AWS Prescriptive Guidance
Migrating Oracle databases
to the AWS Cloud
# Table of Contents

Introduction .................................................................................................................................................. 1

Overview .................................................................................................................................................. 1

Oracle database migration strategies ........................................................................................................ 3

- Choosing the right migration strategy ................................................................................................. 3
- Online and offline migration .................................................................................................................. 4

Homogeneous database migration .......................................................................................................... 6

- Amazon RDS for Oracle ........................................................................................................................ 6
  - When to choose Amazon RDS .............................................................................................................. 7
  - High availability ................................................................................................................................... 7
  - Read replicas ....................................................................................................................................... 8
- Amazon RDS Custom for Oracle .......................................................................................................... 10
  - When to choose Amazon RDS for Oracle .......................................................................................... 10
  - How it works ..................................................................................................................................... 10
- Amazon EC2 for Oracle ....................................................................................................................... 12
  - When to choose Amazon EC2 ........................................................................................................... 12
  - High availability .................................................................................................................................. 12
- VMware Cloud on AWS for Oracle .................................................................................................... 14
  - When to choose VMware Cloud on AWS .......................................................................................... 14

Tools ......................................................................................................................................................... 14

- Oracle SQL Developer ......................................................................................................................... 16
- Oracle SQL *Loader ............................................................................................................................... 16
- Oracle Export and Import ..................................................................................................................... 16
- AWS DMS ............................................................................................................................................ 16
- Oracle Data Pump ............................................................................................................................... 17
- Oracle GoldenGate ............................................................................................................................. 18
- Oracle Data Guard ............................................................................................................................... 19
- Oracle RMAN ........................................................................................................................................ 19
- CloudEndure Migration ......................................................................................................................... 20
- VMware HCX ......................................................................................................................................... 20

Licensing options .................................................................................................................................... 21

- License Included ................................................................................................................................... 21
- BYOL ................................................................................................................................................... 21

Heterogeneous database migration ......................................................................................................... 23

Tools ......................................................................................................................................................... 24

- AWS SCT .............................................................................................................................................. 24
- AWS DMS ............................................................................................................................................. 25

Best practices for migrating to Amazon RDS for Oracle ........................................................................ 26

- Provisioning your target database ......................................................................................................... 26
- Exporting data from your source database .......................................................................................... 26
- Transferring data dump files to AWS .................................................................................................... 27
- Importing data to your target database .................................................................................................. 27
- Post-import steps ................................................................................................................................... 27
- Testing the migration ............................................................................................................................. 27
- Operating and optimizing your Amazon RDS database ....................................................................... 28

Oracle database migration patterns ....................................................................................................... 29

Partners .................................................................................................................................................... 30

Additional resources .................................................................................................................................. 31

Appendix: Oracle migration questionnaire ............................................................................................... 32

- General information ............................................................................................................................... 32
- Infrastructure .......................................................................................................................................... 32
- Database backups .................................................................................................................................... 33
- Database security .................................................................................................................................... 33
- Database high availability and disaster recovery ................................................................................. 33

AWS Prescriptive Guidance glossary ...................................................................................................... 34
Migrating Oracle databases to the AWS Cloud

Sagar Patel, Senior Database Specialty Architect, AWS Professional Services

June 2022 (p. 42)

Amazon Web Services (AWS) provides a comprehensive set of services and tools for deploying Oracle Database on the reliable and secure AWS Cloud infrastructure. This guide explains the options available for migrating your Oracle on-premises databases to the AWS Cloud. It also dives into the best practices and scenarios for exercising these migration options.

This guide is for program or project managers, product owners, database administrators, database engineers, and operations or infrastructure managers who are planning to migrate their on-premises Oracle databases to AWS.

Overview

Before you migrate your Oracle databases to AWS, you should understand and evaluate your migration strategy by using the framework discussed in Migration strategy for relational databases.

The first step is to perform an analysis of your application and Oracle Database workloads to understand the complexity, compatibility, and cost of migration. Here are some of the top points you should consider when you plan to migrate:

- Check the database current size and overall capacity growth. For example, if you’re planning to migrate your Oracle database to Amazon Relational Database Service (Amazon RDS) or Amazon RDS Custom, you can create DB instances with up to 64 TiB of storage. For the latest information, see Amazon RDS DB Instance Storage in the Amazon RDS documentation.
- Review Oracle Automatic Workload Repository (AWR) reports to check the resource usage and database health of your on-premises database.
- Check for application dependencies. If your database supports legacy, custom, or packaged applications, Amazon RDS Custom for Oracle might be a good choice. This service lets you retain control over database configurations, shared file systems, and operating system patches.
- Check current database dependencies on other databases. If your database is dependent on other databases, you can either migrate them together or create dependencies after you migrate your main database.
- Determine the IOPS and throughput of your databases. If you’re planning to migrate to Amazon RDS, consider the I/O performance of Amazon RDS DB instances.
- Review your current architecture and auditing or compliance needs, to make sure you can satisfy these requirements after moving to either Amazon RDS or Amazon Elastic Compute Cloud (Amazon EC2).
- Check the version and edition of your Oracle Database software to make sure it’s supported if you’re planning to move to Amazon RDS for Oracle (see currently supported versions for Amazon RDS and Amazon RDS Custom).
- Check the network connectivity between your on-premises environment and AWS, to make sure that it provides enough bandwidth for fast transfers of data between on premises and AWS.
- Determine the amount of downtime you have available for migration so you can plan your migration approach and decide whether you want to use online or offline migration.
• Identify your recovery time objective (RTO), recovery point objective (RPO), and service-level agreement (SLA) requirements for your existing database workloads.

• Check the chipset endian platform of the database workload. AWS supports x86-x64 little-endian platforms. Other platforms, such as Sun SPARC, HP Tru64, or IBM zSeries-based big-endian platforms, require cross-platform migration.

• AWS supports Linux (32-bit and 64-bit) and Windows operating systems. It doesn’t support Solaris, HP-UX, or IBM AIX operating systems, which are commonly used for Oracle databases. Migrating Oracle databases from these operating systems requires platform conversion.
Oracle database migration strategies

At a high level, there are two options for migrating an Oracle database from on premises to the AWS Cloud: either stay on Oracle (homogenous migration) or move off Oracle (heterogenous migration). In a homogenous migration, you don’t change the database engine (that is, your target database is also an Oracle database). In a heterogenous migration, you switch either to an open-source database engine such as MySQL, PostgreSQL, or MariaDB, or to an AWS Cloud-native database such as Amazon Aurora, Amazon DynamoDB, or Amazon Redshift.

There are three common strategies for migrating your Oracle databases to AWS: rehost, replatform, and re-architect (refactor). These are part of the 7 Rs of application migration strategies and described in the following table.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Type</th>
<th>When to choose</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rehost</td>
<td>Homogeneous</td>
<td>You want to migrate your Oracle database as is, with or without changing the operating system, database software, or configuration.</td>
<td>Oracle Database to Amazon EC2 (Browse Rehost patterns)</td>
</tr>
<tr>
<td>Replatform</td>
<td>Homogeneous</td>
<td>You want to reduce the time you spend managing database instances by using a database-as-a-service (DBaaS) offering.</td>
<td>Oracle Database to Amazon RDS for Oracle (Browse Replatform patterns)</td>
</tr>
<tr>
<td>Re-architect (refactor)</td>
<td>Heterogeneous</td>
<td>You want to restructure, rewrite, and re-architect your database and application to take advantage of open-source and cloud-native database features.</td>
<td>Oracle Database to Aurora PostgreSQL-Compatible, Aurora MySQL-Compatible, or MariaDB (Browse Re-architect patterns)</td>
</tr>
</tbody>
</table>

Choosing the right migration strategy

Choosing the correct strategy depends on your business requirements, your resource constraints, your migration timeframe, and cost considerations. The following diagram shows the effort and complexity involved in migrations, including all seven strategies.
Refactoring your Oracle database and migrating to an open-source or AWS Cloud-native database such as Amazon Aurora PostgreSQL-Compatible Edition or Amazon Aurora MySQL-Compatible Edition can help you modernize and optimize your database. By moving to an open-source database, you can avoid expensive licenses (resulting in lower costs), vendor lock-in periods, and audits, and you won’t have to pay additional fees for new features. However, depending on the complexity of your workload, refactoring your Oracle database can be a complicated, time-consuming, and resource-intensive effort.

To reduce complexity, instead of migrating your database in a single step, you might consider a phased approach. In the first phase, you can focus on core database functionality. In the next phase, you can integrate additional AWS services into your cloud environment, to reduce costs, and to optimize performance, productivity, and compliance. For example, if your goal is to replace your on-premises Oracle database with Aurora PostgreSQL-Compatible, you might consider rehosting your database on Amazon EC2 or replatforming your database on Amazon RDS for Oracle in the first phase, and then refactor to Aurora PostgreSQL-Compatible in a subsequent phase. This approach helps reduce costs, resources, and risks during the migration phase and focuses on optimization and modernization in the second phase.

