

Creating production-ready ML pipelines on AWS

AWS Prescriptive Guidance



Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Prescriptive Guidance: Creating production-ready ML pipelines on AWS

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

Introduction	1
	1
	3
	4
Using processing jobs	4
Using the main guard clause	5
Unit testing	6
	8
Using the Step Functions SDK	8
Extending the Step Functions SDK	9
	10
Implementation with AWS CloudFormation	10
Modifying the output from the Step Functions SDK	11
	12
	14
Different levels of automation	14
Different platforms for ML workloads	14
Different engines for pipeline orchestration	15
	17
Additional resources	18
Document history	19
Glossary	20
#	20
A	21
В	24
C	26
D	29
E	
F	35
G	
H	38
I	39
L	41
M	43

0	47
P	49
Q	
R	
S	
T	
U	
V	
W	
Z	

Creating production-ready ML pipelines on AWS

Josiah Davis, Verdi March, Yin Song, Baichuan Sun, Chen Wu, and Wei Yih Yap, Amazon Web Services (AWS)

January 2021 (document history)

Machine learning (ML) projects require a significant, multi-stage effort that includes modeling, implementation, and production to deliver business value and to solve real-world problems. Numerous alternatives and customization options are available at each step, and these make it increasingly challenging to prepare an ML model for production within the constraints of your resources and budget. Over the past few years at Amazon Web Services (AWS), our Data Science team has worked with different industry sectors on ML initiatives. We identified pain points shared by many AWS customers, which originate from both organizational problems and technical challenges, and we have developed an optimal approach for delivering production-ready ML solutions.

This guide is for data scientists and ML engineers who are involved in ML pipeline implementations. It describes our approach for delivering production-ready ML pipelines. The guide discusses how you can transition from running ML models interactively (during development) to deploying them as a part of a pipeline (during production) for your ML use case. For this purpose, we have also developed a set of example templates (see the ML Max project project), to accelerate the delivery of custom ML solutions to production, so you can get started quickly without having to make too many design choices.

Overview

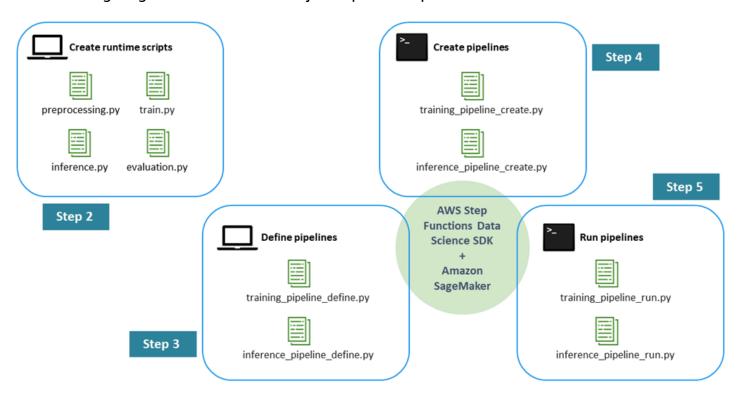
The process for creating a production-ready ML pipeline consists of the following steps:

- <u>Step 1</u>. Perform EDA and develop the initial model Data scientists make raw data available in Amazon Simple Storage Service (Amazon S3), perform exploratory data analysis (EDA), develop the initial ML model, and evaluate its inference performance. You can conduct these activities interactively through Jupyter notebooks.
- Step 2. Create the runtime scripts You integrate the model with runtime Python scripts so that it can be managed and provisioned by an ML framework (in our case, Amazon SageMaker AI). This is the first step in moving away from the interactive development of a standalone model toward production. Specifically, you define the logic for preprocessing, evaluation, training, and inference separately.

1

- <u>Step 3</u>. **Define the pipeline** You define the input and output placeholders for each step of the pipeline. Concrete values for these will be supplied later, during runtime (step 5). You focus on pipelines for training, inference, cross-validation, and back-testing.
- <u>Step 4</u>. Create the pipeline You create the underlying infrastructure, including the AWS Step Functions state machine instance in an automated (nearly one-click) fashion, by using AWS CloudFormation.
- <u>Step 5</u>. Run the pipeline You run the pipeline defined in step 4. You also prepare metadata and data or data locations to fill in concrete values for the input/output placeholders that you defined in step 3. This includes the runtime scripts defined in step 2 as well as model hyperparameters.
- Step 6. Expand the pipeline You implement continuous integration and continuous deployment (CI/CD) processes, automated retraining, scheduled inference, and similar extensions of the pipeline.

