ORACLE UNIVERSITY

Oracle Database 12*c*: RAC and Grid Infra Deployment Workshop

Activity Guide

D87007GC10
Edition 1.0 | November 2014 | D89033

Learn more from Oracle University at **oracle.com/education/**

ORACLE

Authors

Peter Fusek
Jim Womack

Technical Contributors and Reviewers

Allan Graves
Andrey Gusev
Anil Nair
Branislav Valny
Dominique Jeunot
Donna Keesling
Douglas Williams
Harald van Breederode
Harendra Mishra
Harish Nandyala
Janet Stern
Jean-Francois Verrier
Jerry Lee
Jim Williams
Joel Goodman
John McHugh
Jonathan Creighton
Larry Carpenter
Mark Scardina
Markus Michalewicz
Prasad Bagal
Raj Kammend
Rick Wessman
Sean Kim
Soma Prasad
Subhransu Basu
Srinagesh Battula

Editors

Aju Kumar
Anwesha Ray
Malavika Jinka

Graphic Designer

Rajiv Chandrabhanu

Publishers

Veena Narasimhan
Jayanthy Keshavamurthy
Syed Ali

Table of Contents

Practices for Lesson 1: Rolling Database Upgrade Using Transient Logical Standby

Chapter 1

Practices for Lesson 1: Overview

Practice Overview

In this practice, you will perform a rolling database upgrade using a transient logical standby database.

Practice 1-1: Database Rolling Upgrade Using Transient Logical Standby

Overview

In this practice, you will upgrade a two-node Oracle RAC database and an Oracle Data Guard physical standby database running on another two-node cluster. The initial Oracle Database software version on both clusters is 11.2.0.4, and the target software version for the upgraded environment is 12.1.0.1.

The type of upgrade you will perform is known as a rolling database upgrade using a transient logical standby database. The aim of this upgrade method is to maximize database availability during the upgrade process. To achieve this aim, the process requires many operations and role changes to be performed in a precise order.

To simplify the process and reduce the likelihood of administrator error, Oracle has created a script-based utility, which is available for download from My Oracle Support. In this practice, you will use the script to automate various parts of the process and provide advice about what to do when manual intervention is required. The script will also perform many checks along the way to maximize the probability of success.

Tasks

Part 1: Introduction

In the first part of this practice, you will be introduced to the technical environment that you will use to perform a rolling upgrade.

1. Establish a terminal session connected to `c01n01` as the `oracle` user and configure the terminal environment as shown below.

   Note: Make sure that you use `ssh` with the `-X` option. The `-X` option is required to ensure that your terminal environment is correctly configured to run GUI applications and utilities.

```
$ ssh -X oracle@c01n01
[oracle@c01n01 ~]$ . oraenv
ORACLE_SID = [oracle] ? c01orcl
The Oracle base has been set to /u01/app/oracle
[oracle@c01n01 ~]$ export ORACLE_SID=c01orcl1
[oracle@c01n01 ~]$
```

2. `c01n01` is part of a two-node cluster named `cluster01`. The other server in `cluster01` is `c01n02`. The cluster contains an Oracle RAC database named `c01orcl`. Take a moment to explore the configuration and status of `c01orcl`.

```
[oracle@c01n01 ~]$ srvctl config database -d c01orcl -v
Database unique name: c01orcl
Database name: c01orcl
Oracle home: /u01/app/oracle/product/11.2.0/dbhome_1
Oracle user: oracle
Spfile: +DATA/c01orcl/spfilec01orcl.ora
Domain: example.com
Start options: open
Stop options: immediate
```

```
Database role: PRIMARY
Management policy: AUTOMATIC
Server pools: c01orcl
Database instances: c01orcl1,c01orcl2
Disk Groups: DATA,FRA
Mount point paths:
Services:
Type: RAC
Database is administrator managed
[oracle@c01n01 ~]$ srvctl status database -d c01orcl -v
Instance c01orcl1 is running on node c01n01. Instance status:
Open.
Instance c01orcl2 is running on node c01n02. Instance status:
Open.
[oracle@c01n01 ~]$
```

3. Leave your other terminal session open and establish another terminal session connected to `c02n01` as the `oracle` user. Configure the terminal environment as shown in the following:

```
$ ssh -X oracle@c02n01
[oracle@c02n01 ~]$ . oraenv
ORACLE_SID = [oracle] ? c02orcl
The Oracle base has been set to /u01/app/oracle
[oracle@c02n01 ~]$ export ORACLE_SID=c02orcl1
[oracle@c02n01 ~]$
```

4. `c02n01` is part of a two-node cluster named `cluster02`. The other server in `cluster02` is `c02n02`. The cluster contains an Oracle RAC database named `c02orcl`. Take a few minutes to explore the configuration and status of `c02orcl`.

```
[oracle@c02n01 ~]$ srvctl config database -d c02orcl -v
Database unique name: c02orcl
Database name:
Oracle home: /u01/app/oracle/product/11.2.0/dbhome_1
Oracle user: oracle
Spfile:
Domain: example.com
Start options: open
Stop options: immediate
Database role: PHYSICAL_STANDBY
Management policy: AUTOMATIC
Server pools: c02orcl
Database instances: c02orcl1,c02orcl2
Disk Groups: DATA
Mount point paths:
Services:
```

```
Type: RAC
Database is administrator managed
[oracle@c02n01 ~]$ srvctl status database -d c02orcl -v
Instance c02orcl1 is running on node c02n01. Instance status:
Open.
Instance c02orcl2 is running on node c02n02. Instance status:
Open.
[oracle@c02n01 ~]$
```

Note: From your investigation of `c01orcl` and `c02orcl`, you may have seen clues indicating that both databases are part of an Oracle Data Guard configuration. In this configuration, `c01orcl` is nominally the primary database and `c02orcl` is a physical standby database; however, these roles will switch back and forth during different phases of the rolling upgrade process. Oracle Data Guard Broker is also enabled.

5. Use the Data Guard Broker management utility (`dgmgrl`) to explore the configuration and current status of the Oracle Data Guard configuration.

```
[oracle@c01n01 ~]$ dgmgrl sys/oracle_4U
DGMGRL for Linux: Version 11.2.0.4.0 - 64bit Production

Copyright (c) 2000, 2009, Oracle. All rights reserved.

Welcome to DGMGRL, type "help" for information.
Connected.
DGMGRL> show configuration

Configuration - dg_config

  Protection Mode: MaxPerformance
  Databases:
    c01orcl - Primary database
    c02orcl - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS

DGMGRL> show database c01orcl

Database - c01orcl

  Role:            PRIMARY
  Intended State:  TRANSPORT-ON
  Instance(s):
```

```
    c01orcl1
    c01orcl2

Database Status:
SUCCESS

DGMGRL> show database c02orcl

Database - c02orcl

  Role:            PHYSICAL STANDBY
  Intended State:  APPLY-ON
  Transport Lag:   0 seconds (computed 0 seconds ago)
  Apply Lag:       0 seconds (computed 0 seconds ago)
  Apply Rate:      110.00 KByte/s
  Real Time Query: ON
  Instance(s):
    c02orcl1 (apply instance)
    c02orcl2

Database Status:
SUCCESS

DGMGRL> exit
[oracle@c01n01 ~]$
```

Part 2: Grid Infrastructure Upgrade

In any installation, the Oracle Database software version cannot be newer than the Oracle Grid Infrastructure software that supports it. Because of this, the Oracle Grid Infrastructure software must be upgraded before the database upgrade can be performed. In the next part of this practice, you will upgrade the Grid Infrastructure installations across your practice environment.

In line with best practice recommendations, you will upgrade each cluster while it is configured as the standby cluster. This minimizes down time for the primary cluster and the databases it supports.

Here is an outline of the tasks you will perform:

- Upgrade Grid Infrastructure on `cluster02` while it is the standby cluster.
- Perform a Data Guard switchover making `cluster02` the new primary and `cluster01` the standby.
- Upgrade Grid Infrastructure on `cluster01` while it is the standby cluster.
- Perform another switchover to return to the original Data Guard configuration. Note that this switchover is optional, which means that it is not required to complete the upgrade process. However, you will perform the switchover so that `cluster01` can remain the primary cluster and `cluster02` can remain the standby cluster as much as possible.

6. Leave your other terminal sessions open and establish another terminal session connected to `c02n01` as the `grid` user. Configure the terminal environment as shown in the following:

```
$ ssh -X grid@c02n01
[grid@c02n01 ~]$ . oraenv
ORACLE_SID = [grid] ? +ASM1
The Oracle base has been set to /u01/app/grid
[grid@c02n01 ~]$
```

7. Confirm the current active version of Oracle Grid Infrastructure software on `cluster02`.

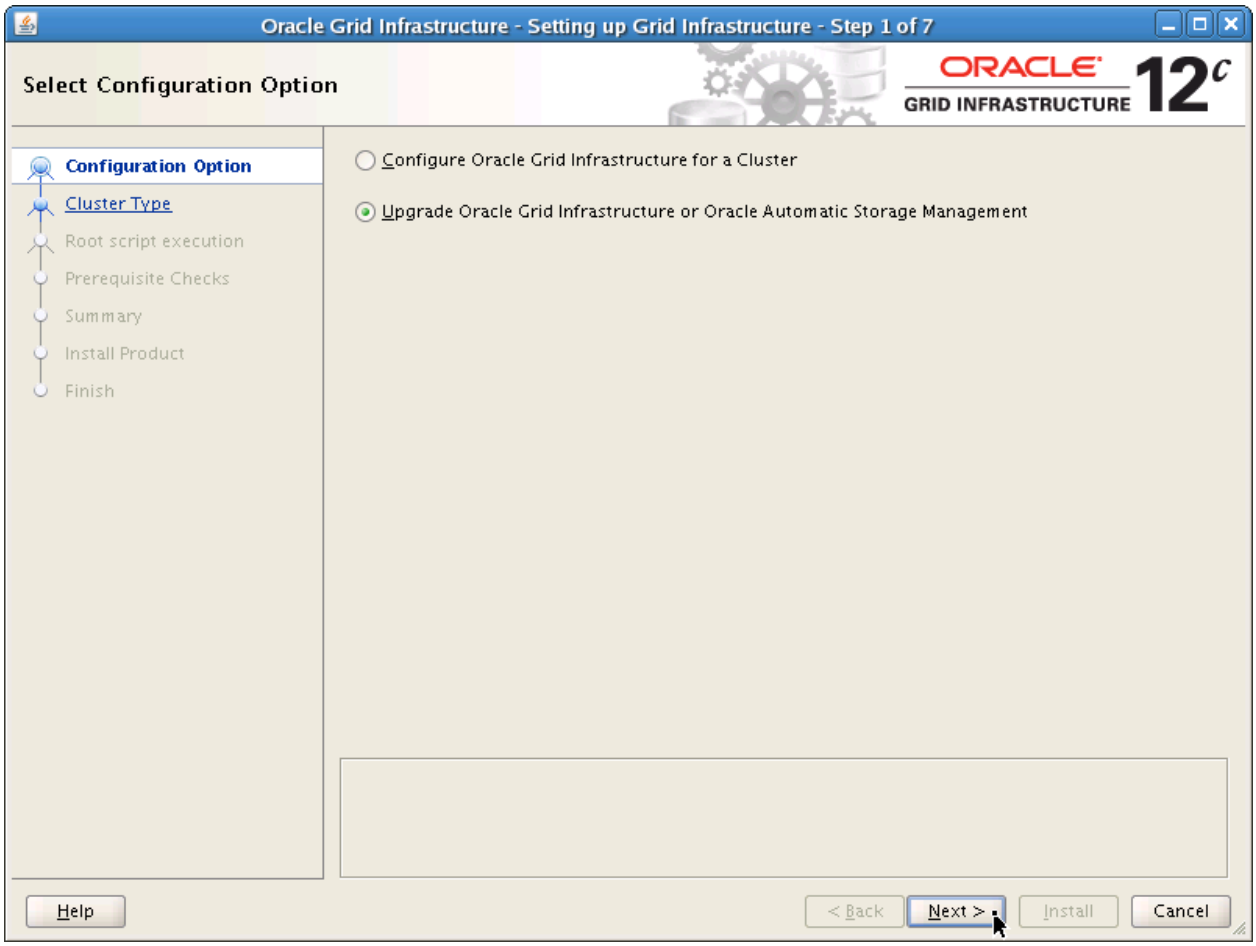
```
[grid@c02n01 ~]$ crsctl query crs activeversion
Oracle Clusterware active version on the cluster is [11.2.0.4.0]
[grid@c02n01 ~]$
```

8. Launch the Oracle Grid Infrastructure 12*c* configuration utility.

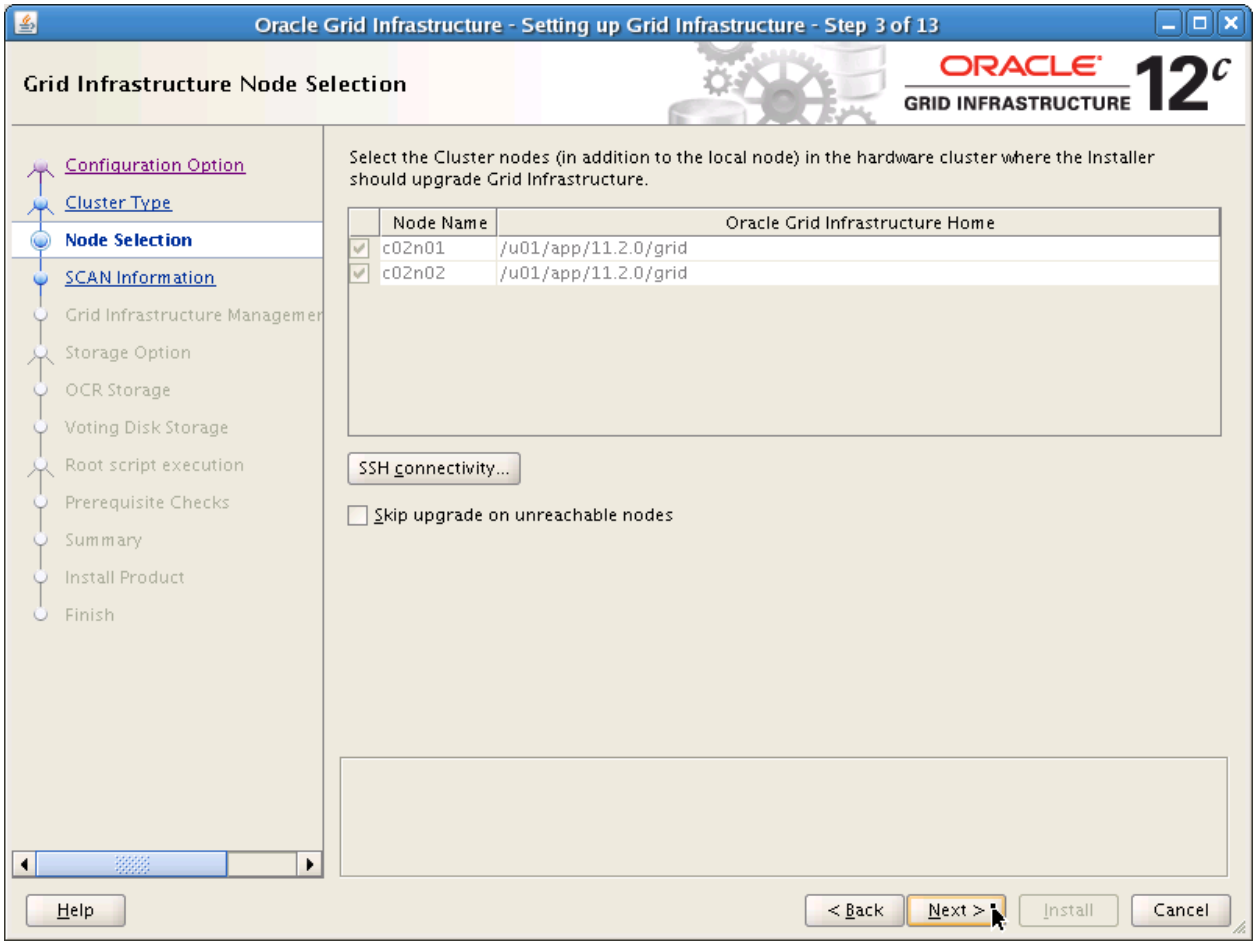
```
[grid@c02n01 ~]$ /u01/app/12.1.0/grid/crs/config/config.sh &
```

Note: In this practice, you will use the Oracle Grid Infrastructure configuration utility to perform the Grid Infrastructure upgrade on each cluster. The configuration utility is launched from a preinstalled copy of the Oracle Grid Infrastructure 12*c* binaries. Your practice environment was built this way in order to speed up the upgrade process. Alternatively, the Oracle Universal Installer can be used to install the upgraded software and upgrade the cluster in one operation. Note that if you use Oracle Universal Installer, you will see essentially the same configuration screens as the ones provided by the Oracle Grid Infrastructure configuration utility.

9. On the Select Configuration Option page, ensure that the option to “Upgrade Oracle Grid Infrastructure or Oracle Automatic Storage Management” is selected and click Next to proceed.

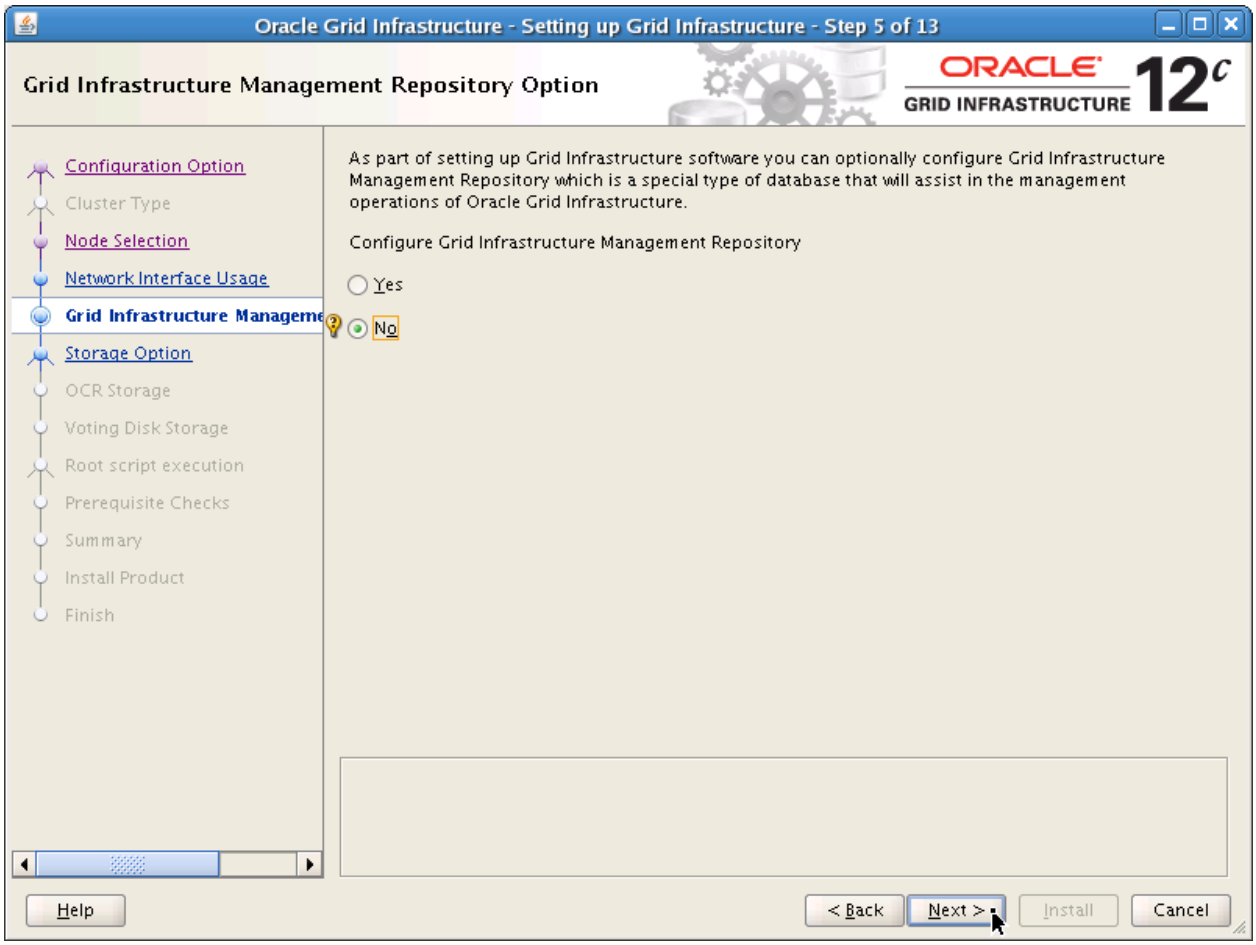


10. On the Grid Infrastructure Node Selection page, ensure that the details on the page match the composition of the cluster and click Next to proceed.



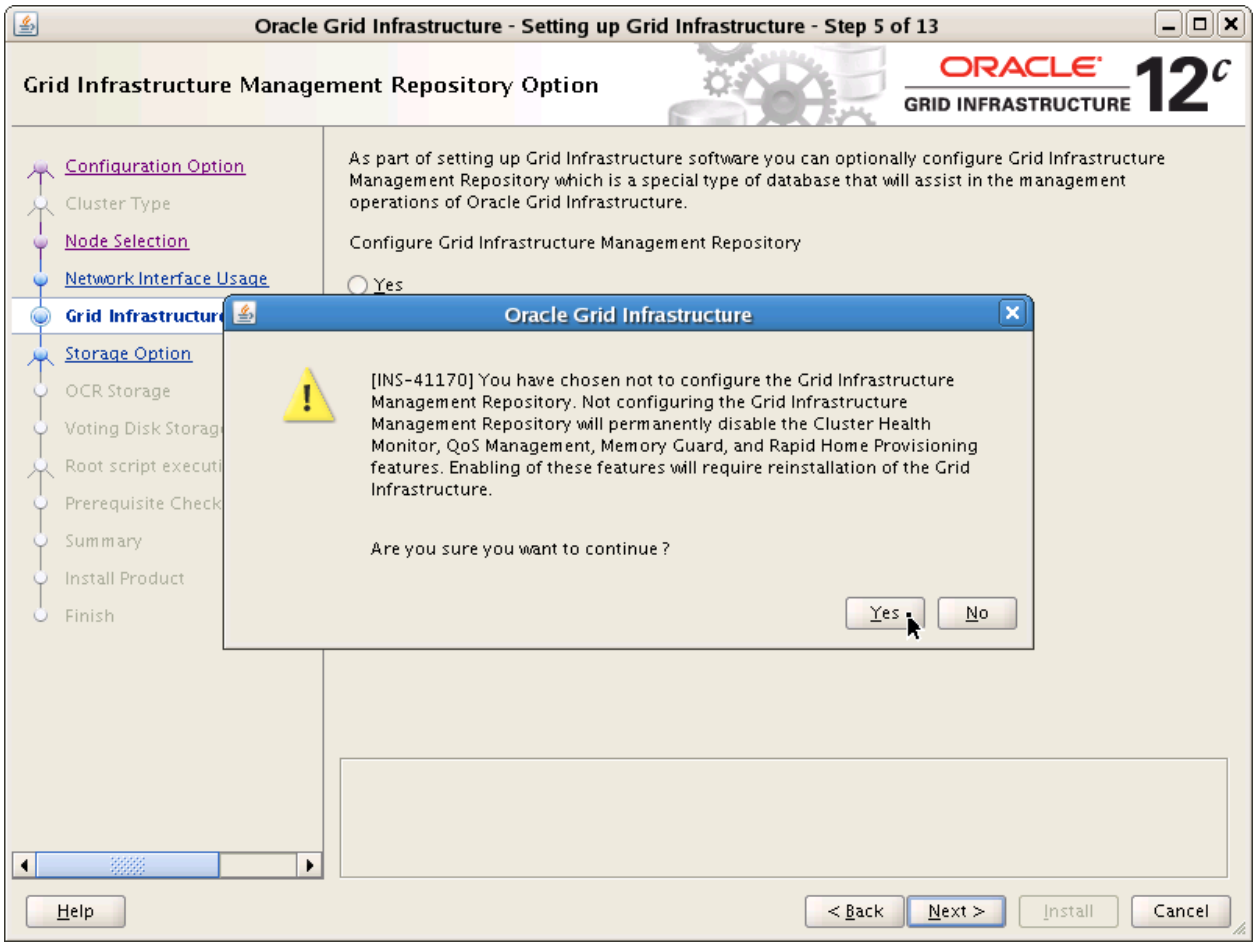
11. On the Grid Infrastructure Management Repository Option page, select the option to not configure the Grid Infrastructure Management Repository and click Next to proceed.

    Note: It is recommended that you do not configure the Grid Infrastructure Management Repository in your practice environment because it is not used in this practice; it takes a significant amount of time to create the repository database and it consumes a significant amount of machine resources (relative to the modest capacity of the practice environment).

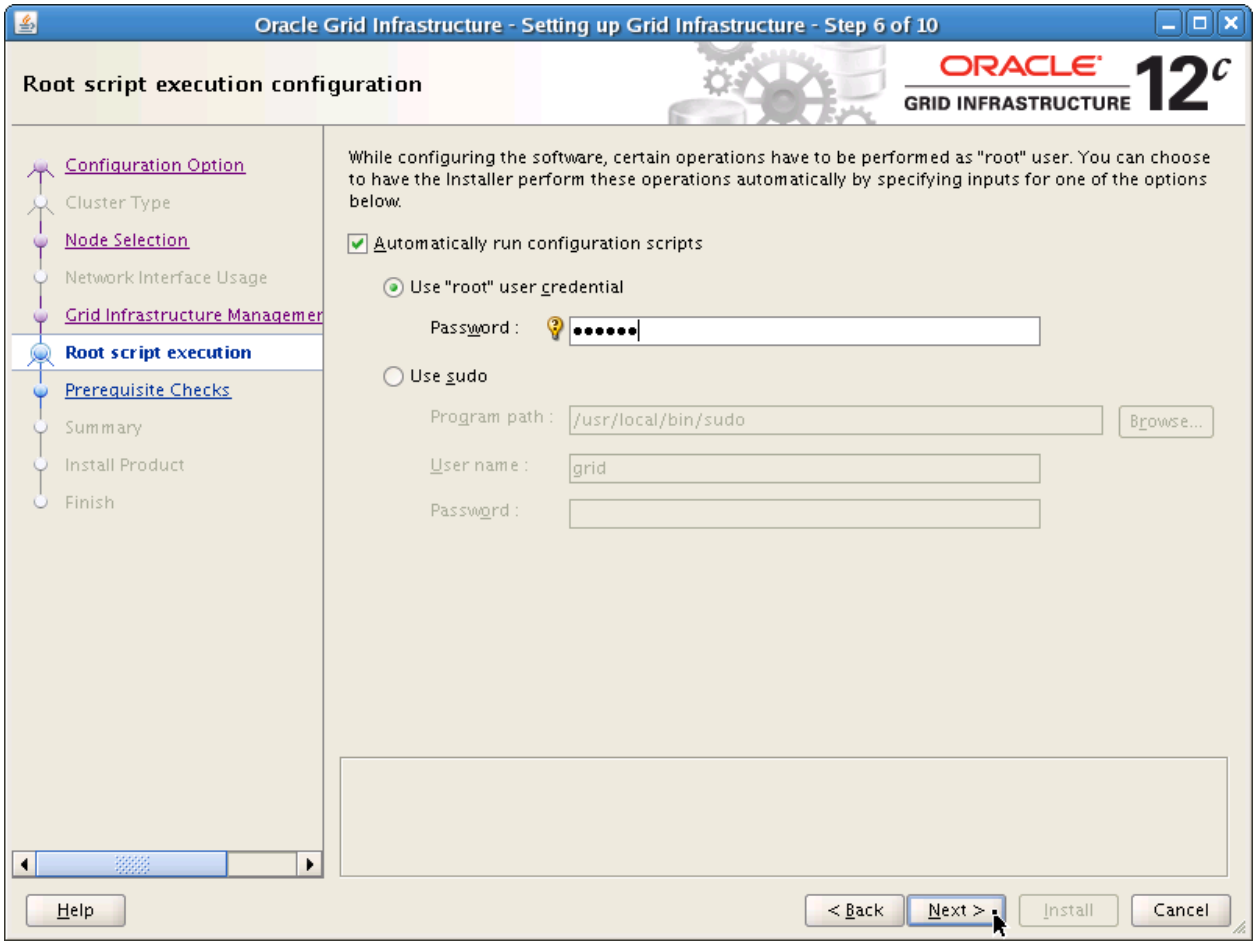


12. Click Yes to acknowledge the additional warning about the Grid Infrastructure Management Repository.

    Note: Because the Grid Infrastructure Management Repository cannot be added without reinstalling Oracle Grid Infrastructure, it is generally recommended that you should configure the repository.

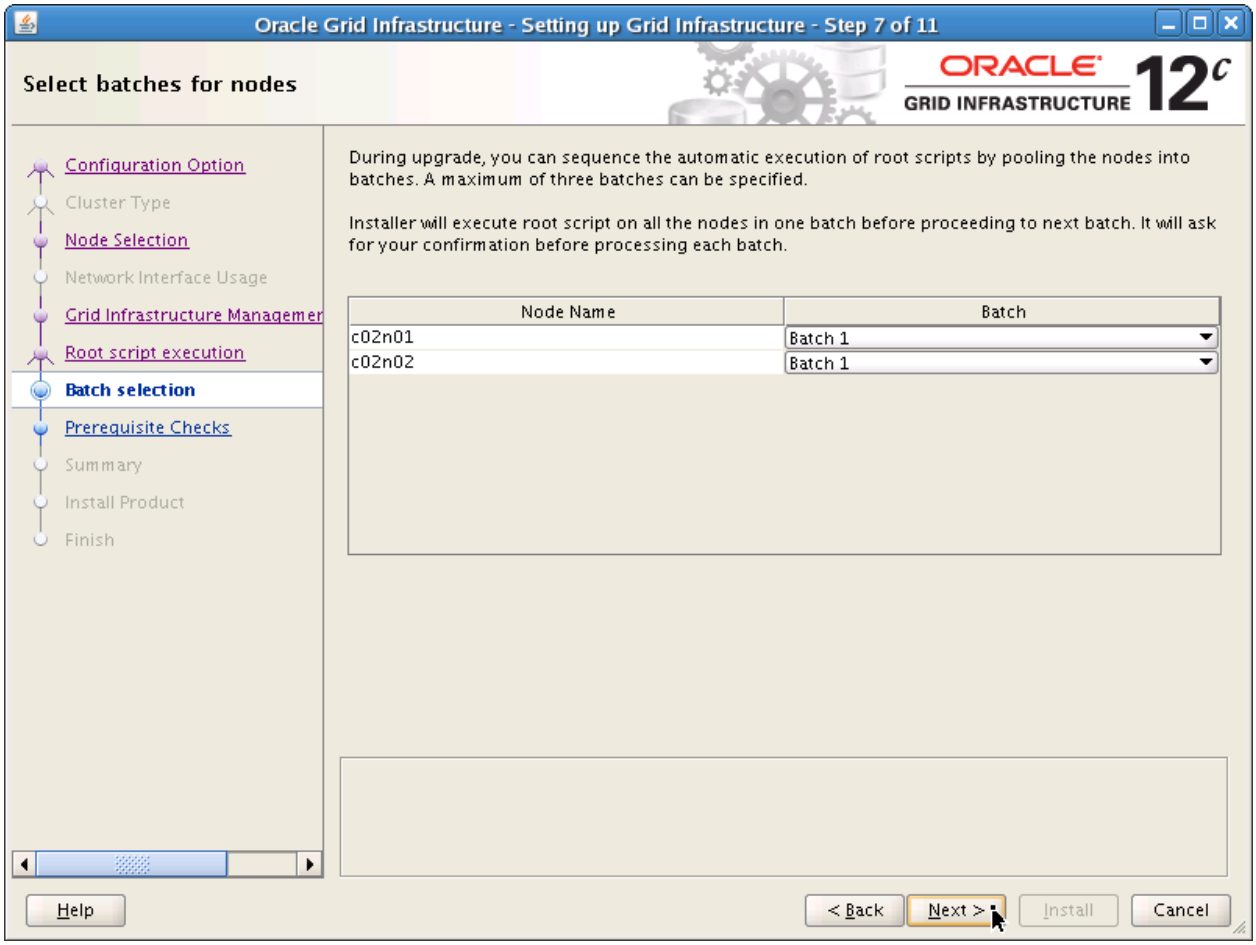


13. On the “Root script execution configuration” page, select the option to automatically run the configuration scripts by using the `root` user credential. Enter `oracle` as the `root` user password and click Next to proceed.

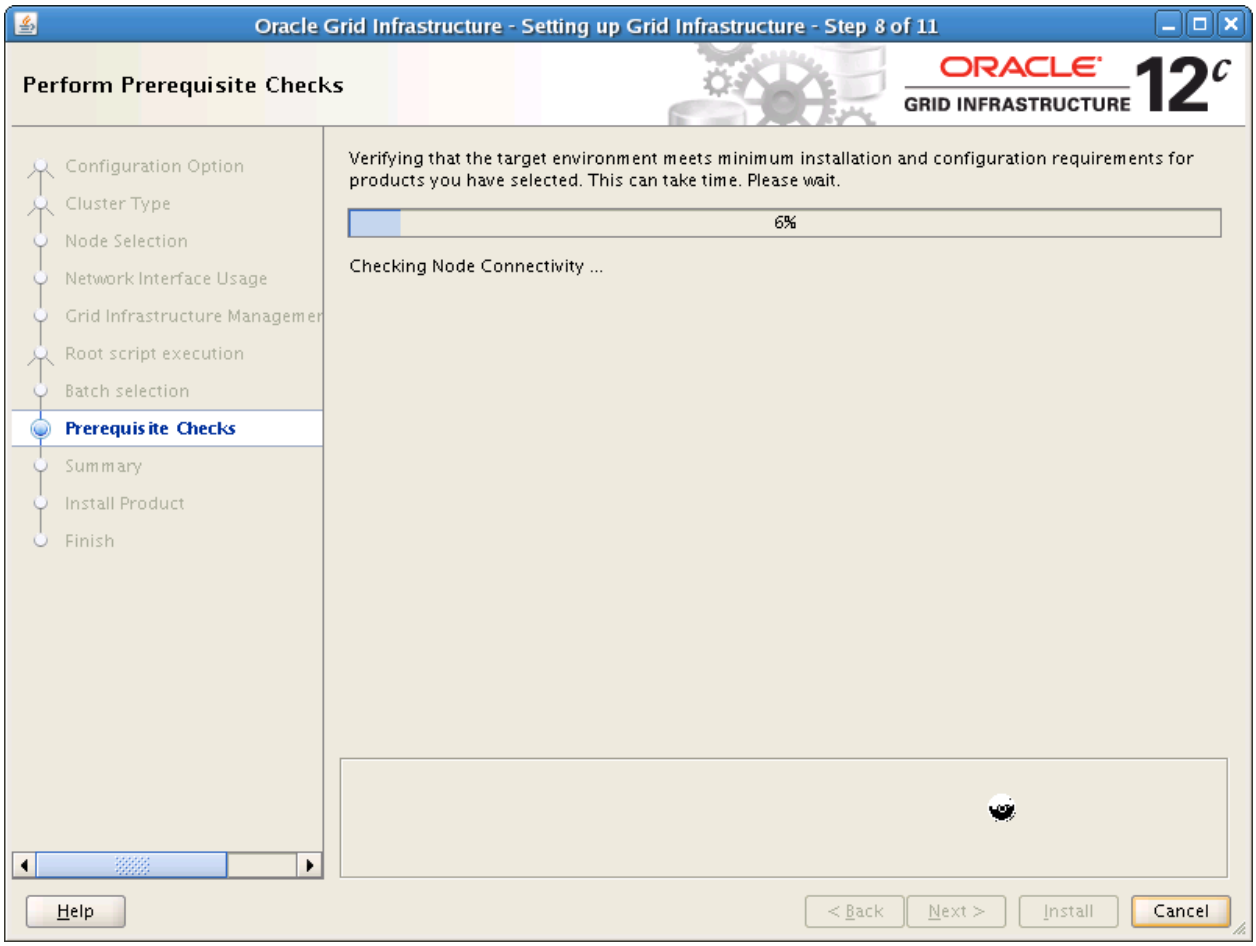


14. On the “Select batches for nodes” page, leave the default configuration and click Next to proceed.

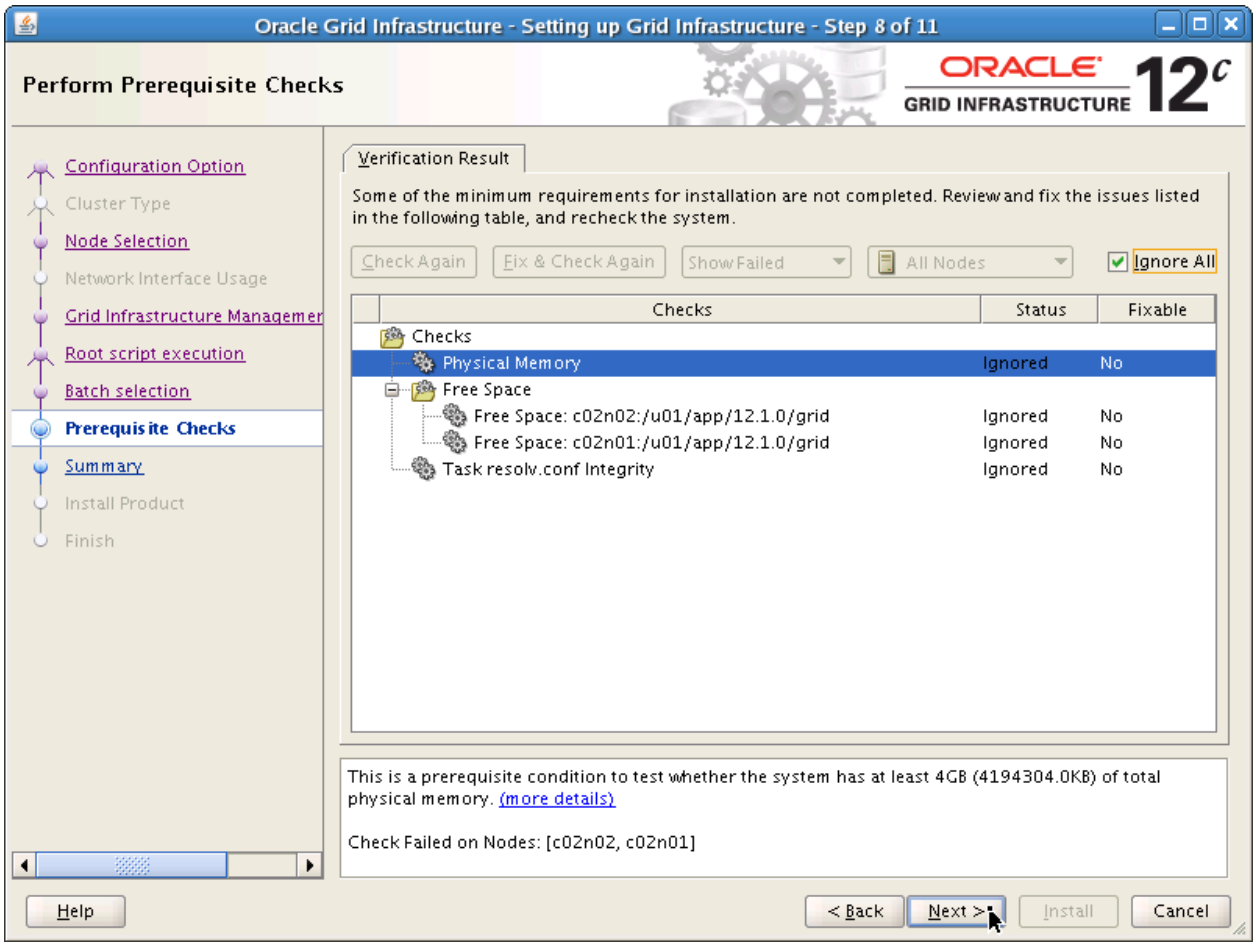
    Note: The upgrade process provides an option to configure groups of nodes in batches. This facility enables administrators to move services around the cluster during the upgrade in order to maintain the availability of vital services throughout the upgrade process. For example, you could define two batches and ensure that your databases continue to run on one batch while the other batch is being upgraded. Then, after the first batch is upgraded, you could move your database on to the upgraded batch of servers while the remaining servers are upgraded.
    You will not use this feature during this practice; however, take note of its existence because you may find it useful for managing more complex upgrades involving larger clusters or clusters that support multiple databases.



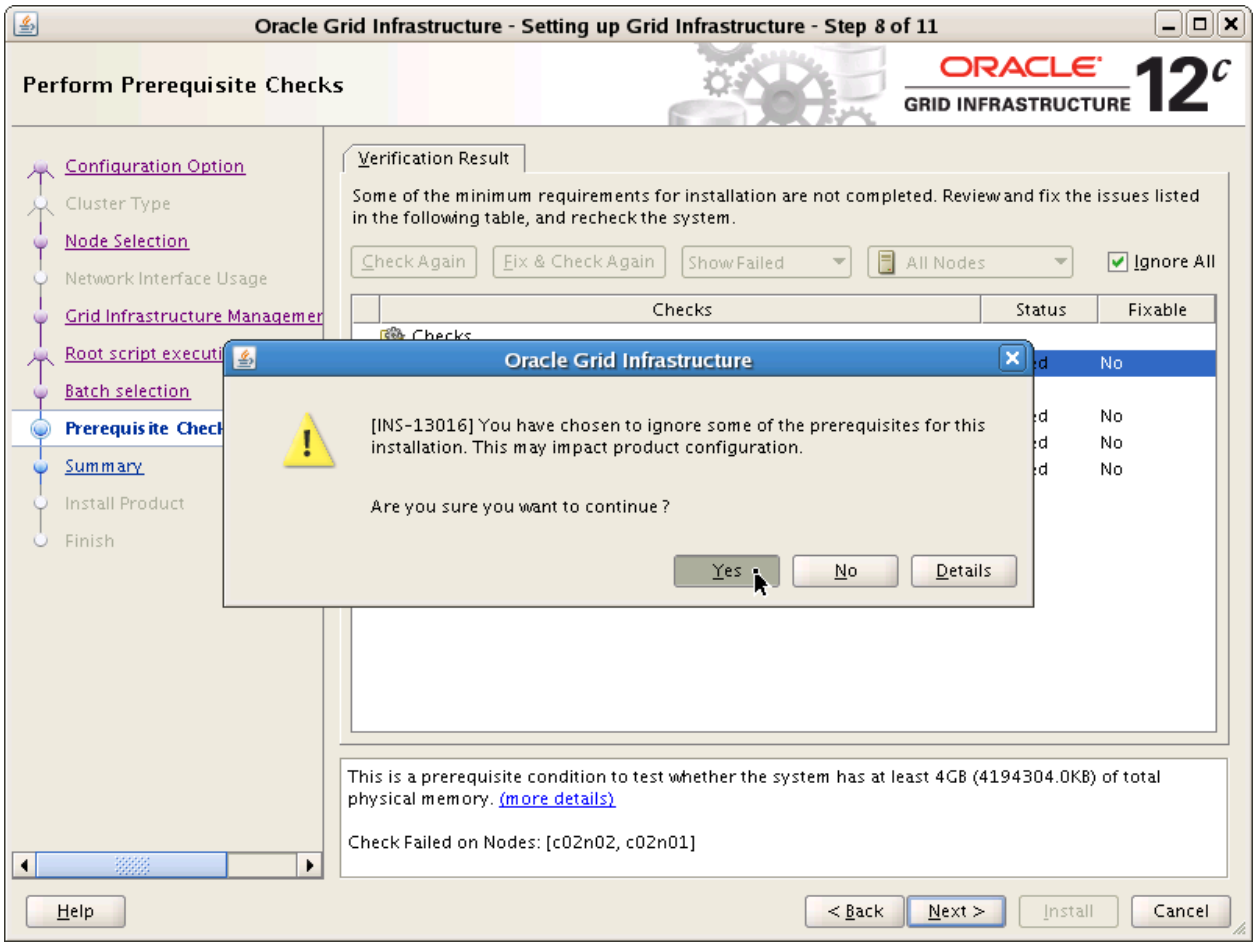
15. Wait while the configuration utility performs a series of checks.



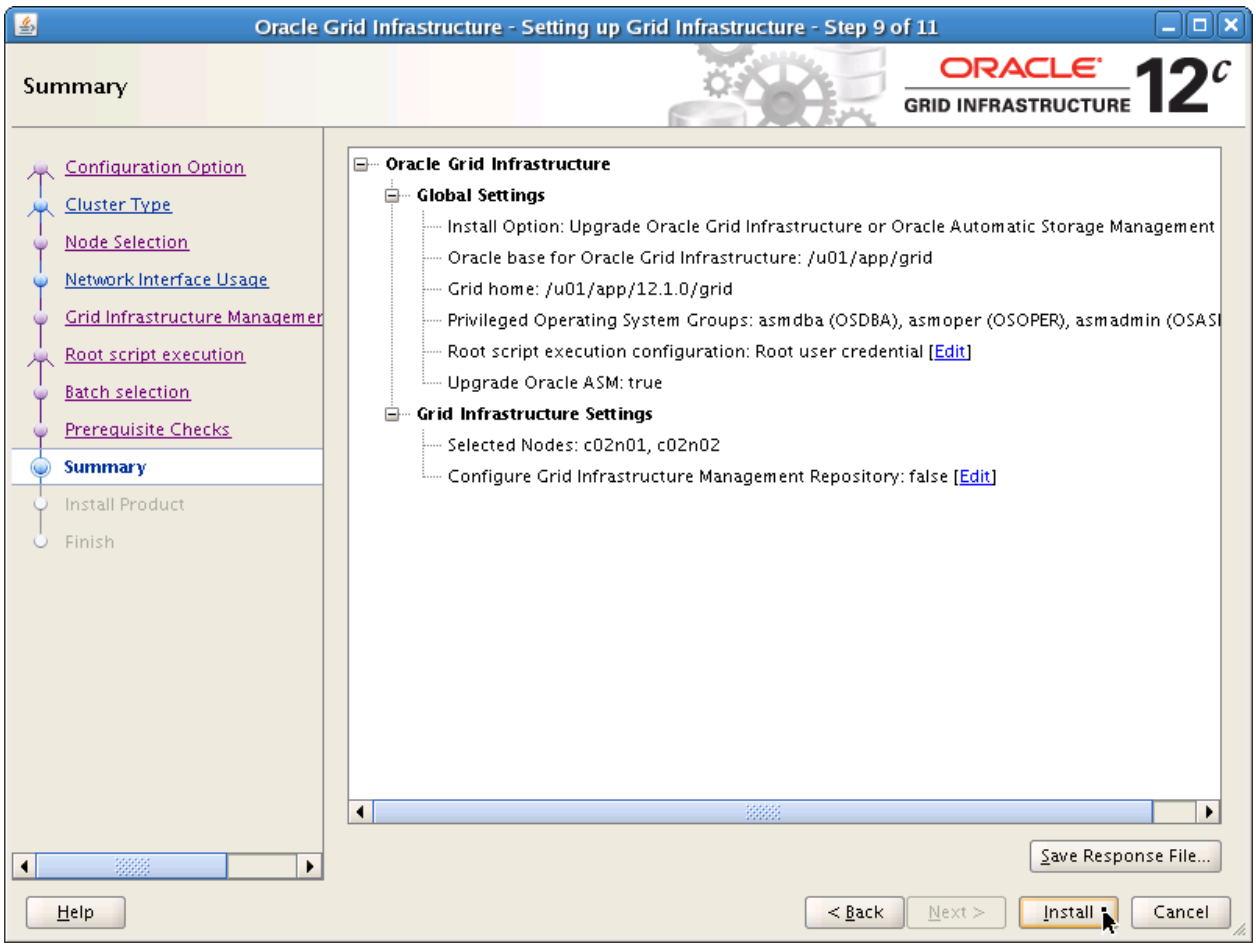
16. Because of constraints associated with your practice environment, some of the prerequisite checks will fail. However, none of these are critical issues and you can ignore the warnings associated with them. To proceed, select the option to ignore all the warnings and click Next.



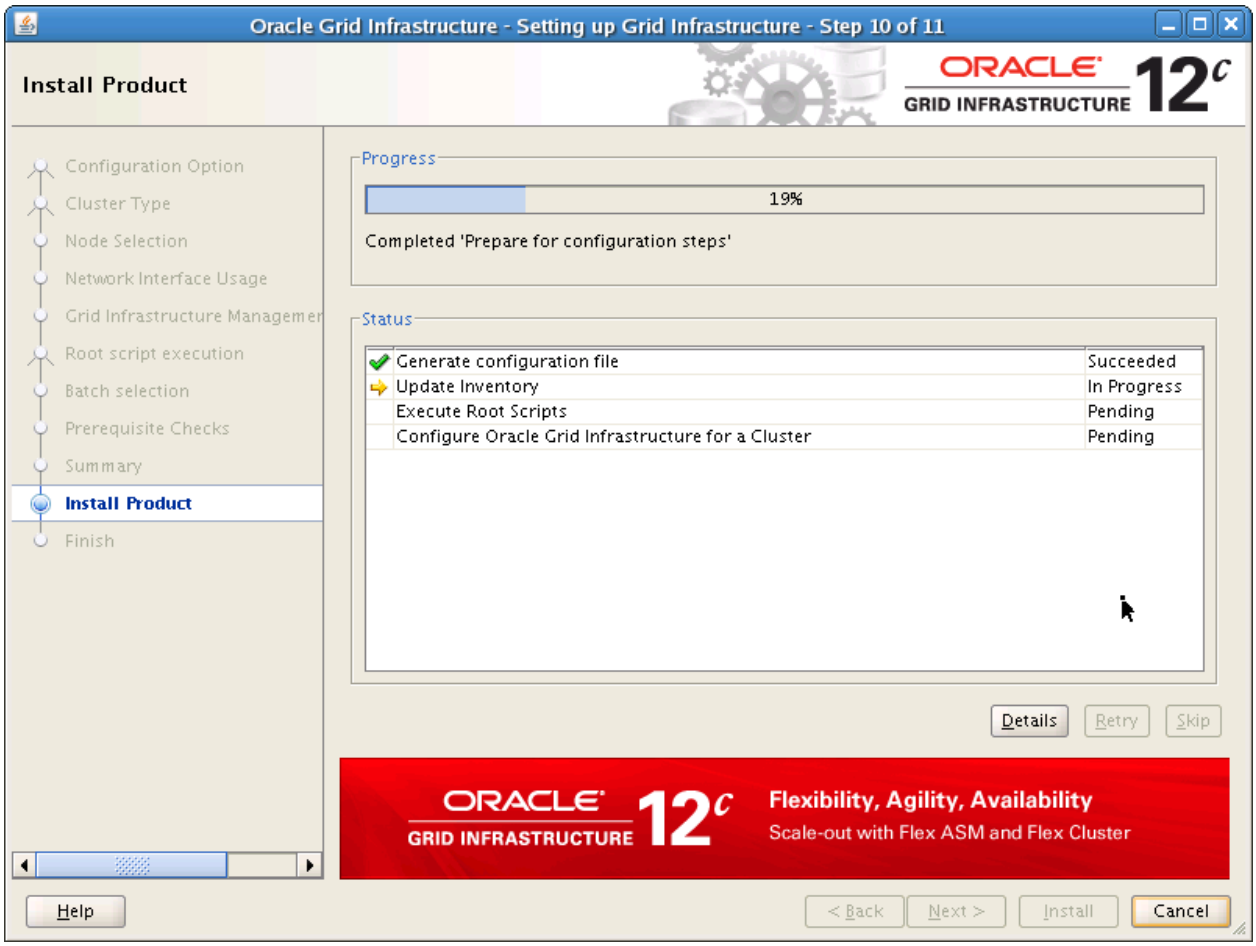
17. Click Yes to acknowledge the additional warning about the installation prerequisites.



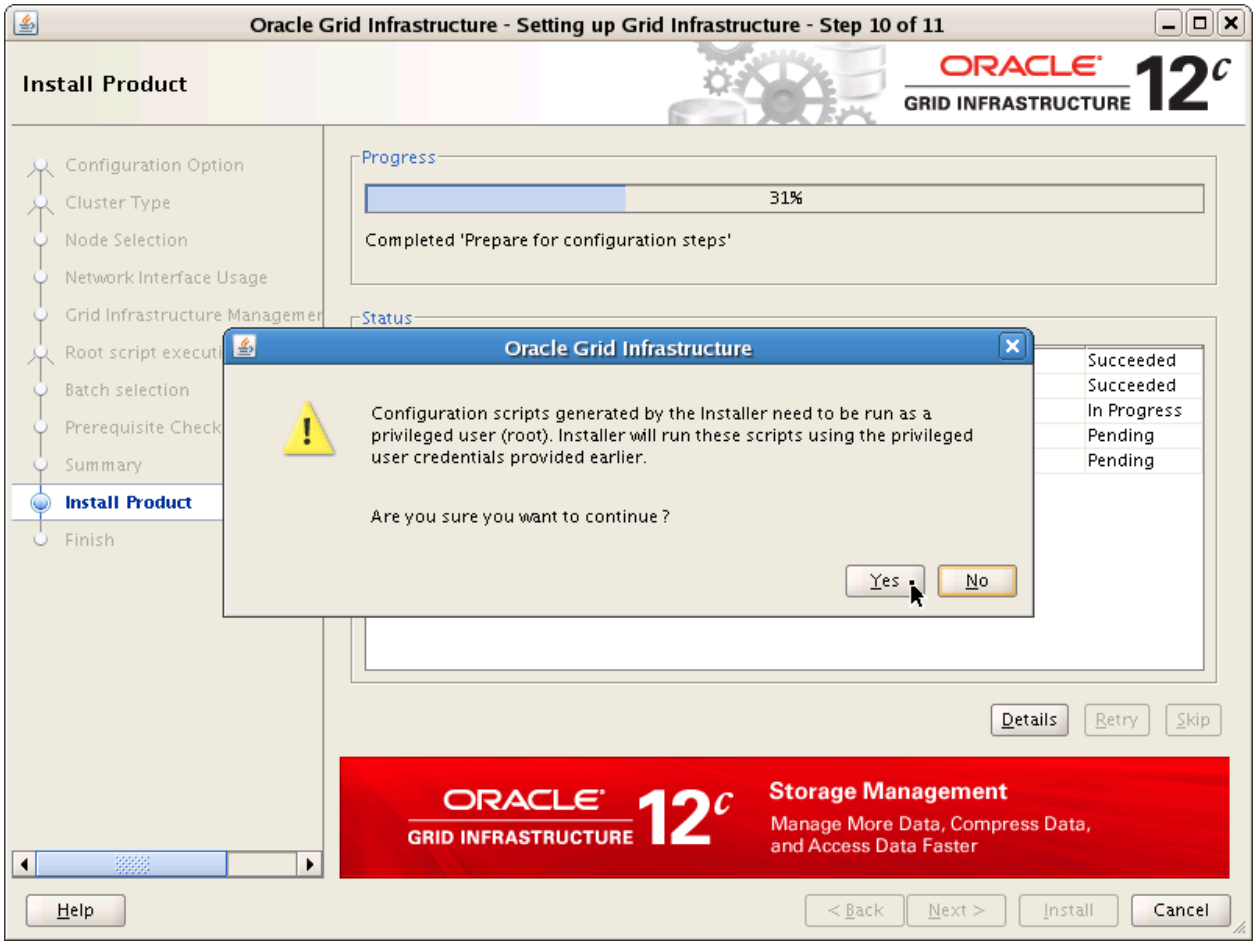
18. On the Summary page, confirm the configuration options that are displayed. When you are satisfied, click Install to upgrade Grid Infrastructure on `cluster02`.



19. You can monitor the configuration process by using the Install Product page.

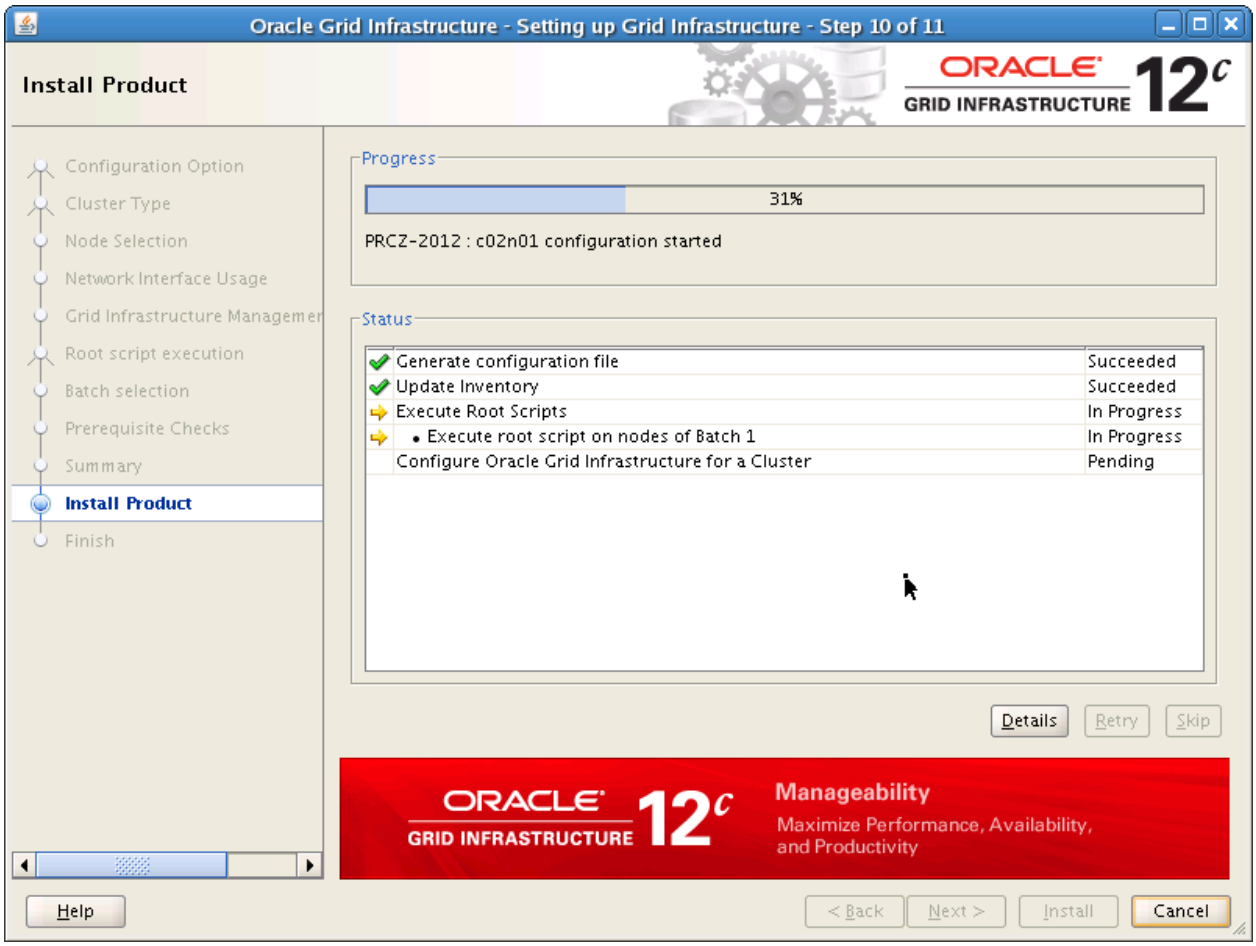


20. Before the `root` configuration scripts are executed, the configuration process will pause and wait until you acknowledge the dialog box. This is your last opportunity to safely exit the configuration process without upgrading the cluster. Click Yes to proceed with the cluster upgrade process.

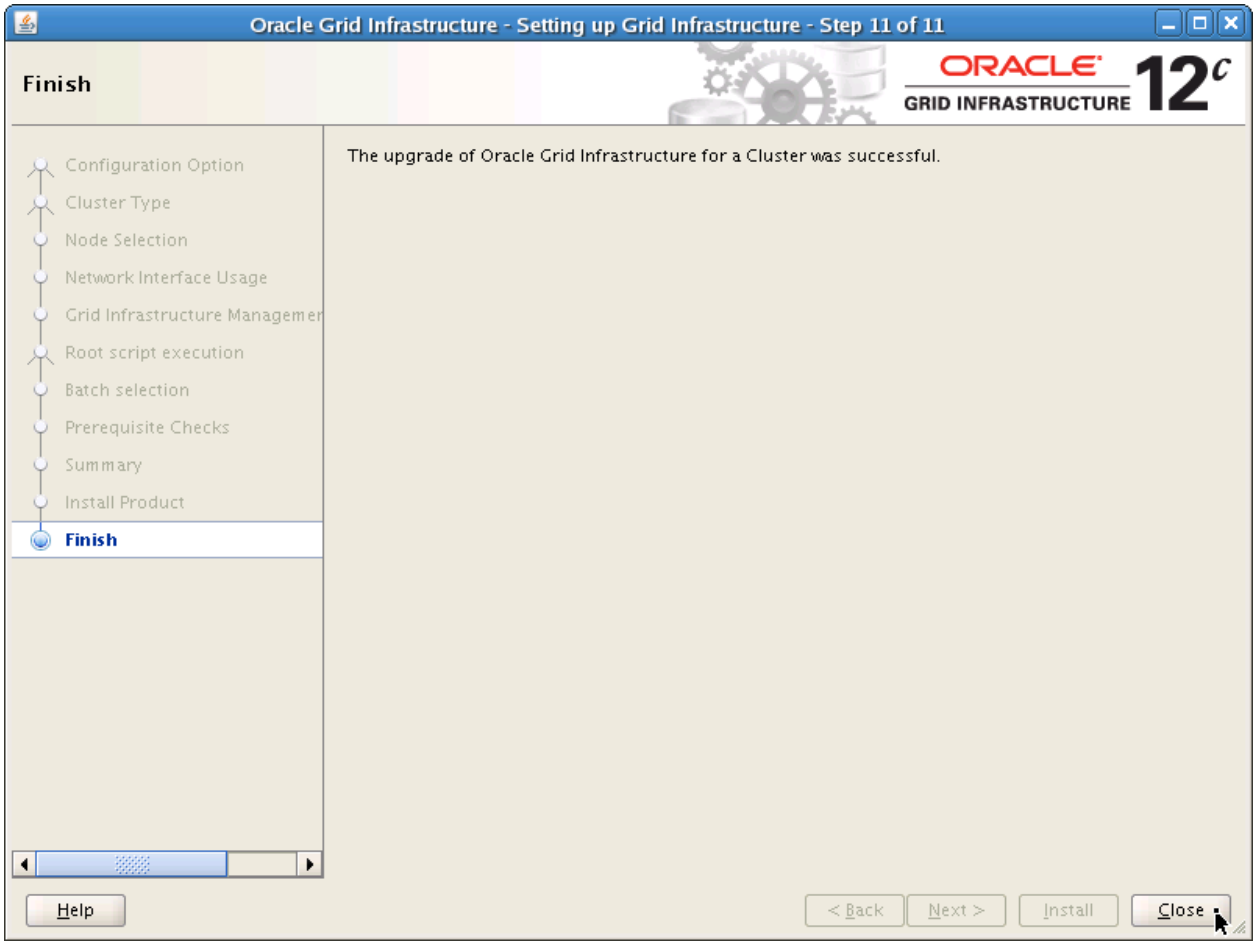


21. Use the Install Product page to monitor the Grid Infrastructure upgrade process. In addition, you can view the log files in `/u01/app/oraInventory/logs` on `c02n01` and also in `/u01/app/12.1.0/grid/cfgtoollogs/crsconfig` on `c02n01` and `c02n02`. You can also monitor each database instance by viewing its alert log file to determine when they are stopped and restarted. In your practice environment, the cluster upgrade process can take approximately 30 minutes to complete.

    Note: During the cluster upgrade process, the redo apply process on the standby database is unavailable for a short period of time when Oracle Clusterware is shut down on the node running the apply instance. When this happens, redo apply starts on the other standby database instance. During this changeover period, a failure of the primary database could result in a longer outage than would normally be expected for a Data Guard environment. Depending on the nature of the failure, some data loss could also be experienced. To mitigate these risks, various approaches can be taken, including the use of an archived redo log repository or a bystander standby database. These options are not explored during this practice; however, be aware of them during your own upgrade planning.



22. When the Grid Infrastructure upgrade process completes, you will see the following page. Click Close to exit the Grid Infrastructure configuration utility.



23. Refresh the environment settings in your `grid@c02n01` terminal session to ensure that you use the upgraded Grid Infrastructure software from now on.

```
[grid@c02n01 ~]$ . oraenv
ORACLE_SID = [+ASM1] ? +ASM1
The Oracle base remains unchanged with value /u01/app/grid
[grid@c02n01 ~]$
```

24. Confirm that Oracle Grid Infrastructure has been upgraded on `cluster02`.

```
[grid@c02n01 ~]$ crsctl query crs activeversion
Oracle Clusterware active version on the cluster is [12.1.0.1.0]
```

25. Return to your `oracle@c01n01` terminal session and launch the Data Guard Broker management utility.

```
[oracle@c01n01 ~]$ dgmgrl sys/oracle_4U
DGMGRL for Linux: Version 11.2.0.4.0 - 64bit Production

Copyright (c) 2000, 2009, Oracle. All rights reserved.

Welcome to DGMGRL, type "help" for information.
Connected.
DGMGRL>
```

26. Show the current configuration. Confirm that the configuration status is `SUCCESS`.

```
DGMGRL> show configuration

Configuration - dg_config

  Protection Mode: MaxPerformance
  Databases:
    c01orcl - Primary database
    c02orcl - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS

DGMGRL>
```

27. Show the status of the physical standby database (`c02orcl`) and ensure that both the transport lag and the apply lag are at or near zero.

```
DGMGRL> show database c02orcl

Database - c02orcl

  Role:            PHYSICAL STANDBY
  Intended State:  APPLY-ON
  Transport Lag:   0 seconds (computed 0 seconds ago)
  Apply Lag:       0 seconds (computed 1 second ago)
  Apply Rate:      38.00 KByte/s
  Real Time Query: ON
  Instance(s):
    c02orcl1
    c02orcl2 (apply instance)
```

```
Database Status:
SUCCESS

DGMGRL>
```

28. Perform a switchover to make `c02orcl` the primary database and `c01orcl` the standby database.

```
DGMGRL> switchover to c02orcl
Performing switchover NOW, please wait...
Operation requires a connection to instance "c02orcl2" on
database "c02orcl"
Connecting to instance "c02orcl2"...
Connected.
New primary database "c02orcl" is opening...
Operation requires startup of instance "c01orcl2" on database
"c01orcl"
Starting instance "c01orcl2"...
ORACLE instance started.
Database mounted.
Database opened.
Switchover succeeded, new primary is "c02orcl"
DGMGRL>
```

29. Confirm that the switchover has succeeded. If the configuration status reports anything other than `SUCCESS`, wait a moment and retry the `show configuration` command.

```
DGMGRL> show configuration

Configuration - dg_config

  Protection Mode: MaxPerformance
  Databases:
    c02orcl - Primary database
    c01orcl - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS

DGMGRL>
```

30. Show the status of the new physical standby database (`c01orcl`) and ensure that both the transport lag and the apply lag are at or near zero. If a lag exceeding 5 seconds is reported, wait a moment and retry the `show database` command.

```
DGMGRL> show database c01orcl

Database - c01orcl

  Role:            PHYSICAL STANDBY
  Intended State:  APPLY-ON
  Transport Lag:   0 seconds (computed 1 second ago)
  Apply Lag:       0 seconds (computed 1 second ago)
  Apply Rate:      25.00 KByte/s
  Real Time Query: ON
  Instance(s):
    c01orcl1
    c01orcl2 (apply instance)

Database Status:
SUCCESS

DGMGRL>
```

31. Exit the Data Guard Broker management utility.

```
DGMGRL> exit
[oracle@c01n01 ~]$
```

32. Leave your other terminal sessions open and establish another terminal session connected to `c01n01` as the `grid` user. Configure the terminal environment as shown in the following:

```
$ ssh -X grid@c01n01
[grid@c01n01 ~]$ . oraenv
ORACLE_SID = [grid] ? +ASM1
The Oracle base has been set to /u01/app/grid
[grid@c01n01 ~]$
```

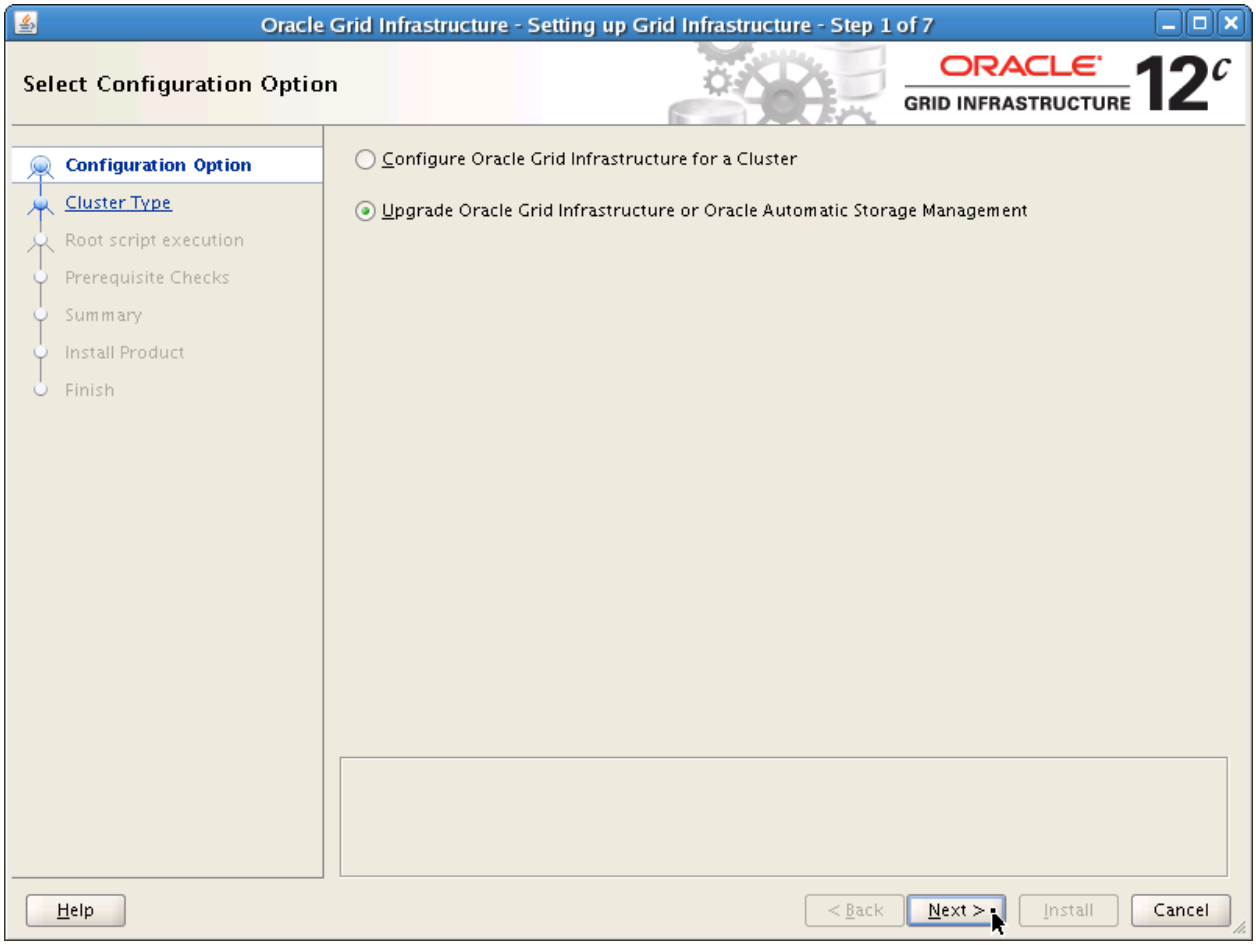
33. Confirm the current active version of Oracle Grid Infrastructure software on `cluster01`.

```
[grid@c01n01 ~]$ crsctl query crs activeversion
Oracle Clusterware active version on the cluster is [11.2.0.4.0]
[grid@c01n01 ~]$
```

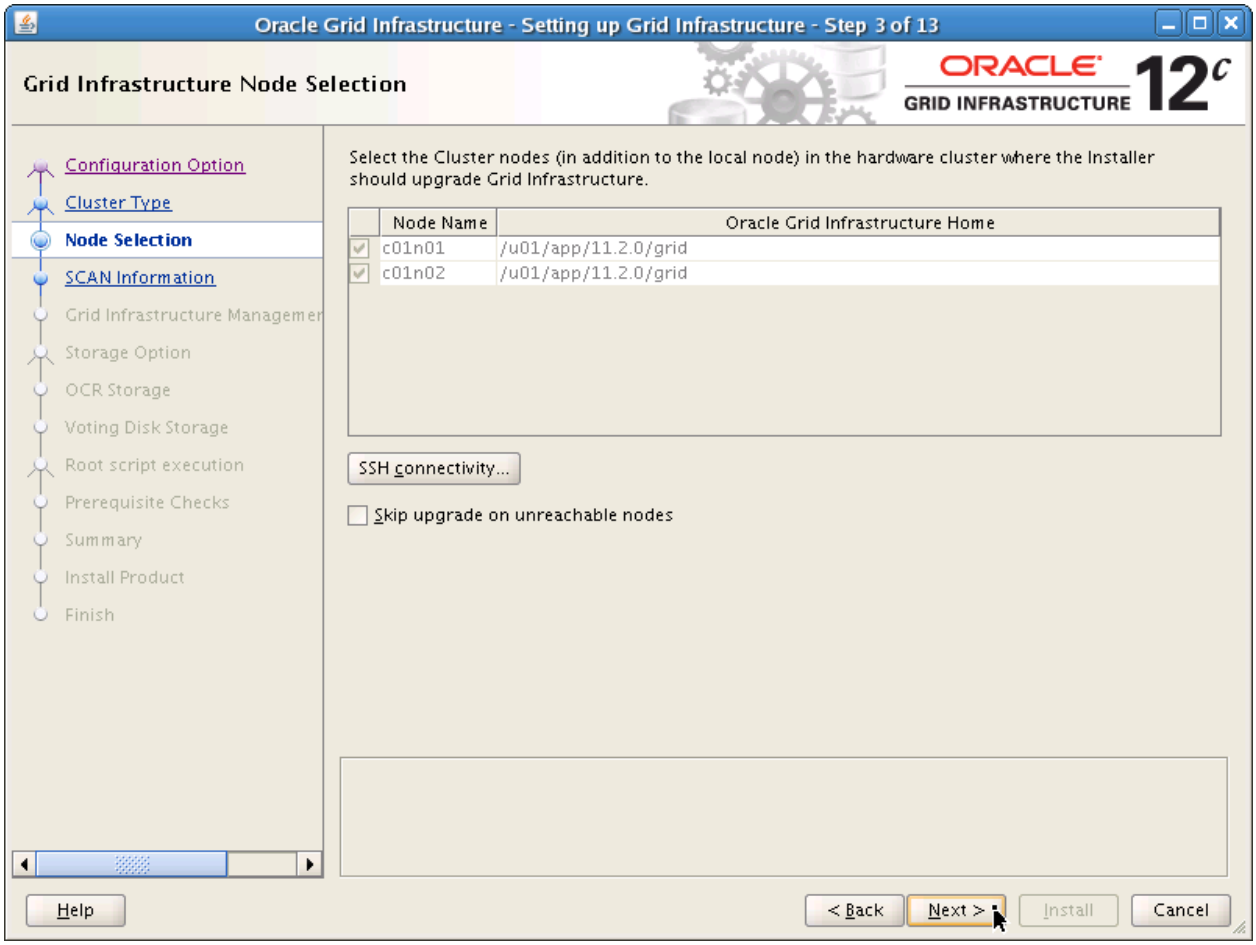
34. Launch the Oracle Grid Infrastructure configuration utility.

```
[grid@c01n01 ~]$ /u01/app/12.1.0/grid/crs/config/config.sh &
```

35. On the Select Configuration Option page, ensure that the option to “Upgrade Oracle Grid Infrastructure or Oracle Automatic Storage Management” is selected and click Next to proceed.

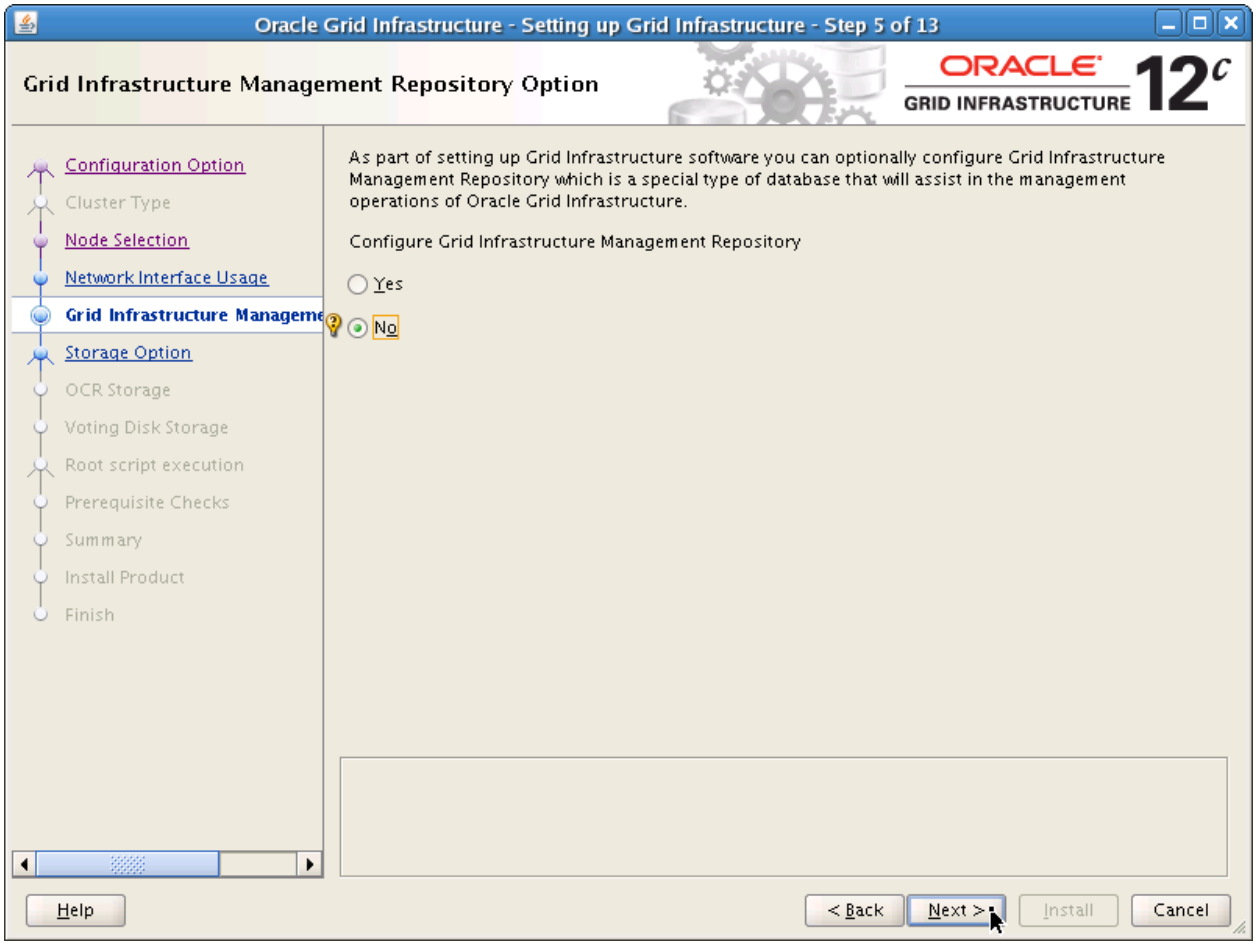


36. On the Grid Infrastructure Node Selection page, ensure that the details on the page match the composition of the cluster and click Next to proceed.