### Online and offline migration

You can use two methods to migrate Oracle Database from an on-premises environment to the AWS Cloud, based on your migration timeline and how much downtime you can allow: online migration or offline migration.

- **Offline migration**: This method is used when your application can afford a planned downtime. In offline migration, the source database is offline during the migration period. While the source database is offline, it is migrated over to the target database on AWS. After the migration is complete, validation and verification checks are performed to ensure data consistency with the source database. When the database passes all validation checks, you perform a cutover to AWS by connecting your application to the target database on AWS.

- **Online migration**: This method is used when your application requires near zero to minimal downtime. In online migration, the source database is migrated in multiple steps to AWS. In the initial steps, the data in the source database is copied to the target database while the source database is still running. In subsequent steps, all changes from the source database are propagated to the target database. When the source and target databases are in sync, they are ready for cutover.
the cutover, the application switches its connections over to the target database on AWS, leaving no connections to the source database. You can use AWS Database Migration Service (AWS DMS), Oracle GoldenGate, Quest SharePlex, or tools available from AWS Marketplace (such as Attunity) to synchronize the source and target databases.
Homogeneous database migration for Oracle databases

AWS offers you the ability to run Oracle Database in a cloud environment. For developers and database administrators, running Oracle Database in the AWS Cloud is very similar to running Oracle Database in a data center. This section describes options for migrating Oracle Database from an on-premises environment or a data center to the AWS Cloud.

AWS offers four options for running Oracle Database on AWS, as described in the following table.

<table>
<thead>
<tr>
<th>Option</th>
<th>Highlights</th>
<th>More information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle Database on Amazon RDS</td>
<td>Managed service, provides easy provisioning and licensing</td>
<td>Amazon RDS for Oracle (p. 6) section</td>
</tr>
<tr>
<td>Oracle Database on Amazon RDS Custom</td>
<td>Managed service, but provides administrative rights to the database and the underlying operating system</td>
<td>Amazon RDS Custom for Oracle (p. 10) section</td>
</tr>
<tr>
<td>Oracle Database on Amazon EC2</td>
<td>Self-managed, provides full control and flexibility</td>
<td>Amazon EC2 for Oracle (p. 12) section</td>
</tr>
<tr>
<td>Oracle Database on VMware Cloud on AWS</td>
<td>Minimal disruption, easy to manage</td>
<td>VMware Cloud on AWS for Oracle (p. 14) section</td>
</tr>
</tbody>
</table>

Your application requirements, database features, functionality, growth capacity, and overall architecture complexity will determine which option to choose. If you are migrating multiple Oracle databases to AWS, some of them might be a great fit for Amazon RDS whereas others might be better suited to run directly on Amazon EC2. You might have databases that are running on Oracle Enterprise Edition (EE) but are a good fit for Oracle Standard Edition One (SE1) or Standard Edition Two (SE2). You can save on cost and licenses for those databases. Many AWS customers run multiple Oracle Database workloads across Amazon RDS, Amazon EC2, and VMware Cloud on AWS. If you’re moving to Amazon RDS Custom, make sure to review the requirements and limitations for Amazon RDS Custom for Oracle.

Amazon RDS for Oracle

Amazon RDS for Oracle is a managed database service that simplifies the provisioning and management of Oracle Database on AWS. Amazon RDS makes it easy to set up, operate, and scale Oracle Database deployments in the cloud. You can deploy your database in minutes and choose either General Purpose (SSD) storage or Provisioned IOPS storage. (For details, see Amazon RDS Storage Types in the AWS documentation.)

Amazon RDS frees you up to focus on application development, because it manages time-consuming database administration tasks, including provisioning, backups, software patching, monitoring, and
When to choose Amazon RDS

Amazon RDS for Oracle is a good migration option when:

• You want to focus on your business and applications, and you want AWS to take care of undifferentiated heavy lifting tasks such as the provisioning of the database, management of backup and recovery tasks, management of security patches, minor Oracle version upgrades, and storage management.

• You need a highly available database solution, and you want to take advantage of the push-button, synchronous Multi-AZ replication offered by Amazon RDS, without having to manually set up and maintain a standby database.

• You want to have synchronous replication to a standby instance, to provide high availability for your Oracle Database Standard Edition One (SE1) or Standard Edition Two (SE2) database, instead of having to pay for Oracle Database Enterprise Edition (EE).

• You want to pay for the Oracle license as part of the instance cost on an hourly basis instead of making a large, upfront investment.

• Your database size and IOPS needs are supported by Amazon RDS for Oracle. See Amazon RDS DB Instance Storage in the AWS documentation for the current maximum limits.

• You don’t want to manage backups or point-in-time recoveries of your database.

• You would rather focus on high-level tasks, such as performance tuning and schema optimization, instead of the daily administration of the database.

• You want to scale the instance type up or down based on your workload patterns without being concerned about licensing complexities.

After assessing your database and project requirements, if you decide to migrate to Amazon RDS for Oracle, see the details provided in the following sections, and review the migration best practices (p. 26) we discuss later in this guide.

High availability

Amazon RDS provides high availability and failover support for databases that are deployed with the Multi-AZ option. When you provision your database with the Multi-AZ option, Amazon RDS automatically provisions and maintains a synchronous standby replica in a different Availability Zone. The primary database synchronously replicates the data to the standby replica across Availability Zones. In case of infrastructure failure or Availability Zone disruption, Amazon RDS performs an automatic failover to the standby replica so you can resume database operations as soon as the failover is complete. This provides high redundancy, durability, and enhanced availability of your primary database. It also offloads your primary database by taking automated backups from the standby replica. For more information, see High Availability (Multi-AZ) for Amazon RDS in the AWS documentation.

The following diagram illustrates the Amazon RDS for Oracle Multi-AZ deployment option. The database application and users connect to the primary Oracle database, and all changes are synchronously replicated to the secondary database, which is in a different Availability Zone. The secondary database is not available to users until the failover is complete. After failover, the endpoint remains the same, so users and database applications can resume database operations without any manual intervention.
Read replicas

A read replica is a special type of Amazon RDS for Oracle DB instance that helps reduce the load on your primary DB instance. Updates made to your primary DB instance are asynchronously copied to the read replica, which you can set up in the same AWS Region or in another AWS Region.

You can provision an Amazon RDS for Oracle database with read replicas by using Oracle Active Data Guard to offload your read-only workload from the primary Oracle database. Oracle Active Data Guard replicates database changes from the source DB instance to the read replicas. This feature supports managed disaster recovery for mission-critical databases by allowing a read replica in another AWS Region to be promoted as a new, standalone, production database. You can provision up to five read replicas for your Amazon RDS for Oracle database.

Amazon RDS for Oracle makes it easy to create the read replicas by managing the configuration of Active Data Guard and maintaining secure network connections between a primary DB instance and its read replicas. For more information, see Working with Oracle read replicas for Amazon RDS in the Amazon RDS documentation.

To use the read replica feature, you must use the Bring Your Own License (BYOL) model with Oracle Database Enterprise Edition (EE) and also have an Active Data Guard license.

Using a read replica in the same AWS Region

The following diagram illustrates an Amazon RDS for Oracle DB instance in a Multi-AZ environment with a read replica in another Availability Zone in the same AWS Region. Not all AWS Regions offer more than two Availability Zones, so you should check the Region you’re planning to use before adopting this strategy.
Using a read replica in another AWS Region

Amazon RDS for Oracle also supports cross-Region read replicas. It uses Oracle Active Data Guard to create and manage the configuration of physical standby DB instances in different AWS Regions from the primary DB instance. It replicates data over secure network connections between a primary DB instance and its read replicas across Regions.

Cross-Region read replicas provide:

- High availability and data protection against single-Region failure.
- The ability to scale read operations to another AWS Region that's closer to your application's users.

You can promote an Oracle read replica to a standalone DB instance explicitly, or you can promote it implicitly by deleting its source DB instance. When you promote a read replica, the DB instance is rebooted before it becomes available. The promoted read replica behaves the same as any other Oracle DB instance.

The following diagram shows the configuration of Amazon RDS for Oracle cross-Region read replicas.
The data transferred for cross-Region replication incurs Amazon RDS data transfer charges.

For more information about using read replicas, see Working with read replicas and Working with Oracle read replicas for Amazon RDS in the AWS documentation. For more information about data transfer pricing, see Amazon RDS Pricing.

Amazon RDS Custom for Oracle

If you're unable to move to a fully managed service such as Amazon RDS because of customization requirements, you can migrate to Amazon RDS Custom for Oracle. With Amazon RDS Custom, you can retain administrative rights to the database and its underlying operating system.

