Applying the AWS Well-Architected Framework to Amazon AppStream 2.0

AWS Prescriptive Guidance

AWS Prescriptive Guidance: Applying the AWS Well-Architected Framework to Amazon AppStream 2.0

Table of Contents

Applying the AWS Well-Architected Framework to Amazon AppStream 2.0

Mohamed Wali, Amazon Web Services

July 2025 (document history)

This guide covers best practices for applying the AWS Well-Architected Framework when you use Amazon AppStream 2.0. AppStream 2.0 is a fully managed application streaming service that enables you to stream desktop applications to users without rewriting them.

The AWS Well-Architected Framework helps cloud architects build secure, high-performing, resilient, and efficient infrastructures for a variety of applications and workloads. It also provides a consistent approach for users and AWS Partners to evaluate architectures and implement scalable designs.

The AWS Well-Architected Framework is built around six pillars:

- Operational excellence
- Security
- Reliability
- Performance efficiency
- Cost optimization
- Sustainability

This guide discusses how these pillars and best practices apply to using AppStream 2.0.

Intended audience

This guide is for:

- Cloud architects and engineers who design and implement AppStream 2.0 solutions, and need to ensure that their architectures follow AWS Well-Architected Framework best practices.
- IT operations teams that manage and maintain AppStream 2.0 environments; handle fleet management, scaling, and monitoring; and need to optimize costs and performance.

- Organizations or businesses that are considering or already using AppStream 2.0, want to stream desktop applications to their users, and need to build secure, high-performing, resilient, and efficient infrastructures.

Objectives

Following the best practices in this guide helps you:

- Build a secure, high-performing, resilient, and efficient infrastructure for streaming desktop applications in the AWS Cloud.
- Apply a consistent approach when evaluating AppStream 2.0 architectures and implementing scalable designs.

Operational excellence pillar

Operational excellence (OE) represents a dedication to crafting high-quality software solutions that consistently meet and exceed user expectations. The operational excellence pillar of the AWS Well-Architected Framework encompasses proven strategies for effective team organization, robust workload design, efficient large-scale operations, and seamless adaptation to changing requirements over time. By adhering to these principles, organizations can ensure that their systems remain resilient, performant, and aligned with evolving business needs.

Key focus areas for applying this pillar to your AppStream 2.0 streaming environment:

- Monitoring and observability
- Automation and DevOps
- Operational procedures and documentation
- Support and incident management

Organize teams around business outcomes

Create a cloud-aligned operating model with strong leadership commitment, where business goals and key performance indicators (KPIs) drive organizational transformation through optimized people, processes, and technology.

- **Team structure.** Establish dedicated teams that align with application streaming outcomes. For example:
  - Image management team is responsible for application packaging and image optimization.
  - Fleet operations team manages capacity, performance, and scaling.
  - User experience team handles end-user support and satisfaction.
- **KPIs and metrics.** Define and track business-aligned metrics such as:
  - Application availability rates
  - Time to deploy new applications
  - Cost per application streaming hour
- **Operating model.** Create clear processes for:
  - Application onboarding and updates

- Fleet capacity management
- User access provisioning
- Incident response and resolution

Implement observability for actionable insights

Implement comprehensive monitoring and observability to track KPIs and workload health. This principle enables data-driven decisions and proactive improvements across performance, reliability, and cost.

- Implement performance monitoring. Configure Amazon CloudWatch to:
  - Ensure sufficient capacity to meet demand. For example, you can use the following metrics:
    - `AvailableCapacity` to monitor available streaming instances
    - `InUseCapacity` to track currently used instances
    - `CapacityUtilization` to monitor the percentage of fleet usage
  - Monitor user experience and performance.
  - Identify and address service issues promptly.
- Track and analyze AppStream 2.0 usage reports.
- Capture and analyze application logs. For more information, see the AWS blog posts Using Kinesis Agent for Linux to stream application logs in AppStream 2.0 and Using Kinesis Agent for Microsoft Windows to store AppStream 2.0 Windows event logs.
- Monitor AppStream 2.0 metrics and events through chat notifications. For more information, see the AWS blog post Monitor and automate AWS end user computing (EUC) with AWS Chatbot.
- Enable proactive session management through visual cues. For more information, see the AWS blog post Display session expiration and a countdown timer in Amazon AppStream 2.0.
- Create visualizations for usage patterns and trends. For more information, see the AWS blog post Ingest and visualize Amazon AppStream 2.0 usage reports in Amazon OpenSearch Service.
- Utilize the EUC toolkit to monitor active sessions, track fleet inventory, and generate session reports (CSV export). For more information, see the AWS blog post Use the EUC Toolkit to manage Amazon AppStream 2.0 and Amazon WorkSpaces.

Safely automate where possible

Apply infrastructure as code (IaC) principles to automate all aspects of your workload operations. Use guardrails to help ensure safe and consistent execution while reducing manual intervention.

- Automate the creation and configuration of AppStream 2.0 images by using the Image Assistant CLI. For more information, see Create your Amazon AppStream 2.0 image programmatically by using the Image Assistant CLI operations in the AppStream 2.0 documentation.
  - Application installation: Use the Image Assistant CLI to automate the installation of applications during image creation.
  - Image creation: Programmatically create AppStream 2.0 images by using the Image Assistant CLI commands.
  - Configuration management: Automate the configuration of default application settings and launch parameters.
- Automate the customization of AppStream 2.0 images. For more information, see the AWS blog post Automatically create customized AppStream 2.0 Windows images.
- Apply IaC to deploy the infrastructure and application components for AppStream 2.0. For more information, see the AWS blog post Automation of infrastructure and application deployment for Amazon AppStream 2.0 with Terraform.
- Implement automated processes for fleet management, including:
  - Fleet scaling based on demand. Configure automatic scaling policies to adjust fleet capacity automatically based on utilization metrics. For more information, see the AWS blog post Use AWS Lambda to adjust scaling steps and thresholds for Amazon AppStream 2.0.
  - Base image updates. Benefit from automatic updates to the AppStream 2.0 base image that's provided by AWS.
  - Capacity optimization. Set up automated scaling thresholds to optimize resource usage based on demand patterns.
- Configure guardrails to automate safety controls:
  - Maximum fleet size limits. Set upper bounds on fleet capacity to prevent over-provisioning.
  - Scaling policy configuration. Implement step scaling or target tracking scaling policies with appropriate thresholds.
  - Service quotas. Use AWS service quotas as built-in limits to prevent excessive resource allocation.