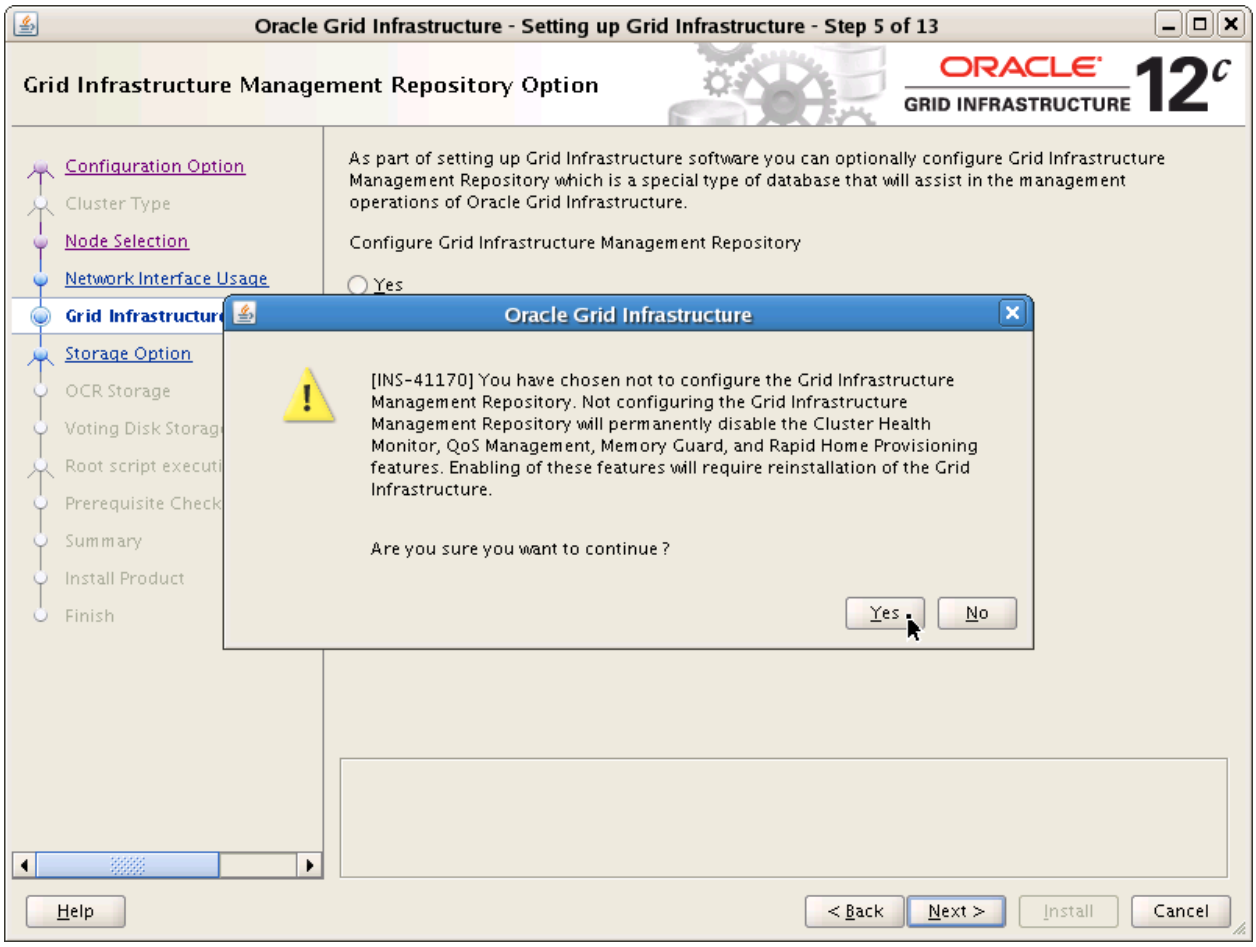
37. On the Grid Infrastructure Management Repository Option page, select the option to not configure the Grid Infrastructure Management Repository and click Next to proceed.

    Note: It is recommended that you do not configure the Grid Infrastructure Management Repository in your practice environment because it is not used in this practice. Moreover, it takes a significant amount of time to create the repository database and it consumes a significant amount of machine resources (relative to the modest capacity of the practice environment).

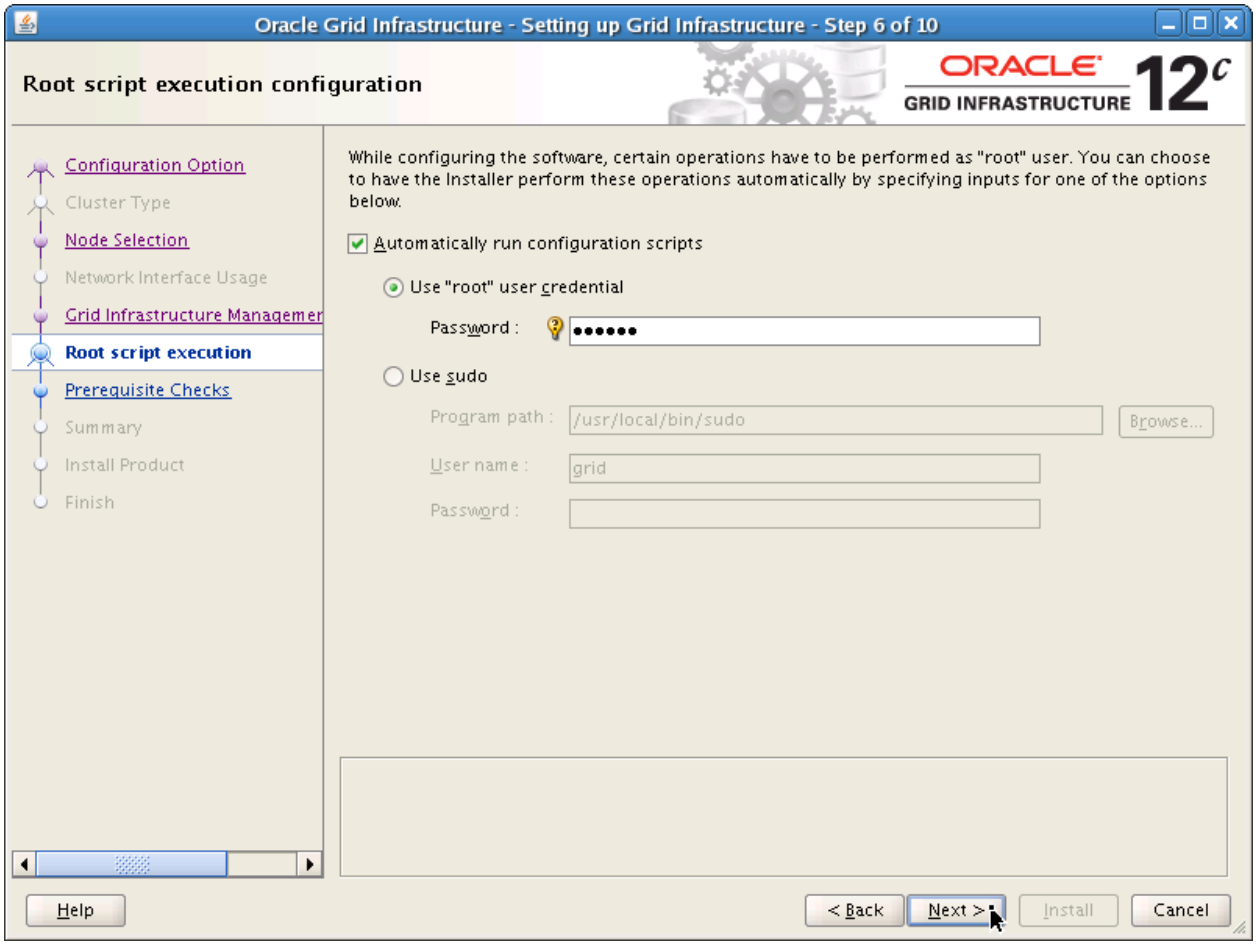


38. Click Yes to acknowledge the additional warning about the Grid Infrastructure Management Repository.

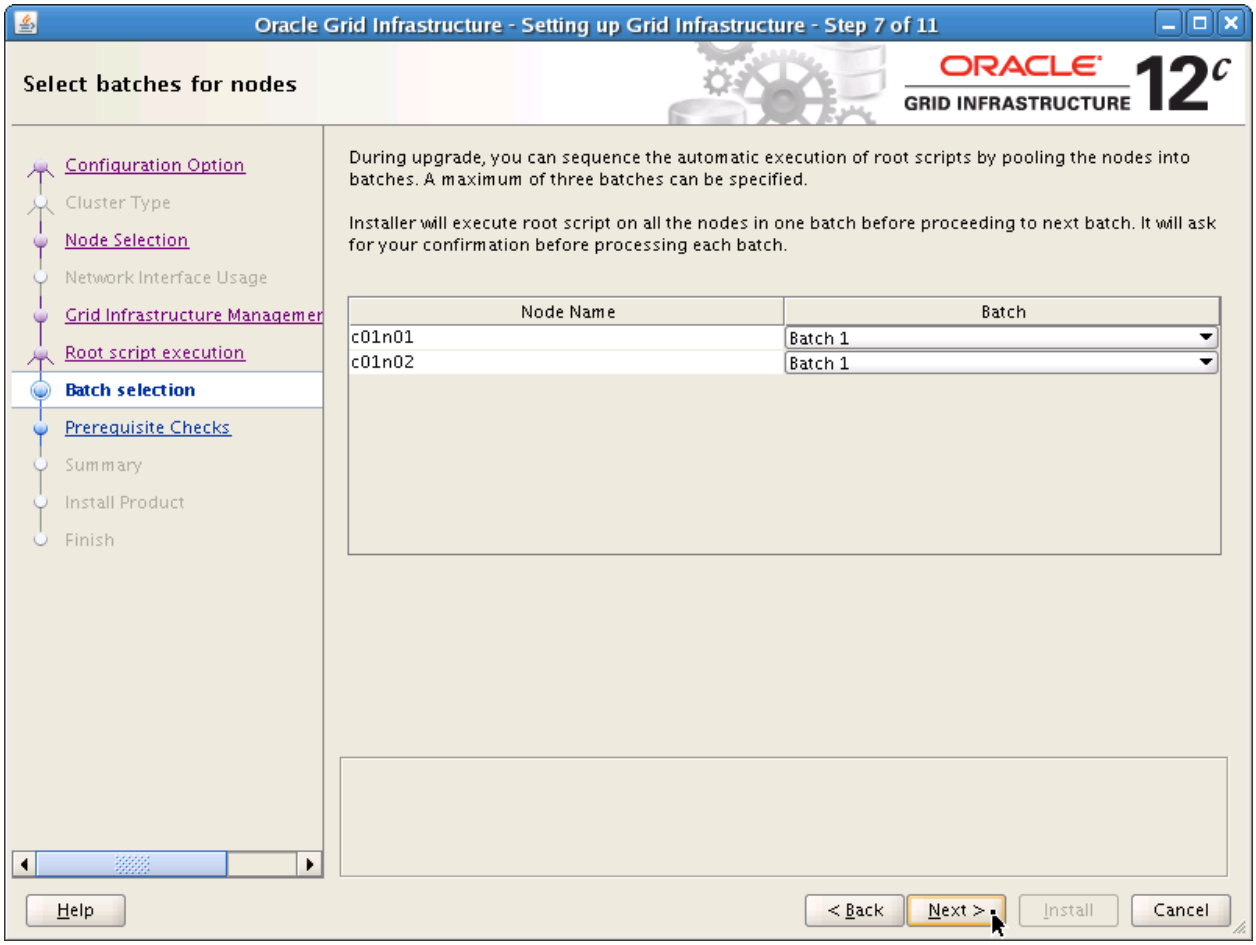
    Note: Because the Grid Infrastructure Management Repository cannot be added without reinstalling Oracle Grid Infrastructure, it is generally recommended that you should configure the repository.



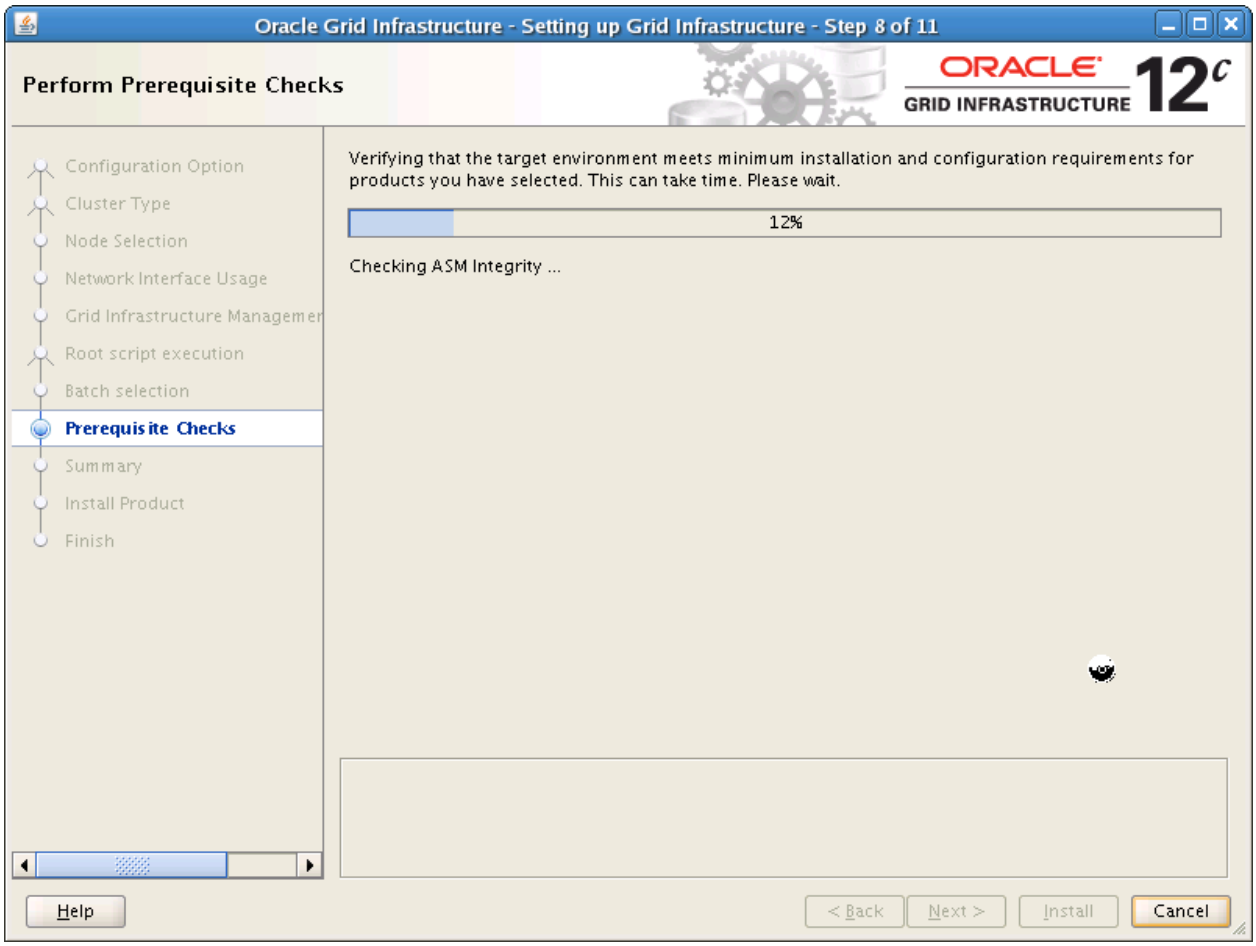
39. On the “Root script execution configuration” page, select the option to automatically run the configuration scripts by using the `root` user credential. Enter `oracle` as the `root` user password and click Next to proceed.



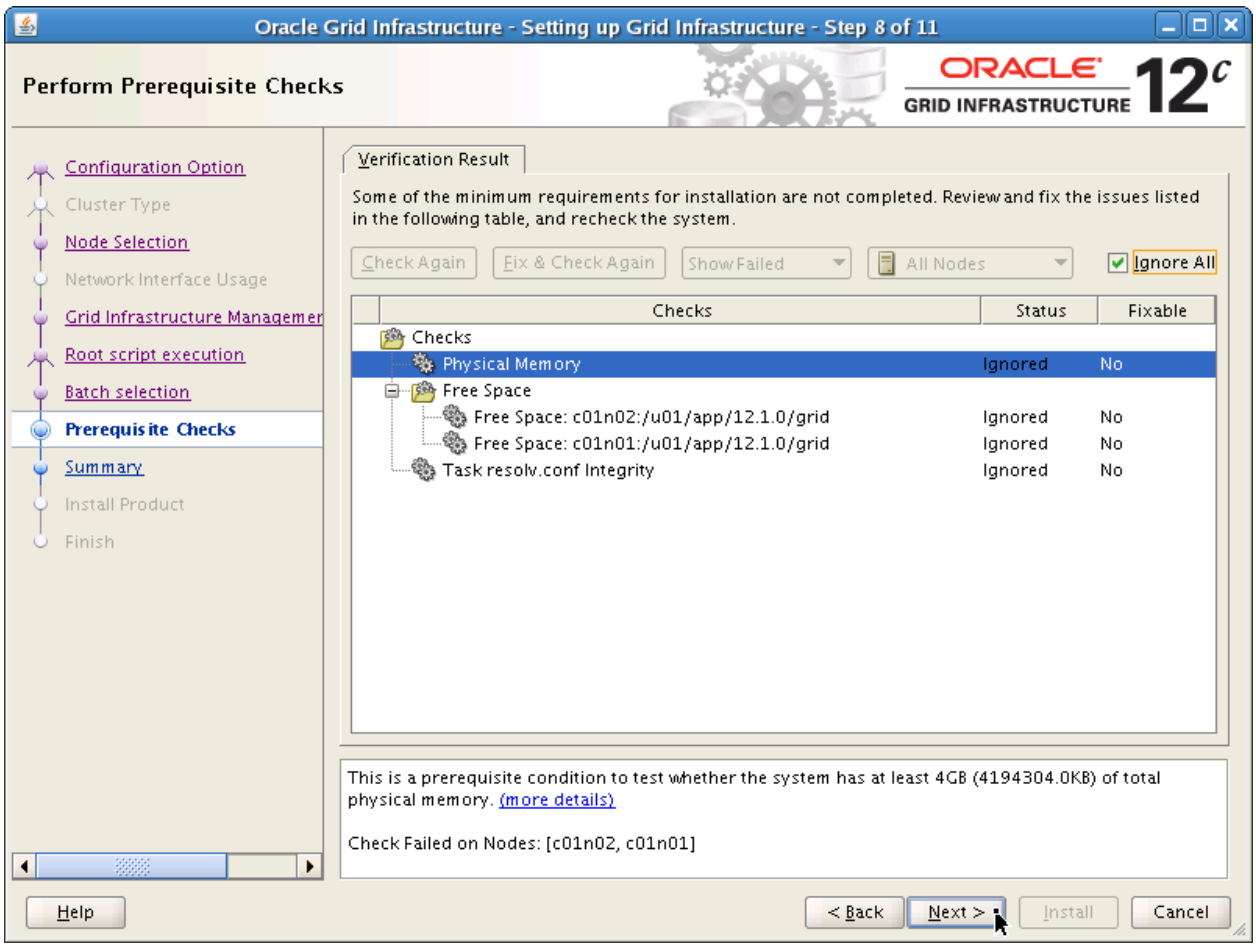
40. On the “Select batches for nodes” page, leave the default configuration and click Next to proceed.



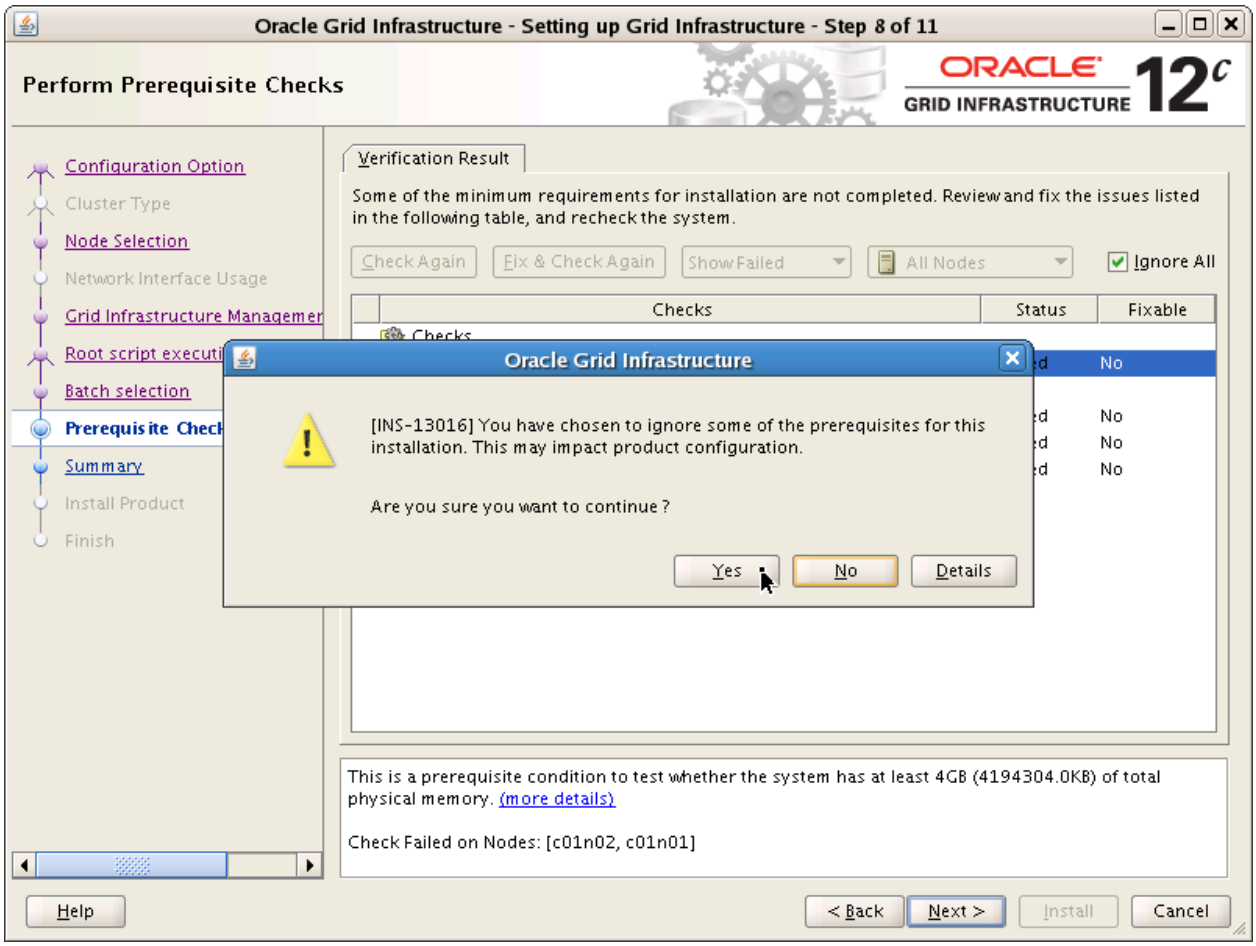
41. Wait while the configuration utility performs a series of checks.



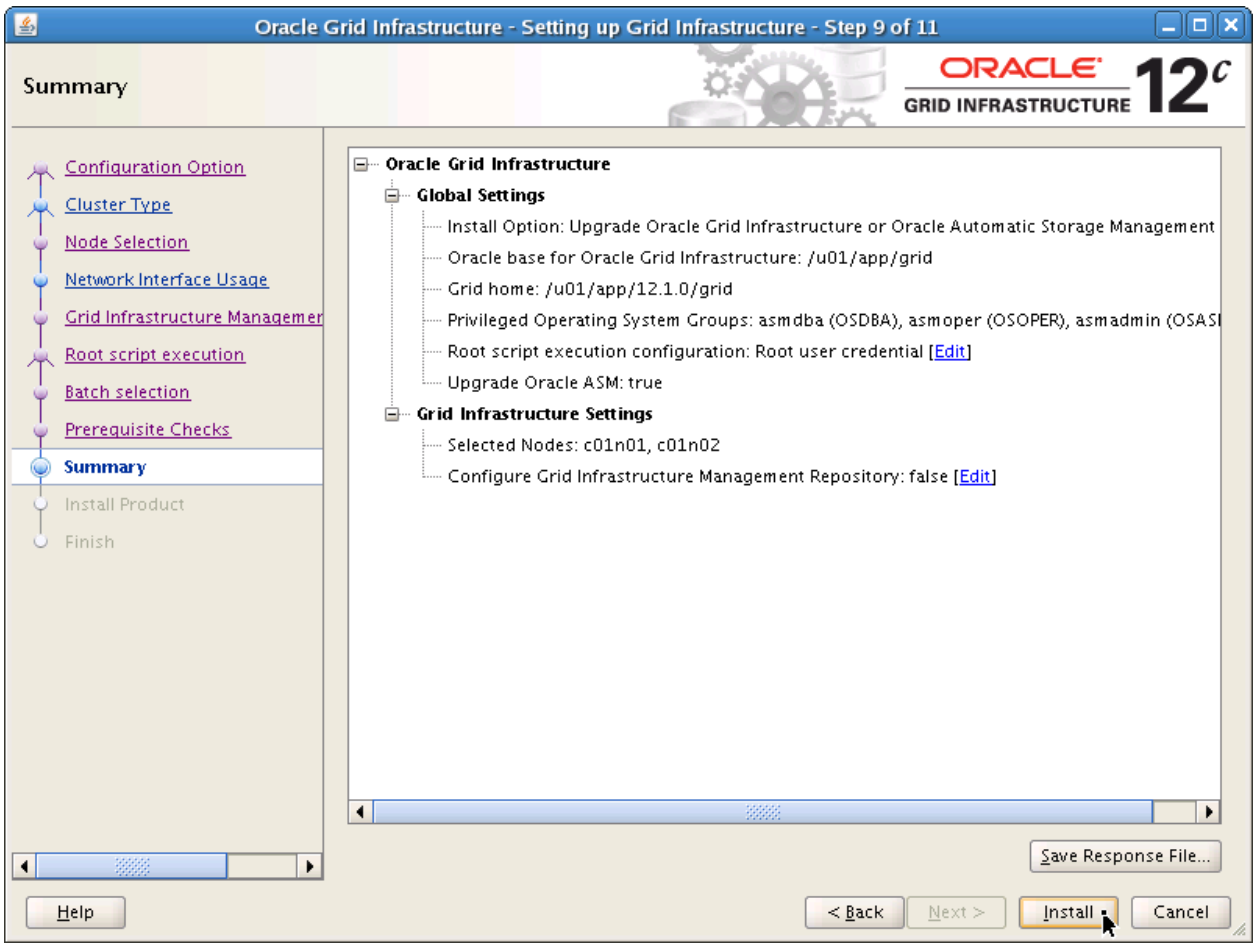
42. When the verification results appear, select the option to ignore all the warnings and click Next to proceed.



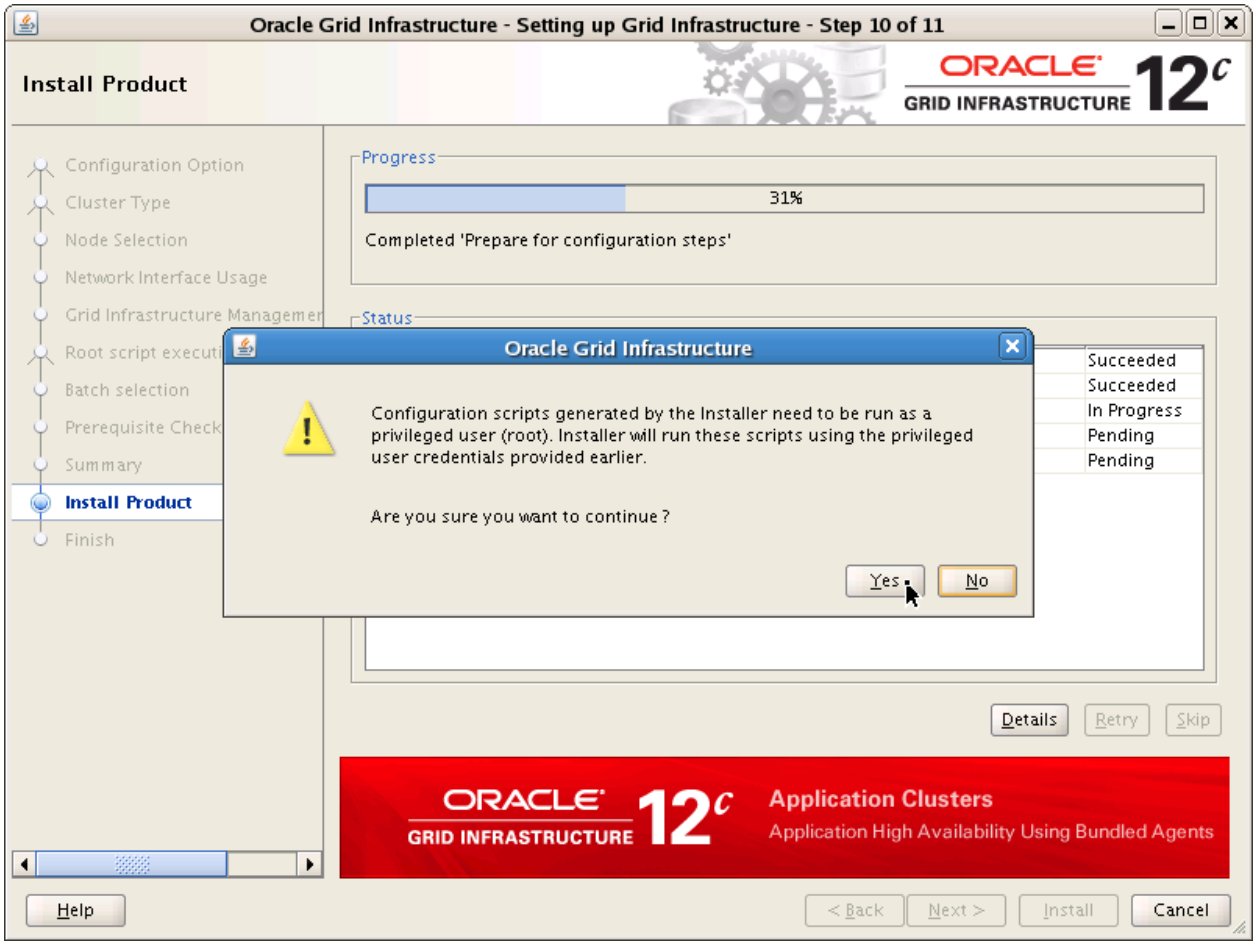
43. Click Yes to acknowledge the additional warning about the installation prerequisites.



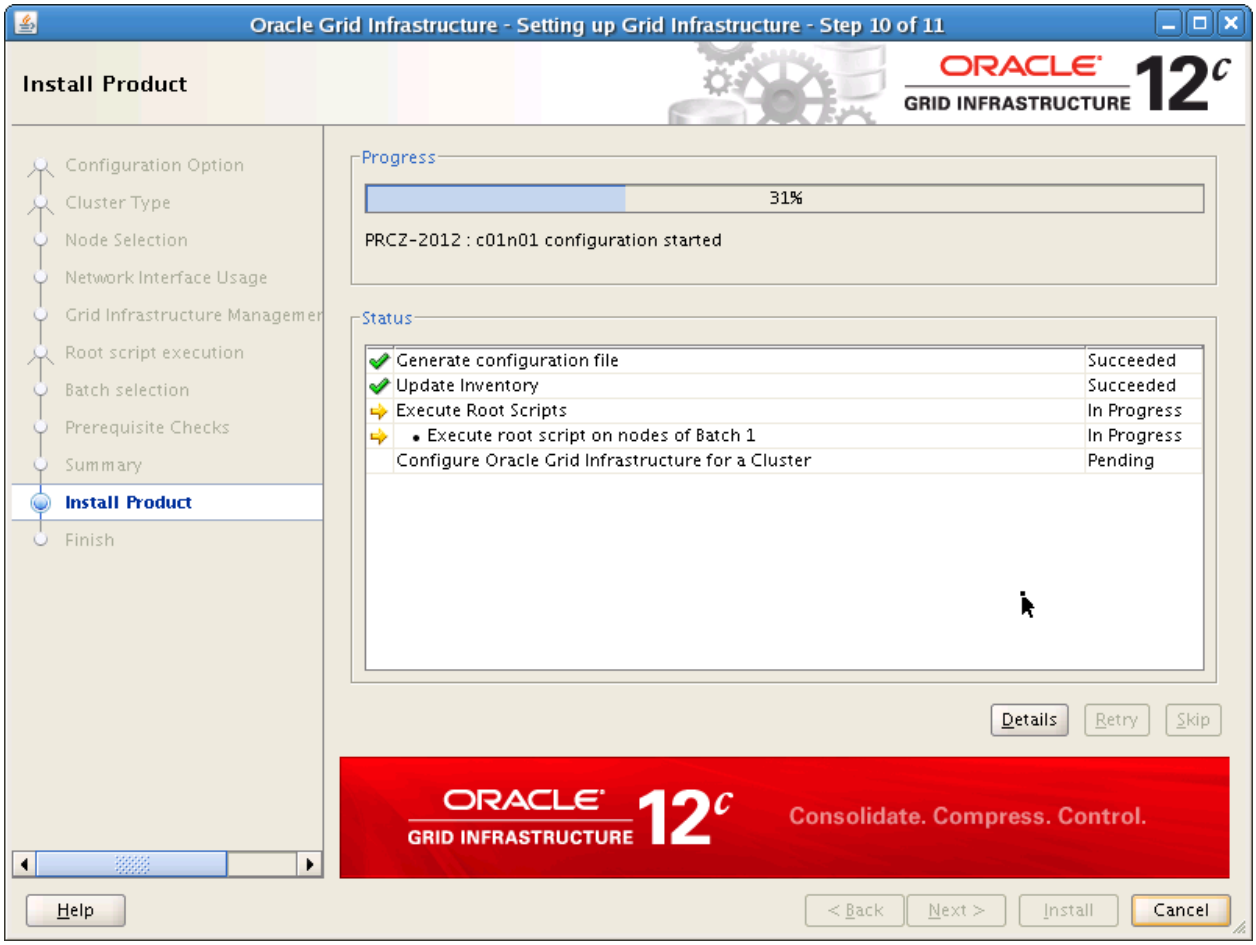
44. On the Summary page, confirm the configuration options that are displayed. When you are satisfied, click Install to upgrade Grid Infrastructure on `cluster01`.



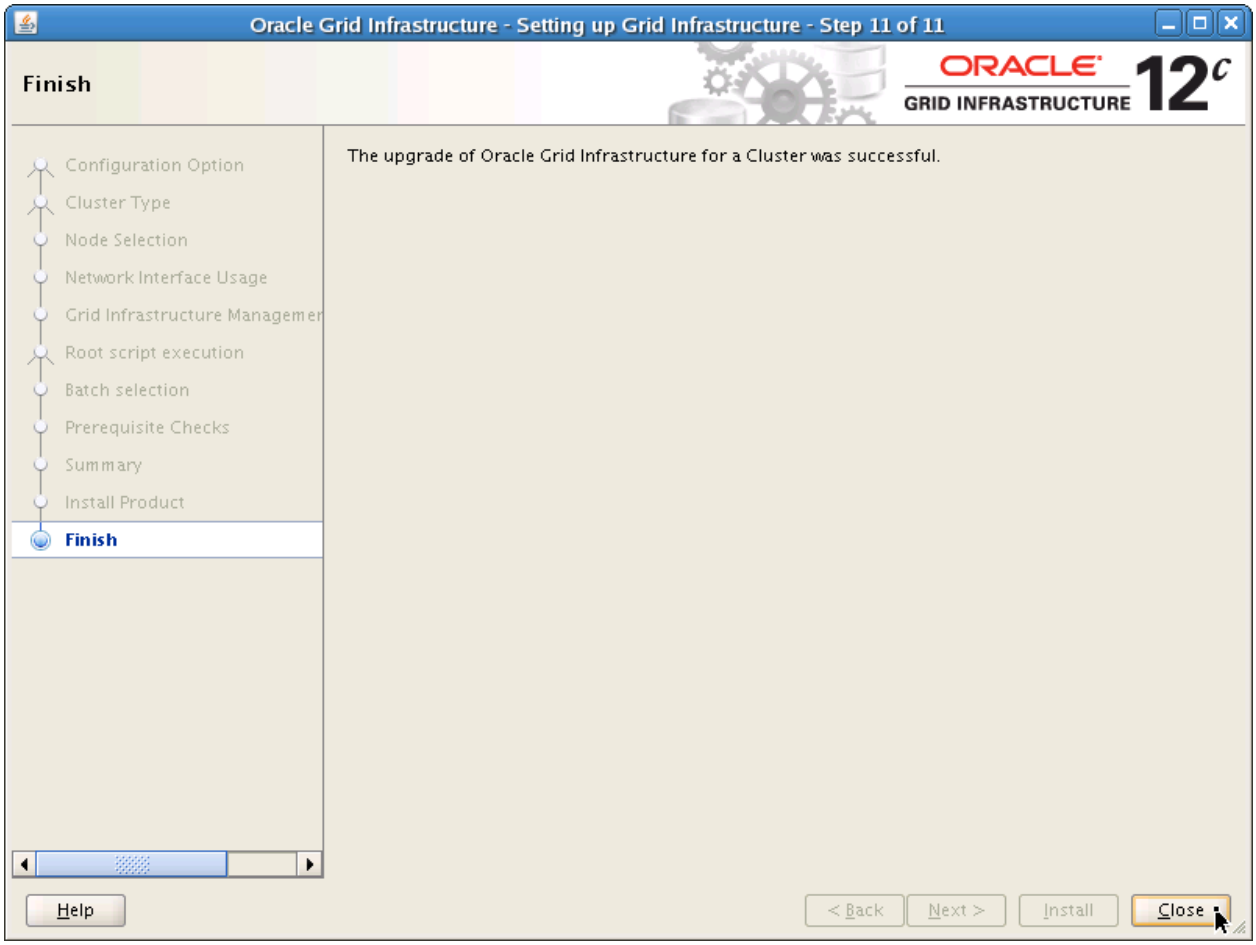
45. When the `root` script confirmation dialog box appears, click Yes to proceed with the cluster upgrade process.



46. Use the Install Product page to monitor the Grid Infrastructure upgrade process. In addition, you can view the log files in `/u01/app/oraInventory/logs` on `c01n01` and also in `/u01/app/12.1.0/grid/cfgtoollogs/crsconfig` on `c01n01` and `c01n02`. You can also monitor each database instance by viewing its alert log file to determine when they are stopped and restarted. Like `cluster02`, the upgrade process for `cluster01` can take approximately 30 minutes to complete.



47. When the Grid Infrastructure upgrade process completes, you see the following page. Click Close to exit the Grid Infrastructure configuration utility.



48. Refresh the environment settings in your `grid@c01n01` terminal session to ensure that you use the upgraded Grid Infrastructure software from now on.

```
[grid@c01n01 ~]$ . oraenv
ORACLE_SID = [+ASM1] ? +ASM1
The Oracle base remains unchanged with value /u01/app/grid
[grid@c01n01 ~]$
```

49. Confirm that Oracle Grid Infrastructure has been upgraded on `cluster01`.

```
[grid@c01n01 ~]$ crsctl query crs activeversion
Oracle Clusterware active version on the cluster is [12.1.0.1.0]
[grid@c01n01 ~]$
```

50. Return to your `oracle@c01n01` terminal session, and launch the Data Guard Broker management utility.

```
[oracle@c01n01 ~]$ dgmgrl sys/oracle_4U
DGMGRL for Linux: Version 11.2.0.4.0 - 64bit Production

Copyright (c) 2000, 2009, Oracle. All rights reserved.

Welcome to DGMGRL, type "help" for information.
Connected.
DGMGRL>
```

51. Show the current configuration. Confirm that the configuration status is `SUCCESS`.

```
DGMGRL> show configuration

Configuration - dg_config

  Protection Mode: MaxPerformance
  Databases:
    c02orcl - Primary database
    c01orcl - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS

DGMGRL>
```

52. Show the status of the physical standby database (`c01orcl`) and ensure that both the transport lag and the apply lag are at or near zero.

```
DGMGRL> show database c01orcl

Database - c01orcl

  Role:            PHYSICAL STANDBY
  Intended State:  APPLY-ON
  Transport Lag:   0 seconds (computed 0 seconds ago)
  Apply Lag:       0 seconds (computed 1 second ago)
  Apply Rate:      14.00 KByte/s
  Real Time Query: ON
  Instance(s):
    c01orcl1 (apply instance)
    c01orcl2
```

```
Database Status:
SUCCESS

DGMGRL>
```

53. Perform a switchover to once again make `c01orcl` the primary database and `c02orcl` the standby database.

```
DGMGRL> switchover to c01orcl
Performing switchover NOW, please wait...
New primary database "c01orcl" is opening...
Operation requires startup of instance "c02orcl2" on database
"c02orcl"
Starting instance "c02orcl2"...
ORACLE instance started.
Database mounted.
Database opened.
Switchover succeeded, new primary is "c01orcl"
DGMGRL>
```

54. Confirm that the switchover has succeeded. If the configuration status reports anything other than `SUCCESS`, wait a moment and retry the `show configuration` command.

```
DGMGRL> show configuration

Configuration - dg_config

  Protection Mode: MaxPerformance
  Databases:
    c01orcl - Primary database
    c02orcl - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS

DGMGRL>
```

55. Show the status of the new physical standby database (`c01orcl`) and ensure that both the transport lag and the apply lag are at or near zero. If a lag exceeding 5 seconds is reported, wait a moment and retry the `show database` command.

```
DGMGRL> show database c02orcl

Database - c02orcl

  Role:            PHYSICAL STANDBY
```

```
  Intended State:  APPLY-ON
  Transport Lag:   0 seconds (computed 0 seconds ago)
  Apply Lag:       0 seconds (computed 1 second ago)
  Apply Rate:      18.00 KByte/s
  Real Time Query: ON
  Instance(s):
    c02orcl1
    c02orcl2 (apply instance)

Database Status:
SUCCESS

DGMGRL>
```

Part 3: Database Preparation for Rolling Upgrade

At this point, you have upgraded both clusters in your Oracle RAC and Data Guard environment. Next, you will prepare the databases on both clusters so that they can participate in the database rolling upgrade process.

Here is an outline of the tasks that you will perform in this section:

1. Disable the Data Guard Broker.
2. Ensure that the recovery area for each database is sized appropriately for the upgrade process.
3. Ensure that the flashback database feature is enabled for each database.
4. Ensure that managed recovery is enabled on the standby database.

56. Disable the Data Guard Broker configuration and exit the Data Guard Broker management utility.

```
DGMGRL> disable configuration
Disabled.
DGMGRL> exit
[oracle@c01n01 ~]$
```

57. Using SQL*Plus as `oracle` on `c01n01`, connect to your primary database (`c01orcl`) and disable the Data Guard Broker.

```
[oracle@c01n01 ~]$ sqlplus / as sysdba

SQL*Plus: Release 11.2.0.4.0 Production ...

SQL> alter system set dg_broker_start=false scope=both;

System altered.

SQL>
```

58. Set the size of the database recovery area so that it is large enough to accommodate the archived redo logs generated during the rolling upgrade process.

    Notes

    - The amount of space required here depends on the composition of your database, including which database options are installed, along with the transaction volume generated during the rolling upgrade. Although the size specified here (`9000M`) is sufficient for this environment, it is likely to be insufficient in many customer situations.
    - Setting the `db_recovery_file_dest_size` parameter sets the limit for amount of recovery area space used by that database. It does not ensure that this amount of space is actually available. You should check that the file system or ASM disk group hosting your recovery area has sufficient space available.

```
SQL> alter system set db_recovery_file_dest_size=9000M
scope=both;

System altered.

SQL>
```

59. Enable the flashback database feature on your primary database and exit SQL*Plus.

```
SQL> alter database flashback on;

Database altered.

SQL> exit
Disconnected from Oracle Database 11g Enterprise Edition ...
[oracle@c01n01 ~]$
```

60. Using SQL*Plus as `oracle` on `c02n01`, connect to your standby database (`c02orcl`) and disable the Data Guard Broker.

```
[oracle@c02n01 ~]$ sqlplus / as sysdba

SQL*Plus: Release 11.2.0.4.0 Production ...

SQL> alter system set dg_broker_start=false scope=both;

System altered.

SQL>
```

61. Set the size of the recovery area in line with the size that you previously set on the primary database.

```
SQL> alter system set db_recovery_file_dest_size=9000M
scope=both;

System altered.

SQL>
```

62. Stop the managed recovery process on the standby database.

```
SQL> alter database recover managed standby database cancel;

Database altered.

SQL>
```

63. Enable the flashback database feature on your standby database.

```
SQL> alter database flashback on;

Database altered.

SQL>
```

64. Restart the managed recovery process on the standby database and exit SQL*Plus.

```
SQL> alter database recover managed standby database using
current logfile disconnect;

Database altered.

SQL> exit
Disconnected from Oracle Database 11g Enterprise Edition ...
[oracle@c02n01 ~]$
```

Part 4: Database Rolling Upgrade

At this point, your environment should be ready to perform a database rolling upgrade. The database rolling upgrade process requires many operations and role changes to be performed in a precise order. To simplify the process and reduce the likelihood of administrator error, the `physru.sh` script is available through My Oracle Support bulletin 949322.1. This script automates much of the process and provides guidance regarding the required manual steps. The script also performs many checks along the way to maximize the probability of success. In your practice environment, the script is preloaded in the `oracle` user's home directory on `c01n01`.

65. Launch `physru.sh` without any parameters and examine the output.

```
[oracle@c01n01 ~]$ ./physru.sh

Usage: physru <username> <primary_tns> <standby_tns>
              <primary_name> <standby_name> <upgrade_version>
```

```
Purpose:
  Perform a rolling upgrade between a primary and physical standby database.

  This script simplifies a physical standby rolling upgrade.  While numerous
  steps have been automated, this script must be called at least three times
  in order to complete a rolling upgrade.  When this script reaches a point
  where user intervention is required, it outputs a message indicating what
  is expected of the user.  Once the user action is complete, this script can
  be called to resume the rolling upgrade.  In the event of an error, a user
  can take corrective action, and simply call this script again to resume the
  rolling upgrade.  In the event one wishes to abandon the rolling upgrade,
and
  revert the configuration back to its pre-upgrade state, this script creates
  guaranteed flashback database restore points on both the primary and standby
  databases, and backs up each databases' associated control file.  The names
  of the restore points and backup control files are output to the console and
  logfile when they are initially created.

  When this script is called, it assumes all databases to be either mounted or
  open.  It requires flashback database to be enabled on both the primary and
  standby instances.  RAC configurations are permitted but there is limited
  automation provided by the script.  At specific points it may become
  necessary to manually shutdown/startup instances and change init.ora
  parameter values.  When appropriate, the script will output when these
  requirements are expected of the user.  RAC configurations are also required
  to define static tns services since this script expects a given tns service
  name to contact the same instance on successive calls.

Arguments:
  <username>        = dba username
  <primary_tns>     = tns service name to primary
  <standby_tns>     = tns service name to physical standby
  <primary_name>    = db_unique_name of primary
  <standby_name>    = db_unique_name of standby
  <upgrade_version> = target rdbms version

Example:
  physru sys hq_tnspri hq_tnsstb hq_primary hq_standby 11.2.0.2.0

  NOTE: This script performs role transitions, and it is not necessary to
        adjust the tns and db name arguments to their respective database
roles
        on successive calls.  That is, the arguments must remain the same from
        first-invocation to completion.

ERROR: 0 of the 6 required parameters have been specified

[oracle@c01n01 ~]$
```

66. Launch `physru.sh` as shown below in order to commence the database rolling upgrade process. Enter `oracle_4U` when you are prompted for the `sysdba` password.

Notes

- Pay careful attention to the output from `physru.sh`. If your output differs significantly, check with your instructor before proceeding to the next step. To help you better understand what is happening, additional notes are included throughout the example output. In addition, particularly interesting output will be highlighted along the way.
- While `physru.sh` is executing, you may find it interesting to monitor the alert log for your primary (`c01orcl`) and standby (`c02orcl`) databases because the contents of these logs will provide additional insight about the operations being performed on both databases.
- In this practice, Oracle Net connection strings using easy connect naming are used to connect to the primary and standby databases. These connection strings leverage the static service registration that is already configured to support Data Guard Broker. Alternatively, administrators can use Oracle Net service names. However, note that such Oracle Net service names must always resolve to the same database instance in RAC configurations, which means that no Oracle Net connection load balancing mechanisms can be used.
- During the rolling upgrade process, the standby database is periodically unable to accept updates from the primary. While the standby database is unavailable, a failure of the primary database could result in a longer outage than would normally be expected for a Data Guard environment. Depending on the nature of the failure, some data loss could also be experienced. To mitigate these risks, various approaches can be taken, including the use of an archived redo log repository or a bystander standby database. These options are not explored during this practice; however, be aware of them during your own upgrade planning.

```
[oracle@c01n01 ~]$ ./physru.sh sys c01n01:1521/c01orcl_DGMGRL.example.com
c02n01:1521/c02orcl_DGMGRL.example.com c01orcl c02orcl 12.1.0.1.0
Please enter the sysdba password: <oracle_4U>

### Initialize script to either start over or resume execution
Aug 25 06:16:02 2014 [0-1] Identifying rdbms software version
Aug 25 06:16:03 2014 [0-1] database c01orcl is at version 11.2.0.4.0
Aug 25 06:16:03 2014 [0-1] database c02orcl is at version 11.2.0.4.0
Aug 25 06:16:03 2014 [0-1] verifying flashback database is enabled at c01orcl
and c02orcl
Aug 25 06:16:04 2014 [0-1] verifying available flashback restore points
Aug 25 06:16:04 2014 [0-1] verifying DG Broker is disabled
Aug 25 06:16:04 2014 [0-1] looking up prior execution history
Aug 25 06:16:04 2014 [0-1] purging script execution state from database
c01orcl
Aug 25 06:16:04 2014 [0-1] purging script execution state from database
c02orcl
Aug 25 06:16:05 2014 [0-1] starting new execution of script
```

Note: Every time you run `physru.sh`, it will first perform an initialization sequence to ensure that the main elements of your environment are in the expected state. If any issues are detected, the script will produce an error message and exit. After you have corrected the problem, you can safely restart the script.

```
### Stage 1: Backup user environment in case rolling upgrade is aborted
Aug 25 06:16:05 2014 [1-1] stopping media recovery on c02orcl
Aug 25 06:16:07 2014 [1-1] creating restore point PRU_0000_0001 on database
c02orcl
Aug 25 06:16:08 2014 [1-1] backing up current control file on c02orcl
Aug 25 06:16:13 2014 [1-1] created backup control file
/u01/app/oracle/product/11.2.0/dbhome_1/dbs/PRU_0001_c02orcl_f.f
Aug 25 06:16:13 2014 [1-1] creating restore point PRU_0000_0001 on database
c01orcl
Aug 25 06:16:15 2014 [1-1] backing up current control file on c01orcl
Aug 25 06:16:19 2014 [1-1] created backup control file
/u01/app/oracle/product/11.2.0/dbhome_1/dbs/PRU_0001_c01orcl_f.f

NOTE: Restore point PRU_0000_0001 and backup control file PRU_0001_c02orcl_f.f
      can be used to restore c02orcl back to its original state as a
      physical standby, in case the rolling upgrade operation needs to be
      aborted prior to the first switchover done in Stage 4.
```

Note: Stage 1 uses the flashback database feature to create restore points in both the primary and standby databases. These can be used to return your environment to the preupgrade state, if required, any time before the first switchover that occurs later in stage 4. Note that flashing back to these restore points will not only undo any upgrade operations, but will also undo valid user transactions which will continue to flow into the primary database throughout the upgrade process.

```
### Stage 2: Create transient logical standby from existing physical standby
Aug 25 06:16:20 2014 [2-1] verifying RAC is disabled at c02orcl

WARN: c02orcl is a RAC database.  Before this script can continue, you
      must manually reduce the RAC to a single instance, disable the RAC, and
      restart instance c02orcl1 in mounted mode.  This can be accomplished
      with the following steps:

        1) Shutdown all instances other than instance c02orcl1.
           eg: srvctl stop instance -d c02orcl -i c02orcl2 -o abort

        2) On instance c02orcl1, set the cluster_database parameter to FALSE.
           eg: SQL> alter system set cluster_database=false scope=spfile;

        3) Shutdown instance c02orcl1.
           eg: SQL> shutdown abort;

        4) Startup instance c02orcl1 in mounted mode.
           eg: SQL> startup mount;
```

```
      Once these steps have been performed, enter 'y' to continue the script.
      If desired, you may enter 'n' to exit the script to perform the required
      steps, and recall the script to resume from this point.

Are you ready to continue? (y/n):
```

Note: Stage 2 converts the existing physical standby database into a transient logical standby database. The transient logical standby database is a key element of the rolling upgrade process because it allows user transactions to flow into it from the primary database at the same time as it is being upgraded. However, before the physical-to-logical conversion can take place, Oracle RAC must be disabled on the existing physical standby database.

67. Using your `oracle@c02n01` terminal session, follow the instructions supplied in the latest `physru.sh` output to disable Oracle RAC and restart a single instance of the physical standby database.

```
[oracle@c02n01 ~]$ srvctl stop instance -d c02orcl -i c02orcl2 -o abort
[oracle@c02n01 ~]$ sqlplus / as sysdba

SQL*Plus: Release 11.2.0.4.0 Production ...

SQL> alter system set cluster_database=false scope=spfile;

System altered.

SQL> shutdown abort
ORACLE instance shut down.
SQL> startup mount
ORACLE instance started.

Total System Global Area 1503199232 bytes
Fixed Size                  2253424 bytes
Variable Size             452988304 bytes
Database Buffers         1040187392 bytes
Redo Buffers                7770112 bytes
Database mounted.
SQL> exit
Disconnected from Oracle Database 11g Enterprise Edition ...
[oracle@c02n01 ~]$
```

68. After reconfiguring your standby database, return to your `physru.sh` session and enter `y` to continue.

```
Are you ready to continue? (y/n): y

Aug 25 06:20:49 2014 [2-1] continuing
Aug 25 06:20:49 2014 [2-1] verifying RAC is disabled at c02orcl
Aug 25 06:20:49 2014 [2-1] verifying database roles
Aug 25 06:20:49 2014 [2-1] verifying physical standby is mounted
Aug 25 06:20:49 2014 [2-1] verifying database protection mode
Aug 25 06:20:49 2014 [2-1] verifying transient logical standby datatype
support
```

```
WARN: Objects have been identified on the primary database which will not be
      replicated on the transient logical standby.  The complete list of
      objects and their associated unsupported datatypes can be found in the
      dba_logstdby_unsupported view.  For convenience, this script has written
      the contents of this view to a file - physru_unsupported.log.

      Various options exist to deal with these objects such as:
        - disabling applications that modify these objects
        - manually resolving these objects after the upgrade
        - extending support to these objects (see metalink note: 559353.1)

      If you need time to review these options, you should enter 'n' to exit
      the script.  Otherwise, you should enter 'y' to continue with the
      rolling upgrade.

Are you ready to proceed with the rolling upgrade? (y/n):
```

Note: Logical standby databases have limitations that prevent the replication of some primary database objects. If your databases include objects that are affected by these limitations, `physru.sh` pauses during stage 2 to alert you and the script produces a file that lists the affected objects. You can use this information to determine the best course of action for dealing with this situation. It is highly recommended that you should understand how the limitations associated with logical standby databases affect your database as one of the key planning tasks before attempting any rolling upgrade. To do this, you can also connect to your database and examine the contents of the `dba_logstdby_unsupported` view.

69. Leave your other terminal sessions open and establish another terminal session connected to `c01n01` as the `oracle` user. Examine the contents of the `physru_unsupported.log` file to list the objects in your database that are not supported by the logical standby database.

```
$ ssh oracle@c01n01
[oracle@c01n01 ~]$ cat physru_unsupported.log

OWNER                          TABLE_NAME                     --
------------------------------ ------------------------------
COLUMN_NAME                    ATTRIBUTES
------------------------------ ------------------------------
DATA_TYPE
-----------------------------------------------------------------
OE                             PURCHASEORDER
SYS_NC_ROWINFO$                Unsupported XML
OPAQUE

OE                             CATEGORIES_TAB
CATEGORY_NAME                  Object Table
VARCHAR2
```

```
OE                              CATEGORIES_TAB
CATEGORY_DESCRIPTION            Object Table
VARCHAR2

OE                              CATEGORIES_TAB
CATEGORY_ID                     Object Table
NUMBER

OE                              CATEGORIES_TAB
PARENT_CATEGORY_ID              Object Table
NUMBER

PM                              PRINT_MEDIA
AD_TEXTDOCS_NTAB                Unsupported Virtual Column
NESTED TABLE

PM                              PRINT_MEDIA
AD_GRAPHIC
BFILE

...

IX                              AQ$_STREAMS_QUEUE_TABLE_C
MSGCNT                          AQ queue table
NUMBER

SH                              DIMENSION_EXCEPTIONS
BAD_ROWID
ROWID

[oracle@c01n01 ~]$ exit
logout
Connection to c01n01 closed.
$
```

Note: In this practice, you will deal with the logical standby database limitations by ensuring that none of the affected objects are updated during the rolling upgrade process.

70. Return to your `physru.sh` session and enter `y` to continue.

```
Are you ready to proceed with the rolling upgrade? (y/n): y

Aug 25 06:22:45 2014 [2-1] continuing
Aug 25 06:22:46 2014 [2-2] starting media recovery on c02orcl
Aug 25 06:22:52 2014 [2-2] confirming media recovery is running
Aug 25 06:22:53 2014 [2-2] waiting for apply lag to fall under 30 seconds
Aug 25 06:26:42 2014 [2-2] apply lag measured at 229 seconds
Aug 25 06:26:46 2014 [2-2] apply lag measured at 4 seconds
```

```
Aug 25 06:26:46 2014 [2-2] stopping media recovery on c02orcl
Aug 25 06:26:47 2014 [2-2] executing dbms_logstdby.build on database c01orcl
Aug 25 06:27:33 2014 [2-2] converting physical standby into transient logical
standby
Aug 25 06:27:43 2014 [2-3] opening database c02orcl
Aug 25 06:27:57 2014 [2-4] configuring transient logical standby parameters
for rolling upgrade
Aug 25 06:27:59 2014 [2-4] starting logical standby on database c02orcl
Aug 25 06:28:08 2014 [2-4] waiting until logminer dictionary has fully loaded
Aug 25 06:31:03 2014 [2-4] dictionary load 22% complete
Aug 25 06:31:13 2014 [2-4] dictionary load 41% complete
Aug 25 06:31:23 2014 [2-4] dictionary load 62% complete
Aug 25 06:31:33 2014 [2-4] dictionary load 75% complete
Aug 25 06:31:54 2014 [2-4] dictionary load 99% complete
Aug 25 06:32:04 2014 [2-4] dictionary load is complete
Aug 25 06:32:07 2014 [2-4] waiting for apply lag to fall under 30 seconds
Aug 25 06:32:22 2014 [2-4] apply lag measured at 5 seconds

NOTE: Database c02orcl is now ready to be upgraded.  This script has left the
      database open in case you want to perform any further tasks before
      upgrading the database.  Once the upgrade is complete, the database must
      opened in READ WRITE mode before this script can be called to resume the
      rolling upgrade.

NOTE: Database c02orcl may be reverted back to a RAC database upon completion
      of the rdbms upgrade.  This can be accomplished by performing the
      following steps:

          1) On instance c02orcl1, set the cluster_database parameter to TRUE.
          eg: SQL> alter system set cluster_database=true scope=spfile;

          2) Shutdown instance c02orcl1.
          eg: SQL> shutdown abort;

          3) Startup and open all instances for database c02orcl.
          eg: srvctl start database -d c02orcl

[oracle@c01n01 ~]$
```

Note: During the final part of stage 2, the standby database (`c02orcl`) is converted into a transient logical standby. After this operation completes, the standby database is ready to be upgraded. In the next part of this practice you will use the Database Upgrade Assistant (DBUA) to upgrade `c02orcl`. However, before DBUA is started, the following preparations are required.

71. When DBUA is used on an Oracle RAC database, it requires all the database instances to be open at the beginning of the DBUA session. Therefore, using the following commands, re-enable Oracle RAC for `c02orcl` and restart all the database instances.

```
[oracle@c02n01 ~]$ sqlplus / as sysdba

SQL*Plus: Release 11.2.0.4.0 Production ...

SQL> alter system set cluster_database=true scope=spfile;

System altered.

SQL> shutdown abort
ORACLE instance shut down.
SQL> exit
Disconnected from Oracle Database 11g Enterprise Edition ...
[oracle@c02n01 ~]$ srvctl start database -d c02orcl
[oracle@c02n01 ~]$
```

Note: In your practice environment, you will use a preinstalled set of Oracle Database 12*c* binaries, which were previously installed by using Oracle Universal Installer to perform a software-only installation. This approach was taken so that you would not have to wait for the binaries to be installed. However, to use the pre-installed binaries in conjunction with DBUA, you must first run the root configuration script (`root.sh`) located in the Oracle Database 12*c* home directory.

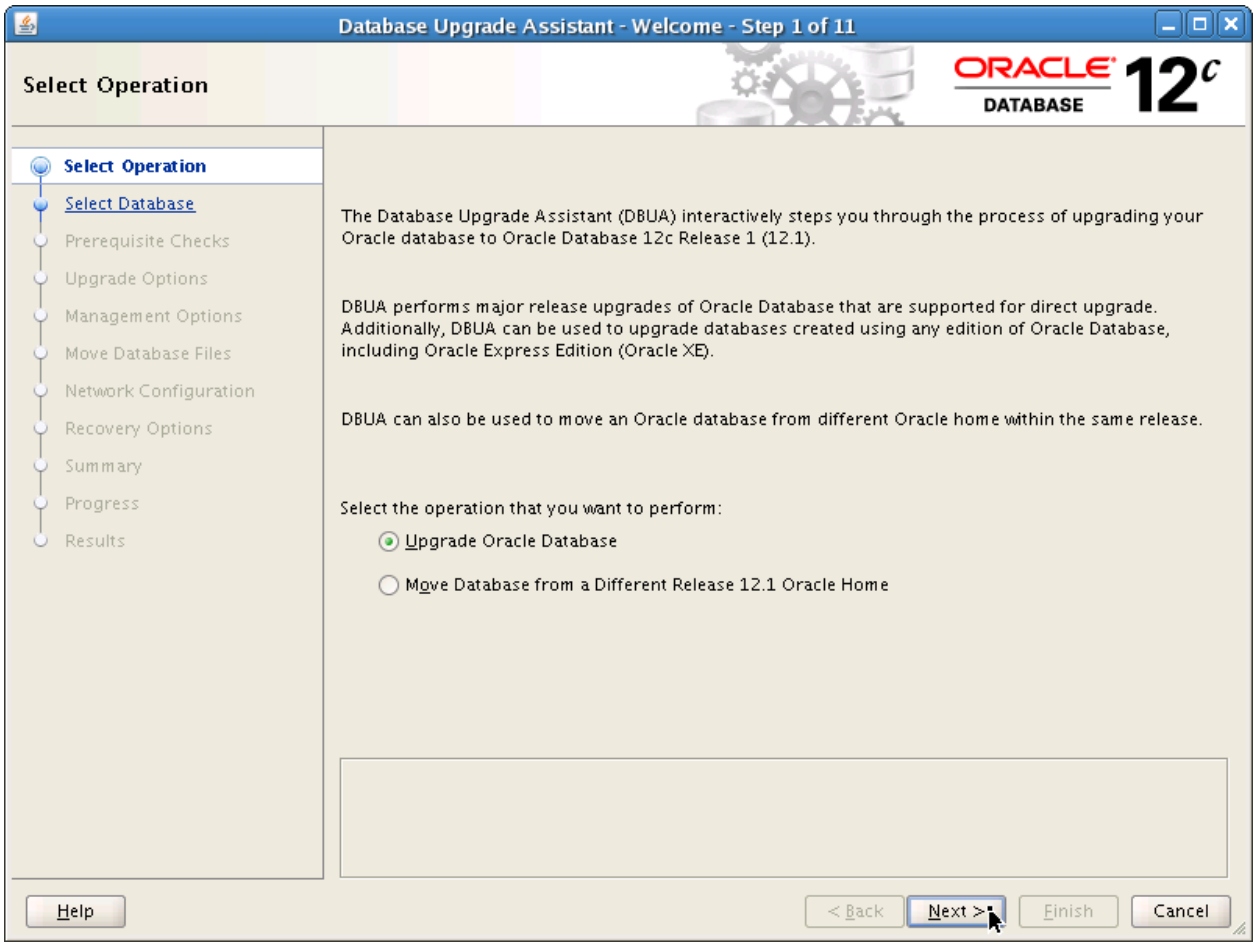
72. Run the root configuration script (`root.sh`) located in `/u01/app/oracle/product/12.1.0/dbhome_1` on `c02n01` and `c02n02`.

```
[oracle@c02n01 ~]$ su -
Password: <oracle>
[root@c02n01 ~]# /u01/app/oracle/product/12.1.0/dbhome_1/root.sh
Check
/u01/app/oracle/product/12.1.0/dbhome_1/install/root_c02n01.example.com_2014-
08-25_07-06-04.log for the output of root script
[root@c02n01 ~]# ssh c02n02
[root@c02n02 ~]# /u01/app/oracle/product/12.1.0/dbhome_1/root.sh
Check
/u01/app/oracle/product/12.1.0/dbhome_1/install/root_c02n02.example.com_2014-
08-25_07-06-34.log for the output of root script
[root@c02n02 ~]# exit
logout
Connection to c02n02 closed.
[root@c02n01 ~]# exit
logout
[oracle@c02n01 ~]$
```

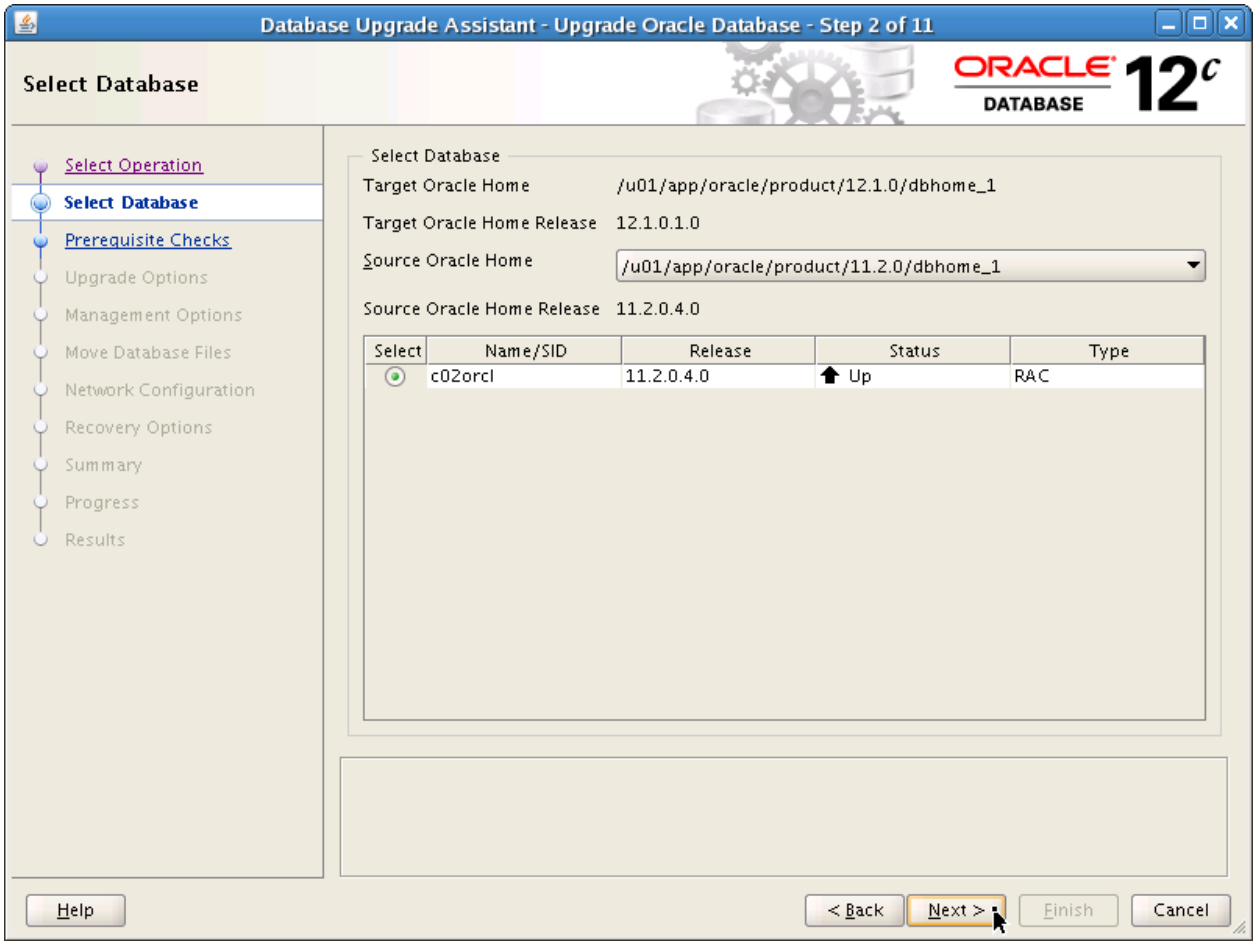
73. Launch the Oracle Database 12*c* Database Upgrade Assistant (DBUA).

```
[oracle@c02n01 ~]$ /u01/app/oracle/product/12.1.0/dbhome_1/bin/dbua &
```

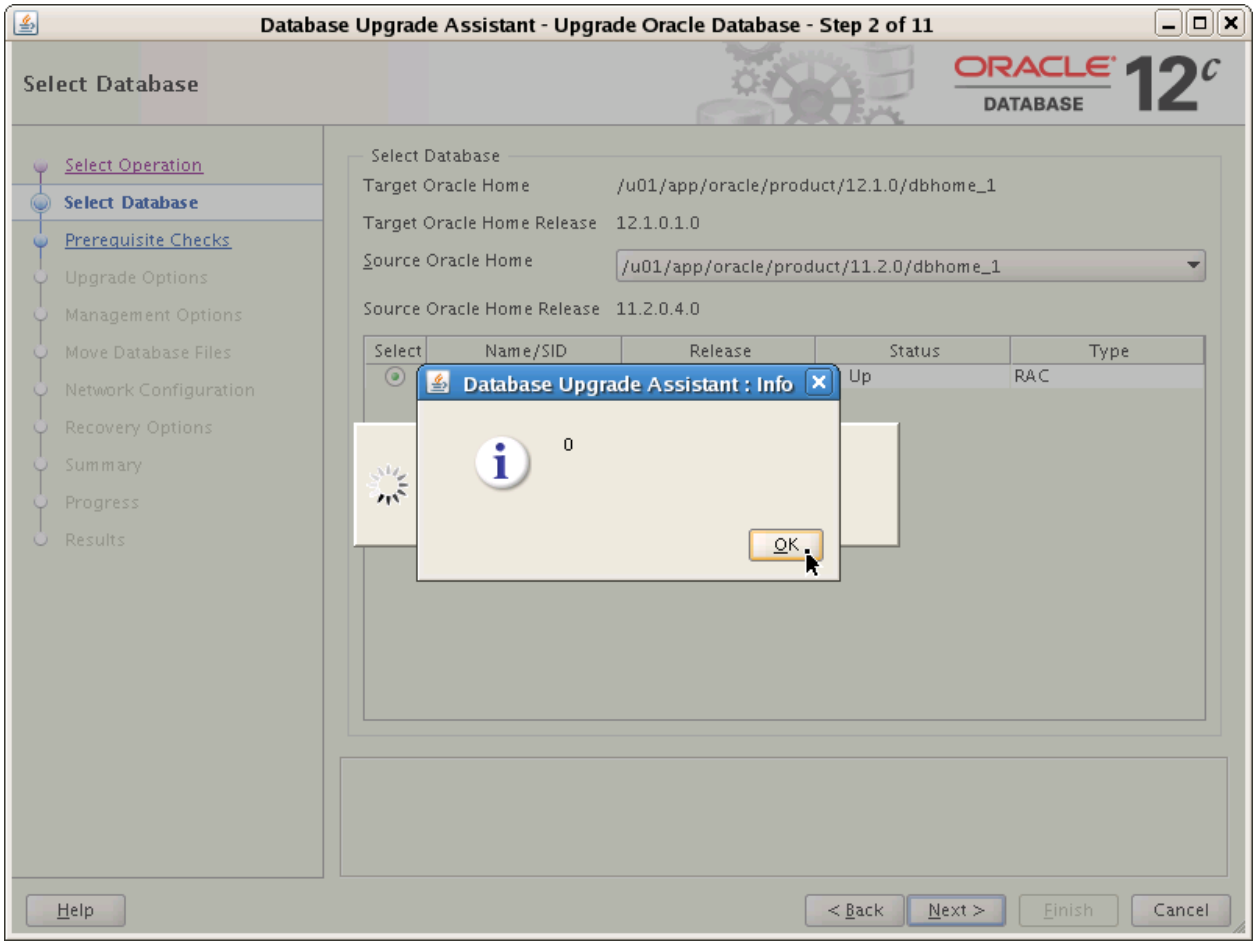
74. On the Select Operation page, ensure that the Upgrade Oracle Database option is selected and click Next to proceed.



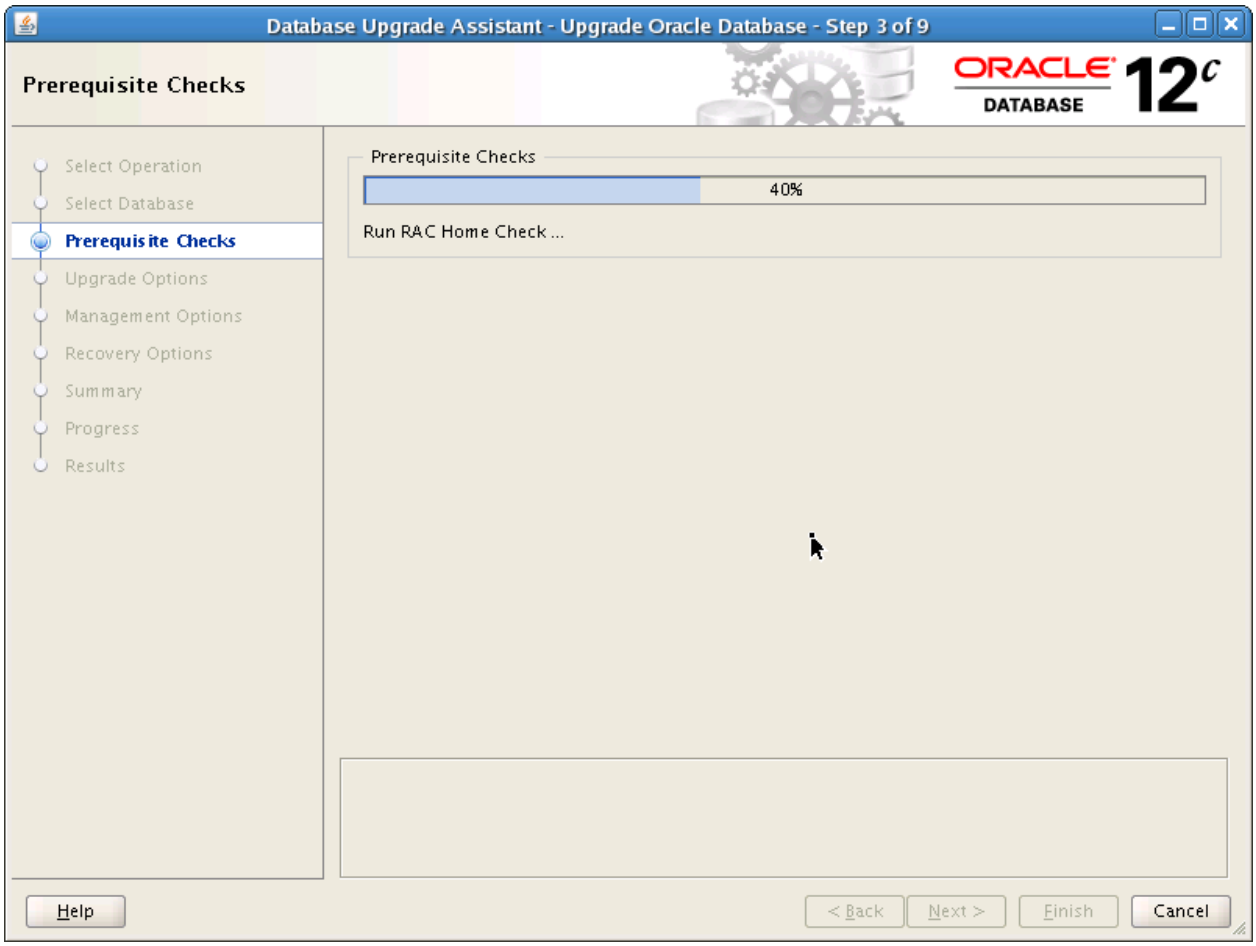
75. On the Select Database page, ensure that the database `c02orcl` is selected. Then, click Next to proceed.



76. Because of an undocumented feature of this DBUA version, you may see a dialog box like the one in the following screenshot. Click OK to close the dialog box and proceed.



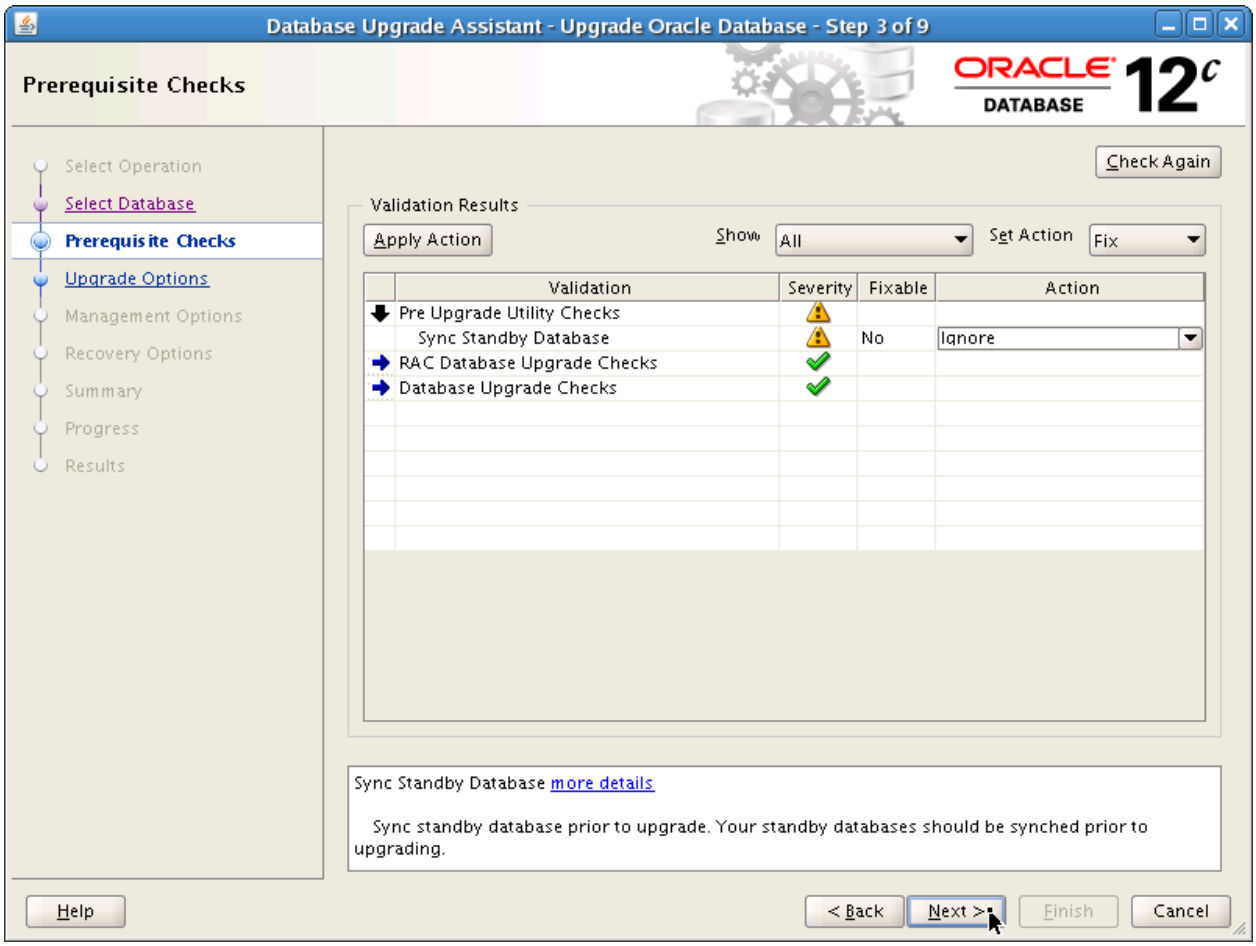
77. Wait while a series of checks are performed.



Notes

- The results of the prerequisite checks will vary depending on the composition of the database being upgraded. It is recommended that you run the DBUA prerequisite checks and understand their output as part of your upgrade planning.
- In your practice environment database, a warning will be generated indicating that your standby database should be synchronized before upgrading. However, you can safely ignore this warning because the `physru.sh` script has already prepared the database to be upgraded.

