AWS Cloud Adoption Framework: Platform perspective

AWS Prescriptive Guidance

AWS Prescriptive Guidance: AWS Cloud Adoption Framework: Platform perspective

Table of Contents

AWS Cloud Adoption Framework: Platform perspective

Amazon Web Services (contributors)

October 2023 (document history)

Digital transformation is the single largest enabler for executives to improve customer experience, innovation, and flexibility. It uses machine learning (ML), artificial intelligence (AI), big data, and the speed and scale of the cloud to meet changing business conditions and evolving customer needs.

Amazon Web Services (AWS) is the world's most comprehensive and broadly adopted cloud platform. It can help you transform your organization while reducing business risk, improving environmental, social, and governance (ESG) performance, growing revenue, and improving operational efficiency.

The AWS Cloud Adoption Framework (AWS CAF) uses AWS best practices to help you accelerate your business outcomes. Use the AWS CAF to identify and prioritize transformation opportunities, evaluate and improve your cloud readiness, and iteratively evolve your transformation roadmap.

AWS CAF groups its guidance in six perspectives: *Business*, *People*, *Governance*, *Platform*, *Security*, and *Operations*. Each perspective is covered in a separate guide. This guide covers the *Platform* perspective, which focuses on accelerating the delivery of your cloud workloads with an enterprise-grade, scalable, hybrid cloud environment.

Introduction

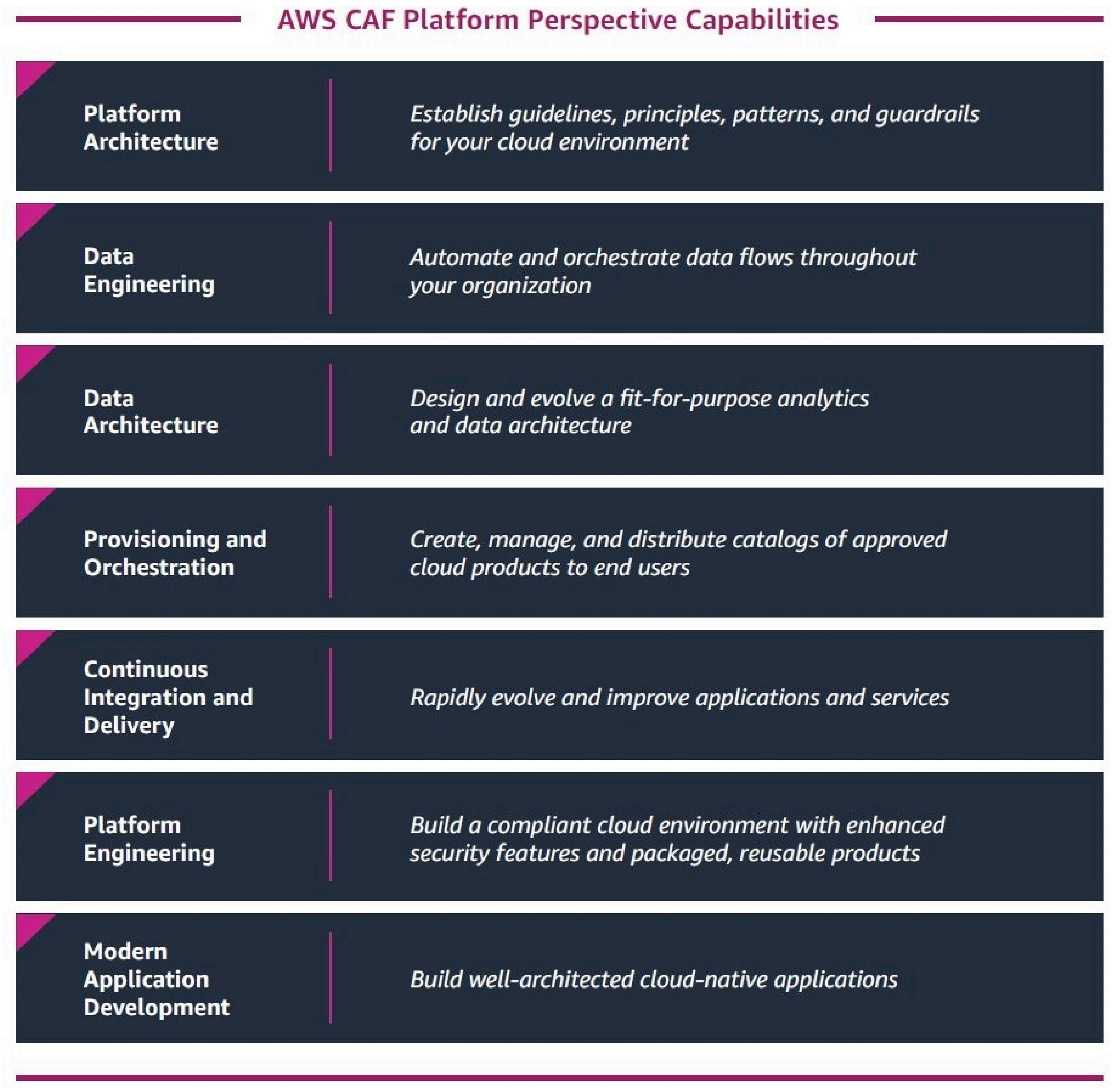
Millions of customers, including the fastest-growing startups, largest enterprises, and leading government organizations, use AWS. (See Customer Success Stories on the AWS website.) They can migrate and modernize legacy workloads, become more data-driven, automate and optimize business processes, and reinvent operating models. They are able to improve their business outcomes by reducing business risk, improving environmental, social, and governance (ESG) performance, increasing revenue, and improving operational efficiency.

The organizational ability to effectively use the cloud to digitally transform (organizational cloud readiness) is bolstered by a set of foundational capabilities. A *capability* is an organizational ability to use processes to deploy resources (people, technology, and any other tangible or intangible assets) to achieve a particular outcome. The AWS CAF identifies these capabilities and provides prescriptive guidance that thousands of organizations around the world have successfully used to improve their cloud readiness and accelerate their cloud transformation journeys.

AWS CAF groups its capabilities in six perspectives:

- Business
- People
- Governance
- Platform
- Security
- Operations

The Platform perspective focuses on accelerating the delivery of your cloud workloads with an enterprise-grade, scalable, hybrid cloud environment. This environment comprises seven capabilities shown in the following diagram. These capabilities are managed by stakeholders who are functionally related in their cloud transformation journey. Typical stakeholders include the chief technology officer (CTO), technology leaders, architects, and engineers.



AWS CAF Platform Perspective Capabilities

Platform Architecture	*Establish guidelines, principles, patterns, and guardrails for your cloud environment*
Data Engineering	*Automate and orchestrate data flows throughout your organization*
Data Architecture	*Design and evolve a fit-for-purpose analytics and data architecture*
Provisioning and Orchestration	*Create, manage, and distribute catalogs of approved cloud products to end users*
Continuous Integration and Delivery	*Rapidly evolve and improve applications and services*
Platform Engineering	*Build a compliant cloud environment with enhanced security features and packaged, reusable products*
Modern Application Development	*Build well-architected cloud-native applications*

These capabilities are discussed in detail in the following sections of this guide. Each section provides guidelines on how to start, advance, and ultimately excel in a particular capability.

- Platform architecture
- Platform engineering
- Data architecture

- Data engineering
- Provisioning and orchestration
- Modern application development
- Continuous integration and continuous delivery (CI/CD)

The Platform perspective is a pivotal piece of the AWS CAF. It is the nexus in which the decisions made across all other perspectives converge to provide business agility and value. The decisions made here help or hinder your business objectives at a foundational level. The AWS CAF Platform perspective facilitates the creation of an enterprise-grade, scalable, cloud environment that underpins your organization's transformation. Through this perspective, the AWS CAF guides you in establishing a robust platform that can enable your cloud journey, ultimately leading to significant business transformation and growth.

As you work through the Platform perspective, take into account the cross-functional connections with business leaders that need to be developed, and the value they bring to your teams and organization. Place additional focus on operating model changes and team topologies to ensure that requirements are met. In addition, look to develop the skills your teams require to build the platform and enable its usage across application teams. Keep in mind the people, business, governance, security, and operational objectives of your organization as you make these decisions—these are pivotal in ensuring the adoption of the platform and the success of your efforts.

