The Architect's Guide to Event-Driven Agentic AI

Designing resilient, real-time systems for next-generation autonomy

Jesse Menning
Solution Architect | Solace

solace.

Agentic AI—collaborative systems of autonomous agents that can reason, plan, and act—is rapidly transitioning from theoretical exploration to real-world adoption. As enterprises seek to orchestrate LLMs, APIs, and human workflows, the need for scalable, event-driven infrastructure has become urgent.

Agentic AI systems, made popular by frameworks such as AutoGPT, CrewAI, LangChain, MetaGPT, and Solace Agent Mesh, represent a significant leap from traditional AI and generative models. Unlike passive models that respond only when prompted, agentic AI systems are autonomous, proactive, capable of coordinating complex tasks across diverse environments, and of involving humans when necessary.

Enterprise architects, platform engineers, and AI innovation teams are increasingly being asked to design the infrastructure that supports these systems. This guide provides a comprehensive architectural perspective on how to implement agentic AI at scale, with an emphasis on real-time responsiveness, scalability, and reliability. The guide does not recommend specific technologies, but it surfaces key principles of event-driven architecture (EDA) as a critical foundation for enabling autonomy in complex, distributed environments.

Introduction to Agentic AI

Agentic AI represents a shift from task-specific machine learning to systems capable of long-running, context-aware decision-making. These systems aren't just generating text or predictions—they're initiating actions, orchestrating steps, reacting to changing conditions, and learning from outcomes. What sets agentic AI apart is its ability to dynamically set goals, plan actions, and delegate to external tools and systems in pursuit of those goals.

1. An event/request comes into the orchestrator via gateway
2. Orchestrator uses an LLM to divide the request and determine which skills/agents are required
3. Orchestrator delegates subtasks to appropriate agents
4. Agent execute subtasks to appropriate agents
5. Agent execute subtasks and return information to orchestrator
6. Orchestrator ensures the answer is complete, summarizes it, and sends to the requestor

Example Use Cases

Here are a few ways agentic AI systems are demonstrating impact:

- **In customer service**, intelligent agents are resolving inquiries and learning from each interaction. Financial institutions are employing agents to autonomously collect, analyze, and synthesize market data.
- **Real-Time Operational Intelligence & Analytics**: Creating an AI layer on top of operational systems such as order management or inventory management that allows for natural language querying against those systems, rendering visualizations and dashboards, finding issues and root causes, unusual patterns, and resolving issues.
- **Customer Support Enhancement**: Integrating agentic frameworks with CRM systems to provide service representatives with real-time customer insights, enhance ticket context and recommendations for resolutions, summarization, and AI-based CSAT scores.
- **Knowledge Management & onboarding**: Building enterprise knowledge platforms that can access, analyze, and synthesize information across organizational silos and help onboard employees to the new system.
- **Employee Onboarding Automation**: Transforming manual onboarding processes by responding to hiring events with orchestrated multi-agent workflows spanning IT, finance, and facilities. This event-driven approach ensures consistent execution of onboarding tasks, allocation of resources and granting acccess while eliminating manual checklists and reducing administrative overhead.

Key Business Requirements for Agentic AI

No one can predict where agentic AI will be a year from now, but it's safe to say it will be very different. We will continue to see disruptions like DeepSeek, consolidation around various communication protocols, enforcement of new regulations, and the entry of new players into the market.

At the same time, enterprises will expand the use cases they entrust to AI, making it critical to create an architecture that 1) Can deal with sudden, radical changes without being reworked from the ground up, and 2) Establishes a flexible foundation that can incorporate and embrace whatever innovation brings to the table.

The ecosystem supporting these innovations includes rapidly maturing frameworks like LangGraph, Crew AI, Solace Agent Mesh and MetaGPT. Each brings distinct approaches to agent planning, memory, and action. However, these frameworks alone are not enough—architects must create environments in which agents can access information, communicate, and evolve safely and reliably. The complexity lies not in the agent logic, but in orchestrating the ecosystem around it.

