AWS Prescriptive Guidance

Defining S3 bucket and path names for data lake layers on the AWS Cloud

AWS Prescriptive Guidance: Defining S3 bucket and path names for data lake layers on the AWS Cloud

Table of Contents

Defining S3 bucket and path names for data lake layers on the AWS Cloud

Isabelle Imacseng, Professional Services Consultant, AWS Professional Services

Samuel Schmidt, Principal Data Lake Architect, AWS Professional Services

Andrés Cantor, Senior Data Architect, AWS Professional Services

November 2021

This guide helps you create a consistent naming standard for Amazon Simple Storage Service (Amazon S3) buckets and paths in data lakes hosted on the Amazon Web Services (AWS) Cloud. The guide's naming standard for S3 buckets and paths helps you to improve governance and observability in your data lakes, identify costs by data layer and AWS account, and provides an approach for naming AWS Identity and Access Management (IAM) roles and policies.

We recommend that you use at least three data layers in your data lakes and that each layer uses a separate S3 bucket. However, some use cases might require an additional S3 bucket and data layer, depending on the data types that you generate and store. For example, if you store sensitive data, we recommend that you use a landing zone data layer and a separate S3 bucket. The following list describes the three recommended data layers for your data lake:

- **Raw data layer** – Contains raw data and is the layer in which data is initially ingested. If possible, we recommend that you retain the original file format and turn on versioning in the S3 bucket.
- **Stage data layer** – Contains intermediate, processed data that is optimized for consumption (for example CSV to Apache Parquet converted raw files or data transformations). An AWS Glue job reads the files from the raw layer and validates the data. The AWS Glue job then stores the data in an Apache Parquet-formatted file and the metadata is stored in a table in the AWS Glue Data Catalog.
- **Analytics data layer** – Contains the aggregated data for your specific use cases in a consumption-ready format (for example, Apache Parquet).

This guide's recommendations are based on the authors' experience in implementing data lakes with the serverless data lake framework (SDLF) and are intended for data architects, data engineers, or solutions architects who want to set up a data lake on the AWS Cloud. However, you must make sure that you adapt this guide's approach to meet your organization's policies and requirements.

The guide contains the following sections:

Targeted business outcomes

You should expect the following five outcomes after implementing a naming standard for S3 buckets and paths in data lakes on the AWS Cloud:

- Improved governance and observability in your data lake.
- Increased visibility into your overall costs for individual AWS accounts by using the relevant AWS account ID in the S3 bucket name and for data layers by using cost allocation tags for the S3 buckets.
- More cost-effective data storage by using layer-based versioning and path-based lifecycle policies.
- Meet security requirements for data masking and data encryption.
- Simplify data source tracing by enhancing developer visibility to the AWS Region and AWS account of the underlying data storage.

Recommended data layers

If you work with non-sensitive data, such as non-personally identifiable information (PII) data, we recommend that you use at least three different data layers in a data lake on the AWS Cloud.

However, you might require additional layers depending on the data's complexity and use cases. For example, if you work with sensitive data (for example, PII data), we recommend that you use an additional Amazon Simple Storage Service (Amazon S3) bucket as a landing zone and then mask the data before it is moved into the raw data layer. For more information about this, see the Handling sensitive data (p. 9) section of this guide.

Each data layer must have an individual S3 bucket; the following table describes our recommended data layers:

