



Evaluating your ML project with the MLOps checklist

# AWS Prescriptive Guidance



# AWS Prescriptive Guidance: Evaluating your ML project with the MLOps checklist

Copyright © 2024 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

<b>Introduction .....</b>	<b>1</b>
<b>MLOps checklist components .....</b>	<b>2</b>
1. Data-centric management .....	2
2. Experimentation .....	4
3. Observability and model management .....	6
4. Robust pipelines and promotion .....	8
5. Continuous integration .....	10
6. Continuous monitoring .....	13
7. Continuous deployment .....	16
8. Continuous training .....	18
9. Governance .....	19
<b>Example usage .....</b>	<b>21</b>
<b>Resources .....</b>	<b>22</b>
<b>Document history .....</b>	<b>23</b>
<b>Glossary .....</b>	<b>24</b>
# .....	24
A .....	25
B .....	28
C .....	30
D .....	33
E .....	37
F .....	39
G .....	40
H .....	41
I .....	42
L .....	44
M .....	45
O .....	49
P .....	52
Q .....	54
R .....	55
S .....	57
T .....	61
U .....	62

V ..... 63

W ..... 63

Z ..... 64

# Evaluating your ML project with the MLOps checklist

*Charles Frenzel, Sharath Nagaraja, and Spencer Romo Amazon Web Services (AWS)*

July 2023 ([document history](#))

The [MLOps checklist](#) is a workable checklist that you can use at any phase in your machine learning (ML) project. The checklist is a tool for assessing overall readiness, examining system coverage, and identifying new areas of opportunity in distributed ML systems. MLOps is the combination of people, technology, and processes for delivering ML solutions. Well-architected MLOps helps businesses to deploy ML models to production effectively and consistently, and can deliver business value.

Using the MLOps checklist helps you to do the following:

- Assess your MLOps system.
- Find areas of opportunity.
- Find areas for improvement.
- Evaluate and update your strategic roadmap on AWS.
- Generate backlog items.

We recommend using the MLOps checklist at the start of your MLOps project, but it's possible to use parts of it during any phase.

# MLOps checklist components

The [MLOps checklist](#) consists of nine different components. Each component represents an aspect of a mature MLOps system. This section covers the checklist components in detail. Note that there is some overlap in this section because certain components depend on other components.

The checklist covers the following areas:

- [1. Data-centric management](#)
- [2. Experimentation](#)
- [3. Observability and model management](#)
- [4. Robust pipelines and promotion](#)
- [5. Continuous integration](#)
- [6. Continuous monitoring](#)
- [7. Continuous deployment](#)
- [8. Continuous training](#)
- [9. Governance](#)

**Note:** Security is already well-covered by AWS. We recommend looking at security-related AWS Prescriptive Guidance [guides](#) and [patterns](#), and reviewing [AWS security best practices](#).

## 1. Data-centric management

Data management is the practice of ensuring that data used in training, testing, and inference is properly managed, secured, and validated. When building models at scale, data is the primary commodity that enables high model performance.

### 1.1 Data repository

A data repository requires the ability to track data and see its point of origination. When new data is added or removed, the data repository records those changes in point-in-time recovery. The data repository should take into account how label data is tracked

and processed, and how intermediate data artifacts are tracked.

## 1.2 Diverse data source integration

Depending on the application, training your model might require data from many sources. Designing and maintaining a manifest that informs ML practitioners of the available data sources and how they tie together is critical to building models.

## 1.3 Data schema validation

To feed models data, it's important that the training data be homogeneous. Transformations or other exploratory analysis might be required for data that is stored in data lake solutions such as Amazon Simple Storage Service (Amazon S3) or in document data stores.

## 1.4 Data versioning and lineage

When training models that might be used in production, you must be able to reproduce results and have a reliable way to perform [ablation studies](#) to better understand the overall model performance. Tracking the state of the training data is critical to this reproducibility. Tools such as [Data Version Control \(DVC\)](#) can assist with this.

## 1.5 Labeling workflow

In cases where labeled data is not available at the project start, creating labeled data is often a necessary step. Tools such as Amazon SageMaker Ground Truth require input data to be appropriately structured, and they require a defined and tested labeling job. A workforce of either internal or external labelers must be used. Data should then be validated, using either redundant labeling or machine learning approaches to identify outliers or errors in the training dataset.

## 1.6 Online and offline feature storage

The ML system has a [Feature Store](#) or a centralized store for features and associated metadata so that it's possible to reuse features, or model inputs. You can create an online or an offline store. Use an online store for low-latency, real-time inference use cases. Use an offline store for training and batch inference.

# 2. Experimentation

Experimentation covers experiment logging, tracking, and metrics. This translates to experiment metadata integration across the platform, in source control, and in development environments. Experimentation also includes being able to optimize model performance and accuracy through debugging.

## 2.1 Integrated development environments

An integrated development environment (IDE) is integrated directly with the cloud. The IDE can interact with and submit commands to the larger system. Ideally, it supports the following:

- Local development
- Version control integration



- Debugging in place, with all logs and artifacts generated going into the version control

## 2.2 Code version control

To help ensure reproducibility and reusability, all code is committed into the source repository with proper version control. This includes infrastructure code, application code, model code, and even notebooks (if you opt to use them).

## 2.3 Tracking

An ML project requires a tool that can track and analyze machine learning experiments. This tool should log all metrics, parameters, and artifacts during a machine learning experiment run, recording all metadata into a central location. The central location will provide the ability to analyze, visualize, and audit all experiments that you run.

## 2.4 Cross-platform integration

Historical results for experiments and all their metadata are accessible in other parts of the system. For example, the orchestration pipelines in place can access this data, as can the monitoring tools.

