Governing and architecting the diversity of Agentic AI at scale

AWS Prescriptive Guidance

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

Table of Contents

Governing and architecting the diversity of Agentic AI at scale

Amazon Web Services (contributors)

April 2026 (document history)

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 AWS guides in this series, see Agentic AI on the 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:

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

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.

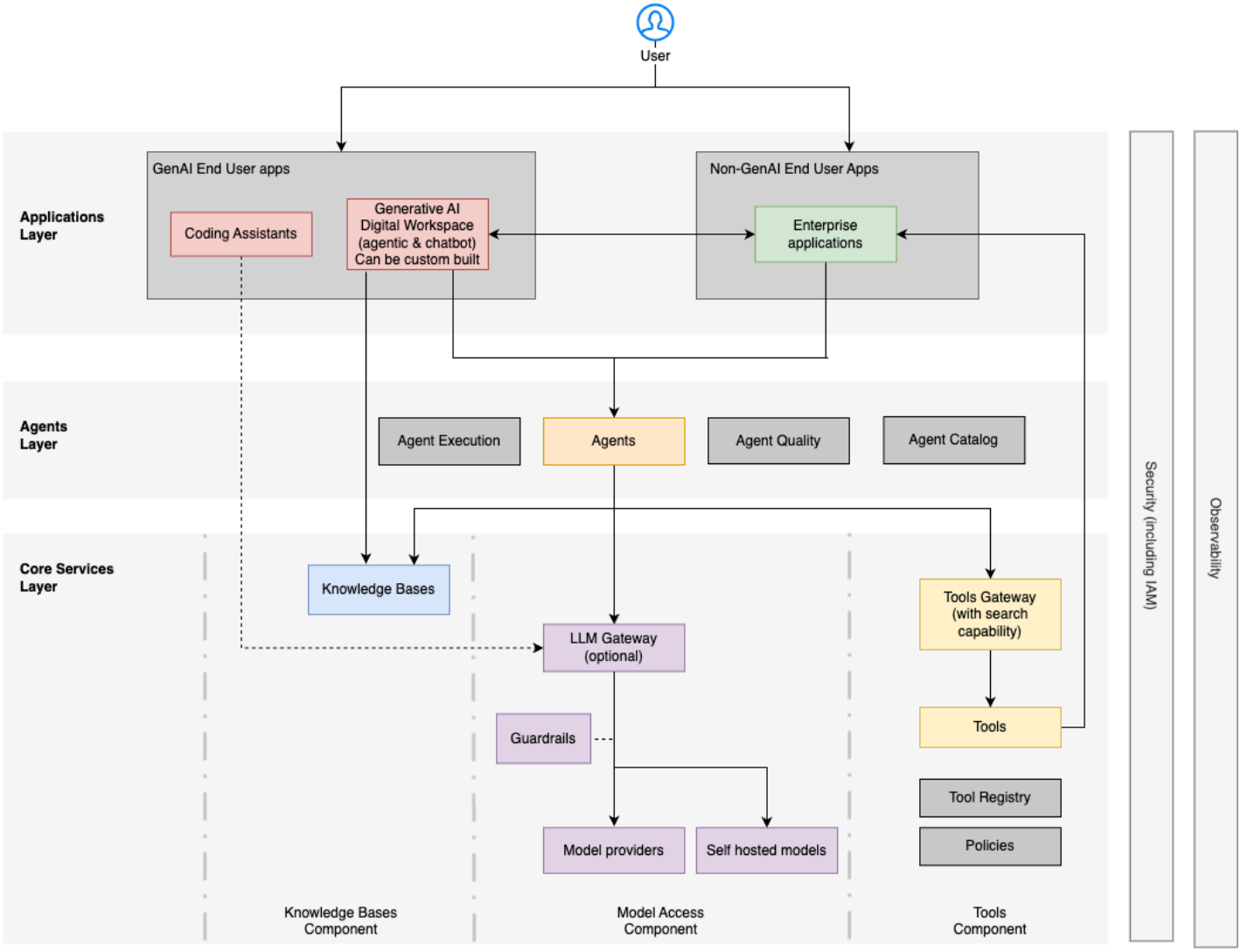
Provide foundational training on AI fundamentals, responsible AI principles, security considerations, and governance requirements. Training programs should emphasize principles over rigid rules, teaching developers to think critically about risks and trade-offs. Role-based certification programs match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

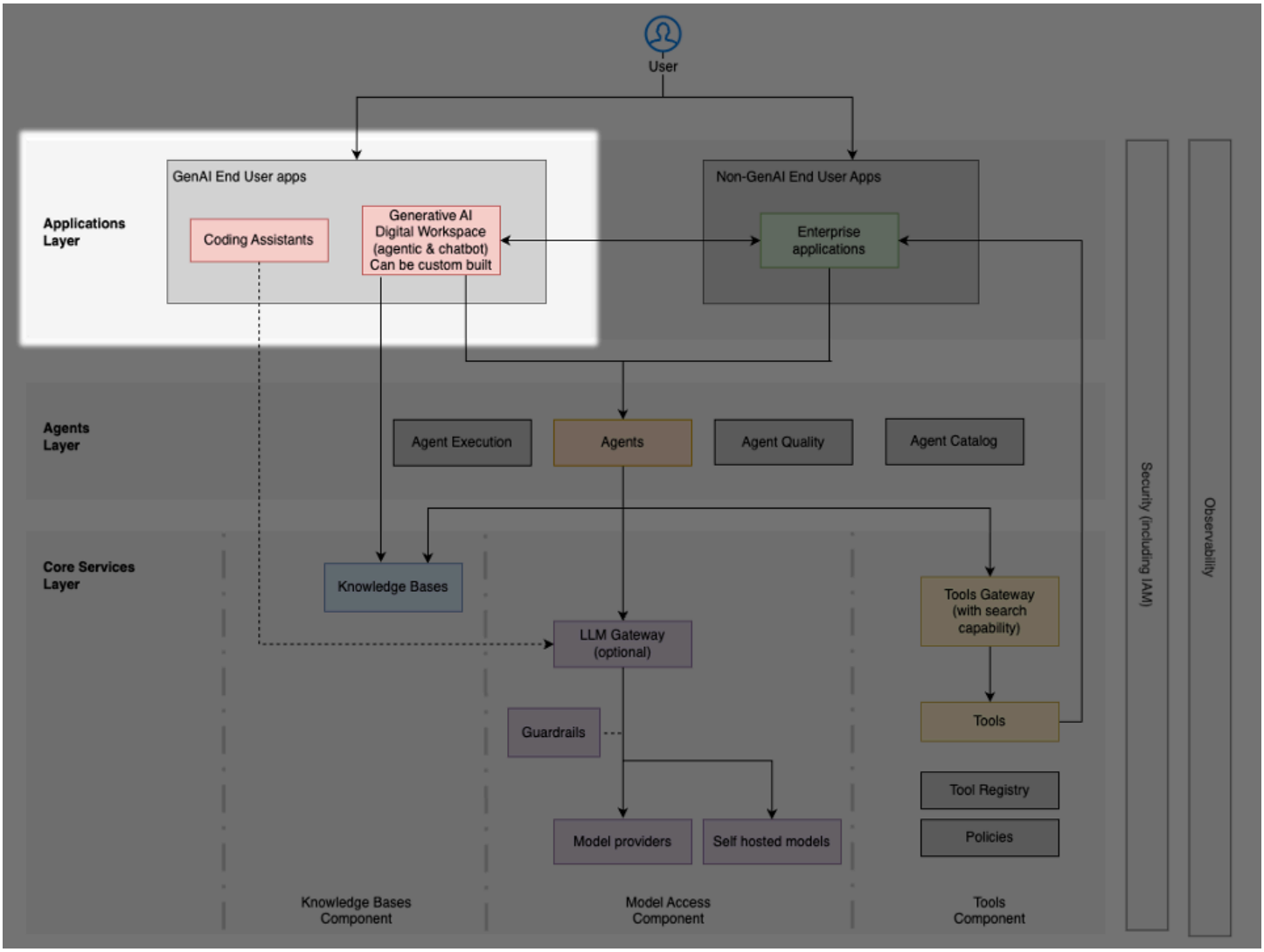
- **Model zccess component** – controls access to foundation models with policy enforcement, safety measures including guardrails, and cost tracking/allocation capabilities.
- **Tools component** – manages discovery and secure execution of tools. This component provides authorization capabilities to 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 IAM Identity Center, OIDC and OAuth 2.0 with integration with enterprise identity providers for secure end-user access control

Implementation approaches

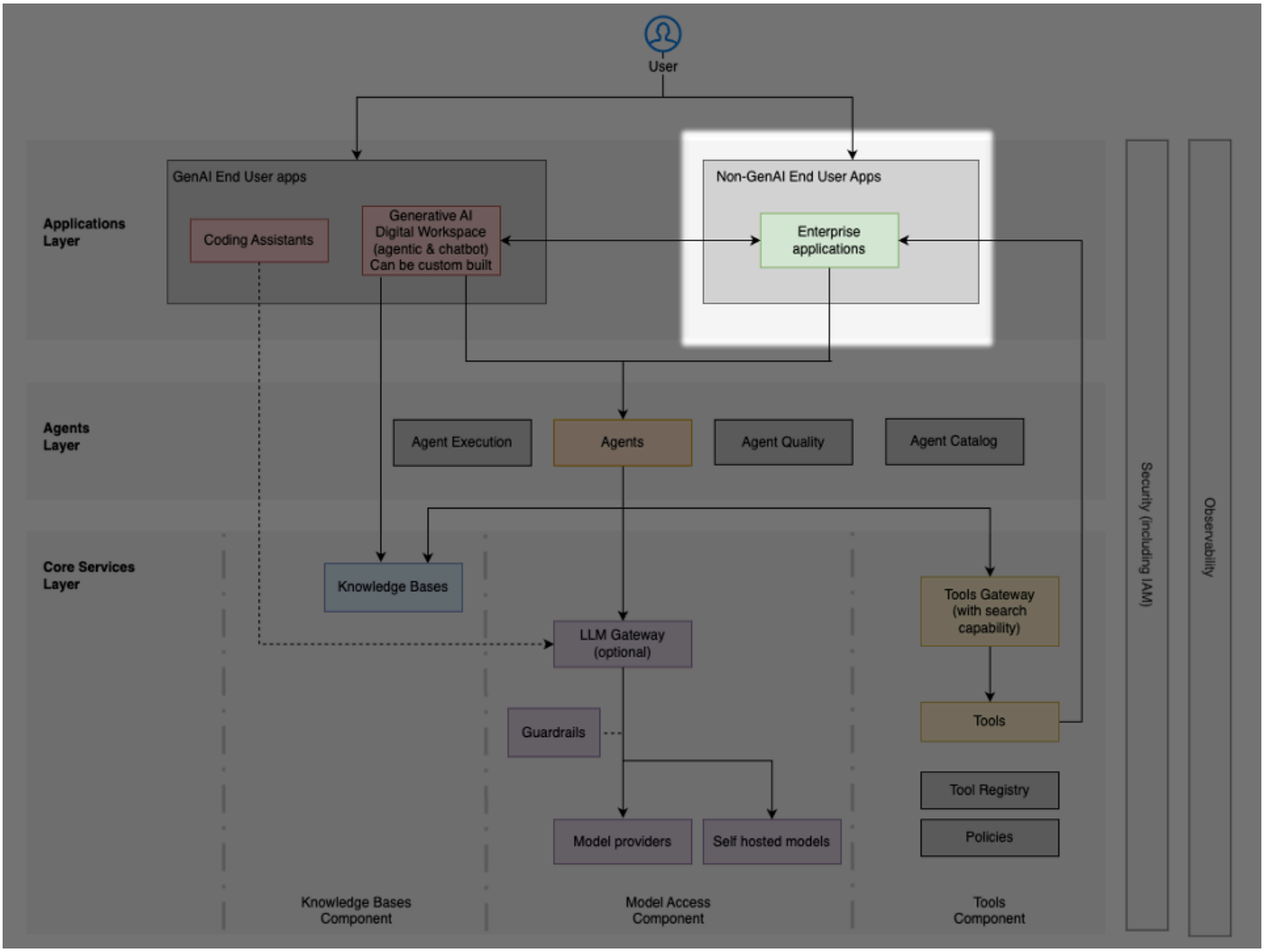
Organizations can choose between two primary approaches:

- 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

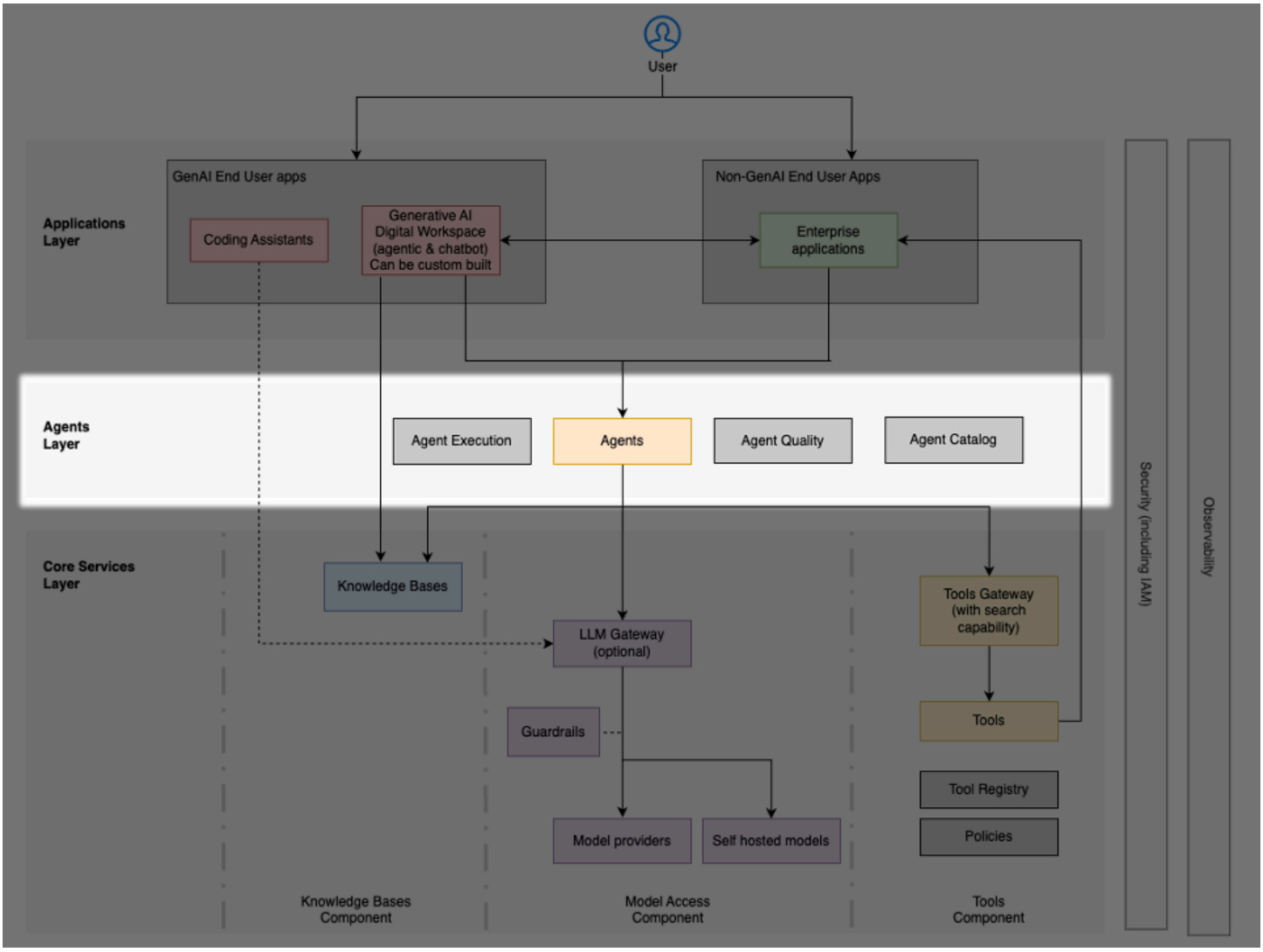
AWS provides several services to enable enterprise application integration:

- 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

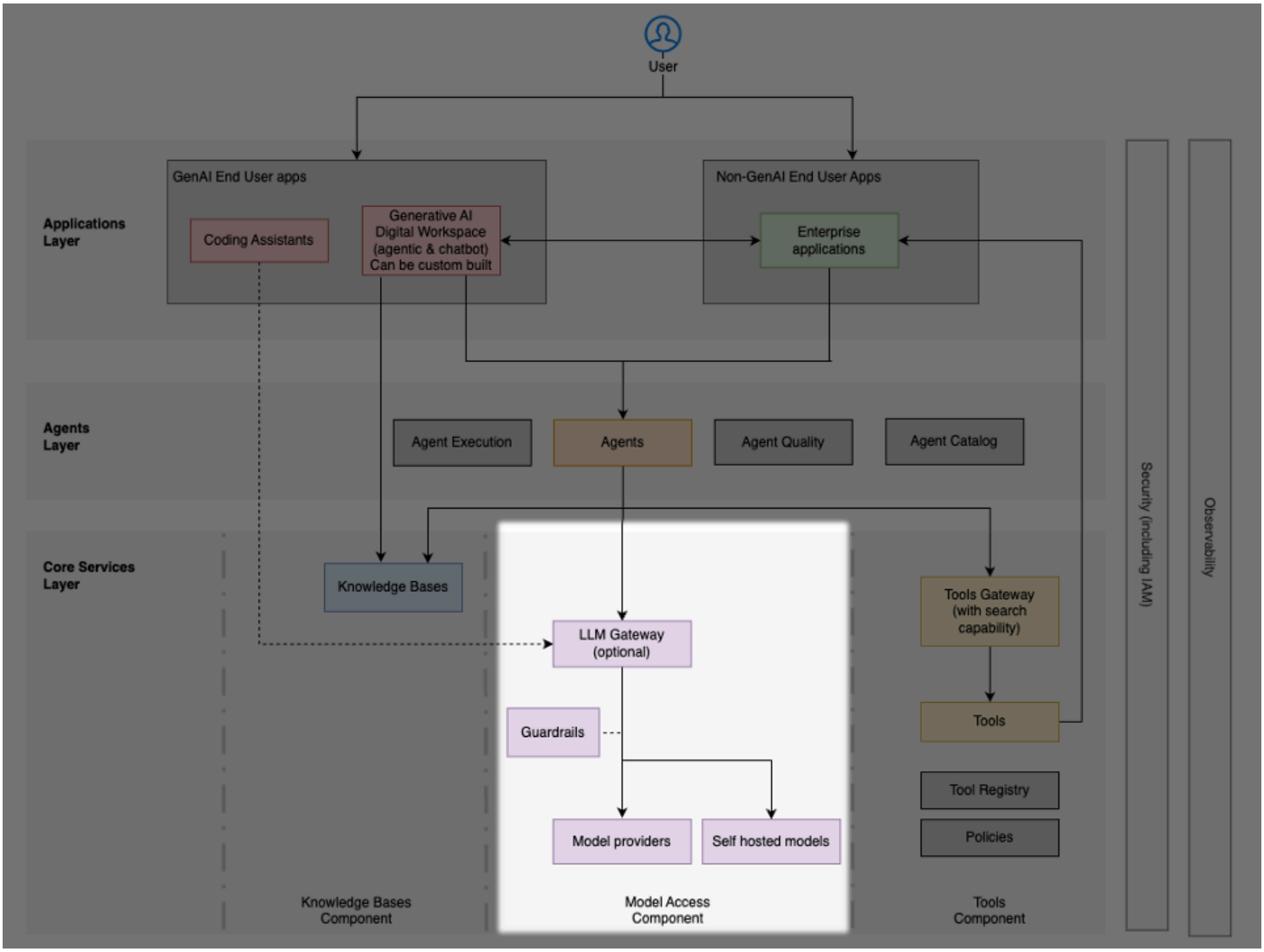
Effective agent operations require robust identity and permission management through identity propagation from users through agent chains, permission boundaries for agent actions and tool access, audit trails of agent decisions and actions, and circuit breakers for abnormal behavior patterns.

