Operationalizing agentic AI on AWS

AWS Prescriptive Guidance

AWS Prescriptive Guidance: Operationalizing agentic AI on AWS

Table of Contents

Operationalizing agentic AI on AWS

Aaron Sempf, Brad Ryan, Bhargs Srivathsan, and Akhil Bhaskar, Amazon Web Services

August 2025 (document history)

Agentic AI is not a feature—it's a new operational paradigm. Organizations that invest in disciplined architecture, trust frameworks, and business-aligned deployment models will lead the next generation of adaptive, intelligent enterprises.

Agentic AI represents the convergence of autonomous software agents and generative AI. It fuses the decision-making and goal-directed behavior of agents with the language understanding and generation capabilities of large language models (LLMs). These agents can reason, act, adapt, and collaborate across dynamic enterprise environments. To operationalize this potential, enterprises must shift their mindset from model deployment to agent infrastructure.

This guide provides an organizational strategy to transform agentic AI from isolated experiments into enterprise-scale, value-generating infrastructure. It can help you embed intelligent agents across workflows with governance, scalability, and business alignment.

Key focus areas and recommendations

This guide focuses on the following foundational areas when operationalizing agentic AI. Organizational and business recommendations are provided for each focus area:

- Focus area 1: Clarify agent intent and scope – Align agents with business priorities and cognitive bottlenecks. Treat agents as digital teammates, not just as tools.
- Focus area 2: Design for composability and collaboration – Embrace multi-agent systems with modular architecture, semantic protocols, and dynamic delegation through arbiter agents.
- Focus area 3: Architect for multi-tenancy and control – Build scalable, tenant-aware infrastructure with shared agent services, centralized governance, and role-based access.
- Focus area 4: Build trust through identity, guardrails, and observability – Enforce traceability, runtime controls, and explainability to earn stakeholder trust.
- Focus area 5: Manage the lifecycle – Establish continuous integration and continuous deployment (CI/CD) pipelines, prompt versioning, telemetry, and continuous retraining to support agentic AI performance and efficiency.

- Focus area 6: Align agent models with business models – Monetize agent capabilities through usage-based models, internal ROI metrics, and commercial offerings.

You can use the recommendations in this guide to prepare your business for agentic AI at scale. It outlines how organizations must restructure around agentic AI, including building DevOps for agents (AgentOps) teams, interoperable systems, and change management strategies that scale adoption. It emphasizes decision-first thinking and alignment with the AWS Well-Architected Framework.

Intended audience

This guide is intended for enterprise architects, AI/ML engineering leads, and digital transformation strategists who are designing and scaling agentic systems, embedding AI into core business workflows, and operationalizing LLMs and autonomous agents in production environments. To understand the concepts and recommendations in this guide, you should be familiar with modern cloud-native architectures and distributed systems, large language models, foundation model capabilities, and the principles of AI governance, DevOps, and platform engineering.

Objectives

By implementing the recommendations in this guide, your organization can achieve the following business outcomes:

- Accelerated decision-making and workflow execution through autonomous, goal-oriented agents that reduce human bottlenecks and cognitive load.
- Scalable, cost-efficient deployments of intelligent capabilities across business units, through reusable, multi-tenant agent platforms.
- Greater resilience, trust, and governance in AI systems, which enables confident adoption in regulated, mission-critical, or customer-facing environments.

About this content series

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

Strategic foundations for agentic AI

Agentic systems are not new. Software agents, including robotic process automation (RPA) and decision engines, have existed for decades. But they were simple and deterministic, designed to follow predefined rules and symbolic logic to execute repetitive, low-variation tasks. With the rise of generative AI, the game has changed. Large language models (LLMs) can now interpret complex inputs, generate responses dynamically, and quickly synthesize knowledge. You can now scale agency without brittle or hard-coded logic. Now, agents can reason, make decisions, invoke tools, adapt to context, and coordinate with other agents across workflows. They can operate autonomously toward goals, maintain memory, and reflect on outcomes.

However, raw capability is not enough. Intelligence without integration yields novelty, not impact. To unlock value from powerful LLMs, enterprises must shift beyond isolated experiments to engineered ecosystems. Agents must be treated as production-grade services that operate under the same discipline as any enterprise system. That includes governance, observability, secure identity models, and lifecycle management. They must also result in real business outcomes, not speculative potential. These systems should be architected with clear boundaries for decision-making and fault tolerance. It's important to incorporate automated recovery mechanisms, real-time performance monitoring, and scalable resource management. This helps you handle the dynamic, non-deterministic nature of agent interactions while maintaining consistent service levels across enterprise workflows.

At a foundational level, enterprises must rethink how intelligence is embedded into the fabric of operations. Agents must be designed to integrate with core systems, comply with enterprise policies, and deliver measurable value. They need to operate at scale, across departments, domains, and user contexts. Operationalizing agentic AI is ultimately about use; it's the difference between deploying AI that performs isolated tasks and deploying agents that evolve your business model.

Agentic AI represents a new operating philosophy that requires a fundamental shift in how we approach systems, processes, and people to scale intelligence across the organization. Agents become strategic assets that amplify human capabilities. By integrating agentic AI into their operations, organizations can unlock insights that drive business value, augment human capabilities, and optimize complex workflows.

Strategic focus areas for agentic AI

To move from early prototypes to production-grade and value-generating systems, teams need a coherent strategy that blends architecture, process, and product thinking.

Many organizations still approach AI with a tool-first or model-centric mindset. Generative AI has amplified experimentation, but often without clear alignment to business strategy or measurable outcomes. Without a defined strategic role, agents risk becoming novel experiments that drain resources rather than deliver scalable value. To establish the strategic role of agentic AI, organizations must start with business priorities. Identify areas of cognitive overload, decision bottlenecks, or fragmented workflows where autonomy can provide relief. Use domain-specific problem statements to shape agent responsibilities. Treat agents as digital teammates—not tools—who can reason, delegate, and adapt.