AWS and the AWS Partner Network provide tools and services, such as workshops and trainings, that can help you on this journey to implement and improve your security posture. AWS Professional Services is a global team of experts who can help you achieve specific outcomes related to your cloud transformation through a collection of AWS CAF-aligned offerings.

Platform architecture

Establish and maintain guidelines, principles, patterns, and guardrails for your cloud environment.

A well-architected cloud environment helps you accelerate implementation, reduce risk, and drive cloud adoption. The platform architecture capability creates consensus within your organization for enterprise standards that drive cloud adoption. You define best practice blueprints and guardrails to facilitate authentication, security, networking, and logging and monitoring. Additionally you take into consideration and plan for workloads you might need to retain on premises due to latency, data processing, or data residency requirements and evaluate hybrid cloud use cases such as cloud bursting, backup and disaster recovery to the cloud, distributed data processing, and edge computing.

Start

Define a multi-account strategy

A good multi-account strategy considers scale and operational efficiency concerns. This means isolating your workloads into a logical pattern that best meets your operational needs. We suggest that you start with a foundational set of accounts to accommodate centralized and decentralized services in your enterprise. You can centralize security, financial, and operational functions to effectively manage and govern your distributed and autonomous teams and accounts. You will want to align across your organization to understand how the platform and your workloads will be segmented and managed. Understanding this structure helps you ensure that security principles are in place for authentication and authorization while aligning to evolving acceptable use policies for the platform.

Define preventative controls

Plan for a secure, multi-account environment with an embedded set of default controls (*guardrails*). Begin to understand and use a mechanism such as service control policies (SCPs) to manage service use across your organization, including the AWS Regions that are available for consumption within your cloud platform. Policies provide a centralized mechanism for controlling the maximum permissions available for all accounts and ensuring that they adhere to the organization's access control guidelines.

Define organizational unit structure

Organizational units (OUs) serve as a practical way to manage and categorize accounts based on regulatory requirements and software development lifecycle (SDLC) environments. By using OUs, organizations streamline the process of applying for appropriate policies and permissions across their cloud infrastructure. Workload OUs are specifically designed for accounts that support application infrastructure resources, and ensure that the right policies are enforced. Using OUs and SCPs help enhance your organization's cloud infrastructure's security and compliance while also ensuring the smooth operation of your applications and services. This ultimately leads to a more efficient and robust cloud adoption process.

Define network connectivity

Network connectivity is a crucial aspect of any cloud infrastructure that supports the creation of secure, scalable, and highly available networks to support applications and workloads. A well-designed network provides consistently high performance and ensures seamless operations across different environments.

When you design your network architecture, consider if you have workloads that you want to retain on premises due to latency, data processing, or data residency requirements. By evaluating hybrid cloud use cases such as cloud bursting, backup and disaster recovery to the cloud, distributed data processing, and edge computing, you can identify the key requirements for the following aspects:

- **Connectivity to and from the internet.** This aspect involves providing secure and reliable connections between your applications or workloads and the internet. This connectivity is essential for facilitating access to web-based resources, enabling communications between users and applications, and ensuring that your services are accessible to the public when needed.
- **Connectivity across your cloud environments.** This area focuses on establishing robust connections among various components and services within your cloud infrastructure. It ensures that data and resources are easily shared and accessed across different cloud services, promoting efficient collaboration and smoother operations. A key consideration here is your use of virtual private clouds (VPCs). To keep things simple, consider creating standards on how VPCs are created and tracked. Consider creating these standards programmatically, and plan to use an IP address management (IPAM) solution. Allocate enough IP space to allow for growth, and design subnet structures for easy troubleshooting when using multiple Availability Zones. Make sure to follow security best practices for VPCs when you design and implement network connectivity.
- **Connectivity between your on-premises network and your cloud environments.** This aspect deals with the integration of your on-premises infrastructure with your cloud-based

environment. By creating secure and reliable connections between the two, organizations benefit from the advantages of hybrid architectures. For example, you can use on-premises resources and cloud services simultaneously for improved performance, scalability, and cost optimization.

By addressing these three key areas of network connectivity, you can build a robust cloud infrastructure that supports your applications and workloads effectively, so you can capitalize on the benefits of cloud adoption. Take note of networking requirements, and create a simple design that enables you to scale in accordance with your multi-account strategy.

Define DNS strategy

A well-planned DNS strategy helps you avoid complications as your cloud environments grow. If you maintain on-premises DNS capabilities, we recommend that you design hybrid DNS architectures that use on-premises DNS infrastructure along with cloud DNS for any cloud-based DNS requirements. Integrate DNS resolution with on-premises DNS environments by using resolver endpoints and forwarding rules. Use private hosted zones to hold information about how you want cloud DNS to respond to queries for a domain and its subdomains within one or more networks.

Define tagging standards

Tagging resources is an essential practice to manage costs effectively and identify ownership of resources. Consider how your organization will further allow consumption in the cloud, including the use of specific services within the platform. Define a tagging strategy that tracks which resources are being deployed by which teams. Take inputs from the AWS CAF Operations perspective and use tags to automate tasks for your deployed infrastructure.

Additionally, by tagging resources with relevant metadata, you can group and track your spending based on your organizational requirements dictated in the Cloud Financial Management (CFM) capability in the AWS CAF Governance perspective. Identify a mechanism for reporting that supports your accounting and financial practices, including actions to be taken when financial policies are violated.

Define an observability strategy

Establishing an observability strategy is a critical step toward optimizing and securing your cloud architecture. This strategy revolves around transforming the metrics and logs produced by your cloud services into actionable insights for strategic decision-making. Prioritize monitoring key performance indicators and setting up alerts to preemptively address potential issues. To prevent tool proliferation, optimize costs, and focus on what matters most to your organization, incorporate this observability strategy across both your platform and applications. For further guidance, see our presentation on Developing an observability strategy (AWS re:Invent 2022).

Advance

Define proactive and detective controls

To advance, your organization must identify the need for proactive and detective controls (*guardrails*) within the environment. Create policies that define the guardrails or limits that roles and users have in the accounts located within an organizational unit (OU). Review any default detective guardrails for the platform, and choose which guardrails to apply. Create additional preventive and detective controls as required, and group them by OUs to align them to your multi-account strategy. Consider which organizational tools and mechanisms you need to inspect non-compliant resources that are identified by detective controls.

Define standards for service onboarding

Create standards for the acceptable use of the platform and the patterns associated with service consumption and how that will be governed. Consider which initial services are allowed for use. Create a document that outlines these standards and publish them to users and operators of the platform. Ensure that these standards adapt over time to meet the changing objectives of the organization and the evolving capabilities of cloud computing.

Define patterns and principles

Consider which architectural patterns will be allowed within your organization by using inputs from application owners, and begin to define blueprints for standardization. Standardization allows for greater governance and lower administrative burden as you scale in the cloud. Define patterns that will use infrastructure as code (IaC) and plan for a simplified deployment model by using a service catalog that's integrated into your change control processes and IT service management (ITSM) systems. Define how these blueprints will be used and the circumstances for allowing exceptions. Plan for those exceptions and their governance, with considerations for authentication, security monitoring, and guardrails.

Excel

Define remediation patterns

Consider how to annotate and prioritize your detective guardrail findings so they can be remediated in accordance with your security and compliance frameworks. Plan to use automation to detect out-of-policy provisioning of resources, including those that violate budgetary and tagging policies. Identify the capabilities needed to set and measure service-level objectives while updating your runbooks and playbooks. Set periodic reviews of these practices and a feedback mechanism to capture data related to platform evolution. Define mechanisms to create and update runbooks and playbooks accordingly.

Communicate and refine policies

Create a centralized content management system for all documentation and distribute it to the users and operators of the platform. Create a mechanism to capture feedback for future consideration on changes to the policy.

Understand financial management capabilities

Organizations thrive when they maintain a transparent and comprehensive understanding of their budget. This empowers them to make well-informed decisions, allocate resources efficiently, and accomplish their strategic objectives. A clear view of the budget helps organizations excel by facilitating informed decision-making, effective resource allocation, cost control, performance measurement, and the maintenance of accountability and compliance. This ultimately results in a more efficient, financially stable, and prosperous organization. When you have a successful tagging strategy, you can use cost filters in AWS Budgets to filter expenses based on resource tags. This helps you create a budget that's tailored to specific projects, departments, environments, or other criteria, further enhancing financial management capabilities. You can associate cost allocation tags and AWS Cost Categories with tags to drive financial insights and transparency when reporting on cost.

