Oracle Database@AWS User Guide

Oracle Database@AWS

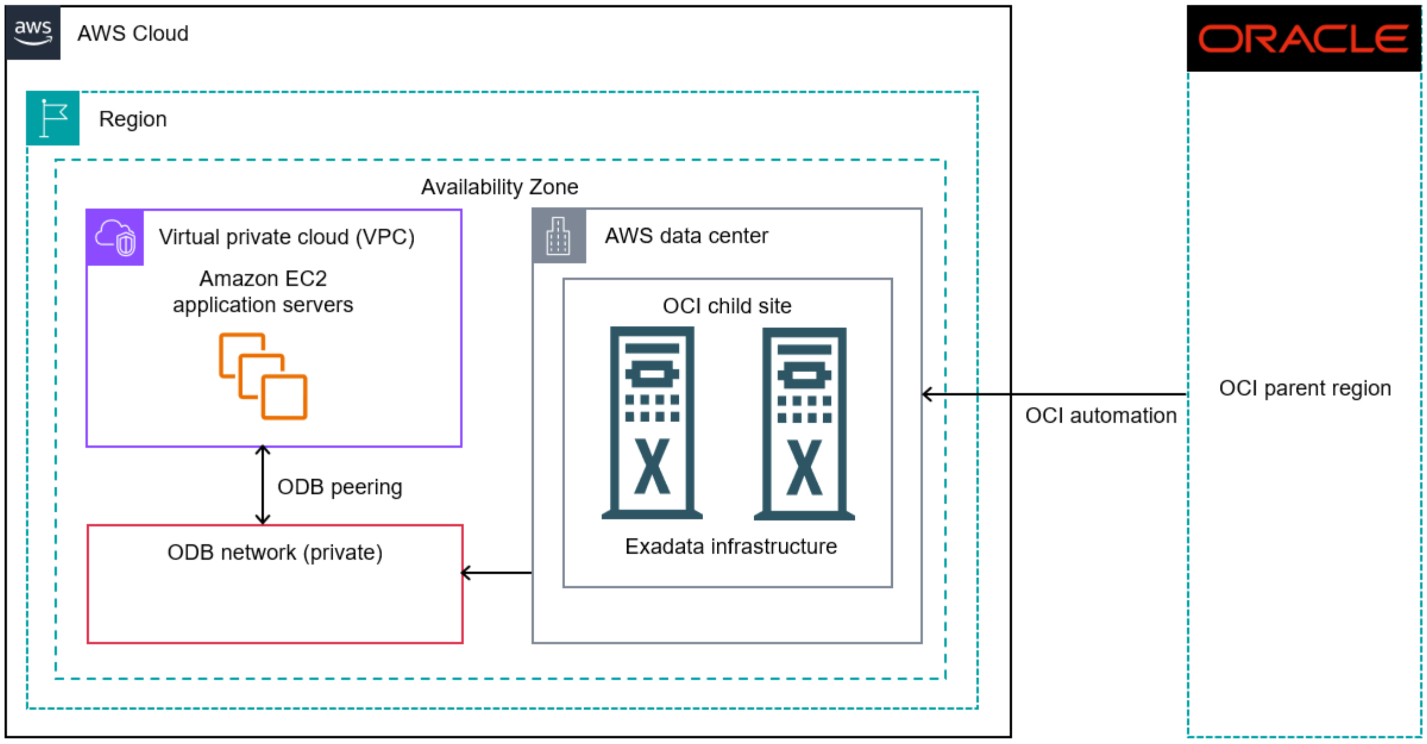
Oracle Database@AWS: Oracle Database@AWS User Guide

Table of Contents

What is Oracle Database@AWS?

Oracle Database@AWS is an offering that enables you to access Oracle Exadata infrastructure managed by Oracle Cloud Infrastructure (OCI) inside AWS data centers. You can migrate your Oracle Exadata workloads, establish low-latency connectivity with applications running on AWS, and integrate with AWS services. You get a single invoice through AWS Marketplace, which counts towards AWS commitments and Oracle Support rewards.

The following diagram shows a high-level overview of an OCI region tied to an AWS data center that hosts Oracle Exadata infrastructure. Within an AWS Availability Zone (AZ), you can establish one or more peering connections (up to 45) between your Amazon VPCs and the private network that is tied to the data center. By peering these networks, application servers in the VPCs can access Oracle databases running on the Oracle Exadata infrastructure.



Features of Oracle Database@AWS

With Oracle Database@AWS, you benefit from the following features:

Migration of Oracle Exadata database workloads to AWS

With Oracle Database@AWS, you can easily migrate your Oracle Exadata workloads to Oracle Exadata Database Service on Dedicated Infrastructure or Oracle Autonomous Database on Dedicated Exadata Infrastructure within AWS. The migration offers minimal changes, full feature availability, architectural compatibility, and the same performance as on-premises Exadata deployments. You can use standard Oracle database migration tools such as Recovery Manager (RMAN), Oracle Data Guard, transportable tablespaces, Oracle Data Pump, Oracle GoldenGate, AWS Database Migration Service, and Oracle Zero Downtime Migration.

Reduced application latency

You can establish low-latency connectivity between Oracle Exadata and applications running on AWS. Proximity to applications hosted in AWS ensures minimal network delays and improved performance.

Innovation through data unification

You can generate deeper insights and develop new innovation by using zero-ETL integrations to unify your data across Oracle and AWS for analytics, machine learning, and generative AI. With zero-ETL integration using Amazon Redshift, you can enable near real-time analysis and machine learning (ML) on transactional data stored in Oracle Database@AWS.

Simplified management and operations

You can benefit from a unified experience between Oracle and AWS with collaborative support, purchasing, management, and operations. Your usage of Oracle Database services qualifies for your existing AWS commitments and Oracle license benefits, such as Oracle Support Rewards. You can use familiar AWS tools and interfaces to purchase, provision, and manage your Oracle Database@AWS resources. You can provision and manage your resources using AWS APIs, CLI, or SDKs. The AWS APIs call the corresponding OCI APIs necessary to provision and manage the resources.

Seamless integration with AWS services

You can integrate with other AWS services and applications running in the same environment. For example, Oracle Database@AWS integrates with Amazon EC2, Amazon VPC, and IAM. You can also integrate Oracle Database@AWS with AWS services such as Amazon CloudWatch for monitoring and Amazon EventBridge for event management. For database backups, you can use Amazon S3, which is designed to exceed 11 9s of durability.

Related AWS services

Oracle Database@AWS works with the following services to improve the availability and scalability of your Oracle database applications:

- **Amazon EC2** — Provides virtual servers that function as Oracle application servers. You can configure your load balancer to route traffic to your EC2 application servers. For more information, see the Amazon EC2 User Guide.
- **Amazon Virtual Private Cloud (VPC)** — Enables you to launch AWS resources in a logically isolated virtual network that you've defined. Oracle Exadata infrastructure resides in a special network called the ODB network that you can peer to a VPC. You can then run application servers in your VPC and access your Exadata databases. For more information, see the Amazon VPC User Guide.
- **Amazon VPC Lattice** — Provides native access to AWS services such as Amazon S3 and Oracle managed backups from the ODB network. For more information, see the What is Amazon VPC Lattice?.
- **Amazon CloudWatch** — Provides a monitoring service for Oracle Database@AWS. OCI gathers metric data about your Oracle Exadata system and sends it to CloudWatch. For more information, see Monitoring Oracle Database@AWS with Amazon CloudWatch.
- **AWS Identity and Access Management (IAM)** — Helps you securely control access to Oracle Database@AWS resources for your users. Use IAM to control who can use your AWS resources (authentication) and what resources users can use in which ways (authorization). For more information, see Identity and access management for Oracle Database@AWS.
- **AWS analytics services** — Provide a broad and cost-effective set of analytics services to help you gain insights faster from your Exadata database. Each service is purpose-built for a wide range of analytics use cases such as interactive analysis, big data processing, data warehousing, real-time analytics, operational analytics, dashboards, and visualizations. For more information, see Analytics on AWS.

Accessing Oracle Database@AWS

You can create, access, and manage Oracle Database@AWS using the AWS Management Console. It provides a web interface that you can use to access Oracle Database@AWS.

Pricing for Oracle Database@AWS

You can purchase Oracle Database@AWS offerings from AWS Marketplace. You first contact an Oracle sales representative. Oracle then makes the offer available to you in AWS Marketplace based on the private pricing agreement. Your AWS bill shows charges based on your usage.

There are no data transfer charges when your Oracle application and Oracle database are hosted in the same Availability Zone (AZ). Standard data transfer charges apply for communication between AZs.

When using Oracle Database@AWS managed integrations such as zero-ETL, Oracle managed backups, and Amazon S3, standard data processing charges for sharing and accessing resources through VPC Lattice apply. There is no hourly charge for Oracle Database@AWS managed integrations. For more information, see Amazon VPC Lattice pricing.

What's next?

You're now ready to begin creating your Oracle Database@AWS resources.

1. Learn about how Oracle Database@AWS works. For more information, see How Oracle Database@AWS works.

   Note

   If you're familiar with AWS and Oracle Exadata and want to get started right away, skip this step.

2. Request a private offer for Oracle Database@AWS through the AWS Management Console, and then accept the offer. For more information, see Request a private offer for Oracle Database@AWS.

   Note

   To request a private offer in this preview, you must contact AWS to get your AWS account added to an allow list.

3. Create your ODB network, Oracle Exadata infrastructure, and Exadata VM clusters using the AWS console. Create your Exadata databases using OCI tools. For more information, see Getting started with Oracle Database@AWS.

4. Share your resources across accounts with AWS Resource Access Manager (AWS RAM). For more information, see Working with shared Oracle Database@AWS resources in a trusted account.

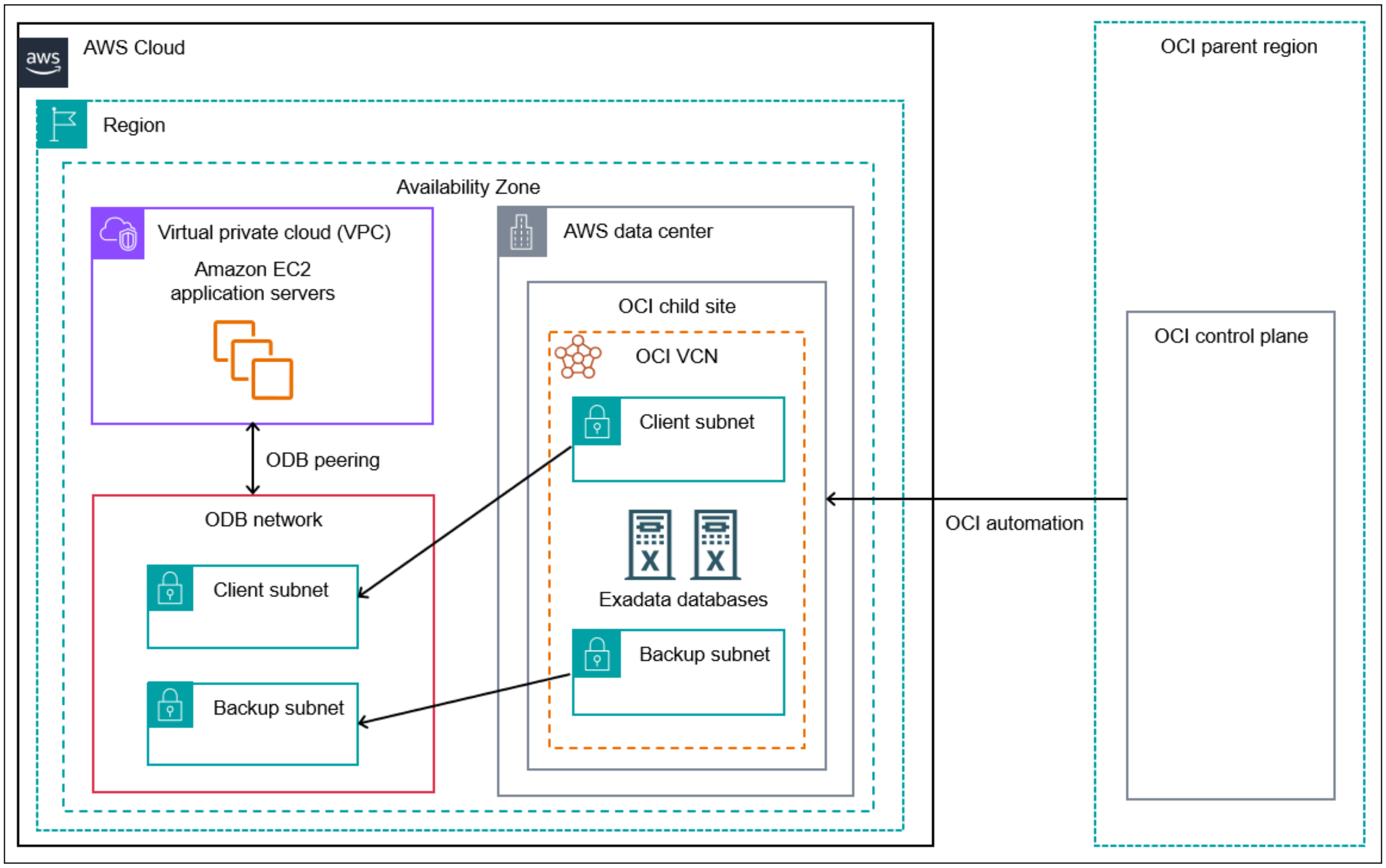
How Oracle Database@AWS works

Oracle Database@AWS integrates Oracle Cloud Infrastructure (OCI) with the AWS Cloud. In the following sections, you can learn about the key components of this multicloud architecture.

Oracle Exadata Database Service on Dedicated Infrastructure is an OCI service that provides Exadata Database Machine. Oracle Exadata Database Machine is an integrated, preconfigured, and pretested full-stack platform for use in enterprise data centers. You can create the Oracle Exadata infrastructure and VM clusters in an AWS Availability Zone (AZ) using the AWS console, CLI, or APIs.

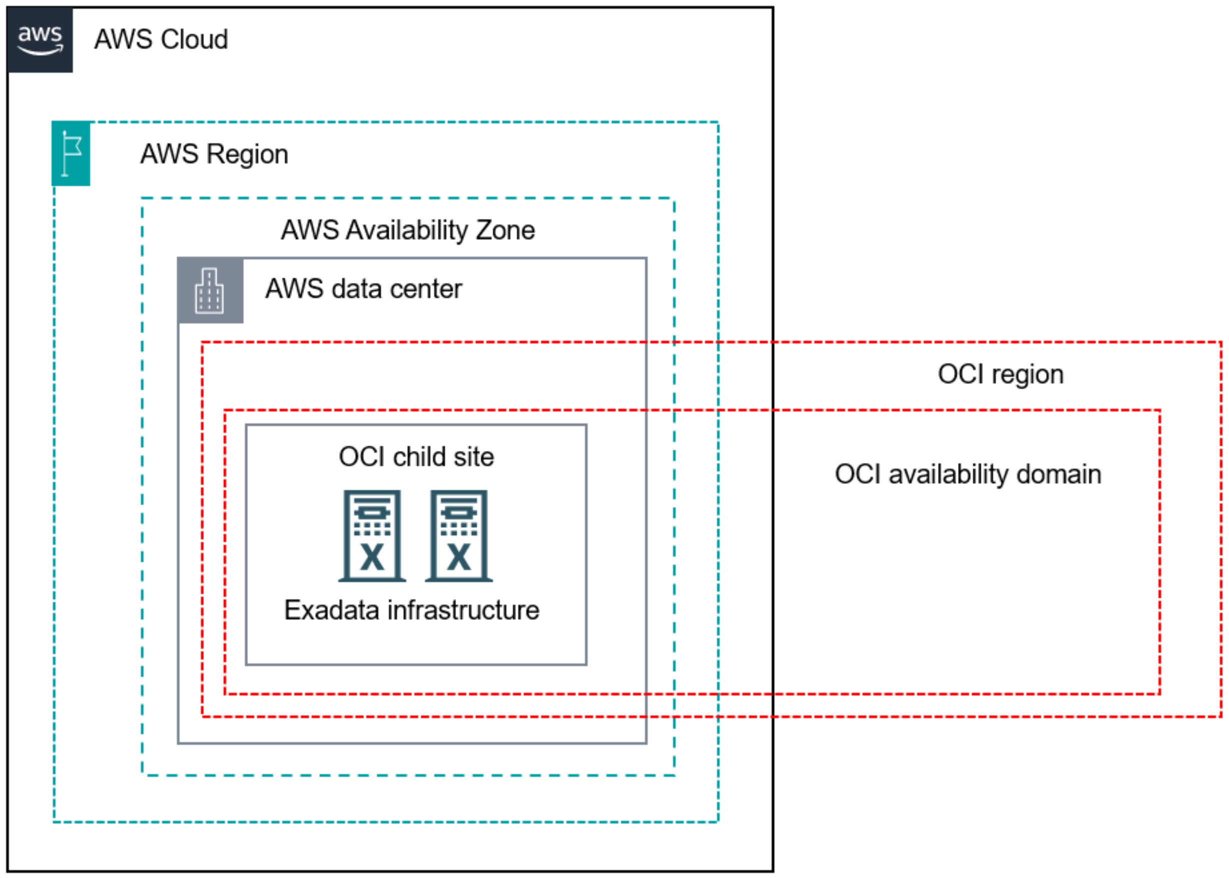
After you have created your resources in AWS, you use OCI APIs to create and manage Oracle Exadata databases. An ODB network, which you peer to an Amazon VPC, enables Amazon EC2 application servers to access your Exadata databases. In this way, Oracle Exadata databases are integrated into the AWS environment.

The following diagram shows the Oracle Database@AWS architecture.



OCI child sites

Oracle Cloud Infrastructure is hosted in OCI regions and availability domains. An *OCI region* consists of *OCI availability domains (ADs)*, which are isolated data center clusters within an OCI region. An *OCI child site* is a data center that extends an OCI availability domain to an Availability Zone (AZ) in an AWS Region. The Exadata infrastructure logically resides in an OCI region and physically resides in an AWS Region.



The OCI child site for Oracle Database@AWS physically resides in an AWS data center. AWS hosts the Exadata infrastructure, and OCI provisions and maintains the Exadata infrastructure hardware inside the data center. You can configure the Exadata infrastructure, private network, and VM clusters using the AWS console, CLI, or APIs. You can use AWS services such as Amazon EC2 and Amazon VPC to allow application access to Oracle Exadata databases running on the infrastructure.

Oracle Exadata infrastructure

The Oracle Exadata infrastructure is the underlying architecture of database servers and storage servers that runs Oracle Exadata databases. The infrastructure resides in an AWS Availability Zone (AZ). To create VM clusters on Exadata infrastructure, you use the AWS console, CLI, or APIs.

The Oracle Exadata infrastructure is distributed on physical machines called *database servers*. These servers provide the compute resources, similar to Amazon EC2 dedicated servers. Each database server hosts one or more virtual machines (VMs) running on a hypervisor. For architectural diagrams that illustrate these relationships, see Exadata Database Service on Dedicated Infrastructure Technical Architecture.

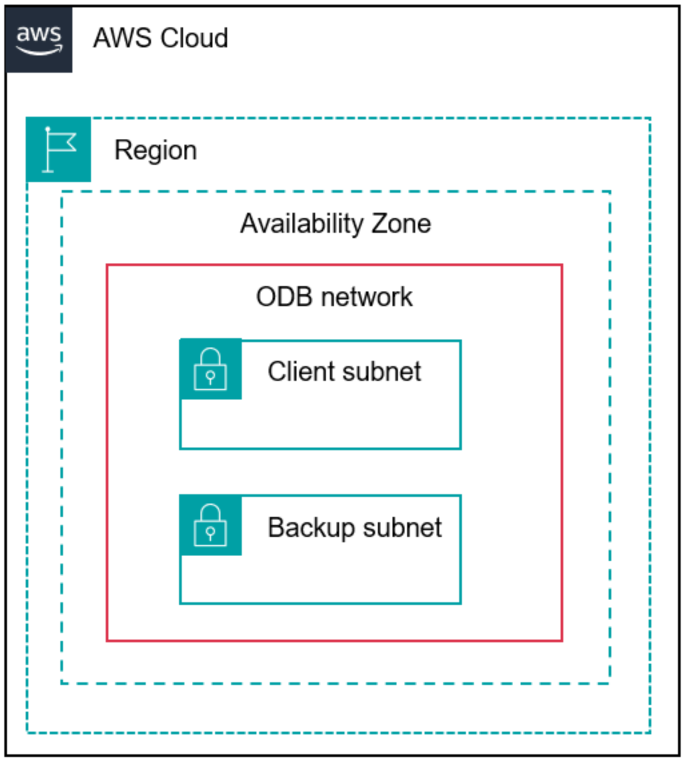
When you create Exadata infrastructure in Oracle Database@AWS, you specify information such as the following:

- The total number of database servers
- The total number of storage servers
- The Exadata system model (X11M)
- The AZ that hosts the infrastructure (see Supported Regions for Oracle Database@AWS)

To learn how to create Oracle Exadata infrastructure, see Step 2: Create an Oracle Exadata infrastructure in Oracle Database@AWS.

ODB network

An *ODB network* is a private isolated network that hosts OCI infrastructure in an AWS Availability Zone (AZ). The ODB network consists of a CIDR range of IP addresses. The ODB network maps directly to the network that exists within the OCI child site, thus serving as the means of communication between AWS and OCI. You must specify an ODB network when you create your Exadata VM clusters (see Step 3: Create an Exadata VM cluster or Autonomous VM cluster in Oracle Database@AWS).



You provision resources in an ODB network using Oracle Database@AWS APIs. The ODB network is managed by AWS, but you can set up an ODB peering connection to connect an Amazon VPC to the ODB network. For more information, see enODB peering.

When you create an ODB network, you specify information such as the following:

- Availability Zone — The ODB network is specific to an AZ.

  You can use Oracle Database@AWS in the following AWS Regions:

  US East (N. Virginia)

  You can use the AZs with the physical IDs `use1-az4` and `use1-az6`.

  US West (Oregon)

  You can use the AZs with the physical IDs `usw2-az3` and `usw2-az4`.

  Asia Pacific (Tokyo)

  You can use the AZs with the physical IDs `apne1-az1` and `apne1-az4`.

  US East (Ohio)

  You can use the AZs with the physical IDs `use2-az1` and `use2-az2`.

Europe (Frankfurt)

You can use the AZs with the physical IDs `euc1-az1` and `euc1-az2`.

Canada (Central)

You can use the AZ with the physical ID `cac1-az4`.

Asia Pacific (Sydney)

You can use the AZ with the physical ID `apse2-az4`.

Europe (Ireland)

You can use the AZ with the physical ID `euw1-az3`.

To find the logical AZ names in your account that map to the preceding physical AZ IDs, run the following command.

```
aws ec2 describe-availability-zones \
  --region us-east-1 \
  --query "AvailabilityZones[*].{ZoneName:ZoneName, ZoneId:ZoneId}" \
  --output table
```

- Client CIDR addresses — The ODB network requires a client subnet CIDR for Exadata VM clusters and Autonomous VM clusters.
- Backup CIDR addresses — The ODB network requires a backup subnet CIDR for managed database backups of VM clusters. The backup subnet is optional for Exadata VM clusters.
- AWS service integrations — You can configure a network path for AWS service integrations such as Amazon S3 and zero-ETL with Amazon Redshift. For more information, see AWS service integrations.

For more information, see Step 1: Create an ODB network in Oracle Database@AWS.

Virtual Private Cloud (VPC)

A *Virtual Private Cloud (VPC)* is a virtual network that you create in the AWS cloud. It is logically isolated from other virtual networks in the AWS cloud, providing you with complete control over the virtual networking environment, including selection of your own IP address range, creation of subnets, and configuration of route tables and network gateways. For more information, see What is Amazon VPC?

You can launch Amazon EC2 instances into your Amazon VPC. The EC2 instances can host application servers that communicate with Oracle Exadata databases. You can manage and launch the application servers just like any other EC2 instances in your VPC. For more information, see What is Amazon EC2?

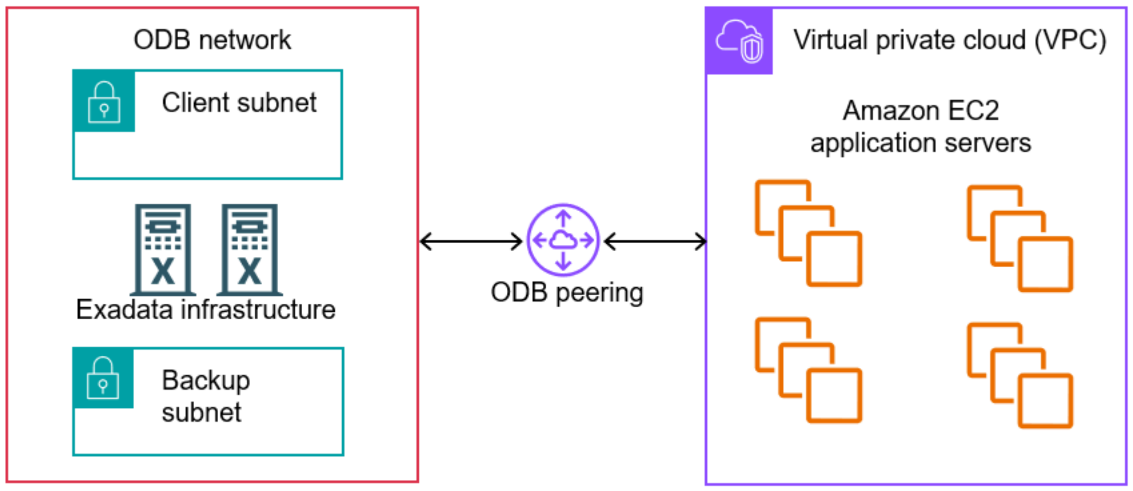
By default, the ODB network doesn't have connectivity to VPCs. To connect the ODB network to your existing AWS infrastructure, create one or more peering connections (up to 45) between the ODB network and your VPCs. For more information, see Step 1: Create an ODB network in Oracle Database@AWS.

ODB peering

ODB peering is a user-created network connection that enables traffic to be routed privately between an Amazon VPC and an ODB networkAfter peering, an Amazon EC2 instance within the VPC can communicate with an Oracle Exadata database in the ODB network as if they were within the same network.

Note

ODB peering is different from VPC peering, which is a peering connection between two VPCs that routes traffic between them.



You can peer an ODB network in one account and a VPC in another account using AWS RAM. If you share an ODB network with another account, the trust account can directly initiate peering. The account that initiates the ODB peering connection owns and manages the connection.

You can specify peer network CIDRs when you create or update ODB peering connections. In this way, you control which subnets in the peer VPC have access to your ODB network. A VPC account can update the CIDR ranges without also owning the ODB network. For more information, see Configuring ODB peering to an Amazon VPC in Oracle Database@AWS.

Resources in a VPC can span Availability Zones (AZs). In an ODB network, resources are bound to a single AZ. You define this AZ when you create the ODB network.

Creation of an ODB peering connection

An ODB peering connection isn't a characteristic of an ODB network but is an independent resource with its own ID (prefixed with `odbpcx-`) and lifecycle. You manage a peering connection with a set of dedicated APIs. For example, you create an ODB peering connection to an existing ODB network using the Oracle Database@AWS console or the `CreateOdbPeeringConnection` API. For more information, see Creating an ODB peering connection in Oracle Database@AWS.

When you create an ODB peering connection, Oracle Database@AWS performs the following actions automatically:

1. Validates the network configurations, including checking for overlapping CIDR blocks with the Oracle VCN CIDR
2. Sets up the underlying network peering infrastructure
3. Configures the ODB network (not the VPC) route tables with the VPC CIDR addresses

After you create your ODB peering connection, update your VPC route tables manually using the Amazon EC2 `create-route` command. For more information, see Configuring VPC route tables for ODB peering.

AWS service integrations

To provide enhanced functionality and connectivity options for your Oracle databases, Oracle Database@AWS integrates with AWS services using Amazon VPC Lattice. You can configure network paths to AWS services directly from your ODB network without requiring additional VPCs or complex networking setups.

Oracle Database@AWS supports the following AWS managed service integrations:

Amazon S3

You can integrate Amazon S3 with Oracle Database@AWS in the following ways:

- Oracle managed automatic backups to Amazon S3 – Oracle Database@AWS automatically enables network access for automatic backups. This integration can't be disabled. If you set Amazon S3 as your managed backup target in the OCI console, then OCI uploads automatic backups to an S3 bucket.
- Direct access to Amazon S3 from your ODB network – You can enable direct ODB network access to S3 and then store scripts, import and export files, and related files in an S3 bucket. You can disable this access. This setting is independent of the automatic network access for Oracle managed automatic backups.

Zero-ETL integration with Amazon Redshift

You can enable zero-ETL integration of your ODB network with Amazon Redshift. This integration enables you to replicate data to Amazon Redshift from your Oracle databases running in Oracle Database@AWS without the traditional extract, transform, and load (ETL) process. This integration enables real-time analytics and AI workloads by automatically synchronizing your Oracle data with Amazon Redshift.

