AWS multi-Region fundamentals

AWS Prescriptive Guidance

AWS Prescriptive Guidance: AWS multi-Region fundamentals

Table of Contents

AWS multi-Region fundamentals

John Formento, Amazon Web Services (AWS)

December 2024 (document history)

This advanced, 300-level guide is intended for cloud architects and senior leaders who build workloads on AWS and are interested in using a multi-Region architecture to improve resilience for their workloads. This guide assumes baseline knowledge of AWS infrastructure and services. It outlines common multi-Region use cases, shares fundamental multi-Region concepts and implications around design, development, and deployment, and provides prescriptive guidance to help you better determine whether a multi-Region architecture is right for your workloads.

Are you Well-Architected?

The AWS Well-Architected Framework helps you understand the pros and cons of the decisions you make when you build systems in the cloud. The six pillars of the Framework provide architectural best practices for designing and operating reliable, secure, efficient, cost-effective, and sustainable systems. You can use the AWS Well-Architected Tool, which is available at no charge on the AWS Management Console, to review your workloads against these best practices by answering a set of questions for each pillar.

For additional expert guidance and best practices for your cloud architecture, including reference architecture deployments, diagrams, and technical guides, see the AWS Architecture Center.

Introduction

Each AWS Region consists of multiple independent and physically separate Availability Zones within a geographic area. Strict logical separation between the software services in each Region is maintained. This purposeful design ensures that an infrastructure or service failure in one Region doesn't result in a correlated failure in another Region.

Most AWS users can achieve their resilience objectives for a workload in a single Region by using multiple Availability Zones or Regional AWS services. However, a subset of users pursue multi-Region architectures for three reasons:

- They have high availability and continuity of operations requirements for their highest tier workloads and want to establish a bounded recovery time from impairments that impact resources in a single Region.
- They need to satisfy data sovereignty requirements (such as adherence to local laws, regulations, and compliance) that require workloads to operate within a certain jurisdiction.
- They need to improve performance and customer experience for the workload by running the workloads in locations that are closest to their end users.

This guide focuses on high availability and continuity of operations requirements, and helps you navigate the considerations for adopting a multi-Region architecture for a workload. It describes fundamental concepts that apply to design, development, and deployment of a multi-Region workload, and provides a prescriptive framework to help you determine whether a multi-Region architecture is the right choice for a particular workload. You need to ensure that a multi-Region architecture is the right choice for your workload because these architectures are challenging, and if the multi-Region architecture isn't built correctly, it's possible for the overall availability of the workload to decrease.

Engineering and operating for resillience in a single Region

Before you dive into multi-Region concepts, start by confirming that your workload is already as resilient as possible in a single Region. To achieve this, evaluate your workload against the reliability pillar and operational excellence pillar of the AWS Well-Architected Framework, and make any necessary changes based on trade-offs and risk assessment. The following concepts are covered in the AWS Well-Architected Framework:

- Workload segmentation based on domain boundaries
- Well-defined service contracts
- Dependency management and coupling
- Handling failures, retries, and back-off strategies
- Idempotent operations and stateful versus stateless transactions
- Operational readiness and change management
- Understanding workload health
- Responding to events

To take single-Region resilience further, review and apply the concepts that are discussed in the paper Advanced Multi-AZ Resilience Patterns: Detecting and Mitigating Gray Failures. This paper provides best practices for using replicas in each Availability Zone to contain failures and expands on multi-AZ concepts that are introduced in the AWS Well Architected Framework. Although a multi-Region architecture can mitigate failure modes that are bound to Availability Zones, there are trade-offs that come with a multi-Region approach that you should consider. That is why we recommend that you start with a multi-AZ approach, and then evaluate a specific workload against fundamentals for multi-Region architectures to determine if a multi-Region approach can increase the workload's resilience.

Multi-Region fundamental 1: Understanding the requirements

As mentioned previously, high availability and continuity of operations are common reasons for pursuing multi-Region architectures. Availability metrics measure the percentage of time a workload is available for use over a defined period, whereas continuity of operations metrics measure recovery time for large-scale, and typically longer, duration events.

Measuring availability is a nearly continuous process. Specific measurements can vary but typically coalesce around a target availability metric, most often referred to as *nines* (such as 99.99 percent availability). With availability goals, one size does not fit all. You should establish availability goals at a workload level and separate non-critical components from critical components, instead of applying a single goal across all workloads.

