AWS Prescriptive Guidance

Guide for AWS large migrations

AWS Prescriptive Guidance: Guide for AWS large migrations

Table of Contents

Guide for AWS large migrations

Amazon Web Services (AWS)

February 2022 (last update (p. 27): May 2022)

Many Amazon Web Services (AWS) customers want to migrate a large number of servers to the AWS Cloud as fast as possible, such as migrating 1,000 servers within 6 months. Migrating 300 or more servers is considered a large migration. This is not an easy task because the people, process, and technology challenges of a large migration project are typically new to an enterprise.

As a team specializing in large migrations, we have learned a lot from our customers' experiences. The aim of our content is to help you apply the correct strategy and best practices from the outset, applying lessons learned and streamlining your journey.

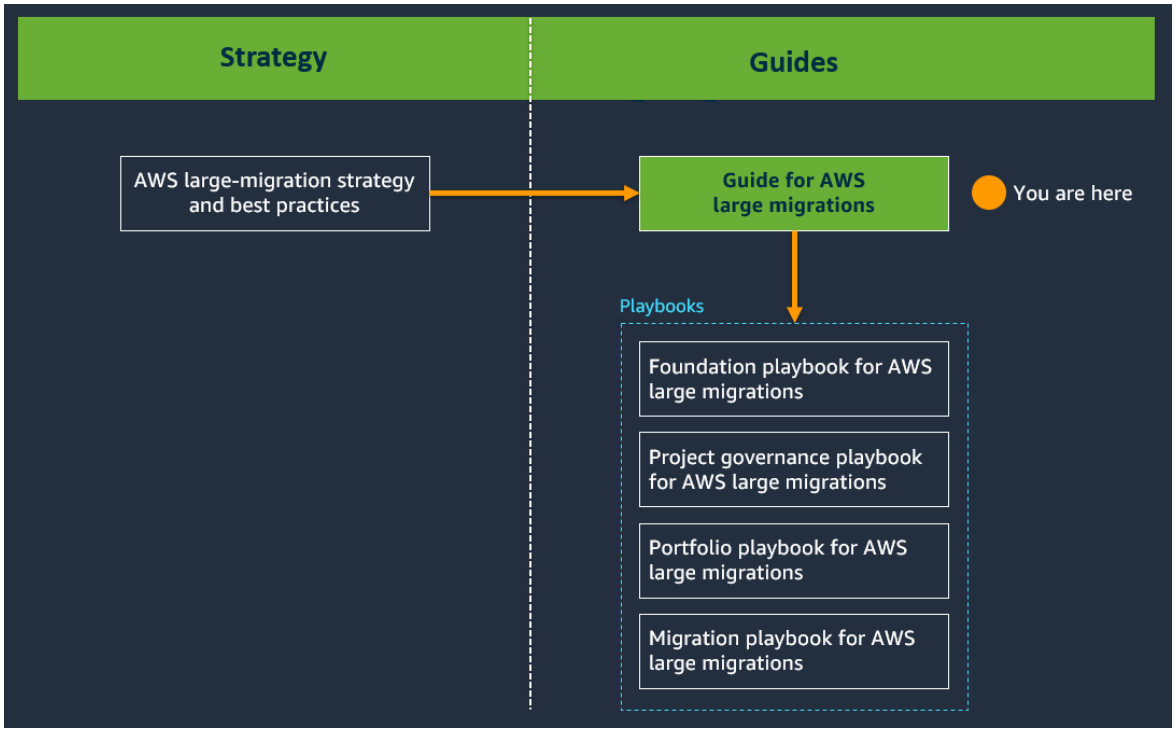
About this guide

This guide is the second document in a series about large migrations to the AWS Cloud. If you haven't already done so, we highly recommend reading AWS large-migration strategy and best practices. The strategy document discusses best practices for large migrations and provides use cases from customers across various industries.

This guide describes a high-level, phased approach for implementing the best practices outlined in the strategy document. Details of the stages and steps are described in the corresponding playbooks within this document series. We recommend reading the playbooks in the following order:

1. Foundation playbook for AWS large migrations
2. Project governance playbook for AWS large migrations
3. Portfolio playbook for AWS large migrations
4. Migration playbook for AWS large migrations

The following figure shows the structure of the AWS documentation series for large migrations. Review the strategy first, then this guide, and then proceed to the playbooks.



About the tools

This guide includes a health-check matrix. You can use this tool to assess the health of your migration project throughout the migration. This tool helps you apply best practices and evaluate the efficiency and progress of your large migration project throughout its life cycle.

