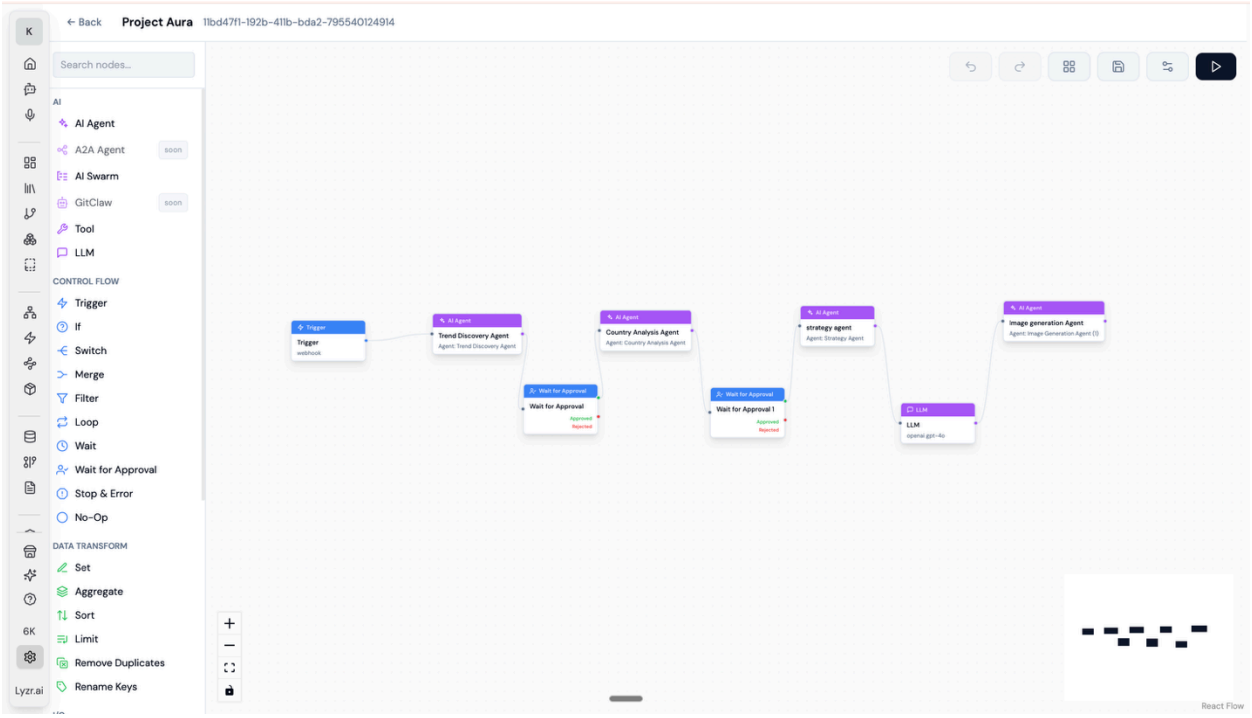
How development on Lyzr is dramatically faster

Speed isn't a single feature — it's what falls out of having the whole development surface in one tool. A developer (or a non-developer) goes from idea to deployed agent without ever leaving the platform.

SuperFlow — drag-and-drop agent design

SuperFlow is the visual canvas where agents are built. You drag in nodes for prompts, tools, memory, conditionals, and outputs, wire them together, and you have a working agent/workflow pipeline. No framework setup, no Python boilerplate, no separate orchestration service. Business users design the flow they want; the platform handles the runtime.

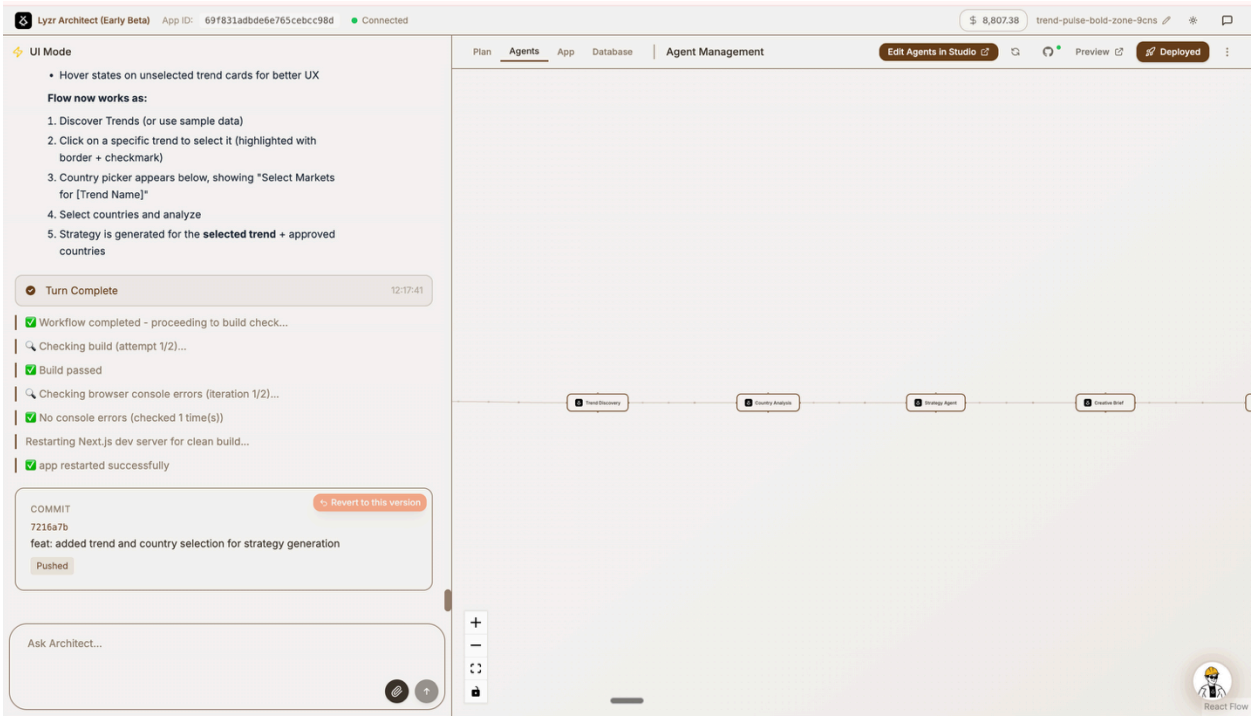
- **From blank canvas to running agent** in minutes — not the days it takes to scaffold a Vertex pipeline.
- **Logic, tools, and memory** all live in the same view, so a non-engineer can read the flow and understand what the agent does.
- **One-click deploy** ships the flow to production with versioning, rollback, and observability built in.



