TAMING

ORACLE DATABASE ADMINISTRATION ON AWS

'Best way to understand and Manage Oracle on AWS'

-**Gary Thomas**

Oracle Databases on AWS

Amazon Relational Database Service (Amazon RDS)

Oracle on Amazon RDS

Administering your Oracle DB instance

Performing common system tasks for Oracle DB instances

Performing common log-related tasks for Oracle DB instances

Performing common RMAN tasks for Oracle DB instances

Performing common scheduling tasks for Oracle DB instances

Performing common diagnostic tasks for Oracle DB instances

Performing miscellaneous tasks for Oracle DB instances

Importing data into Oracle on Amazon RDS

Working with Oracle replicas for Amazon RDS

Amazon S3 integration

Creating an Oracle DB instance

Connecting to your Oracle DB instance

Managing an Amazon RDS DB instance

Modifying an Amazon RDS DB instance

Maintaining a DB instance

Upgrading a DB instance engine version

Upgrading the Oracle DB engine

Upgrading an Oracle DB snapshot

Renaming a DB instance

Rebooting a DB instance

Working with read replicas

Working with storage for Amazon RDS DB instances

Deleting a DB instance

Relational Database Services (RDS) for Oracle – *Step-by-Step*

Installation of Oracle on EC2 – *Step-by-Step*

Oracle Application Express (APEX)

Oracle Enterprise Manager Database Express

Oracle Management Agent for Enterprise Manager Cloud Control

Setting up Amazon RDS to host tools and third-party software for Oracle

Using Oracle GoldenGate with Amazon RDS

Using the Oracle Repository Creation Utility on Amazon RDS for Oracle

Oracle database engine release notes

Security in Amazon RDS

Data protection in Amazon RDS

Resilience in Amazon RDS

Infrastructure security in Amazon RDS

Amazon RDS API and interface VPC endpoints (AWS PrivateLink)

Security best practices for Amazon RDS

Master user account privileges

Amazon Virtual Private Cloud VPCs and Amazon RDS

Troubleshooting for Amazon RDS

Best Practices for Running Oracle Database on AWS

Oracle Databases on AWS

This article presents an overview of options for deploying Oracle databases on AWS.

Some common terms used in this editorial.

- Infrastructure as a Service (IaaS): Virtual hardware supplied by the cloud vendor. This can include Virtual Machines (VMs), Storage, DNS, load balancers, VLANs and VPN amongst other things. This is like having your machines in someone else's data center. Your day-to-day job will be unchanged.
- Platform as a Service (PaaS): Typically a whole environment packaged as a service. For example, operating system and database, or operating system and application server. Some vendors now supply whole development environments, including development tools, in this manner. This

should come with tooling to make your day-to-day job easier. It will probably also limit your access to the underlying infrastructure.

- Software as a Service (SaaS): Applications delivered over the cloud. You have no involvement in the installation, upgrades or management of the service. You are just a user.
- Database as a Service (DBaaS): The database is hosted and managed by the cloud provider. Typically this will include cloud-based tooling to reduce the work done by system administrators and DBAs. DBaaS is a PaaS offering.
- Bring Your Own License (BYOL): You buy the Oracle licenses directly from Oracle, so the cloud provider is not responsible for charging or validating your licenses.

- On-Demand License: The cost of the cloud service includes the Oracle licensing. This is useful for short-term PoCs, but ultimately ends up being more expensive than BYOL.
- Proof of Concept (PoC): Short term tests to prove something is possible. Systems will typically be scrapped at the end of the PoC.

A Word about Pricing

I've avoided mentioning pricing in this article for the following reasons.

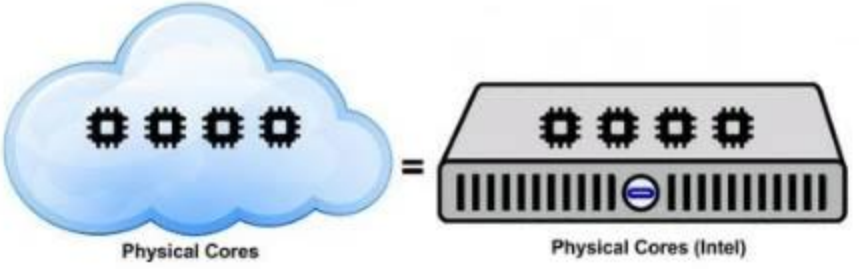
- Some of the DBaaS offerings include the On-Demand Licensing costs, which make them seem incredibly expensive at first sight. Some allow a switch to BYOL, some do not.
- Oracle list prices are not indicative of the actual costs.

- Cloud pricing varies depending on a number of variables, including performance, disk space, memory, network traffic, length of contract. The cost and viability of a cloud database would have to be judged on a case-by-case basis.
- Many cloud services are priced by the hour, but reserving a service for an extended period of time (1-3 years) will give a heavily discounted price.
- High Availability costs money. Failover to another host in the data centre is normal. Failover to a different data centre is an extra cost. The level of availability has to be factored into the costs.

General Oracle Licensing for Cloud Providers

For certified cloud providers (Oracle, AWS, Microsoft Azure) Oracle products are licensed based on a physical core on the cloud. The relationship used to

be a physical core in the cloud was the equivalent of a physical Intel core.



How the physical cores are presented in the cloud depends on the cloud provider.

- AWS : Each vCPU is associated with a single thread presented from Intel hyper-threading, so 2 vCPUs equal 1 physical core.

The licensing multiplier for Intel cores is "0.5". Prior to the update on the 23-Jan-2017 the cloud licensing policy could be described as follows.

Type	vCPUs	Physical Cores	Licenses
AWS	16	8	8 * 0.5 = 4
Physical	N/A	8	8 * 0.5 = 4

The cloud licensing policy document changed on 23-Jan-2017 to say that, "When counting Oracle Processor license requirements in Authorized Cloud Environments, the Oracle Processor Core Factor Table is not applicable." This suggests the following licensing approach.

Type	vCPUs	Physical Cores	Licenses
AWS	16	8	8
Physical	N/A	8	8 * 0.5 = 4

All features/options are licensed in the normal way, based on the above formula. Licensing for Standard Edition is based on sockets. The relationship between physical cores and sockets is described in the document linked below. This licensing policy is described in the following document.

For the purposes of licensing Oracle programs in an Authorized Cloud Environment, customers are required to count as follows:

- Amazon EC2 and RDS - count two vCPUs as equivalent to one Oracle Processor license if hyper-threading is enabled, and one vCPU as equivalent to one Oracle Processor license if hyper-threading is not enabled.

Oracle Processor Core Factor Table is not applicable. When licensing Oracle programs with Standard Edition One, Standard Edition 2, or Standard Edition in the product name, the pricing is based on the size of the instance. Authorized Cloud Environment instances with four or fewer Amazon vCPUs, or four or fewer Azure vCPUs, are counted as 1 socket, which is considered equivalent to an Oracle processor license. For Authorized Cloud Environment instances with more than four Amazon vCPUs, or more than four Azure vCPUs,

every four Amazon vCPUs used (rounded up to the nearest multiple of four), and every four Azure vCPUs used (rounded up to the nearest multiple of four) equate to a licensing requirement of one socket.

Under this cloud computing policy, Oracle Database Standard Edition may only be licensed on Authorized Cloud Environment instances up to 16 Amazon vCPUs or 16 Azure vCPUs. Oracle Standard Edition One and Standard Edition 2 may only be licensed on Authorized Cloud Environment instances up to eight Amazon vCPUs or eight Azure vCPUs. If licensing Database Standard Edition 2 by Named User Plus metric, the minimums are 10 NUP licenses per 8 Amazon vCPUs or 8 Azure vCPUs.

Example, for Database Enterprise Edition licensing in an Authorized Cloud Environment: Licensing

Oracle Database Enterprise Edition on a single instance of four Amazon vCPUs, where hyper-threading is enabled, would require two processor licenses. (Two Amazon vCPUs are considered equivalent to an Oracle Processor license).

Standard Named User Plus licensing applies, including counting the minimums where applicable. For Oracle Linux purposes, two VMs whose combined size are no more than 64 vCPUs are counted as a single Oracle Linux Basic Limited or Premier Limited system in Authorized Cloud Environments. A single VM consisting of more than 64 vCPUs is counted as an Oracle Linux Basic or Premier system.

You should always confirm your licensing requirements with Oracle Licensing Management Services (LMS) or an official licensing partner before embarking on any work with Oracle products in the cloud. It is important you have a contract specify-

ing what you will be using and what you will pay to use it!

Amazon AWS (EC2 and RDS)

Amazon's cloud platform is called Amazon Web Services (AWS), which amongst many other services allows Oracle databases to be run in two distinct ways.

- Elastic Compute Cloud (EC2) : A conventional virtual machine that is self-managed. You pay for the VM (compute power, storage and network traffic) like any AWS VM. Oracle licensing is your responsibility, BYOL.
- Relational Data Services (RDS) for Oracle : A DBaaS offering, with no OS system administration and tooling to greatly reduce DBA work. This can be licensed using BYOL or On-Demand Licensing. Some functionality, including file system access, is restricted

or prevented entirely, as is typical of many DBaaS offerings. There are also RDS offerings for MySQL, MariaDB, Amazon Aurora, PostgreSQL and MS SQL Server.

Both methods allow SQL*Net access to the database, so existing applications can connect without modification to the technology stack.

Using X emulation from most cloud providers is incredibly slow and painful, but it's easy to avoid it.

Get used to doing silent installations! Build a response file on a client using the Oracle Universal Installer (OUI).

```
$ ./runInstaller -record -destinationFile /
tmp/11gR2.rsp
```

Use the response file on the cloud server to perform a silent installation.

```
$ ./runInstaller -silent -ignoreSysPrereqs -re-
sponseFile /tmp/11gR2.rsp
```

In a similar way, you should get used to using the Database Configuration Assistant (DBCA) in silent mode. You can either specify all parameters on the command line, or use a response file.

```
$ dbca -silent -createDatabase \
-templateName General_Purpose.dbc \
-gdbname cdb1 -sid cdb1 -responseFile NO_
VALUE \
-characterSet AL32UTF8 \
-sysPassword OraPasswd1 \
-systemPassword OraPasswd1 \
-createAsContainerDatabase true \
-numberOfPDBs 1 \
-pdbName pdb1 \
-pdbAdminPassword OraPasswd1 \
-databaseType MULTIPURPOSE \
```

```
-memoryMgmtType auto_sga \
-storageType FS \
-ignorePreReqs
```

Connections to cloud services are over public networks (the internet), so you need to make sure all connections are encrypted. This is easy to do with Oracle using Native Network Encryption or TCP/IP with SSL/TLS.

Amazon Relational Database

Service (Amazon RDS)

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the AWS Cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks.

Overview of Amazon RDS

Why do you want a managed relational database service? Because Amazon RDS takes over many of the difficult and tedious management tasks of a relational database:

- When you buy a server, you get CPU, memory, storage, and IOPS, all bundled together. With Amazon RDS, these are split apart so that you can scale them independently. If you need more CPU, less IOPS, or more storage, you can easily allocate them.
- Amazon RDS manages backups, software patching, automatic failure detection, and recovery.
- To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances. It also restricts access to certain system procedures and tables that require advanced privileges.

- You can have automated backups performed when you need them, or manually create your own backup snapshot. You can use these backups to restore a database. The Amazon RDS restore process works reliably and efficiently.
- You can use the database products you are already familiar with: MySQL, MariaDB, PostgreSQL, Oracle, Microsoft SQL Server.
- You can get high availability with a primary instance and a synchronous secondary instance that you can fail over to when problems occur. You can also use MariaDB, Microsoft SQL Server, MySQL, Oracle, and PostgreSQL read replicas to increase read scaling.
- In addition to the security in your database package, you can help control who can access your RDS databases by using AWS Identity and Access Management (IAM) to define

users and permissions. You can also help protect your databases by putting them in a virtual private cloud.

DB instances

The basic building block of Amazon RDS is the DB instance. A *DB instance* is an isolated database environment in the AWS Cloud. Your DB instance can contain multiple user-created databases. You can access your DB instance by using the same tools and applications that you use with a standalone database instance. You can create and modify a DB instance by using the AWS Command Line Interface, the Amazon RDS API, or the AWS Management Console.

Each DB instance runs a *DB engine*. Amazon RDS currently supports the MySQL, MariaDB, PostgreSQL, Oracle, and Microsoft SQL Server DB engines. Each DB engine has its own supported features, and each version of a DB engine may include

specific features. Additionally, each DB engine has a set of parameters in a DB parameter group that control the behavior of the databases that it manages.

The computation and memory capacity of a DB instance is determined by its *DB instance class*. You can select the DB instance that best meets your needs. If your needs change over time, you can change DB instances.

DB instance storage comes in three types: Magnetic, General Purpose (SSD), and Provisioned IOPS (PIOPS). They differ in performance characteristics and price, allowing you to tailor your storage performance and cost to the needs of your database. Each DB instance has minimum and maximum storage requirements depending on the storage type and the database engine it supports. It's important to have sufficient storage so that your databases have room to grow. Also, sufficient stor-

age makes sure that features for the DB engine have room to write content or log entries.

You can run a DB instance on a virtual private cloud (VPC) using the Amazon Virtual Private Cloud (Amazon VPC) service. When you use a VPC, you have control over your virtual networking environment. You can choose your own IP address range, create subnets, and configure routing and access control lists. The basic functionality of Amazon RDS is the same whether it's running in a VPC or not. Amazon RDS manages backups, software patching, automatic failure detection, and recovery. There's no additional cost to run your DB instance in a VPC.

Amazon RDS uses Network Time Protocol (NTP) to synchronize the time on DB Instances.

AWS Regions and Availability Zones

Amazon cloud computing resources are housed in highly available data center facilities in different areas of the world (for example, North America, Europe, or Asia). Each data center location is called an AWS Region.

Each AWS Region contains multiple distinct locations called Availability Zones, or AZs. Each Availability Zone is engineered to be isolated from failures in other Availability Zones. Each is engineered to provide inexpensive, low-latency network connectivity to other Availability Zones in the same AWS Region. By launching instances in separate Availability Zones, you can protect your applications from the failure of a single location.

You can run your DB instance in several Availability Zones, an option called a Multi-AZ deployment. When you choose this option, Amazon automatically provisions and maintains a secondary standby DB instance in a different Availability

Zone. Your primary DB instance is synchronously replicated across Availability Zones to the secondary instance. This approach helps provide data redundancy and failover support, eliminate I/O freezes, and minimize latency spikes during system backups.

Security

A *security group* controls the access to a DB instance. It does so by allowing access to IP address ranges or Amazon EC2 instances that you specify.

Monitoring an Amazon RDS DB instance

There are several ways that you can track the performance and health of a DB instance. You can use the Amazon CloudWatch service to monitor the performance and health of a DB instance. CloudWatch performance charts are shown in the Amazon RDS console. You can also subscribe to Amazon RDS events to be notified about changes to a DB

instance, DB snapshot, DB parameter group, or DB security group.

How to work with Amazon RDS

There are several ways that you can interact with Amazon RDS.

AWS Management Console

The AWS Management Console is a simple web-based user interface. You can manage your DB instances from the console with no programming required.

Command line interface

You can use the AWS Command Line Interface (AWS CLI) to access the Amazon RDS API interactively.

Programming with Amazon RDS

If you are a developer, you can access the Amazon RDS programmatically.

For application development, we recommend that you use one of the AWS Software Development Kits (SDKs). The AWS SDKs handle low-level details such as authentication, retry logic, and error handling, so that you can focus on your application logic. AWS SDKs are available for a wide variety of languages.

AWS also provides libraries, sample code, tutorials, and other resources to help you get started more easily.

Oracle on Amazon RDS

Amazon RDS supports DB instances that run the following versions and editions of Oracle Database:

- Oracle 19c, Version 19.0.0.0
- Oracle 18c, Version 18.0.0.0
- Oracle 12c, Version 12.2.0.1
- Oracle 12c, Version 12.1.0.2

You can create DB instances and DB snapshots, point-in-time restores, and automated or manual backups. You can use DB instances running Oracle inside a VPC. You can also add features to your Oracle DB instance by enabling various options. Amazon RDS supports Multi-AZ deployments for Oracle as a high-availability, failover solution.

To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances. It also restricts access to certain system procedures

and tables that require advanced privileges. You can access databases on a DB instance using any standard SQL client application such as Oracle SQL*Plus. However, you can't access the host directly by using Telnet or Secure Shell (SSH).

When you create a DB instance using your master account, the account gets DBA privileges, with some limitations. Use this account for administrative tasks such as creating additional database accounts. You can't use SYS, SYSTEM, or other Oracle-supplied administrative accounts.

Oracle versions on Amazon RDS

Amazon RDS for Oracle supports the following major database releases.

Oracle 19c with Amazon RDS

Amazon RDS supports Oracle version 19c, which includes Oracle Enterprise Edition and Oracle Standard Edition Two.

Oracle 19c version 19.0.0.0 includes many new features and updates from the previous version. In this section, you can find the features and changes important to using Oracle 19c version 19.0.0.0 on Amazon RDS. For a complete list of the changes, see the Oracle database 19c documentation. For a complete list of features supported by each Oracle 19c edition, see Permitted features, options, and management packs by Oracle database offering in the Oracle documentation.

Amazon RDS parameter changes for Oracle 19c version 19.0.0.0

Oracle 19c version 19.0.0.0 includes several new parameters and parameters with new ranges and new default values.

The following table shows the new Amazon RDS parameters for Oracle 19c version 19.0.0.0.

Name	Values	Modifiable	Description
lob_signature_enable	TRUE, FALSE (default)	Y	Enables or disables the LOB locator signature feature.
max_datapump_parallel_per_job	1 to 1024, or AUTO	Y	Specifies the maximum number of parallel processes allowed for each Oracle Data Pump job.

The compatible parameter has a new maximum value for Oracle 19c version 19.0.0.0 on Amazon RDS. The following table shows the new default value.

Parameter name	Oracle 19c version 19.0.0.0 maximum value	Oracle 18c version 18.0.0.0 maximum value
compatible	19.0.0	18.0.0

The following parameters were removed in Oracle 19c Version 19.0.0.0:

- exafusion_enabled
- max_connections
- o7_dictionary_access

Oracle 18c on Amazon RDS

Oracle Corporation intends to deprecate support for Oracle Database version 18.0.0.0 on July 1, 2021. On this date, Amazon RDS plans to do the following:

- Deprecate support for Oracle version 18c for both BYOL and LI
- Begin upgrading all 18c DB instances automatically

The following schedule includes upgrade recommendations.

Action or recommendation	Oracle Database version 18c
We recommend that you upgrade Oracle 18c DB instances manually to Oracle 19c and validate your applications.	Now–June 30, 2021
We recommend that you upgrade Oracle 18c snapshots manually to Oracle 19c.	May 1, 2021
You can no longer create new Oracle 18c instances with Amazon RDS. You can continue to restore 18c DB snapshots without being automatically upgraded until June 30, 2021.	May 1, 2021
Amazon RDS for Oracle plans to	July 1,

start automatic upgrades of your Oracle 18c instances to version 19c.	2021
Amazon RDS for Oracle plans to start automatic upgrades to version 19c for any Oracle 18c DB instances restored from snapshots.	July 1, 2021

Oracle 12c with Amazon RDS

Amazon RDS supports Oracle version 12c, which includes Oracle Enterprise Edition and Oracle Standard Edition Two. Oracle version 12c includes two major versions:

Oracle 12c version 12.2.0.1 with Amazon RDS

Oracle 12c version 12.2.0.1 includes many new features and updates from the previous version. In this section, you can find the features and changes important to using Oracle 12c version 12.2.0.1 on Amazon RDS.

Amazon RDS parameter changes for Oracle 12c version 12.2.0.1

Oracle 12c version 12.2.0.1 includes 20 new parameters in addition to several parameters with new ranges and new default values.

The following table shows the new Amazon RDS parameters for Oracle 12c version 12.2.0.1.

Name	Values	Modifiable	Description
allow_global_dblinks	TRUE, FALSE (default)	Y	Specifies whether LDAP lookup for database links is allowed for the database.
approx_for_aggregation	TRUE, FALSE (default)	Y	Replaces exact query processing for aggregation queries with approximate query processing.

approx_for_count_distinct	TRUE, FALSE (default)	Y	Automatically replaces COUNT (DISTINCT *expr*) queries with APPROX_COUNT_DISTINCT queries.
approx_for_percentile	NONE (default), PERCENTILE_CONT, PERCENTILE_CONT DETERMINISTIC, PERCENTILE_DISC, PERCENTILE_DISC DETERMINISTIC, ALL, ALL DETERMINISTIC	Y	Converts exact percentile functions to their approximate percentile function counterparts.
cursor_invalidation	DEFERRED, IMMEDIATE (default)	Y	Controls whether deferred cursor invalidation or immediate cursor invalidation is used for DDL statements by default.
data_guard_sync_latency	0 (default) to the	Y	Controls

	number of seconds specified by the NET_TIMEOUT attribute for the LOG_ARCHIVE_DEST_n parameter		how many seconds the Log Writer (LGWR) process waits beyond the response of the first in a series of Oracle Data Guard SYNC redo transport mode connections.
data_transfer_cache_size	0 – 512M, rounded up to the next granule size	Y	Sets the size of the data transfer cache (in bytes) used to receive data blocks (typically from a primary database in an Oracle Data Guard environment) for consumption by an instance when an

			RMAN RECOVER ... NONLOGGED BLOCK command is running.
inmemory_adg_enabled	TRUE (default), FALSE	Y	Indicates whether in-memory for Active Data Guard is enabled in addition to the in-memory cache size.
inmemory_expressions_usage	STATIC_ONLY, DYNAMIC_ONLY,ENABLE (default), DISABLE	Y	Controls which In-Memory Expressions (IM expressions) are populated into the In-Memory Column Store (IM column store) and are available for queries.
inmemory_virtual_columns	ENABLE, MANUAL (default), DISABLE	Y	Controls which In-Memory Expressions

			(IM expressions) are populated into the In-Memory Column Store (IM column store) and are available for queries.
instance_abort_delay_time	0 (default) and higher	Y	Specifies how much time to delay an internally initiated instance shutdown (in seconds), such as when a fatal process dies or an unrecoverable instance error occurs.
instance_mode	READ-WRITE (default), READ-ONLY, READ-MOSTLY	N	Indicates whether the instance is read-write, read-only, or read-mostly.

long_module_action	TRUE (default), FALSE	Y	Enables the use of longer lengths for modules and actions.
max_idle_time	0 (default) to the maximum integer. The value of 0 indicates that there is no limit.	Y	Specifies the maximum number of minutes that a session can be idle. After that point, the session is automatically terminated.
optimizer_adaptive_plans	TRUE (default), FALSE	Y	Controls adaptive plans. Adaptive plans are execution plans built with alternative choices that are decided at run time based on statistics collected as the query executes.
optimizer_adaptive_statistics	TRUE, FALSE	Y	Controls

	(default)		adaptive statistics. Some query shapes are too complex to rely on base table statistics alone, so the optimizer augments these statistics with adaptive statistics.
outbound_dblink_protocols	ALL (default), NONE, TCP, TCPS, IPC	N	Specifies the network protocols allowed for communicating for outbound database links in the database.
resource_manage_goldengate	TRUE, FALSE (default)	Y	Determines whether Oracle GoldenGate apply processes in the database are resource managed.
standby_db_preserve_states	NONE (default),	N	Controls

	SESSION, ALL		whether user sessions and other internal states of the instance are retained when a readable physical standby database is converted to a primary database.
uniform_log_timestamp_format	TRUE (default), FALSE	Y	Specifies that a uniform timestamp format be used in Oracle Database trace (.trc) files and log files (such as the alert log).

The compatible parameter has a new default value for Oracle 12c version 12.2.0.1 on Amazon RDS. The following table shows the new default value.

Parameter name	Oracle 12c version 12.2.0.1 default value	Oracle 12c version 12.1.0.2 default value
compatible	12.2.0	12.0.0

The optimizer_features_enable parameter has a new value range for Oracle 12c version 12.2.0.1 on Amazon RDS. For the old and new value ranges, see the following table.

Parameter name	12c version 12.2.0.1 range	12c version 12.1.0.2 range
optimizer_features_enable	8.0.0 to 12.2.0.1	8.0.0 to 12.1.0.2

The following parameters were removed in Oracle 12c Version 12.2.0.1:

- global_context_pool_size
- max_enabled_roles
- optimizer_adaptive_features
- parallel_automatic_tuning

- parallel_degree_level
- use_indirect_data_buffers

The following parameter is not supported in Oracle 12c Version 12.2.0.1 and later:

- sec_case_sensitive_logon

Amazon RDS security changes for Oracle 12c version 12.2.0.1

In Oracle 12c version 12.2.0.1, direct grant of the privilege ADMINISTER DATABASE TRIGGER is required for the owners of database-level triggers. During a major version upgrade to Oracle 12c version 12.2.0.1, Amazon RDS grants this privilege to any user that owns a trigger so that the trigger owner has the required privileges.

Oracle 12c version 12.1.0.2 with Amazon RDS

Oracle 12c version 12.1.0.2 brings over 500 new features and updates from the previous version. In

this section, you can find the features and changes important to using Oracle 12c version 12.1.0.2 on Amazon RDS.

Oracle 12c version 12.1.0.2 includes 16 new parameters that impact your Amazon RDS DB instance, and also 18 new system privileges, several no longer supported packages, and several new option group settings. For more information on these changes, see the following sections.

Amazon RDS parameter changes for Oracle 12c version 12.1.0.2

Oracle 12c version 12.1.0.2 includes 16 new parameters in addition to several parameters with new ranges and new default values.

The following table shows the new Amazon RDS parameters for Oracle 12c version 12.1.0.2.

Name	Values	Modifiable	Description
connection_brokers	CONNECTION_BROKERS = broker_description[,...]	N	Specifies connection broker types, the number of connection brokers of each type, and the maximum number of connections per broker.
db_index_compression_inheritance	TABLESPACE, TABL, ALL, NONE	Y	Displays the options that are set for table or tablespace level compression inheritance.

db_big_table_cache_percent_target	0-90	Y	Specifies the cache section target size for automatic big table caching, as a percentage of the buffer cache.
heat_map	ON, OFF	Y	Enables the database to track read and write access of all segments and modification of database blocks that are due to data manipulation language (DML) and data definition language (DDL) statements.
inmemory_clause_default	INMEMORY, NO INMEMORY	Y	INMEMORY_CLAUSE_DEFAULT enables you to specify a default In-Memory Column Store (IM column store) clause for new tables and materialized views.
inmemory_clause_default_memcompress	NO MEMCOMPRESS, MEMCOMPRESS FOR DML, MEMCOMPRESS FOR QUERY, MEMCOMPRESS FOR QUERY LOW, MEMCOMPRESS FOR QUERY HIGH, MEMCOMPRESS FOR CAPACITY, MEMCOMPRESS FOR CAPACITY LOW, MEMCOMPRESS FOR CAPACITY HIGH	Y	See INMEMORY_CLAUSE_DEFAULT.
inmemory_clause_default_priority	PRIORITY LOW, PRIORITY MEDIUM, PRIORITY HIGH, PRIORITY CRITICAL, PRIORITY NONE	Y	See INMEMORY_CLAUSE_DEFAULT.
inmemory_force	DEFAULT, OFF	Y	INMEMORY_FORCE allows you to specify whether tables and materialized view that are specified as INMEMORY are populated into the In-Memory Column Store (IM column store) or not.
inmemory_max_populate_servers	Null	N	INMEMORY_MAX_POPULATE_SERVERS specifies the maximum number of background populate servers to use for In-Memory Column Store (IM column store) population, so that these servers don't overload the rest of the system.
inmemory_query	ENABLE (default), DISABLE	Y	INMEMORY_QUERY is used to enable or disable in-memory queries for the entire database at the session or system level.
inmemory_size	0, 104857600-274877906944	Y	INMEMORY_SIZE sets the size of the In-Memory Column Store (IM column store) on a database instance.
inmemory_trickle_repopulate_servers_percent	0 to 50	Y	INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENT limits the maximum number of background populate servers used for In-Memory Column Store (IM column store) repopulation. This limit is applied because trickle repopulation is designed to use only a small percentage of the populate servers.
max_string_size	STANDARD (default), EXTENDED	N	Controls the maximum size of VARCHAR2, NVARCHAR2, and RAW.
optimizer_adaptive_features	TRUE (default), FALSE	Y	Enables or disables all of the adaptive optimizer features.
optimizer_adaptive_reporting_only	TRUE, FALSE (default)	Y	Controls reporting-only mode for adaptive optimizations.
pdb_file_name_convert	There is no default value.	N	Maps names of existing files to new file names.
pga_aggregate_limit	0-max of memory	Y	Specifies a limit on the aggregate PGA memory consumed by the instance.
processor_group_name	There is no default value.	N	Instructs the database instance to run itself within the specified operating system processor group.
spatial_vector_acceleration	TRUE, FALSE	N	Enables or disables the spatial vector acceleration, part of spatial option.
temp_undo_enabled	TRUE, FALSE (default)	Y	Determines whether transactions within a particular session can have a temporary undo log.
threaded_execution	TRUE, FALSE	N	Enables the multithreaded Oracle model, but prevents OS authentication.
unified_audit_sga_queue_size	1 MB - 30 MB	Y	Specifies the size of the system global area (SGA) queue for unified auditing.
use_dedicated_broker	TRUE, FALSE	N	Determines how dedicated servers are spawned.

Several parameters have new value ranges for Oracle 12c version 12.1.0.2 on Amazon RDS. For the old and new value ranges, see the following table.

Parameter name	12c version 12.1.0.2 range
audit_trail	os | db [, extended] | xml [, extended]
compatible	If you upgrade to 12.2.0.1, 18c, or 19c, COMPATIBLE must be 11.2.0 or higher. We recommend that you use the default settings for COMPATIBLE for your version of Oracle Database unless you have a reason to change it. If COMPATIBLE is not explicitly set, Amazon RDS automatically sets this parameter to 12.0.0.
db_securefile	PERMITTED | PREFERRED | ALWAYS | IGNORE | FORCE
db_writer_processes	1-100
optimizer_features_enable	8.0.0 to 12.1.0.2
parallel_degree_policy	MANUAL,LIMITED,AUTO,ADAPTIVE
parallel_min_server	0 to parallel_max_servers

One parameter has a new default value for Oracle 12c on Amazon RDS. The following table shows the new default value.

Parameter name	Oracle 12c default value
job_queue_processes	50

Amazon RDS system privileges for Oracle 12c version 12.1.0.2

Several new system privileges have been granted to the system account for Oracle 12c version 12.1.0.2. These new system privileges include the following:

- ALTER ANY CUBE BUILD PROCESS
- ALTER ANY MEASURE FOLDER
- ALTER ANY SQL TRANSLATION PROFILE
- CREATE ANY SQL TRANSLATION PROFILE
- CREATE SQL TRANSLATION PROFILE

- DROP ANY SQL TRANSLATION PROFILE
- EM EXPRESS CONNECT
- EXEMPT DDL REDACTION POLICY
- EXEMPT DML REDACTION POLICY
- EXEMPT REDACTION POLICY
- LOGMINING
- REDEFINE ANY TABLE
- SELECT ANY CUBE BUILD PROCESS
- SELECT ANY MEASURE FOLDER
- USE ANY SQL TRANSLATION PROFILE

Amazon RDS options for Oracle 12c version 12.1.0.2

Several Oracle options changed between Oracle 11g and Oracle 12c version 12.1.0.2, though most of the options remain the same between the two versions. The Oracle 12c version 12.1.0.2 changes include the following:

- Oracle Enterprise Manager Database Express 12c replaced Oracle Enterprise Manager 11g Database Control.
- The option XMLDB is installed by default in Oracle 12c version 12.1.0.2. You no longer need to install this option yourself.

Amazon RDS PL/SQL packages for Oracle 12c version 12.1.0.2

Oracle 12c version 12.1.0.2 includes a number of new built-in PL/SQL packages. The packages included with Amazon RDS for Oracle 12c version 12.1.0.2 include the following.

Package name	Description
CTX_ANL	The CTX_ANL package is used with AUTO_LEXER and provides procedures for adding and dropping a custom dictionary from the lexer.
DBMS_APP_CONT	The DBMS_APP_CONT package provides an interface to determine if the in-flight transaction on a now unavailable session committed or not, and if the last call on that session completed or not.
DBMS_AUTO_REPORT	The DBMS_AUTO_REPORT package provides an interface to view SQL Monitoring and Real-time Automatic Database Diagnostic Monitor (ADDM) data that has been captured into Automatic Workload Repository (AWR).

DBMS_GOLDENGATE_AUTH	The DBMS_GOLDENGATE_AUTH package provides subprograms for granting privileges to and revoking privileges from GoldenGate administrators.
DBMS_HEAT_MAP	The DBMS_HEAT_MAP package provides an interface to externalize heatmaps at various levels of storage including block, extent, segment, object, and tablespace.
DBMS_ILM	The DBMS_ILM package provides an interface for implementing Information Lifecycle Management (ILM) strategies using Automatic Data Optimization (ADO) policies.
DBMS_ILM_ADMIN	The DBMS_ILM_ADMIN package provides an interface to customize Automatic Data Optimization (ADO) policy execution.
DBMS_PART	The DBMS_PART package provides an interface for maintenance and management operations on partitioned objects.
DBMS_PRIVILEGE_CAPTURE	The DBMS_PRIVILEGE_CAPTURE package provides an interface to database privilege analysis.
DBMS_QOPATCH	The DBMS_QOPATCH package provides an interface to view the installed database patches.
DBMS_REDACT	The DBMS_REDACT package provides an interface to Oracle Data Redaction, which enables you to mask (redact) data that is returned from queries issued by low-privileged users or an application.
DBMS_SPD	The DBMS_SPD package provides subprograms for managing SQL plan directives (SPD).
DBMS_SQL_TRANSLATOR	The DBMS_SQL_TRANSLATOR package provides an interface for creating, configuring, and using SQL translation profiles.
DBMS_SQL_MONITOR	The DBMS_SQL_MONITOR package provides information about real-time SQL Monitoring and real-time Database Operation Monitoring.
DBMS_SYNC_REFRESH	The DBMS_SYNC_REFRESH package provides an interface to perform a synchronous refresh of materialized views.
DBMS_TSDP_MANAGE	The DBMS_TSDP_MANAGE package provides an interface to import and manage sensitive columns and sensitive column types in the database. DBMS_TSDP_MANAGE is used with the DBMS_TSDP_PROTECT package for transparent

	sensitive data protection (TSDP) policies. DBMS_TSDP_MANAGE is available with the Enterprise Edition only.
DBMS_TSDP_PROTECT	The DBMS_TSDP_PROTECT package provides an interface to configure transparent sensitive data protection (TSDP) policies in conjunction with the DBMS_TSDP_MANAGE package. DBMS_TSDP_PROTECT is available with the Enterprise Edition only.
DBMS_XDB_CONFIG	The DBMS_XDB_CONFIG package provides an interface for configuring Oracle XML DB and its repository.
DBMS_XDB_CONSTANTS	The DBMS_XDB_CONSTANTS package provides an interface to commonly used constants. Oracle recommends using constants instead of dynamic strings to avoid typographical errors.
DBMS_XDB_REPOS	The DBMS_XDB_REPOS package provides an interface to operate on the Oracle XML database Repository.
DBMS_XMLSCHEMA_ANNOTATE	The DBMS_XMLSCHEMA_ANNOTATE package provides an interface to manage and configure the structured storage model, mainly through the use of pre-registration schema annotations.
DBMS_XMLSTORAGE_MANAGE	The DBMS_XMLSTORAGE_MANAGE package provides an interface to manage and modify XML storage after schema registration has been completed.
DBMS_XSTREAM_ADM	The DBMS_XSTREAM_ADM package provides interfaces for streaming database changes between an Oracle database and other systems. XStream enables applications to stream out or stream in database changes.
DBMS_XSTREAM_AUTH	The DBMS_XSTREAM_AUTH package provides subprograms for granting privileges to and revoking privileges from XStream administrators.
UTL_CALL_STACK	The UTL_CALL_STACK package provides an interface to provide information about currently executing subprograms.

Oracle 12c version 12.1.0.2 packages not supported

Several Oracle 11g PL/SQL packages are not supported in Oracle 12c version 12.1.0.2. These packages include the following:

- DBMS_AUTO_TASK_IMMEDIATE
- DBMS_CDC_PUBLISH
- DBMS_CDC_SUBSCRIBE
- DBMS_EXPFIL
- DBMS_OBFUSCATION_TOOLKIT
- DBMS_RLMGR
- SDO_NET_MEM

Oracle licensing options

Amazon RDS for Oracle has two licensing options: License Included (LI) and Bring Your Own License (BYOL). After you create an Oracle DB instance on Amazon RDS, you can change the licensing model by modifying the DB instance.

License Included

In the License Included model, you don't need to purchase Oracle licenses separately. AWS holds the license for the Oracle database software. In this model, if you have an AWS Support account with case support, contact AWS Support for both Amazon RDS and Oracle Database service requests. The License Included model is only supported on Amazon RDS for Oracle Database Standard Edition Two (SE2).

Bring Your Own License (BYOL)

In the BYOL model, you can use your existing Oracle Database licenses to run Oracle deployments on Amazon RDS. You must have the appropriate Oracle Database license (with Software Update License and Support) for the DB instance class and Oracle Database edition you wish to run. You must also follow Oracle's policies for licensing

Oracle Database software in the cloud computing environment.

In this model, you continue to use your active Oracle support account, and you contact Oracle directly for Oracle Database service requests. If you have an AWS Support account with case support, you can contact AWS Support for Amazon RDS issues. Amazon Web Services and Oracle have a multi-vendor support process for cases that require assistance from both organizations.

Amazon RDS supports the BYOL model only for Oracle Database Enterprise Edition (EE) and Oracle Database Standard Edition Two (SE2).

Integrating with AWS License Manager

To make it easier to monitor Oracle license usage in the BYOL model, AWS License Manager integrates with Amazon RDS for Oracle. License Manager supports tracking of RDS for Oracle engine editions and

licensing packs based on virtual cores (vCPUs). You can also use License Manager with AWS Organizations to manage all of your organizational accounts centrally.

The following table shows the product information filters for RDS for Oracle.

Filter	Name	Description
Engine Edition	oracle-ee	Oracle Database Enterprise Edition (EE)
	oracle-se2	Oracle Database Standard Edition Two (SE2)
License Pack	data guard	See Working with Oracle replicas for Amazon RDS (Oracle Active Data Guard)
	olap	See Oracle OLAP
	ols	See Oracle Label Security
	diagnostic pack sqlt	See Oracle SQLT

	tuning pack sqlt	See Oracle SQLT

To track license usage of your Oracle DB instances, you can create a license configuration. In this case, RDS for Oracle resources that match the product information filter are automatically associated with the license configuration. Discovery of Oracle DB instances can take up to 24 hours.

Console

To create a license configuration to track the license usage of your Oracle DB instances

1. Go to https://console.aws.amazon.com/license-manager/.
2. Create a license configuration.

Add a rule for an **RDS Product Information Filter** in the **Product Information** panel.

Migrating between Oracle editions

If you have an unused BYOL Oracle license appropriate for the edition and class of DB instance that you plan to run, you can migrate from Standard Edition 2 (SE2) to Enterprise Edition (EE). You can't migrate from Enterprise Edition to other editions.

To change the edition and retain your data

1. Create a snapshot of the DB instance.
2. Restore the snapshot to a new DB instance, and select the Oracle database edition you want to use.
3. (Optional) Delete the old DB instance, unless you want to keep it running and have the appropriate Oracle Database licenses for it.

Licensing Oracle Multi-AZ deployments

Amazon RDS supports Multi-AZ deployments for Oracle as a high-availability, failover solution. We recommend Multi-AZ for production workloads.

If you use the Bring Your Own License model, you must have a license for both the primary DB instance and the standby DB instance in a Multi-AZ deployment.

RDS for Oracle instance classes

The computation and memory capacity of a DB instance is determined by its DB instance class. The DB instance class you need depends on your processing power and memory requirements.

The following are the DB instance classes supported for Oracle.

Oracle edition	19c, 18c, and 12c version 12.2.0.1 support	12c version 12.1.0.2 support
Enterprise Edition (EE) Bring Your Own License (BYOL)	db.m5.large–db.m5.24xlarge db.m4.large–db.m4.16xlarge db.z1d.large–db.z1d.12xlarge db.x1e.xlarge–db.x1e.32xlarge db.x1.16xlarge–db.x1.32xlarge db.r5.large–db.r5.24xlarge db.r5b.large–db.r5b.24xlarge	db.m5.large–db.m5.24xlarge db.m4.large–db.m4.16xlarge db.z1d.large–db.z1d.12xlarge db.x1e.xlarge–db.x1e.32xlarge db.x1.16xlarge–db.x1.32xlarge db.r5.large–db.r5.24xlarge db.r5b.large–db.r5b.24xlarge

	db.r4.large–db.r4.16xlarge db.t3.small–db.t3.2xlarge	db.r4.large–db.r4.16xlarge db.t3.micro–db.t3.2xlarge
Standard Edition 2 (SE2) Bring Your Own License (BYOL)	db.m5.large–db.m5.4xlarge db.m4.large–db.m4.4xlarge db.z1d.large–db.z1d.3xlarge db.r5.large–db.r5.4xlarge db.r5b.large–db.r5b.4xlarge db.r4.large–db.r4.4xlarge db.t3.small–db.t3.2xlarge	db.m5.large–db.m5.4xlarge db.m4.large–db.m4.4xlarge db.z1d.large–db.z1d.3xlarge db.r5.large–db.r5.4xlarge db.r5b.large–db.r5b.4xlarge db.r4.large–db.r4.4xlarge db.t3.micro–db.t3.2xlarge
Standard Edition 2 (SE2) License Included	db.m5.large–db.m5.4xlarge db.m4.large–db.m4.4xlarge db.r5.large–db.r5.4xlarge db.r4.large–db.r4.4xlarge db.t3.small–db.t3.2xlarge	db.m5.large–db.m5.4xlarge db.m4.large–db.m4.4xlarge db.r5.large–db.r5.4xlarge db.r4.large–db.r4.4xlarge db.t3.micro–db.t3.2xlarge

Deprecated DB instance classes for Oracle

Following are the DB instance classes deprecated for Amazon RDS for Oracle:

- db.m1, db.m2, db.m3
- db.t1, db.t2
- db.r1, db.r2, db.r3

The preceding DB instance classes have been replaced by better performing DB instance classes that are generally available at a lower cost. Amazon

RDS for Oracle automatically scales DB instances to DB instance classes that are not deprecated.

If you have DB instances that use deprecated DB instance classes, Amazon RDS will modify each one automatically to use a comparable DB instance class that is not deprecated. You can change the DB instance class for a DB instance yourself by modifying the DB instance.

If you have DB snapshots of DB instances that were using deprecated DB instance classes, you can choose a DB instance class that is not deprecated when you restore the DB snapshots.

RDS for Oracle features

Amazon RDS for Oracle supports most of the features and capabilities of Oracle Database. Some features might have limited support or restricted privileges. Some features are only available in

Enterprise Edition, and some require additional licenses. For more information about Oracle Database features for specific Oracle Database versions, see the *Oracle Database Licensing Information User Manual* for the version you're using.

Supported features for RDS for Oracle

Amazon RDS Oracle supports the following Oracle Database features:

- Advanced Compression
- Application Express (APEX)
- Automatic Memory Management
- Automatic Undo Management
- Automatic Workload Repository (AWR)
- Active Data Guard with Maximum Performance in the same AWS Region or across AWS Regions
- Continuous Query Notification (version 12.1.0.2.v7 and later)
- Data Redaction

- Database Change Notification
- Database In-Memory (version 12.1 and later)
- Distributed Queries and Transactions
- Edition-Based Redefinition
- EM Express (12c and later)
- Fine-Grained Auditing
- Flashback Table, Flashback Query, Flashback Transaction Query
- Import/export (legacy and Data Pump) and SQL*Loader
- Java Virtual Machine (JVM)
- Label Security (version 12.1 and later)
- Locator
- Materialized Views
- Multimedia
- Network encryption
- Partitioning
- Spatial and Graph
- Star Query Optimization
- Streams and Advanced Queuing

- Summary Management – Materialized View Query Rewrite
- Text (File and URL data store types are not supported)
- Total Recall
- Transparent Data Encryption (TDE)
- Unified Auditing, Mixed Mode (version 12.1 and later)
- XML DB (without the XML DB Protocol Server)
- Virtual Private Database

Unsupported features for RDS for Oracle

Amazon RDS Oracle doesn't support the following Oracle Database features:

- Automatic Storage Management (ASM)
- Database Vault
- Flashback Database
- Messaging Gateway
- Multitenant

- Oracle Enterprise Manager Cloud Control Management Repository
- Real Application Clusters (Oracle RAC)
- Real Application Testing
- Unified Auditing, Pure Mode
- Workspace Manager (WMSYS) schema

In general, Amazon RDS doesn't prevent you from creating schemas for unsupported features. However, if you create schemas for Oracle features and components that require SYS privileges, you can damage the data dictionary and affect the availability of your instance.

RDS for Oracle parameters

In Amazon RDS, you manage parameters using parameter groups. To view the supported parameters for a specific Oracle Database edition and version, run the AWS CLI `describe-engine-default-parameters` command.

For example, to view the supported parameters for Oracle Enterprise Edition version 12.2, run the following command.

```
aws rds describe-engine-default-parameters \
  --db-parameter-group-family oracle-ee-12.2
```

RDS for Oracle character sets

Amazon RDS for Oracle supports two types of character sets: the DB character set and national character set.

DB character set

The Oracle database character set is used in the CHAR, VARCHAR2, and CLOB data types. The database also uses this character set for metadata such as table names, column names, and SQL statements. The Oracle database character set is typically referred to as the *DB character set*.

You set the character set when you create a DB instance. You can't change the DB character set after you create the database.

Supported DB character sets

The following table lists the Oracle DB character sets that are supported in Amazon RDS. You can use a value from this table with the `--character-set-name` parameter of the AWS CLI create-db-instance command or with the `CharacterSetName` parameter of the Amazon RDS API CreateDBInstance operation.

Value	Description
AL32UTF8	Unicode 5.0 UTF-8 Universal character set (default)
AR8ISO8859P6	ISO 8859-6 Latin/Arabic

AR8MSWIN1256	Microsoft Windows Code Page 1256 8-bit Latin/ Arabic
BLT8ISO8859P13	ISO 8859-13 Baltic
BLT8MSWIN1257	Microsoft Windows Code Page 1257 8-bit Baltic
CL8ISO8859P5	ISO 88559-5 Latin/Cyrillic
CL8MSWIN1251	Microsoft Windows Code Page 1251 8-bit Latin/ Cyrillic
EE8ISO8859P2	ISO 8859-2 East European
EL8ISO8859P7	ISO 8859-7 Latin/Greek
EE8MSWIN1250	Microsoft Windows Code Page 1250 8-bit East European
EL8MSWIN1253	Microsoft Windows Code Page 1253 8-bit Latin/ Greek

IW8ISO8859P8	ISO 8859-8 Latin/Hebrew
IW8MSWIN1255	Microsoft Windows Code Page 1255 8-bit Latin/ Hebrew
JA16EUC	EUC 24-bit Japanese
JA16EUCTILDE	Same as JA16EUC except for mapping of wave dash and tilde to and from Unicode
JA16SJIS	Shift-JIS 16-bit Japanese
JA16SJISTILDE	Same as JA16SJIS except for mapping of wave dash and tilde to and from Unicode
KO16MSWIN949	Microsoft Windows Code Page 949 Korean
NE8ISO8859P10	ISO 8859-10 North European

NEE8ISO8859P4	ISO 8859-4 North and Northeast European
TH8TISASCII	Thai Industrial Standard 620-2533-ASCII 8-bit
TR8MSWIN1254	Microsoft Windows Code Page 1254 8-bit Turkish
US7ASCII	ASCII 7-bit American
UTF8	Unicode 3.0 UTF-8 Universal character set, CESU-8 compliant
VN8MSWIN1258	Microsoft Windows Code Page 1258 8-bit Vietnamese
WE8ISO8859P1	Western European 8-bit ISO 8859 Part 1
WE8ISO8859P15	ISO 8859-15 West European
WE8ISO8859P9	ISO 8859-9 West Euro-

	pean and Turkish
WE8MSWIN1252	Microsoft Windows Code Page 1252 8-bit West European
ZHS16GBK	GBK 16-bit Simplified Chinese
ZHT16HKSCS	Microsoft Windows Code Page 950 with Hong Kong Supplementary Character Set HKSCS-2001. Character set conversion is based on Unicode 3.0.
ZHT16MSWIN950	Microsoft Windows Code Page 950 Traditional Chinese
ZHT32EUC	EUC 32-bit Traditional Chinese

NLS_LANG environment variable

A *locale* is a set of information addressing linguistic and cultural requirements that corresponds to a given language and country. Setting the NLS_LANG environment variable in your client's environment is the simplest way to specify locale behavior for Oracle. This variable sets the language and territory used by the client application and the database server. It also indicates the client's character set, which corresponds to the character set for data entered or displayed by a client application.

NLS initialization parameters

You can also set the following National Language Support (NLS) initialization parameters at the instance level for an Oracle DB instance in Amazon RDS:

- NLS_DATE_FORMAT
- NLS_LENGTH_SEMANTICS
- NLS_NCHAR_CONV_EXCP

- NLS_TIME_FORMAT
- NLS_TIME_TZ_FORMAT
- NLS_TIMESTAMP_FORMAT
- NLS_TIMESTAMP_TZ_FORMAT

You can set other NLS initialization parameters in your SQL client. For example, the following statement sets the NLS_LANGUAGE initialization parameter to GERMAN in a SQL client that is connected to an Oracle DB instance:

```
ALTER SESSION SET NLS_LANGUAGE=GERMAN;
```

National character set

The national character set is used in the NCHAR, NVARCHAR2, and NCLOB data types. The national character set is typically referred to as the *NCHAR character set*. Unlike the DB character set, the NCHAR character set doesn't affect database metadata.

The NCHAR character set supports the following character sets:

- AL16UTF16 (default)
- UTF8

You can specify either value with the `--nchar-character-set-name` parameter of the create-db-instance command (AWS CLI version 2 only). If you use the Amazon RDS API, specify the `NcharCharacterSetName` parameter of CreateDBInstance operation. You can't change the national character set after you create the database.

RDS for Oracle limitations

Following are important limitations of using Amazon RDS for Oracle.

Oracle file size limits in Amazon RDS

The maximum file size on Amazon RDS Oracle DB instances is 16 TiB (tebibytes). If you try to resize a data file in a bigfile tablespace to a value over the limit, you receive an error such as the following.

```
ORA-01237: cannot extend datafile 6
ORA-01110: data file 6: '/rdsdbdata/db/mydir/
datafile/myfile.dbf'
ORA-27059: could not reduce file size
Linux-x86_64 Error: 27: File too large
Additional information: 2
```

Public synonyms for Oracle-supplied schemas

Don't create or modify public synonyms for Oracle-supplied schemas, including SYS, SYSTEM, and RDSADMIN. Such actions might result in invalidation of core database components and affect the availability of your DB instance.

You can create public synonyms referencing objects in your own schemas.

Schemas for unsupported features

In general, Amazon RDS doesn't prevent you from creating schemas for unsupported features. However, if you create schemas for Oracle features and components that require SYS privileges, you can damage the data dictionary and affect your instance availability.

Limitations for Oracle DBA privileges

In the database, a *role* is a collection of privileges that you can grant to or revoke from a user. An Oracle database uses roles to provide security.

The predefined role DBA normally allows all administrative privileges on an Oracle database. When you create a DB instance, your master user account gets DBA privileges (with some limitations). To

deliver a managed experience, an RDS for Oracle database doesn't provide the following privileges for the DBA role:

- ALTER DATABASE
- ALTER SYSTEM
- CREATE ANY DIRECTORY
- DROP ANY DIRECTORY
- GRANT ANY PRIVILEGE
- GRANT ANY ROLE

Use the master user account for administrative tasks such as creating additional user accounts in the database. You can't use SYS, SYSTEM, and other Oracle-supplied administrative accounts.

Administering your Oracle DB instance

Following are the common management tasks you perform with an Amazon RDS DB instance. Some tasks are the same for all RDS DB instances. Other tasks are specific to RDS for Oracle.

The following tasks are common to all RDS databases, but Oracle has special considerations. For example, connect to an Oracle Database using the Oracle clients SQL*Plus and SQL Developer.

Task area	Relevant documentation
Instance Classes, Storage, and PIOPS If you are creating a production instance, learn how instance classes, storage types, and Provisioned IOPS work in Amazon RDS.	RDS for Oracle instance classes Amazon RDS storage types

Multi-AZ Deployments A production DB instance should use Multi-AZ deployments. Multi-AZ deployments provide increased availability, data durability, and fault tolerance for DB instances.	High availability (Multi-AZ) for Amazon RDS
Amazon Virtual Private Cloud (VPC) If your AWS account has a default VPC, then your DB instance is automatically created inside the default VPC. If your account doesn't have a default VPC, and you want the DB instance in a VPC, create the VPC and subnet	Determining whether you are using the EC2-VPC or EC2-Classic platform Working with a DB instance in a VPC

groups before you create the instance.	
Security Groups By default, DB instances use a firewall that prevents access. Make sure you create a security group with the correct IP addresses and network configuration to access the DB instance. The security group you create depends on which Amazon EC2 platform your DB instance is on, and whether you will access your DB instance from an Amazon EC2 instance. In general, if your DB instance is on the EC2-	Determining whether you are using the EC2-VPC or EC2-Classic platform Controlling access with security groups

Classic platform, you should create a DB security group. Also, if your instance is on the EC2-VPC platform, you should create a VPC security group.	
Parameter Groups If your DB instance is going to require specific database parameters, you should create a parameter group before you create the DB instance.	Working with DB parameter groups
Option Groups If your DB instance will require specific database options, you should create an option group before you create the DB instance.	Adding options to Oracle DB instances

Connecting to Your DB Instance After creating a security group and associating it to a DB instance, you can connect to the DB instance using any standard SQL client application such as Oracle SQL*Plus.	Connecting to your Oracle DB instance
Backup and Restore You can configure your DB instance to take automated backups, or take manual snapshots, and then restore instances from the backups or snapshots.	Backing up and restoring an Amazon RDS DB instance
Monitoring You can monitor an Oracle DB instance	Viewing DB instance metrics

by using CloudWatch Amazon RDS metrics, events, and enhanced monitoring.	Viewing Amazon RDS events
Log Files You can access the log files for your Oracle DB instance.	Accessing Amazon RDS database log files

Following, you can find a description for Amazon RDS–specific implementations of common DBA tasks for RDS Oracle. To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances, and restricts access to certain system procedures and tables that require advanced privileges. In many of the tasks, you run the rdsadmin package, which is an Amazon RDS–specific tool that enables you to administer your database.

The following are common DBA tasks for DB instances running Oracle:

- System tasks

Disconnecting a session	Amazon RDS method: rdsadmin.rdsadmin_util.disconnect Oracle method: alter system disconnect session
Terminating a session	Amazon RDS method: rdsadmin.rdsadmin_util.kill Oracle method: alter system kill session
Canceling a SQL statement in a session	Amazon RDS method: rdsadmin.rdsadmin_util.cancel Oracle method: alter system cancel sql
Enabling and disabling restricted sessions	Amazon RDS method: rdsadmin.rdsadmin_util.restricted_session Oracle method: alter system enable restricted session
Flushing the shared pool	Amazon RDS method: rdsadmin.rdsadmin_util.flush_shared_pool Oracle method: alter system flush shared_pool
Flushing the buffer cache	Amazon RDS method: rdsadmin.rdsadmin_util.flush_buffer_cache Oracle method: alter system flush buffer_cache
Granting SELECT or EXECUTE privileges to SYS objects	Amazon RDS method: rdsadmin.rdsadmin_util.grant_sys_object Oracle method: grant
Revoking SELECT or EXECUTE privileges on SYS objects	Amazon RDS method: rdsadmin.rdsadmin_util.revoke_sys_object Oracle method: revoke
Granting privileges to non-master users	Amazon RDS method: grant Oracle method: grant
Creating custom functions to verify passwords	Amazon RDS method: rdsadmin.rdsadmin_password_verify.create_verify_function Amazon RDS method: rdsadmin.rdsadmin_password_verify.create_passthrough_verify_fcn
Setting up a custom DNS server	—
Listing allowed system diagnostic events	Amazon RDS method: rdsadmin.rdsadmin_util.list_allowed_system_events Oracle method: —

Setting system diagnostic events	Amazon RDS method: rdsadmin.rdsadmin_util.set_allowed_system_events Oracle method: ALTER SYSTEM SET EVENTS *'set_event_clause'*
Listing system diagnostic events that are set	Amazon RDS method: rdsadmin.rdsadmin_util.list_set_system_events Oracle method: ALTER SESSION SET EVENTS 'IMMEDIATE EVENTDUMP(SYSTEM)'
Unsetting system diagnostic events	Amazon RDS method: rdsadmin.rdsadmin_util.unset_system_event Oracle method: ALTER SYSTEM SET EVENTS *'unset_event_clause'*

- Database tasks

Changing the global name of a database	Amazon RDS method: rdsadmin.rdsadmin_util.rename_global_name Oracle method: alter database rename
Creating and sizing tablespaces	Amazon RDS method: create tablespace Oracle method: alter database
Setting the default tablespace	Amazon RDS method: rdsadmin.rdsadmin_util.alter_default_tablespace Oracle method: alter database default tablespace
Setting the default temporary tablespace	Amazon RDS method: rdsadmin.rdsadmin_util.alter_default_temp_tablespace Oracle method: alter database default temporary tablespace
Checkpointing a database	Amazon RDS method: rdsadmin.rdsadmin_util.checkpoint Oracle method: alter system checkpoint
Setting distributed recovery	Amazon RDS method: rdsadmin.rdsadmin_util.enable_distr_recovery Oracle method: alter system enable distributed recovery
Setting the database time zone	Amazon RDS method: rdsadmin.rdsadmin_util.alter_db_time_zone Oracle method: alter database set time_zone
Working with Oracle external tables	—
Generating performance reports with Automatic Workload Repository (AWR)	Amazon RDS method: rdsadmin.rdsadmin_diagnostic_util procedures Oracle method: dbms_workload_repository package
Adjusting database links for use with DB instances in a VPC	—
Setting the default edition for a DB instance	Amazon RDS method: rdsadmin.rdsadmin_util.alter_default_edition Oracle method: alter database default edition

Enabling auditing for the SYS.AUD$ table	Amazon RDS method: rdsadmin.rdsadmin_master_util.audit_all_sys_aud_table Oracle method: audit
Disabling auditing for the SYS.AUD$ table	Amazon RDS method: rdsadmin.rdsadmin_master_util.noaudit_all_sys_aud_table Oracle method: noaudit
Cleaning up interrupted online index builds	Amazon RDS method: rdsadmin.rdsadmin_dbms_repair.online_index_clean Oracle method: dbms_repair.online_index_clean
Skipping corrupt blocks	Amazon RDS method: Several rdsadmin.rdsadmin_dbms_repair procedures Oracle method: dbms_repair package
Resizing the temporary tablespace in a read replica	Amazon RDS method: rdsadmin.rdsadmin_util.resize_temp_tablespace or rdsadmin.rdsadmin_util.resize_tempfile procedure Oracle method: —
Purging the recycle bin	Amazon RDS method: exec rdsadmin.rdsadmin_util.purge_dba_recyclebin Oracle method: purge dba_recyclebin

- Log tasks

Setting force logging	Amazon RDS method: rdsadmin.rdsadmin_util.force_logging Oracle method: alter database force logging
Setting supplemental logging	Amazon RDS method: rdsadmin.rdsadmin_util.alter_supplemental_logging Oracle method: alter database add supplemental log
Switching online log files	Amazon RDS method: rdsadmin.rdsadmin_util.switch_logfile Oracle method: alter system switch logfile
Adding online redo logs	Amazon RDS method: rdsadmin.rdsadmin_util.add_logfile
Dropping online redo logs	Amazon RDS method: rdsadmin.rdsadmin_util.drop_logfile
Resizing online redo logs	—
Retaining archived redo logs	Amazon RDS method: rdsadmin.rdsadmin_util.set_configuration

Accessing transaction logs	Amazon RDS method: rdsadmin.rdsadmin_master_util.create_archivelog_dir Amazon RDS method: rdsadmin.rdsadmin_master_util.create_onlinelog_dir

- RMAN tasks

Validating DB instance files	Amazon RDS method: rdsadmin_rman_util.*procedure* Oracle method: RMAN VALIDATE
Enabling and disabling block change tracking	Amazon RDS method: rdsadmin_rman_util.*procedure* Oracle method: ALTER DATABASE
Crosschecking archived redo logs	Amazon RDS method: rdsadmin_rman_util.crosscheck_archivelog Oracle method: RMAN BACKUP
Backing up archived redo logs	Amazon RDS method: rdsadmin_rman_util.*procedure* Oracle method: RMAN BACKUP
Performing a full database backup	Amazon RDS method: rdsadmin_rman_util.backup_database_full Oracle method: RMAN BACKUP
Performing an incremental database backup	Amazon RDS method: rdsadmin_rman_util.backup_database_incremental Oracle method: RMAN BACKUP
Performing a tablespace backup	Amazon RDS method: rdsadmin_rman_util.backup_database_tablespace Oracle method: RMAN BACKUP

- Oracle Scheduler tasks

Modifying DBMS_SCHEDULER jobs	Amazon RDS method: dbms_scheduler.set_attribute Oracle method: dbms_scheduler.set_attribute
Setting the time zone for Oracle Scheduler jobs	Amazon RDS method: dbms_scheduler.set_attribute Oracle method: dbms_scheduler.set_attribute
Disabling SYS-owned Oracle Scheduler jobs	Amazon RDS method: rdsadmin.rdsadmin_dbms_scheduler.disable Oracle method: dbms_scheduler.disable
Enabling SYS-owned Oracle Scheduler jobs	Amazon RDS method: rdsadmin.rdsadmin_dbms_scheduler.enable Oracle method: dbms_scheduler.enable

Modifying the repeat interval for jobs of CALENDAR type	Amazon RDS method: rdsadmin.rdsadmin_dbms_scheduler.set_attribute Oracle method: dbms_scheduler.set_attribute
Modifying the repeat interval for jobs of NAMED type	Amazon RDS method: rdsadmin.rdsadmin_dbms_scheduler.set_attribute Oracle method: dbms_scheduler.set_attribute

- Diagnostic tasks

Listing incidents	Amazon RDS method: rdsadmin.rdsadmin_adrci_util.list_adrci_incidents Oracle method: ADRCI command show incident
Listing problems	Amazon RDS method: rdsadmin.rdsadmin_adrci_util.list_adrci_problem Oracle method: ADRCI command show problem
Creating incident packages	Amazon RDS method: rdsadmin.rdsadmin_adrci_util.create_adrci_package Oracle method: ADRCI command ips create package
Showing trace files	Amazon RDS method: rdsadmin.rdsadmin_adrci_util.show_adrci_tracefile Oracle method: ADRCI command show tracefile

- Other tasks

Creating and dropping directories in the main data storage space	Amazon RDS method: rdsadmin.rdsadmin_util.create_directory Oracle method: CREATE DIRECTORY Amazon RDS method: rdsadmin.rdsadmin_util.drop_directory Oracle method: DROP DIRECTORY
Listing files in a DB instance directory	Amazon RDS method: rdsadmin.rds_file_util.listdir Oracle method: —
Reading files in a DB instance directory	Amazon RDS method: rdsadmin.rds_file_util.read_text_file Oracle method: —
Accessing Opatch files	Amazon RDS method: rdsadmin.rds_file_util.read_text_file or rdsadmin.tracefile_listing Oracle method: opatch
Setting parameters for advisor tasks	Amazon RDS method: rdsadmin.rdsadmin_util.advisor_task_set_parameter Oracle method: Various stored package procedures

Disabling AUTO_STATS_ADVISOR_TASK	Amazon RDS method: rdsadmin.rdsadmin_util.advisor_task_drop Oracle method: —
Re-enabling AUTO_STATS_ADVISOR_TASK	Amazon RDS method: rdsadmin.rdsadmin_util.dbms_stats_init Oracle method: —
Enabling HugePages for an Oracle DB instance	Amazon RDS method: use_large_pages RDS parameter Oracle method: use_large_pages initialization parameter
Enabling extended data types	Amazon RDS method: max_string_size RDS parameter Oracle method: max_string_size initialization parameter

You can also use Amazon RDS procedures for Amazon S3 integration with Oracle and for running OEM Management Agent database tasks.

Disconnecting a session

To disconnect the current session by ending the dedicated server process, use the Amazon RDS procedure rdsadmin.rdsadmin_util.disconnect. The disconnect procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
sid	number	—	Yes	The session identifier.
serial	number	—	Yes	The serial number of the session.

method	varchar	'IMMEDIATE'	No	Valid values are 'IMMEDIATE' or 'POST_TRANSACTION'.

The following example disconnects a session.

```
begin
  rdsadmin.rdsadmin_util.disconnect(
    sid    => sid,
    serial => serial_number);
end;
/
```

To get the session identifier and the session serial number, query the V$SESSION view. The following example gets all sessions for the user AWSUSER.

```
select SID, SERIAL#, STATUS from V$SESSION where
USERNAME = 'AWSUSER';
```

The database must be open to use this method.

Terminating a session

To terminate a session, use the Amazon RDS procedure rdsadmin.rdsadmin_util.kill. The kill procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
sid	number	—	Yes	The session identifier.
serial	number	—	Yes	The serial number of the session.
method	varchar	null	No	Valid values are 'IMMEDIATE' or 'PROCESS'.

The following example terminates a session.

```
begin
  rdsadmin.rdsadmin_util.kill(
    sid    => sid,
    serial => serial_number);
end;
/
```

To get the session identifier and the session serial number, query the V$SESSION view. The following example gets all sessions for the user AWSUSER.

```
SELECT SID, SERIAL#, STATUS FROM V$SESSION
WHERE USERNAME = 'AWSUSER';
```

You can specify either IMMEDIATE or PROCESS as a value for the method parameter. By specifying PROCESS as the method value, you can terminate the processes associated with a session. Do this only if terminating the session using IMMEDIATE as the method value was unsuccessful.

Canceling a SQL statement in a session

To cancel a SQL statement in a session, use the Amazon RDS procedure rdsadmin.rdsadmin_util.cancel.

The cancel procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
sid	number	—	Yes	The session identifier.
serial	number	—	Yes	The serial number of the session.
sql_id	varchar2	null	No	The SQL identifier of the SQL statement.

The following example cancels a SQL statement in a session.

```
begin
  rdsadmin.rdsadmin_util.cancel(
    sid    => sid,
    serial => serial_number,
    sql_id => sql_id);
end;
/
```

To get the session identifier, the session serial number, and the SQL identifier of a SQL statement, query the V$SESSION view. The following example gets all sessions and SQL identifiers for the user AWSUSER.

```
select SID, SERIAL#, SQL_ID, STATUS from V$SESSION where USERNAME = 'AWSUSER';
```

Enabling and disabling restricted sessions

To enable and disable restricted sessions, use the Amazon RDS procedure rdsadmin.rdsadmin_util.restricted_session. The restricted_session procedure has the following parameters.

Parameter name	Data type	Default	Yes	Description
p_enable	boolean	true	No	Set to true to enable restricted

				sessions, false to disable restricted sessions.

The following example shows how to enable and disable restricted sessions.

```
/* Verify that the database is currently unrestricted.
*/

SELECT LOGINS FROM V$INSTANCE;


LOGINS
-------
ALLOWED

/* Enable restricted sessions */

exec rdsadmin.rdsadmin_util.restricted_ses-
sion(p_enable => true);
```

```
/* Verify that the database is now restricted. */

SELECT LOGINS FROM V$INSTANCE;

LOGINS
----------
RESTRICTED


/* Disable restricted sessions */

exec rdsadmin.rdsadmin_util.restricted_ses-
sion(p_enable => false);


/* Verify that the database is now unrestricted again.
*/

SELECT LOGINS FROM V$INSTANCE;

LOGINS
-------
```

```
ALLOWED
```

Flushing the shared pool

To flush the shared pool, use the Amazon RDS procedure rdsadmin.rdsadmin_util.flush_shared_pool. The flush_shared_pool procedure has no parameters.

The following example flushes the shared pool.

```
exec rdsadmin.rdsadmin_util.flush_shared_pool;
```

Flushing the buffer cache

To flush the buffer cache, use the Amazon RDS procedure rdsadmin.rdsadmin_util.flush_buffer_cache. The flush_buffer_cache procedure has no parameters.

The following example flushes the buffer cache.

```
exec rdsadmin.rdsadmin_util.flush_buffer_cache;
```

Granting SELECT or EXECUTE privileges to SYS objects

Usually you transfer privileges by using roles, which can contain many objects. To grant privileges to a single object, use the Amazon RDS procedure rdsadmin.rdsadmin_util.grant_sys_object. The procedure grants only privileges that the master user has already been granted through a role or direct grant.

The grant_sys_object procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
p_obj_name	varchar2	—	Yes	The name of the object to grant privileges for. The object can be a directory, function, package, procedure, sequence, table, or view. Object names must be spelled exactly as they appear in DBA_OBJECTS. Most system objects are defined in uppercase, so we recommend that you try that first.

p_grantee	varchar2	—	Yes	The name of the object to grant privileges to. The object can be a schema or a role.
p_privilege	varchar2	null	Yes	—
p_grant_option	boolean	false	No	Set to true to use the with grant option. The p_grant_option parameter is supported for 12.1.0.2.v4 and later, all 12.2.0.1 versions, all 18.0.0.0 versions, and all 19.0.0 versions.

The following example grants select privileges on an object named `V_$SESSION` to a user named `USER1`.

```
begin
  rdsadmin.rdsadmin_util.grant_sys_object(
    p_obj_name  => 'V_$SESSION',
    p_grantee   => 'USER1',
    p_privilege => 'SELECT');
end;
/
```

The following example grants select privileges on an object named `V_$SESSION` to a user named `USER1` with the grant option.

```
begin
  rdsadmin.rdsadmin_util.grant_sys_object(
    p_obj_name    => 'V_$SESSION',
    p_grantee     => 'USER1',
    p_privilege   => 'SELECT',
    p_grant_option => true);
end;
/
```

To be able to grant privileges on an object, your account must have those privileges granted to it directly with the grant option, or via a role granted using `with admin option`. In the most common case, you may want to grant SELECT on a DBA view that has been granted to the `SELECT_CATALOG_ROLE` role. If that role isn't already directly granted to your user using `with admin option`, then you can't transfer the privilege. If you have the DBA privilege, then you can grant the role directly to another user.

The following example grants the SELECT_CATALOG_ROLE and EXECUTE_CATALOG_ROLE to USER1. Since the with admin option is used, USER1 can now grant access to SYS objects that have been granted to SELECT_CATALOG_ROLE.

```
GRANT SELECT_CATALOG_ROLE TO USER1 WITH ADMIN OPTION;
GRANT EXECUTE_CATALOG_ROLE to USER1 WITH ADMIN OPTION;
```

Objects already granted to PUBLIC do not need to be re-granted. If you use the grant_sys_object procedure to re-grant access, the procedure call succeeds.

Revoking SELECT or EXECUTE

privileges on SYS objects

To revoke privileges on a single object, use the Amazon RDS procedure rdsadmin.rdsadmin_u-

til.revoke_sys_object. The procedure only revokes privileges that the master account has already been granted through a role or direct grant.

The revoke_sys_object procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
p_obj_name	varchar2	—	Yes	The name of the object to revoke privileges for. The object can be a directory, function, package, procedure, sequence, table, or view. Object names must be spelled exactly as they appear in DBA_OBJECTS. Most system objects are defined in upper case, so we recommend you try that first.
p_revokee	varchar2	—	Yes	The name of the object to revoke privileges for. The object can be a schema or a role.
p_privilege	varchar2	null	Yes	—

The following example revokes select privileges on an object named V_$SESSION from a user named USER1.

```
begin
  rdsadmin.rdsadmin_util.revoke_sys_object(
    p_obj_name  => 'V_$SESSION',
    p_revokee   => 'USER1',
    p_privilege => 'SELECT');
end;
/
```

Granting privileges to non-master users

You can grant select privileges for many objects in the SYS schema by using the SELECT_CATALOG_ROLE role. The SELECT_CATALOG_ROLE role gives users SELECT privileges on data dictionary views. The following example grants the role SELECT_CATALOG_ROLE to a user named user1.

```
GRANT SELECT_CATALOG_ROLE TO user1;
```

You can grant EXECUTE privileges for many objects in the SYS schema by using the EXECUTE_CATALOG_ROLE role. The EXECUTE_CATALOG_ROLE role gives users EXECUTE privileges for packages and procedures in the data dictionary. The following example grants the role EXECUTE_CATALOG_ROLE to a user named *user1*.

```
GRANT EXECUTE_CATALOG_ROLE TO user1;
```

The following example gets the permissions that the roles SELECT_CATALOG_ROLE and EXECUTE_CATALOG_ROLE allow.

```
SELECT *
 FROM ROLE_TAB_PRIVS
 WHERE ROLE IN ('SELECT_CATALOG_ROLE','EXECUTE_CATALOG_ROLE')
ORDER BY ROLE, TABLE_NAME ASC;
```

The following example creates a non-master user named user1, grants the CREATE SESSION privilege, and grants the SELECT privilege on a database named *sh.sales*.

```
CREATE USER user1 IDENTIFIED BY PASSWORD;
GRANT CREATE SESSION TO user1;
GRANT SELECT ON sh.sales TO user1;
```

Creating custom functions to verify passwords

You can create a custom password verification function in two ways. If you want to use standard verification logic, and to store your function in the SYS schema, use the create_verify_function procedure. If you want to use custom verification logic, or you don't want to store your function in the SYS schema, use the create_passthrough_verify_fcn procedure.