Decision sciences is the discipline of combining data science, analytics, and behavioral modelling to improve decision-making. It should be integrated early in the agent architecture process to align the design with business outcomes. By identifying decision patterns, simulating trade-offs, and quantifying value impact, decision sciences can help you pinpoint where agentic autonomy can deliver the highest value. Decision sciences can accelerate decisions, reduce errors, and enable real-time adaptations. This data-informed foundation grounds agent design in measurable insights, and it enables tighter integration with existing enterprise technologies, such as rules engines, analytics platforms, and predictive models.

To help establish the strategic role of agents, this section introduces foundational focus areas that form the backbone of operationalizing agentic AI. Each maps to a core job to be done from the perspective of a technical leader, architect, or product owner who is responsible for how agents are conceived and designed. These focus areas are not sequential steps. Each is worth revisiting throughout the system lifecycle to cultivate resilient, scalable, and monetizable agent ecosystems.

This section contains the following focus areas:

Focus area 1: Clarify agent intent and scope

Job to be done: "Help me make sure that each agent solves a real problem with clear boundaries, not just a cool demo."

Agentic AI is not just about building capability. It's about solving the right problem, in the right way, for the right outcome. That starts with being completely clear on the intent of the agentic AI solution.

Strategy

Too often, organizations start with what the model can do (such as call APIs, answer questions, or generate summaries) and retrofit a use case around it. This leads to scope creep, poor integration, and agents that are technically impressive but operationally useless. Instead, start by defining the agent's role through specific questions like the following:

- What specific outcome is the agent responsible for?
- Who is it acting on behalf of?
- Who benefits?
- Where does the agent's autonomy begin and end?
- What happens when it fails?

A well-scoped agent has a clear job, defined responsibilities, and measurable success criteria. Don't think of the agent as an assistant or chatbot. Instead, give it a job title. Think of it as a customer success agent, a product returns handler, or a compliance monitor.

When engaging stakeholders or customers, emphasize the scalability and adaptability of agentic AI systems. These agents evolve with the business, continuously improving through learning and feedback. To reduce resistance and accelerate adoption, highlight how agentic tools are designed with worker empathy in mind. They provide transparency, control, and optional override mechanisms that build trust. Rather than replacing people, agents augment human capability and decision making, helping employees to stay in the loop and focus on high-value tasks.

The key to successful implementation is aligning agentic AI with specific, high-impact business outcomes. Encourage teams and partners to start with focused pilot projects that solve visible pain points. Quick wins generate measurable return on investment (ROI), build internal buy-in, and create momentum for broader adoption.

To guide adoption and maturity, organizations can frame agent design along an evolutionary model. The agent autonomy, complexity, and business impact progressively increases. The following are the stages of this model:

- *Observer agents* surface insights from noise. An example is a market sentiment agent that tracks brand perception across digital channels.
- *Assistant agents* support human decision-making. An example is a deal advisory agent that synthesizes competitor data and market conditions for sales teams.
- *Autonomous agents* act independently within defined boundaries. An example is a resource allocation agent that dynamically adjusts cloud infrastructure based on demand.
- *Orchestrator agents* coordinate multi-agent workflows. An example is a supply chain optimization agent that manages interactions between inventory, logistics, and forecasting agents.
- *Innovator agents* generate new strategic possibilities. An example is a business model innovation agent that analyzes market trends and recommends new revenue streams.

Framing agents around these strategic outcomes and maturity levels increases focus, accelerates adoption, and builds stakeholder confidence.

To support alignment in this focus area, AWS services, such as Amazon Quick, can visualize key performance indicators (KPIs) that are linked to agent-driven outcomes. You can use Amazon CloudWatch to monitor agent behavior, performance metrics, and system health in near real time. Use the operational feedback to tune agent interactions and resource use. AWS CloudTrail can provide visibility into agent activity and integration patterns during early experimentation and refinement phases.

Business value of defining intent and scope

The adoption of agentic AI represents a pivotal shift in how organizations approach digital transformation and operational excellence. This is not merely about automation. It is about enabling intelligent autonomy that accelerates decision making and value realization.

Key business drivers include the following:

- **Competitive advantage** – Early adopters gain strategic advantage through faster insights, better service, and adaptive operations.

- **Customer experience enhancement** – Agents offer real-time, personalized, always-on support that boosts satisfaction and loyalty.
- **Operational efficiency** – Agentic AI significantly reduces human cognitive load by automating complex, repetitive decision tasks. This allows staff to focus on higher-value activities and can reduce costs.

Real-world use cases across industries include the following:

- **Financial services** – AI agents could deliver personalized financial advice and detect fraud.
- **Healthcare** – Triage and treatment plan agents could improve clinical throughput.
- **Retail** – Agents could act as intelligent shopping assistants or optimize inventory in real time.
- **Manufacturing** – Agents could perform predictive maintenance or coordinate supply chains.

Focus area 2: Design for composability and collaboration

Job to be done: "Let me build agents like I build services – modular and testable, so that they can be composed and orchestrated as needed."

Many AI efforts begin as monolithic, model-centric pilots. They're useful, but they're hard to scale across domains or adapt to complex problems. Value compounds when these agents are designed to interoperate. In technology, *composability* is the act of combining modular components to create a flexible, scalable solution that can adapt to change. Without composability, intelligence becomes locked within specific workflows. Furthermore, agent collaboration introduces orchestration, state management, and protocol negotiation complexities that traditional automation teams might not be equipped to handle.

Strategy

Embrace the multi-agent paradigm. Model agents like organizational departments: modular, specialized, and interoperable. Define clear interfaces, shared context formats, and standard communication protocols, such as Model Context Protocol (MCP) or Agent2Agent (A2A). Adopt multi-agent orchestration patterns, such as swarm, graph, or hierarchical coordination. These patterns help agents discover capabilities and request services from one another dynamically, either in parallel, sequential, or consensus-driven workflows, depending on the task structure and trust level.

