Best practices for streamlining Amazon EKS observability

AWS Prescriptive Guidance

AWS Prescriptive Guidance: Best practices for streamlining Amazon EKS observability

Table of Contents

Best practices for streamlining Amazon EKS observability

Ishwar Chauthaiwale, Naveen Suthar, and Pratap Kumar Nanda, Amazon Web Services (AWS)

March 2026 (document history)

Amazon Elastic Kubernetes Service (Amazon EKS) requires comprehensive observability solutions to monitor and troubleshoot containerized workloads effectively. Distributed systems and microservices have complex architectures in Amazon EKS environments, so implementing proper observability practices is crucial for maintaining reliable operations. Effective observability in Amazon EKS environments enables teams to gain deep insights into application performance, troubleshoot issues efficiently, and maintain optimal cluster health.

The challenge lies in navigating the vast ecosystem of tools and techniques available for Amazon EKS observability while adhering to best practices that align with organizational goals and industry standards. Effective observability strategies must balance comprehensive data collection with performance considerations, cost-effectiveness, and scalability.

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

- Establishing efficient **logging** mechanisms
- Implementing robust **monitoring** solutions
- Using distributed **tracing** for complex architectures
- Implementing **alerting** and incident response strategies

By adopting these best practices, your organization can enhance their ability to gain deep insights into their Amazon EKS environment, which leads to improved reliability, performance, and operational efficiency. This streamlined approach to observability aids in troubleshooting and maintenance, and supports data-driven decision-making for continuous improvement of Kubernetes-based applications and infrastructure. (For detailed information about Amazon EKS, see the service documentation.)

This guide dives deep into each aspect of Amazon EKS observability and explores the tools and strategies that you can tailor to meet the specific needs of your Amazon EKS deployments, from small-scale applications to large, complex microservices architectures.

In this guide:

Objectives

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

- **Enhanced operational visibility** – Achieve comprehensive insight into your Amazon EKS clusters and applications through effective observability practices.

  This objective emphasizes the importance of maintaining complete visibility across your Amazon EKS environment. Tools such as AWS X-Ray, Amazon CloudWatch Container Insights, and AWS Distro for OpenTelemetry help you understand system behavior, identify issues quickly, and maintain optimal performance.
- **Improved troubleshooting efficiency** – Reduce mean time to detection (MTTD) and mean time to resolution (MTTR) through effective tracing and monitoring strategies.

  This objective focuses on implementing observability practices that enable quick identification and resolution of issues. Techniques such as distributed tracing, effective logging, and comprehensive metrics collection are key to achieving this objective.
- **Proactive performance management** – Enable early detection of potential issues before they affect end users.

  Proactive monitoring is crucial for maintaining high service availability and performance. This objective addresses the importance of implementing proper alerting, trend analysis, and predictive monitoring to prevent service disruptions.
- **Cost-effective observability** – Optimize observability costs while maintaining comprehensive system visibility.

Cost optimization encompasses implementing efficient sampling strategies, appropriate data retention policies, and optimal instrumentation approaches. The goal is to balance observability needs with cost considerations while ensuring effective system monitoring.

- **Scalable monitoring architecture** – Make sure that your observability solutions scale seamlessly with your Amazon EKS environment.

  This objective focuses on implementing monitoring solutions that can grow with your application. Whether you're running a single cluster or a multi-cluster, multi-Region deployment, your observability strategy should scale accordingly

Logging in Amazon EKS

Logging is a critical aspect of managing and maintaining applications that run on Amazon EKS. Effective logging practices in Amazon EKS environments help developers, operations teams, and system administrators gain valuable insights into the behavior, performance, and health of their containerized applications and their underlying infrastructure.

Implementing a robust logging strategy in Amazon EKS is essential for several reasons:

- **Troubleshooting**: Logs help identify and diagnose issues quickly, which reduces downtime and improves overall system reliability.
- **Compliance**: Many industries require comprehensive logging for auditing and regulation purposes.
- **Security**: Log analysis can help you detect and investigate potential security threats or breaches.
- **Performance optimization**: Logs provide insights into application and system performance, so you can identify bottlenecks and optimize resource utilization.
- **Monitoring and alerting**: Log data can be used to set up monitoring systems and trigger alerts for specific events or conditions.

In this section:

Types of logging in Amazon EKS

In Amazon EKS, logging involves capturing, storing, and analyzing various types of log data that's generated by different components of the Kubernetes cluster, including:

- **System logs**: Information about the underlying Amazon Elastic Compute Cloud (Amazon EC2) instances or AWS Fargate nodes
- **Kubernetes component logs**: Data from core Kubernetes components such as the API server, scheduler, and controller manager
- **Container runtime logs**: Information from the container runtime, such as Docker or containerd

- **Application logs**: Output from containerized applications

To manage logs in your Amazon EKS environment effectively, you typically employ a combination of AWS services, third-party tools, and best practices. This might include using Amazon CloudWatch, Fluent Bit, Elasticsearch, Kibana, and other logging and analysis tools to collect, store, and visualize log data.

The following sections explore various aspects of logging in Amazon EKS, including best practices, tools, and techniques for implementing a comprehensive logging strategy in your Kubernetes clusters on AWS.

System logs

Logging for underlying EC2 instances or Fargate nodes in Amazon EKS involves different approaches depending on the node type.

To implement logging for EC2 instances in Amazon EKS, you can use the following tools:

- CloudWatch agent: Install and configure the CloudWatch agent on your EC2 instances. Configure it to collect system logs such as `/var/log/messages` and `/var/log/secure`. You can use user data scripts or configuration management tools to automate this process.
- Fluent Bit: Deploy Fluent Bit as a DaemonSet to collect logs from all nodes. Configure it to forward logs to CloudWatch Logs or other centralized logging systems.
- Container Insights: Enable Container Insights in your EKS cluster to automatically collect metrics and logs from EC2 instances.
- Custom scripts: Develop custom scripts to collect specific logs and send them to your preferred logging destination.
- SSM Agent: Use AWS Systems Manager Agent (SSM Agent) to collect and forward logs to CloudWatch Logs.