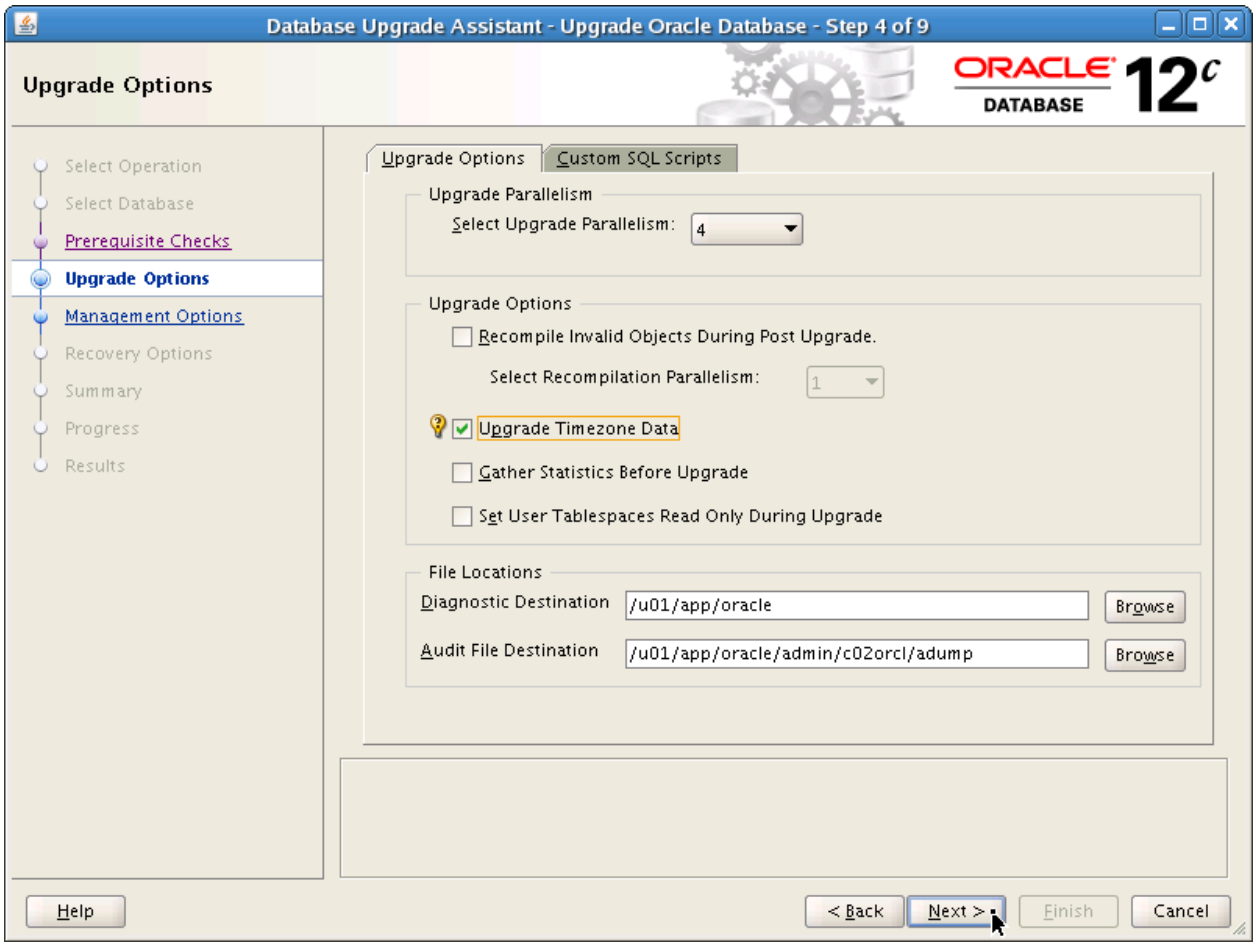
78. Click the arrow beside Pre Upgrade Utility Checks to reveal the Sync Standby Database warning. After you have confirmed that there are no other errors or warnings, click Next to proceed.



79. On the Upgrade Options page, deselect the Recompile Invalid Objects During Post Upgrade check box and select the Upgrade Timezone Data check box. Then click Next to proceed.

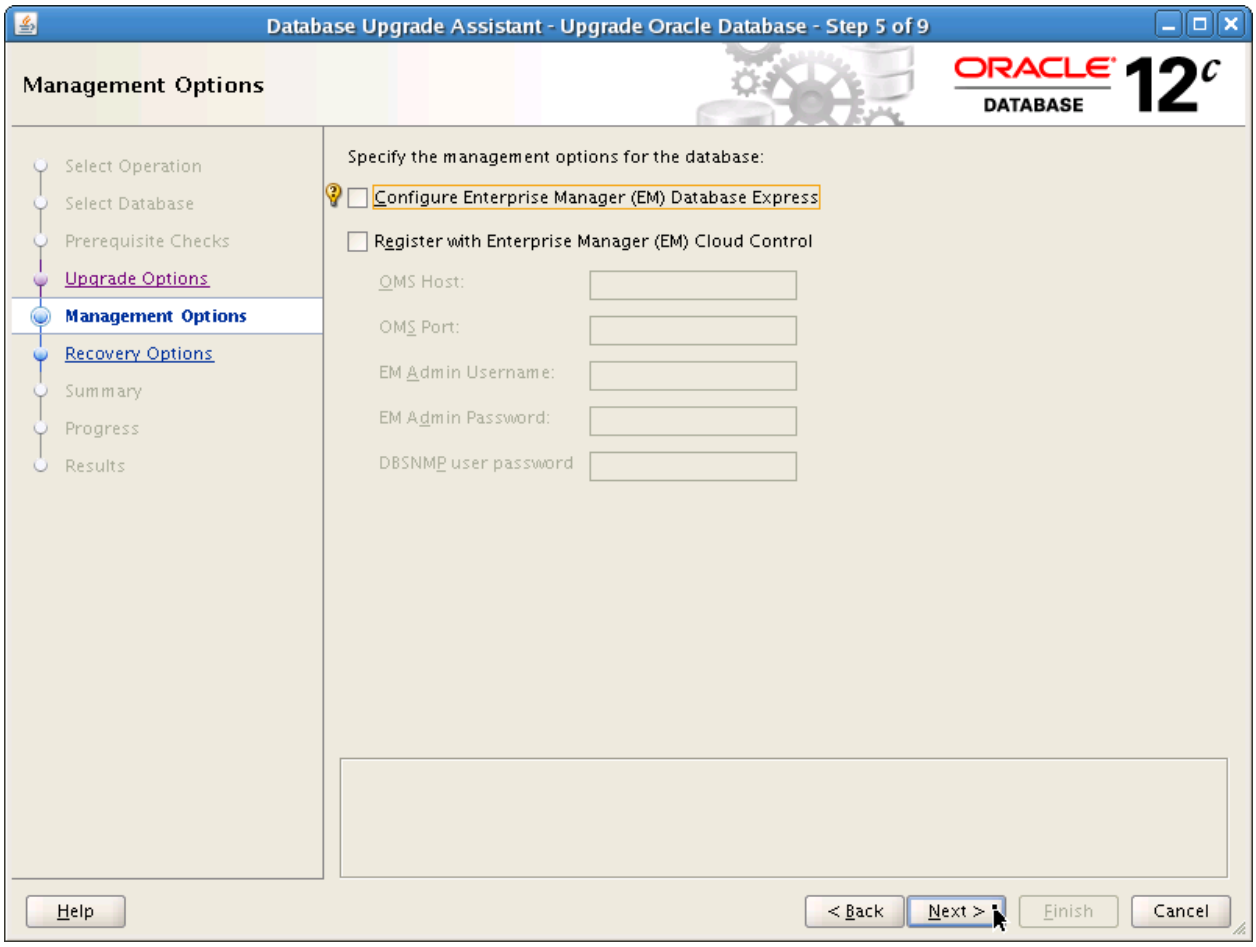
Notes

- It is generally recommended to allow DBUA to recompile invalid database objects during post upgrade processing. However, because this is an optional task, and because it can add approximately 30 minutes to the upgrade process in your practice environment, it is recommended to deselect this option for this practice exercise.
- Regarding the available DBUA options, you should consider the best options to use when performing your own upgrades.



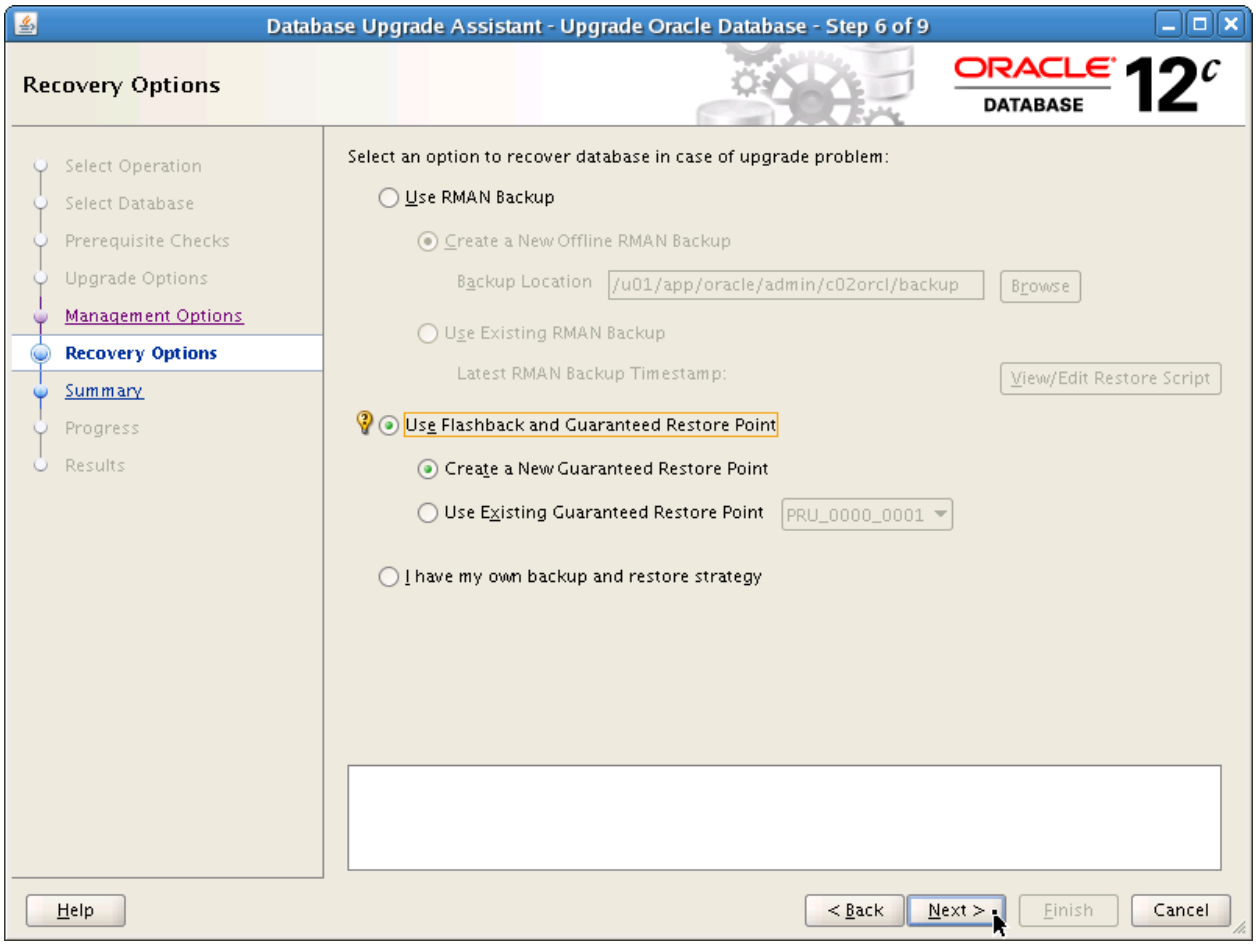
80. On the Management Options page, deselect the Configure Enterprise Manager Database Express check box. Then click Next to proceed.

    Note: If you want to use Enterprise Manager Database Express in your upgraded Oracle Data Guard environment, it should be configured separately on each database after the rolling upgrade process is completed.

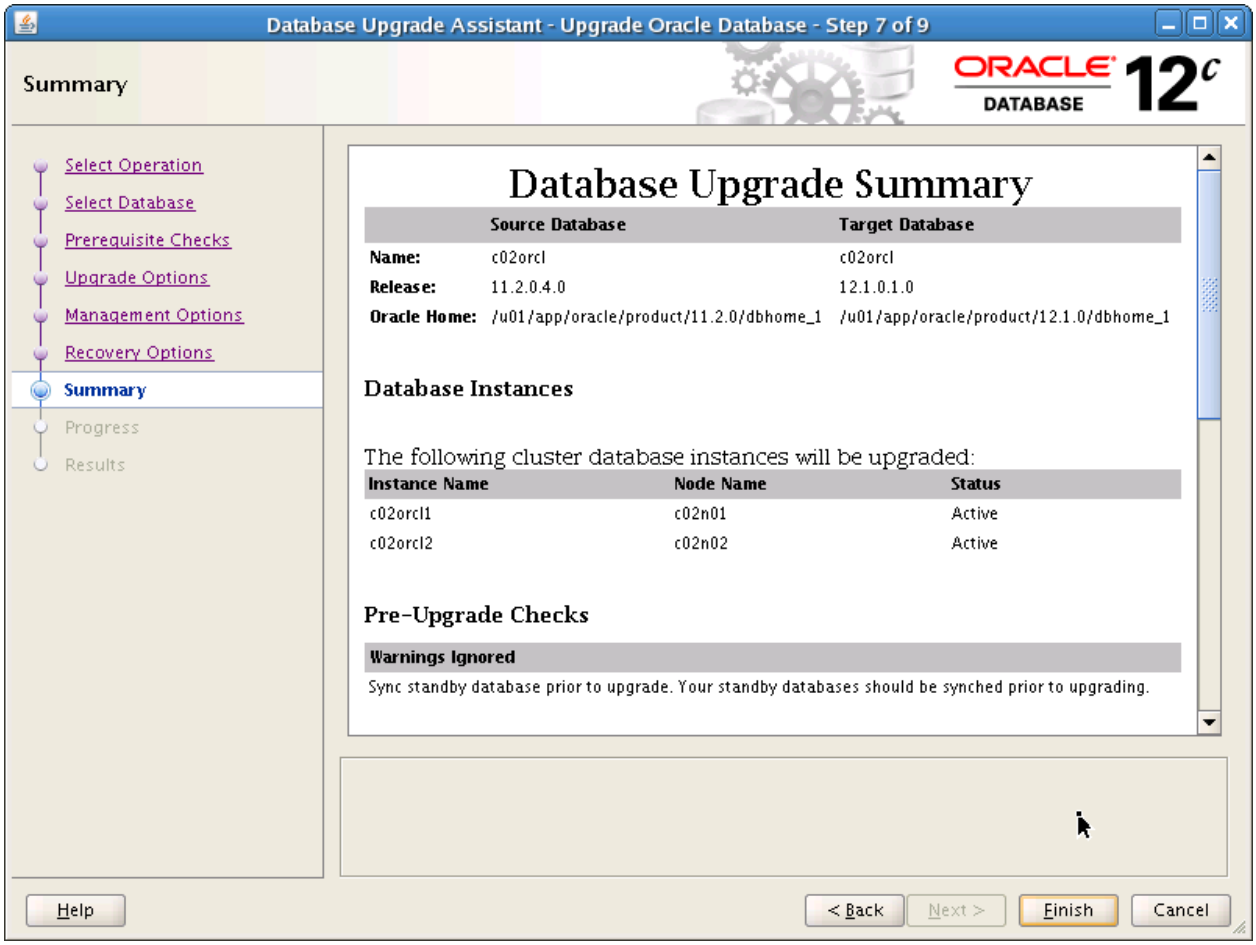


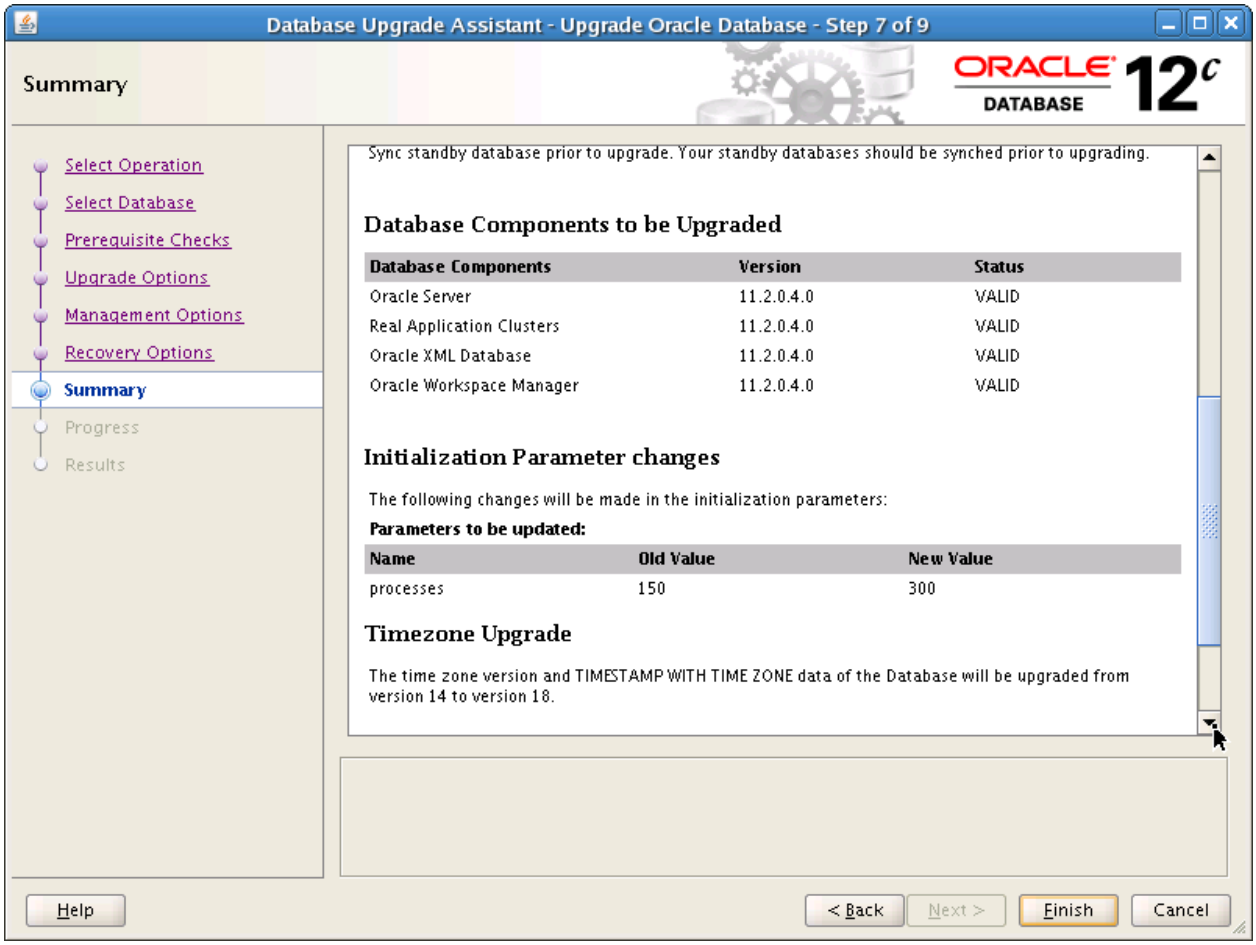
81. On the Recovery Options page, select the “Use Flashback and Guaranteed Restore Point” check box. Then, select the “Create a New Guaranteed Restore Point” check box. Finally, click Next to proceed.

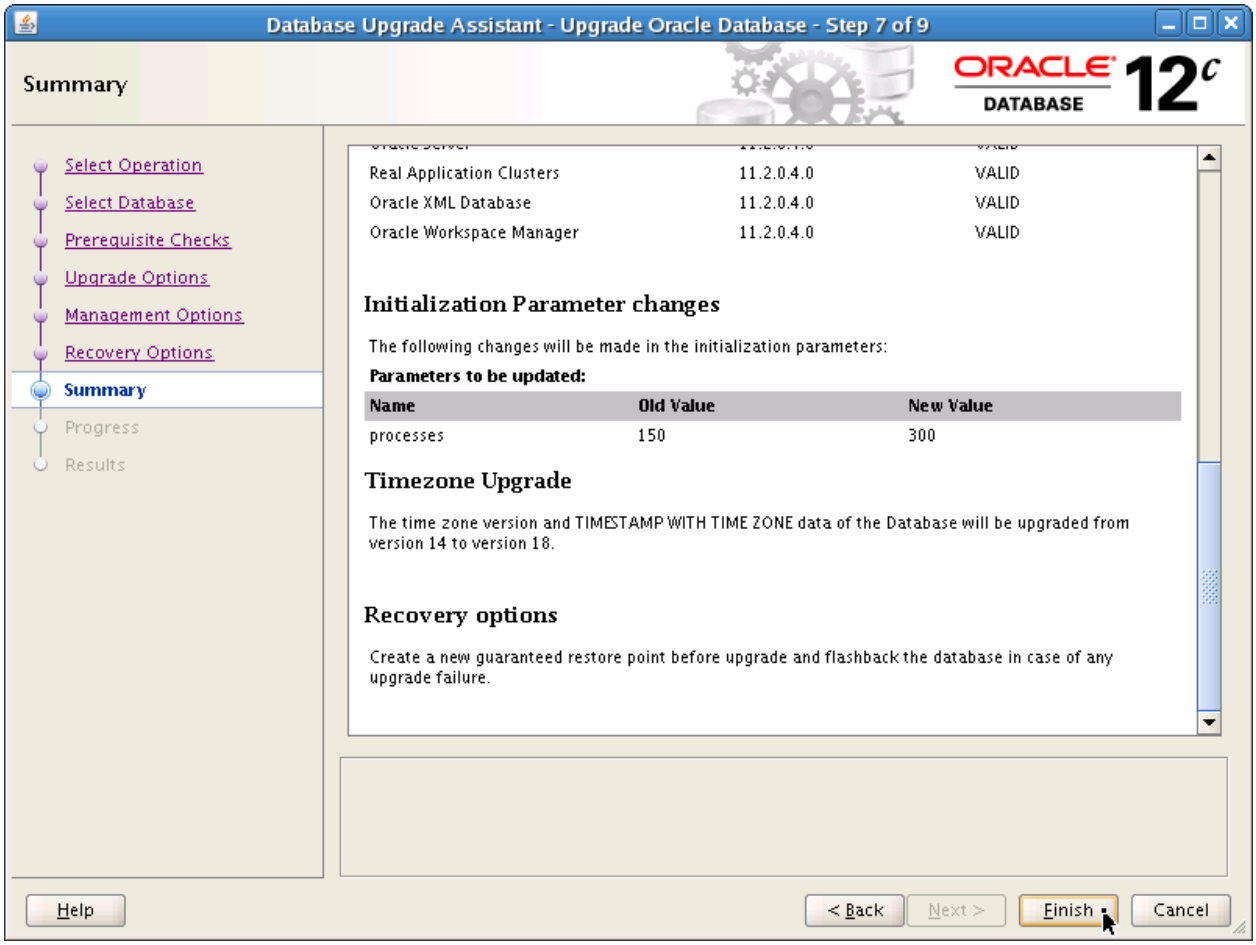
    Note: Though not an absolute requirement, a guaranteed restore point provides a quick and easy mechanism to recover the database back to this point in case of a problem during the upgrade.



82. Investigate the information on the Summary page. After you are satisfied, click Finish to start the database upgrade.

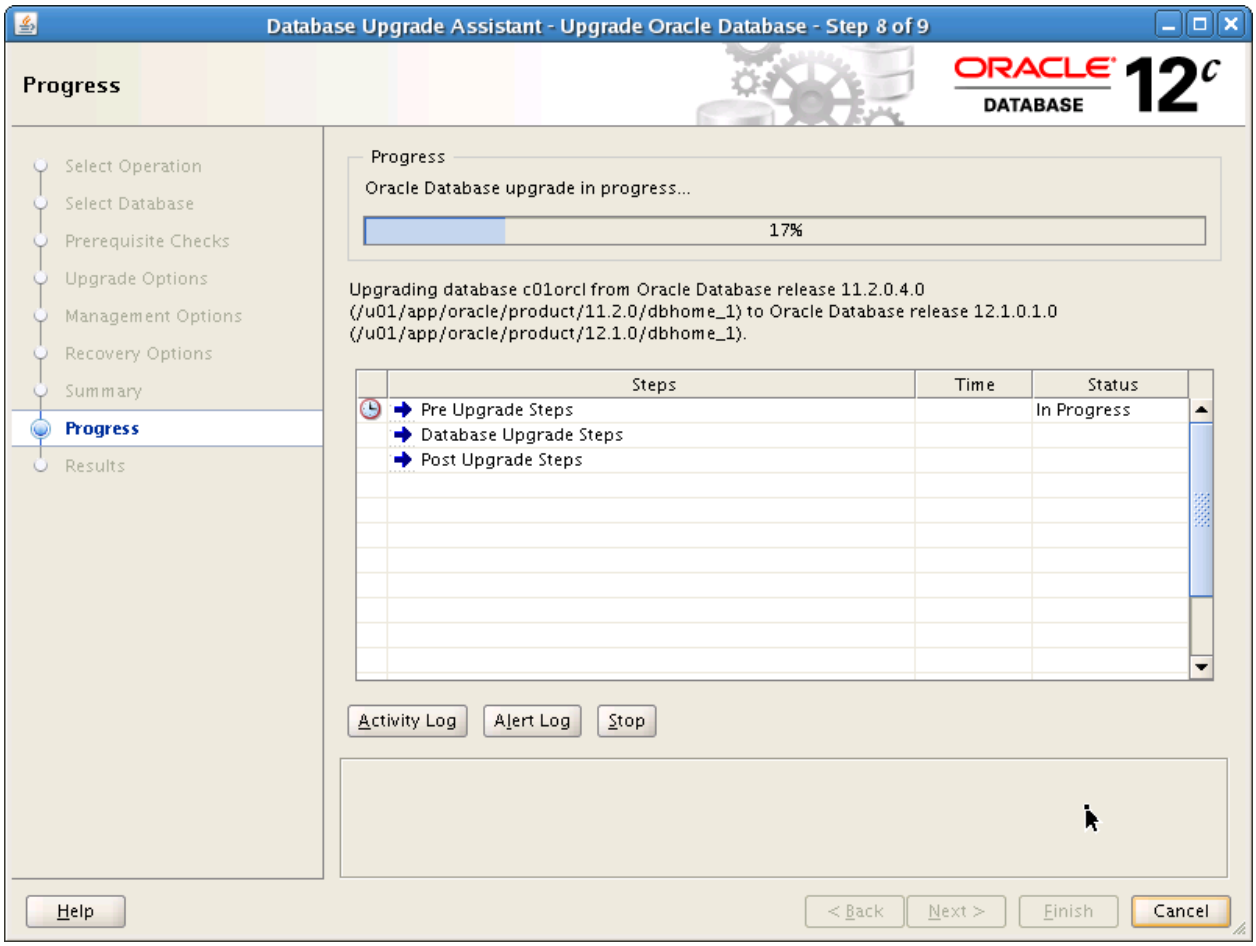




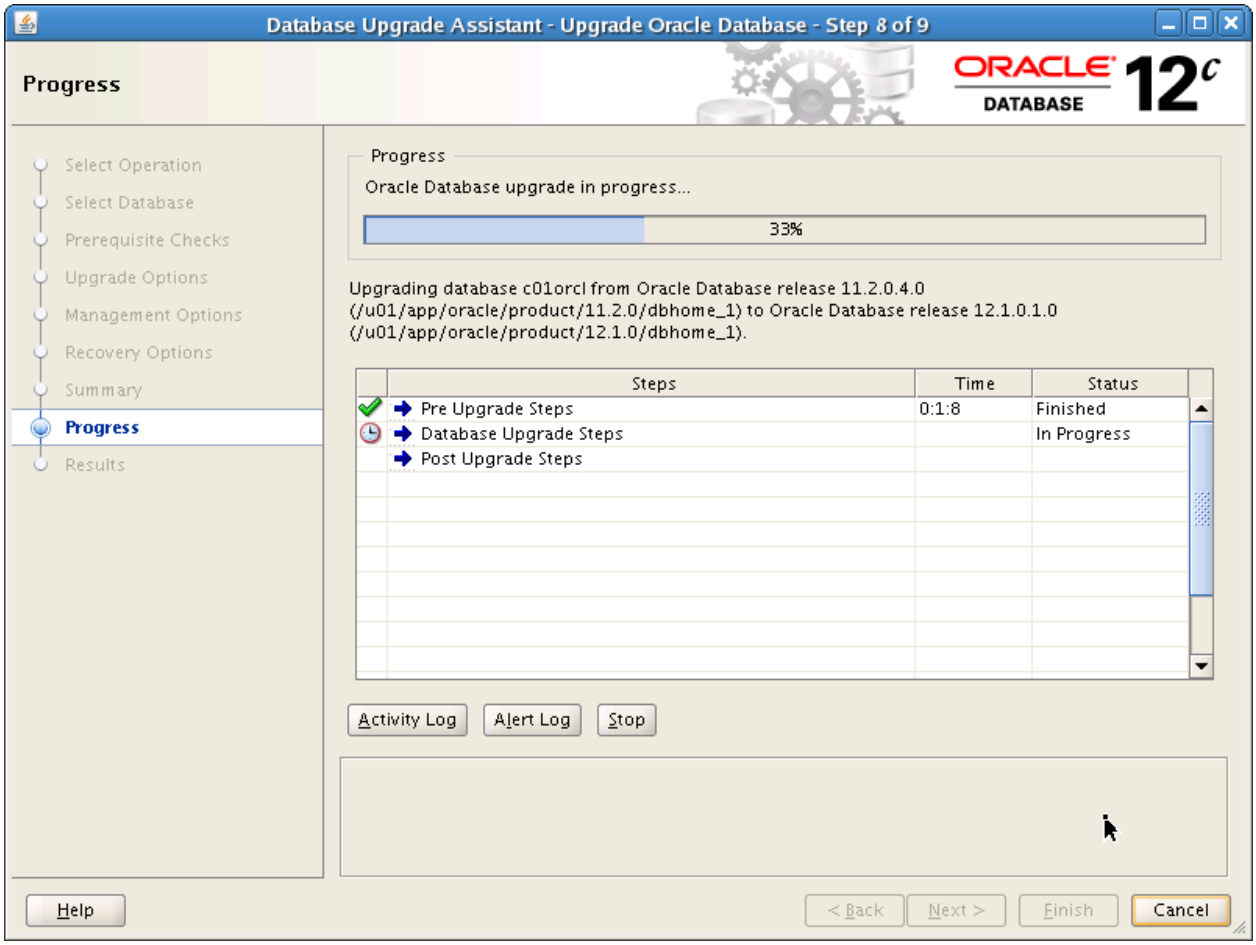


83. Monitor DBUA by viewing the Progress page.

Note: You can also monitor the DBUA session by viewing the log files at `/u01/app/oracle/cfgtoollogs/dbua/c02orcl/upgrade1` on `c02n01`. You can also monitor the database alert log files for `c02orcl`.



Note: During the Pre Upgrade Steps, the database is shut down and only one instance is restarted to perform the upgrade.

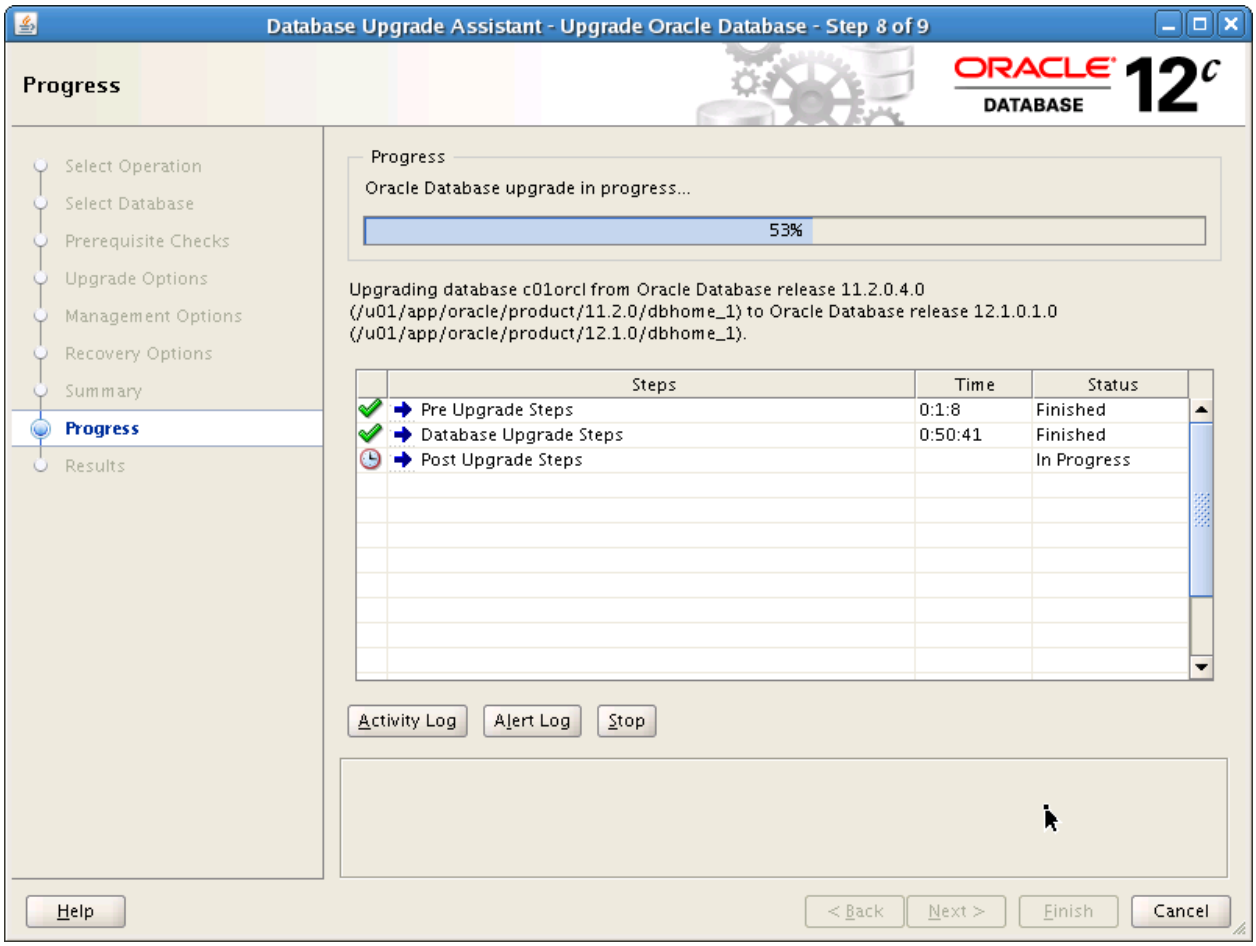


84. While the standby database (`c02orcl`) is being upgraded, the primary database (`c01orcl`) remains open for business, and all the transactions executed on the primary database will be applied to the transient logical standby database after it is upgraded. To illustrate this, use your `oracle@c01n01` terminal session to launch SQL*Plus and add some data to the `hr.regions` table. Later, after the rolling upgrade is finished, you will check to ensure that this database change has been persisted through the entire upgrade process.

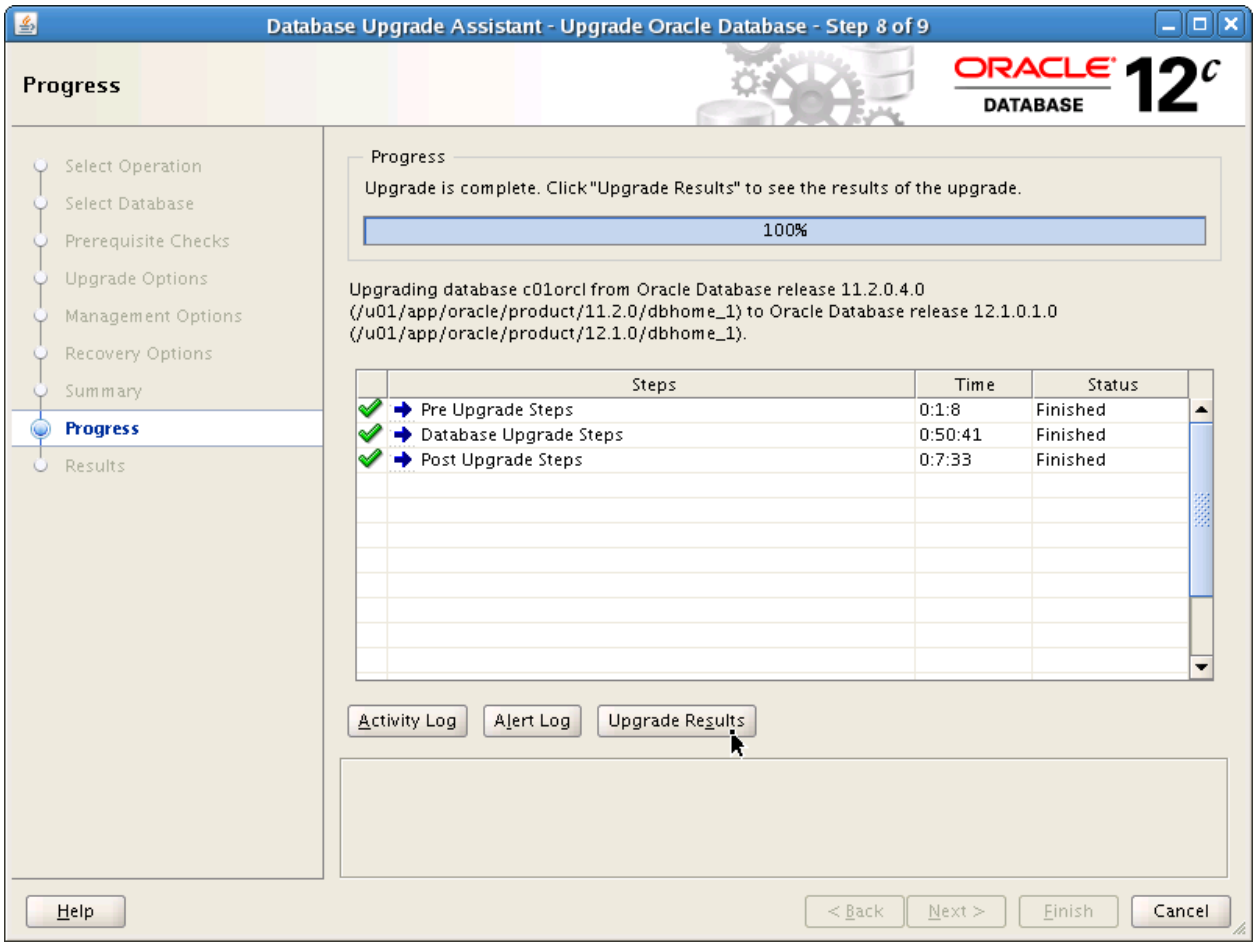
```
[oracle@c01n01 ~]$ sqlplus hr/hr

SQL*Plus: Release 11.2.0.4.0 Production ...

SQL> select * from regions;

 REGION_ID REGION_NAME
---------- -------------------------
         1 Europe
         2 Americas
         3 Asia
         4 Middle East and Africa
```

```
SQL> insert into regions values (5,'Australia');

1 row created.

SQL> commit;

Commit complete.

SQL> exit
Disconnected from Oracle Database 11g Enterprise Edition ...
[oracle@c01n01 ~]$
```

Note: The DBUA Database Upgrade Steps can take approximately 50 minutes to complete in your practice environment.

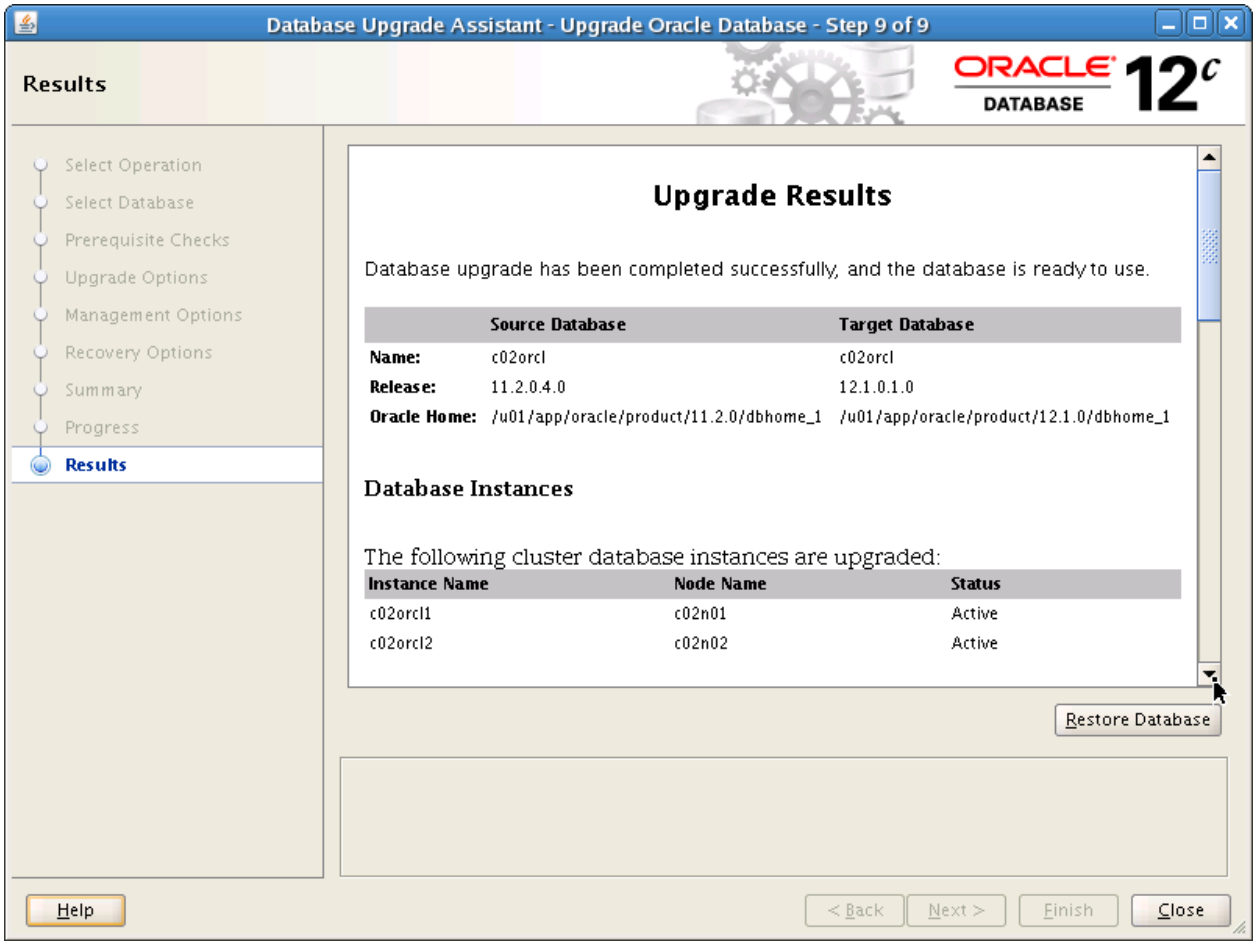


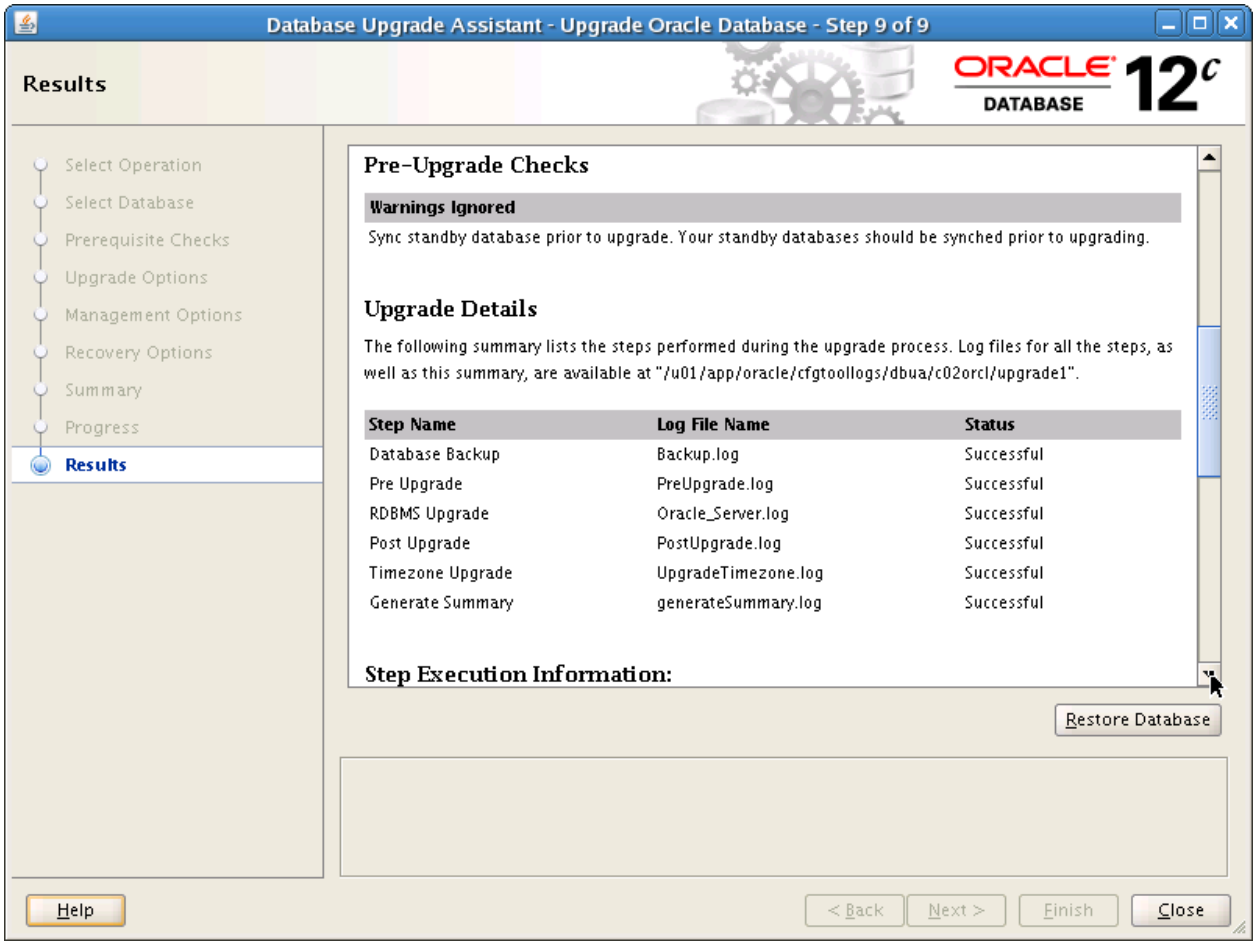
85. When the database upgrade is finished, you should see a page that looks similar to the following. Click Upgrade Results to leave the Progress page and view the detailed results page.

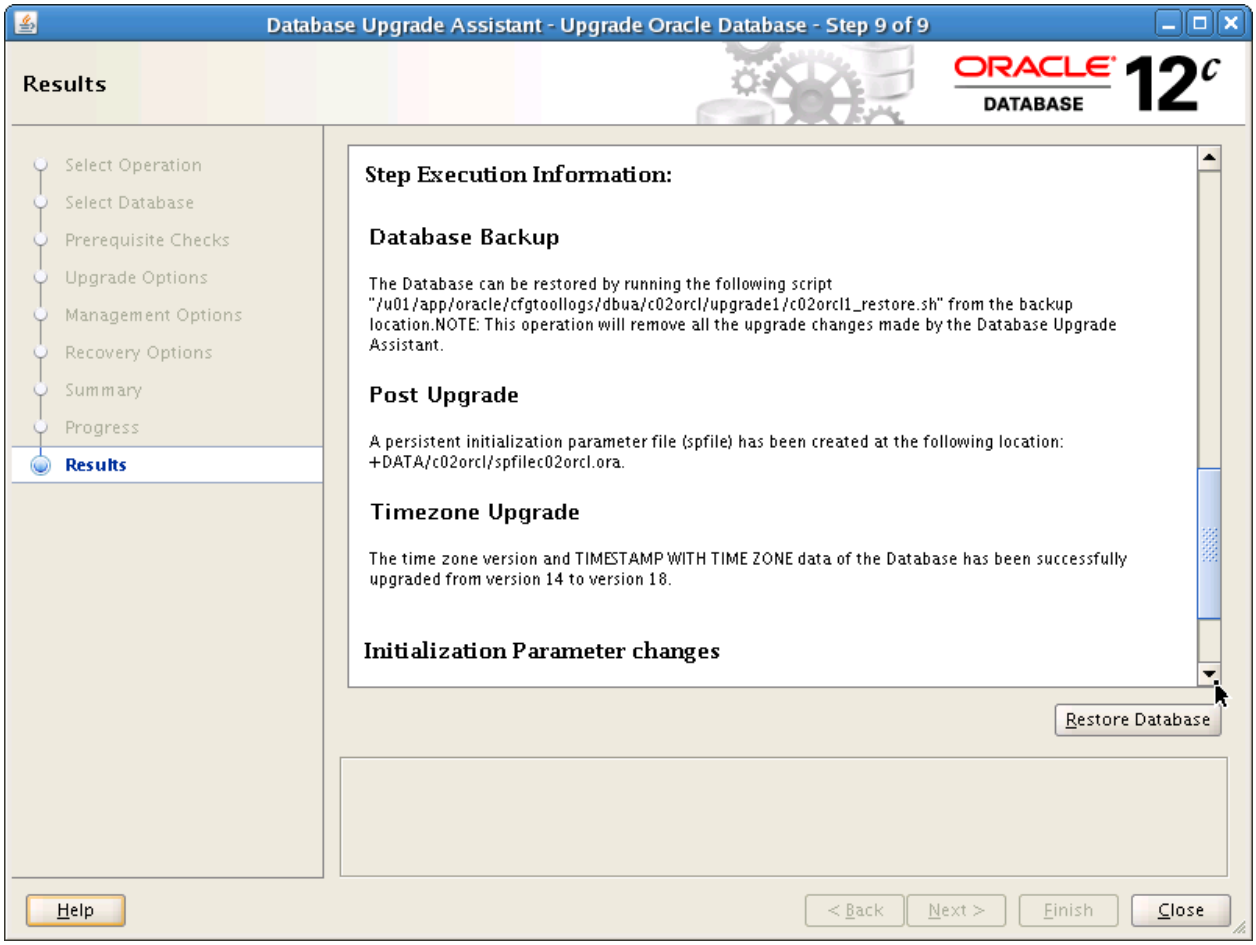


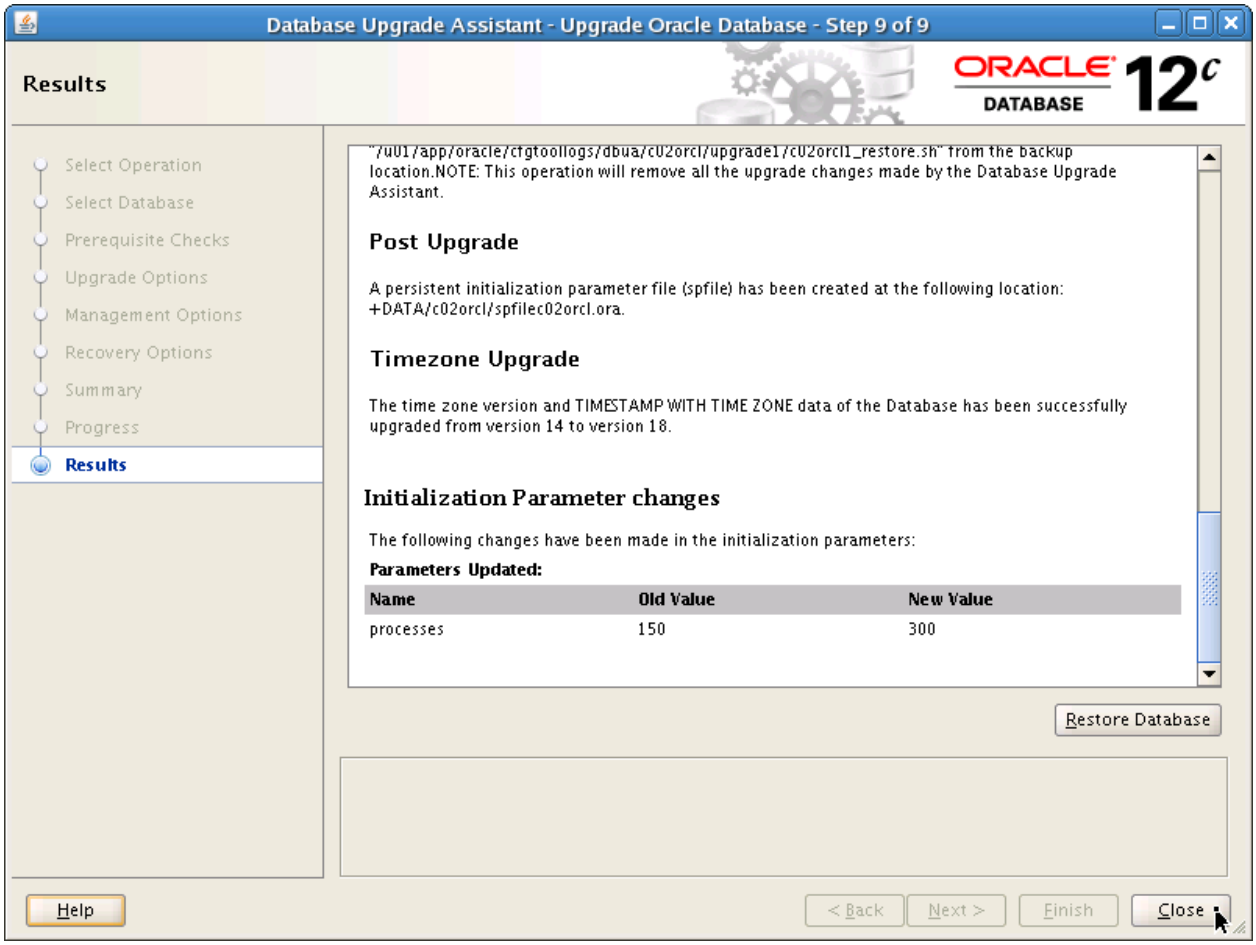
86. Investigate the information on the Results page. After you are satisfied, click Close to exit DBUA.

    Note: The results page may contain information regarding errors or warning generated during the database upgrade. Make sure that you note any information that requires further action. The results are also saved in a file for later examination, if required. The upgrade results for the upgrade you just performed can be found on `c02n01` at `/u01/app/oracle/cfgtoollogs/dbua/c02orcl/upgrade1/UpgradeResults.html`.









87. Refresh the environment settings in your `oracle@c02n01` terminal session to ensure that you use the upgraded Oracle Database 12*c* software from now on.

```
[oracle@c02n01 ~]$ . oraenv
ORACLE_SID = [c02orcl1] ? c02orcl
The Oracle base remains unchanged with value /u01/app/oracle
[oracle@c02n01 ~]$ export ORACLE_SID=c02orcl1
[oracle@c02n01 ~]$
```

88. Check the configuration of the standby database (`c02orcl`) and confirm that it is now associated with the Oracle Database 12*c* binaries at `/u01/app/oracle/product/12.1.0/dbhome_1`.

```
[oracle@c02n01 ~]$ srvctl config database -d c02orcl -v
Database unique name: c02orcl
Database name:
Oracle home: /u01/app/oracle/product/12.1.0/dbhome_1
Oracle user: oracle
Spfile: +DATA/c02orcl/spfilec02orcl.ora
Password file:
Domain: example.com
Start options: open
Stop options: immediate
```

```
Database role: PHYSICAL_STANDBY
Management policy: AUTOMATIC
Server pools: c02orcl
Database instances: c02orcl1,c02orcl2
Disk Groups: DATA
Mount point paths:
Services:
Type: RAC
Start concurrency:
Stop concurrency:
Database is administrator managed
[oracle@c02n01 ~]$
```

89. Confirm that the upgraded standby database is running.

```
[oracle@c02n01 ~]$ srvctl status database -d c02orcl -v
Instance c02orcl1 is running on node c02n01. Instance status:
Open.
Instance c02orcl2 is running on node c02n02. Instance status:
Open.
[oracle@c02n01 ~]$
```

Note: At this point, the standby database (`c02orcl`) is upgraded; however, the Oracle Net configuration surrounding the database has not been updated. Before `physru.sh` is restarted, you must perform the following configuration tasks.

90. On every node in `cluster02`, copy the existing `tnsnames.ora` files from the current location under the Oracle Database 11*g* home directory into the corresponding location under the Oracle Database 12*c* home directory.

```
[oracle@c02n01 ~]$ cp
/u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.o
ra /u01/app/oracle/product/12.1.0/dbhome_1/network/admin
[oracle@c02n01 ~]$ ssh c02n02 cp
/u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.o
ra /u01/app/oracle/product/12.1.0/dbhome_1/network/admin
[oracle@c02n01 ~]$
```

91. Using your `grid@c02n01` terminal session, view the Oracle Net Listener configuration file (`listener.ora`). Notice that the static registration entry associated with `c02orcl1` still references the Oracle Database 11*g* home directory.

```
[grid@c02n01 ~]$ cat /u01/app/12.1.0/grid/network/admin/listener.ora
LISTENER=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER))))
# line added by Agent
LISTENER_SCAN3=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER
_SCAN3))))          # line added by Agent
LISTENER_SCAN2=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER
_SCAN2))))          # line added by Agent
LISTENER_SCAN1=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER
_SCAN1))))          # line added by Agent
ENABLE_GLOBAL_DYNAMIC_ENDPOINT_LISTENER_SCAN1=ON                # line added
by Agent
```

```
ENABLE_GLOBAL_DYNAMIC_ENDPOINT_LISTENER_SCAN2=ON                # line added
by Agent
ENABLE_GLOBAL_DYNAMIC_ENDPOINT_LISTENER_SCAN3=ON                # line added
by Agent
SID_LIST_LISTENER=(SID_LIST=(SID_DESC=(GLOBAL_DBNAME=c02orcl_DGMGRL.example.co
m)(ORACLE_HOME=/u01/app/oracle/product/11.2.0/dbhome_1)(SID_NAME=c02orcl1)))
VALID_NODE_CHECKING_REGISTRATION_LISTENER_SCAN1=OFF             # line added
by Agent
VALID_NODE_CHECKING_REGISTRATION_LISTENER_SCAN2=OFF             # line added
by Agent
VALID_NODE_CHECKING_REGISTRATION_LISTENER_SCAN3=OFF             # line added
by Agent
ENABLE_GLOBAL_DYNAMIC_ENDPOINT_LISTENER=ON              # line added by Agent
VALID_NODE_CHECKING_REGISTRATION_LISTENER=SUBNET                # line added
by Agent
[grid@c02n01 ~]$
```

92. Use the following command to update the `listener.ora` file so that all references to `11.2` are replaced by `12.1`.

```
[grid@c02n01 ~]$ sed -i s#11.2#12.1#g
/u01/app/12.1.0/grid/network/admin/listener.ora
[grid@c02n01 ~]$
```

93. View the Oracle Net Listener configuration file again and confirm that the static registration entry associated with `c02orcl1` now references the Oracle Database 12*c* home directory.

```
[grid@c02n01 ~]$ cat /u01/app/12.1.0/grid/network/admin/listener.ora
LISTENER=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER))))
# line added by Agent
LISTENER_SCAN3=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER
_SCAN3))))          # line added by Agent
LISTENER_SCAN2=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER
_SCAN2))))          # line added by Agent
LISTENER_SCAN1=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER
_SCAN1))))          # line added by Agent
ENABLE_GLOBAL_DYNAMIC_ENDPOINT_LISTENER_SCAN1=ON                # line added
by Agent
ENABLE_GLOBAL_DYNAMIC_ENDPOINT_LISTENER_SCAN2=ON                # line added
by Agent
ENABLE_GLOBAL_DYNAMIC_ENDPOINT_LISTENER_SCAN3=ON                # line added
by Agent
SID_LIST_LISTENER=(SID_LIST=(SID_DESC=(GLOBAL_DBNAME=c02orcl_DGMGRL.example.co
m)(ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1)(SID_NAME=c02orcl1)))
VALID_NODE_CHECKING_REGISTRATION_LISTENER_SCAN1=OFF             # line added
by Agent
VALID_NODE_CHECKING_REGISTRATION_LISTENER_SCAN2=OFF             # line added
by Agent
VALID_NODE_CHECKING_REGISTRATION_LISTENER_SCAN3=OFF             # line added
by Agent
ENABLE_GLOBAL_DYNAMIC_ENDPOINT_LISTENER=ON              # line added by Agent
VALID_NODE_CHECKING_REGISTRATION_LISTENER=SUBNET                # line added
by Agent
[grid@c02n01 ~]$
```

94. Using the same method as before, modify the `listener.ora` file on `c02n02`.

```
[grid@c02n01 ~]$ ssh c02n02 sed -i s#11.2#12.1#g
/u01/app/12.1.0/grid/network/admin/listener.ora
[grid@c02n01 ~]$
```

95. Confirm the update that you made in the previous step.

```
[grid@c02n01 ~]$ ssh c02n02 cat
/u01/app/12.1.0/grid/network/admin/listener.ora
LISTENER=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER))))
# line added by Agent
LISTENER_SCAN3=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER
_SCAN3))))          # line added by Agent
LISTENER_SCAN2=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER
_SCAN2))))          # line added by Agent
LISTENER_SCAN1=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER
_SCAN1))))          # line added by Agent
ENABLE_GLOBAL_DYNAMIC_ENDPOINT_LISTENER_SCAN1=ON                # line added
by Agent
SID_LIST_LISTENER=(SID_LIST=(SID_DESC=(GLOBAL_DBNAME=c02orcl_DGMGRL.example.co
m)(ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1)(SID_NAME=c02orcl2)))
ENABLE_GLOBAL_DYNAMIC_ENDPOINT_LISTENER_SCAN2=ON                # line added
by Agent
ENABLE_GLOBAL_DYNAMIC_ENDPOINT_LISTENER_SCAN3=ON                # line added
by Agent
VALID_NODE_CHECKING_REGISTRATION_LISTENER_SCAN1=OFF             # line added
by Agent
ENABLE_GLOBAL_DYNAMIC_ENDPOINT_LISTENER=ON              # line added by Agent
VALID_NODE_CHECKING_REGISTRATION_LISTENER=SUBNET                # line added
by Agent
[grid@c02n01 ~]$
```

96. Stop and restart the Oracle Net listener across `cluster02`.

```
[grid@c02n01 ~]$ srvctl stop listener
[grid@c02n01 ~]$ srvctl start listener
[grid@c02n01 ~]$
```

97. Using your `oracle@c01n01` terminal session, restart `physru.sh` as shown below in order to continue the database rolling upgrade process. Enter `oracle_4U` when you are prompted for the `sysdba` password.

```
[oracle@c01n01 ~]$ ./physru.sh sys c01n01:1521/c01orcl_DGMGRL.example.com
c02n01:1521/c02orcl_DGMGRL.example.com c01orcl c02orcl 12.1.0.1.0
Please enter the sysdba password: <oracle_4U>

### Initialize script to either start over or resume execution
Aug 25 08:35:23 2014 [0-1] Identifying rdbms software version
Aug 25 08:35:23 2014 [0-1] database c01orcl is at version 11.2.0.4.0
Aug 25 08:35:23 2014 [0-1] database c02orcl is at version 12.1.0.1.0
Aug 25 08:35:24 2014 [0-1] verifying flashback database is enabled at c01orcl
and c02orcl
Aug 25 08:35:25 2014 [0-1] verifying available flashback restore points
Aug 25 08:35:26 2014 [0-1] verifying DG Broker is disabled
Aug 25 08:35:26 2014 [0-1] looking up prior execution history
```

```
Aug 25 08:35:27 2014 [0-1] last completed stage [2-4] using script version
0001
Aug 25 08:35:27 2014 [0-1] resuming execution of script
```

Note: This script detects that the last completed stage was Stage 2-4 and resumes at the beginning of Stage 3.

```
### Stage 3: Validate upgraded transient logical standby
Aug 25 08:35:27 2014 [3-1] database c02orcl is no longer in OPEN MIGRATE mode
Aug 25 08:35:27 2014 [3-1] database c02orcl is at version 12.1.0.1.0
```

Note: Stage 3 performs validations to ensure that the standby database (`c02orcl`) is upgraded.

```
### Stage 4: Switch the transient logical standby to be the new primary
Aug 25 08:35:29 2014 [4-1] waiting for c02orcl to catch up (this could take a
while)
Aug 25 08:35:29 2014 [4-1] starting logical standby on database c02orcl
Aug 25 08:36:02 2014 [4-1] waiting for apply lag to fall under 30 seconds
Aug 25 08:37:09 2014 [4-1] apply lag measured at 66 seconds
Aug 25 08:37:19 2014 [4-1] apply lag measured at 10 seconds
Aug 25 08:37:20 2014 [4-2] switching c01orcl to become a logical standby
Aug 25 08:38:02 2014 [4-2] c01orcl is now a logical standby
Aug 25 08:38:03 2014 [4-3] waiting for standby c02orcl to process end-of-redo
from primary
Aug 25 08:38:05 2014 [4-4] switching c02orcl to become the new primary
Aug 25 08:38:55 2014 [4-4] c02orcl is now the new primary
```

Note: In Stage 4, the transient logical standby database is synchronized with the primary database. During this phase, the standby is updated with the transactions that were captured on the primary database, while the standby database was being upgraded. At the end of Stage 4, the databases switch roles, leaving the updated and upgraded former standby database (`c02orcl`) as the new primary database. The former primary database (`c01orcl`) now assumes the standby database role.

```
### Stage 5: Flashback former primary to pre-upgrade restore point and convert
to physical
Aug 25 08:39:06 2014 [5-1] verifying instance c01orcl1 is the only active
instance

WARN: c01orcl is a RAC database.  Before this script can continue, you
      must manually reduce the RAC to a single instance.  This can be
      accomplished with the following step:

        1) Shutdown all instances other than instance c01orcl1.
           eg: srvctl stop instance -d c01orcl -i c01orcl2 -o abort

      Once these steps have been performed, enter 'y' to continue the script.
      If desired, you may enter 'n' to exit the script to perform the required
```

```
      steps, and recall the script to resume from this point.

Are you ready to continue? (y/n):
```

Note: In Stage 5, the former primary database (`c01orcl`) flashed back to the preupgrade restore point so that it can be synchronized with the updated and upgraded former standby database (`c02orcl`), which is now a fully functioning primary database. However, before this can occur, `c01orcl` must be reduced to one instance.

98. Leave your other terminal sessions open and establish another terminal session connected to `c01n01` as the `oracle` user. Use this terminal session to shut down all instances of `c01orcl` except for `c01orcl1`; that is, shut down `c01orcl2`.

```
$ ssh oracle@c01n01
[oracle@c01n01 ~]$ . oraenv
ORACLE_SID = [oracle] ? c01orcl
The Oracle base has been set to /u01/app/oracle
[oracle@c01n01 ~]$ srvctl stop instance -d c01orcl -i c01orcl2 -o abort
[oracle@c01n01 ~]$ exit
logout
Connection to c01n01 closed.
$
```

99. Return to your `physru.sh` session and enter `y` to continue.

```
Are you ready to continue? (y/n): y

Aug 25 08:41:29 2014 [5-1] continuing
Aug 25 08:41:29 2014 [5-1] verifying instance c01orcl1 is the only active
instance
Aug 25 08:41:29 2014 [5-1] shutting down database c01orcl
Aug 25 08:41:49 2014 [5-1] mounting database c01orcl
Aug 25 08:42:11 2014 [5-2] flashing back database c01orcl to restore point
PRU_0000_0001
Aug 25 08:42:20 2014 [5-3] converting c01orcl into physical standby
Aug 25 08:42:22 2014 [5-4] shutting down database c01orcl

NOTE: Database c01orcl has been shutdown, and is now ready to be started
      using the newer version Oracle binary.  This script requires the
      database to be mounted (on all active instances, if RAC) before calling
      this script to resume the rolling upgrade.

NOTE: Database c01orcl is no longer limited to single instance operation since
      the database has been successfully converted into a physical standby.
      For increased availability, Oracle recommends starting all instances in
      the RAC on the newer binary by performing the following step:

        1) Startup and mount all instances for database c01orcl
        eg: srvctl start database -d c01orcl -o mount

[oracle@c01n01 ~]$
```

Note: At this point, the former primary database (`c01orcl`) has been flashed back to the preupgrade restore point. It has also been converted from a transient logical standby database to a physical standby database. Next, you will manually reconfigure `cluster01` so that `c01orcl` can be opened using Oracle Database 12*c* binaries, which have been preinstalled on the servers.

100. On every node in `cluster01`, copy the existing Oracle Database password files from the current location under the Oracle Database 11*g* home directory into the corresponding location under the Oracle Database 12*c* home directory.

```
[oracle@c01n01 ~]$ cp
/u01/app/oracle/product/11.2.0/dbhome_1/dbs/orapwc01orcl1
/u01/app/oracle/product/12.1.0/dbhome_1/dbs
[oracle@c01n01 ~]$ ssh c01n02 cp
/u01/app/oracle/product/11.2.0/dbhome_1/dbs/orapwc01orcl2
/u01/app/oracle/product/12.1.0/dbhome_1/dbs
```

101. On every node in `cluster01`, copy the existing database initialization files (`init.ora` files) from the current location under the Oracle Database 11*g* home directory into the corresponding location under the Oracle Database 12*c* home directory.

```
[oracle@c01n01 ~]$ cp
/u01/app/oracle/product/11.2.0/dbhome_1/dbs/initc01orcl1.ora
/u01/app/oracle/product/12.1.0/dbhome_1/dbs
[oracle@c01n01 ~]$ ssh c01n02 cp
/u01/app/oracle/product/11.2.0/dbhome_1/dbs/initc01orcl2.ora
/u01/app/oracle/product/12.1.0/dbhome_1/dbs
[oracle@c01n01 ~]$
```

102. On every node in `cluster01`, copy the existing `tnsnames.ora` files from the current location in the Oracle Database 11*g* home directory into the corresponding location in the Oracle Database 12*c* home directory.

```
[oracle@c01n01 ~]$ cp
/u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.o
ra /u01/app/oracle/product/12.1.0/dbhome_1/network/admin
[oracle@c01n01 ~]$ ssh c01n02 cp
/u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.o
ra /u01/app/oracle/product/12.1.0/dbhome_1/network/admin
[oracle@c01n01 ~]$
```

103. On every node in `cluster01`, execute the `setasmgid` utility so that Oracle Database 12*c* is able to access ASM.

```
[oracle@c01n01 ~]$ su -
Password: <oracle>
[root@c01n01 ~]# . oraenv
ORACLE_SID = [root] ? +ASM1
The Oracle base has been set to /u01/app/grid
[root@c01n01 ~]# setasmgid
o=/u01/app/oracle/product/12.1.0/dbhome_1/bin/oracle
[root@c01n01 ~]# ssh c01n02
```

```
[root@c01n02 ~]# . oraenv
ORACLE_SID = [root] ? +ASM2
The Oracle base has been set to /u01/app/grid
[root@c01n02 ~]# setasmgid
o=/u01/app/oracle/product/12.1.0/dbhome_1/bin/oracle
[root@c01n02 ~]# exit
logout
Connection to c01n02 closed.
[root@c01n01 ~]# exit
logout
[oracle@c01n01 ~]$
```

104. Remove the clusterware service entry for the `c01orcl` database.

```
[oracle@c01n01 ~]$ srvctl remove database -d c01orcl
Remove the database c01orcl? (y/[n]) y
[oracle@c01n01 ~]$
```

105. Refresh the environment settings in your `oracle@c01n01` terminal session to ensure that you use the upgraded Oracle Database 12*c* software from now on, as shown in the following:

```
[oracle@c01n01 ~]$ . oraenv
ORACLE_SID = [c01orcl1] ? c01orcl1
ORACLE_HOME = [/home/oracle] ?
/u01/app/oracle/product/12.1.0/dbhome_1
The Oracle base remains unchanged with value /u01/app/oracle
[oracle@c01n01 ~]$
```

106. Re-create the clusterware service entry for the `c01orcl` database and its instances.

```
[oracle@c01n01 ~]$ srvctl add database -d c01orcl -o
/u01/app/oracle/product/12.1.0/dbhome_1
[oracle@c01n01 ~]$ srvctl add instance -d c01orcl -i c01orcl1 -n
c01n01
[oracle@c01n01 ~]$ srvctl add instance -d c01orcl -i c01orcl2 -n
c01n02
[oracle@c01n01 ~]$
```

107. Start the `c01orcl` database in `mount` mode by using the upgraded Oracle Database 12*c* software.

```
[oracle@c01n01 ~]$ srvctl start database -d c01orcl -o mount
[oracle@c01n01 ~]$
```

108. Using your `grid@c01n01` terminal session, modify the Oracle Net Listener configuration file (`listener.ora`) on every node in `cluster01` so that the static registration entries associated with `c01orcl` are updated to reference the Oracle Database 12*c* home directory.

```
[grid@c01n01 ~]$ sed -i s#11.2#12.1#g
/u01/app/12.1.0/grid/network/admin/listener.ora
```

```
[grid@c01n01 ~]$ ssh c01n02 sed -i s#11.2#12.1#g
/u01/app/12.1.0/grid/network/admin/listener.ora
[grid@c01n01 ~]$
```

109. Stop and restart the Oracle Net listener across `cluster01`.

```
[grid@c01n01 ~]$ srvctl stop listener
[grid@c01n01 ~]$ srvctl start listener
[grid@c01n01 ~]$
```

110. Using your `oracle@c01n01` terminal session, restart `physru.sh` as shown below to continue the database rolling upgrade process. Enter `oracle_4U` when you are prompted for the `sysdba` password.

```
[oracle@c01n01 ~]$ ./physru.sh sys c01n01:1521/c01orcl_DGMGRL.example.com
c02n01:1521/c02orcl_DGMGRL.example.com c01orcl c02orcl 12.1.0.1.0
Please enter the sysdba password: <oracle_4U>

### Initialize script to either start over or resume execution
Aug 25 08:49:05 2014 [0-1] Identifying rdbms software version
Aug 25 08:50:51 2014 [0-1] database c01orcl is at version 12.1.0.1.0
Aug 25 08:50:53 2014 [0-1] database c02orcl is at version 12.1.0.1.0
Aug 25 08:51:02 2014 [0-1] verifying flashback database is enabled at c01orcl
and c02orcl
Aug 25 08:51:03 2014 [0-1] verifying available flashback restore points
Aug 25 08:51:15 2014 [0-1] verifying DG Broker is disabled
Aug 25 08:51:20 2014 [0-1] looking up prior execution history
Aug 25 08:51:28 2014 [0-1] last completed stage [5-4] using script version
0001
Aug 25 08:51:28 2014 [0-1] resuming execution of script
```

Note: This script detects that the last completed stage was Stage 5-4 and resumes at the beginning of Stage 6.

```
### Stage 6: Run media recovery through upgrade redo
Aug 25 08:51:44 2014 [6-1] upgrade redo region identified as scn range
[412934, 979273]
Aug 25 08:51:54 2014 [6-1] starting media recovery on c01orcl
Aug 25 08:53:36 2014 [6-1] confirming media recovery is running
Aug 25 08:53:45 2014 [6-1] waiting for media recovery to initialize
v$recovery_progress
Aug 25 08:55:39 2014 [6-1] monitoring media recovery's progress
Aug 25 08:55:42 2014 [6-2] last applied scn 400465 is approaching upgrade redo
start scn 412934
Aug 25 08:55:58 2014 [6-2] last applied scn 402153 is approaching upgrade redo
start scn 412934
Aug 25 08:56:34 2014 [6-3] recovery of upgrade redo at 05% - estimated
complete at Aug 25 09:12:39
Aug 25 08:56:50 2014 [6-3] recovery of upgrade redo at 06% - estimated
complete at Aug 25 09:12:35
Aug 25 08:57:26 2014 [6-3] recovery of upgrade redo at 07% - estimated
complete at Aug 25 09:17:30
Aug 25 08:58:01 2014 [6-3] recovery of upgrade redo at 10% - estimated
complete at Aug 25 09:18:00
```

```
Aug 25 08:58:18 2014 [6-3] recovery of upgrade redo at 11% - estimated
complete at Aug 25 09:18:33
Aug 25 08:58:52 2014 [6-3] recovery of upgrade redo at 14% - estimated
complete at Aug 25 09:17:04
Aug 25 08:59:08 2014 [6-3] recovery of upgrade redo at 17% - estimated
complete at Aug 25 09:15:50
Aug 25 08:59:26 2014 [6-3] recovery of upgrade redo at 22% - estimated
complete at Aug 25 09:11:49
Aug 25 08:59:44 2014 [6-3] recovery of upgrade redo at 26% - estimated
complete at Aug 25 09:10:51
Aug 25 09:00:01 2014 [6-3] recovery of upgrade redo at 28% - estimated
complete at Aug 25 09:11:02
Aug 25 09:00:18 2014 [6-3] recovery of upgrade redo at 32% - estimated
complete at Aug 25 09:09:57
Aug 25 09:00:35 2014 [6-3] recovery of upgrade redo at 33% - estimated
complete at Aug 25 09:10:09
Aug 25 09:00:52 2014 [6-3] recovery of upgrade redo at 38% - estimated
complete at Aug 25 09:09:12
Aug 25 09:01:13 2014 [6-3] recovery of upgrade redo at 39% - estimated
complete at Aug 25 09:09:33
Aug 25 09:01:57 2014 [6-3] recovery of upgrade redo at 40% - estimated
complete at Aug 25 09:10:25
Aug 25 09:02:16 2014 [6-3] recovery of upgrade redo at 41% - estimated
complete at Aug 25 09:10:53
Aug 25 09:02:33 2014 [6-3] recovery of upgrade redo at 42% - estimated
complete at Aug 25 09:11:19
Aug 25 09:02:50 2014 [6-3] recovery of upgrade redo at 44% - estimated
complete at Aug 25 09:11:26
Aug 25 09:03:06 2014 [6-3] recovery of upgrade redo at 46% - estimated
complete at Aug 25 09:11:13
Aug 25 09:03:23 2014 [6-3] recovery of upgrade redo at 48% - estimated
complete at Aug 25 09:11:07
Aug 25 09:03:40 2014 [6-3] recovery of upgrade redo at 51% - estimated
complete at Aug 25 09:10:56
Aug 25 09:03:58 2014 [6-3] recovery of upgrade redo at 55% - estimated
complete at Aug 25 09:10:23
Aug 25 09:04:14 2014 [6-3] recovery of upgrade redo at 63% - estimated
complete at Aug 25 09:09:03
Aug 25 09:04:31 2014 [6-3] recovery of upgrade redo at 66% - estimated
complete at Aug 25 09:08:49
Aug 25 09:04:48 2014 [6-3] recovery of upgrade redo at 67% - estimated
complete at Aug 25 09:09:04
Aug 25 09:05:06 2014 [6-3] recovery of upgrade redo at 68% - estimated
complete at Aug 25 09:09:13
Aug 25 09:05:23 2014 [6-3] recovery of upgrade redo at 69% - estimated
complete at Aug 25 09:09:24
Aug 25 09:05:40 2014 [6-3] recovery of upgrade redo at 72% - estimated
complete at Aug 25 09:09:22
Aug 25 09:05:57 2014 [6-3] recovery of upgrade redo at 74% - estimated
complete at Aug 25 09:09:20
Aug 25 09:06:13 2014 [6-3] recovery of upgrade redo at 76% - estimated
complete at Aug 25 09:09:26
Aug 25 09:06:31 2014 [6-3] recovery of upgrade redo at 78% - estimated
complete at Aug 25 09:09:25
Aug 25 09:06:48 2014 [6-3] recovery of upgrade redo at 90% - estimated
complete at Aug 25 09:07:53
```

```
Aug 25 09:07:13 2014 [6-3] recovery of upgrade redo at 94% - estimated
complete at Aug 25 09:07:54
Aug 25 09:07:30 2014 [6-3] recovery of upgrade redo at 95% - estimated
complete at Aug 25 09:07:58
Aug 25 09:07:47 2014 [6-3] recovery of upgrade redo at 97% - estimated
complete at Aug 25 09:08:06
Aug 25 09:08:04 2014 [6-3] recovery of upgrade redo at 98% - estimated
complete at Aug 25 09:08:13
Aug 25 09:08:21 2014 [6-4] media recovery has finished recovering through
upgrade
```

Note: In Stage 6, media recovery is performed on the former primary database (`c01orcl`). This operation upgrades the database and applies all transactions on user data up to the current time. At the end of Stage 6 both databases are fully upgraded.

```
### Stage 7: Switch back to the original roles prior to the rolling upgrade

NOTE: At this point, you have the option to perform a switchover
     which will restore c01orcl back to a primary database and
     c02orcl back to a physical standby database.  If you answer 'n'
     to the question below, c01orcl will remain a physical standby
     database and c02orcl will remain a primary database.

Do you want to perform a switchover? (y/n):
```

Note: Stage 7 performs a role switch so that the databases are returned to their original roles. This stage is optional and can be skipped; however, customers often perform this stage.

