Migrating Oracle Databases from Amazon Web Services to Oracle Cloud Infrastructure Database
# Table of Contents

Purpose of This White Paper 4
Scope and Assumptions 4
Introduction 4
Oracle Cloud Infrastructure Database 5
Online Versus Offline Migrations 6
Migration Utilities 7
  - SQL Developer 7
  - RMAN 7
  - Data Pump 8
  - SQL*Loader 8
  - Oracle GoldenGate 8
  - AWS DMS 8
Migration Operating Systems 8
Interconnectivity Between AWS and Oracle Cloud Infrastructure 9
  - Connectivity Concepts 10
  - Connectivity Options 11
  - Choosing Your Connectivity Option 17
AWS to Oracle Cloud Infrastructure Database Migration Guidance 19
Amazon RDS to Oracle Cloud Infrastructure Migration Options 22
  - RDS Migration Option 1: SQL Developer Data Pump Export/Import 23
  - RDS Migration Option 2: SQL Developer Copy 27
  - RDS Migration Option 3: SQL Developer SQL*Loader 29
  - RDS Migration Option 4: SQL Developer Materialized Views 32
  - RDS Migration Option 5: Migrate Using AWS DMS 34
Purpose of This White Paper

Oracle Cloud Infrastructure is an environment that matches and surpasses the performance, control, and governance of enterprise data centers, while delivering the scale, elasticity, and cost-savings of public clouds. Oracle Cloud Infrastructure is the best cloud platform on which to run your Oracle workloads. When you decided to migrate your databases to the Oracle Cloud Infrastructure, there are several options to consider. This paper explains in detail the different options available for such migrations.

Scope and Assumptions

Several database platforms are available for use with their applications deployed on private or public clouds. This document assumes that you have Oracle databases currently deployed in an Amazon Web Services (AWS) environment and that you want to migrate to Oracle Cloud Infrastructure. Based on the release of the Oracle Relational Database Management System (RDBMS) that you have deployed and the mode of deployment, you could have several different migration options. This document does not cover all the available options but focuses on the options that will enable the fastest and easiest migration to Oracle Cloud Infrastructure.

Although some of the examples in the document perform version upgrades during the migration, it is not the intent of this document to address database version upgrades.

Introduction

Oracle Cloud Infrastructure was built to take advantage of the latest technology to provide a highly secure and reliable environment that enables the management of large and complex infrastructure and application deployments. To ensure high availability, Oracle Cloud Infrastructure deploys the cloud in multiple geographical regions. Every region contains at least three fault-independent availability domains, which have independent data centers with power, thermal, and network isolation. Our flat and fast network provides the latency and throughput of rack adjacency across the whole network, allowing synchronous replication and constant uptime. No network or CPU oversubscription also provides predictable bandwidth and performance.

Oracle Database is an ACID-compliant relational database management system (RDBMS) that has been in popular use for the last 38 years. Oracle databases are used in most of the large enterprise workloads all over the world and are usually the most critical part of such workloads. Oracle Corporation has used those 38 years of unmatched leadership in the database market and its deep knowledge of Enterprise IT environments to design and build the next generation cloud infrastructure, aptly called the Oracle Cloud Infrastructure. The Oracle Cloud Infrastructure Database service makes the same Oracle RDBMS available in the public cloud environment.
Oracle Cloud Infrastructure Database provides the following unique advantages when compared to Oracle database deployments on AWS (RDS or EC2 instances).

- Deploys the latest versions of Oracle databases
- Deploys and uses all the features and options of the Oracle databases (for example, Oracle Database In-Memory, Oracle Multitenant option, and so on)
- Deploys Oracle databases on the best cloud platform engineered to meet and exceed the performance and availability demands of critical Oracle database workloads
- Provides the best price performance for Oracle databases on the public cloud
- Provides the ability to use different editions of Oracle (SE1, EE, and so on) with the same licensing model, and a single database support model
- Seamlessly implements and manages hybrid Oracle database environments (on-premises and on the public cloud) with no vendor lock-in
- Implements Real Application clusters for high availability and scalability of your Oracle databases
- Implements your most demanding Oracle database workloads on the Exadata Cloud Service to get the best performance, scalability, and security from a platform engineered by Oracle to be the best platform to run Oracle databases

The remaining sections of this paper describe the Oracle Cloud Infrastructure Database service, the networking options for connecting your AWS environment to Oracle Cloud Infrastructure, the tools that you need for the migration, and step-by-step instructions for each migration method. This paper describes 17 different methods for migration, and guidance on when to use a method is located in the “AWS to Oracle Cloud Infrastructure Database Migration Guidance” section.

Oracle Cloud Infrastructure Database

Oracle Cloud Infrastructure is designed to provide unmatched performance for Enterprise IT and cloud native workloads. It’s the best place to run Oracle applications and Oracle databases, and it’s the only database service that provides Oracle Real Applications Clusters (RAC) and Oracle Exadata systems. The Database service in Oracle Cloud Infrastructure enables you to easily build, scale, and secure Oracle databases in the cloud. You can manage your databases by using simplified tools like patching, Data Guard, and backup and recovery, all of which you can access by using the Oracle Cloud Infrastructure REST APIs or Console. Alternatively, you can access your database host and use your existing tools to manage your databases in the cloud, the same way you manage them on-premises.
Oracle Database versions 11.2.0.4, 12.1.0.2, 12.2.0.1, and 18.0.0.1 are supported on the Oracle Cloud Infrastructure Database service.

The Database service provides customers the following shapes:

- **Virtual machine shapes**: The Database service is available on a variety of virtual machines (VMs) based on standard VM compute shapes. The choice of VM shapes provides cost efficiency and flexibility to select 1 to 24 cores and 256 GB to 40 TB of scalable and durable remote storage. VMs are the best choice for customers who do not need a whole dedicated server for their database applications. A 2-node Oracle RAC environment is also available with VMs.

- **Bare metal shapes**: The Database service supports bare metal shapes that can meet the extreme performance demands of critical enterprise applications. With this option, a whole server, with locally attached NVME or SSD drives, is dedicated to a single customer. You can select 2 to 52 cores, 512 GB to 768 GB of memory, and 3.5 TB to 16.5 TB of usable database storage with the bare metal option.

- **Exadata shapes**: Oracle Exadata Database Machine is the best-performing, most available, and most secure architecture for running Oracle databases. You can use the Exadata Database Machine in the Oracle Cloud Infrastructure and select 22 to 335 CPUs, 1440 GB to 5760 GB of RAM, and 84 TB to 336 TB of usable storage. These environments can service the largest of your database workloads and provide the best performance and availability.

**Online Versus Offline Migrations**

The migration method that you choose depends on the amount of downtime that the application can tolerate.

- If the application can afford between 24 to 72 hours of downtime, consider performing an **offline migration**.

  With offline migrations, the source database is shut down for the duration of the migration, while the database is being migrated in a single step from AWS to Oracle Cloud Infrastructure. The shutdown is required so that data is not changed when the migration is in progress and the database is consistent.

- If the uptime requirements for the database don’t allow for prolonged downtimes, and the migration has to be achieved with minimal to zero downtime, then consider an **online migration**.

  Online migrations typically involve multiple steps in the migration process. An initial step copies the full database from the source in the Amazon Relational Database Service
MIGRATING ORACLE DATABASES FROM AMAZON WEB SERVICES TO ORACLE CLOUD INFRASTRUCTURE DATABASE

(Amazon RDS) to the target database in Oracle Cloud Infrastructure. Additional steps synchronize the target database with the changes that occur in the source database. At the time of the cutover, the final changes are replicated and the applications switch to using the target databases in Oracle Cloud Infrastructure. Tools like Oracle GoldenGate and AWS Database Migration Service (DMS) can be used to keep the source and target databases synchronized in preparation for an online migration.

Migration Utilities

Several tools are available for migrating from AWS to Oracle Cloud Infrastructure. Tools like Oracle SQL Developer, Oracle SQL*Loader, Oracle Data Pump, AWS DMS, and Oracle GoldenGate can be used to perform logical migrations of a database (where data is unloaded from the source and reloaded to the target). Tools like Oracle Recovery Manager (RMAN) can be used to perform physical migrations (where data is copied as-is).

- When you migrate very large databases, it’s often beneficial to perform a physical migration because you don’t have to rebuild the indexes, gather statistics, and so on after the migration.
- Logical migrations are preferred if you want to perform partial migrations, change the database layout, change character sets, change block sizes, or defragment the database. Logical migrations also help to significantly reduce the amount of data to be transferred over a network from the source to the target.

SQL Developer

SQL Developer is a free integrated development environment (IDE) provided by Oracle that simplifies the development and management of Oracle databases. A Java based platform, this IDE can run on Linux, Mac OS X, and Windows platforms. SQL Developer facilitates database migrations by providing options to use Oracle tools like Data Pump export, database copy, and SQL*Loader.

RMAN

RMAN is the Oracle database recovery manager tool that is normally used to back up and recover databases to ensure data protection. RMAN has numerous options that facilitate database migrations like duplicate and transportable tablespaces.
Data Pump

The Data Pump utility can be used to extract data from a source database into an output file in an Oracle specific format (that is OS agnostic) and to load this output file into a target database. You can perform Data Pump exports of the whole database or specific schemas, depending on your requirements. The utility provides options to parallelize the exports and imports and to encrypt the backup files.

SQL*Loader

SQL*Loader is a bulk, flat-file loading utility provided by Oracle. Using SQL*Loader minimizes the need to have properly formatted flat files because it has a powerful declarative language that can be used to format almost any input flat files for loading into the Oracle database.

Oracle GoldenGate

Oracle GoldenGate enables real-time data integration between different Oracle databases. It provides all the necessary programs to capture, transform, propagate, and apply changes happening in a source database to a target database. You can use Oracle GoldenGate to make an initial copy of the source database from Amazon RDS to Oracle Database instances on Oracle Cloud Infrastructure, and also to keep them synchronized until the actual cutover happens. This enables you to cut over to the newly migrated database with minimal downtime.

AWS DMS

You can use the AWS Database Migration Service (DMS) to migrate Oracle databases from AWS to Oracle databases in the Oracle Cloud Infrastructure Database service. With the AWS DMS, you can copy the full schema from the source database to the target database. You can also replicate all the changes from the source database to the target to keep the tables and data synchronized with the source until you decide to cut over to the database in Oracle Cloud Infrastructure.

Migration Operating Systems

Most Oracle database implementations on AWS (Amazon RDS or Amazon EC2) use Linux 64-bit operating systems. Oracle Cloud Infrastructure Database service instances also use 64-bit Linux operating systems. All the migration options described in this document assume that the source environment uses 64-bit Linux operating systems. If the source database instance uses Microsoft Windows 64-bit operating systems, we recommend using Oracle Data Guard to migrate the databases.
Interconnectivity Between AWS and Oracle Cloud Infrastructure

To migrate large databases, you have to securely transfer large amounts of data from the source instances to the target instances over the network as quickly and securely as possible. A robust network with dedicated bandwidth and consistent low latencies ensures that this data transfer is completed in a short timeframe.

Internet gateway (IGW) and dynamic routing gateway (DRG) are service gateway options available for connecting the Oracle Cloud Infrastructure virtual cloud network (VCN) with the internet, on-premises data center, or other cloud providers.

This section describes the connectivity service options that are available to help you plan your network connectivity to the Oracle Cloud in general, and it discusses connectivity options between the cloud providers. All major cloud service providers (CSPs) offer three distinct network connectivity service options: public internet, IPSec VPN, and dedicated connections. Oracle’s dedicated connectivity service is called Oracle Cloud Infrastructure FastConnect. Depending on the workloads and the amount of data that needs to be transferred, one, two, or all three network connectivity service options are required.

<table>
<thead>
<tr>
<th>Service</th>
<th>Max (Mb/s)</th>
<th>Latency</th>
<th>Jitter</th>
<th>Cost</th>
<th>Secure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public internet</td>
<td>&lt; 10,000</td>
<td>Variable</td>
<td>Variable</td>
<td>Variable</td>
<td>No</td>
</tr>
<tr>
<td>IPSec VPN</td>
<td>&lt; 250</td>
<td>Variable</td>
<td>Variable</td>
<td>Variable</td>
<td>Yes</td>
</tr>
<tr>
<td>FastConnect</td>
<td>&lt; 100,000</td>
<td>Predictable</td>
<td>Predictable</td>
<td>Predictable</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Public internet provides accessibility from any internet-connected device. IPSec VPN is a secured encrypted network that provides access by extending your network into the cloud. FastConnect provides dedicated connectivity and offers an alternative connectivity to internet. Because of the exclusive nature of this service, it is more reliable and offers low latency, dedicated bandwidth, and secure access.

The FastConnect service offers the following connectivity models:

- Connectivity via an Oracle network provider or exchange partner
- Connectivity via direct peering within the data center
- Connectivity via dedicated circuits from a third-party network
Connectivity Concepts

Before reading the specific connectivity models available for the migration practices, review some of the important components of Oracle Cloud Infrastructure network connectivity.

**Private peering:** Enables you to extend your existing infrastructure into a VCN in Oracle Cloud Infrastructure. Communication across the connection is with IPv4 private addresses (typically RFC 1918).

**Public peering:** Enables you to access public services in Oracle Cloud Infrastructure without using the internet (for example, Oracle Cloud Infrastructure Object Storage, the Oracle Cloud Infrastructure Console and APIs, or public load balancers in your VCN). Communication across the connection is with IPv4 public IP addresses.