    - Scale-in protection. Configure scale-in protection to prevent the removal of active instances during scaling events.
- Perform testing and validation, including image builder, fleet, and integration testing.
    - Image builder testing:
        - Test applications directly in the image builder interface.
        - Verify application launch and functionality.
        - Test user settings and configurations.
        - Validate application compatibility.
    - Fleet testing:
        - Test streaming sessions from different client devices.
        - Verify user entitlements and access.
        - Validate application performance.
        - Test user experience for elements and operations such as the clipboard, file transfer, and printing.
    - Integration testing:
        - Test Active Directory or SAML 2.0-based authentication.
        - Test home folders and persistent storage.
        - Test application entitlements.
        - Test USB device redirection (if configured).
- Use the AppStream 2.0 applications manager to automate application packaging and deployment. For more information, see the AWS blog post Streamline application onboarding with applications manager for Amazon AppStream 2.0.
- Automate the deployment of new application versions by using continuous integration and continuous delivery (CI/CD) pipelines. For more information, see the AWS blog post Screening Eagle: Optimize CI/CD and end user experience in Amazon AppStream 2.0.

Make frequent, small, reversible changes

Build loosely coupled, scalable workloads that enable frequent, small-scale automated deployments with minimal risk and easy rollback capabilities.

- For image updates, use versioned image creation and incremental updates.

  - Versioned image creation:
    - Create new images for each set of changes by using an image builder.
    - Maintain multiple image versions to support rollback scenarios.
    - Use AWS tagging strategies to track image versions and attributes.
  - Incremental updates:
    - Make small, incremental changes to applications or configurations.
    - Test updates thoroughly in the image builder before you create a new image.
    - Document all the changes that you made in each new image version.
- For control fleet updates:
  - Create new fleets with updated images for testing.
  - Modify existing fleet attributes without disrupting active sessions.
- Establish change management procedures for documentation, testing protocols, approval workflows, and monitoring processes.
  - Documentation:
    - Maintain detailed change logs for all image and fleet updates.
    - Document testing procedures and results for each change.
    - Use AWS CloudTrail to track and audit configuration changes.
  - Testing protocols:
    - Establish a comprehensive testing process for all changes.
    - Include application functionality, performance, and user experience tests.
    - Conduct testing in the image builder before you create new images.
    - Perform additional testing on non-production fleets before full deployment.
  - Approval workflows:
    - Implement an approval process for changes to production environments.
    - Define criteria for changes that require approval versus standard updates.
    - Establish roles and responsibilities for change approval.
  - Monitoring and validation:
    - Use Amazon CloudWatch to monitor fleet and application performance after changes.
    - Set up alerts for key metrics to quickly identify issues after updates.
    - Conduct post-implementation reviews to validate change success and gather learnings.

Refine operations procedures frequently

Continuously improve operational procedures through regular reviews, updates, and team engagement to keep all stakeholders informed and aligned with best practices.

- **Documentation management.** Maintain current, version-controlled documentation of AppStream 2.0 procedures in a central location to ensure operational consistency and knowledge sharing across teams.
  - Required documentation: Maintain up-to-date documentation for critical AppStream 2.0 operations for image creation and management, fleet operations, and troubleshooting.
  - Operational reviews: Monitor and review key operational aspects, including performance metrics and incident management.
- **Continuous improvement.** Systematically enhance AppStream 2.0 operations by incorporating AWS service updates, operational metrics, and learned best practices into standard procedures.
  - Service updates: Monitor AppStream 2.0 release notes for new features, service improvements, security updates, and Regional availability.
  - Best practices: Review and incorporate AWS Well-Architected Framework updates, AppStream 2.0 best practices, AWS reference architectures, and AWS security recommendations.
  - Knowledge management: Maintain and update standard operating procedures, runbooks, troubleshooting guides, and user support documentation.

Anticipate failure

Conduct failure scenario testing regularly to understand risks, validate response procedures, and improve team readiness for handling real incidents.

- **Failure testing.** Regularly simulate and test for failures such as fleet capacity exhaustion, application launch failures, and network connectivity issues.
  - Fleet capacity exhaustion:
    - Monitor and test fleet scaling behavior when approaching capacity limits.
    - Configure CloudWatch alarms for `CapacityUtilization` and `AvailableCapacity` metrics.
    - Implement procedures for handling capacity constraints during peak usage.
  - Application launch failures:

    - Test application launch behavior on streaming instances.
    - Validate application access and performance across different fleet configurations.
  - Network connectivity issues:
    - Test streaming session performance across different network conditions.
    - Monitor `StreamingSessionLatency` for connection quality issues.
    - Ensure proper configuration of VPC settings and security groups.
- **Recovery procedures.** Develop and test procedures for:
  - Fleet failover between AWS Availability Zones. In addition, document procedures for scaling fleet capacity, managing fleet updates, and responding to instance health issues.
  - User data management:
    - Configure and test application settings persistence and storage solutions for home folders in Amazon Simple Storage Service (Amazon S3) for Windows fleets and shared file systems in Amazon Elastic File System (Amazon EFS) for Linux fleets.
    - Validate data synchronization between sessions.
  - Service continuity. Maintain procedures for creating new fleet instances, managing image updates, and handling session disconnections.
- **Risk management.** Identify and mitigate:
  - Capacity constraints by setting appropriate fleet minimum capacity, configuring automatic scaling policies based on demand patterns, and monitoring fleet utilization trends by using CloudWatch metrics such as `CapacityUtilization`, `InUseCapacity`, and `AvailableCapacity`.
  - Performance bottlenecks by tracking key metrics such as `StreamingSessionLatency` and configuring the appropriate CloudWatch alarms.