To implement logging for Fargate nodes in Amazon EKS, use these tools:

- Fargate logging: Fargate automatically collects `stdout` and `stderr` logs from your containers. Configure your Fargate profile to send these logs to CloudWatch Logs.
- Fluent Bit for Fargate: AWS provides a Fluent Bit image specifically for Fargate logging. Deploy it as a sidecar container in your Fargate pods to collect and forward logs.

- Container Insights for Fargate: Enable Container Insights to collect metrics and logs from Fargate nodes.

Kubernetes component logs

Collecting logs from Kubernetes components such as the API server, scheduler, and controller manager in Amazon EKS requires a slightly different approach from application logging. These components run as part of the Amazon EKS control plane, which is managed by AWS. Here's how you can collect and access these logs:

- **Enable control plane logging:** You can enable control plane logging for your EKS cluster through the AWS Management Console, AWS Command Line Interface (AWS CLI), or infrastructure as code (IaC) tools such as AWS CloudFormation or Terraform. When you enable control plane logging, the logs are sent to Amazon CloudWatch Logs. You can view them in the CloudWatch console in the `/aws/eks/<cluster-name>/cluster` log group. Within this log group, each control plane component has its own log stream as follows:

Stream name	Description
kube-apiserver	Kubernetes API server logs
kube-scheduler	Scheduler decision logs
kube-controller-manager	Controller manager logs
authenticator	IAM authenticator logs
audit	Kubernetes audit logs (must be explicitly enabled)

  To view logs for a specific component, navigate to the cluster log group and filter by the target log stream name.

- **Use CloudWatch Logs Insights**: You can use CloudWatch Logs Insights to perform complex queries on your logs.
- **Export logs to Amazon S3**: For long-term storage or further analysis, you can export logs to Amazon Simple Storage Service (Amazon S3).

- **Use third-party tools**: You can use tools such as Fluent Bit to collect these logs and forward them to other logging systems such as Elasticsearch or Splunk.
- **Use AWS CloudTrail**: The AWS CloudTrail service can provide additional insights into API calls made to your EKS cluster.

Container runtime logs

Logging container runtime logs in Amazon EKS involves capturing and managing logs from the container runtime, which is typically `containerd` for Amazon EKS. Here's how you can approach logging container runtime logs in Amazon EKS:

- Directly access the logs on Amazon EC2 nodes. For self-managed EC2 nodes, you can directly access the container runtime logs on the host from these locations:
  - `containerd` logs: `/var/log/containers/`
  - Docker logs (if you're using the Docker runtime): `/var/log/docker.log`
- Use a DaemonSet for log collection.
- Deploy a log collection agent (such as Fluent Bit) as a DaemonSet to collect logs from all nodes.
- Configure the CloudWatch agent to collect container runtime logs.
- Enable Container Insights to collect container runtime metrics and logs.
- Use Fargate. For Fargate nodes, container runtime logs are automatically collected and can be accessed through CloudWatch Logs.
- Implement custom logging solutions by using tools such as Fluent Bit or Logstash. Set up CloudWatch alarms or use tools such as Prometheus to monitor for specific patterns or issues in container runtime logs. Consider using third-party logging solutions that integrate well with Kubernetes and Amazon EKS, such as Datadog, Splunk, or the Elastic Stack (ELK Stack). Use log aggregation tools to collect logs from multiple sources and forward them to a centralized logging system.

Application logs

Application logs in Amazon EKS are a crucial part of maintaining and troubleshooting your applications. To implement application logging in Amazon EKS, you can choose from these options:

- Write logs to `stdout/stderr`: The simplest and most Kubernetes-native way to handle application logs is to write them to `stdout` and `stderr`. Kubernetes automatically captures these streams.
- Implement log aggregation: Use a log aggregator such as Fluent Bit to collect logs from all your pods.
- Configure log routing: Configure your log aggregator to route logs to your desired destination (such as CloudWatch Logs or Elasticsearch).
- Use CloudWatch Container Insights: Enable Container Insights for comprehensive logging and monitoring.

Best practices for logging in Amazon EKS

The following best practices help create a robust, scalable, and efficient logging system for your Amazon EKS environment, and provide better troubleshooting, monitoring, and overall management of your Kubernetes clusters.

- **Centralize log collection**: Use a centralized logging solution such as CloudWatch Logs, Elasticsearch, or a third-party service to aggregate logs from all components. This provides a single point of access for log analysis and simplifies management.
- **Implement structured logging**: Use structured log formats such as JSON so that logs can be parsed and searched more easily. Include relevant metadata such as timestamps, log levels, and source identifiers.
- **Use log levels appropriately**: Implement proper log levels (such as `DEBUG`, `INFO`, `WARN`, and `ERROR`) in your applications. Configure production environments to log at appropriate levels to avoid excessive logging.
- **Enable container logging**: Configure your containers to log to `stdout` and `stderr`. This allows Kubernetes to capture and forward these logs to your chosen logging solution.
- **Enable application logging**: Configure applications to write logs to `stdout` and `stderr` instead of writing to log files. This follows the 12-factor app methodology and aligns with cloud-native best practices.
- **Use Kubernetes DaemonSets for log collection**: Deploy log collection agents (such as Fluent Bit) as DaemonSets to ensure that they run on every node in your cluster.
- **Implement retention policies**: Define and enforce log retention policies to comply with regulations and to manage storage costs.

- **Secure log data**: Encrypt logs in transit and at rest. Implement access controls to restrict who can view and manage logs.
- **Monitor log ingestion**: Set up alerts for log ingestion failures or delays to ensure continuous logging.
- **Use Kubernetes annotations and labels**: Use Kubernetes annotations and labels to add metadata to your logs, to improve searchability and filtering.
- **Implement distributed tracing**: Use distributed tracing tools such as AWS X-Ray or Jaeger to correlate logs across microservices.
- **Optimize log volume**: Be selective about what you log to avoid unnecessary costs and performance issues. Use sampling for high-volume, low-value logs.
- **Implement log aggregation**: Use tools such as Logstash to aggregate logs from multiple sources before sending them to your central logging system.
- **Use AWS services when possible**: Services such as CloudWatch Logs and Container Insights provide seamless integration with other AWS services.
- **Implement log analysis and visualization**: Use tools such as CloudWatch Logs Insights, Elasticsearch with Kibana, or third-party solutions for log analysis and visualization.
- **Implement automated log analysis**: Use machine learning and AI-powered tools to detect anomalies and patterns in your logs automatically.
- **Document your logging strategy**: Maintain clear documentation of your logging architecture, practices, and tools for your team.