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

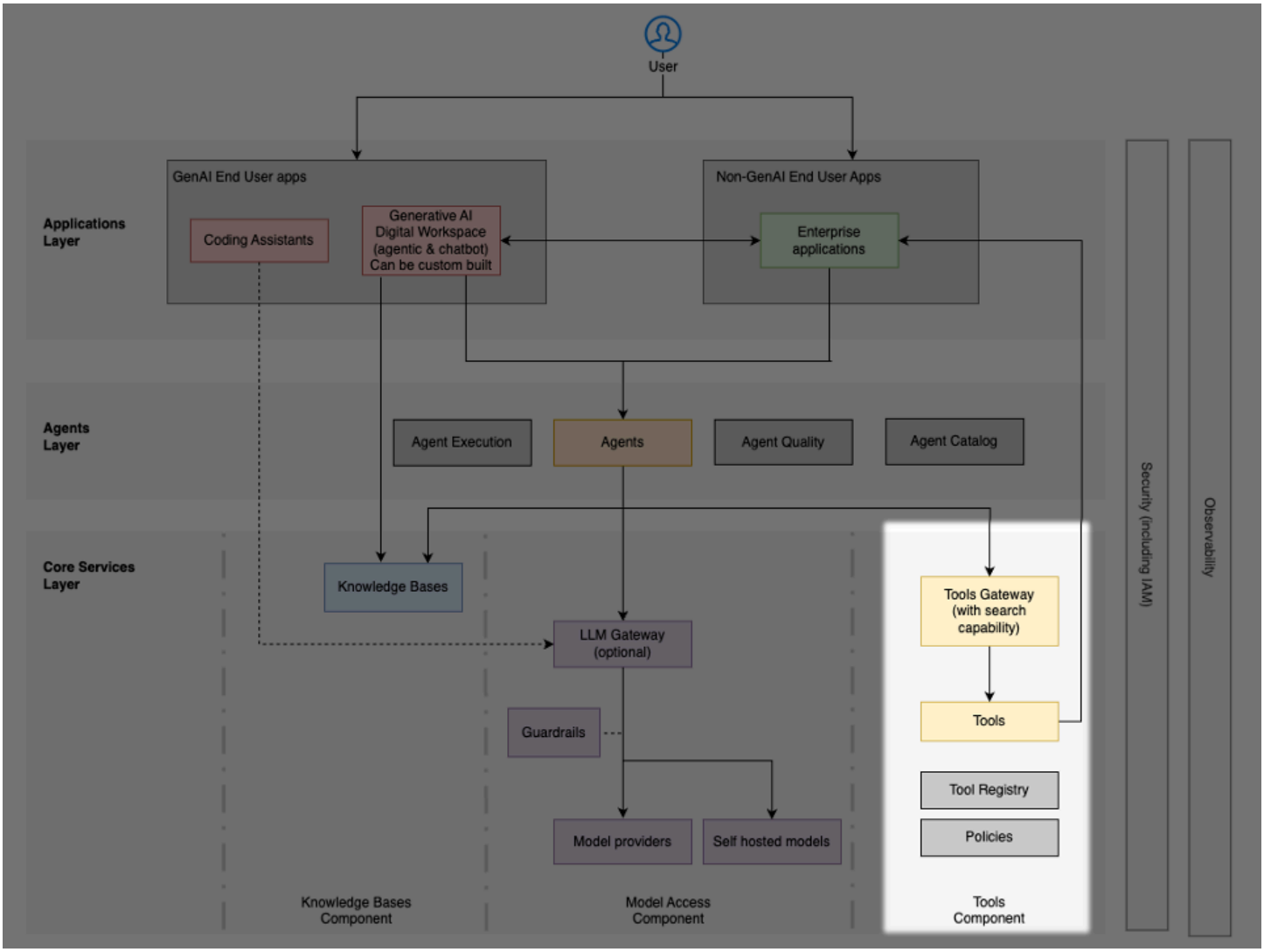
Guardrails implementation

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:

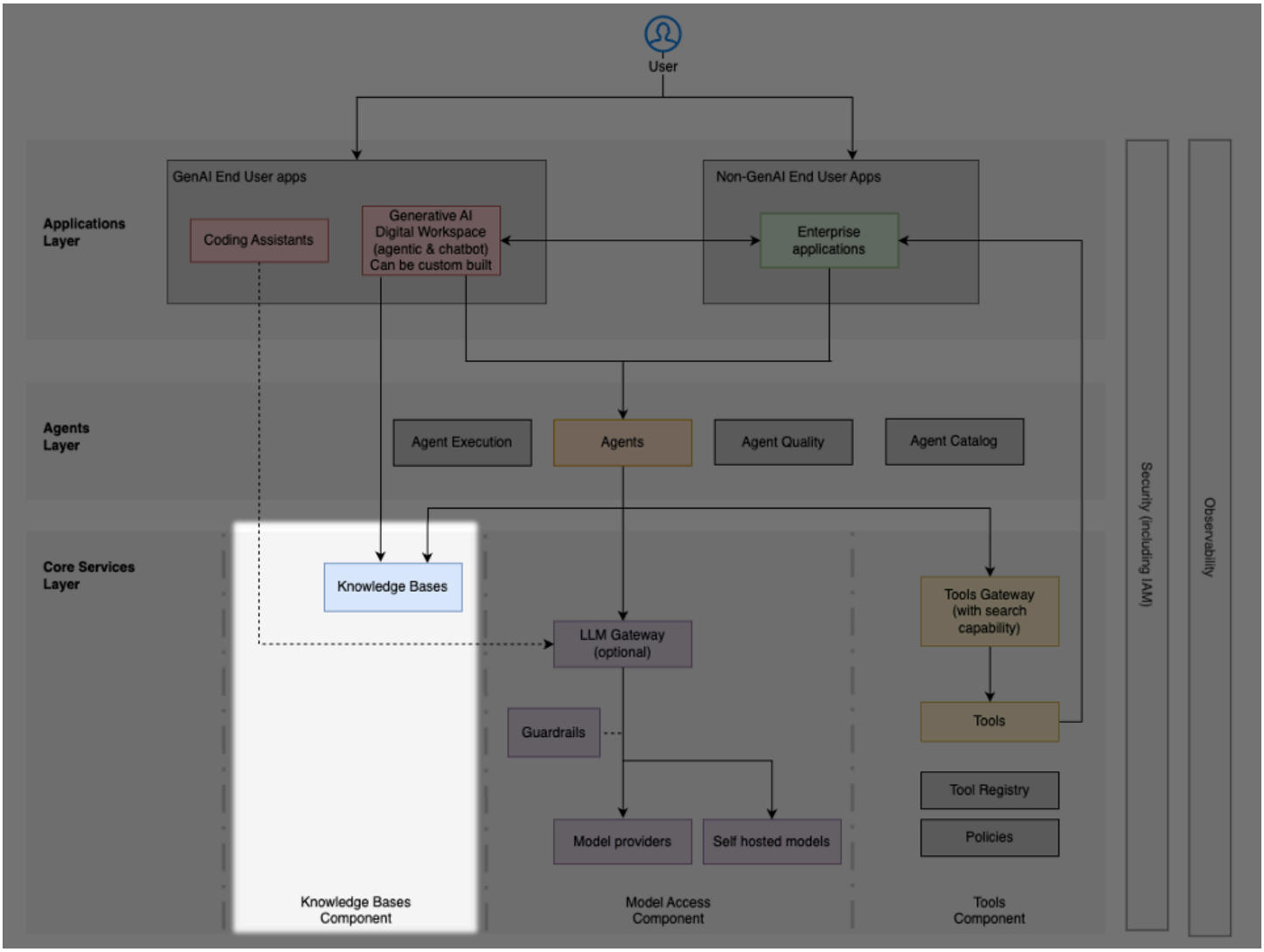
- Native MCP support - built with native support for the Model Context Protocol, enabling seamless agent-to-tool communication while abstracting away security, infrastructure, and protocol-level complexities
- REST API transformation - converts existing REST APIs into MCP servers, allowing legacy systems 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

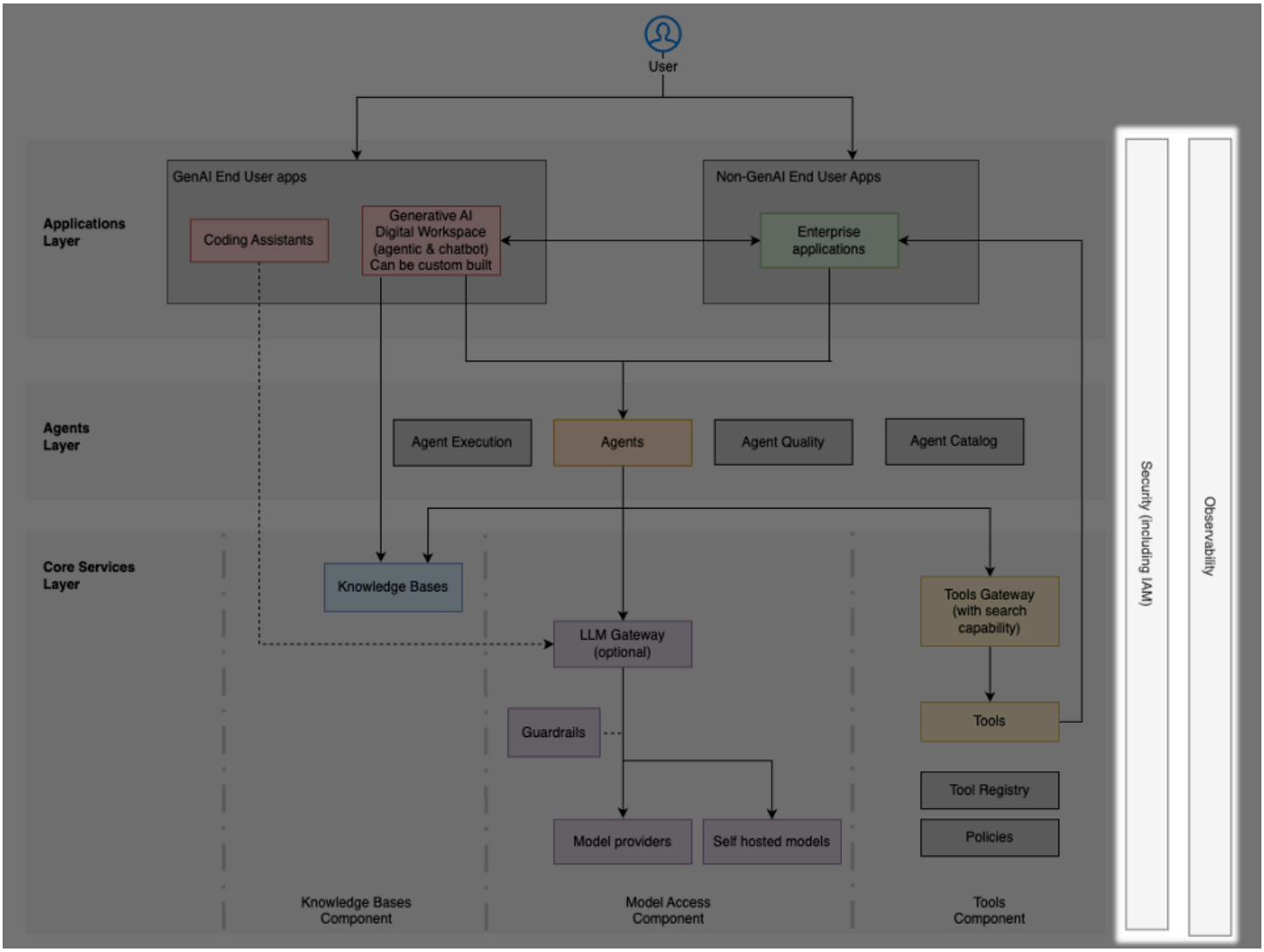
Agents interact with knowledge bases through a retrieval and generation workflow. The agent analyzes the user request and formulates retrieval queries. The knowledge base performs a similarity search, returning relevant documents or passages with relevance scores. Retrieved information is ranked and 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

AWS Prescriptive Guidance glossary

The following are commonly used terms in strategies, guides, and patterns provided by AWS Prescriptive Guidance. To suggest entries, please use the **Provide feedback** link at the end of the glossary.