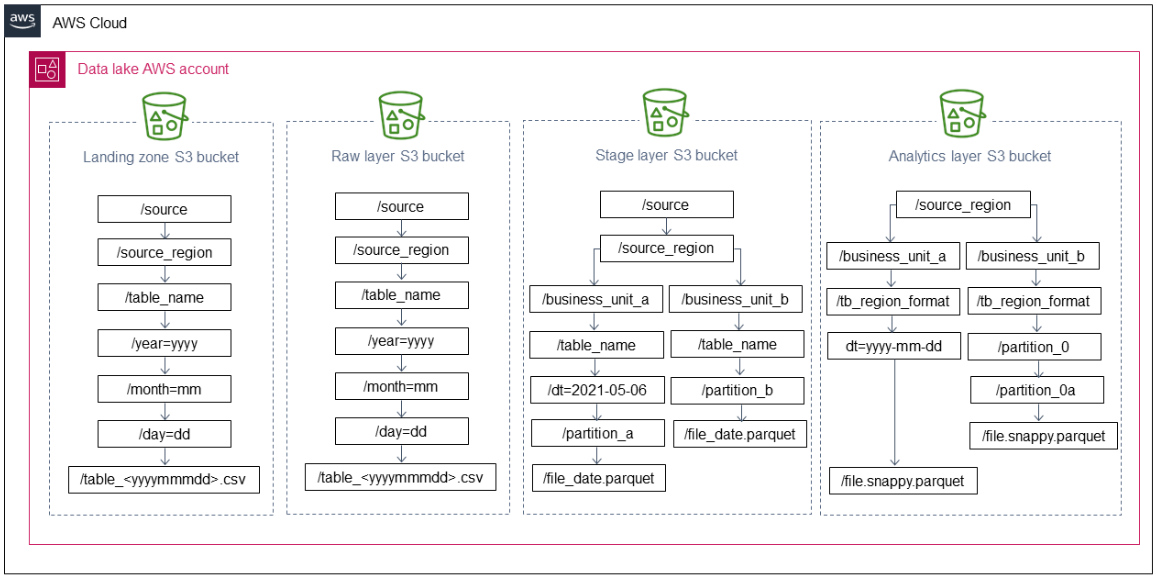
Data layer name	Description	Sample lifecycle policy strategy
Raw	Contains the raw, unprocessed data and is the layer in which data is ingested into the data lake. If possible, you should keep the original file format and turn on versioning in the S3 bucket.	After one year, move files into the Amazon S3 infrequent access (IA) storage class. After two years in Amazon S3 IA, archive them to Amazon S3 Glacier.
Stage	Contains intermediate, processed data that is optimized for consumption (for example CSV to Apache Parquet converted raw files or data transformations). An AWS Glue job reads the files from the raw layer and validates the data. The AWS Glue job then stores the data in an Apache Parquet-formatted file and the metadata is stored in a table in the AWS Glue Data Catalog.	Data can be deleted after a defined time period or according to your organization's requirements. Some data derivatives (for example, an Apache Avro transform of an original JSON format) can be removed from the data lake after a shorter amount of time (for example, after 90 days).
Analytics	Contains the aggregated data for your specific use cases in a consumption-ready format (for example, Apache Parquet).	Data can be moved to Amazon S3 IA and then deleted after a defined time period or according to your organization's requirements.

Note
You must evaluate all the recommended lifecycle policy strategies against your organizational needs, regulatory requirements, query patterns, and cost considerations.

Naming S3 buckets in your data layers

The following sections provide naming structures for Amazon Simple Storage Service (Amazon S3) buckets in your data lake layers. However, you can customize the S3 bucket and path names according to your organization's requirements. We recommend that you create separate S3 buckets for each individual layer because archiving, versioning, access, and encryption requirements can vary for each layer.

The following diagram shows the recommended naming structure for S3 buckets in the three recommended data lake layers, including separating multiple business units, file formats, and partitions. You can adapt data partitions according to your organization's requirements, but you should use lowercase and key-value pairs (For example, `year=yyyy`, not `yyyy`) so that you can update the catalog with the `MSCK REPAIR TABLE` command.



Important
S3 buckets must follow the naming guidelines from Bucket naming rules in the Amazon S3 documentation.

Landing zone S3 bucket

You require an Amazon Simple Storage Service (Amazon S3) bucket for your landing zone if sensitive datasets contain elements that must be masked before data is moved to the raw bucket.

The following table provides the naming structure, a description of the naming structure, and a name example for the S3 bucket in your landing zone layer.

