Applying the AWS Well-Architected Framework for Amazon Neptune Analytics

AWS Prescriptive Guidance

AWS Prescriptive Guidance: Applying the AWS Well-Architected Framework for Amazon Neptune Analytics

Table of Contents

Introduction **1**
- Intended audience 1
- Objectives 2

Operational excellence pillar **3**
- Automate deployment using an IaC approach 3
- Design for operations 4
- Make frequent, small, reversible changes 4
- Implement observability for actionable insights 5
- Learn from all operational failures 5
- Use logging capabilities to monitor for unauthorized or anomalous activity 6

Security pillar **7**
- Implement data security 7
- Secure your networks 8
- Implement authentication and authorization 8

Reliability pillar **10**
- Understand Neptune service quotas 10
- Understand Neptune deployment patterns 11
- Manage and scale Neptune clusters 12
- Manage backups and failover events 13

Performance efficiency pillar **14**
- Understand graph modeling for analytics 14
- Optimize queries 16
- Optimize writes 17
- Right-size graphs 18

Cost optimization pillar **19**
- Understand usage patterns and services needed 19
- Select resources with attention to cost 20

Sustainability pillar **22**
- Consider your AWS Region selection 22
- Optimize consumption 22
- Optimize software development and architecture patterns 23

Resources **25**
- References 25
- Blog posts and videos 25

Training 25
Document history 26
Glossary 27
..... 27
A 28
B 31
C 33
D 36
E 40
F 42
G 44
H 45
I 46
L 49
M 50
O 54
P 57
Q 59
R 60
S 63
T 67
U 68
V 69
W 69
Z 70

Applying the AWS Well-Architected Framework for Amazon Neptune Analytics

Michael Havey, Amazon Web Services (AWS)

December 2024 (document history)

You can build graph-based solutions on Amazon Web Services (AWS) by using Amazon Neptune. Neptune includes Neptune Analytics, a memory-optimized graph analytics engine that can quickly analyze large amounts of graph data to get insights and find trends. It can perform analytics on data in your existing Neptune database cluster, or you can load and analyze data from external datasets. This guide provides prescriptive guidance for applying the AWS Well-Architected Framework principles when you plan your Neptune Analytics deployment. Applying the AWS Well-Architected Framework for Amazon Neptune covers the same topic for a Neptune database.

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

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

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

This guide provides information from the Well-Architected Framework design pillars and best practices, and considerations to keep in mind when you deploy Neptune Analytics on AWS.

Intended audience

This guide is intended for data engineers, solutions architects, and data analysts who design and implement solutions that use graphs on AWS.

Objectives

This guide can help you and your organization do the following:

- Choose from the supported deployment options.
- Follow the AWS Well-Architected design patterns that help you improve resiliency and security.
- Design your queries for optimal performance and cost savings.
- Learn how to be operationally efficient when managing your Neptune Analytics graph in production.

Operational excellence pillar

The operational excellence pillar of the AWS Well-Architected Framework focuses on running and monitoring systems, and continually improving processes and procedures. It includes the ability to support development and run workloads effectively, gain insight into their operation, and continuously improve supporting processes and procedures to deliver business value. You can reduce operational complexity through self-healing workloads, which detect and remediate most issues without human intervention. You can work toward this goal by following the best practices described in this section, and use Amazon Neptune Analytics metrics, APIs, and mechanisms to properly respond when your workload deviates from expected behavior.

This discussion of the operational excellence pillar focuses on the following key areas:

- Infrastructure as code (IaC)
- Change management
- Resiliency strategies
- Incident management
- Audit reporting for compliance
- Logging and monitoring

Automate deployment using an IaC approach

Best practices for automating deployment on Neptune by using IaC include the following:

- Apply IaC to deploy Neptune Analytics graphs and related resources. For consistent environment configuration, use the support for Neptune Analytics provided by AWS CloudFormation to provision graphs and private endpoints.
- Use CloudFormation to provision Neptune notebook instances on Amazon SageMaker AI. You can use notebooks to query and visualize data in a Neptune Analytics graph.
- When you create a Neptune Analytics graph from an existing source, such as a Neptune database cluster or snapshot, or data files staged in Amazon Simple Storage Service (Amazon S3), monitor the bulk import task.
- Automate Neptune Analytics operational procedures, such as resizing the graph, deleting and snapshotting the graph, restoring the graph from a snapshot, and resetting and reloading

the graph. Use the Neptune Analytics API, which is available through the AWS Command Line Interface (AWS CLI) or SDKs.

- Assess the required uptime of your graph. Analytics is often ephemeral; the graph is required only for the time you need to run algorithms. If this is the case, use the AWS CLI or SDKs to snapshot and delete the graph when it is no longer required. You can then restore it from a snapshot later, if necessary.
- Store connection strings externally from your client. You can store connection strings in AWS Secrets Manager, Amazon DynamoDB, or any location where they can be changed dynamically.
- Use tags to add metadata to your Neptune Analytics resources, and track usage based on tags. Tags help organize your resources. For example, you can apply a common tag to resources in a specific environment or application. You can also use tags to analyze billing of resource usage; for more information, see Organizing and tracking costs using AWS cost allocation tags in the *AWS Billing User Guide*. Additionally, you can use conditions in your AWS Identity and Access Management (IAM) policies to control access to AWS resources based on the tags used on that resource. You can do this by using the global `aws:ResourceTag/tag-key` condition key. For more information, see Controlling access to AWS resources in the *IAM User Guide*.

