Governing and architecting the diversity of agentic AI at scale

AWS Prescriptive Guidance

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Prescriptive Guidance: Governing and architecting the diversity of agentic AI at scale

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

Introduction **1**
Intended audience 1
Objectives 1
About this content series 2
Use case typologies – understanding generative AI and agentic applications **3**
Internal productivity applications 3
Simple chatbot 4
Personal AI assistant 4
Autonomous team agent 5
Business applications 5
Specialized solutions for specialized needs 5
Customer facing 6
Agentic AI governance **7**
Governance scope 7
Agent, tool, and MCP registry management 7
LLM model management 8
Quality control and approval processes 8
Platform standards and agentic platform 8
Mitigating shadow AI 9
Multi-level access and agent permissions 9
Lineage and audit requirements 10
Governance models 10
Centralized vs. federated vs. hybrid approaches 10
Balancing innovation with control 12
Citizen developer enablement 12
Agentic AI architecture in the enterprise **13**
Three core service categories 14
Cross-layer concerns 14
Applications layer – generative AI end-user solutions 15
Purpose and characteristics 15
AWS solutions for this layer 16
Integration patterns 17
Implementation approaches 18
Applications layer: non-generative-AI solutions 18

Purpose and characteristics 15
Enterprise systems in this layer 20
Integration approaches 20
Agents layer 22
Agent execution 23
Agent registry and catalog 23
Multi-agent coordination 24
Agent quality and safety 24
Access control and authentication 25
Core services: model access 25
Cloud native pattern 26
LLM gateway pattern 27
Core services: tools 28
Architecture patterns 29
Choosing the right pattern 30
Governance considerations 32
Implementation on AWS 33
Core services: knowledge bases 34
Implementation on AWS 33
Integration with agents 36
Governance considerations 32
Cross-layer concerns 36
Observability 37
Security 38
Resources 40
Contributors 41
Authoring 41
Reviewing 41
Technical writing 41
Document history 42

Governing and architecting the diversity of agentic AI at scale

Jocelyn Pinault, Massimiliano Angelino, Sebastien Grazzini, Florent Lacroute, and Lior Perez, Amazon Web Services

This guidance addresses a critical challenge for any organization deploying AI at scale: how to govern and architect the growing diversity of agentic AI across an entire organization, not just within individual workloads or projects. As organizations deploy an increasing variety of AI agents, from simple chatbots to autonomous customer-facing systems, they need cohesive strategies that span use cases, teams, and business units.

Intended audience

This guidance is designed for CIOs, CTOs, VP AI, Enterprise architects, AI governance leaders, platform engineers, and technical decision-makers who need to scale agentic AI across their organizations while maintaining control and compliance.

Objectives

Understanding how to govern and architect agentic AI at scale requires a structured approach that moves from conceptual clarity to practical implementation.

Use case typologies examines the full spectrum of use case typologies, from simple chatbots and personal AI Assistants to autonomous team agents and customer-facing applications, Each category demands different governance rigor and architectural patterns.

Agentic AI governance explores governance frameworks covering agent/tool/MCP registry management, LLM model management, platform standards, multi-level access controls, and audit requirements. It presents three governance models centralized, federated, and hybrid and addresses citizen developer enablement.

Agentic AI architecture in the enterprise presents distinct layers covering applications (GenAI solutions such as Quick Suite and Kiro, and business system integrations), agents (Bedrock AgentCore services for runtime and orchestration), and core services (model access, tools, and knowledge bases).

This structure reflects an important reality governance and architectural requirements evolve with organizational maturity. While foundational elements such as security apply universally, other elements, such as compliance rigor, reliability standards, and output precision, intensify as deployments move from internal productivity tools to customer-facing applications. Use this guidance based on your enterprise's maturity stage:

- Early stage, when you are exploring chatbots and personal assistants – focus on use case typologies and foundational governance. Do not attempt to build the entire reference architecture in the first iteration. Instead, focus on one or two pilot use cases and build only the necessary components for it. Then, add capability iteratively as new use cases are onboarded.
- Growing maturity, when you are deploying team agents and business applications – emphasize agentic AI governance models and architectural patterns
- Advanced maturity, when you are building customer-facing applications – prioritize enterprise agentic AI architecture, with providing advanced controls for agent permissions, referring to governance and audit requirements.

About this content series

This guide is part of a series about agentic AI on AWS. For more information and to view the other guides in this series, see Agentic AI on the AWS Prescriptive Guidance website.

Use case typologies – understanding generative AI and agentic applications

Generative AI and agentic applications serve diverse purposes across organizations. Understanding these use case categories helps establish appropriate governance frameworks, risk assessments, and operational controls.

Each use case category requires its own approach to governance, risk management, and operational oversight. For example:

- Personal assistants require strong privacy controls but can tolerate occasional errors.
- Autonomous team agents need robust process validation, audit trails, and must ensure information accuracy and provide access controls.
- Domain-specialized applications require deep expertise and rigorous testing.
- Customer-facing applications demand the highest levels of reliability, security, and user experience.

By understanding where your AI applications fit within this landscape, you can apply the appropriate frameworks to ensure that they deliver value while maintaining security, compliance, and reliability standards appropriate to their role and impact.

This section contains the following topics:

- The internal productivity landscape
- Business applications

Internal productivity applications

These applications operate with a mix of internal data and web sources, creating a need to establish a secure environment for innovation while respecting the boundaries of corporate information security. They represent the foundation upon which many organizations build their AI capabilities, and can be categorized as:

- Simple chatbots, which answer questions based on internal and public knowledge
- Personal AI assistants, which use an agentic workflow to act, not just provide answers

- Autonomous team agents, which work on your tasks in the background

Simple chatbot

Whether accessed through an internal web page, via a plugin, or in a desktop app, this company chatbot should be your team's go-to tool for securely rewriting text, conducting research, or searching internal knowledge bases. Accessibility, integration with the local environment, knowledge depth, and ease of use are the key factors driving adoption. Examples uses:

- Summarize a document
- Rewrite raw notes into meeting minutes
- Find specific information from internal knowledge bases

Personal AI assistant

Consider having a digital assistant focused entirely on your individual needs – a tireless companion that helps manage your tasks, organizes your schedule, and retrieves information exactly when you need it. These assistants represent the most personal form of AI integration in the workplace, working alongside you throughout your day to handle the routine cognitive tasks that can distract from higher-value work.

These assistants integrate seamlessly with your email, calendar, and personal document management systems, adapting to your preferences and patterns over time. They can draft responses to routine emails, suggest optimal meeting times based on your schedule and priorities, summarize long email threads, and quickly locate the document that you saved months ago but can't quite remember where.

Access rights are tied to each individual user – your assistant only accesses information you're authorized to see, and doesn't inadvertently access information meant for others.

Examples:

- Send a personalized email to a large audience
- Log your customer relationship management (CRM) activities
- Create a ticket
- Automate the "copy paste" of data from one system to another by extracting, formatting and submitting it in appropriate formats

- Interact with browser-based applications and populate web forms

Autonomous team agent

Autonomous team agents are designed to handle complete workflows for specific organizational roles. These agents guide business processes in the background while keeping humans in control at critical decision points.

Examples:

- Autonomous supply chain management agent, which monitors sales and inventory while factoring in supply chain variables such supplier delays, geopolitical events, and minimum batch quantities. Based on this data, the agent predicts out-of-stock and overstocking risks and can propose placing purchase orders directly in your enterprise resource planning (ERP) system.
- Request for quotation (RFQ) agent, which scans your team folders to find previous quotation requests, then anonymizes and shares with your team your answers, thus capitalizing on existing knowledge.
- Agent that auto-enriches project tasks with relevant ERP data (budgets, suppliers, costs, lead-time).
- Autonomous compliance agent for document tagging and classification, including export control and sensitivity labeling.

Business applications

Business applications are those that directly impact business outcomes, customer experiences, and operational efficiency, which may use AI agents that are embedded within operational systems. These applications require higher reliability and domain-specific compliance standards as well as greater rigor in their development, deployment, and governance.

Specialized solutions for specialized needs

These domain-specialized applications are tailored to specific industries and technical domains, bringing deep expertise and specialized capabilities that generic AI tools cannot provide.

Examples:

- Production scheduling agent based on real-time parts tracking

- Integrated quality control agent and recommendation for visual inspections
- Design clash detection agent, to predict, identify and resolve conflicts between different building elements and systems within a 3D digital model
- Customer churn prediction and suggestion of retention strategies
- Automated contract compliance analysis
- Insurance claims processing
- Coding for software development

Customer facing

Perhaps most visible are the customer-facing applications that interact directly with your customers, representing your organization's face to the world. These include chatbots that handle ordering processes, provide support, answer questions, and guide customers through complex decisions. Unlike internal applications where users have training and context, these external-facing systems must work flawlessly for people with varying levels of technical sophistication, different languages and cultural contexts, and diverse needs and expectations.

These applications require the highest standards of reliability – they cannot fail during peak usage times or when customers need them most. They demand exceptional user experience, with natural conversational flows, accurate understanding of customer intent, and responses that are helpful, empathetic, and aligned with your brand voice. They must handle edge cases gracefully, knowing when to escalate to human agents and doing so smoothly without frustrating customers.

Security is paramount for these customer-facing applications, as they often handle sensitive personal information, payment details, and account access. They must protect against malicious use while remaining accessible and helpful to legitimate customers. They directly impact customer satisfaction and brand reputation. A helpful, efficient AI interaction can strengthen customer loyalty, while a frustrating or inaccurate one can drive customers to competitors.

Examples:

- Customer service assistant offering predefined options for common issues
- Multilingual & accessibility translation agent for your retail website
- Personalized travel booking management based on your preferences
- Prescription management to help patients understand medication and manage refills

Agentic AI governance

As organizations scale agentic AI deployments, the expanding ecosystem of agents, tools, and models demands clear governance structures. The challenge is multifaceted – without proper oversight layered on top of regular workload governance, enterprises face agent proliferation, shadow AI emergence, security vulnerabilities, and compliance failures that can undermine the entire AI initiative.