Architect by Lyzr — system design on a canvas

Where SuperFlow builds individual agents, Architect designs the whole system. It's a visual workspace for laying out multi-agent architectures: which agents exist, how they talk to each other through Agent Mesh, what tools and memory they share, where the guardrails sit, and how the data flows. You can sketch the architecture, generate the underlying agents directly from the design, and iterate on it as a team.

On Vertex this is a whiteboard exercise that turns into a Jira epic. On Lyzr the architecture diagram is the implementation.



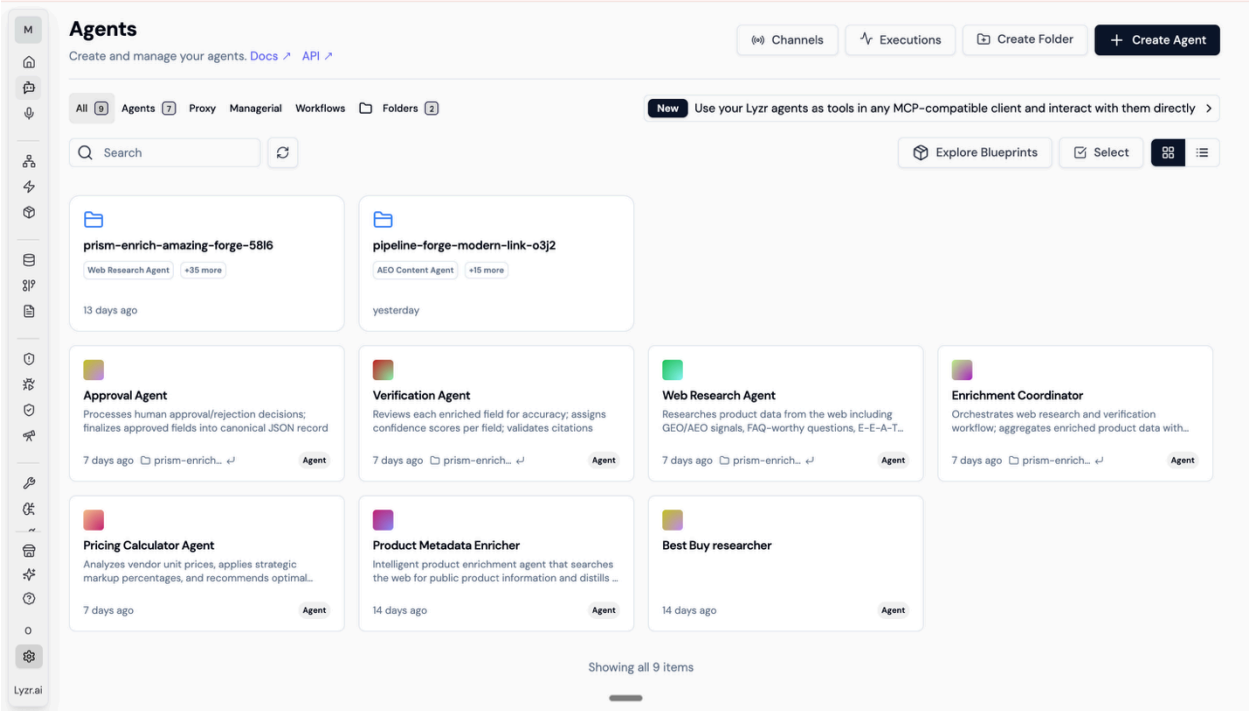
Agent Registry — one versioned home for every agent

This is the part that's typically missing from agent stacks and it's bigger than a model registry. The Lyzr Agent Registry is the central control plane for every agent in your workspace: the agent itself, its tools, its memory, its guardrails, the LLM behind it, and the connections to other agents. It replaces the conventional 'model history' page with a single, versioned source of truth for everything that runs in production.

Architect, SuperFlow, and you're a2A (agent-to-agent) deployments all reach into the same registry. An agent built once is available everywhere the same agent powering a SuperFlow node can be called by another agent through Agent Mesh, exposed as a tool, or surfaced in a custom UI, without copying configuration or redeploying.

- **Native version control:** every change to an agent is tracked, comparable, and rollback-able — something typical agent frameworks don't ship by default.

- **Reachable from everywhere:** one agent, accessed across SuperFlow, Architect, A2A flows, and external integrations — no duplication.
- **No redeploy on update:** edit the agent in the registry and every consumer picks up the new version on the next call.