Platform engineering

Build a secure, compliant multi-account cloud environment with packaged, reusable cloud products.

To support innovation by enabling development teams, the platform needs to adapt at a rapid pace to keep up with the demands of the business. (See the AWS CAF Business perspective.) It must do so while being flexible enough to adapt to product management demands, rigid enough to adhere to security constraints, and fast enough to enable operational needs. This process requires the building of a compliant multi-account cloud environment with enhanced security features, and packaged, reusable cloud products.

An effective cloud environment allows your teams to easily provision new accounts while ensuring that those accounts conform to organizational policies. A curated set of cloud products enables you to codify best practices, helps you with governance, and helps increase the speed and consistency of your cloud deployments. Deploy your best practice blueprints, and detective and preventative guardrails. Integrate your cloud environment with your existing landscape to enable desired hybrid cloud use cases.

Automate the account provisioning workflow and use multiple accounts to support your security and governance goals. Set up connectivity between your on-premises and cloud environments as well as between different cloud accounts. Implement federation between your existing identity provider (IdP) and your cloud environment so that users can authenticate by using their existing login credentials. Centralize logging, establish cross-account security audits, create inbound and outbound DNS resolvers, and get dashboard visibility into your accounts and guardrails.

Evaluate and certify cloud services for consumption in alignment with corporate standards and configuration management. Package and continuously improve enterprise standards as self-service deployable products and consumable services. Leverage infrastructure as code (IaC) to define configurations in a declarative way. Create enablement teams to evangelize the platform to developers and business users and allow them to build integrations that accelerate adoption across your organization.

Completing the tasks discussed in the following sections requires you to build capabilities and teams to evolve your organizations toward modern platform engineering. For technical details, see the Establishing your Cloud Foundation on AWS whitepaper.

Start

Build a landing zone and deploy guardrails

As you start your journey to mature platform engineering, you must first deploy your landing zone with detective and preventative guardrails as defined in the platform architecture capability. Guardrails ensure that organizational standards aren't violated as application owners consume cloud resources. With this mechanism, you automate the account provisioning workflow to use multiple accounts that support your security and governance goals.

Establish authentication

Implement identity management and access control across all environments, systems, workloads, and processes in accordance with standards dictated in the AWS CAF Security perspective. For workforce identities, restrict the use of AWS Identity and Access Management (IAM) users and instead rely on an identity provider that enables you to manage identities in a centralized place. This makes it easier to manage access across multiple applications and services, because you are creating, managing, and revoking access from a single location. Use existing processes to manage the creation, update, and removal of access to include your AWS environments.

Deploy your network

In accordance with your platform architecture designs, create a centralized network account to control inbound and outbound traffic to and from your environment. We recommend that you design your networks for rapidly provisioned connectivity between your on-premises network and your AWS environments, to and from the internet, and across your AWS environments. Centralizing your network management enables you to deploy network controls to isolate networks and connectivity across your environment by using preventive and reactive controls.

Collect, aggregate, and protect event and log data

Use Amazon CloudWatch cross-account observability. It provides a unified interface to search, visualize, and analyze metrics, logs, and traces across your linked accounts, and eliminates account boundaries.

If your organization has specific compliance requirements for centralized log control and security, consider setting up a dedicated log archive account. This offers a centralized, encrypted repository specifically for log data. Enhance the security of this archive by regularly rotating encryption keys.

Implement robust policies for protecting sensitive log data, using masking techniques as necessary. Use log aggregation for compliance, security, and audit logs, and ensure the use of strict guardrails and identity constructs to prevent unauthorized changes to log configurations.

Establish controls

In accordance with the definitions from the AWS CAF Security perspective, deploy foundational security capabilities that meet your business requirements. Deploy additional preventative and detective controls, and provision those programmatically and consistently across all your accounts where required. Integrate detective controls into operational tooling as defined by the platform architecture capability so that non-compliant resources can be reviewed by operational mechanisms.

Implement cloud financial management

In accordance with the AWS CAF Governance perspective, implement cost allocation tags, and AWS Cost Categories that align your organization's tagging strategy with financial accountability for cloud consumption. AWS Cost Categories let you charge or show cloud charges back to internal cost centers by using tools such as AWS Cost Explorer and billing data published in AWS Cost and Usage Report.

Advance

Build infrastructure automation

Before you proceed, evaluate and certify cloud services for consumption in alignment with your platform architecture. Then, package and continuously improve enterprise standards as deployable products and consumable services, and use infrastructure as code (IaC) to define configurations in a declarative way. Infrastructure automation mimics software development cycles by allowing access to specific services in each account with role-based access control (RBAC) or attribute-based access control (ABAC). Deploy a method to rapidly provision new accounts and align them with your service and incident management capabilities by using APIs, or develop self-service capabilities. Automate network integration and IP allocation as accounts are created to ensure compliance and network security. Integrate new accounts with your IT service management (ITSM) solution by using native connectors that are configured to operate with AWS. Update your playbooks and runbooks as appropriate.

Provide centralized observability services

To achieve effective cloud observability, your platform should support real-time search and analysis of both local and centralized log data. As your operations scale, your platform's ability to index, visualize, and interpret log, metrics, and traces is key to turning raw data into actionable insights.

By correlating logs, metrics, and traces, you can extract actionable conclusions and develop targeted, informed responses. Establish rules that allow proactive responses to security events or patterns that are identified in your logs, metrics, or traces. As your AWS solutions expand, ensure that your monitoring strategy scales in tandem to maintain and enhance your observability capabilities.

Implement systems management and AMI governance

Organizations that use Amazon Elastic Compute Cloud (Amazon EC2) instances extensively require operational tooling to manage instances at scale. Software asset management, endpoint detection and response, inventory management, vulnerability management, and access management are foundational capabilities for many organizations. These capabilities are often delivered through software agents that are installed on instances. Develop a capability to package agents and other custom configurations into Amazon Machine Images (AMIs), and make these AMIs available to consumers of the cloud platform. Use preventative and detective controls that govern the use of these AMIs. AMIs should contain tooling that enables management of long-running EC2 instances at scale, particularly for mutable Amazon EC2 workloads that don't consume new AMIs on a regular basis. You can use AWS Systems Manager at scale to automate agent upgrades, collect system inventory, access EC2 instances remotely, and patch operating system vulnerabilities.

Manage credential use

In accordance with the AWS CAF Security perspective, implement roles and temporary credentials. Use tooling to manage remote access to instances or on-premises systems by using a pre-installed agent without storing secrets. Reduce reliance on long-term credentials, and scan for hardcoded credentials in your IaC templates. If you can't use temporary credentials, use programmatic tools such as application tokens and database passwords to automate credential rotation and management. Codify users, groups, and roles by using principles of least privilege with IaC, and prevent the manual creation of identity accounts by using guardrails.

Establish security tooling

Security monitoring tools should support granular security monitoring across infrastructure, applications, and workloads and provide aggregated views for pattern analysis. As with all other security management tools, you should extend your extended detection and response (XDR) tools to provide functions to assess, detect, respond to, and remediate the security of your applications, resources, and environments on AWS in accordance with requirements defined in the AWS CAF Security perspective.

Excel

Source and distribute identity constructs with automation

Codify and version identity constructs such as roles, policies, and templates with IaC tools. Use policy validation tools to check for security warnings, errors, general warnings, suggested changes to your IAM policies, and other findings. Where appropriate, deploy and remove identity constructs that provide temporary access to the environment in an automated manner, and prohibit deployment by individuals who are using the console.

Add detection and alerts for anomalous patterns across environments

Proactively assess environments for known vulnerabilities and add detection for unusual event and activity patterns. Review findings and make recommendations to platform architecture teams for changes that drive further efficiency and innovation.

Analyze and model for threats

Implement continuous monitoring and measurement against industry and security benchmarks in accordance with the requirements from the AWS CAF Security perspective. When you implement your instrumentation approach, determine which types of event data and information will best inform your security management functions. This monitoring encompasses several attack vectors, including service usage. Your security foundations should include a comprehensive capability for secure logging and analytics across your multi-account environments that includes the ability to correlate events from multiple sources. Prevent changes to this configuration with specific controls and guardrails.

