Oracle Maximum
Availability Architecture

Disaster Recovery to the Oracle Cloud

Production on Premises, DR in the Cloud

ORACLE WHITE PAPER | APRIL 2016

Table of Contents

Introduction

Oracle’s Maximum Availability Architecture (Oracle MAA) is the best practices blueprint for data protection and availability for Oracle databases deployed on private, public or hybrid clouds. Data Guard and Active Data Guard provide disaster recovery (DR) for databases with recovery time objectives (RTOs) that cannot be met by restoring from backup. Customers use these solutions to deploy one or more synchronized replicas (standby databases) of a production database (the primary database) in physically separate locations to provide high availability, comprehensive data protection, and disaster recovery for mission critical data.

An effective disaster recovery plan can be costly due to the need to establish, equip and manage a remote data center. The Oracle Cloud offers a great alternative for hosting standby databases for customers who do not have a DR site or who prefer not to deal with the cost or complexity of managing a remote data center. Existing production databases remain on-premises and standby databases used for DR are deployed on the Oracle Cloud. This mode of deployment is commonly referred to as a hybrid cloud implementation.

Customers may choose to deploy either a Data Guard or an Active Data Guard standby on the cloud depending upon their requirements. While there are some unique considerations to a hybrid cloud DR configuration, it follows the same Oracle MAA best practices as with any Data Guard deployment. This Oracle MAA blueprint details Oracle MAA Best Practices and provides a procedural overview for deploying DR on the Oracle Cloud using Database as a Service. This paper is intended for a technical audience having knowledge of Oracle Database, Data Guard or Active Data Guard, and Oracle Database backup and recovery. This paper also assumes a basic understanding of services offered on the Oracle Cloud[1].

1 https://cloud.oracle.com/home

Disaster Recovery to the Cloud with Data Guard and Active Data Guard

The Oracle Cloud[2] offers an extensive set of cloud services tailored to specific customer requirements: Platform as a Service (PaaS), Infrastructure as a Service (IaaS), and Software as a Service (SaaS). Disaster Recovery (DR) for on-premises systems is deployed using the Oracle Database Cloud Service[3] (PaaS).

There are two options for DR to the cloud using Oracle Database Cloud Services:

- » Data Guard utilizing Enterprise Edition Service or High Performance Service.
- » Active Data Guard utilizing the Extreme Performance Service or Exadata Service.

Data Guard is included in Oracle Database Enterprise Edition (no separate license is required for on-premises systems) and is supported by all editions of Oracle Database Cloud Services (Enterprise, High Performance and Extreme Performance).

Active Data Guard extends Data Guard capabilities by providing advanced features for data protection and availability as well as offloading read-only workload and backups from a production database. Active Data Guard is included in the Extreme Performance Database Cloud Service and Exadata Service. When used in a hybrid configuration, Active Data Guard must also be licensed for the on-premises system. Refer to Oracle software license documentation[4] for more information on capabilities licensed with Active Data Guard.

Unless otherwise noted, the procedure explained in this paper applies equally to Data Guard and Active Data. For more information on Data Guard and Active Data Guard please refer to the Data Guard home page on the Oracle Technology Network and the Active Data Guard white paper[5].

Note: Data Guard may also be deployed using the Oracle Database Cloud Service - Virtual Image service level. The Virtual Image service level while suited for dev or test instances is not recommended for DR for on-premises production databases due to the reduced level of automation and tooling included with this basic service.

Enabling DR on the Oracle Cloud

Enabling DR on the cloud requires instantiation of a Data Guard standby database in the Oracle Database Cloud Service. Once instantiated, Data Guard maintains synchronization between the primary database on premises and the standby database in the cloud.

The Oracle Cloud provides all backend infrastructure and capabilities required for disaster recovery should the customer's on-premises database become unavailable for any reason. This includes:

1. Ability to monitor the standby database and alert on major issues. See Appendix A.
2. Ability to activate the standby to validate DR readiness and then convert it back to a synchronized standby.
3. Utilization of the same Oracle MAA best practices as on-premises deployment. Use of additional Oracle MAA best practices specific to hybrid cloud deployments that are specified in this paper.

2 https://www.oracle.com/cloud/cloud-summary.html

3 https://cloud.oracle.com/database

4 https://docs.oracle.com/database/121/DBLIC/options.htm#DBLIC141

5 http://www.oracle.com/technetwork/database/availability/active-data-guard-wp-12c-1896127.pdf

4. Ability to switchover (planned event) or failover (unplanned event) production to the standby database in the cloud during planned maintenance or unplanned outages. Once the failed on-premises database is repaired, the ability to automatically resynchronize it with the new production database in the cloud and then switch production back to the on-premises database.

5. Ability to failover database connections from an on-premises application tier to a new primary database in the Oracle Cloud following a Data Guard switchover or failover.

6. Ability to failover both the application and database tiers to the Oracle Cloud to enable production applications to fully run in the Oracle Cloud when there is a complete site outage.

7. Flexibility for a standby database in the Oracle Cloud to support additional use cases beyond disaster recovery, including: offloading read-only production workloads to the cloud, development and test, source for thin-provisioned database clones, and offloading backups to the cloud, as shown in Figure 1.

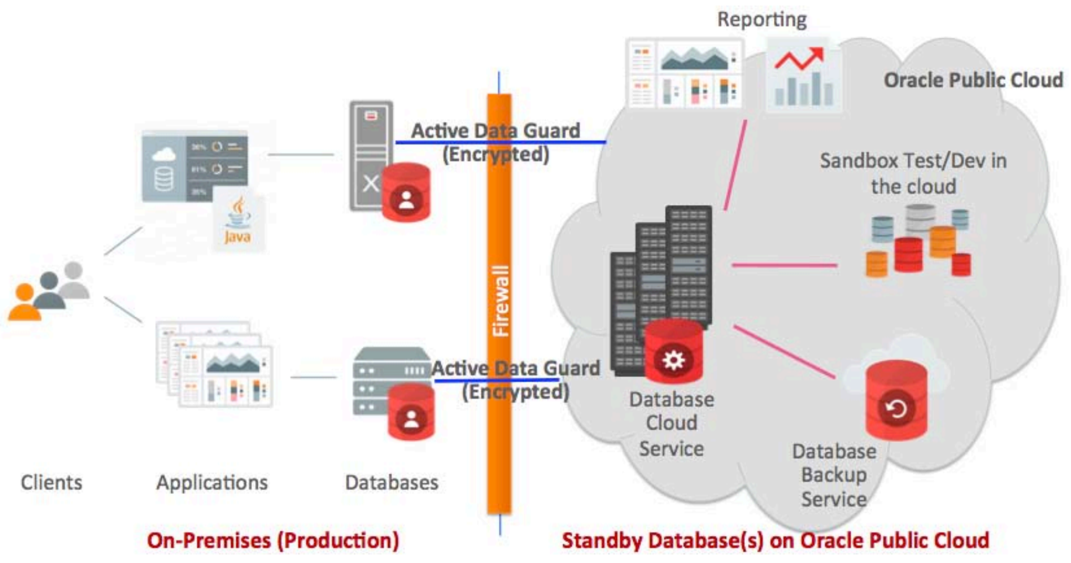


Figure 1: Hybrid Cloud: Disaster Recovery

Service Level Requirements

Hybrid cloud deployments are by definition user-managed environments. The administrator must determine service level expectations for availability, data protection, and performance that are practical for a given configuration and application. Service Levels must be established for each of three dimensions relevant to disaster recovery that are applicable to any Data Guard configuration:

» **Availability**: Recovery Time Objective (RTO) describes the maximum acceptable downtime should an outage occur. This includes time required to detect the outage and to failover both the database and application connections so that service is resumed.

» **Data Protection**: Recovery Point Objective (RPO) describes the maximum amount of data loss that can be tolerated. Achieving a desired RPO depends upon:
  » Available bandwidth relative to network volume.
  » The ability of the network to provide reliable, uninterrupted transmission.

  - The Data Guard transport method used: either asynchronous for near-zero data loss protection, or synchronous for zero data loss protection.

- **Performance**: Database response time may be different after failover if less capacity – compute, memory, I/O, etc, are provisioned at the standby system than in the on-premises production system. This occurs when administrators purposefully under-configure standby resources to reduce cost; accepting reduced service level while in DR mode. MAA best practices recommend configuring symmetrical capacity at both primary and standby so there is no change in response time after failover. Rapid provisioning available with the cloud can enable a middle ground where there is less capacity deployed steady-state, but the new primary is rapidly scaled-up should a failover be required.

Note: Independent of the service levels related to DR, all standby database instances created in the Oracle cloud conform to the service descriptions defined by the applicable Database Cloud Service[6].

Security Requirements

Oracle MAA best practice recommends using Oracle Transparent Data Encryption (TDE) to encrypt primary and standby databases at rest. Conversion to TDE enables automatic encryption at rest for all DATA/INDEX tablespaces and encryption-in-flight of user data redo changes during replication to the cloud. Oracle Net encryption is also required for encryption-in-flight for other redo changes that are not encrypted by TDE (e.g. data from unencrypted tablespaces such as SYSTEM and SYSAUX).

While strongly recommended for enhanced security and required for production databases initially deployed on the Oracle Cloud, using TDE to encrypt data at rest is not mandatory for hybrid cloud deployments used for disaster recovery. However, Oracle Net encryption is still required for encryption-in-flight of all redo during replication. This provides customers with on-premises production databases in a hybrid cloud configuration the option of deciding whether or not to use TDE depending upon their requirements.

Note: Data Guard and Active Data Guard use redo-based replication – a process that transmits redo generated by a primary database and applies those changes to a standby database using continuous media recovery. This means that primary and standby databases are block for block identical copies of each other. Using TDE to encrypt a standby database on the cloud also requires that the on-premises primary database be encrypted with TDE.

Using TDE to protect data is an important part of improving the security of the system. Users should however be aware of certain considerations when using any encryption solution, including:

- **Additional CPU overhead**: Encryption requires additional CPU cycles to calculate encrypted and decrypted values. TDE, however, is optimized to minimize the overhead by taking advantage of database caching capabilities and leveraging hardware acceleration for AES on Intel and SPARC CPUs. Most TDE users see little performance impact on their production systems after enabling TDE. If performance overhead is a concern, please refer to the Oracle Database Advanced Security Guide[7].
- **Lower data compression**: Encrypted data compresses poorly because it must reveal no information about the original plain text data. Thus, any compression applied to TDE encrypted data will have low compression ratios. Hence, when TDE encryption is used, it is not recommended to use with redo transport compression. However,

6 http://www.oracle.com/us/corporate/contracts/paas-iaas-public-cloud-2140609.pdf

7 https://docs.oracle.com/database/121/ASOAG/asotrans_faq.htm#ASOAG10544

when TDE is used in conjunction with Oracle database compression technologies such as Advanced Compression or Hybrid Columnar Compression, compression is performed before the encryption occurs, and the benefits of compression and encryption are both achieved.

» **Key management**: Encryption is only as strong as the key used to encrypt. Furthermore, the loss of the encryption key is tantamount to losing all data protected by that key. If encryption is enabled on a few databases, keeping track of the key and its lifecycle is relatively easy. As the number of encrypted databases grows, managing keys becomes an increasingly difficult problem. For users with large number of encrypted databases, it is recommended that Oracle Key Vault[8] on-premise be used to store and manage TDE master keys.

Database, Operating Environment and Prerequisites

Prerequisites for hybrid cloud DR configurations are described in the support matrix in Table 1.

	On-Premises	Database Cloud Service
Operating System	Linux, Windows or Solaris X86 (My Oracle Support Note 413484.1 for Data Guard cross-platform compatibility)	Oracle Enterprise Linux (64-bit)
Oracle Database*	Oracle Database Enterprise Edition 11.2.0.4 (64-bit) (or) Oracle Database Enterprise Edition 12.1.0.2 (64-bit)	**Data Guard**: Database as a Service (or) Virtual Image: Enterprise, High Performance and Extreme Performance editions **Active Data Guard**: Database as a Service (or) Virtual Image: Extreme Performance Edition (or) Exadata Cloud Service
RAC	RAC or non-RAC	RAC** or non-RAC[9]
Physical Vs Virtual	Physical or Virtual	Virtual
Database Size	Any size	DBaaS: 2.3TB with backups *** DBaaS: 5TB without backups *** Exadata Service: Any size
Encryption	Optional. If you want the standby in the cloud to be encrypted at rest, then the primary must be encrypted using TDE.	Optional. If the primary is encrypted, standby is also encrypted.

Table 1: Support Matrix

* Oracle Database version on primary and standby databases must match except when using standby first patching or database rolling upgrades where a standby database may be at a higher version than the primary database.

** Subject to availability. Please refer to Oracle Database Cloud Service documentation for the latest information.

*** Subject to change. Please refer to Oracle Database Cloud Service documentation for the latest information.

Data transfers from on-premises to Oracle Cloud use the public network or the high bandwidth option provided by Oracle FastConnect - Partner Edition. A Virtual Private Network (VPN) option is also available for Exadata Cloud

8 http://www.oracle.com/us/products/database/security/key-vault/overview/index.html

9 https://cloud.oracle.com/database

(database tier) and Dedicated Compute (application tier). Additional network and VPN options are planned for future availability. Refer to Oracle Network Cloud Service[10] for the latest information.

Oracle FastConnect - Partner Edition requires the customer datacenter to be co-located in an Equinix datacenter

Deployment Process

Deployment of DR on the cloud involves the following 8 steps.

1. Subscribe to Oracle Database Cloud Service
2. Create an Oracle instance
3. Configure Network
4. Encrypt Primary Database (Optional)
5. Instantiate Data Guard Standby
6. Perform Data Guard health check
7. Enable Runtime Monitoring
8. Enable Redo Transport Compression (Optional)

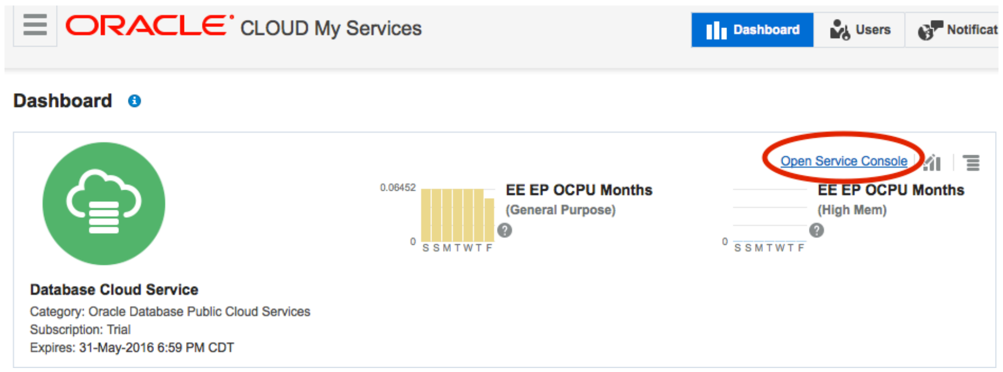
1. Subscribe to Oracle Database Cloud Service

Begin by subscribing to the Oracle Database Cloud Service[11]. This is usually offered as pre-paid subscription for a specific duration. Refer to the DBaaS cloud documentation for subscription, activation and service creation details[12].

2. Create an Oracle Instance

After activating the service you can create a database instance.

» Sign in to https://cloud.oracle.com . From *My Services* page, choose the appropriate Data Center / Region. Click on the *Open Service Console* for the "Database Cloud Service", see figure 2.