Numbers

7 Rs

Seven common migration strategies for moving applications to the cloud. These strategies build upon the 5 Rs that Gartner identified in 2011 and consist of the following:

- Refactor/re-architect – Move an application and modify its architecture by taking full advantage of cloud-native features to improve agility, performance, and scalability. This typically involves porting the operating system and database. Example: Migrate your on-premises Oracle database to the Amazon Aurora PostgreSQL-Compatible Edition.
- Replatform (lift and reshape) – Move an application to the cloud, and introduce some level of optimization to take advantage of cloud capabilities. Example: Migrate your on-premises Oracle database to Amazon Relational Database Service (Amazon RDS) for Oracle in the AWS Cloud.
- Repurchase (drop and shop) – Switch to a different product, typically by moving from a traditional license to a SaaS model. Example: Migrate your customer relationship management (CRM) system to Salesforce.com.
- Rehost (lift and shift) – Move an application to the cloud without making any changes to take advantage of cloud capabilities. Example: Migrate your on-premises Oracle database to Oracle on an EC2 instance in the AWS Cloud.
- Relocate (hypervisor-level lift and shift) – Move infrastructure to the cloud without purchasing new hardware, rewriting applications, or modifying your existing operations. You migrate servers from an on-premises platform to a cloud service for the same platform. Example: Migrate a Microsoft Hyper-V application to AWS.
- Retain (revisit) – Keep applications in your source environment. These might include applications that require major refactoring, and you want to postpone that work until a later time, and legacy applications that you want to retain, because there's no business justification for migrating them.

- Retire – Decommission or remove applications that are no longer needed in your source environment.

A

A2A (Agent-to-Agent)

A stateful protocol for agent-to-agent collaboration supporting task delegation and state transfer.

ABAC

See attribute-based access control.

abstracted services

See managed services.

ACID

See atomicity, consistency, isolation, durability.

active-active migration

A database migration method in which the source and target databases are kept in sync (by using a bidirectional replication tool or dual write operations), and both databases handle transactions from connecting applications during migration. This method supports migration in small, controlled batches instead of requiring a one-time cutover. It's more flexible but requires more work than active-passive migration.

active-passive migration

A database migration method in which the source and target databases are kept in sync, but only the source database handles transactions from connecting applications while data is replicated to the target database. The target database doesn't accept any transactions during migration.

Agent

An AI system that can autonomously reason, plan, and take actions using tools to achieve goals.

Agent Ops

Operational practices for building, testing, deploying, and running AI agents in production at scale.

aggregate function

A SQL function that operates on a group of rows and calculates a single return value for the group. Examples of aggregate functions include `SUM` and `MAX`.

AI

See artificial intelligence.

AIOps

See artificial intelligence operations.

anonymization

The process of permanently deleting personal information in a dataset. Anonymization can help protect personal privacy. Anonymized data is no longer considered to be personal data.

anti-pattern

A frequently used solution for a recurring issue where the solution is counter-productive, ineffective, or less effective than an alternative.

application control

A security approach that allows the use of only approved applications in order to help protect a system from malware.

application portfolio

A collection of detailed information about each application used by an organization, including the cost to build and maintain the application, and its business value. This information is key to the portfolio discovery and analysis process and helps identify and prioritize the applications to be migrated, modernized, and optimized.

artificial intelligence (AI)

The field of computer science that is dedicated to using computing technologies to perform cognitive functions that are typically associated with humans, such as learning, solving problems, and recognizing patterns. For more information, see What is Artificial Intelligence?

artificial intelligence operations (AIOps)

The process of using machine learning techniques to solve operational problems, reduce operational incidents and human intervention, and increase service quality. For more information about how AIOps is used in the AWS migration strategy, see the operations integration guide.

asymmetric encryption

An encryption algorithm that uses a pair of keys, a public key for encryption and a private key for decryption. You can share the public key because it isn't used for decryption, but access to the private key should be highly restricted.

atomicity, consistency, isolation, durability (ACID)

A set of software properties that guarantee the data validity and operational reliability of a database, even in the case of errors, power failures, or other problems.

attribute-based access control (ABAC)

The practice of creating fine-grained permissions based on user attributes, such as department, job role, and team name. For more information, see ABAC for AWS in the AWS Identity and Access Management (IAM) documentation.

authoritative data source

A location where you store the primary version of data, which is considered to be the most reliable source of information. You can copy data from the authoritative data source to other locations for the purposes of processing or modifying the data, such as anonymizing, redacting, or pseudonymizing it.

Availability Zone

A distinct location within an AWS Region that is insulated from failures in other Availability Zones and provides inexpensive, low-latency network connectivity to other Availability Zones in the same Region.

AWS Cloud Adoption Framework (AWS CAF)

A framework of guidelines and best practices from AWS to help organizations develop an efficient and effective plan to move successfully to the cloud. AWS CAF organizes guidance into six focus areas called perspectives: business, people, governance, platform, security, and operations. The business, people, and governance perspectives focus on business skills and processes; the platform, security, and operations perspectives focus on technical skills and processes. For example, the people perspective targets stakeholders who handle human resources (HR), staffing functions, and people management. For this perspective, AWS CAF provides guidance for people development, training, and communications to help ready the organization for successful cloud adoption. For more information, see the AWS CAF website and the AWS CAF whitepaper.

AWS Workload Qualification Framework (AWS WQF)

A tool that evaluates database migration workloads, recommends migration strategies, and provides work estimates. AWS WQF is included with AWS Schema Conversion Tool (AWS SCT). It analyzes database schemas and code objects, application code, dependencies, and performance characteristics, and provides assessment reports.

B

bad bot

A bot that is intended to disrupt or cause harm to individuals or organizations.

BCP

See business continuity planning.

behavior graph

A unified, interactive view of resource behavior and interactions over time. You can use a behavior graph with Amazon Detective to examine failed logon attempts, suspicious API calls, and similar actions. For more information, see Data in a behavior graph in the Detective documentation.

big-endian system

A system that stores the most significant byte first. See also endianness.

binary classification

A process that predicts a binary outcome (one of two possible classes). For example, your ML model might need to predict problems such as “Is this email spam or not spam?" or "Is this product a book or a car?"

bloom filter

A probabilistic, memory-efficient data structure that is used to test whether an element is a member of a set.

blue/green deployment

A deployment strategy where you create two separate but identical environments. You run the current application version in one environment (blue) and the new application version in the other environment (green). This strategy helps you quickly roll back with minimal impact.

bot

A software application that runs automated tasks over the internet and simulates human activity or interaction. Some bots are useful or beneficial, such as web crawlers that index information on the internet. Some other bots, known as *bad bots*, are intended to disrupt or cause harm to individuals or organizations.

botnet

Networks of bots that are infected by malware and are under the control of a single party, known as a *bot herder* or *bot operator*. Botnets are the best-known mechanism to scale bots and their impact.

branch

A contained area of a code repository. The first branch created in a repository is the *main branch*. You can create a new branch from an existing branch, and you can then develop features or fix bugs in the new branch. A branch you create to build a feature is commonly referred to as a *feature branch*. When the feature is ready for release, you merge the feature branch back into the main branch. For more information, see About branches (GitHub documentation).

break-glass access

In exceptional circumstances and through an approved process, a quick means for a user to gain access to an AWS account that they don't typically have permissions to access. For more information, see the Implement break-glass procedures indicator in the AWS Well-Architected guidance.

brownfield strategy

The existing infrastructure in your environment. When adopting a brownfield strategy for a system architecture, you design the architecture around the constraints of the current systems and infrastructure. If you are expanding the existing infrastructure, you might blend brownfield and greenfield strategies.

buffer cache

The memory area where the most frequently accessed data is stored.

business capability

What a business does to generate value (for example, sales, customer service, or marketing). Microservices architectures and development decisions can be driven by business capabilities.

For more information, see the Organized around business capabilities section of the Running containerized microservices on AWS whitepaper.

business continuity planning (BCP)

A plan that addresses the potential impact of a disruptive event, such as a large-scale migration, on operations and enables a business to resume operations quickly.

C

CAF

See AWS Cloud Adoption Framework.

canary deployment

The slow and incremental release of a version to end users. When you are confident, you deploy the new version and replace the current version in its entirety.

CCoE

See Cloud Center of Excellence.

CDC

See change data capture.

change data capture (CDC)

The process of tracking changes to a data source, such as a database table, and recording metadata about the change. You can use CDC for various purposes, such as auditing or replicating changes in a target system to maintain synchronization.

chaos engineering

Intentionally introducing failures or disruptive events to test a system's resilience. You can use AWS Fault Injection Service (AWS FIS) to perform experiments that stress your AWS workloads and evaluate their response.

CI/CD

See continuous integration and continuous delivery.

classification

A categorization process that helps generate predictions. ML models for classification problems predict a discrete value. Discrete values are always distinct from one another. For example, a model might need to evaluate whether or not there is a car in an image.

Citizen Developer

A business user who creates AI applications using no-code/low-code platforms without specialized technical skills.

client-side encryption

Encryption of data locally, before the target AWS service receives it.

Cloud Center of Excellence (CCoE)

A multi-disciplinary team that drives cloud adoption efforts across an organization, including developing cloud best practices, mobilizing resources, establishing migration timelines, and leading the organization through large-scale transformations. For more information, see the CCoE posts on the AWS Cloud Enterprise Strategy Blog.

cloud computing

The cloud technology that is typically used for remote data storage and IoT device management. Cloud computing is commonly connected to edge computing technology.

cloud operating model

In an IT organization, the operating model that is used to build, mature, and optimize one or more cloud environments. For more information, see Building your Cloud Operating Model.

cloud stages of adoption

The four phases that organizations typically go through when they migrate to the AWS Cloud:

- Project – Running a few cloud-related projects for proof of concept and learning purposes
- Foundation – Making foundational investments to scale your cloud adoption (e.g., creating a landing zone, defining a CCoE, establishing an operations model)
- Migration – Migrating individual applications
- Re-invention – Optimizing products and services, and innovating in the cloud

These stages were defined by Stephen Orban in the blog post The Journey Toward Cloud-First & the Stages of Adoption on the AWS Cloud Enterprise Strategy blog. For information about how they relate to the AWS migration strategy, see the migration readiness guide.

