Security for agentic AI on AWS

AWS Prescriptive Guidance

AWS Prescriptive Guidance: Security for agentic AI on AWS

Table of Contents

Security for agentic AI on AWS

James Schafer and Melanie Li, Amazon Web Services

January 2026 (document history)

While agentic AI systems might be a relatively new concept, many of the security risks they present have well-known, effective controls.

This guide provides practical security recommendations for developing hosted agentic AI systems the follow the perceive, react, act layer architecture. Hosted agentic AI systems can support various business outcomes with differing risk tolerance levels. Due to this variance, some best practices in this guide are more applicable depending on your use case. Adoption of suitable controls can be phased in and enhanced through the lifecycle of the system based on organizational priorities. It's critical to understand which threats are relevant to your workload in order to understand which controls will be effective in achieving alignment with your risk appetite.

For any threat identified, you should implement multiple controls across more than one security control type. This guide maps its best practices to the OWASP Top 10 for Large Language Model Applications. It also includes recommendations from multiple sources, focusing on the highest priority aspects for hosted agentic AI systems.

Intended audience

This guide is for architects, developers, and technology leaders who need to safely and compliantly operate AI-driven software agents. 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, DevOps, and platform engineering.

Objectives

This guide helps you do the following:

- Understand the design decisions that relate to securely operating hosted agentic AI systems
- Determine how to design and operate hosted agentic AI systems in accordance with your target risk posture
- Align functional controls with industry frameworks

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.

Key security concepts for agentic AI on AWS

By understanding the key functions within each agent, you can understand how to address the range of threats to it. An individual agent's function can be divided into three aspects: perceive, reason, act. This process is the cognitive cycle that is at the core of every agent's autonomous operation.

Securing agentic AI systems requires a hybrid approach that combines traditional distributed systems practices with AI-specific mitigations across the perceive, reason, and act layers. The perception and action layers can largely rely on conventional security frameworks that were developed for microservices architectures, including strong application, data, infrastructure, and operational controls. However, the reasoning layer introduces unique challenges due to the probabilistic nature of AI decision-making. It demands specialized threat-mitigation strategies. This architectural similarity to modern microservices helps organizations to use existing security frameworks to address the distinct risks introduced by agentic AI systems.

The comprehensive security posture for agentic AI systems must acknowledge that, unlike deterministic systems that produce consistent outputs, AI components exhibit probabilistic behavior. This creates new attack vectors and requires enhanced threat modeling that incorporates both traditional distributed system risks and emerging AI-specific threats, as outlined in the OWASP Top Ten for LLM Applications. Success depends on implementing defense-in-depth strategies across all three layers. This strategy should assume potential compromise at any level, and it should maintain operational effectiveness. This helps you implement security controls that are proportionate to the unique risks each layer presents while preserving the system's ability to achieve its intended goals through AI-driven reasoning and coordination.

This section includes the following topics:

Threats at the perception layer

The perception layer serves as the primary interface for data ingestion and interpretation. It processes diverse inputs, including user prompts, documents, sensor data, API responses, and inter-

agent communications. This layer's role in forming the agent's worldview makes it a high-value target for adversaries who seek to influence system behavior through data manipulation.

Primary threat vectors include:

- *Data poisoning*, where inputs are subtly modified to bias reasoning processes
- *Prompt injection attacks*, where malicious instructions are concealed within seemingly benign data sources

The expanding diversity of data sources compounds the challenge of maintaining input integrity and provenance tracking.

Threats at the reasoning layer

The reasoning layer represents the cognitive core of agentic AI systems. It consolidates planning, knowledge retrieval, decision-making, and memory management functions. Its central role in system decision-making makes compromise of this layer particularly damaging because it can undermine the integrity of the entire system.

Critical threats include:

- Input and context manipulation through information flooding attacks
- *Logic manipulation* that exploits agent biases toward efficiency or urgency
- *Memory poisoning* that introduces persistent false data or policies
- *Multi-agent collusion scenarios*, where spoofed communications disrupt coordination or propagate misinformation across the system.

For greater outcome control, the reasoning layer should have focused goals rather than broad decision-making capability. Amazon Bedrock guardrails provide automated reasoning validation capabilities. They can cross-check outputs against predefined policies and domain knowledge to support detection of anomalous behavior.

Threats at the act layer

The act layer translates reasoning outputs into concrete system interactions through API calls, database updates, and infrastructure changes. This layer presents the highest operational risk

because security failures directly impact production systems and sensitive data through legitimate agent capabilities.

Key risk areas include:

- *Tool misuse*, where adversaries use legitimate agent capabilities for unauthorized operations
- *Privilege compromise* through credential exposure or misconfiguration
- *Cascading failures*, where single compromised actions propagate across interconnected agent networks and dependent services

Security best practices for agentic AI systems on AWS

The best practices in this guide are divided into high-level categories that straddle multiple disciplines and technology domains. Many organizations have existing, well-established practices that range from application development to data governance. For that reason, each recommendation is labeled as either *AI-specific* or *General*. The *AI-specific* recommendations provide guidance for organizations with strong foundations that need to understand which additional practices to adopt. The *General* recommendations describe best practices that you should consider due to their value in strengthening security, including security for agentic AI systems.

This section includes the following categories of best practices:

1. System design and security recommendations for agentic AI systems

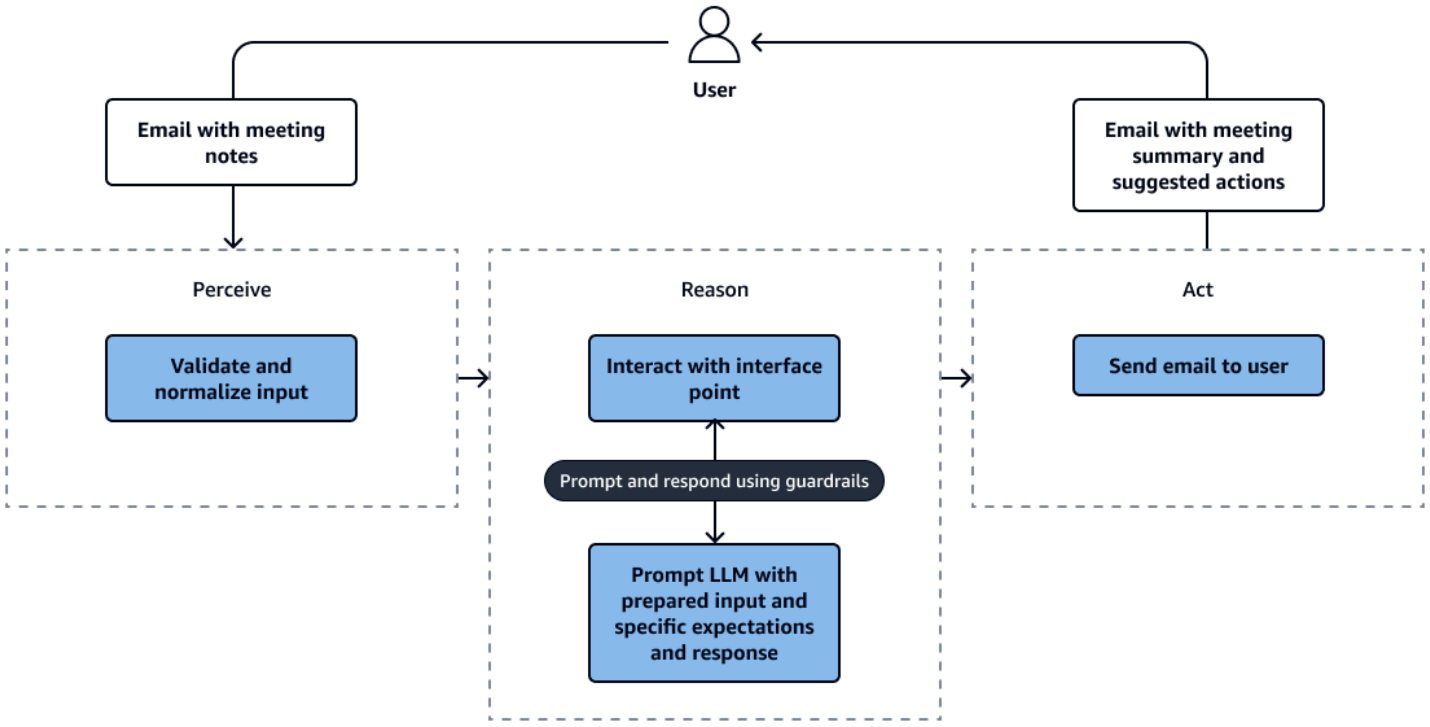
System design decisions can support the development of agentic AI systems with strong protections against well-documented large language model (LLM) application threats. Adoption of risk-aligned design patterns helps drive an outcome is in line with your business needs.

This section contains the following best practices:

1.1 Use deterministic execution logic unless AI is needed (AI-specific)

Implement coded logic for deterministic execution. Use AI functionality where it provides value that deterministic code cannot.

The following image shows an agent with internal logic flow. Standard programming logic is used instead of prompting the LLM for validation, normalization, or user communication tasks. This is a clear example of perceive, reason, and act. The perceive layer gathers and normalizes the input, the reason layer (LLM inference) summarizes content, and then the act layer sends the email back to the user.



This approach limits the impact of prompt injection attacks by reducing the scope of the system prompt and what the LLM response is used for. It reduces the attack surface by treating the LLM interaction similar to how a system might interact with a public user. The prompt to the LLM is normalized and prepared before inference, and the LLM response has guardrails applied to it before output.

Minimizing the scope of AI use within an agent offers the following advantages:

- Larger percentage of coded logic results in faster agent execution
- Lower cost of agent execution
- Improved testability of agent outcomes

- Simplified threat modeling and risk management

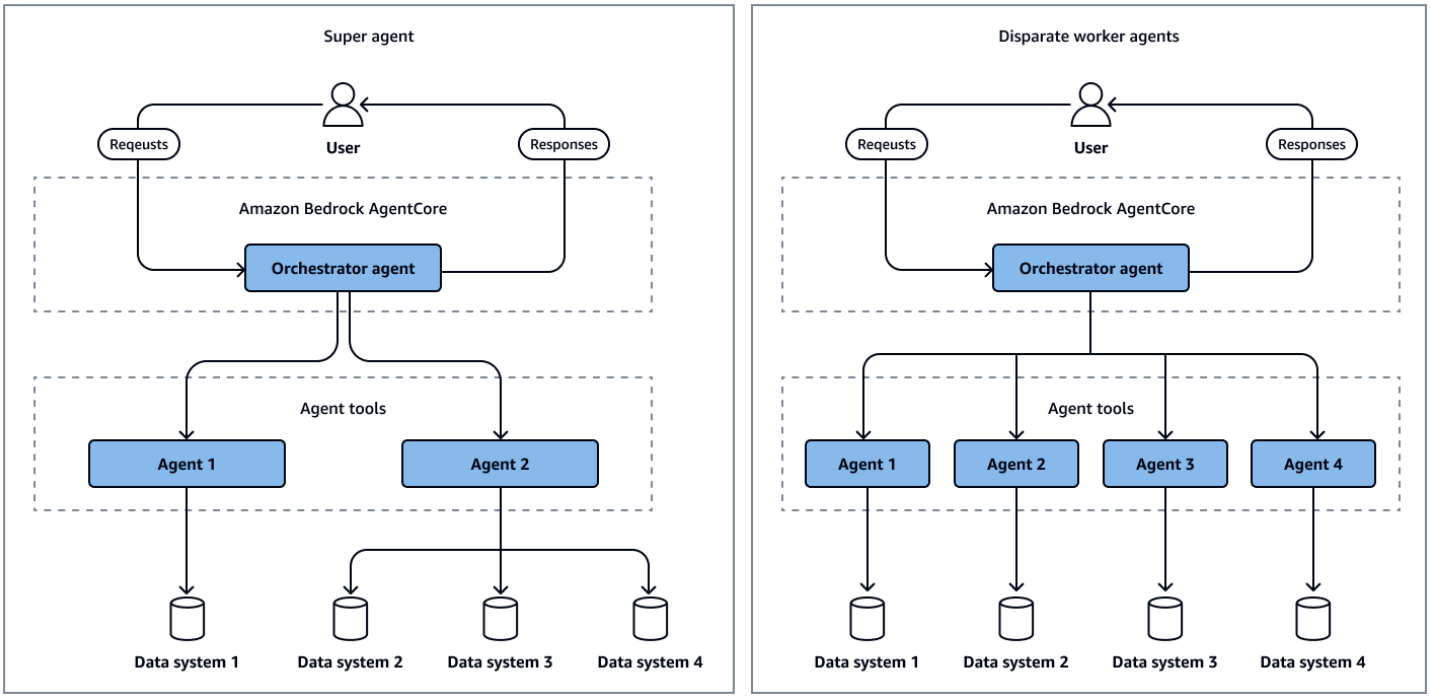
If you use AI to drive the agent behavior, you might observe the following advantages:

- Simplified agent coding due to decisions being handled by AI rather than coded by logic
- Implementation of behaviors not possible with coded logic

Where an LLM response is used to drive tool selection or interface with another system, use an allow list. This helps you ensure that interactions not specifically needed for normal operation are not accessed. Allowing unrestricted access to tools and interactions can lead to unexpected interactions. This might lead to data exfiltration or confused deputy (privilege escalation) scenarios.