Learn from all operational events and metrics

Foster a culture of continuous improvement by sharing lessons learned from operational events and failures across the organization. Emphasize their impact on business outcomes.

- **Event analysis.** Document and analyze service interruptions, performance degradation, user complaints, and capacity issues.
- **Metrics review.** Analyze usage patterns, performance trends, cost metrics, and user satisfaction data on a regular basis.

- **Knowledge sharing.** Establish processes for team learning sessions, best practice documentation, cross-team knowledge transfer, and incident retrospectives.

Use managed services

Minimize operational overhead by using AWS managed services and building standardized procedures around them. Integrate with the following AWS managed services:

- AWS Systems Manager for automation
- Amazon CloudWatch for monitoring
- AWS Identity and Access Management (IAM) for access control
- Amazon S3 for user storage for Windows fleets
- Amazon EFS for user storage for Linux fleets
- AWS Directory Service for user authentication

Security pillar

The security pillar of the AWS Well-Architected Framework focuses on taking advantage of cloud capabilities to help establish robust protection mechanisms for your information, infrastructure, and resources. These principles help enhance your overall security posture while enabling innovation.

Key focus areas for applying this pillar to your AppStream 2.0 streaming environment:

- Data integrity and confidentiality
- Managing user permissions
- Establishing controls to detect security events

Implement a strong identity foundation

Use minimum required permissions to access AWS resources while centralizing identity management and avoiding long-term credentials.

- Grant least privileged permissions for AppStream 2.0 resources:
  - Create specific IAM roles for AppStream 2.0 fleets with minimal required permissions.
  - Configure limited IAM permissions for image builders.
  - Restrict administrative access to AppStream 2.0 management functions.
  - Define granular permissions for stack and fleet management.
- Implement proper user authentication mechanisms:
  - Configure SAML 2.0 federation for enterprise identity provider integration.
  - Set up AWS IAM Identity Center for user management.
  - Use custom identity brokers only when required for specific authentication scenarios.
  - Implement multi-factor authentication (MFA) where supported.
- Control user access to applications:
  - Configure application entitlements to restrict access to specific applications.
  - Create application assignment groups based on user roles.
  - Manage application access through stack permissions.
  - Implement session policies to control application behavior.

- Secure user sessions with appropriate controls:
  - Configure session timeout policies.
  - Set disconnect timeout actions.
  - Implement session persistence requirements.
  - Control file system redirection permissions.
- Configure certificate-based authentication for AppStream 2.0. For more information, see the AWS blog post Simplify certificate-based authentication for AppStream 2.0 and WorkSpaces with AWS Private CA Connector for Active Directory.
- Use session tags to implement fine-grained access control. For more information, see the AWS blog post Use session tags to simplify AppStream 2.0 permissions.

Maintain traceability

Implement real-time monitoring and automated response systems for all environment changes and activities.

- Configure CloudWatch logging for application logs to monitor application-specific events, including application launches, crashes, and errors. Configure session logs to track streaming session information, including session starts, stops, and user connection events.
- Activate CloudTrail to log all AppStream 2.0 API calls and to track management events such as fleet creation and modifications, image builder operations, stack configurations, and user management activities.
- Monitor AppStream 2.0 instance activity:
  - Configure instance logging to capture system-level events.
  - Track application launches and failures.
  - Monitor system resource usage and performance.
- Track user activity:
  - Monitor user authentication attempts and failures. Use CloudWatch metrics and CloudWatch Logs to track user login attempts, session start and end times, and session disconnect events.
  - Track application usage patterns. Enable AppStream 2.0 usage reports to retrieve information such as session duration, start and end times, instance types used, and applications accessed.
  - Record file system activities through enabled home folders.

    - Configure clipboard settings and printing operations to achieve your data loss prevention goals.
- Configure CloudWatch alarms for security-related metrics such as failed user authentications, unusual session patterns, and resource access violations.
- Use the EUC toolkit to track active sessions and states, monitor IP addresses for in-use active sessions, and export session data for auditing. For more information, see the AWS blog post Use the EUC toolkit to manage Amazon AppStream 2.0 and Amazon WorkSpaces.

Apply security at all layers

Implement multiple layers of security controls across all components of your infrastructure, from network edge to application code.

- Configure network layer security:
    - Implement strict security group rules.
    - Place AppStream 2.0 fleet instances in private subnets that have no direct internet access. Control internet access through NAT devices.
    - Use virtual private cloud (VPC) endpoints to access supported AWS services (such as Amazon S3).
    - Implement network access control lists (ACLs) as an additional network security layer.
    - Restrict streaming port (TCP 8443 for HTTPS and WebSocket Secure) access to specific IP ranges.
- Configure access layer security:
    - Implement session timeout policies to automatically disconnect inactive users.
    - Implement attribute-based access control by using session tags. For more information, see the AWS blog post Use session tags to simplify AppStream 2.0 permissions.
- Configure application layer security:
    - Configure application entitlements to control which users can access specific applications.
    - Enable file system redirection controls to restrict access to local drives.
    - Configure clipboard, file transfer, and printing permissions based on security requirements.
    - Set up USB device access controls according to security policies.
- Configure image layer security:

- Create and maintain hardened base images that meet security requirements.
- Keep base images updated with the latest security patches.
- Configure Windows security settings in base images.
- Disable unnecessary Windows services and features in base images.

Automate security best practices

Use automated, code-defined security controls in version-controlled templates to enable secure and scalable infrastructure deployment.