For continuity of operations, the following point-in-time measurements are typically used:

- **Recovery time objective (RTO)** – RTO is the maximum acceptable delay between the interruption of service and restoration of service. This value determines an acceptable duration for which the service is impaired.
- **Recovery point objective (RPO)** – RPO is the maximum acceptable amount of time since the last data recovery point. This determines what is considered an acceptable data loss between the latest recovery point and a service interruption.

Similar to setting availability goals, RTO and RPO should also be defined at a workload level. More aggressive continuity of operations or high availability requires increased investment. That said, not every application can demand or requires the same level of resilience. Aligning business and IT owners to assess the criticality of applications based on business impact and then tiering them accordingly can help provide a starting point. The following tables provide examples of tiering.

This table shows an example of resilience tiering for service-level agreements (SLAs).

Resilience tier	Availability SLA	Acceptable downtime/year
Platinum	99.99%	52.60 minutes
Gold	99.90%	8.77 hours

Resilience tier	Availability SLA	Acceptable downtime/year
Silver	99.5%	1.83 days

The following table shows an example of resilience tiering for RTO and RPO.

Resilience tier	Maximum RTO	Maximum RPO	Criteria	Cost
Platinum	15 minutes	5 minutes	Mission-critical workloads	$$$
Gold	15 minutes – 6 hours	2 hours	Important but not mission-c ritical workloads	$$
Silver	6 hours – a few days	24 hours	Non-critical workloads	$

When you design workloads for resilience, consider the relationship between high availability and continuity of operations. For example, if a workload requires 99.99 percent availability, no more than 53 minutes of downtime per year is tolerable. It can take at least 5 minutes to detect a failure and another 10 minutes for an operator to engage, make decisions on recovery steps, and perform these steps. It's not unusual to take 30 to 45 minutes to recover from a single issue. In this case, it's beneficial to have a multi-Region strategy to provide an isolated instance that removes correlated impact. This allows for continued operations by failing over within a bounded time while you triage the initial impairment independently. This is where defining the appropriate bounded recovery time and ensuring there is alignment are required.

A multi-Region approach might be appropriate for mission-critical workloads that have extreme availability needs (for example, 99.99 percent or higher availability) or stringent continuity of operations requirements that can be met only by failing over into another Region. However, these requirements are typically applicable only to a small subset of an enterprise's workload portfolio that has a bounded recovery time measured in minutes or hours. Unless an application needs a recovery time of minutes or a few hours, it might be a better approach to wait for a Regional disruption to the application to be remediated within the affected Region. This approach is typically aligned with lower-tier workloads.

Before implementing a multi-Region architecture, business decision-makers and technical teams should be aligned on cost implications, including operational and infrastructure cost drivers. A typical multi-Region architecture can incur a cost that's twice as large as a single-Region approach. Although there are several multi-Region patterns for business continuity, such as running with a hot standby, warm standby, or pilot light, the pattern with the lowest risk of meeting recovery objectives will involve running hot standby, and will double the cost for your workload.

Key guidance

- Availability and continuity of operations goals such as RTO and RPO should be established per workload and aligned with business and IT stakeholders.
- Most availability and continuity of operations goals can be met within a single Region. For goals that cannot be met within a single Region, consider multi-Region with a clear view on trade-offs between cost, complexity, and benefits.

Multi-Region fundamental 2: Understanding the data

Managing data is a non-trivial problem when you adopt multi-Region architectures. The geographical distance between Regions imposes an unavoidable latency that manifests as the time it takes to replicate data across Regions. Trade-offs between availability, data consistency, and introducing higher latency into a workload that uses a multi-Region architecture will be necessary. Whether you use asynchronous or synchronous replication, you will need to modify your application to handle the behavioral changes the replication technology imposes. Challenges around data consistency and latency make it very difficult to take an existing application that was designed for a single Region and make it multi-Region. Understanding the data consistency requirements and data access patterns for particular workloads is critical to weighing the trade-offs.

2.a: Understanding data consistency requirements

The CAP theorem provides a reference for reasoning about the trade-offs among data consistency, availability, and network partitions. Only two of these requirements can be satisfied at the same time for a workload. By definition, a multi-Region architecture includes network partitions between Regions, so you have to choose between availability and consistency.

If you select availability of data across Regions, you won't incur significant latency during transactional write operations, because the reliance on asynchronous replication of committed data between Regions results in reduced consistency across Regions until the replication completes. With asynchronous replication, when there is a failure in the primary Region, there is a high probability that write operations will be pending replication from the primary Region. This leads to a scenario where the latest data is unavailable until replication resumes, and a reconciliation process is needed to handle in-flight transactions that didn't replicate from the Region that experienced the disruption. This scenario requires understanding your business logic and creating a specific process to replay the transaction or compare data stores between Regions.

