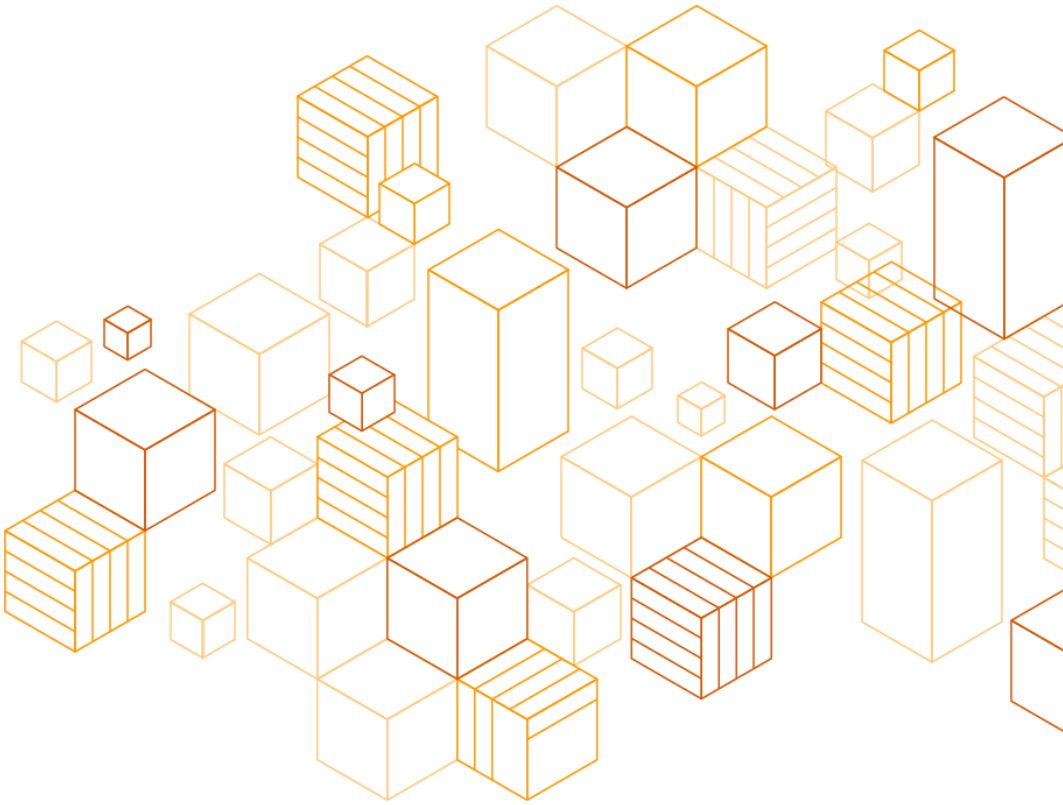
Enabling High Availability with Data Guard on Amazon RDS Custom for Oracle

Technical Guide

October 26, 2021



Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Contents

About this guide

This guide explains how to enable high availability (HA) with Oracle Data Guard software on your Amazon Relational Database Service (Amazon RDS) Custom for Oracle database (DB) instances. HA is defined as when data from the primary DB instance is synchronously replicated to a standby instance in a different Availability Zone (AZ). Running a DB instance with HA can enhance availability during planned system maintenance, and help protect your databases against DB instance failure and AZ disruption. You can also configure standby instances to be read-only. Depending on your Oracle Data Guard configuration, you can fail over or switch over from the primary database instance to the standby instance with no data loss. To migrate an on-premises database instance to RDS Custom, you can configure HA for your on-premises database instance, and then fail over or switch over to the RDS Custom standby instance.

This guide also explains how to encrypt your HA instances with a VPN tunnel, configure Oracle Fast-Failover Observer (FSFO) to monitor your HA instances, and allow the observer to perform automatic failover when the necessary conditions are met.

Overview

Amazon Relational Database Service (Amazon RDS) makes it easy to set up, operate, and scale a relational database in the AWS Cloud. Amazon RDS Custom is a new deployment type that provides granular control capabilities which enable customers to access and customize the database environment, and provides an operating system which allows customers to run third-party software on AWS.

Like Amazon RDS, Amazon RDS Custom automates the undifferentiated heavy lifting, such as database provisioning, patching, and backups. Amazon RDS Custom gives you privileged access to the database and underlying operating system in order to support legacy applications, manually install OS patches and packages, apply custom database patches, and install third-party security and diagnostic software. You can also configure database settings and Network File System (NFS) on Amazon RDS Custom.

Oracle Data Guard is a feature of Oracle Database, and it allows you to create and maintain standby databases as transactionally consistent copies of a production database. If the production database becomes unavailable, Oracle Data Guard can promote any standby database to be the production database. This technique minimizes downtime of the production database. Oracle Data Guard ensures HA, data protection, and disaster recovery for your production database.

This guide presents one approach to enable HA on your RDS Custom Oracle instances using Oracle Data Guard. While there are other approaches that are possible, this guide outlines the supported configuration.

Before you begin/considerations

Before you decide to set up HA for your instance, this guide recommends that you have good background knowledge of data guard setups, and that you design HA in a database according to your latency requirements. We strongly recommend thoroughly testing applications and databases before proceeding with production.

Procedural sections

> **Note:** The examples below are described in rds-preview stage in us-east-2 region. Choose your own AWS regions as appropriate.

Step 1: Create and pre-configure the RDS Custom instances

To prepare the environment, create two RDS Custom instances (one for the primary database, and the other for the standby database) by following these steps:

1. Create an RDS Custom DB instance by following the steps in the "Creating an Amazon RDS Custom DB instance" section of the Amazon RDS User Guide. This instance is the primary and the following examples refer to this instance as the `primary_instance`. Make sure the backup retention of the primary instance is greater than 0.

> **Note:** You can also use your on-premises instance as the `primary_instance`. In this case, skip to step 2. In step 3, create a new `standby_instance` in the RDS Custom platform with the same binary that you use for the `primary_instance`.

2. Take a snapshot of the `primary_instance` using the instructions in the "Creating an RDS Custom snapshot" section of the Amazon RDS User Guide, or use the read replica API if the `primary_instance` is an RDS Custom instance. You should choose the same DB name as the `primary_instance` for the `secondary_instance`. You can use the `DBNAME` parameter to specify the database name, which should be same as the primary name.
3. Restore the snapshot to the second RDS Custom instance using the instructions in the "Restoring an RDS Custom DB snapshot" section of the Amazon RDS User Guide. Make sure the instance is restored to the same VPC and subnet group, and use this instance as the standby instance. The following examples refer to this instance as the `standby_instance`.
    a. You can find the subnet group name of the `primary_instance` with the following commands:

```
aws rds --region us-east-2 --endpoint-url https://rds-preview.us-
east-2.amazonaws.com \
describe-db-instances \
--db-instance-identifier <Identifier for primary_instance> \
--query 'DBInstances[*].[DBSubnetGroup.DBSubnetGroupName]' \
--output text
```

   b. AWS highly recommends that you create the `standby_instance` in a different availability zone from the `primary_instance` to be resilient to availability zone failure. You can do this by specifying a different value for the `--availability-zone` parameter.

4. Pause automation on both the `primary_instance` and the `standby_instance` using the instructions in the "Pausing and resuming RDS Custom automation" section of the Amazon RDS User Guide. Pause the instances for as long as needed to complete the configuration.
5. Ensure that the security groups for both the `primary_instance` and the `standby_instance` permit inbound and outbound connections on port 1140 (Type: `All TCP`, Protocol: `TCP`, Port range: `1140`, Source: `Custom`, plus the security group or IP address of the other instance). Port 1140 is reserved for Oracle Data Guard, so don't use this port for any other applications. To learn how to configure your instance security group, see "Tutorial: Create an Amazon VPC for use with a DB instance".

Step 2: Fetch the SYS user password of `primary_instance`

Fetch the SYS user password for the `primary_instance` from AWS Secrets Manger, and make sure the Secrets Manager password is the same between the `primary_instance` and the `standby_instance`.

AWS Management Console: Fetch the SYS password

> **Note**: If you have changed the password for the SYS user in the past, please use the updated password in future steps when the SYS password is needed, and still go through the steps below to make sure the password in Secrets Manager is the same between the `primary_instance` and the `standby_instance`.

1. Sign in to the AWS Management Console (the 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 `primary_instance`.
3. Choose **Configuration**.
4. Note the **resource ID** for the `primary_instance` (it will be in this format: `db-ABCDEFGHIJKLMNOPQRS0123456`). This paper refers to the resource ID as `primary_resource_id` in this section.
5. Open the AWS Secrets Manager console at: https://console.aws.amazon.com/secretsmanager/.
6. Choose the secret that has the same name as **do-not-delete-custom-<primary_resource_id>-*****.
7. Choose **Retrieve secret value** and note the password for SYS user.

Make sure the `primary_instance` and `standby_instance` passwords in Secrets Manager are the same.

