AWS Prescriptive Guidance

Migration playbook for AWS large migrations

AWS Prescriptive Guidance: Migration playbook for AWS large migrations

Table of Contents

Migration playbook for AWS large migrations

Amazon Web Services (AWS)

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

In a large migration, the migration workstream uses the wave plans and migration metadata supplied by the portfolio workstream in order to migrate workloads to the Amazon Web Services (AWS) Cloud. The migration workstream is responsible for submitting any change requests, migrating the application, coordinating application testing with the application owners, performing cutover, and monitoring the application through the hypercare period. In the first stage, initializing a large migration, you create the runbooks that the migration workstream uses to migrate the applications and servers. In the second stage, implementing a large migration, the migration workstream plans sprints and uses the migration runbooks in order to migrate and cutover the applications.

For more information about core and supporting workstreams, see Workstreams in a large migration in the *Foundation playbook for AWS large migrations*.

About this playbook

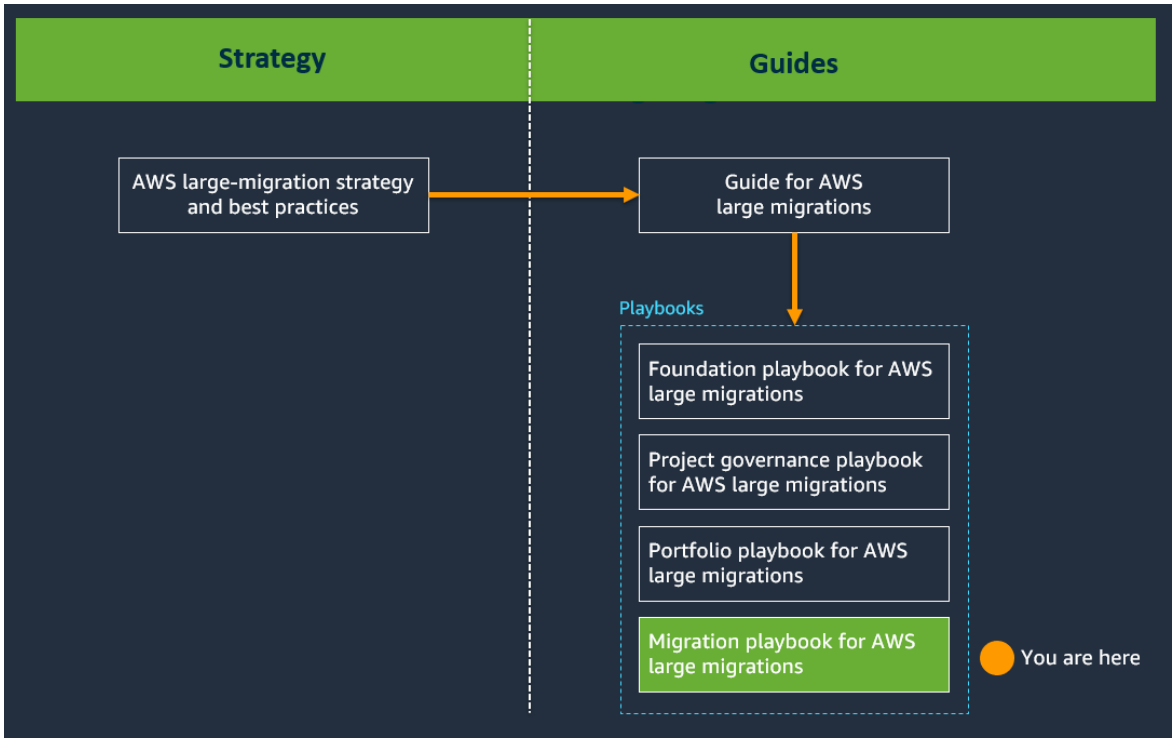
This guide is part of a series about large migrations to the AWS Cloud. Migrating 300 or more servers is considered a *large migration*. If you haven't already done so, we highly recommend reading the following:

1. AWS large-migration strategy and best practices – This strategy discusses best practices for large migrations and provides use cases from customers across various industries.
2. Guide for AWS large migrations – This guide describes a high-level, phased approach to implementing the best practices outlined in the strategy document.