Database Cloud Service
Category: Oracle Database Public Cloud Services
Subscription: Trial
Expires: 31-May-2016 6:59 PM CDT

Figure 2: Dashboard

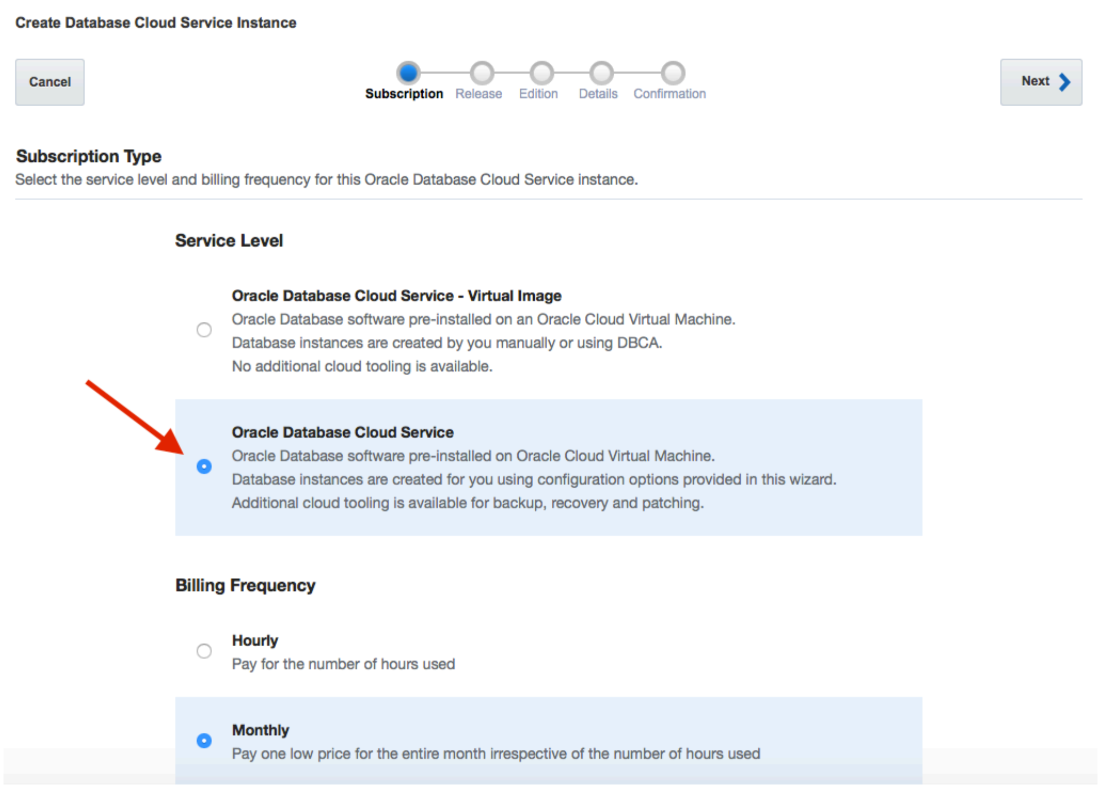
10 https://cloud.oracle.com/en_US/network

11 http://docs.oracle.com/cloud/latest/dbcs_dbaas/index.html

12 http://docs.oracle.com/cloud/latest/dbcs_dbaas/CSDBI/GUID-1A380E9C-6DE2-4042-8A31-B43A0081B194.htm#CSDBI3312

» The next page displays all the configured database services. Click "*Create Service*". Choose Oracle Database Cloud Service – Figure 3. This is the recommended level of service for DR for production databases.

Note: The Database Cloud Service - Virtual Image service level may also be selected but is generally recommended for use by development and test databases.



Subscription Type

Select the service level and billing frequency for this Oracle Database Cloud Service instance.

Service Level

( ) **Oracle Database Cloud Service - Virtual Image**
Oracle Database software pre-installed on an Oracle Cloud Virtual Machine.
Database instances are created by you manually or using DBCA.
No additional cloud tooling is available.

[✓ **Oracle Database Cloud Service**
Oracle Database software pre-installed on Oracle Cloud Virtual Machine.
Database instances are created for you using configuration options provided in this wizard.
Additional cloud tooling is available for backup, recovery and patching.]

Billing Frequency

( ) **Hourly**
Pay for the number of hours used

[✓ **Monthly**
Pay one low price for the entire month irrespective of the number of hours used]

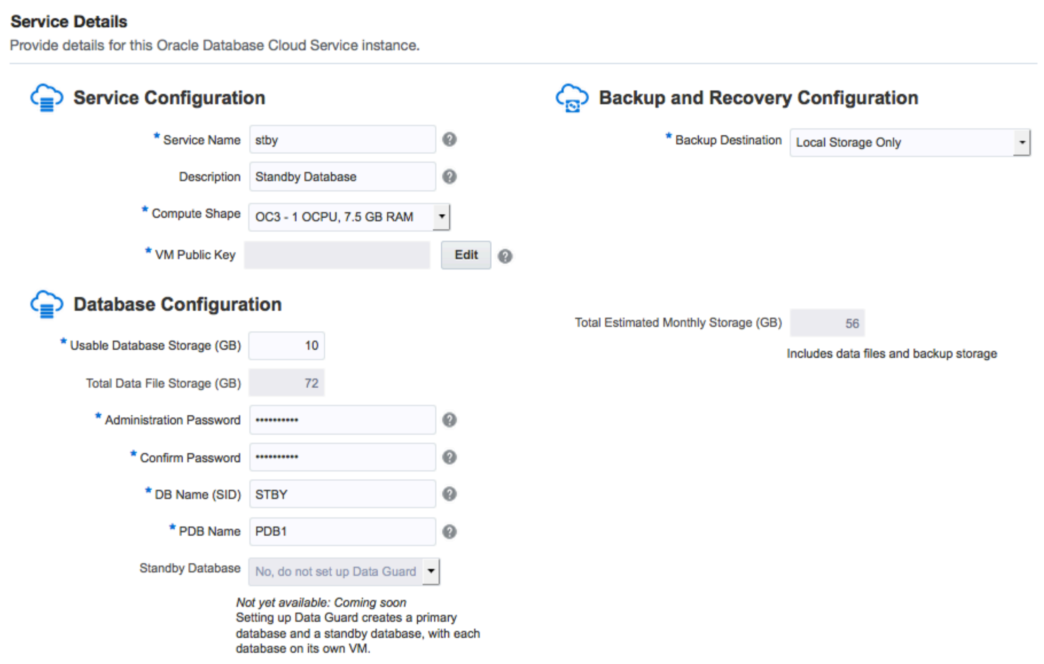
Figure 3: Subscription

» Click *Next* after choosing your desired service.

» In the "Release" screen, choose either 11.2.0.4 or 12.1.0.2 depending on your primary site's instance. Click *Next*.

» In the "Edition" screen, choose Enterprise Edition – Extreme Performance. Click *Next*.

» In the "Service Details" screen, enter the service name, choose compute shape, and enter the VM Public Key (you can generate using *ssh-keygen* utility, for example). See Figure 4.



Service Details

Provide details for this Oracle Database Cloud Service instance.

Service Configuration

* Service Name	stby
Description	Standby Database
* Compute Shape	OC3 - 1 OCPU, 7.5 GB RAM
* VM Public Key	

Backup and Recovery Configuration

* Backup Destination	Local Storage Only
Total Estimated Monthly Storage (GB)	56

Includes data files and backup storage

Database Configuration

* Usable Database Storage (GB)	10
Total Data File Storage (GB)	72
* Administration Password	••••••••••
* Confirm Password	••••••••••
* DB Name (SID)	STBY
* PDB Name	PDB1
Standby Database	No, do not set up Data Guard

Not yet available: Coming soon
Setting up Data Guard creates a primary database and a standby database, with each database on its own VM.

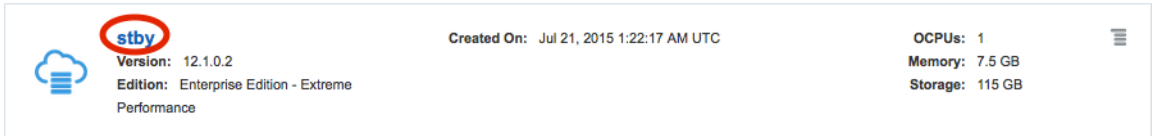
Figure 4: Service Details

Note: In order to retain the auto-generated scripts and configuration in the cloud VM, it is recommended to provide the service name and SID to be the standby DB names that you want in your configuration. In this example, we are using STBY as the service name & SID.

» Click *Next*.

» The details are then validated and a confirmation screen is shown. Once confirmed, it takes some time to create the service. Once the service is provisioned, proceed with the next steps.

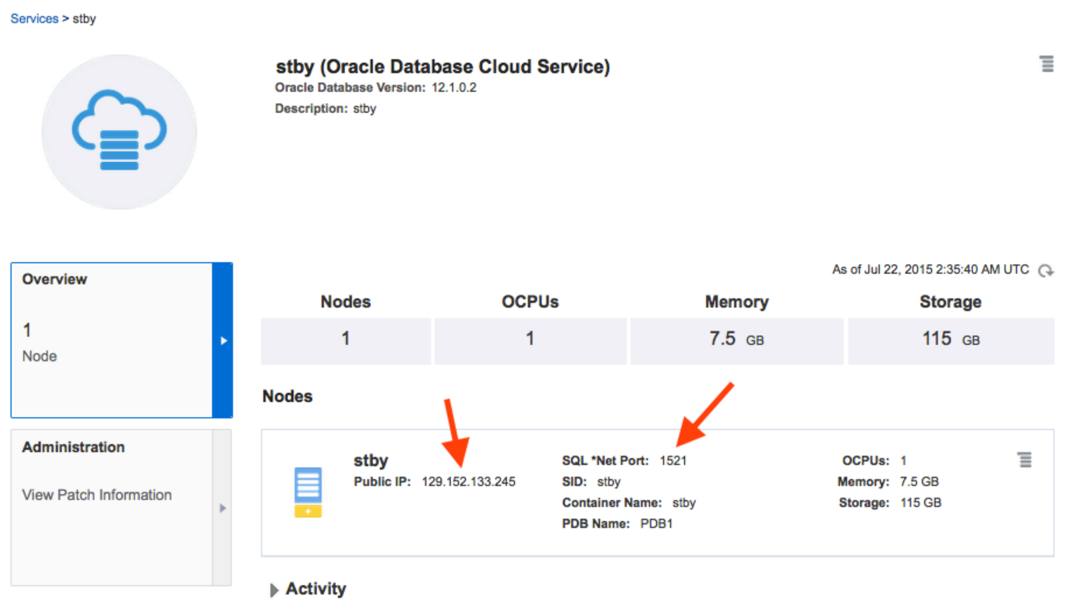
» Once the service is created, click on the service name from the dashboard. See Figure 5.



stby
Version: 12.1.0.2
Edition: Enterprise Edition - Extreme Performance
Created On: Jul 21, 2015 1:22:17 AM UTC
OCPUs: 1
Memory: 7.5 GB
Storage: 115 GB

Figure 5: Service Information

» A detail screen is shown with information about the service – such as public IP address, SQL port # etc. – See Figure 6.


Services > stby

stby (Oracle Database Cloud Service)
Oracle Database Version: 12.1.0.2
Description: stby

As of Jul 22, 2015 2:35:40 AM UTC

Nodes	OCPUs	Memory	Storage
1	1	7.5 GB	115 GB

Nodes

stby
Public IP: 129.152.133.245
SQL *Net Port: 1521
SID: stby
Container Name: stby
PDB Name: PDB1
OCPUs: 1
Memory: 7.5 GB
Storage: 115 GB

Figure 6: Instance Details

Note: If Database Cloud Service- Virtual Image service level is chosen, you will still have to choose the software release (11g or 12c), Software Edition (Enterprise – Extreme Performance) , Service name and public key. A service will be created with just Oracle binaries with the chosen release installed. No database will be created for you and no other tooling will be available for monitoring and backups.

3. Configure Network

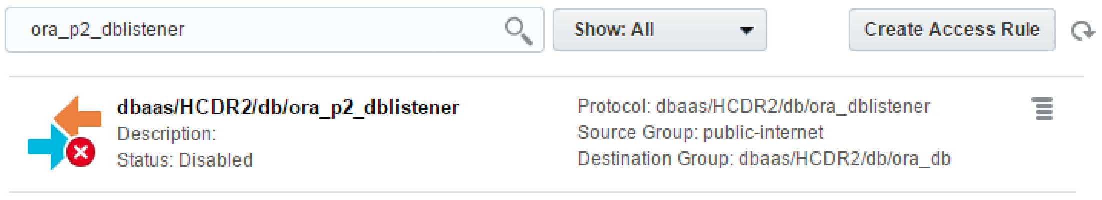
This section provides steps required for configuring the network on cloud and on-premises.

3.1 Cloud Network Configuration

It is critical to secure the port connectivity in the cloud. To enable SSH tunneling and also to make sure only specific on-premises IP addresses can access the listener port in the cloud, the following steps are done. All these are configured from the cloud Dashboard (services) → Oracle Compute Cloud Service → Network tab.

By default, cloud security rule for port 1521 is disabled. Also, this default pre-configured port in the cloud VM has open access from public internet. The first step is to delete the default 1521 listener.

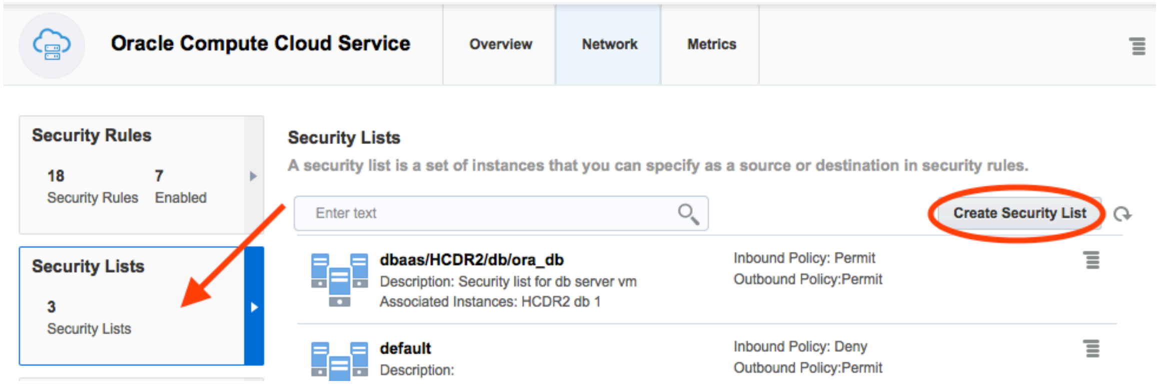
Click on “Security Rules” and will list a bunch of rules. Search for the listener, for example, ‘ora_p2_dblistener’. A listener will be listed for each service, as shown in Figure 7. Click the bars and click Delete. This will delete this listener.


ora_p2_dblistener

dbaas/HCDR2/db/ora_p2_dblistener
Description:
Status: Disabled
Protocol: dbaas/HCDR2/db/ora_dblistener
Source Group: public-internet
Destination Group: dbaas/HCDR2/db/ora_db

Figure 7: View Security Rules

Next step is to create Security IP list. Click on “Security Lists” and click “*Create Security List*” as shown in Figure 8.