The create_verify_function procedure

The create_verify_function procedure is supported for Oracle version 12.1.0.2.v5 and later, all 12.2.0.1 versions, all 18.0.0.0 versions, and all 19.0.0 versions.

You can create a custom function to verify passwords by using the Amazon RDS procedure rdsadmin.rdsadmin_password_verify.create_verify_function. The create_verify_function procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
p_verify_function_name	varchar2	—	Yes	The name for your custom function. This function is created for you in the SYS schema. You assign this function to user profiles.
p_min_length	number	8	No	The minimum number of characters required.
p_max_length	number	256	No	The maximum number of characters allowed.

p_min_letters	number	1	No	The minimum number of letters required.
p_min_uppercase	number	0	No	The minimum number of upper-case letters required.
p_min_lowercase	number	0	No	The minimum number of lowercase letters required.
p_min_digits	number	1	No	The minimum number of digits required.
p_min_special	number	0	No	The minimum number of special characters required.
p_min_different_chars	number	3	No	The minimum number of different characters required between the old and new password.
p_disallow_username	boolean	true	No	Set to true to disallow the user name in the password.
p_disallow_reverse	boolean	true	No	Set to true to disallow the reverse of the user name in the password.
p_disallow_db_name	boolean	true	No	Set to true to disallow the database or server name in the password.
p_disallow_simple_strings	boolean	true	No	Set to true to disallow simple strings as the password.

p_disallow_whitespace	boolean	false	No	Set to true to disallow white space characters in the password.
p_disallow_at_sign	boolean	false	No	Set to true to disallow the @ character in the password.

You can create multiple password verification functions.

There are restrictions on the name of your custom function. Your custom function can't have the same name as an existing system object. The name can be no more than 30 characters long. Also, the name must include one of the following strings: PASSWORD, VERIFY, COMPLEXITY, ENFORCE, or STRENGTH.

The following example creates a function named CUSTOM_PASSWORD_FUNCTION. The function requires that a password has at least 12 characters, 2 uppercase characters, 1 digit, and 1 special character, and that the password disallows the @ character.

```
begin
  rdsadmin.rdsadmin_password_verify.create_ver-
ify_function(
    p_verify_function_name => 'CUSTOM_PASS-
WORD_FUNCTION',
    p_min_length       => 12,
    p_min_uppercase    => 2,
    p_min_digits       => 1,
    p_min_special      => 1,
    p_disallow_at_sign   => true);
end;
/
```

To see the text of your verification function, query DBA_SOURCE. The following example gets the text of a custom password function named CUSTOM_PASSWORD_FUNCTION.

```
COL TEXT FORMAT a150

 SELECT TEXT
  FROM DBA_SOURCE
```

```
  WHERE OWNER = 'SYS'
    AND NAME = 'CUSTOM_PASSWORD_FUNCTION'
ORDER BY LINE;
```

To associate your verification function with a user profile, use alter profile. The following example associates a verification function with the DEFAULT- user profile.

```
ALTER PROFILE DEFAULT LIMIT PASSWORD_VERIFY_FUNCTION CUSTOM_PASSWORD_FUNCTION;
```

To see what user profiles are associated with what verification functions, query DBA_PROFILES.
The following example gets the profiles that are associated with the custom verification function named CUSTOM_PASSWORD_FUNCTION.

```
SELECT * FROM DBA_PROFILES WHERE RESOURCE_NAME = 'PASSWORD' AND LIMIT = 'CUSTOM_PASSWORD_FUNCTION';
```

```
PROFILE                  RESOURCE_NAME                  RE-
SOURCE  LIMIT
------------------------ ------------------------------
-------- ----------------------
DEFAULT                  PASSWORD_VERIFY_FUNCTION
PASSWORD  CUSTOM_PASSWORD_FUNCTION
```

The following example gets all profiles and the password verification functions that they are associated with.

```
SELECT * FROM DBA_PROFILES WHERE RE-
SOURCE_NAME = 'PASSWORD_VERIFY_FUNC-
TION';

PROFILE                  RESOURCE_NAME                  RE-
SOURCE  LIMIT
------------------------ ------------------------------
-------- ----------------------
```

```
DEFAULT                PASSWORD_VERIFY_FUNCTION
PASSWORD  CUSTOM_PASSWORD_FUNCTION
RDSADMIN               PASSWORD_VERIFY_FUNC-
TION      PASSWORD  NULL
```

The create_passthrough_verify_fcn procedure

The create_passthrough_verify_fcn procedure is supported for Oracle version 12.1.0.2.v7 and later, all 12.2.0.1 versions, all 18.0.0.0 versions, and all 19.0.0 versions.

You can create a custom function to verify passwords by using the Amazon RDS procedure rdsadmin.rdsadmin_password_verify.create_passthrough_verify_fcn. The create_passthrough_verify_fcn procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
p_verify_function_name	varchar2	—	Yes	The name for your custom verification function. This is a wrapper function that

				is created for you in the SYS schema, and it doesn't contain any verification logic. You assign this function to user profiles.
p_target_owner	varchar2	—	Yes	The schema owner for your custom verification function.
p_target_function_name	varchar2	—	Yes	The name of your existing custom function that contains the verification logic. Your custom function must return a boolean. Your function should return true if the password is valid and false if the password is invalid.

The following example creates a password verification function that uses the logic from the function named PASSWORD_LOGIC_EXTRA_STRONG.

```
begin
  rdsadmin.rdsadmin_password_verify.create_
passthrough_verify_fcn(
    p_verify_function_name => 'CUSTOM_PASS-
WORD_FUNCTION',
```

```
    p_target_owner      => 'TEST_USER',
    p_target_function_name => 'PASSWORD_LOG-
IC_EXTRA_STRONG');
end;
/
```

To associate the verification function with a user profile, use alter profile. The following example associates the verification function with the DEFAULT user profile.

```
ALTER PROFILE DEFAULT LIMIT PASSWORD_VERI-
FY_FUNCTION CUSTOM_PASSWORD_FUNCTION;
```

Setting up a custom DNS server

Amazon RDS supports outbound network access on your DB instances running Oracle. Amazon RDS Oracle allows Domain Name Service (DNS) resolution from a custom DNS server owned by the customer. You can resolve only fully qualified domain names

from your Amazon RDS DB instance through your custom DNS server.

After you set up your custom DNS name server, it takes up to 30 minutes to propagate the changes to your DB instance. After the changes are propagated to your DB instance, all outbound network traffic requiring a DNS lookup queries your DNS server over port 53.

To set up a custom DNS server for your Amazon RDS for Oracle DB instance, do the following:

- From the DHCP options set attached to your virtual private cloud (VPC), set the `domain-name-servers` option to the IP address of your DNS name server.
- Ensure that your DNS server can resolve all lookup queries, including public DNS names, Amazon EC2 private DNS names, and customer-specific DNS names. If the outbound network traffic contains any DNS lookups

that your DNS server can't handle, your DNS server must have appropriate upstream DNS providers configured.

- Configure your DNS server to produce User Datagram Protocol (UDP) responses of 512 bytes or less.
- Configure your DNS server to produce Transmission Control Protocol (TCP) responses of 1024 bytes or less.
- Configure your DNS server to allow inbound traffic from your Amazon RDS DB instances over port 53. If your DNS server is in an Amazon VPC, the VPC must have a security group that contains inbound rules that permit UDP and TCP traffic on port 53. If your DNS server is not in an Amazon VPC, it must have appropriate firewall allow-listing to permit UDP and TCP inbound traffic on port 53.
- Configure the VPC of your Amazon RDS DB instance to allow outbound traffic over port

53. Your VPC must have a security group that contains outbound rules that allow UDP and TCP traffic on port 53.
- The routing path between the Amazon RDS DB instance and the DNS server has to be configured correctly to allow DNS traffic.
    - If the Amazon RDS DB instance and the DNS server are not in the same VPC, a peering connection has to be set up between them.

Setting and unsetting system diagnostic events

To set and unset diagnostic events at the session level, you can use the Oracle SQL statement `ALTER SESSION SET EVENTS`. However, to set events at the system level you can't use Oracle SQL. Instead, use the system event procedures in the `rdsadmin.rdsadmin_util` package. The system event procedures are available in the following engine versions:

- 19.0.0.0.ru-2020-10.rur-2020-10.r1 or higher 19c versions
- 18.0.0.0.ru-2020-10.rur-2020-10.r1 or higher 18c versions
- 12.2.0.1.ru-2020-10.rur-2020-10.r1 or higher 12.2.0.1 versions
- 12.1.0.2.V22 or higher 12.1 versions

Internally, the rdsadmin.rdsadmin_util package sets events by using the ALTER SYSTEM SET EVENTS statement. This ALTER SYSTEM statement isn't documented in the Oracle Database documentation. Some system diagnostic events can generate large amounts of tracing information, cause contention, or affect database availability. We recommend that you test specific diagnostic events in your nonproduction database, and only set events in your production database under guidance of Oracle Support.

Listing allowed system diagnostic events

To list the system events that you can set, use the Amazon RDS procedure rdsadmin.rdsadmin_util.list_allowed_system_events. This procedure accepts no parameters.

The following example lists all system events that you can set.

```
SET SERVEROUTPUT ON
EXEC rdsadmin.rdsadmin_util.list_allowed_system_events;
```

The following sample output lists event numbers and their descriptions. Use the Amazon RDS procedures set_system_event to set these events and unset_system_event to unset them.

```
604  - error occurred at recursive SQL level
942  - table or view does not exist
1401  - inserted value too large for column
1403  - no data found
1410  - invalid ROWID
```

1422 - exact fetch returns more than requested number of rows
1426 - numeric overflow
1427 - single-row subquery returns more than one row
1476 - divisor is equal to zero
1483 - invalid length for DATE or NUMBER bind variable
1489 - result of string concatenation is too long
1652 - unable to extend temp segment by in tablespace
1858 - a non-numeric character was found where a numeric was expected
4031 - unable to allocate bytes of shared memory ("","","","")
6502 - PL/SQL: numeric or value error
10027 - Specify Deadlock Trace Information to be Dumped
10046 - enable SQL statement timing
10053 - CBO Enable optimizer trace

10173 - Dynamic Sampling time-out error
10442 - enable trace of kst for ORA-01555 diagnostics
12008 - error in materialized view refresh path
12012 - error on auto execute of job
12504 - TNS:listener was not given the SERVICE_NAME in CONNECT_DATA
14400 - inserted partition key does not map to any partition
31693 - Table data object failed to load/unload and is being skipped due to error:

Setting system diagnostic events

To set a system event, use the Amazon RDS procedure rdsadmin.rdsadmin_util.set_system_event. You can only set events listed in the output of rdsadmin.rdsadmin_util.list_allowed_system_events. The set_system_event procedure accepts the following parameters.

Parameter name	Data type	Default	Required	Description

p_event	number	—	Yes	The system event number. The value must be one of the event numbers reported by list_allowed_system_events.
p_level	number	—	Yes	The event level. See the Oracle Database documentation or Oracle Support for descriptions of different level values.

The procedure set_system_event constructs and runs the required ALTER SYSTEM SET EVENTS statements according to the following principles:

- The event type (context or errorstack) is determined automatically.
- A statement in the form ALTER SYSTEM SET EVENTS '*event* LEVEL *event_level*' sets the context events. This notation is equivalent to ALTER SYSTEM SET EVENTS '*event* TRACE NAME CONTEXT FOREVER, LEVEL *event_level*'.
- A statement in the form ALTER SYSTEM SET EVENTS '*event* ERRORSTACK (*event_level*)' sets the error stack events. This notation is equivalent to ALTER SYSTEM SET

EVENTS '*event* TRACE NAME ERRORSTACK LEVEL *event_level*'.

The following example sets event 942 at level 3, and event 10442 at level 10. Sample output is included.

```
SQL> SET SERVEROUTPUT ON
SQL> EXEC rdsadmin.rdsadmin_util.set_system_event(942,3);
Setting system event 942 with: alter system set events '942 errorstack (3)'

PL/SQL procedure successfully completed.

SQL> EXEC rdsadmin.rdsadmin_util.set_system_event(10442,10);
Setting system event 10442 with: alter system set events '10442 level 10'

PL/SQL procedure successfully completed.
```

Listing system diagnostic events that are set

To list the system events that are currently set, use the Amazon RDS procedure rdsadmin.rdsadmin_util.list_set_system_events. This procedure reports only events set at system level by set_system_event.

The following example lists the active system events.

```
SET SERVEROUTPUT ON
EXEC rdsadmin.rdsadmin_util.list_set_system_events;
```

The following sample output shows the list of events, the event type, the level at which the events are currently set, and the time when the event was set.

```
942 errorstack (3) - set at 2020-11-03 11:42:27
10442 level 10 - set at 2020-11-03 11:42:41

PL/SQL procedure successfully completed.
```

Unsetting system diagnostic events

To unset a system event, use the Amazon RDS procedure rdsadmin.rdsadmin_util.unset_system_event. You can only unset events listed in the output of rdsadmin.rdsadmin_util.list_allowed_system_events. The unset_system_event procedure accepts the following parameter.

Parameter name	Data type	Default	Required	Description
p_event	number	—	Yes	The system event number. The value must be one of the event numbers reported by list_allowed_system_events.

The following example unsets events 942 and 10442. Sample output is included.

```
SQL> SET SERVEROUTPUT ON
SQL> EXEC rdsadmin.rdsadmin_util.unset_system_event(942);
Unsetting system event 942 with: alter system set events '942 off'
```

```
PL/SQL procedure successfully completed.

SQL> EXEC rdsadmin.rdsadmin_util.unset_sys-
tem_event(10442);
Unsetting system event 10442 with: alter system
set events '10442 off'

PL/SQL procedure successfully completed.
```

Performing common system tasks

for Oracle DB instances

Following, you can find how to perform certain common DBA tasks related to the system on your Amazon RDS DB instances running Oracle. To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances, and restricts access to certain system procedures and tables that require advanced privileges.

Disconnecting a session

To disconnect the current session by ending the dedicated server process, use the Amazon RDS procedure `rdsadmin.rdsadmin_util.disconnect`. The `disconnect` procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
sid	number	—	Yes	The session identifier.

serial	number	—	Yes	The serial number of the session.
method	varchar	'IMMEDIATE'	No	Valid values are 'IMMEDIATE' or 'POST_TRANSACTION'.

The following example disconnects a session.

```
begin
  rdsadmin.rdsadmin_util.disconnect(
    sid    => sid,
    serial => serial_number);
end;
/
```

To get the session identifier and the session serial number, query the V$SESSION view. The following example gets all sessions for the user AWSUSER.

```
select SID, SERIAL#, STATUS from V$SESSION where
USERNAME = 'AWSUSER';
```

The database must be open to use this method.

Terminating a session

To terminate a session, use the Amazon RDS procedure rdsadmin.rdsadmin_util.kill. The kill procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
sid	number	—	Yes	The session identifier.
serial	number	—	Yes	The serial number of the session.
method	varchar	null	No	Valid values are 'IMMEDIATE' or 'PROCESS'.

The following example terminates a session.

```
begin
  rdsadmin.rdsadmin_util.kill(
    sid   => sid,
```

```
    serial => serial_number);
end;
/
```

To get the session identifier and the session serial number, query the V$SESSION view. The following example gets all sessions for the user AWSUSER.

```
SELECT SID, SERIAL#, STATUS FROM V$SESSION
WHERE USERNAME = 'AWSUSER';
```

You can specify either IMMEDIATE or PROCESS as a value for the method parameter. By specifying PROCESS as the method value, you can terminate the processes associated with a session. Do this only if terminating the session using IMMEDIATE as the method value was unsuccessful.

Canceling a SQL statement in a session

To cancel a SQL statement in a session, use the Amazon RDS procedure `rdsadmin.rdsadmin_util.cancel`.

This procedure is supported for Oracle version 18.0.0.0 and later.

The `cancel` procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
sid	number	—	Yes	The session identifier.
serial	number	—	Yes	The serial number of the session.
sql_id	varchar2	null	No	The SQL identifier of the SQL statement.

The following example cancels a SQL statement in a session.

```
begin
  rdsadmin.rdsadmin_util.cancel(
    sid   => sid,
    serial => serial_number,
    sql_id => sql_id);
end;
/
```

To get the session identifier, the session serial number, and the SQL identifier of a SQL statement, query the V$SESSION view. The following example gets all sessions and SQL identifiers for the user AWSUSER.

```
select SID, SERIAL#, SQL_ID, STATUS from V$SESSION where USERNAME = 'AWSUSER';
```

Enabling and disabling restricted sessions

To enable and disable restricted sessions, use the Amazon RDS procedure rdsadmin.rdsadmin_u-

til.restricted_session. The restricted_session procedure has the following parameters.

Parameter name	Data type	Default	Yes	Description
p_enable	boolean	true	No	Set to true to enable restricted sessions, false to disable restricted sessions.

The following example shows how to enable and disable restricted sessions.

```
/* Verify that the database is currently unrestricted.
*/

SELECT LOGINS FROM V$INSTANCE;

LOGINS
-------
ALLOWED

/* Enable restricted sessions */
```

```
exec rdsadmin.rdsadmin_util.restricted_ses-
sion(p_enable => true);

/* Verify that the database is now restricted. */

SELECT LOGINS FROM V$INSTANCE;

LOGINS
----------
RESTRICTED

/* Disable restricted sessions */

exec rdsadmin.rdsadmin_util.restricted_ses-
sion(p_enable => false);

/* Verify that the database is now unrestricted again.
*/
```

```
SELECT LOGINS FROM V$INSTANCE;

LOGINS
-------
ALLOWED
```

Flushing the shared pool

To flush the shared pool, use the Amazon RDS procedure rdsadmin.rdsadmin_util.flush_shared_pool. The flush_shared_pool procedure has no parameters.

The following example flushes the shared pool.

```
exec rdsadmin.rdsadmin_util.flush_shared_pool;
```

Flushing the buffer cache

To flush the buffer cache, use the Amazon RDS procedure rdsadmin.rdsadmin_util.flush_buffer-_cache. The flush_buffer_cache procedure has no parameters.

The following example flushes the buffer cache.

```
exec rdsadmin.rdsadmin_util.flush_buffer_cache;
```

Granting SELECT or EXECUTE

privileges to SYS objects

Usually you transfer privileges by using roles, which can contain many objects. To grant privileges to a single object, use the Amazon RDS procedure rdsadmin.rdsadmin_util.grant_sys_object. The procedure grants only privileges that the master user has already been granted through a role or direct grant.

The grant_sys_object procedure has the following parameters.

For all parameter values, use uppercase unless you created the user with a case-sensitive identifier. For example, if you run CREATE USER myuser or CREATE USER MYUSER, the data dictionary stores MYUSER. However, if you use double quotes in CREATE USER "MyUser", the data dictionary stores MyUser.

Parameter name	Data type	Default	Required	Description
p_obj_name	varchar2	—	Yes	The name of the object to grant privileges for. The object can be a directory, function, package, procedure, sequence, table, or view. Object names must be spelled exactly as they appear in DBA_OBJECTS. Most system objects are defined in uppercase, so we recommend that you try that first.
p_grantee	varchar2	—	Yes	The name of the object to grant privileges to. The object can be a schema or a role.
p_privilege	varchar2	null	Yes	—
p_grant_option	boolean	false	No	Set to true to use the with grant option. The p_grant_option parameter is supported for 12.1.0.2.v4 and later, all 12.2.0.1 versions, all 18.0.0.0 versions, and all 19.0.0 versions.

The following example grants select privileges on an object named V_$SESSION to a user named USER1.

```
begin
  rdsadmin.rdsadmin_util.grant_sys_object(
    p_obj_name => 'V_$SESSION',
    p_grantee  => 'USER1',
    p_privilege => 'SELECT');
end;
/
```

The following example grants select privileges on an object named V_$SESSION to a user named USER1 with the grant option.

```
begin
  rdsadmin.rdsadmin_util.grant_sys_object(
    p_obj_name   => 'V_$SESSION',
    p_grantee    => 'USER1',
    p_privilege  => 'SELECT',
```

```
    p_grant_option => true);
end;
/
```

To be able to grant privileges on an object, your account must have those privileges granted to it directly with the grant option, or via a role granted using `with admin option`. In the most common case, you may want to grant `SELECT` on a DBA view that has been granted to the `SELECT_CATALOG_ROLE` role. If that role isn't already directly granted to your user using `with admin option`, then you can't transfer the privilege. If you have the DBA privilege, then you can grant the role directly to another user.

The following example grants the `SELECT_CATALOG_ROLE` and `EXECUTE_CATALOG_ROLE` to `USER1`. Since the `with admin option` is used, `USER1` can now grant access to SYS objects that have been granted to `SELECT_CATALOG_ROLE`.

```
GRANT SELECT_CATALOG_ROLE TO USER1 WITH
ADMIN OPTION;
GRANT EXECUTE_CATALOG_ROLE to USER1
WITH ADMIN OPTION;
```

Objects already granted to PUBLIC do not need to be re-granted. If you use the grant_sys_object procedure to re-grant access, the procedure call succeeds.

Revoking SELECT or EXECUTE

privileges on SYS objects

To revoke privileges on a single object, use the Amazon RDS procedure rdsadmin.rdsadmin_util.revoke_sys_object. The procedure only revokes privileges that the master account has already been granted through a role or direct grant.

The revoke_sys_object procedure has the following parameters.

Parameter name	Data type	Default	Required	Description

p_obj_name	varchar2	—	Yes	The name of the object to revoke privileges for. The object can be a directory, function, package, procedure, sequence, table, or view. Object names must be spelled exactly as they appear in DBA_OBJECTS. Most system objects are defined in upper case, so we recommend you try that first.
p_revokee	varchar2	—	Yes	The name of the object to revoke privileges for. The object can be a schema or a role.
p_privilege	varchar2	null	Yes	—

The following example revokes select privileges on an object named V_$SESSION from a user named USER1.

```
begin
  rdsadmin.rdsadmin_util.revoke_sys_object(
    p_obj_name  => 'V_$SESSION',
    p_revokee   => 'USER1',
    p_privilege => 'SELECT');
end;
```

```
/
```

Granting privileges to non-master users

You can grant select privileges for many objects in the SYS schema by using the SELECT_CATALOG_ROLE role. The SELECT_CATALOG_ROLE role gives users SELECT privileges on data dictionary views. The following example grants the role SELECT_CATALOG_ROLE to a user named user1.

```
GRANT SELECT_CATALOG_ROLE TO user1;
```

You can grant EXECUTE privileges for many objects in the SYS schema by using the EXECUTE_CATALOG_ROLE role. The EXECUTE_CATALOG_ROLE role gives users EXECUTE privileges for packages and procedures in the data dictionary. The following example grants the role EXECUTE_CATALOG_ROLE to a user named *user1*.

```
GRANT EXECUTE_CATALOG_ROLE TO user1;
```

The following example gets the permissions that the roles SELECT_CATALOG_ROLE and EXECUTE_CATALOG_ROLE allow.

```
SELECT *
 FROM ROLE_TAB_PRIVS
 WHERE ROLE IN ('SELECT_CATALOG_ROLE','EXECUTE_CATALOG_ROLE')
ORDER BY ROLE, TABLE_NAME ASC;
```

The following example creates a non-master user named user1, grants the CREATE SESSION privilege, and grants the SELECT privilege on a database named *sh.sales*.

```
CREATE USER user1 IDENTIFIED BY PASSWORD;
GRANT CREATE SESSION TO user1;
GRANT SELECT ON sh.sales TO user1;
```

Creating custom functions to verify passwords

You can create a custom password verification function in two ways. If you want to use standard verification logic, and to store your function in the SYS schema, use the create_verify_function procedure. If you want to use custom verification logic, or you don't want to store your function in the SYS schema, use the create_passthrough_verify_fcn procedure.

The create_verify_function procedure

The create_verify_function procedure is supported for Oracle version 12.1.0.2.v5 and later, all 12.2.0.1 versions, all 18.0.0.0 versions, and all 19.0.0 versions.

You can create a custom function to verify passwords by using the Amazon RDS procedure rd-

sadmin.rdsadmin_password_verify.create_verify_function. The create_verify_function procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
p_verify_function_name	varchar2	—	Yes	The name for your custom function. This function is created for you in the SYS schema. You assign this function to user profiles.
p_min_length	number	8	No	The minimum number of characters required.
p_max_length	number	256	No	The maximum number of characters allowed.
p_min_letters	number	1	No	The minimum number of letters required.
p_min_uppercase	number	0	No	The minimum number of uppercase letters required.
p_min_lowercase	number	0	No	The minimum number of lowercase letters required.
p_min_digits	number	1	No	The minimum number of digits required.

p_min_special	number	0	No	The minimum number of special characters required.
p_min_different_chars	number	3	No	The minimum number of different characters required between the old and new password.
p_disallow_username	boolean	true	No	Set to true to disallow the user name in the password.
p_disallow_reverse	boolean	true	No	Set to true to disallow the reverse of the user name in the password.
p_disallow_db_name	boolean	true	No	Set to true to disallow the database or server name in the password.
p_disallow_simple_strings	boolean	true	No	Set to true to disallow simple strings as the password.
p_disallow_whitespace	boolean	false	No	Set to true to disallow white space characters in the password.
p_disallow_at_sign	boolean	false	No	Set to true to disallow the @ character in the password.

You can create multiple password verification functions.

There are restrictions on the name of your custom function. Your custom function can't have the same name as an existing system object. The name can be no more than 30 characters long. Also, the name must include one of the following strings: PASSWORD, VERIFY, COMPLEXITY, ENFORCE, or STRENGTH.

The following example creates a function named CUSTOM_PASSWORD_FUNCTION. The function requires that a password has at least 12 characters, 2 uppercase characters, 1 digit, and 1 special character, and that the password disallows the @ character.

```
begin
  rdsadmin.rdsadmin_password_verify.create_verify_function(
    p_verify_function_name => 'CUSTOM_PASSWORD_FUNCTION',
    p_min_length       => 12,
```

```
    p_min_uppercase     => 2,
    p_min_digits        => 1,
    p_min_special       => 1,
    p_disallow_at_sign  => true);
end;
/
```

To see the text of your verification function, query DBA_SOURCE. The following example gets the text of a custom password function named CUSTOM_PASSWORD_FUNCTION.

```
COL TEXT FORMAT a150

 SELECT TEXT
  FROM DBA_SOURCE
 WHERE OWNER = 'SYS'
   AND NAME = 'CUSTOM_PASSWORD_FUNCTION'
ORDER BY LINE;
```

To associate your verification function with a user profile, use alter profile. The following example as-

sociates a verification function with the DEFAULT-user profile.

```
ALTER PROFILE DEFAULT LIMIT PASSWORD_VERIFY_FUNCTION CUSTOM_PASSWORD_FUNCTION;
```

To see what user profiles are associated with what verification functions, query DBA_PROFILES.
The following example gets the profiles that are associated with the custom verification function named CUSTOM_PASSWORD_FUNCTION.

```
SELECT * FROM DBA_PROFILES WHERE RESOURCE_NAME = 'PASSWORD' AND LIMIT = 'CUSTOM_PASSWORD_FUNCTION';

PROFILE                        RESOURCE_NAME                  RESOURCE  LIMIT
------------------------------ ------------------------------ --------  ----------------------
DEFAULT                        PASSWORD_VERIFY_FUNCTION       PASSWORD  CUSTOM_PASSWORD_FUNCTION
```

The following example gets all profiles and the password verification functions that they are associated with.

```
SELECT * FROM DBA_PROFILES WHERE RE-
SOURCE_NAME = 'PASSWORD_VERIFY_FUNC-
TION';

PROFILE                RESOURCE_NAME                RE-
SOURCE  LIMIT
---------------------- ------------------------------
-------- ----------------------
DEFAULT                PASSWORD_VERIFY_FUNCTION
PASSWORD  CUSTOM_PASSWORD_FUNCTION
RDSADMIN               PASSWORD_VERIFY_FUNC-
TION       PASSWORD  NULL
```

The create_passthrough_verify_fcn procedure

The create_passthrough_verify_fcn procedure is supported for Oracle version 12.1.0.2.v7 and later,

all 12.2.0.1 versions, all 18.0.0.0 versions, and all 19.0.0 versions.

You can create a custom function to verify passwords by using the Amazon RDS procedure rdsadmin.rdsadmin_password_verify.create_passthrough_verify_fcn. The create_passthrough_verify_fcn procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
p_verify_function_name	varchar2	—	Yes	The name for your custom verification function. This is a wrapper function that is created for you in the SYS schema, and it doesn't contain any verification logic. You assign this function to user profiles.
p_target_owner	varchar2	—	Yes	The schema owner for your custom verification function.
p_target_function_name	varchar2	—	Yes	The name of your existing custom function that contains the verification logic. Your custom function must return a bool-

				ean. Your function should return true if the password is valid and false if the password is invalid.

The following example creates a password verification function that uses the logic from the function named PASSWORD_LOGIC_EXTRA_STRONG.

```
begin
  rdsadmin.rdsadmin_password_verify.create_
passthrough_verify_fcn(
    p_verify_function_name => 'CUSTOM_PASS-
WORD_FUNCTION',
    p_target_owner       => 'TEST_USER',
    p_target_function_name => 'PASSWORD_LOG-
IC_EXTRA_STRONG');
end;
/
```

To associate the verification function with a user profile, use alter profile. The following example

associates the verification function with the DEFAULT user profile.

```
ALTER PROFILE DEFAULT LIMIT PASSWORD_VERIFY_FUNCTION CUSTOM_PASSWORD_FUNCTION;
```

Setting up a custom DNS server

Amazon RDS supports outbound network access on your DB instances running Oracle. Amazon RDS Oracle allows Domain Name Service (DNS) resolution from a custom DNS server owned by the customer. You can resolve only fully qualified domain names from your Amazon RDS DB instance through your custom DNS server.

After you set up your custom DNS name server, it takes up to 30 minutes to propagate the changes to your DB instance. After the changes are propagated to your DB instance, all outbound network traffic

requiring a DNS lookup queries your DNS server over port 53.

To set up a custom DNS server for your Amazon RDS for Oracle DB instance, do the following:

- From the DHCP options set attached to your virtual private cloud (VPC), set the `domain-name-servers` option to the IP address of your DNS name server.
- Ensure that your DNS server can resolve all lookup queries, including public DNS names, Amazon EC2 private DNS names, and customer-specific DNS names. If the outbound network traffic contains any DNS lookups that your DNS server can't handle, your DNS server must have appropriate upstream DNS providers configured.
- Configure your DNS server to produce User Datagram Protocol (UDP) responses of 512 bytes or less.

- Configure your DNS server to produce Transmission Control Protocol (TCP) responses of 1024 bytes or less.
- Configure your DNS server to allow inbound traffic from your Amazon RDS DB instances over port 53. If your DNS server is in an Amazon VPC, the VPC must have a security group that contains inbound rules that permit UDP and TCP traffic on port 53. If your DNS server is not in an Amazon VPC, it must have appropriate firewall allow-listing to permit UDP and TCP inbound traffic on port 53.
- Configure the VPC of your Amazon RDS DB instance to allow outbound traffic over port 53. Your VPC must have a security group that contains outbound rules that allow UDP and TCP traffic on port 53.
- The routing path between the Amazon RDS DB instance and the DNS server has to be configured correctly to allow DNS traffic.

    - If the Amazon RDS DB instance and the DNS server are not in the same VPC, a peering connection has to be set up between them.

Setting and unsetting system diagnostic events

To set and unset diagnostic events at the session level, you can use the Oracle SQL statement `ALTER SESSION SET EVENTS`. However, to set events at the system level you can't use Oracle SQL. Instead, use the system event procedures in the `rdsadmin.rdsadmin_util` package. The system event procedures are available in the following engine versions:

- 19.0.0.0.ru-2020-10.rur-2020-10.r1 or higher 19c versions
- 18.0.0.0.ru-2020-10.rur-2020-10.r1 or higher 18c versions

- 12.2.0.1.ru-2020-10.rur-2020-10.r1 or higher 12.2.0.1 versions
- 12.1.0.2.V22 or higher 12.1 versions

Internally, the rdsadmin.rdsadmin_util package sets events by using the ALTER SYSTEM SET EVENTS statement. This ALTER SYSTEM statement isn't documented in the Oracle Database documentation. Some system diagnostic events can generate large amounts of tracing information, cause contention, or affect database availability. We recommend that you test specific diagnostic events in your nonproduction database, and only set events in your production database under guidance of Oracle Support.

Listing allowed system diagnostic events

To list the system events that you can set, use the Amazon RDS procedure rdsadmin.rdsadmin_util.list_allowed_system_events. This procedure accepts no parameters.

The following example lists all system events that you can set.

```
SET SERVEROUTPUT ON
EXEC rdsadmin.rdsadmin_util.list_allowed_system_events;
```

The following sample output lists event numbers and their descriptions. Use the Amazon RDS procedures `set_system_event` to set these events and `unset_system_event` to unset them.

```
604  - error occurred at recursive SQL level
942  - table or view does not exist
1401  - inserted value too large for column
1403  - no data found
1410  - invalid ROWID
1422  - exact fetch returns more than requested number of rows
1426  - numeric overflow
```

1427 - single-row subquery returns more than one row

1476 - divisor is equal to zero

1483 - invalid length for DATE or NUMBER bind variable

1489 - result of string concatenation is too long

1652 - unable to extend temp segment by in tablespace

1858 - a non-numeric character was found where a numeric was expected

4031 - unable to allocate bytes of shared memory ("","","","")

6502 - PL/SQL: numeric or value error

10027 - Specify Deadlock Trace Information to be Dumped

10046 - enable SQL statement timing

10053 - CBO Enable optimizer trace

10173 - Dynamic Sampling time-out error

10442 - enable trace of kst for ORA-01555 diagnostics

12008 - error in materialized view refresh path
12012 - error on auto execute of job
12504 - TNS:listener was not given the SERVICE_NAME in CONNECT_DATA
14400 - inserted partition key does not map to any partition
31693 - Table data object failed to load/unload and is being skipped due to error:

Setting system diagnostic events

To set a system event, use the Amazon RDS procedure rdsadmin.rdsadmin_util.set_system_event. You can only set events listed in the output of rdsadmin.rdsadmin_util.list_allowed_system_events. The set_system_event procedure accepts the following parameters.

Parameter name	Data type	Default	Required	Description
p_event	number	—	Yes	The system event number. The value must be one of the event numbers reported by list_allowed_system_events.
p_level	number	—	Yes	The event level. See the Oracle Database documentation or Oracle Support for descriptions of different level values.

The procedure set_system_event constructs and runs the required ALTER SYSTEM SET EVENTS statements according to the following principles:

- The event type (context or errorstack) is determined automatically.
- A statement in the form ALTER SYSTEM SET EVENTS '*event* LEVEL *event_level*' sets the context events. This notation is equivalent to ALTER SYSTEM SET EVENTS '*event* TRACE NAME CONTEXT FOREVER, LEVEL *event_level*'.
- A statement in the form ALTER SYSTEM SET EVENTS '*event* ERRORSTACK (*event_level*)' sets the error stack events. This notation is equivalent to ALTER SYSTEM SET EVENTS '*event* TRACE NAME ERRORSTACK LEVEL *event_level*'.

The following example sets event 942 at level 3, and event 10442 at level 10. Sample output is included.

```
SQL> SET SERVEROUTPUT ON
SQL> EXEC rdsadmin.rdsadmin_util.set_system_event(942,3);
Setting system event 942 with: alter system set events '942 errorstack (3)'

PL/SQL procedure successfully completed.

SQL> EXEC rdsadmin.rdsadmin_util.set_system_event(10442,10);
Setting system event 10442 with: alter system set events '10442 level 10'

PL/SQL procedure successfully completed.
```

Listing system diagnostic events that are set

To list the system events that are currently set, use the Amazon RDS procedure rdsadmin.rdsadmin_util.list_set_system_events. This procedure reports only events set at system level by set_system_event.

The following example lists the active system events.

```
SET SERVEROUTPUT ON
EXEC rdsadmin.rdsadmin_util.list_set_system_events;
```

The following sample output shows the list of events, the event type, the level at which the events are currently set, and the time when the event was set.

```
942 errorstack (3) - set at 2020-11-03 11:42:27
10442 level 10 - set at 2020-11-03 11:42:41

PL/SQL procedure successfully completed.
```

Unsetting system diagnostic events

To unset a system event, use the Amazon RDS procedure rdsadmin.rdsadmin_util.unset_system_event. You can only unset events listed in the output of rdsadmin.rdsadmin_util.list_allowed_system_events. The unset_system_event procedure accepts the following parameter.

Parameter name	Data type	Default	Required	Description
p_event	number	—	Yes	The system event number. The value must be one of the event numbers reported by list_allowed_system_events.

The following example unsets events 942 and 10442. Sample output is included.

```
SQL> SET SERVEROUTPUT ON
SQL> EXEC rdsadmin.rdsadmin_util.unset_system_event(942);
Unsetting system event 942 with: alter system set events '942 off'
```

```
PL/SQL procedure successfully completed.

SQL> EXEC rdsadmin.rdsadmin_util.unset_sys-
tem_event(10442);
Unsetting system event 10442 with: alter system
set events '10442 off'

PL/SQL procedure successfully completed.
```

Changing the global name of a database

To change the global name of a database, use the Amazon RDS procedure rdsadmin.rdsadmin_util.rename_global_name. The rename_global_name procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
p_new_global_name	varchar2	—	Yes	The new global name for the database.

The database must be open for the name change to occur.

The following example changes the global name of a database to new_global_name.

```
exec rdsadmin.rdsadmin_util.rename_global_
name(p_new_global_name => 'new_global_name');
```

Creating and sizing tablespaces

Amazon RDS only supports Oracle Managed Files (OMF) for data files, log files, and control files. When you create data files and log files, you can't specify the physical file names.

By default, tablespaces are created with auto-extend enabled, and no maximum size. Because of these default settings, tablespaces can grow to consume all allocated storage. We recommend that you specify an appropriate maximum size on permanent and temporary tablespaces, and that you carefully monitor space usage.

The following example creates a tablespace named `users2` with a starting size of 1 gigabyte and a maximum size of 10 gigabytes:

```
CREATE TABLESPACE users2 DATAFILE SIZE 1G AUTOEXTEND ON MAXSIZE 10G;
```

The following example creates temporary tablespace named temp01:

```
CREATE TEMPORARY TABLESPACE temp01;
```

We recommend that you don't use smallfile tablespaces because you can't resize smallfile tablespaces with Amazon RDS for Oracle. However, you can add a datafile to a smallfile tablespace.

You can resize a bigfile tablespace by using ALTER TABLESPACE. You can specify the size in kilobytes (K), megabytes (M), gigabytes (G), or terabytes (T).

The following example resizes a bigfile tablespace named users2 to 200 MB.

```
ALTER TABLESPACE users2 RESIZE 200M;
```

The following example adds an additional datafile to a smallfile tablespace named **users2**.

```
ALTER TABLESPACE users2 ADD DATAFILE SIZE 100000M AUTOEXTEND ON NEXT 250m MAXSIZE UNLIMITED;
```

Setting the default tablespace

To set the default tablespace, use the Amazon RDS procedure rdsadmin.rdsadmin_util.alter_default_tablespace. The alter_default_tablespace procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
tablespace_name	varchar	—	Yes	The name of the default tablespace.

The following example sets the default tablespace to *users2*:

```
EXEC rdsadmin.rdsadmin_util.alter_default_tablespace(tablespace_name => 'users2');
```

Setting the default temporary tablespace

To set the default temporary tablespace, use the Amazon RDS procedure rdsadmin.rdsadmin_util.alter_default_temp_tablespace. The alter_default_temp_tablespace procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
tablespace_name	varchar	—	Yes	The name of the default temporary tablespace.

The following example sets the default temporary tablespace to *temp01*.

```
EXEC rdsadmin.rdsadmin_util.alter_default_temp_tablespace(tablespace_name => 'temp01');
```

Checkpointing a database

To checkpoint the database, use the Amazon RDS procedure rdsadmin.rdsadmin_util.checkpoint. The checkpoint procedure has no parameters.

The following example checkpoints the database.

```
EXEC rdsadmin.rdsadmin_util.checkpoint;
```

Setting distributed recovery

To set distributed recovery, use the Amazon RDS procedures rdsadmin.rdsadmin_util.enable_distr_recovery and disable_distr_recovery. The procedures have no parameters.

The following example enables distributed recovery.

```
EXEC rdsadmin.rdsadmin_util.enable_distr_recovery;
```

The following example disables distributed recovery.

```
EXEC rdsadmin.rdsadmin_util.disable_distr_recovery;
```

Setting the database time zone

You can set the time zone of your Amazon RDS Oracle database in the following ways:

- The Timezone option

The Timezone option changes the time zone at the host level and affects all date columns and values such as SYSDATE.

- The Amazon RDS procedure rdsadmin.rdsadmin_util.alter_db_time_zone

The alter_db_time_zone procedure changes the time zone for only certain data types, and doesn't change SYSDATE.

The alter_db_time_zone procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
p_new_tz	varchar2	—	Yes	The new time zone as a named region or an absolute offset from Coordinated Universal Time (UTC). Valid offsets range from -12:00 to +14:00.

The following example changes the time zone to UTC plus three hours.

```
EXEC rdsadmin.rdsadmin_util.alter_db_time_
zone(p_new_tz => '+3:00');
```

The following example changes the time zone to the Africa/Algiers time zone.

```
EXEC rdsadmin.rdsadmin_util.alter_db_time_
zone(p_new_tz => 'Africa/Algiers');
```

After you alter the time zone by using the `alter_db_time_zone` procedure, reboot your DB instance for the change to take effect.

Working with Oracle external tables

Oracle external tables are tables with data that is not in the database. Instead, the data is in external files that the database can access. By using external

tables, you can access data without loading it into the database.

With Amazon RDS, you can store external table files in directory objects. You can create a directory object, or you can use one that is predefined in the Oracle database, such as the DATA_PUMP_DIR directory. You can query the ALL_DIRECTORIES view to list the directory objects for your Amazon RDS Oracle DB instance.

Directory objects point to the main data storage space (Amazon EBS volume) used by your instance. The space used—along with data files, redo logs, audit, trace, and other files—counts against allocated storage.

You can move an external data file from one Oracle database to another by using the DBMS_FILE_TRANSFER package or the UTL_FILE package. The external data file is moved from a directory on the

source database to the specified directory on the destination database.

After you move the external data file, you can create an external table with it. The following example creates an external table that uses the emp_xt_file1.txt file in the USER_DIR1 directory.

```
CREATE TABLE emp_xt (
 emp_id    NUMBER,
 first_name  VARCHAR2(50),
 last_name  VARCHAR2(50),
 user_name  VARCHAR2(20)
)
ORGANIZATION EXTERNAL (
 TYPE ORACLE_LOADER
 DEFAULT DIRECTORY USER_DIR1
 ACCESS PARAMETERS (
  RECORDS DELIMITED BY NEWLINE
  FIELDS TERMINATED BY ','
  MISSING FIELD VALUES ARE NULL
```

```
  (emp_id,first_name,last_name,user_name)
 )
 LOCATION ('emp_xt_file1.txt')
)
PARALLEL
REJECT LIMIT UNLIMITED;
```

Suppose that you want to move data that is in an Amazon RDS Oracle DB instance into an external data file. In this case, you can populate the external data file by creating an external table and selecting the data from the table in the database. For example, the following SQL statement creates the orders_xt external table by querying the orders table in the database.

```
CREATE TABLE orders_xt
 ORGANIZATION EXTERNAL
 (
  TYPE ORACLE_DATAPUMP
  DEFAULT DIRECTORY DATA_PUMP_DIR
```

```
  LOCATION ('orders_xt.dmp')
)
AS SELECT * FROM orders;
```

In this example, the data is populated in the orders_xt.dmp file in the DATA_PUMP_DIR directory.

Generating performance reports with

Automatic Workload Repository (AWR)

To gather performance data and generate reports, Oracle recommends Automatic Workload Repository (AWR). AWR requires Oracle Database Enterprise Edition and a license for the Diagnostics and Tuning packs. To enable AWR, set the CONTROL_MANAGEMENT_PACK_ACCESS initialization parameter to either DIAGNOSTIC or DIAGNOSTIC+TUNING.

Working with AWR reports in RDS

To generate AWR reports, you can run scripts such as awrrpt.sql. These scripts are installed on the database host server. In Amazon RDS, you don't have direct access to the host. However, you can get copies of SQL scripts from another installation of Oracle Database.

You can also use AWR by running procedures in the SYS.DBMS_WORKLOAD_REPOSITORY PL/SQL package. You can use this package to manage baselines and snapshots, and also to display ASH and AWR reports. For example, to generate an AWR report in text format run the DBMS_WORKLOAD-_REPOSITORY.AWR_REPORT_TEXT procedure. However, you can't reach these AWR reports from the AWS Management Console.

When working with AWR, we recommend using the rdsadmin.rdsadmin_diagnostic_util procedures. You can use these procedures to generate the following:

- AWR reports
- Active Session History (ASH) reports
- Automatic Database Diagnostic Monitor (ADDM) reports
- Oracle Data Pump Export dump files of AWR data

The rdsadmin_diagnostic_util procedures save the reports to the DB instance file system. You can access these reports from the console. You can also access reports using the rdsadmin.rds_file_util procedures, and you can access reports that are copied to Amazon S3 using the S3 Integration option.

You can use the rdsadmin_diagnostic_util procedures in the following Amazon RDS for Oracle DB engine versions:

- 12.1.0.2.v20 or higher 12.1 versions
- 12.2.0.1.ru-2020-04.rur-2020-04.r1 or higher 12.2 versions

- 18.0.0.0.ru-2020-04.rur-2020-04.r1 or higher 18c versions
- 19.0.0.0.ru-2020-04.rur-2020-04.r1 or higher 19c versions

Common parameters for the diagnostic utility package

You typically use the following parameters when managing AWR and ADDM with the rdsadmin_diagnostic_util package.

Parameter	Data type	Default	Required	Description
begin_snap_id	NUMBER	—	Yes	The ID of the beginning snapshot.
end_snap_id	NUMBER	—	Yes	The ID of the ending snapshot.
dump_directory	VARCHAR2	BDUMP	No	The directory to write the

				report or export file to. If you specify a nondefault directory, the user that runs the rdsadmin_diagnostic_util procedures must have write permissions for the directory.
report_type	VARCHAR2	HTML	No	The format of the report. Valid values are TEXT and HTML.
dbid	NUMBER	—	No	A valid database

				identifier (DBID) shown in the D-BA_HIST_ DATABASE _IN-STANCE vi ew for Oracle. If this pa-rameter is not speci-fied, RDS uses the current DBID, which is shown in the V $DATA-BASE.D-BID view.

You typically use the following parameters when managing ASH with the rdsadmin_diagnostic_util package.

Parameter	Data type	Default	Required	Description
begin_time	DATE	—	Yes	The beginning time of the ASH analysis.
end_time	DATE	—	Yes	The ending time of the ASH analysis.
slot_width	NUMBER	0	No	The duration of the slots (in seconds) used in the "Top Activity" section of the ASH report. If this parameter isn't specified, the time interval between begin_time and end_time uses no more than 10 slots.
sid	NUMBER	Null	No	The session ID.
sql_id	VARCHAR2	Null	No	The SQL ID.
wait_class	VARCHAR2	Null	No	The wait class name.
service_hash	NUMBER	Null	No	The service name hash.
module_name	VARCHAR2	Null	No	The module name.
action_name	VARCHAR2	Null	No	The action name.
client_id	VARCHAR2	Null	No	The application-specific ID of the database session.
plsql_entry	VARCHAR2	Null	No	The PL/SQL entry point.

Generating an AWR report

To generate an AWR report, use the `rdsadmin.rdsadmin_diagnostic_util.awr_report` procedure.

The following example generates a AWR report for the snapshot range 101–106. The output text file is named `awrrpt_101_106.txt`. You can access this report from the AWS Management Console.

```
exec rdsadmin.rdsadmin_diagnostic_util.awr_report(101,106,'TEXT');
```

The following example generates an HTML report for the snapshot range 63–65. The output HTML file is named awrrpt_63_65.html. The procedure writes the report to the nondefault database directory named AWR_RPT_DUMP.

```
exec rdsadmin.rdsadmin_diagnostic_util.awr_report(63,65,'HTML','AWR_RPT_DUMP');
```

Extracting AWR data into a dump file

To extract AWR data into a dump file, use the rdsadmin.rdsadmin_diagnostic_util.awr_extract procedure.

The following example extracts the snapshot range 101–106. The output dump file is named awrextract_101_106.dmp. You can access this file through the console.

```
exec rdsadmin.rdsadmin_diagnostic_util.awr_extract(101,106);
```

The following example extracts the snapshot range 63–65. The output dump file is named awrextract_63_65.dmp. The file is stored in the nondefault database directory named AWR_RPT_DUMP.

```
exec rdsadmin.rdsadmin_diagnostic_util.awr_extract(63,65,'AWR_RPT_DUMP');
```

Generating an ADDM report

To generate an ADDM report, use the rdsadmin.rdsadmin_diagnostic_util.addm_report procedure.

The following example generates an ADDM report for the snapshot range 101–106. The output text file is named addmrpt_101_106.txt. You can access the report through the console.

```
exec rdsadmin.rdsadmin_diagnostic_util.addm_report(101,106);
```

The following example generates an ADDM report for the snapshot range 63–65. The output text file is named addmrpt_63_65.txt. The file is stored in the nondefault database directory named ADDM_RPT_DUMP.

```
exec rdsadmin.rdsadmin_diagnostic_util.addm_report(63,65,'ADDM_RPT_DUMP');
```

Generating an ASH report

To generate an ASH report, use the rdsadmin.rdsadmin_diagnostic_util.ash_report procedure.

The following example generates an ASH report that includes the data from 14 minutes ago until the current time. The name of the output file uses the format ashrpt*begin_time*_*end_time*.txt, where *begin_time* and *end_time* use the format YYYYMMDDHH24MISS. You can access the file through the console.

```
BEGIN
  rdsadmin.rdsadmin_diagnostic_util.ash_report(
    begin_time   =>   SYSDATE-14/1440,
    end_time     =>   SYSDATE,
    report_type  =>   'TEXT');
END;
/
```

The following example generates an ASH report that includes the data from November 18, 2019, at 6:07 PM through November 18, 2019, at 6:15 PM. The name of the output HTML report is ashrpt_20190918180700_20190918181500.html. The report is stored in the nondefault database directory named AWR_RPT_DUMP.

```
BEGIN
  rdsadmin.rdsadmin_diagnostic_util.ash_report(
    begin_time   =>   TO_DATE('2019-09-18
18:07:00', 'YYYY-MM-DD HH24:MI:SS'),
```

```
    end_time     =>  TO_DATE('2019-09-18
18:15:00', 'YYYY-MM-DD HH24:MI:SS'),
    report_type  =>  'html',
    dump_directory =>  'AWR_RPT_DUMP');
END;
/
```

Accessing AWR reports from the console or CLI

To access AWR reports or export dump files, you can use the AWS Management Console or AWS CLI.

Adjusting database links for use

with DB instances in a VPC

To use Oracle database links with Amazon RDS DB instances inside the same virtual private cloud (VPC) or peered VPCs, the two DB instances should have a valid route between them. Verify the valid route between the DB instances by using your

VPC routing tables and network access control list (ACL).

The security group of each DB instance must allow ingress to and egress from the other DB instance. The inbound and outbound rules can refer to security groups from the same VPC or a peered VPC.

If you have configured a custom DNS server using the DHCP Option Sets in your VPC, your custom DNS server must be able to resolve the name of the database link target.

Setting the default edition for a DB instance

You can redefine database objects in a private environment called an edition. You can use edition-based redefinition to upgrade an application's database objects with minimal downtime.

You can set the default edition of an Amazon RDS Oracle DB instance using the Amazon RDS procedure rdsadmin.rdsadmin_util.alter_default_edition.

The following example sets the default edition for the Amazon RDS Oracle DB instance to RELEASE_V1.

```
exec rdsadmin.rdsadmin_util.alter_default_edition('RELEASE_V1');
```

The following example sets the default edition for the Amazon RDS Oracle DB instance back to the Oracle default.

```
exec rdsadmin.rdsadmin_util.alter_default_edition('ORA$BASE');
```

Enabling auditing for the SYS.AUD$ table

To enable auditing on the database audit trail table SYS.AUD$, use the Amazon RDS procedure rdsadmin.rdsadmin_master_util.audit_all_sys_aud_table. The only supported audit property is ALL. You can't audit or not audit individual statements or operations.

Enabling auditing is supported for Oracle DB instances running the following versions:

- 12.1.0.2.v14 and later 12.1 versions
- All 12.2.0.1 versions
- All 18.0.0.0 versions
- All 19.0.0.0 versions

The audit_all_sys_aud_table procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
p_by_access	boolean	true	No	Set to true to

				audit BY ACCESS. Set to false to audit BY SESSION.

The following query returns the current audit configuration for SYS.AUD$ for a database.

```
SELECT * FROM DBA_OBJ_AUDIT_OPTS WHERE
OWNER='SYS' AND OBJECT_NAME='AUD$';
```

The following commands enable audit of ALL on SYS.AUD$ BY ACCESS.

```
exec rdsadmin.rdsadmin_master_util.audit_al-
l_sys_aud_table;

exec rdsadmin.rdsadmin_master_util.audit_al-
l_sys_aud_table(p_by_access => true);
```

The following command enables audit of ALL on SYS.AUD$ BY SESSION.

```
exec rdsadmin.rdsadmin_master_util.audit_al-
l_sys_aud_table(p_by_access => false);
```

Disabling auditing for the SYS.AUD$ table

To disable auditing on the database audit trail table SYS.AUD$, use the Amazon RDS procedure rdsadmin.rdsadmin_master_util.noaudit_all_sys_aud_table. This procedure takes no parameters.

The following query returns the current audit configuration for SYS.AUD$ for a database:

```
SELECT * FROM DBA_OBJ_AUDIT_OPTS WHERE
OWNER='SYS' AND OBJECT_NAME='AUD$';
```

The following command disables audit of ALL on SYS.AUD$.

```
exec rdsadmin.rdsadmin_master_util.noaudit_al-
l_sys_aud_table;
```

Cleaning up interrupted online index builds

To clean up failed online index builds, use the Amazon RDS procedure `rdsadmin.rdsadmin_dbms_repair.online_index_clean`.

The `online_index_clean` procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
object_id	binary_integer	ALL_INDEX_ID	No	The object ID of the index. Typically, you can use the object ID from the ORA-08104 error text.
wait_for_lock	binary_integer	rdsadmin.rdsadmin_dbms_repair.lock_wait	No	Specify rdsadmin.rdsadmin_dbms_repair.lock_wait, the default, to try to get a lock on the underlying object and retry until an internal limit is reached if the lock fails. Specify rdsadmin.rdsadmin_dbms_repair.lock_nowait to try to get a lock on the underlying object but not retry if the lock fails.

The following example cleans up a failed online index build:

```
declare
 is_clean boolean;
begin
```

```
 is_clean := rdsadmin.rdsadmin_dbms_repair.on-
line_index_clean(
  object_id    => 1234567890,
  wait_for_lock => rdsadmin.rdsadmin_dbms_re-
pair.lock_nowait
 );
end;
/
```

Skipping corrupt blocks

To skip corrupt blocks during index and table scans, use the rdsadmin.rdsadmin_dbms_repair package.

The following procedures wrap the functionality of the sys.dbms_repair.admin_table procedure and take no parameters:

- rdsadmin.rdsadmin_dbms_repair.create_repair_table

- rdsadmin.rdsadmin_dbms_repair.create_orphan_keys_table
- rdsadmin.rdsadmin_dbms_repair.drop_repair_table
- rdsadmin.rdsadmin_dbms_repair.drop_orphan_keys_table
- rdsadmin.rdsadmin_dbms_repair.purge_repair_table
- rdsadmin.rdsadmin_dbms_repair.purge_orphan_keys_table

The following procedures take the same parameters as their counterparts in the DBMS_REPAIR package for Oracle databases:

- rdsadmin.rdsadmin_dbms_repair.check_object
- rdsadmin.rdsadmin_dbms_repair.dump_orphan_keys
- rdsadmin.rdsadmin_dbms_repair.fix_corrupt_blocks

- rdsadmin.rdsadmin_dbms_repair.rebuild_freelists
- rdsadmin.rdsadmin_dbms_repair.segment_fix_status
- rdsadmin.rdsadmin_dbms_repair.skip_corrupt_blocks

Example Responding to corrupt blocks

This example shows the basic workflow for responding to corrupt blocks. Your steps will depend on the location and nature of your block corruption.

To skip corrupt blocks during index and table scans

1. Run the following procedures to create repair tables if they don't already exist.

```
exec rdsadmin.rdsadmin_dbms_repair.create_repair_table;
```

```
exec rdsadmin.rdsadmin_dbms_repair.create_or-
phan_keys_table;
```

2. Run the following procedures to check for existing records and purge them if appropriate.

```
SELECT COUNT(*) FROM SYS.REPAIR_TABLE;
SELECT COUNT(*) FROM SYS.OR-
PHAN_KEY_TABLE;
SELECT COUNT(*) FROM SYS.DBA_REPAIR_TABLE;
SELECT COUNT(*) FROM SYS.DBA_OR-
PHAN_KEY_TABLE;

exec rdsadmin.rdsadmin_dbms_repair.purge_re-
pair_table;
exec rdsadmin.rdsadmin_dbms_repair.purge_or-
phan_keys_table;
```

3. Run the following procedure to check for corrupt blocks.

```
SET SERVEROUTPUT ON
DECLARE v_num_corrupt INT;
BEGIN
 v_num_corrupt := 0;
```

```
 rdsadmin.rdsadmin_dbms_repair.check_object (
  schema_name => '&corruptionOwner',
  object_name => '&corruptionTable',
  corrupt_count =>  v_num_corrupt
 );
 dbms_output.put_line('number corrupt: '||
to_char(v_num_corrupt));
END;
/

COL CORRUPT_DESCRIPTION FORMAT a30
COL REPAIR_DESCRIPTION FORMAT a30

SELECT OBJECT_NAME, BLOCK_ID, CORRUPT_
TYPE, MARKED_CORRUPT,
    CORRUPT_DESCRIPTION, REPAIR_DESCRIP-
TION
FROM  SYS.REPAIR_TABLE;

SELECT SKIP_CORRUPT
FROM  DBA_TABLES
WHERE  OWNER = '&corruptionOwner'
```

```
AND   TABLE_NAME = '&corruptionTable';
```

4. Use the skip_corrupt_blocks procedure to enable or disable corruption skipping for affected tables. Depending on the situation, you may also need to extract data to a new table, and then drop the table containing the corrupt block.

Run the following procedure to enable corruption skipping for affected tables.

```
begin
 rdsadmin.rdsadmin_dbms_repair.skip_corrup-
t_blocks (
  schema_name => '&corruptionOwner',
  object_name => '&corruptionTable',
  object_type => rdsadmin.rdsadmin_dbms_re-
pair.table_object,
  flags => rdsadmin.rdsadmin_dbms_re-
pair.skip_flag);
end;
/
```

```
select skip_corrupt from dba_tables where owner
= '&corruptionOwner' and table_name = '&corrup-
tionTable';
```

Run the following procedure to disable corruption skipping.

```
begin
 rdsadmin.rdsadmin_dbms_repair.skip_corrup-
t_blocks (
  schema_name => '&corruptionOwner',
  object_name => '&corruptionTable',
  object_type => rdsadmin.rdsadmin_dbms_re-
pair.table_object,
  flags => rdsadmin.rdsadmin_dbms_re-
pair.noskip_flag);
end;
/

select skip_corrupt from dba_tables where owner
= '&corruptionOwner' and table_name = '&corrup-
tionTable';
```

5. When you have completed all repair work, run the following procedures to drop the repair tables.

```
exec rdsadmin.rdsadmin_dbms_repair.drop_repair_table;
exec rdsadmin.rdsadmin_dbms_repair.drop_orphan_keys_table;
```

Resizing the temporary tablespace

in a read replica

By default, Oracle tablespaces are created with auto-extend enabled and no maximum size. Because of these default settings, tablespaces can grow too large in some cases. We recommend that you specify an appropriate maximum size on permanent and temporary tablespaces, and that you carefully monitor space usage.

To resize the temporary space in a read replica for an Oracle DB instance, use either the rdsadmin.rdsadmin_util.resize_temp_tablespace or the rdsadmin.rdsadmin_util.resize_tempfile Amazon RDS procedure.

The resize_temp_tablespace procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
temp_tbs	varchar2	—	Yes	The name of the temporary tablespace to resize.
size	varchar2	—	Yes	You can specify the size in bytes (the default), kilobytes (K), megabytes (M), or gigabytes (G).

The resize_tempfile procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
file_id	binary_integer	—	Yes	The file identifier of the temporary tablespace to resize.

size	varchar2	—	Yes	You can specify the size in bytes (the default), kilobytes (K), megabytes (M), or gigabytes (G).

The following examples resize a temporary tablespace named `TEMP` to the size of 4 gigabytes on a read replica.

```
exec rdsadmin.rdsadmin_util.resize_temp_tablespace('TEMP','4G');
exec rdsadmin.rdsadmin_util.resize_temp_tablespace('TEMP','4096000000');
```

The following example resizes a temporary tablespace based on the tempfile with the file identifier `1` to the size of 2 megabytes on a read replica.

```
exec rdsadmin.rdsadmin_util.resize_tempfile(1,'2M');
```

Purging the recycle bin

When you drop a table, your Oracle database doesn't immediately remove its storage space. The database renames the table and places it and any associated objects in a recycle bin. Purging the recycle bin removes these items and releases their storage space.

To purge the entire recycle bin, use the Amazon RDS procedure `rdsadmin.rdsadmin_util.purge_dba_recyclebin`. However, this procedure can't purge the recycle bin of `SYS` and `RDSADMIN` objects. If you need to purge these objects, contact AWS Support.

The following example purges the entire recycle bin.

```
exec rdsadmin.rdsadmin_util.purge_dba_recyclebin;
```

Performing common log-related tasks for Oracle DB instances

Following, you can find how to perform certain common DBA tasks related to logging on your Amazon RDS DB instances running Oracle. To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances, and restricts access to certain system procedures and tables that require advanced privileges.

Setting force logging

In force logging mode, Oracle logs all changes to the database except changes in temporary tablespaces and temporary segments (NOLOGGING clauses are ignored). To set force logging, use the Amazon RDS procedure rdsadmin.rdsadmin_util.force_logging.

The force_logging procedure has the following parameters.

Parameter name	Data type	Default	Yes	Description
p_enable	boolean	true	No	Set to true to put the database in force logging mode, false to remove the database from force logging mode.

The following example puts the database in force logging mode.

```
exec rdsadmin.rdsadmin_util.force_logging(p_enable => true);
```

Setting supplemental logging

If you enable supplemental logging, LogMiner has the necessary information to support chained rows and clustered tables.

Oracle Database doesn't enable supplemental logging by default. To enable and disable supplemental logging, use the Amazon RDS procedure rdsadmin.rdsadmin_util.alter_supplemental_logging.

The alter_supplemental_logging procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
p_action	varchar2	—	Yes	'ADD' to add supplemental logging, 'DROP' to drop supplemental logging.
p_type	varchar2	null	No	The type of supplemental logging. Valid values are 'ALL', 'FOREIGN KEY', 'PRIMARY KEY', 'UNIQUE', or PROCEDURAL.

The following example enables supplemental logging.

```
begin
  rdsadmin.rdsadmin_util.alter_supplemental_logging(
```

```
    p_action => 'ADD');
end;
/
```

The following example enables supplemental logging for all fixed-length maximum size columns.

```
begin
  rdsadmin.rdsadmin_util.alter_supplemental_logging(
    p_action => 'ADD',
    p_type   => 'ALL');
end;
/
```

The following example enables supplemental logging for primary key columns.

```
begin
  rdsadmin.rdsadmin_util.alter_supplemental_logging(
    p_action => 'ADD',
```

```
    p_type  => 'PRIMARY KEY');
end;
/
```

Switching online log files

To switch log files, use the Amazon RDS procedure rdsadmin.rdsadmin_util.switch_logfile. The switch_logfile procedure has no parameters.

The following example switches log files.

```
exec rdsadmin.rdsadmin_util.switch_logfile;
```

Adding online redo logs

An Amazon RDS DB instance running Oracle starts with four online redo logs, 128 MB each. To add additional redo logs, use the Amazon RDS procedure rdsadmin.rdsadmin_util.add_logfile.

The add_logfile procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
bytes	positive	null	No	The size of the log file in bytes.
p_size	varchar2	—	Yes	The size of the log file. You can specify the size in kilobytes (K), megabytes (M), or gigabytes (G).

The following command adds a 100 MB log file.

```
exec rdsadmin.rdsadmin_util.add_logfile(p_size => '100M');
```

Dropping online redo logs

To drop redo logs, use the Amazon RDS procedure rdsadmin.rdsadmin_util.drop_logfile.

The drop_logfile procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
grp	positive	—	Yes	The group number of the log.

The following example drops the log with group number 3.

```
exec rdsadmin.rdsadmin_util.drop_logfile(grp => 3);
```

You can only drop logs that have a status of unused or inactive. The following example gets the statuses of the logs.

```
SELECT GROUP#, STATUS FROM V$LOG;

GROUP#    STATUS
--------- ----------
1         CURRENT
```

```
2       INACTIVE
3       INACTIVE
4       UNUSED
```

Resizing online redo logs

An Amazon RDS DB instance running Oracle starts with four online redo logs, 128 MB each. The following example shows how you can use Amazon RDS procedures to resize your logs from 128 MB each to 512 MB each.

```
/* Query V$LOG to see the logs.       */
/* You start with 4 logs of 128 MB each. */

SELECT GROUP#, BYTES, STATUS FROM V$LOG;

GROUP#    BYTES     STATUS
---------- ---------- ----------------
1       134217728 INACTIVE
2       134217728 CURRENT
```

```
3          134217728 INACTIVE
4          134217728 INACTIVE

/* Add four new logs that are each 512 MB */

exec rdsadmin.rdsadmin_util.add_logfile(bytes =>
536870912);
exec rdsadmin.rdsadmin_util.add_logfile(bytes =>
536870912);
exec rdsadmin.rdsadmin_util.add_logfile(bytes =>
536870912);
exec rdsadmin.rdsadmin_util.add_logfile(bytes =>
536870912);

/* Query V$LOG to see the logs. */
/* Now there are 8 logs.       */

SELECT GROUP#, BYTES, STATUS FROM V$LOG;

GROUP#    BYTES     STATUS
---------- ---------- ----------------
```

```
1       134217728 INACTIVE
2       134217728 CURRENT
3       134217728 INACTIVE
4       134217728 INACTIVE
5       536870912 UNUSED
6       536870912 UNUSED
7       536870912 UNUSED
8       536870912 UNUSED

/* Drop each inactive log using the group number. */

exec rdsadmin.rdsadmin_util.drop_logfile(grp =>
1);
exec rdsadmin.rdsadmin_util.drop_logfile(grp =>
3);
exec rdsadmin.rdsadmin_util.drop_logfile(grp =>
4);

/* Query V$LOG to see the logs. */
/* Now there are 5 logs.      */
```

```
select GROUP#, BYTES, STATUS from V$LOG;

GROUP#    BYTES     STATUS
---------- ---------- ----------------
2       134217728 CURRENT
5       536870912 UNUSED
6       536870912 UNUSED
7       536870912 UNUSED
8       536870912 UNUSED

/* Switch logs so that group 2 is no longer current. */

exec rdsadmin.rdsadmin_util.switch_logfile;


/* Query V$LOG to see the logs.      */
/* Now one of the new logs is current. */

SQL>SELECT GROUP#, BYTES, STATUS FROM V
$LOG;

GROUP#    BYTES     STATUS
```

```
---------- ---------- ----------------
2       134217728 ACTIVE
5       536870912 CURRENT
6       536870912 UNUSED
7       536870912 UNUSED
8       536870912 UNUSED

/* If the status of log 2 is still "ACTIVE", issue a check-
point to clear it to "INACTIVE".  */

exec rdsadmin.rdsadmin_util.checkpoint;

/* Query V$LOG to see the logs.        */
/* Now the final original log is inactive. */

select GROUP#, BYTES, STATUS from V$LOG;

GROUP#    BYTES     STATUS
---------- ---------- ----------------
2       134217728 INACTIVE
5       536870912 CURRENT
```

```
6       536870912 UNUSED
7       536870912 UNUSED
8       536870912 UNUSED

# Drop the final inactive log.

exec rdsadmin.rdsadmin_util.drop_logfile(grp =>
2);

/* Query V$LOG to see the logs.   */
/* Now there are four 512 MB logs. */

SELECT GROUP#, BYTES, STATUS FROM V$LOG;

GROUP#    BYTES     STATUS
---------- ---------- ----------------
5       536870912 CURRENT
6       536870912 UNUSED
7       536870912 UNUSED
8       536870912 UNUSED
```

Retaining archived redo logs

You can retain archived redo logs locally on your DB instance for use with products like Oracle LogMiner (DBMS_LOGMNR). After you have retained the redo logs, you can use LogMiner to analyze the logs.

To retain archived redo logs, use the Amazon RDS procedure rdsadmin.rdsadmin_util.set_configuration. The set_configuration procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
name	varchar	—	Yes	The name of the configuration to update.
value	varchar	—	Yes	The value for the

				configu-ration.

The following example retains 24 hours of redo logs.

```
begin
  rdsadmin.rdsadmin_util.set_configuration(
    name  => 'archivelog retention hours',
    value => '24');
end;
/
commit;
```

To view how long archived redo logs are kept for your DB instance, use the Amazon RDS procedure rdsadmin.rdsadmin_util.show_configuration.

The following example shows the log retention time.

```
set serveroutput on
```

```
exec rdsadmin.rdsadmin_util.show_configuration;
```

The output shows the current setting for archivelog retention hours. The following output shows that archived redo logs are kept for 48 hours.

```
NAME:archivelog retention hours
VALUE:48
DESCRIPTION:ArchiveLog expiration specifies the duration in hours before archive/redo log files are automatically deleted.
```

Because the archived redo logs are retained on your DB instance, ensure that your DB instance has enough allocated storage for the retained logs. To determine how much space your DB instance has used in the last X hours, you can run the following query, replacing X with the number of hours.

```
select sum(BLOCKS * BLOCK_SIZE) bytes
 from V$ARCHIVED_LOG
```

```
where FIRST_TIME >= SYSDATE-(X/24) and
DEST_ID=1;
```

Archived redo logs are only generated if the backup retention period of your DB instance is greater than zero. By default the backup retention period is greater than zero, so unless you explicitly set yours to zero, archived redo logs are generated for your DB instance.

After the archived redo logs are removed from your DB instance, you can't download them again to your DB instance. Amazon RDS retains the archived redo logs outside of your DB instance to support restoring your DB instance to a point in time. Amazon RDS retains the archived redo logs outside of your DB instance based on the backup retention period configured for your DB instance.

Accessing transaction logs

Accessing transaction logs is supported for Oracle version 12.1.0.2.v7 and later, all 12.2.0.1 versions, all 18.0.0.0 versions, and all 19.0.0 versions.

You might want to access your online and archived redo log files for mining with external tools such as GoldenGate, Attunity, Informatica, and others. If you want to access your online and archived redo log files, you must first create directory objects that provide read-only access to the physical file paths.

The following code creates directories that provide read-only access to your online and archived redo log files:

This code also revokes the DROP ANY DIRECTORY privilege.

```
exec rdsadmin.rdsadmin_master_util.create_archivelog_dir;
exec rdsadmin.rdsadmin_master_util.create_onlinelog_dir;
```

After you create directory objects for your online and archived redo log files, you can read the files by using PL/SQL.

The following code drops the directories for your online and archived redo log files.

```
exec rdsadmin.rdsadmin_master_util.
drop_archivelog_dir;
exec rdsadmin.rdsadmin_master_util.drop_on-
linelog_dir;
```

The following code grants and revokes the DROP ANY DIRECTORY privilege.

```
exec rdsadmin.rdsadmin_master_util.revoke_
drop_any_directory;
exec rdsadmin.rdsadmin_master_util.grant_
drop_any_directory;
```

Performing common RMAN tasks

for Oracle DB instances

In the following section, you can find how you can perform Oracle Recovery Manager (RMAN) DBA tasks on your Amazon RDS DB instances running Oracle. To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances. It also restricts access to certain system procedures and tables that require advanced privileges.

You can use the Amazon RDS package rdsadmin.rdsadmin_rman_util to perform RMAN backups of your Amazon RDS for Oracle database to disk. The rdsadmin.rdsadmin_rman_util package supports full and incremental database file backups, tablespace backups, and archive log backups.

RMAN backups consume storage space on the Amazon RDS DB instance host. When you perform

a backup, you specify an Oracle directory object as a parameter in the procedure call. The backup files are placed in the specified directory. You can use default directories, such as DATA_PUMP_DIR, or create a new directory.

After an RMAN backup has finished, you can copy the backup files off the Amazon RDS for Oracle DB instance host. You might do this for the purpose of restoring to a non-RDS host or for long-term storage of backups.