## 2.5 Debugging: accuracy and system performance

A comprehensive model debugging framework is in place to examine runs for the following:

- Find bottlenecks
- Alert about anomalies
- Maximize resource utilization
- Aid in analysis of experiments

When training is intensive, the ability to maximize throughput is crucial and makes this a necessary tool for cost optimization.

## 3. Observability and model management

The observability and model management section of the checklist encompasses model version control and lineage tracking across the entire ML system. Model versioning helps to track and control all changes applied to a model so that you can recover a previous version when needed. Lineage tracking provides a view into model inflows and outflows. Another key benefit of lineage tracking is point-in-time recovery (PITR), which automates deployment and system recovery.

### 3.1 Versioned model registry

In general, a model registry supports version control and lineage tracking of model components. A good registry can associate metadata with the versioned model, including the following:

- The data used
- Information about the model
- Evaluation metric results
- Associated model code

### 3.2 Bias, fairness, and explainability

At a bare minimum, an ML system should have a process whereby a model's predictions are explainable to other parties. Users should be able to check results for bias by each feature. Ideally, measure data bias before inputting the data into the ML model, and record these metrics for model cards and auditing.

### 3.3 Lineage tracking: data inputs and outputs

Tracking is in place to follow the flow of data in and out of the system (for example, runs from the data lake to the training pipeline). This tracking acts as a record from which all system processes can be recreated, and it provides an audit trail for analysis.

### 3.4 Lineage tracking: environment information

This tracking captures information about the runtime environment setup, such as container

images for all model code and the containers' associated dependencies.

### **3.5 Lineage tracking: model**

This tracking captures information about the model. It includes everything from information on the model's algorithm to parameters and hyperparameters that go into the model.

### **3.6 Integration with deployment and monitoring**

The system should be linked directly with monitoring and deployment subsystems for PITR. For monitoring, this means testing the model's performance against its training runs to detect model-quality deterioration. For deployment, this supports PITR and the ability to roll back to a previous model version as needed.

### **3.7 Pipeline parameter configuration**

Technically, pipeline parameter configuration falls under both lineage tracking and experiment tracking because the pipeline configuration must be versioned and associated directly with a model. Pipeline parameter configuration is listed in this section because it's imperative to track all system orchestration configurations and version them.

### **3.8 Issues are traceable, debuggable, and reproducible.**

An engineer can trace, debug, and reproduce all issues within the system without much effort. This implies that a sufficient level of observability is in place. This check is primarily derived from fulfilling the other items under the Observability and model management section.

### 3.9 Performance visualization

The system can capture and gather logs into a time-series database type format and ingest them directly into dashboard. The dashboard provides a holistic view of both model and computer metrics with the ability to drill down and query.

## 4. Robust pipelines and promotion

Pipelines provide many options for hyperparameter tuning, AutoML, and processing routines. Pipelines are logged from end to end. Robust pipelines can run training in parallel across multiple instances and frameworks, scaling load sizes as needed. Robust pipelines can promote models into production, deploying in real-time, streaming, and batch. These deployments can support single-model or multi-model inference.

### 4.1 Large-scale and distributed training

A mature ML system supports the ability to run training on large compute-optimized instances in parallel. It has the tooling in place to help ensure that these resources are fully used and that the training scales evenly across the compute cluster.

### 4.2 Support for multiple frameworks

Developers can port different platform frameworks, such as PyTorch or Flax, to run training and inference jobs. Likewise, different languages and versions are supported and usable. Switching to another framework will not break the system.

### 4.3 Hyperparameter tuning

A hyperparameter tuning step is a part of the training pipeline. Deployed models have their hyperparameters tuned. Multiple options are available to tune hyperparameters. For accuracy improvement, at least one of the tuning options should have a Bayesian inference or approach.

#### 4.4 AutoML option

To reduce manual experimentation and comparison, a mature ML system supports running AutoML, which automatically selects the best feature pipeline, hyperparameters, and model. Note that AutoML is a feature to use pragmatically, but it's not a panacea.

#### 4.5 Inference support: real time

This is commonly called model as a service (MaaS). The system supports real-time inference with REST API operations, for inference request on demand. It's able to ship MaaS infrastructure on which the model can scale both horizontally and vertically as a standalone API or as an endpoint associated with other applications. Alternatively, it's possible to deploy by using serverless technology.

#### 4.6 Inference support: streaming

Models can be promoted to a real-time inference format such as Amazon Kinesis or Amazon Managed Streaming for Apache Kafka, whereby inference is run in streaming fashion on the model. This requires at least 90 percent of the checklist to be complete, because guardrails, observability, and monitoring are essential for real-time inference.

#### 4.7 Inference support: batch

The system supports batch deployment of models as scheduled or initiated jobs. The system can run models as part of an extract, transform, and load (ETL) process or in isolation. Batch jobs record the state from each step and run in an ordered pattern, such as a directed acyclic graph. Alternatively, jobs can write to a database, which acts as the server of model inference.

#### **4.8 Preprocessing and post-processing routines**

When needed, data is featurized as part of the model intake process or the batch jobs. If there are multiple models or multiple steps at play, post-processing routines take care of featurizing the data.

#### **4.9 Ability to invoke hierarchical or simultaneous models**

The ML system can deploy many models together or run them in a sequential fashion. The former means hosting on a single model endpoint across a fleet of resources. The latter implies that multiple models need to run in a chained fashion one after the other. The system can handle both these types of complexity resiliently.

#### **4.10 Horizontal and vertical scaling strategies**

A pipeline should have the ability to support both types of scaling strategies for training and inference. The ML system can increase its size and distribute traffic across multiple machines when either latency or throughput increases. Policies for this type of behavior are set, and they consider optimal resource allocation.

#### **4.11 End-to-end logging**

The development team should have logging set inside all pipeline code so that logging can capture inputs, outputs, and intermediate steps in the system. Logging should support tracing runs in the pipeline and debugging errors.