CMDB

See configuration management database.

code repository

A location where source code and other assets, such as documentation, samples, and scripts, are stored and updated through version control processes. Common cloud repositories include GitHub or Bitbucket Cloud. Each version of the code is called a *branch*. In a microservice structure, each repository is devoted to a single piece of functionality. A single CI/CD pipeline can use multiple repositories.

cold cache

A buffer cache that is empty, not well populated, or contains stale or irrelevant data. This affects performance because the database instance must read from the main memory or disk, which is slower than reading from the buffer cache.

cold data

Data that is rarely accessed and is typically historical. When querying this kind of data, slow queries are typically acceptable. Moving this data to lower-performing and less expensive storage tiers or classes can reduce costs.

computer vision (CV)

A field of AI that uses machine learning to analyze and extract information from visual formats such as digital images and videos. For example, Amazon SageMaker AI provides image processing algorithms for CV.

configuration drift

For a workload, a configuration change from the expected state. It might cause the workload to become noncompliant, and it's typically gradual and unintentional.

configuration management database (CMDB)

A repository that stores and manages information about a database and its IT environment, including both hardware and software components and their configurations. You typically use data from a CMDB in the portfolio discovery and analysis stage of migration.

conformance pack

A collection of AWS Config rules and remediation actions that you can assemble to customize your compliance and security checks. You can deploy a conformance pack as a single entity in

an AWS account and Region, or across an organization, by using a YAML template. For more information, see Conformance packs in the AWS Config documentation.

continuous integration and continuous delivery (CI/CD)

The process of automating the source, build, test, staging, and production stages of the software release process. CI/CD is commonly described as a pipeline. CI/CD can help you automate processes, improve productivity, improve code quality, and deliver faster. For more information, see Benefits of continuous delivery. CD can also stand for *continuous deployment*. For more information, see Continuous Delivery vs. Continuous Deployment.

CV

See computer vision.

D

data at rest

Data that is stationary in your network, such as data that is in storage.

data classification

A process for identifying and categorizing the data in your network based on its criticality and sensitivity. It is a critical component of any cybersecurity risk management strategy because it helps you determine the appropriate protection and retention controls for the data. Data classification is a component of the security pillar in the AWS Well-Architected Framework. For more information, see Data classification.

data drift

A meaningful variation between the production data and the data that was used to train an ML model, or a meaningful change in the input data over time. Data drift can reduce the overall quality, accuracy, and fairness in ML model predictions.

data in transit

Data that is actively moving through your network, such as between network resources.

data mesh

An architectural framework that provides distributed, decentralized data ownership with centralized management and governance.

data minimization

The principle of collecting and processing only the data that is strictly necessary. Practicing data minimization in the AWS Cloud can reduce privacy risks, costs, and your analytics carbon footprint.

data perimeter

A set of preventive guardrails in your AWS environment that help make sure that only trusted identities are accessing trusted resources from expected networks. For more information, see Building a data perimeter on AWS.

data preprocessing

To transform raw data into a format that is easily parsed by your ML model. Preprocessing data can mean removing certain columns or rows and addressing missing, inconsistent, or duplicate values.

data provenance

The process of tracking the origin and history of data throughout its lifecycle, such as how the data was generated, transmitted, and stored.

data subject

An individual whose data is being collected and processed.

data warehouse

A data management system that supports business intelligence, such as analytics. Data warehouses commonly contain large amounts of historical data, and they are typically used for queries and analysis.

database definition language (DDL)

Statements or commands for creating or modifying the structure of tables and objects in a database.

database manipulation language (DML)

Statements or commands for modifying (inserting, updating, and deleting) information in a database.

DDL

See database definition language.

deep ensemble

To combine multiple deep learning models for prediction. You can use deep ensembles to obtain a more accurate prediction or for estimating uncertainty in predictions.

deep learning

An ML subfield that uses multiple layers of artificial neural networks to identify mapping between input data and target variables of interest.

defense-in-depth

An information security approach in which a series of security mechanisms and controls are thoughtfully layered throughout a computer network to protect the confidentiality, integrity, and availability of the network and the data within. When you adopt this strategy on AWS, you add multiple controls at different layers of the AWS Organizations structure to help secure resources. For example, a defense-in-depth approach might combine multi-factor authentication, network segmentation, and encryption.

delegated administrator

In AWS Organizations, a compatible service can register an AWS member account to administer the organization's accounts and manage permissions for that service. This account is called the *delegated administrator* for that service. For more information and a list of compatible services, see Services that work with AWS Organizations in the AWS Organizations documentation.

deployment

The process of making an application, new features, or code fixes available in the target environment. Deployment involves implementing changes in a code base and then building and running that code base in the application's environments.

development environment

See environment.

detective control

A security control that is designed to detect, log, and alert after an event has occurred. These controls are a second line of defense, alerting you to security events that bypassed the preventative controls in place. For more information, see Detective controls in *Implementing security controls on AWS*.

development value stream mapping (DVSM)

A process used to identify and prioritize constraints that adversely affect speed and quality in a software development lifecycle. DVSM extends the value stream mapping process originally designed for lean manufacturing practices. It focuses on the steps and teams required to create and move value through the software development process.

digital twin

A virtual representation of a real-world system, such as a building, factory, industrial equipment, or production line. Digital twins support predictive maintenance, remote monitoring, and production optimization.

dimension table

In a star schema, a smaller table that contains data attributes about quantitative data in a fact table. Dimension table attributes are typically text fields or discrete numbers that behave like text. These attributes are commonly used for query constraining, filtering, and result set labeling.

disaster

An event that prevents a workload or system from fulfilling its business objectives in its primary deployed location. These events can be natural disasters, technical failures, or the result of human actions, such as unintentional misconfiguration or a malware attack.

disaster recovery (DR)

The strategy and process you use to minimize downtime and data loss caused by a disaster. For more information, see Disaster Recovery of Workloads on AWS: Recovery in the Cloud in the AWS Well-Architected Framework.

DML

See database manipulation language.

domain-driven design

An approach to developing a complex software system by connecting its components to evolving domains, or core business goals, that each component serves. This concept was introduced by Eric Evans in his book, *Domain-Driven Design: Tackling Complexity in the Heart of Software* (Boston: Addison-Wesley Professional, 2003). For information about how you can use domain-driven design with the strangler fig pattern, see Modernizing legacy Microsoft ASP.NET (ASMX) web services incrementally by using containers and Amazon API Gateway.

DR

See disaster recovery.

drift detection

Tracking deviations from a baselined configuration. For example, you can use AWS CloudFormation to detect drift in system resources, or you can use AWS Control Tower to detect changes in your landing zone that might affect compliance with governance requirements.

DVSM

See development value stream mapping.

E

EDA

See exploratory data analysis.

EDI

See electronic data interchange.

edge computing

The technology that increases the computing power for smart devices at the edges of an IoT network. When compared with cloud computing, edge computing can reduce communication latency and improve response time.

electronic data interchange (EDI)

The automated exchange of business documents between organizations. For more information, see What is Electronic Data Interchange.

encryption

A computing process that transforms plaintext data, which is human-readable, into ciphertext.

encryption key

A cryptographic string of randomized bits that is generated by an encryption algorithm. Keys can vary in length, and each key is designed to be unpredictable and unique.

endianness

The order in which bytes are stored in computer memory. Big-endian systems store the most significant byte first. Little-endian systems store the least significant byte first.

endpoint

See service endpoint.

endpoint service

A service that you can host in a virtual private cloud (VPC) to share with other users. You can create an endpoint service with AWS PrivateLink and grant permissions to other AWS accounts or to AWS Identity and Access Management (IAM) principals. These accounts or principals can connect to your endpoint service privately by creating interface VPC endpoints. For more information, see Create an endpoint service in the Amazon Virtual Private Cloud (Amazon VPC) documentation.

enterprise resource planning (ERP)

A system that automates and manages key business processes (such as accounting, MES, and project management) for an enterprise.

envelope encryption

The process of encrypting an encryption key with another encryption key. For more information, see Envelope encryption in the AWS Key Management Service (AWS KMS) documentation.

environment

An instance of a running application. The following are common types of environments in cloud computing:

- development environment – An instance of a running application that is available only to the core team responsible for maintaining the application. Development environments are used to test changes before promoting them to upper environments. This type of environment is sometimes referred to as a *test environment*.
- lower environments – All development environments for an application, such as those used for initial builds and tests.
- production environment – An instance of a running application that end users can access. In a CI/CD pipeline, the production environment is the last deployment environment.

- upper environments – All environments that can be accessed by users other than the core development team. This can include a production environment, preproduction environments, and environments for user acceptance testing.

epic

In agile methodologies, functional categories that help organize and prioritize your work. Epics provide a high-level description of requirements and implementation tasks. For example, AWS CAF security epics include identity and access management, detective controls, infrastructure security, data protection, and incident response. For more information about epics in the AWS migration strategy, see the program implementation guide.

ERP

See enterprise resource planning.

exploratory data analysis (EDA)

The process of analyzing a dataset to understand its main characteristics. You collect or aggregate data and then perform initial investigations to find patterns, detect anomalies, and check assumptions. EDA is performed by calculating summary statistics and creating data visualizations.

F

fact table

The central table in a star schema. It stores quantitative data about business operations. Typically, a fact table contains two types of columns: those that contain measures and those that contain a foreign key to a dimension table.

fail fast

A philosophy that uses frequent and incremental testing to reduce the development lifecycle. It is a critical part of an agile approach.

fault isolation boundary

In the AWS Cloud, a boundary such as an Availability Zone, AWS Region, control plane, or data plane that limits the effect of a failure and helps improve the resilience of workloads. For more information, see AWS Fault Isolation Boundaries.

feature branch

See branch.

features

The input data that you use to make a prediction. For example, in a manufacturing context, features could be images that are periodically captured from the manufacturing line.

feature importance

How significant a feature is for a model's predictions. This is usually expressed as a numerical score that can be calculated through various techniques, such as Shapley Additive Explanations (SHAP) and integrated gradients. For more information, see Machine learning model interpretability with AWS.

feature transformation

To optimize data for the ML process, including enriching data with additional sources, scaling values, or extracting multiple sets of information from a single data field. This enables the ML model to benefit from the data. For example, if you break down the "2021-05-27 00:15:37" date into "2021", "May", "Thu", and "15", you can help the learning algorithm learn nuanced patterns associated with different data components.

few-shot prompting

Providing an LLM with a small number of examples that demonstrate the task and desired output before asking it to perform a similar task. This technique is an application of in-context learning, where models learn from examples (*shots*) that are embedded in prompts. Few-shot prompting can be effective for tasks that require specific formatting, reasoning, or domain knowledge. See also zero-shot prompting.

FGAC

See fine-grained access control.

fine-grained access control (FGAC)

The use of multiple conditions to allow or deny an access request.

flash-cut migration

A database migration method that uses continuous data replication through change data capture to migrate data in the shortest time possible, instead of using a phased approach. The objective is to keep downtime to a minimum.