Design for operations

Adopt approaches to improve how you operate Neptune Analytics graphs:

- Maintain separate Neptune Analytics graphs for development, test, and production use. These graphs might have different datasets, users, and operational controls.
- Maintain separate Neptune Analytics graphs for different uses. For example, if two groups of analytical users require separate graphs with different timelines, models, performance and availability SLAs, and usage patterns, maintain separate graphs for each group.
- Prepare users and operational staff for Neptune Analytics maintenance updates.

Make frequent, small, reversible changes

The following recommendations focus on small, reversible changes you can make to minimize complexity and reduce the likelihood of workload disruption:

- Store IaC templates and scripts in a source control service such as GitHub or GitLab.

Important

Do not store AWS credentials in source control.

- Require IaC deployments to use a continuous integration and continuous delivery (CI/CD) service such as AWS CodeDeploy or AWS CodeBuild. Compile, test, and deploy code in a non-production Neptune Analytics environment before promoting it to a production graph.

Implement observability for actionable insights

Gain a comprehensive understanding of workload behavior, performance, reliability, cost, and health. The following recommendations help you gain that level of understanding in Neptune Analytics:

- Monitor Amazon CloudWatch metrics for Neptune Analytics. From these metrics, you can determine the size of a graph (number of nodes, edges, and vectors, plus total byte size), CPU utilization, and query request and error rates.
- Create CloudWatch dashboards and alarms for key metrics such as `NumQueuedRequestsPerSec`, `NumOpenCypherRequestsPerSec`, `GraphStorageUsagePercent`, `GraphSizeBytes`, and `CPUUtilization` as well as Neptune client responses found in your application logs.
- Set notifications to monitor the health of the Neptune Analytics graph such as when graph size, request rate, or CPU utilization exceeds your threshold. For example, if `GraphStorageUsagePercent` has climbed to 90 percent on a graph you intend to grow significantly, decide whether to increase memory-optimized Neptune Capacity Unit (m-NCU) capacity. If current m-NCU is 128, increasing it to 256 will reduce storage by about 45 percent. If `NumQueuedRequestsPerSec` is often greater than zero, consider increasing m-NCU capacity to provide more compute capacity. Alternatively, you can reduce client-side concurrency.

Learn from all operational failures

A self-healing infrastructure is a long-term effort that develops in iterations as rare problems occur or responses are not as effective as desired. Adopting the following practices drives focus toward that goal:

- Drive improvement by learning from all failures.
- Share what is learned across teams and the organization. If multiple teams within your organization use Neptune, create a common chatroom or user group to share learnings and best practices.

Use logging capabilities to monitor for unauthorized or anomalous activity

Use logging to observe anomalous performance and activity patterns. Consider the following best practices:

- Neptune Analytics supports logging of control plane actions by using AWS CloudTrail. For more information, see Logging Neptune Analytics API calls using AWS CloudTrail. Through this capability, you can track the creation, update, and deletion of Neptune Analytics resources. For robust monitoring and alerting, you can also integrate CloudTrail events with Amazon CloudWatch Logs. To enhance your analysis of Neptune Analytics service activity and identify changes in activities for an AWS account, you can query CloudTrail logs by using Amazon Athena. For example, you can use queries to identify trends and further isolate activity by attributes such as source IP address or user.
- You can also use CloudTrail to enable logging of Neptune Analytics data plane activities such as query executions. You can view which queries are being run, their frequency, and their source. By default, CloudTrail doesn't log data events. Additional charges apply for data events. For more information, see AWS CloudTrail pricing.
- You can also log your application calls to Neptune Analytics in either the control plane or the data plane. For example, if you use the AWS SDK for Python (Boto3) to make queries, you can enable debug-level logging to obtain a trace of queries to the console or file. This is useful during development. We also recommend that you catch and log exceptions from your application.

Security pillar

Cloud security is the highest priority at AWS. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations. Security is a shared responsibility between you and AWS. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of AWS security as part of AWS compliance programs. To learn about the compliance programs that apply to Neptune, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors, including the sensitivity of your data, your company's requirements, and applicable laws and regulations. For more information about data privacy, see the Data Privacy FAQs. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post.