Continuously collect, review, and refine permissions

Record changes to identity roles and permissions and implement alerts when detective guardrails detect deviations from your expected configuration state. Use aggregated and pattern identification tools to review your centralized collection of events and refine permissions as required.

Select, measure, and continuously improve your platform metrics

To enable successful platform operations, establish and routinely review comprehensive metrics. Ensure that they align with organizational goals and stakeholder needs. Track both platform performance and improvement metrics, and combine operational parameters such as patch, backup, and compliance by using team enablement and tool adoption indicators.

Use CloudWatch cross-account observability for efficient metric management. This service streamlines data aggregation and visualization to enable informed decisions and targeted enhancements. Use these metrics as indicators of success and drivers of change to foster an environment of continuous improvement.

Data architecture

Design and evolve a fit-for-purpose data and analytics architecture.

A well-designed data and analytics architecture is essential to gain actionable insights. By designing and evolving a fit-for-purpose data and analytics architecture, organizations reduce complexity, cost, and technical debt while unlocking valuable insights from their ever-growing data volumes. By aligning with AWS CAF principles, businesses can create a data architecture that seamlessly integrates with their existing platform. This alignment positions organizations to capitalize on the advantages offered by modern data processing and analytics technologies.

The data and analytics architecture is the blueprint of an organization's capabilities to derive value out of data. It helps the organization gain new business insights and is a catalyst for business growth. To support business needs, a modern data architecture should align with short-term and long-term business goals and be unique to the organization's cultural and contextual requirements. In today's world, the successful implementation and adoption of a data and analytics architecture are based on the principle of enabling the right data at the right time to the right consumer.

This is achieved by planning and organizing how an organization's data assets are modeled, physically or logically, how the data is secured, and how these data models interact with one another to address business problems and to derive unknown patterns and generate insights.

Start

Define overarching capability

In the current business environment, it is critical for the modern data analytics platform to derive value from data to support various domains in the organization. Instead of adopting a single data architecture approach, modern data architecture should include toolsets and patterns that are purpose-built and optimized for specific use cases. The architecture should be able to evolve and includes basic building blocks, such as scalable data lakes, purpose-built analytics services, unified data access, and unified governance.

Organize data zones

How the data is organized and stored for quick and easy access is a critical aspect of data architecture. This can be achieved by setting up custom data zones within a data lake. The data zones are categorized as follows:

- Raw data that's collected from heterogeneous sources
- Curated and transformed data to support the analytical needs of each domain
- Use case or product-based data marts for reporting needs
- Externally exposed data with security and compliance controls

Plan for agility and democratization of data

The effectiveness of an analytics platform depends on the speed of provisioning data as well as democratizing the provisioned data for consumption. Data provisioning agility is achieved by the ability of the data architecture to procure and process data in a variety of ways—such as real-time, near-real time, batch, micro-batch, or hybrid—based on the use case. Data democratization is achieved by defining data sharing and access control workflows that are monitored by data stewards. Implementing a data marketplace is one of the enablers for democratizing data.

Define secure data delivery

A modern data architecture is a fortress to the outside world in security but allows easy access to employees or data users, as defined by their job functions, and adheres to compliance restrictions such as the Health Insurance Portability and Accountability Act (HIPAA), personally identifiable information (PII), General Data Protection Regulation (GDPR), and so on. This is achieved by role-based access control (RBAC) and tag-based access control (TBAC) methods. On AWS, tags are used to control access to data to simplify access control management. Do this in alignment with the principles that are outlined in the AWS CAF Security perspective.

Plan for cost-effectiveness

Traditional data warehouses provide tightly coupled computing and storage with a high cost of resource utilization. A modern architecture decouples computing and storage, and implements tiered storage based on the data lifecycle. For example, on AWS, you can use Amazon Simple Storage Service (Amazon S3) to control costs and decouple data storage from compute. Amazon S3 storage classes are purpose-built to provide the lowest cost storage for different access patterns. In addition, AWS compute tools (such as Amazon Athena, AWS Glue, Amazon Redshift, and Amazon SageMaker Runtime) are serverless, so you don't have to manage infrastructure, and you pay only for what you use.

Advance

Modern data architecture could be further enhanced to increase the breadth of data usage—from standard analytics that supports business and operational functions to more complex capabilities that support predictions and insights—and helps support faster decision-making. To achieve this, the architecture supports the capabilities described in the following sections.

Understand feature engineering

Feature engineering uses machine learning and involves setting up feature stores or feature marts. Data science teams create new features (derived attributes) for both supervised and unsupervised learning models and store them in feature marts for simplified transformation and enhanced data accuracy. Enterprises can reuse the features across multiple analytics models, which improves speed to market.

Plan to denormalize datasets

Constructing denormalized datasets or data marts could significantly simplify the datasets for business users by making the required data readily available at a single location and increasing the speed of analytics. If designed carefully, one record could support multiple usage models and reduce the overall development lifecycle. Effective governance of denormalized datasets is also significant for two reasons. Implementing denormalized data could create a large number of redundant datasets, which could become a challenge to manage at scale. In addition, these datasets could be increasingly difficult to repurpose if they aren't modeled correctly.

Design portability and scalability

Large organizations seldom have all their applications and users on a single data platform. Their applications and data stores are typically distributed across legacy on-premises and cloud platforms, making it difficult for analytics teams to mix and merge data. We recommend that you containerize data based on characteristics such as domain, geography, business use cases, and so on. This containerization increases portability between various platforms and applications and supports more effective consumption. Segmenting data into containers and exposing them through APIs helps you scale your data architecture more easily. It enables hybrid, end-to-end data flow and helps on-premises and cloud-based applications work seamlessly.

Excel

As a modern analytics architecture evolves within an organization, it is important to manage that change by introducing reusable concepts. These concepts increase durability and adoption while keeping costs in check. Some of the concepts to consider are discussed in the following sections.

Design a configurable framework

Organizations often create multiple, complex models to address their unique business needs. These models require the creation of multiple data pipelines and engineered features. Over time, this creates significant redundancy and increases operating costs. Creating a framework that incorporates a set of parameter-driven, configurable base models reduces the development time and operating costs. The analytical engine can implement these configurable models to provide the desired output.

Plan to build a unified analytical engine

Business problems are unique and often require custom technologies to address requirements, resulting in multiple analytical engines in an organization. Designing and developing a unified AI-based analytical engine interface that can support multiple programming paradigms simplifies usage and reduces costs.

Define DataOps

Most data professionals spend a significant amount of time performing data operations such as locating the right data, transforming, modeling, and so on. Having agile data operations (DataOps) can greatly enhance the data architecture by breaking down the silos of data engineers, data scientists, data owners, and analysts. DataOps enables better communication between teams, reduces cycle time, and ensures high data quality. Data and analytics architectures have undergone numerous transformations over time because of changing business needs and technological advancements. An organization must strive to develop, implement, and maintain a data and analytics architecture that evolves over time and supports its business.

Data engineering

Automate and orchestrate data flows across your organization.

Use metadata to automate pipelines that process raw data and generate optimized outputs. Leverage existing architectural guardrails and security controls as defined across the AWS CAF platform architecture and platform engineering capabilities, as well as the Operations perspective. Work with the platform engineering enablement team to develop reusable blueprints for common patterns that simplify pipeline deployment.

Start

Deploy a data lake

Establish foundational data storage capabilities by using suitable storage solutions for structured and unstructured data. This enables you to collect and store data from various sources, and makes the data accessible for further processing and analysis. Data storage is a critical component of a data engineering strategy. A well-designed data storage architecture allows organizations to store, manage, and access their data efficiently and cost-effectively. AWS offers a variety of data storage services to meet specific business needs.

For example, you can establish foundational data storage capabilities by using Amazon Simple Storage Service (Amazon S3) for object storage, Amazon Relational Database Service (Amazon RDS) for relational databases, and Amazon Redshift for data warehousing. These services help you store data securely and cost-effectively, and make the data easily accessible for further processing and analysis. We recommend that you also implement data storage best practices, such as data partitioning and compression, to improve performance and reduce costs.

Develop data ingestion patterns

To automate and orchestrate data flows, establish data ingestion processes to gather data from diverse sources, including databases, files, and APIs. Your data ingestion processes should support business agility and take governance controls into account.