To promote scalable and governed collaboration, use an *arbiter agent*. This kind of agent is a neutral authority that facilitates task delegation based on known capabilities and fallback strategies. While not a centralized controller, an arbiter agent plays a critical role in trust and compliance. It makes sure that sensitive or regulated tasks are routed only to agents that meet identity and policy requirements. It acts as a gatekeeper for policy-bound workflows. It enforces isolation and enables explainable delegation. Crucially, an arbiter agent is not a bottleneck; it coexists with self-coordinating agents that operate in a horizontal, peer-to-peer manner. These agents delegate sub-tasks, share context, and resolve dependencies directly.

This hybrid model supports both deterministic assignment (through the arbiter agent) and emergent collaboration. It blends structure with flexibility. Within this architecture, agents can be classified into the following specialized roles:

- *Decision agents*, such as policy enforcers, resource allocators, and risk evaluators
- *Knowledge agents*, such as context aggregators, pattern recognizers, and anomaly detectors
- *Execution agents*, such as task executors, quality controllers, and integration managers

To coordinate effectively, multi-agent systems must support robust interaction protocols for state management, failure recovery, and conflict resolution. This promotes stability and accountability even as agents operate independently.

Establish clear rules for scaling, such as load-based agent instantiation, context-aware resource allocation, and automated capability discovery and registration. These measures help the system to grow dynamically in response to demand or complexity.

Design agents to be ready-to-use modules within a distributed messaging substrate. For example, you might use Amazon EventBridge with A2A or MCP rather than siloed services. Adopt versioning, CI/CD pipelines, and agent templates to support system stability while accelerating internal adoption and lifecycle evolution. Encourage code reuse and standardization to reduce integration friction and promote a resilient ecosystem.

Collaboration is a force multiplier. It unlocks scale, specialization, and resilience across multi-agent environments. To support this dynamic collaboration, organizations should architect a lightweight control plane for agent coordination. This control plane includes the following:

- Capability registries that define what each agent can do and support versioned metadata for peer discovery

- Task arbitration logic that uses arbiter or supervisor agents to route tasks based on context, availability, and policy
- Lifecycle and state tracking that enables real-time decision context and safe handoffs

Control planes make sure that multi-agent systems remain extensible, policy-aligned, and fault-tolerant, without centralizing authority or slowing operations.

However, multi-agent environments also bring operational challenges. Maintaining context across agent interactions, managing shared state, and coordinating actions can drive complexity and cost. Costs can increase if you use LLMs that consume tokens during inter-agent communication. These costs must be weighed against the compounded business benefits of intelligent autonomy at scale.

To address these challenges, consider agentic platforms that abstract key concerns, such as the following:

- Standardized communication protocols and semantic formats
- Built-in orchestration logic and dynamic routing
- Shared context and memory management between agents
- Fallback handling and graceful degradation during failures

For teams adopting multi-agent strategies, the best approach is to start small and design for scale. Begin with targeted single-agent solutions that solve real problems. Then, progressively compose these agents into a cooperative system where each can discover, coordinate, and delegate based on shared goals and system-wide context.

Importantly, robust error handling and graceful degradation must be primary design principles. Multi-agent systems should be capable of continuing partial workflows or initiating backup logic when agents are unavailable or fail. This promotes reliability without rigid coupling.

AWS services offer robust features to support this architecture at scale. Amazon EventBridge and EventBridge Pipes provide the structured, event-driven backbone for multi-agent messaging. For managing modular behavior, AWS AppConfig enables safe, dynamic configuration toggling across agent instances. To support shared context and memory management, use Amazon DynamoDB for lightweight, tenant-aware state persistence and fast context retrieval across agents. You can use Amazon Simple Storage Service (Amazon S3) for storing structured prompt histories, shared artifacts, or agent-generated outputs. For more complex workflows that require stateful coordination, AWS Step Functions can orchestrate long-running processes with checkpoints

and error recovery logic. Together, these services help you create composable, resilient, and semantically connected multi-agent systems that scale with enterprise demands.

Business value of multi-agent systems

While many organizations begin their AI journey with single-agent solutions, the full potential of agentic AI is unlocked through scalable multi-agent systems. These systems are key to solving complex, distributed problems and creating robust, flexible AI ecosystems that evolve with business needs.

The core business benefits of multi-agent systems include the following:

- **Scalability** – Tasks and workloads can be distributed across specialized agents to increase capacity and performance.
- **Flexibility** – Agents can be added, replaced, or modified with minimal disruption, enabling agility in dynamic environments.
- **Resilience** – System stability is preserved even when individual agents fail, thanks to redundant roles and intelligent failover.
- **Specialization** – Purpose-built agents perform tasks with greater efficiency and precision.
- **Cost efficiency** – Reusable agent components accelerate development and reduce the cost of new capability deployment.

While multi-agent systems require more upfront planning, they deliver long-term agility, speed, and innovation capacity. Enterprises that invest in flexible agent collaboration architectures are positioned to deploy new AI capabilities rapidly, adapt to changing demands, and lead in an increasingly agent-driven competitive landscape.

Focus area 3: Architect for multi-tenancy and control

Job to be done: "Help me scale agent usage across multiple customers without losing control, accountability, or visibility."

Early prototypes are fine for proving value in isolation, but most businesses need to simultaneously support multiple customers, departments, or workflows. That means each agent must operate within clearly defined policy, data, and identity boundaries. Without multi-tenancy, operations become brittle and costly, and governance becomes a patchwork.

Strategy

Follow principles from software as a service (SaaS) architectures. For example, design for tenant isolation, policy enforcement, and resource control. Architect agents and orchestration platforms with tenant-aware memory, configuration, and identity. To enforce boundaries, use tagging, role-based access control (RBAC), and identity and access management scoping.

Adopt a unified observability layer where agent telemetry is aggregated by tenant context. Implement centralized policy engines and config-based capability toggling to enforce dynamic behavior rules.

Build agent deployment as a service. Enable internal teams or customers to consume agent capabilities as scalable, governed APIs. AWS provides a strong foundation for these patterns. You can use Amazon Cognito for managing user and tenant identity, AWS Organizations and service control policies (SCPs) for cross-account governance, and AWS Resource Access Manager (AWS RAM) for securely sharing capabilities. Additionally, AWS AppConfig can dynamically manage agent behavior by tenant or environment. These services help enforce boundaries and policies while supporting shared infrastructure.