111. Enter `y` to switch the databases back to their original roles.

```
### Stage 7: Switch back to the original roles prior to the rolling upgrade

NOTE: At this point, you have the option to perform a switchover
     which will restore c01orcl back to a primary database and
     c02orcl back to a physical standby database.  If you answer 'n'
     to the question below, c01orcl will remain a physical standby
     database and c02orcl will remain a primary database.

Do you want to perform a switchover? (y/n): y

Aug 25 09:10:30 2014 [7-1] continuing
Aug 25 09:10:30 2014 [7-2] verifying instance c02orcl1 is the only active
instance

WARN: c02orcl is a RAC database.  Before this script can continue, you
      must manually reduce the RAC to a single instance.  This can be
      accomplished with the following step:

        1) Shutdown all instances other than instance c02orcl1.
           eg: srvctl stop instance -d c02orcl -i c02orcl2

      Once these steps have been performed, enter 'y' to continue the script.
```

```
      If desired, you may enter 'n' to exit the script to perform the required
      steps, and recall the script to resume from this point.

Are you ready to continue? (y/n):
```

112. To perform the final switchover, the `physru.sh` script requires `c02orcl` to be reduced to one instance. Use your `oracle@c02n01` terminal session to shut down all instances of `c02orcl` except for `c02orcl1`; that is, shut down `c02orcl2`.

```
[oracle@c02n01 ~]$ srvctl stop instance -d c02orcl -i c02orcl2
[oracle@c02n01 ~]$
```

113. Return to your `physru.sh` session and enter `y` to continue.

```
Are you ready to continue? (y/n): y

Aug 25 09:12:28 2014 [7-2] continuing
Aug 25 09:12:28 2014 [7-2] verifying instance c02orcl1 is the only active
instance
Aug 25 09:12:30 2014 [7-2] waiting for apply lag to fall under 30 seconds
Aug 25 09:12:37 2014 [7-2] apply lag measured at 7 seconds
Aug 25 09:12:41 2014 [7-3] switching c02orcl to become a physical standby
Aug 25 09:12:56 2014 [7-3] c02orcl is now a physical standby
Aug 25 09:12:56 2014 [7-3] shutting down database c02orcl
Aug 25 09:12:56 2014 [7-3] mounting database c02orcl
Aug 25 09:13:17 2014 [7-4] waiting for standby c01orcl to process end-of-redo
from primary
Aug 25 09:13:19 2014 [7-5] switching c01orcl to become the new primary
Aug 25 09:13:21 2014 [7-5] c01orcl is now the new primary
Aug 25 09:13:21 2014 [7-5] opening database c01orcl
Aug 25 09:13:56 2014 [7-6] starting media recovery on c02orcl
Aug 25 09:14:05 2014 [7-6] confirming media recovery is running

NOTE: Database c01orcl has completed the switchover to the primary role, but
      instance c01orcl1 is the only open instance.  For increased
availability,
      Oracle recommends opening the remaining active instances which are
      currently in mounted mode by performing the following steps:

        1) Shutdown all instances other than instance c01orcl1.
        eg: srvctl stop instance -d c01orcl -i c01orcl2

        2) Startup and open all inactive instances for database c01orcl.
        eg: srvctl start database -d c01orcl

NOTE: Database c02orcl is no longer limited to single instance operation since
      it has completed the switchover to the physical standby role.  For
      increased  availability, Oracle recommends starting the inactive
      instances in the RAC by performing the following step:

        1) Startup and mount inactive instances for database c02orcl
        eg: srvctl start database -d c02orcl -o mount
```

Note: At the end of Stage 7, the databases are returned to their original roles. That is, `c01orcl` is the primary database and `c02orcl` is the physical standby database. However, manual intervention is required to restart all the Oracle RAC database instances. The script output contains instructions that include the required commands.

```
### Stage 8: Statistics
script start time:                                  25-Aug-14
06:16:07
script finish time:                                 25-Aug-14
09:14:41
total script execution time:                              +00
02:58:34
wait time for user upgrade:                               +00
02:03:06
active script execution time:                             +00
00:55:28
transient logical creation start time:              25-Aug-14
06:22:46
transient logical creation finish time:             25-Aug-14
06:27:43
primary to logical switchover start time:           25-Aug-14
08:37:19
logical to primary switchover finish time:          25-Aug-14
08:39:03
primary services offline for:                             +00
00:01:44
total time former primary in physical role:               +00
00:27:54
time to reach upgrade redo:                               +00
00:00:34
time to recover upgrade redo:                             +00
00:12:06
primary to physical switchover start time:          25-Aug-14
09:10:30
physical to primary switchover finish time:         25-Aug-14
09:13:54
primary services offline for:                             +00
00:03:24

SUCCESS: The physical rolling upgrade is complete

[oracle@c01n01 ~]$
```

Note: Stage 8 outputs statistics for the entire rolling database upgrade process. You can use this information to see how long the entire process took and how long the primary database was unavailable. In the example that forms the basis for this practice, the rolling database upgrade process spanned almost three hours. However, during that time, primary database services were unavailable for a total of 5 minutes and 8 seconds. By using additional custom scripting to automate some of the critical manual tasks, you can expect to reduce primary database down time even further.

Part 5: Postupgrade Steps

At this point, your environment is fully upgraded. However, there are a few more operations required to make it fully operational and to leverage the full power of Oracle Database 12*c*. In the final part of this practice, you will perform the following tasks:

- Start all the Oracle RAC database instances.
- Re-enable the Oracle Data Guard Broker.
- Raise the `compatible` database instance parameter to enable all the Oracle Database 12*c* new features.
- Confirm that the user data changes made during the upgrade process made it through.

114. Confirm the status of `c01orcl` and reconfigure the database so that all the instances are running and open.

```
[oracle@c01n01 ~]$ srvctl status database -d c01orcl -v
Instance c01orcl1 is running on node c01n01. Instance status: Open.
Instance c01orcl2 is running on node c01n02. Instance status: Mounted
(Closed).
[oracle@c01n01 ~]$ srvctl stop instance -d c01orcl -i c01orcl2
[oracle@c01n01 ~]$ srvctl start database -d c01orcl
[oracle@c01n01 ~]$ srvctl status database -d c01orcl -v
Instance c01orcl1 is running on node c01n01. Instance status: Open.
Instance c01orcl2 is running on node c01n02. Instance status: Open.
[oracle@c01n01 ~]$
```

115. Using your `oracle@c02n01` terminal session, confirm the status of `c02orcl` and reconfigure the database so that all the instances are running and mounted.

```
[oracle@c02n01 ~]$ srvctl status database -d c02orcl -v
Instance c02orcl1 is running on node c02n01. Instance status: Mounted
(Closed).
Instance c02orcl2 is not running on node c02n02
[oracle@c02n01 ~]$ srvctl start database -d c02orcl -o mount
[oracle@c02n01 ~]$ srvctl status database -d c02orcl -v
Instance c02orcl1 is running on node c02n01. Instance status: Mounted
(Closed).
Instance c02orcl2 is running on node c02n02. Instance status: Mounted
(Closed).
[oracle@c02n01 ~]$
```

116. Re-enable Oracle Data Guard Broker on `c02orcl`.

```
[oracle@c02n01 ~]$ sqlplus / as sysdba

SQL*Plus: Release 12.1.0.1.0 Production ...

SQL> alter system set dg_broker_start=true scope=both;

System altered.

SQL> exit
Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 ...
[oracle@c02n01 ~]$
```

117. Using your `oracle@c01n01` terminal session, re-enable Oracle Data Guard Broker on `c01orcl`.

```
[oracle@c01n01 ~]$ sqlplus / as sysdba

SQL*Plus: Release 12.1.0.1.0 Production ...

SQL> alter system set dg_broker_start=true scope=both;

System altered.

SQL> exit
Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 ...
[oracle@c01n01 ~]$
```

118. Use the Data Guard Broker management utility (`dgmgrl`) to re-enable the broker configuration. Then, examine the configuration and the databases to confirm that everything is functioning as expected.

```
[oracle@c01n01 ~]$ dgmgrl sys/oracle_4U
DGMGRL for Linux: Version 12.1.0.1.0 - 64bit Production

Copyright (c) 2000, 2012, Oracle. All rights reserved.

Welcome to DGMGRL, type "help" for information.
Connected as SYSDG.
DGMGRL> enable configuration
Enabled.
DGMGRL> show configuration

Configuration - dg_config

  Protection Mode: MaxPerformance
  Databases:
  c01orcl - Primary database
    c02orcl - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS

DGMGRL> show database c01orcl

Database - c01orcl
```

```
  Role:               PRIMARY
  Intended State:     TRANSPORT-ON
  Instance(s):
    c01orcl1
    c01orcl2

Database Status:
SUCCESS

DGMGRL> show database c02orcl

Database - c02orcl

  Role:               PHYSICAL STANDBY
  Intended State:     APPLY-ON
  Transport Lag:      0 seconds (computed 1 second ago)
  Apply Lag:          0 seconds (computed 1 second ago)
  Apply Rate:         63.00 KByte/s
  Real Time Query:    OFF
  Instance(s):
    c02orcl1 (apply instance)
    c02orcl2

Database Status:
SUCCESS

DGMGRL> exit
[oracle@c01n01 ~]$
```

Notes

- Next, you will raise the `compatible` database instance parameter to enable all the Oracle Database 12*c* new features. Oracle recommends increasing the `compatible` parameter only after thorough testing of the upgraded database has been performed. This is because the database cannot subsequently be downgraded to releases earlier than what is set for compatibility.
- The flashback database feature cannot be used to return a database back to a point prior to raising the `compatible` parameter. Because of this, you must first remove all guaranteed restore points before raising the value of the `compatible` parameter.
- In a Data Guard Environment, you must raise the `compatible` parameter on the standby database before raising it on the primary database.

119. On your standby database (`c02orcl`), use SQL*Plus to perform the following tasks:
    - Find the name of any remaining guaranteed restore points. You should expect that the `physru.sh` script has already removed the guaranteed restore points that it created; however, you should expect to see one remaining guaranteed restore point that was created by the Database Upgrade Assistant (DBUA).
    - Drop any remaining guaranteed restore points.
    - Confirm the current `compatible` setting.
    - Set `compatible='12.1.0.1.0'` inside the database server parameter file.

```
[oracle@c02n01 ~]$ sqlplus / as sysdba

SQL*Plus: Release 12.1.0.1.0 Production ...

SQL> select name from v$restore_point;

NAME
------------------------------------------------------------------
GRP_1408952037429

SQL> drop restore point GRP_1408952037429;

Restore point dropped.

SQL> show parameter compatible

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------
compatible                           string      11.2.0.4.0
noncdb_compatible                    boolean     FALSE
SQL> alter system set compatible='12.1.0.1.0' scope=spfile;

System altered.

SQL> exit
Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 ...
[oracle@c02n01 ~]$
```

120. Stop and restart the `c02orcl` database in mount mode. Then confirm that the `compatible` value is now `12.1.0.1.0`.

```
[oracle@c02n01 ~]$ srvctl stop database -d c02orcl
[oracle@c02n01 ~]$ srvctl start database -d c02orcl -o mount
[oracle@c02n01 ~]$ sqlplus / as sysdba
```

```
SQL*Plus: Release 12.1.0.1.0 Production ...

SQL> show parameter compatible

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------
compatible                           string      12.1.0.1.0
noncdb_compatible                    boolean     FALSE
SQL> exit
Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 ...
[oracle@c02n01 ~]$
```

121. Use your `oracle@c01n01` terminal session to raise the compatible parameter on the `c01orcl` database. Notice that there should be no guaranteed restore points to drop on this database because DBUA was never run on the database.

```
[oracle@c01n01 ~]$ sqlplus / as sysdba

SQL*Plus: Release 12.1.0.1.0 Production ...

SQL> select name from v$restore_point;

no rows selected

SQL> show parameter compatible

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------
compatible                           string      11.2.0.4.0
noncdb_compatible                    boolean     FALSE
SQL> alter system set compatible='12.1.0.1.0' scope=spfile;

System altered.

SQL> exit
Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 ...
[oracle@c01n01 ~]$
```

122. Stop and restart the `c01orcl` database. Then confirm that the `compatible` value is now `12.1.0.1.0`.

```
[oracle@c01n01 ~]$ srvctl stop database -d c01orcl
[oracle@c01n01 ~]$ srvctl start database -d c01orcl
[oracle@c01n01 ~]$ sqlplus / as sysdba
```

```
SQL*Plus: Release 12.1.0.1.0 Production ...

SQL> show parameter compatible

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------
compatible                           string      12.1.0.1.0
noncdb_compatible                    boolean     FALSE
SQL> exit
Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.1.0
[oracle@c01n01 ~]$
```

123. Finally, connect to your primary database (`c01orcl`) and query the `hr.regions` table to confirm that the changes you made during the upgrade process (back in step 89) have survived.

```
[oracle@c01n01 ~]$ sqlplus hr/hr

SQL*Plus: Release 12.1.0.1.0 Production ...

SQL> select * from regions;

 REGION_ID REGION_NAME
---------- -------------------------
         5 Australia
         1 Europe
         2 Americas
         3 Asia
         4 Middle East and Africa

SQL> exit
Disconnected from Oracle Database 12c Enterprise Edition Release
12.1.0.1.0 ...
[oracle@c01n01 ~]$
```

124. Close all of your terminal sessions.

Congratulations! You have performed a rolling upgrade from Oracle Database release 11.2.0.4 to Oracle Database 12.1.0.1 by using Oracle RAC and the Oracle Data Guard transient logical standby feature to maximize the availability of the primary database during the upgrade process.

Practice 1-2: Environment Reconfiguration

Overview

In this practice, you will reconfigure your practice environment in preparation for future practices.

Tasks

1. Establish a terminal session on your practice environment. Do not connect to any of the virtual machines.
2. Execute the following command to perform a series of reconfiguration steps in preparation for the next set of practices. Note that the reconfiguration process may run for more than 30 minutes. Ensure that you allow the reconfiguration process to complete and check to make sure that the VM status output includes entries for `c03n01`, `c03n02`, `c03n03`, and `c03n04`.

```
$ sudo /OVS/seed_pool/setup_Cluster03.sh
Removing current VMs...
Extracting cluster03 VMs...
Starting VMs...
VM status...
Name                                ID    Mem VCPUs      State
Time(s)
Domain-0                             0   1024     2     r-----   5275.8
c03n01                               5   5000     1     -b----     13.4
c03n02                               6   5000     1     -b----      8.5
c03n03                               7   1500     1     -b----      6.3
c03n04                               8   1500     1     -b----      4.6
$
```

Practices for Lesson 2: ASM Filter Driver

Chapter 2

Practices for Lesson 2: Overview

Practice Overview

In this practice, you will configure and use Oracle ASM Filter Driver.

Practice 2-1: Configuring and Using ASM Filter Driver

Overview

In this practice, you will configure and use the new Oracle ASM Filter Driver. You will see how to configure ASM Filter Driver on a new cluster running Oracle Grid Infrastructure version 12.1.0.2, and you will perform tests to demonstrate how ASM Filter Driver protects data integrity by preventing unauthorized changes to the disk devices that it manages.

Tasks

Part 1: Configuring ASM Filter Driver

By default, ASM Filter Driver is not configured on a new or an upgraded cluster. In the first part of this practice, you will configure ASM Filter Driver in a rolling manner on a two-node cluster running Oracle Grid Infrastructure version 12.1.0.2.

1. Establish a terminal session connected to `c03n01` as the `grid` user and configure the terminal environment as shown in the following:

```
$ ssh grid@c03n01
[grid@c03n01 ~]$ . oraenv
ORACLE_SID = [grid] ? +ASM1
The Oracle base has been set to /u01/app/grid
[grid@c03n01 ~]$
```

2. ASM Filter Driver should be configured on all the hub nodes in a cluster. Examine your cluster and take note of the hub nodes in the cluster.

```
[grid@c03n01 ~]$ olsnodes -a
c03n01  Hub
c03n02  Hub
[grid@c03n01 ~]$
```

3. Examine the ASM environment and take note of the ASM disk string setting.

```
[grid@c03n01 ~]$ asmcmd dsget
parameter:/dev/xvd*
profile:/dev/xvd*
[grid@c03n01 ~]$
```

4. Add `AFD:*` to the ASM disk string to enable future discovery of disks controlled by ASM Filter Driver. Confirm the setting afterwards.

```
[grid@c03n01 ~]$ asmcmd dsset '/dev/xvd*','AFD:*'
[grid@c03n01 ~]$ asmcmd dsget
parameter:/dev/xvd*, AFD:*
profile:/dev/xvd*,AFD:*
[grid@c03n01 ~]$
```

5. ASM Filter Driver can be configured in a rolling fashion, that is, one node at a time. To configure ASM Filter Driver on each node, first stop Oracle Clusterware.

```
[grid@c03n01 ~]$ su -c "crsctl stop crs"
Password: <oracle>
CRS-2791: Starting shutdown of Oracle High Availability
Services-managed resources on 'c03n01'
CRS-2673: Attempting to stop 'ora.crsd' on 'c03n01'
CRS-2790: Starting shutdown of Cluster Ready Services-managed
resources on 'c03n01'
CRS-2673: Attempting to stop 'ora.c03orcl.db' on 'c03n01'
...
CRS-2673: Attempting to stop 'ora.gipcd' on 'c03n01'
CRS-2677: Stop of 'ora.gipcd' on 'c03n01' succeeded
CRS-2793: Shutdown of Oracle High Availability Services-managed
resources on 'c03n01' has completed
CRS-4133: Oracle High Availability Services has been stopped.
[grid@c03n01 ~]$
```

6. After Oracle Clusterware is stopped, execute the ASM Filter Driver configuration command. Notice that the command must be executed as the `root` user.

```
[grid@c03n01 ~]$ su -c "asmcmd afd_configure"
Password: <oracle>
Connected to an idle instance.
AFD-627: AFD distribution files found.
AFD-636: Installing requested AFD software.
AFD-637: Loading installed AFD drivers.
AFD-9321: Creating udev for AFD.
AFD-9323: Creating module dependencies - this may take some
time.
AFD-9154: Loading 'oracleafd.ko' driver.
AFD-649: Verifying AFD devices.
AFD-9156: Detecting control device '/dev/oracleafd/admin'.
AFD-638: AFD installation correctness verified.
Modifying resource dependencies - this may take some time.
[grid@c03n01 ~]$
```

7. Confirm the configuration and ensure that the ASM Filter Driver state is reported as `LOADED`.

```
[grid@c03n01 ~]$ asmcmd afd_state
Connected to an idle instance.
ASMCMD-9526: The AFD state is 'LOADED' and filtering is
'DEFAULT' on host 'c03n01.example.com'
[grid@c03n01 ~]$
```

8. Restart Oracle Clusterware.

```
[grid@c03n01 ~]$ su -c "crsctl start crs -wait"
Password: <oracle>
CRS-4123: Starting Oracle High Availability Services-managed
resources
CRS-2672: Attempting to start 'ora.mdnsd' on 'c03n01'
CRS-2672: Attempting to start 'ora.evmd' on 'c03n01'
...
CRS-2672: Attempting to start 'ora.c03orcl.db' on 'c03n01'
CRS-2676: Start of 'ora.c03orcl.db' on 'c03n01' succeeded
CRS-6016: Resource auto-start has completed for server c03n01
CRS-6024: Completed start of Oracle Cluster Ready Services-
managed resources
CRS-4123: Oracle High Availability Services has been started.
[grid@c03n01 ~]$
```

9. Examine the ASM Filter Driver disk string setting. You will find that the ASM Filter Driver disk string is initially empty.

   Note: Do not confuse the ASM disk string with the ASM Filter Driver disk string.

```
[grid@c03n01 ~]$ asmcmd afd_dsget
AFD discovery string:
[grid@c03n01 ~]$
```

10. To enable future disk discovery by ASM Filter Driver, set the ASM Filter Driver disk string to the original ASM disk string setting, which you examined in step 3.

    Note: Ensure that you are acting as the `grid` user (not the `root` user) when you set the ASM Filter Driver disk string.

```
[grid@c03n01 ~]$ asmcmd afd_dsset '/dev/xvd*'
[grid@c03n01 ~]$ asmcmd afd_dsget
AFD discovery string: '/dev/xvd*'
[grid@c03n01 ~]$
```

11. Repeat the configuration steps 5 through 10 on every other hub node in the cluster (`c03n02` in this example).

    Notes

    - Ensure that you configure all your nodes consistently because any variation between nodes could cause problems later.
    - The ASM Filter Driver disk string must be configured separately on every node. This differs from the ASM disk string, which can be set on one node and is automatically propagated throughout the cluster.

```
[grid@c03n01 ~]$ ssh c03n02
[grid@c03n02 ~]$ . oraenv
ORACLE_SID = [grid] ? +ASM2
The Oracle base has been set to /u01/app/grid
[grid@c03n02 ~]$ su -c "crsctl stop crs"
Password: <oracle>
```

```
CRS-2791: Starting shutdown of Oracle High Availability
Services-managed resources on 'c03n02'
CRS-2673: Attempting to stop 'ora.crsd' on 'c03n02'
CRS-2790: Starting shutdown of Cluster Ready Services-managed
resources on 'c03n02'
CRS-2673: Attempting to stop 'ora.cvu' on 'c03n02'
CRS-2673: Attempting to stop 'ora.c03orcl.db' on 'c03n02'
...
CRS-2673: Attempting to stop 'ora.gipcd' on 'c03n02'
CRS-2677: Stop of 'ora.gipcd' on 'c03n02' succeeded
CRS-2793: Shutdown of Oracle High Availability Services-managed
resources on 'c03n02' has completed
CRS-4133: Oracle High Availability Services has been stopped.
[grid@c03n02 ~]$ su -c "asmcmd afd_configure"
Password: <oracle>
Connected to an idle instance.
AFD-627: AFD distribution files found.
AFD-636: Installing requested AFD software.
AFD-637: Loading installed AFD drivers.
AFD-9321: Creating udev for AFD.
AFD-9323: Creating module dependencies - this may take some
time.
AFD-9154: Loading 'oracleafd.ko' driver.
AFD-649: Verifying AFD devices.
AFD-9156: Detecting control device '/dev/oracleafd/admin'.
AFD-638: AFD installation correctness verified.
Modifying resource dependencies - this may take some time.
[grid@c03n02 ~]# asmcmd afd_state
Connected to an idle instance.
ASMCMD-9526: The AFD state is 'LOADED' and filtering is
'DEFAULT' on host 'c03n02.example.com'
[grid@c03n02 ~]$ su -c "crsctl start crs -wait"
Password: <oracle>
CRS-4123: Starting Oracle High Availability Services-managed
resources
CRS-2672: Attempting to start 'ora.mdnsd' on 'c03n02'
CRS-2672: Attempting to start 'ora.evmd' on 'c03n02'
...
CRS-2672: Attempting to start 'ora.c03orcl.db' on 'c03n02'
CRS-2676: Start of 'ora.c03orcl.db' on 'c03n02' succeeded
CRS-6016: Resource auto-start has completed for server c03n02
CRS-6024: Completed start of Oracle Cluster Ready Services-
managed resources
CRS-4123: Oracle High Availability Services has been started.
```

```
[grid@c03n02 ~]$ asmcmd afd_dsset '/dev/xvd*'
[grid@c03n02 ~]$ asmcmd afd_dsget
AFD discovery string: '/dev/xvd*'
[grid@c03n02 ~]$ exit
logout
Connection to c03n02 closed.
[grid@c03n01 ~]$
```

Part 2: Enabling ASM Filter Driver on an ASM Disk Group Without Clusterware Files

After you have configured ASM Filter Driver, the next step is to enable disk devices to use it. Typically, you will enable all the devices associated with a disk group at the same time. The required process differs depending on whether or not the disk group contains the Oracle Cluster Registry (OCR) or voting files. In the next part of this practice, you will enable ASM Filter Driver for all the disks in the FRA disk group, which does not contain the OCR or voting files.

12. Return to your grid terminal session on c03n01 and examine your cluster to confirm the disk group that contains the OCR and voting files (DATA in this example).

```
[grid@c03n01 ~]$ ocrcheck -config
Oracle Cluster Registry configuration is :
         Device/File Name         :      +DATA
[grid@c03n01 ~]$ crsctl query css votedisk
##  STATE    File Universal Id               File Name Disk group
--  -----    -----------------               --------- ---------
 1. ONLINE   61a73f34e7aa4f53bfa2471795b913bb (/dev/xvdf1) [DATA]
 2. ONLINE   8c15638681224faebf10c850d66a8270 (/dev/xvdf2) [DATA]
 3. ONLINE   5ecca473038b4f7dbf7a4b0233922003 (/dev/xvdf3) [DATA]
Located 3 voting disk(s).
[grid@c03n01 ~]$
```

13. Examine your ASM configuration and take note of the disks groups that do not contain the OCR and voting files. Select a disk group to be the first candidate for ASM Filter Driver (FRA in this example).

```
[grid@c03n01 ~]$ asmcmd lsdg
State    Type    Rebal  Sector  Block       AU  Total_MB  Free_MB
Req_mir_free_MB  Usable_file_MB  Offline_disks  Voting_files  Name
MOUNTED  NORMAL  N         512   4096  1048576     34434    17593
1913            7840               0             Y  DATA/
MOUNTED  NORMAL  N         512   4096  1048576      3826     3634
0               1817                0             N  FRA/
[grid@c03n01 ~]$
```

14. Examine the candidate disk group and take note of the disk devices that it contains.

```
[grid@c03n01 ~]$ asmcmd lsdsk -G FRA
Path
/dev/xvdg10
/dev/xvdg11
[grid@c03n01 ~]$
```

15. Stop the candidate disk group.

    Note: In this example, the disk group is forcibly stopped; however, you could choose to shut down the dependent resources first and stop the disk group without using the force option.

```
[grid@c03n01 ~]$ srvctl stop diskgroup -diskgroup FRA -f
[grid@c03n01 ~]$
```

16. Apply an ASM Filter Driver label to the disk devices associated with the candidate disk group.

    Notes

    - A label name (`XVDG10`, `XVDG11`) is applied to each disk, and the administrator is free to specify the label of his or her choice.
    - Because these disks already belong to an existing disk group, the `--migrate` option must be specified. This option is not required when disks are labeled before use by ASM.

```
[grid@c03n01 ~]$ asmcmd afd_label XVDG10 /dev/xvdg10 --migrate
[grid@c03n01 ~]$ asmcmd afd_label XVDG11 /dev/xvdg11 --migrate
[grid@c03n01 ~]$
```

17. Verify that all the disks are labeled as expected.

```
[grid@c03n01 ~]$ asmcmd afd_lsdsk
--------------------------------------------------------------------
Label                     Filtering   Path
====================================================================
XVDG10                      ENABLED   /dev/xvdg10
XVDG11                      ENABLED   /dev/xvdg11
[grid@c03n01 ~]$
```

    At this point, the disks in the candidate disk group are labeled and visible to the node where the labeling occurred. However, the label information is not automatically propagated throughout the cluster, and this must occur on every node that wishes to mount the disk group.

18. Leave your current terminal session in the background and establish a new terminal session connected to `c03n02` as the `grid` user and configure the terminal environment as shown in the following:

```
$ ssh grid@c03n02
[grid@c03n02 ~]$ . oraenv
ORACLE_SID = [grid] ? +ASM2
The Oracle base has been set to /u01/app/grid
[grid@c03n02 ~]$
```

19. Confirm that there are currently no labeled devices visible on this node.

```
[grid@c03n02 ~]$ asmcmd afd_lsdsk
There are no labelled devices.
[grid@c03n02 ~]$
```

20. Perform an ASM Filter Driver disk scan. This operation scans all the disks in the ASM Filter Driver disk string path and discovers all the labels. Afterwards, confirm the visibility of the labels.

    Note: The ASM Filter Driver disk scan must be performed on every node that wants to mount the disk group.

```
[grid@c03n02 ~]$ asmcmd afd_scan
[grid@c03n02 ~]$ asmcmd afd_lsdsk
--------------------------------------------------------------------
Label                      Filtering   Path
====================================================================
XVDG10                       ENABLED   /dev/xvdg10
XVDG11                       ENABLED   /dev/xvdg11
[grid@c03n02 ~]$
```

21. After the labels are visible on all the required nodes, restart the disk group and confirm that it is running.

```
[grid@c03n02 ~]$ srvctl start diskgroup -diskgroup FRA
[grid@c03n02 ~]$ srvctl status diskgroup -diskgroup FRA
Disk Group FRA is running on c03n01,c03n02
[grid@c03n02 ~]$
```

    Now ASM Filter Driver is enabled on the candidate disk group, and the above procedure can be repeated on other disk groups that do not contain Oracle Clusterware files. In the next part of this practice, you will attempt to intentionally corrupt an ASM disk protected by ASM Filter Driver and see what happens.

22. As the `root` user, execute the following `dd` command, which attempts to write a random block of data on to one of the ASM Filter Driver–enabled disks. Notice that the write attempt fails and an I/O error is reported.

```
[grid@c03n02 ~]$ su -c "dd if=/dev/random of=/dev/xvdg11 bs=4096
count=1 oflag=sync"
Password: <oracle>
dd: writing `/dev/xvdg11': Input/output error
0+1 records in
0+0 records out
0 bytes (0 B) copied, 0.00145229 s, 0.0 kB/s
[grid@c03n02 ~]$
```

23. View the tail end of the system log at `/var/log/messages`. Notice the messages indicating that ASM Filter Driver rejected and unauthorized the write attempt. Notice also that the error messages identify the process responsible for the unauthorized write attempt (`dd[8628]`) along with the relevant disk device (`xvdg11`). Finally, exit the `root` shell and return to your `grid` user session.

```
[grid@c03n02 ~]$ su -c "tail /var/log/messages"
Password: <oracle>
...
Aug 27 22:30:31 c03n02 ntpd[1724]: Listening on interface #16
eth0:2, 192.0.2.139#123 Enabled
Aug 27 22:39:24 c03n02 kernel: F 4295892.228/140827223924
dd[15015]  afd_mkrequest_fn: write IO on ASM managed device
(major=202/minor=107)  not supported i=2 start=35278803 seccnt=1
pstart=35278803  pend=39198600
Aug 27 22:39:24 c03n02 kernel: Buffer I/O error on device
xvdg11, logical block 0
Aug 27 22:39:24 c03n02 kernel: lost page write due to I/O error
on xvdg11
[grid@c03n02 ~]$
```

Part 3: Enabling ASM Filter Driver on an ASM Disk Group with Clusterware Files

In the final part of this practice, you will enable ASM Filter Driver for all the disks in the `DATA` disk group, which contains the OCR and voting files.

24. Return to your `grid` terminal session on `c03n01` and examine the disk group containing the Oracle Clusterware files (`DATA` in this example). Take note of the disk devices that it contains.

```
[grid@c03n01 ~]$ asmcmd lsdsk -G DATA
Path
/dev/xvdf1
/dev/xvdf10
/dev/xvdf11
/dev/xvdf2
/dev/xvdf3
/dev/xvdf5
/dev/xvdf6
/dev/xvdf7
/dev/xvdf8
/dev/xvdf9
/dev/xvdg1
/dev/xvdg2
/dev/xvdg3
/dev/xvdg5
/dev/xvdg6
/dev/xvdg7
```

```
/dev/xvdg8
/dev/xvdg9
[grid@c03n01 ~]$
```

25. Stop the entire cluster.

    Note: This is an unavoidable requirement for enabling ASM Filter Driver on a disk group that contains Oracle Clusterware files.

```
[grid@c03n01 ~]$ su -c "crsctl stop cluster -all"
Password: <oracle>
CRS-2673: Attempting to stop 'ora.crsd' on 'c03n01'
CRS-2790: Starting shutdown of Cluster Ready Services-managed
resources on 'c03n01'
CRS-2673: Attempting to stop 'ora.LISTENER_SCAN1.lsnr' on
'c03n01'
CRS-2673: Attempting to stop 'ora.c03orcl.db' on 'c03n01'
...
CRS-2673: Attempting to stop 'ora.cluster_interconnect.haip' on
'c03n02'
CRS-2677: Stop of 'ora.cluster_interconnect.haip' on 'c03n02'
succeeded
CRS-2673: Attempting to stop 'ora.cssd' on 'c03n02'
CRS-2677: Stop of 'ora.cssd' on 'c03n02' succeeded
[grid@c03n01 ~]$
```

26. Apply an ASM Filter Driver label to all the disk devices associated with the disk group.

```
[grid@c03n01 ~]$ asmcmd afd_label XVDF1 /dev/xvdf1 --migrate
Connected to an idle instance.
[grid@c03n01 ~]$ asmcmd afd_label XVDF10 /dev/xvdf10 --migrate
Connected to an idle instance.
[grid@c03n01 ~]$ asmcmd afd_label XVDF11 /dev/xvdf11 --migrate
Connected to an idle instance.
[grid@c03n01 ~]$ asmcmd afd_label XVDF2 /dev/xvdf2 --migrate
Connected to an idle instance.
[grid@c03n01 ~]$ asmcmd afd_label XVDF3 /dev/xvdf3 --migrate
Connected to an idle instance.
[grid@c03n01 ~]$ asmcmd afd_label XVDF5 /dev/xvdf5 --migrate
Connected to an idle instance.
[grid@c03n01 ~]$ asmcmd afd_label XVDF6 /dev/xvdf6 --migrate
Connected to an idle instance.
[grid@c03n01 ~]$ asmcmd afd_label XVDF7 /dev/xvdf7 --migrate
Connected to an idle instance.
[grid@c03n01 ~]$ asmcmd afd_label XVDF8 /dev/xvdf8 --migrate
Connected to an idle instance.
[grid@c03n01 ~]$ asmcmd afd_label XVDF9 /dev/xvdf9 --migrate
Connected to an idle instance.
```

```
[grid@c03n01 ~]$ asmcmd afd_label XVDG1 /dev/xvdg1 --migrate
Connected to an idle instance.
[grid@c03n01 ~]$ asmcmd afd_label XVDG2 /dev/xvdg2 --migrate
Connected to an idle instance.
[grid@c03n01 ~]$ asmcmd afd_label XVDG3 /dev/xvdg3 --migrate
Connected to an idle instance.
[grid@c03n01 ~]$ asmcmd afd_label XVDG5 /dev/xvdg5 --migrate
Connected to an idle instance.
[grid@c03n01 ~]$ asmcmd afd_label XVDG6 /dev/xvdg6 --migrate
Connected to an idle instance.
[grid@c03n01 ~]$ asmcmd afd_label XVDG7 /dev/xvdg7 --migrate
Connected to an idle instance.
[grid@c03n01 ~]$ asmcmd afd_label XVDG8 /dev/xvdg8 --migrate
Connected to an idle instance.
[grid@c03n01 ~]$ asmcmd afd_label XVDG9 /dev/xvdg9 --migrate
Connected to an idle instance.
[grid@c03n01 ~]$
```

27. Verify that all the disks are labeled as expected.

```
[grid@c03n01 ~]$ asmcmd afd_lsdsk
Connected to an idle instance.
--------------------------------------------------------------------
Label                     Filtering   Path
====================================================================
XVDG10                      ENABLED   /dev/xvdg10
XVDG11                      ENABLED   /dev/xvdg11
XVDF1                       ENABLED   /dev/xvdf1
XVDF10                      ENABLED   /dev/xvdf10
XVDF11                      ENABLED   /dev/xvdf11
XVDF2                       ENABLED   /dev/xvdf2
XVDF3                       ENABLED   /dev/xvdf3
XVDF5                       ENABLED   /dev/xvdf5
XVDF6                       ENABLED   /dev/xvdf6
XVDF7                       ENABLED   /dev/xvdf7
XVDF8                       ENABLED   /dev/xvdf8
XVDF9                       ENABLED   /dev/xvdf9
XVDG1                       ENABLED   /dev/xvdg1
XVDG2                       ENABLED   /dev/xvdg2
XVDG3                       ENABLED   /dev/xvdg3
XVDG5                       ENABLED   /dev/xvdg5
XVDG6                       ENABLED   /dev/xvdg6
XVDG7                       ENABLED   /dev/xvdg7
XVDG8                       ENABLED   /dev/xvdg8
```

```
XVDG9                          ENABLED   /dev/xvdg9
[grid@c03n01 ~]$
```

28. Return to your `grid` terminal session on `c03n02` and perform an ASM Filter Driver disk scan. Then confirm the visibility of the labels.

    Note: The ASM Filter Driver disk scan must be performed on every node in the cluster.

```
[grid@c03n02 ~]$ asmcmd afd_scan
Connected to an idle instance.
[grid@c03n02 ~]$ asmcmd afd_lsdsk
Connected to an idle instance.
--------------------------------------------------------------------
Label                      Filtering   Path
====================================================================
XVDG10                       ENABLED   /dev/xvdg10
XVDG11                       ENABLED   /dev/xvdg11
XVDF1                        ENABLED   /dev/xvdf1
XVDF2                        ENABLED   /dev/xvdf2
XVDF3                        ENABLED   /dev/xvdf3
XVDF5                        ENABLED   /dev/xvdf5
XVDF6                        ENABLED   /dev/xvdf6
XVDF7                        ENABLED   /dev/xvdf7
XVDF8                        ENABLED   /dev/xvdf8
XVDF9                        ENABLED   /dev/xvdf9
XVDF10                       ENABLED   /dev/xvdf10
XVDF11                       ENABLED   /dev/xvdf11
XVDG1                        ENABLED   /dev/xvdg1
XVDG2                        ENABLED   /dev/xvdg2
XVDG3                        ENABLED   /dev/xvdg3
XVDG5                        ENABLED   /dev/xvdg5
XVDG6                        ENABLED   /dev/xvdg6
XVDG7                        ENABLED   /dev/xvdg7
XVDG8                        ENABLED   /dev/xvdg8
XVDG9                        ENABLED   /dev/xvdg9
[grid@c03n02 ~]$
```

29. After the labels are visible on all the cluster nodes, restart Oracle Clusterware.

```
[grid@c03n02 ~]$ su -c "crsctl start cluster -all"
Password: <oracle>
CRS-2672: Attempting to start 'ora.cssdmonitor' on 'c03n01'
CRS-2672: Attempting to start 'ora.evmd' on 'c03n01'
CRS-2672: Attempting to start 'ora.cssdmonitor' on 'c03n02'
CRS-2672: Attempting to start 'ora.evmd' on 'c03n02'
...
CRS-2676: Start of 'ora.crsd' on 'c03n01' succeeded
```

```
CRS-2676: Start of 'ora.storage' on 'c03n02' succeeded
CRS-2672: Attempting to start 'ora.crsd' on 'c03n02'
CRS-2676: Start of 'ora.crsd' on 'c03n02' succeeded
[grid@c03n02 ~]$
```

30. Repeat the unauthorized write test on one of the newly labeled disks. Examine the associated error messages in the system log.

```
[grid@c03n02 ~]$ su -c "dd if=/dev/random of=/dev/xvdf1 bs=4096
count=1 oflag=sync"
Password: <oracle>
dd: writing `/dev/xvdf1': Input/output error
0+1 records in
0+0 records out
0 bytes (0 B) copied, 0.00143349 s, 0.0 kB/s
[grid@c03n02 ~]$ su -c "tail /var/log/messages"
Password: <oracle>
...
Aug 27 22:46:13 c03n02 kernel: F 4296301.152/140827224613
dd[18945]  afd_mkrequest_fn: write IO on ASM managed device
(major=202/minor=81)  not supported i=1 start=63 seccnt=1
pstart=63  pend=3919860
Aug 27 22:46:13 c03n02 kernel: Buffer I/O error on device xvdf1,
logical block 0
Aug 27 22:46:13 c03n02 kernel: lost page write due to I/O error
on xvdf1
[grid@c03n02 ~]$
```

31. Examine the ASM disks. Notice that each disk path now shows the ASM Filter Driver label instead of the device path that was previously used. Notice also the `library` attribute, which indicates that ASM Filter Driver is enabled for each disk.

```
[grid@c03n02 ~]$ asmcmd lsdsk -k
Total_MB  Free_MB  OS_MB  Name       Failgroup  Failgroup_Type
Library                                    Label   UDID
Product  Redund   Path
    1913      946   1913  DATA_0000  DATA_0000  REGULAR
AFD Library - Generic , version 3 (KABI_V3)  XVDF1
UNKNOWN  AFD:XVDF1
    1913      973   1913  DATA_0008  DATA_0008  REGULAR
AFD Library - Generic , version 3 (KABI_V3)  XVDF10
UNKNOWN  AFD:XVDF10
    1913      982   1913  DATA_0009  DATA_0009  REGULAR
AFD Library - Generic , version 3 (KABI_V3)  XVDF11
UNKNOWN  AFD:XVDF11
    1913      966   1913  DATA_0001  DATA_0001  REGULAR
AFD Library - Generic , version 3 (KABI_V3)  XVDF2
UNKNOWN  AFD:XVDF2
```

```
    1913      949   1913  DATA_0002  DATA_0002  REGULAR
AFD Library - Generic , version 3 (KABI_V3)  XVDF3
UNKNOWN  AFD:XVDF3
    1913      979   1913  DATA_0003  DATA_0003  REGULAR
AFD Library - Generic , version 3 (KABI_V3)  XVDF5
UNKNOWN  AFD:XVDF5
    1913      973   1913  DATA_0004  DATA_0004  REGULAR
AFD Library - Generic , version 3 (KABI_V3)  XVDF6
UNKNOWN  AFD:XVDF6
    1913      980   1913  DATA_0005  DATA_0005  REGULAR
AFD Library - Generic , version 3 (KABI_V3)  XVDF7
UNKNOWN  AFD:XVDF7
    1913      987   1913  DATA_0006  DATA_0006  REGULAR
AFD Library - Generic , version 3 (KABI_V3)  XVDF8
UNKNOWN  AFD:XVDF8
    1913      977   1913  DATA_0007  DATA_0007  REGULAR
AFD Library - Generic , version 3 (KABI_V3)  XVDF9
UNKNOWN  AFD:XVDF9
    1913      974   1913  DATA_0010  DATA_0010  REGULAR
AFD Library - Generic , version 3 (KABI_V3)  XVDG1
UNKNOWN  AFD:XVDG1
    1913     1817   1913  FRA_0000   FRA_0000   REGULAR
AFD Library - Generic , version 3 (KABI_V3)  XVDG10
UNKNOWN  AFD:XVDG10
    1913     1817   1913  FRA_0001   FRA_0001   REGULAR
AFD Library - Generic , version 3 (KABI_V3)  XVDG11
UNKNOWN  AFD:XVDG11
    1913      997   1913  DATA_0011  DATA_0011  REGULAR
AFD Library - Generic , version 3 (KABI_V3)  XVDG2
UNKNOWN  AFD:XVDG2
    1913      973   1913  DATA_0012  DATA_0012  REGULAR
AFD Library - Generic , version 3 (KABI_V3)  XVDG3
UNKNOWN  AFD:XVDG3
    1913      993   1913  DATA_0013  DATA_0013  REGULAR
AFD Library - Generic , version 3 (KABI_V3)  XVDG5
UNKNOWN  AFD:XVDG5
    1913      990   1913  DATA_0014  DATA_0014  REGULAR
AFD Library - Generic , version 3 (KABI_V3)  XVDG6
UNKNOWN  AFD:XVDG6
    1913      985   1913  DATA_0015  DATA_0015  REGULAR
AFD Library - Generic , version 3 (KABI_V3)  XVDG7
UNKNOWN  AFD:XVDG7
    1913      985   1913  DATA_0016  DATA_0016  REGULAR
AFD Library - Generic , version 3 (KABI_V3)  XVDG8
UNKNOWN  AFD:XVDG8
    1913      984   1913  DATA_0017  DATA_0017  REGULAR
AFD Library - Generic , version 3 (KABI_V3)  XVDG9
UNKNOWN  AFD:XVDG9
[grid@c03n02 ~]$
```

32. After all your ASM disks are under ASM Filter Driver control, you can adjust the ASM disk string parameter accordingly. Confirm the setting afterwards.

```
[grid@c03n02 ~]$ asmcmd dsget
parameter:/dev/xvd*, AFD:*
profile:/dev/xvd*,AFD:*
[grid@c03n02 ~]$ asmcmd dsset 'AFD:*'
[grid@c03n02 ~]$ asmcmd dsget
parameter:AFD:*
profile:AFD:*
[grid@c03n02 ~]$
```

33. Exit all your terminal sessions.

Practices for Lesson 3: Flex ASM

Chapter 3

Practices for Lesson 3: Overview

Practices Overview

In these practices, you will convert an existing two-node cluster from standard ASM to Flex ASM. You will also practice setting the number of ASM instances and monitor Flex ASM to determine how database clients are using the available ASM instances. Finally, you will see how Flex ASM insulates database clients from ASM instance failures, thus increasing database availability.

Practice 3-1: Converting to Flex ASM

Overview

In this practice, you will convert an existing two-node cluster from standard ASM to Flex ASM.

Tasks

1. Establish a terminal session connected to `c03n01` as the `grid` user and configure the terminal environment as shown below.

   Note: Make sure that you use `ssh` with the `-X` option. The `-X` option is required to ensure that your terminal environment is correctly configured to run GUI applications and utilities.

```
$ ssh -X grid@c03n01
[grid@c03n01 ~]$ . oraenv
ORACLE_SID = [grid] ? +ASM1
The Oracle base has been set to /u01/app/grid
[grid@c03n01 ~]$
```

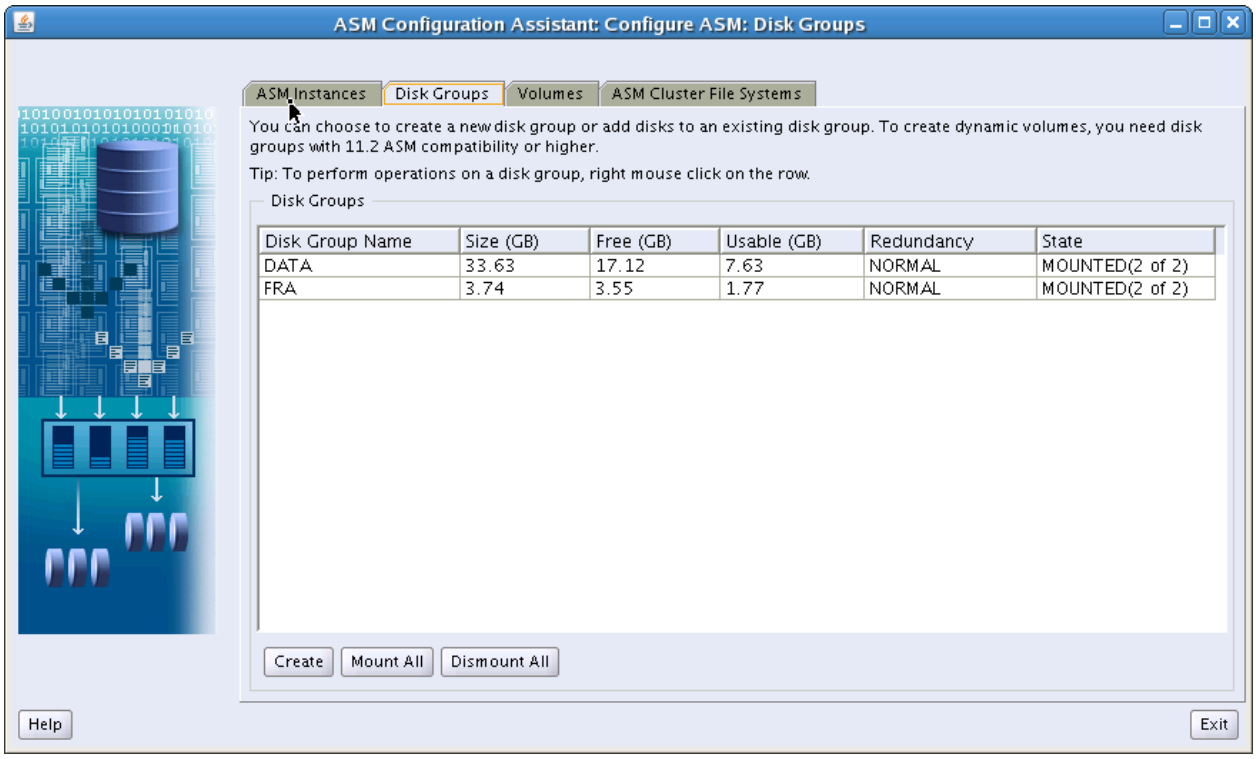
2. Examine the current ASM configuration. Notice that Flex ASM is currently disabled and that an ASM instance is running on each of the cluster nodes.

```
[grid@c03n01 ~]$ asmcmd showclustermode
ASM cluster : Flex mode disabled
[grid@c03n01 ~]$ srvctl config asm
ASM home: <CRS home>
Password file: +DATA/orapwASM
ASM listener: LISTENER
[grid@c03n01 ~]$ srvctl status asm -verbose
ASM is running on c03n01,c03n02
Detailed state on node c03n01: Started
Detailed state on node c03n02: Started
[grid@c03n01 ~]$
```

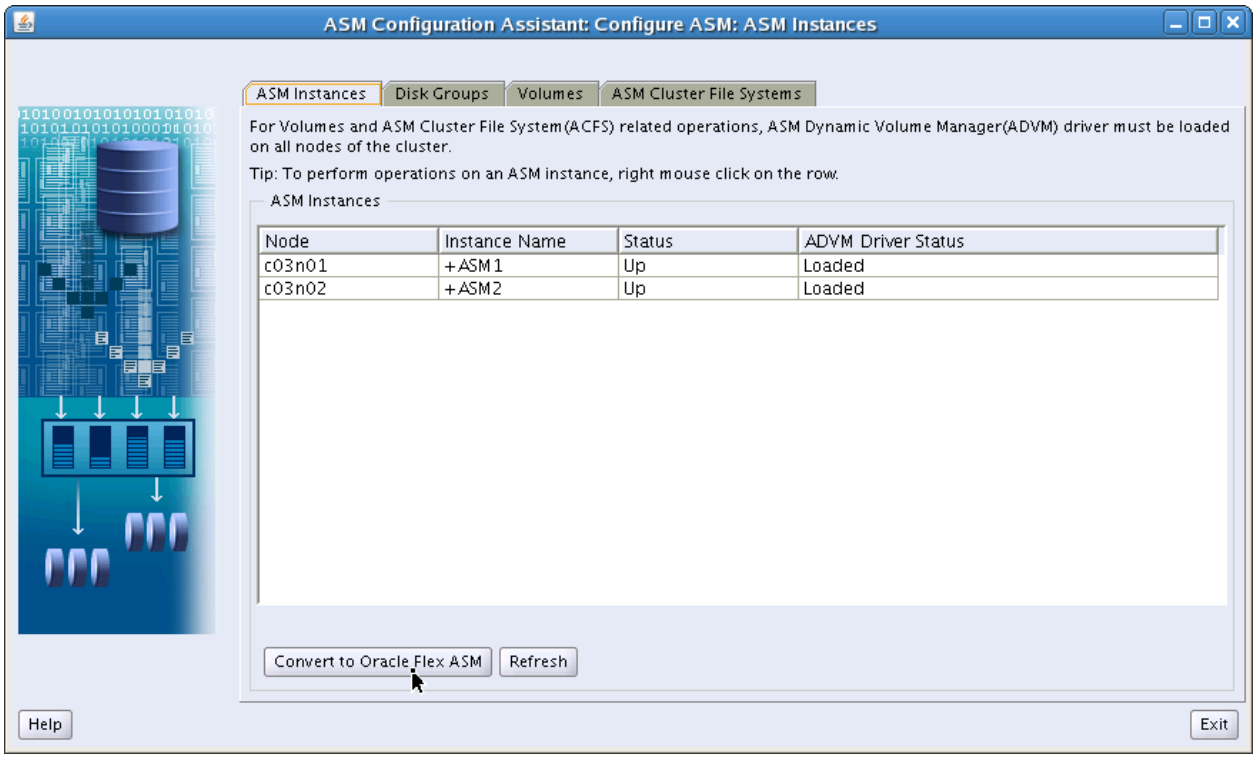
3. Launch the ASM Configuration Assistant. Ensure that you launch `asmca` in the background because you will need to use your terminal window for other tasks during the `asmca` session.

```
[grid@c03n01 ~]$ asmca &
[1] 26375
[grid@c03n01 ~]$
```

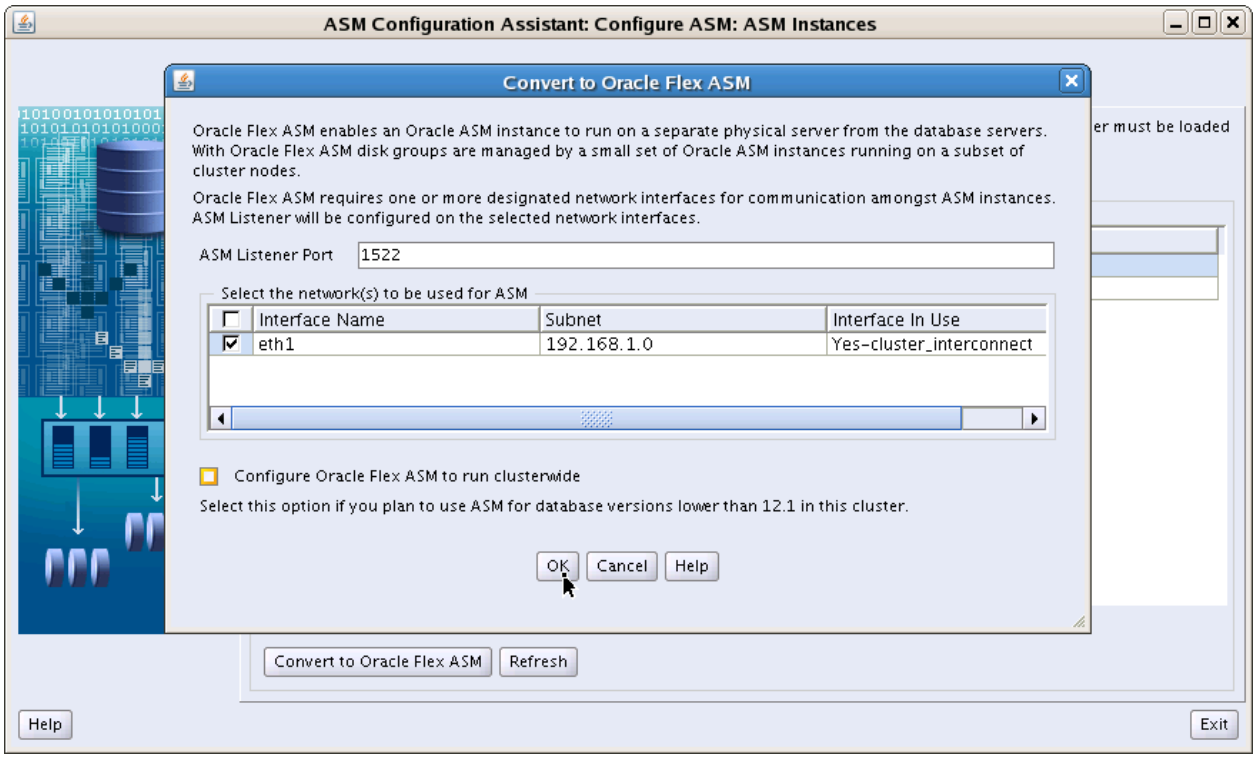
4. After the ASM Configuration Assistant appears, navigate to the ASM Instances tab.



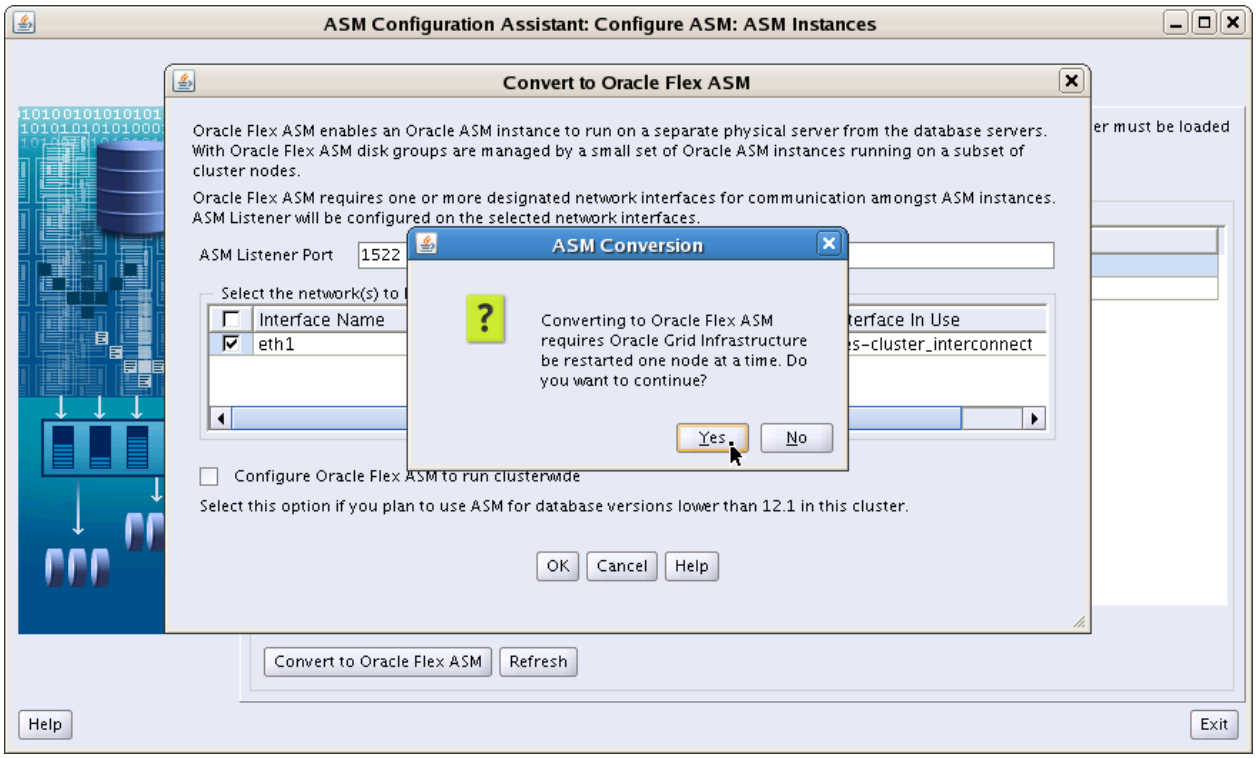
5. On the ASM Instances page, click “Convert to Oracle Flex ASM.”



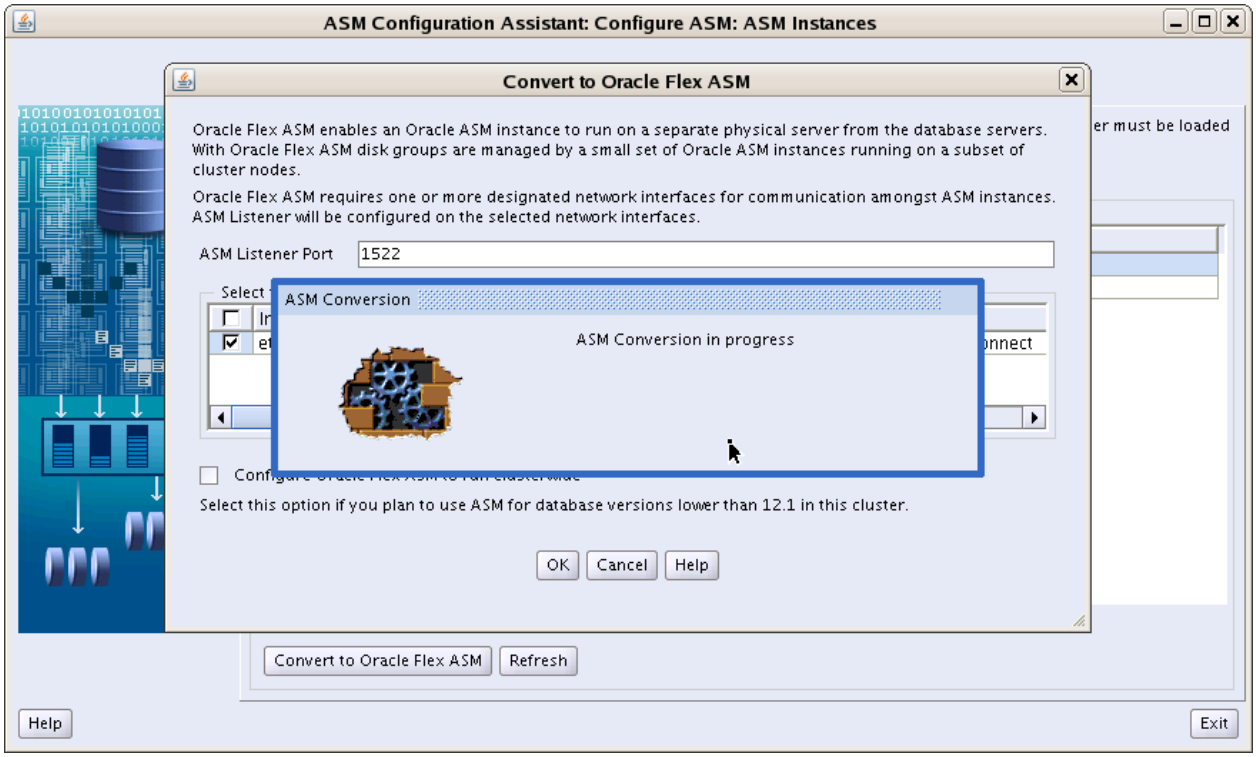
6. In the dialog box that appears, specify the ASM Listener Port number. You must specify a port number that is not already in use (such as 1522). Also, select the network interface to use for the ASM network. Typically, the ASM network shares the same interface as the cluster interconnect; however, you can choose to specify a separate ASM network if one is available. Finally, you must specify whether or not to configure an ASM on every cluster node. You must select the “Configure Oracle Flex ASM to run clusterwide” check box if you want to use database versions lower than 12.1 on the cluster. Otherwise, you can deselect this option and configure a set number of ASM instances (3 by default) across the cluster. For this practice, deselect the “Configure Oracle Flex ASM to run clusterwide” check box. After you make all the required configuration settings, click OK to continue.



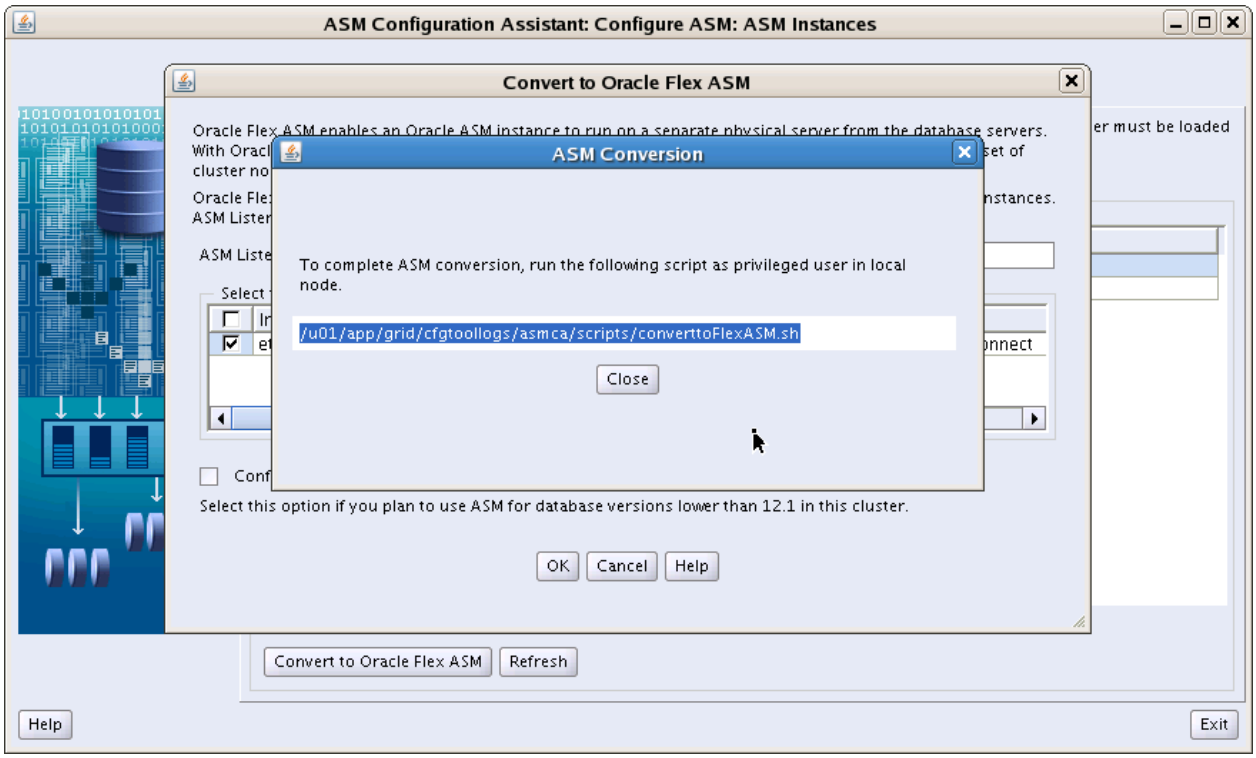
7. Converting to Flex ASM requires Oracle Grid Infrastructure to be restarted on every cluster node in a rolling manner; that is, one node at a time. Click Yes when you are ready to continue.



8. Wait while the first part of the conversion process executes.



9. Eventually you will be instructed to execute a script to complete the conversion process.

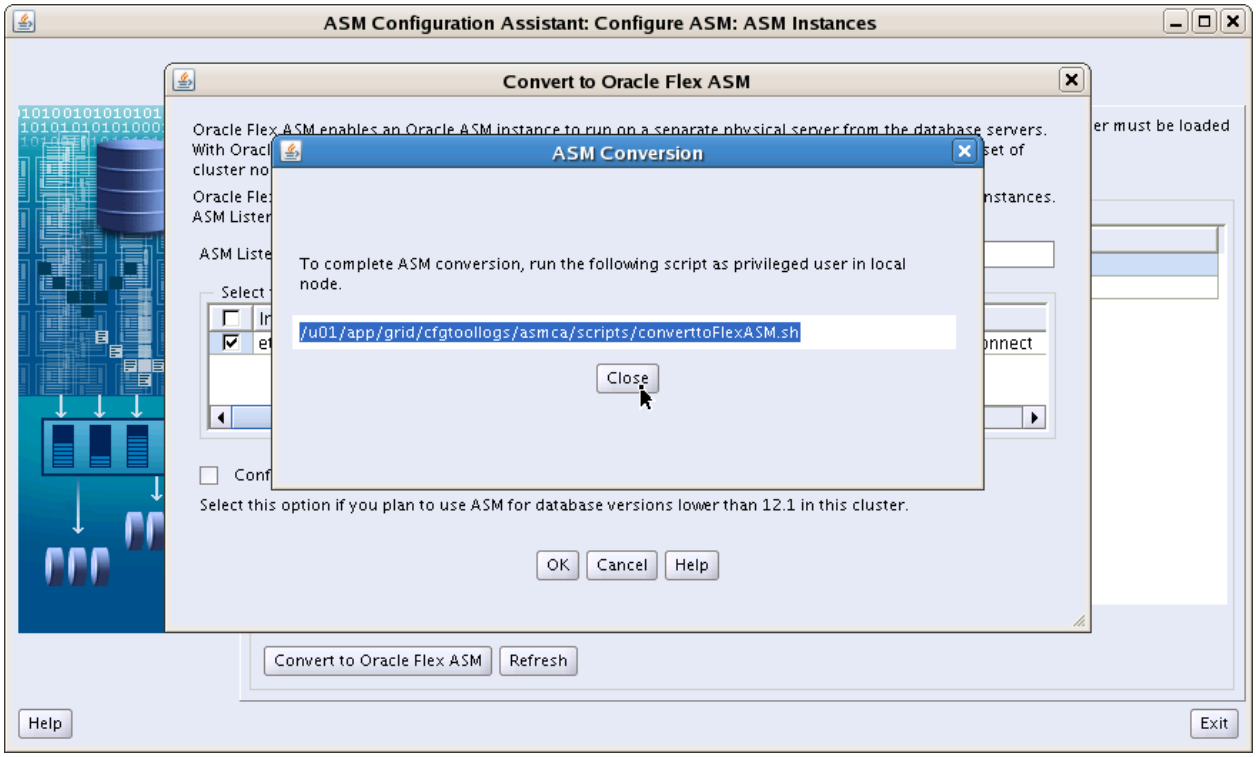


10. Using your existing terminal session, execute the ASM conversion script as the `root` user.

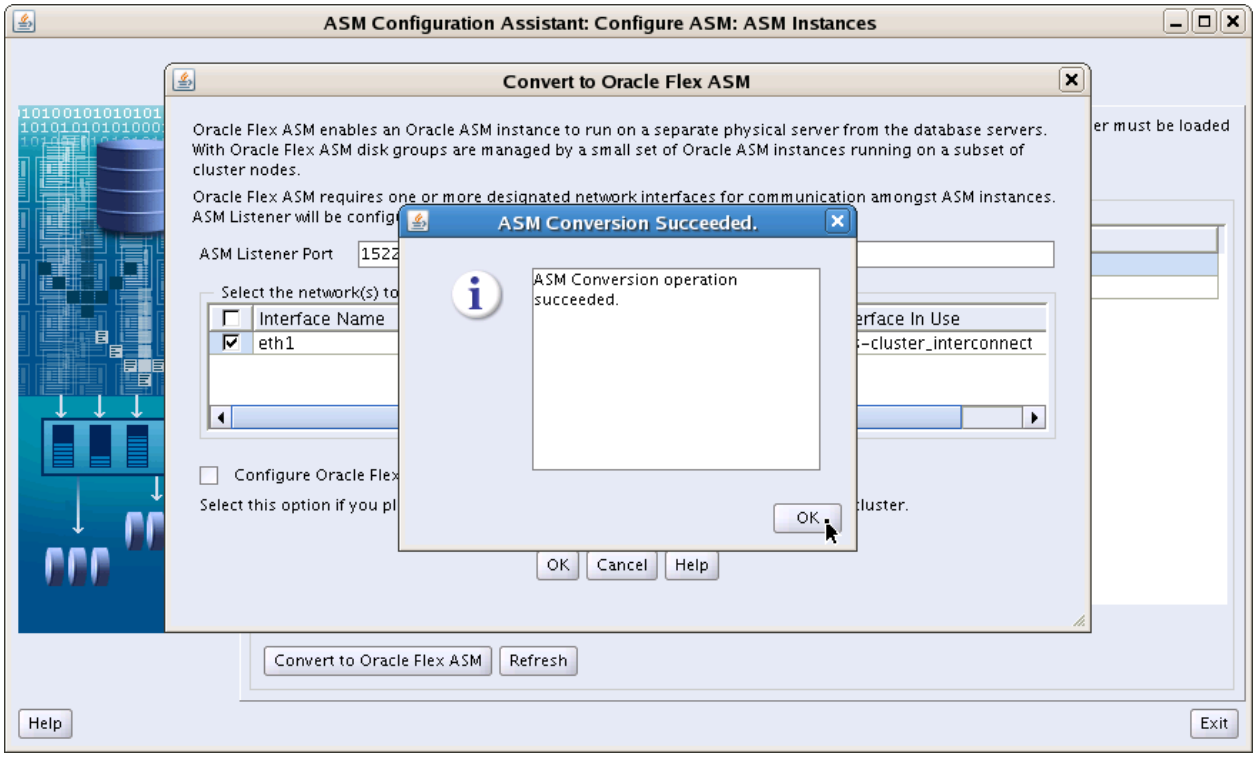
Note: The ASM conversion script stops and restarts Oracle Clusterware on every node in the cluster. The script must be run once, and only on the same node as the ASM Configuration Assistant.

```
[grid@c03n01 ~]$ su -c
"/u01/app/grid/cfgtoollogs/asmca/scripts/converttoFlexASM.sh"
Password: <oracle>
CRS-2673: Attempting to stop 'ora.crsd' on 'c03n01'
CRS-2790: Starting shutdown of Cluster Ready Services-managed
resources on 'c03n01'
CRS-2673: Attempting to stop 'ora.oc4j' on 'c03n01'
CRS-2673: Attempting to stop 'ora.FRA.dg' on 'c03n01'
CRS-2673: Attempting to stop 'ora.DATA.dg' on 'c03n01'
...
CRS-2672: Attempting to start 'ora.asm' on 'c03n02'
CRS-2676: Start of 'ora.asm' on 'c03n02' succeeded
CRS-2672: Attempting to start 'ora.storage' on 'c03n02'
CRS-2676: Start of 'ora.storage' on 'c03n02' succeeded
CRS-2672: Attempting to start 'ora.crsd' on 'c03n02'
CRS-2676: Start of 'ora.crsd' on 'c03n02' succeeded
Oracle Grid Infrastructure restarted in node c03n02
[grid@c03n01 ~]$
```

11. After the ASM conversion script completes, return to your ASM Configuration Assistant session and click Close.



12. At this point, you should see a message indicating that the ASM conversion operation succeeded. Click OK to continue.



13. After the ASM Configuration Assistant terminates, re-examine your ASM configuration. Notice that Flex ASM is now enabled with an ASM instance count setting of 3. Notice also that only two ASM instances are currently running because there are only two nodes in the cluster.

```
[grid@c03n01 ~]$ asmcmd showclustermode
ASM cluster : Flex mode enabled
[grid@c03n01 ~]$ srvctl config asm
ASM home: <CRS home>
Password file: +DATA/orapwASM
ASM listener: LISTENER
ASM instance count: 3
Cluster ASM listener: ASMNET1LSNR_ASM
[grid@c03n01 ~]$ srvctl status asm -verbose
ASM is running on c03n01,c03n02
Detailed state on node c03n01: Started
Detailed state on node c03n02: Started
[grid@c03n01 ~]$
```

14. Exit your terminal session.

Practice 3-2: Using Flex ASM

Overview

In this practice, you will perform a variety of tasks using Flex ASM. You will practice setting the number of ASM instances and monitor Flex ASM to determine how database clients are using the available ASM instances. Finally, you will see how Flex ASM insulates database clients from ASM instance failures, thus increasing database availability.

Tasks

1. Establish a terminal session connected to `c03n01` as the `grid` user and configure the terminal environment as shown in the following:

```
$ ssh grid@c03n01
[grid@c03n01 ~]$ . oraenv
ORACLE_SID = [grid] ? +ASM1
The Oracle base has been set to /u01/app/grid
[grid@c03n01 ~]$
```

2. Examine the ASM configuration. Notice that the ASM instance count is set to 3. This is the default instance count setting for Flex ASM when you select the option not to run ASM on every node in the cluster.

```
[grid@c03n01 ~]$ srvctl config asm
ASM home: <CRS home>
Password file: +DATA/orapwASM
ASM listener: LISTENER
ASM instance count: 3
Cluster ASM listener: ASMNET1LSNR_ASM
[grid@c03n01 ~]$
```

3. Use the following command to examine the ASM status across the cluster. Notice that the output shows provision for three instances, which is in line with the ASM instance count setting. However, only two instances are available because there are only two nodes in the cluster.

```
[grid@c03n01 ~]$ crsctl stat res ora.asm -t
--------------------------------------------------------------------------------
Name           Target  State        Server                   State details
--------------------------------------------------------------------------------
Cluster Resources
--------------------------------------------------------------------------------
ora.asm
      1        ONLINE  ONLINE       c03n01                   Started,STABLE
      2        ONLINE  ONLINE       c03n02                   Started,STABLE
      3        OFFLINE OFFLINE                               STABLE
--------------------------------------------------------------------------------
[grid@c03n01 ~]$
```

4. Set the ASM instance count to 2.

   Note: The minimum allowed ASM instance count setting is 2.

   Note: In this case, changing the ASM instance count does not change the actual number of running ASM instances. When required, ASM instances are automatically added and removed. If a running ASM instance is removed, the databases connected to it are transparently and automatically connected to another ASM instance. This is essentially the same as when a running ASM instance fails (or is aborted). You will examine this scenario later in the practice.

```
[grid@c03n01 ~]$ srvctl modify asm -count 2
[grid@c03n01 ~]$
```

5. Re-examine the ASM configuration and verify the change.

```
[grid@c03n01 ~]$ srvctl config asm
ASM home: <CRS home>
Password file: +DATA/orapwASM
ASM listener: LISTENER
ASM instance count: 2
Cluster ASM listener: ASMNET1LSNR_ASM
[grid@c03n01 ~]$ crsctl stat res ora.asm -t
--------------------------------------------------------------------------------
Name           Target  State        Server                   State details
--------------------------------------------------------------------------------
Cluster Resources
--------------------------------------------------------------------------------
ora.asm
      1        ONLINE  ONLINE       c03n01                   Started,STABLE
      2        ONLINE  ONLINE       c03n02                   Started,STABLE
--------------------------------------------------------------------------------
[grid@c03n01 ~]$
```

   So far you have seen how to modify the ASM instance count. In the next part of this practice, you will examine the environment to determine the database clients supported by each ASM instance. You will also terminate one of the ASM instances and see how the associated database instance automatically and transparently reconnects to the surviving ASM instance without affecting the database workload.