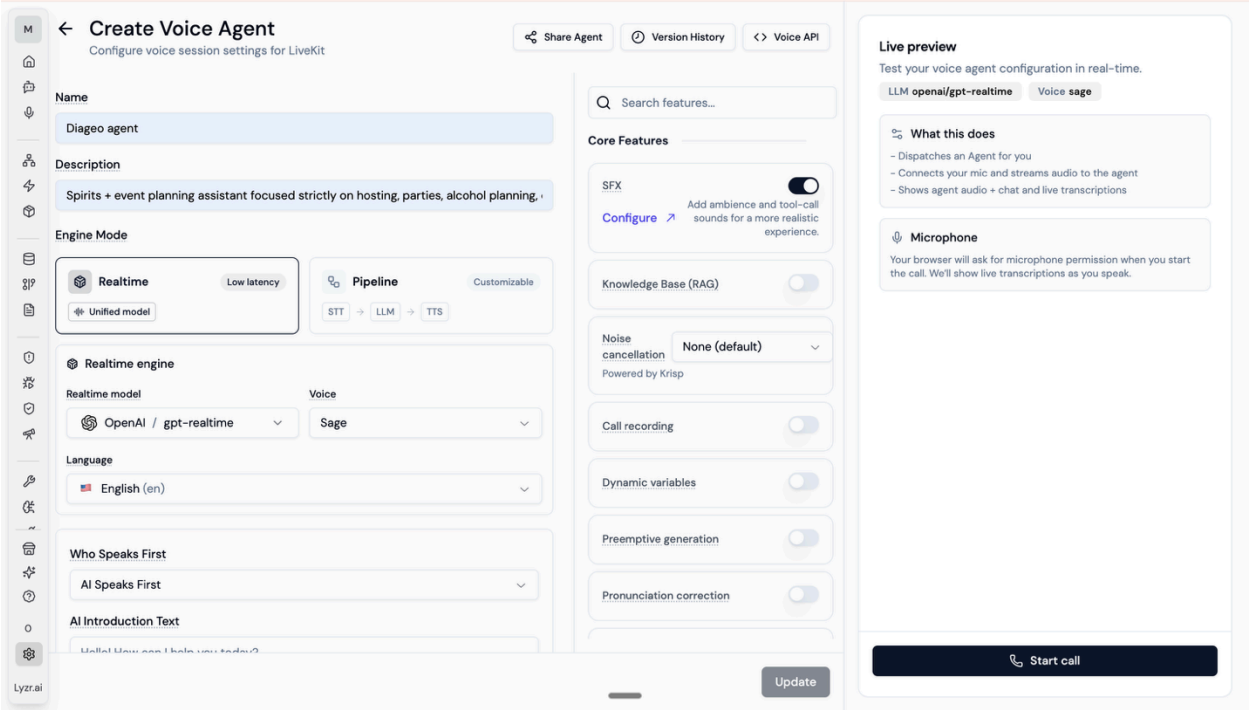
Voice Agent Builder — production voice agents without the plumbing

Voice agents are usually one of the hardest things to ship. Even with a great LLM behind them, getting a production voice agent live means stitching together speech-to-text, text-to-speech, dialect tuning, telephony, audio pipelines, latency optimization, and a scaling story for the voice servers themselves. On Vertex, this is months of integration work — and it has to be redone for every new region or language.

The Lyzr Voice System Builder collapses that into configuration. You design voice agents on the same canvas you use for any other agent, pick from a library of dialects and accents covering major regions and ethnicities, and deploy onto a managed voice-server infrastructure that scales horizontally. The platform handles the audio pipeline, the telephony, and the server orchestration.

- **Multi-dialect by default:** voice agents that speak in the local accent your customer expects — across regions, languages, and ethnicities.
- **Scalable voice infrastructure:** managed voice servers with horizontal scaling — no DevOps work, no per-region setup.

- **Same agent, voice or text:** the same Lyzr agent can be exposed as a chatbot or a voice agent without rebuilding it.



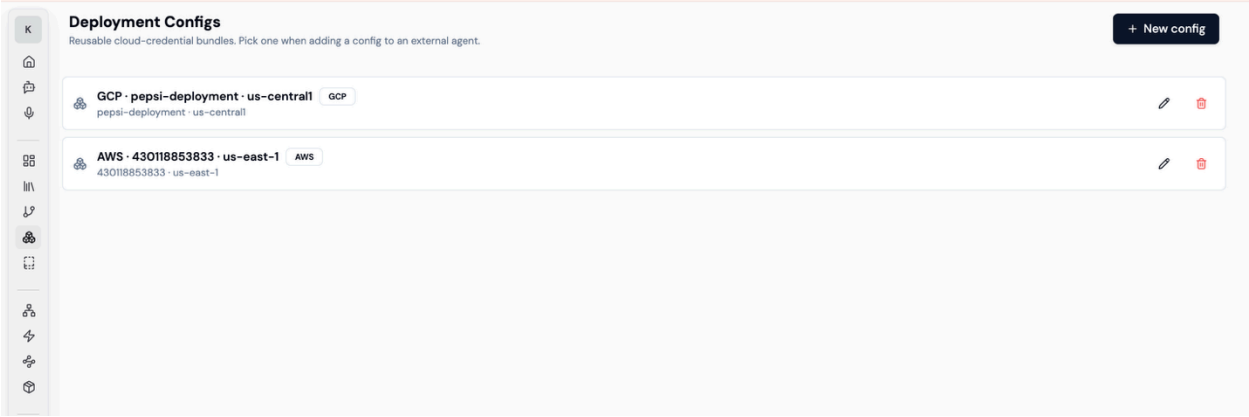
Lyzr Agent Studio — full VPC deployment, with Vertex compatibility