The following are essential elements to both efforts:

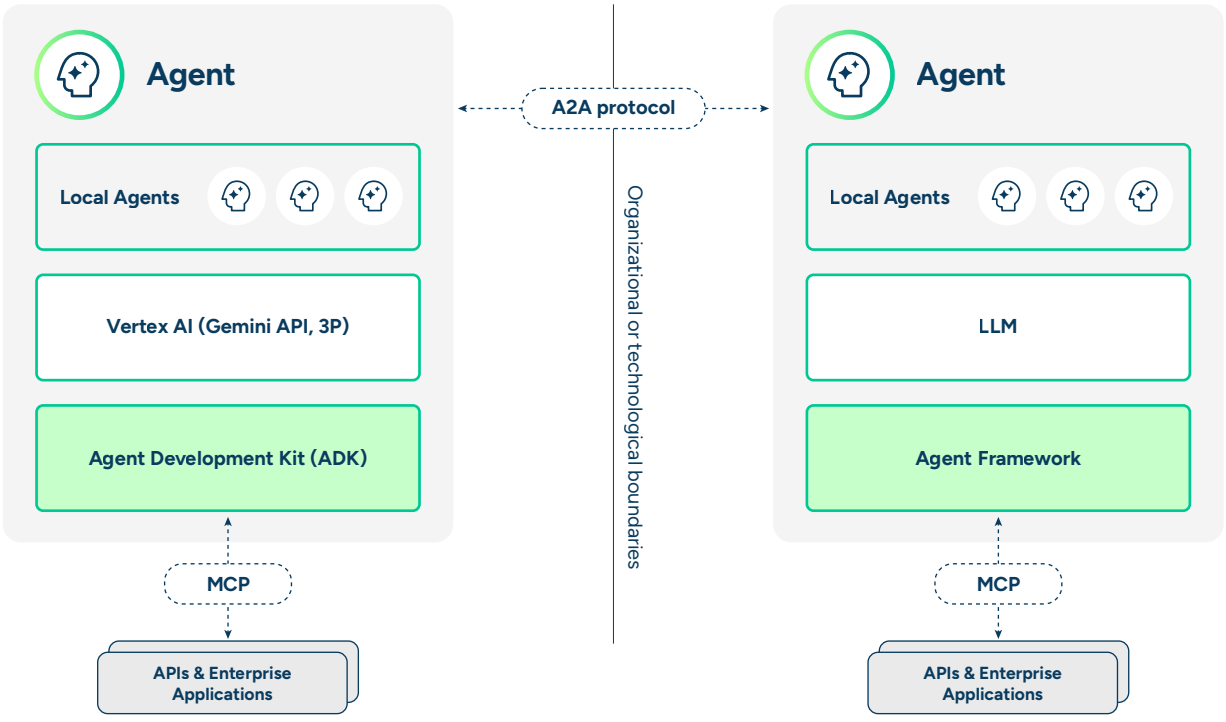
Maintaining Openness

As companies move from single-use LLMs to networks of intelligent agents, they need those agents to talk to each other, ideally in a way that's standardized so they don't get locked in to one approach or protocol, or mediate communications in the middle.

Two protocols that have captured attention in the agentic space (although there is much work to be done, and other protocols may yet emerge):

- **Google's Agent2Agent (A2A)** protocol gives businesses a standardized way to connect AI agents across tools, vendors, and workflows.
- **Model Context Protocol (MCP)** developed by Anthropic, in turn gives agents the ability to use tools to complete their task. MCP often serves as a bridge to traditional APIs defined by OpenAPI, another open standard.

These two standards are not rivals; rather, they work together to solve different challenges in agentic AI systems as shown here:



Fast Recognition, Immediate Action

Agents must respond to changes in the environment, user input, or system state as they happen—not minutes later. The ability to process and respond to information in near-real-time underpins the value proposition of autonomous systems.