The following diagram illustrates the major steps in this process.



Steps for creating the training and inference pipeline

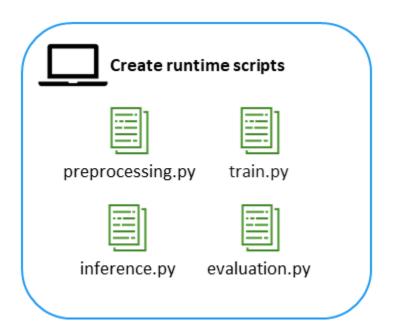
Step 1. Perform EDA and develop the initial model

In this step, data scientists perform exploratory data analysis (EDA) in order to understand the ML use case and data. They then develop the ML models (for example, classification and regression models) to solve the problem in a given use case. During model development, the data scientist often makes assumptions about inputs and outputs, such as data formats, the data lifecycle, and locations of intermediate output. These assumptions should be documented so that they can be used for verification during unit tests in step 2.

Although this step focuses on model development, data scientists often have to write a minimum amount of helper code for preprocessing, training, evaluation, and inference. The data scientist should be able to run this code in the development environment. We also recommend providing optional runtime arguments so that this helper code can be dynamically configured to run in other environments without extensive manual changes. This will accelerate the integration between the model and the pipeline in steps 2 and 3. For example, code for reading the raw data should be encapsulated in functions so that data can be preprocessed in a consistent manner.

We recommend that you start with a framework such as <u>scikit-learn</u>, <u>XGBoost</u>, <u>PyTorch</u>, <u>Keras</u>, or <u>TensorFlow</u> to develop the ML model and its helper code. For example, scikit-learn is a a free ML library that's written in Python. It provides a uniform API convention for objects, and includes four main objects—*estimator*, *predictor*, *transformer*, and *model*—that cover lightweight data transforms, support label and feature engineering, and encapsulate preprocessing and modeling steps. These objects help avoid boilerplate code proliferation and prevent validation and test data from leaking into the training dataset. Similarly, every ML framework has its own implementation of key ML artifacts, and we recommend that you comply with the API conventions of your selected framework when you develop ML models.

Step 2. Create the runtime scripts



In this step, you integrate the model that you developed in step 1 and its associated helper code into an ML platform for production-ready training and inference. Specifically, this involves the development of runtime scripts so that the model can be incorporated into SageMaker AI. These standalone Python scripts include predefined SageMaker AI callback functions and environment variables. They are run inside a SageMaker AI container that is hosted on an Amazon Elastic Compute Cloud (Amazon EC2) instance. The Amazon SDK documentation provides detailed information about how these callbacks and auxiliary setup work together for training and inference. (For example, see Prepare a scikit-learn training script and Deploy a scikit-learn model in the SageMaker AI documentation.) The following sections provide additional recommendations for developing ML runtime scripts, based on our experience working with AWS customers.

Using processing jobs

SageMaker AI provides two options for performing batch-mode model inference. You can use a SageMaker AI *processing job* or a *batch transform job*. Each option has advantages and disadvantages.

A processing job consists of a Python file that runs inside a SageMaker AI container. The processing job consists of whatever logic you put in your Python file. It has these advantages:

Using processing jobs

- When you understand the basic logic of a training job, processing jobs are straightforward to set up and easy to understand. They share the same abstractions as training jobs (for example, tuning the instance count and data distribution).
- Data scientists and ML engineers have full control over data manipulation options.
- The data scientist doesn't have to manage any I/O component logic except for familiar read/ write functionality.
- It's somewhat easier to run the files in non-SageMaker AI environments, which aids rapid development and local testing.
- If there is an error, a processing job fails as soon as the script fails, and there is no unexpected waiting for retry.