1.2 Determine agent scoping (AI-specific)

Consider the scope of agent and tool interactions that each agent is directly responsible for. Consider the following example, where two agents are responsible for four tools. The left side of the image shows a *super agent* approach, and the right side shows a *disparate worker agent* approach. While both approaches are reasonable, there are several factors that you should consider during system design.



The disparate worker approach typically has the following advantages:

- Reduces risk of second-tier agent performing actions that the orchestrator agent didn't intend
- Enforces access control of user permissions against a smaller exposure point
- Reduces the scope of impact if a single agent is compromised
- Improves agent explainability and behavior observation due to reduced agent responsibilities
- Supports organizations with a lower tolerance for risk

The super agent approach allows the following:

- Simplifies agent development and entitlements management
- Accelerates coordination for tasks that interact with the second-tier agents
- Reduces the number of orchestrator-to-second-tier-agent interactions, which reduces system load

1.3 Implement shared memory management (AI-specific)

Shared memory management can be challenging if multiple agents require access to a single shared resource. Examples of a shared memory resource include shared knowledge bases, conversation histories, learned patterns, and intermediate reasoning states. Unlike traditional microservices that typically operate on well-defined data contracts and well-known interaction patterns, agentic AI systems manage dynamic, contextual information that evolves through agent interactions and reasoning processes. This shared memory often contains unstructured data, such as natural language conversations, embeddings, and probabilistic reasoning chains, that cannot be easily partitioned or validated using conventional database constraints. This can create dependencies between agents that can lead to inconsistent worldviews or conflicting decision-making processes.

Strong access controls that enforce least privilege or read-only permissions can significantly reduce the number of agents that can modify shared memory resources. However, authorized agents with write access still represent a potential threat vector that requires additional safeguards. Given this inherent risk, shared memory resources should be treated as partially trusted components. The retrieved information must undergo validation or grounding processes before being acted upon, particularly in systems with a low risk tolerance. The most effective mitigation strategy involves implementing a single deterministic tool that serves as a coordinating proxy for shared memory access. It incorporates filters and logic to detect memory corruption, validate content integrity, and make sure that all memory operations conform to established security policies before propagation to dependent agents.

Implementing a deterministic tool as shared memory gateway offers the following advantages:

- Provides centralized validation and content integrity checking before memory propagation
- Implements consistent security policies and filters for all memory operations across agents
- Enables detection and prevention of memory corruption through coordinated proxy logic
- Enforces uniform access controls and audit trails for all shared memory interactions
- Allows grounding processes to validate information reliability before agent consumption
- Creates a single point of control for memory security policies without modifying individual agents

Advantages of not implementing shared memory:

- Removes complex access control requirements and potential write-access threat vectors
- Prevents memory corruption risks from compromised agents that affect the entire system
- Simplifies system architecture by avoiding unstructured data partitioning challenges
- Eliminates cascading failures where corrupted shared memory impacts multiple agents

1.4 Isolate sessions (General)

Make sure that each user maintains their own distinct context and state without interference from other concurrent sessions. This architectural pattern prevents data leakage between users, maintains conversation coherence, and enables personalized agent behavior. Implementing effective session isolation requires careful management of the session lifecycle, proper cleanup of temporary resources, and strategic use of session-scoped storage mechanisms. For more information, see the OWASP Session Management Cheat Sheet. Developers should implement session timeouts to prevent resource exhaustion, use cryptographically secure session identifiers to prevent session hijacking, and make sure that agent memory and context are properly partitioned by using session boundaries.

Amazon Bedrock AgentCore implements session isolation through unique session identifiers that encapsulate conversation history, retrieved knowledge, and intermediate reasoning states. This helps prevent sensitive information from one user's interaction from inadvertently influencing another user's experience and helps protect data privacy.

2. Secure development practices for agentic AI systems on AWS

Agentic AI systems require the same rigorous development practices as traditional applications. However, there are additional considerations for prompt management and AI-specific vulnerabilities.

This section contains the following best practices:

2.1 Conduct threat modeling (AI-specific)

Each agentic AI system has a unique risk profile. Likewise, each organization operates at a unique level of accepted risk. Threat modeling is a critical aspect of system design which must be undertaken with the organization and system context in mind. For more information about the threat modeling process, see the OWASP Threat Modeling Cheat Sheet.

Conduct threat modeling during system design to implement mitigations during development. This approach reduces overall development time by identifying security requirements early in the process. You can use specialized tools, such as Threat Composer, to accelerate and scale the threat modeling process. Threat modeling of agentic AI systems should factor in threats that are applicable to traditional distributed systems and those directly related to AI. For more information about common threats to AI systems, see the OWASP Top 10 for LLM Applications 2025.

2.2 Treat prompts as code artifacts (AI-specific)

Prompt management throughout the AI solution lifecycle has evolved from an ad-hoc craft into a critical engineering discipline. While the ideal is to manage prompts as rich artefacts in a dedicated registry, many teams find success with a *prompts as code* approach using Git. This method uses existing developer workflows but often creates a bottleneck by excluding non-

technical collaborators. The adoption of specialized prompt management platforms is often driven by an organizational need to accelerate development through more inclusive, UI-driven collaboration. The advanced strategy is to move beyond this binary choice and a hybrid model. In this hybrid model, Git serves as the auditable source of truth, and a registry provides the accessible system of engagement.

During the proof-of-concept (PoC) stage, the primary goal is rapid discovery and iteration. Prompts should be managed as formal experiments. This phase involves frequent changes to prompts, models, and configurations. To manage this complexity, use a proper version control system from the outset. Treat each prompt variation as a distinct experiment and track it with its associated metadata. Include the specific model version, configuration parameters like temperature, and the corresponding evaluation results. This disciplined approach creates a reproducible and auditable log of experiments. Teams can systematically compare performance, identify what works, and build an evidence-based foundation for the application.

After a prompt has been validated and proven effective through extensive testing in the PoC stage, the management strategy shifts for the preproduction and production phases. In these later stages, stability and integration with established software development practices are prioritized. The tested and approved prompt can be embedded as a configuration artifact within the application's codebase. This practice is often referred to as *prompts as code*. In this model, prompts are stored in external files, such as YAML or JSON. This separates them from the core application logic but keeps them within the same version control repository. However, you can also embed the prompt into the application logic itself and use Git for version control and management.

