AWS Prescriptive Guidance
Machine learning model interpretability with AWS
AWS Prescriptive Guidance: Machine learning model interpretability with AWS

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

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

Introduction .......................................................................................................................... 1
   Targeted business outcomes .............................................................................................. 1
Overview ............................................................................................................................... 2
Best practices ......................................................................................................................... 5
   Local interpretability .......................................................................................................... 5
      For tree ensemble models, use tree SHAP ................................................................. 5
      For neural networks and differentiable models, use integrated gradients and conductance ............................................................................................................. 6
      For all other cases, use Kernel SHAP ........................................................................... 6
   Recommended visualizations .......................................................................................... 6
Global interpretability .......................................................................................................... 8
Interpretability on AWS ...................................................................................................... 10
FAQ .................................................................................................................................... 11
   There are many methods for determining feature importance that are not discussed here. Why are they not mentioned? ................................................................. 11
   What are the weakness of the recommended methods? .................................................. 11
Resources ............................................................................................................................. 12
AWS Prescriptive Guidance glossary .................................................................................. 13
Document history ................................................................................................................. 21
Machine learning model interpretability with AWS

Adewale Akinfaderin, Matthew Chasse, Michele Donini, and Benjamin Fenker

Amazon Web Services (AWS)

February 2022

It's easier for end-users to employ machine learning algorithms responsibly when they can understand why a model makes a specific prediction. For model developers, greater insight into how a model makes predictions can aid in feature engineering and selection. There is no standard definition of what it means to explain a model, except that an explanation should be a prerequisite for standards such as trust, robustness, causality, informativeness, model transferability, and fair and ethical decision-making. There are some common methods for generating interpretations, but they come with different weaknesses and strengths. This is not unexpected: Typically, the heuristic or set of simplifying assumptions that you use to interpret a complex model can simultaneously be a source of inaccuracy for the interpretation.

This guide provides general guidance about model interpretability methods for machine learning practitioners. For brevity, the guide omits many details and implementation specifics, and provides references to help you investigate specific use cases in more depth.

Targeted business outcomes

In some cases, regulations such as those in the healthcare and finance industries require model interpretability as a desired business outcome. Model interpretations also provide additional insight that both model developers and users can utilize. Additional targeted business outcomes for employing model interpretability include the following:

- Justify important decisions (for example, in healthcare and finance) that affect customer well-being when fairness is critical.
- Control model inaccuracies and distortions when making business decisions.
- Improve and expedite model development and feature engineering when model interpretations are used by data scientists.
- Discover reasons for general model behaviors, and provide new insights about both the data and the model.

These business outcomes map directly to the four reasons for explainability that are identified in [1 (p. 12)].
Overview

There is no universally accepted definition for what an interpretable model is, or what information is adequate as an interpretation of a model. This guide focuses on the commonly used notion of *feature importance*, where an importance score for each input feature is used to interpret how it affects model outputs. This method provides insight but also requires caution. Feature importance scores can be misleading and should be analyzed carefully, including validation with subject matter experts if possible. Specifically, we advise you not to trust feature importance scores without verification, because misinterpretations can lead to poor business decisions.

In the following illustration, the measured features of an iris are passed into a model that predicts the species of the plant, and associated feature importances (SHAP attributions) for this prediction are displayed. In this case, the petal length, petal width, and sepal length all contribute positively to the classification of *Iris virginica*, but sepal width has a negative contribution. (This information is based on the iris dataset from [4 (p. 12)].)
Feature importance scores can be **global**, indicating that the score is valid for the model across all inputs, or **local**, indicating that the score applies to a single model output. Local feature importance scores are often scaled and summed to produce the model output value, and thus termed **attributions**. Simple models are considered more interpretable, because the effects of the input features on model output are more easily understood. For example, in a linear regression model, the magnitudes of the coefficients provide a global feature importance score, and for a given prediction, a local feature attribution is the product of its coefficient and the feature value. In the absence of a direct local feature importance...
score for a prediction, you can compute an importance score from a set of baseline input features to understand how a feature contributes relative to the baseline.
Best practices

Model interpretability is a growing area of study, and best practices must be understood in this context. This section provides current recommendations that are most likely to lead to positive results in a short amount of time. You can use these with other methods to fully interrogate models as time permits. The following flowchart outlines AWS recommendations for finding the best interpretability method. The sections that follow discuss these methods in more detail.