**Virtual cloud network (VCN):** Your virtual network in Oracle Cloud Infrastructure. You can use a VCN to extend your infrastructure into the cloud. For more information, see [VCNs and Subnets](#).

**Virtual circuit:** An isolated network path that runs over one or more physical network connections to provide a single, logical connection between the edge of your existing network and Oracle Cloud Infrastructure. **Private virtual circuits** support private peering, and **public virtual circuits** support public peering (see [FastConnect Private and Public Peering](#)). Each virtual circuit is made up of information shared between you and Oracle, as well as a network service provider or exchange provider (if you're connecting through an Oracle FastConnect partner). You could have multiple private virtual circuits, for example, to isolate traffic from different parts of your organization.

**Internet gateway (IGW):** A router that connects the edge of the cloud network with the internet. Traffic that originates in your VCN and is destined for a public IP address outside the VCN goes through the internet gateway.

**Dynamic routing gateway (DRG):** A virtual edge router that is attached to your VCN and is necessary for private peering. The DRG is a single point of entry for private traffic coming in to your VCN, whether it's over FastConnect or an [IPSec VPN](#). After creating the DRG, you must attach it to your VCN and add a route for the DRG in the VCN's route table to enable traffic flow.

These concepts are explained in detail in the [FastConnect documentation](#).
Connectivity Options

This section describes the optimal connectivity options for database migration. To compare the options based on speed, cost, and time, see “Choosing Your Connectivity Option.”

Option 1: Connecting via a Cloud Exchange

Exchange providers can provide connectivity to a large ecosystem of cloud providers over the same dedicated physical connection between on-premises and the exchange provider. Some available providers are Megaport, Equinix, and Digital Realty.

To route between the clouds, you have the following options:

- Use the virtual router service from the exchange provider—for example, Megaport Cloud Router (MCR).
- Position a physical customer edge (CE) device in colocation with the exchange provider.

The following table shows the pros and cons of using a virtual router service versus placing a physical router collocated with exchange provider:

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using a virtual router service</td>
<td></td>
</tr>
<tr>
<td>• Ease of deployment</td>
<td>• Flexibility to make routing changes is within the</td>
</tr>
<tr>
<td>• Bandwidth on demand</td>
<td>scope of support from the cloud exchange</td>
</tr>
<tr>
<td>• Cost effective in deploying and</td>
<td>• Non-availability of public IP communication</td>
</tr>
<tr>
<td>maintaining</td>
<td></td>
</tr>
<tr>
<td>Using a dedicated physical router</td>
<td></td>
</tr>
<tr>
<td>• Flexibility in managing routing</td>
<td>• Long deployment times</td>
</tr>
<tr>
<td>functions</td>
<td>• Scaling limitations</td>
</tr>
<tr>
<td>• Ability to deploy hardware of choice</td>
<td>• Hardware maintenance and associated</td>
</tr>
<tr>
<td></td>
<td>monetary costs</td>
</tr>
</tbody>
</table>

Although the scope of this paper is to provide optimal connectivity options with a partner-agnostic approach, we chose to discuss the Megaport Cloud Router (MCR) option in detail because of its ease of deployment.

Setting up the connectivity involves the following steps:

1. Connect FastConnect with Megaport through the Oracle Cloud Infrastructure Console.

2. Connect AWS Direct Connect with Megaport through the AWS console.
3. Create the MCR:
   
   A. Create a Virtual Cross Connect (VXC) connection to FastConnect from MCR.
   
   B. Create a VXC connection to AWS Direct Connect from MCR.

Connect FastConnect with Megaport Through the Oracle Cloud Infrastructure Console

This procedure assumes the following prerequisites:

- You are setting up a private peering connection.
- Tenancy and IAM policies are already set up on Oracle Cloud Infrastructure.

Perform the following steps in the Oracle Cloud Infrastructure Console. The links provide access to detailed instructions.

1. Create a VCN.
2. Create a DRG.
3. Attach the DRG to your VCN.
4. Set up a route table by creating a route rule.
5. Set up a virtual circuit.
   
   A. Click the **Networking** tab and select **FastConnect**. Ensure that you are in the correct compartment.
   
   B. Click **Create Connection**.
   
   C. Select **Connect Through a Provider**, select **Megaport**, and then click **Continue**.
   
   D. Select **Private Virtual Circuit** and then select the DRG.
   
   E. Provide the BGP (Border Gateway Protocol) IP address and ASN information.
   
6. Finish creating the virtual circuit. Note the OCID and the Customer BGP IP address.
Connect AWS Direct Connect with Megaport Through the AWS Console

This procedure assumes the following prerequisites:

- The Amazon Virtual Private Cloud (VPC) is created with an appropriate CIDR (not overlapping with the VCN on Oracle).
- A route table with route tags is created and available.

The following steps cover how to establish a virtual private gateway, a private virtual interface, and a Direct Connect gateway.

1. Create a virtual private gateway:
   A. Select the customer ASN and provide a private ASN.
   B. Attach the virtual private gateway to the VPC.

2. In the AWS Direct Connect Connections pane, create a private virtual interface:
   A. Select the virtual interface type as **Private**.
   B. Select the appropriate virtual interface owner.
   C. For **Connection To**, select the **Virtual Private Gateway**.

3. Create a Direct Connect gateway to access your VPCs in remote AWS regions (required if access to other region VPCs is required):
   A. Create the Direct Connect gateway.
   B. Attach the virtual private gateway to the Direct Connect gateway.

Note the ASN provided while creating virtual private gateway and Direct Connect gateway. If you want to connect to VPCs in remote regions, you need to provide the corresponding ASN on the Megaport portal.
Create the MCR and the VXC Connections

In the Megaport portal, you create an MCR and two VXC connections, one to Oracle and one to AWS from the MCR.

Create the MCR

1. Log in to the Megaport portal and select Create MCR.

2. Select the location and choose the rate limit. The available options are 100 Mbps, 500 Mbps, and 1 Gbps through 5 Gbps in increments of one.

Create the VXC Connection to FastConnect

Before creating the VXC, you must have the Oracle virtual circuit ID (OCID) and the Oracle ASN (31898).

1. From the MCR, select +Connection and choose Cloud.

2. From list of cloud providers, choose Oracle.

3. Paste the Oracle virtual circuit ID (OCID).

4. Provide the connection name. The rate limit is auto-populated based on the capped maximum value queried from the OCID.
5. Provide the VCN IP CIDR and BGP details. The following screenshot shows a sample.

![Screenshot showing VCN IP CIDR and BGP details]

6. Finish the process and verify the service status and BGP status.


**Create the VXC Connection to AWS Direct Connect**

1. From the MCR, select **Connection** and choose **Cloud**.

2. From list of cloud providers, choose **AWS**.

3. Provide the connection name and select the rate limit.

4. Under **Connection Details for AWS Service**, select **Private**.

5. Provide the BGP peering information. You can choose a private ASN or Megaport default ASN 133937.

6. Finish the process and verify that the VXC is successfully attached to the MCR.

7. On the AWS Direct Connect dashboard, check the inbound virtual interface request.

8. Click **Accept Virtual Interface**.

9. Select either the virtual private gateway (for one-to-one VPC mapping in a single region) or the Direct Connect gateway that you created earlier.
10. Verify the BGP status change from Pending Acceptance to Pending and then Available after the BGP is established.

11. Enable VPC route propagation and check the received route table.
   A. Navigate to the VPC on the Routes Tables tab.
   B. On the Route Propagation tab, click Edit, select the Propagate check box for the associated virtual private gateway, and then click Save.

MCR will start receiving routes.

Step-by-step instructions for creating the VXC connection to AWS via Direct Connect from an MCR are available at https://knowledgebase.megaport.com/megaport-cloud-router/mcr-aws-directconnect/.

Option 2: Connecting via an IPSec VPN

IPSec VPN provides added security by encrypting data traffic. The achievable bandwidth over VPN is limit to 250 Mbps and, therefore, multiple VPN tunnels might be required depending on the total amount of data to transfer and the required transfer rate.

Steps-by-step instructions for creating a secure connection between Oracle Cloud Infrastructure and other cloud providers are available in the following blog post:


Choosing Your Connectivity Option

The following high-level pointers can help you determine the connectivity choice. The data provides a quick and overall idea, but ultimately the best connectivity option varies for different use cases.

Speed

- FastConnect offers 1G and 10G port speeds.
- Direct Connect offers port speeds of 50M, 100M, 200M, 300M, 400M, 500M, 1G, and 10G.
- IPSec VPN speeds are limited under 500Mb/s in most cases.
Cost

- Oracle FastConnect charges a flat port-hour fee, and there are no charges for data transfer. For more information, see [Oracle FastConnect Pricing](#).

- Oracle IPSec VPN service does not charge for inbound data transfer, outbound data transfer is free up to a 10 TB transfer, and there is a small fee after the 10 TB limit is exceeded. For more information, see [Oracle IPSec VPN Pricing](#).

- Amazon pricing has a port-fee and data transfer charge. Inbound data is not metered but outbound data is metered and charged. For more information, see [Amazon Direct Connect Pricing](#).

- Megaport pricing is based on the rate limit that you choose when you create the MCR. The options available are 100 Mbps, 500 Mbps, and 1, 2, 3, 4, and 5 Gbps. Charging rates (per monthly values) are displayed at the time of deployment based on where you are deploying the MCR and the regions that your connection spans.

Time

Data transfer times depend on the speed choices made at each hop. Comparing dedicated connectivity and IPSec VPN, dedicated connectivity provides a deterministic timeframe because the connectivity uses a private medium and is more reliable and consistent.

The following table shows hypothetical cost scenarios based on bandwidth for the time to data transfer from AWS to Oracle Cloud Infrastructure:

<table>
<thead>
<tr>
<th>Rate Gb/s</th>
<th>Data (TB)</th>
<th>10</th>
<th>100</th>
<th>1,000</th>
<th>10,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>22h13m12s</td>
<td>9d6h13m12s</td>
<td>92d14h13m12s</td>
<td>925d22h13m12s</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>2h13m12s</td>
<td>22h13m12s</td>
<td>9d6h13m12s</td>
<td>92d14h13m12s</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>13m12s</td>
<td>2h13m12s</td>
<td>22h13m12s</td>
<td>9d6h13m12s</td>
</tr>
</tbody>
</table>

100 Gb/s speed can be achieved by link-aggregating multiple 10G links.
AWS to Oracle Cloud Infrastructure Database Migration Guidance

The following table provides guidelines to help you choose a migration option that meets your needs. The migration options are presented in detail in the next section.