## **5. Continuous integration**

The ML system runs tests to validate that the system works from end to end, checking for possible points of failure. Tests are run automatically on commit, and longer tests are run on a fixed

schedule. Tests check traditional software engineering areas such as at the unit and system level. In addition, tests capture the particulars of ML by checking data, features, and the model.

### **5.1 Local code checks**

Before committing code into a centralized code repository, developers locally run checks such as basic unit tests and static analysis. Running these checks before committing increases overall code quality and catches problems before they enter version control.

### **5.2 Static code analysis**

The central code repository has static code analysis tools that run on commit, quickly. This tooling should improve code style and formatting. It should also check for common security vulnerabilities within source and infrastructure code, common bugs, and other weaknesses in the code.

### **5.3 Data quality tests**

Data quality tests should, at a bare minimum, check that the data has not violated a fixed schema. A more comprehensive approach is to compute data statistics at ingest, set constraints on the data, and run tests against these.

It's possible to set up data quality tests independently or as part of the pipeline. The statistics and constraints are reused for monitoring.

### **5.4 Feature tests**

As part of a complete pipeline, feature importance is generated. Feature tests assert that the importance of features, or the model's way of attributing feature values, does not change. Feature tests can feed into monitoring because they can alert and track violations in a model's inputs.

## 5.5 Unit tests

Unit tests for all code—model, application, and infrastructure—run before commit and on commit. Each unit test provides a check on an important piece of code to confirm that it functions as expected. In the case of ML code, tests can run for algorithmic correctness.

## 5.6 Integration tests

An integration test verifies that the pipeline runs end to end successfully, including standing up the associated infrastructure for the pipeline. This test validates that the system is working and logging as expected. If deployment is separate, there should be an end-to-end test for this as well to make sure deployment works.

## 5.7 Smoke tests

The system has smoke tests that run in mini and rapid regression of each piece of functionality. Smoke tests are part of continuous integration, and can run in a containerized environment to mimic cloud functionality.

## 5.8 Load testing

On-demand load testing is in place. In addition to capturing how the ML system behaves under high and low loads, load tests provide statistics on system-wide throughput or latency. Data gathered through load tests provides information about resource sizes and scaling policies.

## 5.9 Model functional tests

Model outputs and inputs run through automated functional tests. To check a behavior within a capability, both outputs and inputs for the model are tested on real or fake data with basic examples.



### 5.10 Model inference tests with extreme cases

As part of the minimum functionality testing, model tests should check for extreme behavior given certain inputs before model promotion . This places an additional guardrail to help prevent unexpected behavior.

## 6. Continuous monitoring

In continuous monitoring, automated processes observe and detect performance issues and model issues. Owners can then identify potential problems and threats in real time to address them quickly.

Continuous monitoring surfaces possible model issues such as data quality, distribution shift, model concept shift, and model quality degradation. Continuous monitoring also includes comprehensive logging for traditional system measures such as saturation, latency, traffic, and errors. A practical notification and alert strategy are set up to notify owners when issues arise.

### 6.1 Model monitoring: data quality detection

Rule-based monitoring is in place to know when incoming data deviates from model training data. This type of monitoring creates a schema from the training data, sets constraints based on that schema, and then runs exceptions when a violation occurs.

### 6.2 Model monitoring: distribution shift

Monitoring is set up to look at the incoming data distribution and check that it hasn't deviated from the model training data distribution. For example, the incoming data is sampled in [as a moving window](#) over inference data. A job is then run to test the sampled distribution and training distribution to see if they are the same.

### 6.3 Model monitoring: model concept drift

A concept drift check looks for the relationship between a model's inputs and target variable to remain unchanged from the training data.

An additional check is to confirm that the relative features and their importance don't change.

#### **6.4 Model monitoring: model evaluation check**

This is a monitoring check that evaluates whether the model's quality has degraded. The model evaluation check compares baseline evaluation metrics from training time with the incoming results to assess whether the model's accuracy level has decreased on new data. Because it computes accuracy metrics, this check requires the ground truth of new data to be available after inference.

#### **6.5 System captures: input schemas**

The ML system captures the schema of training, testing, and validation data. In addition to providing information about inputs, schemas provide statistics regarding their skew and completeness. Schemas are used for immediate testing and data quality monitoring checks in production.

#### **6.6 System captures: evaluation results and statistics**

The ML system outputs accuracy information on validation and training data. It can output the predictions and true labels from validation and training runs. These are used as monitoring constraints for the live production model.

#### **6.7 System captures: anomalies**

There is a tracking mechanism in place to flag anomalies in incoming data streams. If outliers occur in incoming data or if during a specified timeframe the key feature distribution changes, the system recognizes this as an anomaly and flags it.

## 6.8 Logging: saturation and resources

There is logging in place for how full the system is. Resource and saturation metrics should focus on CPU utilization, graphics processing unit (GPU) utilization, memory utilization, and disk utilization. These metrics should be available in time-series format with the ability to measure in percentiles. For batch jobs, this provides information on throughput, which shows how many units of information the system can process in each amount of time.

## 6.9 Logging: latency

Logging should be in place to measure the delay in network communication or the time it takes to service a request. An engineer should be able to judge how long the inference models are taking to serve predictions and how long the model takes to load.

## 6.10 Logging: traffic

The logging setup for traffic measures the volume of traffic on each instance. Traffic is measured by the number of HTTP requests and bytes or packets sent or received during a certain amount of time. Logging traffic provides insights of the total workload that is placed on a system.

## 6.11 Logging: errors

The logging setup for errors captures the number of requests that fail. Failures are of the following types:

- Explicit (for example, HTTP 500 errors)
- Implicit (for example, an HTTP 200 success response that's coupled with the wrong content)
- Policy (for example, if you commit to one-second response times, any request over one second is an error)

Where protocol response codes are insufficient to express all failure conditions, secondary (internal) protocols might be necessary to track partial failure modes.