In addition to managed integrations for AWS services, you can also use VPC Lattice to access services and resources hosted in other VPCs, or access ODB network instances from your VPC. You can manage access and resources using the VPC Lattice console, CLI, and APIs. For more information, see the following resources:

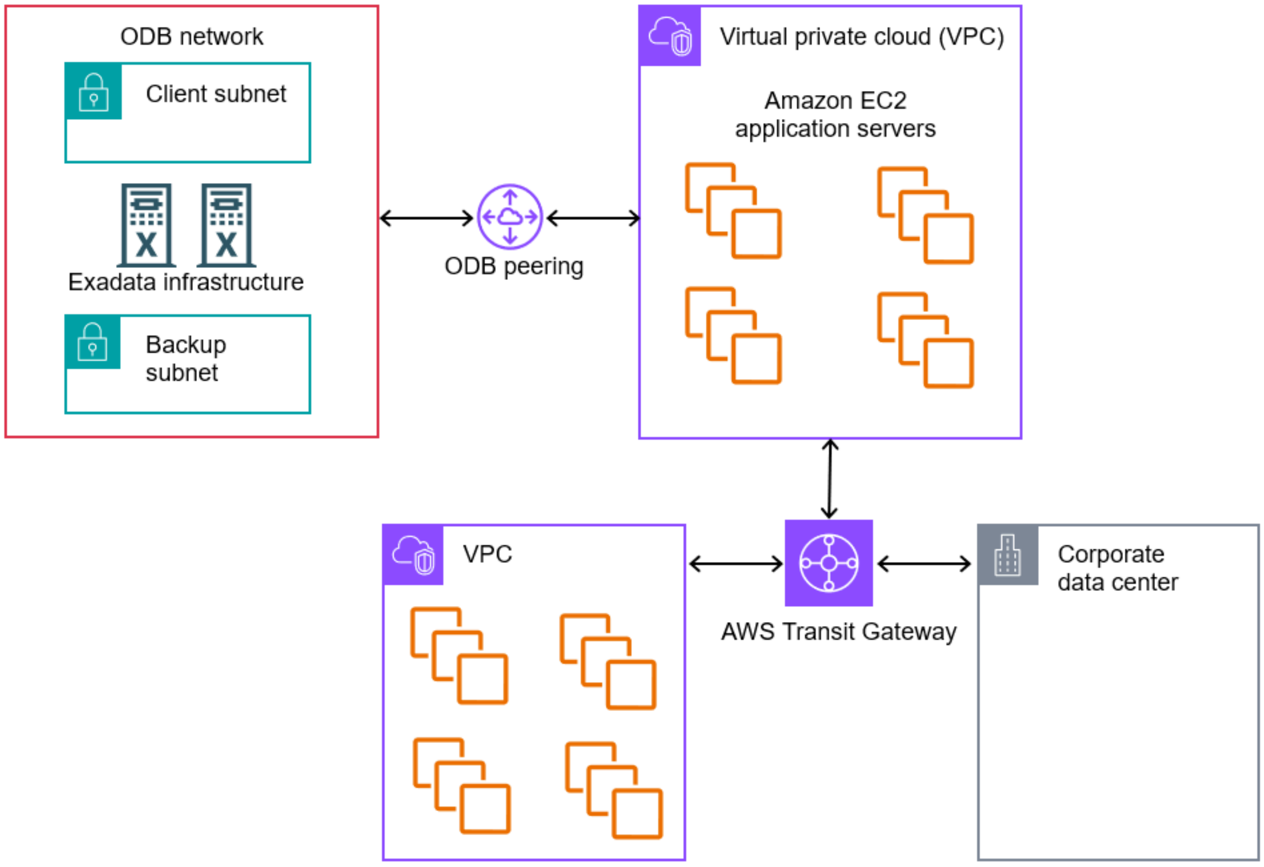
- Backing up in Oracle Database@AWS
- Oracle Database@AWS Zero-ETL integration with Amazon Redshift
- What is Amazon VPC Lattice? and VPC Lattice for Oracle Database@AWS

Routing traffic from multiple VPCs

To allow multiple VPCs to access Oracle Database@AWS resources in one ODB network, you can use AWS Transit Gateway or AWS Cloud WAN.

AWS Transit Gateway

An Amazon VPC transit gateway is a network transit hub used to interconnect VPCs and on-premises networks. An ODB network supports up to 45 direct peering connections. You can establish direct peering connections between your ODB network and multiple VPCs, or use a transit gateway for centralized routing. To use a transit gateway, peer your ODB network to a VPC and then attach this VPC to the transit gateway. The gateway can connect to multiple VPCs. With this transit gateway configuration, you can route traffic between multiple VPC subnets and your ODB network through a central hub.



For more information, see Configuring Amazon VPC Transit Gateways for Oracle Database@AWS.

AWS Cloud WAN

AWS Cloud WAN is a managed wide-area networking (WAN) service that enables you to build, manage, and monitor a unified global network connecting resources across your cloud and on-premises environments. Using the central dashboard, you can connect on-premises branch offices, data centers, and VPCs across the AWS global network.

You can peer your ODB network to a VPC, and then attach this VPC to the Cloud WAN core network. With this configuration, you can use Cloud WAN to route traffic between multiple VPCs or on-premises networks and your ODB network. For more information, see Configuring AWS Cloud WAN for Oracle Database@AWS.

Exadata VM clusters

An *Exadata VM cluster* is a set of tightly coupled Exadata VMs. Each VM has a complete Oracle database installation that includes all features of Oracle Enterprise Edition, including Oracle Real Application Clusters (Oracle RAC) and Oracle Grid Infrastructure. You can create one or more Oracle Exadata databases on a VM cluster. For diagrams that show the architecture of VMs and VM clusters, see Exadata Database Service on Dedicated Infrastructure Technical Architecture.

When you create a *VM cluster*, you specify information that includes the following:

- An ODB network
- An Oracle Exadata infrastructure
- The database servers on which to place the VMs in the cluster
- The total amount of usable Exadata storage

You can configure the CPU cores, memory, and local storage for each VM in a VM cluster. For more information, see Step 3: Create an Exadata VM cluster or Autonomous VM cluster in Oracle Database@AWS.

Autonomous VM clusters

Autonomous VM clusters are fully managed databases that automate key management tasks using machine learning and AI. Unlike traditional databases, autonomous databases automatically provision, secure, update, backup, and tune the database with no human intervention required.

You can configure the ECPU core count per VM, database memory per CPU, database storage, and maximum number of autonomous container databases. For more information, see Step 3: Create an Exadata VM cluster or Autonomous VM cluster in Oracle Database@AWS.

Oracle Exadata databases

Oracle Exadata is an engineered system that provide a high-performance platform for running Oracle databases. With Oracle Database@AWS, you use the AWS console to create the Oracle Exadata infrastructure and VM clusters that host the Exadata databases. You then use OCI APIs to create and manage the Oracle databases. For more information, see Step 4: Create Oracle Exadata databases in Oracle Cloud Infrastructure.

Onboarding to Oracle Database@AWS

Before you can begin using Oracle Database@AWS, make sure you're signed up for AWS and create necessary users. Then you can purchase Oracle Database@AWS from AWS Marketplace by accepting a private offer from Oracle.

Sign up for an AWS account

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call or text message and entering a verification code on the phone keypad.

   When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to a user, and use only the root user to perform tasks that require root user access.

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to https://aws.amazon.com/ and choosing **My Account**.

Create a user with administrative access

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the AWS Management Console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

For help signing in by using root user, see Signing in as the root user in the *AWS Sign-In User Guide*.

2. Turn on multi-factor authentication (MFA) for your root user.

   For instructions, see Enable a virtual MFA device for your AWS account root user (console) in the *IAM User Guide*.

Create a user with administrative access

1. Enable IAM Identity Center.

   For instructions, see Enabling AWS IAM Identity Center in the *AWS IAM Identity Center User Guide*.

2. In IAM Identity Center, grant administrative access to a user.

   For a tutorial about using the IAM Identity Center directory as your identity source, see Configure user access with the default IAM Identity Center directory in the *AWS IAM Identity Center User Guide*.

Sign in as the user with administrative access

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

  For help signing in using an IAM Identity Center user, see Signing in to the AWS access portal in the *AWS Sign-In User Guide*.

Assign access to additional users

1. In IAM Identity Center, create a permission set that follows the best practice of applying least-privilege permissions.

   For instructions, see Create a permission set in the *AWS IAM Identity Center User Guide*.

2. Assign users to a group, and then assign single sign-on access to the group.

   For instructions, see Add groups in the *AWS IAM Identity Center User Guide*.

Request a private offer for Oracle Database@AWS

The AWS Marketplace seller private offer feature enables you to request and receive Oracle Database@AWS pricing and EULA terms from Oracle. You negotiate pricing and terms with Oracle, and then Oracle creates a private offer for the AWS account that you designate. You accept the private offer and receive the negotiated price and terms of use. At this time, you can use the Oracle Database@AWS dashboard. When the private offer agreement reaches its expiration date, you're either moved automatically to the product's public pricing or unsubscribed from Oracle Database@AWS. For more information about private offers, see Private offers in AWS Marketplace.

To request and accept a private offer for Oracle Database@AWS

1. Sign in to the AWS Management Console.
2. Search for and then choose Oracle Database@AWS.
3. Choose **Request private offer**.

   > **Note**
   >
   > The Oracle Database@AWS dashboard isn't available until after you have accepted a private offer.

4. On the Oracle Cloud Infrastructure (OCI) site, specify details such as the region and your contact information.
5. Wait for an OCI representative to contact you and make a private offer available.
6. In the AWS Management Console, choose **View private offer**.
7. Choose the offer and then choose **View offer**.
8. Choose **Create contract** and respond to the subsequent prompts to accept the private offer.
9. After accepting the private offer, you'll need to activate your OCI account. You can access the Oracle activation links directly from AWS Management Console.

   1. In the console, navigate to the **Get started** section.
   2. Click on the Oracle activation link provided in the console. Alternatively, you can also use the activation link sent to you via email.
   3. On the Oracle activation page, choose whether to create a new Oracle cloud account or add to an existing account.
   4. Complete the activation process by following the on-screen instructions.

    5. After submitting your activation request, you'll see an **Activation in progress** status in the AWS Management Console, and the dashboard will be temporarily disabled with a reason displayed.
    6. After activation is complete, the Oracle Database@AWS dashboard becomes available, allowing you to manage your resources.

10. In the AWS Management Console, choose **Dashboard**.

Subscribe to Oracle Database@AWS in multiple Regions

When you subscribe to Oracle Database@AWS through AWS Marketplace and finish onboarding, your AWS account is linked to your OCI tenancy. This link, along with related resources, is automatically replicated to all AWS Regions where Oracle Database@AWS is available. You subscribe and onboard once rather than repeating the process for each Region.

To use Oracle Database@AWS in multiple Regions, perform the following steps:

1. Subscribe to Oracle Database@AWS through AWS Marketplace and complete the onboarding process.

   When you first subscribe to Oracle Database@AWS, your account is activated in a home Region. You specify the home Region in Oracle Cloud Infrastructure (OCI).
2. Enable your preferred Regions through the OCI console.

   If you don't enable a Region in OCI, and then you switch to this Region in the Oracle Database@AWS console, you receive an error stating that you haven't subscribed. In this case, you must enable this Region in OCI before you can use the Oracle Database@AWS dashboard in this Region.
3. Access Oracle Database@AWS in any supported AWS Region without repeating the subscription process.

Getting started with Oracle Database@AWS

To begin using Oracle Database@AWS, you can create the following resources using the Oracle Database@AWS console, CLI, or APIs:

1. ODB network
2. Oracle Exadata infrastructure
3. Exadata VM cluster or Autonomous VM cluster
4. ODB peering connection

To create Oracle Exadata databases on your infrastructure, you must use the Oracle Cloud Infrastructure (OCI) console or APIs rather than the Oracle Database@AWS dashboard. Thus, you deploy resources in two cloud environments: network and infrastructure resources are in AWS, while the database administration control plane is in OCI. For more information, see Oracle Database@AWS in the Oracle Cloud Infrastructure documentation.

Prerequisites for setting up Oracle Database@AWS

Before configuring your Oracle Exadata infrastructure, make sure that you do the following:

- Perform the steps in Onboarding to Oracle Database@AWS. You must have accepted a private offer to use Oracle Database@AWS.
- Grant your IAM principal the policy permissions listed in Allow users to provision Oracle Database@AWS resources. These permissions are necessary to use Oracle Database@AWS.

Supported OCI services on Oracle Database@AWS

Oracle Database@AWS supports the following Oracle Cloud Infrastructure (OCI) services:

- Oracle Exadata Database Service on Dedicated Infrastructure – Provides a fully managed, dedicated Exadata environment accessible within AWS. For more information, see Oracle Cloud Exadata Database Service on Dedicated Infrastructure in the OCI documentation.
- Autonomous Database on Dedicated Exadata Infrastructure – Provides a highly automated, fully managed database environment running in OCI with committed hardware and software

resources. For more information, see About Autonomous Database on Dedicated Exadata Infrastructure in the OCI documentation.

Supported Regions for Oracle Database@AWS

You can use Oracle Database@AWS in the following AWS Regions:

US East (N. Virginia)

You can use the AZs with the physical IDs `use1-az4` and `use1-az6`.

US West (Oregon)

You can use the AZs with the physical IDs `usw2-az3` and `usw2-az4`.

Asia Pacific (Tokyo)

You can use the AZs with the physical IDs `apne1-az1` and `apne1-az4`.

US East (Ohio)

You can use the AZs with the physical IDs `use2-az1` and `use2-az2`.

Europe (Frankfurt)

You can use the AZs with the physical IDs `euc1-az1` and `euc1-az2`.

Canada (Central)

You can use the AZ with the physical ID `cac1-az4`.

Asia Pacific (Sydney)

You can use the AZ with the physical ID `apse2-az4`.

Europe (Ireland)

You can use the AZ with the physical ID `euw1-az3`.

To find the logical AZ names in your account that map to the preceding physical AZ IDs, run the following command.

```
aws ec2 describe-availability-zones \
  --region us-east-1 \
  --query "AvailabilityZones[*].{ZoneName:ZoneName, ZoneId:ZoneId}" \
  --output table
```

Planning IP address space in Oracle Database@AWS

Plan carefully for IP address space in Oracle Database@AWS. Consider the IP address consumption based on the number of VM clusters, including the number of VMs per cluster that you can provision into the ODB network. For more information, see ODB Network Design in the Oracle Cloud Infrastructure cocumentation.

Topics

Restrictions for IP addresses in the ODB network

Note the following restrictions regarding CIDR ranges in the ODB network:

- You can't modify the client or backup subnet CIDR range for the ODB network after you create it.
- You can't use the VPC CIDR ranges in the **Restricted associations** column in the table in IPv4 CIDR block association restrictions.
- For Exadata X9M, IP addresses 100.106.0.0/16 and 100.107.0.0/16 are reserved for the cluster interconnect by OCI automation, so you can't do the following:
  - Assign these ranges to the client or backup CIDR range of the ODB network.
  - Use these ranges for a VPC CIDR that is used to connect to the ODB network.
- The following CIDR ranges are reserved for Oracle Cloud Infrastructure and can't be used for the ODB network:
  - Oracle Cloud reserved range CIDR 169.254.0.0/16
  - Reserved Class D 224.0.0.0 — 239.255.255.255
  - Reserved Class E 240.0.0.0 — 255.255.255.255
- You can't overlap the IP address CIDR ranges for the client and backup subnets.
- You can't overlap the IP address CIDR ranges allocated for the client and backup subnets with the VPC CIDR ranges used to connect to the ODB network.

- You can't provision VMs in a VM cluster into different ODB networks. The network is a property of the VM cluster, which means you can only provision the VMs in the VM cluster into the same ODB network.

Client subnet CIDR requirements for the ODB network

In the following table, you can find the number of IP addresses consumed by the service and infrastructure for the client subnet CIDR. The minimum CIDR size for the client subnet is /27, and the maximum size is /16.

Number of IP addresses	Consumed by	Notes
6	Oracle Database@AWS	These IP addresses are reserved regardless of how many VM clusters you provision in the ODB network. Oracle Database@AWS consumes the following: • 3 IP addresses reserved for the ODB network resources in AWS • 3 IP addresses reserved for the OCI networking service
3	Each VM cluster	These IP addresses are reserved for Single Client Access Names (SCANs) regardless of how many VMs are present in each VM cluster.
4	Each VM	These IP addresses depend solely on the number of VMs in the infrastructure.

Backup subnet CIDR requirements for the ODB network

In the following table, you can find the number of IP addresses consumed by the service and infrastructure for the backup subnet CIDR. The minimum CIDR size for the backup subnet is /28, and the maximum size is /16.

Number of IP addresses	Consumed by	Notes
3	Oracle Database@AWS	These IP addresses are reserved regardless of how many VM clusters you provision in the ODB network. Oracle Database@AWS consumes the following: • 2 IP addresses at the beginning of the CIDR range • 1 IP address at the end of the CIDR range
3	Each VM	These IP addresses depend solely on the number of VMs in the infrastructure.

IP consumption scenarios for the ODB network

In the following table, you can see the IP addresses consumed in the ODB network for different configurations of VM clusters. Whereas /28 is the technical minimum CIDR range for the client subnet CIDR to deploy 1 VM cluster with 2 VMs, we recommend that you use at least a /27 CIDR range. In this case, the IP range isn't fully consumed by the VM clusters and permits allocation of additional IP addresses.

Configuration	Client IPs consumed	Client IPs minimum	Backup IPs consumed	Backup IPs minimum
1 VM cluster with 2 VMs	17 (6 service + 3 cluster + 4*2)	32 (/27 CIDR range)	9 (3 service + 3*2)	16 (/28 CIDR range)
1 VM cluster with 3 VMs	21 (6 service + 3 cluster + 4*3)	32 (/27 CIDR range)	12 (3 service + 3*3)	16 (/28 CIDR range)
1 VM cluster with 4 VMs	25 (6 service + 3 cluster + 4*4)	32 (/27 CIDR range)	15 (3 service + 3*4)	16 (/28 CIDR range)
1 VM cluster with 8 VMs	41 (6 service + 3 cluster + 4*8)	64 (/26 CIDR range)	27 (3 service + 3*8)	32 (/27 CIDR range)

The following table shows how many instances of each configuration are possible given a specific client CIDR range. For example, 1 VM cluster with 4 VMs consumes 24 IP addresses in the client subnet. If the CIDR range is /25, 128 IP addresses are available. Thus, you can provision 5 VM clusters in the subnet.

VM cluster configura tion	Number with /27 (32 IPs)	Number with /26 (64 IPs)	Number with /25 (128 IPs)	Number with /24 (256 IPs)	Number when /23 (512 IPs)	Number when /22 (1024 IPs)
1 VM cluster with 2 VMs (16 IPs)	1	3	7	15	30	60
1 VM cluster with 3 VMs (20 IPs)	1	3	6	12	24	48
1 VM cluster with 4 VMs (24 IPs)	1	2	5	10	20	40
2 VM clusters with 2 VMs each (27 IPs)	1	2	4	9	18	36
2 VM clusters with 3 VMs each (35 IPs)	0	1	3	7	14	28
2 VM clusters with 4 VMs each (43 IPs)	0	1	2	5	11	23

Step 1: Create an ODB network in Oracle Database@AWS

An ODB network is a private isolated network that hosts OCI infrastructure in an Availability Zone (AZ). An ODB network and an Oracle Exadata infrastructure are preconditions for provisioning VM clusters and creating Exadata databases. You can create the ODB network and Oracle Exadata infrastructure in either order. For more information, see ODB network and ODB peering.

This task assumes that you have read Planning IP address space in Oracle Database@AWS. To modify or delete the ODB network later, see Managing Oracle Database@AWS.

To create an ODB network

1. Sign in to the AWS Management Console and open the Oracle Database@AWS console at https://console.aws.amazon.com/odb/.
2. Choose your AWS Region in the upper right. For more information, see Supported Regions for Oracle Database@AWS.
3. From the left pane, choose **ODB networks**.
4. Choose **Create ODB network**.
5. For **ODB network name**, enter a network name. The name must be 1–255 characters and begin with an alphabetic character or underscore. It can't contain consecutive hyphens.
6. For **Availability Zone**, choose an AZ name. For supported AZs, see Supported Regions for Oracle Database@AWS.
7. For **Client subnet CIDR**, specify a CIDR range for the client connections. For more information, see Client subnet CIDR requirements for the ODB network.
8. For **Backup subnet CIDR**, specify a CIDR range for the backup connections. To isolate the backup traffic and improve resiliency, we recommend that you don't overlap the backup CIDR and the client CIDR. For more information, see Backup subnet CIDR requirements for the ODB network.
9. For **DNS configuration**, choose either of the following options:

   Default

   For **Domain name prefix**, enter a name to use as a prefix to your domain. The domain name is fixed as **oraclevcn.com**. For example, if you enter **`myhost`**, the fully qualified domain name is **myhost.oraclevcn.com**.

   Custom domain name

   For **Domain name**, enter a complete domain name. For example, you might enter **myhost.myodb.com**.
10. (Optional) For **Service integrations**, select a service to integrate with your network using VPC Lattice. Oracle Database@AWS integrates with various AWS services to provide enhanced functionality and connectivity options for your Oracle databases. Select either of the following integrations:

Amazon S3

Enable direct ODB network access to Amazon S3. Your databases can access S3 for data import/export or custom backups. You can enter a JSON policy. For more information, see User-managed backups to Amazon S3 in Oracle Database@AWS.

Zero-ETL

Enable real-time analytics and machine learning on transactional data using Amazon Redshift. For more information, see Oracle Database@AWS Zero-ETL integration with Amazon Redshift.

Note

When you create your ODB network, Oracle Database@AWS automatically preconfigures network access for Oracle managed backups to Amazon S3. You can't enable or disable this integration. For more information, see AWS service integrations.

11. (Optional) For **Tags**, enter up to 50 tags for the network. A tag is a key-value pair that you can use to organize and track your resources.
12. Choose **Create ODB network**.

After you have created an ODB network, you can peer it to a VPC. *ODB peering* is a user-created network connection that enables traffic to be routed privately between an Amazon VPC and an ODB network. After peering, an Amazon EC2 instance within the VPC can communicate with resources in the ODB network as if they were within the same network. For more information, see Configuring ODB peering to an Amazon VPC in Oracle Database@AWS.

Step 2: Create an Oracle Exadata infrastructure in Oracle Database@AWS

The Oracle Exadata infrastructure is the underlying architecture of database servers, storage servers, and networking that run Oracle Exadata databases. Choose either Exadata X9M or X11M as the system model. You can then create VM clusters on Exadata infrastructure using the AWS console.

You can create the Oracle Exadata infrastructure and the ODB network in either order. You don't need to specify networking information when you create the infrastructure.

You can't modify an Oracle Exadata infrastructure after you create it. To delete an Exadata infrastructure, see Deleting an Oracle Exadata infrastructure in Oracle Database@AWS.

To create an Exadata infrastructure

1. Sign in to the AWS Management Console and open the Oracle Database@AWS console at https://console.aws.amazon.com/odb/.
2. From the left pane, choose **Exadata infrastructures**.
3. Choose **Create Exadata infrastructure**.
4. For **Exadata infrastructure name**, enter a name. The name must be 1–255 characters and begin with an alphabetic character or underscore. It can't contain consecutive hyphens.
5. For **Availability Zone**, choose one of the supported AZs. Then choose **Next**.
6. For **Exadata system model**, choose either **Exadata.X9M** or **Exadata.X11M**. For **Exadata.X11M**, also choose the following server types:
   - For **Database server type**, choose the database server model type of your Exadata infrastructure. Currently, the only choice is **X11M**.
   - For **Storage server type**, choose the storage server model type of your Exadata infrastructure. Currently, the only choice is **X11M-HC**.
7. For **Database servers**, leave the default of 2 or move the slider to choose up to 32 servers. To specify more than 2, request a limit increase from OCI.

   Each Exadata X9M database server supports 126 OCPUs. Each Exadata X11M database server supports 760 ECPUs. The total compute count changes as you change the number of servers. For more information about OCPUs and ECPUs, see Compute Models in Autonomous Database in the Oracle documentation.
8. For **Storage servers**, leave the default of 3 or move the slider to choose up to 64 servers. To specify more than 3, request a limit increase from OCI. Each X9M storage server provides 64 TB. Each X11m storage server provides 80 TB. The total TB of storage changes as you change the number of servers. Then choose **Next**.
9. For **Maintenance window**, configure when system maintenance can occur:
   a. For **Scheduling preference**, select one of the following options:

- **Oracle-managed schedule** - Oracle determines the optimal time for maintenance activities.
- **Customer-managed schedule** - You specify when maintenance activities can occur.

b. For **Patching mode**, select one of the following options:

- **Rolling** - Updates are applied to one node at a time, allowing the database to remain available during patching.
- **Non-rolling** - Updates are applied to all nodes simultaneously, which may require downtime.

c. If you selected **Customer-managed schedule**, configure the following additional settings:

- For **Maintenance months**, select the months when maintenance can be performed.
- For **Week of the month**, select which week of the month maintenance can be performed (First, Second, Third, Fourth, or Last).
- For **Day of week**, select the day when maintenance can be performed (Monday through Sunday).
- For **Start hour**, select the hour when the maintenance window begins. The time is in UTC.
- For **Notification lead time**, select how many days in advance you want to be notified about upcoming maintenance.

Note

Oracle Cloud Infrastructure performs system maintenance during this window. During maintenance, your Exadata infrastructure remains available, but you might experience brief periods of higher latency.

10. (Optional) For **OCI maintenance notification contacts**, enter up to 10 email addresses. AWS forwards these email addresses to OCI. When updates occur, OCI mails notifications to the listed addresses.
11. (Optional) For **Tags**, enter up to 50 tags for the infrastructure. A tag is a key-value pair that you can use to organize and track your resources.
12. Choose **Next** and review your infrastructure settings.
13. Choose **Create Exadata infrastructure**.

Step 3: Create an Exadata VM cluster or Autonomous VM cluster in Oracle Database@AWS

An Exadata VM cluster is a set of VMs on which you can create Oracle Exadata databases. You create the VM clusters on Exadata infrastructure. You can deploy multiple VM clusters with different Oracle Exadata infrastructures in the same ODB network. You have full administrative control over the databases that you create on Exadata VM clusters.

An Autonomous VM cluster is a preallocated pool of Oracle Exadata compute and storage resources, virtualized at the VM level, that runs Autonomous Databases (ADB). Unlike user-managed databases that you create on an Exadata VM cluster, an Autonomous database is self-tuning, self-patching, and managed by Oracle rather than a database administrator.

Consider the following limitations when you create VM clusters:

- You can deploy a VM cluster only into the AZ where you created your ODB network and Oracle Exadata infrastructure.
- If you don't share a VM cluster across accounts, it must be in the same AWS account as the Oracle Exadata infrastructure. If you use AWS RAM to share an ODB network and Oracle Exadata infrastructure from one AWS account with a trusted account, the trusted account can create VM clusters in its own account.
- You can deploy only VM clusters in your ODB network. No other resources are permitted.
- You can't change the storage allocation after you create a VM cluster.

Important

The creation process can take over 6 hours, depending on the size of the VM cluster.

Exadata VM cluster

To create an Exadata VM cluster

1. Sign in to the AWS Management Console and open the Oracle Database@AWS console at https://console.aws.amazon.com/odb/.
2. From the left pane, choose **Exadata VM clusters**.

3. Choose **Create VM cluster**.
4. For **VM cluster name**, enter a name. The name must be 1–255 characters and begin with an alphabetic character or underscore. It can't contain consecutive hyphens.
5. (Optional) For **Grid Infrastructure cluster name**, enter a Grid infrastructure version for your VM cluster that matches the Oracle Database version you are using. The name must be 1–11 characters and can't contain hyphens.
6. For **Time zone**, enter a time zone.
7. For **License options**, choose **Bring Your Own License (BYOL)** or **License Included**, and then choose **Next**. This license is the OCI license provided by Oracle, not a license provided by AWS.
8. Configure Exadata infrastructure settings as follows:

   a. For **Infrastructure**, choose the following:

      - For **Exadata infrastructure name**, choose the infrastructure to use for this VM cluster.
      - For **Grid Infrastructure version**, choose the version to use for this VM cluster.
      - For **Exadata image version**, choose the version to use for this VM cluster. We recommend that you choose the version shown, which is the highest version available.

   b. For **Database servers**, select one or more database servers to host your VM cluster.

   c. For **Configuration**, do the following:

      - Choose the **CPU core count**, **Memory**, and **Local storage** for each VM, or accept the defaults.
      - Choose the total amount of **Exadata storage** for the VM cluster, or accept the default.

   d. (Optional) For **Storage allocation**, select any of the following options:

      - **Enable storage allocation for Exadata sparse snapshots**
      - **Enable storage allocation for local backups**

      The usable storage allocation changes as you select options. You can't change this storage allocation later. Review your selection, and then choose **Next**.

