Scaling Amazon EKS infrastructure to optimize compute, workloads, and network performance

# AWS Prescriptive Guidance

# AWS Prescriptive Guidance: Scaling Amazon EKS infrastructure to optimize compute, workloads, and network performance

# Table of Contents

**Introduction** ..... **1**
Objectives ..... 2
**Compute scaling** ..... **4**
Cluster AutoScaler ..... 4
Cluster Autoscaler with over-provisioning ..... 5
Karpenter ..... 5
**Workload scaling** ..... **7**
Horizontal Pod Autoscaler ..... 7
Cluster Proportional Autoscaler ..... 8
Kubernetes-based Event-Driven Autoscaler ..... 9
**Network scaling** ..... **11**
Amazon VPC CNI plugin for Kubernetes ..... 11
Custom networking ..... 12
Prefix delegation ..... 13
Amazon VPC Lattice ..... 14
**Cost optimization** ..... **16**
Kubecost ..... 16
Goldilocks ..... 17
AWS Fargate ..... 18
Spot Instances ..... 18
Reserved Instances ..... 19
AWS Graviton instances ..... 20
**Next steps** ..... **21**
**Resources** ..... **22**
**Document history** ..... **23**
**Glossary** ..... **24**
# ..... 24
A ..... 25
B ..... 28
C ..... 30
D ..... 33
E ..... 37
F ..... 39
G ..... 41

H ..... 42
I ..... 44
L ..... 46
M ..... 47
O ..... 51
P ..... 54
Q ..... 57
R ..... 57
S ..... 60
T ..... 64
U ..... 65
V ..... 66
W ..... 66
Z ..... 67

# Scaling Amazon EKS infrastructure to optimize compute, workloads, and network performance

*Aniket Dekate, Aniket Kurzadkar, and Ishwar Chauthaiwale, Amazon Web Services (AWS)*

*November 2024* (document history)

Amazon Elastic Kubernetes Service (Amazon EKS) is a managed Kubernetes service. With Amazon EKS, you can run Kubernetes pods in a containerized cloud environment without needing to install and operate your own control plane. With AWS managing the control plane, Amazon EKS reduces organizational operational management. Other benefits of using Amazon EKS include scaling, reliability, and security in the cloud environment.

This guide is designed to help organizations to optimize their Amazon EKS infrastructure across the following areas:

- Compute scaling is a critical component to application performance in a dynamic Kubernetes environment:
  - **Efficient resource allocation** – Learn about techniques for allocating computed resources dynamically to meet varying demand.
  - **Automation tools** – Get an overview of tools and services that automate compute scaling, reducing the need for manual intervention.
- Workload scaling helps to make sure that applications can handle varying workloads without performance degradation:
  - **Horizontal pod autoscaler** – Take an in-depth look at how an HPA helps in scaling workloads based on real-time metrics.
  - **Cluster Proportional Autoscaler** – Learn how CPA automatically scales and maintains a proportional relationship between nodes and replicas, scaling workloads up or down as the cluster size changes.
  - **Event-driven scaling** – Review strategies for scaling applications in response to specific events or triggers.
- Network scaling helps to maintain seamless communication between services and efficient data flow in dynamic environments:
  - **Amazon VPC CNI plugin** – Learn how the VPC CNI plugin enables scalable networking within Amazon EKS clusters.

  - **Custom networking** - Review IP address management and network traffic segregation on Amazon EKS clusters.
  - **Prefix delegation** - Get an overview of streamlining IP management in large and scalable Amazon EKS clusters.
  - **Amazon VPC Lattice** – Get an overview of how VPC Lattice can manage cross-VPC and service-to-service networking for seamless scaling.
- Cost optimization helps businesses see where their resources are being spent and appropriately assign expenses to departments or projects:
  - **Right-sizing resources** – Consider techniques for sizing cloud resources appropriately for the workload.
  - **Cost monitoring and control** – Review tools and best practices for tracking and optimizing cloud expenses.

Each section focuses on particular goals that are necessary to create a reliable, effective, and affordable cloud environment.

# Objectives

This guide can help you and your organization achieve the following business objectives:

- **Enhanced resource efficiency** – Achieve optimal resource utilization by dynamically scaling compute, workloads, and network resources based on real-time demands.

  This objective emphasizes the importance of scaling resources up and down in response to actual usage patterns. Tools such as horizontal pod autoscalers and the Amazon VPC CNI plugin help organizations only use the resources that they need, minimizing waste and maximizing performance.

- **Improved application performance** – Maintain high performance and responsiveness of applications, even under fluctuating workloads and traffic patterns.

  This objective focuses on strategies to help make sure that applications can handle peak traffic and heavy workloads without compromising performance. Techniques such as event-driven workload scaling, efficient compute allocation, and scalable network architectures are key to achieving this objective.

- **Seamless scalability** – Enable smooth scaling of infrastructure components, allowing for effortless growth and adaptation to changing business needs.

  Seamless scalability is crucial for organizations that anticipate growth or experience varying traffic levels. This objective addresses the importance of implementing scalable solutions across compute, workload, and network resources, so that scaling can be automatic, efficient, and transparent.

- **Cost optimization** – Minimize cloud costs while maintaining or improving performance and scalability.

  Cost optimization can encompass reducing expenses, such as right-sizing resources, using cost-effective scaling solutions, and monitoring spending. The goal is to balance cost savings with the need for high performance and scalability.