1. Repeat steps 4, 5, 6, and 7 above for the `standby_instance` to fetch the password there and compare it with the one fetched in step 7. If it is different, proceed to step 2 below. Otherwise, skip step 2.
2. Choose **Retrieve secret value** for the secret of the `standby_instance`, choose **Edit, and** replace the secret value with the one retrieved in step 7 above.

AWS Command Line Interface (CLI)

Example:

To find the DB resource ID of your RDS Custom DB instance, use the `describe-db-instances` command.

```
aws rds --region us-east-2 --endpoint-url https://rds-preview.us-
east-2.amazonaws.com \
describe-db-instances \
 --query 'DBInstances[*].[DBInstanceIdentifier,DbiResourceId]' \
 --output text
```

The following sample output shows the resource ID for your RDS Custom instance. The prefix is `db-`.

```
db-ABCDEFGHIJKLMNOPQRS0123456
```

To find the secret name of the instance's SYS user's secret, use `aws secretsmanager list-secrets`. The following example uses `db-ABCDEFGHIJKLMNOPQRS0123456` for the resource ID.

```
aws secretsmanager --region us-east-2 \
list-secrets \
--query 'SecretList[*].[Name]' \
--output text | grep "db-ABCDEFGHIJKLMNOPQRS0123456"
```

The following sample output shows the secret name.

```
do-not-delete-custom-db-ABCDEFGHIJKLMNOPQRS0123456-7af9e5
```

You can run the commands in the preceding examples to get the secret name of the `primary_instance` and the `standby_instance`.

To find the secret value of the secret, use `aws secretsmanager get-secret-value`. The following example uses `do-not-delete-custom-db-ABCDEFGHIJKLMNOPQRS0123456-7af9e5` as the secret name.

```
aws secretsmanager --region us-east-2 \
get-secret-value \
--secret-id do-not-delete-custom-db-ABCDEFGHIJKLMNOPQRS0123456-
7af9e5 \
--query SecretString \
--output text
```

The following sample output shows the secret value. The secret value is the password this section is trying to retrieve.

```
hTeudjeijndFDFQLZXwyNDKbKUoJtf1233_9
```

To update the secret value of a secret, use `aws secretsmanager update-secret`. The following examples updates the secret value of the secret: `do-not-delete-custom-db-ABCDEFGHIJKLMNOPQRS0123456-7af9e5` to `someNewSecret123`.

```
aws secretsmanager --region us-east-2 \
update-secret --secret-id do-not-delete-custom-db-
ABCDEFGHIJKLMNOPQRS0123456-7af9e5 \
--secret-string someNewSecret123
```

You can update the secret value for the `standby_instance` with the `primary_instance`'s secret value using the preceding examples if they are different.

Step 3: Drop the database on `standby_instance`

This step is necessary because you must create a new Oracle home for the standby database instance.

1. Connect to the `standby_instance` by following the steps in the “Connecting to your RDS Custom DB instance using AWS Systems Manager” section of the Amazon RDS User Guide.
2. Drop your database using a SQL statement. Complete the following tasks:
   a. Switch your root user to `rdsdb`. RDS Custom uses the `rdsdb` user and `rdsdb` group to run the database.

```
$ sudo su - rdsdb
```

   b. Start SQL*Plus, shut down the database, and then drop the database.

```
$ sqlplus / as sysdba
SQL> SHUTDOWN IMMEDIATE
```

```
SQL> STARTUP MOUNT EXCLUSIVE RESTRICT
SQL> DROP DATABASE;
```

Step 4: Note the private IP addresses of the Amazon Elastic Compute Cloud (Amazon EC2) instances

Find the IP addresses for both the `primary_instance` and the `standby_instance`.

AWS Management Console

1. Sign in to the 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 RDS Custom DB instance whose IP address you want to find.
3. Choose **Configuration**.
4. Note the **resource ID** for your DB instance. For example, the resource ID might be `dbABCDEFGHIJKLMNOPQRS0123456`.
5. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/ (sign-in required).
6. In the upper right, choose **US East (Ohio) us-east-2** as the region.
7. In the navigation pane, choose **Instances**.
8. Find the instance whose **name** is the same as the **resource ID** in step 4.
9. In the **Details** section, note the value for **Private IPv4 addresses**.

AWS CLI

Example:

To find the DB resource ID of your RDS Custom DB instance, use `aws rds describe-db-instances`.

```
aws rds --region us-east-2 --endpoint-url https://rds-preview.us-
east-2.amazonaws.com \
describe-db-instances \
```

```
 --query 'DBInstances[*].[DBInstanceIdentifier,DbiResourceId]' \
 --output text
```

The following sample output shows the resource ID for your RDS Custom instance. The prefix is `db-`.

```
db-ABCDEFGHIJKLMNOPQRS0123456.
```

To find the private IP address of the corresponding Amazon EC2 instance, use `aws ec2 describe-instances`.

```
aws ec2 describe-instances --region us-east-2 \
   --filters Name=tag:Name,Values=db-ABCDEFGHIJKLMNOPQRS0123456 \
   --query
'Reservations[*].Instances[*].NetworkInterfaces[*].[PrivateIpAddres
s]' \
   --output text
```

In this document, the private IP address for the `primary_instance` is `primary_instance_ipv4`. The private IP address for the `standby_instance` is `standby_instance_ipv4`.

Step 5: Configure the Oracle Data Guard listeners

Create new Oracle Data Guard listeners for both the `primary_instance` and the `standby_instance`. `ORCL` below is the database name.

1. Connect to the `primary_instance` by following the steps in the “Connecting to your RDS Custom DB instance using AWS Systems Manager” section of the Amazon RDS User Guide.
2. Switch your root user to `rdsdb`. RDS Custom uses the `rdsdb` user and `rdsdb` group to run the database.

   ```
   $ sudo su - rdsdb
   ```

3. Create the `listener.ora` file (if it doesn’t exist) in `/rdsdbdata/config`.

Note: there should be a symlink in `/rdsdbbin/oracle/network/admin/listener.ora` that points to the `listener.ora` file in `/rdsdbdata/config/`. If not, create the symlink by using:

```
$ cd /rdsdbbin/oracle/network/admin/
$ ln -s /rdsdbdata/config/listener.ora
```

4. Append the following entry in a new line without changing the existing entries in the file:

```
ADR_BASE_L_ORCL_DG=/rdsdbdata/log
SID_LIST_L_ORCL_DG=(SID_LIST = (SID_DESC = (SID_NAME =
ORCL)(GLOBAL_DBNAME = ORCL) (ORACLE_HOME = /rdsdbbin/oracle)))
L_ORCL_DG=(DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(PORT =
1140)(HOST = <primary_instance_ipv4>)))
SUBSCRIBE_FOR_NODE_DOWN_EVENT_L_ORCL_DG=OFF
```

5. Start the new Oracle Data Guard listener with the following command. Make sure you see the message “The command completed successfully” in the output.

```
$ /rdsdbbin/oracle/bin/lsnrctl start L_ORCL_DG
```

6. It is also recommended to check the status of the Oracle Data Guard listener with the command below.

```
$ /rdsdbbin/oracle/bin/lsnrctl status L_ORCL_DG
```

7. Connect to the `standby_instance` by following the steps in the “Connecting to your RDS Custom DB instance using AWS Systems Manager” section of the Amazon RDS User Guide.
8. Switch your root user to `rdsdb`. RDS Custom uses the `rdsdb` user and `rdsdb` group to run the database.

```
$ sudo su - rdsdb
```

9. Create the `listener.ora` file in `/rdsdbdata/config`, if the file doesn't already exist.

> **Note**: there should be a symlink in `/rdsdbbin/oracle/network/admin/listener.ora` that points to the `listener.ora` file in `/rdsdbdata/config/`. If not, create the symlink by using:

```
$ cd /rdsdbbin/oracle/network/admin/
$ ln -s /rdsdbdata/config/listener.ora
```

10. Append the following entry in a new line without changing the existing entries in the file:

```
ADR_BASE_L_ORCL_DG=/rdsdbdata/log
SID_LIST_L_ORCL_DG=(SID_LIST = (SID_DESC = (SID_NAME =
ORCL)(GLOBAL_DBNAME = ORCL) (ORACLE_HOME = /rdsdbbin/oracle)))
L_ORCL_DG=(DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(PORT =
1140)(HOST = <standby_instance_ipv4>)))
SUBSCRIBE_FOR_NODE_DOWN_EVENT_L_ORCL_DG=OFF
```

11. Start the new Oracle Data Guard listener. Make sure you see message "The command completed successfully" in the output.

```
$ /rdsdbbin/oracle/bin/lsnrctl start L_ORCL_DG
```

12. It is also recommended to check the status of the Oracle Data Guard listener with the command below:

```
$ /rdsdbbin/oracle/bin/lsnrctl status L_ORCL_DG
```

Step 6: Configure the `tnsnames.ora` files

Configure the `tnsnames.ora` file on both the primary and standby instances. This configuration is necessary to define the database connection addresses and alias.