Additionally, each of the playbooks in this documentation series include templates and tools that can help you build each workstream.

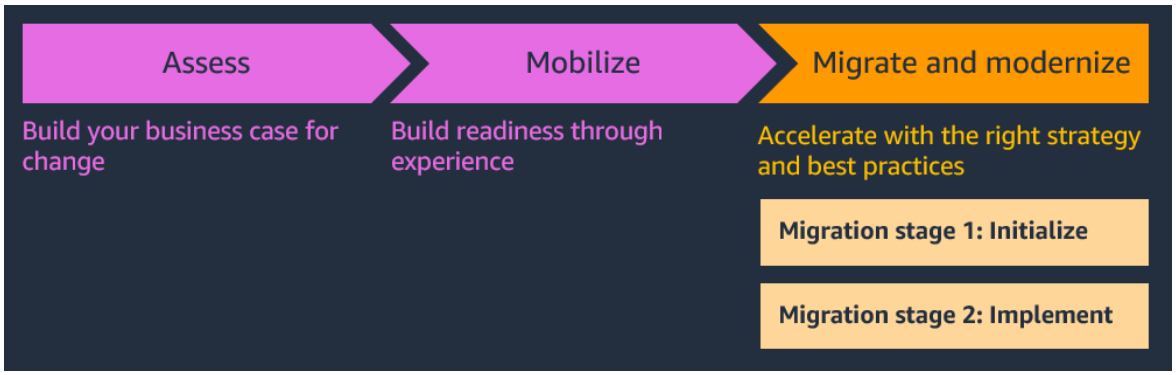
Phases of a large migration

AWS segments the large migration process into three sequential phases:

1. Assess phase (p. 3) – You build the business case for the migration.
2. Mobilize phase (p. 3) – You prepare the organization and mobilize the resources needed for the migration.
3. Migrate and modernize phase of a large migration project (p. 4) – You use your strategy, plan, and the best practices to migrate and modernize. In this phase, migration is divided into two stages, initialize and implement.

The assess and mobilize phases are the foundation of any large migration and prepare the organization for the migration. This guide includes a brief summary of the first two phases and provides references for additional support. The primary purpose of this guide is to describe the migrate part of the third phase, migrate and modernize, and its two stages, initialize and implement. We recommend that you first migrate to the AWS Cloud and then modernize the workload. Modernization is not included in this documentation set for large migrations. For more information about modernization, see Phased approach to modernizing applications in the AWS Cloud on the AWS Prescriptive Guidance website.

The following figure shows the three phases of a large migration: assess, mobilize, and migrate and modernize. Migration is divided into two stages, initialize and implement.



Even though this document set for large migration is focused on the migrate phase, we highly recommend you go through all three phases of a large migration in sequence. Completing the assess and mobilize phases builds a solid foundation to support the migration.

Assess phase

This is the start of your large-migration journey. In this phase, you build your business case for the large migration. You assess your organization's readiness for cloud transformation, take an initial look at your portfolio, and confirm that the key stakeholders in your organization are aligned.

For more information about the assess phase, see Evaluating migration readiness on the AWS Prescriptive Guidance website.

Mobilize phase

Now that you have built your business case for a cloud transformation, you use the readiness assessment from the previous phase and begin to fill in the gaps. You mobilize your organization and set up

workstreams that prepare your organization for the large migration in the next phase. In this phase, you typically build your AWS landing zone, conduct a more thorough portfolio assessment, build your security and operating model, and prepare teams for change.

For more information about the mobilize phase, see Mobilize your organization to accelerate large-scale migrations on the AWS Prescriptive Guidance website.

Migrate and modernize phase of a large migration project

After the mobilize phase, your organization should have a solid foundation and be ready to start migration and modernization. Before proceeding, you evaluate the foundations, including the people and the platform, and make sure they are ready to support a large migration. The key for this step is to build standard operating procedures and automations that simplify and accelerate a repeatable pattern. This approach is known as a *migration factory*.

Just like a manufacturing factory, when you build a migration factory, you must initialize and calibrate it, define the standard operating procedures, measure the factory's performance, and continually improve the process and tools. In a large migration, initialization is a critical first stage. Running and improving the factory is the second stage, implementation, which migrates your servers at scale in batches known as waves. You organize your resources into workstreams, which are focused on accomplishing one aspect of the large migration.

This section consists of the following topics:

About workstreams

Each workstream in your large migration project is dedicated to completing certain tasks, and those tasks change as you progress from stage 1, initialization, into stage 2, implementation. Although each workstream is independent, they work together to accomplish the same goal—migrate servers at scale.