This complexity requires a nuanced approach to governance, balancing structure with adaptability, providing clear guidance today while remaining flexible enough to accommodate tomorrow's innovations.

This section contains the following topics:

- Governance scope
- Governance models

Governance scope

Effective AI governance requires organizations to establish comprehensive oversight of their agentic AI ecosystem through centralized registries, quality controls, and access management frameworks. As AI agents proliferate through low-code platforms, maintaining visibility into deployed assets – including their capabilities, permissions, and interdependencies – becomes essential for security, compliance, and operational excellence. This governance foundation enables organizations to balance innovation velocity with risk management, ensuring that AI systems operate within defined boundaries while preventing unauthorized shadow AI deployments.

Agent, tool, and MCP registry management

Comprehensive visibility into AI assets forms the foundation of effective governance. Organizations must maintain centralized registries documenting all deployed agents, including their capabilities, permissions, security classifications, rate limits, and approval processes, data access patterns, ownership, and business purpose. Each entry captures critical metadata, such as version history, dependencies, performance metrics, and incident history.

In a rapidly evolving landscape, documentation must be strategic rather than exhaustive. Focus on decision rationale, architectural patterns, and integration points – elements that provide lasting

value even as implementations change. Your goal is to capture institutional knowledge that helps teams find and understand existing implementations and avoid repeating mistakes.

LLM model management

The LLM model catalog tracks which foundation models are approved for which use cases, including regional availability, quota allocations, and lifecycle information for identifying deprecated models. Ideally, the catalog contains the list of applications using each model, their product owners, region-specific deployments, and usage limits to enable proactive updates and capacity management, preventing disruption when older models reach end-of-life or capacity constraints arise.

Quality control and approval processes

As agents become easier to build through low-code platforms, quality control mechanisms ensure that only agents meeting minimum standards can be shared through the organization.

Agent evaluation frameworks assess functional correctness, response quality, safety boundaries, and alignment with organizational policies before deployment. These evaluations combine automated testing for technical performance with human review for business appropriateness, creating a balanced quality gate that scales as agents proliferate.

Model approval considers performance benchmarks, cost constraints, data residency requirements, and alignment with responsible AI principles. As model capabilities advance rapidly, you should establish expedited review processes for performance updates while maintaining thorough evaluation for fundamentally new features.

Calibrate these standards in accordance with the risk associated with the applications while recognizing that best practices are still emerging, thereby establishing baseline safety and security requirements while remaining open to adjusting quality metrics as experience grows.

Platform standards and agentic platform

Organizations must define the approved technology stack for agentic AI development, specifying which platforms and frameworks are approved. As protocols used in agentic applications like MCP (model context protocol) or A2A (agent-to-agent) evolve at a fast pace, and new protocols might appear over time, you should establish standards for secure inter-agent communication, balancing standardization with flexibility.

The platform should offer tested patterns and configurations meeting security and compliance standards, enabling developers to start from compliant foundations. Governance as code embeds policy enforcement directly into the platform, while designated sandbox environments provide spaces for experimentation with clear boundaries.

Mitigating shadow AI

The ease of building AI applications creates significant shadow AI risk. Making the governed path more attractive than the shadow path becomes critical. Create this environment through responsive approval processes, comprehensive self-service capabilities, and clear value propositions. Education and communication ensure that employees understand why governance exists and how to access approved resources.

Beyond reducing the risk of shadow AI, organizations must establish comprehensive access management policies governing who can interact with agents and how agents interact with each other.

Multi-level access and agent permissions

Organizations must implement access governance for AI agents to prevent users from exploiting agents as proxies to access information or systems they lack direct permission to use.

Tool invocation controls form one of the foundations of agent governance. Each agent operates with a core identity that defines its scope of permissions and capabilities.

Establish policies ensuring that:

- Tool access is restricted based on the agent's identity.
- Agents can only invoke tools for which they have explicit authorization.
- Tool invocations respect both the agent's capabilities and, in a case where the agent is invoked by a user, the user's access rights, to avoid unauthorized access to data.

In multi-agent systems, governance must extend further to include agent-to-agent permissions. Agents should only be able to invoke other agents for which they have explicit authorization, adding an additional layer of control. Amazon Bedrock AgentCore Identity provides a centralized identity and credential management service specifically designed for AI agents, enabling secure authentication, authorization, and credential management.

Lineage and audit requirements

Governance frameworks must require comprehensive tracking of agent execution paths to enable debugging, security analysis, and compliance verification. Establish policies mandating that agent and tool invocations be logged with sufficient detail to reconstruct the complete chain of actions. However, lineage policies must balance comprehensiveness with performance, storage, and privacy considerations. For example, if an agent has calculated a credit score, it should be possible to determine how and why it was calculated this way without necessarily logging all the intermediate steps.

Access management policies require enforcement mechanisms and ongoing monitoring to remain effective. Establish governance requirements mandating regular access reviews, detection and escalation procedures for violations, clear accountability for decisions, and mechanisms for regular policy review based on operational experience.

Governance models

Organizations must choose governance approaches aligning with their culture, risk tolerance, operational maturity, and strategic objectives. The choice of a centralized, federated, or hybrid model significantly impacts innovation velocity, risk management effectiveness, and resource efficiency.

By thoughtfully implementing governance elements, you can harness the transformative potential of agentic AI while maintaining necessary control. Critically, governance frameworks must remain adaptive, evolving as the technology matures and organizational capabilities grow. The goal is not perfect governance today but establishing structures that enable safe innovation while remaining flexible enough to incorporate tomorrow's learnings.

Centralized vs. federated vs. hybrid approaches

The choice of governance model is influenced by, and influences, architectural decisions.

Unlike traditional cloud infrastructure, where organizations rarely build abstraction layers to switch between providers due to the complexity and deep integration of platform-specific services, generative AI introduces a different dynamic. Because large language model (LLM) inference is stateless, it may be effectively abstracted through gateway layers. This makes centralized architectures, and by extension, centralized governance, appear more achievable across multiple providers than it would be for general cloud services.

However, this abstraction advantage diminishes with the rise of agentic AI and managed services that introduce stateful, platform-specific capabilities that are difficult to abstract. Organizations must therefore consider how their governance model will evolve as their AI capabilities mature beyond simple LLM inference. We explore these architectural considerations in more detail in the Enterprise Architecture section.

This interplay between abstraction possibilities and cloud-native capabilities shapes the governance choices organizations face:

Centralized Governance establishes a single enterprise-wide authority for policies, approvals, and ecosystem management. This provides strong control, consistent policy enforcement, and simplified compliance management. The centralized approach is coupled with platforms like LLM gateways to centralize access to LLM's. It works well for highly regulated industries or organizations that are at an early stage of their AI journey. However, centralized approaches can create bottlenecks and may struggle to accommodate diverse use cases as adoption scales.

Federated Governance distributes responsibility to business units while maintaining alignment through shared standards. This enables faster innovation, closer alignment with business needs, and scalability. It works well for large enterprises with diverse business units and mature IT governance. The challenges include risks of inconsistent policy interpretation and difficulty maintaining enterprise-wide visibility.

Hybrid Governance combines centralized oversight with federated execution. A central policy framework guides distributed implementation, with enterprise-wide standards for high-risk agents and local autonomy for low-risk applications. This balances control with agility and suits most enterprise organizations. Success requires clear delineation of responsibilities and strong communication channels.

Governance models must align with organizational culture to be effective. Organizations that emphasize autonomy typically struggle with highly centralized governance, while risk- averse cultures or highly regulated environments naturally gravitate toward centralized control. Also, the procurement culture of the organization plays a pivotal role in the governance model choice. In all cases, the quickly evolving nature of agentic AI requires an adaptable organization.

- A **single-cloud approach** offers deeper integration with the cloud provider's ecosystem, simplified governance through unified tooling, potential cost benefits through volume commitments, and reduced complexity in operations.

- A **multi cloud approach with federated governance** allows you to keep a deep integration with your cloud provider's ecosystem but may reduce advantages like unified tooling or cost benefits based on volumes.

Some organizations may choose a **multi-cloud strategy with a centralized approach**, which entails these considerations: increased governance complexity, additional integration work, potentially higher costs, and demands for broader expertise. Also, the abstraction layer reduces the ability to take advantage of each cloud provider's pace of innovation.

Balancing innovation with control

The central challenge of establishing a governance model is is to balance innovation with control in alignment with the organization needs in terms of regulation.

With the unprecedented pace of generative AI innovation, it is crucial to build an agile team able to experiment quickly with new LLMs, frameworks, protocols and to enable them to switch between technologies. Innovation spaces such as sandbox environments, innovation labs, or proof-of-concept programs, allow exploration while protecting production systems. Positioning governance as providing the foundation for safe innovation rather than bureaucratic obstruction improves the dynamic. Feedback loops ensure that governance policies evolve based on operational experience, operating on compressed timelines in early stages.

Citizen developer enablement

The growing accessibility to AI development creates both opportunity and risk. Effective governance enables citizen developers while maintaining appropriate controls. A citizen developer is a business user who creates AI applications without specialized technical skills.