Important considerations for logging in Amazon EKS

This section discusses important considerations to keep in mind when you implement logging in Amazon EKS.

- **Performance impact**: Excessive logging can affect application performance. Be mindful of the volume and frequency of logs generated.
- **Cost management**: Log storage and processing can incur significant costs, especially at scale. Implement log retention policies and consider using log aggregation to reduce costs.
- **Security and compliance**: Make sure that logs don't contain sensitive information such as passwords or personal data. Implement encryption for logs in transit and at rest. Consider compliance requirements such as General Data Protection Regulation (GDPR) or Health Insurance Portability and Accountability Act (HIPAA) when you handle logs.

- **Scalability**: Make sure that your logging solution can scale with your cluster size and log volume. Consider using buffering and batching for log transmission.
- **Log retention**: Define and implement appropriate log retention periods. Balance compliance requirements against storage costs.
- **Access control**: Implement proper AWS Identity and Access Management (IAM) roles and policies for log access. Follow the least privilege principle for log management.
- **Log consistency**: Use consistent log formats across different applications and services. Use structured logging for easier parsing and analysis.
- **Time synchronization**: Synchronize time across all nodes to get consistent timestamps in logs.
- **Resource allocation**: Allocate appropriate resources (such as CPU and memory) for logging agents. Monitor the resource usage of logging components.
- **Fargate considerations**: Fargate has specific logging mechanisms that differ from EC2-based nodes. Understand the limitations and capabilities of Fargate logging.
- **Multi-tenant clusters**: In multi-tenant environments, make sure that logs are properly isolated between tenants.
- **Log parsing and analysis**: Consider the tools and skills required for effective log analysis. Implement log parsing for structured data extraction.
- **Monitoring the logging system**: Set up monitoring for the logging infrastructure itself. Generate alerts for logging system failures or backlogs.
- **Network impact**: Be aware of the network bandwidth used by log transmission. Consider using compression for log data.
- **Kubernetes events**: Don't overlook Kubernetes events as a source of important information.
- **Control plane logging**: Understand the implications and costs of enabling control plane logging.
- **Debugging capabilities**: Make sure that your logging solution allows for easy debugging and troubleshooting.
- **Integration with existing tools**: Consider how your Amazon EKS logging solution integrates with existing monitoring and alerting tools.
- **Testing**: Regularly test your logging setup, especially after cluster upgrades.
- **Documentation**: Maintain clear documentation of your logging architecture and practices.
- **Log aggregation latency**: Be aware of any latency in log aggregation and how it might affect real-time monitoring.

Monitoring in Amazon EKS

Monitoring in Amazon EKS provides critical visibility into the health, performance, and security of your Kubernetes workloads. Without proper monitoring, you risk service disruptions, security breaches, and inefficient resource utilization that can impact business operations and increase costs. Effective monitoring enables you to proactively identify and resolve issues, optimize resource usage, and maintain compliance requirements across your containerized applications. By implementing comprehensive monitoring solutions, you can ensure high availability, detect anomalies early, and make data-driven decisions for scaling and improving your Amazon EKS infrastructure.

This section explores the various aspects of Amazon EKS monitoring, including different monitoring types, available tools, and best practices to help you build a robust monitoring strategy for your Kubernetes environment.

In this section:

Types of monitoring in Amazon EKS

Effective observability in Amazon EKS involves infrastructure, application, and security monitoring activities.

Infrastructure monitoring

Infrastructure monitoring is a fundamental component of Amazon EKS observability that provides deep insights into the health and performance of your Kubernetes cluster's foundational elements. At its core, it involves tracking the vital signs of both control plane components and worker nodes, and making sure that the underlying platform remains stable and efficient.

- **Control plane monitoring** is crucial because it oversees key components such as the API server, etcd database, and scheduler. By monitoring API server latency, you can quickly identify

performance bottlenecks that might affect application deployments or scaling operations. Etcd performance monitoring validates that the cluster's state database operates efficiently and prevents data consistency issues that could impact the entire cluster.

- **Node-level monitoring** is equally critical because it focuses on the compute resources that run your containerized workloads. This includes tracking CPU utilization, memory consumption, disk I/O, and network performance across all worker nodes. Understanding these metrics helps prevent resource exhaustion, optimize node scaling decisions, and ensure appropriate capacity planning.
- **Network monitoring** plays a vital role in maintaining reliable communication between pods, services, and external resources. By monitoring network throughput, latency, and connection states, you can identify connectivity issues early and ensure smooth application communication. Storage monitoring complements network monitoring by tracking volume performance, capacity utilization, and I/O patterns, to help prevent data-related bottlenecks.

Infrastructure monitoring serves as an early warning system for potential issues, enables proactive maintenance, and ensures optimal resource allocation. Without robust infrastructure monitoring, you risk unexpected downtime, degraded performance, and inefficient resource usage that can significantly impact business operations and costs.

Application monitoring

Application monitoring is essential for maintaining healthy, performant, and reliable containerized applications in your Amazon EKS environment. This level of monitoring focuses on the actual workloads that run within your cluster and provides critical insights into how your applications behave, perform, and interact with other services.

Application monitoring includes container-level monitoring, service-level monitoring, and distributed tracing.

- At the **container level**, application monitoring tracks crucial metrics such as container health status, restart counts, and resource consumption patterns. These metrics help you identify problematic containers that might be consuming excessive resources or experiencing frequent restarts, which could indicate underlying issues such as memory leaks or configuration problems. By monitoring container lifecycle events, you can ensure proper application behavior and quickly troubleshoot deployment issues.
- **Service-level monitoring** provides visibility into application performance and reliability metrics such as response times, error rates, and request throughput. These metrics are vital for

maintaining service-level objectives (SLOs) and ensuring a positive end-user experience. You can track latency across different service endpoints, identify performance bottlenecks, and monitor error patterns to maintain application reliability.