The following are the four core workstreams, and you can create additional, supporting workstreams as needed to support your use case:

- **Foundation workstream** – This workstream is focused on preparing the people and platform for the large migration.
- **Project governance workstream** – This workstream manages the overall migration project, facilitates communication, and focuses on completing the project within budget and on time.
- **Portfolio workstream** – The teams in this workstream collect metadata to support the migration, prioritize applications, and perform wave planning.
- **Migration workstream** – Using the wave plan and collected metadata from the portfolio workstream, the teams in this workstream migrate and cutover the applications and servers.

For more information and instructions about how to establish and resource the workstreams in your large migration project, see the Foundation playbook for AWS large migrations.

About the initialize and implement stages

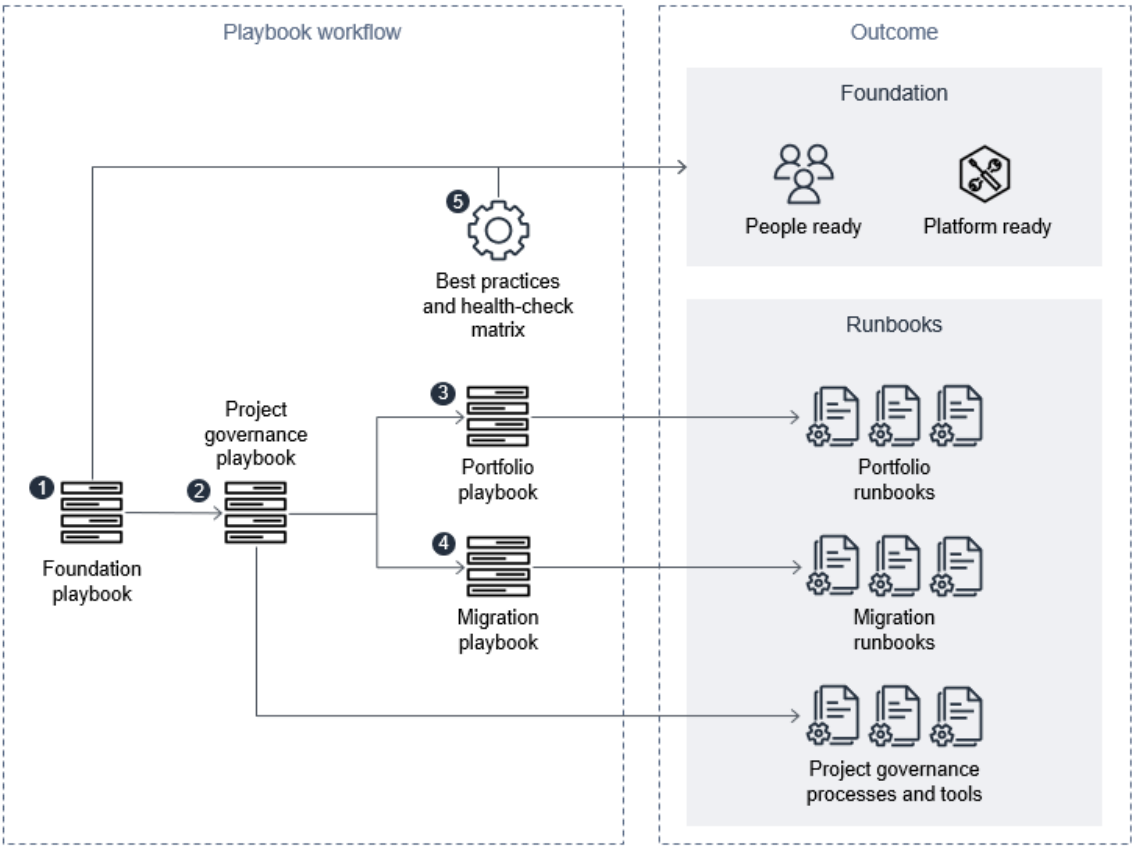
Migrate and modernize is the third and final phase of your migration journey. Generally, you migrate first and then modernize. Migrate consists of two stages:

- **Stage 1: Initialize a large migration** – You prepare your platform and people for a large migration. For example, you review the landing zone design and verify that it has enough bandwidth to support the scale, and you design and implement a training plan. In this stage, based on your organization's policies and processes, you also define the standard operating procedures (or *runbooks*) for the large migration. The runbooks and automations simplify and accelerate implementation of the large migration in stage 2.
- **Stage 2: Implement a large migration** – In this stage, you migrate servers at scale by using the runbooks defined in stage 1. You manage the migration factory with project governance tools, monitor the migration with a health-check matrix, and continuously improve the runbooks in order to increase the velocity of the migration.