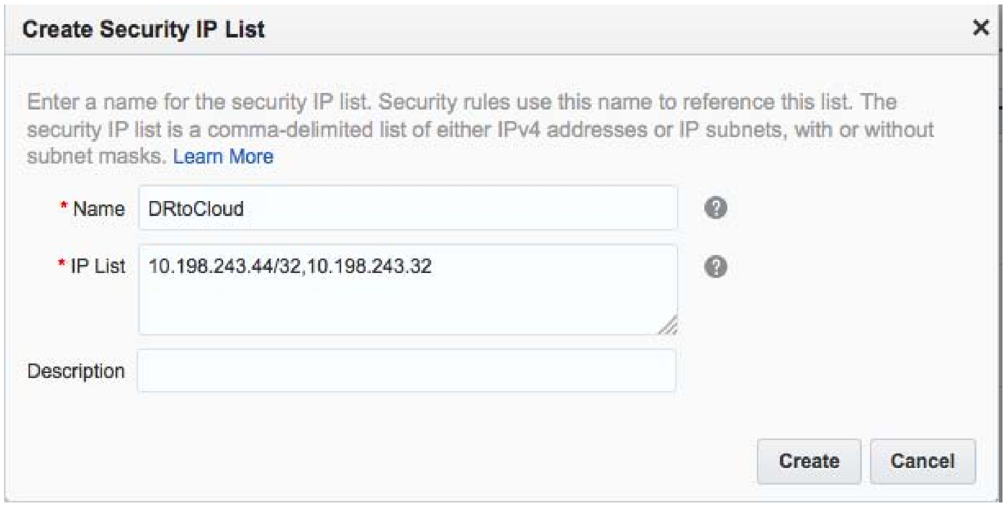
Security Lists

A security list is a set of instances that you can specify as a source or destination in security rules.

dbaas/HCDR2/db/ora_db Description: Security list for db server vm Associated Instances: HCDR2 db 1	Inbound Policy: Permit Outbound Policy:Permit
default Description:	Inbound Policy: Deny Outbound Policy:Permit

Figure 9: Create Security IP List

Give a name for the IP list and add the list of on-premises IP addresses that are allowed to access the port in the cloud VM. You can optionally provide subnet information. Clicking the (?) provides additional information. Refer to Figure 9. Click *Create*.



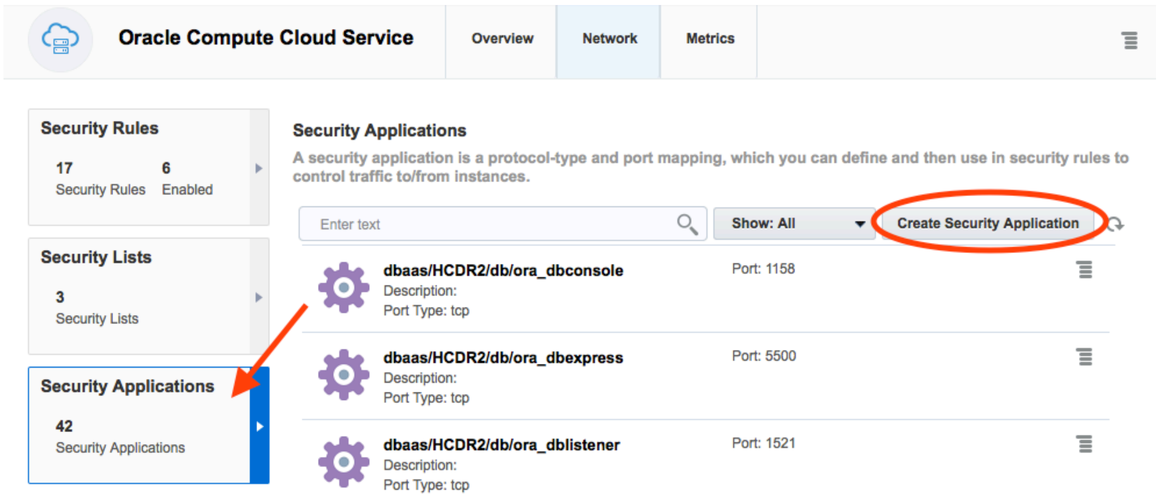
Create Security IP List

Enter a name for the security IP list. Security rules use this name to reference this list. The security IP list is a comma-delimited list of either IPv4 addresses or IP subnets, with or without subnet masks. Learn More

* Name	DRtoCloud
* IP List	10.198.243.44/32,10.198.243.32
Description	

Figure 9: Security Access List

Create a protocol (Security Application) with a listener port. You can use either 1521 or define any other port number for security reasons. We recommend creating a non-1521 port. Under Cloud Compute Services → Networks tab, click on Security Applications and "*Create Security Application*", See Figure 10.



Security Applications

A security application is a protocol-type and port mapping, which you can define and then use in security rules to control traffic to/from instances.

dbaas/HCDR2/db/ora_dbconsole
Description:
Port Type: tcp
Port: 1158

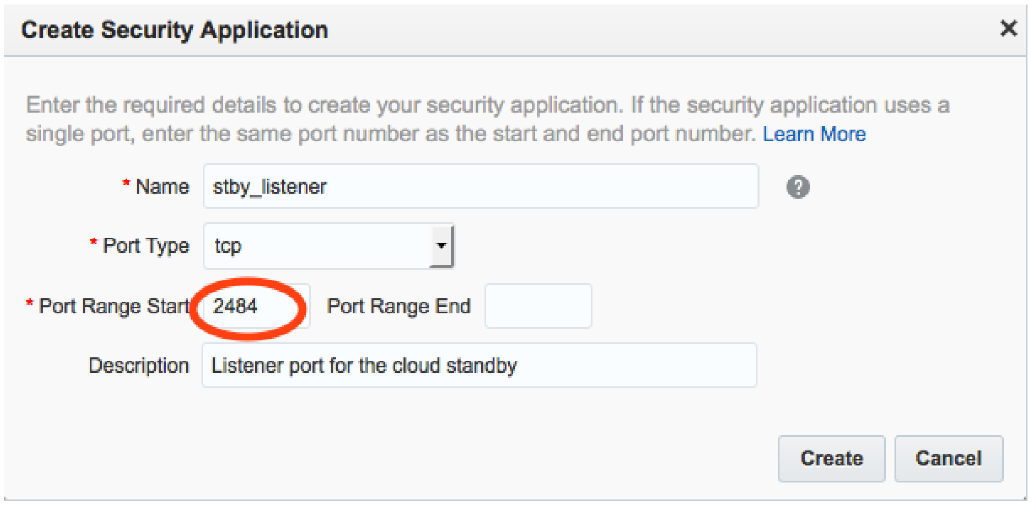
dbaas/HCDR2/db/ora_dbexpress
Description:
Port Type: tcp
Port: 5500

dbaas/HCDR2/db/ora_dblistener
Description:
Port Type: tcp
Port: 1521

Figure 10: Choose Security Applications (Protocols)

Provide name, port type (TCP), port # and a description. For example, port # 2484 is provided in Figure 11. If port 1521 is preferred, input that port number.

Note: When choosing an alternate port number, choose a number larger than 1024 as most port numbers in the 1-1024 range are used for other purposes.

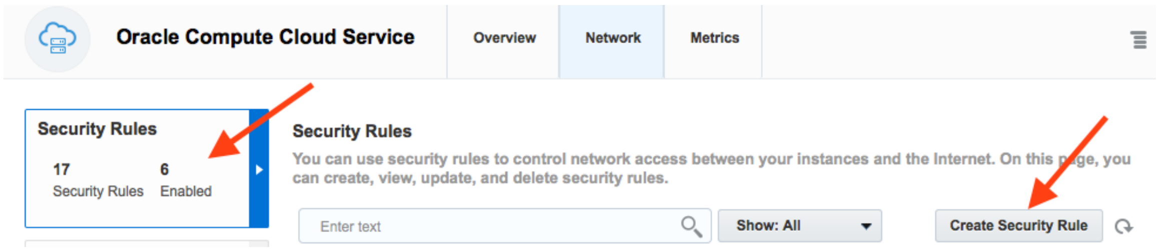


Enter the required details to create your security application. If the security application uses a single port, enter the same port number as the start and end port number. Learn More

* Name	stby_listener
* Port Type	tcp
* Port Range Start	2484
Port Range End	
Description	Listener port for the cloud standby

Figure 11: Create Security Application (Protocol)

Once this is done, enable that port by creating a Security Rule. See Figure 12.



Oracle Compute Cloud Service

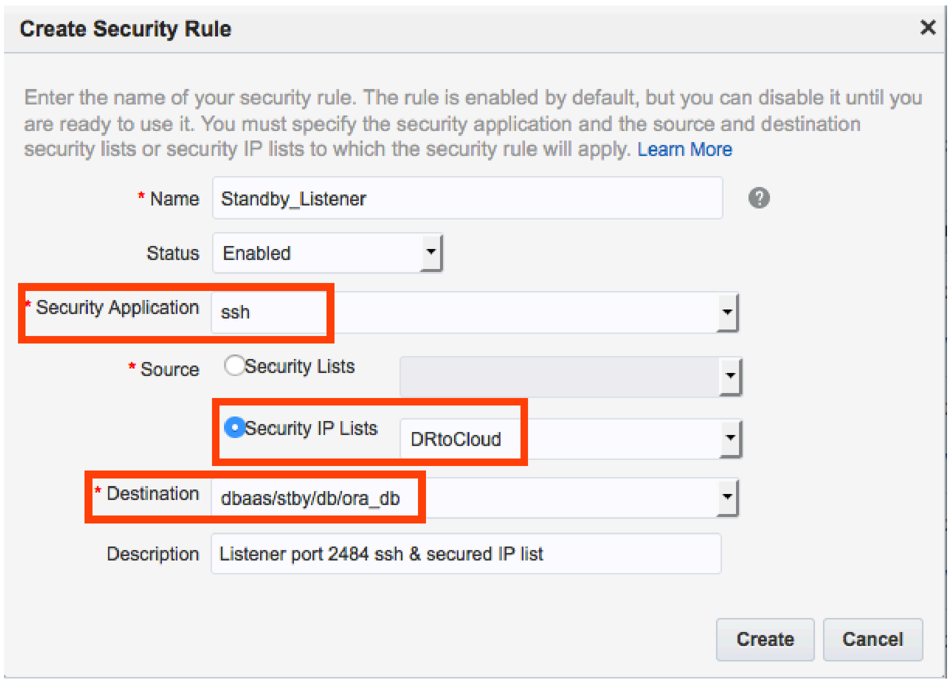
Security Rules
17 Security Rules
6 Enabled

Security Rules
You can use security rules to control network access between your instances and the Internet. On this page, you can create, view, update, and delete security rules.

Figure 12: Create Security Rule

Create a Security Rule. Provide a rule name, choose "**ssh**" from the Security Applications drop-down, choose the status "**Enabled**", choose the Security IP lists that we defined earlier (for example, *DRtoCloud*) from the drop-down and destination as the standby database. See Figure 13. Click *Create.*

Do not select the default "public-internet" from the Security IP lists – as that will enable access from all IP addresses to the cloud database listener. Using public-internet is not recommended from the security perspective,.



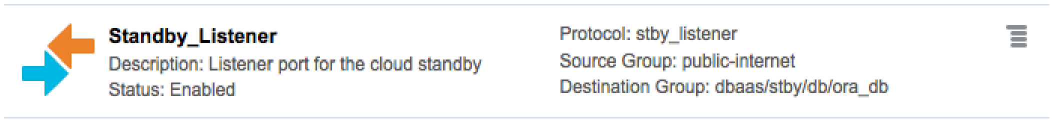
Create Security Rule

Enter the name of your security rule. The rule is enabled by default, but you can disable it until you are ready to use it. You must specify the security application and the source and destination security lists or security IP lists to which the security rule will apply. Learn More

* Name	Standby_Listener
Status	Enabled
* Security Application	ssh
* Source	Security Lists
	[✓ Security IP Lists] DRtoCloud
* Destination	dbaas/stby/db/ora_db
Description	Listener port 2484 ssh & secured IP list

Figure 13: Create Security rule

Verify the new port is enabled. You can then configure *listener.ora* for the standby to use the port 2484 or any port number you configured, see Figure 14.



Standby_Listener
Description: Listener port for the cloud standby
Status: Enabled

Protocol: stby_listener
Source Group: public-internet
Destination Group: dbaas/stby/db/ora_db

Figure 14: Verification of new port

With this step, a secured port is created and opened for SQL*Net access from on-premises primary database.

3.2 Oracle Net Encryption Configuration

» Oracle Net encryption must be enabled by setting the following in the sqlnet.ora on the primary (On Premises) and standby (On Cloud) database servers located in $ORACLE_HOME/network/admin.

```
SQLNET.ENCRYPTION_SERVER = requested (for on-premises)
SQLNET.ENCRYPTION_SERVER = required (for Cloud)
SQLNET.ENCRYPTION_TYPES_SERVER = (RC4_256, AES256) (both)
SQLNET.ENCRYPTION_CLIENT = requested (both)
SQLNET.ENCRYPTION_TYPES_CLIENT = (RC4_256, AES256) (both)
```

» For the procedure to create and disperse wallets, refer to Appendix H.

3.3 On-Premises Network Configuration

In a Data Guard configuration, information is transmitted in both directions between primary and standby databases. This requires basic configuration, network tuning and opening of ports at both primary and standby databases.

On-Premises Prerequisites

The following prerequisites must be met before instantiating the standby database:

1. Name resolution to the Oracle Cloud VM needs to be configured. This can be done either through a static file like /etc/hosts, or configuring the on-premises DNS to properly resolve the public IP address of the Oracle cloud instance. Also, the on-premises Firewall will need to have Access Control Lists properly configured to allow SSH and Oracle Net to be accessed from the on-premises system to the Oracle Cloud VM. Note that configuration is simplified when Virtual Private Network (VPN) is used with Exadata Service.

2. Since Data Guard in a DR situation requires access from the cloud instance to the on-premises database, the primary database listener port must be opened with restricted access from the cloud IP addresses using features like **iptables** etc. Since every corporation has different network security policies, the network administrator will need to perform operations similar to the cloud-side network configuration shown in preceding sections.

3. Prompt-less SSH from the Oracle Cloud VM to the On-Premises machine. This is configured both for on-premises to the Cloud during the provisioning process and from the Cloud to on-premises. See 'Appendix B - Configure SSH' for additional details.

4. Configuration of the On-Premises firewall to allow inbound ssh connectivity from the Oracle Cloud VM to the On-Prem machine.

5. The primary database intended for Data Guard replication must be running in archivelog mode. Refer to details in 'Appendix B - Archivelog mode' for placing a database in archivelog mode.

6. Standby Redo Logs (SRLs) must be configured to enable real-time apply of redo. Configure SRLs at the primary database prior to instantiating the standby database to reduce the number of manual steps required during instantiation. See 'Appendix B - Create Standby Redo Logs' for additional details.

7. The Oracle Home for the on-premises database must be the same Oracle patchset and have the same patches as the standby database. Compare the output of '`$ORACLE_HOME/OPatch/opatch lspatches`' between the two sites and apply those that are missing on each side (both those that are installed on-premises but not on cloud and those installed on cloud but not on-premises). If not already installed, the master note of OPatch can be found here. Note that some patches to the cloud VM will be noted as specific to the cloud and thus are not relevant to the on-premises system.

Note: Refer to the My Oracle Support Note 1265700.1 on Data Guard Standby-First patch apply for applying Exadata bundle patches when using the Exadata Cloud service.

8. The steps outlined in this document assume that the on-premises primary database is not already part of an existing Data Guard broker configuration. If there is an existing broker configuration for the on-premises database it is assumed that the administrator has prior knowledge of the broker and knows how to add the new standby database to an existing broker configuration. Refer to Appendix D for setting up Data Guard Broker.

   A value other than 'NOCONFIG' for the following query implies an existing broker configuration.

   ```
   SQL> select decode(count(1),0,'NOCONFIG') from v$DG_BROKER_CONFIG;
   ```

9. Use the default listener named LISTENER. The steps outlined in this document assume the default listener name LISTENER is used. To verify run the following command from the on-premises machine. The expected result is shown.

   ```
   $ lsnrctl show current_listener | grep 'Current Listener'
   Current Listener is LISTENER
   ```

10. Verify the listener port by running the following command from the on-premises machine. The expected result is shown.

    ```
    $ lsnrctl stat | grep 'Connecting to'
    Connecting to (ADDRESS=(PROTOCOL=tcp)(HOST=)(PORT=(1521)))
    ```

4. Encrypt Primary Database (Optional)

Data Guard primary and standby databases are physical replicas of each other. If a standby database in the cloud is to be encrypted at rest, then the on-premises primary must be encrypted first. Oracle MAA Best Practices provide guidance for converting to TDE with minimal downtime for Oracle Database 11.2[13] or for Oracle Database 12c[14].