After reading the strategy and guide, the playbooks help you focus on how to implement the strategy and connect all the moving parts of a large migration. A *playbook* helps you build the assets and processes that you use in a large migration, such as runbooks, governing documents, or tools. A *runbook* is a standard operating procedure for a task that you perform repeatedly. The following playbooks are available in this document series for large migrations, and we recommend that you read them 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 (this guide)

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



This migration playbook outlines the tasks of the migration workstream, which spans both stages of a large migration, initialization and implementation:

- In stage 1, *initialize*, you draft, test, and refine the runbooks, and then you automate manual tasks for each migration pattern.
- In stage 2, *implement*, you perform the migration with the predefined runbooks built in stage 1.

About the runbooks, tools, and templates

We recommend using the migration playbook templates and then customizing them for your portfolio, processes, and environment. The provided templates include standard processes, typical cutover processes, and placeholders for processes that are unique to your environment. The instructions in this playbook tell you when and how to customize each of these templates. This playbook includes the following templates:

- Rehost migration runbook template
- Rehost migration task list template

For migration patterns, from which you can build your own runbooks, see AWS Prescriptive Guidance migration patterns.

Migration runbooks require varying levels of detail:

- **Detailed runbooks** – Detailed runbooks are best suited for migration patterns that you will repeat many times. For these patterns, we recommend starting with the *Rehost migration runbook template* (Microsoft Word format). This template captures as many details as possible, including screenshots and step-by-step instructions, and it is designed to help multiple people perform the same task consistently.

- **Task List** – For migration patterns that are one-off or very simple, a short task list is a better option. For these patterns, we recommend starting with the *Rehost migration task list* template (Microsoft Excel format). This template contains a high-level task list and is typically used for tracking and managing ownership of tasks. You can also use a task list to track the status of tasks that are documented in a runbook.

Whether you are using a detailed runbook or a short task list, verify that your runbook describes the tasks in sequence. For complex tasks, you can provide links to external documentation.

Stage 1: Initializing a large migration

In the initialize stage, the goal is to define standard operating procedures (SOPs) for the large migration, also known as *runbooks*. You build custom runbooks based on your company's policies and processes. If another team member is responsible for defining the runbooks in your large migration project, skip to Stage 2: Implementing a large migration (p. 13), where you will use the runbooks to prioritize applications and perform wave planning. Stage 1 consists of the following tasks and steps:

With the migration runbooks in place, in stage 2, the migration teams follow the procedures and perform large migrations that have predictable and measurable outcomes.

Task 1: Validating the migration patterns and metadata

In this task, you validate the migration patterns identified in the assessment and wave planning activities in the portfolio workstream, and then you validate the migration metadata source. The goal is to verify that sufficient data has been collected to support each migration pattern.

This task consists of the following steps:

Step 1: Validate the migration patterns

In the portfolio workstream, you performed an initial assessment of the application portfolio, selected migration strategies, and identified migration patterns for each strategy. This information should be contained in your portfolio assessment runbook. For more information, see the Portfolio playbook for AWS large migrations.

In this step, you review the migration strategies, verify that you have identified all migration patterns, and confirm that you are ready to draft migration runbooks. You might repeat this task throughout

the project, and as your understanding of the portfolio matures, it is likely you will identify additional migration patterns in later stages of the migration.

1. **Review the migration strategies for the portfolio**

   A *migration strategy* is the approach used to migrate an on-premises application to 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. For more information about the 7 Rs, see the Guide for AWS large migrations.

   Based on the output of the initial portfolio assessment, you have a list of all the required migration strategies for the portfolio and determined how much of the portfolio is allocated to each strategy. For example:

   - Rehost – 70%
   - Replatform – 20%
   - Retire – 10%

2. **Verify that the migration patterns for the portfolio**

   A *migration pattern* is a repeatable migration task that details the strategy, the destination, and the application or service used. In this step, you verify that the migration patterns include detailed information, such as which tools to use and which AWS services are targeted. For example:

   - Rehost to Amazon Elastic Compute Cloud (Amazon EC2) by using AWS Application Migration Service (AWS MGN) or Cloud Migration Factory
   - Replatform to Amazon EC2 by using AWS CloudFormation templates to build new infrastructure in the AWS Cloud
   - Replatform to Amazon Relational Database Service (Amazon RDS) by using AWS Database Migration Service (AWS DMS) or a native database technology

   In the Portfolio playbook for AWS large migrations, you map each migration pattern to its migration strategy and document the results in a table like the following example.