Stage	Duration	Purpose
Stage 1: Initialize	1–3 months	• Prepare your platform and people for a large migration. • Build your standard operating procedures (runbooks).
Stage 2: Implement	Varies by project scope and strategy	• Use runbooks to implement the large migration. • Manage, monitor, and improve the migration.

Stage 1: Initializing a large migration

In the initialize stage, it is critical to define the standard operating procedures, or *runbooks*, for most of the migration tasks. Because there are many people, processes, and tools involved in a large migration, you can improve efficiency by defining, connecting, and automating procedures in a runbook. Runbooks provide clear guidance that everyone must follow. Implementing a large migration without runbooks increases the complexity of the migration and might cause the migration effort to lose momentum or fail. To help you get started, this guide reviews the workflow and playbooks, which contain useful templates for building your own runbooks.



The playbooks help you complete the first stage, *initialize*, and prepare for the second stage, *implement*:

1. Foundation playbook for AWS large migrations – This playbook helps you prepare your infrastructure and people.
2. Project governance playbook for AWS large migrations – This playbook helps you build project governance processes and tools that effectively manage a large migration project. It includes templates to help you get started.
3. Portfolio playbook for AWS large migrations – This playbook helps you build your own portfolio runbooks for wave planning and portfolio assessment. It includes templates to help you get started.
4. Migration playbook for AWS large migrations – This playbook helps you build a runbook for each migration pattern. It includes several templates for different patterns.
5. Health-check matrix for AWS large migrations – This helps you apply best practices and evaluate the efficiency and progress of your large migration throughout the life cycle of the project.

Step 1: Use the foundation playbook to get ready

In this playbook, you get your platform and people ready.

The following are examples of how you might prepare the people in your organization:

- Make sure your stakeholders are committed to the migration and ready to provide support.
- Define the workstreams in your large migration.
- Define your Cloud Enablement Engine, also known as Cloud Center of Excellence, team.
- Create a responsibility assignment model, in the form of a responsible, accountable, consulted, informed (RACI) matrix.
- Establish a training plan for your migration resources.

The following are examples of how you might prepare your organization's platform:

- Get the AWS landing zone ready to support a large migration.
- Evaluate the readiness of your on-premises infrastructure.

For more information, see the Foundation playbook for AWS large migrations.

Step 2: Use the project governance playbook to define the rules

In this step, you define the rules, boundaries, and plans for running the large migration. This includes defining the following plans and processes:

- Communication plan
- Benefit-tracking office
- Escalation process
- Migration and cutover communication gates
- Project management processes and tools

For more information, see the Project governance playbook for AWS large migrations.

Step 3: Use the portfolio playbook to create portfolio runbooks

Now, you get a better understanding of the scope and strategy. In this step, you follow the portfolio playbook to build detailed runbooks for portfolio assessment and wave planning. The goal is to make sure the migration factory doesn't run out of raw materials (servers) and has all the information needed to support the migration. Tasks include the following:

- Review the migration strategy from the mobilize phase (if you defined it in that phase). For more information, see About the migration strategies (p. 12).
- Define metadata requirements for each migration pattern.

- Define the metadata collection process and a metadata storage location that serves as a single source of truth.
- Define application prioritization rules.
- Define application deep dive rules.
- Define the wave planning process.

For more information, see the Portfolio playbook for AWS large migrations.

Step 4: Use the migration playbook to build a migration runbook for each pattern

This step can run in parallel with the previous step. Using the migration playbook, you build your runbook for each of the migration patterns, such as rehosting to Amazon Elastic Compute Cloud (Amazon EC2) with AWS Cloud Migration Factory, or replatforming storage to Amazon Elastic File System (Amazon EFS) with AWS DataSync. You repeatedly test the runbooks until there are minimal to no errors. Tasks include the following:

- Understand the migration patterns.
- Get the runbook templates and add your on-premises process and tools.
- Test and improve the runbooks.
- Automate the runbooks as much as possible.

For more information, see the Migration playbook for AWS large migrations.

Step 5: Use the health-check matrix to evaluate your current state

In this final step, you verify that your project team has completed the previous steps and identify any remaining gaps. Evaluating the health of the migration and improving the process are critical to the migration's success and helps maintain alignment with best practices.

Use the health-check matrix to evaluate the current state of your migration from a people, process, and technology perspective. This is not a one-off process. You should perform this evaluation on a regular basis.