9. Configure connectivity as follows:

   a. For **ODB network**, choose an existing ODB network.

   b. For **Host name prefix**, enter a prefix for the VM cluster. Make sure not to include the domain name. The prefix forms the first portion of the Oracle Exadata VM cluster host name.

      Note

      The **Host domain name** is fixed as **oraclevcn.com**.

   c. For **SCAN listener port (TCP/IP)**, enter a port number that for TCP access to the single client access name (SCAN) listener. The default port is **1521**. Or you can enter a custom SCAN port in the range **1024–8999**, excluding the following port numbers: **2484**, **6100**, **6200**, **7060**, **7070**, **7085**, and **7879**. Then choose **Next**.

   d. For **SSH key pairs**, enter the public key portion of one or more key pairs used for SSH access to the VM cluster. Then choose **Next**.

10. (Optional) Choose diagnostics and tags as follows:

   a. Choose whether to enable diagnostic collection for **Diagnostic events**, **Health monitor**, and **Incident logs and trace collections**. Oracle can use this diagnostic information to identify, track, and resolve issues.

   b. For **Tags**, enter up to 50 tags for the VM cluster. A tag is a key-value pair that you can use to organize and track your resources. Then choose **Next**.

11. Review your settings. Then choose **Create VM cluster**.

Autonomous VM cluster

To create an Autonomous VM cluster

1. Sign in to the AWS Management Console and open the Oracle Database@AWS console at https://console.aws.amazon.com/odb/.
2. From the left pane, choose **Autonomous VM clusters**.
3. Choose **Create Autonomous VM cluster**.
4. For **VM cluster name**, enter a name. The name must be 1–255 characters and begin with an alphabetic character or underscore. It can't contain consecutive hyphens.
5. For **Time zone**, enter a time zone.

6. For **License options**, choose **Bring Your Own License (BYOL)** or **License Included**, and then choose **Next**. This license is the OCI license provided by Oracle, not a license provided by AWS.
7. Configure Exadata infrastructure settings as follows:

   a. For **Exadata infrastructure name**, choose the infrastructure to use for this Autonomous VM cluster.

   b. For **Database servers**, select one or more database servers to host your Autonomous VM cluster.

   c. For **Configuration**, do the following:

      - Choose the **ECPU core count per VM**, **Database memory per CPU**, **Database storage**, and **Maximum number of Autonomous Container Database** or accept the defaults.
      - Choose the total amount of **Exadata storage** for the Autonomous VM cluster, or accept the default.
8. Configure connectivity as follows:

   a. For **ODB network**, choose an existing ODB network.

   b. For **SCAN listener port (TCP/IP)**, enter a port number for Port (non-TLS). The default port is **1521**. Or you can enter a Port(TLS) in the range **1024–8999**, excluding the following port numbers: **2484**, **6100**, **6200**, **7060**, **7070**, **7085**, and **7879**. Then choose **Next**.

      Select **Enable mutual TLS (mTLS) authentication** to allow mutual TLS authentication.
9. (Optional) Choose diagnostics and tags as follows:

   a. Choose whether to schedule modification configuration to **Oracle-managed schedule** or **Customer-managed schedule**. If you choose **Customer-managed schedule**, set the **Maintenance months**, **Weeks of the month**, **Day of the week**, and **Start hour (UTC)**.

   b. For **Tags**, enter up to 50 tags for the Autonomous VM cluster. A tag is a key-value pair that you can use to organize and track your resources. Then choose **Next**.
10. Review your settings. Then choose **Create Autonomous VM cluster**.

Step 4: Create Oracle Exadata databases in Oracle Cloud Infrastructure

In Oracle Database@AWS, you can create and manage the following resources using the AWS console, CLI, or APIs:

- ODB networks
- Oracle Exadata infrastructure
- Exadata VM clusters and Autonomous VM clusters
- ODB peering connections

To create and manage Oracle Exadata databases on the infrastructure that you created, you must use the Oracle Cloud Infrastructure console rather than the Oracle Database@AWS dashboard. You can create a user-managed Exadata database on an Exadata VM cluster and an Autonomous Database on an Autonomous Exadata VM cluster. For information about creating Oracle databases in OCI, see Exadata Database in the Oracle Cloud Infrastructure documentation.

To create Oracle Exadata databases

1. Sign in to the AWS Management Console and open the Oracle Database@AWS console at https://console.aws.amazon.com/odb/.
2. From the left pane, choose **Exadata VM clusters** or **Autonomous VM clusters**.
3. Choose a VM cluster to see the details page.
4. Choose **Manage in OCI** to be redirected to the Oracle Cloud Infrastructure console.
5. Create your user-managed Exadata database or Autonomous Database in OCI.

Configuring ODB peering to an Amazon VPC in Oracle Database@AWS

ODB peering is a user-created network connection that enables traffic to be routed privately between an Amazon VPC and an ODB networkAfter you create a peering connection using the console, CLI, or API, make sure to update your VPC route tables and configure DNS resolution. For a conceptual overview of ODB peering, see ODB peering.

Creating an ODB peering connection in Oracle Database@AWS

Note

Now, you can now have up to 45 ODB peering connections between your Amazon VPCs and ODB network, allowing you to establish low latency connectivity at scale between your Exadata databases in your ODB network and applications in your VPCs.

With ODB peering connections, you can establish private network connectivity between your Oracle Exadata infrastructure and the applications running in your Amazon VPCs. Each ODB peering connection is a separate resource that you can create, view, and delete independently of the ODB network.

When creating an ODB peering connection, you can specify peer network CIDR ranges. This technique limits network access to the required subnets, reduces potential targets for attacks, and enables more granular network segmentation for compliance requirements.

You can create the following types of ODB peering connections:

Same-account ODB peering

You can create an ODB peering connection between an ODB network and an Amazon VPC in the same AWS account.

Cross-account ODB peering

You can create an ODB peering connection between an ODB network in one account and an Amazon VPC in a different account, after the ODB network has been shared using AWS RAM.

VPC owner accounts can manage CIDR ranges specified in the peering connection without also owning the ODB network.

You can create up to 45 peerings for a single ODB network.

Console

1. Sign in to the AWS Management Console and open the Oracle Database@AWS console at https://console.aws.amazon.com/odb/.
2. In the navigation pane, choose **ODB peering connections**.
3. Choose **Create ODB peering connection**.
4. (Optional) For **ODB peering name**, enter a unique name for your connection.
5. For **ODB network**, choose the ODB network to peer.
6. For **Peer network**, choose the Amazon VPC to peer with your ODB network.
7. (Optional) For **Peer network CIDRs**, specify additional CIDR blocks from the peer VPC that can access the ODB network. If you don't specify CIDRs, all CIDRs from the peer VPC are allowed access.
8. (Optional) In **Tags**, add a key and value pair.
9. Choose **Create ODB peering connection**.

After creating an ODB peering connection, configure your Amazon VPC route tables to route traffic to the peered ODB network. For more information, see Configuring VPC route tables for ODB peering. Note that Oracle Database@AWS automatically configures the ODB network route tables.

AWS CLI

To create an ODB peering connection, use the `create-odb-peering-connection` command.

```
aws odb create-odb-peering-connection \
    --odb-network-id odbnet-1234567890abcdef \
    --peer-network-id vpc-abcdef1234567890
```

To limit access to the ODB network to specific CIDR ranges, use the `--peer-network-cidrs-to-be-added` parameter. If you don't specify CIDR ranges, all ranges have access.

```
aws odb create-odb-peering-connection \
    --odb-network-id odbnet-1234567890abcdef \
    --peer-network-id vpc-abcdef1234567890 \
    --peer-network-cidrs-to-be-added "10.0.1.0/24,10.0.2.0/24"
```

To list your ODB peering connections, use the `list-odb-peering-connections` command.

```
aws odb list-odb-peering-connections
```

To get details about a specific ODB peering connection, use the `get-odb-peering-connection` command.

```
aws odb get-odb-peering-connection \
    --odb-peering-connection-id odbpcx-1234567890abcdef
```

Updating an ODB peering connection

You can update an existing ODB peering connection to add or remove peer network CIDRs. You control which subnets in the peer VPC have access to your ODB network.

Console

1. Sign in to the AWS Management Console and open the Oracle Database@AWS console at https://console.aws.amazon.com/odb/.
2. In the navigation pane, choose **ODB peering connections**.
3. Select the ODB peering connection that you want to update.
4. Choose **Actions**, and then choose **Update peering connection**.
5. In the **Peer network CIDRs** section, add or remove CIDR blocks as needed:
   - To add CIDRs, choose **Add CIDR** and enter the CIDR block.
   - To remove CIDRs, choose the **X** next to the CIDR block you want to remove.
6. Choose **Update peering connection**.

AWS CLI

To add peer network CIDRs to an ODB peering connection, specify the parameter `--peer-network-cidrs-to-be-added` in the `update-odb-peering-connection` command.

```
aws odb update-odb-peering-connection \
    --odb-peering-connection-id odbpcx-1234567890abcdef \
    --peer-network-cidrs-to-be-added "10.0.1.0/24,10.0.3.0/24"
```

To remove peer network CIDRs from an ODB peering connection, specify the parameter `--peer-network-cidrs-to-be-removed` in the `update-odb-peering-connection` command.

```
aws odb update-odb-peering-connection \
    --odb-peering-connection-id odbpcx-1234567890abcdef \
    --peer-network-cidrs-to-be-removed "10.0.1.0/24,10.0.3.0/24"
```

Configuring VPC route tables for ODB peering

A *route table* contains a set of rules, called *routes*, that determine where network traffic from your subnet or gateway is directed. The destination CIDR in a route table is a range of IP addresses where you want traffic to go. If you specified a VPC for ODB peering to your ODB network, update your VPC route table with the destination IP range in your ODB network. For more information about ODB peering, see ODB peering.

To update a route table, use the AWS CLI `ec2 create-route` command. The following examples updates Amazon VPC route tables. For more information, see Configuring VPC route tables for ODB peering.

```
aws ec2 create-route \
    --route-table-id rtb-1234567890abcdef \
    --destination-cidr-block 10.0.0.0/16 \
    --odb-network-arn arn:aws:odb:us-east-1:111111111111:odb-network/
odbnet_1234567890abcdef
```

The ODB network route tables are automatically updated with the VPC CIDRs. To allow access to the ODB network for only specific subnet CIDRs rather than all CIDRs in the VPC, you can specify peer network CIDRs when creating an ODB peering connection or update an existing ODB peering connection to add or remove peered CIDR ranges. For more information, see Creating an ODB peering connection in Oracle Database@AWS and Updating an ODB peering connection.

For more information about VPC route tables, see Subnet route tables in the *Amazon Virtual Private Cloud User Guide* and ec2 create-route in the *AWS CLI Command Reference*.

Configuring DNS for Oracle Database@AWS

Amazon Route 53 is a highly available and scalable Domain Name System (DNS) web service that you can use for DNS routing. When you create an ODB peering connection between your ODB network and a VPC, you need a mechanism to resolve DNS queries for ODB network resources from within the VPC. You can use Amazon Route 53 to configure the following resources:

- An outbound endpoint

  The endpoint is required to send DNS queries to the ODB network.
- A resolver rule

  This rule specifies the domain name of the DNS queries that the Route 53 Resolver forwards to the DNS for the ODB network.

How DNS works in Oracle Database@AWS

Oracle Database@AWS manages Domain Name System (DNS) configuration for the ODB network automatically. For the domain name, you can either specify a custom prefix for the default domain name `oraclevcn.com` or a fully custom domain name. For more information, see Step 1: Create an ODB network in Oracle Database@AWS.

When Oracle Database@AWS provisions an ODB network, it creates the following resources:

- An Oracle Cloud Infrastructure (OCI) virtual cloud network (VCN) with the same CIDR blocks as the ODB network

  This VCN resides in the customer's linked OCI tenancy. There is a 1:1 mapping between an ODB network and an OCI VCN. Every ODB network is associated with an OCI VCN.
- A private DNS resolver within the OCI VCN

  This DNS resolver handles DNS queries within the OCI VCN. OCI automation creates records for the VM cluster. Scans use the `*.oraclevcn.com` fully qualified domain name (FQDN).
- A DNS listening endpoint within the OCI VCN for the private DNS resolver

  You can find the DNS listening endpoint in the ODB network details page on the Oracle Database@AWS console.

Configuring an outbound endpoint in an ODB network in Oracle Database@AWS

An outbound endpoint allows DNS queries to be sent from your VPC to a network or IP address. The endpoint specifies the IP addresses from which queries originate. To forward DNS queries from your VPC to your ODB network, create an outbound endpoint using the Route 53 console. For more information, see Forwarding outbound DNS queries to your network.

To configure an outbound endpoint in an ODB network

1. Sign in to the AWS Management Console and open the Route 53 console at https://console.aws.amazon.com/route53/.
2. From the left pane, choose **Outbound endpoints**.
3. On the navigation bar, choose the **Region** for the VPC where you want to create the outbound endpoint.
4. Choose **Create outbound endpoint**.
5. Complete the **General settings for outbound endpoint** section as follows:

   a. Choose a **Security group** that allows outbound TCP and UDP connectivity to the following:

      - IP addresses that the resolvers use for DNS queries on your ODB network
      - Ports that the resolvers use for DNS queries on your ODB network

   b. For **Endpoint Type**, choose **IPv4**.

   c. For **Protocols for this endpoint**, choose **Do53**.
6. In **IP addresses**, provide the following information:

   - Either specify IP addresses or let the Route 53 Resolver choose IP addresses for you from the available addresses in the subnet. Choose a minimum of 2 up to a maximum of 6 IP addresses for DNS queries. We recommend that you choose IP addresses in at least two different Availability Zones.
   - For **Subnet**, choose subnets that have the following:
     - Route tables that include routes to the IP addresses of the DNS listener on ODB network
     - Network access control lists (ACLs) that allow UDP and TCP traffic to the IP addresses and the ports that the resolvers use for DNS queries on ODB network
     - Network ACLs that allow traffic from resolvers on destination port range 1024-65535

7. (Optional) For **Tags**, specify tags for the endpoint.
8. Choose **Submit**.

Configuring a resolver rule in Oracle Database@AWS

A resolver rule is a set of criteria that determines how to route DNS queries. Either reuse or create a rule that specifies the domain name of the DNS queries that the resolver forwards to the DNS for the ODB network.

Using an existing resolver rule

To use an existing resolver rule, your action depends on the type of rule:

A rule for the same domain in the same AWS Region as the VPC in your AWS account

Associate the rule with your VPC instead of creating a new rule. Choose the rule from the rule dashboard and associate it with the applicable VPCs in the AWS Region.

A rule for the same domain in the same Region as your VPC but in a different account

Use AWS Resource Access Manager to share the rule from the remote account to your account. When you share a rule, you also share the corresponding outbound endpoint. After you share the rule with your account, choose the rule from the rule dashboard and associate it with the VPCs in your account. For more information, see Managing forwarding rules.

Creating a new resolver rule

If you can't reuse an existing resolver rule, create a new rule using the Amazon Route 53 console.

To create a new resolver rule

1. Sign in to the AWS Management Console and open the Route 53 console at https://console.aws.amazon.com/route53/.
2. From the left pane, choose **Rules**.
3. On the navigation bar, choose the **Region** for the VPC where the outbound endpoint exists.
4. Choose **Create rule**.
5. Complete the **Rule for outbound traffic** sections as follows:

    a. For **Rule type**, choose **Forward rule**.

   b. For **Domain name**, specify the full domain name from ODB network.
   c. For **VPCs that use this rule**, associate it with the VPC from where DNS queries are forwarded to your ODB network.
   d. For **Outbound endpoint**, choose the outbound endpoint that you created in Configuring an outbound endpoint in an ODB network in Oracle Database@AWS.

   Note

   The VPC associated with this rule doesn't need to be the same VPC where you created the outbound endpoint.

6. Complete the **Target IP addresses** section as follows:
   a. For **IP address**, specify the IP address of the DNS listener IP on your ODB network.
   b. For **Port**, specify **53**. This is the port that the resolver use for DNS queries.

   Note

   The Route 53 Resolver forwards DNS queries that match this rule and originate from a VPC associated with this rule to the referenced outbound endpoint. These queries are forwarded to the target IP addresses that you specify in the **Target IP addresses**.

   c. For **Transmission protocol**, choose **Do53**.
7. (Optional) For **Tags**, specify tags for the rule.
8. Choose **Submit**.

Testing your DNS configuration in Oracle Database@AWS

After you have creating your outbound endpoint and resolver rule, test to make sure that the DNS resolves correctly. Using an Amazon EC2 instance in your application VPC, perform a DNS resolution as follows:

For Linux or MacOS

Use a command of the form `dig` *`record-name record-type`*.

For Windows

Use a command of the form `nslookup -type=record-name record-type`.

Configuring Multiple Application VPCs for Oracle Database@AWS

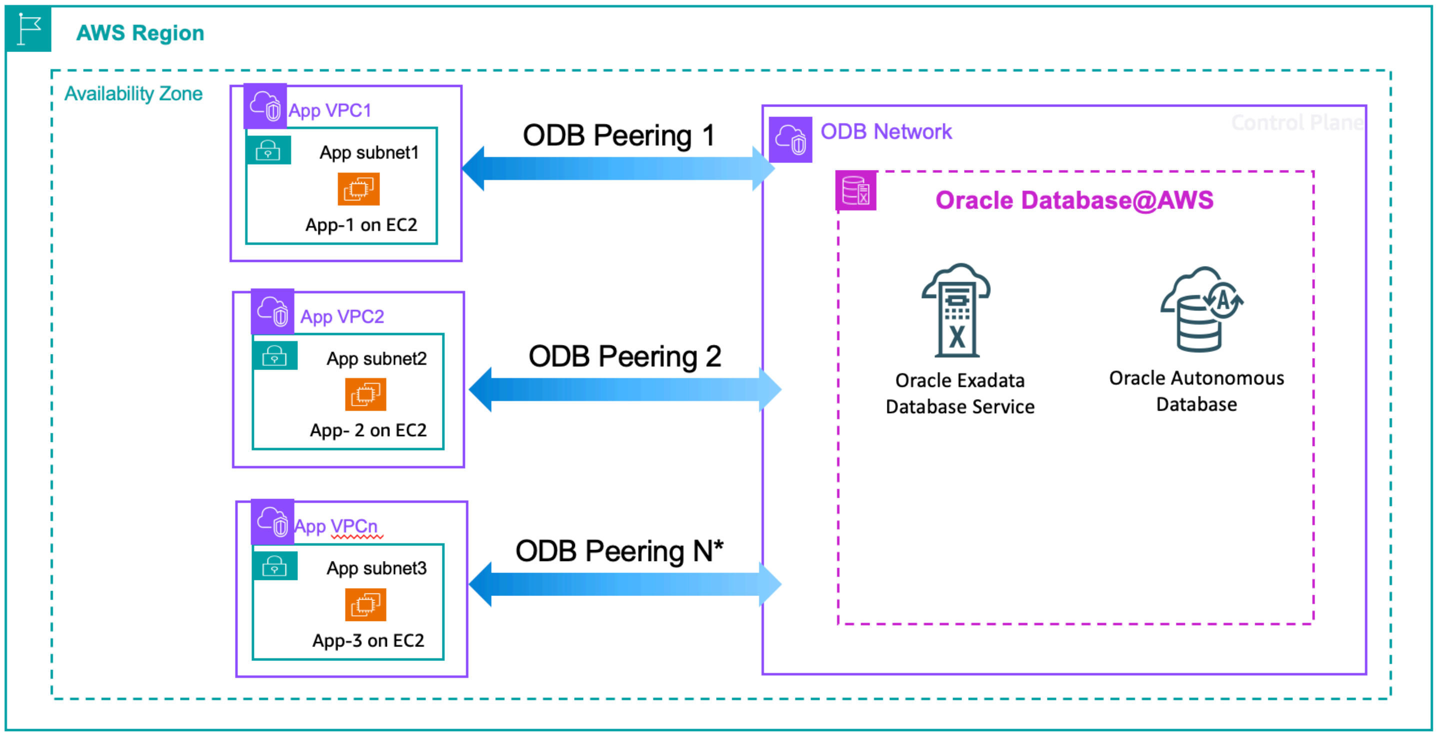
You can configure multiple application VPCs to connect directly to your ODB network through ODB peering connections. This architecture supports the following connectivity patterns:

- **Multiple VPCs to one ODB network** – Connect multiple application VPCs to a single ODB network
- **One VPC to multiple ODB networks –** Connect a single application VPC to multiple ODB networks
- **Multiple VPCs to multiple ODB networks** - Connect multiple application VPCs to multiple ODB networks
- **Multiple VPCs to multiple ODB networks via TGW** – Connect multiple application VPCs peered with multiple ODB networks with TGW and cloud WAN.

Architecture

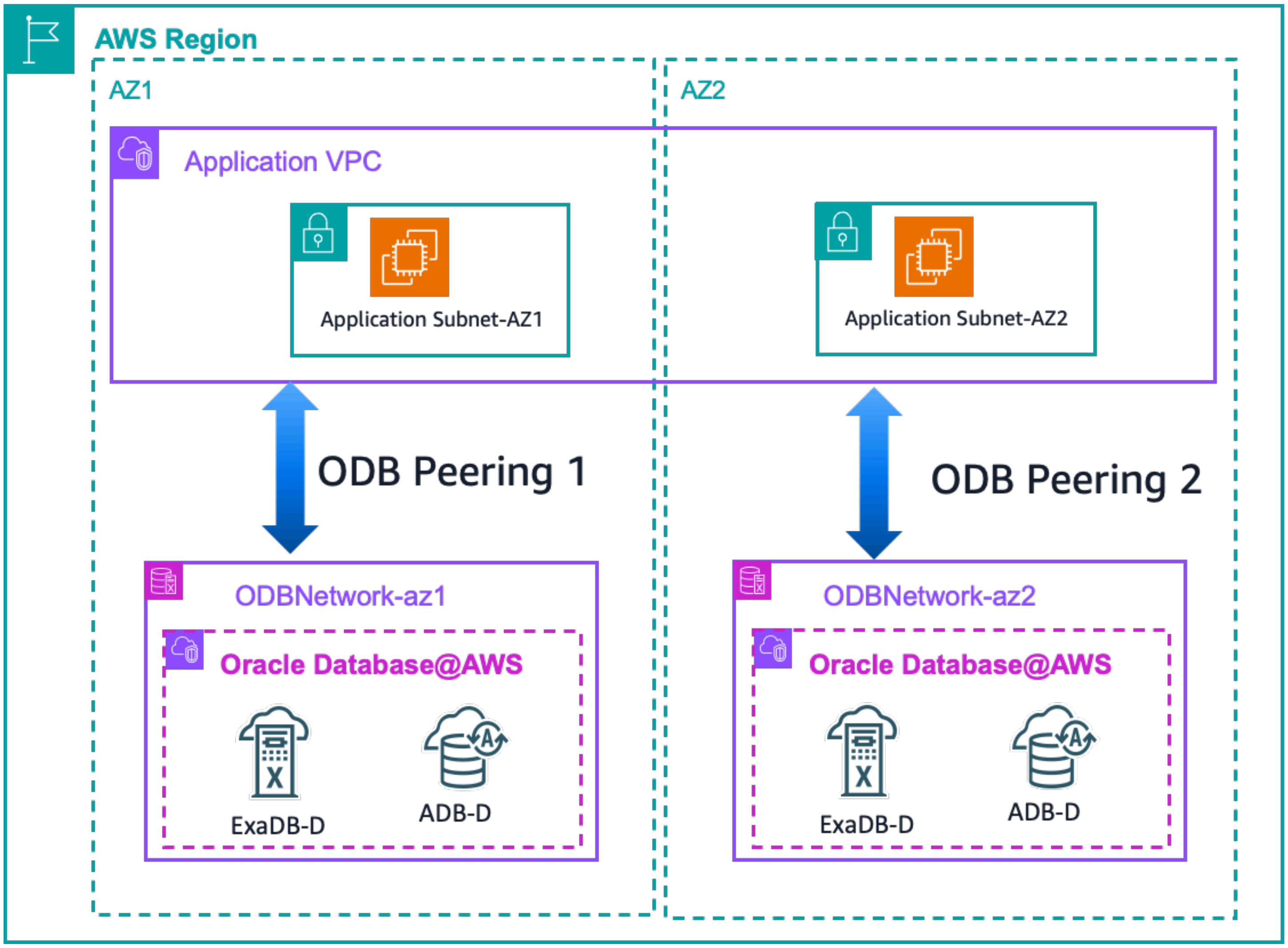
The following diagrams show example architectures for multiple application VPC configurations:

Single AZ with multiple Application VPC peers:

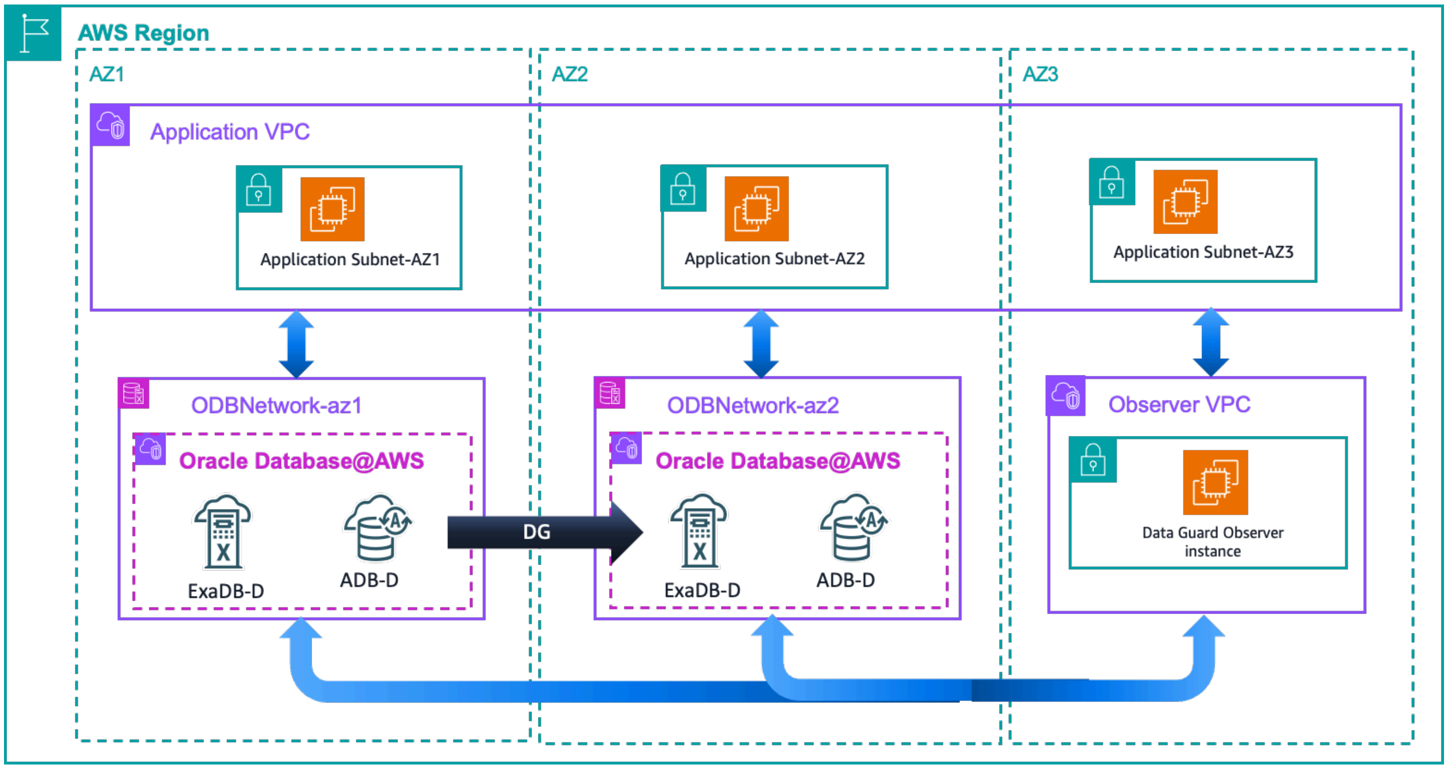


* Currently the number of max peering connection is 15 (subject to change)

Multi-AZ with a single application VPC peer:



Multi-AZ with Data Guard Observer for fast start failover (FSFO):



Note

Data Guard communications between ODB networks must be routed through the OCI network or a transit gateway with transit VPCs.

Benefits

- **Direct connectivity** – Application traffic flows directly between VPCs and the ODB network, reducing latency by eliminating intermediate routing hops.
- **Independent management** – Create, modify, or delete each peering connection independently without affecting other connections.
- **Network isolation** – Traffic between different application VPCs remains isolated, as each VPC communicates only with the ODB network through its own peering connection.
- **Flexible scaling** – Add or remove application VPCs or ODB networks as needed by creating or deleting individual peering connections.
- **Multi-network access** – Connect a single VPC to multiple ODB networks for cross-database operations, migrations, or disaster recovery scenarios.