The backup files for RMAN backups remain on the Amazon RDS DB instance host until you remove them manually. You can use the UTL_FILE.FREMOVE Oracle procedure to remove files from a directory.

When backing up archived redo logs or performing a full or incremental backup that includes archived redo logs, redo log retention must be set to a nonzero value.

Currently, RMAN restore isn't supported for Amazon RDS for Oracle DB instances.

Common parameters for RMAN procedures

You can use procedures in the Amazon RDS package rdsadmin.rdsadmin_rman_util to perform tasks with RMAN. Several parameters are common to the procedures in the package. The package has the following common parameters.

Parameter name	Data type	Valid values	Default	Required	Description
p_owner	varchar2	A valid owner of the directory specified in p_directory_name.	—	Yes	The owner of the directory to contain the backup files.
p_directory_name	varchar2	A valid database directory name.	–	Yes	The name of the directory to contain the backup files.
p_label	varchar2	a-z, A-Z, 0-9, _, ., :	—	No	A unique string that is included in the backup file names. **Note** The limit is 30 characters.
p_compress	boolean	TRUE, FALSE	FALSE	No	Specify TRUE to enable BASIC backup compression. Specify FALSE to disable BASIC backup compression.
p_include_archive_logs	boolean	TRUE, FALSE	FALSE	No	Specify TRUE to include archived redo logs in the backup. Specify FALSE to exclude archived redo logs from the backup. If you include archived redo logs in the backup, set retention to one hour or greater using the rdsadmin.rdsadmin_util.set_configuration procedure. Also, call the rdsadmin.rdsadmin_rman_util.crosscheck_archivelog procedure immediately before running the backup. Otherwise, the backup might fail due to missing archived redo log files that have been deleted by Amazon RDS management procedures.
p_include_controlfile	boolean	TRUE, FALSE	FALSE	No	Specify TRUE to include the control file in the backup. Specify FALSE to exclude the control file from the backup.
p_optimize	boolean	TRUE, FALSE	TRUE	No	Specify TRUE to enable backup optimization, if archived redo logs are included, to reduce backup size. Specify FALSE to disable backup optimization.
p_parallel	number	A valid integer between 1 and 254 for Oracle Database Enterprise Edition (EE) 1 for other Oracle Database editions	1	No	Number of channels.
p_rman_to_dbms_output	boolean	TRUE, FALSE	FALSE	No	When TRUE, the RMAN output is sent to the DBMS_OUTPUT package in addition to a file in the BDUMP directory. In SQL*Plus, use SET SERVEROUTPUT ON to see the output.

					When FALSE, the RMAN output is only sent to a file in the BDUMP directory.
p_section_size_mb	number	A valid integer	NULL	No	The section size in megabytes (MB). Validates in parallel by dividing each file into the specified section size. When NULL, the parameter is ignored.
p_validation_type	varchar2	'PHYSICAL', 'PHYSICAL+LOGICAL'	'PHYSICAL'	No	The level of corruption detection. Specify 'PHYSICAL' to check for physical corruption. An example of physical corruption is a block with a mismatch in the header and footer. Specify 'PHYSICAL+LOGICAL' to check for logical inconsistencies in addition to physical corruption. An example of logical corruption is a corrupt block.

Validating DB instance files

You can use the Amazon RDS package `rdsadmin.rdsadmin_rman_util` to validate Amazon RDS for Oracle DB instance files, such as data files, tablespaces, control files, or server parameter files (SPFILEs).

Validating a DB instance

To validate all of the relevant files used by an Amazon RDS Oracle DB instance, use the Amazon RDS procedure `rdsadmin.rdsadmin_rman_util.validate_database`.

This procedure uses the following common parameters for RMAN tasks:

- `p_validation_type`

- p_parallel
- p_section_size_mb
- p_rman_to_dbms_output

The following example validates the DB instance using the default values for the parameters.

```
exec rdsadmin.rdsadmin_rman_util.validate_
database;
```

The following example validates the DB instance using the specified values for the parameters.

```
BEGIN
  rdsadmin.rdsadmin_rman_util.validate_
database(
    p_validation_type    => 'PHYSICAL+LOGICAL',
    p_parallel           => 4,
    p_section_size_mb    => 10,
    p_rman_to_dbms_output => FALSE);
END;
/
```

When the p_rman_to_dbms_output parameter is set to FALSE, the RMAN output is written to a file in the BDUMP directory.

To view the files in the BDUMP directory, run the following SELECT statement.

```
SELECT * FROM table(rdsadmin.rds_file_util.list-
dir('BDUMP')) order by mtime;
```

To view the contents of a file in the BDUMP directory, run the following SELECT statement.

```
SELECT text FROM table(rdsadmin.rds_file_u-
til.read_text_file('BDUMP','rds-rman-validate-
nnn.txt'));
```

Replace the file name with the name of the file you want to view.

Validating a tablespace

To validate the files associated with a tablespace, use the Amazon RDS procedure `rdsadmin.rdsadmin_rman_util.validate_tablespace`.

This procedure uses the following common parameters for RMAN tasks:

- `p_validation_type`
- `p_parallel`
- `p_section_size_mb`
- `p_rman_to_dbms_output`

This procedure also uses the following additional parameter.

Parameter name	Data type	Valid values	Default	Required	Description
p_tablespace_name	varchar2	A valid tablespace name	—	Yes	The name of the tablespace.

Validating a control file

To validate only the control file used by an Amazon RDS Oracle DB instance, use the Amazon RDS procedure `rdsadmin.rdsadmin_rman_util.validate_current_controlfile`.

This procedure uses the following common parameter for RMAN tasks:

- `p_validation_type`
- `p_rman_to_dbms_output`

Validating an SPFILE

To validate only the server parameter file (SPFILE) used by an Amazon RDS Oracle DB instance, use the Amazon RDS procedure `rdsadmin.rdsadmin_rman_util.validate_spfile`.

This procedure uses the following common parameter for RMAN tasks:

- `p_validation_type`
- `p_rman_to_dbms_output`

Validating a data file

To validate a data file, use the Amazon RDS procedure rdsadmin.rdsadmin_rman_util.validate_datafile.

This procedure uses the following common parameters for RMAN tasks:

- p_validation_type
- p_parallel
- p_section_size_mb
- p_rman_to_dbms_output

This procedure also uses the following additional parameters.

Parameter name	Data type	Valid values	Default	Required	Description
p_datafile	varchar2	A valid datafile ID number or a valid datafile name	—	Yes	The datafile ID number (from v$datafile.file#) or the

		including complete path			full datafile name including the path (from v $datafile. name).
p_from_block	number	A valid integer	NULL	No	The number of the block where the validation starts within the data file. When this is NUL-L, 1 is used.
p_to_block	number	A valid integer	NULL	No	The number of the block where the valida-tion ends within the data file. When this is NULL, the maxi-mum block in the data file is used.

Enabling and disabling block change tracking

Block changing tracking records changed blocks in a tracking file. This technique can improve the performance of incremental backups.

To enable block change tracking for a DB instance, use the Amazon RDS procedure `rdsadmin.rdsadmin_rman_util.enable_block_change_tracking`. To disable block change tracking, use `disable_block_change_tracking`. These procedures take no parameters.

Read-only replicas support block change tracking. If you create a read-only replica from a source DB that uses block change tracking, the replica uses block change tracking. You can't enable block change tracking on a mounted replica. If you place a mounted replica in read-only mode, block change tracking isn't enabled, but you can enable it us-

ing enable_block_change_tracking. If you promote an Oracle replica to a source DB, you can use block change tracking just as for any other Oracle DB instance.

Block change tracking procedures are supported for the following DB engine versions:

- 12.1.0.2.v15 or higher 12.1 versions
- 12.2.0.1.ru-2019-01.rur-2019-01.r1 or higher 12.2 versions
- All 18.0.0.0 versions
- All 19.0.0.0 versions

To determine whether block change tracking is enabled for your DB instance, run the following query.

```
SELECT STATUS, FILENAME FROM V$BLOCK-
_CHANGE_TRACKING;
```

The following example enables block change tracking for a DB instance.

```
EXEC rdsadmin.rdsadmin_rman_util.en-
able_block_change_tracking;
```

The following example disables block change tracking for a DB instance.

```
EXEC rdsadmin.rdsadmin_rman_util.dis-
able_block_change_tracking;
```

Crosschecking archived redo logs

You can crosscheck archived redo logs using the Amazon RDS procedure rdsadmin.rdsadmin_rman_util.crosscheck_archivelog.

You can use this procedure to crosscheck the archived redo logs registered in the control file and optionally delete the expired logs records. When RMAN makes a backup, it creates a record in the control file. Over time, these records increase the

size of the control file. We recommend that you remove expired records periodically.

This procedure uses the common parameter p_rman_to_dbms_output for RMAN tasks.

This procedure also uses the following additional parameter.

Parameter name	Data type	Valid values	Default	Required	Description
p_delete_expired	boolean	TRUE, FALSE	TRUE	No	When TRUE, delete expired archived redo log records from the control file. When FALSE, retain the expired archived redo log records in the control file.

This procedure is supported for the following Amazon RDS for Oracle DB engine versions:

- 12.1.0.2.v15 or higher 12.1 versions
- 12.2.0.1.ru-2019-01.rur-2019-01.r1 or higher 12.2 versions
- All 18.0.0.0 versions
- All 19.0.0.0 versions

The following example marks archived redo log records in the control file as expired, but does not delete the records.

```
BEGIN
  rdsadmin.rdsadmin_rman_util.crosscheck-
_archivelog(
    p_delete_expired    => FALSE,
    p_rman_to_dbms_output => FALSE);
END;
/
```

The following example deletes expired archived redo log records from the control file.

```
BEGIN
  rdsadmin.rdsadmin_rman_util.crosscheck-
_archivelog(
    p_delete_expired    => TRUE,
    p_rman_to_dbms_output => FALSE);
END;
```

/

Backing up archived redo logs

You can use the Amazon RDS package rdsadmin.rdsadmin_rman_util to back up archived redo logs for an Amazon RDS Oracle DB instance.

The procedures for backing up archived redo logs are supported for the following Amazon RDS for Oracle DB engine versions:

- 12.1.0.2.v15 or higher 12.1 versions
- 12.2.0.1.ru-2019-01.rur-2019-01.r1 or higher 12.2 versions
- All 18.0.0.0 versions
- All 19.0.0.0 versions

Backing up all archived redo logs

To back up all of the archived redo logs for an Amazon RDS Oracle DB instance, use the Amazon RDS

procedure rdsadmin.rdsadmin_rman_util.backup_archivelog_all.

This procedure uses the following common parameters for RMAN tasks:

- p_owner
- p_directory_name
- p_label
- p_parallel
- p_compress
- p_rman_to_dbms_output

The following example backs up all archived redo logs for the DB instance.

```
BEGIN
  rdsadmin.rdsadmin_rman_util.backup_archivelog_all(
    p_owner             => 'SYS',
    p_directory_name    => 'MYDIRECTORY',
    p_parallel          => 4,
```

```
    p_rman_to_dbms_output => FALSE);
END;
/
```

Backing up an archived redo log from a date range

To back up specific archived redo logs for an Amazon RDS Oracle DB instance by specifying a date range, use the Amazon RDS procedure rdsadmin.rdsadmin_rman_util.backup_archivelog_date. The date range specifies which archived redo logs to back up.

This procedure uses the following common parameters for RMAN tasks:

- p_owner
- p_directory_name
- p_label
- p_parallel
- p_compress

- p_rman_to_dbms_output

This procedure also uses the following additional parameters.

Parameter name	Data type	Valid values	Default	Required	Description
p_from_date	date	A date that is between the start_date and next_date of an archived redo log that exists on disk. The value must be less than or equal to the value specified for p_to_date.	—	Yes	The starting date for the archived log backups.
p_to_date	date	A date that is between the start_date and next_date of an archived redo log that exists on disk. The value must be greater than or equal to the value specified for p_from_date.	—	Yes	The ending date for the archived log backups.

The following example backs up archived redo logs in the date range for the DB instance.

```
BEGIN
  rdsadmin.rdsadmin_rman_util.backup_archivelog_date(
    p_owner             => 'SYS',
    p_directory_name    => 'MYDIRECTORY',
    p_from_date         => '03/01/2019 00:00:00',
    p_to_date           => '03/02/2019 00:00:00',
    p_parallel          => 4,
```

```
    p_rman_to_dbms_output => FALSE);
END;
/
```

Backing up an archived redo log from an SCN range

To back up specific archived redo logs for an Amazon RDS Oracle DB instance by specifying a system change number (SCN) range, use the Amazon RDS procedure `rdsadmin.rdsadmin_rman_util.back-up_archivelog_scn`. The SCN range specifies which archived redo logs to back up.

This procedure uses the following common parameters for RMAN tasks:

- `p_owner`
- `p_directory_name`
- `p_label`
- `p_parallel`

- p_compress
- p_rman_to_dbms_output

This procedure also uses the following additional parameters.

Parameter name	Data type	Valid values	Default	Required	Description
p_from_scn	number	An SCN of an archived redo log that exists on disk. The value must be less than or equal to the value specified for p_to_scn.	—	Yes	The starting SCN for the archived log backups.
p_to_scn	number	An SCN of an archived redo log that exists on disk. The value must be greater than or equal to the value specified for p_from_scn.	—	Yes	The ending SCN for the archived log backups.

The following example backs up archived redo logs in the SCN range for the DB instance.

```
BEGIN
  rdsadmin.rdsadmin_rman_util.backup_archivel-
og_scn(
    p_owner          => 'SYS',
    p_directory_name    => 'MYDIRECTORY',
```

```
    p_from_scn      => 1533835,
    p_to_scn        => 1892447,
    p_parallel      => 4,
    p_rman_to_dbms_output => FALSE);
END;
/
```

Backing up an archived redo log from a sequence number range

To back up specific archived redo logs for an Amazon RDS Oracle DB instance by specifying a sequence number range, use the Amazon RDS procedure `rdsadmin.rdsadmin_rman_util.back-up_archivelog_sequence`. The sequence number range specifies which archived redo logs to back up.

This procedure uses the following common parameters for RMAN tasks:

- `p_owner`
- `p_directory_name`

- p_label
- p_parallel
- p_compress
- p_rman_to_dbms_output

This procedure also uses the following additional parameters.

Parameter name	Data type	Valid values	Default	Required	Description
p_from_sequence	number	A sequence number an archived redo log that exists on disk. The value must be less than or equal to the value specified for p_to_sequence.	—	Yes	The starting sequence number for the archived log backups.
p_to_sequence	number	A sequence number of an archived redo log that exists on disk. The value must be greater than or equal to the value specified for p_from_sequence.	—	Yes	The ending sequence number for the archived log backups.

The following example backs up archived redo logs in the sequence number range for the DB instance.

```
BEGIN
  rdsadmin.rdsadmin_rman_util.backup_archivel-
og_sequence(
    p_owner             => 'SYS',
    p_directory_name    => 'MYDIRECTORY',
```

```
    p_from_sequence     => 11160,
    p_to_sequence       => 11160,
    p_parallel          => 4,
    p_rman_to_dbms_output => FALSE);
END;
/
```

Performing a full database backup

You can perform a backup of all blocks of data files included in the backup using Amazon RDS procedure rdsadmin.rdsadmin_rman_util.backup_database_full.

This procedure uses the following common parameters for RMAN tasks:

- p_owner
- p_directory_name
- p_label
- p_parallel

- p_section_size_mb
- p_include_archive_logs
- p_optimize
- p_compress
- p_rman_to_dbms_output

This procedure is supported for the following Amazon RDS for Oracle DB engine versions:

- 12.1.0.2.v15 or higher 12.1 versions
- 12.2.0.1.ru-2019-01.rur-2019-01.r1 or higher 12.2 versions
- All 18.0.0.0 versions
- All 19.0.0.0 versions

The following example performs a full backup of the DB instance using the specified values for the parameters.

```
BEGIN
  rdsadmin.rdsadmin_rman_util.backup_
database_full(
```

```
    p_owner          => 'SYS',
    p_directory_name    => 'MYDIRECTORY',
    p_parallel       => 4,
    p_section_size_mb   => 10,
    p_rman_to_dbms_output => FALSE);
END;
/
```

Performing an incremental database backup

You can perform an incremental backup of your DB instance using the Amazon RDS procedure rdsadmin.rdsadmin_rman_util.backup_database_incremental.

This procedure uses the following common parameters for RMAN tasks:

- p_owner
- p_directory_name
- p_label

- p_parallel
- p_section_size_mb
- p_include_archive_logs
- p_include_controlfile
- p_optimize
- p_compress
- p_rman_to_dbms_output

This procedure is supported for the following Amazon RDS for Oracle DB engine versions:

- 12.1.0.2.v15 or higher 12.1 versions
- 12.2.0.1.ru-2019-01.rur-2019-01.r1 or higher 12.2 versions
- All 18.0.0.0 versions
- All 19.0.0.0 versions

This procedure also uses the following additional parameter.

Parameter name	Data type	Valid values	Default	Required	Description
p_level	number	0, 1	0	No	Specify 0 to enable a full incremental backup.

					Specify 1 to enable a non-cumulative incremental backup.

The following example performs an incremental backup of the DB instance using the specified values for the parameters.

```
BEGIN
  rdsadmin.rdsadmin_rman_util.backup_
database_incremental(
    p_owner             => 'SYS',
    p_directory_name    => 'MYDIRECTORY',
    p_level             => 1,
    p_parallel          => 4,
    p_section_size_mb   => 10,
    p_rman_to_dbms_output => FALSE);
END;
/
```

Performing a tablespace backup

You can perform a DB instance tablespace using the Amazon RDS procedure rdsadmin.rdsadmin_rman_util.backup_tablespace.

This procedure uses the following common parameters for RMAN tasks:

- p_owner
- p_directory_name
- p_label
- p_parallel
- p_section_size_mb
- p_include_archive_logs
- p_include_controlfile
- p_optimize
- p_compress
- p_rman_to_dbms_output

This procedure also uses the following additional parameter.

Parameter name	Data type	Valid values	Default	Required	Description

p_tablespace_name	varchar2	A valid tablespace name.	—	Yes	The name of the tablespace to back up.

This procedure is supported for the following Amazon RDS for Oracle DB engine versions:

- 12.1.0.2.v15 or higher 12.1 versions
- 12.2.0.1.ru-2019-01.rur-2019-01.r1 or higher 12.2 versions
- All 18.0.0.0 versions
- All 19.0.0.0 versions

The following example performs a tablespace backup using the specified values for the parameters.

```
BEGIN
  rdsadmin.rdsadmin_rman_util.backup_tablespace(
    p_owner          => 'SYS',
    p_directory_name    => 'MYDIRECTORY',
    p_tablespace_name    => MYTABLESPACE,
    p_parallel       => 4,
```

```
    p_section_size_mb    => 10,
    p_rman_to_dbms_output => FALSE);
END;
/
```

Performing common scheduling tasks for Oracle DB instances

Some SYS-owned scheduler jobs can interfere with normal database operations, and Oracle Support recommends they be disabled or the job schedule be modified. You can use the Amazon RDS package `rdsadmin.rdsadmin_dbms_scheduler` to perform tasks for SYS-owned Oracle Scheduler jobs.

These procedures are supported for the following Amazon RDS for Oracle DB engine versions:

- 19c
- 18c
- 12.2.0.2.ru-2019-07.rur-2019-07.r1 or higher 12.2 versions
- 12.1.0.2.v17 or higher 12.1 versions

Common parameters for Oracle Scheduler procedures

To perform tasks with Oracle Scheduler, use procedures in the Amazon RDS package rdsadmin.rdsadmin_dbms_scheduler. Several parameters are common to the procedures in the package. The package has the following common parameters.

Parameter name	Data type	Valid values	Default	Required	Description
name	varchar2	'SYS.BSLN_MAINTAIN_STATS_JOB','SYS.CLEANUP_ONLINE_IND_BUILD'	—	Yes	The name of the job to modify. **Note** Currently, you can only modify SYS.CLEANUP_ONLINE_IND_BUILD and SYS.BSLN_MAINTAIN_S-

					TATS_JOB jobs.
attribute	varchar2	'REPEAT_INTERVAL','SCHEDULE_NAME'	–	Yes	Attribute to modify. To modify the repeat interval for the job, specify 'REPEAT_INTERVAL'. To modify the schedule name for the job, specify 'SCHEDULE_NAME'.
value	varchar2	A valid schedule interval or schedule name, depending on attribute used.	–	Yes	The new value of the attribute.

Modifying DBMS_SCHEDULER jobs

You can use the Oracle procedure dbms_scheduler.set_attribute to modify certain components of Oracle Scheduler.

When working with Amazon RDS DB instances, prepend the schema name SYS to the object name. The following example sets the resource plan attribute for the Monday window object.

```
begin
  dbms_scheduler.set_attribute(
    name      => 'SYS.MONDAY_WINDOW',
    attribute => 'RESOURCE_PLAN',
    value     => 'resource_plan_1');
end;
/
```

Setting the time zone for Oracle Scheduler jobs

To modify the time zone for Oracle Scheduler, you can use the Oracle procedure dbms_scheduler.set_scheduler_attribute.

To modify the current time zone setting

1. Connect to the database using a client such as SQL Developer.
2. Set the default time zone as following, substituting your time zone for *time_zone_name*.

```
begin
 dbms_scheduler.set_scheduler_attribute(
  attribute => 'default_timezone',
  value => 'time_zone_name'
 );
end;
/
```

In the following example, you change the time zone to Asia/Shanghai.

Start by querying the current time zone, as shown following.

```
SELECT VALUE FROM DBA_SCHEDULER_GLOB-
AL_ATTRIBUTE WHERE ATTRIBUTE_NAME='DE-
FAULT_TIMEZONE';
```

The output shows that the current time zone is ETC/UTC.

```
VALUE
-------
Etc/UTC
```

Then you set the time zone to Asia/Shanghai.

```
begin
 dbms_scheduler.set_scheduler_attribute(
  attribute => 'default_timezone',
  value => 'Asia/Shanghai'
 );
end;
/
```

Disabling SYS-owned Oracle Scheduler jobs

To disable a SYS-owned Oracle Scheduler job, use the rdsadmin.rdsadmin_dbms_scheduler.disable procedure.

This procedure uses the name common parameter for Oracle Scheduler tasks.

The following example disables the SYS.CLEANUP_ONLINE_IND_BUILD Oracle Scheduler job.

```
BEGIN
  rdsadmin.rdsadmin_dbms_scheduler.disable('SYS.CLEANUP_ONLINE_IND_BUILD');
END;
/
```

Enabling SYS-owned Oracle Scheduler jobs

To enable a SYS-owned Oracle Scheduler job, use the rdsadmin.rdsadmin_dbms_scheduler.enable procedure.

This procedure uses the name common parameter for Oracle Scheduler tasks.

The following example enables the SYS.CLEANUP_ONLINE_IND_BUILD Oracle Scheduler job.

```
BEGIN
  rdsadmin.rdsadmin_dbms_scheduler.enable('SYS.CLEANUP_ONLINE_IND_BUILD');
END;
/
```

Modifying the repeat interval for

jobs of CALENDAR type

To modify the repeat interval to modify a SYS-owned Oracle Scheduler job of CALENDAR type, use the rdsadmin.rdsadmin_dbms_scheduler.disable procedure.

This procedure uses the following common parameters for Oracle Scheduler tasks:

- name
- attribute
- value

The following example modifies the repeat interval of the SYS.CLEANUP_ONLINE_IND_BUILD Oracle Scheduler job.

```
BEGIN
  rdsadmin.rdsadmin_dbms_scheduler.set_attribute(
    name      => 'SYS.CLEANUP_ONLINE_IND_BUILD',
    attribute => 'repeat_interval',
```

```
    value    =>
'freq=daily;byday=FRI,SAT;byhour=20;byminute=
0;bysecond=0');
END;
/
```

Modifying the repeat interval for

jobs of NAMED type

Some Oracle Scheduler jobs use a schedule name instead of an interval. For this type of jobs, you must create a new named schedule in the master user schema. Use the standard Oracle sys.dbms_scheduler.create_schedule procedure to do this. Also, use the rdsadmin.rdsadmin_dbms_scheduler.set_attribute procedure to assign the new named schedule to the job.

This procedure uses the following common parameter for Oracle Scheduler tasks:

- name
- attribute
- value

The following example modifies the repeat interval of the SYS.BSLN_MAINTAIN_STATS_JOB Oracle Scheduler job.

```
BEGIN
   dbms_scheduler.create_schedule (
      schedule_name  => 'rds_master_us-
er.new_schedule',
      start_date     => SYSTIMESTAMP,
      repeat_interval =>
'freq=daily;byday=MON,TUE,WED,THU,FRI;byhour
=0;byminute=0;bysecond=0',
      end_date       => NULL,
      comments       => 'Repeats daily forever');
END;
/
```

```
BEGIN
  rdsadmin.rdsadmin_dbms_scheduler.set_at-
tribute (
      name     => 'SYS.BSLN_MAINTAIN_STATS_JOB',
      attribute => 'schedule_name',
      value    => 'rds_master_user.new_schedule');
END;
/
```

Performing common diagnostic

tasks for Oracle DB instances

Oracle Database includes a fault diagnosability infrastructure that you can use to investigate database problems. In Oracle terminology, a *problem* is a critical error such as a code bug or data corruption. An *incident* is the occurrence of a problem. If the same error occurs three times, then the infrastructure shows three incidents of this problem.

The Automatic Diagnostic Repository Command Interpreter (ADRCI) utility is an Oracle command-line tool that you use to manage diagnostic data. For example, you can use this tool to investigate problems and package diagnostic data. An *incident package* includes diagnostic data for an incident or all incidents that reference a specific problem. You can upload an incident package, which is implemented as a .zip file, to Oracle Support.

To deliver a managed service experience, Amazon RDS doesn't provide shell access to ADRCI. To perform diagnostic tasks for your Oracle instance, instead use the Amazon RDS package rdsadmin.rdsadmin_adrci_util.

By using the functions in rdsadmin_adrci_util, you can list and package problems and incidents, and also show trace files. All functions return a task ID. This ID forms part of the name of log file that contains the ADRCI output, as in dbtask-*task_id*.log. The log file resides in the BDUMP directory.

Common parameters for diagnostic procedures

To perform diagnostic tasks, use functions in the Amazon RDS package rdsadmin.rdsadmin_adrci_util. The package has the following common parameters.

Parameter name	Data type	Valid values	Default	Required	Description

incident_id	number	A valid incident ID or null	Null	No	If the value is null, the function shows all incidents. If the value isn't null and represents a valid incident ID, the function shows the specified incident.
problem_id	number	A valid problem ID or null	Null	No	If the value is null, the function shows all problems. If the value isn't null and represents a valid problem ID, the function shows the specified problem.
last	number	A valid integer greater than 0 or null	Null	No	If the value is null, then the function displays at most 50 items. If the value isn't null, the function displays the specified number.

Listing incidents

To list diagnostic incidents for Oracle, use the Amazon RDS function `rdsadmin.rdsadmin_adrci_util.list_adrci_incidents`. You can list incidents in either basic or detailed mode. By default, the function lists the 50 most recent incidents.

This function uses the following common parameters:

- incident_id
- problem_id

If you specify both of the preceding parameters, incident_id overrides problem_id.

This function uses the following additional parameter.

Parameter name	Data type	Valid values	Default	Required	Description
detail	boolean	TRUE or FALSE	FALSE	No	If TRUE, the function lists incidents in detail mode. If FALSE, the function lists incidents in basic mode.

To list all incidents, call the rdsadmin.rdsadmin_adrci_util.list_adrci_incidents function without any arguments. You can store the output in a SQL client variable.

```
SQL> VAR task_id VARCHAR2(80);
SQL> exec :task_id := rdsadmin.rdsadmin_adrci_u-
til.list_adrci_incidents;

PL/SQL procedure successfully completed.
```

To get the task ID, specify the variable in a query of the dual table.

```
SQL> SELECT :task_id FROM DUAL;

:TASK_ID
----------------
1590786706158-3126
```

To read the log file, call the Amazon RDS procedure rdsadmin.rds_file_util.read_text_file. Supply the task ID as part of the file name. The following output shows three incidents: 53523, 53522, and 53521.

```
SQL> SELECT * FROM TABLE(rdsad-
min.rds_file_util.read_text_file('BDUMP',
'dbtask-'||:task_id||'.log'));

TEXT
--------------------------------------------------------------------
--------------------------------------------------
2020-05-29 21:11:46.193 UTC [INFO ] Listing
ADRCI incidents.
2020-05-29 21:11:46.256 UTC [INFO ]
ADR Home = /rdsdbdata/log/diag/rdbms/orcl_a/
ORCL:
**************************************************************
*******
INCIDENT_ID PROBLEM_KEY
CREATE_TIME
----------
----------------------------------------------------------
-----------------------------------
```

```
53523     ORA 700 [EVENT_CREATED_INCIDENT]
[942] [SIMULATED_ERROR_003 2020-05-29
20:15:20.928000 +00:00
53522     ORA 700 [EVENT_CREATED_INCIDENT]
[942] [SIMULATED_ERROR_002 2020-05-29
20:15:15.247000 +00:00
53521     ORA 700 [EVENT_CREATED_INCIDENT]
[942] [SIMULATED_ERROR_001 2020-05-29
20:15:06.047000 +00:00
3 rows fetched

2020-05-29 21:11:46.256 UTC [INFO ] The ADRCI
incidents were successfully listed.
2020-05-29 21:11:46.256 UTC [INFO ] The task
finished successfully.

14 rows selected.
```

To list a particular incident, specify its ID using the incident_id parameter. In the following example, you query the log file for incident 53523 only.

```
SQL> exec :task_id := rdsadmin.rdsadmin_adrci_u-
til.list_adrci_incidents(incident_id=>53523);

PL/SQL procedure successfully completed.

SQL> SELECT * FROM TABLE(rdsad-
min.rds_file_util.read_text_file('BDUMP',
'dbtask-'||:task_id||'.log'));

TEXT
--------------------------------------------------------------------------------
2020-05-29 21:15:25.358 UTC [INFO ] Listing
ADRCI incidents.
2020-05-29 21:15:25.426 UTC [INFO ]
ADR Home = /rdsdbdata/log/diag/rdbms/orcl_a/
ORCL:
*************************************************************************
*******
INCIDENT_ID       PROBLEM_KEY
CREATE_TIME
```

```
------------------
------------------------------------------------------------
------------------------------
53523           ORA 700 [EVENT_CREATED_INCI-
DENT] [942] [SIMULATED_ERROR_003 2020-05-29
20:15:20.928000 +00:00
1 rows fetched


2020-05-29 21:15:25.427 UTC [INFO ] The ADRCI
incidents were successfully listed.
2020-05-29 21:15:25.427 UTC [INFO ] The task
finished successfully.

12 rows selected.
```

Listing problems

To list diagnostic problems for Oracle, use the Amazon RDS function rdsadmin.rdsadmin_adrci_util.list_adrci_problems.

By default, the function lists the 50 most recent problems.

This function uses the common parameter problem_id.

To get the task ID for all problems, call the rdsadmin.rdsadmin_adrci_util.list_adrci_problems function without any arguments, and store the output in a SQL client variable.

```
SQL> exec :task_id := rdsadmin.rdsadmin_adrci_util.list_adrci_problems;

PL/SQL procedure successfully completed.
```

To read the log file, call the rdsadmin.rds_file_util.read_text_file function, supplying the task ID as part of the file name. In the following output, the log file shows three problems: 1, 2, and 3.

```
SQL> SELECT * FROM TABLE(rdsad-
min.rds_file_util.read_text_file('BDUMP',
'dbtask-'||:task_id||'.log'));

TEXT
--------------------------------------------------------------------------------
2020-05-29 21:18:50.764 UTC [INFO ] Listing
ADRCI problems.
2020-05-29 21:18:50.829 UTC [INFO ]
ADR Home = /rdsdbdata/log/diag/rdbms/orcl_a/
ORCL:
*********************************************************************
*******
PROBLEM_ID  PROBLEM_KEY
LAST_INCIDENT      LASTINC_TIME
----------
-------------------------------------------------------------
------------ -------------------------------
```

```
2       ORA 700 [EVENT_CREATED_INCIDENT] [942]
[SIMULATED_ERROR_003 53523      2020-05-29
20:15:20.928000 +00:00
3       ORA 700 [EVENT_CREATED_INCIDENT] [942]
[SIMULATED_ERROR_002 53522      2020-05-29
20:15:15.247000 +00:00
1       ORA 700 [EVENT_CREATED_INCIDENT] [942]
[SIMULATED_ERROR_001 53521      2020-05-29
20:15:06.047000 +00:00
3 rows fetched


2020-05-29 21:18:50.829 UTC [INFO ] The ADRCI
problems were successfully listed.
2020-05-29 21:18:50.829 UTC [INFO ] The task
finished successfully.

14 rows selected.
```

In the following example, you list problem 3 only.

```
SQL> exec :task_id := rdsadmin.rdsadmin_adrci_u-
til.list_adrci_problems(problem_id=>3);

PL/SQL procedure successfully completed.
```

To read the log file for problem 3, call rdsadmin.rds_file_util.read_text_file. Supply the task ID as part of the file name.

```
SQL> SELECT * FROM TABLE(rdsad-
min.rds_file_util.read_text_file('BDUMP',
'dbtask-'||:task_id||'.log'));

TEXT
--------------------------------------------------------------------------------
2020-05-29 21:19:42.533 UTC [INFO ] Listing
ADRCI problems.
2020-05-29 21:19:42.599 UTC [INFO ]
ADR Home = /rdsdbdata/log/diag/rdbms/orcl_a/
ORCL:
```

```
**********************************************************************
*******
PROBLEM_ID PROBLEM_KEY
LAST_INCIDENT LASTINC_TIME
---------
-------------------------------------------------------------
------------ ------------------------------
3        ORA 700 [EVENT_CREATED_INCIDENT] [942]
[SIMULATED_ERROR_002 53522       2020-05-29
20:15:15.247000 +00:00
1 rows fetched

2020-05-29 21:19:42.599 UTC [INFO ] The ADRCI
problems were successfully listed.
2020-05-29 21:19:42.599 UTC [INFO ] The task
finished successfully.

12 rows selected.
```

Creating incident packages

You can create incident packages using the Amazon RDS function rdsadmin.rdsadmin_adrci_util.create_adrci_package. The output is a .zip file that you can supply to Oracle Support.

This function uses the following common parameters:

- problem_id
- incident_id

Make sure to specify one of the preceding parameters. If you specify both parameters, incident_id overrides problem_id.

To create a package for a specific incident, call the Amazon RDS function rdsadmin.rdsadmin_adrci_util.create_adrci_package with the incident_id parameter. The following example creates a package for incident 53523.

```
SQL> exec :task_id := rdsadmin.rdsadmin_adrci_u-
til.create_adrci_package(incident_id=>53523);

PL/SQL procedure successfully completed.
```

To read the log file, call the rdsadmin.rds_-file_util.read_text_file. You can supply the task ID as part of the file name. The output shows that you generated incident package O-RA700EVE_20200529212043_COM_1.zip.

```
SSQL> SELECT * FROM TABLE(rdsad-
min.rds_file_util.read_text_file('BDUMP',
'dbtask-'||:task_id||'.log'));

TEXT
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----
2020-05-29 21:20:43.031 UTC [INFO ] The ADRCI
package is being created.
```

```
2020-05-29 21:20:47.641 UTC [INFO ] Gener-
ated package 1 in file /rdsdbdata/log/trace/
ORA700EVE_20200529212043_COM_1.zip, mode
complete
2020-05-29 21:20:47.642 UTC [INFO ] The ADRCI
package was successfully created.
2020-05-29 21:20:47.642 UTC [INFO ] The task
finished successfully.
```

To package diagnostic data for a particular problem, specify its ID using the problem_id parameter. In the following example, you package data for problem 3 only.

```
SQL> exec :task_id := rdsadmin.rdsadmin_adrci_u-
til.create_adrci_package(problem_id=>3);

PL/SQL procedure successfully completed.
```

To read the task output, call rdsadmin.rds_file_util.read_text_file, supplying the task ID as part of the file name. The output shows that you generated

incident package ORA700EVE_20200529212111_COM_1.zip.

```
SQL> SELECT * FROM TABLE(rdsadmin.rds_file_util.read_text_file('BDUMP', 'dbtask-'||:task_id||'.log'));

TEXT
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------------------------
2020-05-29 21:21:11.050 UTC [INFO ] The ADRCI package is being created.
2020-05-29 21:21:15.646 UTC [INFO ] Generated package 2 in file /rdsdbdata/log/trace/ORA700EVE_20200529212111_COM_1.zip, mode complete
2020-05-29 21:21:15.646 UTC [INFO ] The ADRCI package was successfully created.
2020-05-29 21:21:15.646 UTC [INFO ] The task finished successfully.
```

Showing trace files

You can show trace files using the Amazon RDS function `rdsadmin.rdsadmin_adrci_util.show_adrci_tracefile`.

This function uses the following parameter.

Parameter name	Data type	Valid values	Default	Required	Description
filename	varchar2	A valid trace file name	Null	No	If the value is null, the function shows all trace files. If it isn't null, the function shows the specified file.

To show the trace file, call the Amazon RDS function `rdsadmin.rdsadmin_adrci_util.show_adrci_tracefile` with the `incident_id` parameter.

```
SQL> exec :task_id := rdsadmin.rdsadmin_adrci_util.show_adrci_tracefile;
```

```
PL/SQL procedure successfully completed.
```

To list the trace file names, call the Amazon RDS procedure rdsadmin.rds_file_util.read_text_file, supplying the task ID as part of the file name.

```
SQL> SELECT * FROM TABLE(rdsadmin.rds_file_u-
til.read_text_file('BDUMP', 'dbtask-'||:task_id||'.log'))
WHERE TEXT LIKE '%/alert_%';

TEXT
--------------------------------------------------------------
  diag/rdbms/orcl_a/ORCL/trace/alert_OR-
CL.log.2020-05-28
  diag/rdbms/orcl_a/ORCL/trace/alert_OR-
CL.log.2020-05-27
  diag/rdbms/orcl_a/ORCL/trace/alert_OR-
CL.log.2020-05-26
  diag/rdbms/orcl_a/ORCL/trace/alert_OR-
CL.log.2020-05-25
```

```
   diag/rdbms/orcl_a/ORCL/trace/alert_OR-
CL.log.2020-05-24
   diag/rdbms/orcl_a/ORCL/trace/alert_OR-
CL.log.2020-05-23
   diag/rdbms/orcl_a/ORCL/trace/alert_OR-
CL.log.2020-05-22
   diag/rdbms/orcl_a/ORCL/trace/alert_OR-
CL.log.2020-05-21
   diag/rdbms/orcl_a/ORCL/trace/alert_ORCL.log

9 rows selected.
```

In the following example, you generate output for alert_ORCL.log.

```
SQL> exec :task_id := rdsadmin.rdsadmin_adrci_u-
til.show_adrci_tracefile('diag/rdbms/orcl_a/ORCL/
trace/alert_ORCL.log');

PL/SQL procedure successfully completed.
```

To read the log file, call `rdsadmin.rds_file_u-til.read_text_file`. Supply the task ID as part of the

file name. The output shows the first 10 lines of alert_ORCL.log.

```
SQL> SELECT * FROM TABLE(rdsadmin.rds_file_u-
til.read_text_file('BDUMP', 'dbtask-'||:task_id||'.log'))
WHERE ROWNUM <= 10;

TEXT
--------------------------------------------------------------
----------------------
2020-05-29 21:24:02.083 UTC [INFO ] The trace
files are being displayed.
2020-05-29 21:24:02.128 UTC [INFO ] Thu May 28
23:59:10 2020
Thread 1 advanced to log sequence 2048 (LGWR
switch)
  Current log# 3 seq# 2048 mem# 0: /rdsdbdata/db/
ORCL_A/onlinelog/o1_mf_3_hbl2p8xs_.log
Thu May 28 23:59:10 2020
Archived Log entry 2037 added for thread 1 se-
quence 2047 ID 0x5d62ce43 dest 1:
```

```
Fri May 29 00:04:10 2020
Thread 1 advanced to log sequence 2049 (LGWR
switch)
  Current log# 4 seq# 2049 mem# 0: /rdsdbdata/db/
ORCL_A/onlinelog/o1_mf_4_hbl2qgmh_.log
Fri May 29 00:04:10 2020

10 rows selected.
```

Performing miscellaneous tasks for Oracle DB instances

Following, you can find how to perform miscellaneous DBA tasks on your Amazon RDS DB instances running Oracle. To deliver a managed service experience, Amazon RDS doesn't provide shell access to DB instances, and restricts access to certain system procedures and tables that require advanced privileges.

Creating and dropping directories in the main data storage space

To create directories, use the Amazon RDS procedure rdsadmin.rdsadmin_util.create_directory. You can create up to 10,000 directories, all located in your main data storage space. To drop directories,

use the Amazon RDS procedure rdsadmin.rdsadmin_util.drop_directory.

The create_directory and drop_directory procedures have the following required parameter.

Parameter name	Data type	Default	Required	Description
p_directory_name	varchar2	—	Yes	The name of the directory.

The following example creates a new directory named PRODUCT_DESCRIPTIONS.

```
exec rdsadmin.rdsadmin_util.create_directory(p_directory_name => 'product_descriptions');
```

The data dictionary stores the directory name in uppercase. You can list the directories by querying DBA_DIRECTORIES. The system chooses the actual host pathname automatically. The following example gets the directory path for the directory named PRODUCT_DESCRIPTIONS:

```
SELECT DIRECTORY_PATH
 FROM DBA_DIRECTORIES
 WHERE DIRECTORY_NAME='PRODUCT_DESCRIP-
TIONS';

DIRECTORY_PATH
--------------------------------------
/rdsdbdata/userdirs/01
```

The master user name for the DB instance has read and write privileges in the new directory, and can grant access to other users. EXECUTE privileges are not available for directories on a DB instance. Directories are created in your main data storage space and will consume space and I/O bandwidth.

The following example drops the directory named PRODUCT_DESCRIPTIONS.

```
exec rdsadmin.rdsadmin_util.drop_directory(p_di-
rectory_name => 'product_descriptions');
```

Dropping a directory doesn't remove its contents. Because the rdsadmin.rdsadmin_util.create_directory procedure can reuse pathnames, files in dropped directories can appear in a newly created directory. Before you drop a directory, we recommend that you use UTL_FILE.FREMOVE to remove files from the directory.

Listing files in a DB instance directory

To list the files in a directory, use the Amazon RDS procedure rdsadmin.rds_file_util.listdir. The listdir procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
p_directory	varchar2	—	Yes	The name of the directory to list.

The following example lists the files in the directory named PRODUCT_DESCRIPTIONS.

```
SELECT * FROM TABLE(rdsadmin.rds_file_util.listdir(p_directory => 'PRODUCT_DESCRIPTIONS'));
```

Reading files in a DB instance directory

To read a text file, use the Amazon RDS procedure rdsadmin.rds_file_util.read_text_file. The read_text_file procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
p_directory	varchar2	—	Yes	The name of the directory that contains the file.
p_filename	varchar2	—	Yes	The name of the file to read.

The following example creates the file rice.txt in the directory PRODUCT_DESCRIPTIONS.

```
declare
 fh sys.utl_file.file_type;
begin
 fh := utl_file.fopen(loca-
tion=>'PRODUCT_DESCRIPTIONS', file-
name=>'rice.txt', open_mode=>'w');
 utl_file.put(file=>fh, buffer=>'AnyCompany
brown rice, 15 lbs');
 utl_file.fclose(file=>fh);
end;
/
```

The following example reads the file rice.txt from the directory PRODUCT_DESCRIPTIONS.

```
SELECT * FROM TABLE
  (rdsadmin.rds_file_util.read_text_file(
     p_directory => 'PRODUCT_DESCRIPTIONS',
```

```
    p_filename  => 'rice.txt'));
```

Accessing Opatch files

Opatch is an Oracle utility that enables the application and rollback of patches to Oracle software. The Oracle mechanism for determining which patches have been applied to a database is the `opatch lsinventory` command. To open service requests for Bring Your Own Licence (BYOL) customers, Oracle Support requests the `lsinventory` file and sometimes the `lsinventory_detail` file generated by Opatch.

To deliver a managed service experience, Amazon RDS doesn't provide shell access to Opatch. Instead, the Oracle DB instance automatically creates the inventory files every hour in the BDUMP directory. You have read and write access on this directory.

If you don't see your files in BDUMP, or the files are out of date, wait an hour and then try again.

The examples in this section assume that the BDUMP directory is named BDUMP. On a read replica, the BDUMP directory name is different.

The inventory files use the Amazon RDS naming convention lsinventory-*dbv*.txt and lsinventory_detail-*dbv*.txt, where *dbv* is the full name of your DB version. The lsinventory-*dbv*.txt file is available on all DB versions. The corresponding detail file is available on the following DB versions:

- 19.0.0.0, ru-2020-01.rur-2020-01.r1 or later
- 18.0.0.0, ru-2020-01.rur-2020-01.r1 or later
- 12.2.0.1, ru-2020-01.rur-2020-01.r1 or later
- 12.1.0.2, v19 or later

For example, if your DB version is 19.0.0.0.ru-2020-04.rur-2020-04.r1, then your inventory files have the following names.

```
lsinventory-19.0.0.0.ru-2020-04.rur-2020-04.r1.txt
lsinventory_detail-19.0.0.0.ru-2020-04.rur-2020-04.r1.txt
```

Ensure that you download the files that match the current version of your DB engine.

Console

To download an inventory file using the console

1. Open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Databases**.
3. Choose the name of the DB instance that has the log file that you want to view.
4. Choose the **Logs & events** tab.
5. Scroll down to the **Logs** section.
6. In the **Logs** section, search for lsinventory.
7. Select the file that you want to access, and then choose **Download**.

Managing advisor tasks

Oracle Database includes a number of advisors. Each advisor supports automated and manual tasks. You can use procedures in the rdsadmin.rdsadmin_util package to manage some advisor tasks.

The advisor task procedures are available in the following engine versions:

- Version 19.0.0.0.ru-2021-01.rur-2021-01.r1 or higher 19c versions
- Version 18.0.0.0.ru-2021-01.rur-2021-01.r1 or higher 18c versions
- Version 12.2.0.1.ru-2021-01.rur-2021-01.r1 or higher 12.2.0.1 versions

Setting parameters for advisor tasks

To set parameters for some advisor tasks, use the Amazon RDS procedure rdsadmin.rdsadmin_util.advisor_task_set_parameter. The advisor_task_set_parameter procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
p_task_name	varchar2	—	Yes	The name of the advisor task whose parameters you want to change. The following values are valid: • AUTO_STATS_ADVISOR_TASK • INDIVIDUAL_STATS_ADVISOR_TASK • SYS_AUTO_SPM_EVOLVE_TASK • SYS_AUTO_SQL_TUNING_TASK
p_parameter	varchar2	—	Yes	The name of the task parameter. To find valid parameters for an advisor task, run the following query. Substitute *p_task_name* with a valid value for p_task_name: COL PARAMETER_NAME FORMAT a30 COL PARAMETER_VALUE FORMAT a30 SELECT PARAMETER_NAME, PARAMETER_VALUE FROM DBA_ADVISOR_PARAMETERS WHERE TASK_NAME='*p_task_name*' AND PARAMETER_VALUE != 'UNUSED' ORDER BY PARAMETER_NAME;
p_value	varchar2	—	Yes	The value for a task parameter. To find valid values for task parameters, run the following query. Substitute *p_task_name* with a valid value for p_task_name: COL PARAMETER_NAME FORMAT a30

				COL PARAMETER_VALUE FORMAT a30 SELECT PARAMETER_NAME, PARAMETER_ VALUE FROM DBA_ADVISOR_PARAMETERS WHERE TASK_NAME='*p_task_name*' AND PARAMETER_VALUE != 'UNUSED' ORDER BY PARAMETER_NAME;

The following PL/SQL program sets ACCEPT_PLANS to FALSE for SYS_AUTO_SPM_EVOLVE_TASK. The SQL Plan Management automated task verifies the plans and generates a report of its findings, but does not evolve the plans automatically. You can use a report to identify new SQL plan baselines and accept them manually.

```
BEGIN
 rdsadmin.rdsadmin_util.advisor_task_set_param-
eter(
  p_task_name => 'SYS_AUTO_SPM_EVOLVE_
TASK',
  p_parameter => 'ACCEPT_PLANS',
  p_value    => 'FALSE');
END;
```

The following PL/SQL program sets EXECUTION_DAYS_TO_EXPIRE to 10 for AUTO_STATS_ADVISOR_TASK. The predefined task AUTO_STATS_ADVISOR_TASK runs automatically in the maintenance window once per day. The example sets the retention period for the task execution to 10 days.

```
BEGIN
 rdsadmin.rdsadmin_util.advisor_task_set_parameter(
  p_task_name => 'AUTO_STATS_ADVISOR_TASK',
  p_parameter => 'EXECUTION_DAYS_TO_EXPIRE',
  p_value    => '10');
END;
```

Disabling AUTO_STATS_ADVISOR_TASK

To disable AUTO_STATS_ADVISOR_TASK, use the Amazon RDS procedure rdsadmin.rdsadmin_u-

til.advisor_task_drop. The advisor_task_drop procedure accepts the following parameter.

Parameter name	Data type	Default	Required	Description
p_task_name	varchar2	—	Yes	The name of the advisor task to be disabled. The only valid value is AUTO_STATS_ADVISOR_TASK.

The following command drops AUTO_STATS_ADVISOR_TASK.

```
EXEC rdsadmin.rdsadmin_util.advisor_task_drop('AUTO_STATS_ADVISOR_TASK')
```

You can re-enabling AUTO_STATS_ADVISOR_TASK using rdsadmin.rdsadmin_util.dbms_stats_init.

Re-enabling AUTO_STATS_ADVISOR_TASK

To re-enable AUTO_STATS_ADVISOR_TASK, use the Amazon RDS procedure rdsadmin.rdsadmin_util.dbms_stats_init. The dbms_stats_init procedure takes no parameters.

The following command re-enables AUTO_STATS_ADVISOR_TASK.

```
EXEC rdsadmin.rdsadmin_util.dbms_stats_init()
```

Enabling HugePages for an Oracle DB instance

Amazon RDS for Oracle supports Linux kernel HugePages for increased database scalability. HugePages results in smaller page tables and less CPU time spent on memory management, increasing the performance of large database instances.

You can use HugePages with the following versions and editions of Oracle Database:

- 19.0.0.0, all editions
- 18.0.0.0, all editions
- 12.2.0.1, all editions
- 12.1.0.2, all editions

The use_large_pages parameter controls whether HugePages are enabled for a DB instance. The possible settings for this parameter are ONLY, FALSE, and {DBInstanceClassHugePagesDefault}. The use_large_pages parameter is set to {DBInstanceClassHugePagesDefault} in the default DB parameter group for Oracle.

To control whether HugePages are enabled for a DB instance automatically, you can use the DBInstanceClassHugePagesDefault formula variable in parameter groups. The value is determined as follows:

- For the DB instance classes mentioned in the table following, DBInstanceClassHugePagesDefault always evaluates to FALSE by default, and use_large_pages evaluates to FALSE. You can enable HugePages manually for these DB instance classes if the DB instance class has at least 14 GiB of memory.

- For DB instance classes not mentioned in the table following, if the DB instance class has less than 14 GiB of memory, DBInstanceClassHugePagesDefault always evaluates to FALSE. Also, use_large_pages evaluates to FALSE.
- For DB instance classes not mentioned in the table following, if the instance class has at least 14 GiB of memory and less than 100 GiB of memory, DBInstanceClassHugePagesDefault evaluates to TRUE by default. Also, use_large_pages evaluates to ONLY. You can disable HugePages manually by setting use_large_pages to FALSE.
- For DB instance classes not mentioned in the table following, if the instance class has at least 100 GiB of memory, DBInstanceClassHugePagesDefault always evaluates to TRUE. Also, use_large_pages evaluates to ONLY and HugePages can't be disabled.

HugePages are not enabled by default for the following DB instance classes.

DB instance class family	DB instance classes with HugePages not enabled by default
db.m5	db.m5.large
db.m4	db.m4.large, db.m4.xlarge, db.m4.2xlarge, db.m4.4xlarge, db.m4.10xlarge
db.t3	db.t3.micro, db.t3.small, db.t3.medium, db.t3.large

To enable HugePages for new or existing DB instances manually, set the `use_large_pages` parameter to `ONLY`. You can't use HugePages with Oracle Automatic Memory Management (AMM). If you set the parameter `use_large_pages` to `ONLY`, then you must also set both `memory_target` and `memory_max_target` to `0`.

You can also set the sga_target, sga_max_size, and pga_aggregate_target parameters. When you set system global area (SGA) and program global area (PGA) memory parameters, add the values together. Subtract this total from your available instance memory (DBInstanceClassMemory) to determine the free memory beyond the HugePages allocation. You must leave free memory of at least 2 GiB, or 10 percent of the total available instance memory, whichever is smaller.

After you configure your parameters, you must reboot your DB instance for the changes to take effect.

The Oracle DB instance defers changes to SGA-related initialization parameters until you reboot the instance without failover. In the Amazon RDS console, choose **Reboot** but *do not* choose **Reboot with failover**. In the AWS CLI, call the reboot-db-instance command with the --no-force-failover parameter. The DB instance does not process the SGA-

related parameters during failover or during other maintenance operations that cause the instance to restart.

The following is a sample parameter configuration for HugePages that enables HugePages manually. You should set the values to meet your needs.

```
memory_target        = 0
memory_max_target    = 0
pga_aggregate_target = {DBInstanceClassMem-
ory*1/8}
sga_target           = {DBInstanceClassMemory*3/4}
sga_max_size         = {DBInstanceClassMem-
ory*3/4}
use_large_pages      = ONLY
```

Assume the following parameters values are set in a parameter group.

```
memory_target        = IF({DBInstance-
ClassHugePagesDefault}, 0, {DBInstanceClassMem-
ory*3/4})
memory_max_target    = IF({DBInstance-
ClassHugePagesDefault}, 0, {DBInstanceClassMem-
ory*3/4})
pga_aggregate_target = IF({DBInstance-
ClassHugePagesDefault}, {DBInstanceClassMem-
ory*1/8}, 0)
sga_target           = IF({DBInstanceClassHugePages-
Default}, {DBInstanceClassMemory*3/4}, 0)
sga_max_size         = IF({DBInstance-
ClassHugePagesDefault}, {DBInstanceClassMem-
ory*3/4}, 0)
use_large_pages      = {DBInstanceClassHugePages-
Default}
```

The parameter group is used by a db.r4 DB instance class with less than 100 GiB of memory. With these parameter settings and use_large_pages set to {D-

BInstanceClassHugePagesDefault}, HugePages are enabled on the db.r4 instance.

Consider another example with following parameters values set in a parameter group.

```
memory_target        = IF({DBInstance-
ClassHugePagesDefault}, 0, {DBInstanceClassMem-
ory*3/4})
memory_max_target    = IF({DBInstance-
ClassHugePagesDefault}, 0, {DBInstanceClassMem-
ory*3/4})
pga_aggregate_target  = IF({DBInstance-
ClassHugePagesDefault}, {DBInstanceClassMem-
ory*1/8}, 0)
sga_target           = IF({DBInstanceClassHugePages-
Default}, {DBInstanceClassMemory*3/4}, 0)
sga_max_size         = IF({DBInstance-
ClassHugePagesDefault}, {DBInstanceClassMem-
ory*3/4}, 0)
use_large_pages      = FALSE
```

The parameter group is used by a db.r4 DB instance class and a db.r5 DB instance class, both with less than 100 GiB of memory. With these parameter settings, HugePages are disabled on the db.r4 and db.r5 instance.

If this parameter group is used by a db.r4 DB instance class or db.r5 DB instance class with at least 100 GiB of memory, the `FALSE` setting for `use_large_pages` is overridden and set to `ONLY`. In this case, a customer notification regarding the override is sent.

After HugePages are active on your DB instance, you can view HugePages information by enabling enhanced monitoring.

Enabling extended data types

Amazon RDS Oracle version 12c supports extended data types. With extended data types, the maximum size is 32,767 bytes for the VARCHAR2, NVARCHAR2, and RAW data types. To use extended data types, set the MAX_STRING_SIZE parameter to EXTENDED.

If you don't want to use extended data types, keep the MAX_STRING_SIZE parameter set to STANDARD (the default). When this parameter is set to STANDARD, the size limits are 4,000 bytes for the VARCHAR2 and NVARCHAR2 data types, and 2,000 bytes for the RAW data type.

You can enable extended data types on a new or existing DB instance. For new DB instances, DB instance creation time is typically longer when you enable extended data types. For existing DB instances, the DB instance is unavailable during the conversion process.

The following are considerations for a DB instance with extended data types enabled:

- When you enable extended data types for a DB instance, you can't change the DB instance back to use the standard size for data types. After a DB instance is converted to use extended data types, if you set the MAX_STRING_SIZE parameter back to S-TANDARD it results in the incompatible-parameters status.
- When you restore a DB instance that uses extended data types, you must specify a parameter group with the MAX_STRING_SIZE parameter set to EXTENDED. During restore, if you specify the default parameter group or any other parameter group with MAX_STRING_SIZE set to STANDARD it results in the incompatible-parameters status.

- We recommend that you don't enable extended data types for Oracle DB instances running on the t2.micro DB instance class.

When the DB instance status is incompatible-parameters because of the MAX_STRING_SIZE setting, the DB instance remains unavailable until you set the MAX_STRING_SIZE parameter to EXTENDED and reboot the DB instance.

Enabling extended data types for a new DB instance

To enable extended data types for a new DB instance

1. Set the MAX_STRING_SIZE parameter to EXTENDED in a parameter group.

To set the parameter, you can either create a new parameter group or modify an existing parameter group.

2. Create a new Amazon RDS Oracle DB instance, and associate the parameter group with MAX_STRING_SIZE set to EXTENDED with the DB instance.

Enabling extended data types for an existing DB instance

When you modify a DB instance to enable extended data types, the data in the database is converted to use the extended sizes. The DB instance is unavailable during the conversion. The amount of time it takes to convert the data depends on the DB instance class used by the DB instance and the size of the database.

To enable extended data types for an existing DB instance

1. Take a snapshot of the database.

If there are invalid objects in the database, Amazon RDS tries to recompile them. The conversion to

extended data types can fail if Amazon RDS can't re-compile an invalid object. The snapshot enables you to restore the database if there is a problem with the conversion. Always check for invalid objects before conversion and fix or drop those invalid objects. For production databases, we recommend testing the conversion process on a copy of your DB instance first.

2. Set the MAX_STRING_SIZE parameter to EXTENDED in a parameter group.

To set the parameter, you can either create a new parameter group or modify an existing parameter group.

3. Modify the DB instance to associate it with the parameter group with MAX_STRING_SIZE set to EXTENDED.
4. Reboot the DB instance for the parameter change to take effect.

Importing data into Oracle on Amazon RDS

How you import data into an Amazon RDS DB instance depends on the amount of data you have and the number and variety of database objects in your database. For example, you can use Oracle SQL Developer to import a simple, 20 MB database. You can use Oracle Data Pump to import complex databases, or databases that are several hundred megabytes or several terabytes in size.

You can also use AWS Database Migration Service (AWS DMS) to import data into an Amazon RDS DB instance. AWS DMS can migrate databases without downtime and, for many database engines, continue ongoing replication until you are ready to switch over to the target database. You can migrate to Oracle from either the same database engine or a different database engine using AWS DMS. If you are migrating from a different database engine, you

can use the AWS Schema Conversion Tool to migrate schema objects that are not migrated by AWS DMS.

Before you use any of these migration techniques, we recommend the best practice of taking a backup of your database. After you import the data, you can back up your Amazon RDS DB instances by creating snapshots. Later, you can restore the database from the snapshots.

Importing using Oracle SQL Developer

For small databases, you can use Oracle SQL Developer, a graphical Java tool distributed without cost by Oracle. You can install this tool on your desktop computer (Windows, Linux, or Mac) or on one of your servers. Oracle SQL Developer provides options for migrating data between two Oracle databases, or for migrating data from other databases,

such as MySQL, to Oracle. Oracle SQL Developer is best suited for migrating small databases. We recommend that you read the Oracle SQL Developer product documentation before you begin migrating your data.

After you install SQL Developer, you can use it to connect to your source and target databases. Use the **Database Copy** command on the Tools menu to copy your data to your Amazon RDS instance.

Importing using Oracle Data Pump

Oracle Data Pump is a long-term replacement for the Oracle Export/Import utilities. Oracle Data Pump is the preferred way to move large amounts of data from an Oracle installation to an Amazon RDS DB instance. You can use Oracle Data Pump for several scenarios:

- Import data from an Oracle database (either on-premises or Amazon EC2 instance) to an Amazon RDS for Oracle DB instance.
- Import data from an RDS for Oracle DB instance to an Oracle database (either on-premises or Amazon EC2 instance).
- Import data between RDS for Oracle DB instances (for example, to migrate data from EC2-Classic to VPC).

When you import data with Oracle Data Pump, you must transfer the dump file that contains the data from the source database to the target database. You can transfer the dump file using an Amazon S3 bucket or by using a database link between the two databases.

The following are best practices for using Oracle Data Pump to import data into an Amazon RDS for Oracle DB instance:

- Perform imports in schema or table mode to import specific schemas and objects.
- Limit the schemas you import to those required by your application.
- Do not import in full mode.

Because Amazon RDS for Oracle does not allow access to SYS or SYSDBA administrative users, importing in full mode, or importing schemas for Oracle-maintained components, might damage the Oracle data dictionary and affect the stability of your database.

- When loading large amounts of data, transfer the dump file to the target Amazon RDS for Oracle DB instance, take a DB snapshot of your instance, and then test the import to verify that it succeeds. If database components are invalidated, you can delete the DB instance and re-create it from the DB snapshot. The restored DB instance includes any

dump files staged on the DB instance when you took the DB snapshot.

- Do not import dump files that were created using the Oracle Data Pump export parameters TRANSPORT_TABLESPACES, TRANSPORTABLE, or TRANSPORT_FULL_CHECK. Amazon RDS for Oracle DB instances do not support importing these dump files.

The examples in this section show one way to import data into an Oracle database, but there are many ways to do so with Oracle Data Pump.

The examples in this section use the DBMS_DATAPUMP package. The same tasks can be accomplished by using the Oracle Data Pump command line utilities impdp and expdp. You can install these utilities on a remote host as part of an Oracle Client installation, including Oracle Instant Client.

Importing data with Oracle Data Pump and an Amazon S3 bucket

The following import process uses Oracle Data Pump and an Amazon S3 bucket. The process exports data on the source database using the Oracle DBMS_DATAPUMP package and puts the dump file in an Amazon S3 bucket. It then downloads the dump file from the Amazon S3 bucket to the DATA_PUMP_DIR directory on the target Amazon RDS for Oracle DB instance. The final step imports the data from the copied dump file into the Amazon RDS for Oracle DB instance using the package DBMS_DATAPUMP.

The process has the following requirements:

- You must have an Amazon S3 bucket available for file transfers, and the Amazon S3 bucket must be in the same AWS Region as the DB instance.
- The object that you upload into the Amazon S3 bucket must be 5 TB or less.

- You must prepare the Amazon S3 bucket for Amazon RDS integration by following the instructions in Prerequisites for Amazon RDS for Oracle integration with Amazon S3.
- You must ensure that you have enough storage space to store the dump file on the source instance and the target DB instance.

The import process using Oracle Data Pump and an Amazon S3 bucket has the following steps.

Step 1: Grant privileges to the user on the Amazon RDS target instance

To grant privileges to the user on the RDS target instance, take the following steps:

1. Use SQL Plus or Oracle SQL Developer to connect to the Amazon RDS target Oracle DB instance into which the data will be imported. Connect as the Amazon RDS master user.

2. Create the required tablespaces before you import the data.
3. If the user account into which the data is imported doesn't exist, create the user account and grant the necessary permissions and roles. If you plan to import data into multiple user schemas, create each user account and grant the necessary privileges and roles to it.

For example, the following commands create a new user and grant the necessary permissions and roles to import the data into the user's schema.

```
CREATE USER schema_1 IDENTIFIED BY <password>;
GRANT CREATE SESSION, RESOURCE TO schema_1;
ALTER USER schema_1 QUOTA 100M ON users;
```

This example grants the new user the CREATE SESSION privilege and the RESOURCE role. Additional privileges and roles might be required depending on the database objects that you import.

Step 2: Use DBMS_DATAPUMP to create a dump file

Use SQL Plus or Oracle SQL Developer to connect to the source Oracle instance with an administrative user. If the source database is an Amazon RDS for Oracle DB instance, connect with the Amazon RDS master user. Next, use the Oracle Data Pump utility to create a dump file.

The following script creates a dump file named *sample.dmp* in the DATA_PUMP_DIR directory that contains the *SCHEMA_1* schema. Replace *SCHEMA_1* with the name of the schema you want to export.

```
DECLARE
 v_hdnl NUMBER;
BEGIN
 v_hdnl := DBMS_DATAPUMP.OPEN( operation
=> 'EXPORT', job_mode => 'SCHEMA', job_name=
>null);
```

```
 DBMS_DATAPUMP.ADD_FILE(
  handle   => v_hdnl,
  filename  => 'sample.dmp',
  directory => 'DATA_PUMP_DIR',
  filetype  => dbms_datapump.ku$_file_type_
dump_file);
 DBMS_DATAPUMP.ADD_FILE(
  handle   => v_hdnl,
  filename  => 'sample_exp.log',
  directory => 'DATA_PUMP_DIR',
  filetype  => dbms_datapump.ku$_file_type_log_-
file);
 DBMS_DATAPUMP.METADATA_FIL-
TER(v_hdnl,'SCHEMA_EXPR','IN ("SCHEMA_1")');
 DBMS_DATAPUMP.START_JOB(v_hdnl);
END;
/
```

Step 3: Upload the dump file to your Amazon S3 bucket

Upload the dump file to the Amazon S3 bucket.

Use the Amazon RDS procedure rdsadmin.rdsadmin_s3_tasks.upload_to_s3 to copy the dump file to the Amazon S3 bucket. The following example uploads all of the files from the *DATA_PUMP_DIR* directory to an Amazon S3 bucket named *mys3bucket*.

```
SELECT rdsadmin.rdsadmin_s3_tasks.upload_
to_s3(
 p_bucket_name    => 'mys3bucket',
 p_directory_name => 'DATA_PUMP_DIR')
AS TASK_ID FROM DUAL;
```

The SELECT statement returns the ID of the task in a VARCHAR2 data type.

Step 4: Copy the exported dump file from the Amazon S3 bucket to the target DB instance

Use SQL Plus or Oracle SQL Developer to connect to the Amazon RDS target Oracle DB instance. Next,

use the Amazon RDS procedure rdsadmin.rdsadmin_s3_tasks.download_from_s3 to copy the dump file from the Amazon S3 bucket to the target DB instance. The following example downloads all of the files from an Amazon S3 bucket named *mys3bucket* to the *DATA_PUMP_DIR* directory.

```
SELECT rdsadmin.rdsadmin_s3_tasks.download_from_s3(
 p_bucket_name    => 'mys3bucket',
 p_directory_name => 'DATA_PUMP_DIR')
AS TASK_ID FROM DUAL;
```

The SELECT statement returns the ID of the task in a VARCHAR2 data type.

Step 5: Use DBMS_DATAPUMP to import the data file on the target DB instance

Use Oracle Data Pump to import the schema in the DB instance. Additional options such as METADATA_REMAP might be required.

Connect to the DB instance with the Amazon RDS master user account to perform the import.

```
DECLARE
 v_hdnl NUMBER;
BEGIN
 v_hdnl := DBMS_DATAPUMP.OPEN(
  operation => 'IMPORT',
  job_mode  => 'SCHEMA',
  job_name  => null);
 DBMS_DATAPUMP.ADD_FILE(
  handle    => v_hdnl,
  filename  => 'sample_copied.dmp',
  directory => 'DATA_PUMP_DIR',
  filetype  => dbms_datapump.ku$_file_type_
dump_file);
 DBMS_DATAPUMP.ADD_FILE(
  handle    => v_hdnl,
  filename  => 'sample_imp.log',
  directory => 'DATA_PUMP_DIR',
```

```
  filetype => dbms_datapump.ku$_file_type_log_-
file);
 DBMS_DATAPUMP.METADATA_FIL-
TER(v_hdnl,'SCHEMA_EXPR','IN ("SCHEMA_1")');
 DBMS_DATAPUMP.START_JOB(v_hdnl);
END;
/
```

You can verify the data import by viewing the user's tables on the DB instance. For example, the following query returns the number of tables for *SCHEMA_1*.

```
SELECT COUNT(*) FROM DBA_TABLES WHERE
OWNER='SCHEMA_1';
```

Step 6: Clean up

After the data has been imported, you can delete the files that you don't want to keep. You can list the files in the DATA_PUMP_DIR using the following command.

```
SELECT * FROM TABLE(rdsadmin.rds_file_util.list-
dir('DATA_PUMP_DIR')) ORDER BY MTIME;
```

To delete files in the DATA_PUMP_DIR that you no longer require, use the following command.

```
EXEC UTL_FILE.FREMOVE('DATA_PUMP_DIR','<file
name>');
```

For example, the following command deletes the file named "sample_copied.dmp".

```
EXEC UTL_FILE.FREMOVE('DATA_PUMP_DIR','sam-
ple_copied.dmp');
```

Importing data with Oracle Data Pump and a database link

The following import process uses Oracle Data Pump and the Oracle DBMS_FILE_TRANSFER package. The process connects to a source Oracle instance, which can be an on-premises or Amazon EC2 instance, or an Amazon RDS for Oracle

DB instance. The process then exports data using the DBMS_DATAPUMP package. Next, it uses the DBMS_FILE_TRANSFER.PUT_FILE method to copy the dump file from the Oracle instance to the DATA_PUMP_DIR directory on the target Amazon RDS for Oracle DB instance that is connected using a database link. The final step imports the data from the copied dump file into the Amazon RDS for Oracle DB instance using the DBMS_DATAPUMP package.

The process has the following requirements:

- You must have execute privileges on the DBMS_FILE_TRANSFER and DBMS_DATA-PUMP packages.
- You must have write privileges to the DATA_PUMP_DIR directory on the source DB instance.

- You must ensure that you have enough storage space to store the dump file on the source instance and the target DB instance.

The import process using Oracle Data Pump and the DBMS_FILE_TRANSFER package has the following steps.

Step 1: Grant privileges to the user on the Amazon RDS target instance

To grant privileges to the user on the RDS target instance, take the following steps:

1. Use SQL Plus or Oracle SQL Developer to connect to the Amazon RDS target Oracle DB instance into which the data will be imported. Connect as the Amazon RDS master user.
2. Create the required tablespaces before you import the data.
3. If the user account into which the data is imported doesn't exist, create the user account

and grant the necessary permissions and roles. If you plan to import data into multiple user schemas, create each user account and grant the necessary privileges and roles to it.

For example, the following commands create a new user and grant the necessary permissions and roles to import the data into the user's schema.

```
CREATE USER schema_1 IDENTIFIED BY <password>;
GRANT CREATE SESSION, RESOURCE TO schema_1;
ALTER USER schema_1 QUOTA 100M ON users;
```

This example grants the new user the CREATE SESSION privilege and the RESOURCE role. Additional privileges and roles might be required depending on the database objects that you import.

Step 2: Grant privileges to the user on the source database

Use SQL*Plus or Oracle SQL Developer to connect to the Oracle instance that contains the data to be imported. If necessary, create a user account and grant the necessary permissions.

The following commands create a new user and grant the necessary permissions.

```
CREATE USER export_user IDENTIFIED BY
<password>;
GRANT CREATE SESSION, CREATE TABLE, CREATE
DATABASE LINK TO export_user;
ALTER USER export_user QUOTA 100M ON users;
GRANT READ, WRITE ON DIRECTORY
data_pump_dir TO export_user;
GRANT SELECT_CATALOG_ROLE TO export_user;
GRANT EXECUTE ON DBMS_DATAPUMP TO
export_user;
GRANT EXECUTE ON DBMS_FILE_TRANSFER TO
export_user;
```

Step 3: Use DBMS_DATAPUMP to create a dump file

Use SQL Plus or Oracle SQL Developer to connect to the source Oracle instance with an administrative user or with the user you created in step 2. If the source database is an Amazon RDS for Oracle DB instance, connect with the Amazon RDS master user. Next, use the Oracle Data Pump utility to create a dump file.

The following script creates a dump file named *sample.dmp* in the DATA_PUMP_DIR directory.

```
DECLARE
 v_hdnl NUMBER;
BEGIN
 v_hdnl := DBMS_DATAPUMP.OPEN(
  operation => 'EXPORT',
  job_mode  => 'SCHEMA',
  job_name  => null);
 DBMS_DATAPUMP.ADD_FILE(
```

```
  handle   => v_hdnl,
  filename  => 'sample.dmp',
  directory => 'DATA_PUMP_DIR',
  filetype  => dbms_datapump.ku$_file_type_
dump_file);
 DBMS_DATAPUMP.ADD_FILE(
  handle   => v_hdnl,
  filename  => 'sample_exp.log',
  directory => 'DATA_PUMP_DIR',
  filetype  => dbms_datapump.ku$_file_type_log_-
file);
 DBMS_DATAPUMP.METADATA_FIL-
TER(v_hdnl,'SCHEMA_EXPR','IN ("SCHEMA_1")');
 DBMS_DATAPUMP.START_JOB(v_hdnl);
END;
/
```

Step 4: Create a database link to the target DB instance

Create a database link between your source instance and your target DB instance. Your local Oracle instance must have network connectivity to the DB instance in order to create a database link and to transfer your export dump file.