- **Distributed tracing** is another critical aspect of application monitoring, especially in microservices architectures. By implementing tracing, you can follow requests as they flow through different services, understand dependencies, and identify performance bottlenecks. This end-to-end visibility helps you optimize service interactions and troubleshoot complex issues that span multiple components.

Custom application metrics play a crucial role in providing business-specific insights. These might include metrics such as order processing rates, user login frequencies, or transaction success rates. You can correlate these custom metrics with infrastructure and container metrics to better understand how infrastructure performance affects business operations and to make data-driven decisions for scaling and optimization.

The importance of application monitoring lies in its ability to provide a comprehensive view of application health and performance. This monitoring enables you to maintain high service quality, quickly resolve issues, and continuously optimize your applications to meet business objectives.

Security monitoring

Security monitoring in Amazon EKS is a critical activity that helps organizations maintain the integrity, confidentiality, and compliance of their Kubernetes environments. This comprehensive security approach combines continuous surveillance, threat detection, and compliance monitoring to protect containerized workloads from potential security risks and unauthorized access.
It includes authentication and authorization monitoring, network security monitoring, and configuration and compliance monitoring.

- **Authentication and authorization monitoring** forms the first line of defense by tracking all attempts to access the cluster. This includes monitoring API server requests, tracking successful and failed login attempts, and auditing role-based access control (RBAC) changes. By maintaining detailed audit logs of who accessed which resources and when, you can quickly detect potential security breaches, unauthorized access attempts, or privilege escalation activities. This is particularly crucial in multi-tenant environments where maintaining strict access controls is essential.
- **Network security monitoring** focuses on detecting and preventing unauthorized communication between pods and services. By monitoring network policy violations and unusual traffic

patterns, you can identify potential security threats such as container escape attempts or lateral movement within the cluster. This includes tracking both internal cluster communication and external traffic patterns to ensure that containers communicate only with authorized endpoints and follow defined security policies.

- **Configuration and compliance monitoring** is essential for maintaining security baselines and meeting regulatory requirements. It involves scanning container images continuously for vulnerabilities, monitoring runtime security, and tracking configuration changes that might impact the security posture. Regular compliance audits ensure adherence to industry standards and organizational security policies, and configuration drift detection helps prevent unauthorized changes that could introduce security risks.

Security monitoring in Amazon EKS provides the necessary visibility and control to help protect against modern security threats while ensuring compliance with regulatory requirements. By implementing comprehensive security monitoring, your organization can maintain a strong security posture, respond quickly to security incidents, and demonstrate compliance with various regulatory standards.

Monitoring tools for Amazon EKS

This section discusses three categories of Amazon EKS monitoring tools: AWS monitoring services, open source or proprietary solutions, and specialized tools.

AWS services

- Amazon CloudWatch: Comprehensive monitoring and logging service

  CloudWatch forms the backbone of AWS monitoring solutions and provides extensive capabilities for Amazon EKS environments. It delivers Container Insights for granular container and cluster metrics, so you can monitor performance, resource utilization, and application health. The service excels in log aggregation and analysis, and supports centralized logging across containers and nodes. CloudWatch integrates naturally with AWS services. It provides automated alarm configuration and supports custom metrics and dashboards, which make it an essential tool for Amazon EKS monitoring.
- AWS X-Ray: Advanced distributed tracing platform

  X-Ray elevates observability by providing sophisticated distributed tracing capabilities. Its service map visualization offers clear insights into application architecture and dependencies,

and detailed request tracking helps identify performance bottlenecks across services. X-Ray can trace requests through complex microservices architectures, which makes it invaluable for troubleshooting and optimization, especially in distributed systems that span multiple AWS services.

- AWS Distro for OpenTelemetry: Unified observability framework

  Distro for OpenTelemetry provides unified data collection capabilities with cross-platform support, which makes it ideal for hybrid environments. This service integrates with other AWS services, supports custom instrumentation, and offers flexibility in implementing comprehensive monitoring solutions while maintaining compatibility with industry standards.

- Amazon Managed Grafana: Enterprise-grade visualization

  Amazon Managed Grafana provides a fully managed service for data visualization and analytics. It offers seamless integration with other AWS services, built-in security features, and enterprise-grade scalability. The service simplifies dashboard creation and management while providing advanced features such as cross-account data source access and integration with AWS IAM Identity Center.

- Amazon Managed Service for Prometheus: Highly available, secure, managed monitoring

  Amazon Managed Service for Prometheus is a fully managed, Prometheus-compatible monitoring service. It provides automated scaling, high availability, and secure metric ingestion and querying. The service integrates seamlessly with Amazon EKS and eliminates the operational overhead of managing Prometheus servers.

Open source or proprietary solutions

The AWS tools described in the previous section offer seamless integration and managed services. The open source tools listed in this section complement AWS services by providing flexibility and extensive customization options. Understanding the capabilities and use cases of each tool helps you design monitoring strategies that best meet your specific requirements.

- Prometheus: Metrics collection toolkit

  Prometheus is an open source solution for metrics collection in Kubernetes environments. Its time-series database and PromQL query language enable sophisticated metrics analyses. The platform's service discovery capabilities automatically adapt to dynamic Kubernetes environments, and its alert management system keep you informed of critical issues.

Prometheus provides extensive integration options, which make it a versatile choice for comprehensive metrics monitoring.

- Grafana: Advanced visualization engine

  Grafana transforms complex monitoring data into actionable insights through its visualization capabilities. The platform creates customized dashboards that combine data from multiple sources and provide a unified view of infrastructure and application metrics. Its support for various data sources and alert management features provide comprehensive monitoring. Grafana can help you visualize both real-time and historical data, so you can identify trends and make informed decisions.

- Fluent Bit: Unified logging layer

  This logging solution provides log collection and management for Kubernetes environments. Its native Kubernetes integration ensures seamless log gathering from containers and nodes, and its support for multiple output destinations offers flexibility in log storage and analysis. Advanced features such as log parsing and filtering enable you to process and route logs based on specific requirements. The lightweight nature of Fluent Bit makes it particularly suitable for containerized environments.

- Datadog: Full-stack observability

  Datadog provides comprehensive monitoring capabilities with native Kubernetes support. It offers infrastructure monitoring, application performance monitoring (APM), log management, and real-time analytics. You can use the platform's automatic service discovery and extensive integration catalog for Amazon EKS monitoring, and its machine learning capabilities to detect anomalies and predict potential issues.