Local interpretability

The most popular methods for local interpretability of complex models are based on either Shapley Additive Explanations (SHAP) [8 (p. 12)] or integrated gradients [11 (p. 12)]. Each method has a number of variants that are specific to a model type.

For tree ensemble models, use tree SHAP

In the case of tree-based models, dynamic programming allows for fast and exact computation of the Shapley values for each feature, and this is the recommended approach for local interpretations in tree ensemble models. (See 7 (p. 12)], implementation is at https://github.com/slundberg/shap.)
For neural networks and differentiable models, use integrated gradients and conductance

Integrated gradients provide a straightforward way to compute feature attributions in neural networks. Conductance builds on integrated gradients to help you interpret attributions from portions of neural networks such as layers and individual neurons. (See [3, 11 (p. 12)], implementation is at https://captum.ai/.) You cannot use these methods on models without using a gradient; in such cases, you can use Kernel SHAP (discussed in the next section) instead. When the gradient is available, integrated gradient attributions can be computed more quickly than attributions from Kernel SHAP. A challenge to using integrated gradients is choosing the best base point for deriving an interpretation. For example, if the base point for an image model is the image of zero intensity in all the pixels, important regions of an image that are darker might not have attributions that align with human intuition. One approach to address this problem is to use multiple base point attributions and add them together. This is part of the approach taken in the XRAI feature attribution method for images [5 (p. 12)], where the integrated gradient attributions that use a black reference image and a white reference image are added together to produce more consistent attributions.

For all other cases, use Kernel SHAP

You can use Kernel SHAP to compute feature attributions for any model, but it is an approximation to computing the full Shapley values and remains computationally expensive (see [8 (p. 12)]). The computational resources required for Kernel SHAP grow quickly with the number of features. This requires approximation methods that can reduce the fidelity, repeatability, and robustness of explanations. Amazon SageMaker Clarify provides convenience methods that deploy prebuilt containers for computing Kernel SHAP values in separate instances. (For an example, see the GitHub repository Fairness and Explainability with SageMaker Clarify.)

For single tree models, the split variables and leaf values provide an immediately explainable model, and the methods discussed previously do not provide additional insight. Similarly, for linear models, the coefficients provide a clear explanation of model behavior. (SHAP and integrated gradient methods both return contributions that are determined by the coefficients.)

Both SHAP and integrated gradient-based methods have weaknesses. SHAP requires attributions to be derived from a weighted average of all feature combinations. Attributions obtained in this way can be misleading when estimating feature importance if there is a strong interaction between features. Methods that are based on integrated gradients can be difficult to interpret because of the large number of dimensions that are present in large neural networks, and these methods are sensitive to the choice of a base point. More generally, models can use features in unexpected ways to achieve a certain level of performance and these can vary with the model—feature importance is always model dependent.

Recommended visualizations

The following chart presents several recommended ways to visualize the local interpretations that were discussed in the previous sections. For tabular data we advise a simple bar graph that shows the attributions, so they can be easily compared and used to infer how the model is making predictions.
For text data, embedding tokens leads to a large number of scalar inputs. The methods recommended in the previous sections produce an attribution for each dimension of the embedding and for each output. In order to distill this information into a visualization, the attributions for a given token can be summed. The following example shows the sum of the attributions for the BERT-based question answering model that was trained on the SQUAD dataset. In this case, the predicted and true label is the token for the word “france.”

