AWS Partner: Advanced Migrating to AWS (Technical)

Student Guide

Version 3.0.0

300-PTADMI-30-EN-SG

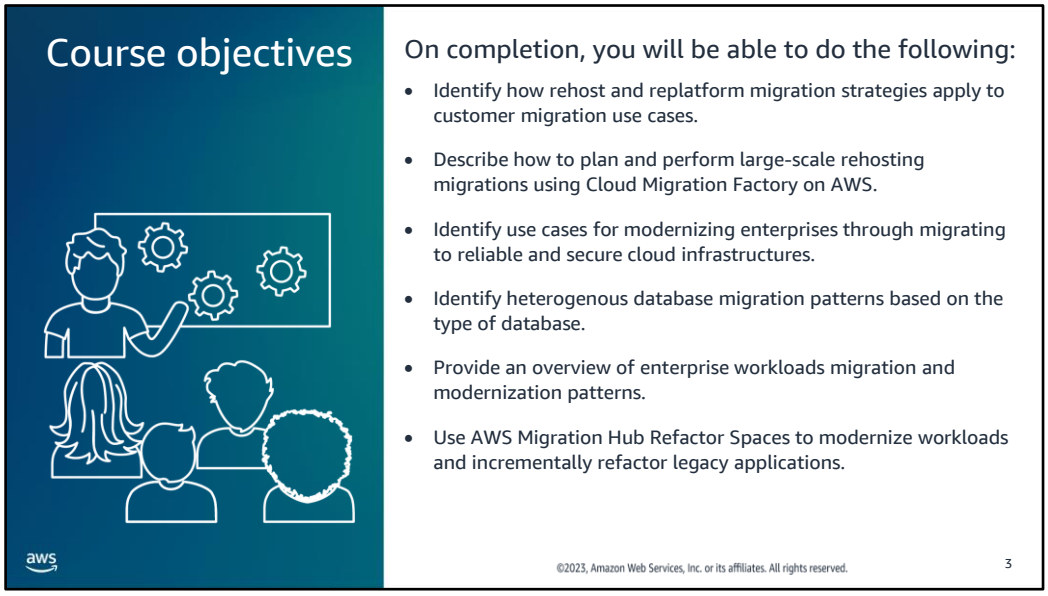
Corrections, feedback, or other questions? Contact us at
https://support.aws.amazon.com/#/contacts/aws-training.

Contents

Welcome to the AWS Partner: Advanced Migrating to AWS course.

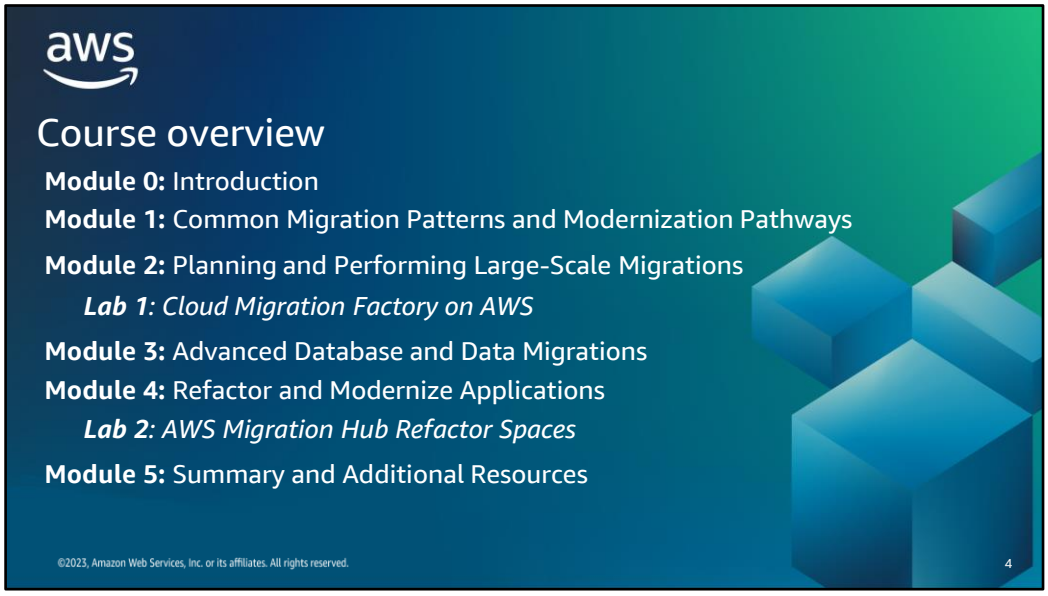
Welcome to AWS Partner: Advanced Migrating to AWS.
Our course today will run 09:00–17:00.

A few important notes for today's course. This session is not being recorded, but the slide deck will be sent following the class. To get credit for course completion, you will need to attend a majority of the course.

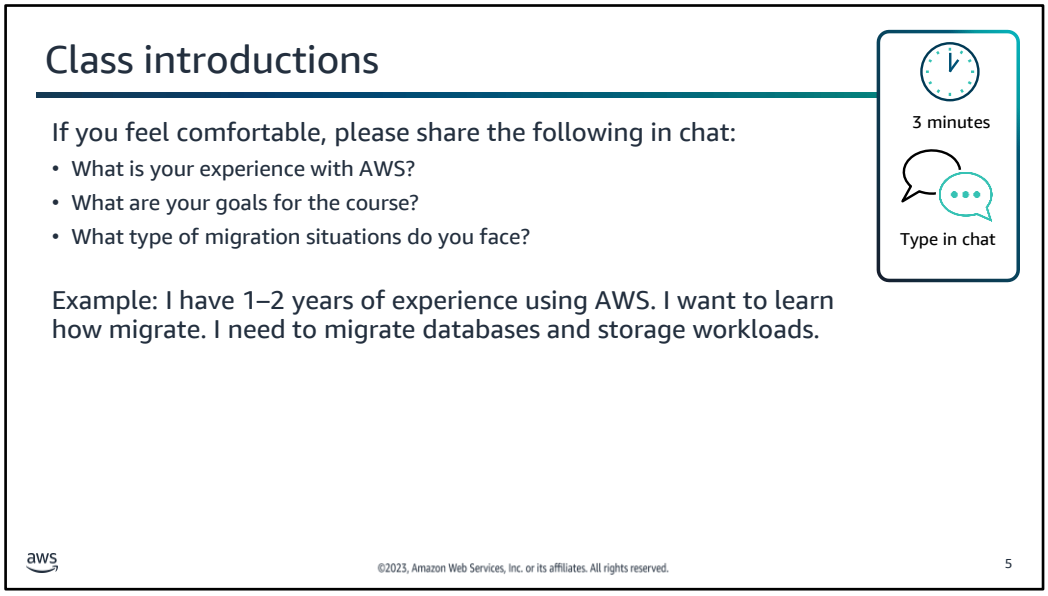


On completion of this course, you will be able to do the following:

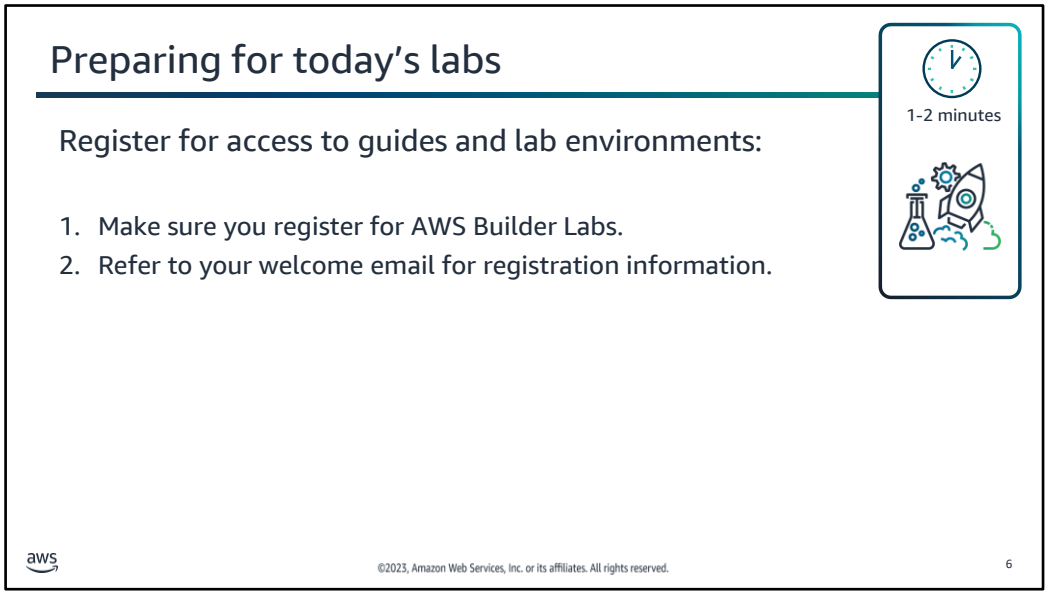
- Identify how rehost and replatform migration strategies apply to customer migration use cases.
- Describe how to plan and perform large-scale rehosting migrations using Cloud Migration Factory on Amazon Web Services (AWS).
- Identify use cases for modernizing enterprises through migrating to reliable and secure cloud infrastructures.
- Identify heterogenous database migration patterns based on the type of database.
- Provide an overview of enterprise workloads migration and modernization patterns.
- Use AWS Migration Hub Refactor Spaces to modernize workloads and incrementally refactor legacy applications.



Here is an outline of the course topics and labs for today.

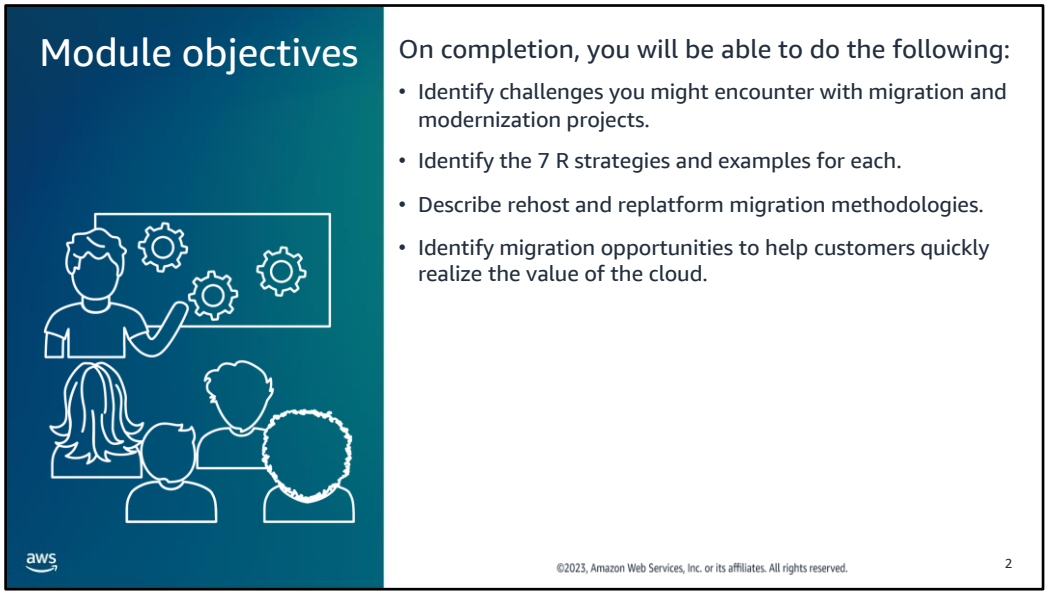


One of the advantages of a live class is getting to meet others that are learning alongside you. Take a few moments to learn about your colleagues in this class and consider sharing a bit about yourself.



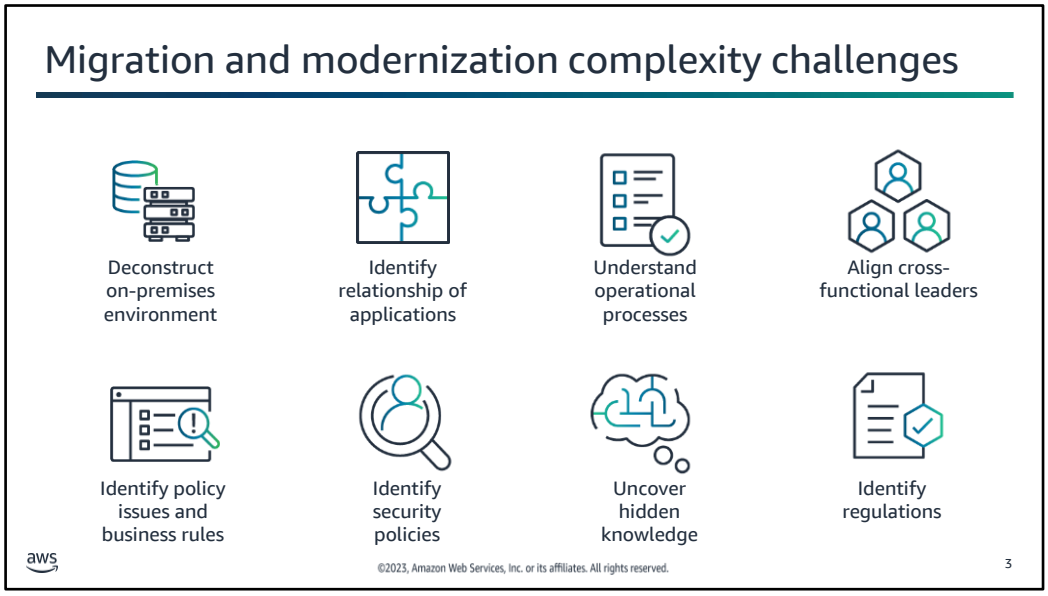
Check your inbox for a welcome email from your instructor. In this email, you will find your unique student registration URL for the class. Use this URL link to create an account or to log in to your existing AWS Builder Labs account. In AWS Builder Labs, you can access your lab environments and lab guide.

Welcome to Module 1: Common Migration Patterns and Modernization Pathways.



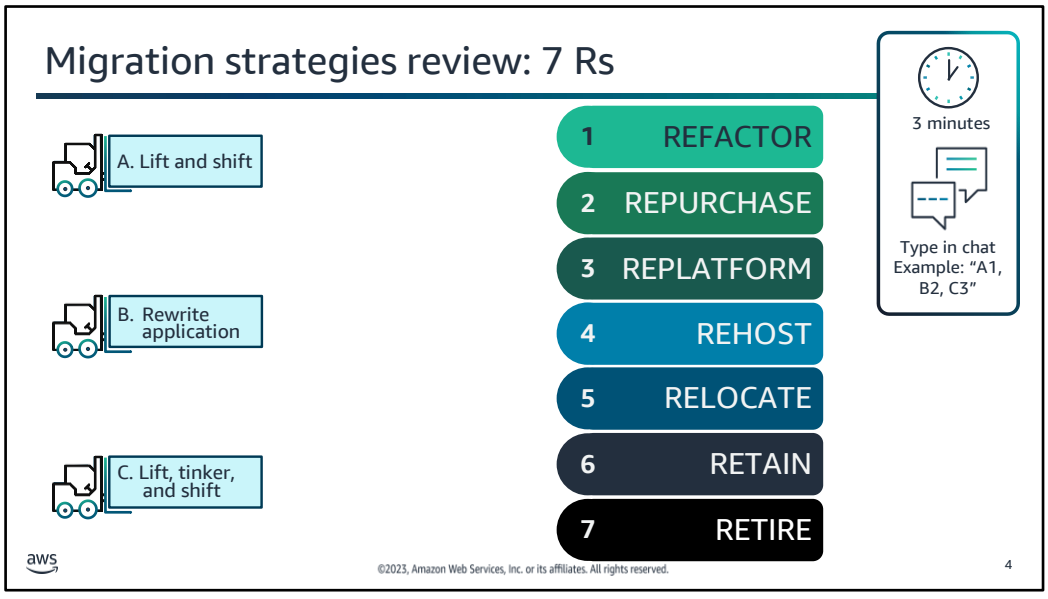
On completion of this course, you will be able to do the following:

- Identify challenges you might encounter with migration and modernization projects.
- Identify the 7 R strategies and examples for each.
- Describe rehost and replatform migration methodologies.
- Identify migration opportunities to help customers quickly realize the value of the cloud.



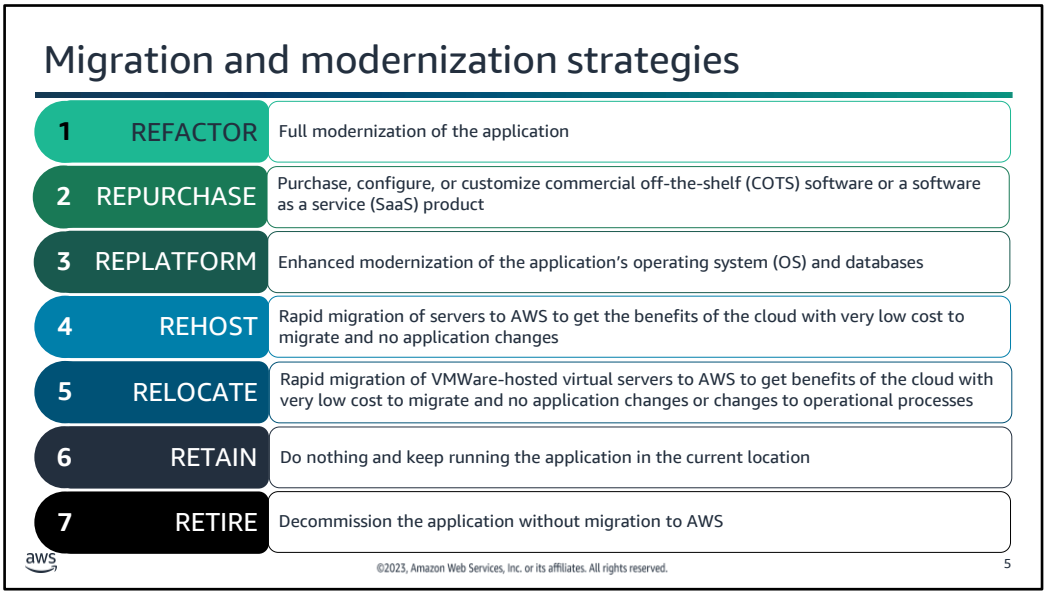
Migration and modernization projects can be challenging because of the complexity of these projects. You might encounter challenges as you complete the following:

- Deconstruct the on-premises technical environment.
- Identify the performance and relationship of migrating applications.
- Understand the current operational processes.
- Align and mobilize cross-functional leaders to encourage new ways of working.
- Identify policy issues and business rules.
- Identify security policies.
- Uncover hidden practices and knowledge.
- Identify regulations.



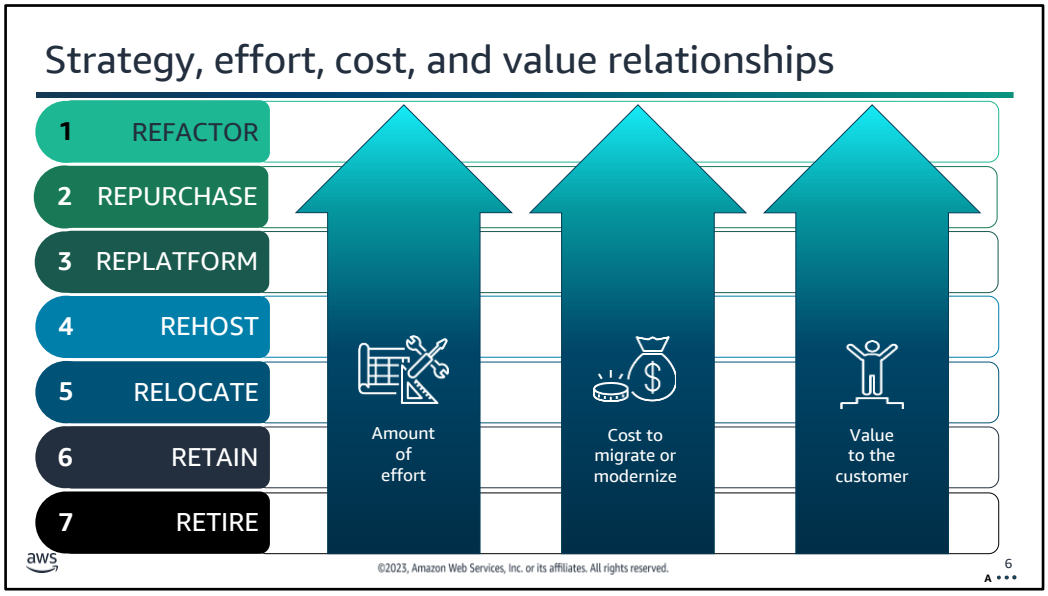
In this exercise, match the phrase on the left with the type of migration on the right. Type your answers in chat.
Starting with A. Lift and shift. Which migration strategy does this describe?
And B. Rewrite application. Which strategy does that describe?
And finally, C. Lift, tinker, and shift is which strategy?

Migration answers:
A. Which migration strategy is a lift and shift? **Rehost.**
B. Which migration strategy requires rewriting or redesigning the application? **Refactor.**
C. Which migration strategy is a lift, tinker, and shift that might require operating system (OS) changes or updated versions? **Replatform.**



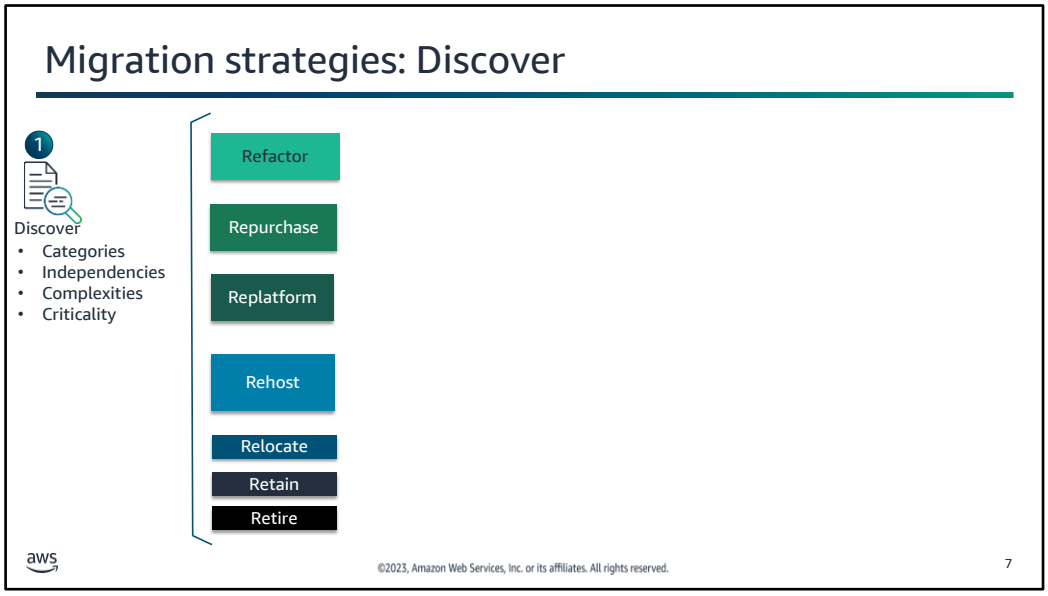
Let's look at the following examples for some of the strategies:

- **Refactor:** Rewriting monolith application code to microservices using cloud-focused features such as serverless technology
- **Repurchase:** Migrating an application to Salesforce SaaS
- **Replatform:** Upgrading to the latest OS version, migrating to an open source or managed database, like Amazon Aurora PostgreSQL Compatible Edition, and deploying all infrastructure and applications through code pipelines—this can include minor application code changes
- **Rehost:** Rapid, low-cost migration of servers running supported Windows and Linux versions—there is not a good business case for enhancing the application
- **Relocate:** Moving a private cloud-hosted virtual machine to VMware on AWS



There is a direct relationship between the seven migration strategies and the amount of effort, how much it costs to migrate or modernize, and the value to the customer. For example, it doesn't take much effort or cost to retire an application, and it also doesn't provide any value to the customer.

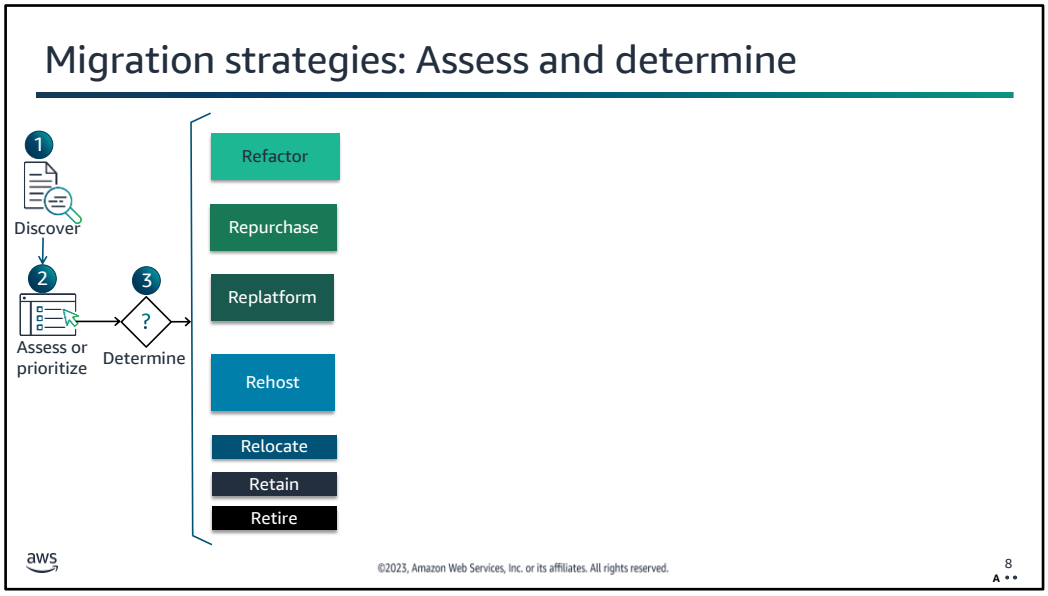
As you work through the strategies, from retire at the bottom to refactor at the top, the amount of effort to migrate or modernize, the cost, and the value to the customer increases.



For the first step to develop a migration strategy, you will do the following:

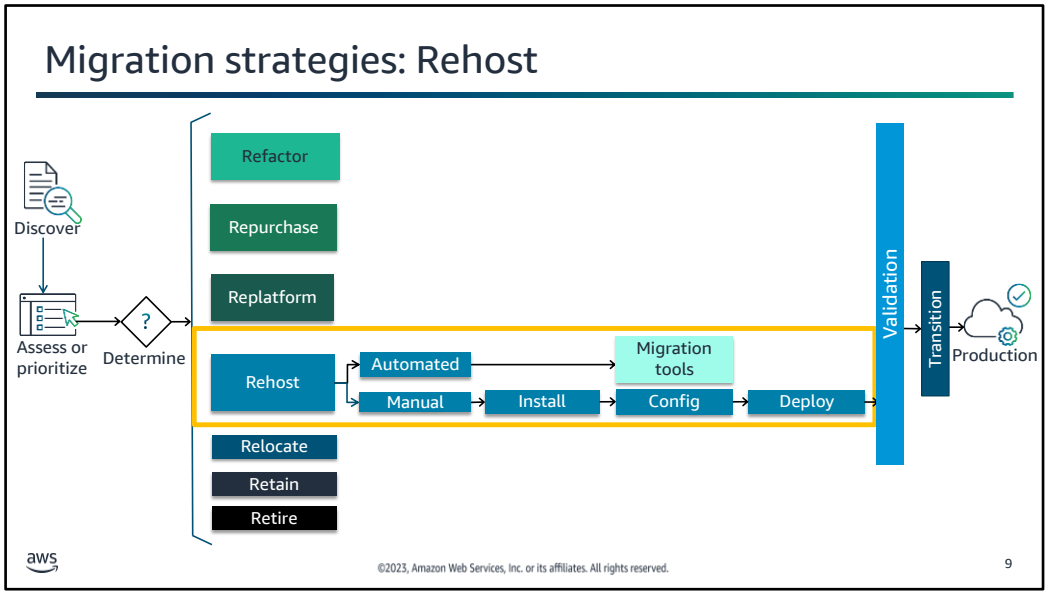
1. Discover and collect application portfolio data and rationalize it into the 7 Rs framework.

Using this method, categorize the applications in your environment and their interdependencies. Include each application's technical complexity and business criticality. Also include how to migrate each application or set of applications.

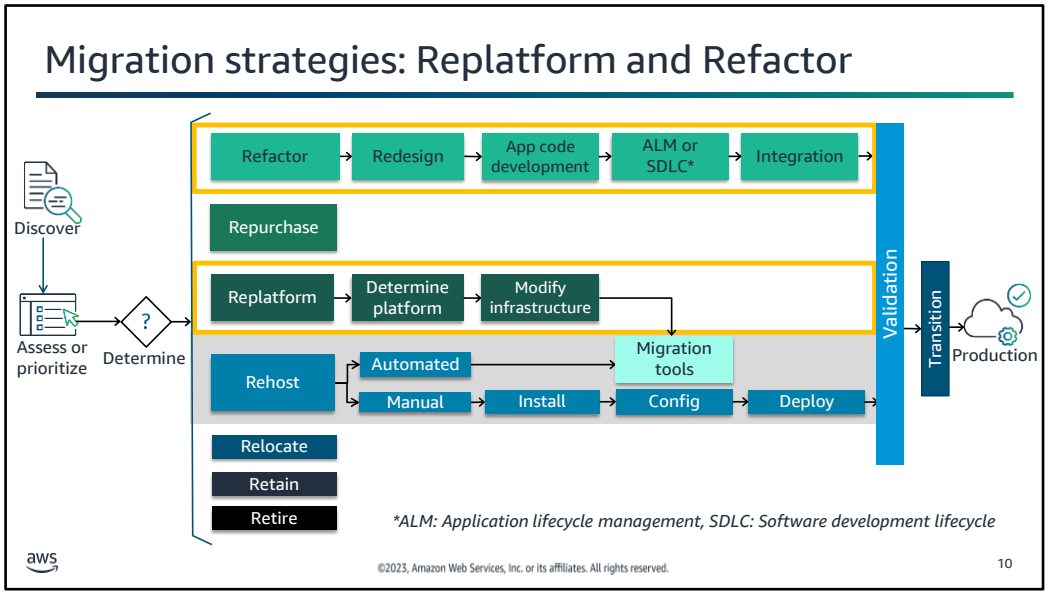


The next critical steps in planning and performing a successful migration are as follows:

2. Assess the application portfolio.
3. Determine the application migration strategy for all applications.

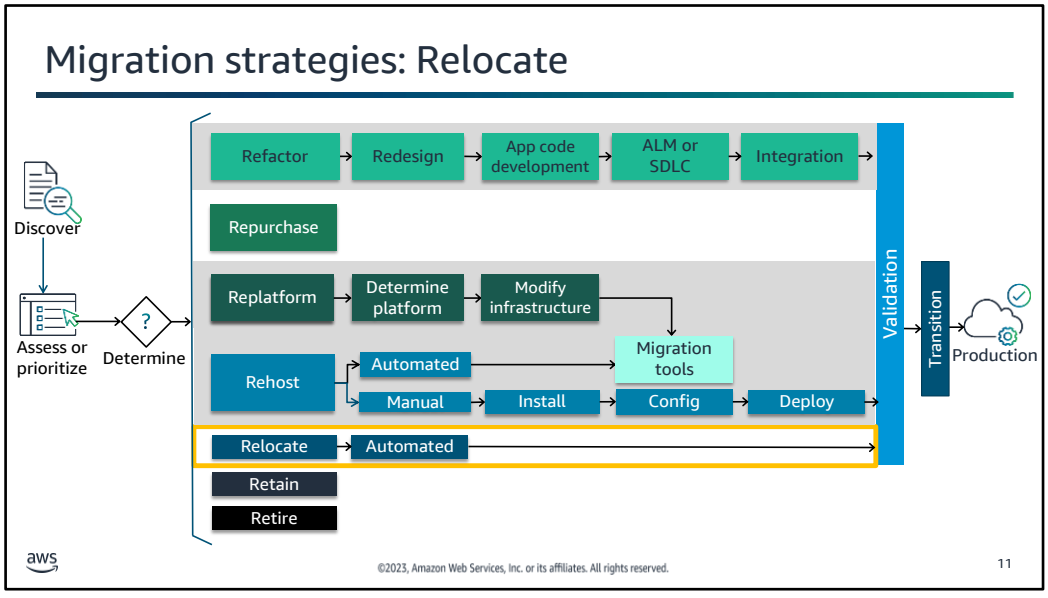


As a trusted advisor for your customers, begin by assessing their position in the migration journey. If a customer is new to this journey, initiate the process with a lift and shift (rehost) approach. When using the rehost approach, your customer can migrate quickly, gain cloud experience, and see cost savings.



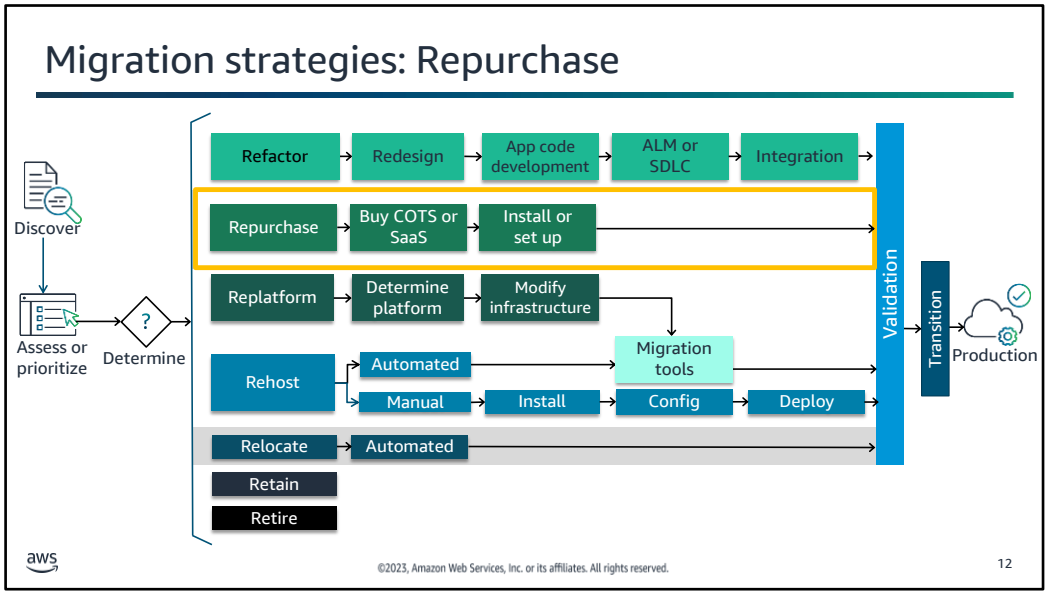
After the customer has gained experience in the cloud environment and refined their operational model during the transitional phase, you can guide them toward a strategic operations state where their teams can evolve.

In this state, your customer's teams can effectively operate and manage workloads that have been replatformed or refactored.

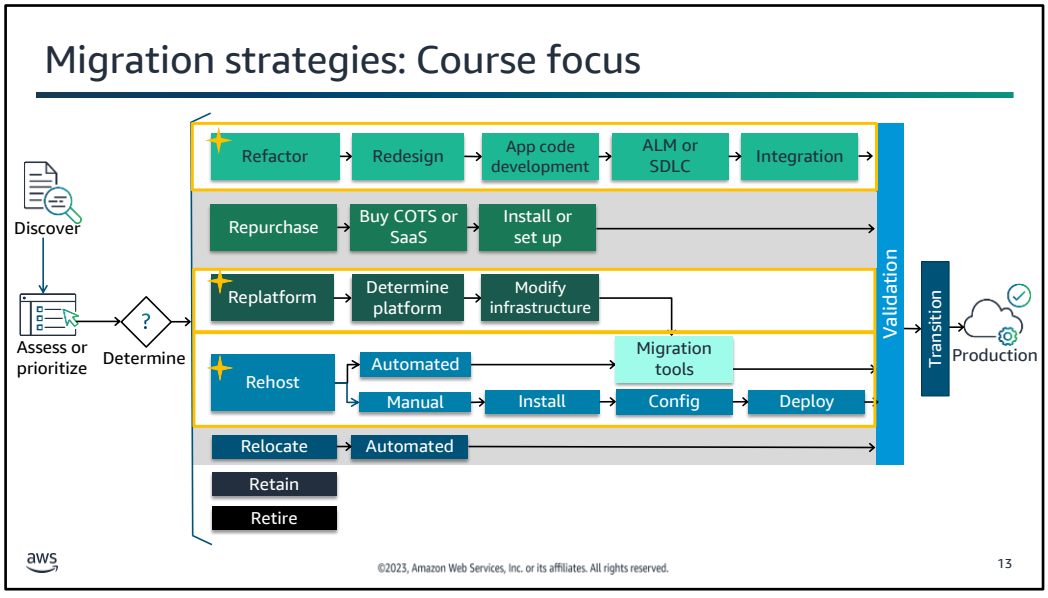


Relocate is an emerging migration strategy, especially for customers who must maintain large hybrid environments. You can relocate applications based on VMware and container technologies to AWS quickly and with minimal effort and complexity.

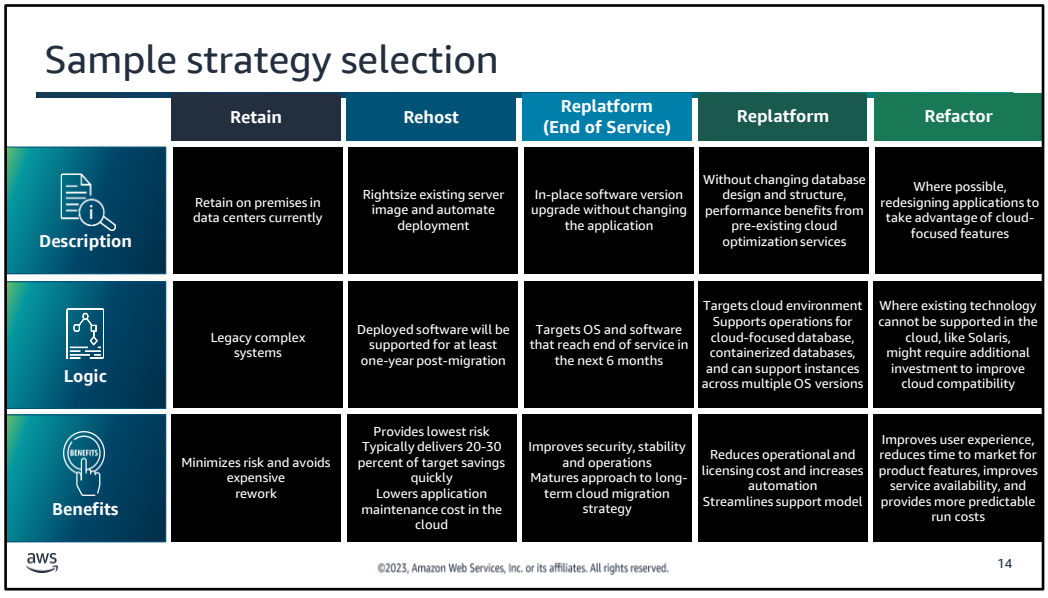
Relocate, at times, can also be applied to other solutions, such as Amazon Elastic Kubernetes Service (Amazon EKS).



Another option is the customer can move to repurchase, configure, or customize a COTS software or a SaaS product.

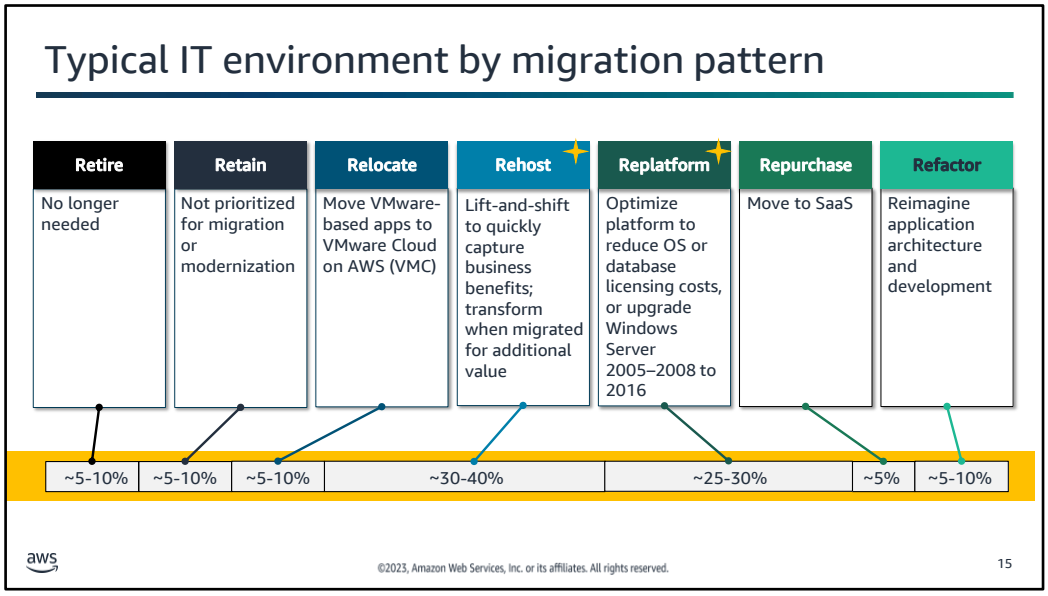


This course focuses on rehost, replatform, and refactor migration strategies.



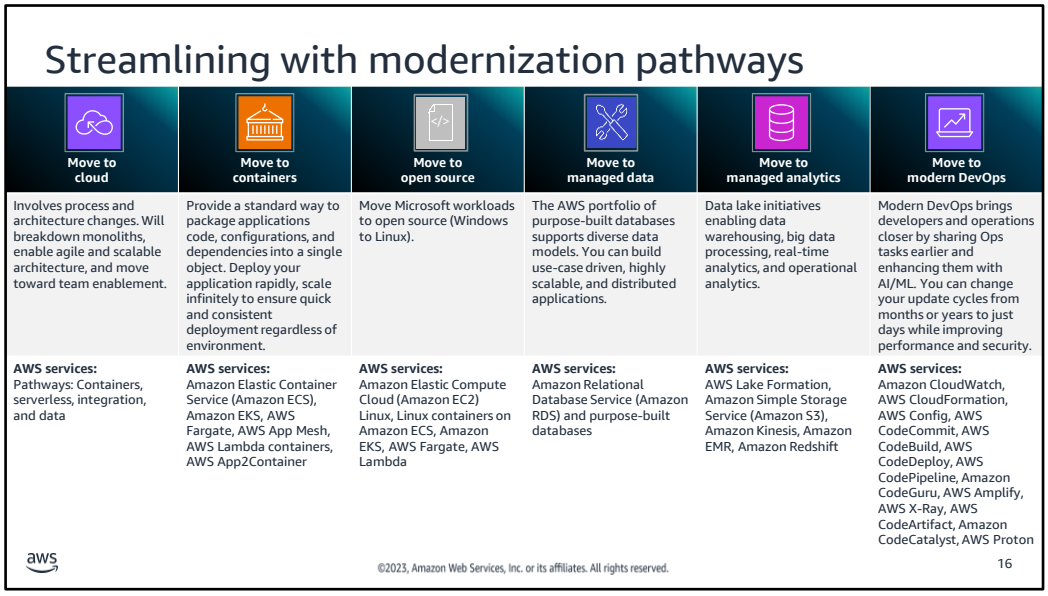
This slide provides the descriptions for sample strategy selection as a reference and a customer success story exploring a large-scale migration to AWS.

Customer success story: AWS Application Migration Service with 3M Company
Global manufacturer, 3M Company, migrated 2,200 applications to AWS in 24 months with minimal downtime. This strategy improved its scalability and resiliency and optimized its costs to save millions of dollars. To read this case study, see *Accelerating Migration at Scale Using AWS Application Migration Service with 3M Company* at https://aws.amazon.com/solutions/case-studies/3m-accelerating-migration-case-study/?did=cr_card&trk=cr_card.



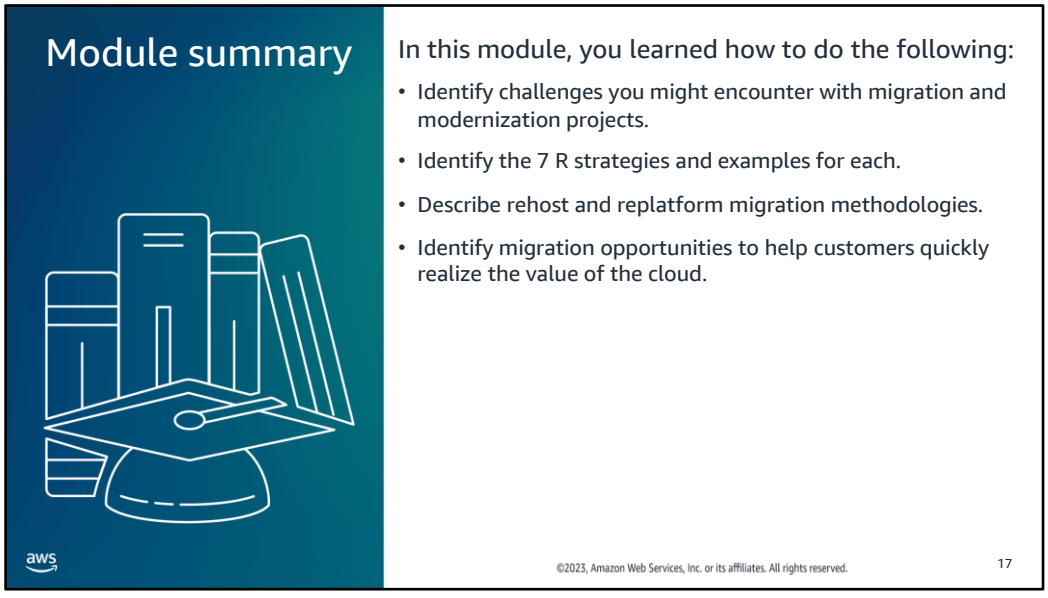
Although any two IT environments are not alike, you might wonder about the typical IT environment migration strategy percentages. Typically, companies use a mix of migration strategies, not just one. The largest percentages of migration strategies are in rehosting, with 30–40 percent and replatforming with 25–30 percent. These areas are where customers work to reduce OS or database licensing costs or move from Windows Server 2005–2008 to 2016.

To fully realize cloud benefits, use both migration and modernization. Use migrations to quickly realize the value of the cloud. Focus modernization on business differentiating applications.



Why is the move to modernization important to our customers? As organizations look to increase the pace of innovation, get more out of data, and build new customer experiences, they need to modernize the way they build and run applications. With modernization, customers can shift undifferentiated heavy lifting of operational responsibilities to AWS. This frees up resources for innovation-led activities. However, even with these benefits, many AWS customers are unsure on how they can start their modernization journey. They might lack of internal skill sets, broadness of modernization, or workload complexity.

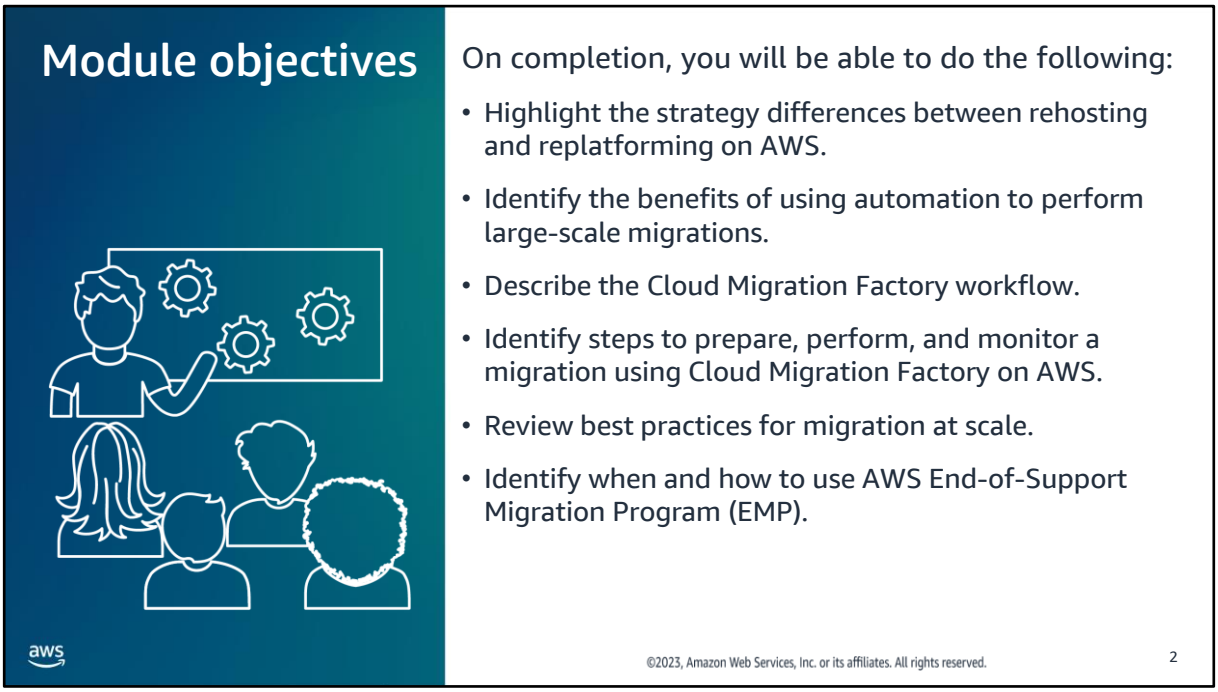
AWS has defined these modernization pathways that provide prescriptive modernization methodology and tooling to simplify and accelerate the customer modernization journey. Each pathway provides common cloud actions that include moving to the cloud, containerizing, open source, managed data and analytics, and modern DevOps. Pathways map to AWS services and guide customers on how to optimize for the cloud. One or many modernization pathways can be applied to a target application based on the customer's business objectives.



In this module you learned how to do the following:

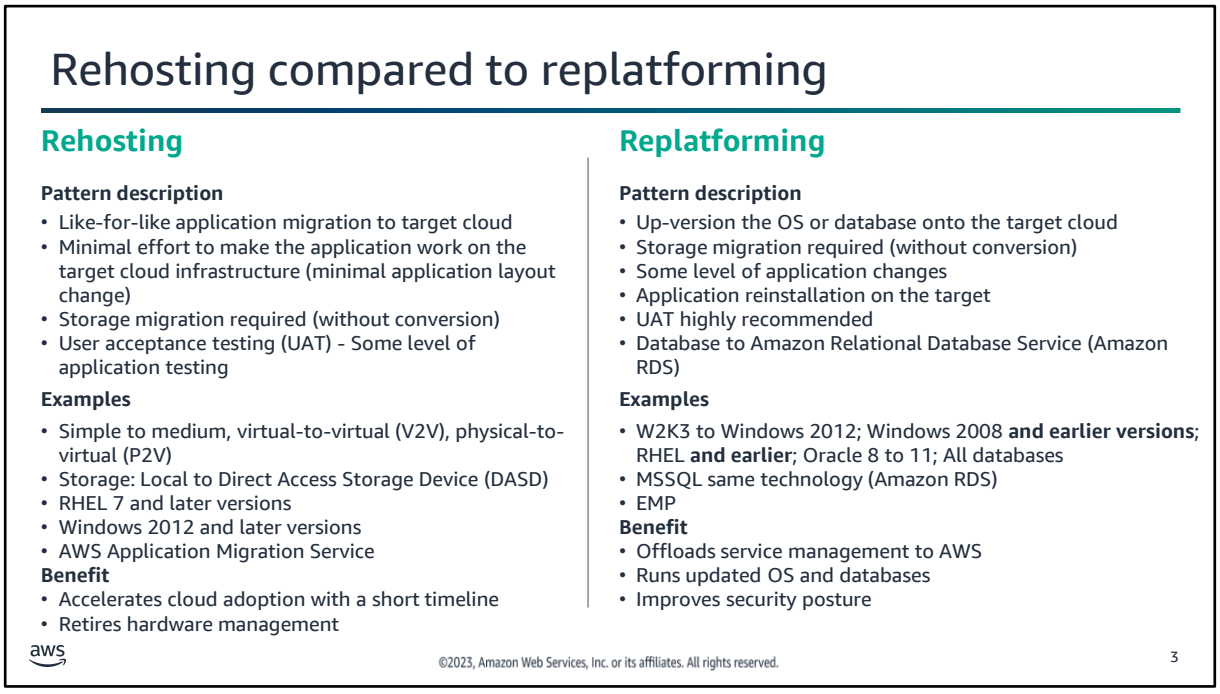
- Identify challenges you might encounter with migration and modernization projects.
- Identify the 7 R strategies and examples for each.
- Describe rehost and replatform migration methodologies.
- Identify migration opportunities to help customers quickly realize the value of the cloud.