1. Connect to the `primary_instance` and the `standby_instance` in separate browser instances by following the instructions in the “Connecting to your RDS Custom DB instance using AWS Systems Manager” section of the Amazon RDS User Guide.
2. Switch your root user to `rdsdb`. RDS Custom uses the `rdsdb` user and `rdsdb` group to run the database.

```
$ sudo su - rdsdb
```

3. Create the `tnsnames.ora` file (if it doesn’t exist) in `/rdsdbdata/config/`

Note: there should be a symlink in `/rdsdbbin/oracle/network/admin/tnsnames.ora` that points to the `tnsnames.ora` file in `/rdsdbdata/config/`. If not, create the symlink by using:

```
$ cd /rdsdbbin/oracle/network/admin/
$ ln -s /rdsdbdata/config/tnsnames.ora
```

4. Define the Transparent Network Substrate (TNS) address names for the `primary_instance` and the `standby_instance`. To allow better identification for the Oracle Data Guard instances, the TNS address name is usually the same as the database unique name. In this document, the TNS address name and the database unique name for the `primary_instance` is `ORCL_A`, and for the `standby_instance` is `ORCL_B`.
5. Put the following entry in the `tnsnames.ora`. Replace `ORCL_A` and `ORCL_B` with the TNS address names you choose. Replace `<primary_instance_ipv4>` and `<standby_instance_ipv4>` as the values obtained in the "Step 4: Note the private IP addresses of the Amazon EC2 instances” section of the Amazon RDS User Guide.

```
ORCL_A
=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=<primar
y_instance_ipv4>)(PORT=1140)))(CONNECT_DATA=(SID=ORCL)))
ORCL_B
=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=<standb
y_instance_ipv4>)(PORT=1140)))(CONNECT_DATA=(SID=ORCL)))
```

6. Verify the networking connection and TNS entries by performing the following commands on both the `primary_instance` and the `standby_instance`. A successful configuration will return `OK`.

```
$ tnsping ORCL_A
$ tnsping ORCL_B
```

Note: If you have multiple standbys then please make sure to add all members' TNS entries in `tnsnames.ora` file of all the standbys and primary database servers.

Step 7: Configure the `primary_instance` in force logging mode

1. Connect to the `primary_instance` by following the steps in the "Connecting to your RDS Custom DB instance using AWS Systems Manager" section of the Amazon RDS User Guide.
2. Switch your root user to `rdsdb`. RDS Custom uses the `rdsdb` user and `rdsdb` group to run the database.

```
$ sudo su - rdsdb
```

3. Start SQL*Plus, and put the database into force logging mode.

```
$ sqlplus / as sysdba
SQL> ALTER DATABASE FORCE LOGGING;
```

Step 8: Configure the initialization parameter files

Synchronize the parameter file on the `standby_instance` with the parameter file on the `primary_instance`.

1. Create the PFILE from the SPFILE on the `primary_instance` as follows:
   a. Connect to the `primary_instance` by following the steps in the "Connecting to your RDS Custom DB instance using AWS Systems Manager" section of the Amazon RDS User Guide.
   b. Switch your root user to `rdsdb`. RDS Custom uses the `rdsdb` user and `rdsdb` group to run the database.

```
$ sudo su - rdsdb
```

   c. Start SQL*Plus, and then create the PFILE from the SPFILE.

```
$ sqlplus / as sysdba
SQL> create pfile='/tmp/initORCL.ora' from spfile;
```

2. Copy `/tmp/initORCL.ora` to the `standby_instance` in the following path: `$ORACLE_HOME/dbs/initORCL.ora`. Either use `scp`, copy and paste the content, or use the base64 encoding to copy, following the example below:

```
(On primary_instance)
$ base64 /tmp/initORCL.ora
<copy the exact output of that command>

(On standby_instance)
$ base64 -d > $ORACLE_HOME/dbs/initORCL.ora
<paste the content copied above>
<press enter>
<press "ctrl + d">
```

3. Check the `md5sum` of the source and target file, and make sure the output is the same by running the commands below:

```
(on primary_instance) $ md5sum /tmp/initORCL.ora
(On standby_instance) $ md5sum $ORACLE_HOME/dbs/initORCL.ora
```

4. Connect to the `standby_instance` by following the steps in the “Connecting to your RDS Custom DB instance using AWS Systems Manager” section of the Amazon RDS User Guide.
5. Change the PFILE content, and create any missing directories as follows:
   a. Open the file using `vi $ORACLE_HOME/dbs/initORCL.ora`. The content of the file appears as follows:

```
ORCL.__data_transfer_cache_size=0
ORCL.__db_cache_size=6039797760
ORCL.__inmemory_ext_roarea=0
ORCL.__inmemory_ext_rwarea=0
ORCL.__java_pool_size=0
ORCL.__large_pool_size=33554432
ORCL.__oracle_base='/rdsdbbin'#ORACLE_BASE set from environment
ORCL.__pga_aggregate_target=4966055936
ORCL.__sga_target=7449083904
ORCL.__shared_io_pool_size=134217728
ORCL.__shared_pool_size=1207959552
ORCL.__streams_pool_size=0
ORCL.__unified_pga_pool_size=0
*.archive_lag_target=300
*.audit_file_dest='/rdsdbdata/admin/ORCL/adump'
*.compatible='19.0.0'
*.control_files='/rdsdbdata/db/ORCL_A/controlfile/control-
01.ctl'
*.db_block_checking='MEDIUM'
*.db_create_file_dest='/rdsdbdata/db'
*.db_name='ORCL'
*.db_recovery_file_dest_size=1073741824
*.db_unique_name='ORCL_A'
*.dbfips_140=FALSE
*.diagnostic_dest='/rdsdbdata/log'
*.filesystemio_options='setall'
*.heat_map='OFF'
*.job_queue_processes=50
*.local_listener='(address=(protocol=tcp)(host=)(port=8200))'
*.log_archive_dest_1='location="/rdsdbdata/db/ORCL_A/arch/redolo
g", valid_for=(ALL_LOGFILES,ALL_ROLES)'
*.log_archive_format='-%s-%t-%r.arc'
*.max_string_size='STANDARD'
*.memory_max_target=12385852416
*.memory_target=12385852416
*.open_cursors=300
*.pga_aggregate_target=0
*.processes=1673
*.recyclebin='OFF'
*.sga_target=0
*.spfile='/rdsdbbin/oracle/dbs/spfileORCL.ora'
*.undo_tablespace='UNDO_T1'
*.use_large_pages='FALSE'
```

b. Adjust the memory parameters to the values that are suitable for the instance class of the `standby_instance`. The list of memory parameters is:

```
ORCL.__data_transfer_cache_size=0
ORCL.__db_cache_size=6039797760
ORCL.__inmemory_ext_roarea=0
ORCL.__inmemory_ext_rwarea=0
ORCL.__java_pool_size=0
ORCL.__large_pool_size=33554432
ORCL.__pga_aggregate_target=4966055936
ORCL.__sga_target=7449083904
ORCL.__shared_io_pool_size=134217728
ORCL.__shared_pool_size=1207959552
ORCL.__streams_pool_size=0
ORCL.__unified_pga_pool_size=0
*.memory_max_target=12385852416
*.memory_target=12385852416
*.open_cursors=300
*.pga_aggregate_target=0
*.processes=1673
*.sga_target=0
```

c. Change the `db_unique_name` initialization parameter to `ORCL_B` and replace every occurrence of `ORCL_A` in the paths with `ORCL_B`. The `db_unique_name` is the same as the TNS address name as mentioned in the preceding Step 6: Configure the `tnsnames.ora` filesection. After the change, the example content looks as follows:

```
ORCL.__data_transfer_cache_size=0
ORCL.__db_cache_size=6039797760
ORCL.__inmemory_ext_roarea=0
ORCL.__inmemory_ext_rwarea=0
ORCL.__java_pool_size=0
ORCL.__large_pool_size=33554432
ORCL.__oracle_base='/rdsdbbin'#ORACLE_BASE set from environment
ORCL.__pga_aggregate_target=4966055936
ORCL.__sga_target=7449083904
ORCL.__shared_io_pool_size=134217728
ORCL.__shared_pool_size=1207959552
ORCL.__streams_pool_size=0
ORCL.__unified_pga_pool_size=0
*.archive_lag_target=300
*.audit_file_dest='/rdsdbdata/admin/ORCL/adump'
*.compatible='19.0.0'
*.control_files='/rdsdbdata/db/ORCL_B/controlfile/control-
01.ctl'
*.db_block_checking='MEDIUM'
*.db_create_file_dest='/rdsdbdata/db'
*.db_name='ORCL'
*.db_recovery_file_dest_size=1073741824
*.db_unique_name='ORCL_B'
*.dbfips_140=FALSE
*.diagnostic_dest='/rdsdbdata/log'
*.filesystemio_options='setall'
*.heat_map='OFF'
*.job_queue_processes=50
*.local_listener='(address=(protocol=tcp)(host=)(port=8200))'
*.log_archive_dest_1='location="/rdsdbdata/db/ORCL_B/arch/redolo
g", valid_for=(ALL_LOGFILES,ALL_ROLES)'
*.log_archive_format='-%s-%t-%r.arc'
*.max_string_size='STANDARD'
*.memory_max_target=12385852416
*.memory_target=12385852416
*.open_cursors=300
*.pga_aggregate_target=0
*.processes=1673
*.recyclebin='OFF'
*.sga_target=0
*.spfile='/rdsdbbin/oracle/dbs/spfileORCL.ora'
*.undo_tablespace='UNDO_T1'
*.use_large_pages='FALSE'
```