Perform this step connected with the same user account as the previous step.

If you are creating a database link between two DB instances inside the same VPC or peered VPCs, the two DB instances should have a valid route between them. The security group of each DB instance must allow ingress to and egress from the other DB instance. The security group inbound and outbound rules can refer to security groups from the same VPC or a peered VPC.

The following command creates a database link named `to_rds` that connects to the Amazon RDS master user at the target DB instance.

```
CREATE DATABASE LINK to_rds
 CONNECT TO <master_user_account> IDENTIFIED
BY <password>
 USING '(DESCRIPTION=(ADDRESS=(PROTOCOL-
=TCP)(HOST=<dns or ip address of remote db>)
(PORT=<listener port>))(CONNECT_DATA=(SID=
<remote SID>)))';
```

Step 5: Use DBMS_FILE_TRANSFER to copy the exported dump file to the target DB instance

Use DBMS_FILE_TRANSFER to copy the dump file from the source database instance to the target DB instance. The following script copies a dump file named sample.dmp from the source instance to a target database link named *to_rds* (created in the previous step).

```
BEGIN
 DBMS_FILE_TRANSFER.PUT_FILE(
  source_directory_object      => 'DATA_PUMP_DIR',
  source_file_name             => 'sample.dmp',
```

```
  destination_directory_object =>
'DATA_PUMP_DIR',
  destination_file_name      => 'sample_
copied.dmp',
  destination_database       => 'to_rds' );
END;
/
```

Step 6: Use DBMS_DATAPUMP to import the data file to the target DB instance

Use Oracle Data Pump to import the schema in the DB instance. Additional options such as METADATA_REMAP might be required.

Connect to the DB instance with the Amazon RDS master user account to perform the import.

```
DECLARE
 v_hdnl NUMBER;
BEGIN
 v_hdnl := DBMS_DATAPUMP.OPEN(
```

```
  operation => 'IMPORT',
  job_mode  => 'SCHEMA',
  job_name  => null);
 DBMS_DATAPUMP.ADD_FILE(
  handle    => v_hdnl,
  filename  => 'sample_copied.dmp',
  directory => 'DATA_PUMP_DIR',
  filetype  => dbms_datapump.ku$_file_type_
dump_file );
 DBMS_DATAPUMP.ADD_FILE(
  handle    => v_hdnl,
  filename  => 'sample_imp.log',
  directory => 'DATA_PUMP_DIR',
  filetype  => dbms_datapump.ku$_file_type_log_-
file);
 DBMS_DATAPUMP.METADATA_FIL-
TER(v_hdnl,'SCHEMA_EXPR','IN ("SCHEMA_1")');
 DBMS_DATAPUMP.START_JOB(v_hdnl);
END;
/
```

You can verify the data import by viewing the user's tables on the DB instance. For example, the following query returns the number of tables for *schema_1*.

```
SELECT COUNT(*) FROM DBA_TABLES WHERE
OWNER='SCHEMA_1';
```

Step 7: Clean up

After the data has been imported, you can delete the files that you don't want to keep. You can list the files in DATA_PUMP_DIR using the following command.

```
SELECT * FROM TABLE(rdsadmin.rds_file_util.list-
dir('DATA_PUMP_DIR')) ORDER BY MTIME;
```

To delete files in DATA_PUMP_DIR that you no longer require, use the following command.

```
EXEC UTL_FILE.FREMOVE('DATA_PUMP_DIR','<file
name>');
```

For example, the following command deletes the file named "sample_copied.dmp".

```
EXEC UTL_FILE.FREMOVE('DATA_PUMP_DIR','sample_copied.dmp');
```

Oracle Export/Import utilities

The Oracle Export/Import utilities are best suited for migrations where the data size is small and data types such as binary float and double are not required. The import process creates the schema objects so you do not need to run a script to create them beforehand, making this process well suited for databases with small tables. The following example demonstrates how these utilities can be used to export and import specific tables.

Export the tables from the source database using the command below. Substitute username/password as appropriate.

```
exp cust_dba@ORCL FILE=exp_file.dmp TABLES=(tab1,tab2,tab3) LOG=exp_file.log
```

The export process creates a binary dump file that contains both the schema and data for the specified tables. Now this schema and data can be imported into a target database using the command:

```
imp cust_dba@targetdb FROMUSER=cust_schema TOUSER=cust_schema \
TABLES=(tab1,tab2,tab3) FILE=exp_file.dmp LOG=imp_file.log
```

There are other variations of the Export and Import commands that might be better suited to your needs. See Oracle's documentation for full details.

Oracle SQL*Loader

Oracle SQL*Loader is well suited for large databases that have a limited number of objects in them. Since the process involved in exporting from a source database and loading to a target database is very specific to the schema, the following example creates the sample schema objects, exports from a source, and then loads it into a target database.

1. Create a sample source table using the command below.

```
CREATE TABLE customer_0 TABLESPACE users
 AS (SELECT ROWNUM id, o.*
   FROM  ALL_OBJECTS o, ALL_OBJECTS x
   WHERE  ROWNUM <= 1000000);
```

2. On the target Amazon RDS instance, create a destination table that is used to load the data. The clause `WHERE 1=2` ensures that

you copy the structure of ALL_OBJECTS, but don't copy any of the rows.

```
CREATE TABLE customer_1 TABLESPACE users
 AS (SELECT 0 AS ID, OWNER, OBJECT_NAME,
CREATED
   FROM  ALL_OBJECTS
   WHERE  1=2);
```

3. The data is exported from the source database to a flat file with delimiters. This example uses SQL*Plus for this purpose. For your data, you will likely need to generate a script that does the export for all the objects in the database.

```
ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY/
MM/DD HH24:MI:SS'

SET LINESIZE 800 HEADING OFF FEEDBACK OFF
ARRAY 5000 PAGESIZE 0
SPOOL customer_0.out
SET MARKUP HTML PREFORMAT ON
SET COLSEP ','
```

```
SELECT id, owner, object_name, created
FROM  customer_0;
SPOOL OFF
```

4. You need to create a control file to describe the data. Again, depending on your data, you need to build a script that does this step.

```
cat << EOF > sqlldr_1.ctl
load data
infile customer_0.out
into table customer_1
APPEND
fields terminated by "," optionally enclosed by '"'
(
 id        POSITION(01:10)   INTEGER EXTERNAL,
 owner     POSITION(12:41)   CHAR,
 object_name  POSITION(43:72)   CHAR,
 created    POSITION(74:92)   date "YYYY/MM/DD
HH24:MI:SS"
)
```

If needed, copy the files generated by the preceding code to a staging area, such as an Amazon EC2 instance.

5. Finally, import the data using SQL*Loader with the appropriate username and password for the target database.

```
sqlldr cust_dba@targetdb CONTROL=sqlldr_1.ctl
BINDSIZE=10485760 READSIZE=10485760
ROWS=1000
```

Oracle materialized views

You can also make use of Oracle materialized view replication to migrate large datasets efficiently. Replication allows you to keep the target tables in sync with the source on an ongoing basis, so the actual cutover to Amazon RDS can be done later, if needed. The replication is set up using a database link from the Amazon RDS instance to the source database.

One requirement for materialized views is to allow access from the target database to the source database. In the following example, access rules were enabled on the source database to allow the Amazon RDS target database to connect to the source over SQLNet.

1. Create a user account on both source and Amazon RDS target instances that can authenticate with the same password.

```
CREATE USER dblink_user IDENTIFIED BY <password>
 DEFAULT TABLESPACE users
 TEMPORARY TABLESPACE temp;

GRANT CREATE SESSION TO dblink_user;

GRANT SELECT ANY TABLE TO dblink_user;

GRANT SELECT ANY DICTIONARY TO dblink_user;
```

2. Create a database link from the Amazon RDS target instance to the source instance using the newly created dblink_user.

```
CREATE DATABASE LINK remote_site
 CONNECT TO dblink_user IDENTIFIED BY
<password>
 USING '(description=(address=(protocol=tcp)
(host=<myhost>)    (port=<listener port>)) (connec-
t_data=(sid=<sourcedb sid>)))';
```

3. Test the link:

```
SELECT * FROM V$INSTANCE@remote_site;
```

4. Create a sample table with primary key and materialized view log on the source instance.

```
CREATE TABLE customer_0 TABLESPACE users
 AS (SELECT ROWNUM id, o.*
   FROM   ALL_OBJECTS o, ALL_OBJECTS x
   WHERE  ROWNUM <= 1000000);

ALTER TABLE customer_0 ADD CONSTRAINT pk_
customer_0 PRIMARY KEY (id) USING INDEX;
```

```
CREATE MATERIALIZED VIEW LOG ON cus-
tomer_0;
```

5. On the target Amazon RDS instance, create a materialized view.

```
CREATE MATERIALIZED VIEW customer_0
 BUILD IMMEDIATE REFRESH FAST
 AS (SELECT *
   FROM  cust_dba.customer_0@remote_site);
```

6. On the target Amazon RDS instance, refresh the materialized view.

```
EXEC DBMS_MV.REFRESH('CUSTOMER_0', 'f');
```

7. Drop the materialized view and include the PRESERVE TABLE clause to retain the materialized view container table and its contents.

```
DROP MATERIALIZED VIEW customer_0 PRE-
SERVE TABLE;
```

The retained table has the same name as the dropped materialized view.

Working with Oracle replicas for Amazon RDS

To configure replication between Oracle DB instances, you can create replica databases.

Overview of Oracle replicas

An Oracle *replica* database is either mounted or read-only. An Oracle replica in read-only mode is called a *read replica*. An Oracle replica in mounted mode is called a *mounted replica*.

Read-only and mounted replicas

When creating or modifying an Oracle replica, you can place it in either of the following modes:

- Read-only. This is the default. Active Data Guard transmits and applies changes from the source database to all read replica databases.

You can create up to five read replicas from one source DB instance.

- Mounted. In this case, replication uses Oracle Data Guard, but the replica database doesn't accept user connections. The primary use for mounted replicas is cross-Region disaster recovery.

A mounted replica can't serve a read-only workload. The mounted replica deletes archived redo log files after it applies them, regardless of the archived log retention policy.

You can create a combination of mounted and read-only DB replicas for the same source DB instance. You can change a read-only replica to mounted mode, or change a mounted replica to read-only mode. In either case, the Oracle database preserves the archived log retention setting.

Outages during replication

When you create an Oracle replica, no outage occurs for the source DB instance. Amazon RDS takes a snapshot of the source DB instance. This snapshot becomes the replica. Amazon RDS sets the necessary parameters and permissions for the source DB and replica without service interruption. Similarly, if you delete a replica, no outage occurs.

Replica requirements for Oracle

Before creating an Oracle replica, check the following requirements.

Version and licensing requirements for Oracle replicas

Before creating an Oracle replica, check the version and licensing requirements:

- If the replica is in read-only mode, make sure that you have an Active Data Guard license.

If you place the replica in mounted mode, you don't need an Active Data Guard license. Only the Oracle DB engine supports mounted replicas.

- Oracle replicas are only available on the Oracle Enterprise Edition (EE) engine.
- Oracle replicas are available for Oracle version 12.1.0.2.v10 and higher 12.1 versions, for all 12.2 versions, for all 18c versions, and for all 19c versions.
- Oracle replicas are available for DB instances only on the EC2-VPC platform.
- Oracle replicas are available for DB instances running only on DB instance classes with two or more vCPUs. A source DB instance can't use the db.t3.micro instance class.
- The Oracle DB engine version of the source DB instance and all of its replicas must be the same. Amazon RDS upgrades the replicas immediately after upgrading the source DB in-

stance, regardless of a replica's maintenance window. For major version upgrades of cross-Region replicas, Amazon RDS automatically does the following:

- Generates an option group for the target version.
- Copies all options and option settings from the original option group to the new option group.
- Associates the upgraded cross-Region replica with the new option group.

Option requirements for Oracle replicas

Before creating a replica for Oracle, check the requirements for option groups:

- If your Oracle replica is in the same AWS Region as its source DB instance, make sure that it belongs to the same option group as the source DB instance. Modifications to the source option group or source option group

membership propagate to replicas. These changes are applied to the replicas immediately after they are applied to the source DB instance, regardless of the replica's maintenance window.

- When you create an Oracle cross-Region replica, Amazon RDS creates a dedicated option group for it.

You can't remove an Oracle cross-Region replica from its dedicated option group. No other DB instances can use the dedicated option group for an Oracle cross-Region replica.

You can only add or remove the following nonreplicated options from a dedicated option group:

- NATIVE_NETWORK_ENCRYPTION
- OEM
- OEM_AGENT
- SSL

To add other options to an Oracle cross-Region replica, add them to the source DB instance's op-

tion group. The option is also installed on all of the source DB instance's replicas. For licensed options, make sure that there are sufficient licenses for the replicas.

When you promote an Oracle cross-Region replica, the promoted replica behaves the same as other Oracle DB instances, including the management of its options. You can promote a replica explicitly or implicitly by deleting its source DB instance.

Miscellaneous requirements for Oracle replicas

Before creating an Oracle replica, check the following miscellaneous requirements:

- If a DB instance is a source for one or more cross-Region replicas, the source DB retains its archived redo logs until they are applied on all cross-Region replicas. The archived redo logs might result in increased storage consumption.

- A logon trigger on a primary instance must permit access to the RDS_DATAGUARD user and to any user whose AUTHENTICATED_IDENTITY value is RDS_DATAGUARD or rdsdb. Also, the trigger must not set the current schema for the RDS_DATAGUARD user.
- To avoid disrupting RDS automation, system triggers must permit specific users to log on to the primary and replica database. System triggers include DDL, logon, and database role triggers. We recommend that you add code to your triggers to exclude the users listed in the following sample code:

```
-- Determine who the user is
SELECT SYS_CONTEXT('USERENV','AUTHENTICATED_IDENTITY') INTO CURRENT_USER FROM DUAL;
-- The following users should always be able to login to either the Primary or Replica
```

```
IF CURRENT_USER IN ('master_user', 'SYS', 'SYS-
TEM', 'RDS_DATAGUARD', 'rdsdb') THEN
RETURN;
END IF;
```

- To avoid blocking connections from the Data Guard broker process, don't enable restricted sessions.
- Block change tracking is supported for read-only replicas, but not for mounted replicas. You can change a mounted replica to a read-only replica, and then enable block change tracking.

Preparing to create an Oracle replica

Before you can begin using your replica, perform the following tasks.

Enabling automatic backups

Before a DB instance can serve as a source DB instance, make sure to enable automatic backups on the source DB instance.

Enabling force logging mode

We recommend that you enable force logging mode. In force logging mode, the Oracle database writes redo records even when NOLOGGING is used with data definition language (DDL) statements.

To enable force logging mode

1. Log in to your Oracle database using a client tool such as SQL Developer.
2. Enable force logging mode by running the following procedure.

```
exec rdsadmin.rdsadmin_util.force_logging(p_enable => true);
```

Changing your logging configuration

If you want to change your logging configuration, we recommend that you complete the changes before making a DB instance the source for replicas. Also, we recommend that you not modify the logging configuration after you create the replicas. Modifications can cause the online redo logging configuration to get out of sync with the standby logging configuration.

Modify the logging configuration for a DB instance by using the Amazon RDS procedures rdsadmin.rdsadmin_util.add_logfile and rdsadmin.rdsadmin_util.drop_logfile.

Setting the MAX_STRING_SIZE parameter

Before you create an Oracle replica, ensure that the setting of the MAX_STRING_SIZE parameter is the same on the source DB instance and the replica. You can do this by associating them with the same parameter group. If you have different parameter

groups for the source and the replica, you can set MAX_STRING_SIZE to the same value.

Planning compute and storage resources

Ensure that the source DB instance and its replicas are sized properly, in terms of compute and storage, to suit their operational load. If a replica reaches compute, network, or storage resource capacity, the replica stops receiving or applying changes from its source. Amazon RDS for Oracle doesn't intervene to mitigate high replica lag between a source DB instance and its replicas. You can modify the storage and CPU resources of a replica independently from its source and other replicas.

Creating an Oracle replica in mounted mode

By default, Oracle replicas are read-only. To create a replica in mounted mode, use the console, the AWS CLI, or the RDS API.

Console

To create a mounted replica from a source Oracle DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console.
2. In the navigation pane, choose **Databases**.
3. Choose the Oracle DB instance that you want to use as the source for a mounted replica.
4. For **Actions**, choose **Create replica**.
5. For **Replica mode**, choose **Mounted**.
6. Choose the settings that you want to use. For **DB instance identifier**, enter a name for the read replica. Adjust other settings as needed.
7. For **Regions**, choose the Region where the mounted replica will be launched.
8. Choose your instance size and storage type. We recommend that you use the same DB

instance class and storage type as the source DB instance for the read replica.

9. For **Multi-AZ deployment**, choose **Create a standby instance** to create a standby of your replica in another Availability Zone for failover support for the mounted replica. Creating your mounted replica as a Multi-AZ DB instance is independent of whether the source database is a Multi-AZ DB instance.
10. Choose the other settings that you want to use.
11. Choose **Create replica**.

In the **Databases** page, the mounted replica has the role Replica.

Modifying the Oracle replica mode

To change the replica mode of an existing replica, use the console, AWS CLI, or RDS API. When you change to mounted mode, the replica disconnects

all active connections. When you change to read-only mode, Amazon RDS initializes Active Data Guard.

The change operation can take a few minutes. During the operation, the DB instance status changes to **modifying**.

Console

To change the replica mode of an Oracle replica from mounted to read-only

1. Sign in to the AWS Management Console and open the Amazon RDS console.
2. In the navigation pane, choose **Databases**.
3. Choose the mounted replica database.
4. Choose **Modify**.
5. For **Replica mode**, choose **Read-only**.
6. Choose the other settings that you want to change.
7. Choose **Continue**.

8. For **Scheduling of modifications**, choose **Apply immediately**.
9. Choose **Modify DB instance**.

Troubleshooting Oracle replicas

This section describes possible replication problems and solutions.

Replication lag

To monitor replication lag in Amazon CloudWatch, view the Amazon RDS `ReplicaLag` metric.

If replication lag is too long, query the following views:

- `V$ARCHIVED_LOG` – Shows which commits have been applied to the read replica.
- `V$DATAGUARD_STATS` – Shows a detailed breakdown of the components that make up the `replicaLag` metric.

- V$DATAGUARD_STATUS – Shows the log output from Oracle's internal replication processes.

Replication failure after adding or modifying triggers

If you add or modify any triggers, and if replication fails afterward, the problem may be the triggers. Ensure that the trigger excludes the following user accounts, which are required by RDS for replication:

- User accounts with administrator privileges
  - SYS
  - SYSTEM
  - RDS_DATAGUARD
  - rdsdb

Amazon S3 integration

You can transfer files between an Amazon RDS for Oracle DB instance and an Amazon S3 bucket. You can use Amazon S3 integration with Oracle features such as Data Pump. For example, you can download Data Pump files from Amazon S3 to the DB instance host.

Prerequisites for Amazon RDS for Oracle integration with Amazon S3

For Amazon RDS for Oracle to integrate with Amazon S3, the Amazon RDS DB instance must have access to an Amazon S3 bucket. Prepare for the integration as follows:

1. Create an AWS Identity and Access Management (IAM) policy with the permissions required to transfer files from your bucket to RDS.

To create the policy, you need the Amazon Resource Name (ARN) value for your bucket. Also, RDS for Oracle supports SSE-KMS and SSE-S3 encryption. If your bucket is encrypted, you need the ARN for your AWS KMS key.

2. Create an IAM role, attach your new policy to it, and then attach the role to your Oracle DB instance. The status of the DB instance must be available.

The Amazon VPC used by your DB instance doesn't need to provide access to the Amazon S3 endpoints.

Console

To create an IAM policy to allow Amazon RDS access to an Amazon S3 bucket

1. Open the IAM Management Console.
2. Under **Access management**, choose **Policies**.
3. Choose **Create policy**.

4. On the **Visual editor** tab, choose **Choose a service**, and then choose **S3**.
5. For **Actions**, choose **Expand all**, and then choose the bucket permissions and object permissions required to transfer files from an Amazon S3 bucket to Amazon RDS. For example, do the following:
    - Expand **List**, and then select **ListBucket**.
    - Expand **Read**, and then select **GetObject**.
    - Expand **Write**, and then select **PutObject**.

Object permissions are permissions for object operations in Amazon S3, and must be granted for objects in a bucket, not the bucket itself.

6. Choose **Resources**, and choose **Add ARN** for **bucket**.
7. In the **Add ARN(s)** dialog box, provide the details about your resource, and choose **Add**.

Specify the Amazon S3 bucket to allow access to. For instance, to allow Amazon RDS to access the Amazon S3 bucket named example-bucket, set the ARN value to arn:aws:s3:::example-bucket.

8. If the **object** resource is listed, choose **Add ARN** for **object**.
9. In the **Add ARN(s)** dialog box, provide the details about your resource.

For the Amazon S3 bucket, specify the Amazon S3 bucket to allow access to. For the object, you can choose **Any** to grant permissions to any object in the bucket.

10. (Optional) Choose **Add additional permissions** to add resources to the policy. For example, do the following:
    - If your bucket is encrypted with a custom KMS key, select **KMS** for the service. Select **Encrypt**, **ReEncrypt**, **Decrypt**, **DescribeKey**, and **Gener-**

ateDataKey for actions. Enter the ARN of your custom key as the resource.

- If you want Amazon RDS to access to access other buckets, add the ARNs for these buckets. Optionally, you can also grant access to all buckets and objects in Amazon S3.

11. Choose **Next: Tags** and then **Next: Review**.
12. For **Name**, enter a name for your IAM policy, for example rds-s3-integration-policy. You use this name when you create an IAM role to associate with your DB instance. You can also add an optional **Description** value.
13. Choose **Create policy**.

To create an IAM role to allow Amazon RDS access to an Amazon S3 bucket

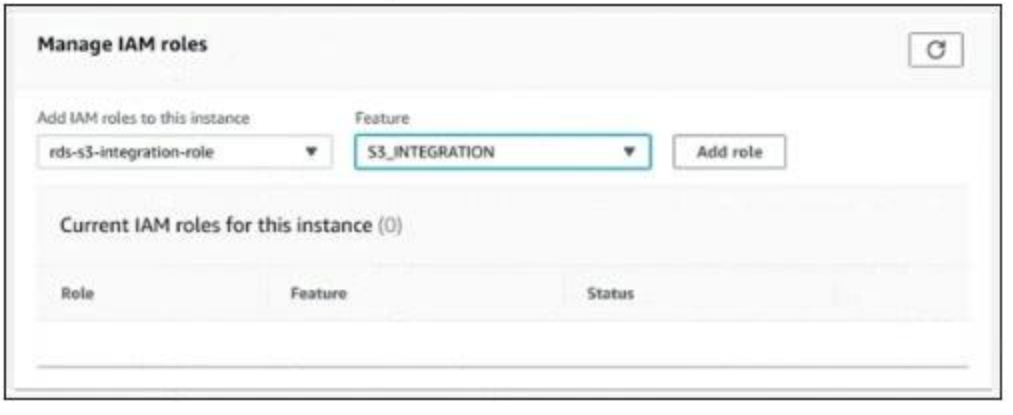
1. In the navigation pane, choose **Roles**.
2. Choose **Create role**.
3. For **AWS service**, choose **RDS**.

4. For **Select your use case**, choose **RDS – Add Role to Database**.
5. Choose **Next: Permissions**.
6. For **Search** under **Attach permissions policies**, enter the name of the IAM policy you created, and choose the policy when it appears in the list.
7. Choose **Next: Tags** and then **Next: Review**.
8. Set **Role name** to a name for your IAM role, for example rds-s3-integration-role. You can also add an optional **Role description** value.
9. Choose **Create role**.

To associate your IAM role with your DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console.
2. Choose **Databases** from the navigation pane.
3. If your database instance is unavailable, choose **Actions** and then **Start**. When the instance status shows **Started**, go to the next step.

4. Choose the Oracle DB instance name to display its details.
5. On the **Connectivity & security** tab, in the **Manage IAM roles** section, choose the role to add under **Add IAM roles to this instance**.
6. For **Feature**, choose **S3_INTEGRATION**.



7. Choose **Add role**.

Adding the Amazon S3 integration option

To use Amazon RDS for Oracle Integration with Amazon S3, your Amazon RDS for Oracle DB instance must be associated with an option group that includes the S3_INTEGRATION option.

Console

To configure an option group for Amazon S3 integration

1. Create a new option group or identify an existing option group to which you can add the S3_INTEGRATION option.
2. Add the S3_INTEGRATION option to the option group.
3. Create a new Oracle DB instance and associate the option group with it, or modify an Oracle DB instance to associate the option group with it.

Transferring files between Amazon RDS for Oracle and an Amazon S3 bucket

You can use Amazon RDS procedures to transfer files between an Oracle DB instance and an Amazon S3 bucket.

Uploading files from an Oracle DB instance to an Amazon S3 bucket

You can upload files from an Oracle DB instance to an Amazon S3 bucket. For example, you can upload Oracle Recovery Manager (RMAN) backup files. The maximum object size in an Amazon S3 bucket is 5 TB.

You upload files using the `rdsadmin.rdsadmin_s3_tasks.upload_to_s3` procedure. This procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
p_bucket_name	VARCHAR2	–	required	The name of the Amazon S3 bucket to upload files to.
p_directory_name	VARCHAR2	–	required	The name of the Oracle directory object to upload files from. The directory can be any user-created directory object or the Data Pump directory, such as DATA_PUMP_DIR.
p_s3_prefix	VARCHAR2	–	required	An Amazon S3 file name prefix that files are up-

				loaded to. An empty prefix uploads all files to the top level in the specified Amazon S3 bucket and doesn't add a prefix to the file names. For example, if the prefix is folder_1/oradb, files are uploaded to folder_1. In this case, the oradb prefix is added to each file.
p_prefix	VARCHAR2	–	required	A file name prefix that file names must match to be uploaded. An empty prefix uploads all files in the specified directory.

The return value for the `rdsadmin.rdsadmin_s3_tasks.upload_to_s3` procedure is a task ID.

The following example uploads all of the files in the *DATA_PUMP_DIR* directory to the Amazon S3 bucket named *mys3bucket*.

```
SELECT rdsadmin.rdsadmin_s3_tasks.upload_
to_s3(
   p_bucket_name   => 'mys3bucket',
   p_prefix        => '',
```

```
  p_s3_prefix    => '',
  p_directory_name => 'DATA_PUMP_DIR')
 AS TASK_ID FROM DUAL;
```

The following example uploads all of the files with the prefix *db* in the *DATA_PUMP_DIR* directory to the Amazon S3 bucket named *mys3bucket*.

```
SELECT rdsadmin.rdsadmin_s3_tasks.upload_
to_s3(
   p_bucket_name   => 'mys3bucket',
   p_prefix        => 'db',
   p_s3_prefix     => '',
   p_directory_name => 'DATA_PUMP_DIR')
 AS TASK_ID FROM DUAL;
```

The following example uploads all of the files in the *DATA_PUMP_DIR* directory to the Amazon S3 bucket named *mys3bucket*. The files are uploaded to a dbfiles folder.

```
SELECT rdsadmin.rdsadmin_s3_tasks.upload_
to_s3(
   p_bucket_name   => 'mys3bucket',
   p_prefix        => '',
   p_s3_prefix     => 'dbfiles/',
   p_directory_name => 'DATA_PUMP_DIR')
 AS TASK_ID FROM DUAL;
```

The following example uploads all of the files in the *DATA_PUMP_DIR* directory to the Amazon S3 bucket named *mys3bucket*. The files are uploaded to a dbfiles folder and ora is added to the beginning of each file name.

```
SELECT rdsadmin.rdsadmin_s3_tasks.upload_
to_s3(
   p_bucket_name   => 'mys3bucket',
   p_prefix        => '',
   p_s3_prefix     => 'dbfiles/ora',
```

```
  p_directory_name => 'DATA_PUMP_DIR')
AS TASK_ID FROM DUAL;
```

In each example, the SELECT statement returns the ID of the task in a VARCHAR2 data type.

You can view the result by displaying the task's output file.

```
SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('BDUMP','dbtask-task-id.log'));
```

Replace *task-id* with the task ID returned by the procedure.

Downloading files from an Amazon S3 bucket to an Oracle DB instance

To download files from an Amazon S3 bucket to an Oracle DB instance, use the Amazon RDS

procedure rdsadmin.rdsadmin_s3_tasks.download_from_s3. The rdsadmin.rdsadmin_s3_tasks.download_from_s3 procedure has the following parameters.

Parameter name	Data type	Default	Required	Description
p_bucket_name	VARCHAR2	–	required	The name of the Amazon S3 bucket to download files from.
p_directory_name	VARCHAR2	–	required	The name of the Oracle directory object to download files to. The directory can be any user-created directory object or the Data Pump directory, such as DATA_PUMP_DIR.
p_s3_prefix	VARCHAR2	''	optional	A file name prefix that file names must match to be downloaded. An empty prefix downloads all of the top level files in the specified Amazon S3 bucket, but not the files in folders in the bucket. The procedure downloads Amazon S3 objects only from the first level folder that matches the prefix. Nested directory structures matching the specified prefix are not downloaded. For example, suppose that an Amazon S3 bucket has the folder structure folder_1/folder_2/folder_3. Suppose also that you specify

				the 'folder_1/folder_2/' prefix. In this case, only the files in folder_2 are downloaded, not the files in folder_1 or folder_3. If, instead, you specify the 'folder_1/folder_2' prefix, all files in folder_1 that match the 'folder_2' prefix are downloaded, and no files in folder_2 are downloaded.

The return value for the rdsadmin.rdsadmin_s3_tasks.download_from_s3 procedure is a task ID.

The following example downloads all of the files in the Amazon S3 bucket named *mys3bucket* to the *DATA_PUMP_DIR* directory.

```
SELECT rdsadmin.rdsadmin_s3_tasks.download_from_s3(
   p_bucket_name    => 'mys3bucket',
   p_directory_name => 'DATA_PUMP_DIR')
 AS TASK_ID FROM DUAL;
```

The following example downloads all of the files with the prefix *db* in the Amazon S3 bucket named *mys3bucket* to the *DATA_PUMP_DIR* directory.

```
SELECT rdsadmin.rdsadmin_s3_tasks.download_from_s3(
    p_bucket_name    => 'mys3bucket',
    p_s3_prefix      => 'db',
    p_directory_name => 'DATA_PUMP_DIR')
  AS TASK_ID FROM DUAL;
```

The following example downloads all of the files in the folder *myfolder/* in the Amazon S3 bucket named *mys3bucket* to the *DATA_PUMP_DIR* directory. Use the prefix parameter setting to specify the Amazon S3 folder.

```
SELECT rdsadmin.rdsadmin_s3_tasks.download_from_s3(
   p_bucket_name   => 'mys3bucket',
   p_s3_prefix     => 'myfolder/',
   p_directory_name => 'DATA_PUMP_DIR')
 AS TASK_ID FROM DUAL;
```

In each example, the SELECT statement returns the ID of the task in a VARCHAR2 data type.

You can view the result by displaying the task's output file.

```
SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('BDUMP','dbtask-task-id.log'));
```

Replace *task-id* with the task ID returned by the procedure.

You can use the UTL_FILE.FREMOVE Oracle procedure to remove files from a directory.

Monitoring the status of a file transfer

File transfer tasks publish Amazon RDS events when they start and when they complete.

You can view the status of an ongoing task in a bdump file. The bdump files are located in the /rdsdbdata/log/trace directory. Each bdump file name is in the following format.

```
dbtask-task-id.log
```

Replace *task-id* with the ID of the task that you want to monitor.

You can use the rdsadmin.rds_file_util.read_text_file stored procedure to view the contents of bdump files. For example, the following query returns

the contents of the *dbtask-1546988886389-2444.log* bdump file.

```
SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('BDUMP','dbtask-1546988886389-2444.log'));
```

Removing the Amazon S3 integration option

You can remove Amazon S3 integration option from a DB instance.

To remove the Amazon S3 integration option from a DB instance, do one of the following:

- To remove the Amazon S3 integration option from multiple DB instances, remove the S3_INTEGRATION option from the option group to which the DB instances belong.

This change affects all DB instances that use the option group.

To remove the Amazon S3 integration option from a single DB instance, modify the DB instance and specify a different option group that doesn't include the S3_INTEGRATION option. You can specify the default (empty) option group or a different custom option group.

Creating an Oracle DB instance

The basic building block of Amazon RDS is the DB instance. Your Amazon RDS DB instance is similar to your on-premises Oracle database.

In this topic, you create a sample Oracle DB instance. You then connect to the DB instance and run a simple query. Finally, you delete the sample DB instance.

Creating a sample Oracle DB instance

The DB instance is where you run your Oracle databases.

Console

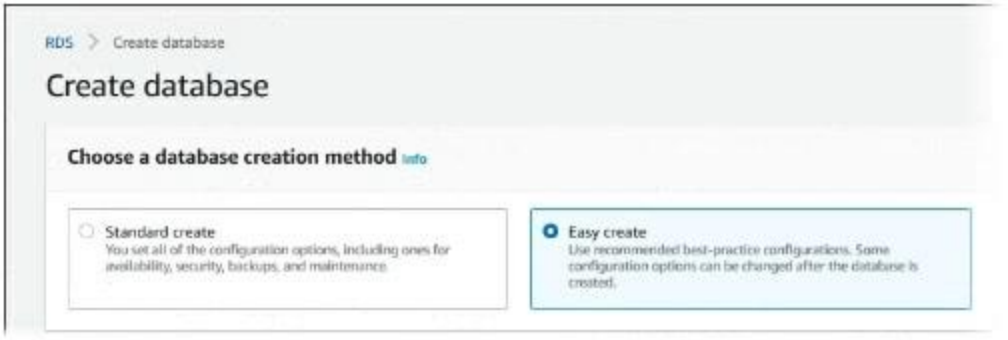
You can create a DB instance running Oracle with the AWS Management Console with **Easy create** enabled or not enabled. With **Easy create** enabled, you specify only the DB engine type, DB instance

size, and DB instance identifier. **Easy create** uses the default setting for other configuration options. With **Easy create** not enabled, you specify more configuration options when you create a database, including ones for availability, security, backups, and maintenance.

For this example, you use **Easy create** to create a DB instance running the Oracle database engine with a db.m4.large DB instance class.

To create an Oracle DB instance with Easy create enabled

1. Sign in to the AWS Management Console and open the Amazon RDS console.
2. In the upper-right corner of the Amazon RDS console, choose the AWS Region in which you want to create the DB instance.
3. In the navigation pane, choose **Databases**.
4. Choose **Create database** and ensure that **Easy create** is chosen.



5. In **Configuration**, choose **Oracle**.
6. For **DB instance size**, choose **Free tier**. If **Free tier** isn't available, choose **Dev/Test**.
7. For **DB instance identifier**, enter a name for the DB instance, or leave the default name.
8. For **Master username**, enter a name for the master user, or leave the default name.

The **Create database** page should look similar to the following image.

Create database

Choose a database creation method Info

- ○ Standard Create
  You set all of the configuration options, including ones for availability, security, backups, and maintenance.
- ◉ Easy Create
  Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

Configuration

Engine type Info

- ○ Amazon Aurora 
- ○ MySQL 
- ○ MariaDB 
- ○ PostgreSQL 
- ◉ Oracle ORACLE
- ○ Microsoft SQL Server Microsoft SQL Server

DB instance size

- ○ Production
  db.r4.large
  2 vCPUs
  15.25 GiB RAM
  500 GiB
- ◉ Dev/Test
  db.m4.large
  2 vCPUs
  8 GiB RAM
  100 GiB

DB instance identifier
Type a name for your DB instance. The name must be unique cross all DB instances owned by your AWS account in the current AWS Region.

database-1

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens (1 to 15 for SQL Server). First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

Master username Info
Type a login ID for the master user of your DB instance.

admin

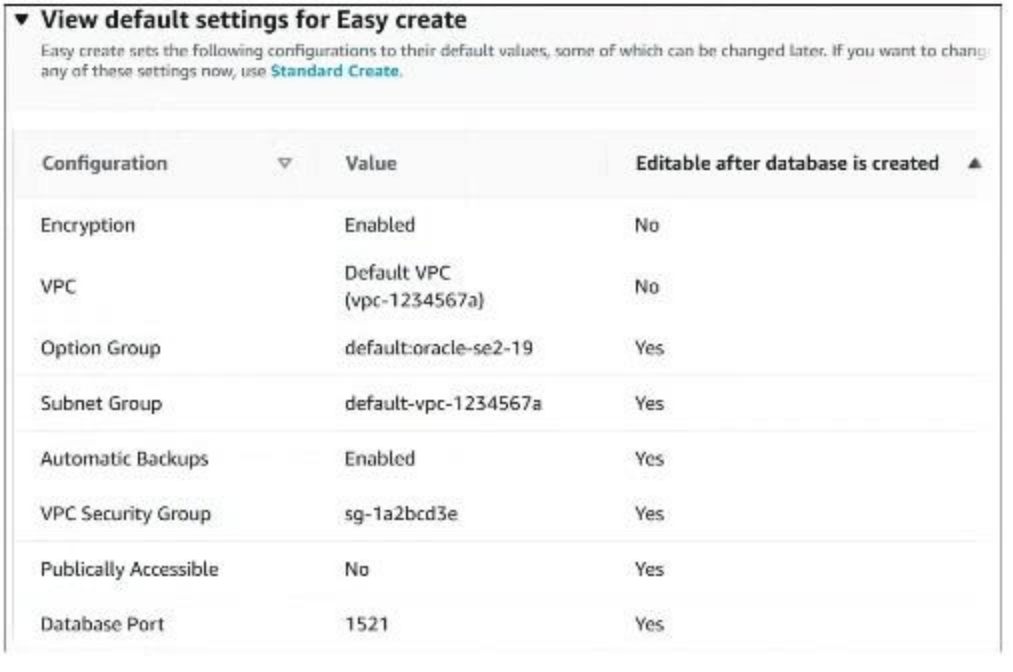
1 to 16 alphanumeric characters. First character must be a letter

☑ Auto generate a password
Amazon RDS can generate a password for you, or you can specify your own password

9. To use an automatically generated master password for the DB instance, make sure that the **Auto generate a password** check box is chosen.

To enter your master password, clear the **Auto generate a password** check box, and then enter the same password in **Master password** and **Confirm password**.

10. (Optional) Open **View default settings for Easy create**.



You can examine the default settings that are used when **Easy create** is enabled. If you want to change one or more settings during database creation, choose **Standard create** to set them. The **Editable after database creation** column shows which options you can change after database creation. To change a setting with **No** in that column, use **Standard create**. For settings with **Yes** in that column, you can either use **Standard create** or modify the DB instance after it's created to change the setting.

11. Choose **Create database**.

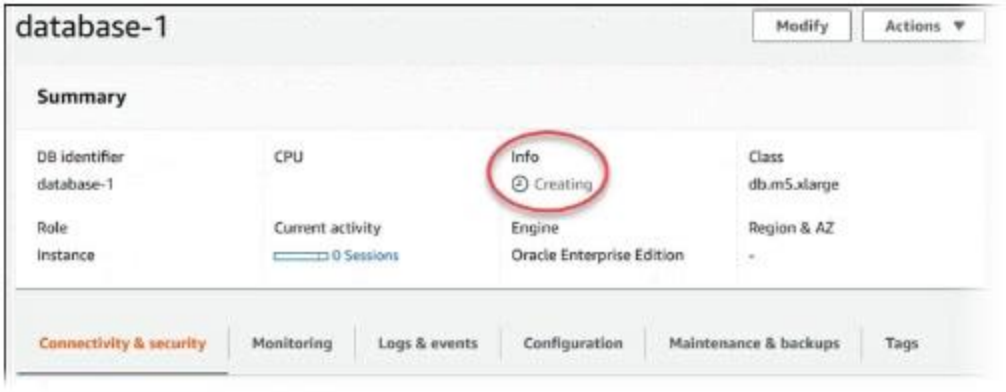
If you used an automatically generated password, the **View credential details** button appears on the **Databases** page.

To view the master user name and password for the DB instance, choose **View credential details**.

To connect to the DB instance as the master user, use the user name and password that appear.

12. For **Databases**, choose the name of the new Oracle DB instance.

On the RDS console, the details for new DB instance appear. The DB instance has a status of **creating** until the DB instance is ready to use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and the amount of storage, it can take up to 20 minutes before the new instance is available.



Connecting to your sample Oracle DB instance

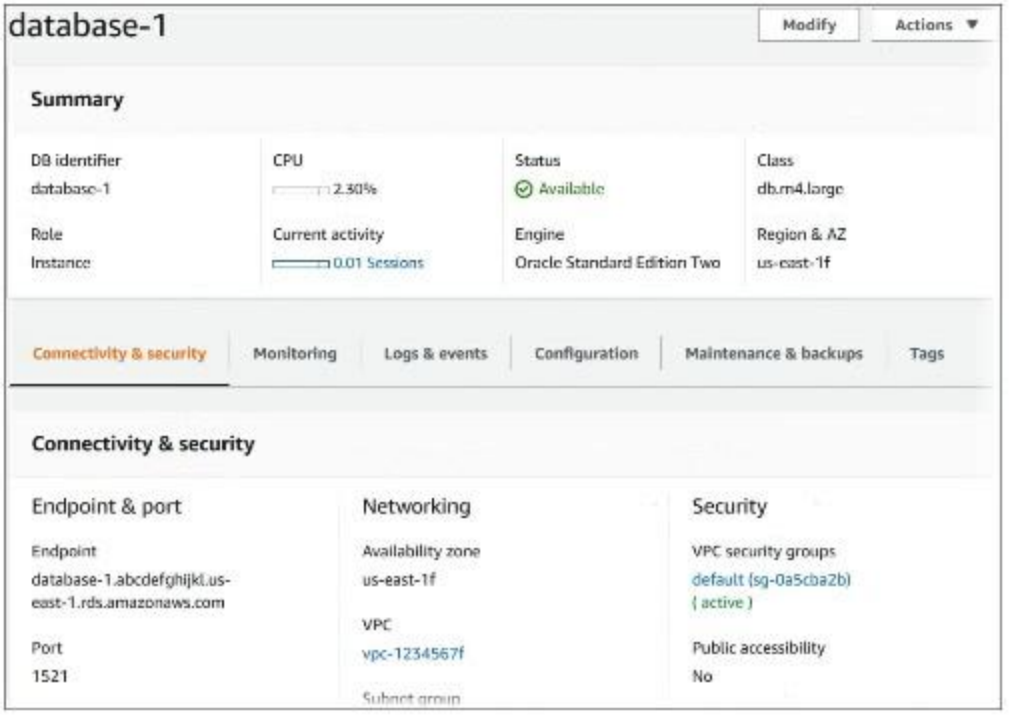
After Amazon RDS provisions your DB instance, you can use any standard SQL client application to connect to the DB instance. In this procedure, you

connect to your sample DB instance by using the Oracle sqlplus command line utility.

To connect to a DB instance using SQL*Plus

1. Find the endpoint (DNS name) and port number for your DB Instance.
    a. Open the RDS console and then choose **Databases** to display a list of your DB instances.
    b. Choose the Oracle DB instance name to display its details.
    c. On the **Connectivity & security** tab, copy the following pieces of information:
        - Endpoint
        - Port

You need both the endpoint and the port number to connect to the DB instance.



d. On the **Configuration** tab, copy the following pieces of information:

- DB name (not the DB instance ID)
- Master username

You need both the DB name and the master username to connect to the DB instance.

2. Enter the following command on one line at a command prompt to connect to your DB instance by using the sqlplus utility. Use the following values:

- For *dbuser*, enter the name of the master user that you copied in the preceding steps.
- For HOST=*endpoint*, enter the endpoint that you copied in the preceding steps.
- For PORT=*portnum*, enter the port number that you copied in the preceding steps.
- For SID=*DB_NAME*, enter the Oracle database name (not the instance name) that you copied in the preceding steps.

```
sqlplus
'dbuser@(DESCRIPTION=(ADDRESS=(PROTOCOL=
TCP)(HOST=endpoint)(PORT=portnum))(CONNEC-
T_DATA=(SID=DB_NAME)))'
```

You should see output similar to the following.

```
SQL*Plus: Release 11.1.0.7.0 - Production on Wed
May 25 15:13:59 2011

SQL>
```

How do I configure Amazon RDS Oracle DB instances to work with shared servers?

How do I configure Amazon Relational Database Service (Amazon RDS) Oracle DB instances to work with shared servers?

Short Description

Oracle users can connect to RDS DB instances using either dedicated or shared server processes. Before using shared servers, consider the following:

- Using shared servers incurs CPU overhead, which might cause performance issues.
- Using a shared server means that the UGA allocation is allocated inside a large pool. Be sure that you have sufficient free space inside SGA to accommodate shared servers. Insufficient free space can cause "large pool free" errors to appear in the instance's alert log and trace files.

- Using shared servers might cause more frequent dynamic reallocation of SGA memory, which can cause performance issues.
- During database reboot or failover, a large increase in application connections can overwhelm the dispatchers if the DISPATCHERS parameter isn't set high enough relative to connection throughput.
- Running batch processes, long-running queries, heavy loads, or long-running DBA tasks on shared servers can cause other jobs to queue up, which can cause performance issues. Use dedicated servers for large jobs.

Resolution

To balance the benefits and limitations of using shared servers:

- Use shared servers for a high number of OLTP sessions that connect and disconnect often and perform light operations.
- Use dedicated servers for long-running batch operations and heavy administrative tasks such as creating indexes.

Note: The following examples are provided as a baseline for enabling shared servers with the specified instance size (db.r3.large instance). Administrators should apply parameter group settings that optimize memory based on their use cases.

1. Modify the custom parameter group to set the following parameters to the maximum permitted value or to a value that meets your use case:
dispatchers=(PROTOCOL=TCP)(DISPATCHERS=30)
max_dispatchers=30
2. Log in to the instance, and then view the default value of SESSIONS for the host size. If you're not

using the default settings, test the instance with the default parameter group:

```
SQL> show parameter sessions
2428
```

3. Set SHARED_SERVERS and MAX_SHARED_SERVERS to 10% of that value:

```
sessions=2428
shared_servers=243
max_shared_servers=243
```

4. Set LARGE_POOL_SIZE equal to SHARED_SERVERS value * 1 MB (243 * 1048576 bytes in this case).

```
large_pool_size= 254803968
```

5. Query v$sgastat for large_pool_size "free memory" to be sure that large pool is adequately sized.

```
SQL> select name, pool, bytes/1024/1024 megs
from v$sgastat where name='free memory' and
pool='large pool';
Name        POOL      Megs
------------ ---------- -------
```

free memory large pool 243

6. View the parameter group settings that are applied to a running Oracle instance, and then run the following SQL query from the instance:

```
select name, value from v$parameter where
name in ('processes', 'sessions', 'shared_servers',
'dispatchers', 'memory_target', 'memory_max_
target', 'large_pool_size');
```

7. Run the following SQL query from your Oracle instance to see if sessions are connecting as shared:

```
SQL> select
decode(server,'NONE','SHARED',server), count(*)
from v$session
group by decode(server,'NONE','SHARED',server);
```

To enable dedicated and shared server access to the same Oracle instance, use dual tnsnames.ora entries, such as in the following example:

```
 # make the default shared

  dbname =
  (DESCRIPTION=
```

```
    (ADDRESS_LIST=
    (ADDRESS=(PROTOCOL=TCP)
(HOST=dbname.endpoint.amazonaws.com)
(PORT=1521))
  )
  (CONNECT_DATA=
    (SID=dbname)
  )
 )

 # use the dedicated one for batch processes and
dba tasks such as creating indexes

 dbname_d=
 (DESCRIPTION=
  (ADDRESS_LIST=
    (ADDRESS=(PROTOCOL=TCP)
(HOST=dbname.endpoint.amazonaws.com)
(PORT=1521))
  )
  (CONNECT_DATA=
```

```
    (SID=dbname)
    (SERVER=DEDICATED)
  )
)
```

Connecting to your Oracle DB instance

After Amazon RDS provisions your Oracle DB instance, you can use any standard SQL client application to connect to the DB instance. In this topic, you connect to a DB instance that is running the Oracle database engine by using Oracle SQL Developer or SQL*Plus.

Finding the endpoint of your Oracle DB instance

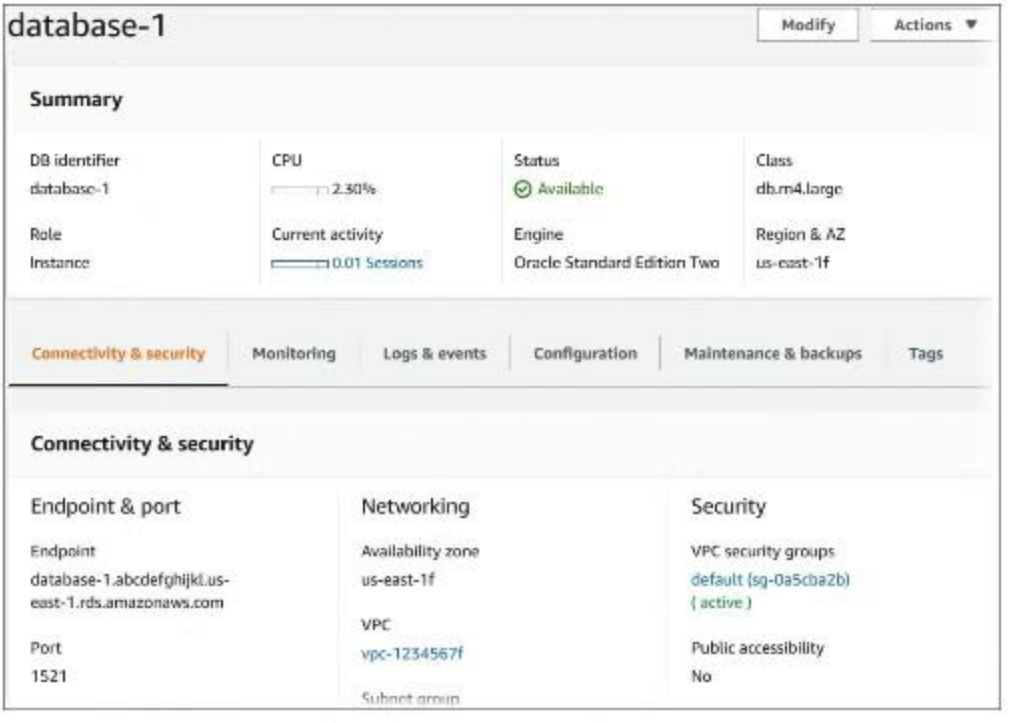
Each Amazon RDS DB instance has an endpoint, and each endpoint has the DNS name and port number for the DB instance. To connect to your DB instance using a SQL client application, you need the DNS name and port number for your DB instance.

You can find the endpoint for a DB instance using the Amazon RDS console or the AWS CLI.

Console

To find the endpoint using the console

1. Sign in to the AWS Management Console and open the Amazon RDS console.
2. In the upper-right corner of the console, choose the AWS Region of your DB instance.
3. Find the DNS name and port number for your DB Instance.
    a. Choose **Databases** to display a list of your DB instances.
    b. Choose the Oracle DB instance name to display the instance details.
    c. On the **Connectivity & security** tab, copy the endpoint. Also, note the port number. You need both the endpoint and the port number to connect to the DB instance.



AWS CLI

To find the endpoint of an Oracle DB instance by using the AWS CLI, call the describe-db-instances command.

Example To find the endpoint using the AWS CLI

```
aws rds describe-db-instances
```

Search for `Endpoint` in the output to find the DNS name and port number for your DB instance. The `Address` line in the output contains the DNS name. The following is an example of the JSON endpoint output.

```
"Endpoint": {
  "HostedZoneId": "Z1PVIF0B656C1W",
  "Port": 3306,
  "Address": "myinstance.123456789012.us-west-2.rds.amazonaws.com"
},
```

Connecting to your DB instance

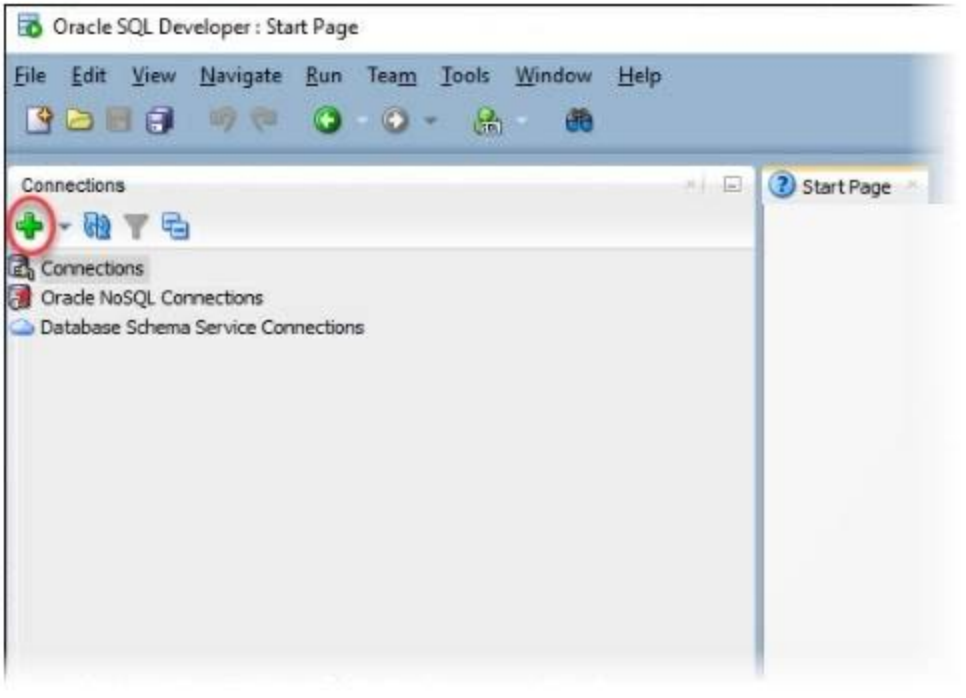
using Oracle SQL developer

In this procedure, you connect to your DB instance by using Oracle SQL Developer.

To connect to your DB instance, you need its DNS name and port number.

To connect to a DB instance using SQL developer

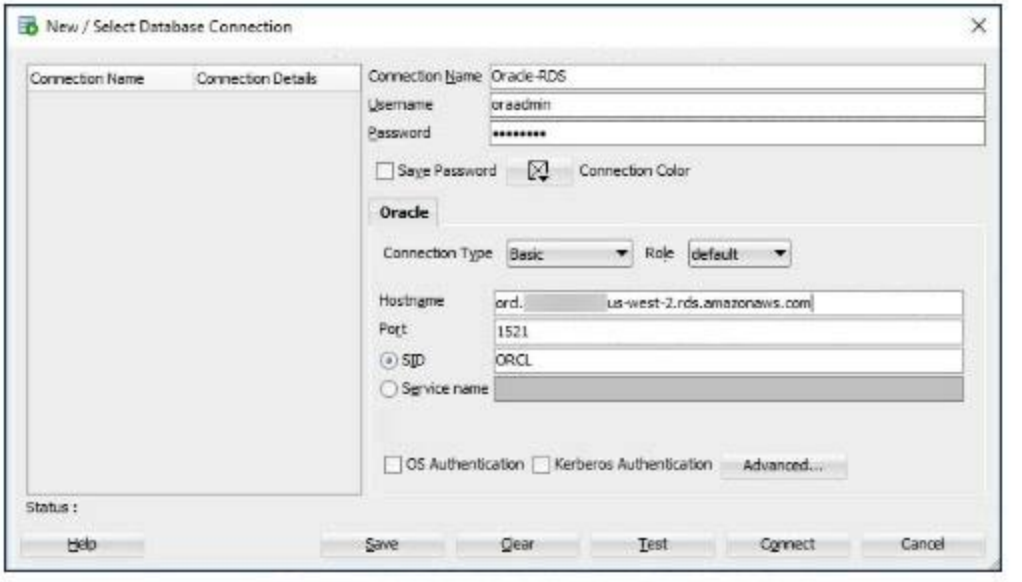
1. Start Oracle SQL Developer.
2. On the **Connections** tab, choose the **add (+)** icon.



3. In the **New/Select Database Connection** dialog box, provide the information for your DB instance:

- For **Connection Name**, enter a name that describes the connection, such as Oracle-RDS.
- For **Username**, enter the name of the database administrator for the DB instance.
- For **Password**, enter the password for the database administrator.
- For **Hostname**, enter the DNS name of the DB instance.
- For **Port**, enter the port number.
- For **SID**, enter the Oracle database SID.

The completed dialog box should look similar to the following.

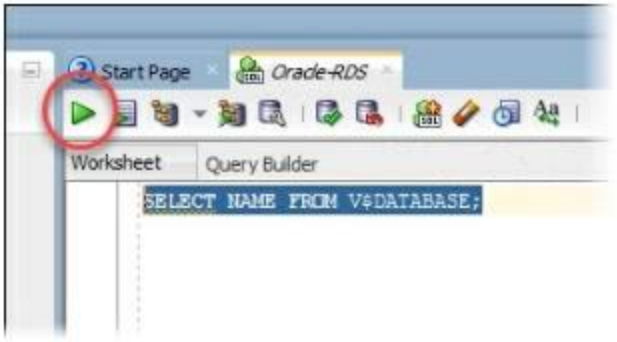


4. Choose **Connect**.
5. You can now start creating your own databases and running queries against your DB instance and databases as usual. To run a test query against your DB instance, do the following:

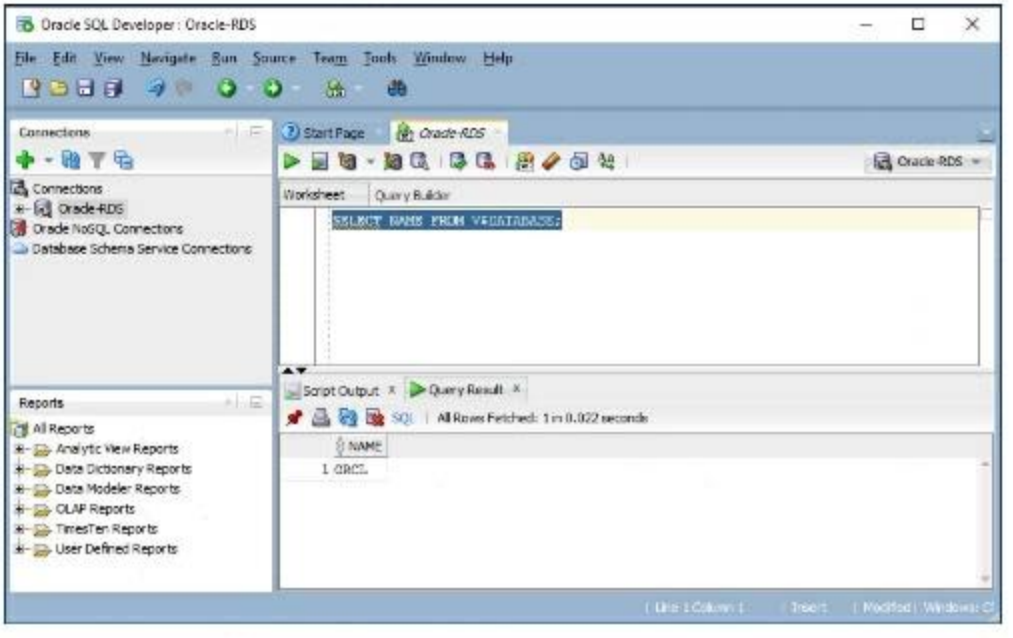
    a. In the **Worksheet** tab for your connection, enter the following SQL query.

```
SELECT NAME FROM V$DATABASE;
```

    b. Choose the **execute** icon to run the query.



SQL Developer returns the database name.



Connecting to your DB instance using SQL*Plus

You can use a utility like SQL*Plus to connect to an Amazon RDS DB instance running Oracle.

To connect to your DB instance, you need its DNS name and port number.

Example To connect to an Oracle DB instance using SQL*Plus

In the following examples, substitute the user name of your DB instance administrator. Also, substitute the DNS name for your DB instance, and then include the port number and the Oracle SID. The SID value is the name of the DB instance's database that you specified when you created the DB instance, and not the name of the DB instance.

For Linux, macOS, or Unix:

```
sqlplus 'user_name@(DESCRIPTION=(ADDRESS=
(PROTOCOL=TCP)(HOST=dns_name)(PORT=port))
(CONNECT_DATA=(SID=database_name)))'
```

For Windows:

```
sqlplus user_name@(DESCRIPTION=(ADDRESS=
(PROTOCOL=TCP)(HOST=dns_name)(PORT=port))
(CONNECT_DATA=(SID=database_name)))
```

You should see output similar to the following.

```
SQL*Plus: Release 12.1.0.2.0 Production on Mon
Aug 21 09:42:20 2017
```

After you enter the password for the user, the SQL prompt appears.

```
SQL>
```

The shorter format connection string (Easy connect or EZCONNECT), such as `sqlplus USER/PASSWORD@LONGER-THAN-63-CHARS-RDS-ENDPOINT-HERE:1521/DATABASE_IDENTIFIER`,

might encounter a maximum character limit and should not be used to connect.

Considerations for security groups

For you to connect to your DB instance, it must be associated with a security group that contains the necessary IP addresses and network configuration. Your DB instance might use the default security group. If you assigned a default, nonconfigured security group when you created the DB instance, the firewall prevents connections.

To create a new security group, security group that you create depends on the Amazon EC2 platform for your DB instance. In general, if your DB instance is on the *EC2-Classic* platform, you create a DB security group; if your DB instance is on the *VPC* platform, you create a VPC security group.

After you create the new security group, you modify your DB instance to associate it with the security group.

Considerations for process architecture

Server processes handle user connections to an Oracle DB instance. By default, the Oracle DB instance uses dedicated server processes. With dedicated server processes, each server process services only one user process. You can optionally configure shared server processes. With shared server processes, each server process can service multiple user processes.

You might consider using shared server processes when a high number of user sessions are using too much memory on the server. You might also consider shared server processes when sessions connect and disconnect very often, resulting in

performance issues. There are also disadvantages to using shared server processes. For example, they can strain CPU resources, and they are more complicated to configure and administer.

Troubleshooting connections to your Oracle DB instance

The following are issues you might encounter when you try to connect to your Oracle DB instance.

Issue	Troubleshooting suggestions
Unable to connect to your DB instance.	For a newly created DB instance, the DB instance has a status of **creating** until it is ready to use. When the state changes to **available**, you can connect to the DB instance. Depending on the DB instance class and the amount of storage, it can take up to 20

	minutes before the new DB instance is available.
Unable to connect to your DB instance.	If you can't send or receive communications over the port that you specified when you created the DB instance, you can't connect to the DB instance. Check with your network administrator to verify that the port you specified for your DB instance allows inbound and outbound communication.
Unable to connect to your DB instance.	The access rules enforced by your local firewall and the IP addresses you authorized to access your DB instance in the security group for the DB instance might not match. The problem is most likely the inbound or outbound rules on your firewall. You can add or edit an inbound rule in the security group. For

	Source, choose **My IP**. This allows access to the DB instance from the IP address detected in your browser.
Connect failed because target host or object does not exist – Oracle, Error: ORA-12545	Make sure that you specified the server name and port number correctly. For **Server name**, enter the DNS name from the console.
Invalid username/ password; logon denied – Oracle, Error: ORA-01017	You were able to reach the DB instance, but the connection was refused. This is usually caused by providing an incorrect user name or password. Verify the user name and password, and then retry.

Modifying connection properties

using sqlnet.ora parameters

The sqlnet.ora file includes parameters that configure Oracle Net features on Oracle database servers and clients. Using the parameters in the sqlnet.ora file, you can modify properties for connections in and out of the database.

Setting sqlnet.ora parameters

Amazon RDS for Oracle parameter groups include a subset of sqlnet.ora parameters. You set them in the same way that you set other Oracle parameters. The `sqlnetora.` prefix identifies which parameters are sqlnet.ora parameters. For example, in an Oracle parameter group in Amazon RDS, the `default_sdu_size` sqlnet.ora parameter is `sqlnetora.default_sdu_size`.

Supported sqlnet.ora parameters

Amazon RDS supports the following sqlnet.ora parameters. Changes to dynamic sqlnet.ora parameters take effect immediately.

Parameter	Valid values	Static/Dynamic	Description
sqlnetora.default_sdu_size	Oracle 12c – 512 to 2097152	Dynamic	The session data unit (SDU) size, in bytes. The SDU is the amount of data that is put in a buffer and sent across the network at one time.
sqlnetora.diag_adr_enabled	ON, OFF	Dynamic	A value that enables or disables Automatic Diagnostic Repository (ADR) tracing. ON specifies that ADR file tracing is used.

			OFF specifies that non-ADR file tracing is used.
sqlnetora.recv_buf_size	8192 to 268435456	Dynamic	The buffer space limit for receive operations of sessions, supported by the TCP/IP, TCP/IP with SSL, and SDP protocols.
sqlnetora.send_buf_size	8192 to 268435456	Dynamic	The buffer space limit for send operations of sessions, supported by the TCP/IP, TCP/IP with SSL, and SDP protocols.
sqlnetora.sqlnet.allowed_logon_version_client	8, 10, 11, 12	Dynamic	Minimum authentication protocol version allowed for clients, and servers acting as clients, to establish a connec-

			tion to Oracle DB instances.
sqlnetora.sqlnet.allowed_logon_version_server	8, 9, 10, 11, 12, 12a	Dynamic	Minimum authentication protocol version allowed to establish a connection to Oracle DB instances.
sqlnetora.sqlnet.expire_time	0 to 1440	Dynamic	Time interval, in minutes, to send a check to verify that client-server connections are active.
sqlnetora.sqlnet.inbound_connect_timeout	0 or 10 to 7200	Dynamic	Time, in seconds, for a client to connect with the database server and provide the necessary authentication information.
sqlnetora.sqlnet.outbound_connect_timeout	0 or 10 to 7200	Dynamic	Time, in seconds,

			for a client to establish an Oracle Net connection to the DB instance.
sqlnetora.sqlnet.recv_timeout	0 or 10 to 7200	Dynamic	Time, in seconds, for a database server to wait for client data after establishing a connection.
sqlnetora.sqlnet.send_timeout	0 or 10 to 7200	Dynamic	Time, in seconds, for a database server to complete a send operation to clients after establishing a connection.
sqlnetora.tcp.connect_timeout	0 or 10 to 7200	Dynamic	Time, in seconds, for a client to establish a TCP connection to the database server.

sqlnetora.trace_level_server	0, 4, 10, 16, OFF, USER, ADMIN, SUPPORT	Dynamic	For non-ADR tracing, turns server tracing on at a specified level or turns it off.

The default value for each supported sqlnet.ora parameter is the Oracle default for the release.

Viewing sqlnet.ora parameters

You can view sqlnet.ora parameters and their settings using the AWS Management Console, the AWS CLI, or a SQL client.

Viewing sqlnet.ora parameters using the console

In Oracle parameter groups, the sqlnetora. prefix identifies which parameters are sqlnet.ora parameters.

Viewing sqlnet.ora parameters using the AWS CLI

To view the sqlnet.ora parameters that were configured in an Oracle parameter group, use the AWS CLI describe-db-parameters command.

To view the all of the sqlnet.ora parameters for an Oracle DB instance, call the AWS CLI download-db-log-file-portion command. Specify the DB instance identifier, the log file name, and the type of output.

Example

The following code lists all of the sqlnet.ora parameters for mydbinstance.

For Linux, macOS, or Unix:

```
aws rds download-db-log-file-portion \
  --db-instance-identifier mydbinstance \
  --log-file-name trace/sqlnet-parameters \
  --output text
```

For Windows:

```
aws rds download-db-log-file-portion ^
  --db-instance-identifier mydbinstance ^
  --log-file-name trace/sqlnet-parameters ^
  --output text
```

Viewing sqlnet.ora parameters using a SQL client

After you connect to the Oracle DB instance in a SQL client, the following query lists the sqlnet.ora parameters.

```
SELECT * FROM TABLE
  (rdsadmin.rds_file_util.read_text_file(
     p_directory => 'BDUMP',
     p_filename  => 'sqlnet-parameters'));
```

Deleting your sample DB instance

After you are done exploring the sample DB instance that you created, you should delete the DB instance so that you are no longer charged for it.

To delete a DB instance with no final DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console.
2. In the navigation pane, choose **Databases**.
3. Choose the DB instance that you want to delete.
4. For **Actions**, choose **Delete**.
5. For **Create final snapshot?**, choose **No**, and choose the acknowledgment.
6. Choose **Delete**.

Managing an Amazon RDS DB instance

Stopping an Amazon RDS DB instance temporarily

If you use a DB instance intermittently, for temporary testing, or for a daily development activity, you can stop your Amazon RDS DB instance temporarily to save money. While your DB instance is stopped, you are charged for provisioned storage (including Provisioned IOPS) and backup storage (including manual snapshots and automated backups within your specified retention window), but not for DB instance hours.

Stopping and starting a DB instance is supported for all DB instance classes, and in all AWS Regions.

You can stop and start a DB instance whether it is configured for a single Availability Zone or for Multi-AZ, for database engines that support Multi-

AZ deployments. You can't stop an Amazon RDS for SQL Server DB instance in a Multi-AZ configuration.

When you stop a DB instance, the DB instance performs a normal shutdown and stops running. The status of the DB instance changes to stopping and then stopped. Any storage volumes remain attached to the DB instance, and their data is kept. Any data stored in the RAM of the DB instance is deleted.

Stopping a DB instance removes pending actions, except for pending actions for the DB instance's option group or DB parameter group.

Automated backups aren't created while a DB instance is stopped. Backups can be retained longer than the backup retention period if a DB instance has been stopped. RDS doesn't include time spent in the stopped state when the backup retention window is calculated.

Important

You can stop a DB instance for up to seven days. If you don't manually start your DB instance after seven days, your DB instance is automatically started so that it doesn't fall behind any required maintenance updates.

Benefits

Stopping and starting a DB instance is faster than creating a DB snapshot, and then restoring the snapshot.

When you stop a DB instance it retains its ID, Domain Name Server (DNS) endpoint, parameter group, security group, and option group. When you start a DB instance, it has the same configuration as when you stopped it. In addition, if you stop a DB instance, Amazon RDS retains the Amazon Simple Storage Service (Amazon S3) transaction logs so you can do a point-in-time restore if necessary.

Limitations

The following are some limitations to stopping and starting a DB instance:

- You can't stop a DB instance that has a read replica, or that is a read replica.
- You can't stop an Amazon RDS for SQL Server DB instance in a Multi-AZ configuration.
- You can't modify a stopped DB instance.
- You can't delete an option group that is associated with a stopped DB instance.
- You can't delete a DB parameter group that is associated with a stopped DB instance.

Option and parameter group considerations

You can't remove persistent options (including permanent options) from an option group if there are DB instances associated with that option group.

This functionality is also true of any DB instance with a state of stopping, stopped, or starting.

You can change the option group or DB parameter group that is associated with a stopped DB instance, but the change does not occur until the next time you start the DB instance. If you chose to apply changes immediately, the change occurs when you start the DB instance. Otherwise the changes occurs during the next maintenance window after you start the DB instance.

Public IP address

When you stop a DB instance, it retains its DNS endpoint. If you stop a DB instance that has a public IP address, Amazon RDS releases its public IP address. When the DB instance is restarted, it has a different public IP address.

You should always connect to a DB instance using the DNS endpoint, not the IP address.

Stopping a DB instance temporarily

You can stop a DB using the AWS Management Console, the AWS CLI, or the RDS API.

Console

To stop a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console.
2. In the navigation pane, choose **Databases**, and then choose the DB instance that you want to stop.
3. For **Actions**, choose **Stop**.
4. (Optional) In the **Stop DB Instance** window, choose **Yes** for **Create Snapshot?** and enter the snapshot name for **Snapshot name**.

Choose **Yes** if you want to create a snapshot of the DB instance before stopping it.

5. Choose **Yes, Stop Now** to stop the DB instance, or choose **Cancel** to cancel the operation.

AWS CLI

To stop a DB instance by using the AWS CLI, call the stop-db-instance command with the following option:

- `--db-instance-identifier` – the name of the DB instance.

Example

```
aws rds stop-db-instance --db-instance-identifier mydbinstance
```

RDS API

To stop a DB instance by using the Amazon RDS API, call the StopDBInstance operation with the following parameter:

- DBInstanceIdentifier – the name of the DB instance.

Starting an Amazon RDS DB instance that was previously stopped

You can stop your Amazon RDS DB instance temporarily to save money. After you stop your DB instance, you can restart it to begin using it again.

When you start a DB instance that you previously stopped, the DB instance retains the ID, Domain Name Server (DNS) endpoint, parameter group, security group, and option group. When you start a stopped instance, you are charged a full instance hour.

Console

To start a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console.

2. In the navigation pane, choose **Databases**, and then choose the DB instance that you want to start.
3. For **Actions**, choose **Start**.

AWS CLI

To start a DB instance by using the AWS CLI, call the start-db-instance command with the following option:

- `--db-instance-identifier` – The name of the DB instance.

Example

```
aws rds start-db-instance --db-instance-identifier mydbinstance
```

RDS API

To start a DB instance by using the Amazon RDS API, call the StartDBInstance operation with the following parameter:

- DBInstanceIdentifier – The name of the DB instance.

Modifying an Amazon RDS DB instance

You can change the settings of a DB instance to accomplish tasks such as adding additional storage or changing the DB instance class. In this topic, you can find out how to modify an Amazon RDS DB instance and learn about the settings for DB instances.