Naming format	Example

s3://companyname-landingzone-awsregion-awsaccount|uniqid-env/source/source_region/table/year=yyyy/month=mm/day=dd/table_<yearmonthday>.avro|csv • companyname – The organization’s name (optional). • awsregion – The AWS Region (for example, us-east-1, or sa-east-1). • awsaccount|uniqid – The unique identifier or AWS account ID. • env – The deployment environment (for example, dev, test, or prod). • source – The source or content (for example, MySQL database, ecommerce, or SAP). • source_region – For example, us or asia. • table – tb_customer, tb_transactions, or tb_products.	s3://anycompany-landingzone-useast1-12345-dev/socialmedia/us/tb_products/year=2021/month=03/day=01/products_20210301.csv

Raw layer S3 bucket

The raw data layer contains ingested data that has not been transformed and is in its original file format (for example, JSON or CSV). This data is typically organized by data source and the date that it was ingested into the raw data layer's Amazon Simple Storage Service (Amazon S3) bucket.

The following table provides the naming structure, a description of the naming structure, and a name example for the S3 bucket in your raw data layer.

Naming format	**Example**
s3://companyname-raw-awsregion-awsaccount|uniqid-env/source/source_region/table/year=yyyy/month=mm/day=dd/table_<yearmonthday>.avro|csv • companyname – The organization’s name (optional). • awsregion – The AWS Region (for example, us-east-1, or sa-east-1). • awsaccount|uniqid – The unique identifier or AWS account ID. • env – The deployment environment (for example, dev, test, or prod). • source – The source or content (for example, MySQL database, ecommerce, or SAP). • source_region – For example, us or asia. • table – tb_customer, tb_transactions, or tb_products.	s3://anycompany-raw-useast1-12345-dev/socialmedia/us/tb_products/year=2021/month=03/day=01/products_20210301.csv

Stage layer S3 bucket

Data in the stage layer is read and transformed from the raw layer (for example, by using an AWS Glue or Amazon EMR job). This process validates the data (for example, by checking data types and headers) and then stores it in a consumption-ready file format such as Apache Parquet. The metadata is stored in a table in the AWS Glue Data Catalog.

The following table provides the naming structure, a description of the naming structure, and a name example for the S3 bucket in your stage data layer.

Naming format	Example
`s3://companyname-stage-awsregion-awsaccount|uniqid-env/source/source_region/business_unit/table/<partitions>/table_<table_name>_<yearmonthday>.snappy.parquet` • `companyname` – The organization's name (optional). • `awsregion` – The AWS Region (for example, `us-east-1`, or `sa-east-1`). • `awsaccount|uniqid` – The unique identifier or AWS account ID. • `env` – The deployment environment (for example, `dev`, `test`, or `prod`). • `source` – The source or content (for example, MySQL database, ecommerce, or SAP). • `source_region` – For example, `us` or `asia`. • `business_unit` – The business unit that the data is processed for. • `table` – `tb_customer`, `tb_transactions`, or `tb_products`. • `partitions` – Partitions that provide the best performance for the consumer, allowing the query engine to avoid full data scans.	`s3://anycompany-stage-saeast1-12345-dev/sap/br/customers/validated/dt=2021-03-01/table_customers_20210301.snappy.parquet`

Analytics layer S3 bucket

The analytics layer is similar to the stage layer because the data is in a processed file format, but the data is then aggregated according to your organization's requirements.

The following table provides the naming structure, a description of the naming structure, and a name example for the S3 bucket in your analytics data layer.

Naming format	Example
`s3://companyname-analytics-awsregion-awsaccount|uniqid-env/source_region/business_unit/`	`s3://anycompany-analytics-useast1-12345-dev/us/sales/`

`tb_<region>_<table_name>_<file_format>/<partition_0>/<partition_1>/.../<partition_n>/xxxxx.<compression>.<file_format>` • `companyname` – The organization's name (optional). • `awsregion` – The AWS Region (for example, `us-east-1`, or `sa-east-1`). • `awsaccount|uniqid` – The unique identifier or AWS account ID. • `env` – The deployment environment (for example, `dev`, `test`, or `prod`). • `source` – The source or content (for example, MySQL database, ecommerce, or SAP). • `source_region` – For example, `us` or `asia`. • `business_unit` – The business unit that the data is processed for. • `table` – `tb_customer`, `tb_transactions`, or `tb_products`. • `partitions` – Partitions that provide the best performance for the consumer, allowing the query engine to avoid full data scans.	`tb_us_customers_parquet/<partitions>/part-000001-20218c886790.c000.snappy.parquet`