When to choose Amazon RDS for Oracle

Amazon RDS Custom for Oracle is a good migration option when:

- You have legacy, custom, and packaged applications that require access to the underlying operating system and database environment.
- You need access to SYS or SYSTEM user to meet vendor-based application deployment requirements.
- You need access to the underlying operating system to configure settings, install patches, and enable native features to meet the dependent application's requirements.
- You want to access and customize the database environment (by applying custom database patches or modifying operating system packages) to meet your database and application needs.

How it works

To use Amazon RDS Custom for Oracle, you follow these steps, which are illustrated in the following diagram:

1. Upload your database software to an Amazon Simple Storage Service (Amazon S3) bucket.
2. Create a custom engine version (CEV) and DB instance.
3. Connect your application to the DB instance endpoint and access the host to customize your software.
4. Monitor the notifications generated by Amazon RDS Custom automation.
In Amazon RDS Custom for Oracle, you use your own media, patches, and Oracle licenses. When you create a custom engine version (CEV), Amazon RDS Custom installs the media that you provide. You have access to the underlying EC2 instance that hosts the DB engine. You can access the EC2 instance by using Secure Shell (SSH) or AWS Systems Manager and perform your customizations.

You can also install software to run custom applications and agents. Because you have privileged access to the host, you can modify file systems to support legacy applications. You can also apply custom database patches or modify operating system packages on your Amazon RDS Custom DB instances.

Amazon RDS Custom automatically provides monitoring, backups, and instance recovery, and ensures that your DB instance uses a supported AWS infrastructure, operating system, and database. If you want to customize your instance, you can pause Amazon RDS Custom automation for up to 24 hours and then resume it when your customization work is complete. Pausing the automation prevents Amazon RDS Custom automation from directly interfering with your customizations.

When you resume automation, the support perimeter determines whether your customization of the database or operating system environment interferes with, or breaks, Amazon RDS Custom automation. Amazon RDS Custom supports your customization of the host and database environment as long as your changes don’t put the DB instance outside the support perimeter. The support perimeter checks are performed every 30 minutes by default, and also occur after events such as snapshot deletions or uninstalling the Amazon RDS Custom agent, which monitors the DB instance. The Amazon RDS Custom agent is a critical component for ensuring Amazon RDS Custom functionality. If you uninstall the agent, Amazon RDS Custom runs the support perimeter check after one minute and moves the DB instance outside the support perimeter.

Amazon RDS Custom for Oracle is supported in a limited selection of AWS Regions and with limited DB instance classes. It is available on the Oracle Linux operating system and currently supports Oracle Database Enterprise Edition with the Oracle Database versions listed in the documentation. For specifics, see Requirements and limitations for Amazon RDS Custom for Oracle in the AWS documentation.

For additional information, see the following resources:

- Amazon RDS Custom for Oracle – New Control Capabilities in Database Environment (AWS News blog)
Amazon EC2 for Oracle

Amazon EC2 supports a self-managed Oracle database—that is, it gives you full control over the setup of the infrastructure and the database environment. Running the database on Amazon EC2 is very similar to running the database on your own server. You have full control of the database and operating system-level access, so you can use your choice of tools to manage the operating system, database software, patches, data replication, backup, and restoration. This migration option requires you to set up, configure, manage, and tune all the components, including Amazon EC2 instances, storage volumes, scalability, networking, and security, based on AWS architecture best practices.

For more information about migrating from Oracle to Amazon EC2, see the rehost patterns on the AWS Prescriptive Guidance website. To automatically deploy Oracle Database on an EC2 instance in your AWS account, you can use the AWS Quick Start for Oracle Database.

When to choose Amazon EC2

Amazon EC2 is a good migration option for your Oracle database when:

- You need full control over the database and access to its underlying operating system.
- You want to control your backups, replication, and clustering.
- You want to use features and options that aren’t currently supported by Amazon RDS. For details, see Oracle Database Feature Support in the Amazon RDS documentation.
- You need a specific Oracle Database version that isn’t supported by Amazon RDS. For a list of supported version and editions, see Oracle on Amazon RDS in the Amazon RDS documentation.
- Your database size and performance needs exceed Amazon RDS offerings. For details, see Amazon RDS DB Instance Storage in the Amazon RDS documentation.
- You want to avoid automatic software patches that might not be compliant with your applications.
- You want to achieve higher IOPS and provision storage capacity than the current limits. For details, see Amazon RDS DB Instance Storage in the Amazon RDS documentation.

High availability

Oracle Database on Amazon EC2 can work with any Oracle-supported replication technology to achieve high availability and disaster recovery. Some of the common solutions are Oracle Data Guard, Oracle Active Data Guard, and Oracle GoldenGate.
An Oracle database on Amazon EC2 uses Oracle Data Guard or Oracle Active Data Guard to achieve high availability, data protection, and disaster recovery.

- Oracle Data Guard provides a set of services for creating, maintaining, and managing standby databases, to help protect Oracle production databases against disasters and data corruption. Oracle Data Guard automatically maintains each standby database by transmitting redo changes from the primary database, and then applying the redo to the standby database. If the primary database goes down for any planned or unplanned reason, you can fail over to the standby database by converting it into a primary read-write database. Oracle Data Guard is included with Oracle Database Enterprise Edition (EE) only and doesn’t require a separate license.

- Oracle Active Data Guard provides read-only access to a physical standby database for queries, sorting, reporting, and other read operations while it applies redo changes continuously from the primary database. Oracle Active Data Guard requires a separate license that must be additionally purchased with Oracle Database EE. Oracle Active Data Guard features include Real-Time Query, Automatic Block Repair, Far Sync, Standby Block Change Tracking, Active Data Guard Rolling Upgrade, Global Database Services, and Application Continuity.

The following diagram shows how you can use Oracle Database on Amazon EC2 in two Availability Zones within a single AWS Region. The primary database is a read-write database, and the standby database is configured with either Data Guard (physical standby with no read access) or Active Data Guard. All the redo data from the primary database is transferred and applied to the standby database asynchronously by default.

You can also use Oracle Data Guard or Oracle Active Data Guard to configure high availability and disaster recovery across multiple AWS Regions, using Oracle Database on Amazon EC2 for your primary database and standby database, as illustrated in the following diagram.
VMware Cloud on AWS for Oracle

VMware Cloud on AWS is an integrated cloud offering jointly developed by AWS and VMware. When you migrate Oracle Database to VMware Cloud on AWS, you have full control of the database and operating system-level access, as with Amazon EC2. You can run advanced architectures like Oracle Real Application Cluster (RAC) and Oracle RAC extended clusters (across different Availability Zones) in VMware Cloud on AWS. You can choose from a number of migration methods and tools based on your needs and your existing system.

For online migrations, VMware technologies like VMware Hybrid Cloud Extension (VMware HCX) and HCX vMotion help you migrate VM workloads from on-premises VMware clusters to VMware Cloud on AWS. For offline migrations of Oracle workloads, you can use Oracle Recovery Manager (RMAN), AWS Snowball, AWS Storage Gateway, or VMware HCX.

When to choose VMware Cloud on AWS

VMware Cloud on AWS is a good option for your Oracle database when:

- Your Oracle databases are already running in an on-premises data center in vSphere virtualized environments.
- You need to run Oracle RAC in the cloud.
- You have a large number of databases and you need fast migration (for example, only a few hours) to the cloud without requiring any additional work from the migration team.

For more information, see the blog posts How to Migrate Oracle Workloads to VMware Cloud on AWS and Best Practices for Virtualizing Oracle RAC with VMware Cloud on AWS on the AWS Partner Network (APN) blog.

Tools for homogeneous database migrations

A number of tools and technologies are available for data migration. You can perform the migration in a single phase or in multiple phases, based on your database size, consistency, the bandwidth of
the network connection between your on-premises environment and AWS, and the allowed time for database migration. The following chart provides a list of tools and information to help you choose the option that best meets your needs.

