Best practices for assessing applications to be retired during a migration to the AWS Cloud

AWS Prescriptive Guidance

Best practices for assessing applications to be retired during a migration to the AWS Cloud: AWS Prescriptive Guidance

Table of Contents

Best practices for assessing applications to be retired during a migration to the AWS Cloud

Damien Renner, Migrations Specialist, AWS Prescriptive Guidance

September 2020 (last update (p. 19): May 2022)

Choosing to retire applications during a migration to the Amazon Web Services (AWS) Cloud can be a complex decision. By following this guide's best practices you can implement a smoother and more efficient retirement process for your applications.

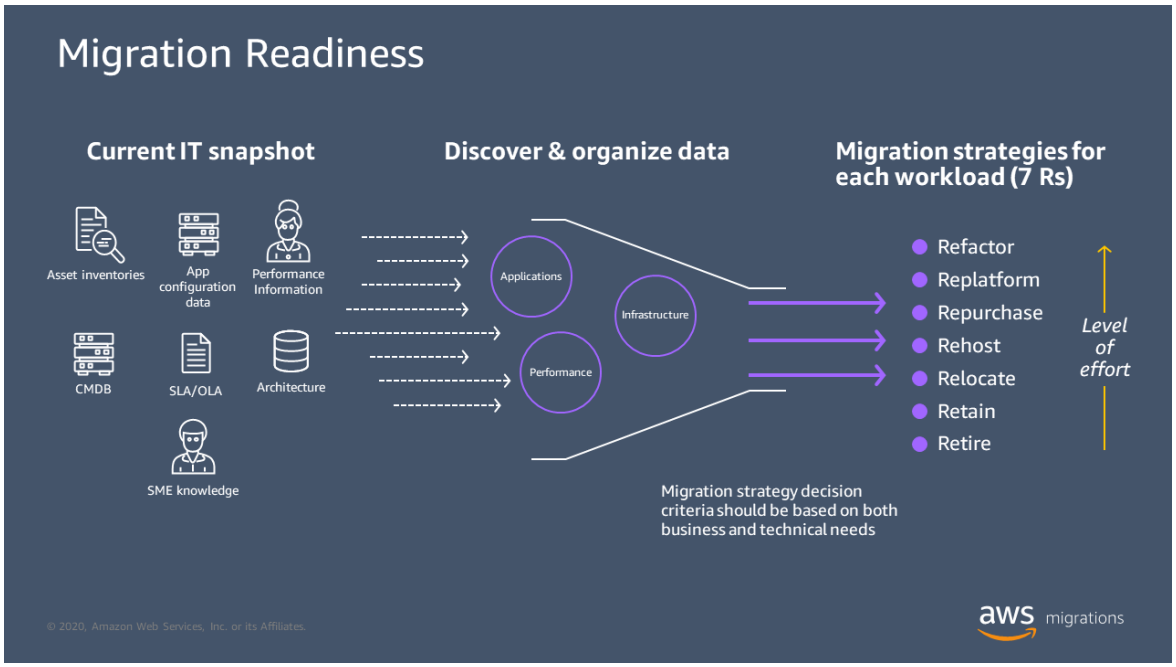
This guide is for technology or application owners who have analyzed their estate and identified applications that can be retired or decommissioned.

Overview

The purpose of this guide is to provide best practices for applications categorized as “retire” in a migration plan for the Amazon Web Services (AWS) Cloud.

A critical first step in creating a migration strategy is collecting application portfolio data, such as extracts from a configuration management database. This data must be evaluated against the seven common migration strategies (7 Rs) for moving applications to the AWS Cloud. These strategies are refactor, replatform, repurchase, rehost, relocate, retain, and retire. For more information about the 7 Rs, see the 7 Rs entry in the AWS Prescriptive Guidance glossary (p. 11).

After you complete this initial portfolio analysis, you should have an initial plan for how each application will be migrated. This plan should be regularly optimized for future migration waves and teams, based on lessons learned and new data that becomes available during the migration process. The following diagram illustrates the strategy planning process.