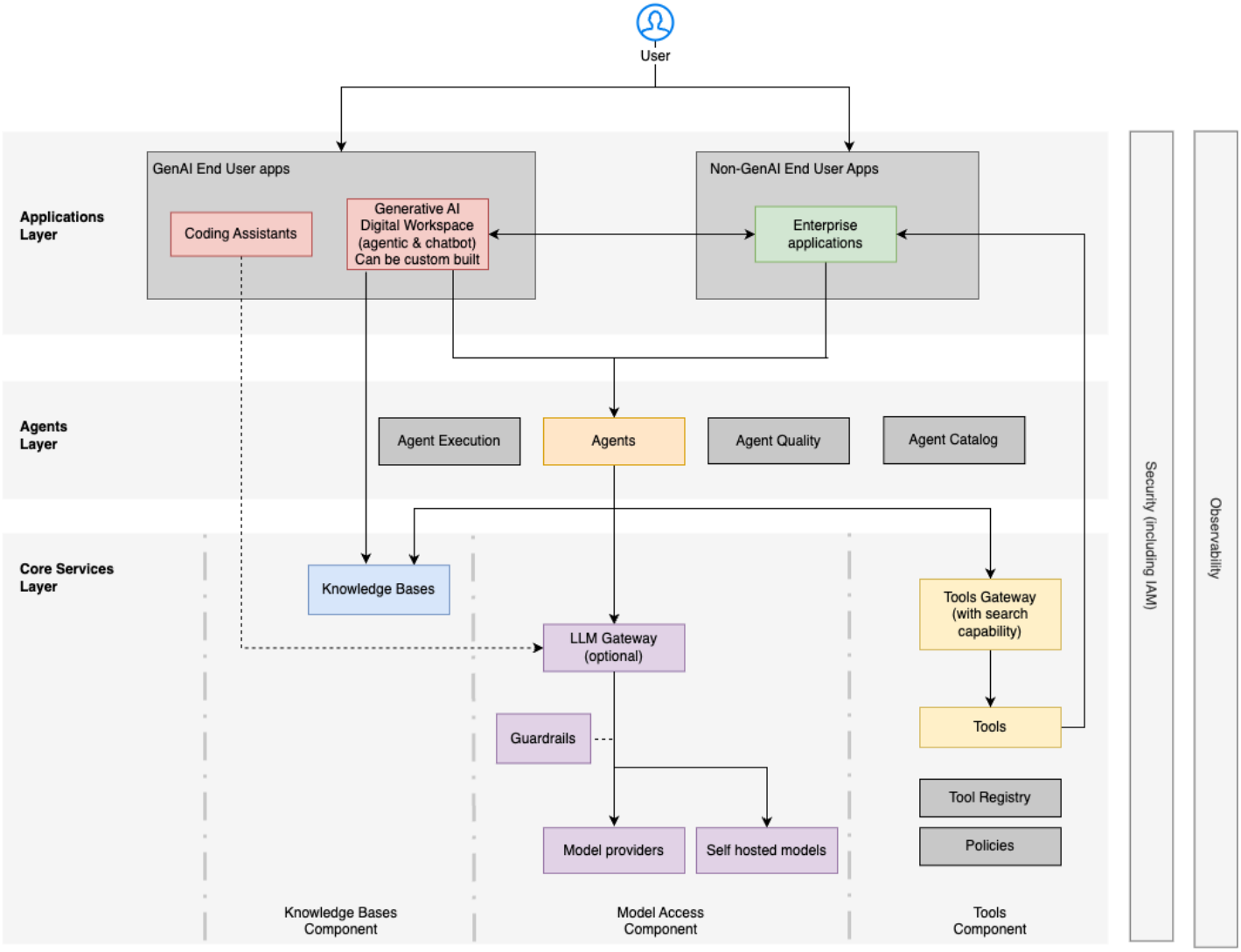
Provide foundational training on AI fundamentals, responsible AI principles, security considerations, and governance requirements. Training programs should emphasize principles over rigid rules, teaching developers to think critically about risks and trade-offs. Role-based certification programs match autonomy levels to demonstrated competence.

You should also provide libraries of tested and compliant agent templates, tool integrations, and prompt patterns as building blocks. Development platforms with embedded guidance suggest best practices and automatically enforce policies. Citizen developers start with limited capabilities, and their permissions are extended as they demonstrate competence.

Agentic AI architecture in the enterprise

The reference architecture for enterprise agentic AI systems demonstrates how organizations can implement production-grade AI capabilities while maintaining governance, security, and operational excellence.

The architecture is organized into layers that work together to enable AI agents while maintaining enterprise control. Observability, security and discoverability span multiple layers, ensuring that AI operations are monitored, auditable, and compliant with enterprise policies.



Applications layer:

- [Generative AI End-User Applications] - These are user-facing applications that enable interaction with agentic AI systems. These applications may provide conversational interfaces for

natural language interaction, productivity features that augment workflows with AI assistance, and no-code tools that allow business users to create and deploy custom agents. An application can be off the shelf, like integrated development environment (IDE) and productivity tools, or custom built.

- **Non-GenAI Applications** - These are business systems that may be off-the-shelf software, custom-built applications, or industry-specific platforms. Most of these applications are not inherently agentic but can consume agentic AI services or expose their functions as tools that agents can invoke to perform business operations.

Agents layer - This layer contains the components that enable AI agents to function and interact. It also provides agent discoverability. Agents typically require access to LLM to interpret user goals, to reason and plan actions, to obtain access to tools to perform operations, and to retrieve information from knowledge sources. They also need the ability to store conversations and insights derived from the conversations in short and long term memory respectively. The layer also provides the features for agent-to-agent communication and orchestration, allowing multiple agents to collaborate on complex tasks.

Three core service categories

The architecture defines three distinct types of services that agents interact with:

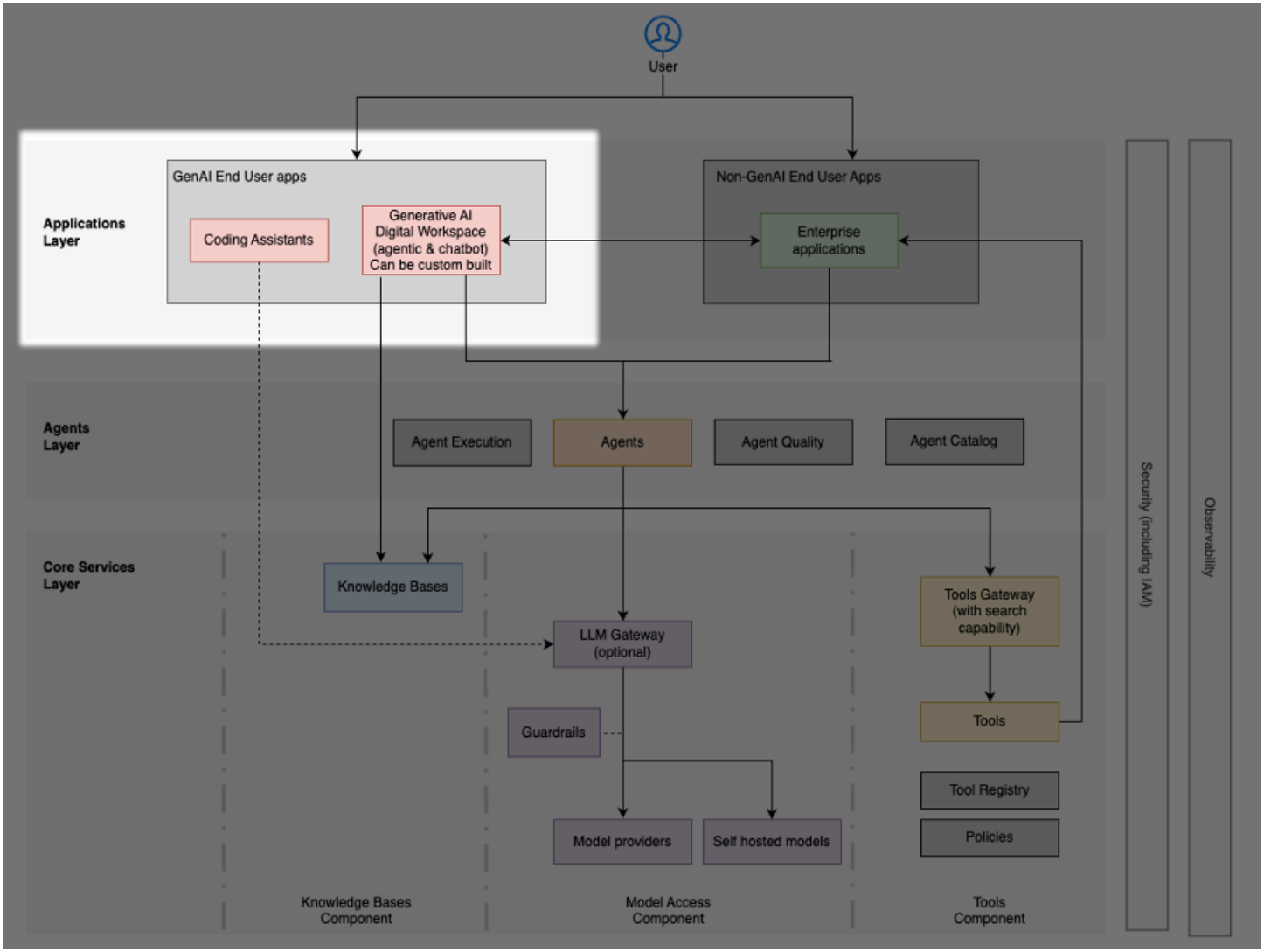
- **Model access component** – controls access to foundation models with policy enforcement, safety measures including guardrails, and cost tracking/allocation capabilities.
- **Tools component** – manages discovery and secure execution of tools. This component provides authorization capabilities to ensure tools can be used only by the right actors and for the right context.
- **Knowledge bases component** – provides access to enterprise data through knowledge bases that can leverage vector stores, graph storage, and provide interfaces for semantic information retrieval. It also provides role-based access control (RBC) for the data to enforce least privilege and need-to-know principles. This component is required for retrieval-augmented generation (RAG) implementations.

Cross-layer concerns

Observability, security and discoverability span multiple layers, ensuring that AI operations are monitored, auditable, and compliant with enterprise policies.

Applications layer – generative AI end-user solutions

This layer includes user-facing applications that enable interaction with agentic AI systems. The components within this layer serve as the primary interface between end users and the underlying agentic infrastructure, providing conversational interfaces for natural language interaction, productivity features that augment workflows with AI assistance, and no-code tools that allow business users to create and deploy custom agents. They also provide interfaces to schedule and monitor the execution of agents.



Purpose and characteristics

The generative AI End-User Solutions Component provides capabilities that include:

- Conversational interfaces – enable natural language interaction with AI agents through chat (text or multimodal) and voice experiences.
- Productivity augmentation – integrate AI capabilities directly into daily workflows and productivity tools to enhance efficiency. Examples include code assistants, meeting summarization tools, and document generation assistants.
- Citizen developer enablement – provide no-code/low-code platforms for creating custom agents or defining AI augmented workflows without specialized technical skills.