Strategy	Pattern
Rehost	Rehost to Amazon EC2 by using Application Migration Service or Cloud Migration Factory
Replatform	Replatform to Amazon RDS by using AWS DMS or a native database technology
Replatform	Replatform to Amazon EC2 by using AWS CloudFormation templates to build new infrastructure in the AWS Cloud

Step 2: Validate the migration metadata and wave plan

In this step, you validate the source location of the migration metadata. You check that the data structure, such as the available columns in an Excel document, is suitable to hold the required metadata, and you check that all the metadata is available.

1. **Validate the migration metadata for your migration patterns**

   Each migration pattern needs a different set of migration metadata in order to migrate the servers and apps. For example, a rehost migration to Amazon EC2 requires that you provide specifications for the target instance, such as the VPC subnet, security group, and instance type information. However, a storage migration, database migration, or replatform migration requires a different set of migration metadata. You typically define migration metadata requirements in the portfolio assessment runbook, but you need to make sure that you have sufficient metadata to support each of your migration patterns. For more information about metadata identification and collection, see the Portfolio playbook for AWS large migrations.

2. **Validate the source location of the migration metadata and the wave plan**

   You typically document the source location of the migration metadata in your metadata management runbook. Ideally, the location acts as a single source of truth, such as a wave-planning spreadsheet. It is also possible that the metadata is still in multiple places, including the following common locations:

   Validate the following for the metadata source location:

   - Discovery tool
   - Configuration management database (CMDB)
   - App owner questionnaire
   - Migration wave-planning spreadsheet

   Validate the following for the metadata source location:

   a. Is the source catalog being maintained with locations of all metadata sources and owners?
   b. Does the source location (for example, wave-planning spreadsheet) have all the required migration metadata?
   c. Are there clear instructions for accessing each metadata source?
   d. If there is no single source, is each metadata source clearly mapped to its attributes?
   e. Is there a clear wave plan for the servers and apps, and are at least five waves ready for the migration workstream?
   f. Is there a process to update the sources? If so, what is the frequency and notification process?

Task exit criteria

When you have met the following exit criteria, proceed to the next task:

- You have validated the list of clearly defined migration patterns.
- The source location of the migration metadata has all the required metadata for each pattern, or a process is in place to capture any missing metadata.
- You have validated the wave plan and migration metadata for at least five waves, and you have defined a process for notifications and updates.

Task 2: Creating drafts of the migration runbooks

In this task, you draft and review migration runbooks for each migration pattern. For example, you draft a migration runbook for rehost to Amazon EC2 and another runbook for replatform to Amazon RDS. You repeat this task until you have drafted a migration runbook for every migration pattern identified in the previous task.

You can use the provided runbook templates available in the migration playbook templates and customize them for your environment. For migration patterns that are repeated frequently, we

recommend using the *Rehost migration runbook template* (Microsoft Word format), and for patterns that are one-off or very simple, we recommend the *Rehost migration task list template* (Microsoft Excel format). You can also use a task list to track the status of tasks that are documented in a runbook. For more information, see About the runbooks, tools, and templates (p. 2).

This task consists of the following steps:

Step 1: Create a migration runbook draft for each pattern

In this step, you draft runbooks for each of your migration patterns. A complete migration runbook typically contains instructions for how to use the selected migration service or tool, any tasks that are unique to your environment, and cutover instructions.

1. Open the *Rehost migration runbook template* (Microsoft Word format), available in the migration playbook templates.
2. Update the *Premigration tasks* section, *Migration tasks* section, and *Cutover tasks* section with instructions that are specific to your migration pattern. Depending on your use case, you might need to update all three sections. Include the following when customizing your tasks:
   - **Standard migration instructions for the selected service** – You can typically find the information needed to complete your template in AWS documentation. For example, see the following:
     - How to use the new AWS Application Migration Service for lift-and-shift migrations
     - Getting started with AWS DataSync
     - AWS Database Migration Service step-by-step walkthroughs
   - **Tasks that are unique to your IT environment** – Record the tasks that are unique to your IT operations and environment. The goal is that a new person joining your migration teams can follow the runbook with minimal learning curve. For example, what monitoring software do you need to install on the target machine after cutover? Which Domain Name System (DNS) server do you use for that subnet? How do you submit a request for change (RFC)?
   - **Cutover tasks** – Every environment has a slightly different cutover process. It is important to document all the steps for cutover in your environment because you want everyone to follow the same process. Documenting these steps minimizes time spent in the cutover window and helps you plan the amount of time needed to complete the cutover.

Step 2: Update the migration runbooks with your policies and processes

Runbook and task list templates cover the majority of the migration tasks, or the portion of the process that is standard. The remaining tasks are unique to your environment, and you must customize the runbook accordingly. For example, consider whether your runbooks should contain custom tasks for the following processes in your environment.