This transition from static deployment to dynamic provisioning turns agentic AI into an enterprise-wide platform.

Business value of multi-tenant agent platforms

Multi-tenancy is more than an architectural convenience—it's a business accelerator. As intelligent agents proliferate across departments and teams, organizations must support growth without duplicating infrastructure or fragmenting governance.

The key business benefits of multi-tenant systems include the following:

- **Scalability** – A multi-tenant agent platform allows internal teams, business units, or clients to onboard AI capabilities faster without needing bespoke environments.
- **Cost efficiency** – Shared infrastructure minimizes redundant deployments, consolidates operational costs, and simplifies maintenance across environments.
- **Governance and risk reduction** – Centralized policy controls, identity models, and observability help agents operate more securely and in compliance, across all tenants.
- **Service reusability** – To promote reuse and reduce duplication, tenant-aware agents can be offered as internal services, such as for enrichment, compliance, or summarization.

Example use cases for multi-tenant systems include the following:

- A compliance agent that is deployed across subsidiaries adapts its logic to local regulations through tenant-specific configuration. This eliminates the need to build separate agents for each region.
- An internal workflow automation agent serves multiple departments with different data boundaries and permissions. It maintains isolation while accelerating task fulfillment.

By designing agents as multi-tenant-aware services, organizations avoid the overhead of siloed AI initiatives. Instead, they foster a unified intelligence platform. This architecture enables scalable rollout, operational consistency, and better ROI. It also makes it easier to expand AI adoption across the enterprise.

Focus area 4: Build trust through identity, guardrails, and observability

Job to be done: "Give me confidence that agents will act safely and predictably, especially when no one's watching."

Autonomous agents challenge traditional control models. Their ability to reason and act independently introduces risk if they're not properly managed. Without clear ownership, auditability, or policy constraints, they may drift from their intended behavior. Building organizational trust requires more than just technical reliability. It demands explainability, accountability, and consistency.

Strategy

Build an identity-first control system as the backbone of trusted autonomy. Each agent must operate with a verifiable identity, scoped permissions, and traceable execution history. Agents should be embedded in a zero-trust framework that includes tenant binding, contextual access inheritance, and runtime enforcement through guardrails and policy engines. This allows you to audit, reverse, or restrict agent actions based on organizational rules and risk posture.

Embed trust enforcement at runtime through intelligent guardrails. This includes rate controls and throttling based on behavioral patterns or workload conditions, resource boundaries enforced alongside auto-scaling, and decision scoring to evaluate risk. Build triggers to engage human-in-the-loop workflows when thresholds are exceeded.

Every agent must also be transparent and explainable. Embed structured telemetry through logging, traces, and reasoning summaries to expose decision logic. Support decision trails and impact tracing. This helps you connect agent actions back to key metrics or outcomes. Implement drift detection mechanisms that monitor deviations from expected behavior or policies.

Introduce reflective agents that continuously observe agent behavior and system patterns. They should flag anomalies or inconsistencies in real time. These agents contribute to governance feedback loops that can initiate revalidation, adaptation, or decommissioning of capabilities.

Establish governance boards that review agent policies, approve capability changes, and oversee incident response protocols. Trust must be earned, measured, and continually reinforced.

AWS provides a strong foundation for implementing this trust framework:

- AWS Identity and Access Management (IAM) enforces role-based execution and permission boundaries
- Amazon CloudWatch and AWS X-Ray support full visibility and traceability.
- Amazon GuardDuty and AWS Config detect security anomalies or policy drift.

Together, these services enable identity enforcement, runtime safety, and trust-based governance at scale. They can help make autonomous systems both powerful and dependable.

Business value of trusted autonomy

As agents become more autonomous, trust becomes a critical driver for enterprise adoption, governance, and operational performance. Establishing a foundation of identity, observability, and guardrails helps organizations to scale agentic AI into sensitive domains, without sacrificing governance or control.

Key business drivers include the following:

- **Governance assurance** – Strong identity models, audit trails, and permission boundaries reduce compliance risk and support regulatory alignment.
- **Operational continuity** – Runtime guardrails and anomaly detection help prevent unintended behaviors and support self-recovery from edge-case failures.
- **Stakeholder confidence** – Decision explainability and telemetry build trust with internal stakeholders, risk managers, and external auditors.

- **Incident resilience** – Embedded observability accelerates root cause analysis and response time when issues arise.

Example use cases include:

- In financial services, fraud detection agents must expose their reasoning, log every action with traceable identity, and operate under tightly scoped IAM roles.
- In healthcare, autonomous triage agents must enforce runtime safety checks, escalate to human review when thresholds are met, and provide full logs for clinical oversight.

By embedding trust mechanisms into the agent lifecycle, organizations can permit their systems to operate autonomously with accountability. This foundation reduces risk and empowers agents to act on behalf of the business with transparency and integrity.

Ultimately, trusted autonomy accelerates adoption by giving both users and leadership the confidence to scale intelligent agents across core operations.

Focus area 5: Manage the lifecycle

Job to be done: "Make sure my team can improve agents over time, without chaos or heroics."

Unlike traditional applications that are shaped only by code, agent behavior is also shaped by prompts, memory, tools, and training context. These factors drift over time. Drift erodes reliability, inflates cost, and makes debugging near impossible. Without lifecycle controls, agents stop delivering value and start accumulating risk.

Strategy

Establish DevOps for agents (AgentOps) as a practice. Integrate CI/CD pipelines that are tailored for agents. Use these pipelines to test prompt outputs, validate tool integrations, and profile cost-performance behavior. Maintain version histories of prompts, policies, and model interactions.

Use feedback loops from observability data to initiate retraining, prompt tuning, or agent retirement. Incorporate system-wide reflection mechanisms, such as an improvement register, to institutionalize learning.