For workloads where asynchronous replication is favored, you can use services such as Amazon Aurora and Amazon DynamoDB for asynchronous cross-Region replication. Both Amazon Aurora global databases and Amazon DynamoDB global tables have default Amazon CloudWatch metrics to aid in monitoring replication lag. An Aurora global database consists of one primary Region where your data is written, and up to five read-only secondary Regions. DynamoDB global tables consist of multi-active replica tables across any number of Regions that your data is written to and read from.

Engineering the workload to take advantage of event-driven architectures is a benefit for a multi-Region strategy, because it means that the workload can embrace asynchronous replication of data and enables the reconstruction of state by replaying events. Because streaming and messaging services buffer message payload data in a single Region, a Regional failover or failback process must include a mechanism to redirect client input data flows. The process must also reconcile in-flight or undelivered payloads stored in the Region that experienced the disruption.

If you choose the CAP consistency requirement and use a synchronously replicated database across Regions to support your applications that run concurrently from multiple Regions, you remove the risk of data loss and keep the data in sync between Regions. However, this introduces higher latency characteristics, because writes need to commit to more than one Region, and the Regions can be hundreds or thousands of miles from one another. You need to account for this latency characteristic in your application design. In addition, synchronous replication can introduce the chance for correlated failures because writes will need to be committed to more than one Region to be successful. If there is an impairment within one Region, you will need to form a quorum for writes to be successful. This typically involves setting up your database in three Regions and establishing a quorum of two out of three Regions. Technologies such as Paxos can help replicate and commit data synchronously but require significant developer investment.

When writes involve synchronous replication across multiple Regions to meet strong consistency requirements, write latency increases by an order of magnitude. A higher write latency is not something that you can typically retrofit into an application without significant changes, such as revisiting the timeout and retry strategy for your application. Ideally, it must be taken into consideration when the application is first being designed. For multi-Region workloads where synchronous replication is a priority, AWS Partner solutions can help.

2.b: Understanding data access patterns

Workload data access patterns are either *read-intensive* or *write-intensive*. Understanding this characteristic for a particular workload will help you select an appropriate multi-Region architecture.

For read-intensive workloads such as static content that is completely read-only, you can achieve an active-active multi-Region architecture that has less engineering complexity when compared with a write-intensive workload. Serving static content at the edge by using a content delivery network (CDN) ensures availability by caching content that's closest to the end user; using feature sets such as origin failover within Amazon CloudFront can help achieve this. Another option is to

deploy stateless compute in multiple Regions and use DNS to route users to the closest Region to read the content. You can use Amazon Route 53 with a geolocation routing policy to achieve this.

For read-intensive workloads that have a larger percentage of read traffic than write traffic, you can use a read local, write global strategy. This means that all write requests go to a database in a specific Region, the data is replicated asynchronously to all other Regions, and reads can be done in any Region. This approach requires a workload to embrace eventual consistency, because local reads might become stale as a result of increased latency for cross-Region replication of writes.

Aurora global databases can help provision read replicas in a standby Region that can solely handle all read traffic locally, and provision a single primary data store in a specific Region to handle write traffic. Data is asynchronously replicated from the primary database to standby databases (read replicas), and the standby databases can be promoted to primary if you need to fail over operations to the standby Region. You can also use DynamoDB in this approach. DynamoDB global tables can provision replica tables across Regions that can each scale to support any volume of local read or write traffic. When an application writes data to a replica table in one Region, DynamoDB automatically propagates the write to the other replica tables in the other Regions. With this configuration, data is asynchronously replicated from a defined primary Region to replica tables in standby Regions. Replica tables in any Region can always accept writes, so promoting a standby Region to primary is managed at the application level. Again, the workload has to embrace eventual consistency, which might require it to be rewritten if it wasn't designed for this from the start.

For write-intensive workloads, a primary Region should be selected and the capability to fail over to a standby Region should be engineered into the workload. Compared with an active-active approach, a primary-standby approach has additional trade-offs. This is because for an active-active architecture, the workload has to be rewritten to handle intelligent routing to Regions, establish session affinity, ensure idempotent transactions, and handle potential conflicts.