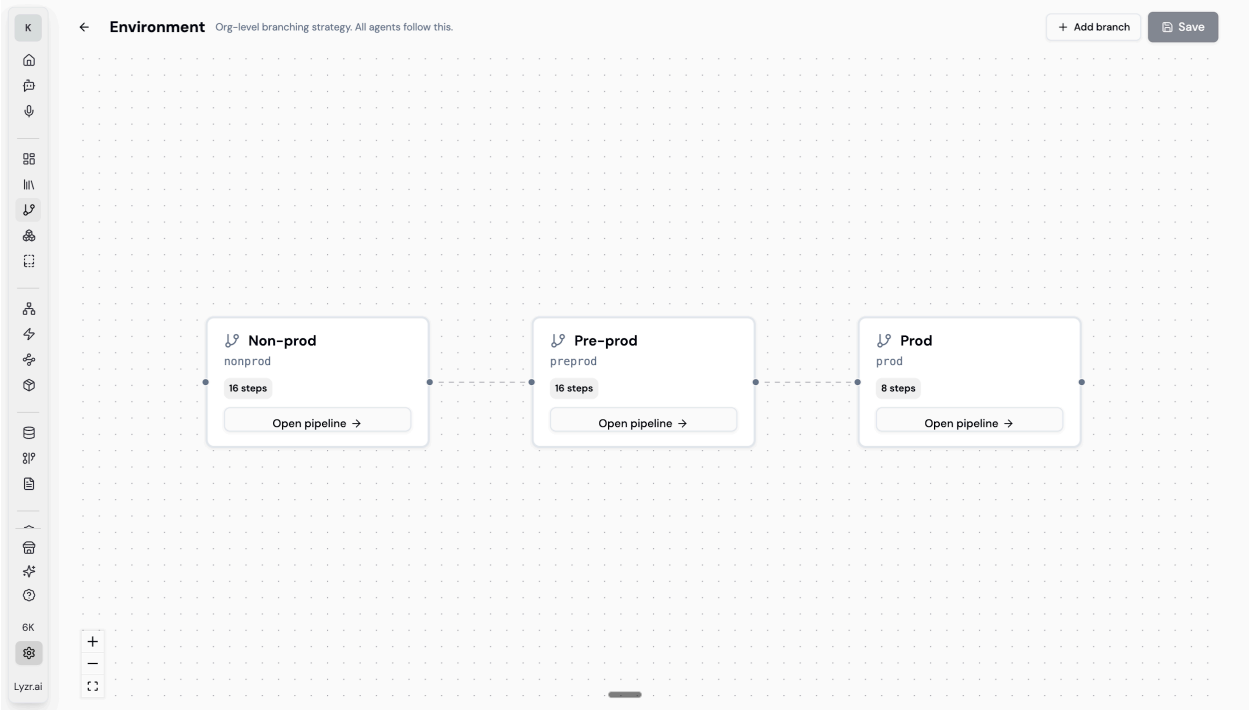
Lyzr Agent Studio is the end-to-end platform - SuperFlow, Architect, Agent Registry, Voice System Builder, Agent Simulation Engine, and Agent Improvement Engine — and the whole thing deploys inside your VPC on any major cloud, including Google Cloud Platform.

The crucial detail: this is not a Lyzr-vs-Vertex choice. Agent Studio integrates natively with Vertex models, so Gemini, PaLM, or any Vertex-trained custom model can sit underneath your Lyzr agents. Lyzr runs the agent layer — governance, simulation, orchestration, voice, registry while Vertex stays where you've already invested, on the model layer.

That's what 'no lock-in, enterprise-ready' means in practice: Lyzr runs where your security team needs it to run, on the cloud you've already standardized on, with the models you've already built.

The Lyzr Control Plane dev, staging, production, and multi-cloud failover





The Lyzr Control Plane is the deployment fabric that sits between the agents you build and the cloud they run on. Whether an agent is built natively in Lyzr Studio or developed in Google ADK, LangGraph, CrewAI, or any other framework, it gets deployed, versioned, and operated through the same control plane with full dev, staging, and production environments out of the box.

The Control Plane gives you three things that are normally several separate platforms:

- **Dev, staging, and production environments** built into the platform. Every agent moves through the same promotion path, so what you tested in staging is exactly what runs in production. No bespoke CI/CD glue, no environment drift, no surprise behaviour the first time real users hit the agent.

- **Multi-hyperscaler deployment** — deploy any agent to GCP, AWS, or both, from the same control surface. Move workloads between clouds without rebuilding the agent or rewriting the deployment scripts.
- **Native rollback and platform failover** — every deployment is versioned and instantly reversible, and if one hyperscaler has an outage you can switch the workload to another without touching the agent code. Business continuity is a configuration option, not a six-month project.

This is the layer that makes “production-grade” a real claim instead of a marketing one. Vertex AI does not provide a generalized control plane agent deployment is tied to Google Cloud and assumes you’re building inside Google’s ecosystem. With Lyzr, the cloud you run on is a deployment choice, not an architectural commitment.

Proxy Agents bring your existing agents into the Lyzr ecosystem

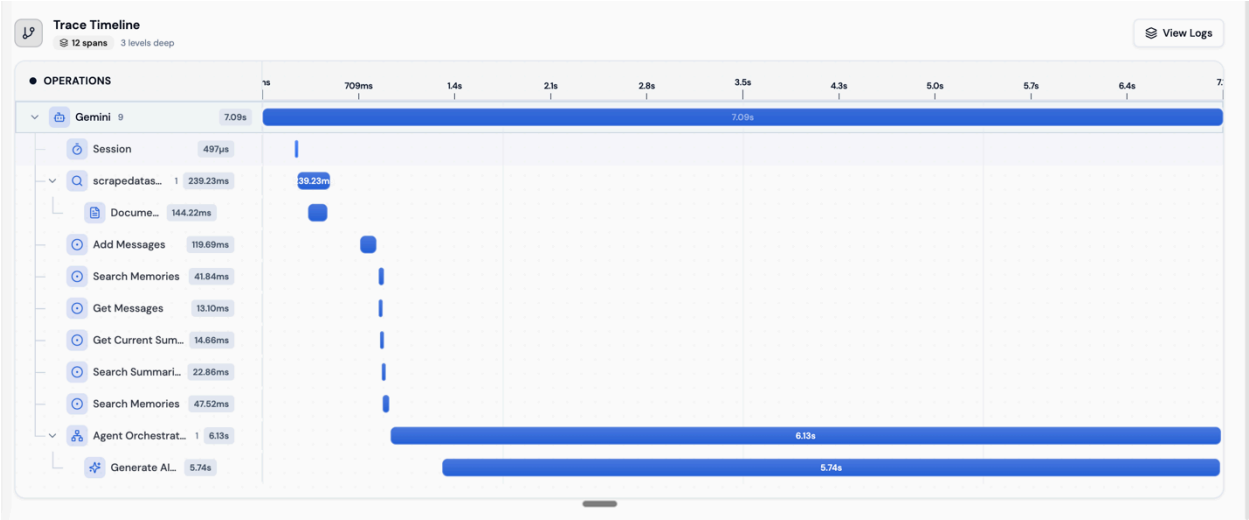
If your team has already built agents on LangGraph, Google ADK, CrewAI, or any other framework, you don’t have to throw them away to use Lyzr. Proxy Agents let you migrate or reference those existing agents directly inside the Lyzr workspace, where they show up as first-class citizens in the agent nursery alongside agents built in SuperFlow or Architect.