Deciding if applications should be retired can often become complex and involves a level of risk. This can result in action being postponed, especially if subject matter experts (SMEs) have left an organization. Also, documentation about legacy systems can be sparse.

However, identifying and retiring applications that are no longer useful helps boost your business case and directs your team’s attention toward maintaining resources that are more widely used. This guide outlines six best practices (p. 4) to use when assessing applications to be retired in your migration strategy.

Targeted business outcomes

After you assign a migration strategy to each application, it may become apparent that some of those applications can be retired. It's not unusual to find that more than 10 percent of workloads in an enterprise IT portfolio are no longer useful and can be turned off, thereby creating savings for your organization.

These savings can help direct your team's attention to systems or applications that provide value to your business; it also reduces the scope of applications you have to migrate, secure, and operate. Additionally, it's worth evaluating the long-term benefits of keeping unused applications, because they might still need to be licensed, maintained, and upgraded to remain operational.

Retiring applications can cause uncertainty and a level of risk. Because institutional knowledge about legacy systems can often be limited by outdated documentation or personnel changes, there's a risk that IT assets labeled as "no longer required" are, in fact, being consumed elsewhere within your organization.

To avoid this situation, you must fully understand all upstream dependencies of an application within your organization. This is especially important when you are working toward a business outcome, such as migrating workloads to the AWS Cloud so a data center can be closed by a defined date.

Deploying this guide's six best practices (p. 4) and its data-driven methodology will help reduce this risk level when choosing to retire applications.

Best practices

Choosing to retire applications can be a complex decision, especially during a migration to the AWS Cloud. The following sections provide best practices to use in your decision-making process.

Topics

Consider a migration factory approach

An important part of any large-scale migration is establishing a *migration factory* after the initial pilot workloads are migrated.

A migration factory consists of teams, tools, and processes that work together to streamline migrations in a systematic way, incorporating lessons learned from previous migration waves. The migration factory applies patterns, which accelerate workload migrations and improve the final outcome.

Based on the size of the IT portfolio you need to retire, it's worth considering if there's value in implementing a migration factory approach. The methodologies and principles outlined in this guide will also complement this approach and can be embedded into its mechanisms.

Typically, twenty to fifty percent of an enterprise application portfolio consists of repeated patterns that can be optimized by using a migration factory approach. For an example of a pattern, see the AWS Cloud Migration Factory solution, which can be implemented by a migration team to coordinate and automate migrations.

The team should begin with applications that have the lowest business criticality, before gradually moving toward more critical systems. By the time the team begins to migrate business-critical systems, they will have migrated hundreds, if not thousands, of workloads and learned many lessons.

Before the assessment phase begins, you can create a process to capture one month of dependency data for applications that are identified for retirement. A team is notified and given access to the data when it is ready. The team then gives the data a score based on the potential for an application to cause an impact. The application owners might then do a deeper analysis of the connections before next steps begin.

For more information about the migration factory methodology, see the AWS Migration whitepaper.

Identify and retire applications early in your migration

Identifying and then retiring applications early in the migration process is important and should be carried out while workloads are being migrated.

Migration projects often prioritize migrating workloads, so it's common for systems that are identified as to be retired (and not migrated) to receive focus toward the end of the project. However, there's a risk that leaving these applications until the end of the project will leave very little time to include it in the migration if the application is later deemed important.

Retiring workloads early during a migration project reduces the workload of teams that maintain them. For example, retiring servers during a migration project's early phases means that operating system teams have fewer servers to patch, upgrade, maintain, or support. Those teams are then able to focus on the migration project itself.

Finally, some of this guide's best practices are most effective when you follow them for longer periods of time. If you begin the retirement process early but later determine that an application is actually required by another service, you will be able to modify your migration plan and include it in a future migration wave.

Be data-driven and use discovery tooling to avoid disruptions

Being data-driven is critical when considering retiring applications. Architecture diagrams and institutional knowledge can easily be outdated or incomplete. Sometimes unanticipated problems can also surface, such as another application becoming dependent on your system without formal engagement due to a break-fix scenario.