Considerations

Before implementing this architecture, consider the following:

- An ODB network supports up to a maximum of 45 peering connections.
- Each VPC CIDR block consumes routing resources in the ODB network.
- CIDR blocks must not overlap between the ODB network and peered VPCs to avoid routing conflicts.
- A VPC can establish multiple peering connections to different ODB networks, but only one peering connection to each ODB network.
- Supernet CIDR blocks that encompass multiple existing subnets are not supported in peered configurations.
- The error message "security rules per network security group count limit exceeded" indicates that you have reached your NSG rule limit on OCI. To fix this issue, you raise the quota limit from the OCI console to increase your NSG rule limit (limit-name: securityrules-per-networksecuritygroup-count). This limit request is auto approved.Once your NSG rules are increased, you can add more ODB peering.
- ODB networks in US East (N. Virginia) and US West (Oregon) created before February 7, 2026, require a network upgrade before adding more than 1 ODB peering. To upgrade, you need to fully recreate your ODB network. Deletion of ODB network requires you to delete all Exadata VMs but does not require you to delete or recreate your Exadata Infrastructure.

Prerequisites

To implement this architecture, you need:

- An ODB network configured in your AWS account, or access to an ODB network shared with your account via AWS Resource Access Manager. For more information, see Step 1: Create an ODB network in Oracle Database@AWS.
- One or more application VPCs in the same AWS Region as the ODB network. For more information, see Create a VPC in the *Amazon Virtual Private Cloud User Guide*.
- Non-overlapping CIDR blocks for all VPCs and the ODB network

Configuration steps

1. Identify your application VPCs that require access to the ODB network, or identify the ODB networks that a single VPC needs to access.
2. Verify CIDR blocks to ensure no overlaps exist between VPCs and the ODB network.
3. Create ODB peering connections for each VPC-to-ODB network relationship that you need to establish.
4. Test connectivity from each application VPC to verify successful communication with the ODB network.

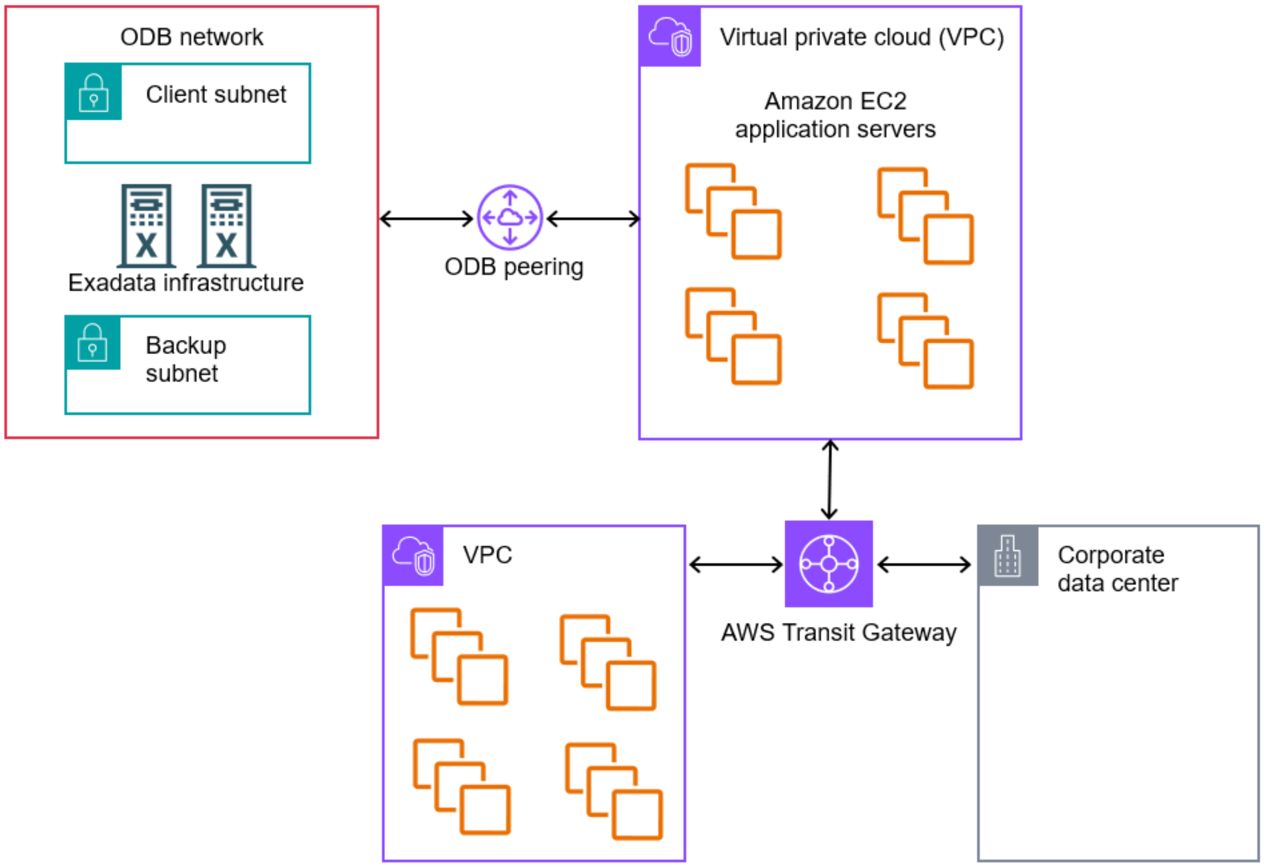
For more information about creating and managing ODB peering connections, see Creating an ODB peering connection in Oracle Database@AWS.

Configuring Amazon VPC Transit Gateways for Oracle Database@AWS

Amazon VPC Transit Gateways is a network transit hub that interconnects virtual private clouds (VPCs) and on-premises networks. Each VPC in the hub-and-spoke architecture can connect to the transit gateway to gain access to other connected VPCs. AWS Transit Gateway supports traffic for both IPv4 and IPv6.

In Oracle Database@AWS, an ODB network supports up to 45 peering connections. You can establish direct peering connections between your ODB network and multiple VPCs. Alternatively, if you connect a transit gateway to a VPC that is peered to an ODB network, you can route traffic from multiple VPCs through this central hub. Applications running in these different VPCs can access an Exadata VM cluster running in your ODB network.

The following diagram shows a transit gateway that is connected to two VPCs and one on-premises network.



In the preceding diagram, one VPC is peered to an ODB network. In this configuration, the ODB network can route traffic to all VPCs attached to the transit gateway. The route table for each VPC includes both the local route and routes that send traffic destined for the ODB network to the transit gateway.

In AWS Transit Gateway, you're charged for the number of connections that you make to the transit gateway per hour and the amount of traffic that flows through AWS Transit Gateway. For cost information, see AWS Transit Gateway pricing.

Requirements

Make sure your Oracle Database@AWS environment meets the following requirements:

- The VPC that is peered to your ODB network must be in the same AWS account. If the peered VPC is in a different account from the ODB network, transit gateway attachments fail regardless of the sharing configurations.
- The VPC that is peered to your ODB network must have a transit gateway attachment.

> **Note**
>
> If the transit gateway is configured for sharing, it can reside in any account. Thus, the gateway itself doesn't need to be in the same account as the VPC and ODB network.

- The transit gateway attachment must be in the same Availability Zone (AZ) as the ODB network.

Limitations

Note the following limitations of Amazon VPC Transit Gateways for Oracle Database@AWS:

- Amazon VPC Transit Gateways doesn't offer native integration to use an ODB network as an attachment. Therefore, VPC features such as the following aren't available:
  - Resolution of public DNS hostnames to private IP addresses
  - Event notification for changes in the ODB network topology, routing, and connection status
- Multicast traffic to the ODB network isn't supported.

Setting up and configuring a transit gateway

You create and configure a transit gateway by using the Amazon VPC console or `aws ec2` commands. The following procedure assumes that you don't have an ODB network peered to a VPC in your AWS account. If an ODB network and VPC are already peered in your account, skip steps 1–3.

> **Note**
>
> If you attach or reattach the attachments on your VPC, make sure you re-enter the CIDR ranges to the ODB ODB network.

To set up and configure a transit gateway for Oracle Database@AWS

1. Create an ODB network. For more information, see Step 1: Create an ODB network in Oracle Database@AWS.
2. Create a VPC, using the same account that contains the ODB network. For more information, see Create a VPC in the Amazon VPC User Guide.

3. Create an ODB peering connection between your ODB network and your VPC. For more information, see Configuring ODB peering to an Amazon VPC in Oracle Database@AWS.
4. Set up a transit gateway by following the steps in Get started with using Amazon VPC Transit Gateways. The gateway must be either in the same AWS account as the ODB network and VPC, or shared by another account.

   Important

   Create the transit gateway attachment in the same AZ as the ODB network.

5. Add CIDR ranges to your ODB network for the VPCs and on-premises networks that you plan to attach to your core network. For more information, see Updating an ODB network in Oracle Database@AWS.

   If you're using the CLI, run the command `update-odb-network` with `--peered-cidrs-to-be-added` and `--peered-cidrs-to-be-removed`. For more information, see the *AWS CLI Command Reference*.

Configuring AWS Cloud WAN for Oracle Database@AWS

AWS Cloud WAN is a managed wide-area networking (WAN) service. You can use AWS Cloud WAN to build, manage, and monitor a unified global network that connects resources running across your cloud and on-premises environments.

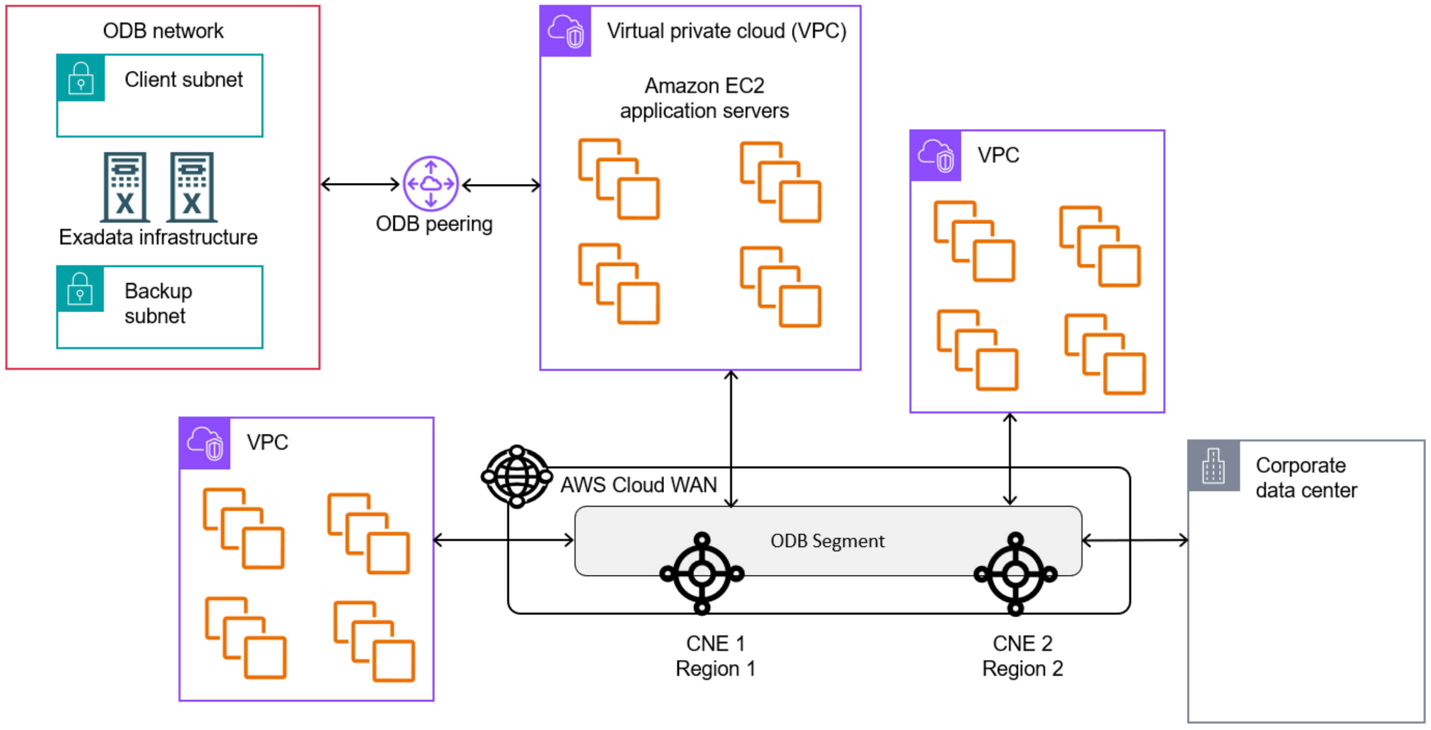
In AWS Cloud WAN, a *global network* is a single, private network that acts as the high-level container for your network objects. A *core network* is the part of your global network managed by AWS.

AWS Cloud WAN provides the following key benefits:

- Centralized network management that simplifies operations while maintaining security across multiple Regions
- Core networks with built-in segmentation to isolate traffic through multiple routing domains
- Support for policies to automate network management and define consistent configurations across your global network

In Oracle Database@AWS, an ODB network supports up to 45 peering connections. You can connect multiple VPCs directly to your ODB network, or use AWS Cloud WAN for global traffic routing. If you connect a AWS Cloud WAN core network to one or more peered VPCs, it enables global traffic routing. Applications in attached VPCs across multiple Regions can access Exadata VM clusters in your ODB network. You can isolate ODB network traffic in its own segment or enable access to other segments.

The following diagram shows an AWS Cloud WAN core network that is connected to three VPCs and one on-premises network.



AWS Cloud WAN doesn't offer native integration to use an ODB network as an attachment. Therefore, VPC features such as the following aren't available:

- Resolution of public DNS hostnames to private IP addresses
- Event notification for changes in the ODB network topology, routing, and connection status

In AWS Cloud WAN, you're charged hourly for the following:

- Number of Regions (core network edges)
- Number of core network attachments
- The amount of traffic that flows through your core network through the attachments

For detailed pricing information, see AWS Cloud WAN pricing.

To configure a core network for Oracle Database@AWS

1. Add CIDR ranges to your ODB network for the VPCs and on-premises networks that you plan to attach to your core network. For more information, see Updating an ODB network in Oracle Database@AWS.

   Note

   If you attach or reattach the attachments on your VPC, make sure you re-enter the CIDR ranges to the ODB ODB network.

2. Follow the steps in Create an AWS Cloud WAN global network and core network.

Entitlement sharing in Oracle Database@AWS

With Oracle Database@AWS, you can share AWS Marketplace entitlements for Oracle Database@AWS across AWS accounts in the same AWS organization. This allows other accounts to provision their own Oracle Exadata infrastructure and ODB network resources using your subscription.

Sharing methods

Oracle Database@AWS supports two methods for sharing:

Entitlement sharing with AWS License Manager

- Grant other accounts the ability to provision their own Oracle Exadata infrastructure and ODB network resources
- Each account operates independently with full resource lifecycle control
- Best for enabling self-service provisioning across teams or business units

Resource sharing with AWS Resource Access Manager (AWS RAM)

- Share already provisioned Oracle Exadata infrastructure and ODB network resources
- Centralize infrastructure management while allowing recipient accounts to create VM clusters
- Optimize costs by having multiple accounts use the same infrastructure

You can use both sharing methods simultaneously based on your organizational needs.

Limitations for Oracle Database@AWS entitlement sharing

When sharing Oracle Database@AWS entitlements, keep the following limitations in mind:

- You can only share with AWS accounts within your AWS organization
- You cannot share with an entire organizational unit (OU) or the entire organization
- An account can receive entitlements from only one buyer account (from one private offer)
- A buyer account cannot share entitlements with another buyer account

- Recipient accounts must initialize the Oracle Database@AWS service before they can use the shared entitlement
- Entitlement grant operations can only be performed from the US East (N. Virginia) Region

Sharing Oracle Database@AWS entitlements across accounts

To enable collaboration while optimizing costs, share Oracle Database@AWS entitlements with other AWS accounts within the same AWS organization. This topic explains how to share entitlements using AWS License Manager.

Prerequisites for sharing entitlements

Before you share Oracle Database@AWS entitlements, make sure that you have the following:

- An active Oracle Database@AWS subscription (you must be the buyer account that accepted the private offer through AWS Marketplace)
- The IDs of the AWS accounts in your organization that you want to share entitlements with
- Necessary permissions for grantor and grantee to use AWS License Manager resources and operations (for more information, see Identity and access management for License Manager in the *AWS License Manager User Guide*)
- Permissions listed below for you (grantor) and entitlement recipient (grantee)

Permissions required for entitlement sharing

In addition to AWS License Manager permissions, Oracle Database@AWS requires the following permissions:

Grantor permissions

- `odb:CreateGrantShare`
- `odb:UpdateGrantShare`
- `odb:DeleteGrantShare`

Grantee permissions

- `odb:UpdateGrantShare`

- `odb:DeleteGrantShare`

Sharing Oracle Database@AWS entitlements with another account using AWS License Manager

To share entitlements with another AWS account, you create a grant using AWS License Manager. For more information, see Distribute License Manager entitlements in the *AWS License Manager User Guide*.

After you create the grant, the recipient (grantee) must:

- Accept and activate the grant. For more information, see Grant acceptance and activation in License Manager in the *AWS License Manager User Guide*.
- Follow the initialization instructions for Oracle Database@AWS.

After initialization completes, the grantee can provision Oracle Database@AWS resources using the shared entitlement.

Resource sharing in Oracle Database@AWS

With Oracle Database@AWS, you can share Exadata infrastructure and your ODB network across multiple AWS accounts in the same AWS organization. This enables you to provision infrastructure once and reuse it across trusted accounts, allowing you to reduce costs while separating responsibilities.

When you share resources:

- The account that owns the resource (owner account) maintains control over the resource lifecycle.
- Accounts that receive access to shared resources (trusted accounts) can view and use these resources based on the permissions granted.
- Trusted accounts can create their own resources on shared infrastructure but cannot delete the underlying shared resources.

Oracle Database@AWS integration with AWS RAM

Oracle Database@AWS uses AWS Resource Access Manager (AWS RAM) to enable secure, controlled sharing of resources across accounts. With AWS RAM, you can securely share your Oracle Database@AWS resources across multiple AWS accounts within the same AWS organization. AWS RAM simplifies resource sharing, reduces operational overhead, and provides security and visibility into shared Oracle Database@AWS resources.

With AWS RAM, you share resources that you own by creating a *resource share*. A resource share specifies the resources to share, and the AWS accounts with whom to share them.

Benefits of resource sharing in Oracle Database@AWS

Sharing Oracle Database@AWS resources across accounts provides the following benefits:

- **Cost optimization** – Provision expensive Exadata infrastructure once through an administrative account and share it with multiple accounts, reducing overall costs.
- **Separation of responsibilities** – Maintain clear boundaries between infrastructure administrators and database users while allowing collaboration.

- **Simplified management** – Centralize infrastructure provisioning and management while enabling distributed database operations.
- **Consistent governance** – Apply consistent policies and controls across shared resources.

For example, an administrator can provision the Oracle Exadata infrastructure and ODB network in their AWS account and share it with developer accounts. Developers can then create VM clusters on this shared infrastructure without needing to provision their own expensive hardware. This approach significantly reduces costs while maintaining proper separation of responsibilities between accounts.

How resource sharing works in Oracle Database@AWS

You can share the following Oracle Database@AWS resources:

- Oracle Exadata infrastructure
- ODB network

Oracle Database@AWS shares the preceding resources through the following process:

1. The buyer account (the account that accepts the Oracle Database@AWS private offer via AWS Marketplace) provisions Oracle Database@AWS resources, such as Exadata infrastructure and an ODB network.
2. The buyer account creates a resource share using AWS RAM, specifying the resources to share and the trusted accounts to share them with.
3. The resource shares for the trusted accounts within the same organization are accepted automatically.
4. Before using shared resources, trusted accounts must initialize the Oracle Database@AWS service in their account by using the `aws odb initialize-service` command or by choosing **Activate account** in the Oracle Database@AWS console.
5. After initialization, trusted accounts can create their own resources on the shared infrastructure, such as VM clusters on shared Exadata infrastructure and ODB network.

Permissions on shared resources for trusted accounts

When you share resources, Oracle Database@AWS automatically selects specific actions (managed permissions) for each resource type:

For Exadata infrastructure

Oracle Database@AWS grants the following permissions to trusted accounts:

- `odb:CreateCloudVmCluster`
- `odb:CreateCloudAutonomousVmCluster`
- `odb:GetCloudExadataInfrastructure`
- `odb:ListCloudExadataInfrastructures`
- `odb:GetCloudExadataInfrastructureUnallocatedResources`
- `odb:ListDbServers`
- `odb:GetDbServer`
- `odb:ListCloudVmClusters`
- `odb:ListCloudAutonomousVmClusters`

For ODB network

The following permissions are granted to trusted accounts:

- `odb:CreateCloudVmCluster`
- `odb:CreateCloudAutonomousVmCluster`
- `odb:GetOdbNetwork`
- `odb:ListOdbNetworks`
- `odb:CreateOdbPeeringConnection`
- `odb:ListOdbPeeringConnections`

Resource sharing respects the hierarchical nature of Oracle Database@AWS resources. For example, if you share Exadata infrastructure, trusted accounts can create VM clusters on this infrastructure, but they can't modify or delete the Exadata infrastructure itself.

When a resource is unshared, trusted accounts lose the ability to create new resources on the shared infrastructure. However, any resources they've already created remain accessible and functional.

Limitations for Oracle Database@AWS resource sharing

Before sharing resources, keep the following limitations in mind.

Limitations for sharing resources

When sharing Oracle Database@AWS resources, keep in mind the following limitations:

- You can share resources only with AWS account IDs.
- You can share resources only for AWS accounts within the same AWS organization.
- You share resources within a specific AWS Region. To share resources across Regions, you must create separate resource shares in each Region.
- When you create a resource share, the actions (managed permissions) for each resource type are automatically selected and can't be modified.
- You can't use Oracle Database@AWS as a resource and share with other AWS accounts.
- A trusted account can use shared resources from only one buyer account (from one private offer). Thus, two buyer accounts can't share resources with the same trusted account.
- A buyer account can't share resources with another buyer account.
- Resources shared with a trusted account must be shared by the buyer account in the buyer's home region first.
- When you unshare a resource, we recommend that you wait approximately 15 minutes before resharing the same resource with the same trusted account.

Limitations for creating and using shared resources

When creating or using Oracle Database@AWS resources, keep in mind the following limitations:

- Only the buyer account can create Exadata infrastructure and ODB network resources. The buyer account is the one that accepts the Oracle Database@AWS private offer.
- Trusted accounts can create resources only on Exadata infrastructure shared by the buyer account.
- Trusted accounts must initialize the Oracle Database@AWS service in their account before they can use shared resources.

Limitations for deleting shared resources

- You can't delete Exadata infrastructure that has VM clusters created by trusted accounts until those VM clusters are removed.
- You can't delete an ODB network that has an ODB peering connection created by a trusted account until the ODB peering connection has been removed.
- The buyer account can't delete Oracle Database@AWS resources created by trusted accounts.
- Trusted accounts can view shared resources but can't modify or delete Oracle Database@AWS resources owned by the buyer account.

Sharing Oracle Database@AWS resources across accounts

To enable collaboration while optimizing costs, share Oracle Database@AWS resources with other AWS accounts within the same AWS organization. This topic explains how to share resources using AWS Resource Access Manager (AWS RAM).

Topics

Prerequisites for sharing resources

Before you share Oracle Database@AWS resources, make sure that you have the following:

- An active Oracle Database@AWS subscription (you must be the buyer account that accepted the private offer through AWS Marketplace)
- The IDs or names of the resources you want to share, such as Exadata infrastructure or ODB networks
- The IDs of the AWS accounts in your organization that you want to share resources with
- Necessary permissions to create resource shares in AWS RAM

- The ability to share resources with AWS Organizations using AWS RAM (for more information, see Enable resource sharing within AWS Organizations in the *AWS Resource Access Manager User Guide*)

Sharing Oracle Database@AWS resources with another account using AWS RAM

To share an Exadata infrastructure or ODB network with another AWS account, you create a resource share using AWS RAM. This allows the trusted account to create VM clusters on your Exadata infrastructure.

Console

1. Open the AWS RAM console at https://console.aws.amazon.com/ram/.
2. Choose **Create resource share**.
3. For **Name**, enter a descriptive name for your resource share.
4. Under **Select resource type**, either of the following resources:
   - **Oracle Database@AWS ODB network**
   - **Oracle Database@AWS Exadata Infrastructure**
5. Select the Exadata infrastructure resources you want to share. Choose Next until you get to **Grant access to principals**.
6. Under **Principals**, choose **AWS accounts**, and then enter the AWS account IDs you want to share with.
7. Under **Managed permissions**, select the following permissions to allow the trusted account to create VM clusters on the shared Exadata infrastructure:
   - **AWSRAMDefaultPermissionODBNetwork**
   - **AWSRAMDefaultPermissionODBCloudExadataInfrastructure**
8. Choose **Create resource share**.

AWS CLI

To share resources using the AWS CLI, use the `aws ram create-resource-share` command. The following example creates a resource share named `ExadataInfraShare` that shares the specified Exadata infrastructure with account 222222222222, allowing this account to create VM clusters on the shared infrastructure.

```
aws ram create-resource-share --region us-east-1 \
    --name "ExadataInfraShare" \
    --resource-arns arn:aws:odb:us-east-1:111111111111:cloud-exadata-infrastructure/
exa_infra_1 \
    --principals 222222222222
```

Viewing your resource shares

To view the resources you've shared and the accounts you've shared them with:

Console

1. Open the AWS RAM console at https://console.aws.amazon.com/ram/.
2. Choose **Shared resources** to view resources you've shared with other accounts.
3. Select a resource share to view its details, including the resources shared and the principals they're shared with.

AWS CLI

To view your resource shares using the AWS CLI, use the `get-resource-shares` command:

```
aws ram get-resource-shares --resource-owner SELF
```

To view the resources in a specific resource share, use the `list-resources` command:

```
aws ram list-resources \
    --resource-owner SELF \
    --resource-share-arns arn:aws:ram:us-east-1:111111111111:resource-share/12345678-
abcd-1234-efgh-111111111111
```

To view the principals (accounts) that a resource share is shared with, use the `list-principals` command:

```
aws ram list-principals \
    --resource-owner SELF \
    --resource-share-arns arn:aws:ram:us-east-1:111111111111:resource-share/12345678-
abcd-1234-efgh-111111111111
```

Updating or deleting resource shares using AWS RAM

To stop sharing a resource with a trusted account using AWS RAM, take any of the following actions:

- Remove the resource from the resource share.
- Remove the trusted account from the resource share.
- Delete the resource share.

Before you revoke access to or delete a shared resource, consider the following implications:

- Trusted accounts can no longer create new resources on the unshared infrastructure.
- Existing resources created by trusted accounts on the shared Exadata infrastructure continue to function and remain accessible to those AWS accounts.
- You can't delete Exadata infrastructure that has VM clusters created by trusted accounts until those VM clusters are removed.

Before unsharing resources, we recommend that you coordinate with the trusted accounts to ensure a smooth transition.

For more information, see Update a resource share in AWS RAM and Deleting a resource share in AWS RAM in the *AWS Resource Access Manager User Guide*.

Initializing Oracle Database@AWS in a trusted account

A trusted account is an AWS account that you designate as eligible to receive resource shares. It must be another individual AWS account in your AWS organization. Before you can use shared Oracle Database@AWS resources in a trusted account, you must initialize the service. Initialization creates the necessary metadata and establishes the connection between your AWS account and Oracle Cloud Infrastructure.

Topics

- What is Oracle Database@AWS initialization?
- Next steps

What is Oracle Database@AWS initialization?

After a resource has been shared with your account, you must initialize the Oracle Database@AWS service before you can access or use the shared resource. If you try to use Oracle Database@AWS APIs without initializing the service first, you receive an error.

Initialization is a one-time process. It creates the necessary metadata and establishes a connection between your AWS account and Oracle Cloud Infrastructure.

You can initialize the service using either the AWS Management Console or the AWS CLI.

Console

1. Open the Oracle Database@AWS console at https://console.aws.amazon.com/odb/.
2. If this is your first time accessing the Oracle Database@AWS console in this account, you see a welcome page.
3. Choose **Activate account**.
4. The service initialization process begins. This process might take a few minutes to complete.
5. Refresh the welcome page periodically until the **Activate account** button changes to the **Dashboard** button.
6. Choose **Dashboard** to begin using Oracle Database@AWS.

AWS CLI

To initialize Oracle Database@AWS in your trusted account using the AWS CLI, use the `initialize-service` command.

```
aws odb initialize-service
```

To check the initialization status, use the `get-oci-onboarding-status` command.

```
aws odb get-oci-onboarding-status
```

When initialization is complete, the output shows a status of `ACTIVE_LIMITED`, indicating that your account can access shared resources but can't create a new Exadata infrastructure or ODB network.