AWS solutions for this layer

Organizations can implement this layer using these AWS solutions:

Amazon Quick

Amazon Quick provides a unified productivity platform that delivers:

- Enterprise-wide AI-powered search and knowledge management across structured and unstructured company data
- No-code/low-code agent creation capabilities enabling citizen developers to build custom agents
- Integrated workflows and automation through Quick Flows
- Deep research capabilities (Quick Research) for comprehensive analysis and long-form report generation
- Collaboration tools including Spaces for organizing files, dashboards, and knowledge bases
- Extensibility through integration with internal and external tools, via MCP and connectorsKiro

Kiro serves as an AI-powered code assistant that:

- Transforms natural language requirements into comprehensive technical specifications through its spec mode
- Accelerates software development workflows through intelligent code generation
- Generates comprehensive project deliverables including functional specifications, design documents, documentation, and test cases
- Can create custom agents powered by MCP tools to interact with internal systems

Claude Code with Amazon Bedrock

Claude Code with Amazon Bedrock brings Anthropic's AI-powered coding assistant to enterprises with the security and enterprise-grade capabilities of Amazon Bedrock. This solution integrates directly with developer terminals and IDEs, enabling natural language interaction for code generation, review, and modification while leveraging Bedrock's fully managed infrastructure for strict security controls, compliance features, and scalability.

AWS provides guidance that demonstrates how organizations can implement Claude Code with secure enterprise authentication using industry-standard protocols and AWS services. The solution enables terminal-integrated development through conversational commands, replaces long-term access keys with temporary session-based credentials through standardized federation protocols, supports Model Context Protocol (MCP) for connecting external tools and data sources, and provides observability into developer productivity patterns and usage metrics.

Custom-built solutions

Organizations requiring tailored implementations can leverage:

- AWS Bedrock Chat (https://github.com/aws-samples/bedrock-chat) as an open-source foundation for self-managed deployments
- Custom enterprise portals integrated with existing systems using AWS services
- Bespoke applications built on Amazon Bedrock for domain-specific requirements leveraging 3rd party frontend solutions such as v0 AI SDK, CopilotKit, and Open WebUI.

Integration patterns

End-user solutions in this layer integrate with the underlying architecture through:

- Connector based integration – Pre-built and custom connectors that enable seamless integration with third-party applications and data sources without custom code development. For instance, Quick Suite provides connectors to collaboration platforms (Slack, Microsoft Teams), productivity tools (Asana, Jira), cloud storage services (OneDrive, SharePoint, Google Drive), and more. Similarly, a bespoke application could use Bedrock Knowledge Bases, which provides connectors to enterprise data sources including Confluence, SharePoint, Salesforce, and web crawlers for authorized public web pages.
- API-based integration – RESTful and WebSocket APIs for real-time communication with enterprise systems

- Agent-to-system protocols – Standardized protocols (such as MCP) for connecting AI applications with external data sources and tools, enabling agents to access enterprise resources through unified interfaces
- Event-driven patterns – Webhook and event subscriptions using Amazon EventBridge for asynchronous workflows
- Embedded experiences – Widgets and iframes for seamless integration into existing applications
- Identity federation – Single sign-on (SSO) through AWS IAM Identity Center, OIDC and OAuth 2.0 with integration with enterprise identity providers for secure end-user access control

Implementation approaches

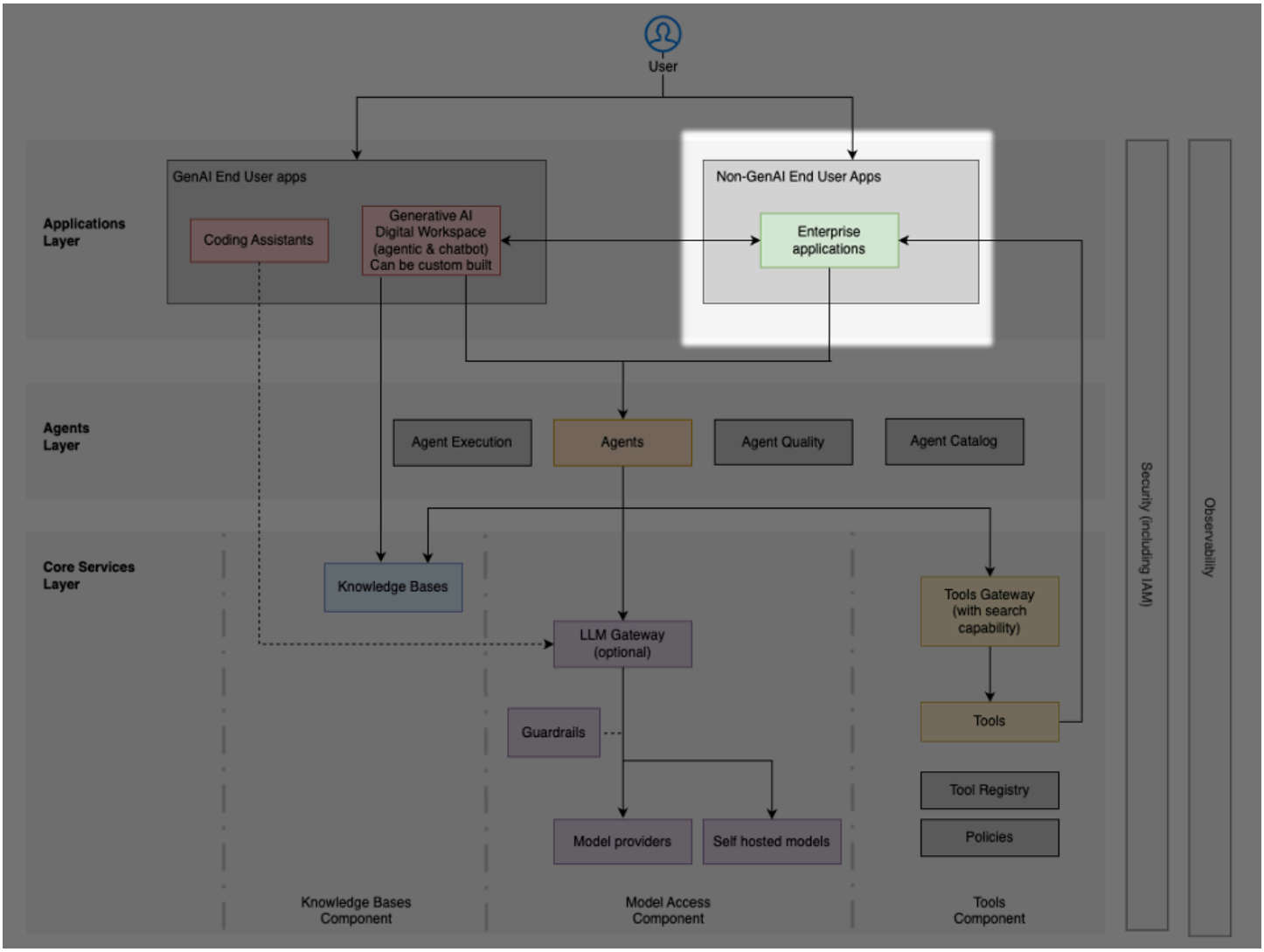
Organizations can choose between two primary approaches:

- Off-the-shelf solutions – Platforms like Amazon Quick , Kiro, and Claude Code with Amazon Bedrock that provide immediate productivity gains with minimal configuration
- Custom applications – Tailored implementations to meet specific organizational requirements

This modular approach allows organizations to balance speed of deployment with customization needs, selecting the right solution based on their technical capabilities, timeline, and specific use case requirements.

Applications layer: non-generative-AI solutions

This layer consists of existing business systems, whether off-the-shelf software, custom-built applications, or industry-specific platforms, that integrate bidirectionally with agentic AI capabilities. These applications are not inherently agentic but can consume agentic AI services or expose their functions as tools that agents can invoke to perform business operations.



Purpose and characteristics

The Enterprise Applications Layer provides these capabilities to agents:

- Bidirectional integration – enables two-way communication where applications both consume AI services and provide capabilities to agents
- Business logic access – exposes domain-specific rules and processes that agents can leverage
- System of record integration – connects agents to authoritative data sources for reading and writing business data
- Process automation – allows agents to initiate and participate in enterprise workflows
- Secure delegation – enables agents to act on behalf of users with appropriate permissions

Enterprise systems in this layer

This layer encompasses diverse business systems including business management platforms such as ERP (enterprise resource planning), CRM (customer relationship management), HRMS (Human Resources Management System), and SCM (supply chain management), operational systems such as Jira, ServiceNow, and Salesforce, workflow engines, and business intelligence platforms, and content systems such as document management, enterprise wikis, and collaborative workspaces. These applications, whether off-the-shelf software, custom-built solutions, or industry-specific platforms, integrate bidirectionally with agentic AI capabilities.

Integration approaches

Enterprise applications integrate with agentic systems through tools, events and direct data access. Each of these integrations requires specific features to make it possible.

