Generative AI inference architecture and best practices on AWS

AWS Prescriptive Guidance

AWS Prescriptive Guidance: Generative AI inference architecture and best practices on AWS

Table of Contents

Generative AI inference architecture and best practices on AWS

Amazon Web Services (???)

December 2025 (document history)

Organizations that deploy generative AI (gen AI) models such as large language models (LLMs) and multimodal generation models face critical challenges in serving production workloads at scale. These challenges span multiple dimensions including how to:

- Maintain consistent low-latency responses for real-time applications and user experiences.
- Implement dynamic capacity scaling to handle unpredictable traffic patterns.
- Optimize infrastructure costs.
- Ensure high availability across deployments.

This guide provides comprehensive prescriptive guidance on selecting and implementing appropriate AWS inference services, designing resilient architectures, and applying proven best practices. This guidance can help organizations achieve performant, cost-effective, and reliable gen AI deployments

Intended audience

This guide is intended for the following:

- AI and machine learning (ML) teams who have trained gen AI models, like LLMs and multimodal generation models such as diffusion models, and want to deploy them for production inference
- Solution architects and technical leaders evaluating AWS inference options
- Organizations transitioning from model development to production deployment
- AWS Partners and system integrators providing inference solutions and implementation services
- Technical decision-makers assessing infrastructure requirements for AI model inference

To understand the concepts and recommendations in this guide, you should have a basic understanding of ML concepts and AWS services.

Objectives

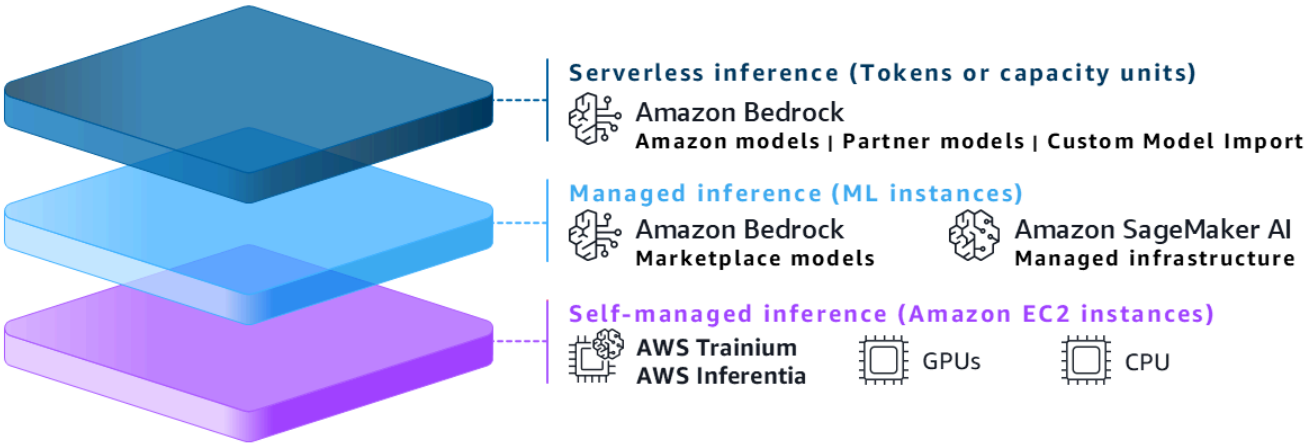
The recommendations in this guide can help you achieve the following:

- Understand the differences between AI training and inference workloads and their unique challenges.
- Navigate the AWS inference stack (serverless, managed, and self-managed) and select the appropriate service based on key decision criteria.
- Learn about model and system optimization techniques for efficient inference deployment.
- Review recommended security controls and learn about responsible AI (RAI) practices for production inference workloads.
- Leverage the AWS Partner Network to accelerate time-to-market and model distribution.

AWS inference stack

AI inference is the process of using a trained machine learning (ML) model to make predictions based on input data. Traditional ML inference typically involves relatively small models with modest compute requirements. However, generative AI (gen AI) inference has introduced new requirements in terms of compute, specialized hardware, and system optimizations.

As shown in the following diagram, AWS provides a multi-layer AI inference stack to address customers' diverse inference needs.



Following are descriptions of each layer of the AI inference stack.

Serverless inference layer

Following are key capabilities of the serverless inference layer, which is the top layer of the AI inference stack:

- Powered by Amazon Bedrock which offers a wide choice of industry leading foundation models (FMs) along with a broad set of capabilities to build generative AI applications.

- Supports importing custom models with popular architectures such as Llama, Mistral, and Qwen.
- Abstracts infrastructure management completely including high availability, scaling, and performance tuning.
- Pay-as-you-go pricing by using tokens or capacity units. For more information, see Amazon Bedrock pricing.
- Ideal for organizations who want to minimize operational and development effort.

Managed inference layer

Following are key capabilities of the managed inference layer, which is the middle layer of the AI inference stack:

- Powered by Amazon SageMaker AI which provides a broad set of tools to enable high-performance, low-cost ML for building, training and deploying AI models at scale. It provides all these capabilities in one integrated development environment (IDE).
- Enables serving pre-trained or custom models on managed infrastructure.
- Balances control with simplicity and with flexibility in choosing inference engines, scaling policies, and instance types.
- Suitable for organizations who want more control over deployment configuration, scaling behavior, and cost optimization while maintaining operational simplicity.

Self-managed inference layer

Following are key capabilities of the self-managed inference layer, which is the bottom layer of the AI inference stack:

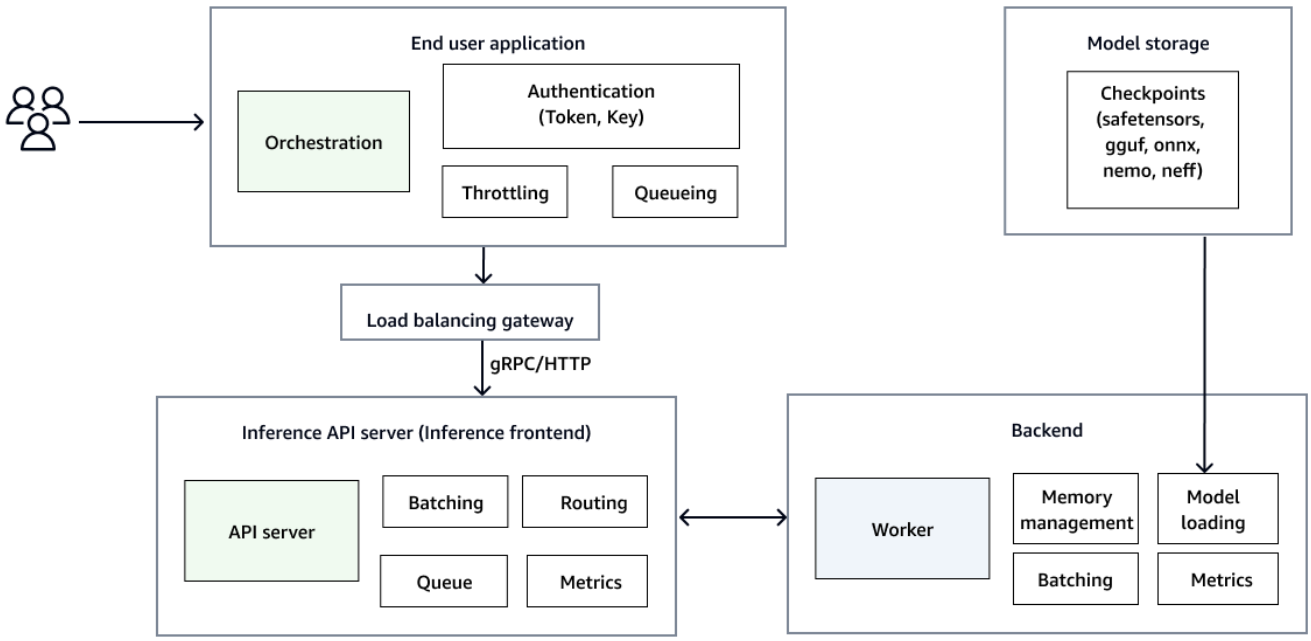
- Powered by Amazon EC2 instances with AWS Trainium and AWS Inferentia chips, AWS and NVIDIA GPUs, and CPUs. Offers broad and deep compute capabilities with over 1,000 instances and choice of the latest processor, storage, networking, operating system, and purchase model.
- Can operate through services like Amazon Elastic Kubernetes Service (Amazon EKS) or Amazon Elastic Container Service (Amazon ECS).
- Offers maximum flexibility and control of their infrastructure and software.
- Ideal for organizations who require complete control over their infrastructure.

Components of an AI inference stack

A production AI inference stack typically consists of the following three core components:

- **End user application** – Serves as the entry point for user interactions and request handling
- **Inference API server (inference frontend)** – Manages API serving and request processing
- **Inference backend** – Handles the core model execution and resource management

These components communicate through standardized protocols like gRPC for reliable and efficient model serving. The following diagram shows the high-level architecture of a real-time inference workload. It showcases the core components and how they communicate with each other.



This section provides details about each component.

End user application

The end user application component serves as the initial touchpoint for all client interactions, providing a secure and controlled entry point to the inference system. It incorporates essential security features like authentication through tokens and keys, implements throttling mechanisms to prevent system overload, and includes queuing capabilities to manage high request volumes. Some emerging tools use proxies and load balancers to enable key provisioning, multi-tenant or

multi-user throttling, queueing, and automatic failover in multi-region scenarios. Examples of these emerging tools include LiteLLM, Portkey, and Kong AI Gateway.

Inference API server or frontend

Acting as the bridge between client requests and model execution, the frontend component provides a robust API server that orchestrates the flow of inference requests. This component implements sophisticated request handling mechanisms including the following:

- Request batching for improved throughput
- Intelligent routing to distribute load across available workers
- Queue management for optimal request processing
- Comprehensive metrics collection to monitor system performance

The frontend communicates with the backend through standardized gRPC protocols, ensuring reliable and efficient data transfer. Popular frontends include NVIDIA Triton, NVIDIA Dynamo, Deep Java Library (DJL), Ray Serve, and Text Generation Inference (TGI) from Hugging Face.

Inference backend

The inference backend is where the model execution takes place, managed by specialized workers that handle the computational heavy lifting. The backend component is responsible for critical functions such as the following:

- Memory management to optimize resource utilization
- Model loading and initialization to ensure efficient startup
- Computation-level batching for improved throughput
- Performance metrics tracking

It works in close coordination with the frontend component to process requests while managing system resources effectively to maintain optimal performance and reliability. You can use a Python backend where you write your own inference logic. You can also use more optimized and popular backends such as vLLM, TensorRT LLM, ONNX Runtime, and SGLang. The backend component also maintains a connection to model storage, which houses various model formats including Safetensors, GGUF, ONNX, NeMo, NEFF, and Hugging Face models. This capability helps provide efficient access to the required model checkpoints.

Challenges of inference compared to training

Training large foundation models is often seen as the most resource-intensive phase of the AI lifecycle. However, moving to inference brings a different set of technical and operational challenges.

Training workloads are typically predictable, compute-bound, and throughput-oriented, whereas inference workloads are often more unpredictable, memory-bound, and latency sensitive.

This shift impacts every aspect of system design—from scaling and resource management to operational security. The following table outlines the key differences between training and inference challenges.

Category	Training	Inference
Operational complexity	• Planned, fixed-duration runs with predictable resource needs. • Scaling needs are clear in advance.	• Continuous or bursty workloads for real-time inference. • Autoscaling and traffic management and routing are essential to handle dynamic demand.
Latency and throughput	• Optimized for throughput. • Latency is secondary	• Latency-sensitive and concurrency-heavy. • Must deliver low time-to-first-token while supporting many simultaneous sessions.
Resource management	• Primarily compute-bound. • Large models stay loaded throughout the training. • Checkpoint writing must be handled efficiently.	• Often memory-bound because of model size and per-request state. • Requires memory-efficient serving techniques to prevent compute underutilization.

	• Node failure management for long-running training workloads.	• Checkpoint loading must be efficient to enable fast scaling with dynamic demand.
Optimization strategies	• Focus on maximizing accelerator utilization and fast convergence. • Optimizing data pipelines, using mixed precision. • Adaptive hyperparameters.	• Focus on minimizing per-request latency. • High efficiency and cost-per-request. • Leverage hardware accelerator architecture-aware optimizations, model compression, and caching to serve responses faster.
Security and compliance	• Controlled, internal environments with limited external exposure.	• Public or customer-facing endpoints. • Often requires tenant isolation and API key management. • Traffic encryption and compliance enforcement.
Cost control	• Predictable, capped costs tied to the length of the training run and the provisioned hardware resources.	• Variable, usage-driven costs. • Risk of over-provisioning or idle spend because of request spikes. • Scaling must be carefully tuned to match demand patterns.

Selecting an AWS service for AI inference

When selecting an AWS service for AI inference, organizations need to consider multiple factors that align with their operational requirements and technical capabilities.

Key decision criteria to consider when choosing an AWS service include the following:

- Level of infrastructure management desired
- Existing compute commitments
- Pricing model preferences
- Required model architecture support
- Level of control needed in inference engine selection, configuration, and infrastructure

Relevant AWS services include Amazon Bedrock, Amazon SageMaker AI managed endpoints, Amazon SageMaker HyperPod, or self-managed infrastructure using Amazon EC2, Amazon EKS, and Amazon ECS.