<table>
<thead>
<tr>
<th>Source</th>
<th>Method</th>
<th>Type</th>
<th>Online/Offline</th>
<th>Zero Downtime?</th>
<th>Database Size</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDS, EC2</td>
<td>SQL Developer Data Pump Export/Import</td>
<td>Logical</td>
<td>Online</td>
<td>Yes</td>
<td>&lt;= 5 TB</td>
<td>Use for databases up to 5 TB in size. Zero downtime can be achieved by combining with Oracle GoldenGate.</td>
</tr>
<tr>
<td>RDS, EC2</td>
<td>SQL Developer Copy</td>
<td>Logical</td>
<td>Online</td>
<td>No</td>
<td>&lt;= 200 MB</td>
<td>Use for small databases, with any number of objects.</td>
</tr>
<tr>
<td>RDS, EC2</td>
<td>SQL Developer SQL*Loader</td>
<td>Logical</td>
<td>Online</td>
<td>No</td>
<td>&lt;= 10 GB</td>
<td>Use for small to medium databases with a limited number of objects. This method can't migrate objects like packages, procedures, and functions.</td>
</tr>
<tr>
<td>RDS, EC2</td>
<td>SQL Developer Materialized Views</td>
<td>Logical</td>
<td>Online</td>
<td>Yes</td>
<td>&lt;= 500 GB</td>
<td>Use for small databases with a limited number of objects.</td>
</tr>
<tr>
<td>RDS, EC2</td>
<td>AWS DMS</td>
<td>Logical</td>
<td>Online</td>
<td>Yes</td>
<td>&lt;= 5 TB</td>
<td>Use for databases up to 5 TB in size, and when you need zero downtime migrations.</td>
</tr>
<tr>
<td>EC2</td>
<td>Data Pump Conventional Tablespace Export/Import</td>
<td>Logical</td>
<td>Online</td>
<td>Yes</td>
<td>&lt;= 20 TB</td>
<td>Use for any database up to 20 TB in size, in the following conditions: • You want to move the whole database or a subset of schemas. • You want to minimize the amount of data transferred over the network from source to target. • You want to change the physical layout of the database on the target. • You want to do character set conversions.</td>
</tr>
<tr>
<td>Source</td>
<td>Method</td>
<td>Type</td>
<td>Online/Offline</td>
<td>Zero Downtime?</td>
<td>Database Size</td>
<td>Remarks</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>---------------</td>
<td>----------------</td>
<td>----------------</td>
<td>---------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Combine with Oracle GoldenGate to perform zero downtime migrations.</td>
</tr>
<tr>
<td>EC2</td>
<td>Data Pump Full Transportable</td>
<td>Physical</td>
<td>Offline</td>
<td>No</td>
<td>Any size</td>
<td>Use when you want to transport all the tablespaces in the database.</td>
</tr>
<tr>
<td>EC2</td>
<td>Data Pump Transportable Tablespace</td>
<td>Physical</td>
<td>Offline</td>
<td>No</td>
<td>Any size</td>
<td>Use when you want to transport only a subset of the tablespaces from the source database to the target.</td>
</tr>
<tr>
<td>EC2</td>
<td>Remote Clone a PDB</td>
<td>Physical</td>
<td>Online</td>
<td>No</td>
<td>&lt;= 5 TB</td>
<td>Use this method in the following conditions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- You are using a 12c multitenant database on the source.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- The databases are development or test databases. (This method performs a point-in-time recovery on the target and hence does not guarantee full transaction consistency with the source database.)</td>
</tr>
<tr>
<td>EC2</td>
<td>Remote Clone a Non-CDB</td>
<td>Physical</td>
<td>Online</td>
<td>No</td>
<td>&lt;= 5 TB</td>
<td>Use this method in the following conditions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- You have a 12c non-CDB database (container database) on the source and you want to convert to using multitenant on the target.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>- The databases are development or test databases. (This method performs a point-in-time recovery on the target and hence does not guarantee full transaction consistency with the source database.)</td>
</tr>
<tr>
<td>Source</td>
<td>Method</td>
<td>Type</td>
<td>Zero Downtime?</td>
<td>Database Size</td>
<td>Remarks</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>------</td>
<td>----------------</td>
<td>---------------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>EC2</td>
<td>RMAN Transportable PDB</td>
<td>Physical</td>
<td>Online</td>
<td>No</td>
<td>Any size</td>
<td>Use this method when you are using 12c pluggable databases (PDBs) in the source, want to migrate a PDB, and want to minimize downtime. (The downtime is limited to the time taken for the last incremental backup.)</td>
</tr>
<tr>
<td>EC2</td>
<td>RMAN Transportable Backup Set</td>
<td>Physical</td>
<td>Offline</td>
<td>No</td>
<td>Any size</td>
<td>Use this method when you are using 12c PDBs in the source, want to migrate only a specific set of tablespaces, and want to minimize downtime. (The downtime is limited to the time taken for the last incremental backup.)</td>
</tr>
</tbody>
</table>
| EC2    | RMAN Duplicate from Active Database | Physical | Online | No | <= 5 TB | Use this method in the following conditions:  
- You want a one-step method that does not require any intermediate temporary storage.  
- The databases are development or test databases. (This method performs a point-in-time recovery on the target and hence does not guarantee full transaction consistency with the source database.) |
| EC2    | Unplug and Plug a PDB | Physical | Offline | No | Any size | Use this method in the following conditions:  
- You are using 12c and multitenant databases on the source.  
- The business requirements allow for the database to incur downtime for the full migration. |
<table>
<thead>
<tr>
<th>Source</th>
<th>Method</th>
<th>Type</th>
<th>Online/Offline</th>
<th>Zero Downtime?</th>
<th>Database Size</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC2</td>
<td>Unplug and Plug a Non-CDB</td>
<td>Physical</td>
<td>Offline</td>
<td>No</td>
<td>Any size</td>
<td>Use this method in the following conditions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• You are using 12c and non-CDB databases on the source.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• The business requirements allow for the database to incur downtime for the full migration.</td>
</tr>
<tr>
<td>EC2</td>
<td>Migrate Using Data Guard</td>
<td>Physical</td>
<td>Online</td>
<td>Yes</td>
<td>Any size</td>
<td>Use this method if you want to achieve zero downtime migration without using any additional tools like Oracle GoldenGate.</td>
</tr>
<tr>
<td>EC2</td>
<td>Migrate Using RMAN</td>
<td>Physical</td>
<td>Online</td>
<td>No</td>
<td>Any size</td>
<td>Use this method for migrating the largest of databases with minimal downtime. The downtime can be limited to the time it takes to create the final incremental backup, transport it to the target, and restore and recover by using the backup.</td>
</tr>
</tbody>
</table>

Amazon RDS to Oracle Cloud Infrastructure Migration Options

Because Amazon Relational Database Service (RDS) doesn’t provide any operating system logins to the database server, the database migration options are limited to the ones that can be achieved over SQL*Net connections. To provide the simplest end-user experience, the first four of the following options use SQL Developer to execute the migration. These options assume that SQL Developer is already installed.

- SQL Developer Data Pump Export/Import
- SQL Developer Copy
- SQL Developer SQL*Loader
- SQL Developer Materialized Views
- AWS DMS
Note: All of the procedures in this section can also be used to migrate databases from AWS EC2 instances.

RDS Migration Option 1: SQL Developer Data Pump Export/Import

This migration method uses the Oracle Data Pump utility to extract the data from the source database. This method of migrating databases is very scalable and highly performant. The data is extracted into export files that have a platform-independent format. These files are transferred to the target by using a database utility called DBMS_FILE_TRANSFER, and then the import process is executed to load the data into the target database.

Advantages: This method allows for a logical migration of the database, can be parallelized, and provides restart ability.

Online/offline: This method can be performed online. Transactions can be synchronized by using Oracle GoldenGate for capture and apply.

Database size: Use for databases up to 5 TB size. This method can be used for larger databases if there is sufficient free space in the file system on the RDS instance, where the Data Pump dump file is created.

1. In the SQL Developer UI, navigate to View > DBA.

2. Expand the connection to your database instance, right-click Data Pump, and choose Data Pump Export Wizard.

3. On the Source page of the wizard, choose the export type.
4. On the Schemas page, choose the schemas to export.

5. Move through the next two pages, Filter and Table Data, without making any changes.

6. On the Options page, choose the parallelism for the export by specifying the **Thread Number** value.

Parallelism of 2 means that during the export process, two parallel processes will read data from the database and write to the output files. Ensure that you have enough CPU, memory, and I/O resources on the RDS instance to accommodate these processes.
7. On the Output Files page, choose the output file name format.

![Output Files Page](image)

**DATA_PUMP_DIR** points to the `/rdsdbdata/datapump` directory of the Amazon RDS instance.

8. Move through the next two screens to submit the export job. Monitor the Data Pump export job until it is complete.

9. List the files in **DATA_PUMP_DIR** by running the **RDS_FILE_UTIL.LISTDIR** package.

![List Files in DATA_PUMP_DIR](image)
10. Create a database link from the Amazon RDS source database to the Oracle Cloud Infrastructure target database. For example:

```sql
create database link to_tgtdb3 connect to System identified by WelCome_12
using
' (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=129.213.83.23)(PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=tgtdb3_iad14p.rsub1ad1.rvcn1.oraclevcn.com)))';
```

11. Execute `dbms_file_transfer.put_file` to read the files from `DATA_PUMP_DIR` and transfer them to the target database host. The database link created in the previous step is used for the file transfer.

```sql
dbmss_file_transfer.put_file

12. Import the Data Pump files into the target database.

If you have operating-system-level access to the Database service instance on Oracle Cloud Infrastructure, you can log in to the Database service instance and execute the Data Pump import process

```bash
[oracle@rkcomp1]$ impdp system/WelCome_12@tgtdb3 schemas=HR
directory=DATA_PUMP_DIR dumpfile=DATA_PUMP_DIR:hrexp01.DMP,hrexp02.DMP
logfile=DATA_PUMP_DIR:hrimp01.log
```

13. After the data is successfully imported, check the object and record counts to ensure that everything imported correctly.
RDS Migration Option 2: SQL Developer Copy

This method uses the copy tool available from SQL Developer to copy the contents of data from a source database to a target database. This quick method is good for copying small schemas with small amounts of data.

**Advantages:** This method provides a wizard-based quick migration of all objects from a source database schema.

**Online/offline:** This method can be performed online. It doesn’t provide the ability to perform zero-downtime upgrades.

**Database size:** Use for databases up to 200 MB in size.

1. In the SQL Developer UI, navigate to **Tools > Database Copy** to open the Database Copy Wizard.

2. On the Source/Destination page, choose the source and destination databases (and hence schemas). In the **Copy Options** area, choose to copy the whole schema, and select the **Copy DDL** and **Copy Data** check boxes.
3. On the Object Types pages, choose the schema object types that you want to copy.

4. On the Select Schemas page, choose the schemas that you want to copy.
5. Move through the next three pages to complete the wizard and initiate the copy.

You will see dialog boxes like the following one while the copy is in progress.

![Dialog box](image)

After the copy process has completed without any errors, the schemas with all the objects and their data will be in the target database.

**RDS Migration Option 3: SQL Developer SQL*Loader**

SQL*Loader is an Oracle utility that loads data from external files into Oracle database tables. It has a powerful data parsing engine, which puts little limitation on the format of the data in the data file. Data is typically stored in a *loader file*, and the control information that tells SQL*Loader how to format the data is in a *control file*.

You can use SQL Developer to extract the DDL and data from your source database schemas in Amazon RDS and then use SQL*Loader to load them into database tables in a database in Oracle Cloud Infrastructure. Consider that you can use the process to load only the tables; other schema objects like packages, procedures, functions, and triggers must be manually created in the target database.

Before you start this process, ensure that you have created the schema (for example, HR) and its accompanying tablespace (for example, HRTS) on the target database.

**Advantages:** This method provides a wizard-based quick migration of tables and data.

**Online/offline:** This method can be performed online. It doesn’t provide the ability to perform zero-downtime migrations.

**Database size:** Use for databases up to 10 GB in size.

1. In the SQL Developer UI, navigate to **Tools > Database Export** to open the Export wizard.

2. On the Source/Destination page, perform the following steps:

   A. Choose the connection to connect to the schema that you want to transport via SQL*Loader (for example, HR).
B. In the **Export Data** section, choose **loader** as the format.

C. Choose a directory to create the SQL*Loader files in.

![Source/Destination](sourceDestination.png)

3. Move through the four remaining pages to submit the request and initiate the file creation.

The schema export progress bar is displayed as the export runs.

![Exporting Progress](exportingProgress.png)

After the export is complete, a screen is displayed that shows the names of the files that were created. The export creates the following type of files:

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;object_name&gt;.sql</code></td>
<td>Object creation DDL</td>
</tr>
<tr>
<td><code>&lt;object_name&gt;_SEQ.sql</code></td>
<td>Sequence creation DDL</td>
</tr>
<tr>
<td><code>&lt;object_name&gt;_CONSTRAINT.sql</code></td>
<td>Constraint creation DDL</td>
</tr>
<tr>
<td><code>&lt;object_name&gt;_REFCONSTRAINT.sql</code></td>
<td>Foreign key constraint creation DDL</td>
</tr>
<tr>
<td><code>&lt;object_name&gt;_UK.sql</code></td>
<td>Unique index creation DDL</td>
</tr>
<tr>
<td>File Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>&lt;object_name&gt;_PK.sql</code></td>
<td>Unique index creation DDL for primary keys</td>
</tr>
<tr>
<td><code>&lt;object_name&gt;_IX.sql</code></td>
<td>Index creation DDL</td>
</tr>
<tr>
<td><code>&lt;object_name&gt;_DATA_TABLE.ctl</code></td>
<td>SQL*Loader control file for the object</td>
</tr>
<tr>
<td><code>&lt;object_name&gt;_DATA_TABLE.ldr</code></td>
<td>SQL*Loader data file for the object</td>
</tr>
<tr>
<td><code>Generated-&lt;timestamp&gt;.sql</code></td>
<td>A file that has the commands to load the objects</td>
</tr>
</tbody>
</table>

4. **Create the objects in the target database.**

A. **Invoke SQL*Plus and log in to the target database and schema in Oracle Cloud Infrastructure to which you want to load the objects.**

B. **Run the SQL scripts to create the objects that you want to load.** An easy way to do this is to run the `Generated-<timestamp>.sql` file from SQL*Plus, which will create all the objects in the database.

**Tip:** Consider creating indexes on the objects at a later time, after the data is fully loaded, to improve load performance.
5. Using the `.ctl` files in the unload directory, load the objects one at a time to the target database, by using the `sqlldr` utility.

```
$ RANDAS@GRANDAS-LAP18$ downloads/hr1 sqlldr userid=hr/hr@129.213.28.50:1521/rkmpdb2.rksubien1.rkvcm1.oraclevech.com control=REGIONS_DATA_TABLE.ctl log=REGIONS_DATA_TABLE.log  
SQL*Loader: Release 11.2.0.1.0 - Production on Mon Mar 26 10:18:32 2018  
Copyright (c) 1982, 2017, Oracle and/or its affiliates. All rights reserved.  
Path used: Conventional  
Semit point reached - logical record count 3  
Semit point reached - logical record count 4  
Table "HR."."REGIONS":  
4 Rows successfully loaded.  
Check the log file:  
REGIONS_DATA_TABLE.log  
For more information about the load.  
$ RANDAS@GRANDAS-LAP18$ downloads/hr1 sqlldr userid=hr/hr@129.213.28.50:1521/rkmpdb2.rksubien1.rkvcm1.oraclevech.com control=COUNTRIES_DATA_TABLE.ctl log=COUNTRIES_DATA_TABLE.log  
SQL*Loader: Release 11.2.0.1.0 - Production on Mon Mar 26 13:16:56 2018  
Copyright (c) 1982, 2017, Oracle and/or its affiliates. All rights reserved.  
Path used: Conventional  
Semit point reached - logical record count 14  
Semit point reached - logical record count 15  
Table "HR."."COUNTRIES":  
15 Rows successfully loaded.  
Check the log file:  
COUNTRIES_DATA_TABLE.log  
For more information about the load.  
$ RANDAS@GRANDAS-LAP18$ downloads/hr1  
```

RDS Migration Option 4: SQL Developer Materialized Views

A materialized view in Oracle is a database object that stores the results of a query. You can use this feature to create copies of source database tables in a target database, which are updated periodically. You can create an initial copy of a large table in the target database, and then keep it updated by using materialized view refreshes. This allows for zero-downtime cutovers to the target database. This method of migration is suitable when you have only a handful of tables to migrate.

**Advantages:** With this method, you can use a GUI to create the materialized views.

**Online/offline:** This method can be performed online. It enables zero-downtime migrations.

**Database size:** Use for databases up to 500 GB in size.

1. Create the materialized view log on the source table.

   Oracle stores rows that describe DML changes to table data in materialized view logs, so that the incremental changes can be replicated to the target database and applied in an incremental fashion to the target tables. To set up materialized views that can be
refreshed using fast refreshes (instead of full refreshes), you first need to create a materialized view log on the source table.

A. In the SQL Developer UI, open the Connections pane.

B. Expand your source database connection, right-click Materialized View Logs, and choose New Materialized View Log.

C. In the Create Materialized View Log dialog box, choose the table on which you want to create the materialized view log, choose all the columns, and then click OK.

2. Create a database link from the source database to the target database, so that the materialized view can be created based on the source table.