We recommend that you test any changes on a test instance before modifying a production instance, so that you fully understand the impact of each change. Testing is especially important when upgrading database versions.

Most modifications to a DB instance you can either apply immediately or defer until the next maintenance window. Some modifications, such as parameter group changes, require that you manually reboot your DB instance for the change to take effect.

Console

To modify a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console.
2. In the navigation pane, choose **Databases**, and then choose the DB instance that you want to modify.
3. Choose **Modify**. The **Modify DB Instance** page appears.
4. Change any of the settings that you want.
5. When all the changes are as you want them, choose **Continue** and check the summary of modifications.
6. (Optional) Choose **Apply immediately** to apply the changes immediately. Choosing this option can cause an outage in some cases.

7. On the confirmation page, review your changes. If they are correct, choose **Modify DB Instance** to save your changes.

Or choose **Back** to edit your changes or **Cancel** to cancel your changes.

AWS CLI

To modify a DB instance by using the AWS CLI, call the modify-db-instance command. Specify the DB instance identifier and the values for the options that you want to modify. For information about each option, see Settings for DB instances.

Example

The following code modifies `mydbinstance` by setting the backup retention period to 1 week (7 days). The code enables deletion protection by using `--deletion-protection`. To disable deletion protection, use `--no-deletion-protection`. The changes are applied during the next maintenance window by

using `--no-apply-immediately`. Use `--apply-immediately` to apply the changes immediately.

For Linux, macOS, or Unix:

```
aws rds modify-db-instance \
  --db-instance-identifier mydbinstance \
  --backup-retention-period 7 \
  --deletion-protection \
  --no-apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
  --db-instance-identifier mydbinstance ^
  --backup-retention-period 7 ^
  --deletion-protection ^
  --no-apply-immediately
```

RDS API

To modify a DB instance by using the Amazon RDS API, call the ModifyDBInstance operation. Specify

the DB instance identifier, and the parameters for the settings that you want to modify.

Using the Apply Immediately setting

When you modify a DB instance, you can apply the changes immediately. To apply changes immediately, you choose the **Apply Immediately** option in the AWS Management Console. Or you use the `--apply-immediately` parameter when calling the AWS CLI or set the `ApplyImmediately` parameter to `true` when using the Amazon RDS API.

If you don't choose to apply changes immediately, the changes are put into the pending modifications queue. During the next maintenance window, any pending changes in the queue are applied. If you choose to apply changes immediately, your new changes and any changes in the pending modifications queue are applied.

Important

If any of the pending modifications require downtime, choosing the apply immediately option can cause unexpected downtime.

When you choose to apply a change immediately, any pending modifications are also applied immediately, instead of during the next maintenance window.

If you don't want a pending change to be applied in the next maintenance window, you can modify the DB instance to revert the change. You can do this by using the AWS CLI and specifying the `--apply-immediately` option.

Changes to some database settings are applied immediately, even if you choose to defer your changes.

Settings for DB instances

In the following table, you can find details about which settings you can and can't modify, when changes can be applied, and whether the changes cause downtime for your DB instance.

You can modify a DB instance using the console, the modify-db-instance CLI command, or the ModifyDBInstance RDS API operation.

Console setting and description	CLI option and RDS API parameter	When the change occurs	Downtime notes	Supported DB engines
Allocated storage The storage, in gibibytes, that you want to allocate for your DB instance. You can only increase the allocated storage. You can't reduce the allocated storage. You can't modify the storage of some older DB instances, or DB instances restored from older DB snapshots. The **Allocated storage** setting is disabled in the console if your DB instance isn't eligible. You can check whether you can allocate more storage by using the CLI command describe-valid-db-instance-modifications. This command returns the valid storage options for your DB instance. You can't modify allocated storage if the DB instance status is **storage-optimization** or if the allocated storage for the	**CLI option:** --allocated-storage **RDS API parameter:** AllocatedStorage	If you choose to apply the change immediately, it occurs immediately. If you don't choose to apply the change immediately, it occurs during the next maintenance window.	An outage doesn't occur during this change. Performance might be degraded during the change.	All DB engines

DB instance has been modified in the last six hours. The maximum storage allowed depends on your DB engine and the storage type.				

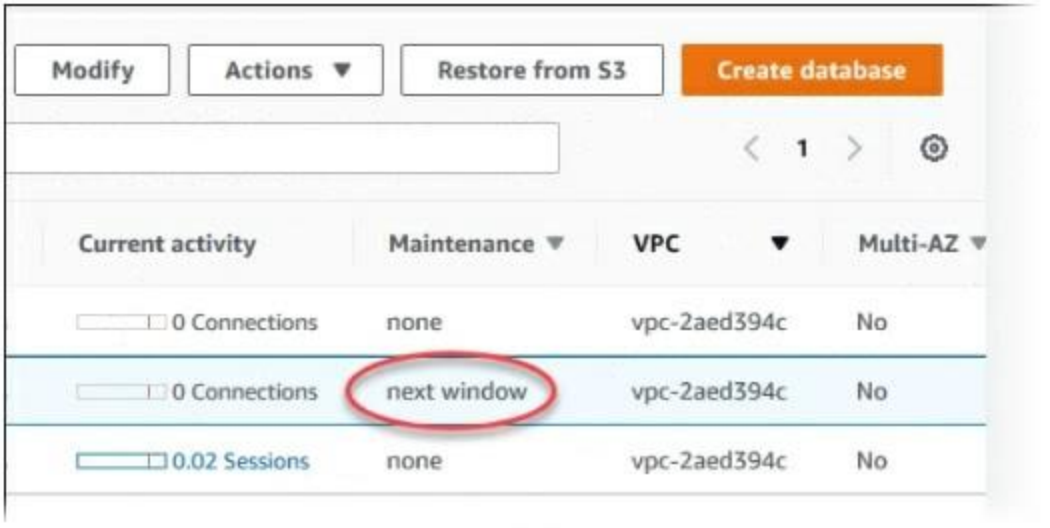
Maintaining a DB instance

Periodically, Amazon RDS performs maintenance on Amazon RDS resources. Maintenance most often involves updates to the DB instance's underlying hardware, underlying operating system (OS), or database engine version. Updates to the operating system most often occur for security issues and should be done as soon as possible.

Some maintenance items require that Amazon RDS take your DB instance offline for a short time. Maintenance items that require a resource to be offline include required operating system or database patching. Required patching is automatically scheduled only for patches that are related to security and instance reliability. Such patching occurs infrequently (typically once every few months) and seldom requires more than a fraction of your maintenance window.

Deferred DB instance modifications that you have chosen not to apply immediately are also applied during the maintenance window. For example, you might choose to change the DB instance class or parameter group during the maintenance window. Such modifications that you specify using the **pending reboot** setting don't show up in the **Pending maintenance** list.

You can view whether a maintenance update is available for your DB instance by using the RDS console, the AWS CLI, or the Amazon RDS API. If an update is available, it is indicated in the **Maintenance** column for the DB instance on the Amazon RDS console, as shown following.



If no maintenance update is available for a DB instance, the column value is **none** for it.

If a maintenance update is available for a DB instance, the following column values are possible:

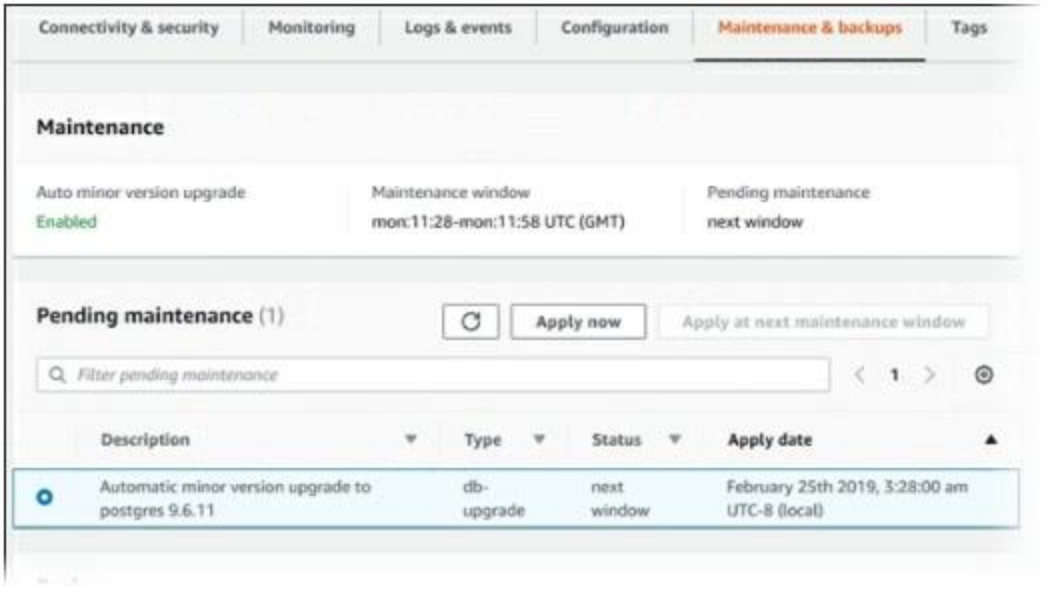
- **required** – The maintenance action will be applied to the resource and can't be deferred indefinitely.
- **available** – The maintenance action is available, but it will not be applied to the resource automatically. You can apply it manually.

- **next window** – The maintenance action will be applied to the resource during the next maintenance window.
- **In progress** – The maintenance action is in the process of being applied to the resource.

If an update is available, you can take one of the actions:

- If the maintenance value is **next window**, defer the maintenance items by choosing **Defer upgrade** from **Actions**. You can't defer a maintenance action if it has already started.
- Apply the maintenance items immediately.
- Schedule the maintenance items to start during your next maintenance window.
- Take no action.

To take an action, choose the DB instance to show its details, then choose **Maintenance & backups**. The pending maintenance items appear.



The maintenance window determines when pending operations start, but doesn't limit the total run time of these operations. Maintenance operations aren't guaranteed to finish before the maintenance window ends, and can continue beyond the specified end time.

Applying updates for a DB instance

With Amazon RDS, you can choose when to apply maintenance operations. You can decide when Amazon RDS applies updates by using the RDS con-

sole, AWS Command Line Interface (AWS CLI), or RDS API.

Console

To manage an update for a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console.
2. In the navigation pane, choose **Databases**.
3. Choose the DB instance that has a required update.
4. For **Actions**, choose one of the following:
    - **Upgrade now**
    - **Upgrade at next window**

Maintenance for Multi-AZ deployments

Running a DB instance as a Multi-AZ deployment can further reduce the impact of a maintenance event, because Amazon RDS applies operating system updates by following these steps:

1. Perform maintenance on the standby.
2. Promote the standby to primary.
3. Perform maintenance on the old primary, which becomes the new standby.

When you modify the database engine for your DB instance in a Multi-AZ deployment, then Amazon RDS upgrades both the primary and secondary DB instances at the same time. In this case, the database engine for the entire Multi-AZ deployment is shut down during the upgrade.

The Amazon RDS maintenance window

Every DB instance has a weekly maintenance window during which any system changes are applied. You can think of the maintenance window as an opportunity to control when modifications and software patching occur, in the event either are requested or required. If a maintenance event

is scheduled for a given week, it is initiated during the 30-minute maintenance window you identify. Most maintenance events also complete during the 30-minute maintenance window, although larger maintenance events may take more than 30 minutes to complete.

The 30-minute maintenance window is selected at random from an 8-hour block of time per region. If you don't specify a preferred maintenance window when you create the DB instance, then Amazon RDS assigns a 30-minute maintenance window on a randomly selected day of the week.

RDS will consume some of the resources on your DB instance while maintenance is being applied. You might observe a minimal effect on performance. For a DB instance, on rare occasions, a Multi-AZ failover might be required for a maintenance update to complete.

Following, you can find the time blocks for each region from which default maintenance windows are assigned.

Region Name	Region	Time Block
US East (Ohio)	us-east-2	03:00–11:00 UTC
US East (N. Virginia)	us-east-1	03:00–11:00 UTC
US West (N. California)	us-west-1	06:00–14:00 UTC
US West (Oregon)	us-west-2	06:00–14:00 UTC
Africa (Cape Town)	af-south-1	03:00–11:00 UTC
Asia Pacific (Hong Kong)	ap-east-1	06:00–14:00 UTC

Asia Pacific (Mumbai)	ap-south-1	06:00–14:00 UTC
Asia Pacific (Osaka)	ap-north-east-3	22:00–23:59 UTC
Asia Pacific (Seoul)	ap-north-east-2	13:00–21:00 UTC
Asia Pacific (Singapore)	ap-south-east-1	14:00–22:00 UTC
Asia Pacific (Sydney)	ap-south-east-2	12:00–20:00 UTC
Asia Pacific (Tokyo)	ap-north-east-1	13:00–21:00 UTC
Canada (Central)	ca-central-1	03:00–11:00 UTC
China (Beijing)	cn-north-1	06:00–14:00 UTC
China (Ningxia)	cn-north-	06:00–

	west-1	14:00 UTC
Europe (Frankfurt)	eu-central-1	21:00–05:00 UTC
Europe (Ireland)	eu-west-1	22:00–06:00 UTC
Europe (London)	eu-west-2	22:00–06:00 UTC
Europe (Paris)	eu-west-3	23:59–07:29 UTC
Europe (Milan)	eu-south-1	02:00–10:00 UTC
Europe (Stockholm)	eu-north-1	23:00–07:00 UTC
Middle East (Bahrain)	me-south-1	06:00–14:00 UTC
South America (São Paulo)	sa-east-1	00:00–08:00 UTC

AWS GovCloud (US-East)	us-gov-east-1	17:00–01:00 UTC
AWS GovCloud (US-West)	us-gov-west-1	06:00–14:00 UTC

Adjusting the preferred DB instance maintenance window

The maintenance window should fall at the time of lowest usage and thus might need modification from time to time. Your DB instance will only be unavailable during this time if the system changes, such as a change in DB instance class, are being applied and require an outage, and only for the minimum amount of time required to make the necessary changes.

In the following example, you adjust the preferred maintenance window for a DB instance.

For the purpose of this example, we assume that the DB instance named *mydbinstance* exists and has a preferred maintenance window of "Sun:05:00-Sun:06:00" UTC.

Console

To adjust the preferred maintenance window

1. Sign in to the AWS Management Console and open the Amazon RDS console.
2. In the navigation pane, choose **Databases**, and then select the DB instance that you want to modify.
3. Choose **Modify**. The **Modify DB Instance** page appears.
4. In the **Maintenance** section, update the maintenance window.
5. Choose **Continue**.

On the confirmation page, review your changes.

6. To apply the changes to the maintenance window immediately, select **Apply immediately**.
7. Choose **Modify DB Instance** to save your changes.

Alternatively, choose **Back** to edit your changes, or choose **Cancel** to cancel your changes.

AWS CLI

To adjust the preferred maintenance window, use the AWS CLI `modify-db-instance` command with the following parameters:

- `--db-instance-identifier`
- `--preferred-maintenance-window`

Example

The following code example sets the maintenance window to Tuesdays from 4:00-4:30AM UTC.

For Linux, macOS, or Unix:

```
aws rds modify-db-instance \
```

```
--db-instance-identifier mydbinstance \
--preferred-maintenance-window Tue:04:00-
Tue:04:30
```

For Windows:

```
aws rds modify-db-instance ^
--db-instance-identifier mydbinstance ^
--preferred-maintenance-window Tue:04:00-
Tue:04:30
```

RDS API

To adjust the preferred maintenance window, use the Amazon RDS API `ModifyDBInstance` operation with the following parameters:

- `DBInstanceIdentifier`
- `PreferredMaintenanceWindow`

Upgrading a DB instance engine version

Amazon RDS provides newer versions of each supported database engine so you can keep your DB instance up-to-date. Newer versions can include bug fixes, security enhancements, and other improvements for the database engine. When Amazon RDS supports a new version of a database engine, you can choose how and when to upgrade your database DB instances.

There are two kinds of upgrades: major version upgrades and minor version upgrades. In general, a *major engine version upgrade* can introduce changes that are not compatible with existing applications. In contrast, a *minor version upgrade* includes only changes that are backward-compatible with existing applications.

The version numbering sequence is specific to each database engine. For example, RDS for MySQL 5.7

and 8.0 are major engine versions and upgrading from any 5.7 version to any 8.0 version is a major version upgrade. RDS for MySQL version 5.7.22 and 5.7.23 are minor versions and upgrading from 5.7.22 to 5.7.23 is a minor version upgrade.

For major version upgrades, you must manually modify the DB engine version through the AWS Management Console, AWS CLI, or RDS API. For minor version upgrades, you can manually modify the engine version, or you can choose to enable auto minor version upgrades.

Manually upgrading the engine version

To manually upgrade the engine version of a DB instance, you can use the AWS Management Console, the AWS CLI, or the RDS API.

Console

To upgrade the engine version of a DB instance by using the console

1. Sign in to the AWS Management Console and open the Amazon RDS console.
2. In the navigation pane, choose **Databases**, and then choose the DB instance that you want to upgrade.
3. Choose **Modify**. The **Modify DB Instance** page appears.
4. For **DB engine version**, choose the new version.
5. Choose **Continue** and check the summary of modifications.
6. To apply the changes immediately, choose **Apply immediately**. Choosing this option can cause an outage in some cases.
7. On the confirmation page, review your changes. If they are correct, choose **Modify DB Instance** to save your changes.

Alternatively, choose **Back** to edit your changes, or choose **Cancel** to cancel your changes.

AWS CLI

To upgrade the engine version of a DB instance, use the CLI modify-db-instance command. Specify the following parameters:

- `--db-instance-identifier` – the name of the DB instance.
- `--engine-version` – the version number of the database engine to upgrade to.

For information about valid engine versions, use the AWS CLI describe-db-engine-versions command.

- `--allow-major-version-upgrade` – to upgrade the major version.
- `--no-apply-immediately` – to apply changes during the next maintenance window. To apply changes immediately, use `--apply-immediately`.

Example

For Linux, macOS, or Unix:

```
aws rds modify-db-instance \
  --db-instance-identifier mydbinstance \
  --engine-version new_version \
  --allow-major-version-upgrade \
  --no-apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
  --db-instance-identifier mydbinstance ^
  --engine-version new_version ^
  --allow-major-version-upgrade ^
  --no-apply-immediately
```

RDS API

To upgrade the engine version of a DB instance, use the ModifyDBInstance action. Specify the following parameters:

- DBInstanceIdentifier – the name of the DB instance, for example *mydbinstance*.
- EngineVersion – the version number of the database engine to upgrade to. For information about valid engine versions, use the DescribeDBEngineVersions operation.
- AllowMajorVersionUpgrade – whether to allow a major version upgrade. To do so, set the value to true.
- ApplyImmediately – whether to apply changes immediately or during the next maintenance window. To apply changes immediately, set the value to true. To apply changes during the next maintenance window, set the value to false.

Automatically upgrading the

minor engine version

A *minor engine version* is an update to a DB engine version within a major engine version. For example, a major engine version might be 9.6 with the minor engine versions 9.6.11 and 9.6.12 within it.

If you want Amazon RDS to upgrade the DB engine version of a database automatically, you can enable auto minor version upgrades for the database.

When Amazon RDS designates a minor engine version as the preferred minor engine version, each database that meets both of the following conditions is upgraded to the minor engine version automatically:

- The database is running a minor version of the DB engine that is lower than the preferred minor engine version.
- The database has auto minor version upgrade enabled.

When you perform these tasks, you can control whether auto minor version upgrade is enabled for the DB instance in the following ways:

- Using the console, set the **Auto minor version upgrade** option.
- Using the AWS CLI, set the `--auto-minor-version-upgrade|--no-auto-minor-version-upgrade` option.
- Using the RDS API, set the `AutoMinorVersionUpgrade` parameter.

To determine whether a maintenance update, such as a DB engine version upgrade, is available for your DB instance, you can use the console, AWS CLI, or RDS API. You can also upgrade the DB engine version manually and adjust the maintenance window.

Upgrading the Oracle DB engine

When Amazon RDS supports a new version of Oracle, you can upgrade your DB instances to the new version.

Overview of Oracle DB engine upgrades

Before upgrading your Oracle DB instance, familiarize yourself with the following concepts.

Major and minor version upgrades

Amazon RDS supports the following upgrades to an Oracle DB instance:

- Major version upgrades

In general, a *major version upgrade* for a database engine can introduce changes that aren't compatible with existing applications. To upgrade your DB instance to a major version, you must perform the action manually.

- Minor version upgrades

A *minor version upgrade* includes only changes that are backward-compatible with existing applications. If you enable auto minor version upgrades on your DB instance, minor version upgrades occur automatically. In all other cases, you upgrade the DB instance manually.

When you upgrade the DB engine, an outage occurs. The duration of the outage depends on your engine version and instance size.

Oracle engine version management

With DB engine version management, you control when and how the database engine is patched and upgraded. You get the flexibility to maintain compatibility with database engine patch versions. You can also test new patch versions to ensure they work with your application before deploying them

in production. In addition, you upgrade the versions on your own terms and timelines.

Automatic snapshots during engine upgrades

During upgrades of an Oracle DB instance, snapshots offer protection against upgrade issues. If the backup retention period for your DB instance is greater than 0, Amazon RDS takes the following DB snapshots during the upgrade:

1. A snapshot of the DB instance before any upgrade changes have been made. If the upgrade fails, you can restore this snapshot to create a DB instance running the old version.
2. A snapshot of the DB instance after the upgrade completes.

Oracle upgrades in a Multi-AZ deployment

If your DB instance is in a Multi-AZ deployment, Amazon RDS upgrades both the primary and

standby replicas. If no operating system updates are required, the primary and standby upgrades occur simultaneously. The instances are not available until the upgrade completes.

If operating system updates are required in a Multi-AZ deployment, Amazon RDS applies the updates when you request the DB upgrade. Amazon RDS performs the following steps:

1. Updates the operating system on the standby DB instance.
2. Upgrades the standby DB instance.
3. Fails over the primary instance to the standby DB instance.
4. Upgrades the operating system on the new standby DB instance, which was formerly the primary instance.
5. Upgrades the new standby DB instance.

Oracle upgrades of read replicas

The Oracle DB engine version of the source DB instance and all of its read replicas must be the same. Amazon RDS performs the upgrade in the following stages:

1. Upgrades the source DB instance. The read replicas are available during this stage.
2. Upgrades the read replicas in parallel, regardless of the replica maintenance windows. The source DB is available during this stage.

For major version upgrades of cross-Region read replicas, Amazon RDS performs additional actions:

- Generates an option group for the target version automatically
- Copies all options and option settings from the original option group to the new option group
- Associates the upgraded cross-Region read replica with the new option group

Oracle upgrades of micro DB instances

We don't recommend upgrading databases running on micro DB instances. Because these instances have limited CPU, the upgrade can take hours to complete.

You can upgrade micro DB instances with small amounts of storage (10–20 GiB) by copying your data using Data Pump. Before you migrate your production DB instances, we recommend that you test by copying data using Data Pump.

Major version upgrades

Amazon RDS supports the following major version upgrades.

To perform a major version upgrade, modify the DB instance manually. Major version upgrades don't occur automatically.

Supported versions for major upgrades

Amazon RDS supports the following major version upgrades.

Current version	Upgrade supported
18.0.0.0	19.0.0.0
12.2.0.1	19.0.0.0 18.0.0.0
12.1.0.2	19.0.0.0 18.0.0.0 12.2.0.1

A major version upgrade of Oracle Database must upgrade to a Release Update (RU) that was released in the same month or later. Major version downgrades aren't supported for any Oracle versions.

Supported instance classes for major upgrades

Your current Oracle DB instance might run on a DB instance class that isn't supported for the version to which you are upgrading. In this case, before you upgrade, migrate the DB instance to a supported DB instance class.

Gathering statistics before major upgrades

Before you perform a major version upgrade, Oracle recommends that you gather optimizer statistics on the DB instance that you are upgrading. This action can reduce DB instance downtime during the upgrade.

To gather optimizer statistics, connect to the DB instance as the master user, and run the DBMS_STATS.GATHER_DICTIONARY_STATS procedure, as in the following example.

```
EXEC DBMS_STATS.GATHER_DICTIONARY_STATS;
```

Allowing major upgrades

A major engine version upgrade might be incompatible with your application. The upgrade is irreversible. If you specify a major version for the EngineVersion parameter that is different from the current major version, you must allow major version upgrades.

If you upgrade a major version using the CLI command modify-db-instance, specify `--allow-major-version-upgrade`. This setting isn't persistent, so you must specify `--allow-major-version-upgrade` whenever you perform a major upgrade. This parameter has no impact on upgrades of minor engine versions.

If you upgrade a major version using the console, you don't need to choose an option to allow the upgrade. Instead, the console displays a warning that major upgrades are irreversible.

Oracle minor version upgrades

A minor version upgrade applies an Oracle Database Patch Set Update (PSU) or Release Update (RU) in a major version.

An Amazon RDS for Oracle DB instance is scheduled to be upgraded automatically during its next maintenance window when it meets the following conditions:

- The DB instance has the **Auto minor version upgrade** option enabled.
- The DB instance is not running the latest minor DB engine version.

The DB instance is upgraded to the latest quarterly PSU or RU four to six weeks after it is made available by Amazon RDS for Oracle.

The following minor version upgrades aren't supported.

Current version	Upgrade not supported
12.1.0.2.v6	12.1.0.2.v7
12.1.0.2.v5	12.1.0.2.v7
12.1.0.2.v5	12.1.0.2.v6

Oracle SE2 upgrade paths

The following table shows supported upgrade paths to Standard Edition Two (SE2).

Your existing configuration	Supported SE2 configuration
12.2.0.1 SE2, BYOL	12.2.0.1 SE2, BYOL or License Included

12.1.0.2 SE2, BYOL	12.2.0.1 SE2, BYOL or License Included 12.1.0.2 SE2, BYOL or License Included

To upgrade from your existing configuration to a supported SE2 configuration, use a supported upgrade path.

Considerations for Oracle DB upgrades

Before upgrading, review the implications for option groups, parameter groups, and time zones.

Option group considerations

If your DB instance uses a custom option group, sometimes Amazon RDS can't automatically assign a new option group. For example, this situation occurs when you upgrade to a new major version.

In such cases, specify a new option group when you upgrade. We recommend that you create a new option group, and add the same options to it as in your existing custom option group.

If your DB instance uses a custom option group that contains the APEX option, you can sometimes reduce the upgrade time. To do this, upgrade your version of APEX at the same time as your DB instance.

Parameter group considerations

If your DB instance uses a custom parameter group, sometimes Amazon RDS can't automatically assign your DB instance a new parameter group. For example, this situation occurs when you upgrade to a new major version. In such cases, make sure to specify a new parameter group when you upgrade. We recommend that you create a new parameter group, and configure the parameters as in your existing custom parameter group.

Time zone considerations

You can use the time zone option to change the *system time zone* used by your Oracle DB instance. For example, you might change the time zone of a DB instance to be compatible with an on-premises environment, or a legacy application. The time zone option changes the time zone at the host level. Amazon RDS for Oracle updates the system time zone automatically throughout the year.

When you create an Oracle DB instance, the database automatically sets the *database time zone*. The database time zone is also known as the Daylight Saving Time (DST) time zone. The database time zone is distinct from the system time zone.

Between Oracle Database releases, patch sets or individual patches may include new DST versions. These patches reflect the changes in transition rules for various time zone regions. For example, a

government might change when DST takes effect. Changes to DST rules may affect existing data of the TIMESTAMP WITH TIME ZONE data type.

If you upgrade an RDS for Oracle instance, Amazon RDS does not upgrade the database time zone automatically. To upgrade the database time zone manually, create a new Oracle DB instance that has the desired DST patch. Then migrate the data from your current instance to the new instance. You can migrate data using several techniques, including the following:

- Oracle GoldenGate
- AWS Database Migration Service
- Oracle Data Pump
- Original Export/Import (desupported for general use)

Preparing for the automatic upgrade of Oracle 18c

On July 1, 2021, Amazon RDS plans to begin automatically upgrading Oracle 18c instances to Oracle 19c. The automatic upgrades are not guaranteed to occur in your maintenance window. All Oracle 18c instances, including reserved instances, will move to the latest available Release Update (RU).

Before the automatic upgrades begin, we highly recommend that you upgrade your existing 18c DB instances to version 19c manually. When you upgrade manually, you can validate that your applications work correctly. To avoid the automatic upgrade, use one of the following strategies before July 1, 2021.

Upgrade your Oracle 18c DB instance

You can upgrade your Oracle version 18c DB instance to Oracle version 19c. Before upgrading, consider the following:

- Your SQL statements might perform differently after the upgrade. If so, you can use the OPTIMIZER_FEATURES_ENABLE parameter to retain the behavior of the 18c optimizer.
- If you have Extended Support for Oracle 18c on the BYOL model, consider the implications. In this case, you must have Extended Support agreements from Oracle Support for Oracle 19c.

Upgrade your Oracle 18c DB snapshots

You can upgrade your existing snapshots to Oracle 19c, and then restore them.

If you plan to upgrade using snapshots, the planned deadline to avoid the automatic upgrade is June 30, 2021.

Downgrade your Oracle 18c DB instance

You may decide not to upgrade your DB instances to Oracle 19c. In this case, you can downgrade your instance to Oracle Database version 12.1.0.2 or 12.2.0.1. Use any of the following techniques:

- Oracle Data Pump
- AWS Database Migration Service (DMS)
- Any supported logical replication tool

Testing an Oracle DB upgrade

Before you upgrade your DB instance to a major version, thoroughly test your database and all applications that access the database for compatibility with the new version. We recommend that you use the following procedure.

To test a major version upgrade

1. Review the Oracle upgrade documentation for the new version of the database engine

to see if there are compatibility issues that might affect your database or applications.

2. If your DB instance uses a custom option group, create a new option group compatible with the new version you are upgrading to.
3. If your DB instance uses a custom parameter group, create a new parameter group compatible with the new version you are upgrading to.
4. Create a DB snapshot of the DB instance to be upgraded.
5. Restore the DB snapshot to create a new test DB instance.
6. Modify this new test DB instance to upgrade it to the new version, by using one of the following methods:
   - Console
   - AWS CLI
   - RDS API
7. Perform testing:

- Run as many of your quality assurance tests against the upgraded DB instance as needed to ensure that your database and application work correctly with the new version.
- Implement any new tests needed to evaluate the impact of any compatibility issues that you identified in step 1.
- Test all stored procedures, functions, and triggers.
- Direct test versions of your applications to the upgraded DB instance. Verify that the applications work correctly with the new version.
- Evaluate the storage used by the upgraded instance to determine if the upgrade requires additional storage. You might need to choose a larger instance class to support the new version in production.

8. If all tests pass, upgrade your production DB instance. We recommend that you confirm that the DB instance working correctly before allowing write operations to the DB instance.

Upgrading an Oracle DB snapshot

If you have existing manual DB snapshots, you can upgrade them to a later version of the Oracle database engine.

When Oracle stops providing patches for a version, and Amazon RDS deprecates the version, you can upgrade your snapshots that correspond to the deprecated version.

The following snapshot upgrades are currently supported.

Current snapshot version	Supported snapshot upgrade

12.1.0.1	12.1.0.2.v8
11.2.0.4	12.1.0.2, 12.2.0.1, 18c, and 19c when the following conditions are met: • The minor version is not a downgrade. • You upgrade the snapshot to the RU, RUR, or PSU for July 2020 or later.
11.2.0.3	11.2.0.4.v11
11.2.0.2	11.2.0.4.v12

Amazon RDS supports upgrading snapshots in all AWS Regions.

Console

To upgrade an Oracle DB snapshot

1. Sign in to the AWS Management Console and open the Amazon RDS console.
2. In the navigation pane, choose **Snapshots**, and then select the DB snapshot that you want to upgrade.
3. For **Actions**, choose **Upgrade snapshot**. The **Upgrade snapshot** page appears.
4. Choose the **New engine version** to upgrade the snapshot to.
5. (Optional) For **Option group**, choose the option group for the upgraded DB snapshot. The same option group considerations apply when upgrading a DB snapshot as when upgrading a DB instance.
6. Choose **Save changes** to save your changes.

During the upgrade process, all snapshot actions are disabled for this DB snapshot. Also, the DB snapshot status changes from **available** to **upgrading**, and then changes to **active** upon completion. If the DB snapshot can't be upgraded because of snapshot

corruption issues, the status changes to **unavailable**. You can't recover the snapshot from this state.

AWS CLI

To upgrade an Oracle DB snapshot by using the AWS CLI, call the modify-db-snapshot command with the following parameters:

- `--db-snapshot-identifier` – The name of the DB snapshot.
- `--engine-version` – The version to upgrade the snapshot to.

You might also need to include the following parameter. The same option group considerations apply when upgrading a DB snapshot as when upgrading a DB instance.

- `--option-group-name` – The option group for the upgraded DB snapshot.

Example

The following example upgrades a DB snapshot.

For Linux, macOS, or Unix:

```
aws rds modify-db-snapshot \
  --db-snapshot-identifier mydbsnapshot \
  --engine-version 11.2.0.4.v12 \
  --option-group-name default:oracle-se1-11-2
```

For Windows:

```
aws rds modify-db-snapshot ^
  --db-snapshot-identifier mydbsnapshot ^
  --engine-version 11.2.0.4.v12 ^
  --option-group-name default:oracle-se1-11-2
```

RDS API

To upgrade an Oracle DB snapshot by using the Amazon RDS API, call the ModifyDBSnapshot operation with the following parameters:

- DBSnapshotIdentifier – The name of the DB snapshot.
- EngineVersion – The version to upgrade the snapshot to.

You might also need to include the OptionGroupName parameter. The same option group considerations apply when upgrading a DB snapshot as when upgrading a DB instance.

Renaming a DB instance

You can rename a DB instance by using the AWS Management Console, the AWS CLI modify-db-instance command, or the Amazon RDS API ModifyDBInstance action. Renaming a DB instance can have far-reaching effects. The following is a list of considerations before you rename a DB instance.

- When you rename a DB instance, the endpoint for the DB instance changes, because the URL includes the name you assigned to the DB instance. You should always redirect traffic from the old URL to the new one.
- When you rename a DB instance, the old DNS name that was used by the DB instance is immediately deleted, although it could remain cached for a few minutes. The new DNS name for the renamed DB instance becomes effective in about 10 minutes. The renamed

DB instance is not available until the new name becomes effective.

- You cannot use an existing DB instance name when renaming an instance.
- All read replicas associated with a DB instance remain associated with that instance after it is renamed. For example, suppose you have a DB instance that serves your production database and the instance has several associated read replicas. If you rename the DB instance and then replace it in the production environment with a DB snapshot, the DB instance that you renamed will still have the read replicas associated with it.
- Metrics and events associated with the name of a DB instance are maintained if you reuse a DB instance name. For example, if you promote a read replica and rename it to be the name of the previous primary DB instance, the events and metrics associated with the

primary DB instance are associated with the renamed instance.

- DB instance tags remain with the DB instance, regardless of renaming.
- DB snapshots are retained for a renamed DB instance.

Renaming to replace an existing DB instance

The most common reasons for renaming a DB instance are that you are promoting a read replica or you are restoring data from a DB snapshot or point-in-time recovery (PITR). By renaming the database, you can replace the DB instance without having to change any application code that references the DB instance. In these cases, you would do the following:

1. Stop all traffic going to the primary DB instance. This can involve redirecting traffic

from accessing the databases on the DB instance or some other way you want to use to prevent traffic from accessing your databases on the DB instance.

2. Rename the primary DB instance to a name that indicates it is no longer the primary DB instance as described later in this topic.
3. Create a new primary DB instance by restoring from a DB snapshot or by promoting a read replica, and then give the new instance the name of the previous primary DB instance.
4. Associate any read replicas with the new primary DB instance.

If you delete the old primary DB instance, you are responsible for deleting any unwanted DB snapshots of the old primary DB instance.

Console

To rename a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Databases**.
3. Choose the DB instance that you want to rename.
4. Choose **Modify**.
5. In **Settings**, enter a new name for **DB instance identifier**.
6. Choose **Continue**.
7. To apply the changes immediately, choose **Apply immediately**. Choosing this option can cause an outage in some cases.
8. On the confirmation page, review your changes. If they are correct, choose **Modify DB Instance** to save your changes.

Alternatively, choose **Back** to edit your changes, or choose **Cancel** to cancel your changes.

Rebooting a DB instance

You might need to reboot your DB instance, usually for maintenance reasons. For example, if you make certain modifications, or if you change the DB parameter group associated with the DB instance , you must reboot the instance for the changes to take effect.

If a DB instance isn't using the latest changes to its associated DB parameter group, the AWS Management Console shows the DB parameter group with a status of **pending-reboot**. The **pending-reboot** parameter groups status doesn't result in an automatic reboot during the next maintenance window. To apply the latest parameter changes to that DB instance, manually reboot the DB instance.

Rebooting a DB instance restarts the database engine service. Rebooting a DB instance results in a

momentary outage, during which the DB instance status is set to *rebooting*.

If the Amazon RDS instance is configured for Multi-AZ, you can perform the reboot with a failover. An Amazon RDS event is created when the reboot is completed. If your DB instance is a Multi-AZ deployment, you can force a failover from one Availability Zone (AZ) to another when you reboot. When you force a failover of your DB instance, Amazon RDS automatically switches to a standby replica in another Availability Zone, and updates the DNS record for the DB instance to point to the standby DB instance. As a result, you need to clean up and re-establish any existing connections to your DB instance. Rebooting with failover is beneficial when you want to simulate a failure of a DB instance for testing, or restore operations to the original AZ after a failover occurs.

When you force a failover from one Availability Zone to another when you reboot, the Availability Zone change might not be reflected in the AWS Management Console, and in calls to the AWS CLI and RDS API, for several minutes.

You can't reboot your DB instance if it is not in the available state. Your database can be unavailable for several reasons, such as an in-progress backup, a previously requested modification, or a maintenance-window action.

The time required to reboot your DB instance depends on the crash recovery process, database activity at the time of reboot, and the behavior of your specific DB engine. To improve the reboot time, we recommend that you reduce database activity as much as possible during the reboot process. Reducing database activity reduces rollback activity for in-transit transactions.

For a DB instance with read replicas, you can reboot the source DB instance and its read replicas independently. After a reboot completes, replication resumes automatically.

Console

To reboot a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Databases**, and then choose the DB instance that you want to reboot.
3. For **Actions**, choose **Reboot**.

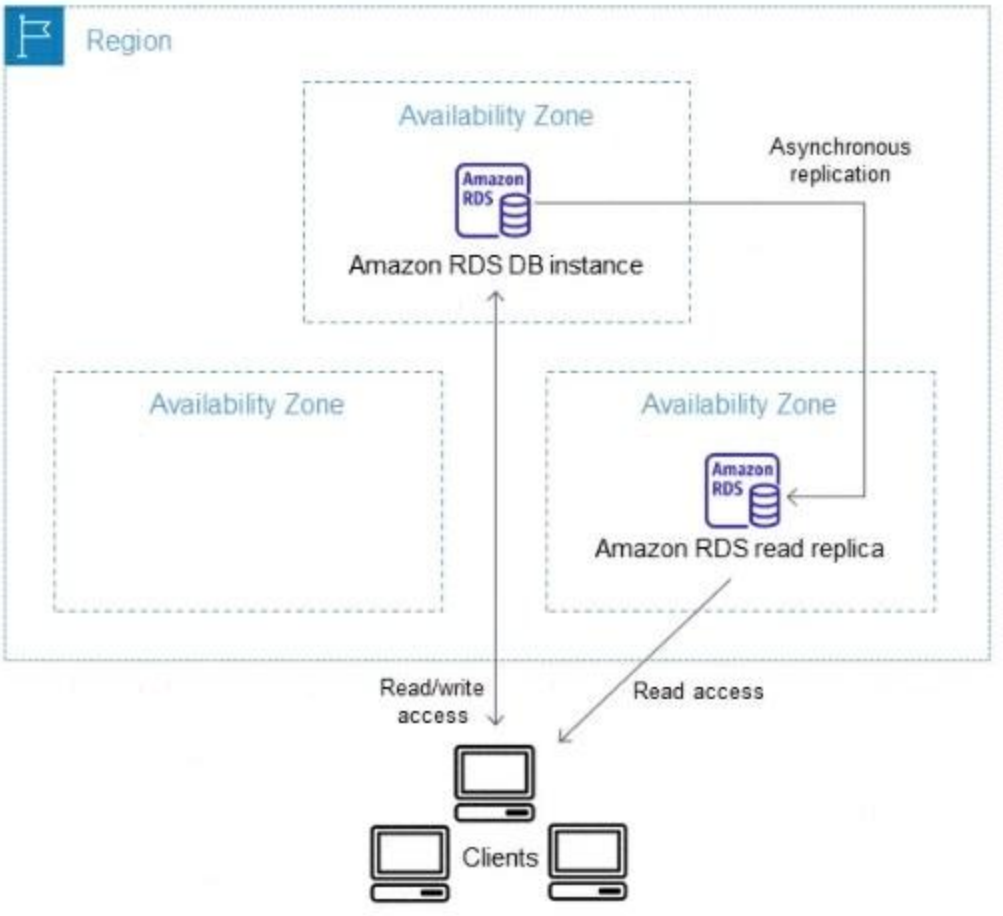
The **Reboot DB Instance** page appears.

4. (Optional) Choose **Reboot with failover?** to force a failover from one AZ to another.
5. Choose **Reboot** to reboot your DB instance.

Alternatively, choose **Cancel**.

Working with read replicas

Amazon RDS uses the Oracle DB engines' built-in replication functionality to create a special type of DB instance called a read replica from a source DB instance. The source DB instance becomes the primary DB instance. Updates made to the primary DB instance are asynchronously copied to the read replica. You can reduce the load on your primary DB instance by routing read queries from your applications to the read replica. Using read replicas, you can elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads.

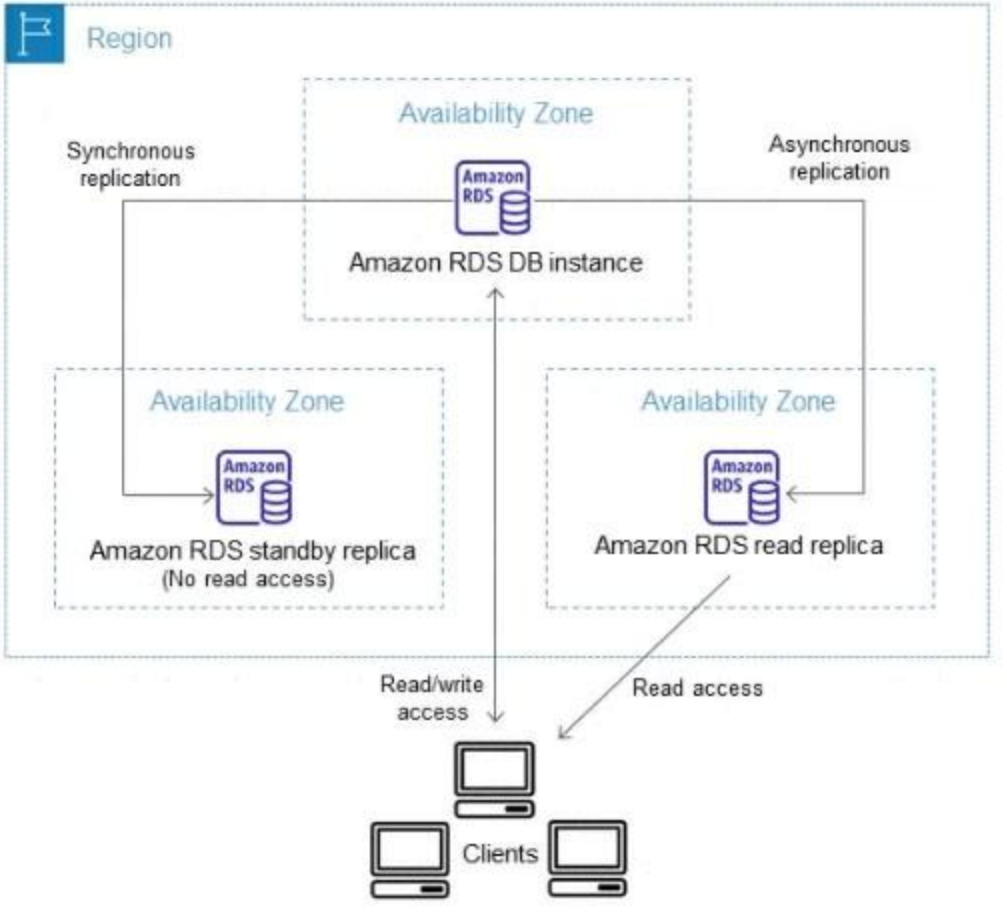


The information following applies to creating Amazon RDS read replicas either in the same AWS Region as the source DB instance, or in a separate AWS Region. The information following doesn't apply to setting up replication with an instance that is running on an Amazon EC2 instance or that is on-premises.

When you create a read replica, you first specify an existing DB instance as the source. Then Amazon RDS takes a snapshot of the source instance and creates a read-only instance from the snapshot. Amazon RDS then uses the asynchronous replication method for the DB engine to update the read replica whenever there is a change to the primary DB instance. The read replica operates as a DB instance that allows only read-only connections. Applications connect to a read replica the same way they do to any DB instance. Amazon RDS replicates all databases in the source DB instance.

In some cases, a read replica resides in a different AWS Region from its primary DB instance. In these cases, Amazon RDS sets up a secure communications channel between the primary DB instance and the read replica. Amazon RDS establishes any AWS security configurations needed to enable the secure channel, such as adding security group entries.

You can configure a read replica for a DB instance that also has a standby replica configured for high availability. Replication with the standby replica is synchronous, and the standby replica can't serve read traffic.



Read replicas are supported by the Oracle DB engines. In this section, you can find general infor-

mation about using read replicas with all of these engines.

Overview of Amazon RDS read replicas

Deploying one or more read replicas for a given source DB instance might make sense in a variety of scenarios, including the following:

- Scaling beyond the compute or I/O capacity of a single DB instance for read-heavy database workloads. You can direct this excess read traffic to one or more read replicas.
- Serving read traffic while the source DB instance is unavailable. In some cases, your source DB instance might not be able to take I/O requests, for example due to I/O suspension for backups or scheduled maintenance. In these cases, you can direct read traffic to your read replicas. For this use case, keep in mind that the data on the read replica might

be "stale" because the source DB instance is unavailable.

- Business reporting or data warehousing scenarios where you might want business reporting queries to run against a read replica, rather than your production DB instance.
- Implementing disaster recovery. You can promote a read replica to a standalone instance as a disaster recovery solution if the primary DB instance fails.

By default, a read replica is created with the same storage type as the source DB instance. However, you can create a read replica that has a different storage type from the source DB instance based on the options listed in the following table.

Source DB instance storage type	Source DB instance storage allocation	Read replica storage type options

PIOPS	100 GiB–32 TiB	PIOPS, GP2, Standard
GP2	100 GiB–32 TiB	PIOPS, GP2, Standard
GP2	<100 GiB	GP2, Standard
Standard	100 GiB–6 TiB	PIOPS, GP2, Standard
Standard	<100 GiB	GP2, Standard

Amazon RDS doesn't support circular replication. You can't configure a DB instance to serve as a replication source for an existing DB instance. You can only create a new read replica from an existing DB instance. For example, if **MyDBInstance** replicates to **ReadReplica1**, you can't configure **ReadReplica1** to replicate back to **MyDBInstance**.
For MariaDB and MySQL you can create a read

replica from an existing read replica. For example, from **ReadReplica1**, you can create a new read replica, such as **ReadReplica2**. For Oracle you can't create a read replica from an existing read replica.

If you no longer need read replicas, you can explicitly delete them using the same mechanisms for deleting a DB instance. If you delete a source DB instance without deleting its read replicas in the same AWS Region, each read replica is promoted to a standalone DB instance.

Differences between read replicas for different DB engines

Because Amazon RDS DB engines implement replication differently, there are several significant differences you should know about, as shown in the following table.

Feature or behavior	Oracle

What is the replication method?	Physical replication.
How are transaction logs purged?	If a primary DB instance has no cross-Region read replicas, Amazon RDS for Oracle keeps a minimum of two hours of transaction logs on the source DB instance. Logs are purged from the source DB instance after two hours or after the archive log retention hours setting has passed, whichever is longer. Logs are purged from the read replica after

	the archive log retention hours setting has passed only if they have been successfully applied to the database. In some cases, a primary DB instance might have one or more cross-Region read replicas. If so, Amazon RDS for Oracle keeps the transaction logs on the source DB instance until they have been transmitted and applied to all cross-Region read replicas.
Can a replica be made	No. An Oracle read

writable?	replica is a physical copy, and Oracle doesn't allow for writes in a read replica. You can promote the read replica to make it writable. The promoted read replica has the replicated data to the point when the request was made to promote it.
Can backups be performed on the replica?	No. You can't create manual snapshots of Amazon RDS for Oracle read replicas or enable automatic backups for them.
Can you use parallel	Yes. Redo log data is

replication?	always transmitted in parallel from the primary database to all of its read replicas.
Can you maintain a replica in a mounted rather than a read-only state?	Yes. The primary use for mounted replicas is cross-Region disaster recovery. An Active Data Guard license isn't required for mounted replicas.

Creating a read replica

You can create a read replica from an existing DB instance using the AWS Management Console, AWS CLI, or RDS API. You create a read replica by specifying SourceDBInstanceIdentifier, which is the DB instance identifier of the source DB instance that you want to replicate from.

When you create a read replica, Amazon RDS takes a DB snapshot of your source DB instance and begins replication. As a result, you experience a brief I/O suspension on your source DB instance while the DB snapshot occurs.

An active, long-running transaction can slow the process of creating the read replica. We recommend that you wait for long-running transactions to complete before creating a read replica. If you create multiple read replicas in parallel from the same source DB instance, Amazon RDS takes only one snapshot at the start of the first create action.

You can't create a read replica in a different AWS account from the source DB instance.

Console

To create a read replica from a source DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Databases**.
3. Choose the DB instance that you want to use as the source for a read replica.
4. For **Actions**, choose **Create read replica**.
5. For **DB instance identifier**, enter a name for the read replica.
6. Choose your instance specifications. We recommend that you use the same DB instance class and storage type as the source DB instance for the read replica.
7. For **Multi-AZ deployment**, choose **Yes** to create a standby of your replica in another Availability Zone for failover support for the replica.
8. To create an encrypted read replica:
    a. Choose **Enable encryption**.

b. For **Master key**, choose the AWS Key Management Service (AWS KMS) key identifier of the customer master key (CMK).

9. Choose other options, such as storage autoscaling.
10. Choose **Create read replica**.

Promoting a read replica to be a

standalone DB instance

You can promote a read replica into a standalone DB instance. When you promote a read replica, the DB instance is rebooted before it becomes available.

There are several reasons you might want to promote a read replica to a standalone DB instance:

- **Performing DDL operations (MySQL and MariaDB only)** – DDL operations, such as creating or rebuilding indexes, can take time and impose a significant performance

penalty on your DB instance. You can perform these operations on a MySQL or MariaDB read replica once the read replica is in sync with its primary DB instance. Then you can promote the read replica and direct your applications to use the promoted instance.

- **Sharding** – Sharding embodies the "share-nothing" architecture and essentially involves breaking a large database into several smaller databases. One common way to split a database is splitting tables that are not joined in the same query onto different hosts. Another method is duplicating a table across multiple hosts and then using a hashing algorithm to determine which host receives a given update. You can create read replicas corresponding to each of your shards (smaller databases) and promote them when you decide to convert them into standalone shards. You can then carve out the key space

(if you are splitting rows) or distribution of tables for each of the shards depending on your requirements.

- **Implementing failure recovery** – You can use read replica promotion as a data recovery scheme if the primary DB instance fails. This approach complements synchronous replication, automatic failure detection, and failover.

If you are aware of the ramifications and limitations of asynchronous replication and you still want to use read replica promotion for data recovery, you can. To do this, first create a read replica and then monitor the primary DB instance for failures. In the event of a failure, do the following:

1. Promote the read replica.
2. Direct database traffic to the promoted DB instance.
3. Create a replacement read replica with the promoted DB instance as its source.

When you promote a read replica, the new DB instance that is created retains the option group and the parameter group of the former read replica. The promotion process can take several minutes or longer to complete, depending on the size of the read replica. After you promote the read replica to a new DB instance, it's just like any other DB instance. For example, you can create read replicas from the new DB instance and perform point-in-time restore operations. Because the promoted DB instance is no longer a read replica, you can't use it as a replication target. If a source DB instance has several read replicas, promoting one of the read replicas to a DB instance has no effect on the other replicas.

Backup duration is a function of the number of changes to the database since the previous backup. If you plan to promote a read replica to a standalone instance, we recommend that you enable backups and complete at least one backup prior to promo-

tion. In addition, you can't promote a read replica to a standalone instance when it has the backing-up status. If you have enabled backups on your read replica, configure the automated backup window so that daily backups don't interfere with read replica promotion.

The following steps show the general process for promoting a read replica to a DB instance:

1. Stop any transactions from being written to the primary DB instance, and then wait for all updates to be made to the read replica. Database updates occur on the read replica after they have occurred on the primary DB instance, and this replication lag can vary significantly. Use the Replica Lag metric to determine when all updates have been made to the read replica.
2. For MySQL and MariaDB only: If you need to make changes to the MySQL or MariaDB read

replica, you must set the read_only parameter to 0 in the DB parameter group for the read replica. You can then perform all needed DDL operations, such as creating indexes, on the read replica. Actions taken on the read replica don't affect the performance of the primary DB instance.

3. Promote the read replica by using the **Promote** option on the Amazon RDS console, the AWS CLI command promote-read-replica, or the PromoteReadReplica Amazon RDS API operation.
4. (Optional) Modify the new DB instance to be a Multi-AZ deployment.

Console

To promote a read replica to a standalone DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console.

2. In the Amazon RDS console, choose **Databases**.

The **Databases** pane appears. Each read replica shows **Replica** in the **Role** column.

3. Choose the read replica that you want to promote.
4. For **Actions**, choose **Promote**.
5. On the **Promote Read Replica** page, enter the backup retention period and the backup window for the newly promoted DB instance.
6. When the settings are as you want them, choose **Continue**.
7. On the acknowledgment page, choose **Promote Read Replica**.

Monitoring read replication

You can monitor the status of a read replica in several ways. The Amazon RDS console shows the status of a read replica in the **Availability and dura-**

bility section of the read replica details. To view the details for a read replica, choose the name of the read replica in the list of instances in the Amazon RDS console.

You can also see the status of a read replica using the AWS CLI `describe-db-instances` command or the Amazon RDS API `DescribeDBInstances` operation.

The status of a read replica can be one of the following:

- **replicating** – The read replica is replicating successfully.

- **replication degraded (SQL Server only) –** Replicas are receiving data from the primary instance, but one or more databases might be not getting updates. This can occur, for example, when a replica is in the process of setting up newly created databases.

The status doesn't transition from replication degraded to error, unless an error occurs during the degraded state.

- **error** – An error has occurred with the replication. Check the **Replication Error** field in the Amazon RDS console or the event log to determine the exact error.

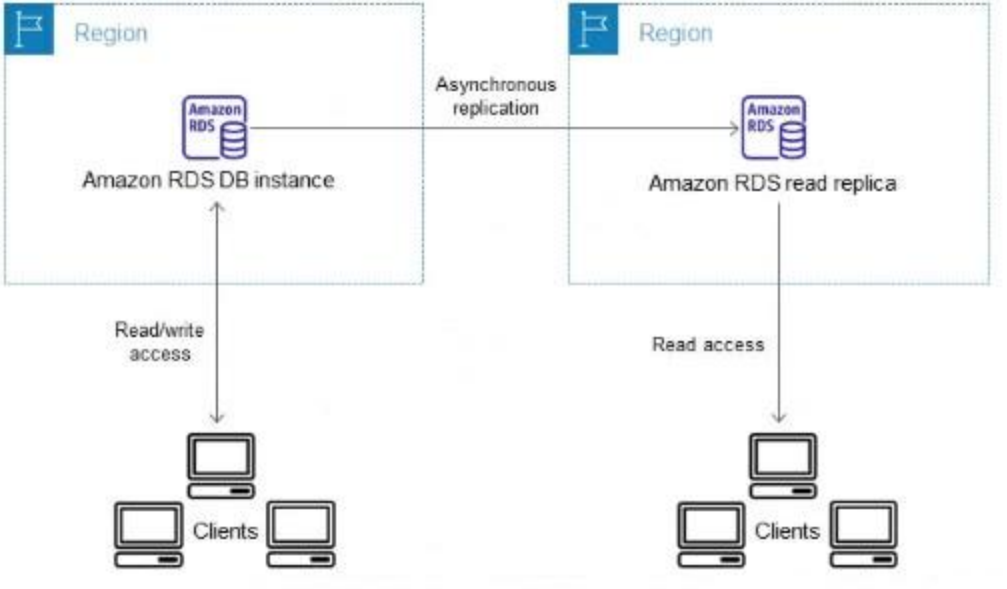
Monitoring replication lag

You can monitor replication lag in Amazon CloudWatch by viewing the Amazon RDS ReplicaLag metric.

ReplicaLag returns -1 if RDS can't determine the lag, such as during replica setup, or when the read replica is in the error state.

Creating a read replica in a different AWS Region

With Amazon RDS, you can create a Oracle read replica in a different AWS Region from the source DB instance.



You create a read replica in a different AWS Region to do the following:

- Improve your disaster recovery capabilities.

- Scale read operations into an AWS Region closer to your users.
- Make it easier to migrate from a data center in one AWS Region to a data center in another AWS Region.

Creating a read replica in a different AWS Region from the source instance is similar to creating a replica in the same AWS Region. You can use the AWS Management Console, run the create-db-instance-read-replica command, or call the CreateDBInstanceReadReplica API operation.

Creating a cross-Region read replica

The following procedures show how to create a read replica from a source Oracle DB instance in a different AWS Region.

Console

You can create a read replica across AWS Regions using the AWS Management Console.

To create a read replica across AWS Regions with the console

1. Sign in to the AWS Management Console and open the Amazon RDS console.
2. In the navigation pane, choose **Databases**.
3. Choose the Oracle DB instance that you want to use as the source for a read replica.
4. For **Actions**, choose **Create read replica**.
5. For **DB instance identifier**, enter a name for the read replica.
6. Choose the **Destination Region.**
7. Choose the instance specifications you want to use. We recommend that you use the same DB instance class and storage type for the read replica.
8. To create an encrypted read replica in another AWS Region:
    a. Choose **Enable encryption**.
    b. For **Master key**, choose the AWS Key Management Service (AWS KMS) key

identifier of the customer master key (CMK) of the destination AWS Region.

9. Choose other options, such as storage autoscaling.
10. Choose **Create read replica**.

How Amazon RDS does cross-Region replication

Amazon RDS uses the following process to create a cross-Region read replica. Depending on the AWS Regions involved and the amount of data in the databases, this process can take hours to complete. You can use this information to determine how far the process has proceeded when you create a cross-Region read replica:

1. Amazon RDS begins configuring the source DB instance as a replication source and sets the status to *modifying*.
2. Amazon RDS begins setting up the specified read replica in the destination AWS Region and sets the status to *creating*.

3. Amazon RDS creates an automated DB snapshot of the source DB instance in the source AWS Region. The format of the DB snapshot name is rds:<InstanceID>-<timestamp>, where <InstanceID> is the identifier of the source instance, and <timestamp> is the date and time the copy started. For example, rds:mysourceinstance-2013-11-14-09-24 was created from the instance mysourceinstance at 2013-11-14-09-24. During the creation of an automated DB snapshot, the source DB instance status remains *modifying*, the read replica status remains *creating*, and the DB snapshot status is *creating*. The progress column of the DB snapshot page in the console reports how far the DB snapshot creation has progressed. When the DB snapshot is complete, the status of both the DB snapshot and source DB instance are set to *available*.

4. Amazon RDS begins a cross-Region snapshot copy for the initial data transfer. The snapshot copy is listed as an automated snapshot in the destination AWS Region with a status of *creating*. It has the same name as the source DB snapshot. The progress column of the DB snapshot display indicates how far the copy has progressed. When the copy is complete, the status of the DB snapshot copy is set to *available*.
5. Amazon RDS then uses the copied DB snapshot for the initial data load on the read replica. During this phase, the read replica is in the list of DB instances in the destination, with a status of *creating*. When the load is complete, the read replica status is set to *available*, and the DB snapshot copy is deleted.
6. When the read replica reaches the available status, Amazon RDS starts by replicating the

changes made to the source instance since the start of the create read replica operation. During this phase, the replication lag time for the read replica is greater than 0.

Cross-Region replication considerations

All of the considerations for performing replication within an AWS Region apply to cross-Region replication. The following extra considerations apply when replicating between AWS Regions:

- You can only replicate between AWS Regions when using the following Amazon RDS DB instances:
    - Oracle Enterprise Edition (EE) engine version 12.1.0.2.v10 and higher 12.1 versions, and all versions of 12.2, 18c, and 19c.

An Active Data Guard license is required.

- A source DB instance can have cross-Region read replicas in multiple AWS Regions.
- You can only create a cross-Region Amazon RDS read replica from a source Amazon RDS DB instance that is not a read replica of another Amazon RDS DB instance.
- You can replicate between the AWS GovCloud (US-East) and AWS GovCloud (US-West) Regions, but not into or out of AWS GovCloud (US).
- You can expect to see a higher level of lag time for any read replica that is in a different AWS Region than the source instance. This lag time comes from the longer network channels between regional data centers.
- For cross-Region read replicas, any of the create read replica commands that specify the `--db-subnet-group-name` parameter must specify a DB subnet group from the same VPC.

- You can create a cross-Region read replica:
  - In a VPC from a source DB instance that is in a VPC in another AWS Region
  - In a VPC from a source DB instance that isn't in a VPC
  - That isn't in a VPC from a source DB instance that is in a VPC
- Due to the limit on the number of access control list (ACL) entries for a VPC, we can't guarantee more than five cross-Region read replica instances.
- The read replica uses the default DB parameter group for the specified DB engine.
- The read replica uses the default security group.
- For Oracle DB instances, when the source for a cross-Region read replica is deleted, the read replica is promoted.

Requesting a cross-Region read replica

To communicate with the source Region to request the creation of a cross-Region read replica, the requester (IAM role or IAM user) must have access to the source DB instance and the source Region.

Certain conditions in the requester's IAM policy can cause the request to fail. The following examples assume that the source DB instance is in US East (Ohio) and the read replica is created in US East (N. Virginia). These examples show conditions in the requester's IAM policy that cause the request to fail:

- The requester's policy has a condition for aws:RequestedRegion.

```
...
"Effect": "Allow",
"Action": "rds:CreateDBInstanceReadReplica",
"Resource": "*",
"Condition": {
  "StringEquals": {
    "aws:RequestedRegion": "us-east-1"
```

```
  }
}
```

The request fails because the policy doesn't allow access to the source Region. For a successful request, specify both the source and destination Regions.

```
...
"Effect": "Allow",
"Action": "rds:CreateDBInstanceReadReplica",
"Resource": "*",
"Condition": {
  "StringEquals": {
    "aws:RequestedRegion": [
      "us-east-1",
      "us-east-2"
    ]
  }
}
```

- The requester's policy doesn't allow access to the source DB instance.

```
...
"Effect": "Allow",
"Action": "rds:CreateDBInstanceReadReplica",
"Resource": "arn:aws:rds:us-
east-1:123456789012:db:myreadreplica"
...
```

For a successful request, specify both the source instance and the replica.

```
...
"Effect": "Allow",
"Action": "rds:CreateDBInstanceReadReplica",
"Resource": [
  "arn:aws:rds:us-
east-1:123456789012:db:myreadreplica",
  "arn:aws:rds:us-
east-2:123456789012:db:mydbinstance"
]
...
```

- The requester's policy denies `aws:ViaAWSService`.

```
...
"Effect": "Allow",
"Action": "rds:CreateDBInstanceReadReplica",
"Resource": "*",
"Condition": {
  "Bool": {"aws:ViaAWSService": "false"}
}
```

Communication with the source Region is made by RDS on the requester's behalf. For a successful request, don't deny calls made by AWS services.

- The requester's policy has a condition for `aws:SourceVpc` or `aws:SourceVpce`.

These requests might fail because when RDS makes the call to the remote Region, it isn't from the specified VPC or VPC endpoint.

If you need to use one of the previous conditions that would cause a request to fail, you can include a second statement with `aws:CalledVia` in your policy to make the request succeed. For example, you can

use aws:CalledVia with aws:SourceVpce as shown here:

```
...
"Effect": "Allow",
"Action": "rds:CreateDBInstanceReadReplica",
"Resource": "*",
"Condition": {
  "Condition" : {
     "ForAnyValue:StringEquals" : {
      "aws:SourceVpce": "vpce-1a2b3c4d"
     }
   }
},
{
  "Effect": "Allow",
  "Action": [
     "rds:CreateDBInstanceReadReplica"
  ],
  "Resource": "*",
  "Condition": {
```

```
    "ForAnyValue:StringEquals": {
      "aws:CalledVia": [
        "rds.amazonaws.com"
      ]
    }
  }
}
```

Authorizing the read replica

After a cross-Region DB read replica creation request returns `success`, RDS starts the replica creation in the background. An authorization for RDS to access the source DB instance is created. This authorization links the source DB instance to the read replica, and allows RDS to copy only to the specified read replica.

The authorization is verified by RDS using the `rds:CrossRegionCommunication` permission in the service-linked IAM role. If the replica is authorized,

RDS communicates with the source Region and completes the replica creation.

RDS doesn't have access to DB instances that weren't authorized previously by a `CreateDBInstanceReadReplica` request. The authorization is revoked when read replica creation completes.

RDS uses the service-linked role to verify the authorization in the source Region. If you delete the service-linked role during the replication creation process, the creation fails.

Using AWS Security Token Service credentials

Session tokens from the global AWS Security Token Service (AWS STS) endpoint are valid only in AWS Regions that are enabled by default (commercial Regions). If you use credentials from the `assumeRole` API operation in AWS STS, use the regional endpoint if the source Region is an opt-in Region. Otherwise, the request fails. This happens

because your credentials must be valid in both Regions, which is true for opt-in Regions only when the regional AWS STS endpoint is used.

To use the global endpoint, make sure that it's enabled for both Regions in the operations. Set the global endpoint to Valid in all AWS Regions in the AWS STS account settings.

The same rule applies to credentials in the pre-signed URL parameter.

Working with storage for Amazon

RDS DB instances

To specify how you want your data stored in Amazon RDS, choose a storage type and provide a storage size when you create or modify a DB instance. Later, you can increase the amount or change the type of storage by modifying the DB instance.

Increasing DB instance storage capacity

If you need space for additional data, you can scale up the storage of an existing DB instance. To do so, you can use the Amazon RDS Management Console, the Amazon RDS API, or the AWS Command Line Interface (AWS CLI).

To monitor the amount of free storage for your DB instance so you can respond when necessary, we recommend that you create an Amazon CloudWatch alarm.

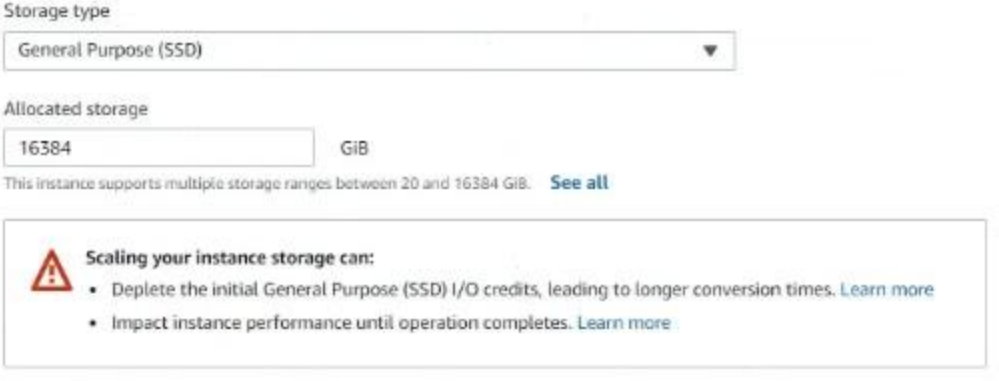
In most cases, scaling storage doesn't require any outage and doesn't degrade performance of the server. After you modify the storage size for a DB instance, the status of the DB instance is **storage-optimization**. The DB instance is fully operational after a storage modification.

However, a special case is if you have a SQL Server DB instance and haven't modified the storage configuration since November 2017. In this case, you might experience a short outage of a few minutes when you modify your DB instance to increase the allocated storage. After the outage, the DB instance is online but in the storage-optimization state. Performance might be degraded during storage optimization.

Console

To increase storage for a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console.
2. In the navigation pane, choose **Databases**.
3. Choose the DB instance that you want to modify.
4. Choose **Modify**.
5. Enter a new value for **Allocated storage**. It must be greater than the current value.



6. Choose **Continue** to move to the next screen.
7. Choose **Apply immediately** in the **Scheduling of modifications** section to apply the storage changes to the DB instance immediately. Or choose **Apply during the next scheduled maintenance window** to apply

the changes during the next maintenance window.

8. When the settings are as you want them, choose **Modify DB instance**.

Managing capacity automatically with

Amazon RDS storage autoscaling

If your workload is unpredictable, you can enable storage autoscaling for an Amazon RDS DB instance. To do so, you can use the Amazon RDS console, the Amazon RDS API, or the AWS CLI.

For example, you might use this feature for a new mobile gaming application that users are adopting rapidly. In this case, a rapidly increasing workload might exceed the available database storage. To avoid having to manually scale up database storage, you can use Amazon RDS storage autoscaling.

With storage autoscaling enabled, when Amazon RDS detects that you are running out of free database space it automatically scales up your storage. Amazon RDS starts a storage modification for an autoscaling-enabled DB instance when these factors apply:

- Free available space is less than 10 percent of the allocated storage.
- The low-storage condition lasts at least five minutes.
- At least six hours have passed since the last storage modification.

The additional storage is in increments of whichever of the following is greater:

- 5 GiB
- 10 percent of currently allocated storage

- Storage growth prediction for 7 hours based on the FreeStorageSpace metrics change in the past hour.

The maximum storage threshold is the limit that you set for autoscaling the DB instance. You can't set the maximum storage threshold for autoscaling-enabled instances to a value greater than the maximum allocated storage.

For example, SQL Server Standard Edition on db.m5.xlarge has a default allocated storage for the instance of 20 GiB (the minimum) and a maximum allocated storage of 16,384 GiB. The default maximum storage threshold for autoscaling is 1,000 GiB. If you use this default, the instance doesn't autoscale above 1,000 GiB. This is true even though the maximum allocated storage for the instance is 16,384 GiB.

The following limitations apply to storage autoscaling:

- Autoscaling doesn't occur if the maximum storage threshold would be exceeded by the storage increment.
- Autoscaling can't completely prevent storage-full situations for large data loads, because further storage modifications can't be made until six hours after storage optimization has completed on the instance. If you perform a large data load, and autoscaling doesn't provide enough space, the database might remain in the storage-full state for several hours. This can harm the database.
- If you start a storage scaling operation at the same time that Amazon RDS starts an autoscaling operation, your storage modification takes precedence. The autoscaling operation is canceled.
- Autoscaling can't be used with magnetic storage.

- Autoscaling can't be used with the following previous-generation instance classes that have less than 6 TiB of orderable storage: db.m3.large, db.m3.xlarge, and db.m3.2xlarge.
- Autoscaling operations aren't logged by AWS CloudTrail.

Although automatic scaling helps you to increase storage on your Amazon RDS DB instance dynamically, you should still configure the initial storage for your DB instance to an appropriate size for your typical workload.

Enabling storage autoscaling for a new DB instance

When you create a new Amazon RDS DB instance, you can choose whether to enable storage autoscaling. You can also set an upper limit on the storage that Amazon RDS can allocate for the DB instance.

Console

To enable storage autoscaling for a new DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console.
2. In the upper-right corner of the Amazon RDS console, choose the AWS Region where you want to create the DB instance.
3. In the navigation pane, choose **Databases**.
4. Choose **Create database**. On the **Select engine** page, choose your database engine and specify your DB instance information.
5. In the **Storage autoscaling** section, set the **Maximum storage threshold** value for the DB instance.
6. Specify the rest of your DB instance information.

Changing the storage autoscaling settings for a DB instance

You can turn storage autoscaling on for an existing Amazon RDS DB instance. You can also change the upper limit on the storage that Amazon RDS can allocate for the DB instance.

Console

To change the storage autoscaling settings for a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console.
2. In the navigation pane, choose **Databases**.
3. Choose the DB instance that you want to modify, and choose **Modify**. The **Modify DB instance** page appears.
4. Change the storage limit in the **Autoscaling** section.
5. When all the changes are as you want them, choose **Continue** and check your modifications.

6. On the confirmation page, review your changes. If they're correct, choose **Modify DB Instance** to save your changes. If they aren't correct, choose **Back** to edit your changes or **Cancel** to cancel your changes.

Changing the storage autoscaling limit occurs immediately. This setting ignores the **Apply immediately** setting.

Turning off storage autoscaling for a DB instance

If you no longer need Amazon RDS to automatically increase the storage for an Amazon RDS DB instance, you can turn off storage autoscaling. After you do, you can still manually increase the amount of storage for your DB instance.

Console

To turn off storage autoscaling for a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console.
2. In the navigation pane, choose **Databases**.
3. Choose the DB instance that you want to modify and choose **Modify**. The **Modify DB instance** page appears.
4. Clear the **Enable storage autoscaling** check box in the **Storage autoscaling** section.
5. When all the changes are as you want them, choose **Continue** and check the modifications.
6. On the confirmation page, review your changes. If they're correct, choose **Modify DB Instance** to save your changes. If they aren't correct, choose **Back** to edit your changes or **Cancel** to cancel your changes.