6. Using your `grid` user terminal session on `c03n01`, connect to ASM and examine the database instance to ASM instance mapping that is currently in force (use the SQL script located at `/stage/labs/3_2/asmclients.sql` if you prefer). In the following example output, the `c03orcl1` database instance is connected to the `+ASM1` ASM instance, and the `c03orcl2` database instance is connected to the `+ASM2` ASM instance. Notice that the connections in your practice environment may be different.

```
[grid@c03n01 ~]$ sqlplus / as sysasm

SQL*Plus: Release 12.1.0.2.0 Production ...

SQL> col client_instance_name format a20
SQL> select distinct i.instance_name asm_instance_name,
```

```
  2  c.instance_name client_instance_name, c.db_name, c.status
  3  from gv$instance i, gv$asm_client c
  4  where i.inst_id=c.inst_id;

ASM_INSTANCE_NAM CLIENT_INSTANCE_NAME DB_NAME  STATUS
---------------- -------------------- -------- ------------
+ASM1            +ASM1                +ASM     CONNECTED
+ASM1            c03orcl1             c03orcl  CONNECTED
+ASM2            +ASM2                +ASM     CONNECTED
+ASM2            c03orcl2             c03orcl  CONNECTED

SQL>
```

7. Later in this practice, you will terminate the ASM instance running on node `c03n02`. In preparation for this, examine the environment and determine which ASM instance is running on `c03n02`. In the following example output, `+ASM2` is running on `c03n02`. Note that this may differ in your practice environment.

```
SQL> select instance_name,host_name
  2  from gv$instance where host_name like 'c03n02%';

INSTANCE_NAME
----------------
HOST_NAME
----------------------------------------------------------------
+ASM2
c03n02.example.com


SQL> exit
Disconnected from Oracle Database 12c ...
[grid@c03n01 ~]$
```

8. Examine the `c03orcl` database and take note of the node on which each instance is running.

```
[grid@c03n01 ~]$ srvctl status database -d c03orcl
Instance c03orcl1 is running on node c03n01
Instance c03orcl2 is running on node c03n02
[grid@c03n01 ~]$
```

9. Leave your current terminal session running in the background. Then, using the information from the previous steps, start a SQL*Plus session on the database instance that is connected to the ASM instance running on `c03n02`. In the following example output, you connect to the `c03orcl2` database instance, which is also running on `c03n02`. Note that your practice environment may differ.

```
$ ssh oracle@c03n02
[oracle@c03n02 ~]$ . oraenv
```

```
ORACLE_SID = [oracle] ? c03orcl
The Oracle base has been set to /u01/app/oracle
[oracle@c03n02 ~]$ export ORACLE_SID=c03orcl2
[oracle@c03n02 ~]$ sqlplus / as sysdba

SQL*Plus: Release 12.1.0.2.0 Production ...

SQL>
```

10. Use the SQL script located at `/stage/labs/3_2/workload.sql` to perform some work on the database. After the workload starts, leave the script running and proceed to the next step.

```
SQL> @/stage/labs/3_2/workload

System altered.

SYSTIMESTAMP
---------------------------------------------------------------------------
27-AUG-14 11.56.03.250362 PM +00:00

  COUNT(*) AVG(AMOUNT_SOLD)
---------- ----------------
    630183       107.363386

99999 rows updated.

...
```

11. Return to your `grid` user terminal session on `c03n01` and abort the ASM instance running on `c03n02`. Afterwards, verify that the ASM instance on `c03n02` is shut down.

```
[grid@c03n01 ~]$ srvctl stop asm -node c03n02 -stopoption ABORT
-force
[grid@c03n01 ~]$ srvctl status asm -verbose
ASM is running on c03n01
Detailed state on node c03n01: Started
Detailed state on node c03n02: Instance Shutdown
[grid@c03n01 ~]$
```

12. Connect to ASM and re-examine the client mapping (use the SQL script located at `/stage/labs/3_2/asmclients.sql` if you prefer). Notice that all the database instances are now connected to the remaining ASM instance.

```
[grid@c03n01 ~]$ sqlplus / as sysasm

SQL*Plus: Release 12.1.0.2.0 Production ...
```

```
SQL> col client_instance_name format a20
SQL> select distinct i.instance_name asm_instance_name,
  2  c.instance_name client_instance_name, c.db_name, c.status
  3  from gv$instance i, gv$asm_client c
  4  where i.inst_id=c.inst_id;

ASM_INSTANCE_NAM CLIENT_INSTANCE_NAME DB_NAME  STATUS
---------------- -------------------- -------- ------------
+ASM1            +ASM1                +ASM     CONNECTED
+ASM1            c03orcl1             c03orcl  CONNECTED
+ASM1            c03orcl2             c03orcl  CONNECTED

SQL> exit
Disconnected from Oracle Database 12c ...
[grid@c03n01 ~]$
```

13. Check the workload that you started in step 11. Notice that the workload continues to run even though the original supporting ASM instance was aborted. This demonstrates how Flex ASM insulates databases from ASM instance failures. The same also occurs when an ASM instance is removed due to a change in the ASM instance count setting.

```
...

System altered.

SYSTIMESTAMP
---------------------------------------------------------------------
27-AUG-14 11.57.14.312809 PM +00:00

  COUNT(*) AVG(AMOUNT_SOLD)
---------- ----------------
    361976       130.624775

99999 rows updated.

Commit complete.

...
```

14. Return to your `grid` user terminal session on `c03n01` and restart the ASM instance on `c03n02`. Note that this may take a few minutes. Afterwards, verify that the ASM instance on `c03n02` is running.

```
[grid@c03n01 ~]$ srvctl start asm -node c03n02
[grid@c03n01 ~]$ srvctl status asm -verbose
ASM is running on c03n01,c03n02
Detailed state on node c03n01: Started
Detailed state on node c03n02: Started
[grid@c03n01 ~]$
```

15. Connect to ASM and re-examine the client mapping (use the SQL script located at `/stage/labs/3_2/asmclients.sql` if you prefer). Notice that all the database instances are still connected to one ASM instance. This is because ASM does not automatically redistribute database client connections when an ASM instance is added.

```
[grid@c03n01 ~]$ sqlplus / as sysasm

SQL*Plus: Release 12.1.0.2.0 Production ...

SQL> col client_instance_name format a20
SQL> select distinct i.instance_name asm_instance_name,
  2  c.instance_name client_instance_name, c.db_name, c.status
  3  from gv$instance i, gv$asm_client c
  4  where i.inst_id=c.inst_id;

ASM_INSTANCE_NAM CLIENT_INSTANCE_NAME DB_NAME  STATUS
---------------- -------------------- -------- ------------
+ASM1            +ASM1                +ASM     CONNECTED
+ASM1            c03orcl1             c03orcl  CONNECTED
+ASM1            c03orcl2             c03orcl  CONNECTED

SQL>
```

You have just seen that ASM does not automatically redistribute database client connections when an ASM instance is added. However, administrators can manually relocate a database client. In the final part of this practice, you will use this capability.

16. Check the workload you started in step 11. If required, restart the workload script so that you have a workload running for the rest of the practice.

```
...

SYSTIMESTAMP
---------------------------------------------------------------------
27-AUG-14 11.58.38.344022 PM +00:00

  COUNT(*) AVG(AMOUNT_SOLD)
---------- ----------------
```

```
    641599       122.908626

99999 rows updated.

...
```

17. Back in your ASM administrator SQL session, relocate the database instance that is supporting the workload you started earlier.

    Notes

    - When you perform manual relocation, you specify the database instance to relocate and ASM automatically relocates the database instance to the least-loaded ASM instance. You cannot specify the target ASM instance.
    - If the database instance is already connected to the least-loaded ASM instance, the database may be disconnected from and reconnected to the same ASM instance.

```
SQL> alter system relocate client 'c03orcl2:c03orcl';

System altered.

SQL>
```

18. Re-examine the client mapping and confirm that the manual relocation has occurred. (Use the SQL script located at `/stage/labs/3_2/asmclients.sql` if you prefer.)

```
SQL> select distinct i.instance_name asm_instance_name,
  2  c.instance_name client_instance_name, c.db_name, c.status
  3  from gv$instance i, gv$asm_client c
  4  where i.inst_id=c.inst_id;

ASM_INSTANCE_NAM CLIENT_INSTANCE_NAME DB_NAME  STATUS
---------------- -------------------- -------- ------------
+ASM1            +ASM1                +ASM     CONNECTED
+ASM1            c03orcl1             c03orcl  CONNECTED
+ASM2            c03orcl2             c03orcl  CONNECTED

SQL>
```

19. Check the workload you started earlier. Notice that the workload continues to run, even though the underlying ASM instance has changed. This capability is especially useful in situations where you need to perform maintenance operations on ASM without affecting database availability, such as patching the ASM binaries.

```
...

SYSTIMESTAMP
---------------------------------------------------------------
27-AUG-14 11.59.54.230765 PM +00:00

  COUNT(*) AVG(AMOUNT_SOLD)
```

```
---------- ----------------
    314986       171.379367

99999 rows updated.

Commit complete.

...
```

20. If the workload is still running, press Ctrl + C to stop the workload and exit SQL*Plus.

```
...

System altered.

SYSTIMESTAMP
---------------------------------------------------------------------
28-AUG-14 12.00.19.009559 AM +00:00

^Cselect count(*),avg(amount_sold) from sh.sales where rownum <
dbms_random.value(1,900000)
*
ERROR at line 1:
ORA-01013: user requested cancel of current operation

SQL> exit
Disconnected from Oracle Database 12c ...
[oracle@c03n02 ~]$
```

21. Examine the alert log file for the client database instance that was subjected to the aborted ASM instance and the manual relocation. If you prefer, use a text editor (such as `vi`) to examine the entire file, or use a `grep` command (such as the one shown in the example output) to highlight the log entries that contain references to `ASMB`. Working back from the end of the file, you should find entries that show how the database instance responded to the manual ASM instance relocation (see lines 2009–2024 in the example output) and the aborted ASM instance (see lines 1946–1958 in the example output).

```
[oracle@c03n02 ~]$ grep -B 1 -n ASMB
/u01/app/oracle/diag/rdbms/c03orcl/c03orcl2/trace/alert_c03orcl2
.log
...
1801-Wed Aug 27 23:40:54 2014
1802:NOTE: ASMB registering with ASM instance as Flex client
0xffffffffffffffff (reg:1502964290) (new connection)
--
1808-Wed Aug 27 23:40:55 2014
1809:NOTE: ASMB connected to ASM instance +ASM2 osid: 21375
(Flex mode; client id 0x10001)
```

```
--
1826-Wed Aug 27 23:40:57 2014
1827:NOTE: ASMB mounting group 1 (DATA)
--
1946-Wed Aug 27 23:56:08 2014
1947:NOTE: ASMB registering with ASM instance as Flex client
0x10001 (reg:67043542) (reconnect)
--
1951-Wed Aug 27 23:56:09 2014
1952:NOTE: ASMB connected to ASM instance +ASM1 osid: 32411
(Flex mode; client id 0x10001)
1953:NOTE: ASMB rebuilding ASM server state
1954:NOTE: ASMB rebuilt 1 (of 1) groups
1955:NOTE: ASMB rebuilt 12 (of 12) allocated files
--
1957-NOTE: 0 locks established; 0 pending writes sent to server
1958:SUCCESS: ASMB reconnected & completed ASM server state
--
2009-Thu Aug 28 00:00:02 2014
2010:NOTE: ASMB relocating from ASM instance +ASM1 (ASM-
initiated)
2011:NOTE: ASMB registering with ASM instance as Flex client
0x10001 (reg:3863476238) (reconnect)
--
2017-Thu Aug 28 00:00:04 2014
2018:NOTE: ASMB connected to ASM instance +ASM2 osid: 28977
(Flex mode; client id 0x10001)
2019:NOTE: ASMB rebuilding ASM server state
2020:NOTE: ASMB rebuilt 1 (of 1) groups
2021:NOTE: ASMB rebuilt 12 (of 12) allocated files
--
2023-NOTE: 0 locks established; 0 pending writes sent to server
2024:SUCCESS: ASMB reconnected & completed ASM server state
[oracle@c03n02 ~]$
```

22. Exit all your SQL*Plus sessions and terminal sessions.

Practices for Lesson 4: Policy-Based Cluster Management, Policy-Managed Database, and Oracle Multitenant Architecture

Chapter 4

Practices for Lesson 4: Overview

Practice Overview

In this practice, you will use policy-based cluster management capabilities to manage cluster resources in conjunction with Oracle RAC.

Practice 4-1: Using Policy-Based Cluster Management with Oracle RAC

Overview

In this practice, you will:

- Convert an existing administrator-managed RAC database to a policy-managed database
- Define a cluster policy set and examine how cluster policies can be used to share cluster resources between multiple Oracle RAC databases
- Create a container database and examine how policy-based management can be used to control access to pluggable database services

Tasks

Part 1: Converting to a Policy-Managed Database

In the first part of this practice, you will convert an existing administrator-managed RAC database to a policy-managed database.

1. Establish a terminal session connected to `c03n01` as the `oracle` user and configure the terminal environment as shown below.

   Note: Make sure that you use `ssh` with the `-X` option because this option is required for you to ensure that the terminal environment is correctly configured to run GUI applications and utilities.

```
$ ssh -X oracle@c03n01
[oracle@c03n01 ~]$ . oraenv
ORACLE_SID = [oracle] ? c03orcl
The Oracle base has been set to /u01/app/oracle
[oracle@c03n01 ~]$ export ORACLE_SID=c03orcl1
[oracle@c03n01 ~]$
```

2. Examine the configuration of the `c03orcl` database. Notice that it is configured as an administrator-managed database, and that it is not associated with a server pool.

```
[oracle@c03n01 ~]$ srvctl config database -d c03orcl
Database unique name: c03orcl
Database name: c03orcl
Oracle home: /u01/app/oracle/product/12.1.0/dbhome_1
Oracle user: oracle
Spfile: +DATA/C03ORCL/PARAMETERFILE/spfile.289.856528741
Password file: +DATA/C03ORCL/PASSWORD/pwdc03orcl.276.856528001
Domain: example.com
Start options: open
Stop options: immediate
Database role: PRIMARY
Management policy: AUTOMATIC
Server pools:
```

```
Disk Groups: FRA,DATA
Mount point paths:
Services:
Type: RAC
Start concurrency:
Stop concurrency:
OSDBA group: dba
OSOPER group: oper
Database instances: c03orcl1,c03orcl2
Configured nodes: c03n01,c03n02
Database is administrator managed
[oracle@c03n01 ~]$
```

Policy-managed databases are associated with one or more server pools, which define where the database runs. Administrator-managed databases are not explicitly associated with a server pool and they can run on any of the nodes associated with the built-in `Generic` server pool.

3. Examine the current server pool status. Notice that the cluster contains two built-in server pools and that all of the cluster nodes are allocated to the `Generic` pool.

```
[oracle@c03n01 ~]$ srvctl status srvpool
Server pool name: Free
Active servers count: 0
Server pool name: Generic
Active servers count: 2
[oracle@c03n01 ~]$
```

4. Add a new server pool named `dev`. Specify that it contains a minimum of 1 server and a maximum of 2 servers. Then, examine the server pool status and confirm the creation of the new server pool. Notice that all of the servers remain in the `Generic` pool even though you specified a minimum of 1 server for the `dev` pool. This is because your administrator-managed database is currently using all the servers in the `Generic` pool, so there are no servers available for the `dev` pool at this time.

```
[oracle@c03n01 ~]$ srvctl add srvpool -serverpool dev -min 1
-max 2
[oracle@c03n01 ~]$ srvctl status srvpool
Server pool name: Free
Active servers count: 0
Server pool name: Generic
Active servers count: 2
Server pool name: dev
Active servers count: 0
[oracle@c03n01 ~]$
```

5. Stop the `c03orcl` database.

```
[oracle@c03n01 ~]$ srvctl stop database -d c03orcl
[oracle@c03n01 ~]$
```

6. Modify the `c03orcl` database and associate it with the `dev` server pool. When you explicitly associate a database with a server pool, it becomes a policy-managed database.

```
[oracle@c03n01 ~]$ srvctl modify database -d c03orcl
-serverpool dev
[oracle@c03n01 ~]$
```

7. Examine the configuration of the `c03orcl` database. Confirm that it is associated with the `dev` server pool and that it is now identified as a policy-managed database.

```
[oracle@c03n01 ~]$ srvctl config database -d c03orcl
Database unique name: c03orcl
Database name: c03orcl
Oracle home: /u01/app/oracle/product/12.1.0/dbhome_1
Oracle user: oracle
Spfile: +DATA/C03ORCL/PARAMETERFILE/spfile.289.856528741
Password file: +DATA/C03ORCL/PASSWORD/pwdc03orcl.276.856528001
Domain: example.com
Start options: open
Stop options: immediate
Database role: PRIMARY
Management policy: AUTOMATIC
Server pools: dev
Disk Groups: FRA,DATA
Mount point paths:
Services:
Type: RAC
Start concurrency:
Stop concurrency:
OSDBA group: dba
OSOPER group: oper
Database instances:
Configured nodes:
Database is policy managed
[oracle@c03n01 ~]$
```

8. Restart the `c03orcl` database. Then, examine the status of the database.

Notes:

The naming convention for policy-managed database instances adds an underscore in each instance name; for example, `c03orcl_1` instead of `c03orcl1`.

The instance numbering is not static, and may differ from the example shown below. For example, when the database was administrator-managed, instance `c03orcl1` always ran

on node c03n01. However, with policy-managed database, instance c03orcl_1 can run on any node associated with the dev server pool.

```
[oracle@c03n01 ~]$ srvctl start database -d c03orcl
[oracle@c03n01 ~]$ srvctl status database -d c03orcl
Instance c03orcl_1 is running on node c03n02
Instance c03orcl_2 is running on node c03n01
[oracle@c03n01 ~]$
```

9. Re-examine the server pool status. Notice that both nodes moved from the Generic pool to the dev pool. When you stopped your administrator-managed database, the nodes were no longer required by the Generic pool, and when you restarted the database as a policy-managed database, both nodes were available for assignment to the dev pool.

```
[oracle@c03n01 ~]$ srvctl status srvpool
Server pool name: Free
Active servers count: 0
Server pool name: Generic
Active servers count: 0
Server pool name: dev
Active servers count: 2
[oracle@c03n01 ~]$
```

Part 2: Creating a New Policy-Managed Database.

In the next part of this practice, you will create an additional server pool and a new policy-managed database.

10. Use the following command to attempt to add another server pool. Notice that the attempt fails because it would require stopping a resource in another server pool with equal priority.

```
[oracle@c03n01 ~]$ srvctl add srvpool -serverpool rep -min 1
-max 2
PRCS-1009 : Failed to create server pool rep
PRCR-1071 : Failed to register or update server pool ora.rep
CRS-2736: The operation requires stopping resource
'ora.c03orcl.db' on server 'c03n02'
CRS-2737: Unable to register server pool 'ora.rep' as this will
affect running resources, but the force option was not specified
[oracle@c03n01 ~]$
```

11. Add a new server pool named rep. Specify that it contains a minimum of 1 server and a maximum of 2 servers. Also, specify the force option (-f).

```
[oracle@c03n01 ~]$ srvctl add srvpool -serverpool rep -min 1
-max 2 -f
[oracle@c03n01 ~]$
```

12. Examine the server pool configuration and confirm the existence of the rep server pool.

```
[oracle@c03n01 ~]$ srvctl config srvpool
Server pool name: Free
Importance: 0, Min: 0, Max: -1
Category:
Candidate server names:
Server pool name: Generic
Importance: 0, Min: 0, Max: -1
Category:
Candidate server names:
Server pool name: dev
Importance: 0, Min: 1, Max: 2
Category: hub
Candidate server names:
Server pool name: rep
Importance: 0, Min: 1, Max: 2
Category: hub
Candidate server names:
[oracle@c03n01 ~]$
```

13. Examine the server pool status. Notice that there is now one server in the `dev` pool and one server in the `rep` pool. This is because Oracle Clusterware always attempts to satisfy each server pool's minimum setting before allocating additional servers to any pool.

```
[oracle@c03n01 ~]$ srvctl status srvpool -detail
Server pool name: Free
Active servers count: 0
Active server names:
Server pool name: Generic
Active servers count: 0
Active server names:
Server pool name: dev
Active servers count: 1
Active server names: c03n01
NAME=c03n01 STATE=ONLINE
Server pool name: rep
Active servers count: 1
Active server names: c03n02
NAME=c03n02 STATE=ONLINE
[oracle@c03n01 ~]$
```

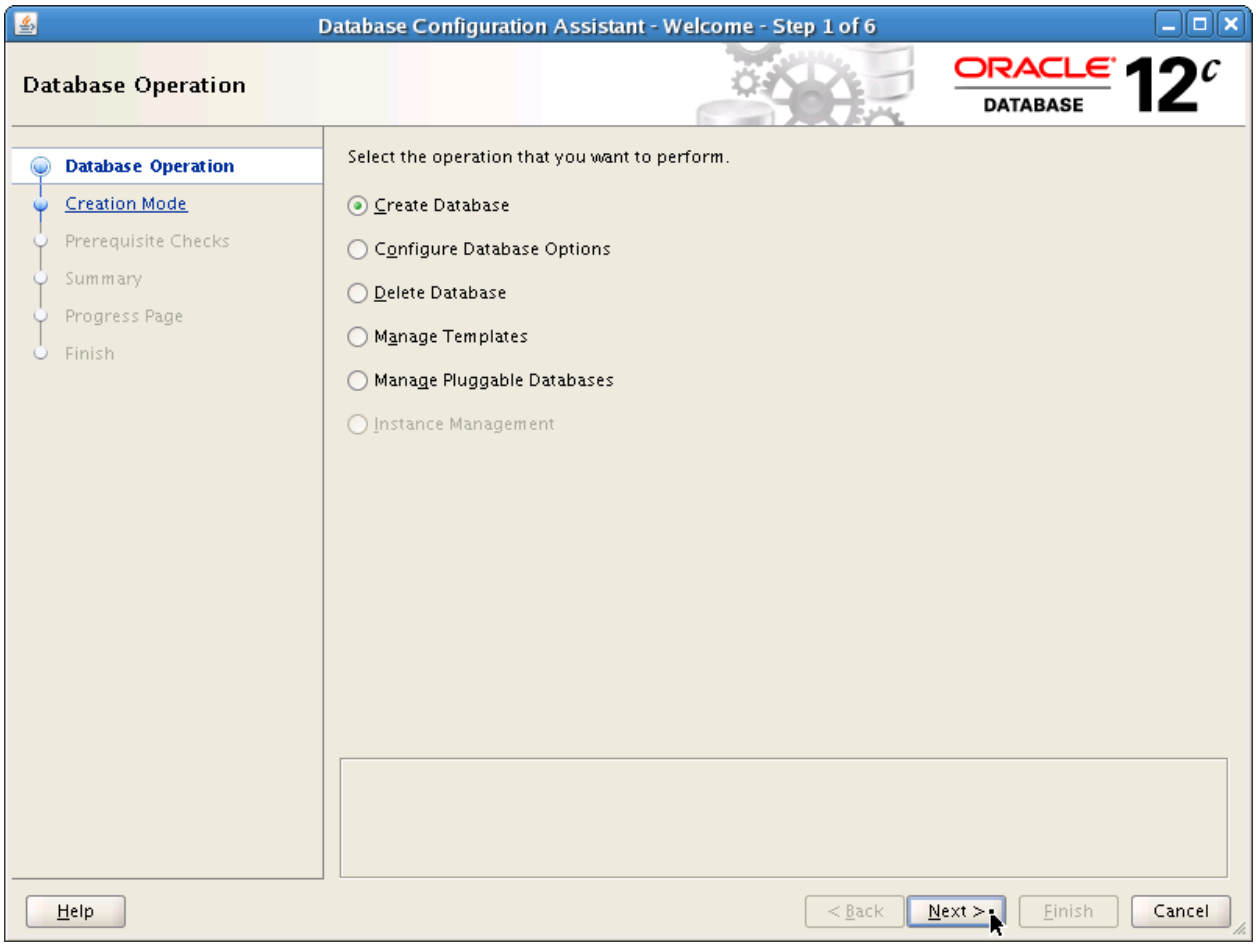
14. Examine the status of your policy-managed database. Cross-reference the information in the server pool status output to confirm that your policy-managed database is running in the `dev` server pool.

```
[oracle@c03n01 ~]$ srvctl status database -d c03orcl
Instance c03orcl_2 is running on node c03n01
[oracle@c03n01 ~]$
```

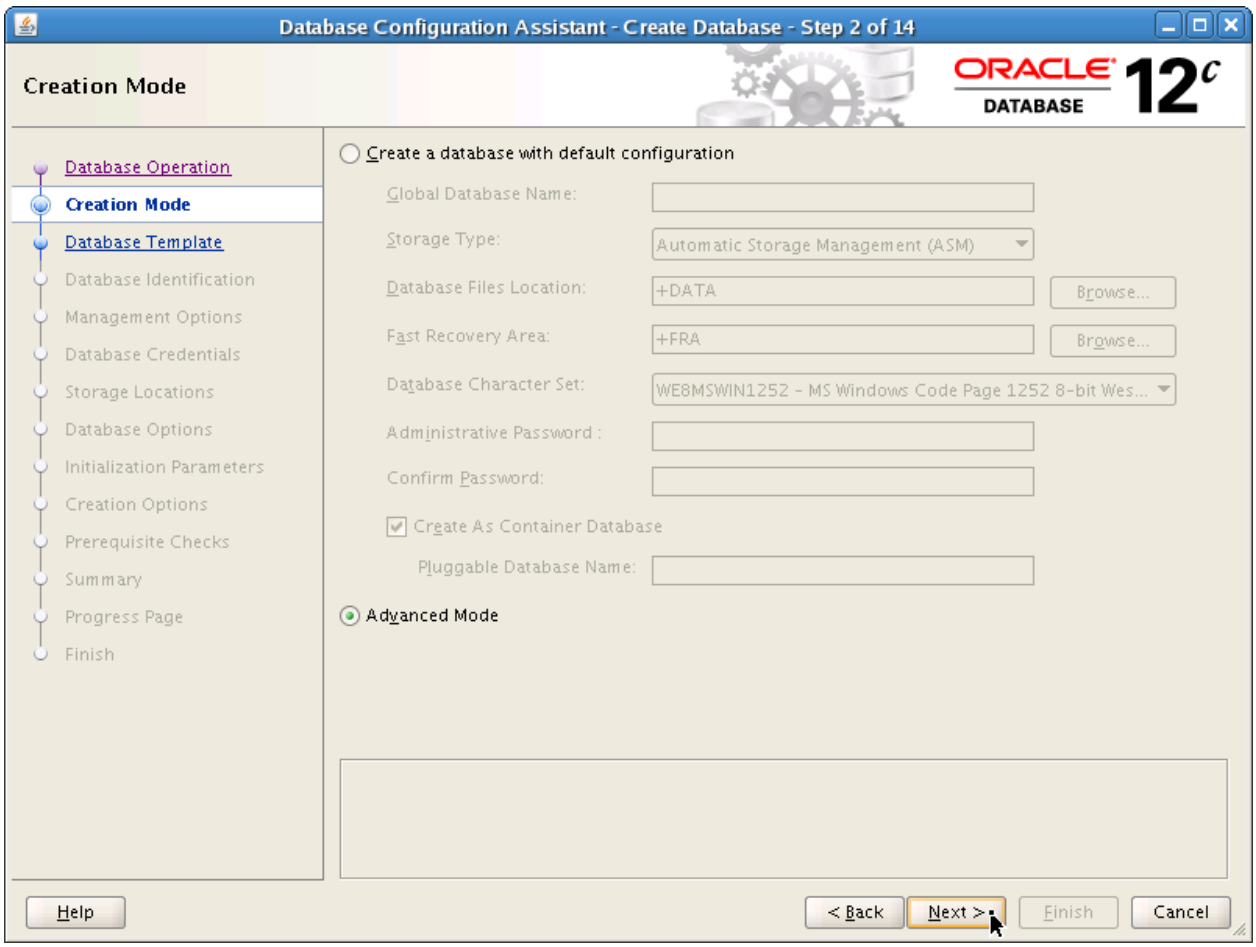
15. Next, you will create another policy-managed database that is associated with the `rep` server pool. Launch the Database Configuration Assistant (DBCA).

```
[oracle@c03n01 ~]$ dbca &
[1] 31889
[oracle@c03n01 ~]$
```

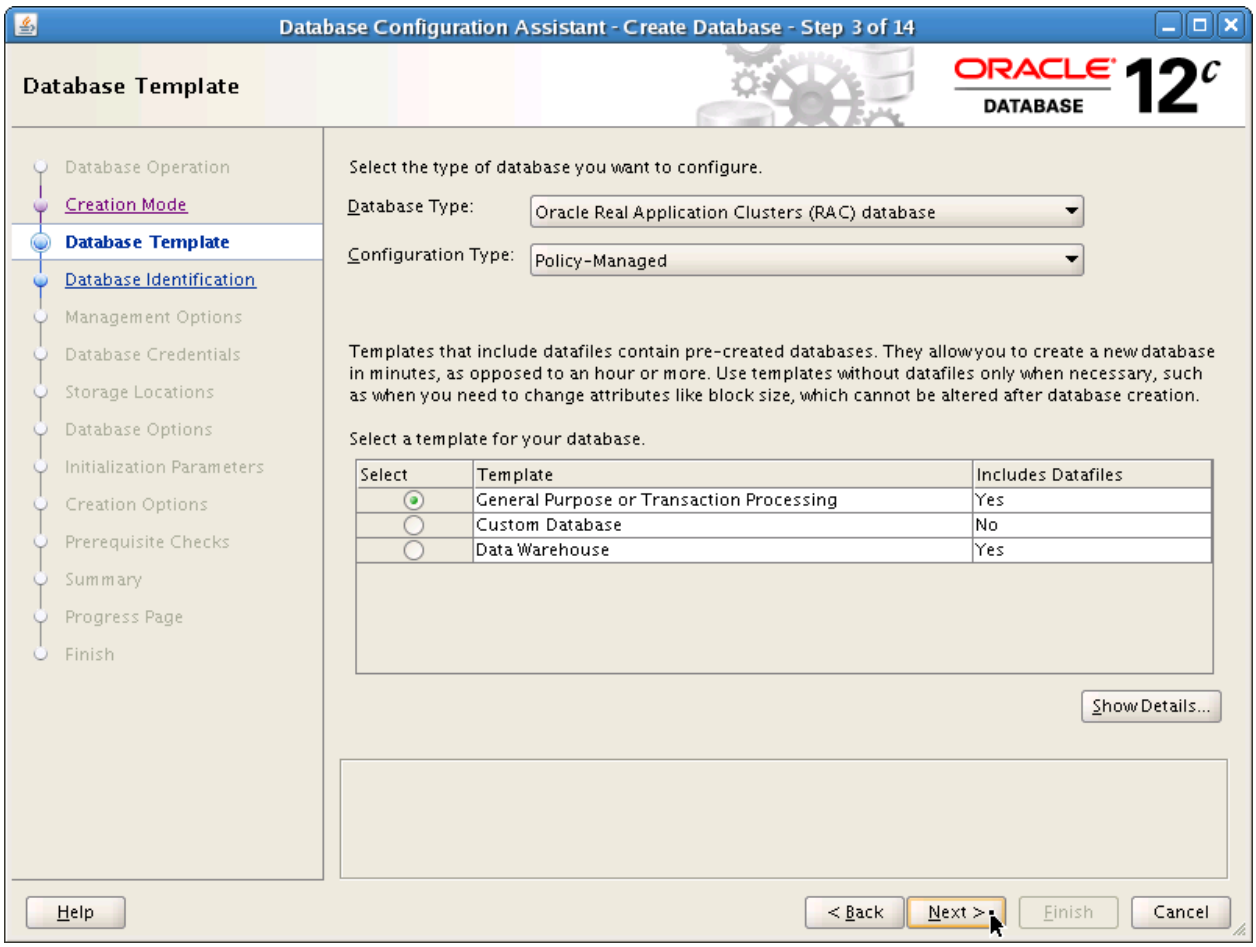
16. On the Database Operation page, accept the default option to create a new database and click Next to continue.



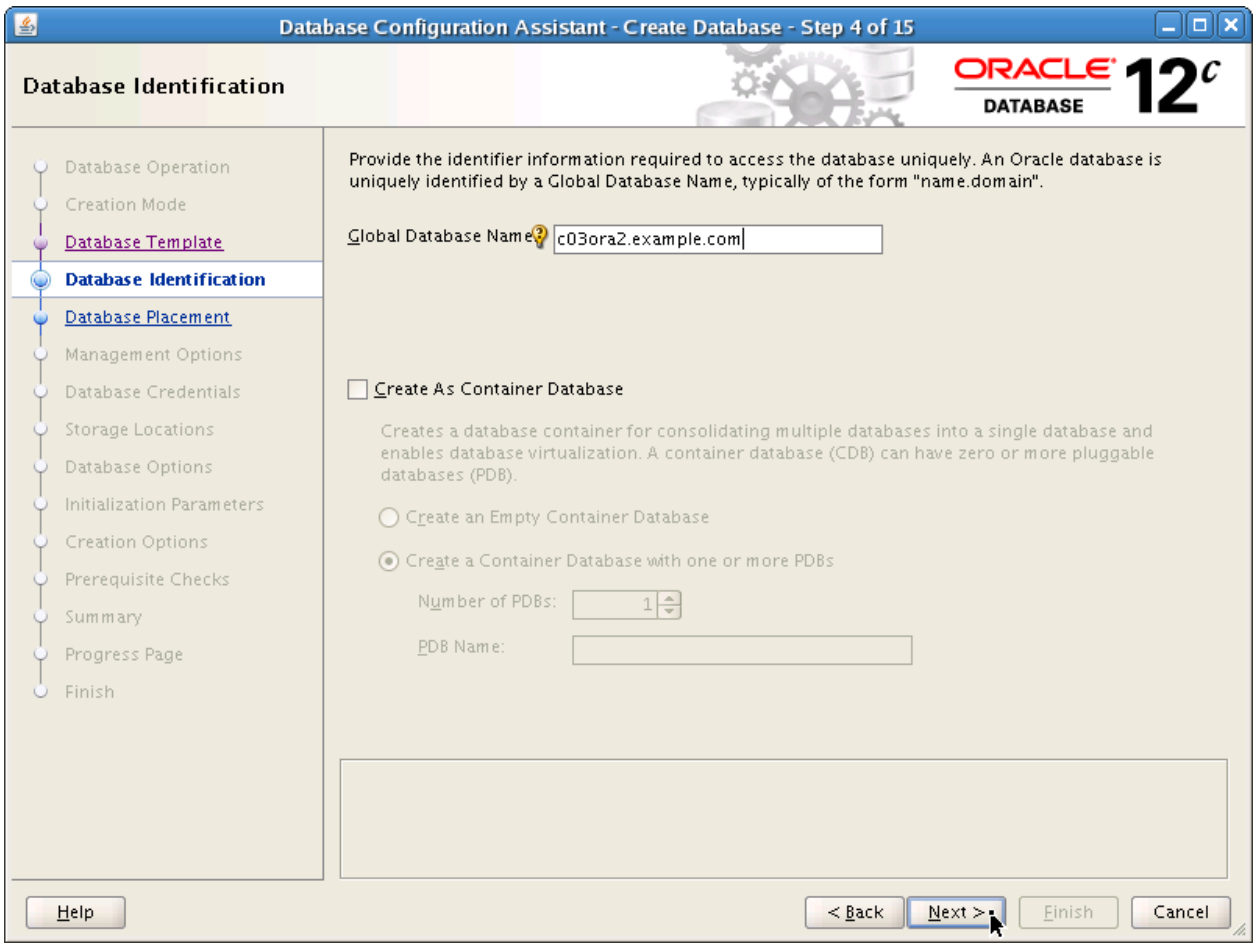
17. On the Creation Mode page, accept the default option to use the advanced mode and click Next to continue.



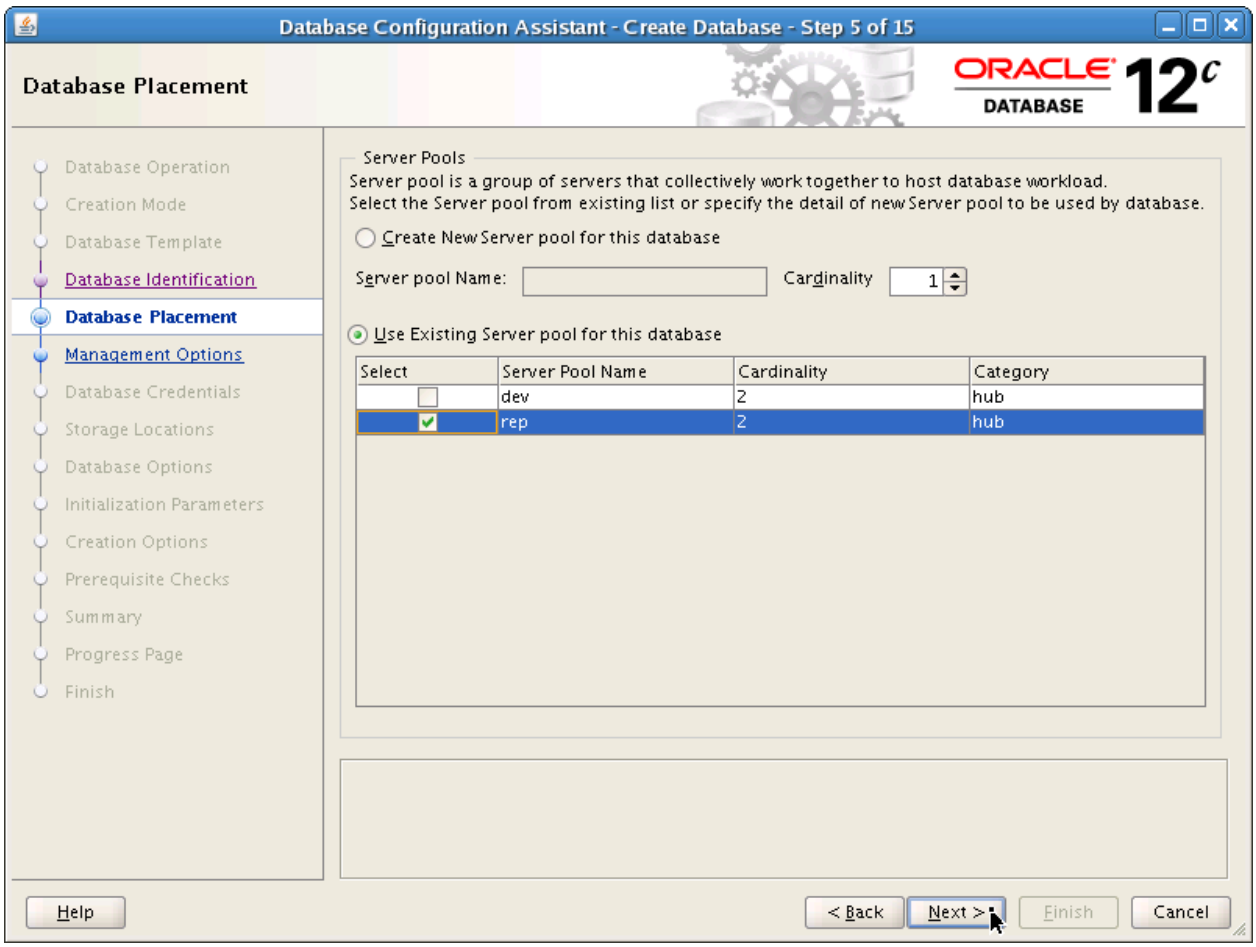
18. On the Database Template page, accept all of the default options and click Next to continue.



19. On the Database Identification page, specify `c03ora2.example.com` as the Global Database Name and click Next to continue.

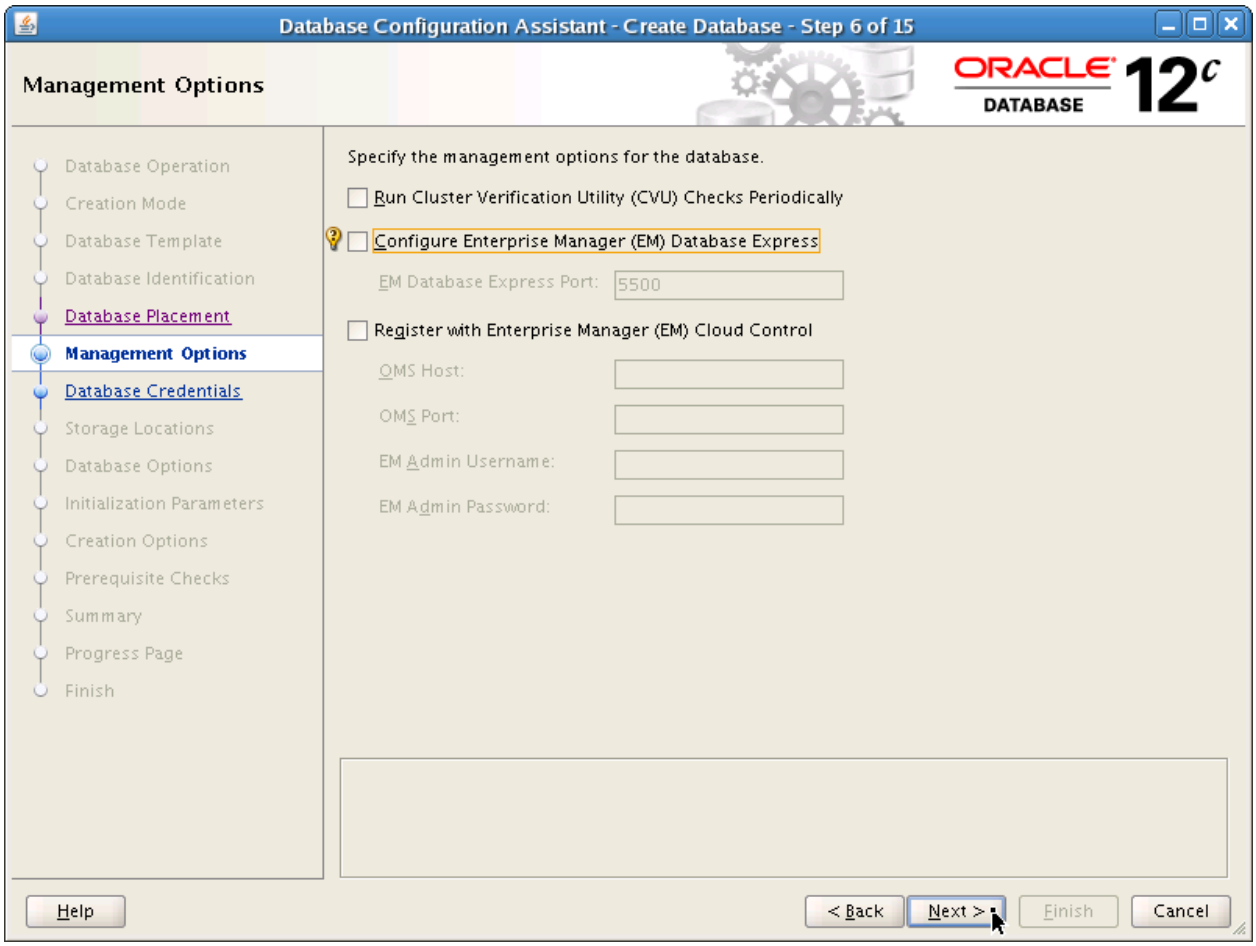


20. On the Database Placement page, select the option to use an existing server pool and select the `rep` server pool. Then, click Next to continue.

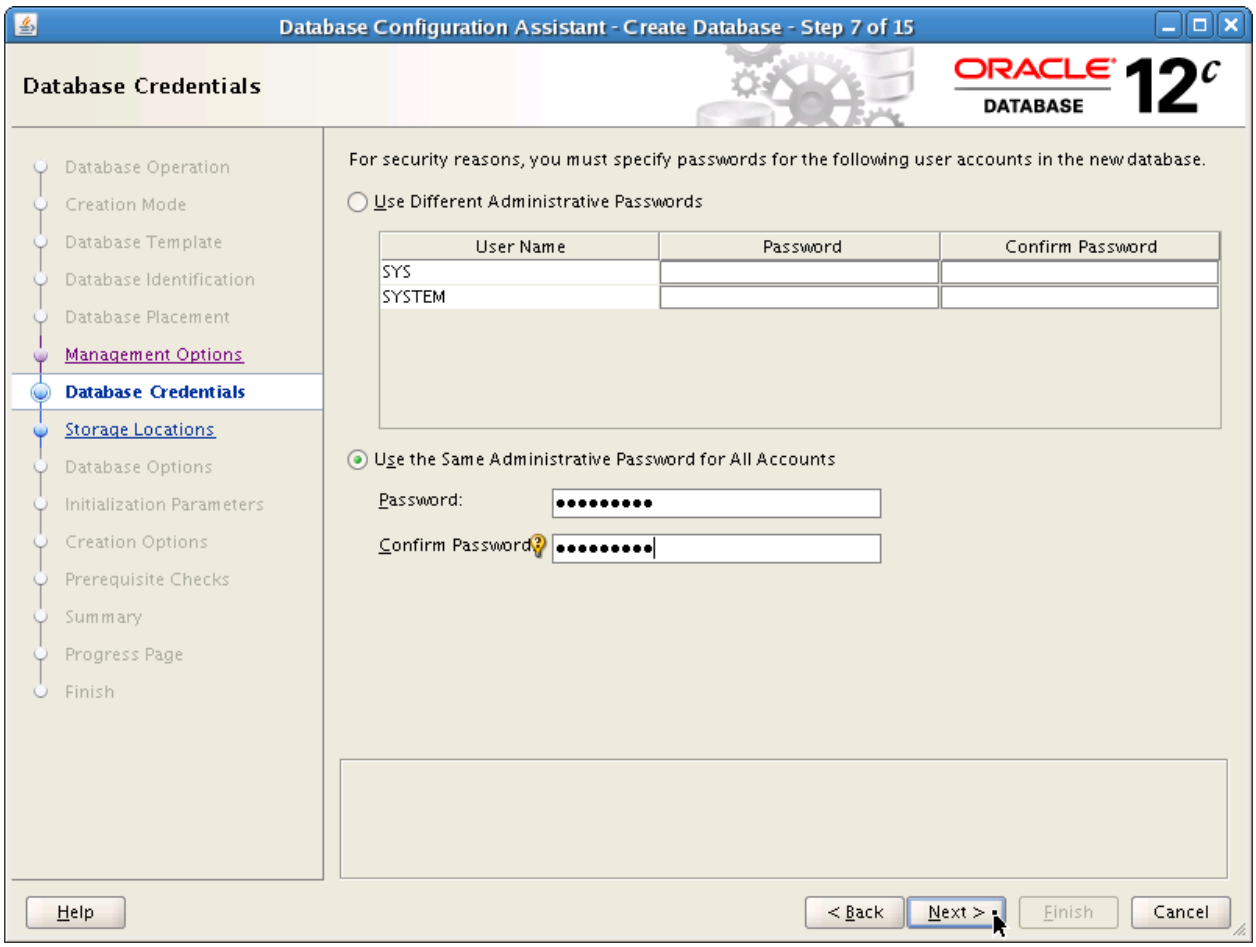


21. On the Management Options page, deselect all of the selected options and click Next to continue.

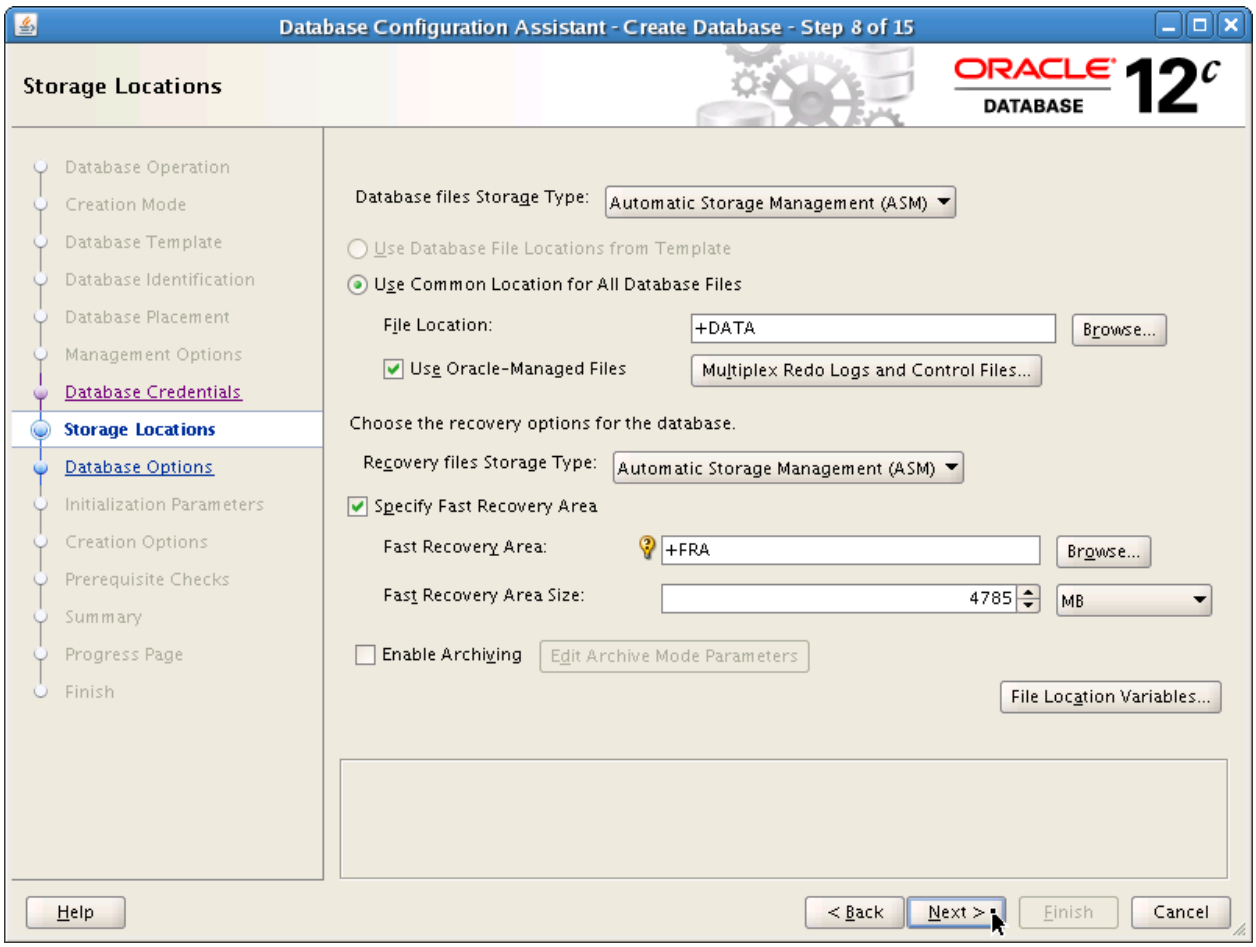
    Note: This configuration is not required or generally recommended and is only being used because none of the management options are required later in the practice.



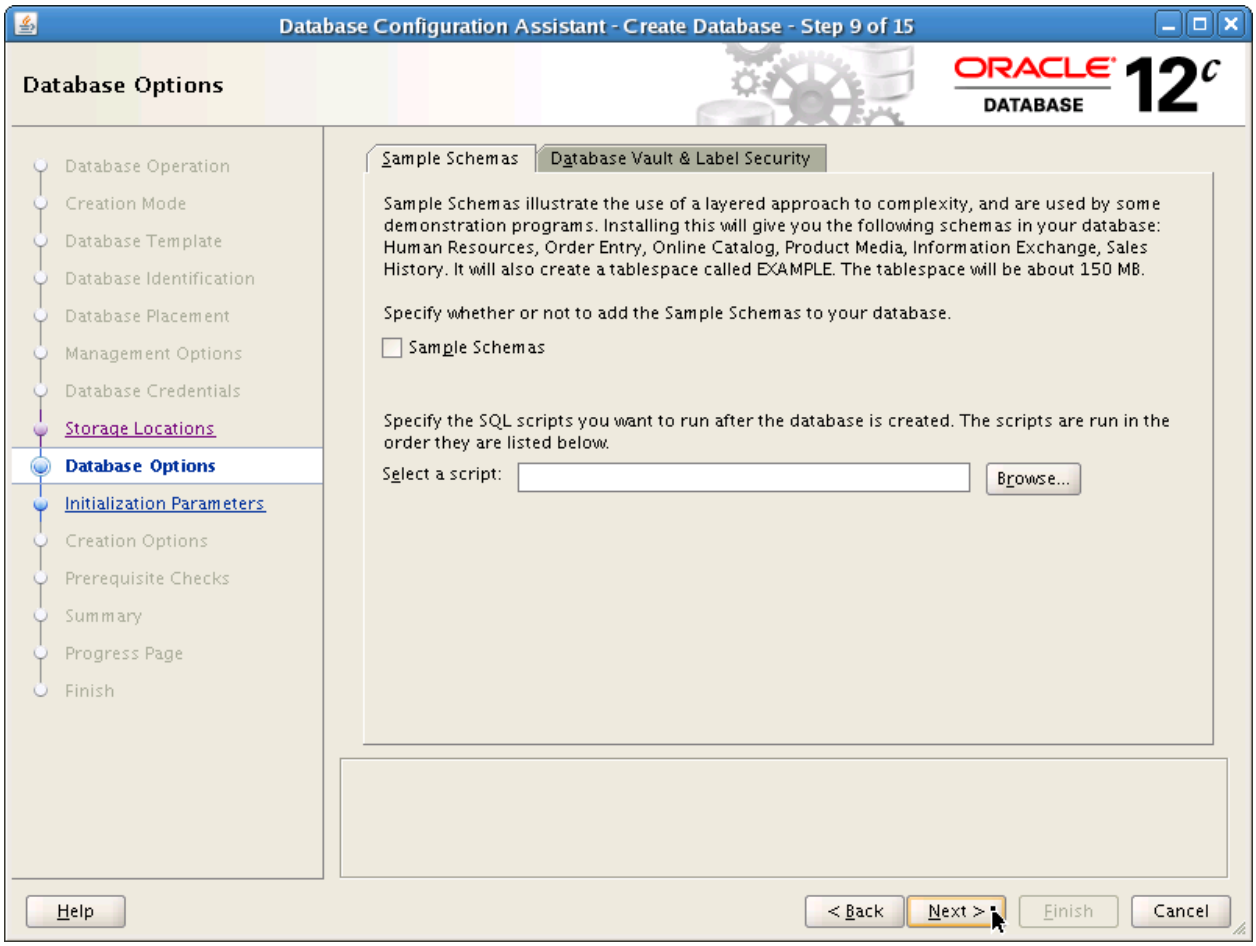
22. On the Database Credentials page, select the option to use the same password for all administrative accounts. Specify `oracle_4U` as the password and click Next to continue.



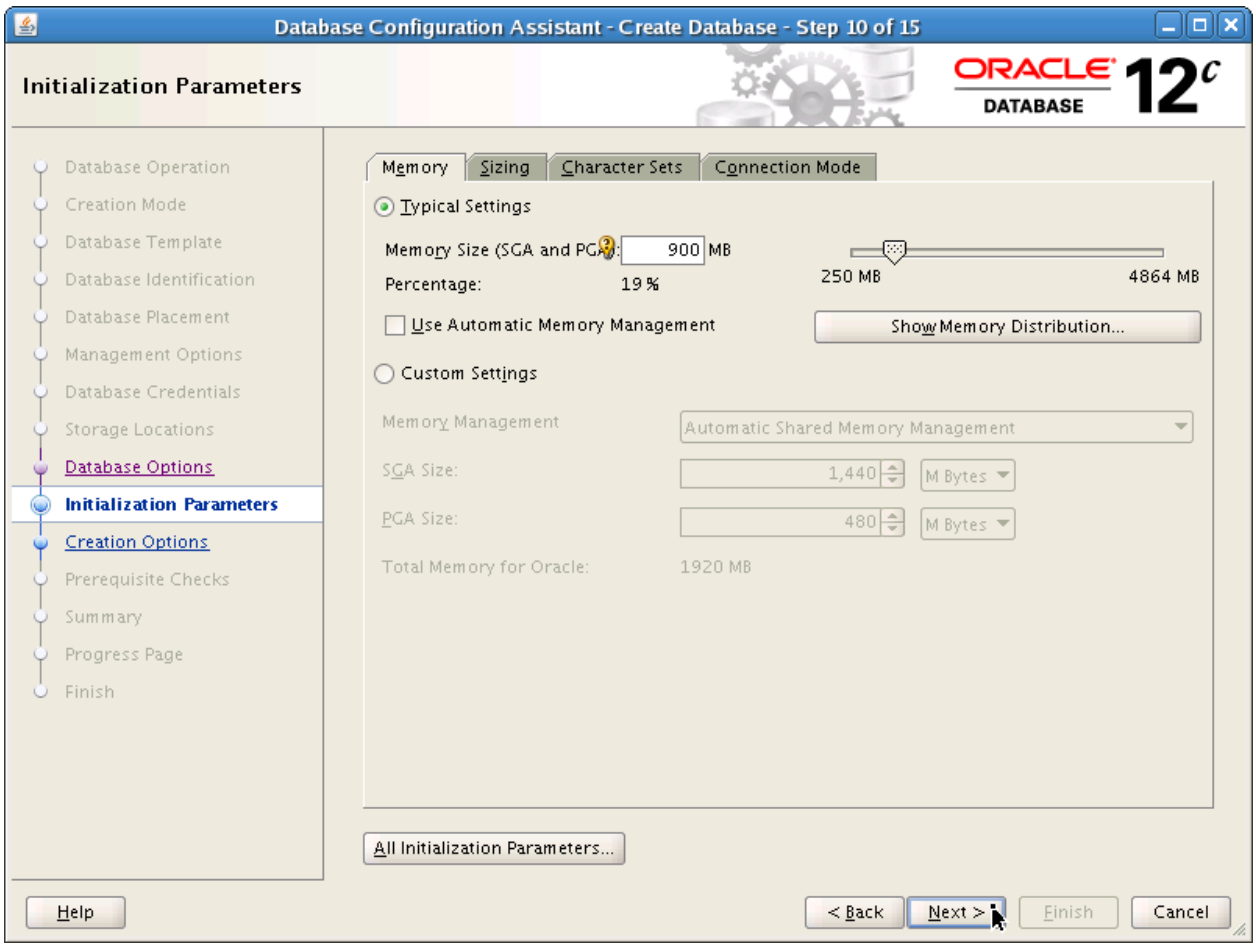
23. On the Storage Locations page, specify `+FRA` as the Fast Recovery Area location. Accept all of the other defaults and click Next to continue.



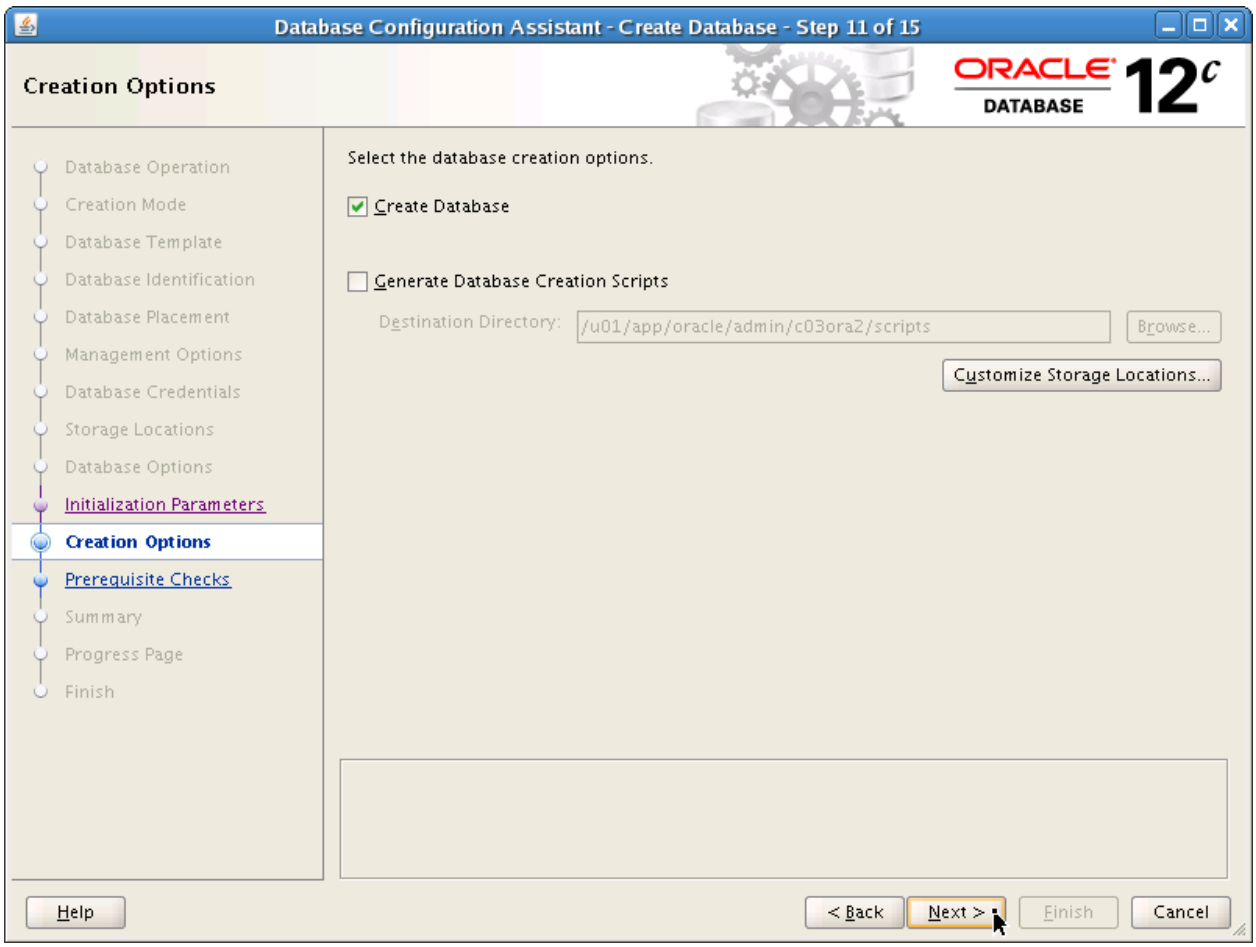
24. On the Database Options page, accept all of the default options and click Next to continue.



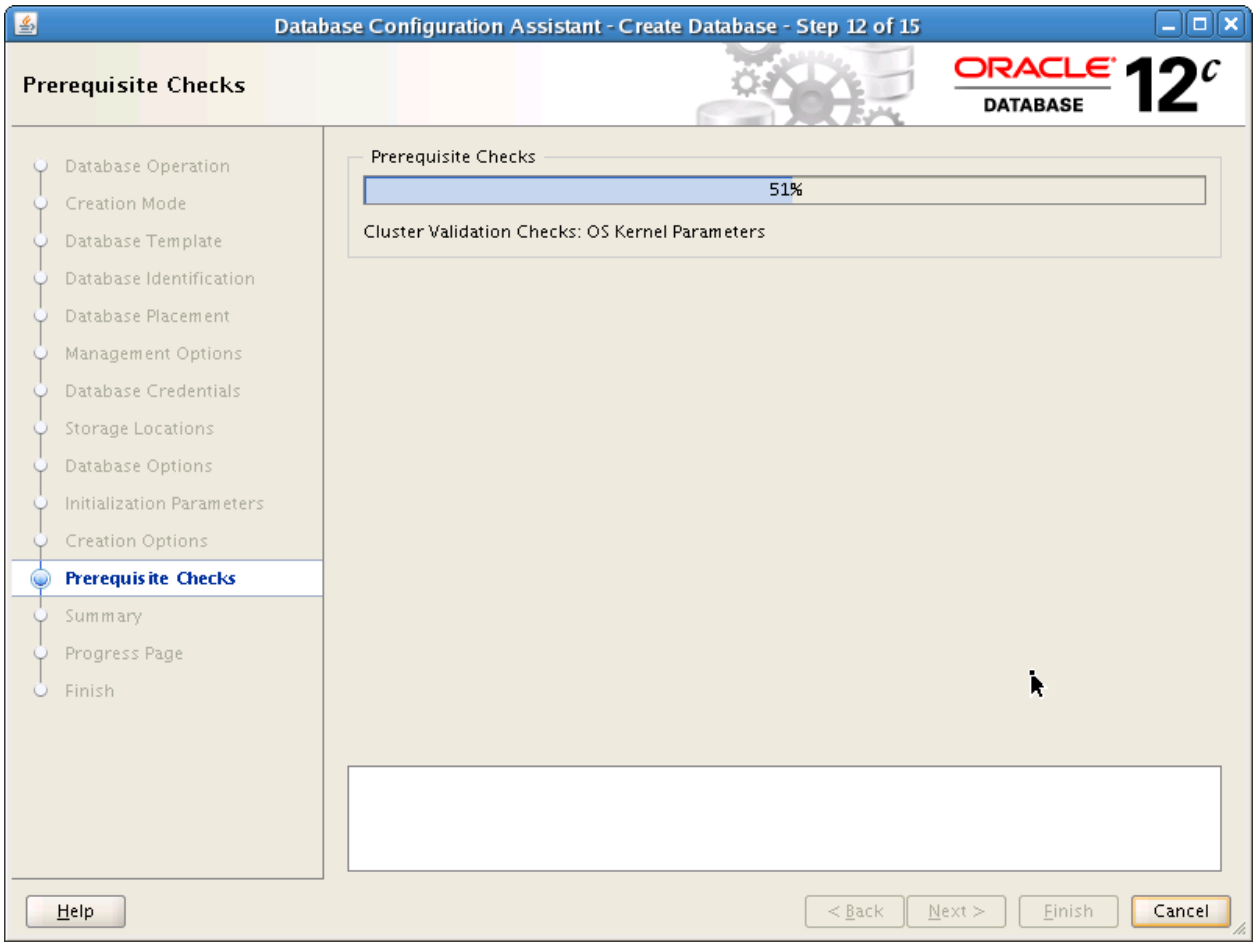
25. On the Initialization Parameters page, specify `900MB` as the database memory size. Accept all the other default options and click Next to continue.



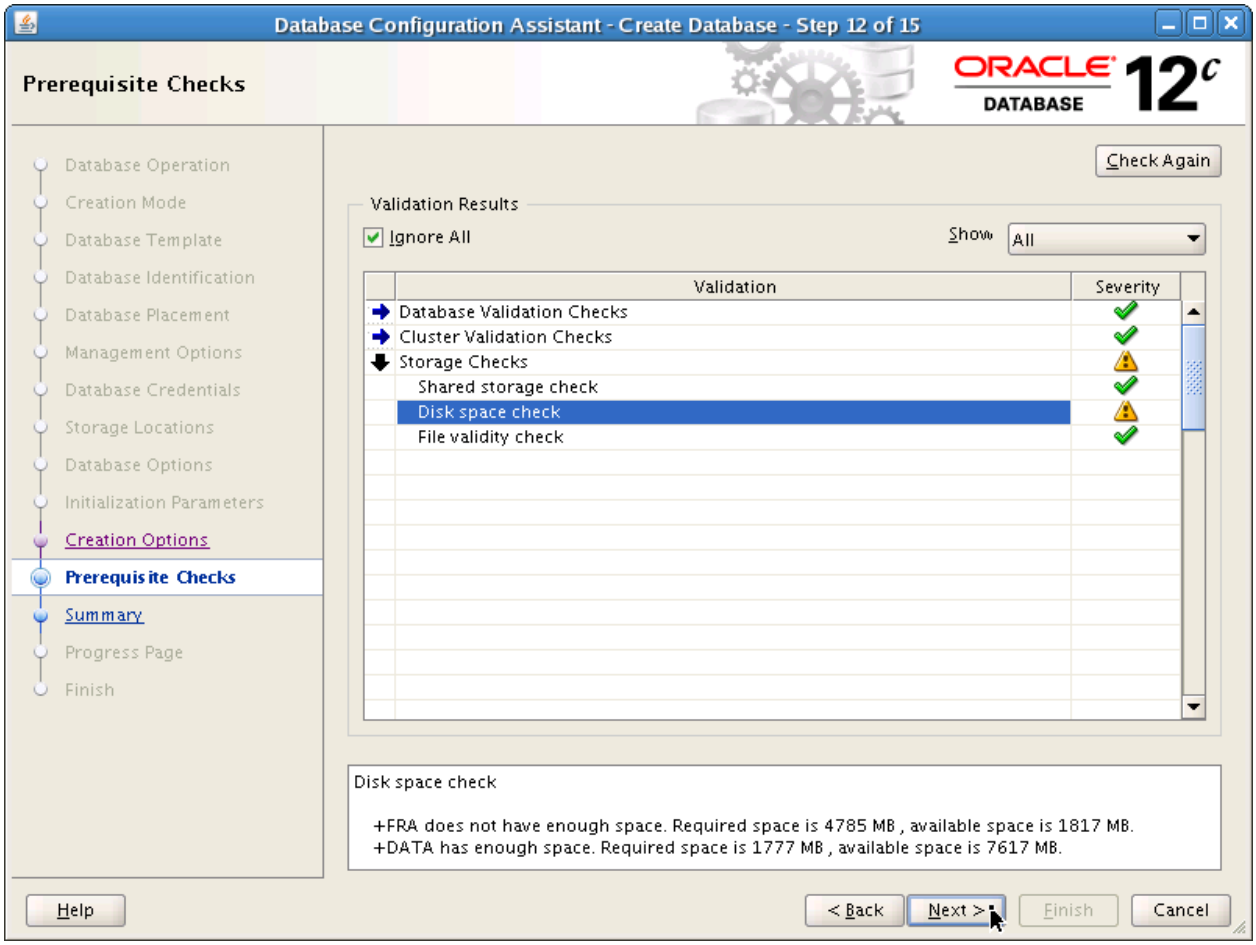
26. On the Creation Options page, accept the default option to create the database and click Next to continue.



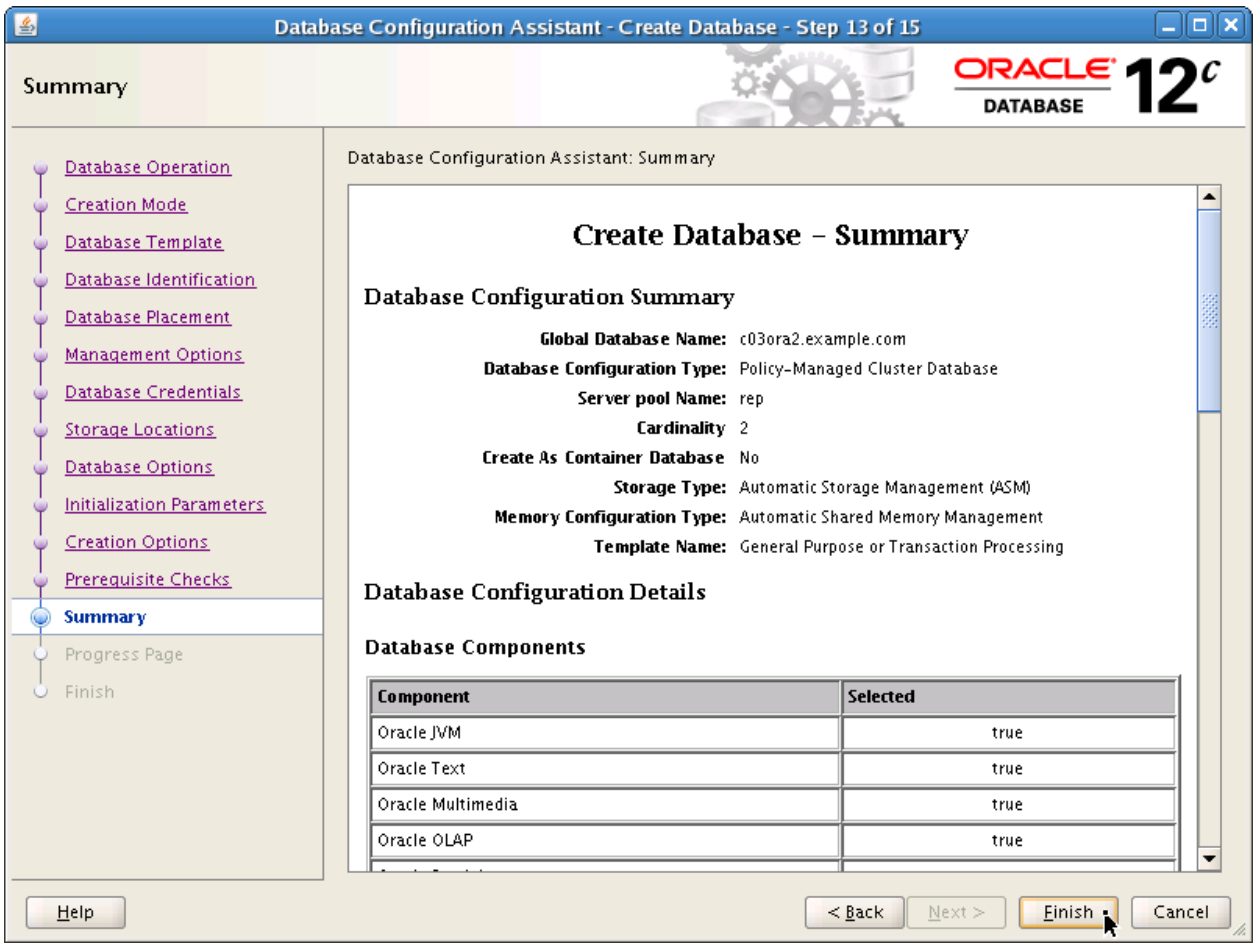
27. Wait while a series of prerequisite checks are performed.



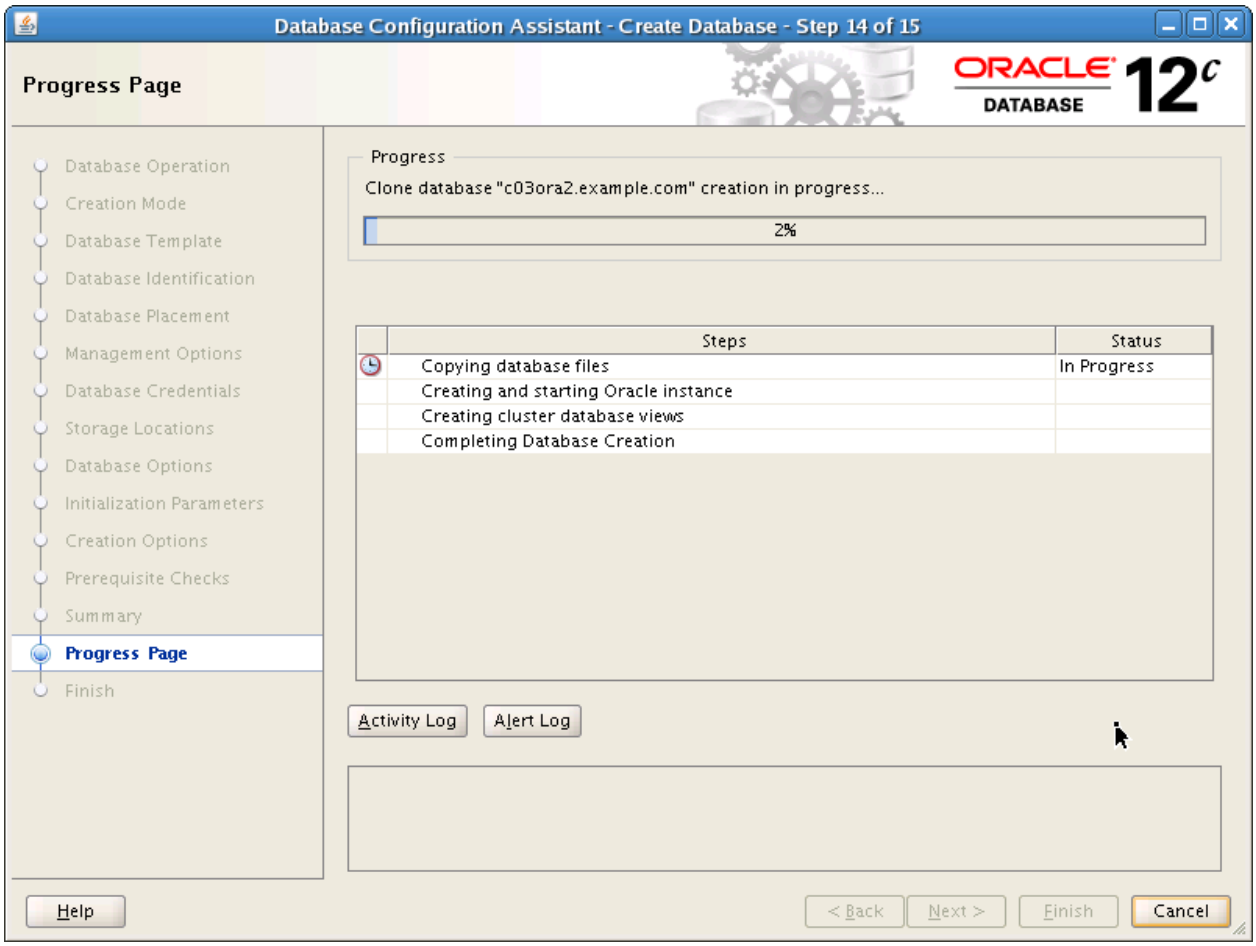
28. When the prerequisite checks are complete, examine the results and confirm that the only warning relates to the size of the `+FRA` disk group. Then, select the option to ignore all warnings and click Next to continue.



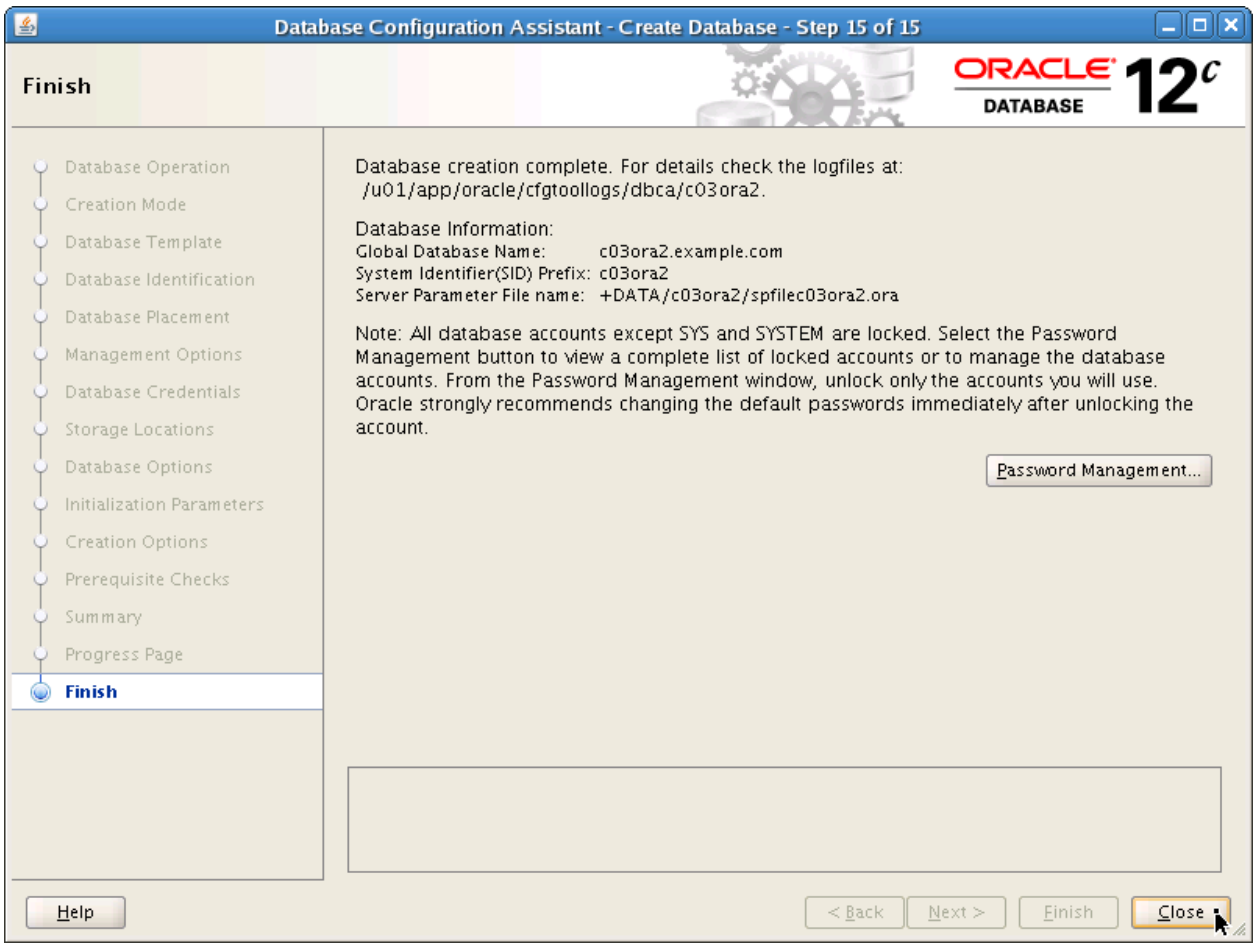
29. On the Summary page, examine the summary information. When you are ready, click Finish to start the database creation process.



30. Monitor the database creation process by viewing the Progress Page.



31. When the database creation process is finished, you will see the Finish page. Examine the information in this page and click Close to exit the Database Configuration Assistant.



32. Return to your `oracle` terminal session on `c03n01` and examine the status of your policy-managed databases.

```
[oracle@c03n01 ~]$ srvctl status database -thishome
Database unique name: c03ora2
Instance c03ora2_1 is running on node c03n02

Database unique name: c03orcl
Instance c03orcl_2 is running on node c03n01
[oracle@c03n01 ~]$
```

33. Re-examine the server pool status. Cross-reference the output from the previous step to confirm that database `c03orcl` is running in the `dev` pool, and database `c03ora2` is running in the `rep` pool.

```
[oracle@c03n01 ~]$ srvctl status srvpool -detail
Server pool name: Free
Active servers count: 0
Active server names:
Server pool name: Generic
```

```
Active servers count: 0
Active server names:
Server pool name: dev
Active servers count: 1
Active server names: c03n01
NAME=c03n01 STATE=ONLINE
Server pool name: rep
Active servers count: 1
Active server names: c03n02
NAME=c03n02 STATE=ONLINE
[oracle@c03n01 ~]$
```

Part 3: Using Policy-Based Cluster Management to Automatically Share Resources Between Oracle RAC Databases.

So far you have seen how the current server pool configuration governs the resource allocation to policy-managed databases.

In the next part of this practice, you will define a cluster policy set and examine how a cluster policy can be used to automatically share resources between two Oracle RAC databases. Also, you will see how quick and easy it is to switch between policies and how Oracle Clusterware manages resources when policy switches occur.

The capabilities that you will explore in this part of the practice form the foundation for effectively consolidating multiple Oracle RAC databases on a single cluster.