Tool exposure

- Function wrapping – business functionality exposed as callable tools that agents can invoke using protocols such as MCP
- Parameter validation – input validation to protect enterprise systems from malformed requests
- Response formatting – structured outputs that agents can reliably parse

Event-driven integration

- State change notifications – systems publish events when business objects change
- Process triggers – workflow status updates that agents can subscribe to
- Alert generation – exception conditions that prompt agent intervention

Data access patterns

- Direct querying – secure, parameterized data access interfaces
- Cached views – pre-aggregated data for efficient agent consumption
- Permission-aware access – data access that respects user authorization contexts
- Semantic transformation – index unstructured enterprise data such as text, audio, and video, into generative AI-friendly semantic representations using embeddings and vector databases

- Knowledge graphs – represent complex data relations via graphs and ontologies to allow AI systems to better understand specific industry domains

Implementation considerations

Organizations implementing this layer should address these challenges:

- Authentication context propagation – ensuring that user identity is preserved when agents act on their behalf, typically through token exchange solutions that issue delegated credentials, on-behalf-of tokens, to the agents
- Rate limiting – protecting traditional enterprise systems that were designed for human access from overload due to automated agent
- Maintenance window coordination – ensure that agents are not trying to access systems during maintenance window hours
- Credential management –secure storage and rotation of secrets necessary for agent to access tools and other systems
- Error handling – graceful degradation when backend systems are unavailable
- Semantic or format transformation – converting between AI-friendly formats and enterprise data structures

AWS implementation approaches

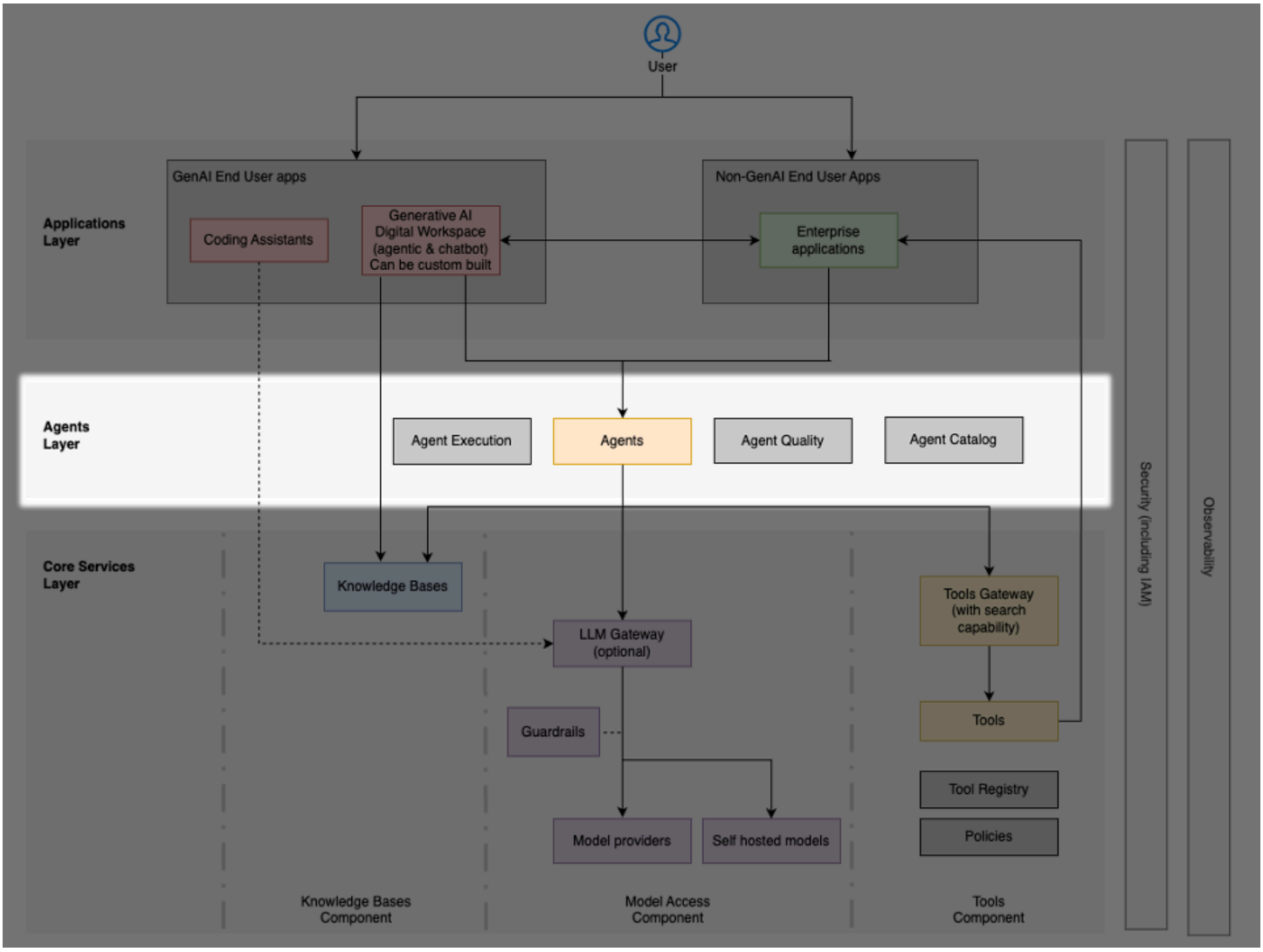
AWS provides several services to enable enterprise application integration:

- Amazon EventBridge – serverless event bus for connecting applications with real-time data
- Amazon API Gateway – create and secure APIs for agent access to business functions, providing API throttling controls and bi-directional websocket transport
- Amazon AppFlow – managed integration for connecting SaaS applications with AWS services
- AWS Glue – ETL service for preparing and transforming data
- Amazon OpenSearch Service – enterprise search capabilities for content discovery
- Amazon BedrockKnowledge Bases – unified API to interact with knowledge bases backed by several popular vector databases and graph databases.

This integration layer ensures that agentic AI capabilities can access the business context, data, and operations necessary to provide value within organizational workflows while maintaining appropriate security boundaries and governance controls.

Agents layer

The Agents layer serves as the central coordination hub for interactions between users, foundation models, tools, and knowledge sources. This layer contains the agent runtime environments, orchestration mechanisms, and supporting infrastructure that enable AI agents to function. Agents use LLMs for reasoning and planning, call tools to perform operations, retrieve information from knowledge bases, and maintain memory of past interactions.



Agent execution

Agents have specific requirements that set them apart from the traditional application and microservices. Agents can run in autonomy for hours, need to retain context and at the same time be fully isolated from other agent instances. Agents need to securely discover, select and access tools and other agents via well-known protocols like MCP and A2A. Agents also require services to persist conversation across executions, and secure ways to execute code and interact with browser and web-based systems.

Organizations can implement these capabilities using:

- Amazon Bedrock AgentCore runtime – a foundational service for executing agents securely at scale with no infrastructure management needed. The runtime provides enterprise-grade security and dynamic scaling, session persistence and isolation, large payloads, multi-protocol support and bidirectional streaming.
- Amazon Bedrock AgentCore memory – provides persistence for both short-term conversation history and long-term extracted insights.
- Amazon Bedrock AgentCore identity – ensures seamless integration of authentication with IAM and OAuth providers for both inbound and outbound.
- AWS Lambda – provides serverless computing for custom agent logic and tool implementations. Provides execution for lightweight agent operations and tool invocations.
- Amazon ECS or AWS Fargate – provides container-based agent deployments for complex requirements, stateful operations, or resource-intensive agent workloads requiring dedicated compute environments.

Agent registry and catalog

A centralized agent registry maintains an inventory of deployed agents, capturing:

- Agent capabilities, permissions, and purpose
- Metadata on ownership, version history, and dependencies
- Performance metrics and usage statistics
- Approval status and governance classifications

The registry supports discovery, reuse, and governance while preventing unnecessary duplication of capabilities across the organization. Registries should support self-service access requests

and authorization delegation to the tool or agent owners. Registries should integrate with MCP gateway access controls to enforce the required access policies. Organizations should maintain registries documenting approved agents with quality control and lifecycle management.

Multi-agent coordination

As multi-agent systems become prevalent, architectures must support agent-to-agent communication through standardized protocols such as MCP and A2A, message formatting and state sharing conventions, appropriate state isolation, authentication and authorization between collaborating agents, and permissions verification for delegated actions.

Supporting infrastructure includes:

- Amazon Bedrock AgentCore memory – managed memory service for storing agent state, conversation history, and long-term extracted insights that can be shared across agent sessions with fine-grained access control and storage isolation
- Amazon EventBridge – event-driven backbone for multi-agent messaging
- AWS Step Functions – orchestration for complex multi-agent workflows with checkpoints and error recovery
- Amazon DynamoDB – fast, scalable storage for agent state and shared memory

Agent quality and safety

As agents become more autonomous and handle critical business operations, architectures must support comprehensive quality assurance and safety mechanisms through evaluation frameworks for reliability and accuracy, safety testing for harmful outputs or behaviors, performance monitoring and regression detection, and feedback loops for continuous improvement.

Supporting infrastructure includes:

- Amazon Bedrock Guardrails – manages capabilities for content filtering, denied topics, word filters, and sensitive information redaction
- Amazon Bedrock AgentCore Evaluations – built-in evaluation frameworks providing automated assessment tools to measure how well agents or tools perform specific tasks, handle edge cases, and maintain consistency across different inputs and contexts.
- Amazon CloudWatch – helps you monitor the metrics of your AWS resources and the applications you run on AWS in real time.