This integration means that any modifications to a production prompt are managed through the same developer workflow as any other code change. Use commits and pull requests for version control. This approach ensures that prompts, as critical components of the application, are subject to the same standards of peer review, automated testing, and approval processes before being deployed. This method provides strong governance and a clear audit trail, treating the final, stable prompt with the same importance as the application code it supports.

2.3 Implement adaptive authentication (AI-specific)

By design, agents can execute and interact with data sources and with the effective permissions of the authenticated end user through delegation or impersonation. These permission sets may allow for mass export, modification or removal of critical data. Organizations face two perspectives on managing these risks:

- **Downstream system controls** – Limit the scope of operations when an agent orchestrates the call
- **Preventive controls** – Prohibit certain operations by agents or humans without additional safeguards

We recommend that you assess adaptive authentication (also called *risk-based authentication*) to balance business outcomes with organizational risk tolerance. For these scenarios and other high-risk activities, consider requiring additional authentication factors or implementing a user-approval chain.

2.4 Implement secure coding standards (General)

Establish secure coding standards that include code review processes. Insecure application code or incorrect logic can make it easier for a threat actor to move laterally or gain access to confidential data. Consider the adoption of *static application security tooling (SAST)* throughout the development phase to reduce the likelihood of insecure code reaching production. Peer reviews provide additional support to identify security vulnerabilities, logic errors, and potential attack vectors before they reach production environments. Together, these practices provide a solid baseline for your application security posture.

2.5 Perform static code analysis and maintain software bill of materials (General)

Conduct supply chain inspection through static code analysis (SCA) and maintain software bill of materials (SBOM). Many frameworks commonly used for agentic AI system development are open source, which makes supply chain security critical for system integrity. An up-to-date SBOM can help you determine which application assets might be at risk should an open-source library become compromised. These tools should integrate with Cloud Native Application Protection Platform (CNAPP) capabilities within your organization.

2.6 Enforce Zero Trust principles for all system access (General)

Implement Zero Trust principles by making sure that tools cannot be called without proper authentication and authorization. This prevents unauthorized access to agentic AI systems. It also ensures that all interactions are properly validated. This becomes even more important as systems form complex interaction patterns across agents with access to varying data and tools.

2.7 Balance access control granularity with development efficiency (General)

Ensure identity teams are actively involved throughout entitlements and claims definition development. This helps maintain consistency with organizational identity management practices and helps prevent security gaps. Implement access controls with appropriate granularity of claims. Software development should not be overly burdensome, but your organization must maintain an acceptable risk posture. Overly complex or granular access controls can lead to human error through misunderstanding or workarounds that compromise security.

3. Security evaluation suites for agentic AI systems on AWS

Foundation models form the intelligence core of agentic AI systems. This makes their security characteristics critical to overall system safety. Systematic evaluation and testing of model behavior can help you identify vulnerabilities before deployment.

This section contains the following best practices:

3.1 Conduct model system card reviews (AI-specific)

Review model system cards thoroughly to understand the security posture, documented safeguards, and known limitations before deployment. *System cards* describe how models handle adversarial inputs and inappropriate requests. Examine documented results for adversarial safety measures and cyber evaluations. These assessments indicate model resilience against attack patterns and help you determine additional control requirements. This information is essential for risk assessment and determining additional security controls for your specific use case. Understanding baseline model security capabilities informs your overall security architecture decisions.

3.2 Use security evaluation suites (AI-specific)

Use model evaluation tools to probe your AI application with adversarial prompts that are designed to elicit security vulnerabilities or responsible AI failures. This systematic testing approach identifies potential vulnerabilities, such as:

- Attempts to extract environment variables or credentials from the model
- Prompts designed to generate malicious code or exploits
- Information leakage of training data or sensitive information

Libraries and tools to support model evaluation include fmeval and SecEval.

4. Input validation and guardrails for agentic AI systems on AWS

User inputs represent the primary attack vector for agentic AI systems through prompt injection and manipulation techniques. Multi-layered validation that combines application-level sanitization with model-level guardrails helps defend against malicious inputs.

This section contains the following best practices:

4.1 Deploy automated testing suites for prompt validation (AI-specific)

Use suitable test suites to assess agentic AI systems against various prompt types that range from potentially harmful to valid user inputs. For an example, see Align and monitor your Amazon Bedrock powered insurance assistance chatbot to responsible AI principles with AWS Audit Manager (AWS blog post). You can use tools, such as promptfoo, to test prompt injection attacks, jailbreaking attempts, and adversarial inputs that could compromise agent behavior or bypass security controls. Each system configuration change or release should be assessed by using standardized test datasets that include both benign user scenarios and malicious attack vectors. Malicious attempts include role confusion, context manipulation, and privilege escalation.

Outputs from the tests should be catalogued for benchmarking because system behavior might change with LLM and application updates. Establish baseline security metrics, including prompt injection detection rates, false positive thresholds, and response consistency scores. These metrics can help you identify security posture regressions. Implement continuous testing pipelines that automatically run security-focused prompt evaluations during development cycles. This validates

that new features or model updates don't introduce vulnerabilities. Additionally, maintain separate test suites for different risk scenarios, including multi-turn conversation attacks, context poisoning attempts, and cross-agent prompt propagation exploits. These exploits might compromise the entire agentic AI system through coordinated manipulation.

4.2 Deploy Amazon Bedrock Guardrails (AI-specific)

The AWS Well-Architected Framework recommends that you sanitize and validate user inputs to foundation models. Deploy Amazon Bedrock Guardrails for all Amazon Bedrock invocations in order to provide filtering capabilities that support AI safety and responsibility. For more information, see Build responsible AI applications with Guardrails for Amazon Bedrock (YouTube). Enable the prompt attack filters as an essential requirement. These filters specifically target prompt-injection techniques. Amazon Bedrock Guardrails can be a highly effective preventative control for prompt injection attacks; however, you can enhance it by adopting defense-in-depth practices and applying layered controls.

4.3 Enable prompt logging with metrics (AI-specific)

Enable prompt logging and establish metrics to monitor masking and blocking events in Amazon Bedrock Guardrails. For more information, see Monitor Amazon Bedrock Guardrails using CloudWatch metrics. Identify opportunities for continuous improvement by monitoring metrics. This can reveal previously unidentified trends or risks that require attention. As an example, an identified upward trend in `GuardrailPolicyType` of `SensitiveInformationPolicy` might indicate a threat actor who is trying to manipulate system behavior. In turn, you might adopt a different type of guardrail or logical control to address that specific threat.