A proxied agent inherits everything the framework you originally used didn’t ship: the Lyzr governance layer, the simulation engine, the observability stack, and Agent Mesh so it can communicate with native Lyzr agents, share memory, and be orchestrated like everything else in the platform.

- **Reference, don’t rewrite** keep your existing LangGraph, ADK, or CrewAI agent running and call it through Lyzr without rebuilding it from scratch.
- **Full migration when you’re ready** — when you want the rest of the platform, lift the agent into the Lyzr framework natively and pick up guardrails, simulation, traces, and the Control Plane in one move.
- **Work alongside Lyzr-native agents** — proxied agents talk to native agents through Agent Mesh, so a hybrid team of “legacy” and “new” agents operates as one coherent system.

This is the practical version of “no vendor lock-in.” You’re not asked to choose between your existing investment and a new platform both can coexist, and you can move at your own pace.

Explicit traces and observability token-level visibility, cost optimization built in



Most agent platforms give you logs. Lyzr gives you explicit, structured traces of everything that happens inside every agent every model call, every tool call, every memory read, every guardrail check with token counts and cost attached at each step. The traces and observability shipped by Lyzr are state-of-the-art and built specifically for the realities of multi-agent production systems.

The result is something agent teams almost never have: the ability to see exactly which agent, which tool call, and which prompt is driving cost — and where in a multi-agent flow a failure originated.

- **Per-agent and per-step token attribution** see exactly how many tokens a specific agent or tool call consumed in a single conversation, and across the fleet over time.
- **Pinpoint failure analysis** when something goes wrong in a multi-agent system, the trace shows which agent failed, which tool returned what, and why instead of forcing your team to reconstruct it from scattered logs.
- **Built-in token and cost optimization** the platform surfaces obvious wins (over-long prompts, redundant tool calls, expensive models on cheap tasks) so you can right-size every agent without rebuilding it. Cost stays configurable and predictable, even at scale.

This level of observability is the reason customers can keep agent costs predictable in production rather than discovering a runaway bill at the end of the month.

A modular ecosystem swap any layer, keep the rest

Because Lyzr Studio is built from the ground up as a modular agent framework rather than a bundled monolith, every layer of the stack model, memory, guardrails, vector store,

orchestration, observability has a clean contract, and any of them can be swapped out without re-architecting the rest.

- **Models are interchangeable** use a Lyzr-managed model, a Vertex model, OpenAI, Anthropic, an open-source model in your VPC, or a mix of all of them. The agent code doesn't change.
- **Guardrails are interchangeable** use Lyzr's native guardrails, Vertex's safety layer, NeMo Guardrails, or any third-party guardrail system. Drop it in, the rest of the platform stays the same.
- **Memory is interchangeable** use Lyzr's world-class memory system, or bring your own. The same applies to the vector store, the auth provider, and the trace backend.

This is what makes "responsible AI and guardrails in place" compatible with "configurable for your stack." You get the compliance, safety, and cost-control story enterprises need, without being forced into a single vendor for every layer of the agent.

Integration and tools , built for speed

Most of the time spent shipping an enterprise agent is spent on plumbing: connecting to the CRM, the data warehouse, the help-desk, the auth system. Lyzr ships with 200+ pre-built integrations, and adding a new one is a no-code action point at an API, give it a name, and it's now a tool any agent can call.

- **200+ business tool integrations** out of the box — Salesforce, HubSpot, Slack, Snowflake, ServiceNow, LinkedIn, and more.
- **Custom tools without code** — add any REST API as an agent tool through a configuration UI, no SDK or wrapper needed.
- **Agent Mesh** lets agents call each other natively, with shared memory and routing — no message-bus glue to write.
- **Reusable across agents** — a tool added once shows up in every agent's palette in the same workspace.

A genuinely modular architecture

The Lyzr platform is built as a set of composable modules — agents, tools, memory layers, guardrails, simulation, observability, runtime — each with a clear contract and each independently swappable. You're never locked into a stack-wide upgrade or a monolithic release.

- **Pick what you need:** use just the agent runtime, or the full stack including governance and UI builder.
- **Bring your own:** your LLM, your vector DB, your auth provider — modules plug in without forcing the rest of the stack to change.

- **Independent upgrades:** swap the simulation engine or the guardrails layer without redeploying every agent.