for intermediate layers in deep learning models, similar aggregations can be applied to conductances for visualization, as shown in the following example. This vector norm of the token conductance for transformer layers shows the eventual activation for the end token prediction (“france”).
**Concept activation vectors** provide a method for studying deep neural networks in more detail [6 (p. 12)]. This method extracts features from a layer in an already trained network and trains a linear classifier on those features to make inferences about the information in the layer. For example, you might want to determine which layer of a BERT-based language model contains the most information about the parts of speech. In this case, you could train a linear part-of-speech model on each layer output and make a rough estimate that the best performing classifier is associated with the layer that has the most part-of-speech information. Although we do not recommend this as a primary method for interpreting neural networks, it can be an option for more detailed study and aid in the design of network architecture.

**Global interpretability**

Understanding how features contribute to a model’s output overall provides general insight that is useful for feature selection and model development. To measure the effect of adding a new feature, you typically run cross-validation with and without the feature. However, running a cross-validation for all feature combinations and all model types under consideration is often infeasible due to the computational cost. Other methods for determining feature importance are therefore useful for making quick decisions. Our recommendation for determining global feature attributions is to aggregate the local feature attribution scores recommended in the previous section (p. 5) across all data. We also recommend computing the change in the cross-validation score when a feature is removed, if time and computational constraints allow it. The following example illustrates the aggregation of local attribution scores. It averages the magnitudes of the SHAP values for the iris classification model (from the overview (p. 2)) and plots them as a heatmap. You can see that the sepal measurements don’t play a strong role in the model for determining the iris class.
For a specified model output, the collection of SHAP values across the evaluation instances can be visualized in a beeswarm plot, as illustrated in the following diagram (for a subset of data from the iris dataset [4 (p. 12)]). Here you can see that the petal_width attribute has the largest effect on the model output for the class Iris-versicolor, and that a high petal_width value contributes negatively to the class prediction. When more than one data point has the same or very similar feature attribution value, the dots are stacked to indicate the larger prevalence at that location.
Interpretability on AWS

You can use Jupyter instances that are managed by Amazon SageMaker to easily install Python modules through Conda and `pip`. For information about Python packages for SHAP and integrated gradient-based methods, see the Resources (p. 12) section. For smaller jobs and local testing on a SageMaker Jupyter instance, using the methods from these Python packages might be sufficient. If you are using a SageMaker managed model, SageMaker Clarify provides convenience methods for launching Kernel SHAP on a dedicated instance, and offloading the computation while a model developer continues to work on their Jupyter instance. For more information, see Create Feature Attribute Baselines and Explainability Reports in the SageMaker documentation.
FAQ

There are many methods for determining feature importance that are not discussed here. Why are they not mentioned?

This guide focuses on what we believe to be the most effective and direct methods for model interpretability. Other methods have advantages in speed and ease of computation, and might be appropriate depending on the model. The guidance in this article is prescriptive, not proscriptive.

What are the weakness of the recommended methods?

SHAP requires attributions that are derived from a weighted average of all feature combinations. Attributions that are obtained this way can be misleading in estimating feature importance when there are strong interactions among features. Methods that are based on integrated gradients can be difficult to interpret because of the large number of dimensions that are present in large neural networks. Models can use features in unexpected ways to achieve a certain level of performance and these can vary with the model, so feature importance is always model dependent.
Resources

References


External software packages

- SHAP: https://github.com/slundberg/shap
- Captum: https://captum.ai/

Additional reading

- Amazon SageMaker Clarify Model Explainability (SageMaker documentation)
- Amazon SageMaker Clarify repository (GitHub)
AWS Prescriptive Guidance glossary

AI and ML terms

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

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>binary classification</td>
<td>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?”</td>
</tr>
<tr>
<td>classification</td>
<td>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.</td>
</tr>
<tr>
<td>data preprocessing</td>
<td>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.</td>
</tr>
<tr>
<td>deep ensemble</td>
<td>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.</td>
</tr>
<tr>
<td>deep learning</td>
<td>An ML subfield that uses multiple layers of artificial neural networks to identify mapping between input data and target variables of interest.</td>
</tr>
<tr>
<td>exploratory data analysis (EDA)</td>
<td>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.</td>
</tr>
<tr>
<td>features</td>
<td>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.</td>
</tr>
<tr>
<td>feature importance</td>
<td>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.</td>
</tr>
</tbody>
</table>
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.