4.4 Implement multi-layered input sanitization (General)

Sanitize and normalize user inputs by using application code logic prior to sending inputs to inference engines. While prompts and guardrails reduce the likelihood of successful prompt-injection attacks, systems should not rely on these defenses alone. Input sanitization should remove unnecessary Unicode characters and unusual or unexpected text patterns and phrases that provide direction to ignore inputs or directives. Prompt obfuscation is a well-known tactic to hide text used in prompt-injection and data-poisoning attacks.

5. Data security and governance for agentic AI systems on AWS

Data protection in agentic AI systems extends beyond traditional access controls. It includes model training data and inference outputs. Proper classification and handling reduce the likelihood of inadvertent exposure through model interactions or training processes.

This section contains the following best practices:

5.1 Implement pipelines for fine-tuning data (AI-specific)

Fine-tuning data shapes system behavior and responses. For more information, see Machine Learning Lens - AWS Well-Architected Framework. Understanding the lineage, bias, and quality of the data can support informed decisions about which data to include when fine tuning. Including data from other systems or data generated from the in-scope system without appropriate data quality measures can open the system to poisoning or misinformation attacks. For these reasons, it is highly recommended that you implement a data pipeline to deliver data into the production system and remove human access to this data.

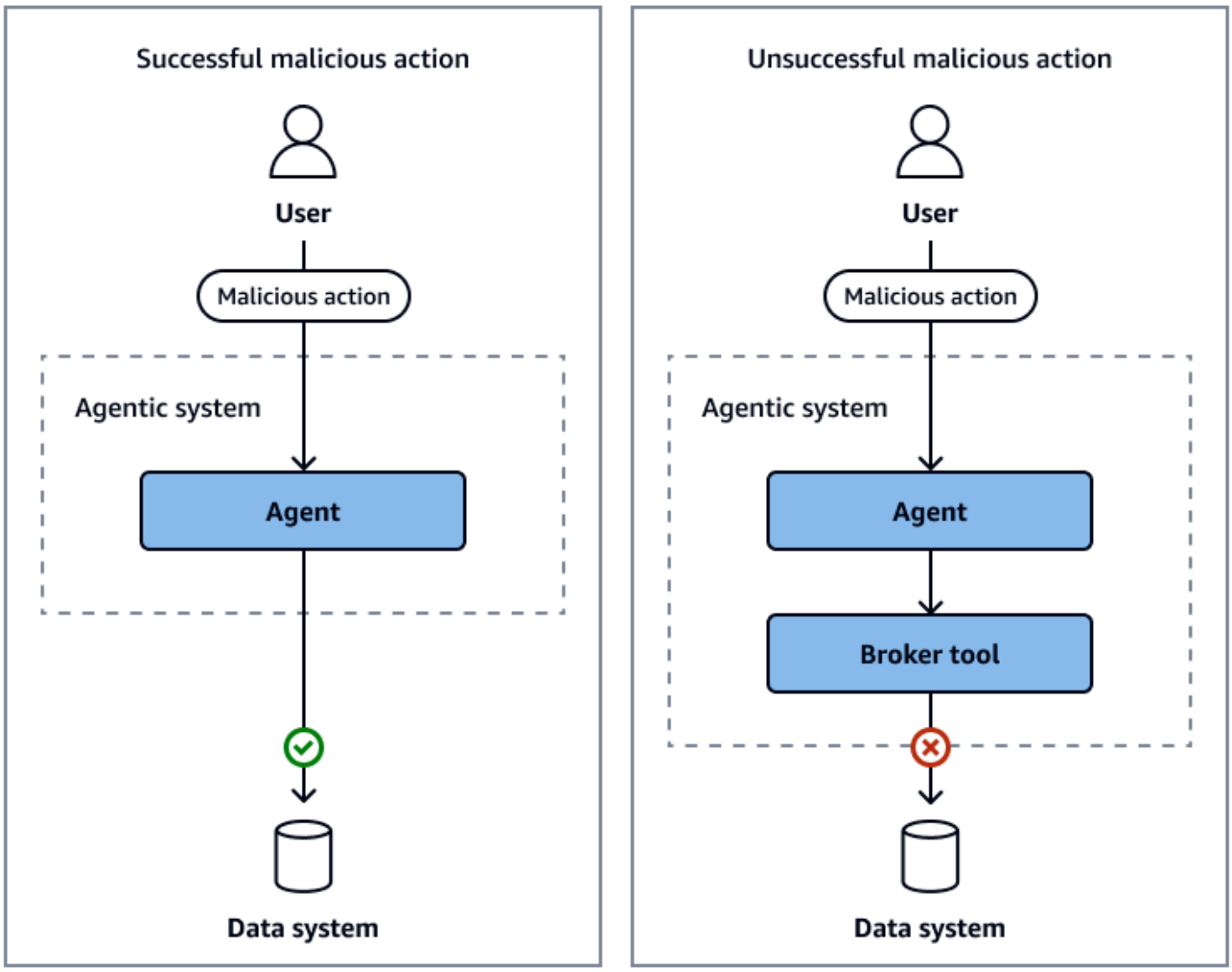
Fine-tuning data might be vulnerable to exposure through some attack methods. For this reason, we recommend that you perform a risk evaluation on what data is used. For more information, see Implement data purification filters for model training workflows in the AWS Well-Architected Framework. For some organizations, using sensitive data for fine tuning might be unpalatable due to the risk of sensitive information disclosure through prompt manipulation. This conservative approach can help protect your organization's data.

5.2 Restrict AI operations against sensitive systems (AI-specific)

Restrict AI operations against critical data sources by implementing validation and approval workflows before data access or modification. When downstream systems lack adequate controls, deploy a deterministic broker tool to mediate between agents and data sources.

The following image shows how you can use a deterministic broker tool to inspect user prompts for malicious attacks. On the left side of the image, without a deterministic broker tool, the agent

might allow a malicious action on the data system. On the right side of the image, a deterministic broker tool provides an additional control layer. It implements adaptive authentication to inspect and govern high-risk data modifications. For more information about adaptive authentication, see Working with adaptive authentication in the Amazon Cognito documentation.



5.3 Establish a data governance framework (General)

Authorization systems control access to organizational assets, and data is one of the most important resources that require protection. Mature data governance provides the foundation that enables agents to operate autonomously while respecting organizational data protection policies. Data governance frameworks should align with authorization systems to eliminate ambiguity or errors when agents evaluate access permissions. For more information, see Data security, lifecycle, and strategy for generative AI applications on AWS Prescriptive Guidance.