- Use infrastructure as code (IaC) by using services such as AWS CloudFormation to implement consistent security configurations across all fleet deployments. For more information, see the AWS blog post Automatically attach additional security groups to Amazon AppStream 2.0 and Amazon WorkSpaces.
- Automate image creation security processes by using the Image Assistant CLI.
- Configure automated responses for capacity utilization thresholds exceeded, unauthorized access attempts, and security group changes by using Amazon CloudWatch alarms, Amazon EventBridge rules, and AWS Lambda functions for automated responses.

Keep people away from data

Automate data handling processes to minimize direct human access and reduce the risk of errors or mishandling.

- Configure application entitlements to control which users can access specific applications.
- Use the dynamic application framework to build a dynamic app provider to make applications available dynamically based on user attributes.
- Configure file system redirection to control which local drives users can access, to restrict access to specific folders, and to manage file transfer permissions between local and streaming sessions.
- Implement clipboard restrictions to disable clipboard sharing between local and streaming sessions, enable one-way clipboard flow where needed, and prevent unauthorized data copying.
- Configure application settings persistence to automatically save and restore application configurations, eliminate manual configuration needs, and maintain consistent user experiences.

Prepare for security events

Develop and practice incident response plans by using automated tools to enable swift detection, investigation, and recovery from security events.

- Set up CloudWatch alarms for failed authentication attempts, changes to fleet security groups, modifications to image configurations, and unusual streaming session patterns.
- Document response procedures for common AppStream 2.0 security scenarios such as:
  - Unauthorized access attempts
    - Detection: Monitor authentication failures.
    - Response: Revoke user entitlements, review session logs, and update access policies.
  - Compromised streaming instances
    - Detection: Monitor instance behavior.
    - Response: Terminate affected sessions, replace fleet instances, and review security group configurations.
  - Data exfiltration attempts
    - Detection: Monitor file transfer activities.
    - Response: Review clipboard and file transfer logs, adjust file transfer permissions, and update data protection policies.
- Implement automated recovery processes for fleet instance replacement, security group restoration, user access reconfiguration, and application settings recovery.
- Use AWS services for security management, such as AWS Security Hub for security findings and Amazon GuardDuty for threat detection.

Reliability pillar

The reliability pillar of the AWS Well-Architected Framework addresses how well a system maintains its intended functionality and performance levels during expected operational periods throughout its lifespan. It provides comprehensive guidelines for building and maintaining dependable systems on AWS, including strategies for testing and validation across all stages of the workload lifecycle.

Key focus areas for applying this pillar to your AppStream 2.0 streaming environment:

- Fleet management and scaling
- Session reliability
- Application availability
- Recovery procedures

Automatically recover from failure

Monitor KPIs for business value to trigger automated responses that can predict, prevent, or recover from failures before they impact operations.

- Make sure that your IP subnet allocation accounts for expansion and availability.
- Monitor critical CloudWatch metrics to ensure service availability and performance, including fleet capacity metrics such as `AvailableCapacity` and `InUseCapacity`, and streaming quality metrics such as `StreamingSessionLatency`.
- Configure alerts for capacity thresholds, session health metrics, performance degradation, and fleet health status changes.
- Use built-in AppStream 2.0 automatic scaling capabilities to:
  - Configure minimum and maximum fleet capacity.
  - Set scaling policies based on capacity utilization.
  - Define scale-out and scale-in thresholds based on user experience metrics and business requirements instead of only technical metrics.
- Build a disaster recovery environment for your AppStream 2.0 environment. For more information, see the AWS blog post Disaster recovery considerations with Amazon AppStream 2.0.

Test recovery procedures

Cloud environments enable automated testing of failure scenarios and recovery procedures. These capabilities help you identify and fix vulnerabilities before real failures occur.

- **Fleet recovery testing.** Implement comprehensive fleet recovery testing across multiple scenarios:
  - Simulate instance termination to verify automatic scaling response.
  - Validate fleet minimum capacity maintenance.
  - Test instance replacement timing and user redirection.
  - Validate scaling policies effectiveness.
  - Test fleet capacity limits and overflow handling.
- **Session recovery testing.** Implement session recovery validation procedures:
  - Test disconnect and reconnect scenarios.
  - Verify application state preservation.
  - Test various network interruption scenarios.
  - Validate session timeout behaviors.
  - Verify user authentication persistence.
  - Verify temporary storage handling.

Scale horizontally to increase aggregate workload availability

Distribute your workload across multiple, smaller resources to minimize the impact of individual failures and to eliminate single points of failure.

- Deploy fleet instances across multiple Availability Zones.
- Configure appropriate minimum fleet capacity.
- Configure automatic scaling for fleets and set appropriate scaling thresholds.
- Monitor capacity utilization across the fleet.
- Deploy AppStream 2.0 stacks across multiple Regions. For more information, see the AWS blog post Optimize user experience with latency-based routing for Amazon AppStream 2.0.

Stop guessing capacity

Use the automatic scaling capabilities of the cloud to dynamically adjust resources based on demand. This helps prevent resource saturation while maintaining optimal efficiency.

- Monitor key metrics such as `CapacityUtilization`, `AvailableCapacity`, and `InUseCapacity` to understand capacity needs.
- Track fleet utilization trends across different time periods. Monitor daily patterns, weekly variations, monthly trends, and seasonal peaks.
- Set up scaling policies and configure scaling thresholds.
- Make sure that a sufficient gap exists between the current quotas and the maximum usage to accommodate failover.
- Accommodate fixed service quotas and constraints through your architecture.

Manage change through automation

Implement infrastructure changes through automation, including version-controlled changes to the automation code itself.

- Use IaC for fleet configuration.
- Implement consistent scaling policies.
- Use the Image Assistant CLI for consistent image creation.