34. Establish a terminal session connected to `c03n01` as the `grid` user and configure the terminal environment as shown below.

```
$ ssh grid@c03n01
[grid@c03n01 ~]$ . oraenv
ORACLE_SID = [grid] ? +ASM1
The Oracle base has been set to /u01/app/grid
[grid@c03n01 ~]$
```

35. Examine the contents of the file at `/stage/labs/4_1/policy.txt`. This file contains the policy set definition that you will use in this practice. Notice that the policy set contains two policies, named `day` and `night`. In the `day` policy, the `dev` server pool has higher importance and a minimum server count of `2`. In the `night` policy, the `rep` server pool has higher importance, and a minimum server count of 1.

```
[grid@c03n01 ~]$ more /stage/labs/4_1/policy.txt
SERVER_POOL_NAMES=Free ora.dev ora.rep
POLICY
        NAME=day
        DESCRIPTION=The day policy
        SERVERPOOL
                NAME=ora.dev
                IMPORTANCE=10
                MAX_SIZE=2
```

```
                MIN_SIZE=2
        SERVERPOOL
                NAME=ora.rep
                IMPORTANCE=0
                MAX_SIZE=2
                MIN_SIZE=1
POLICY
        NAME=night
        DESCRIPTION=The night policy
        SERVERPOOL
                NAME=ora.dev
                IMPORTANCE=0
                MAX_SIZE=2
                MIN_SIZE=1
        SERVERPOOL
                NAME=ora.rep
                IMPORTANCE=5
                MAX_SIZE=2
                MIN_SIZE=1

[grid@c03n01 ~]$
```

36. Examine the cluster policy set. At this point, the policy set contains only the `Current` policy, which is a special built-in policy that reflects the current state of the cluster.

```
[grid@c03n01 ~]$ crsctl status policyset
ACL=owner:grid:rwx,pgrp:oinstall:rwx,other::r--
LAST_ACTIVATED_POLICY=
SERVER_POOL_NAMES=Free
POLICY
  NAME=Current
  DESCRIPTION=This policy is built-in and managed automatically
to reflect current configuration
  SERVERPOOL
    NAME=Free
    IMPORTANCE=0
    MAX_SIZE=-1
    MIN_SIZE=0
    SERVER_CATEGORY=
    SERVER_NAMES=
  SERVERPOOL
    NAME=Generic
    IMPORTANCE=0
    MAX_SIZE=-1
```

```
    MIN_SIZE=0
    SERVER_CATEGORY=
    SERVER_NAMES=
  SERVERPOOL
    NAME=ora.dev
    IMPORTANCE=0
    MAX_SIZE=2
    MIN_SIZE=1
    SERVER_CATEGORY=ora.hub.category
    SERVER_NAMES=
  SERVERPOOL
    NAME=ora.rep
    IMPORTANCE=0
    MAX_SIZE=2
    MIN_SIZE=1
    SERVER_CATEGORY=ora.hub.category
    SERVER_NAMES=
[grid@c03n01 ~]$
```

37. Modify the cluster policy set by using the definition you examined in `/stage/labs/policy.txt`.

```
[grid@c03n01 ~]$ crsctl modify policyset -file
/stage/labs/4_1/policy.txt
[grid@c03n01 ~]$
```

38. Re-examine the cluster policy set and confirm the changes made in the previous step. Notice that at this point, the `LAST_ACTIVATED_POLICY` attribute is empty and the `Current` policy does not reflect either the `day` policy or the `night` policy.

```
[grid@c03n01 ~]$ crsctl status policyset
ACL=owner:grid:rwx,pgrp:oinstall:rwx,other::r--
LAST_ACTIVATED_POLICY=
SERVER_POOL_NAMES=Free ora.dev ora.rep
POLICY
  NAME=Current
  DESCRIPTION=This policy is built-in and managed automatically
to reflect current configuration
  SERVERPOOL
    NAME=Free
    IMPORTANCE=0
    MAX_SIZE=-1
    MIN_SIZE=0
    SERVER_CATEGORY=
    SERVER_NAMES=
  SERVERPOOL
```

```
    NAME=Generic
    IMPORTANCE=0
    MAX_SIZE=-1
    MIN_SIZE=0
    SERVER_CATEGORY=
    SERVER_NAMES=
  SERVERPOOL
    NAME=ora.dev
    IMPORTANCE=0
    MAX_SIZE=2
    MIN_SIZE=1
    SERVER_CATEGORY=ora.hub.category
    SERVER_NAMES=
  SERVERPOOL
    NAME=ora.rep
    IMPORTANCE=0
    MAX_SIZE=2
    MIN_SIZE=1
    SERVER_CATEGORY=ora.hub.category
    SERVER_NAMES=
POLICY
  NAME=day
  DESCRIPTION=The day policy
  SERVERPOOL
    NAME=Free
    IMPORTANCE=0
    MAX_SIZE=-1
    MIN_SIZE=0
    SERVER_CATEGORY=
    SERVER_NAMES=
  SERVERPOOL
    NAME=ora.dev
    IMPORTANCE=10
    MAX_SIZE=2
    MIN_SIZE=2
    SERVER_CATEGORY=ora.hub.category
    SERVER_NAMES=
  SERVERPOOL
    NAME=ora.rep
    IMPORTANCE=0
    MAX_SIZE=2
    MIN_SIZE=1
```

```
    SERVER_CATEGORY=ora.hub.category
    SERVER_NAMES=
POLICY
  NAME=night
  DESCRIPTION=The night policy
  SERVERPOOL
    NAME=Free
    IMPORTANCE=0
    MAX_SIZE=-1
    MIN_SIZE=0
    SERVER_CATEGORY=
    SERVER_NAMES=
  SERVERPOOL
    NAME=ora.dev
    IMPORTANCE=0
    MAX_SIZE=2
    MIN_SIZE=1
    SERVER_CATEGORY=ora.hub.category
    SERVER_NAMES=
  SERVERPOOL
    NAME=ora.rep
    IMPORTANCE=5
    MAX_SIZE=2
    MIN_SIZE=1
    SERVER_CATEGORY=ora.hub.category
    SERVER_NAMES=
[grid@c03n01 ~]$
```

39. Activate the `day` policy. Make sure that you use the force option (`-f`). Notice how different database resources are automatically stopped and started in response to the policy change.

```
[grid@c03n01 ~]$ crsctl modify policyset -attr
"LAST_ACTIVATED_POLICY='day'" -f
CRS-2673: Attempting to stop 'ora.c03ora2.db' on 'c03n02'
CRS-2677: Stop of 'ora.c03ora2.db' on 'c03n02' succeeded
CRS-2672: Attempting to start 'ora.c03orcl.db' on 'c03n02'
CRS-2676: Start of 'ora.c03orcl.db' on 'c03n02' succeeded
[grid@c03n01 ~]$
```

40. Re-examine the cluster policy set. Confirm that `LAST_ACTIVATED_POLICY=day` and that the `Current` policy reflects the `day` policy.

```
[grid@c03n01 ~]$ crsctl status policyset
ACL=owner:grid:rwx,pgrp:oinstall:rwx,other::r--
LAST_ACTIVATED_POLICY=day
```

```
SERVER_POOL_NAMES=Free ora.dev ora.rep
POLICY
  NAME=Current
  DESCRIPTION=This policy is built-in and managed automatically
to reflect current configuration
  SERVERPOOL
    NAME=Free
    IMPORTANCE=0
    MAX_SIZE=-1
    MIN_SIZE=0
    SERVER_CATEGORY=
    SERVER_NAMES=
  SERVERPOOL
    NAME=Generic
    IMPORTANCE=0
    MAX_SIZE=-1
    MIN_SIZE=0
    SERVER_CATEGORY=
    SERVER_NAMES=
  SERVERPOOL
    NAME=ora.dev
    IMPORTANCE=10
    MAX_SIZE=2
    MIN_SIZE=2
    SERVER_CATEGORY=ora.hub.category
    SERVER_NAMES=
  SERVERPOOL
    NAME=ora.rep
    IMPORTANCE=0
    MAX_SIZE=2
    MIN_SIZE=1
    SERVER_CATEGORY=ora.hub.category
    SERVER_NAMES=
POLICY
  NAME=day
  DESCRIPTION=The day policy
  SERVERPOOL
    NAME=Free
    IMPORTANCE=0
    MAX_SIZE=-1
    MIN_SIZE=0
    SERVER_CATEGORY=
```

```
    SERVER_NAMES=
  SERVERPOOL
    NAME=ora.dev
    IMPORTANCE=10
    MAX_SIZE=2
    MIN_SIZE=2
    SERVER_CATEGORY=ora.hub.category
    SERVER_NAMES=
  SERVERPOOL
    NAME=ora.rep
    IMPORTANCE=0
    MAX_SIZE=2
    MIN_SIZE=1
    SERVER_CATEGORY=ora.hub.category
    SERVER_NAMES=
POLICY
  NAME=night
  DESCRIPTION=The night policy
  SERVERPOOL
    NAME=Free
    IMPORTANCE=0
    MAX_SIZE=-1
    MIN_SIZE=0
    SERVER_CATEGORY=
    SERVER_NAMES=
  SERVERPOOL
    NAME=ora.dev
    IMPORTANCE=0
    MAX_SIZE=2
    MIN_SIZE=1
    SERVER_CATEGORY=ora.hub.category
    SERVER_NAMES=
  SERVERPOOL
    NAME=ora.rep
    IMPORTANCE=5
    MAX_SIZE=2
    MIN_SIZE=1
    SERVER_CATEGORY=ora.hub.category
    SERVER_NAMES=
[grid@c03n01 ~]$
```

41. Return to your `oracle` terminal session on `c03n01` and examine the server pool status. Confirm that two nodes are allocated to the `dev` pool.

```
[oracle@c03n01 ~]$ srvctl status srvpool -detail
Server pool name: Free
Active servers count: 0
Active server names:
Server pool name: Generic
Active servers count: 0
Active server names:
Server pool name: dev
Active servers count: 2
Active server names: c03n01,c03n02
NAME=c03n01 STATE=ONLINE
NAME=c03n02 STATE=ONLINE
Server pool name: rep
Active servers count: 0
Active server names:
[oracle@c03n01 ~]$
```

42. Examine the current database status and confirm that database `c03orcl` is running on all the `dev` pool servers.

```
[oracle@c03n01 ~]$ srvctl status database -thishome
Database unique name: c03ora2
Database is not running.

Database unique name: c03orcl
Instance c03orcl_1 is running on node c03n02
Instance c03orcl_2 is running on node c03n01
[oracle@c03n01 ~]$
```

43. Switch back to your `grid` terminal session and activate the `night` policy. Remember the force option (`-f`) and notice the automatic database operations that accompany the policy switch.

```
[grid@c03n01 ~]$ crsctl modify policyset -attr
"LAST_ACTIVATED_POLICY='night'" -f
CRS-2673: Attempting to stop 'ora.c03orcl.db' on 'c03n02'
CRS-2677: Stop of 'ora.c03orcl.db' on 'c03n02' succeeded
CRS-2672: Attempting to start 'ora.c03ora2.db' on 'c03n02'
CRS-2676: Start of 'ora.c03ora2.db' on 'c03n02' succeeded
[grid@c03n01 ~]$
```

44. Using your `oracle` terminal session, examine the server pool status. Confirm that the `dev` pool contains 1 server and that the `rep` pool contains 1 server.

```
[oracle@c03n01 ~]$ srvctl status srvpool -detail
Server pool name: Free
Active servers count: 0
Active server names:
Server pool name: Generic
Active servers count: 0
Active server names:
Server pool name: dev
Active servers count: 1
Active server names: c03n01
NAME=c03n01 STATE=ONLINE
Server pool name: rep
Active servers count: 1
Active server names: c03n02
NAME=c03n02 STATE=ONLINE
[oracle@c03n01 ~]$
```

45. Examine the database status. Cross-reference the output from the previous step to confirm that database `c03orcl` is running in the `dev` pool, and database `c03ora2` is running in the `rep` pool.

```
[oracle@c03n01 ~]$ srvctl status database -thishome
Database unique name: c03ora2
Instance c03ora2_1 is running on node c03n02

Database unique name: c03orcl
Instance c03orcl_2 is running on node c03n01
[oracle@c03n01 ~]$
```

So far you have seen how the cluster automatically responds to policy changes. In the next part of this practice, you will also see how a policy-managed cluster responds to node outages.

46. Using your `grid` terminal session, shut down Oracle Clusterware on the `rep` pool server. Use the information from the previous steps to determine which node to stop. Note that your environment may differ from the example below.

```
[grid@c03n01 ~]$ su -c "crsctl stop cluster -n c03n02 -f"
Password: <oracle>
CRS-2673: Attempting to stop 'ora.crsd' on 'c03n02'
CRS-2790: Starting shutdown of Cluster Ready Services-managed
resources on 'c03n02'
CRS-2673: Attempting to stop 'ora.DATA.dg' on 'c03n02'
CRS-2673: Attempting to stop 'ora.c03ora2.db' on 'c03n02'
...
```

```
CRS-2677: Stop of 'ora.asm' on 'c03n02' succeeded
CRS-2673: Attempting to stop 'ora.cluster_interconnect.haip' on
'c03n02'
CRS-2677: Stop of 'ora.cluster_interconnect.haip' on 'c03n02'
succeeded
CRS-2673: Attempting to stop 'ora.cssd' on 'c03n02'
CRS-2677: Stop of 'ora.cssd' on 'c03n02' succeeded
[grid@c03n01 ~]$
```

47. Using your `oracle` terminal session, examine the server pool status. Notice that the remaining server (`c03n01` in the example below) has automatically moved from the `dev` pool to the `rep` pool. This is because the `rep` pool has a higher importance than the `dev` pool in the current policy (the `night` policy).

    Notes:

    You may need to wait for a short period while the cluster is reconfigured. If required, periodically re-examine the server pool status until you see that the remaining node is online in the `rep` pool.

    In the previous step, if you stopped the cluster on `c03n01`, you will need to establish a new terminal session on `c03n02` to execute this step and the next step. If this is the case, remember to use the `oraenv` script to configure the terminal environment.

```
[oracle@c03n01 ~]$ srvctl status srvpool -detail
Server pool name: Free
Active servers count: 0
Active server names:
Server pool name: Generic
Active servers count: 0
Active server names:
Server pool name: dev
Active servers count: 0
Active server names:
Server pool name: rep
Active servers count: 1
Active server names: c03n01
NAME=c03n01 STATE=ONLINE
[oracle@c03n01 ~]$
```

48. Examine the database status and confirm that `c03ora2` is running in the `rep` pool.

    Note: You may need to wait for a short period while the database instances are reconfigured. If required, periodically re-examine the database status until you see that the remaining node is hosting an instance of the `c03ora2` database, which is associated with the `rep` server pool.

```
[oracle@c03n01 ~]$ srvctl status database -thishome
Database unique name: c03ora2
Instance c03ora2_1 is running on node c03n01

Database unique name: c03orcl
Database is not running.
[oracle@c03n01 ~]$
```

49. Using your `grid` terminal session, restart the cluster on the node that you previously shut down.

```
[grid@c03n01 ~]$ su -c "crsctl start cluster -n c03n02"
Password: <oracle>
CRS-2672: Attempting to start 'ora.evmd' on 'c03n02'
CRS-2672: Attempting to start 'ora.cssdmonitor' on 'c03n02'
CRS-2676: Start of 'ora.cssdmonitor' on 'c03n02' succeeded
...
CRS-2676: Start of 'ora.storage' on 'c03n02' succeeded
CRS-2672: Attempting to start 'ora.crsd' on 'c03n02'
CRS-2676: Start of 'ora.crsd' on 'c03n02' succeeded
[grid@c03n01 ~]$
```

50. Return to your `oracle` terminal session on `c03n01` and examine the server pool status. Notice that the newly restarted server is allocated to the `dev` server pool.

```
[oracle@c03n01 ~]$ srvctl status srvpool -detail
Server pool name: Free
Active servers count: 0
Active server names:
Server pool name: Generic
Active servers count: 0
Active server names:
Server pool name: dev
Active servers count: 1
Active server names: c03n02
NAME=c03n02 STATE=ONLINE
Server pool name: rep
Active servers count: 1
Active server names: c03n01
NAME=c03n01 STATE=ONLINE
[oracle@c03n01 ~]$
```

51. Examine the database status. Cross-reference the output from the previous step to confirm that database `c03orcl` is running in the `dev` pool, and database `c03ora2` is running in the `rep` pool.

    Note: You may need to wait for a short period while the `c03orcl` database starts. If required, periodically re-examine the database status until you see that both databases are running.

```
[oracle@c03n01 ~]$ srvctl status database -thishome
Database unique name: c03ora2
Instance c03ora2_1 is running on node c03n01

Database unique name: c03orcl
Instance c03orcl_1 is running on node c03n02
[oracle@c03n01 ~]$
```

52. In preparation for the remainder of the practice, shut down all of the databases (`c03orcl` and `c03ora2`).

```
[oracle@c03n01 ~]$ srvctl stop database -d c03orcl
[oracle@c03n01 ~]$ srvctl stop database -d c03ora2
[oracle@c03n01 ~]$
```

Part 4: Creating a New Policy-Managed Database with Oracle Multitenant Capabilities.

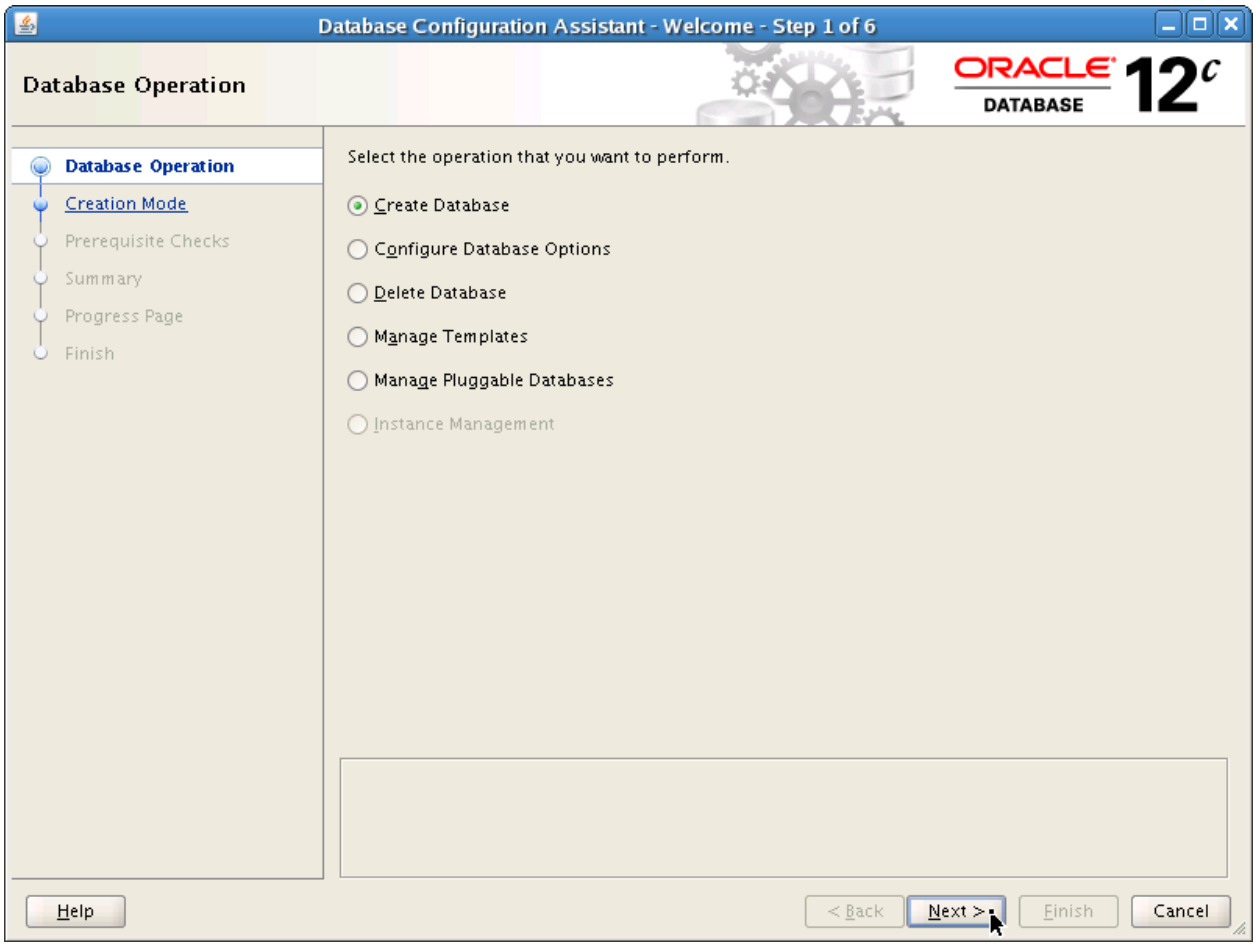
You have just seen how policy-based cluster management can be used to share resources between multiple Oracle RAC databases.

In the next part of this practice, you will create a new policy-based container database with two pluggable databases.

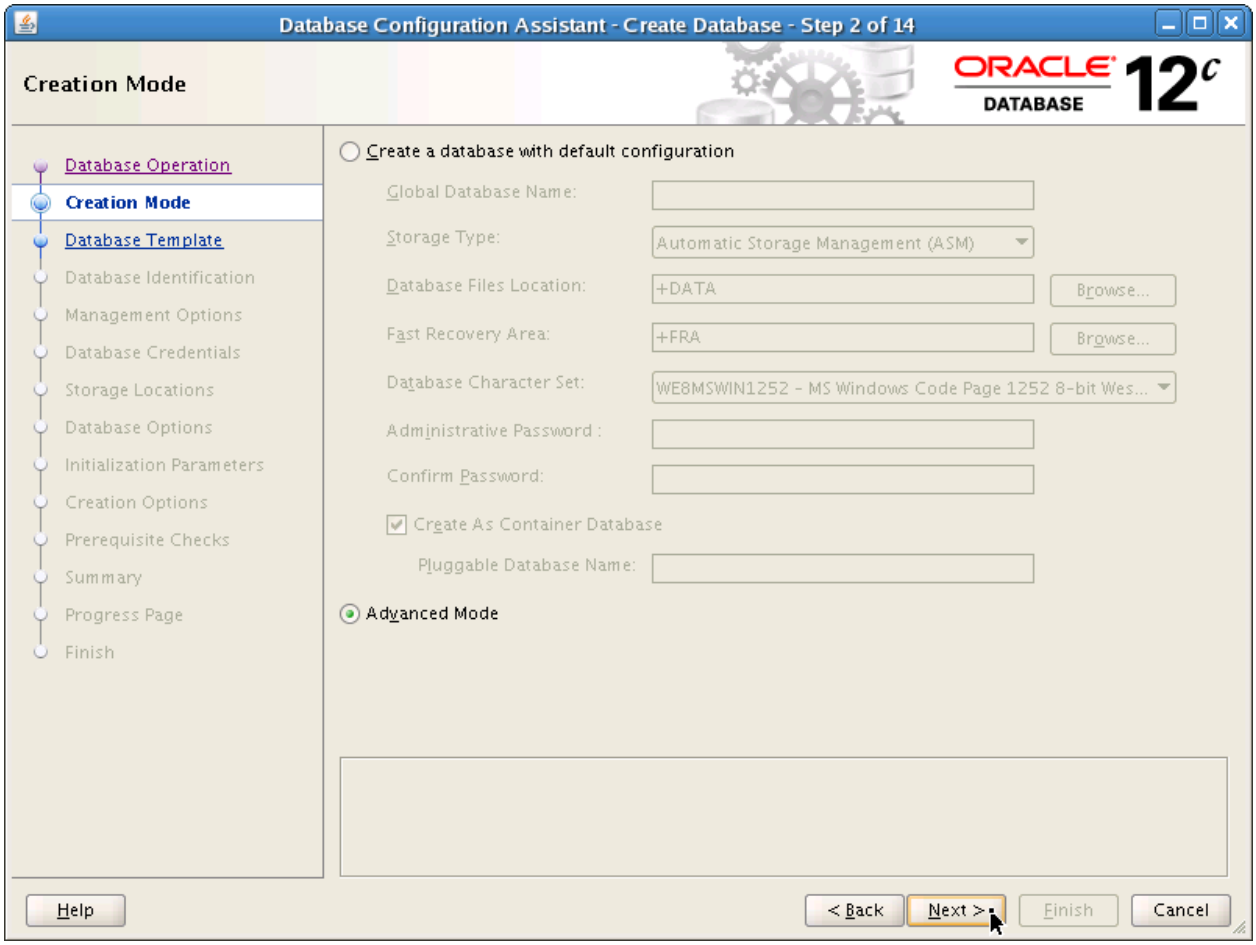
53. Launch the Oracle Database Configuration Assistant (DBCA).

```
[oracle@c03n01 ~]$ dbca &
[1] 27332
[oracle@c03n01 ~]$
```

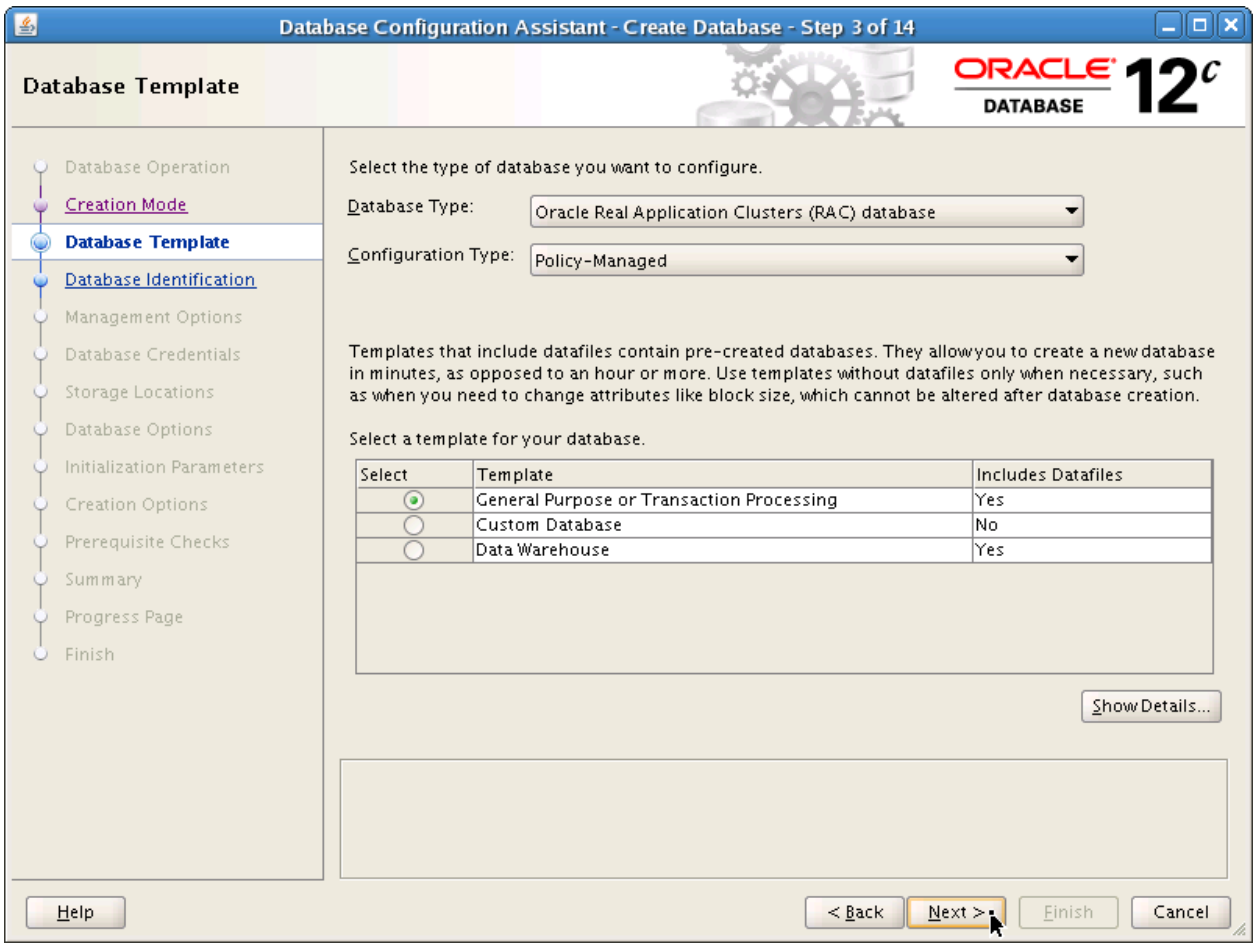
54. On the Database Operation page, accept the default option to create a new database and click Next to continue.



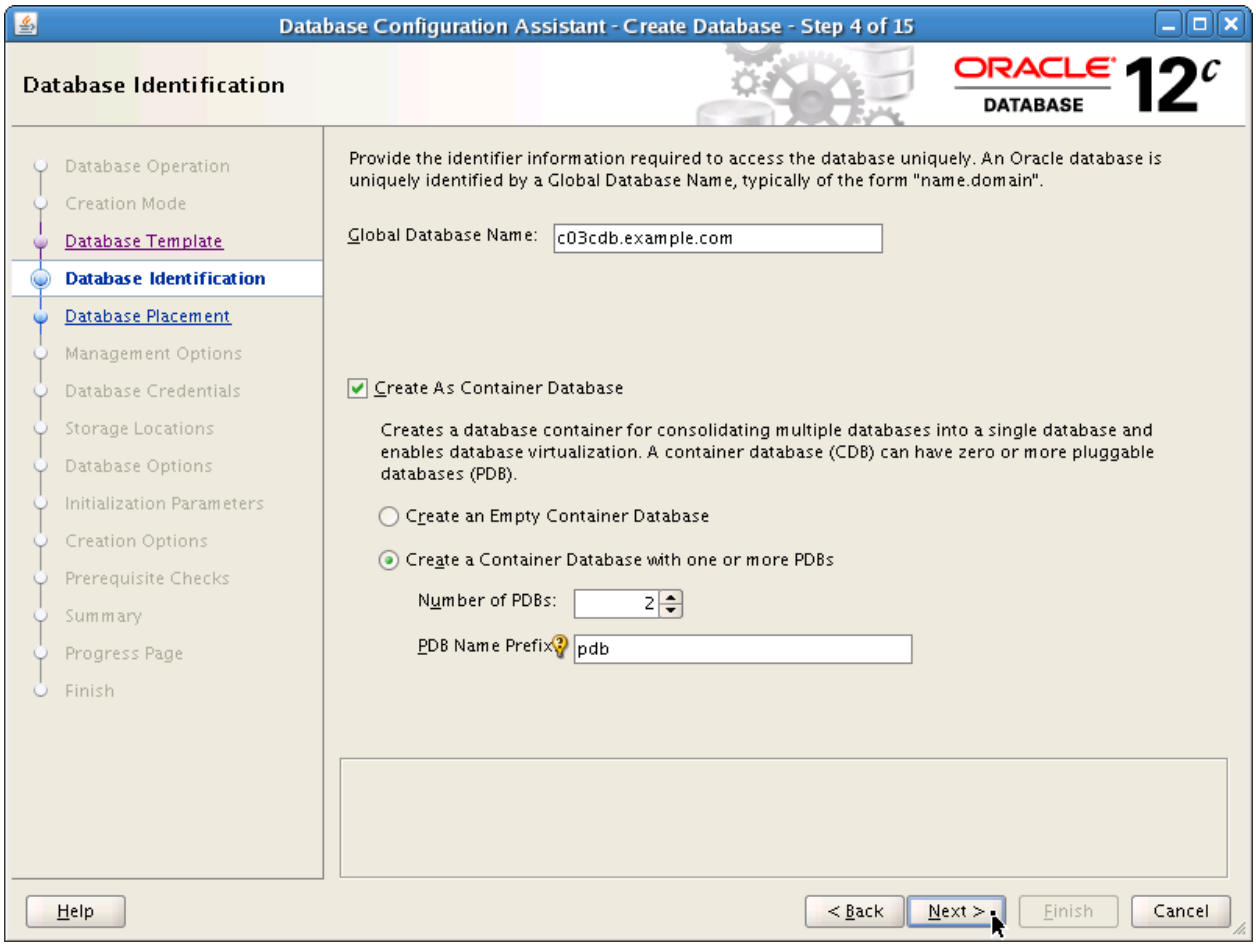
55. On the Creation Mode page, accept the default option to use advanced mode and click Next to continue.



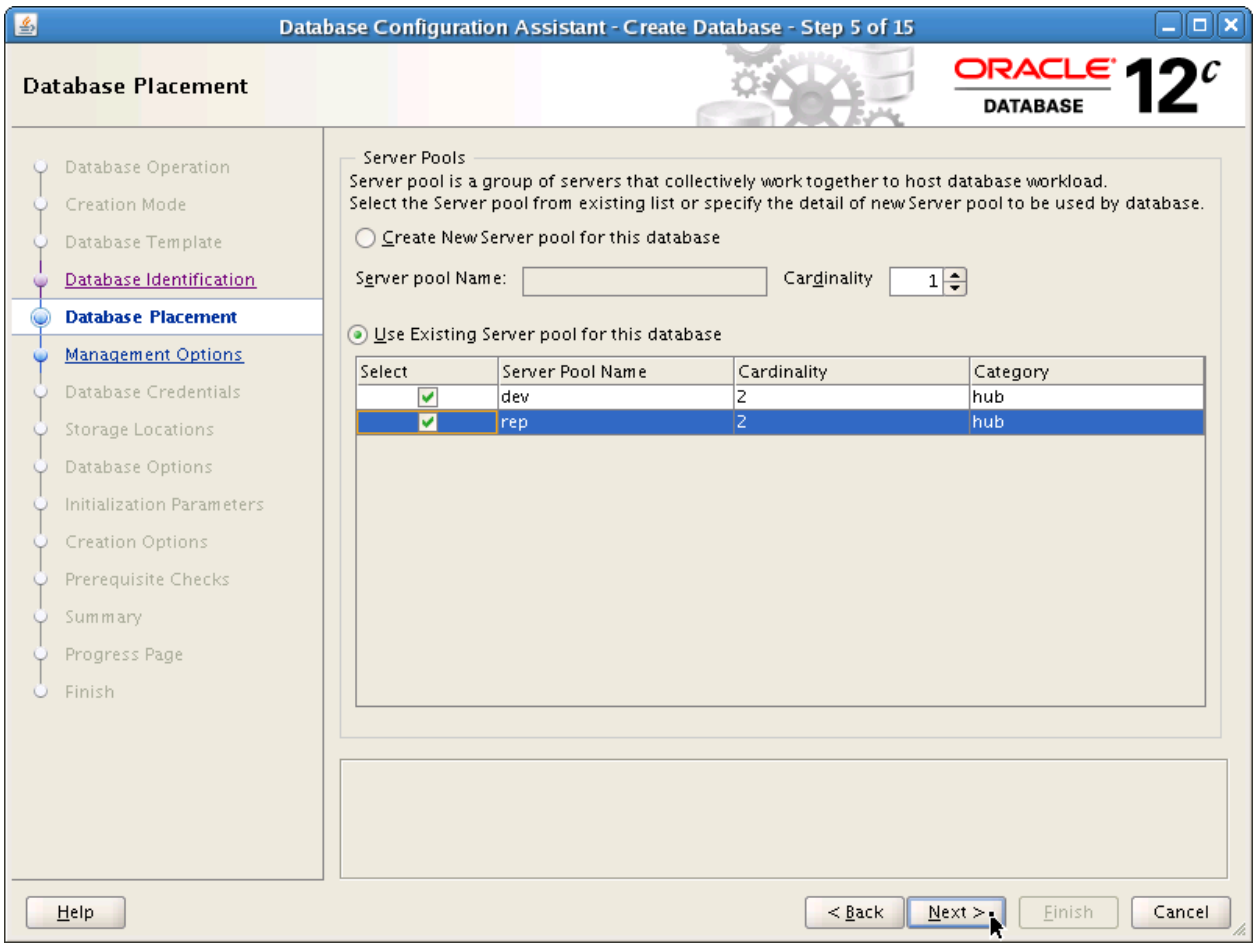
56. On the Database Template page, accept all of the default options and click Next to continue.



57. On the Database Identification page, specify `c03cdb.example.com` as the Global Database Name. Also, select the option to create a container database and select the option to create one or more pluggable databases. Specify `2` for the number of pluggable databases and specify `pdb` as the pluggable database prefix. Finally, click Next to continue.

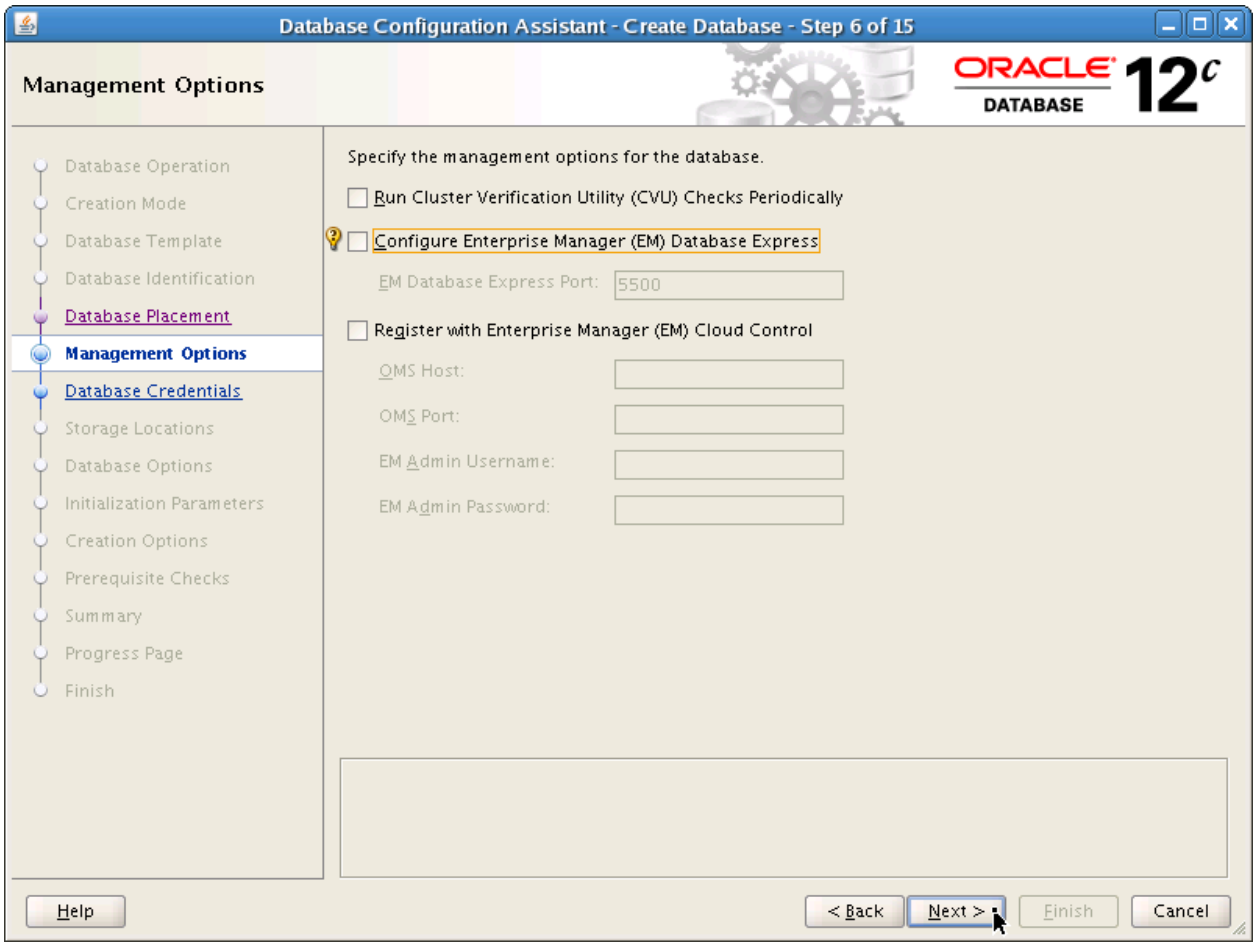


58. On the Database Placement page, select the option to use an existing server pool and select both the `dev` and `rep` server pools. Then, click Next to continue.

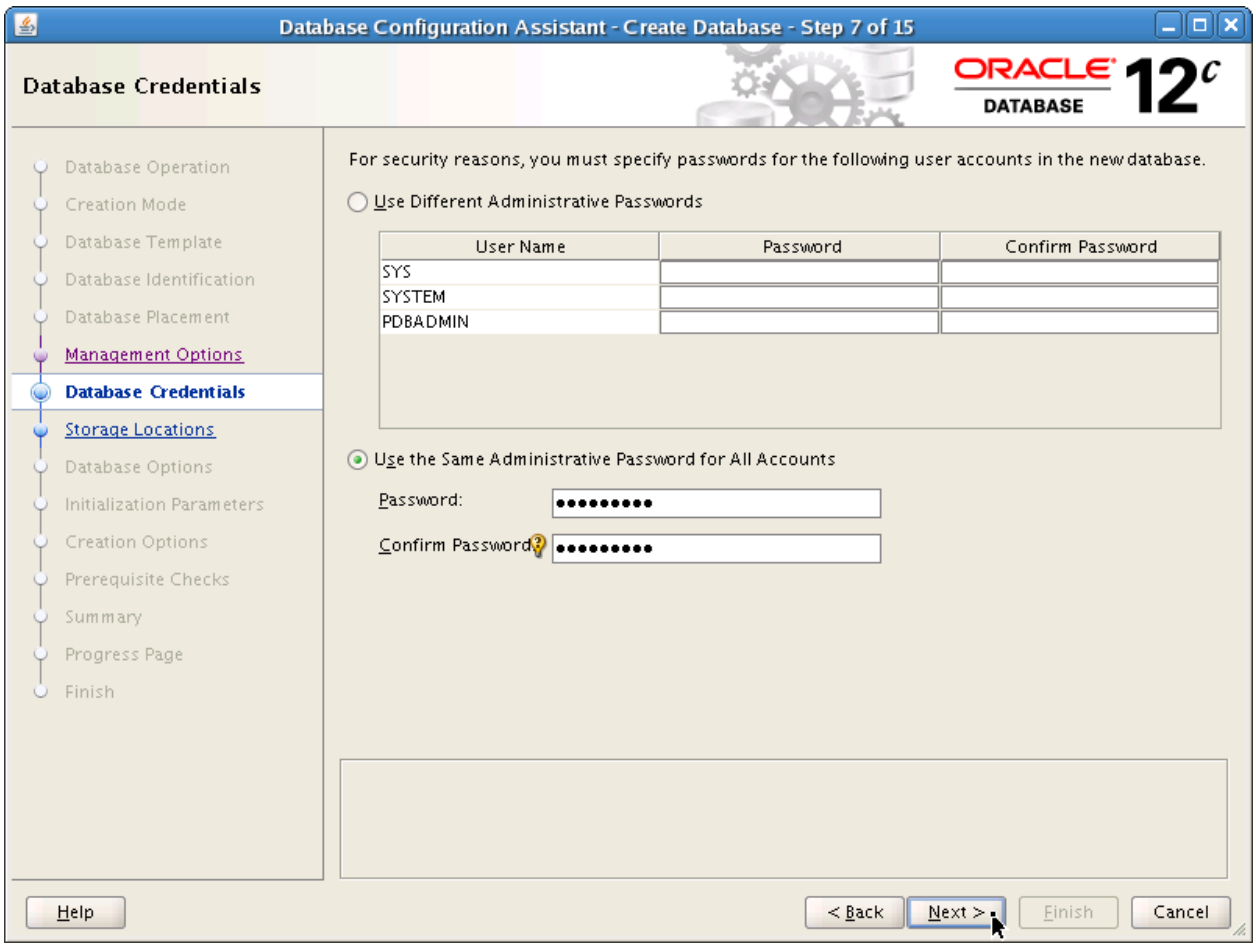


59. On the Management Options page, deselect all of the selected options and click Next to continue.

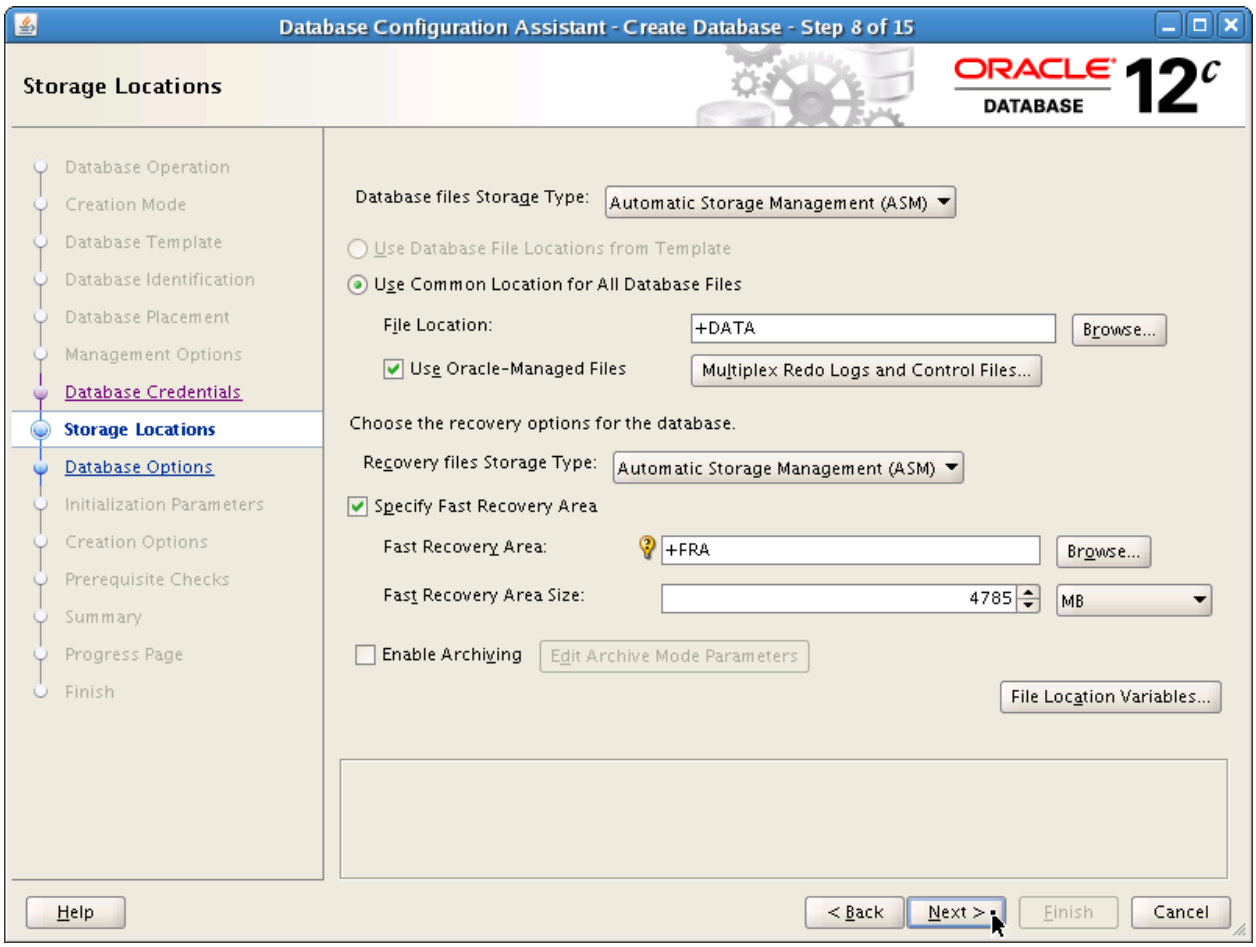
    Note: This configuration is not required or generally recommended and is being used only because none of the management options are required later in the practice.



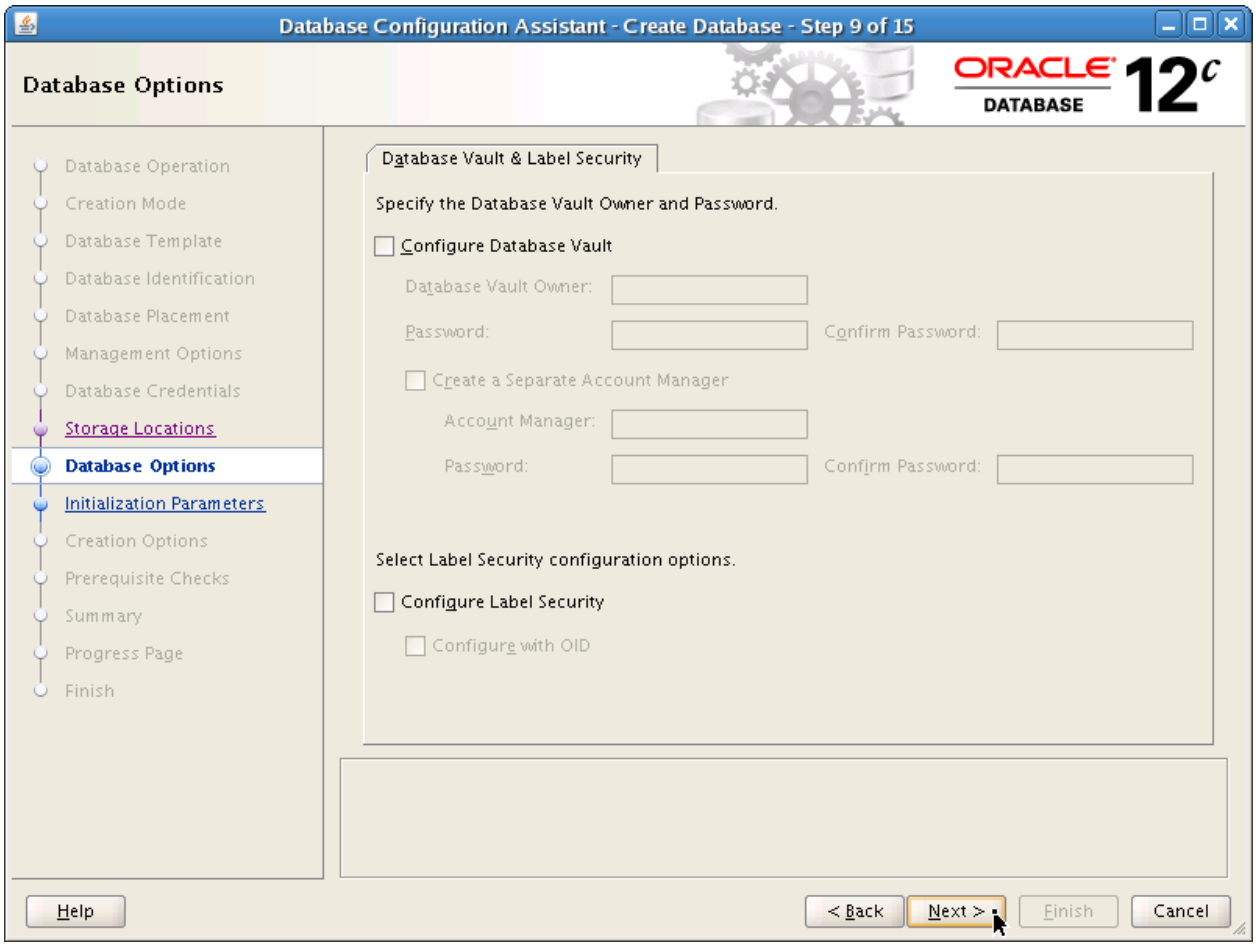
60. On the Database Credentials page, select the option to use the same password for all administrative accounts. Specify `oracle_4U` as the password and click Next to continue.



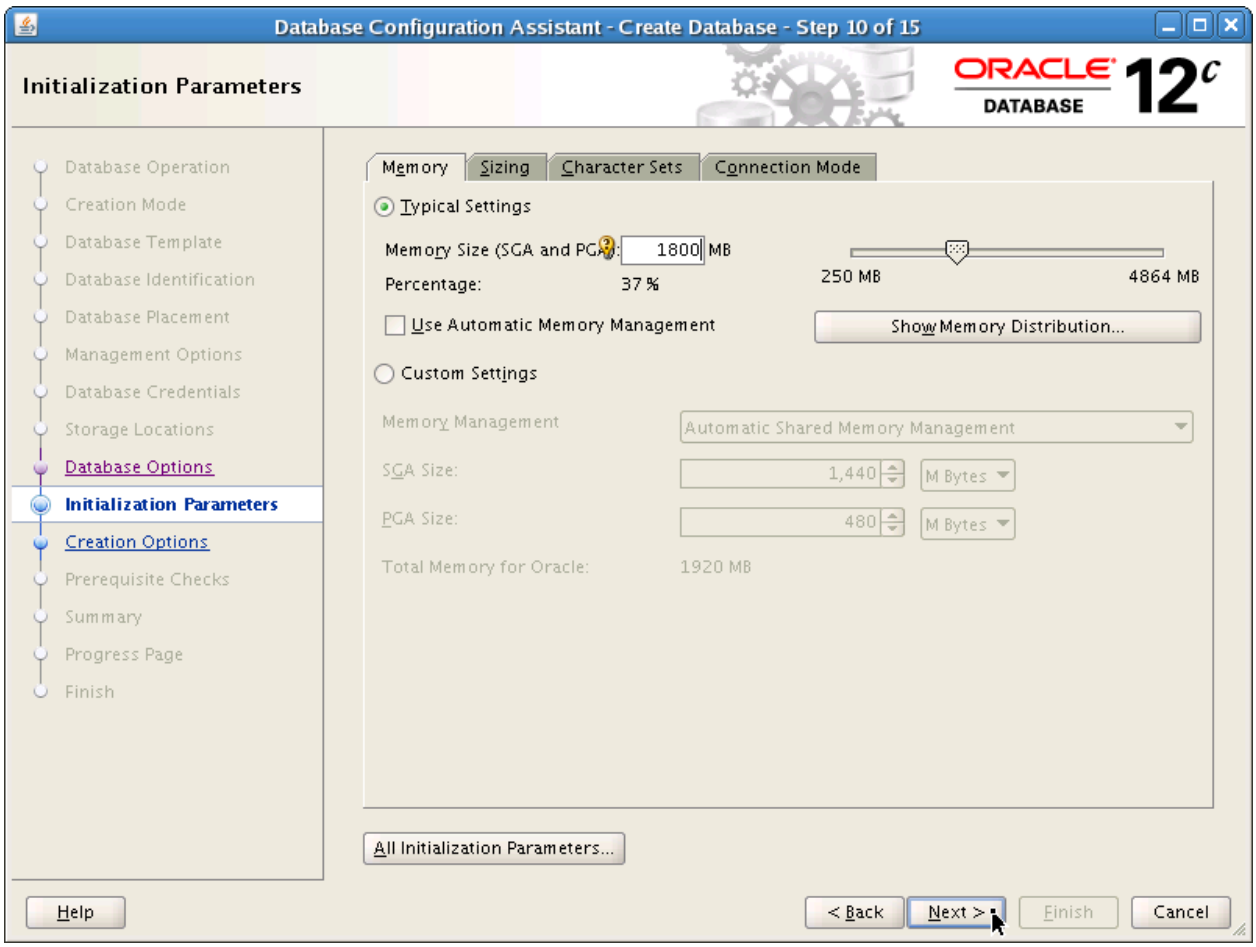
61. On the Storage Locations page, specify `+FRA` as the Fast Recovery Area location. Accept all of the other defaults and click Next to continue.



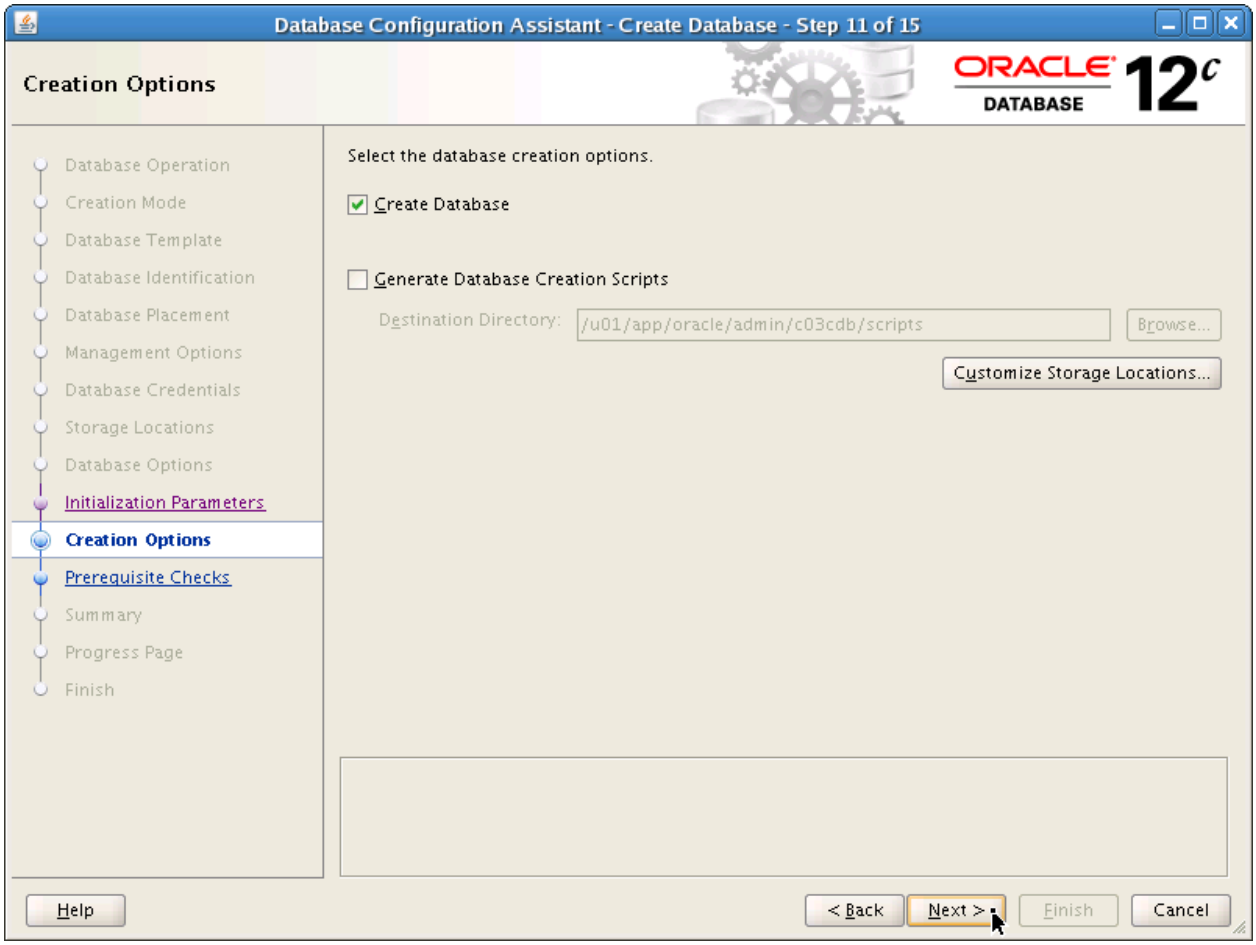
62. On the Database Options page, accept all of the default options and click Next to continue.



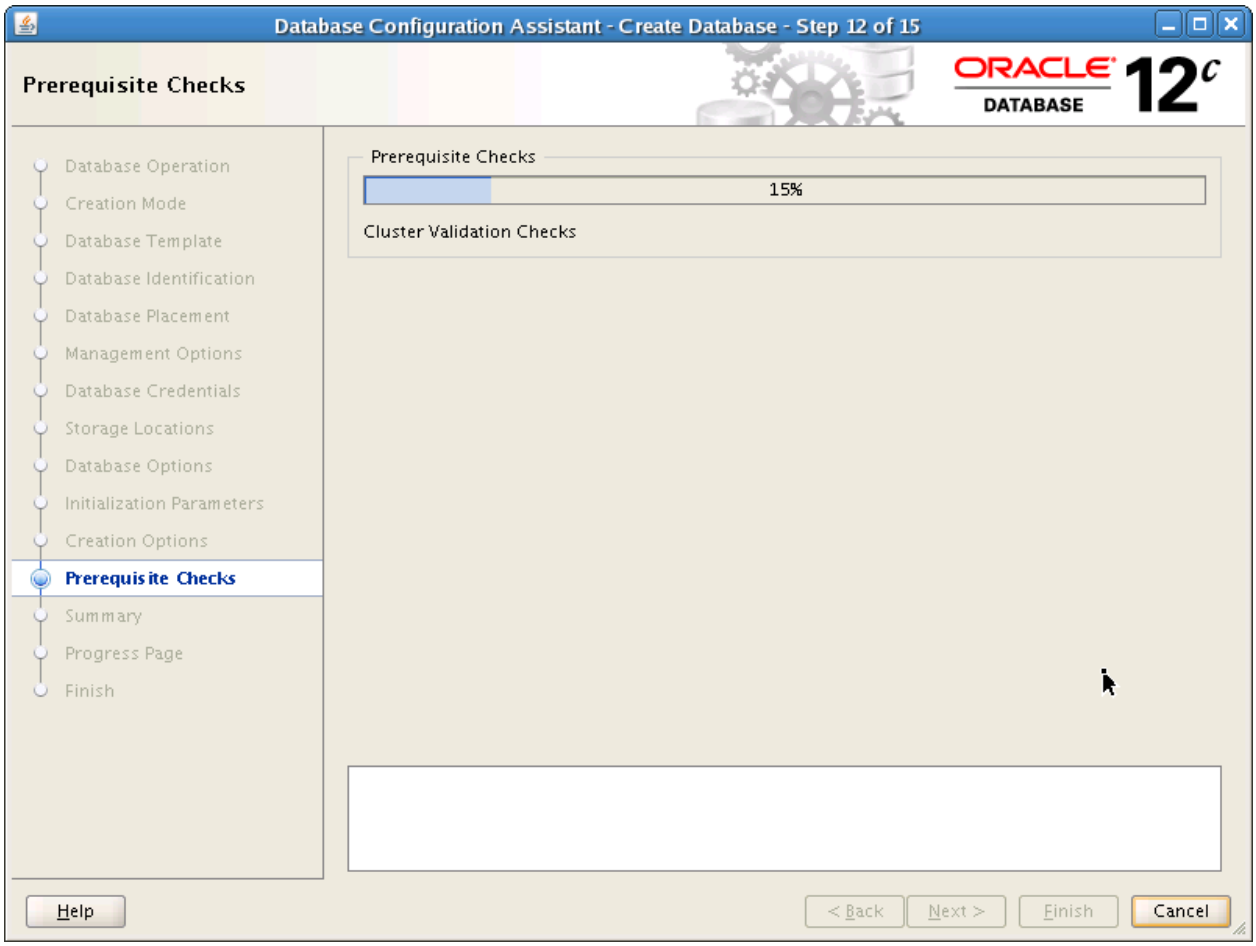
63. On the Initialization Parameters page, specify `1800MB` as the database memory size. Accept all of the other default options and click Next to continue.



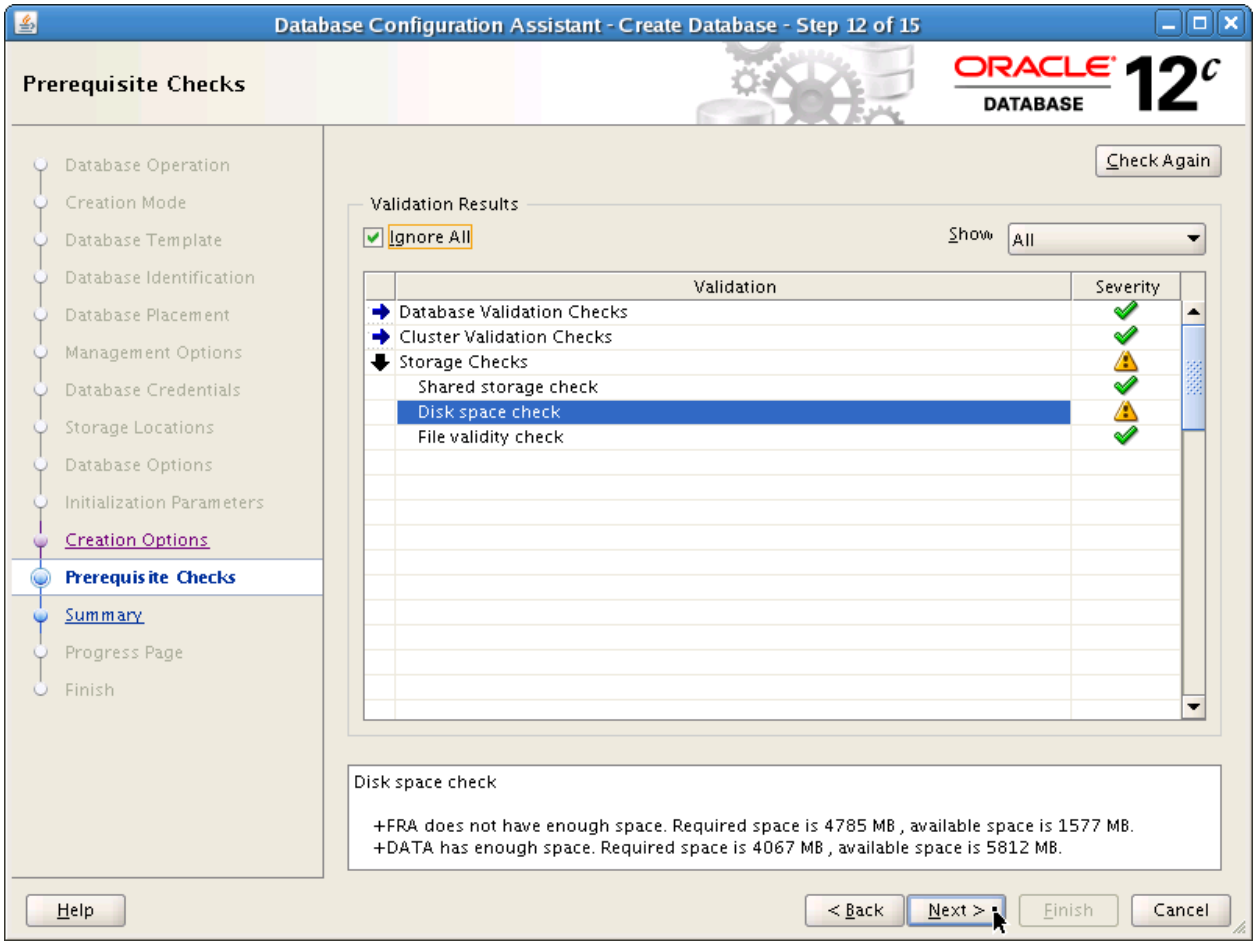
64. On the Creation Options page, accept the default option to create the database and click Next to continue.



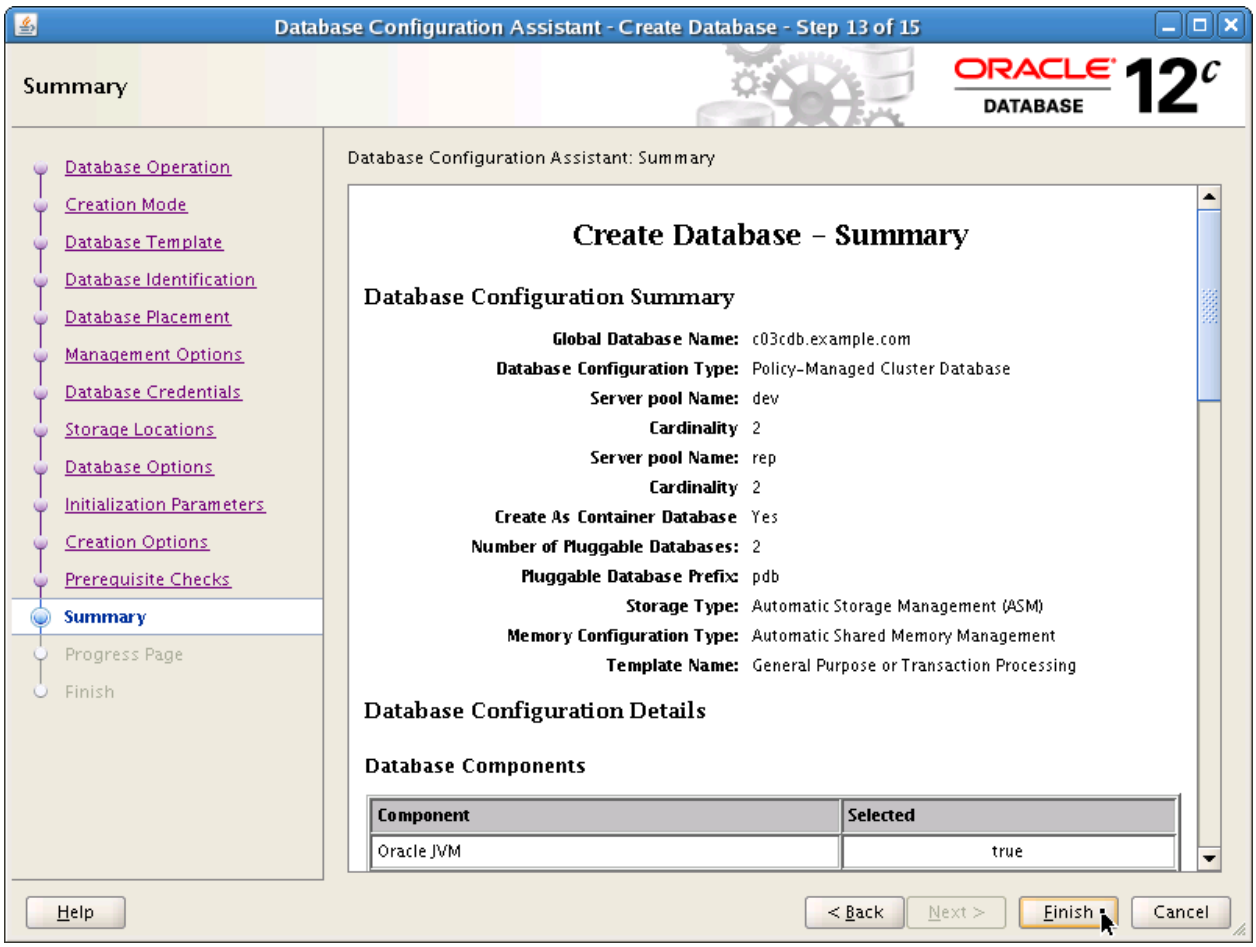
65. Wait while a series of prerequisite checks are performed.



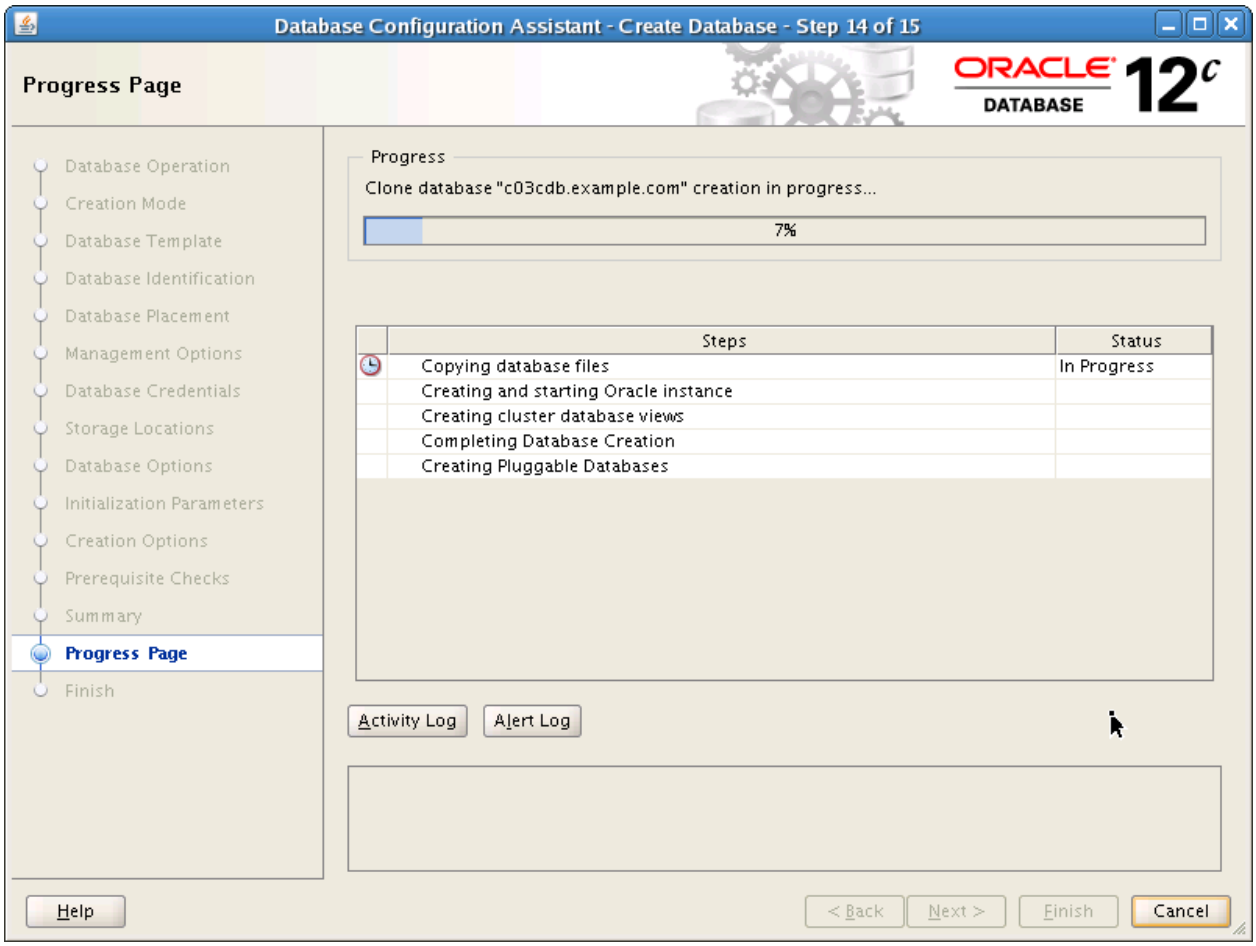
66. When the prerequisite checks complete, examine the results and confirm that the only warning relates to the size of the `+FRA` disk group. Then, select the option to ignore all warnings and click Next to continue.



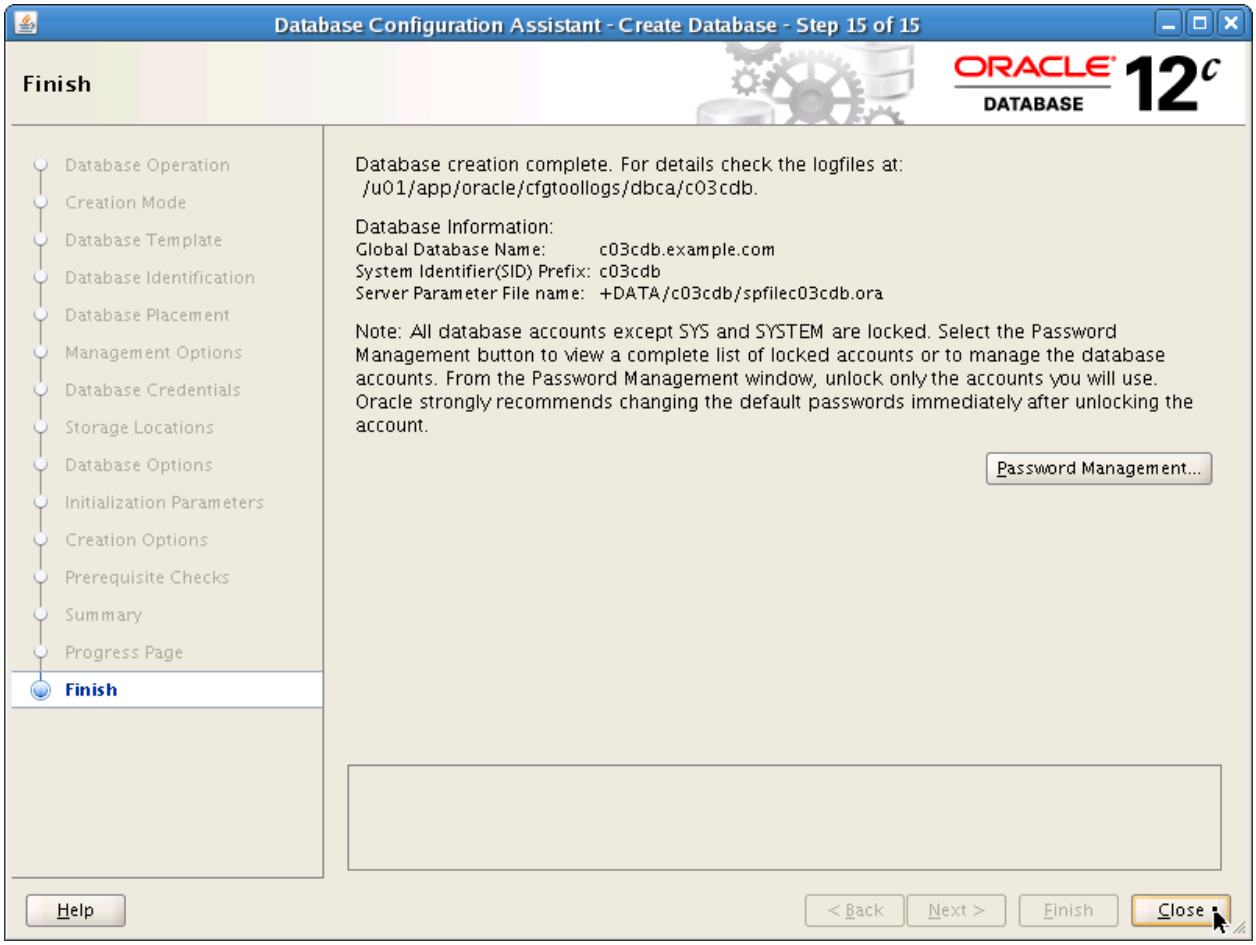
67. On the Summary page, examine the summary information. When you are ready, click Finish to start the database creation process.



68. Monitor the database creation process by viewing the Progress Page.



69. When the database creation process is finished, you will see the Finish page. Examine the information in this page and click Close to exit the Database Configuration Assistant.



70. Return to your `oracle` terminal session on `c03n01` and examine the status of your newly created database. Remember that you earlier configured the database to use both the `dev` and `rep` server pools. Because of this, the database will run across both cluster nodes when the `day` policy is enabled and also when the `night` policy is enabled.

```
[oracle@c03n01 ~]$ srvctl status database -d c03cdb
Instance c03cdb_1 is running on node c03n02
Instance c03cdb_2 is running on node c03n01
[oracle@c03n01 ~]$
```

Part 5: Using Policy-Based Cluster Management to Manage Pluggable Database Services

In the final part of this practice, you will create services for each of your newly created pluggable databases and use policy-based management to govern the availability of those services.

The capabilities that you will explore in this part of the practice form the foundation for effectively consolidating multiple logical databases into a single multitenant Oracle RAC database.