6. Except for `'/rdsdbbin/oracle/dbs/spfileORCL.ora'`, Make sure all files and paths in the file exist. If not, create them. The files and paths are as follows:

```
# paths
/rdsdbdata/admin/ORCL/adump
/rdsdbdata/db/ORCL_B/controlfile
/rdsdbdata/db
/rdsdbdata/log
/rdsdbdata/db/ORCL_B/arch/
```

a. The follow example shows how to create one path:

```
$ mkdir -p /rdsdbdata/db/ORCL_B/controlfile
```

7. Create the SPFILE from the PFILE on the `standby_instance`:

   a. Connect to the `standby_instance` by following the steps in the "Connecting to your RDS Custom DB instance using AWS Systems Manager" section of the Amazon RDS User Guide.

   b. Switch your root user to `rdsdb`. RDS Custom uses the `rdsdb` user and `rdsdb` group to run the database.

```
$ sudo su - rdsdb
```

   c. Start SQL*Plus, and create the SPFILE from the PFILE.

```
$ sqlplus / as sysdba
SQL> create spfile =
'/rdsdbdata/admin/ORCL/pfile/spfileORCL.ora' from pfile;
```

8. Find the path and the symbolic link of the SPFILE on the `primary_instance`. Use this path to create the symbolic link on the `standby_instance`.

   a. Connect to the `primary_instance` by following the steps in the "Connecting to your RDS Custom DB instance using AWS Systems Manager" section of the Amazon RDS User Guide.

   b. Switch your root user to `rdsdb`. RDS Custom uses the `rdsdb` user and `rdsdb` group to run the database.

```
$ sudo su - rdsdb
```

c. Start SQL*Plus, and then show the location of the PFILE.

```
$ sqlplus / as sysdba
SQL> show parameter pfile
```

d. Find the path of the SPFILE from the `VALUE` column. The following example uses the path: `/rdsdbbin/oracle/dbs/spfileORCL.ora`.

e. Exit SQL*Plus.

f. Find out where the symbolic link `/rdsdbbin/oracle/dbs/spfileORCL.ora` points to by running the following commands:

```
cd /rdsdbbin/oracle/dbs
ls -lrt
```

g. Note down the path, as you'll need it in the next step. The following example uses the path: `/rdsdbdata/admin/ORCL/pfile/spfileORCL.ora`.

9. Create the same symbolic link on the `standby_instance` as follows:

a. Connect to the `standby_instance` by following the steps in the "Connecting to your RDS Custom DB instance using AWS Systems Manager" section of the Amazon RDS User Guide.

b. Switch your root user to `rdsdb`. RDS Custom uses the `rdsdb` user and `rdsdb` group to run the database.

```
$ sudo su - rdsdb
```

c. Run the following commands. Replace the value of `/rdsdbdata/admin/ORCL/pfile/spfileORCL.ora` to the real path fetched above.

```
cd $ORACLE_HOME/dbs
ln -s /rdsdbdata/admin/ORCL/pfile/spfileORCL.ora
```

d. Move `initORCL.ora` to your temporary directory for backup purposes.

```
mv initORCL.ora /tmp
```

Step 9: Turn on archive mode on the primary_instance

1. Connect to the `primary_instance` by following the steps in the "Connecting to your RDS Custom DB instance using AWS Systems Manager" section of the Amazon RDS User Guide.
2. Switch your root user to `rdsdb`. RDS Custom uses the `rdsdb` user and `rdsdb` group to run the database.

```
$ sudo su - rdsdb
```

3. Start SQL*Plus, and then check the log mode of the database.

```
$ sqlplus / as sysdba
SQL> archive log list;
```

4. If the output shows that the database is in archive mode, then skip the following step:

```
Database log mode Archive Mode
Automatic archival Enabled
```

5. Place the database in archiving mode.

```
SQL> SHUTDOWN IMMEDIATE
SQL> STARTUP MOUNT
SQL> ALTER DATABASE ARCHIVELOG;
SQL> ALTER DATABASE OPEN;
```

Step 10: Configure Oracle Data Guard

Configure Oracle Data Guard on both the `primary_instance` and the `standby_instance`.

1. Connect to the `primary_instance` and the `standby_instance` in separate browser instances by following the instructions in the "Connecting to your RDS Custom DB instance using AWS Systems Manager" section of the Amazon RDS User Guide.
2. On the host of the `primary_instance`, use the `SYS` password to create and encrypt the password file. The `SYS` password is the master user password that you retrieved in the following section:Step 2: Fetch the SYS user password of primary_instance . Use the following commands:

```
$ sudo su - rdsdb
$ $ORACLE_HOME/bin/orapwd file=/rdsdbdata/config/orapw
> Input password for SYS

$ ln -sf /rdsdbdata/config/orapw /rdsdbbin/oracle/dbs/orapwORCL
```

3. On the `primary_instance`, create and set the dedicated user for Oracle Data Guard replication. In this example, the name of the user is `RDS_DATAGUARD`, and the password is `dg_12345$`. It is recommended to use the same username here, but please choose a different password for this user.

```
$ sqlplus / as sysdba
SQL> CREATE USER RDS_DATAGUARD IDENTIFIED BY dg_12345$;
SQL> GRANT SYSOPER, SYSDG, ADMINISTER DATABASE TRIGGER TO
RDS_DATAGUARD;
SQL> ALTER SYSTEM SET REDO_TRANSPORT_USER=RDS_DATAGUARD
SCOPE=BOTH;
```

4. Copy the password file to the `standby_instance` using `scp` or base64 encoding as in the example below:

```
(On the primary_instance)
$ base64 /rdsdbdata/config/orapw
<copy the exact output of that command>

(On the standby_instance)
$ base64 -d > /rdsdbdata/config/orapw
<paste the content copied above>
<press enter>
<press "ctrl + d">

$ ln -sf /rdsdbdata/config/orapw /rdsdbbin/oracle/dbs/orapwORCL
```

5. Check the `md5sum` of the source and target file, and make sure the output is the same by running the commands below:

```
(On the primary_instance)$ md5sum /rdsdbdata/config/orapw
(On the standby_instance)$ md5sum /rdsdbdata/config/orapw
```

6. On the `standby_instance`, start the database in `nomount` mode.

```
$ sqlplus / as sysdba
SQL> STARTUP NOMOUNT;
```

7. On the `standby_instance`, use RMAN to duplicate the database from the `primary_instance`:
   a. Switch to the `rdsdb` user: `$ sudo su - rdsdb`.
   b. Connect RMAN to the target and auxiliary databases. **Note**: Both instances of `<syspwd>` from the command below are the SYS password for the `primary_instance`:

```
$ rman target sys/<syspwd>@ORCL_A auxiliary sys/<syspwd>@ORCL_B
```

   c. Create a duplicate database from the sync-up primary and replica by running the following command:

```
RMAN> run {
allocate channel prmy1 type disk;
allocate channel prmy2 type disk;
allocate channel prmy3 type disk;
allocate channel prmy4 type disk;
allocate auxiliary channel stby type disk;
duplicate target database for standby from active database;
}
```

8. On the `standby_instance`, update the symlinks.

```
ln -sfn /rdsdbdata/log/diag/rdbms/orcl_b/ORCL/trace
/rdsdbbin/oracle/log/trace
ln -sfn /rdsdbdata/log/diag/rdbms/orcl_b/ORCL/incident
/rdsdbbin/oracle/log/incident
```

9. On both the `primary_instance` and the `standby_instance`, start Oracle Data Guard.

```
$ sudo su - rdsdb
$ sqlplus / as sysdba
SQL> ALTER SYSTEM SET dg_broker_start=true;
```

10. Enable the Oracle Data Guard configuration on the `primary_instance`, and add the `standby_instance` to the configuration.
    a. Connect to the `primary_instance`.
    b. Switch to the `rdsdb` user: `$ sudo su - rdsdb`.
    c. Start the Oracle Data Guard software by running: `dgmgrl /`.
    d. Inside the software interface, create the configuration for the `primary_instance`. In the following example, the database unique name for the `primary_instance` is `ORCL_A`.

```
DGMGRL> CREATE CONFIGURATION my_dg_config AS PRIMARY DATABASE IS
ORCL_A CONNECT IDENTIFIER IS ORCL_A;
```

e. Check that the configuration was created successfully by running the following command: `DGMGRL> SHOW CONFIGURATION VERBOSE`; The output should look something like the following:

Members:

```
orcl_a - Primary database
```

f. Add the `standby_instance` to this configuration. In the following example, the database unique name for the `standby_instance` is `ORCL_B`:

```
DGMGRL> ADD DATABASE ORCL_B AS CONNECT IDENTIFIER IS ORCL_B
MAINTAINED AS PHYSICAL;
```

g. Check that the configuration was created successfully by running the following command: `DGMGRL> SHOW CONFIGURATION VERBOSE;`. The output should look something like the following:

Members:

```
orcl_a - Primary database
orcl_b - Physical standby database
```

h. Set the static connect identifiers for both the primary database and the standby database. Replace `<primary_instance_ipv4>` and `<standby_instance_ipv4>` with the IP addresses fetched in Step 4: Note the private IP addresses of the Amazon Elastic Compute Cloud (Amazon EC2) instancesection.

```
DGMGRL> edit database orcl_a set property
StaticConnectIdentifier='(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(PO
RT=1140)(HOST=<primary_instance_ipv4>))(CONNECT_DATA=(SID=ORCL)(
SERVER=DEDICATED)))';
DGMGRL> edit database orcl_b set property
StaticConnectIdentifier='(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(PO
RT=1140)(HOST=<standby_instance_ipv4>))(CONNECT_DATA=(SID=ORCL)(
SERVER=DEDICATED)))';
```

i. On the `primary_instance`, enable the Oracle Data Guard configuration (the following command could take a few minutes to finish).

```
DGMGRL> ENABLE CONFIGURATION;
```

j. On the `standby_instance`, check and update the `fal_client` & `fal_server` values.

i. Start SQL*Plus, create and set the user: `$ sqlplus / as sysdba`.
ii. Check the value of the `fal_client` and `fal_server` by running the statement below:

```
SQL> show parameter fal;
```

iii. If the `VALUE` column for both `fal_client` and `fal_server` is null, then set these values as follows:

```
SQL> alter system set fal_server = 'ORCL_A';
SQL> alter system set fal_client = 'ORCL_B';
```

k. Update the local listener parameters on both the `primary_instance` and the `standby_instance`:

i. On the host of the `primary_instance`, switch to the `rdsdb` user: `$ sudo su - rdsdb`.
ii. Check the `local_listener` parameter with the following commands:

```
$ sqlplus / as sysdba
SQL> show parameter local_listener
```

iii. The value will look like the example below:

```
(address=(protocol=tcp)(host=)(port=8200))
```

iv. Update the `local_listener` parameter with the commands below:

```
SQL> alter system set local_listener='(ADDRESS = (PROTOCOL =
TCP)(PORT = 8200)(HOST = 127.0.0.1)),(ADDRESS = (PROTOCOL =
TCP)(PORT = 1140)(HOST = <primary_instance_ipv4>))';
```

v. Check the `local_listener` parameter again to make sure the value is updated.

vi. Go to the host of the `standby_instance`, and repeat the steps above with the new value of the `local_listener` parameter as below:

```
(ADDRESS = (PROTOCOL = TCP)(PORT = 8200)(HOST =
127.0.0.1)),(ADDRESS = (PROTOCOL = TCP)(PORT = 1140)(HOST =
<standby_instance_ipv4>))
```

Step 11: Configure the standby redo log files on the `primary_instance` and the `standby_instance`

1. Connect to the `primary_instance` and the `standby_instance` in separate browser instances by following the instructions in the "Connecting to your RDS Custom DB instance using AWS Systems Manager" section of the Amazon RDS User Guide.
2. On the `standby_instance`, add n+1 standby redo log files using the following steps:
   a. If you have *n* number of redo logs on the primary, add n+1 standby redo logs on the primary and standby for real-time replication.
   b. Switch to the `rdsdb` user: `$ sudo su - rdsdb`.
   c. Start SQL*Plus: `$ sqlplus / as sysdba`.
   d. Stop MRP: `SQL> alter database recover managed standby database cancel;`.
   e. Add the standby redo log files as follows:

```
SQL> ALTER DATABASE ADD STANDBY LOGFILE THREAD 1 ('slog1.rdo')
SIZE 128M;
SQL> ALTER DATABASE ADD STANDBY LOGFILE THREAD 1 ('slog2.rdo')
SIZE 128M;
SQL> ALTER DATABASE ADD STANDBY LOGFILE THREAD 1 ('slog3.rdo')
SIZE 128M;
SQL> ALTER DATABASE ADD STANDBY LOGFILE THREAD 1 ('slog4.rdo')
SIZE 128M;
SQL> ALTER DATABASE ADD STANDBY LOGFILE THREAD 1 ('slog5.rdo')
SIZE 128M;
```

f. Restart MRP: `SQL> alter database recover managed standby database disconnect from session;`

3. On the `primary_instance`, add the standby redo log files:
   a. Switch to the `rdsdb` user: `$ sudo su - rdsdb`.
   b. Start SQL*Plus: `$ sqlplus / as sysdba`.
   c. Add the standby redo log.

```
SQL> ALTER DATABASE ADD STANDBY LOGFILE THREAD 1 ('slog1.rdo')
SIZE 128M;
SQL> ALTER DATABASE ADD STANDBY LOGFILE THREAD 1 ('slog2.rdo')
SIZE 128M;
SQL> ALTER DATABASE ADD STANDBY LOGFILE THREAD 1 ('slog3.rdo')
SIZE 128M;
SQL> ALTER DATABASE ADD STANDBY LOGFILE THREAD 1 ('slog4.rdo')
SIZE 128M;
SQL> ALTER DATABASE ADD STANDBY LOGFILE THREAD 1 ('slog5.rdo')
SIZE 128M;
```

4. Check the Oracle Data Guard configuration status, and wait for the configuration status to become successful.

```
$ sudo su - rdsdb
$ dgmgrl /
DGMGRL> SHOW CONFIGURATION VERBOSE;
```

a. **Note**: `Error: ORA-16664: unable to receive the result from a member` could appear right after the setup, and should be gone in a few minutes. If this error persists, please check the TNS entries following Step 6 in this guide.

5. Resume automation on both the `primary_instance` and the `standby_instance` using the instructions in the "Pausing and resuming RDS Custom automation" section in the Amazon RDS User Guide.

(Optional) Change the replication mode to use maxavailability mode

1. Connect to the `primary_instance` by following the steps in the "Connecting to your RDS Custom DB instance using AWS Systems Manager" section of the Amazon RDS User Guide.
2. Switch to the `rdsdb` user: `$ sudo su - rdsdb`.
3. Start the Oracle Data Guard software by running the following commands:

```
dgmgrl /
DGMGRL> SHOW CONFIGURATION VERBOSE;
DGMGRL> EDIT DATABASE ORCL_B SET PROPERTY LOGXPTMODE='SYNC';
DGMGRL> EDIT DATABASE ORCL_A SET PROPERTY LOGXPTMODE='SYNC';
DGMGRL> EDIT CONFIGURATION SET PROTECTION MODE AS
MAXAVAILABILITY;
```

(Optional) Try manual switchover and failover features

After you have the HA instances configured, you could optionally follow these steps to perform a manual switchover and failover to test these features:

Manual switchover

1. Connect to the `primary_instance` and the `standby_instance` in separate browser instances by following the instructions in the "Connecting to your RDS Custom DB instance using AWS Systems Manager" section of the Amazon RDS User Guide.
2. Switch to the `rdsdb` user: `$ sudo su - rdsdb`.
3. Start the Oracle Data Guard software by running: `dgmgrl /`.

4. Connect to the `RDS_DATAGUARD` user that was created in Step 10: Configure Oracle Data Guard with the password you chose in that step. `ORCL_A` in the command is the `db_unique_name` for the `primary_instance`.

```
DGMGRL> CONNECT RDS_DATAGUARD/<password>@ORCL_A
```

5. Validate the databases to make sure they are ready for switchover. `ORCL_A` in the command is the `db_unique_name` for the `primary_instance`. `ORCL_B` in the command is the `db_unique_name` for the `standby_instance`.

```
DGMGRL> VALIDATE DATABASE ORCL_A
DGMGRL> VALIDATE DATABASE ORCL_B
```

   a. A healthy output should contain

   b. Ready for Switchover: Yes

6. Perform the switchover.

```
DGMGRL> SWITCHOVER TO ORCL_B
```

   a. A successful switchover will have an output similar to the following:

```
Performing switchover NOW, please wait...
Operation requires a connection to database "orcl_b"
Connecting ...
Connected to "ORCL_B"
Connected as SYSDG.
New primary database "orcl_b" is opening...
Operation requires start up of instance "ORCL" on database
"orcl_a"
Starting instance "ORCL"...
Connected to an idle instance.
ORACLE instance started.
Connected to "ORCL_A"
Database mounted.
Connected to "ORCL_A"
Switchover succeeded, new primary is "orcl_b"
```

7. Check the Oracle Data Guard configuration to make sure the role has changed and the configuration is still `SUCCESS`.

```
DGMGRL> SHOW CONFIGURATION VERBOSE
```

Manual failover

To perform the manual failover, you will need to have the Flashback Database turned on and initiate a database restart for both the `primary_instance` and the `standby_instance`.