- New Relic: Application performance monitoring

  New Relic offers visibility into application performance and infrastructure health. Its Kubernetes integration provides detailed container insights, distributed tracing, and custom dashboards. The platform helps you correlate application performance with infrastructure metrics, so you can quickly identify and resolve issues.

- Elastic Stack (ELK Stack): Log analysis and search

  The ELK Stack combines Elasticsearch, Logstash, and Kibana to provide log management and analysis capabilities. It offers advanced search functionality, visualization tools, and machine learning features. You can use the stack to handle large volumes of log data from your Amazon EKS environments.

Specialized tools

You can mix and match the following tools based on your specific monitoring requirements, scale of operations, and organizational preferences. The key is to create a monitoring stack that provides comprehensive visibility while remaining manageable and cost-effective.

- kube-state-metrics (KSM): Kubernetes state monitoring

  This add-on service listens to the Kubernetes API server and generates metrics about the state of objects. It provides insights into the health of deployments, pods, and other Kubernetes resources.
- Kubernetes Metrics Server: Resource metrics

  This metrics server collects resource metrics from kubelets and exposes them through the Kubernetes metrics API. It provides horizontal pod autoscaling and basic CPU and memory metrics.
- Kubecost: Kubernetes cost monitoring

  Tools such as Kubecost provide detailed cost analysis and optimization recommendations for EKS clusters. They help you understand and optimize cloud spending across different namespaces, deployments, and services.

Implementing high availability for Amazon EKS monitoring solutions

A robust high availability (HA) strategy for Amazon EKS monitoring is crucial to ensure continuous visibility into your Kubernetes environment. This section discusses a comprehensive approach to implementing HA across different aspects of your monitoring infrastructure.

Architectural redundancy and scalability

Building a highly available monitoring system begins with proper architectural design. Monitoring components should be distributed across multiple AWS Availability Zones to protect against zone failures. This includes implementing horizontal scaling for critical monitoring components such as Prometheus servers, log collectors, and alert managers. You can use AWS managed services such as Amazon Managed Service for Prometheus and Amazon Managed Grafana to help reduce operational overhead while ensuring high availability. Configure automatic failover mechanisms to

maintain service continuity during component failures, with health checks and automated recovery procedures in place.

Resilient data storage strategy

Data storage resilience is fundamental to maintaining monitoring system reliability. Implementing distributed storage solutions ensures that metric data and logs remain accessible even if individual storage nodes fail. This includes configuring proper data replication across multiple Availability Zones and using different storage backends for redundancy. Establish regular backup procedures for historical data, with documented recovery processes for various failure scenarios. For time-series databases such as Prometheus, implementing remote storage solutions helps separate storage concerns from data collection and improves overall system reliability.

Redundant alert management

Alert management requires special attention in an HA setup. Deploying redundant alert managers ensures that critical notifications reach the intended recipients even during system failures. Configure multiple notification channels such as email, SMS, Slack, and PagerDuty to provide alternate communication paths. Use alert deduplication mechanisms to prevent alert storms during partial system failures, and fallback notification methods to ensure that critical alerts are never missed. Implementing alert correlation helps maintain context during failover scenarios and prevents duplicate notifications from redundant systems.

Load balancing and service discovery

Proper load balancing is essential for maintaining stable monitoring services. AWS Application Load Balancers distribute incoming monitoring traffic across multiple endpoints, and health checks ensure that traffic is routed only to healthy instances. Service discovery mechanisms help monitoring components automatically adapt to changes in the environment, such as the addition of new nodes or services. Deploy monitoring agents consistently across all nodes by using DaemonSets to ensure comprehensive coverage as the cluster scales.

Additional HA considerations

Network resilience:

- Implement redundant network paths.
- Configure proper subnet design across Availability Zones.

- Use AWS Direct Connect with backup routes.
- Configure appropriate security groups and network access control lists (network ACLs).

Monitoring the monitors:

- Deploy secondary monitoring systems.
- Implement cross-Region monitoring.
- Configure alerts for unresponsive systems.
- Test failover procedures regularly.

Capacity planning:

- Monitor resource usage trends.
- Implement predictive scaling.
- Test performance on a regular basis.

Data management:

- Implement data retention policies.
- Configure metric aggregation.
- Plan for data lifecycle management.
- Optimize storage on a regular basis.

Recovery procedures:

- Document recovery processes.
- Test disaster recovery regularly.
- Implement automated recovery where possible.
- Identify and implement clear escalation paths.

By implementing these high availability practices, you can ensure that your Amazon EKS monitoring infrastructure remains reliable and resilient, and that you have continuous visibility into your Kubernetes environments even during various failure scenarios. Regular testing and updates to these HA configurations ensure that they remain effective as the environment evolves.

Best practices for monitoring in Amazon EKS

Strategic implementation approach

A successful Amazon EKS monitoring strategy begins with a well-planned, phased implementation approach.

- Start by identifying and monitoring critical metrics that directly affect your business operations and application reliability. This foundation should include essential infrastructure metrics, key application performance indicators, and critical security metrics. Gradually expand monitoring coverage based on operational needs and lessons learned, and make sure that each addition provides meaningful value.
- Implement automated deployment processes by using infrastructure as code (IaC) tools such as Terraform or CloudFormation to ensure consistency and repeatability.
- Test and validate monitoring systems to help maintain reliability and accuracy.
- Refine monitoring parameters continuously in alignment with evolving business needs.

Effective data management

Proper data management is crucial for maintaining an efficient and cost-effective monitoring solution.

- Implement clear data retention policies that balance historical analysis needs with storage costs.
- Configure appropriate sampling rates for different metric types: higher frequency for critical metrics and lower frequency for less critical ones.
- Use metric aggregation to reduce data volume while maintaining meaningful insights, especially for long-term trend analysis.
- Implement systematic log retention and archival procedures for centralized logging systems (such as CloudWatch Logs) to manage storage costs and maintain access to important data remains accessible.

> **Note**
>
> Container-level log rotation is handled automatically by the kubelet in Amazon EKS version 1.21 or later.