## 6.12 Notifications and alerting

Notifications and alerts are set up from monitoring. Notifications include the ability to get Slack, email notification, pages, and Short Message Service (SMS) messages.

Alerting doesn't mean sending notifications for all possible violations. Instead, it means setting alerts to specific exceptions that are meaningful and important to the development team. In this way, alert fatigue is avoided.

# 7. Continuous deployment

For an ML system to be continuously deployed, it must be able to divert traffic from or between live models. A continuously deployed system has at least one way by which models are promoted to production: canary, shadow, blue/green, or A/B. Confirm that in the ML system, you have at least one way to roll back models.

## 7.1 Model switching

The system can switch between versioned models in staging and production. It can divert traffic all at once or incrementally to new production variants.

## 7.2 Model promotion processes

A staged validation process is in place for model promotion. The process uses offline tests that don't impact the production system, such as running against validation data in a staging environment. A runbook and metrics for model promotion are set. Promotion follows one of the rollout strategies.

## 7.3 Rollback strategies

A rollback strategy exists so that when an error occurs or the model deviates from expected behavior, a rollback, fallback, or roll through happens. In a rollback, the model reverts to a previous deployment version. In a fallback, the model is replaced with a strong heuristic. Roll through will promote the next model to production, rolling through the previous model. Runbooks are in place for all of these strategies.

## 7.4 Canary deployment

The system can deploy by using a canary. A small portion of traffic is sent to the new model initially. Over time, all traffic shifts to the new model. This shift is closely monitored because the testing happens in the production environment.

## 7.5 Model shadow deployment

The system can run a shadow deployment in which the new model works alongside the existing model. Both models receive traffic, but only the earlier model outputs inference. Assessments are run on the new model

compared with the existing model, and then the new model is manually promoted.

## 7.6 Blue/green deployment

The system can deploy with a new model (green, which is staging) and the earlier model (blue, which is production), with both running at the same time. After testing is complete, traffic is diverted from the blue environment to the green one. This strategy prevents downtime because identical environments are stood up.

## 7.7 Support for A/B testing or more

The system supports using model versions in the deployed environment to run A/B tests on incoming traffic. This can include the ability to promote automatically based on the newer model winning in the tests. More advanced setups will use a [multi-armed bandit](#) process.

# 8. Continuous training

Continuous training means that the ML system automatically and continuously retrains machine learning models to adapt to changes in the data before it is redeployed. Possible triggers for rebuilding include data changes, model changes, or code changes.

## 8.1 Checks: model input validation

Checks are in place to verify a model's input doesn't deviate from a certain standard. Input validation means running functional testing during model promotion. It also means having immediate verification of input requests, such as using assertions and enumerated types.

## 8.2 Retrain triggering: scheduled jobs

This is the most basic form of training automation. Model retraining is set on a schedule (for example, every week). In this scenario, automation is likely low, with a

manual review and spot check on the results before model promotion.

### 8.3 Retrain triggering: new training data

Retraining is initiated by an incoming data threshold. The model can retrain from scratch or run updates incrementally. Given a specified amount of data in place, a training job kicks off.

### 8.4 Retrain triggering: model performance degradation

This technique uses monitoring and observability to run model retraining, and it requires a mature level of automation. For example, accuracy lowers from a given range, which acts as a trigger for retraining a model on all or part of the data.

### 8.5 Retrain triggering: data distribution shift

Monitoring data distribution shift provides a way set triggers to retrain the model when its underlying data changes. A violation set on [concept shift or data distribution shift](#) initiates a model retraining job.

## 9. Governance

ML governance encompasses a set of processes and frameworks that help in the deployment of ML models. It includes model explainability, auditability, traceability, and other more abstract but essential requirements of a successful end-to-end ML lifecycle.

### 9.1 Data quality and compliance

The ML system accounts for personal identifying information (PII) considerations, including anonymization. It has documented and reviewed column-level lineage for understanding the source, quality, and appropriateness of the data. It also has automated data quality checks for anomalies.

## 9.2 Audit and documentation

The ML system has a full log of all changes during development, including experiments run and reasons for choices made for regulatory compliance.

## 9.3 Reproducibility and traceability

The ML system includes a full data snapshot for precise and rapid model re-instantiation, or it has the ability to recreate the environment and retrain with a data sample.

## 9.4 Human-in-the-loop signoff

The ML system has manual verification and authorization for regulatory compliance. The system requires signoffs for every environment move (for example, Dev, QA, pre-Prod, and Prod).

## 9.5 Bias and adversarial attacks testing

The ML system has *Red Team* adversarial testing using multiple tools and attack vectors, and automated bias checking on specific subpopulations. This component ties back to the Observability and model management section.



## Example usage of the MLOps checklist

There is no one way to use the [MLOps checklist](#). A common pattern, however, is to iterate on it until system quality and performance are at the levels that you want.

An iteration cycle can look like the following:

1. Go through the checklist item by item.
2. Review the existing checklist results.
3. Identify any gaps.
4. Prioritize gaps for remediation.
5. Develop implementations for new features that you want.
6. Repeat the iteration cycle.

# Resources

## References

- David Nigenda, Zohar Karnin, Muhammad Bilal Zafar, Raghu Ramesha, Alan Tan, Michele Donini, and Krishnaram Kenthapadi. 2022. "Amazon SageMaker Model Monitor: A System for Real-Time Insights into Deployed Machine Learning Models": [arXiv:2111.13657](https://arxiv.org/abs/2111.13657).
- Richard Meyes, Melanie Lu, Constantin Waubert de Puiseau, and Tobias Meisen. 2019. "Ablation Studies in Artificial Neural Networks": [arXiv:1901.08644](https://arxiv.org/abs/1901.08644).
- Brian Babcock, Mayur Datar, and Rajeev Motwani. "Sampling From a Moving Window Over Streaming Data": [Princeton University](https://princetonuniversity.github.io/).
- Chip Huyen. [\*Designing Machine Learning Systems: An Iterative Process for Production-Ready Applications\*](#). Sebastopol, California: O'Reilly Media, 2022.
- Cathy Chen, Niall Murphy, Kranti Parisa, D. Sculley, and Todd Underwood. [\*Reliable Machine Learning: Applying SRE Principles to ML in Production\*](#). Sebastopol, California: O'Reilly Media, 2022.