Welcome to Module 2: Planning and Performing Large-Scale Migrations.



On completion of this module, you will be able to do the following:

- Highlight the strategy differences between rehosting and replatforming on AWS.
- Identify the benefits of using automation to perform large-scale migrations.
- Describe the Cloud Migration Factory workflow.
- Identify steps to prepare, perform, and monitor a migration using Cloud Migration Factory on AWS.
- Review best practices for migration at scale.
- Identify when and how to use AWS End-of-Support Migration Program (EMP).



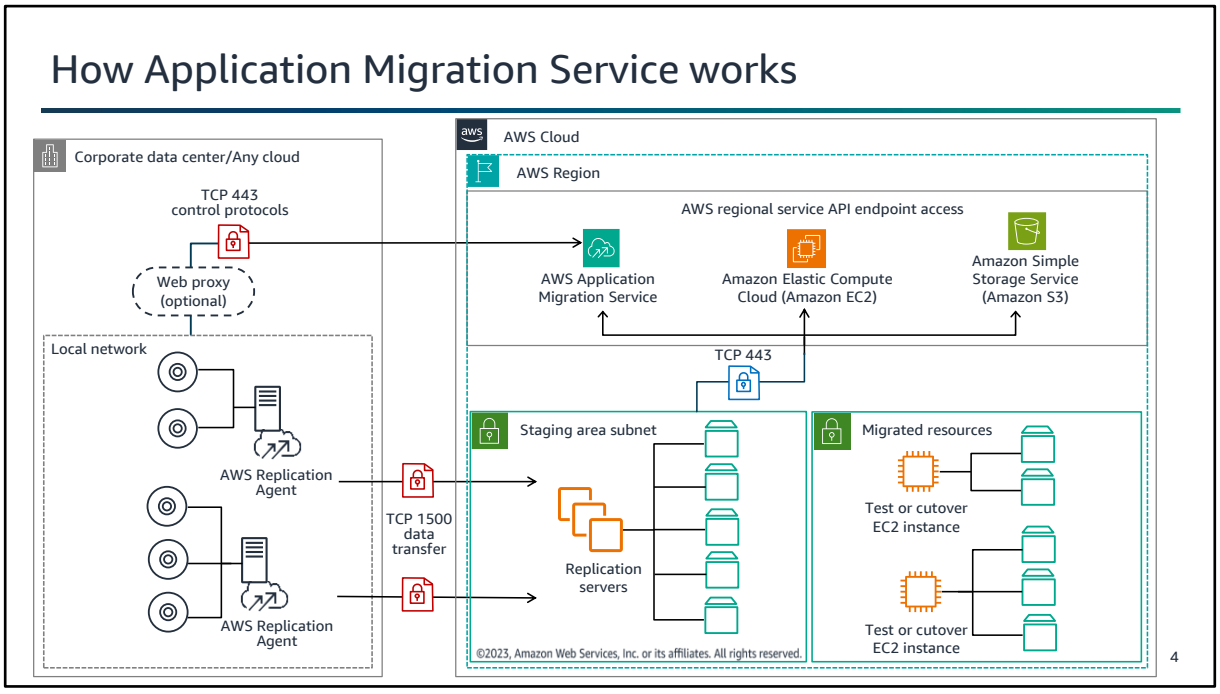
Let's start by comparing rehosting and replatforming migrations.

Rehosting using Application Migration Service pattern description
- Like-for-like application migration to target cloud
- Minimal effort to make the application work on the target cloud infrastructure (minimal application layout change)
- Storage migration required (without conversion)
- User acceptance testing (UAT)—Some level of application testing

Benefits
- Accelerates cloud adoption with a short timeline
- Retires hardware management

Replatforming using EMP pattern description
- Up-version the OS or database onto the target cloud
- Storage migration required (without conversion)
- Some level of application changes
- Application reinstallation on the target
- UAT highly recommended
- Database to Amazon Relational Database Service (Amazon RDS)

Benefits
- Offloads service management to AWS
- Runs updated OS and databases
- Improves security posture



This high-level architecture diagram illustrates how Application Migration Service can deliver rapid, reliable migration with minimal business disruption.

The source environment is on the left of the diagram. It can be composed of different types of applications or databases. Implementation begins by installing the AWS Replication Agent on your source servers. This does not require a reboot, nor does it impact system performance in any noticeable way. After you have installed agents, you can view and define replication settings for your servers on the Application Migration Service console. For more information, see https://console.aws.amazon.com/mgn/home.

Application Migration Service uses your settings to create and manage a staging area subnet in your selected AWS Region. The staging area subnet includes lightweight Amazon Elastic Compute Cloud (Amazon EC2) instances that act as replication servers and low-cost staging Amazon Elastic Block Store (Amazon EBS) volumes. Application Migration Service maps each source disk to a low-cost EBS volume in the staging area subnet. The replication servers receive data from the agent running on your source servers and write this data to the staging EBS volumes. One replication server can handle multiple source servers replicating concurrently. For more information about Amazon EC2, see https://aws.amazon.com/ec2. For more information about Amazon EBS, see *Amazon Elastic Block Store* at https://aws.amazon.com/ebs.

Your replicated data is compressed and encrypted in transit. It can be encrypted at rest using Amazon EBS encryption. You can control the data replication path using private connectivity options such as AWS Direct Connect or a virtual private network (VPN). Application Migration Service keeps your source servers up to date on AWS using continuous, asynchronous, block-level data replication. This

replication works for any application that runs on supported x86-based Windows and Linux operation systems. For information about AWS Direct Connect, see https://aws.amazon.com/directconnect.

Application Migration Service uses your defined launch settings to launch instances when you conduct non-disruptive tests or perform a cutover. Your launch settings determine where your instances are launched and how they will be provisioned. When you launch test or cutover instances, Application Migration Service automatically converts your source servers to boot and run on AWS. The conversion process is typically minutes, regardless of the size of your server, and runs on all your selected servers in parallel.

After confirming that your launched instances are operating properly on AWS, you can decommission your source servers. You can then choose to modernize your applications by using additional AWS services and capabilities.

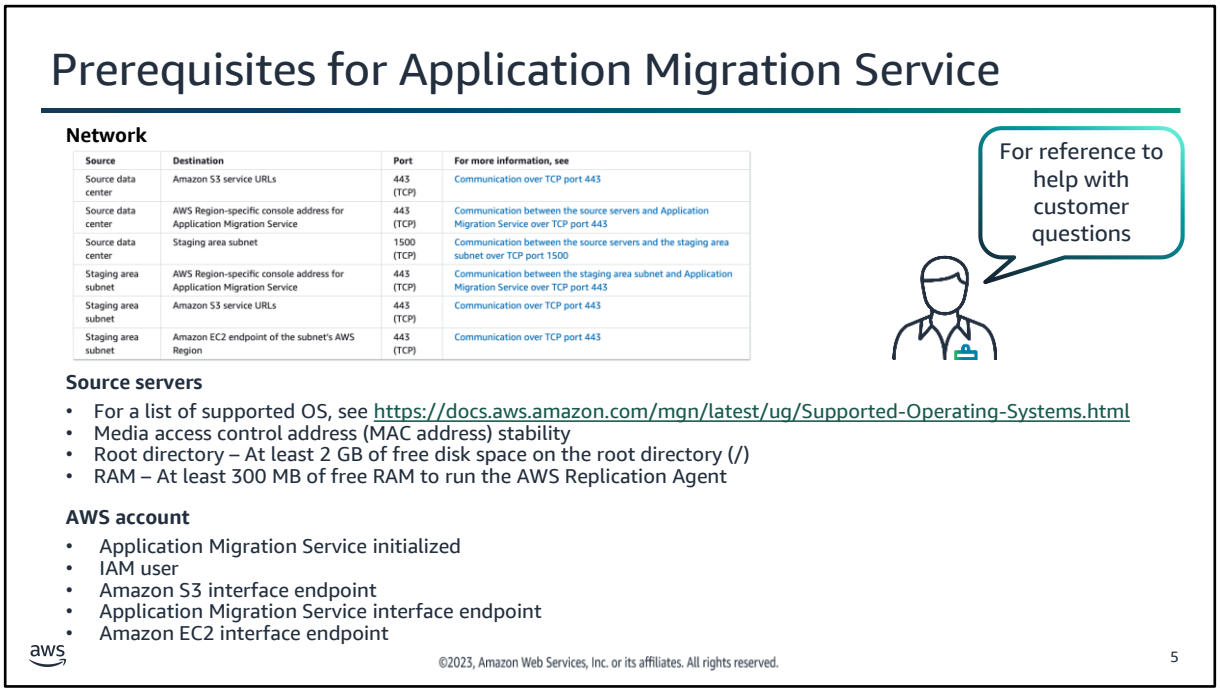
Agent install

The agent install includes the following:

- IAM user for replication
- The replication servers launched by Application Migration Service in your staging area subnet need to send data over TCP port 443 to the Application Migration Service API endpoint at https://mgn.{region}.amazonaws.com.
- The source servers that hold the AWS Replication Agent need to send data over TCP port 1500 to the replication servers in the staging area subnet. They also need to send data to Application Migration Service API endpoint at https://mgn.{region}.amazonaws.com. Replace “{region}” with the AWS Region code to which you will replicate.
- Each source server that is added to Application Migration Service must continuously communicate with Application Migration Service (https://mgn.{region}.amazonaws.com) over TCP port 443.
- Access to Amazon Simple Storage Service (Amazon S3) service URLs, which are required for downloading Application Migration Service software.

Application Migration Service cutover process

Application Migration Service doesn’t support cutover across accounts. It needs to be set up separately in every account that will host the migrated workload.



Here are prerequisites for Application Migration Service to help answer customer questions.

Network – For destination, ports, and more information, see *Network Requirements* at https://docs.aws.amazon.com/mgn/latest/ug/Network-Requirements.html.

- Source data center – Amazon S3 service URLs, communication over TCP port 443
- Source data center – AWS Region console address for Application Migration Service, communication between the source servers and the Application Migration Service over TCP port 443
- Source data center – Staging area subnet, communication between the source servers and the staging area subnet over TCP port 1500
- Staging area subnet – AWS Region console address for Application Migration Service, communication between the staging area subnet and Application Migration Service over TCP port 443
- Staging area subnet – Amazon S3 service URLs, communication over TCP port 443
- Staging area subnet – Amazon EC2 endpoint of the subnet's AWS Region, communication over TCP port 443

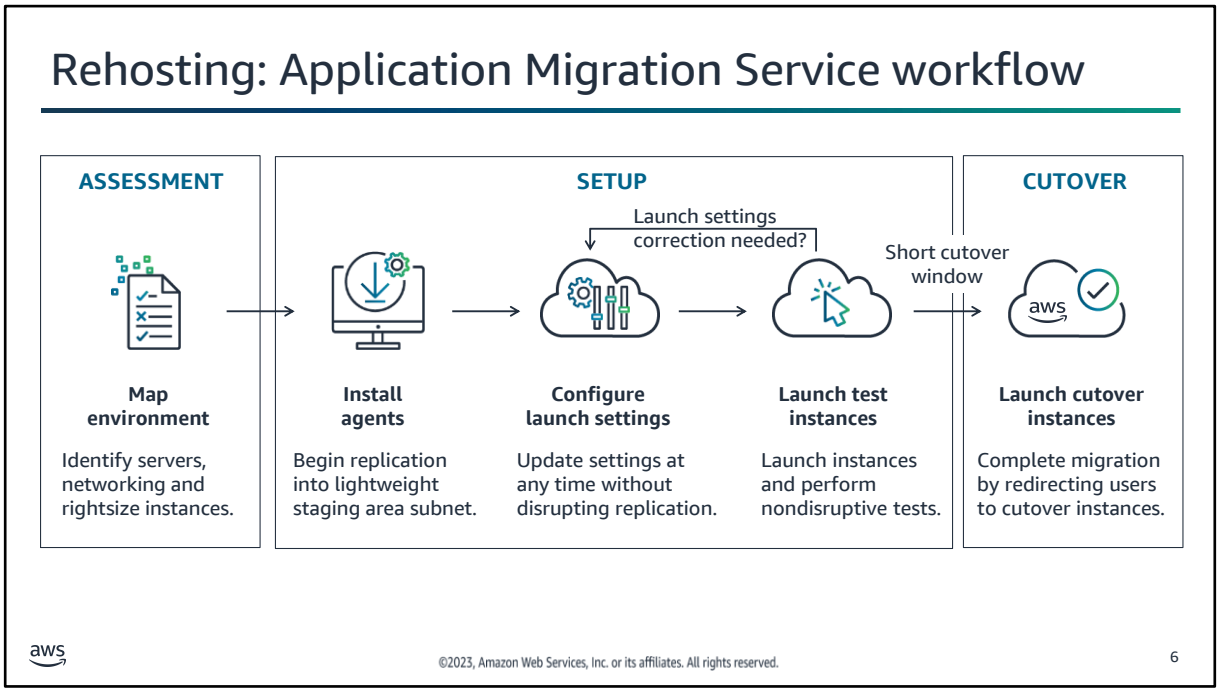
Source servers

- For a list of supported OSs, see https://docs.aws.amazon.com/mgn/latest/ug/Supported-Operating-Systems.html.
- Media access control address (MAC address) stability
- Root directory – At least 2 GB of free disk space on the root directory (/)
- RAM – At least 300 MB of free RAM to run the AWS Replication Agent

AWS account

- Application Migration Service initialized

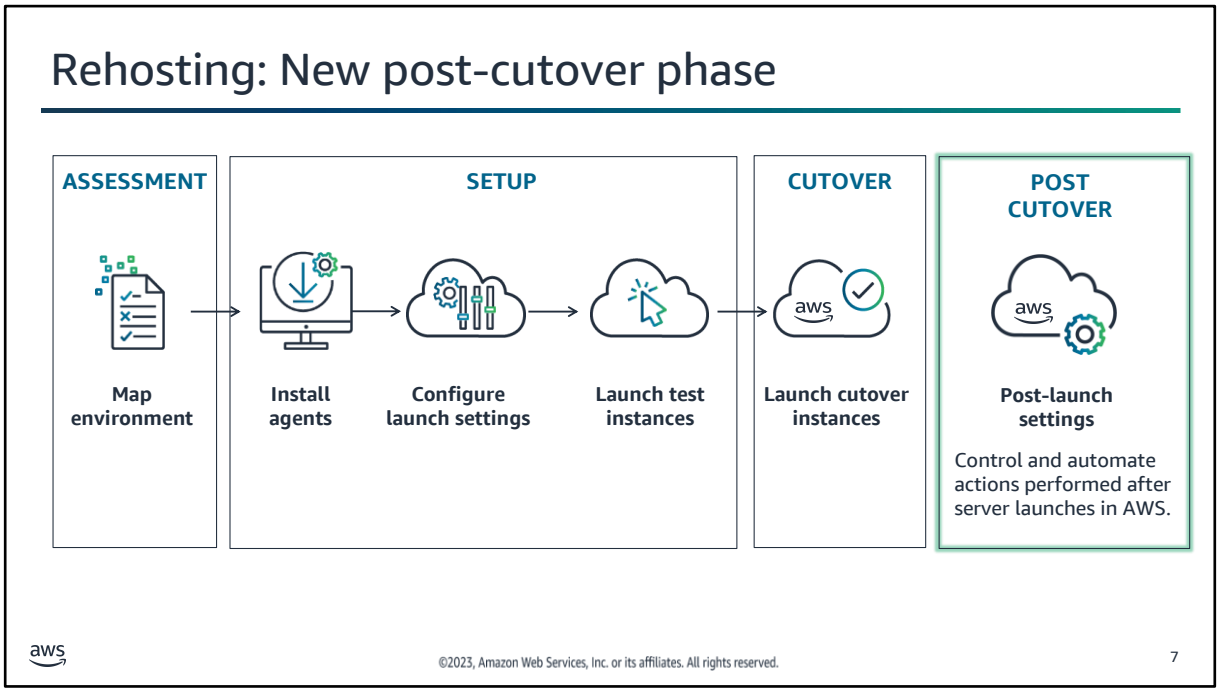
- IAM user
- Amazon S3 interface endpoint
- Application Migration Service interface endpoint
- Amazon EC2 interface endpoint



If you took the Migrations Essentials course, you should be familiar with the Application Migration Service workflow for rehosting.

The following list shows the tasks in each phase:

- **Assessment**
  - Map environment – In this stage, you identify servers and networking and rightsize instances.
- **Setup**
  - Install agents – Begin replication into lightweight staging area subnet.
  - Configure launch settings – Update settings at any time without disrupting replication.
  - Launch test instances – Launch instances and perform nondisruptive tests.
- **Cutover**
  - Launch cutover instances – Complete migration by redirecting users to cutover instances.

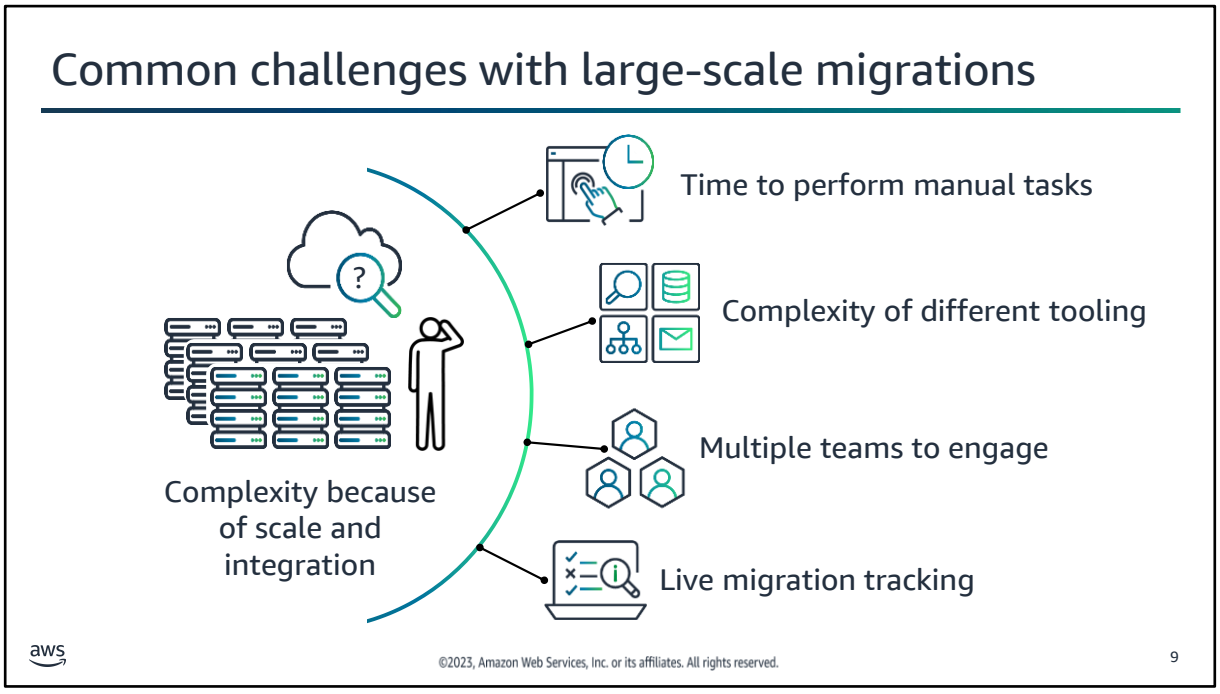


There is a new post-cutover phase that now offers post-launch settings to make it possible for you to control and automate actions performed after the server launches in AWS.

This is helpful, as now you can run any AWS Systems Manager documents after you migrate. For example, if you migrate the workload and you want to change or adjust the DNS or active directory tasks. It can also be used for scripts management. For more information on scripts management, see https://docs.aws.amazon.com/solutions/latest/cloud-migration-factory-on-aws/scripts-management.html.

To learn more about post-cutover actions, see *Migrating and automating patching at scale with AWS Application Migration Service* at https://aws.amazon.com/blogs/mt/migrating-and-automating-patching-at-scale-with-aws-application-migration-service.

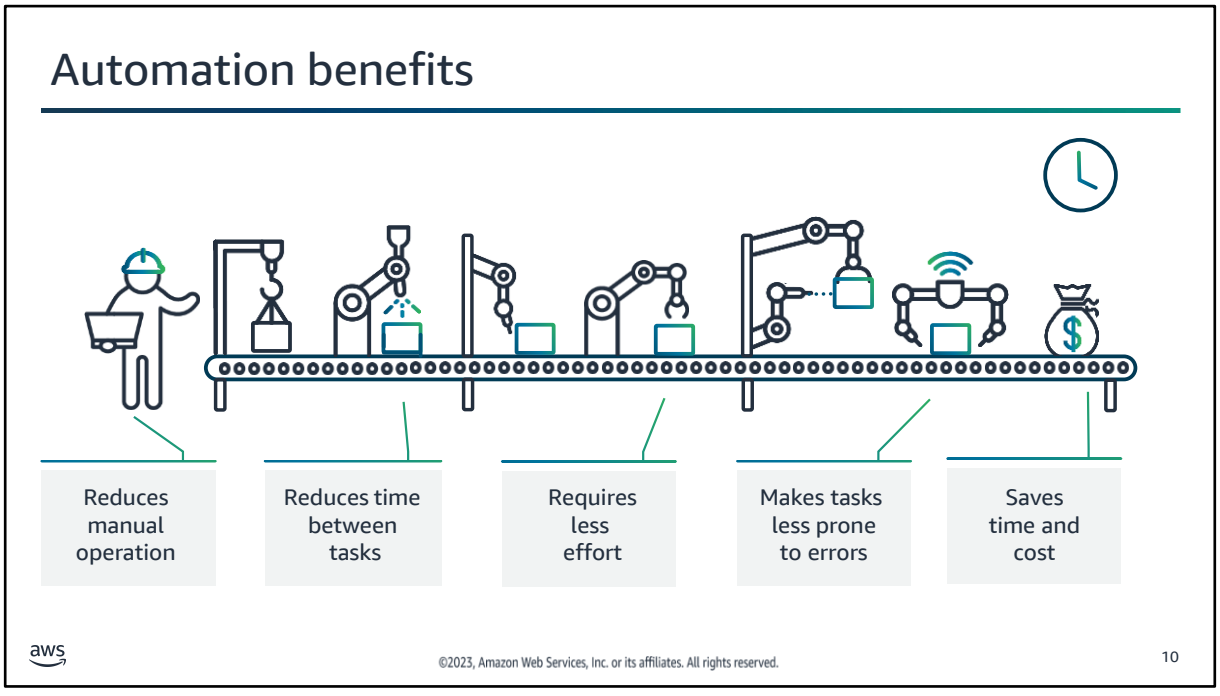
In this section, you learn about automating large-scale migrations by using Cloud Migration Factory on AWS.



Companies want to migrate their servers to AWS as efficiently as possible. This is a complex job, especially if you want to migrate thousands of servers in a short period, such as 6–12 months. Large-scale migrations present the following challenges:

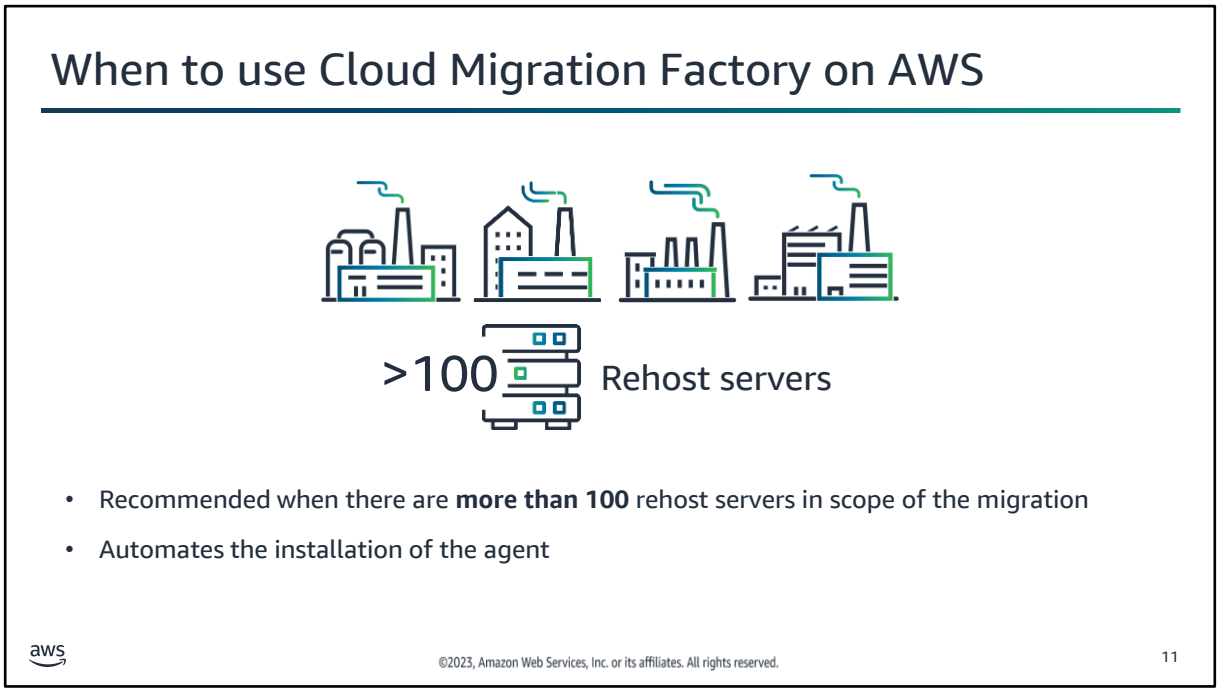
- Typically, large scale migrations have a lot of small, manual steps that need to take place per server. As an example, consider the prerequisites for installing the Application Migration Service agent. Although it might only take a few minutes to check per server, this can quickly snowball into a large amount of time after you multiply it by the total number of servers. Manual processes are slow and hard to scale.
- You also have the complexity of managing the end-to-end migration for all servers in scope alongside the integration of different tooling. If we focus on installing the Application Migration Service agent, it's likely that you will have multiple OSs you need to manage (both Linux and Windows). Then, you can have multiple authentication mechanisms to the server. Finally, you want to coordinate the activities aligned with your migration plan by focusing on waves that might involve different tools depending on the migration stage.
- In a large-scale migration, it's likely you will need to engage multiple teams to meet the desired business outcome. The following examples are typical teams and what they need to know:
    - Networks teams – Need to know which servers will be migrated in the following week to forecast the impact on the available bandwidth.
    - Support teams – Need to know which servers will be migrated in each wave to understand the impact.
    - Executive sponsors – Need to know how many servers have been migrated and if the program is on track.
    - Decommissioning teams – Need to know which servers have been migrated so that their processes can commence. The decommissioning teams become even more important

given that there's moving pieces in a migration and the data changes frequently. For example, a server needs to be descoped from a migration wave because of service issues a week prior to the migration.



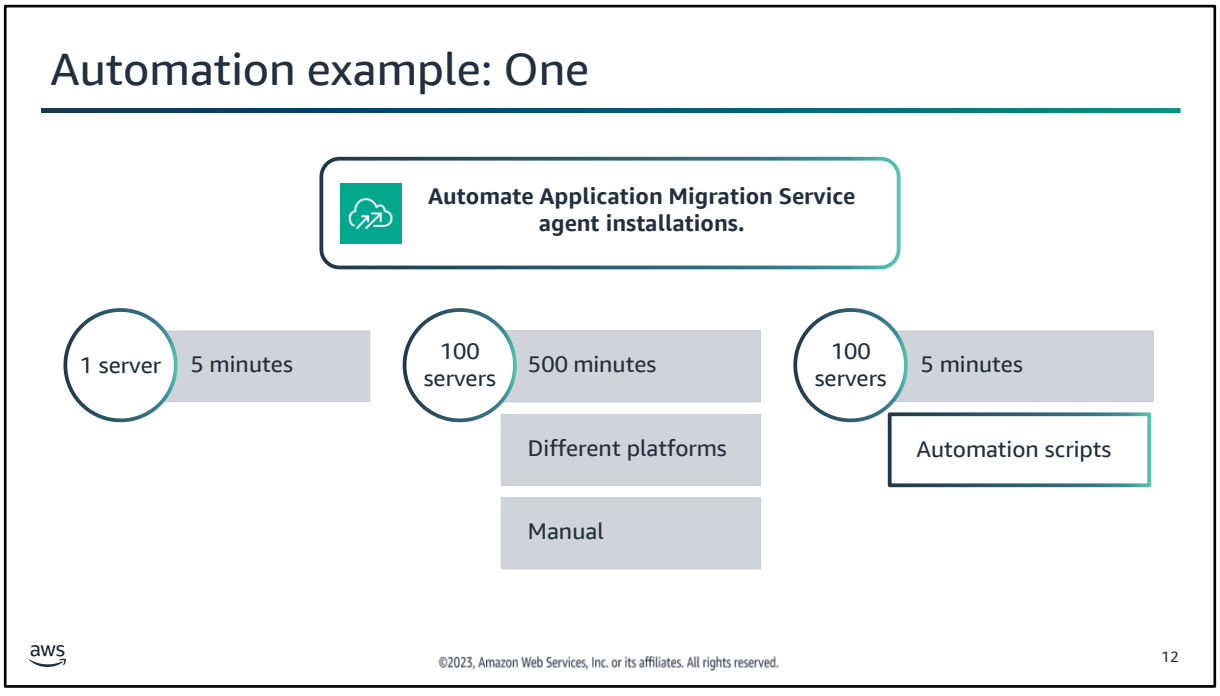
A complete lift-and-shift migration can require dozens of tasks in each phase. Each task could take a few minutes to run. When running manually, tasks are prone to error. They risk being missed, skipped, or run in the wrong order.

Automating migration tasks reduces manual operation. It removes the time between tasks, which makes migrations more efficient. You can complete large-scale migrations with less effort, which saves time and cost. All these benefits can be realized by using Cloud Migration Factory on AWS.



Use Cloud Migration Factory on AWS when there are more than 100 rehost servers as part of the migration.

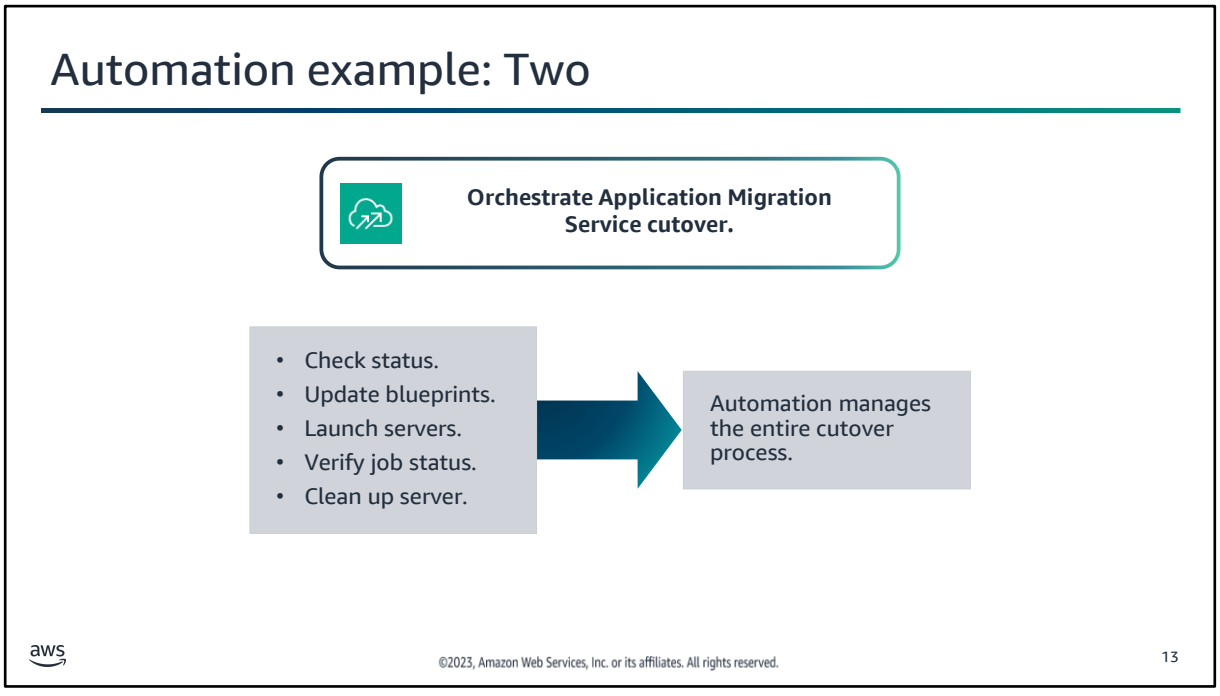
Agent installation
Installing the agent on one server takes about 5 minutes. If you have 100 servers running Windows and Linux for 10 different Cloud Migration Factory projects, the process could take over 500 minutes. The agent on the source machines can be installed in many ways. An automation script reduces the agent installation time from 500 minutes to less than 5 minutes of operator time. It works for Windows and Linux OSs and any target AWS account.



Automation scripts help you save significant time and effort with large-scale migration tasks, as shown in this example. Automate an Application Migration Service agent installation for more than 100 servers. Installing the Application Migration Service agent on one server takes about 5 minutes.

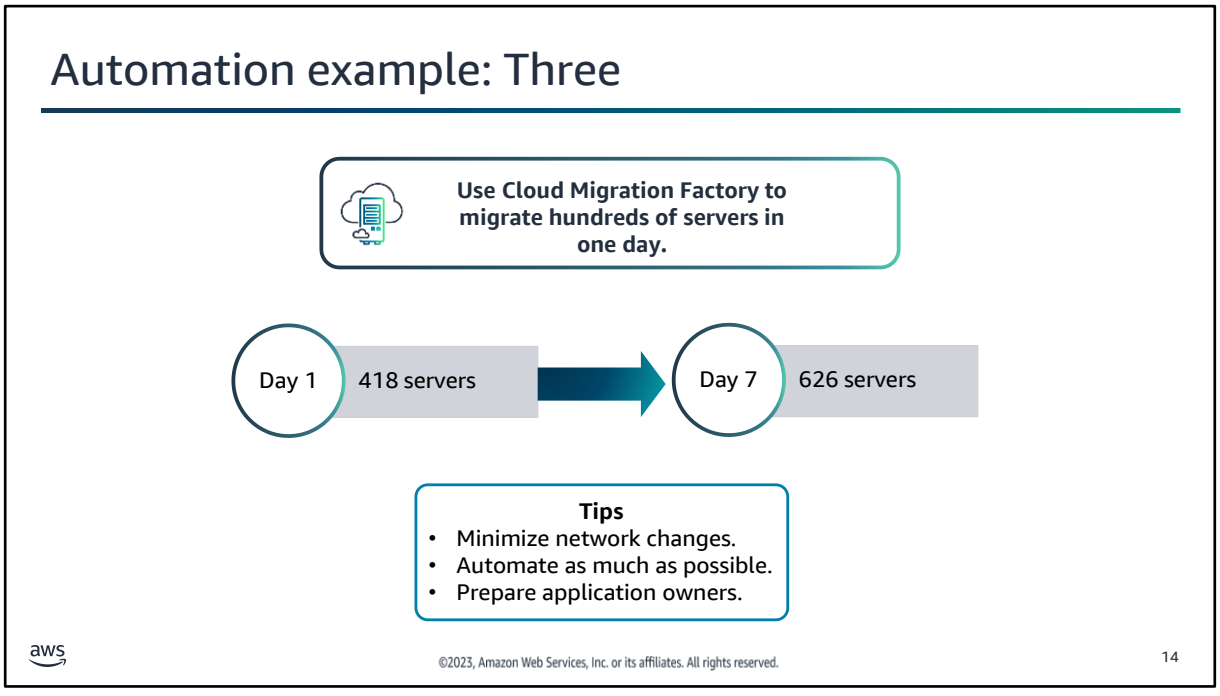
If you have 100 servers running Windows and Linux for 10 different MGN projects, the process could take over 500 minutes. The agent on the source machines can be installed in many ways.

An automation script reduces the agent installation time from 500 minutes to less than 5 minutes of operator time. It works for Windows and Linux OSs and any target AWS account.



This example describes how automation scripts help customers during the Application Migration Service cutover process.

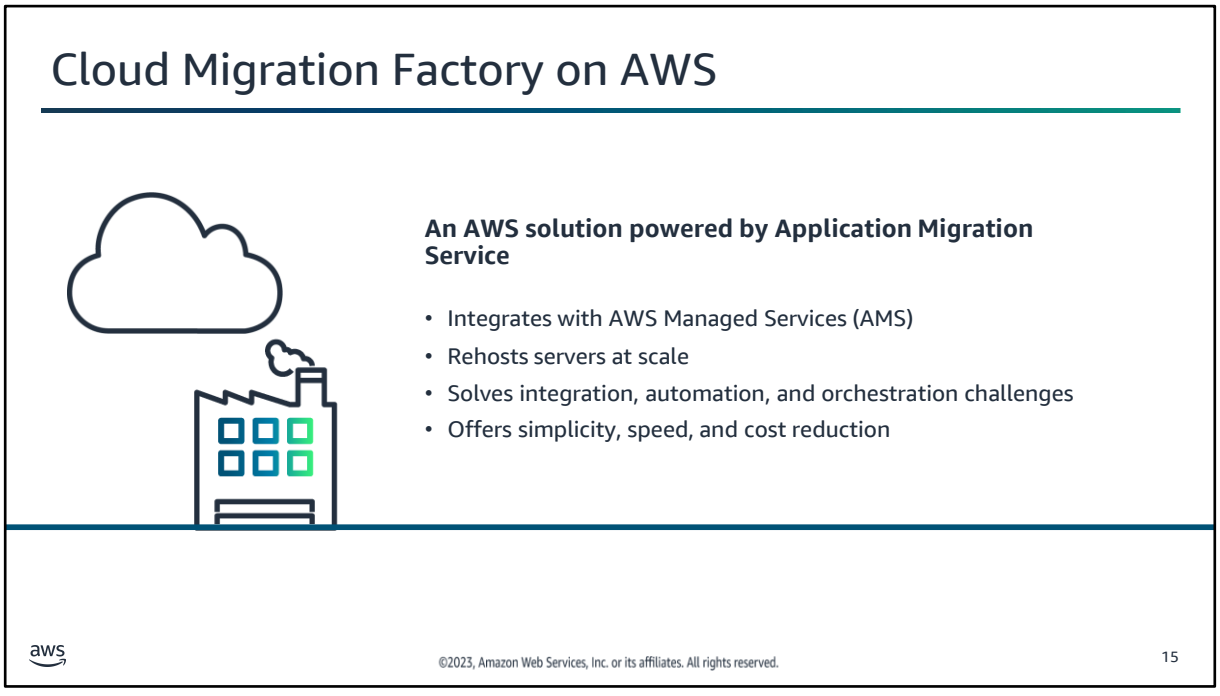
You can orchestrate the Application Migration Service cutover process. The cutover process involves many tasks, such as checking replication status, checking server status, updating the blueprint, launching servers in cutover mode, verifying job status, and cleaning up the server. It is a long process for just one server. For hundreds of servers in a single cutover, it could be an enormous undertaking. This solution automates and orchestrates the entire process for you.



Automation scripts can help you save your customers' time and effort.

Here is a customer example: An AWS Partner used Cloud Migration Factory to help a customer migrate 1,400 servers. They successfully migrated 418 servers in one day and 626 servers in one day one week later. You can migrate 600 servers in a few hours.

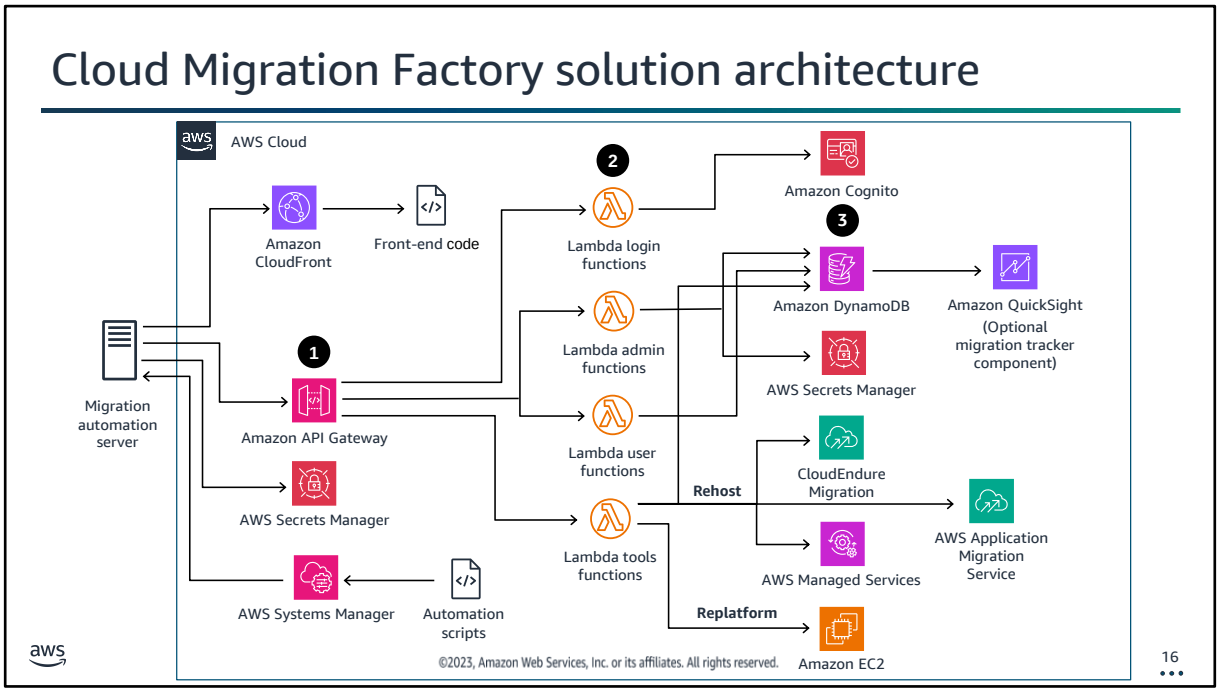
The following tips can help make that happen:
- Minimize changes, including IP and security groups.
- Automate migration with Cloud Migration Factory.
- Train application owners for running their applications in the cloud.



What is Cloud Migration Factory? Cloud Migration Factory is an orchestration platform powered by Application Migration Service. It is integrated with AWS Managed Services (AMS) for rehosting servers to the AWS Cloud at scale.

Cloud Migration Factory offers the following benefits:

- Helps solve the integration, automation, and orchestration challenges typically encountered in large-scale migrations
- Offers simplicity, speed, and cost reduction to cloud migrations by automating Application Migration Service migration activities



As we mentioned, the Cloud Migration Factory solution is an orchestration platform powered by Application Migration Service for rehosting servers to AWS at scale. Here is an example of how Cloud Migration Factory works.

Step 1: Amazon API Gateway receives migration requests from the migration automation server through Rest APIs.

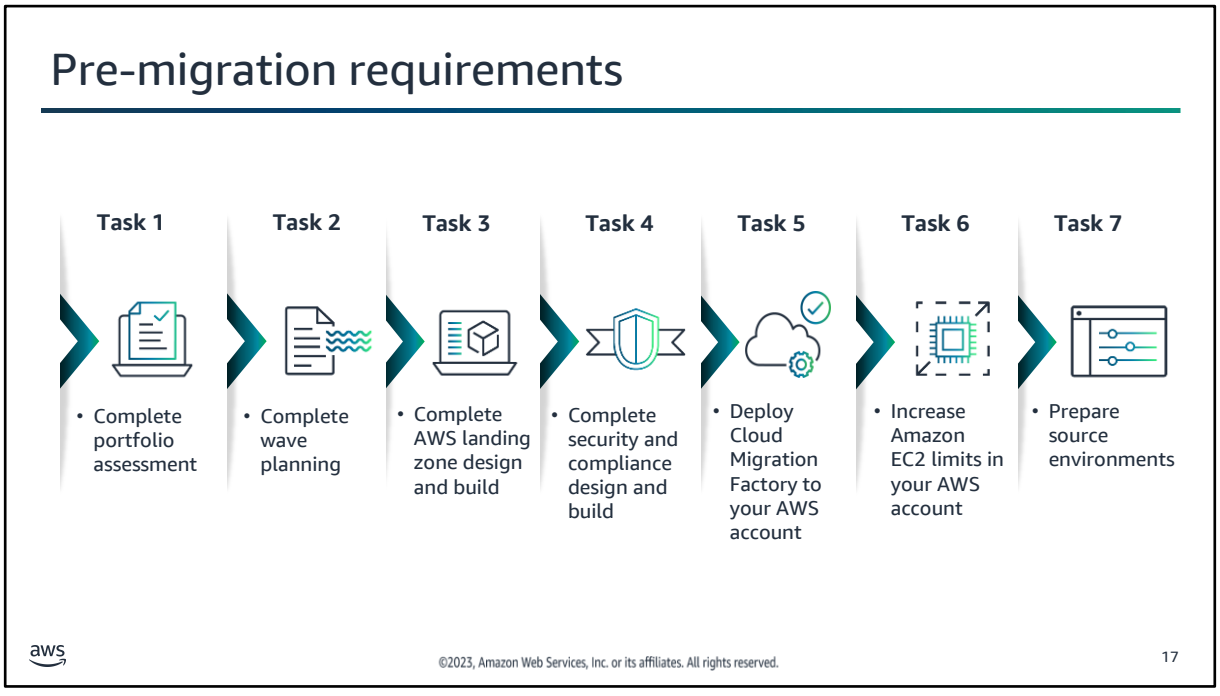
Step 2: AWS Lambda functions provide the necessary services for you to log in to the web interface, perform the necessary administrative functions to manage the migration, and connect to third-party APIs to automate the migration process.

- The user Lambda function ingests the migration metadata into an Amazon DynamoDB table. Standard HTTP status codes are returned to you through the Rest API from the API Gateway. An Amazon Cognito user pool authenticates the user to the web interface and Rest APIs. You can optionally configure it to authenticate against external SAML identity providers.
- The Lambda tools functions process external Rest APIs and call the CloudEndure Migration and Application Migration Service for AWS migration. The Lambda tools functions also call Amazon EC2 for launching EC2 instances and AWS Systems Manager to run automation scripts on the migration automation server.

Step 3: The migration metadata stored in DynamoDB is routed to the Application Migration Service API to initiate a rehost migration job and launch servers. If your migration pattern is replatform to Amazon EC2, the tools Lambda function launch AWS CloudFormation templates in the target AWS account to launch the Amazon EC2 instances.

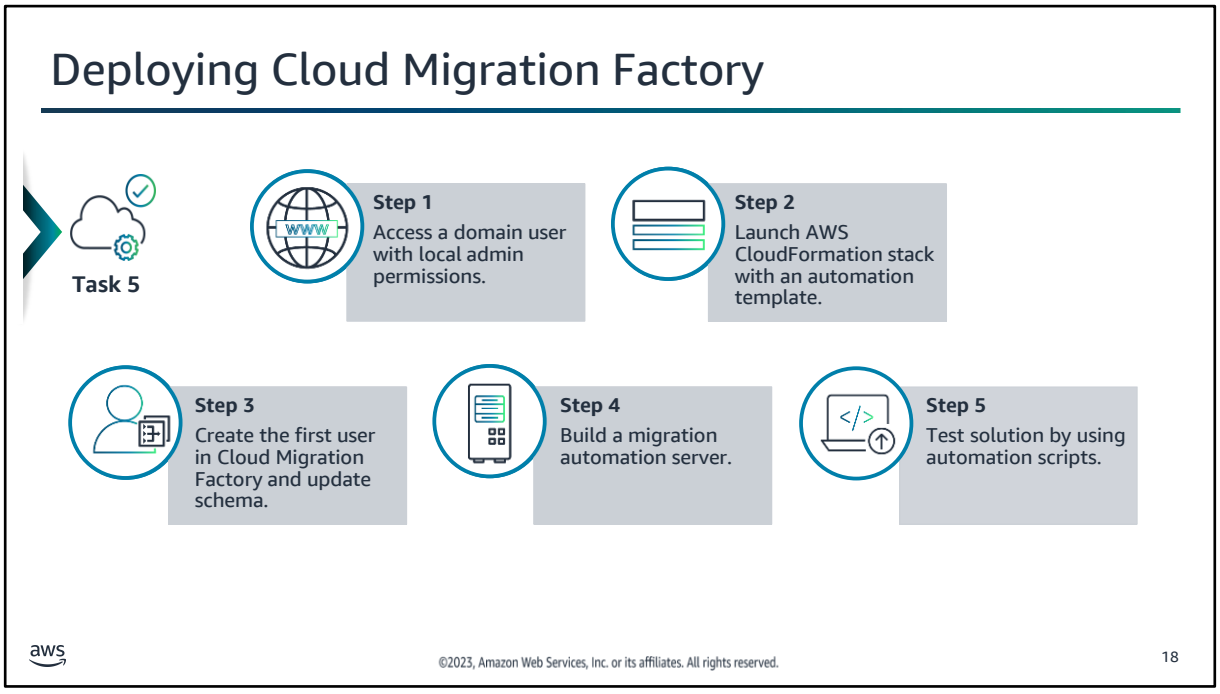
This solution also deploys an optional migration tracker component that tracks the progress of your migration. AWS Glue retrieves migration metadata from the Cloud Migration Factory DynamoDB table and exports the metadata to Amazon S3. You can then create visualizations and build a dashboard to view the progress of the migration.

For more information, see *Cloud Migration Factory on AWS* at https://aws.amazon.com/solutions/implementations/cloud-migration-factory-on-aws. For estimated costs, see https://docs.aws.amazon.com/solutions/latest/cloud-migration-factory-on-aws/cost.html.



Before starting Cloud Migration Factory, you must complete the following tasks:

1. Complete portfolio assessment. This task helps you estimate the level of effort and costs associated with the migration and prioritize application migrations.
2. Complete wave planning. Split servers into groups of 25–35 servers and repeat the migration activities.
3. Complete AWS landing zone design and build. The AWS infrastructure is ready with a secure, cloud-based environment.
4. Complete security and compliance operating model design and build.
5. Deploy the Cloud Migration Factory solution to your AWS account.
6. Increase Amazon EC2 limits in the target AWS account to handle migrations. Contact support for this request.
7. Prepare your source environments. This task is described later in this module.