The orchestrator should be capable of running cloud-based services and provide a automated scheduling mechanism. It should offer options for conditional links and dependencies among tasks, along with polling and error-handling capabilities. Additionally, it should seamlessly integrate with alerting and monitoring systems to ensure that pipelines run smoothly.

A few popular orchestration mechanisms include:

- *Time-based orchestration* starts a workflow on a recursive interval and at a defined frequency.
- *Event-based orchestration* starts a workflow based on the occurrence of an event such as creation of a file or an API request.
- *Polling* implements a mechanism in which a task or workflow calls a service (for example, through an API) and waits for a defined response before proceeding to the next step.

Modern architecture design emphasizes taking advantage of managed services that simplify infrastructure management in the cloud and reduce the burden on developers and infrastructure teams. This approach also applies to data engineering. We recommend that you use managed services where applicable to build data ingestion pipelines to accelerate your data engineering processes. Two examples of these types of services are Amazon Managed Workflows for Apache Airflow (Amazon MWAA)and AWS Step Functions:

- **Apache Airflow** is a popular orchestration tool for programmatically authoring, scheduling, and monitoring workflows. AWS offers Amazon Managed Workflows for Apache Airflow (Amazon MWAA) as a managed service that enables developers to focus on building rather than managing infrastructure for the orchestration tool. Amazon MWAA makes it easy to author workflows by using Python scripts. A directed acyclic graph (DAG) represents a workflow as a collection of tasks in a way that shows each task's relationships and dependencies. You can have as many DAGs as you want, and Apache Airflow will run them according to each task's relationships and dependencies.
- AWS Step Functions helps developers build a low-code visual workflow for automating IT and business processes. The workflows that you build with Step Functions are called *state machines*, and each step of your workflow is called a *state*. You can use Step Functions to create workflows for built-in error handling, parameter passing, recommended security settings, and state management. These reduce the amount of code you have to write and maintain. Tasks perform work by coordinating with another AWS service or an application that you host either on premises or in a cloud environment.

Accelerate data processing

Data processing is a crucial step in making sense of the vast amounts of data collected by modern organizations. To get started with data processing, AWS offers managed services such as AWS Glue or AWS Data Pipeline, which provide powerful extract, transform, and load (ETL) capabilities.

Organizations can use these services to start processing and transforming raw data, including cleaning, normalizing, and aggregating data to prepare it for analysis.

Data processing starts with simple techniques such as aggregation and filtering to perform initial data transformations. As data processing needs evolve, you can implement more advanced ETL processes that enable you to extract data from various sources, transform it to fit your specific needs, and load it into a centralized data warehouse or database for unified analysis. This approach ensures that data is accurate, complete, and available for analysis in a timely manner.

By using AWS managed services for data processing, organizations can benefit from a higher level of automation, scalability, and cost-effectiveness. These services automate many routine data processing tasks, such as schema discovery, data profiling, and data transformation, and free up valuable resources for more strategic activities. Additionally, these services scale automatically to support growing data volumes.

Provide data visualization services

Find ways to make data available to decision-makers who use data visualization to interpret data meaningfully and quickly. Through visualizations you can interpret patterns and increase engagement across a diverse set of stakeholders, regardless of their technical skills. A good platform enables data engineering teams to provision resources that provide data visualization rapidly and with little overhead. You can also provide self-service capabilities by using tools that can easily query data stores without the need for engineering expertise. Consider using built-in tooling that can provide serverless business intelligence through data visuals and interactive dashboards, and that can use natural language to query back-end data.

Advance

Implement near real-time data processing

Data processing is an essential component of any data engineering pipeline, which enables organizations to transform raw data into meaningful insights. In addition to traditional batch processing, real-time data processing has become increasingly important in today's fast-paced business environment. Real-time data processing enables organizations to respond to events as they occur, and improves decision-making and operational efficiency.

Validate data quality

Data quality directly impacts the accuracy and reliability of insights and decisions that are derived from data. Implementing data validation and cleansing processes is essential to ensure that you use high-quality and trustworthy data for analysis.

Data validation involves verifying the accuracy, completeness, and consistency of the data by checking it against predefined rules and criteria. This helps identify any discrepancies or errors in the data, and ensures that it is fit for purpose. Data cleansing involves the identification and correction of any inaccuracies, inconsistencies, or duplications in the data.

By implementing data quality processes and tools, organizations can improve the accuracy and reliability of insights derived from the data, resulting in better decision-making and operational efficiency. This not only enhances the organization's performance but also increases stakeholder confidence and trust in the data and analysis produced.

Prove data transformation services

Data transformation prepares data for advanced analytics and machine learning models. It involves using techniques such as data normalization, enrichment, and deduplication to ensure that the data is clean, consistent, and ready for analysis.

- *Data normalization* involves organizing data into a standard format, eliminating redundancies, and ensuring that data is consistent across different sources. This makes it easier to analyze and compare data from multiple sources and enables organizations to gain a more comprehensive understanding of their operations.
- *Data enrichment* involves enhancing existing data with additional information from external sources such as demographic data or market trends. This provides valuable insights into customer behavior or industry trends that might not be apparent from internal data sources alone.
- *Deduplication* involves identifying and removing duplicate data entries, and ensuring that the data is accurate and free from errors. This is especially important when dealing with large datasets, where even a small percentage of duplication might skew the results of the analysis.

By using advanced data transformation techniques, organizations ensure that their data is of high quality, accurate, and ready for more complex analysis. This leads to better decision-making, increased operational efficiency, and a competitive advantage in the marketplace.

Enable data democratization

Promote a culture of data democratization by making data accessible, understandable, and usable for all employees. Data democratization helps employees make data-driven decisions and contributes to the organization's data-driven culture. This means breaking down silos and creating a culture where data is shared and used by all employees to drive decision-making.

Overall, data democratization is about creating a culture where data is valued, accessible, and understandable by everyone in the organization. By enabling data democratization, organizations foster a data-driven culture that drives innovation, improves decision-making, and ultimately leads to business success.

Excel

Provide UI-based orchestration

To build organizations that are agile and use effective approaches, it is important to plan for a modern orchestration platform that is used by development and operations resources across lines of business. The goal is to develop, deploy, and share data pipelines and workflows without being dependent on a single team, technology, or support model. This is achieved through capabilities such as UI-based orchestrating. Features such as drag-and-drop interaction enables users who have little technical expertise to construct DAGs and state machine data flows. These components can then generate executable code that orchestrates data pipelines.

DataOps helps overcome the complexities of data management and ensures a seamless data flow across organizations. A metadata-driven approach ensures data quality and compliance in accordance with your organization's mandates. Investment in toolsets such as microservices, containerization, and serverless functions improves scalability and agility.

Relying on data engineering teams to generate value out of data and leaving day-to-day infrastructure tasks to automation enables organizations to achieve excellence in automation and orchestration. Near real-time monitoring and logging of data flow management tasks support immediate remediation actions and improve the performance and security of the data flow pipeline. These principles help achieve scalability and performance while ensuring a secure data-sharing model, and set organizations up for success in the future.

Integrate DataOps

DataOps is a modern approach to data engineering that emphasizes the integration of development and operations processes to streamline data pipeline creation, testing, and deployment. To implement DataOps best practices, organizations use infrastructure as code (IaC) and continuous integration and continuous delivery (CI/CD) tools. These tools support automated pipeline creation, testing, and deployment, which significantly improve efficiency and reduce errors. DataOps teams work with platform engineering enablement teams to build these automations, so each team can focus on what they do best.

Implementing DataOps methodologies helps foster a collaborative environment for data engineers, data scientists, and business users, and enables rapid development, deployment, and monitoring of data pipelines and analytics solutions. This approach provides more seamless communication and collaboration across teams, which leads to faster innovation and better outcomes.

To take full advantage of the benefits of DataOps, it is important to streamline data engineering processes. This is achieved by using best practices from platform engineering teams, including code review, continuous integration, and automated testing. By implementing these practices, organizations ensure that data pipelines are reliable, scalable, and secure, and that they meet the needs of both business and technical stakeholders.

Provisioning and orchestration

Create, manage, and distribute catalogs of approved cloud products to users.

Provisioning infrastructure in a consistent, scalable, and repeatable manner becomes more challenging as your organization grows. Streamlined provisioning and orchestration help you achieve consistent governance and meet your compliance requirements while enabling users to deploy only approved cloud products.