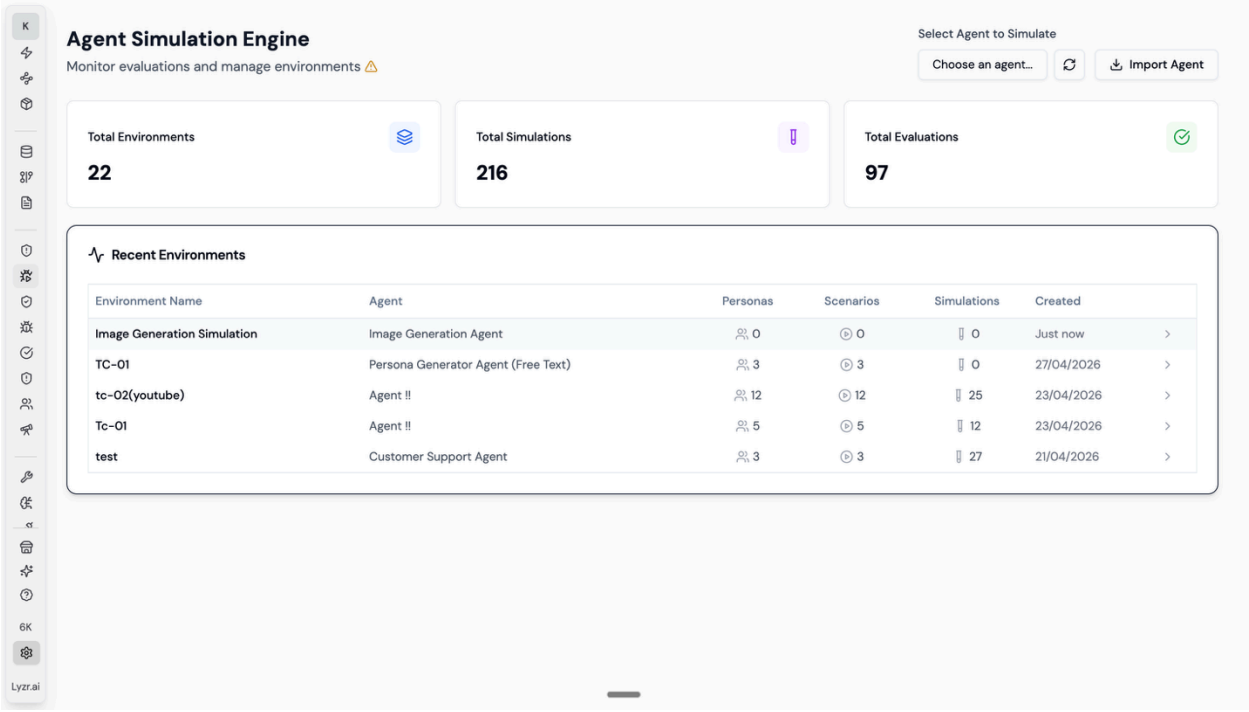
This is the opposite of a Vertex deployment, where the value comes from being inside the Google Cloud ecosystem and the cost is being inside the Google Cloud ecosystem.

Simulation and continuous improvement — built in

This is the single biggest reason Lyzr projects make it to production while industry-average AI projects don't.

Every agent built on Lyzr is run through the Agent Simulation Engine before it goes live. The engine generates thousands of scenarios — happy paths, edge cases, adversarial inputs, domain-specific stress tests — and tracks regression on every update. Failures get caught in simulation, not by your customers.

- **Pre-deployment scenario coverage** across thousands of generated cases per agent.
- **Domain-specific test suites** for BFSI, healthcare, telco, sales, and more — included with the platform.
- **Regression on every update,** so a prompt or tool change can't silently break previously-passing flows.



Agent Improvement Engine — closing the loop

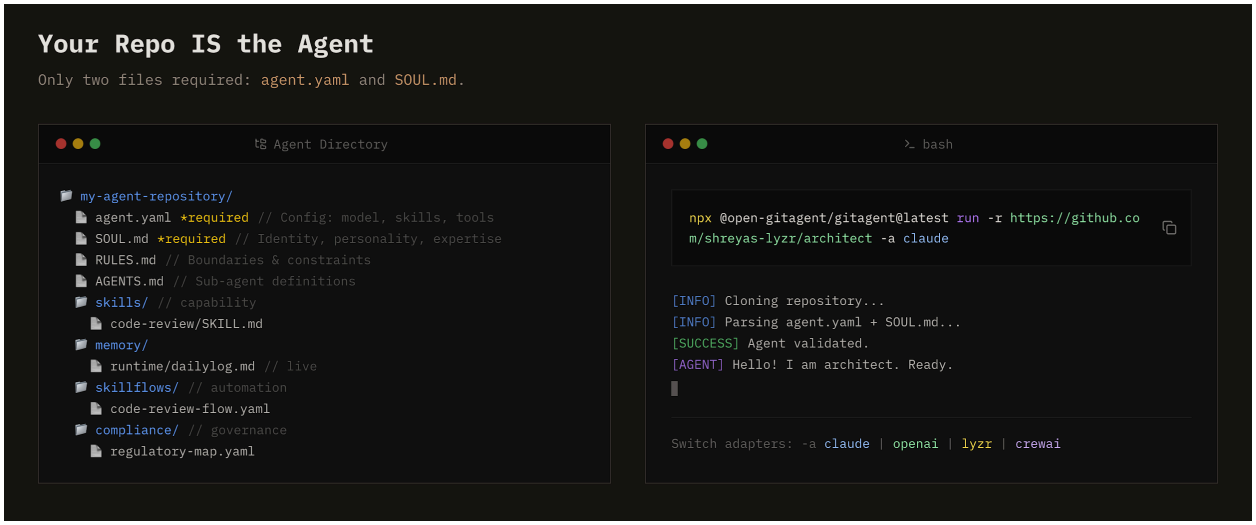
The Agent Improvement Engine is the production-side counterpart to the Simulation Engine. Real production traces, user feedback, and edge cases that surface in the wild get fed back into

the simulation suite automatically — every agent gets better over time without a human in the loop. Failures discovered in production become tests that protect the next version.

On Vertex, this loop has to be assembled manually from logs, evaluation pipelines, and ML retraining jobs. On Lyzr it's a feature you turn on.