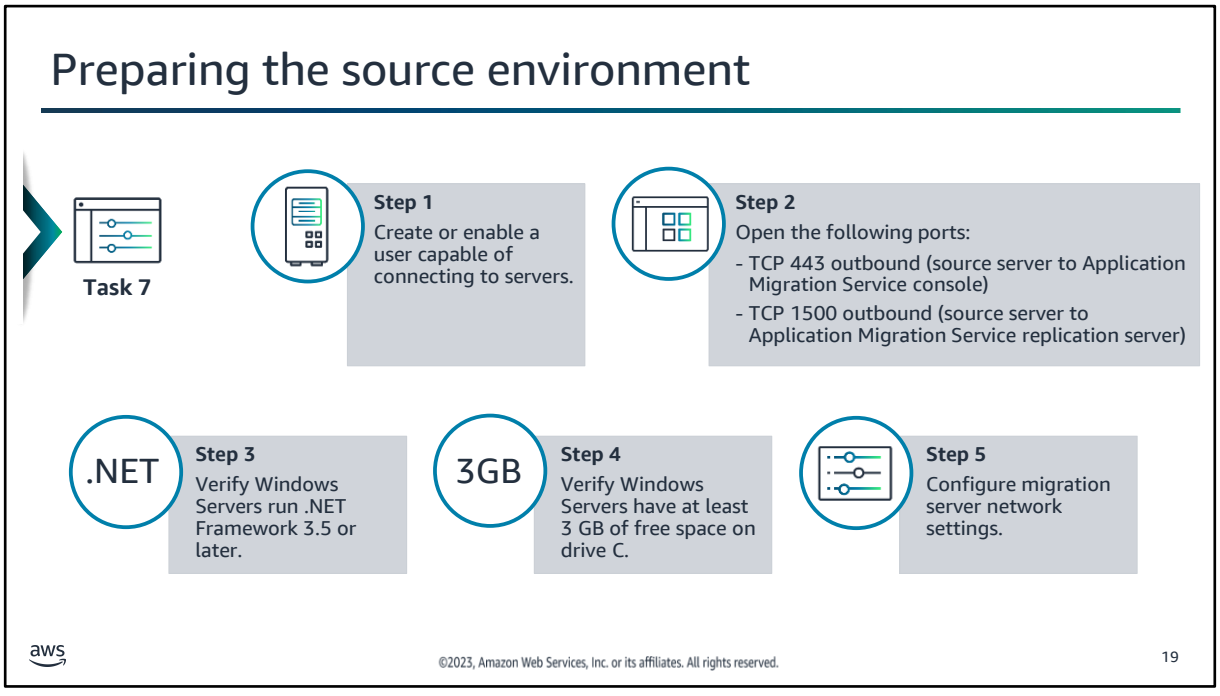
Let's look closer at task 5. Before starting, you must have a domain user for Windows and Linux (sudo permissions) servers. The user must have local administrator permissions to the source servers. Verify that you have the necessary permissions or coordinate with the appropriate person in the organization with domain permissions before launching the solution.

Deploy the solution in an existing AWS account. To build Cloud Migration Factory, complete the following steps:

1. Launch the AWS CloudFormation stack into your AWS account*. During the launch, supply the stack name, application name, environment name, version, security group ID (if not using the default security group), and subnet ID (if not using the default subnet).
2. Create the initial user in Cloud Migration Factory, update the schema, and change the default Application Migration Service project name.
3. Build a migration server. The migration server runs the automation tasks. You can build the server in the AWS Cloud or on premises. The migration server runs Windows Server 2016 or later. It's where you install Python and the required packages. Download the sample automation scripts from the GitHub repository and customize the configuration to match the outputs that were produced by the AWS CloudFormation deployment.
4. Test the solution using the provided automation scripts. To perform a test run, you need a user with administrator permissions on the source servers. Conduct a test run of the migration automation.

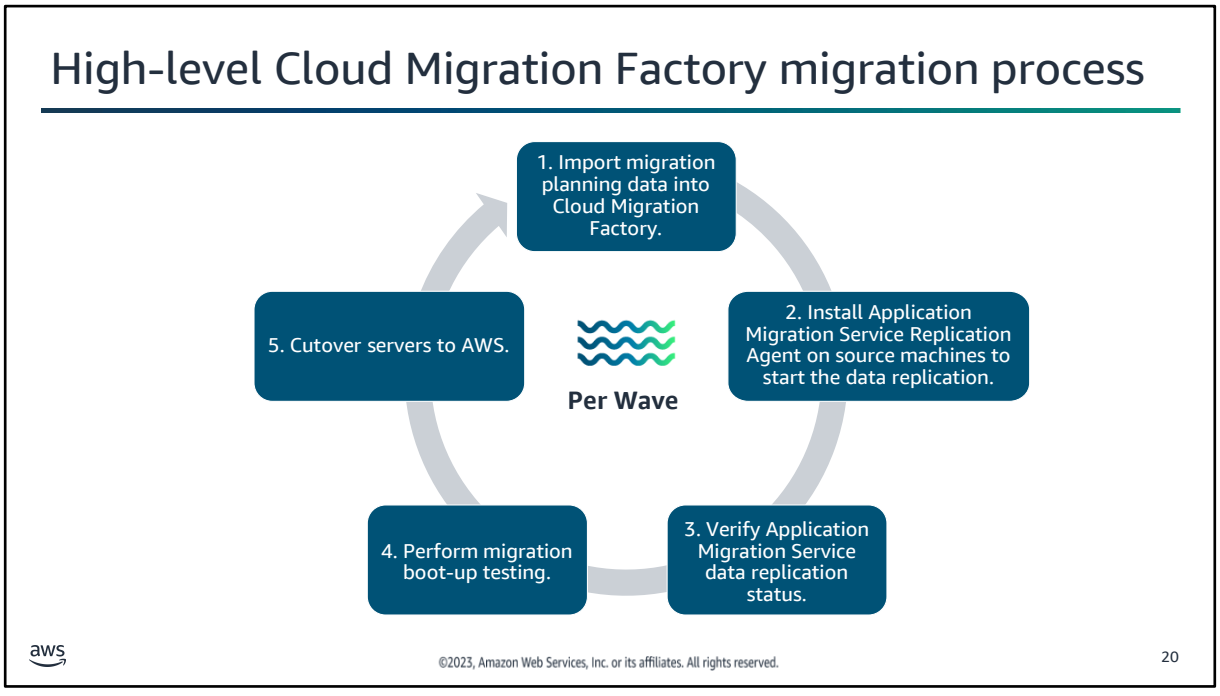
*The administrator can install Cloud Migration Factory into the target account or another management account. To do so, they must create the appropriate AWS Identity and Access Management (IAM) permissions to Application Migration Service on the target account.

For more information about Cloud Migration Factory deployment, see *Deploy the Solution* at https://docs.aws.amazon.com/solutions/latest/cloud-migration-factory-on-aws/deployment.html.



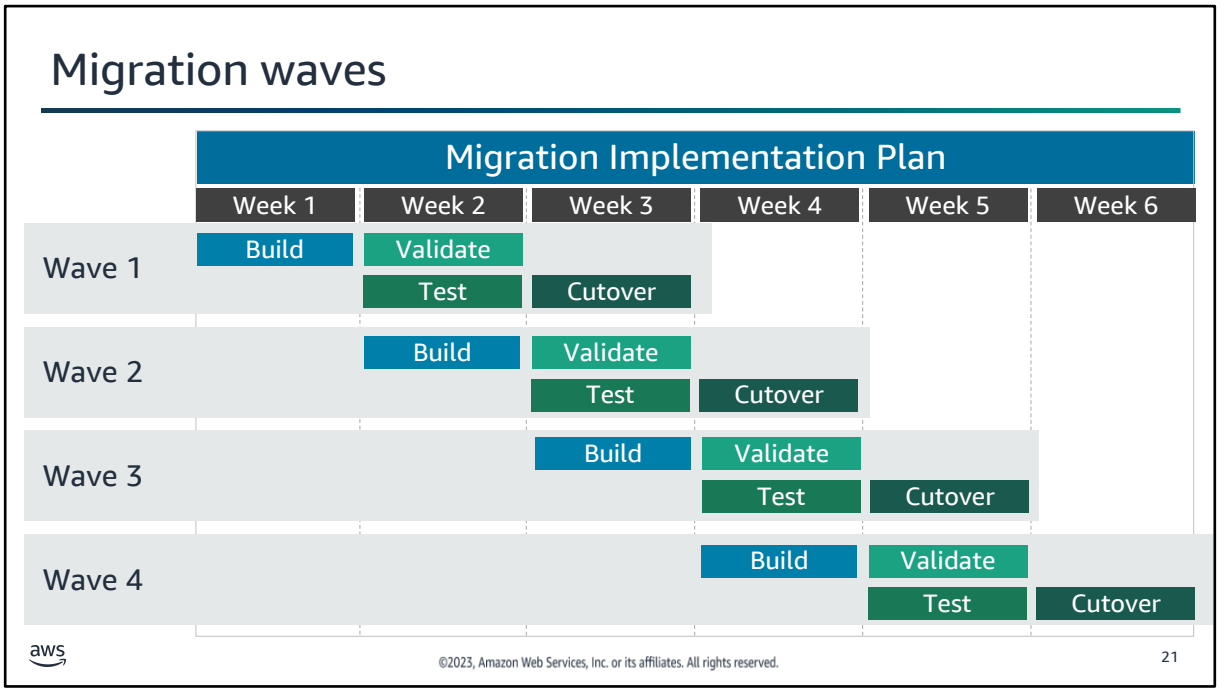
Let's also take a closer look at task 7—preparing your source environments.
To prepare the source environment, complete the following steps:

1. Create or enable a user (or user in domain) that can connect to all the servers. For example, a domain user or an SSH (Secure Shell) user with pseudo permissions that has access to all servers.
2. Open the following firewall ports:
    - TCP 443 outbound from source server to Application Migration Service console
    - TCP 1500 outbound from source server to Application Migration Service replication server in the target virtual private cloud (VPC) on AWS
3. Verify that Windows Servers run .NET Framework 3.5 or later.
4. Verify that Windows Servers have at least 3 GB of free space on drive C.
5. Configure the migration server network settings to connect to AWS and the source servers.



Cloud Migration Factory comes with a predefined migration process as shown.
Before starting, be sure to verify prerequisites. During the migration implementation phase, the migration team is responsible for running predefined tasks that automate the migration process. These tasks can include the following:

1. Import migration planning data into Cloud Migration Factory.
2. Install the Application Migration Service Replication Agent to the source machines for a given wave.
3. Verify the Application Migration Service data replication status.
4. Perform migration boot-up testing.
5. Schedule a window for cutover to AWS.

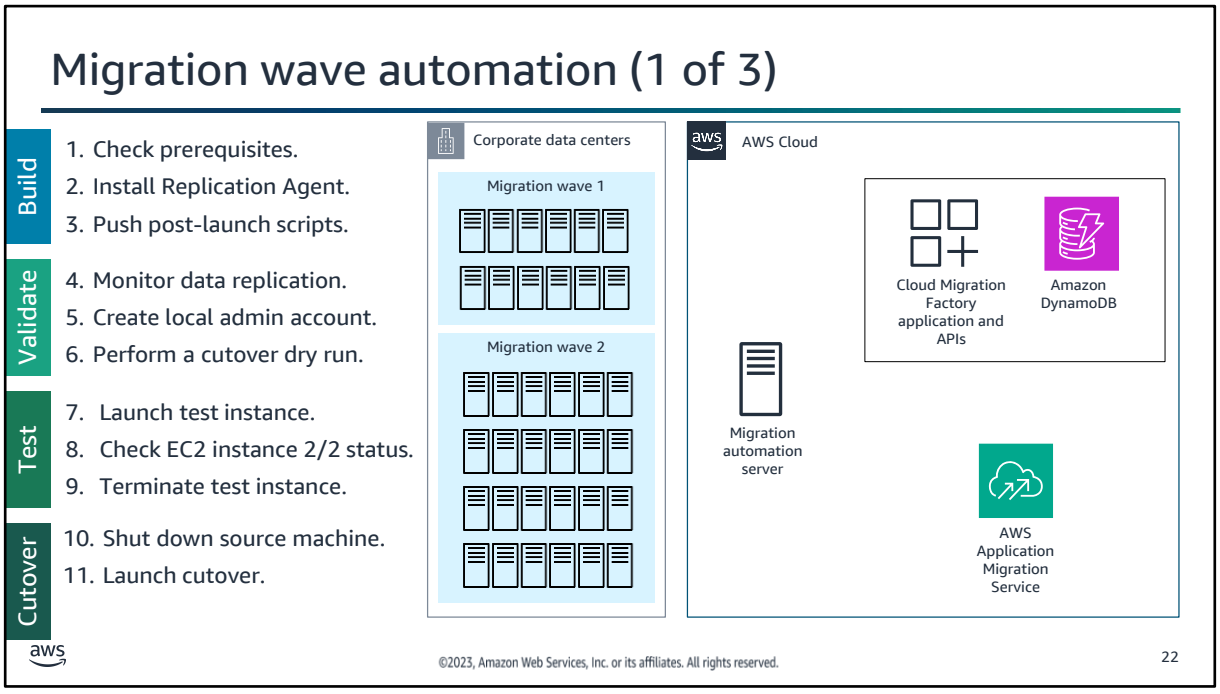


Migration tasks are scheduled in *waves*. Each wave consists of a group of applications and servers with the same cutover date. Waves that contain 25–35 servers are easier to manage and troubleshoot.

Each wave should be completed in a predefined period. For example, in the following period shown:

- Week 1 is the build stage.
- Week 2 is the validate and boot-up testing stage.
- Week 3 is the cutover stage.
- Weeks 4–6 repeat as necessary. The waves run in parallel.

It's important to remember that although an initial migration plan would have been produced during the first phase of a migration, changes can happen. Be open to change because new data can be uncovered during the migration automation.



Let's switch focus to migration tasks that are automated in each wave—tasks 1–11.

As mentioned, Cloud Migration Factory completes on a wave basis instead of a server basis. This means that the automation scripts can be applied on a per wave basis. This is important for large-scale migrations. During the planning phases, you formed migration waves that outline the servers that should be migrated together, such as dependencies with low latency requirements.

Cloud Migration Factory implementation automation scripts are used for the following tasks:

Build phase:

- Checking prerequisites for the migration—For example, does the server have enough storage and required network connectivity for the Application Migration Service Replication Agent?
- Installing Application Migration Service Replication Agent for multiple servers on a per wave basis
- Pushing the post-launch scripts to your source servers that will be run after the cutover has taken place

Validate phase:

- Monitoring replication for all servers within a given migration wave
- Creating a local administrator account to debug the migration
- Performing a migration dry run for your migration wave

Test phase:

- Testing the EC2 instance launch for the entire migration wave
- Performing 2/2 (system status and instance status) health checks on the EC2 instances
- Terminating test instances

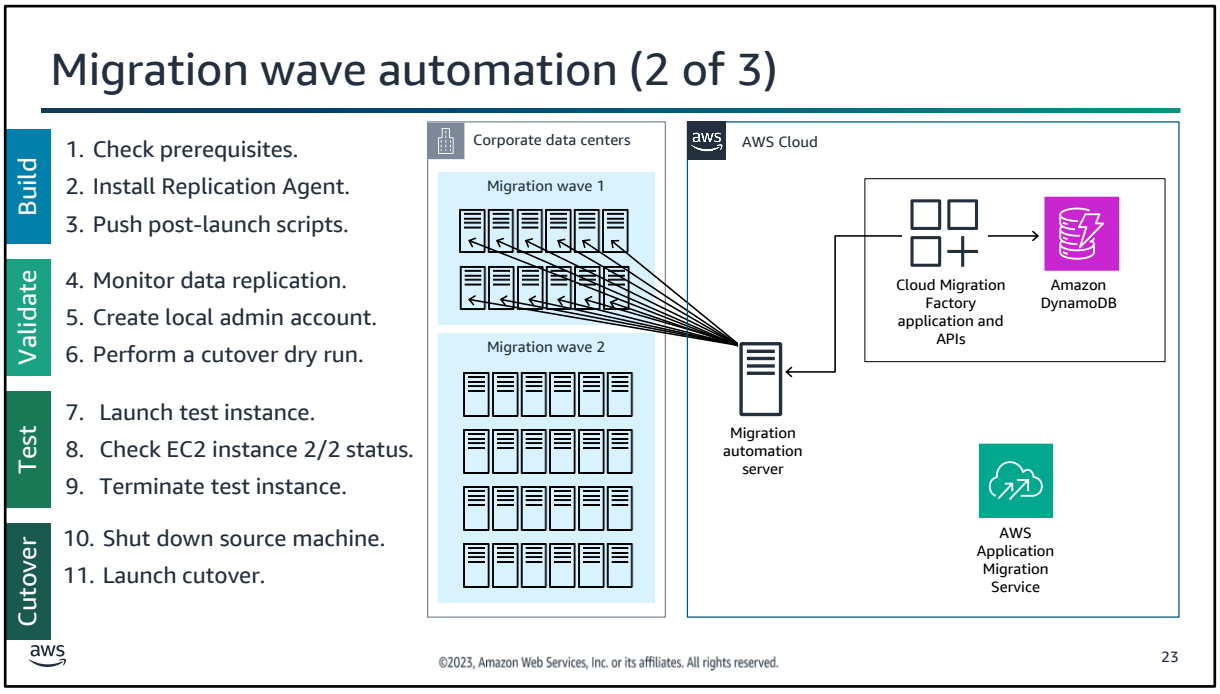
Cutover phase:

- Shutting down servers at the source location

- Orchestrating the Application Migration Service cutover process

These automation scripts help you reduce complexity and save significant time and effort in your large-scale migration. For example:

- **Automating the Application Migration Service Replication Agent installation for more than 100 servers:** Installing the agent on one server takes about 5 minutes. However, if you have 100 servers running Microsoft Windows and Linux, the process could take over 8 hours. The automation script reduces the agent installation time from 500 minutes to less than 5 minutes of operator time. It works for both Windows and Linux OSs.
- **Orchestrating the migration cutover process:** This process involves checking the replication status, checking the server status, updating the launch settings, launching servers in cutover mode, verifying job status, cleaning up the server, and many other tasks. It is a long process even for one server. If you have hundreds of servers in a single cutover, the process is complex and takes even longer. The Cloud Migration Factory solution automates and orchestrates the entire process.

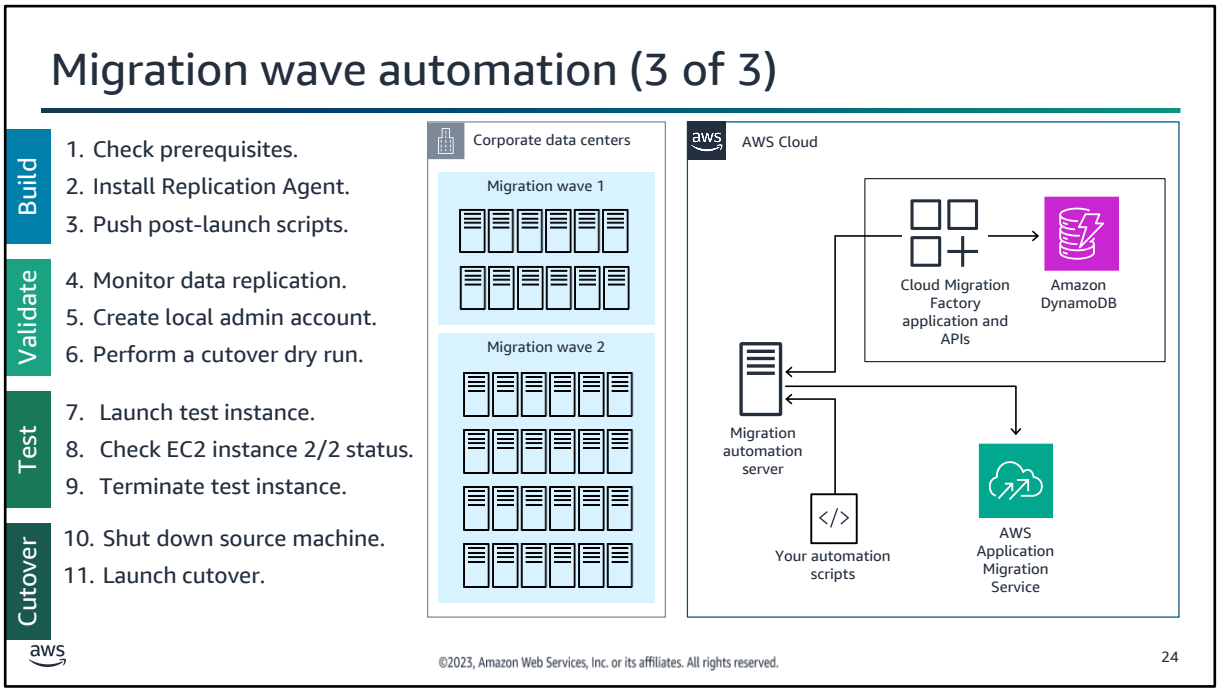


On the right side of the screen is a high-level diagram, outlining how Cloud Migration Factory works, alongside the associated activity on the left.

- DynamoDB is a key component of Cloud Migration Factory. It stores the migration metadata, such as your migration wave plans, source server information, and migration status per server.
- DynamoDB is accessed through the Cloud Migration Factory web application, which is a React JavaScript portal. This portal is also exposed through APIs to enable automation.
- The migration automation server is used for the tasks that require connecting to source machines.

Using the installation of the Application Migration Service replication again as an example:

- First, the migration operative will run the associated command on the migration automation server.
- This will initiate a call to the factory web application to obtain the data relating to the servers within the specified wave. In this example, it's wave 1 that contains 12 servers.
- With the data obtained, such as fully qualified domain name (FQDN) and OS, it will then connect to the 12 source machines using the credentials provided and install the agent.
- Finally, the automation server updates the factory application with a progress status for the 12 servers.



Now, you're monitoring the data replication for the 12 servers that are in migration wave 1.

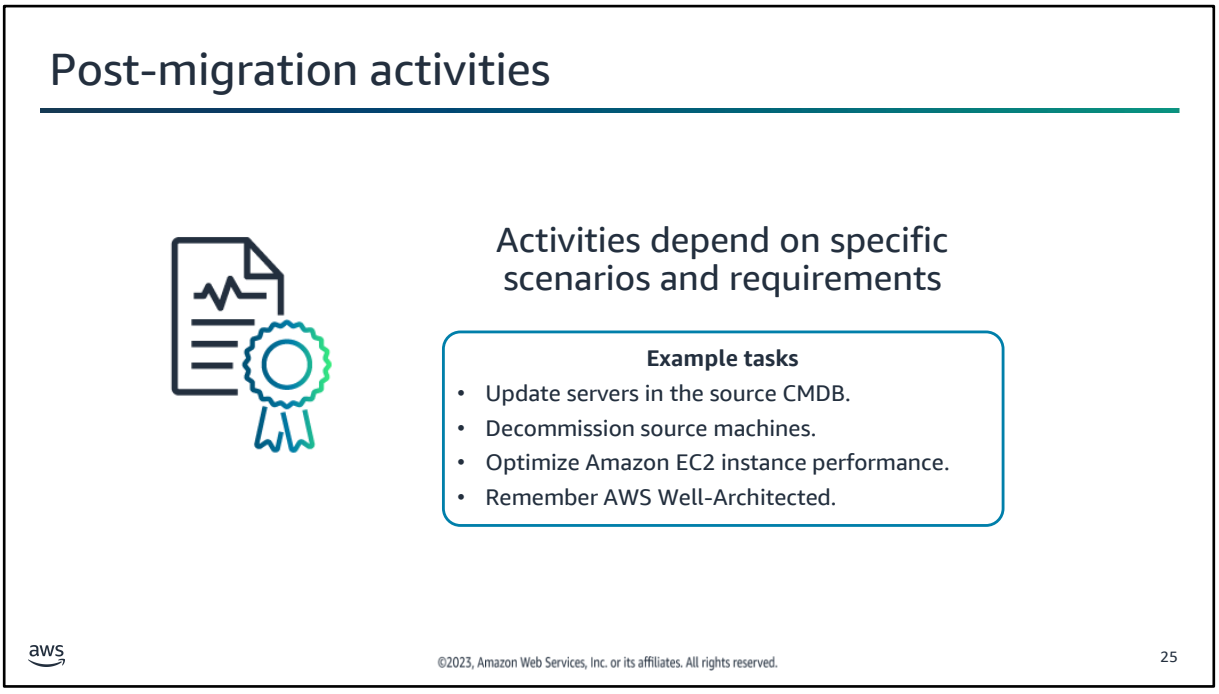
- After the command initiates on the migration automation server, the server makes an API call to the Cloud Migration Factory application to obtain data about the migration wave (like fully qualified domain name or FQDN)
- Using this information, the migration automation server makes API calls to AWS Application Migration Service service to capture the data replication status.
- The data replication status is subsequently updated in the Cloud Migration Factory application.

What about other activities that are specific to my organization?

- A key theme regarding Cloud Migration Factory is to automate manual tasks to decrease the complexity and increase the velocity of the migration. If you have specific requirements, consider creating and embedding scripts into Cloud Migration Factory to meet your requirements. Consider developing additional scripts to provide the end-to-end automation based on requirements for your organization. When all integrations are automated, you have a velocity rehost migration at scale.

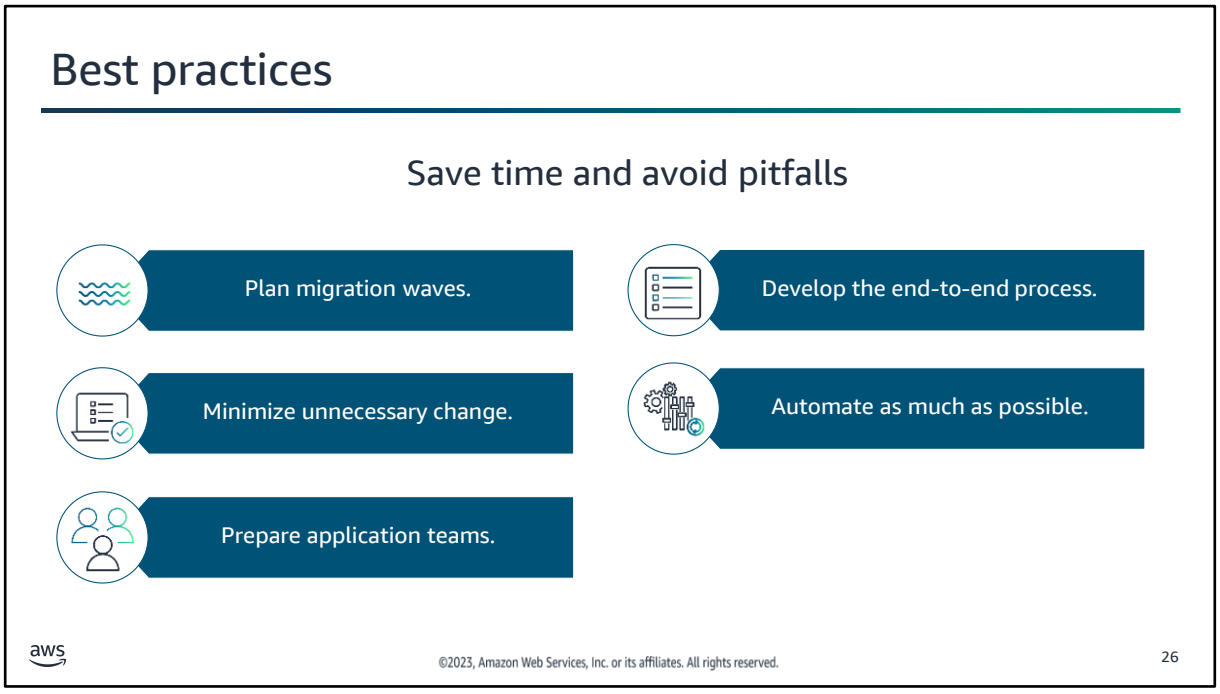
Consider the following examples of automation:

- Integrate existing configuration management database (CMDB) (or other inventory sources) directly into Cloud Migration Factory so that data changes in your CMDB are automatically reflected. This helps keep the factory data accurate.
- Use CyberArk to store and retrieve Active Directory credentials required for authenticating to the source servers. This reduces the manual task that would have been required for each wave had it not been automated.



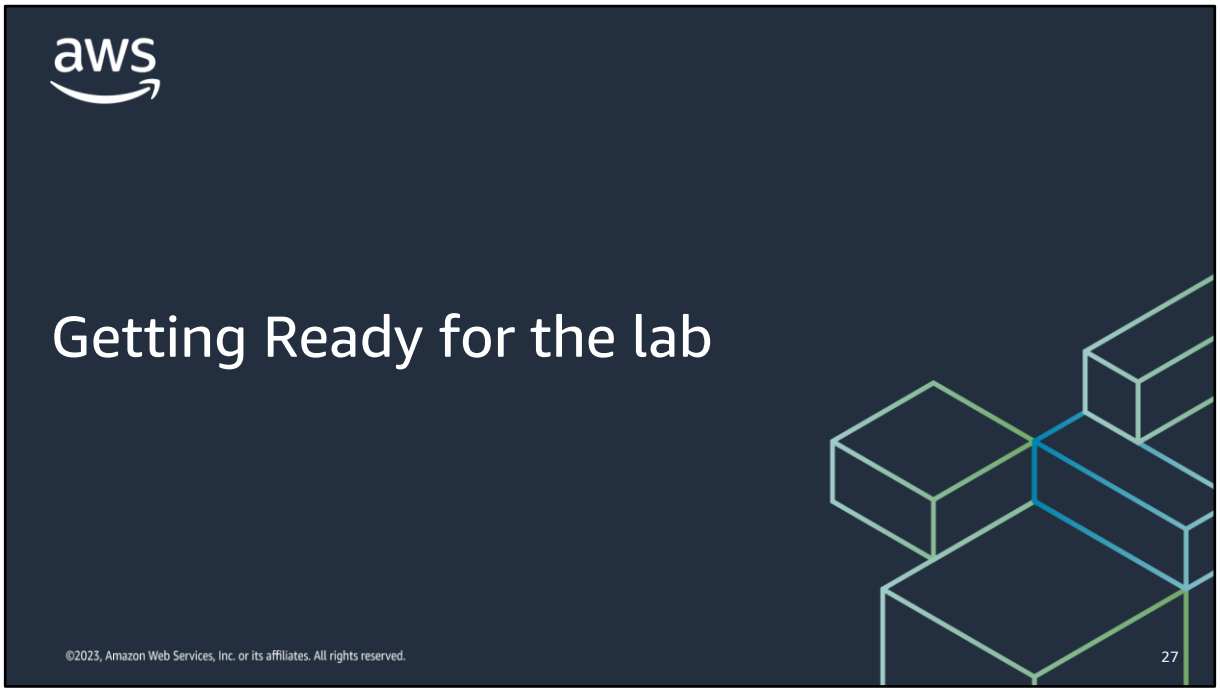
When the migration activities are complete, post-migration activities can take place. The specific migration scenarios and customer requirements determine the activities. Tasks and activities might include the following:

- Update servers in the source CMDB. The servers are migrated by lift-and-shift to preserve server names and shut down source servers. You can update the CMDB to reflect the same server running in the cloud.
- Decommission source machines. When the migration is complete and testing is successful, you can start to decommission the source machines.
- Optimize performance for the target Amazon EC2 instances. Start to measure performance metrics against standard tests, and match the size, type, and parameters of the EC2 instances where the services run to benefit performance at the lowest possible cost.
- Remember AWS Well-Architected. For more information, see *AWS Well-Architected* at https://aws.amazon.com/architecture/well-architected.

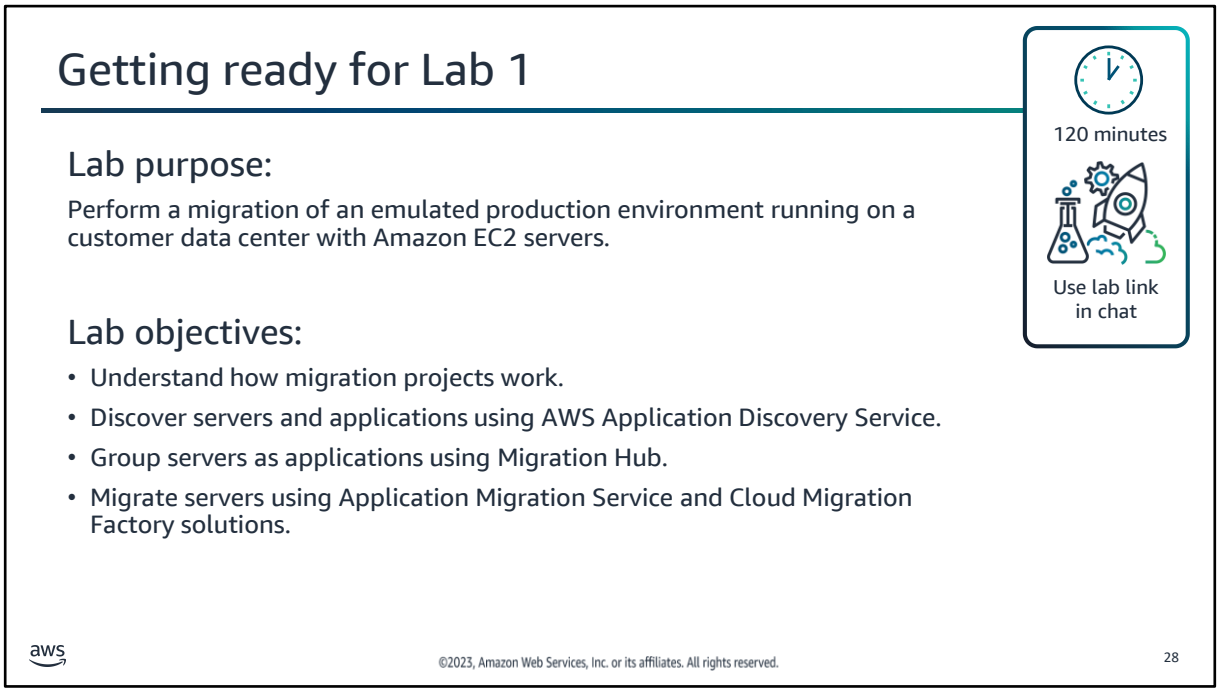


You can use the following migration best practices for your migrations at scale:

- Plan migration waves in advance. This saves time when configuring the migration factory and validating the automation steps.
- Develop end-to-end process and automation in the early stage. You can customize the process and the automation to more precisely match the customer's requirements.
- Minimize unnecessary change. This is not a wish list. Rehost the servers while changing less to reduce troubleshooting.
- Automate migration with Cloud Migration Factory as much as possible. The more you use automation, the faster the migration.
- Prepare the application teams for migration and document as much as possible.

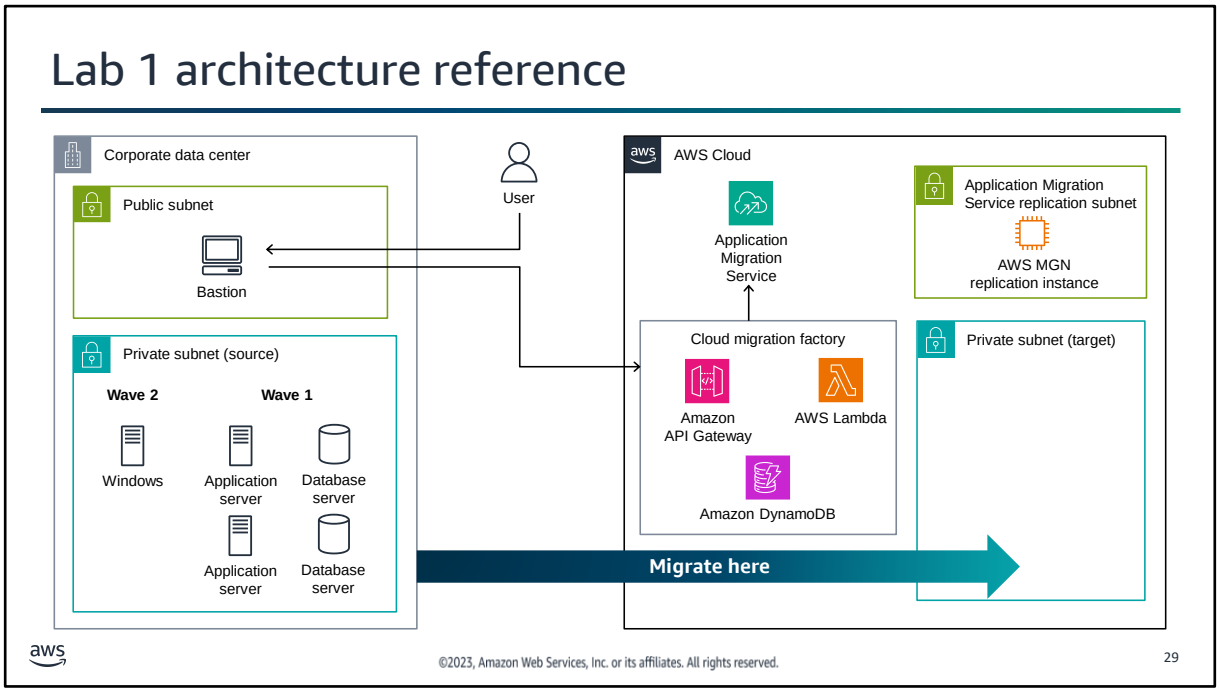


This section will review the details you need to know before starting Lab 1.



In this lab, you will do the following:

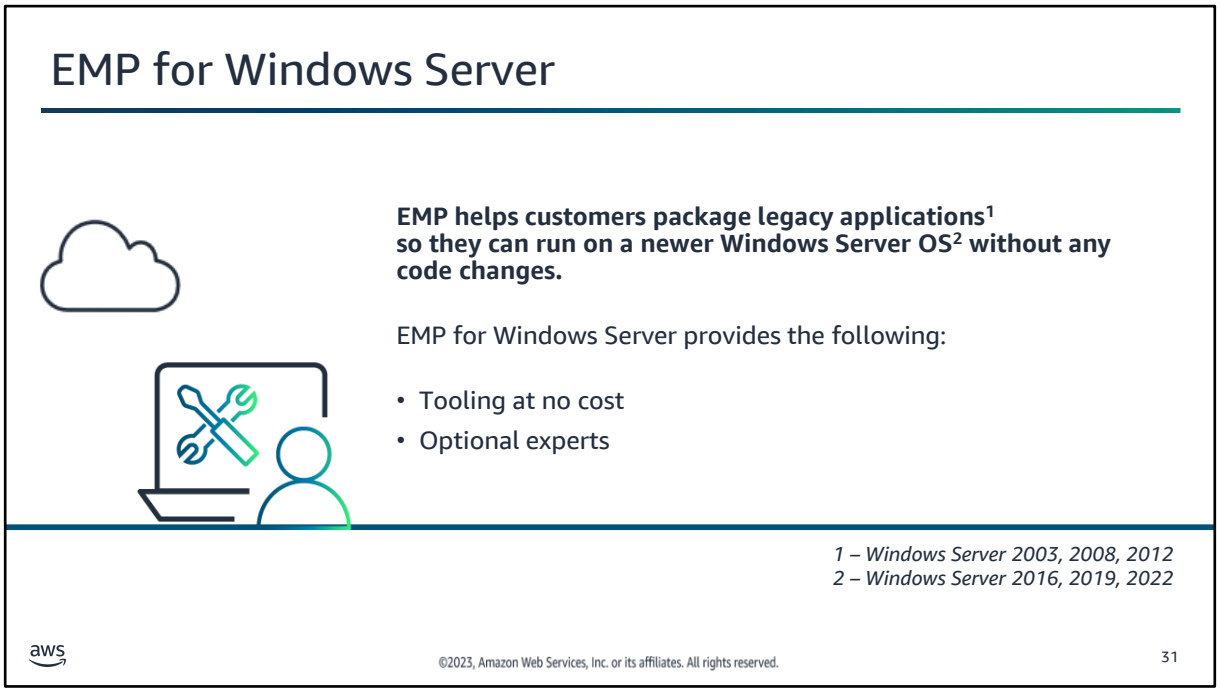
- Understand how migration projects work.
- Discover servers and applications using AWS Application Discovery Service.
- Group servers as applications using AWS Migration Hub.
- Migrate servers with Application Migration Service and Cloud Migration Factory solutions.



Here is the architecture you will be using for lab 1.
The corporate data center in this example has two subnets. The first subnet is a public subnet with a bastion host. The second subnet is a private subnet or source that has two waves of servers. Wave 1 has two application servers and two database servers. Wave 2 has one windows server.
You will access the lab environment through the bastion host. This is where you will perform all the configuration and setup for the Cloud Migration Factory. The goal is to migrate two waves of servers from their source environment to the new AWS environment using Application Migration Service and Cloud Migration Factory solutions. This Cloud Migration Factory solution consists of three services: an API Gateway, Lambda, and DynamoDB. Note, although these are the three used here, the solution could use several other services.

The target AWS environment will have two subnets. The first is a staging subnet with the Application Migration Service replication subnet and the AWS MGN replication instance. The second subnet is a private subnet you will use for the target to migrate the two waves of servers.

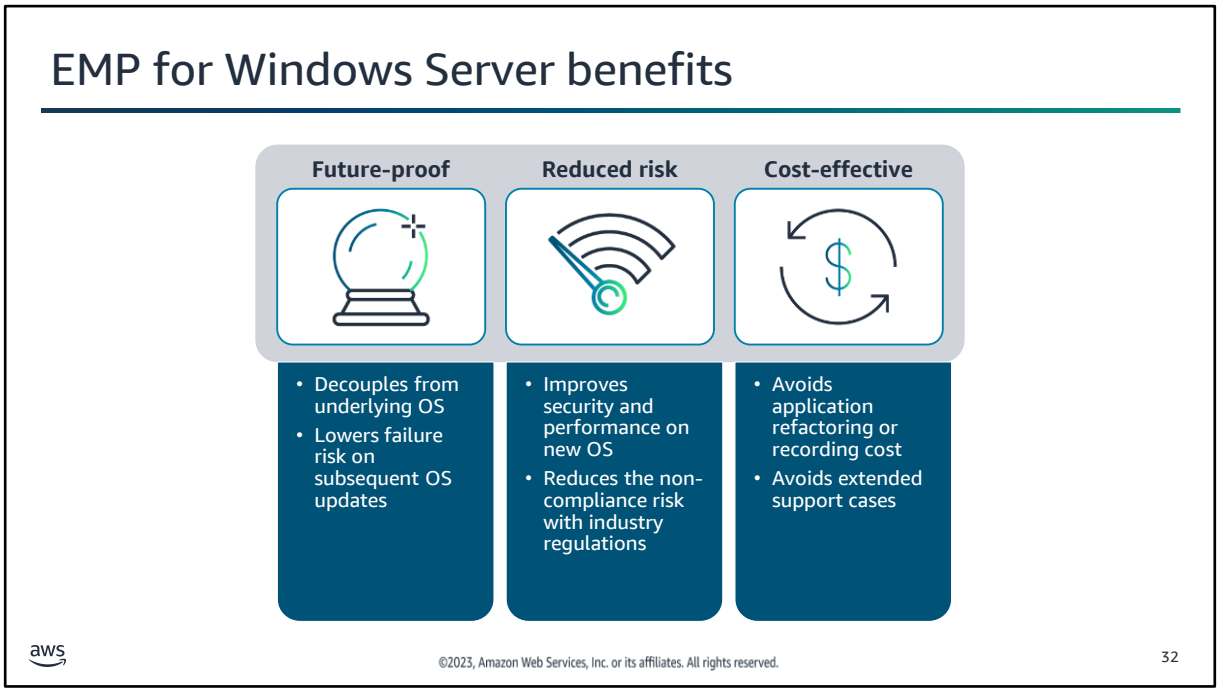
In this section, you learn about the EMP for Windows Server.



EMP for Windows Server is a combined offering of technology, tools, and experts from our select AWS Partners or AWS ProServe. Customers can also self-package legacy applications by downloading the tool directly from the EMP webpage.

EMP technology identifies the dependencies that a legacy application has on an outdated OS. It also creates a package that includes all the resources required by the application.

This package then can be deployed on a newer version of Windows Server without making **any code changes to the application**. The package essentially **decouples** the legacy application from the underlying OS.

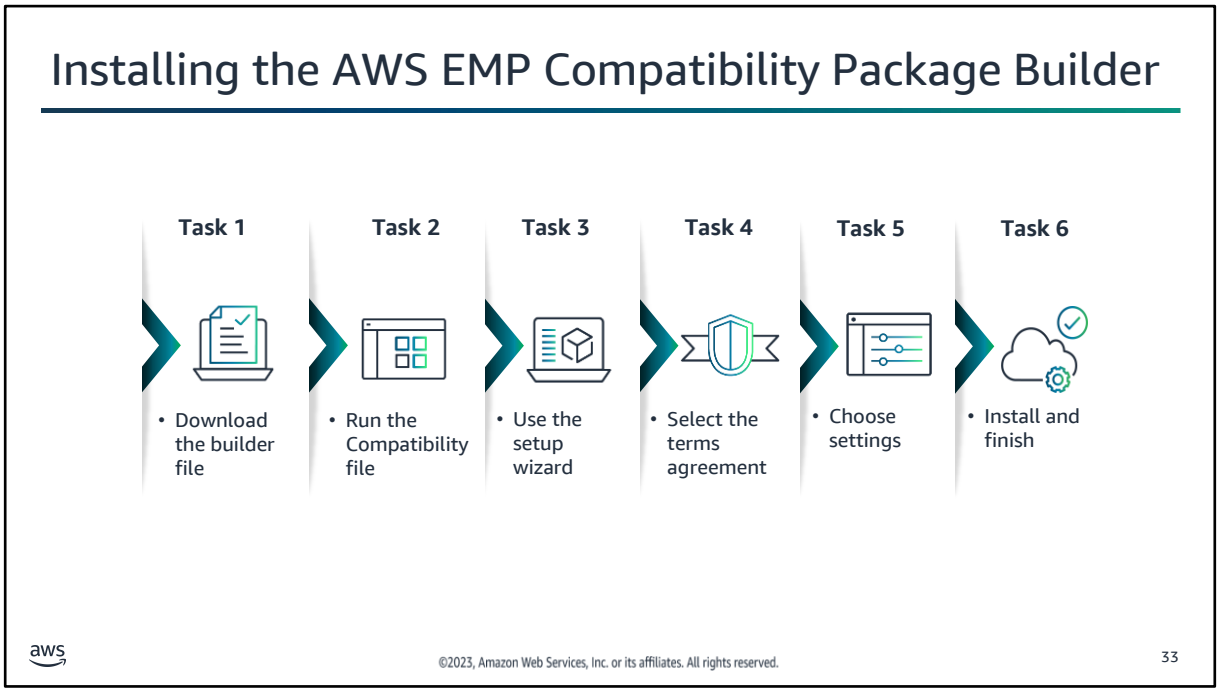


Customers have certain options when faced with end-of-support, including the following:

- **Do nothing:** Continuing to run applications on an unsupported Windows Server OS exposes the customer to high security and compliance risk. It can also result in high operational cost if the customer decides to isolate the applications behind firewalls.
- **Get extended support:** Extended support is costly and even if you remove the cost from the equation, extended support is only a temporary solution. It helps you put off the problem until a later time and does not solve the end-of-support issue. Also, extended support can only be obtained for Windows Server 2008/R2 and not for Windows Server 2003.
- **Refactor and upgrade applications:** This option can be very complex or even impossible for many applications. This requires time and resource requirements.

Compared to these options, EMP offers a superior solution with the following benefits:

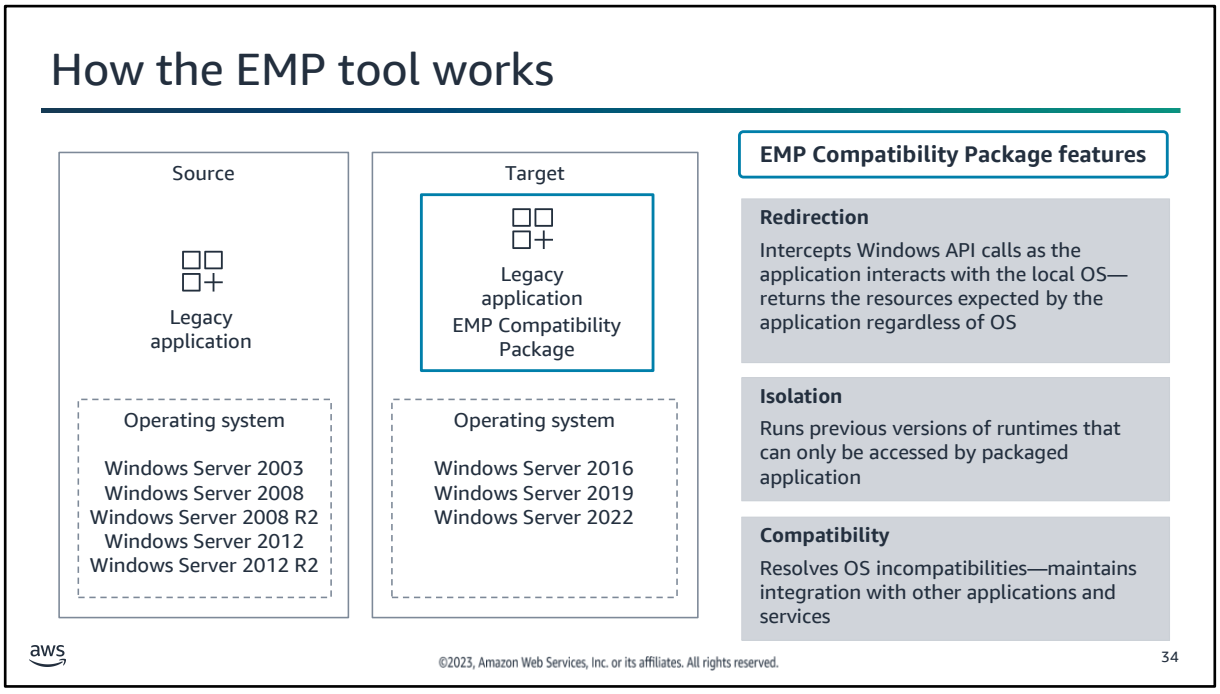
- **Future-proof:** EMP Compatibility Package decouples the legacy application from the underlying OS. This means your risk of upgrade failure is significantly reduced for the next minor or major Windows Server OS update. EMP provides a permanent solution to periodic end-of-support challenges.
- **Reduced risk:** Upgrading the underlying OS significantly reduces the security and compliance risk. The new Windows Server OS offers performance and access to other non-security features like support for Windows Container.
- **Cost-effective:** EMP does not require refactoring or recoding of legacy applications. Upgrading from an end-of-support OS also means no extended support costs. EMP works with a wide range of applications, including those with high complexity where refactoring is either not possible or too resource and cost intensive.



Before starting, perform the steps to install the AWS EMP Compatibility Package Builder. To get the latest steps, see *Install AWS EMP Compatibility Package Builder* at https://docs.aws.amazon.com/pt_br/emp/latest/userguide/emp-install-compatibility-package-builder.html.

Perform the following steps to install the AWS EMP Compatibility Package Builder:

1. If you have not done so already, download AWS EMP Compatibility Package Builder from the EMP for Windows Server product page.
2. After you have downloaded the EMP tools, choose (double-click) the Compatibility Package Builder file to run it.
3. On the Welcome to the Compatibility Package Builder Setup Wizard box, choose Next.
4. In the End-User License Agreement, select the terms agreement, and choose Next.
5. Under EMP Telemetry, select the check box to enable telemetry (optional), and choose Next. Accept the default Destination Folder, or modify it, and choose Next.
6. Choose Install and when the application installation completes, choose Finish.

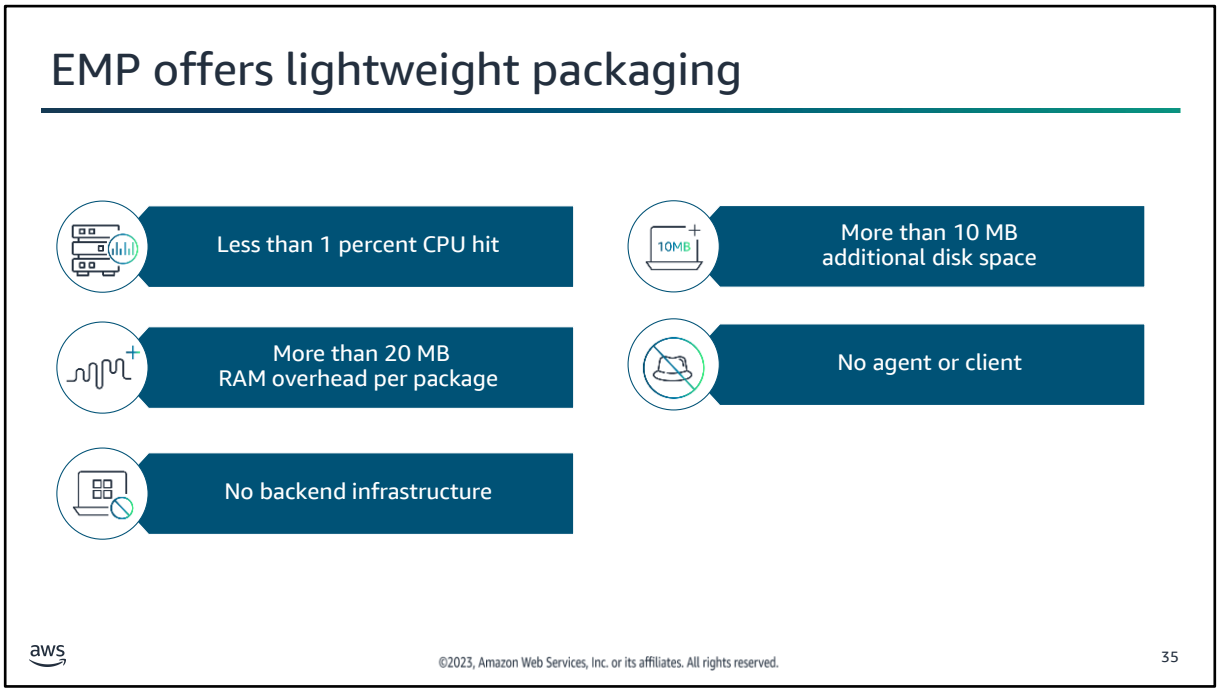


On the left side, we have a legacy application deployed on a previous version of the Windows Server. After going through the EMP process, the target state has the same legacy application without any code changes wrapped in the EMP Compatibility Package. It is then deployed on the newer Windows Server OS.