On the other hand, batch transform jobs are an extension of the concept of a SageMaker AI endpoint. At runtime, these jobs import callback functions, which then handle the I/O for reading the data, loading the model, and making the predictions. Batch transform jobs have these advantages:

- They use a data distribution abstraction that differs from the abstraction used by training jobs.
- They use the same core file and function structure for both batch inference and realtime inference, which is convenient.
- They have a built-in, retry-based fault-tolerance mechanism. For example, if an error occurs on a batch of records, it will retry multiple times before the job is terminated as a failure.

Because of its transparency, its ease of use in multiple environments, and its shared abstraction with training jobs, we decided to use the processing job instead of the batch transform job in the reference architecture presented in this guide.

You should run Python runtime scripts locally before you deploy them in the cloud. Specifically, we recommend that you use the main guard clause when you structure your Python scripts, and perform unit testing.

Using the main guard clause

Use a main guard clause to support module import and to run your Python script. Running Python scripts individually is beneficial for debugging and isolating issues in the ML pipeline. We recommend the following steps:

Using the main quard clause 5

- Use an argument parser in the Python processing files to specify input/output files and their locations.
- Provide a main guide and test functions for each Python file.
- After you test a Python file, incorporate it into the different stages of the ML pipeline, whether you're using an AWS Step Functions model or a SageMaker AI processing job.
- Use Assert statements in critical sections of the script to facilitate testing and debugging. For
 example, you can use an Assert statement to ensure that the number of dataset features is
 consistent after loading.

Unit testing

Unit testing of runtime scripts that were written for the pipeline is an important task that's frequently ignored in ML pipeline development. This is because machine learning and data science are relatively new fields and have been slow to adopt well-established software engineering practices such as unit testing. Because the ML pipeline will be used in the production environment, it is essential to test the pipeline code before applying the ML model to real-world applications.

Unit testing the runtime script also provides the following unique benefits for ML models:

- It prevents unexpected data transformations. Most ML pipelines involve many data transformations, so it is critical for these transformations to perform as expected.
- It validates the reproducibility of the code. Any randomness in the code can be detected by unit testing with different use cases.
- It enforces the modularity of the code. Unit tests are usually associated with the test coverage
 measure, which is the degree to which a particular test suite (a collection of test cases) runs
 the source code of a program. To achieve high test coverage, developers modularize the code,
 because it's difficult to write unit tests for a large amount of code without breaking it down into
 functions or classes.
- It prevents low-quality code or errors from being introduced into production.

We recommend that you use a mature unit testing framework such as <u>pytest</u> to write the unit test cases, because it's easier to manage extensive unit tests within a framework.

Unit testing 6

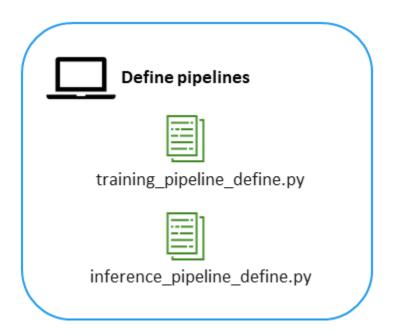


▲ Important

Unit testing cannot guarantee that all corner cases are tested, but it can help you proactively avoid mistakes before you deploy the model. We recommend that you also monitor the model after deployment, to ensure operational excellence.

Unit testing

Step 3. Define the pipeline



In this step, the sequence and logic of actions that the pipeline will perform are defined. This includes discrete steps as well as their logical inputs and outputs. For example, what is the state of the data at the beginning of the pipeline? Does it come from multiple files that are at different levels of granularity or from a single flat file? If the data comes from multiple files, do you need a single step for all files or separate steps for each file to define the preprocessing logic? The decision depends on the complexity of the data sources and the extent to which they are preprocessed.

In our reference implementation, we use <u>AWS Step Functions</u>, which is a serverless function orchestrator, to define the workflow steps. However, the <u>ML Max framework</u> also supports other pipeline or state machine systems such as Apache AirFlow (see the <u>Different engines for pipeline</u> orchestration section) to drive the development and deployment of ML pipelines.