Connectivity

- How to connect to a VMware environment
- How to connect to a DNS server and update DNS records

- How to connect to the migration automation server
- How to connect to the source environment
- How to connect to a document repository, such as SharePoint or Confluence

Permissions and change management

- How to submit an RFC in your environment
- How to review the status of the RFC for each wave
- How to create a user account for a new migration engineer
- How to request permissions to the source servers
- How to request permissions to the target AWS account
- Who has permission to connect to the target server after the cutover

Migration implementation and cutover

- Which software to install or uninstall on the target server
- How to change infrastructure settings, such as firewall, routing, and load balancer settings
- Who can change infrastructure settings
- How to change the application configuration during cutover
- How to conduct application testing
- How to complete a cutover and go-live
- How to complete tasks that occur after cutover, such as configuring monitoring or backups

Some of these tasks might sound trivial, but knowledge and permissions vary in any environment. It is important to document these tasks in the same migration runbook.

> **Tip**
> We highly recommend using automation to accelerate your large migration. Using a migration factory model simplifies and reduces the number of issues with repetitive tasks, especially for rehost and replatform migration patterns.
> AWS Cloud Migration Factory Solution was designed to help customers migrate at scale with automation. You can deploy the solution and use predefined automation scripts in your runbook.

Task exit criteria

Repeat this task as necessary, and when you have met the following exit criteria, proceed to the next task:

- You have drafted a runbook for each migration pattern.
- Each runbook draft contains three main sections: pre-migration tasks, migration tasks, and cutover tasks.
- Your runbook drafts include tasks that are unique to your environment.
- Your detailed runbook drafts include step-by-step guidance and screenshots.

Task 3: Analyzing and testing your migration runbooks

In this task, you walk through each runbook that you built in the previous task, analyze any identified gaps, conduct a migration proof of concept (POC), and review the notes and feedback.

This task consists of the following steps:

- Step 1: Conduct a walkthrough of each runbook (p. 9)
- Step 2: Conduct a POC that tests each migration pattern (p. 9)
- Step 3: Review and identify the gaps in the current migration runbook drafts (p. 10)

Step 1: Conduct a walkthrough of each runbook

In this step, the migration teams assess the runbook and task sequence as though they were performing it for real. The migration teams meet and review each step, and the team members ask questions and share their feedback. This walkthrough process helps the teams identify missing steps and sequence issues. Complete the walkthrough as follows:

1. Gather the migration teams that are responsible for completing the tasks in the runbook.
2. Walk through the steps in the runbook one by one, as if this was a live migration. As you go, identify and make note of any gaps or issues. Do not perform the migration or tasks as part of the walkthrough.
3. Update the runbook draft to address any gaps or issues identified in the walkthrough.

Step 2: Conduct a POC that tests each migration pattern

1. Select a POC candidate from the already prepared waves.
2. Open the migration runbook draft.
3. Complete the runbook step by step in order to migrate the POC candidate as follows:
   - Follow every step in the runbook. Do not make assumptions or make your own decisions.
   - Assume the person using the runbook has no prior knowledge about migration or your environment.
   - If a step is not clear but you can continue, make note of the step and continue.

- If a step is missing and you can't continue, stop, and highlight the section from which you could not proceed. Work with the runbook owner to clarify the missing step so that you can continue and complete the POC.

Step 3: Review and identify the gaps in the current migration runbook drafts

1. Review any issues or gaps identified in the previous steps.
2. Analyze the gaps and consider the following questions:
   - Does the runbook have the steps needed to complete a migration and cutover, from end to end?
   - Does the runbook contain reference links for the tasks that are predefined in your environment?
   - Does the runbook clearly defined who, what, when, and how to complete a task?

Task exit criteria

When you have met the following exit criteria, proceed to the next task:

- You have reviewed and tested each migration runbook.
- For each runbook, you have completed a migration POC for at least one application and for more than two operating system (OS) variants.
- You have identified and documented the identified gaps and issues in each runbook.

Task 4: Improving your migration runbooks

In this task, you improve the runbooks by repeating the POC multiple times. With each wave, the POC test and *retrospective*, a meeting in which the team reviews the completed wave, provide opportunity to improve the runbooks. You also improve your runbooks by automating repetitive tasks, which increases the velocity of the migration and reduces the risk of manual configuration errors.