FM

See foundation model.

foundation model (FM)

A large deep-learning neural network that has been training on massive datasets of generalized and unlabeled data. FMs are capable of performing a wide variety of general tasks, such as understanding language, generating text and images, and conversing in natural language. For more information, see What are Foundation Models.

FM gateway

A centralized intermediary that controls and normalizes access to foundation models. Also known as an *LLM gateway*.

G

generative AI

A subset of AI models that have been trained on large amounts of data and that can use a simple text prompt to create new content and artifacts, such as images, videos, text, and audio. For more information, see What is Generative AI.

geo blocking

See geographic restrictions.

geographic restrictions (geo blocking)

In Amazon CloudFront, an option to prevent users in specific countries from accessing content distributions. You can use an allow list or block list to specify approved and banned countries. For more information, see Restricting the geographic distribution of your content in the CloudFront documentation.

Gitflow workflow

An approach in which lower and upper environments use different branches in a source code repository. The Gitflow workflow is considered legacy, and the trunk-based workflow is the modern, preferred approach.

golden image

A snapshot of a system or software that is used as a template to deploy new instances of that system or software. For example, in manufacturing, a golden image can be used to provision

software on multiple devices and helps improve speed, scalability, and productivity in device manufacturing operations.

greenfield strategy

The absence of existing infrastructure in a new environment. When adopting a greenfield strategy for a system architecture, you can select all new technologies without the restriction of compatibility with existing infrastructure, also known as brownfield. If you are expanding the existing infrastructure, you might blend brownfield and greenfield strategies.

guardrail

A high-level rule that helps govern resources, policies, and compliance across organizational units (OUs). *Preventive guardrails* enforce policies to ensure alignment to compliance standards. They are implemented by using service control policies and IAM permissions boundaries. *Detective guardrails* detect policy violations and compliance issues, and generate alerts for remediation. They are implemented by using AWS Config, AWS Security Hub CSPM, Amazon GuardDuty, AWS Trusted Advisor, Amazon Inspector, and custom AWS Lambda checks.

guardrails (AI)

Safety mechanisms that filter, validate, and constrain agent inputs and outputs to help ensure responsible and safe AI behavior.

H

HA

See high availability.

heterogeneous database migration

Migrating your source database to a target database that uses a different database engine (for example, Oracle to Amazon Aurora). Heterogeneous migration is typically part of a re-architecting effort, and converting the schema can be a complex task. AWS provides AWS SCT that helps with schema conversions.

high availability (HA)

The ability of a workload to operate continuously, without intervention, in the event of challenges or disasters. HA systems are designed to automatically fail over, consistently deliver

high-quality performance, and handle different loads and failures with minimal performance impact.

historian modernization

An approach used to modernize and upgrade operational technology (OT) systems to better serve the needs of the manufacturing industry. A *historian* is a type of database that is used to collect and store data from various sources in a factory.

holdout data

A portion of historical, labeled data that is withheld from a dataset that is used to train a machine learning model. You can use holdout data to evaluate the model performance by comparing the model predictions against the holdout data.

human-in-the-loop (HitL)

A workflow pattern where agent execution pauses for human review and approval at critical decision points.

homogeneous database migration

Migrating your source database to a target database that shares the same database engine (for example, Microsoft SQL Server to Amazon RDS for SQL Server). Homogeneous migration is typically part of a rehosting or replatforming effort. You can use native database utilities to migrate the schema.

hot data

Data that is frequently accessed, such as real-time data or recent translational data. This data typically requires a high-performance storage tier or class to provide fast query responses.

hotfix

An urgent fix for a critical issue in a production environment. Due to its urgency, a hotfix is usually made outside of the typical DevOps release workflow.

hypercare period

Immediately following cutover, the period of time when a migration team manages and monitors the migrated applications in the cloud in order to address any issues. Typically, this period is 1–4 days in length. At the end of the hypercare period, the migration team typically transfers responsibility for the applications to the cloud operations team.

I

IaC

See infrastructure as code.

identity-based policy

A policy attached to one or more IAM principals that defines their permissions within the AWS Cloud environment.

idle application

An application that has an average CPU and memory usage between 5 and 20 percent over a period of 90 days. In a migration project, it is common to retire these applications or retain them on premises.

IIoT

See industrial Internet of Things.

immutable infrastructure

A model that deploys new infrastructure for production workloads instead of updating, patching, or modifying the existing infrastructure. Immutable infrastructures are inherently more consistent, reliable, and predictable than mutable infrastructure. For more information, see the Deploy using immutable infrastructure best practice in the AWS Well-Architected Framework.

inbound (ingress) VPC

In an AWS multi-account architecture, a VPC that accepts, inspects, and routes network connections from outside an application. The AWS Security Reference Architecture recommends setting up your Network account with inbound, outbound, and inspection VPCs to protect the two-way interface between your application and the broader internet.

incremental migration

A cutover strategy in which you migrate your application in small parts instead of performing a single, full cutover. For example, you might move only a few microservices or users to the new system initially. After you verify that everything is working properly, you can incrementally move additional microservices or users until you can decommission your legacy system. This strategy reduces the risks associated with large migrations.

Industry 4.0

A term that was introduced by Klaus Schwab in 2016 to refer to the modernization of manufacturing processes through advances in connectivity, real-time data, automation, analytics, and AI/ML.

infrastructure

All of the resources and assets contained within an application's environment.

infrastructure as code (IaC)

The process of provisioning and managing an application's infrastructure through a set of configuration files. IaC is designed to help you centralize infrastructure management, standardize resources, and scale quickly so that new environments are repeatable, reliable, and consistent.

industrial Internet of Things (IIoT)

The use of internet-connected sensors and devices in the industrial sectors, such as manufacturing, energy, automotive, healthcare, life sciences, and agriculture. For more information, see Building an industrial Internet of Things (IIoT) digital transformation strategy.

inspection VPC

In an AWS multi-account architecture, a centralized VPC that manages inspections of network traffic between VPCs (in the same or different AWS Regions), the internet, and on-premises networks. The AWS Security Reference Architecture recommends setting up your Network account with inbound, outbound, and inspection VPCs to protect the two-way interface between your application and the broader internet.

Internet of Things (IoT)

The network of connected physical objects with embedded sensors or processors that communicate with other devices and systems through the internet or over a local communication network. For more information, see What is IoT?

interpretability

A characteristic of a machine learning model that describes the degree to which a human can understand how the model's predictions depend on its inputs. For more information, see Machine learning model interpretability with AWS.

IoT

See Internet of Things.

IT information library (ITIL)

A set of best practices for delivering IT services and aligning these services with business requirements. ITIL provides the foundation for ITSM.

IT service management (ITSM)

Activities associated with designing, implementing, managing, and supporting IT services for an organization. For information about integrating cloud operations with ITSM tools, see the operations integration guide.

ITIL

See IT information library.

ITSM

See IT service management.

L

label-based access control (LBAC)

An implementation of mandatory access control (MAC) where the users and the data itself are each explicitly assigned a security label value. The intersection between the user security label and data security label determines which rows and columns can be seen by the user.

landing zone

A landing zone is a well-architected, multi-account AWS environment that is scalable and secure. This is a starting point from which your organizations can quickly launch and deploy workloads and applications with confidence in their security and infrastructure environment. For more information about landing zones, see Setting up a secure and scalable multi-account AWS environment.

large language model (LLM)

A deep learning AI model that is pretrained on a vast amount of data. An LLM can perform multiple tasks, such as answering questions, summarizing documents, translating text into other languages, and completing sentences. For more information, see What are LLMs.

large migration

A migration of 300 or more servers.

LBAC

See label-based access control.

least privilege

The security best practice of granting the minimum permissions required to perform a task. For more information, see Apply least-privilege permissions in the IAM documentation.

lift and shift

See 7 Rs.

little-endian system

A system that stores the least significant byte first. See also endianness.

LLM

See large language model.

lower environments

See environment.

M

machine learning (ML)

A type of artificial intelligence that uses algorithms and techniques for pattern recognition and learning. ML analyzes and learns from recorded data, such as Internet of Things (IoT) data, to generate a statistical model based on patterns. For more information, see Machine Learning.

main branch

See branch.

malware

Software that is designed to compromise computer security or privacy. Malware might disrupt computer systems, leak sensitive information, or gain unauthorized access. Examples of malware include viruses, worms, ransomware, Trojan horses, spyware, and keyloggers.

managed services

AWS services for which AWS operates the infrastructure layer, the operating system, and platforms, and you access the endpoints to store and retrieve data. Amazon Simple Storage

Service (Amazon S3) and Amazon DynamoDB are examples of managed services. These are also known as *abstracted services*.

manufacturing execution system (MES)

A software system for tracking, monitoring, documenting, and controlling production processes that convert raw materials to finished products on the shop floor.

MAP

See Migration Acceleration Program.

MCP

See Model Context Protocol.

Model Context Protocol (MCP)

A stateless protocol for agent-to-tool communication.

MCP server

A service that exposes one or more tools through the Model Context Protocol.

mechanism

A complete process in which you create a tool, drive adoption of the tool, and then inspect the results in order to make adjustments. A mechanism is a cycle that reinforces and improves itself as it operates. For more information, see Building mechanisms in the AWS Well-Architected Framework.

member account

All AWS accounts other than the management account that are part of an organization in AWS Organizations. An account can be a member of only one organization at a time.

MES

See manufacturing execution system.

Message Queuing Telemetry Transport (MQTT)

A lightweight, machine-to-machine (M2M) communication protocol, based on the publish/subscribe pattern, for resource-constrained IoT devices.

microservice

A small, independent service that communicates over well-defined APIs and is typically owned by small, self-contained teams. For example, an insurance system might include

microservices that map to business capabilities, such as sales or marketing, or subdomains, such as purchasing, claims, or analytics. The benefits of microservices include agility, flexible scaling, easy deployment, reusable code, and resilience. For more information, see Integrating microservices by using AWS serverless services.

microservices architecture

An approach to building an application with independent components that run each application process as a microservice. These microservices communicate through a well-defined interface by using lightweight APIs. Each microservice in this architecture can be updated, deployed, and scaled to meet demand for specific functions of an application. For more information, see Implementing microservices on AWS.

Migration Acceleration Program (MAP)

An AWS program that provides consulting support, training, and services to help organizations build a strong operational foundation for moving to the cloud, and to help offset the initial cost of migrations. MAP includes a migration methodology for executing legacy migrations in a methodical way and a set of tools to automate and accelerate common migration scenarios.

migration at scale

The process of moving the majority of the application portfolio to the cloud in waves, with more applications moved at a faster rate in each wave. This phase uses the best practices and lessons learned from the earlier phases to implement a *migration factory* of teams, tools, and processes to streamline the migration of workloads through automation and agile delivery. This is the third phase of the AWS migration strategy.

migration factory

Cross-functional teams that streamline the migration of workloads through automated, agile approaches. Migration factory teams typically include operations, business analysts and owners, migration engineers, developers, and DevOps professionals working in sprints. Between 20 and 50 percent of an enterprise application portfolio consists of repeated patterns that can be optimized by a factory approach. For more information, see the discussion of migration factories and the Cloud Migration Factory guide in this content set.

migration metadata

The information about the application and server that is needed to complete the migration. Each migration pattern requires a different set of migration metadata. Examples of migration metadata include the target subnet, security group, and AWS account.

migration pattern

A repeatable migration task that details the migration strategy, the migration destination, and the migration application or service used. Example: Rehost migration to Amazon EC2 with AWS Application Migration Service.