A data-driven approach provides the foundation on which you can make decisions or validate an approach. When assessing if an application can be retired, you must confirm that the workloads you are migrating aren't dependent on it. Migrating those workloads and then retiring a dependency could cause a service degradation, or worse, a service interruption.

Fortunately, it's fairly straightforward to understand these dependencies by using data to monitor the network inbound and outbound connections on a server that is scheduled for retirement. Network inbound connections, such as an application connecting to your application, and outbound connections, such as a file upload to a Network File System (NFS) share, indicate a potential upstream dependency. This dependency needs to be investigated, because if a workload that's due to be migrated to the AWS Cloud connects to the application, there's a potential for service disruption if the application is retired later on. This process might require diving deep to follow the dependency chain. Following the previous example, if the application uploads a file to an NFS share, the next step is to determine which system consumes that file and the status of that application.

You might decide to investigate those connections and assess the impact level. To do this, you can use discovery tooling to show the connections being initiated to a server that is scheduled for retirement. You might notice that most connections are from management servers and can be ignored, because they are tools that collect performance metrics or system administrator proxy instances. However, if there are applications connecting to the server that are scheduled for migration, you should dive deeper and check the potential impact of migration on that application.

AWS Application Discovery Service helps customers plan migration projects by gathering information about on-premises data centers that they are planning to retire. After you deploy the agent to your servers, Application Discovery Service records inbound and outbound network activity for each server, along with other information. By using Amazon Athena to analyze this data, you can identify if other applications are depending on servers that are planned for retirement. AWS Migration Competency Partners can also provide deep discovery and planning tooling.

For example, the following screen illustration shows four source IP addresses connecting to the server on port 22 (destination = 172.31.1.117).

Results

	account_number	agent_id	source_ip	source_port	destination_ip	destination_port	ip_version	transport_protocol	agent_assigned_process_id	agent_creation_date
1	[redacted]	o-57bo0a320czmd6v4d	172.31.4.134		172.31.1.117	80	IPv4	TCP	3E43D4CF02E35EB1CC57D121CAFCA3DB132A9FA1	2020-08-15 17:15:00.000
2	[redacted]	o-57bo0a320czmd6v4d	172.31.12.72		172.31.1.117	80	IPv4	TCP	3E43D4CF02E35EB1CC57D121CAFCA3DB132A9FA1	2020-08-15 17:15:00.000
3	[redacted]	o-57bo0a320czmd6v4d	67.54.159.145		172.31.1.117	22	IPv4	TCP	3B5033CF35BDEFB7BD9441490269A77D7337433E	2020-08-15 17:15:00.000
4	[redacted]	o-57bo0a320czmd6v4d	67.54.159.145		172.31.1.117	22	IPv4	TCP	57A2A51EC7C18629216CF0D381BC5E1E28A9CA60	2020-08-15 17:15:00.000
5	[redacted]	o-57bo0a320czmd6v4d	192.35.168.219		172.31.1.117	22	IPv4	TCP	3B5033CF35BDEFB7BD9441490269A77D7337433E	2020-08-15 16:30:00.000
6	[redacted]	o-57bo0a320czmd6v4d	77.37.222.242		172.31.1.117	22	IPv4	TCP	3B5033CF35BDEFB7BD9441490269A77D7337433E	2020-08-15 16:45:00.000

These are bastion hosts that are used by the system administrators and can be ignored. The image also shows two servers connecting to this application on port 80, which are in scope of a planned migration. At this stage, you would need to dive deeper and understand the connecting applications. This deeper analysis will allow you to assess if there will be any upstream impact after retirement.

Schedule a controlled stop

In your migration plan, be sure to schedule time for a controlled stop during the migration process. A controlled stop pauses the migration process to identify the potential for disruption if an application is retired. It simulates the application's retirement, and allows you to observe the consequences. When the controlled stop period is completed, the migration can easily resume.