<table>
<thead>
<tr>
<th>Migration tool</th>
<th>Database size</th>
<th>Supports</th>
<th>Recommended for</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle SQL Developer (p. 16) (Database Copy feature)</td>
<td>Up to 200 MB</td>
<td>Amazon RDS, Amazon EC2</td>
<td>Small databases with any number of objects.</td>
</tr>
<tr>
<td>Oracle SQL *Loader (p. 16)</td>
<td>Up to 10 GB</td>
<td>Amazon RDS, Amazon EC2</td>
<td>Small to medium-size databases with a limited number of objects.</td>
</tr>
<tr>
<td>Oracle Export and Import (p. 16)</td>
<td>Up to 10 GB</td>
<td>Amazon RDS, Amazon EC2</td>
<td>Small to medium-size databases with a large number of objects.</td>
</tr>
<tr>
<td>Oracle Data Pump (p. 17)</td>
<td>Up to 20 TB</td>
<td>Amazon RDS, Amazon EC2</td>
<td>Preferred method for any database that's 10 GB – 20 TB in size.</td>
</tr>
<tr>
<td>AWS DMS (p. 16)</td>
<td>Any size</td>
<td>Amazon RDS, Amazon EC2</td>
<td>Minimal downtime migration. Database size is limited by bandwidth. You can use AWS DMS with Oracle Data Pump for large database migrations.</td>
</tr>
<tr>
<td>Oracle GoldenGate (p. 18)</td>
<td>Any size</td>
<td>Amazon RDS, Amazon EC2, VMware Cloud on AWS</td>
<td>Minimal downtime migration. Used with Oracle Data Pump for large database migrations.</td>
</tr>
<tr>
<td>Oracle Data Guard (p. 19)</td>
<td>Any size</td>
<td>Amazon RDS Custom, Amazon EC2, VMware Cloud on AWS</td>
<td>Minimal downtime migration. Used with Oracle RMAN to replicate changes after initial data transfer.</td>
</tr>
<tr>
<td>Oracle RMAN (p. 19)</td>
<td>Any size</td>
<td>Amazon RDS Custom, Amazon EC2, VMware Cloud on AWS</td>
<td>Databases over 2 TB, or if database backup is already in Amazon Simple Storage Service (Amazon S3).</td>
</tr>
<tr>
<td>CloudEndure Migration (p. 20)</td>
<td>Any size</td>
<td>Amazon EC2, VMware Cloud on AWS</td>
<td>Fast replication with minimal downtime during cutover.</td>
</tr>
<tr>
<td>VMware HCX (p. 20)</td>
<td>Any size</td>
<td>VMware Cloud on AWS</td>
<td>HCX vMotion provides online or offline migration of a single virtual machine (VM)</td>
</tr>
</tbody>
</table>
Oracle SQL Developer

Oracle SQL Developer is a free GUI tool from Oracle for data manipulation, administration, development, and management. This Java-based tool is available for Microsoft Windows, Linux, or macOS. You can use the Database Copy feature to migrate small databases to AWS, where the total size of your data is under 200 MB. The data transfer between source and target database is done directly over the network. To use this option, you will need a reliable network connection between the source and target database. In addition, keep in mind that this method does not encrypt data during transfer.

Oracle SQL Developer supports both Amazon RDS for Oracle and Oracle databases on Amazon EC2.

Oracle SQL *Loader

Oracle SQL *Loader is a bulk data load utility available from Oracle for loading data from external files into a database. SQL *Loader is included with the full Oracle Database client binaries. You can use SQL *Loader for small to medium-size databases under 10 GB that contain a limited number of objects. Because this is a schema-based method, it involves exporting specific schemas individually from the source database and loading them into the target database. If you have multiple schemas in a database, you have to repeat the process for each schema.

Oracle SQL *Loader supports both Amazon RDS for Oracle and Oracle databases on Amazon EC2.

Oracle Export and Import

Oracle Export and Import utilities help you migrate databases that are smaller than 10 GB and don't include binary float and double data types. The import process creates the schema objects, so you don't have to run a script to create them beforehand. This makes the process well suited for databases that have a large number of small tables.

You can use this tool for both Amazon RDS for Oracle and Oracle databases on Amazon EC2.

AWS DMS

AWS Database Migration Service (AWS DMS) is a managed service that helps you move data to and from AWS easily and securely. AWS DMS supports most commercial and open-source databases, and facilitates both homogeneous and heterogeneous migrations. AWS DMS offers both one-time full database copy and change data capture (CDC) technology to keep the source and target databases in sync and to minimize downtime during a migration.

AWS DMS can perform a full copy of your Oracle database schema for small (10-20 GB) to medium (100-200 GB) databases. For very large databases, you can migrate the data to Amazon RDS or Amazon EC2 by using Oracle Data Pump, and then use the AWS DMS CDC feature for ongoing replication with minimal downtime. When the data is synchronized, you can cut over to the target database.

The following diagram shows how you can use Oracle Data Pump and AWS DMS together to migrate an on-premises database to Amazon RDS for Oracle with minimal downtime. The Oracle Data Pump export utility exports the schema to database dump files, and then transfers those files to Amazon S3 by using
either AWS Direct Connect or AWS Snowball (depending on the size of the database, network bandwidth, and allowed migration time). After the dump files are loaded into Amazon S3, you can upload the files over to an Amazon RDS for Oracle DB instance. The Oracle Data Pump import utility then imports the data to Amazon RDS for Oracle, and AWS DMS CDC replicates all the changes from the source database to the target Amazon RDS for Oracle database.

For more information about using AWS DMS to migrate Oracle source databases, see Using an Oracle Database as a Source for AWS DMS in the AWS documentation.

**Oracle Data Pump**

*Oracle Data Pump* is an enhanced version of Oracle Export and Import. This utility is used to export and import data and metadata from or to Oracle databases. You can run Data Pump export/import on an entire database, selective schemas, tablespaces, or database objects. Data Pump is the recommended tool for migrating data to AWS, for large databases that range from 10 GB to 20 TB in size. It allows a high degree of parallelism, flexible data extraction options, and scalable operations, which enable high-speed movement of data and metadata from source database to target database. Oracle Data Pump also supports encryption and compression when exporting your data to data dump files.

You can use this tool for both Amazon RDS for Oracle and Oracle databases on Amazon EC2. You can also use Oracle Data Pump with AWS DMS and Oracle GoldenGate to handle the initial data transfer for large databases.

For Amazon RDS for Oracle, after the data is exported into dump files using the Oracle Data Pump export utility, the Oracle Data Pump import utility requires the data files to be available in the database server instance to import them into the database. You can’t access the file system in the Amazon RDS DB instance directly, so you will need to transfer the dump files to Amazon RDS using one of these options:

- Use a database link between the two databases. This process uses Oracle Data Pump and the Oracle DBMS_FILE_TRANSFER package. It creates a database link between the source (on-premises) Oracle database and the target Amazon RDS for Oracle database. This option requires higher bandwidth connectivity between source and target databases; we recommend that you use AWS Direct Connect. This option is recommended only for small databases. For more information, see Importing Data with Oracle Data Pump and a Database Link in the Amazon RDS documentation.
• Use an Amazon S3 bucket. Amazon RDS for Oracle supports Amazon S3 integration. This option is recommended when you have large data dump files and your database size is in terabytes. You can then copy the data dump files from on premises to your S3 bucket by using AWS Direct Connect (if your data size is from 10 GB to 5 TB) or AWS Snowball (if your data size is more than 5 TB) depending on the required migration time for your database.

After the data pump file is uploaded to Amazon S3, you can download it to the `DATA_PUMP_DIR` directory on the target Amazon RDS for Oracle DB instance, and then import the data into the DB instance. For more information, see Importing Data with Oracle Data Pump and an Amazon S3 Bucket in the Amazon RDS documentation.

With Oracle Data Pump, you can migrate larger databases in phases, on a schema-by-schema basis. You can migrate to a different version of the Oracle Database software and also migrate to platforms that have different hardware and software configurations.

Oracle GoldenGate

Oracle GoldenGate is a tool for replicating data between a source database and one or more destination databases with minimal downtime. You can use it to build high availability architectures, and to perform real-time data integration, transactional change data capture, replication in heterogeneous environments, and continuous data replication.

You can run Oracle GoldenGate from your on-premises server in your source environment. However, we recommend that you install and run this tool from an EC2 instance, which serves as the GoldenGate hub, on AWS for better performance. You can have multiple GoldenGate hubs, especially if you are migrating data from one source database to multiple destinations. You can use GoldenGate with Amazon RDS for Active-Active database replication, zero-downtime migration and upgrades, disaster recovery, data protection, and in-region and cross-region replication. For details, see Using Oracle GoldenGate with Amazon RDS in the AWS documentation.

The following diagram shows how to use Oracle Data Pump and Oracle GoldenGate together to migrate an on-premises Oracle database to Amazon RDS for Oracle.

Oracle GoldenGate requires a separate license from Oracle.
Oracle GoldenGate supports both Amazon RDS for Oracle and Oracle databases running on Amazon EC2 or VMware Cloud on AWS.

**Oracle Data Guard**

Oracle Data Guard provides a set of services for creating, maintaining, monitoring, and managing Oracle standby databases. You can migrate your entire Oracle database from on premises to Amazon EC2 with minimal downtime by using Oracle Recovery Manager (RMAN) and Oracle Data Guard. With RMAN, you restore your database to the target standby database on Amazon EC2, using either backup/restore or the duplicate database method. You then configure the target database as a physical standby database with Oracle Data Guard, allowing all the transaction/redo data changes from the primary on-premises database to the standby database.

When the primary on-premises Oracle database is in sync with the target standby database on the EC2 instance, you can switch over to the target database, which will convert it to a read-write database. You can then point your application connections to the new primary database. With this option, you can achieve minimum downtime and get an exact physical copy of your database on AWS. The migration is illustrated in the following diagram.

Oracle Data Guard supports Oracle databases running on Amazon EC2 Amazon RDS Custom, and VMware Cloud on AWS.