5. Instantiate Data Guard Standby

As illustrated earlier, a default database instance is automatically created when you initially subscribe for Oracle Database Cloud Service. This default database cannot be used as a Data Guard standby database and hence needs to be deleted manually.

Note: This manual step will be eliminated in future version of database cloud service

13 http://www.oracle.com/technetwork/database/availability/tde-conversion-11g-2531187.pdf

14 http://www.oracle.com/technetwork/database/availability/tde-conversion-12c-2537297.pdf

There are several options to instantiate a Data Guard standby in the cloud:

- Instantiate from on-premises production.
- Instantiate from Oracle Database Backup Cloud Service
- For very large databases, 10s - 100s of TB, use bulk import capability (in controlled availability at the time this paper was published)

5.1 From On-Premises Production Database

This mode of standby creation is shown in Figure 14. The on-premises production database is used to instantiate the standby database in the cloud. Appendices B, C & D list detailed steps for instantiating the standby from an active on-premises primary database.

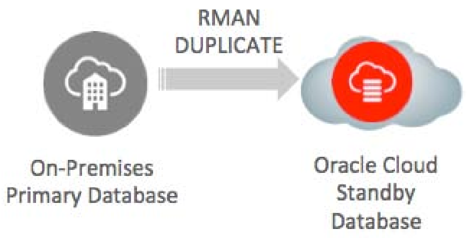


Figure 14: Instantiation using RMAN duplicate an On-Premises Primary

5.2 From Oracle Database Backup Cloud Service

This mode of standby creation is shown in Figure 15. On-premises production database backups stored in Oracle Database Backup Cloud Service (ODBCS)[15] are used to create the standby database using RMAN restore and recovery commands. Restore from the backup requires the same Password or TDE key.

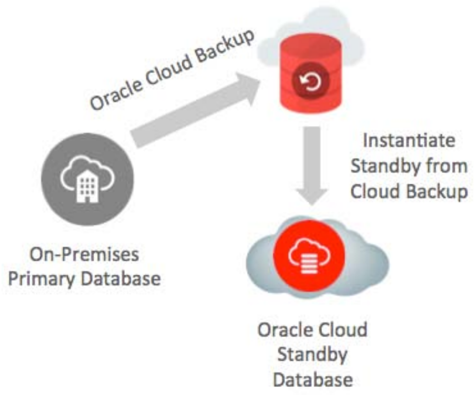


Figure 15: Instantiation using Oracle Database Backup Cloud Service

Refer to Appendix I for the steps to create a standby database from the Database Backup Service.

15 https://cloud.oracle.com/database_backup

6. Perform Data Guard Health Check

After the standby is instantiated, a health check should be performed to ensure the Data Guard databases (primary and standby) are compliant with Oracle MAA best practices. It is also advisable to perform the health check on a monthly basis as well as before and after database maintenance. There are several methods for checking the health of a Data Guard configuration:

Oracle MAA Scorecard

Oracle provides several automated health check tools that can be downloaded from My Oracle Support specific for the type of hardware platform:

- ORAchk applicable to generic platform (suitable for Database Cloud Service)[16]
- exachk applicable to Exadata Database Machine (suitable for Exadata Cloud Service)[17]
- ODAchk applicable to the Oracle Database Appliance (relevant only to on-premises databases in a hybrid cloud configuration)[18]

Each of the automated checks include an Oracle MAA Scorecard that reports on a number of key Data Guard configuration best practices in addition to many other checks.

Oracle strongly recommends the use of these automated tools for comprehensive health check of not only the Data Guard configuration but the system as a whole. The health checks are regularly updated with current information. Be sure to download the latest version of the health checks applicable to your platform.

Data Guard Specific Queries (Applicable from Oracle Database 11g onward)

A set of Data Guard specific queries are provided in Appendix E along with sample output that can be used to validate the health of the Data Guard configuration.

Data Guard VALIDATE DATABASE (Applicable from Oracle Database 12c onward)

The Data Guard Broker VALIDATE DATABASE command is highly recommended for the most comprehensive Data Guard specific health check. VALIDATE DATABASE performs an extensive configuration check and validates the configuration’s readiness for switchover or failover.

Example:

```
DGMGRL> validate database stby;
```

```
Database Role:     Physical standby database
Primary Database:  pri

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

16 https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1268927.2

17 https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1070954.1

18 https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1485630.1

See the Data Guard broker documentation for more information on the extensive checks performed by the VALIDATE DATABASE command[19].

7. Enable Run Time Monitoring

There are two options for monitoring the run-time status of a Data Guard configuration: command line queries or Enterprise Manager Cloud Control.

Option #1: Command Line Monitoring

Monitoring and alerting is required to ensure that the hybrid DR configuration is able to meet service levels established by the administrator for data protection (RPO) and availability (RTO).

See Appendix A for a series of queries that monitor key aspects of the run time health of a Data Guard configuration:

» Transport Lag
» Apply Lag
» Primary/Standby connection status
» Data Guard Overall Status and Role Transition Readiness

The administrator will need to create scripts to automate the monitoring of the configuration such that alerts are sent when values exceed service level thresholds. This manual effort is not required if using Enterprise Manager 12c.

Option #2: Enterprise Manager 12c or 13

Oracle Enterprise Manager 12c or 13 streamlines and automates complex management tasks across the complete cloud lifecycle. On-premises administrators can monitor and manage cloud services, and vice versa.

By deploying Management Agents onto the Oracle Cloud virtual hosts serving Oracle Cloud services, you are able to manage Oracle Cloud targets just as you would any other targets. The communication between Management Agents and on-premise Oracle management service instances is secure from external interference (requires Enterprise Manager Cloud Control 12c Release 1 (12.1.0.5) or higher). Support is provided for managing Oracle Database and Fusion Middleware PaaS targets, as well as JVMD support for monitoring JVMs on Oracle Cloud virtual hosts.

Oracle Cloud Management includes the following features:

» Automated agent deployment and configuration
» Database and Java PaaS instances monitoring
» Incident management including notifications and ticketing integration
» Configuration management including Search and Inventory, comparison between on-premise and cloud instances, configuration history, and compliance
» Cloning between on-premise and Oracle Cloud
» One-off patching of Oracle Cloud database instances

19 http://docs.oracle.com/database/121/DGBKR/dbresource.htm#DGBKR3855

Enterprise Manager provides a simple interface to monitor and manage a Data Guard environment, including:

» Data Guard status
» Transport Lag
» Apply Lag
» Estimated Database Role Transition Time
» Primary or standby is or is not accessible

Alerts can be customized to notify of any event or when any of the above metrics exceed desired thresholds.

Refer to the Enterprise Manager Cloud Control Administrator's Guide for guidance on how to enable Hybrid Cloud Management[20].

Note: Support for managing a hybrid cloud DR configuration using Enterprise Management Cloud Control had not been certified as of the date this paper was published. Download the current version of this paper to see if this limitation has been lifted or see Enterprise Manager documentation for latest information.

8. Enable Redo Transport Compression (Optional)

Customers who have licensed Oracle Advanced Compression for their on-premises systems may use redo transport compression if available network bandwidth is less than the uncompressed redo volume. Redo transport compression is only recommended for environments that have sufficient CPU headroom on the primary and standby but low network bandwidth between primary and standby.

Customers who wish to use redo transport compression but who are concerned for the impact to primary database performance have the additional option of using Oracle 12c Active Data Guard Far Sync to offload compression overhead to a Far Sync instance deployed on premises. See the Oracle MAA Best Practice Paper, *"Oracle Active Data Guard Far Sync - Zero Data Loss at Any Distance",*[21] for additional information.

Note: Redo Transport Compression will produce minimal benefit for TDE encrypted databases since encrypted redo cannot be compressed.

DR Operations

Once the DR to cloud is successfully established, you can verify the standby for DR readiness as well as perform maintenance tasks as listed below:

1. Validate DR Readiness
2. Using Standby Database to reduce downtime during planned maintenance
3. Failover/Switchover to the cloud
4. Failback/Switchback to on-premises
5. Connecting to the Application Tier

20 http://docs.oracle.com/cd/E24628_01/doc.121/e24473/part_hybrid_cloud.htm

21 http://www.oracle.com/technetwork/database/availability/farsync-2267608.pdf

1. Validate DR Readiness

Best practice is to use Active Data Guard to offload read-only workload to the standby database to provide continuous, application-level validation that the standby is ready for production. This provides a level of assurance that is in addition to continuous Oracle block-level validation performed by Data Guard apply processes. It is also best practice to periodically place the standby in read/write mode (using Data Guard Snapshot Standby) to validate its readiness to support read-write production workloads. A snapshot standby may also be used for a final level of pre-production functional and performance testing of patches and upgrades since the DR system is frequently sized similar to the production system. A Snapshot Standby continues to receive redo from the primary database where it is archived for later use, thus providing data protection at all times. Recovery time (RTO), however, will be extended by the amount of time required to convert the Snapshot Standby back to the standby database if a failover is required while testing is in progress. Note that additional storage is required for the fast recovery area when a standby is in snapshot mode (to hold archived redo received from the primary production database for later use and current redo and flashback logs generated by the snapshot standby). Steps for converting a standby to a snapshot standby and back are listed in Appendix F. Please refer to Oracle documentation for additional details on Data Guard Snapshot Standby[22]. Optionally you may perform an actual switchover or failover operation to the cloud for a complete end-to-end DR test; for more details see Failover/Switchover to the Cloud.

2. Using Standby Database to reduce downtime during Planned Maintenance

There are several options for utilizing a standby database on the cloud for reducing planned downtime of the primary production database:

Standby-first Patch Apply: Many patches may be applied first to a physical standby for thorough validation. Customers who wish to minimize downtime will frequently patch the standby first, then switch production to the standby database, and then patch the original primary. Data Guard physical replication is supported between primary and standby running at mixed patch versions for patches that are standby-first eligible (documented in the patch read-me). The customer may also choose to run for a period of time with mixed patch versions between primary and standby to enable fast fallback to the unpatched version should there be any unanticipated problems with the patch. See My Oracle Support Note 1265700.1, *"Oracle Patch Assurance - Data Guard Standby-First Patch Apply"*[23] for more details on patches eligible for the standby-first process.

Database Rolling Upgrade: Another beneficial use case for standby in the Oracle cloud is for database rolling upgrade to reduce downtime when upgrading to new database patch-sets and full Oracle releases. The transient logical process is used in Oracle 11g and Oracle 12c to temporarily convert a physical standby database to a logical standby, upgrade the logical standby to the new version, validate and when ready execute a Data Guard switchover. After the switchover completes, the original primary database is converted to a synchronized physical standby also operating at the new release. Refer to Oracle 11g Database Rolling Upgrades Made Easy[24] or Oracle 12c DBMS_Rolling[25] for more information.

22 http://docs.oracle.com/database/121/SBYDB/manage_ps.htm#BACIEJJI

23 https://support.oracle.com/epmos/faces/DocumentDisplay?id=1265700.1

24 http://www.oracle.com/technetwork/database/features/availability/maa-wp-11g-upgrades-made-easy-131972.pdf

25 http://docs.oracle.com/database/121/SBYDB/dbms_rolling_upgrades.htm#CJACBBBC

3. Failover/Switchover to the Cloud

At any time you can manually execute a Data Guard switchover (planned event) or failover (unplanned event). Customers may also choose to automate Data Guard failover by configuring Fast-Start failover. Switchover and failover reverse the roles of the databases in a Data Guard configuration – the standby in the cloud becomes primary and the original on-premises primary becomes a standby database. Refer to Oracle MAA Best Practices[26] for additional information on Data Guard role transitions. For failover of the application tier, please refer to the "Connecting the Application Tier" section of this paper.

Switchovers are always a planned event that guarantees no data is lost. To execute a switchover perform the following in Data Guard Broker

```
DGMGRL> validate database stby;

Database Role:     Physical standby database
Primary Database:  pri
```

Ready for Switchover: Yes

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

DGMGRL> switchover to <target standby>;
```

A failover is an unplanned event that assumes the primary database is lost. The standby database is converted to a primary database immediately; after all available redo from the primary has been applied. After a failover the old primary database must be reinstated as a physical standby which is made simpler with flashback database and Data Guard broker enabled. To execute a failover and reinstatement execute the following in Data Guard Broker.

```
DGMGRL> failover to stby;

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

Execute startup mount on one instance of the old primary before reinstating.