Amazon Bedrock

Amazon Bedrock provides a fully managed environment for model inference, automatically handling all infrastructure provisioning and scaling. It is optimized for seamless use of off-the-shelf foundation models, while also supporting the import of customized models for select architectures.

Infrastructure management

AWS manages the complete infrastructure lifecycle, including provisioning, scaling, and maintenance. This capability allows teams to focus on application development.

Pricing model

Amazon Bedrock offers flexible pricing based on actual consumption. Organizations pay for input and output tokens processed during on-demand usage, or they can use and scale imported FMs measured in capacity units.

Model architecture support

Amazon Bedrock provides access to a wide selection of foundation models, including Amazon Nova, Anthropic Claude, Mistral, Meta Llama, and Qwen. With the Custom Model Import feature, you can use custom models that align with supported architectures, enabling deployment of fine-tuned models.

Automatic scaling

AWS automatically adjusts capacity based on demand patterns, which helps to provide consistent performance during traffic fluctuations.

Inference engine choice

Amazon Bedrock provides a fully managed inference engine optimized for the supported model architectures, with AWS handling all engine configuration and optimization.

Inference configuration

You can control model behavior through model-specific request parameters. This capability allows customization of generation characteristics such as temperature, top-p sampling, and maximum token length for each inference request.

Supported clients and protocols

Applications can access models in Amazon Bedrock through the InvokeModel or Converse API actions over HTTPS or integrate using the AWS SDK. This approach provides straightforward integration paths for various application architectures. You can also use third-party open source libraries like Strands Agents or Crew AI to interact with Amazon Bedrock.

Amazon SageMaker AI inference

SageMaker AI inference endpoints deliver a managed inference environment specifically designed for deploying ML models at scale.

Infrastructure management

SageMaker AI manages the underlying ML instances while providing organizations with control over instance types and configurations. The service handles instance provisioning, health

monitoring, and infrastructure maintenance. With these capabilities, teams can focus on model deployment configuration with containers maintained by AWS or custom containers and create inference pipelines for multi-model workflows.

Pricing model

Customers are charged based on the instance types deployed, with costs covering both compute resources and storage. This instance-based pricing model enables accurate cost forecasting based on instance selection and deployment duration.

Model architecture support

SageMaker AI accommodates any compatible model framework or architecture, providing flexibility to deploy models built with TensorFlow, PyTorch, Hugging Face, scikit-learn, XGBoost, or custom frameworks.

Automatic scaling

You can configure automatic scaling policies that adjust the number of instances or replicas based on multiple factors. For example, factors include time schedules, built-in Amazon CloudWatch metrics such as invocation rates, concurrent requests, resource utilization, or custom metrics tailored to specific application requirements.

Inference engine choice

Teams can choose AWS Deep Learning Containers images that are optimized for popular frameworks and maintained by AWS. Or, teams can bring custom container images that include specialized inference engines or dependencies.

Inference configuration

The service enables flexible configuration of both model parameters and system-level settings, including the following:

- Model's precision, such as FP16, FP8, or INT4
- Engine-specific optimization strategies like prefix caching and speculative decoding to improve performance
- Model sharding techniques for distributing large models across multiple GPUs or Trainium NeuronCores

Supported clients and protocols

Applications can invoke endpoints through the Amazon SageMaker Runtime API actions over HTTPS, integrate using the AWS SDK, or leverage the Amazon SageMaker Python SDK for streamlined Python-based interactions.

Amazon SageMaker HyperPod

Amazon SageMaker HyperPod with Amazon EKS orchestration enables large-scale, distributed model inference with enterprise-grade orchestration capabilities. For more information, see Orchestrating SageMaker HyperPod clusters with Amazon EKS in the Amazon SageMaker AI documentation.

Infrastructure management

SageMaker HyperPod provides managed ML instances that AWS automatically provisions and monitors as part of the HyperPod cluster. Kubernetes orchestration through Amazon EKS enables sophisticated workload management and scheduling.

Training compute reuse

With SageMaker HyperPod, you can use the same compute resources for both training and inference workloads, maximizing infrastructure efficiency. Additionally, you can govern the usage of compute resources across different types of workloads by using SageMakerHyperPod task governance.

Pricing model

Customers are charged based on provisioned instances covering compute and storage resources. SageMaker training plans are flexible and offer commitment-based pricing that provides reduced rates and guaranteed capacity for predictable workloads.

Model architecture support

The service accommodates any model format or architecture, with full customization of model frameworks and deployment configurations. This flexibility supports diverse use cases from standard transformer models to custom architectures with specialized requirements.

Automatic scaling

Pod scaling operates through Kubernetes Event-driven Autoscaling (KEDA) and Karpenter for node scaling, responding to built-in Amazon CloudWatch metrics, engine-specific performance indicators, or custom metrics defined by your organization. SageMaker HyperPod provides comprehensive observability over cluster resources and software components with Amazon CloudWatch Container Insights, Amazon Managed Service for Prometheus, and Amazon Managed Grafana.

Inference engine choice

You can select from Deep Learning Container images that are optimized for various frameworks and maintained by AWS. Or, you can deploy custom container images with specialized inference engines, which supports diverse deployment scenarios.

Inference configuration

The platform provides comprehensive configuration flexibility for both model and system parameters. Organizations can do the following:

- Define precision levels.
- Implement prompt caching strategies.
- Configure model sharding across GPUs, Trainium NeuronCores, or even nodes.
- Select quantization approaches.
- Enable multi-node deployments.
- Choose communication protocols.
- Support various client types through the runtime and container configuration.

Supported clients and protocols

Deployed inference workloads can be accessed by any of the following approaches, providing maximum integration flexibility:

- Directly over Application Load Balancers over HTTPS
- Through the Amazon SageMaker Runtime API
- By using the AWS SDK

- By using the SageMaker SDK
- Through any protocol or client compatible with the chosen inference engine

Self-managed inference

Self-managed inference on Amazon EC2, Amazon EKS, or Amazon ECS provides organizations with complete control over their deployment environment and configuration.

Infrastructure management

Organizations maintain full control over infrastructure configuration and management. On Amazon EC2, teams manage instances directly. On Amazon EKS, Kubernetes provides container orchestration. On Amazon ECS, AWS manages the container orchestration layer while organizations control task and service definitions.

Training compute reuse

Customers can expand or reallocate their existing Amazon EC2 capacity—currently used for non-inference tasks like training or evaluation—to support AI inference workloads.

Pricing model

Customers are charged based on provisioned compute instances, storage, and networking resources. On-Demand Capacity Reservations and Capacity Blocks for ML provide commitment-based pricing options that deliver reduced rates and guaranteed capacity for planned workloads.