1. Connect to the `primary_instance` and the `standby_instance` in separate browser instances by following the instructions in the "Connecting to your RDS Custom DB instance using AWS Systems Manager" section of the Amazon RDS User Guide.
2. On the `standby_instance`, turn on Flashback Database. Restart the database as required.
   a. Switch to `rdsdb` user: `$ sudo su - rdsdb`.
   b. Create the path for flashback files if it is not already created (replace `ORCL_B` with the unique name you use for the `standby_instance`)

```
$ mkdir -p /rdsdbdata/db/ORCL_B/flashback
```

   a. Start SQL*Plus: `$ sqlplus / as sysdba`.
      b. Turn on Flashback Database (replace `ORCL_B` with the unique name you use for the `standby_instance`).

```
SQL> alter system set
db_recovery_file_dest='/rdsdbdata/db/ORCL_B/flashback'
scope=both;
SQL> alter system set db_recovery_file_dest_size=10g scope=both;
# you can choose a different suitable size
SQL> alter database recover managed standby database cancel;
SQL> alter database flashback on;
SQL> alter database recover managed standby database disconnect
from session;
```

3. On the `primary_instance`, turn on Flashback Database. Restarting the database is required.
   a. Switch to `rdsdb` user: `$ sudo su - rdsdb`.

b. Create the path for flashback files if it is not already created (replace `ORCL_A` with the unique name you use for the `primary_instance`)

```
$ mkdir -p /rdsdbdata/db/ORCL_A/flashback
```

c. Start SQL*Plus: `$ sqlplus / as sysdba`.

d. Turn on Flashback Database (replace `orcl_a` with the unique name you use for the `primary_instance`).

```
SQL> alter system set
db_recovery_file_dest='/rdsdbdata/db/ORCL_A/flashback'
scope=both;
SQL> alter system set db_recovery_file_dest_size=10g scope=both;
# you can choose a different suitable size
SQL> alter database flashback on;
```

4. On the `standby_instance`, switch to the `rdsdb` user: `$ sudo su - rdsdb`.
5. Start the Oracle Data Guard software by running: `dgmgrl /`.
6. Connect to the `RDS_DATAGUARD` user that was created in step 10.2 with the password you chose in that step. `ORCL_B` in the command is the `db_unique_name` for the `standby_instance`.

```
DGMGRL> CONNECT RDS_DATAGUARD/<password>@ORCL_B
```

7. Validate the database of the `standby_instance` to make sure it is ready for failover. `ORCL_B` in the command is the `db_unique_name` for the `standby_instance`.

```
DGMGRL> VALIDATE DATABASE ORCL_B
```

a. A healthy output should contain:

```
Ready for Failover: Yes (Primary Running)
```

8. Perform the failover.

```
DGMGRL> FAILOVER TO ORCL_B
```

a. A successful failover will have an output similar to the following:

```
Performing failover NOW, please wait...
Failover succeeded, new primary is "orcl_b"
```

9. Check the Oracle Data Guard configuration with the following:

```
DGMGRL> SHOW CONFIGURATION VERBOSE
```

a. The role transition should be finished, but you will see an error message for the new standby instance similar to the following:

```
orcl_a - Physical standby database (disabled)
ORA-16661: the standby database needs to be reinstated
```

b. Perform the reinstate:

```
DGMGRL> REINSTATE DATABASE ORCL_A
```

A successful reinstate will have an output similar to the following:

```
REINSTATE DATABASE ORCL_A
Reinstating database "orcl_a", please wait...
Operation requires shut down of instance "ORCL" on database
"orcl_a"
Shutting down instance "ORCL"...
Connected to "ORCL_A"
ORACLE instance shut down.
Operation requires start up of instance "ORCL" on database
"orcl_a"
Starting instance "ORCL"...
Connected to an idle instance.
ORACLE instance started.
Connected to "ORCL_A"
Database mounted.
Connected to "ORCL_A"
Continuing to reinstate database "orcl_a" ...
Reinstatement of database "orcl_a" succeeded
```

c. Check the Oracle Data Guard configuration to make sure the role has changed and the configuration is still `SUCCESS`.

```
DGMGRL> SHOW CONFIGURATION VERBOSE
```

(Optional) Configure the FSFO to enable automatic failover

This task is only required if you want to enable the automatic failover with FSFO for the HA instances.

Configure the FSFO

1. Launch another Custom instance to serve as the observer host in the same subnet group. This step is similar to the step where you launched the `standby_instance` in the preceding Procedural sections. Launching the instance in a different availability zone from the primary and standby instances is highly recommended.

> **Note**: You can also use an on-premises instance with either the Oracle Client Administrator software or the full Oracle Database software stack.

2. Connect to the `primary_instance` and the `standby_instance` in separate browser instances by following the instructions in the "Connecting to your RDS Custom DB instance using AWS Systems Manager" section of the Amazon RDS User Guide.
3. On the `standby_instance`, turn on the Flashback Database.
   a. Switch to `rdsdb` user: `$ sudo su - rdsdb`.
   b. Create the path for the flashback files if it is not already created (replace `ORCL_B` with the unique name you use for the `standby_instance`).

```
$ mkdir -p /rdsdbdata/db/ORCL_B/flashback
```

   c. Start SQL*Plus: `$ sqlplus / as sysdba`.
   d. Turn on Flashback Database (replace `ORCL_B` with the unique name you use for the `standby_instance`).

```
SQL> alter system set
db_recovery_file_dest='/rdsdbdata/db/ORCL_B/flashback'
scope=both;
SQL> alter system set db_recovery_file_dest_size=10g scope=both;
# you can choose a different suitable size
SQL> alter database recover managed standby database cancel;
SQL> alter database flashback on;
SQL> alter database recover managed standby database disconnect
from session;
```

4. On the `primary_instance`, turn on Flashback Database.
   a. Switch to `rdsdb` user: `$ sudo su - rdsdb`.
   b. Create the path for flashback files if it is not already created (replace `ORCL_A` with the unique name you use for the `primary_instance`).

```
$ mkdir -p /rdsdbdata/db/ORCL_A/flashback
```

   c. Start SQL*Plus: `$ sqlplus / as sysdba`.
   d. Turn on Flashback Database (replace `ORCL_A` with the unique name you use for the `primary_instance`).

```
SQL> alter system set
db_recovery_file_dest='/rdsdbdata/db/ORCL_A/flashback'
scope=both;
SQL> alter system set db_recovery_file_dest_size=10g scope=both;
# you can choose a different suitable size
SQL> alter database flashback on;
```

5. On the `primary_instance`:
   a. Switch to the `rdsdb` user: `$ sudo su - rdsdb`.
   b. Start the Oracle Data Guard software and run the following commands (replace `ORCL_A` and `ORCL_B` with the DB unique names you use for the `primary_instance` and the `standby_instance`):

```
dgmgrl /
DGMGRL> EDIT DATABASE 'ORCL_B' SET PROPERTY
FASTSTARTFAILOVERTARGET='ORCL_A';
DGMGRL> EDIT DATABASE 'ORCL_A' SET PROPERTY
FASTSTARTFAILOVERTARGET='ORCL_B';
DGMGRL> ENABLE FAST_START FAILOVER;
```

6. Connect to the observer host. Follow the instructions in the “Connecting to your RDS Custom DB instance using AWS Systems Manager” section of the Amazon RDS User Guide.
7. On the observer, copy the `/rdsdbbin/oracle/network/admin/tnsnames.ora` file from the `primary_instance` either by `scp` or base64 encoding as in the example below:
   - Before executing the commands, switch to the rdsdb user: $ sudo su - rdsdb

```
(On primary_instance)
$ base64 /rdsdbdata/config/tnsnames.ora
<copy the exact output of that command>

(On observer host)
$ base64 -d > /rdsdbdata/config/tnsnames.ora
<paste the content copied above>
<press enter>
<press "ctrl + d">
```

8. Verify the networking connection and TNS entries by running the following commands:

```
$ tnsping ORCL_A
$ tnsping ORCL_B
```

9. Start the Oracle Data Guard software and start the observer.
   a. Switch to the `rdsdb` user: `$ sudo su - rdsdb`.
   b. Run this command: `$ nohup dgmgrl -silent sys/<sys_password>@orcl_a "start observer" &`.
10. Go to the `primary_instance` and switch to the `rdsdb` user: `$ sudo su - rdsdb`.
11. Start the Oracle Data Guard software by running: `dgmgrl /`.
12. Check the Oracle Data Guard configuration by running the following command:

```
DGMGRL> SHOW CONFIGURATION VERBOSE
```

   a. Wait until the following warning disappears: `Warning: ORA-16819: fast-start failover observer not started`.

Trigger the automatic failover

To trigger the failover, shut down the primary database.