Most workloads that use a multi-Region approach for resilience won't require an active-active approach. You can use a sharding strategy to provide increased resilience by limiting the scope of impact of an impairment across the client base. If you can effectively shard a client base, you can select different primary Regions for each shard. For example, you can shard clients so that half of the clients are aligned to Region one and half are aligned to Region two. By treating Regions as cells, you can create a multi-Region cell approach, which results in reducing the scope of impact for your workload. For more information, see the AWS re:Invent presentation about this approach.

You can combine the sharding approach with a primary-standby approach to provide failover capabilities for the shards. You will need to engineer a tested failover process into the workload

and a process for data reconciliation as well, to ensure transactional consistency of the data stores after failover. These are covered in greater detail later in this guide.

Key guidance

- There is a high probability that writes pending for replication won't be committed to the standby Region when there is a failure. Data will be unavailable until replication resumes (assuming asynchronous replication).
- As part of failover, a data reconciliation process will be needed to ensure that a transactionally consistent state is maintained for data stores that use asynchronous replication. This requires specific business logic and is not something that is handled by the data store itself.
- When strong consistency is required, workloads will need to be modified to tolerate the required latency of a data store that synchronously replicates.

Multi-Region fundamental 3: Understanding your workload dependencies

A specific workload might have several dependencies in a Region, such as AWS services used, internal dependencies, third-party dependencies, network dependencies, certificates, keys, secrets, and parameters. To ensure operation of the workload during a failure scenario, there should be no dependencies between the primary Region and the standby Region; each should be able to operate independently of the other. To achieve this, scrutinize all dependencies in the workload to make sure that they are available within each Region. This is required because a failure in the primary Region should not affect the standby Region. In addition, you must understand how the workload operates when a dependency is in a degraded state or completely unavailable, so you can engineer solutions to handle this appropriately.

3.a: AWS services

When you design a multi-Region architecture, it's important to understand the AWS services that will be used, the multi-Region features of those services, and what solutions you will need to engineer to accomplish multi-Region goals. For example, Amazon Aurora and Amazon DynamoDB can asynchronously replicate data to a standby Region. All AWS service dependencies will need to be available in all Regions that a workload is going to run from. To confirm that the services you use are available in the desired Regions, review the AWS services by Region list.

3.b: Internal and third-party dependencies

Make sure that every workload's internal dependencies are available in the Regions from which they operate. For example, if the workload is composed of many microservices, identify all the microservices that comprise a business capability and verify that all those microservices are deployed in each Region from which the workload operates. Alternatively, define a strategy to gracefully handle microservices that become unavailable.

Cross-Region calls between microservices within a workload are not advised, and Regional isolation should be maintained. This is because creating cross-Region dependencies adds the risk of correlated failure, which offsets the benefits of isolated Regional implementations of the workload. On-premises dependencies might be part of the workload as well, so it is important to understand how characteristics of these integrations could change if the primary Region were to change. For

example, if the standby Region is located farther from the on-premises environment, the increased latency might have a negative impact.

Understanding software as a service (SaaS) solutions, software development kits (SDKs), and other third-party product dependencies, and being able to exercise scenarios where these dependencies are either degraded or unavailable will provide more insight into how the chain of systems operates and behaves under different failure modes. These dependencies could be within your application code, such as managing secrets externally by using AWS Secrets Manager, or they could involve a third-party vault solution (such as HashiCorp), or authentication systems that have a dependency on AWS IAM Identity Center for federated logins.

Having redundancy when it comes to dependencies can increase resilience. If a SaaS solution or third-party dependency uses the same primary AWS Region as the workload, work with the vendor to determine if their resilience posture matches your requirements for the workload.

Additionally, be aware of shared fate between the workload and its dependencies, such as third-party applications. If the dependencies are not available in (or from) a secondary Region after a failover, the workload might not recover fully.

3.c: Failover mechanism

DNS is commonly used as a failover mechanism to shift traffic away from the primary Region to a standby Region. Critically review and scrutinize all dependencies the failover mechanism takes. For example, if your workload uses Amazon Route 53, understanding that the control plane is hosted in `us-east-1` means you are taking a dependency on the control plane in that specific Region. This is not recommended as part of a failover mechanism if the primary Region is also `us-east-1` because it creates a single point of failure. If you use another failover mechanism, you should have a deep understanding of scenarios in which failover wouldn't work as expected, and then plan for contingency or develop a new mechanism if required. Review the blog post Creating Disaster Recovery Mechanisms Using Amazon Route 53 to learn about approaches you can use to fail over successfully.