# Compute scaling

Compute scaling is a critical component to application performance in a dynamic Kubernetes environment. Kubernetes reduces waste through the dynamic adjustment of computing resources (such as CPU and memory) in response to real-time demand. This capability helps to avoid over- or under-provisioning, which can also save operating expenses. Kubernetes effectively eliminates the need for manual intervention by enabling the infrastructure to automatically scale up during peak hours and down during off-peak periods.

The overall compute scaling of Kubernetes automates the scaling process, which boosts the application's flexibility and scalability and enhances its fault-tolerant behavior. Ultimately, the capabilities of Kubernetes enhance operational excellence and productivity.

This section discusses the following types of compute scaling:

- Cluster Autoscaler
- Cluster Autoscaler with over-provisioning
- Karpenter

## Cluster AutoScaler

Depending on the needs of the pods, the Cluster Autoscaler tool automatically modifies the size by adding nodes when necessary or removing nodes when they're not needed and are underutilized.

Consider the Cluster Autoscaler tool as a scaling solution for workloads where demand increases gradually and latency in scaling isn't a major issue.

The Cluster Autoscaler tool provides the following key features:

- **Scaling** – Scales nodes up and down dynamically in response to actual resource demands.
- **Pod scheduling** – Helps to make sure that every pod is operating and has the resources it needs to function, preventing the scarcity of resources.
- **Cost-efficiency** – Eliminates the unnecessary expenses of operating under-utilized nodes by eliminating them.

# Cluster Autoscaler with over-provisioning

Cluster Autoscaler with over-provisioning functions similarly to the Cluster Autoscaler in that it deploys nodes efficiently and saves time by running low-priority pods on the nodes. With this technique, traffic is redirected into these pods in response to sudden spikes in demand, allowing the application to continue operating without interruption.

Cluster Autoscaler with over-provisioning offers the features of dummy pods that can be used to easily deploy and run nodes when the workload is very large, latency isn't needed, and scaling needs to be quick.

Cluster Autoscaler with over-provisioning provides the following key features:

- **Better responsiveness** – By making excess capacity constantly accessible, it takes less time to scale up the cluster in response to spikes in demand.
- **Resource reservation** – Managing unexpected spikes in traffic effectively assists correct management with little downtime.
- **Smooth scaling** – Minimizing resource allocation delays facilitates a more seamless scaling process.

# Karpenter

Karpenter for Kubernetes outperforms the traditional Cluster Autoscaler tool in terms of open source, performance, and customizability. With Karpenter, you can automatically launch only the required compute resources to handle your cluster's demands in real time. Karpenter is designed to deliver more efficient and responsive scaling.

Applications with extremely variable or complex workloads, where quick scaling decisions are essential, benefit greatly from the use of Karpenter. It integrates with AWS, offering improved deployment and node selection optimization.

Karpenter includes the following key features:

- **Dynamic provisioning** – Karpenter provides the right instances and sizes for the purpose and provisions new nodes dynamically based on the particular requirements of pods.
- **Advanced scheduling** – Using clever pod placement, Karpenter arranges nodes such that resources like GPU, CPU, memory, and storage are used as effectively as possible.

- **Quick scaling** – Karpenter can scale quickly, frequently reacting in seconds. This responsiveness is helpful for patterns of sudden traffic or when the workload demands immediate scaling
- **Cost efficiency** – By carefully choosing the most effective instance, you can lower operating costs and take advantage of additional cost-saving alternatives offered by AWS, such as On-Demand Instances, Spot Instances, and Reserved Instances.

# Workload scaling

Workload scaling in Kubernetes is essential for maintaining application performance and resource efficiency in dynamic environments. Scaling helps to make sure that applications can handle varying workloads without performance degradation. Kubernetes provides the ability to automatically scale resources up or down based on real-time metrics, allowing organizations to respond quickly to changes in traffic. This elasticity not only improves user experience but also optimizes resource utilization, helping to minimize costs associated with underused or overprovisioned resources.

Additionally, effective workload scaling supports high availability, ensuring that applications remain responsive even during peak demand periods. Workload scaling in Kubernetes enables organizations to make better use of cloud resources by dynamically adjusting capacity to meet current needs.

This section discusses the following types of workload scaling:

- Horizontal Pod Autoscaler
- Cluster Proportional Autoscaler
- Kubernetes-based Event Driven Autoscaler

## Horizontal Pod Autoscaler

The Horizontal Pod Autoscaler (HPA) is a Kubernetes feature that automatically adjusts the number of pod replicas in a deployment, replication controller, or stateful set, based on observed CPU utilization or other select metrics. The HPA makes sure that applications can manage fluctuating traffic and workload levels without the need for manual intervention. The HPA offers a means of preserving optimal performance while making effective use of available resources.

In contexts where user demand might fluctuate considerably over time, such web apps, microservices, and APIs, the HPA is especially helpful.

The Horizontal Pod Autoscaler provides the following key features:

- **Automatic scaling** – HPA automatically increases or decreases the number of pod replicas in response to real-time metrics, ensuring that applications can scale to meet user demand.

- **Metrics-based decisions** – By default, HPA scales based on CPU utilization. However, it can also use custom metrics, such as memory usage or application-specific metrics, allowing for more tailored scaling strategies.
- **Configurable parameters** – You can choose the minimum and maximum replica counts and the desired utilization percentages, giving you authority over how severe the scaling should be.
- **Integration with Kubernetes** – To monitor and modify resources, HPA works in tandem with other elements of the Kubernetes ecosystem, including the Metrics Server, Kubernetes API, and custom metrics adapters.
- **Better resource utilization** – HPA assists in making sure that resources are used effectively, lowering costs and improving performance, by dynamically modifying the number of pods.