Real-time information can initiate and augment the behavior of AI agents in three key ways:

- **Trigger Agentic AI Workflows:** Real-time information can trigger an agentic AI system. If a bus encounters a flat tire, the agentic system can be notified and immediately begin making mitigation recommendations.
- **Enrich Decision Making:** Streaming data to continuously populate vector and traditional databases with up to the second information enhances the relevance and accuracy of RAG queries.
- **Accelerate Action and Response:** Up-to-date context, along with the ability to interact with a variety of information sources and enterprise applications, enables agents to respond more quickly and effectively.

Unified Intelligence Across the Enterprise

Agentic AI systems must be deeply integrated across domains. Agents need access to real-time data streams, document repositories, structured records, and signals from business systems. This cross-domain connectivity enables agents to act with full awareness of business context.

For example, to deal with a public transit bus getting a flat tire on a busy day, you might need to access and interact with a variety of information sources like maintenance records, weather, historical data, rider sentiment and more.

Generative AI remains expensive and slower than other enterprise components, which makes it essential that the resulting insights be distributed widely, not in a silo for one particular team. The mitigation plan for the bus with a flat tire would be important for many departments to know about, like maintenance, staffing, public communication and likely more.

Modular Design for Maximum Flexibility

The architecture should enforce loose coupling between components. Memory services, planners, toolchains, and output processors should be modular and swappable, allowing the system to evolve over time. This modularity makes it possible to adopt new and improved frameworks, LLMs, and components into your agentic framework, and to embrace new protocols or agent strategies without disrupting the system.

Agentic AI is powerful, but not always necessary. For many workflows, simpler rule-based automation or APIs remain the best fit. By enforcing modularity, organizations retain the ability to incrementally adopt agentic patterns where their adaptability and intelligence offer clear business value. Simply put, modularity is what allows agentic AI systems to evolve with the AI ecosystem, rather than being rebuilt for every innovation.

Elastic Scale for Global and Growing Reach

Some agents may be short-lived and stateless, while others maintain context across long-running workflows. Supporting both models requires infrastructure that can scale elastically, coordinate across geographies, and maintain consistency when needed.

Protection of Assets, Governance of Process

Security and access control must be considered from the outset. Agents making autonomous decisions must operate within strict boundaries. Authentication, authorization, and policy enforcement need to be baked into every level of the architecture. Delivery guarantees and logging ensure that decisions are reliable, observable, and auditable.

As agents evolve and make more autonomous decisions, it becomes critical to ensure every action is traceable—who made it, why, and based on what data. This makes functions like agent versioning, decision chain auditing, and data lineage essential for regulatory compliance and enterprise trust.

Poorly scoped permissions or unverified inputs can result in cascading failures, so enterprises must extend patterns like zero-trust and secure-by-default into agentic AI environments to safeguard systems and data.

Robustness to Ensure Business Continuity

Resilience must be engineered from the ground up. This includes handling failure gracefully—whether through retry logic, dead-letter queues, fallback agents with default behaviors, or escalation to human decision-makers. Robustness isn't just about uptime; it's about ensuring agents can fail safely without disrupting business processes.

Because agents operate based on probabilistic reasoning and real-time data, their behavior can be unpredictable. Traditional test cases and deterministic logs often fall short. Observability must focus on capturing the context, rationale, and path taken to arrive at decisions.

Agentic AI introduces probabilistic and context-dependent behavior that defies traditional debugging. Ensuring continuity means building systems that capture not only what went wrong, but why—via contextual logging, decision rationale, and agent observability.

The Advantages of Approaching Agentic AI with an Event-Driven Approach

Agentic AI systems thrive in environments that are dynamic, distributed, and data-rich. Event-driven architecture (EDA) provides a natural foundation for these systems, enabling real-time responsiveness and loose coupling at scale. Event-driven integration turns conventional integration architecture inside-out—from a centralized system with connectivity and transformation in the middle to a distributed event-driven approach, whereby integration occurs at the edge of an event-driven core. Here are the high-level advantages of EDA:

- **Improve scalability, reliability and useful life of legacy applications** and systems of record by enabling them to send and receive information in real-time.
- **Accelerate innovation** by making it easy to incorporate new apps, cloud services and IoT devices into existing business processes.
- **Simplify the design of your growing system** by replacing point-to-point integration that relies on synchronous communications with more flexible asynchronous and one-to-many interactions.