As discussed in the previous section, all microservices that are part of a business capability need to be available in each Region in which the workload is deployed. As part of the failover strategy, all microservices that are part of the business capability should fail over together to remove the chance of cross-Region calls. Alternatively, if microservices fail over independently, there is a potential for undesirable behavior such as microservices potentially making cross-Region calls. This introduces latency and could lead to the workload becoming unavailable during client timeouts.

3.d: Configuration dependencies

Certificates, keys, secrets, Amazon Machine Images (AMIs), container images, and parameters are part of the dependency analysis needed when designing for a multi-Region architecture. Whenever possible, it's best to localize these components within each Region so they do not have shared fate between Regions for these dependencies. For example, you should vary the expiration dates of certificates to prevent a scenario where an expiring certificate (with alarms set to "notify in advance") impacts multiple Regions.

Encryption keys and secrets should be Region-specific as well. That way, if there is an error in the rotation of a key or secret, the impact is limited to a specific Region.

Lastly, any workload parameters should be stored locally for the workload to retrieve in the specific Region.

Key guidance

- A multi-Region architecture benefits from physical and logical separation between Regions. Introducing cross-Region dependencies at the application layer breaks this benefit. Avoid such dependencies.
- Failover controls should work with no dependencies on the primary Region.
- Failover should be coordinated across a user journey to remove the possibility of increased latency and dependency of cross-Region calls.

Multi-Region fundamental 4: Operational readiness

Operating a multi-Region workload is a complex task that comes with operational challenges that are specific to a multi-Region architecture. These include AWS account management, retooled deployment processes, creating a multi-Region observability strategy, creating and testing recovery processes, and then managing the cost. An Operational Readiness Review (ORR) can help teams prepare a workload for production, whether it's running in a single Region or across multiple Regions.

4.a: AWS account management

To deploy a workload across AWS Regions, make sure that there is parity across all AWS service quotas within an account across Regions. First, identify all AWS services that are part of the architecture, look at the planned usage in the standby Regions, and then compare planned usage to current usage. In some cases, if the standby Region hasn't been used before, you can reference the default service quotas to understand the starting point. Then, across all the services that will be used, request a quota increase by using the Service Quotas console (login required) or APIs.

Configure AWS Identity and Access Management (IAM) roles in each Region to give operators, automation tooling, and AWS services the appropriate permissions to resources within the standby Region. To achieve Regional isolation for multi-Region architectures, isolate roles by Region. Make sure that permissions are in place before going live with a standby Region.

4.b: Deployment practices

Multi-Region capabilities can make it complicated to deploy a workload to multiple Regions. You need to make sure that you deploy to one Region at a time. For example, if you use an active-passive approach, you should deploy to the primary Region first and then to the standby Region. AWS CloudFormation helps you deploy infrastructure to a single or multiple Regions, and can be tailored according to your needs. AWS CodePipeline helps you build a continuous integration/continuous delivery (CI/CD) pipeline, which has cross-Region actions that allow deployment to Regions that are different from the Region the pipeline is in. This, combined with robust deployment strategies such as blue/green, allows for a minimum to zero downtime deployment.

However, the deployment of stateful capabilities can become more complex when the state of the application or data is not externalized to a persistent store. In these situations, carefully tailor the

deployment process to suit your needs. Design the deployment pipeline and process to deploy to one Region at a time instead of deploying to multiple Regions simultaneously. This reduces the chance of correlated failures between the Regions. To learn about techniques Amazon uses to automate software deployments, see the AWS Builders' Library article Automating safe, hands-off deployments.

4.c: Observability

When you design for multi-Region, consider how you will monitor the health of all components in each Region to get a holistic view of Regional health. This could include monitoring metrics for replication lag, which is not a consideration for a single-Region workload.

When you build a multi-Region architecture, consider observing the performance of the workload from the standby Regions as well. This includes having health checking and canaries (synthetic testing) running from the standby Region to provide an outside view of the health of the primary Region. In addition, you can use Amazon CloudWatch Internet Monitor to understand the state of the external network and performance of your workloads from an end user's perspective. The primary Region should have the same observability in place to monitor the standby Region.

The canaries from the standby Region should monitor customer experience metrics to determine the overall health of the workload. This is required because if there is a problem in the primary Region, the observability in the primary could be impaired and would impact your ability to assess the health of the workload.

In that case, observing outside that Region can provide insight. These metrics should be rolled up into dashboards that are available in each Region and alarms that are created in each Region. Because CloudWatch is a Regional service, having alarms in both Regions is a requirement. This monitoring data will be used to make the call to fail over from a primary to a standby Region.