## Cluster Proportional Autoscaler

The Cluster Proportional Autoscaler (CPA) is a Kubernetes component designed to automatically adjust the number of pod replicas in a cluster based on the number of nodes available. Unlike traditional autoscalers that scale based on resource utilization metrics (like CPU and memory), CPA scales workloads in proportion to the size of the cluster itself.

This approach is particularly useful for applications that need to maintain a certain level of redundancy or availability relative to the cluster size, such as CoreDNS and other infrastructure services. Some of the main use cases for CPA include the following:

- Over-provisioning
- Scale out core platform services
- Scale out workloads because CPA doesn't require a metrics server or Prometheus Adapter

By automating the scaling process, CPA assists businesses in maintaining a balanced workload distribution, increasing resource efficiency, and making sure that applications are suitably provisioned to satisfy user demand.

The Cluster Proportional Autoscaler provides the following key features:

- **Node-based scaling** – CPA scales replicas according to the number of cluster nodes that can be scheduled, enabling applications to expand or contract in proportion to the size of the cluster.
- **Proportionate adjustment** – To ensure that the application can scale in accordance with changes in the cluster size, the autoscaler establishes a proportionate relationship between the number

of nodes and the number of replicas. This relationship is used to compute the desired number of replicas for a workload.

- **Integration with Kubernetes components** – CPA works with standard Kubernetes components like the Horizontal Pod Autoscaler (HPA) but focuses specifically on node count rather than resource utilization metrics. This integration allows for a more comprehensive scaling strategy.
- **Golang API clients** – To monitor the number of nodes and their available cores, CPA uses Golang API clients that run inside of pods and talk to the Kubernetes API server.
- **Configurable parameters** – Using a `ConfigMap`, users can set thresholds and scaling parameters that CPA uses to modify its behavior and make sure it follows the intended scaling plan.

## Kubernetes-based Event-Driven Autoscaler

Kubernetes-based Event Driven Autoscaler (KEDA) is an open-source project that enables Kubernetes workloads to scale based on the number of events that need to be processed. KEDA enhances the scalability of applications by allowing them to respond dynamically to varying workloads, particularly those that are event-driven.

By automating the scaling process based on events, KEDA helps organizations optimize resource utilization, improve application performance, and reduce costs associated with over-provisioning. This approach is especially valuable for applications that experience varying traffic patterns, such as microservices, serverless functions, and real-time data processing systems.

KEDA provides the following key features:

- **Event-driven scaling** – KEDA allows you to define scaling rules based on external event sources, such as message queues, HTTP requests, or custom metrics. This capability helps make sure that applications scale in response to real-time demand.
- **Lightweight component** – KEDA is a single-purpose, lightweight component that doesn't require a lot of setup or overhead to be readily integrated into existing Kubernetes clusters.
- **Integration with Kubernetes** – KEDA extends the capabilities of Kubernetes-native components, such as the Horizontal Pod Autoscaler (HPA). KEDA adds event-driven scaling capabilities to these components, enhancing rather than replacing them.
- **Support for multiple event sources** – KEDA is compatible with a wide range of event sources, including popular messaging platforms like RabbitMQ, Apache Kafka, and others. Because of this adaptability, you can customize scaling to fit your unique event-driven architecture.

- **Custom scalers** – Using custom scalers, you can designate specific metrics that KEDA can use to initiate scaling actions in response to specific business logic or requirements.
- **Declarative configuration** – In line with Kubernetes principles, you can use KEDA to describe scaling behavior declaratively by using Kubernetes custom resources to define how scaling should happen.

# Network scaling

Network scaling in Kubernetes is critical for maintaining seamless communication between services and supporting efficient data flow in dynamic environments. Scaling the network infrastructure helps to make sure that the cluster can handle varying levels of traffic without experiencing bottlenecks or latency issues. Kubernetes provides tools and mechanisms to scale network resources, allowing organizations to maintain optimal performance as traffic patterns change.

This elasticity in network scaling enhances the overall user experience by ensuring fast and reliable connections. Network scaling also optimizes the use of network resources, helping to reduce the costs associated with underutilized or overburdened network components.

Furthermore, effective network scaling is vital for supporting high availability and resilience. By dynamically adjusting network capacity and routing, organizations can ensure that services remain accessible and responsive even during periods of peak demand or unexpected traffic spikes. This approach allows for better utilization of cloud networking resources, ensuring that the infrastructure is always aligned with current requirements.

This section discusses the following types of network scaling:

- Amazon VPC CNI plugin for Kubernetes
- Custom networking
- Prefix delegation
- Amazon VPC Lattice

# Amazon VPC CNI plugin for Kubernetes

The Amazon VPC Container Network Interface (CNI) plugin for Kubernetes is a critical component in Amazon EKS. The VPC CNI plugin provides advanced networking capabilities by integrating Kubernetes pods with Amazon VPC. With this plugin, each pod is assigned a unique IP address from the virtual private cloud (VPC), thereby enhancing network isolation and performance. As clusters grow and network demands fluctuate, the Amazon VPC CNI plugin plays a key role in ensuring efficient and scalable network operations.