Using the Step Functions SDK

To define the ML pipeline, we first use the high-level Python API provided by the <u>AWS Step Functions Data Science SDK</u> (the Step Functions SDK) to define two key components of the pipeline: *steps* and *data*. If you think of a pipeline as a directed acyclic graph (DAG), steps represent the nodes on the graph, and data is shown as directed edges that connect one node (step) to the next. Typical examples of ML steps include preprocessing, training, and evaluation. The Step Functions SDK provides a number of built-in steps (such as the <u>TrainingStep</u>) that you can use.

Examples of data include input, output, and many intermediate datasets that are produced by some steps in the pipeline. When you're designing an ML pipeline, you don't know the concrete values of the data items. You can define *data placeholders* that serve as a template (similar to function parameters) and contain only the name of the data item and primitive data types. In this way, you can design a complete pipeline blueprint without knowing the concrete values of data traveling on the graph in advance. For this purpose, you can use the placeholder classes in the Step Functions SDK to explicitly model these data templates.

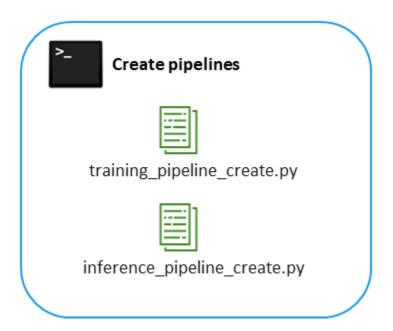
ML pipelines also require configuration parameters to perform fine-grained control over the behavior of each ML step. These special data placeholders are called *parameter placeholders*. Many of their values are unknown when you're defining the pipeline. Examples of parameter placeholders include infrastructure-related parameters that you define during pipeline design (for example, AWS Region or container image URL) and ML modeling-related parameters (such as hyperparameters) that you define when you run the pipeline.

Extending the Step Functions SDK

In our reference implementation, one requirement was to separate ML pipeline definitions from concrete ML pipeline creation and deployment by using specific parameter settings. However, some of the built-in steps in the Step Functions SDK didn't allow us to pass in all these placeholder parameters. Instead, parameter values were expected to be directly obtained during pipeline design time through SageMaker AI configuration API calls. This works fine if the SageMaker AI design-time environment is identical to the SageMaker AI runtime environment, but this is rarely the case in real-world settings. Such a tight coupling between pipeline design-time and runtime, and the assumption that the ML platform infrastructure will remain constant, significantly hinders the applicability of the designed pipeline. In fact, the ML pipeline breaks immediately when the underlying deployment platform undergoes even the slightest change.

To overcome this challenge and produce a robust ML pipeline (which we wanted to design once and run anywhere), we implemented our own custom steps by extending some of the built-in steps, including **TrainingStep**, **ModelStep**, and **TransformerStep**. These extensions are provided in the ML Max project. The interfaces of these custom steps support far more parameter placeholders that can be populated with concrete values during pipeline creation or when the pipeline is running.

Step 4. Create the pipeline



After you define the pipeline logically, it's time to create the infrastructure to support the pipeline. This step requires the following capabilities, at a minimum:

- Storage, to host and manage pipeline inputs and outputs, including code, model artifacts, and data used in training and inference runs.
- Compute (GPU or CPU), for modeling and inference as well as data preprocessing and postprocessing.
- Orchestration, to manage the resources being used and to schedule any regular runs. For example, the model might be retrained on a periodic basis as new data becomes available.
- Logging and alerting, to monitor the pipeline model accuracy, for resource utilization, and for troubleshooting.

Implementation with AWS CloudFormation

To create the pipeline we used AWS CloudFormation, which is an AWS service for deploying and managing infrastructure as code. The AWS CloudFormation templates include the Step Functions definition that was created in the previous step with the Step Functions SDK. This step includes the creation of the AWS-managed Step Functions instance, which is called the *Step Functions state machine*. No resources for training and inference are created at this stage, because training and

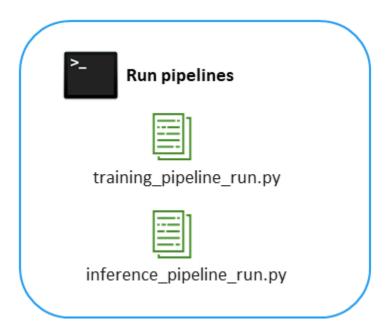
and inference jobs run on demand, only when they're needed, as SageMaker AI jobs. This step also includes creating AWS Identity and Access Management (IAM) roles to run the Step Functions, run SageMaker AI, and read and write from Amazon S3.