**Oracle RMAN**

Oracle Recovery Manager (RMAN) is a tool provided by Oracle for performing and managing Oracle database backups and restorations. You can use RMAN to back up your Oracle database from on premises or from your data center, and restore it to an Oracle database on an EC2 instance. Use this method if you are planning to move your entire database to a self-managed Oracle database on an EC2 instance. The database can be of any size, and you can use parallelism, compression, and encryption in your backups.
You can place the Oracle RMAN backup of your on-premises Oracle database directly in an S3 bucket by using the Oracle Secure Backup (OSB) Cloud module, Storage Gateway, or AWS DataSync. You can then use an AWS Identity and Access Management (IAM) role to give the S3 bucket access to your target Oracle database on an EC2 instance, and restore the database by using the RMAN backup files. You can use Oracle Data Guard to capture incremental backups from your on-premises Oracle database and apply them to the target Oracle database on the EC2 instance until the on-premises and target databases are in sync. You can then perform the switchover at a convenient time.

Oracle RMAN supports Amazon EC2, Amazon RDS Custom, and VMware Cloud on AWS migrations. It’s the recommended approach when you can allow enough downtime for migrating your data to AWS.

**CloudEndure Migration**

**Important**

CloudEndure Migration will be discontinued in all AWS Regions except for the AWS GovCloud (US), Mainland China (Beijing), and Mainland China (Ningxia) Regions on December 30, 2022.

Learn more

AWS Application Migration Service is the primary migration service recommended for lift-and-shift migrations to the AWS Cloud. Customers who currently use CloudEndure Migration are encouraged to switch to Application Migration Service for future migrations.

If you are looking to rehost a large number of machines from an on-premises environment to the AWS Cloud, you can use CloudEndure Migration without worrying about compatibility, performance disruption, or long cutover windows. CloudEndure is a highly automated migration tool that expedites and reduces the cost of cloud migration by offering a highly automated lift-and-shift solution.

Using CloudEndure Migration, you can migrate all applications and databases that run on supported versions of Windows and Linux OS. This includes Windows Server versions 2003, 2008, 2012, 2016, 2019 and Linux distributions such as CentOS, Red Hat Enterprise Linux (RHEL), Oracle Linux (OEL), SUSE Linux Enterprise Server (SLES), Ubuntu, and Debian.

CloudEndure Migration works through an agent that you install on your source machines. You do not have to reboot your on-premises servers, and there is no performance impact on your source environment.

CloudEndure Migration automated rehosting consists of three main steps:

1. Install the agent. The CloudEndure agent replicates entire machines to a staging area in your target environment without causing downtime or impacting performance.
2. Configure and test. You configure your target machine settings and launch non-disruptive tests.
3. Cut over. When you’re ready to launch the production machines, CloudEndure automatically converts them to the AWS infrastructure so they can boot and run natively on AWS.

For more information about using CloudEndure Migration, see Automating your lift-and-shift migration at no cost with CloudEndure Migration in the AWS Compute blog.

**VMware HCX**

VMware Hybrid Cloud Extension (HCX) enables you to migrate your on-premises Oracle databases to AWS without having to retrofit your VMware infrastructure. It includes several migration methods that are detailed in the blog posts How to Migrate Oracle Workloads to VMware Cloud on AWS and Migrating Workloads to VMware Cloud on AWS with Hybrid Cloud Extension (HCX). One of these methods, HCX vMotion, provides a live migration of a single VM with no downtime and high availability.

HCX is available free of charge to VMware Cloud on AWS customers.
Licensing options

Oracle Database licensing on AWS is based on the size of the instance on which the database is installed. Many Oracle Database workloads need high memory, storage, and I/O bandwidth, but are not CPU-bound, so you can reduce the number of virtual CPUs (vCPUs) in your deployment without affecting performance.

AWS offers the following CPU options for optimizing your Amazon RDS and EC2 instances for specific workload or business needs:

- Number of CPU cores: You can customize the number of CPU cores for the instance.
- Threads per core: You can disable multithreading by specifying a single thread per CPU core.

For more information, see Optimizing CPU Options in the Amazon EC2 documentation and Introducing Optimize CPUs for Amazon RDS for Oracle on the AWS website.

You can run Oracle Database on AWS under two different licensing models:

- License Included
- Bring Your Own License (BYOL)

License Included

In the License Included model, the Oracle Database software license is made available by AWS, so you don’t have to purchase your own Oracle license separately. The License Included model pricing includes software, underlying hardware resources, and Amazon RDS management capabilities for Amazon RDS for Oracle. You pay for compute capacity by the hour your DB instance runs, with no long-term commitments. This frees you from the costs and complexities of planning, purchasing, and maintaining hardware.

For both Single-AZ and Multi-AZ deployments, pricing is per DB instance-hour consumed, from the time you launch a DB instance until you stop or delete the instance.

The License Included model supports Oracle Database Standard Edition One (SE1) and Standard Edition Two (SE2). For pricing information, see Amazon RDS for Oracle pricing on the AWS website.

Bring Your Own License

The Bring Your Own License (BYOL) model is intended for customers who prefer to use their existing Oracle Database licenses or purchase new Oracle licenses. If you already own an Oracle database license, you can use the BYOL model to run your Oracle database on Amazon RDS. If you're migrating your Oracle database to Amazon EC2 or to VMware Cloud on AWS, you must use your own Oracle license.

To run a DB instance under the BYOL model, you must have the appropriate Oracle Database license for the DB instance class and Oracle Database edition you want to run. You must also follow Oracle's policies for licensing Oracle Database software in the cloud computing environment.

If you use the BYOL model, you must have a license for both the primary DB instance and the standby DB instance in a Multi-AZ deployment. Amazon RDS supports Multi-AZ deployments for Oracle as a high availability, failover solution. We recommend Multi-AZ for production workloads. For more information, see High Availability (Multi-AZ) for Amazon RDS in the Amazon RDS documentation.

The BYOL model supports Oracle Database Enterprise Edition (EE), Standard Edition (SE), Standard Edition One (SE1), and Standard Edition Two (SE2).
For more information about licensing options for Amazon RDS for Oracle, see Oracle Licensing and the Amazon RDS for Oracle FAQs on the AWS website.
Heterogeneous database migration

Because of the innovations and improvements in open-source databases and cloud computing platforms like AWS, many organizations are moving from proprietary (online transaction processing or OLTP) database engines such as Oracle to open-source engines. Oracle databases are mission-critical systems for any organization, but being locked into a particular vendor is a risky and costly situation. Low operating cost and no licensing fees are compelling reasons to consider switching the underlying database technology to open-source or AWS Cloud-native databases.

Other reasons for migrating off Oracle are vendor lock-in periods, licensing audits, expensive licensing, and cost. Oracle's list pricing is based on a per-core model with additional costs for features such as partitioning and high availability. For this reason, many organizations choose to migrate their Oracle databases to either open-source databases (such as PostgreSQL, MySQL, or MariaDB) or AWS Cloud-native databases (such as Amazon Aurora or Amazon DynamoDB) when they migrate to AWS.

You can also migrate your Oracle data warehouse database to Amazon Redshift, which is a fast, fully managed cloud data warehouse. Amazon Redshift is integrated with your data lake, offers up to three times faster performance than any other data warehouse, and costs up to 75 percent less than any other cloud data warehouse. For more information, see Migrate from Oracle to Amazon Redshift on the AWS website.

To migrate to an open-source or AWS Cloud-native database, choose the right database depending on the type of data you have, the access model, scalability, application practicalities, and complexity. For example, PostgreSQL databases have become very popular in recent years for their powerful functionality and high degree of compatibility with commercial databases, and they're the most common migration target for users who are refactoring their Oracle databases. But migrating from Oracle to PostgreSQL and to other open-source databases has often been difficult and time-consuming, and requires careful assessment, planning, and testing.

This process becomes easier with services like AWS DMS and AWS Schema Conversion Tool (AWS SCT), which help you migrate your commercial database to an open-source database on AWS with minimal downtime.

In heterogeneous database migrations, the source and target databases engines are different, as in Oracle to Amazon Aurora, or Oracle to PostgreSQL, MySQL, or MariaDB migrations. The schema structure, data types, and database code in the source and target databases can be quite different, so the schema and code must be transformed before the data migration starts. For this reason, heterogeneous migration is a two-step process:

- **Step 1.** Convert the source schema and code to match that of the target database. You can use AWS SCT for this conversion.
- **Step 2.** Migrate data from the source database to the target database. You can use AWS DMS for this process.
AWS DMS handles all required data type conversions automatically during migration. The source database can be located in your own premises outside AWS, it can be a database that's running on an EC2 instance, or it can be an Amazon RDS database (see Sources for Data Migration in the AWS DMS documentation). The target can be a database in Amazon EC2, Amazon RDS, or Aurora.

For more information about refactoring your Oracle database on AWS, see the re-architect patterns on the AWS Prescriptive Guidance website.