Migration Portfolio Assessment (MPA)

An online tool that provides information for validating the business case for migrating to the AWS Cloud. MPA provides detailed portfolio assessment (server right-sizing, pricing, TCO comparisons, migration cost analysis) as well as migration planning (application data analysis and data collection, application grouping, migration prioritization, and wave planning). The MPA tool (requires login) is available free of charge to all AWS consultants and APN Partner consultants.

Migration Readiness Assessment (MRA)

The process of gaining insights about an organization's cloud readiness status, identifying strengths and weaknesses, and building an action plan to close identified gaps, using the AWS CAF. For more information, see the migration readiness guide. MRA is the first phase of the AWS migration strategy.

migration strategy

The approach used to migrate a workload to the AWS Cloud. For more information, see the 7 Rs entry in this glossary and see Mobilize your organization to accelerate large-scale migrations.

ML

See machine learning.

modernization

Transforming an outdated (legacy or monolithic) application and its infrastructure into an agile, elastic, and highly available system in the cloud to reduce costs, gain efficiencies, and take advantage of innovations. For more information, see Strategy for modernizing applications in the AWS Cloud.

modernization readiness assessment

An evaluation that helps determine the modernization readiness of an organization's applications; identifies benefits, risks, and dependencies; and determines how well the organization can support the future state of those applications. The outcome of the assessment is a blueprint of the target architecture, a roadmap that details development phases and

milestones for the modernization process, and an action plan for addressing identified gaps. For more information, see Evaluating modernization readiness for applications in the AWS Cloud.

monolithic applications (monoliths)

Applications that run as a single service with tightly coupled processes. Monolithic applications have several drawbacks. If one application feature experiences a spike in demand, the entire architecture must be scaled. Adding or improving a monolithic application's features also becomes more complex when the code base grows. To address these issues, you can use a microservices architecture. For more information, see Decomposing monoliths into microservices.

MPA

See Migration Portfolio Assessment.

MQTT

See Message Queuing Telemetry Transport.

multiclass classification

A process that helps generate predictions for multiple classes (predicting one of more than two outcomes). For example, an ML model might ask "Is this product a book, car, or phone?" or "Which product category is most interesting to this customer?"

mutable infrastructure

A model that updates and modifies the existing infrastructure for production workloads. For improved consistency, reliability, and predictability, the AWS Well-Architected Framework recommends the use of immutable infrastructure as a best practice.

O

OAC

See origin access control.

OAI

See origin access identity.

OCM

See organizational change management.

offline migration

A migration method in which the source workload is taken down during the migration process. This method involves extended downtime and is typically used for small, non-critical workloads.

OI

See operations integration.

OLA

See operational-level agreement.

online migration

A migration method in which the source workload is copied to the target system without being taken offline. Applications that are connected to the workload can continue to function during the migration. This method involves zero to minimal downtime and is typically used for critical production workloads.

OPC-UA

See Open Process Communications - Unified Architecture.

Open Process Communications - Unified Architecture (OPC-UA)

A machine-to-machine (M2M) communication protocol for industrial automation. OPC-UA provides an interoperability standard with data encryption, authentication, and authorization schemes.

operational-level agreement (OLA)

An agreement that clarifies what functional IT groups promise to deliver to each other, to support a service-level agreement (SLA).

operational readiness review (ORR)

A checklist of questions and associated best practices that help you understand, evaluate, prevent, or reduce the scope of incidents and possible failures. For more information, see Operational Readiness Reviews (ORR) in the AWS Well-Architected Framework.

operational technology (OT)

Hardware and software systems that work with the physical environment to control industrial operations, equipment, and infrastructure. In manufacturing, the integration of OT and information technology (IT) systems is a key focus for Industry 4.0 transformations.

operations integration (OI)

The process of modernizing operations in the cloud, which involves readiness planning, automation, and integration. For more information, see the operations integration guide.

organization trail

A trail that's created by AWS CloudTrail that logs all events for all AWS accounts in an organization in AWS Organizations. This trail is created in each AWS account that's part of the organization and tracks the activity in each account. For more information, see Creating a trail for an organization in the CloudTrail documentation.

organizational change management (OCM)

A framework for managing major, disruptive business transformations from a people, culture, and leadership perspective. OCM helps organizations prepare for, and transition to, new systems and strategies by accelerating change adoption, addressing transitional issues, and driving cultural and organizational changes. In the AWS migration strategy, this framework is called *people acceleration*, because of the speed of change required in cloud adoption projects. For more information, see the OCM guide.

origin access control (OAC)

In CloudFront, an enhanced option for restricting access to secure your Amazon Simple Storage Service (Amazon S3) content. OAC supports all S3 buckets in all AWS Regions, server-side encryption with AWS KMS (SSE-KMS), and dynamic `PUT` and `DELETE` requests to the S3 bucket.

origin access identity (OAI)

In CloudFront, an option for restricting access to secure your Amazon S3 content. When you use OAI, CloudFront creates a principal that Amazon S3 can authenticate with. Authenticated principals can access content in an S3 bucket only through a specific CloudFront distribution. See also OAC, which provides more granular and enhanced access control.

ORR

See operational readiness review.

OT

See operational technology.

outbound (egress) VPC

In an AWS multi-account architecture, a VPC that handles network connections that are initiated from within an application. The AWS Security Reference Architecture recommends

setting up your Network account with inbound, outbound, and inspection VPCs to protect the two-way interface between your application and the broader internet.

P

permissions boundary

An IAM management policy that is attached to IAM principals to set the maximum permissions that the user or role can have. For more information, see Permissions boundaries in the IAM documentation.

personally identifiable information (PII)

Information that, when viewed directly or paired with other related data, can be used to reasonably infer the identity of an individual. Examples of PII include names, addresses, and contact information.

PII

See personally identifiable information.

playbook

A set of predefined steps that capture the work associated with migrations, such as delivering core operations functions in the cloud. A playbook can take the form of scripts, automated runbooks, or a summary of processes or steps required to operate your modernized environment.

PLC

See programmable logic controller.

PLM

See product lifecycle management.

policy

An object that can define permissions (see identity-based policy), specify access conditions (see resource-based policy), or define the maximum permissions for all accounts in an organization in AWS Organizations (see service control policy).

polyglot persistence

Independently choosing a microservice's data storage technology based on data access patterns and other requirements. If your microservices have the same data storage technology, they can encounter implementation challenges or experience poor performance. Microservices are more easily implemented and achieve better performance and scalability if they use the data store best adapted to their requirements.

portfolio assessment

A process of discovering, analyzing, and prioritizing the application portfolio in order to plan the migration. For more information, see Evaluating migration readiness.

predicate

A query condition that returns `true` or `false`, commonly located in a `WHERE` clause.

predicate pushdown

A database query optimization technique that filters the data in the query before transfer. This reduces the amount of data that must be retrieved and processed from the relational database, and it improves query performance.

preventative control

A security control that is designed to prevent an event from occurring. These controls are a first line of defense to help prevent unauthorized access or unwanted changes to your network. For more information, see Preventative controls in *Implementing security controls on AWS*.

principal

An entity in AWS that can perform actions and access resources. This entity is typically a root user for an AWS account, an IAM role, or a user. For more information, see *Principal* in Roles terms and concepts in the IAM documentation.

privacy by design

A system engineering approach that takes privacy into account through the whole development process.

private hosted zones

A container that holds information about how you want Amazon Route 53 to respond to DNS queries for a domain and its subdomains within one or more VPCs. For more information, see Working with private hosted zones in the Route 53 documentation.

proactive control

A security control designed to prevent the deployment of noncompliant resources. These controls scan resources before they are provisioned. If the resource is not compliant with the control, then it isn't provisioned. For more information, see the Controls reference guide in the AWS Control Tower documentation and see Proactive controls in *Implementing security controls on AWS*.

product lifecycle management (PLM)

The management of data and processes for a product throughout its entire lifecycle, from design, development, and launch, through growth and maturity, to decline and removal.

production environment

See environment.

programmable logic controller (PLC)

In manufacturing, a highly reliable, adaptable computer that monitors machines and automates manufacturing processes.

prompt chaining

Using the output of one LLM prompt as the input for the next prompt to generate better responses. This technique is used to break down a complex task into subtasks, or to iteratively refine or expand a preliminary response. It helps improve the accuracy and relevance of a model's responses and allows for more granular, personalized results.

pseudonymization

The process of replacing personal identifiers in a dataset with placeholder values. Pseudonymization can help protect personal privacy. Pseudonymized data is still considered to be personal data.

publish/subscribe (pub/sub)

A pattern that enables asynchronous communications among microservices to improve scalability and responsiveness. For example, in a microservices-based MES, a microservice can publish event messages to a channel that other microservices can subscribe to. The system can add new microservices without changing the publishing service.

Q

query plan

A series of steps, like instructions, that are used to access the data in a SQL relational database system.

query plan regression

When a database service optimizer chooses a less optimal plan than it did before a given change to the database environment. This can be caused by changes to statistics, constraints, environment settings, query parameter bindings, and updates to the database engine.

R

RACI matrix

See responsible, accountable, consulted, informed (RACI).

RAG

See Retrieval Augmented Generation.

ransomware

A malicious software that is designed to block access to a computer system or data until a payment is made.

RASCI matrix

See responsible, accountable, consulted, informed (RACI).

RCAC

See row and column access control.

read replica

A copy of a database that's used for read-only purposes. You can route queries to the read replica to reduce the load on your primary database.

re-architect

See 7 Rs.

recovery point objective (RPO)

The maximum acceptable amount of time since the last data recovery point. This determines what is considered an acceptable loss of data between the last recovery point and the interruption of service.

recovery time objective (RTO)

The maximum acceptable delay between the interruption of service and restoration of service.

refactor

See 7 Rs.