Model architecture support

The environment accommodates any model format or architecture, enabling deployment of models built with any framework, custom architectures, or proprietary implementations.

Automatic scaling

Scaling mechanisms are tailored to each AWS service's capabilities:

- **Amazon EKS** – Kubernetes Horizontal Pod Autoscaler (HPA) or KEDA adjust pod replicas based on resource utilization or custom metrics. Also, for Amazon EKS, Karpenter dynamically provisions and scales nodes to match workload demands efficiently.

- **Amazon ECS** – Service autoscaling adjusts task counts based on CloudWatch metrics, target tracking policies, or step scaling configurations.
- **Amazon EC2** – Autoscaling groups manage instance fleets using dynamic scaling policies, predictive scaling, or scheduled actions.

All approaches can respond to built-in metrics (such as CPU, memory, and network) and to engine-specific performance indicators (such as request latency, queue depth, and throughput). They can also respond to custom metrics tailored to specific application requirements from Amazon CloudWatch, Amazon Managed Service for Prometheus, or third-party sources.

Inference engine choice

You can select from Deep Learning Container images that are optimized for popular frameworks and maintained by AWS. Or, they can deploy custom container images with any inference engine or specialized software stack.

Inference configuration

The environment provides complete customization of system configuration. Organizations can implement any quantization level, configure model sharding strategies, enable prompt caching mechanisms, define batching behaviors, select communication protocols, and optimize for specific performance characteristics.

Supported clients and protocols

Inference API servers can be exposed through Application Load Balancers, Network Load Balancers, or Kubernetes ingress controllers. All approaches support any protocol or client compatible with the deployed inference engine or container. This flexibility enables integration with diverse application architectures and client requirements.

Model optimization techniques

You can apply several key optimization techniques to improve gen AI model performance and efficiency.

Pruning

Pruning removes less important or redundant connections and neurons from a model, resulting in a sparser, more efficient network. This process can involve removing entire units like neurons or channels or setting individual weight parameters to zero based on various importance criteria. The goal is to reduce the model's size and computational requirements while maintaining its performance capabilities. When properly implemented, pruning can significantly decrease the resource requirements of a model without substantially impacting its accuracy. Pruning is an effective optimization technique for deployment scenarios where efficiency is crucial.

Quantization

Quantization reduces both model and cache memory footprint while accelerating memory I/O and compute latency. This approach works by converting higher precision weights to lower precision formats, such as converting from FP16 to INT8. The key advantage of quantization is its ability to significantly reduce model size and improve performance while minimizing accuracy loss. Quantization is particularly effective for deployment in resource-constrained environments.

Model compilation

Model compilation optimizes the underlying operators for specific hardware architectures. This process involves combining multiple operations for more efficient execution and eliminating redundant computations throughout the model. By creating hardware-specific optimized code paths, compilation ensures that models run as efficiently as possible on the target hardware platform. Compilation can result in improved inference speed and resource utilization.

Speculative decoding

Speculative decoding is a technique to speed up the decoding process of large LLMs. It optimizes models for latency without compromising the quality of the generated text.

This technique can use a smaller but faster "draft" model to propose candidate tokens, which are then validated by the larger but slower "target" model. This is known as the draft-target approach. Alternatively, the Extrapolation Algorithm for Greater Language-model Efficiency (EAGLE) approach attaches a lightweight "EAGLE head" directly to the target model, rather than using a separate draft model. The EAGLE head generates an entire tree of potential token candidates by extrapolating from the target model's internal hidden state features. The target model can then efficiently verify and prune this draft token tree in a single parallel pass.

In either case, at each iteration, the draft or EAGLE mechanism generates multiple candidate tokens. The target model verifies the tokens, and if it finds that a particular token is not acceptable, it rejects that token and regenerates it. So, the target model both verifies tokens and generates a small amount of them.

The draft or EAGLE mechanism is significantly faster than the target model. It generates all the token candidates quickly and then sends them to the target model for parallel verification. This speeds up the final response compared to standard sequential token generation.

SageMaker AI offers a pre-built draft model that you can use, so you don't have to build your own. If you prefer to use your own custom draft model, SageMaker AI also supports this option.

Artifact storage

Efficient artifact storage maintains model weights and parameters in a ready-to-use state, enabling quick pre-loading and caching of examples. This optimization technique reduces cold-start times and streamlines the storage and retrieval of model weights. Proper artifact management is important for maintaining consistent performance in production environments, especially with large models, where checkpoints can approach hundreds of gigabytes. SageMaker AI provides functionalities to facilitate artifact management.

Applying inference optimizations in SageMaker AI

SageMaker AI leverages the techniques described earlier. With SageMaker AI, you can improve the performance of your generative AI models by applying inference optimization techniques. By optimizing your models, you can attain better cost performance for your use case. When you optimize a model, you choose which supported optimization technique to apply, including quantization, speculative decoding, compilation, and fast model loading. After your model is optimized, you can run an evaluation to see performance metrics for latency, throughput, and price.

Right-sizing and auto-scaling an inference system

Fundamental to designing an inference system is the selection of the underlying compute infrastructure and the policies for dynamically scaling it based on inference demand.

Selecting Amazon EC2 instance types and instance count

Choosing a suitable Amazon Elastic Compute Cloud (Amazon EC2) instance type is the first step in right-sizing infrastructure for a given model. EC2 instance types provide a flexible choice of GPUs, CPUs, and custom chips like AWS Inferentia and AWS Trainium. The instance type and count on a given instance constrains the throughput compute and memory that is available for hosting the model.

Model memory requirements should be satisfied by the available high-bandwidth memory on each individual instance (for example, using multiple GPUs through tensor parallelism). Model compute requirements can be split among multiple instances that serve multiple copies of the model. These instances are fronted by an inference API server that routes requests to individual instances. An example of a compute requirement is the required Requests per Second (RPS).

Inference memory requirements are mainly driven by model parameters and key-value (KV) cache size. You can estimate the memory requirements for model parameters based on the parameter count and the numerical precision of the weights. The following table shows approximate memory requirements for model parameters at a given precision for popular model sizes.

Model size (parameters)	16-bit	8-bit	4-bit
7B	14 GB	7 GB	3.5 GB
13B	26 GB	13 GB	6.5 GB
70B	140 GB	70 GB	35 GB

The memory requirements for KV cache scale roughly linearly with the number of tokens in the context length, the batch size (for example, multiple users), and the number of attention heads. They also depend on the numerical precision used to store the cache. If multiple users are using

the same endpoint on a given GPU, this will additionally scale the memory requirements. Several calculation tools can estimate the KV cache size for open source models, for example, Hugging Face and lmcache.ai. As a rough rule of thumb, the KV cache can take up memory on the same order as the model weights themselves (often approximately 50% but can be higher for long contexts).