```
cREATE DATABASE LINK to_prddb 
CONNECT TO hr IDENTIFIED BY hr 
USING 
' (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP) (HOST = srcdbl.chyqlovvrniy.us-east-2.rds.amazonaws.com) (PORT = 1521)) (CONNECT_DATA = (SERVICE_NAME = ORCL)))';
```
3. Create a materialized view in the target database.

![Create Materialized View]

4. After the materialized view is created, refresh it as follows:

In the Connections pane, expand your database, expand Materialized Views, right-click your materialized view (for example, JOB_HISTORY), and choose Refresh > Force
Refresh Now > Apply.

RDS Migration Option 5: Migrate Using AWS DMS

You can use the Database Migration Service (DMS) provided by AWS to migrate Oracle
databases from AWS to Oracle databases in the Oracle Cloud Infrastructure Database service.
With AWS DMS, you can copy the full schema from the source database to the target database.
You can also replicate all the changes from the source database to the target to keep the tables
and data synchronized with the source until you decide to cut over to the database in Oracle Cloud
Infrastructure.

Advantages: This method enables you to migrate databases by using an existing migration
service.

Online/offline: This method can be performed online. It enables zero-downtime migrations.

Database size: Use for databases up to 5 TB in size.
To use this method, you must meet the following prerequisites:

- You have created an AWS RDS database instance, and you know your login credentials.
- You have created a database service instance in Oracle Cloud Infrastructure, and you know your login credentials.
- You want to migrate the contents of the HR schema (for example) in a database in AWS RDS to the HR schema in a Database service instance in Oracle Cloud Infrastructure.
- You have pre-created the HR schema (empty schema) and the default tablespace HRTS.

Perform the following steps:

1. Create a replication instance.
   
   A replication instance coordinates the replication of data from the source database instance to the target database instance. A replication instance is an EC2 instance with the replication software installed. It provides a mechanism for connecting to the source database, reading the data, formatting the data for consumption by the target data store, and loading the data into the target database instance.

   A. Log in to the AWS Management Console.
   
   B. On the Services page, under Database, click DMS.
   
   C. In the menu pane of the DMS console, click Replication instances, and then click Create replication instance.
   
   D. In the Create replication instance dialog box, enter a name, a description, and a instance shape. If you are transferring a large database, pick a larger instance. Then, click Create replication instance.
2. After the instance is created, create the replication endpoints.

A. On the DMS console, click **Endpoints** and then click **Create endpoint**.

B. Create a source endpoint that helps you connect to the source instance on AWS RDS. Enter the source database engine, host, port, and database SID for the source instance on RDS. Then click **Create endpoint**.
C. Create the target endpoint to connect to the Oracle database on Oracle Cloud Infrastructure. Ensure that you specify the schema name that you are transferring (for example, HR), and ensure that this schema and the underlying tablespace are pre-created.
3. Create the data transfer task.
   
   A. On the DMS console, click **Tasks**, and then click **Create task**.
B. Navigate to the **Tasks** section and choose the HR schema from the source database for replication.

![Homepage](image.png)

C. After entering the details, click **Create task**.

D. After the task is successfully created, click **Start/Resume** to execute the data migration task.

This concludes the five migration methods that you can use to migrate databases from AWS RDS to Oracle Cloud Infrastructure Database. The next section addresses the migration methods that you can use to migrate Oracle databases deployed on AWS EC2 instances.

**Amazon EC2 to Oracle Cloud Infrastructure Migration Options**

AWS provides customers the ability to install and configure Oracle Database software and Oracle databases on Amazon EC2 instances. When you do this, you must install and configure all the required packages, kernel parameters, and OS configurations, in addition to installing and configuring the Oracle software. This option provides more flexibility than setting up Oracle Database environments in AWS RDS. However, when you use this option, you do not get the benefits of the database service tooling, in terms of taking backups and patching the environments.

Because you have full control of the host running the Oracle databases, you can choose from more options when migrating from Amazon EC2 environments to the Database service on Oracle...
Cloud Infrastructure. With this type of a deployment, you can use Oracle utilities like Data Pump and RMAN to move the environments from AWS to Oracle Cloud Infrastructure.

**EC2 Migration Option 1: Data Pump Conventional Tablespace Export/Import**

Data Pump is an Oracle utility that enables high-speed data transfer from one database to another. Use Data Pump to move an entire schema (or multiple schemas) from a source database that resides in Amazon EC2 to a database created in the Database service in Oracle Cloud Infrastructure.

Although this method is similar to the "RDS Migration Option 1: SQL Developer Data Pump Export/Import" method detailed earlier, it’s different in the following aspects:

- You use the expdp utility to perform the export (instead of SQL Developer).
- Because you have full access to the underlying host, you use scp to transfer the files from the EC2 instance to the Oracle Cloud Infrastructure instance (instead of DBMS_FILE_TRANSFER).

**Advantages:** Data Pump allows for fast unloading and loading of large amounts of data, which can be parallelized, compressed, and encrypted. Because indexes are not exported, significantly reduces the amount of data that needs to be transferred over the network.

**Online/offline:** This method can be performed online. When used with Oracle GoldenGate, it enables zero-downtime migrations.

**Database size:** Use for databases up to 20 TB in size.

1. Create the export data file location.

   Data Pump extracts the data from the database and writes it to output files to a file system that is available from the database host. To get the best performance while exporting large databases, provision a large file system with very good write throughput.

   Log in to the source database and create the data file destination directory.

   ```sql
   SQL> create directory MIG_DATA as '/u01/oradump';
   Directory created.
   SQL> grant all on directory MIG_DATA to public;
   Grant succeeded.
   SQL>
   ```
2. Export the schemas that you want to transport.
   ```
   expdp system/manager@prdpdb1 SCHEMASES=HR DIRECTORY=MIG_DATA
   DUMPFILE=exphr.dmp LOGFILE=MIG_DATA:exphr.log
   ```

3. Copy the file from the source host to the target host.
   ```
   scp -i ~/.ssh/ociob /u01/oradump/exphr.dmp
   oracle@129.213.28.50:/u01/oradump/.
   ```

4. Create the Data Pump input file directory in the target database.
   ```
   SQL> create directory MIG_DATA as '/u01/oradump';
   Directory created.
   SQL> grant all on directory MIG_DATA to public;
   Grant succeeded.
   ```

5. Create the tablespaces required by the schemas that you are importing.
   ```
   SQL> create tablespace hrts datafile size 1G autoextend on maxsize 5G;
   ```

6. Import the data to the target databases.
   ```
   impdp
   system/WelCome#_12@129.213.28.50:1521/rkbmpdb2.rksublad1.rkvcn1.oraclevcn.com
   SCHEMASES=HR DIRECTORY=MIG_DATA DUMPFILE=exphr.dmp
   ```

The whole database has been migrated to the Database service in Oracle Cloud Infrastructure and is ready for use.

**EC2 Migration Option 2: Data Pump Full Transportable**

You can use the Data Pump full transportable method to copy an entire database from your on-premises host to the Database service on Oracle Cloud Infrastructure.

**Advantages:** Because you're copying the entire database from source to target, you don't have to re-create indexes and gather statistics after the migration. This reduces the time needed to instantiate the database on the target.

**Online/offline:** This is an offline method. It doesn't provide the ability to perform zero-downtime migrations.

**Database size:** Use for databases of any size.

In the following example, you copy a full 11.2.0.4 version database from a source host, place the data files on Oracle Automatic Storage Management (Oracle ASM), and import into a 12.2 database as a pluggable database (PDB).
1. Create a directory in the source database in which to place the Data Pump metadata export output file.

   SQL> create directory MIG_DATA as '/u01/oradump';
   Directory created.
   SQL> grant all on directory MIG_DATA to public;
   Grant succeeded.
   SQL>

2. Place the source tablespaces in read-only mode.

   SQL> alter tablespace hrts read only;
   Tablespace altered.
   SQL> alter tablespace oets read only;
   Tablespace altered.
   SQL> alter tablespace users read only;
   Tablespace altered.

3. Run a Data Pump full transportable export on the source databases.

   expdp system FULL=Y TRANSPORTABLE=ALWAYS VERSION=12 DIRECTORY=MIG_DATA DUMPFILE=MIG_DATA:expallts.dmp LOGFILE=MIG_DATA:expallts.log

4. Copy the data files and the export dump file from the source to a temporary directory on the target.

5. Copy the data files from the temporary location to Oracle ASM on the target host.

   Log in as the grid user and set the environment variables for Oracle ASM. Invoke the ASMCMD command-line utility from the operating system command line.

   ASMCMD> cp /u01/tmp/hrts01 +DATA/RKDB3_IAD14P/6762A4F294A1C7B5E0534601000AB58F/DATAFILE/hrts01.dbf
   copying /u01/tmp/hrts01 ->
   +DATA/RKDB3_IAD14P/6762A4F294A1C7B5E0534601000AB58F/DATAFILE/hrts01.dbf
   ASMCMD> cp /u01/tmp/oets01 +DATA/RKDB3_IAD14P/6762A4F294A1C7B5E0534601000AB58F/DATAFILE/oets01.dbf
   copying /u01/tmp/oets01 ->
   +DATA/RKDB3_IAD14P/6762A4F294A1C7B5E0534601000AB58F/DATAFILE/oets01.dbf
   ASMCMD> cp /u01/tmp/users02.dbf
   +DATA/RKDB3_IAD14P/6762A4F294A1C7B5E0534601000AB58F/DATAFILE/users01.dbf
   copying /u01/tmp/users02.dbf ->
   +DATA/RKDB3_IAD14P/6762A4F294A1C7B5E0534601000AB58F/DATAFILE/users01.dbf
   ASMCMD>
6. After the data files have been copied to the target location, set the source tablespaces to read-write mode.

```sql
SQL> alter tablespace hrts read write;
Tablespace altered.
SQL> alter tablespace oe read write;
Tablespace altered.
SQL> alter tablespace users read write;
Tablespace altered.
```

7. Use Data Pump to import the metadata into the target databases.

```
impdp
system/Welcome#_120129.213.83.23:1521/rkpdb3.rksbladl.rkvcnl.oraclevcn.com FULL=y DIRECTORY=MIG_DATA DUMPFILE=MIG_DATA:expallts.dmp LOGFILE=MIG_DATA:impallts.log \
TRANSPORT_DATAFILES='+DATA/RKDB3_IAD14P/6762A4F294A1C7B5E0534601000AB58F/DATAFILE/hrts01.dbf','+DATA/RKDB3_IAD14P/6762A4F294A1C7B5E0534601000AB58F/DATAFILE/oets01.dbf','+DATA/RKDB3_IAD14P/6762A4F294A1C7B5E0534601000AB58F/DATAFILE/users01.dbf'
```

The migration is complete, and the database is ready for use.

**EC2 Migration Option 3: Data Pump Transportable Tablespace**

You can use the Data Pump transportable tablespace method to copy specific tablespaces from your on-premises host to the Database service on Oracle Cloud Infrastructure.

**Advantages:** Because you’re copying the entire tablespace from source to target, you don’t have to re-create indexes and gather statistics after the migration. This reduces the time needed to instantiate the database on the target.

**Online/offline:** This is an offline method. It doesn’t provide the ability to perform zero-downtime migrations.

**Database size:** Use for databases of any size.

In the following example, you copy a few tablespaces from a source host, place the data files on Oracle ASM, and import into a 12.2 database as a pluggable database (PDB).

1. Create a directory in the source database in which to place the Data Pump metadata export output file.

```sql
SQL> create directory MIG_DATA as '/u01/oradump';
Directory created.
SQL> grant all on directory MIG_DATA to public;
Grant succeeded.
SQL>
```
2. Place the source tablespaces in read-only mode.

```sql
SQL> alter tablespace hrts read only;
Tablespace altered.
SQL> alter tablespace oets read only;
Tablespace altered.
```

3. List the data files for the tablespaces that you are moving.

```sql
SQL> select file_name from dba_data_files where tablespace_name = 'HRTS';
FILE_NAME
-----------------------------
/u01/oradata/prddb/prdpdb1/hrts01.dbf

SQL> select file_name from dba_data_files where tablespace_name = 'OETS';
FILE_NAME
-----------------------------
/u01/oradata/prddb/prdpdb1/oets01.dbf
```

4. Place the tablespaces that you are moving in read-only mode.