Changing the storage autoscaling limit occurs immediately. This setting ignores the **Apply immediately** setting.

Modifying SSD storage settings

for Provisioned IOPS

You can modify the settings for a DB instance that uses Provisioned IOPS SSD storage by using the Amazon RDS console, AWS CLI, or Amazon RDS API. Specify the storage type, allocated storage, and the amount of Provisioned IOPS that you require. You can choose from a range between 1,000 IOPS and 100 GiB of storage up to 80,000 IOPS and 64 TiB (64,000 GiB) of storage. The range depends on your database engine and instance type.

Although you can reduce the amount of IOPS provisioned for your instance, you can't reduce the amount of General Purpose SSD or magnetic storage allocated.

In most cases, scaling storage doesn't require any outage and doesn't degrade performance of the

server. After you modify the storage IOPS for a DB instance, the status of the DB instance is **storage-optimization**. The DB instance is fully operational after a storage modification.

Console

To change the Provisioned IOPS settings for a DB instance

1. Sign in to the AWS Management Console and open the Amazon RDS console.
2. In the navigation pane, choose **Databases**.
3. Choose the DB instance with Provisioned IOPS that you want to modify.
4. Choose **Modify**.
5. On the **Modify DB Instance page**, choose Provisioned IOPS for **Storage type** and then provide a Provisioned IOPS value.

If the value you specify for either **Allocated storage** or **Provisioned IOPS** is outside the limits supported by the other parameter, a warning message is displayed. This message gives the range of values required for the other parameter.

6. Choose **Continue**.
7. To apply the changes to the DB instance immediately, choose **Apply immediately** in the **Scheduling of modifications** section. Or choose **Apply during the next scheduled maintenance window** to apply the changes during the next maintenance window.
8. Review the parameters to be changed, and choose **Modify DB instance** to complete the modification.

The new value for allocated storage or for Provisioned IOPS appears in the **Status** column.

Deleting a DB instance

To delete a DB instance, you must do the following:

- Provide the name of the instance
- Enable or disable the option to take a final DB snapshot of the instance
- Enable or disable the option to retain automated backups

If the DB instance that you want to delete has a read replica, you should either promote the read replica or delete it.

Deletion protection

You can only delete instances that don't have deletion protection enabled. When you create or modify a DB instance, you have the option to enable deletion protection so that users can't delete the DB

instance. Deletion protection is disabled by default for you when you use AWS CLI and API commands. Deletion protection is enabled for you when you use the AWS Management Console to create a production DB instance. However, Amazon RDS enforces deletion protection when you use the console, the CLI, or the API to delete a DB instance. To delete a DB instance that has deletion protection enabled, first modify the instance and disable deletion protection. Enabling or disabling deletion protection doesn't cause an outage.

Creating a final snapshot and

retaining automated backups

When you delete a DB instance, you can choose to do one or both of the following:

- Create a final DB snapshot.

    - To be able to restore your deleted DB instance later, create a final DB snapshot. The final snapshot is retained, along with any manual snapshots that were taken.
    - To delete a DB instance quickly, you can skip creating a final DB snapshot.
- Retain automated backups.
    - Your automated backups are retained for the retention period that is set on the DB instance at the time when you delete it. This set retention period occurs whether or not you choose to create a final DB snapshot.
    - If you don't choose to retain automated backups, your automated backups in the same AWS Region as the DB instance are deleted. They can't be recovered after you delete the DB instance.

    - You typically don't need to retain automated backups if you create a final DB snapshot.
    - To delete a retained automated backup.
- Use an earlier manual snapshot of the DB instance to restore the DB instance to that DB snapshot's point in time.
- Retain automated backups. You can use them to restore your DB instance during your retention period, but not after your retention period has ended.

Deleting a DB instance

You can delete a DB instance using the AWS Management Console, the AWS CLI, or the RDS API.

The time required to delete a DB instance can vary depending on the backup retention period (that is,

how many backups to delete), how much data is deleted, and whether a final snapshot is taken.

Console

To delete a DB instance

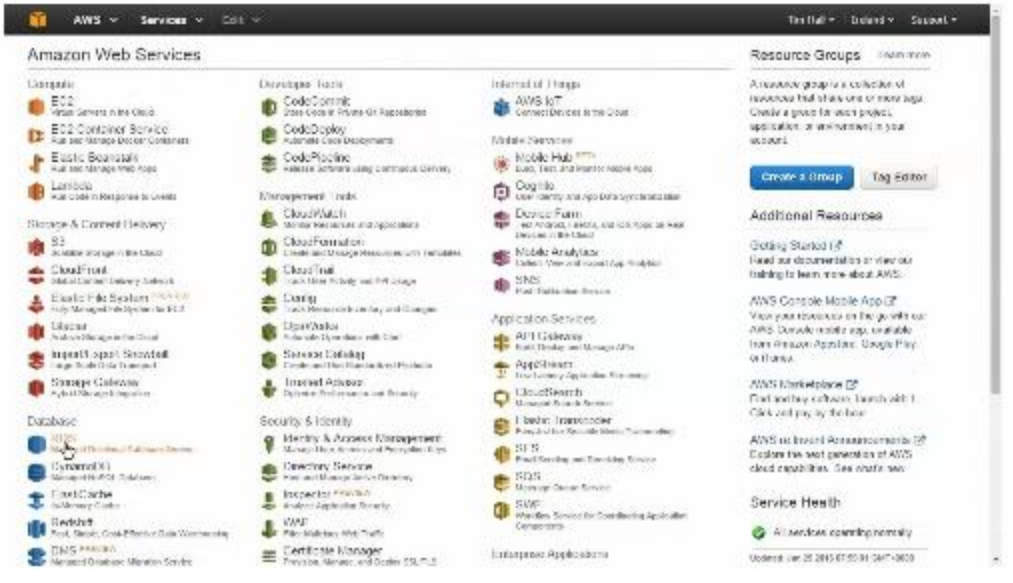
1. Sign in to the AWS Management Console and open the Amazon RDS console.
2. In the navigation pane, choose **Databases**, and then choose the DB instance that you want to delete.
3. For **Actions**, choose **Delete**.
4. To create a final DB snapshot for the DB instance, choose **Create final snapshot?**.
5. If you chose to create a final snapshot, enter the **Final snapshot name**.
6. To retain automated backups, choose **Retain automated backups**.
7. Enter **delete me** in the box.
8. Choose **Delete**.

Relational Database Services (RDS) for Oracle – *Step-by-Step*

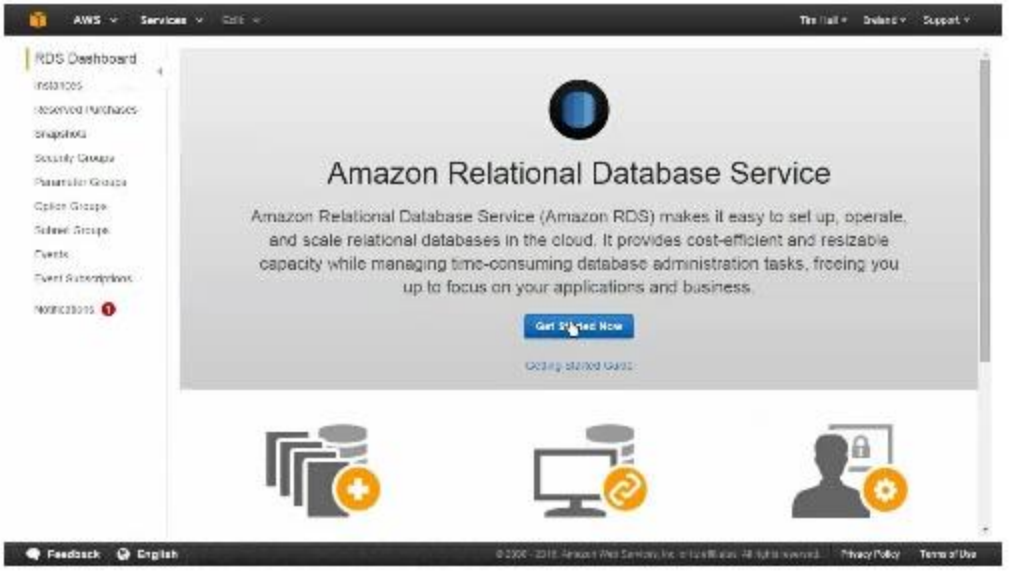
This artifact describes the creation of a database using Amazon Web Services (AWS) Relational Database Services (RDS) for Oracle, a Database as a Service (DBaaS) offering.

Create a New Oracle Service

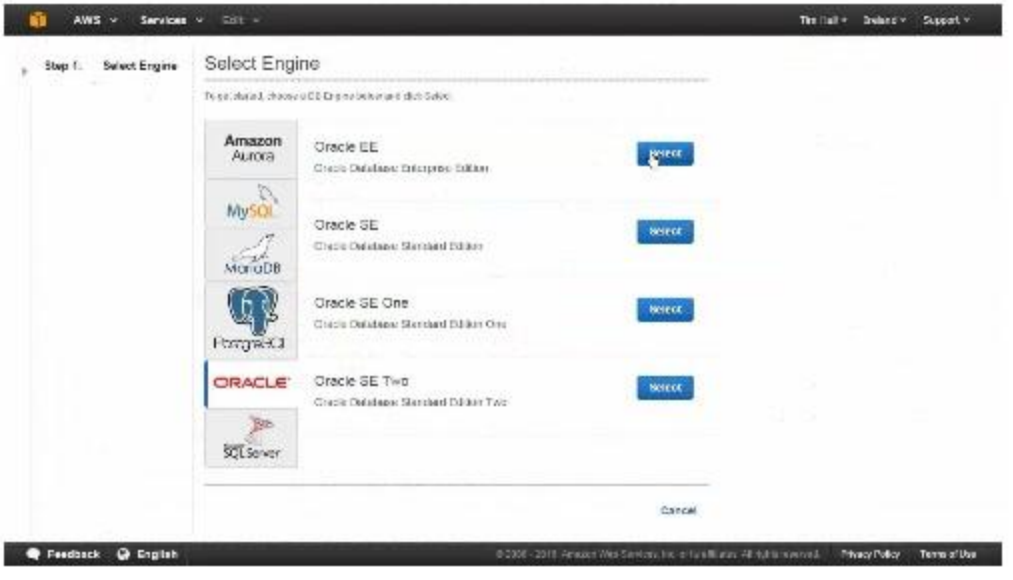
Select the RDS option from the service catalog.



If this is your first RDS service, click the "Get Started Now" button.

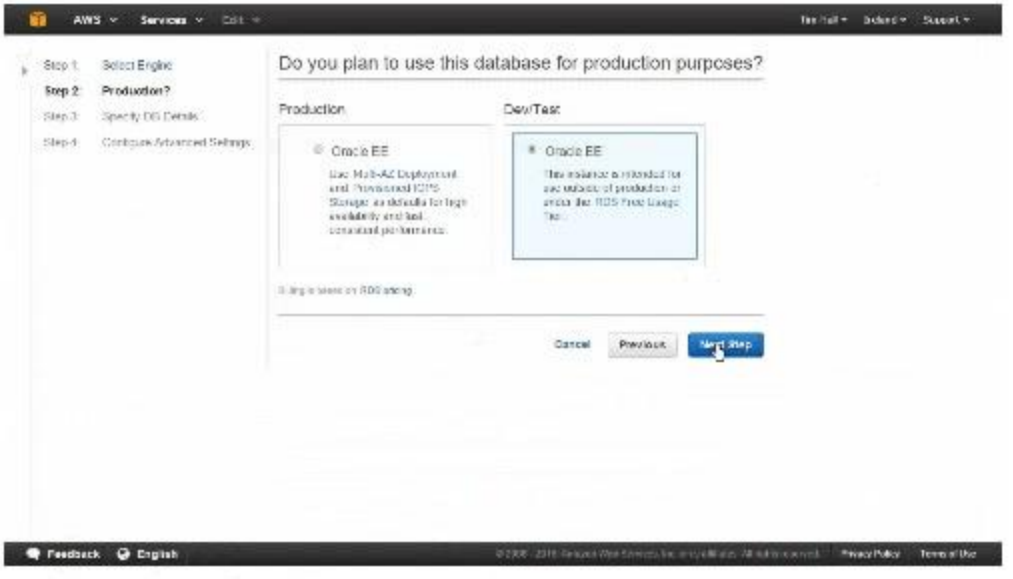


Select the "Oracle EE" option by clicking the "Oracle" tab and clicking the "Select" button next to the "Oracle EE" option.

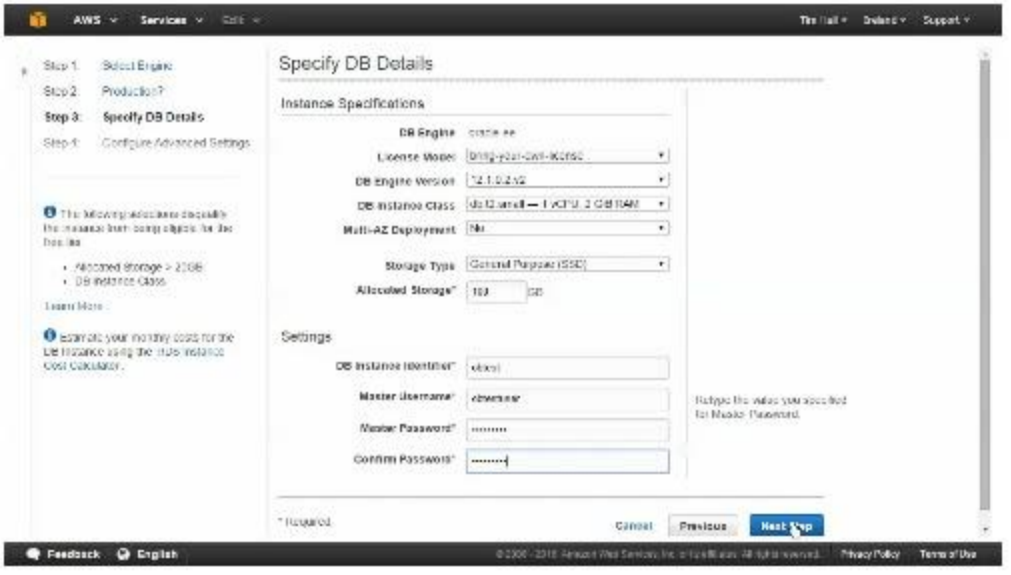


Decide if you want the service to be production or non-production. This will affect the high availabil-

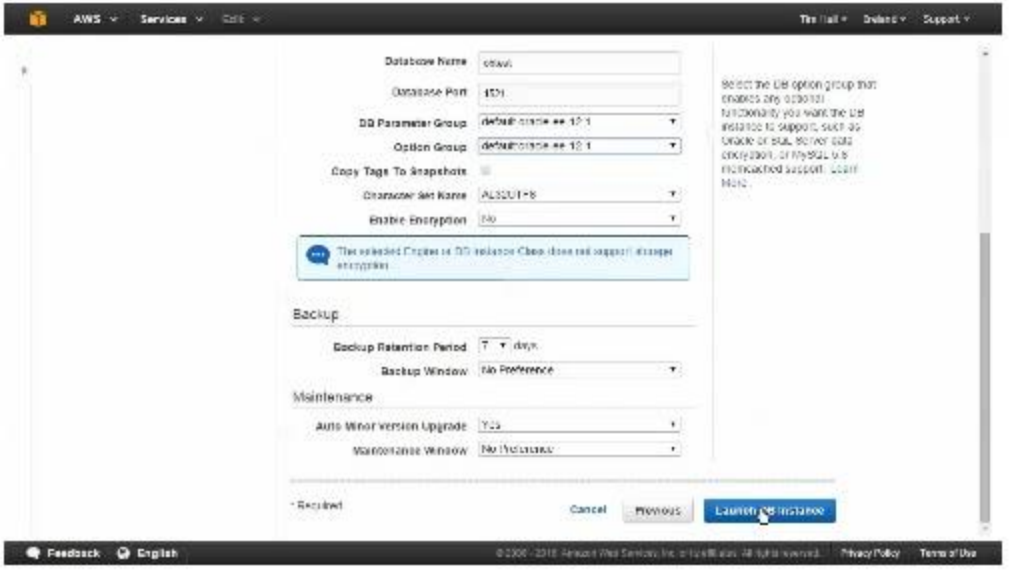
ity and storage performance options available later. In this case I'm using the Dev/Test option. When you've made your choice, click the "Next Step" button.



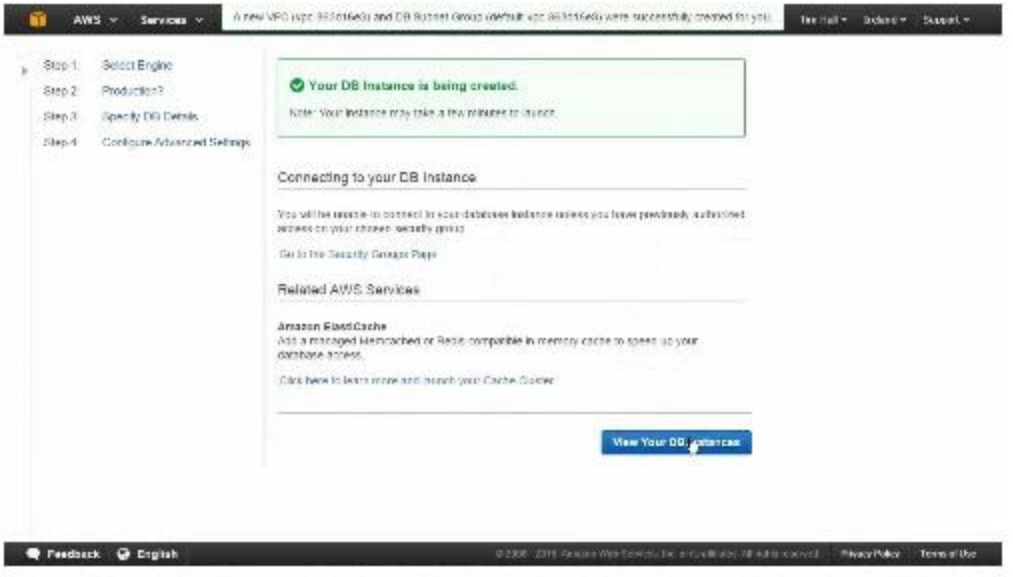
Enter the database details. If you are trialling the service, be careful about the options you pick. If you pick the on-demand license model your service costs will include the Oracle licensing, making the hourly cost significantly more expensive. Once you are happy with the settings click the "Next Step" button.



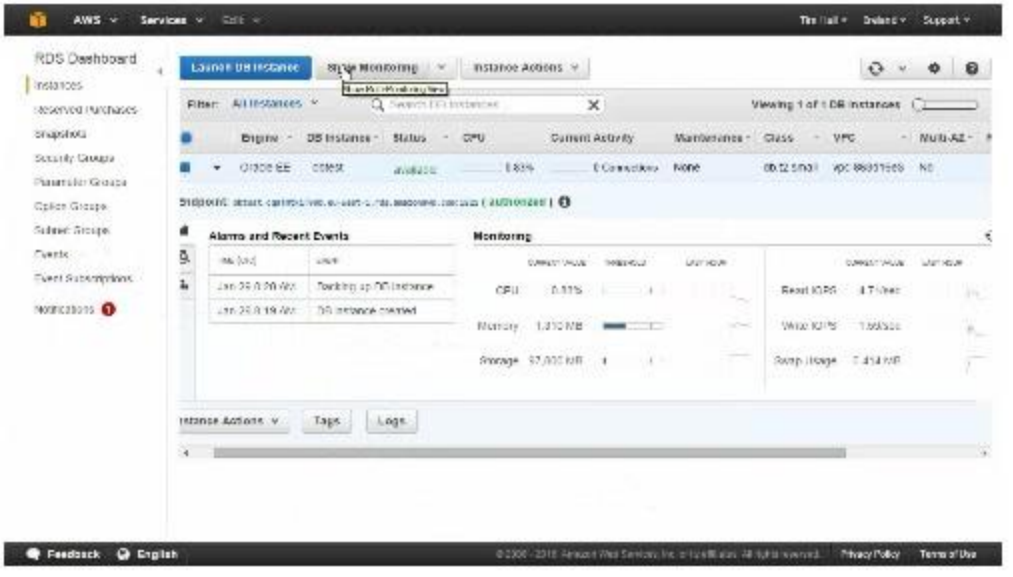
Enter the advanced settings, making sure the character set, backup and maintenance settings are correct for your service. Once you are happy with the settings click the "Launch DB Instance" button.



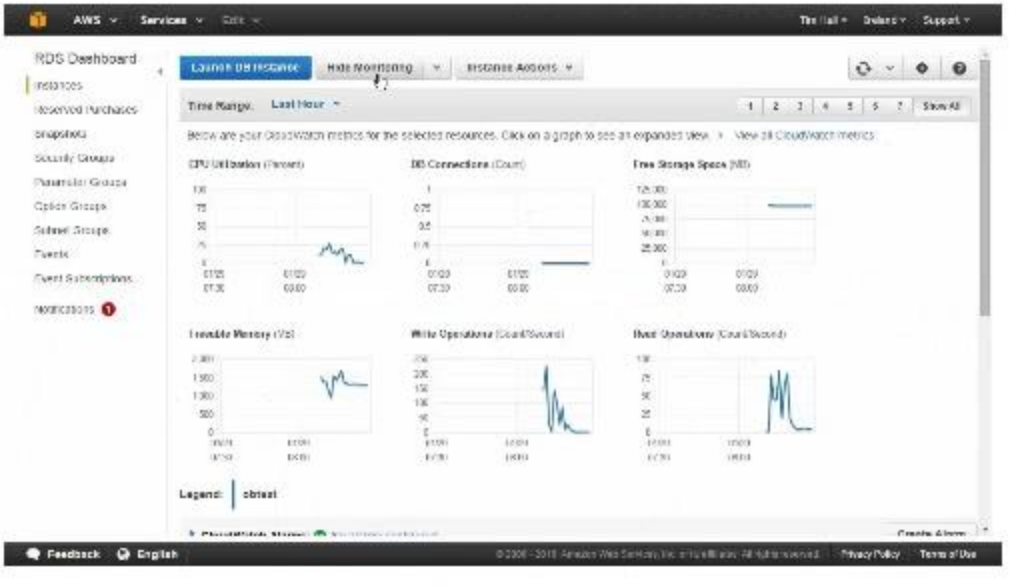
Click the "View Your DB Instances" button to see the status of your service.



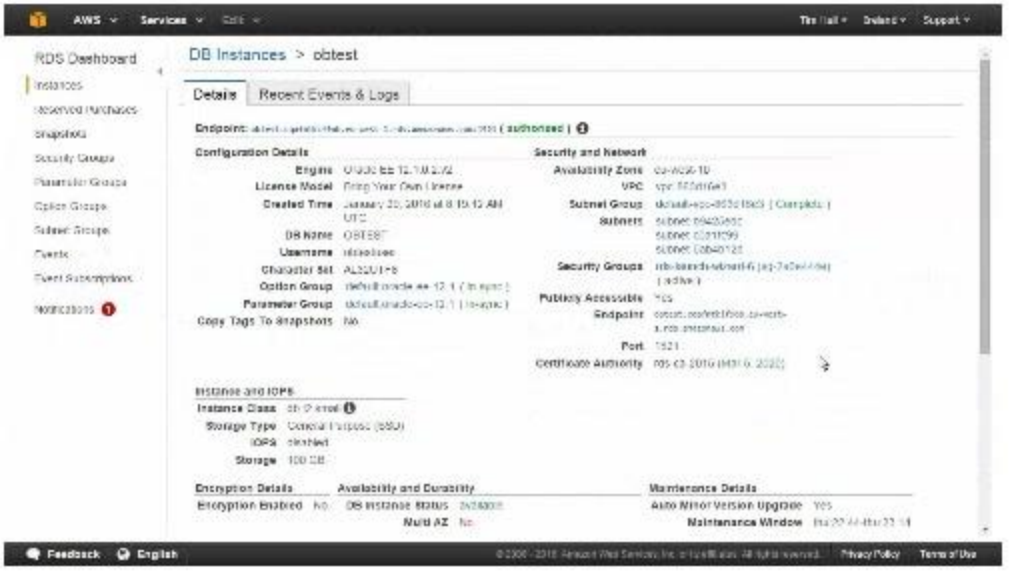
The resulting page shows the status of your services. Once the status is "available", you can connect to the service. If you want to see the monitoring information for the specific service, click the checkbox next to the service and click the "Show Monitoring" button.



The following picture shows the monitoring graphs.



To display the configuration details of the service,s-elect the "Instance Actions > See Details" option.



Connecting to the Service

Once the service is available you can connect to it using a regular TNS connection, using either the EZconnect URL or a "tnsnames.ora" entry.

```
# EZconnect
//obtest.cqafmtkif9ob.us-
west-2.rds.amazonaws.com:1521/obtest

# tnsnames.ora
aws_obtest=
 (DESCRIPTION=
  (ADDRESS=
```

```
    (PROTOCOL=TCP)
    (HOST=obtest.cqafmtkif9ob.us-
west-2.rds.amazonaws.com)
    (PORT=1521)
  )
  (CONNECT_DATA=
    (SERVER=dedicated)
    (SERVICE_NAME=obtest)
  )
 )
```

Restrictions

There are a number of restrictions when using RDS for Oracle, most of which are to protect the service from you doing stupid things.

- You do not get access to the SYS user. You only have access to a DBA user.
- Some of the operations you would typically expect to do as a DBA are restricted. Instead, you have to use the RDSADMIN.RD-

SADMIN_UTIL package, which provides an API to perform these tasks. Its usage is described here.

- With a few exceptions, you do not have direct access to the file system, so if you have processes that sFTP/SCP data to the server to be loaded, you may have to think again.
- The service was limited to Oracle 11gR2 until April 2015, when support for 12cR1 (12.1.0.1) was announced. In July 2015, they also included support for 12.1.0.2.

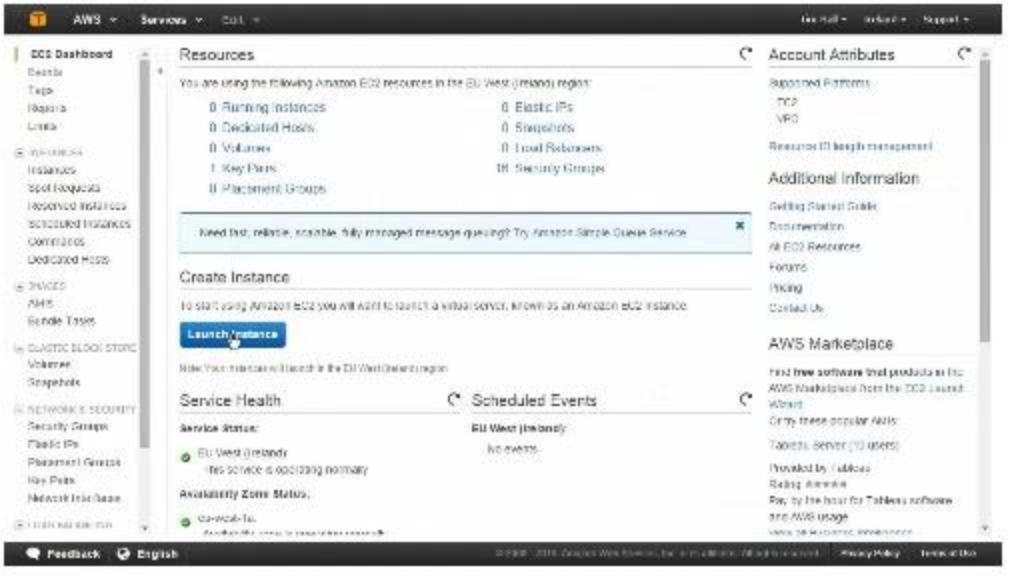
Installation of Oracle on EC2 – *Step-by-Step*

This article gives an overview of creating an Oracle database in an Amazon Web Services (AWS) Elastic Compute Cloud (EC2) virtual machine (VM). This example uses Oracle Linux 7 (OL7) and Oracle Database 12c Release 1 (12.1.0.2), although the process is similar for any other Linux and database version.

Create a Virtual Machine

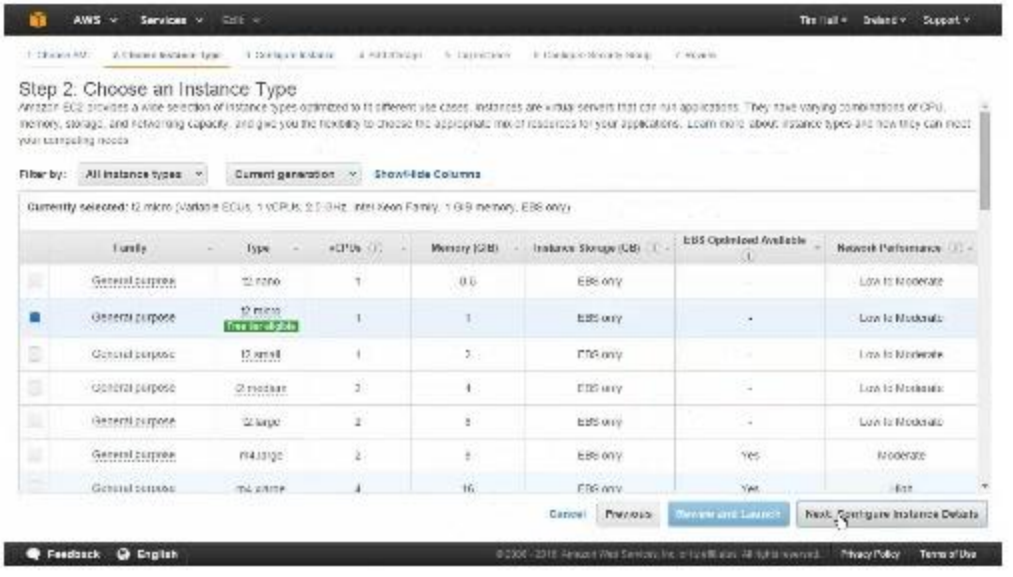
Select the EC2 option from the service catalog.

Click the "Launch Instance" button.

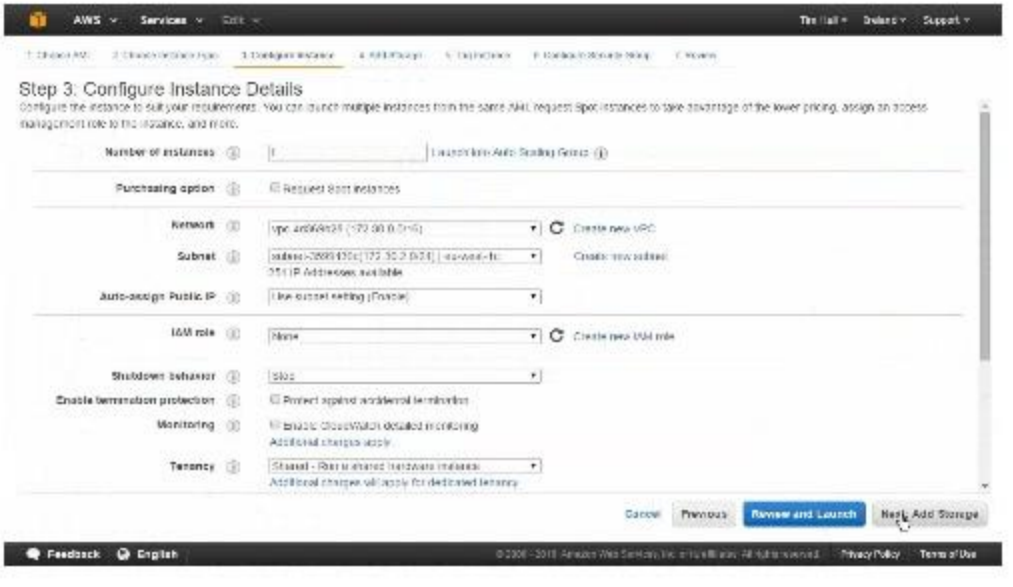


Select the Amazon Machine Image (AMI) of your choice. There are some images available with Oracle already installed, but this article assumes you want to install the database yourself. Click the "Select" button for the required image. In this case we clicked on the "Community AMIs" tab and searched for "OL7.2", then used the "Oracle Linux 7.2 (HVM)" AMI.

Select the required instance type, then click the "Next: Configure Instance Details" button.

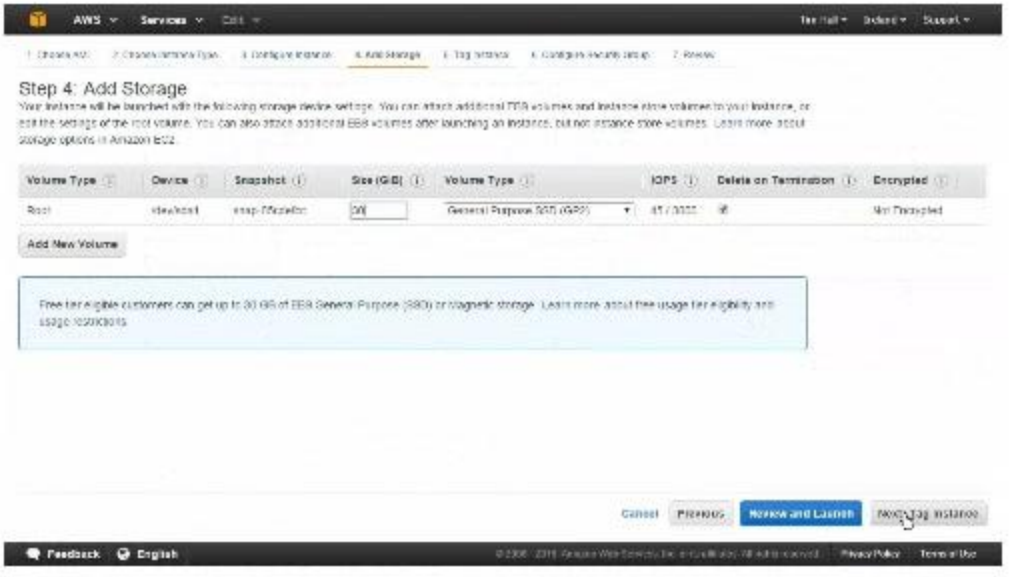


Make any required changes to the instance configuration, then click the "Next: Add Storage" button.

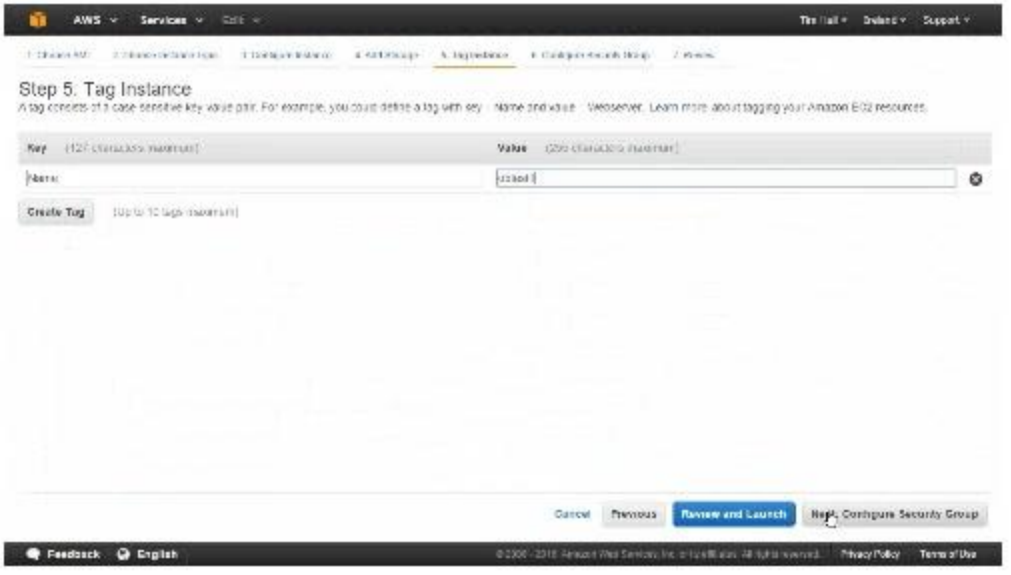


Edit the size and volume type to the required settings (30GiB and General Purpose (SSD)). If you

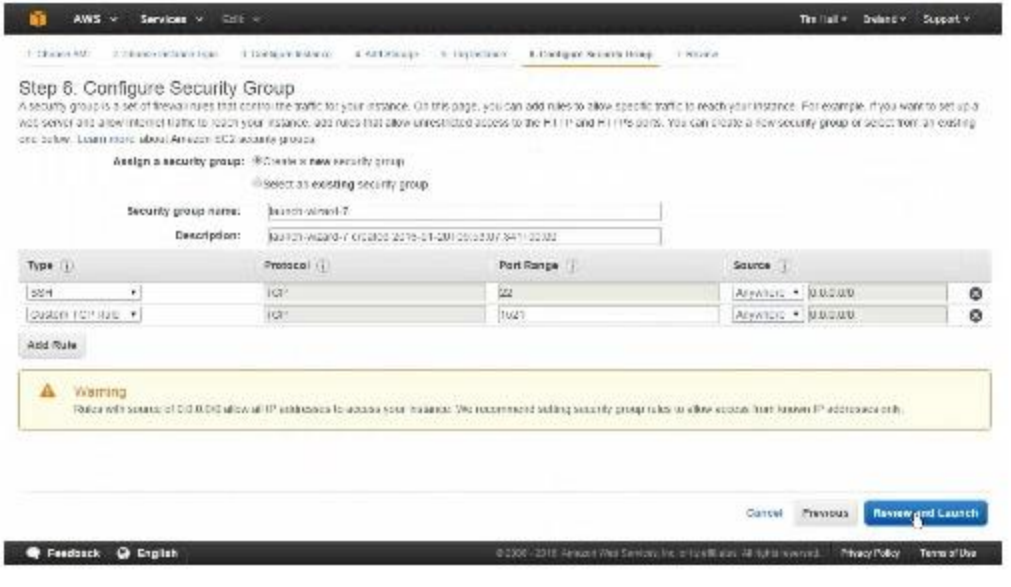
require any other volumes add them now. Click the "Next: Tag Instance" button.



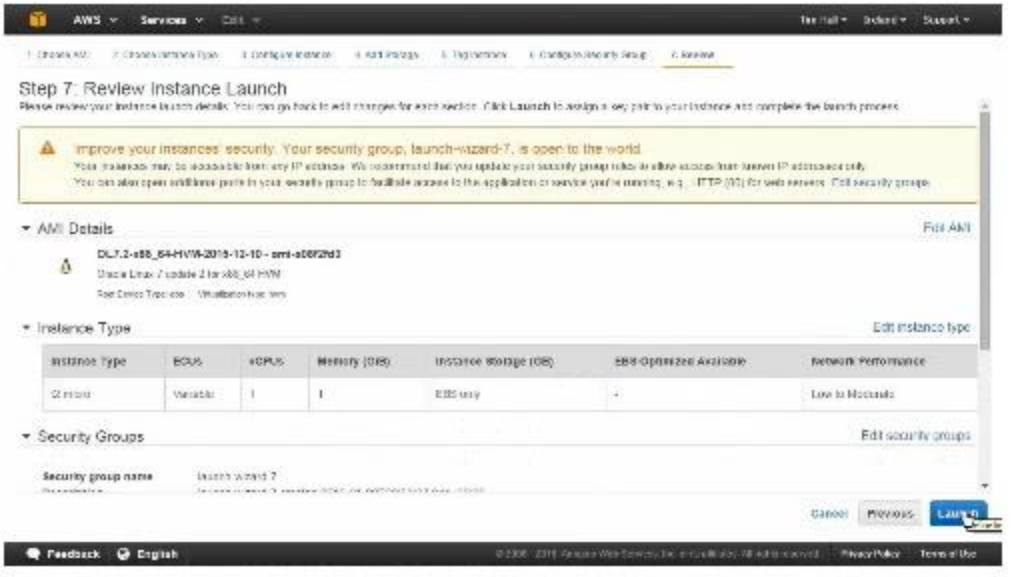
Tags allow you to associate name-value pairs with an instance. Add any required tags. In this case, I've just defined a name tag. When you are ready, click the "Next: Configure Security Group" button.



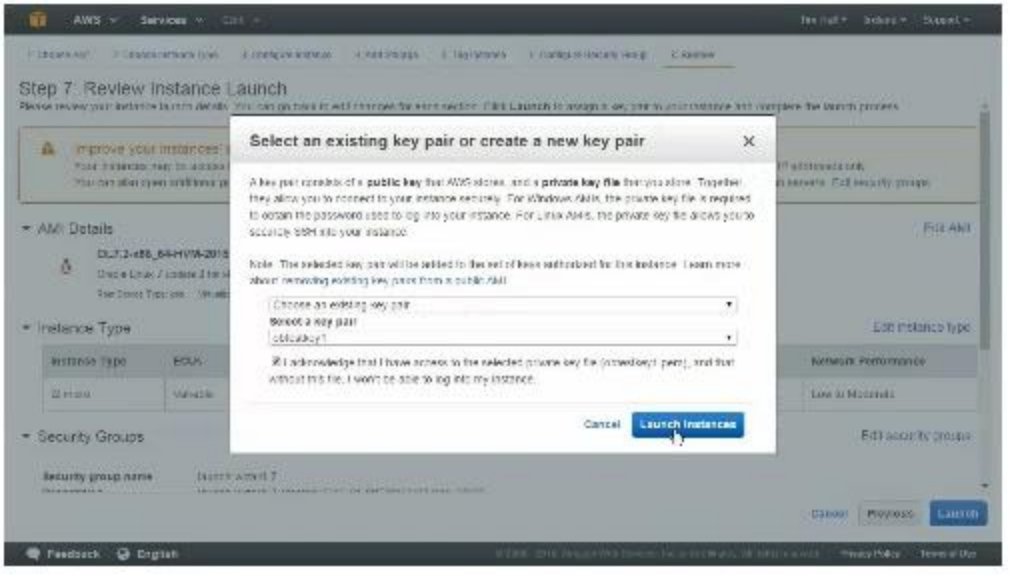
Add a rule for TCP with the port 1521 and click the "Review and Launch" button. It's better to lock this down to specific IP addresses, but in this case I'm making it accessible from anywhere.



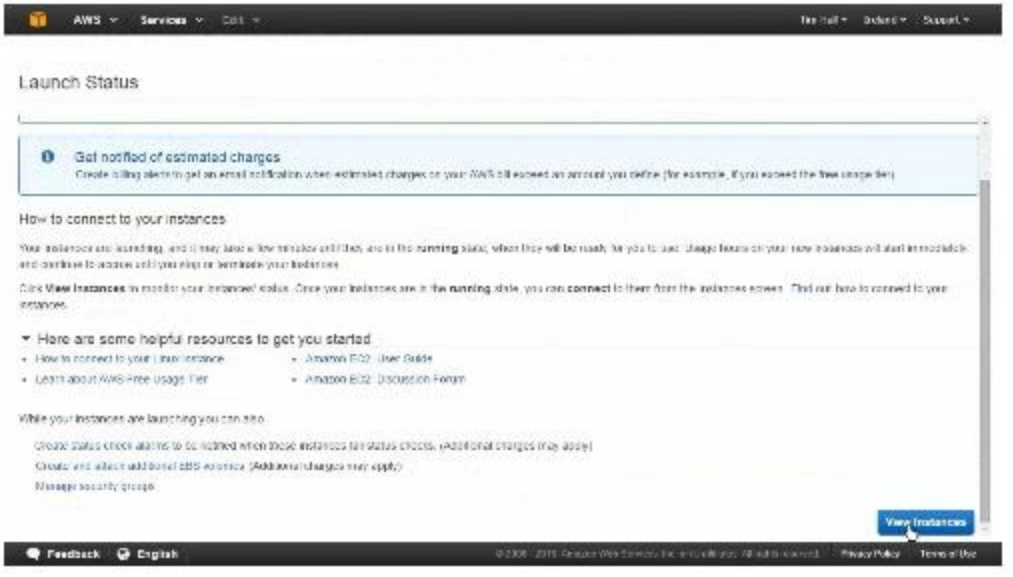
On the "Review Instance Launch" page, click the "Launch" button.



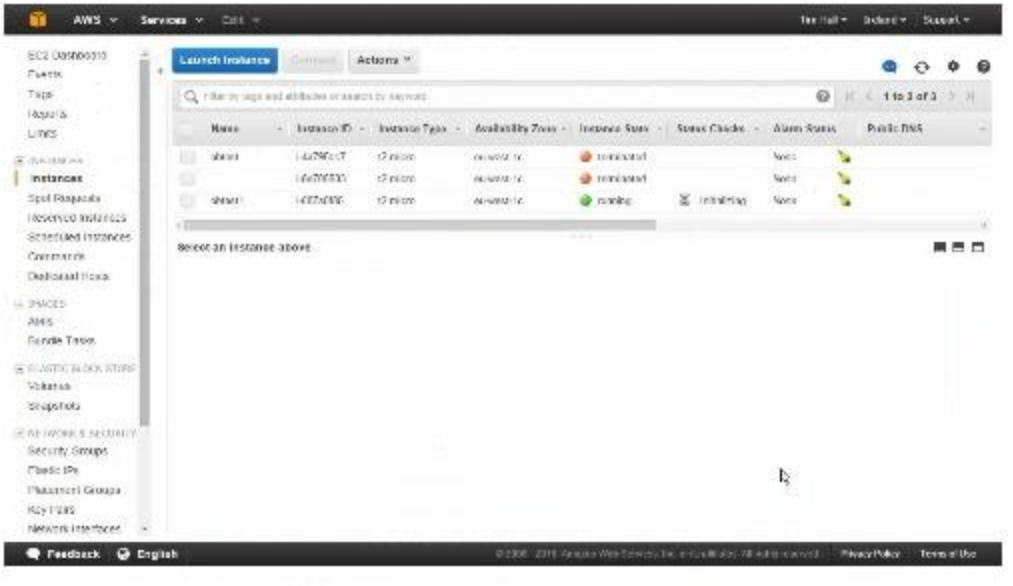
You can either download a new key pair, or use an existing one. Once you've selected the option you want, click the "Launch Instances" button.



Click the "View Instances" button.

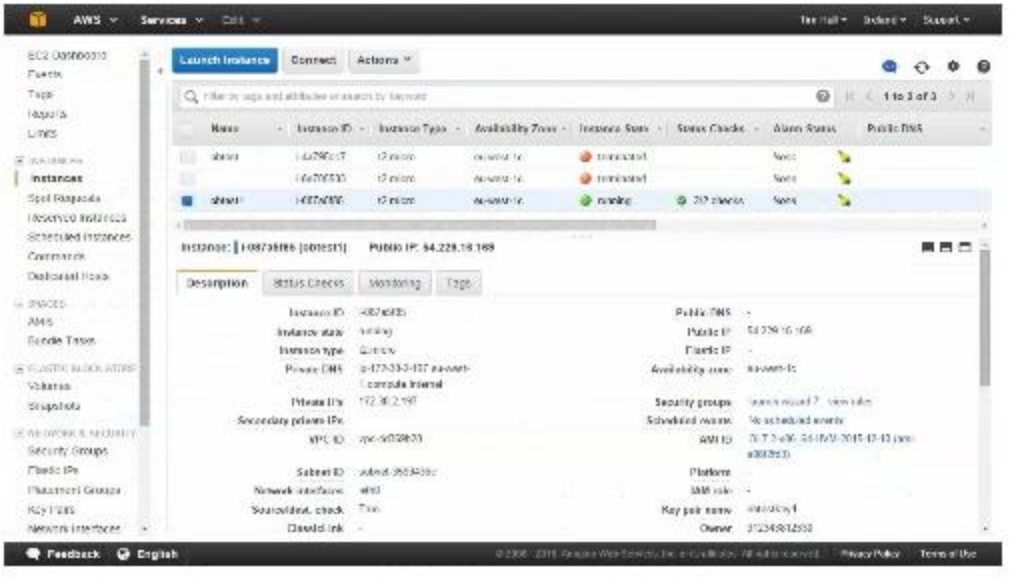


Wait while the instance initializes.



Once initialized, you can connect to virtual machine. It may take some time for the Public DNS

to be propagated, but you can use the Public IP immediately.



Prepare the OS

Connect to the virtual machine using the following type of commands. Remember to adjust to the name of your VM and the user you created.

```
$ cd Dropbox/cloud/aws-keys/
$ chmod 600  obtestkey1.pem
$ ssh -i obtestkey1.pem ec2-
user@54.229.16.169
[ec2-user@ip-172-30-2-197 ~]$
```

Either use "sudo" before all your admin commands, or switch to the "root" user.

```
[ec2-user@ip-172-30-2-197 ~]$ sudo su -
[root@ip-172-30-2-197 ~]#
```

Follow the setup instructions for the Oracle installation you are attempting. In this case, I used the automatic setup from the following.

- Oracle Database 12c Release 1 (12.1) Installation On Oracle Linux 7 (OL7)

You will probably also need the following.

```
yum install wget zip unzip -y
yum update -y
```

Unzip the installation software.

```
mkdir -p /u01/software
cd /u01/software

# Copy the Oracle software to the directory.
```

```
unzip linuxamd64_12102_database_1of2.zip
unzip linuxamd64_12102_database_2of2.zip
cd database
chown -R oracle.oinstall /u01
```

Oracle Software Installation

Silent installations are the simplest solution for most cloud-based VMs. The following command shows a silent software-only installation using a response file (oui12102.rsp).

```
# su - oracle
$ cd /u01/software/database
$ ./runInstaller -silent -ignoreSysPrereqs -responseFile /tmp/oui12102.rsp
Starting Oracle Universal Installer...

Checking Temp space: must be greater than 500 MB.   Actual 24071 MB    Passed
Checking swap space: 0 MB available, 150 MB required.    Failed <<<<
```

>>> Ignoring required pre-requisite failures. Continuing...

Preparing to launch Oracle Universal Installer from /tmp/OraInstall2015-04-06_11-35-07AM. Please wait ...[oracle@ip-172-30-2-197 database] $ [WARNING] [INS-13014] Target environment does not meet some optional requirements.
CAUSE: Some of the optional prerequisites are not met. See logs for details. /tmp/OraInstal-l2015-04-06_11-35-07AM/installActions2015-04-06_11-35-07AM.log
ACTION: Identify the list of failed prerequisite checks from the log: /tmp/OraInstall2015-04-06_11-35-07AM/installActions2015-04-06_11-35-07AM.log. Then either from the log file or from installation manual find the appropriate configuration to meet the prerequisites and fix it manually.
You can find the log of this install session at:

```
/u01/app/oraInventory/logs/installAction-
s2015-04-06_11-35-07AM.log
The installation of Oracle Database 12c was
successful.
Please check '/u01/app/oraInventory/logs/
silentInstall2015-04-06_11-35-07AM.log' for
more details.

As a root user, execute the following script(s):
        1. /u01/app/oraInventory/orainstRoot.sh
        2. /u01/app/oracle/product/12.1.0.2/
db_1/root.sh


Successfully Setup Software.

$
```

Remember to run the "orainstRoot.sh" and "root.sh" scripts as directed.

Database Creation (DBCA)

Before you start the database creation, start the listener using the following command.

```
[oracle@ip-172-30-0-173 database]$ lsnrctl start

LSNRCTL for Linux: Version 12.1.0.2.0 - Production on 06-APR-2015 11:45:11

Copyright (c) 1991, 2014, Oracle.  All rights reserved.

Starting /u01/app/oracle/product/12.1.0.2/db_1/bin/tnslsnr: please wait...

TNSLSNR for Linux: Version 12.1.0.2.0 - Production
Log messages written to /u01/app/oracle/diag/tnslsnr/ip-172-30-2-197/listener/alert/log.xml
Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=ip-172-30-2-197.eu-west-1.compute.internal)(PORT=1521)))
```

```
Connecting to (ADDRESS=(PROTOCOL=tcp)
(HOST=)(PORT=1521))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version
12.1.0.2.0 - Production
Start Date                06-APR-2015 11:45:11
Uptime                    0 days 0 hr. 0 min. 0 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Log File         /u01/app/oracle/diag/
tnslsnr/ip-172-30-2-197/listener/alert/log.xml
Listening Endpoints Summary...
 (DESCRIPTION=(ADDRESS=(PROTO-
COL=tcp)(HOST=ip-172-30-2-197.eu-
west-1.compute.internal)(PORT=1521)))
The listener supports no services
The command completed successfully
```

```
[oracle@ip-172-30-2-197 database]$
```

As with the software installation, the database creation should be run in silent mode. The following command creates a container database with a single PDB.

```
[oracle@ip-172-30-2-197 database]$ dbca -silent
-createDatabase \
 -templateName General_Purpose.dbc \
 -gdbname cdb1 -sid cdb1 -responseFile NO_
VALUE \
 -characterSet AL32UTF8 \
 -sysPassword OraPasswd1 \
 -systemPassword OraPasswd1 \
 -createAsContainerDatabase true \
 -numberOfPDBs 1 \
 -pdbName pdb1 \
 -pdbAdminPassword OraPasswd1 \
 -databaseType MULTIPURPOSE \
 -automaticMemoryManagement false \
```

```
 -storageType FS \
 -ignorePreReqs
Copying database files
1% complete
2% complete
8% complete
13% complete
19% complete
27% complete
Creating and starting Oracle instance
29% complete
32% complete
33% complete
34% complete
38% complete
42% complete
43% complete
45% complete
Completing Database Creation
48% complete
```

```
51% complete
53% complete
62% complete
70% complete
72% complete
Creating Pluggable Databases
78% complete
100% complete
Look at the log file "/u01/app/oracle/cfgtoollogs/
dbca/cdb1/cdb10.log" for further details.
[oracle@ip-172-30-2-197 database]$
```

You can now connect to the database from the local server.

```
[oracle@ip-172-30-2-197 database]$ sqlplus / as
sysdba

SQL*Plus: Release 12.1.0.2.0 Production on Mon
Apr 6 05:17:57 2015
```

```
Copyright (c) 1982, 2014, Oracle.  All rights
reserved.


Connected to:
Oracle Database 12c Enterprise Edition Release
12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics
and Real Application Testing options

SQL> SELECT name, open_mode FROM v$pdbs;

NAME                 OPEN_MODE
---------- ----------
PDB$SEED             READ ONLY
PDB1                 READ WRITE

SQL>
```

Remote Connections

With the public endpoint in place, remote connections are now possible. Create the following "tnsnames.ora" entry on a remote PC.

```
aws_pdb1=
 (DESCRIPTION=
  (ADDRESS=
   (PROTOCOL=TCP)
   (HOST=54.229.16.169)
   (PORT=1521)
  )
  (CONNECT_DATA=
   (SERVER=dedicated)
   (SERVICE_NAME=pdb1)
  )
 )
```

Connect to the cloud database.

```
C:\>sqlplus sys@aws_pdb1 as sysdba
```

SQL*Plus: Release 11.2.0.3.0 Production on Mon Apr 6 10:46:33 2015

Enter password:

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options

sys@cdb1>

Oracle Application Express (APEX)

Amazon RDS supports Oracle Application Express (APEX) through the use of the APEX and APEX-DEV options. Oracle APEX can be deployed as a run-time environment or as a full development environment for web-based applications. Using Oracle APEX, developers can build applications entirely within the web browser.

Oracle APEX consists of the following main components:

- A *repository* that stores the metadata for APEX applications and components. The repository consists of tables, indexes, and other objects that are installed in your Amazon RDS DB instance.
- A *listener* that manages HTTP communications with Oracle APEX clients. The listener accepts incoming connections from web

browsers, forwards them to the Amazon RDS DB instance for processing, and then sends results from the repository back to the browsers. Amazon RDS for Oracle supports the following types of listeners:

- For APEX version 5.0 and later, use Oracle Rest Data Services (ORDS) version 19.1 and higher. We recommend that you use the latest supported version of Oracle APEX and ORDS. The documentation describes older versions for backwards compatibility only.
- For APEX version 4.1.1, you can use Oracle APEX Listener version 1.1.4.
- Oracle HTTP Server and mod_plsql.

When you add the Amazon RDS APEX options to your DB instance, Amazon RDS installs the Oracle APEX repository only. Install your listener on a separate host, such as an Amazon EC2 instance, an on-

premises server at your company, or your desktop computer.

The APEX option uses storage on the DB instance class for your DB instance. Following are the supported versions and approximate storage requirements for Oracle APEX.

APEX version	Storage requirements	Supported Oracle database versions	Notes
Oracle APEX version 20.2.v1	148 MiB	All	This version includes patch p32006852_2020_Generic. You can see the patch number and date by running the following query: `SELECT PATCH_VERSION, PATCH_NUMBER FROM APEX_PATCHES;`
Oracle APEX version 20.1.v1	173 MiB	All	This version includes patch 30990551.
Oracle APEX version 19.2.v1	149 MiB	All	
Oracle APEX version 19.1.v1	148 MiB	All	
Oracle APEX version 18.2.v1	146 MiB	All except 19c	
Oracle APEX version 18.1.v1	145 MiB	All except 19c	
Oracle APEX version 5.1.4.v1	220 MiB	All except 19c	
Oracle APEX version 5.1.2.v1	150 MiB	12.1 only	
Oracle APEX version 5.0.4.v1	140 MiB	12.1 only	

Oracle APEX version 4.2.6.v1	160 MiB	12.1 only	

Prerequisites for Oracle APEX and ORDS

To use Oracle APEX and ORDS, make sure you have the following:

- The Java Runtime Environment (JRE)
- An Oracle client installation that includes the following:
  - SQL*Plus or SQL Developer for administration tasks
  - Oracle Net Services for configuring connections to your Oracle instance

Adding the Amazon RDS APEX options

The general process for adding the Amazon RDS APEX options to a DB instance is the following:

1. Create a new option group, or copy or modify an existing option group.
2. Add the options to the option group.
3. Associate the option group with the DB instance.

When you add the Amazon RDS APEX options, a brief outage occurs while your DB instance is automatically restarted.

To add the APEX options to a DB instance

1. Determine the option group you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group with the following settings:
    a. For **Engine**, choose the Oracle edition that you want to use. The APEX options are supported on all editions.

b. For **Major engine version**, choose the version of your DB instance.

2. Add the options to the option group.
   If you want to deploy only the Oracle APEX run-time environment, add only the APEX option. If you want to deploy the full development environment, add both the APEX and APEX-DEV options. For Oracle 12c, add the **APEX** and **APEX-DEV** options.

For **Version**, choose the version of APEX that you want to use. If you don't choose a version, version 4.2.6.v1 is the default for 12c.

3. Apply the option group to a new or existing DB instance:
   - For a new DB instance, you apply the option group when you launch the instance.
   - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option

group. When you add the APEX options to an existing DB instance, a brief outage occurs while your DB instance is automatically restarted.

Unlocking the public user account

After the Amazon RDS APEX options are installed, you must change the password for the APEX public user account, and then unlock the account. You can do this by using the Oracle SQL*Plus command line utility. Connect to your DB instance as the master user, and issue the following commands. Replace new_password with a password of your choice.

```
alter user APEX_PUBLIC_USER identified by
new_password;
alter user APEX_PUBLIC_USER account unlock;
```

Configuring RESTful services for Oracle APEX

To configure RESTful services in APEX (not needed for APEX 4.1.1.V1), use SQL*Plus to connect to your DB instance as the master user. After you do this, run the `rdsadmin.rdsadmin_run_apex_rest_config` stored procedure. When you run the stored procedure, you provide passwords for the following users:

- `APEX_LISTENER`
- `APEX_REST_PUBLIC_USER`

The stored procedure runs the `apex_rest_config.sql` script, which creates new database accounts for these users.

The following command runs the stored procedure.

```
exec rdsadmin.rdsadmin_run_apex_rest_config('apex_listener_password', 'apex_rest_public_user_password');
```

Setting up ORDS for Oracle APEX

You are now ready to install and configure Oracle Rest Data Services (ORDS) for use with Oracle APEX. For APEX version 5.0 and later, use Oracle Rest Data Services (ORDS) version 19.1 and higher.

Install the listener on a separate host such as an Amazon EC2 instance, an on-premises server at your company, or your desktop computer. For the examples in this section, we assume that the name of your host is myapexhost.example.com, and that your host is running Linux.

Preparing to install ORDS

Before you can install ORDS, you need to create a nonprivileged OS user, and then download and unzip the APEX installation file.

To prepare for ORDS installation

1. Log in to myapexhost.example.com as root.
2. Create a nonprivileged OS user to own the listener installation. The following command creates a new user named *apexuser*.

```
useradd -d /home/apexuser apexuser
```

The following command assigns a password to the new user.

```
passwd apexuser;
```

3. Log in to myapexhost.example.com as apexuser, and download the APEX installation file from Oracle to your /home/apexuser directory
4. Unzip the file in the /home/apexuser directory.

```
unzip apex_<version>.zip
```

After you unzip the file, there is an apex directory in the /home/apexuser directory.

5. While you are still logged into myapex-host.example.com as apexuser, download the Oracle REST Data Services file from Oracle to your /home/apexuser directory.

Installing and configuring ORDS

Before you can use APEX, you need to download the ords.war file, use Java to install ORDS, and then start the listener.

To install and configure ORDS for use with Oracle APEX

1. Create a new directory based on ORDS, and then unzip the listener file.

```
mkdir /home/apexuser/ORDS
cd /home/apexuser/ORDS
```

2. Download the file ords.*version.number*.zip from Oracle REST data services.
3. Unzip the file into the /home/apexuser/ORDS directory.
4. Grant the master user the required privileges to install ORDS.

After the Amazon RDS APEX option is installed, give the master user the required privileges to install the ORDS schema. You can do this by connecting to the database and running the following commands. Replace *master_user* with the name of your master user.

```
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_OBJECTS', 'master_user', 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_ROLE_PRIVS', 'master_user', 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_TAB_COLUMNS', 'master_user', 'SELECT', true);
```

```
exec rdsadmin.rdsadmin_util.grant_sys_ob-
ject('USER_CONS_COLUMNS', 'master_user', 'SE-
LECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_ob-
ject('USER_CONSTRAINTS', 'master_user', 'SELECT',
true);
exec rdsadmin.rdsadmin_util.grant_sys_ob-
ject('USER_OBJECTS', 'master_user', 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_ob-
ject('USER_PROCEDURES', 'master_user', 'SELECT',
true);
exec rdsadmin.rdsadmin_util.grant_sys_ob-
ject('USER_TAB_COLUMNS', 'master_user', 'SELECT',
true);
exec rdsadmin.rdsadmin_util.grant_sys_ob-
ject('USER_TABLES', 'master_user', 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_ob-
ject('USER_VIEWS', 'master_user', 'SELECT', true);
exec rdsadmin.rdsadmin_util.grant_sys_ob-
ject('WPIUTL', 'master_user', 'EXECUTE', true);
```

```
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_SESSION', 'master_user', 'EXECUTE', true);
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_UTILITY', 'master_user', 'EXECUTE', true);
```

5. Install the ORDS schema using the downloaded ords.war file.

```
java -jar ords.war install advanced
```

The program prompts you for the following information. The default values are in brackets.

- Enter the location to store configuration data:

Enter /home/apexuser/ORDS. This is the location of the ORDS configuration files.

- Specify the database connection type to use. Enter number for [1] Basic [2] TNS [3] Custom URL [1]:

Choose the desired connection type.

- Enter the name of the database server [localhost]: *DB_instance_endpoint*

Choose the default or enter the correct value.

- Enter the database listener port [1521]: *DB_instance_port*

Choose the default or enter the correct value.

- Enter 1 to specify the database service name, or 2 to specify the database SID [1]:

Choose 2 to specify the database SID.

- Database SID [xe]

Choose the default or enter the correct value.

- Enter 1 if you want to verify/install Oracle REST Data Services schema or 2 to skip this step [1]:

Choose 1. This step creates the Oracle REST Data Services proxy user named ORDS_PUBLIC_USER.

- Enter the database password for ORDS_PUBLIC_USER:

Enter the password, and then confirm it.

- Requires to login with administrator privileges to verify Oracle REST Data Services schema.

Enter the administrator user name: *master_user*

Enter the database password for *master_user*: *master_user_password*

Confirm the password: *master_user_password*

- Enter the default tablespace for ORDS_METADATA [SYSAUX].

Enter the temporary tablespace for ORDS_METADATA [TEMP].

Enter the default tablespace for ORDS_PUBLIC_USER [USERS].

Enter the temporary tablespace for ORDS_PUBLIC_USER [TEMP].

- Enter 1 if you want to use PL/SQL Gateway or 2 to skip this step. If you're using Oracle Application Express or migrating from mod_plsql, you must enter 1 [1].

Choose the default.

- Enter the PL/SQL Gateway database user name [APEX_PUBLIC_USER]

Choose the default.

- Enter the database password for APEX_PUBLIC_USER:

Enter the password, and then confirm it.

- Enter 1 to specify passwords for Application Express RESTful Services database users (APEX_LISTENER, APEX_REST_PUBLIC_USER) or 2 to skip this step [1]:

Choose 2 for APEX 4.1.1.V1; choose 1 for all other APEX versions.

- [Not needed for APEX 4.1.1.v1] Database password for APEX_LISTENER

Enter the password (if required), and then confirm it.

- [Not needed for APEX 4.1.1.v1] Database password for APEX_REST_PUBLIC_USER

Enter the password (if required), and then confirm it.

- Enter a number to select a feature to enable:

Enter `1` to enable all features: SQL Developer Web, REST Enabled SQL, and Database API.

- Enter 1 if you wish to start in standalone mode or 2 to exit [1]:

Enter `1`.

- Enter the APEX static resources location:

If you unzipped APEX installation files into `/home/apexuser`, enter `/home/apexuser/apex/images`. Otherwise, enter *unzip_path*`/apex/images`, where *unzip_path* is the directory where you unzipped the file.

- Enter 1 if using HTTP or 2 if using HTTPS [1]:

If you enter `1`, specify the HTTP port. If you enter `2`, specify the HTTPS port and the SSL host name. The HTTPS option prompts you to specify how you will provide the certificate:

- Enter 1 to use the self-signed certificate.
- Enter 2 to provide your own certificate. If you enter 2, specify the path for the SSL certificate and the path for the SSL certificate private key.

6. Set a password for the APEX admin user. To do this, use SQL*Plus to connect to your DB instance as the master user, and then run the following commands.

```
EXEC rdsadmin.rdsadmin_util.grant_apex_admin_role;
grant APEX_ADMINISTRATOR_ROLE to master;
@/home/apexuser/apex/apxchpwd.sql
```

Replace *master* with your master user name. When the apxchpwd.sql script prompts you, enter a new admin password.

7. Start the ORDS listener. Run the following code.

```
java -jar ords.war
```

The first time you start ORDS, you are prompted to provide the location of the APEX Static resources. This images folder is located in the /apex/images directory in the installation directory for APEX.

8. Return to the APEX administration window in your browser and choose **Administration**. Next, choose **Application Express Internal Administration**. When you are prompted for credentials, enter the following information:
    - **User name** – admin
    - **Password** – the password you set using the apxchpwd.sql script

Choose **Login**, and then set a new password for the admin user.

Your listener is now ready for use.

Setting up Oracle APEX listener

Amazon RDS for Oracle continues to support APEX version 4.1.1 and Oracle APEX Listener version 1.1.4. We recommend that you use the latest supported versions of Oracle APEX and ORDS.

Install Oracle APEX Listener on a separate host such as an Amazon EC2 instance, an on-premises server at your company, or your desktop computer. We assume that the name of your host is myapexhost.example.com, and that your host is running Linux.

Preparing to install Oracle APEX listener

Before you can install Oracle APEX Listener, you need to create a nonprivileged OS user, and then download and unzip the APEX installation file.

To prepare for Oracle APEX listener installation

1. Log in to myapexhost.example.com as root.

2. Create a nonprivileged OS user to own the listener installation. The following command creates a new user named *apexuser*.

```
useradd -d /home/apexuser apexuser
```

The following command assigns a password to the new user.

```
passwd apexuser;
```

3. Log in to myapexhost.example.com as apexuser, and download the APEX installation file from Oracle to your /home/apexuser directory
4. Unzip the file in the /home/apexuser directory.

```
unzip apex_<version>.zip
```

After you unzip the file, there is an apex directory in the /home/apexuser directory.

5. While you are still logged into myapexhost.example.com as apexuser, download the Oracle APEX Listener file from Oracle to your /home/apexuser directory.

Installing and configuring Oracle APEX listener

Before you can use APEX, you need to download the apex.war file, use Java to install Oracle APEX Listener, and then start the listener.

To install and configure Oracle APEX listener

1. Create a new directory based on Oracle APEX Listener and open the listener file.

Run the following code:

```
mkdir /home/apexuser/apexlistener
cd /home/apexuser/apexlistener
unzip ../apex_listener.version.zip
```

2. Run the following code.

```
java -Dapex.home=./apex -Dapex.images=/home/
apexuser/apex/images -Dapex.erase -jar ./apex.war
```

3. Enter information for the program prompts following:
    - The APEX Listener Administrator user name. The default is *adminlistener*.

- A password for the APEX Listener Administrator.
- The APEX Listener Manager user name. The default is *managerlistener*.
- A password for the APEX Listener Administrator.

The program prints a URL that you need to complete the configuration, as follows.

```
INFO: Please complete configuration at: http://
localhost:8080/apex/listenerConfigure
Database is not yet configured
```

4. Leave Oracle APEX Listener running so that you can use Oracle Application Express. When you have finished this configuration procedure, you can run the listener in the background.
5. From your web browser, go to the URL provided by the APEX Listener program. The Oracle Application Express Listener adminis-

tration window appears. Enter the following information:

- **Username** – APEX_PUBLIC_USER
- **Password** – the password for *APEX_PUBLIC_USER*. This password is the one that you specified earlier when you configured the APEX repository.
- **Connection type** – Basic
- **Hostname** – the endpoint of your Amazon RDS DB instance, such as mydb.f9rbfa893tft.us-east-1.rds.amazonaws.com.
- **Port** – 1521
- **SID** – the name of the database on your Amazon RDS DB instance, such as mydb.

6. Choose **Apply**. The APEX administration window appears.

7. Set a password for the APEX admin user. To do this, use SQL*Plus to connect to your DB instance as the master user, and then run the following commands.

```
EXEC rdsadmin.rdsadmin_util.grant_apex_admin_role;
grant APEX_ADMINISTRATOR_ROLE to master;
@/home/apexuser/apex/apxchpwd.sql
```

Replace *master* with your master user name. When the apxchpwd.sql script prompts you, enter a new admin password.

8. Return to the APEX administration window in your browser and choose **Administration**. Next, choose **Application Express Internal Administration**. When you are prompted for credentials, enter the following information:
    - **User name** – admin
    - **Password** – the password you set using the apxchpwd.sql script

Choose **Login**, and then set a new password for the admin user.

Your listener is now ready for use.

Upgrading the APEX version

To upgrade APEX with your DB instance, do the following:

- Create a new option group for the upgraded version of your DB instance.
- Add the upgraded versions of APEX and APEX-DEV to the new option group. Be sure to include any other options that your DB instance uses.
- When you upgrade your DB instance, specify the new option group for your upgraded DB instance.

After you upgrade your version of APEX, the APEX schema for the previous version might still exist in your database. If you don't need it anymore, you can drop the old APEX schema from your database after you upgrade.

If you upgrade the APEX version and RESTful services were not configured in the previous APEX version, we recommend that you configure RESTful services.

In some cases when you plan to do a major version upgrade of your DB instance, you might find that you're using an APEX version that isn't compatible with your target database version. In these cases, you can upgrade your version of APEX before you upgrade your DB instance. Upgrading APEX first can reduce the amount of time that it takes to upgrade your DB instance.

Removing the APEX option

You can remove the Amazon RDS APEX options from a DB instance. To remove the APEX options from a DB instance, do one of the following:

- To remove the APEX options from multiple DB instances, remove the APEX options from the option group they belong to. This change affects all DB instances that use the option group. When you remove the APEX options from an option group that is attached to multiple DB instances, a brief outage occurs while all the DB instances are restarted.
- To remove the APEX options from a single DB instance, modify the DB instance and specify a different option group that doesn't include the APEX options. You can specify the default (empty) option group, or a different custom

option group. When you remove the APEX options, a brief outage occurs while your DB instance is automatically restarted.

When you remove the APEX options from a DB instance, the APEX schema is removed from your database.

Oracle Enterprise Manager Database Express

Amazon RDS supports Oracle Enterprise Manager (OEM) Database Express through the use of the OEM option. Amazon RDS supports Oracle Enterprise Manager Database Express for Oracle 19c, Oracle 18c, and Oracle 12c.

OEM Database Express and Database Control are similar tools that have a web-based interface for Oracle database administration.

The following is a limitation for OEM Database Express:

- OEM Database Express isn't supported on the db.t3.micro or db.t3.small DB instance classes.

OEM Database option settings

Amazon RDS supports the following settings for the OEM option.

Option setting	Valid values	Description
Port	An integer value	The port on the DB instance that listens for OEM Database. The default for OEM Database Express is 5500.
Security Groups	—	A security group that has access to **Port**.

Adding the OEM Database option

The general process for adding the OEM option to a DB instance is the following:

1. Create a new option group, or copy or modify an existing option group.
2. Add the option to the option group.
3. Associate the option group with the DB instance.

When you add the OEM option for an Oracle 12c or later DB instance, a brief outage occurs while your DB instance is automatically restarted.