4.d: Processes and procedures

The best time to answer the question, "When should I fail over?" is long before you need to. Define recovery plans that are inclusive of people, processes, and technology well in advance of an issue, and test them regularly. Decide on a recovery decision framework. If there is a well-practiced recovery process and the time to recovery is well understood, you can choose to start the recovery process by using a failover that meets the RTO target. This point in time could be immediately after an issue with the application in the primary Region is identified, or it could be further into an event when recovery options within the application in the Region have been exhausted.

The failover action itself should be 100 percent automated, but the decision to activate the failover should be made by humans—usually a small number of predetermined individuals in the organization. These individuals should consider data loss and information about the event. Also, the criteria for a failover need to be clearly defined and globally understood within the organization. To define and complete these processes, you can use AWS Systems Manager runbooks, which allow for complete end-to-end automation and ensure consistency of processes running during testing and failover.

These runbooks should be available in the primary and standby Regions to start the failover or failback processes. When this automation is in place, define and follow a regular testing cadence. This ensures that when there is an actual event, the response follows a well-defined, practiced process that the organization has confidence in. It's also important to consider the established tolerances for data reconciliation processes. Confirm that the proposed process meets established RPO/RTO requirements.

4.e: Testing

Having an untested recovery approach is equal to not having a recovery approach. A basic level of testing would be to run a recovery procedure to switch the operating Region for your application. Sometimes this is referred to as an *application rotation* approach. We recommend that you build the capability to switch Regions into your normal operating posture; however, this test alone is not enough.

Resilience testing is also critical for validating an application's recovery approach. This involves injecting particular failure scenarios, understanding how your application and recovery process react, and then implementing any mitigations required if the test didn't go as planned. Testing your recovery procedure in the absence of errors won't tell you how your application behaves as a whole when faults occur. You must develop a plan to test your recovery against expected failure scenarios. AWS Fault Injection Service provides a growing list of scenarios to get you started.

This is especially important for high availability applications, where rigorous testing is required to ensure that business continuity targets are met. Proactively testing recovery capabilities reduces the risk of failures in production, which builds confidence that the application can achieve a desired bounded recovery time. Regular testing also builds operational expertise, which allows the team to quickly and reliably recover from outages when they occur. Exercising the human element, or process, of your recovery approach is just as critical as the technical aspects.

4.f: Cost and complexity

Cost implications of a multi-Region architecture are driven by higher infrastructure usage, operational overhead, and resource time. As mentioned previously, the infrastructure cost in a standby Region is similar to the infrastructure cost in a primary Region when pre-provisioning, so it doubles your total cost. Provision capacity so that it is sufficient for daily operations but still reserves enough buffer capacity to tolerate spikes in demand. Then configure the same limits in each Region.

Additionally, if you are adopting an active-active architecture, you might have to make application-level changes to run your application successfully in a multi-Region architecture. These changes can be time-intensive and resource-intensive to design and operate. At a minimum, organizations need to spend time understanding technical and business dependencies in each Region, and designing failover and failback processes.

Teams should also go through normal failover and failback exercises to feel comfortable with runbooks that would be used during an event. Although these exercises are crucial to getting the expected outcome from a multi-Region investment, they represent an opportunity cost, and take time and resources away from other activities.

4.g: Organizational multi-Region failover strategy

AWS Regions provide fault isolation boundaries that prevent correlated failure and contain the impact from AWS service impairments, when they occur, to a single Region. You can use these fault boundaries to build multi-Region applications that consist of independent, fault-isolated replicas in each Region to limit shared fate scenarios. This allows you to build multi-Region applications and use a range of approaches—from backup and restore, to pilot light, to active-active—to implement your multi-Region architecture. However, applications typically don't operate in isolation, so consider both the components you will use and their dependencies as part of your failover strategy. Generally, multiple applications work together to support a *user story*, which is a specific capability offered to an end user, such as posting a picture and caption on a social media app or checking out on an ecommerce site. Because of this, you should develop an organizational multi-Region failover strategy that provides the necessary coordination and consistency to make your approach successful.

There are four high-level strategies that organizations can pick from to guide a multi-Region approach. These are listed from the most granular to the broadest approach:

- Component-level failover
- Individual application failover
- Dependency graph failover
- Entire application portfolio failover

Each strategy has trade-offs and addresses different challenges, including flexibility of failover decision-making, ability to test the failover combinations, presence of modal behavior, and organizational investment in planning and implementation. To dive into each strategy in more detail, see the AWS blog post Creating an organizational multi-Region failover strategy.