1. On the `primary_instance`:
   a. Switch to `rdsdb` user: `$ sudo su - rdsdb`.
   b. Start SQL*Plus: `$ sqlplus / as sysdba`.
   c. Before you test the automatic failover with FSFO, make sure the Flashback Database has at least 30 minutes of history.

```
SQL> select (sysdate - oldest_flashback_time)*24*60 as history
from v$flashback_database_log;
```

   d. Shut down the primary database by running the following command:

```
SQL> shutdown abort
```

2. On observer host, the log is captured in `~/nohup.out`. The log should look something like the following:

```
[W000 2021-05-20T01:12:32.823+00:00] Try to connect to the
standby.
[W000 2021-05-20T01:12:32.823+00:00] Making a last connection
attempt to primary database before proceeding with Fast-Start
Failover.
[W000 2021-05-20T01:12:32.823+00:00] Check if the standby is
ready for failover.
[S005 2021-05-20T01:12:33.855+00:00] Fast-Start Failover
started...

2021-05-20T01:12:33.855+00:00
Initiating Fast-Start Failover to database "orcl_b"...
[S005 2021-05-20T01:12:33.855+00:00] Initiating Fast-start
Failover.
Performing failover NOW, please wait...
Failover succeeded, new primary is "orcl_b"
```

3. The output of the observer logs looks as follows:

```
[W000 2021-05-20T01:28:11.641+00:00] Connection to the new
standby restored!
[W000 2021-05-20T01:28:11.643+00:00] Failed to ping the new
standby.
```

   a. If the output indicates the old primary/new standby is down, start it as follows:

      i. On the new standby host, switch to `rdsdb` user: `$ sudo su - rdsdb`.
      ii. Start SQL*Plus:

```
$ sqlplus / as sysdba.

Connected to an idle instance.
```

```
SQL> startup mount
```

iii. After this, you will see the log on observer like below, which indicates the FSFO failover is successful:

```
[W000 2021-05-20T01:19:37.975+00:00] Connection to the primary
restored!
[W000 2021-05-20T01:19:37.975+00:00] Wait for new primary to be
ready to reinstate.
[W000 2021-05-20T01:19:38.978+00:00] New primary is now ready to
reinstate.
[W000 2021-05-20T01:19:39.978+00:00] Issuing REINSTATE command.

2021-05-20T01:19:39.978+00:00
Initiating reinstatement for database "orcl_a"...
Reinstating database "orcl_a", please wait...
[W000 2021-05-20T01:19:59.000+00:00] The standby orcl_a is ready
to be a FSFO target
Reinstatement of database "orcl_a" succeeded
```

4. Check the Oracle Data Guard configuration to make sure the role has changed and the configuration is still `SUCCESS`.

```
DGMGRL> SHOW CONFIGURATION VERBOSE
```

(Optional) Manual Promote a standby to a primary when only one standby instance is configured

Note: RDS Custom does not currently provide API based functionality to promote a read replica to Primary. But you can choose to perform manual promotion of read replica using following steps:

1. Pause automation on both the `primary_instance` and the `standby_instance` using the instructions in the “Pausing and resuming RDS Custom automation” section of the Amazon RDS User Guide. Pause the instances for as long as needed to complete the configuration.
2. Connect to `primary_instance` in one browser instance and `standby_instance` in another browser instance. Follow the instructions in “Connecting to your RDS Custom DB instance using AWS Systems Manager.” (`primary_instance: ORCL_A, standby_instance: ORCL_B`).

3. On `primary_instance`, switch to the `rdsdb` user: `$ sudo su - rdsdb`.
4. Start the Oracle Data Guard software by running: `$ dgmgrl /`.

```
DGMGRL> show configuration verbose;
DGMGRL> disable configuration;
DGMGRL> show configuration verbose;
DGMGRL> disable database "ORCL_B";
DGMGRL> remove configuration;
DGMGRL> show configuration verbose;
```

Output should be as follows:

```
ORA-16532: Oracle Data Guard broker configuration does not exist
```

5. Exit Oracle Data Guard software: `DGMGRL> exit;`
6. On `primary_instance`, Start SQL*Plus: `$ sqlplus / as sysdba`.
7. Reset broker setting and remove broker metadata files.

```
SQL> ALTER SYSTEM SET DG_BROKER_START=FALSE SCOPE=BOTH;
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='' SCOPE=BOTH;
SQL> ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='' SCOPE=BOTH;
SQL> SHOW PARAMETER DG_BROKER_CONFIG_FILE1
   NAME                          TYPE        VALUE
   ----------------------    ----------- -------------------------
   dg_broker_config_file1      string /rdsdbdata/config/dr1ORCL.dat
SQL> ALTER SYSTEM RESET DG_BROKER_CONFIG_FILE1;
SQL> !rm /rdsdbdata/config/dr1ORCL.dat --(Replace the value with
output of VALUE above, same to the similar commands below)
SQL> SHOW PARAMETER DG_BROKER_CONFIG_FILE2
NAME                           TYPE        VALUE
 ----------------------    ----------- -------------------------
dg_broker_config_file2       string /rdsdbdata/config/dr2ORCL.dat

SQL> ALTER SYSTEM RESET DG_BROKER_CONFIG_FILE2;
SQL> !rm /rdsdbdata/config/dr2ORCL.dat
```

8. On `standby_instance`, switch to the rdsdb user: `$ sudo su - rdsdb`.
9. Start SQL*Plus: `$ sqlplus / as sysdba`.
10. Reset broker setting and remove broker metadata files

```
SQL> ALTER SYSTEM SET DG_BROKER_START=FALSE SCOPE=BOTH;
SQL> ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='' SCOPE=BOTH;
SQL> SHOW PARAMETER DG_BROKER_CONFIG_FILE1
   NAME                           TYPE        VALUE
   ------------------------------ ----------- ------------------------
dg_broker_config_file1         string /rdsdbdata/config/dr1ORCL.dat

SQL> ALTER SYSTEM RESET DG_BROKER_CONFIG_FILE1;
SQL> !rm /rdsdbdata/config/dr1ORCL.dat
SQL> SHOW PARAMETER DG_BROKER_CONFIG_FILE2;
   NAME                           TYPE        VALUE
   ------------------------------ ----------- ------------------------
dg_broker_config_file2         string /rdsdbdata/config/dr2ORCL.dat

SQL> ALTER SYSTEM RESET DG_BROKER_CONFIG_FILE2;
SQL> !rm /rdsdbdata/config/dr2ORCL.dat
```

11. Converting the `standby_instance` to Primary:
12. On `standby_instance`, switch to the `rdsdb` user: `$ sudo su - rdsdb`.
13. Start SQL*Plus: `$ sqlplus / as sysdba`.

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH;
```

14. Validate that `standby_instance` is ready for role conversion by running following queries

```
SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE; --(Confirm result is
TO PRIMARY or SESSIONS ACTIVE);
```

15. Converting `standby_instance` to primary

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY WITH SESSION
SHUTDOWN;
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP;
```

16. Final Step is to Stop and Remove DG Listener from the `listener.ora` and remove the `tnsnames.ora` from the standby and primary host.

a. Stop Dataguard listener on `primary_instance` and `standby_instance` (for example, `L_ORCL_DG` is listener name for we created in step and should be changed accordingly.)

   On `primary_instance`:

   ```
   $ $ORACLE_HOME/bin/lsnrctl stop L_ORCL_DG;
   ```

   On `standby_instance`:

   ```
   $ $ORACLE_HOME/bin/lsnrctl stop L_ORCL_DG;
   ```

b. Modify the /rdsdbbin/oracle/network/admin/tnsnames.ora.

   On `primary_instance`:

   - Remove the entry for `standby_instance`. In the example, it is the entry start with "`ORCL_B = (`"

   On `standby_instance`:

   - Remove the entry for primary_instance. In the example, it is the entry start with "`ORCL_A = (`"

17. Resume automation on both the `primary_instance` and the `standby_instance` using the instructions in the "Pausing and resuming RDS Custom automation" section in the Amazon RDS User Guide.

(Optional) Manual promote a standby to a primary when multiple standby instances are configured

This section presents the manual promotion steps on the setup where you have more standby instances besides `primary_instance` and `standby_instance`. Following is an example, where you have two more standby instances whose `db_unique_name` is `ORCL_C` and `ORCL_D`. Those two instances will also be referred as `ORCL_C` and `ORCL_D`. The example promotes `ORCL_D` to primary instance.

1. Pause automation on both the `primary_instance` and the `standby_instance` using the instructions in the “Pausing and resuming RDS Custom automation” section of the Amazon RDS User Guide. Pause the instances for as long as needed to complete the configuration.
2. Connect to `primary_instance` in one browser instance and `ORCL_D` in another browser instance. Follow the instructions in the “Connecting to your RDS Custom DB instance using AWS Systems Manager” section of the Amazon RDS User Guide.
3. On `primary_instance`, switch to `rdsdb` user: `$ sudo su - rdsdb`.
4. Start SQL*Plus: `$ sqlplus / as sysdba`.
5. Change/Remove DG Configuration for database

```
$ dgmgrl /
DGMGRL> show configuration verbose;
DGMGRL> disable database "ORCL_D";
DGMGRL> remove database "ORCL_D";
DGMGRL> show configuration verbose;
```

`ORCL_D` is removed from the member list.