The security pillar of the AWS Well-Architected Framework helps you understand how to apply the shared responsibility model when you use Neptune Analytics. The following topics explain how to configure Neptune Analytics to meet your security and compliance objectives. You also learn how to use other AWS services that help you monitor and secure your Neptune Analytics resources. The security pillar includes the following key focus areas:

- Data security
- Network security
- Authentication and authorization

Implement data security

Data leakage and breaches put your customers at risk and can cause substantial negative impact on your company. The following best practices help protect your customer data from inadvertent and malicious exposure:

- Graph names, tags, IAM roles, and other metadata should not contain confidential or sensitive information, because that data might appear in billing or diagnostic logs.
- URIs or links to external servers stored as data in Neptune should not contain credential information to validate requests.
- A Neptune Analytics graph is encrypted at rest. You can use the default key or an AWS Key Management Service (AWS KMS) key of your choosing to encrypt the graph. You can also encrypt snapshots and data that's exported to Amazon S3 during bulk import. You can remove the encryption when the import is complete.
- When you use the openCypher language, practice proper input validation and parameterization techniques to prevent SQL injection and other forms of attacks. Avoid constructing queries that use string concatenation with user-supplied input. Use parameterized queries or prepared statements to safely pass input parameters to the graph database. For more information, see Examples of openCypher parameterized queries in the Neptune documentation.

Secure your networks

You can enable a Neptune Analytics graph for public connectivity so it can be reached from outside a virtual private cloud (VPC). This connectivity is disabled by default. The graph requires IAM authentication. The caller must obtain an identity and have permissions to use the graph. For example, to run an openCypher query, the caller would need to have read, write, or delete permissions on the specific graph.

You can also create private endpoints for the graph to access the graph from within a VPC. When you create the endpoint, you specify the VPC, subnets, and security groups to restrict access to call the graph.

To protect your data in transit, Neptune Analytics enforces SSL connections through HTTPS to the graph. For more information, see Data protection in Neptune Analytics in the Neptune Analytics documentation.

Implement authentication and authorization

Calls to a Neptune Analytics graph require IAM authentication. The caller must obtain an identity and possess sufficient permissions to perform the action on the graph. For descriptions of API actions and their required permissions, see the Neptune Analytics API documentation. You can enforce condition checks to restrict access by tag.

IAM authentication uses the AWS Signature Version 4 (SigV4) protocol. To simplify usage from your application, we recommend that you use an AWS SDK. For example, in Python, use the Boto3 client for Neptune Graph, which abstracts SigV4.

When you load data into the graph, batch loading uses the IAM credentials of the caller. The caller must have permissions to download data from Amazon S3 with the trust relationship set up so that Neptune Analytics can assume the role to load the data into the graph from Amazon S3 files.

Bulk import can be performed either during graph creation (by the infrastructure team) or on an existing, empty graph (by the data engineering team that has permissions to start import tasks). In both cases, Neptune Analytics assumes the IAM role that the caller provides as input. This role gives it permission to read and list the contents of the Amazon S3 folder where input data is staged.

Reliability pillar

The AWS Well-Architected Framework reliability pillar encompasses the ability of a workload to perform its intended function correctly and consistently when it's expected to. This includes the ability to operate and test the workload through its entire lifecycle.

A reliable workload starts with upfront design decisions for both software and infrastructure. Your architecture choices impact your workload behavior across all the Well-Architected pillars. For reliability, there are specific patterns you must follow, as discussed in this section.

The reliability pillar focuses on the following key areas:

- Workload architecture, including service quotas and deployment patterns
- Change management
- Failure management

Understand Neptune service quotas

Your AWS account has default quotas (formerly referred to as *limits*) for each AWS service. Unless otherwise noted, each quota is Region-specific. You can request increases for some, but not all, quotas.

To find quotas for Neptune Analytics, open the Service Quotas console. In the navigation pane, choose **AWS services**, and then select **Amazon Neptune Analytics**. Pay attention to quotas on the number of graphs and snapshots, maximum provisioned memory for a graph, and API request rates.

If the maximum provisioned memory isn't sufficient for your dataset, assess which node and edge types are essential for your intended analytical usage. Load a subset of the data so that analytics are possible within an allowable provisioned capacity. Many analytics workloads, especially those that run graph algorithms, only need the topology with a limited set of properties instead of the full transactional graph. (For a discussion of the differences between transactional and analytical workloads, see the Performance efficiency pillar section.)

If the maximum number of graphs isn't sufficient for your intended use:

- Consider combining graphs that have similar uses.

- Assess how many graphs have to run at a given time. If you have an ephemeral analytics use case, snapshot and delete a graph when it is no longer needed. This reduces the number of graphs against the quota.
- Consider provisioning graphs in different AWS accounts.

Understand Neptune deployment patterns

Understand the following decision points when you plan to deploy a Neptune Analytics graph:

- **Seeding**: Decide whether to create an empty graph or load data into it at creation time with data from Amazon S3, an existing Neptune database cluster, or an existing Neptune database snapshot.

  Recommendation: If the source is a Neptune cluster or snapshot, you must load its data at graph creation time. If the source is Amazon S3, load the data at creation time if the effort of loading it is significant and best performed as an infrastructure provisioning activity. If you prefer to load data as an data engineering or application activity, create an empty graph and load data from Amazon S3 later.
- **Capacity**: Estimate the required provisioned capacity for a graph, given the data size and expected application usage.