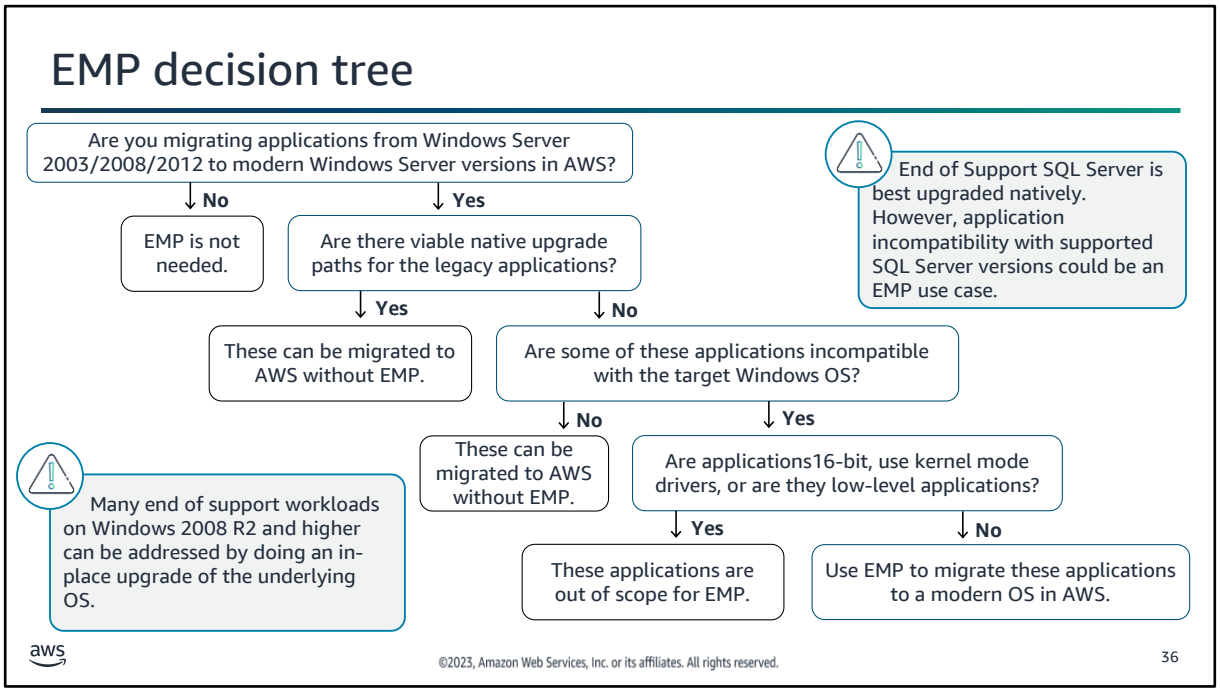
There are three key features with the EMP Compatibility Package:

- **Redirection** – Compatibility Package intercepts any API calls that the legacy application makes to the OS and resolves incompatibilities. For example, if the application depends on a previous version of .NET, like .NET 1.0, this previous runtime is included in the package itself. Any calls that the legacy application makes to the underlying OS for .NET runtime redirect to the .NET runtime included in the package. Similarly, the package can perform redirections for registry keys, hard-coded windows file paths, and networking ports.
- **Isolation** – With this feature, the Compatibility Package runs several versions of a runtime on the same OS. The previous runtime is isolated from the external environment and can only be accessed by the legacy application. This further improves the security posture.
- **Compatibility** – EMP provides application to OS compatibility while maintaining the application behavior and integrations with other application tiers, databases, services, and more. Additionally, EMP solves for several OS incompatibilities like DEPOptOut (which identifies and handles DEP exceptions), or Common Object Model (COM) Virtualization, and reporting different operating systems back to the application.

For more information, see https://docs.aws.amazon.com/emp/latest/userguide/emp-install-compatibility-package-builder.html.



A common question with any packaging technology is if the application performance will be impacted because of the package. With EMP packaging, the answer is the performance will not be affected. EMP packaging is very lightweight and uses minimal resources. Based on our experience, CPU usage, disk space, and RAM used is negligible. Additionally, the package, which is essentially a windows folder structure, is fully self-sufficient. It doesn't require an agent or client to be installed on the target environment and does not make any connection to any backend infrastructure.



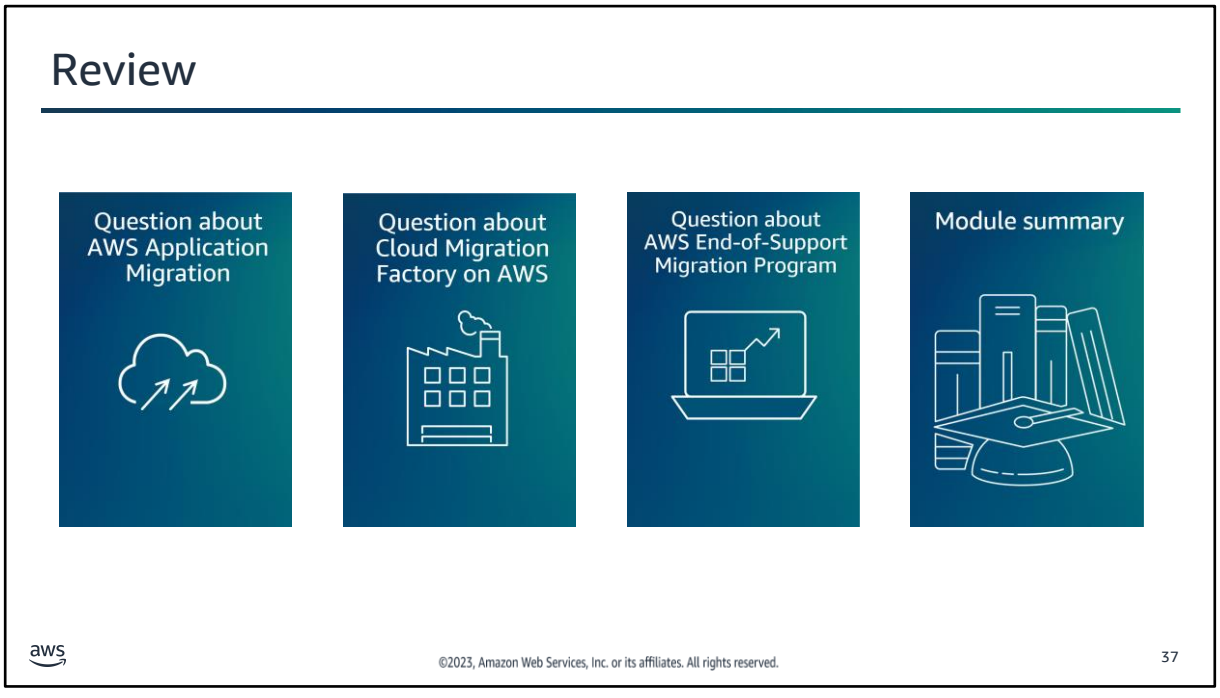
You can use this decision tree determine if the EMP will help when you need to migrate applications to a modern OS in AWS.

Before you start, it is important to note the following:

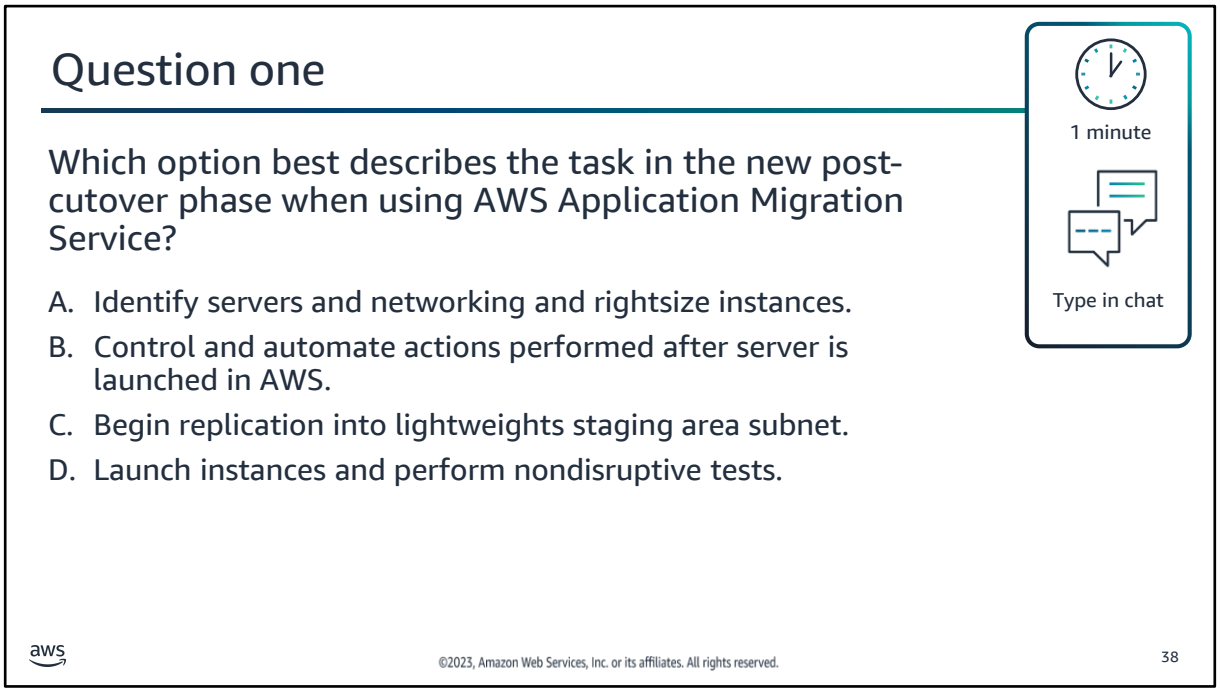
- Many end of support workloads on Windows 2008 R2 and higher can be addressed by doing an in-place upgrade of the underlying OS.
- End of Support SQL Server is best upgraded natively. However, application incompatibility with supported SQL Server versions could be an EMP use case.

Start by asking, “Are you migrating applications from Windows Server 2003/2008/2012 to modern Windows Server versions in AWS?”

- If no, EMP is not needed.
- If yes, ask, “Are there viable native upgrade paths for the legacy applications?”
    - If no, these can be migrated to AWS without EMP.
    - If yes, ask, “Are some of these applications known to be incompatible with the target Windows OS?”
        - If no, these can be migrated to AWS without EMP.
        - If yes, ask, “Are applications 16 bit, use kernel mode drivers, or are low level applications?" For more information, see AWS End-of-Support Migration Program (EMP) for Windows Server limitations at https://docs.aws.amazon.com/emp/latest/userguide/emp-limitations.html.
            - If yes, these applications are out of scope for EMP.
            - If no, use EMP to migrate these application to a modern OS in AWS.

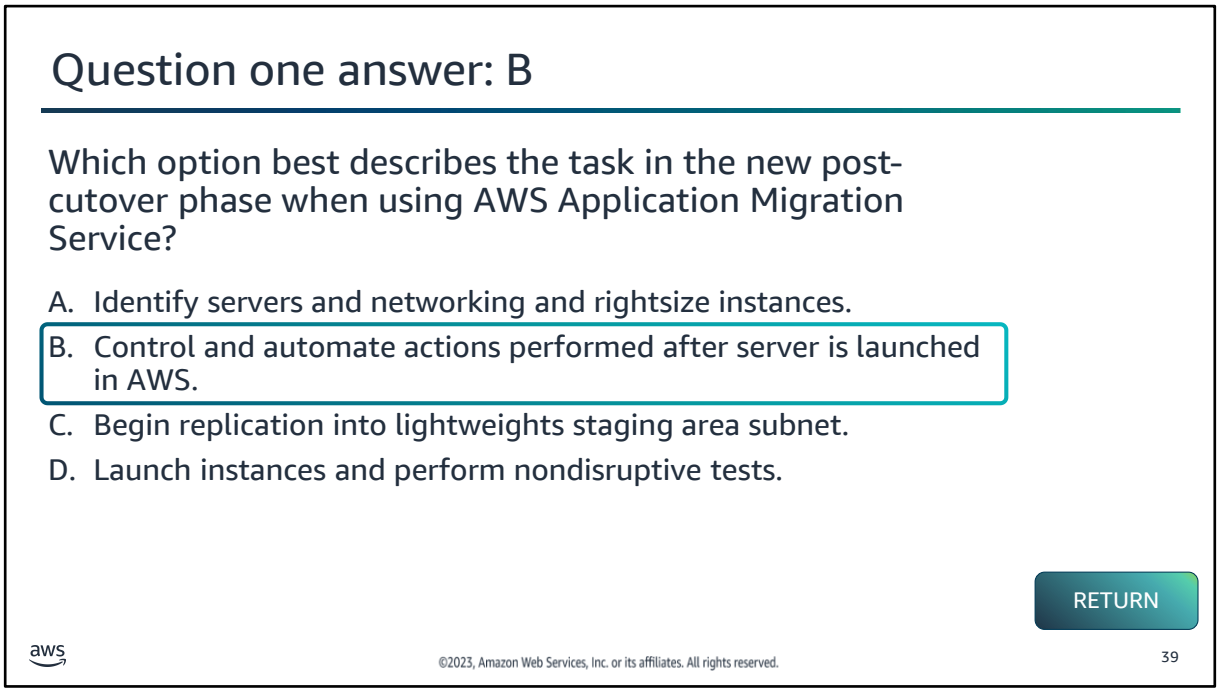


Choose the option that matches the activity you want to perform. The **Module summary** option takes you to the next section of the course.



Which option best describes the task in the new post-cutover phase when using AWS Application Migration Service?

A. Identify servers and networking and rightsize instances.
B. Control and automate actions performed after server is launched in AWS.
C. Begin replication into lightweights staging area subnet.
D. Launch instances and perform nondisruptive tests.



Which option best describes the task in the new post-cutover phase when using AWS Application Migration Service?

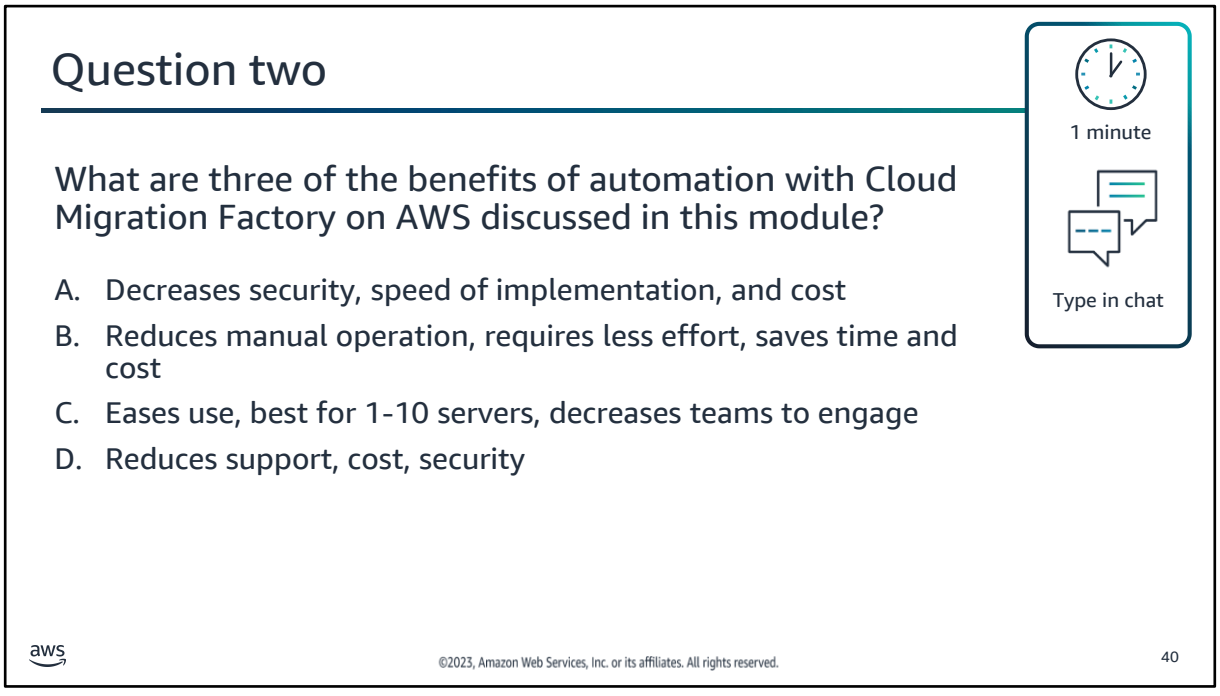
The answer is B – Control and automate actions performed after the server is launched.

A. Identify servers and networking and right-size instances. – This is incorrect, that is in the assessment phase.

B. Control and automate actions performed after server is launched in AWS. – This is correct.

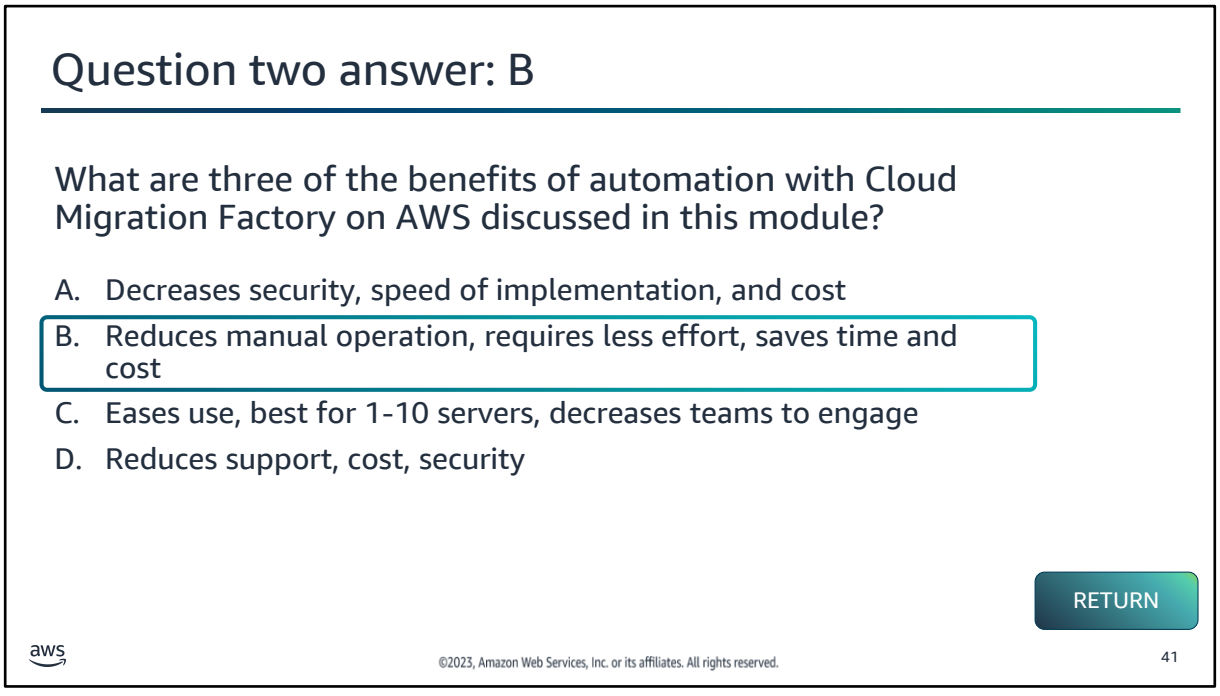
C. Replication begins into lightweights staging area subnet. – This is incorrect, that is in the setup phase.

D. Launch instances and perform nondisruptive tests. – This is incorrect, that is in the setup phase.



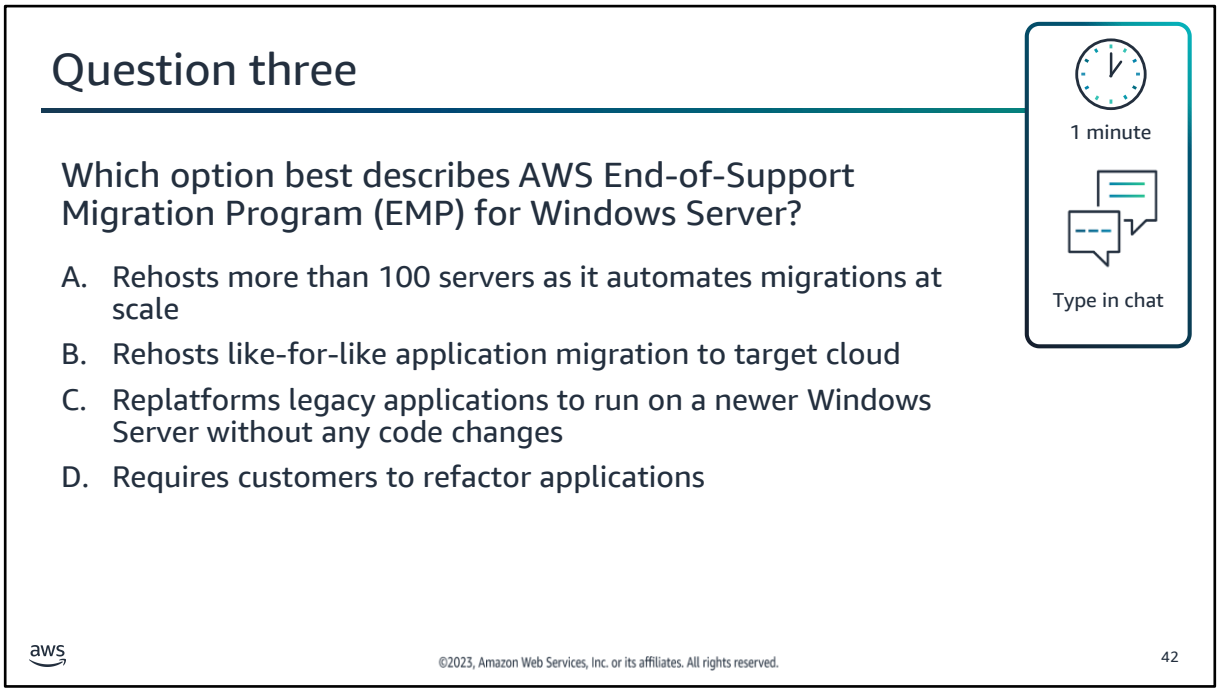
What are three of the benefits of automation with Cloud Migration Factory on AWS discussed in this module?

A. Decreases security, speed of implementation, and cost
B. Reduces manual operation, requires less effort, saves time and cost
C. Ease of use, best for 1-10 servers, decreases teams to engage
D. Reduces support, cost, security



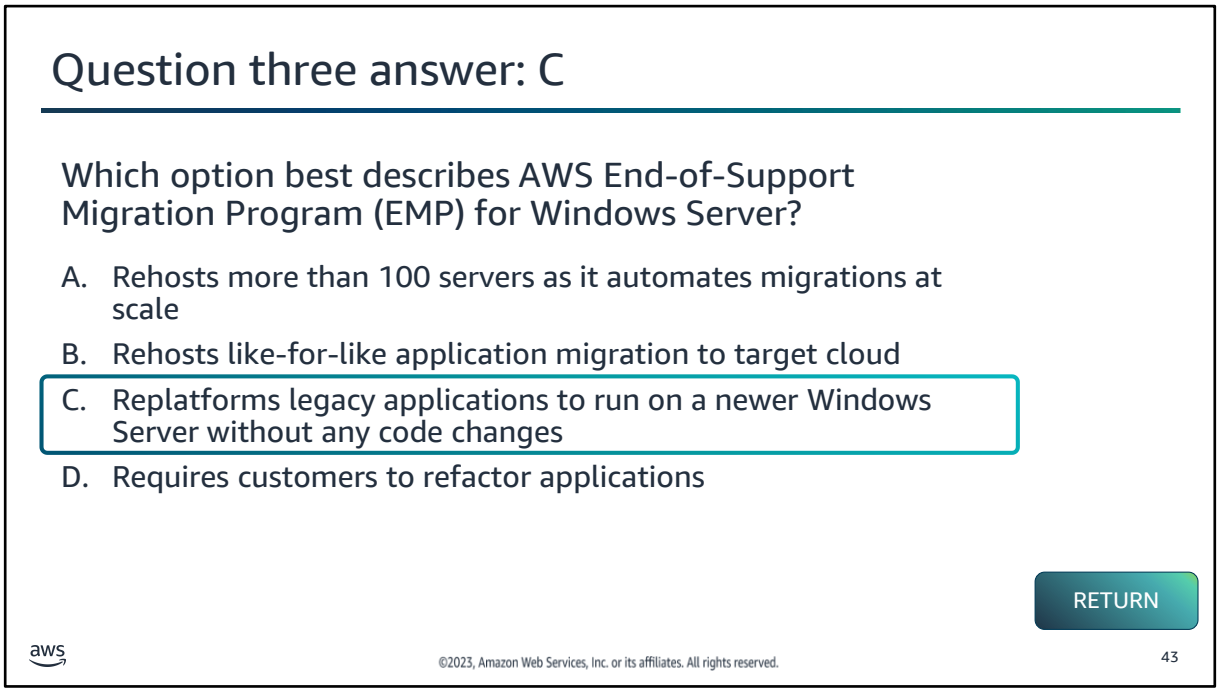
What are three of the benefits of automation with Cloud Migration Factory on AWS discussed in this module? The answer is B.

A. Decreases security, speed of implementation, and cost – This is incorrect. It doesn't decrease security.
B. Reduces manual operation, requires less effort, saves time and cost
C. Ease of use, best for 1-10 servers, decreases teams to engage – This is incorrect because it is best for >100 servers.
D. Reduces support, cost, security – This is incorrect. It doesn't reduce support.



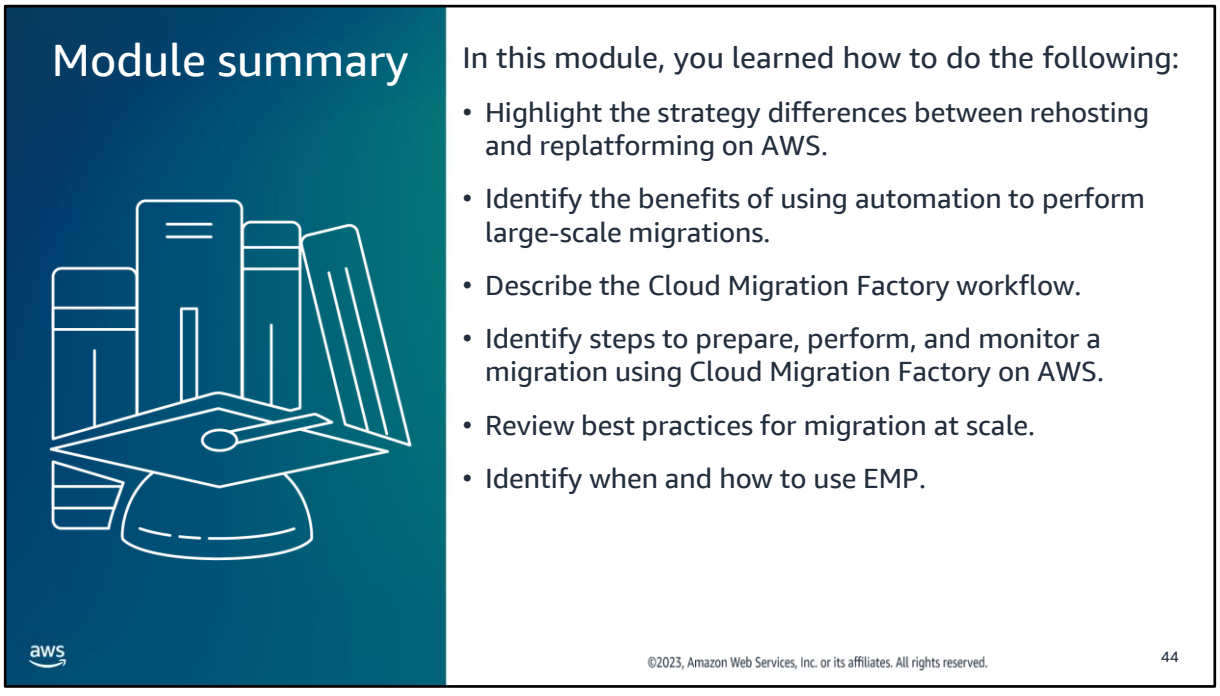
Which option best describes AWS End-of-Support Migration Program (EMP) for Windows Server?

A. Rehosts more than 100 servers as it automates migrations at scale
B. Rehosts like-for-like application migration to target cloud
C. Replatforms legacy applications to run on a newer Windows Server without any code changes
D. Requires customers to refactor applications



Which option best describes AWS End-of-Support Migration Program (EMP) for Windows Server? The answer is C – It is used to replatform legacy applications to run on a newer Windows Server without any code changes.

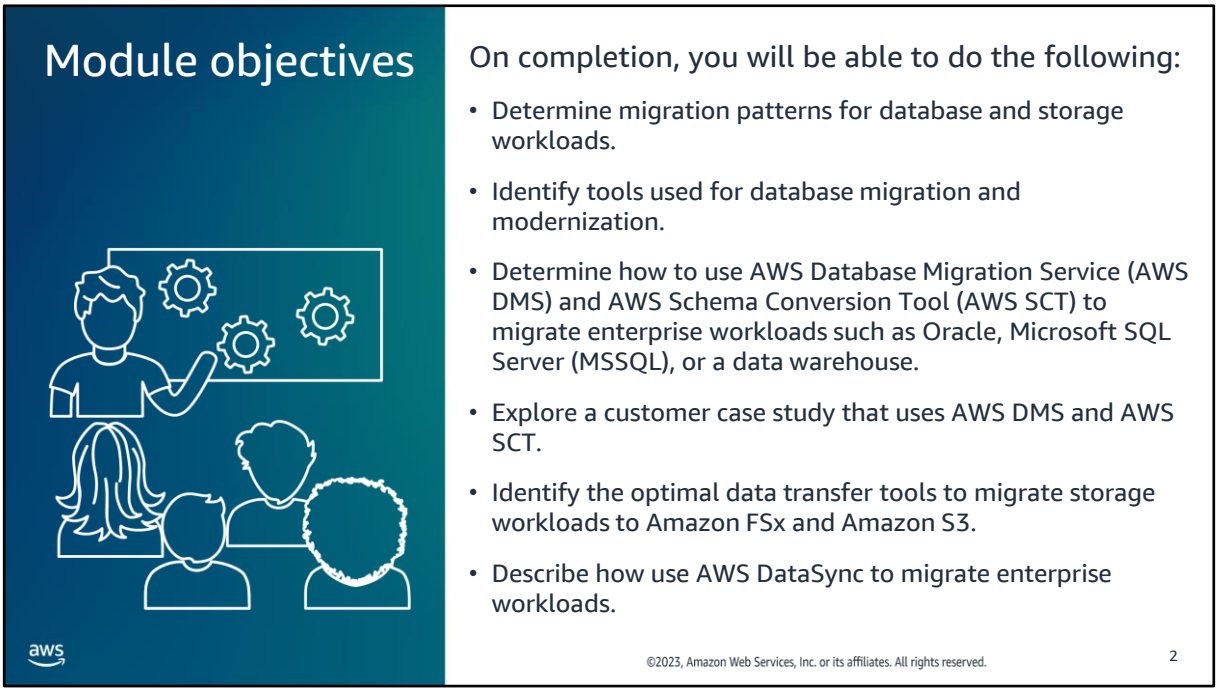
A. Rehosts more than 100 servers as it automates migrations at scale – Incorrect, Cloud Migration Factory is the service that automates and rehosts.
B. Rehosts like-for-like application migration to target cloud – Incorrect, it is for replatforming NOT rehosting.
C. Replatforms legacy applications to run on a newer Windows Server without any code changes – This is the correct answer.
D. Requires customers to refactor applications – Incorrect, it does NOT require refactoring.



In this module, you learned to do the following:

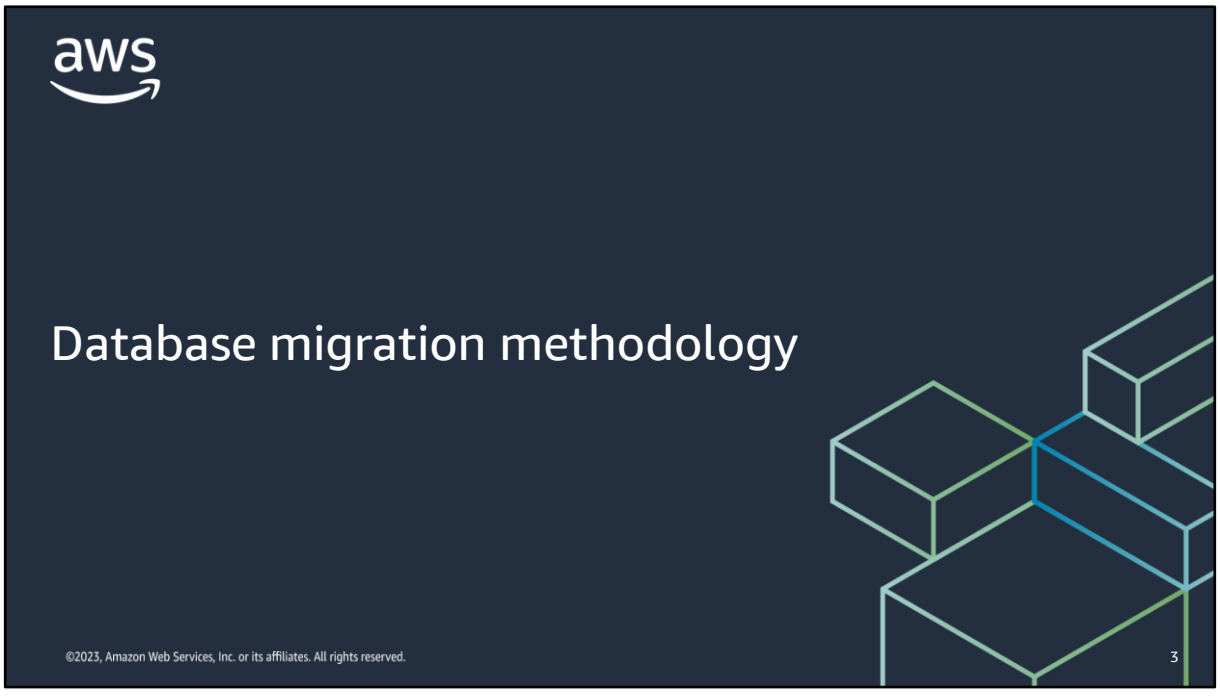
- Highlight the strategy differences between rehosting and replatforming on AWS.
- Identify the benefits of using automation to perform large-scale migrations.
- Describe the Cloud Migration Factory workflow.
- Identify steps to prepare, perform, and monitor a migration using Cloud Migration Factory on AWS.
- Review best practices for migration at scale.
- Identify when and how to use EMP.

Welcome to Module 3: Advanced Database and Data Migrations.
In this section, you will learn about how to migrate structured data, specifically heterogenous databases. Then you will learn about unstructured data, data warehouse migration methodology, and then finish up with network-attached storage (NAS).

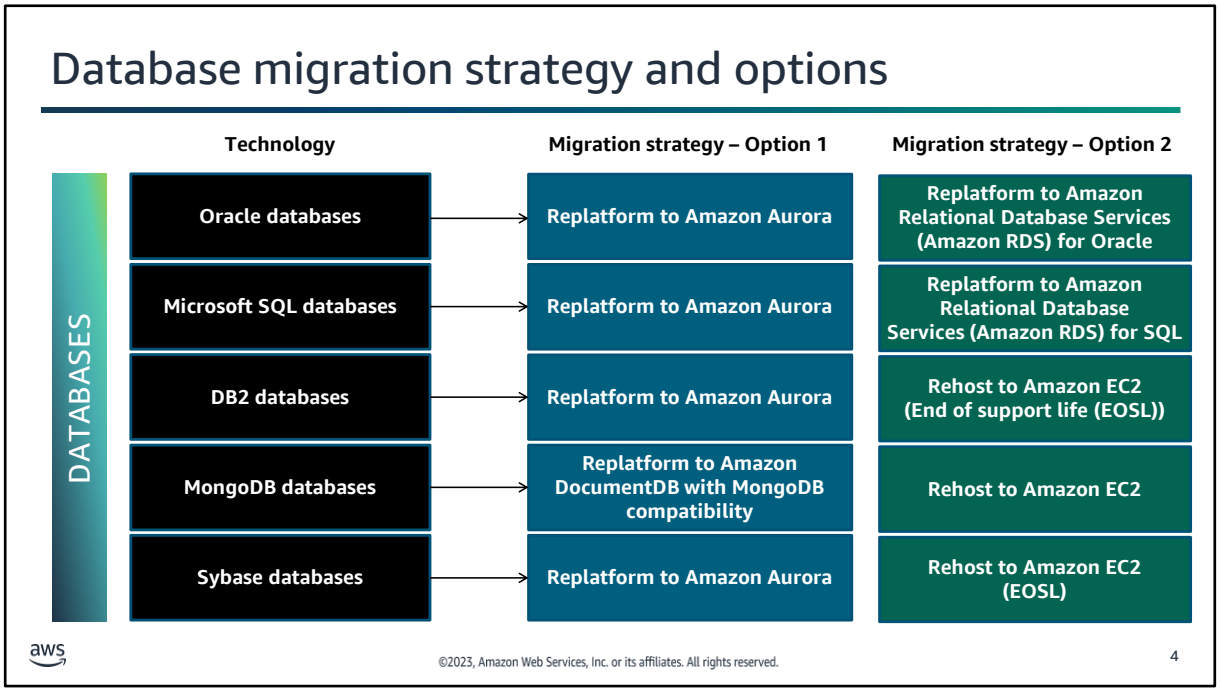


On completion of this module, you will be able to do the following:

- Determine migration patterns for database and storage workloads.
- Identify tools used for database migration and modernization.
- Determine how to use AWS Database Migration Service (AWS DMS) and AWS Schema Conversion Tool (AWS SCT) to migrate enterprise workloads such as Oracle, Microsoft SQL Server (MSSQL), or a data warehouse.
- Explore a customer case study that uses AWS DMS and AWS SCT.
- Identify the optimal data transfer tools to migrate storage workloads to Amazon FSx and Amazon S3.
- Describe how to use AWS DataSync to migrate enterprise workloads.



In this section, you learn about database migration methodologies and strategies.



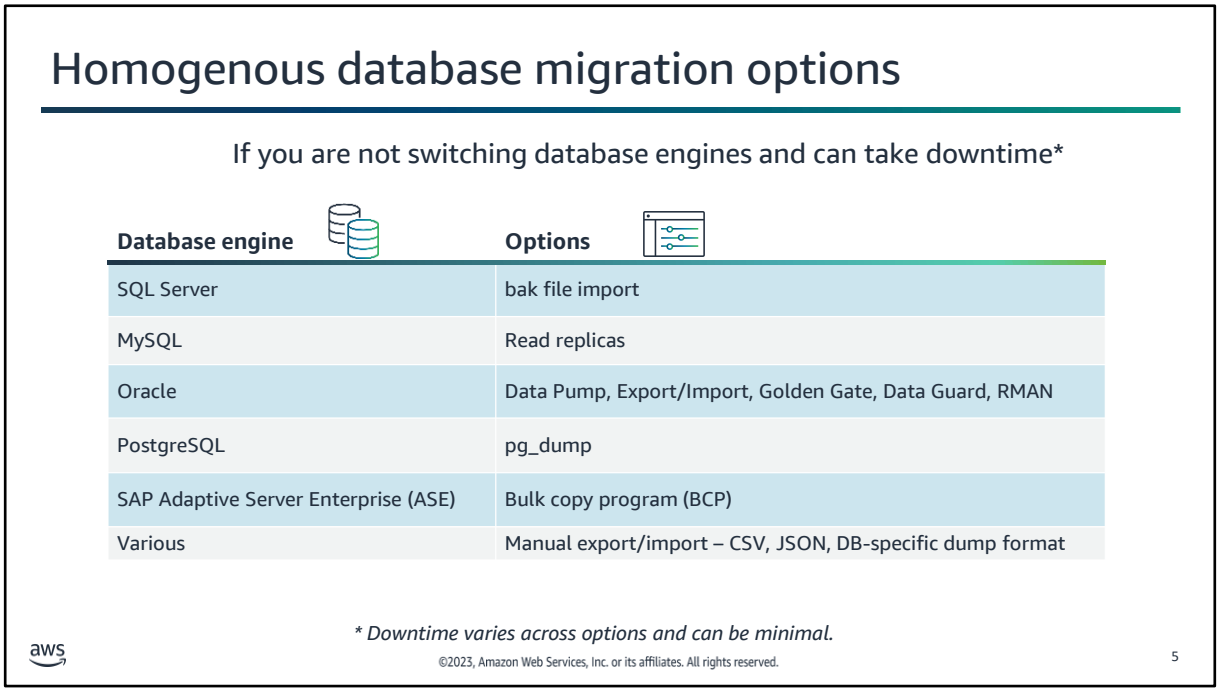
After assessing your customer's application and considering their business goals and constraints like cost and timelines, you can recommend a primary and secondary database migration strategy. The strategy depends on detailed insights from an in-depth analysis, including tooling and workshops with the application owners.

For example, the primary strategy involves migrating to a managed and cloud-optimized service like Amazon Aurora. However, your customer might require OS-level control, making a rehost to Amazon EC2 the preferred migration strategy. In such instances, the focus is on moving the existing application to an EC2 instance. The customer can defer database modernization until they have time for comprehensive application refactoring.

By taking these factors into account and tailoring the migration strategy to the customer's specific requirements, you can align with their objectives and constraints. This approach results in a successful and efficient database migration that meets both immediate and long-term needs.

Technology: Migration strategy Option 1 – Migration strategy Option 2
Oracle databases option 1 – Replatform to Aurora, option 2 – Replatform to Amazon Relational Database Service (Amazon RDS) for Oracle
Microsoft SQL databases: option 1 – Replatform to Aurora, option 2 – Replatform to Amazon Relational Database Service (Amazon RDS) for SQL
DB2 databases: option 1 – Replatform to Aurora, option 2 – Rehost to Amazon EC2 – End of support life (EOSL)
MongoDB databases: option 1 – Replatform to Amazon DocumentDB with MongoDB compatibility,

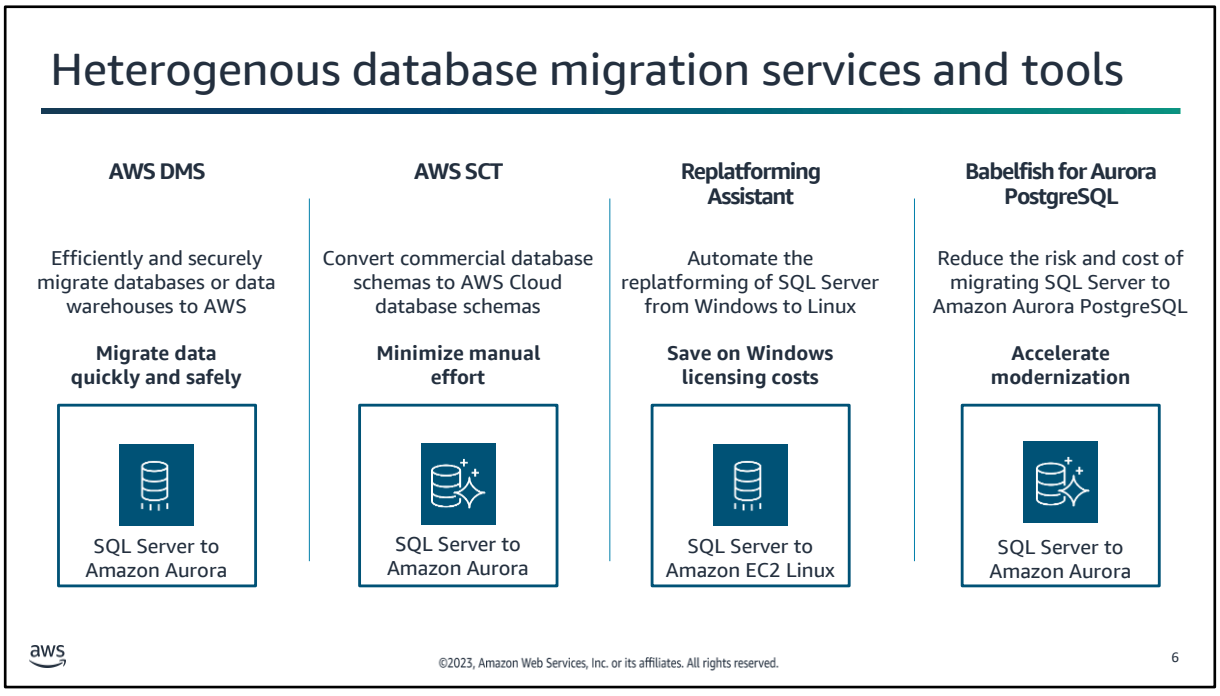
option 2 – Rehost to Amazon EC2
Sysbase databases: option 1 – Replatform to Aurora, option 2 – Rehost to Amazon EC2 (EOSL)



When it comes to migrating a homogeneous database on AWS, there are several options available if you can take downtime. Application Migration Service is also an option. Remember a homogenous database migration is migrating a database to another database of the same type. The choice depends on the specific database engine you are using and your migration requirements. In this table, you can see some common options for migrating homogeneous databases on AWS.

Note that you can also perform a manual export and import. Depending on the database engine, you can manually export data from the source database into a file format, such as CSV, JSON, or database-specific dump format. Then you can import it into the target database. This method requires careful planning and scripting. It might not be suitable for large-scale or complex migrations.

When performing a database migration, you must carefully plan and consider factors like data volume, downtime requirements, and data consistency. We recommend consulting the documentation and resources specific to each database engine for detailed instructions on the migration process. Downtime varies across options and can be minimal.



Regarding heterogenous database migrations, AWS provides a comprehensive suite of tools designed to expedite and streamline your database modernization journey. The following tools offer efficient solutions for migrating your databases and minimizing disruptions:

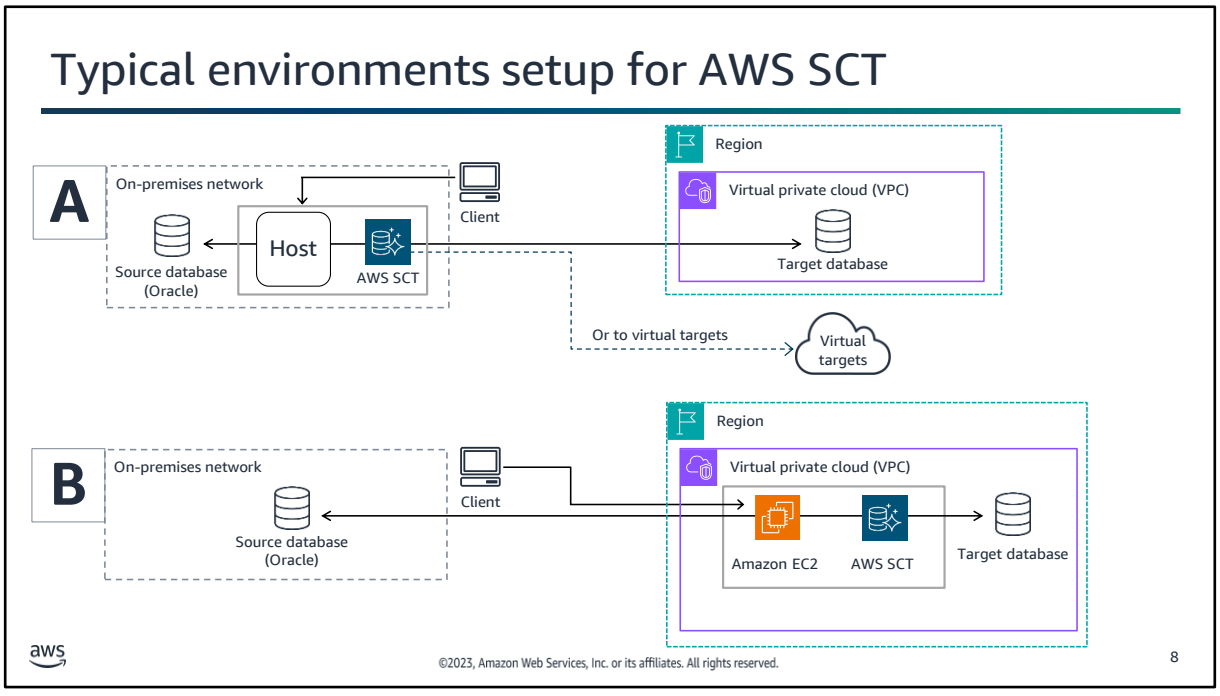
- **AWS DMS:** You can migrate databases to AWS quickly and securely with AWS DMS. The source database remains fully operational during the migration. This functionality minimizes downtime to applications that rely on the database. AWS DMS can migrate your data to and from the most used commercial and open source databases.
- **AWS SCT:** Your heterogeneous database migrations are predictable with AWS SCT. AWS SCT automatically converts the source database schema and most of the database code objects, including views, stored procedures, and functions, to a format compatible with the target database. You can convert from a relational online transactional processing (OLTP) schema or any supported data warehouse online analytic processing (OLAP) schema to Amazon RDS. Examples include Amazon Aurora MySQL compatible edition or Amazon Aurora PostgreSQL compatible edition.
- **Replatforming assistant:** The Windows to Linux replatforming assistant for Microsoft SQL Server databases service is a scripting tool. With this tool, you can move existing Microsoft SQL Server workloads from a Windows OS to a Linux OS. You can use the replatforming assistant with any Windows Server virtual machines (VMs) hosted in the cloud or with on-premises environments running Microsoft SQL Server 2008 and later.

    Optional: The tool checks for common incompatibilities, exports databases from the Windows VM, and imports into an EC2 instance running Microsoft SQL Server 2017 on Ubuntu 16.04. The automated process results in a ready-to-use Linux VM configured with your selected SQL Server databases that can be used for experimenting and testing.

- **Babelfish for Amazon Aurora**: You can use Babelfish to accelerate modernization because it does the following:
    - Runs SQL Server applications on PostgreSQL with little to no code changes
    - Keeps existing queries: With the translation layer, Aurora PostgreSQL understands Microsoft SQL Server's T-SQL
    - Runs T-SQL code side-by-side with new open source functionality and continues developing with familiar tools
    - Lowers risk and completes migrations faster to save you months to years of work

By harnessing these powerful tools, you can simplify and expedite your database modernization journey, ensuring smooth migrations, minimal downtime, and compatibility with AWS services.

In this section, you learn about AWS SCT and AWS DMS schema conversion. Remember, database schema refers to how data is organized in your database.



When setting up the environment for the AWS SCT, you have multiple deployment options available. Although you can install AWS SCT directly on the source host (A), it is generally considered a best practice to follow the principle of least privilege and create a dedicated host or hosts for AWS SCT (B). This approach offers benefits such as improved security, scalability, and flexibility. In situations where an on-premises spare host is not available, hosting AWS SCT in the AWS Cloud on Amazon EC2 is a viable alternative.