Performance efficiency pillar

The performance efficiency pillar of the AWS Well-Architected Framework focuses on optimizing the use of cloud resources to meet or exceed performance goals while ensuring adaptability to fluctuating demands and emerging technologies. It emphasizes the importance of continuously fine-tuning systems to maintain peak efficiency in a dynamic cloud environment.

Key focus areas for applying this pillar to your AppStream 2.0 streaming environment:

- Instance type selection and optimization
- Streaming performance optimization
- Fleet capacity management

Democratize advanced technologies

Take advantage of cloud vendor-managed services for complex technologies so your team can focus on product development instead of infrastructure management.

- Configure appropriate instance types based on application requirements:
  - Select GPU-enabled instances for graphics-intensive applications.
  - Choose appropriate GPU families (such as Graphics G4dn or Graphics G5) based on application needs.
- Choose and configure one of the following authentication methods:
  - Set up integration with a SAML 2.0-based identity provider.
  - Configure user pool settings.
  - Integrate with AWS Directory Service.
- Enable and configure storage options based on user needs:
  - Set up home folders in Amazon S3 for Windows-based fleets.
  - Set up shared file systems in Amazon EFS for Linux-based fleets.
  - Configure persistent storage permissions.
  - Enable application settings persistence.

Go global in minutes

Use multi-Region deployment to improve global user experiences through reduced latency.

- Configure fleets in multiple AWS Regions by deploying fleets in Regions that are closest to your users while creating separate stacks for each Region.
- Implement cross-Region redirection to automatically redirect AppStream 2.0 users to the AppStream stacks that are closest to their current location.
- If you are using any of the optional features in AppStream 2.0, such as application settings persistence, home folders, or elastic fleets, you need to configure Amazon S3 cross-Region replication for user data for Windows-based fleets and cross-Region replication for Linux-based fleets.
- Replicate images across Regions. For more information, see Copy an image that you own to another AWS Region in Amazon AppStream 2.0 in the AWS documentation.
- For domain-joined fleets, make sure that an Active Directory infrastructure, including Active Directory Federation Services (AD FS) (unless you're using SAML 2.0 and Amazon Cognito as an alternative), is properly configured in the other Regions, and that you use AWS Directory Service for Microsoft Active Directory for multi-Region replication capabilities.
- Direct users to the lowest-latency AppStream 2.0 endpoints. For more information, see the AWS blog post Optimize user experience with latency-based routing for Amazon AppStream 2.0.

Use serverless architectures

Serverless architectures eliminate server management overhead and reduce costs by using cloud-managed services for compute functions.

Use AWS serverless services such as the following:

- AWS Lambda to automate tasks and integrate custom logic through event-driven functions
- Amazon S3 to provide scalable storage for AppStream 2.0 user data, application files, and session artifacts
- Amazon CloudWatch to provide monitoring, logging, and alerting for AppStream 2.0 performance and usage metrics
- Amazon Cognito to facilitate user authentication and access control for AppStream 2.0 applications

- Amazon API Gateway to create RESTful APIs to interface between AppStream 2.0 and other services or custom applications

Experiment more often

Cloud infrastructure enables rapid testing of various resource configurations to optimize performance and cost.

- Test different instance types to optimize performance and cost:
  - Compare stream performance across different instance families.
  - Evaluate GPU vs non-GPU instances for graphics applications.
  - Test memory-optimized instances for memory-intensive applications.
- Test application configurations by using Image Builder:
  - Create test images with different application configurations.
  - Validate application performance before deployment.
  - Test application compatibility with different instance types.
- Test fleet settings by using fleet capacity configurations such as minimum and maximum capacity, scaling policies, session settings such as maximum session duration, and disconnect timeout settings.

Consider mechanical sympathy

Choose cloud services based on your workload's specific requirements and usage patterns to ensure optimal performance and efficiency.

- Choose Graphics G5 instances for graphics-intensive applications, applications that require DirectX, OpenGL, OpenCL, or 3D visualization software.
- Select `stream.standard` instances for business applications, web browsers, and light graphics applications
- Monitor and adjust the streaming protocol based on CloudWatch metrics such as `StreamingSessionLatency`.
- Configure AppStream 2.0 in VPCs that are closest to your users, and use appropriate network bandwidth based on your application's requirements.

- Choose the appropriate fleet type based on application behavior. For example, choose single-session fleets for applications that require dedicated resources and multi-session fleets for applications that can share resources efficiently.
- Consider application compatibility with multi-session environments.
- Use the file system redirection feature to handle the interactions between remote and local applications. For more information, see the AWS blog post Launching local applications from an Amazon AppStream 2.0 streaming session.

Cost optimization pillar

The cost optimization pillar of the AWS Well-Architected Framework focuses on maximizing business value while minimizing expenditure. It helps ensure that every dollar you spend on cloud resources contributes effectively to achieving your organizational objectives.

Key focus areas for applying this pillar to your AppStream 2.0 streaming environment:

- Fleet capacity management and instance type selection
- Scaling and scheduling optimization
- Monitoring and analysis of usage patterns
- Cost allocation and tracking

Implement cloud financial management

Build dedicated organizational capability in cloud financial management and cost optimization through structured programs and processes to maximize cloud value and efficiency.

- Monitor AppStream 2.0 costs by using AWS Cost Explorer and usage reports to track streaming hours usage, analyze fleet instance costs, and monitor regional cost distribution.
- Plan and set cost controls by using AWS Budgets to set alerts for overall AppStream 2.0 service costs, create budget thresholds for the service, and monitor actual spending against budgeted amounts. For more information, see the AWS blog post How to use automation to optimize and control cost of Amazon AppStream 2.0.

Add a consumption model

Scale computing resources and costs based on actual usage patterns. For example, you can shut down non-production environments during off-hours to optimize spending.

- Choose the appropriate pricing model. For example, use always-on fleets for consistent usage and on-demand fleets for variable workloads.
- Select optimal instance types. For example, use `stream.standard` instances for general applications and use graphics instances (G4dn) only when required.