So how does EDA benefit agentic AI systems, specifically?

Loose Coupling

EDA enables AI agents to communicate via events or messages rather than point-to-point calls—it's a seemingly simple shift that makes a profound difference in how systems interact and evolve by decoupling them, i.e. replacing tight relationships built on request/reply interactions with flexible ones that revolve around asynchronous communications.

When agents are loosely coupled like this, each can be developed, deployed, and modified independently without risking system-wide disruptions. Teams can work on specialized agents without needing to understand the entire ecosystem. This independence is invaluable when you're dealing with complex, multi-agent systems where different specialized teams own different agent types written using various agentic frameworks, as they can each build their agents on their own timeline and project plan.

An event signaling a completed transaction, a change in account status or a user's location can trigger one or many agents to act—without direct orchestration. As agents move from request/reply where humans are an essential part of most flow, to increasingly autonomous and automated behavior, this ability to add new agents to the mix will become more crucial, allowing multiple actions to occur simultaneously.

Fault Resistance

As agentic AI systems transition from experimental prototypes to mission-critical production services, scalability and reliability become non-negotiable requirements. EDA excels at addressing both challenges.

Event-driven systems built on the right kind of event broker scale horizontally by design. Need more processing power for a particular agent type? Simply add more instances t hat consume from the same event queue. Experiencing sudden load spikes? No problem, your agents are protected with the broker acting as a shock absorber. Events can buffer in the queue until resources are available to process them. This approach potentially allows agent networks to scale from handling dozens to thousands of requests without architectural changes.

This same architecture creates natural fault isolation boundaries that contribute to system resilience. If one agent experiences issues or goes down entirely, messages intended for it can queue up or be rerouted, while the rest of the system continues operating normally. When the agent recovers, it can process its backlog of events. This isolation helps prevent cascading failures and contributes to making the overall system more robust against the inevitable issues that arise in complex deployments.

An event mesh extends this concept across deployment environments. Whether agents operate in the cloud, at the edge, or on-premises, the mesh ensures they can receive relevant events and contribute actions into the broader system. This enables scalable, location-agnostic collaboration across networks.