Most modern inference backends support tensor parallelism, which distributes the model parameters and KV cache across multiple GPUs on the same node. For example, a p5.48xlarge instance has 8× H100 GPUs, each with 80 GB of memory, for a total of 640 GB. This approach allows inference on models whose parameters and KV cache exceed the memory of a single GPU. The tradeoff is some synchronization overhead due to cross-GPU communication.

Inference compute requirements are mainly driven by the size and architecture of the model and the workload characteristics (for example, concurrent requests, acceptable latency, and sequence length). Primary indicators for the capability of an inference setup are Requests per Second (RPS) for throughput and Time to First Token (TTFT) for latency. Based on the use case requirements (for example, number of concurrent users), these indicators can be used to estimate the required number of GPUs and instances. Following are two main options to deduct the achievable throughput and latency based on a given model and use case:

- If the model is a (potentially fine-tuned) popular open source model, throughput benchmark results for popular GPU and instance types are available on websites such as artificialanalysis.ai. Throughput benchmark results for several Llama, Mistral, and Falcon models are available in Benchmark and optimize endpoint deployment in Amazon SageMaker JumpStart (AWS Blog).
- If the model is heavily customized or the GPU choices are not included in public benchmarks, individual benchmarks can be performed using open source tools such as LLMPerf and the vLLM benchmark suite.

The throughput metric RPS depends heavily on the specific requests that were used within the benchmark. Popular benchmarking datasets such as ShareGPT are based on real user conversations and can approximate real-world workloads.

Although we generally recommend accelerated computing instance types (Inf, Trn, G and P instance families) for most inference use cases, CPU-based instances can be cost-effective for inference with smaller quantized models.

Configuring auto-scaling policies

After the baseline infrastructure setup is defined, auto-scaling ensures capacity adjusts dynamically with demand. This approach helps to ensure both sufficient capacity to meet latency and throughput requirements and effective resource utilization, minimizing idle infrastructure costs.

Gen AI inference systems typically use horizontal scaling (adding or removing instances) in contrast to vertical scaling (adjusting instance types or GPU utilization).

Scaling strategies can generally be classified as reactive (based on the current load on the system) and scheduled (for workloads with predictable temporal patterns). Both strategies can be combined, depending on the use case specifications.

Auto-scaling policies rely on metrics that reflect the load of the inference system. Common metrics include:

- **Requests per Second** - Tracking RPS against the maximum tested throughput per instance type (see the previous section) helps maintain predictable performance.
- **KV Cache Usage** - Average key-value cache utilization across instances indicates that GPUs are saturated and additional instances might be required.
- **Request Queue Length** or **Pending Requests** – A growing request queue indicates demand is exceeding available throughput. This metric correlates directly with user-facing latency.

AWS provides multiple approaches for implementing auto-scaling, depending on the chosen inference service:

- Amazon Bedrock provides fully managed and automatic scaling of the inference capacity for both publicly available foundation models and models imported through Custom Model Import. No manual setup is required.
- Amazon SageMaker endpoint auto scaling provides fully managed scaling based on configurable thresholds on invocation metrics as well as custom CloudWatch metrics such as CPU utilization.
- Amazon SageMaker HyperPod and inference setups based on Amazon EKS can use Kubernetes-based auto scaling functionality through KEDA and Karpenter.
- Amazon ECS Managed Instances provides flexible cost-effective auto-scaling for inference workloads.

There are two primary challenges with auto-scaling setups:

- **Cold start latency** - Scale-out events have latency due to instance provisioning, container image pull, and model weight loading.
- **Capacity availability** - Depending on the AWS Region, the desired instance types can experience varying levels of availability, increasing latency in scale-out events or even prohibiting them during certain periods.

System-level optimizations

Beyond instance selection and scaling, the following system-level optimizations can improve inference efficiency, reduce latency, and increase throughput:

- **Routing and load balancing** - Efficient request routing ensures balanced utilization across GPUs and instances. Common strategies include:
  - **Round-robin routing** for even distribution of independent requests.
  - **Session-aware routing** to keep user conversations on the same instance, preserving KV cache locality.
  - **Latency-aware routing** to send requests to the least-loaded instance for reduced queueing delay.
- **Throughput and latency** - Batch size is one of the primary levers for tuning performance. Smaller batches favor low latency, while larger batches improve throughput. Optimal settings depend on workload characteristics (for example, interactive chat or batch inference).
- **Prefix caching** - Caching identical prompt prefixes avoids redundant compute and decoding, significantly lowering latency for repeated queries. This approach is especially effective in applications with high overlap in user prompts. Frameworks such as vLLM (`PagedAttention`), Text Generation Inference (TGI), SGLang, and `Llama.cpp` natively support prefix caching.
- **Chunked prefill** - Large context inputs can stall GPUs during prefill. Splitting prefill into smaller chunks reduces idle decoding cycles, improving utilization for long-sequence workloads.
- **Continuous batching** - Continuous batching dynamically groups incoming requests, prioritizing memory I/O and compute to maximize throughput while keeping latency within acceptable bounds. This approach prevents resource underutilization when request patterns are uneven.
- **Sequence size tuning** - Right-sizing the maximum sequence length reduces unnecessary computation for workloads that rarely need the full context window. This tuning balances memory footprint, throughput, and latency.

Implementing security and responsible AI

Building generative AI (gen AI) models involves complex technical and business decisions. To speed up experimentation, some responsible AI (RAI) evaluations might be overlooked initially. However, conducting an RAI review *before* moving to inference helps organizations strengthen risk assessment and resource management through a structured evaluation process. This pre-release review serves as the final controlled stage where organizations can thoroughly assess models for potential harm. For example, organizations can assess models for bias amplification, privacy violations, safety risks, or misalignment with organizational values.

The RAI review consists of two phases:

- Before moving to inference
- After inference deployment

Before moving to inference

To establish a foundation for responsible deployment, implement a model registry to track lineage, versions, and artifacts, ensuring traceability throughout the model's lifecycle. The model evaluation report provides a detailed assessment that quantifies robustness, fairness across demographic groups, and potential failure modes. This assessment offers visibility into unexpected behavior and supports informed trade-off discussions. These evaluations guide the creation of comprehensive governance structures, including formal approval workflows, stakeholder sign-offs, and escalation paths that maintain organizational awareness of potential risks and impacts.

To complement these governance mechanisms, implement technical safeguards such as multi-layered defensive controls for input validation, output filtering, and personally identifiable information (PII) sanitization. Finally, practical access controls—including API key authentication, identity-based authorization, and network isolation—protect inference endpoints from unauthorized use, abuse, and resource exhaustion.