```
SQL> shutdown abort
SQL> startup mount
DGMGRL> reinstate database pri
Reinstating database "pri", please wait...
```

For more information on role transitions using the Data Guard Broker see the broker documentation for Oracle Database 11g[27] or 12c[28].

4. Switch back to On-Premises

The same role transition procedure mentioned in the failover/switchover process is applied again when you are ready to move production back to the on-premises database.

26 http://www.oracle.com/technetwork/database/availability/maa-roletransitionbp-2621582.pdf

27 http://docs.oracle.com/cd/E11882_01/server.112/e40771/toc.htm

28 http://docs.oracle.com/database/121/DGBKR/toc.htm

5. Connecting the Application Tier

It is important to understand the implications of network latency between an on-premises application tier and a remote database residing in the Oracle Cloud in a hybrid cloud DR configuration. This is especially relevant when there has been a switchover or failover of production to the cloud standby while the application tier remains on premises, or when an on-premises application tier makes remote read-only connections to an Active Data Guard standby on the cloud. Some applications are especially chatty with the database and thus vulnerable to substantial performance overhead when making a remote connection. Other applications, WebLogic-based applications for example, store their metadata in the database itself.

Ideally, the application tier is able to tolerate the latency of a remote connection. During steady state operation the application tier will access services running on the on-premises primary database. Following a switch-over or failover to the cloud, application connections are automatically redirected to the new primary database. Automating the failover of the application tier is done as follows:

- » The application tier is pre-configured to automatically connect to whichever database in its Oracle Net connection descriptor offers the desired database service.
- » Network connectivity between the on premises application servers and Cloud database server(s) has been configured similar to the database tiers. The following pre-requisites must be met:
    - » Name resolution to the Oracle Cloud VM needs to be configured for the application tier. This can be done either through a static file like /etc/hosts, or configuring the on-premises DNS to properly resolve the public IP address of the Oracle cloud instance. Also, the on-premises Firewall will need to have Access Control Lists properly configured to allow SSH and Oracle Net to be accessed from the on-premises systems to the Oracle Cloud VM.
    - » Configuration of the OnPrem firewall to allow inbound ssh connectivity from the Oracle Cloud VM to the OnPrem machine will be needed.
- » Role-specific database services are preconfigured so that the appropriate services start automatically and are only ever available on the database that is functioning in the primary role.
- » Oracle Fast Application Notification (FAN) or alternatively careful configuration of tcp time-outs, enables clients connected to the failed database to quickly drop their connection when there is an outage and automatically reconnect to the new primary database on the cloud.

Additional information for configuring automatic failover of an application tier to a new primary database is provided in Appendix G.

If the application tier is not able to tolerate the latency of a remote connection or if the entire primary site is unavailable, then failover/switchover of both application and database tiers to the Oracle Cloud will be required.

Oracle Cloud provides two compute offerings[29] as Infrastructure-as-a-Service (IaaS) to host an application tier:

- » Dedicated Compute Cloud Service that runs on dedicated hardware with network isolation
- » Compute Cloud Service that provides on-demand, scalable compute resources

29 https://cloud.oracle.com/compute

6. Full Stack Failover: Application and Database Tiers

Oracle Compute Services complement Oracle Database PaaS by enabling customers to host a DR copy of their on-premises application tier in the Oracle Cloud. Customers install the application binaries using their desired level of compute service to enable full stack failover to the cloud in the event of a complete site outage.

It is not common for an on-premises application tier to also write persistent data to its local file system. A full stack failover of both application and database tiers will require both database data and any application tier data that resides in the file system outside of the Oracle Database be replicated to the cloud.

Data Guard and Active Data Guard replicate data residing in the Oracle Database. Additional measures must be taken to replicate file system data outside of the Oracle Database. Customers licensed for Active Data Guard and utilizing either Extreme Performance or Exadata cloud services have the option of periodically copying file system data from their on-premises application tier and into the Oracle Database File System (DBFS). Once in DBFS, Active Data Guard is able to replicate all data to the Oracle cloud to provide full-stack DR protection.

The following diagram depicts the flow of full stack DR to cloud:

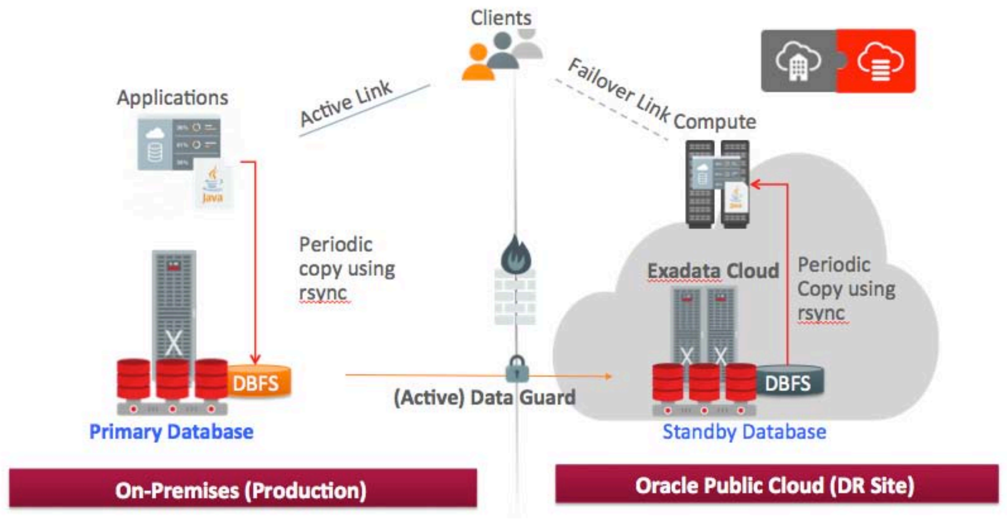


Figure 17: Full Stack DR using DBFS and Active Data Guard

DBFS[30] *provides a standard file system interface on top of files and directories that are stored in database tables. DBFS is similar to NFS in that it provides a shared network file system that looks like a local file system. In DBFS, the server is the Oracle Database. Files are stored as SecureFiles LOBs in a database table.*

The periodic copy of file system data into DBFS is automated via a script that uses *rsync*, a simple and commonly used utility to copy and synchronize files from one location to another. The script, rsync_copy.sh, is available in My Oracle Support Note 2099044.1[31]. The process uses prompt-less ssh and is scheduled in the cron to run at whatever interval is desired by the administrator. The script is configured to run on the primary site to copy data from the on-premises application server into DBFS. It is also configured to run on the cloud to copy data from DBFS to

30 http://docs.oracle.com/cd/E11882_01/appdev.112/e18294/adlob_fs.htm#ADLOB45943

31 http://support.oracle.com/epmos/faces/DocumentDisplay?id=2099044.1

the remote application servers in order to keep application files current (DBFS must be open read-only at the standby database – thus the requirement for Active Data Guard).

The combination of *rsync* and DBFS also simplifies network configuration; data on-premises and in cloud is automatically synchronized using Active Data Guard replication over a single connection. Refer to Appendix J for setup requirements, directions and steps to complete a full stack role transition. Customers who are not licensed for Active Data Guard or who are unable to use DBFS have the option of using direct *rsync* copy from on-premises to Oracle Cloud by configuring a second secure network connection between their data center and the cloud.

rsync is a freely available open source file synchronization utility which provides a bandwidth-efficient delta transfer methodology. It is an effective and commonly used method to synchronize files and directories; however it does not guarantee write-ordering or file system transaction ordering; file system and directories will be at an approximate point-in-time relative to one another. Similarly this solution does not provide point-in-time synchronization between all files and the database.

Conclusion

Disaster recovery in a hybrid cloud configuration consists of an on-premises production database and a DR copy on the Oracle Cloud synchronized by Oracle Data Guard or Active Data Guard. Disaster Recovery on the Oracle Cloud eliminates the costs and complexity of owning and managing a remote facility as well as the capital expense of standby systems and software.

The use of Data Guard or Active Data Guard for disaster recovery eliminates the downtime and potential risk of relying upon a remote backup to restore service; production is quickly failed over to an already running, synchronized copy of your production database on the Oracle Cloud. The standby database on the cloud not only provides disaster recovery, it can also be used to seed clone databases for development and test as well as offloading read-only workloads from on-premises production databases.

Appendices

Appendix A – Data Guard Run-Time Monitoring Queries

As of the release of this document monitoring the standby database in a Hybrid Data Guard configuration must be done with queries and scripts. There are a few basic recommendations for monitoring which include:

1) Validate redo transport is functioning and there is no transport lag- This means redo is being shipped to the standby in a timely manner.

2) Validate redo apply is functioning and there is no apply lag- This means redo is being applied in a timely manner as it is received.

Each of these things can be monitored via Data Guard broker or SQL*Plus queries.

Monitoring with Data Guard Broker

Monitoring with Data Guard should focus on Database Status of the show database <standby db_unique_name> command. A Database Status of SUCCESS implies there are no issues.

```
DGMGRL> show database stby;

Database - stby

  Role:                PHYSICAL STANDBY
  Intended State:      APPLY-ON
  Transport Lag:       0 seconds (computed 1 second ago)
  Apply Lag:           0 seconds (computed 1 second ago)
  Average Apply Rate:  20.00 KByte/s
  Real Time Query:     ON
  Instance(s):
    stby1
    stby2 (apply instance)

Database Status:
SUCCESS
```

When there are issues with transport or apply functionality the status will read WARNING or ERROR.

Setting the ApplyLagThreshold and TransportLagThreshold properties on the standby database enables monitoring lag in the same method (Database Status). Each of these properties is expressed in seconds. By setting these properties to a non-zero value, when the value is exceeded the Database Status will be changed to WARNING.

Monitoring with SQL*Plus

<u>To Validate Redo Transport and Redo Apply</u>

The basics of Data Guard are that redo is transported from the primary and then applied on the standby. The queries below can be used to validate these functions are working properly.

PRIMARY DATABASE QUERY: Current state of Data Guard Transport:

To validate that the primary database is shipping redo to the standby execute the following query on the primary database. A STATUS of VALID is expected. ERROR will be populated with additional information if the STATUS does not equal VALID. gv$archive_dest_status should be used in case of a RAC primary database in order to validate each primary instance.

```
SQL> select sysdate,status,error from v$archive_dest_status where
type='PHYSICAL';

SYSDATE              STATUS    ERROR
-------------------- --------- ----------
13-mar-2015 10:42:10 VALID
```

STANDBY DATABASE QUERY: Current State of the Standby Apply

A standby can be receiving redo without applying it. Therefore, the query above is just once piece of the validation. To validate that redo apply is running on the standby database execute the following query. A RECOVERY_MODE of 'MANAGED REAL TIME APPLY' implies that redo is applied as it is received. GAP_STATUS indicates whether there is a gap in redo transport from the primary.

```
SQL> select sysdate,database_mode,recovery_mode, gap_status from
v$archive_dest_status where type='PHYSICAL';

SYSDATE              DATABASE_MODE   RECOVERY_MODE           GAP_STATUS
-------------------- --------------- ----------------------- ----------
13-mar-2015 10:43:44 OPEN_READ-ONLY  MANAGED REAL TIME APPLY NO GAP
```

Data Guard Performance (Standby Lag)

Redo that has been successfully written but not applied to the standby is called an apply lag. An apply lag indicates there are insufficient resources at the standby, often due to I/O or CPU contention. Apply lag does not indicate potential data loss exposure since the redo is present at the standby it can be applied prior to failover. However, when there is a transport lag, meaning redo is not being received and written to the standby redo logs in a timely fashion, there is a potential to lose the unwritten data should the primary be lost.

Network latency is often the cause of transport lag but can also be the result of insufficient resources at the standby.

The following queries will help identify the presence and size of either lag.

STANDBY DATABASE QUERY: Monitor for Apply and Transport Time **Lags**

Run the following query to identify an apply or transport lag. A VALUE of '+00 00:00:00' indicates no lag.

```
SQL> select name,value,time_computed,datum_time from v$dataguard_stats where
name like '%lag%';
NAME          VALUE                TIME_COMPUTED                  DATUM_TIME
------------- -------------------- ------------------------------ -------------
-------
transport lag +00 00:00:00         06/18/2015 09:26:29            06/18/2015
09:26:27
apply lag     +00 00:00:00         06/18/2015 09:26:29            06/18/2015
09:26:27
```

STANDBY DATABASE QUERY: Monitor the Size of a Transport Lag.

The following query can be used to determine how far behind in blocks the standby is from the primary when a transport lag is present.

```
SQL> select t.thread#,t.LAST_REDO_SEQUENCE#,t.LAST_REDO_BLOCK, s.thread#,
s.sequence#, s.block# from v$thread t, gv$managed_standby s where
s.process='LNS' and t.LAST_REDO_SEQUENCE#=s.sequence#;

   THREAD# LAST_REDO_SEQUENCE# LAST_REDO_BLOCK    THREAD#  SEQUENCE#     BLOCK#
---------- ------------------- --------------- ---------- ---------- ----------
         1                 453             482          1        453        482
         2                 406             786          2        406        786
```

STANDBY DATABASE QUERY: Monitor Standby Apply Process

```
SQL> select sysdate,process,status,thread#,sequence#,block# from
v$managed_standby where status!='IDLE';

SYSDATE              PROCESS   STATUS          THREAD#  SEQUENCE#     BLOCK#
-------------------- --------- ------------ ---------- ---------- ----------
13-mar-2015 10:58:31 ARCH      CLOSING               2        403          1
13-mar-2015 10:58:31 ARCH      CONNECTED             0          0          0
13-mar-2015 10:58:31 ARCH      CLOSING               1        449          1
13-mar-2015 10:58:31 ARCH      CLOSING               2        402       4096
13-mar-2015 10:58:31 MRP0      APPLYING_LOG          2        404       1614
```

STANDBY DATABASE QUERY: Monitor Recovery (advanced monitoring)

The following query can be used to gather valuable information about the apply process such as apply rates and apply lag. gv$recovery progress keeps a history of each recovery invocation for the life of the instance(until instance is restarted), this query select only metrics for the most recent invocation.

```
SQL> select start_time, item, units, sofar from gv$recovery_progress where
START_TIME in (select max(START_TIME) from v$recovery_progress);

START_TIM ITEM                             UNITS                                 SOFAR
--------- -------------------------------- -------------------------------- ----------
18-JUN-15 Active Apply Rate                KB/sec                                   23
18-JUN-15 Average Apply Rate               KB/sec                                    1
18-JUN-15 Maximum Apply Rate               KB/sec                                   24
18-JUN-15 Redo Applied                     Megabytes                                 0
18-JUN-15 Last Applied Redo                SCN+Time                                  0
18-JUN-15 Active Time                      Seconds                                  42
18-JUN-15 Elapsed Time                     Seconds                                 576
18-JUN-15 Standby Apply Lag                Seconds                                   1
```

Appendix B – (Active) Data Guard Configuration

Configure SSH

Prompt-less ssh should be configured in both directions. Generate an ssh key for the cloud environment to the on premises environment.

NOTE: this is already configured from on premises->cloud

In the Oracle Cloud environment generate the ssh key and copy the file to the cloud

```
$ ssh-keygen
$ scp ~/.ssh/id_rsa.pub oracle@<onpremisesIP>:~/.ssh/id_rsa.pub_cloud
```

In the On-Premises primary database envrionment copy the generated key to the authorized_users file for prompt-less ssh connectivity.

```
$ cat ~/.ssh/id_rsa.pub_cloud >> ~/.ssh/authorized_users
$ chmod 700 ~/.ssh/authorized_users
```

Oracle Restart Installation

The Oracle Grid infrastructure has become an integral part of the application failover features for Oracle Data Guard. By default the Grid Infrastructure is not installed with an Oracle Database Service in a standalone configuration and should be done manually following the appropriate documentation for 12.1 or 11.2. The rest of the steps for configuration assume Oracle restart has been installed and configured.

Note: The installation software for Oracle Grid infrastructure is not present on the cloud service and must be downloaded from OTN and installed. Downloading from OTN can be done directly to the OPC VM with firefox or to on-premises and scp'd to the OPC VM.

Note: The Grid Infrastructure (GI) that is installed with Oracle Restart, RAC One Node or RAC, is not required for the hybrid configuration to function properly however the steps in this document expect that the GI is installed. Therefore, commands which expect the GI will be different where it is not installed. Those alternate commands are not provided. Additionally, the GI is required for Fast Application Failover (FAN) functionality. Without the Grid Infrastructure FAN is not configurable.

Add swap space

The installer expects that swap space equals the amount of memory for systems with less than 16GB memory. The default configuration for an OPC VM has only 4GB of swap so additional swap should be configured with a swap file. These steps should be done via the root user.

1. Determine how much memory and swap is installed on the OPC VM(both values are in kB). In this example there is 4GB of swap and about 7.4GB memory.

```
# egrep MemTotal /proc/meminfo
MemTotal:        7710316 kB

# swapon -s
```

```
Filename                              Type          Size          Used    Priority
/dev/xvdb3                               partition  4194300       11772   -1
```

2. Make sure there is sufficient space for the new file, in this example the extra swap file will be about 3.4GB

```
# df -k /

Filesystem            1K-blocks       Used         Available       Use%    Mounted on
/dev/xvdb2             16313216    7229436                 8255384         47%      /
```

3. Create an empty file equal to or greater than the size of the difference between allocated swap space and memory to be used for swap

```
 # dd if=/dev/zero of=/extraswap bs=1M count=3400
```

4. Backup /etc/fstab

```
cp /etc/fstab /etc/fstab.mybackup
```

5. Edit /etc/fstab with a new line for swap

```
/extraswap       swap                    swap    defaults        0 0
```

6. Add the new swap file to swap

```
# swapon -a
```

7. Check that the new swap is available

```
#  swapon -s
Filename                              Type          Size          Used    Priority
/dev/xvdb3                               partition  4194300       11772   -1
/extraswap                               file               3481600 0             -2
```

Run the Installer

Use the appropriate 12.1 or 11.2 documentation for installation steps.

In order to run the installer on the OPC VM through the ssh connection, X11 forwarding must be enabled. Logged in as root via the opc user, set the following in /etc/ssh/sshd_config:

» X11forwarding yes
» X11DisplayOffset 10
» X11UseLocalhost yes

As root, restart the ssh deamon:

```
    $ service sshd stop
    $ service sshd start
```

When connecting to the OPC VM used the following options to identify the connection as trusted:

```
$ ssh -o ServerAliveInterval=100 -Y oracle@<IPC VM IP address>
```

When installing Oracle Restart:

» Select the installation option ‘Install Oracle Grid Infrastructure Software Only’
» Install the software in /u01/app/<12.1.0|11.2>/grid
» Ignore the warning for NTP, as this is not a RAC configuration.

Configure the Listener

After Restart has been installed, copy the listener.ora from the ORACLE_HOME/network/admin directory to the new GRID_HOME/network/admin directory

```
$ lsnrctl stop listener
$ cp $ORACLE_HOME/network/admin/listener.ora
/u01/app/12.1.0/grid/network/admin/listener.ora
```

Add a listener to the grid infrastructure and start it:

```
$ srvctl add listener -listener listener -oraclehome /u01/app/12.1.0/grid

$ srvctl start listener
```

The rest of the steps for configuration assume Oracle restart has been installed and configured.

Archivelog Mode

NOTE: Changing a database’s archivelog mode requires database restart.

1. Set LOG_ARCHIVE_DEST_*n* to a disk location local to the primary database

```
SQL> show parameter DB_RECOVERY_FILE_DEST
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------
------
db_recovery_file_dest                string      +RECO
db_recovery_file_dest_size           big integer 60000M

SQL> alter system set
log_archive_dest_1=’location=USE_DB_RECOVERY_FILE_DEST
valid_for=(ALL_LOGFILES,ALL_ROLES) DB_UNIQUE_NAME=pri’;

System altered.
```

If DB_RECOVERY_FILE_DEST is not set consider setting it and using Oracle Managed Files(OMF). Otherwise, set the location to the path where the archive logs will be stored.

2. Shutdown the database

```
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
```

3. Optionally, you can back up the database before placing in the archivelog mode..

4. Start the database in mount mode

```
SQL> startup mount
ORACLE instance started.