## Additional reading

- [AWS Well-Architected Framework: Machine Learning Lens](#)
- [Detect NLP data drift using custom Amazon SageMaker Model Monitor](#) (blog post)
- [Dynamic A/B testing for machine learning models with Amazon SageMaker MLOps projects](#) (blog post)

## Tools

- [Amazon SageMaker](#)
- [Data Version Control \(DVC\)](#)

## Consulting

- [AWS Professional Services](#)

## 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
<a href="#">Initial publication</a>	—	July 26, 2023

# AWS Prescriptive Guidance glossary

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

## Numbers

### 7 Rs

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

- **Refactor/re-architect** – Move an application and modify its architecture by taking full advantage of cloud-native features to improve agility, performance, and scalability. This typically involves porting the operating system and database. Example: Migrate your on-premises Oracle database to the Amazon Aurora PostgreSQL-Compatible Edition.
- **Replatform (lift and reshape)** – Move an application to the cloud, and introduce some level of optimization to take advantage of cloud capabilities. Example: Migrate your on-premises Oracle database to Amazon Relational Database Service (Amazon RDS) for Oracle in the AWS Cloud.
- **Repurchase (drop and shop)** – Switch to a different product, typically by moving from a traditional license to a SaaS model. Example: Migrate your customer relationship management (CRM) system to Salesforce.com.
- **Rehost (lift and shift)** – Move an application to the cloud without making any changes to take advantage of cloud capabilities. Example: Migrate your on-premises Oracle database to Oracle on an EC2 instance in the AWS Cloud.
- **Relocate (hypervisor-level lift and shift)** – Move infrastructure to the cloud without purchasing new hardware, rewriting applications, or modifying your existing operations. This migration scenario is specific to VMware Cloud on AWS, which supports virtual machine (VM) compatibility and workload portability between your on-premises environment and AWS. You can use the VMware Cloud Foundation technologies from your on-premises data centers when you migrate your infrastructure to VMware Cloud on AWS. Example: Relocate the hypervisor hosting your Oracle database to VMware Cloud on AWS.
- **Retain (revisit)** – Keep applications in your source environment. These might include applications that require major refactoring, and you want to postpone that work until a later

time, and legacy applications that you want to retain, because there's no business justification for migrating them.

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

## 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 the source and target databases are kept in sync, but only the source database handles transactions from connecting applications while data is replicated to the target database. The target database doesn't accept any transactions during migration.

### aggregate function

A SQL function that operates on a group of rows and calculates a single return value for the group. Examples of aggregate functions include SUM and MAX.

### AI

See [artificial intelligence](#).

## AIOps

See [artificial intelligence operations](#).

## anonymization

The process of permanently deleting personal information in a dataset. Anonymization can help protect personal privacy. Anonymized data is no longer considered to be personal data.

## anti-pattern

A frequently used solution for a recurring issue where the solution is counter-productive, ineffective, or less effective than an alternative.

## application control

A security approach that allows the use of only approved applications in order to help protect a system from malware.

## application portfolio

A collection of detailed information about each application used by an organization, including the cost to build and maintain the application, and its business value. This information is key to [the portfolio discovery and analysis process](#) and helps identify and prioritize the applications to be migrated, modernized, and optimized.

## artificial intelligence (AI)

The field of computer science that is dedicated to using computing technologies to perform cognitive functions that are typically associated with humans, such as learning, solving problems, and recognizing patterns. For more information, see [What is Artificial Intelligence?](#)

## artificial intelligence operations (AIOps)

The process of using machine learning techniques to solve operational problems, reduce operational incidents and human intervention, and increase service quality. For more information about how AIOps is used in the AWS migration strategy, see the [operations integration guide](#).

## asymmetric encryption

An encryption algorithm that uses a pair of keys, a public key for encryption and a private key for decryption. You can share the public key because it isn't used for decryption, but access to the private key should be highly restricted.

## atomicity, consistency, isolation, durability (ACID)

A set of software properties that guarantee the data validity and operational reliability of a database, even in the case of errors, power failures, or other problems.

## attribute-based access control (ABAC)

The practice of creating fine-grained permissions based on user attributes, such as department, job role, and team name. For more information, see [ABAC for AWS](#) in the AWS Identity and Access Management (IAM) documentation.

## authoritative data source

A location where you store the primary version of data, which is considered to be the most reliable source of information. You can copy data from the authoritative data source to other locations for the purposes of processing or modifying the data, such as anonymizing, redacting, or pseudonymizing it.

## Availability Zone

A distinct location within an AWS Region that is insulated from failures in other Availability Zones and provides inexpensive, low-latency network connectivity to other Availability Zones in the same Region.

## AWS Cloud Adoption Framework (AWS CAF)

A framework of guidelines and best practices from AWS to help organizations develop an efficient and effective plan to move successfully to the cloud. AWS CAF organizes guidance into six focus areas called perspectives: business, people, governance, platform, security, and operations. The business, people, and governance perspectives focus on business skills and processes; the platform, security, and operations perspectives focus on technical skills and processes. For example, the people perspective targets stakeholders who handle human resources (HR), staffing functions, and people management. For this perspective, AWS CAF provides guidance for people development, training, and communications to help ready the organization for successful cloud adoption. For more information, see the [AWS CAF website](#) and the [AWS CAF whitepaper](#).