Build a performance telemetry dashboard that shows decision accuracy, latency, cost, and reliability. To streamline and accelerate lifecycle management using AWS infrastructure, teams can use agent toolkits. One example is the Strands Agents SDK, which provides structured tooling

for prompt versioning, tool registration, and CI/CD integration with AWS services, such as AWS CodePipeline, AWS Cloud Development Kit (AWS CDK), and AWS Lambda. Additionally, use Amazon S3 and Amazon Elastic File System (Amazon EFS) for storing agent artifacts and training data. Use AWS Step Functions to automate complex retraining or validation workflows. You can use Amazon SageMaker AI when agents require custom model tuning or fine-tuning workflows beyond LLM orchestration. Lifecycle discipline transforms agents from experiments into durable, evolving assets.

Over time, this lifecycle system forms the backbone of innovation. It helps you to recompose, retrain, and redeploy capabilities with agility. This transforms the agent layer into a living system, capable of evolving in response to both feedback and opportunity.

Business value of lifecycle management

Effective lifecycle management is a key driver of agent performance and cost efficiency. It makes sure that intelligent agents continue to deliver accurate, reliable, and value-aligned outcomes as they evolve. Agents don't remain valuable by default. They must evolve in sync with changing business requirements, workflows, and data environments. A disciplined AgentOps team helps agents remain accurate, efficient, and aligned with enterprise goals over time.

Key business drivers include the following:

- **Performance consistency** – Continuous testing, prompt validation, and retraining help agents maintain decision quality across changing conditions and datasets.
- **Cost optimization** – Telemetry-driven profiling identifies inefficient tools, high-token prompts, or unnecessary executions. Then, you can tune to reduce operational costs.
- **Faster iteration** – Lifecycle automation with CI/CD accelerates development cycles, helping teams to experiment, deploy, and improve agents with confidence.
- **Risk reduction** – Prompt versioning, rollback support, and structured evaluation mechanisms help prevent regressions and support safe, reliable change management.

Example use cases include the following:

- A customer support agent is monitored for latency, model cost, and user feedback. Observability reveals a cost spike, which prompts re-tuning of its embedded prompts and fallback model logic.
- A contract summarization agent is updated based on feedback from legal teams. Versioned prompts are tested in sandboxed environments before production release, supporting safety and quality.

With structured lifecycle management, organizations move beyond reactive maintenance to proactive, continuous improvement. Agents become adaptive digital assets that are measured, refined, and revalidated against business goals. This practice transforms agent ecosystems into high-performing, cost-aware, and resilient systems that deliver durable value while keeping pace with change.

Focus area 6: Align agent models with business models

Job to be done: "Show me the impact, so that I can justify continued investment."

Even technically capable agents become liabilities if they aren't tied to business outcomes. Agents must serve either efficiency, monetization, or strategic differentiation. Yet most businesses struggle to define how agents fit into pricing, packaging, or usage models. Without clear alignment to business value, it's hard to justify scaling or even maintaining the investment.

Strategy

Adopt product management practices. Treat agents as monetizable services with a measurable ROI. Define pricing strategies based on decisions, sessions, or outcomes. Then, package agent capabilities into tiered offerings that are aligned with customer segments or internal business units.

To promote sustainability, organizations must capture both direct value and growth multipliers through agent deployment. Consider using the following ROI metrics to measure immediate value:

- **Cost per decision** – Benchmark agent processing costs against human equivalents.
- **Time compression** – Quantify the value of accelerated cycles, such as faster sales or approvals.
- **Error reduction** – Measure savings from improved accuracy, consistency, and compliance.

Beyond these immediate gains, agents can unlock the following long-term growth opportunities:

- **Capability stacking** – Combine agent services to create domain-specific vertical solutions.
- **Network effects** – Increase value through multi-agent ecosystems where coordination compounds utility.
- **Market extension** – Generate new revenue streams through externally consumable, agent-enabled services.

Create feedback loops from business metrics (such as cost savings, conversion lift, or time-to-resolution) to drive continuous agent evolution. Analyze usage telemetry and user satisfaction scores to refine your value alignment and roadmap priorities. By linking agent capabilities directly to business models, organizations position themselves to capture sustainable, compoundable value—not just technical outcomes.

The following AWS services support this alignment by providing robust tracking and monetization frameworks:

- AWS Cost Explorer and Amazon CloudWatch deliver insight into per-agent costs and operational efficiency.
- Amazon API Gateway enables metered access, rate-limiting, and tiered pricing for agent endpoints.
- AWS Marketplace provides a channel for publishing agents and agentic solutions as commercial products.

These services help you to transform agent functionality into scalable, value-driven digital offerings that align with enterprise growth and monetization strategies.

Evolving software delivery for agentic AI

Modern software delivery has been shaped by a simple assumption—that you control the systems you ship. You define requirements, write logic, test against expected outcomes, and deploy predictable services. Even Agile and DevOps approaches still rely on the principle that each sprint delivers something deterministic, verifiable, and largely within human oversight.

Agentic AI upends that foundation. Agentic systems interpret, reason, and adapt rather than follow scripts. Their behavior depends on the code you write, the context they operate in, the inputs they're given, the tools they can access, and the goals they're assigned. In short, they don't follow orders; they pursue outcomes.

This makes delivery less about control and more about alignment. Rather than providing instructions, you must shape how it behaves. This means that the traditional software development lifecycle (SDLC) no longer fits because it was designed for logic-based, human-controlled systems.

This section contains the following topics:

Zones of intent for agentic AI

Instead of rigid stages, such as *define*, *build*, *test*, and *release*, we need a model that embraces autonomy, uncertainty, and emergence. Instead, you use zones of intent. A *zone of intent* defines a bounded space where an agent can operate with autonomy, within constraints. The goal is to shift from micromanaging every task to designing environments where agents can act, learn, and collaborate safely. You specify the what (the desired outcome), the why (the intent), and the guardrails (the constraints, policies, and trust boundaries). Given those boundaries and this information, the agent figures out the how.

Instead of an assembly line, think of the environment as an airspace. You control who can enter, what they can do, and where they can go. But once inside, they're free to navigate as needed. That's how agentic systems scale without chaos.

This isn't just a philosophical shift; it's a practical one. The non-deterministic output of agent-based systems can't be fully tested through unit tests. It can't be versioned like static binaries. Agents

change over time, adapt to new data, and interact with other systems in unpredictable ways. Trying to deliver them using traditional models leads to fragile, unscalable architectures. At worst, it leads to false confidence in systems that you can't actually govern.