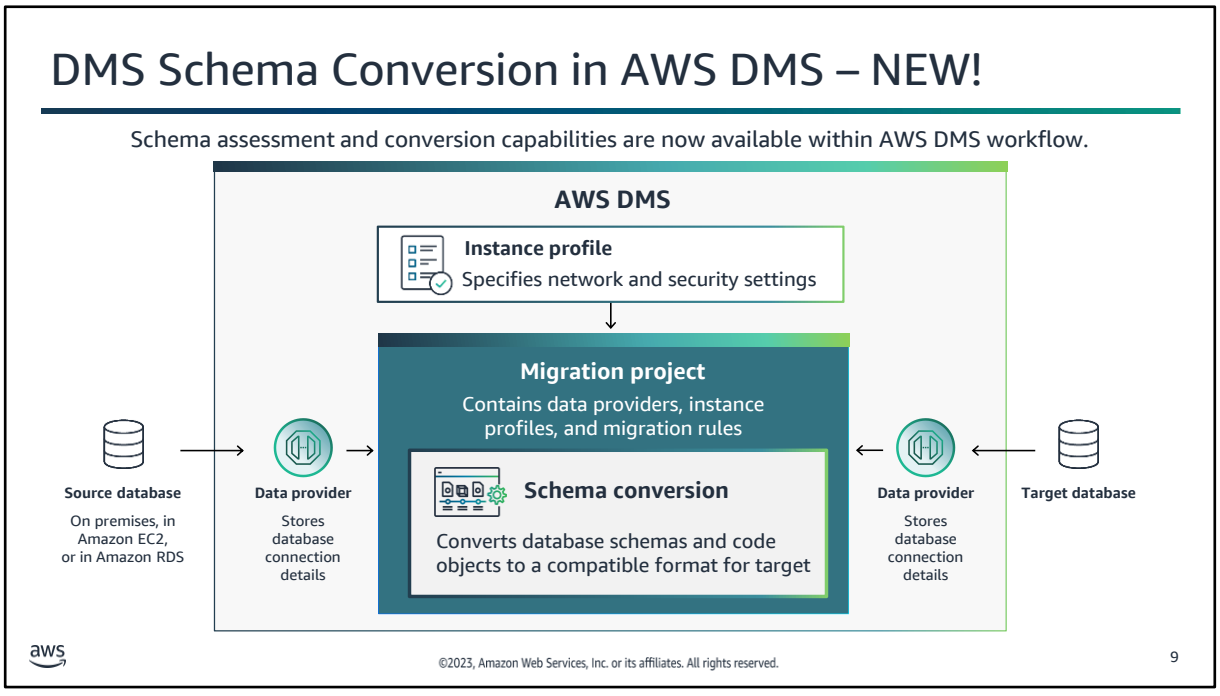
By dedicating a separate host for AWS SCT (B), you can ensure that the tool operates independently from the source host (A). This isolation minimizes the impact on the source system and reduces the risk of unintended consequences during the schema conversion process. When you separate AWS SCT from the source host, you simplify the management and maintenance tasks. You can focus on the conversion process without interfering with the source database's operations.

Setting up AWS SCT on an EC2 instance in the AWS Cloud is particularly advantageous when you have limited or unavailable on-premises resources. AWS offers a range of EC2 instance types and sizes to suit your specific requirements. Additionally, using the AWS Cloud provides scalability and elasticity. You can scale up or down the AWS SCT environment based on demand. This flexibility ensures that the AWS SCT deployment can accommodate varying workloads and handle large-scale database migrations efficiently.

When deploying AWS SCT on a dedicated host or an EC2 instance, consider security measures, such as configuring security groups and access controls, to restrict access to AWS SCT and the source or target databases. By following best practices, you can maintain a secure environment while using the benefits

of dedicated AWS SCT deployment and the scalability of the AWS Cloud.

Follow the principle of least privilege and create a dedicated host or hosts for the AWS SCT function (B). Option B can also be an on-premises configuration. Some developers install AWS SCT on their laptops, although this is not best practice for security and scale.

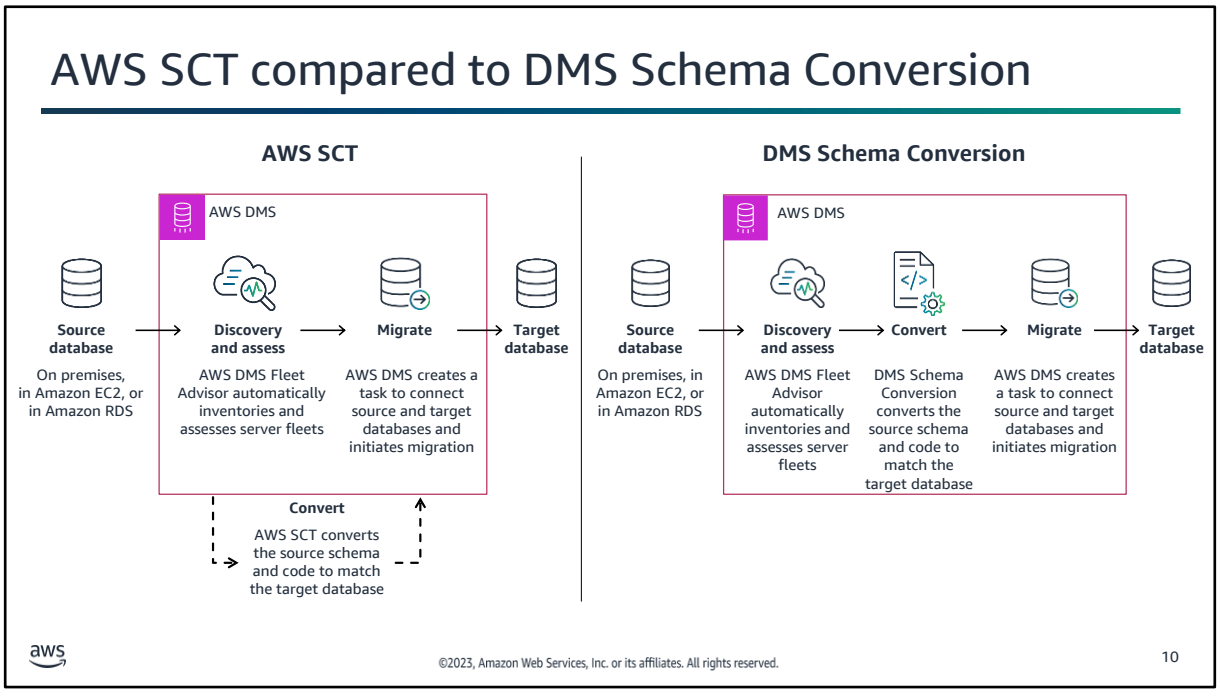


DMS Schema Conversion in AWS DMS simplifies and enhances database migrations between different databases. It helps you assess the complexity of your migration and convert database schemas and code objects. The converted code can be applied to your target database.

DMS Schema Conversion automatically converts your source database schemas and most of the database code objects to a compatible format for the target database. This includes tables, views, stored procedures, functions, data types, synonyms, and more. It also identifies objects that cannot be automatically converted so you can manually convert those objects to complete the migration.

With DMS Schema Conversion, you can conduct assessments and convert schemas directly inside AWS DMS. There is no need to download AWS SCT or piece together tools. You also won't spend significant time and cost assessing and converting the schemas.

DMS Schema Conversion uses database metadata to assess the migration complexity of the source database. It converts the database schemas and code objects without accessing or transferring the customer's data. By securely storing the credentials of their users in AWS Secrets Manager, customers can control access to their source and target databases. Additionally, DMS Schema Conversion encrypts connections to database endpoints by managing the encryption keys.

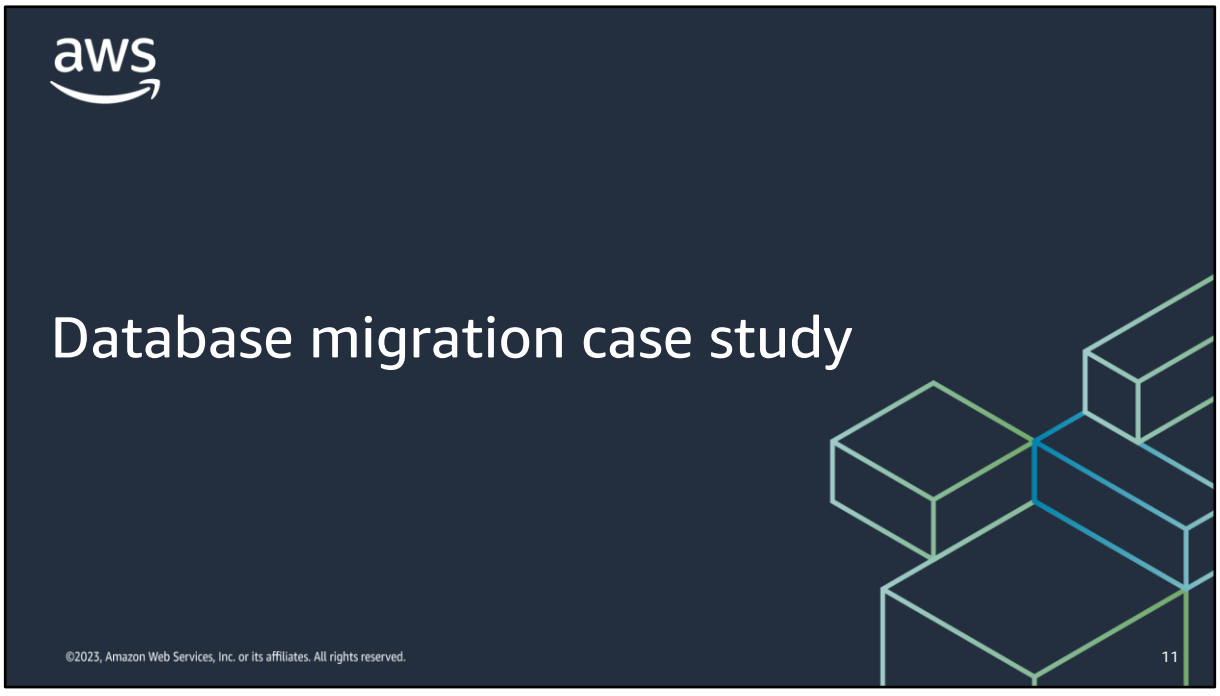


In this side-by-side comparison, the left side of the table represents AWS SCT, which lives outside AWS DMS. With AWS SCT, you move back and forth between AWS DMS and outside tools to complete a migration.

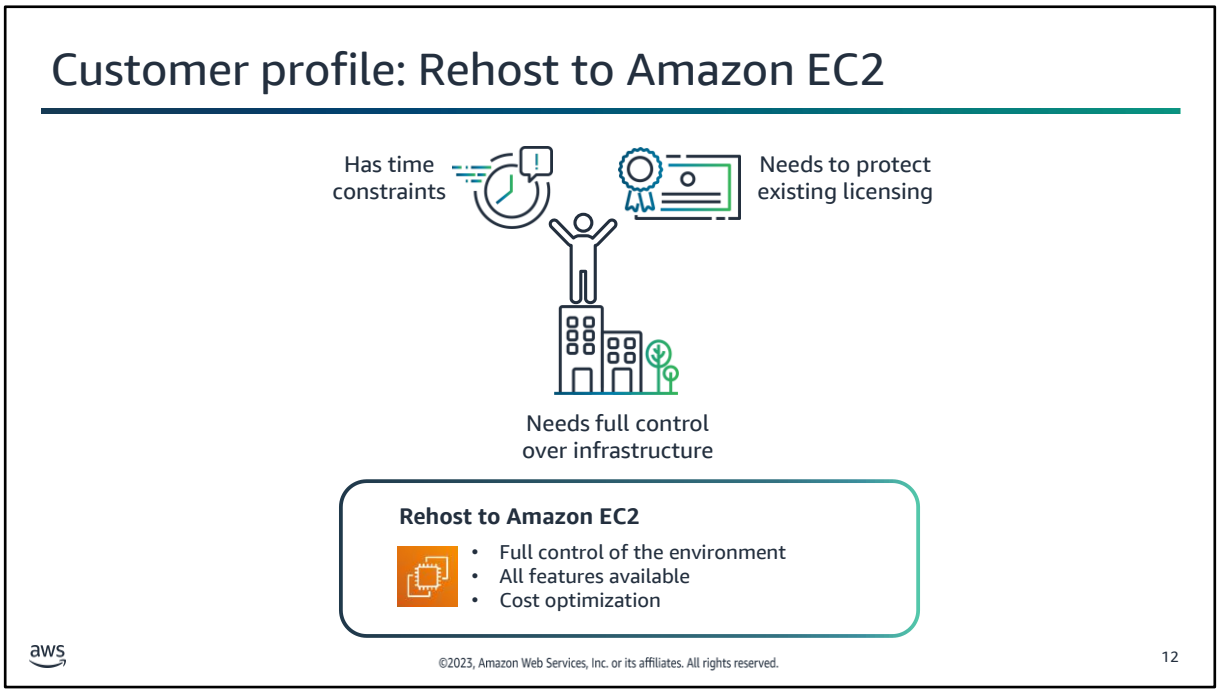
The right side of the table represents DMS Schema Conversion. With DMS Schema Conversion, AWS DMS has AWS SCT as a fully managed capability. Now, AWS DMS offers an end-to-end database migration solution under one centrally managed service. With AWS DMS, you plan, assess, convert, and migrate to the cloud.

Today, DMS Schema Conversion is compatible with the most popular databases, such as Oracle, Microsoft SQL Server, MySQL, and PostgreSQL. AWS plans to quickly expand to support additional engines.

For more information, see *AWS Schema Conversion Tool* at https://aws.amazon.com/dms/schema-conversion-tool and https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_Welcome.html.



In this section, you learn about database migration strategies by customer profile and explore a database migration case study.



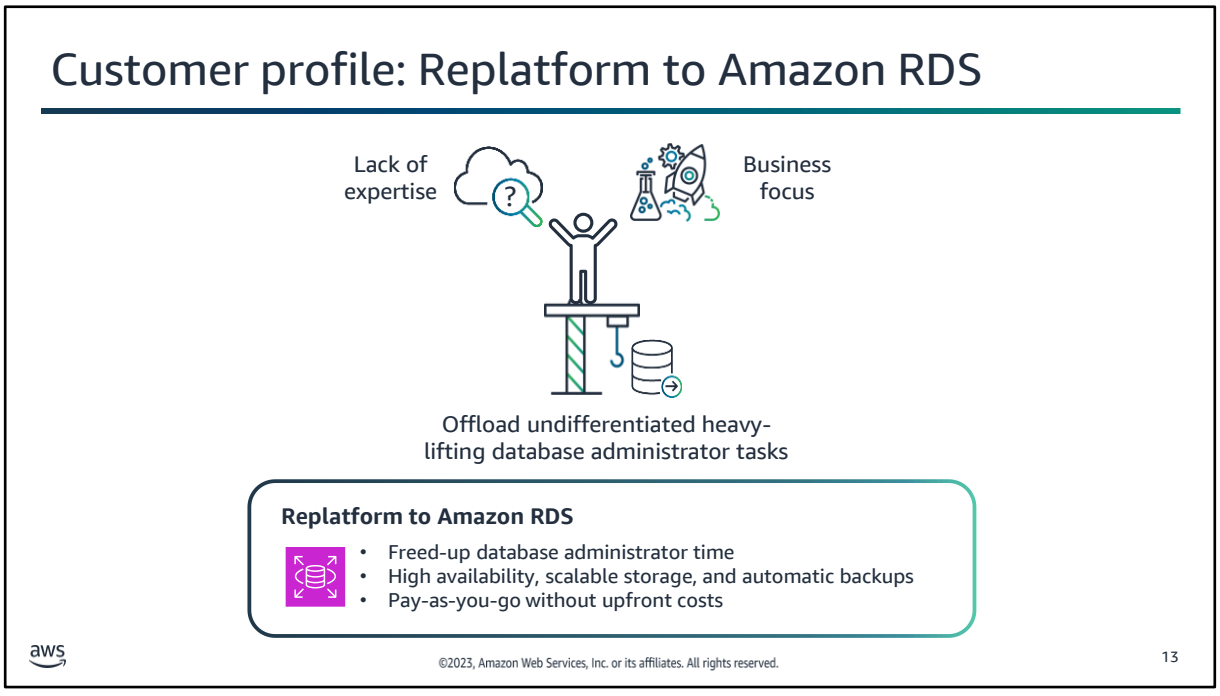
For targeted customers, the combination of rehosting with Amazon EC2 addresses these following needs:

- Rapid migration: It quickly moves applications to AWS without extensive modifications.
- License protection: It maintains existing commercial software licenses.
- Full infrastructure control: It customizes the environment to meet specific requirements.

Rehosting to Amazon EC2 offers the following benefits:

- Fast and streamlined migration to AWS
- Full control of the environment
- Access to all AWS features and services
- Seamless integration with existing infrastructure
- Opportunities for cost optimization

By choosing Amazon EC2 for rehosting, you can quickly migrate your applications, retain control over the environment, use the full range of AWS services, seamlessly integrate with existing resources, and optimize costs.

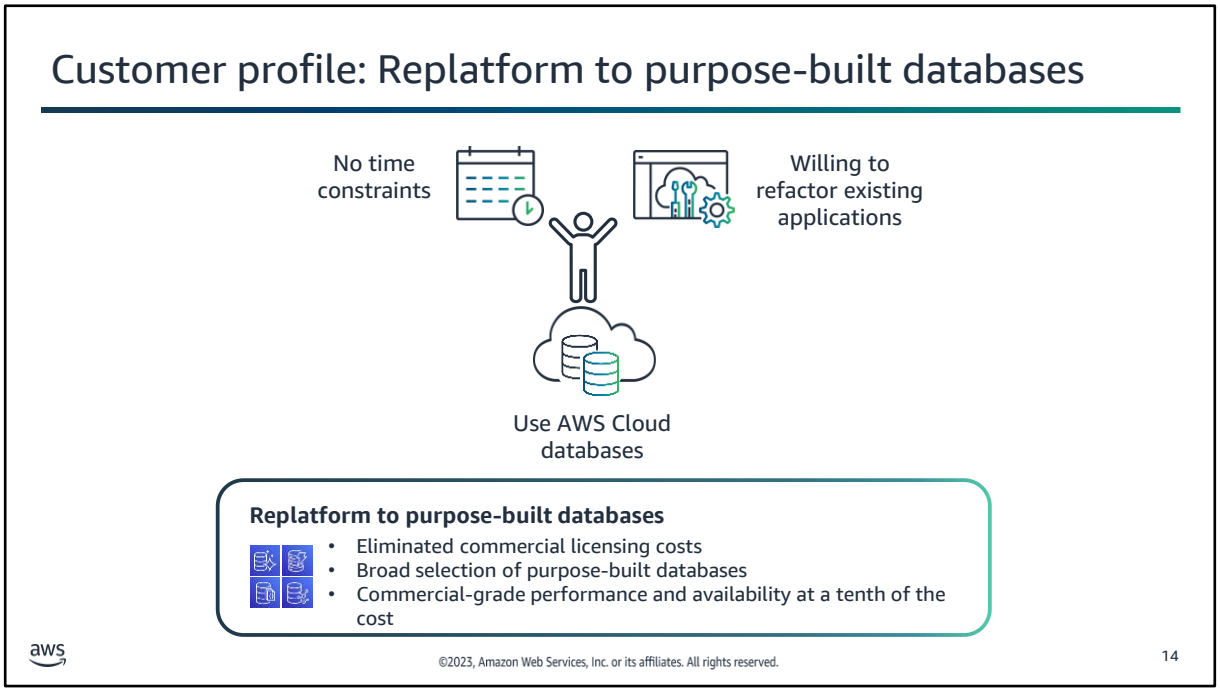


To enhance replatforming to Amazon RDS and address the lack of expertise, this customer can do the following:

- Use managed database services like Amazon RDS to offload database administrators' tasks.
- Engage AWS Professional Services or an AWS Partner for assistance.
- Use AWS DMS for smooth data migration.
- Use automated backups and maintenance offered by Amazon RDS.
- Monitor and optimize performance using AWS tools.
- Invest in training and upskilling for in-house expertise.

Remember, you can offload the undifferentiated heavy-lifting of database administrator tasks and ensure a smoother migration to Amazon RDS while maintaining a business focus. With a managed service like Amazon RDS, expert assistance, and a focus on your core business activities, you can experience the following benefits:

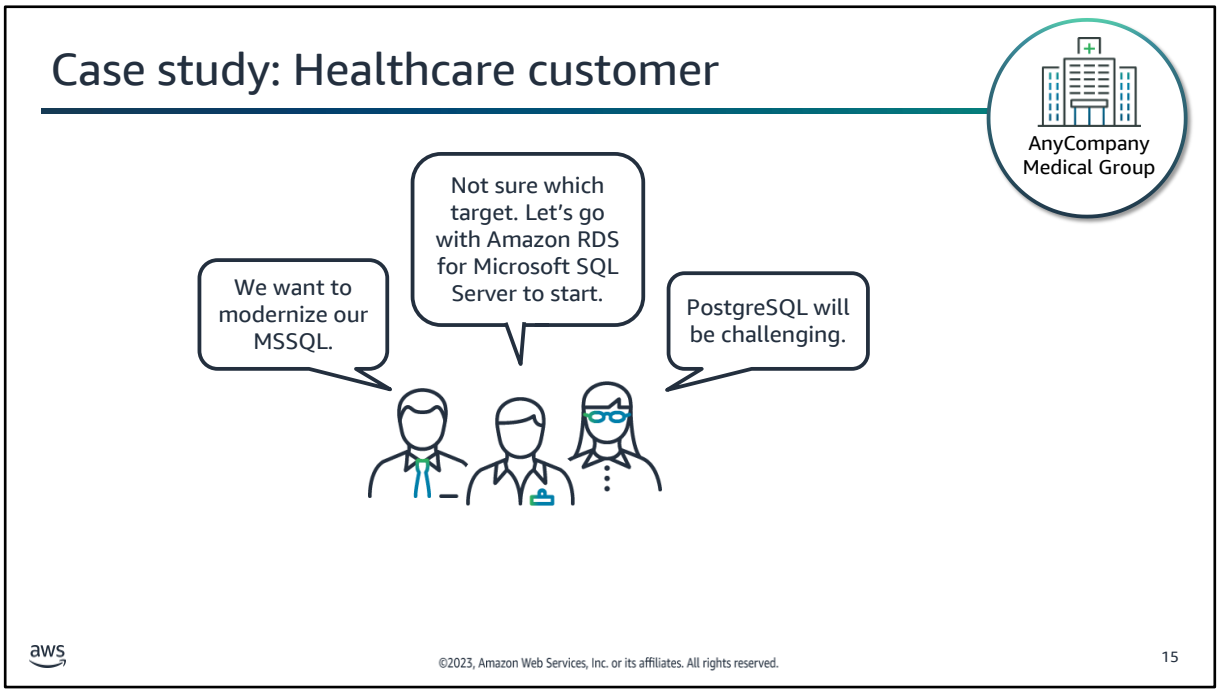
- Freed-up database administrators' time with managed service
- High availability, scalable storage, and auto backups
- Pay-as-you-go without upfront costs



The customer's characteristics, including no time constraints, willingness to refactor existing apps, and desire to use AWS Cloud databases, creates an ideal scenario for refactoring applications. This customer values comprehensive optimization, commitment to improvement, and a forward-thinking approach to using advanced technologies for enhanced performance and scalability.

The benefits of replatforming to purpose-build databases include the following:

- Eliminated commercial licensing costs
- Access to a wide range of purpose-built databases
- Commercial-grade performance
- Availability at a significantly reduced cost



Next, let's look at a migration strategy with AWS SCT. A healthcare customer, AnyCompany Medical Group, seeks to modernize their Microsoft SQL Server database for a strategic business application. Although the customer wanted to modernize, they were not certain of the target database service to use. They decided to take action initially and begin a proof-of-concept with Amazon RDS for Microsoft SQL Server.

The customer eventually engaged with an AWS Partner to assess moving to an open source database technology and improve the TCO of their migration effort. In the requirements gathering phase, the partner performed a detailed discovery of the on-premise database features.

AnyCompany Medical was also creating a manual data pipeline by exporting CSV files to Amazon S3 and then importing to Amazon Redshift. The AWS Partner learned the customer would benefit from an Amazon Redshift federated query to reduce data movement over the network and improve performance. This further justified their decision to move to an open source database engine. The customer was confident they could eventually modernize their application. But thought PostgreSQL would be more challenging that MySQL. PostgreSQL was the preferred open source option for more of the enterprise grade features they required.

Case study example: MSSQL database refactor

Steps to refactor

1. Understand the following application and business requirements:
   - Database use case
   - Detailed discovery of the database instance
   - Current and long-term plan for the application and database
2. Run the AWS SCT.
3. Analyze the AWS SCT result.
4. Determine the target database engine based on analyzing the AWS SCT results.
5. Convert Schema and migrate the data or perform a proof of concept (POC).
6. Perform the application testing and tune the database parameters.

aws

16

MSSQL database refactoring is a process that involves enhancing and optimizing an existing database instance to meet the application and business requirements. These steps outline the key components of this process:

1. Begin by thoroughly understanding the needs and objectives of the application and business. This includes identifying the specific functionalities and performance requirements that the database should fulfill.
   - Define the purpose and scope of the database, including its primary use case. Determine how the application will use the database. Also determine the specific operations the application needs to provide support.
   - Conduct a comprehensive analysis of the existing database instance. This includes examining the schema, data structures, indexes, stored procedures, and any other relevant components. Identify areas that require improvement or optimization.
   - Develop a clear understanding of the current state of the application and the future roadmap. Consider how the application and database evolve over time and align the refactoring efforts accordingly.
2. Use AWS SCT to assess the compatibility and feasibility of migrating the existing database to a different database engine. This tool identifies potential challenges and provides recommendations for the migration process so you can make a data-driven decision.
3. Evaluate the output generated by the AWS SCT, including the assessment of database objects, conversion difficulties, and potential issues. Use this analysis to make informed decisions regarding the refactoring approach and the target database engine.
4. Based on the analysis and requirements, select the most suitable database engine for the migration or refactoring process. Consider factors such as performance, scalability, cost, and

compatibility with the application.

5. Convert the existing database schema and migrate the data to the target database engine. This involves transforming the schema structures, modifying data types, and ensuring data integrity during the migration process. Alternatively, a POC can be performed to validate the effectiveness of the new database engine before proceeding with a full migration.
6. Conduct thorough testing of the application to ensure its compatibility and functionality with the refactored database. Additionally, fine-tune the database parameters, such as indexing, query optimization, and caching, to achieve optimal performance and responsiveness.

By following these steps, AnyCompany Medical Group can effectively enhance and optimize their MSSQL databases to meet the evolving needs of their applications and businesses.

Modernization case study: Level of effort

Report includes estimate of effort required to convert schema items that cannot be converted automatically to the target database engine.

Simple
< 2 hours

Medium
2-6 hours

Complex
> 6 hours

MSSQL to target database engine summary report (number of manual actions)

Complexity	Amazon Aurora MySQL	Amazon Aurora PostgreSQL	Amazon RDS for MySQL	Amazon RDS for PostgreSQL
Simple	9,191	9,578	9,170	9,544
Medium	35	0	13	0
Complex	1,030	1,186	1,020	1,104
Total	10,256	10,764	10,203	10,648

aws

17

One of the first steps for migrating or modernizing a SQL server is to have the customer run AWS SCT. This tool provides data you can use to determine the best database engine choice and effort required to modernize your customer's SQL databases.

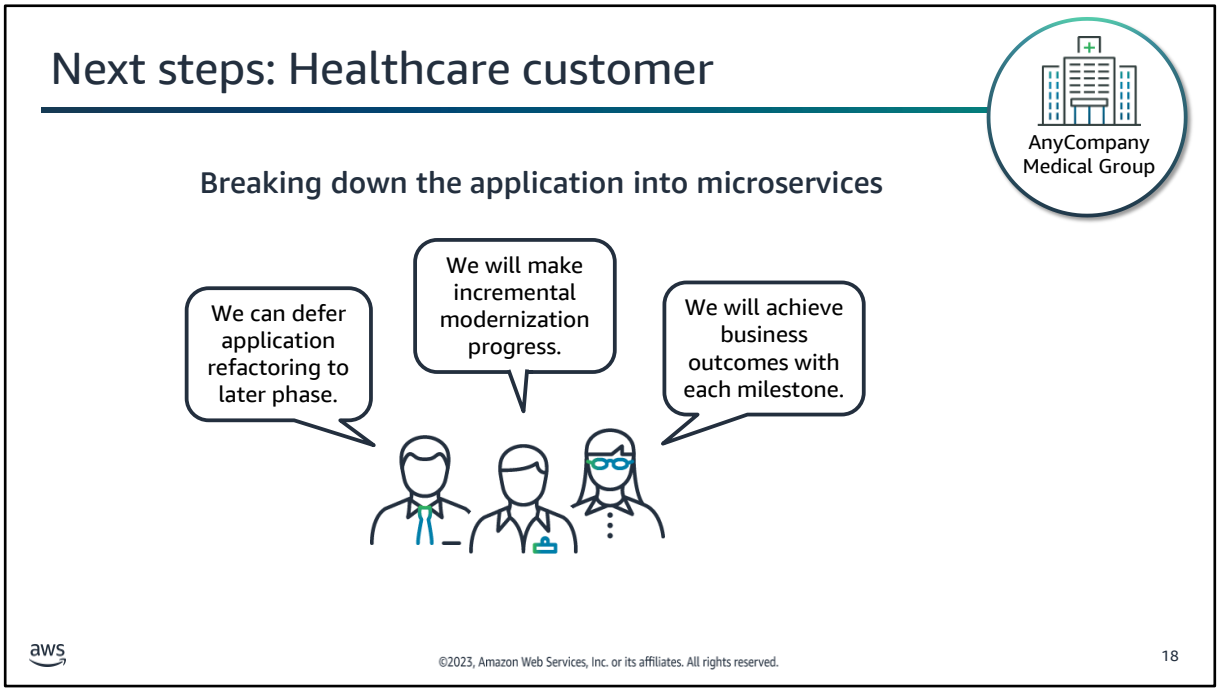
For schema items that cannot be converted automatically to the target database engine, this summary report includes an estimate of the effort required to create schema items in your target DB instance that are equivalent to those in your source. The report categorizes the estimated time to convert these schema items as follows:
Simple – Actions that can be completed in less than 2 hours
Medium – Actions that are more complex and can be completed in 2–6 hours
Complex – Actions that are very complex and take more than 6 hours to complete

You can use the AWS SCT to convert your customer's existing database schema from one database engine to another. You can convert relational OLTP schema or data warehouse schema. Your converted schema is suitable for the following:

- Amazon RDS for: MySQL, MariaDB, Oracle, SQL Server, PostgreSQL
- Amazon Aurora DB cluster
- Amazon Redshift cluster

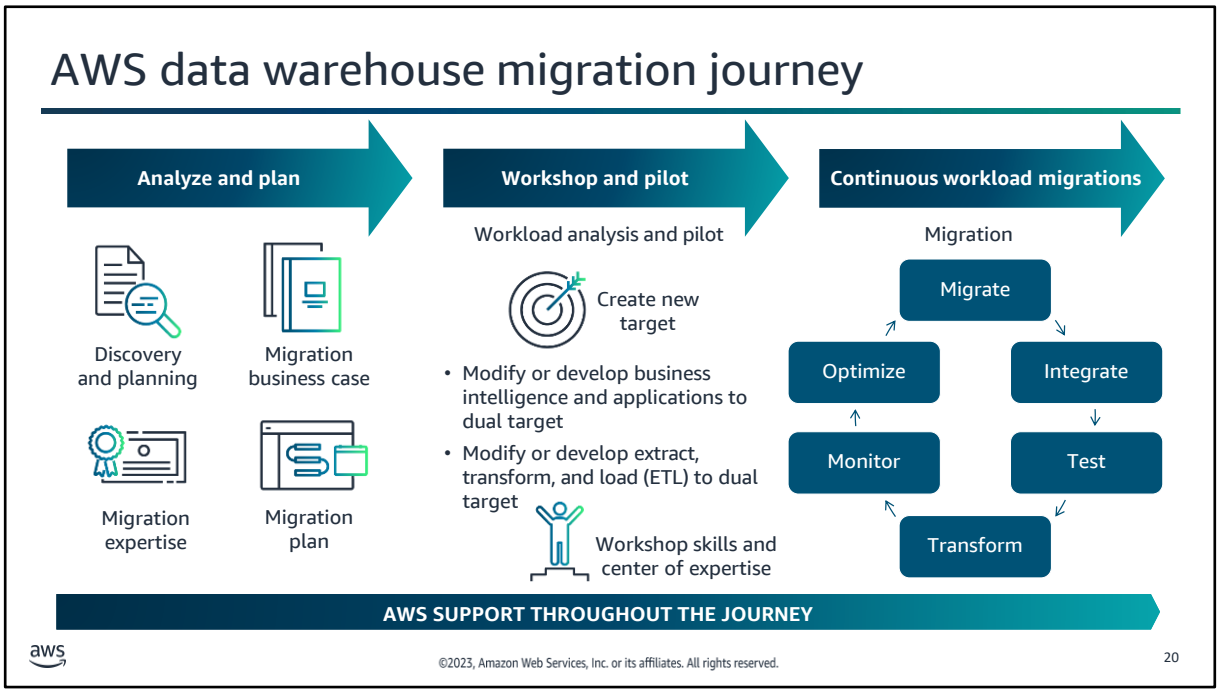
Though Amazon Aurora PostgreSQL has slightly more actions, it was the preferred database for long-term enterprise grade feature sets. Also keep in mind that actions can be refined after you determine duplicates or actions that aren't necessary as part of functional requirements. For more information to compare the difference, see *What's the Difference Between MySQL and PostgreSQL?* at

https://aws.amazon.com/compare/the-difference-between-mysql-vs-postgresql.
To learn more about Microsoft workloads on AWS , see https://aws.amazon.com/blogs/modernizing-with-aws/how-cloudsoft-helped-asp-modernize-ms-sql-server-workloads-to-amazon-aurora-to-save-cost-and-reduce-operational-overhead.



In their next steps, AnyCompany Medical Group will break down their application into microservices. They decide to make incremental modernization progress with each milestone achieving a valuable business outcome. They will defer deeper application refactoring to a later phase.

In this section, you will explore migrating data warehouses. Because you are dealing with terabytes of data and they require some planning, you will look at data warehouse migration approaches, processes, and tools. This is a high-level overview of data warehouse migrations. We have other courses or tracks that you might want to take to dive deeper into these types of migrations.



The AWS Data Warehouse Migration Journey involves analyzing and planning the migration, conducting pilot projects, and gradually migrating workloads. Testing, optimizing, and monitoring the migrated data warehouse are crucial for ensuring data integrity, performance, and efficiency. Collaboration, stakeholder involvement, and using AWS resources are essential for a successful migration.

To look closer, this journey typically involves the following phases:

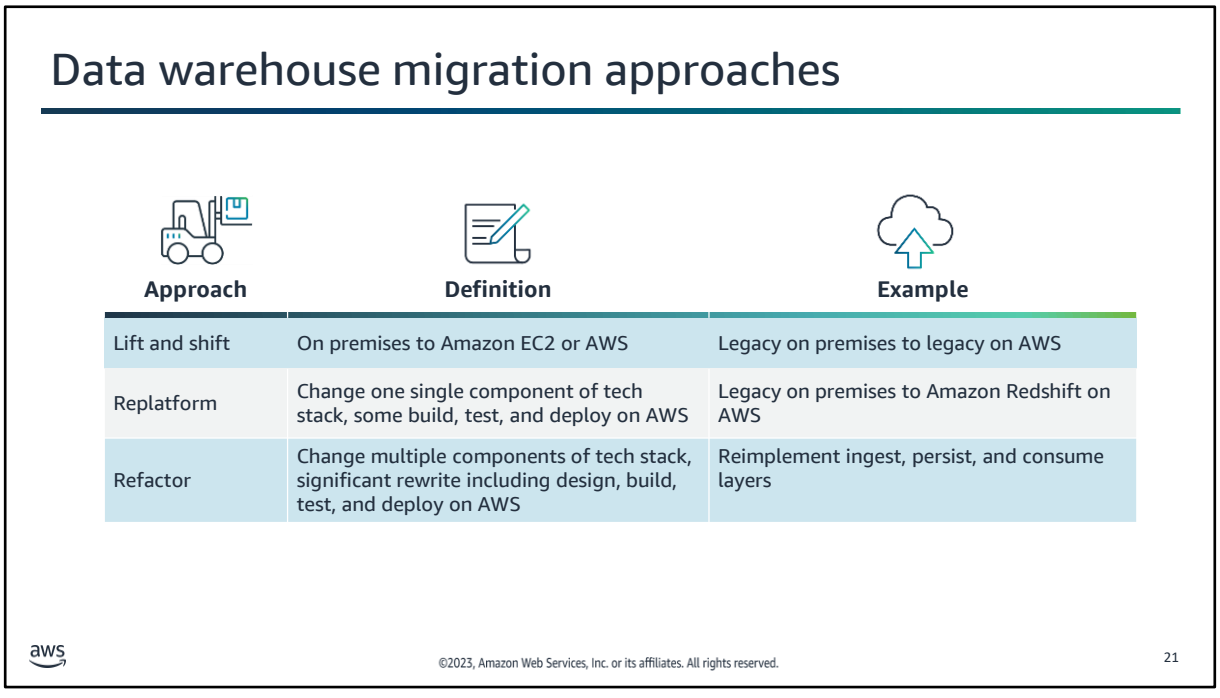
1. **Analyze and plan**: Begin by analyzing your current data warehouse environment and identifying the specific business and technical requirements for migration. Determine the target AWS data warehouse service that best aligns with your needs, such as Amazon Redshift or Amazon Athena. Develop a migration plan, including a timeline, resource allocation, and a comprehensive understanding of data sources, dependencies, and transformations.
2. **Workshop and pilot**: During the workshop and pilot phase, it is important to involve the relevant stakeholders, such as end-users, IT teams, and data experts. Their feedback and insights will help refine the migration plan and ensure its success. Any issues or challenges encountered during the pilot phase should be thoroughly investigated and resolved before proceeding with the full-scale migration. The workshop and pilot phase includes the following steps:
    1. **Create a new target:** In this phase, create a new target for the migration plan. This involves identifying the requirements and specifications for the target data warehouse and setting up the necessary infrastructure.
    2. **Modify or develop BI and applications to dual target:** Then modify or develop the business intelligence (BI) tools and applications to support the dual target environment. This includes updating data models, queries, and reports to ensure compatibility with both the source and target data warehouses.

3. **Modify or develop ETL to dual target:** To migrate data to the dual target environment, modify or develop the extract, transform, and load (ETL) processes. This involves updating the data extraction, transformation, and loading logic to work with both the source and target data warehouses.
4. **Workshop skills and Center of Expertise (CoE):** Conduct workshops and pilot projects to validate the feasibility and effectiveness of the migration plan. This involves selecting a subset of data, migrating it to the target data warehouse, and evaluating the performance, functionality, and compatibility. Identify and address any issues or challenges encountered during the pilot phase.

3. **Continue workload migrations**: After the pilot phase is successful, proceed with the migration of the remaining workloads. You can do this in a phased approach, gradually migrating subsets of data or specific business units. Use AWS DMS or other relevant tools to facilitate the movement of data from the source data warehouse to the target AWS data warehouse service.

   Other aspects of continuous workload migration include:
   - Thoroughly test and validate the migrated workloads in the AWS data warehouse environment: Verify data integrity and query performance and functionality to ensure that the migrated workloads are functioning as expected. Conduct load testing and optimize the performance, as necessary.
   - Transform and optimize of data: During the migration process, take the opportunity to optimize and transform your data warehouse architecture. Use AWS services such as AWS Glue for data transformation, Amazon S3 for data storage, and Lambda for serverless processing. Implement best practices for schema design, partitioning, indexing, and query optimization to maximize the performance and cost-efficiency of your data warehouse.
   - Continuously monitor and optimize: Continuously monitor the performance and usage patterns of your AWS data warehouse. Use Amazon CloudWatch, AWS CloudTrail, and other monitoring tools to gain insights into resource use, query performance, and data access patterns. Use these insights to optimize your data warehouse configuration, adjust capacity, and improve the overall efficiency.

     Throughout the migration journey, collaborate with stakeholders, involve the necessary teams (such as database administrators, developers, and business analysts), and use AWS resources, documentation, and support. Regularly review and refine your migration plan based on feedback and lessons learned to ensure a smooth and successful transition to an AWS data warehouse environment.



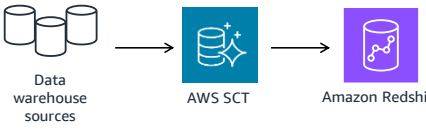
The choice of the migration approach depends on factors such as the complexity of the existing data warehouse, business requirements, time constraints, and the desired level of optimization and innovation. Each approach has its advantages and considerations. Organizations should carefully evaluate their specific needs and goals before deciding on the most suitable approach for their data warehouse migration.

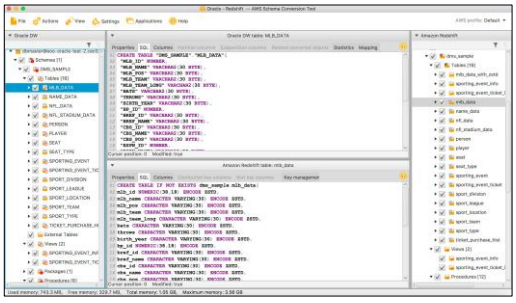
Data warehouse migration approaches can be categorized into the following three types:

- **Lift and shift** is when the existing data warehouse is migrated to the cloud environment with minimal changes. The goal is to replicate the existing infrastructure and configurations as closely as possible in the cloud. The data, schema, and application code are migrated without significant modifications. This approach offers a quick and straightforward migration process and minimizes disruptions. However, it might not fully use the benefits and capabilities of the cloud.
- **Replatform** involves migrating the data warehouse to the cloud while also making some structural and architectural modifications. The focus is on optimizing the data model, schema, and configurations to take advantage of the cloud-focused features and capabilities. This involves redesigning the schema, optimizing indexing and partitioning strategies, and reevaluating data integration and transformation processes. The goal is to enhance performance, scalability, and cost efficiency in the cloud environment.
- **Refactor** is when the existing data warehouse is completely reimagined and rebuilt from scratch in the cloud environment. The migration involves rewriting the application code, redesigning the data model, and rearchitecting the entire data warehouse solution. The emphasis is on using the capabilities and services offered by the cloud, such as serverless computing, managed services, and advanced analytics. This approach provides maximum flexibility, scalability, and optimization. It also requires a significant investment in terms of time, resources, and expertise.

Data warehouse migration using AWS SCT

- Convert schemas for source data warehouses.
- Migrate and modernize to AWS services, such as Amazon Redshift.
- Generate a detailed migration assessment report.
- Complete automatic schema optimization.
- Use AWS SCT data migration agents to extract, prepare, optimize, and upload data securely and in parallel.





AWS SCT can convert schemas for source data warehouses to a supported target. Supported source data warehouses include Snowflake, Azure Synapse Analytics, Oracle, Teradata, Greenplum, IBM Netezza, HPE Vertica, MSSQL Server, and Amazon Redshift.

Migrate and modernize your database and data warehouse schemas to AWS services such as Amazon Redshift.

AWS data warehouse migration using the AWS SCT involves assessing the existing data warehouses, installing and configuring AWS SCT, extracting and converting the schema, migrating the data, testing and validating the migrated data warehouse, and optimizing performance.

AWS SCT simplifies the migration process by automating conversions and providing guidance for a successful migration.

For more information, see https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_Source-Data-Warehouses.html.

AWS DMS compared to AWS SCT extractors

Q: For data warehouse migrations, when do you use AWS DMS compared to AWS SCT extractors?

A: It depends on customer use case. The following table shows some examples.

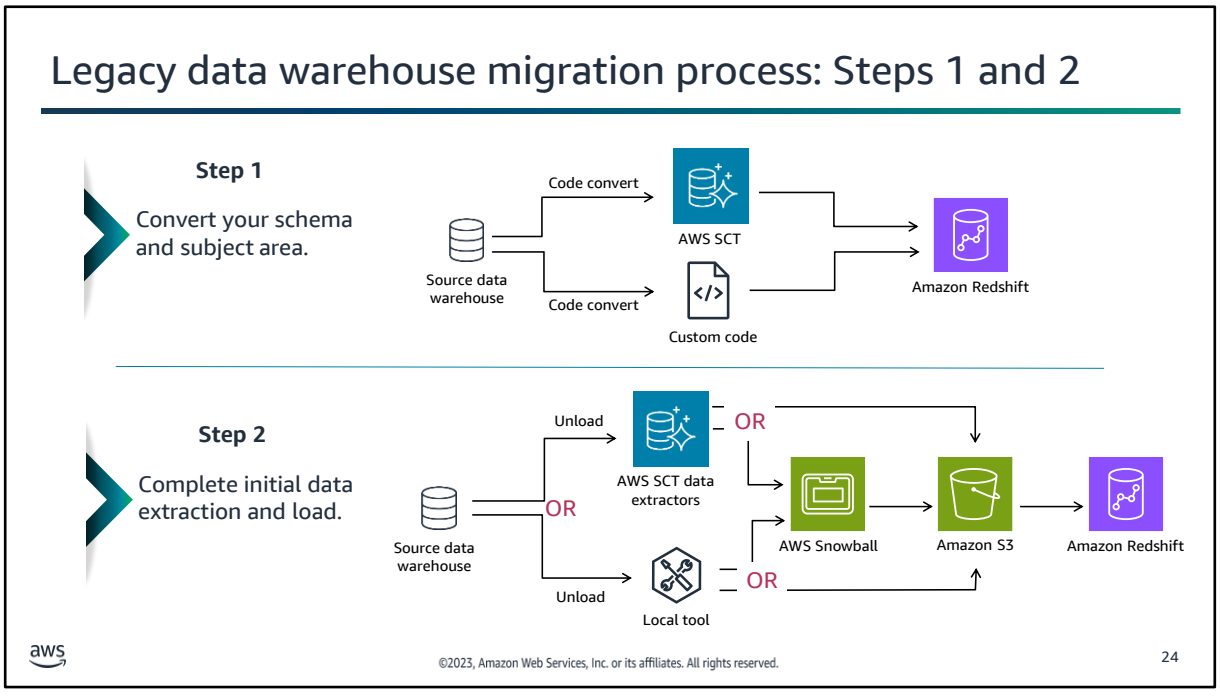
	Sources	Full load	CDC	Managed Service	Configuration options
AWS DMS	Many	**Yes**	Yes	Yes	Some
AWS SCT extractors	Data warehouse only	Yes	No	No	Many

23

When considering AWS DMS compared to AWS SCT extractors for a data warehouse migration, you must understand their respective capabilities. AWS DMS excels at capturing real-time data changes from the source database and replicating them to the target. AWS DMS ensures continuous synchronization between the two databases. This service is ideal for ongoing replication scenarios.

On the other hand, AWS SCT extractors are specifically designed to analyze the schema and content of the source database. They play a crucial role in the schema conversion and data extraction process. By analyzing the structure and metadata of database objects, AWS SCT extractors convert the schema into a format compatible with the target database.

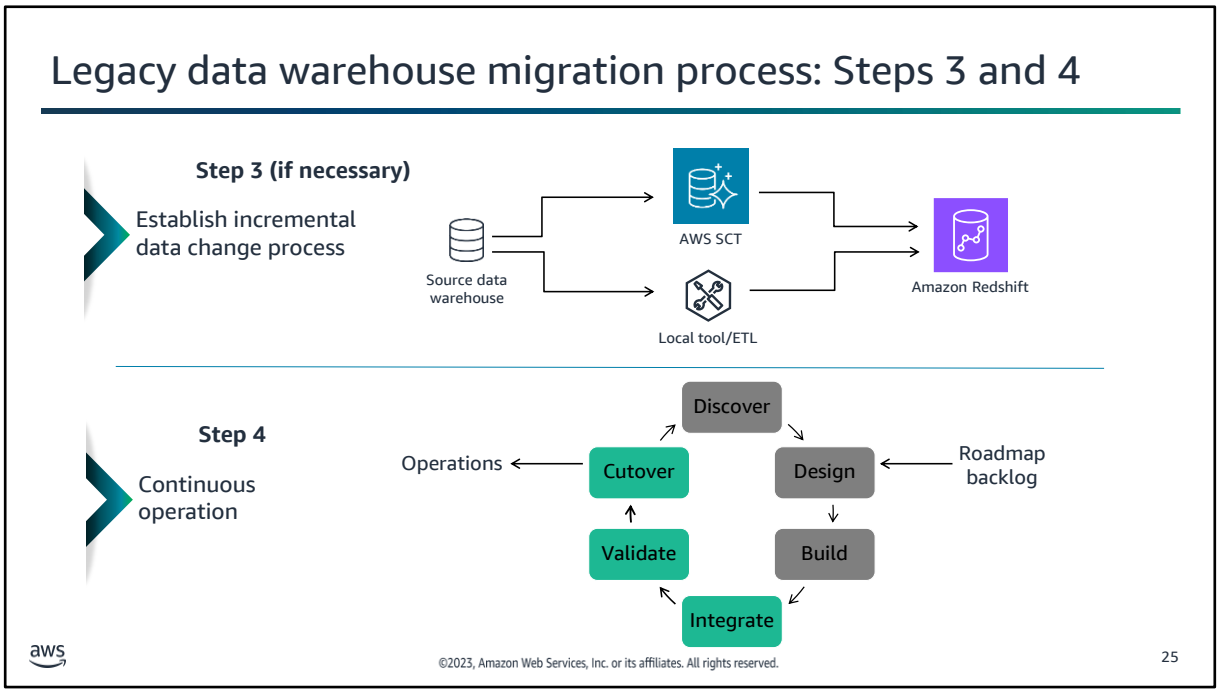
When deciding between the two, consider your migration requirements. If real-time replication and continuous synchronization are key factors, AWS DMS is the appropriate choice. However, if schema conversion and data extraction are primary concerns, AWS SCT extractors are well-suited to handle these tasks. Ultimately, the choice depends on the specific goals and requirements of your migration project. Assess factors such as the need for real-time replication, schema conversion, and data extraction to determine whether AWS DMS or AWS SCT extractors are the right fit for your data warehouse migration.



By following these steps, organizations can smoothly migrate their legacy data warehouse to a new environment. When an organization converts the schema and subject areas, they extract and load the data, synchronize incremental changes, and establish an efficient migration factory.

Step 1: Convert your schema and subject area: Begin by converting the existing schema and subject areas to align with the target data warehouse architecture. This can include redesigning the data model, optimizing indexing strategies, and ensuring compatibility with the new platform.

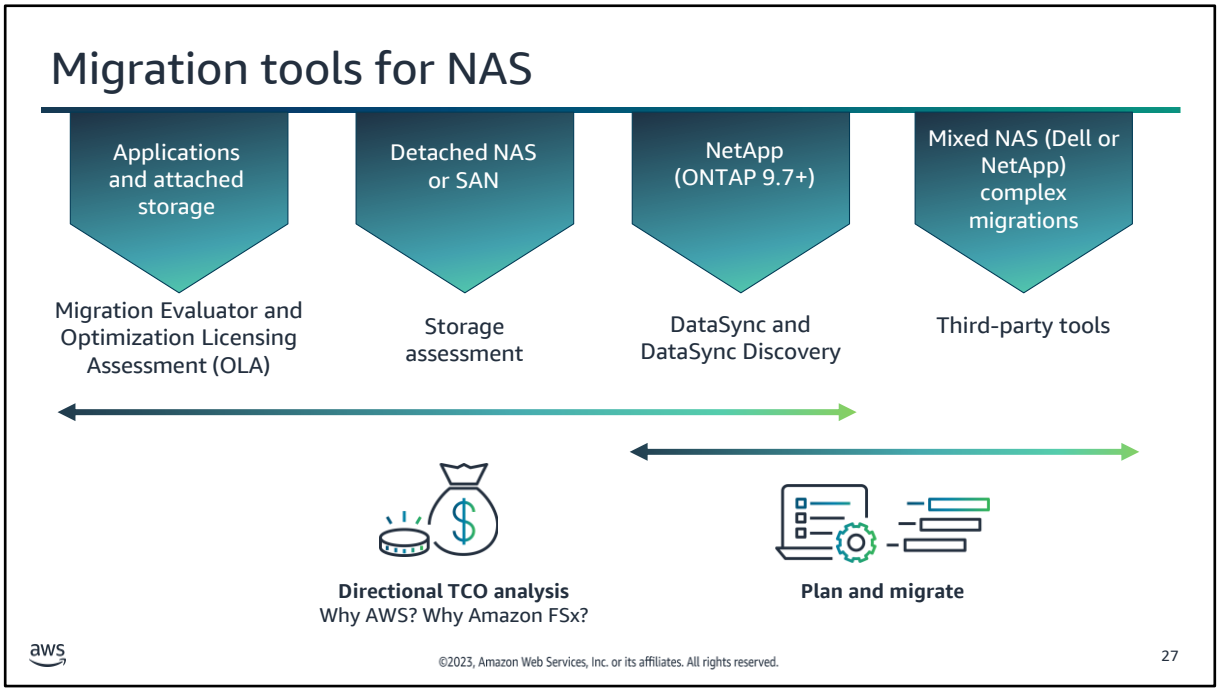
Step 2: Complete initial data extraction and load: AWS SCT has an extractor agent that transforms data during complex migration. Extract data from the legacy data warehouse and load it into the new environment. This step transfers large volumes of data to the target data warehouse using tools like AWS DMS or custom-built ETL processes.