## AWS Workload Qualification Framework (AWS WQF)

A tool that evaluates database migration workloads, recommends migration strategies, and provides work estimates. AWS WQF is included with AWS Schema Conversion Tool (AWS SCT). It analyzes database schemas and code objects, application code, dependencies, and performance characteristics, and provides assessment reports.

## B

### bad bot

A [bot](#) that is intended to disrupt or cause harm to individuals or organizations.

### BCP

See [business continuity planning](#).

### behavior graph

A unified, interactive view of resource behavior and interactions over time. You can use a behavior graph with Amazon Detective to examine failed logon attempts, suspicious API calls, and similar actions. For more information, see [Data in a behavior graph](#) in the Detective documentation.

### big-endian system

A system that stores the most significant byte first. See also [endianness](#).

### binary classification

A process that predicts a binary outcome (one of two possible classes). For example, your ML model might need to predict problems such as "Is this email spam or not spam?" or "Is this product a book or a car?"

### bloom filter

A probabilistic, memory-efficient data structure that is used to test whether an element is a member of a set.

### blue/green deployment

A deployment strategy where you create two separate but identical environments. You run the current application version in one environment (blue) and the new application version in the other environment (green). This strategy helps you quickly roll back with minimal impact.

### bot

A software application that runs automated tasks over the internet and simulates human activity or interaction. Some bots are useful or beneficial, such as web crawlers that index information on the internet. Some other bots, known as *bad bots*, are intended to disrupt or cause harm to individuals or organizations.



## botnet

Networks of [bots](#) that are infected by [malware](#) and are under the control of a single party, known as a *bot herder* or *bot operator*. Botnets are the best-known mechanism to scale bots and their impact.

## branch

A contained area of a code repository. The first branch created in a repository is the *main branch*. You can create a new branch from an existing branch, and you can then develop features or fix bugs in the new branch. A branch you create to build a feature is commonly referred to as a *feature branch*. When the feature is ready for release, you merge the feature branch back into the main branch. For more information, see [About branches](#) (GitHub documentation).

## break-glass access

In exceptional circumstances and through an approved process, a quick means for a user to gain access to an AWS account that they don't typically have permissions to access. For more information, see the [Implement break-glass procedures](#) indicator in the AWS Well-Architected guidance.

## brownfield strategy

The existing infrastructure in your environment. When adopting a brownfield strategy for a system architecture, you design the architecture around the constraints of the current systems and infrastructure. If you are expanding the existing infrastructure, you might blend brownfield and [greenfield](#) strategies.

## buffer cache

The memory area where the most frequently accessed data is stored.

## business capability

What a business does to generate value (for example, sales, customer service, or marketing). Microservices architectures and development decisions can be driven by business capabilities. For more information, see the [Organized around business capabilities](#) section of the [Running containerized microservices on AWS](#) whitepaper.

## business continuity planning (BCP)

A plan that addresses the potential impact of a disruptive event, such as a large-scale migration, on operations and enables a business to resume operations quickly.

# C

## CAF

See [AWS Cloud Adoption Framework](#).

## canary deployment

The slow and incremental release of a version to end users. When you are confident, you deploy the new version and replace the current version in its entirety.

## CCoE

See [Cloud Center of Excellence](#).

## CDC

See [change data capture](#).

## change data capture (CDC)

The process of tracking changes to a data source, such as a database table, and recording metadata about the change. You can use CDC for various purposes, such as auditing or replicating changes in a target system to maintain synchronization.

## chaos engineering

Intentionally introducing failures or disruptive events to test a system's resilience. You can use [AWS Fault Injection Service \(AWS FIS\)](#) to perform experiments that stress your AWS workloads and evaluate their response.

## CI/CD

See [continuous integration and continuous delivery](#).

## classification

A categorization process that helps generate predictions. ML models for classification problems predict a discrete value. Discrete values are always distinct from one another. For example, a model might need to evaluate whether or not there is a car in an image.

## client-side encryption

Encryption of data locally, before the target AWS service receives it.

## Cloud Center of Excellence (CCoE)

A multi-disciplinary team that drives cloud adoption efforts across an organization, including developing cloud best practices, mobilizing resources, establishing migration timelines, and leading the organization through large-scale transformations. For more information, see the [CCoE posts](#) on the AWS Cloud Enterprise Strategy Blog.

## cloud computing

The cloud technology that is typically used for remote data storage and IoT device management. Cloud computing is commonly connected to [edge computing](#) technology.

## cloud operating model

In an IT organization, the operating model that is used to build, mature, and optimize one or more cloud environments. For more information, see [Building your Cloud Operating Model](#).

## cloud stages of adoption

The four phases that organizations typically go through when they migrate to the AWS Cloud:

- Project – Running a few cloud-related projects for proof of concept and learning purposes
- Foundation – Making foundational investments to scale your cloud adoption (e.g., creating a landing zone, defining a CCoE, establishing an operations model)
- Migration – Migrating individual applications
- Re-invention – Optimizing products and services, and innovating in the cloud

These stages were defined by Stephen Orban in the blog post [The Journey Toward Cloud-First & the Stages of Adoption](#) on the AWS Cloud Enterprise Strategy blog. For information about how they relate to the AWS migration strategy, see the [migration readiness guide](#).

## CMDB

See [configuration management database](#).

## code repository

A location where source code and other assets, such as documentation, samples, and scripts, are stored and updated through version control processes. Common cloud repositories include GitHub or AWS CodeCommit. Each version of the code is called a *branch*. In a microservice structure, each repository is devoted to a single piece of functionality. A single CI/CD pipeline can use multiple repositories.