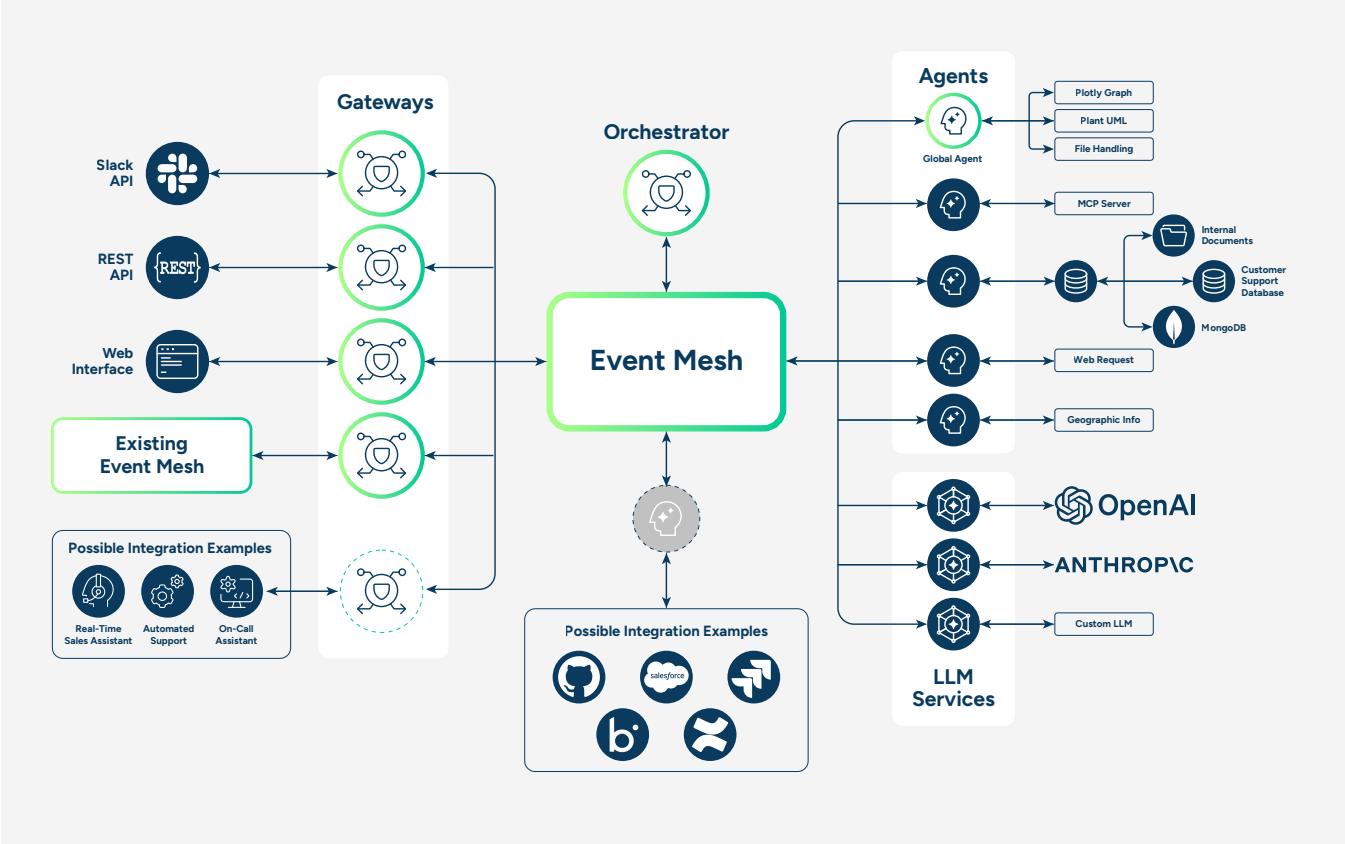
Benefits That Match Agentic AI

Coordination through events enables agents to evolve workflows organically. Instead of hardcoded process maps, agents respond to conditions as they arise, initiating cascades of activity across domains. When events carry rich metadata—such as priority, origin, or tags—agents can subscribe selectively and prioritize their responses accordingly.

Consider an example where a customer service agent detects a billing anomaly. That agent emits an event with a detailed payload. A summarizer agent subscribes to anomaly reports and creates a synopsis. A translator agent reformats the report for global teams. A third agent handles outbound communication, routing the message via Slack, email, or SMS depending on recipient preferences. Each step is autonomous, modular, and traceable.

Reference Architecture for Event-Driven Agentic AI

To enable this style of interaction at scale, we propose a reference architecture that combines event-driven principles with collaboration between AI agents



Agents

The heart and soul of agentic AI is obviously agents—each agent performs a specific task, such as language understanding, context retrieval, task planning, or API execution. These agents operate independently, can be deployed and scaled on demand, and can be versioned separately. They are connected via an event mesh that handles routing, filtering, and persistence.

Way to Ingest Triggers (Gateways)

Agents can be triggered in several ways. A human user may initiate an action via a chatbot or submitting a web contact form. So might a change in a CRM record, an update in an ERP system, or a temperature spike detected by an IoT sensor. Agents also respond to time-based and conditional triggers. This multi-channel initiation model ensures agents are not limited to a single interaction mode—they are part of the business fabric.

Real-Time Data Distribution (Event Mesh)

Once triggers have kicked off a request, you need decoupled, event-driven communication across clouds and systems so your agentic AI projects can scale from POC to production. Such a system enables horizontal scaling of agents, graceful handling of speed mismatches, failures and retries, complete observability, and more.