Measure overall efficiency

Calculate and track the cost-per-unit of business output to quantify efficiency improvements and guide optimization efforts.

- Track session efficiency.
- Monitor fleet utilization by using the following CloudWatch metrics:
  - `AvailableCapacity` to track unused capacity
  - `InUseCapacity` to measure actual usage
- Calculate and track per-session costs such as cost per streaming hour, cost per user, and cost per application.
- Implement the Cost Optimizer for AppStream 2.0 to monitor your builders.
- Compare costs between fleet types. For example, compare:
  - License costs for single sessions and multi-sessions
  - Resource utilization rates
  - User density per instance
- Use process tracking data to identify underutilized or unnecessary applications. For more information, see the AWS blog post Track user processes in Amazon AppStream 2.0 sessions.

Stop spending money on undifferentiated heavy lifting

AWS manages infrastructure operations and offers managed services so your organization can focus on business objectives instead of IT maintenance.

- Create and maintain application images by using Image Builder to package your applications, configure application settings, and test application compatibility.
- Configure fleet specifications by selecting appropriate instance types and defining scaling thresholds and setting desired capacity limits.
- Set up persistent storage options by configuring home folders in Amazon S3 for Windows-based fleets and shared file systems in Amazon EFS for Linux-based fleets. Set storage permissions and define retention policies.

Analyze and attribute expenditure

The cloud enables precise tracking of resource usage and costs per workload, which allows for accurate return on investment (ROI) measurements and targeted optimization opportunities.

- Implement a comprehensive tagging strategy for fleets for cost allocation, images for asset tracking, image builders for environment designation, and stacks for organizational grouping.
- Use AWS Cost and Usage Reports (AWS CUR) to break down AppStream 2.0 costs by tagged resources, and analyze costs per fleet, stack, and image.
- Use AWS Cost Explorer to visualize AppStream 2.0 spending trends, and compare costs across different dimensions such as Regions and instance types.
- Monitor and analyze fleet utilization rates, instance type efficiency, and streaming hours by application.
- Track unused reserved capacity, underutilized fleets or stacks, and idle periods in fleet usage.
- Calculate and track cost per user for each application, streaming hours per application, and user adoption rates for streamed applications.
- Set up detailed usage analysis by configuring AppStream 2.0 usage reports, using Amazon Athena to query usage data, and creating visualizations in Amazon QuickSight for cost and usage insights.
- Evaluate total cost considerations such as Windows Server licensing, application licensing models, and per-user licensing compared with per-device licensing.
- Use Amazon Athena to query and analyze home folder storage costs and usage patterns by user. For more information, see the AWS blog post How to report Amazon AppStream 2.0 home folder use with Amazon Athena.

Sustainability pillar

The sustainability pillar of the AWS Well-Architected Framework emphasizes minimizing your environmental footprint and optimizing energy usage and efficiency. It guides architects to make environmentally conscious decisions in their system designs and resource allocation strategies.

Key focus areas for applying this pillar to your AppStream 2.0 streaming environment:

- Understanding and optimizing resource allocation to match actual demand and minimize waste in streaming environments
- Analyzing and adapting to user consumption patterns to improve efficiency of application delivery and streaming sessions
- Selecting and using appropriate hardware configurations to maximize energy efficiency while meeting performance requirements
- Using AWS managed service capabilities to benefit from the economies of scale and built-in efficiency features offered by these services

Understand your impact

Monitor and optimize your workload's environmental impact by measuring resource efficiency and emissions per unit of output. Use this data to establish KPIs and guide sustainability improvements.

- Monitor fleet utilization patterns.
- Track streaming hours per user.
- Analyze fleet capacity usage trends.

Establish sustainability goals

Set measurable sustainability goals for each workload that align with organizational objectives. Focus on reducing resource intensity per transaction as you scale.

- Set targets for fleet utilization rates, instance type efficiency, and streaming hours optimization.
- Plan capacity based on actual usage patterns.

Maximize utilization

Optimize workload efficiency by right-sizing resources and maximizing utilization. Reduce idle capacity to minimize energy consumption and improve sustainability.

- Configure automatic scaling to match actual demand.
- Right-size fleet capacity based on usage patterns.
- Implement appropriate minimum and maximum capacity limits.
- Choose appropriate instance types for workloads.
- Monitor and optimize streaming session density.
- Reduce idle capacity during off-peak hours.

Anticipate and adopt new, more efficient hardware and software offerings

Stay informed about, and quickly adopt, new efficient technologies from partners and suppliers to continuously improve your workload's environmental impact.

- Use current generation instance types.
- Upgrade to newer instance types when available.
- Optimize application streaming settings.
- Configure appropriate streaming protocols.
- Update to the latest AppStream 2.0 features.

Used managed services

Take advantage of shared cloud services and managed solutions to maximize resource utilization efficiency while minimizing environmental impact through automated scaling and lifecycle management.

- Use Amazon S3 for user storage for Windows-based fleets and Amazon EFS for shared file systems for Linux-based fleets.
- Implement CloudWatch for monitoring.

- Configure IAM for access management.

Reduce the downstream impact of your cloud workloads

Design services to minimize client-side resource requirements, to reduce energy consumption and extend device lifespans for users.

- Adjust maximum session duration to prevent unnecessary resource consumption.
- Configure appropriate session timeouts.
- Set disconnect timeout policies.
- Implement session persistence policies where needed.