```sql
SQL> alter tablespace hrts read only;
Tablespace altered.
SQL> alter tablespace oets read only;
Tablespace altered.
SQL>
```

5. Invoke Data Pump export to perform the transportable tablespace export.

```sql
expdp system/manager@prdpdb1 TRANSPORT_TABLESPACES=HRTS,OETS,
TRANSPORT_FULL_CHECK=YES DIRECTORY=MIG_DATA DUMPFILE=exphrts.dmp
LOGFILE=MIG_DATA:exphrts.log
```

6. Copy the data files and export the output file to the target host.

```bash
[oracle@rkcomp1 122]$ scp -i ~/.ssh/ociob /u01/oradata/prddb/prdpdb1/hrts01.dbf oracle@129.213.28.50:/u01/oradump/.hrts01.dbf
100% 1024MB 125.5MB/s  00:08

[oracle@rkcomp1 122]$ scp -i ~/.ssh/ociob /u01/oradata/prddb/prdpdb1/oets01.dbf oracle@129.213.28.50:/u01/oradump/.oets01.dbf
100% 1024MB 124.0MB/s  00:08

[oracle@rkcomp1 122]$ scp -i ~/.ssh/ociob /u01/oradump/exphrts.dmp oracle@129.213.28.50:/u01/oradump/.exphrts.dmp
```

---

44 | MIGRATING ORACLE DATABASES FROM AMAZON WEB SERVICES TO ORACLE CLOUD INFRASTRUCTURE DATABASE
7. Copy the data files to the target Oracle ASM disk group.

```
ASMCMD> cp /u01/oradump/hrts01.dbf
+DATA/rkbmdb2_iad1r2/684206652ED426EBE0535701000AECE3/DATAFILE/hrts01.dbf
+DATA/rkbmdb2_iad1r2/684206652ED426EBE0535701000AECE3/DATAFILE/hrts01.dbf
ASMCMD> cp /u01/oradump/oets01.dbf
+DATA/rkbmdb2_iad1r2/684206652ED426EBE0535701000AECE3/DATAFILE/oets01.dbf
+DATA/rkbmdb2_iad1r2/684206652ED426EBE0535701000AECE3/DATAFILE/oets01.dbf
```

8. Set the tablespaces in the source database to read-write mode.

```
SQL> alter tablespace hrts read write;
Tablespace altered.
SQL> alter tablespace oets read write;
Tablespace altered.
```

9. Create the users on the target databases.

```
SQL> create user hr identified by hr temporary tablespace temp account unlock;
User created.
SQL> create user oe identified by oe temporary tablespace temp account unlock;
User created.
```

10. Invoke Data Pump and import the metadata into the target database.

```
impdp
system/Welcome#_12@129.213.28.50:1521/rkbmpdb2.rksub1ad1.rkvcn1.oraclevcn.com
DIRECTORY=MIG_DATA DUMPFILE=exphrts.dmp LOGFILE=MIG_DATA:exphrts.log
\TRANSPORT_DATAFILES='+DATA/rkbmdb2_iad1r2/684206652ED426EBE0535701000AECE3/DATAFILE/hrts01.dbf',
'+DATA/rkbmdb2_iad1r2/684206652ED426EBE0535701000AECE3/DATAFILE/oets01.dbf'
```

11. Set the tablespaces on the target database to read-write mode.

```
SQL> alter tablespace hrts read write;
Tablespace altered.
SQL> alter tablespace oets read write;
Tablespace altered.
```
EC2 Migration Option 4: Remote Clone a PDB

If you use multitenant databases and your source database is version 12.1 or later, you can use the remote cloning feature for pluggable databases (PDBs) to clone a PDB from the source database to the target.

**Advantages:** Because you’re copying the entire PDB from source to target, you don’t have to re-create indexes and gather statistics after the migration. This reduces the time needed to instantiate the database on the target.

**Online/offline:** This is an online method, well-suited for migrating development and test databases. It doesn’t provide the ability to perform zero-downtime migrations.

**Database size:** Can be used for databases <= 5TB in size.

1. Create a database link to the source database.

   Log in as `sys` to the container root of the target database and create a database link. Ensure that you are connecting to the PDB that is going to be cloned, using the database link.

   ```
   create database link onprem_pdb connect to system identified by manager using '72.55.128.32:1521/prdpdb1';
   ```

2. Clone the source PDB to the target database.

   ```
   SQL> create pluggable database prdpdb1 from prdpdb1@onprem_pdb;
   Pluggable database created.
   ```

3. Open the PDB.

   ```
   SQL> alter pluggable database prdpdb1 open;
   ```

4. Run the following SQL to identify any issues with the PDB, and perform the necessary steps to rectify them.

   ```
   SQL> select message,time from pdb_plug_in_violations;
   ```

EC2 Migration Option 5: Remote Clone a Non-CDB

If you use multitenant databases and your source database is version 12.1 or later, you can use the remote cloning feature for pluggable databases (PDBs) to clone a non-CDB (container database) from the source database to the target.

**Advantages:** You are copying the entire non-CDB database from the source to the target.
**Online/offline:** This is an online method. It doesn't provide the ability to perform zero-downtime migrations.

**Database size:** Can be used for databases up to 5TB in size.

1. Create a database link to the source database.

   Log in as `sys` to the container root of the target database and create a database link. Ensure that you are connecting to the non-CDB that is going to be cloned, using the database link.

   ```sql
   create database link onprem_noncdb connect to system identified by manager using '72.55.128.32:1521/prdncdb';
   ```

2. Create the pluggable database by cloning the non-CDB from the source.

   ```sql
   SQL> CREATE PLUGGABLE DATABASE rkbmpdb2 FROM prdncdb@onprem_noncdb;
   Pluggable database created.
   ```

3. Log in as `sys` to the newly created `rkbmpdb2` PDB in the target database and run the `noncdb_to_pdb` script.

   ```sql
   SQL> @$ORACLE_HOME/rdbms/admin/noncdb_to_pdb.sql
   ```

4. Open the PDB in read-write mode.

   ```sql
   SQL> alter pluggable database rkbmpdb2 open;
   ```

5. Run the following SQL to identify any issues with the PDB, and perform the necessary steps to rectify them.

   ```sql
   SQL> select message,time from pdb_plug_in_violations;
   ```

**EC2 Migration Option 6: RMAN Transportable PDB**

If you use multitenant databases and your source database is version 12.1 or later, you can use Oracle Recovery Manager (RMAN) to transport the pluggable database (PDB) from the source database to the destination.

**Advantages:** This method allows for a multi-step migration. You can perform multiple incremental recoveries of the target database and minimize the time required for the final recovery to be done during the cutover.

**Online/offline:** This is an online method. Downtime is equal to the time spent taking the final backup, transporting it, and applying it to the target.

**Database size:** Use for databases of any size.
There are slight variations in the options and steps that you would perform to move the PDB, depending on the source database version. The following steps are for moving a 12.2 source PDB using inconsistent backups (that is, the source PDB does not have to be closed).

1. Log in as sys to the container root of the source database and query the current Checkpoint Change number. Note the number.

   ```
   SQL> SELECT CHECKPOINT_CHANGE# FROM V$DATABASE;
   CHECKPOINT_CHANGE#------------------------------------
   1446203
   ```

2. Use RMAN to take an inconsistent backup of the source PDB.

   ```
   rman target /
   RMAN> BACKUP INCREMENTAL LEVEL 0
       FOR TRANSPORT
       ALLOW INCONSISTENT
       PLUGGABLE DATABASE prdpdb1 FORMAT '/u01/backups/hr_pdb_level0.bck';
   ```

3. Use RMAN to take a final consistent backup of the source PDB.

   ```
   sqlplus / as sysdba
   SQL> alter pluggable database prdpdb1 close immediate;
   Pluggable database altered.
   SQL> exit;
   rman target /
   connected to target database: PRDDB (DBID=3107372461)
   RMAN> BACKUP INCREMENTAL FROM SCN 1446203
       FOR TRANSPORT
       UNPLUG INTO '/u01/backups/metadata_hr_pdb.xml'
       PLUGGABLE DATABASE prdpdb1 FORMAT '/u01/backups/hr_pdb_level1_con.bck';
   ```
   This command takes a final incremental backup of the PDB named prbpdb1, unplugs the PDB, and creates the XML file for the unplugged PDB.

4. Copy the backups to the target database host.

   ```
   [oracle@rkcomp1 backups]$ scp -i ~/.ssh/ociob */
   oracle@129.213.28.50:/u01/backups/.
   hr_pdb_level0.bck  100% 486MB 127.1MB/s   00:03
   hr_pdb_level1_con.bck  100% 12MB 146.2MB/s   00:00
   metadata_hr_pdb.xml 100% 9138     3.9MB/s   00:00
   ```
5. Restore the level 0 backup to the target database.

   Here you use the FORMAT switch so that the data files will be restored to the Oracle ASM disk group on the Oracle Cloud Infrastructure Database service.

   RMAN> RESTORE
       FOREIGN PLUGGABLE DATABASE hr_pdb FORMAT '+DATA/RKDB4_IAD1TD/%u'
       FROM BACKUPSET '/u01/backups/hr_pdb_level0.bck';

6. Recover the target PDB by using the level 1 backup from the source database.

   RMAN> RECOVER
       USING '/u01/backups/metadata_hr_pdb.xml'
       FOREIGN DATAFILECOPY
           '+DATA/RKDB4_IAD1TD/7jsublcs','+DATA/RKDB4_IAD1TD/jdsublcs','+DATA/RKDB4_I
           AD1TD/c3sublcs','+DATA/RKDB4_IAD1TD/r7sublcs','+DATA/RKDB4_IAD1TD/u2sublcs
           ','+DATA/RKDB4_IAD1TD/edsublcs'
       FROM BACKUPSET '/u01/backups/hr_pdb_level1_con.bck';

7. Find the name of the restored PDB. (The RMAN restore and recover process gives the PDB a unique new name.)

   RMAN> select name from v$pdbs;

8. Open the PDB.

   RMAN> ALTER PLUGGABLE DATABASE RKDB4_IAD1TD_63249 open;
   Statement processed

EC2 Migration Option 7: RMAN Transportable Backup Set

If you use multitenant databases and your source database is version 12.1 or later, you can use Oracle Recovery Manager (RMAN) to transport selected tablespaces from the source database to the target database.

When you use this method, you can create the target database and keep it ready by performing an initial full backup and then additional incremental backups. The only downtime you will need on the source is the time that you spend taking the last incremental backup and recovering the target database by using this final backup.

**Advantages:** This method allows for a multi-step migration. You can perform multiple incremental recoveries of the target database and minimize the time required for the final recovery to be done during the cutover.

**Online/offline:** This is an online method. The downtime is limited to the time spent to take the final incremental backup, transport the files, and apply the files on the target.
**Database size:** Use for databases of any size.

1. **Use RMAN to take a full backup of the tablespaces from the source database that you are going to move to the target database.**

   ```sql
   rman target sys/manager@rkcomp1:1521/prdpdb1
   RMAN> BACKUP
   FOR TRANSPORT
   ALLOW INCONSISTENT
   INCREMENTAL LEVEL 0
   FORMAT '/u01/backups/hrts_inconsist_l0.bck'
   TABLESPACE HRTS;
   ```

   While this backup is being taken, and afterwards, you can continue to query and transact against this source database.

2. **Take a final incremental backup of the source tablespaces.**
   
   **A. Place the tablespaces in read-only mode.**
   
   ```sql
   sqlplus sys/manager@rkcomp1:1521/prdpdb1 as sysdba
   SQL> alter tablespace hrts read only;
   Tablespace altered.
   ```
   
   **B. Run a final incremental level 1 backup to back up only the changed blocks since the last level 1 backup of the tablespaces.**
   
   ```sql
   oracle@rkcomp1 122]$ rman target sys/manager@rkcomp1:1521/prdpdb1
   connected to target database: PRDDB:PRDPDB1 (DBID=2789311088)
   RMAN> BACKUP
   FOR TRANSPORT
   INCREMENTAL LEVEL 1
   TABLESPACE HRTS
   FORMAT '/u01/backups/hrts_inconsist_l1.bck'
   DATAPUMP FORMAT '/u01/backups/hrts_incr_dmp.dmp';
   ```
   
   Among the backup messages on the screen, you’ll see a few lines like the following ones. Note the file name because you will need it during the restore and recovery steps.
   
   ```
   EXPDP>***********************************************************
   EXPDP> Dump file set for SYS.TRANSPORT_EXP_PRDDB_kDBs is:
   EXPDP> /u01/app/oracle/product/12.2.0.1/db_1/dbs/backup_tts_PRDDB_20164.dmp
   EXPDP>***********************************************************
   ```

3. **Switch the tablespaces in the source database back to read-write mode.**

   ```sql
   sqlplus sys/manager@rkcomp1:1521/prdpdb1 as sysdba
   SQL> alter tablespace hrts read write;
   Tablespace altered.
   ```
4. Use an operating system secure copy utility to copy the files to the target host in Oracle Cloud Infrastructure.

```
[oracle@rkcompl backups]$ scp -i ~/.ssh/ociob /u01/backups/*
oracle@129.213.28.50:/u01/backups/.
```

5. Create the users on the target database.