Key guidance

- Review all AWS service quotas to make sure that they are in parity across all Regions in which the workload will operate.
- The deployment process should target one Region at a time instead of involving multiple Regions simultaneously.
- Additional metrics such as replication lag are specific to multi-Region scenarios and should be monitored.
- Extend monitoring for the workload beyond the primary Region. Monitor customer experience metrics for each Region, and measure this data from outside each Region in which a workload is running.
- Test failover and failback regularly. Implement a single runbook for failover and failback processes and use it both for testing and live events. Runbooks for testing and live events should not be different.
- Understand the trade-offs of the failover strategies. Implement a dependency graph or entire application portfolio strategy.

Conclusion and resources

This guide covered common use cases for multi-Region architectures, fundamentals of implementing these architectures, and implications of this approach. You can apply these fundamentals to any workload and use the information as a framework to help decide whether a multi-Region architecture is the right approach for your business.

For more information, see the following resources:

- AWS Architecture Center
- AWS Well-Architected Framework
- AWS Well-Architected Tool
- Creating an organizational multi-Region failover strategy (AWS blog post)
- AWS Multi-Region Capabilities (AWS re:Post article)

Document history

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

Change	Description	Date
Updates	Updates throughout the guide.	December 27, 2024
Initial publication	—	December 20, 2022

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 Bitbucket Cloud. Each version of the code is called a *branch*. In a microservice structure, each repository is devoted to a single piece of functionality. A single CI/CD pipeline can use multiple repositories.

cold cache

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

cold data

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

computer vision (CV)

A field of AI that uses machine learning to analyze and extract information from visual formats such as digital images and videos. For example, AWS Panorama offers devices that add CV to on-premises camera networks, and Amazon SageMaker AI provides image processing algorithms for CV.

configuration drift

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

configuration management database (CMDB)

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

conformance pack

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

continuous integration and continuous delivery (CI/CD)

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

CV

See computer vision.

D

data at rest

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

data classification

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

data drift

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

data in transit

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

data mesh

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

data minimization

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

data perimeter

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

data preprocessing

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

data provenance

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

data subject

An individual whose data is being collected and processed.

data warehouse

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

database definition language (DDL)

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

database manipulation language (DML)

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

DDL

See database definition language.

deep ensemble

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

deep learning

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

defense-in-depth

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

delegated administrator

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

deployment

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

development environment

See environment.

detective control

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

development value stream mapping (DVSM)

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

digital twin

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

dimension table

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

disaster

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

disaster recovery (DR)

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

DML

See database manipulation language.

domain-driven design

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

DR

See disaster recovery.

drift detection

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

DVSM

See development value stream mapping.

E

EDA

See exploratory data analysis.

EDI

See electronic data interchange.

edge computing

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

electronic data interchange (EDI)

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

encryption

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

encryption key

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

endianness

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

endpoint

See service endpoint.

endpoint service

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

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

enterprise resource planning (ERP)

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

envelope encryption

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

environment

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

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

epic

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

ERP

See enterprise resource planning.

exploratory data analysis (EDA)

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

F

fact table

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

fail fast

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

fault isolation boundary

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

feature branch

See branch.

features

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

feature importance

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

feature transformation

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

few-shot prompting

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

FGAC

See fine-grained access control.

fine-grained access control (FGAC)

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

flash-cut migration

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

FM

See foundation model.

foundation model (FM)

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

G

generative AI

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

geo blocking

See geographic restrictions.

geographic restrictions (geo blocking)

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

Gitflow workflow

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

golden image

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

greenfield strategy

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

guardrail

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

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

H

HA

See high availability.

heterogeneous database migration

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

high availability (HA)

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

historian modernization

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

holdout data

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

homogeneous database migration

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

hot data

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

hotfix

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

hypercare period

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

I

IaC

See infrastructure as code.

identity-based policy

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

idle application

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

IIoT

See industrial Internet of Things.

immutable infrastructure

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

inbound (ingress) VPC

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

incremental migration

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

Industry 4.0

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

infrastructure

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

infrastructure as code (IaC)

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

industrial Internet of Things (IIoT)

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

inspection VPC

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

Internet of Things (IoT)

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

interpretability

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

IoT

See Internet of Things.

IT information library (ITIL)

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

IT service management (ITSM)

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

ITIL

See IT information library.

ITSM

See IT service management.

L

label-based access control (LBAC)

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

landing zone

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

large language model (LLM)

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

large migration

A migration of 300 or more servers.