**Tools for heterogeneous database migrations**

The following chart shows the tools that you can use to migrate from Oracle Database to another database engine.

<table>
<thead>
<tr>
<th>Migration tool</th>
<th>Target database support</th>
<th>Used for</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS SCT (p. 24)</td>
<td>Amazon RDS for MySQL</td>
<td>Schema conversion</td>
</tr>
<tr>
<td></td>
<td>Amazon RDS for PostgreSQL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Amazon Aurora MySQL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Amazon Aurora PostgreSQL</td>
<td></td>
</tr>
<tr>
<td>AWS DMS (p. 25)</td>
<td>Amazon RDS for MySQL</td>
<td>Data migration</td>
</tr>
<tr>
<td></td>
<td>Amazon RDS for PostgreSQL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Amazon Aurora MySQL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Amazon Aurora PostgreSQL</td>
<td></td>
</tr>
</tbody>
</table>

The following subsections provide more information about each tool.

**AWS SCT**

AWS Schema Conversion Tool (AWS SCT) converts your existing commercial database schemas to an open-source engine or to an AWS Cloud-native database. AWS SCT makes heterogeneous database migrations predictable by automatically converting the source database schema and a majority of the
database code objects, including views, stored procedures, and functions, to a format that’s compatible with the target database. Any objects that can’t be automatically converted are clearly marked for manual conversion. AWS SCT can also scan your application source code for embedded SQL statements and convert them as part of a database schema conversion project.

**AWS DMS**

*AWS Database Migration Service (AWS DMS)* migrates your data rapidly and securely to AWS. During migration, the source database remains fully operational, minimizing application downtime. AWS DMS supports homogenous migrations such as Oracle to Oracle as well as heterogenous migrations between different database platforms, such as Oracle to an open-source database or to an AWS Cloud-native database. AWS DMS manages the complexities of the migration process, including automatically replicating data changes that occur in the source database to the target database. After the database migration is complete, the target database remains synchronized with the source database for as long as you choose, and you can switch over to the target database at a convenient time.
Best practices for migrating to Amazon RDS for Oracle

Based on the assessment of your database and your project requirements, if your goal is to migrate to Amazon RDS for Oracle, follow the best practices in this section to provision your target database, perform the migration, and test, operate, and optimize your Amazon RDS for Oracle database.

**Important**
Make sure that you have a rollback plan before you migrate your database.

Provisioning your target database

After you finish assessing, planning, and preparing your database migration strategy, follow these best practices when provisioning your Amazon RDS for Oracle database:

- Right-size the Amazon RDS for Oracle DB instance based on your requirements for CPU, memory, IOPS, and storage type.
- Set the correct time zone and character set.
- Make sure to launch Amazon RDS in the correct virtual private cloud (VPC).
- Create the security groups with correct port and IP addresses.
- Provision your Amazon RDS database in a private subnet for security.
- If possible, provision the DB instance by using the latest Oracle Database version, which is currently 19c. Earlier versions are nearing end of support. For more information, see the Oracle Database end-of-support timeline on the AWS discussion forums and Amazon RDS support for Oracle Database 19c.
- If you want to use encryption, always enable it while you are provisioning the Amazon RDS database.
- Create a separate option group and parameter group for each Amazon RDS database.

Exporting data from your source database

There are many tools for migrating an Oracle database to an Amazon RDS for Oracle database. The most commonly used tool is Oracle Data Pump. Before you export your source Oracle database, check the following to facilitate the export process:

- Check the database size, to see if you can migrate it schema by schema, instead of migrating the full database. Migrating schemas individually is less error prone and more manageable than migrating them all at once.
- Export data in parallel mode, by using the Oracle Data Pump `PARALLEL` parameter, for better performance.
- Check if the tables have large objects (LOBs). If you have large tables with LOBs, we recommend that you export those tables separately.
- During the export process, avoid running long database transactions on your source database to avoid Oracle read inconsistency errors.
- If you are using replication tools such as AWS DMS, Oracle GoldenGate, or Quest SharePlex, make sure that you have enough space on your on-premises server to hold archive logs for 24-72 hours, depending on how long the migration takes.
Transferring data dump files to AWS

If you’re using AWS Direct Connect, which provides high bandwidth connectivity between your on-premises environment and AWS, you can copy the Data Pump files by using either the Oracle DBMS_FILE_TRANSFER utility or the Amazon S3 integration feature. If you do not have high bandwidth through AWS Direct Connect, use AWS Snowball to transfer large database export dump files.

Importing data to your target database

- If you’re migrating a very large database, we recommend that you provision a bigger Amazon RDS instance type initially, for the duration of the migration, for faster data loads. After the migration is complete, you can change the DB instance to the right-sized instance type.
- Increase the size of redo log files, undo tablespaces, and temporary tablespaces to improve performance during migration, if needed.
- Disable the Multi-AZ option during the import process, and enable it after migration is complete.
- Disable the generation of archive logs by setting the backup retention to zero to achieve faster data load.
- Prepare the target database by creating tablespaces, users, roles, profiles, and schemas in advance.
- If you have large tables with LOBs, import each LOB table separately.

Post-import steps

- Check the import log files for errors, and fix any errors after the import is complete.
- Check for invalid objects. If you find any, compile and fix them.
- Some procedures might not compile due to lack of permissions on SYS objects that are not allowed or supported in Amazon RDS. These procedures have to be rewritten.
- If you are using sequences, validate the sequence values against the source database to avoid sequence inconsistency.
- Make sure that the object count in your Amazon RDS database is the same as in the source database. Validate tables, indexes, procedures, triggers, functions, packages, constraints, and other objects.
- If your source database has database links to other databases, test the connectivity to confirm that the links still work.

Testing the migration

We recommend the following tests to validate your application against your new Amazon RDS for Oracle database:

- You might have to upgrade your Oracle client software or JDBC software based on the Amazon RDS for Oracle database version. If you’ve migrated to a newer version of Oracle Database, it might not support older versions of Oracle client software.
- Perform functional testing.
- Compare the performance of SQL queries in your source and target databases, and tune the queries as needed. Some queries might perform more slowly in the target database, so we recommend that you capture the baselines of the SQL queries in the source database.
• When the application team finishes testing and confirms that your Amazon RDS database is functioning properly, you can:
  • Right-size the Amazon RDS DB instance based on your assessment.
  • Enable backup retentions.
  • Enable archive logs.
  • Reset the size of redo log files.
  • Enable the Multi-AZ option.
  • Create Amazon CloudWatch alarms and set up Amazon Simple Notification Service (Amazon SNS) topics for alerts.

For additional validation during the proof-of-concept (POC) phase, we recommend the following supplemental tests:

• Run performance tests to ensure that they meet your business expectations.
• Test database failover, recovery, and restoration to make sure that you're meeting RPO and RTO requirements.
• List all critical jobs and reports, and run them on Amazon RDS to evaluate their performance against your service-level agreements (SLAs).

Operating and optimizing your Amazon RDS database

When your database is on AWS, make sure that you are following best practices in areas such as monitoring, alerting, backups, and high availability in the cloud. For example:

• Set up CloudWatch monitoring, and enable detailed monitoring.
• Use Amazon RDS Performance Insights and the Oracle Enterprise Manager (OEM) Management Agent to monitor your database.
• Set up alerts by using SNS topics.
• Set up automatic backups using AWS Backup. You can also use Oracle Data Pump backups or take manual snapshots.
• For high availability, set up the Amazon RDS Multi-AZ feature.
• If you need read-only databases, set up a read replica (p. 8) within the same or across AWS Regions according to your needs.
Oracle database migration patterns

Use the following links to see the AWS Prescriptive Guidance patterns for migrating an Oracle database to AWS:

- Rehost patterns (from Oracle to Amazon EC2)
- Replatform patterns (from Oracle to Amazon RDS for Oracle)
- Re-architect patterns (from Oracle to open-source and AWS Cloud-native databases)

If you’re looking for patterns that cover the use of a specific tool, type in the tool name in the search box or choose it from a filter. For example, you can use this query to see all Oracle migration patterns that use AWS DMS.
Partners

Database migration can be a challenging project that requires expertise and tools. You can accelerate your migration and time to results through partnership. AWS Database Migration Service partners have the required expertise to help customers migrate to the cloud easily and securely. These partners have the expertise for both homogenous migrations such as Oracle to Oracle, and heterogeneous migrations between different database platforms, such as Oracle to Amazon Aurora or Microsoft SQL Server to MySQL.