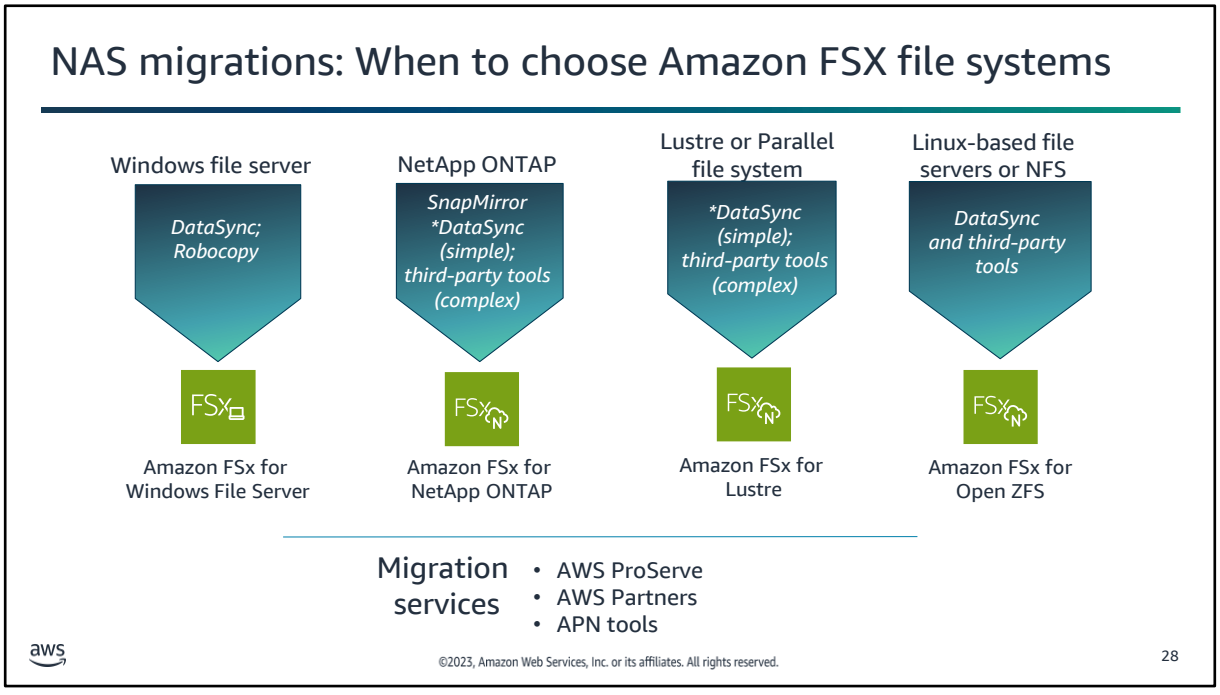
Step 3: Establish incremental data change process: Establish a process for capturing and loading incremental data changes from the legacy system to the new data warehouse. This ensures that any new or modified data is synchronized and kept up-to-date in the target environment. This step can be performed on-demand or scheduled at regular intervals based on the specific requirements.
For prescriptive guidance, see https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/incrementally-migrate-from-amazon-rds-for-oracle-to-amazon-rds-for-postgresql-using-oracle-sql-developer-and-aws-sct.html

Step 4: Large-scale data warehouse migration presents a challenge in terms of project complexity and poses a risk in terms of resources, time, and cost. You can significantly reduce the complexity of migrating your legacy data warehouse and workloads with a waved-based migration methodology. It's an iterative process, you're going to integrate it with your applications, validate it all works, then finally you will do the cutover.

In this section, you will learn to migrate your data when it is very large in size. To do this, you use NAS migration tools.



Migrating a NAS solution to AWS involves careful planning and following a series of steps to ensure a smooth transition. AWS provides several tools and services that can assist in assessing, planning, and implementing application and storage migrations to the cloud. Additionally, third-party tools are available to complement AWS offerings.



AWS provides two services, Amazon FSx for Windows File Server and Amazon FSx for NetApp ONTAP, that cater to different target users and offer specific benefits for Windows file server deployments. Let's highlight the key features and advantages of each service:

FSx for Windows File Server:

Target audience: FSx for Windows File Server is suitable for Windows administrators and users who are not using NetApp on premises.

Use cases: You can use FSx for Windows File Server for a wide range of scenarios such as user and group shares, SQL Server high availability (Failover Clustering), Windows applications, and virtual desktop infrastructure (VDI).

Benefits: FSx for Windows File Server supports the SMB protocol. It is compatible with Windows-based applications and systems. Because FSx for Windows File Server uses Windows Servers on the backend, it has compatibility with existing Windows infrastructure. FSx for Windows File Server also provides on-premises caching through the Amazon FSx File Gateway, which improves performance and reduces latency. It integrates with AWS services like Amazon Kendra for enhanced search capabilities. It also integrates with Amazon S3 and Kinesis Firehose for seamless audit log integration.

FSx for ONTAP:

Target audience: FSx for ONTAP is suitable for NetApp users who are already using NetApp storage solutions on premises.

Use cases: You can use FSx for ONTAP for scenarios that require user and group shares, SQL Server high availability (Failover Clustering), Windows applications, and integration with VMware Cloud (VMC).

Benefits: FSx for ONTAP supports both SMB and iSCSI protocols. It has flexibility to access and manage data. It offers larger file system sizes, including file systems larger than 64TB. It can scale to petabyte-

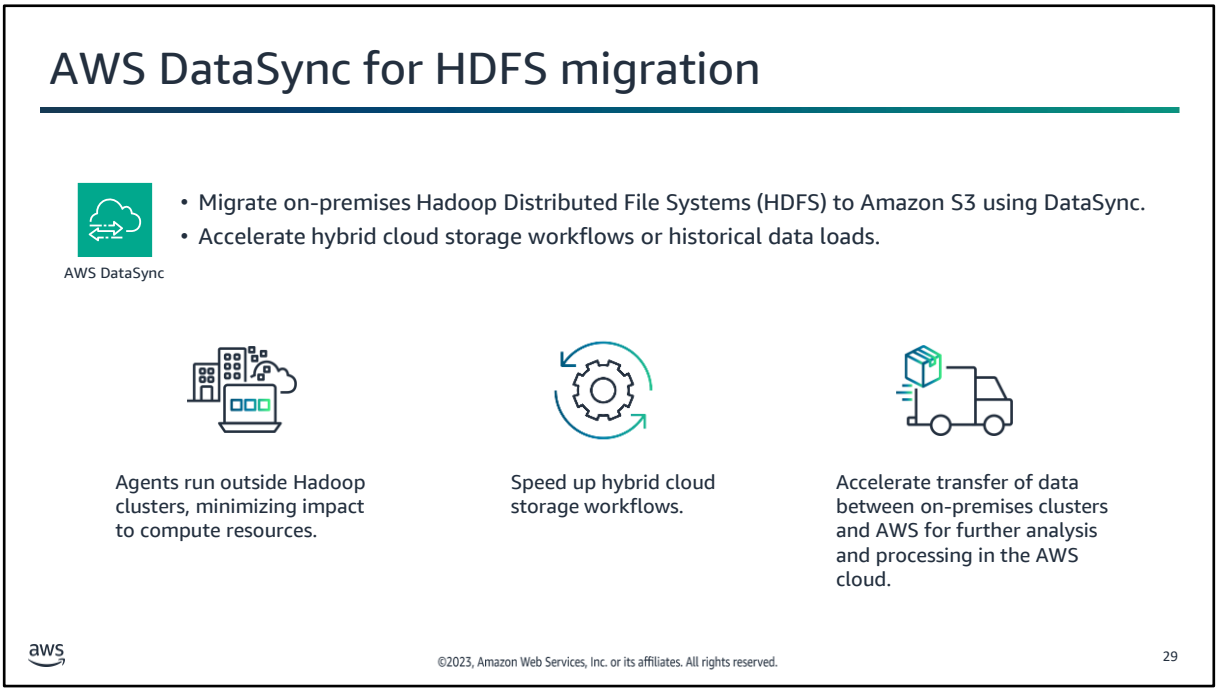
level file systems without the need for a Distributed File System (DFS) namespace server. FSx for ONTAP includes features like cloning, replication, snapshots, compression for storage efficiency, and intelligent tiering for optimized performance and cost savings.
By providing these specialized services, AWS meets user needs and preferences. FSx for Windows File Server is particularly suited for Windows environments, offering seamless integration and compatibility. FSx for ONTAP targets NetApp users who need specific features and capabilities that align with their existing on-premises infrastructure.

Amazon FSx for Lustre:

Target audience: FSx for Lustre is suitable for Lustre file system users who like a fully managed service that provides cost-effective, high-performance, scalable storage for compute workloads.
Use cases: You can use Amazon FSx for Lustre for scenarios to accelerate machine learning, enable high performance computing (HPC), unlock big data analytics, and increase media workload agility
Benefits: It is highly-performant and scalable, easy to use with S3 data, and cost effective

Amazon FSx for Open ZFS:

Target audience: Existing Open ZFS users
Use cases: You can use Amazon FSx for Open ZFS to migrate your workloads to AWS seamlessly, deliver insights for data analytics workloads, accelerate content management, and increase dev/test velocity.
Benefits: Migrate Linux file servers to AWS, simplify building and testing applications, accelerate workloads with high-performance storage, optimize costs
For more information on how to choose, see: https://aws.amazon.com/fsx/when-to-choose-fsx.
For more information on third party tools see: https://partners.amazonaws.com/search/partners?facets=Use%20Case%20%3A%20Migration%20and%20Modernization%20%3A%20Data%20Mobility



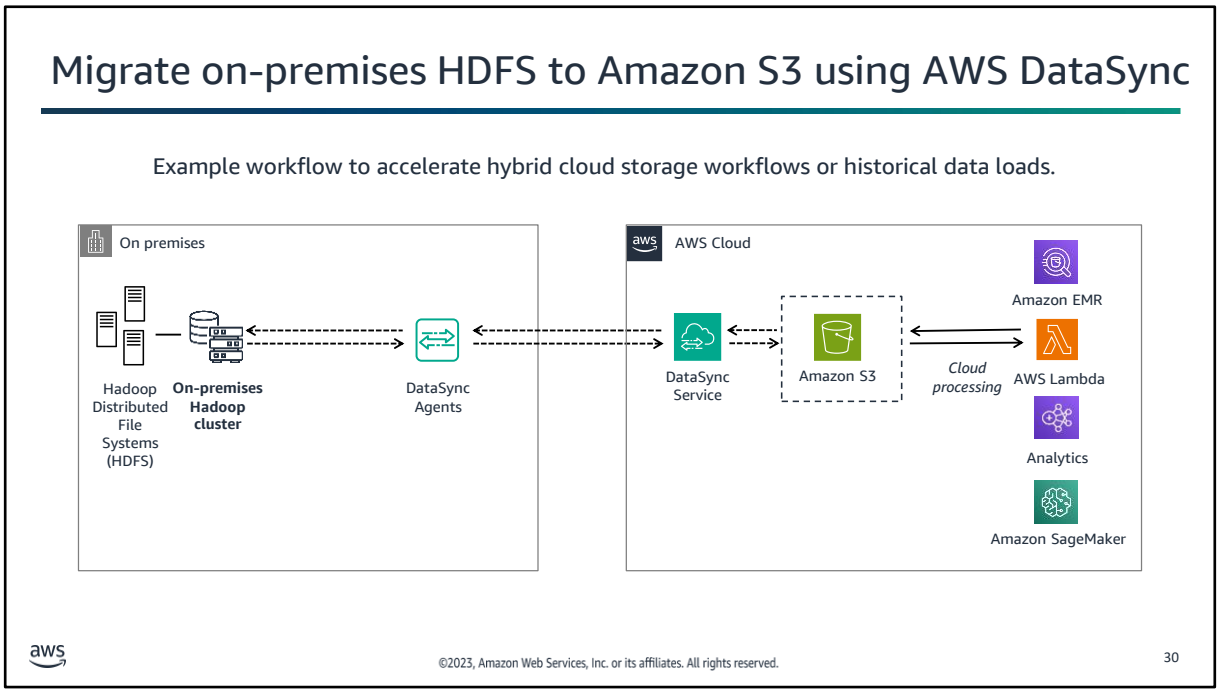
DataSync facilitates the migration of your Hadoop cluster to the cloud and takes advantage of the DataSync features and benefits. It supports transferring data between Hadoop Distributed File Systems (HDFS) and AWS Storage services. With this new capability, your customers can seamlessly move files and folders from their Hadoop clusters HDFS to various AWS storage options. They can populate their data lakes on Amazon S3, transfer data into Amazon EMR for processing, archive data to different Amazon S3 storage classes, or move data between their clusters and AWS for further analysis and processing.

DataSync is an online data migration service that provides a straightforward method for automating and expediting data copying over the internet or with AWS Direct Connect. It offers a range of features such as built-in scheduling, monitoring, encryption, and end-to-end data integrity validation. DataSync streamlines and automates the data copying process to and from AWS, with the flexibility of pay-as-you-go pricing.

In addition to HDFS support, DataSync also facilitates data transfers between network file system (NFS) shares, SMB shares, self-managed object storage. It also facilitates AWS Snowcone, Amazon S3 buckets, Amazon Elastic File System (Amazon EFS), FSx for Windows File Server file systems, and Amazon FSx for Lustre file systems.

DataSync provides comprehensive support for various storage systems. Because it can seamlessly transfer data between on-premises Hadoop clusters and AWS, your customers can use cloud scalability and optimize costs. They can also decouple storage from processing and unlock advanced capabilities for data analytics and machine learning on the AWS platform. For more information, see Using AWS

DataSync to Move Data from Hadoop to Amazon S3 at https://aws.amazon.com/blogs/storage/using-aws-datasync-to-move-data-from-hadoop-to-amazon-s3.



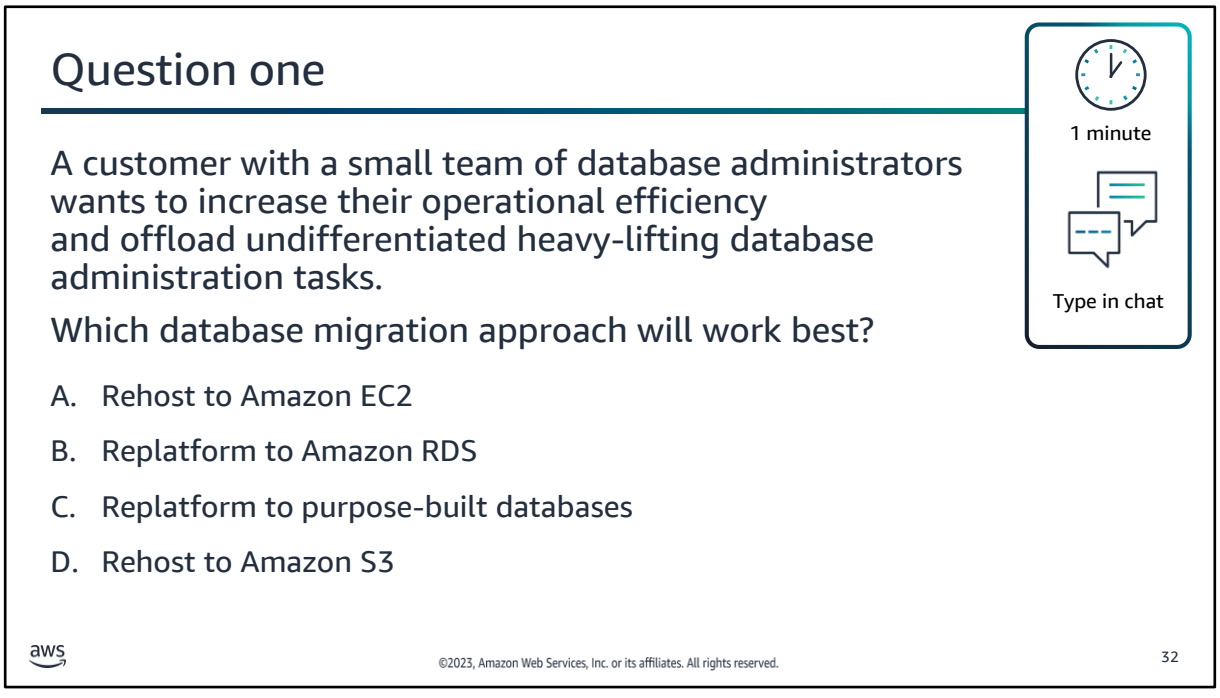
The following diagram shows a high-level overview of DataSync transferring files between on-premises HDFS and Amazon S3 with cloud processing of other AWS services.

The diagram illustrates a common DataSync use case:
A DataSync agent copying data from an on-premises storage system.
Data moving into AWS via Transport Layer Security (TLS).
DataSync copying data to a supported AWS storage service.

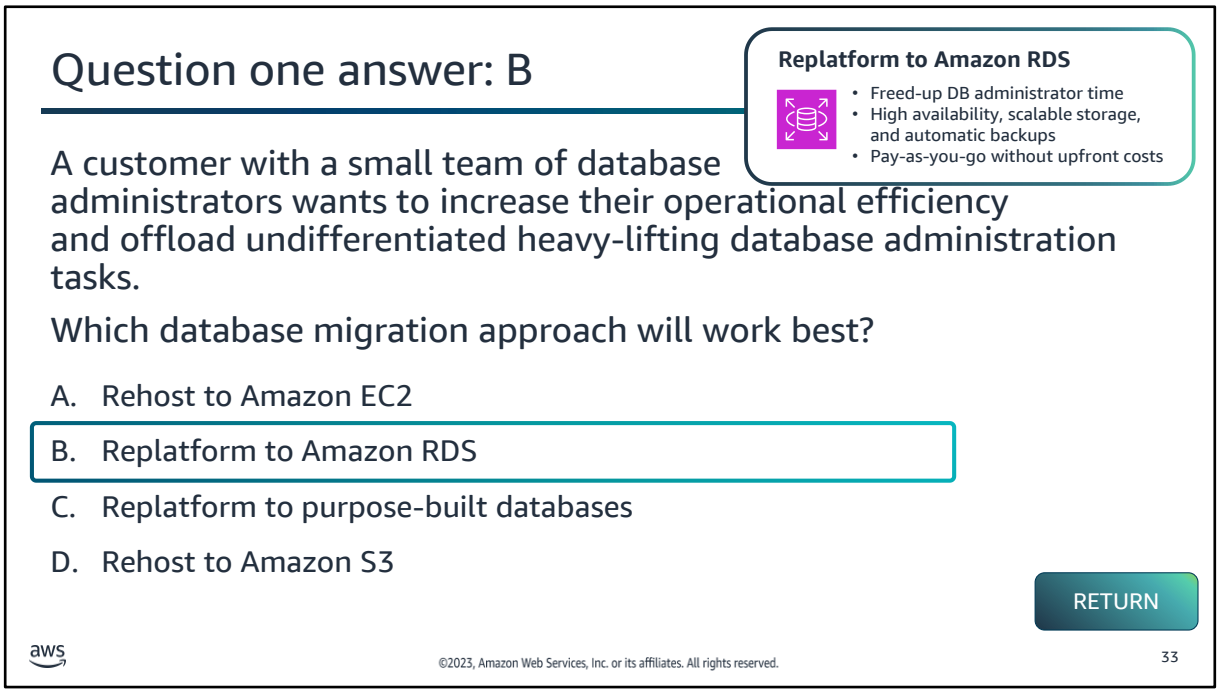
Review

aws

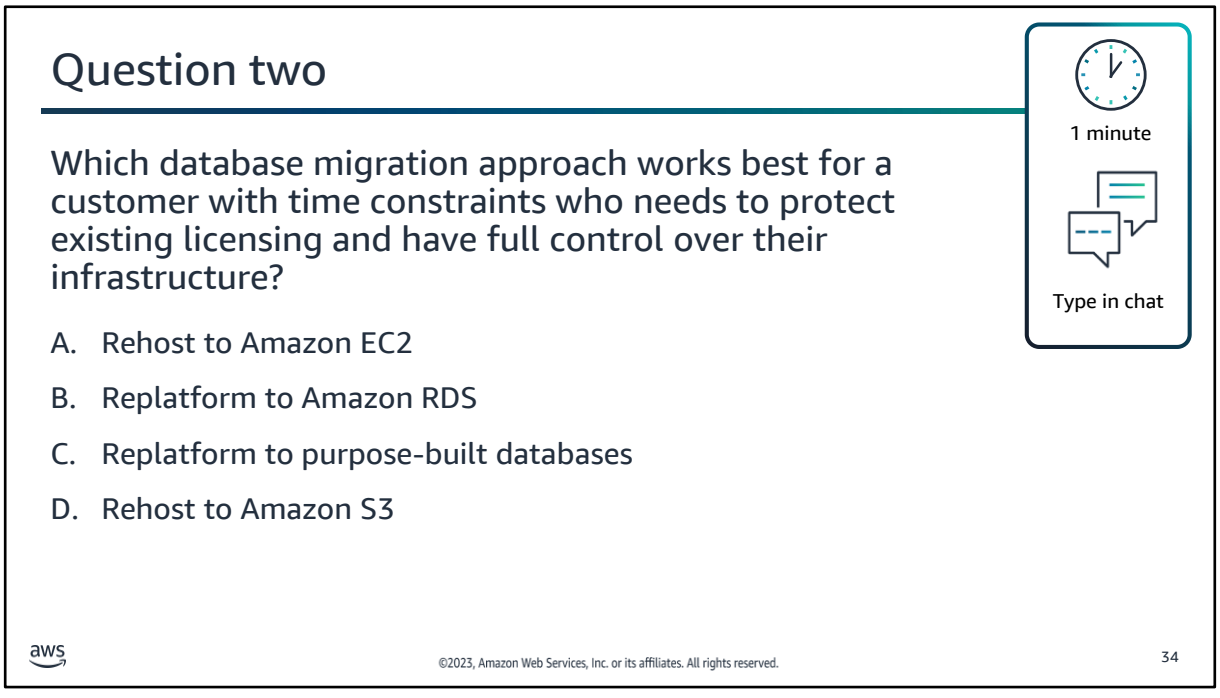
31



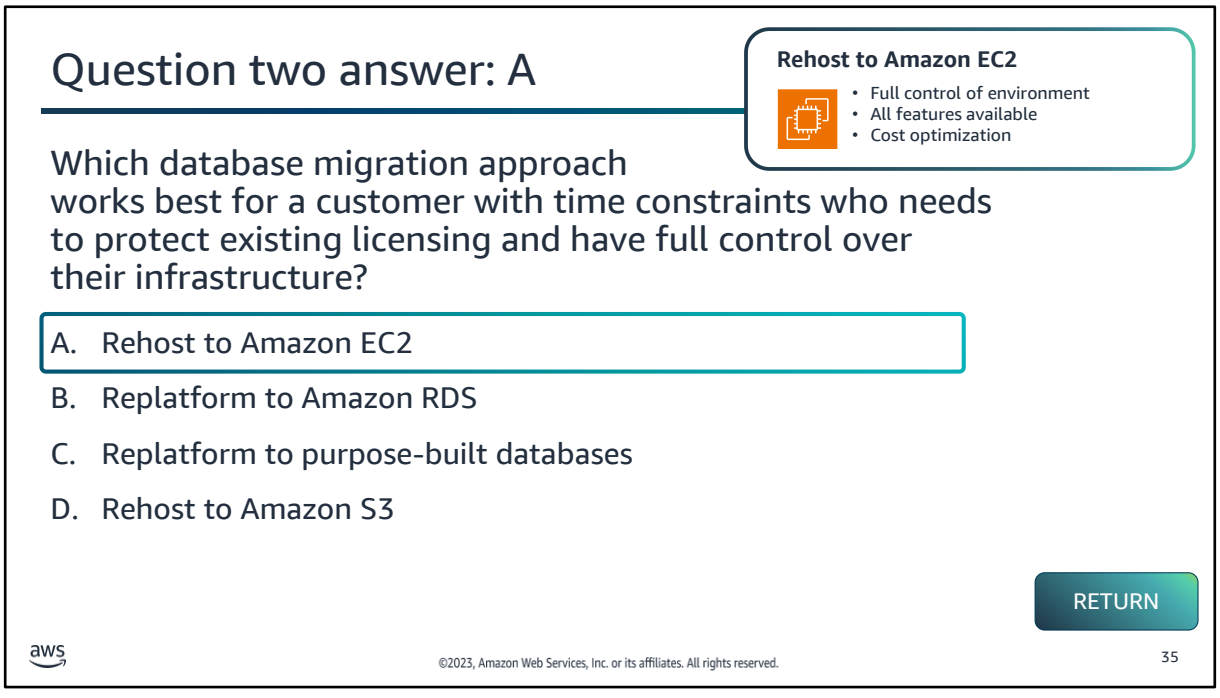
A customer with a small team of database administrators wants to increase their operational efficiency and offload undifferentiated heavy-lifting database administration tasks.
Which database migration approach will work best for this customer?
A. Rehost to Amazon EC2
B. Replatform to Amazon RDS
C. Replatform to purpose-built databases
D. Rehost to Amazon S3



A customer with a small team of database administrators wants to increase their operational efficiency and offload undifferentiated heavy-lifting database administration tasks. Which database migration approach will work best for this customer?
A. Rehost to Amazon EC2 would work for customer with time constraints, needs to protect existing licensing, and needs to control their infrastructure.
B. Replatform to Amazon RDS – correct answer
C. Replatform to purpose-built databases
D. Rehost to Amazon S3



Which database migration approach works best for a customer with time constraints who needs to protect existing licensing and have full control over their infrastructure?
A. Rehost to Amazon EC2
B. Replatform to Amazon RDS
C. Replatform to purpose-built databases
D. Rehost to Amazon S3



Which database migration approach works best for a customer with time constraints who needs to protect existing licensing and have full control over their infrastructure?

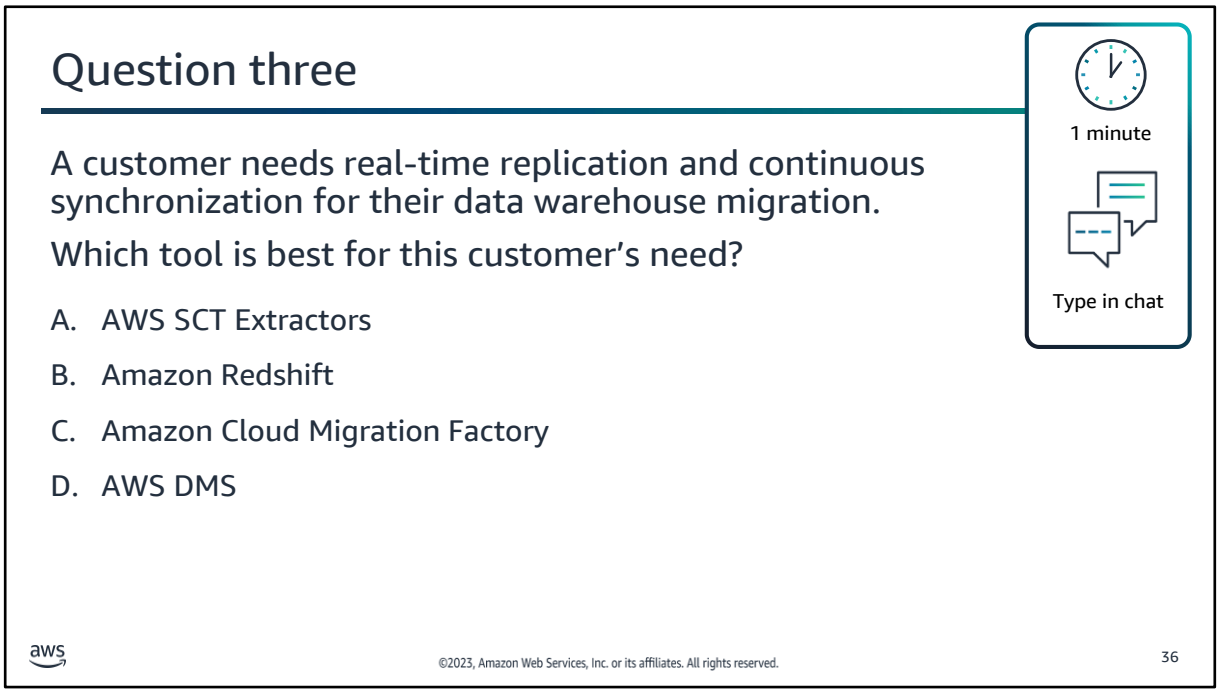
A. Rehost to Amazon EC2-correct answer

B. Replatform to Amazon RDS

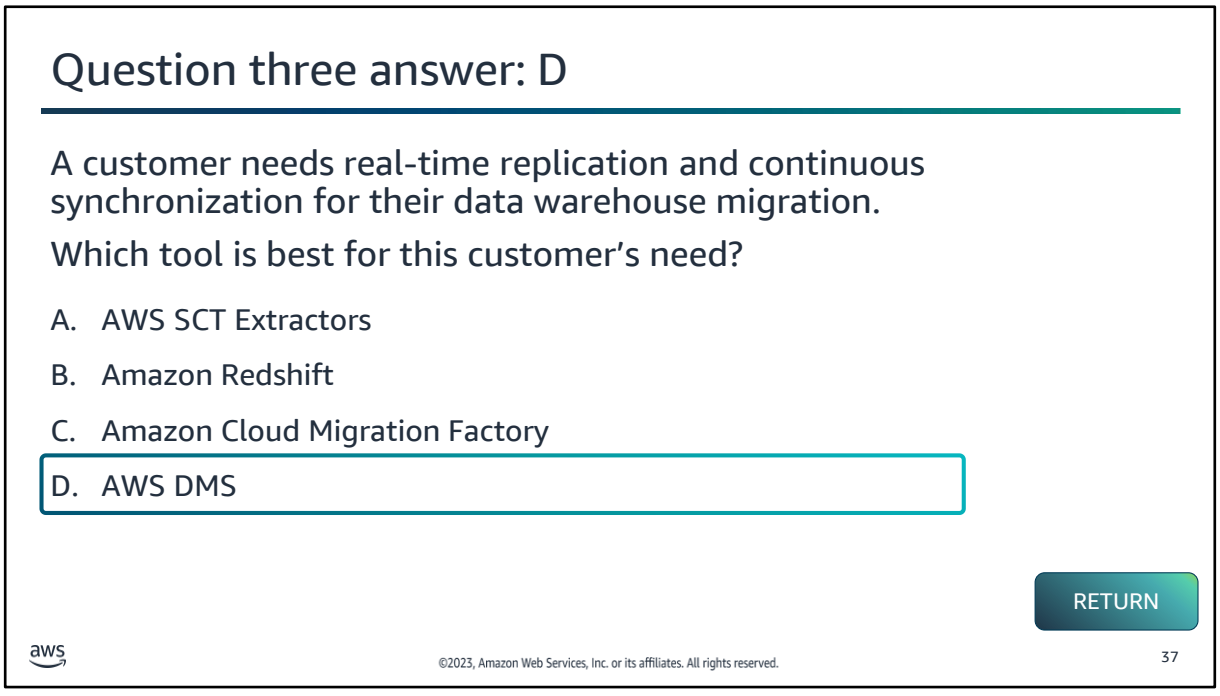
C. Replatform to purpose-built databases

D. Rehost to Amazon S3

The correct answer is A – Rehost to Amazon EC2 because it is fastest and easiest migration which works because the customer is in a hurry. It provides full control of their environment, and they have all features available.

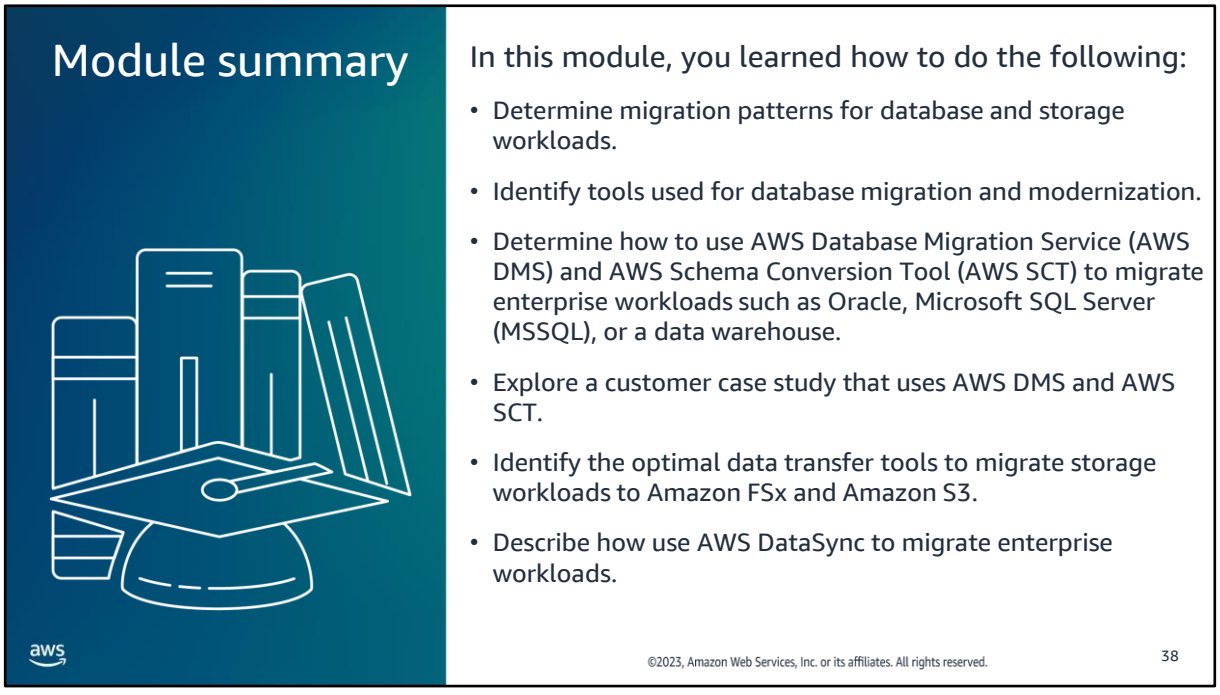


The customer needs real-time replication and continuous synchronization for their data warehouse migration. Which tool is best for this customer's need?
A. AWS DMS
B. AWS SCT extractors
C. Amazon Redshift
D. Amazon Cloud Migration Factory



The customer needs real-time replication and continuous synchronization for their data warehouse migration. Which tool is best for this customer's need?
A. AWS Schema Conversion Tool (AWS SCT) extractors: AWS SCT extractors is not correct. The AWS SCT extractor uploads to Amazon S3 or AWS Snow Family device. AWS DMS serves the same purpose as AWS SCT extractor but is using AWS DMS to do the continuous copy.
B. Amazon Redshift
C. Amazon Cloud Migration Factory
D. AWS Database Migration Service (AWS DMS) – is correct.

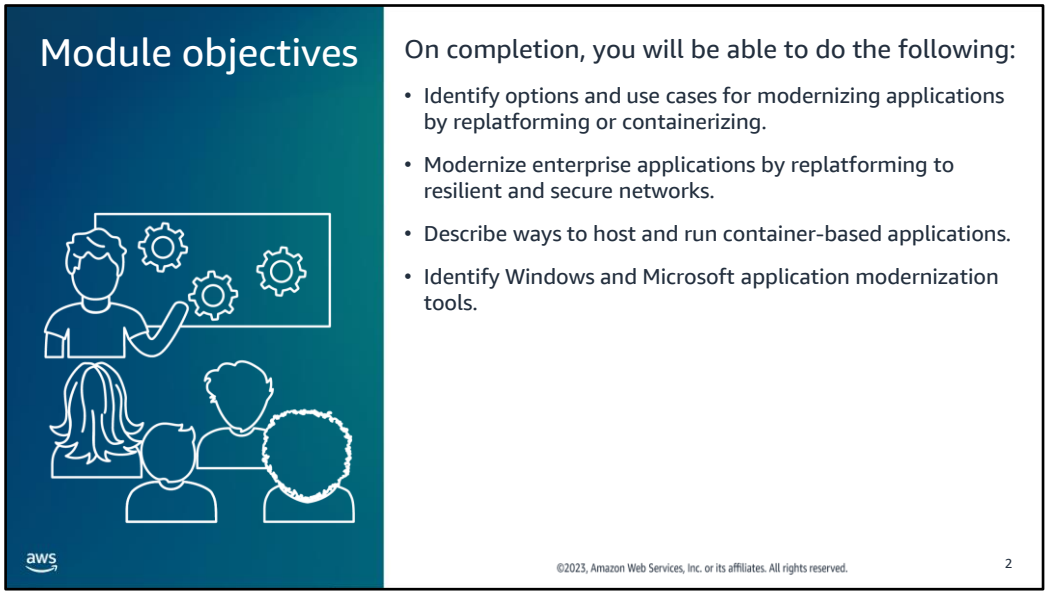
The correct answer is D. For real-time replication and continuous synchronization for their data warehouse migration, the customer should use AWS DMS.



In this module, you learned to do the following:

- Determine migration patterns for database and storage workloads.
- Identify tools used for database migration and modernization.
- Determine how to use AWS DMS and AWS SCT to migrate enterprise workloads such as Oracle, MSSQL, and data warehouse.
- Explore how AWS DMS and AWS SCT are used in a customer case study.
- Identify the optimal data transfer tools to migrate storage workloads to Amazon FSx, Amazon S3.
- Describe how DataSync is used to migrate enterprise workloads.

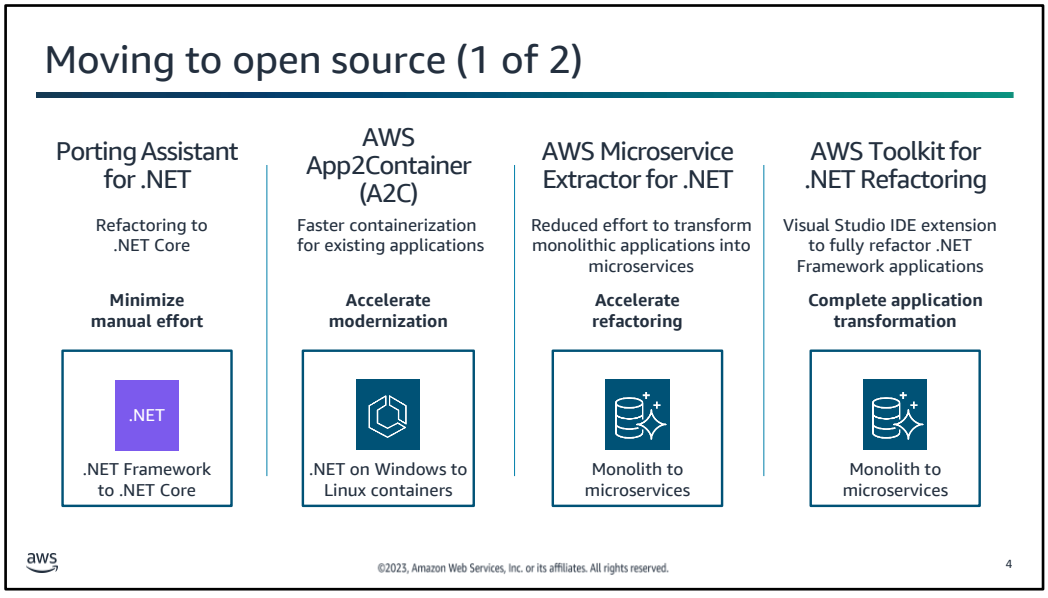
Welcome to Module 4: Refactor and Modernize Applications.



On completion of this module, you will be able to do the following:

- Identify options and use cases for modernizing applications by replatforming or containerizing.
- Modernize enterprise applications by replatforming to resilient and secure networks.
- Describe ways to host and run container-based applications.
- Identify Windows and Microsoft application modernization tools.

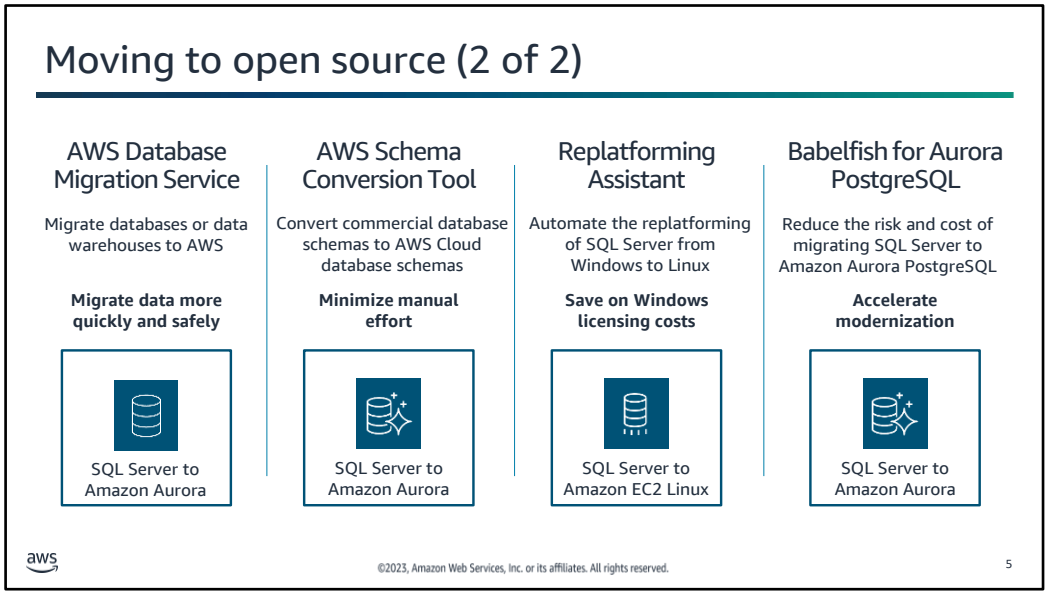
In this section, you learn about Windows and Microsoft application modernization tools.



AWS offers the following tools to expedite your modernization journey away from commercial licenses and unlock innovation.

- Porting Assistant for .NET helps refactor the old .NET implementation to the latest .NET Core.
- App2Container (A2C) is a command-line tool to modernize .NET and Java applications into containerized applications.
- AWS Microservice Extractor for .NET transforms monolithic applications into microservices you can use.
- Visual Studio IDE extension helps modernize applications.

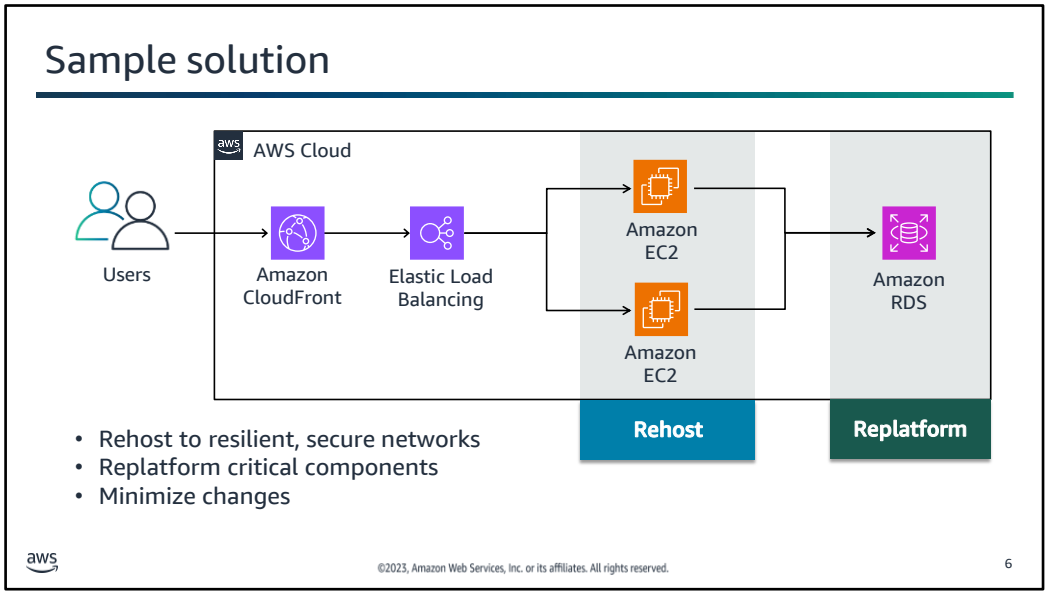
As a result, you have increased agility and time to market with ample room for experimentation.



AWS also offers alternatives for databases and computing. You can use AWS Database Migration Service (AWS DMS) in conjunction with AWS Schema Conversion Tool (AWS SCT) to move away from commercial license-based databases.

Two additional replatforming tools offered by AWS are:
Replatforming Assistant:
The Windows to Linux replatforming assistant for Microsoft SQL Server database service is a scripting tool. It helps you move existing Microsoft SQL Server workloads from a Windows OS to a Linux OS. You can use the replatforming assistant with any Windows Server virtual machines (VMs) hosted in the cloud or with on-premises environments running Microsoft SQL Server 2008 and later.

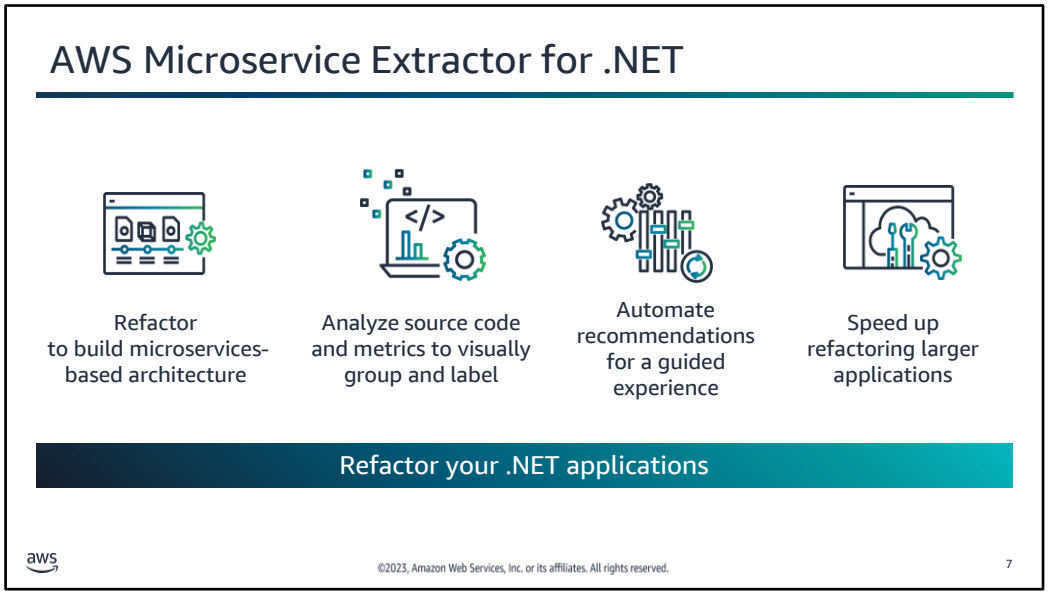
Babelfish for Amazon Aurora:
Babelfish for Amazon Aurora PostgreSQL runs SQL Server applications on PostgreSQL with little to no code changes. It keeps existing queries: With the translation layer, Aurora PostgreSQL understands Microsoft SQL Server T-SQL. It runs T-SQL code side-by-side with new open source functionality and continue developing with familiar tools and lowers risk and completes modernizations faster.



This solution shows a basic approach to modernization. The migration process combines rehosting and replatforming with minimal changes.

Rehosting migrates existing servers to Amazon EC2 instances. Replatforming moves the database infrastructure to Amazon RDS. Infrastructure changes added Amazon CloudFront Elastic Load Balancing to provide scalability and resilience to the rehosted applications and servers.

By migrating existing applications and services to Amazon EC2 instances, customers gain the benefits of reliable, scalable computing in a secure network. They will not have to deconstruct their applications, which refactoring sometimes requires.



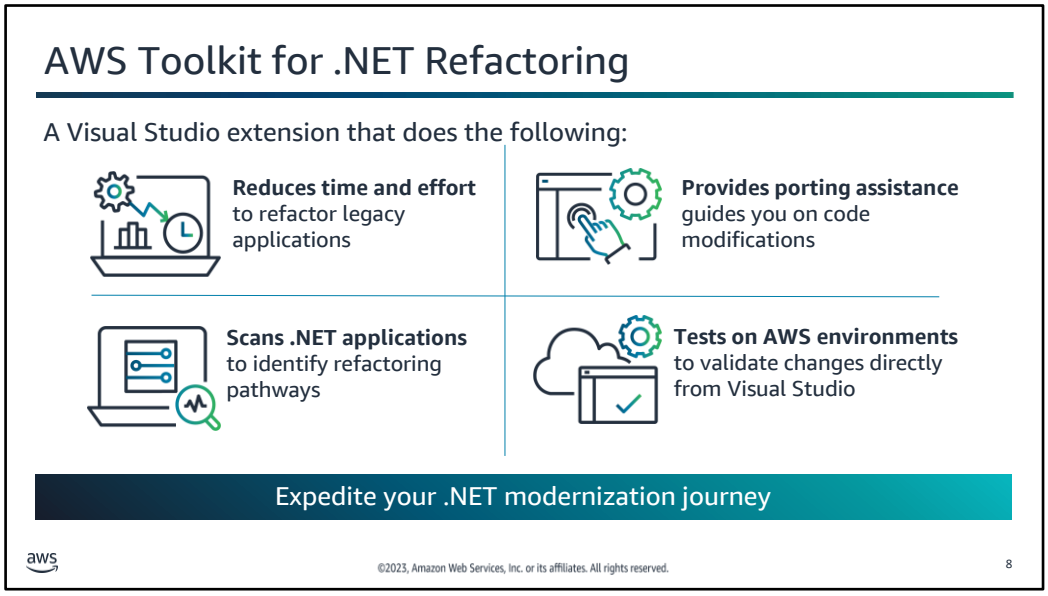
AWS Microservice Extractor streamlines the process of refactoring older monolithic applications into smaller code projects to build a microservices-based architecture.

Modernize and transform applications with an assistive tool that analyzes source code and runtime metrics to create a visual representation of the application and its dependencies.

With AWS Microservice Extractor providing automated recommendations, developers get guided experience to refactor legacy applications. AWS Microservice Extractor identifies common extraction candidates using heuristics-based techniques. You can use these recommendations as is or as a starting point to extract microservices off a monolithic code base.

Automated recommendations from AWS Microservice Extractor help to speed up refactoring large applications even if the developer is unfamiliar with the code base. You can also extract the code base into separate projects that teams can develop, build, and operate independently to improve agility, uptime, and scalability.

For more information, see AWS Microservice Extractor for .NET at https://aws.amazon.com/microservice-extractor.