To add the OEM option to a DB instance

1. Determine the option group you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group with the following settings:
    a. For **Engine** choose the oracle edition for your DB instance.
    b. For **Major engine version** choose the version of your DB instance.

2. Add the OEM option to the option group, and configure the option settings.
3. Apply the option group to a new or existing DB instance:
    - For a new DB instance, you apply the option group when you launch the instance.
    - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group. When you add the OEM option for an Oracle 19c, Oracle 18c, or Oracle 12c DB instance, a brief outage occurs while your DB instance is automatically restarted.

Using OEM Database

After you enable the OEM option, you can begin using the OEM Database tool from your web browser.

You can access either OEM Database Control or OEM Database Express from your web browser. For example, if the endpoint for your Amazon RDS DB instance is mydb.f9rbfa893tft.us-east-1.rds.amazonaws.com, and your OEM port is 1158, then the URL to access the OEM Database Control the following.

```
https://mydb.f9rbfa893tft.us-east-1.rds.amazonaws.com:1158/em
```

When you access either tool from your web browser, a login window appears that prompts you for a user name and password. Type the master user name and master password for your DB instance. You are now ready to manage your Oracle databases.

Modifying OEM Database settings

After you enable OEM Database, you can modify the Security Groups setting for the option.

You can't modify the OEM port number after you have associated the option group with a DB instance. To change the OEM port number for a DB instance, do the following:

1. Create a new option group.
2. Add the OEM option with the new port number to the new option group.
3. Remove the existing option group from the DB instance.
4. Add the new option group to the DB instance.

Using OEM Database

You can use Amazon RDS procedures to run certain OEM Database Express tasks. By running these procedures, you can do the tasks listed following.

Switching the website front end for OEM Database Express to Adobe Flash

This task is only available on Oracle DB instances running version 19c or later.

Starting with Oracle 19c, Oracle has deprecated the former OEM Database Express user interface, which was based on Adobe Flash. Instead, OEM Database Express now uses an interface built with Oracle JET. If you experience difficulties with the new interface, you can switch back to the deprecated Flash-based interface. Difficulties you might experience with the new interface include being stuck on a Loading screen after logging in to OEM Database Express. You might also miss certain features that

were present in the Flash-based version of OEM Database Express.

To switch the OEM Database Express website front end to Adobe Flash, run the Amazon RDS procedure rdsadmin.rdsadmin_oem_tasks.em_express_frontend_to_flash. This procedure is equivalent to the execemx emx SQL command.

Security best practices discourage the use of Adobe Flash. Although you can revert to the Flash-based OEM Database Express, we recommend the use of the JET-based OEM Database Express websites if possible. If you revert to using Adobe Flash and want to switch back to using Oracle JET, use the rdsadmin.rdsadmin_oem_tasks.em_express_frontend_to_jet procedure. After an Oracle database upgrade, a newer version of Oracle JET might resolve JET-related issues in OEM Database Express.

The following procedure invocation creates a task to switch the OEM Database Express website to Adobe Flash and returns the ID of the task.

```
SELECT rdsadmin.rdsadmin_oem_tasks.em_express_frontend_to_flash() as TASK_ID from DUAL;
```

You can view the result by displaying the task's output file.

```
SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('BDUMP','dbtask-task-id.log'));
```

Replace *task-id* with the task ID returned by the procedure.

You can also view the contents of the task's output file in the AWS Management Console by searching the log entries in the **Logs & events** section for the task-id.

Switching the website front end for OEM Database Express to Oracle JET

To switch the OEM Database Express website front end to Oracle JET, run the Amazon RDS procedure rdsadmin.rdsadmin_oem_tasks.em_express_frontend_to_jet. This procedure is equivalent to the execemx omx SQL command.

By default, the OEM Database Express websites for Oracle DB instances running 19c or later use Oracle JET. If you used the rdsadmin.rdsadmin_oem_tasks.em_express_frontend_to_flash procedure to switch the OEM Database Express website front end to Adobe Flash, you can switch back to Oracle JET. To do this, use the rdsadmin.rdsadmin_oem_tasks.em_express_frontend_to_jet procedure.

The following procedure invocation creates a task to switch the OEM Database Express website to Oracle JET and returns the ID of the task.

```
SELECT rdsadmin.rdsadmin_oem_tasks.em_ex-
press_frontend_to_jet() as TASK_ID from DUAL;
```

You can view the result by displaying the task's output file.

```
SELECT text FROM table(rdsadmin.rds_file_u-
til.read_text_file('BDUMP','dbtask-task-id.log'));
```

Replace *task-id* with the task ID returned by the procedure.

You can also view the contents of the task's output file in the AWS Management Console by searching the log entries in the **Logs & events** section for the task-id.

Removing the OEM Database option

You can remove the OEM option from a DB instance. When you remove the OEM option for an Oracle 12c or later DB instance, a brief outage occurs while

your instance is automatically restarted. Therefore, after you remove the OEM option, you don't need to restart your DB instance.

To remove the OEM option from a DB instance, do one of the following:

- Remove the OEM option from the option group it belongs to. This change affects all DB instances that use the option group.
- Modify the DB instance and specify a different option group that doesn't include the OEM option. This change affects a single DB instance. You can specify the default (empty) option group, or a different custom option group.

Oracle Management Agent for Enterprise Manager Cloud Control

Oracle Enterprise Manager (OEM) Management Agent is a software component that monitors targets running on hosts and communicates that information to the middle-tier Oracle Management Service (OMS).

Amazon RDS supports Management Agent through the use of the OEM_AGENT option. Management Agent requires an Amazon RDS DB instance running Oracle Database version 19.0.0.0, 18.0.0.0, 12.2.0.1, or 12.1.0.2.

Amazon RDS supports Management Agent for the following versions of OEM:

- Oracle Enterprise Manager Cloud Control for 13c

- Oracle Enterprise Manager Cloud Control for 12c

Prerequisites for Management Agent

To use Management Agent, ensure that you meet the following prerequisites.

General prerequisites

Following are general prerequisites for using Management Agent:

- You need an Oracle Management Service (OMS) that is configured to connect to your Amazon RDS DB instance.
- In most cases, you must configure your VPC to allow connections from OMS to your DB instance.

- Management Agent version 13.4.0.9.v1 requires OMS version 13.4.0.9 or later and patch 32198287.
- Ensure that you have sufficient storage space for your OEM release:
    - At least 8.5 GiB for OEM 13c Release 4
    - At least 8.5 GiB for OEM 13c Release 3
    - At least 5.5 GiB for OEM 13c Release 2
    - At least 4.5 GiB OEM 13c Release 1
    - At least 2.5 GiB for OEM 12c
- If you are using Management Agent versions OEM_AGENT 13.2.0.0.v3 and 13.3.0.0.v2, and if you want to use TCPS connectivity, follow the instructions in Configuring third party CA certificates for communication with target databases in the Oracle documentation. Also, update the JDK on your OMS by following the instructions in the Oracle document with the Oracle Doc ID 2241358.1.

This step ensures that OMS supports all the cipher suites that the database supports.

TCPS connectivity between the Management Agent and the DB instance is only supported for Management Agent versions `OEM_AGENT 13.2.0.0.v3` and `13.3.0.0.v2`.

Oracle Database release prerequisites

Following are the supported Oracle Database versions for each Management Agent version.

Management Agent version	Oracle 19c	Oracle 18c	Oracle 12c version 12.2	Oracle 12c version 12.1
13.4.0.9.v1	Supported	Supported	Supported	Supported
13.3.0.0.v2	Supported	Supported	Supported	Supported
13.3.0.0.v1	Supported	Supported	Supported	Supported
13.2.0.0.v3	Supported	Supported	Supported	Supported
13.2.0.0.v2	Supported	Supported	Supported	Supported
13.2.0.0.v1	Supported	Supported	Supported	Supported
13.1.0.0.v1	Supported	Supported	Supported	Supported
12.1.0.5.v1	Not supported	Supported	Supported	Supported
12.1.0.4.v1	Not supported	Supported	Supported	Supported

Following are prerequisites for different database versions:

- For an Amazon RDS DB instance running Oracle Database version 19.0.0.0, the minimum AGENT_VERSION is 13.1.0.0.v1.
- For an Amazon RDS DB instance running Oracle Database version 18.0.0.0 or higher, meet the following requirements:
  - For OMS 13c2, apply the Enterprise Manager 13.2 Master Bundle Patch List, which includes plugins 13.2.1, 13.2.2, 13.2.3, 13.2.4 (Oracle Doc ID 2219797.1).
  - For OMS 13c2, apply the OMS PSU System Patch 28970534.
  - For OMS 13c2, apply the OMS-Side Plugin System 13.2.2.0.190131 Patch 29201709.

- For an Amazon RDS DB instance running Oracle version 12.2.0.1 or lower, meet the following requirements:
    - For OMS 13c Release 2 with Oracle patch 25163555 applied, use OEM Agent 13.2.0.0.v2 or later.

Use OMSPatcher to apply the patch.

- For unpatched OMS 13c Release 2, use OEM Agent 13.2.0.0.v1.

Use OMSPatcher to apply patches.

OMS host communication prerequisites

Make sure that your OMS host and your Amazon RDS DB instance can communicate. Do the following:

- To connect from the Management Agent to your OMS, if your OMS is behind a firewall, add the IP addresses of your DB instances to your OMS.

Make sure the firewall for the OMS allows traffic from both the DB listener port (default 1521) and the OEM Agent port (default 3872), originating from the IP address of the DB instance.

- To connect from your OMS to the Management Agent, if your OMS has a publicly resolvable host name, add the OMS address to a security group. Your security group must have inbound rules that allow access to the DB listener port and the Management Agent port.
- To connect from your OMS to the Management Agent, if your OMS doesn't have a publicly resolvable host name, use one of the following:
    - If your OMS is hosted on an Amazon Elastic Compute Cloud (Amazon EC2) instance in a private VPC, you can set up VPC peering to connect from OMS to Management Agent.

    - If your OMS is hosted on-premises, you can set up a VPN connection to allow access from OMS to Management Agent.

Limitations for Management Agent

Following are some limitations to using Management Agent:

- Administrative tasks such as job execution and database patching, that require host credentials, aren't supported.
- Host metrics and the process list aren't guaranteed to reflect the actual system state. Thus, you shouldn't use OEM to monitor the root file system or mount point file system.
- Autodiscovery isn't supported. You must manually add database targets.
- OMS module availability depends on your database edition. For example, the database

performance diagnosis and tuning module is only available for Oracle Database Enterprise Edition.

- Management Agent consumes additional memory and computing resources. If you experience performance problems after enabling the OEM_AGENT option, we recommend that you scale up to a larger DB instance class.
- The user running the OEM_AGENT on the Amazon RDS host doesn't have operating system access to the alert log. Thus, you can't collect metrics for DB Alert Log and DB Alert Log Error Status in OEM.

Option settings for Management Agent

Amazon RDS supports the following settings for the Management Agent option.

Option setting	Required	Valid values	Description

Version (AGENT_VERSION)	Yes	13.4.0.9.v1 13.3.0.0.v2 13.3.0.0.v1 13.2.0.0.v3 13.2.0.0.v2 13.2.0.0.v1 13.1.0.0.v1 12.1.0.5.v1 12.1.0.4.v1	The version of the Management Agent software. The AWS CLI option name is OptionVersion. **Note** In the AWS GovCloud (US) Regions, 12.1 and 13.1 versions aren't available.
Port (AGENT_PORT)	Yes	An integer value	The port on the DB instance that listens for the OMS host. The default is 3872. Your OMS host must belong to a security group that has access to this port. The AWS CLI option name is Port.
Security Groups	Yes	Existing security groups	A security group that has access to **Port**. Your OMS host must belong to this security group. The AWS CLI option name is VpcSecurity-GroupMemberships or DBSecurityGroupMem-berships.
OMS_HOST	Yes	A string value, for example *my.example.oms*	The publicly accessible host name or IP address of the OMS. The AWS CLI option name is OMS_HOST.
OMS_PORT	Yes	An integer value	The HTTPS upload port on the OMS Host that listens for the Management Agent. To determine the HTTPS upload port, connect to the OMS host, and run the following command (which requires the SYSMAN password): emctl status oms -details The AWS CLI option name is OMS_PORT.
AGENT_REGISTRATION_PASSWORD	Yes	A string value	The password that the Management Agent uses to authenticate itself with the OMS. We recommend that you create a persistent password in your OMS before enabling the OEM_A-GENT option. With a persistent password you can share a single Management Agent option group among multiple Amazon RDS databases. The AWS CLI option name is AGENT_REGIS-TRATION_PASSWORD.
ALLOW_TLS_ONLY	No	true, false (default)	A value that configures the OEM Agent to support only the TLSv1 protocol while the agent listens as a server. This setting is only supported for 12.1 agent versions. Later agent versions only support Transport Layer Security (TLS) by default.
MINIMUM_TLS_VERSION	No	TLSv1 (default), TLSv1.2	A value that specifies the minimum TLS version supported by the OEM Agent while the agent listens as a server. This setting is only supported for agent versions 13.1.0.0.v1 and higher. Earlier agent versions only support the TLSv1 setting.
TLS_CIPHER_SUITE	No	TLS_RSA_WITH_AES_128_CBC_SHA (Default supported by all agent versions) TLS_RSA_WITH_AES_128_CBC_SHA256 (Requires version 13.1.0.0.v1 or above) TLS_RSA_WITH_AES_256_CBC_SHA (Requires version 13.2.0.0.v3 or above) TLS_RSA_WITH_AES_256_CBC_SHA256 (Requires version 13.2.0.0.v3 or above)	A value that specifies the TLS cipher suite used by the OEM Agent while the agent listens as a server.

Adding the Management Agent option

The general process for adding the Management Agent option to a DB instance is the following:

1. Create a new option group, or copy or modify an existing option group.
2. Add the option to the option group.
3. Associate the option group with the DB instance.

After you add the Management Agent option, you don't need to restart your DB instance. As soon as the option group is active, the OEM Agent is active.

If your OMS host is using an untrusted third-party certificate, Amazon RDS returns the following error.

You successfully installed the OEM_AGENT option.
Your OMS host is using an untrusted third party certificate.
Configure your OMS host with the trusted certificates from your third party.

Console

To add the Management Agent option to a DB instance

1. Determine the option group you want to use. You can create a new option group or use an existing option group. If you want to use an existing option group, skip to the next step. Otherwise, create a custom DB option group with the following settings:
    a. For **Engine** choose the oracle edition for your DB instance.
    b. For **Major engine version** choose the version of your DB instance.

2. Add the **OEM_AGENT** option to the option group, and configure the option settings.
3. Apply the option group to a new or existing DB instance:
    - For a new DB instance, you apply the option group when you launch the instance.
    - For an existing DB instance, you apply the option group by modifying the instance and attaching the new option group.

Using the Management Agent

After you enable the Management Agent option, take the following steps to begin using it.

To use the Management Agent

1. Unlock and reset the DBSNMP account credential. Do this by running the following code on your target database on your DB instance and using your master user account.

```
ALTER USER dbsnmp IDENTIFIED BY new_password ACCOUNT UNLOCK;
```

2. Add your targets to the OMS console manually:
    a. In your OMS console, choose **Setup**, **Add Target**, **Add Targets Manually**.
    b. Choose **Add Targets Declaratively by Specifying Target Monitoring Properties**.
    c. For **Target Type**, choose **Database Instance**.
    d. For **Monitoring Agent**, choose the agent with the identifier that is the same as your RDS DB instance identifier.
    e. Choose **Add Manually**.

f. Enter the endpoint for the Amazon RDS DB instance, or choose it from the host name list. Make sure that the specified host name matches the endpoint of the Amazon RDS DB instance.

g. Specify the following database properties:

- For **Target name**, enter a name.
- For **Database system name**, enter a name.
- For **Monitor username**, enter **dbsnmp**.
- For **Monitor password**, enter the password from step 1.
- For **Role**, enter **normal**.
- For **Oracle home path**, enter **/oracle**.
- For **Listener Machine name**, the agent identifier already appears.
- For **Port**, enter the database port. The RDS default port is 1521.
- For **Database name**, enter the name of your database.

h. Choose **Test Connection**.
i. Choose **Next**. The target database appears in your list of monitored resources.

Modifying Management Agent settings

After you enable the Management Agent, you can modify settings for the option.

Performing database tasks with the Management Agent

You can use Amazon RDS procedures to run certain EMCTL commands on the Management Agent. By running these procedures, you can do the tasks listed following.

Getting the status of the Management Agent

To get the status of the Management Agent, run the Amazon RDS procedure rdsadmin.rdsadmin_oem_agent_tasks.get_status_oem_agent. This procedure is equivalent to the emctl status agent command.

The following procedure creates a task to get the Management Agent's status and returns the ID of the task.

```
SELECT rdsadmin.rdsadmin_oem_agent_
tasks.get_status_oem_agent() as TASK_ID from
DUAL;
```

Restarting the Management Agent

To restart the Management Agent, run the Amazon RDS procedure rdsadmin.rdsadmin_oem_agent_tasks.get_status_oem_agent. This procedure is equivalent to running the emctl stop agent and emctl start agent commands.

The following procedure creates a task to restart the Management Agent and returns the ID of the task.

```
SELECT rdsadmin.rdsadmin_oem_agent_
tasks.restart_oem_agent() as TASK_ID from DUAL;
```

Listing the targets monitored by the Management Agent

To list the targets monitored by the Management Agent, run the Amazon RDS procedure `rdsadmin.rdsadmin_oem_agent_tasks.list_targets_oem_agent`. This procedure is equivalent to running the `emctl config agent listtargets` command.

The following procedure creates a task to list the targets monitored by the Management Agent and returns the ID of the task.

```
SELECT rdsadmin.rdsadmin_oem_agent_
tasks.list_targets_oem_agent() as TASK_ID from
DUAL;
```

Listing the collection threads monitored by the Management Agent

To list of all the running, ready, and scheduled collection threads monitored by the Management Agent, run the Amazon RDS procedure `rdsadmin.rdsadmin_oem_agent_tasks.list_clxn_threads_oem_agent`. This procedure is equivalent to the `emctl status agent scheduler` command.

The following procedure creates a task to list the collection threads and returns the ID of the task.

```
SELECT rdsadmin.rdsadmin_oem_agent_
tasks.list_clxn_threads_oem_agent() as TASK_ID
from DUAL;
```

Clearing the Management Agent state

To clear the Management Agent's state, run the Amazon RDS procedure `rdsadmin.rdsadmin_oem_agent_tasks.clearstate_oem_agent`. This procedure is equivalent to running the `emctl clearstate agent` command.

The following procedure creates a task that clears the Management Agent's state and returns the ID of the task.

```
SELECT rdsadmin.rdsadmin_oem_agent_tasks.
clearstate_oem_agent() as TASK_ID from DUAL;
```

Making the Management Agent upload its OMS

To make the Management Agent upload the Oracle Management Server (OMS) associated with it, run the Amazon RDS procedure `rdsadmin.rdsadmin_oem_agent_tasks.upload_oem_agent`. This procedure is equivalent to running the `emclt upload agent` command.

The following procedure creates a task that makes the Management Agent upload its associated OMS and return the ID of the task.

```
SELECT rdsadmin.rdsadmin_oem_agent_tasks.up-
load_oem_agent() as TASK_ID from DUAL;
```

Pinging the OMS

To ping the Management Agent's OMS, run the Amazon RDS procedure `rdsadmin.rdsadmin_oem_agent_tasks.ping_oms_oem_agent`. This procedure is equivalent to running the `emctl pingOMS` command.

The following procedure creates a task that pings the Management Agent's OMS and returns the ID of the task.

```
SELECT rdsadmin.rdsadmin_oem_agent_
tasks.ping_oms_oem_agent() as TASK_ID from
DUAL;
```

Viewing the status of an ongoing task

You can view the status of an ongoing task in a bdump file. The bdump files are located in the /rds-dbdata/log/trace directory. Each bdump file name is in the following format.

```
dbtask-task-id.log
```

When you want to monitor a task, replace *task-id* with the ID of the task that you want to monitor.

To view the contents of bdump files, run the Amazon RDS procedure rdsadmin.rds_file_util.read_text_file. The following query returns the contents of the dbtask-1546988886389-2444.log bdump file.

```
SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('BDUMP','dbtask-1546988886389-2444.log'));
```

Removing the Management Agent option

You can remove the OEM Agent from a DB instance. After you remove the OEM Agent, you don't need to restart your DB instance.

To remove the OEM Agent from a DB instance, do one of the following:

- Remove the OEM Agent option from the option group it belongs to. This change affects all DB instances that use the option group.
- Modify the DB instance and specify a different option group that doesn't include the OEM Agent option. This change affects a single DB instance. You can specify the default (empty) option group, or a different custom option group.

Setting up Amazon RDS to host tools and third-party software for Oracle

You can use Amazon RDS to host an Oracle DB instance that supports software and components such as the following:

- Siebel Customer Relationship Management (CRM)
- Oracle Fusion Middleware Metadata — installed by the Repository Creation Utility (RCU)

The following procedures help you create an Oracle DB instance on Amazon RDS that you can use to host additional software and components for Oracle.

Creating a VPC for use with an Oracle database

In the following procedure, you create a virtual private cloud (VPC) based on the Amazon VPC service, a private subnet, and a security group. Your Amazon RDS DB instance needs to be available only to your middle-tier components, and not to the public internet. Thus, your Amazon RDS DB instance is hosted in a private subnet, providing greater security.

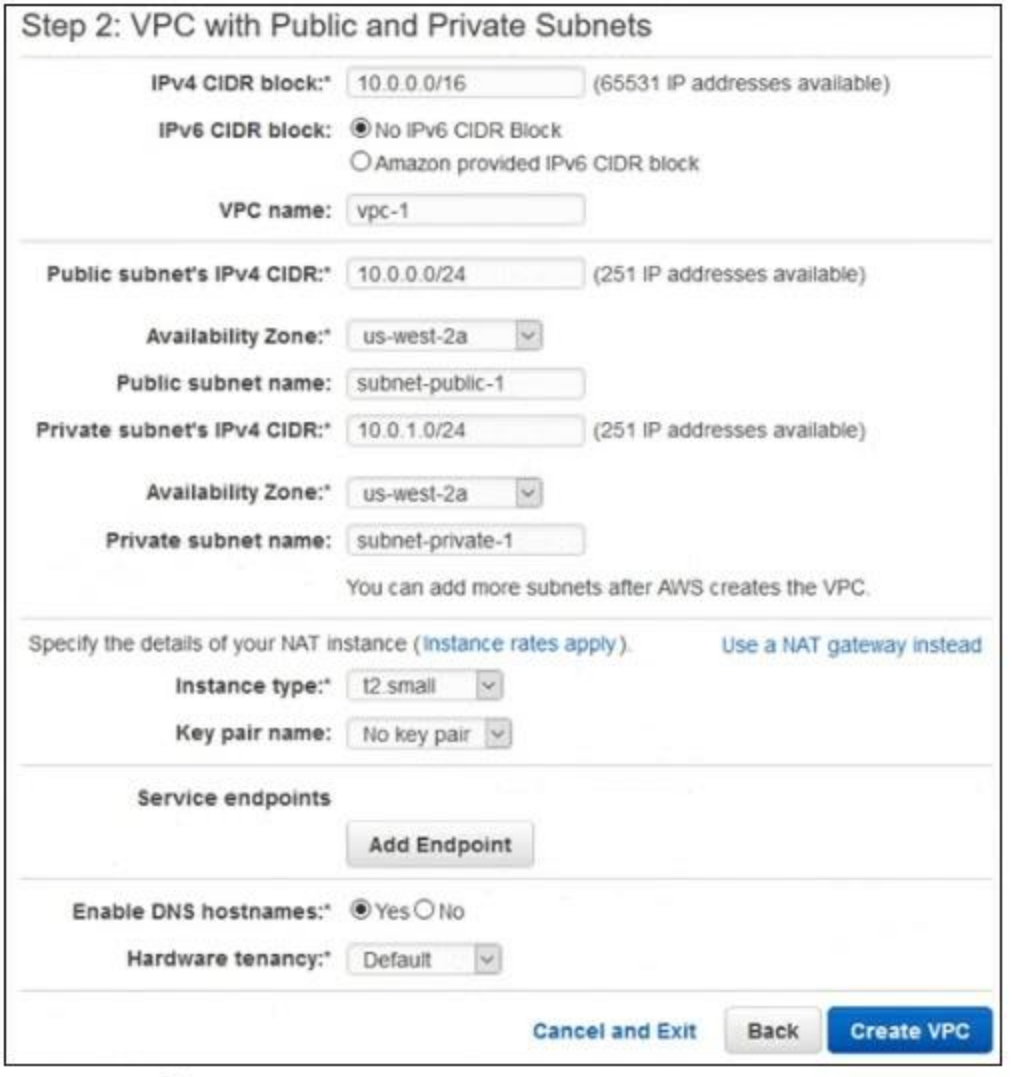
To create a VPC based on Amazon VPC

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the upper-right corner of the AWS Management Console, choose the AWS Region for your VPC. This example uses the US West (Oregon) region.
3. In the upper-left corner, choose **VPC Dashboard**, and then choose **Start VPC Wizard**.

4. On the page **Step 1: Select a VPC Configuration**, choose **VPC with Public and Private Subnets**, and then choose **Select**.
5. On the page **Step 2: VPC with Public and Private Subnets**, shown following, set the following values.

Option	Value
IPv4 CIDR block	10.0.0.0/16
IPv6 CIDR block	**No IPv6 CIDR Block**
VPC name	The name for your VPC, for example **vpc-1**.
Public subnet's IPv4 CIDR	10.0.0.0/24
Availability Zone	An Availability Zone for your AWS Region.
Public subnet name	The name for your public subnet, for example **subnet-public-1**.

Private subnet's IPv4 CIDR	10.0.1.0/24
Availability Zone	An Availability Zone for your AWS Region.
Private subnet name	The name for your private subnet, for example **subnet-private-1**.
Instance type	An instance type for your NAT instance, for example t2.small.
Key pair name	No key pair
Service endpoints	None
Enable DNS hostnames	Yes
Hardware tenancy	Default



6. Choose **Create VPC**.

An Amazon RDS DB instance in a VPC requires at least two private subnets or at least two public subnets, to support Multi-AZ deployment.

To create an additional subnet

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the upper-right corner of the AWS Management Console, confirm that you are in the correct AWS Region for your VPC.
3. In the upper-left corner, choose **VPC Dashboard**, choose **Subnets**, and then choose **Create Subnet**.
4. On the **Create Subnet** page, set the following values.

Option	Value
Name tag	The name for your second private subnet, for example **subnet-private-2**.
VPC	Your VPC, for example **vpc-1**.
Availability Zone	An Availability Zone for your AWS Region. **Note**

	Choose an Availability Zone different from the one that you chose for the first private subnet.
CIDR block	10.0.2.0/24

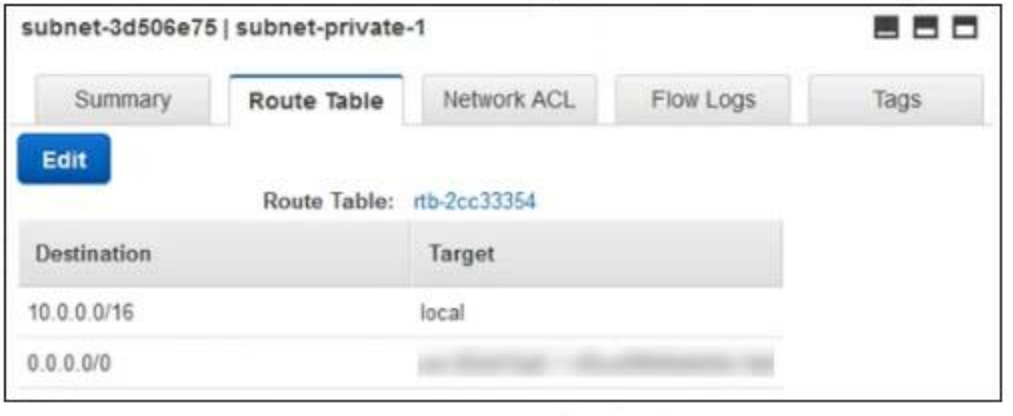
5. Choose **Yes, Create**.

Both private subnets must use the same route table. In the following procedure, you check to make sure the route tables match, and if not you edit one of them.

To ensure the subnets use the same route table.

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the upper-right corner of the AWS Management Console, confirm that you are in the correct AWS Region for your VPC.

3. In the upper-left corner, choose **VPC Dashboard**, choose **Subnets**, and then choose your first private subnet, for example **subnet-private-1**.
4. At the bottom of the console, choose the **Route Table** tab, shown following.



5. Make a note of the route table, for example rtb-0d9fc668.
6. In the list of subnets, choose the second private subnet, for example **subnet-private-2**.
7. At the bottom of the console, choose the **Route Table** tab.
8. If the route table for the second subnet is not the same as the route table for the first subnet, edit it to match:

a. Choose **Edit**.
b. For **Change to**, choose the route table that matches your first subnet.
c. Choose **Save**.

A security group acts as a virtual firewall for your DB instance to control inbound and outbound traffic. In the following procedure, you create a security group for your DB instance.

To create a VPC security group for a private Amazon RDS DB instance

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the upper-right corner of the AWS Management Console, confirm that you are in the correct AWS Region for your VPC.
3. In the upper-left corner, choose **VPC Dashboard**, choose **Security Groups**, and then choose **Create Security Group**.

4. On the page **Create Security Group**, set the following values.

Option	Value
Name tag	The name for your security group, for example **sgdb-1**.
Group name	The name for your security group, for example **sgdb-1**.
De-scrip-tion	A description for your security group.
VPC	Your VPC, for example **vpc-1**.

5. Choose **Yes, Create**.

In the following procedure, you add rules to your security group to control inbound traffic to your DB instance.

To add inbound rules to the security group

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the upper-right corner of the AWS Management Console, confirm that you are in the correct AWS Region for your VPC.
3. In the upper-left corner, choose **VPC Dashboard**, choose **Security Groups**, and then choose your security group, for example **sgdb-1**.
4. At the bottom of the console, choose the **Inbound Rules** tab, and then choose **Edit**.
5. Set these values, as shown following.

Option	Value
Type	Oracle (1521)
Protocol	TCP (6)
Port Range	1521

Source	The identifier of your security group. When you choose the box, you see the name of your security group, for example **sgdb-1**.

6. Choose **Save**.

Creating an Oracle DB instance

You can use Amazon RDS to host an Oracle DB instance. When you create the new DB instance, specify the VPC and security group. Also, choose **No** for **Publicly accessible**.

Additional Amazon RDS interfaces

In the preceding procedures, we use the AWS Management Console to perform tasks. Amazon Web Services also provides the AWS Command Line Interface (AWS CLI), and an application programming interface (API). You can use the AWS CLI or the API to automate many of the tasks for managing Amazon RDS, including tasks to manage an Oracle DB instance with Amazon RDS.

Using Oracle GoldenGate with Amazon RDS

Oracle GoldenGate (GoldenGate) collects, replicates, and manages transactional data between databases. It is a log-based change data capture (CDC) and replication software package used with Oracle databases for online transaction processing (OLTP) systems. GoldenGate creates trail files that contain the most recent changed data from the source database and then pushes these files to the target database. Amazon RDS supports GoldenGate for Oracle Database Standard Edition Two (SE2) and Enterprise Edition (EE).

You can use GoldenGate with Amazon RDS to do the following:

- Active-Active database replication
- Zero-downtime migration and upgrades
- Disaster recovery
- Data protection

- In-region and cross-region replication

When working with GoldenGate on Amazon RDS, consider the following:

- You are responsible for setting up and managing GoldenGate for use with Amazon RDS.
- You are responsible for managing GoldenGate licensing (BYOL) for use with Amazon RDS in all AWS regions.
- Amazon RDS supports GoldenGate version 11.2.1 and later.
- Amazon RDS supports migration and replication across Oracle databases using GoldenGate. We do not support nor prevent customers from migrating or replicating across heterogeneous databases.
- You can use GoldenGate on Amazon RDS for Oracle DB instances that use Oracle Transparent Data Encryption (TDE). To maintain the integrity of replicated data, you should

configure encryption on the GoldenGate hub using EBS encrypted volumes or trail file encryption. You should also configure encryption for data sent between the GoldenGate hub and the source and target database instances. Amazon RDS for Oracle DB instances support encryption with Oracle Secure Sockets Layer or Oracle native network encryption.

- GoldenGate DDL is supported with GoldenGate version 12.1 and later when using Integrated capture mode.

Overview

The GoldenGate architecture for use with Amazon RDS consists of three decoupled modules. The source database can be either an on-premises Oracle database, an Oracle database on an Amazon EC2 instance, or an Oracle database on an Amazon RDS

DB instance. You also work with a GoldenGate hub, which moves transaction information from the source database to the target database. Your hub can be either of these:

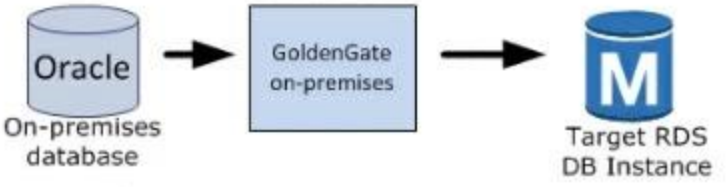
- An Amazon EC2 instance with Oracle Database and GoldenGate installed
- An on-premises Oracle installation.

You can have more than one Amazon EC2 hub, and we recommend that you use two hubs if you are using GoldenGate for cross-region replication.

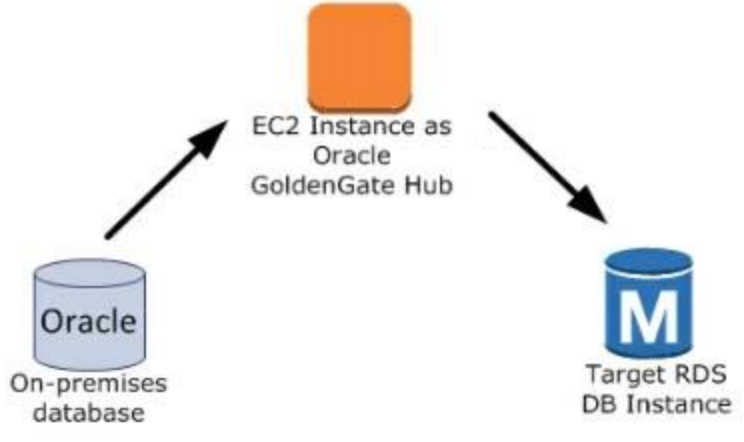
Your target database can be on either an Amazon RDS DB instance, an Amazon EC2 instance, or an on-premises location.

GoldenGate on Amazon RDS supports the following common scenarios:

Scenario 1: An on-premises Oracle source database and on-premises GoldenGate hub, that provides data to a target Amazon RDS DB instance.

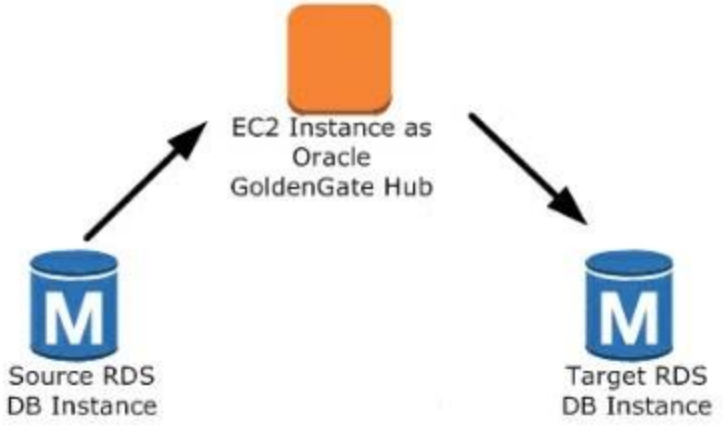


Scenario 2: An on-premises Oracle database that acts as the source database, connected to an Amazon EC2 instance hub that provides data to a target Amazon RDS DB instance.

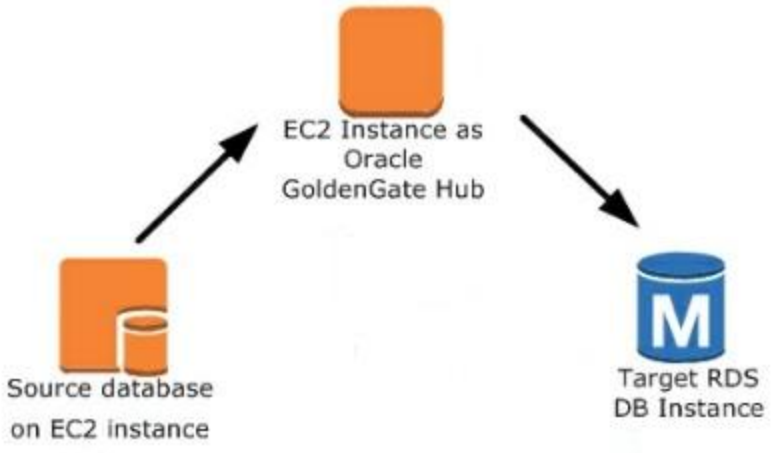


Scenario 3: An Oracle database on an Amazon RDS DB instance that acts as the source database, con-

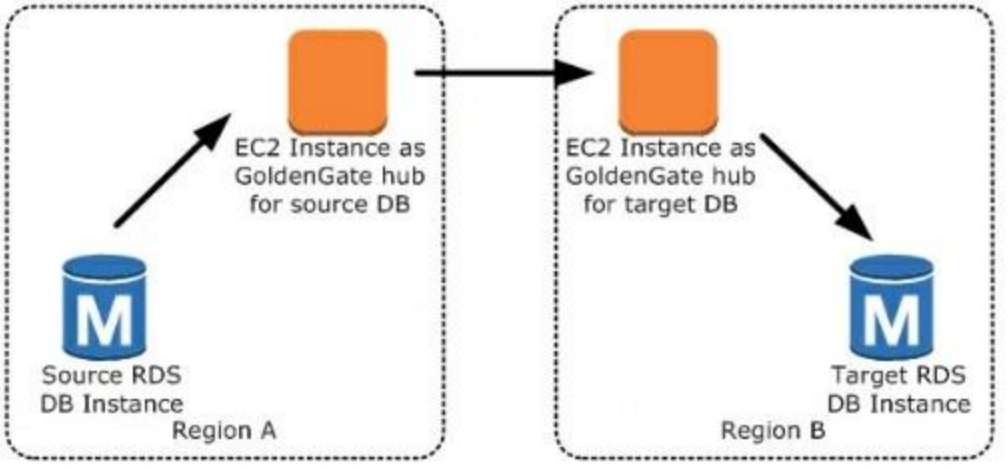
nected to an Amazon EC2 instance hub that provides data to a target Amazon RDS DB instance.



Scenario 4: An Oracle database on an Amazon EC2 instance that acts as the source database, connected to an Amazon EC2 instance hub that provides data to a target Amazon RDS DB instance.



Scenario 5: An Oracle database on an Amazon RDS DB instance connected to an Amazon EC2 instance hub in the same AWS Region. In this scenario, the hub is connected to an Amazon EC2 instance hub in a different AWS Region. This second hub provides data to the target Amazon RDS DB instance in the same AWS Region as the second Amazon EC2 instance hub.



Any issues that affect running GoldenGate on an on-premises environment also affect running GoldenGate on AWS. We strongly recommend that you monitor the GoldenGate hub to ensure that EXTRACT and REPLICAT are resumed if a failover occurs. Because the GoldenGate hub is run on an Amazon EC2 instance, Amazon RDS does not manage the GoldenGate hub and cannot ensure that it is running.

You can use GoldenGate using Amazon RDS to upgrade to major versions of Oracle. For example, you can use GoldenGate with Amazon RDS to upgrade from an Oracle version 8 on-premises database to

an Oracle database running version 19c on an Amazon RDS DB instance.

To set up GoldenGate using Amazon RDS, you configure the hub on the Amazon EC2 instance, and then configure the source and target databases. The following steps show how to set up GoldenGate for use with Amazon RDS. Each step is explained in detail in the following sections:

Setting up a GoldenGate hub on Amazon EC2

To create a GoldenGate hub on an Amazon EC2 instance, you complete several steps. First, you create an Amazon EC2 instance with a full client installation of Oracle RDBMS. The Amazon EC2 instance must also have Oracle GoldenGate software installed. The exact software versions depend on the source and target database versions.

The Amazon EC2 instance that serves as the GoldenGate hub stores and processes the transaction information from the source database into trail files. To support this process, make sure that you meet the following conditions:

- You have allocated enough storage for the trail files
- The Amazon EC2 instance has enough processing power to manage the amount of data.
- The EC2 instance has enough memory to store the transaction information before it's written to the trail file.

The following tasks set up a GoldenGate hub on an Amazon EC2 instance; each task is explained in detail in this section:

1. Create the GoldenGate subdirectories.
2. Update the GLOBALS parameter file.

3. Configure the mgr.prm file and start the manager.

Create subdirectories in the GoldenGate directory using the Amazon EC2 command line shell and *ggsci*, the GoldenGate command interpreter. The subdirectories are created under the gg directory and include directories for parameter, report, and checkpoint files.

```
prompt$ cd /gg
prompt$ ./ggsci
 GGSCI> CREATE SUBDIRS
```

Create a GLOBALS parameter file using the Amazon EC2 command line shell. Parameters that affect all GoldenGate processes are defined in the GLOBALS parameter file. The following example creates the necessary file:

```
$ cd $GGHOME
$ vi GLOBALS
```

```
CheckpointTable oggadm1.oggchkpt
```

The last step in setting up and configuring the GoldenGate hub is to configure the *manager*. Add the following lines to the *mgr.prm* file, then start the *manager* using *ggsci*:

```
PORT 8199
PurgeOldExtracts ./dirdat/*, UseCheckpoints, MIN-
KEEPDAYS 5
GGSCI>  start mgr
```

Once you have completed these steps, the GoldenGate hub is ready for use. Next, you set up the source and target databases.

Setting up a source database for use

with GoldenGate on Amazon RDS

When your source Oracle database is running version 12c or later, complete the following tasks to set up a source database for use with GoldenGate:

1. Set the ENABLE_GOLDENGATE_REPLICATION parameter to *True*. This parameter turns on supplemental logging for the source database. If your source database is on an Amazon RDS DB instance, make sure that you have a parameter group assigned to the DB instance with the ENABLE_GOLDENGATE_REPLICATION parameter set to *true*.
2. Set the retention period for archived redo logs for the GoldenGate source database.
3. Create a GoldenGate user account on the source database.
4. Grant the necessary privileges to the GoldenGate user.
5. Add a TNS alias for the source database to the tnsnames.ora file on the GoldenGate hub.

Enable supplemental logging on the source DB

The ENABLE_GOLDENGATE_REPLICATION parameter, when set to *True*, turns on supplemental logging for the source database and configures the required GoldenGate permissions. If your source database is on an Amazon RDS DB instance, make sure that you have a parameter group assigned to the DB instance with ENABLE_GOLDENGATE_REPLICATION set to *true*.

Set the log retention period on the source DB

The source database must also retain archived redo logs. For example, set the retention period for archived redo logs to 24 hours.

```
exec rdsadmin.rdsadmin_util.set_configuration('archivelog retention hours',24);
```

Specify the duration for log retention in hours. The duration should exceed any potential downtime of

the source instance, any potential period of communication, and any potential period of networking issues for the source instance. Such a duration lets Oracle GoldenGate recover logs from the source instance as needed.

The absolute minimum value required is one hour of logs retained. If you don't have log retention enabled, or if the retention value is too small, you receive the following message.

```
2014-03-06 06:17:27  ERROR   OGG-00446  error 2
(No such file or directory)
opening redo log /rdsdbdata/db/GGTEST3_A/on-
linelog/o1_mf_2_9k4bp1n6_.log
for sequence 1306Not able to establish initial posi-
tion for begin time 2014-03-06 06:16:55.
```

The logs are retained on your DB instance. Ensure that you have sufficient storage on your instance for the files. To see how much space you have used

in the last *X* hours, use the following query, replacing *X* with the number of hours.

```
SELECT SUM(BLOCKS * BLOCK_SIZE) BYTES FROM
V$ARCHIVED_LOG
  WHERE NEXT_TIME>=SYSDATE-X/24 AND
DEST_ID=1;
```

Create a user account on the source

GoldenGate runs as a database user and requires the appropriate database privileges to access the redo and archive logs for the source database. To provide these, create a GoldenGate user account on the source database.

The following statements create a user account named *oggadm1*.

```
CREATE TABLESPACE administrator;
CREATE USER oggadm1  IDENTIFIED BY "password"
```

```
 DEFAULT TABLESPACE ADMINISTRATOR TEMPO-
RARY TABLESPACE TEMP;
```

Grant account privileges on the source DB

Grant the necessary privileges to the GoldenGate user account using the SQL command grant and the rdsadmin.rdsadmin_util procedure grant_sys_object. The following statements grant privileges to a user named *oggadm1*.

```
GRANT CREATE SESSION, ALTER SESSION TO
oggadm1;
GRANT RESOURCE TO oggadm1;
GRANT SELECT ANY DICTIONARY TO oggadm1;
GRANT FLASHBACK ANY TABLE TO oggadm1;
GRANT SELECT ANY TABLE TO oggadm1;
GRANT SELECT_CATALOG_ROLE TO rds_mas-
ter_user_name WITH ADMIN OPTION;
exec rdsadmin.rdsadmin_util.grant_sys_object
('DBA_CLUSTERS', 'OGGADM1');
```

```
GRANT EXECUTE ON DBMS_FLASHBACK TO
oggadm1;
GRANT SELECT ON SYS.V_$DATABASE TO
oggadm1;
GRANT ALTER ANY TABLE TO oggadm1;
```

Finally, grant the privileges needed by a user account to be a GoldenGate administrator. The package that you use to perform the grant, dbms_goldengate_auth or rdsadmin_dbms_goldengate_auth, depends on the Oracle DB engine version.

- For Oracle DB versions that are *earlier than* Oracle 12.2, run the following PL/SQL program.

```
exec dbms_goldengate_auth.grant_admin_privi-
lege (grantee=>'OGGADM1',
  privilege_type=>'capture',
  grant_select_privileges=>true,
  do_grants=>TRUE);
```

- For Oracle DB versions that are *later than or equal to* Oracle 12.2, which requires patch level 12.2.0.1.ru-2019-04.rur-2019-04.r1 or later, run the following PL/SQL program.

```
exec rdsadmin.rdsadmin_dbms_goldengate_auth.
grant_admin_privilege (grantee=>'OGGADM1',
 privilege_type=>'capture',
 grant_select_privileges=>true,
 do_grants=>TRUE);
```

To revoke privileges, use the procedure revoke_admin_privilege in the same package.

Add a TNS alias for the source DB

Add the following entry to $ORACLE_HOME/network/admin/tnsnames.ora in the Oracle Home to be used by the EXTRACT process.

```
OGGSOURCE=
 (DESCRIPTION=
    (ENABLE=BROKEN)
```

```
    (ADDRESS_LIST=
      (ADDRESS=(PROTOCOL=TCP)
(HOST=goldengate-source.abcdef12345.us-
west-2.rds.amazonaws.com)(PORT=8200)))
    (CONNECT_DATA=(SID=ORCL))
  )
```

Setting up a target database for use with GoldenGate on Amazon RDS

The following tasks set up a target DB instance for use with GoldenGate:

1. Set the ENABLE_GOLDENGATE_REPLICATION parameter to TRUE. If your target database is on an Amazon RDS DB instance, make sure that you have a parameter group assigned to the DB instance with the ENABLE_GOLDENGATE_REPLICATION parameter set to TRUE.

2. Create and manage a GoldenGate user account on the target database
3. Grant the necessary privileges to the GoldenGate user
4. Add a TNS alias for the target database to tnsnames.ora on the GoldenGate hub.

Create a user account on the target DB

GoldenGate runs as a database user and requires the appropriate database privileges. To make sure it has these, create a GoldenGate user account on the target database.

The following statements create a user named *oggadm1*:

```
CREATE TABLESPACE administrator;
CREATE TABLESPACE administrator_idx;
CREATE USER oggadm1  IDENTIFIED BY "password"
  DEFAULT TABLESPACE administrator
  TEMPORARY TABLESPACE temp;
```

```
ALTER USER oggadm1 QUOTA UNLIMITED ON
administrator;
ALTER USER oggadm1 QUOTA UNLIMITED ON
administrator_idx;
```

Grant account privileges on the target DB

Grant necessary privileges to the GoldenGate user account on the target DB. In the following example, you grant privileges to *oggadm1*.

```
GRANT CREATE SESSION      TO oggadm1;
GRANT ALTER SESSION       TO oggadm1;
GRANT CREATE CLUSTER      TO oggadm1;
GRANT CREATE INDEXTYPE    TO oggadm1;
GRANT CREATE OPERATOR     TO oggadm1;
GRANT CREATE PROCEDURE    TO oggadm1;
GRANT CREATE SEQUENCE     TO oggadm1;
GRANT CREATE TABLE        TO oggadm1;
GRANT CREATE TRIGGER      TO oggadm1;
GRANT CREATE TYPE         TO oggadm1;
```

```
GRANT SELECT ANY DICTIONARY TO oggadm1;
GRANT CREATE ANY TABLE    TO oggadm1;
GRANT ALTER ANY TABLE     TO oggadm1;
GRANT LOCK ANY TABLE      TO oggadm1;
GRANT SELECT ANY TABLE    TO oggadm1;
GRANT INSERT ANY TABLE    TO oggadm1;
GRANT UPDATE ANY TABLE    TO oggadm1;
GRANT DELETE ANY TABLE    TO oggadm1;
```

Finally, grant the privileges needed by a user account to be a GoldenGate administrator. The package that you use to perform the grant, dbms_goldengate_auth or rdsadmin_dbms_goldengate_auth, depends on the Oracle DB engine version.

- For Oracle DB versions that are *earlier than* Oracle 12.2, run the following PL/SQL program.

```
exec dbms_goldengate_auth.grant_admin_privilege (grantee=>'OGGADM1',
  privilege_type=>'capture',
```

```
  (DESCRIPTION=
    (ENABLE=BROKEN)
    (ADDRESS_LIST=
      (ADDRESS=(PROTOCOL=TCP)
(HOST=goldengate-target.abcdef12345.us-
west-2.rds.amazonaws.com)(PORT=8200)))
    (CONNECT_DATA=(SID=ORCL))
  )
```

Working with the EXTRACT and

REPLICAT utilities of GoldenGate

The GoldenGate utilities EXTRACT and REPLICAT work together to keep the source and target databases in sync via incremental transaction replication using trail files. All changes that occur on the source database are automatically detected by EXTRACT, then formatted and transferred to trail files on the GoldenGate on-premises or EC2-instance hub. After initial load is completed, the data

is read from these files and replicated to the target database by the REPLICAT utility.

Running the GoldenGate EXTRACT utility

The EXTRACT utility retrieves, converts, and outputs data from the source database to trail files. EXTRACT queues transaction details to memory or to temporary disk storage. When the transaction is committed to the source database, EXTRACT flushes all of the transaction details to a trail file. The trail file routes these details to the GoldenGate on-premises or the EC2 instance hub and then to the target database.

The following tasks enable and start the EXTRACT utility:

1. Configure the EXTRACT parameter file on the GoldenGate hub (on-premises or EC2 instance). The following listing shows an example EXTRACT parameter file.

```
EXTRACT EABC
SETENV (ORACLE_SID=ORCL)
SETENV (NLSLANG=AL32UTF8)

USERID oggadm1@OGGSOURCE, PASSWORD
XXXXXX
EXTTRAIL /path/to/goldengate/dirdat/ab

IGNOREREPLICATES
GETAPPLOPS
TRANLOGOPTIONS EXCLUDEUSER OGGADM1

TABLE EXAMPLE.TABLE;
```

2. On the GoldenGate hub, launch the GoldenGate command line interface (*ggsci*). Log into the source database. The following example shows the format for logging in:

```
dblogin userid <user>@<db tnsname>
```

3. Add a checkpoint table for the database:

```
add checkpointtable
```

4. Add transdata to turn on supplemental logging for the database table:

```
add trandata <user>.<table>
```

Alternatively, you can add transdata to turn on supplemental logging for all tables in the database:

```
add trandata <user>.*
```

5. Using the ggsci command line, enable the EXTRACT utility using the following commands:

```
add extract <extract name> tranlog, INTEGRATED
tranlog, begin now
add exttrail <path-to-trail-from-the param-file>
 extract <extractname-from-paramfile>,
 MEGABYTES Xm
```

6. Register the EXTRACT utility with the database so that the archive logs are not deleted. This allows you to recover old, uncommitted transactions if necessary. To register the EXTRACT utility with the database, use the following command:

```
register EXTRACT <extract process name>,
DATABASE
```

7. To start the EXTRACT utility, use the following command:

```
start <extract process name>
```

Running the GoldenGate REPLICAT utility

The REPLICAT utility is used to "push" transaction information in the trail files to the target database.

The following tasks enable and start the REPLICAT utility:

1. Configure the REPLICAT parameter file on the GoldenGate hub (on-premises or EC2 instance). The following listing shows an example REPLICAT parameter file.

```
REPLICAT RABC
SETENV (ORACLE_SID=ORCL)
SETENV (NLSLANG=AL32UTF8)
```

```
USERID oggadm1@OGGTARGET, password
XXXXXX

ASSUMETARGETDEFS
MAP EXAMPLE.TABLE, TARGET EXAMPLE.TABLE;
```

2. Launch the GoldenGate command line interface (ggsci). Log into the target database. The following example shows the format for logging in.

```
dblogin userid <user>@<db tnsname>
```

3. Using the *ggsci* command line, add a checkpoint table. The user indicated should be the GoldenGate user account, not the target table schema owner. The following example creates a checkpoint table named *gg_checkpoint*.

```
add checkpointtable <user>.gg_checkpoint
```

4. To enable the REPLICAT utility, use the following command.

```
add replicat <replicat name> EXTTRAIL <extract trail file> CHECKPOINTTABLE <user>.gg_checkpoint
```

5. To start the REPLICAT utility, use the following command.

```
start <replicat name>
```

Troubleshooting issues when using

GoldenGate with Amazon RDS

This section explains the most common issues when using Oracle GoldenGate with Amazon RDS.

Log retention

To work with Oracle GoldenGate with Amazon RDS, make sure that you have log retention enabled.

Specify the duration for log retention in hours. The duration should exceed any potential downtime of the source instance, any potential period of com-

munication, and any potential period of networking issues for the source instance. Such a duration lets Oracle GoldenGate recover logs from the source instance as needed.

The absolute minimum value required is one hour of logs retained. If you don't have log retention enabled, or if the retention value is too small, you receive the following message.

```
2014-03-06 06:17:27  ERROR   OGG-00446  error 2
(No such file or directory)
opening redo log /rdsdbdata/db/GGTEST3_A/on-
linelog/o1_mf_2_9k4bp1n6_.log
for sequence 1306Not able to establish initial posi-
tion for begin time 2014-03-06 06:16:55.
```

GoldenGate appears to be properly configured but replication is not working

For pre-existing tables, GoldenGate must be told which SCN it should work from. Take the following steps to fix this issue:

1. Launch the GoldenGate command line interface (ggsci). Log into the source database. The following example shows the format for logging.

```
dblogin userid <user>@<db tnsname>
```

2. Using the ggsci command line, set up the start SCN for the EXTRACT process. The following example sets the SCN to 223274 for the extract.

```
ALTER EXTRACT <extract process name> SCN
223274
start <extract process name>
```

3. Log in to the target database. The following example shows the format for logging in.

```
dblogin userid <user>@<db tnsname>
```

4. Using the ggsci command line, set up the start SCN for the REPLICAT process. The following example sets the SCN to 223274 for the REPLICAT.

```
start <replicat process name> atcsn 223274
```

Integrated REPLICAT slow due to query on sys."_DBA_APPLY_CDR_INFO"

Oracle GoldenGate Conflict Detection and Resolution (CDR) provides basic conflict resolution routines. For example, CDR can resolve a unique conflict for an INSERT statement.

When CDR resolves a collision, it can insert records into the exception table _DBA_APPLY_CDR_INFO temporarily. Integrated REPLICAT deletes these records later. In a rare scenario, the integrated REPLICAT can process a large number of collisions, but a new integrated REPLICAT does not replace it. Instead of being removed, the existing

rows in _DBA_APPLY_CDR_INFO are orphaned. Any new integrated REPLICAT processes slow down because they are querying orphaned rows in _DBA_APPLY_CDR_INFO.

To remove all rows from _DBA_APPLY_CDR_INFO, use the Amazon RDS procedure rdsadmin_util.truncate_apply$_cdr_info. This procedure is released as part of the October 2020 release and patch update. The procedure is available in the following database versions:

- Version 19.0.0.0.ru-2020-10.rur-2020-10.r1
- Version 18.0.0.0.ru-2020-10.rur-2020-10.r1
- Version 12.2.0.1.ru-2020-10.rur-2020-10.r1
- Version 12.1.0.2.v22

The following example truncates the table _DBA_APPLY_CDR_INFO.

```
SET SERVEROUTPUT ON SIZE 2000
```

```
EXEC rdsadmin.rdsadmin_util.truncate_apply
$_cdr_info;
```

Using the Oracle Repository Creation Utility on Amazon RDS for Oracle

You can use Amazon RDS to host an Oracle DB instance that holds the schemas to support your Fusion Middleware components. Before you can use Fusion Middleware components, you must create and populate schemas for them in your database. You create and populate the schemas by using the Oracle Repository Creation Utility (RCU).

You can store the schemas for any Fusion Middleware components in your Amazon RDS DB instance. The following is a list of schemas that have been verified to install correctly:

- Analytics (ACTIVITIES)
- Audit Services (IAU)
- Audit Services Append (IAU_APPEND)
- Audit Services Viewer (IAU_VIEWER)
- Discussions (DISCUSSIONS)

- Metadata Services (MDS)
- Oracle Business Intelligence (BIPLATFORM)
- Oracle Platform Security Services (OPSS)
- Portal and Services (WEBCENTER)
- Portlet Producers (PORTLET)
- Service Table (STB)
- SOA Infrastructure (SOAINFRA)
- User Messaging Service (UCSUMS)
- WebLogic Services (WLS)

Licensing and versions

Amazon RDS supports Oracle Repository Creation Utility (RCU) version 12c only. You can use the RCU in the following configurations:

- RCU 12c with Oracle database 12.2.0.1
- RCU 12c with Oracle database 12.1.0.2.v4 or later

Before you can use RCU, you need a license for Oracle Fusion Middleware. You also need to follow the Oracle licensing guidelines for the Oracle database that hosts the repository.

Fusion MiddleWare supports repositories on Oracle Database Enterprise Edition and Standard Editions (SE, SE One, or SE Two). Oracle recommends Enterprise Edition for production installations that require partitioning and installations that require online index rebuild.

Before you create your Oracle DB instance, confirm the Oracle database version that you need to support the components that you want to deploy. You can use the Certification Matrix to find the requirements for the Fusion Middleware components and versions you want to deploy.

Amazon RDS supports Oracle database version upgrades as needed.

Before you begin

Before you begin, you need an Amazon VPC. Because your Amazon RDS DB instance needs to be available only to your Fusion Middleware components, and not to the public Internet, your Amazon RDS DB instance is hosted in a private subnet, providing greater security.

Recommendations

The following are some recommendations for working with your DB instance in this scenario:

- We recommend that you use Multi-AZ for production workloads.
- For additional security, Oracle recommends that you use Transparent Data Encryption (TDE) to encrypt your data at rest. If you have

an Enterprise Edition license that includes the Advanced Security Option, you can enable encryption at rest by using the TDE option.

Amazon RDS also provides an encryption at rest option for all database editions.

- Configure your VPC Security Groups to allow communication between your application servers and your Amazon RDS DB instance. The application servers that host the Fusion Middleware components can be on Amazon EC2 or on-premises.

Using the Oracle Repository Creation Utility

You use the Oracle Repository Creation Utility (RCU) to create and populate the schemas to support your Fusion Middleware components.

Running RCU using the command line in one step

If you don't need to edit any of your schemas before populating them, you can run RCU in a single step. Otherwise, see the following section for running RCU in multiple steps.

You can run the RCU in silent mode by using the command-line parameter `-silent`. When you run RCU in silent mode, you can avoid typing passwords on the command line by creating a text file containing the passwords. Create a text file with the password for `dbUser` on the first line, and the password for each component on subsequent lines. You specify the name of the password file as the last parameter to the RCU command.

Example

The following example creates and populates schemas for the SOA Infrastructure component (and its dependencies) in a single step.

For Linux, macOS, or Unix:

```
export ORACLE_HOME=/u01/app/oracle/prod-
uct/12.2.1.0/fmw
export JAVA_HOME=/usr/java/jdk1.8.0_65
${ORACLE_HOME}/oracle_common/bin/rcu \
-silent \
-createRepository \
-connectString ${dbhost}:${dbport}:${dbname} \
-dbUser ${dbuser} \
-dbRole Normal \
-honorOMF \
-schemaPrefix ${SCHEMA_PREFIX} \
-component MDS \
-component STB \
-component OPSS \
-component IAU \
-component IAU_APPEND \
-component IAU_VIEWER \
-component UCSUMS \
-component WLS \
-component SOAINFRA \
```

```
-f < /tmp/passwordfile.txt
```

Running RCU using the command line in multiple steps

If you need to manually edit your schema scripts, you can run the RCU in multiple steps:

1. Run RCU in **Prepare Scripts for System Load** mode by using the `-generateScript` command-line parameter to create the scripts for your schemas.
2. Manually edit and run the generated script `script_systemLoad.sql`.
3. Run RCU again in **Perform Product Load** mode by using the `-dataLoad` command-line parameter to populate the schemas.
4. Run the generated clean-up script `script_postDataLoad.sql`.

You can run the RCU in silent mode by using the command-line parameter `-silent`. When you run

RCU in silent mode, you can avoid typing passwords on the command line by creating a text file containing the passwords. Create a text file with the password for dbUser on the first line, and the password for each component on subsequent lines. You specify the name of the password file as the last parameter to the RCU command.

Example

The following example creates schema scripts for the SOA Infrastructure component (and its dependencies).

For Linux, macOS, or Unix:

```
export ORACLE_HOME=/u01/app/oracle/prod-
uct/12.2.1.0/fmw
export JAVA_HOME=/usr/java/jdk1.8.0_65
${ORACLE_HOME}/oracle_common/bin/rcu \
-silent \
-generateScript \
```

```
-connectString ${dbhost}:${dbport}:${dbname} \
-dbUser ${dbuser} \
-dbRole Normal \
-honorOMF \
[-encryptTablespace true] \
-schemaPrefix ${SCHEMA_PREFIX} \
-component MDS \
-component STB \
-component OPSS \
-component IAU \
-component IAU_APPEND \
-component IAU_VIEWER \
-component UCSUMS \
-component WLS \
-component SOAINFRA \
-scriptLocation /tmp/rcuscripts \
-f < /tmp/passwordfile.txt
```

Now you can edit the generated script, connect to your Oracle DB instance, and run the script. The generated script is named `script_systemLoad.sql`.

The following example populates the schemas for the SOA Infrastructure component (and its dependencies).

For Linux, macOS, or Unix:

```
export JAVA_HOME=/usr/java/jdk1.8.0_65
${ORACLE_HOME}/oracle_common/bin/rcu \
-silent \
-dataLoad \
-connectString ${dbhost}:${dbport}:${dbname} \
-dbUser ${dbuser} \
-dbRole Normal \
-honorOMF \
-schemaPrefix ${SCHEMA_PREFIX} \
-component MDS \
-component STB \
-component OPSS \
-component IAU \
-component IAU_APPEND \
-component IAU_VIEWER \
```

```
-component UCSUMS \
-component WLS \
-component SOAINFRA \
-f < /tmp/passwordfile.txt
```

To finish, you connect to your Oracle DB instance, and run the clean-up script. The script is named script_postDataLoad.sql.

Running RCU in interactive mode

To use the RCU graphical user interface, you can run RCU in interactive mode. To run RCU in interactive mode, include the -interactive parameter and omit the -silent parameter.

Example

The following example starts RCU in interactive mode and pre-populates the connection information.

For Linux, macOS, or Unix:

```
export ORACLE_HOME=/u01/app/oracle/prod-
uct/12.2.1.0/fmw
export JAVA_HOME=/usr/java/jdk1.8.0_65
${ORACLE_HOME}/oracle_common/bin/rcu \
-interactive \
-createRepository \
-connectString ${dbhost}:${dbport}:${dbname} \
-dbUser ${dbuser} \
-dbRole Normal
```

Known issues

The following are some known issues for working with RCU, with some troubleshooting suggestions:

- Oracle Managed Files (OMF) — Amazon RDS uses OMF data files to simplify storage management. You can customize tablespace attributes, such as size and extent management. However, specifying a data file name

when you run RCU causes tablespace code to fail with ORA-20900. The RCU can be used with OMF in the following ways:

    - In RCU 12.2.1.0 and later, use the -honorOMF command-line parameter.
    - In RCU 12.1.0.3 and later, use multiple steps and edit the generated script.

- SYSDBA — Because Amazon RDS is a managed service, you don't have full SYSDBA access to your Oracle DB instance. However, RCU 12c supports users with lower privileges. In most cases, the master user privilege is sufficient to create repositories. In some cases, the RCU might fail with ORA-01031 when attempting to grant SYS object privileges. You can retry and run the RDSADMIN_UTIL.GRANT_SYS_OBJECT() stored procedure, or contact AWS Support.
- Dropping Enterprise Scheduler Service — When you use the RCU to drop an Enterprise

Scheduler Service repository, the RCU might fail with Error: Component drop check failed.

Oracle database engine release notes

Updates to your Amazon RDS for Oracle DB instances keep them current. If you apply updates, you can be confident that your DB instance is running a version of the database software that has been tested by both Oracle and Amazon. We don't support applying one-off patches to individual DB instances.

You can specify any currently supported Oracle version when creating a new DB instance. You can specify the major version (such as Oracle 12.1), and any supported minor version for the specified major version. If no version is specified, Amazon RDS defaults to a supported version, typically the most recent version. If a major version is specified but a minor version is not, Amazon RDS defaults to a recent release of the major version that you have specified. To see a list of supported versions and

defaults for newly created DB instances, use the describe-db-engine-versions AWS CLI command.

Oracle versions 19.0.0, 18.0.0, and 12.2.0.1

For Amazon RDS for Oracle versions 19.0.0.0, 18.0.0.0, and 12.2.0.1, Amazon RDS incorporates bug fixes from Oracle by using Release Updates (RUs) and Release Updates Revisions (RURs). We don't support applying one-off patches to individual DB instances.

To find what RUs and RURs are applied to Amazon RDS for Oracle versions 19.0.0, 18.0.0.0, and 12.2.0.1, see the following table.

RU and RUR	Version 19.0.0.0	Version 18.0.0.0	Version 12.2.0.1
2021 January	Version 19.0.0.0.ru-2021-01.rur-2021-01.r2 and Version 19.0.0.0.ru-2021-01.rur-2021-01.r1	Version 18.0.0.0.ru-2021-01.rur-2021-01.r1	Version 12.2.0.1.ru-2021-01.rur-2021-01.r1
2020 October	19.0.0.0.ru-2020-10.rur-2020-10.r1	18.0.0.0.ru-2020-10.rur-2020-10.r1	12.2.0.1.ru-2020-10.rur-2020-10.r1
2020 July	19.0.0.0.ru-2020-07.rur-2020-07.r1	18.0.0.0.ru-2020-07.rur-2020-07.r1	12.2.0.1.ru-2020-07.rur-2020-07.r1
2020 April	19.0.0.0.ru-2020-04.rur-2020-04.r1	18.0.0.0.ru-2020-04.rur-2020-04.r1	12.2.0.1.ru-2020-04.rur-2020-04.r1
2020 January	19.0.0.0.ru-2020-01.rur-2020-01.r1	18.0.0.0.ru-2020-01.rur-2020-01.r1	12.2.0.1.ru-2020-01.rur-2020-01.r1
2019 October	19.0.0.0.ru-2019-10.rur-2019-10.r1	18.0.0.0.ru-2019-10.rur-2019-10.r1	12.2.0.1.ru-2019-10.rur-2019-10.r1
2019 July	19.0.0.0.ru-2019-07.rur-2019-07.r1	18.0.0.0.ru-2019-07.rur-2019-07.r1	12.2.0.1.ru-2019-07.rur-2019-07.r1
2019 April	—	—	12.2.0.1.ru-2019-04.rur-2019-04.r1
2019 January	—	—	12.2.0.1.ru-2019-01.rur-2019-01.r1
2018 October	—	—	12.2.0.1.ru-2018-10.rur-2018-10.r1

Oracle versions 12.1.0.2 and 11.2.0.4

For Amazon RDS for Oracle versions 12.1.0.2 and 11.2.0.4, Amazon RDS incorporates bug fixes from Oracle by using their quarterly Database Patch Set Updates (PSUs). If you apply updates, you can be confident that your DB instance is running a version of the database software that has been tested by both Oracle and Amazon. We don't support applying one-off patches to individual DB instances.

To find what Oracle Patch Set Updates (PSUs) are applied to Amazon RDS for Oracle versions 12.1.0.2 and 11.2.0.4, see the following table.

PSU	Version 12.1.0.2	Version 11.2.0.4
2021 January	12.1.0.2.v23	N/A
2020 October	12.1.0.2.v22	11.2.0.4.v26

2020 July	12.1.0.2.v21	11.2.0.4.v25
2020 April	12.1.0.2.v20	11.2.0.4.v24
2020 January	12.1.0.2.v19	11.2.0.4.v23
2019 October	12.1.0.2.v18	11.2.0.4.v22
2019 July	12.1.0.2.v17	11.2.0.4.v21
2019 April	12.1.0.2.v16	11.2.0.4.v20
2019 January	12.1.0.2.v15	11.2.0.4.v19
2018 October	12.1.0.2.v14	11.2.0.4.v18
2018 July	12.1.0.2.v13	11.2.0.4.v17
2018 April	12.1.0.2.v12	11.2.0.4.v16
2018 January	12.1.0.2.v11	11.2.0.4.v15
2017 October	12.1.0.2.v10	11.2.0.4.v14
2017 July	12.1.0.2.v9	11.2.0.4.v13
2017 April	12.1.0.2.v8	11.2.0.4.v12
2017 January	12.1.0.2.v7	11.2.0.4.v11
2016 October	12.1.0.2.v6	11.2.0.4.v10

2016 July	12.1.0.2.v5	11.2.0.4.v9
2016 April	12.1.0.2.v4	11.2.0.4.v8
2016 January	12.1.0.2.v3	11.2.0.4.v7
2015 October	12.1.0.2.v2	11.2.0.4.v6 11.2.0.4.v5
2015 April	12.1.0.2.v1	11.2.0.4.v4
2014 October	—	11.2.0.4.v3
2014 July	—	11.2.0.4.v2 (Deprecated)
2014 January	—	11.2.0.4.v1

Security in Amazon RDS

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

Security is a shared responsibility between AWS 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 Amazon RDS. The following topics show you how to configure Amazon RDS to meet your security and compliance objectives. You also learn how to use other AWS services that help you monitor and secure your Amazon RDS resources.

You can manage access to your Amazon RDS resources and your databases on a DB instance. The method you use to manage access depends on what type of task the user needs to perform with Amazon RDS:

- Run your DB instance in a virtual private cloud (VPC) based on the Amazon VPC service for the greatest possible network access control.
- Use AWS Identity and Access Management (IAM) policies to assign permissions that determine who is allowed to manage Amazon RDS resources. For example, you can use IAM to determine who is allowed to create, describe, modify, and delete DB instances, tag resources, or modify security groups.
- Use security groups to control what IP addresses or Amazon EC2 instances can connect to your databases on a DB instance. When you first create a DB instance, its firewall prevents any database access except through rules specified by an associated security group.
- Use Secure Socket Layer (SSL) or Transport Layer Security (TLS) connections with DB

instances running the MySQL, MariaDB, PostgreSQL, Oracle, or Microsoft SQL Server database engines.

- Use Amazon RDS encryption to secure your DB instances and snapshots at rest. Amazon RDS encryption uses the industry standard AES-256 encryption algorithm to encrypt your data on the server that hosts your DB instance.
- Use network encryption and transparent data encryption with Oracle DB instances.
- Use the security features of your DB 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.

For more information on managing access to Amazon RDS resources and your databases on a DB instance, see the following topics.

Database authentication with Amazon RDS

Amazon RDS supports several ways to authenticate database users.

Password, Kerberos, and IAM database authentication use different methods of authenticating to the database. Therefore, a specific user can log in to a database using only one authentication method.

For PostgreSQL, use only one of the following role settings for a user of a specific database:

- To use IAM database authentication, assign the `rds_iam` role to the user.
- To use Kerberos authentication, assign the `rds_ad` role to the user.
- To use password authentication, don't assign either the `rds_iam` or `rds_ad` roles to the user.

Don't assign both the `rds_iam` and `rds_ad` roles to a user of a PostgreSQL database either directly or indirectly by nested grant access. If the `rds_iam` role is

added to the master user, IAM authentication takes precedence over password authentication so the master user has to log in as an IAM user.

Password authentication

With *password authentication,* your DB instance performs all administration of user accounts. You create users with SQL statements such as `CREATE USER` and specify passwords in the `IDENTIFIED BY` clause.

All RDS DB engines support password authentication. For more information about password authentication, see the documentation for your DB engine.

With password authentication, your database controls and authenticates user accounts. If a DB engine has strong password management features, they can enhance security. Database authentication might be easier to administer using password

authentication when you have small user communities. Because clear text passwords are generated in this case, integrating with AWS Secrets Manager can enhance security.