- Consider implementing a hot-warm-cold architecture for log storage to optimize both access speed and cost efficiency.

Alert configuration and management

Alert configuration requires careful consideration to maintain effectiveness without causing alert fatigue.

- Define clear, actionable thresholds based on service level objectives (SLOs) and historical performance patterns.
- Implement a tiered alert severity system that clearly differentiates between critical issues that require immediate attention and less urgent matters.
- Make sure that alerts provide sufficient context and actionable information to facilitate quick problem resolution.
- Establish clear escalation procedures with defined ownership and response times for different alert severities.
- Review and refine alert configurations regularly to help maintain their relevance and effectiveness.

Resource optimization

Continuous monitoring of resource utilization is essential for maintaining cost-effective operations.

- Implement comprehensive resource monitoring across all cluster components, including nodes, pods, and persistent volumes.
- Configure automatic scaling based on actual usage patterns and performance requirements to ensure efficient resource utilization while maintaining performance.
- Use cost allocation tags to track resource consumption by different teams, applications, or environments.
- Regularly analyze resource efficiency metrics to identify optimization opportunities and implement improvements.
- Consider implementing cost management tools to track and optimize cloud spending.

Security

Security considerations should be integral to your monitoring strategy.

- Implement least privilege access principles for all monitoring components to ensure that users and services have only the permissions they need.
- Enable comprehensive audit logging to track all access and changes to monitoring systems.
- Conduct regular security reviews of monitoring configurations and access patterns to identify potential vulnerabilities.
- Implement encryption for sensitive monitoring data both in transit and at rest.
- Integrate security monitoring with existing security information and event management (SIEM) systems for comprehensive security visibility.

Advanced monitoring considerations in Amazon EKS

Performance optimization:

- Optimize metric collection intervals.
- Configure efficient query patterns.
- Implement metric pre-aggregation.
- Use appropriate storage solutions.

Compliance and governance:

- Maintain audit trails.
- Implement compliance monitoring.
- Provide regular compliance reporting.
- Document monitoring procedures.

Disaster recovery:

- Back up monitoring configurations regularly.
- Document recovery procedures.
- Test recovery processes.

Continuous improvement:

- Monitor review sessions regularly.
- Optimize performance cycles.
- Update monitoring based on incidents.
- Incorporate user feedback.

These best practices provide a framework for implementing and maintaining effective monitoring solutions for Amazon EKS environments. Regularly review and update these practices so they remain aligned with your organizational needs and industry standards. Monitoring is not a one-time setup—it's a continuous process that requires regular attention and refinement.

Tracing in Amazon EKS

Tracing is a critical component of application observability in Amazon EKS. Tracing provides detailed visibility into request flows and service interactions by collecting, processing, and visualizing the path of requests as they travel through various microservices that are deployed on EKS clusters. This capability helps you understand system behavior, identify bottlenecks, and troubleshoot issues effectively in your Amazon EKS environment. Effective tracing eliminates the complexity of debugging distributed systems by providing end-to-end visibility into request flows. It makes it possible to track transactions across service boundaries and identify performance issues or failures within Amazon EKS workloads.

The overall tracing implementation in Amazon EKS enables you to understand system behavior, optimize performance, and maintain reliability of your containerized applications. Ultimately, the capabilities of tracing enhance operational visibility and system maintainability in Amazon EKS environments.

AWS X-Ray plays a significant role in tracing data about your application. Tracing involves monitoring various aspects of the service interactions, including the following:

- **Request paths and dependencies** provide crucial insights into your distributed system's behavior. They track the complete journey of requests as they traverse through different microservices and components. Mapping service dependencies helps you understand communication patterns and identify critical paths in your application architecture. For implementation details, see Using the AWS X-Ray service trace map in the X-Ray documentation.
- **Service latencies and bottlenecks** are essential metrics for maintaining optimal system performance. By measuring and analyzing response times between services, you can identify performance issues effectively. This data allows you to pinpoint specific services or operations that are causing delays in the request chain and enable targeted optimization efforts. To learn more about latency analysis, see Interacting with the Analytics console in the X-Ray documentation.
- **Error propagation patterns** help you understand system reliability and fault tolerance. By understanding how failures cascade through the system by tracking error paths across services, you can better architect your applications. This visibility helps you identify the root cause of errors and their impact on dependent services, which leads to more resilient systems. For implementation details, see Traces in the X-Ray documentation.

- **Resource utilization across services** provides insights into system efficiency and cost optimization. You can monitor CPU, memory, and network usage patterns that are correlated with trace data to understand resource demands. This data helps you analyze resource consumption trends to optimize service performance and cost across your EKS cluster. For monitoring setup, see Monitor your cluster performance and view logs in the Amazon EKS documentation.
- **End-user transaction flows** are critical for understanding and improving the user experience. By tracking complete user interactions from frontend to backend services, you can ensure optimal application performance. You can measure and optimize end-to-end response times for critical user journeys, which directly impacts customer satisfaction. To implement end-user monitoring, use the AWS X-Ray SDK for your programming language.
- **API gateway interactions** form the front line of your application's performance and security. You can monitor request patterns and performance at API entry points to ensure optimal service delivery. This visibility helps you track authentication, authorization, and rate limiting impacts on request flows, to maintain both security and performance requirements. Learn more about API tracing in the Amazon API Gateway with X-Ray documentation.

Effective tracing in Amazon EKS goes beyond collecting spans and traces. It requires a well-structured strategy that balances observability needs with system performance. This strategy should focus on:

- **Implementing appropriate sampling rates**: Configure sampling rules based on traffic patterns and business priorities to optimize cost while maintaining the visibility of critical transactions. To learn more, see Configuring sampling rules in the X-Ray documentation.
- **Defining critical paths and services to trace**: Identify and prioritize essential services and user journeys that require detailed tracing to ensure optimal performance monitoring. For more information, see Send metric and trace data with ADOT Operator in the Amazon EKS documentation.
- **Establishing proper data retention policies**: Set up data lifecycle management rules to balance observability needs with storage costs and compliance requirements. To view CloudWatch retention policies, see Working with log groups and log streams in the CloudWatch Logs documentation.
- **Setting up effective visualization and analysis tools**: Deploy and configure visualization tools such as the AWS X-Ray Analytics console or Amazon Managed Grafana to analyze trace

data effectively. For more information, see Interacting with the Analytics console in the X-Ray documentation.