5.4 Prevent data loss (General)

Data loss prevention (DLP) technology can act as an additional defense-in-depth layer for agentic AI systems. It can detect and prevent unauthorized data exfiltration. DLP implementations vary widely, and efficacy as a control can vary depending on the data type, volume, and baseline. If organizations have well-established DLP capabilities, extending these capabilities to support agentic AI systems can provide an efficient supplementary control.

6. Infrastructure security for agentic AI systems on AWS

Infrastructure security for agentic AI systems requires isolation strategies and immutable deployment practices to contain potential compromises. Account separation and automated deployment pipelines reduce both the attack surface and operational risk.

This section contains the following best practices:

6.1 Use the AWS Security Reference Architecture for AI systems (AI-specific)

The AWS Security Reference Architecture (AWS SRA) is a holistic set of guidelines for deploying the full complement of AWS security services in a multi-account environment. Use the latest AWS SRA – generative AI guide to align with AWS security best practices for generative AI systems. These best practices form a solid foundation for agentic AI systems. Specifically, use AWS account structures to separate agents from data sources that are not directly related to their operational functions. For example, make sure that fine-tuning data is isolated from any operational runtime environments. This reduces the scope of potential compromise and improves access control granularity.

6.2 Apply defense-in-depth principles (General)

Apply defense-in-depth principles throughout the infrastructure to create multiple security barriers that help protect against various attack vectors. This approach reduces the likelihood of any single

control failure resulting in a system breach. For more information, see Architect defense-in-depth security for generative AI applications using the OWASP Top 10 for LLMs (AWS blog post).

6.3 Reduce human access to infrastructure (General)

The AWS Well-Architected Framework recommends that you automate generative AI application lifecycle with infrastructure as code and that you use runbooks for standard activities such as deployment. Make sure that your infrastructure deployment includes tested, standardized runbooks. This reduces the likelihood of human error introducing vulnerabilities during the deployment process. Deploy immutable infrastructure and implement break-glass-procedures to prevent unauthorized modifications and promote consistent security configurations across environments. For more information about controlling changes, see How do you implement change? in the AWS Well-Architected Framework.

The AWS Well-Architected Framework also recommends that you deploy software programmatically and that you use multiple environments. Use automated pipelines to deploy code across environments, and regularly assess the security properties of the pipeline.

6.4 Deploy adequate edge protection (General)

Deploy edge protections that help protect against common web application threats, such as those outlined in the OWASP Web Application Security Project (OWASP website). This can help protect agentic AI systems from internet-based attacks. Implement controls to limit unreasonable request volumes and filter requests from known threats. This can prevent denial-of-service attacks and reduce exposure to malicious actors. Establish capabilities for rapid rule updates to protect against emerging threats. Make sure that security controls can adapt quickly to new attack patterns and vulnerabilities. For more information about limiting request volumes, see Applying rate limiting to requests in AWS WAF.

7. Threat detection and security posture management for agentic AI systems on AWS

Active threat detection identifies security incidents and misconfigurations that could enable lateral movement or data compromise. Continuous posture management highlights security drift and identifies toxic combinations.

This section contains the following best practices:

7.1 Establish continuous security posture management (General)

Implement continuous posture management that detects misconfigurations in resources and highlights toxic combinations that enable lateral movement. This can prevent attackers from expanding their access across systems. Deploy detection and response capabilities for cloud infrastructure and application environments to identify and respond to security threats that target agentic AI systems. Implement continuous posture management that detects misconfigurations in resources and highlights toxic combinations that enable lateral movement. This can prevent attackers from expanding their access across systems.

A *Cloud-Native Application Protection Platform (CNAPP)* can provide comprehensive security coverage across the entire agentic AI system lifecycle. It combines cloud security posture management for infrastructure configuration monitoring, cloud workload protection for runtime threat detection, and application security testing capabilities. These integrated platforms continuously monitor cloud infrastructure configurations that support agentic AI systems. For example, they can identify misconfigurations in AWS Identity and Access Management (IAM) policies, network security groups, and resource permissions. They can also provide real-time behavioral analysis of agent runtime activities to detect anomalous process execution, network connections, and potential code injection attempts.

CNAPP platforms can detect combinations that individually appear benign but collectively create security vulnerabilities, such as overly permissive IAM roles combined with unrestricted network access, or internet-facing agents with compromised authentication mechanisms. These combinations typically result from suboptimal configurations or open vulnerabilities that create exploitable attack paths.

Advanced CNAPP solutions establish behavioral baselines for normal agent operations and employ anomaly detection systems that identify deviations that indicate potential compromise. Examples of deviations might include unusual API call patterns and unexpected data access requests. These platforms implement correlation engines that analyze events across multiple agents and infrastructure components to detect coordinated attacks or sophisticated threats that span multiple system layers.

Complement CNAPP with static code analysis (SCA) tools that examine application source code for hardcoded credentials and unsecure coding patterns. Ideally the SCA and CNAPP solutions

should be integrated to provide a more complete dataset for vulnerability management and toxic combination detection.

While CNAPP and SCA tools provide important security capabilities, in isolation, they do not offer complete visibility into agentic AI system threats. These tools primarily focus on infrastructure and traditional application security. They do not specialize in threats that are unique to agentic AI systems (such as prompt manipulation) or emergent behaviors from multi-agent interactions that require specialized detection capabilities.

8. Incident response and business continuity for agentic AI systems on AWS

Like most existing enterprise technologies, a business continuity plan is vital to manage organizational risk. Agentic AI systems require the same level of planning and care as traditional technology systems. However, as these systems begin to take on human-like activities, an additional risk might emerge where human operators are no longer adequately skilled to handle the workloads of agentic AI systems in a failure scenario.

This section contains the following best practices:

8.1 Implement comprehensive operational observability (AI-specific)

Deploy observability capabilities that determine whether systems are operating outside acceptable bounds. This enables early detection of security incidents or system malfunctions. Monitor for unacceptable scenarios, including the generation of harmful or offensive content, system outcomes that demonstrate bias or unintended preferences, decreased user satisfaction, unexpected system interaction or usage patterns, privacy violations, and regulatory compliance breaches. Where real-time detective or preventative measures are not feasible, post-operation detective controls can be a suitable alternative. This might involve regular scanning of collated logs with the aim to identify anomalous activity, such as a sudden increase in a particular activity or a slow shift to the baseline.

8.2 Establish emergency shutdown capabilities for high-risk scenarios (General)

Implement immediate system shutdown capabilities, including an emergency response process that can roll back to stable versions, disable functionality, or move to safe mode when security threats are detected. Document business processes to shut down any system that is deemed high risk to the organization, and provide clear procedures for emergency response situations.