Total System Global Area 3808428032 bytes
Fixed Size                  2931232 bytes
Variable Size            1711277536 bytes
Database Buffers         1946157056 bytes
Redo Buffers              148062208 bytes
Database mounted.
```

5. Place the database in archivelog mode and open the database.

```
SQL> alter database archivelog;

Database altered.

SQL> alter database open;

Database altered.
```

6. Switch a log to write the first archive log.

```
SQL> alter system archive log current;

System altered.
```

Size Online Redo Logs Appropriately

Online redo logs and standby redo logs should use the larger of redo log size = 1GB or redo log size >= peak redo rate/minute x 20 minutes. To extract peak redo rates, please refer to AWR reports during peak workload periods such as batch processing, quarter or year end processing. It is very important to use peak workload and not averages (averages can obscure peak redo rates and lead to provisioning too small of a redo log). Table 1 provides a quick mapping of redo-rate to the minimum recommended redo log size:

TABLE 1: RECOMMENDED REDO LOG SIZE

Peak redo rate according to EM or AWR reports	Recommended redo log group size
<= 1 MB/sec	1GB
<= 5 MB/sec	4 GB
<= 25 MB/sec	16 GB
<= 50 MB/sec	32 GB
> 50 MB/sec	64 GB

Create Standby Redo Logs on the Primary On-Premises Database

1. Check that Standby Redo Logs(SRLs) are not currently created on the Primary database. If they exist and meet the following requirements no additional SRLs are required.

```
SQL> select count(1) from v$standby_log;

  COUNT(1)
----------
         0
```

2. SRLs should be created the same size as the largest of the Online Redo Logs(ORLs). To gather the size of the ORLs execute the following query. In this example the size of all SRLs will be 4294967296 or 4GB.

```
SQL> select max(bytes) from v$log;

MAX(BYTES)
----------
4294967296
```

3. Additionally, the MAA Best practice for standby redo logs is that there is the same number of groups as there are groups of online redo logs **plus 1.** Therefore, if there are three ORL groups like the example below , 4 SRL groups should be created.

```
SQL> select thread#,count(group#) from v$log group by thread#;

   THREAD# COUNT(GROUP#)
---------- -------------
         1             3
```

4. We also need to know the highest group number for current ORLs. Since ORLs and SRLs are in the same namespace we cannot share group numbers. In this example the numbering of SRL groups will begin at 4.

```
SQL> select max(group#) from v$log;

MAX(GROUP#)
-----------
          3
```

NOTE: If there is more than one thread in the previous query the primary database is a Real Application Cluster (RAC) database and the following steps should be converted to single instance.

5. Finally, add the SRL groups using the information gathered.

```
SQL> alter database add standby logfile thread 1 group 4 ('+RECO') size
4294967296, group 5 ('+RECO') size 4294967296, group 6 ('+RECO') size
4294967296, group 7 ('+RECO') size 4294967296;
```

Replace '+RECO' with the ASM diskgroup used or <path>/<filename> if ASM is not used (commonly the same location as ORLs)

The MAA Best practice is that SRLs are not duplexed like ORLs

Set TCP socket size

Check the TCP socket sizes for the on premises system as well as the cloud instance with the following command run as root

```
# /sbin/sysctl -a | egrep net.core.[w,r]mem_max
net.core.wmem_max = 2097152
net.core.rmem_max = 4194304

# /sbin/sysctl -a | egrep net.core.[w,r]mem_max
net.core.wmem_max = 1048576
net.core.rmem_max = 4194304
```

If necessary adjust all socket size maximums to 10MB or 10485760. For on premises systems consult your operating system guide for details about how to accomplish this. For the cloud instance edit the /etc/sysctl.conf file settings for net.core.wmem_max and net.core.rmem_max. If the values between on premises and OPC do not match, the network protocol will negotiate the lower of the two values. Therefore, the values between sites is not required to match though that is recommended in order to attain optimal transport performance.

```
net.core.rmem_max = 10485760

net.core.wmem_max = 10485760
```

As documented in the OPC documentation you can connect to the provisioned cloud instance as root by connecting to the ops user using the same ssh key as oracle followed by 'sudo su -'

```
ssh opc@<cloudip> [-i <key>]
sudo su -
```

Standby Instantiation and Configuration

The following steps can be used to instantiate a standby database on the Oracle Cloud. These steps assume the prerequisites defined previously are met.

1. Remove default database from cloud

The initial deployment of a cloud service creates a default database with the name provided during the instance creation. In our example, we used STBY to retain the other scripts and configurations in the cloud VM. This database must be removed using the following command. The password used is the password given in the deployment web tool.

```
$dbca -silent -deleteDatabase -sourceDB STBY -sysDBAUserName sys -
sysDBAPassword <passwd>
```

Additionally, edit the /home/oracle/.bashrc file setting ORACLE_SID to the intended instance name

```
$ vi ~/.bashrc
#change
export ORACLE_SID=STBY
#to
export ORACLE_SID=<target sid name>
```

2. Make sure that you configured network port (1521 or the custom port) and enabled the protocol as explained in step #2 of the deployment process.

3. Configure Oracle Net Encryption

To protect from plaintext from being visible on the WAN place the following entries in the sqlnet.ora file on all on premises and cloud machines which are located in $ORACLE_HOME/network/admin.

```
SQLNET.ENCRYPTION_SERVER = requested (for On-Premises)
SQLNET.ENCRYPTION_SERVER = required (for Cloud)
SQLNET.ENCRYPTION_TYPES_SERVER = (RC4_256, AES256) (both)
SQLNET.ENCRYPTION_CLIENT = requested (both)
SQLNET.ENCRYPTION_TYPES_CLIENT = (RC4_256, AES256) (both)
```

4. Configure TNS entries for redo transport

» Entries for each database are needed in both primary and standby tnsnames.ora files for proper redo transport. Use the following example, replacing bolded values with values relevant for the configuration.

» NOTE: The primary database may already have a TNS entry in the on-premises tnsnames.ora with a server name for the HOST. In this case simply change the server name in that entry to use the IP address for the host instead.

» NOTE: IP addresses are used since there is no DNS between on premises and cloud environments to resolve server names to IP addresses.

```
<primary db_unique_name> =
 (DESCRIPTION =
   (SDU=65536)
   (RECV_BUF_SIZE=10485760)
   (SEND_BUF_SIZE=10485760)
   (ADDRESS = (PROTOCOL = TCP)(HOST = <primary IP address>)(PORT =
{1521|<port#>}))
   (CONNECT_DATA =
     (SERVER = DEDICATED)
     (SERVICE_NAME = <primary db_unique_name>)
   )
  )
```

```
<standby db_unique_name> =
 (DESCRIPTION =
   (SDU=65536)
   (RECV_BUF_SIZE=10485760)
   (SEND_BUF_SIZE=10485760)
   (ADDRESS = (PROTOCOL = TCP)(HOST = <standby IP address>)(PORT =
{1521|<port#>}))
   (CONNECT_DATA =
     (SERVER = DEDICATED)
     (SERVICE_NAME = <standby db_unique_name>)
   )
  )
```

5. Configure static listeners

» A static listener is needed for initial instantiation of a standby database. The static listener enables remote connection to an instance while the database is down in order to start a given instance. The following steps will aid in configuring the static listener on the cloud VM. See MOS 1387859.1 for additional details.

» Add the following entry to the listener.ora on the cloud VM after replacing the variables. The listener.ora resides in $ORACLE_HOME/network/admin. This listener entry should be deleted after instantiation when using Oracle 12c.

```
SID_LIST_LISTENER =
(SID_LIST =
  (SID_DESC =
   (GLOBAL_DBNAME = <Local Instance name>)
   (ORACLE_HOME = <Local Oracle Home>)
   (SID_NAME = <Local Instance Name>)
  )
)
```

» For 11.2 configurations, a static listener is also required for Data Guard Broker. Add the following entry to the listener.ora on premises after replacing the variables.

```
SID_LIST_LISTENER =
(SID_LIST =
 (SID_DESC =
   (GLOBAL_DBNAME = <Local Instance_name>_DGMGRL)
   (ORACLE_HOME = <Local Oracle Home>)
   (SID_NAME = <Local Instance Name>)
 )
)
```

» If the configuration is 11.2 also add the DGMGRL SID_DESC block to the SID_LIST on the cloud VM.

Static "_DGMGRL" entries are no longer needed as of Oracle Database 12.1.0.2 in Oracle Data Guard Broker configurations that are managed by Oracle Restart, RAC One Node or RAC as the Broker will use the clusterware to restart an instance.

» Finally reload the listeners with:

```
$ $ORACLE_HOME/bin/lsnrctl reload <listener name>
```

6. Create Audit Directory

» Create the audit file directory for the cloud standby database as oracle:

```
$ mkdir -p /u01/app/oracle/admin/<STANDBY DBNAME>/adump
```

7. Create Auxiliary Database Password File and init.ora.

» Create a password file for the auxiliary instance which will be used to instantiate the standby database.

```
$ $ORACLE_HOME/bin/orapwd file='$ORACLE_HOME/dbs/orapw<INSTANCE_NAME>'
password=${passwd} force=y
```

» Additionally, create an init file to start the auxiliary instance

```
$ echo “db_name=<primary db_name>” > /tmp/aux.pfile
$ echo “db_unique_name=<standby db_name>” >> /tmp/aux.pfile
$ echo “sga_target=800M” >> /tmp/aux.pfile
```

NOTE: The size of sga size in this pfile does not need to match the primary; a new spfile will be generated during the instantiation process with the proper values. This auxiliary pfile is only used to start the auxiliary instance.

8. Start the Auxiliary Instance

The auxiliary instance is a temporary instance (just memory structures) which enables communication between the primary and standby sites. Think of it as a place-holder until the primary is able to copy its files making the auxiliary instance a database. Start this auxiliary instance with the following command:

```
$ export ORACLE_SID=<standby instance name (STBY)>
$ sqlplus “/ as sysdba”

SQL> startup nomount pfile='/tmp/aux.pfile'
ORACLE instance started.

Total System Global Area  835104768 bytes
Fixed Size                  2856056 bytes
Variable Size             377490312 bytes
Database Buffers          444596224 bytes
Redo Buffers               10162176 bytes
```

NOTE: The pfile used is the one generated in the previous step

9. If TDE is configured on premises or desired in the configuration, refer to Appendix H for further configuration details before proceeding.

Note: The Advanced Security Option is required for TDE on premises

10. Instantiate the Standby Database via RMAN Duplicate From Active Database. Appendix C provides a script to instantiate the standby in the cloud. Set these additional parameters manually after the standby database is created in the cloud and restart the databases. It is recommended these values be set at both the primary and standby databases but minimally the standby is necessary.

```
alter system set DB_FLASHBACK_RETENTION_TARGET=120 scope=both sid='*';
alter system set remote_login_passwordfile='exclusive' scope=spfile
sid='*';
alter system set DB_BLOCK_CHECKSUM=FULL;
alter system set DB_BLOCK_CHECKING=MEDIUM;
alter system set DB_LOST_WRITE_PROTECT=TYPICAL;
alter system set LOG_BUFFER=256M scope=spfile sid='*';
```

NOTE: DB_BLOCK_CHECKSUM, DB_BLOCK_CHECKING and DB_LOST_WRITE_PROTECT are parameters which assist in preventing block corruption. The listed settings are recommended for the protection benefits though DB_BLOCK_CHECKING can have an impact on performance which should be considered when choosing a value. A value of FALSE turns DB_BLOCK_CHECKING off.

NOTE: Setting LOG_BUFFER to 256MB will aid asynchronous redo transport in reading redo from memory instead of having to do disk I/Os from the online redo logs.

11. Register the database Oracle Restart

Now that the database has been created the database must be registered with Oracle Restart.

<u>On the standby</u>

```
$srvctl add database -d <standby db_unique_name> -c SINGLE <result of
hostname -s> -oh <oracle home> -r physical_standby -s <mount|open>
```

12. Configure client Failover

Refer to Appendix G for configuring client failover

13. Configure Data Guard Broker

» Appendix D provides a script which will set up Data Guard Broker. The contents of Appendix D can be pasted into a file on the on-premises server and executed after setting the designated variables. Variables which already have values (non-'<>' values) should be left as is as they are representative of the cloud deployment and do not need to be changed. The setting for variable which do need to be set can be the same values as used in the RMAN duplicate script.

Appendix C – Creating Standby Database using RMAN DUPLICATE

The contents of Appendix C can be pasted into a shell script of any arbitrary name on the cloud server and executed after setting the designated variables. Variables which already have values(non ‘<>’ values) should be left as is as they are representative of the cloud deployment and do not need to be changed.

NOTE: If files are spread across multiple mount points on the primary database additional changes will be required as noted in Appendix C.

Additional details of the duplicate from active process can be seen in MOS 1617946.1

RMAN Duplicate from Active Script

Copy the following script, modify for your environment & paste to the shell to execute it.

```
#!/bin/bash
function gen_rman {
echo "connect target sys/${passwd}@${PREMISES_DBNM}"
echo "connect auxiliary sys/${passwd}@${CLOUD_DBNM}"
echo "run {"
echo "allocate channel prmy1 type disk;"
echo "allocate auxiliary channel stby1 type disk;"
echo "allocate auxiliary channel stby2 type disk;"
echo "allocate auxiliary channel stby3 type disk;"
echo "allocate auxiliary channel stby4 type disk;"
echo "duplicate target database for standby from active database"
echo "spfile"
echo "PARAMETER_VALUE_CONVERT= '${PREMISES_DBNM}', '${CLOUD_DBNM}'"
echo "set db_unique_name='${CLOUD_DBNM}'"
echo "set
control_files='/u02/app/oracle/oradata/${CLOUD_DBNM}/control01.ctl','/u03
/app/oracle/fast_recovery_area/${CLOUD_DBNM}/control02.ctl'"
echo "set audit_file_dest='/u01/app/oracle/admin/${CLOUD_DBNM}/adump'"
echo "set
local_listener='(ADDRESS=(PROTOCOL=tcp)(HOST=${CLOUD_IP})(PORT=${CLOUD_PO
RT}))'"
echo "set fal_server='${PREMISES_DBNM}'"
echo "set log_file_name_convert='${PREMISES_FILE_LOC}/${PREMISES_DBNM}',
'${CLOUD_FILE_LOC}/${CLOUD_DBNM}',
'${PREMISES_RECOV_LOC}/${PREMISES_DBNM}',
'${CLOUD_REDO_LOC}/${CLOUD_DBNM}'"
echo "set db_file_name_convert='${PREMISES_FILE_LOC}/${PREMISES_DBNM}',
'${CLOUD_FILE_LOC}/${CLOUD_DBNM}'"
echo "set db_create_online_log_dest_1='${CLOUD_FILE_LOC}'"
echo "set db_create_online_log_dest_2='${CLOUD_REDO_LOC}'"
echo "set db_create_file_dest='${CLOUD_FILE_LOC}'"
echo "set db_recovery_file_dest='${CLOUD_RECOV_LOC}'"
echo "set diagnostic_dest='/u01/app/oracle/diag/rdbms'"
echo "set db_domain=''"
echo "section size 10M"
echo "dorecover"
echo ";"
echo "}"
}

export passwd='<sys password>'

export PREMISES_DBNM='<Premise db_unique_name>'
export PREMISES_IP='<Premises IP address>'
export PREMISES_PORT='1521'

export CLOUD_DBNM='<STANDBY db_unique_name>'
export CLOUD_IP='<Cloud IP address>'
export CLOUD_PORT='{1521|<port#>}'

export PREMISES_FILE_LOC='<Location of datafiles on premises>'
export PREMISES_RECOV_LOC='<Location of recovery files on premises>'
export CLOUD_FILE_LOC='/u02/app/oracle/oradata'
export CLOUD_RECOV_LOC='/u03/app/oracle/fast_recovery_area'
export CLOUD_REDO_LOC='/u04/app/oracle/redo'

ssh ${CLOUD_IP} "mkdir -p /u01/app/oracle/admin/${CLOUD_DBNM}/adump
/u02/app/oracle/oradata/${CLOUD_DBNM}
/u03/app/oracle/fast_recovery_area/${CLOUD_DBNM}"
gen_rman > /tmp/${CLOUD_DBNM}.rman
scp /tmp/${CLOUD_DBNM}.rman ${CLOUD_IP}:/tmp

rman << EOFdupe
@/tmp/${CLOUD_DBNM}.rman
EOFdupe
#--------------END of Script ---------------------
```

NOTE: If files are spread across multiple mount points additional entries will be required for db_file_name_convert. For each different path to the data files a ***'<primary path>/${PREMISES_DBNM}' , '${CLOUD_FILE_LOC}/${CLOUD_DBNM}'*** *pair is required. Retrieve the data file paths from the primary database with '*`select name from v$datafile;`*'*

NOTE: If log files are spread across multiple mount points additional entries will be required for log_file_name_convert. For each different path to the log files a ***'<primary path>/' , '${CLOUD_FILE_LOC}/${CLOUD_DBNM}'*** *pair is required. Retrieve the log file paths from the primary database with '*`select member from v$logfile;`*'*

NOTE : The log_file_name_convert and db_file_name_convert parameters must be set on the primary in the event of a role transition. Please set them accordingly.

Appendix D – Data Guard Broker

Data Guard Broker Configuration Script

NOTE: Paste this script into a file on the on premises machine, set the variable requiring input(those with '<>') and run the script. **This assumes there is no existing Data Guard Broker configuration on the primary.**

Copy the following script, modify for your environment & paste to the shell to execute it.