```
sqlplus / as sysdba
SQL> alter session set container=rkbmpdb2;
SQL> create user hr identified by hr temporary tablespace temp;
SQL> GRANT CREATE SESSION, CREATE VIEW, ALTER SESSION, CREATE SEQUENCE TO hr;
SQL> GRANT CREATE SYNONYM, CREATE DATABASE LINK, RESOURCE, UNLIMITED TABLESPACE TO hr;
```

6. Restore the tablespace from the first full backup.

On the target database service instance, restore the tablespace backup into a PDB. Here you specify the `FORMAT` keyword to specify that the data files have to be restored to the `DATA` disk group.

```
rman target
sys/WelCome#_12@rkbmdb2:1521/rkbmpdb2.rksublad1.rkvcn1.oraclevcn.com
connected to target database: RKBMDB2:RKBMPDB2 (DBID=4247588157)
RMAN> RESTORE FOREIGN TABLESPACE hrts
   FORMAT '+DATA/rkbmdb2_iad1r2/datafile/%u'
   FROM BACKUPSET '/u01/backups/hrts_inconsist_l0.bck';
```

Because you specified the `FORMAT` string `%u`, Oracle assigns a random file name that it generates to the restored data files.

7. Log in to the Oracle ASM instance and use ASMCMD to list the data files from the `+DATA/rkbmdb2_iad1r2/datafile/` directory to get the new file names.

8. Recover the tablespace by using the final incremental backup.

```
rman target
sys/WelCome#_12@rkbmdb2:1521/rkbmpdb2.rksublad1.rkvcn1.oraclevcn.com
connected to target database: RKBMDB2:RKBMPDB2 (DBID=4247588157)
RMAN> RECOVER FOREIGN DATAFILECOPY '+DATA/rkbmdb2_iad1r2/datafile/21surg08'
   FROM BACKUPSET '/u01/backups/hrts_inconsist_l1.bck';
```
9. Restore the Data Pump export dump file from the RMAN backup. Use the file name that you noted in step 2 for the DUMP FILE clause.

```sql
rman target
sys/Welcome#_12@rkbmdb2:1521/rksublad1.rkvcn1.oraclevcn.com
RMAN> RESTORE
    DUMP FILE 'backup_tts_PRDB_20164.dmp'
    DATAPUMP DESTINATION '/u01/oradump'
    FROM BACKUPSET '/u01/backups/hrts_incr_dmp.dmp';
```

10. Now that you have restored and recovered the data files to the target database, import the tablespace metadata by using Data Pump to complete the process.

```sql
impdp
system/Welcome#_12@rkbmdb2:1521/rksublad1.rkvcn1.oraclevcn.com
directory=MIG_DATA dumpfile=backup_tts_PRDB_20164.dmp
transport_datafiles='+DATA/rkbmdb2_iad1r2/datafile/21surg08'
logfile=MIG_DATA:hrts_incr_dmp.log
```

11. Open the tablespaces in read-write mode.

```sql
sqlplus
system/Welcome#_12@rkbmdb2:1521/rksublad1.rkvcn1.oraclevcn.com
SQL> alter tablespace hrts read write;
Tablespace altered.
SQL> alter user hr default tablespace hrts;
User altered
```

**EC2 Migration Option 8: RMAN Duplicate from Active Database**

RMAN can create a copy of the source database in a target database system by copying over all the required files while the database is up and running. When you use a backup-based RMAN duplication process, you must first take a backup of the source system, which you then copy to a staging location on the target server before performing the RMAN duplicate process. When you use the RMAN duplication process from an active database, that additional step of creating a backup becomes unnecessary.

Because you are creating a new database instance in the target environment, you have to provision the required storage for the new instance before you duplicate the database.

**Advantages:** This is a single-step migration that allows for the direct duplication of the source database to the target, with no interim storage required.

**Online/offline:** This is an online method. It doesn’t provide the ability to perform zero-downtime migrations.
**Database size:** Use for databases up to 5 TB in size.

1. Create the database storage for the new database by using `dbcli`.
   
   ```bash
   [root@rkbmdb1 ~]# dbcli create-dbstorage --dbname prddb --dataSize 30
   ```

2. Use the `list-dbstorage` command to verify that the creation has completed.
   
   ```bash
   [root@rkbmdb1 ~]# dbcli list-dbstorages
   ```

3. Run `dbcli` and find the location of the Oracle Home.
   
   ```bash
   [root@rkbmdb1 ~]# dbcli list-dbhomes
   ```

4. Create a new listener by using a non-default port from the Oracle Home. To do this, create a file named `listener.ora` in the `$ORACLE_HOME/network/admin` directory, and add the following lines to the file:

   ```ora
   LISTENER_aux_prddb=
     (DESCRIPTION=
       (ADDRESS_LIST=
         (ADDRESS=(PROTOCOL=TCP)(HOST=rkbmdb1.rksublad1.rkvcn1.oraclevcn.com)(PORT=1528))
       )
     )
   SID_LIST_LISTENER_aux_prddb=
     (SID_LIST=
       (SID_DESC=
         (GLOBAL_DBNAME=prddb)
         (ORACLE_HOME=/u01/app/oracle/product/12.2.0.1/dbhome_1)
         (SID_NAME=prddb)
       )
     )
   ```

5. Open the firewall ports.

   **A. Open port 1528 from the firewall on the database service host.**

   ```bash
   [root@rkbmdb1 ~]# iptables -I INPUT 5 -m state --state NEW -p tcp --port 1528 -j ACCEPT
   [root@rkbmdb1 ~]# service iptables save
   [root@rkbmdb1 ~]# service iptables restart
   ```
B. Open port 1528 on the security list in Oracle Cloud Infrastructure.

In the Oracle Cloud Infrastructure Console, navigate to the security list for the subnet that the Database service is in. Edit the security list, and add a line with the following values:

- **Source CIDR**: 0.0.0.0/0
- **IP Protocol**: TCP
- **Source Port Range**: All
- **Destination Port**: 1528

6. Create an initialization parameter with the following entries to start the auxiliary instance. This is just a temporary file to start the RMAN duplication process. This file will be overwritten by the file from the source instance during the cloning process.

Create the `$ORACLE_HOME/dbs/initprddb.ora` file and add the following entries:

```sql
db_name=prddb
control_files=’+RECO/PRDDB/CONTROLFIE/’
db_block_size=8192
db_file_name_convert=('/u01/oradata/prddb','+DATA/PRDDB/DATAFILE')
log_file_name_convert=('/u01/oradata/prddb','+DATA/PRDDB/DATAFILE')
db_create_file_dest=’+DATA’
db_create_online_log_dest_1=’+RECO’
db_recovery_file_dest=’+RECO’
db_recovery_file_dest_size=50G
```

7. Start the database listener.

```bash
export ORACLE_HOME=/u01/app/oracle/product/12.2.0.1/dbhome_1
export ORACLE_SID=prddb
export ORACLE_UNQNAME=prddb
export NLS_DATE_FORMAT="mm/dd/yyyy hh24:mi:ss"
export PATH=$ORACLE_HOME/bin:$PATH

oracle@rkbmdbl ~]$ lsnrctl start LISTENER_aux_prddb
```
8. Copy the instance password file from the source database to the target database host.

The password file is required to connect remotely to the instance to start the cloning process.

```
scp -i ~/.ssh/ociob /u01/app/oracle/product/12.2.0.1/db_1/dbs/orapwprddb
oracle@rkbmdb1:/u01/app/oracle/product/12.2.0.1/dbhome_1/dbs/.
```

9. Start the auxiliary instance in **nomount** mode on the target node.

```
SQL> sqlplus / as sysdba
SQL> startup nomount;
```

10. Log in to the source database server and execute **rman** to perform the RMAN active duplication. This process makes a full copy of the source database and clones it to the target database server.

```
rman target sys/manager@129.213.89.242:1521/prddb auxiliary
sys/manager@129.213.18.112:1528/prddb
RMAN> uplicate target database to prddb from active database
spfile
PARAMETER_VALUE_CONVERT prddb prddb PRDDB PRDDB
set cluster_database='false'
set db_name='prddb'
set db_unique_name='prddb'
set db_create_file_dest='+DATA'
set db_create_online_log_dest_1='+RECO'
set db_recovery_file_dest='+RECO'
set db_recovery_file_dest_size='50G'
set audit_file_dest = '/u01/app/oracle/product/12.2.0.1/dbhome_1/rdbms/audit'
set log_archive_dest_1='LOCATION=+DATA'
reset control_files;
```

11. On the target database server, log in as root and run **dbcli** to register the database.

```
[root@rkbmdb1 ~]# dbcli register-database --dbclass OLTP --dbshape odbl --servicename prddb -syspassword
```

The database is now available for use.

**EC2 Migration Option 9: Unplug and Plug a PDB**

When you are using the multitenant option of the Oracle database, you can unplug a whole pluggable database (PDB) from the source container database and plug it into a target container database on the Oracle Cloud Infrastructure Database service.

**Advantages:** This is a single-step migration method that requires downtime.
**Online/offline**: This is an offline method. It doesn't provide the ability to perform zero-downtime migrations.

**Database size**: Use for databases of any size.

1. Close the PDB in the source database.
   ```sql
   sqlplus / as sysdb
   SQL> Alter pluggable database prdpdb1 close;
   ```

2. Unplug the PDB.
   ```sql
   SQL> ALTER PLUGGABLE DATABASE prdpdb1 UNPLUG INTO '/u01/tmp/prdpdb1.xml';
   ```

3. Use a secure copy utility to transfer the XML file and the data files to the Database service compute node.
   ```bash
   scp -i ~/.ssh/ociob /u01/tmp/prdpdb1.xml oracle@129.213.18.112:/u01/tmp/.
   scp -i ~/.ssh/ociob /u01/oradata/prddb/prdpdb1/* oracle@129.213.18.112:/u01/tmp/.
   ```

4. On the target database instance, execute the `CREATE PLUGGABLE DATABASE` command to create the prdpdb1 PDB.
   ```sql
   SQL> CREATE PLUGGABLE DATABASE prdpdb1 USING '/u01/tmp/prdpdb1.xml'
   SOURCE_FILE_DIRECTORY = '/u01/tmp'
   COPY
   FILE_NAME_CONVERT = ('/u01/tmp', '+DATA') ;
   ```
   When you specify `FILE_NAME_CONVERT`, the data files are copied from `/u01/tmp` to the DATA disk group.

5. Open the PDB.
   ```sql
   SQL> alter pluggable database prdpdb1 open;
   ```

6. Query the following view to identify any errors in the PDB and rectify them.
   ```sql
   SQL> select message,time from pdb_plug_in_violations;
   ```

**EC2 Migration Option 10: Unplug and Plug a Non-CDB**

If your source database is a 12c non-CDB (container database), you can use the following steps to move the whole non-CDB as a pluggable database (PDB) into a CDB on the Oracle Cloud Infrastructure Database service.

**Advantages**: This is a single-step migration
Online/offline: This is an offline method. It doesn't provide the ability to perform zero-downtime migrations.

Database size: Use for databases of any size.

1. Start the source non-CDB database in read-only mode.

   sqlplus / as sysdba
   SQL> startup mount;
   SQL> alter database open read only;

2. On the source database host, execute the DBMS_PDB DESCRIBE command to generate an XML file that contains the list of data files that will be plugged in on the target database.

   sqlplus / as sysdba
   SQL> BEGIN
           DBMS_PDB.DESCRIBE(
               pdb_descr_file => '/u01/oradump/ncdb.xml');
           END;

3. Use a secure copy utility to copy the output of the DBMS_PDB DESCRIBE command and the data files to the target database host.

   scp -i ~/.ssh/ociob /u01/oradump/ncdb.xml oracle@129.213.18.112:/u01/tmp/.
   scp -i ~/.ssh/ociob /u01/oradata/prdncdb/* oracle@129.213.18.112:/u01/tmp/.

4. Create the PDB in the CDB in Oracle Cloud Infrastructure.

   SQL> CREATE PLUGGABLE DATABASE rkbmpdb2 USING '/u01/tmp/ncdb.xml'
       SOURCE_FILE_DIRECTORY = '/u01/tmp'
       COPY
       FILE_NAME_CONVERT = ('/u01/tmp', '+DATA') ;

5. Execute the cleanup script by connecting to the rkbmpdb2 PDB (the one you just created in the target CDB) and run the $ORACLE_HOME/rdbms/admin/noncdb_to_pdb script.

   SQL> alter session set container=rkbmpdb2;
   SQL> @$ORACLE_HOME/rdbms/admin/noncdb_to_pdb

6. Open the CDB.

   SQL> alter pluggable database rkbmpdb2 open;

7. Query the following view to identify any errors in the PDB and rectify them.

   SQL> select message,time from pdb_plug_in_violations;
EC2 Migration Option 11: Migrate by Using Data Guard

Oracle Data Guard provides a set of services that create, maintain, and manage standby databases. Standby databases are full copies of the primary database and can be used to reduce the downtime involved in migrating to the Oracle Cloud Infrastructure Database service. The standby database can be kept in sync with the primary database by using the redo transport and redo apply services. When you are ready to cut over to Oracle Cloud Infrastructure, you can simply perform a Data Guard failover, and you will have an up-to-date, fully functional database on Oracle Cloud Infrastructure.

Advantages: Using Data Guard enables you to set up the target database on Oracle Cloud Infrastructure and, if required, run some tests (by using active Data Guard and or snapshot standby databases) before cutting over to the target database.

Online/offline: This is an online method and allows for zero-downtime migrations.

Database size: Use for databases of any size.

The following steps describe how to create a standby database by using the RMAN duplication method, and how to failover to the standby database for the migration.

Because you are creating a new database instance in the target environment, you have to provision the required storage for the new instance before you duplicate the database.

1. Create the database storage for the new database by using dbcli.
   ```bash
   [root@rkbmdb2 ~]# dbcli create-dbstorage --dbname prddb --dataSize 30
   ```

2. Use the `list-dbstorage` command to verify that the creation has completed.
   ```bash
   [root@rkbmdb2 ~]# dbcli list-dbstorages
   ```

3. Run `dbcli` and find the location of the Oracle Home.
   ```bash
   [root@rkbmdb2 ~]# dbcli list-dbhomes
   ```

4. Set up force logging in the source database.
   ```sql
   SQL> alter database force logging;
   SQL> ALTER SYSTEM SET STANDBY_FILE_MANAGEMENT=AUTO;
   ```

5. Configure the primary database to receive redo log data.
   ```sql
   SQL> ALTER DATABASE ADD STANDBY LOGFILE ('/u01/oradata/prddb/sbredolog1.rdo') SIZE 100M;
   SQL> ALTER DATABASE ADD STANDBY LOGFILE ('/u01/oradata/prddb/sbredolog2.rdo') SIZE 100M;
   ```
6. Update the `tnsnames.ora` file with the service name entries.

Add the following entries to the `$ORACLE_HOME/rdbms/admin/tnsnames.ora` files in both the source and target database systems. In the entries, `18.218.43.72` is the IP address for the host where the source database is currently running, and `rkbmdb2.rksublad1.rkvcn1.oraclevcn.com` is the hostname of the target database host in Oracle Cloud Infrastructure.