Modifying the output from the Step Functions SDK

We had to make some minor modifications to the AWS CloudFormation output from the previous section. We used simple Python string matching to do the following:

- We added logic for creating the Parameters section of the AWS CloudFormation template. This is because we want to create two roles and define the pipeline name as a parameter along with the deployment environment. This step also covers any additional resources and roles that you might want to create, as discussed in step 6.
- We reformatted three fields to have the required ! Sub prefix and quotation marks so that they can be updated dynamically as a part of the deployment process:
 - The StateMachineName property, which names the state machine.
 - The DefinitionString property, which defines the state machine.
 - The RoleArn property, which is returned by the state machine.

Step 5. Run the pipeline



This step runs the training or inference pipeline that was created in the AWS CloudFormation stacks in step 4. The pipeline can't be run until its internal placeholder parameters have been populated with concrete values. This action of assigning values to placeholder parameters is the primary activity of step 5. Example placeholder parameters include:

- The location of input, output, and intermediate datasets
- The Amazon S3 location of the runtime scripts and other preprocessing or evaluation code that
 was developed in step 2 (for example, sm_submit_url for the training pipeline)
- The name of the AWS Region

You have to make sure that these path values point to valid data or code before you run the pipeline. For example, if you populate the placeholder parameter that represents the Amazon S3 URL of the Python runtime scripts, you must upload those scripts to that URL. The person who runs the pipeline is responsible for consistency checking and data uploading. People who define or create the pipeline do not have to worry about any of this.

Depending on the maturity of the pipeline, this step may be automated to run on a regular (weekly or monthly) basis. Automation also requires robust monitoring, which is an important area but outside the scope of this guide. For the training pipeline run, it would be appropriate to monitor evaluation metrics. For the inference pipeline, it would be appropriate to monitor the input data

distribution drift, and, if possible, collect labels periodically and measure the drift in prediction accuracy. These records from the training and inference runs should be recorded in a database for analysis at a later point in time.

Step 6. Expand the pipeline

This guide explains how you can get started building ML pipelines on AWS quickly, with concrete architecture. There are additional considerations for maturing the pipeline, such as metadata management, experiment tracking, and monitoring. These are important topics that are outside the scope of this guide. The following sections discuss another aspect of pipeline management, which is pipeline automation.

Different levels of automation

Although you can set up a training pipeline manually in the SageMaker AI console, in practice, we recommend minimizing manual touchpoints in the deployment of ML training pipelines to ensure that ML models are deployed consistently and repeatedly. Depending on your requirements and the business problems you're addressing, you can determine and implement a deployment strategy on three levels: semi-automated, fully automated, and fully managed.

- Semi-automated By default, the steps discussed in the previous section follow a semiautomated approach, because they deploy the training and inference pipeline by using AWS CloudFormation templates. This helps ensure the reproducibility of the pipeline and helps you change and update it easily.
- Fully automated A more advanced option is to use continuous integration and continuous deployment (CI/CD) to the development, staging, and production environments. Incorporating CI/CD practices to the deployment of the training pipeline can ensure that automation includes traceability as well as quality gates.
- Fully managed Ultimately, you can develop a fully managed system so that you can deploy
 a ML training pipeline with a set of simple manifests, and the system can self-configure and
 coordinate required AWS services.

In this guide, we chose to present a concrete architecture. However, there are alternative technologies you can consider. The next two sections discuss some alternative choices for the platform and the orchestration engine.

Different platforms for ML workloads

Amazon SageMaker AI is the AWS managed service for training and serving ML models. Many users appreciate its wide array of built-in features and the many options it offers for running ML

Different levels of automation

workloads. SageMaker AI is particularly useful if you are just getting started with implementing ML in the cloud. The key features of SageMaker AI include:

- Built-in traceability (including labeling, training, model tracking, optimization, and inference).
- Built-in one-click options for training and inference with minimal Python and ML experience.
- Advanced hyperparameter tuning.
- Support for all major artificial intelligence and machine learning (ML/AI) frameworks and custom Docker containers.
- Built-in monitoring capabilities.
- Built-in tracking of histories, including training jobs, processing jobs, batch transform jobs, models, endpoints, and searchability. Some histories, such as training, processing, and batch transform, are immutable and append-only.