Stage 2: Implementing a large migration

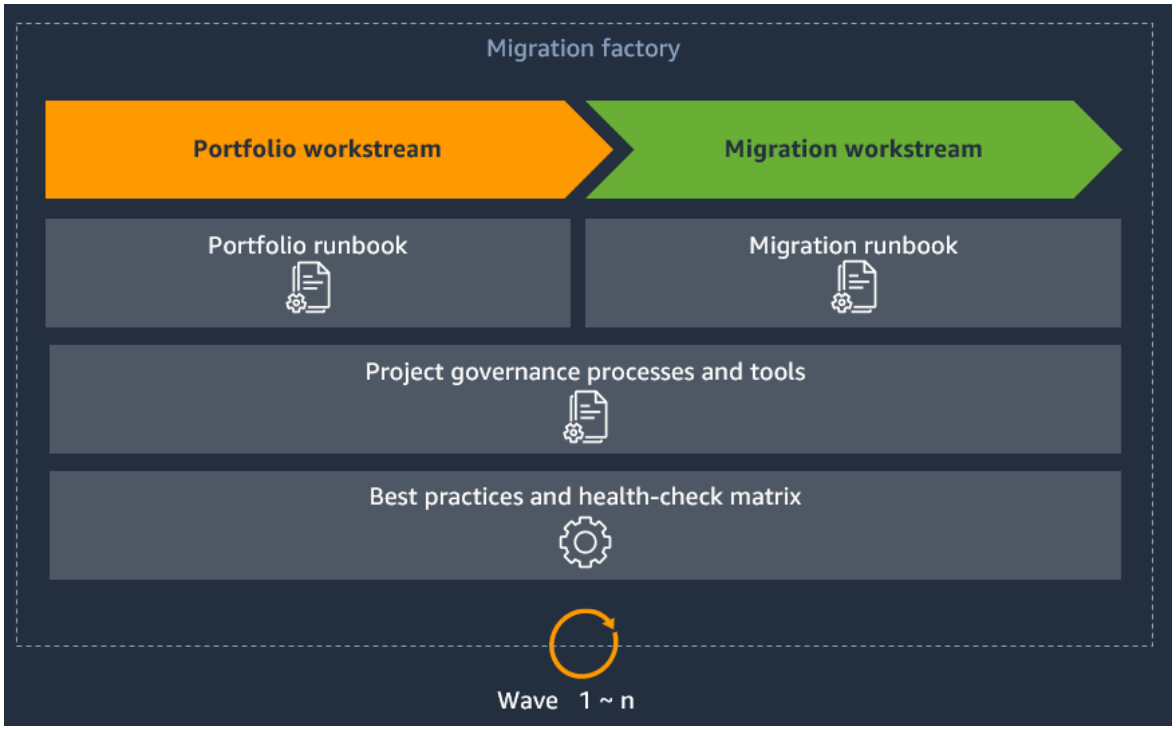
In stage 2 of a large migration, the goal is to migrate your servers at scale. For example, to migrate 1,000 servers in 6 months, you might start by migrating 5 servers per week and then gradually increase the velocity up to 50–100 servers per week.

Now you use the runbooks that you developed in stage 1 to migrate servers in waves. The first couple of waves are typically small because the migration and portfolio workstreams are adopting and adjusting the processes in their runbooks. Improving the runbooks is key to the success of a large migration. Runbooks are living documents. You must review, revise, and improve your runbooks after each cutover. As the runbooks evolve over time, the velocity should increase with every wave.

In stage 2, you use the following components to operate the migration factory:

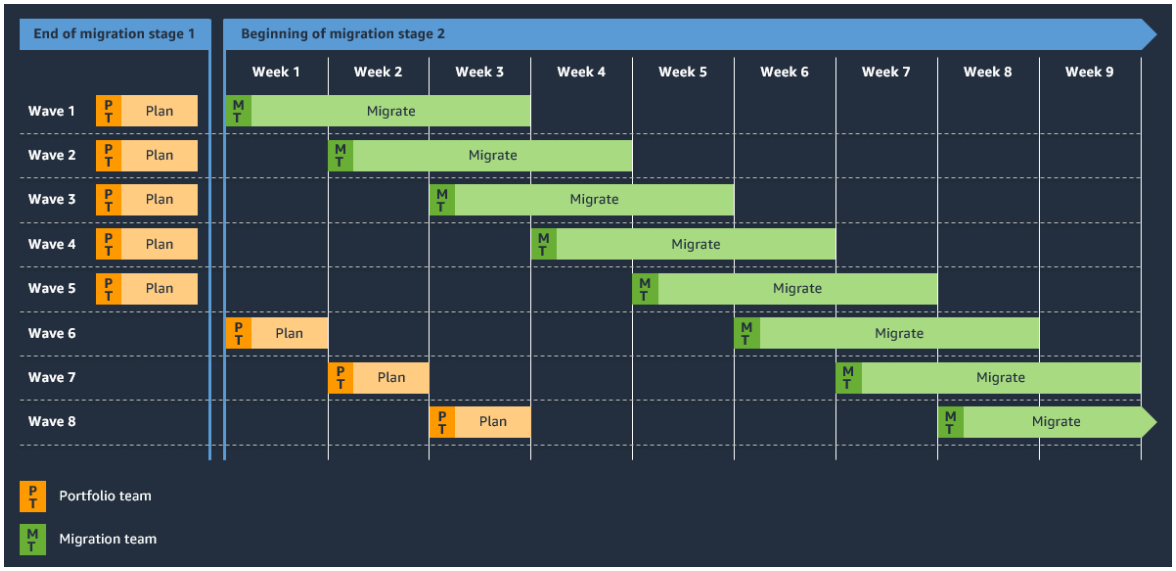
- **Project governance rules** – You follow the project governance processes to manage waves, communication, timelines, and cutovers. These processes and tools make sure that everyone does the right thing at the right time and in the right order.
- **Portfolio runbooks** – You use the portfolio runbooks to prioritize applications, plan waves, and collect the required metadata that supports the migration. This metadata is the equivalent of raw materials in a manufacturing factory.
- **Migration runbooks** – You use the migration runbooks to migrate apps and servers, load the metadata to your migration tools, and complete the cutover process at the end of each wave. When following the migration runbooks, you adhere to the wave plan in the portfolio runbooks and use the metadata in the portfolio runbooks or from another single source of truth.
- **Large migration best practices and health-check matrix** – You use the health-check matrix to evaluate your current state frequently and regularly to make sure that everything is on track.

The following figure shows a typical migration factory for large migrations.



The runbooks are the key components of the migration factory, and they work together to form a data flow through two workstreams, portfolio and migration. For more information about these workstreams, see the Foundation playbook for AWS large migrations. Rather than seeing a wave all the way through the migration factory, teams are usually dedicated to certain parts of the factory, and the waves flow through each workstream. The duration of each workstream varies based on your project timeline, scope, and resource availability. For example, the portfolio workstream might be 3 weeks, and the migration workstream might be 2–5 weeks. Prevent supply chain problems in your migration factory by ensuring that there are sufficient server waves lined up for migration. We recommend that the portfolio workstream is five waves ahead of the migration workstream.

The following figure shows a dynamic view of a typical migration factory. For each wave, the portfolio workstream runs 1–2 weeks, and the migration workstream typically runs 3–4 weeks. The portfolio workstream is five waves ahead of the migration workstream, so there is always a five-wave buffer between the portfolio and migration workstreams. At the end of migration stage 1, initializing, the portfolio workstream completes wave planning for a buffer of five waves. When the migration workstream starts migrating applications, this indicates that you have entered stage 2, implementing. Both the portfolio and the migration workstreams continue to process waves, and the buffer prevents the migration workstream from running out of servers to migrate.


End of migration stage 1
Beginning of migration stage 2
Week 1
Week 2
Week 3
Week 4
Week 5
Week 6
Week 7
Week 8
Week 9
Wave 1
Wave 2
Wave 3
Wave 4
Wave 5
Wave 6
Wave 7
Wave 8
PT
Plan
MT
Migrate
Portfolio team
Migration team

About the migration strategies

A *migration strategy* is the approach used to migrate a workload into the AWS Cloud. There are seven migration strategies for moving applications to the cloud, known as the *7 Rs*:

Common strategies for large migrations include rehost, replatform, relocate, and retire. Refactor is not recommended for large migrations because it involves modernizing the application during the migration. This is the most complex of the migration strategies, and it can be complicated to manage for a large number of applications. Instead, we recommend rehosting, relocating, or replatforming the application and then modernizing the application after the migration is complete.

Selecting migration strategies is critical to a large migration. You might have selected migration strategies in the mobilize phase or during the initial portfolio assessment. This section reviews each migration strategy and their common use cases.