Resources

AWS documentation

- AWS Well-Architected Framework
- Amazon AppStream 2.0 Administration Guide
- Amazon CloudWatch User Guide
- Amazon EFSS User Guide
- Amazon S3 User Guide
- IAM User Guide

AWS blog posts

- Active Directory Group Membership Based AppStream 2.0 Application Targeting
- Create a Single Identity Provider for all your Amazon AppStream 2.0 Stacks with Azure AD
- Configuring Windows Remote Assistance for Amazon WorkSpaces and Amazon AppStream 2.0
- Creating an AS2TrustedDomains DNS TXT record to redirect the Amazon AppStream 2.0 native client to a third-party identity provider
- Creating custom logging and Amazon CloudWatch alerting in Amazon AppStream 2.0
- Cross-Region redirection with Geo Targetly and Amazon AppStream 2.0
- Cross-account resources and Amazon AppStream 2.0
- Enable federation with Bio-key PortalGuard and Amazon AppStream 2.0
- Enabling Federation with SimpleSAMLphp and Amazon AppStream 2.0
- Enabling identity federation with Duo Single Sign-On and Amazon AppStream 2.0
- Enabling Identity Federation with Shibboleth and Amazon AppStream 2.0
- Failover strategies for on-premises VDI with Amazon End User Computing
- How Amazon Uses Amazon AppStream 2.0 to Provide Data Scientists and Analysts with Access to Sensitive Data
- How to configure certificate-based authentication for Amazon AppStream 2.0
- How to use Okta claims with application entitlements for Amazon AppStream 2.0

- Managing Computer Labs on Amazon AppStream 2.0 with Open Source Virtual Application Management
- Methods of allocating your AppStream 2.0 costs to your business units
- Monitoring Amazon AppStream 2.0 with Amazon OpenSearch Service and Amazon Kinesis Data Firehose
- Network Separation and Data Sanitization using Amazon WorkSpaces, Amazon AppStream 2.0, and Amazon Macie
- OneLogin SSO with Amazon AppStream 2.0
- Optimize your Amazon Connect call audio path with Amazon AppStream 2.0
- Persistent storage for Amazon AppStream 2.0 Linux Fleets on Amazon Elastic File System
- Redirect an Okta SAML app to the Amazon AppStream 2.0 native client
- Simplify Amazon AppStream 2.0 image management with Application Masking
- Stream applications at a lower cost with Amazon AppStream 2.0 Elastic fleets and Linux compatibility
- Streaming from interface VPC endpoints for Regulated environments with AppStream 2.0
- Use Amazon AppStream 2.0 application entitlements with Azure AD
- User Issue Reporter for Amazon AppStream 2.0
- Using Amazon AppStream 2.0 application entitlements with Google Workspace
- Using Auth0 with Microsoft Active Directory on Amazon AppStream 2.0
- Using Microsoft AppLocker to manage application experience on Amazon AppStream 2.0
- Using Python to power an AppStream 2.0 Linux Imaging Assistant GUI
- Web application redirection options for the AppStream 2.0 Client

Document history

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

Change	Description	Date
Initial publication	—	July 23, 2025

AWS Prescriptive Guidance glossary

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

Numbers

7 Rs

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

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

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

A

ABAC

See attribute-based access control.

abstracted services

See managed services.

ACID

See atomicity, consistency, isolation, durability.

active-active migration

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

active-passive migration

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

aggregate function

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

AI

See artificial intelligence.

AIOps

See artificial intelligence operations.

anonymization

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

anti-pattern

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

application control

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

application portfolio

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

artificial intelligence (AI)

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

artificial intelligence operations (AIOps)

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

asymmetric encryption

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

atomicity, consistency, isolation, durability (ACID)

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

attribute-based access control (ABAC)

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

authoritative data source

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

Availability Zone

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

AWS Cloud Adoption Framework (AWS CAF)

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

AWS Workload Qualification Framework (AWS WQF)

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

B

bad bot

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

BCP

See business continuity planning.

behavior graph

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

big-endian system

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

binary classification

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

bloom filter

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

blue/green deployment

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

bot

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

botnet

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

branch

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

break-glass access

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

brownfield strategy

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

buffer cache

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

business capability

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

business continuity planning (BCP)

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

C

CAF

See AWS Cloud Adoption Framework.

canary deployment

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

CCoE

See Cloud Center of Excellence.

CDC

See change data capture.

change data capture (CDC)

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

chaos engineering

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

CI/CD

See continuous integration and continuous delivery.

classification

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

client-side encryption

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

Cloud Center of Excellence (CCoE)

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

cloud computing

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

cloud operating model

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

cloud stages of adoption

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

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

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

CMDB

See configuration management database.

code repository

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

cold cache

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

cold data

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

computer vision (CV)

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

configuration drift

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

configuration management database (CMDB)

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

conformance pack

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

continuous integration and continuous delivery (CI/CD)

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

CV

See computer vision.

D

data at rest

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

data classification

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

data drift

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

data in transit

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

data mesh

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

data minimization

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

data perimeter

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

data preprocessing

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

data provenance

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

data subject

An individual whose data is being collected and processed.

data warehouse

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

database definition language (DDL)

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

database manipulation language (DML)

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

DDL

See database definition language.

deep ensemble

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

deep learning

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

defense-in-depth

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

delegated administrator

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

deployment

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

development environment

See environment.

detective control

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

development value stream mapping (DVSM)

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

digital twin

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

dimension table

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

disaster

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

disaster recovery (DR)

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

DML

See database manipulation language.

domain-driven design

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

DR

See disaster recovery.

drift detection

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

DVSM

See development value stream mapping.

E

EDA

See exploratory data analysis.