- Amazon Bedrock Evaluations– evaluate LLMs and semantic retrieval systems using programmatic metrics, human evaluators and LLM-as-judge.
- AWS Lambda - Serverless execution of custom validation logic and safety checks
- Amazon S3 - Storage for evaluation datasets, test results, and safety audit logs
- Open-source observability tools - Platforms like LangFuse for tracing, evaluation, prompt management, and metrics; deployable on AWS

Access control and authentication

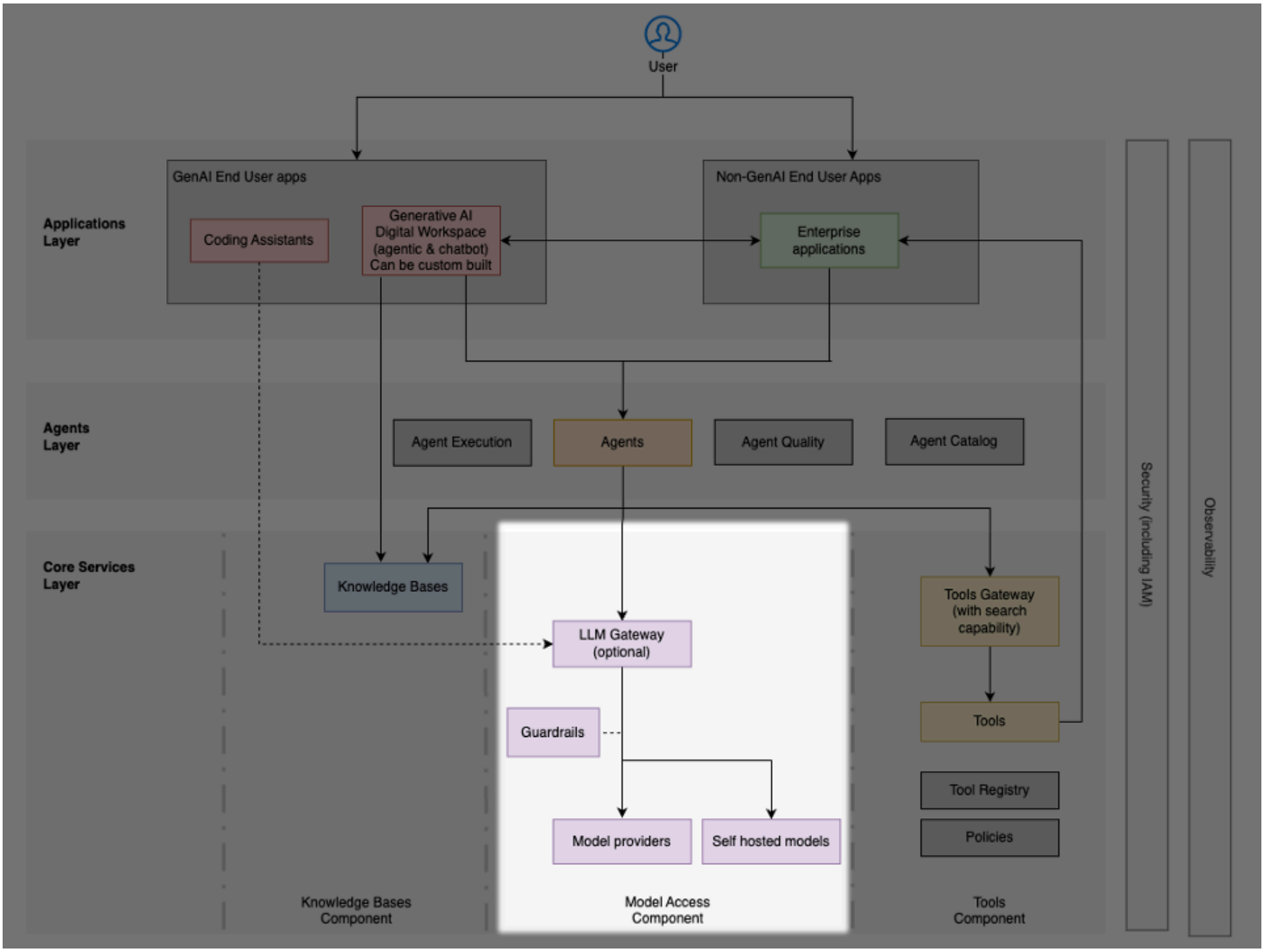
Effective agent operations require robust identity and permission management through identity propagation from users through agent chains, permission boundaries for agent actions and tool access, audit trails of agent decisions and actions, and circuit breakers for abnormal behavior patterns.

Supporting infrastructure includes:

- Amazon Bedrock AgentCore Identity – integrated identity management and authentication context propagation across agent interactions
- Amazon Bedrock AgentCore Gateway Interceptors – Lambda based request and response interceptor to evaluate, filter, manipulate and block MCP tool calls and responses
- Amazon Bedrock AgentCore Policy – Cedar-based engine to implement contextual grounded fine-grained authorization on AgentCore data plane operations
- AWS Identity and Access Management and AWS IAM Identity Center – enterprise authentication, authorization, and single sign-on integration with identity providers
- AWS Secrets Manager – secure credential storage and automatic rotation for service accounts
- AWS CloudTrail – comprehensive API activity logging and audit trail generation
- Amazon EventBridge – real-time alerting and remediation

Core services: model access

Organizations must make a fundamental architectural decision regarding how agents access foundation models. This choice shapes security enforcement, operational governance, and the overall system architecture.



Two primary patterns exist for model access, each with advantages and disadvantages:

- Cloud native
- LLM gateway

Cloud native pattern

- Direct access to cloud provider model services
- Native tool integrations and action frameworks deeply tied to cloud ecosystems
- Managed identity and access management specific to each platform
- Orchestration and memory management services optimized for cloud-native architectures
- Enterprise-grade observability, security and governance features embedded in managed services

LLM gateway pattern

- Unified API interfaces across multiple model providers through a controlled intermediary
- Uniform and centralized policy enforcement across model providers for security and compliance
- Consistent monitoring, cost management, and API normalization

Decision factors

When choosing between these patterns, consider:

- Governance maturity – organizations with strict compliance requirements or in highly regulated industries typically benefit from centralized control
- Performance requirements – use cases requiring minimal latency may favor direct access
- Multi-provider strategy – plans to use models from multiple providers benefit from the gateway's abstraction layer
- Operational capacity – teams with limited resources may prefer cloud-native simplicity; those with established API management can leverage gateway benefits
- Innovation velocity – direct access provides faster adoption of new model capabilities
- Performance requirements – direct access avoids the additional routing layer latency introduced by gateways, though this overhead is often negligible compared to LLM inference time
- Features of managed services - the rise of agentic AI and managed services (such as Amazon Bedrock Knowledge Bases) introduces platform-specific capabilities that are bringing value but are difficult to abstract

Organizations may also adopt a hybrid approach–implementing the gateway pattern for selected production applications while enabling cloud-native access for innovation workloads.

AWS implementation approaches

AWS supports both model access patterns through complementary services:

Cloud native pattern

- Amazon Bedrock – managed foundation model service providing unified API access to multiple models with built-in guardrails, security controls, and enterprise features

- Amazon SageMaker – platform for deploying and hosting custom or third-party models with full control over infrastructure and model serving

Gateway pattern

- AWS Guidance for Multi-Provider Generative AI Gateway - reference architecture providing centralized access layer across Amazon Bedrock, Amazon SageMaker, and third-party model providers with unified usage tracking, cost management, rate limiting, model routing, and governance controls

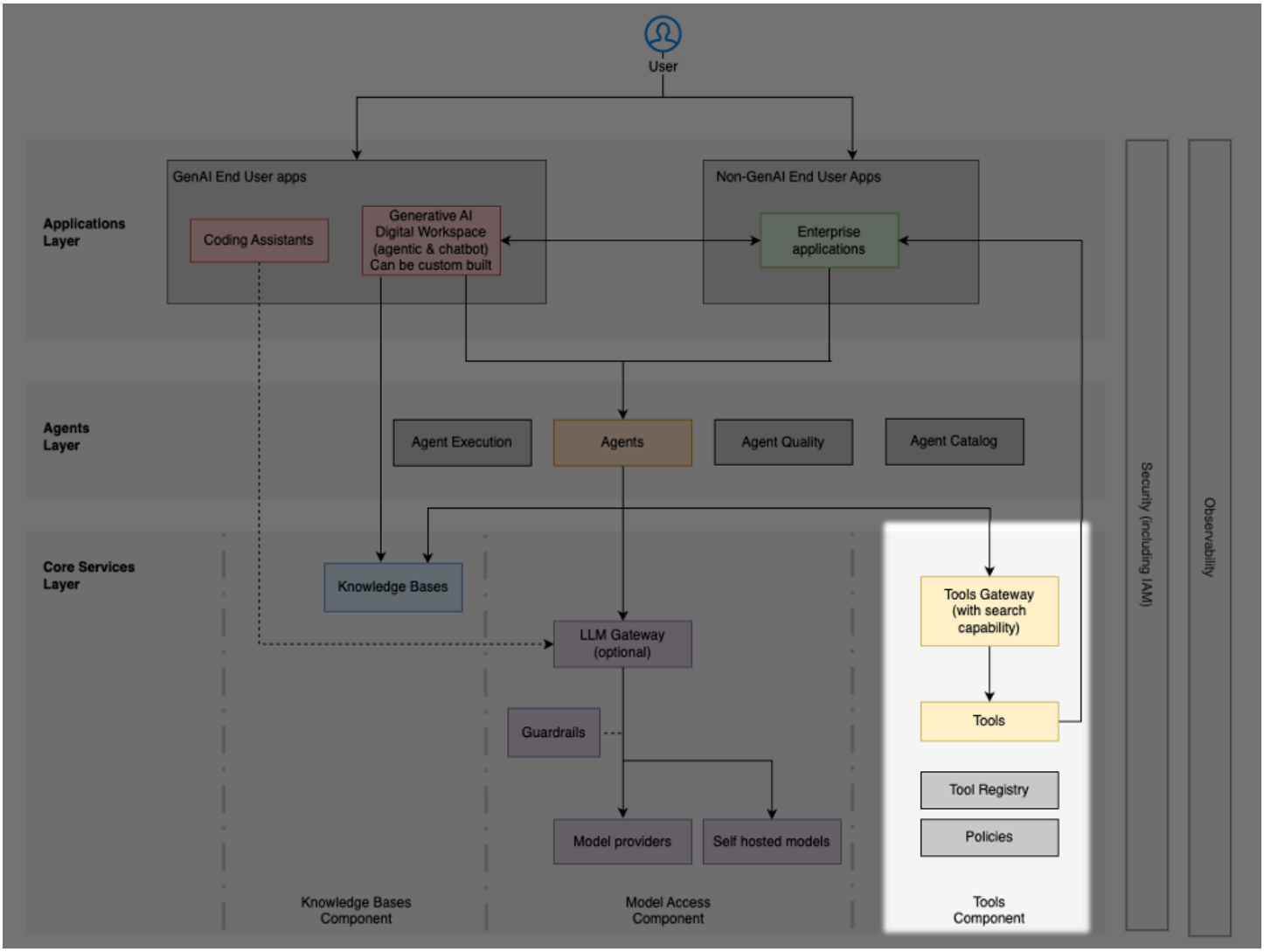
Guardrails implementation

Regardless of pattern, organizations must implement guardrails that filter inappropriate content, enforce compliance requirements, validate inputs and outputs to prevent prompt injection, and support domain-specific customization.