LBAC

See label-based access control.

least privilege

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

lift and shift

See 7 Rs.

little-endian system

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

LLM

See large language model.

lower environments

See environment.

M

machine learning (ML)

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

main branch

See branch.

malware

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

managed services

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

manufacturing execution system (MES)

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

MAP

See Migration Acceleration Program.

mechanism

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

member account

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

MES

See manufacturing execution system.

Message Queuing Telemetry Transport (MQTT)

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

microservice

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

microservices architecture

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

Migration Acceleration Program (MAP)

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

migration at scale

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

migration factory

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

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

migration metadata

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

migration pattern

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

Migration Portfolio Assessment (MPA)

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

Migration Readiness Assessment (MRA)

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

migration strategy

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

ML

See machine learning.

modernization

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

modernization readiness assessment

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

monolithic applications (monoliths)

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

MPA

See Migration Portfolio Assessment.

MQTT

See Message Queuing Telemetry Transport.

multiclass classification

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

mutable infrastructure

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

O

OAC

See origin access control.

OAI

See origin access identity.

OCM

See organizational change management.

offline migration

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

OI

See operations integration.

OLA

See operational-level agreement.

online migration

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

OPC-UA

See Open Process Communications - Unified Architecture.

Open Process Communications - Unified Architecture (OPC-UA)

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

operational-level agreement (OLA)

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

operational readiness review (ORR)

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

operational technology (OT)

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

operations integration (OI)

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

organization trail

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

organizational change management (OCM)

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

origin access control (OAC)

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

origin access identity (OAI)

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

ORR

See operational readiness review.

OT

See operational technology.

outbound (egress) VPC

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

P

permissions boundary

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

personally identifiable information (PII)

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

PII

See personally identifiable information.

playbook

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

PLC

See programmable logic controller.

PLM

See product lifecycle management.

policy

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

polyglot persistence

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

portfolio assessment

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

predicate

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

predicate pushdown

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

preventative control

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

principal

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

privacy by design

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

private hosted zones

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

proactive control

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

product lifecycle management (PLM)

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

production environment

See environment.

programmable logic controller (PLC)

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

prompt chaining

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

pseudonymization

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

publish/subscribe (pub/sub)

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

Q

query plan

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

query plan regression

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

R

RACI matrix

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

RAG

See Retrieval Augmented Generation.

ransomware

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

RASCI matrix

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

RCAC

See row and column access control.

read replica

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

re-architect

See 7 Rs.

recovery point objective (RPO)

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

recovery time objective (RTO)

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

refactor

See 7 Rs.

Region

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

regression

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

rehost

See 7 Rs.

release

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

relocate

See 7 Rs.

replatform

See 7 Rs.

repurchase

See 7 Rs.

resiliency

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

resource-based policy

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

responsible, accountable, consulted, informed (RACI) matrix

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

responsive control

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

retain

See 7 Rs.

retire

See 7 Rs.

Retrieval Augmented Generation (RAG)

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

rotation

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

row and column access control (RCAC)

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

RPO

See recovery point objective.

RTO

See recovery time objective.

runbook

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

S

SAML 2.0

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

SCADA

See supervisory control and data acquisition.

SCP

See service control policy.

secret

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

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

security by design

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

security control

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

security hardening

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

security information and event management (SIEM) system

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

security response automation

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

server-side encryption

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

service control policy (SCP)

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

service endpoint

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

service-level agreement (SLA)

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

service-level indicator (SLI)

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

service-level objective (SLO)

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

shared responsibility model

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

SIEM

See security information and event management system.

single point of failure (SPOF)

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

SLA

See service-level agreement.

SLI

See service-level indicator.

SLO

See service-level objective.

split-and-seed model

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

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

SPOF

See single point of failure.

star schema

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

strangler fig pattern

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

subnet

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

supervisory control and data acquisition (SCADA)

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

symmetric encryption

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

synthetic testing

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

system prompt

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

T

tags

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

target variable

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

task list

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

test environment

See environment.

training

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

transit gateway

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

trunk-based workflow

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

trusted access

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

tuning

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

two-pizza team

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

U

uncertainty

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

undifferentiated tasks

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

upper environments

See environment.

V

vacuuming

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

version control

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

VPC peering

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

vulnerability

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

W

warm cache

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

warm data

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

window function

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

workload

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

workstream

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

WORM

See write once, read many.

WQF

See AWS Workload Qualification Framework.

write once, read many (WORM)

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

Z

zero-day exploit

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

zero-day vulnerability

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

zero-shot prompting

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

zombie application

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