The controlled stop approach varies depending on the type of application and the associated processes you are working with. Common controlled stop patterns include:

- Implementing a host-based firewall to block all traffic, which simulates retirement
- Pausing a virtual machine
- Stopping a service on the host
- Blocking all traffic by using an external firewall

The migration project and application owners need to define the duration of a controlled stop, depending on the application type. For example, if you're retiring a batch-based workload that only runs once a month or once a quarter, performing a one-week controlled stop might not be enough to determine the impact on other systems.

Continuing with the example from the previous section, another application was connecting to the server that was scheduled for retirement. An initial assessment concluded that there should not be an impact on the upstream servers. A controlled stop can now be carried out to understand the impact.

This controlled stop would be carried out by implementing a host-based firewall to block all traffic, simulating the effect of shutting down the server. If this causes service issues for applications that are scheduled to be migrated to the AWS Cloud, a firewall rule is added and all traffic resumes. After the controlled stop, the server's retirement is reconsidered due to this service degradation or interruption.

Reassess if the application should be migrated

The last two best practices we discussed help determine if it's appropriate to continue action on the systems that are scheduled for retirement. If those best practices highlight a potential upstream business impact, you should consider reassessing the application's migration pattern. By beginning the application retirement process early, you now have sufficient time to include the application in a subsequent migration wave if you encounter issues or dependencies that mean it cannot be retired.

If, after following those best practices, you are not fully confident in retiring the application, consider if it should be rehosted to the AWS Cloud. This is particularly important if your migration has a set end date, such as a data center lease expiry.

Services such as AWS Application Migration Service simplify the rehost migration approach. When you migrate applications to AWS, you could take a daily snapshot of the Amazon Elastic Block Store (Amazon EBS) volumes and terminate the Amazon Elastic Compute Cloud (Amazon EC2) instance to reduce costs and test the retirement of the application over an extended period of time. If an impact or issue arises, you then have the confidence of being able to create the EBS volumes based on the snapshot to resume the EC2 instance.

Important
Test this recovery process before you terminate the EC2 instance.

Retire the application

After following this guide's previous five best practices, you might have determined that it's safe to retire an application. You deployed a migration factory approach, began the retirement process early, used data and discovery tools to monitor the inbound connections, performed a successful controlled stop, and assessed if the application should be retired. Retiring the application is now possible as part of your migration strategy.

At this point, you should check if the application contains data that might be useful in the future. Machine learning (ML) and analytics have given data greater value than ever before. Although you might not be developing ML algorithms now, historical data can prove beneficial in the future. You might also have regulatory or compliance requirements to store the data for a defined period of time, even if the application has been retired.

AWS offers a comprehensive set of cloud storage services for long-term retention, compliance, and digital preservation. AWS storage solutions for data archiving help provide unlimited scale, 99.999999999% durability, data reliability, and data security.

To assist your compliance efforts, AWS regularly achieves third-party validation for thousands of global compliance requirements. These are continually monitored to help you meet security and compliance standards for finance, retail, healthcare, government, and beyond.

For more information on data archiving with AWS, see Data Archiving on the AWS website.

Conclusion

This guide outlined six best practices to consider when retiring applications in your migration plan. They provide a framework for you to use when evaluating the applications scheduled for retirement and their potential impact.

Although this framework is not exhaustive and you'll want to embed processes that are specific to your organization (such as tracking the retired infrastructure in your configuration management database), it will help manage this migration workstream.

Following these guidelines helps you gain confidence and earn trust in retiring applications, while providing enough time to migrate applications that have been incorrectly categorized.

Tools

- AWS Application Discovery Service
- AWS Application Migration Service

Frequently asked questions

This section provides answers to commonly raised questions about retiring applications.

My applications are containerized. Can I still adopt a data-driven approach to migration?

Yes, by using the port and IP address to distinguish between different containers.

How long should I run discovery tooling?