Amazon BedrockGuardrails provides managed capabilities for content filtering, denied topics, word filters, and sensitive information redaction.

Core services: tools

The tools component enables agents to interact with external systems, execute functions, and perform actions beyond language processing. This layer bridges the gap between agent reasoning and real-world operations, providing the mechanisms through which agents retrieve data, invoke business logic, and integrate with enterprise systems.



Architecture patterns

Organizations can implement tool access through three primary architectural patterns, each offering tradeoffs between simplicity, governance, and operational control:

In-runtime tools

Tools execute directly within the agent's runtime environment as inline function calls.

This pattern is characterized by:

- Minimal latency for simple operations
- Simplified architecture with fewer components
- Suitable for lightweight, stateless tool operations

- Direct integration within the agent execution context

Direct access with remote tools

Agents invoke tools through direct connections to remote services using standardized protocols such as the Model Context Protocol (MCP). Tools run in separate processes or services but are accessed without intermediary gateways.

This pattern provides:

- Separation of concerns between agent logic and tool execution
- Support for both local and remote tool deployment
- Protocol-based interoperability across frameworks
- Better isolation while maintaining reasonable latency

Tools gateway pattern

A centralized gateway service manages tool discovery, security, versioning, and invocation. Agents dispatch requests to the gateway, which forwards them to the tools.

This pattern delivers:

- Centralized tool registry and discovery - agents can search and discover available tools through a unified catalog
- Standardized security monitoring and access control - consistent authentication, authorization, and audit across all tool invocations
- Version management and lifecycle control - support for tool evolution, deprecation, and migration without breaking agent workflows
- Governance enablement - approval workflows, usage tracking, quality validation, and compliance controls

Choosing the right pattern

The choice between these patterns mirrors the decision between direct API invocation and API management in traditional enterprise architectures – organizations must balance architectural

simplicity against governance requirements, scalability needs, and operational control. However, agentic systems introduce a unique constraint: unlike traditional software with predetermined API calls, LLMs must dynamically select appropriate tools at runtime, based on their description.

The table below compares these patterns across key dimensions. The choice depends primarily on three factors:

- Governance requirements - Organizations requiring centralized control, approval workflows, and comprehensive audit trails typically favor the Tools Gateway pattern, while those prioritizing development velocity may begin with direct access approaches
- Tool catalog scale - As the number of available tools grows, providing all tool descriptions within the LLM's context window becomes impractical and increases hallucination rates and selection errors. Use cases with a small, stable set of tools can work without centralized discovery, but scalability on the number of tools requires search and discovery capabilities such as those provided by a gateway pattern
- Operational maturity - Teams may start with direct access for rapid prototyping and migrate to gateway-based approaches as their tool ecosystem and governance requirements mature

Factor	In-runtime tools	Direct access with remote tools	Tools Gateway pattern
Latency	Lowest – inline function calls within the agent runtime	Low – direct connections, but network overhead for remote calls	Higher – additiona l hop through the gateway
Architecture complexity	Simplest – fewest components	Moderate – separate processes/services, but no intermediary	Most complex – requires gateway infrastructure
Governance and access control	None built in – must be implemented per tool	Limited – depends on individual tool/serv ice controls	Centralized – approval workflows , usage tracking, compliance controls
Audit and monitorin g	Manual – no centraliz ed visibility	Per-service – inconsist ent across tools	Centralized – consistent authentic

			ation, authorization, and audit
Tool discovery	N/A – tools are hardcoded in the runtime	Manual – agents must know tool endpoints	Unified catalog – agents search and discover tools dynamically
Version management	Manual – updates require redeployment	Per-service – each tool manages its own versioning	Centralized – lifecycle control, deprecation, and migration support
Scalability (number of tools)	Limited – all tools must fit in the agent context	Moderate – works with a small, stable set of tools	Best – search and discovery prevent context window overload and reduce hallucination
Isolation	None – tools share the agent's runtime	Good – separate processes provide fault isolation	Good – gateway adds an additional isolation layer
Best suited for	Lightweight, stateless operations with a small tool set	Small teams prototyping with a known set of remote tools	Enterprise environments needing governance, scale, and operational control
Typical maturity stage	Early exploration	Rapid prototyping and growing maturity	Production at scale

Governance considerations

The tools layer requires governance controls for:

- Tool registration and approval workflows - maintain catalogs of authorized tools with quality control and lifecycle management

- Access control policies - define which agents can invoke specific tools, with fine-grained permission scoping
- Usage tracking and audit logs - comprehensive logging of all tool invocations for compliance and cost management
- Version management - support tool evolution without breaking existing agent workflows, including deprecation and migration paths
- Quality validation - ensure tool reliability, performance standards, and security compliance
- Discovery and search capabilities - enable agents to find appropriate tools through searchable registries

Implementation on AWS

The Tools gateway pattern is implemented using Amazon Bedrock AgentCore gateway, which serves as a centralized tool server providing a unified interface where agents can discover, access, and invoke tools.

AgentCore Gateway provides:

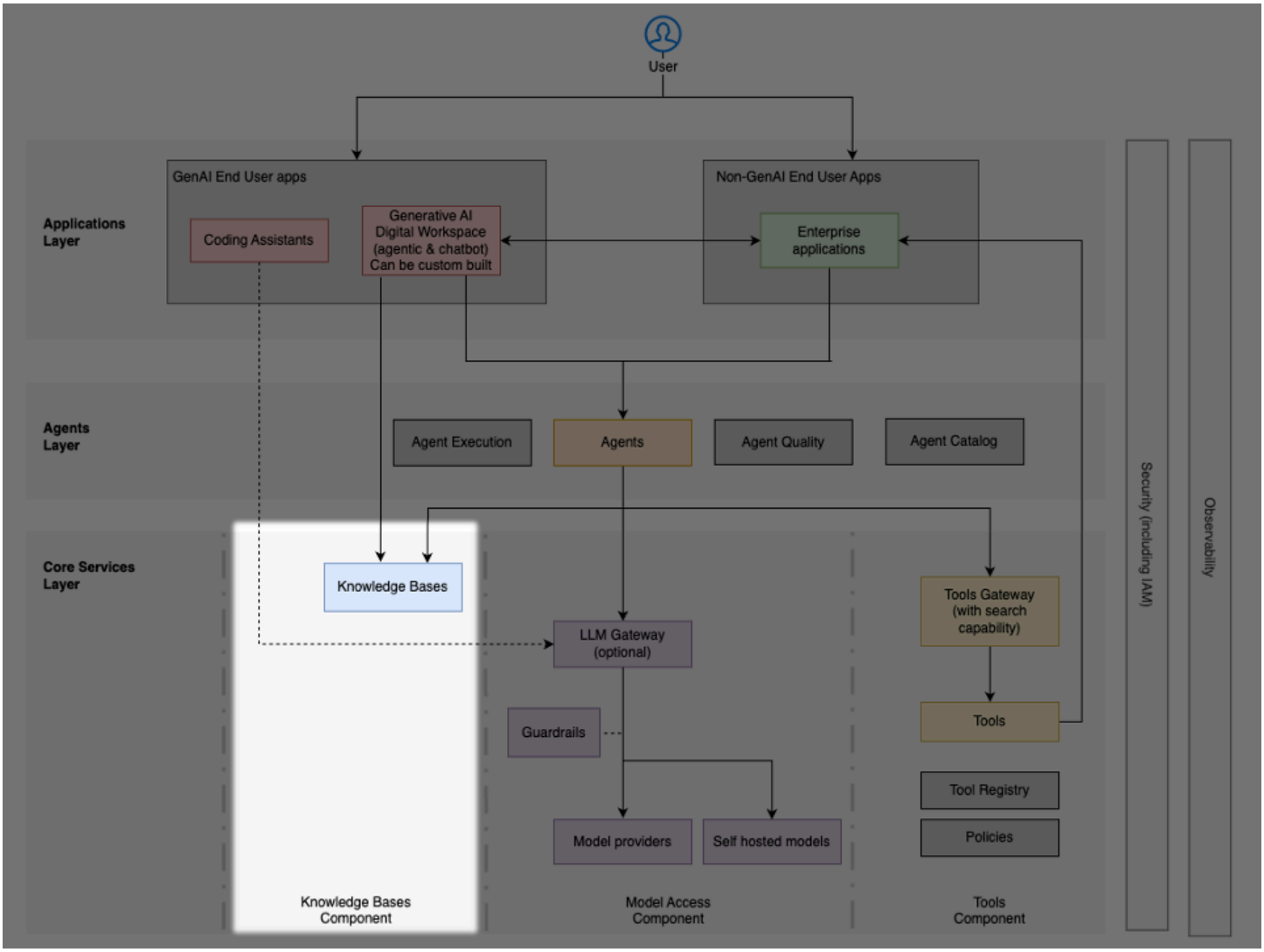
- Native MCP support - built with native support for the Model Context Protocol, enabling seamless agent-to-tool communication while abstracting away security, infrastructure, and protocol-level complexities
- REST API transformation - converts existing REST APIs into MCP servers, allowing legacy systems to become discoverable tools
- MCP server integration - treats existing MCP servers as native targets, providing a single point of control for routing, authentication, and tool management
- Centralized tool discovery - agents can search and discover available tools through a unified catalog across multiple targets
- Target-based architecture - each target (API, MCP server, or service) exposes multiple tools, with each function or path/method becoming a discoverable tool
- Security and governance - handles authentication, authorization, and audit logging across all tool invocations

Amazon Bedrock AgentCore code execution and browser Tools - In addition to the gateway pattern, Amazon Bedrock AgentCore provides secure and scalable access to isolated code execution and

web browsing environments, enabling agents to perform computational tasks and interact with web content without requiring infrastructure provisioning or management.

Core services: knowledge bases

The knowledge bases component provides agents with access to enterprise data and domain-specific information through Retrieval Augmented Generation (RAG). RAG enables agents to ground their responses in factual, up-to-date information from organizational data sources, reducing hallucinations and enabling domain-specific intelligence without requiring model retraining.