71. Use SQL*Plus to connect to your newly created container database.

```
[oracle@c03n01 ~]$ sqlplus sys/oracle_4U@cluster03-
scan:1521/c03cdb.example.com as sysdba

SQL*Plus: Release 12.1.0.2.0 Production ...

SQL>
```

72. Display the current container for your connection. Notice that you are connected to the root container (`CDB$ROOT`).

```
SQL> show con_name

CON_NAME
------------------------------
CDB$ROOT
SQL>
```

73. Display all of the available pluggable databases. The output will contain the seed pluggable database (`PDB$SEED`), which is used as the template for new pluggable databases, along with the two pluggable databases that you specified during database creation (`PDB1` and `PDB2`).

```
SQL> show pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB1                           READ WRITE NO
         4 PDB2                           READ WRITE NO

SQL> exit
Disconnected from Oracle Database 12c Enterprise Edition ...
[oracle@c03n01 ~]$
```

74. Create a new service named `devsrv`. Associate `devsrv` with the `dev` server pool and the `pdb1` pluggable database. Specify `-policy automatic` to ensure that the service is automatically started when the database is restarted. Also, specify `-cardinality uniform` to ensure that the service is offered on all instances in the server pool.

```
[oracle@c03n01 ~]$ srvctl add service -database c03cdb
-service devsrv -serverpool dev -pdb pdb1 -policy automatic
-cardinality uniform
[oracle@c03n01 ~]$
```

75. Create a new service named `repsrv`. Associate `repsrv` with the `rep` server pool and the `pdb2` pluggable database. Specify the same additional attributes that you used for the `devsrv` service.

```
[oracle@c03n01 ~]$ srvctl add service -database c03cdb
-service repsrv -serverpool rep -pdb pdb2 -policy automatic
-cardinality uniform
[oracle@c03n01 ~]$
```

76. Start both of your newly created services. Then examine the status of the services.

```
[oracle@c03n01 ~]$ srvctl start service -database c03cdb
[oracle@c03n01 ~]$ srvctl status service -database c03cdb
Service devsrv is running on nodes: c03n02
Service repsrv is running on nodes: c03n01
[oracle@c03n01 ~]$
```

77. Examine the server pool status. Cross-reference the output from the previous step to confirm that service `devsrv` is running in the `dev` pool, and service `repsrv` is running in the `rep` pool.

```
[oracle@c03n01 ~]$ srvctl status srvpool -detail
Server pool name: Free
Active servers count: 0
Active server names:
Server pool name: Generic
Active servers count: 0
Active server names:
Server pool name: dev
Active servers count: 1
Active server names: c03n02
NAME=c03n02 STATE=ONLINE
Server pool name: rep
Active servers count: 1
Active server names: c03n01
NAME=c03n01 STATE=ONLINE
[oracle@c03n01 ~]$
```

78. Connect to your database using the `devsrv` service and confirm that the connection uses the `PDB1` pluggable database.

```
[oracle@c03n01 ~]$ sqlplus sys/oracle_4U@cluster03-
scan:1521/devsrv.example.com as sysdba

SQL*Plus: Release 12.1.0.2.0 Production ...

SQL> show con_name

CON_NAME
```

```
------------------------------
PDB1
SQL> show pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         3 PDB1                           READ WRITE NO
SQL> exit
Disconnected from Oracle Database 12c Enterprise Edition ...

[oracle@c03n01 ~]$
```

79. Connect to your database using the `repsrv` service and confirm that the connection uses the `PDB2` pluggable database.

```
[oracle@c03n01 ~]$ sqlplus sys/oracle_4U@cluster03-
scan:1521/repsrv.example.com as sysdba

SQL*Plus: Release 12.1.0.2.0 Production ...

SQL> show con_name

CON_NAME
------------------------------
PDB2
SQL> show pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         4 PDB2                           READ WRITE NO
SQL> exit
Disconnected from Oracle Database 12c Enterprise Edition ...
[oracle@c03n01 ~]$
```

80. Using your `grid` terminal session on `c03n01`, activate the `day` policy. Notice the effect on your service placements. Also notice how quickly the services are stopped and started compared with how long it would take to stop and start database instances.

```
[grid@c03n01 ~]$ crsctl modify policyset -attr
"LAST_ACTIVATED_POLICY='day'" -f
CRS-2673: Attempting to stop 'ora.c03cdb.repsrv.svc' on 'c03n01'
CRS-2677: Stop of 'ora.c03cdb.repsrv.svc' on 'c03n01' succeeded
CRS-2672: Attempting to start 'ora.c03cdb.devsrv.svc' on
'c03n01'
CRS-2676: Start of 'ora.c03cdb.devsrv.svc' on 'c03n01' succeeded
[grid@c03n01 ~]$
```

81. Return to your `oracle` terminal session and examine the status of your services. Confirm that the `devsrv` service is available on both nodes and that the `repsrv` service is not running.

```
[oracle@c03n01 ~]$ srvctl status service -database c03cdb
Service devsrv is running on nodes: c03n02,c03n01
Service repsrv is not running.
[oracle@c03n01 ~]$
```

82. Confirm the availability of the `devsrv` service by connecting to your database.

```
[oracle@c03n01 ~]$ sqlplus sys/oracle_4U@cluster03-
scan:1521/devsrv.example.com as sysdba

SQL*Plus: Release 12.1.0.2.0 Production ...

SQL> exit
Disconnected from Oracle Database 12c Enterprise Edition ...
[oracle@c03n01 ~]$
```

83. Confirm that `repsrv` is currently not available by attempting a connection using the service.

```
[oracle@c03n01 ~]$ sqlplus sys/oracle_4U@cluster03-
scan:1521/repsrv.example.com as sysdba

SQL*Plus: Release 12.1.0.2.0 Production ...

ERROR:
ORA-12514: TNS:listener does not currently know of service
requested in connect
descriptor


Enter user-name: ^C
[oracle@c03n01 ~]$
```

84. Return to the `grid` terminal session and activate the `night` policy.

```
[grid@c03n01 ~]$ crsctl modify policyset -attr
"LAST_ACTIVATED_POLICY='night'" -f
CRS-2673: Attempting to stop 'ora.c03cdb.devsrv.svc' on 'c03n02'
CRS-2677: Stop of 'ora.c03cdb.devsrv.svc' on 'c03n02' succeeded
CRS-2672: Attempting to start 'ora.c03cdb.repsrv.svc' on
'c03n02'
CRS-2676: Start of 'ora.c03cdb.repsrv.svc' on 'c03n02' succeeded
[grid@c03n01 ~]$
```

85. Return to the `oracle` terminal session and confirm that both services (`devsrv` and `repsrv`) are running again.

```
[oracle@c03n01 ~]$ srvctl status service -database c03cdb
Service devsrv is running on nodes: c03n01
Service repsrv is running on nodes: c03n02
[oracle@c03n01 ~]$
```

86. Examine the server pool status. Cross-reference the output from the previous step to again confirm that service `devsrv` is running in the `dev` pool, and service `repsrv` is running in the `rep` pool.

```
[oracle@c03n01 ~]$ srvctl status srvpool -detail
Server pool name: Free
Active servers count: 0
Active server names:
Server pool name: Generic
Active servers count: 0
Active server names:
Server pool name: dev
Active servers count: 1
Active server names: c03n01
NAME=c03n01 STATE=ONLINE
Server pool name: rep
Active servers count: 1
Active server names: c03n02
NAME=c03n02 STATE=ONLINE
[oracle@c03n01 ~]$
```

So far you have seen how the services automatically respond to policy changes. In the last part of this practice, you will also see how policy-managed pluggable database services respond to node outages.

87. Using your `grid` terminal session, shut down Oracle Clusterware on the `rep` pool server. Use the information from the previous steps to determine which node to stop. Note that your environment may differ from the example below.

```
[grid@c03n01 ~]$ su -c "crsctl stop cluster -n c03n02 -f"
Password: <oracle>
CRS-2673: Attempting to stop 'ora.crsd' on 'c03n02'
CRS-2790: Starting shutdown of Cluster Ready Services-managed
resources on 'c03n02'
CRS-2673: Attempting to stop 'ora.LISTENER_SCAN1.lsnr' on
'c03n02'
CRS-2673: Attempting to stop 'ora.c03cdb.repsrv.svc' on 'c03n02'
...
CRS-2677: Stop of 'ora.asm' on 'c03n02' succeeded
CRS-2673: Attempting to stop 'ora.cluster_interconnect.haip' on
'c03n02'
```

```
CRS-2677: Stop of 'ora.cluster_interconnect.haip' on 'c03n02'
succeeded
CRS-2673: Attempting to stop 'ora.cssd' on 'c03n02'
CRS-2677: Stop of 'ora.cssd' on 'c03n02' succeeded
[grid@c03n01 ~]$
```

88. Using your `oracle` terminal session, examine the server pool status. Notice that the remaining server (`c03n01` in the example below) has automatically moved from the `dev` pool to the `rep` pool.

    Note: In the previous step, if you stopped the cluster on `c03n01`, you will need to establish a new terminal session on `c03n02` to execute this step and the following steps performed as the `oracle` user. If this is the case, remember to use the `oraenv` script to configure the terminal environment.

```
[oracle@c03n01 ~]$ srvctl status srvpool -detail
Server pool name: Free
Active servers count: 0
Active server names:
Server pool name: Generic
Active servers count: 0
Active server names:
Server pool name: dev
Active servers count: 0
Active server names:
Server pool name: rep
Active servers count: 1
Active server names: c03n01
NAME=c03n01 STATE=ONLINE
[oracle@c03n01 ~]$
```

89. Examine the service status and confirm that `repsrv` is running in the `rep` pool.

```
[oracle@c03n01 ~]$ srvctl status service -database c03cdb
Service devsrv is not running.
Service repsrv is running on nodes: c03n01
[oracle@c03n01 ~]$
```

90. Using your `grid` terminal session, activate the `day` policy.

```
[grid@c03n01 ~]$ crsctl modify policyset -attr
"LAST_ACTIVATED_POLICY='day'" -f
CRS-2673: Attempting to stop 'ora.c03cdb.repsrv.svc' on 'c03n01'
CRS-2677: Stop of 'ora.c03cdb.repsrv.svc' on 'c03n01' succeeded
CRS-2672: Attempting to start 'ora.c03cdb.devsrv.svc' on
'c03n01'
CRS-2676: Start of 'ora.c03cdb.devsrv.svc' on 'c03n01' succeeded
[grid@c03n01 ~]$
```

91. Back in your `oracle` terminal session, confirm that the only available node has moved from the `rep` pool to the `dev` pool.

```
[oracle@c03n01 ~]$ srvctl status srvpool -detail
Server pool name: Free
Active servers count: 0
Active server names:
Server pool name: Generic
Active servers count: 0
Active server names:
Server pool name: dev
Active servers count: 1
Active server names: c03n01
NAME=c03n01 STATE=ONLINE
Server pool name: rep
Active servers count: 0
Active server names:
[oracle@c03n01 ~]$
```

92. Examine the service status and confirm that the service availability matches expectations for the current server pool configuration.

```
[oracle@c03n01 ~]$ srvctl status service -database c03cdb
Service devsrv is running on nodes: c03n01
Service repsrv is not running.
[oracle@c03n01 ~]$
```

93. Return to your `grid` session and restart the cluster on the node that you previously shut down.

```
[grid@c03n01 ~]$ su -c "crsctl start cluster -n c03n02"
Password: <oracle>
CRS-2672: Attempting to start 'ora.cssdmonitor' on 'c03n02'
CRS-2672: Attempting to start 'ora.evmd' on 'c03n02'
CRS-2676: Start of 'ora.cssdmonitor' on 'c03n02' succeeded
...
CRS-2676: Start of 'ora.storage' on 'c03n02' succeeded
CRS-2672: Attempting to start 'ora.crsd' on 'c03n02'
CRS-2676: Start of 'ora.crsd' on 'c03n02' succeeded
[grid@c03n01 ~]$
```

94. Using your `oracle` terminal session on `c03n01`, re-examine the server pool status and confirm that both servers are associated with the `dev` server pool, which is in line with the current (`day`) policy.

```
[oracle@c03n01 ~]$ srvctl status srvpool -detail
Server pool name: Free
Active servers count: 0
Active server names:
Server pool name: Generic
Active servers count: 0
Active server names:
Server pool name: dev
Active servers count: 2
Active server names: c03n01,c03n02
NAME=c03n01 STATE=ONLINE
NAME=c03n02 STATE=ONLINE
Server pool name: rep
Active servers count: 0
Active server names:
[oracle@c03n01 ~]$
```

95. Examine the service status and confirm that `devsrv` is being offered on both cluster nodes.

    Note: You may have to wait for a few minutes for the service to appear on both nodes because of the time required to restart the database instance and service on the recently restarted node.

```
[oracle@c03n01 ~]$ srvctl status service -database c03cdb
Service devsrv is running on nodes: c03n02,c03n01
Service repsrv is not running.
[oracle@c03n01 ~]$
```

    Congratulations! You have completed the practice.

96. In preparation for future practices, stop the `c03cdb` database and restart the `c03orcl` database. Then, confirm the database status as shown below.

```
[oracle@c03n01 ~]$ srvctl stop database -d c03cdb
[oracle@c03n01 ~]$ srvctl start database -d c03orcl
[oracle@c03n01 ~]$ srvctl status database -thishome
Database unique name: c03cdb
Instance c03cdb_1 is not running on node c03n02
Instance c03cdb_2 is not running on node c03n01

Database unique name: c03ora2
Database is not running.

Database unique name: c03orcl
Instance c03orcl_1 is running on node c03n02
Instance c03orcl_2 is running on node c03n01
[oracle@c03n01 ~]$
```

97. Exit all the terminal sessions.

Practices for Lesson 5: Flex Clusters

Chapter 5

Practices for Lesson 5: Overview

Practices Overview

In these practices, you will:

- Convert an existing 2-node standard cluster to a Flex Cluster.
- Add two leaf nodes to your Flex Cluster.
- Create a series of highly available application resources running on one of the Flex Cluster Leaf Nodes.

Practice 5-1: Converting to Flex Clusters

Overview

In this practice, you will convert an existing 2-node standard cluster to a Flex Cluster.

Tasks

1. Establish a terminal session connected to `c03n01` as the `grid` user and configure the terminal environment as shown below.

```
$ ssh grid@c03n01
[grid@c03n01 ~]$ . oraenv
ORACLE_SID = [grid] ? +ASM1
The Oracle base has been set to /u01/app/grid
[grid@c03n01 ~]$
```

2. Examine the current cluster mode status and confirm that the cluster is running in standard mode.

```
[grid@c03n01 ~]$ crsctl get cluster mode status
Cluster is running in "standard" mode
[grid@c03n01 ~]$
```

3. Examine the ASM cluster mode status and confirm that Flex ASM is enabled.

   Note: A Flex Cluster requires Flex ASM. If Flex ASM is not enabled, it will need to be enabled before the cluster can be converted to a Flex Cluster.

```
[grid@c03n01 ~]$ asmcmd showclustermode
ASM cluster : Flex mode enabled
[grid@c03n01 ~]$
```

4. Examine the Grid Naming Service (GNS) configuration in the cluster.

   Note: A Flex Cluster requires GNS. You will configure GNS on your cluster later in this practice.

```
[grid@c03n01 ~]$ srvctl config gns
PRKF-1110 : Neither GNS server nor GNS client is configured on
this cluster
[grid@c03n01 ~]$
```

5. For the remainder of this practice, you will require system administrator privileges. Use the `su` command to assume system administrator privileges.

```
[grid@c03n01 ~]$ su
Password: <oracle>
[root@c03n01 grid]#
```

6. Confirm the existence of the GNS VIP address.

   Note: Your practice environment is preconfigured with a GNS VIP address. Typically, you would need to coordinate with your network administrator to define a GNS VIP address prior to configuring GNS.

```
[root@c03n01 grid]# nslookup cluster03-gns.example.com
Server:         192.0.2.1
Address:        192.0.2.1#53

Name:   cluster03-gns.example.com
Address: 192.0.2.123

[root@c03n01 grid]#
```

7. Add GNS to your cluster. Notice that you must specify the GNS VIP address to configure GNS.

```
[root@c03n01 grid]# srvctl add gns
-vip cluster03-gns.example.com
[root@c03n01 grid]#
```

8. Start GNS and then examine the GNS status to confirm that it is started.

```
[root@c03n01 grid]# srvctl start gns
[root@c03n01 grid]# srvctl status gns
GNS is running on node c03n02.
GNS is enabled on node c03n02.
[root@c03n01 grid]#
```

   Now that all of the prerequisites are in place, you are ready to convert the cluster to a Flex Cluster.

9. Set the cluster mode to `flex`. Notice that you must restart the cluster, including Oracle High Availability Services (OHAS), in order to complete the cluster mode conversion.

```
[root@c03n01 grid]# crsctl set cluster mode flex
CRS-4933: Cluster mode set to "flex"; restart Oracle High
Availability Services on all nodes for cluster to run in "flex"
mode.
[root@c03n01 grid]#
```

10. Stop Oracle Clusterware, including OHAS, on all of the cluster nodes.

```
[root@c03n01 grid]# crsctl stop crs
CRS-2791: Starting shutdown of Oracle High Availability
Services-managed resources on 'c03n01'
CRS-2673: Attempting to stop 'ora.crsd' on 'c03n01'
CRS-2790: Starting shutdown of Cluster Ready Services-managed
resources on 'c03n01'
...
CRS-2677: Stop of 'ora.driver.afd' on 'c03n01' succeeded
CRS-2677: Stop of 'ora.gipcd' on 'c03n01' succeeded
```

```
CRS-2793: Shutdown of Oracle High Availability Services-managed
resources on 'c03n01' has completed
CRS-4133: Oracle High Availability Services has been stopped.
[root@c03n01 grid]# ssh c03n02 "/u01/app/12.1.0/grid/bin/crsctl
stop crs"
CRS-2791: Starting shutdown of Oracle High Availability
Services-managed resources on 'c03n02'
CRS-2673: Attempting to stop 'ora.crsd' on 'c03n02'
CRS-2790: Starting shutdown of Cluster Ready Services-managed
resources on 'c03n02'
...
CRS-2677: Stop of 'ora.driver.afd' on 'c03n02' succeeded
CRS-2677: Stop of 'ora.gipcd' on 'c03n02' succeeded
CRS-2793: Shutdown of Oracle High Availability Services-managed
resources on 'c03n02' has completed
CRS-4133: Oracle High Availability Services has been stopped.
[root@c03n01 grid]#
```

11. Restart the cluster.

```
[root@c03n01 grid]# crsctl start crs -wait
CRS-4123: Starting Oracle High Availability Services-managed
resources
CRS-2672: Attempting to start 'ora.mdnsd' on 'c03n01'
CRS-2672: Attempting to start 'ora.evmd' on 'c03n01'
...
CRS-2676: Start of 'ora.c03orcl.db' on 'c03n01' succeeded
CRS-6016: Resource auto-start has completed for server c03n01
CRS-6024: Completed start of Oracle Cluster Ready Services-
managed resources
CRS-4123: Oracle High Availability Services has been started.
[root@c03n01 grid]# ssh c03n02 "/u01/app/12.1.0/grid/bin/crsctl
start crs -wait"
CRS-4123: Starting Oracle High Availability Services-managed
resources
CRS-2672: Attempting to start 'ora.mdnsd' on 'c03n02'
CRS-2672: Attempting to start 'ora.evmd' on 'c03n02'
...
CRS-2676: Start of 'ora.c03orcl.db' on 'c03n02' succeeded
CRS-6016: Resource auto-start has completed for server c03n02
CRS-6024: Completed start of Oracle Cluster Ready Services-
managed resources
CRS-4123: Oracle High Availability Services has been started.
[root@c03n01 grid]#
```

12. Re-examine the cluster mode setting and confirm that the cluster is now a Flex Cluster.

```
[root@c03n01 grid]# crsctl get cluster mode status
Cluster is running in "flex" mode
[root@c03n01 grid]#
```

13. Exit your terminal session.

Practice 5-2: Adding Leaf Nodes to a Flex Cluster

Overview

In this practice, you will add two leaf nodes to your Flex Cluster.

Tasks

1. Establish a terminal session connected to `c03n01` as the `grid` user and configure the terminal environment as shown below.

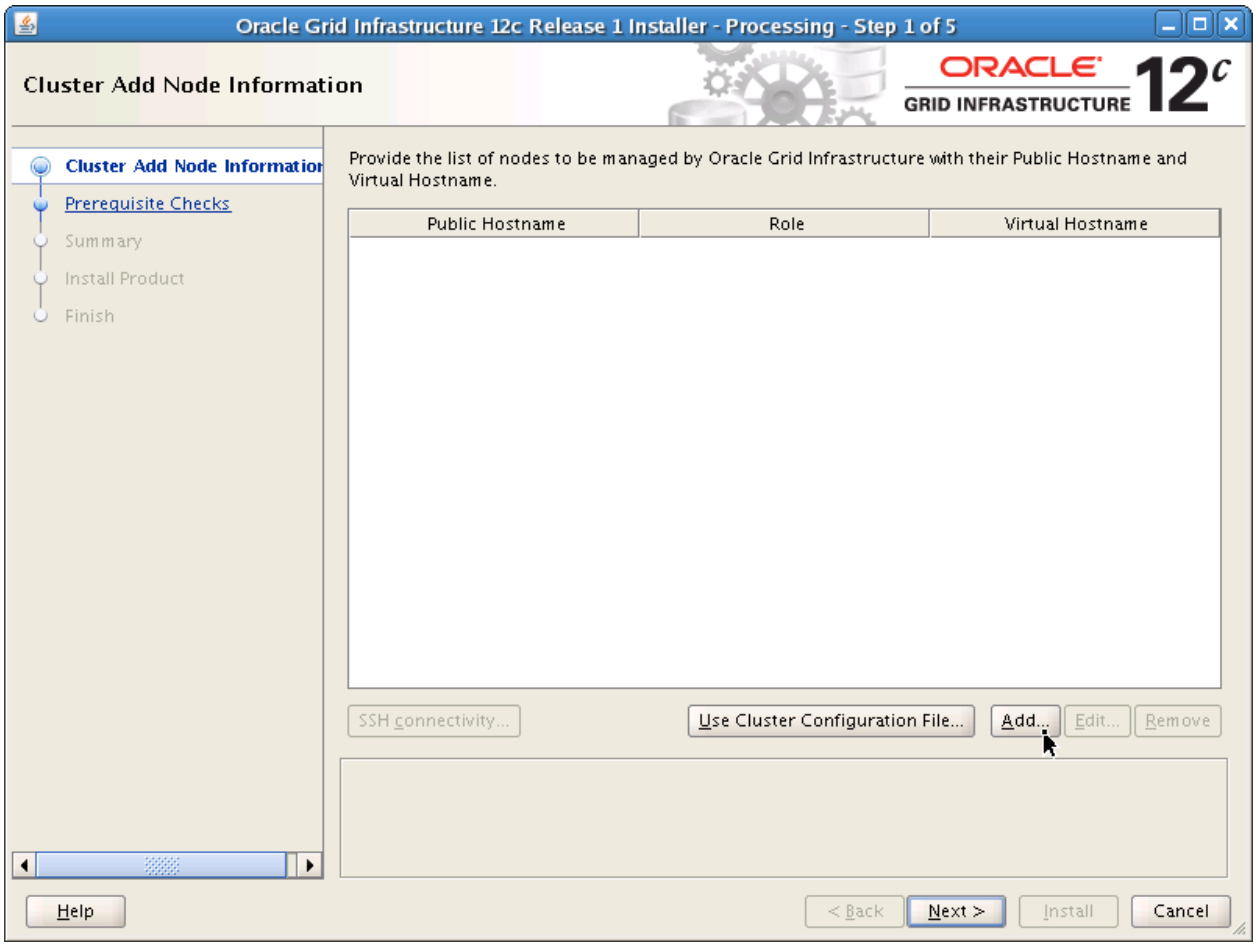
   Note: Make sure that you use `ssh` with the `-X` option because this option is required to ensure you terminal environment is correctly configured to run GUI applications and utilities.

```
$ ssh -X grid@c03n01
[grid@c03n01 ~]$ . oraenv
ORACLE_SID = [grid] ? +ASM1
The Oracle base has been set to /u01/app/grid
[grid@c03n01 ~]$
```

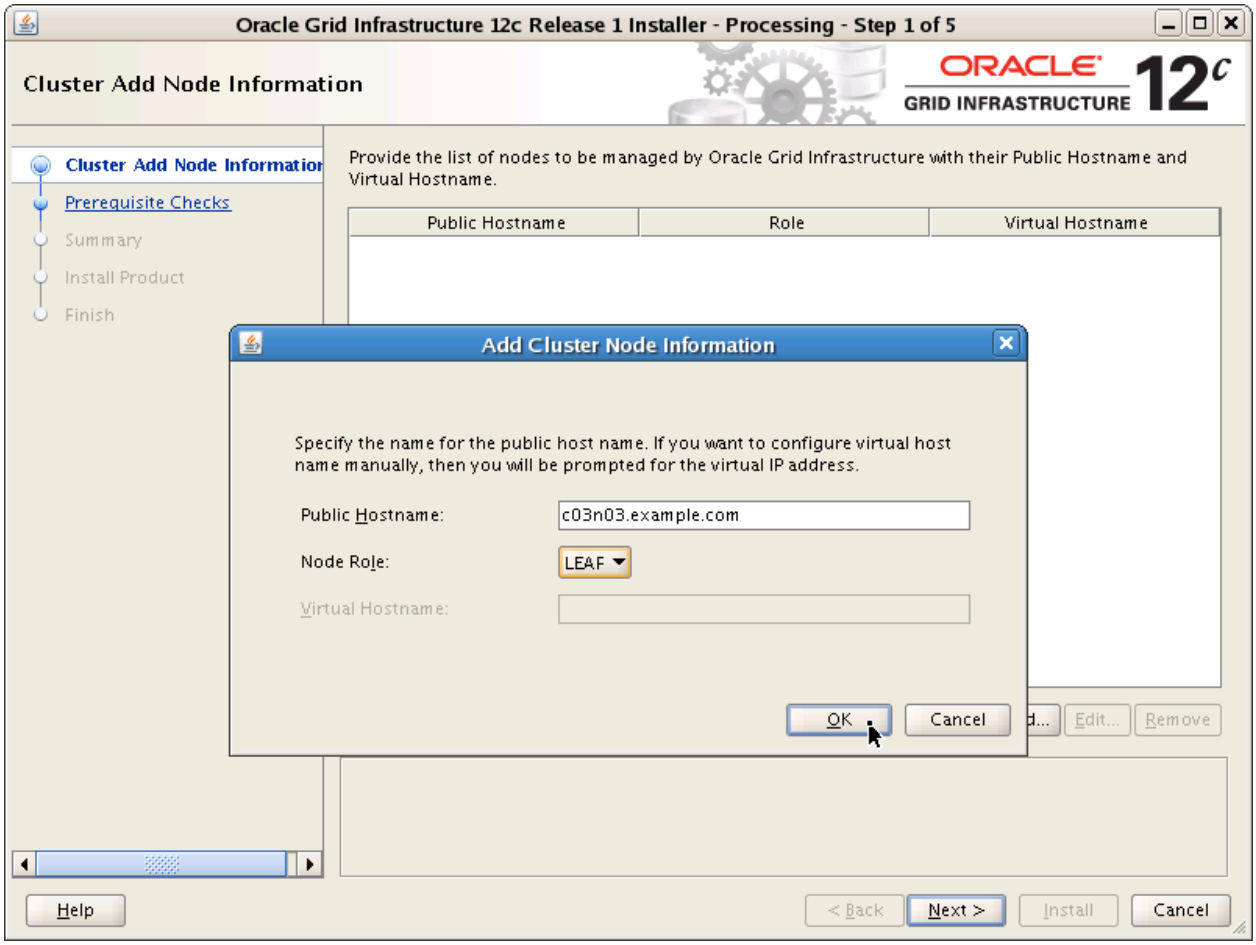
2. Start the script at `$ORACLE_HOME/addnode/addnode.sh`. Ensure that you run the script in the background so that you can use your terminal session while the script is running.

```
[grid@c03n01 ~]$ $ORACLE_HOME/addnode/addnode.sh &
[1] 4461
Starting Oracle Universal Installer...
```

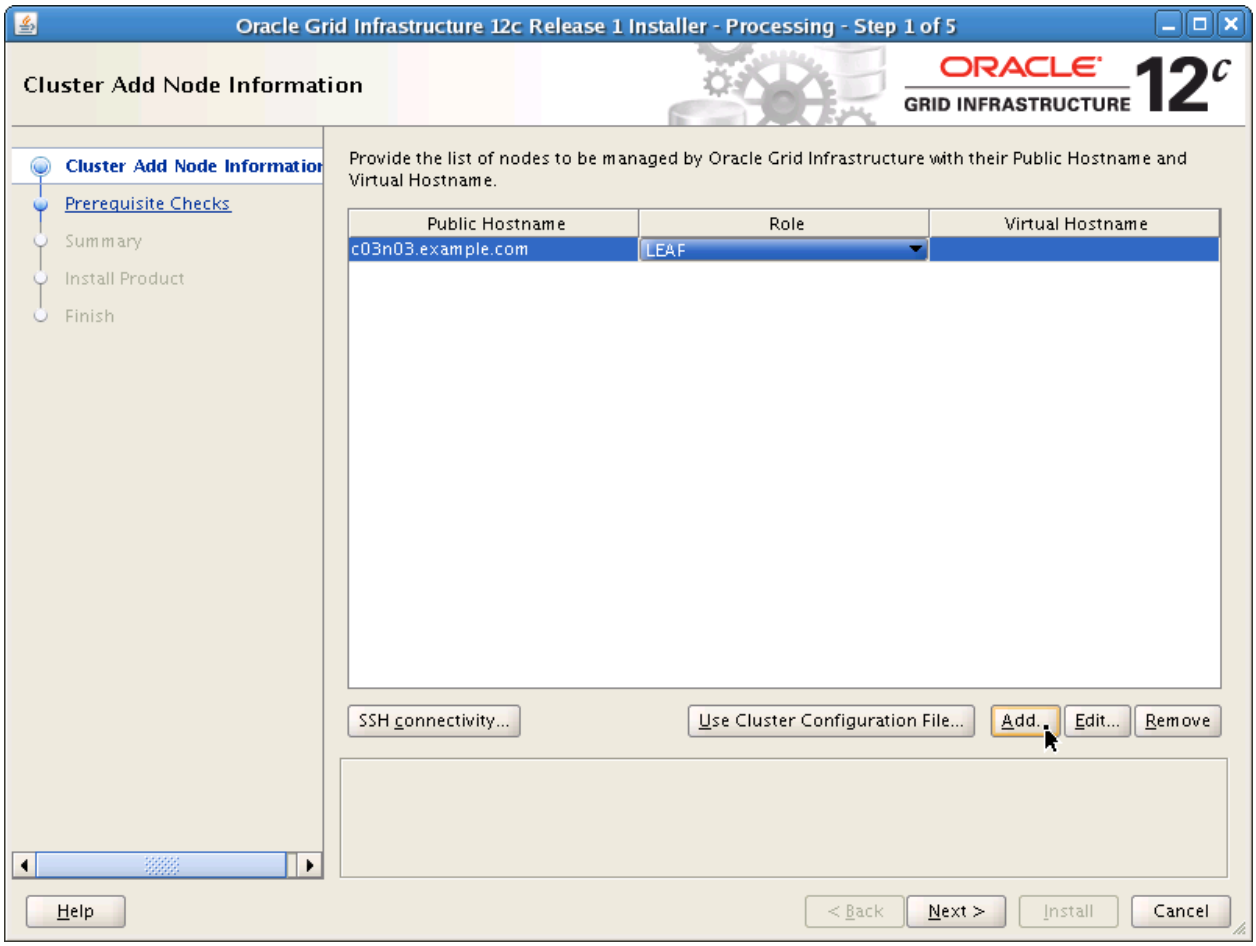
3. On the Cluster Add Node Information page, click Add.



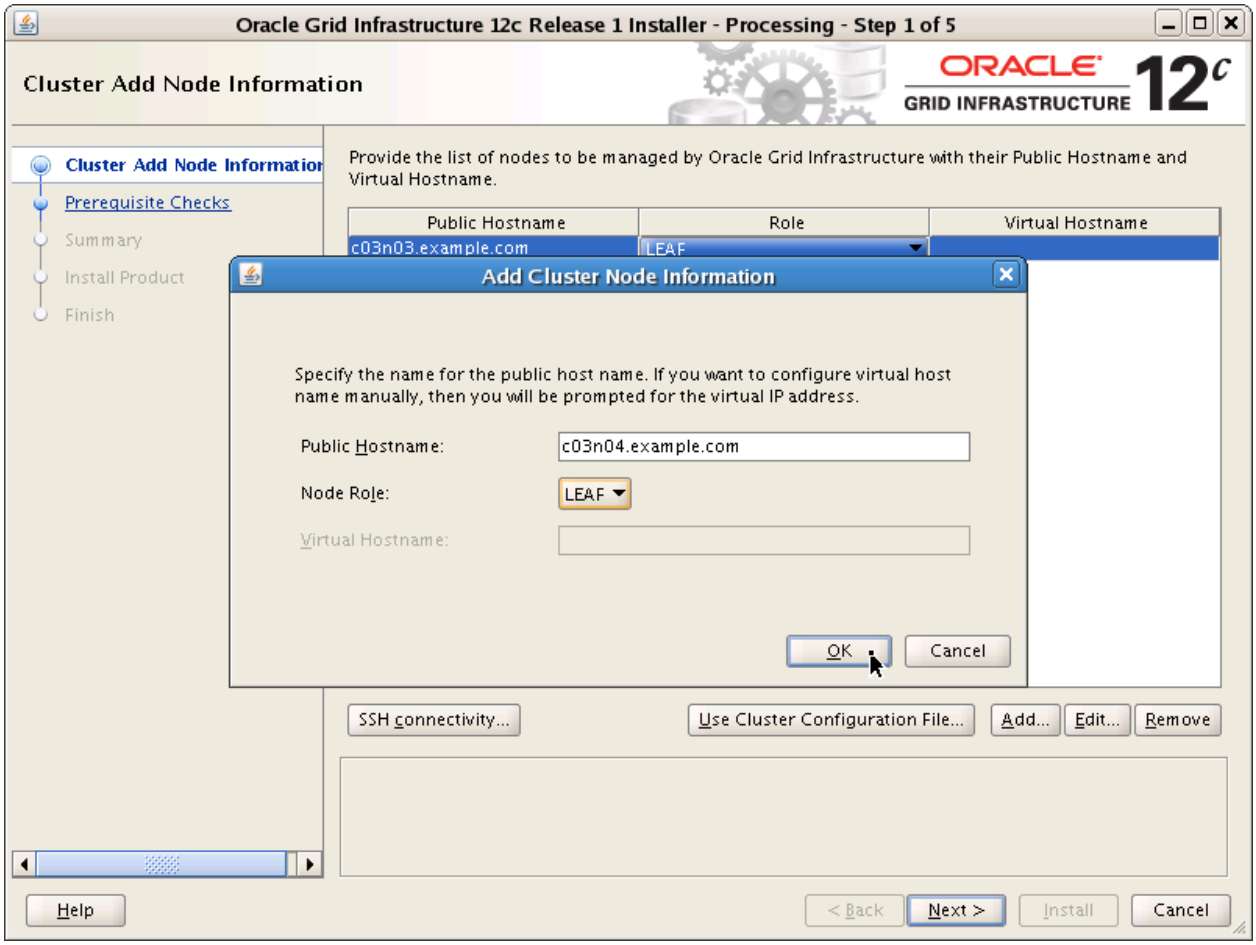
4. Provide the details for the first node to be added. Specify `c03n03.example.com` as the hostname and specify `LEAF` as the node role. Then, click OK to continue.



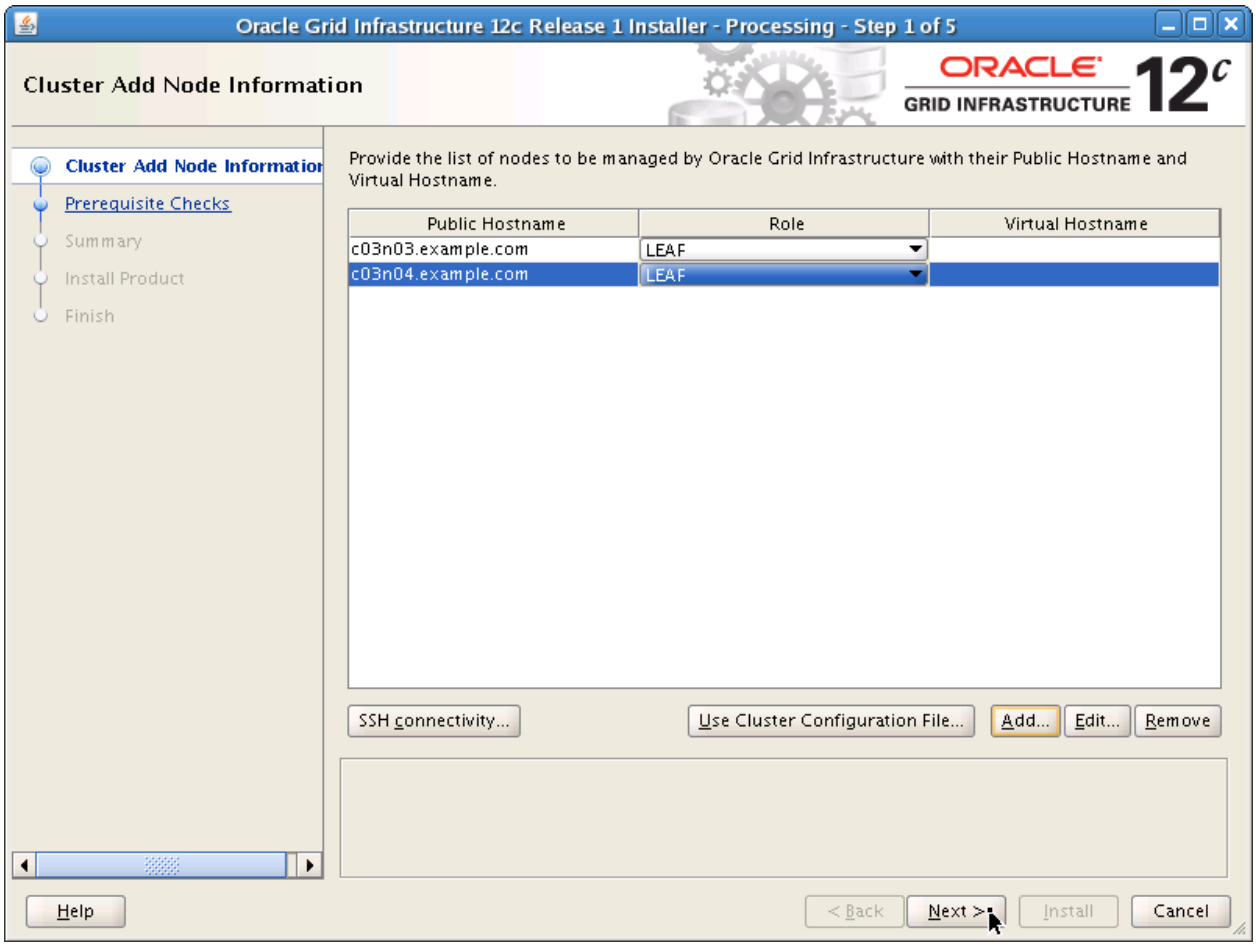
5. Click Add again on the Cluster Add Node Information page.



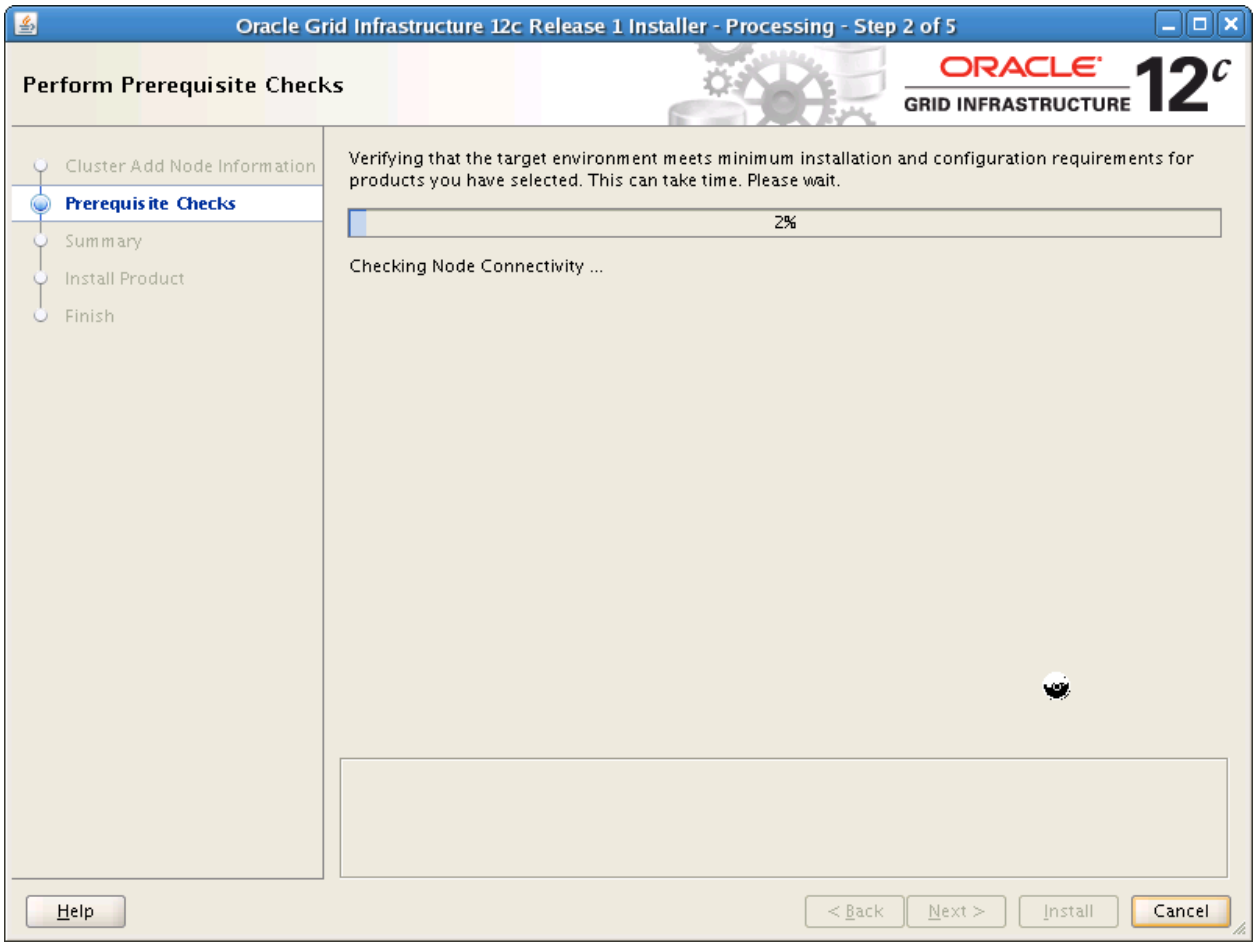
6. Provide the details for the second node to be added. Specify `c03n04.example.com` as the hostname and specify `LEAF` as the node role. Then, click OK to continue.



7. Click Next to continue past the Cluster Add Node Information page.

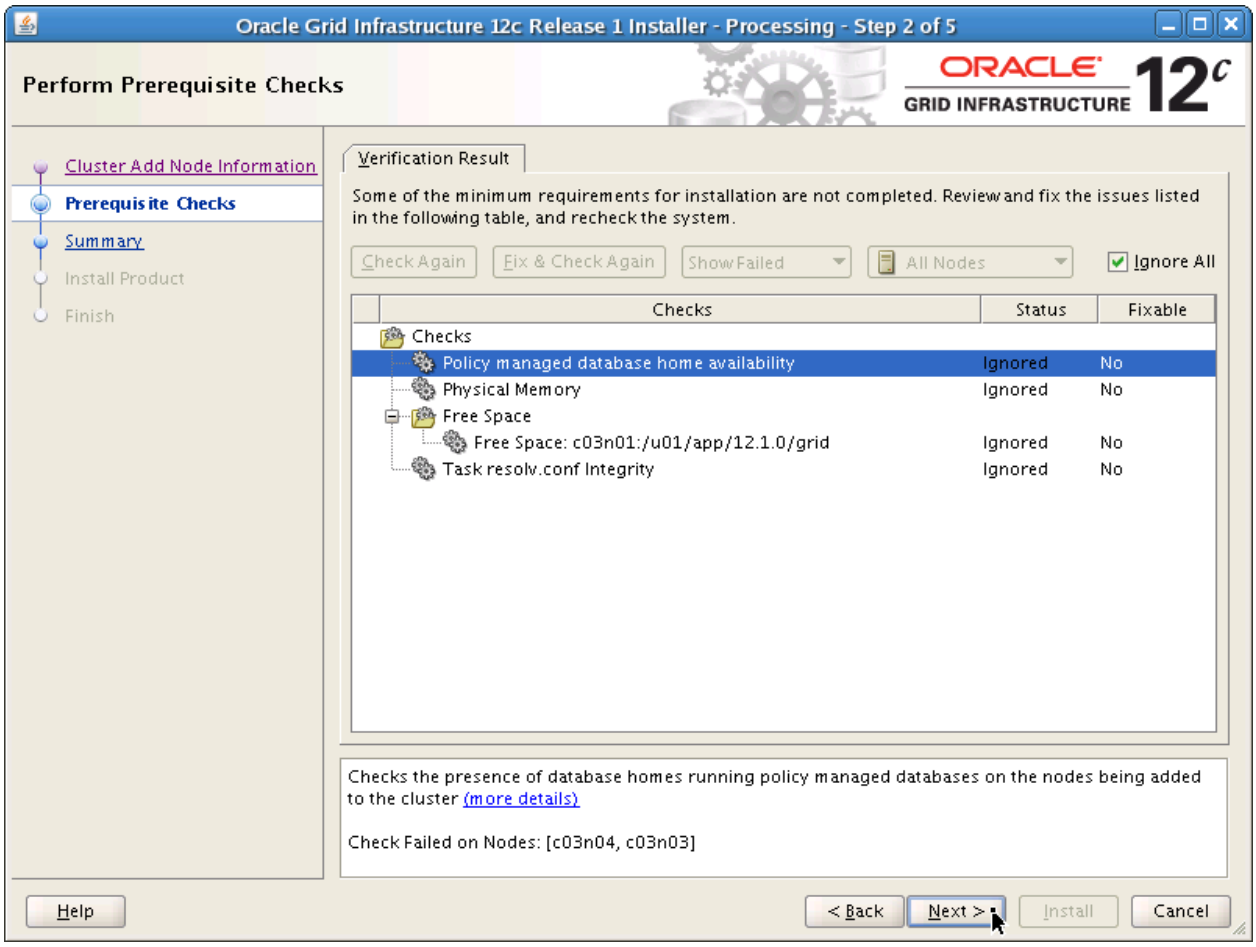


8. Wait while a series of prerequisite checks is preformed.

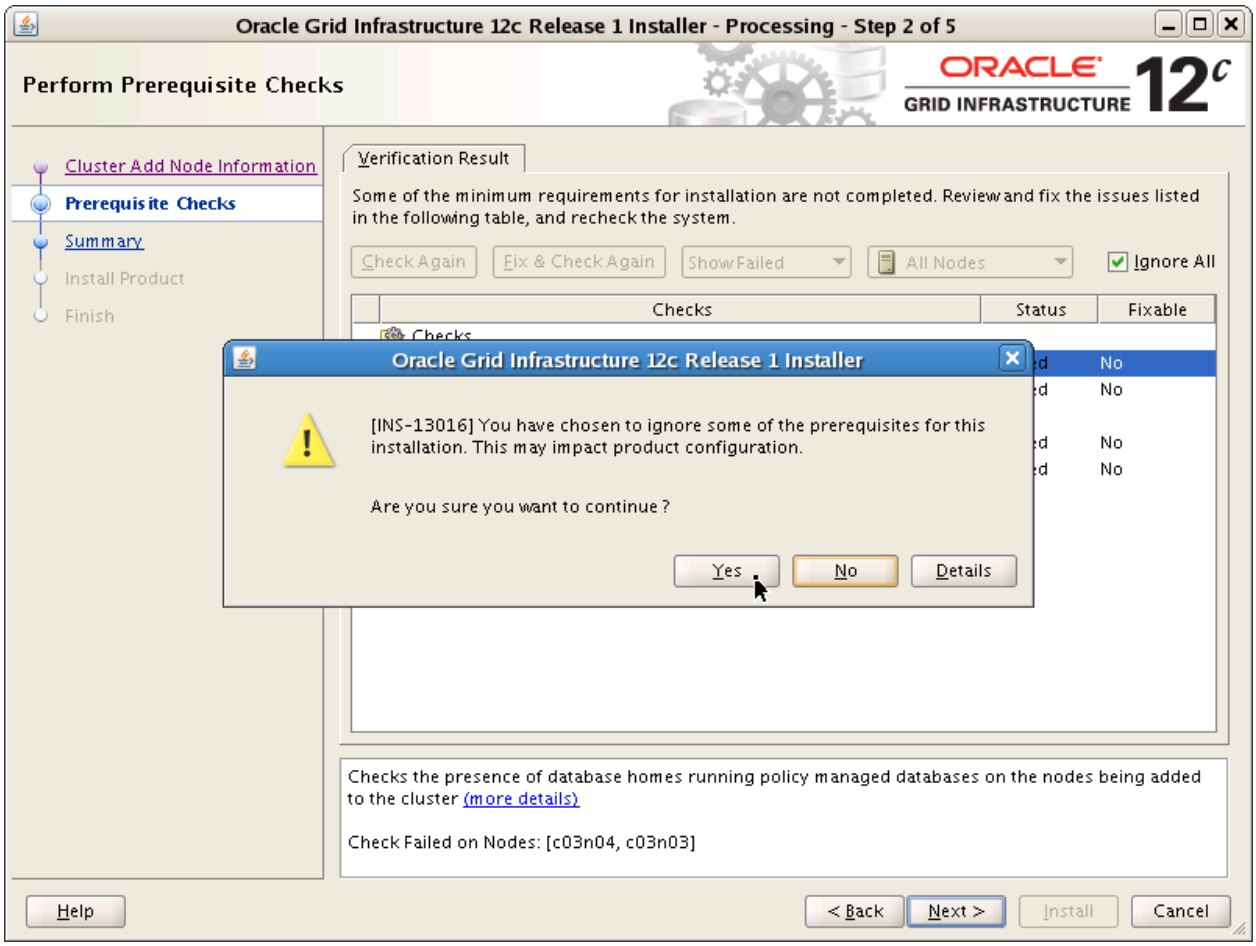


9. When the checks are complete, you will see a series of warnings. Select the option to ignore all of the warnings and click Next.

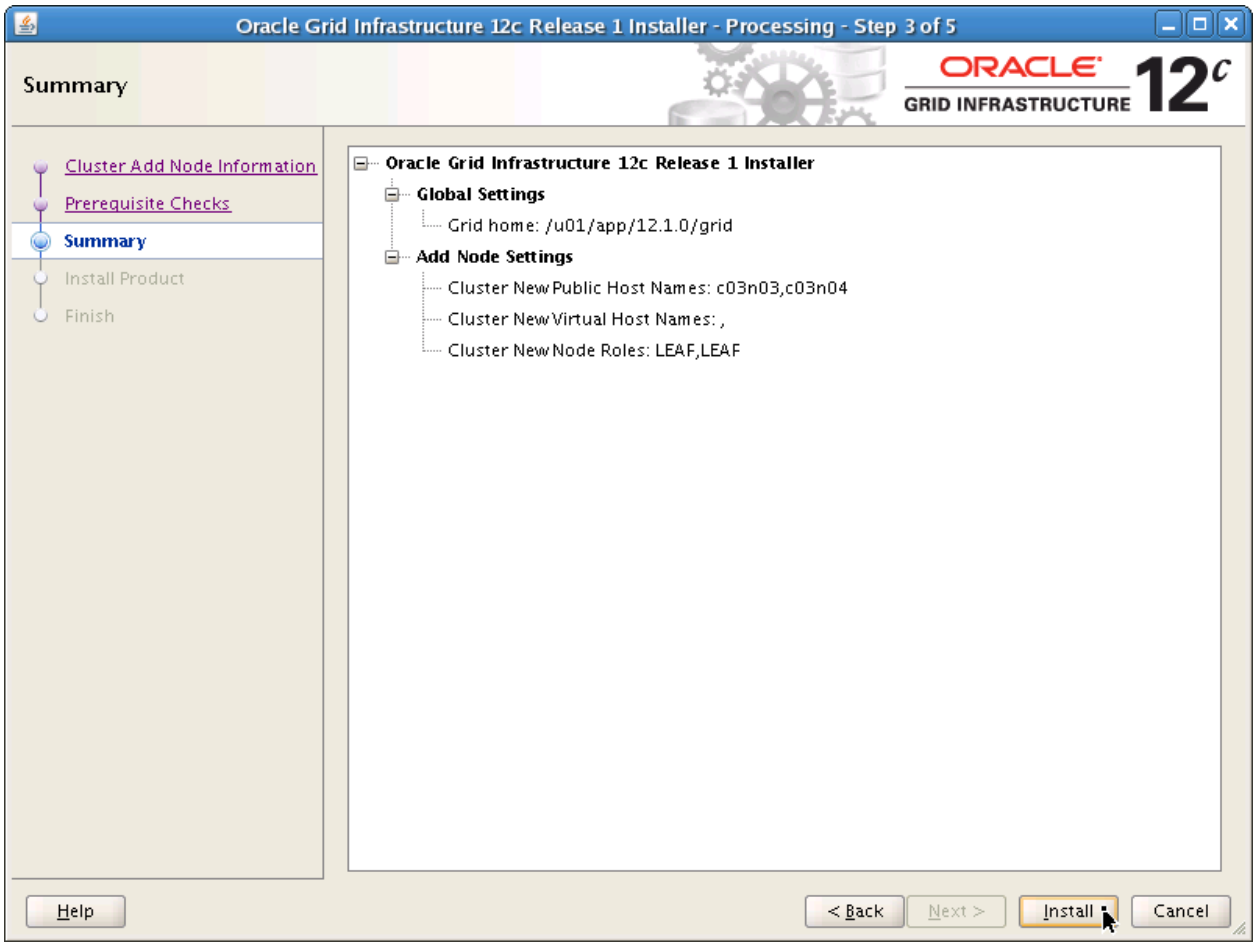
   Note: Usually, you would need to resolve any issues at this stage before proceeding. However, in this case the underlying issues are related to limitations within your practice environment, and the warnings can be safely ignored.



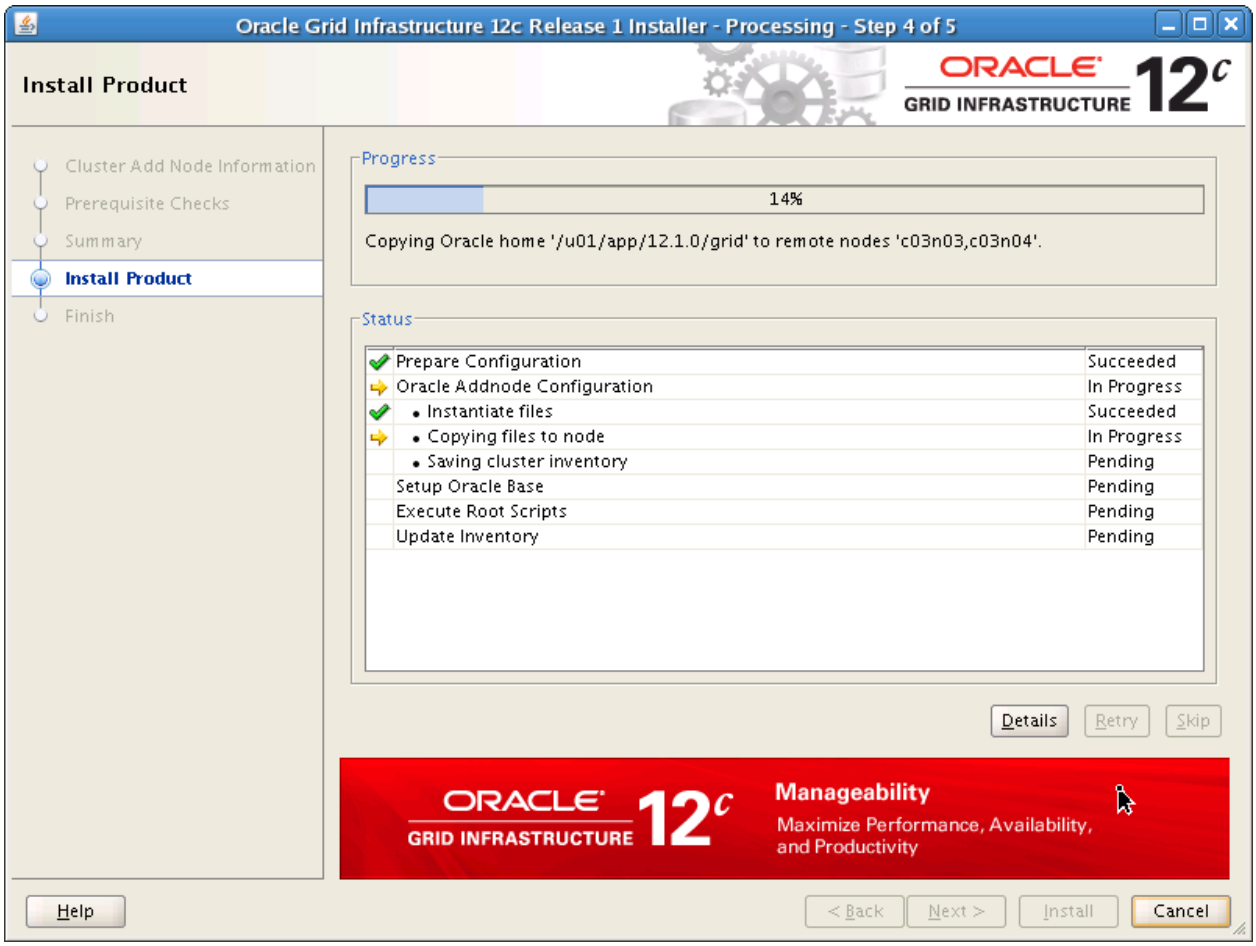
10. Click Yes to confirm that you wish to proceed.



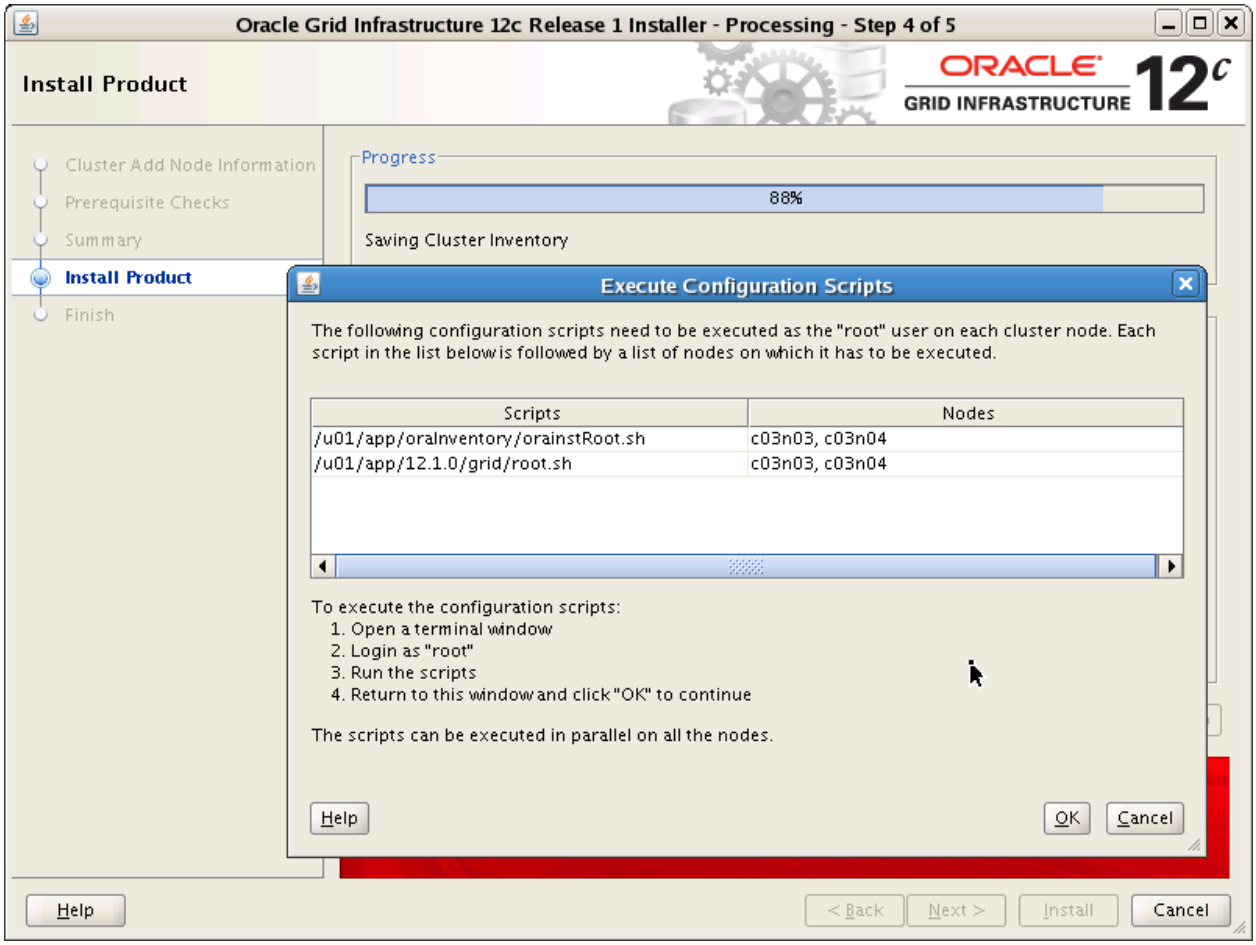
11. Click Install to commence the installation process on the new Leaf Nodes.



12. Monitor the installation progress.



13. Eventually, you will be instructed to execute a series of scripts on each of the new cluster nodes.



14. As the `root` user, execute the `orainstRoot.sh` script on each new Leaf Node.

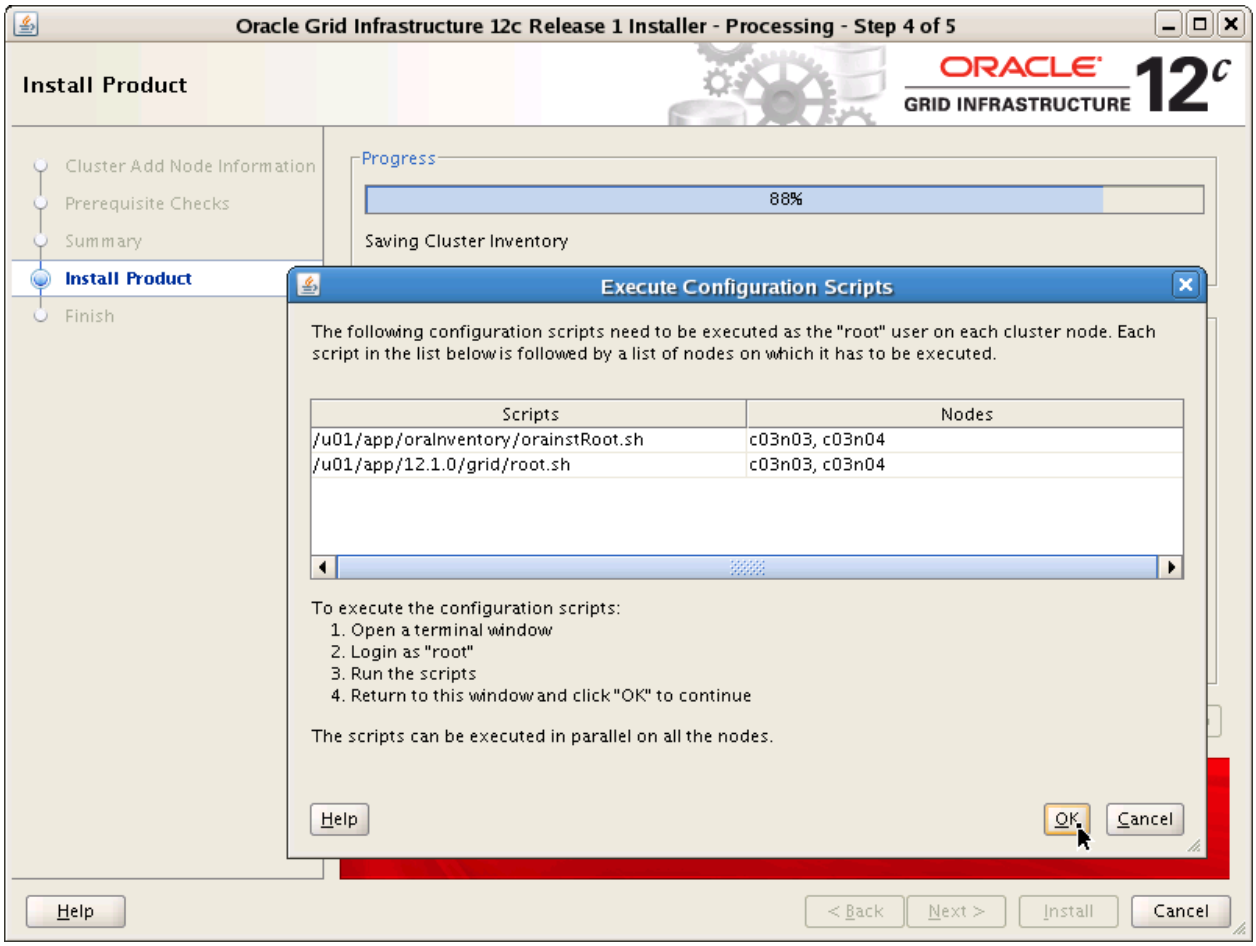
```
[grid@c03n01 ~]$ ssh root@c03n03
"/u01/app/oraInventory/orainstRoot.sh"
root@c03n03's password: <oracle>
Changing permissions of /u01/app/oraInventory.
Adding read,write permissions for group.
Removing read,write,execute permissions for world.

Changing groupname of /u01/app/oraInventory to oinstall.
The execution of the script is complete.
[grid@c03n01 ~]$ ssh root@c03n04
"/u01/app/oraInventory/orainstRoot.sh"
root@c03n04's password: <oracle>
Changing permissions of /u01/app/oraInventory.
Adding read,write permissions for group.
Removing read,write,execute permissions for world.

Changing groupname of /u01/app/oraInventory to oinstall.
The execution of the script is complete.
[grid@c03n01 ~]$
```

15. As the `root` user, execute the `root.sh` script on each new Leaf Node. Press Return when you are prompted for the path of the local bin directory.

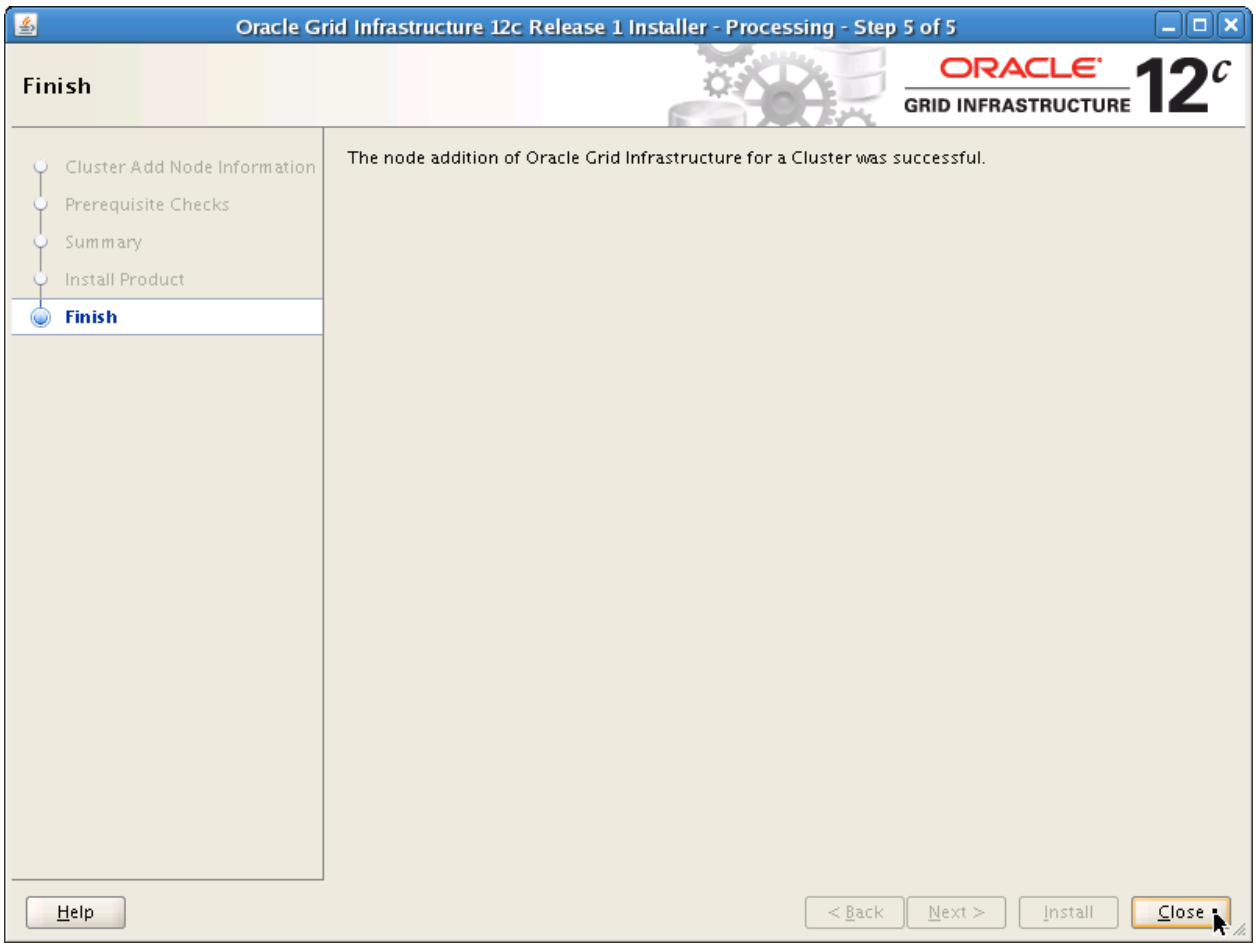
```
[grid@c03n01 ~]$ ssh root@c03n03 "/u01/app/12.1.0/grid/root.sh"
root@c03n03's password: <oracle>
Performing root user operation.

The following environment variables are set as:
    ORACLE_OWNER= grid
    ORACLE_HOME=  /u01/app/12.1.0/grid

Enter the full pathname of the local bin directory:
[/usr/local/bin]:
   Copying dbhome to /usr/local/bin ...
   Copying oraenv to /usr/local/bin ...
   Copying coraenv to /usr/local/bin ...


Creating /etc/oratab file...
Entries will be added to the /etc/oratab file as needed by
Database Configuration Assistant when a database is created
Finished running generic part of root script.
Now product-specific root actions will be performed.
...
CRS-2672: Attempting to start 'ora.crsd' on 'c03n03'
CRS-2676: Start of 'ora.crsd' on 'c03n03' succeeded
CRS-6017: Processing resource auto-start for servers: c03n03
CRS-6016: Resource auto-start has completed for server c03n03
CRS-6024: Completed start of Oracle Cluster Ready Services-
managed resources
CRS-4123: Oracle High Availability Services has been started.
2014/07/17 01:52:39 CLSRSC-343: Successfully started Oracle
Clusterware stack

Successfully accumulated necessary OCR keys.
Creating OCR keys for user 'root', privgrp 'root'..
Operation successful.
2014/07/17 01:52:46 CLSRSC-325: Configure Oracle Grid
Infrastructure for a Cluster ... succeeded

[grid@c03n01 ~]$ ssh root@c03n04 "/u01/app/12.1.0/grid/root.sh"
root@c03n04's password: <oracle>
Performing root user operation.
```

```
The following environment variables are set as:
    ORACLE_OWNER= grid
    ORACLE_HOME=  /u01/app/12.1.0/grid

Enter the full pathname of the local bin directory:
[/usr/local/bin]:
   Copying dbhome to /usr/local/bin ...
   Copying oraenv to /usr/local/bin ...
   Copying coraenv to /usr/local/bin ...


Creating /etc/oratab file...
Entries will be added to the /etc/oratab file as needed by
Database Configuration Assistant when a database is created
Finished running generic part of root script.
Now product-specific root actions will be performed.
...
CRS-2672: Attempting to start 'ora.crsd' on 'c03n04'
CRS-2676: Start of 'ora.crsd' on 'c03n04' succeeded
CRS-6017: Processing resource auto-start for servers: c03n04
CRS-6016: Resource auto-start has completed for server c03n04
CRS-6024: Completed start of Oracle Cluster Ready Services-
managed resources
CRS-4123: Oracle High Availability Services has been started.
2014/07/17 02:13:39 CLSRSC-343: Successfully started Oracle
Clusterware stack

Successfully accumulated necessary OCR keys.
Creating OCR keys for user 'root', privgrp 'root'..
Operation successful.
2014/07/17 02:13:46 CLSRSC-325: Configure Oracle Grid
Infrastructure for a Cluster ... succeeded

[grid@c03n01 ~]$
```

16. After all of the root scripts are completed, click OK in the Execute Configuration Scripts dialog.



17. Finally, you will see a message indicating that the installation in complete. Click Close to exit the installer.



18. Back in your terminal session, check the status of the cluster. Notice that the new Leaf Nodes (`c03n03` and `c03n04`) are now part of the cluster.

```
[grid@c03n01 ~]$ crsctl check cluster -all
**************************************************************
c03n01:
CRS-4537: Cluster Ready Services is online
CRS-4529: Cluster Synchronization Services is online
CRS-4533: Event Manager is online
**************************************************************
c03n02:
CRS-4537: Cluster Ready Services is online
CRS-4529: Cluster Synchronization Services is online
CRS-4533: Event Manager is online
**************************************************************
c03n03:
CRS-4537: Cluster Ready Services is online
CRS-4529: Cluster Synchronization Services is online
CRS-4533: Event Manager is online
```

```
**************************************************************
c03n04:
CRS-4537: Cluster Ready Services is online
CRS-4529: Cluster Synchronization Services is online
CRS-4533: Event Manager is online
**************************************************************
[grid@c03n01 ~]$
```

19. Check the cluster node role status and confirm that the node roles are as expected.

```
[grid@c03n01 ~]$ crsctl get node role status -all
Node 'c03n01' active role is 'hub'
Node 'c03n02' active role is 'hub'
Node 'c03n03' active role is 'leaf'
Node 'c03n04' active role is 'leaf'
[grid@c03n01 ~]$
```

20. Exit your terminal session.

Practice 5-3: Configuring Highly Available Application Resources on Flex Cluster Leaf Nodes

Overview

In this practice, you will create a series of highly available application resources running on one of the Flex Cluster Leaf Nodes.