```
#!/bin/bash
function brokeradd {
sqlplus -s sys/${passwd}@${1} as sysdba <<EOFBKRSQLP
alter system set dg_broker_start=FALSE;
alter system set dg_broker_config_file1='${3}/${1}/dr1.dat';
alter system set dg_broker_config_file2='${4}/${1}/dr2.dat';
alter system set dg_broker_start=TRUE;
exit
EOFBKRSQLP
#
sqlplus -s sys/${passwd}@${2} as sysdba <<EOFBKRSQLS
alter system set dg_broker_start=FALSE;
alter system set dg_broker_config_file1='${5}/${2}/dr1.dat';
alter system set dg_broker_config_file2='${6}/${2}/dr2.dat';
alter system set dg_broker_start=TRUE;
exit
EOFBKRSQLS
#
sleep 30
dgmgrl sys/${passwd}@${1} <<EOFBRKR
 create configuration 'DGconfig' as primary database is ${1} connect
identifier is ${1};
 add database ${2} as connect identifier is ${2};
 edit database ${1} set property RedoRoutes='(LOCAL:${2} ASYNC)'; # must
be removed for 11g
 edit database ${2} set property RedoRoutes='(LOCAL:${1} ASYNC)'; # must
be removed for 11g
 EDIT CONFIGURATION SET PROTECTION MODE AS MaxPerformance;
 enable configuration;
exit
EOFBRKR
}
##########
#MAIN
##########
  echo "#################"
  echo "Add Broker config"
  echo "#################"
export PREMISES_DBNM='<Premise db_unique_name>'
export CLOUD_DBNM='<STANDBY db_unique_name>'
export PREMISES_FILE_LOC='<Location of datafiles on premises>'
export PREMISES_RECOV_LOC='<Location of recovery files on premises>'
export CLOUD_RECOV_LOC='/u03/app/oracle/fast_recovery_area'
export CLOUD_REDO_LOC='/u04/app/oracle/redo'
export passwd='<sys passwd>'

  brokeradd ${PREMISES_DBNM} ${CLOUD_DBNM} ${PREMISES_FILE_LOC}
${PREMISES_RECOV_LOC} ${CLOUD_FILE_LOC} ${CLOUD_RECOV_LOC} ${passwd}
```

#--------------END of Script ---------------------

NOTE: Variables settings used for the RMAN duplicate script can and should be used here.

NOTE: Restart the new standby database after the configuration has been enabled.

Appendix E – Health Check Queries

As noted in the section Data Guard Health Check a periodic check of best practices should be executed on the Data Guard configuration to ensure best practices are followed. It is strongly recommended that the automated tools outlined in the 'Data Guard Health Check' section be used as they are updated regularly. However, the following queries can be run in absence of the tools.   Also refer to Monitoring a Data Guard Configuration (Doc ID 2064281.1)

Standby Redo Logs(SRL) Health Check

There are a few different best practices for SRLs

1. Number of SRL groups should equal the number of Online Redo Log(ORL) groups +1 for each thread. Run the following queries on the primary and standby database.

```
SQL> select thread#,count(group#)from v$log group by thread#;

   THREAD# COUNT(GROUP#)
---------- -------------
         1             4
SQL> select thread#,count(group#)from v$standby_log group by thread#;

   THREAD# COUNT(GROUP#)
---------- -------------
      1             5
```

2. All log files (ORL & SRL) should be of the same size. Run the following queries, which should return the same single value.

```
SQL> select distinct bytes from v$log;

     BYTES
----------
4294967296

SQL> select distinct bytes from v$standby_log;

     BYTES
----------
4294967296
```

3. SRL groups should have only one member. Run the following query to verify that the count column is one for all groups.

```
SQL> select group#,count(member) from v$logfile where type='STANDBY'
group by group#;

    GROUP# COUNT(MEMBER)
---------- -------------
         5             1
         6              1
         7              1
         8              1
         9              1
```

Flashback Database and Forced Logging

It is recommended that flashback database be enabled for easy reinstatement from failover operations. Run the following query to verify, the result should be ‘YES’.

```
SQL> select flashback_on from v$database;

FLASHBACK_ON
------------------
YES
```

It is recommended that force logging be enabled to prevent the loss of data when nologging queries are run on the primary database. Verify with the following query.

```
SQL> select force_logging from v$database;

FORCE_LOGGING
---------------------------------------
YES
```

Additional parameters

Check that the following parameters are set accordingly. DB_BLOCK_CHECKING can be left at FALSE if it is deemed too much of a performance impact on recovery.

```
SQL> show parameter checking

NAME                                 TYPE        VALUE
------------------------------------ ----------- ----------------------------
--
db_block_checking                    string      MEDIUM

SQL> show parameter checksum

NAME                                 TYPE        VALUE
------------------------------------ ----------- ----------------------------
--
db_block_checksum                    string      FULL

SQL> show parameter lost_write

NAME                                 TYPE        VALUE
------------------------------------ ----------- ----------------------------
--
db_lost_write_protect                string      typical

SQL> show parameter disk_asynch_io

NAME                                 TYPE        VALUE
------------------------------------ ----------- -------------------------
----
```

```
disk_asynch_io                       boolean     TRUE
```

LOG_ARCHIVE_MAX_PROCESSES

The setting for LOG_ARCHIVE_MAX_PROCESSES should be equal to the number of remote destinations (likely 1) Plus the number of threads/instances (greater than 1 for RAC only). The following queries can be used to help determine the current and proper setting.

```
SQL> show parameter log_archive_max_processes

NAME                                 TYPE        VALUE

------------------------------------ ----------- ------------------------

log_archive_max_processes            integer     2


SQL> select ((select count(1) from v$archive_dest where TARGET='STANDBY') +
(select count(distinct thread#) from v$log)) PROPER_LOG_ARCHIVE_MAX_PROC from
dual;

PROPER_LOG_ARCHIVE_MAX_PROC
---------------------------
                          2
```

Appendix F – Converting Standby Database to a Snapshot Standby

A snapshot standby is a fully updatable standby database that is created from a physical standby database. On snapshot standby databases, redo data is received but not applied until the snapshot standby database is converted back to a physical standby database.

The benefits of using a snapshot standby database include the following:

1. It provides an exact replica of a production database for development and testing purposes, while maintaining data protection at all times. You can use the Oracle Real Application Testing option to capture primary database workload and then replay it for test purposes on the snapshot standby.

2. It can be easily refreshed to contain current production data by converting to a physical standby and resynchronizing.

Follow the steps below to convert a physical standby database to a snapshot standby

Convert the standby to a snapshot standby and validate

Via Data Guard broker issue the following commands

```
DGMGRL> convert database 'stby' to snapshot standby;

DGMGRL> SHOW CONFIGURATION;

Configuration - DRSolution

  Protection Mode: MaxPerformance
  Databases:
    prmy  - Primary database
      stby  - Snapshot standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS
```

NOTE: A snapshot standby database cannot be the target of a switchover or failover. A snapshot standby database must first be converted back into a physical standby database before performing a role transition to it.

Convert the snapshot standby back into a physical standby database

Via Data Guard broker issue the following commands

```
DGMGRL> CONVERT DATABASE 'stby' to PHYSICAL STANDBY;
```

f Data Guard broker is not configured, see Managing a Snapshot Standby Database in the Standard documentation for 11.2[32] or 12.1 [33]

32 http://docs.oracle.com/cd/E11882_01/server.112/e41134/manage_ps.htm#SBYDB4801

33 http://docs.oracle.com/database/121/SBYDB/manage_ps.htm#SBYDB4801

Appendix G – Configuring Client Failover

Automating client failover, the process by which clients are reconnected to the active primary database after a failure, includes relocating database services to the new primary database as part of a Data Guard failover, notifying clients that a failure has occurred in order to break them out of TCP timeout, and redirecting clients to the new primary database. Configuration details are thoroughly covered in the papers MAA Best Practices for Client Failover for Oracle Database11g[34] and for Oracle Database 12c[35]. Please consult these papers and configure your environment appropriately.

34 http://www.oracle.com/technetwork/database/features/availability/maa-wp-11gr2-client-failover-173305.pdf

35 http://www.oracle.com/technetwork/database/availability/client-failover-2280805.pdf

Appendix H – Creating and Dispersing Wallets

Creating and dispersing wallets in 12c

1. Create encryption wallet

Set the wallet location in the sqlnet.ora on all nodes of primary and standby.

```
ENCRYPTION_WALLET_LOCATION =
 (SOURCE = (METHOD = FILE)
 (METHOD_DATA =
 (DIRECTORY = /u01/app/oracle/admin/TDE/$ORACLE_SID)
 )
 )
```

NOTE: Using ORACLE_SID in the directory path ensures that all databases do not share the wallet. If there is just one database on the system the ORACLE_SID is not necessary.

2. Create the corresponding directory on all nodes with the proper ORACLE_SID.

```
$mkdir -p /u01/app/oracle/admin/TDE/$ORACLE_SID
```

3. Initiate a new SQL*Plus session. This causes the changes to sqlnet.ora to be picked up.

4. Create the password-based keystore

```
SQL>ADMINISTER KEY MANAGEMENT CREATE KEYSTORE
'/u01/app/oracle/admin/TDE/<ORACLE_SID>' IDENTIFIED BY "AbCdEfGh!";
```

NOTE: Ensure the password string in double quotation marks (" ").

5. Open the wallet

```
SQL>ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "AbCdEfGh!";
```

6. Set the Encryption Key

```
SQL>ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY "AbCdEfGh!" WITH BACKUP
USING 'TDE';
```

7. Create Auto-login wallet

An Auto-login wallet removes the requirement of manually opening the wallet when the database is started.

```
SQL>ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE
'/u01/app/oracle/admin/TDE/$ORACLE_SID' IDENTIFIED BY “AbCdEfGh!”;
```

8. Copy the files generated in the keystore directory to all nodes of the primary and standby.

Copy files to each node:

```
$scp /u01/app/oracle/admin/TDE/$ORACLE_SID/*
oracle@<host>:/u01/app/oracle/admin/TDE/<SID_NAME>/
```

9. Ensure the wallet is open on all nodes

```
SQL> select * from gv$encryption_wallet;
 INST_ID WRL_TYPE WRL_PARAMETER
STATUS
```

```
---------- -------------------- --------------------------------------------
---------------------------------- ------------------
         1 file           /u01/app/oracle/admin/TDE/primary1
OPEN
```

Creating and dispersing wallets in 11gR2

1. Create encryption wallet

Set the wallet location in the sqlnet.ora on all nodes of primary and standby.

```
ENCRYPTION_WALLET_LOCATION =
   (SOURCE = (METHOD = FILE)
     (METHOD_DATA =
       (DIRECTORY = /u01/app/oracle/admin/TDE/$ORACLE_SID)
     )
   )
```

NOTE: Using ORACLE_SID in the directory path ensures that all databases do not share the wallet. If there is just one database on the system the ORACLE_SID is not necessary.

2. Create the corresponding directory on all nodes with the proper ORACLE_SID.

```
$mkdir -p /u01/app/oracle/admin/TDE/$ORACLE_SID
```

3. Initiate a new SQL*Plus session. This causes the changes to sqlnet.ora to be picked up.

4. Set the Master Encryption Key

```
SQL>ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY "AbCdEfGh!";
```

NOTE: Ensure the password string in double quotation marks (" ").

5. Create Auto-login wallet

An Auto-login wallet removes the requirement of manually opening the wallet when the database is started.

```
$ orapki wallet create -wallet /u01/app/oracle/admin/TDE/$ORACLE_SID -
auto_login
```

6. Copy the files generated in the keystore directory to all nodes of the primary and standby.

Copy files to each node:

```
$ scp /u01/app/oracle/admin/TDE/$ORACLE_SID/*
oracle@<host>:/u01/app/oracle/admin/TDE/<SID_NAME>/
```

7. Ensure the wallet is open on all nodes

```
SQL> select * from gv$encryption_wallet;
 INST_ID WRL_TYPE WRL_PARAMETER
STATUS
---------- -------------------- --------------------------------------------
---------------------------------- ------------------
 1 file        /u01/app/oracle/admin/TDE/primary1
OPEN
```

Appendix I – Standby Instantiation From Database Backup Cloud Service

This section provides the steps required to create a database from the Oracle Database Backup Cloud Service. You need to know DBID of the source database and the CONTROLFILE must be backed up with AUTOBACKUP.

1. Install the Cloud Backup Tool to the Cloud VM.

The library and wallet must be created on the OPC VM in the same way it was done on the primary database machine. This step downloads the library (libopc.so), creates an Oracle wallet and opc<SID>.ora configuration file.

Example:

```
$ mkdir -p /u01/app/oracle/OPC/wallet
$ mkdir -p /u01/app/oracle/OPC/lib
$ export ORACLE_SID=stby
$ export ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1
$ java -jar opc_install.jar -serviceName <defined service name> -
identityDomain <backup service domain> -opcId '<user@company.com>' -opcPass
'<OPC password>' -walletDir /u01/app/oracle/OPC/wallet -libDir
/u01/app/oracle/OPC/lib
Oracle Database Cloud Backup Module Install Tool, build 2015-05-12
Oracle Database Cloud Backup Module credentials are valid.
Oracle Database Cloud Backup Module wallet created in directory
/u01/app/oracle/OPC/wallet.
Oracle Database Cloud Backup Module initialization file
/u01/app/oracle/product/12.1.0/dbhome_1/dbs/opcstby.ora created.
Downloading Oracle Database Cloud Backup Module Software Library from file
opc_linux64.zip.
Downloaded 23169388 bytes in 20 seconds.
Download complete.
```

2. Retrieve the DBID from the primary on-premises database.

```
SQL> select dbid from v$database;

      DBID

----------

 812240971
```

3. On the OPC VM, start the instance, set the DBID, password and restore the spfile from the backup and create a pfile from the restored spfile.