The plugin automatically manages the allocation and routing of IP addresses within the VPC, simplifying network management and reducing the risk of IP conflicts. It supports features like prefix delegation, which allows for more flexible IP management.

The VPC CNI Plugin helps organizations optimize network performance, enhance security, and reduce the risk of IP exhaustion. These capabilities are especially valuable for large-scale, dynamic environments where network demands fluctuate, such as microservices architectures, high-density workloads, and multi-tenant applications.

The Amazon VPC CNI Plugin provides the following key features:

- **Enhanced networking** – The VPC CNI plugin allows each pod to receive its own IP address directly from the VPC, providing strong isolation and network performance. This approach is crucial for workloads requiring high network throughput and low latency.
- **Prefix delegation** – To overcome IP address exhaustion issues in large clusters, prefix delegation dynamically allocates larger blocks of IPs to nodes, which are then subdivided for pod use. This approach ensures efficient IP utilization and simplifies network scaling.
- **Custom networking** – Users can configure custom network interfaces (ENIs) for pods, which helps distribute pod traffic across multiple interfaces, reducing network congestion and improving scalability.
- **Support for IPv6** – By enabling IPv6 in Amazon EKS clusters, users can significantly expand the available IP address space, facilitating the scaling of large, distributed applications without the constraints of IPv4 limitations.
- **Integration with Kubernetes** – The VPC CNI Plugin works seamlessly with Kubernetes networking components, ensuring that IPs are managed efficiently across pods, services, and external endpoints, and it supports advanced features like security groups for pods.

# Custom networking

Custom networking in Amazon EKS enables the assignment of specific network interfaces to pods, providing enhanced control over IP address management and network traffic. This approach is especially useful in scenarios where IP address exhaustion is a concern or when there is a need to segregate network traffic for security, compliance, or performance reasons. Custom networking helps organizations efficiently manage IP address space, segregate traffic, and ensure scalable network performance.

With custom networking, administrators can manage network resources more efficiently. Admins can use custom networking to help make sure that pods have the necessary network isolation and that the cluster can scale without encountering IP address limitations.

Custom networking provides the following key features:

- **Enhanced IP management** – Custom networking allows the assignment of specific network interfaces (ENIs) to pods, helping to manage IP address exhaustion by distributing pod traffic across multiple ENIs. This capability is particularly important in clusters with high-density workloads.
- **Traffic segregation** – With custom network interfaces, you can separate pod traffic based on specific criteria, such as application type or security requirements. This approach provides greater control over how traffic flows within and outside the cluster.
- **Support for IPv6** – Custom networking in Amazon EKS also supports IPv6, offering a solution to the limitations of IPv4 addresses. The network can scale efficiently without IP address conflicts, even in large-scale deployments.
- **Scalability and flexibility** – As the cluster scales, custom networking enables dynamic management of network interfaces. New pods are assigned appropriate network resources without manual intervention. This approach helps maintain a flexible and scalable network environment that can adapt to changing workloads.

## Prefix delegation

Prefix delegation in Kubernetes, particularly within Amazon EKS, is designed to streamline and optimize IP address management as clusters scale. By dynamically allocating larger blocks of IP addresses (prefixes) to nodes, prefix delegation reduces the risk of IP exhaustion and simplifies the management of IP space.

This approach enhances network efficiency, minimizes fragmentation, and helps clusters scale smoothly without manual IP range adjustments. Prefix delegation is particularly valuable for large-scale deployments, high-density workloads, and environments where flexible, dynamic IP management is critical to maintaining network performance and scalability.

Prefix delegation provides the following key features:

- **Efficient IP address management** – Prefix delegation allows for dynamic allocation of IP ranges, reducing the risk of IP exhaustion and ensuring efficient use of available IP space.
- **Simplified network management** – By allowing nodes to handle their own IP allocations, prefix delegation minimizes network fragmentation and simplifies the routing process, making it easier to scale clusters as needed.

- **Support for large-scale deployments** – In large clusters with high-density workloads, prefix delegation enables seamless scaling by allowing new nodes to join the cluster without manual IP range adjustments.

## Amazon VPC Lattice

Amazon VPC Lattice enables efficient and secure service-to-service communication within and across VPCs, particularly in microservices architectures. VPC Lattice uses security measures like security groups and network access control lists (network ACLs) in addition to AWS Identity and Access Management (IAM) integration for fine-grained application authentication. A layer-7 proxy service at the heart of VPC Lattice offers connection, load balancing, authentication, authorization, observability, traffic management, and service discovery.

By simplifying networking and security configurations, VPC Lattice helps organizations optimize traffic management, enhance application performance, and scale seamlessly across multiple VPCs and AWS Regions. This is especially valuable for distributed applications that require consistent and reliable networking, such as microservices, cross-Region deployments, and complex cloud-native environments.

Amazon VPC Lattice provides the following key features:

- **Service-to-service networking** – VPC Lattice simplifies the networking and security configuration between services within a microservices architecture. It provides a unified platform for managing communication, so that services can scale independently while maintaining high performance and security.
- **Cross-VPC networking** – VPC Lattice is crucial for managing traffic across multiple VPCs or Regions. It provides a consistent networking framework that allows services to communicate seamlessly, regardless of their physical location. This capability is particularly important for large-scale applications that span multiple VPCs or geographic Regions.
- **Enhanced security management** – By integrating security policies directly into the network layer, VPC Lattice supports service-to-service communication that's both secure and efficient. This feature reduces the complexity of managing security across a distributed environment, allowing for easier scaling and reduced operational overhead.
- **Simplified traffic management** – VPC Lattice offers advanced traffic management features, including routing, load balancing, and failover mechanisms. With these features, traffic is

distributed efficiently across services, optimizing network performance and enhancing the scalability of the application.