When teams embrace intent-based delivery, they gain two advantages:

- **Control where it matters most** – They define boundaries instead of outputs.
- **Scalability through delegation** – They enable agents to handle complexity humans can't hardcode.

This is how you move from isolated prototypes to real, production-grade agentic systems that can repeatedly and reliably deliver value.

Evolving the delivery lifecycle for agentic AI

To support intelligent, adaptive behavior, the SDLC must be reframed from deterministic control to adaptive intent. The following are the changes necessary to evolve the traditional SDLC for agentic AI:

- *Planning* becomes *intent design*. Teams define goals, constraints, and expected agent behaviors. Policies and success criteria are framed in terms of alignment, not logic.
- *Architecture* becomes *scaffolding*. Teams focus on defining roles, interfaces, guardrails, fallback mechanisms, and observability rather than scripting every decision path.
- *Testing* becomes *behavioral evaluation*. Rather than asserting specific outputs, teams validate whether agents stay within acceptable bounds and fulfill intent under varied inputs.
- *Deployment* becomes *continuous orchestration*. Agentic systems are deployed with runtime controls, live monitoring, and feedback channels that enable real-time tuning.
- *Iteration* becomes *feedback* and *adaptation*. Instead of traditional code-change patch cycles, teams observe how agents evolve, where they succeed, or when they drift. As necessary, teams intervene through updated constraints, retraining, and adding or modifying control mechanisms.

Existing practices that focus on iteration, experimentation, and rapid feedback are halfway there. The shift to agentic systems isn't a rejection of Agile principles. In fact, it's a natural evolution of them. Agile thinking emphasizes adaptability, feedback, and working solutions over rigid plans. That aligns perfectly with the nature of agentic systems, which learn, adapt, and respond to

context in real time. If you're already running short cycles, validating assumptions quickly, and managing uncertainty through continuous delivery, you're well equipped to lead this transition.

But there are key differences. The traditional Agile approach assumes that the thing being delivered is deterministic. It assumes that, once built, the thing will behave consistently and predictably, with repeatable outcomes for the same inputs. This repeatability helps you debug, test, and iterate with confidence. Agentic systems break that model. They're probabilistic, context-sensitive, and capable of evolving independently. That means some Agile practices become less useful, such as velocity tracking based on story completion, strict acceptance criteria, or deterministic sprint planning.

The following aspects of the traditional SDLC apply to agentic AI:

- Iterative development and delivery
- Customer feedback as a primary signal
- Cross-functional collaboration
- Continuous integration and deployment

The following aspects of the traditional SDLC must evolve for agentic AI:

- Redefine *done* as *aligned to intent.* Focus on whether the agent's behavior satisfies its intended goal within the defined constraints.
- Shift from acceptance criteria to behavioral guardrails.
- Expand the definition of *done* to include runtime readiness, which includes observability, explainability, and feedback mechanisms that support continuous learning and trust.
- Prioritize real-time feedback loops and behavior tracking over upfront planning

The good news is you don't need to throw out the SDLC playbook. You just need to evolve it from managing code to shaping conduct. In agentic systems, success isn't just about whether software runs, but how it behaves.

Preparing teams for agentic AI

Software engineering isn't going away. It's evolving. The job shifts from writing functions to shaping frameworks and control mechanisms for intelligent behavior. In the world of agentic AI, building is no longer the hard part—managing emergence is. For most engineering teams, the

evolution feels like a mindset shift rather than a technical leap. Instead of asking "What will the system do?" the question becomes "What have we empowered it to pursue, and how will we know if it's staying on course?"

For engineering teams, the evolution toward agent AI requires the following changes:

- **A cultural shift** – Teams must become comfortable with uncertainty and autonomy in systems they don't fully control.
- **New roles** – Intent designers, behavioral testers, and observability engineers become core to delivery.
- **Shared language** – Teams need a clear, shared understanding of goals, guardrails, and success signals, just like how they once needed specs and test cases.

As generative AI matures, we'll see more agentic systems interacting with customers, products, and operations. The organizations that succeed won't be the ones with the best models. It will be the ones that can integrate agents into real-world workflows with confidence, control, and velocity. That means that the delivery models and engineering teams must evolve together. Zones of intent give you the abstraction to do that. They help you operationalize autonomy without surrendering accountability. They also offer a shared framework across teams to help govern systems that can't be hard-coded.

For more information about preparing teams for agentic AI, see the Preparing the business for agentic AI at scale section of this guide.

Preparing the business for agentic AI at scale

As the focus areas described in this guide converge, agentic AI shifts from isolated functions into a unified intelligence layer that can be understood as a capability platform. This platform doesn't just execute tasks. It evolves, adapts, and coordinates across domains. Agents become modular, reusable, and discoverable services that accelerate innovation, reduce cognitive load, and drive measurable outcomes across the enterprise. This platform view sets the stage for scalable intelligence embedded throughout the operating model.

Operationalizing agentic AI requires more than deploying intelligent agents. It demands a fundamental transformation in how the business organizes teams, designs processes, and governs technology. Just as the shift to cloud or DevOps redefined operating models, agentic AI introduces a new era of decision automation, continuous learning, and autonomous coordination. Success depends on aligning the systems, the people, and the processes around this new operating philosophy.

This section contains the following topics:

Aligning teams and ownership models

The first step toward maturity is cross-functional alignment. Businesses must establish AgentOps teams that include AI/ML practitioners and domain specialists, such distributed systems architects, software engineers, product owners, compliance leads, and platform architects. These teams jointly own the entire lifecycle of an agent, from design and deployment to retraining and monitoring.

Agent provisioning and release should follow cloud-native practices, such as using the AWS Cloud Development Kit (AWS CDK) and AWS CodePipeline for infrastructure as code and automated deployment. This structure fosters shared accountability and accelerates iteration. Just as DevOps

unifies development and operations, AgentOps connects intelligence with governance and execution.