Next steps

After you initialize Oracle Database@AWS in your trusted account, you can do the following:

- View shared resources using the `list` and `get` commands or in the AWS console.
- Create VM clusters and Autonomous VM clusters on a shared Exadata infrastructure and ODB network.
- Create an ODB peering connection on a shared ODB network.

For more information about working with shared resources, see Working with shared Oracle Database@AWS resources in a trusted account.

Working with shared Oracle Database@AWS resources in a trusted account

After a resource has been shared with your trusted account and you've initialized the Oracle Database@AWS service, you can view and use the shared resource. This topic explains how to work with shared resources in a trusted account.

Topics

- Limitations for shared resources in a trusted account
- Creating VM clusters on shared Exadata infrastructure
- Viewing shared resources in a trusted account
- Setting up ODB peering with shared ODB networks

Limitations for shared resources in a trusted account

When working with shared Oracle Database@AWS resources, be aware of the following limitations:

- Resource sharing is supported only within the same AWS organization.
- Only the buyer account (the account that accepts the Oracle Database@AWS private offer) can create Exadata infrastructure and ODB network resources.
- You can create resources only on shared infrastructure and only if you have the necessary permissions.

- The specific actions (managed permissions) for each resource type are automatically selected during resource share creation and can't be modified.
- You can't modify or delete resources owned by another account.
- Resources that you create on shared infrastructure are owned by your account and count toward your OCI quotas. The same applies to parent resources.
- If the owner account unshares a resource, you can no longer create new resources on this shared infrastructure. However, your existing resources continue to function.
- Cross-Region resource sharing isn't supported. You can only share resources within the same AWS Region.
- Trusted account resources are billed to the buyer of the Oracle Database@AWS subscription.
- When using a resource that is shared, you must provide the Amazon Resource Name (ARN).

Creating VM clusters on shared Exadata infrastructure

If your trusted account has access to a shared Exadata infrastructure and ODB network, you can create Exadata VM clusters, Autonomous VM clusters, or ODB peerings on this infrastructure.

Note

When using a resource that is shared to you, instead of only specifying the resource ID, you must specifying the Amazon Resource Name (ARN).

Console

1. Open the Oracle Database@AWS console at https://console.aws.amazon.com/odb/.
2. In the navigation pane, choose **Exadata VM clusters** or **Autonomous VM clusters**.
3. Choose **Create VM cluster** or **Create Autonomous VM cluster**.
4. For **Exadata infrastructure**, select the shared Exadata infrastructure on which you want to create the VM cluster.
5. Complete the remaining fields as required for your VM cluster configuration.
6. Choose **Create VM cluster** or **Create Autonomous VM cluster**.

AWS CLI

To create a VM cluster on shared Exadata infrastructure using the AWS CLI, use the `create-cloud-vm-cluster` command:

```
aws odb create-cloud-vm-cluster --region us-east-1 \
    --cloud-exadata-infrastructure-id arn:aws:odb:us-east-1:111111111111:cloud-exadata-infrastructure/exa_aaaaaaaaaa \
    --odb-network-id arn:aws:odb:us-east-1:111111111111:odb-network/odbnet_aaaaaaaaaa \
    --cpu-core-count 4 \
    --display-name "Shared-VMC-1" \
    --gi-version "19.0.0.0" \
    --hostname "vmchost" \
    --ssh-public-keys "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQ..." \
```

To create an Autonomous VM cluster on shared Exadata infrastructure using the AWS CLI, use the `create-cloud-vm-cluster` command:

```
aws odb create-cloud-autonomous-vm-cluster --region us-east-1  \
    --cloud-exadata-infrastructure-id arn:aws:odb:us-east-1:111111111111:cloud-exadata-infrastructure/exa_aaaaaaaaaa \
    --odb-network-id arn:aws:odb:us-east-1:111111111111:odb-network/odbnet_aaaaaaaaaa\
    --display-name "Shared-AVMC-1" \
    --autonomous-data-storage-size-in-tbs 8 \
    --cpu-core-count-per-node 16
```

The VM cluster is created on the specified shared Exadata infrastructure and is owned by your trusted account.

Viewing shared resources in a trusted account

You can view resources that have been shared with your account using the AWS Management Console or the AWS CLI.

Console

1. Open the Oracle Database@AWS console at https://console.aws.amazon.com/odb/.
2. In the navigation pane, choose the resource type you want to view: **Exadata infrastructure** or **ODB network**.

3. The console displays resources shared with you.
4. Select a shared resource to view its details.

AWS CLI

To view shared resources using the AWS CLI, use the appropriate `list` command for the resource type. For example, to list Exadata infrastructure:

```
aws odb list-cloud-exadata-infrastructures
```

The response shows resources shared with you.

To get detailed information about a specific shared resource, use the appropriate `get` command with the resource ID:

```
aws odb get-cloud-exadata-infrastructure --cloud-exadata-infrastructure-id exa_infra_1
```

Setting up ODB peering with shared ODB networks

To enable communication between your applications and databases on shared ODB networks, you can set up ODB peering between your VPC and the shared ODB network. For more information about ODB peering, see Creating an ODB peering connection in Oracle Database@AWS.

Console

1. Open the Oracle Database@AWS console at https://console.aws.amazon.com/odb/.
2. In the navigation pane, choose **ODB peering**.
3. Choose **Create ODB network peering**.
4. For **ODB network**, select the shared ODB network you want to peer with.
5. For **Peer network**, select your VPC.
6. Choose **Create ODB network peering**.

AWS CLI

To create a network peering connection between your VPC and a shared ODB network using the AWS CLI, use the `create-odb-peering-connection` command.

```
aws odb create-odb-peering-connection \
    --odb-network-id odbnet_1234567890abcdef \
    --peer-network-id vpc-abcdef1234567890
```

After creating the peering connection, update your route tables to enable traffic between the peered networks.

```
aws ec2 create-route \
    --route-table-id rtb-1234567890abcdef \
    --destination-cidr-block 10.0.0.0/16 \
    --odb-network-arn arn:aws:odb:us-east-1:111111111111:odb-network/
odbnet_1234567890abcdef
```

Managing Oracle Database@AWS

You can modify and delete some Oracle Database@AWS resources after you create them.

Updating an ODB network in Oracle Database@AWS

You can update the following ODB network resources:

- The ODB network name
- The Amazon VPC to use for establishing an ODB peering connection to the ODB network
- The VPC CIDR ranges that can access Exadata resources in the ODB network

> **Note**
>
> By specifying CIDR ranges, you limit connectivity to the necessary VPC subnets instead of making the entire VPC available to the ODB network.

This section assumes that you have already created an ODB network in Step 1: Create an ODB network in Oracle Database@AWS.

To update an ODB network

1. Sign in to the AWS Management Console and open the Oracle Database@AWS console at https://console.aws.amazon.com/odb/.
2. From the left pane, choose **ODB networks**.
3. Select the network that you want to modify.
4. Choose **Modify**.
5. (Optional) For **ODB network name**, enter a new network name. The name must be 1–255 characters and begin with an alphabetic character or underscore. It can't contain consecutive hyphens.
6. (Optional) For **Peered CIDRs**, specify CIDR ranges from the peered VPC that need connectivity to the ODB network. To limit access, we recommend that you specify the minimum required CIDR ranges.
7. (Optional) For **Configure service integrations**, select or deselect **Amazon S3** or **Zero-ETL**.

8. Choose **Continue**, and then choose **Modify**.

Deleting an ODB network in Oracle Database@AWS

You can delete an ODB network. This section assumes that you have already created an ODB network in Step 1: Create an ODB network in Oracle Database@AWS. You can't delete an ODB network that is currently in use by a VM cluster.

To delete an ODB network

1. Sign in to the AWS Management Console and open the Oracle Database@AWS console at https://console.aws.amazon.com/odb/.
2. From the left pane, choose **ODB networks**.
3. Select the network that you want to delete.
4. Choose **Delete**.
5. (Optional) Choose **Delete associated OCI resources** to delete the OCI resources that were created along with the ODB network.
6. In the text box, enter **`delete me`**.
7. Choose **Delete**.

Deleting a VM cluster in Oracle Database@AWS

You can delete an Exadata VM cluster or Autonomous VM cluster. This section assumes that you have already created a VM cluster in Step 3: Create an Exadata VM cluster or Autonomous VM cluster in Oracle Database@AWS.

To delete an VM cluster

1. Sign in to the AWS Management Console and open the Oracle Database@AWS console at https://console.aws.amazon.com/odb/.
2. From the left pane, choose **Exadata VM clusters** or **Autonomous VM clusters**.
3. Choose a VM cluster to delete.
4. Choose **Delete**.
5. When prompted, enter **`delete me`** and then choose **Delete**.

Deleting an Oracle Exadata infrastructure in Oracle Database@AWS

You can delete an Oracle Exadata infrastructure. This section assumes that you have already created an Oracle Exadata infrastructure in Step 2: Create an Oracle Exadata infrastructure in Oracle Database@AWS. You can't delete an Exadata infrastructure that is currently in use by a VM cluster.

To delete an Oracle Exadata infrastructure

1. Sign in to the AWS Management Console and open the Oracle Database@AWS console at https://console.aws.amazon.com/odb/.
2. From the left pane, choose **Exadata infrastructures**.
3. Choose an Exadata infrastructure to delete.
4. Choose **Delete**.
5. When prompted, enter **`delete me`** and then choose **Delete**.

Deleting an ODB peering connection

When you no longer need an ODB peering connection, you can delete it. You must delete all ODB peering connections before you can delete an ODB network.

Console

1. Sign in to the AWS Management Console and open the Oracle Database@AWS console at https://console.aws.amazon.com/odb/.
2. In the navigation pane, choose **ODB peering connections**.
3. Select the ODB peering connection to delete.
4. Choose **Delete**.
5. To confirm deletion, enter **`delete me`** and choose **Delete**.

AWS CLI

To delete an ODB peering connection, use the `delete-odb-peering-connection` command.

```
aws odb delete-odb-peering-connection \
    --odb-peering-connection-id odbpcx-1234567890abcdef
```

Backing up in Oracle Database@AWS

Oracle Database@AWS provides multiple backup options to protect your Oracle databases. You can use Oracle managed backups that integrate seamlessly with Amazon S3 or create your own user-managed backups using Oracle Recovery Manager (RMAN).

Oracle managed backups to Amazon S3

When you create an ODB network, Oracle Database@AWS automatically configures network access for Oracle managed backups to Amazon S3. OCI configures necessary DNS entries and security lists. These configurations allow traffic between the OCI Virtual Cloud Network (VCN) and Amazon S3. The ODB network doesn't enable or control automatic backups.

Oracle managed backups are fully managed by OCI. When you create your Oracle Exadata database, you can enable automatic backups by choosing **Enable automatic backups** in the OCI console. Choose one of the following backup destinations:

- **Amazon S3**
- **OCI Object Storage**
- **Autonomous Recovery Service**

For more information, see Backup Exadata Database in the OCI documentation.

User-managed backups to Amazon S3 in Oracle Database@AWS

With Oracle Database@AWS, you can create user-managed backups of your database using the Exadata Database Service on Dedicated Infrastructure. You back up your data with Oracle Recovery Manager (RMAN) and store it in your Amazon S3 buckets. You have full control over backup scheduling, retention policies, and storage costs while maintaining the managed service benefits of Oracle Database@AWS.

Note

Oracle Database@AWS doesn't support user-managed backups for Autonomous Database on Dedicated Infrastructure.

User-managed backups complement the AWS managed backup solutions provided by Oracle Database@AWS. You can use manual backups for compliance requirements, cross-Region disaster recovery, or integration with existing backup management workflows.

You can use the following user-managed backup techniques:

Storage Gateway

Use Storage Gateway for file-based backups that use an NFS share.

S3 mount point

Use a file client to mount an Amazon S3 bucket as a local file system.

Prerequisites for user-managed backups to Amazon S3 in Oracle Database@AWS

Before you can back up your Oracle Exadata databases to Amazon S3, do the following:

1. Enable direct access to Amazon S3 from your ODB network.
2. Configure network connectivity and routing between Oracle Database@AWS and Amazon S3.

Enabling access from your ODB network to Amazon S3

To back up your database manually to Amazon S3, enable direct access to S3 from your ODB network. This technique allows your databases to access Amazon S3 for your business needs, such as data import/export or user-managed backups. You have full control over the target destination of backup storage and can use policies to restrict access to Amazon S3 using VPC Lattice.

Direct access to Amazon S3 from your ODB network isn't enabled by default. You can enable S3 access when you create or modify your ODB network.

Console

To enable direct access to Amazon S3 from your ODB network

1. Open the Oracle Database@AWS console at https://console.aws.amazon.com/odb/.
2. In the navigation pane, choose **ODB networks**.
3. Select the ODB network for which you want to enable Amazon S3 access.

4. Choose **Modify**.
5. Select **Amazon S3**.
6. (Optional) Configure an Amazon S3 policy document to control access to Amazon S3. If you don't specify a policy, the default policy grants full access.
7. Choose **Continue** and then **Modify**.

AWS CLI

To enable direct Amazon S3 access from your ODB network, use the `update-odb-network` command with the `s3-access` parameter:

```
aws odb update-odb-network \
  --odb-network-id odb-network-id \
  --s3-access ENABLED
```

To configure an Amazon S3 policy document, use the `--s3-policy-document` parameter:

```
aws odb update-odb-network \
  --odb-network-id odb-network-id \
  --s3-policy-document file://s3-policy.json
```

When Amazon S3 access is enabled, you can access Amazon S3 from your ODB network by using the regional DNS `s3.`*`region`*`.amazonaws.com`. OCI configures this DNS name by default. To use a custom DNS name, modify your VCN DNS to ensure the custom DNS resolves to the IP address of the service network endpoint.

Configuring network connectivity between Oracle Database@AWS and Amazon S3

To allow user-managed backups to Amazon S3, your VM must be able to access the S3 Amazon VPC endpoint. In the OCI console, you can edit the security rules in a network security group (NSG) to control the ingress and egress traffic. For user-managed backups, traffic flows over the client subnet rather than the backup subnet. In the following steps, you update the NSGs for the client subnet to add the egress rule for the VPC endpoint IP address.

To allow VM access to the Amazon S3 endpoint

1. Open the Oracle Database@AWS console at https://console.aws.amazon.com/odb/.

2. Choose **ODB networks**.
3. Choose the name of the ODB network.
4. Choose **OCI resources**.
5. Choose the **Service integrations** tab.
6. Under **Amazon S3**, note the following pieces of information:
   - The IPv4 address of the Amazon VPC S3 endpoint. You need this information later. For example, the IP address might be `192.168.12.223`.
   - The domain name of the Amazon VPC S3 endpoint. You need this information later. For example, the domain name might be `s3.us-east-1.amazonaws.com`.
7. In the left navigation pane, choose Exadata VM clusters and then choose your VM cluster name.
8. At the top of the page, choose the **Summary** tab.
9. Choose **Virtual machines** and then choose the name of your VM.
10. Note the value in **DNS Name**. This is the host name that you specify when you connect to your VM using `ssh`.
11. In the top right, choose **Manage in OCI**. This opens the OCI console.
12. On the **Virtual Cloud Networks** list page, choose the VCN that contains the network security group (NSG) for the ODB network client subnet (`exa_static_nsg`). For more information, see Managing Security Rules for an NSG in the OCI documentation.
13. On the details page, perform one of the following actions depending on the option that you see:
   - On the **Security** tab, go to **Network Security Groups**.
   - Under **Resources**, choose **Network Security Groups**.
14. Choose the NSG for the client subnet (`exa_static_nsg`).
15. Add an egress rule for the VPC endpoint address that you noted earlier.

To test connectivity to S3 from your VM

1. Use `ssh` to connect as `root` to the VM whose DNS name you obtained previously. When you connect, specify a `.pem` file with your SSH keys.
2. Run the following commands to make sure that the VM can access the Amazon S3 Amazon VPC endpoint. Use the S3 domain name that you noted down previously.

```
# nslookup s3.us-east-1.amazonaws.com
# curl -v https://s3.us-east-1.amazonaws.com/
# aws s3 ls --endpoint-url https://s3.us-east-1.amazonaws.com
```

Backing up to Amazon S3 using AWS Storage Gateway on Amazon EC2

AWS Storage Gateway is a hybrid service that connects your on-premises environment to AWS Cloud storage services. For Oracle Database@AWS backups, you can use Storage Gateway to create a file-based backup workflow that writes directly to Amazon S3. You manage the lifecycle of the backups.

In this solution, you create a separate Amazon EC2 instance for configuring Storage Gateway. You also add an Amazon EBS volume to cache the reads and writes to Amazon S3.

This technique offers the following benefits:

- You don't require a media manager.
- No intermediate backup storage is necessary.

To deploy your Storage Gateway and create a file share

1. Open the AWS Management Console at https://console.aws.amazon.com/storagegateway/home/, and choose the AWS Region where you want to create your gateway.
2. Deploy and activate an Amazon S3 file gateway, using an Amazon EC2 instance as the hub. Follow the instructions in Deploy a customized Amazon EC2 host for S3 File Gateway in the *Storage Gateway User Guide*.

   When you configure your file gateway, make sure that you do the following:

   - Add at least one Amazon EBS volume for cache storage, with a size of at least 150 GiB.
   - Open TCP/UDP port 2049 for NFS access in your security group. This allows you to create NFS file shares.
   - Open TCP port 80 for inbound traffic to allow one-time HTTP access during gateway activation. After activation, you can close this port.

3. Create a Amazon VPC endpoint for private connectivity between your ODB network and the Storage Gateway. For more information, see Access an AWS service using an interface VPC endpoint.
4. Create a file share for your Amazon S3 bucket through the Storage Gateway console. For more information, see Creating a file share.

To back up your database to Amazon S3 using Storage Gateway

1. In a terminal, use `ssh` to connect to the DNS name of the Exadata VM. To find the DNS name, see Prerequisites for user-managed backups to Amazon S3 in Oracle Database@AWS.
2. Create a directory on the Exadata VM cluster server for the NFS mount. The following example creates the directory `/home/oracle/sgw_mount/`.

   ```
   mkdir /home/oracle/sgw_mount/
   ```

3. Mount the NFS share on the directory that you just created. The following example creates the share on the directory `/home/oracle/sgw_mount/`. Replace *`SG-IP-address`* with your Storage Gateway IP address and *`your-bucket-name`* with the name of your S3 bucket.

   ```
   sudo mount -t nfs -o nolock,hard SG-IP-address:/your-bucket-name /home/oracle/
   sgw_mount/
   ```

4. Connect to RMAN and back up the database to the mounted directory. The following example creates the channel `rman_local_bkp` and uses the mount point path to format the backup pieces.

   ```
   $ rman TARGET /
   RMAN> ALLOCATE CHANNEL rman_local_bkp DEVICE TYPE DISK;
   RMAN> BACKUP FORMAT '/home/oracle/sgw_mount/%U' DATABASE;
   ```

5. Verify that the backup files are created in the mount directory. The following example shows two backup pieces.

   ```
   $ ls -lart /home/oracle/sgw_mount/
   total 8569632
   -rw-r----- 1 oracle asmdba 1112223334 Jul 10 20:51 1a2b34cd_1234_1_1
   drwxrwxrwx 1 nobody nobody 0 Jul 10 20:56 .
   -rw-r----- 1 oracle asmdba 5556667778 Jul 10 20:56 1a2b34cd_1235_1_1
   ```

Backing up to Amazon S3 using an S3 mount point

You can use Amazon S3 mount point to create backups locally first and then copy them to Amazon S3. This technique creates backups on local storage and then transfers them to Amazon S3 using the mount point interface. The backup time is longer than in other techniques because you need to back up data twice.

Note

Direct backup to Amazon S3 using the mount point, without staging, isn't supported. RMAN requires specific file system permissions that aren't compatible with the Amazon S3 mount point interface.

This technique doesn't require you to license a media manager. You manage the lifecycle of your backups.

To back up to Amazon S3 using an S3 mount point

1. In a terminal, use `ssh` to connect to the DNS name of the Exadata VM. To find the DNS name, see Prerequisites for user-managed backups to Amazon S3 in Oracle Database@AWS.
2. Install the Amazon S3 mount point on the Exadata VM cluster server. For more information about installation and configuration, see Mountpoint for Amazon S3 in the *Amazon S3 User Guide*.

```
$ sudo yum install ./mount-s3.rpm
```

3. Verify the installation by running the `mount-s3` command.

```
$ mount-s3 --version
mount-s3 1.19.0
```

4. Create an intermediate backup directory on the Exadata VM cluster server local storage. You will back up your database to this local directory and then copy the backup to your S3 bucket. The following example creates directory `/u02/rman_bkp_local`.

```
mkdir /u02/rman_bkp_local
```

5. Create a directory for the Amazon S3 mount point. The following example creates directory /home/oracle/s3mount.

   ```
   $ mkdir /home/oracle/s3mount
   ```

6. Mount your Amazon S3 bucket using the mount point. The following example mounts an S3 bucket on directory /home/oracle/s3mount. Replace *your-s3-bucket-name* with your actual Amazon S3 bucket name.

   ```
   $ mount-s3 s3://your-s3-bucket-name /home/oracle/s3mount
   ```

7. Verify that you can access the Amazon S3 bucket contents.

   ```
   $ ls -lart /home/oracle/s3mount
   ```

8. Connect RMAN to your target database and back it up to your local staging directory. The following example creates the channel rman_local_bkp and uses the path /u02/rman_bkp_local/ to format the backup pieces.

   ```
   $ rman TARGET /

   RMAN> ALLOCATE CHANNEL rman_local_bkp DEVICE TYPE DISK;
   RMAN> BACKUP FORMAT '/u02/rman_bkp_local/%U' DATABASE;
   ```

9. Verify that the backups are created in the local directory:

   ```
   $ cd /u02/rman_bkp_local/
   $ ls -lart
   total 4252128
   drwxr-xr-x 8 oracle oinstall 4096 Jul 10 02:13 ..
   -rw-r----- 1 oracle asmdba 1112223334 Jul 10 02:13 abcd1234_1921_1_1
   drwxr-xr-x 2 oracle oinstall 4096 Jul 10 02:13 .
   -rw-r----- 1 oracle asmdba 5556667778 Jul 10 02:14 abcd1234_1922_1_1
   ```

10. Copy the backup files from the local staging directory to the Amazon S3 mount point.

   ```
   cp /u02/rman_bkp_local/* /home/oracle/s3mount/
   ```

11. Verify that you copied the files successfully to Amazon S3.

   ```
   $ ls -lart /home/oracle/s3mount/
   ```

```
total 4252112
drwx------ 6 oracle oinstall 225 Jul 10 02:09 ..
drwxr-xr-x 2 oracle oinstall 0 Jul 10 02:24 .
-rw-r--r-- 1 oracle oinstall 1112223334 Jul 10 02:24 abcd1234_1921_1_1
-rw-r--r-- 1 oracle oinstall 5556667778 Jul 10 02:24 abcd1234_1922_1_1
```

Disabling direct access to Amazon S3

If you no longer need direct access to Amazon S3 from your ODB network, you can disable it. Enabling or disabling direct network access to S3 doesn't affect network access to Oracle managed backups to Amazon S3.

Console

To disable direct access to Amazon S3

1. Open the Oracle Database@AWS console at https://console.aws.amazon.com/odb/.
2. In the navigation pane, choose **ODB networks**.
3. Select the ODB network for which you want to disable Amazon S3 access.
4. Choose **Modify**.
5. Clear the **Enable S3 access** checkbox.
6. Choose **Modify ODB network**.

AWS CLI

Use the `update-odb-network` command with the `s3-access` parameter.

```
aws odb update-odb-network \
  --odb-network-id odb-network-id \
  --s3-access DISABLED
```

Troubleshooting the Amazon S3 integration

If you encounter issues with Oracle managed backups to Amazon S3 or direct access to Amazon S3, consider the following troubleshooting steps:

Cannot access Amazon S3 from your database

Check the following:

- Verify that Amazon S3 access is enabled for your ODB network. Use the `GetOdbNetwork` action to check whether the `s3Access` status is `Enabled`.
- Ensure you are using the correct regional DNS name: `s3.`*`region`*`.amazonaws.com`.
- Check that your Oracle database has the necessary permissions to access Amazon S3.

Oracle managed backups failing

Check the following:

- Oracle managed backups to Amazon S3 are enabled by default and cannot be disabled. If backups are failing, check the Oracle database logs for specific error messages.
- Verify that the Amazon VPC Lattice resources are properly configured by viewing the service integration resources.
- Contact Oracle Support for assistance with Oracle managed automatic backup issues. For more information, see Getting support for Oracle Database@AWS.

Oracle Database@AWS Zero-ETL integration with Amazon Redshift

Zero-ETL integration is a fully managed solution that makes transactional and operational data available in Amazon Redshift from multiple sources. With this solution, you can replicate data to Amazon Redshift from your Oracle databases running on Oracle Exadata or Autonomous Database on Dedicated Exadata Infrastructure. The automatic synchronization avoids the traditional extract, transform, and load (ETL) process. It also enables real-time analytics and AI workloads. For more information, see Zero-ETL integrations in the *Amazon Redshift Management Guide*.

Zero-ETL integration provides the following benefits:

- **Real-time data replication** – Continuous data synchronization from Oracle databases to Amazon Redshift with minimal latency
- **Elimination of complex ETL pipelines** – No need to build and maintain custom data integration solutions
- **Reduced operational overhead** – Automated setup and management through AWS APIs
- **Simplified data integration architecture** – Seamless integration between Oracle Database@AWS and AWS analytics services
- **Enhanced security** – Built-in encryption and AWS IAM access controls

Amazon Redshift doesn't charge an additional fee for the zero-ETL integration with Oracle Database@AWS. You pay for the existing Amazon Redshift resources used to create and process the change data created as part of a zero-ETL integration. For more information, see Amazon Redshift pricing.

Supported database versions for Zero-ETL integration in Oracle Database@AWS

Zero-ETL integration supports the following Oracle database versions:

- **Oracle Exadata** – Oracle Database 19c
- **Autonomous Database on Dedicated Infrastructure** – Oracle Database 19c and 23ai

How Zero-ETL integration works in Oracle Database@AWS

Zero-ETL integration allows Oracle Database@AWS to replicate data to Amazon Redshift. The integration leverages Amazon VPC Lattice to create secure network connectivity. Change data capture (CDC) technology ensures real-time data synchronization. You manage the integration through AWS Glue APIs.

The Zero-ETL integration architecture includes the following:

- **Secure connectivity –** Uses SSL/TLS encryption over TLS port 2484 for data transfer
- **AWS Secrets Manager –** Stores database credentials and certificates securely using AWS Key Management Service
- **AWS Glue integration –** Provides unified management interface for zero-ETL integrations

Replication proceeds through the following steps:

1. Establishing secure connection to the Oracle database using SSL on port 2484
2. Performing an initial full dump of selected databases, schemas, and tables
3. Setting up change data capture (CDC) for ongoing real-time replication
4. Writing the replicated data to the target Amazon Redshift cluster

Important

Zero-ETL integration isn't enabled by default. You must configure it using AWS Glue APIs. You can't set up zero-ETL integration directly using Oracle Database@AWS APIs.

Prerequisites for zero-ETL integration in Oracle Database@AWS

Before setting up zero-ETL integration, ensure that you meet the following prerequisites.

General prerequisites

- **Oracle Database@AWS setup –** Make sure you have at least one VM cluster provisioned and running.

- **Integration with zero-ETL enabled** – Make sure your VM cluster or Autonomous VM cluster is associated with an ODB network that has zero-ETL enabled.
- **Supported Oracle Database versions** – You must use Oracle Database 19c (Oracle Exadata) or Oracle Database 19c/23ai (Autonomous Database on Dedicated Infrastructure).
- **Same AWS Region** – The source Oracle database and target Amazon Redshift cluster must be in the same AWS Region.

Oracle database prerequisites

You must configure your Oracle database with the following settings.

Replication user setup

Create a dedicated replication user in each pluggable database (PDB) that you want to replicate:

- **For Oracle Exadata** – Create user `ODBZEROETLADMIN` with a secure password.
- **For Autonomous Database on Dedicated Infrastructure** – Use the existing `GGADMIN` user.