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.

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?"

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).

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.

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.

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.

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.

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

7 Rs

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

- Refactor/re-architect – Move an application and modify its architecture by taking full advantage of cloud-native features to improve agility, performance, and scalability. This typically involves porting the operating system and database. Example: Migrate your on-premises Oracle database to the Amazon Aurora PostgreSQL-Compatible Edition.
• Replatform (lift and reshape) – Move an application to the cloud, and introduce some level of optimization to take advantage of cloud capabilities. Example: Migrate your on-premises Oracle database to Amazon Relational Database Service (Amazon RDS) for Oracle in the AWS Cloud.

• Repurchase (drop and shop) – Switch to a different product, typically by moving from a traditional license to a SaaS model. Example: Migrate your customer relationship management (CRM) system to Salesforce.com.

• Rehost (lift and shift) – Move an application to the cloud without making any changes to take advantage of cloud capabilities. Example: Migrate your on-premises Oracle database to Oracle on an EC2 instance in the AWS Cloud.

• Relocate (hypervisor-level lift and shift) – Move infrastructure to the cloud without purchasing new hardware, rewriting applications, or modifying your existing operations. This migration scenario is specific to VMware Cloud on AWS, which supports virtual machine (VM) compatibility and workload portability between your on-premises environment and AWS. You can use the VMware Cloud Foundation technologies from your on-premises data centers when you migrate your infrastructure to VMware Cloud on AWS. Example: Relocate the hypervisor hosting your Oracle database to VMware Cloud on AWS.

• Retain (revisit) – Keep applications in your source environment. These might include applications that require major refactoring, and you want to postpone that work until a later time, and legacy applications that you want to retain, because there’s no business justification for migrating them.

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

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

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

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

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

**AWS Workload Qualification Framework (AWS WQF)**
A tool that evaluates database migration workloads, recommends migration strategies, and provides work estimates. AWS WQF is included with AWS Schema
AWS Prescriptive Guidance Machine
learning model interpretability with AWS

Conversion Tool (AWS SCT). It analyzes database schemas and code objects, application code, dependencies, and performance characteristics, and provides assessment reports.

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

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

**cloud stages of adoption**
The four phases that organizations typically go through when they migrate to the AWS Cloud:

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

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

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

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

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

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

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

**IT information library (ITIL)**
A set of best practices for delivering IT services and aligning these services with business requirements. ITIL provides the foundation for ITSM.
### IT service management (ITSM)
Activities associated with designing, implementing, managing, and supporting IT services for an organization. For information about integrating cloud operations with ITSM tools, see the [operations integration guide](#).

### large migration
A migration of 300 or more servers.

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

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

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

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

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

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

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

### migration strategy
The approach used to migrate a workload to the AWS Cloud. For more information, see the 7 Rs (p. 14) entry in this glossary and see [Mobilize your organization to accelerate large-scale migrations](#).

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

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

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

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

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

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

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

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

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

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

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

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

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)
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.

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

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

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

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

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

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

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

**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.

two-pizza team

A small DevOps team that you can feed with two pizzas. A two-pizza team size ensures the best possible opportunity for collaboration in software development. For more information, see the Two-pizza team section of the Introduction to DevOps on AWS whitepaper.
Document history

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

<table>
<thead>
<tr>
<th>update-history-change</th>
<th>update-history-description</th>
<th>update-history-date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial publication (p. 21)</td>
<td>—</td>
<td>February 28, 2022</td>
</tr>
</tbody>
</table>