# Cost optimization

To support effective resource control, Kubernetes cost minimization is crucial for enterprises using this container orchestration technology. It's difficult to properly track spending in Kubernetes settings because of their complexity, which include multiple components such as pods and nodes. Through the application of cost optimization techniques, businesses can see where their resources are being spent and appropriately assign expenses to departments or projects.

Although dynamic scaling has advantages, if not properly managed, it can result in unforeseen expenses. Efficient cost management helps to allocate resources only when they're truly required, averting unanticipated surges in expenditures.

This section discusses the following approaches to cost optimization:

- Kubecost
- Goldilocks
- AWS Fargate
- Spot Instances
- Reserved Instances
- AWS Graviton instances

## Kubecost

Kubecost is a cost management solution that helps businesses track, control, and maximize their expenditure on cloud infrastructure. It's made specifically for Kubernetes clusters. Kubecost gives you insights into resource utilization and real-time cost awareness, enabling you to better understand where and how much of your cloud resources are being used. With these insights, you can optimize your infrastructure spending, improve resource efficiency, and make more informed decisions about your cloud investments.

Kubecost provides the following key features:

- **Cost allocation** – Kubecost offers thorough cost allocation for Kubernetes resources, including workloads, services, namespaces, and labels. This feature helps teams to monitor costs by environment, project, or team.

- **Real-time cost monitoring** – It offers real-time monitoring of cloud costs, giving organizations immediate insights into spending patterns and helping to prevent unexpected cost overruns.
- **Optimization recommendations** – Kubecost offers practical suggestions for minimizing resource utilization, including reducing idle resources, right-sizing workloads, and maximizing storage expenses.
- **Budgeting and alerts** – Kubecost users can create budgets and receive reminders when an expenditure approaches or surpasses predetermined criteria. This feature helps teams adhere to financial constraints.

## Goldilocks

Goldilocks is a Kubernetes utility designed to help users optimize their resource requests and limits for Kubernetes workloads. It provides recommendations on how to configure the CPU and memory resources for containers running in a Kubernetes cluster. These recommendations help you make sure that applications have the right number of resources to perform efficiently without waste. This optimization can lead to cost savings, improved performance, and more efficient use of Kubernetes clusters.

Goldilocks provides the following key features:

- **Resource recommendations** – Goldilocks determines the ideal settings for resource requests and restrictions by analyzing past CPU and memory consumption statistics for Kubernetes workloads. By doing this, it becomes easier to avoid under- or over-provisioning, which can result in performance problems and resource waste.
- **VPA integration** – Goldilocks leverages the Kubernetes Vertical Pod Autoscaler (VPA) to gather data and provide recommendations. It runs in a "recommendation mode," meaning it doesn't actually change resource settings but offers guidance on what those settings should be.
- **Namespace-based analysis** – Goldilocks gives you the ability to finely regulate which workloads are optimized and monitored by allowing you to target particular namespaces for analysis.
- **Visual dashboard** – The web-based dashboard displays suggested resource requests and restrictions visually, which makes it straightforward for you to understand and take action on the data.
- **Non-intrusive operation** – Goldilocks doesn't alter the cluster's setup because it operates in recommendation mode. If you want, you can manually apply the recommended resource settings after reviewing the recommendations.

# AWS Fargate

In the context of Amazon EKS, https://docs.aws.amazon.com/eks/latest/userguide/fargate.htmlAWS Fargate allows you to run Kubernetes pods without managing the underlying Amazon EC2 instances. It's a serverless compute engine that lets you focus on deploying and scaling containerized applications without worrying about the infrastructure.

AWS Fargate provides the following key features:

- **No infrastructure management** – Fargate eliminates the need to provision, manage, or scale Amazon EC2 instances or Kubernetes nodes. AWS handles all the infrastructure management, including patching and scaling.
- **Pod-level isolation** – Unlike worker nodes that are based on Amazon EC2, Fargate provides task or pod-level isolation. Each pod runs in its own isolated compute environment, which enhances security and performance.
- **Automatic scaling** – Fargate automatically scales Kubernetes pods based on demand. You don't need to manage scaling policies or node pools.
- **Per-second billing** – You only pay for the vCPU and memory resources consumed by each pod for the exact duration it runs, which is a cost-effective option for certain workloads.
- **Reduced overhead** – By eliminating the need to manage EC2 instances, Fargate allows you to focus on building and managing your applications rather than infrastructure operations.

# Spot Instances

Spot Instances offer significant savings over On-Demand Instance pricing and are an affordable option for running Amazon EC2 worker nodes in an Amazon EKS cluster. However, AWS can interrupt Spot Instances in the event that On-Demand Instance capacity is needed. AWS can reclaim Spot Instances with a 2-minute notice when the capacity is needed, making them less reliable for critical, stateful workloads.

For workloads that are sensitive to cost and can withstand disruptions, Spot Instances in Amazon EKS are a good option. Using a combination of Spot Instances and On-Demand Instances in a Kubernetes cluster helps you save money without sacrificing availability for vital workloads.