Mapping S3 buckets to IAM policies in your data lake

We recommend that you map the data lake's Amazon Simple Storage Service (Amazon S3) buckets and paths to AWS Identity and Access Management (IAM) policies and roles by using the bucket names or paths in the IAM policy or role name. The following table shows a sample S3 bucket name and a sample IAM policy that is used to access this S3 bucket.

Sample Amazon S3 object path	Sample IAM policy

S3 bucket name – `<companyname>-raw-<aws_region>-<aws_accountid>-dev`
 	
S3 bucket path – `nosql/us/customers/year=2020/month=03/day=01/table_customers_20210301.csv`
	`{`
	`“Version” : “2012-10-17",`
	`“Statement” : [`
	`{`
	`“Sid” : “s3-nosql-us-customers-get-list",`
	`“Effect” : “Allow”,`
	`“Principal” : “*”,`
	`“Action” : [`
	`“s3:GetObject”,`
	`“s3:ListBucket”`
	`],`
	`“Resource” : [`
	`“arn:aws:s3:::<companyname>-raw-<aws_region>-<aws_accountid>-dev/*”`
	`]`
	`}`
	`]`
	`}`

Note
This is a sample IAM policy that shows our recommended naming standard for S3 buckets; however, you should ensure that you correctly configure S3 bucket policies according to your organization's policies and requirements.

Handling sensitive data

Typically, sensitive data contains PII or confidential information that must be secured for compliance or legal reasons. If encryption is only required on a row or column level, we recommend that you use a landing zone layer. This is *partially-sensitive* data.

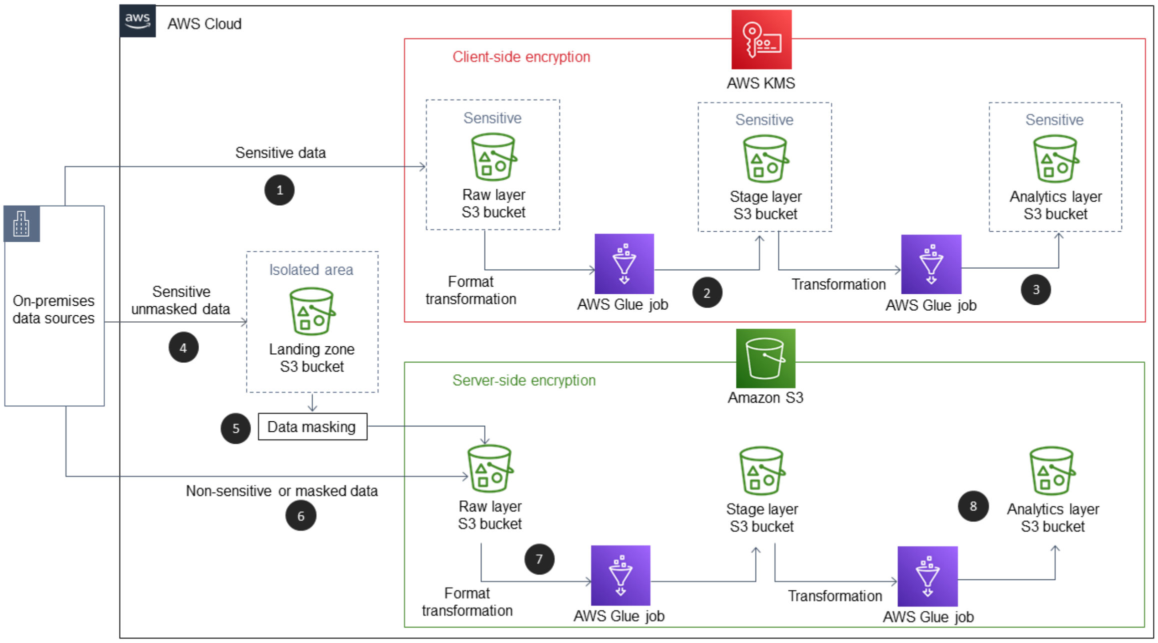
However, if the entire dataset is considered sensitive, we recommend using separate Amazon Simple Storage Service (Amazon S3) buckets to contain the data. This is *highly-sensitive* data. These separate S3 buckets must be used for each data layer and "*sensitive*" should be included in the bucket's name. We recommend that you encrypt sensitive buckets with AWS Key Management Service (AWS KMS) using Client-Side Encryption. You must also use client-side encryption to encrypt the AWS Glue jobs that transform your data.