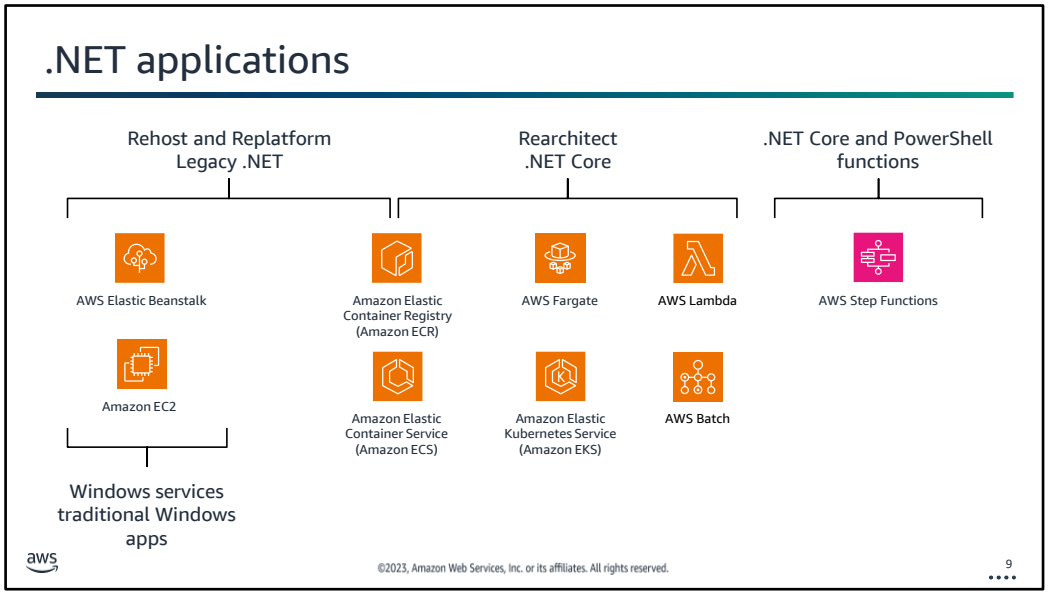
AWS Toolkit for .NET Refactoring is a Visual Studio extension that reduces the time and effort required for developers to refactor legacy .NET applications to cloud-based alternatives on AWS running on .NET Core 3.1, .NET 5, or .NET 6.

AWS Toolkit for .NET Refactoring assesses the application source code to recommend possible modernization pathways, such as porting to .NET Core. It also identifies Windows-specific IIS and Active Directory dependency configurations, performs code modifications where possible to enable Linux compatibility, and helps validate or test the refactored application on AWS services. It does all this inside Visual Studio IDE.

It eliminates the burden of learning multiple tools or working across multiple interfaces, removes the guesswork in application refactoring, and expedites the modernization journey with AWS.

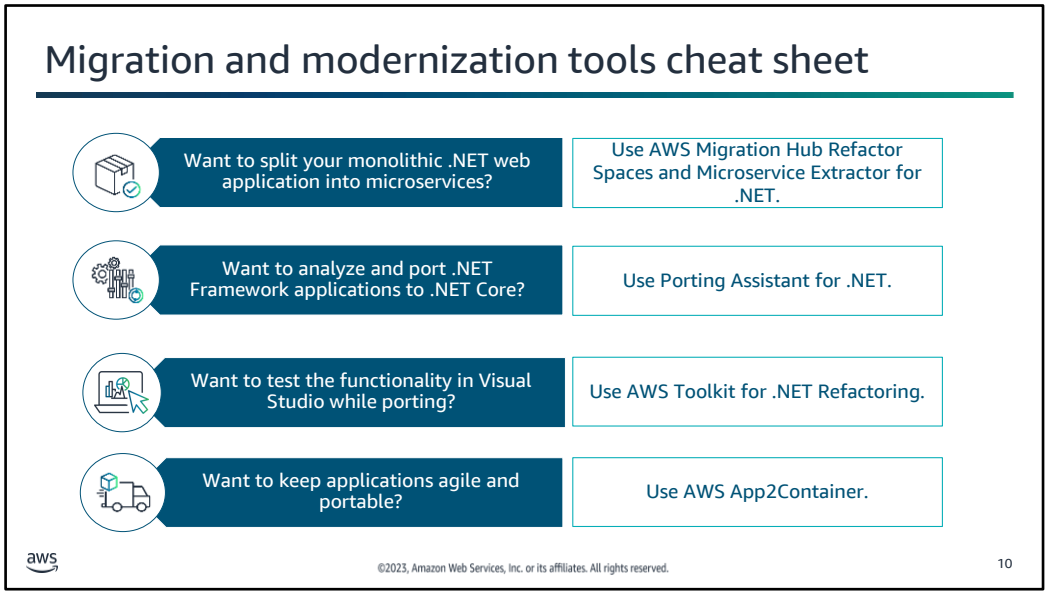
AWS Toolkit for .NET Refactoring is available as a Visual Studio IDE extension for Visual Studio 2019 and Visual Studio 2022.

To learn more about the Visual Studio 2019 version, see AWS Toolkit for .NET Refactoring VS 2019 at https://marketplace.visualstudio.com/items?itemName=AWSTR.refactoringtoolkit2019.
To learn more about the Visual Studio 2022 version, see AWS Toolkit for .NET Refactoring VS 2022 at https://marketplace.visualstudio.com/items?itemName=AWSTR.refactoringtoolkit2022.



Although this is not a definitive list, the following examples show how you can deploy different types of .NET applications:

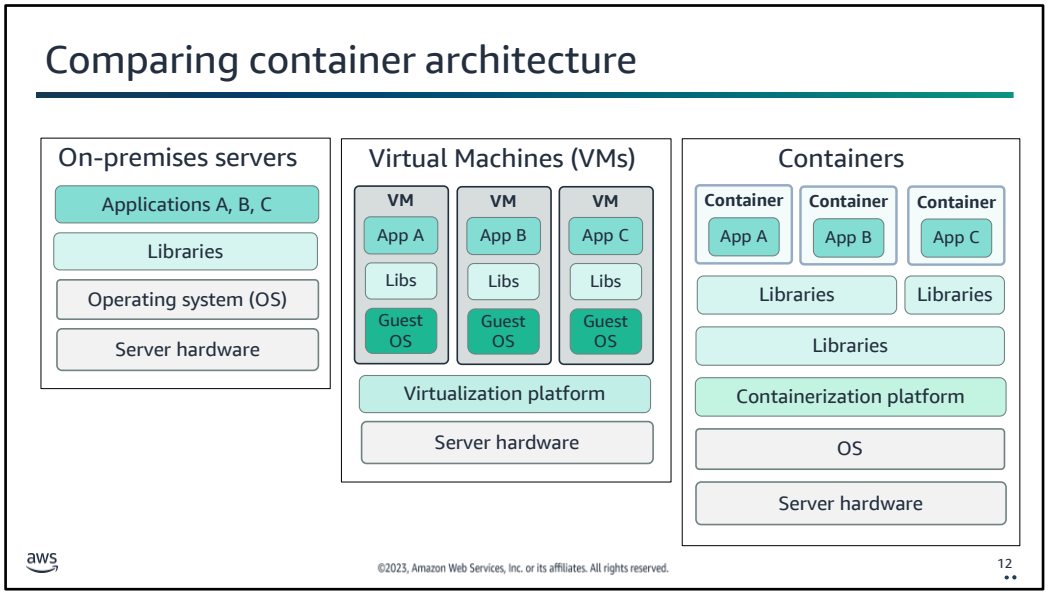
- For traditional rehosting, you can move to Amazon EC2 or replatform to AWS Elastic Beanstalk with few, if any, changes to the application.
- You can replatform to container-based technology that will use several AWS services, such as Amazon ECR, Amazon ECS, Amazon EKS, and AWS Fargate.
- You can also build serverless .NET applications on AWS Lambda and run code without provisioning or managing servers.
- AWS Batch is an alternative that supports .NET.
- You can build distributed .NET applications and orchestrate microservices using AWS Step Functions.



The following guidance helps you decide which tool to use to modernize Windows applications to AWS:

- If you want to split your monolithic .NET web application into microservices – Use AWS Migration Hub Refactor Spaces and Microservice Extractor for .NET.
- If you want to analyze and port .NET Framework applications to .NET Core – Use Porting Assistant for .NET.
- If you want to test the functionality as well in Visual Studio while porting – Use AWS Toolkit for .NET Refactoring.
- If you want to keep applications agile and portable – Use AWS App2Container.

You can modernize applications by running them in containers. In this section, you learn about containerizing applications.

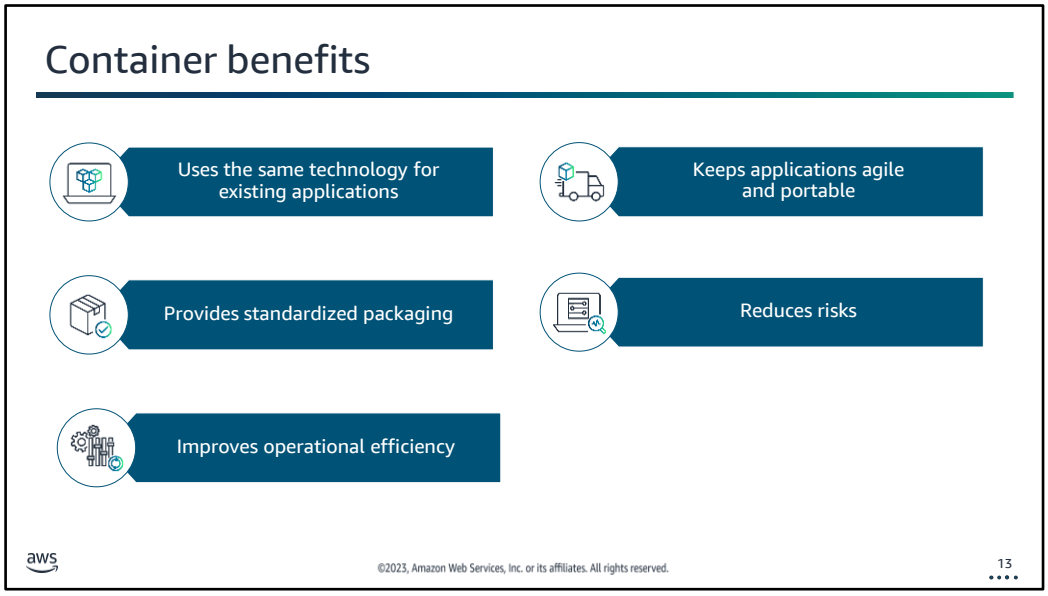


As we start the discussion on containers, let's look at the evolution of technology.

On-premises servers – Many customers are familiar with on-premises servers. On-premises servers run a standalone OS with one or many applications using libraries, such as Java. Costs remain constant, regardless of whether the server is in use. To scale, customers must buy and configure additional servers.

Virtual machines – Adding a virtualization layer on an on-premises server enables VMs. The virtualization layer is an abstraction from the underlying host OS and kernel resources. This layer has more separation. It uses more of the host CPU and RAM because more VMs can be added to a single physical machine. The virtualization layer, whether software or hardware, adds performance overhead. Furthermore, because a VM emulates a physical machine, each VM needs its own full guest OS, which also adds to overhead. Now, customers must manage and patch an OS for each of their VMs instead of one OS. Additionally, customers have the start-up cost of a full boot cycle associated with a VM.

Containers – Containers deliver efficiency and performance. Containers share a machine’s OS system kernel. They expose the underlying OS file system to provide for shared libraries, but they can permit individual libraries, as necessary. With a container image, containers start nearly instantly. The fast spin-up and spin-down of containers, with the immutable, read-only nature of the container image, provide quick scaling and high usage of the container host.



To accelerate application delivery and reduce costs, use container technology. Containers are effective, agile, and deliver applications quickly. By containerizing existing applications, customers can use the same technology for the applications they migrate. With containers, your customers can achieve a higher level of agility while also updating the skills of their teams with one of the latest technologies.

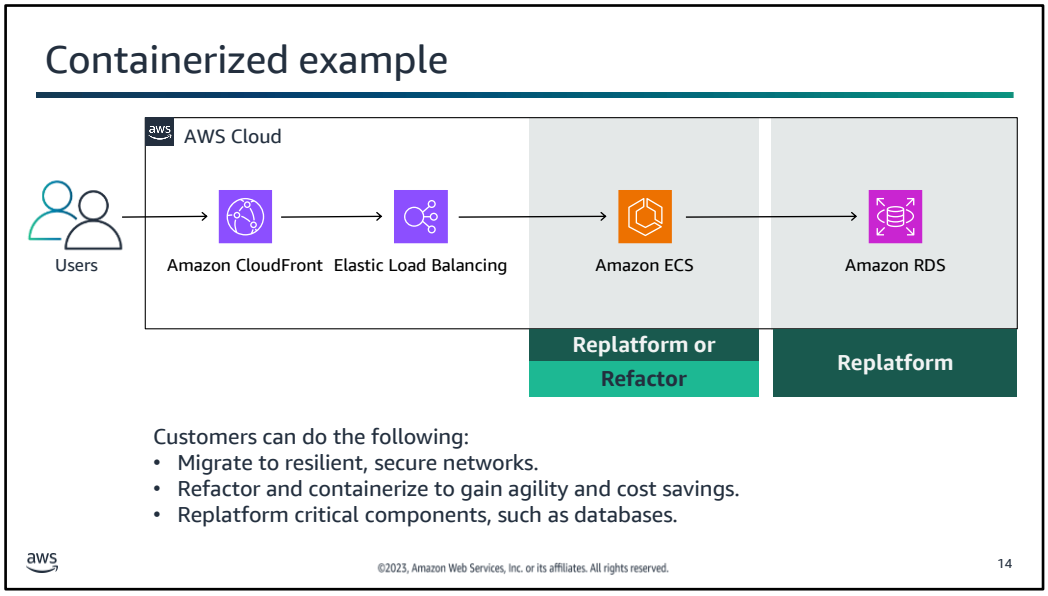
Businesses know that getting software products and services to market faster translates into increased market share. Using containers can help modernize legacy applications and create new cloud-focused applications. When used with DevOps and agile practices, DevOps teams who need to adapt to growing business needs use containers to deliver applications quickly.

Container frameworks, such as Docker, provide a standardized way to package applications such as code, runtime, and libraries. They also provide a way to run applications across the entire software development lifecycle, regardless of the environment.

Containers are less risky because they can be defined as code. Containers package code with the configuration files for building the container and the dependencies it needs to run in any environment. Containers also help streamline testing.

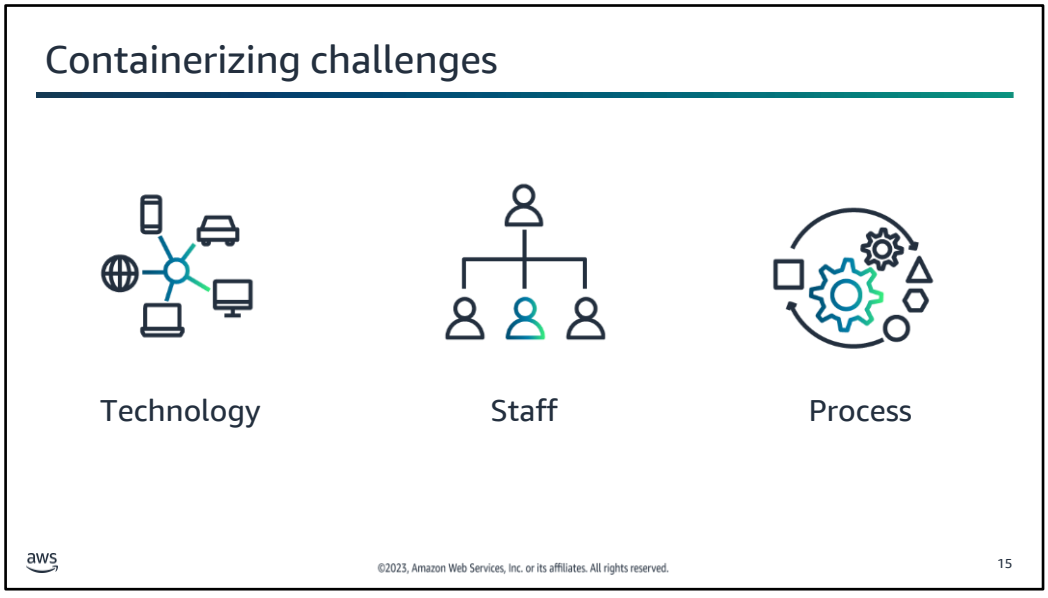
Containers improve operational efficiency by reducing IT operational overhead costs and achieving optimized compute infrastructure. For example, your customers can reduce costs by containerizing instead of a lift-and-shift to a VM. Containers might take up less space than

VMs because container images are typically tens of MBs. Containers can handle more applications and require fewer VMs and OSs. Therefore, your customers can fit more applications on each server or cluster.



In this example, an application that previously required compute services and dedicated servers now runs in a container managed by Amazon ECS. The data is accessible to the application because the database was replatformed to Amazon RDS. By using Amazon CloudFront and Elastic Load Balancing, the application benefits from a modernized infrastructure that makes it scalable and globally available.

By containerizing and deploying applications in the AWS Cloud, your customers can do the following:
- Migrate to resilient, secure networks
- Refactor and containerize to gain agility and cost savings
- Replatform critical components, such as databases, for more cost savings



Although containerizing an application has many benefits, your customers might face challenges when working with applications that were built before cloud technology. The main challenges of containerizing applications are technology, staff, and process related.

Technology – When your customers containerize applications, they face challenges with legacy versions of OSs, Java, Component Object Model (COM), COM+, and .NET. Plus, they must differentiate operating in containers compared to VMs. With legacy systems, customers often must overcome challenges related to the following:

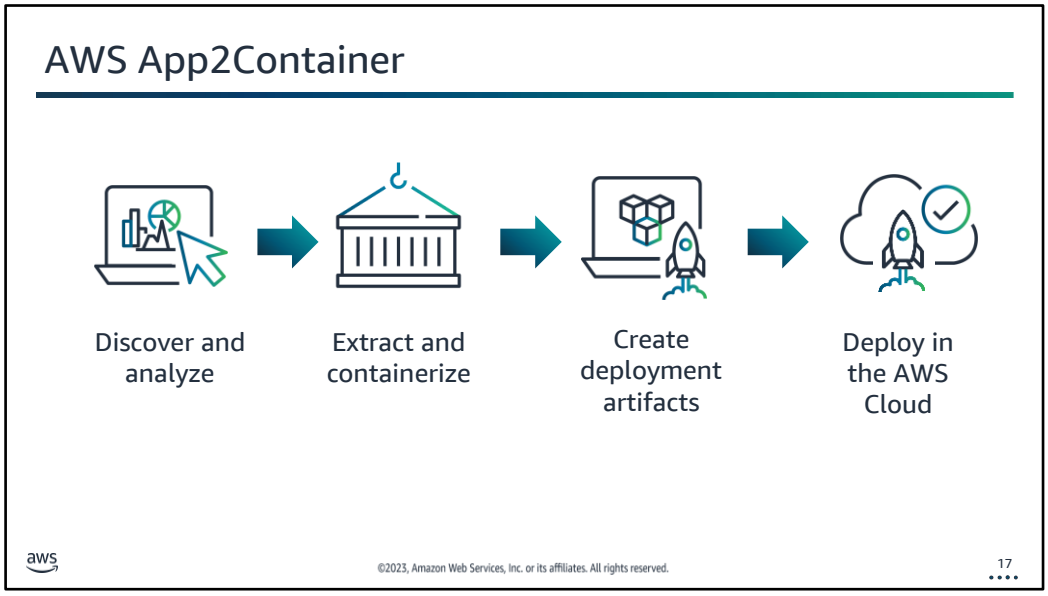
- Monolithic on-premises applications built with different underlying development frameworks and Windows OS versions that are tedious to refractor and containerize
- Applications based on legacy architecture, like Windows 2003 or 2008, that need to be migrated to modern, supported OSs
- Optimum use of Windows licensing and suggestions during the process that don't slow down the migration
- Migration of AIX applications to open systems without affecting development timelines

Staff – When your customers are containerizing applications, they face challenges with their staff for several reasons. One reason is an absence of developer tooling support for cloud migration and container migration. Another reason is lack of knowledge about legacy applications, containers and cloud technologies. In some cases, it may change the way they work altogether. For example, if a customer must apply a patch or a hotfix to an application, they usually log in to the VM and apply the patch or hotfix to the application directly. In containers, the customer must terminate the old container, build a new container image, and relaunch the application from scratch.

Process – Customers often replatform applications to open systems to reduce operating expenses. They face challenges with the complications of moving workloads to different platforms. When moving applications, you must consider the efforts to deploy and maintain the systems. Even script-based build and deploy methods can be time-consuming if the process requires repeated manual effort. For example, moving from Windows to Linux would require rewriting and recompiling the code. However, the same automated deployment, such as Puppet or Chef, can be retained to

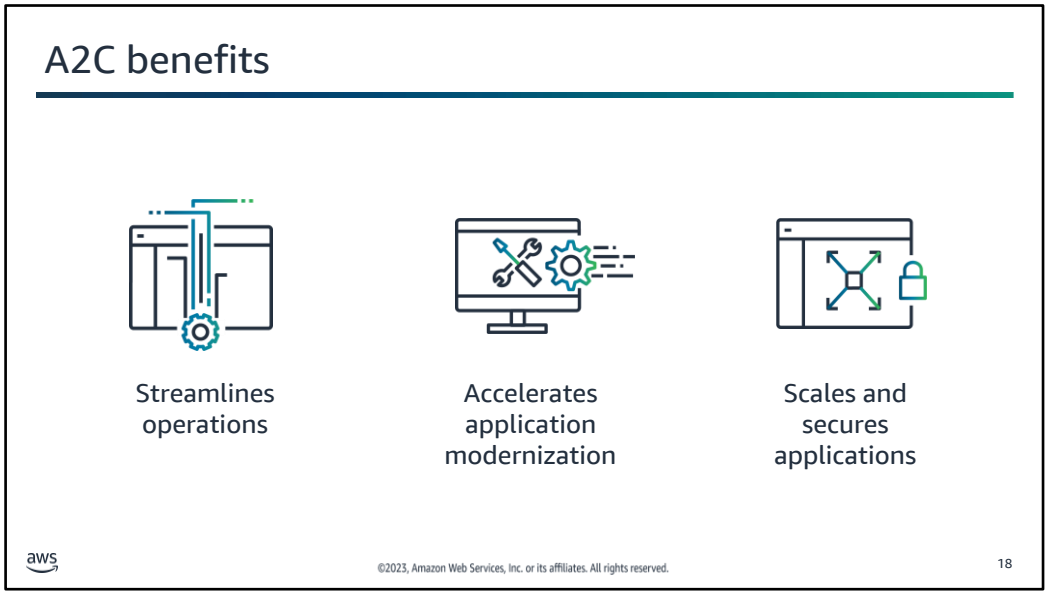
maintain most of the processes already in place.

When applications to be containerized are portable and ready, A2C can automate the process.



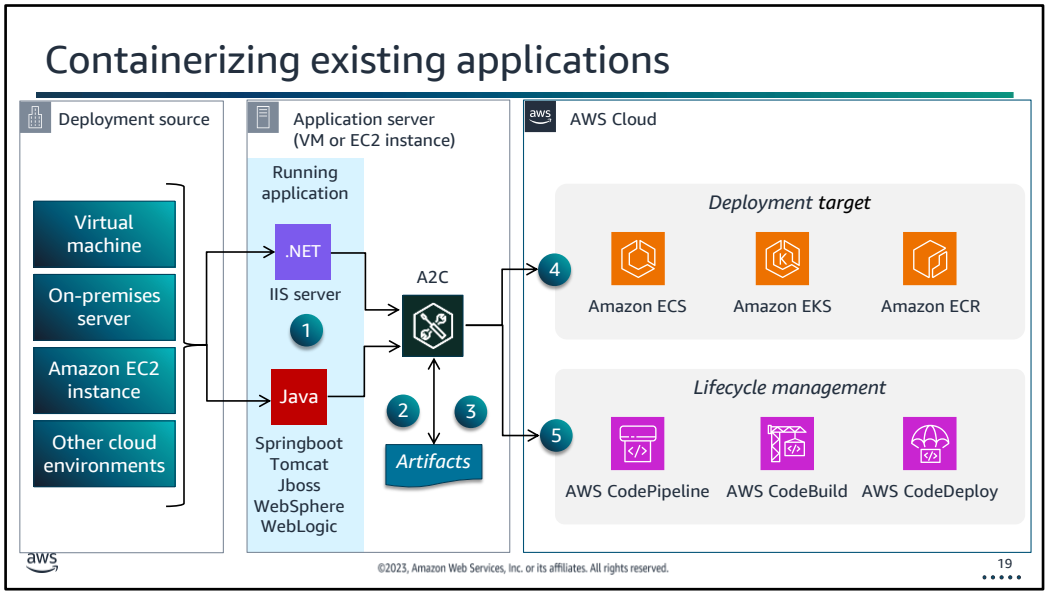
A2C helps customers lift and shift applications that run in an on-premises data center or on VMs. Applications run in containers managed by Amazon ECS or Amazon EKS.

With A2C, your customers can do the following:
- Discover and analyze candidate applications
- Extract and containerize, building an application container
- Create deployment artifacts to deploy the containers on Amazon ECS or Amazon EKS
- Deploy in the AWS Cloud



A2C provides the following three primary benefits:

- Streamlines operations – With A2C, customers can containerize existing applications. For monitoring, operations, and software delivery, A2C standardizes on a single set of tooling. By containerizing applications with A2C, customers can unify the infrastructure and skill sets needed to operate applications. This can save time in addition to infrastructure and training costs.
- Accelerates application modernization – A2C analyzes applications that are candidates for being containerized. It automatically generates a container image configured with the correct dependencies, network configurations, and deployment instructions for Amazon ECS or Amazon EKS. This process saves time and avoids manual settings and errors from negligence.
- Scales and secures applications – Customers can use A2C to deploy existing applications on the cloud that are provisioned with the correct networking and security configurations.

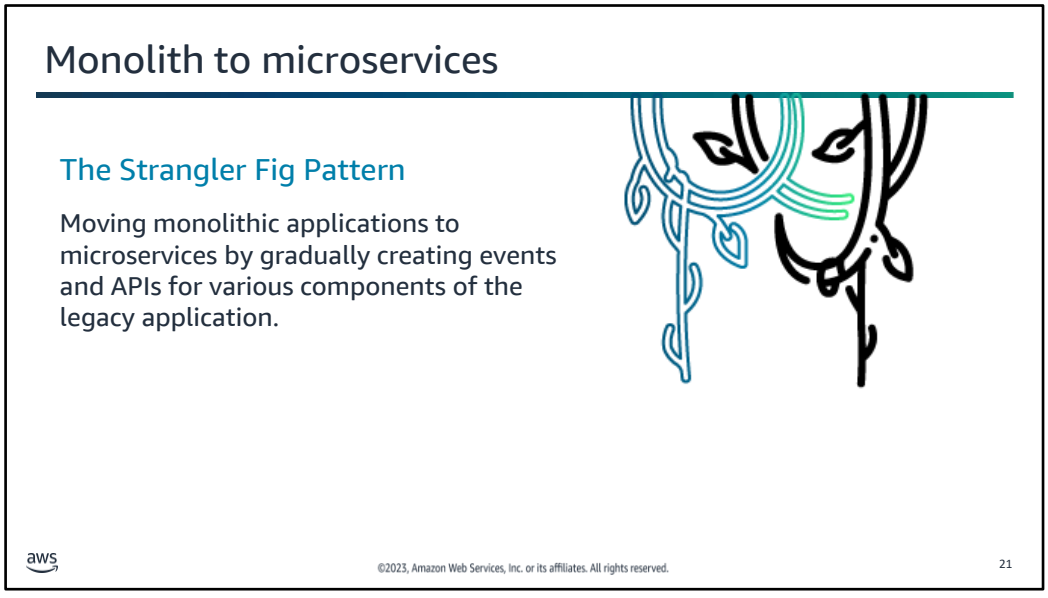


A2C is a command line tool that helps customers lift and shift applications that run in an on-premises data center or on VMs. Applications run in containers managed by Amazon ECS or Amazon EKS.

The AWS App2Container workflow is as follows:

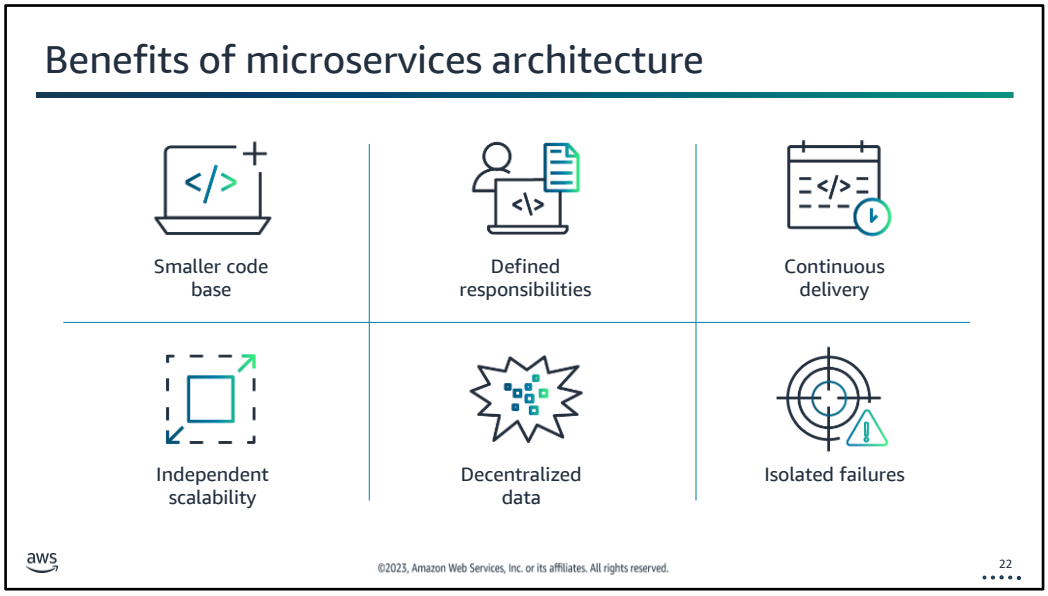
1. Discover and analyze – The first step identifies and lists candidate applications (.NET or Java) for the application server to containerize.
2. Extract and containerize – The next step extracts application artifacts for containerization and generates a Dockerfile. It then initiates builds for the application container.
3. Create deployment artifacts – Next, A2C generates AWS artifacts. These artifacts deploy the containers on either Amazon ECS or Amazon EKS.
4. Deploy in the AWS Cloud – The final step deploys the containers in AWS Cloud.
5. (Optional) Create a continuous integration and continuous delivery (CI/CD) pipeline with AWS CodePipeline – This optional step automates building and deploying the containers.

Refactor spaces is a way to move from monoliths to microservices. In this section, you learn how that works and the tools to help.



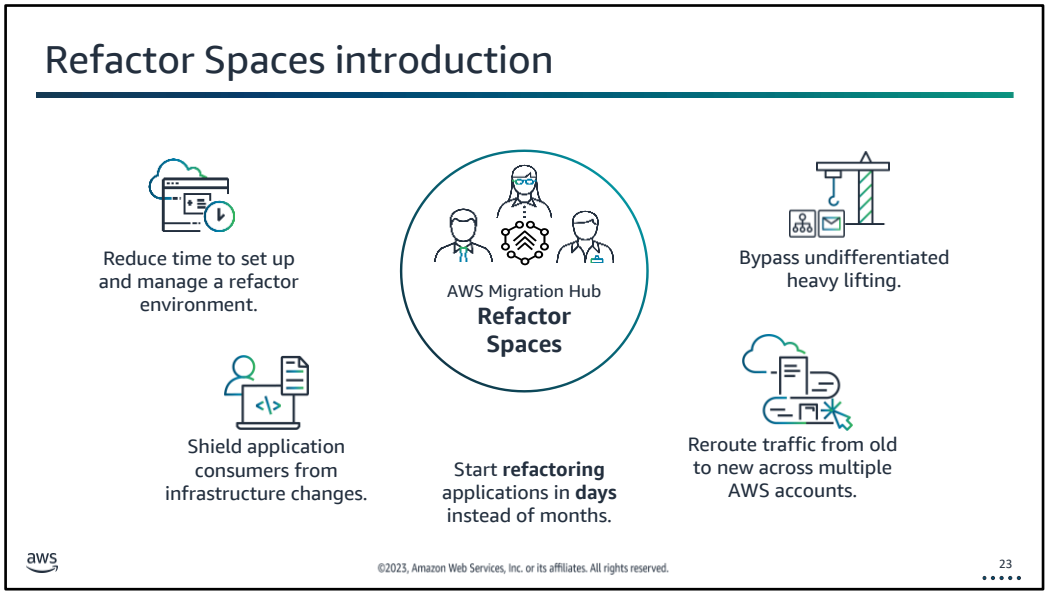
Let’s begin with a little backstory about the microservices architecture.

Incremental refactoring is also known as the “Strangler Fig Pattern”. Martin Fowler coined the term after he noticed how the vines of a strangler fig tree eventually take over the host tree. He saw this process as a metaphor for describing how you complete a rewrite of an important system. You incrementally rewrite the system until the monolithic application is gone, and you are left with a serverless microservices application.



The following are benefits of using a microservices architecture:

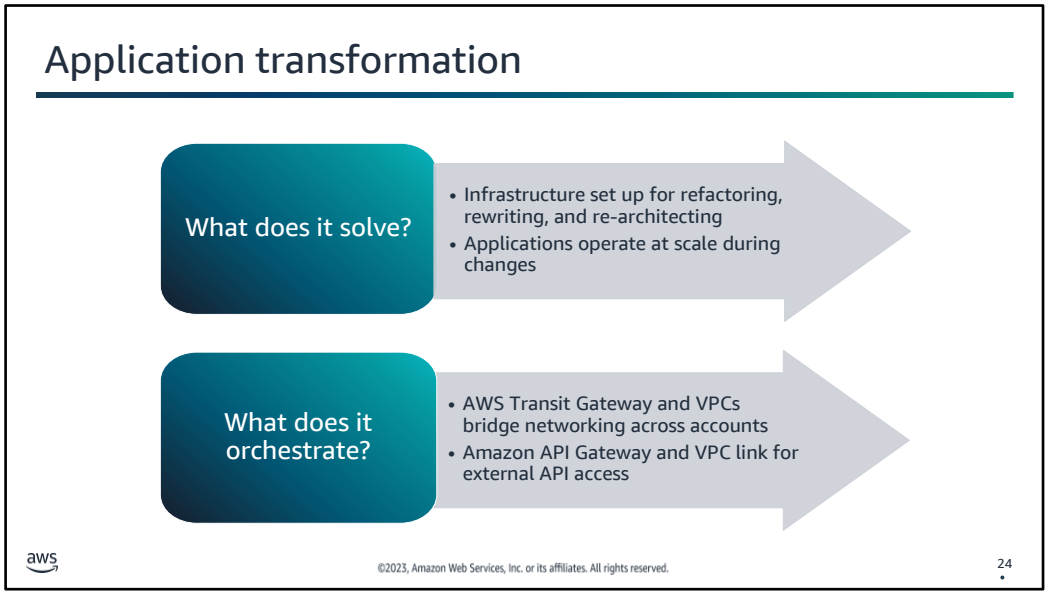
- Developers can work on a smaller code base compared to a larger one in a monolithic enterprise app.
- Developers have more defined responsibilities. The application's technology stack can differ through microservices.
- Continuous delivery is more streamlined. Developers don't need to redeploy the system for a small change.
- Scalability is independent to each microservices. Developers can choose to scale each component of their application depending on the required resource.
- Developers can decentralize data and use different databases and storage for the microservices.
- Developers can isolate failures so an error or a bug in one microservice does not bring the entire system down.



The following are four primary benefits of incrementally refactoring your applications with Refactor Spaces:

- Reduces the time to set up and manage a refactor environment for your application modernization activities
- Shields your application consumers as you update and change the infrastructure and microservices
- Seamlessly reroutes traffic from exiting applications, often monoliths, to new capabilities across multiple AWS accounts, such as serverless or containers
- Bypasses the undifferentiated heavy lifting as AWS makes the right refactor environment for you

These benefits can complete refactoring your applications in days instead of months.

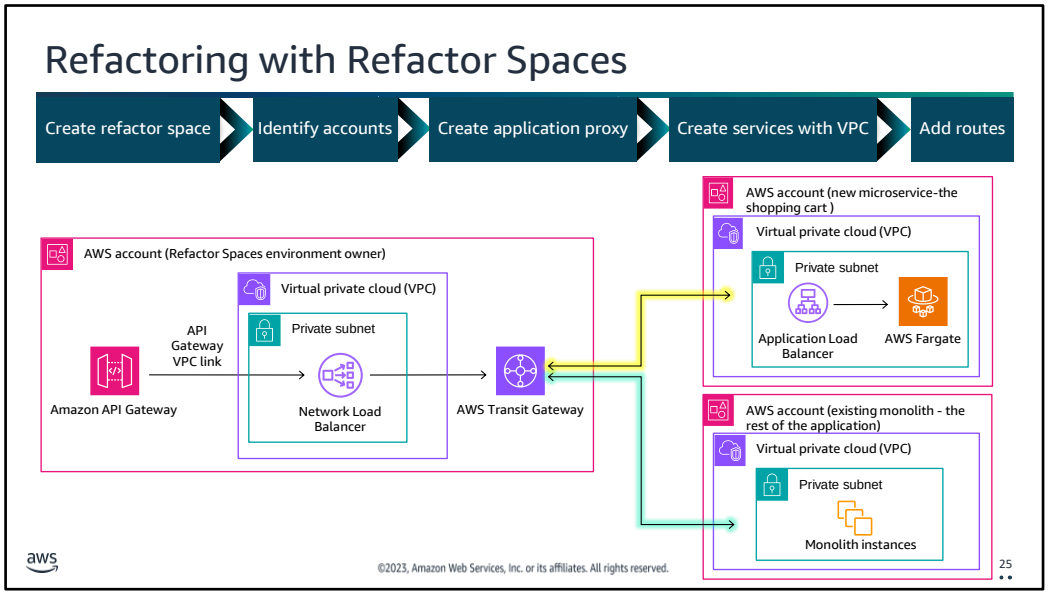


What does Refactor Spaces solve?

- It sets up an infrastructure for refactoring, rewriting, and re-architecting applications. Applications operate at scale while they are changing.

What does Refactor Spaces orchestrate?

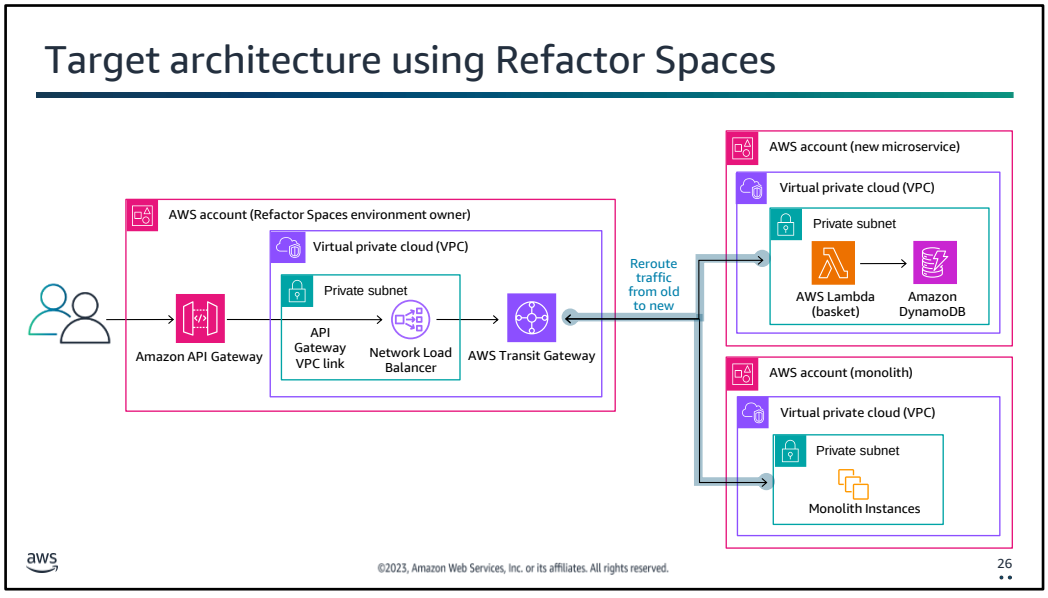
- It orchestrates AWS Transit Gateway and VPCs to bridge networking across accounts to simplify old and new communication services.
- It also orchestrates API Gateway and VPC link for external API access to transparently add new services and incrementally route traffic from old to new.



Let’s review a basic application refactor using Refactor Spaces:

1. Create the Refactor Spaces environment in an account to be used as the environment owner. Refactor Spaces configures Transit Gateway in this account. You create this environment inside your account. It is not owned by the Refactor Spaces service. You can alter it if needed.
2. Identify other accounts for refactoring to the Refactor Spaces environment. In this example, you have two accounts. One account contains an existing monolith already in AWS, and the other is for your new microservice. Using AWS Resource Access Manager (AWS RAM), Refactor Spaces has shared Transit Gateway with the other accounts. It is ready to bridge traffic between Refactor Spaces Services in any of these three accounts.
3. Create an Refactor Spaces application proxy in the Refactor Spaces environment owner’s account. Refactor Spaces configures API Gateway and Network Load Balancer to allow for external HTTP-based access to Refactor Spaces services in the Refactor Spaces environment.
4. Add a Refactor Spaces service with its VPC from an account. Refactor Spaces configures Transit Gateway to bridge the VPCs. Traffic can move between the service VPCs in each of the accounts.
5. Add a route for external access to your app. Add a default route to send all application traffic to the monolith. After you have identified your first business domain, build a new microservice with Application Load Balancer and Fargate. Refactor Spaces doesn’t care about compute target.
6. Route a specific set of paths to the new microservice. External application users won't know that traffic is now being handled by a new microservice.

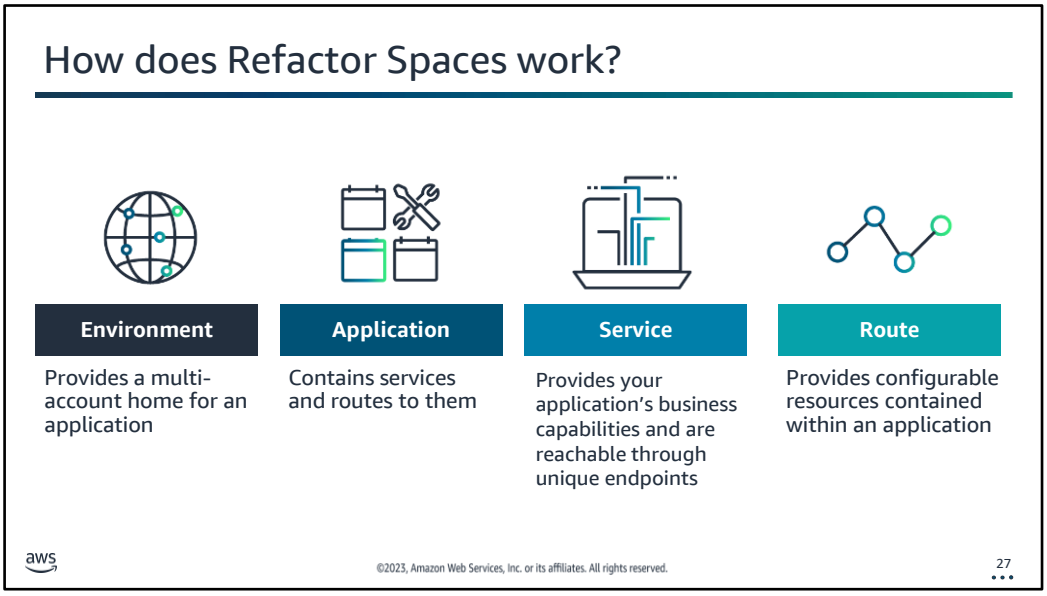
Because Refactor Spaces is orchestrating AWS services, it is not in your application’s critical path. Availability, latency, and limits will be the same as those of the API Gateway, Network Load Balancer, and Transit Gateway.



This architecture illustrates that you can decompose to AWS Lambda. You will practice this in the next lab.
Now you can see the basket feature from your application has been moved to the new microservice. Refactor Spaces is routing calls to the right AWS account.

The monolith application is still in place. It provides other functionality for the ecommerce application. Refactor Spaces is also routing certain calls to it.

You can strip the application's functionality piece by piece until you completely break the monolith into microservices.

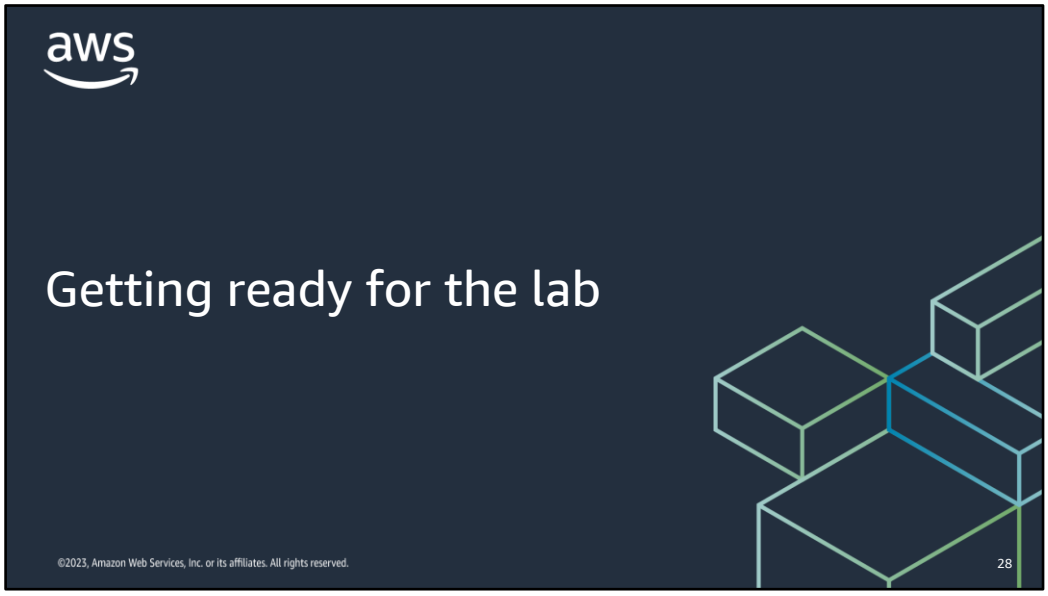


When you think about environment, Refactor Spaces provides a multi-account home for an application. This means it provides the following:
- Unified networking and services across accounts
- Account-level isolation of services to support development process and team alignment

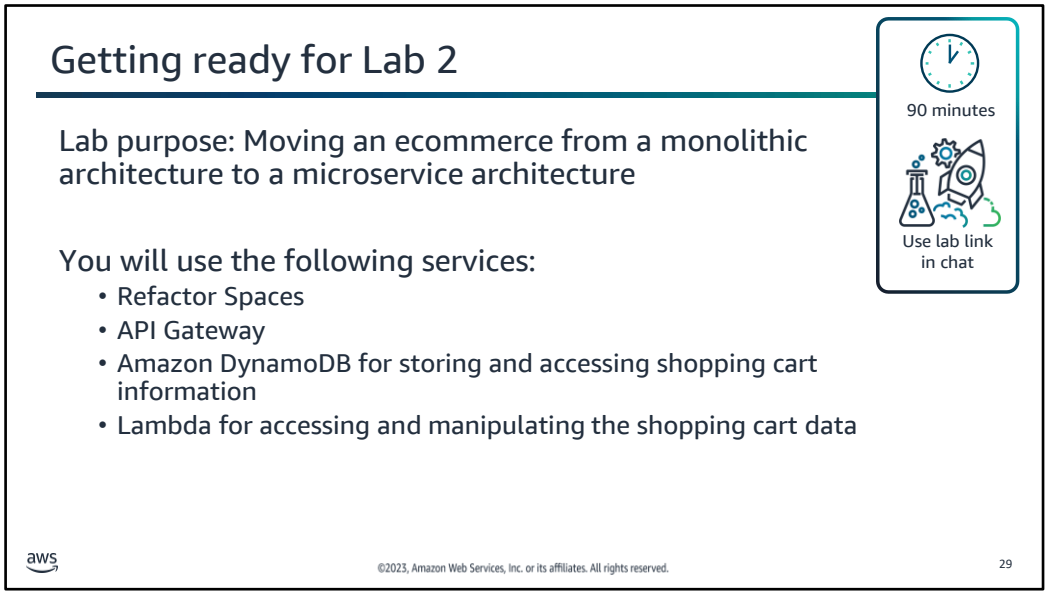
When you look at application, Refactor Spaces contains services and routes to them. This means it provides the following:
- Configurable request routing to legacy and refactored microservices
- Transparent underlying architecture changes to application consumers
- Proxy for the Strangler Fig pattern

For the service, Refactor Spaces provides business capabilities that are reachable through unique endpoints. It can do the following:
- Exist inside an environment and an application
- Communicate across accounts within an environment
- Accessible through configurable routes for inbound requests

And finally when it comes to route, Refactor Spaces provides configurable resources contained within an application. Therefore, it can do the following:
- Request routing from external HTTP gateway to targeted application services
- Route targets including protocol, DNS name, port, and URI path



Before starting Lab 2, let's review what it does at a high level and which services you will be using.



Although Refactor Spaces also supports containers, you will use API Gateway with Lambda as the computing layer during this lab.

Some of the following services might be new to you, here is a brief description of the services you will use in the lab:

- **Amazon API Gateway:** A fully managed service that creates, publishes, maintains, monitors, and secures APIs at any scale. APIs act like the front door for applications when accessing data, business logic, or functionality from your backend services. Using API Gateway, you can create RESTful APIs and WebSocket APIs that provide real-time two-way communication applications. API Gateway supports containerized and serverless workloads and web applications.
- **Amazon DynamoDB:** A fully managed, serverless, key-value NoSQL database designed to run high-performance applications at any scale. DynamoDB offers built-in security, continuous backups, automated multi-Region replication, in-memory caching, and data import and export tools.
- **AWS Lambda:** A serverless, event-driven compute service you can use to run code for virtually any type of application or backend service without provisioning or managing servers. You can activate Lambda from over 200 AWS services and software as a service (SaaS) applications and only pay for what you use.