Retire

This is the migration strategy for the applications that you want to decommission or archive. Retiring the application means that you can shut down the servers within that application stack. The following are common use cases for the retire strategy:

- There is no business value in retaining the application or moving it to cloud.
- You want to eliminate the cost of maintaining and hosting the application.
- You want to reduce the security risks of operating an application that uses an operating system (OS) version or components that are no longer supported.
- You might want to retire applications based on their performance. For example, you might want to retire applications that have an average CPU and memory usage below 5 percent, known as *zombie applications*. You might also choose to retire some applications that have an average CPU and memory usage between 5 and 20 percent over a period of 90 days, known as *idle applications*. You can use the utilization and performance data from your discovery tool to identify zombie and idle applications.
- There has been no inbound connection to the application for the last 90 days.

For more information, see Best practices for assessing applications to be retired during a migration to the AWS Cloud.

Retain

This is the migration strategy for applications that you want to keep in your source environment or applications that you are not ready to migrate. You might choose to migrate these applications in the future.

The following are common use cases for the retain strategy:

- **Security and compliance** – You might want to retain applications in order to remain in compliance with data residency requirements.
- **High risk** – You might decide to retain an application because it requires a detailed assessment and plan prior to migration.
- **Dependencies** – You might decide to retain an application if you need to migrate one or more other applications first.
- **Applications that are recently upgraded** – You might want to postpone migrating the application until the next technical refresh because you recently invested in upgrading your current system.
- **No business value to migrate** – There is no business value for migrating some applications to the cloud, such as those with only a few internal users.
- **Plans to migrate to software as a service (SaaS)** – You might choose retain an application until the SaaS version is released by the vendor. This is a common strategy for vendor-based applications.
- **Unresolved physical dependencies** – You might choose to retain an application that is dependent on specialized hardware that does not have a cloud equivalent, such as machines in a manufacturing plant.
- **Mainframe or mid-range applications and non-x86 Unix applications** – These applications require careful assessment and planning before migrating them to the cloud. Examples of mid-range applications include IBM AS/400 and Oracle Solaris.
- **Performance** – You might want to retain applications based on their performance. For example, you might want to keep zombie or idle applications in your source environment.

Rehost

This strategy is also known as *lift and shift*. Using this strategy, you move your applications from your source environment to the AWS Cloud without making any changes to the application. For example, you migrate your application stack from on-premises to the AWS Cloud.

With rehost, you can migrate a large number of machines from multiple source platforms (physical, virtual, or another cloud) to the AWS Cloud without worrying about compatibility, performance disruption, long cutover windows, or long-distance data replications.

Your application continues to serve users while the workloads are being migrated, which minimizes disruption and downtime. The downtime depends on your cutover strategy.

This strategy helps you to scale your applications without implementing any cloud optimizations that could save you time or money. Applications are easier to optimize or re-architect when they are already running in cloud because it is easier to integrate to AWS services and manage your workloads.

You can automate rehosting by using the following services:

- AWS Application Migration Service
- AWS Cloud Migration Factory Solution
- VM Import/Export

For a list of migration patterns for the rehost migration strategy, see Rehost on the AWS Prescriptive Guidance website.

Relocate

Using this strategy, you can transfer a large number of servers, comprising one or more applications, at a given time from on-premises platform to a cloud version of the platform. You can also use the relocate strategy to move instances or objects to a different virtual private cloud (VPC), AWS Region, or AWS account. For example, you can use this strategy to transfer servers in bulk from VMware software-defined data center (SSDC) to VMware Cloud on AWS, or you can transfer an Amazon Relational Database Service (Amazon RDS) DB instance to another VPC or AWS account.

The relocate strategy doesn't require that you purchase new hardware, rewrite applications, or modify your existing operation. During relocation, the application continues to serve users, which minimizes disruption and downtime. Relocate is the quickest way to migrate and operate your workload in the cloud because it does not impact the overall architecture of your application.

For a list of migration patterns for the relocate migration strategy, see Relocate on the AWS Prescriptive Guidance website.

Repurchase

This strategy is also known as *drop and shop*. You replace your application with a different version or product. The new application should provide more business value than the existing, on-premises application, including features such as accessibility from anywhere, no infrastructure to maintain, and pay-as-you-go pricing models. Repurchasing the application typically reduces costs associated with maintenance, infrastructure, and licensing.

The following are common use cases for the repurchase migration strategy:

- **Moving from a traditional license to SaaS** – This removes the burden of managing and maintaining the infrastructure and helps reduce licensing issues.
- **Version upgrades or third-party equivalents** – By replacing your existing on-premises application with the vendor's latest version or third-party equivalent in the cloud, you can leverage new features, integrate with cloud services, and scale the application more easily.
- **Replacing a custom application** – You can avoid recoding and re-architecting a custom application by repurchasing a vendor-based SaaS or cloud-based application.