  Recommendation: At creation time, specify maximum provisioned memory to limit the graph size. This setting is mandatory. You can change capacity later if necessary.
- **Availability and fault tolerance**: Decide whether replicas are required for availability. A replica acts as a warm standby for recovery in case of graph failure. A graph with replicas recovers faster than a graph without replicas. Also consider how long the graph is needed, whether it is for ephemeral analytics only, and, if so, when it will be removed.

  Recommendation: Determine availability requirements—such as how long the graph can be unavailable and when it can be removed—before you create a graph.
- **Networking and security**: Determine whether you need public connectivity, private connectivity, or both, and whether you want to encrypt your data.

  Recommendation**:** Understand organizational requirements—such as whether public connectivity is allowed and where graph client applications will be deployed—before you create a graph.

- **Backups and recovery**: Determine whether snapshots should be created, and, if so, when or under which conditions. Consider whether your organization has disaster recovery (DR) requirements.

  Recommendation: Creating snapshots is a manual activity. Decide when to create snapshots and consider your DR requirements before you create a graph.

Manage and scale Neptune clusters

A Neptune Analytics graph consists of a single, memory-optimized instance. The capacity (m-NCU) of the instance is set at creation time. The instance can be vertically scaled by increasing provisioned capacity through an administrative action; provisioned capacity can also be decreased. Replicas are passive failover targets, so they do not increase the scale of a graph. In this respect, a graph replica differs from a Neptune database read replica, which is an active instance in a Neptune cluster that can process read operations from applications.

Replicas incur cost. The replica is priced at the m-NCU rate of the graph. For example, if a graph is provisioned for 128 m-NCU and has a single replica, the cost is twice that of an equivalent graph that has no replicas.

In analytics, there are two primary reasons to scale up:

- To provide more memory and CPU for analytical queries and algorithms, because the individual query is expensive, the graph algorithm to run is inherently complex and requires more resources given its input, or the concurrent request rate is high. If such queries are encountering out-of-memory errors, scaling up is a reasonable remedy.
- To support a larger graph size than you planned for. For example, if the current provisioned capacity is 128 m-NCU to support 60 GB of source data and you need an additional 40 GB of source data, an increase to 256 m-NCU is warranted.

Monitor CloudWatch metrics for Neptune Analytics, such as `NumQueuedRequestsPerSec`, `NumOpenCypherRequestsPerSec`, `GraphStorageUsagePercent`, `GraphSizeBytes`, and `CPUUtilization`, to determine if scaling is necessary. You can update a graph's configuration through the console, AWS CLI, or SDKs. (For examples and best practices, see the Operational excellence pillar section.)

Manage backups and failover events

Use replicas to ensure that a graph remains available in case of failure. A graph uses log-based persistence to commit changes across Availability Zones in an AWS Region. The replica acts as a warm standby and has access to this data. If there is a failure, the graph resumes operations on the replica. The application continues to use the same endpoint to connect to the graph. In-flight requests during the failure generate errors with a *service unavailable* exception. Consider using a retry with backoff pattern in the application code to catch the error, and try again after a brief interval. New requests made during failover are queued and might experience longer latency.

If no replica is configured and the graph fails, Neptune Analytics recovers from durable storage, but recovery takes longer because Neptune has to re-initialize resources.

Create snapshots of the graph. (Neptune Analytics doesn't take automatic snapshots.) If the graph is modified on a regular basis after creation, take frequent snapshots to capture its current state. Delete older snapshots if restoration to an earlier point in time isn't required.

You can share snapshots with other accounts and across AWS Regions. If you have DR requirements, consider whether restoring the graph in a different Region from a snapshot meets your recovery time objective (RTO) and recovery point objective (RPO) requirements.

Performance efficiency pillar

The performance efficiency pillar of the AWS Well-Architected Framework focuses on how to optimize performance while ingesting or querying data. Performance optimization is an incremental and continual process of the following:

- Confirming business requirements
- Measuring workload performance
- Identifying under-performing components
- Tuning components to meet your business needs

The performance efficiency pillar provides use case-specific guidelines that can help you identify the right graph data model and query languages to use. It also includes best practices to follow when ingesting data into, and consuming data from, Neptune Analytics.

The performance efficiency pillar focuses on the following key areas:

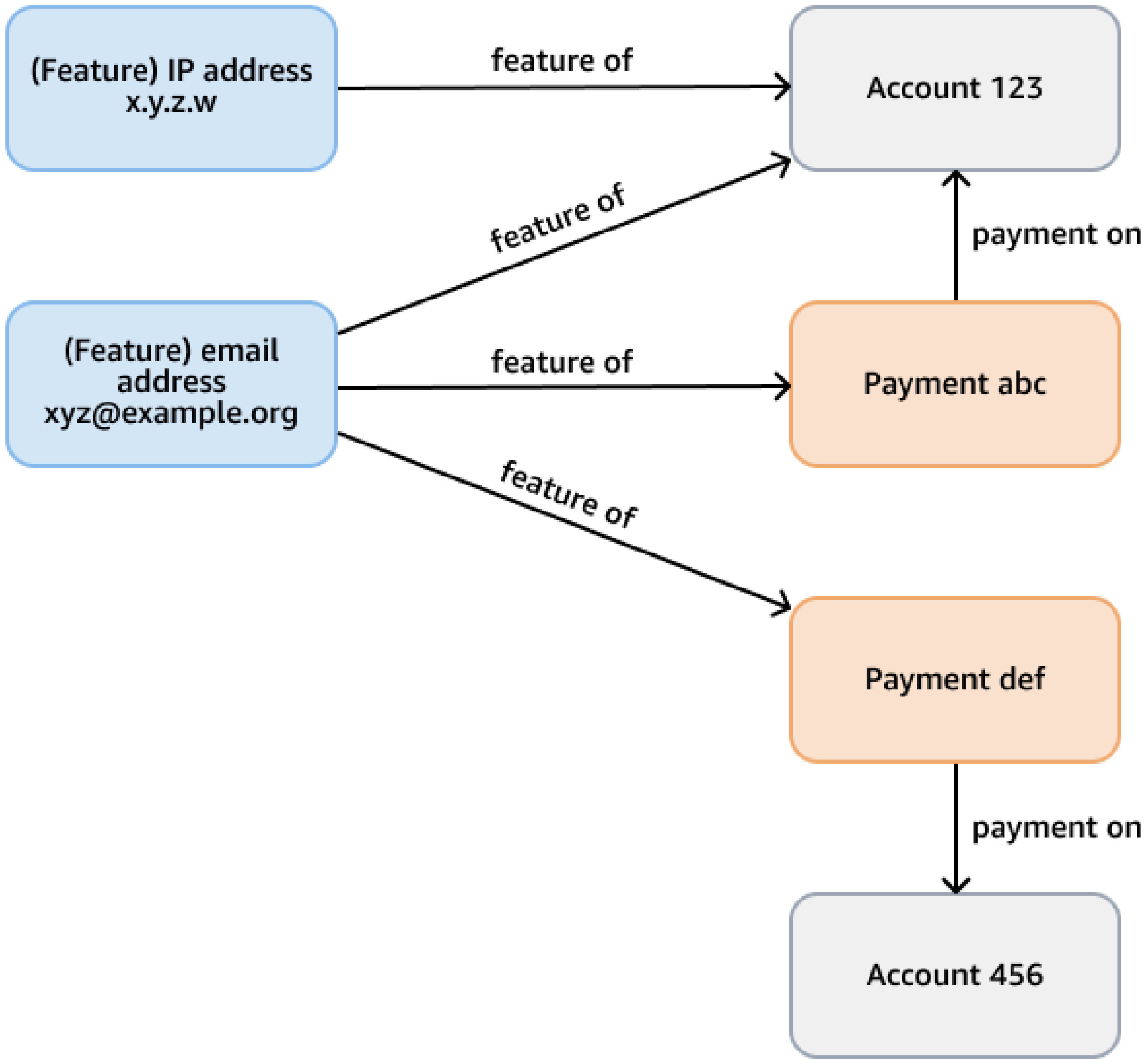
- Graph modeling
- Query optimization
- Graph right-sizing
- Write optimization

Understand graph modeling for analytics

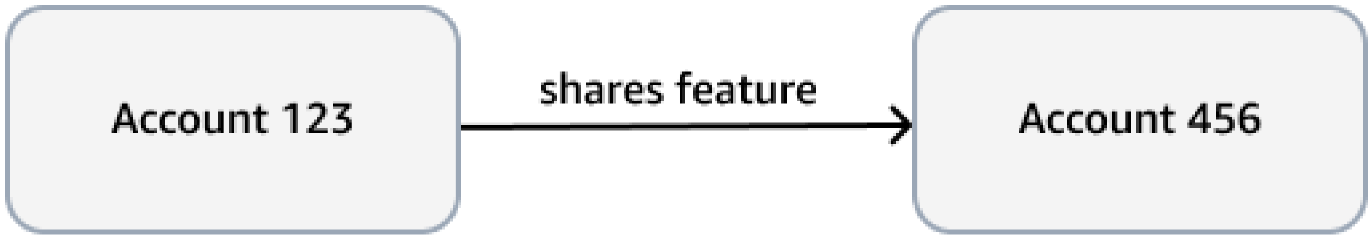
The guide *Applying the AWS Well-Architected Framework for Amazon Neptune* discusses graph modeling for performance efficiency. Modeling decisions that affect performance include choosing which nodes and edges are required, their IDs, their labels and properties, the direction of edges, whether labels should be generic or specific, and generally how efficiently the query engine can navigate the graph to process common queries.

These considerations apply to Neptune Analytics, too; however, it is important to distinguish between transactional and analytical usage patterns. A graph model that is efficient for queries in a transactional database such as a Neptune database might need to be reshaped for analytics.

For example, consider a fraud graph in a Neptune database whose purpose is to check for fraudulent patterns in credit card payments. This graph might have nodes that represent accounts, payments, and features (such as email address, IP address, phone number) of both the account and the payment. This connected graph supports queries such as traversing a variable-length path that starts from a given payment and takes several hops to find related features and accounts. The following figure shows such a graph.