```
$ rman target /

Recovery Manager: Release 12.1.0.2.0 - Production on Fri Jul 24 12:25:24 2015
Copyright (c) 1982, 2014, Oracle and/or its affiliates.  All rights reserved.

connected to target database (not started)

RMAN> startup nomount force
startup failed: ORA-01078: failure in processing system parameters
```

```
LRM-00109: could not open parameter file
'/u01/app/oracle/product/12.1.0/dbhome_1/dbs/initstby.ora'

starting Oracle instance without parameter file for retrieval of spfile
Oracle instance started

Total System Global Area    1073741824 bytes

Fixed Size                     2932632 bytes
Variable Size                281018472 bytes
Database Buffers             784334848 bytes
Redo Buffers                   5455872 bytes

RMAN> set dbid 812240971;
executing command: SET DBID

RMAN> set decryption identified by <password used when taking backup>;
executing command: SET decryption

RMAN> run {
allocate channel dev1 device type sbt
parms='SBT_LIBRARY=/u01/app/oracle/OPC/lib/libopc.so';
restore spfile TO '/tmp/spfile.ora' from autobackup;
}

allocated channel: dev1
channel dev1: SID=12 device type=SBT_TAPE
channel dev1: Oracle Database Backup Service Library VER=3.15.1.16

Starting restore at 24-JUL-15

channel dev1: looking for AUTOBACKUP on day: 20150724
channel dev1: looking for AUTOBACKUP on day: 20150723
channel dev1: AUTOBACKUP found: c-812240971-20150723-00
channel dev1: restoring spfile from AUTOBACKUP c-812240971-20150723-00
channel dev1: SPFILE restore from AUTOBACKUP complete
Finished restore at 24-JUL-15
released channel: dev1

RMAN> create pfile='/tmp/pfile' from spfile='/tmp/spfile.ora';
Statement processed

RMAN> exit

Recovery Manager complete.
```

4. Edit the pfile, minimally changing the following:
   - Remove all double underscore '<primary>.__*' parameters
   - Change control_files using /u02 and /u03 mount points

     ```
     *.control_files='/u02/app/oracle/oradata/<db_unique_name>/control01.ctl','
     /u03/app/oracle/fast_recovery_area/<db_unique_name>/control02.ctl'
     ```

- Change or set the following as listed

```
*.db_create_file_dest='/u02/app/oracle/oradata'
*.db_create_online_log_dest_1='/u02/app/oracle/oradata'
       *.db_create_online_log_dest_2='/u04/app/oracle/redo'
*.db_recovery_file_dest='/u03/app/oracle/fast_recovery_area'
*.dg_broker_start=FALSE
*.log_archive_dest_1='location=USE_DB_RECOVERY_FILE_DEST','valid_for=(ALL_LOG
FILES, ALL_ROLES)'
*.diagnostic_dest='/u01/app/oracle'
```

- Remove the following if they exist:

```
*.dg_broker_config_file1
*.dg_broker_config_file2
*.db_domain
*.fal_server
*.log_archive_config
```

- Change the following according to your environment.

```
*.db_unique_name='<standby_db_unique_name>'
*.local_listener='LISTENER_STBY'
*.audit_file_dest='/u01/app/oracle/admin/<standby_db_unique_name>/adump'
*.db_file_name_convert='< premises datafile location >',
'/u02/app/oracle/oradata'
*.log_file_name_convert='< premises logfile location 1 >',
'/u04/app/oracle/redo', '< premises logfile location 2 >',
'/u02/app/oracle/oradata'
```

NOTE: If files are spread across multiple mount points additional entries will be required for db_file_name_convert. For each different path to the data files a ***'<primary path>/${PREMISES_DBNM}' , '${CLOUD_FILE_LOC}/${CLOUD_DBNM}'*** *pair is required. Retrieve the data file paths from the primary database with* '`select name from v$datafile`;'

NOTE: If log files are spread across multiple mount points additional entries will be required for log_file_name_convert. For each different path to the log files a ***'<primary path>/' , '${CLOUD_FILE_LOC}/${CLOUD_DBNM}'*** *pair is required. Retrieve the log file paths from the primary database with* '`select member from v$logfile;`'

5. Create directories for adump, control files and datafiles

```
$mkdir -p /u01/app/oracle/admin/<standby_db_unique_name>/adump
$mkdir -p /u02/app/oracle/oradata/<db_unique_name>/
$mkdir -p /u03/app/oracle/fast_recovery_area/<db_unique_name>
```

6. Create a new spfile from the edited pfile and restart the database.

```
$ rman target /
```

```
Recovery Manager: Release 12.1.0.2.0 - Production on Fri Jul 24 14:31:59
2015

Copyright (c) 1982, 2014, Oracle and/or its affiliates.  All rights
reserved.

connected to target database: PRI (DBID=812240971, not open)

RMAN> create spfile from pfile='/tmp/pfile';

Statement processed

RMAN> startup nomount force;

Oracle instance started

Total System Global Area     838860800 bytes

Fixed Size                     2929936 bytes
Variable Size                608176880 bytes
Database Buffers             222298112 bytes
Redo Buffers                   5455872 bytes
```

7. Restore a standby control file from the backup of the primary

```
RMAN> set dbid 812240971;
executing command: SET DBID

RMAN> set decryption identified by welcome1;
executing command: SET decryption

RMAN> run {
allocate channel dev1 device type sbt
parms='SBT_LIBRARY=/u01/app/oracle/OPC/lib/libopc.so';
restore standby controlfile from autobackup;
alter database mount;
}

allocated channel: dev1
channel dev1: SID=12 device type=SBT_TAPE
channel dev1: Oracle Database Backup Service Library VER=3.15.1.16

Starting restore at 24-JUL-15

channel dev1: looking for AUTOBACKUP on day: 20150724
channel dev1: looking for AUTOBACKUP on day: 20150723
channel dev1: AUTOBACKUP found: c-812240971-20150723-00
channel dev1: restoring control file from AUTOBACKUP c-812240971-20150723-00
channel dev1: control file restore from AUTOBACKUP complete
output file name=/u02/app/oracle/oradata/stby/control01.ctl
output file name=/u03/app/oracle/fast_recovery_area/stby/control02.ctl
Finished restore at 24-JUL-15

Statement processed
released channel: dev1
```

```
RMAN> exit
```

8. Reconnect to RMAN with the new control file and restore the database from the backup service.

```
$ rman target /
Recovery Manager: Release 12.1.0.2.0 - Production on Fri Jul 24 14:31:59 2015
Copyright (c) 1982, 2014, Oracle and/or its affiliates.  All rights reserved.

connected to target database: PRI (DBID=812240971, not open)

RMAN> CONFIGURE CHANNEL DEVICE TYPE 'SBT_TAPE' PARMS
'SBT_LIBRARY=/u01/app/oracle/OPC/lib/libopc.so,
ENV=(OPC_PFILE=/u01/app/oracle/product/12.1.0/dbhome_1/dbs/opcstby.ora)';

old RMAN configuration parameters:
CONFIGURE CHANNEL DEVICE TYPE 'SBT_TAPE' PARMS
'SBT_LIBRARY=/home/oracle/OPC/lib/libopc.so,
ENV=(OPC_PFILE=/home/oracle/app/oracle/product/12.1.0/dbhome_1/dbs/opcpri.ora
)';
new RMAN configuration parameters:
CONFIGURE CHANNEL DEVICE TYPE 'SBT_TAPE' PARMS
'SBT_LIBRARY=/u01/app/oracle/OPC/lib/libopc.so,
ENV=(OPC_PFILE=/u01/app/oracle/product/12.1.0/dbhome_1/db/opcstby.ora)';
new RMAN configuration parameters are successfully stored

RMAN> set decryption identified by <password used for backup>;

executing command: SET decryption
using target database control file instead of recovery catalog

RMAN>  run {
set newname for database to '/u02/app/oracle/oradata/stby/datafile/%b';
restore database;
switch datafile all;
}
2> 3> 4> 5>
executing command: SET NEWNAME

Starting restore at 24-JUL-15
using channel ORA_SBT_TAPE_1
using channel ORA_DISK_1

channel ORA_SBT_TAPE_1: starting datafile backup set restore
channel ORA_SBT_TAPE_1: specifying datafile(s) to restore from backup set
channel ORA_SBT_TAPE_1: restoring datafile 00001 to
/u02/app/oracle/oradata/stby/datafile/system01.dbf
channel ORA_SBT_TAPE_1: restoring datafile 00003 to
/u02/app/oracle/oradata/stby/datafile/sysaux01.dbf
channel ORA_SBT_TAPE_1: restoring datafile 00004 to
/u02/app/oracle/oradata/stby/datafile/undotbs01.dbf
channel ORA_SBT_TAPE_1: restoring datafile 00006 to
/u02/app/oracle/oradata/stby/datafile/users01.dbf
channel ORA_SBT_TAPE_1: reading from backup piece 5dqcoqd8_1_1
channel ORA_SBT_TAPE_1: piece handle=5dqcoqd8_1_1 tag=TAG20150723T104704
channel ORA_SBT_TAPE_1: restored backup piece 1
channel ORA_SBT_TAPE_1: restore complete, elapsed time: 00:02:55
Finished restore at 24-JUL-15

datafile 1 switched to datafile copy
input datafile copy RECID=5 STAMP=885912874 file
name=/u02/app/oracle/oradata/stby/datafile/system01.dbf
datafile 3 switched to datafile copy
input datafile copy RECID=6 STAMP=885912875 file
name=/u02/app/oracle/oradata/stby/datafile/sysaux01.dbf
datafile 4 switched to datafile copy
input datafile copy RECID=7 STAMP=885912876 file
name=/u02/app/oracle/oradata/stby/datafile/undotbs01.dbf
datafile 6 switched to datafile copy
input datafile copy RECID=8 STAMP=885912877 file
name=/u02/app/oracle/oradata/stby/datafile/users01.dbf

RMAN> exit

Recovery Manager complete.
```

9. Complete the Data Guard broker configuration mentioned in Appendix D

Appendix J - rsync_copy.sh script setup and usage

Setup Requirements

The process assumes that DBFS has been configured in the database and is mounted read-write on the primary and read-only on the standby site. In the case of RAC, the DBFS needs only be mounted on one node.

The script requires 1) ssh keys are configured so that no passwords are needed between the servers and 2) rsync rpm exists on each node.

The ssh keys must be configured between the operating system users of the application server and database server which will be involved in the syncing of files. This gives the rsync_copy.sh script the ability to be run via cron without passwords being stored in plain text.

This initial setup can be done manually or via the script's setup mode.

Manual Setup

To complete the setup manually, configure prompt-less ssh between the users of the application server and database server in each direction and install the rsync rpm. To check whether the rsync rpm is already installed run:

```
rpm -qa rsync
```

If anything is returned from this command the rsync rpm has already been installed. For example:

```
$ rpm -qa rsync


rsync-3.0.6-6.el5_11.x86_64
```

If nothing is returned arrange for the rpm to be installed by root. The recommended rsync rpm is included in the dbfs_copy.tar tar ball.

Setup by script

In lieu of manual setup, completing the setup using the dbfs_copy.sh setup mode requires the script be run on each node as the root user. The directories, users and server names are passed as arguments to the script which then completes setup of ssh keys and installs the rsync rpm where necessary. Below is an example of how to complete the setup by the script between the application server *appserver* and the database server *dbserver.*

NOTE: When executing setup, node1 is always the server where the script is running while node 2 is the remote server

As root@dbserver

```
# ./dbfs_copy.sh --setup  --node1--dir <path to dbfs directory> --node1--user
oracle --node1--server dbserver --node2--dir <path to application directory> --
node2--user appuser --node2--server appserver
```

As root@appserver

```
# ./dbfs_copy.sh --setup  --node1--dir <path to application directory> --node1-
-user appuser --node1--server appserver --node2--dir <path to dbfs directory> -
-node2--user oracle --node2--server dbserver
```

Execution

The --sync mode of the script executes rsync from one node to the other. Provide the starting directory username and server name of each node. The direction of the rsync is determined by the --push or --pull flag. Push executes the rsync from node2 to node1 while pull goes in the opposite direction from node1 to node2. This is useful in the event the role of the database changes, simply change the push/pull flag instead of having to rearrange the node inputs.

The rsync command is configured to be recursive from the source directory therefore all subdirectories and files will be copied from source to target.

```
dbfs_copy.sh --sync --node1--dir <path to application directory> --node1--user
appuser --node1--server appserver --node2--dir <path to dbfs directory> --
node2--user oracle --node2--server dbserver --push

dbfs_copy.sh --sync --node1--dir <path to application directory> --node1--user
appuser --node1--server appserver --node2--dir <path to dbfs directory> --
node2--user oracle --node2--server dbserver --pull
```

Scheduling

Below is an example cron entry which will run the script every 30 minutes moving data from node2 to node1 as one might on the primary side of the configuration. With the parameter arranged in this order changing --push to --pull would be used if the database were the standby.

```
00,30 * * * * /home/oracle/scripts/ dbfs_copy.sh --sync --node1--dir <path to
application directory> --node1--user appuser --node1--server appserver --node2-
-dir <path to dbfs directory> --node2--user oracle --node2--server dbserver --
push
```

You can dictate the frequency depending on your application needs.   In some cases, the application files may not need to be synched that frequently or only during planned maintenance activities.

In the event of a role transition, which is covered in more detail below, a final rsync from the remote DBFS to the remote application servers in the cloud will be performed. Therefore no regularly scheduled copy at the remote site is required. However, to reduce the synchronization time of the post-role transition rsync a periodic copy from the remote DBFS to the remote application servers is recommended.

Role Transitions

Switchover:

As switchovers are a planned activity, after stopping the application processing, an additional manual step to make a final copy for the primary application server to the DBFS should be done manually.

1. Stop application processing

2. Execute Data Guard Switchover

3. Re-mount DBFS (new primary read-write, new standby read-only)

4. do a final rsync from DBFS to the remote application server
   dbfs_copy.sh --sync --node1--dir <path to application directory> --node1--user appuser --node1--server appserver --node2--dir <path to dbfs directory> --node2--user oracle --node2--server dbserver --push

5. Switch the direction of the rsync script at each site. e.g. switch the --push --pull flag
   new primary becomes
   dbfs_copy.sh --sync --node1--dir <path to application directory> --node1--user appuser --node1--server appserver --node2--dir <path to dbfs directory> --node2--user oracle --node2--server dbserver **--pull**

   new standby site becomes
   dbfs_copy.sh --sync --node1--dir <path to application directory> --node1--user appuser --node1--server appserver --node2--dir <path to dbfs directory> --node2--user oracle --node2--server dbserver **--push**

Failover

Perform the following steps to complete the failover:

1. Copy files via script from primary app server to primary database DBFS if possible.

   example:

dbfs_copy.sh --sync --node1--dir <path to application directory> --node1--user appuser --node1--server appserver --node2--dir <path to dbfs directory> --node2--user oracle --node2--server dbserver --pull

2. Execute Data Guard failover.

3. open the new primary database (if necessary)

4. Re-mount DBFS (new primary read-write, new standby read-only)

5. do a final rsync from DBFS to the remote application server
   example:
   dbfs_copy.sh --sync --node1--dir <path to application directory> --node1--user appuser --node1--server appserver --node2--dir <path to dbfs directory> --node2--user oracle --node2--server dbserver --push

6. Switch the direction of the rsync script at each site. e.g. switch the --push --pull flag
   Examples:

   new primary becomes
   dbfs_copy.sh --sync --node1--dir <path to application directory> --node1--user appuser --node1--server appserver --node2--dir <path to dbfs directory> --node2--user oracle --node2--server dbserver ***--pull***

   new standby site becomes
   dbfs_copy.sh --sync --node1--dir <path to application directory> --node1--user appuser --node1--server appserver --node2--dir <path to dbfs directory> --node2--user oracle --node2--server dbserver ***--push***