> **Why this matters**
>
> Vertex AI has no native simulation environment. Failures surface in production, where they cost trust, money, and re-implementation cycles. On Lyzr, agents don't ship until they pass — and they keep getting better automatically once they're live.

OpenGap: *your repository becomes your agent*



The open standard for defining, versioning, and running AI agents natively in git. Version-controlled config that exports to Claude Code, OpenAI, Lyzr Agent, Chimera, NanoBot, CrewAI, OpenClaw, and Agents SDK, with zero code changes

Built so non-developers can ship real agents

On Vertex, building a useful agent requires deep expertise — model selection, fine-tuning, vector setup, orchestration, deployment infrastructure. A team without ML engineers and a serious DevOps function will struggle to make it past a prototype.

Lyzr is built for the opposite reality. CXOs, product managers, and operations leads can build, test, and deploy real agents through SuperFlow and Architect — supported by the embedded Pro-Serv team, which ships the customer's first three agents alongside them. The platform handles the runtime, the security, the scaling, and the observability.

- **No-code design** covers the full agent lifecycle, not just prototyping.
- **Pro-Serv onboarding** gets the first three agents into production with the customer team — knowledge transfer, not dependency.
- **Pre-built agent marketplace** covers the most common use cases out of the box, so most teams start by configuring an existing agent rather than building from scratch.

Business impact comparison
A simplified view of where each platform stands on the dimensions that drive ROI and time-to-value.

Feature	Lyzr	Google Vertex AI
Ease of Use	✓	✓
Pre-Built Solutions	✓	✓
Industry-Specific Agents	✓	✗
Agent Simulation Engine	✓	✓
Time to Deployment	✓	✗
Pricing Structure	✓	✓
Integration Versatility	✓	✓
Data Ownership	✓	✓
Customization	✓	✓
Community & Support	✓	✓
Agent Communication Tech	✓	✗
Adoption Speed	✓	✗
Production Ship Rate	✓	✗

And the same comparison in detail:

Feature	Lyzr	Google Vertex AI
Target Audience	CXOs, devs, product managers	Developers, data scientists, enterprises
Ease of Use	No-code, business-friendly platform	Advanced platform requiring technical expertise
Pre-Built Solutions	200+ pre-built agents across BFSI, healthcare, sales, marketing	None; requires custom development
Industry-Specific Agents	BFSI, healthcare, telco, real estate, SaaS — shipped, not templated	None; custom-built models only
Agent Simulation Engine	Thousands of scenario simulations, domain-specific test suites, regression testing before every deployment	No native simulation environment
Time to Deployment	Rapid with pre-built agents	Longer; custom model development required
Pricing Structure	Free to start; flat-fee/month plus usage-based	Pay-as-you-go via Google Cloud pricing
Integration Versatility	200+ business tool integrations	Seamless within Google's ecosystem
Data Ownership	Full control — model, cloud, and data agnostic	Managed by Google Cloud infrastructure
Customization	Pre-built and fully customizable agents	Fully customizable but requires deep technical expertise
Community & Support	Active community plus professional services	Google Cloud support ecosystem
Agent Communication Tech	Cross-agent communication via Agent Mesh	No native cross-agent communication

Feature	Lyzr	Google Vertex AI
Adoption Speed	Quick with pre-built agents and assisted onboarding	Slower due to technical learning curve
Production Ship Rate	85% of Lyzr projects reach production	Industry average under 30%

Developer's perspective

Many of the foundational capabilities — security baselines, scalability, hosting basics — exist on both platforms. The table below filters to the developer-side capabilities Lyzr ships natively that Vertex AI does not, so the meaningful differences are visible at a glance.

Capability	Lyzr	Google Vertex AI
Native Agent Simulation Engine	✓	✗
Pre-built agent marketplace	✓	✓
Cross-agent communication (Agent Mesh)	✓	✗
No-code visual agent design (SuperFlow & Architect)	✓	✗
Agent Registry with native version control	✓	✗
Voice System Builder with multi-dialect deployment	✓	✗
Agent Improvement Engine (continuous learning)	✓	✗
On-premises / hybrid / VPC deployment options	✓	✗

And the full developer-side comparison in detail, including the dimensions where both platforms compete:

Feature	Lyzr	Google Vertex AI
Hosting Options	Cloud, on-premises, hybrid including full deployment in your VPC	Fully cloud-based with Google Cloud
LLM Integration	Model-agnostic; supports any LLM no lock-in	Google Gemini/PaLM and select external models
Agent Simulation	Built-in simulation engine edge cases, adversarial inputs,	No native simulation; failures surface in production

Feature	Lyzr	Google Vertex AI
	regression testing on every update	
Professional Services	Pro-Serv team plus global partners; embedded support to ship your first 3 agents	Enterprise-level via Google Cloud support
Customer Support	Chat, call, and community help	Google Cloud support tiers, 24/7 for enterprise
Integration Capabilities	200+ tool integrations	Tight integration with Google Workspace and external APIs
Data Privacy	Private cloud and on-premises deployments; GDPR, CCPA, HIPAA compliant	SOC 2, ISO/IEC 27001 compliant via Google Cloud
Agent Communication	Cross-agent communication via Agent Mesh	No native cross-agent communication
Scalability	Enterprise-grade with usage-based pricing	Scalable via Google Cloud infrastructure
Security Standards	SSO, RBAC, audit logs from day one	SOC 2, HIPAA, ISO certifications
Agent Marketplace	Pre-built agent marketplace	No marketplace; custom-built only
Workflow Automation	Automates workflows across 200+ tools	Supports automation via APIs and pipelines