IAM database authentication

You can authenticate to your DB instance using AWS Identity and Access Management (IAM) database authentication. IAM database authentication works with MySQL and PostgreSQL. With this authentication method, you don't need to use a password when you connect to a DB instance. Instead, you use an authentication token.

Kerberos authentication

Amazon RDS supports external authentication of database users using Kerberos and Microsoft Active Directory. Kerberos is a network authenti-

cation protocol that uses tickets and symmetric-key cryptography to eliminate the need to transmit passwords over the network. Kerberos has been built into Active Directory and is designed to authenticate users to network resources, such as databases.

Amazon RDS support for Kerberos and Active Directory provides the benefits of single sign-on and centralized authentication of database users. You can keep your user credentials in Active Directory. Active Directory provides a centralized place for storing and managing credentials for multiple DB instances.

You can enable your database users to authenticate against DB instances in two ways. They can use credentials stored either in AWS Directory Service for Microsoft Active Directory or in your on-premises Active Directory.

Microsoft SQL Server, MySQL, and PostgreSQL DB instances support one- and two-way forest trust relationships. Oracle DB instances support one- and two-way external and forest trust relationships.

Data protection in Amazon RDS

The AWS shared responsibility model applies to data protection in Amazon Relational Database Service. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with 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 recommend TLS 1.2 or later.
- Set up API and user activity logging with AWS CloudTrail.
- 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 personal data that is stored in Amazon S3.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint.

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields

such as a **Name** field. This includes when you work with Amazon RDS or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into Amazon RDS or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

Identity and access management in Amazon RDS

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 Amazon RDS resources. IAM is an AWS service that you can use with no additional charge.

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work you do in A-mazon RDS.

Service user – If you use the Amazon RDS service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Amazon RDS features to do your work, you might need additional permis-sions. Understanding how access is managed can help you request the right permissions from your administrator.

Service administrator – If you're in charge of Amazon RDS resources at your company, you probably have full access to Amazon RDS. It's your job to determine which Amazon RDS features and resources your employees should access. You must then submit requests to your IAM administrator to change the permissions of your service users.

Review the information on this page to understand the basic concepts of IAM.

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to Amazon RDS.

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials.

You must be *authenticated* (signed in to AWS) as the AWS account root user, an IAM user, or by assuming an IAM role. You can also use your company's single sign-on authentication, or even sign in using Google or Facebook. In these cases, your administrator previously set up identity federation using IAM roles. When you access AWS using credentials from another company, you are assuming a role indirectly.

To sign in directly to the AWS Management Console, use your password with your root user email or your IAM user name. You can access AWS programmatically using your root user or IAM user access keys. AWS provides SDK and command line tools to cryptographically sign your request using your credentials. If you don't use AWS tools, you must sign the request yourself. Do this using *Signature Version 4*, a protocol for authenticating inbound API requests.

Regardless of the authentication method that you use, you might also be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account.

AWS account root user

When you first create an AWS account, you begin with a single sign-in identity that has complete

key in the future. Instead, you must generate a new access key pair.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials.

You can authenticate to your DB instance using IAM database authentication.

IAM database authentication works with the following DB engines:

- Amazon RDS for MySQL
- Amazon RDS for PostgreSQL

IAM roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL.

IAM roles with temporary credentials are useful in the following situations:

- **Temporary IAM user permissions** – An IAM user can assume an IAM role to temporarily

take on different permissions for a specific task.

- **Federated user access** – Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an identity provider.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy).
- **AWS service access** – A service role is an IAM role that a service assumes to perform actions on your behalf. Service roles provide

access only within your account and cannot be used to grant access to services in other accounts. An IAM administrator can create, modify, and delete a service role from within IAM.

- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials.

Managing access using policies

You control access in AWS by creating policies and attaching them to IAM identities or AWS resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when an entity (root user, IAM user, or IAM role) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents.

An IAM administrator can use policies to specify who has access to AWS resources, and what actions they can perform on those resources. Every IAM entity (user or role) starts with no permissions. In other words, by default, users can do nothing, not even change their own password. To give a user permission to do something, an administrator

must attach a permissions policy to a user. Or the administrator can add the user to a group that has the intended permissions. When an administrator gives permissions to a group, all users in that group are granted those permissions.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the iam:GetRole action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, role, or group. These policies control what actions that identity can perform, on which resources, and under what conditions.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies.

The following AWS managed policies, which you can attach to users in your account, are specific to Amazon RDS:

- **AmazonRDSReadOnlyAccess** – Grants read-only access to all Amazon RDS resources for the AWS account specified.
- **AmazonRDSFullAccess** – Grants full access to all Amazon RDS resources for the AWS account specified.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the Principal field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum

permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user.

- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow.

Logging and monitoring in Amazon RDS

Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon RDS and your AWS solutions. You should collect monitoring data from all of the parts of your AWS solution so that you can more easily debug a multi-point failure if one occurs. AWS provides several tools for monitoring your Amazon RDS resources and responding to potential incidents:

Amazon CloudWatch Alarms

Using Amazon CloudWatch alarms, you watch a single metric over a time period that you specify. If the metric exceeds a given threshold, a notification is sent to an Amazon SNS topic or AWS Auto Scaling policy. Cloud-Watch alarms do not invoke actions because they are in a particular state. Rather the state

must have changed and been maintained for a specified number of periods.

AWS CloudTrail Logs

CloudTrail provides a record of actions taken by a user, role, or an AWS service in Amazon RDS. CloudTrail captures all API calls for Amazon RDS as events, including calls from the console and from code calls to Amazon RDS API operations. Using the information collected by CloudTrail, you can determine the request that was made to Amazon RDS, the IP address from which the request was made, who made the request, when it was made, and additional details.

Enhanced Monitoring

Amazon RDS provides metrics in real time for the operating system (OS) that your DB instance runs on. You can view the metrics for your DB instance using the console, or consume the Enhanced Monitoring JSON output

from Amazon CloudWatch Logs in a monitoring system of your choice.

Amazon RDS Performance Insights

Performance Insights expands on existing Amazon RDS monitoring features to illustrate your database's performance and help you analyze any issues that affect it. With the Performance Insights dashboard, you can visualize the database load and filter the load by waits, SQL statements, hosts, or users.

Database Logs

You can view, download, and watch database logs using the AWS Management Console, AWS CLI, or RDS API.

Amazon RDS Recommendations

Amazon RDS provides automated recommendations for database resources. These recommendations provide best practice guidance by analyzing DB instance configuration, usage, and performance data.

Amazon RDS Event Notification

Amazon RDS uses the Amazon Simple Notification Service (Amazon SNS) to provide notification when an Amazon RDS event occurs. These notifications can be in any notification form supported by Amazon SNS for an AWS Region, such as an email, a text message, or a call to an HTTP endpoint.

AWS Trusted Advisor

Trusted Advisor draws upon best practices learned from serving hundreds of thousands of AWS customers. Trusted Advisor inspects your AWS environment and then makes recommendations when opportunities exist to save money, improve system availability and performance, or help close security gaps. All AWS customers have access to five Trusted Advisor checks. Customers with a Business or Enterprise support plan can view all Trusted Advisor checks.

Trusted Advisor has the following Amazon RDS-related checks:

- Amazon RDS Idle DB Instances
- Amazon RDS Security Group Access Risk
- Amazon RDS Backups
- Amazon RDS Multi-AZ

Compliance validation for Amazon RDS

Third-party auditors assess the security and compliance of Amazon RDS as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others.

Your compliance responsibility when using Amazon RDS is determined by the sensitivity of your data, your organization's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security and compliance quick start guides – These deployment guides discuss architec-

tural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.

- Architecting for HIPAA security and compliance whitepaper – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- AWS compliance resources – This collection of workbooks and guides that might apply to your industry and location.
- AWS Config – This AWS service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Resilience in Amazon RDS

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 Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

In addition to the AWS global infrastructure, Amazon RDS offers features to help support your data resiliency and backup needs.

Backup and restore

Amazon RDS creates and saves automated backups of your DB instance. Amazon RDS creates a storage volume snapshot of your DB instance, backing up the entire DB instance and not just individual databases.

Amazon RDS creates automated backups of your DB instance during the backup window of your DB instance. Amazon RDS saves the automated backups of your DB instance according to the backup retention period that you specify. If necessary, you can recover your database to any point in time during the backup retention period. You can also back up your DB instance manually, by manually creating a DB snapshot.

You can create a DB instance by restoring from this DB snapshot as a disaster recovery solution if the source DB instance fails.

Replication

Amazon RDS uses the MariaDB, MySQL, Oracle, and PostgreSQL DB engines' built-in replication functionality to create a special type of DB instance called a read replica from a source DB instance. Updates made to the source DB instance are asynchronously copied to the read replica. You can reduce the load on your source DB instance by routing read queries from your applications to the read replica. Using read replicas, you can elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads. You can promote a read replica to a standalone instance as a disaster recovery solution if the source DB instance fails. For some DB engines, Amazon RDS also supports other replication options.

Failover

Amazon RDS provides high availability and failover support for DB instances using Multi-AZ deployments. Amazon RDS uses several different technologies to provide failover support. Multi-AZ deployments for Oracle, PostgreSQL, MySQL, and MariaDB DB instances use Amazon's failover technology. SQL Server DB instances use SQL Server Database Mirroring (DBM).

Infrastructure security in Amazon RDS

As a managed service, Amazon RDS is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of security processes whitepaper.

You use AWS published API calls to access Amazon RDS through the network. Clients must support Transport Layer Security (TLS) 1.0. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve

Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

In addition, Amazon RDS offers features to help support infrastructure security.

Security groups

Security groups control the access that traffic has in and out of a DB instance. By default, network access is turned off to a DB instance. You can specify rules in a security group that allow access from an IP address range, port, or security group. After ingress rules are configured, the same rules apply to all DB

instances that are associated with that security group.

Public accessibility

When you launch a DB instance inside a virtual private cloud (VPC) based on the Amazon VPC service, you can turn on or off public accessibility for that instance. To designate whether the DB instance that you create has a DNS name that resolves to a public IP address, you use the *Public accessibility* parameter. By using this parameter, you can designate whether there is public access to the DB instance. You can modify a DB instance to turn on or off public accessibility by modifying the *Public accessibility* parameter.

Amazon RDS API and interface VPC endpoints (AWS PrivateLink)

You can establish a private connection between your VPC and Amazon RDS API endpoints by creating an *interface VPC endpoint*. Interface endpoints are powered by AWS PrivateLink.

AWS PrivateLink enables you to privately access Amazon RDS API operations without an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC don't need public IP addresses to communicate with Amazon RDS API endpoints to launch, modify, or terminate DB instances. Your instances also don't need public IP addresses to use any of the available RDS API operations. Traffic between your VPC and Amazon RDS doesn't leave the Amazon network.

Each interface endpoint is represented by one or more elastic network interfaces in your subnets.

Considerations for VPC endpoints

Before you set up an interface VPC endpoint for Amazon RDS API endpoints, ensure that you review Interface endpoint properties and limitations.

All RDS API operations relevant to managing Amazon RDS resources are available from your VPC using AWS PrivateLink.

VPC endpoint policies are supported for RDS API endpoints. By default, full access to RDS API operations is allowed through the endpoint.

Availability

Amazon RDS API currently supports VPC endpoints in the following AWS Regions:

- US East (Ohio)
- US East (N. Virginia)
- US West (N. California)
- US West (Oregon)
- Africa (Cape Town)
- Asia Pacific (Hong Kong)
- Asia Pacific (Mumbai)
- Asia Pacific (Seoul)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- Canada (Central)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (London)
- Europe (Paris)
- Europe (Stockholm)

- Europe (Milan)
- Middle East (Bahrain)
- South America (São Paulo)
- China (Beijing)
- China (Ningxia)
- AWS GovCloud (US-East)
- AWS GovCloud (US-West)

Creating an interface VPC endpoint

for Amazon RDS API

You can create a VPC endpoint for the Amazon RDS API using either the Amazon VPC console or the AWS Command Line Interface (AWS CLI).

Create a VPC endpoint for Amazon RDS API using the service name com.amazonaws.*region*.rds.

Excluding AWS Regions in China, if you enable private DNS for the endpoint, you can make API requests to Amazon RDS with the VPC endpoint using

its default DNS name for the AWS Region, for example `rds.us-east-1.amazonaws.com`. For the China (Beijing) and China (Ningxia) AWS Regions, you can make API requests with the VPC endpoint using `rds-api.cn-north-1.amazonaws.com.cn` and `rds-api.cn-northwest-1.amazonaws.com.cn`, respectively.

Creating a VPC endpoint policy

for Amazon RDS API

You can attach an endpoint policy to your VPC endpoint that controls access to Amazon RDS API. The policy specifies the following information:

- The principal that can perform actions.
- The actions that can be performed.
- The resources on which actions can be performed.

Example: VPC endpoint policy for Amazon RDS API actions

The following is an example of an endpoint policy for Amazon RDS API. When attached to an endpoint, this policy grants access to the listed Amazon RDS API actions for all principals on all resources.

```
{
 "Statement":[
   {
     "Principal":"*",
     "Effect":"Allow",
     "Action":[
       "rds:CreateDBInstance",
       "rds:ModifyDBInstance",
       "rds:CreateDBSnapshot"
     ],
     "Resource":"*"
   }
 ]
```

```
}
```

Example: VPC endpoint policy that denies all access from a specified AWS account

The following VPC endpoint policy denies AWS account 123456789012 all access to resources using the endpoint. The policy allows all actions from other accounts.

```
{
 "Statement": [
  {
   "Action": "*",
   "Effect": "Allow",
   "Resource": "*",
   "Principal": "*"
  },
  {
   "Action": "*",
   "Effect": "Deny",
   "Resource": "*",
```

```
  "Principal": {
   "AWS": [
    "123456789012"
   ]
  }
 ]}
```

Security best practices for Amazon RDS

Use AWS Identity and Access Management (IAM) accounts to control access to Amazon RDS API operations, especially operations that create, modify, or delete Amazon RDS resources. Such resources include DB instances, security groups, and parameter groups. Also use IAM to control actions that perform common administrative actions such as backing up and restoring DB instances.

- Create an individual IAM user for each person who manages Amazon RDS resources,

including yourself. Don't use AWS root credentials to manage Amazon RDS resources.

- Grant each user the minimum set of permissions required to perform his or her duties.
- Use IAM groups to effectively manage permissions for multiple users.
- Rotate your IAM credentials regularly.
- Configure AWS Secrets Manager to automatically rotate the secrets for Amazon RDS. You can also retrieve the credential from AWS Secrets Manager programmatically.

Use the AWS Management Console, the AWS CLI, or the RDS API to change the password for your master user. If you use another tool, such as a SQL client, to change the master user password, it might result in privileges being revoked for the user unintentionally.

Controlling access with security groups

Security groups control the access that traffic has in and out of a DB instance. Three types of security groups are used with Amazon RDS: VPC security groups, DB security groups, and EC2-Classic security groups. In simple terms, these work as follows:

- A VPC security group controls access to DB instances and EC2 instances inside a VPC.
- A DB security group controls access to EC2-Classic DB instances that are not in a VPC.
- An EC2-Classic security group controls access to an EC2 instance.

By default, network access is disabled for a DB instance. You can specify rules in a security group that allow access from an IP address range, port, or security group. Once ingress rules are configured, the same rules apply to all DB instances that are associated with that security group. You can specify up to 20 rules in a security group.

VPC security groups

Each VPC security group rule enables a specific source to access a DB instance in a VPC that is associated with that VPC security group. The source can be a range of addresses (for example, 203.0.113.0/24), or another VPC security group. By specifying a VPC security group as the source, you allow incoming traffic from all instances (typically application servers) that use the source VPC security group. VPC security groups can have rules that govern both inbound and outbound traffic, though the outbound traffic rules typically do not apply to DB instances. Outbound traffic rules only apply if the DB instance acts as a client. For example, outbound traffic rules apply to an Oracle DB instance with outbound database links. You must use the Amazon EC2 API or the **Security Group** option on the VPC Console to create VPC security groups.

When you create rules for your VPC security group that allow access to the instances in your VPC, you must specify a port for each range of addresses that the rule allows access for. For example, if you want to enable SSH access to instances in the VPC, then you create a rule allowing access to TCP port 22 for the specified range of addresses.

You can configure multiple VPC security groups that allow access to different ports for different instances in your VPC. For example, you can create a VPC security group that allows access to TCP port 80 for web servers in your VPC. You can then create another VPC security group that allows access to TCP port 3306 for RDS for MySQL DB instances in your VPC.

If your DB instance is in a VPC but isn't publicly accessible, you can also use an AWS Site-to-Site VPN connection or an AWS Direct Connect connection to access it from a private network.

DB security groups

DB security groups are used with DB instances that are not in a VPC and on the EC2-Classic platform. Each DB security group rule enables a specific source to access a DB instance that is associated with that DB security group. The source can be a range of addresses (for example, 203.0.113.0/24), or an EC2-Classic security group. When you specify an EC2-Classic security group as the source, you allow incoming traffic from all EC2 instances that use that EC2-Classic security group. DB security group rules apply to inbound traffic only; outbound traffic is not currently permitted for DB instances.

You don't need to specify a destination port number when you create DB security group rules. The port number defined for the DB instance is used as the destination port number for all rules defined

for the DB security group. DB security groups can be created using the Amazon RDS API operations or the Amazon RDS page of the AWS Management Console.

DB security groups vs. VPC security groups

The following table shows the key differences between DB security groups and VPC security groups.

DB security group	VPC security group
Controls access to DB instances outside a VPC.	Controls access to DB instances in VPC.
Uses Amazon RDS API operations or the Amazon RDS page of the AWS Management Console to create and	Uses Amazon EC2 API operations or the Amazon VPC page of the AWS Management Console to create and

manage group and rules.	manage group and rules.
When you add a rule to a group, you don't need to specify port number or protocol.	When you add a rule to a group, specify the protocol as TCP. In addition, specify the same port number that you used to create the DB instances (or options) that you plan to add as members to the group.
Groups allow access from EC2-Classic security groups in your AWS account or other accounts.	Groups allow access from other VPC security groups in your VPC only.

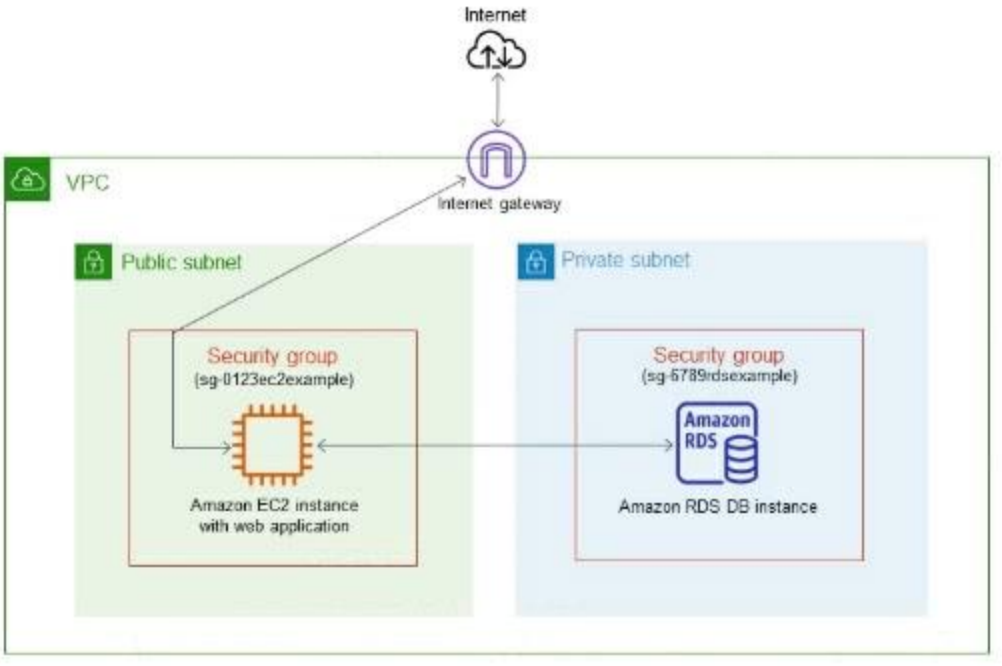
Security group scenario

A common use of a DB instance in a VPC is to share data with an application server running in an Amazon EC2 instance in the same VPC, which is accessed by a client application outside the VPC. For this scenario, you use the RDS and VPC pages on the AWS Management Console or the RDS and EC2 API operations to create the necessary instances and security groups:

1. Create a VPC security group (for example, `sg-0123ec2example`) and define inbound rules that use the IP addresses of the client application as the source. This security group allows your client application to connect to EC2 instances in a VPC that uses this security group.
2. Create an EC2 instance for the application and add the EC2 instance to the VPC security group (`sg-0123ec2example`) that you created in the previous step.

3. Create a second VPC security group (for example, sg-6789rdsexample) and create a new rule by specifying the VPC security group that you created in step 1 (sg-0123ec2example) as the source.
4. Create a new DB instance and add the DB instance to the VPC security group (sg-6789rdsexample) that you created in the previous step. When you create the DB instance, use the same port number as the one specified for the VPC security group (sg-6789rdsexample) rule that you created in step 3.

The following diagram shows this scenario.



Creating a VPC security group

You can create a VPC security group for a DB instance by using the VPC console.

Associating a security group with a DB instance

You can associate a security group with a DB instance by using **Modify** on the RDS console, the

ModifyDBInstance Amazon RDS API, or the modify-db-instance AWS CLI command.

Deleting DB VPC security groups

DB VPC security groups are an RDS mechanism to synchronize security information with a VPC security group. However, this synchronization is no longer required, because RDS has been updated to use VPC security group information directly.

DB VPC security groups are deprecated, and they are different from DB security groups, VPC security groups, and EC2-Classic security groups.

We strongly recommend that you delete any DB VPC security groups that you currently use. If you don't delete your DB VPC security groups, you might encounter unintended behaviors with your DB instances, which can be as severe as losing access to a DB instance. The unintended behaviors

might result from taking an action such as an update to a DB instance, a parameter group, or similar. Such updates cause RDS to resynchronize the DB VPC security group with the VPC security group. This resynchronization can result in your security information being overwritten with incorrect and outdated security information. This in turn can have a severe impact on your ability to access to your DB instances.

How can I determine if I have a DB VPC security group?

Because DB VPC security groups have been deprecated, they don't appear in the RDS console. However, you can call the describe-db-security-groups AWS CLI command or the DescribeDBSecurityGroups API operation to determine if you have any DB VPC security groups.

In this case, you can call the `describe-db-security-groups` AWS CLI command with JSON specified as

the output format. If you do, you can identify DB VPC security groups by the VPC identifier on the second line of the output for the security group as shown in the following example.

```
{
  "DBSecurityGroups": [
    {
      "VpcId": "vpc-abcd1234",
      "DBSecurityGroupDescription": "default:vpc-
abcd1234",
      "IPRanges": [
        {
          "Status": "authorized",
          "CIDRIP": "xxx.xxx.xxx.xxx/n"
        },
        {
          "Status": "authorized",
          "CIDRIP": "xxx.xxx.xxx.xxx/n "
        }
      ],
```

```
        "OwnerId": "123456789012",
        "EC2SecurityGroups": [],
        "DBSecurityGroupName": "default:vpc-
abcd1234"
      }
   ]
}
```

If you run the DescribeDBSecurityGroups API operation, then you can identify DB VPC security groups using the <VpcId> response element as shown in the following example.

```
<DBSecurityGroup>
 <EC2SecurityGroups/>
 <DBSecurityGroupDescription>default:vpc-
abcd1234</DBSecurityGroupDescription>
 <IPRanges>
  <IPRange>
   <CIDRIP>xxx.xxx.xxx.xxx/n</CIDRIP>
   <Status>authorized</Status>
```

```
 </IPRange>
 <IPRange>
  <CIDRIP>xxx.xxx.xxx.xxx/n</CIDRIP>
  <Status>authorized</Status>
 </IPRange>
</IPRanges>
<VpcId>vpc-abcd1234</VpcId>
<OwnerId>123456789012</OwnerId>
<DBSecurityGroupName>default:vpc-
abcd1234</DBSecurityGroupName>
</DBSecurityGroup>
```

How do I delete a DB VPC security group?

Because DB VPC security groups don't appear in the RDS console, you must call the delete-db-security-group AWS CLI command or the DeleteDBSecurityGroup API operation to delete a DB VPC security group.

After you delete a DB VPC security group, your DB instances in your VPC continue to be secured by the VPC security group for that VPC. The DB VPC security group that was deleted was merely a copy of the VPC security group information.

Review your AWS CloudFormation templates

Older versions of AWS CloudFormation templates can contain instructions to create a DB VPC security group. Because DB VPC security groups are not yet fully deprecated, they can still be created. Make sure that any AWS CloudFormation templates that you use to provision a DB instance with security settings don't also create a DB VPC security group. Don't use AWS CloudFormation templates that create an RDS DBSecurityGroup with an EC2VpcId as shown in the following example.

```
{
```

```
 "DbSecurityByEC2SecurityGroup" : {
  "Type" : "AWS::RDS::DBSecurityGroup",
  "Properties" : {
    "GroupDescription" : "Ingress for security
group",
    "EC2VpcId" : "MyVPC",
    "DBSecurityGroupIngress" : [ {
      "EC2SecurityGroupId" : "sg-b0ff1111",
      "EC2SecurityGroupOwnerId" :
"111122223333"
    }, {
      "EC2SecurityGroupId" : "sg-ffd722222",
      "EC2SecurityGroupOwnerId" :
"111122223333"
    } ]
  }
 }
}
```

Instead, add security information for your DB instances in a VPC using VPC security groups, as shown in the following example.

```
{
 "DBInstance" : {
  "Type": "AWS::RDS::DBInstance",
  "Properties": {
   "DBName"        : { "Ref" : "DBName" },
   "Engine"        : "MySQL",
   "MultiAZ"       : { "Ref": "MultiAZDatabase" },
   "MasterUsername"   : { "Ref" : "<master_user-
name>" },
   "DBInstanceClass"  : { "Ref" : "DBClass" },
   "AllocatedStorage"  : { "Ref" : "DBAllocatedStor-
age" },
   "MasterUserPassword": { "Ref" : "<master_pass-
word>" },
   "VPCSecurityGroups" : [ { "Fn::GetAtt": [ "VPCSe-
curityGroup", "GroupId" ] } ]
```

```
    }
  }
}
```

Master user account privileges

When you create a new DB instance, the default master user that you use gets certain privileges for that DB instance.

Using service-linked roles for Amazon RDS

Amazon RDS 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 Amazon RDS. Service-linked roles are predefined by Amazon RDS and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes using Amazon RDS easier because you don't have to manually add the necessary permissions. Amazon RDS defines the permissions of its service-linked roles, and unless defined otherwise, only Amazon RDS can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permis-

sions policy cannot be attached to any other IAM entity.

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

Service-linked role permissions for Amazon RDS

Amazon RDS uses the service-linked role named **AWSServiceRoleForRDS** – to allow Amazon RDS to call AWS services on behalf of your DB instances.

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

- rds.amazonaws.com

The role permissions policy allows Amazon RDS to complete the following actions on the specified resources:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:CreateNetworkInterface",
        "ec2:CreateSecurityGroup",
        "ec2:DeleteNetworkInterface",
        "ec2:DeleteSecurityGroup",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeInternetGateways",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcAttribute",
```

```
        "ec2:DescribeVpcs",
        "ec2:ModifyNetworkInterfaceAttribute",
        "ec2:ModifyVpcEndpoint",
        "ec2:RevokeSecurityGroupIngress",
        "ec2:CreateVpcEndpoint",
        "ec2:DescribeVpcEndpoints",
        "ec2:DeleteVpcEndpoints",
        "ec2:AssignPrivateIpAddresses",
        "ec2:UnassignPrivateIpAddresses"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": "*"
    },
    {
```

```
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/rds/*",
        "arn:aws:logs:*:*:log-group:/aws/docdb/*",
        "arn:aws:logs:*:*:log-group:/aws/neptune/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:*:*:log-group:/aws/rds/*:log-
stream:*",
```

```
        "arn:aws:logs:*:*:log-group:/aws/docdb/*:log-
stream:*",
        "arn:aws:logs:*:*:log-group:/aws/neptune/
*:log-stream:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "kinesis:CreateStream",
        "kinesis:PutRecord",
        "kinesis:PutRecords",
        "kinesis:DescribeStream",
        "kinesis:SplitShard",
        "kinesis:MergeShards",
        "kinesis:DeleteStream",
        "kinesis:UpdateShardCount"
      ],
      "Resource": [
        "arn:aws:kinesis:*:*:stream/aws-rds-das-*"
```

```
        ]
    }
  ]
}
```

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/
rds.amazonaws.com/AWSServiceRoleForRDS",
```

```
  "Condition": {
    "StringLike": {
      "iam:AWSServiceName":"rds.amazonaws.
com"
    }
  }
}
```

Creating a service-linked role for Amazon RDS

You don't need to manually create a service-linked role. When you create a DB instance, Amazon RDS 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 a DB instance, Amazon RDS creates the service-linked role for you again.

Editing a service-linked role for Amazon RDS

Amazon RDS does not allow you to edit the AWSServiceRoleForRDS 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.

Deleting a service-linked role for Amazon RDS

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 DB instances before you can delete the service-linked role.

Cleaning up a service-linked role

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 AWSServiceRoleForRDS 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.

If you want to remove the AWSServiceRoleForRDS role, you must first delete *all* of your DB instances .

Deleting all of your instances

Use one of these procedures to delete each of your instances.

To delete an instance (console)

1. Open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Databases**.
3. Choose the instance that you want to delete.
4. For **Actions**, choose **Delete**.
5. If you are prompted for **Create final Snapshot?**, choose **Yes** or **No**.
6. If you chose **Yes** in the previous step, for **Final snapshot name** enter the name of your final snapshot.
7. Choose **Delete**.

You can use the IAM console, the IAM CLI, or the IAM API to delete the AWSServiceRoleForRDS service-linked role.

Amazon Virtual Private Cloud

VPCs and Amazon RDS

There are two Amazon Elastic Compute Cloud (EC2) platforms that host Amazon RDS DB instances, *EC2-VPC* and *EC2-Classic*. Amazon Virtual Private Cloud (Amazon VPC) enables you to launch AWS resources, such as Amazon RDS DB instances, into a virtual private cloud (VPC).

When you use an Amazon VPC, you have control over your virtual networking environment: you can choose your own IP address range, create subnets, and configure routing and access control lists. The basic functionality of Amazon RDS is the same whether your DB instance is running in an Amazon VPC or not: Amazon RDS manages backups, software patching, automatic failure detection, and recovery. There is no additional cost to run your DB instance in an Amazon VPC.

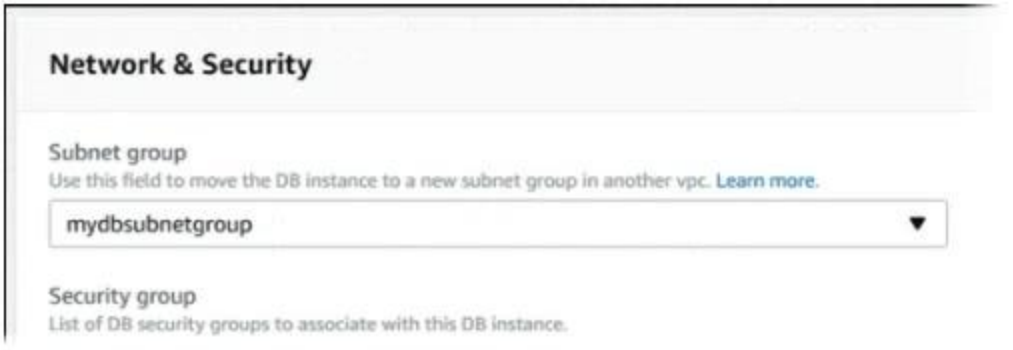
Accounts that support only the *EC2-VPC* platform have a default VPC. All new DB instances are created in the default VPC unless you specify otherwise. If you are a new Amazon RDS customer, if you have never created a DB instance before, or if you are creating a DB instance in an AWS Region you have not used before, you are most likely on the *EC2-VPC* platform and have a default VPC.

Some legacy DB instances on the *EC2-Classic* platform are not in a VPC. The legacy *EC2-Classic* platform does not have a default VPC, but as is true for either platform, you can create your own VPC and specify that a DB instance be located in that VPC.

Updating the VPC for a DB instance

You can use the AWS Management Console to move your DB instance to a different VPC.

In the **Network & Security** section of the modify page, shown following, enter the new subnet group for **Subnet group**. The new subnet group must be a subnet group in a new VPC.



Moving a DB instance not in a VPC into a VPC

Some legacy DB instances on the EC2-Classic platform are not in a VPC. If your DB instance is not in a VPC, you can use the AWS Management Console to easily move your DB instance into a VPC. Before you can move a DB instance not in a VPC, into a VPC, you must create the VPC.

Follow these steps to create a VPC for your DB instance.

- Step 1: Create a VPC
- Step 2: Add subnets to the VPC
- Step 3: Create a DB subnet group
- Step 4: Create a VPC security group

Each DB subnet group must include at least the Availability Zones in which the DB instance is located.

After you create the VPC, follow these steps to move your DB instance into the VPC.

- Updating the VPC for a DB instance

We highly recommend that you create a backup of your DB instance immediately before the migration. Doing so ensures that you can restore the data if the migration fails. For more information, see Backing up and restoring an Amazon RDS DB instance.

The following are some limitations to moving your DB instance into the VPC.

- **Previous generation DB instance classes** – Previous generation DB instance classes might not be supported on the VPC platform. When moving a DB instance to a VPC, choose a db.m3 or db.r3 DB instance class. After you move the DB instance to a VPC, you can scale the DB instance to use a later DB instance class.
- **Multi-AZ** – Moving a Multi-AZ DB instance not in a VPC into a VPC is not currently supported. To move your DB instance to a VPC, first modify the DB instance so that it is a single-AZ deployment. Change the **Multi-AZ deployment** setting to **No**. After you move the DB instance to a VPC, modify it again to make it a Multi-AZ deployment.
- **Read replicas** – Moving a DB instance with read replicas not in a VPC into a VPC is not currently supported. To move your DB instance to a VPC, first delete all of its read

replicas. After you move the DB instance to a VPC, recreate the read replicas.

- **Option groups** – If you move your DB instance to a VPC, and the DB instance is using a custom option group, change the option group that is associated with your DB instance. Option groups are platform-specific, and moving to a VPC is a change in platform. To use a custom option group in this case, assign the default VPC option group to the DB instance, assign an option group that is used by other DB instances in the VPC you are moving to, or create a new option group and assign it to the DB instance.

Alternatives for moving a DB instance not in a VPC into a VPC with minimal downtime

Using the following alternatives, you can move a DB instance not in a VPC into a VPC with minimal downtime. These alternatives cause minimum

disruption to the source DB instance and allow it to serve user traffic during the migration. However, the time required to migrate to a VPC will vary based on the database size and the live workload characteristics.

- **AWS Database Migration Service (AWS DMS)** – AWS DMS enables the live migration of data while keeping the source DB instance fully operational, but it replicates only a limited set of DDL statements. AWS DMS doesn't propagate items such as indexes, users, privileges, stored procedures, and other database changes not directly related to table data. In addition, AWS DMS doesn't automatically use RDS snapshots for the initial DB instance creation, which can increase migration time.
- **DB snapshot restore or point-in-time recovery** – You can move a DB instance to a VPC by restoring a snapshot of the DB instance or by restoring a DB instance to a point in time.

Troubleshooting for Amazon RDS

Use the following sections to help troubleshoot problems you have with DB instances in Amazon RDS and Aurora.

Can't connect to Amazon RDS DB instance

When you can't connect to a DB instance, the following are common causes:

- **Inbound rules** – The access rules enforced by your local firewall and the IP addresses authorized to access your DB instance might not match. The problem is most likely the inbound rules in your security group.

By default, DB instances don't allow access. Access is granted through a security group associated with the VPC that allows traffic into and out of the DB instance. If necessary, add inbound and outbound rules for your particular situation to the security

group. You can specify an IP address, a range of IP addresses, or another VPC security group.

Client connections from IP addresses within the range 169.254.0.0/16 aren't permitted. This is the Automatic Private IP Addressing Range (APIPA), which is used for local-link addressing.

- **Public accessibility** – To connect to your DB instance from outside of the VPC, such as by using a client application, the instance must have a public IP address assigned to it.

To make the instance publicly accessible, modify it and choose **Yes** under **Public accessibility**.

- **Port** – The port that you specified when you created the DB instance can't be used to send or receive communications due to your local firewall restrictions. To determine if your network allows the specified port to be used for inbound and outbound communication, check with your network administrator.

- **Availability** – For a newly created DB instance, the DB instance has a status of creating until the DB instance is ready to use. When the state changes to available, you can connect to the DB instance. Depending on the size of your DB instance, it can take up to 20 minutes before an instance is available.
- **Internet gateway** – For a DB instance to be publicly accessible, the subnets in its DB subnet group must have an internet gateway.

To configure an internet gateway for a subnet

1. Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
2. In the navigation pane, choose **Databases**, and then choose the name of the DB instance.
3. In the **Connectivity & security** tab, write down the values of the VPC ID under **VPC** and the subnet ID under **Subnets**.

4. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
5. In the navigation pane, choose **Internet Gateways**. Verify that there is an internet gateway attached to your VPC. Otherwise, choose **Create Internet Gateway** to create an internet gateway. Select the internet gateway, and then choose **Attach to VPC** and follow the directions to attach it to your VPC.
6. In the navigation pane, choose **Subnets**, and then select your subnet.
7. On the **Route Table** tab, verify that there is a route with `0.0.0.0/0` as the destination and the internet gateway for your VPC as the target.
    a. Choose the ID of the route table (rtb-*xxxxxxxx*) to navigate to the route table.
    b. On the **Routes** tab, choose **Edit routes**. Choose **Add route**, use `0.0.0.0/0` as the

destination and the internet gateway as the target.

c. Choose **Save routes**.

Testing a connection to a DB instance

You can test your connection to a DB instance using common Linux or Microsoft Windows tools.

From a Linux or Unix terminal, you can test the connection by entering the following (replace *DB-instance-endpoint* with the endpoint and *port* with the port of your DB instance).

```
nc -zv DB-instance-endpoint port
```

For example, the following shows a sample command and the return value.

```
nc -zv postgresql1.c6c8mn7fake0.us-
west-2.rds.amazonaws.com 8299
```

```
 Connection to postgresql1.c6c8mn7fake0.us-
west-2.rds.amazonaws.com 8299 port [tcp/vvr-
data] succeeded!
```

Windows users can use Telnet to test the connection to a DB instance. Telnet actions aren't supported other than for testing the connection. If a connection is successful, the action returns no message. If a connection isn't successful, you receive an error message such as the following.

```
C:\>telnet sg-postgresql1.c6c8mntfake0.us-
west-2.rds.amazonaws.com 819

 Connecting To sg-postgresql1.c6c8mntfake0.us-
west-2.rds.amazonaws.com...Could not open
 connection to the host, on port 819: Connect failed
```

If Telnet actions return success, your security group is properly configured.

Troubleshooting connection authentication

If you can connect to your DB instance but you get authentication errors, you might want to reset the master user password for the DB instance. You can do this by modifying the RDS instance.

Amazon RDS security issues

To avoid security issues, never use your master AWS user name and password for a user account. Best practice is to use your master AWS account to create AWS Identity and Access Management (IAM) users and assign those to DB user accounts. You can also use your master account to create other user accounts, if necessary.

Error message "failed to retrieve account attributes, certain console functions may be impaired."

You can get this error for several reasons. It might be because your account is missing permissions, or your account hasn't been properly set up. If your account is new, you might not have waited for the account to be ready. If this is an existing account, you might lack permissions in your access policies to perform certain actions such as creating a DB instance. To fix the issue, your IAM administrator needs to provide the necessary roles to your account.

Resetting the DB instance owner password

If you get locked out of your DB instance, you can log in as the master user. Then you can reset the credentials for other administrative users or roles. If you can't log in as the master user, the AWS account owner can reset the master user password.

Amazon RDS DB instance outage or reboot

A DB instance outage can occur when a DB instance is rebooted. It can also occur when the DB instance is put into a state that prevents access to it, and when the database is restarted. A reboot can occur when you either manually reboot your DB instance or change a DB instance setting that requires a reboot before it can take effect.

A DB instance reboot occurs when you change a setting that requires a reboot, or when you manually cause a reboot. A reboot can occur immediately if you change a setting and request that the change take effect immediately or it can occur during the DB instance's maintenance window.

A DB instance reboot occurs immediately when one of the following occurs:

- You change the backup retention period for a DB instance from 0 to a nonzero value or from a nonzero value to 0 and set **Apply Immediately** to true.
- You change the DB instance class, and **Apply Immediately** is set to true.
- You change the storage type from **Magnetic (Standard)** to **General Purpose (SSD)** or **Provisioned IOPS (SSD)**, or from **Provisioned IOPS (SSD)** or **General Purpose (SSD)** to **Magnetic (Standard)**.

A DB instance reboot occurs during the maintenance window when one of the following occurs:

- You change the backup retention period for a DB instance from 0 to a nonzero value or from a nonzero value to 0, and **Apply Immediately** is set to false.
- You change the DB instance class, and **Apply Immediately** is set to false.

When you change a static parameter in a DB parameter group, the change doesn't take effect until the DB instance associated with the parameter group is rebooted. The change requires a manual reboot. The DB instance isn't automatically rebooted during the maintenance window.

Amazon RDS DB parameter

changes not taking effect

In some cases, you might change a parameter in a DB parameter group but don't see the changes take effect. If so, you likely need to reboot the DB instance associated with the DB parameter group. When you change a dynamic parameter, the change takes effect immediately. When you change a static parameter, the change doesn't take effect until you reboot the DB instance associated with the parameter group.

You can reboot a DB instance using the RDS console or explicitly calling the RebootDBInstance API operation (without failover, if the DB instance is in a Multi-AZ deployment). The requirement to reboot the associated DB instance after a static parameter change helps mitigate the risk of a parameter misconfiguration affecting an API call. An example of this might be calling ModifyDBInstance to change the DB instance class.

Amazon RDS DB instance running out of storage

If your DB instance runs out of storage space, it might no longer be available. We highly recommend that you constantly monitor the FreeStorageSpace metric published in CloudWatch to make sure that your DB instance has enough free storage space.

If your database instance runs out of storage, its status changes to storage-full. For example, a call to the DescribeDBInstances API operation for a DB instance that has used up its storage outputs the following.

```
aws rds describe-db-instances --db-instance-iden-
tifier mydbinstance

DBINSTANCE mydbinstance
2009-12-22T23:06:11.915Z db.m5.large mysql8.0
50 sa
storage-full mydbinstance.clla4j4jgyph.us-
east-1.rds.amazonaws.com 3306
us-east-1b 3
        SECGROUP default active
        PARAMGRP default.mysql8.0 in-sync
```

To recover from this scenario, add more storage space to your instance using the ModifyDBIn-

stance API operation or the following AWS CLI command.

For Linux, macOS, or Unix:

```
aws rds modify-db-instance \
  --db-instance-identifier mydbinstance \
  --allocated-storage 60 \
  --apply-immediately
```

For Windows:

```
aws rds modify-db-instance ^
  --db-instance-identifier mydbinstance ^
  --allocated-storage 60 ^
  --apply-immediately
DBINSTANCE mydbinstance
2009-12-22T23:06:11.915Z db.m5.large mysql8.0
50 sa
storage-full mydbinstance.clla4j4jgyph.us-
east-1.rds.amazonaws.com 3306
us-east-1b 3 60
```

```
        SECGROUP default active
        PARAMGRP default.mysql8.0 in-sync
```

Now, when you describe your DB instance, you see that your DB instance has `modifying` status, which indicates the storage is being scaled.

```
aws rds describe-db-instances --db-instance-iden-
tifier mydbinstance
DBINSTANCE mydbinstance
2009-12-22T23:06:11.915Z db.m5.large mysql8.0
50 sa
modifying mydbinstance.clla4j4jgyph.us-
east-1.rds.amazonaws.com
3306 us-east-1b 3 60
        SECGROUP default active
        PARAMGRP default.mysql8.0 in-sync
```

After storage scaling is complete, your DB instance status changes to `available`.

```
aws rds describe-db-instances --db-instance-iden-
tifier mydbinstance
DBINSTANCE mydbinstance
2009-12-22T23:06:11.915Z db.m5.large mysql8.0
60 sa
available mydbinstance.clla4j4jgyph.us-
east-1.rds.amazonaws.com 3306
us-east-1b 3
        SECGROUP default active
        PARAMGRP default.mysql8.0 in-sync
```

You can receive notifications when your storage space is exhausted using the DescribeEvents operation. For example, in this scenario, if you make a DescribeEvents call after these operations you see the following output.

```
aws rds describe-events --source-type db-instance --
source-identifier mydbinstance
2009-12-22T23:44:14.374Z mydbinstance Allo-
cated storage has been exhausted db-instance
```

```
2009-12-23T00:14:02.737Z mydbinstance Applying modification to allocated storage db-instance
2009-12-23T00:31:54.764Z mydbinstance Finished applying modification to allocated storage
```

Amazon RDS insufficient DB instance capacity

The InsufficientDBInstanceCapacity error can be returned when you try to create or modify a DB instance, or when you try to restore a DB instance from a DB snapshot. When this error is returned, the following are common causes:

- The specific DB instance class isn't available in the requested Availability Zone. You can try one of the following to solve the problem:
  - Retry the request with a different DB instance class.
  - Retry the request with a different Availability Zone.

    - Retry the request without specifying an explicit Availability Zone.
- The DB instance is on the EC2-Classic platform and therefore isn't in a VPC. Some DB instance classes require a VPC. For example, if you're on the EC2-Classic platform and try to increase capacity by switching to a DB instance class that requires a VPC, this error results. To correct the problem, you can move the DB instance into a VPC.

Best Practices for Running Oracle

Database on AWS

Amazon Web Services (AWS) offers you the ability to run your Oracle Database in a cloud environment. Running Oracle Database in the AWS Cloud is very similar to running Oracle Database in your data center. To a database administrator or developer, there are no differences between the two environments. However, there are a number of AWS platform considerations relating to security, storage, compute configurations, management, and monitoring that will help you get the best out of your Oracle Database implementation on AWS.

This whitepaper provides best practices for achieving optimal performance, availability, and reliability, and lowering the total cost of ownership (TCO) while running Oracle Database on AWS. The target audience for this whitepaper includes data-

base administrators, enterprise architects, systems administrators, and developers who would like to run their Oracle Database on AWS.

Introduction

Amazon Web Services (AWS) provides a comprehensive set of services and tools for deploying Oracle Database on the reliable and secure AWS Cloud infrastructure. AWS offers its customers three options for running Oracle Database on AWS:

1. Using Amazon Relational Database Service (Amazon RDS) for Oracle, which is a managed database service that helps simplify the provisioning and management of Oracle databases. Amazon RDS makes it easy to set up, operate, and scale a relational database in the cloud by automating installation, disk provisioning and management, patching,

minor version upgrades, failed instance replacement, as well as backup and recovery tasks. The push-button scaling feature of Amazon RDS allows you to easily scale the database instance up or down for better cost management and performance. Amazon RDS offers both Oracle Enterprise Edition and Standard Editions. Amazon RDS also comes with a License-Included service model, which allows you to pay per use by the hour.

2. Running a self-managed Oracle Database directly on Amazon Elastic Compute Cloud (Amazon EC2). This option gives you full control over the setup of the infrastructure and database environment. Running the database on Amazon EC2 is very similar to running the database on your own server. You have full control of the database and have operating system-level access, so you can run monitoring and management agents

and use your choice of tools for data replication, backup, and restoration. Furthermore, you have the ability to use every optional module available in Oracle Database. However, this option requires you to set up, configure, manage, and tune all the components, including Amazon EC2 instances, storage volumes, scalability, networking, and security, based on AWS architecture best practices. In the fully managed Amazon RDS service, this is all taken care of for you.

3. Running a self-managed Oracle Database directly on VMware Cloud on AWS. VMware Cloud on AWS is an integrated cloud offering jointly developed by AWS and VMware. Like Amazon EC2, you have full control of the database and have operating system-level access. You can run advanced architectures like Oracle Real Application Cluster (RAC), Oracle

RAC extended clusters (across different AZs) in VMware Cloud on AWS.

Whether you choose to run a self-managed Oracle Database on Amazon EC2 or the fully managed Amazon RDS for Oracle, following the best practices discussed in this whitepaper will help you get the most out of your Oracle Database implementation on AWS. We'll discuss Oracle licensing options, considerations for choosing Amazon EC2 or Amazon RDS for your Oracle Database implementation, and how to optimize network configuration, instance type, and database storage in your implementation.

Oracle Licensing Considerations

Oracle Database licensing on AWS is based on the size of the instance on which the database is installed. For information about Oracle Database licensing, please refer to your Oracle contract on

license terms. You can consult with independent third-party license review firms on specific licensing questions and planning on AWS instances. Reach out to your AWS sales representative for more information. A few key points to consider are:

- As stated in Amazon EC2 Instance Types page, each vCPU is a thread of either an Intel Xeon core or an AMD EPYC core, except for A1 instances, T2 instances, and m3.medium.
- AWS offers optimize CPU feature in RDS and EC2 instances. You can specify the following CPU options to optimize your instances for specific workload or business needs.
- **Number of CPU cores**: You can customize the number of CPU cores for the instance.
- **Threads per core**: You can disable multi-threading by specifying a single thread per CPU core.
- VMware Cloud on AWS also offers custom CPU core count feature for its host nodes. You

have an option of selecting 8, 16, or 36 CPU cores per host for I3, or selecting 8, 16 or 48 CPU cores for R5 host type.

- Any discussion of Oracle licensing policies and costs in this whitepaper is for informational purposes only and is based on the information available at the time of publication. For more specific information, users should consult their own Oracle license agreements.

Amazon RDS License Included

You have the option to include the cost of the Oracle Database license in the hourly price of the Amazon RDS service if you use the License Included service model. In this case, you do not need to purchase Oracle licenses separately; the Oracle Database software has been licensed by AWS. License Included per-hour pricing includes software, underlying hardware resources, and Amazon RDS management

capabilities. This service model optimizes license costs, and gives you flexibility when scaling your Amazon RDS instances up or down. You can take advantage of hourly pricing with no upfront fees or long-term commitments. In addition, you can purchase Amazon RDS Reserved Instances under one-year or three-year reservation terms. With Reserved Instances, you can make a low, one-time payment up front for each database instance, and then pay a significantly discounted hourly usage rate.

The hourly license for the License Included model in Amazon RDS is available only for Oracle Standard Edition One and Standard Edition Two. For other editions of Oracle Database on Amazon RDS and any edition of Oracle Database on Amazon EC2, you need to use your own license (that is, acquire a license from Oracle), as discussed in the following section.

Since you are paying for the Oracle license only for the hours in which you use Amazon RDS, the License Included option may help you reduce overall licensing costs for development and testing environments that are active only during business hours. For most businesses, the total business hours per week (10 x 5 = 50 hours) is only about 30% of the total hours in a week (24 x 7 = 168 hours), so this service model could result in considerable savings.

This service model also gives you the flexibility to resize the instance based on your needs, because the license is included in the instance cost. In cases where your regular capacity requirements are much smaller than periodic, predictable spikes, this service model allows you to scale up to absorb the additional capacity needed, and scale down to save on cost. For example, you might have databases that require the performance of a db.m3.large in-

stance for most days of the month except for the last three days. During the last three days of the month, your database might be heavily used due to payroll processing and month-end closing. In this scenario, you can use Oracle Database on Amazon RDS based on the db.m3.large instance type throughout the month, scale up to db.m3.2xlarge for the last three days, and then scale down again. This could translate to 65% or more cost savings compared to using the db.m3.2xlarge instance for the whole month.

Bring Your Own License (BYOL)

If you already own Oracle Database licenses, you can use the BYOL service model to run your Oracle databases on Amazon RDS. This will result in a lower cost for the Amazon RDS instance because the cost of the Oracle license isn't included. The BYOL model is designed for customers who prefer

to use their existing Oracle Database licenses or purchase new licenses directly from Oracle.

If you want to use Oracle Database Enterprise Edition with Amazon RDS, or run your own self-managed Oracle Database on Amazon EC2 or VMware Cloud on AWS, BYOL is the only supported option.

Oracle License Portability to AWS

Subject to the terms and conditions of the specific license agreement, Oracle licenses may be portable to AWS. In other words, your existing licenses can be transferred for use on AWS. These include:

- Server-based licenses (based on CPUs used)
- Enterprise License Agreements (ELA)
- Unlimited License Agreements (ULA)
- Business Process Outsourcing (BPO) licenses
- Oracle PartnerNetwork (OPN) licenses
- Named User Plus licenses

Additional conditions or limitations (including possible costs) may be applicable for licenses that are ported to AWS. Please check your specific license agreement for additional details and limitations.

Oracle licensing applies similarly to Oracle Database on Amazon RDS and on Amazon EC2 with the exception that hourly licensing is available only on Amazon RDS.

Choosing Between Amazon RDS, Amazon EC2, or VMware Cloud on AWS for your Oracle Database

Both Amazon RDS and Amazon EC2 offer different advantages for running Oracle Database. Amazon RDS is easier to set up, manage, and maintain than running Oracle Database in Amazon EC2, and lets you focus on other tasks rather than the day-to-day administration of Oracle Database. Alternatively,

running Oracle Database in Amazon EC2 gives you more control, flexibility, and choice. Depending on your application and your requirements, you might prefer one over the other.

If you are migrating multiple Oracle databases to AWS, you will find that some of them are a great fit for Amazon RDS while others are better suited to run directly on Amazon EC2. Many AWS customers run several databases across Amazon RDS, Amazon EC2, and VMware Cloud on AWS for their Oracle Database workloads.

Amazon RDS might be a better choice for you if:

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

- You need a highly available database solution and want to take advantage of the push-button, synchronous Multi-AZ replication offered by Amazon RDS, without having to manually set up and maintain a standby database.
- You would like to have synchronous replication to a standby instance for high availability for Oracle Database Standard Edition One or Standard Edition Two.
- You want to pay for the Oracle license as part of the instance cost on an hourly basis instead of making a large upfront investment.
- Your database size and IOPS needs are less than the RDS Oracle limits.
- You don't want to manage backups and, most importantly, point-in-time recoveries of your database.
- You would rather focus on high-level tasks, such as performance tuning and schema

optimization, rather than the daily administration of the database.

- You want to scale the instance type up or down based on your workload patterns without being concerned about licensing and the complexity involved.

Amazon EC2 might be a better choice for you if:

- You need full control over the database, including SYS/SYSTEM user access, or you need access at the operating system level.
- Your database size exceeds the 80% of current maximum database size in Amazon RDS.
- You need to use Oracle features or options that are not currently supported by Amazon RDS.
- Your database IOPS needs are higher than the current IOPS limit. See the documentation for the current IOPS limit.

- You need a specific Oracle Database version that is not supported by Amazon RDS. For more information, see Oracle Database Editions.

VMware Cloud on AWS might be a better choice for you if:

- Your Oracle databases are already running in on-premises data center in vSphere virtualized environments.
- You need to run Oracle Real Application Clusters (RAC) in the cloud.
- You have large number of databases and you need faster migration (in order of few hours) to migrate to cloud without any migration team man-hours.
- You need to preserve the IP addresses of the databases and applications, when migrating to cloud, to avoid any post-migration rework.

- You need the performance of NVMe storage in Amazon EC2 bare metal hosts along with data persistence.

Architecting for Security and Performance

Whether you choose to run Oracle Database on Amazon RDS or Amazon EC2, optimizing every component of the infrastructure will enhance security, performance, and reliability. In the following sections, we'll discuss best practices for optimizing network configuration, instance type, and database storage in an Oracle Database implementation on AWS.

Network Configuration

With Amazon Virtual Private Cloud (Amazon VPC), you can provision a logically isolated section of the AWS Cloud that is dedicated to your account. You have complete control over your virtual net-

working environment, including selection of your own IP address range, creation of subnets, security settings, and configuration of route tables and network gateways.

A *subnet* is a range of IP addresses in your Amazon VPC. You can launch AWS resources into a subnet that you select. Use a public subnet for resources that must be connected to the internet, and a private subnet for resources that won't be connected to the internet.

To protect the AWS resources in each subnet, you can use multiple layers of security, including security groups and network access control lists (ACLs).

The following table describes the basic differences between security groups and network ACLs.

Security Group	Network ACL

Operates at the instance level (first layer of defense)	Operates at the subnet level (second layer of defense)
Supports **allow** rules only	Supports **allow** rules and **deny** rules
Stateful: Return traffic is automatically allowed, regardless of any rules	Stateless: Return traffic must be explicitly allowed by rules
Evaluates all rules before deciding whether to allow traffic	Processes rules in numerical order when deciding whether to allow traffic
Applies to an instance only if someone specifies the security group when launching the	Automatically applies to all instances in the subnets it's associated with (backup layer of defense, so you don't

instance, or associates the security group with the instance later on	have to rely on someone specifying the security group)

Amazon VPC provides isolation, additional security, and the ability to separate Amazon EC2 instances into subnets, and allows the use of private IP addresses. All of these are important in database implementation. Deploy the Oracle Database instance in a private subnet and allow only application servers within the Amazon VPC, or a bastion host within the Amazon VPC, to access the database instance. Create appropriate security groups that allow access only to specific IP addresses through the designated ports. These recommendations apply to Oracle Database regardless of whether you're using Amazon RDS or Amazon EC2.

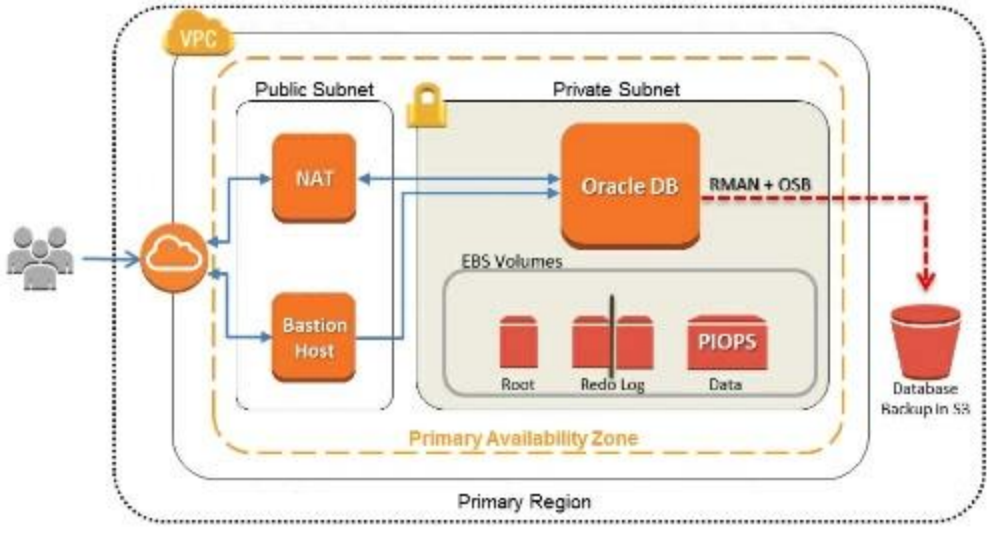


Figure 1: Oracle Database in private subnet of an Amazon VPC

Amazon EC2 Instance Type

AWS has a large number of Amazon EC2 instance types available, so you can choose the instance type that best fits your workload. However, not all the available instance types are best suited for running Oracle Database.

If you use Amazon RDS for your Oracle Database, AWS filters out some of the instance types based on best practices, and gives you the various options

in T- class, M-class and R-class instances. We recommend that you choose db.m- based or r-based Amazon RDS instances for any enterprise database workloads. Your choice of the Amazon RDS instance type should be based on the database workload and the Oracle Database licenses available.

If you're running your self-managed database on Amazon EC2, you have many more choices available for the Amazon EC2 instance type. This is often one of the reasons users opt to run Oracle Database on Amazon EC2 instead of using Amazon RDS. Very small instance types are not suitable because Oracle Database is resource-intensive when it comes to CPU usage. Instances with a larger memory footprint help improve database performance by providing better caching and a bigger system global area (SGA). We recommend that you choose instances that have a good balance of memory and CPU. Choose the instance type that matches the Oracle Database licenses you are planning to use and

the architecture you are planning to implement. For architectures best suited for your business needs, see the whitepaper Advanced Architectures for Oracle Database on Amazon EC2.

Oracle Database uses disk storage heavily for read/write operations, so we highly recommend that you use only instances optimized for Amazon Elastic Block Store (Amazon EBS). Amazon EBS-optimized instances deliver dedicated throughput between Amazon EC2 and Amazon EBS. Bandwidth and throughput to the storage subsystem is crucial for good database performance. Choose instances with higher network performance for better database performance.

The following instance families are best suited for running Oracle Database on Amazon EC2.

Instance Family	Features

M family	• EBS-optimized by default at no additional cost • Balance of compute, memory, and network resources
X family	• Lowest price per GiB of RAM • SSD Storage and EBS-optimized by default and at no additional cost • Ability to control processor C-state and P-state configuration
R family	• Optimized for memory-intensive applications • High-frequency Intel Xeon E5-2686 v4

	(Broadwell) Processors • DDR4 Memory
I family	• Optimized for low latency, very high random I/O performance, high sequential read throughput, and provide high IOPS at a low cost • NVMe SSD ephemeral storage
Z1d family	• Sustained all core frequency of 4.0 GHz • Delivers a 1:8 vCPU to memory ratio

Database Storage

Most users typically use Amazon EBS for database storage. For some very high-performance architectures, you can use instance storage SSDs, but they should be augmented with Amazon EBS storage for reliable persistence.

For high and consistent IOPS and database performance, we highly recommend using General Purpose (GP2) volumes or Provisioned IOPS (PIOPS) volumes. GP2 and PIOPS volumes are available for both Amazon EC2 and Amazon RDS. GP2 volumes provide an excellent balance of price and performance for most database needs. When your database requires higher IOPS than what GP2 can provide, PIOPS volumes are the right choice.

For PIOPS volumes, you specify an IOPS rate when you create the volume, and Amazon EBS delivers within 10% of the Provisioned IOPS performance 99.9% of the time over a given year. The ratio of IOPS provisioned to the volume size requested can

be a maximum of 30. For example, to get 3,000 IOPS your volume size should be at least 100 GB.

Similar to PIOPS volumes, GP2 volumes are also SSD-based, but the IOPS you get from GP2 volumes can vary from a baseline IOPS up to a maximum burstable 3,000 IOPS per volume. This works very well for most database workloads because the IOPS performance needed from the database varies many times during a period of time based on the load size and the number of queries being executed.

General Purpose (SSD) volume performance is governed by volume size, which dictates the base performance level of the volume and how quickly it accumulates I/O credits. Larger volumes have higher base performance levels and accumulate I/O credits faster.

I/O credits represent the available bandwidth that your General Purpose (SSD) volume can use to burst large amounts of I/O when more than the base per-

formance is needed. The more credits your volume has for I/O, the more time it can burst beyond its base performance level and the better it performs when more performance is needed.

Throughput Optimized HDD volumes (st1) offers low-cost HDD volume designed for intensive workloads that require less IOPS but high throughput. Oracle databases used for data warehouses and data analytics purposes can leverage st1 volumes. Any log processing or data staging areas like Oracle external tables or external BLOB storage that require high throughput can leverage st1 volumes. Throughput optimized (st1) volumes can handle max 500 IOPS per volume.

Cold HDD volumes (sc1) are suitable for handling legacy systems, which are kept around for the purposes of occasional reference or archive purposes. These systems are accessed less frequently and a few scans are performed per day on the volume.

A good approach is to estimate the amount of IOPS consistently needed for your database, and allocate enough GP2 storage to obtain that many IOPS. Any additional IOPS needed for periodic spikes should be covered by the burst performance based on the available credits.

The burst duration of a volume is dependent on the size of the volume, the burst IOPS required, and the credit balance when the burst begins. If you notice that your volume performance is frequently limited to the base level (due to an empty I/O credit balance), you should consider using a larger General Purpose (SSD) volume (with a higher base performance level) or switching to a Provisioned IOPS (SSD) volume for workloads that require sustained IOPS performance greater than 10,000 IOPS. For additional details about GP2 volumes, see the Amazon EBS User Guide.

For Amazon RDS, General Purpose (SSD) storage delivers a consistent baseline of 3 IOPS per provisioned GB and provides the ability to burst up to 3,000 IOPS. If you are already using magnetic storage for Amazon RDS, you can convert to General Purpose (SSD) storage, but you will encounter a short availability impact when doing so. Using Provisioned IOPS, you can provision up to the current maximum storage limit and the maximum IOPS per database instance. Your actual realized IOPS may vary from the amount you provisioned based on your database workload, instance type, and database engine. For more information, see Factors That Affect Realized IOPS Rates in the *Amazon RDS User Guide*.

For Oracle Database on Amazon EC2, stripe multiple volumes together for more IOPS and larger capacity. You can use multiple Amazon EBS volumes individually for different data files, but

striping them together allows better balancing and scalability. Oracle Automatic Storage Management (ASM) can be used for striping. Keep data files, log files, and binaries on separate Amazon EBS volumes, and take snapshots of log file volumes on a regular basis. Choosing an instance type with local SSD storage allows you to boost the database performance by using Smart Flash Cache (that is, if the operating system is Oracle Linux) and by using local storage for temporary files and table spaces.

For Oracle Database on VMware Cloud on AWS, vSAN provides necessary virtualized storage striped across the bare metal hosts. vSAN virtualized storage capability can be used in Oracle RAC for high-performance shared storage. The VMDK (virtual machine disk) files created for Oracle RAC have to be eager zero thick type along with multi-writer flag enabled.

Architecting for High Availability

The three options have different approaches to high availability of Oracle databases.

Amazon RDS

The Multi-AZ feature of Amazon RDS operates two databases in multiple Availability Zones with synchronous replication, thus creating a highly available environment with automatic failover. Amazon RDS has failover event detection, and initiates automated failover when failover events occur. You can also initiate manual failover through Amazon RDS API. Amazon RDS provides SLA with monthly uptime percentage of 99.95%. Another option for Amazon RDS Oracle is to use Oracle Active Data Guard. Customers should have their own license for the Oracle Active Data Guard option. Amazon RDS for Oracle support read replicas using Oracle Active

Data Guard. Both Multi-AZ and Oracle Active Data Guard options are within the same AWS Region. Amazon RDS for Oracle is also compatible with Oracle GoldenGate. You can choose to replicate the entire database or few tables and schemas with Oracle GoldenGate. Oracle GoldenGate is installed in a hub architecture in an EC2 instance and accesses the Amazon RDS for Oracle instance remotely. Oracle GoldenGate hub can replicate the data to another Amazon RDS for Oracle instance or Oracle database in Amazon EC2 or VMware Cloud on AWS within the same AWS Region. For cross-region instances, it is recommended to replicate to another regional Oracle GoldenGate hub first.

Amazon EC2

Oracle databases on Amazon EC2 support Oracle Data Guard, Oracle Active DataGuard, Oracle GoldenGate options as well. Third-party solutions available in AWS Marketplace also support replica-

tion for Oracle databases. Both Oracle and third-party solutions can be used to replicate databases within AWS Region and across AWS Regions as well. Oracle databases can be replicated from and to the customer's on-premises data centers as well. If only few tables or schemas are needed, then AWS Database Migration Service (AWS DMS) tasks can be enabled to do ongoing table or schema level replication.

VMware Cloud on AWS

Since Oracle database is self-managed within VMware Cloud on AWS, all options including third-party agent-based replication are available. To replicate databases across AWS Regions or to the customer's on-premises data center, Oracle Data Guard or Oracle GoldenGate can be used. VMware native technologies such as vMotion or Hybrid Cloud Extension (HCX) can be used to migrate databases between on-premises data centers and

VMware Cloud on AWS. For large deployments involving multiple layers of application and database VMs, VMware Site Recovery Manager (SRM) can be considered to orchestrate replication and migration at site level.

Oracle Real Application Cluster (RAC)

VMware Cloud on AWS has capability for multicast support and shared storage. Oracle RAC can be installed on VMware Cloud on AWS. Each Software Defined Data Center (SDDC) in VMware Cloud on AWS can run at the minimum 3 AWS bare metal hosts and maximum of 16 AWS bare metal hosts. VMware Cloud on AWS can run SDDC in stretched cluster fashion across 2 different AWS AZ. This will also allow Oracle RAC to run in extended cluster mode, avoiding the need for a separate Oracle Data Guard setup. VMware Cloud on AWS vSAN supports Oracle ASM. Oracle ASM disk group files are created out of VMDKs. The recommended allocation unit

for Oracle ASM disk group for datafiles and log files should be 4 MB. This option is allowed during the creation of ASM disk group and cannot be modified later. For best performance, VMDK should have multi-writer flag enabled, and eager zero thick format.

Backup Storage

Most Oracle Database users take regular hot and cold backups. Cold backups are taken while the database is shut down, whereas hot backups are taken while the database is active. AWS native storage services offer a choice of solutions for your needs.

Amazon S3

Store your hot and cold backups in Amazon Simple Storage Service (Amazon S3) for high durability and easy access. You can use AWS Storage Gateway file

interface to directly back up the database to Amazon S3. AWS Storage Gateway file interface provides an NFS mount for S3 buckets. Oracle RMAN backups written into the NFS mount are automatically copied to S3 buckets by the AWS Storage Gateway instance.

Amazon S3 Glacier

Amazon S3 Glacier is a secure, durable, and extremely low-cost cloud storage service for data archiving and long-term backup. You can use lifecycle policies in Amazon S3 to move older backups to Amazon S3 Glacier for long-term archiving. Amazon S3 Glacier offers three options for data retrieval with varying access times and costs: Expedited, Standard, and Bulk retrievals.

Amazon S3 Glacier Deep Archive

Amazon S3 Glacier Deep Archive is designed for long-term retention and digital preservation for the

data that might be accessed once or twice a year. All objects stored in S3 Glacier Deep Archive are replicated and stored across at least three geographically dispersed Availability Zones, protected by 99.999999999% of durability, and can be restored within 12 hours.

Amazon EFS

Amazon Elastic File System (Amazon EFS) provides simple, scalable file storage for use with Amazon EC2 instances in the AWS Cloud. With Amazon EFS, storage capacity is elastic, growing and shrinking automatically as you add and remove files, so your applications have the storage they need, when they need it. Backups stored in EFS can be shared with NFS options (read/write, read-only) to other EC2 instances. Amazon EFS uses bursting model for EFS performance. Accumulated burst credits give the file system permission to drive throughput above its baseline rate. A file system can drive throughput

continuously at its baseline rate. Whenever it's inactive or driving throughput below its baseline rate, the file system accumulates burst credits. Amazon EFS is useful when you have to refresh dev and test databases from production database RMAN backups regularly. Amazon EFS can also be mounted in on-premises data centers when connected to your Amazon VPC with AWS Direct Connect. This option is useful when the source Oracle database is in AWS and other refreshed Oracle databases are in on-premises data centers. Backups stored in EFS can be copied to an S3 bucket using AWS CLI commands. See the documentation for getting started on EFS.

Amazon EBS Snapshots

You can back up the data on your Amazon EBS volumes to Amazon S3 by taking point-in-time snapshots. Snapshots are incremental backups, which means that only the blocks on the device

that have changed after your most recent snapshot are saved. When you create an EBS volume based on a snapshot, the new volume begins as an exact replica of the original volume that was used to create the snapshot. The replicated volume loads data lazily in the background so that you can begin using it immediately. If you access data that hasn't been loaded yet, the volume immediately downloads the requested data from Amazon S3, and then continues loading the rest of the volume's data in the background. See the documentation on creating and managing EBS snapshots.

Management

Automation

Oracle database creation and deployment can be automated using AWS CloudFormation templates.

Oracle AMIs

An Amazon Machine Image (AMI) provides the information required to launch an instance, which is a virtual server in the cloud. You specify an AMI when you launch an instance, and you can launch as many instances from the AMI as you need.

Oracle periodically provides official AMIs for some Oracle products, including Oracle Database, on AWS. However, Oracle-provided database AMIs that are available might not always be the latest version. Oracle-supplied AMIs are based on the Oracle Linux operating system.

You are not required to use an Oracle-provided AMI to install and use Oracle Database on Amazon EC2. You can start an Amazon EC2 instance with an operating system AMI, and then download and install Oracle Database software from the Oracle website, just as you would with a physical server.

After you have the first environment set up with all the necessary Oracle software, you can create your own custom AMI for subsequent installations. You can also directly launch AMIs from AWS Marketplace. You should closely scrutinize any community AMIs provided by third parties for security and reliability before using them. AWS is not responsible or liable for their security or reliability.

AWS Systems Manager

AWS Systems Manager is a collection of capabilities that helps you automate management tasks such as systems inventory, applying operational patches, automatic creation of AMIs, and configuring operating systems and applications at scale. Systems Manager uses an SSM (System State Management) Agent to collect inventory, state information within the EC2 instance, and run patch commands. Patch Manager integrates with AWS Identity and Access Management (IAM), AWS CloudTrail, and

Amazon CloudWatch Events to provide a secure patching experience that includes event notifications and the ability to audit usage.

Conclusion

Depending on your usage scenario, you can use Amazon RDS for Oracle Database or run a self-managed Oracle Database on Amazon EC2. Regardless of your choice, by following the best practices provided in this paper you can get the best out of your Oracle Database implementation on AWS.