Model assessment

During the model assessment phase, the evaluation report helps the organization understand the model's robustness and explainability through quantitative and qualitative analyses. This analysis includes evaluating accuracy metrics, fairness scores, and error distributions, complemented by human-in-the-loop reviews and A/B testing to validate real-world performance. The report should

also provide visibility into unexpected behavior, foster trade-off discussions, and guide the team in incorporating mechanisms for exception management.

Documentation preparation

With the model assessment report, the team can now develop internal documentation to record trade-offs and decisions made during the process. This documentation provides valuable context for future development iterations. When the documentation is revisited, it helps the team understand the limitations and requirements of the previous project time frame.

A model card should include detailed specifications of the model architecture, training methodology, and hyperparameter configurations, along with performance metrics across diverse datasets and demographic groups. It must also explicitly outline both intended use cases and scenarios where the model should not be applied, establishing clear boundaries for appropriate deployment.

Technical safeguards

Technical safeguards protect the system from potential harm through engineering controls and architectural design choices. Before model deployment, developers must establish multi-layered defensive mechanisms, including the following:

- Input validation to detect and reject adversarial prompts
- Output filtering systems to identify and block harmful, misleading, or biased content
- Thorough sanitization processes for PII

Amazon Bedrock Guardrails provides safeguards that you can configure for your generative AI applications based on your use cases and responsible AI policies.

Governance setup

Before deploying a model, organizations should establish formal deployment approval workflows. These procedures help to ensure that all stakeholders review evaluation reports and have access to shared documentation such as model cards and decision records. This approach fosters organizational awareness of potential risks and impacts associated with deployment. A structured approval process should be completed before release. Governance execution must align with previously documented requirements and include technical controls with clear purposes, such as configuring access permissions and rate limits to prevent resource abuse.

It's essential to control access to inference endpoints to prevent abuse, unauthorized use, and excessive consumption—particularly for public- or customer-facing systems. The following complementary access control methods are commonly used:

- **API key authentication** – API keys provide a simple form of authentication that is natively supported by most popular frameworks and API specifications such as the OpenAI API. Managed platforms such as Amazon Bedrock and open source frameworks such as LiteLLM provide options for issuing API keys to users through the UI and programmatically. API keys are associated with allowed models and usage limits. API keys should be securely loaded from an environment variable or key management service on the server and not be exposed to client-side code. Furthermore, API keys should only be valid short-term and rotated frequently, to prevent abuse in case of accidental leakage.
- **Identity-based authentication** – Identity-based authentication relies on short-lived credentials and signed requests instead of shared secrets like API keys. Within AWS, callers assume AWS Identity and Access Management (IAM) roles. Authorization is enforced through IAM policies that scope allowed models, endpoints, and actions (for example, invoke, batch, and embeddings). In public- and customer-facing applications, end users typically authenticate through OAuth2/OIDC (for example, through Amazon Cognito). Then, identities are mapped to their respective IAM roles.
- **Network isolation** – For company-internal inference endpoints, network isolation can be enforced by only exposing access with internal load balancers within a virtual private cloud (VPC) and by scoping access with security groups and network access control lists. If public ingress is unavoidable, use IP allowlists where possible. Also in that situation, the inference endpoint should be fronted by services such as Amazon CloudFront and AWS WAF. These AWS services prevent distributed denial of service (DDoS) attacks and other common exploits. Network isolation complements API keys and identity-based authentication so that even leaked credentials can't reach an endpoint directly.

In addition to access control, request payloads should always be encrypted in transit by enforcing SSL/TLS.

After inference deployment

A gen AI model deployment guided by RAI practices integrates three key elements to ensure models remain safe, effective, and aligned with organizational values:

- Continuous monitoring provides observability through real-time tracking of performance metrics such as latency, throughput, and error rates. It also observes contextual metadata that enables analysis across dimensions like criticality, business domain, compliance, and resource use.
- When issues arise, incident response mechanisms act as a rapid intervention system. Predefined procedures and clear roles and responsibilities help stakeholders access information and resources quickly to minimize impact.
- Complementing these systems, feedback mechanisms capture user experiences that automated metrics might overlook. These mechanisms provide early warnings for hallucinations, bias, or harmful outputs while incorporating diverse perspectives to better reveal model limitations in real-world contexts.

Continuous monitoring

Continuous monitoring provides real-time visibility into model behavior and system performance across the inference lifecycle. Amazon SageMaker AI enables this observability through components such as Model Monitor for detecting data and concept drift, SageMaker Clarify for bias detection and explanations, and SageMaker Debugger for real-time performance tracking. These tools, combined with tagging model interactions by use case, generate contextual metadata that supports deeper analysis and operational insights. For example, critical versus non-critical workload tags enable prioritization during incident response. Business domain tags support performance comparisons. Compliance tags facilitate automated policy enforcement. Cost center tags improve resource tracking and financial accountability.

Incident response mechanisms

Establishing incident response mechanisms helps to mitigate potential harm and keep deployed gen AI models reliable. Address incidents as quickly as possible to minimize their impact on end users. These systems must also be designed for accessibility, so that response stakeholders and subject owners have immediate access to relevant information and resources.

Feedback collection mechanisms

The indeterministic nature of gen AI models can still lead to unexpected behavior for your customers. Customer feedback serves as an early warning system, identifying hallucinations, biases, or harmful responses that automated quality metrics may miss. Incorporating feedback from diverse user groups helps organizations better understand model limitations and real-world data distributions. By systematically collecting, categorizing, and analyzing both positive and

negative feedback, organizations can continuously refine their understanding of use cases, user values, and expectations.

AWS Partner Network

The AWS Partner Network (APN) is a global community of partners that leverage programs, expertise, and resources to build, market, and sell customer solutions. AWS AI Competency Partners drive the advancement of services, tools, data strategy, and infrastructure pivotal for implementing generative AI (gen AI) technologies across diverse industries. Their customer success and technical best practices have been validated by gen AI specialists at AWS. Partners in the gen AI category can help organizations in the following ways:

- **Consulting services** – Partners with deep expertise can assist organizations with the following:
  - Adopt and strategize gen AI.
  - Build and test gen AI applications.
  - Train and customize foundation models.
  - Operate and maintain gen AI applications and models.
  - Protect gen AI workloads.
  - Define responsible AI principles and frameworks.
- **Gen AI applications** – Partners provide tools and solutions that deliver customer value across the full AWS gen AI stack, from foundation models to end user applications.
- **Foundation models and app development** – Partners can help organizations with foundation model training, customization, and the development of gen AI-powered applications.
- **Infrastructure and data** – Partners offer expertise in areas like data strategy, governance, storage, and the underlying infrastructure required to power gen AI workloads on AWS.

By leveraging the capabilities of AWS Generative AI Competency Partners, organizations can accelerate their generative AI initiatives, drive innovation, and transform their businesses.