EDI

See electronic data interchange.

edge computing

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

electronic data interchange (EDI)

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

encryption

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

encryption key

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

endianness

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

endpoint

See service endpoint.

endpoint service

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

enterprise resource planning (ERP)

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

envelope encryption

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

environment

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

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

epic

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

ERP

See enterprise resource planning.

exploratory data analysis (EDA)

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

F

fact table

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

fail fast

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

fault isolation boundary

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

feature branch

See branch.

features

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

feature importance

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

feature transformation

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

few-shot prompting

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

FGAC

See fine-grained access control.

fine-grained access control (FGAC)

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

flash-cut migration

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

FM

See foundation model.

foundation model (FM)

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

G

generative AI

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

geo blocking

See geographic restrictions.

geographic restrictions (geo blocking)

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

Gitflow workflow

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

golden image

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

greenfield strategy

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

guardrail

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

H

HA

See high availability.

heterogeneous database migration

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

high availability (HA)

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

historian modernization

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

holdout data

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

homogeneous database migration

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

hot data

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

hotfix

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

hypercare period

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

I

IaC

See infrastructure as code.

identity-based policy

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

idle application

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

IIoT

See industrial Internet of Things.

immutable infrastructure

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

inbound (ingress) VPC

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

incremental migration

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

Industry 4.0

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

infrastructure

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

infrastructure as code (IaC)

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

industrial Internet of Things (IIoT)

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

inspection VPC

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

Internet of Things (IoT)

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

interpretability

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

IoT

See Internet of Things.

IT information library (ITIL)

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

IT service management (ITSM)

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

ITIL

See IT information library.

ITSM

See IT service management.

L

label-based access control (LBAC)

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

landing zone

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

large language model (LLM)

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

large migration

A migration of 300 or more servers.

LBAC

See label-based access control.

least privilege

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

lift and shift

See 7 Rs.

little-endian system

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

LLM

See large language model.

lower environments

See environment.

M

machine learning (ML)

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

main branch

See branch.

malware

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

managed services

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

manufacturing execution system (MES)

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

MAP

See Migration Acceleration Program.

mechanism

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

member account

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

MES

See manufacturing execution system.

Message Queuing Telemetry Transport (MQTT)

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

microservice

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

microservices architecture

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

Migration Acceleration Program (MAP)

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

migration at scale

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

migration factory

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

migration metadata

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

migration pattern

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

Migration Portfolio Assessment (MPA)

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

Migration Readiness Assessment (MRA)

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

migration strategy

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

ML

See machine learning.

modernization

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

modernization readiness assessment

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

monolithic applications (monoliths)

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

MPA

See Migration Portfolio Assessment.

MQTT

See Message Queuing Telemetry Transport.

multiclass classification

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

mutable infrastructure

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

O

OAC

See origin access control.

OAI

See origin access identity.

OCM

See organizational change management.

offline migration

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

OI

See operations integration.

OLA

See operational-level agreement.

online migration

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

OPC-UA

See Open Process Communications - Unified Architecture.

Open Process Communications - Unified Architecture (OPC-UA)

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

operational-level agreement (OLA)

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

operational readiness review (ORR)

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

operational technology (OT)

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

operations integration (OI)

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

organization trail

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

organizational change management (OCM)

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

origin access control (OAC)

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

origin access identity (OAI)

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

ORR

See operational readiness review.

OT

See operational technology.

outbound (egress) VPC

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

P

permissions boundary

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

personally identifiable information (PII)

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

PII

See personally identifiable information.

playbook

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

PLC

See programmable logic controller.

PLM

See product lifecycle management.

policy

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

polyglot persistence

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

portfolio assessment

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

predicate

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

predicate pushdown

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

preventative control

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

principal

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

privacy by design

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

private hosted zones

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

proactive control

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

AWS Control Tower documentation and see Proactive controls in *Implementing security controls on AWS*.

product lifecycle management (PLM)

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

production environment

See environment.

programmable logic controller (PLC)

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

prompt chaining

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

pseudonymization

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

publish/subscribe (pub/sub)

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

Q

query plan

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

query plan regression

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

R

RACI matrix

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

RAG

See Retrieval Augmented Generation.

ransomware

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

RASCI matrix

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

RCAC

See row and column access control.

read replica

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

re-architect

See 7 Rs.

recovery point objective (RPO)

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

recovery time objective (RTO)

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

refactor

See 7 Rs.

Region

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

regression

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

rehost

See 7 Rs.

release

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

relocate

See 7 Rs.

replatform

See 7 Rs.

repurchase

See 7 Rs.

resiliency

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

resource-based policy

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

responsible, accountable, consulted, informed (RACI) matrix

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

responsive control

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

retain

See 7 Rs.

retire

See 7 Rs.

Retrieval Augmented Generation (RAG)

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

rotation

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

row and column access control (RCAC)

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

RPO

See recovery point objective.

RTO

See recovery time objective.

runbook

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

S

SAML 2.0

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

SCADA

See supervisory control and data acquisition.

SCP

See service control policy.

secret

In AWS Secrets Manager, confidential or restricted information, such as a password or user credentials, that you store in encrypted form. It consists of the secret value and its metadata. The secret value can be binary, a single string, or multiple strings. For more information, see What's in a Secrets Manager secret? in the Secrets Manager documentation.

security by design

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

security control

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

security hardening

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

security information and event management (SIEM) system

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

security response automation

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

server-side encryption

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

service control policy (SCP)

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

service endpoint

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

service-level agreement (SLA)

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

service-level indicator (SLI)

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

service-level objective (SLO)

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

shared responsibility model

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

SIEM

See security information and event management system.

single point of failure (SPOF)

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

SLA

See service-level agreement.

SLI

See service-level indicator.

SLO

See service-level objective.

split-and-seed model

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

SPOF

See single point of failure.

star schema

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

strangler fig pattern

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

subnet

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

supervisory control and data acquisition (SCADA)

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

symmetric encryption

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

synthetic testing

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

system prompt

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

T

tags

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

target variable

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

task list

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

test environment

See environment.

training

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

transit gateway

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

trunk-based workflow

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

trusted access

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

tuning

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

two-pizza team

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

U

uncertainty

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

undifferentiated tasks

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

upper environments

See environment.

V

vacuuming

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

version control

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

VPC peering

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

vulnerability

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

W

warm cache

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

warm data

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

window function

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

workload

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

workstream

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

WORM

See write once, read many.

WQF

See AWS Workload Qualification Framework.

write once, read many (WORM)

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

Z

zero-day exploit

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

zero-day vulnerability

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

zero-shot prompting

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

zombie application

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