This task consists of the following steps:

- Step 1: Update the migration runbooks and repeat the testing (p. 10)
- Step 2: Automate repetitive tasks (p. 11)
- Step 3: Build a migration task list (p. 11)

Step 1: Update the migration runbooks and repeat the testing

1. For the issues and gaps identified in the previous task, update the runbooks with detailed instructions. For example:
   - If a step is missing, add step-by-step instructions
   - If a step is not clear, consider updating the text, adding a screenshot, or adding reference links
2. Repeat the previous task until you are satisfied that the instructions are complete and clear.

3. Test the final draft of each runbook by asking a new migration team member, one who has not tested this runbook before, to perform a POC and complete the runbook.

Step 2: Automate repetitive tasks

1. Review each runbook and identify areas of automation for manual tasks. Consider the following probing questions:
   - Are there any repetitive, manual tasks for each server or app in the runbook?
   - Are there any actions that you perform on every server or application?
   - Do you need to install or uninstall software on the target server?
   - Do you need to change network or infrastructure settings one by one for each server?
   - Do you need to manually copy and paste any data?
2. Build automation scripts and update the runbooks.
3. Repeat task 3 and task 4 until you have documented the runbooks with clear and complete information and automated repetitive migration tasks.

> **Note**
> For automating migration tasks, we highly recommend that you build new scripts or customize existing scripts in AWS Cloud Migration Factory Solution.

Step 3: Build a migration task list

A migration task list can help you manage the status and owners of tasks. You build a task list for each migration runbook, and you include the high-level information from the runbook without including the details. A task list typically contains the following information, and you can add more attributes as needed:

- Descriptive name, such as:
  - Check server OS version
  - Install an agent
  - Restart a server
  - Update the DNS
- Dependencies
- Sequence of tasks
- Owner
- Estimation of time required to complete each task
- Status

There are many tools available for creating and managing task lists. You can use the provided *Rehost migration task list template* (Microsoft Excel format) available in the migration playbook templates. You can also use project management tools, such as Jira or a Kanban board.

> **Note**
> We also recommend using the Excel task list template to document small, well-understood, or non-repetitive tasks, such as restarting a server or getting an IP address. These tasks should be captured and tracked but don't require the detailed steps of the Word runbook template.

Task exit criteria

Repeat this task as necessary, and when you have met the following exit criteria, proceed to the next task:

- You have identified opportunities for automation and have either developed automation scripts or have a plan to do so.
- Three or more people have peer-reviewed each runbook.
- Two or more people who were not on the development team for the runbook have tested it end-to-end.
- Using the most up-to-date runbook, you have migrated 20 or more servers to more than one AWS account.
- You have developed a task list to help track and manage the progress of the migration.

Stage 2: Implementing a large migration

In stage 1, you developed migration runbooks for each migration pattern. In stage 2, you use these runbooks to migrate servers and then improve the runbooks in order to accelerate the velocity of the migration. Building and updating runbooks is not a one-off task. You might need to do that throughout your large migration journey. For example, you might need to create new runbooks if the scope increases and you identify new migration patterns, or you might need to improve the existing runbooks if the migration velocity is below the target and introducing more automation would reduce the number of manual tasks and accelerate the migration.

> **Note**
> The wave plan developed in the portfolio workstream determines the activities in the migration workstream. Before starting stage 2, verify that you have validated your wave plan. For instructions and more information about the wave plan, see Portfolio playbook for AWS large migrations.

Stage 2 consists of the following tasks and steps:

Task 1: Performing sprint planning for scheduled waves

In this task, you assign waves to *sprints*, which is a fixed period of time in which the migration team works on all waves within that sprint. If each sprint is 2 weeks in duration, each wave spans at least two sprints. *Sprint planning* refers to the process of assigning owners and due dates to all of the tasks within that sprint.

This task consists of the following steps:

Step 1: Review the backlog for the scheduled waves

In this step, you review existing *backlogs*, or current and pending tasks, for all the concurrent waves, and you use the recommended tools and mechanisms to manage the wave. For example, you might use a Kanban board with a swimlane for each wave, or you might use Jira and track waves with stories and epics. For more information, refer to the Project governance playbook for AWS large migrations.

Step 2: Assign tasks and establish due dates

In this step, for all waves in this sprint, you assign owners to each task and set a due date accordingly. You can use the migration task list spreadsheet you created in stage 1 to manage your wave progress, task ownership, and due dates, and the tasks are defined in detail in the migration runbook for each pattern. Because waves typically overlap, it is common to manage many concurrent tasks from different waves at the same time. In addition, each wave can range from 3-6 weeks, depending on your internal process. For an example of a wave schedule, see the Stage 2: Implement a large migration section of the *Guide for AWS large migrations*.