```plaintext
prddb =
   (DESCRIPTION =
   (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = 18.218.43.72)(PORT = 1521))
   )
   (CONNECT_DATA =
   (SID = prddb)
   )
   )
prddb_stby =
   (DESCRIPTION =
   (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = rkbmdb2.rksublad1.rkvcn1.oraclevcn.com)(PORT = 1528))
   )
   (CONNECT_DATA =
   (SID = prddb)
   )
   )
```

7. Create a new listener by using a non-default port from the Oracle Home. On the target database, create a file named `listener.ora` in the `$ORACLE_HOME/network/admin` directory, and add the following lines to the file:

```plaintext
LISTENER_aux_prddb=
   (DESCRIPTION=
   (ADDRESS_LIST=
   (ADDRESS=(PROTOCOL=TCP)(HOST=rkbmdb2.rksublad1.rkvcn1.oraclevcn.com)(PORT=1528))
   )
   )
SID_LIST_LISTENER_aux_prddb=
   (SID_DESC=
   (GLOBAL_DBNAME=prddb_stby)
   (ORACLE_HOME=/u01/app/oracle/product/12.2.0.1/dbhome_1)
   (SID_NAME=prddb))
```
8. Open the firewall ports.
   
   **A. Open port 1528 from the firewall on the Database service host.**
   
   ```bash
   [root@rkbmdb2 ~]# iptables -I INPUT 5 -m state --state NEW -p tcp --dport 1528 -j ACCEPT
   [root@rkbmdb2 ~]# service iptables save
   [root@rkbmdb2 ~]# service iptables restart
   ```

   **B. Open port 1528 on the security list in Oracle Cloud Infrastructure.**
   
   In the Oracle Cloud Infrastructure Console, navigate to the security list for the subnet that the Database service is in. Edit the security list, and add a line with the following values:
   
   - **Source CIDR:** 0.0.0.0/0
   - **IP Protocol:** TCP
   - **Source Port Range:** All
   - **Destination Port:** 1528

9. On the target database, start the database listener.

   ```
   export ORACLE_HOME=/u01/app/oracle/product/12.2.0.1/dbhome_1
   export ORACLE_SID=prddb
   export ORACLE_UNQNAME=prddb
   export NLS_DATE_FORMAT="mm/dd/yyyy hh24:mi:ss"
   export PATH=$ORACLE_HOME/bin:$PATH
   
   oracle@rkbmdb2 ~]$ lsnrctl start LISTENER_aux_prddb
   ```

10. Copy the instance password file from the source database to the target database host.

    The password file is required to connect remotely to the instance to start the cloning process.
    
    ```
    scp -i ~/.ssh/ociob /u01/app/oracle/product/12.2.0.1/db_1/dbs/orapwprddb
    oracle@rkbmdb2:/u01/app/oracle/product/12.2.0.1/dbhome_1/dbs/.
    ```
11. Create an initialization parameter with the following entries to start the auxiliary instance. This is just a temporary file to start the RMAN duplication process. This file will be overwritten by the file from the source instance during the cloning process.

Create the $ORACLE_HOME/dbs/initprddb.ora file and add the following entries:

```ora
db_name=prddb
control_files='+RECO/PRDDB/CONTROLFIE/'
db_block_size=8192
db_file_name_convert=('/u01/oradata/prddb','+DATA/PRDDB/DATAFILE')
log_file_name_convert=('/u01/oradata/prddb','+DATA/PRDDB/DATAFILE')
db_create_file_dest='+DATA'
db_create_online_log_dest_1='+RECO'
db_recovery_file_dest='+RECO'
db_recovery_file_dest_size=50G
```

12. Start the auxiliary instance in nomount mode on the target node.

```
SQL> sqlplus / as sysdba
SQL> startup nomount;
```

13. Log in to the Oracle ASM instance and create the directory for PRDDB.

```sql
asmcmd
ASMCMD> mkdir +DATA/PRDDB
ASMCMD> mkdir +DATA/PRDDB/DATAFILE
```

14. Log in to the source database server and execute rman to perform the RMAN active duplication. This process makes a full copy of the source database and clones it to the target database server.

```sql
rman target sys/manager@18.218.43.72:1521/prddb auxiliary
sys/manager@129.213.28.50:1528/prddb
RMAN> duplicate target database for standby from active database
dorecover
spfile
set cluster_database='false'
set db_name='prddb'
set db_unique_name='prddb_stby'
set control_files='+DATA/PRDDB/DATAFILE/control01.ctl','+DATA/PRDDB/DATAFILE/control02.ctl'
set db_create_file_dest='+DATA'
set db_create_online_log_dest_1='+RECO'
set db_recovery_file_dest='+RECO'
set db_recovery_file_dest_size=50G
set audit_file_dest = '/u01/app/oracle/product/12.2.0.1/dbhome_1/rdbms/audit'
set log_archive_dest_1='LOCATION='+DATA'
nofilenamecheck
;```
15. Start the managed recovery on the target database.

```sql
sqlplus / as sysdba
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM SESSION;
```

16. Set up the Data Guard broker.

A. From both databases, run the following commands:

```sql
sqlplus / as sysdba
SQL> ALTER SYSTEM SET dg_broker_start=true;
```

B. From the primary database, run the following commands:

```bash
dgmgrl sys/manager@rkcomp1:1521/prddb
Connected to "prddb"
Connected as SYSDBA.
DGMGRL> CREATE CONFIGURATION prddb_dg_config AS PRIMARY DATABASE IS prddb CONNECT IDENTIFIER IS prddb;
Configuration "prddb_dg_config" created with primary database "prddb"
DGMGRL> ADD DATABASE prddb_stby AS CONNECT IDENTIFIER IS prddb_stby MAINTAINED AS PHYSICAL;
Database "prddb_stby" added
DGMGRL> Enable Configuration;
Enabled.
DGMGRL>
```

17. Fail over to the new standby database. On the new standby database host, run the following commands:

```bash
dgmgrl sys/manager@prddb_stby
Connected to "prddb_stby"
Connected as SYSDBA.
DGMGRL> failover to prddb_stby;
Performing failover NOW, please wait...
Failover succeeded, new primary is "prddb_stby"
DGMGRL>
```

At this point you have fully cut over to the database copy in Oracle Cloud Infrastructure.

**EC2 Migration Option 12: Migrate by Using Recovery Manager**

Oracle Recovery Manager (RMAN) provides all the necessary programs and processes to back up and restore your database efficiently. If your source database is running on a little-endian platform, you can use RMAN to back up your database from the source instance and restore it to the Database service instance on Oracle Cloud Infrastructure. RMAN has the necessary mechanisms to parallelize, compress, and encrypt backups on the source instance.
You have two methods of transferring the backups from the source environment to the target environment. You can use secure copy mechanisms as shown in the previous options (for example, scp), or you can back up your database directly to Object Storage in Oracle Cloud Infrastructure and restore directly from Object Storage to the Database service instance. The following procedure details the secure copy method; the section following the steps explains the Object Storage method.

**Advantages:** This method allows for a multi-step migration. You can perform multiple incremental recoveries of the target database and minimize the time required for the final recovery during the cutover.

**Online/offline:** This is an online method. It doesn't provide the ability to perform zero-downtime migrations.

**Database size:** Use for databases of any size.

1. Use RMAN to create a full backup of your source database and archive logs. RMAN places the backup files that it creates on a file system that you specify during the backup.

   The following command performs password-based encryption of the backup, and encryptit is the password used as an example. RMAN backs up the database to files that are stored in /u01/backups in this example.

   ```
   rman target /
   RMAN> set encryption identified by encryptit only
   executing command: SET encryption
   using target database control file instead of recovery catalog
   RMAN> run {
       allocate channel c1 device type disk format '/u01/backups/%U' ;
       backup as compressed backupset incremental level 0 SECTION SIZE=512M
       DATABASE PLUS ARCHIVELOG TAG='fullbackup';
       release channel c1;
   }
   ```

2. Use an OS-level secure copy utility to transfer the RMAN backup files from the source to the target instance. Ensure that you also copy the controlfile and spfile autobackup.

   ```
   scp -i ~/.ssh/ociob /u01/backups/* oracle@129.213.28.50:/u01/backups/.
   scp -i ~/.ssh/ociob /u01/app/oracle/product/12.2.0.1/db_1/dbs/c-3107896219-20180327-02 oracle@129.213.28.50:/u01/backups/.
   ```

3. Run the following queries on the source database and note their output:

   ```
   SQL> Select dbid from v$database;
   SQL> Select member from v$logfile;
   ```
4. Create the database storage for the new database by using `dbcli`.

   ```
   [root@rkbmdb2 ~]# dbcli create-dbstorage --dbname prddb --dataSize 30
   ```

5. Use the `list-dbstorage` command to verify that the creation has completed.

   ```
   [root@rkbmdb2 ~]# dbcli list-dbstorages
   ```

6. Run `dbcli` and find the location of the Oracle Home.

   ```
   root@rkbmdb2 ~]# dbcli list-dbhomes
   ```
   
   Restore the spfile from the backup to the target host. In this command, you are using the DBID that you queried from the source database in step 3.

   ```
   export ORACLE_SID=prddb
   rman target /
   RMAN> startup force nomount;
   RMAN> set dbid 3107896219;
   RMAN> set DECRIPTION identified by encryptit;
   RMAN> allocate channel c1 device type disk format '/u01/backups/%U' ;
   RMAN> RESTORE SPFILE to '/u01/app/oracle/product/12.2.0.1/dbhome_1/dbs/spfileprddb.ora' FROM '/u01/backups/c-3107896219-20180327-02';
   RMAN> shutdown immediate;
   ```

7. Create a new text file in the `$ORACLE_HOME/dbs` directory named `initprddb.ora`, and add the following entry (which points to the spfile that you just restored):

   ```
   SPFILE=/u01/app/oracle/product/12.2.0.1/dbhome_1/dbs/spfileprddb.ora
   ```

8. Modify the initialization parameters for the target.

   ```
   sqlplus / as sysdba
   SQL> alter system set db_create_file_dest= '+DATA' scope=spfile;
   SQL> alter system set db_recovery_file_dest= '+RECO' scope=spfile;
   SQL> alter system set db_recovery_file_dest_size=4385144832 scope=spfile;
   SQL> alter system set control_files= '+DATA/prddb/controlfile/control01.ctl', '+RECO/prddb/controlfile/control02.ctl' scope=spfile;
   shutdown immediate;
   startup nomount;
   ```
9. **Restore the controlfile.** In this command, you are using the DBID that you queried from the source database in step 3.

```
rman target /
RMAN> set dbid 3107896219
RMAN> set DECRYPTION identified by encryptit;
RMAN> SET CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '%F';
RMAN> RESTORE CONTROLFILE FROM '/u01/backups/c-3107896219-20180327-02';
RMAN> shutdown immediate;
RMAN> startup mount;
```

Now the database is mounted with the controlfile that you just restored.

10. **Restore and recover the database.**

    In this command you are using the online redo log file names that you queried in step 3. Once the restore and recovery is complete your database on the Oracle Database service is ready for use.

```
rman target /
set DECRYPTION identified by encryptit;
r
run {
set ARCHIVELOG DESTINATION to '+RECO';
set NEWNAME for database to '+DATA';
allocate channel c1 device type disk format '/u01/backups/%U' ;
SQL "ALTER DATABASE RENAME FILE '/u01/oradata/prddb/redo01.log' TO '+DATA/PRDDB/ONLINELOG/redo01.log' ";
SQL "ALTER DATABASE RENAME FILE '/u01/oradata/prddb/redo02.log' TO '+DATA/PRDDB/ONLINELOG/redo02.log' ";
SQL "ALTER DATABASE RENAME FILE '/u01/oradata/prddb/redo03.log' TO '+DATA/PRDDB/ONLINELOG/redo03.log' ";
restore database;
switch datafile all;
restore archivelog all;
recover database;
alter database open resetlogs;
}
```

After the restore and recovery is completed, your database on the Oracle Cloud Infrastructure Database service is ready for use.