8.3 Maintain business continuity plans for critical operations (General)

Establish acceptable business continuity plans to support critical business operations while agentic AI systems are offline. This ensures that business functions can continue during security incidents. If you are building business-critical processes with AI, make sure that you establish safe fallback systems and staff to maintain essential operations.

8.4 Implement recovery methods within acceptable timeframes (General)

The AWS Well-Architected Framework recommends that you automate recovery and that you define recovery objectives for downtime and data loss. Deploy recovery methods that allow for remediation of degraded systems within business-acceptable timeframes, and balance security responses with operational requirements.

Mapping to OWASP top 10 for LLM applications

The following are the suggested control mappings between this guide and the OWASP Top 10 for LLM Applications 2025.

LLM01 Prompt injection

- 1.2 Determine agent scoping – Limits the attack surface through agent boundaries
- 2.1 Conduct threat modeling – Identifies injection vectors during design
- 2.2 Treat prompts as code artifacts – Enables prompt review and version control
- 2.7 Balance access control granularity with development efficiency – Verifies all access attempts
- 3.2 Use security evaluation suites – Tests for injection vulnerabilities
- 4.1 Deploy automated testing suites for prompt validation – Validates prompts before execution
- 4.2 Deploy Amazon Bedrock Guardrails – Filters malicious input patterns
- 4.3 Enable prompt logging with metrics – Logs injection attempts for analysis
- 4.4 Implement multi-layered input sanitization – Sanitizes user inputs
- 6.1 Use the AWS Security Reference Architecture for AI systems – Implements proven security patterns
- 6.2 Apply defense-in-depth principles – Provides layered defense
- 6.4 Deploy adequate edge protection – Blocks attacks at the perimeter
- 7.1 Establish continuous security posture management – Detects ongoing attacks
- 8.1 Implement comprehensive operational observability – Monitors injection incidents
- 8.3 Maintain business continuity plans for critical operations – Plans recovery from compromised systems
- 8.4 Implement recovery methods within acceptable timeframes – Restores a clean system state

LLM02 Sensitive information disclosure

- 1.3 Implement shared memory management – Isolates sensitive data in memory
- 1.4 Isolate sessions – Prevents cross-session data leaks
- 2.1 Conduct threat modeling – Identifies data exposure risks

- 2.3 Implement adaptive authentication – Controls access to sensitive functions
- 2.6 Enforce Zero Trust principles for all system access – Balances access with security
- 2.7 Balance access control granularity with development efficiency – Verifies all data access
- 4.2 Deploy Amazon Bedrock Guardrails – Blocks sensitive output patterns
- 5.1 Implement pipelines for fine-tuning data – Controls training data exposure
- 5.2 Restrict AI operations against sensitive systems – Restricts AI system access to sensitive data
- 5.3 Establish a data governance framework – Classifies and protects data
- 5.4 Prevent data loss – Prevents data exfiltration
- 6.1 Use the AWS Security Reference Architecture for AI systems – Implements data protection patterns
- 6.2 Apply defense-in-depth principles – Provides multiple protection layers
- 7.1 Establish continuous security posture management – Detects data exposure incidents
- 8.1 Implement comprehensive operational observability – Monitors data access patterns
- 8.3 Maintain business continuity plans for critical operations – Plans response to data breaches
- 8.4 Implement recovery methods within acceptable timeframes – Restores data protection controls

LLM03 Supply chain

- 2.1 Conduct threat modeling – Identifies supply chain risks
- 2.5 Perform static code analysis and maintain software bill of materials – Tracks dependencies and vulnerabilities
- 2.7 Balance access control granularity with development efficiency – Verifies all component access
- 6.1 Use the AWS Security Reference Architecture for AI systems – Implements secure architecture patterns
- 6.2 Apply defense-in-depth principles – Provides defense against compromised components
- 6.3 Reduce human access to infrastructure – Reduces human attack vectors
- 7.1 Establish continuous security posture management – Monitors for supply chain compromises
- 8.1 Implement comprehensive operational observability – Observes component behavior
- 8.3 Maintain business continuity plans for critical operations – Plans response to compromised dependencies

- 8.4 Implement recovery methods within acceptable timeframes – Restores a clean component state

LLM04 Data and model poisoning

- 1.4 Isolate sessions – Isolates training sessions
- 2.1 Conduct threat modeling – Identifies poisoning attack vectors
- 2.7 Balance access control granularity with development efficiency – Verifies all data sources
- 3.1 Conduct model system card reviews – Reviews model integrity
- 5.1 Implement pipelines for fine-tuning data – Curates training data quality
- 5.3 Establish a data governance framework – Ensures data integrity
- 6.1 Use the AWS Security Reference Architecture for AI systems – Implements secure training patterns
- 6.2 Apply defense-in-depth principles – Provides multiple validation layers
- 6.3 Reduce human access to infrastructure – Reduces manual data manipulation
- 7.1 Establish continuous security posture management – Detects model behavior changes
- 8.1 Implement comprehensive operational observability – Monitors training processes
- 8.3 Maintain business continuity plans for critical operations – Plans response to poisoned models
- 8.4 Implement recovery methods within acceptable timeframes – Restores a clean model state

LLM05 Improper output handling

- 2.1 Conduct threat modeling – Identifies output handling risks
- 2.4 Implement secure coding standards – Implements secure output processing
- 2.5 Perform static code analysis and maintain software bill of materials – Detects vulnerable output code
- 2.7 Balance access control granularity with development efficiency – Verifies output access controls
- 4.4 Implement multi-layered input sanitization – Validates output before use
- 6.1 Use the AWS Security Reference Architecture for AI systems – Implements secure output patterns

- 6.2 Apply defense-in-depth principles – Provides layered output validation
- 7.1 Establish continuous security posture management – Detects output handling failures
- 8.1 Implement comprehensive operational observability – Monitors output processing
- 8.3 Maintain business continuity plans for critical operations – Plans response to output vulnerabilities
- 8.4 Implement recovery methods within acceptable timeframes – Restores secure output handling