To be effective, these teams also need a shared language. Business stakeholders must understand what agents are, how they operate, and what outcomes they drive. Training and internal enablement are essential. By demystifying agents and embedding this mental model into everyday conversations, organizations unlock broader participation and more aligned innovation.

To accelerate the development and integration of agents using AWS services, teams can adopt frameworks like Strands Agents SDK, which offers CLI-based tooling for scaffolding, configuring, and packaging agents. Strands Agents is designed to work seamlessly with AWS infrastructure, such as Amazon Bedrock, AWS Lambda, Amazon EventBridge, the AWS CDK, and AWS CodePipeline. It enables rapid prototyping and deployment while maintaining production-grade standards.

But structure and tooling alone are not enough. Scaling agentic AI requires deliberate cultural, educational, and leadership readiness to make sure that adoption takes root across the organization.

Managing change and organizational readiness

Successfully scaling agentic AI requires more than deploying infrastructure or intelligent agents. It demands a structured approach to organizational change. This includes cultural readiness, skills development, metric-driven feedback loops, and executive alignment to make sure that adoption is both intentional and sustainable.

Foster cultural evolution

- Position agents as teammates, not replacements, to reduce resistance and build trust.
- Communicate transparently about agent capabilities and limitations to set realistic expectations.
- Establish clear handoff protocols for when agents should escalate decisions to a higher authority or delegate parts of the process to a human collaborator.

Establish a skills development framework

- Deliver role-based training tailored to engineers, product managers, domain leads, and compliance officers.

- Create centers of excellence to share best practices, tooling patterns, and reusable assets.
- Pair AI specialists with domain experts through mentorship programs to bridge knowledge gaps.

Define metrics and feedback loops

- Anchor technical and business KPIs to strategic value to assess impact. Examples of value include decision latency, resolution accuracy, and cost savings.
- Systematically and continuously capture user feedback to surface friction points and adoption challenges.
- Conduct regular retrospectives to evaluate agent performance, usage trends, and improvement opportunities.

Align leadership from the top

- Obtain executive sponsorship by linking agent initiatives to strategic outcomes and ROI.
- Form cross-functional governance committees that include both technical and business leadership.
- Tailor communication strategies for clarity and engagement across all organizational levels.

This systematic approach to change management makes sure that technology implementation is matched by organizational maturity. It creates a foundation for trust, adoption, and long-term business value.

Architecting for interoperability and collaboration

Isolated agent deployments deliver local wins. But enterprise value emerges when agents can discover, invoke, and collaborate with one another dynamically. This means defining standards for agent registration, authentication, and capability exchange. Architecturally, this mirrors the shift from monoliths to microservices, which are composable, reusable, and loosely coupled units that solve complex problems together.

Emerging protocols, such as A2A and MCP, are foundational. The enable semantic interoperability across agents, tools, and memory systems. A2A supports peer-level interaction, which allows agents to negotiate task ownership, share context, and coordinate workflows. MCP complements this by offering shared schemas for exchanging contextual data between agents and their environments. It standardizes how functions are invoked, APIs are accessed, and states are

maintained. Together, these protocols promote extensibility, consistency, and long-term maintainability across the agent ecosystem.

Governance remains critical. Control layers, such as arbiter agents, enable policy-aware delegation without introducing centralized bottlenecks. These agents act as trust brokers. They enforce boundaries while letting other agents self-organize. Agent collaboration helps organizations to scale their agentic AI ecosystems with both agility and trust.

Building governance into an agentic fabric

With greater autonomy comes greater risk. Governance must be embedded into the agent architecture from the first day. This includes defining policy boundaries that scope what agents are allowed to do, enforcing identity models that determine who they act on behalf of, and implementing explainability and traceability. Observability systems must capture telemetry on agent behavior by using services such as Amazon CloudWatch and AWS X-Ray, which provide centralized logging and distributed tracing across agent workflows. Reflective agents can continuously audit and assess performance based on these telemetry feeds.

Governance must also evolve as the agent ecosystem matures. As agents become more capable and more autonomous, oversight mechanisms must become more adaptive. Policy updates, capability gating, and runtime behavioral constraints need to be dynamic and enforceable at scale. Trust isn't a bolt-on feature. It is continuously reinforced through architecture, behavior, and process. AWS Identity and Access Management (IAM) and AWS AppConfig play a critical role in enforcing secure identities, runtime permission boundaries, and environment-specific behavior toggles across agents.

Adopting a decision-first operating mindset

Traditional automation focuses on process efficiency, which is running predefined scripts or workflows faster and more reliably. Agentic AI, by contrast, introduces decision-first automation. Agents assess context, weigh options, and adapt behavior in real-time. This shift from an execution-first to decision-first mindset requires new thinking about success metrics and outcomes. Instead of measuring success exclusively by task completion, success for agentic AI is measured by how well the decision aligned with the intent, policies, and evolving conditions.

Rather than measuring only task completion or cycle time, organizations must evaluate decision quality, time-to-action, and responsiveness to change. KPIs should include metrics such as:

- **Decision quality** – How well did the agent personalize its response to the specific user or scenario? Did it make nuanced decisions that are aligned with business objectives and user context?
- **Time-to-action** – How quickly and intelligently did the agent assess a situation and respond? Was the latency low enough to feel adaptive and human-like?
- **Cognitive offload** – How much manual analysis, triage, or routine decision-making was the agent able to handle on behalf of a human? Did it reduce effort or just shift it?

Businesses that embrace the decision-first mindset can become more resilient, adaptive, and capable of operating at a new level of complexity.

Scaling with purpose and intent

Successfully scaling agentic AI is not about experimenting with more tools. It's about building a durable layer of enterprise intelligence. This requires investments in platform infrastructure, operational culture, governance frameworks, and strategic alignment. Businesses must adopt an intentional approach. They must treat agents not as experiments but as core components of their digital operating model.

Aligning with the AWS Well-Architected Framework helps your systems meet enterprise standards for reliability, security, performance efficiency, and cost optimization. Tools such as the Strands Agents SDK can accelerate this journey by providing structured prompts, tool registration, and CI/CD readiness. This helps teams shift from experimentation to scalable delivery by using familiar AWS workflows.