Based on your requirements and preferences, you can use the partner to handle the complete migration or to help with only some aspects of the migration. In addition, you can use tools and solutions provided by AWS Partner Network (APN) Partners to help with the migration. For a complete catalog of migration tools and solutions, see AWS Partner tools and solutions.
Additional resources

Blog posts
- Database Migration—What Do You Need to Know Before You Start?
- Migrating Oracle databases with near-zero downtime using AWS DMS
- How to Migrate Your Oracle Database to PostgreSQL
- How to Migrate Your Oracle Database to Amazon Aurora
- How to Migrate Oracle Workloads to VMware Cloud on AWS
- Best Practices for Virtualizing Oracle RAC with VMware Cloud on AWS

AWS documentation
- Amazon Aurora
- Amazon EC2
- Amazon RDS
- Amazon RDS Custom
- Amazon Redshift
- AWS DMS
- AWS SCT
- Using Oracle GoldenGate with Amazon RDS
- Oracle Licensing

Additional information
- Oracle Data Pump
- Oracle Data Guard
- Oracle Export and Import
- Oracle GoldenGate
- Oracle RMAN
- Oracle SQL Developer
- Oracle SQL *Loader
- Licensing Oracle Software in the Cloud Computing Environment
- VMware HCX
Appendix: Oracle migration questionnaire

Use the questionnaire in this section as a starting point to gather information for the assessment and planning phases of your migration project. You can download this questionnaire in Microsoft Excel format and use it to record your information.

Download questionnaire

General information

1. What is the name of your Oracle database?
2. What is the version of your Oracle database?
3. What is the edition of the database: Standard or Enterprise?
4. What is the size of your database?
5. What is the database character set?
6. What is the time zone of the database?
7. What are the average and maximum I/O transactions per second (TPS)?
8. What is the IOPS (on average and maximum) for this database for read/write operations?
9. What is the redo log generation per hour (on average and maximum) per day?
10. How many schemas do you plan to migrate?
11. What is the size of each schema?
12. How many big tables (over 100 GB) do you have per schema?
13. Can you archive the tables that don’t need to migrate?
14. What is the size of system global areas (SGAs) and program global areas (PGAs) or Automatic Memory Management (AMM) usage, in megabytes?
15. How many tables have LOBs? What is the maximum size of the LOBs?
16. Do all your tables with LOBs have primary keys?
17. Do you have database links that point to other databases?
18. What are the SLA requirements for your database?
19. What are the RTO and RPO requirements for your database?
20. How much database downtime can you allow for migration purposes?
21. Do you have any compliance, regulatory, or auditing requirements?

Infrastructure

1. What is the hostname of the database?
2. What is the operating system used for this database?
3. How many CPU cores does the server have?
4. What is the memory size on the server?
5. Are you using local storage?
6. Do you use network-attached storage (NAS) or storage area network (SAN) storage types?
7. Do you have a RAC database? If yes, how many nodes does it have?
8. Do you use partitioning features?
9. Do you use Oracle Spatial?
10. Do you have a multi-tenant database?

Database backups

1. How do you back up your database? How often?
2. What is your retention period for archive logs and backups?
3. Do you use backups to clone your database?
4. Where do you store your backup?

Database security

1. Do you use Oracle Database Vault?
2. Do you use data masking?
3. Do you use Secure Sockets Layer (SSL)?
4. Do you use Oracle Advanced Security features such as Transparent Data Encryption (TDE)?
5. Do you use Oracle Advanced Compression?

Database high availability and disaster recovery

1. What are your high availability requirements?
2. Do you use Oracle Data Guard? Where are your primary and standby database regions?
3. Do you use Oracle Active Data Guard?
4. Do you use a Domain Name System (DNS) alias for database connectivity?
5. Do you use replication tools such as Oracle GoldenGate, Quest SharePlex, or Oracle Streams?
# AWS Prescriptive Guidance glossary

<table>
<thead>
<tr>
<th>AI and ML terms (p. 34)</th>
<th>Migration terms (p. 35)</th>
<th>Modernization terms (p. 39)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AI and ML terms</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The following are commonly used terms in artificial intelligence (AI) and machine learning (ML)-related strategies, guides, and patterns provided by AWS Prescriptive Guidance. To suggest entries, please use the <a href="#">Provide feedback</a> link at the end of the glossary.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>binary classification</td>
<td>A process that predicts a binary outcome (one of two possible classes). For example, your ML model might need to predict problems such as “Is this email spam or not spam?” or “Is this product a book or a car?”</td>
<td></td>
</tr>
<tr>
<td>classification</td>
<td>A categorization process that helps generate predictions. ML models for classification problems predict a discrete value. Discrete values are always distinct from one another. For example, a model might need to evaluate whether or not there is a car in an image.</td>
<td></td>
</tr>
<tr>
<td>data preprocessing</td>
<td>To transform raw data into a format that is easily parsed by your ML model. Preprocessing data can mean removing certain columns or rows and addressing missing, inconsistent, or duplicate values.</td>
<td></td>
</tr>
<tr>
<td>deep ensemble</td>
<td>To combine multiple deep learning models for prediction. You can use deep ensembles to obtain a more accurate prediction or for estimating uncertainty in predictions.</td>
<td></td>
</tr>
<tr>
<td>deep learning</td>
<td>An ML subfield that uses multiple layers of artificial neural networks to identify mapping between input data and target variables of interest.</td>
<td></td>
</tr>
<tr>
<td>exploratory data analysis (EDA)</td>
<td>The process of analyzing a dataset to understand its main characteristics. You collect or aggregate data and then perform initial investigations to find patterns, detect anomalies, and check assumptions. EDA is performed by calculating summary statistics and creating data visualizations.</td>
<td></td>
</tr>
<tr>
<td>features</td>
<td>The input data that you use to make a prediction. For example, in a manufacturing context, features could be images that are periodically captured from the manufacturing line.</td>
<td></td>
</tr>
<tr>
<td>feature importance</td>
<td>How significant a feature is for a model's predictions. This is usually expressed as a numerical score that can be calculated through various techniques, such as Shapley Additive Explanations (SHAP) and integrated gradients. For more information, see <a href="#">Machine learning model interpretability with AWS</a>.</td>
<td></td>
</tr>
</tbody>
</table>
feature transformation  To optimize data for the ML process, including enriching data with additional sources, scaling values, or extracting multiple sets of information from a single data field. This enables the ML model to benefit from the data. For example, if you break down the “2021-05-27 00:15:37” date into “2021”, “May”, “Thu”, and “15”, you can help the learning algorithm learn nuanced patterns associated with different data components.

interpretability  A characteristic of a machine learning model that describes the degree to which a human can understand how the model's predictions depend on its inputs. For more information, see Machine learning model interpretability with AWS.

multiclass classification  A process that helps generate predictions for multiple classes (predicting one of more than two outcomes). For example, an ML model might ask “Is this product a book, car, or phone?” or “Which product category is most interesting to this customer?”

regression  An ML technique that predicts a numeric value. For example, to solve the problem of “What price will this house sell for?” an ML model could use a linear regression model to predict a house's sale price based on known facts about the house (for example, the square footage).

training  To provide data for your ML model to learn from. The training data must contain the correct answer. The learning algorithm finds patterns in the training data that map the input data attributes to the target (the answer that you want to predict). It outputs an ML model that captures these patterns. You can then use the ML model to make predictions on new data for which you don't know the target.

target variable  The value that you are trying to predict in supervised ML. This is also referred to as an outcome variable. For example, in a manufacturing setting the target variable could be a product defect.

tuning  To change aspects of your training process to improve the ML model’s accuracy. For example, you can train the ML model by generating a labeling set, adding labels, and then repeating these steps several times under different settings to optimize the model.

uncertainty  A concept that refers to imprecise, incomplete, or unknown information that can undermine the reliability of predictive ML models. There are two types of uncertainty: Epistemic uncertainty is caused by limited, incomplete data, whereas aleatoric uncertainty is caused by the noise and randomness inherent in the data. For more information, see the Quantifying uncertainty in deep learning systems guide.