RAG combines two processes - agents retrieve relevant information from a knowledge base through semantic search, then provide that retrieved context to the LLM to generate grounded responses.

Implementation on AWS

Amazon Bedrock Knowledge Bases

Amazon Bedrock Knowledge Bases provides fully managed RAG capabilities, handling document ingestion, chunking, embedding generation, vector storage, and retrieval without requiring custom pipeline development. The service automates the entire workflow from data sources through to context-enhanced LLM prompts.

Vector database options

Organizations select vector databases based on specific requirements:

- Amazon OpenSearch Service - best for applications requiring full-text search combined with vector similarity, supporting billions of vectors with millisecond query performance. Ideal when search must integrate with analytics and observability use cases.
- Amazon Aurora PostgreSQL with pgvector - optimal when vector search must coexist with transactional data, leveraging existing PostgreSQL expertise and infrastructure.
- Amazon S3 Vectors - cost-effective for massive-scale vector storage requirements. Reduces storage and search costs by up to 90% compared to traditional vector databases, with sub-second query performance.

Supporting infrastructure includes Amazon S3 for document storage, AWS Lambda for custom retrieval logic and preprocessing, and Amazon Kendra for intelligent enterprise search with natural language understanding.

Graph database

For use cases where connections between data entities are central to queries and analysis:

- Amazon Bedrock Knowledge Bases with GraphRAG – fully managed GraphRAG capability, automatically creates embeddings for semantic search and generates graphs of entities and relationships from unstructured data, and combines vector search with graph traversal. Ideal for applications requiring both semantic similarity and relationship-based retrieval
- Amazon Neptune Analytics – in-memory graph analytics engine for fast analysis of large graph datasets, supports openCypher, and analyzes tens of billions of relationships in seconds

- Amazon Neptune – Persistent graph database supporting property graphs (Gremlin, openCypher) and RDF graphs (SPARQL). Optimized for operational workloads requiring low-latency transactional access

Integration with agents

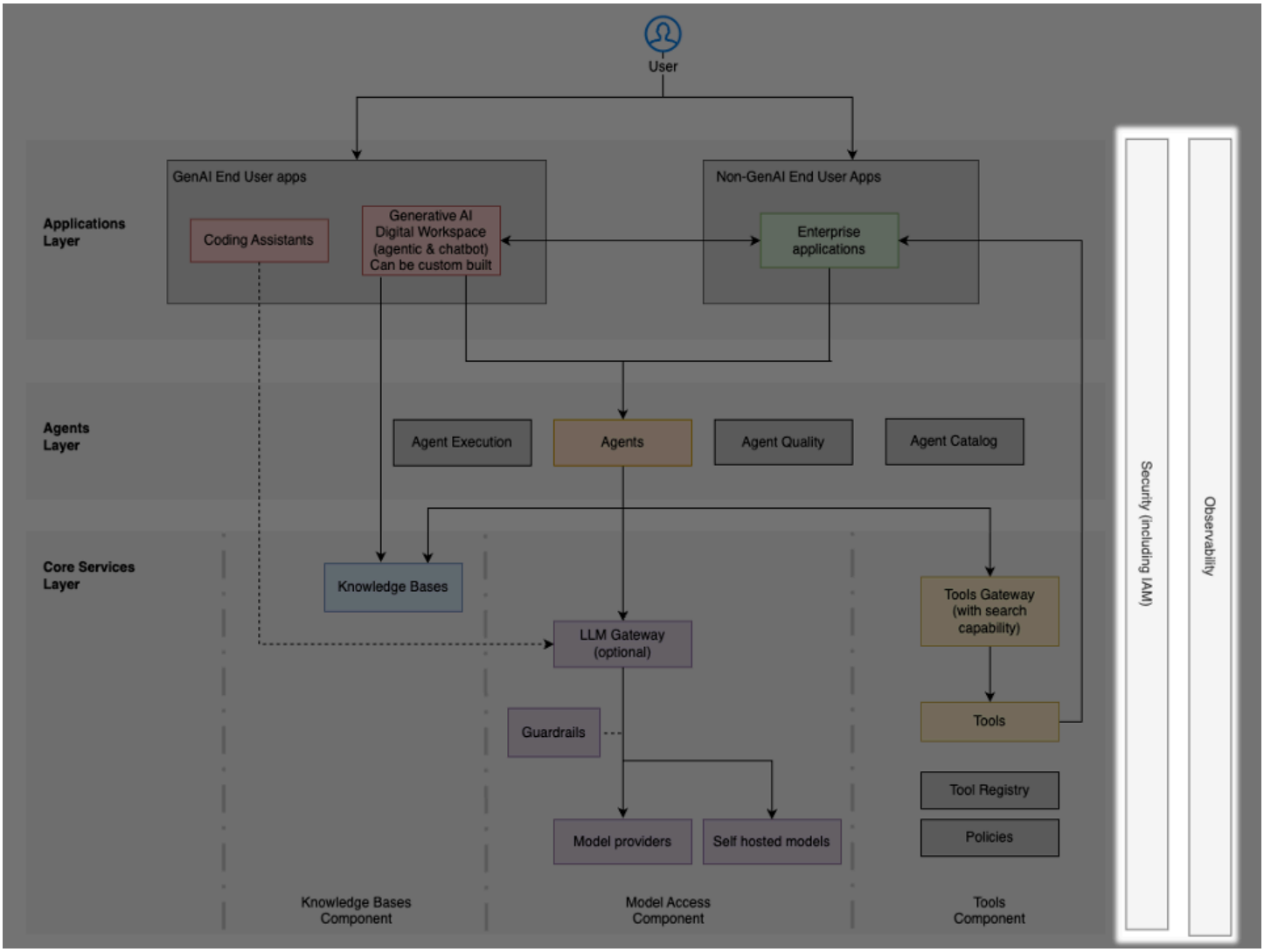
Agents interact with knowledge bases through a retrieval and generation workflow. The agent analyzes the user request and formulates retrieval queries. The knowledge base performs a similarity search, returning relevant documents or passages with relevance scores. Retrieved information is ranked and assembled into context for the LLM prompt. The LLM generates responses grounded in the retrieved knowledge, with citations to source documents.

Governance considerations

The knowledge bases layer requires governance controls for data lineage tracking from source documents through retrieval to agent responses, enabling transparency and compliance. Access control must ensure knowledge base permissions align with source data access policies, with agents respecting organizational data boundaries. Data retention and lifecycle management requires synchronization between source data updates and vector index refreshes, with policies for data deletion and archival. Organizations monitor retrieval quality metrics including relevance scores and retrieval accuracy to ensure knowledge base effectiveness. Source attribution mechanisms enable citing source documents in agent responses, supporting verification and compliance requirements. For shared knowledge bases, multi-tenancy isolation ensures proper data segregation and access control across organizational boundaries.

Cross-layer concerns

Observability and security span all layers of the agentic AI architecture, ensuring that AI operations are monitored, auditable, and compliant with enterprise policies. These foundational elements must be integrated throughout the entire system rather than treated as isolated components.



Observability

Comprehensive observability enables organizations to understand agent behavior, track performance, diagnose issues, and validate compliance across the complete agentic system through end-to-end request tracing across agents, LLMs, tools, and knowledge bases; performance monitoring of response times, success rates, and resource utilization; cost tracking with granular attribution to business units or applications; quality monitoring for response accuracy, hallucination rates, and guideline compliance; and usage analytics providing insights into adoption patterns and business value delivery.

Supporting infrastructure includes:

- Amazon Bedrock AgentCore - observability providing out-of-the box monitoring and tracing of AgentCore services and support for export in OpenTelemetry format for interoperability with third-party observability solutions.
- Amazon CloudWatch - centralized metrics, logs, and alarms with custom dashboards for operational visibility
- Amazon OpenSearch Service - advanced log analytics and visualization for complex pattern detection
- Amazon EventBridge - event-driven alerting and response automation
- AWS Cost Explorer Service and AWS Budgets - detailed cost allocation and management across all agentic AI infrastructure components including model inference, compute, storage, and data transfer

Security

Comprehensive security ensures that agentic AI systems maintain:

- Data protection, access control, compliance with regulations, and protection against AI-specific threats through identity and access management with proper authentication and authorization
- Data protection safeguarding sensitive information in inputs, outputs, and knowledge bases
- Prompt security preventing injection attacks and manipulation
- Supply chain security validating models and dependencies
- Network isolation controlling communication paths between components.

Supporting infrastructure includes:

- IAM and IAM Identity Center - enterprise identity management with fine-grained access control and federated authentication
- Amazon BedrockGuardrails - content filtering, topic constraints, and PII protection for model interactions
- Amazon Bedrock AgentCore Identity - Secure identity propagation across agent interactions and tool invocations
- AWS KMS key- encryption key management for data at rest and in transit
- Amazon Macie - automated discovery and protection of sensitive data in knowledge bases

- AWS WAF (Web Application Firewall) - protection against web-based attacks for exposed agent interfaces
- Amazon Virtual Private Cloud(VPC) and AWS PrivateLink - network isolation and private connectivity between components
- AWS Security Hub - centralized security posture management across the AI ecosystem
- AWS CloudTrail - comprehensive API activity logging for compliance and audit

Resources

- Generative AI Lens - AWS Well-Architected Framework
- Levels in the generative AI maturity model
- Guidance for Multi-Provider Generative AI Gateway on AWS
- Building trust in AI: The AWSapproach to the EU AI Act

Contributors

Authoring

- Lior Perez, Principal Senior Solutions Architect, AWS
- Jocelyn Pinault, Senior Manager, CSM, AWS
- Sebastien Grazzini, Principal Solutions Architect, AWS
- Florent Lacroute, Senior Solutions Architect, AWS
- Massimiliano Angelino, Principal Solutions Architect, AWS

Reviewing

- Ioan Catana, Senior Solutions Architect Specialist, AWS

Technical writing

- Nathan Bigman, Senior Technical Writer, AWS

Document history

The following table describes significant changes to this guide.

Change	Description	Date
Initial publication	—	April 23, 2026