Grant the following permissions to the replication user.

```
-- For Autonomous Database on Dedicated Infrastructure only
ALTER USER GGADMIN ACCOUNT UNLOCK;
ALTER USER GGADMIN IDENTIFIED BY ggadmin-password;

-- For Oracle Exadata only
GRANT SELECT ON any-replicated-table TO "ODBZEROETLADMIN";
GRANT LOGMINING to "ODBZEROETLADMIN";

-- Grant the following permissions to all services.
-- For Oracle Exadata, use the ODBZEROETLADMIN user. For Autonomous Database on
 Dedicated Infrastructure,
-- use the GGADMIN user.
GRANT CREATE SESSION TO "ODBZEROETLADMIN";
GRANT SELECT ANY TRANSACTION TO "ODBZEROETLADMIN";
GRANT SELECT ON V_$ARCHIVED_LOG TO "ODBZEROETLADMIN";
GRANT SELECT ON V_$LOG TO "ODBZEROETLADMIN";
GRANT SELECT ON V_$LOGFILE TO "ODBZEROETLADMIN";
GRANT SELECT ON V_$LOGMNR_LOGS TO "ODBZEROETLADMIN";
GRANT SELECT ON V_$LOGMNR_CONTENTS TO "ODBZEROETLADMIN";
GRANT SELECT ON V_$DATABASE TO "ODBZEROETLADMIN";
GRANT SELECT ON V_$THREAD TO "ODBZEROETLADMIN";
GRANT SELECT ON V_$PARAMETER TO "ODBZEROETLADMIN";
GRANT SELECT ON V_$NLS_PARAMETERS TO "ODBZEROETLADMIN";
GRANT SELECT ON V_$TIMEZONE_NAMES TO "ODBZEROETLADMIN";
GRANT SELECT ON V_$TRANSACTION TO "ODBZEROETLADMIN";
GRANT SELECT ON V_$CONTAINERS TO "ODBZEROETLADMIN";
GRANT SELECT ON ALL_INDEXES TO "ODBZEROETLADMIN";
GRANT SELECT ON ALL_OBJECTS TO "ODBZEROETLADMIN";
GRANT SELECT ON ALL_TABLES TO "ODBZEROETLADMIN";
GRANT SELECT ON ALL_USERS TO "ODBZEROETLADMIN";
GRANT SELECT ON ALL_CATALOG TO "ODBZEROETLADMIN";
GRANT SELECT ON ALL_CONSTRAINTS TO "ODBZEROETLADMIN";
GRANT SELECT ON ALL_CONS_COLUMNS TO "ODBZEROETLADMIN";
GRANT SELECT ON ALL_TAB_COLS TO "ODBZEROETLADMIN";
GRANT SELECT ON ALL_IND_COLUMNS TO "ODBZEROETLADMIN";
GRANT SELECT ON ALL_ENCRYPTED_COLUMNS TO "ODBZEROETLADMIN";
GRANT SELECT ON ALL_LOG_GROUPS TO "ODBZEROETLADMIN";
GRANT SELECT ON ALL_TAB_PARTITIONS TO "ODBZEROETLADMIN";
GRANT SELECT ON SYS.DBA_REGISTRY TO "ODBZEROETLADMIN";
GRANT SELECT ON SYS.OBJ$ TO "ODBZEROETLADMIN";
GRANT SELECT ON DBA_TABLESPACES TO "ODBZEROETLADMIN";
GRANT SELECT ON DBA_OBJECTS TO "ODBZEROETLADMIN";
GRANT SELECT ON SYS.ENC$ TO "ODBZEROETLADMIN";
GRANT SELECT ON GV_$TRANSACTION TO "ODBZEROETLADMIN";
GRANT SELECT ON V_$DATAGUARD_STATS TO "ODBZEROETLADMIN";
GRANT SELECT ON V_$DATABASE_INCARNATION TO "ODBZEROETLADMIN";
GRANT EXECUTE ON SYS.DBMS_CRYPTO TO "ODBZEROETLADMIN";
GRANT SELECT ON SYS.DBA_DIRECTORIES TO "ODBZEROETLADMIN";
GRANT SELECT ON ALL_VIEWS TO "ODBZEROETLADMIN";
GRANT SELECT ON DBA_SEGMENTS TO "ODBZEROETLADMIN";
GRANT SELECT ON V_$TRANSPORTABLE_PLATFORM TO "ODBZEROETLADMIN";
GRANT CREATE ANY DIRECTORY TO "ODBZEROETLADMIN";
GRANT EXECUTE ON DBMS_FILE_TRANSFER TO "ODBZEROETLADMIN";
GRANT EXECUTE ON DBMS_FILE_GROUP TO "ODBZEROETLADMIN";
GRANT EXECUTE on DBMSLOGMNR to "ODBZEROETLADMIN";
GRANT SELECT on V_$LOGMNRLOGS to "ODBZEROETLADMIN";
GRANT SELECT on V_$LOGMNRCONTENTS to "ODBZEROETLADMIN";
GRANT LOGMINING to "ODBZEROETLADMIN";
GRANT SELECT ON GV_$CELL_STATE TO "ODBZEROETLADMIN";
```

Supplemental logging

Enable supplemental logging on your Oracle database to capture change data.

```
-- Check if supplemental logging is enabled
SELECT supplemental_log_data_min FROM v$database;

-- Enable supplemental logging if not already enabled.
-- For Oracle Exadata, enable supplemental logging on both the CDB and PDB.
-- For Autonomous Database on Dedicated Infrastructure, enable supplemental logging on
 the PDB only.
ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;

-- For Autonomous Database on Dedicated Infrastructure only
ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY) COLUMNS;

-- Archive current online redo log
ALTER SYSTEM ARCHIVE LOG CURRENT;
```

To set up a zero-ETL integration between Oracle Database@AWS and Amazon Redshift, you must configure SSL.

For Oracle Exadata databases

You must manually configure SSL on port 2484. This task involves the following:

- Configuring `(PROTOCOL=tcps)(PORT=2484)` in `listener.ora`
- Setting up the wallet using `sqlnet.ora`
- Generating and configuring SSL certificates (see How To Configure SSL/TCPS For Exadata Cloud Database (ExaCC/ExaCS) (Doc ID 2947301.1) in the My Oracle Support documentation)

For Autonomous Databases

SSL on port 2484 is enabled by default. No additional configuration is required.

Important

The SSL port is fixed as 2484.

AWS service prerequisites

Before setting up zero-ETL integration, set up AWS Secrets Manager and configure IAM permissions.

Set up AWS Secrets Manager

Store your Oracle database credentials in AWS Secrets Manager as follows:

1. Create a Customer Managed Key (CMK) in AWS Key Management Service.
2. Store database credentials in AWS Secrets Manager using the CMK.
3. Configure resource policies to allow Oracle Database@AWS access.

To get your TDE key ID and password, use the technique described in Supported encryption methods for using Oracle as a source for AWS Database Migration Service. The following command generates the base64 wallet.

```
base64 -i cwallet.sso > wallet.b64
```

The following example shows a secret for Oracle Exadata. For *asm_service_name*, the *111.11.11.11* represents the virtual IP for the VM node. You can also register the ASM listener with SCAN.

```
{
  "database_info": [
    {
      "name": "ODBDB_ZETLPDB",
      "service_name": "ODBDB_ZETLPDB.paas.oracle.com",
      "username": "ODBZEROETLADMIN",
      "password": "secure_password",
      "tde_key_id": "ORACLE.SECURITY.DB.ENCRYPTION.key_id",
      "tde_password": "tde_password",
      "certificateWallet": "base64_encoded_wallet_content"
    }
  ],
  "asm_info": {
    "asm_user": "odbzeroetlasm",
    "asm_password": "secure_password",
    "asm_service_name": "111.11.11.11:2484/+ASM"
  }
}
```

The following example shows a secret for Autonomous Database on Dedicated Infrastructure.

```
{
  "database_info": [
    {
      "database_name": "ZETLACD_ZETLADBMORECPU",
      "service_name": "ZETLADBMORECPU_high.adw.oraclecloud.com",
      "username": "ggadmin",
      "password": "secure_password",
      "certificateWallet": "base64_encoded_wallet_content"
    }
  ]
}
```

Configure IAM permissions

Create IAM policies that allow zero-ETL integration operations. The following example policy allows describe, create, update, and delete operations for an Exadata VM cluster. For an Autonomous VM cluster, use the value `cloud-autonomous-vm-cluster` instead of `cloud-vm-cluster` for the resource ARN.

Considerations for zero-ETL integration in Oracle Database@AWS

When setting up Zero-ETL integration between Oracle Database@AWS and Amazon Redshift, consider the following guidelines:

Initial data load time

The initial full load time depends on the size of your database. Large databases might take several hours or days to complete the initial synchronization.

Oracle database performance

Change data capture might impact Oracle database performance, especially during high transaction volumes. After enabling Zero-ETL integration, monitor your database performance.

Schema changes

Data Definition Language (DDL) changes in the source Oracle database might require you to intervene manually to re-create the integration. Plan schema changes carefully.

For general considerations, see Considerations when using zero-ETL integrations with Amazon Redshift.

Limitations for zero-ETL integration in Oracle Database@AWS

Note the following general limitations:

Single PDB per integration

Each zero-ETL integration can only replicate data from one pluggable database (PDB). Data filters like `include: pdb1.*.*, include: pdb2.*.*` aren't supported.

Single integration per Autonomous Database or Exadata Infrastructure

Each zero-ETL integration can only replicate data from one Autonomous Database on Dedicated Infrastructure.

Fixed SSL port

SSL connections must use port 2484.

Same Region requirement

The source Oracle Database@AWS VM cluster and target Amazon Amazon Redshift cluster must be in the same AWS Region. Cross-region replication isn't supported.

No mTLS support

Mutual TLS (mTLS) isn't supported. If your OCI database has mTLS enabled, you must disable it to use zero-ETL integration.

Immutable integration settings

After you create the secret ARN or KMS key associated with an integration, you can't modify it. You must delete and re-create the integration to change these settings.

TDE column-level encryption

Column-level Transparent Data Encryption (TDE) isn't supported for Oracle Exadata databases. Only tablespace-level TDE is supported.

Data type support

Some Oracle-specific data types might not be fully supported or might require transformation during replication. Test your specific data types thoroughly before you deploy your database to production.

Setting up Oracle Database@AWS integrations with Amazon Redshift

To set up zero-ETL integration between your Oracle database and Amazon Redshift, complete the following steps:

1. Enable Zero-ETL on your ODB network.
2. Configure Oracle database prerequisites.
3. Set up AWS Secrets Manager and AWS Key Management Service.
4. Configure IAM permissions.
5. Set up Amazon Redshift resource policies.
6. Create the zero-ETL integration.
7. Create the target database in Amazon Redshift.

Step 1: Enable Zero-ETL for your ODB network

You can enable the zero-ETL integration for the ODB network associated with your source VM cluster. By default, this integration is disabled.

Console

To enable zero-ETL integration

1. Open the Oracle Database@AWS console at https://console.aws.amazon.com/odb/.
2. In the navigation pane, choose **ODB networks**.
3. Select the ODB network for which you want to enable the zero-ETL integration.
4. Choose **Modify**.
5. Select **Zero-ETL**.
6. Choose **Continue** and then **Modify**.

AWS CLI

To enable the zero-ETL integration, use the `update-odb-network` command with the `--zero-etl-access` parameter:

```
aws odb update-odb-network \
  --odb-network-id odb-network-id \
  --zero-etl-access ENABLED
```

To enable zero-ETL integration for the ODB network associated with your source VM cluster, use the `update-odb-network` command. This command configures the network infrastructure required for zero-ETL integration.

```
aws odb update-odb-network \
  --odb-network-id your-odb-network-id \
  --zero-etl-access ENABLED
```

Step 2: Configure your Oracle database

Complete the Oracle database configuration as described in the Prerequisites:

- Create replication users and grant necessary permissions.
- Enable archived redo logs.
- Configure SSL (Oracle Exadata only).
- Set up ASM users if applicable (Oracle Exadata only).

Step 3: Set up AWS Secrets Manager and AWS Key Management Service

Create a Customer Managed Key (CMK) and store your database credentials.

1. Create a CMK in AWS Key Management Service using the `create-key` command.

```
aws kms create-key \
  --description "ODB Zero-ETL Integration Key" \
  --key-usage ENCRYPT_DECRYPT \
  --key-spec SYMMETRIC_DEFAULT
```

2. Store your database credentials in AWS Secrets Manager.

```
aws secretsmanager create-secret \
  --name "ODBZeroETLCredentials" \
  --description "Credentials for Oracle Database@AWS Zero-ETL integration" \
  --kms-key-id your-cmk-key-arn \
  --secret-string file://secret-content.json
```

3. Attach a resource policy to the secret to allow Oracle Database@AWS access.

   ```
   aws secretsmanager put-resource-policy \
     --secret-id "ODBZeroETLCredentials" \
     --resource-policy file://secret-resource-policy.json
   ```

   In the preceding command, `secret-resource-policy.json` contains the following JSON.

   JSON

   ```
   {
     "Version":"2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
           "Service": "zetl.odb.amazonaws.com"
         },
         "Action": [
           "secretsmanager:GetSecretValue",
           "secretsmanager:DescribeSecret"
         ],
         "Resource": "*"
       }
     ]
   }
   ```

4. Attach a resource policy to the CMK. The CMK resource policy must include permissions for both the Oracle Database@AWS service principal and the Amazon Redshift service principal to support encrypted Zero-ETL integration.

   ```
   aws kms put-key-policy \
     --key-id your-cmk-key-arn \
     --policy-name default \
     --policy file://cmk-resource-policy.json
   ```

   The `cmk-resource-policy.json` file should include the following policy statements. The first statement allows Oracle Database@AWS service access, and the second statement allows Amazon Redshift to create grants on the KMS key for encrypted data operations.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "Allow ODB service access",
      "Effect": "Allow",
      "Principal": {
        "Service": "zetl.odb.amazonaws.com"
      },
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey",
        "kms:CreateGrant"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Allows the Redshift service principal to add a grant to a KMS
 key",
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift.amazonaws.com"
      },
      "Action": "kms:CreateGrant",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:EncryptionContext:{context-key}": "{context-value}"
        },
        "ForAllValues:StringEquals": {
          "kms:GrantOperations": [
            "Decrypt",
            "GenerateDataKey",
            "CreateGrant"
          ]
        }
      }
    }
  ]
}
```

Step 4: Configure IAM permissions

Create and attach IAM policies that allow zero-ETL integration operations.

```
aws iam create-policy \
  --policy-name "ODBZeroETLIntegrationPolicy" \
  --policy-document file://odb-zetl-iam-policy.json

aws iam attach-user-policy \
  --user-name your-iam-username \
  --policy-arn policy-arn
```

The following policy grants the necessary permissions.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ODBGlueIntegrationAccess",
      "Effect": "Allow",
      "Action": [
        "glue:CreateIntegration",
        "glue:ModifyIntegration",
        "glue:DeleteIntegration",
        "glue:DescribeIntegrations",
        "glue:DescribeInboundIntegrations"
      ],
      "Resource": "*"
    },
    {
      "Sid": "ODBZetlOperations",
      "Effect": "Allow",
      "Action": "odb:CreateOutboundIntegration",
      "Resource": "*"
    },
    {
      "Sid": "ODBRedshiftFullAccess",
      "Effect": "Allow",
      "Action": [
        "redshift:*",
        "redshift-serverless:*",
        "ec2:DescribeAccountAttributes",
        "ec2:DescribeAddresses",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcs",
        "ec2:DescribeInternetGateways",
        "sns:CreateTopic",
        "sns:Get*",
        "sns:List*",
        "cloudwatch:Describe*",
         "cloudwatch:Get*",
        "cloudwatch:List*",
        "cloudwatch:PutMetricAlarm",
        "cloudwatch:EnableAlarmActions",
        "cloudwatch:DisableAlarmActions",
        "tag:GetResources",
        "tag:UntagResources",
        "tag:GetTagValues",
        "tag:GetTagKeys",
        "tag:TagResources"
      ],
      "Resource": "*"
    },
    {
      "Sid": "ODBRedshiftDataAPI",
      "Effect": "Allow",
      "Action": [
        "redshift-data:ExecuteStatement",
        "redshift-data:CancelStatement",
        "redshift-data:ListStatements",
        "redshift-data:GetStatementResult",
        "redshift-data:DescribeStatement",
        "redshift-data:ListDatabases",
        "redshift-data:ListSchemas",
        "redshift-data:ListTables",
        "redshift-data:DescribeTable"
      ],
      "Resource": "*"
    },
    {
      "Sid": "ODBKMSAccess",
      "Effect": "Allow",
      "Action": [
        "kms:CreateKey",
        "kms:DescribeKey",
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:GenerateDataKey",
        "kms:ListKeys",
        "kms:CreateAlias",
        "kms:ListAliases"
      ],
      "Resource": "*"
    },
    {
      "Sid": "ODBSecretsManagerAccess",
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:PutSecretValue",
        "secretsmanager:CreateSecret",
        "secretsmanager:UpdateSecret",
        "secretsmanager:DeleteSecret",
        "secretsmanager:DescribeSecret",
        "secretsmanager:ListSecrets",
        "secretsmanager:GetResourcePolicy",
        "secretsmanager:PutResourcePolicy",
        "secretsmanager:ValidateResourcePolicy"
      ],
      "Resource": "*"
    }
  ]
}
```

Step 5: Configure Amazon Redshift resource policies

Set up resource policies on your Amazon Redshift cluster to authorize inbound integrations.

```
aws redshift put-resource-policy \
  --no-verify-ssl \
  --resource-arn "your-redshift-cluster-arn" \
  --policy '{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
          "Service": "redshift.amazonaws.com"
        },
        "Action": [
          "redshift:AuthorizeInboundIntegration"
        ],
        "Condition": {
          "StringEquals": {
            "aws:SourceArn": "your-vm-cluster-arn"
          }
        }
      },
      {
        "Effect": "Allow",
        "Principal": {
          "AWS": "your-account-id"
        },
        "Action": [
          "redshift:CreateInboundIntegration"
        ]
      }
    ]
  }' \
  --region us-west-2
```

Tip

Alternatively, you can use the **Fix it for me** option in the AWS console. This option automatically configures the required Amazon Redshift policies without your needing to do it manually.

Step 6: Create the zero-ETL integration using AWS Glue

Create the zero-ETL integration using the AWS Glue `create-integration` command. In this command, you specify the source VM cluster and the target Amazon Redshift namespace.

The following example creates an integration with a PDB named `pdb1` running in an Exadata VM cluster. You can also create an Autonomous VM cluster by replacing `cloud-vm-cluster` with `cloud-autonomous-vm-cluster` in the source ARN. Specifying a KMS key is optional. If you specify a key, it can be different from the one that you created in Step 3: Set up AWS Secrets Manager and AWS Key Management Service.

```
aws glue create-integration \
  --integration-name "MyODBZeroETLIntegration" \
  --source-arn "arn:aws:odb:region:account:cloud-vm-cluster/cluster-id" \
  --target-arn "arn:aws:redshift:region:account:namespace/namespace-id" \
  --data-filter "include: pdb1.*.*" \
  --integration-config '{
      "RefreshInterval": "10",
      "IntegrationMode": "DEFAULT",
      "SourcePropertiesMap": {
        "secret-arn": "arn:aws:secretsmanager:region:account:secret:secret-name"
      }
    }' \
  --description "Zero-ETL integration for Oracle to Amazon Redshift" \
  --kms-key-id "arn:aws:kms:region:account:key/key-id"
```

The command returns an integration ARN and sets the status to `creating`. You can monitor the integration status using the `describe-integrations` command.

```
aws glue describe-integrations \
  --integration-identifier integration-id
```

Important

Only one PDB per integration is supported. The data filter must specify a single PDB, for example, `include: pdb1.*.*`. The source must be in the same AWS Region and account in which the integration is being created.

Step 7: Create a target database in Amazon Redshift

After the integration is active, create a target database in your Amazon Redshift cluster.

```
-- Connect to your Amazon Redshift cluster
```

```
psql -h your-redshift-endpoint -U username -d database

-- Create database from integration
CREATE DATABASE target_database_name
FROM INTEGRATION 'integration-id'
DATABASE "source_pdb_name";
```

After creating the target database, you can query the replicated data.

```
-- List databases to verify creation
\l

-- Connect to the new database
\c target_database_name

-- List tables to see replicated data
\dt
```

Verify the zero-ETL integration

Verify that the integration works by querying the integration status in AWS Glue and making sure that your Oracle changes are being replicated to Amazon Redshift.

To verify that your zero-ETL integration is working correctly

1. Check the integration status.

   ```
   aws glue describe-integrations \
     --integration-identifier integration-id
   ```

   The status should be `ACTIVE` or `REPLICATING`.

2. Verify data replication by making changes in your Oracle database and checking that they appear in Amazon Redshift.

3. Monitor replication metrics in Amazon CloudWatch (if available).

Data filtering for zero-ETL integrations in Oracle Database@AWS

Oracle Database@AWS zero-ETL integrations support data filtering. You can use it to control which data your source Oracle Exadata database replicates to your target data warehouse. Instead of replicating the entire database, you can apply one or more filters to selectively include or exclude specific tables. This helps you optimize storage and query performance by ensuring that only relevant data is transferred. Filtering is limited to the database and table levels. Column- and row-level filtering aren't supported.

Oracle Database and Amazon Redshift handle object name casing differently, which affects both data filter configuration and target queries. Note the following:

- Oracle Database stores database, schema, and object names in uppercase unless explicitly quoted in the `CREATE` statement. For example, if you create `mytable` (no quotes), the Oracle data dictionary stores the table name as `MYTABLE`. If you quote the object name in your creation statement, the Oracle data dictionary preserves the case.
- Zero-ETL data filters are case sensitive and must match the exact case of object names as they appear in the Oracle data dictionary. For example, if the Oracle dictionary stores schema and table name `REINVENT.MYTABLE`, then create a filter using `include: ORCL.REINVENT.MYTABLE`.
- Amazon Redshift queries default to lowercase object names unless explicitly quoted. For example, a query of `MYTABLE` (no quotes) searches for `mytable`.

Be mindful of the case differences when you create the Amazon Redshift filter and query the data. The filtering considerations for Oracle Database@AWS are the same as for Amazon RDS for Oracle. For examples of how case can affect data filters in an Oracle database, see RDS for Oracle examples in the *Amazon Relational Database Service User Guide*.

Monitoring zero-ETL integration

Regular monitoring of your zero-ETL integration ensures optimal performance and helps identify issues early.

Integration status monitoring

Monitor the status of your zero-ETL integrations using AWS Glue APIs.

```
# Check status of a specific integration
aws glue describe-integrations \
  --integration-identifier integration-id

# List all integrations in your account
aws glue describe-integrations
```

Integration statuses include:

- **creating** – Integration is being set up
- **active** – Integration is running and replicating data
- **modifying** – Integration configuration is being updated
- **needs_attention** – Integration requires manual intervention
- **failed** – Integration has encountered an error
- **deleting** – Integration is being removed

Performance monitoring

Monitor the following aspects of your zero-ETL integration performance:

- **Replication lag** – The time difference between when a change occurs in Oracle and when it appears in Amazon Redshift
- **Data throughput** – The volume of data being replicated per unit of time
- **Error rates** – The frequency of replication errors or failures
- **Resource utilization** – CPU, memory, and network usage on both source and target systems

Use Amazon CloudWatch to monitor these metrics and set up alarms for critical thresholds.

Managing zero-ETL integrations in Oracle Database@AWS

After creating a zero-ETL integration, you can perform various management operations including modifying and deleting integrations. This section covers the ongoing management of your zero-ETL integrations.

Modifying zero-ETL integrations

You can modify only the name, description, and data filtering options for a zero-ETL integration in a supported data warehouse. You can't modify the AWS Key Management Service key used to encrypt the integration, or the source or target databases.

Prerequisites for modifying integrations

Before you modify a zero-ETL integration, ensure that you have the following:

- **Required permissions** – Your IAM user or role must have the `odb:UpdateOutboundIntegration` permission in addition to the standard AWS Glue permissions.
- **Integration in active state** – The integration must be in an `ACTIVE` state, not in `CREATING`, `MODIFYING`, `DELETING`, or `FAILED`.
- **Valid data filter syntax** – New data filters must follow the supported include/exclude pattern syntax.

Modifying data filters

You can change which tables or schemas are replicated by modifying the data filter. In this way, you can add or remove database objects from replication without recreating the entire integration.

To modify the data filter for an integration, use the `modify-integration` command.

```
aws glue modify-integration \
  --integration-identifier integration-id \
  --data-filter "include: pdb1.new_schema.*"
```

You can also modify the integration name and description at the same time. In the following example, you modify the integration name, descriptions, and filters for two schemas in pdb1.

```
aws glue modify-integration \
  --integration-identifier integration-id \
  --data-filter "include: pdb1.schema1.*, pdb1.schema2.*" \
  --integration-name "Updated Integration Name" \
  --description "Updated integration description"
```

Important

When you modify the data filter, the integration enters a `modifying` state and performs a resynchronization of data. The integration stops replication, applies the new filter settings, and resumes replication with a reload-target operation. Monitor the integration status to ensure the modification completes successfully.

Considerations for data filter modifications to zero-ETL integrations

Consider the following when modifying data filters:

- **Single PDB limitation** – You can only specify one pluggable database (PDB) per integration. Data filters like `include: pdb1.*.*, include: pdb2.*.*` aren't supported
- **Replication interruption** – Data replication stops during the modification process and resumes after the new filter is applied.
- **Data reload** – The integration performs a full reload of data that matches the new filter criteria.
- **Performance impact** – Large data filter changes might take significant time to complete and can affect the source database performance during the reload.

Limitations for modifications to zero-ETL integration settings

You can't modify the following settings after you create a zero-ETL integration:

- **Secret ARN** – The AWS Secrets Manager secret containing database credentials
- **KMS key** – The customer managed key used for encryption
- **Source ARN** – The Oracle Database@AWS VM cluster
- **Target ARN** – The Amazon Redshift cluster or namespace

To change these settings, delete the existing zero-ETL integration and create a new one.

Deleting zero-ETL integrations

When you no longer need a zero-ETL integration, you can delete it to stop replication and clean up associated resources.

Deletion using AWS Glue

Delete a zero-ETL integration using the AWS Glue API.

```
aws glue delete-integration \
  --integration-identifier integration-id
```

You can delete integrations in the following states:

- **active**
- **needs_attention**
- **failed**
- **syncing**

Effects of deletion

When you delete a zero-ETL integration, consider the following effects:

Replication stops.

Oracle Database@AWS doesn't replicate new changes from Amazon Redshift.

Existing data is preserved.

Data already replicated to Amazon Redshift remains available.

The target database remains.

The Amazon Redshift database created from the integration isn't automatically deleted.

Important

Deletion is irreversible. If you need to resume replication after deletion, create a new integration, which performs a full initial load.

Best practices for zero-ETL management

Follow these best practices to ensure optimal performance, security, and cost-effectiveness of your zero-ETL integrations.

Operational best practices

These operational practices help maintain reliable and efficient zero-ETL integrations.

Regular monitoring

Set up CloudWatch alarms to monitor integration health and performance metrics.

Credential rotation

Regularly rotate database passwords and update them in AWS Secrets Manager.

Backup verification

Regularly verify that your Oracle database backups include the necessary components for disaster recovery.

Performance testing

Test the impact of zero-ETL integration on your Oracle database performance, especially during peak usage periods.

Schema change planning

Plan and test schema changes in a development environment before applying them to production.

Security best practices

Implement these security measures to protect your zero-ETL integration and data.

Least privilege access

Grant only the minimum necessary permissions to replication users and AWS IAM roles.

Network security

Use security groups and NACLs to restrict network access to only required ports and sources.

Encryption at rest

Ensure that both Oracle databases and Amazon Redshift clusters use encryption at rest.

Audit logging

Enable audit logging on both Oracle and Amazon Redshift to track data access and changes.

Secret management

Use AWS Secrets Manager automatic rotation features where possible.

Cost optimization

Apply these strategies to optimize costs while maintaining effective zero-ETL integration performance.

Data filtering

Use precise data filters to replicate only the data you need, reducing storage and compute costs.

Amazon Redshift optimization

Use appropriate Amazon Redshift node types and implement data compression to optimize costs.

Monitoring usage

Regularly review your zero-ETL integration usage and costs through AWS Cost Explorer.

Cleanup unused integrations

Delete integrations that are no longer needed to avoid ongoing charges.

Troubleshooting Zero-ETL integration

This section provides guidance for resolving common issues with zero-ETL integration.

Zero-ETL integration setup failures

Authentication failures

- Verify that the replication user exists and has the correct password in AWS Secrets Manager.
- Ensure that all required permissions have been granted to the replication user.
- Check that the secret ARN is correct and accessible by Oracle Database@AWS.
- Verify that the CMK resource policy allows access by Oracle Database@AWS service principal.

Network connectivity issues

- Make sure that your ODB network has the zero-ETL integration enabled.
- Verify that SSL is properly configured on port 2484 (Exadata only).

- Check that the Oracle database listener is running and accepting connections.
- Ensure that network security groups and NACLs allow traffic on port 2484.
- Verify that the service name in your secret matches the actual Oracle service name.

Permission errors

- Check that your IAM user or role has the necessary permissions for AWS Glue integration operations.
- Verify that the Amazon Redshift resource policy allows inbound integrations from your VM cluster.
- Ensure that Oracle Database@AWS has been granted access to your secrets and AWS Key Management Service key.