The analytical requirement might be more specific, such as finding communities of accounts that are linked by a feature. You can use the [weakly connected components (WCC)]() algorithm for this purpose. To run it against the model in the previous example is inefficient, because it needs to traverse through several different types of nodes and edges. The model in the next diagram is more efficient. It links `account` nodes with a `shares feature` edge if the accounts themselves —or payments from the accounts—share a feature. For example, `Account 123` has email feature `xyz@example.org`, and `Account 456` uses that same email for a payment (`Payment def`).



The computational complexity of WCC is `O(|E|logD)`, where `|E|` is the number of edges in the graph, and `D` is the diameter (the length of the longest path) that connects nodes. Because the transactional model omits inessential nodes and edges, it optimizes both the number of edges and the diameter, and reduces the WCC algorithm's complexity.

When you use Neptune Analytics, work back from required algorithms and analytical queries. If necessary, reshape the model to optimize these queries. You can reshape the model before you load data into the graph, or write queries that modify existing data in the graph.

Optimize queries

Follow these recommendations to optimize Neptune Analytics queries:

- Use parameterized queries and the query plan cache, which is enabled by default. When you use the plan cache, the engine prepares the query for later use—provided that the query completes in 100 milliseconds or less—which saves time on subsequent invocations.
- For slow queries, run an explain plan to spot bottlenecks and make improvements accordingly.
- If you use vector similarity search, decide if smaller embeddings yield accurate similarity results. You can create, store, and search smaller embeddings more efficiently.
- Follow documented best practices for using openCypher in Neptune Analytics. For example, use flattened maps in an UNWIND clause and specify edge labels where possible.
- When you use a graph algorithm, understand the algorithm's inputs and outputs, its computational complexity, and broadly how it works.
  - Before you call a graph algorithm, use a `MATCH` clause to minimize the input node set. For example, to limit nodes to do breadth-first search (BFS) from, follow the examples provided in the Neptune Analytics documentation.

- Filter on node and edge labels if possible. For example, BFS has input parameters to filter traversal to a specific node label (`vertexLabel`) or specific edge labels (`edgeLabels`).
- Use bounding parameters such as `maxDepth` to limit results.
- Experiment with the `concurrency` parameter. Try it with a value of 0, which uses all available algorithm threads to parallelize processing. Compare that with single-threaded execution by setting the parameter to 1. An algorithm can complete faster in a single thread, especially on smaller inputs such as shallow breadth-first searches where parallelism offers no measurable reduction in execution time and might introduce overhead.
- Choose between similar types of algorithms. For example, Bellman-Ford and delta-stepping are both single-source shortest path algorithms. When testing with your own dataset, try both algorithms and compare the results. Delta-stepping is often faster than Bellman-Ford because of its lower computational complexity. However, performance depends on the dataset and input parameters, particularly the `delta` parameter.

Optimize writes

Follow these practices to optimize write operations in Neptune Analytics:

- Seek the most efficient way to load data into a graph. When you load from data in Amazon S3, use bulk import if the data is greater than 50 GB in size. For smaller data, use batch load. If you get out-of-memory errors when you run batch load, consider increasing the m-NCU value or splitting the load into multiple requests. One way to accomplish this is to split files across multiple prefixes in the S3 bucket. In that case, call batch load separately for each prefix.
- Use bulk import or the batch loader to populate the initial set of graph data. Use transactional openCypher create, update, and delete operations for **small changes only**.
- Use either bulk import or the batch loader with a concurrency of 1 (single threaded) to ingest embeddings into the graph. Try to load embeddings up front by using one of these methods.
- Assess the dimension of vector embeddings needed for accurate similarity search in vector similarity search algorithms. Use a smaller dimension if possible. This results in faster load speed for embeddings.
- Use mutate algorithms to remember algorithmic results if required. For example, the degree mutate centrality algorithm finds the degree of each input node and writes that value as a property of the node. If the connections surrounding those nodes do not subsequently change, the property holds the correct result. There is no need to run the algorithm again.

- Use the graph reset administrative action to clear all nodes, edges, and embeddings if you need to start over. Dropping all nodes, edges, and embeddings by using an openCypher query is not feasible if your graph is large. A single drop query on a large dataset can time out. As size increases, the dataset takes longer to remove and transaction size increases. By contrast, the time to complete a graph reset is roughly constant, and the action provides the option to create a snapshot before you run it.

Right-size graphs

Overall performance depends on the provisioned capacity of a Neptune Analytics graph. Capacity is measured in units called *memory-optimized Neptune Capacity Units (m-NCUs)*. Make sure that your graph is sufficiently sized to support your graph size and queries. Note that increased capacity doesn't necessarily improve the performance of an individual query.

If possible, create the graph by importing data from an existing source such as Amazon S3 or an existing Neptune cluster or snapshot. You can place bounds on mininum and maximum capacity. You can also change provisioned capacity on an existing graph.