Oracle RMAN has the ability to back up directly to the Oracle Cloud Infrastructure Object Storage service from Amazon EC2 instances. Backing up to Object Storage has the following benefits:

- You do not have to provision Amazon Elastic Block Storage (EBS) on the source instance to store your RMAN backup.
- Object Storage provides you with unlimited storage.
You do not have to transfer the RMAN backup files to the destination as a separate step.

Restores are fast because Object Storage is in the same environment as the target database environment in Oracle Cloud Infrastructure.

You must perform the following steps to back up to Object Storage:

1. Install the Oracle Cloud backup module on your source Amazon EC2 instance.
2. Configure the Oracle Cloud backup module to back up to Object Storage.
3. Set up the RMAN backup destination by using the `SBT_TAPE` parameter to point to Object Storage.

**Oracle GoldenGate for Replication**

Oracle GoldenGate enables real-time data integration between different Oracle databases. It provides all the necessary programs to capture, transform, propagate, and apply changes happening in a source database to a target database. You can use Oracle GoldenGate to make an initial copy of the source database from Amazon RDS to Oracle Database instances on Oracle Cloud Infrastructure, and also to keep them synchronized until the actual cutover happens. This enables you to cut over to the newly migrated database with minimal downtime.

In this example, Oracle GoldenGate is installed and configured as a hub on an Oracle Cloud Infrastructure instance. This instance uses the Oracle GoldenGate extract processes to collect all the changes from a source database. The changes are stored in trail files on the hub, and the Replicat process is used to transfer and apply these changes to the target database.

The following steps show how to set up and use Oracle GoldenGate 12.3.0.1.2 as a hub instance in Oracle Cloud Infrastructure to migrate the HR schema from a 12.2.0.1 Non-CDB database to a PDB in a container database in the Oracle Cloud Infrastructure Database service.

1. Install and configure Oracle GoldenGate on an Oracle Cloud Infrastructure Compute instance.
   A. Create a Compute instance with the appropriate amount of CPUs and memory to serve as the hub instance. Choose Oracle Linux 7.5 as the operating system, and create and attach the required amount of block storage to the instance.
      The instance needs enough CPU power to manage the volume of transactions that need to be replicated and enough block storage attached to store all the trail files.

D. Create a user named oracle on the host and set a complex password.

2. Install the Oracle instant client on the Compute instance:

   A. Log in as oracle to the Compute instance.

   B. Unzip the contents of the instant client downloads directly to the /u01/app/oracle/product/12.2.0.1/db_1 directory. (Move all the files from the instantclient_12_2 directory to db_1.)

3. Install GoldenGate on the Compute instance:

   A. Log in as oracle to the Compute instance.

   B. Unzip the 123012_fbo_ggs_Linux_x64_shiphome.zip file into a staging directory (for example, /u01/tmp).

   C. Create a response file (for example, /u01/tmp/gg.rsp) with the following entries:

```bash
oracle.install.responseFileVersion=/oracle/install/rspfmt_ogginstall_response_schema_v12_1_2
INSTALL_OPTION=ORA12c
SOFTWARE_LOCATION=/u01/app/oracle/product/gg
START_MANAGER=false
INVENTORY_LOCATION=/u01/app/oraInventory
UNIX_GROUP_NAME=oracle
```

   D. Change directory to /u01/tmp/fbo_ggs_Linux_x64_shiphome/Disk1.

   E. Run the following command:

   ```bash
   runInstaller -silent -nowait -responseFile /u01/tmp/gg.rsp
   ```

   When prompted, run orainstRoot.sh.

   F. Set the following parameters in .bash_profile:

```bash
export GGHOME=/u01/app/oracle/product/gg
export ORACLE_HOME=/u01/app/oracle/product/12.2.0.1/db_1
export LD_LIBRARY_PATH=$ORACLE_HOME:$LD_LIBRARY_PATH
export TNS_ADMIN=/u01/app/oracle/product/12.2.0.1/db_1/network/admin
export PATH=$PATH:$HOME/.local/bin:$HOME/bin:$GGHOME
```
4. Configure the Oracle GoldenGate hub.

A. Add entries to the $ORACLE_HOME/network/admin/tnsnames.ora file to connect to the Amazon RDS source database instance.

```plaintext
HRcomp = (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = rdsdb.chyqlovrryi.us-east-2.rds.amazonaws.com)(PORT = 1521))
    (CONNECT_DATA =
        (SERVICE_NAME = prddb3)
    )
)
HRDbaaS = (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = 10.0.0.62)(PORT = 1521))
    (CONNECT_DATA =
        (SERVICE_NAME = prdpdb3.sub04231321440.prdvcn1.oraclevcn.com)
    )
)
```

B. Execute GGSCI and create the necessary directories.

```plaintext
cd $GGHOME
./ggsci
GGSCI> create subdirs
GGSCI> edit param mgr
PORT 7865
GGSCI> start mgr
GGSCI> add credentialstore
GGSCI> alter credentialstore add user ggadmin@hrcomp password ggadmin alias ggadminc
GGSCI> alter credentialstore add user ggadmin@hrdbaas password Welcome#_12 alias ggadmin
GGSCI> quit
```

5. Prepare the source database in AWS.

```plaintext
sqlplus sys/manager@hrcomp as sysdba
alter database add supplemental log data;
alter system set enable_goldengate_replication=true scope=both;
```

6. Create the Oracle GoldenGate user in the source database.

```plaintext
create tablespace ggs_data datafile '/u01/oradata/prddb3/ggs_data01.dbf' size 1024m autoextend on;
create user ggadmin identified by ggadmin default tablespace ggs_data temporary tablespace temp;
```
7. Grant the user the necessary privileges.

```sql
grant connect, resource, create session, alter session to ggadmin;
grant select any dictionary, select any table, create table to ggadmin;
grant alter any table to ggadmin;
grant execute on utl_file to ggadmin;
@marker_setup.sql (Specify ggadmin as the user)
@ddl_setup.sql (Specify ggadmin as the user)
@role_setup.sql (Specify ggadmin as the user)
@ddl_enable
@sequence (Specify ggadmin as the user)
exec dbms_goldengate_auth.grant_admin_privilege('GGADMIN');
grant ggs_ggsuser_role to ggadmin;
```

Log in to the Compute instance in AWS where the source database instance resides.

```sql
sqlplus / as sysdba
shutdown immediate
startup mount
alter system set db_recovery_file_dest_size=20G scope=both;
alter system set db_recovery_file_dest='/u01/orareco' scope=both;
alter database flashback on;
alter database open;
alter database add supplemental log data;
alter database force logging;
grant execute on dbms_logmnr_d to GGADMIN;
grant select on sys.logmnr_buildlog to GGADMIN;
grant execute on utl_file TO GGADMIN;
grant exempt access policy to GGADMIN;
```

8. Configure the target database instance in Oracle Cloud Infrastructure.

```sql
sqlplus sys/manager@hrdbaas as sysdba
alter database add supplemental log data;
alter system set enable_goldengate_replication=true scope=both; (Run this as cdb)
create tablespace ggs_data datafile '+DATA' size 1024m autoextend on;
create user ggadmin identified by Welcome#_12 default tablespace ggs_data
temporary tablespace temp;
grant connect, resource, create session, alter session to ggadmin;
grant select any dictionary, select any table, create table to ggadmin;
grant alter any table to ggadmin;
grant execute on utl_file to ggadmin;
@marker_setup.sql (Specify ggadmin as the user)
@ddl_setup.sql (Specify ggadmin as the user) -- Does not work in pdb
@role_setup.sql (Specify ggadmin as the user)
@ddl_enable -- Does not work in pdb
@sequence (Specify ggadmin as the user)
```
exec dbs_goldengate_auth.grant_admin_privilege('GGADMIN');
grant ggs_ggsuser_role to ggadmin;
@ddl_enable

Log in to the host where the target database service instance resides.

export ORACLE_UNQNAME=prddb3_iad16p
sqlplus / as sysdba
shutdown immediate
startup mount
alter system set db_recovery_file_dest_size=20G scope=both;
alter system set db_recovery_file_dest='/u01/orareco' scope=both;
alter database flashback on;
alter database open;
alter database add supplemental log data;
alter database force logging;
alter session set container=prdpdb3
grant EXECUTE on dbms_logmnr_d to GGADMIN;
grant SELECT on sys.logmnr_buildlog to GGADMIN;
grant execute on utl_file to GGADMIN;
grant exempt access policy to GGADMIN;

9. Configure and start the GoldenGate extract from the target database. Log in to the Oracle Cloud Infrastructure Compute instance as the oracle user.

cd $GGHOME
./ggsci

GGSCI> dblogin useridalias ggadminc
GGSCI> add schematrandata hrcomp ALLCOLS
GGSCI> edit params extehrc
extract extehrc
extrail ./dirdat/ee
tranlogoptions IntegratedParams (max_sga_size 256)
discardfile ./dirrp/silext01.dsc, append megabytes 50
logallsupcols
updaterecordformat compact
reportcount every 2 hours, rate
useridalias ggadminc
table HR.*;

GGSCI> register extract extehrc database
GGSCI> add extract extehrc, integrated tranlog, begin now
GGSCI> add extrail ./dirdat/ee, extract extehrc
GGSCI> start extract extehrc

The register extract step displays a system change number (SCN). Note that number.
10. Prepare the target database for the Data Pump import. Log in to the Oracle Cloud Infrastructure Database service host, where the target database resides, as the operating system user oracle.

    sqlplus / as sysdba
    SQL> create user hr identified by rimjhimgire default tablespace hrts
temporary tablespace temp;
    SQL> GRANT CREATE SESSION, CREATE VIEW, ALTER SESSION, CREATE SEQUENCE TO hr;
    SQL> GRANT CREATE SYNONYM, CREATE DATABASE LINK, RESOURCE, UNLIMITED TABLESPACE TO hr;

11. Export the HR schema from the source database in AWS.

    expdp system/manager schemas=HR directory=dpdir dumpfile=hr.dmp
    logfile=dpdir:hr.log flashback_scn=1625157

    Here the flashback_scn is the SCN that you got from the register extract step -1.

12. Import the HR schema into the target database in Oracle Cloud Infrastructure.

    impdp 'system/WelCome#_12@prddb3:1521/prdpdb3.sub04231321440.prdvcn1.oraclevcn.com' directory=dpdir dumpfile=hr.dmp schemas=hr table_exists_action=replace
    logfile=dpdir:hrimport.log

13. Start the Replicat process on the target database. Log in to the Oracle Cloud Infrastructure Compute instance as the oracle user.

    cd $GGHOME
    ./gsi
    GGSCI> dblogin useridalias ggadmind
    GGSCI> add schematrandata hr allcols
    GGSCI> add replicat rephrd integrated exttrail ./dirdat/ee
    GGSCI> edit params rephrd
    replicat rephrd
    ASSUMETARGETDEFS
    DISCARDFILE ./dirrpt/rephrd01.dsc
    DDL INCLUDE ALL
    USERIDALIAS ggadmind
    REPORTCOUNT EVERY 1 HOURS, RATE
    MAP HR.*, TARGET HR.*;
    GGSCI> start replicat rephrd
GoldenGate replicates all the transactions that have been executed on the source database to the target database. When you are ready to cut over to the database in Oracle Cloud Infrastructure, you can simply stop the source database, stop the GoldenGate replication, and start transacting on the target database.

Post Migration Steps: Transparent Database Encryption

All databases deployed on Oracle Cloud Infrastructure should use transparent data encryption (TDE) to encrypt data at rest. Any tablespaces that are migrated from AWS should be converted to use TDE if the tablespaces were unencrypted on the source.

- If you are using a logical migration method like Data Pump, you can set up the tablespaces with TDE before the data is imported.
- If you are performing a physical migration like RMAN backup restore, the migration to TDE can be accomplished in two ways. You can perform an offline conversion to TDE or, if you have a Data Guard environment, you can perform an online conversion to TDE. The details of this process are beyond the scope of this document and can be found in the [Converting to Transparent Data Encryption with Oracle Data Guard using Fast Offline Conversion](#) white paper.

Conclusion

By migrating your Oracle databases to the Oracle Cloud Infrastructure Database service, you can take advantage of a world class database running on an Enterprise-class cloud platform. You get the best price performance combined with automated management capabilities and the flexibility to use all the database features and options. This white paper provided 17 migration methods, and guidance on when to use them, for migrating Oracle databases from AWS to Oracle Cloud Infrastructure. By following the steps outlined in this paper, you can migrate large databases from AWS to Oracle Cloud Infrastructure with minimal downtime.

References

- [Overview of Oracle Cloud Infrastructure](#)
- [Overview of Oracle Cloud Infrastructure Networking](#)
- [Overview of Oracle Cloud Infrastructure Security Lists](#)
- [Overview of Oracle Cloud Infrastructure Route Tables](#)
- [Overview of Oracle Cloud Infrastructure Load Balancing](#)
- [Overview of Oracle Cloud Infrastructure Database Service](#)
- Overview of Oracle Cloud Infrastructure Compute Service
- Oracle Cloud Infrastructure Virtual Cloud Network Overview and Deployment Guide
- Amazon RDS for Oracle Database