**Migration terms**

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

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

- Refactor/re-architect – Move an application and modify its architecture by taking full advantage of cloud-native features to improve agility, performance, and scalability. This typically involves porting the operating system and database. Example: Migrate your on-premises Oracle database to the Amazon Aurora PostgreSQL-Compatible Edition.
- Replatform (lift and reshape) – Move an application to the cloud, and introduce some level of optimization to take advantage of cloud capabilities. Example: Migrate your on-premises Oracle database to Amazon Relational Database Service (Amazon RDS) for Oracle in the AWS Cloud.
- Repurchase (drop and shop) – Switch to a different product, typically by moving from a traditional license to a SaaS model. Example: Migrate your customer relationship management (CRM) system to Salesforce.com.
- Rehost (lift and shift) – Move an application to the cloud without making any changes to take advantage of cloud capabilities. Example: Migrate your on-premises Oracle database to Oracle on an EC2 instance in the AWS Cloud.
- Relocate (hypervisor-level lift and shift) – Move infrastructure to the cloud without purchasing new hardware, rewriting applications, or modifying your existing operations. This migration scenario is specific to VMware Cloud on AWS, which supports virtual machine (VM) compatibility and workload portability between your on-premises environment and AWS. You can use the VMware Cloud Foundation technologies from your on-premises data centers when you migrate your infrastructure to VMware Cloud on AWS. Example: Relocate the hypervisor hosting your Oracle database to VMware Cloud on AWS.
- Retain (revisit) – Keep applications in your source environment. These might include applications that require major refactoring, and you want to postpone that work until a later time, and legacy applications that you want to retain, because there’s no business justification for migrating them.
- Retire – Decommission or remove applications that are no longer needed in your source environment.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>application portfolio</td>
<td>A collection of detailed information about each application used by an organization, including the cost to build and maintain the application, and its business value. This information is key to the portfolio discovery and analysis process and helps identify and prioritize the applications to be migrated, modernized, and optimized.</td>
</tr>
<tr>
<td>artificial intelligence operations (AIOps)</td>
<td>The process of using machine learning techniques to solve operational problems, reduce operational incidents and human intervention, and increase service quality. For more information about how AIOps is used in the AWS migration strategy, see the operations integration guide.</td>
</tr>
<tr>
<td>AWS Cloud Adoption Framework (AWS CAF)</td>
<td>A framework of guidelines and best practices from AWS to help organizations develop an efficient and effective plan to move successfully to the cloud. AWS CAF organizes guidance into six focus areas called perspectives: business, people, governance, platform, security, and operations. The business, people, and governance perspectives focus on business skills and processes; the platform, security, and operations perspectives focus on technical skills and processes. For example, the people perspective targets stakeholders who handle human resources (HR), staffing functions, and people management. For this perspective, AWS CAF provides guidance for people development, training, and communications to help ready the organization for successful cloud adoption. For more information, see the AWS CAF website and the AWS CAF whitepaper.</td>
</tr>
<tr>
<td>AWS landing zone</td>
<td>A landing zone is a well-architected, multi-account AWS environment that is scalable and secure. This is a starting point from which your organizations can quickly launch and deploy workloads and applications with confidence in their security and infrastructure environment. For more information about landing zones, see Setting up a secure and scalable multi-account AWS environment.</td>
</tr>
</tbody>
</table>
| AWS Workload Qualification Framework (AWS WQF) | A tool that evaluates database migration workloads, recommends migration strategies, and provides work estimates. AWS WQF is included with AWS Schema
Conversion Tool (AWS SCT). It analyzes database schemas and code objects, application code, dependencies, and performance characteristics, and provides assessment reports.

**business continuity planning (BCP)**

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

**Cloud Center of Excellence (CCoE)**

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

**cloud stages of adoption**

The four phases that organizations typically go through when they migrate to the AWS Cloud:

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

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

**configuration management database (CMDB)**

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

**epic**

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

**heterogeneous database migration**

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

**homogeneous database migration**

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

**idle application**

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

**IT information library (ITIL)**

A set of best practices for delivering IT services and aligning these services with business requirements. ITIL provides the foundation for ITSM.
<table>
<thead>
<tr>
<th><strong>IT service management (ITSM)</strong></th>
<th>Activities associated with designing, implementing, managing, and supporting IT services for an organization. For information about integrating cloud operations with ITSM tools, see the <a href="#">operations integration guide</a>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>large migration</strong></td>
<td>A migration of 300 or more servers.</td>
</tr>
<tr>
<td><strong>Migration Acceleration Program (MAP)</strong></td>
<td>An AWS program that provides consulting support, training, and services to help organizations build a strong operational foundation for moving to the cloud, and to help offset the initial cost of migrations. MAP includes a migration methodology for executing legacy migrations in a methodical way and a set of tools to automate and accelerate common migration scenarios.</td>
</tr>
<tr>
<td><strong>Migration Portfolio Assessment (MPA)</strong></td>
<td>An online tool that provides information for validating the business case for migrating to the AWS Cloud. MPA provides detailed portfolio assessment (server right-sizing, pricing, TCO comparisons, migration cost analysis) as well as migration planning (application data analysis and data collection, application grouping, migration prioritization, and wave planning). The <a href="#">MPA tool</a> (requires login) is available free of charge to all AWS consultants and APN Partner consultants.</td>
</tr>
<tr>
<td><strong>Migration Readiness Assessment (MRA)</strong></td>
<td>The process of gaining insights about an organization's cloud readiness status, identifying strengths and weaknesses, and building an action plan to close identified gaps, using the AWS CAF. For more information, see the <a href="#">migration readiness guide</a>. MRA is the first phase of the <a href="#">AWS migration strategy</a>.</td>
</tr>
<tr>
<td><strong>migration at scale</strong></td>
<td>The process of moving the majority of the application portfolio to the cloud in waves, with more applications moved at a faster rate in each wave. This phase uses the best practices and lessons learned from the earlier phases to implement a migration factory of teams, tools, and processes to streamline the migration of workloads through automation and agile delivery. This is the third phase of the <a href="#">AWS migration strategy</a>.</td>
</tr>
<tr>
<td><strong>migration factory</strong></td>
<td>Cross-functional teams that streamline the migration of workloads through automated, agile approaches. Migration factory teams typically include operations, business analysts and owners, migration engineers, developers, and DevOps professionals working in sprints. Between 20 and 50 percent of an enterprise application portfolio consists of repeated patterns that can be optimized by a factory approach. For more information, see the discussion of migration factories and the <a href="#">CloudEndure Migration Factory guide</a> in this content set.</td>
</tr>
<tr>
<td><strong>migration metadata</strong></td>
<td>The information about the application and server that is needed to complete the migration. Each migration pattern requires a different set of migration metadata. Examples of migration metadata include the target subnet, security group, and AWS account.</td>
</tr>
<tr>
<td><strong>migration pattern</strong></td>
<td>A repeatable migration task that details the migration strategy, the migration destination, and the migration application or service used. Example: Rehost migration to Amazon EC2 with AWS Application Migration Service.</td>
</tr>
<tr>
<td><strong>migration strategy</strong></td>
<td>The approach used to migrate a workload to the AWS Cloud. For more information, see the 7 Rs (p. 35) entry in this glossary and see Mobilize your organization to accelerate large-scale migrations.</td>
</tr>
<tr>
<td><strong>operational-level agreement (OLA)</strong></td>
<td>An agreement that clarifies what functional IT groups promise to deliver to each other, to support a service-level agreement (SLA).</td>
</tr>
<tr>
<td><strong>operations integration (OI)</strong></td>
<td>The process of modernizing operations in the cloud, which involves readiness planning, automation, and integration. For more information, see the <a href="#">operations integration guide</a>.</td>
</tr>
</tbody>
</table>
organizational change management (OCM)  
A framework for managing major, disruptive business transformations from a people, culture, and leadership perspective. OCM helps organizations prepare for, and transition to, new systems and strategies by accelerating change adoption, addressing transitional issues, and driving cultural and organizational changes. In the AWS migration strategy, this framework is called people acceleration, because of the speed of change required in cloud adoption projects. For more information, see the OCM guide.

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

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

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

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

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

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

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

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

Modernization terms

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

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

domain-driven design  
An approach to developing a complex software system by connecting its components to evolving domains, or core business goals, that each component serves. This concept was introduced by Eric Evans in his book, Domain-Driven Design: Tackling Complexity in the Heart of Software (Boston: Addison-Wesley
For information about how you can use domain-driven design with the strangler fig pattern, see Modernizing legacy Microsoft ASP.NET (ASMX) web services incrementally by using containers and Amazon API Gateway.

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

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

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

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

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

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

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

**Strangler Fig Pattern**
An approach to modernizing monolithic systems by incrementally rewriting and replacing system functionality until the legacy system can be decommissioned. This pattern uses the analogy of a fig vine that grows into an established tree and eventually overcomes and replaces its host. The pattern was introduced by Martin Fowler as a way to manage risk when rewriting monolithic systems. For an
example of how to apply this pattern, see Modernizing legacy Microsoft ASP.NET (ASMX) web services incrementally by using containers and Amazon API Gateway.

two-pizza team

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

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

<table>
<thead>
<tr>
<th>update-history-change</th>
<th>update-history-description</th>
<th>update-history-date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Added section (p. 42)</td>
<td>Added information about migrating Oracle databases to Amazon RDS Custom.</td>
<td>June 30, 2022</td>
</tr>
<tr>
<td>Updated section (p. 42)</td>
<td>Updated the CloudEndure Migration section with the latest information about product availability.</td>
<td>May 10, 2022</td>
</tr>
<tr>
<td>Updated AWS WQF information (p. 42)</td>
<td>Updated the AWS WQF section with the latest support and availability information.</td>
<td>October 16, 2020</td>
</tr>
<tr>
<td>Added sections (p. 42)</td>
<td>Updated Oracle database migration strategies with additional information, added best practices for migrating to Amazon RDS, and added a questionnaire for migration assessment and planning.</td>
<td>March 16, 2020</td>
</tr>
<tr>
<td>Initial publication (p. 42)</td>
<td>—</td>
<td>February 24, 2020</td>
</tr>
</tbody>
</table>