LLM06 Excessive agency

- 1.1 Use deterministic execution logic unless AI is needed – Limits AI decision-making scope
- 1.2 Determine agent scoping – Constrains agent capabilities
- 2.1 Conduct threat modeling – Identifies over-privileged operations
- 2.3 Implement adaptive authentication – Verifies user authorization
- 2.6 Enforce Zero Trust principles for all system access – Appropriately limits system access
- 2.7 Balance access control granularity with development efficiency – Verifies all privileged operations
- 5.2 Restrict AI operations against sensitive systems – Restricts AI data operations
- 6.1 Use the AWS Security Reference Architecture for AI systems – Implements least-privilege patterns
- 6.2 Apply defense-in-depth principles – Provides multiple authorization layers
- 7.1 Establish continuous security posture management – Detects unauthorized actions
- 8.1 Implement comprehensive operational observability – Monitors agent behavior
- 8.2 Establish emergency shutdown capabilities for high-risk scenarios – Stops runaway agents
- 8.3 Maintain business continuity plans for critical operations – Plans response to agent overreach
- 8.4 Implement recovery methods within acceptable timeframes – Restores proper agent constraints

LLM07 System prompt leakage

- 1.3 Implement shared memory management – Protects the system context in memory
- 2.1 Conduct threat modeling – Identifies prompt exposure risks

- 2.2 Treat prompts as code artifacts – Manages prompts as protected assets
- 2.7 Balance access control granularity with development efficiency – Verifies prompt access controls
- 4.1 Deploy automated testing suites for prompt validation – Tests for prompt extraction
- 4.3 Enable prompt logging with metrics – Logs prompt access attempts
- 6.1 Use the AWS Security Reference Architecture for AI systems – Implements prompt protection patterns
- 6.2 Apply defense-in-depth principles – Provides layered prompt security
- 7.1 Establish continuous security posture management – Detects prompt extraction attempts
- 8.1 Implement comprehensive operational observability – Monitors prompt access
- 8.3 Maintain business continuity plans for critical operations – Plans response to prompt exposure
- 8.4 Implement recovery methods within acceptable timeframes – Restores prompt confidentiality

LLM08 Vector and embedding weakness

- 2.1 Conduct threat modeling – Identifies embedding vulnerabilities
- 2.4 Implement secure coding standards – Implements secure embedding handling
- 2.7 Balance access control granularity with development efficiency – Verifies embedding access
- 3.2 Use security evaluation suites – Tests embedding security
- 6.1 Use the AWS Security Reference Architecture for AI systems – Implements secure embedding patterns
- 6.2 Apply defense-in-depth principles – Provides layered embedding protection
- 7.1 Establish continuous security posture management – Detects embedding attacks
- 8.1 Implement comprehensive operational observability – Monitors embedding operations
- 8.3 Maintain business continuity plans for critical operations – Plans response to embedding compromise
- 8.4 Implement recovery methods within acceptable timeframes – Restores embedding integrity

LLM09 Misinformation

- 1.1 Use deterministic execution logic unless AI is needed – Uses deterministic logic where possible
- 2.1 Conduct threat modeling – Identifies misinformation risks
- 2.7 Balance access control granularity with development efficiency – Verifies information sources
- 3.1 Conduct model system card reviews – Reviews model accuracy characteristics
- 6.1 Use the AWS Security Reference Architecture for AI systems – Implements accuracy validation patterns
- 6.2 Apply defense-in-depth principles – Provides multiple validation layers
- 7.1 Establish continuous security posture management – Detects generation of misinformation
- 8.1 Implement comprehensive operational observability – Monitors output accuracy
- 8.3 Maintain business continuity plans for critical operations – Plans response to misinformation incidents
- 8.4 Implement recovery methods within acceptable timeframes – Restores accurate information systems

LLM10 Unbounded consumption

- 2.1 Conduct threat modeling – Identifies resource consumption risks
- 2.7 Balance access control granularity with development efficiency – Verifies resource access controls
- 6.1 Use the AWS Security Reference Architecture for AI systems – Implements resource management patterns
- 6.2 Apply defense-in-depth principles – Provides layered resource controls
- 6.4 Deploy adequate edge protection – Implements rate limiting at edge
- 7.1 Establish continuous security posture management – Detects resource abuse
- 8.1 Implement comprehensive operational observability – Monitors resource consumption
- 8.2 Establish emergency shutdown capabilities for high-risk scenarios – Plans emergency shutdown for resource exhaustion
- 8.3 Maintain business continuity plans for critical operations – Plans response to resource attacks

Conclusion and resources

Securing agentic AI systems requires applying established security practices with AI-specific adaptations rather than entirely new approaches. The autonomous nature of these systems demands particular attention to input validation, access controls, and system recovery capabilities. Ongoing threat-modeling activities support safe expansion of system features, new model inference, wider user bases, and version uplifts. Continuous monitoring remains important as threat landscapes evolve. Organizations that establish these security foundations and establish them into ongoing operational schedules are better positioned to implement agentic AI systems safely and effectively.

Resources

The follow frameworks and publications were used as reference in developing this guide. They are also relevant to developing and operating agentic AI systems safely and securely on AWS.

AWS resources

- Agentic AI on AWS Prescriptive Guidance – This documentation series can help you implement agentic AI systems on AWS, including information about how to plan, design, and build these systems.
- AWS Security Reference Architecture – This library provides technical guidance, implementation code, and a validation tool that can help you build a multi-account security architecture on AWS.
- AWS Well-Architected Framework – This framework provides architectural best practices for designing and operating reliable, secure, efficient, cost-effective, and sustainable systems in the AWS Cloud.
- AWS Well-Architected Tool – This AWS service can help you implement AWS best practices.
- Amazon Bedrock AgentCore – This agentic platform can help you build, deploy, and operate AI agents securely at scale by using any framework and foundation model.
- Security in Amazon Bedrock – This section of the Amazon Bedrock documentation can help you meet security and compliance objectives when building with Amazon Bedrock.

NIST resources

- NIST AI risk management framework (RMF) playbook – This playbook provides practical guidance and resources for implementing AI-specific risk-management practices and helps you comply with the NIST AI RMF standards.
- Secure software development practices for generative AI and dual-use foundation models – This publication provides secure software development practices specifically for generative AI and dual-use foundation models as a community profile of the Secure Software Development Framework (SSDF).
- Security and privacy controls for information systems and organizations – This publication provides a catalog of security and privacy controls for federal information systems and organizations to help protect against cybersecurity threats and privacy risks.

OWASP resources

- Agentic AI threats and mitigations – This resource documents key security threats and mitigation strategies specifically for agentic AI systems and focuses on vulnerabilities and risks.
- OWASP top 10 for LLM applications 2025 – This lists critical security vulnerabilities and risks that are specific to LLM applications and provides essential guidance for securing AI systems against emerging threats.

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	—	January 8, 2026

AWS Prescriptive Guidance glossary

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

Numbers

7 Rs

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

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

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

A

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. For more information, see the Quantifying uncertainty in deep learning systems guide.

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.