The duration depends on an individual application. Four weeks should be enough. However, for batch applications that run only once a quarter, you'll need to plan accordingly.

Additional resources

AWS resources

- Mobilizing your organization to accelerate large-scale migrations
- Migrating with AWS
- Migrating to AWS: Best Practices and Strategies
- AWS Prescriptive Guidance
- AWS Migration Whitepaper

AWS services

- Amazon Athena
- AWS Application Discovery Service
- AWS Application Migration Service
- AWS - Data Archiving
- Amazon EBS
- Amazon EC2

AWS Partners

- Tool options for different migration phases

AWS Prescriptive Guidance glossary

AI and ML terms (p. 11) | Migration terms (p. 12) | Modernization terms (p. 16)

AI and ML terms

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

binary classification	A process that predicts a binary outcome (one of two possible classes). For example, your ML model might need to predict problems such as “Is this email spam or not spam?" or "Is this product a book or a car?"
classification	A categorization process that helps generate predictions. ML models for classification problems predict a discrete value. Discrete values are always distinct from one another. For example, a model might need to evaluate whether or not there is a car in an image.
data preprocessing	To transform raw data into a format that is easily parsed by your ML model. Preprocessing data can mean removing certain columns or rows and addressing missing, inconsistent, or duplicate values.
deep ensemble	To combine multiple deep learning models for prediction. You can use deep ensembles to obtain a more accurate prediction or for estimating uncertainty in predictions.
deep learning	An ML subfield that uses multiple layers of artificial neural networks to identify mapping between input data and target variables of interest.
exploratory data analysis (EDA)	The process of analyzing a dataset to understand its main characteristics. You collect or aggregate data and then perform initial investigations to find patterns, detect anomalies, and check assumptions. EDA is performed by calculating summary statistics and creating data visualizations.
features	The input data that you use to make a prediction. For example, in a manufacturing context, features could be images that are periodically captured from the manufacturing line.
feature 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.
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.
multiclass classification	A process that helps generate predictions for multiple classes (predicting one of more than two outcomes). For example, an ML model might ask "Is this product a book, car, or phone?" or "Which product category is most interesting to this customer?"
regression	An ML technique that predicts a numeric value. For example, to solve the problem of "What price will this house sell for?" an ML model could use a linear regression model to predict a house's sale price based on known facts about the house (for example, the square footage).
training	To provide data for your ML model to learn from. The training data must contain the correct answer. The learning algorithm finds patterns in the training data that map the input data attributes to the target (the answer that you want to predict). It outputs an ML model that captures these patterns. You can then use the ML model to make predictions on new data for which you don’t know the target.
target variable	The value that you are trying to predict in supervised ML. This is also referred to as an *outcome variable*. For example, in a manufacturing setting the target variable could be a product defect.
tuning	To change aspects of your training process to improve the ML model's accuracy. For example, you can train the ML model by generating a labeling set, adding labels, and then repeating these steps several times under different settings to optimize the model.
uncertainty	A concept that refers to imprecise, incomplete, or unknown information that can undermine the reliability of predictive ML models. There are two types of uncertainty: *Epistemic uncertainty* is caused by limited, incomplete data, whereas *aleatoric uncertainty* is caused by the noise and randomness inherent in the data. For more information, see the Quantifying uncertainty in deep learning systems guide.

Migration terms

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

7 Rs	Seven common migration strategies for moving applications to the cloud. These strategies build upon the 5 Rs that Gartner identified in 2011 and consist of the following: • Refactor/re-architect – Move an application and modify its architecture by taking full advantage of cloud-native features to improve agility, performance, and scalability. This typically involves porting the operating system and database. Example: Migrate your on-premises Oracle database to the Amazon Aurora PostgreSQL-Compatible Edition.