Replication issues

Initial load failures

- Verify that the Oracle database has sufficient resources to support the full load operation.
- Ensure that supplemental logging is enabled on the source database.
- Check for any table-level locks or constraints that might prevent data extraction.

Change data capture issues

- Verify that the Oracle database has adequate redo log space and retention.
- Check that the replication user has access to archived redo logs.
- For ASM-enabled systems, ensure that the ASM user is properly configured.
- Monitor Oracle database performance to ensure CDC is not causing resource contention.

High replication lag

- Monitor the replication lag metrics in CloudWatch.
- Check for high transaction volumes or large transactions in the source database.
- Verify that the Amazon Redshift cluster has adequate capacity to handle incoming data.

Data consistency issues

Missing or incomplete data

- Verify that the data filter includes all required schemas and tables.
- Check for unsupported data types that may be causing replication failures.

- Ensure that the replication user has SELECT permissions on all required tables.

Data type conversion errors

- Review the supported data type mappings between Oracle and Redshift.
- Check for Oracle-specific data types that may require custom handling.
- Consider modifying your Oracle schema to use more compatible data types.

Monitoring and debugging

Use the following approaches to monitor and debug Zero-ETL integration issues:

- **Integration status monitoring –** Regularly check the integration status using `aws glue describe-integrations`.
- **CloudWatch metrics –** Monitor available CloudWatch metrics for replication performance and errors.
- **Oracle database monitoring –** Monitor Oracle database performance and resource utilization.
- **Redshift monitoring –** Monitor Amazon Redshift cluster performance and storage utilization.

For complex issues that cannot be resolved using this troubleshooting guide, contact AWS Support with the following information:

- Integration ARN and current status.
- Error messages from integration describe operations.
- Oracle database and Amazon Redshift cluster configurations.
- Timeline of when the issue started occurring.

Security in Oracle Database@AWS

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS, OCI, and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors, including the sensitivity of your data, your organization's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Oracle Database@AWS. You also learn how to use other AWS services that help you to monitor and secure your Oracle Database@AWS resources.

You can manage access to your Oracle Database@AWS resources. The method you use to manage access depends on what type of task you need to perform with Oracle Database@AWS:

- Use AWS Identity and Access Management (IAM) policies to assign permissions that determine who is allowed to manage Oracle Database@AWS resources. For example, you can use IAM to determine who is allowed to create, describe, modify, and delete Exadata infrastucture, VM clusters or tag resources.
- Use the security features of your Oracle database engine to control who can log in to the databases on a DB instance. These features work just as if the database was on your local network.
- Use Secure Socket Layers (SSL) or Transport Layer Security (TLS) connections with Exadata databases. For more information, see Prepare for TLS Walletless Connections.
- Oracle Database@AWS isn't immediately accessible from the internet and deployed on private subnets in AWS only.

- Oracle Database@AWS uses many default Transmission Control Protocol (TCP) ports for various operations. For the full list of ports, see Default port assignments.
- To store and manage keys by using Transparent Data Encryption (TDE), which is enabled by default, Oracle Database@AWS uses OCI vaults or Oracle Key Vault. Oracle Database@AWS doesn't support AWS Key Management Service.
- By default, the database is configured by using Oracle-managed encryption keys. The database also supports customer-managed keys.
- To enhance data protection, use Oracle Data Safe with Oracle Database@AWS.

The following topics show you how to configure Oracle Database@AWS to meet your security and compliance objectives.

Topics

Data protection in Oracle Database@AWS

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.

- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-3.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with Oracle Database@AWS or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Data encryption

Exadata databases use Oracle Transparent Data Encryption (TDE) to encrypt your data. Your data is also protected in temporary tablespaces, undo segments, redo logs and during internal database operations such as JOIN and SORT. For more information, see Data Security.

Encryption in transit

Exadata databases use native Oracle Net Services encryption and integrity capabilities to secure connections to the database. For more information, see Security of data in transit.

Key management

Transparent Data Encryption includes a keystore to securely store master encryption keys, and a management framework to securely and efficiently manage the keystore and perform key maintenance operations. For more information, see To administer Vault encryption keys.

Identity and access management for Oracle Database@AWS

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Oracle Database@AWS resources. IAM is an AWS service that you can use with no additional charge.

Topics

Audience

How you use AWS Identity and Access Management (IAM) differs based on your role:

- **Service user** - request permissions from your administrator if you cannot access features (see Troubleshooting Oracle Database@AWS identity and access)
- **Service administrator** - determine user access and submit permission requests (see How Oracle Database@AWS works with IAM)
- **IAM administrator** - write policies to manage access (see Identity-based policies for Oracle Database@AWS)

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be authenticated as the AWS account root user, an IAM user, or by assuming an IAM role.

You can sign in as a federated identity using credentials from an identity source like AWS IAM Identity Center (IAM Identity Center), single sign-on authentication, or Google/Facebook credentials. For more information about signing in, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

For programmatic access, AWS provides an SDK and CLI to cryptographically sign requests. For more information, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity called the AWS account *root user* that has complete access to all AWS services and resources. We strongly recommend that you don't use the root user for everyday tasks. For tasks that require root user credentials, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users to use federation with an identity provider to access AWS services using temporary credentials.

A *federated identity* is a user from your enterprise directory, web identity provider, or Directory Service that accesses AWS services using credentials from an identity source. Federated identities assume roles that provide temporary credentials.

For centralized access management, we recommend AWS IAM Identity Center. For more information, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity with specific permissions for a single person or application. We recommend using temporary credentials instead of IAM users with long-term credentials. For more information, see Require human users to use federation with an identity provider to access AWS using temporary credentials in the *IAM User Guide*.

An *IAM group* specifies a collection of IAM users and makes permissions easier to manage for large sets of users. For more information, see Use cases for IAM users in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity with specific permissions that provides temporary credentials. You can assume a role by switching from a user to an IAM role (console) or by calling an AWS CLI or AWS API operation. For more information, see Methods to assume a role in the *IAM User Guide*.

IAM roles are useful for federated user access, temporary IAM user permissions, cross-account access, cross-service access, and applications running on Amazon EC2. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy defines permissions when associated with an identity or resource. AWS evaluates these policies when a principal makes a request. Most policies are stored in AWS as JSON documents. For more information about JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Using policies, administrators specify who has access to what by defining which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. An IAM administrator creates IAM policies and adds them to roles, which users can then assume. IAM policies define permissions regardless of the method used to perform the operation.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you attach to an identity (user, group, or role). These policies control what actions identities can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

Identity-based policies can be *inline policies* (embedded directly into a single identity) or *managed policies* (standalone policies attached to multiple identities). To learn how to choose between managed and inline policies, see Choose between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples include IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. You must specify a principal in a resource-based policy.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Other policy types

AWS supports additional policy types that can set the maximum permissions granted by more common policy types:

- **Permissions boundaries** – Set the maximum permissions that an identity-based policy can grant to an IAM entity. For more information, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – Specify the maximum permissions for an organization or organizational unit in AWS Organizations. For more information, see Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – Set the maximum available permissions for resources in your accounts. For more information, see Resource control policies (RCPs) in the *AWS Organizations User Guide*.
- **Session policies** – Advanced policies passed as a parameter when creating a temporary session for a role or federated user. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How Oracle Database@AWS works with IAM

Before you use IAM to manage access to Oracle Database@AWS, learn what IAM features are available to use with Oracle Database@AWS.

IAM feature	Oracle Database@AWS support
Identity-based policies	Yes
Resource-based policies	No
Policy actions	Yes
Policy resources	Yes
Policy condition keys	Yes
ACLs	No
ABAC (tags in policies)	Partial

IAM feature	Oracle Database@AWS support
Temporary credentials	Yes
Principal permissions	Yes
Service roles	No
Service-linked roles	Yes

To get a high-level view of how Oracle Database@AWS and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for Oracle Database@AWS

Supports identity-based policies: Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for Oracle Database@AWS

To view examples of Oracle Database@AWS identity-based policies, see Identity-based policies for Oracle Database@AWS.

Resource-based policies within Oracle Database@AWS

Supports resource-based policies: No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Policy actions for Oracle Database@AWS

Supports policy actions: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Include actions in a policy to grant permissions to perform the associated operation.

To see a list of Oracle Database@AWS actions, see Actions Defined by Oracle Database@AWS in the *Service Authorization Reference*.

Policy actions in Oracle Database@AWS use the following prefix before the action:

```
odb
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
      "odb:action1",
      "odb:action2"
         ]
```

To view examples of Oracle Database@AWS identity-based policies, see Identity-based policies for Oracle Database@AWS.

Policy resources for Oracle Database@AWS

Supports policy resources: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. As a best practice, specify a resource using its Amazon Resource Name (ARN). For actions that don't support resource-level permissions, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

To see a list of Oracle Database@AWS resource types and their ARNs, see Resources Defined by Oracle Database@AWS in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions Defined by Oracle Database@AWS.

To view examples of Oracle Database@AWS identity-based policies, see Identity-based policies for Oracle Database@AWS.

Policy condition keys for Oracle Database@AWS

Supports service-specific policy condition keys: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element specifies when statements execute based on defined criteria. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

To see a list of Oracle Database@AWS condition keys, see Condition Keys for Oracle Database@AWS in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions Defined by Oracle Database@AWS.

To view examples of Oracle Database@AWS identity-based policies, see Identity-based policies for Oracle Database@AWS.

ACLs in Oracle Database@AWS

Supports ACLs: No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

ABAC with Oracle Database@AWS

Supports ABAC (tags in policies): Partial

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes called tags. You can attach tags to IAM entities and AWS resources, then design ABAC policies to allow operations when the principal's tag matches the tag on the resource.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see Define permissions with ABAC authorization in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

Using temporary credentials with Oracle Database@AWS

Supports temporary credentials: Yes

Temporary credentials provide short-term access to AWS resources and are automatically created when you use federation or switch roles. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM and AWS services that work with IAM in the *IAM User Guide*.

Cross-service principal permissions for Oracle Database@AWS

Supports forward access sessions (FAS): Yes

Forward access sessions (FAS) use the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. For policy details when making FAS requests, see Forward access sessions.

Service roles for Oracle Database@AWS

Supports service roles: No

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Create a role to delegate permissions to an AWS service in the *IAM User Guide*.

Warning

Changing the permissions for a service role might break Oracle Database@AWS functionality. Edit service roles only when Oracle Database@AWS provides guidance to do so.

Service-linked roles for Oracle Database@AWS

Supports service-linked roles: Yes

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing Oracle Database@AWS service-linked roles, see Using service-linked roles for Oracle Database@AWS.

Identity-based policies for Oracle Database@AWS

By default, users and roles don't have permission to create or modify Oracle Database@AWS resources. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Create IAM policies (console) in the *IAM User Guide*.

For details about actions and resource types defined by Oracle Database@AWS, including the format of the ARNs for each of the resource types, see Actions, Resources, and Condition Keys for Oracle Database@AWS in the *Service Authorization Reference*.

Topics

Policy best practices

Identity-based policies determine whether someone can create, access, or delete Oracle Database@AWS resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see Validate policies with IAM Access Analyzer in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API

operations are called, add MFA conditions to your policies. For more information, see [Secure API access with MFA] in the *IAM User Guide*.

For more information about best practices in IAM, see [Security best practices in IAM] in the *IAM User Guide*.

Using the Oracle Database@AWS console

To access the Oracle Database@AWS console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Oracle Database@AWS resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

Allow users to provision Oracle Database@AWS resources

This policy allows users full access to provision Oracle Database@AWS resources. To set up DNS resolution from your VPC, create an outbound Route 53 resolver and add rules to forward DNS traffic with the OCI domain name to OCI DNS listener IP.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowODBAndEC2Actions",
            "Effect": "Allow",
            "Action": [
                "odb:GetOciOnboardingStatus",
                "odb:CreateOdbNetwork",
                "odb:DeleteOdbNetwork",
                "odb:GetOdbNetwork",
                "odb:ListOdbNetworks",
                "odb:UpdateOdbNetwork",
                "odb:CreateOdbPeeringConnection",
                "odb:DeleteOdbPeeringConnection",
                "odb:GetOdbPeeringConnection",
                "odb:ListOdbPeeringConnections",
                "odb:PutResourcePolicy",
                "odb:GetResourcePolicy",
                "odb:DeleteResourcePolicy",
                "ec2:DescribeVpcs",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeVpcEndpointAssociations",
                "ec2:CreateVpcEndpoint",
                "ec2:DeleteVpcEndpoints",
                "ec2:DescribeVpcEndpoints"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowSLRActions",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": [
                        "odb.amazonaws.com",
                        "vpc-lattice.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "AllowTaggingActions",
            "Effect": "Allow",
            "Action": [
                "odb:TagResource",
                "odb:UntagResource",
                "odb:ListTagsForResource"
            ],
            "Resource": "arn:aws:odb:*:*:odb-network/*"
        },
        {
            "Sid": "AllowOdbVpcLatticeActions",
            "Effect": "Allow",
            "Action": [
                "vpc-lattice:CreateServiceNetwork",
                "vpc-lattice:DeleteServiceNetwork",
                "vpc-lattice:GetServiceNetwork",
                "vpc-lattice:CreateServiceNetworkResourceAssociation",
                "vpc-lattice:DeleteServiceNetworkResourceAssociation",
                "vpc-lattice:GetServiceNetworkResourceAssociation",
                "vpc-lattice:CreateResourceGateway",
                "vpc-lattice:DeleteResourceGateway",
                "vpc-lattice:GetResourceGateway",
                "vpc-lattice:CreateServiceNetworkVpcEndpointAssociation"
            ],
            "Resource": "*"
        }
    ]
}
```

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS managed policies for Oracle Database@AWS

To add permissions to permission sets and roles, it's easier to use AWS managed policies than to write policies yourself. It takes time and expertise to create IAM customer managed policies that provide your team with only the permissions they need. To get started quickly, you can use our AWS managed policies. These policies cover common use cases and are available in your AWS account. For more information about AWS managed policies, see AWS managed policies in the *IAM User Guide*.

AWS services maintain and update AWS managed policies. You can't change the permissions in AWS managed policies. Services occasionally add additional permissions to an AWS managed policy to support new features. This type of update affects all identities (permission sets and roles) where the policy is attached. Services are most likely to update an AWS managed policy when a new feature is launched or when new operations become available. Services don't remove permissions from an AWS managed policy, so policy updates don't break your existing permissions.

Additionally, AWS supports managed policies for job functions that span multiple services. For example, the `ReadOnlyAccess` AWS managed policy provides read-only access to all AWS services and resources. When a service launches a new feature, AWS adds read-only permissions for new operations and resources. For a list and descriptions of job function policies, see AWS managed policies for job functions in the *IAM User Guide*.

Topics

- AWS managed policy: AmazonODBServiceRolePolicy

AWS managed policy: AmazonODBServiceRolePolicy

You can't attach the `AmazonODBServiceRolePolicy` policy to your IAM entities. This policy is attached to a service-linked role that allows Oracle Database@AWS to perform actions on your behalf. For more information, see Using service-linked roles for Oracle Database@AWS.

To view more details about the policy, including the latest version of the JSON policy document, see AmazonODBServiceRolePolicy in the *AWS Managed Policy Reference Guide*.

Oracle Database@AWS authentication and authorization in OCI

When you use AWS APIs to create resources for Oracle Database@AWS, those resources logically reside in your linked Oracle Cloud Infrastructure (OCI) tenancy. To deploy these resources, AWS communicates with OCI APIs on your behalf. To mitigate the confused deputy problem, OCI and Oracle Database@AWS use AWS STS as a trusted entity and forward access sessions to authorize your intent to use OCI APIs in your linked tenancy. Consequently, events are recorded for the `sts:getCallerIdentity` API from OCI IP space in your AWS CloudTrail trails and events history. Expect these events when you use Oracle Database@AWS APIs.

Troubleshooting Oracle Database@AWS identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Oracle Database@AWS and IAM.

Topics

I am not authorized to perform an action in Oracle Database@AWS

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but doesn't have the fictional `odb:`*`GetWidget`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 odb:GetWidget on resource: my-example-widget
```

In this case, the policy for the `mateojackson` user must be updated to allow access to the *`my-example-widget`* resource by using the `odb:`*`GetWidget`* action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to Oracle Database@AWS.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in Oracle Database@AWS. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my Oracle Database@AWS resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Oracle Database@AWS supports these features, see How Oracle Database@AWS works with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.

Compliance validation for Oracle Database@AWS

Your compliance responsibility when using Oracle Database@AWS is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. Oracle's documentation about compliance in the cloud is available on the Oracle website

Resilience in Oracle Database@AWS

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

In addition to the AWS global infrastructure, Oracle Database@AWS offers several features to help support your data resiliency and backup needs.

Using service-linked roles for Oracle Database@AWS

Oracle Database@AWS uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to Oracle Database@AWS. Service-linked roles are predefined by Oracle Database@AWS and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes using Oracle Database@AWS easier because you don't have to manually add the necessary permissions. Oracle Database@AWS defines the permissions of its service-linked roles, and unless defined otherwise, only Oracle Database@AWS can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete the roles only after first deleting their related resources. This protects your Oracle Database@AWS resources because you can't inadvertently remove permission to access the resources.

Service-linked role permissions for Oracle Database@AWS

Oracle Database@AWS uses the service-linked role named AWSServiceRoleForODB to allow Oracle Database@AWS to call AWS services on behalf of your resources.

The AWSServiceRoleForODB service-linked role trusts the following services to assume the role:

- `odb.amazonaws.com`
- `vpc-lattice.amazonaws.com`

This service-linked role has a permissions policy attached to it called `AmazonODBServiceRolePolicy` that grants it permissions to operate in your account. For more information, see AWS managed policy: AmazonODBServiceRolePolicy.

Note

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. If you encounter the following error message: **Unable to create the resource. Verify that you have permission to create service-linked role. Otherwise wait and try again later.**
Make sure you have the following permissions enabled:

```
{
    "Action": "iam:CreateServiceLinkedRole",
    "Effect": "Allow",
    "Resource": "arn:aws:iam::*:role/aws-service-role/odb.amazonaws.com/
AWSServiceRoleForODB",
    "Condition": {
        "StringLike": {
            "iam:AWSServiceName":"odb.amazonaws.com",
            "iam:AWSServiceName":"vpc-lattice.amazonaws.com"
        }
    }
}
```

For more information, see Service-linked role permissions in the *IAM User Guide*.

Creating a service-linked role for Oracle Database@AWS

You don't need to manually create a service-linked role. When you create an Exadata database, Oracle Database@AWS creates the service-linked role for you.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you create an Exadata database, Oracle Database@AWS creates the service-linked role for you again.

Editing a service-linked role for Oracle Database@AWS

Oracle Database@AWS does not allow you to edit the AWSServiceRoleForODB service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM For more information, see Editing a service-linked role in the *IAM User Guide*.

Deleting a service-linked role for Oracle Database@AWS

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained. However, you must delete all of your resources before you can delete the service-linked role.

Cleaning up a service-linked role for Oracle Database@AWS

Before you can use IAM to delete a service-linked role, you must first confirm that the role has no active sessions and remove any resources used by the role.

To check whether the service-linked role has an active session in the IAM console

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

2. In the navigation pane of the IAM console, choose **Roles**. Then choose the name (not the check box) of the AWSServiceRoleForODB role.
3. On the **Summary** page for the chosen role, choose the **Access Advisor** tab.
4. On the **Access Advisor** tab, review recent activity for the service-linked role.

Note

If you're unsure whether Oracle Database@AWS is using the AWSServiceRoleForODB role, you can try to delete the role. If the service is using the role, then the deletion fails and you can view the AWS Regions where the role is being used. If the role is being used, then you must wait for the session to end before you can delete the role. You cannot revoke the session for a service-linked role.

If you want to remove the AWSServiceRoleForODB role, you must first delete all of your Oracle Database@AWS resources.

Supported Regions for Oracle Database@AWS service-linked roles

Oracle Database@AWS supports using service-linked roles in all of the AWS Regions where the service is available. For more information, see AWS Regions and Endpoints.

Oracle Database@AWS updates to AWS managed policies

View details about updates to AWS managed policies for Oracle Database@AWS since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the Oracle Database@AWS Document history page.

Change	Description	Date
Service-linked role permissions for Oracle Database@AWS – Update to existing policy	Oracle Database@AWS added new permissions to the `AmazonODBServiceRolePolicy` of the `AWSServiceRoleForODB` service-linked role. These permissions allow Oracle Database@AWS to do the following:	June 30, 2025

Change	Description	Date
	• Describe Amazon VPC Transit Gateways attachments • Describe Amazon EC2 attachments • Activate an Amazon EventBridge source For more information, see Service-linked role permissions for Oracle Database@AWS.	
Service-linked role permissions for Oracle Database@AWS – Update to existing policy	Oracle Database@AWS added new permissions to the `AmazonODBServiceRolePolicy` of the `AWSServiceRoleForODB` service-linked role. These permissions allow Oracle Database@AWS to do the following: • Describe an Amazon EventBridge source • Describe and create an event bus For more information, see Service-linked role permissions for Oracle Database@AWS.	June 26, 2025
AWS managed policy: AmazonODBServiceRolePolicy – New service-linked role policy	Oracle Database@AWS added the `AmazonODBServiceRolePolicy` for the `AWSServiceRoleForODB` service-linked role. For more information, see AWS managed policy: AmazonODBServiceRolePolicy.	December 2, 2024
Oracle Database@AWS started tracking changes	Oracle Database@AWS started tracking changes for its AWS managed policies.	December 2, 2024

Monitoring Oracle Database@AWS

Monitoring is an important part of maintaining the reliability, availability, and performance of Oracle Database@AWS and your other AWS solutions. AWS provides the following monitoring tools to watch Oracle Database@AWS, report when something is wrong, and take automatic actions when appropriate:

- *Amazon CloudWatch* monitors your AWS resources and and the applications you run on AWS in real time. You can collect and track metrics, create customized dashboards, and set alarms that notify you or take actions when a specified metric reaches a threshold that you specify. For example, you can have CloudWatch track CPU usage or other metrics of your Amazon EC2 instances and automatically launch new instances when needed. For more information, see the Amazon CloudWatch User Guide.
- *Amazon CloudWatch Logs* enables you to monitor, store, and access your log files from Amazon EC2 instances, CloudTrail, and other sources. CloudWatch Logs can monitor information in the log files and notify you when certain thresholds are met. You can also archive your log data in highly durable storage. For more information, see the Amazon CloudWatch Logs User Guide.
- *Amazon EventBridge* can be used to automate your AWS services and respond automatically to system events, such as application availability issues or resource changes. Events from AWS services are delivered to EventBridge in near real time. You can write simple rules to indicate which events are of interest to you and which automated actions to take when an event matches a rule. For more information, see Amazon EventBridge User Guide.
- *AWS CloudTrail* captures API calls and related events made by or on behalf of your AWS account and delivers the log files to an Amazon S3 bucket that you specify. You can identify which users and accounts called AWS, the source IP address from which the calls were made, and when the calls occurred. For more information, see the AWS CloudTrail User Guide.

Monitoring Oracle Database@AWS with Amazon CloudWatch

You can monitor Oracle Database@AWS using CloudWatch, which collects raw data and processes it into readable, near real-time metrics. These statistics are kept for 15 months, so that you can access historical information and gain a better perspective on how your web application or service is performing. You can also set alarms that watch for certain thresholds, and send notifications or take actions when those thresholds are met. For more information, see the Amazon CloudWatch User Guide.

Amazon CloudWatch metrics for Oracle Database@AWS

The Oracle Database@AWS service reports metrics to Amazon CloudWatch in the `AWS/ODB` namespace for VM clusters, container databases, and pluggable databases.

Topics

- Metrics for cloud VM clusters
- Metrics for container databases
- Metrics for pluggable databases

Metrics for cloud VM clusters

The Oracle Database@AWS service reports the following metrics in the `AWS/ODB` namespace for cloud VM clusters.

Metric	Description	Units
`ASMDiskgroupUtilization`	The percentage of usable space used in a Disk Group. Usable space is the space available for growth. DATA disk group stores our Oracle database files. RECO disk group contains database files for recovery such as archives and flashback logs.	Percentage
`CpuUtilization`	The percent CPU utilization.	Percentage
`FilesystemUtilization`	The percent utilization of provisioned filesystem.	Percentage
`LoadAverage`	The system load average over 5 minutes.	Integer
`MemoryUtilization`	The percentage of memory available for starting new applications, without	Percentage

Metric	Description	Units
	swapping. The available memory can be obtained via the following command: `cat /proc/meminfo`	
`NodeStatus`	Indicates whether the host is reachable.	Integer
`OcpusAllocated`	The number of OCPUs allocated.	Integer
`SwapUtilization`	The percent utilization of total swap space.	Percentage

Metrics for container databases

The Oracle Database@AWS service reports the following metrics in the `AWS/ODB` namespace for container databases.

Metric	Description	Units
`BlockChanges`	The Average number of blocks changed per second.	Changes per second
`CpuUtilization`	The CPU utilization expressed as a percentage, aggregated across all consumer groups. The utilization percentage is reported with respect to the number of CPUs the database is allowed to use, which is two times the number of OCPUs.	Percentage
`CurrentLogons`	The number of successful logons during the selected interval.	Count

Metric	Description	Units
`ExecuteCount`	The number of user and recursive calls that executed SQL statements during the selected interval.	Count
`ParseCount`	The number of hard and soft parses during the selected interval.	Count
`StorageAllocated`	Total amount of storage space allocated to the database at the collection time.	GB
`StorageAllocatedByTablespace`	Total amount of storage space allocated to the tablespace at the collectio n time. In case of container database, this metric provides root container tablespaces.	GB
`StorageUsed`	Total amount of storage space used by the database at the collection time.	GB
`StorageUsedByTablespace`	Total amount of storage space used by tablespace at the collection time. In case of container database, this metric provides root container tablespaces.	GB

Metric	Description	Units
`StorageUtilization`	The percentage of provisioned storage capacity currently in use. Represents the total allocated space for all tablespaces.	Percentage
`StorageUtilizationByTablespace`	This indicates the percentage of storage space utilized by the tablespace at the collection time. In case of container database, this metric provides root container tablespaces..	Percentage
`TransactionCount`	The combined number of user commits and user rollbacks during the selected interval.	Count
`UserCalls`	The combined number of logons, parses, and execute calls during the selected interval.	Count

Metrics for pluggable databases

The Oracle Database@AWS service reports the following metrics in the `AWS/ODB` namespace for pluggable databases.