## cold cache

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

## cold data

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

## computer vision (CV)

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

## configuration drift

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

## configuration management database (CMDB)

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

## conformance pack

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

## continuous integration and continuous delivery (CI/CD)

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

## CV

See [computer vision](#).

## D

### data at rest

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

### data classification

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

### data drift

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

### data in transit

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

### data mesh

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

### data minimization

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

### data perimeter

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

## data preprocessing

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

## data provenance

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

## data subject

An individual whose data is being collected and processed.

## data warehouse

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

## database definition language (DDL)

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

## database manipulation language (DML)

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

## DDL

See [database definition language](#).

## deep ensemble

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

## deep learning

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

## defense-in-depth

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

## delegated administrator

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

## deployment

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

## development environment

See [environment](#).

## detective control

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

## development value stream mapping (DVSM)

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

## digital twin

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

## dimension table

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

## disaster

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

## disaster recovery (DR)

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

## DML

See [database manipulation language](#).

## domain-driven design

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

## DR

See [disaster recovery](#).

## drift detection

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

## DVSM

See [development value stream mapping](#).



# E

## EDA

See [exploratory data analysis](#).

## edge computing

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

## encryption

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

## encryption key

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

## endianness

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

## endpoint

See [service endpoint](#).

## endpoint service

A service that you can host in a virtual private cloud (VPC) to share with other users. You can create an endpoint service with AWS PrivateLink and grant permissions to other AWS accounts or to AWS Identity and Access Management (IAM) principals. These accounts or principals can connect to your endpoint service privately by creating interface VPC endpoints. For more information, see [Create an endpoint service](#) in the Amazon Virtual Private Cloud (Amazon VPC) documentation.

## enterprise resource planning (ERP)

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

## envelope encryption

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

## environment

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

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

## epic

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

## ERP

See [enterprise resource planning](#).

## exploratory data analysis (EDA)

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

## F

### fact table

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

### fail fast

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

### fault isolation boundary

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

### feature branch

See [branch](#).

### features

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

### feature importance

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

### feature transformation

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

### FGAC

See [fine-grained access control](#).

## fine-grained access control (FGAC)

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

## flash-cut migration

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

# G

## geo blocking

See [geographic restrictions](#).

## geographic restrictions (geo blocking)

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

## Gitflow workflow

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

## greenfield strategy

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

## guardrail

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

for remediation. They are implemented by using AWS Config, AWS Security Hub, Amazon GuardDuty, AWS Trusted Advisor, Amazon Inspector, and custom AWS Lambda checks.

## H

### HA

See [high availability](#).

#### heterogeneous database migration

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

#### high availability (HA)

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

#### historian modernization

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

#### homogeneous database migration

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

#### hot data

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

## hotfix

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

## hypercare period

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

## I

### IaC

See [infrastructure as code](#).

### identity-based policy

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

### idle application

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

### IIoT

See [Industrial Internet of Things](#).

### immutable infrastructure

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

### inbound (ingress) VPC

In an AWS multi-account architecture, a VPC that accepts, inspects, and routes network connections from outside an application. The [AWS Security Reference Architecture](#) recommends

setting up your Network account with inbound, outbound, and inspection VPCs to protect the two-way interface between your application and the broader internet.

## incremental migration

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

## Industry 4.0

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

## infrastructure

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

## infrastructure as code (IaC)

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

## industrial Internet of Things (IIoT)

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

## inspection VPC

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

## Internet of Things (IoT)

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

## interpretability

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

## IoT

See [Internet of Things](#).

## IT information library (ITIL)

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

## IT service management (ITSM)

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

## ITIL

See [IT information library](#).

## ITSM

See [IT service management](#).

# L

## label-based access control (LBAC)

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



## landing zone

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

## large migration

A migration of 300 or more servers.

## LBAC

See [label-based access control](#).

## least privilege

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

## lift and shift

See [7 Rs](#).

## little-endian system

A system that stores the least significant byte first. See also [endianness](#).

## lower environments

See [environment](#).

# M

## machine learning (ML)

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

## main branch

See [branch](#).

## malware

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

## managed services

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

## manufacturing execution system (MES)

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

## MAP

See [Migration Acceleration Program](#).

## mechanism

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

## member account

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

## MES

See [manufacturing execution system](#).

## Message Queuing Telemetry Transport (MQTT)

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

## microservice

A small, independent service that communicates over well-defined APIs and is typically owned by small, self-contained teams. For example, an insurance system might include

microservices that map to business capabilities, such as sales or marketing, or subdomains, such as purchasing, claims, or analytics. The benefits of microservices include agility, flexible scaling, easy deployment, reusable code, and resilience. For more information, see [Integrating microservices by using AWS serverless services](#).

## microservices architecture

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

## Migration Acceleration Program (MAP)

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

## migration at scale

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

## migration factory

Cross-functional teams that streamline the migration of workloads through automated, agile approaches. Migration factory teams typically include operations, business analysts and owners, migration engineers, developers, and DevOps professionals working in sprints. Between 20 and 50 percent of an enterprise application portfolio consists of repeated patterns that can be optimized by a factory approach. For more information, see the [discussion of migration factories](#) and the [Cloud Migration Factory guide](#) in this content set.

## migration metadata

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

## migration pattern

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

## Migration Portfolio Assessment (MPA)

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

## Migration Readiness Assessment (MRA)

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

## migration strategy

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

## ML

See [machine learning](#).

## modernization

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

## modernization readiness assessment

An evaluation that helps determine the modernization readiness of an organization's applications; identifies benefits, risks, and dependencies; and determines how well the organization can support the future state of those applications. The outcome of the assessment is a blueprint of the target architecture, a roadmap that details development phases and

milestones for the modernization process, and an action plan for addressing identified gaps. For more information, see [Evaluating modernization readiness for applications in the AWS Cloud](#).

## monolithic applications (monoliths)

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

## MPA

See [Migration Portfolio Assessment](#).