	• Replatform (lift and reshape) – Move an application to the cloud, and introduce some level of optimization to take advantage of cloud capabilities. Example: Migrate your on-premises Oracle database to Amazon Relational Database Service (Amazon RDS) for Oracle in the AWS Cloud. • Repurchase (drop and shop) – Switch to a different product, typically by moving from a traditional license to a SaaS model. Example: Migrate your customer relationship management (CRM) system to Salesforce.com. • Rehost (lift and shift) – Move an application to the cloud without making any changes to take advantage of cloud capabilities. Example: Migrate your on-premises Oracle database to Oracle on an EC2 instance in the AWS Cloud. • Relocate (hypervisor-level lift and shift) – Move infrastructure to the cloud without purchasing new hardware, rewriting applications, or modifying your existing operations. This migration scenario is specific to VMware Cloud on AWS, which supports virtual machine (VM) compatibility and workload portability between your on-premises environment and AWS. You can use the VMware Cloud Foundation technologies from your on-premises data centers when you migrate your infrastructure to VMware Cloud on AWS. Example: Relocate the hypervisor hosting your Oracle database to VMware Cloud on AWS. • Retain (revisit) – Keep applications in your source environment. These might include applications that require major refactoring, and you want to postpone that work until a later time, and legacy applications that you want to retain, because there's no business justification for migrating them. • Retire – Decommission or remove applications that are no longer needed in your source environment.
application portfolio	A collection of detailed information about each application used by an organization, including the cost to build and maintain the application, and its business value. This information is key to the portfolio discovery and analysis process and helps identify and prioritize the applications to be migrated, modernized, and optimized.
artificial intelligence operations (AIOps)	The process of using machine learning techniques to solve operational problems, reduce operational incidents and human intervention, and increase service quality. For more information about how AIOps is used in the AWS migration strategy, see the operations integration guide.
AWS Cloud Adoption Framework (AWS CAF)	A framework of guidelines and best practices from AWS to help organizations develop an efficient and effective plan to move successfully to the cloud. AWS CAF organizes guidance into six focus areas called perspectives: business, people, governance, platform, security, and operations. The business, people, and governance perspectives focus on business skills and processes; the platform, security, and operations perspectives focus on technical skills and processes. For example, the people perspective targets stakeholders who handle human resources (HR), staffing functions, and people management. For this perspective, AWS CAF provides guidance for people development, training, and communications to help ready the organization for successful cloud adoption. For more information, see the AWS CAF website and the AWS CAF whitepaper.
AWS landing zone	A landing zone is a well-architected, multi-account AWS environment that is scalable and secure. This is a starting point from which your organizations can quickly launch and deploy workloads and applications with confidence in their security and infrastructure environment. For more information about landing zones, see Setting up a secure and scalable multi-account AWS environment.
AWS Workload Qualification Framework (AWS WQF)	A tool that evaluates database migration workloads, recommends migration strategies, and provides work estimates. AWS WQF is included with AWS Schema

	Conversion Tool (AWS SCT). It analyzes database schemas and code objects, application code, dependencies, and performance characteristics, and provides assessment reports.
business continuity planning (BCP)	A plan that addresses the potential impact of a disruptive event, such as a large-scale migration, on operations and enables a business to resume operations quickly.
Cloud Center of Excellence (CCoE)	A multi-disciplinary team that drives cloud adoption efforts across an organization, including developing cloud best practices, mobilizing resources, establishing migration timelines, and leading the organization through large-scale transformations. For more information, see the CCoE posts on the AWS Cloud Enterprise Strategy Blog.
cloud stages of adoption	The four phases that organizations typically go through when they migrate to the AWS Cloud: • Project – Running a few cloud-related projects for proof of concept and learning purposes • Foundation – Making foundational investments to scale your cloud adoption (e.g., creating a landing zone, defining a CCoE, establishing an operations model) • Migration – Migrating individual applications • Re-invention – Optimizing products and services, and innovating in the cloud These stages were defined by Stephen Orban in the blog post The Journey Toward Cloud-First & the Stages of Adoption on the AWS Cloud Enterprise Strategy blog. For information about how they relate to the AWS migration strategy, see the migration readiness guide.
configuration management database (CMDB)	A database that contains information about a company's hardware and software products, configurations, and inter-dependencies. You typically use data from a CMDB in the portfolio discovery and analysis stage of migration.
epic	In agile methodologies, functional categories that help organize and prioritize your work. Epics provide a high-level description of requirements and implementation tasks. For example, AWS CAF security epics include identity and access management, detective controls, infrastructure security, data protection, and incident response. For more information about epics in the AWS migration strategy, see the program implementation guide.
heterogeneous database migration	Migrating your source database to a target database that uses a different database engine (for example, Oracle to Amazon Aurora). Heterogeneous migration is typically part of a re-architecting effort, and converting the schema can be a complex task. AWS provides AWS SCT that helps with schema conversions.
homogeneous database migration	Migrating your source database to a target database that shares the same database engine (for example, Microsoft SQL Server to Amazon RDS for SQL Server). Homogeneous migration is typically part of a rehosting or replatforming effort. You can use native database utilities to migrate the schema.
idle application	An application that has an average CPU and memory usage between 5 and 20 percent over a period of 90 days. In a migration project, it is common to retire these applications or retain them on premises.
IT information library (ITIL)	A set of best practices for delivering IT services and aligning these services with business requirements. ITIL provides the foundation for ITSM.