6. On `primary_instance`, switch to `rdsdb` user: `$ sudo su - rdsdb`.
7. Start SQL*Plus: `$ sqlplus / as sysdba`.
8. You will need to find the values for `LOG_ARCHIVE_DEST_n` where the standby instance (`ORCL_D`) being promoted is getting the archives from (for example, `LOG_ARCHIVE_DEST_4`).

```
SQL> SHOW PARAMETER LOG_ARCHIVE_DEST;
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_4='' SCOPE=BOTH;
```

9. On ORCL_D, switch to rdsdb user: $ sudo su - rdsdb.
10. Start SQL*Plus: $ sqlplus / as sysdba.

```
SQL> ALTER SYSTEM SET DG_BROKER_START=FALSE SCOPE=BOTH;
SQL> ALTER SYSTEM SET LOG_ARCHIVE_CONFIG='' SCOPE=BOTH;
SQL> SHOW PARAMETER DG_BROKER_CONFIG_FILE1
   NAME                           TYPE        VALUE
   ------------------------------ ----------- ------------------------
```

```
dg_broker_config_file1         string /rdsdbdata/config/dr1ORCL.dat
   SQL> ALTER SYSTEM RESET DG_BROKER_CONFIG_FILE1;
   SQL> !rm /rdsdbdata/config/dr1ORCL.dat
SQL> SHOW PARAMETER DG_BROKER_CONFIG_FILE2
   NAME                           TYPE        VALUE
   ------------------------------ ----------- ------------------------
dg_broker_config_file1         string /rdsdbdata/config/dr2ORCL.dat
   SQL> ALTER SYSTEM RESET DG_BROKER_CONFIG_FILE2;
   SQL> !rm /rdsdbdata/config/dr2ORCL.dat
```

11. Convert the Replica to Primary

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH;
```

12. Validate that `ORCL_D` is ready for role conversion.

```
SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE; --(Confirm result is
To PRIMARY or SESSIONS ACTIVE);
```

13. Converting read replica/standby to primary:

```
SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY WITH SESSION
SHUTDOWN;
SQL> SHUTDOWN IMMEDIATE;
SQL> STARTUP;
```

14. Stop and remove DG Listener from the `listener.ora` and `tnsnames.ora` from `ORCL_D` and then remove TNS entry if `ORCL_D` from the `primary_instance`, `standby_instance`, and `ORCL_C`.

    a. On `ORCL_D`, stop the listener:

    ```
    $ $ORACLE_HOME/bin/lsnrctl stop L_ORCL_DG;
    ```

    b. On `ORCL_D`, make changes to `tnsnames.ora` to remove the entries for the rest of the instances, and only keep the entry starts with "`ORCL_D =(`".

    c. On `primary_instance`, `standby_instance`, and `ORCL_C`, remove the entry start with "`ORCL_D = (`".

15. Resume automation on both the `primary_instance` and the `standby_instance` using the instructions in the “Pausing and resuming RDS Custom automation” section in the Amazon RDS User Guide.

Security

Considerations for security groups

Configure VPC:

To connect to your DB instance, the DB must be associated with a security group that contains the necessary IP addresses and network configuration. If your DB instance was assigned to a default, non-configured security group when it was created, the firewall prevents connections.

To create a new security group, you’ll first want to identify the Amazon EC2 platform for your DB instance. To determine your platform, see Determining whether you are using the EC2-VPC or EC2-Classic platform.

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. For information about creating a new security group, see Controlling access with security groups.

(Optional) Configure a VPN Tunnel to encrypt the communication

Only perform this task if you want the traffic among the Oracle Data Guard instances to be encrypted.

1. Allowlist the security groups for both the `primary_instance` and `standby_instance` using the following rules:

```
ACTION FLOW SOURCE PROTO PORT

ALLOW ingress this-SG 50 (ESP) all (N/A)
ALLOW egress this-SG 50 (ESP) all (N/A)

ALLOW ingress this-SG 17 (UDP) 500 (IKE)
ALLOW egress this-SG 17 (UDP) 500 (IKE)
```

2. Connect to the `primary_instance` and the `standby_instance` in separate browser instances by following the instructions in the "Connecting to your RDS Custom DB instance using AWS Systems Manager" section of the Amazon RDS User Guide.
3. Switch to the root user: `$ sudo su - root`.
4. Run the following commands on both the `primary_instance` and the `standby_instance` to install IPSec and initialize the NSS database under the user `root`.

```
$ yum install libreswan -y # This should also be built in the
AMI, only run this if it is missing
$ ipsec initnss --nssdir /etc/ipsec.d
```

5. Generate RSA keys as follows:
   a. On the `primary_instance`, generate the keys.

```
$ ipsec newhostkey --output /etc/ipsec.secrets
Generated RSA key pair with CKAID
d498b6538bc1b18aa3d740151e0b5c6d389b8af1 was stored in the NSS
database
```

   b. Obtain the public key, which you need to create configuration. In the following example, the `primary_instance` is `left` because in IPsec parlance, `left` always refers to the device you are currently configuring, and `right` refers to the device at the other end of the tunnel.

```
$ ipsec showhostkey --left --ckaid
d498b6538bc1b18aa3d740151e0b5c6d389b8af1
# rsakey AwEAAcKdo
leftrsasigkey=0sAwEAAcKdo6n...[truncated]
```

   c. On the `standby_instance`, repeat the generation of keys for the `standby_instance`.

```
$ ipsec newhostkey --output /etc/ipsec.secrets
Generated RSA key pair with CKAID
35b0bbdbf9506ca1c2ab745314b33f69bc65a2ec was stored in the NSS
database
```

d. Obtain the public key, which you need to create the configuration. The following uses the parameter `right`.

```
$ ipsec showhostkey --right --ckaid
35b0bbdbf9506ca1c2ab745314b33f69bc65a2ec
# rsakey AwEAAbx8m
rightrsasigkey=0sAwEAAbx8m6...[truncated]
```

6. Based on the keys, generate the configuration. The configuration is identical for both the `primary_instance` and the `standby_instance`. The `<primary_instance_ipv4>` and `<standby_instance_ipv4>` values are the private IP addresses you obtained in Step 4: Note the private IP addresses of the Amazon Elastic Compute Cloud (Amazon EC2) instances.

```
conn custom-db-tunnel
    type=transport
    auto=add
    authby=rsasig
    left=<primary_instance_ipv4>
    leftrsasigkey=0sAwEAAcKdo6n...[truncated]
    right=<standby_instance_ipv4>
    rightrsasigkey=0sAwEAAbx8m6...[truncated]
```

7. On both the `primary_instance` and the `standby_instance`, save the preceding configuration to `/etc/ipsec.d/custom-fb-tunnel.conf`.
8. On both the `primary_instance` and the `standby_instance`, start the IPSec daemon on both hosts:

```
$ ipsec setup start
```

9. Start the tunnel on either the `primary_instance` or the `standby_instance`. The output should show `IPsec SA established transport mode`, which means the tunnel is set up bidirectionally. The output should look something like the following:

```
$ ipsec auto --up custom-db-tunnel
002 "custom-db-tunnel" #1: initiating Main Mode
104 "custom-db-tunnel" #1: STATE_MAIN_I1: initiate
106 "custom-db-tunnel" #1: STATE_MAIN_I2: sent MI2, expecting
MR2
108 "custom-db-tunnel" #1: STATE_MAIN_I3: sent MI3, expecting
MR3
002 "custom-db-tunnel" #1: Peer ID is ID_IPV4_ADDR:
'172.31.37.170'
003 "custom-db-tunnel" #1: Authenticated using RSA
004 "custom-db-tunnel" #1: STATE_MAIN_I4: ISAKMP SA established
{auth=RSA_SIG cipher=aes_256 integ=sha2_256 group=MODP2048}
002 "custom-db-tunnel" #2: initiating Quick Mode
RSASIG+ENCRYPT+PFS+UP+IKEV1_ALLOW+IKEV2_ALLOW+SAREF_TRACK+IKE_FR
AG_ALLOW+ESN_NO {using isakmp#1 msgid:46308d05 proposal=defaults
pfsgroup=MODP2048}
117 "custom-db-tunnel" #2: STATE_QUICK_I1: initiate
004 "custom-db-tunnel" #2: STATE_QUICK_I2: sent QI2, IPsec SA
established transport mode {ESP=>0xf966b288 <0x510eb36a
xfrm=AES_CBC_128-HMAC_SHA1_96 NATOA=none NATD=none DPD=passive}
```

Conclusion

This guide presents one of the approaches to configure a database for HA on your RDS Custom Oracle instances using Oracle Data Guard. It also shares some of the best practices to configure data guard to achieve HA.

Contributors

Contributors to this document include:

- Jeff (Wenjie) Zhou, Senior Software Engineer, Amazon Web Services – RDS Oracle
- Nitin Saxena, Senior Database Engineer, Amazon Web Services – RDS Oracle

Document revisions

Date	Description
October 2021	First publication