Review

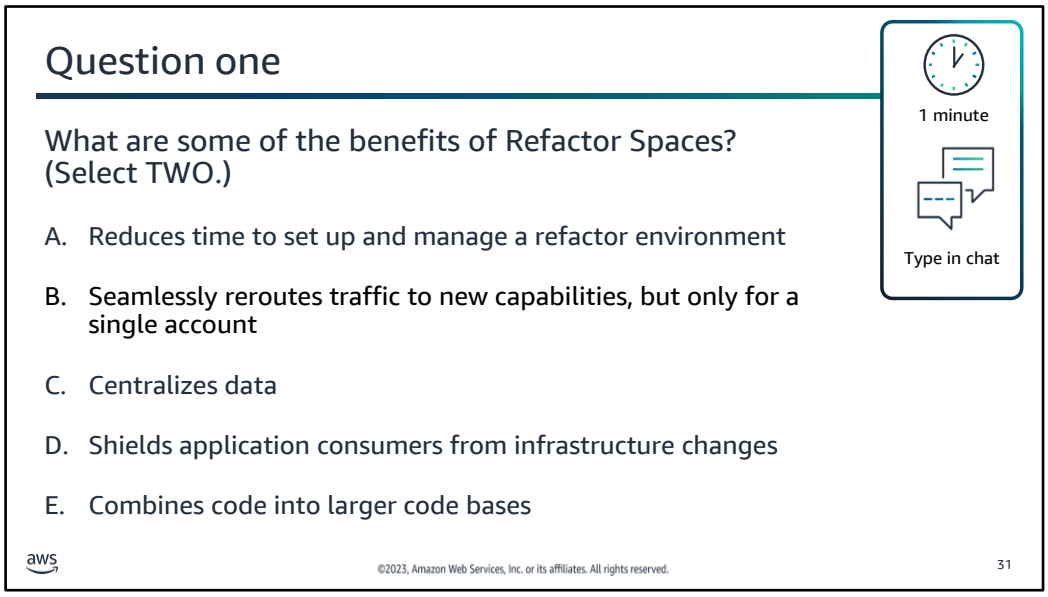
Question on Refactor Spaces

Question on moving away from commercial licensing

Question on containerization

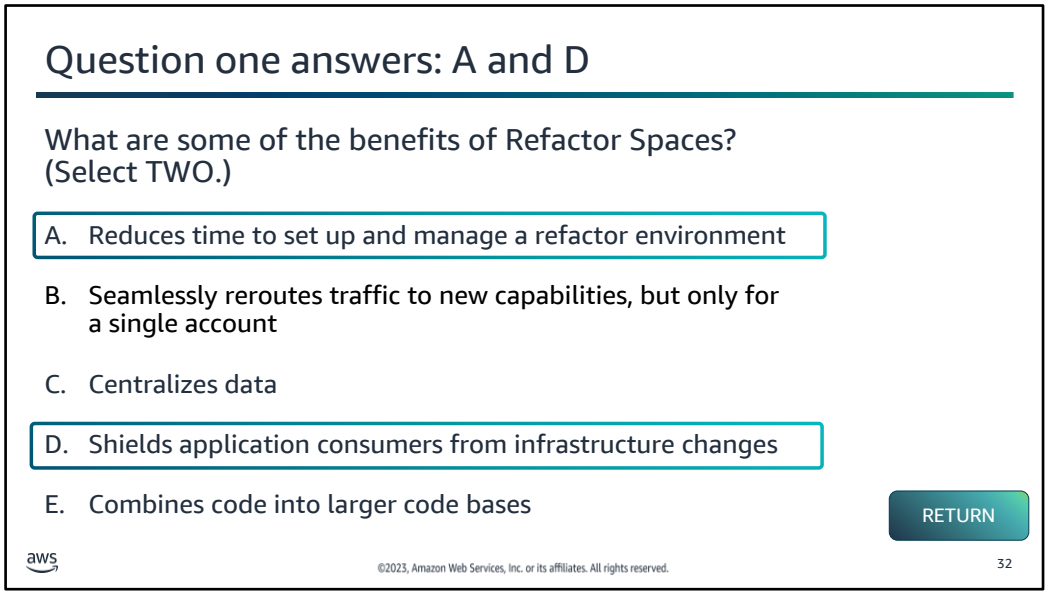
Module summary

30



What are some of the benefits of AWS Refactor Spaces? (Select TWO.)

A. Reduces time to set up and manage a refactor environment
B. Refactors applications in months instead of years
C. Centralizes data
D. Shields application consumers from infrastructure changes
E. Combines code into larger code bases



The correct answers are A) Reduces time to set up and manage a refactor environment and D) Shields application consumers from infrastructure changes.

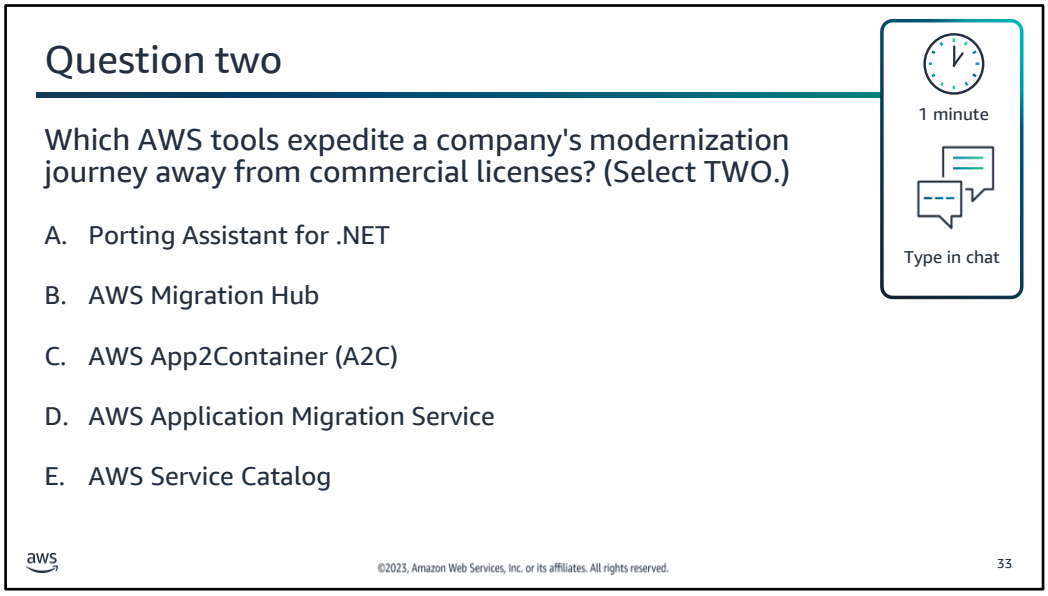
A. Reduces time to set up and manage a refactor environment – Is correct

B. Seamlessly reroutes traffic to new capabilities but only for a single account (Refactor can reroute across multiple accounts.)

C. Centralizes data Is not correct. (Refactor Spaces decentralizes data.)

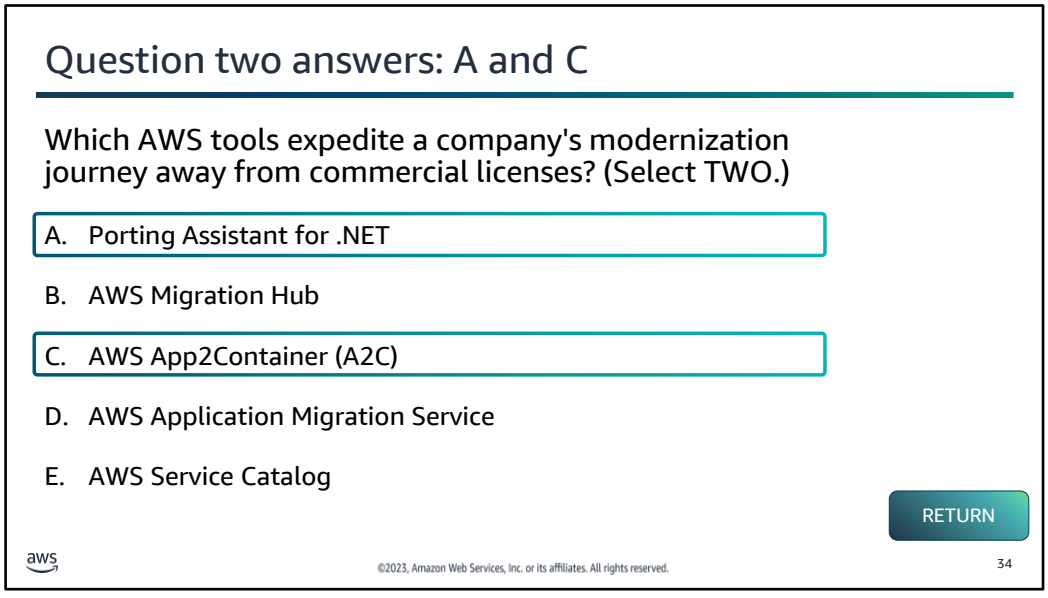
D. Shields application consumers from infrastructure changes – Is correct

E. Combines code into larger code bases- is not correct.-(Refactor Spaces creates smaller code bases.)



Which AWS tools expedite a company's modernization journey away from commercial licenses? (Select TWO.)

A. Porting Assistant for .NET
B. AWS Migration Hub
C. AWS App2Container (A2C)
D. AWS Application Migration Service
E. AWS Service Catalog

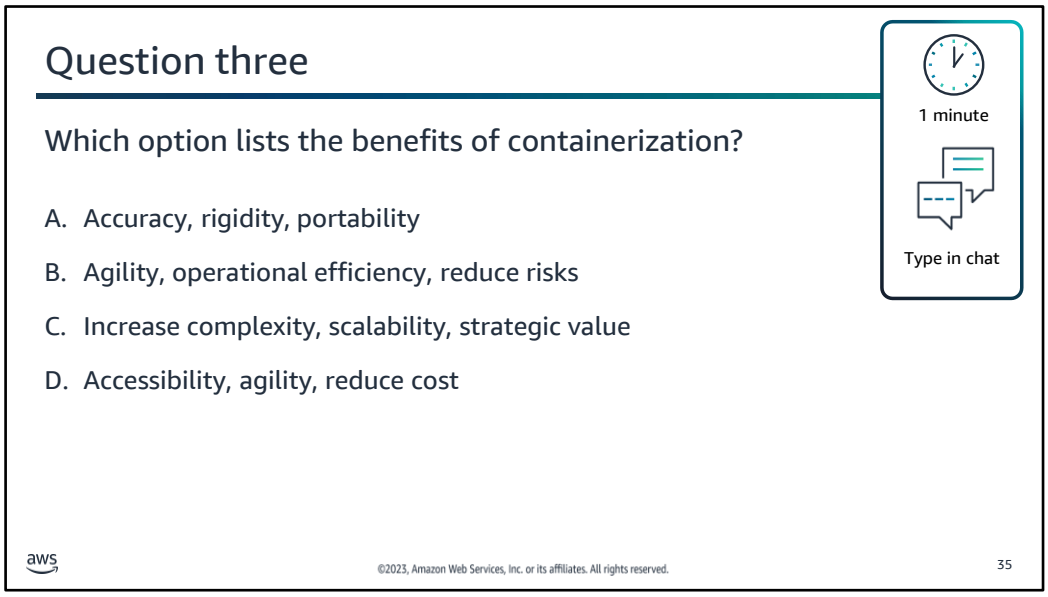


The correct answers are A) Porting Assistant for .NET and C) AWS App2Container.

A. **Porting Assistant for .NET – Is correct.**
B. AWS Migration Hub – is not correct.
C. **AWS App2Container (A2C) – Is correct.**
D. AWS Application Migration Service – is not correct.
E. AWS Service Catalog – is not correct.

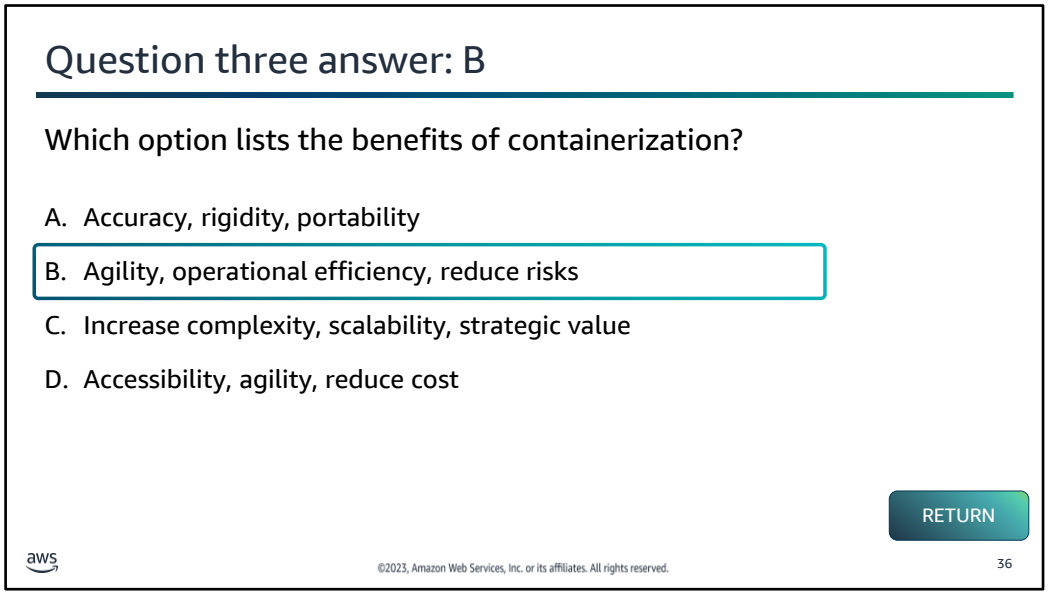
AWS offers the following tools to expedite your modernization journey away from commercial licenses and unlock innovation.

- Porting Assistant for .NET – helps refactor the old .NET implementation to the latest .NET Core.
- A2C – modernizes .NET and Java applications into containerized applications.
- AWS Microservice Extractor for .NET
- AWS Toolkit for .NET Refactoring



Which option lists the benefits of containerization?

A. Accuracy, rigidity, portability
B. Agility, portability, operational efficiency
C. Increase complexity, scalability, strategic value
D. Accessibility, agility, risk aversion



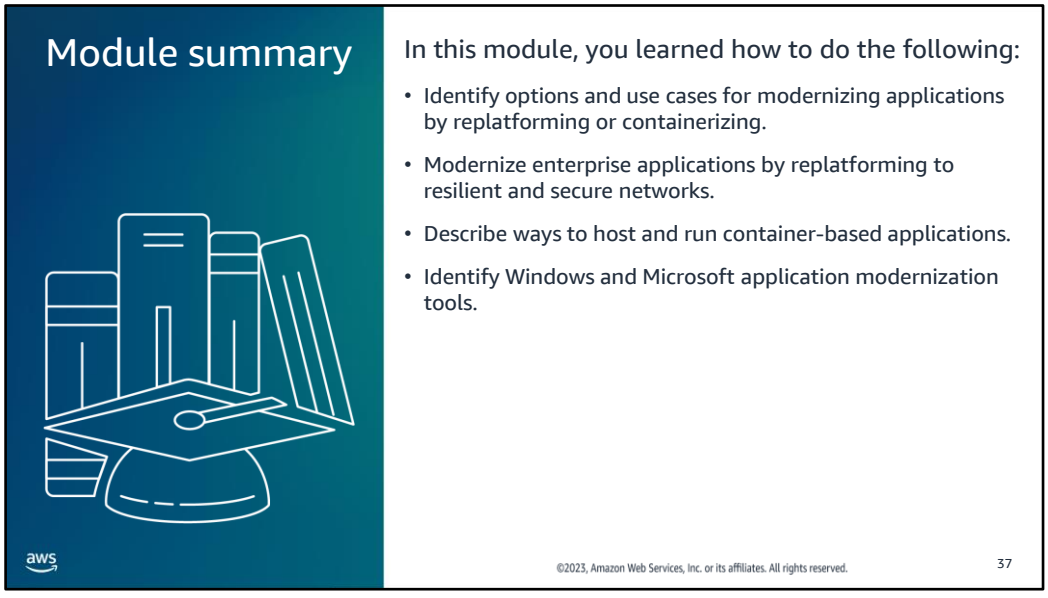
The correct answer is B) Agility, portability, operational efficiency

A. Accuracy, rigidity, portability - Is not correct. Containers are not more accurate than other development approaches. Containers are not rigid.

B. Agility, operational efficiency, reduce risks – Is correct.

C. Increase complexity, scalability, strategic value – Is not correct.

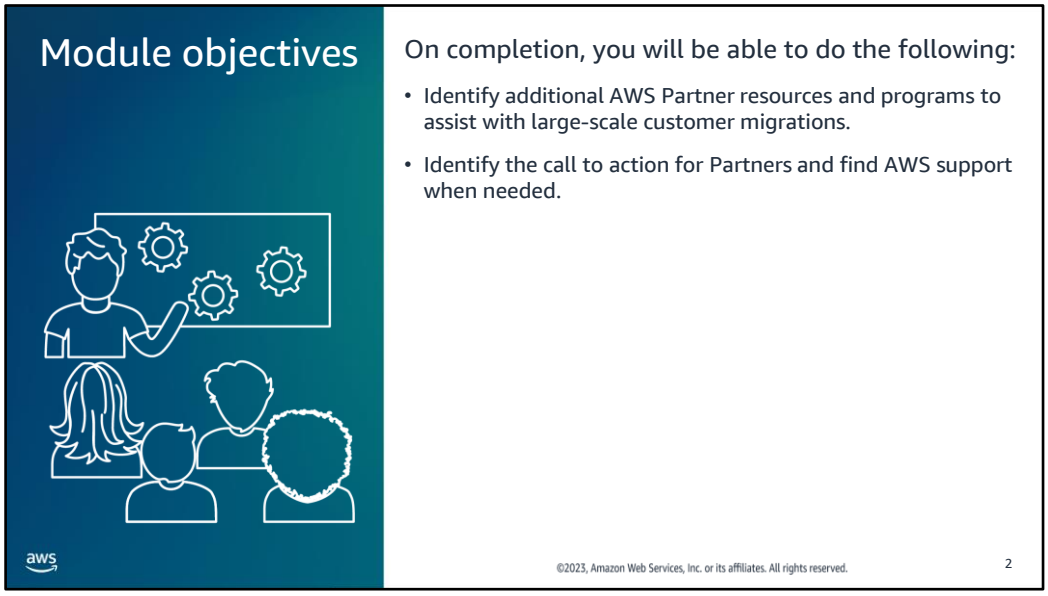
D. Accessibility, agility, reduce cost – Is not correct. Containers will not initially reduce costs.



In this module, you learned to do the following:

- Identify options and use cases for modernizing applications by replatforming or containerizing.
- Modernize enterprise applications by replatforming to resilient, secure networks.
- Describe ways to host and run container-based applications.
- Identify Windows and Microsoft application modernization tools.

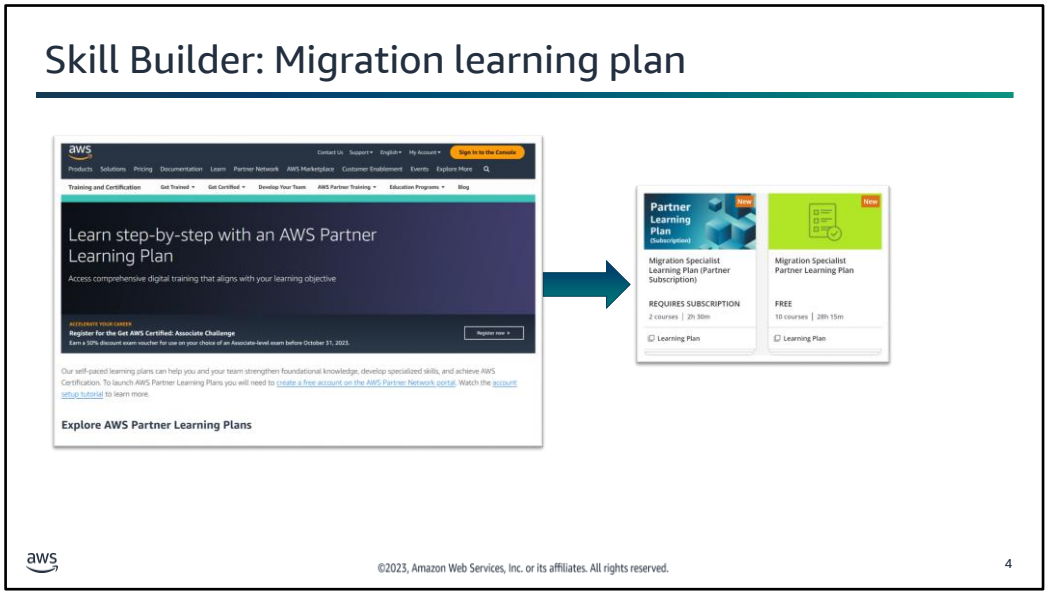
Welcome to Module 5: Migration Resources and Additional Training. This module contains resources and links to support you after this course.



On completion of this module, you will be able to do the following:

- Identify additional AWS Partner programs and resources to assist with large-scale customer migrations
- Identify next steps to engage in large-scale migrations and find AWS support when needed

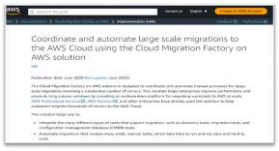
In this section, you will review AWS Partner training and documentation to support you after the course.



To build your AWS cloud skills, you can access Skill Builder, the learning platform for digital training.
There are AWS Partner Learning Plans by certification, job role, or technology. To learn more about migrations, check out the Migration Learning Plan. To launch AWS Partner Learning Plans, you will need to create a free account on the AWS Partner Network portal.

For more information about learning plans, see *Learn step-by-step with an AWS Partner Learning Plan* at https://aws.amazon.com/partners/training/aws-partner-learning-plans. In the technology section, you will see the Migration Learning Plan.

Documentation



Cloud Migration Factory on AWS (CMF)
- Implementation guide
- Plan your deployment

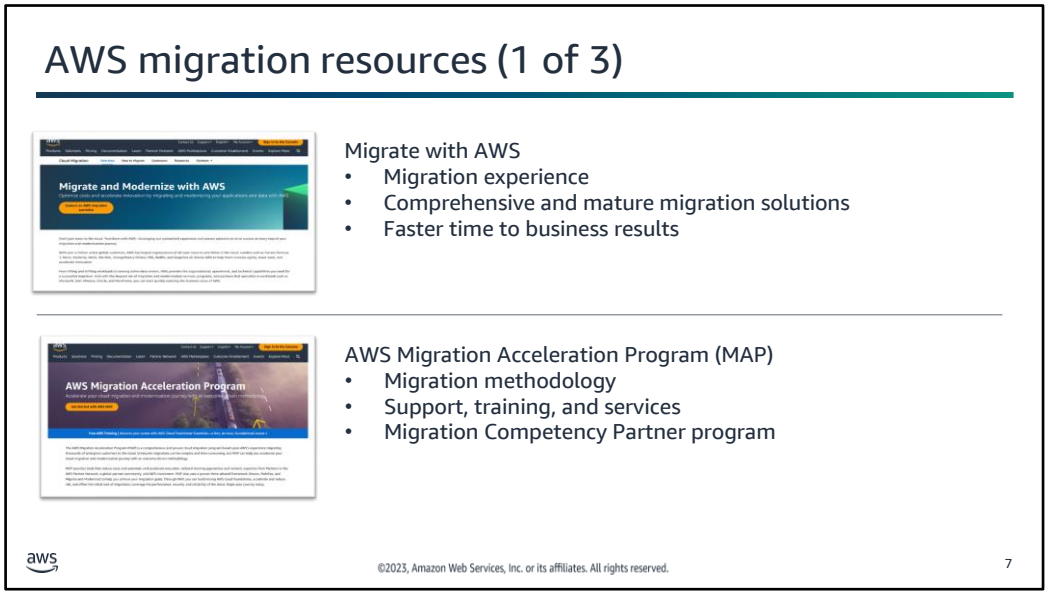
AWS App2Container (A2C)
- Getting started
- User guide
- Command reference

5

For more information about the Cloud Migration Factory on AWS, see the Implementation Guide at https://docs.aws.amazon.com/solutions/latest/cloud-migration-factory-on-aws/solution-overview.html.

For more information about A2C, see *Getting Started with AWS App2Container* at https://docs.aws.amazon.com/app2container/latest/UserGuide/start-intro.html.

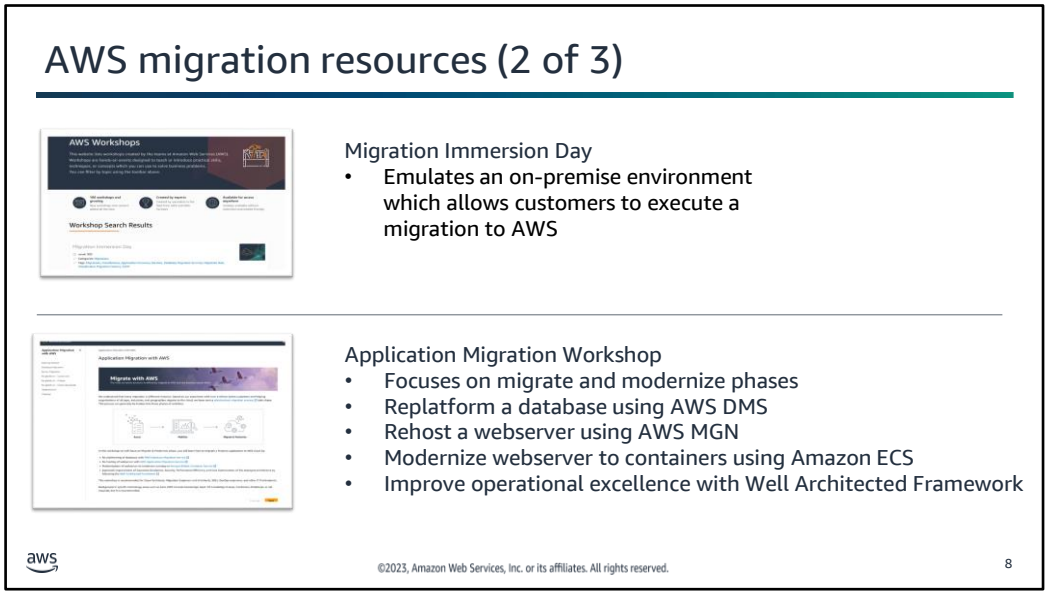
In this section, you will review AWS Partner resources to support you after the course.



AWS Migration resources

Migrating with AWS addresses people, process, technology, and financial considerations throughout the migration journey. The solutions help ensure that your project achieves the desired business outcomes. For more information about migration, see *Migrate and Modernize with AWS* at https://aws.amazon.com/cloud-migration.

AWS Migration Acceleration Program (MAP) helps reduce operating costs and improve agility, global scalability, and resiliency options for IT workloads when migrating to AWS. For more information about MAP, see https://aws.amazon.com/migration-acceleration-program.



AWS Migration Immersion Day

This workshop emulates an on-premise environment which allows customers to execute a migration to AWS. For more information, see https://www.workshops.aws/card/Migration%20Immersion%20Day.

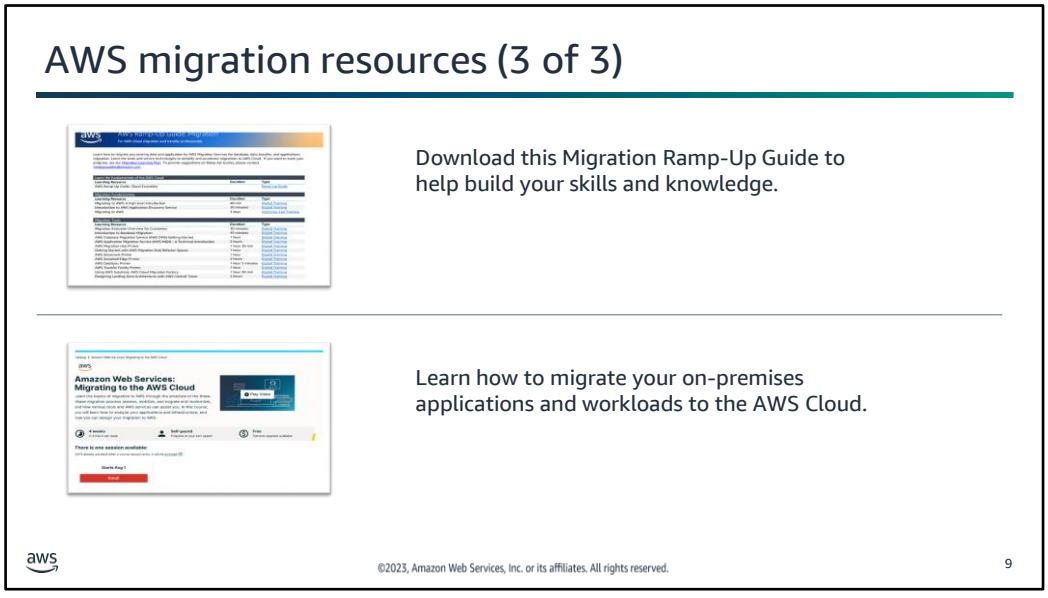
Note: the workshop will include the important instructions for cleaning up after the workshop to avoid charges in your AWS account.

Application Migration with AWS Workshop

Application Migration Workshop

- Focuses on migrate and modernize phases
- Replatform a database using AWS DMS
- Rehost a webserver using AWS MGN
- Modernize webserver to containers using Amazon ECS
- Improve operational excellence with Well Architected Framework

To learn more see, https://catalog.us-east-1.prod.workshops.aws/workshops/c6bdf8dc-d2b2-4dbd-b673-90836e954745/en-US



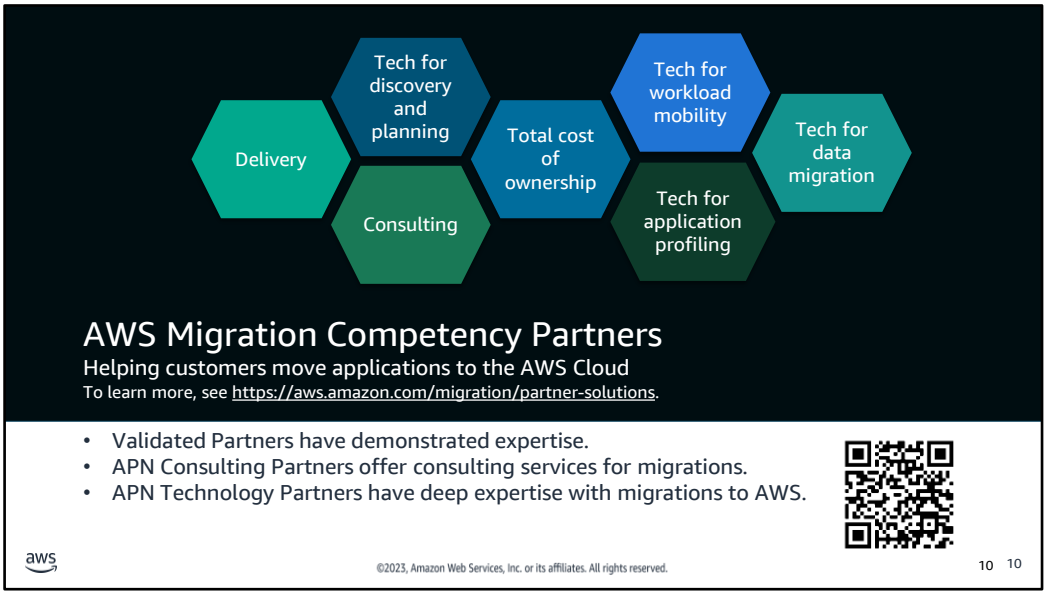
Migration Ramp-Up Guide

To build your skills and knowledge about migrating to AWS, see *AWS Ramp-Up Guide: Migration* at https://d1.awsstatic.com/training-and-certification/ramp-up_guides/Ramp-Up_Guide_Migration.pdf.

Migrating to the AWS Cloud

This 4-week course focuses on migrating workloads to AWS. It focuses on analyzing your current environment, planning your migration, and discussing the actual migration steps. To access this EdX training, see Amazon Web Services: Migrating to AWS at https://www.edx.org/course/migrating-to-the-aws-cloud

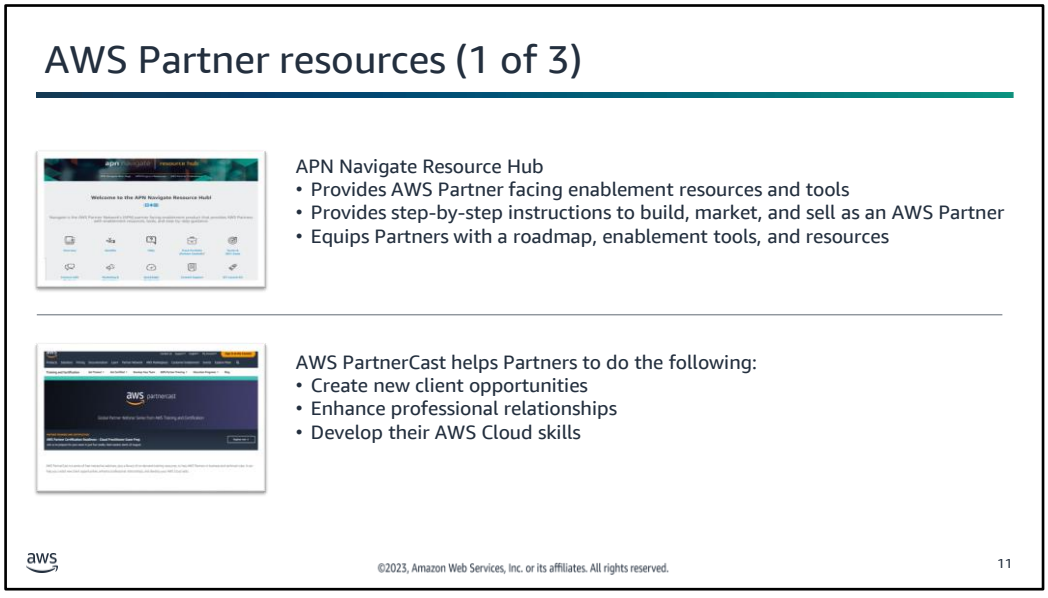
.



AWS Migration Competency Partners help customers move applications to the AWS Cloud. Competency Partners have the following expertise:

- Validated Partners have demonstrated expertise.
- APN Consulting Partners offer consulting services for migrations.
- APN Technology Partners have deep expertise with migrations to AWS.

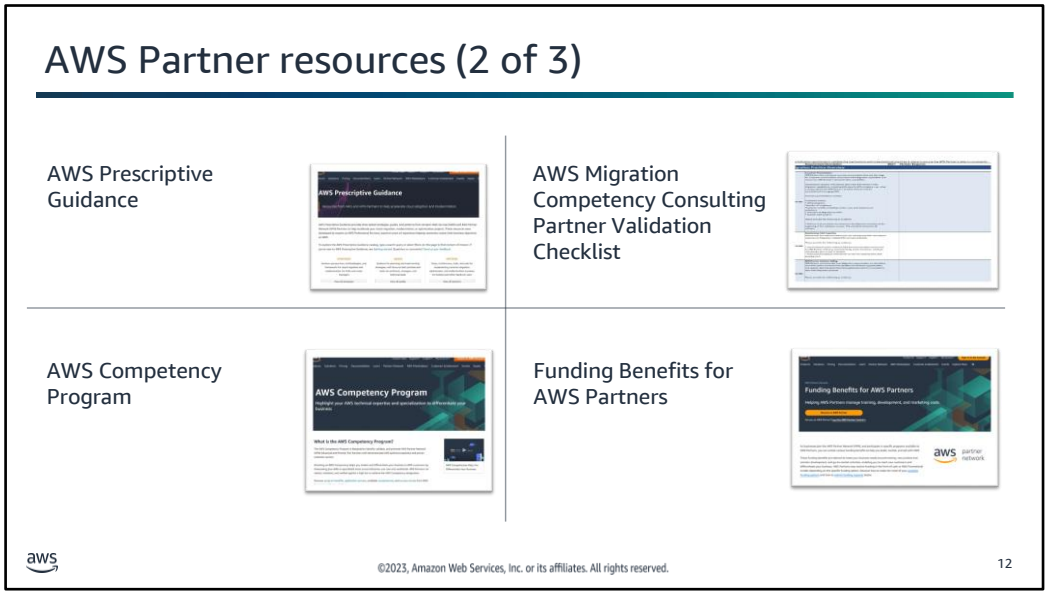
To learn more about the AWS Migration Competency Partners, see https://aws.amazon.com/migration/partner-solutions.



The AWS Partner Network is a global community that helps transform your business on AWS. It provides access to business and technical benefits and enablement content from trusted experts. You can increase visibility with AWS and build connections with key AWS experts by sharing your organization's progress. You develop core go-to-market assets to build your portfolio, highlight your expertise on AWS, and earn trust with customers.

APN Navigate is the AWS Partner facing enablement product that provides AWS Partners with enablement resources, tools, and step-by-step guidance. Navigate provides prescriptive guidance from trusted AWS experts on how to transform your business on AWS. The step-by-step instruction helps you build, market, and sell as an AWS Partner. All AWS Partners can participate in the APN Navigate program to accelerate business growth, regardless of where you are in your Partner journey. APN Navigate equips Partners with a roadmap, enablement tools, and resources to help you achieve your business goals. To learn more, see the wiki at https://w.amazon.com/bin/view/APN_Navigate_Hub. You can also see *APN Navigate* website at https://aws.amazon.com/partners/navigate.

AWS PartnerCast is a series of free interactive webinars and on-demand training resources to help APN Partners in business and technical roles. It is designed to help you create new client opportunities, enhance professional relationships, and develop your AWS Cloud skills. For more information, see https://aws.amazon.com/partners/training/partnercast.

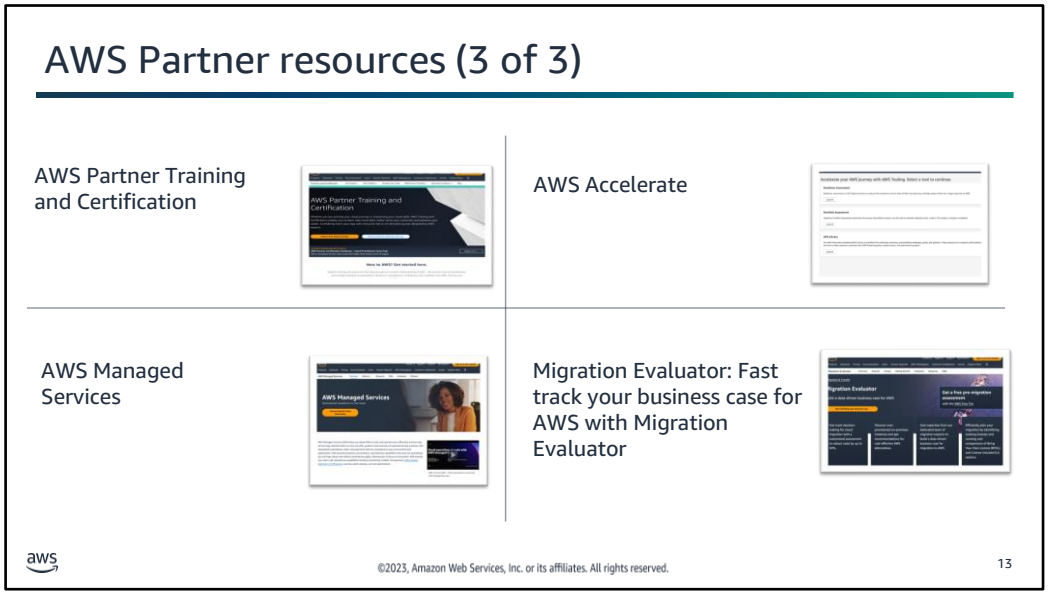


AWS Prescriptive Guidance provides time-tested strategies, guides, and patterns from AWS Partners to help accelerate cloud migration, modernization, or optimization projects. These resources were developed by experts at AWS Professional Services. They are based on years of experience helping customers realize their business objectives on AWS. To learn more, see https://accelerate.amazonaws.com and use AWS Partner Central credentials to log in.

The AWS Competency Program is designed to identify, validate, and promote AWS Advanced and Premier Tier Partners with demonstrated AWS technical expertise and proven customer success. The program helps you market and differentiate your business to AWS customers by showcasing your skills in specialized areas across industries, use cases, and workloads. To learn more, see *AWS Competency Program* at https://aws.amazon.com/partners/programs/competencies.

The AWS Competency Partner Validation Checklist is intended for AWS Partners who are interested in applying for an AWS Competency. The Checklist provides the criteria necessary for you to achieve the designation under the AWS Competency Program. The checklist is available through AWS Partner Central using your login. To access, see https://partnercentral.awspartner.com/ContentFolderPartner?id=0690h000003pc7y

The Partner Opportunity Acceleration (POA) Funding is designed to accelerate sales cycles and customer adoption of your solution or products powered by AWS. It helps you develop wins that can validate and demonstrate your AWS expertise and earn the trust of your customers. To learn more, see *Funding Benefits for AWS Partners* at https://aws.amazon.com/partners/funding.



AWS Partner Training and Certification provides information you can use to support your customers' business and technical needs. We offer both digital and classroom training. To learn about best practices, you can choose to learn online at your own pace or from an AWS instructor. To learn more, see https://aws.amazon.com/partners/training.

The AWS Managed Services helps your customers adopt AWS at scale and learn about managed services. To learn more, see https://aws.amazon.com/managed-services.

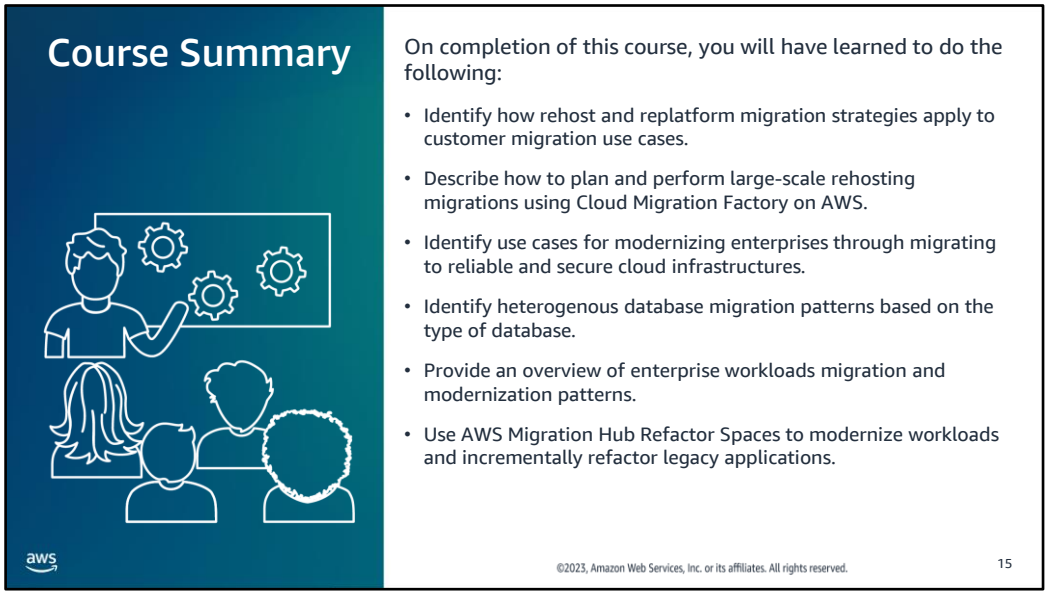
AWS Accelerate provides the following:
- A readiness assessment to evaluate the current state of your customer's cloud journey
- A portfolio assessment to automate the process of portfolio analysis
- AWS Prescriptive Guidance (APG) Library resources to help accelerate migration, modernization, and optimization projects

To learn more, see https://accelerate.amazonaws.com.

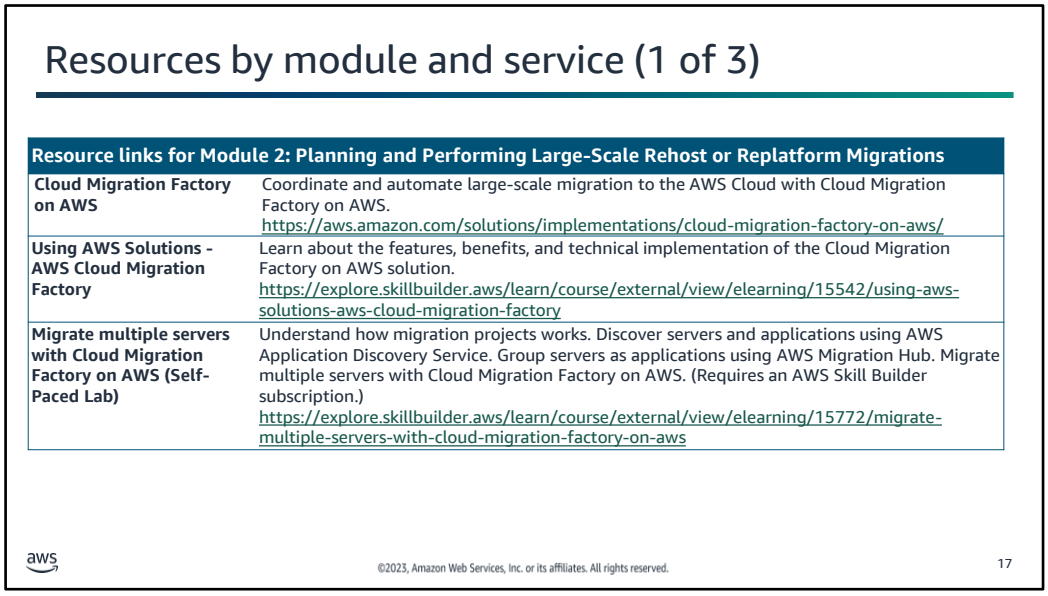
Watch a 28-minute video about Migration Evaluator. This video demonstrates how running a migration assessment with Migration Evaluator, formerly TSO Logic, can fast track your business case. To learn more, see https://aws.amazon.com/migration-evaluator.

As you complete this course there are a few calls to action to help support you as you continue to grow your migration skills. Your call to action includes the following:

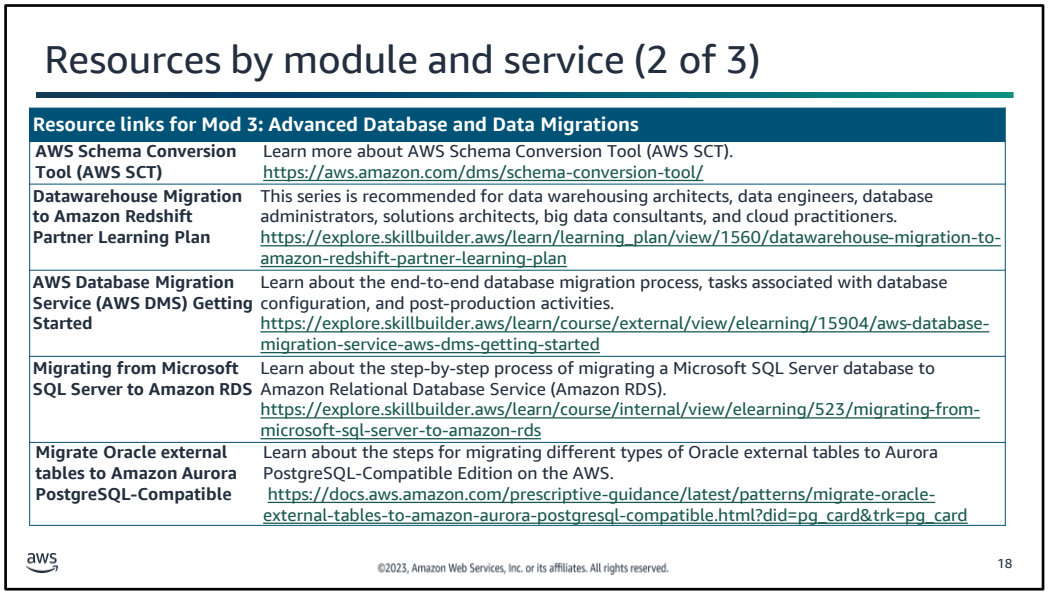
- Engage with your AWS Partner managers to accelerate your ramp-up to AWS.
- Improve your skills with additional training.
- Learn about the available AWS Partner programs that support you.



On completion of this course, here is a list of the migration topics and tasks you have explored. To support you after the session, you will receive a copy of the presentation with notes that you can revisit. We appreciate your feedback on the course survey.



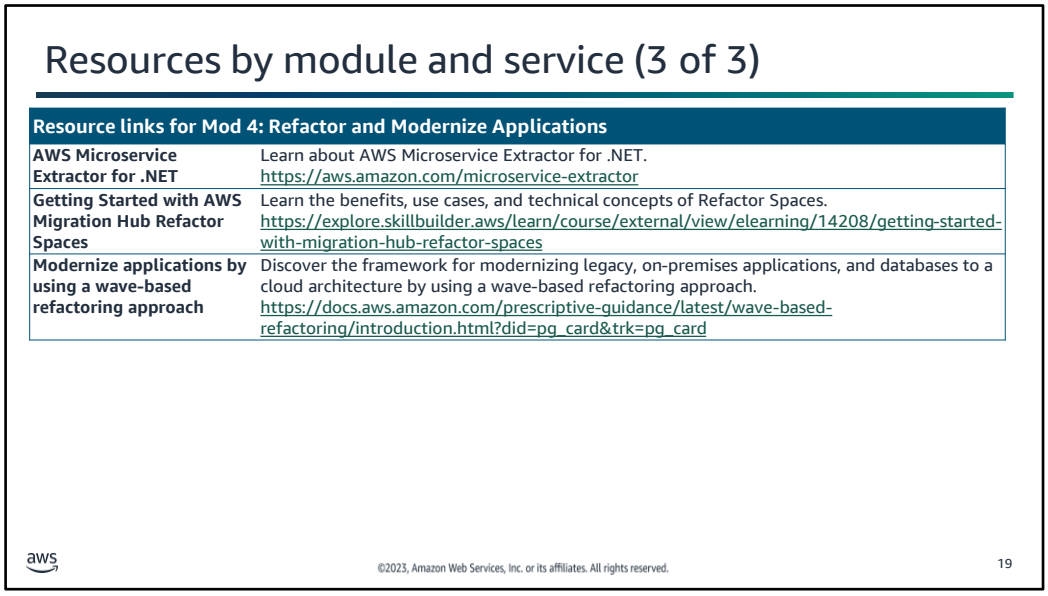
In this section, you will review additional migration resources to support you after the course.



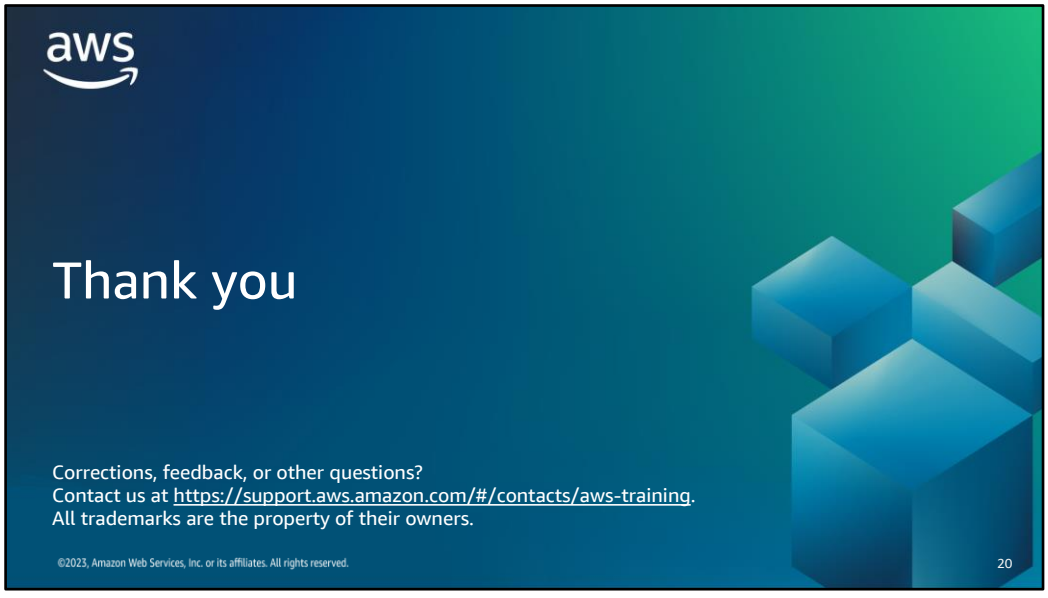
Resources by module and service (2 of 3)

Resource links for Mod 3: Advanced Database and Data Migrations	
AWS Schema Conversion Tool (AWS SCT)	Learn more about AWS Schema Conversion Tool (AWS SCT). https://aws.amazon.com/dms/schema-conversion-tool/
Datawarehouse Migration to Amazon Redshift Partner Learning Plan	This series is recommended for data warehousing architects, data engineers, database administrators, solutions architects, big data consultants, and cloud practitioners. https://explore.skillbuilder.aws/learn/learning_plan/view/1560/datawarehouse-migration-to-amazon-redshift-partner-learning-plan
AWS Database Migration Service (AWS DMS) Getting Started	Learn about the end-to-end database migration process, tasks associated with database configuration, and post-production activities. https://explore.skillbuilder.aws/learn/course/external/view/elearning/15904/aws-database-migration-service-aws-dms-getting-started
Migrating from Microsoft SQL Server to Amazon RDS	Learn about the step-by-step process of migrating a Microsoft SQL Server database to Amazon Relational Database Service (Amazon RDS). https://explore.skillbuilder.aws/learn/course/internal/view/elearning/523/migrating-from-microsoft-sql-server-to-amazon-rds
Migrate Oracle external tables to Amazon Aurora PostgreSQL-Compatible	Learn about the steps for migrating different types of Oracle external tables to Aurora PostgreSQL-Compatible Edition on the AWS. https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/migrate-oracle-external-tables-to-amazon-aurora-postgresql-compatible.html?did=pg_card&trk=pg_card

In this section, you will review additional migration resources to support you after the course.



In this section, you will review additional migration resources to support you after the course.



Thank you for attending the AWS Partner: Advanced Migration course.