Using a landing zone to mask sensitive data

You can use a landing zone layer for partially-sensitive datasets (for example, if encryption is only required at the row or column level). This data is ingested into the landing zone's S3 bucket and is then masked. After the data is masked, it is ingested into the raw layer's S3 bucket that is encrypted with Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3). If required, you can tag data at the object level.

Any data that is already masked can bypass the landing zone and be directly ingested into the raw layer's S3 bucket. There are two access levels in the stage and analytics layers for partially-sensitive datasets; one level has full access to all data and the other level only has access to non-sensitive rows and columns.

The following diagram shows a data lake where partially-sensitive datasets use a landing zone to mask the sensitive data but highly-sensitive datasets use separate, encrypted S3 buckets. The landing zone is isolated using restrictive IAM and S3 bucket policies, and the encrypted buckets use client-side encryption with AWS KMS.



The diagram shows the following workflow:

1. Highly-sensitive data is sent to an encrypted S3 bucket in the raw data layer.
2. An AWS Glue job validates and transforms the data into a consumption-ready format and then places file into an encrypted S3 bucket in the stage layer.
3. An AWS Glue job aggregates data according to business requirements and places the data into an encrypted S3 bucket in the analytics layer.
4. Partially-sensitive data is sent to landing zone bucket.
5. Sensitive rows and columns are masked and data is then sent to the S3 bucket in the raw layer.
6. Non-sensitive data is directly sent to the S3 bucket in the raw layer.
7. An AWS Glue job validates and transforms the data into a consumption-ready format and places the files into the S3 bucket for the stage layer.
8. An AWS Glue job aggregates the data according to your organization's requirements and places the data into an S3 bucket in the analytics layer.

FAQ

This section provides answers to commonly raised questions about defining S3 bucket and path names for data lake layers on the AWS Cloud.

What name should I use for a multi-Region Amazon Simple Storage Service (Amazon S3) bucket?

You can use our recommended S3 bucket naming format and change the AWS Region identifier. For example, `examplecompany-raw-`**`useast1`**`-12345-dev` and `examplecoompany-raw-`**`uswest1`**`-12345-dev`.

Do I need to use raw, stage, and analytics as the names for my data lake layers?

No, you can name your layers according to your requirements. However, we strongly recommended that you use an S3 bucket for the data layer that contains the original file formats and that has versioning enabled.

Is it possible to rename an S3 bucket?

No. If you want to use a different S3 bucket name, you must create a new bucket with the new name. This one reason why we recommend having a clearly defined and consistent naming approach for S3 buckets.

What happens if I delete an S3 bucket and want to reuse the name?

If you delete an S3 bucket and want to create a new bucket with the same name, you must wait several minutes for the name to become available again. S3 bucket names are globally unique and all AWS accounts share the same namespace.

Are there limitations on what I can include in my S3 bucket or path's name?

Only lowercase letters, numbers, dashes, and dots are allowed in S3 bucket names. Bucket names must be three to 63 characters in length, must begin and end with a number or letter, and cannot be in an IP address format. The names must also be globally unique.

For S3 bucket paths, you can use uppercase letters, but we recommend that you only use lowercase letters. Paths can also include additional symbols, but we recommend that you only use underscores, dashes, slashes, and numbers.