One of the alternatives to using SageMaker AI is <u>AWS Batch</u>. AWS Batch provides a lower level of control over the compute and orchestration for your environment, but it isn't custom-built for machine learning. Some of its key features include:

- Out-of-the-box automatic scaling of compute resources based on workload.
- Out-of-the-box support for job priority, retries, and job dependencies.
- Queue-based approach that supports building recurrent and on-demand jobs.
- Support for CPU and GPU workloads. The ability to use GPU for building ML models is critical, because GPU can speed up the training process significantly, especially for deep learning models.
- Ability to define a custom Amazon Machine Image (AMI) for the compute environment.

Different engines for pipeline orchestration

The second main component is the pipeline orchestration layer. AWS provides <u>Step Functions</u> for a fully managed orchestration experience. A popular alternative to Step Functions is Apache Airflow. When making a decision between the two, consider the following:

- Required infrastructure AWS Step Functions is a fully managed service and is serverless, whereas Airflow requires managing your own infrastructure and is based on open-source software. As a result, Step Functions provides high availability out of the box, whereas administering Apache Airflow requires additional steps.
- Scheduling capabilities Both Step Functions and Airflow provide comparable functionalities.

- Visualization capabilities and UI Both Step Functions and Airflow provide comparable functionalities.
- Passing variables within the computational graph Step Functions provides limited functionality for using AWS Lambda functions, whereas Airflow provides XCom interfaces.
- Usage Step Functions is very popular among AWS customers, and Airflow has been widely adopted by the data engineering community.

Conclusion

As machine learning transitions from a research discipline to an applied field, we've seen a yearly growth of 25 percent in ML pipeline development, deployment, and operation in various industries. The business value of ML is realized through day-to-day ML operations and pipelines, which, in turn, drive the research and development of ML models and algorithms. Nonetheless, deploying ML in production presents numerous challenges, because it interweaves significantly different activities and artifacts, such as data management, processing, analysis, modeling, verification, and security. Through numerous AI/ML engagements with AWS customers, our Data Science team has observed that a key challenge is the lack of an end-to-end workflow that would provide a set of templates for optimally fusing or separating different ML DevOps activities and artifacts. In this guide, we presented the ML Max workflow to address this pressing issue. ML Max provides step-by-step guidelines and a set of programming templates. The goal is to enable a fast and cost-effective transition from an interactive model development phase to a complete, scalable ML pipeline configuration that is ready for production.

Additional resources

Related guides and patterns

- Quantifying uncertainty in deep learning systems
- Migrate ML Build, Train, and Deploy workloads to Amazon SageMaker AI using AWS Developer Tools
- AWS Prescriptive Guidance website

AWS services

- AWS Batch
- AWS CloudFormation
- IAM
- Amazon SageMaker Al
- AWS Step Functions

AWS general resources

- AWS documentation
- AWS general reference
- AWS glossary

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	_	January 29, 2021

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 onpremises 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.

ΑI

See artificial intelligence.

AIOps

See artificial intelligence operations.

A 21

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 <u>operations</u> 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.

A 22

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 <u>ABAC for AWS</u> 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.

A 23

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.

B 24

botnet

Networks of <u>bots</u> that are infected by <u>malware</u> 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 <u>About branches</u> (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 <u>Implement break-glass procedures</u> 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 <u>Organized around business capabilities</u> section of the <u>Running containerized microservices on AWS</u> 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.

B 25



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 <u>AWS Fault Injection Service (AWS FIS)</u> 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.

C 26

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 <u>Building your Cloud Operating Model</u>.

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 <u>The Journey Toward Cloud-First</u> & 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.

C 27

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 <u>AI</u> 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 Deployment.

C 28

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 <u>star schema</u>, 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 <u>disaster</u>. For more information, see <u>Disaster Recovery of Workloads on AWS: Recovery in the Cloud</u> 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 <u>cloud computing</u>, 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 <u>service endpoint</u>.