Spot Instances provides the following key features:

- **Cost savings** – Spot Instances can be less expensive than On-Demand Instance pricing, making them ideal for cost-sensitive workloads.
- **Ideal for fault-tolerant workloads** – Well suited for stateless, fault-tolerant workloads such as batch processing, CI/CD jobs, machine learning, or large-scale data processing where instances can be replaced without major disruption.
- **Auto-Scaling group integration** – Amazon EKS integrates Spot Instances with Kubernetes Cluster Autoscaler, which can automatically replace interrupted Spot Instance nodes with other available Spot Instances or On-Demand Instances.

## Reserved Instances

In Amazon EKS, Reserved Instances is a pricing model for the Amazon EC2 worker nodes that run your Kubernetes workloads. By using Reserved Instances, you commit to using specific instance types for a 1- year or 3-year term, in exchange for cost savings compared to On-Demand Instance pricing. Reserving instances in Amazon EKS is an affordable way to perform consistent, long-term workloads on Amazon EC2 worker nodes.

Reserved Instances are commonly used for Amazon EC2. However, the worker nodes in your Amazon EKS cluster (which are EC2 instances) can also benefit from this cost-saving model, provided the workload requires long-term, predictable usage.

Production services, databases, and other stateful applications that need high availability and consistent performance are examples of stable workloads that are well suited for Reserved Instances.

Reserved Instances provides the following key features:

- **Cost savings** – Reserved Instances offer savings compared to On-Demand instances, depending on the term length (1 or 3 years) and payment plan (All Upfront, Partial Upfront, or No Upfront).
- **Long-term commitment** – You commit to a 1-year or 3-year term for a specific instance type, size, and AWS Region. This is ideal for workloads that are stable and run continuously over time.
- **Predictable pricing** – Because you're committed to a specific term, Reserved Instances provide predictable monthly or upfront costs, making it easier to budget for long-term workloads.
- **Instance flexibility** – With Convertible Reserved Instances, you can change the instance type, family, or size during the reservation period. Convertible Reserved Instances offer more flexibility than Standard Reserved Instances, which don't allow changes.

- **Guaranteed capacity** – Reserved Instances ensure that capacity is available in the Availability Zone where the reservation is made, which is crucial for critical workloads that need consistent compute power.
- **No interruption risk** – Unlike Spot Instances, Reserved Instances are not subject to interruption by AWS. This makes them ideal for running mission-critical workloads that require guaranteed uptime.

## AWS Graviton instances

AWS Graviton is a family of ARM-based processors designed by AWS to provide improved performance and cost-efficiency for cloud workloads. In the context of Amazon EKS, you can use Graviton instances as worker nodes to run your Kubernetes workloads, offering significant performance gains and cost savings.

Graviton instances are an excellent option for cloud-native and compute-intensive applications because they offer a higher price-performance ratio than x86 instances. However, when you consider adopting Graviton instances, take ARM compatibility into account.

AWS Graviton instances provide the following key features:

- **ARM-based architecture** – AWS Graviton processors are built on ARM architecture, which is different from traditional x86 architectures but highly efficient for many workloads.
- **Cost-efficient** – Amazon EC2 instances based on Graviton typically offer better price-performance compared to x86-based EC2 instances. This makes them an attractive option for Kubernetes clusters that run Amazon EKS.
- **Performance** – Graviton2 processors, the second generation of AWS Graviton, offer significant improvements in terms of compute performance, memory throughput, and energy efficiency. They're ideal for CPU-intensive and memory-intensive workloads.
- **Diverse instance types** – Graviton instances come in various families, such as t4g, m7g, c7g, and r7g, covering a range of use cases from general purpose to compute-optimized, memory-optimized, and burstable workloads.
- **Amazon EKS node groups** – You can configure node groups that are managed by Amazon EKS or self-managed node groups to include Graviton-based instances. With this approach, you can run workloads that are optimized for ARM architecture on the same Kubernetes cluster alongside x86-based instances.

# Next steps

This guide provides information to help you optimize Amazon EKS with respect to compute scaling, workload scaling, network scaling, and cost optimization. By understanding and applying these concepts, organizations can achieve a highly efficient, scalable, and cost-effective cloud environment that meets their dynamic needs.

Effective implementation of compute and workload scaling helps make sure that resources are used efficiently and applications maintain high performance even during peak times. Embracing network scaling techniques, such as custom networking and prefix delegation, supports management of network resources and seamless scalability. Emphasizing cost optimization helps organizations balance performance with financial efficiency.

Integrating this guidance into your cloud strategy can help you enhance your infrastructure's performance and scalability and drive cost savings. This comprehensive approach can enable you to build a robust cloud environment that supports your organization's growth and adapts to ever-changing business demands.