Before purchasing, you need to assess the application according to your business requirements, especially security and compliance.

After you purchase the new application, the following are the next steps:

- Training your team and users with the new system
- Migrating your data to the newly purchased application
- Integrating the application into your authentication services, such as Microsoft Active Directory, to centralize authentication
- Configuring networking to help secure communication between the purchased application, your users, and your infrastructure

Typically, the application vendor helps you with these activities for a smooth transition.

Replatform

This strategy is also known as *lift, tinker, and shift* or *lift and reshape*. Using this migration strategy, you move the application to the cloud, and you introduce some level of optimization in order to operate the application efficiently, to reduce costs, or to take advantage of cloud capabilities. For example, you might replatform a Microsoft SQL Server database to Amazon RDS for SQL Server.

Using this strategy, you might make a few or many changes to the application, depending on your business goals and your target platform.

The following are common use cases for the replatform migration strategy:

- You want to save time and reduce cost by moving to a fully managed service or serverless service in the AWS Cloud.
- You want to improve your security and compliance stance by upgrading your operating system to the latest version. By using End-of-Support Migration Program (EMP) for Windows Server, you can migrate your legacy Windows Server applications to the latest, supported versions of Windows Server on AWS, without any code changes. You can use this decision tree in the *AWS EMP for Windows Server User Guide* to help you determine your EMP workloads.
- You can reduce costs by using AWS Graviton Processors, custom-built processors developed by AWS.
- You can reduce costs by moving from a Microsoft Windows operating system to a Linux operating system. You can port your .NET Framework applications to .NET Core, which can run on a Linux operating system. Porting Assistant for .NET is an analysis tool that helps you port your applications to Linux.
- You can improve performance by migrating virtual machines into containers, without making any code changes. You can modernize your .NET and Java applications into containerized applications by using the AWS App2Container migration tool.

The replatform strategy keeps your legacy application running without compromising security and compliance.

Replatform reduces cost and improves performance by migrating to a managed or serverless service, moving virtual machines to container, and avoiding licensing expenses.

For a list of migration patterns for the replatform migration strategy, see Replatform on the AWS Prescriptive Guidance website.

Refactor or re-architect

Using this strategy, you move an application to the cloud and modify its architecture by taking full advantage of cloud-native features to improve agility, performance, and scalability. This is driven by strong business demand to scale, accelerate product and feature releases, and to reduce costs.

The following are common use cases for the refactor migration strategy:

- The legacy mainframe application can no longer address the demand of the business due to its limitations or is expensive to maintain.
- You have a monolith application that is already hindering efforts to deliver product quickly or address customer needs and demands.
- You have a legacy application that nobody knows how to maintain, or the source code is unavailable.
- The application is difficult to test, or test coverage is very low. This affects the quality and delivery of new application features and fixes. By redesigning the application for the cloud, you can increase the test coverage and integrate automated testing tools.

- For security and compliance reasons, when moving a database to the cloud, you might need to extract some tables (such as customer information, patient, or patient diagnosis tables) and retain those tables on premises. In this situation, you need to refactor your database in order to separate the tables that will be migrated from those that will be retained on premises.

For a list of migration patterns for the refactor migration strategy, see Re-architect on the AWS Prescriptive Guidance website.

Resources

AWS large migrations

Strategy

- AWS large-migration strategy and best practices

Playbooks

To start your large migration journey, we encourage you to read the detailed playbooks in the following order:

1. Foundation playbook for AWS large migrations
2. Project governance playbook for AWS large migrations
3. Portfolio playbook for AWS large migrations
4. Migration playbook for AWS large migrations

Additional references

- AWS Cloud Migration Factory Solution
- When to Choose AWS Application Migration Service
- Best practices for assessing applications to be retired during a migration to the AWS Cloud
- AWS Prescriptive Guidance migration patterns

AWS Prescriptive Guidance glossary

AI and ML terms (p. 18) | Migration terms (p. 19) | Modernization terms (p. 23)

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. 19) 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.

Contributors

The following individuals contributed to this document:

- Wally Lu, Principal Consultant
- Damien Renner, Senior Migration Specialist

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. 27)	We updated the name of the referenced AWS solution from *CloudEndure Migration Factory* to *Cloud Migration Factory*.	May 2, 2022
Initial publication (p. 27)	—	February 28, 2022