In this section:

Tracing tools for Amazon EKS

Amazon EKS supports several AWS and third-party options for implementing distributed tracing.

AWS services

- AWS X-Ray: Advanced distributed tracing platform

  X-Ray is a fully managed AWS service that provides end-to-end tracing capabilities. It automatically instruments AWS services and provides detailed service maps and analytics for your applications that run on Amazon EKS. X-Ray is integrated with other AWS services, including Amazon CloudWatch, and offers automatic correlation of traces with AWS service calls.

- AWS Distro for OpenTelemetry: Unified observability framework

  Distro for OpenTelemetry is a secure, production-ready, and AWS-supported distribution of OpenTelemetry for cloud-native applications. It offers vendor-neutral instrumentation capabilities while maintaining native AWS service integration, which makes it ideal for hybrid cloud environments. Distro for OpenTelemetry supports multiple observability backends and provides seamless integration with AWS monitoring services.

Open source solutions

- OpenTelemetry: Open source observability framework

  OpenTelemetry provides a standardized observability framework with comprehensive instrumentation libraries that support multiple programming languages. Its flexible backend options and vendor-neutral approach make it ideal for workloads that require consistency across different environments. The framework's extensive ecosystem ensures broad compatibility with various monitoring solutions.

- Jaeger: Open source distributed tracing platform

  Jaeger offers comprehensive tracing capabilities with real-time distributed context propagation. It provides root cause analysis and performance optimization through detailed service dependency visualization. Jaeger's architecture is designed for high scalability and supports various storage backends, which makes it suitable for large-scale Amazon EKS deployments. View Jaeger for EKS setup

- Grafana Tempo: Distributed tracing

  Tempo is a Grafana Labs solution that provides high-scale trace storage and seamless integration with Prometheus metrics. Its cost-effective trace retention model and native integration with Grafana make it suitable for organizations that already use Grafana for visualization. Tempo's architecture is designed specifically for cloud-native environments such as Amazon EKS.

Best practices for tracing in Amazon EKS

This section provides a comprehensive list of best practices and techniques for creating an effective tracing system that enhances observability and troubleshooting for your Kubernetes-based applications in Amazon EKS.

- **Strategic sampling**: Configure different sampling rates based on your application's traffic patterns and the importance of the services you're using. Implement higher sampling rates for critical paths while reducing sampling for high-volume, less critical routes to optimize costs. For guidance, see Configuring sampling rules in the AWS X-Ray documentation.
- **Instrumentation setup**: Use automatic instrumentation tools such as the X-Ray SDK or AWS Distro for OpenTelemetry collectors to minimize the manual instrumentation effort. Maintain consistent naming conventions and context propagation across services for better trace correlation. For more information, see the Distro for OpenTelemetry collector documentation.
- **Data management**: Implement appropriate retention periods and compression strategies to balance storage costs with your observability needs. Establish clear data privacy controls and backup procedures to protect sensitive trace data. For more information, see Change log data retention in CloudWatch Logs in the CloudWatch Logs documentation.
- **Performance optimization**: Monitor and optimize tracing overhead to minimize impact on application performance. Use efficient buffering and asynchronous processing to reduce latency impact. For more information, see Configuring the AWS X-Ray daemon in the X-Ray documentation.

- **Security controls**: Implement proper access controls and data protection measures by using IAM roles and policies. Regular security audits and compliance reviews help ensure that trace data remains secure. For more information, see Security in AWS X-Ray in the X-Ray documentation.
- **Monitoring and alerts**: Set up comprehensive monitoring for trace collection health and configure alerts for collection issues. Track sampling rates and system performance metrics to ensure optimal operation. For more information, see Container Insights in the CloudWatch documentation.
- **High availability**: Deploy redundant collectors across Availability Zones and configure proper failover mechanisms. Regular testing of high availability setup ensures reliable trace collection. For more information, see Using AWS Distro for OpenTelemetry as a collector in the Amazon Managed Service for Prometheus documentation.

By following these best practices, you can create a robust, efficient, and effective tracing system for your Amazon EKS environment. This will help ensure comprehensive observability, efficient troubleshooting, and optimal performance of your Kubernetes-based applications.

Alerting in Amazon EKS

Alerting is a critical component of managing and maintaining applications that run on Amazon EKS. It serves as an early warning system that notifies operators and developers about potential issues, anomalies, or performance degradations before they escalate into serious problems that could impact service availability or user experience. Alerting involves monitoring various aspects of the Kubernetes cluster, including:

- Infrastructure health
- Application performance
- Container metrics
- Custom business metrics

Effective alerting in Amazon EKS goes beyond simply setting up notifications. It requires a well-thought-out strategy that balances the need for timely information with the the potential for alert fatigue. This strategy should:

- Define meaningful thresholds and conditions.
- Prioritize alerts based on severity and impact.
- Implement proper routing and escalation procedures.
- Integrate with incident management and communication tools.

In this section:

Alerting tools for Amazon EKS

Amazon EKS supports several AWS and third-party options for implementing alerting. When you choose a tool for Amazon EKS alerting, consider factors such as integration capabilities, scalability, ease of use, cost, and specific features that align with your monitoring and alerting requirements. Many organizations use a combination of these tools to create a comprehensive monitoring and alerting solution for their Amazon EKS environments.

- Amazon CloudWatch: AWS service for monitoring and observability

  CloudWatch provides metrics, logs, and alarms for EKS clusters, and integrates well with other AWS services.

- Prometheus: Open source monitoring and alerting tool for Kubernetes

  Prometheus provides a powerful query language (PromQL) for defining alert conditions.

- Alertmanager: Companion to Prometheus for handling alerts

  Alertmanager provides deduplication, grouping, and routing of alerts. It supports various notification channels, including email, Slack, and PagerDuty.

- Grafana: Open source platform for monitoring and observability

  Grafana provides visualization and alerting capabilities. It can integrate with various data sources, including Prometheus and CloudWatch.

- Elastic Stack (ELK Stack): Combination of Elasticsearch, Logstash, and Kibana

  This tool is useful for log aggregation, analysis, and alerting. It can be extended with Elastic's observability features.