Region

A collection of AWS resources in a geographic area. Each AWS Region is isolated and independent of the others to provide fault tolerance, stability, and resilience. For more information, see Specify which AWS Regions your account can use.

regression

An ML technique that predicts a numeric value. For example, to solve the problem of "What price will this house sell for?" an ML model could use a linear regression model to predict a house's sale price based on known facts about the house (for example, the square footage).

rehost

See 7 Rs.

release

In a deployment process, the act of promoting changes to a production environment.

relocate

See 7 Rs.

replatform

See 7 Rs.

repurchase

See 7 Rs.

resiliency

An application's ability to resist or recover from disruptions. High availability and disaster recovery are common considerations when planning for resiliency in the AWS Cloud. For more information, see AWS Cloud Resilience.

resource-based policy

A policy attached to a resource, such as an Amazon S3 bucket, an endpoint, or an encryption key. This type of policy specifies which principals are allowed access, supported actions, and any other conditions that must be met.

responsible, accountable, consulted, informed (RACI) matrix

A matrix that defines the roles and responsibilities for all parties involved in migration activities and cloud operations. The matrix name is derived from the responsibility types defined in the matrix: responsible (R), accountable (A), consulted (C), and informed (I). The support (S) type is optional. If you include support, the matrix is called a *RASCI matrix*, and if you exclude it, it's called a *RACI matrix*.

responsive control

A security control that is designed to drive remediation of adverse events or deviations from your security baseline. For more information, see Responsive controls in *Implementing security controls on AWS*.

retain

See 7 Rs.

retire

See 7 Rs.

Retrieval Augmented Generation (RAG)

A generative AI technology in which an LLM references an authoritative data source that is outside of its training data sources before generating a response. For example, a RAG model might perform a semantic search of an organization's knowledge base or custom data. For more information, see What is RAG.

rotation

The process of periodically updating a secret to make it more difficult for an attacker to access the credentials.

row and column access control (RCAC)

The use of basic, flexible SQL expressions that have defined access rules. RCAC consists of row permissions and column masks.

RPO

See recovery point objective.

RTO

See recovery time objective.

runbook

A set of manual or automated procedures required to perform a specific task. These are typically built to streamline repetitive operations or procedures with high error rates.

S

SAML 2.0

An open standard that many identity providers (IdPs) use. This feature enables federated single sign-on (SSO), so users can log into the AWS Management Console or call the AWS API operations without you having to create user in IAM for everyone in your organization. For more information about SAML 2.0-based federation, see About SAML 2.0-based federation in the IAM documentation.

SCADA

See supervisory control and data acquisition.

SCP

See service control policy.

secret

In AWS Secrets Manager, confidential or restricted information, such as a password or user credentials, that you store in encrypted form. It consists of the secret value and its metadata. The secret value can be binary, a single string, or multiple strings. For more information, see What's in a Secrets Manager secret? in the Secrets Manager documentation.

security by design

A system engineering approach that takes security into account through the whole development process.

security control

A technical or administrative guardrail that prevents, detects, or reduces the ability of a threat actor to exploit a security vulnerability. There are four primary types of security controls: preventative, detective, responsive, and proactive.

security hardening

The process of reducing the attack surface to make it more resistant to attacks. This can include actions such as removing resources that are no longer needed, implementing the security best practice of granting least privilege, or deactivating unnecessary features in configuration files.

security information and event management (SIEM) system

Tools and services that combine security information management (SIM) and security event management (SEM) systems. A SIEM system collects, monitors, and analyzes data from servers, networks, devices, and other sources to detect threats and security breaches, and to generate alerts.

security response automation

A predefined and programmed action that is designed to automatically respond to or remediate a security event. These automations serve as detective or responsive security controls that help you implement AWS security best practices. Examples of automated response actions include modifying a VPC security group, patching an Amazon EC2 instance, or rotating credentials.

server-side encryption

Encryption of data at its destination, by the AWS service that receives it.

service control policy (SCP)

A policy that provides centralized control over permissions for all accounts in an organization in AWS Organizations. SCPs define guardrails or set limits on actions that an administrator can delegate to users or roles. You can use SCPs as allow lists or deny lists, to specify which services or actions are permitted or prohibited. For more information, see Service control policies in the AWS Organizations documentation.

service endpoint

The URL of the entry point for an AWS service. You can use the endpoint to connect programmatically to the target service. For more information, see AWS service endpoints in *AWS General Reference*.

service-level agreement (SLA)

An agreement that clarifies what an IT team promises to deliver to their customers, such as service uptime and performance.

service-level indicator (SLI)

A measurement of a performance aspect of a service, such as its error rate, availability, or throughput.

service-level objective (SLO)

A target metric that represents the health of a service, as measured by a service-level indicator.

shared responsibility model

A model describing the responsibility you share with AWS for cloud security and compliance. AWS is responsible for security *of* the cloud, whereas you are responsible for security *in* the cloud. For more information, see Shared responsibility model.

Shadow AI

Unauthorized AI applications built or used outside of governed channels within an organization.

SIEM

See security information and event management system.

single point of failure (SPOF)

A failure in a single, critical component of an application that can disrupt the system.

SLA

See service-level agreement.

SLI

See service-level indicator.

SLO

See service-level objective.

split-and-seed model

A pattern for scaling and accelerating modernization projects. As new features and product releases are defined, the core team splits up to create new product teams. This helps scale your organization's capabilities and services, improves developer productivity, and supports rapid innovation. For more information, see Phased approach to modernizing applications in the AWS Cloud.

SPOF

See single point of failure.

star schema

A database organizational structure that uses one large fact table to store transactional or measured data and uses one or more smaller dimensional tables to store data attributes. This structure is designed for use in a data warehouse or for business intelligence purposes.

strangler fig pattern

An approach to modernizing monolithic systems by incrementally rewriting and replacing system functionality until the legacy system can be decommissioned. This pattern uses the analogy of a fig vine that grows into an established tree and eventually overcomes and replaces its host. The pattern was introduced by Martin Fowler as a way to manage risk when rewriting monolithic systems. For an example of how to apply this pattern, see Modernizing legacy Microsoft ASP.NET (ASMX) web services incrementally by using containers and Amazon API Gateway.

subnet

A range of IP addresses in your VPC. A subnet must reside in a single Availability Zone.

supervisory control and data acquisition (SCADA)

In manufacturing, a system that uses hardware and software to monitor physical assets and production operations.

symmetric encryption

An encryption algorithm that uses the same key to encrypt and decrypt the data.

synthetic testing

Testing a system in a way that simulates user interactions to detect potential issues or to monitor performance. You can use Amazon CloudWatch Synthetics to create these tests.

system prompt

A technique for providing context, instructions, or guidelines to an LLM to direct its behavior. System prompts help set context and establish rules for interactions with users.

T

tags

Key-value pairs that act as metadata for organizing your AWS resources. Tags can help you manage, identify, organize, search for, and filter resources. For more information, see Tagging your AWS resources.

target variable

The value that you are trying to predict in supervised ML. This is also referred to as an *outcome variable*. For example, in a manufacturing setting the target variable could be a product defect.

task list

A tool that is used to track progress through a runbook. A task list contains an overview of the runbook and a list of general tasks to be completed. For each general task, it includes the estimated amount of time required, the owner, and the progress.

test environment

See environment.

training

To provide data for your ML model to learn from. The training data must contain the correct answer. The learning algorithm finds patterns in the training data that map the input data attributes to the target (the answer that you want to predict). It outputs an ML model that captures these patterns. You can then use the ML model to make predictions on new data for which you don't know the target.

tool

A function or API that an agent can invoke to perform operations in external systems.

transit gateway

A network transit hub that you can use to interconnect your VPCs and on-premises networks. For more information, see What is a transit gateway in the AWS Transit Gateway documentation.

trunk-based workflow

An approach in which developers build and test features locally in a feature branch and then merge those changes into the main branch. The main branch is then built to the development, preproduction, and production environments, sequentially.

trusted access

Granting permissions to a service that you specify to perform tasks in your organization in AWS Organizations and in its accounts on your behalf. The trusted service creates a service-linked role in each account, when that role is needed, to perform management tasks for you. For more information, see Using AWS Organizations with other AWS services in the AWS Organizations documentation.

tuning

To change aspects of your training process to improve the ML model's accuracy. For example, you can train the ML model by generating a labeling set, adding labels, and then repeating these steps several times under different settings to optimize the model.

two-pizza team

A small DevOps team that you can feed with two pizzas. A two-pizza team size ensures the best possible opportunity for collaboration in software development.

U

uncertainty

A concept that refers to imprecise, incomplete, or unknown information that can undermine the reliability of predictive ML models. There are two types of uncertainty: *Epistemic uncertainty* is caused by limited, incomplete data, whereas *aleatoric uncertainty* is caused by the noise and randomness inherent in the data.

undifferentiated tasks

Also known as *heavy lifting*, work that is necessary to create and operate an application but that doesn't provide direct value to the end user or provide competitive advantage. Examples of undifferentiated tasks include procurement, maintenance, and capacity planning.

upper environments

See environment.

V

vacuuming

A database maintenance operation that involves cleaning up after incremental updates to reclaim storage and improve performance.

version control

Processes and tools that track changes, such as changes to source code in a repository.

VPC peering

A connection between two VPCs that allows you to route traffic by using private IP addresses. For more information, see What is VPC peering in the Amazon VPC documentation.

vulnerability

A software or hardware flaw that compromises the security of the system.

W

warm cache

A buffer cache that contains current, relevant data that is frequently accessed. The database instance can read from the buffer cache, which is faster than reading from the main memory or disk.

warm data

Data that is infrequently accessed. When querying this kind of data, moderately slow queries are typically acceptable.

window function

A SQL function that performs a calculation on a group of rows that relate in some way to the current record. Window functions are useful for processing tasks, such as calculating a moving average or accessing the value of rows based on the relative position of the current row.

workload

A collection of resources and code that delivers business value, such as a customer-facing application or backend process.

workstream

Functional groups in a migration project that are responsible for a specific set of tasks. Each workstream is independent but supports the other workstreams in the project. For example, the portfolio workstream is responsible for prioritizing applications, wave planning, and collecting migration metadata. The portfolio workstream delivers these assets to the migration workstream, which then migrates the servers and applications.

WORM

See write once, read many.

WQF

See AWS Workload Qualification Framework.

write once, read many (WORM)

A storage model that writes data a single time and prevents the data from being deleted or modified. Authorized users can read the data as many times as needed, but they cannot change it. This data storage infrastructure is considered immutable.

Z

zero-day exploit

An attack, typically malware, that takes advantage of a zero-day vulnerability.

zero-day vulnerability

An unmitigated flaw or vulnerability in a production system. Threat actors can use this type of vulnerability to attack the system. Developers frequently become aware of the vulnerability as a result of the attack.

zero-shot prompting

Providing an LLM with instructions for performing a task but no examples (*shots*) that can help guide it. The LLM must use its pre-trained knowledge to handle the task. The effectiveness of zero-shot prompting depends on the complexity of the task and the quality of the prompt. See also few-shot prompting.

zombie application

An application that has an average CPU and memory usage below 5 percent. In a migration project, it is common to retire these applications.