# Resources

### AWS blogs

- Building for Cost optimization and Resilience for EKS with Spot Instances
- Mixing AWS Graviton with x86 CPUs to optimize cost and resiliency using Amazon EKS

### AWS documentation

- Amazon VPC CNI
- Amazon Elastic Kubernetes Service (AWS whitepaper: Overview of Deployment Options on AWS)
- Amazon EKS Best Practices Guide
- Karpenter
- Learn more about Kubecost
- Simplify compute management with AWS Fargate

### Other resources

- Cluster Autoscaling (Kubernetes documentation)
- Goldilocks: An Open Source Tool for Recommending Resource Requests (Fairwinds Blog)
- Horizontal Pod Autoscaling (Kubernetes documentation)
- Kubecost (Kubecost documentation)
- Kubernetes Event-driven Autoscaling (KEDA documentation)

# Document history

The following table describes significant changes to this guide, Scaling Amazon EKS infrastructure to optimize compute, workloads, and network performance. If you want to be notified about future updates, you can subscribe to an RSS feed.

| Change | Description | Date |
| --- | --- | --- |
| Initial publication | — | November 11, 2024 |

# AWS Prescriptive Guidance glossary

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

## Numbers

7 Rs

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

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

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

# A

A2A (Agent-to-Agent)

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

ABAC

See attribute-based access control.

abstracted services

See managed services.

ACID

See atomicity, consistency, isolation, durability.

active-active migration

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

active-passive migration

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

Agent

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

Agent Ops

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

aggregate function

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

AI

See artificial intelligence.

AIOps

See artificial intelligence operations.

anonymization

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

anti-pattern

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

application control

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

application portfolio

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

artificial intelligence (AI)

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

artificial intelligence operations (AIOps)

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

asymmetric encryption

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

atomicity, consistency, isolation, durability (ACID)

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

attribute-based access control (ABAC)

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

authoritative data source

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

Availability Zone

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

AWS Cloud Adoption Framework (AWS CAF)

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

AWS Workload Qualification Framework (AWS WQF)

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

# B

bad bot

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

BCP

See business continuity planning.

behavior graph

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

big-endian system

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

binary classification

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

bloom filter

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

blue/green deployment

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

bot

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

botnet

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

branch

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

break-glass access

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

brownfield strategy

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

buffer cache

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

business capability

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

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

business continuity planning (BCP)

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

# C

CAF

See AWS Cloud Adoption Framework.

canary deployment

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

CCoE

See Cloud Center of Excellence.

CDC

See change data capture.

change data capture (CDC)

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

chaos engineering

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

CI/CD

See continuous integration and continuous delivery.

classification

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

Citizen Developer

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

client-side encryption

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

Cloud Center of Excellence (CCoE)

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

cloud computing

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

cloud operating model

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

cloud stages of adoption

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

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

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

CMDB

See configuration management database.

code repository

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

cold cache

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

cold data

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

computer vision (CV)

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

configuration drift

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

configuration management database (CMDB)

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

conformance pack

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

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

continuous integration and continuous delivery (CI/CD)

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

CV

See computer vision.

# D

data at rest

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

data classification

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

data drift

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

data in transit

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

data mesh

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

data minimization

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

data perimeter

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

data preprocessing

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

data provenance

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

data subject

An individual whose data is being collected and processed.

data warehouse

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

database definition language (DDL)

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

database manipulation language (DML)

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

DDL

See database definition language.

deep ensemble

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

deep learning

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

defense-in-depth

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

delegated administrator

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

deployment

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

development environment

See environment.

detective control

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

development value stream mapping (DVSM)

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

digital twin

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

dimension table

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

disaster

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

disaster recovery (DR)

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

DML

See database manipulation language.

domain-driven design

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

DR

See disaster recovery.

drift detection

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

DVSM

See development value stream mapping.

# E

EDA

See exploratory data analysis.

EDI

See electronic data interchange.

edge computing

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

electronic data interchange (EDI)

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

encryption

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

encryption key

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

endianness

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

endpoint

See service endpoint.

endpoint service

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

enterprise resource planning (ERP)

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

envelope encryption

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

environment

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

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

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

epic

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

ERP

See enterprise resource planning.

exploratory data analysis (EDA)

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

# F

fact table

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

fail fast

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

fault isolation boundary

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

feature branch

See branch.

features

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

feature importance

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

feature transformation

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

few-shot prompting

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

FGAC

See fine-grained access control.

fine-grained access control (FGAC)

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

flash-cut migration

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

FM

See foundation model.

foundation model (FM)

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

FM gateway

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

# G

generative AI

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

geo blocking

See geographic restrictions.

geographic restrictions (geo blocking)

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

Gitflow workflow

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

golden image

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

greenfield strategy

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

guardrail

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

guardrails (AI)

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

## H

HA

See high availability.

heterogeneous database migration

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

high availability (HA)

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

historian modernization

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

holdout data

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

human-in-the-loop (HitL)

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

homogeneous database migration

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

hot data

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

hotfix

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

hypercare period

Immediately following cutover, the period of time when a migration team manages and monitors the migrated applications in the cloud in order to address any issues. Typically, this

period is 1–4 days in length. At the end of the hypercare period, the migration team typically transfers responsibility for the applications to the cloud operations team.

## I

IaC

See infrastructure as code.

identity-based policy

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

idle application

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

IIoT

See industrial Internet of Things.

immutable infrastructure

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

inbound (ingress) VPC

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

incremental migration

A cutover strategy in which you migrate your application in small parts instead of performing a single, full cutover. For example, you might move only a few microservices or users to the new system initially. After you verify that everything is working properly, you can incrementally

move additional microservices or users until you can decommission your legacy system. This strategy reduces the risks associated with large migrations.

Industry 4.0

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

infrastructure

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

infrastructure as code (IaC)

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

industrial Internet of Things (IIoT)

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

inspection VPC

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

Internet of Things (IoT)

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

interpretability

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

IoT

See Internet of Things.

IT information library (ITIL)

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

IT service management (ITSM)

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

ITIL

See IT information library.