- Third-party solutions

  There are many tools available on the market, including Datadog, New Relic, Sysdig, Dynatrace, Zabbix, Nagios, Splunk, IBM Instana, and AppDynamics.

Best practices for alerting in Amazon EKS

This section describes the best practices for creating a robust alerting system that enhances the reliability and performance of your Kubernetes-based applications in Amazon EKS.

Define clear alert thresholds:

- Set meaningful thresholds based on historical data and business requirements.
- Use dynamic thresholds where appropriate to account for varying workloads.

Implement alert prioritization:

- Categorize alerts by severity (for example, critical, high, medium, low).

- Align alert priorities with business impact.

Avoid alert fatigue:

- Reduce noise by eliminating redundant or low-value alerts.
- Correlate alerts to group related issues.

Use multi-stage alerting:

- Implement warning thresholds before critical levels are reached.
- Use different notification channels for different alert severities.

Implement proper alert routing:

- Make sure that alerts are sent to the right teams or individuals.
- Use on-call schedules and rotations for all day, every day coverage.

Leverage Kubernetes-native metrics:

- Monitor core Kubernetes components (nodes, pods, services).
- Use kube-state-metrics (KSM) for additional Kubernetes object metrics.

Monitor both infrastructure and applications:

- Set up alerts for cluster health, node status, and resource utilization.
- Implement application-specific alerts such as error rates and latency.

Use Prometheus and Alertmanager:

- Use Prometheus for metric collection and PromQL to define alert conditions.
- Use Alertmanager for alert routing and deduplication.

Integrate with Amazon CloudWatch:

- Use CloudWatch Container Insights for Amazon EKS-specific metrics.

- Set up CloudWatch alarms for critical AWS resource metrics.

Implement context-rich alerts:

- Include relevant information in alert messages, such as cluster name, namespace, and pod details.
- Provide links to relevant dashboards or runbooks in alerts.

Use anomaly detection:

- Implement machine learning-based anomaly detection for complex patterns.
- Use services such as CloudWatch anomaly detection or third-party tools.

Implement alert suppression and silencing:

- Allow temporary suppression of known issues.
- Implement maintenance windows to reduce noise during planned downtimes.

Monitor alert performance:

- Track metrics such as alert frequency, resolution time, and false positive rates.
- Regularly review and refine alert rules based on these metrics.

Implement escalation procedures:

- Define clear escalation paths for unresolved alerts.
- Use tools such as PagerDuty or Opsgenie for automated escalations.

Test alert systems regularly:

- Conduct periodic tests of your alerting pipeline.
- Include alert testing in disaster recovery drills.

Use templates for alert consistency:

- Create standardized alert templates for common scenarios.
- Ensure consistent formatting and information across all alerts.

Implement rate limiting:

- Prevent alert storms by implementing rate limiting on frequently triggered alerts.

Use custom metrics:

- Implement custom metrics for application-specific monitoring.
- Use the Kubernetes custom metrics API for automatic scaling based on these metrics.

Implement logging integration:

- Correlate alerts with relevant logs for faster troubleshooting.
- Use tools such as Grafana Loki or the ELK Stack in conjunction with your alerting system.

Consider cost alerts:

- Set up alerts for unexpected spikes in resource usage or costs.
- Use AWS Budgets or third-party cost management tools.

Use distributed tracing:

- Integrate distributed tracing tools such as Jaeger or AWS X-Ray.
- Set up alerts for abnormal trace patterns or latencies.

Document alert runbooks:

- Create clear, actionable runbooks for each alert type.
- Include troubleshooting steps and escalation procedures in runbooks.

By following these best practices, you can create a robust, efficient, and effective alerting system for your Amazon EKS environment. This will help ensure high availability, quick issue resolution, and optimal performance of your Kubernetes-based applications.

Next steps

This guide provided a comprehensive framework for implementing robust observability in Amazon EKS environments, focusing on metrics collection, logging infrastructure, distributed tracing, and cost optimization. By understanding and applying these core components, you can build a highly observable, maintainable, and cost-effective container environment that provides deep insights into application and infrastructure behavior. The integration of AWS services such as Amazon CloudWatch Container Insights and AWS X-Ray, combined with open-source solutions such as Prometheus and OpenTelemetry, creates a powerful foundation for monitoring and troubleshooting containerized applications.

Implementation success relies on a phased approach, starting with core metrics collection and gradually expanding to comprehensive logging and distributed tracing capabilities. We recommend that you begin by assessing your current monitoring capabilities, identifying gaps, and selecting appropriate tooling combinations that align with your operational requirements and team expertise. This methodical approach ensures that each component of the observability stack is properly implemented and integrated, while teams develop the necessary skills and processes to effectively use these tools.

The long-term sustainability of Amazon EKS observability depends on regular optimization of costs, resources, and processes. You should continuously review and adjust your observability infrastructure, including data retention policies, sampling rates, and resource allocation, to maintain the right balance between comprehensive monitoring and operational efficiency. This iterative approach to improvement, combined with ongoing team training and documentation updates, enables your organization to maintain effective observability while supporting business growth and adapting to evolving application architectures.

Resources

AWS documentation

- Amazon EKS Best Practices Guide
- Amazon CloudWatch Container Insights
- Amazon Managed Service for Prometheus
- Amazon Managed Grafana
- AWS Distro for OpenTelemetry and AWS X-Ray
- Amazon OpenSearch Service

AWS blog posts

- Amazon EKS enhances Kubernetes control plane observability
- Automating metrics collection on Amazon EKS with Amazon Managed Service for Prometheus managed scrapers
- Automate monitoring for your Amazon EKS cluster using CloudWatch Container Insights
- Enhancing observability with a managed monitoring solution for Amazon EKS

Other resources

- OpenTelemetry documentation
- Prometheus documentation
- Fluent Bit documentation
- Monitoring, Logging, and Debugging in Kubernetes documentation

Document history

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

Change	Description	Date
Updates	We updated the Logging in Amazon EKS chapter.	March 17, 2026
Initial publication	—	April 10, 2025

AWS Prescriptive Guidance glossary

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

Numbers

7 Rs

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

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

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

A

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.