Reusing pre-approved products in your organization enables your developers to build applications faster and more consistently while meeting the security and governance requirements of your organization.

Start

Deploy a hub-and-spoke catalog model

Software assets that are managed in a service catalog as portfolios are shared with users in one or more accounts in a hub-and-spoke pattern. You can use a private marketplace and private offers to curate an assortment of third-party solutions and distribute them with your infrastructure as code (IaC) templates.

To enable your builders to consume pre-approved products, define a process to review, approve, and publish these products to your users. Start by designing and implementing a centrally managed repository that contains these pre-approved products. Design a system that grants access to the licenses and products in this repository when the users in your organization need to consume each product.

Allow the builders in your organization to submit products for approval to the publishing mechanism, so these products are made available for all users in your organization after they are approved.

Curate templates for reuse

When you have codified the IaC templates for your solutions and defined your hub-and-spoke model, you should define two categories of templates for each spoke account: *provisioned/enforced* and *available to consume*. *Provisioned/enforced* templates are provisioned directly from the management account into each member account as foundational capabilities. *Available to consume* templates are available for builders to browse and provision in a self-service manner.

Apply default parameters for reuse

Implement IaC templates that include default parameters that your builders can preselect. This enables builders to align to governance without having to evaluate the details of each parameter, and prevents them from making incorrect choices. This approach exposes only what is needed for setup. For example, AWS Service Catalog implements this approach with a constraint capability that controls the rules that are applied to a product in a specific portfolio. This customization is preconfigured when the builder team uses self-service provisioning of templates.

Establish an approval process

Users should be able to submit requests to access a product they are not approved for if they have a business justification to use the product. Build a notification system that informs users when updates for the products they are using are available, so they can comply with the latest security updates.

Establish a workflow for builders to submit new products for review through the self-service portal. Builders can use the portal to define the audience for the product and to identify the user groups that should have access to the product. For each submission, use your defined processes to review, approve, and publish the product to the self-service portal.

Advance

Create a self-service portal

Create a self-service portal to distribute, browse, and consume approved cloud products. The users in the organization can use this portal to search for the products they need to build their infrastructure and to deploy applications to their environment. Establish permissions boundaries for users who have access to the products in the portal, and set limits on the number of times a user can consume licensed products. Define a base set of resources that can be directly provisioned or made available as a self-service model in each of your spoke accounts, as the accounts are created by using solutions such as Customizations for AWS Control Tower.

Enable a private marketplace

A private marketplace provides a curated catalog of purchased products (software, data, and professional services) and is implemented in a hub-and-spoke pattern (with one management account and multiple member accounts) so that spoke accounts can subscribe to only the approved software. This product governance helps control software costs and streamlines legal and contractual reviews. Create a private marketplace at the management account level to serve as the primary hub.

Manage entitlements

Enable controls that allow only authorized users and workloads to consume a license within vendor-defined limits. This helps reduce the risk of costly audits and unexpected licensing tune-ups.

Excel

Integrate with procurement systems

Complement your existing procurement processes by integrating them into AWS Marketplace. This is done by extending your procurement systems (Coupa or SAP Ariba) to a private marketplace so your users can follow existing procurement and approval processes to obtain software. Create the appropriate IAM-managed permissions, use AWS Marketplace to generate the necessary information to configure your procurement solution, and configure your procurement solution to complete the integration. For example, you can set up a punchout, attach purchase orders to your AWS invoices, and then align your procurement processes to use the standard provisioning solutions.

Enable your builders to access the pre-approved products through an internal API, so users can incorporate the products into their applications or build their own personalized portals for their teams to consume the products. Integrate the submission and publication process for creating new products, and allow users to request new licenses and access to products through APIs.

Integrate with your ITSM tools

If applicable, connect with IT service management (ITSM) tools and automate any updates to your configuration management database (CMDB). Establish processes and mechanisms to evaluate the products that your organization uses. Establish a mechanism to inform users of pre-approved products that they need to update for compliance. Use your ITSM tools to analyze your environment and to push security and compliance updates to products across your organization when critical updates are needed.

Implement a lifecycle management and version distribution system

Maintain versions of IaC templates, and versions of services provisioned from the templates, throughout their development lifecycle. You can use the hub-and-spoke model you implemented for your catalog to define whether a forced update is required at a spoke level (for example, if concurrent versions are available for self-service provisioning), and which versions need to be marked for obsolescence. Using a hub-and-spoke catalog also helps manage the audit and distribution of new versions as required.

Modern application development

Build well-architected, cloud-native applications.

Modern application development practices are essential for organizations to build well-architected, cloud-native applications and remain competitive. Businesses can use cloud-native technologies such as containers and serverless computing to create scalable and agile applications that adapt to changing market demands. These technologies enable organizations to optimize resource utilization, reduce costs, and improve the performance of their applications.

When you design your modern applications, develop agile solutions for operations and development. A modern application automatically reacts to changes in customer demand and is resilient to failure. Engineers can develop and deploy changes quickly and monitor application performance. A modern application is designed to be self-healing and capable of scaling to both large or small levels of traffic, including no traffic at zero cost, when needed.

Building well-architected, cloud-native applications requires a deep understanding of the underlying technologies and their best practices. Organizations should adopt a microservices architecture and design their applications to be modular and loosely coupled, allowing for independent deployment and scalability. This approach enables organizations to break down their applications into smaller, more manageable components that are developed, tested, and deployed quickly and independently.

Start

Explore modern approaches

Start by investigating containers, serverless technologies, and other approaches that enable the development of microservices, which enhance resource efficiency, help improve security, and minimize infrastructure expenses. Choose to modernize your existing differentiating and enterprise applications to improve efficiency and maximize the value of your existing investments. Contemplate replatforming (transitioning your self-managed containers, databases, or message brokers to managed cloud services) and refactoring (redeveloping your applications to adopt cloud-native architectures) based on value-led decision-making.

When you update your existing cloud-based application, a successful approach involves using the strangler fig pattern to progressively decompose your architecture into microservices. This procedure aids in adopting a contemporary application methodology, so you can realize the inherent benefits and demonstrate its value to the larger organization. Consider constructing your applications as distinct microservices that harness event-driven architectures where applicable. Ensure that your architecture takes into account unchangeable service quotas and physical resources to avoid affecting workload performance or reliability.

Adopt cloud-native compute capabilities

Cloud-native computing capabilities are key for modern application development. This approach requires organizations to consider how they want their compute units to be hosted and identify the best option for each use case or service. For example, AWS Lambda offers a serverless mechanism for running your application code and plays a key role in event-driven architectures. Lambda functions are launched on demand and run in parallel up to a defined maximum concurrency, so they can scale to perform a variety of tasks.

Use containerization

In modern software development, managing applications and their dependencies has become an increasingly complex task, especially when you consider the need to maintain consistency across various environments. To address these challenges, containerization technologies such as Docker have emerged as an effective solution for packaging applications and their dependencies. Containers ensure consistent and reproducible deployments regardless of your application's runtime environment, so development in your local environment behaves in the same manner as production development in the cloud environment. This approach reduces errors that might be caused by mismatches within the environment or its configurations.

Use modern databases

When you use modern databases, each microservice within your application can use the right purpose-built database that meets its requirements, which increases agility and performance while lowering costs. For example, one microservice might use a NoSQL database to achieve high throughput when storing session data, another microservice might use a relational database to do complex table joins, and yet another microservice might use a quantum ledger database to track changes to the blockchain.

Modern databases offer scalability and flexibility. They also help provide better security, compliance, and reliability than traditional databases. They enable organizations to store and manage their data more efficiently and ensure that applications can access the right data at the right time, leading to better performance and user experience.

Migrating to modern databases is a critical component of modern application development. By using the right data storage solutions, organizations can optimize their data management capabilities and deliver more efficient and reliable applications. By making each microservice independent and choosing the right technologies for each microservice, organizations can further optimize their data capabilities to achieve maximum efficiency and scalability while minimizing costs.

Advance

Optimize your modern architecture

To achieve further optimizations, refine your implementation of serverless technologies and develop architectures that can be independently scaled and deployed by using AWS services such as Amazon API Gateway and AWS Lambda. Implement service discovery by using Amazon Route 53 and AWS Cloud Map to ensure seamless communication between components.