ITSM

See IT service management.

# L

label-based access control (LBAC)

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

landing zone

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

large language model (LLM)

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

large migration

A migration of 300 or more servers.

LBAC

See label-based access control.

least privilege

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

lift and shift

See 7 Rs.

little-endian system

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

LLM

See large language model.

lower environments

See environment.

# M

machine learning (ML)

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

main branch

See branch.

malware

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

managed services

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

manufacturing execution system (MES)

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

MAP

See Migration Acceleration Program.

MCP

See Model Context Protocol.

Model Context Protocol (MCP)

A stateless protocol for agent-to-tool communication.

MCP server

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

mechanism

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

member account

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

MES

See manufacturing execution system.

Message Queuing Telemetry Transport (MQTT)

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

microservice

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

microservices architecture

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

Migration Acceleration Program (MAP)

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

migration at scale

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

migration factory

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

migration metadata

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

migration pattern

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

Migration Portfolio Assessment (MPA)

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

Migration Readiness Assessment (MRA)

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

migration strategy

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

ML

See machine learning.

modernization

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

modernization readiness assessment

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

monolithic applications (monoliths)

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

MPA

See Migration Portfolio Assessment.

MQTT

See Message Queuing Telemetry Transport.

multiclass classification

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

mutable infrastructure

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

# O

OAC

See origin access control.

OAI

See origin access identity.

OCM

See organizational change management.

offline migration

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

OI

See operations integration.

OLA

See operational-level agreement.

online migration

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

OPC-UA

See Open Process Communications - Unified Architecture.

Open Process Communications - Unified Architecture (OPC-UA)

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

operational-level agreement (OLA)

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

operational readiness review (ORR)

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

operational technology (OT)

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

operations integration (OI)

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

organization trail

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

organizational change management (OCM)

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

origin access control (OAC)

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

origin access identity (OAI)

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

ORR

See operational readiness review.

OT

See operational technology.

outbound (egress) VPC

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

# P

permissions boundary

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

personally identifiable information (PII)

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

PII

See personally identifiable information.

playbook

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

PLC

See programmable logic controller.

PLM

See product lifecycle management.

policy

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

polyglot persistence

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

portfolio assessment

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

predicate

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

predicate pushdown

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

preventative control

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

principal

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

privacy by design

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

private hosted zones

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

proactive control

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

product lifecycle management (PLM)

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

production environment

See environment.

programmable logic controller (PLC)

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

prompt chaining

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

pseudonymization

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

publish/subscribe (pub/sub)

A pattern that enables asynchronous communications among microservices to improve scalability and responsiveness. For example, in a microservices-based MES, a microservice can

publish event messages to a channel that other microservices can subscribe to. The system can add new microservices without changing the publishing service.

# Q

query plan

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

query plan regression

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

# R

RACI matrix

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

RAG

See Retrieval Augmented Generation.

ransomware

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

RASCI matrix

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

RCAC

See row and column access control.

read replica

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

re-architect

See 7 Rs.

recovery point objective (RPO)

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

recovery time objective (RTO)

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

refactor

See 7 Rs.

Region

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

regression

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

rehost

See 7 Rs.

release

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

relocate

See 7 Rs.

replatform

See 7 Rs.

repurchase

See 7 Rs.

resiliency

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

resource-based policy

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

responsible, accountable, consulted, informed (RACI) matrix

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

responsive control

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

retain

See 7 Rs.

retire

See 7 Rs.

Retrieval Augmented Generation (RAG)

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

rotation

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

row and column access control (RCAC)

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

RPO

See recovery point objective.

RTO

See recovery time objective.

runbook

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

# S

SAML 2.0

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

SCADA

See supervisory control and data acquisition.

SCP

See service control policy.

secret

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

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

security by design

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

security control

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

security hardening

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

security information and event management (SIEM) system

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

security response automation

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

server-side encryption

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

service control policy (SCP)

A policy that provides centralized control over permissions for all accounts in an organization in AWS Organizations. SCPs define guardrails or set limits on actions that an administrator can delegate to users or roles. You can use SCPs as allow lists or deny lists, to specify which services

or actions are permitted or prohibited. For more information, see Service control policies in the AWS Organizations documentation.

service endpoint

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

service-level agreement (SLA)

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

service-level indicator (SLI)

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

service-level objective (SLO)

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

shared responsibility model

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

Shadow AI

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

SIEM

See security information and event management system.

single point of failure (SPOF)

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

SLA

See service-level agreement.

SLI

See service-level indicator.

SLO

See service-level objective.

split-and-seed model

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

SPOF

See single point of failure.

star schema

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

strangler fig pattern

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

subnet

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

supervisory control and data acquisition (SCADA)

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

symmetric encryption

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

synthetic testing

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

system prompt

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

## T

tags

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

target variable

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

task list

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

test environment

See environment.

training

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

tool

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

transit gateway

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

trunk-based workflow

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

trusted access

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

tuning

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

two-pizza team

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

# U

uncertainty

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

undifferentiated tasks

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

upper environments

See environment.

# V

vacuuming

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

version control

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

VPC peering

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

vulnerability

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

# W

warm cache

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

warm data

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

window function

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

workload

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

workstream

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

WORM

See write once, read many.

WQF

See AWS Workload Qualification Framework.

write once, read many (WORM)

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

# Z

zero-day exploit

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

zero-day vulnerability

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

zero-shot prompting

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

zombie application

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