Resources

AWS service documentation

- Amazon Bedrock
- Amazon Bedrock Custom Model Import
- Amazon Bedrock Guardrails
- Amazon SageMaker AI
- Amazon Elastic Compute Cloud
- Amazon Elastic Container Service
- Amazon Elastic Kubernetes Service
- Deploy models for inference on Amazon SageMaker AI
- Deploying models on Amazon SageMaker HyperPod
- How inference works in Amazon Bedrock

Other AWS resources

- AI on Amazon EKS (AWS GitHub repository)
- Benchmark and optimize endpoint deployment in Amazon SageMaker JumpStart (AWS Blog)
- Deploy LLMs on Amazon EKS using vLLM Deep Learning Containers (AWS Blog)
- Managed Tiered KV Cache and Intelligent Routing for Amazon SageMaker HyperPod (AWS Blog)
- Responsible AI
- Amazon SageMaker HyperPod workshop (AWS Workshop)

Other resources

- EAGLE-3: Scaling up Inference Acceleration of Large Language Models via Training-Time Test (arXiv)
- An Introduction to Speculative Decoding for Reducing Latency in AI Inference (Nvidia)
- Overview of quantization techniques (Hugging Face)

Contributors

The following individuals contributed to this guide.

Main authors:

- Oussama Kandakji, Senior AI/ML Go-To-Market Specialist Solutions Architect, AWS
- Nicolas Jourdan, Senior AI/ML Go-To-Market Specialist Solutions Architect, AWS

Co-authors:

- Mia Chang, AI/ML Go-To-Market Specialist Solutions Architect, AWS
- Marco Sgroi, Principal AI/ML Go-To-Market Specialist, AWS

Reviewers:

- Roy Allela, Senior AI/ML Specialist Solutions Architect, AWS
- Robert Northard, Principal Containers Specialist Solutions Architect, AWS

Document history

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

Change	Description	Date
Initial publication	—	December 18, 2025

AWS Prescriptive Guidance glossary

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

Numbers

7 Rs

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

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

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

A

ABAC

See attribute-based access control.

abstracted services

See managed services.

ACID

See atomicity, consistency, isolation, durability.

active-active migration

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

active-passive migration

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

aggregate function

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

AI

See artificial intelligence.

AIOps

See artificial intelligence operations.

anonymization

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

anti-pattern

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

application control

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

application portfolio

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

artificial intelligence (AI)

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

artificial intelligence operations (AIOps)

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

asymmetric encryption

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

atomicity, consistency, isolation, durability (ACID)

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

attribute-based access control (ABAC)

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

authoritative data source

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

Availability Zone

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

AWS Cloud Adoption Framework (AWS CAF)

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

AWS Workload Qualification Framework (AWS WQF)

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

B

bad bot

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

BCP

See business continuity planning.

behavior graph

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

big-endian system

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

binary classification

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

bloom filter

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

blue/green deployment

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

bot

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

botnet

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

branch

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

break-glass access

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

brownfield strategy

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

buffer cache

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

business capability

What a business does to generate value (for example, sales, customer service, or marketing). Microservices architectures and development decisions can be driven by business capabilities. For more information, see the Organized around business capabilities section of the Running containerized microservices on AWS whitepaper.

business continuity planning (BCP)

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

C

CAF

See AWS Cloud Adoption Framework.

canary deployment

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

CCoE

See Cloud Center of Excellence.

CDC

See change data capture.

change data capture (CDC)

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

chaos engineering

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

CI/CD

See continuous integration and continuous delivery.

classification

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

client-side encryption

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

Cloud Center of Excellence (CCoE)

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

cloud computing

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

cloud operating model

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

cloud stages of adoption

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

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

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

CMDB

See configuration management database.

code repository

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

cold cache

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

cold data

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

computer vision (CV)

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

configuration drift

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

configuration management database (CMDB)

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

conformance pack

A collection of AWS Config rules and remediation actions that you can assemble to customize your compliance and security checks. You can deploy a conformance pack as a single entity in an AWS account and Region, or across an organization, by using a YAML template. For more information, see Conformance packs in the AWS Config documentation.

continuous integration and continuous delivery (CI/CD)

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

CV

See computer vision.

D

data at rest

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

data classification

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

data drift

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

data in transit

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

data mesh

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

data minimization

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

data perimeter

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

data preprocessing

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

data provenance

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

data subject

An individual whose data is being collected and processed.

data warehouse

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

database definition language (DDL)

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

database manipulation language (DML)

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

DDL

See database definition language.

deep ensemble

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

deep learning

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

defense-in-depth

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

delegated administrator

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

deployment

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

development environment

See environment.

detective control

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

development value stream mapping (DVSM)

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

digital twin

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

dimension table

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

disaster

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

disaster recovery (DR)

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

DML

See database manipulation language.

domain-driven design

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

DR

See disaster recovery.

drift detection

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

DVSM

See development value stream mapping.

E

EDA

See exploratory data analysis.

EDI

See electronic data interchange.

edge computing

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

electronic data interchange (EDI)

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

encryption

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

encryption key

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

endianness

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

endpoint

See service endpoint.

endpoint service

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

information, see Create an endpoint service in the Amazon Virtual Private Cloud (Amazon VPC) documentation.

enterprise resource planning (ERP)

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

envelope encryption

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

environment

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

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

epic

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

ERP

See enterprise resource planning.

exploratory data analysis (EDA)

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

F

fact table

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

fail fast

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

fault isolation boundary

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

feature branch

See branch.

features

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

feature importance

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

feature transformation

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

few-shot prompting

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

FGAC

See fine-grained access control.

fine-grained access control (FGAC)

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

flash-cut migration

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

FM

See foundation model.

foundation model (FM)

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

G

generative AI

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

geo blocking

See geographic restrictions.

geographic restrictions (geo blocking)

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

Gitflow workflow

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

golden image

A snapshot of a system or software that is used as a template to deploy new instances of that system or software. For example, in manufacturing, a golden image can be used to provision software on multiple devices and helps improve speed, scalability, and productivity in device manufacturing operations.

greenfield strategy

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

guardrail

A high-level rule that helps govern resources, policies, and compliance across organizational units (OUs). *Preventive guardrails* enforce policies to ensure alignment to compliance standards. They are implemented by using service control policies and IAM permissions boundaries.

Detective guardrails detect policy violations and compliance issues, and generate alerts for remediation. They are implemented by using AWS Config, AWS Security Hub CSPM, Amazon GuardDuty, AWS Trusted Advisor, Amazon Inspector, and custom AWS Lambda checks.