IT service management (ITSM)	Activities associated with designing, implementing, managing, and supporting IT services for an organization. For information about integrating cloud operations with ITSM tools, see the operations integration guide.
large migration	A migration of 300 or more servers.
Migration Acceleration Program (MAP)	An AWS program that provides consulting support, training, and services to help organizations build a strong operational foundation for moving to the cloud, and to help offset the initial cost of migrations. MAP includes a migration methodology for executing legacy migrations in a methodical way and a set of tools to automate and accelerate common migration scenarios.
Migration Portfolio Assessment (MPA)	An online tool that provides information for validating the business case for migrating to the AWS Cloud. MPA provides detailed portfolio assessment (server right-sizing, pricing, TCO comparisons, migration cost analysis) as well as migration planning (application data analysis and data collection, application grouping, migration prioritization, and wave planning). The MPA tool (requires login) is available free of charge to all AWS consultants and APN Partner consultants.
Migration Readiness Assessment (MRA)	The process of gaining insights about an organization's cloud readiness status, identifying strengths and weaknesses, and building an action plan to close identified gaps, using the AWS CAF. For more information, see the migration readiness guide. MRA is the first phase of the AWS migration strategy.
migration at scale	The process of moving the majority of the application portfolio to the cloud in waves, with more applications moved at a faster rate in each wave. This phase uses the best practices and lessons learned from the earlier phases to implement a *migration factory* of teams, tools, and processes to streamline the migration of workloads through automation and agile delivery. This is the third phase of the AWS migration strategy.
migration factory	Cross-functional teams that streamline the migration of workloads through automated, agile approaches. Migration factory teams typically include operations, business analysts and owners, migration engineers, developers, and DevOps professionals working in sprints. Between 20 and 50 percent of an enterprise application portfolio consists of repeated patterns that can be optimized by a factory approach. For more information, see the discussion of migration factories and the CloudEndure Migration Factory guide in this content set.
migration metadata	The information about the application and server that is needed to complete the migration. Each migration pattern requires a different set of migration metadata. Examples of migration metadata include the target subnet, security group, and AWS account.
migration pattern	A repeatable migration task that details the migration strategy, the migration destination, and the migration application or service used. Example: Rehost migration to Amazon EC2 with AWS Application Migration Service.
migration strategy	The approach used to migrate a workload to the AWS Cloud. For more information, see the 7 Rs (p. 12) 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

	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.
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.

update-history-change	update-history-description	update-history-date
Updated name of AWS solution (p. 19)	We updated the name of the referenced AWS solution from *CloudEndure Migration Factory* to *Cloud Migration Factory*.	May 6, 2022
Refreshed content (p. 19)	Updated guide based on new AWS services.	January 28, 2022
Initial publication (p. 19)	—	September 11, 2020