Metric	Description	Units
`AllocatedStorageUtilizationByTablespace`	The percentage of space used by tablespace, out of all allocated. For container databases, this metric provides data for root container tablespaces.	Percent

Metric	Description	Units
	(Statistic: Mean, Interval: 30 minutes)	
`AvgGCCRBlockReceiveTime`	The average global cache CR (consistent-read) block receive time. For RAC / cluster databases only. (Statistic: Mean, Interval: 5 minutes)	Milliseconds
`AvgGCCurrentBlockReceiveTime`	The average global cache current blocks receive time. Statistic reports the mean value. For Real Application Cluster (RAC) databases only. (Statistic: Mean, Interval: 5 minutes)	Milliseconds
`BlockChanges`	The average number of blocks changed per second. (Statistic: Mean, Interval: 1 minute)	changes per second
`BlockingSessions`	Current blocking sessions. Not applicable for container databases. (Statistic: Max, Interval: 15 minutes)	Count
`CPUTimeSeconds`	The average rate of accumulation of CPU time by foreground sessions in the database instance over the time interval. The CPU time component of Average Active Sessions. (Statistic: Mean, Interval: 1 minute)	Seconds per second

Metric	Description	Units
`CpuCount`	The number of CPUs during the selected interval.	Count
`CpuUtilization`	The CPU utilization expressed as a percentage, aggregated across all consumer groups. The utilization percentage is reported with respect to the number of CPUs the database is allowed to use, which is two times the number of OCPUs. (Statistic: Mean, Interval: 1 minute)	Percent
`CurrentLogons`	The number of successful logons during the selected interval. (Statistics: Sum, Interval: 1 minute)	Count
`DBTimeSeconds`	The average rate of accumulation of database time (CPU + Wait) by foreground sessions in the database instance over the time interval. Also known as Average Active Sessions. (Statistic: Mean, Interval: 1 minute)	Seconds per second

Metric	Description	Units
`DbmgmtJobExecutionsCount`	The number of SQL job executions on a single managed database or a database group, and their status. Status dimensions can be the following values: "Succeeded," "Failed," "InProgress." (Statistic: Sum, Interval: 1 minute)	Count
`ExecuteCount`	The number of user and recursive calls that executed SQL statements during the selected interval. (Statistic: Sum, Interval: 1 minute)	Count
`FRASpaceLimit`	The flash recovery area space limit. Not applicable for pluggable databases. (Statistic: Max, Interval: 15 minutes)	GB
`FRAUtilization`	The flash recovery area utilization. Not applicable for pluggable databases. (Statistic: Mean, Interval: 15 minutes)	Percent
`GCCRBlocksReceived`	The global cache CR (consistent-read) blocks received per second. For RAC / cluster databases only. (Statistic: Mean, Interval: 5 minutes)	Blocks per second

Metric	Description	Units
`GCCurrentBlocksReceived`	Represents global cache current blocks received per second. Statistic reports the mean value. For Real Application Cluster (RAC) databases only. (Statistic: Mean, Interval: 5 minutes)	Blocks per second
`IOPS`	The average number of input-output operations per second. (Statistic: Mean, Interval: 1 minute)	Operations per second
`IOThroughputMB`	The average throughput in MB per second. (Statistic: Mean, Interval: 1 minute)	MB per second
`InterconnectTrafficMB`	The average internode data transfer rate. For RAC / cluster databases only. (Statistic: Mean, Interval: 5 minutes)	MB per second
`InvalidObjects`	Invalid database objects count. Not applicable for container databases. (Statistic: Max, Interval: 24 hours)	Count
`LogicalBlocksRead`	The average number of blocks read from SGA/Memory (buffer cache) per second. (Statistic: Mean, Interval: 1 minute)	Reads per second

Metric	Description	Units
`MaxTablespaceSize`	The maximum possible tablespace size. For container databases, this metric provides data for root container tablespaces. (Statistic: Max, Interval: 30 minutes)	GB
`MemoryUsage`	Memory pool total size in MB. (Statistic: Mean, Interval: 15 minutes)	MB
`MonitoringStatus`	The monitoring status of the resource. If a metric collectio n fails, error information is captured in this metric. (Statistic: Mean, Interval: 5 minutes)	Not applicable
`NonReclaimableFRA`	The Non-reclaimable fast recovery area. Not applicabl e for pluggable databases. (Statistic: Mean, Interval: 15 minutes)	Percent
`OcpusAllocated`	The actual number of OCPUs allocated by the service during the selected interval of time. (Statistic: Count, Interval: 1 minute)	Integer
`ParseCount`	The number of hard and soft parses during the selected interval. (Statistic: Sum, Interval: 1 minute)	Count

Metric	Description	Units
`ParsesByType`	The number of hard or soft parses per second. (Statistic: Mean, Interval: 1 minute)	Parses per second
`ProblematicScheduledDBMSJobs`	The problematic scheduled database jobs count. Not applicable for container databases. (Statistic: Max, Interval: 15 minutes)	Count
`ProcessLimitUtilization`	The process limit utilization. Not applicable for pluggable databases. (Statistic: Mean, Interval: 1 minute)	Percent
`Processes`	The database processes count. Not applicable for pluggable databases. (Statistic: Max, Interval: 1 minute)	Count
`ReclaimableFRA`	The reclaimable fast recovery area. Not applicable for pluggable databases. (Statistic: Mean, Interval: 15 minutes)	Percent
`ReclaimableFRASpace`	The flash recovery area reclaimable space. Not applicable for pluggable databases. (Statistic: Mean, Interval: 15 minutes)	GB
`RedoSizeMB`	The average amount of redo generated, in MB per second. (Statistic: Mean, Interval: 1 minute)	MB per second

Metric	Description	Units
`SessionLimitUtilization`	The session limit utilization. Not applicable for pluggable databases. (Statistic: Mean, Interval: 1 minute)	Percent
`Sessions`	The number of sessions in the database. (Statistic: Mean, Interval: 1 minute)	Count
`StorageAllocated`	The maximum amount of space allocated by tablespace during the interval. For container databases, this metric provides data for root container tablespaces. (Statistic: Max, Interval: 30 minutes)	GB
`StorageAllocatedByTablespace`	The maximum amount of space allocated by tablespace during the interval. For container databases, this metric provides data for root container tablespaces. (Statistic: Max, Interval: 30 minutes)	GB
`StorageUsed`	The maximum amount of space used during the interval. (Statistic: Max, Interval: 30 minutes)	GB

Metric	Description	Units
`StorageUsedByTablespace`	The maximum amount of space used by tablespace during the interval. For container databases, this metric provides data for root container tablespaces. (Statistic: Max, Interval: 30 minutes)	GB
`StorageUtilization`	The percentage of provisioned storage capacity currently in use. Represents the total allocated space for all tablespaces. (Statistic: Mean, Interval: 30 minutes)	Percent
`StorageUtilizationByTablespace`	The percentage of the space utilized, by tablespace. For container databases, this metric provides data for root container tablespaces. (Statistic: Mean, Interval: 30 minutes)	Percent
`TransactionCount`	The combined number of user commits and user rollbacks during the selected interval. (Statistic: Sum, Interval: 1 minute)	Count
`TransactionsByStatus`	The number of committed or rolled back transactions per second. (Statistic: Mean, Interval: 1 minute)	Transactions per second

Metric	Description	Units
`UnusableIndexes`	Unusable indexes count in database schema. Not applicable for container databases. (Statistic: Max, Interval: 24 hours)	Count
`UsableFRA`	The useable fast recovery area. Not applicable for pluggable databases. (Statistic: Mean, Interval: 15 minutes)	Percent
`UsedFRASpace`	The flash recovery area space usage. Not applicable for pluggable databases. (Statistic: Max, Interval: 15 minutes)	GB
`UserCalls`	The combined number of logons, parses, and execute calls during the selected interval. (Statistic: Sum, Interval: 1 minute)	Count
`WaitTimeSeconds`	The average rate of accumulation of non-idle wait time by foreground sessions in the database instance over the time interval. The wait time component of Average Active Sessions. (Statistic: Mean, Interval: 5 minutes)	Seconds per second

Amazon CloudWatch dimensions for Oracle Database@AWS

You can filter Oracle Database@AWS metrics data by using any dimension in the following table.

Dimension	Filters the requested data for . . .
`cloudVmClusterId`	The identifier of a VM cluster.
`cloudExadataInfrastructureId`	The identifier of the Exadata infrastructure.
`collectionName`	A name of a collection.
`deploymentType`	The type of infrastructure.
`diskgroupName`	A name of a disk group
`errorCode`	An error code.
`errorSeverity`	The severity of an error.
`filesystemName`	The name of a file system.
`hostName`	The name of the host machine.
`instanceName`	The name of a database instance.
`instanceNumber`	The instance number of a database instance.
`ioType`	A type of I/O operation.
`jobId`	A unique identifier for a job.
`managedDatabaseGroupId`	The identifier of a Managed Database Group.
`managedDatabaseId`	The identifier of a Managed Database.
`memoryPool`	A type of memory pool.
`memoryType`	A type of memory.
`ociCloudVmClusterId`	The OCI identifier of a VM cluster.

Dimension	Filters the requested data for . . .
ociCloudExadataInfrastructureId	The OCI identifier of the Exadata infrastructure.
parseType	A type of parse.
resourceId	The identifier of a resource.
resourceId_Database	The identifier of a database.
resourceId_DbNode	The identifier of a database node.
resourceName	The name of a resource.
resourceName_Database	The name of a database.
resourceName_DbNode	The name of a database node.
resourceType	A type of database.
schemaName	The name of a schema.
status	The status of a database.
tablespaceContents	The contents of a tablespace.
tablespaceName	The name of a tablespace.
tablespaceType	A type of tablespace.
transactionStatus	The status of a transaction.
waitClass	A class of wait event.

Monitoring Oracle Database@AWS events in Amazon EventBridge

You can monitor Oracle Database@AWS events in EventBridge, which delivers a stream of real-time data from applications and AWS services. EventBridge routes this data to targets such as AWS Lambda and Amazon Simple Notification Service.

Note

EventBridge was formerly called Amazon CloudWatch Events. For more information, see EventBridge is the evolution of Amazon CloudWatch Events in the *Amazon EventBridge User Guide*.

Overview of Oracle Database@AWS events

Oracle Database@AWS events are structured messages that indicate changes in resource lifecycles. An *event bus* is a router that receives events and delivers them to zero or more destinations, or *targets*. Oracle Database@AWS events can be generated from the following sources:

Events from AWS

These events are generated from Oracle Database@AWS APIs on the AWS side and are delivered to the default event bus in your AWS account.

Events from OCI

These events are generated directly from OCI, such as events related to Oracle Exadata infrastructure or VM clusters. When you subscribe to Oracle Database@AWS, an event bus with prefix `aws.partner/odb/` is created in your AWS account to receive events from OCI.

Oracle Database@AWS events from AWS

Oracle Database@AWS events from AWS include lifecycle changes related to the ODB network during creation and deletion. These events are delivered to the default event bus in your AWS account. The delivery type is *best effort*.

ODB network events

Event	Event ID	Message
Creation	ODB-EVENT-0001	Successfully created ODB network *odbnet_ID*
Creation failed	ODB-EVENT-0011	Failed to create ODB network *odbnet_ID*
Deletion	ODB-EVENT-0002	Successfully deleted ODB network *odbnet_ID*
Deletion failed	ODB-EVENT-0012	Failed to delete ODB network *odbnet_ID*

Example: ODB network creation event

The following example shows an event for a successful ODB network creation.

```
{
  "version": "0",
  "id": "01234567-EXAMPLE",
  "detail-type": "ODB Network Event",
  "source": "aws.odb",
  "account": "123456789012",
  "time": "2025-06-12T10:23:43Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:odb:us-east-1:123456789012:odbnetwork/odbnet-1234567890abcdef"
  ],
  "detail": {
    "eventId": "ODB-EVENT-0001",
    "message": "Successfully created ODB network odbnet-1234567890abcdef"
  }
}
```

Oracle Database@AWS events from OCI

Most events are generated directly from OCI. Oracle Database@AWS creates an event bus with prefix `aws.partner/odb/` in your AWS account to receive events from OCI. We recommend that you do not delete this event bus.

OCI provides comprehensive event types, including the following:

- Oracle Exadata infrastructure
- VM cluster events
- CDB events
- PDB events

For more information about the specific event types and details that OCI supports, see Oracle Exadata Database Service on Dedicated Infrastructure Events and Events for Autonomous Database on Dedicated Exadata Infrastructure.

Filtering Oracle Database@AWS events

You can follow EventBridge suggested best practices on event bus setup at Event buses in Amazon EventBridge. Depending on your use cases, you can set up EventBridge rules to filter events and targets to receive and use events.

Filtering ODB network events from AWS

For ODB network events from AWS, you can filter using the following event pattern:

```
{
  "source": ["aws.odb"],
  "detail-type": ["ODB Network Event"]
}
```

You can apply this pattern using the EventBridge `put-rule` API with the default event bus. For more information, see PutRule in the *Amazon EventBridge API Reference*.

Filtering Oracle Database@AWS events from OCI

For Oracle Database@AWS events from OCI, you can set up a rule using a command similar to the example in PutRule in the *Amazon EventBridge API Reference*. Note the following guidelines:

- Use a custom event pattern depending on the event types that you want to filter.
- Set **EventBusName** to the name of the bus that Oracle Database@AWS created.

For more information about how to filter events and set up EventBridge targets across accounts, see Sending and receiving events between AWS accounts in Amazon EventBridge.

Troubleshooting Oracle Database@AWS events

If you encounter an issue with event delivery or event content, do the following:

- For ODB network events, contact AWS Support.
- For Oracle Database@AWS events other than ODB network events, contact Oracle Cloud Support.

For more information, see Getting support for Oracle Database@AWS.

Logging Oracle Database@AWS API calls using AWS CloudTrail

Oracle Database@AWS is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service. CloudTrail captures all API calls for Oracle Database@AWS as events. The calls captured include calls from the Oracle Database@AWS console and code calls to the Oracle Database@AWS API operations. Using the information collected by CloudTrail, you can determine the request that was made to Oracle Database@AWS, the IP address from which the request was made, when it was made, and additional details.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root user or user credentials.
- Whether the request was made on behalf of an IAM Identity Center user.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

Note

Oracle Database@AWS records `GetCallerIdentity` API calls from AWS Security Token Service (STS) in your CloudTrail logs. These STS API calls verify the identity of Oracle Database@AWS when interacting with OCI on your behalf. They are a normal and secure part of AWS operations and do not expose sensitive information.

CloudTrail is active in your AWS account when you create the account and you automatically have access to the CloudTrail **Event history**. The CloudTrail **Event history** provides a viewable, searchable, downloadable, and immutable record of the past 90 days of recorded management events in an AWS Region. For more information, see Working with CloudTrail Event history in the *AWS CloudTrail User Guide*. There are no CloudTrail charges for viewing the **Event history**.

For an ongoing record of events in your AWS account past 90 days, create a trail or a CloudTrail Lake event data store.

CloudTrail trails

A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. All trails created using the AWS Management Console are multi-Region. You can create a single-Region or a multi-Region trail by using the AWS CLI. Creating a multi-Region trail is recommended because you capture activity in all AWS Regions in your account. If you create a single-Region trail, you can view only the events logged in the trail's AWS Region. For more information about trails, see Creating a trail for your AWS account and Creating a trail for an organization in the *AWS CloudTrail User Guide*.

You can deliver one copy of your ongoing management events to your Amazon S3 bucket at no charge from CloudTrail by creating a trail, however, there are Amazon S3 storage charges. For more information about CloudTrail pricing, see AWS CloudTrail Pricing. For information about Amazon S3 pricing, see Amazon S3 Pricing.

CloudTrail Lake event data stores

CloudTrail Lake lets you run SQL-based queries on your events. CloudTrail Lake converts existing events in row-based JSON format to Apache ORC format. ORC is a columnar storage format that is optimized for fast retrieval of data. Events are aggregated into *event data stores*, which are immutable collections of events based on criteria that you select by applying advanced event selectors. The selectors that you apply to an event data store control which events persist and are available for you to query. For more information about CloudTrail Lake, see Working with AWS CloudTrail Lake in the *AWS CloudTrail User Guide*.

CloudTrail Lake event data stores and queries incur costs. When you create an event data store, you choose the pricing option you want to use for the event data store. The pricing option determines the cost for ingesting and storing events, and the default and maximum retention period for the event data store. For more information about CloudTrail pricing, see AWS CloudTrail Pricing.

Oracle Database@AWS management events in CloudTrail

Management events provide information about management operations that are performed on resources in your AWS account. These are also known as control plane operations. By default, CloudTrail logs management events.

Oracle Database@AWS logs all Oracle Database@AWS control plane operations as management events.

Oracle Database@AWS event examples

An event represents a single request from any source and includes information about the requested API operation, the date and time of the operation, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so events don't appear in any specific order.

The following example shows a CloudTrail event that demonstrates the `CreateOdbNetwork` operation.

```
{
    "eventVersion": "1.09",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AKIAIOSFODNN7EXAMPLE:yourRole",
        "arn": "arn:aws:sts::123456789012:assumed-role/Admin/yourRole",
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AKIAIOSFODNN7EXAMPLE",
                "arn": "arn:aws:iam::123456789012:role/Admin",
                "accountId": "123456789012",
                "userName": "Admin"
            },
            "attributes": {
                "creationDate": "2024-11-06T21:17:29Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2024-11-06T21:17:44Z",
    "eventSource": "odb.amazonaws.com",
    "eventName": "CreateOdbNetwork",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "python-requests/2.28.2",
    "requestParameters": {
        "availabilityZoneId": "use1-az6",
        "backupSubnetCidr": "123.45.6.7/89",
        "clientSubnetCidr": "123.44.6.7/89",
        "clientToken": "testClientToken",
        "defaultDnsPrefix": "testLabel",
        "displayName": "yourOdbNetwork"
    },
    "responseElements": {
        "displayName": "yourOdbNetwork",
        "odbNetworkId": "odbnet_1234567",
        "status": "PROVISIONING"
    },
    "requestID": "daf2e3f5-96a3-4df7-a026-863f96db793e",
    "eventID": "797163d3-5726-441d-80a7-6eeb7464acd4",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
        "clientProvidedHostHeader": "odb.us-east-1.amazonaws.com"
    }
}
```

For information about CloudTrail record contents, see CloudTrail record contents in the *AWS CloudTrail User Guide*.

Troubleshooting Oracle Database@AWS

Use the following sections to help troubleshoot networking issues you may encounter with Oracle Database@AWS.

Topics

Creation of ODB network fails

When you can't create a ODB network, the following are common causes:

Restricted CIDR Ranges

The ODB network uses specific CIDR ranges for the client and backup subnets. Ensure that the CIDR ranges you've chosen for these subnets do not overlap with any restricted or reserved IP address ranges.

The following CIDR ranges are reserved and cannot be used for the ODB network:

- Oracle cloud reserved range: 169.254.0.0/16
- Reserved Class D: 224.0.0.0 - 239.255.255.255
- Reserved Class E: 240.0.0.0 - 255.255.255.255
- Future OCI use: 100.105.0.0/16

Follow the EC2 rules for CIDR ranges as outlined in the VPC documentation. To learn more, see CIDR block association restrictions.

Additionally, avoid overlap between specified CIDR ranges and those used for VPC connectivity to the ODB network.

Overlapping VPC CIDR

The CIDR range you've specified for the ODB network should not overlap with the CIDR ranges used by any of your existing VPCs. Overlapping CIDR ranges can cause routing conflicts and prevent the successful creation of the ODB network. Check the CIDR ranges of ODB peering VPCs and ensure the ODB network CIDR is unique and non-overlapping.

Ownership of VPCs

The ODB network and the VPC you're connecting to must be owned by the same AWS account. If you're trying to peer the ODB network to a VPC owned by a different account, the creation will fail. Verify that the ODB network and VPC are both owned by the same AWS account.

Lack of a transit gateway

If you add a CIDR range to the ODB network peered CIDR list without attaching a transit gateway to the VPC, the create or update operation fails. There is no requirement about the CIDR ranges that the attachment is used for.

Connectivity issues between your VPC and ODB network or VM clusters

When you can't connect from your VPC to the ODB network or the VM clusters within it, the following are common causes:

- **Verifying VPC configuration –** In the Oracle Database@AWS console, locate the VPC that is peered with the ODB network. Confirm the VPC ID matches the one shown in the ODB network details.
- **Inspecting route tables –** In the Amazon VPC console, find the route table attached to the subnet where your application is running. Check for a route with a destination CIDR that matches the client subnet CIDR of the ODB network. Confirm that this route points to the correct ODB network ARN. If the route is missing, add a new one to the ODB network's client subnet CIDR.
- **Validating peered CIDRs –** Review the `Peered CIDRs` section in the ODB network details. Confirm all the relevant CIDR blocks from your VPC are listed. If a required CIDR is missing, update the peered CIDRs.
- **Checking security group rules –** In the Amazon EC2 console, locate the security groups for resources in your VPC. Review the inbound and outbound rules, updating them as needed to permit the necessary traffic.
- **Confirming Availability Zones –** In the Amazon VPC console, identify the Availability Zone (AZ) of your subnet. Verify that the ODB network is also deployed in the same AZ as your subnet.

- **Avoiding multiple ODB network peering connections** – Check your VPC peering connections in the Oracle Database@AWS Console. Make sure you have only one active connection to an ODB network. If you see more than one ODB network peering, remove the extra ones.

Unresolvable hostnames or scannames of VM clusters from VPC

If the hostnames or scannames of the VM clusters are not resolvable from your VPC, configure DNS forwarding on the VPC and the following resources to resolve DNS records hosted on the ODB network:

- An outbound endpoint to send DNS queries to the ODB network. For more information, see Configuring an outbound endpoint in an ODB network in Oracle Database@AWS.
- A resolver rule to specify the domain name of the DNS queries that the resolver forwards to the DNS for ODB network. For more information, see Configuring a resolver rule in Oracle Database@AWS.

Getting support for Oracle Database@AWS

Learn how to get information and support for Oracle Database@AWS.

Oracle support scope and contact information

Oracle Cloud Support is the first line of support for all Oracle Database@AWS questions. To contact support, sign in to the Oracle Cloud Infrastructure (OCI) Console, then select the life raft icon. If you don't have a My Oracle Cloud Support account, see My Oracle Cloud Support accounts and access.

Examples of issues that Oracle Support can help you with include the following:

- Database connection issues (Oracle TNS)
- Oracle Database performance issues
- Oracle Database error resolution
- Networking issues related to communications with the OCI tenancy associated with the service
- Quota (limits) increases to receive more capacity (for more information, see Requesting a Limit Increase for Database Resources)
- Scaling to add more compute and storage capacity to your Oracle Database infrastructure

- New generation hardware upgrades
- Billing issues related to your AWS Marketplace charges

If you need to contact Oracle Support outside of the OCI Console, tell your Oracle Support agent that your issue is related to Oracle Database@AWS. This is because requests for this service are handled by an OCI support team that specializes with these deployments.

Contacting Oracle support by phone

1. Call **1-800-223-1711**. If you are outside of the United States, visit Oracle Support Contacts Global Directory to find contact information for your country or region.
2. Choose option "2" to open a new Service Request (SR).
3. Choose option "4" for "unsure".
4. Let the agent know that you have an issue with your multicloud system, and the name of the product. An internal Service Request will be opened on your behalf and an OCI support engineer will contact you directly.

You can also submit a question to the Multicloud forum in Oracle's Cloud Customer Connect community. This option is available to all customers.

My Oracle Cloud Support accounts and access

To create My Oracle Cloud Support service request tickets, the administrator of your organization's Oracle Database@AWS service must approve your request. If you're the Oracle Database@AWS administrator, complete the My Oracle Cloud Support onboarding instructions included in the Oracle Database@AWS service activation email.

You can find instructions for onboarding with My Oracle Cloud Support in the following topics:

- Configuring Your Oracle Support Account
- Creating a Support Request

For instructions on approving users to open My Oracle Cloud Support support requests, see Administrator Tasks for Support.

AWS Support scope and contact information

AWS Support is your first line of support for all AWS-related issues and questions. Create an AWS Support case for your issue, as you do with other AWS services. The AWS Support team collaborates with OCI Support as needed.

Examples of Oracle Database@AWS issues that AWS Support can help you with include the following:

- Virtual networking issues including those involving network address translation (NAT), firewalls, DNS and traffic management, and AWS subnets
- Bastion and virtual machine (VM) issues including database host connection, software installation, latency, and host performance
- Exadata VM cluster metrics reporting within Amazon CloudWatch
- Billing issues related to AWS services

For information on AWS Support, see Getting started with AWS Support.

Oracle service level agreements

If you have questions about Oracle Database@AWS Service Level Agreements (SLAs), or want to request service credits for SLA breaches, contact your Oracle account manager. See Service Level Agreements for more information.

Quotas for Oracle Database@AWS

Oracle Database@AWS is a multicloud offering. AWS doesn't set or enforce quotas for Oracle Database@AWS resources. Quotas are enforced by Oracle Cloud Infrastructure (OCI). For more information about OCI quotas, see Quotas and Service Limits in the Oracle Cloud Infrastructure documentation.

Document history for the Oracle Database@AWS User Guide

The following table describes the documentation releases for Oracle Database@AWS.

Change	Description	Date
Oracle Database@AWS supports the Europe (Ireland) Region	You can create your Oracle Database@AWS resources in these regions. For more information, see Supported Regions for Oracle Database@AWS.	February 24, 2026
Oracle Database@AWS supports Multiple ODB Peerings	You can peer multiple VPCs to a single ODB network. Additionally, you can have one VPC peered to multiple ODB networks. For more information, see here.	February 19, 2026
Oracle Database@AWS supports the Asia Pacific (Sydney) Region and Canada (Central) Region	You can create your Oracle Database@AWS resources in these regions. For more information, see Supported Regions for Oracle Database@AWS.	February 2, 2026
Oracle Database@AWS supports the Asia Pacific (Tokyo) Region, US East (Ohio) Region, Europe (Frankfurt) Region	You can create your Oracle Database@AWS resources in these regions. For more information, see Supported Regions for Oracle Database@AWS.	December 22, 2025

Oracle Database@AWS supports entitlement sharing across AWS accounts	You can now share AWS Marketplace entitlements for Oracle Database@AWS across AWS accounts in the same AWS organization using AWS License Manager. For more information, see Entitlement sharing in Oracle Database@AWS.	December 19, 2025
Oracle Database@AWS supports modifying zero-ETL integration data filters	Oracle Database@AWS supports modifying data filters for existing zero-ETL integrations with Amazon Redshift. You can update data filter patterns to include or exclude specified schemas and tables from data replicati on. For more information, see Managing zero-ETL integrati ons.	October 15, 2025
Oracle Database@AWS supports peer network CIDR management for peering connections	You can specify peer network CIDRs when you create or update ODB peering connections. You control which subnets in the peer VPC have access to your ODB network. A VPC account can update the CIDR ranges without also owning the ODB network. For more informati on, see Configuring ODB peering to an Amazon VPC in Oracle Database@AWS.	October 10, 2025

Oracle Database@AWS supports zero-ETL integration with Amazon Redshift	Oracle Database@AWS now integrates with VPC Lattice to enable zero-ETL integrati on with Amazon Redshift. For more information, see Service integrations for Oracle Database@AWS.	July 2, 2025
Update to IAM service-linked role permissions	The `AmazonODBServiceRo lePolicy` policy now grants additional permissions to describe VPC transit gateway attachments, describe Amazon EC2 subnets, and activate an Amazon EventBridge source. For more information, see Oracle Database@AWS updates to AWS managed policies.	June 30, 2025
Update to IAM service-linked role permissions	The `AmazonODBServiceRo lePolicy` policy now grants additional permissions to describe events in Amazon EventBridge Scheduler and create or describe an event bus. For more information, see Oracle Database@AWS updates to AWS managed policies.	June 26, 2025

Oracle Database@AWS supports the US West (Oregon) Region	You can create your Oracle Database@AWS resources in the US West (Oregon) Region. The supported physical AZ IDs are `usw2-az3` and `usw2-az4`. For more information, see Supported Regions for Oracle Database@AWS.	June 26, 2025
Oracle Database@AWS supports resource sharing across AWS accounts	You can now share Exadata infrastructure and VM clusters with other AWS accounts within your organization using AWS Resource Access Manager (AWS RAM). You can provision infrastructure once and share it across multiple accounts, reducing costs while maintaining separation of responsibilities. For more information, see Resource sharing in Oracle Database@AWS.	June 26, 2025

Oracle Database@AWS supports events in Amazon EventBridge	Oracle Database@AWS delivers events to Amazon EventBridge for monitoring resource lifecycle changes. Events are generated from both AWS and OCI sources, allowing you to track changes to ODB network, Exadata infrastructure, VM clusters, and databases. For more information, see Monitoring Oracle Database@AWS events in Amazon EventBridge.	June 26, 2025
Oracle Database@AWS supports cross-Region subscription	Oracle Database@AWS supports cross-Region subscription, allowing you to subscribe once and use the service in all available AWS Regions. For more informati on, see Subscribe to Oracle Database@AWS in multiple Regions.	June 26, 2025
Oracle Database@AWS supports ODB peering connections as a separate resource	ODB peering connections are now a separate resource with dedicated APIs for creating, viewing, and deleting peering connections. You can create peering connections between an ODB network and a Amazon VPC in the same account or in different accounts. For more informati on, see Working with ODB peering Connections.	June 26, 2025

Oracle Database@AWS integrates the ODB network with Amazon S3	Oracle Database@AWS now integrates with VPC Lattice to enable Oracle managed backups to Amazon S3 and direct ODB network access to Amazon S3. For more information, see Service integrations for Oracle Database@AWS.	June 26, 2025
Oracle Database@AWS supports Autonomous VM clusters	You can now create Autonomous VM clusters on your Exadata infrastru cture. Autonomous VM clusters are fully managed databases that automate key management tasks using machine learning and AI. For more information, see Step 3: Create an Exadata VM cluster or Autonomous VM cluster in Oracle Database@AWS.	May 28, 2025
Oracle Database@AWS supports customizable maintenance windows	You can now configure maintenance windows for your Exadata infrastructure with options for Oracle-ma naged or Customer-managed schedules. You can also select patching modes (Rolling or Non-rolling) and specify maintenance timing preferenc es. For more information, see Create an Oracle Exadata infrastructure in Oracle Database@AWS.	May 1, 2025

Oracle Database@AWS supports a new Availability Zone (AZ)	You can now create an ODB network in an AZ with the physical ID `use1-az4` or `use1-az6`. For more information, see Oracle Exadata infrastructure.	March 26, 2025
Oracle Database@AWS supports Amazon VPC Transit Gateways	If you connect a transit gateway to a VPC that is peered to an ODB network, you can connect multiple VPCs to this gateway. Applications running in these VPCs can access an Exadata VM cluster running in your ODB network. For more information, see Configuring Amazon VPC Transit Gateways for Oracle Database@AWS.	March 26, 2025
Oracle Database@AWS supports database and storage server types for Exadata X11M	You can specify the database server type and storage server type when you create an infrastructure using Exadata X11M. For more informati on, see Create an Oracle Exadata infrastructure in Oracle Database@AWS.	February 4, 2025

New service-linked role policy	Oracle Database@AWS added a new policy `AmazonODBServiceRolePolicy` for the `AWSServiceRoleForODB` service-linked role. For more information, see Oracle Database@AWS updates to AWS managed policies.	December 2, 2024
Initial release	Initial release of the Oracle Database@AWS User Guide	December 2, 2024