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

E 33

information, see <u>Create an endpoint service</u> 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, <u>MES</u>, 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.

E 34

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 <u>star schema</u>. 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.

F 35

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 <u>LLM</u> 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 <u>change data</u> <u>capture</u> to migrate data in the shortest time possible, instead of using a phased approach. The objective is to keep downtime to a minimum.

FΜ

See <u>foundation model</u>.

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.

F 36

G

generative Al

A subset of <u>AI</u> 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 <u>Restricting the geographic distribution of your content</u> 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 <u>trunk-based workflow</u> 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.

G 37

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.

Н

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 rearchitecting effort, and converting the schema can be a complex task. <u>AWS provides AWS SCT</u> 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.

H 38

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.

lloT

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 <u>mutable infrastructure</u>. For more information, see the <u>Deploy using immutable infrastructure</u> best practice in the AWS Well-Architected Framework.

39

inbound (ingress) VPC

In an AWS multi-account architecture, a VPC that accepts, inspects, and routes network connections from outside an application. The <u>AWS Security Reference Architecture</u> 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 <u>Klaus Schwab</u> 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 <u>Building an industrial Internet of Things</u> (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.

I 40

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.

41

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 <u>AI</u> 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 <u>What are LLMs</u>.

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.

L 42

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 <u>Migration Acceleration Program</u>.

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.

 $\overline{\mathsf{M}}$

MES

See manufacturing execution system.

Message Queuing Telemetry Transport (MQTT)

A lightweight, machine-to-machine (M2M) communication protocol, based on the <u>publish/</u> 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 <u>AWS migration strategy</u>.

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,

M 44

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 <u>discussion of migration</u> 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 <u>migration readiness guide</u>. MRA is the first phase of the <u>AWS migration strategy</u>.

migration strategy

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

ML

See machine learning.

 $\overline{\mathsf{M}}$ 45

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.

M 46



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).

O 47

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 <u>Industry 4.0</u> transformations.

operations integration (OI)

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

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.

O 48

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 <u>AWS Security Reference Architecture</u> 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 <u>Permissions boundaries</u> 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.

P 49

PLM

See product lifecycle management.

policy

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

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 <u>Evaluating migration readiness</u>.

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 <u>Preventative controls</u> 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.

P 50

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 <u>security control</u> 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 <u>Controls reference guide</u> in the AWS Control Tower documentation and see <u>Proactive controls</u> in <u>Implementing security controls on AWS</u>.

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 <u>LLM</u> 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.

P 51

publish/subscribe (pub/sub)

A pattern that enables asynchronous communications among microservices to improve scalability and responsiveness. For example, in a microservices-based <u>MES</u>, 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.

Q 52

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.

R 53

repurchase

See 7 Rs.

resiliency

An application's ability to resist or recover from disruptions. <u>High availability</u> and <u>disaster</u> 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 <u>Responsive controls</u> in *Implementing security controls on AWS*.

retain

See 7 Rs.

retire

See 7 Rs.

Retrieval Augmented Generation (RAG)

A <u>generative AI</u> technology in which an <u>LLM</u> 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 <u>What is RAG</u>.

R 54

rotation

The process of periodically updating a <u>secret</u> 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 <u>detective</u> or <u>responsive</u> 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 <u>Service control policies</u> 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 <u>AWS service endpoints</u> 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 <u>service-level indicator</u>. 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 <u>data warehouse</u> 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 <u>introduced by Martin Fowler</u> as a way to manage risk when rewriting monolithic systems. For an example of how to apply this pattern, see <u>Modernizing legacy Microsoft ASP.NET (ASMX) web services incrementally by using containers and Amazon API Gateway</u>.

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 <u>LLM</u> to direct its behavior. System prompts help set context and establish rules for interactions with users.

Т

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 <u>Tagging</u> 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 <u>What is a transit gateway</u> 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.

T 59

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 <u>Using AWS Organizations with other AWS services</u> 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 <u>Quantifying uncertainty in</u> 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.

U 60



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.

V 61

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 <u>immutable</u>.

Z

zero-day exploit

An attack, typically malware, that takes advantage of a <u>zero-day vulnerability</u>.

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 <u>LLM</u> 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 <u>few-shot prompting</u>.

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.

Z 62