## MQTT

See [Message Queuing Telemetry Transport](#).

## multiclass classification

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

## mutable infrastructure

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

# O

## OAC

See [origin access control](#).

## OAI

See [origin access identity](#).

## OCM

See [organizational change management](#).

## offline migration

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

OI

See [operations integration](#).

OLA

See [operational-level agreement](#).

## online migration

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

OPC-UA

See [Open Process Communications - Unified Architecture](#).

Open Process Communications - Unified Architecture (OPC-UA)

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

operational-level agreement (OLA)

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

operational readiness review (ORR)

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

operational technology (OT)

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

## operations integration (OI)

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

## organization trail

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

## organizational change management (OCM)

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

## origin access control (OAC)

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

## origin access identity (OAI)

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

## ORR

See [operational readiness review](#).

## OT

See [operational technology](#).

## outbound (egress) VPC

In an AWS multi-account architecture, a VPC that handles network connections that are initiated from within an application. The [AWS Security Reference Architecture](#) recommends

setting up your Network account with inbound, outbound, and inspection VPCs to protect the two-way interface between your application and the broader internet.

## P

### permissions boundary

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

### personally identifiable information (PII)

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

### PII

See [personally identifiable information](#).

### playbook

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

### PLC

See [programmable logic controller](#).

### PLM

See [product lifecycle management](#).

### policy

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



## polyglot persistence

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

## portfolio assessment

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

## predicate

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

## predicate pushdown

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

## preventative control

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

## principal

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

## Privacy by Design

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

## private hosted zones

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

## proactive control

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

## product lifecycle management (PLM)

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

## production environment

See [environment](#).

## programmable logic controller (PLC)

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

## pseudonymization

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

## publish/subscribe (pub/sub)

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

# Q

## query plan

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

## query plan regression

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

# R

## RACI matrix

See [responsible, accountable, consulted, informed \(RACI\)](#).

## ransomware

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

## RASCI matrix

See [responsible, accountable, consulted, informed \(RACI\)](#).

## RCAC

See [row and column access control](#).

## read replica

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

## re-architect

See [7 Rs](#).

## recovery point objective (RPO)

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

## recovery time objective (RTO)

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

## refactor

See [7 Rs](#).

## Region

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

## regression

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

## rehost

See [7 Rs](#).

## release

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

## relocate

See [7 Rs](#).

## replatform

See [7 Rs](#).

## repurchase

See [7 Rs](#).

## resiliency

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

## resource-based policy

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

## responsible, accountable, consulted, informed (RACI) matrix

A matrix that defines the roles and responsibilities for all parties involved in migration activities and cloud operations. The matrix name is derived from the responsibility types defined in the

matrix: responsible (R), accountable (A), consulted (C), and informed (I). The support (S) type is optional. If you include support, the matrix is called a *RASCI matrix*, and if you exclude it, it's called a *RACI matrix*.

#### responsive control

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

#### retain

See [7 Rs](#).

#### retire

See [7 Rs](#).

#### rotation

The process of periodically updating a [secret](#) to make it more difficult for an attacker to access the credentials.

#### row and column access control (RCAC)

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

#### RPO

See [recovery point objective](#).

#### RTO

See [recovery time objective](#).

#### runbook

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

## S

#### SAML 2.0

An open standard that many identity providers (IdPs) use. This feature enables federated single sign-on (SSO), so users can log into the AWS Management Console or call the AWS API

operations without you having to create user in IAM for everyone in your organization. For more information about SAML 2.0-based federation, see [About SAML 2.0-based federation](#) in the IAM documentation.

## SCADA

See [supervisory control and data acquisition](#).

## SCP

See [service control policy](#).

## secret

In AWS Secrets Manager, confidential or restricted information, such as a password or user credentials, that you store in encrypted form. It consists of the secret value and its metadata. The secret value can be binary, a single string, or multiple strings. For more information, see [Secret](#) in the Secrets Manager documentation.

## security control

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

## security hardening

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

## security information and event management (SIEM) system

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

## security response automation

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

## server-side encryption

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

## service control policy (SCP)

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

## service endpoint

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

## service-level agreement (SLA)

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

## service-level indicator (SLI)

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

## service-level objective (SLO)

A target metric that represents the health of a service, as measured by a [service-level indicator](#).

## shared responsibility model

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

## SIEM

See [security information and event management system](#).

## single point of failure (SPOF)

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

## SLA

See [service-level agreement](#).

## SLI

See [service-level indicator](#).

## SLO

See [service-level objective](#).

## split-and-seed model

A pattern for scaling and accelerating modernization projects. As new features and product releases are defined, the core team splits up to create new product teams. This helps scale your organization's capabilities and services, improves developer productivity, and supports rapid innovation. For more information, see [Phased approach to modernizing applications in the AWS Cloud](#).

## SPOF

See [single point of failure](#).

## star schema

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

## strangler fig pattern

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

## subnet

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



## supervisory control and data acquisition (SCADA)

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

## symmetric encryption

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

## synthetic testing

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

# T

## tags

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

## target variable

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

## task list

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

## test environment

See [environment](#).

## training

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

## transit gateway

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

## trunk-based workflow

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

## trusted access

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

## tuning

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

## two-pizza team

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

# U

## uncertainty

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

## undifferentiated tasks

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

## upper environments

See [environment](#).

# V

## vacuuming

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

## version control

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

## VPC peering

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

## vulnerability

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

# W

## warm cache

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

## warm data

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

## window function

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

## workload

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

## workstream

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

## WORM

See [write once, read many](#).

## WQF

See [AWS Workload Qualification Framework](#).

## write once, read many (WORM)

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

# Z

## zero-day exploit

An attack, typically malware, that takes advantage of a [zero-day vulnerability](#).

## zero-day vulnerability

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

## zombie application

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