Adopt API versioning, caching, and rate limiting to maintain compatibility and performance across different application versions. Enhance security with AWS Identity and Access Management (IAM) and resource policies. These help ensure that your infrastructure is protected and access is granted only to authorized entities.

If possible, use serverless services to run containers without having to manage the underlying infrastructure. This enables you to focus on developing your core applications and allows for better resource management and performance. It also helps you take full advantage of the benefits of scalability, flexibility, and cost efficiency.

By diving deeper into the intricacies of serverless architectures and incorporating these advanced practices, organizations can uncover opportunities for improvement and fine-tuning, and ultimately maximize the potential of their cloud-native applications. This pursuit facilitates the adoption of more sophisticated application patterns that further elevate the overall user experience. It also empowers organizations to become more agile and efficient in their software development processes.

Use service mesh technologies

As organizations increasingly adopt a microservices architecture for building and deploying applications, managing the complexity, security, and communications among these services becomes critical. Service mesh technologies such as Istio, Linkerd, or Consul play a pivotal role in helping enhance the security, observability, and reliability of microservices.

Ensure visibility and traceability

Modern practices provide greater visibility and traceability in the development process, and make it easier to comply with industry standards and best practices. Visibility and monitoring are essential for modern application development. Implementing monitoring and logging solutions to provide valuable insights into application performance enables organizations to identify areas for improvement and optimize their applications. We recommend that you work with your platform engineering teams to ensure that tools are available to provide end-to-end visibility and monitoring of application errors, performance, and compliance, so you can detect, diagnose, and resolve issues quickly.

Excel

Embrace microservices

For many organizations, modern application development is synonymous with business success. Microservices are at the heart of this transformation, and organizations can benefit from embracing these powerful architectural patterns.

Microservices offer a highly scalable, resilient, and agile application architecture. By breaking an application down into small, independently deployable services, organizations can choose to rapidly iterate on specific components without impacting other parts of the application. Advanced resiliency patterns, such as *circuit breakers* and *bulkheads*, play a crucial role in ensuring the high availability of these applications.

Circuit breakers act as a safety mechanism that prevents cascading failures by temporarily halting or shifting communications from an unhealthy service, so it can recover. Bulkheads isolate resources and limit the scope of impact of potential failures. Together, these patterns create a robust architecture that withstands unforeseen disruptions and maintains optimal performance.

Another critical aspect of implementing microservices is the adoption of domain-driven design (DDD) principles. DDD focuses on creating a shared understanding of the business domain and translating it into a well-structured software design. This approach leads to more cohesive and maintainable microservices, and ensures that the application evolves in step with the organization's needs.

Optimizing inter-service communication is also vital in a microservices-based application. By implementing advanced protocols such as gRPC or GraphQL, organizations can significantly enhance communication efficiency between services. These protocols offer capabilities such as type safety, low latency, and flexibility, which help improve the overall performance and maintainability of the application.

An organization that adopts microservices provides an environment that fosters innovation, agility, and collaboration. The development teams are typically organized around business capabilities and have a strong focus on continuous integration and continuous delivery (CI/CD) practices. They are empowered to make decisions, experiment, and iterate quickly, and they embrace a culture of shared responsibility and accountability.

Continuous integration and continuous delivery

Evolve and improve applications and services faster than organizations that use traditional software development and infrastructure management processes.

Adopting DevOps practices with continuous integration and continuous delivery (CI/CD) promotes a streamlined, automated, and efficient process for building, testing, and deploying applications. CI/CD enables rapid delivery of software, reduces the risk of deployment errors, and ensures that applications are always up to date with the latest features and bug fixes. The main objective is to evolve and improve applications and services at a faster pace by evolving from the use of traditional software development and infrastructure management processes.

Start

Adopt software component management

Software component management is the practice of managing all the individual components used to build software, including libraries, frameworks, source code repositories, modules, artifacts, and third-party dependencies. We recommend that you use a version control system such as Git or Apache Subversion to manage source code, enable collaboration, and maintain a history of code changes. You can monitor changes and events in the repository to automate the process, create pipelines, manage your code (for example, by using AWS CodeCommit), and integrate your workflows with additional services as needed.

Create CI/CD pipelines

CI/CD pipelines are sets of automated instructions that are initiated by changes committed to the version control system. They typically include instructions for building the application, running automated tests, and deploying code to a specific environment. You can set up an automated CI/CD pipeline by using tools such as AWS CodePipeline, Jenkins, GitLab, or CircleCI. You can also set them up directly in version control systems that support pipeline generation.

Start with a minimum viable pipeline for continuous integration, and then transition to a continuous delivery pipeline that includes more actions and stages. Treat your continuous delivery configuration as code. You can use multiple, distinct pipelines for each branch and team, so think about which configuration variables you need to set up and how to best support the teams that will use the pipelines.

Consider deployment windows—which days and times you want to deploy your code. Consider your system's low-demand hours, so if you have to roll back, it will have the least impact on your customers. Other best practices include avoiding deployments on Fridays and implementing a code freeze during high peak dates or before holidays. Consider defining rules about deploying code when the author of the commit isn't available (for example, on vacation). Keep in mind that deployments fail and you might need to rely on external help. Evaluate different deployment methods such as in-place, rolling, immutable, and blue/green deployments. Consider using fully managed services for continuous delivery workflows in order to increase availability and security while minimizing complexity and management.

Deploy automated testing

Modern practices recommend *shifting left* (moving testing closer to the developer and to the IDE, and earlier in the lifecycle) to detect and remediate problems before they are committed to a repository and initiate a pipeline. This practice involves quick feedback loops with the developer, because errors are detected while the developer is coding. Shifting left is associated with lower costs, because testing doesn't require running pipelines, which can result in asynchronous feedback and higher operational expenses.

Automated testing catches errors early in the development process, and includes unit tests, integration tests, and functional tests. We recommend that you encourage developers to use tools to create unit tests as early as possible and to run them before pushing the code to the central repository. Additionally, make sure that your automated processes include static code analysis, performance benchmarking, and security application testing.

Create documentation

In addition to implementing a CI/CD pipeline to streamline development workflows, you should maintain clear and comprehensive documentation to ensure the pipeline's ongoing effectiveness, maintainability, and scalability. Documentation is a vital aspect of CI/CD pipelines, because it provides development teams with a clear understanding of the pipeline's design, components, and processes. When you create documentation, start with a pipeline overview, explain the architecture and design tradeoffs, describe the tools and technologies that are being used, specify the initial configuration and settings, outline the security measures and access control, and include troubleshooting and maintenance information.

Use infrastructure as code

Use tools such as Terraform, Ansible, or AWS CloudFormation to manage infrastructure and to ensure consistent and reproducible environments. Treat your infrastructure as code, make sure you track changes in the infrastructure, and avoid making changes directly in the console. Define all infrastructure—including database provisioning—as code and deploy these changes by using pipelines. Consider running database integration as code in pipelines with a small subset of sanitized production data. When possible, make the changes and track those changes in the code.

As with software code, follow these best practices for your infrastructure code:

- Use version control.
- Make use of bug tracking and ticketing systems.
- Have peers review changes before applying them.
- Establish infrastructure code patterns and designs.
- Test infrastructure changes.

Keep and track standard metrics

To maintain a high level of performance, develop and track against key metrics to understand the health and business impact of your pipelines, including:

- **Build frequency.** The number of builds offers insights into your team's productivity and the complexity of changes.
- **Deployment frequency.** Regular deployments indicate a healthy, agile development process.
- **Lead time for changes.** Measuring the average time for changes to reach production can help you identify bottlenecks in your deployment process.
- **Mean time through pipeline.** The average time from the initial pipeline stage to each subsequent stage can help optimize your workflow.
- **Production change volume.** Keeping track of the number of changes reaching production can provide insights into the stability of your production environment.
- **Build time.** The average build time can indicate potential issues in the codebase or infrastructure.

Advance

Use configuration management

Configuration management tools play a critical role in automating the deployment, configuration, and management of software and infrastructure. They provide a systematic approach to handling changes and maintaining the desired state of infrastructure, software, and configurations across various environments. These tools enable developers to define the desired state of a system by using declarative or imperative languages. The configuration management tool then automates the process of applying these configurations to the target systems, ensuring consistency and repeatability.