Agentic AI isn't a tool; it's a shift in how intelligence is embedded into operations. Organizations that prepare accordingly can automate more, operate smarter, adapt faster, and create lasting advantage in an increasingly complex world.

Conclusion for operationalizing agentic AI

Agentic AI represents more than a technological shift. It marks the emergence of a new operating system for the enterprise. Organizations that embrace this transformation move beyond narrow automation use cases and build intelligence into the foundation of their operations. This shift is about redesigning how decisions are made, how systems adapt, and how outcomes are realized at scale.

In an era defined by growing complexity, real-time demand, and information overload, the traditional model of scripted automation has reached its limits. Success now hinges on the ability to embed intelligence directly into workflows in order to make systems that perceive, reason, act and evolve. Agentic AI can align autonomy with purpose, decision-making with governance, and adaptability with accountability.

This transition requires a move from execution-first to decision-first thinking. Agentic systems do not simply follow instructions. They interpret goals, weigh trade-offs, and pursue outcomes within defined constraints. In this context, success is measured not just by task completion. It is also measured by the quality, agility, and explainability of decisions made in real time. Organizations must rethink metrics, incentives, and system design to support agents that operate intelligently under uncertainty.

Operationalizing agentic AI is not a plug-and-play upgrade. It is an architectural and cultural transformation. It requires disciplined practices across lifecycle management, trust enforcement, interoperability, and alignment to business models. It also calls for the evolution of delivery models, such as shaping zones of intent, embedding runtime guardrails, and continuously aligning agent behavior with strategic outcomes. Teams must adopt shared language, shared ownership, and shared accountability for agent performance and safety.

Enterprise readiness can determine who thrives in this new environment. Organizations must invest in internal enablement, AgentOps capabilities, and governance frameworks that scale and create long-term value. Those who succeed can build smarter systems, and they can also build more adaptive, resilient, and insight-driven businesses.

This guide lays the foundation. It connects strategy to execution and prepares organizations to build scalable platforms of intelligent agents. The broader content series about agentic AI on AWS provides complementary guidance. To view the other guides in this series, see Agentic AI on the AWS Prescriptive Guidance website. This content series offers a roadmap to operationalize autonomy with discipline and intent.

To get started, identify a high-impact decision space where agents can deliver measurable improvements in speed, accuracy, or responsiveness. Then deploy a focused pilot agent that has instrumentation, governance, and feedback loops. Use this to validate the value hypothesis, generate internal momentum, and build trust in the approach. Momentum compounds through learning.

Agentic AI is not a destination; it is a capability layer that evolves alongside your business. It represents a long-term shift toward intelligence as infrastructure. Organizations that lead in this space can automate more, respond faster, adapt better, and build operational models that are capable of navigating complexity at an enterprise scale.

Resources for operationalizing agentic AI

AWS services

The following AWS services and features can help you build and operationalize agentic AI systems in the AWS Cloud:

- Amazon API Gateway can expose agent capabilities as scalable and offers usage-based pricing.
- AWS AppConfig offers runtime configuration management and feature toggling for agents across tenants or environments.
- Amazon Bedrock is a foundation model service that agents can use for reasoning, generation, and prompt execution.
- AWS Cloud Development Kit (AWS CDK) is an infrastructure as code service that you can use to deploy and manage agent stacks.
- AWS CloudTrail records event history so that you can track agent activity, audit trails, and integration behaviors.
- Amazon CloudWatch can manage logs, metrics, and alarms for monitoring agent performance and multi-agent collaboration behavior.
- AWS CodePipeline provides CI/CD automation that you can use to test, validate, and deploy agent code.
- Amazon Cognito is an identity service that you can use to manage user and tenant authentication in multi-agent systems.
- AWS Config offers compliance and drift detection for agent policy and environment configuration.
- AWS Cost Explorer can track agent-level usage and help align costs to maximize your ROI.
- Amazon DynamoDB is a storage service that you can use for agent memory, improvement logs, and contextual state.
- Amazon Elastic File System (Amazon EFS) is a shared file system that you can use for agent collaboration or intermediate processing across workflows.
- Amazon EventBridge is a core event bus that you can use to route tasks and orchestrate communication in the agent fabric.
- Amazon EventBridge Pipes can streamline event ingestion and routing for connecting agents and services.

- Amazon GuardDuty offers threat detection and anomaly monitoring that can support safe agent execution.
- AWS Identity and Access Management (IAM) helps you define fine-grained permissions for agent execution and data access.
- AWS Lambda is a stateless compute service that can run agent logic and swarm drones.
- AWS Marketplace is an external distribution platform that you can use to offer agent capabilities as commercial products.
- AWS Organizations is a cross-account governance and policy enforcement service that can help you manage multi-tenant agent infrastructure.
- AWS Organizations service control policies act as guardrails for controlling permissions at the account or organizational unit level.
- Amazon Quick is a generative AI-powered business intelligence (BI) platform that helps you analyze data, create visualizations, automate workflows, and collaborate with others across your organization.
- AWS Resource Access Manager (AWS RAM) can help you share capabilities between accounts and agent services.
- Amazon SageMaker AI is a service that you can use for model training, fine-tuning, and inference beyond foundational models.
- Amazon Simple Storage Service (Amazon S3) offers object storage for prompt libraries, model artifacts, and agent-generated data.
- AWS Step Functions is a workflow engine that can help you coordinate multi-agent flows and retraining pipelines.
- AWS X-Ray offers distributed tracing that you can use to track agent decision flows and service dependencies.

Other AWS resources

- Foundations of agentic AI on AWS
- Agentic AI patterns and workflows on AWS
- Agentic AI frameworks, protocols, and tools on AWS
- Building serverless architectures for agentic AI on AWS
- Building multi-tenant architectures for agentic AI on AWS

Document history

The following table describes significant changes to this guide. If you want to be notified about future updates, you can subscribe to an RSS feed.

Change	Description	Date
Initial publication	—	August 12, 2025

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

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.

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.

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.

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.

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.

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.

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.

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.

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.