Important
Do not add tasks to the sprint without updating the runbook or task list. These documents that you built in stage 1 should be a source of truth for all your migration activities. If any step is missing or incorrect, update and validate the runbook before adding tasks to the sprint.

Task 2: Performing pre-migration and migration tasks

Now you perform pre-migration and migration tasks and adhere to a schedule based on your sprint planning outcome. A sprint backlog contains a list of all tasks in the migration, for all waves in the current sprint, and organizes the tasks by week. For a list of tasks, see your migration runbooks for each migration pattern, which were created in stage 1 of this playbook. For the wave schedule, see your project management tools, which were established in the Project governance playbook for AWS large migrations. Perform the tasks in the scheduled weeks. The following is an example of a rehost migration task schedule in which there are migration tasks for different waves in the same week.

Task name	Wave	Category	Owner
Verify prerequisites	Wave 1	Build	Jane Doe
Install replication agent	Wave 1	Build	Jane Doe
Validate launch template	Wave 2	Validate	Jane Doe
Launch test instances	Wave 3	Boot-up testing	Jane Doe

Task 3: Performing cutover tasks

At this point, you have completed the migration tasks and tested all of the servers and apps, and you are ready for cutover. Use the RACI matrices you created in the Foundation playbook for AWS large migrations to manage the tasks and ownership of each cutover task, and use your migration runbook for each pattern to perform the cutover activities. The following table is an example of how you might track and manage cutover progress. It is common to have multiple migration patterns in the same wave for different applications.

Task name	Wave	Migration runbook	Owner	Status
Check replication	Wave 1	Rehost to Amazon EC2	Jane Doe	Completed

Task name	Wave	Migration runbook	Owner	Status
Launch cutover EC2 instance	Wave 1	Rehost to Amazon EC2	Jane Doe	Completed
Validate EC2 instance status	Wave 1	Rehost to Amazon EC2	Jane Doe	In progress
Launch databases in Amazon RDS	Wave 1	Replatform to Amazon RDS	John Smith	In progress
Complete storage data transfer	Wave 1	Replatform to Amazon Elastic File System (Amazon EFS)	John Smith	Not started
Perform app testing	Wave 1	All	Jane Doe	Not started
App acceptance decision	Wave 1	All	Jane Doe	Not started

Task 4: Reviewing and improving the migration runbooks

This task consists of the following steps:

Step 1: Review the completed waves and identify gaps in the current migration runbook

Fail fast is a philosophy that uses frequent and incremental testing to reduce the development lifecycle, and it is a critical part of an agile approach to a large migration. After each cutover, schedule a retrospective meeting to review each task with the migration teams. Ask the following probing sample questions. You can also add your own questions:

- Was the cutover successful? If not, what was the issue?
- Does the migration runbook cover all of the tasks to perform the migration and cutover?
- Do any of the tasks take longer than expected?
- Are you aware of any technical issues with any tasks in the runbook?
- Are there any manual tasks that can be automated?

- Are there any process-related issues with the runbook or cutover?

Step 2: Update the migration runbooks and complete testing

After collecting data from the retrospective meeting, update the migration runbooks as follows:

- Add detailed instructions for any missing steps.
- Fix or update any steps as needed.
- Perform an end-to-end migration test with at least one Windows and one Linux server.
- Send the updated runbook to the migration teams for use in the next wave.

Resources

AWS large migrations

Strategy

- AWS large-migration strategy and best practices

Guide

- Guide for AWS large migrations

Playbooks

- Foundation playbook for AWS large migrations
- Project governance playbook for AWS large migrations
- Portfolio playbook for AWS large migrations

Additional references

- AWS Cloud Migration Factory Solution
- 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 Cloud Migration Factory guide in this content set.
migration metadata	The information about the application and server that is needed to complete the migration. Each migration pattern requires a different set of migration metadata. Examples of migration metadata include the target subnet, security group, and AWS account.
migration pattern	A repeatable migration task that details the migration strategy, the migration destination, and the migration application or service used. Example: Rehost migration to Amazon EC2 with AWS Application Migration Service.
migration 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:

- Chris Baker, Senior Migration Consultant, Amazon Web Services
- Wally Lu, Principal Consultant, Amazon Web Services

Document history

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

Change	Description	Date
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