Use configuration management tools to automate the deployment, configuration, and management of software and infrastructure. AWS Systems Manager State Manager is a secure and scalable configuration management service that automates the process of keeping your managed nodes and other AWS resources in a state that you define.

Integrate monitoring and logging

Integrating monitoring and logging solutions into CD pipelines offers numerous benefits for development teams and for the overall software development process. These solutions can provide real-time insights into application performance, enable faster identification and resolution of issues, and promote continuous improvement to help ensure that applications remain reliable, performant, and scalable throughout their lifecycle. Investing in monitoring and logging solutions is a key aspect of maintaining a robust and efficient CD pipeline, and ultimately contributes to the successful delivery of high-quality software.

Create a cadence for merging

Commit or merge code changes to the mainline (trunk or main) branch at least once every day or, ideally, multiple times a day after each task. This cadence leads to multiple daily pipeline invocations. A pull-based branching workflow model aligns with this approach. Use feature flags, dark launching, and similar techniques to customize the features your customers use.

Capture post-deployment behavior

After a deployment, capture production behavior by using automated synthetic tests and synchronize results with the continuous delivery pipeline to ensure that corrective actions occur promptly. The top priority for developers should be to fix errors discovered in pipelines as soon as possible, commit code changes to the source code repository, and verify error resolution in the pipeline.

Best post-deployment practices include observing the most important key performance indicators (KPIs) and validating that there are no errors in the production environment. Automate error handling and evaluating post-deployment KPIs to quantify the impact of your release. Automatically generate speed, security, and stability metrics that developers can use to make improvements. For more information, see the solution DevOps Monitoring Dashboard on AWS.

Excel

Adopt cutting-edge practices and technologies for optimal performance. Continually refining your CI/CD processes helps you improve software quality, reduce time to market, and increase agility. New techniques and tools continuously emerge, which makes it essential for your organization to stay informed and adapt to maintain a competitive edge.

To remain adaptive, consider the following:

- Define everything as code, including your application, configuration, infrastructure, data, AWS accounts and organizations, deployment pipelines, networking, and security and compliance controls.
- Create corresponding deployment pipelines for compute images, shared services, and applications.
- Consider a GitOps model in which pull-based requests initiate a workflow to deploy changes by comparing the existing infrastructure state to the desired state, as described in the code.
- Consider using CD pipelines to deploy machine learning (ML), data, Internet of Things (IoT), and other workloads.
- Digitally sign all build artifacts and store them in a secure repository.
- Track software provenance by automatically generating a software bill of materials that creates a record of all versioned and digitally signed artifacts that are deployed to customers.
- After you eliminate all manual activity in a software delivery process, remove manual review boards.

For applications and services that have automated their entire software delivery process, consider continuous deployment in which teams deploy changes that pass all checks in a pipeline to customers in production. For a visualization, see the first diagram in [What is Continuous Delivery?](#) on the AWS website.

Integrate AI/ML technologies

The integration of artificial intelligence (AI) and machine learning (ML) technologies into CI/CD pipelines offers several benefits, including the following:

- Automated test generation
- Intelligent test prioritization
- Predictive analytics for issue detection
- Anomaly detection and root cause analysis
- Code review and quality assurance
- Deployment optimization

For more information, see [Add intelligence to your developer operations](#) on the AWS website.

Adopt chaos engineering practices

Chaos engineering involves intentionally injecting failures into systems to test their ability to withstand and recover from unexpected events. By identifying weaknesses and addressing them proactively, organizations can improve their overall system reliability and minimize the impact of potential issues.

Adopt chaos engineering practices to test the resilience of your systems by using tools such as Gremlin, Chaos Monkey, or Litmus. Run controlled experiments regularly to identify vulnerabilities, validate fault tolerance, and ensure that your application handles unexpected failures gracefully. This proactive approach helps improve system reliability and contributes to a more robust CI/CD pipeline.

Optimize performance

Optimize your application's performance continuously by using profiling tools, real-time monitoring, and feedback loops. Apply techniques such as the following to ensure that your applications can handle increased traffic and demand:

- Code optimization

- Profiling
- Real-time monitoring
- Feedback loops
- Caching
- Load balancing
- Scalability and performance testing

Implement advanced observability

Elevating your cloud infrastructure's observability goes beyond the basics of collecting, aggregating, and analyzing metrics, logs, and traces. When observability is enhanced with tools such as Amazon CloudWatch and AWS X-Ray, it evolves into a strategic practice that fuels continuous delivery and innovation.

In a robust CI/CD pipeline, advanced observability enables you to uncover insights, not just about your applications and infrastructure but also about your entire system's performance and health, including the pipeline itself. These insights help you:

- Rapidly identify, understand, and address potential issues to improve application stability and reduce downtime
- Streamline your CI/CD processes to create faster and more reliable deliveries
- Gain deeper insights into the impact of code changes and deployments to drive informed decision-making
- Optimize resource utilization to improve operational efficiency and cost-effectiveness

To elevate observability:

- Embed observability into every layer of your applications and infrastructure to create a comprehensive view of your systems' performance, behavior, and health.
- Centralize data collection, storage, and analysis with tools such as Amazon CloudWatch to unify your observability data for easy access and interpretation.
- Use AWS X-Ray for distributed tracing to understand how your applications and their underlying services are performing.
- Establish feedback loops for continuous improvement, and use your observability data to drive iterative enhancements to your systems.

Adopting advanced observability isn't just about maintaining your systems—it's a strategic move toward achieving operational excellence and driving continuous innovation in your organization.

Implement GitOps practices

Implement GitOps practices to manage infrastructure and application configurations by using a Git repository as a single source of truth. This approach simplifies change management, enhances traceability, and ensures consistency across environments.

Conclusion

This guide serves as a playbook for the successful implementation and management of a foundation for successful cloud adoption. It discusses how to:

- Directly address technical challenges and intricacies in platform architecture to establish robust guidelines and principles for your cloud environment and the data residing within it.
- Build out platform engineering with strong provisioning and orchestration.
- Enable the use of a compliant, multi-account cloud environment that manages and distributes approved cloud products to users in a scalable and repeatable manner.
- Support data architecture decisions with the tooling required for data engineering to drive data-driven decision-making.
- Pair these capabilities with modern application development strategies and CI/CD processes to promote agility, efficiency, and innovation within your organization.
- Build cross-functional relationships and take inputs from other AWS CAF perspectives in your own decision-making to ensure the success of your platform and the teams behind it.

Further reading

AWS Cloud Adoption Framework (AWS CAF) resources:

- eBook
- Audiobook
- Infographic
- AWS CAF for Artificial Intelligence, Machine Learning, and Generative AI
- Business perspective
- People perspective
- Governance perspective
- Operations perspective
- Security perspective

Additional resources:

- AWS Architecture Center
- AWS case studies
- AWS General Reference
- AWS glossary
- AWS Knowledge Center
- AWS Prescriptive Guidance
- AWS Partner Solutions (formerly Quick Starts)
- AWS security documentation
- AWS Solutions Library
- AWS Training and Certification
- AWS Well-Architected
- AWS whitepapers and guides
- Getting Started with AWS
- Overview of Amazon Web Services

Contributors

Contributors to this guide include:

- Tony Santiago, Senior Partner Solutions Architect, AWS
- Matias Undurraga, Enterprise Technologist, AWS
- Alex Torres, Senior Solutions Architect, AWS
- Michael Rhyndress, Senior DevSecOps Consultant, AWS
- Alex Livingstone, Principal Solutions Architect and CloudOps Specialist, AWS
- Bruce Cooper, Principal SDE, AWS
- Ravinder Thota, Senior Advisory Consultant, AWS
- Sausan Yazji, Senior Practice Manager, AWS
- Paul Duvall, Director, DevSecOps, AWS
- Jeremy Tennant, Principal Cloud Delivery Manager, AWS
- Sneh Shah, Principal Infrastructure Lead, AWS
- Sasa Baskarada, Worldwide Lead, AWS Cloud Adoption Framework, AWS

Document history

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

Change	Description	Date
Initial publication	—	October 25, 2023

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 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 AWS CodeCommit. 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, AWS Panorama offers devices that add CV to on-premises camera networks, and Amazon SageMaker 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.

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.

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.

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.

G

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.

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.

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 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.

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

An approach in system engineering that takes privacy into account throughout the whole engineering 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.

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).

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.

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 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.

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.

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.