Orchestration of Agents and Human Intervention

You need a way to break requests down into tasks and dispatch them to appropriate agents in real-time. Such an orchestrator can use either dynamic or prescriptive workflows to achieve the end goals by assigning the right task to the right agent, controls access to agents based on authorization scopes, and involve people when necessary.

Human-in-the-loop patterns are supported through event gateways and manual decision queues. When agents require validation, confirmation, or escalation, those interactions are routed through workflows that integrate human review. This keeps automation accountable and enables agents to safely handle edge cases and exceptions.

ERP, CRM, API Integration

Connectivity with enterprise applications can't be an afterthought in agentic AI—it has to be one of the fundamental capabilities. Agents are designed to connect with ERP platforms, CRM tools, public APIs, and sensor networks. The event mesh mediates this interaction, translating data and documents between formats and protocols as needed.

Wide Connectivity

Flexibility across edge and cloud environments is essential. Some agents will require proximity to data sources for latency or privacy reasons. Others will benefit from centralized coordination or scale. The architecture supports consistent deployment patterns across environments, using containers, serverless runtimes, or VMs as needed.

Cloud-Native / Hybrid / Edge

All of this is governed by strong lifecycle and observability practices. Events and decisions are logged. Metrics on success rates, latency, and decision quality are tracked. Agents are rolled out using CI/CD pipelines, with policy-driven controls on access, configuration, and rollback.

The architecture aligns with TOGAF principles to ensure modularity, reusability, and clear stakeholder boundaries. It supports real-world implementation in enterprise environments—not just as an innovation demo, but as a strategic capability.

Conclusion

Agentic AI holds immense promise—but requires equally strong architecture. The future will not be powered by a single model or platform, but by composable agents operating across systems, connected through events.

Mounting evidence from real-world implementations shows that EDA can transform agentic AI systems into more resilient, scalable, and flexible ecosystems. Much like how event-driven microservices have changed traditional application development, event-driven agents are poised to enable a new generation of AI capabilities.

In many ways, agentic AI positions AI Agents as microservices with specialized skills. Like microservices, there will likely be an explosion in the number of agents forcing the systemic challenges of building and operationalizing them at scale. Therefore, architectural choices made today will become increasingly important over time. The event driven microservices pattern has proven to address these issues and it's the right architectural choice for building flexible and agile agentic AI systems.

The shift isn't just theoretical—it appears to be increasingly valuable in practice for teams building robust agentic AI solutions that need to operate reliably at scale. As the field evolves, this pattern is likely to become more standardized as a common foundation for sophisticated multi-agent systems.

What to do Next

A good first step is to work with developers to prototype agentic AI frameworks, because as with any new technology, developer buy-in is crucial. Determine which frameworks make their lives easier and solve business challenges in a repeatable, efficient and complete way. But don't stop with a limited developer desktop PoC. Make sure the architecture is enterprise-worthy:

- Map events across domains where agents can listen, act, and learn.
- Pilot multi-agent workflows for business-critical scenarios.
- Think long and hard about policies for agent trust, safety, and governance and the tooling that will help you socialize and enforce them.
- And most importantly, measure impact—not just accuracy or uptime, but business value delivered.

With a solid EDA foundation, your systems can become more than responsive—they can become intelligent, collaborative, and self-improving. The future isn't just intelligent. It's autonomous—and it's event-driven.

Jesse Menning
Solution Architect | Solace

As an architect in Solace's Office of the CTO, Jesse helps organizations of all kinds design integration systems that take advantage of event-driven architecture and microservices to deliver amazing performance, robustness, and scalability. Prior to his tenure with Solace, Jesse was an independent consultant who helped companies design application infrastructure and middleware systems around IBM products like MQ, WebSphere, DataPower Gateway, Application Connect Enterprise and Transformation Extender.

Jesse holds a BA from Hope College and a masters from the University of Michigan, and has achieved certification with both Boomi and Mulesoft technologies. When he's not designing the fastest, most robust, most scalable enterprise computing systems in the world, Jesse enjoys playing hockey, skiing and swimming.

See all posts written by Jesse