Capability Comparison

CAPABILITY	lyzr	Gemini	Bedrock Agentcore
Cross-cloud + on-prem deployment	✓ AWS, Azure, GCP, on-prem, air-gapped — same platform	△ GCP; on-prem & air-gapped via Google Distributed Cloud	✗ AWS only.
Model neutrality	✓ Any provider, any host, BYO + on-prem models	△ Multi-model via Vertex / Model Garden — runtime is GCP-bound	△ Multi-model via Bedrock catalog — runtime is AWS-bound
Git-native agent versioning	✓ GitAgent — branching, diffs, rollback per agent	△ Memory Revisions (preview) + Git/CI for the rest	△ Runtime versions + Memory Branching, AWS-only
Pre-built enterprise blueprints	✓ 60+ first-party blueprints, deploy in minutes	△ Agent Garden + partner agents in GCP Marketplace	△ AWS Marketplace AI Agents — mostly partner-led
Live knowledge base	✓ Auto-syncing live KB across source systems	△ Drive / SharePoint / Salesforce connectors — scheduled sync	△ Bedrock KBs — scheduled ingestion (S3, SharePoint, Confluence)
Database connectivity	✓ Native DB connectors + natural-language query	△ Native to BigQuery, AlloyDB, Cloud SQL; MCP Toolbox for NL	△ Native MCP for Aurora; broader DBs via Lambda
Voice + telephony	✓ Native voice agents with built-in telephony	△ Separate SKU — Gemini Enterprise for Customer Experience	✗ Assemble Connect + Lex + Polly / Nova Sonic + Lambda
Auto-hardening post-eval	✓ Automated safety, drift, and jailbreak coverage	△ Model Armor built-in; broader drift / eval is separate	△ Bedrock Guardrails + AgentCore Policy as separate SKUs
Non-developer access	✓ Architect — visual builder for full apps + frontend	△ Agent Designer — agents only, no frontend (GCP console)	✗ Code-first SDK; AWS expertise required
Hyperscaler-native depth	✓ Production connectors to all major clouds	✓ Deepest depth in Workspace + BigQuery	✓ Deepest depth across the AWS service mesh
Enterprise security & IAM	✓ RBAC, your IdP, SOC 2, HIPAA, GDPR	✓ GCP IAM, CMEK, EKM/HSM, VPC-SC, audit	✓ AWS IAM, KMS, VPC/PrivateLink, CloudTrail
Cloud + model lock-in	✓ None — fully portable, exit any time	✗ Platform layer (memory, registry, gateway) is GCP-bound	✗ Platform layer (runtime, identity, gateway) is AWS-bound

Seven reasons enterprises pick Lyzr over Vertex AI

1. The Lyzr Control Plane and SuperFlow — build fast, deploy anywhere

SuperFlow is the visual canvas where agents are designed, tested, and shipped without writing framework code what takes weeks of scaffolding on Vertex takes hours on Lyzr. The Control Plane then takes that agent through dev, staging, and production with one-click deploy to GCP, AWS, or both, native rollback on every release, and instant failover if a hyperscaler goes down. Vertex deployment is tied to Google Cloud; on Lyzr, the cloud is a configuration choice, not an architectural commitment.

2. Tailored, shipped agents — not templates

200+ pre-built agents across BFSI, healthcare, telco, sales, and marketing. These are production-grade agents, not starter templates. Most customers go live by configuring an existing agent rather than building from scratch and the same agents are usable by CXOs and product managers, not just engineers.

3. The Agent Simulation Engine

Every agent is tested against thousands of scenarios edge cases, adversarial inputs, domain stress tests before it goes live. The Agent Improvement Engine then feeds real production traces and failures back into the simulation suite automatically, so agents get better without a human in the loop. 85% of Lyzr projects reach production vs. an industry average under 30%. Vertex has no native simulation environment.

4. Explicit traces and token-level observability

Lyzr ships structured traces of every model call, tool call, memory read, and guardrail check with token counts and cost attached at each step. You can pinpoint exactly which agent is driving cost, where a multi-agent flow failed, and which prompts to optimize, without rebuilding anything. Vertex gives you logs; Lyzr gives you the per-agent, per-step visibility that keeps a production agent bill predictable instead of a month-end surprise.

5. HybridFlow for accuracy

HybridFlow combines LLMs with classical ML agents to deliver Six Sigma reliability manufacturing-grade precision on outputs, not just plausible-sounding text. Standard Vertex deployments don't ship this kind of hybrid reliability layer.

6. Enterprise-ready from day one

On-premise, VPC, or cloud deployment — including GCP. SSO, RBAC, audit logs, and Git-native agent management included from the start. The whole Agent Studio SuperFlow, Architect, Agent Registry, Voice Builder, Simulation Engine, and Improvement Engine runs inside your VPC, with native Vertex model integration if you want to use Gemini as the backing LLM.

7. No lock-in — you own your IP

Model-agnostic, cloud-agnostic, data-agnostic. Every layer is swappable: model, guardrails, memory, vector store. GitClaw publishes the open standard for how enterprise agents are defined, versioned, and operated, and GitAgent + Proxy Agents bring LangGraph, ADK, or CrewAI agents into Lyzr without rewriting them. Agents live in your repo, get reviewed like code, and travel with you. You don't have to choose between Lyzr and Vertex — Lyzr can run on GCP and use Vertex models. The difference is that the agent layer, the governance, and the simulation are yours, on infrastructure you control.

Where this lands

This isn't a question of Lyzr instead of Vertex. Vertex AI is excellent at the model layer, and many teams have invested heavily in it. Lyzr Agent Studio sits one layer above taking those models (or any others) and turning them into governed, simulated, production-grade agents through a unified platform that any team can build on.If your team has the resources to assemble a custom AI stack on Google Cloud and the time to wait for it to mature into production, Vertex AI alone is a credible choice. For everyone else including Vertex teams that want a faster development loop and a unified governance story on top of their existing Vertex

investment Lyzr replaces the 14-tool stack with a single platform. It ships agents in days, runs entirely inside your VPC (including on GCP), integrates natively with Vertex models, and tests every release before customers see it.That's the gap, and that's what Lyzr fills.