Monitor CloudWatch metrics such as `NumQueuedRequestsPerSec`, `NumOpenCypherRequestsPerSec`, `GraphStorageUsagePercent`, `GraphSizeBytes`, and `CPUUtilization` to assess whether the graph is right-sized. Determine if more capacity is needed to support your graph size and load. For more information about how to interpret some of these metrics, see the Operational excellence pillar section.

Cost optimization pillar

The cost optimization pillar of the AWS Well-Architected Framework focuses on avoiding unnecessary costs. The following recommendations can help you meet the cost optimization design principles and architectural best practices for Neptune Analytics.

The cost optimization pillar focuses on the following key areas:

- Understanding spending over time and controlling fund allocation
- Selecting resources of the right type and quantity
- Scaling to meet business needs without overspending

Understand usage patterns and services needed

Before you adopt Neptune Analytics, assess whether your use case is a good fit for graph analytics.

- **Graph databases**: A graph database such as Neptune is a good fit for your workload if your data model has a discernible graph structure and your queries need to explore relationships and traverse multiple hops. A graph database isn't a good fit for the following patterns:
  - Mainly single-hop queries. In this use case, consider whether your data might be better represented as attributes of an object.
  - JSON or binary large object (blob) data stored as properties.

- **Graph analytics**: Neptune Analytics is a graph analytics database engine that can quickly analyze large amounts of graph data in memory to get insights and find trends. You can store and query graph data in both a Neptune database and a Neptune Analytics graph. A Neptune database is best suited for scalable online transactional processing (OLTP) needs. Neptune Analytics is best for ephemeral analytics workloads. You can use the two in combination by loading data from your transaction-oriented Neptune database to a Neptune Analytics graph to run analytics of that data. When analysis is complete, you can remove the Neptune Analytics graph. For a more detailed comparison, see When to use Neptune Analytics and when to use Neptune Database in the Neptune Analytics documentation.

Determine, with attention to cost, how best to populate your Neptune Analytics graph.

- Bulk-import graph data that's staged in an S3 bucket. We recommend this option if your data was previously staged for bulk load to a Neptune database, or if you already have, or can readily produce, the data to be analyzed in CSV or other supported formats that bulk import requires. You can run the bulk import as part of the graph creation procedure. You can place bounds on minimum and maximum capacity. You can also run the import on a previously created empty graph and monitor the import task while it runs.
- You can create an empty graph and then populate it through an openCypher query by using batch load. This option is ideal if the data to be loaded is staged in Amazon S3 and is smaller than 50 GB.
- You can populate the graph from data in your Neptune database cluster (supported in Neptune Database version 1.3.0 or later). The intent of this pattern is to run analytics on data that's *currently* in your graph database. Even if the database was initially populated through bulk load, it might have changed significantly since then. To import from the database, Neptune Analytics clones your database and exports data from the clone to an S3 bucket. This procedure incurs costs: notably Neptune database costs for running the clone and Amazon S3 costs for storing and consuming the exported data. The clone is removed when the export is complete. You can delete the exported data in Amazon S3.
- You can populate the graph from the snapshot of a Neptune database cluster. This is similar to the previous option, except that the source is a database snapshot. To import from a snapshot, Neptune Analytics first restores the snapshot to a new database cluster, and then exports the data to an S3 bucket. This procedure incurs costs: notably Neptune database costs for running the restored cluster and Amazon S3 costs for storing and consuming the exported data.
- You can also perform openCypher queries to create, update, or delete data by using atomicity, consistency, isolation, durability (ACID) compliant transactions on the graph. We recommend this approach as a way to make small updates but not as a way of seeding the graph.

If the data needed for analytics is already staged in Amazon S3, we recommend bulk import or batch load. These are more cost-effective than populating the graph from a Neptune database cluster or snapshot.

Select resources with attention to cost

Neptune Analytics pricing uses a unit known as memory-optimized Neptune Capacity Unit (m-NCU). There is a fixed hourly cost for running a graph with a given m-NCU. A graph might have replicas for failover, and these replicas also incur hourly m-NCU cost.

We recommend the following best practices to estimate capacity, to limit costs, and to monitor costs against performance:

- If possible, create the graph by importing data from an existing source: data staged in Amazon S3 or an existing Neptune cluster or snapshot. This saves you effort because Neptune Analytics performs the heavy lifting of seeding the graph, and you can specify a bound maximum capacity.
- You can change provisioned capacity on an existing graph.
- When the graph is no longer needed, you can create a snapshot and delete the graph. If you need to use it again, you can restore the graph from the snapshot.
- You can choose the number of replicas when you create the graph. Set the value according to your analytics availability requirement. Save costs by minimizing this setting. The maximum value of 2 allows two replica instances in separate Availability Zones. The minimum value of 0 means that Neptune Analytics will not run a replica. However, recovery is faster when a replica is available. For an explanation of graph failure and recovery, see the Reliability pillar section.
- Monitor Neptune Analytics expenses for current and past billing periods by using AWS Billing and Cost Management.
- Monitor Neptune Analytics metrics for CloudWatch, especially `NumQueuedRequestsPerSec`, `NumOpenCypherRequestsPerSec`, `GraphStorageUsagePercent`, `GraphSizeBytes`, and `CPUUtlization`, to assess whether the provisioned capacity is appropriately sized for the graph. Determine if a smaller capacity can accommodate the observed request rate, CPU usage, and graph size.
- If you require a private endpoint for your graph, pay attention to costs for elastic IPs, virtual private cloud (VPC) endpoints, NAT gateways, or other VPC-related costs. For more, see Amazon VPC pricing and Amazon EC2 pricing.
- You might want to run one or more Neptune notebook instances to provide a client interface to help developers and analysts query and visualize the graph (see Neptune workbench pricing). To minimize costs, share the instance among users and create separate notebook folders for each user. Shut down the instance when it isn't in use. For an approach to automate the shutdown, see the AWS blog post Automate the stopping and starting of Amazon Neptune environment resources using resource tags.

Sustainability pillar

The sustainability pillar of the AWS Well-Architected Framework focuses on minimizing the environmental impacts of running cloud workloads. Key topics include a shared responsibility model for sustainability, understanding impact, and maximizing use to minimize required resources and reduce downstream impacts.

The sustainability pillar contains the following key focus areas:

- Your impact
- Sustainability goals
- Maximized usage
- Anticipating and adopting new, more efficient software offerings
- Use of managed services
- Downstream impact reduction

This guide focuses on understanding your impact. For more information about the other sustainability design principles, see the AWS Well-Architected Framework.

Your choices and requirements have an impact on the environment. If you can choose AWS Regions that have lower carbon intensity, and if your requirements reflect actual workload needs instead of only maximizing uptime and durability, the sustainability of the workload increases. The next sections discuss best practices and considerations that will have a positive environmental impact if adopted in your workload design and ongoing operations

Consider your AWS Region selection

Some AWS Regions are near Amazon renewable energy projects or located where the grid has a published carbon intensity that is lower than others. Consider the sustainability impact for Regions that might be viable for your workload, and cross-reference your list with the Regions where Neptune Analytics is available.

Optimize consumption

Minimize the consumption of Neptune Analytics by practicing the following:

- Analytics is often ephemeral. The graph is required only for the time to run algorithms and record the results. If this is the case, snapshot and delete the graph when it is no longer needed. You can restore it from a snapshot later if necessary.
- If the workload is ephemeral and you have the flexibility to decide when to run the analytics, consider day-to-day trends in power consumption. Demand for electricity is higher during certain times. If you're in the United States, see the metrics on daily electricity consumption on the U.S. Energy Information Administration (EIA) website. Run workloads during off-peak periods for your Region if possible.
- If the workload is not ephemeral but needs to be available only for limited periods, delete the graph and restore it from a snapshot when it is needed. If its availability follows a schedule, automate the restoration process through scripts so that the graph is ready at the scheduled time.
- If the data is read-only or has not changed since the last snapshot, do not snapshot it again before deletion.
- Stop Neptune notebooks when they are not in use.
- Monitor CloudWatch metrics such as `NumQueuedRequestsPerSec`, `NumOpenCypherRequestsPerSec`, `GraphStorageUsagePercent`, `GraphSizeBytes`, and `CPUUtilization` to assess whether the graph is oversized. Determine if a smaller instance capacity can accommodate the observed request rate, CPU usage, and graph size.

Optimize software development and architecture patterns

To prevent waste, optimize your models and queries, and share compute resources so that you use all the resources available in Neptune instances and clusters. Specific best practices include:

- Optimize queries and graph algorithm invocations. Use parameterized queries and use the query plan cache, which is enabled by default. For slow queries, run an explain plan to make improvements. If you use vector similarity search, decide if smaller embeddings yield accurate similarity results, because smaller embeddings can be created, stored, and searched more efficiently. Before you call a graph algorithm, use a `MATCH` clause to minimize the input node set. Filter on node and edge labels if possible.
- Seek the most efficient way to load data into the graph. If you load from data in Amazon S3, use bulk import if the data is greater than 50 GB in size. Use batch load for smaller data.

- Ask developers to share Neptune notebook instances instead of each creating their own instance. Create separate notebook folders for each developer on a single Jupyter instance. Shut down the instance when it is not in use.

Resources

References

- AWS Well-Architected
- AWS Well-Architected Framework documentation
- Applying the AWS Well-Architected Framework for Amazon Neptune
- Neptune Analytics best practices

Blog posts and videos

- Neptune blog posts (AWS Database Blog)
- Automate the stopping and starting of Amazon Neptune environment resources using resource tags (AWS Database Blog)
- Amazon Neptune snackables (short YouTube videos)

Training

- Getting Started with Amazon Neptune
- Build with Amazon Neptune
- Data Modeling for Amazon Neptune
- Amazon Neptune Analytics Workshop

Document history

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

Change	Description	Date
Initial publication	—	December 20, 2024

AWS Prescriptive Guidance glossary

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

Numbers

7 Rs

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

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

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

A

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, 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.