Can I use more layers than the landing zone, raw, stage, and analytics layers in my data lake?

Yes, you can use as many layers as you want. However, we recommend having a landing zone and raw layer for your raw data, an intermediate layer for formatted data, and a layer for highly-modeled data.

What happens if I have not defined my parameters?

Certain parameters (for example, business units) don't need to be incorporated into the S3 bucket name but can be part of the path. This means that they don't need to be immediately determined because paths can be added after an S3 bucket is created.

How can I track costs at the business unit level?

This depends on your account strategy. If you have business units split up into different AWS accounts, you can assign cost allocation tags to S3 buckets that reflect the bucket costs for each business unit.

If your account strategy doesn't separate out business units into different AWS accounts, then you can use different buckets for each business unit by adding the business unit to the bucket name (for example, `exampleco-businessunit1-raw-useast1-12345-dev`). However, this means that you have to manage many S3 buckets.

What features should I consider when creating an S3 bucket naming standard?

You must ensure that your S3 bucket names use features that are only available at the bucket level. For example, cost tags, bucket encryption, and versioning are features that are only available for an entire S3 bucket. This means that they apply to all objects and paths in the S3 bucket.

Object versioning is also an important feature to consider. You should turn on versioning for your raw layer's S3 buckets, because you want to make sure that you can see previous versions if there are changes to the data. However, versioning might not be necessary for all the layers in your data lake and retaining multiple versions can cause unnecessary costs.

Resources

- AWS Glue developer guide (AWS Glue documentation)
- AWS Glue components (AWS Glue documentation)
- Bucket naming rules (Amazon Simple Storage Service (Amazon S3) documentation)
- Protecting data using Server-Side Encryption with KMS keys stored in AWS Key Management Service (SSE-KMS) (Amazon S3 documentation)
- Using cost allocation S3 bucket tags (Amazon S3 documentation)
- User-defined cost allocation tags (AWS Billing and Cost Management documentation)

AWS Prescriptive Guidance glossary

AI and ML terms (p. 14) | Migration terms (p. 15) | Modernization terms (p. 19)

AI and ML terms

The following are commonly used terms in artificial intelligence (AI) and machine learning (ML)-related strategies, guides, and patterns provided by AWS Prescriptive Guidance. To suggest entries, please use the **Provide feedback** link at the end of the glossary.

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?"

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.

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.

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.

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.

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 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.

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?"

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).

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.

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.

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.

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.

Migration terms

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

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. This migration scenario is specific to VMware Cloud on AWS, which supports virtual machine (VM) compatibility and workload portability between your on-premises environment and AWS. You can use the VMware Cloud Foundation technologies from your on-premises data centers when you migrate your infrastructure to VMware Cloud on AWS. Example: Relocate the hypervisor hosting your Oracle database to VMware Cloud on 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.

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 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.

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 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.

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.

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.

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 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.

configuration management database (CMDB)

A database that contains information about a company's hardware and software products, configurations, and inter-dependencies. You typically use data from a CMDB in the portfolio discovery and analysis stage of migration.

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.

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.

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.

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.

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.

large migration

A migration of 300 or more servers.

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 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 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 CloudEndure 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 strategy

The approach used to migrate a workload to the AWS Cloud. For more information, see the 7 Rs (p. 15) entry in this glossary and see Mobilize your organization to accelerate large-scale migrations.

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).

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.

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.

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.

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.

responsible, accountable, consulted, informed (RACI) matrix

A matrix that defines and assigns roles and responsibilities in a project. For example, you can create a RACI to define security control ownership or to identify roles and responsibilities for specific tasks in a migration project.

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.

service-level agreement (SLA)

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

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.

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.

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.

Modernization terms

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

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.

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.

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.

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.

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.

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. For more information, see the Two-pizza team section of the Introduction to DevOps on AWS whitepaper.

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.

update-history-change	update-history-description	update-history-date
— (p. 21)	Initial publication	November 18, 2021