H

HA

See high availability.

heterogeneous database migration

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

high availability (HA)

The ability of a workload to operate continuously, without intervention, in the event of challenges or disasters. HA systems are designed to automatically fail over, consistently deliver high-quality performance, and handle different loads and failures with minimal performance impact.

historian modernization

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

holdout data

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

homogeneous database migration

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

hot data

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

hotfix

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

hypercare period

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

I

IaC

See infrastructure as code.

identity-based policy

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

idle application

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

IIoT

See industrial Internet of Things.

immutable infrastructure

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

inbound (ingress) VPC

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

incremental migration

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

Industry 4.0

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

infrastructure

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

infrastructure as code (IaC)

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

industrial Internet of Things (IIoT)

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

inspection VPC

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

Internet of Things (IoT)

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

interpretability

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

IoT

See Internet of Things.

IT information library (ITIL)

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

IT service management (ITSM)

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

ITIL

See IT information library.

ITSM

See IT service management.

L

label-based access control (LBAC)

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

landing zone

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

large language model (LLM)

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

large migration

A migration of 300 or more servers.

LBAC

See label-based access control.

least privilege

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

lift and shift

See 7 Rs.

little-endian system

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

LLM

See large language model.

lower environments

See environment.

M

machine learning (ML)

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

main branch

See branch.

malware

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

managed services

AWS services for which AWS operates the infrastructure layer, the operating system, and platforms, and you access the endpoints to store and retrieve data. Amazon Simple Storage Service (Amazon S3) and Amazon DynamoDB are examples of managed services. These are also known as *abstracted services*.

manufacturing execution system (MES)

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

MAP

See Migration Acceleration Program.

mechanism

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

member account

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

MES

See manufacturing execution system.

Message Queuing Telemetry Transport (MQTT)

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

microservice

A small, independent service that communicates over well-defined APIs and is typically owned by small, self-contained teams. For example, an insurance system might include microservices that map to business capabilities, such as sales or marketing, or subdomains, such as purchasing, claims, or analytics. The benefits of microservices include agility, flexible scaling, easy deployment, reusable code, and resilience. For more information, see Integrating microservices by using AWS serverless services.

microservices architecture

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

Migration Acceleration Program (MAP)

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

migration at scale

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

migration factory

Cross-functional teams that streamline the migration of workloads through automated, agile approaches. Migration factory teams typically include operations, business analysts and owners,

migration engineers, developers, and DevOps professionals working in sprints. Between 20 and 50 percent of an enterprise application portfolio consists of repeated patterns that can be optimized by a factory approach. For more information, see the discussion of migration factories and the Cloud Migration Factory guide in this content set.

migration metadata

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

migration pattern

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

Migration Portfolio Assessment (MPA)

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

Migration Readiness Assessment (MRA)

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

migration strategy

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

ML

See machine learning.

modernization

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

modernization readiness assessment

An evaluation that helps determine the modernization readiness of an organization's applications; identifies benefits, risks, and dependencies; and determines how well the organization can support the future state of those applications. The outcome of the assessment is a blueprint of the target architecture, a roadmap that details development phases and milestones for the modernization process, and an action plan for addressing identified gaps. For more information, see Evaluating modernization readiness for applications in the AWS Cloud.

monolithic applications (monoliths)

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

MPA

See Migration Portfolio Assessment.

MQTT

See Message Queuing Telemetry Transport.

multiclass classification

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

mutable infrastructure

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

O

OAC

See origin access control.

OAI

See origin access identity.

OCM

See organizational change management.

offline migration

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

OI

See operations integration.

OLA

See operational-level agreement.

online migration

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

OPC-UA

See Open Process Communications - Unified Architecture.

Open Process Communications - Unified Architecture (OPC-UA)

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

operational-level agreement (OLA)

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

operational readiness review (ORR)

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

operational technology (OT)

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

operations integration (OI)

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

organization trail

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

organizational change management (OCM)

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

origin access control (OAC)

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

origin access identity (OAI)

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

ORR

See operational readiness review.

OT

See operational technology.

outbound (egress) VPC

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

P

permissions boundary

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

personally identifiable information (PII)

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

PII

See personally identifiable information.

playbook

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

PLC

See programmable logic controller.

PLM

See product lifecycle management.

policy

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

polyglot persistence

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

portfolio assessment

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

predicate

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

predicate pushdown

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

preventative control

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

principal

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

privacy by design

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

private hosted zones

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

proactive control

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

product lifecycle management (PLM)

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

production environment

See environment.

programmable logic controller (PLC)

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

prompt chaining

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

pseudonymization

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

publish/subscribe (pub/sub)

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

Q

query plan

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

query plan regression

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

R

RACI matrix

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

RAG

See Retrieval Augmented Generation.

ransomware

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

RASCI matrix

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

RCAC

See row and column access control.

read replica

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

re-architect

See 7 Rs.

recovery point objective (RPO)

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

recovery time objective (RTO)

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

refactor

See 7 Rs.

Region

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

regression

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

rehost

See 7 Rs.

release

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

relocate

See 7 Rs.

replatform

See 7 Rs.

repurchase

See 7 Rs.

resiliency

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

resource-based policy

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

responsible, accountable, consulted, informed (RACI) matrix

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

responsive control

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

retain

See 7 Rs.

retire

See 7 Rs.

Retrieval Augmented Generation (RAG)

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

rotation

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

row and column access control (RCAC)

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

RPO

See recovery point objective.

RTO

See recovery time objective.

runbook

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

S

SAML 2.0

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

SCADA

See supervisory control and data acquisition.

SCP

See service control policy.

secret

In AWS Secrets Manager, confidential or restricted information, such as a password or user credentials, that you store in encrypted form. It consists of the secret value and its metadata.

The secret value can be binary, a single string, or multiple strings. For more information, see What's in a Secrets Manager secret? in the Secrets Manager documentation.

security by design

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

security control

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

security hardening

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

security information and event management (SIEM) system

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

security response automation

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

server-side encryption

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

service control policy (SCP)

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

service endpoint

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

service-level agreement (SLA)

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

service-level indicator (SLI)

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

service-level objective (SLO)

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

shared responsibility model

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

SIEM

See security information and event management system.

single point of failure (SPOF)

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

SLA

See service-level agreement.

SLI

See service-level indicator.

SLO

See service-level objective.

split-and-seed model

A pattern for scaling and accelerating modernization projects. As new features and product releases are defined, the core team splits up to create new product teams. This helps scale your

organization's capabilities and services, improves developer productivity, and supports rapid innovation. For more information, see Phased approach to modernizing applications in the AWS Cloud.

SPOF

See single point of failure.

star schema

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

strangler fig pattern

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

subnet

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

supervisory control and data acquisition (SCADA)

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

symmetric encryption

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

synthetic testing

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

system prompt

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

T

tags

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

target variable

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

task list

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

test environment

See environment.

training

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

transit gateway

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

trunk-based workflow

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

trusted access

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

tuning

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

two-pizza team

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

U

uncertainty

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