Tasks

1. Establish a terminal session connected to `c03n01` as the `grid` user and configure the terminal environment as shown below.

```
$ ssh grid@c03n01
[grid@c03n01 ~]$ . oraenv
ORACLE_SID = [grid] ? +ASM1
The Oracle base has been set to /u01/app/grid
[grid@c03n01 ~]$
```

2. Examine the cluster to identify the role for each node in the cluster.

```
[grid@c03n01 ~]$ crsctl get node role status -all
Node 'c03n01' active role is 'hub'
Node 'c03n02' active role is 'hub'
Node 'c03n03' active role is 'leaf'
Node 'c03n04' active role is 'leaf'
[grid@c03n01 ~]$
```

3. Examine the cluster to identify the currently defined server pools and server allocations. Notice that currently the Hub Nodes are allocated to the `dev` server pool, which contains the RAC database that you have used in previous practices. In addition, both Leaf Nodes are currently in the `Free` pool.

```
[grid@c03n01 ~]$ crsctl status serverpool
NAME=Free
ACTIVE_SERVERS=c03n03 c03n04

NAME=Generic
ACTIVE_SERVERS=

NAME=ora.dev
ACTIVE_SERVERS=c03n01 c03n02

NAME=ora.rep
ACTIVE_SERVERS=

[grid@c03n01 ~]$
```

4. Examine the cluster to identify the currently defined server categories. As the name implies, server categories provide a way of categorizing servers in the cluster. There are two built-in server categories. All of the Hub Nodes in the cluster are implicitly associated with `ora.hub.category`, while all of the Leaf Nodes in the cluster are implicitly associated with `ora.leaf.category`.

```
[grid@c03n01 ~]$ crsctl status category
NAME=ora.hub.category
ACL=owner:root:rwx,pgrp:root:r-x,other::r--
ACTIVE_CSS_ROLE=hub
EXPRESSION=

NAME=ora.leaf.category
ACL=owner:root:rwx,pgrp:root:r-x,other::r--
ACTIVE_CSS_ROLE=leaf
EXPRESSION=

[grid@c03n01 ~]$
```

5. Create a new server pool to house a series of highly available application resources on one of the Flex Cluster Leaf Nodes.

```
[grid@c03n01 ~]$ srvctl add srvpool -serverpool my_app_pool
-min 1 -max 1 -category "ora.leaf.category"
[grid@c03n01 ~]$
```

6. Reexamine the server pools. Notice that one of the leaf nodes has been allocated to the newly created server pool (`my_app_pool`).

```
[grid@c03n01 ~]$ crsctl status serverpool
NAME=Free
ACTIVE_SERVERS=c03n04

NAME=Generic
ACTIVE_SERVERS=

NAME=ora.dev
ACTIVE_SERVERS=c03n01 c03n02

NAME=ora.my_app_pool
ACTIVE_SERVERS=c03n03

NAME=ora.rep
ACTIVE_SERVERS=

[grid@c03n01 ~]$
```

7. Examine the provided action script (`/stage/labs/5_3/action.scr`). Soon you will use this script to create a series of highly available application resources on a Flex Cluster Leaf Node. All the application resources perform the following set of simple tasks.
   - When the application resource is started, it creates an empty file at `/tmp/<resource_name>` on the node running the application resource.
   - When the application resource is stopped or cleaned, the file at `/tmp/<resource_name>` is deleted.
   - When the application resource is checked, a test is performed to check the existence of the file at `/tmp/<resource_name>`.
   - In addition, regardless of the action performed, entries are written to the CRSD agent trace file at `/u01/app/grid/diag/crs/<hostname>/crs/trace/crsd_scriptagent_grid.trc`.

```
[grid@c03n01 ~]$ more /stage/labs/5_3/action.scr
#!/bin/sh
TOUCH=/bin/touch
RM=/bin/rm
PATH_NAME=/tmp/$_CRS_NAME

#
# These messages go into the CRSD agent trace file.
echo " *******   `date` ********** "
echo "Action script '$_CRS_ACTION_SCRIPT' for
resource[$_CRS_NAME] called for action $1"
#

case "$1" in
  'start')
     echo "START entry point has been called.."
     echo "Creating the file: $PATH_NAME"
     $TOUCH $PATH_NAME
     exit 0
     ;;

  'stop')
     echo "STOP entry point has been called.."
     echo "Deleting the file: $PATH_NAME"
     $RM $PATH_NAME
     exit 0
     ;;

  'check')
    echo "CHECK entry point has been called.."
    if [ -e $PATH_NAME ]; then
```

```
        echo "Check -- SUCCESS"
        exit 0
    else
        echo "Check -- FAILED"
        exit 1
    fi
    ;;

  'clean')
     echo "CLEAN entry point has been called.."
     echo "Deleting the file: $PATH_NAME"
     $RM -f $PATH_NAME
     exit 0
     ;;

esac

[grid@c03n01 ~]$
```

8. Examine the file at `/stage/labs/5_3/create_res.sh`. This file contains the commands that you will next use to create three application resources named `my_dep_res1`, `my_dep_res2`, and `my_resource`. Notice that all three resources are associated with the `my_app_pool` server pool. Notice also that `my_resource` has a mandatory (hard) dependency on `my_dep_res1` and an optional (soft) dependency on `my_dep_res2`. Finally, notice that `my_resource` also depends on the RAC database (`ora.c03orcl.db`) that you used previous practices. This illustrates how you can unify the management of databases and applications within a Flex Cluster.

```
[grid@c03n01 ~]$ cat /stage/labs/5_3/create_res.sh
#!/bin/bash
set -v

crsctl add resource my_dep_res1 -type cluster_resource -attr
"ACTION_SCRIPT=/stage/labs/5_3/action.scr,PLACEMENT=restricted,S
ERVER_POOLS=ora.my_app_pool"

crsctl add resource my_dep_res2 -type cluster_resource -attr
"ACTION_SCRIPT=/stage/labs/5_3/action.scr,PLACEMENT=restricted,S
ERVER_POOLS=ora.my_app_pool"

crsctl add resource my_resource -type cluster_resource -attr
"ACTION_SCRIPT=/stage/labs/5_3/action.scr,PLACEMENT=restricted,S
ERVER_POOLS=ora.my_app_pool,START_DEPENDENCIES='hard(my_dep_res1
, global:uniform:ora.c03orcl.db) pullup:always(my_dep_res1,
global:ora.c03orcl.db)
weak(my_dep_res2)',STOP_DEPENDENCIES='hard(my_dep_res1,
global:ora.c03orcl.db)'"

[grid@c03n01 ~]$
```

9. Execute `create_res.sh` to create the application resources.

```
[grid@c03n01 ~]$ /stage/labs/5_3/create_res.sh

crsctl add resource my_dep_res1 -type cluster_resource -attr
"ACTION_SCRIPT=/stage/labs/5_3/action.scr,PLACEMENT=restricted,S
ERVER_POOLS=ora.my_app_pool"

crsctl add resource my_dep_res2 -type cluster_resource -attr
"ACTION_SCRIPT=/stage/labs/5_3/action.scr,PLACEMENT=restricted,S
ERVER_POOLS=ora.my_app_pool"

crsctl add resource my_resource -type cluster_resource -attr
"ACTION_SCRIPT=/stage/labs/5_3/action.scr,PLACEMENT=restricted,S
ERVER_POOLS=ora.my_app_pool,START_DEPENDENCIES='hard(my_dep_res1
, global:uniform:ora.c03orcl.db) pullup:always(my_dep_res1,
global:ora.c03orcl.db)
weak(my_dep_res2)',STOP_DEPENDENCIES='hard(my_dep_res1,
global:ora.c03orcl.db)'"

[grid@c03n01 ~]$
```

10. Examine the newly created cluster resources. Note that currently the resources exist but have not been started.

```
[grid@c03n01 ~]$ crsctl status resource -t -w "NAME co my"
--------------------------------------------------------------------
Name           Target  State        Server
State details
--------------------------------------------------------------------
Cluster Resources
--------------------------------------------------------------------
my_dep_res1
      1        OFFLINE OFFLINE
STABLE
my_dep_res2
      1        OFFLINE OFFLINE
STABLE
my_resource
      1        OFFLINE OFFLINE
STABLE
--------------------------------------------------------------------
[grid@c03n01 ~]$
```

11. Establish another terminal session connected to `c03n01` as the `oracle` user and configure the terminal environment as shown below.

```
$ ssh oracle@c03n01
[oracle@c03n01 ~]$ . oraenv
ORACLE_SID = [oracle] ? c03orcl
The Oracle base has been set to /u01/app/oracle
[oracle@c03n01 ~]$
```

12. So that you can exercise the dependency between `my_resource` and your RAC database, stop the `c03orcl` database.

```
[oracle@c03n01 ~]$ srvctl stop database -d c03orcl
[oracle@c03n01 ~]$
```

13. Back in your grid terminal session, start the `my_resource` application resource. Notice that `my_dep_res1` and `my_dep_res2` are started automatically to fulfill the dependency definitions and that all three resources are started on the server associated with the `my_app_pool` server pool. Note also that the `c03orcl` database is also started as defined in the dependency definitions for the `my_resource` application resource. This illustrates how a Flex Cluster can be used to manage a RAC database and associated application resources.

```
[grid@c03n01 ~]$ crsctl start resource my_resource
CRS-2672: Attempting to start 'my_dep_res1' on 'c03n03'
CRS-2672: Attempting to start 'my_dep_res2' on 'c03n03'
CRS-2672: Attempting to start 'ora.c03orcl.db' on 'c03n02'
CRS-2672: Attempting to start 'ora.c03orcl.db' on 'c03n01'
CRS-2676: Start of 'my_dep_res2' on 'c03n03' succeeded
CRS-2676: Start of 'my_dep_res1' on 'c03n03' succeeded
CRS-2676: Start of 'ora.c03orcl.db' on 'c03n02' succeeded
CRS-2676: Start of 'ora.c03orcl.db' on 'c03n01' succeeded
CRS-2672: Attempting to start 'my_resource' on 'c03n03'
CRS-2676: Start of 'my_resource' on 'c03n03' succeeded
[grid@c03n01 ~]$
```

14. Re-examine the application resources and confirm that they are all online.

```
[grid@c03n01 ~]$ crsctl status resource -t -w "NAME co my"
--------------------------------------------------------------------------------
Name           Target  State        Server       State details
--------------------------------------------------------------------------------
Cluster Resources
--------------------------------------------------------------------------------
my_dep_res1
      1        ONLINE  ONLINE       c03n03       STABLE
my_dep_res2
      1        ONLINE  ONLINE       c03n03       STABLE
my_resource
      1        ONLINE  ONLINE       c03n03       STABLE
--------------------------------------------------------------------------------
[grid@c03n01 ~]$
```

15. Check the files under `/tmp` on the node running the application resources. You will see a series of empty files, each bearing the name of one of the application resources.

```
[grid@c03n01 ~]$ ssh c03n03 ls -l /tmp/my*
-rw-r--r-- 1 grid oinstall 0 Aug 28 11:26 /tmp/my_dep_res1
-rw-r--r-- 1 grid oinstall 0 Aug 28 11:26 /tmp/my_dep_res2
-rw-r--r-- 1 grid oinstall 0 Aug 28 11:28 /tmp/my_resource
[grid@c03n01 ~]$
```

16. Using the `oracle` terminal session, confirm also that your RAC database (`c03orcl`) started as part of starting the `my_resource` application resource.

```
[oracle@c03n01 ~]$ srvctl status database -d c03orcl
Instance c03orcl_1 is running on node c03n02
Instance c03orcl_2 is running on node c03n01
[oracle@c03n01 ~]$
```

17. Now, shut down Oracle Clusterware on the node hosting the application resources.

```
[grid@c03n01 ~]$ su -c "crsctl stop cluster -n c03n03"
Password: <oracle>
CRS-2673: Attempting to stop 'ora.crsd' on 'c03n03'
CRS-2790: Starting shutdown of Cluster Ready Services-managed
resources on 'c03n03'
CRS-2673: Attempting to stop 'my_dep_res2' on 'c03n03'
CRS-2673: Attempting to stop 'my_resource' on 'c03n03'
CRS-2677: Stop of 'my_dep_res2' on 'c03n03' succeeded
CRS-2677: Stop of 'my_resource' on 'c03n03' succeeded
CRS-2673: Attempting to stop 'my_dep_res1' on 'c03n03'
CRS-2677: Stop of 'my_dep_res1' on 'c03n03' succeeded
CRS-2792: Shutdown of Cluster Ready Services-managed resources
on 'c03n03' has completed
CRS-2677: Stop of 'ora.crsd' on 'c03n03' succeeded
CRS-2673: Attempting to stop 'ora.cluster_interconnect.haip' on
'c03n03'
CRS-2673: Attempting to stop 'ora.ctssd' on 'c03n03'
CRS-2673: Attempting to stop 'ora.evmd' on 'c03n03'
CRS-2673: Attempting to stop 'ora.storage' on 'c03n03'
CRS-2677: Stop of 'ora.storage' on 'c03n03' succeeded
CRS-2677: Stop of 'ora.cluster_interconnect.haip' on 'c03n03'
succeeded
CRS-2677: Stop of 'ora.ctssd' on 'c03n03' succeeded
CRS-2677: Stop of 'ora.evmd' on 'c03n03' succeeded
CRS-2673: Attempting to stop 'ora.cssd' on 'c03n03'
CRS-2677: Stop of 'ora.cssd' on 'c03n03' succeeded
[grid@c03n01 ~]$
```

18. Examine the status of the server pools. Notice that the previously unallocated Leaf Node has been moved to the `my_app_pool` server pool.

```
[grid@c03n01 ~]$ crsctl status serverpool
NAME=Free
ACTIVE_SERVERS=

NAME=Generic
ACTIVE_SERVERS=

NAME=ora.dev
ACTIVE_SERVERS=c03n01 c03n02

NAME=ora.my_app_pool
ACTIVE_SERVERS=c03n04

NAME=ora.rep
ACTIVE_SERVERS=

[grid@c03n01 ~]$
```

19. Examine the status of the application resources. Notice that they have been failed over to the surviving Leaf Node.

```
[grid@c03n01 ~]$ crsctl status resource -t -w "NAME co my"
--------------------------------------------------------------------------------
Name           Target  State        Server       State details
--------------------------------------------------------------------------------
Cluster Resources
--------------------------------------------------------------------------------
my_dep_res1
      1        ONLINE  ONLINE       c03n04       STABLE
my_dep_res2
      1        ONLINE  ONLINE       c03n04       STABLE
my_resource
      1        ONLINE  ONLINE       c03n04       STABLE
--------------------------------------------------------------------------------
[grid@c03n01 ~]$
```

20. Examine the contents of the `/tmp` directory on both Leaf Nodes. Notice that the files you saw in step 15 no longer exist and that there is a newer set of files on the other Leaf Node.

```
[grid@c03n01 ~]$ ssh c03n03 ls -l /tmp/my*
ls: cannot access /tmp/my*: No such file or directory
[grid@c03n01 ~]$ ssh c03n04 ls -l /tmp/my*
-rw-r--r-- 1 grid oinstall 0 Aug 28 11:29 /tmp/my_dep_res1
-rw-r--r-- 1 grid oinstall 0 Aug 28 11:29 /tmp/my_dep_res2
-rw-r--r-- 1 grid oinstall 0 Aug 28 11:29 /tmp/my_resource
[grid@c03n01 ~]$
```

21. Restart Oracle Clusterware on the inactive Leaf Node.

```
[grid@c03n01 ~]$ su -c "crsctl start cluster -n c03n03"
Password: <oracle>
CRS-2672: Attempting to start 'ora.evmd' on 'c03n03'
CRS-2672: Attempting to start 'ora.cssdmonitor' on 'c03n03'
CRS-2676: Start of 'ora.cssdmonitor' on 'c03n03' succeeded
CRS-2672: Attempting to start 'ora.cssd' on 'c03n03'
CRS-2672: Attempting to start 'ora.diskmon' on 'c03n03'
CRS-2676: Start of 'ora.diskmon' on 'c03n03' succeeded
CRS-2676: Start of 'ora.evmd' on 'c03n03' succeeded
CRS-2676: Start of 'ora.cssd' on 'c03n03' succeeded
CRS-2672: Attempting to start 'ora.cluster_interconnect.haip' on
'c03n03'
CRS-2672: Attempting to start 'ora.storage' on 'c03n03'
CRS-2672: Attempting to start 'ora.ctssd' on 'c03n03'
CRS-2676: Start of 'ora.storage' on 'c03n03' succeeded
CRS-2676: Start of 'ora.cluster_interconnect.haip' on 'c03n03'
succeeded
CRS-2676: Start of 'ora.ctssd' on 'c03n03' succeeded
CRS-2672: Attempting to start 'ora.crsd' on 'c03n03'
CRS-2676: Start of 'ora.crsd' on 'c03n03' succeeded
[grid@c03n01 ~]$
```

22. Examine the status of the server pools. Notice that the node you just restarted is placed in the `Free` pool.

    Note: You may need to wait for a short period before the restarted leaf node appears in the `Free` pool. If required, periodically re-examine the server pool status until you see the newly restarted node.

```
[grid@c03n01 ~]$ crsctl status serverpool
NAME=Free
ACTIVE_SERVERS=c03n03

NAME=Generic
ACTIVE_SERVERS=
```

```
NAME=ora.dev
ACTIVE_SERVERS=c03n01 c03n02

NAME=ora.my_app_pool
ACTIVE_SERVERS=c03n04

NAME=ora.rep
ACTIVE_SERVERS=

[grid@c03n01 ~]$
```

Congratulations! You have successfully configured highly available application resources on Flex Cluster Leaf Nodes.

23. The next practice does not use the Flex Cluster leaf nodes. Therefore, to conserve server resources and facilitate consistent performance in your practice environment, shut down Oracle Clusterware on `c03n03` and `c03n04`.

```
[grid@c03n01 ~]$ ssh root@c03n03
"/u01/app/12.1.0/grid/bin/crsctl stop crs"
root@c03n03's password: <oracle>
CRS-2791: Starting shutdown of Oracle High Availability
Services-managed resources on 'c03n03'
...
CRS-2677: Stop of 'ora.gipcd' on 'c03n03' succeeded
CRS-2677: Stop of 'ora.mdnsd' on 'c03n03' succeeded
CRS-2793: Shutdown of Oracle High Availability Services-managed
resources on 'c03n03' has completed
CRS-4133: Oracle High Availability Services has been stopped.
[grid@c03n01 ~]$ ssh root@c03n04
"/u01/app/12.1.0/grid/bin/crsctl stop crs"
root@c03n04's password: <oracle>
CRS-2791: Starting shutdown of Oracle High Availability
Services-managed resources on 'c03n04'
...
CRS-2677: Stop of 'ora.gipcd' on 'c03n04' succeeded
CRS-2677: Stop of 'ora.mdnsd' on 'c03n04' succeeded
CRS-2793: Shutdown of Oracle High Availability Services-managed
resources on 'c03n04' has completed
CRS-4133: Oracle High Availability Services has been stopped.
[grid@c03n01 ~]$
```

24. Exit all the terminal sessions.

Practices for Lesson 6: Oracle Database In-Memory

Chapter 6

Practices for Lesson 6: Overview

Practice Overview

In this practice, you will use the new Oracle Database In-Memory features on a 2-node Oracle RAC database.

Practice 6-1: Using Oracle Database In-Memory in conjunction with Oracle RAC

Overview

In this practice, you will use the new Oracle Database In-Memory features on a 2-node Oracle RAC database. You will see how data is populated in the In-Memory Column Store (IMCS) across multiple instances and how queries can use data that is distributed across multiple instances.

Tasks

Part 1: Enabling Oracle Database In-Memory

In the first part of this practice, you will prepare your Oracle RAC database to use Oracle Database In-Memory.

1. Establish a terminal session connected to `c03n01` as the `oracle` user and configure the terminal environment as shown below.

```
$ ssh oracle@c03n01
[oracle@c03n01 ~]$ . oraenv
ORACLE_SID = [oracle] ? c03orcl
The Oracle base has been set to /u01/app/oracle
[oracle@c03n01 ~]$
```

2. Examine the status of the `c03orcl` database. Take note of the instance name on each node. Be aware that your environment may differ from the example below.

```
[oracle@c03n01 ~]$ srvctl status database -d c03orcl
Instance c03orcl_1 is running on node c03n02
Instance c03orcl_2 is running on node c03n01
[oracle@c03n01 ~]$
```

3. Configure the `ORACLE_SID` environment variable so that you can connect to the database instance on `c03n01`.

```
[oracle@c03n01 ~]$ export ORACLE_SID=c03orcl_2
[oracle@c03n01 ~]$
```

4. Prepare to enable the Oracle Database In-Memory capability by adjusting the following instance parameter settings:
   - `SGA_TARGET` specifies the total size of all SGA components, including the in-memory store. While this setting does not directly control Oracle Database In-Memory, it is commonly used to ensure that the SGA is large enough to contain the IMCS.
   - `INMEMORY_SIZE` sets the size of the IMCS on a database instance. A nonzero value is required to enable Oracle Database In-Memory. The database must be restarted after setting this parameter to enable the IMCS. The minimum size to which this parameter can be set is 100 MB.
   - `INMEMORY_MAX_POPULATE_SERVERS` specifies the maximum number of background populate servers to use for IMCS population. A nonzero value is required to enable IMCS population. Care should be taken to ensure that this parameter is not set so high that population tasks monopolize the system CPU resources.
   - `PARALLEL_DEGREE_POLICY` must be set to `AUTO` to enable in-memory parallel execution.

```
[oracle@c03n01 ~]$ sqlplus / as sysdba

SQL*Plus: Release 12.1.0.2.0 Production ...

SQL> alter system set sga_target=2000M scope=spfile;

System altered.

SQL> alter system set inmemory_size=1000M scope=spfile;

System altered.

SQL> alter system set inmemory_max_populate_servers=2
scope=spfile;

System altered.

SQL> alter system set parallel_degree_policy=AUTO scope=spfile;

System altered.

SQL> exit
Disconnected from Oracle Database 12c Enterprise Edition ...
[oracle@c03n01 ~]$
```

5. Restart the database to enable the Oracle Database In-Memory capability.

```
[oracle@c03n01 ~]$ srvctl stop database -d c03orcl
[oracle@c03n01 ~]$ srvctl start database -d c03orcl
[oracle@c03n01 ~]$
```

6. Connect to your database as the `mem` database user, which contains the schema objects that you will use throughout this practice.

```
[oracle@c03n01 ~]$ sqlplus mem/mem

SQL*Plus: Release 12.1.0.2.0 Production ...

SQL>
```

7. Examine the database instance parameter settings associated with Oracle Database In-Memory and confirm that the settings you made earlier are now enabled.

```
SQL> show parameter inmemory

NAME                                 TYPE        VALUE
------------------------------------ ----------- --------------
inmemory_clause_default              string
```

```
inmemory_force                         string      DEFAULT
inmemory_max_populate_servers          integer     2
inmemory_query                         string      ENABLE
inmemory_size                          big integer 1008M
inmemory_trickle_repopulate_servers_   integer     1
percent
optimizer_inmemory_aware               boolean     TRUE
SQL>
```

Part 2: Populating tables in the IMCS

In the second part of this practice, you will examine how tables are populated in the IMCS.

8. Examine the tables in the `mem` schema (you may use the script at `/stage/labs/6_1/tabq.sql` if you prefer). At this point, note that none of them are configured to use Oracle Database In-Memory (`INMEMORY=DISABLED`).

```
SQL> set pagesize 100
SQL> select table_name, INMEMORY, INMEMORY_PRIORITY "IM_PRIORITY",
  2  INMEMORY_DISTRIBUTE "IM_DISTRIBUTE",
  3  INMEMORY_COMPRESSION "IM_COMPRESSION",
  4  INMEMORY_DUPLICATE "IM_DUPLICATE" from user_tables;

TABLE_NAME
--------------------------------------------------------------------------------
INMEMORY IM_PRIOR IM_DISTRIBUTE   IM_COMPRESSION    IM_DUPLICATE
-------- -------- -------------- ----------------- -------------
DATE_DIM
DISABLED

ORDERLINE
DISABLED

ORDERLINE_HISTORY
DISABLED

SUPPLIER
DISABLED


SQL>
```

9. Modify the `orderline` table to use Oracle Database In-Memory.

```
SQL> alter table orderline inmemory;

Table altered.

SQL>
```

10. Re-examine the `mem` schema tables (you may use the script at `/stage/labs/6_1/tabq.sql` if you prefer). Notice the default in-memory configuration settings that are now associated with the `orderline` table.

```
SQL> select table_name, INMEMORY, INMEMORY_PRIORITY "IM_PRIORITY",
  2  INMEMORY_DISTRIBUTE "IM_DISTRIBUTE",
  3  INMEMORY_COMPRESSION "IM_COMPRESSION",
  4  INMEMORY_DUPLICATE "IM_DUPLICATE" from user_tables;

TABLE_NAME
--------------------------------------------------------------------------------
INMEMORY IM_PRIOR IM_DISTRIBUTE   IM_COMPRESSION    IM_DUPLICATE
-------- -------- --------------- ----------------- -------------
SUPPLIER
DISABLED

ORDERLINE_HISTORY
DISABLED

ORDERLINE
ENABLED  NONE     AUTO            FOR QUERY LOW     NO DUPLICATE

DATE_DIM
DISABLED


SQL>
```

11. Examine `gv$im_segments`. This view contains information about the segments that are populated in the IMCS memory areas on all of the instances in your Oracle RAC database. At this point, it will be empty because no segments are populated.

```
SQL> select * from gv$im_segments;

no rows selected

SQL>
```

Notes:

The `orderline` table is not automatically populated because the `INMEMORY_PRIORITY` attribute is set to `NONE`. Objects with `INMEMORY_PRIORITY=NONE` are only populated after they are initially scanned following a database restart. Alternatively, `INMEMORY_PRIORITY` can be set to `LOW`, `MEDIUM`, `HIGH`, or `CRITICAL`. Objects with `INMEMORY_PRIORITY` set to a value other than `NONE` are automatically populated when the in-memory population process next wakes up, which occurs approximately every two minutes. `CRITICAL` objects are populated first, then `HIGH`, `MEDIUM`, and `LOW` objects in that order.

The population process does not change for different `INMEMORY_PRIORITY` settings; that is, `CRITICAL` priority object population does not use more resources and does not complete faster than `LOW` priority object population. The only difference is the order in which the objects are populated. This ordering is most critical in situations where the IMCS size is not large enough to accommodate all of the in-memory enabled objects.

12. Execute a query that scans the `orderline` table.

```
SQL> select count(*) from orderline;

  COUNT(*)
----------
    700000

SQL>
```

13. Re-examine `gv$im_segments`. The in-memory population process wakes periodically to perform outstanding population requests, so you may need to re-examine `gv$im_segments` periodically until you see an entry relating to the `orderline` table. Notice that the entire `orderline` table is contained in the IMCS on only one instance (instance 2 in the example output below).

```
SQL> select * from gv$im_segments;

no rows selected

SQL> select * from gv$im_segments;

   INST_ID
----------
OWNER
--------------------------------------------------------------------------------
SEGMENT_NAME
--------------------------------------------------------------------------------
PARTITION_NAME
--------------------------------------------------------------------------------
SEGMENT_TYPE       TABLESPACE_NAME                 INMEMORY_SIZE      BYTES
------------------ ------------------------------ ------------- ----------
BYTES_NOT_POPULATED POPULATE_ INMEMORY INMEMORY_DISTRI INMEMORY_DUPL
------------------- --------- -------- --------------- -------------
INMEMORY_COMPRESS     CON_ID
----------------- ----------
         2
MEM
ORDERLINE

TABLE              EXAMPLE                              34799616   83886080
                  0 COMPLETED NONE     AUTO            NO DUPLICATE
FOR QUERY LOW              0


SQL>
```

14. Modify the `orderline` table to use the `DUPLICATE ALL` in-memory attribute setting.

```
SQL> alter table orderline inmemory duplicate all;

Table altered.

SQL>
```

15. Re-examine the `mem` schema tables (you may use the script at `/stage/labs/6_1/tabq.sql` if you prefer). Confirm that the `DUPLICATE ALL` in-memory attribute setting is associated with the `orderline` table.

```
SQL> select table_name, INMEMORY, INMEMORY_PRIORITY "IM_PRIORITY",
  2  INMEMORY_DISTRIBUTE "IM_DISTRIBUTE",
  3  INMEMORY_COMPRESSION "IM_COMPRESSION",
  4  INMEMORY_DUPLICATE "IM_DUPLICATE" from user_tables;

TABLE_NAME
--------------------------------------------------------------------------------
INMEMORY IM_PRIOR IM_DISTRIBUTE   IM_COMPRESSION    IM_DUPLICATE
-------- -------- --------------- ----------------- -------------
SUPPLIER
DISABLED

ORDERLINE_HISTORY
DISABLED

ORDERLINE
ENABLED  NONE     AUTO            FOR QUERY LOW     DUPLICATE ALL

DATE_DIM
DISABLED


SQL>
```

16. Execute a query that scans the `orderline` table.

```
SQL> select count(*) from orderline;

  COUNT(*)
----------
    700000

SQL>
```

17. Re-examine `gv$im_segments` periodically until you see an entry relating to the `orderline` table.

    Note: The entire `orderline` table is contained in the IMCS on only one instance (instance 2 in the example output below) even though `INMEMORY_DUPLICATE=DUPLICATE ALL`. This is because data duplication across multiple Oracle RAC instances is only supported when using Oracle Database In-Memory on Oracle Engineered Systems, such as Exadata

Database Machine. On all other platforms, the INMEMORY_DUPLICATE setting is ignored and effectively set to NO DUPLICATE.

```
SQL> select * from gv$im_segments;

no rows selected

SQL> select * from gv$im_segments;

   INST_ID
----------
OWNER
------------------------------------------------------------------------------
SEGMENT_NAME
------------------------------------------------------------------------------
PARTITION_NAME
------------------------------------------------------------------------------
SEGMENT_TYPE       TABLESPACE_NAME                 INMEMORY_SIZE      BYTES
------------------ ------------------------------ ------------- ----------
BYTES_NOT_POPULATED POPULATE_ INMEMORY INMEMORY_DISTRI INMEMORY_DUPL
------------------- --------- -------- --------------- -------------
INMEMORY_COMPRESS     CON_ID
----------------- ----------
         2
MEM
ORDERLINE

TABLE              EXAMPLE                                34799616   83886080
                  0 COMPLETED NONE     AUTO            DUPLICATE ALL
FOR QUERY LOW               0


SQL>
```

18. You can set the inmemory_clause_default parameter to specify customized defaults for in-memory attribute settings. Use the following command to specify a default population priority of MEDIUM and to explicitly set the NO DUPLICATE attribute.

```
SQL> alter system set
inmemory_clause_default="PRIORITY MEDIUM NO DUPLICATE";

System altered.

SQL>
```

19. Reconfigure the orderline table to use Oracle Database In-Memory.

```
SQL> alter table orderline inmemory;

Table altered.

SQL>
```

20. Re-examine the table properties in `user_tables` (you may use the script at `/stage/labs/6_1/tabq.sql` if you prefer) and `gv$im_segements` to ensure that the customized attribute defaults are being used, and also that the table is populated automatically. Use the output from `gv$im_segments` to determine the instance where the `orderline` table is populated (instance 2 in the following example). Notice that you may need to wait for a couple of minutes for automatic population to complete.

```
SQL> select table_name, INMEMORY, INMEMORY_PRIORITY "IM_PRIORITY",
  2  INMEMORY_DISTRIBUTE "IM_DISTRIBUTE",
  3  INMEMORY_COMPRESSION "IM_COMPRESSION",
  4  INMEMORY_DUPLICATE "IM_DUPLICATE" from user_tables;
TABLE_NAME
--------------------------------------------------------------------------------
INMEMORY IM_PRIOR IM_DISTRIBUTE   IM_COMPRESSION    IM_DUPLICATE
-------- -------- --------------- ----------------- -------------
SUPPLIER
DISABLED

ORDERLINE_HISTORY
DISABLED

ORDERLINE
ENABLED  MEDIUM   AUTO            FOR QUERY LOW     NO DUPLICATE

DATE_DIM
DISABLED


SQL> select * from gv$im_segments;

   INST_ID
----------
OWNER
--------------------------------------------------------------------------------
SEGMENT_NAME
--------------------------------------------------------------------------------
PARTITION_NAME
--------------------------------------------------------------------------------
SEGMENT_TYPE       TABLESPACE_NAME                INMEMORY_SIZE      BYTES
------------------ ------------------------------ ------------- ----------
BYTES_NOT_POPULATED POPULATE_ INMEMORY INMEMORY_DISTRI INMEMORY_DUPL
------------------- --------- -------- --------------- -------------
INMEMORY_COMPRESS     CON_ID
----------------- ----------
         2
MEM
ORDERLINE

TABLE              EXAMPLE                             34799616   83886080
                  0 COMPLETED MEDIUM   AUTO            NO DUPLICATE
FOR QUERY LOW              0


SQL>
```

21. Use the following query to determine the instance mapping for your database. Then, exit your SQL*Plus session.

```
SQL> select inst_id, instance_name, host_name from gv$instance;

   INST_ID INSTANCE_NAME
---------- ----------------
HOST_NAME
----------------------------------------------------------------
         2 c03orcl_2
c03n01.example.com

         1 c03orcl_1
c03n02.example.com


SQL> exit
Disconnected from Oracle Database 12c Enterprise Edition ...
[oracle@c03n01 ~]$
```

Part 3: Using Oracle Database In-Memory in conjunction with Oracle RAC to query small tables

With Oracle Database In-Memory, small tables are completely populated in the IMCS on only one Oracle RAC database instance. In the next part of this practice, you will examine how small tables are queried by using Oracle Database In-Memory in conjunction with Oracle RAC.

22. Using the information in the output from your previous steps, establish a SQL*Plus session connected to the database instance **that contains the `orderline` table in the IMCS.** Notice that in the example output for this practice, the `orderline` table is populated in the IMCS in instance `2`, which resides on `c03n01`. Be aware that this may differ in your practice environment and that you may need to establish a new terminal environment to perform this step.

```
[oracle@c03n01 ~]$ sqlplus mem/mem

SQL*Plus: Release 12.1.0.2.0 Production ...

SQL> select instance_number from v$instance;

INSTANCE_NUMBER
---------------
              2

SQL>
```

23. Examine the query plan for `select count(*) from orderline;`
Notice that the query execution plan contains the operation `TABLE ACCESS INMEMORY FULL`, which indicates that the optimizer intends to satisfy the query by using an in-memory scan of the `orderline` table.

```
SQL> explain plan for select count(*) from orderline;

Explained.

SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------
Plan hash value: 2194458862

------------------------------------------------------------------------------
| Id  | Operation                   | Name      | Rows  | Cost (%CPU)| Time
------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |           |     1 |   103   (1)| 0:00:01
|   1 |  SORT AGGREGATE             |           |     1 |            |
|   2 |   TABLE ACCESS INMEMORY FULL| ORDERLINE |   700K|   103   (1)| 0:00:01
------------------------------------------------------------------------------

Note
-----
   - automatic DOP: Computed Degree of Parallelism is 1 because of parallel
threshold

13 rows selected.

SQL>
```

24. Enable the SQL*Plus timing option (`set timing on`) and reset the session statistics by reconnecting to the database. Then, execute `select count(*) from orderline;`
Notice that the query returns almost instantaneously.

```
SQL> set timing on
SQL> connect mem/mem
Connected.
SQL> select count(*) from orderline;

  COUNT(*)
----------
    700000

Elapsed: 00:00:00.01
SQL>
```

25. Execute the following query to examine the current session statistics (you may use the script at `/stage/labs/6_1/statq.sql` if you prefer). Notice that the values for the `IM scan` statistics indicate that the query was satisfied by using rows from the IMCS. Finally, exit your SQL*Plus session.

    Note: The statistics may also indicate an amount of non-in-memory read activity. Remember that the statistics are session statistics, which include statistics for operations apart from the query execution. For example, the statistics query will by itself increment various statistical counters.

```
SQL> SELECT name, value FROM v$mystat m, v$statname n
  2  WHERE m.statistic# = n.statistic# AND value !=0
  3  AND (name like '%IM%'
  4  OR name in ('logical read bytes from cache','physical read bytes'));

NAME                                                                  VALUE
---------------------------------------------------------------- ----------
logical read bytes from cache                                       696320
physical read bytes                                                   8192
table scans (IM)                                                         1
table scan disk non-IMC rows gotten                                   3642
IM scan CUs memcompress for query low                                    2
session logical reads - IM                                           10097
IM scan bytes in-memory                                           33490629
IM scan bytes uncompressed                                        65769131
IM scan CUs columns accessed                                            34
IM scan CUs columns theoretical max                                     34
IM scan rows                                                        700000
IM scan rows valid                                                  700000
IM scan rows projected                                              700000
IM scan CUs split pieces                                                 2

14 rows selected.

Elapsed: 00:00:00.05
SQL> exit
Disconnected from Oracle Database 12c Enterprise Edition Release ...
[oracle@c03n01 ~]$
```

26. Establish a SQL*Plus session connected to the database instance **that DOES NOT contain the `orderline` table in the IMCS.** Notice that in the example output for this practice, the `orderline` table is populated in the IMCS in instance 2, which resides on `c03n01`. So, for this step and the following steps, the example commands are run on the database instance on `c03n02`. Be aware that this may differ in your practice environment.

```
$ ssh oracle@c03n02
[oracle@c03n02 ~]$ . oraenv
ORACLE_SID = [oracle] ? c03orcl
The Oracle base has been set to /u01/app/oracle
[oracle@c03n02 ~]$ srvctl status database -d c03orcl
Instance c03orcl_1 is running on node c03n02
Instance c03orcl_2 is running on node c03n01
```

```
[oracle@c03n02 ~]$ export ORACLE_SID=c03orcl_1
[oracle@c03n02 ~]$ sqlplus mem/mem

SQL*Plus: Release 12.1.0.2.0 Production ...

SQL> select instance_number from v$instance;

INSTANCE_NUMBER
---------------
              1

SQL>
```

27. Examine the query plan for `select count(*) from orderline;`
Confirm that the query execution plan again indicates the intention to satisfy the query using an in-memory scan.

```
SQL> explain plan for select count(*) from orderline;

Explained.

SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------
Plan hash value: 2194458862

------------------------------------------------------------------------------
| Id  | Operation                   | Name      | Rows  | Cost (%CPU)| Time
------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |           |     1 |   103   (2)| 0:00:01
|   1 |  SORT AGGREGATE             |           |     1 |            |
|   2 |   TABLE ACCESS INMEMORY FULL| ORDERLINE |   700K|   103   (2)| 0:00:01
------------------------------------------------------------------------------

Note
-----
   - automatic DOP: Computed Degree of Parallelism is 1 because of parallel
threshold

13 rows selected.

SQL>
```

28. Enable the SQL*Plus timing option (`set timing on`) and reset the session statistics by reconnecting to the database. Then, execute `select count(*) from orderline;` Notice that there is a discernible pause before the query result is displayed.

```
SQL> set timing on
SQL> connect mem/mem
Connected.
SQL> select count(*) from orderline;

  COUNT(*)
----------
    700000

Elapsed: 00:00:03.44
SQL>
```

29. Examine the current session statistics (you may use the script at `/stage/labs/6_1/statq.sql` if you prefer). Notice this time that the statistics indicate a lot of physical read activity. Also note that no value is displayed for `IM scan rows projected`, and the value for `table scan disk non-IMC rows gotten` exceeds the number of records in the `orderline` table. All of this indicates that the query did not use the IMCS and that physical disk reads were used instead.
Why did this occur? It is because parallel query must be used to access the IMCS on another node and in this case the optimizer chose to use a serial query (see the query plan above).

```
SQL> SELECT name, value FROM v$mystat m, v$statname n
  2  WHERE m.statistic# = n.statistic# AND value !=0
  3  AND (name like '%IM%'
  4  OR name in ('logical read bytes from cache','physical read bytes'));

NAME                                                                  VALUE
---------------------------------------------------------------- ----------
logical read bytes from cache                                     16793600
physical read bytes                                               78995456
table scan disk non-IMC rows gotten                                 706531
IM scan segments disk                                                    1

Elapsed: 00:00:00.18
SQL>
```

30. Reset the session statistics by reconnecting to the database, and execute:
`select /*+ parallel */ count(*) from orderline;`
Notice that this time the query is processed almost instantaneously.

```
SQL> connect mem/mem
Connected.
SQL> select /*+ parallel */ count(*) from orderline;

  COUNT(*)
----------
```

```
    700000

Elapsed: 00:00:00.18
SQL>
```

31. Examine the current session statistics (you may use the script at `/stage/labs/6_1/statq.sql` if you prefer). Note that now the statistics indicate no (or very few) physical read operations, and that the value for `IM scan rows projected` matches the number of rows in the `orderline` table. This indicates that the query was satisfied using rows from the IMCS, which resides on the other Oracle RAC database instance.

```
SQL> SELECT name, value FROM v$mystat m, v$statname n
  2  WHERE m.statistic# = n.statistic# AND value !=0
  3  AND (name like '%IM%'
  4  OR name in ('logical read bytes from cache','physical read bytes'));

NAME                                                                  VALUE
---------------------------------------------------------------- ----------
logical read bytes from cache                                      1130496
table scans (IM)                                                         5
table scan disk non-IMC rows gotten                                   3642
IM scan CUs memcompress for query low                                    5
session logical reads - IM                                           17732
IM scan bytes in-memory                                           56623911
IM scan bytes uncompressed                                       107660807
IM scan CUs columns accessed                                            85
IM scan CUs columns theoretical max                                     85
IM scan rows                                                       1145764
IM scan rows valid                                                  700000
IM scan rows range excluded                                         445764
IM scan rows projected                                              700000
IM scan CUs split pieces                                                 5

14 rows selected.

Elapsed: 00:00:00.02
SQL>
```

Part 4: Using Oracle Database In-Memory in conjunction with Oracle RAC to query large tables

With Oracle Database In-Memory, large tables are distributed across the in-memory stores on multiple Oracle RAC database instances. In the next part of this practice, you will use Oracle Database In-Memory in conjunction with Oracle RAC to examine how large tables are distributed across multiple instances and how they are queried.

32. Modify the `orderline_history` table to use Oracle Database In-Memory, and examine the in-memory table attributes (you may use the script at `/stage/labs/6_1/tabq.sql` if you prefer).

```
SQL> alter table orderline_history inmemory;

Table altered.

Elapsed: 00:00:00.84
SQL> set pagesize 100
SQL> select table_name, INMEMORY, INMEMORY_PRIORITY "IM_PRIORITY",
  2  INMEMORY_DISTRIBUTE "IM_DISTRIBUTE",
  3  INMEMORY_COMPRESSION "IM_COMPRESSION",
  4  INMEMORY_DUPLICATE "IM_DUPLICATE" from user_tables;

TABLE_NAME
--------------------------------------------------------------------------------
INMEMORY IM_PRIOR IM_DISTRIBUTE   IM_COMPRESSION    IM_DUPLICATE
-------- -------- --------------- ----------------- -------------
DATE_DIM
DISABLED

ORDERLINE
ENABLED  MEDIUM   AUTO            FOR QUERY LOW     NO DUPLICATE

ORDERLINE_HISTORY
ENABLED  MEDIUM   AUTO            FOR QUERY LOW     NO DUPLICATE

SUPPLIER
DISABLED


Elapsed: 00:00:00.29
SQL>
```

Note: The `INMEMORY_DISTRIBUTE` attribute is set to `AUTO` by default. With this setting enabled, Oracle automatically decides how to distribute the table across multiple instances. Alternative options allow administrations to specify distribution by rowid, partition, and subpartition.

33. Periodically re-examine `gv$im_segments` until you see entries relating to the `orderline_history` table. Notice that the table is distributed across both instances. Using the information in `gv$segments`:
    - Verify that the entire `orderline_history` table is populated across both instances.
    - Determine how much of the `orderline_history` table is populated on each instance.
    - Determine the compression ratio achieved for each in-memory segment.

```
SQL> select * from gv$im_segments;

   INST_ID
----------
OWNER
--------------------------------------------------------------------------------
```

```
SEGMENT_NAME
--------------------------------------------------------------------------------
PARTITION_NAME
--------------------------------------------------------------------------------
SEGMENT_TYPE       TABLESPACE_NAME                INMEMORY_SIZE      BYTES
------------------ ------------------------------ ------------- ----------
BYTES_NOT_POPULATED POPULATE_ INMEMORY INMEMORY_DISTRI INMEMORY_DUPL
------------------- --------- -------- -------------- -------------
INMEMORY_COMPRESS     CON_ID
----------------- ----------
         1
MEM
ORDERLINE_HISTORY

TABLE              EXAMPLE                              68419584  637534208
          471244800 COMPLETED MEDIUM   AUTO            NO DUPLICATE
FOR QUERY LOW              0


         2
MEM
ORDERLINE_HISTORY

TABLE              EXAMPLE                             200867840  637534208
          162955264 COMPLETED MEDIUM   AUTO            NO DUPLICATE
FOR QUERY LOW              0


         2
MEM
ORDERLINE

TABLE              EXAMPLE                              34799616   83886080
                  0 COMPLETED MEDIUM   AUTO            NO DUPLICATE
FOR QUERY LOW              0


Elapsed: 00:00:00.01
SQL>
```

34. Examine the query plan for `select count(*) from orderline_history;` Confirm that the query execution plan indicates the intention to satisfy the query using an in-memory scan. Also confirm that the execution plan indicates the use of parallel query.

```
SQL> explain plan for select /*+ parallel */ count(*) from orderline_history;

Explained.

Elapsed: 00:00:00.03
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
```

```
Plan hash value: 608897724

-------------------------------------------------------------------------------
| Id  | Operation                         | Name               | Rows  | Cost
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                  |                    |     1 |  5917
|   1 |  SORT AGGREGATE                   |                    |     1 |
|   2 |   PX COORDINATOR                  |                    |       |
|   3 |    PX SEND QC (RANDOM)            | :TQ10000           |     1 |
|   4 |     SORT AGGREGATE                |                    |     1 |
|   5 |      PX BLOCK ITERATOR            |                    |  5600K|  5917
|   6 |       TABLE ACCESS INMEMORY FULL| ORDERLINE_HISTORY |  5600K|  5917
-------------------------------------------------------------------------------

Note
-----
   - automatic DOP: Computed Degree of Parallelism is 2
   - parallel scans affinitized for inmemory

18 rows selected.

Elapsed: 00:00:00.06
SQL>
```

35. Reset the session statistics by reconnecting to the database, and execute `select count(*) from orderline_history;`
    Notice how quickly the query executes.

```
SQL> connect mem/mem
Connected.
SQL> select /*+ parallel */ count(*) from orderline_history;

  COUNT(*)
----------
   5600000

Elapsed: 00:00:00.18
SQL>
```

36. Examine the current session statistics (use the script at `/stage/labs/6_1/statq.sql` if you prefer). The statistics should show few (or no) physical reads, and the value for `IM scan rows projected` should match the number of rows in the `orderline_history` table, indicating the use of Oracle Database In-Memory. Also , because the `orderline_history` table is distributed across two Oracle RAC instances, the query must be accessing the IMCS on both instances.

```
SQL> SELECT name, value FROM v$mystat m, v$statname n
  2  WHERE m.statistic# = n.statistic# AND value !=0
  3  AND (name like '%IM%'
  4  OR name in ('logical read bytes from cache','physical read bytes'));

logical read bytes from cache                                  1523712
table scans (IM)                                                    12
```

```
table scan disk non-IMC rows gotten                              3642
IM scan CUs memcompress for query low                              12
session logical reads - IM                                      83029
IM scan bytes in-memory                                     279222679
IM scan bytes uncompressed                                  559746373
IM scan CUs columns accessed                                     204
IM scan CUs columns theoretical max                              204
IM scan rows                                                 5957478
IM scan rows valid                                           5600000
IM scan rows range excluded                                   357478
IM scan rows projected                                       5600000
IM scan CUs split pieces                                          14

14 rows selected.

Elapsed: 00:00:00.00
SQL>
```

Part 5: What happens to the IMCS when the number of Oracle RAC database instances changes

In the next part of this practice, you will examine the effect on the IMCS when the number of Oracle RAC database instances changes.

37. Reconfirm the identity of the database instance you are connected to.

```
SQL> select instance_number, instance_name from v$instance;

INSTANCE_NUMBER INSTANCE_NAME
--------------- ----------------
              1 c03orcl_1

Elapsed: 00:00:00.01
SQL>
```

38. Stop the Oracle RAC database instance that you are not connected to.

```
SQL> !srvctl stop instance -d c03orcl -instance c03orcl_2

SQL>
```

39. Examine `gv$im_segments`. Notice that immediately after the instance is stopped, the contents of the IMCS on that instance is obviously no longer available. As a result, the `orderline_history` table is only partially populated in the IMCS on the remaining instance. At this time, queries on the `orderline_history` table may be satisfied using a combination of reads from the available IMCS and normal reads (from the buffer cache or the physical disk).

```
SQL> select * from gv$im_segments;

   INST_ID
----------
OWNER
```

```
--------------------------------------------------------------------------------
SEGMENT_NAME
--------------------------------------------------------------------------------
PARTITION_NAME
--------------------------------------------------------------------------------
SEGMENT_TYPE       TABLESPACE_NAME                INMEMORY_SIZE      BYTES
------------------ ------------------------------ ------------- ----------
BYTES_NOT_POPULATED POPULATE_ INMEMORY INMEMORY_DISTRI INMEMORY_DUPL
------------------- --------- -------- --------------- -------------
INMEMORY_COMPRESS     CON_ID
----------------- ----------
         1
MEM
ORDERLINE_HISTORY

TABLE              EXAMPLE                             68419584  637534208
          471244800 COMPLETED MEDIUM   AUTO            NO DUPLICATE
FOR QUERY LOW              0


Elapsed: 00:00:00.01
SQL>
```

40. Wait for at least 5 minutes. Then, periodically re-examine `gv$im_segments` until you see that the `orderline` and `orderline_history` tables are both completely populated in the available instance.

    Note: Repopulation in response to a change in the number of Oracle RAC instances is not immediate. This is by design in order to avoid multiple repopulations when an instance is unavailable only for a short period of time.

```
SQL> select * from gv$im_segments;

   INST_ID
----------
OWNER
--------------------------------------------------------------------------------
SEGMENT_NAME
--------------------------------------------------------------------------------
PARTITION_NAME
--------------------------------------------------------------------------------
SEGMENT_TYPE       TABLESPACE_NAME                INMEMORY_SIZE      BYTES
------------------ ------------------------------ ------------- ----------
BYTES_NOT_POPULATED POPULATE_ INMEMORY INMEMORY_DISTRI INMEMORY_DUPL
------------------- --------- -------- --------------- -------------
INMEMORY_COMPRESS     CON_ID
----------------- ----------
         1
MEM
ORDERLINE_HISTORY

TABLE              EXAMPLE                           269221888  637534208
```

```
                  0 COMPLETED MEDIUM   AUTO            NO DUPLICATE
FOR QUERY LOW               0


         1
MEM
ORDERLINE

TABLE              EXAMPLE                              34799616   83886080
                  0 COMPLETED MEDIUM   AUTO            NO DUPLICATE
FOR QUERY LOW               0


Elapsed: 00:00:00.00
SQL>
```

41. Restart the stopped instance.

```
SQL> !srvctl start instance -d c03orcl -instance c03orcl_2


SQL>
```

42. Examine `gv$im_segments`. Notice that the existing contents of the IMCS are unchanged immediately after an instance is added.

```
SQL> select * from gv$im_segments;

   INST_ID
----------
OWNER
--------------------------------------------------------------------------------
SEGMENT_NAME
--------------------------------------------------------------------------------
PARTITION_NAME
--------------------------------------------------------------------------------
SEGMENT_TYPE       TABLESPACE_NAME                INMEMORY_SIZE      BYTES
------------------ ------------------------------ ------------- ----------
BYTES_NOT_POPULATED POPULATE_ INMEMORY INMEMORY_DISTRI INMEMORY_DUPL
------------------- --------- -------- --------------- -------------
INMEMORY_COMPRESS     CON_ID
----------------- ----------
         1
MEM
ORDERLINE_HISTORY

TABLE              EXAMPLE                             269221888  637534208
                  0 COMPLETED MEDIUM   AUTO            NO DUPLICATE
FOR QUERY LOW               0


         1
MEM
ORDERLINE
```

```
TABLE               EXAMPLE                                 34799616   83886080
                   0 COMPLETED MEDIUM   AUTO               NO DUPLICATE
FOR QUERY LOW               0



Elapsed: 00:00:00.23
SQL>
```

43. Wait for at least 5 minutes. Then, periodically re-examine `gv$im_segments` until you see that IMCS data is distributed across all of the available instances.

```
SQL> select * from gv$im_segments;

   INST_ID
----------
OWNER
--------------------------------------------------------------------------------
SEGMENT_NAME
--------------------------------------------------------------------------------
PARTITION_NAME
--------------------------------------------------------------------------------
SEGMENT_TYPE       TABLESPACE_NAME                INMEMORY_SIZE      BYTES
------------------ ------------------------------ ------------- ----------
BYTES_NOT_POPULATED POPULATE_ INMEMORY INMEMORY_DISTRI INMEMORY_DUPL
------------------- --------- -------- --------------- -------------
INMEMORY_COMPRESS     CON_ID
----------------- ----------
         1
MEM
ORDERLINE_HISTORY

TABLE               EXAMPLE                                 68419584  637534208
          471244800 COMPLETED MEDIUM   AUTO               NO DUPLICATE
FOR QUERY LOW               0

         2
MEM
ORDERLINE_HISTORY

TABLE               EXAMPLE                                200867840  637534208
          162955264 COMPLETED MEDIUM   AUTO               NO DUPLICATE
FOR QUERY LOW               0

         2
MEM
ORDERLINE

TABLE               EXAMPLE                                 34799616   83886080
                   0 COMPLETED MEDIUM   AUTO               NO DUPLICATE
FOR QUERY LOW               0


Elapsed: 00:00:00.04
SQL>
```

Part 6: Using Oracle Database In-Memory in conjunction with Oracle RAC to perform analytical queries

In the next part of this practice, you will examine how a typical analytical query, summarizing data from multiple joined tables, is processed using Oracle Database In-Memory in conjunction with Oracle RAC.

44. To establish a baseline, start a new database session and disable in-memory query capabilities. Then, execute an analytical query to determine the value of imports from each country for the orders in the `orderline_history` table (henceforth, the import summary query). If you like, use the script at `/stage/labs/6_1/summaryq.sql`. Note the time taken to execute the query. Finally, examine the session statistics (use the script at `/stage/labs/6_1/statq.sql` if you prefer) and confirm that the query does not use Oracle Database In-Memory.

    Note: Dynamic sampling is disabled so that the associated I/O does not show up in the session statistics. This is being done for the sake of the practice so that the statistics highlight the operation of Oracle Database In-Memory as much as possible. However, it is important to note that there is no specific need to disable dynamic sampling to use Oracle Database In-Memory.

```
SQL> connect mem/mem
Connected.
SQL> alter session set inmemory_query=disable;

Session altered.

Elapsed: 00:00:00.00
SQL> select /*+ parallel dynamic_sampling(0) */ s_nation "COUNTRY",
  2  sum(lo_supplycost) "SUPPLY_VALUE"
  3  from supplier, orderline_history
  4  where s_suppkey = lo_suppkey
  5  group by s_nation
  6  order by 2 desc;

COUNTRY         SUPPLY_VALUE
--------------- ------------
CHINA             2.4134E+10
INDONESIA         2.3917E+10
SAUDI ARABIA      2.3162E+10
JAPAN             2.2425E+10
INDIA             2.1977E+10
EGYPT             2.1626E+10
GERMANY           2.1480E+10
IRAQ              2.1216E+10
IRAN              2.1054E+10
BRAZIL            2.0941E+10
ALGERIA           2.0786E+10
MOZAMBIQUE        2.0641E+10
VIETNAM           2.0492E+10
UNITED KINGDOM    1.9772E+10
FRANCE            1.9598E+10
RUSSIA            1.9401E+10
```

```
UNITED STATES     1.9121E+10
PERU              1.8788E+10
KENYA             1.8455E+10
MOROCCO           1.8381E+10
CANADA            1.8243E+10
ARGENTINA         1.8210E+10
ETHIOPIA          1.8026E+10
JORDAN            1.6919E+10
ROMANIA           1.5123E+10

25 rows selected.

Elapsed: 00:01:14.68
SQL> SELECT name, value FROM v$mystat m, v$statname n
  2  WHERE m.statistic# = n.statistic# AND value !=0
  3  AND (name like '%IM%'
  4  OR name in ('logical read bytes from cache','physical read bytes'));

NAME                                                                  VALUE
---------------------------------------------------------------- ----------
logical read bytes from cache                                     1292140544
physical read bytes                                                631218176
gc IM expands                                                          76921
table scan disk non-IMC rows gotten                                  5618289
IM scan segments disk                                                     12

Elapsed: 00:00:00.02
SQL>
```

45. Examine the query plan for the baseline import summary query that does not use Oracle Database In-Memory (use the script at `/stage/labs/6_1/expsumq.sql` if you prefer).

```
SQL> explain plan for
  2  select /*+ parallel dynamic_sampling(0) */ s_nation "COUNTRY",
  3  sum(lo_supplycost) "SUPPLY_VALUE"
  4  from supplier, orderline_history
  5  where s_suppkey = lo_suppkey
  6  group by s_nation
  7  order by 2 desc;

Explained.

Elapsed: 00:00:00.01
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 1562805287

--------------------------------------------------------------------------------
| Id  | Operation                     | Name              | Rows  | Bytes |
```

```
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                    |    25 |   725 |
|   1 |  PX COORDINATOR              |                    |       |       |
|   2 |   PX SEND QC (ORDER)         | :TQ10002           |    25 |   725 |
|   3 |    SORT ORDER BY             |                    |    25 |   725 |
|   4 |     PX RECEIVE               |                    |    25 |   725 |
|   5 |      PX SEND RANGE           | :TQ10001           |    25 |   725 |
|   6 |       HASH GROUP BY          |                    |    25 |   725 |
|   7 |        PX RECEIVE            |                    |    25 |   725 |
|   8 |         PX SEND HASH         | :TQ10000           |    25 |   725 |
|   9 |          HASH GROUP BY       |                    |    25 |   725 |
|* 10 |           HASH JOIN          |                    |  5600K|   154M|
|  11 |            TABLE ACCESS FULL | SUPPLIER           |  2000 | 40000 |
|  12 |            PX BLOCK ITERATOR |                    |  5600K|    48M|
|  13 |             TABLE ACCESS FULL| ORDERLINE_HISTORY  |  5600K|    48M|
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

  10 - access("S_SUPPKEY"="LO_SUPPKEY")

Note
-----
   - automatic DOP: Computed Degree of Parallelism is 2
   - parallel scans affinitized for buffer cache

30 rows selected.

Elapsed: 00:00:00.32
SQL>
```

46. Start a new database session and examine the query plan for the import summary query while Oracle Database In-Memory is enabled (use the script at `/stage/labs/6_1/expsumq.sql` if you prefer). Compare this query plan with the previous one. Notice that now the query joins the `orderline_history` table, which is in-memory, with the `supplier` table, which is not in-memory.

```
SQL> connect mem/mem
Connected.
SQL> explain plan for
  2  select /*+ parallel dynamic_sampling(0) */ s_nation "COUNTRY",
  3  sum(lo_supplycost) "SUPPLY_VALUE"
  4  from supplier, orderline_history
  5  where s_suppkey = lo_suppkey
  6  group by s_nation
  7  order by 2 desc;

Explained.

Elapsed: 00:00:00.03
```

```
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------
Plan hash value: 1562805287

------------------------------------------------------------------------------------
| Id  | Operation                                | Name               | Rows  |
------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                         |                    |    25 |
|   1 |  PX COORDINATOR                          |                    |       |
|   2 |   PX SEND QC (ORDER)                     | :TQ10002           |    25 |
|   3 |    SORT ORDER BY                         |                    |    25 |
|   4 |     PX RECEIVE                           |                    |    25 |
|   5 |      PX SEND RANGE                       | :TQ10001           |    25 |
|   6 |       HASH GROUP BY                      |                    |    25 |
|   7 |        PX RECEIVE                        |                    |    25 |
|   8 |         PX SEND HASH                     | :TQ10000           |    25 |
|   9 |          HASH GROUP BY                   |                    |    25 |
|* 10 |           HASH JOIN                      |                    |  5600K|
|  11 |            TABLE ACCESS FULL             | SUPPLIER           |  2000 |
|  12 |            PX BLOCK ITERATOR             |                    |  5600K|
|  13 |             TABLE ACCESS INMEMORY FULL| ORDERLINE_HISTORY |  5600K|
------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

  10 - access("S_SUPPKEY"="LO_SUPPKEY")

Note
-----
   - automatic DOP: Computed Degree of Parallelism is 2
   - parallel scans affinitized for inmemory

30 rows selected.

Elapsed: 00:00:00.15
SQL>
```

47. Reset the session statistics by reconnecting to the database, and execute the import summary query (use the script at `/stage/labs/6_1/summaryq.sql` if you prefer). Notice how quickly the query executes compared with the baseline with the previous baseline result.

```
SQL> connect mem/mem
Connected.
SQL> select /*+ parallel dynamic_sampling(0) */ s_nation "COUNTRY",
  2  sum(lo_supplycost) "SUPPLY_VALUE"
  3  from supplier, orderline_history
  4  where s_suppkey = lo_suppkey
  5  group by s_nation
  6  order by 2 desc;

COUNTRY         SUPPLY_VALUE
--------------- ------------
CHINA             2.4134E+10
INDONESIA         2.3917E+10
SAUDI ARABIA      2.3162E+10
JAPAN             2.2425E+10
INDIA             2.1977E+10
EGYPT             2.1626E+10
GERMANY           2.1480E+10
IRAQ              2.1216E+10
IRAN              2.1054E+10
BRAZIL            2.0941E+10
ALGERIA           2.0786E+10
MOZAMBIQUE        2.0641E+10
VIETNAM           2.0492E+10
UNITED KINGDOM    1.9772E+10
FRANCE            1.9598E+10
RUSSIA            1.9401E+10
UNITED STATES     1.9121E+10
PERU              1.8788E+10
KENYA             1.8455E+10
MOROCCO           1.8381E+10
CANADA            1.8243E+10
ARGENTINA         1.8210E+10
ETHIOPIA          1.8026E+10
JORDAN            1.6919E+10
ROMANIA           1.5123E+10

25 rows selected.

Elapsed: 00:00:01.91
SQL>
```

48. Examine the session statistics (use the script at `/stage/labs/6_1/statq.sql` if you prefer). Compared with the earlier baseline query statistics, you will notice much less physical and logical read activity, and a large number of in-memory scans.

```
SQL> SELECT name, value FROM v$mystat m, v$statname n
  2  WHERE m.statistic# = n.statistic# AND value !=0
  3  AND (name like '%IM%'
  4  OR name in ('logical read bytes from cache','physical read bytes'));

NAME                                                                  VALUE
---------------------------------------------------------------- ----------
logical read bytes from cache                                      4186112
physical read bytes                                                  24576
table scans (IM)                                                        12
table scan disk non-IMC rows gotten                                  10852
IM scan CUs memcompress for query low                                   12
session logical reads - IM                                           83029
IM scan bytes in-memory                                          279222676
IM scan bytes uncompressed                                       559746373
IM scan CUs columns accessed                                            24
IM scan CUs columns theoretical max                                    204
IM scan rows                                                       5957478
IM scan rows valid                                                 5600000
IM scan rows range excluded                                         357478
IM scan rows projected                                             5600000
IM scan CUs split pieces                                                14

15 rows selected.

Elapsed: 00:00:00.01
SQL>
```

In the previous query you examined an example where a table in the IMCS is joined to a table that is not in the IMCS. Next, you will examine what happens when both tables are in memory.

49. Modify the `supplier` table to use Oracle Database In-Memory. Then, periodically examine `gv$im_segments` until you see that the `supplier` table is populated in the IMCS. Notice that the `supplier` table is very small, and is only populated in the IMCS on one database instance.

```
SQL> alter table supplier inmemory;

Table altered.

Elapsed: 00:00:00.02
SQL> select * from gv$im_segments;

   INST_ID
----------
OWNER
--------------------------------------------------------------------------------
SEGMENT_NAME
```

```
--------------------------------------------------------------------------------
PARTITION_NAME
--------------------------------------------------------------------------------
SEGMENT_TYPE       TABLESPACE_NAME                INMEMORY_SIZE      BYTES
------------------ ------------------------------ ------------- ----------
BYTES_NOT_POPULATED POPULATE_ INMEMORY INMEMORY_DISTRI INMEMORY_DUPL
------------------- --------- -------- --------------- -------------
INMEMORY_COMPRESS     CON_ID
----------------- ----------
         1
MEM
SUPPLIER

TABLE              EXAMPLE                                1179648     327680
                  0 COMPLETED MEDIUM   AUTO             NO DUPLICATE
FOR QUERY LOW              0

         1
MEM
ORDERLINE_HISTORY

TABLE              EXAMPLE                               68419584  637534208
          471244800 COMPLETED MEDIUM   AUTO             NO DUPLICATE
FOR QUERY LOW              0

         2
MEM
ORDERLINE_HISTORY

TABLE              EXAMPLE                              200867840  637534208
          162955264 COMPLETED MEDIUM   AUTO              NO DUPLICATE
FOR QUERY LOW              0

         2
MEM
ORDERLINE

TABLE              EXAMPLE                               34799616   83886080
                  0 COMPLETED MEDIUM   AUTO             NO DUPLICATE
FOR QUERY LOW              0


Elapsed: 00:00:00.01
SQL>
```

50. Re-examine the query execution plan for the import summary query (use the script at `/stage/labs/6_1/expsumq.sql` if you prefer). Notice that the query plan indicates memory scans for both tables; `supplier` and `orderline_history`.

```
SQL> explain plan for
  2  select /*+ parallel dynamic_sampling(0) */ s_nation "COUNTRY",
  3  sum(lo_supplycost) "SUPPLY_VALUE"
  4  from supplier, orderline_history
  5  where s_suppkey = lo_suppkey
  6  group by s_nation
  7  order by 2 desc;

Explained.

Elapsed: 00:00:00.03
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 1562805287

-------------------------------------------------------------------------------
| Id  | Operation                          | Name              | Rows  |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |                   |    25 |
|   1 |  PX COORDINATOR                    |                   |       |
|   2 |   PX SEND QC (ORDER)               | :TQ10002          |    25 |
|   3 |    SORT ORDER BY                   |                   |    25 |
|   4 |     PX RECEIVE                     |                   |    25 |
|   5 |      PX SEND RANGE                 | :TQ10001          |    25 |
|   6 |       HASH GROUP BY                |                   |    25 |
|   7 |        PX RECEIVE                  |                   |    25 |
|   8 |         PX SEND HASH               | :TQ10000          |    25 |
|   9 |          HASH GROUP BY             |                   |    25 |
|* 10 |           HASH JOIN                |                   |  5600K|
|  11 |            TABLE ACCESS INMEMORY FULL | SUPPLIER       |  2000 |
|  12 |            PX BLOCK ITERATOR       |                   |  5600K|
|  13 |             TABLE ACCESS INMEMORY FULL| ORDERLINE_HISTORY |  5600K|
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

  10 - access("S_SUPPKEY"="LO_SUPPKEY")

Note
-----
   - automatic DOP: Computed Degree of Parallelism is 2
   - parallel scans affinitized for inmemory
```

```
30 rows selected.

Elapsed: 00:00:00.09
SQL>
```

51. Reset the session statistics by reconnecting to the database, and execute the import summary query (use the script at `/stage/labs/6_1/summaryq.sql` if you prefer). Notice how quickly the query executes compared with previous executions.

```
SQL> connect mem/mem
Connected.
SQL> select /*+ parallel dynamic_sampling(0) */ s_nation "COUNTRY",
  2  sum(lo_supplycost) "SUPPLY_VALUE"
  3  from supplier, orderline_history
  4  where s_suppkey = lo_suppkey
  5  group by s_nation
  6  order by 2 desc;

COUNTRY         SUPPLY_VALUE
--------------- ------------
CHINA             2.4134E+10
INDONESIA         2.3917E+10
SAUDI ARABIA      2.3162E+10
JAPAN             2.2425E+10
INDIA             2.1977E+10
EGYPT             2.1626E+10
GERMANY           2.1480E+10
IRAQ              2.1216E+10
IRAN              2.1054E+10
BRAZIL            2.0941E+10
ALGERIA           2.0786E+10
MOZAMBIQUE        2.0641E+10
VIETNAM           2.0492E+10
UNITED KINGDOM    1.9772E+10
FRANCE            1.9598E+10
RUSSIA            1.9401E+10
UNITED STATES     1.9121E+10
PERU              1.8788E+10
KENYA             1.8455E+10
MOROCCO           1.8381E+10
CANADA            1.8243E+10
ARGENTINA         1.8210E+10
ETHIOPIA          1.8026E+10
JORDAN            1.6919E+10
ROMANIA           1.5123E+10

25 rows selected.

Elapsed: 00:00:02.03
SQL>
```

52. Examine the session statistics (use the script at `/stage/labs/6_1/statq.sql` if you prefer). Now that both tables are in the IMCS, you will see more in-memory activity; however, there is still a reasonable amount of non-in-memory activity. **Why?** It is because the small supplier table is only populated in the IMCS on one node, so the portion of the query executed on the other node could only leverage the IMCS for the `orderline_history` table, and was forced to perform normal reads (from the buffer cache or the physical disk) for the `supplier` table.

```
SQL> SELECT name, value FROM v$mystat m, v$statname n
  2  WHERE m.statistic# = n.statistic# AND value !=0
  3  AND (name like '%IM%'
  4  OR name in ('logical read bytes from cache','physical read bytes'));

NAME                                                                  VALUE
---------------------------------------------------------------- ----------
logical read bytes from cache                                      1957888
physical read bytes                                                 172032
table scans (IM)                                                        13
table scan disk non-IMC rows gotten                                   5642
IM scan CUs memcompress for query low                                   13
session logical reads - IM                                           83064
IM scan segments disk                                                    1
IM scan bytes in-memory                                          279377466
IM scan bytes uncompressed                                       559950362
IM scan CUs columns accessed                                            26
IM scan CUs columns theoretical max                                    211
IM scan rows                                                       5959478
IM scan rows valid                                                 5602000
IM scan rows range excluded                                         357478
IM scan rows projected                                             5602000
IM scan CUs split pieces                                                14

16 rows selected.

Elapsed: 00:00:00.00
SQL>
```

In this example, you saw how a large table and a small table cannot be joined completely in-memory because the small table is populated only in the IMCS in one instance, while the large table is distributed across multiple instances.

Note: This restriction does not exist on Oracle Engineered Systems, such as Exadata Database Machine, where data duplication is supported. Therefore, on Engineered Systems key, small tables (dimension tables) are typically duplicated on every IMCS, while large tables (fact tables) are typically distributed across all of the in-memory stores.

Part 7: Using Oracle Database In-Memory in conjunction with Oracle RAC to join large and small tables in memory by using partition-wise hash joins

In the final part of this practice, you will examine a strategy that enables large and small tables to be completely joined in-memory by using partition-wise hash joins.

53. Create hash partitioned versions of the `supplier` and `orderline_history` tables (use the script at `/stage/labs/6_1/hashcreate.sql` if you prefer). To ensure that the tables can be joined using a partition-wise hash join, make sure that:
    - The number of hash partitions is a multiple of the number of database instances in your Oracle RAC environment. In this example, create two hash partitions for each table.
    - The join column for each table is specified as the hash key.
    - The `inmemory` clause specifies `distribute by partition`.

```
SQL> create table hash_supplier
  2  partition by hash (s_suppkey) partitions 2
  3  inmemory distribute by partition
  4  as select * from supplier;

Table created.

Elapsed: 00:00:03.71
SQL> create table hash_orderline_history
  2  partition by hash (lo_suppkey) partitions 2
  3  inmemory distribute by partition
  4  as select * from orderline_history;

Table created.

Elapsed: 00:02:24.69
SQL>
```

54. Verify the in-memory attributes for the hash-partitioned tables (use the script at `/stage/labs/6_1/hashq.sql` if you prefer). Notice that the `INMEMORY_DISTRIBUTE` attribute is set to `BY PARTITION`.

```
SQL> select segment_name, partition_name, INMEMORY,
  2  INMEMORY_PRIORITY "IM_PRIORITY", INMEMORY_DISTRIBUTE "IM_DISTRIBUTE",
  3  INMEMORY_COMPRESSION "IM_COMPRESSION", INMEMORY_DUPLICATE "IM_DUPLICATE"
  4  from user_segments where segment_name like 'HASH%';

SEGMENT_NAME
--------------------------------------------------------------------------------
--
PARTITION_NAME
--------------------------------------------------------------------------------
--
INMEMORY IM_PRIOR IM_DISTRIBUTE   IM_COMPRESSION    IM_DUPLICATE
-------- -------- --------------- ----------------- -------------
HASH_ORDERLINE_HISTORY
SYS_P309
ENABLED  MEDIUM   BY PARTITION    FOR QUERY LOW     NO DUPLICATE

HASH_ORDERLINE_HISTORY
SYS_P310
ENABLED  MEDIUM   BY PARTITION    FOR QUERY LOW     NO DUPLICATE

HASH_SUPPLIER
```

```
SYS_P307
ENABLED  MEDIUM   BY PARTITION    FOR QUERY LOW     NO DUPLICATE

HASH_SUPPLIER
SYS_P308
ENABLED  MEDIUM   BY PARTITION    FOR QUERY LOW     NO DUPLICATE


Elapsed: 00:00:00.88
SQL>
```

55. Examine the `gv$im_segments` entries relating to your hash-partitioned tables. Verify the data distribution for each table.

```
SQL> select * from gv$im_segments where segment_name like 'HASH%';

   INST_ID
----------
OWNER
--------------------------------------------------------------------------------
SEGMENT_NAME
--------------------------------------------------------------------------------
PARTITION_NAME
--------------------------------------------------------------------------------
SEGMENT_TYPE       TABLESPACE_NAME                INMEMORY_SIZE      BYTES
------------------ ------------------------------ ------------- ----------
BYTES_NOT_POPULATED POPULATE_ INMEMORY INMEMORY_DISTRI INMEMORY_DUPL
------------------- --------- -------- --------------- -------------
INMEMORY_COMPRESS     CON_ID
----------------- ----------
         1
MEM
HASH_ORDERLINE_HISTORY
SYS_P309
TABLE PARTITION    EXAMPLE                              127336448  318767104
                  0 COMPLETED MEDIUM   BY PARTITION    NO DUPLICATE
FOR QUERY LOW              0


         1
MEM
HASH_SUPPLIER
SYS_P307
TABLE PARTITION    EXAMPLE                               1179648    8388608
                  0 COMPLETED MEDIUM   BY PARTITION    NO DUPLICATE
FOR QUERY LOW              0


         2
MEM
HASH_ORDERLINE_HISTORY
SYS_P310
TABLE PARTITION    EXAMPLE                              126287872  318767104
```

```
                  0 COMPLETED MEDIUM   BY PARTITION    NO DUPLICATE
FOR QUERY LOW              0


         2
MEM
HASH_SUPPLIER
SYS_P308
TABLE PARTITION    EXAMPLE                               1179648    8388608
                  0 COMPLETED MEDIUM   BY PARTITION    NO DUPLICATE
FOR QUERY LOW              0



Elapsed: 00:00:00.31
SQL>
```

56. Examine the execution plan for the import summary query by using the hash-partitioned tables (use the script at `/stage/labs/6_1/exphashsumq.sql` if you prefer). Compare this query plan with earlier version that did not use hash partitions. Notice the `PX PARTITION HASH ALL` operation, which indicates a parallel partition-wise hash join.

```
SQL> explain plan for
  2  select /*+ parallel dynamic_sampling(0) */ s_nation "COUNTRY",
  3  sum(lo_supplycost) "SUPPLY_VALUE"
  4  from hash_supplier, hash_orderline_history
  5  where s_suppkey = lo_suppkey
  6  group by s_nation
  7  order by 2 desc;

Explained.

Elapsed: 00:00:00.02
SQL> select * from table(dbms_xplan.display);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------------
Plan hash value: 2878539842

-------------------------------------------------------------------------------
| Id  | Operation                          | Name                  | Rows
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |                       |    25
|   1 |  PX COORDINATOR                    |                       |
|   2 |   PX SEND QC (ORDER)               | :TQ10002              |    25
|   3 |    SORT ORDER BY                   |                       |    25
|   4 |     PX RECEIVE                     |                       |    25
|   5 |      PX SEND RANGE                 | :TQ10001              |    25
|   6 |       HASH GROUP BY                |                       |    25
|   7 |        PX RECEIVE                  |                       |    25
|   8 |         PX SEND HASH               | :TQ10000              |    25
|   9 |          HASH GROUP BY             |                       |    25
|  10 |           PX PARTITION HASH ALL    |                       | 5600K
```

```
|* 11 |            HASH JOIN                   |                          | 5600K
|  12 |             TABLE ACCESS INMEMORY FULL| HASH_SUPPLIER           |  2000
|  13 |             TABLE ACCESS INMEMORY FULL| HASH_ORDERLINE_HISTORY | 5600K
-------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

  11 - access("S_SUPPKEY"="LO_SUPPKEY")

Note
-----
   - automatic DOP: Computed Degree of Parallelism is 2
   - parallel scans affinitized for inmemory

30 rows selected.

Elapsed: 00:00:00.06
SQL>
```

57. Reset the session statistics by reconnecting to the database, and execute the import summary query using the hash-partitioned tables (use the script at `/stage/labs/6_1/hashsummaryq.sql` if you prefer). Notice how quickly the query executes compared with previous executions.

```
SQL> connect mem/mem
Connected.
SQL> select /*+ parallel dynamic_sampling(0) */ s_nation "COUNTRY",
  2  sum(lo_supplycost) "SUPPLY_VALUE"
  3  from hash_supplier, hash_orderline_history
  4  where s_suppkey = lo_suppkey
  5  group by s_nation
  6  order by 2 desc;

COUNTRY         SUPPLY_VALUE
--------------- ------------
CHINA             2.4134E+10
INDONESIA         2.3917E+10
SAUDI ARABIA      2.3162E+10
JAPAN             2.2425E+10
INDIA             2.1977E+10
EGYPT             2.1626E+10
GERMANY           2.1480E+10
IRAQ              2.1216E+10
IRAN              2.1054E+10
BRAZIL            2.0941E+10
ALGERIA           2.0786E+10
MOZAMBIQUE        2.0641E+10
VIETNAM           2.0492E+10
UNITED KINGDOM    1.9772E+10
FRANCE            1.9598E+10
```

```
RUSSIA            1.9401E+10
UNITED STATES     1.9121E+10
PERU              1.8788E+10
KENYA             1.8455E+10
MOROCCO           1.8381E+10
CANADA            1.8243E+10
ARGENTINA         1.8210E+10
ETHIOPIA          1.8026E+10
JORDAN            1.6919E+10
ROMANIA           1.5123E+10

25 rows selected.

Elapsed: 00:00:01.78
SQL>
```

58. Examine the session statistics (use the script at `/stage/labs/6_1/statq.sql` if you prefer). This time you will see that there is little (or no) physical read activity. Also, notice that `IM scan rows`, `IM scan rows valid` and `IM scan rows projected` equal the sum of the rows in both tables; `hash_orderline_history` (5,600,000) and `hash_supplier` (2,000). This indicates that the query is making optimal use of the IMCS.

```
SQL> SELECT name, value FROM v$mystat m, v$statname n
  2  WHERE m.statistic# = n.statistic# AND value !=0
  3  AND (name like '%IM%'
  4  OR name in ('logical read bytes from cache','physical read bytes'));

NAME                                                                  VALUE
---------------------------------------------------------------- ----------
logical read bytes from cache                                     12001280
physical read bytes                                                  32768
table scans (IM)                                                         4
table scan disk non-IMC rows gotten                                   9157
IM scan CUs memcompress for query low                                   14
session logical reads - IM                                           76913
IM scan bytes in-memory                                          232372627
IM scan bytes uncompressed                                       526357037
IM scan CUs columns accessed                                            28
IM scan CUs columns theoretical max                                    218
IM scan rows                                                       5602000
IM scan rows valid                                                 5602000
IM scan rows projected                                             5602000
IM scan CUs split pieces                                                19

14 rows selected.

Elapsed: 00:00:00.04
SQL>
```

59. Exit all of your SQL*Plus sessions and terminal sessions.

Practices for Lesson 7: Application Continuity

Chapter 7

Practices for Lesson 7: Overview

Practices Overview

In this practice, you will explore Application Continuity.

Practice 7-1: Using Application Continuity

Overview

In this practice, you will use Application Continuity against a RAC database to demonstrate how Application Continuity helps an application to seamlessly recover after the failure of a RAC instance.

Tasks

1. Establish a terminal session connected to `c03n01` as the `oracle` user and configure the terminal environment as shown below.

```
$ ssh oracle@c03n01
[oracle@c03n01 ~]$ . oraenv
ORACLE_SID = [oracle] ? c03orcl
The Oracle base has been set to /u01/app/oracle
[oracle@c03n01 ~]$
```

2. Confirm that both instances of the RAC database are up and running.

```
[oracle@c03n01 ~]$ srvctl status database -d c03orcl
Instance c03orcl_1 is running on node c03n02
Instance c03orcl_2 is running on node c03n01
[oracle@c03n01 ~]$
```

3. Change into the directory that contains the files for this practice.

```
[oracle@c03n01 ~]$ cd /stage/labs/7_1
[oracle@c03n01 7_1]$
```

4. Examine the file `create_actest_service.sh`. This file contains a command to create a database service on the `c03orcl` database. This service is configured for use in conjunction with Application Continuity.

   Note: Services that support Application Continuity must specify `FAILOVER_TYPE=TRANSACTION` and `COMMIT_OUTCOME=TRUE`. The remaining service attribute settings used in this example are based on general recommendations for Application Continuity in conjunction with Oracle RAC; however, they may be adjusted to suit different circumstances.

```
[oracle@c03n01 7_1]$ cat create_actest_service.sh
#!/bin/bash
set -v

srvctl add service -db c03orcl -service actest \
       -serverpool ora.dev \
       -cardinality singleton \
       -clbgoal SHORT \
       -rlbgoal SERVICE_TIME \
       -failovertype TRANSACTION \
       -commit_outcome TRUE \
       -failoverretry 60 \
```

```
        -failoverdelay 5 \
        -retention 86400 \
        -replay_init_time 1800 \
        -notification TRUE
[oracle@c03n01 7_1]$
```

5. Execute `create_actest_service.sh` to create the database service.

```
[oracle@c03n01 7_1]$ ./create_actest_service.sh

srvctl add service -db c03orcl -service actest \
        -serverpool ora.dev \
        -cardinality singleton \
        -clbgoal SHORT \
        -rlbgoal SERVICE_TIME \
        -failovertype TRANSACTION \
        -commit_outcome TRUE \
        -failoverretry 60 \
        -failoverdelay 5 \
        -retention 86400 \
        -replay_init_time 1800 \
        -notification TRUE
[oracle@c03n01 7_1]$
```

6. Start the service that you created in the previous step.

```
[oracle@c03n01 7_1]$ srvctl start service -d c03orcl -s actest
[oracle@c03n01 7_1]$
```

7. Examine the status of the newly created service. Take note of the node it is running on (`c03n01` in the example below), because it may be different in your environment.

```
[oracle@c03n01 7_1]$ srvctl status service -d c03orcl
Service actest is running on nodes: c03n02
[oracle@c03n01 7_1]$
```

8. Establish another terminal session connected to `c03n01` as the `oracle` user and configure the terminal environment as shown below. To differentiate this session from your primary session, it will be referred to as the ADMIN session for the rest of the practice.

```
$ ssh oracle@c03n01
[oracle@c03n01 ~]$ . oraenv
ORACLE_SID = [oracle] ? c03orcl
The Oracle base has been set to /u01/app/oracle
[oracle@c03n01 ~]$ export PS1='ADMIN $ '
ADMIN $
```

9. Back in your primary session, examine the scripts that you will soon use to execute the practice application. Notice that both scripts execute the same application code (in `actest.jar`). The only difference is that each script references a different properties file.

```
[oracle@c03n01 7_1]$ cat runnoreplay.sh
java -classpath
./actest.jar:$ORACLE_HOME/ucp/lib/ucp.jar:$ORACLE_HOME/jdbc/lib/
ojdbc6.jar actest.ACTest actest_noreplay.properties
[oracle@c03n01 7_1]$ cat runreplay.sh
java -classpath
./actest.jar:$ORACLE_HOME/ucp/lib/ucp.jar:$ORACLE_HOME/jdbc/lib/
ojdbc6.jar actest.ACTest actest_replay.properties
[oracle@c03n01 7_1]$
```

10. Examine the properties files. Notice that the only difference is the `datasource` specification.

```
[oracle@c03n01 7_1]$ cat actest_noreplay.properties
username=scott
password=tiger
autoCommit=false

# Use standard  12.1 datasource no replay
datasource=oracle.jdbc.pool.OracleDataSource

url=jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=
cluster03-
scan.example.com)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=actest.
example.com)))

# UCP setting:
ucp_pool_size=2
ucp_validate_connection_on_borrow=true
ucp_connection_wait_timeout=60

# Think Time taken to process the results from the database.
Time in milliseconds.
# -1 means no sleep.
thread_think_time=20

# Number of concurrent threads running in the application
# UCP is tuned to have MAX and MIN limit set to this
number_of_threads=6

verbose=true
[oracle@c03n01 7_1]$ cat actest_replay.properties
username=scott
```

```
password=tiger
autoCommit=false

# Use new 12.1 replay datasource
datasource=oracle.jdbc.replay.OracleDataSourceImpl

url=jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=
cluster03-
scan.example.com)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=actest.
example.com)))

# UCP setting:
ucp_pool_size=2
ucp_validate_connection_on_borrow=true
ucp_connection_wait_timeout=60

# Think Time taken to process the results from the database.
Time in milliseconds.
# -1 means no sleep.
thread_think_time=20

# Number of concurrent threads running in the application
# UCP is tuned to have MAX and MIN limit set to this
number_of_threads=6

verbose=true
[oracle@c03n01 7_1]$ diff actest_noreplay.properties
actest_replay.properties
5,6c5,6
< # Use standard  12.1 datasource no replay
< datasource=oracle.jdbc.pool.OracleDataSource
---
> # Use new 12.1 replay datasource
> datasource=oracle.jdbc.replay.OracleDataSourceImpl
[oracle@c03n01 7_1]$
```

Next, you will execute the practice Java application twice – once without the benefit of Application Continuity, and once with Application Continuity enabled. Notice that you will execute the same application and the only difference is the JDBC data source that is used on each occasion. The source files containing the application code are contained in the `src` directory. Feel free to examine the application code if you like.

11. Execute the practice application without the benefit of Application Continuity. Notice that while the application runs, a periodic status message is displayed.

```
[oracle@c03n01 7_1]$ ./runnoreplay.sh
#########################################################
Connecting to
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=clus
ter03-
scan.example.com)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=actest.
example.com)))
  # of Threads             : 6
  UCP pool size            : 2
  Thread think time        : 20 ms
#########################################################

2 active connections, avg response time from db 21 ms
2 active connections, avg response time from db 11 ms
...
```

12. While the application continues to execute in the primary window, return to your ADMIN session and remind yourself about which node is running the `actest` service. Then, crash the database instance running the `actest` service by killing the `smon` process. Ensure that you crash the instance on the node running the service (`c03n02` in the example shown below) and not the other database node.

```
ADMIN $ srvctl status service -d c03orcl
Service actest is running on nodes: c03n02
ADMIN $ ssh root@c03n02
root@c03n02's password: <oracle>
[root@c03n02 ~]# ps -ef|grep smon_c03orcl
oracle   13467     1  0 04:24 ?        00:00:00 ora_smon_c03orcl_1
root     27362 27188  0 04:48 pts/0    00:00:00 grep smon_c03orcl
[root@c03n02 ~]# kill -9 13467
[root@c03n02 ~]# exit
logout
Connection to c03n02 closed.
ADMIN $
```

13. Return to your primary window and you will see a series of errors caused by crashing the database instance. This is typical of applications that do not use Application Continuity. Press Ctrl + C to abort the application.

```
...
.Exception occurred while getting connection:
oracle.ucp.UniversalConnectionPoolException: Cannot get
Connection from Datasource: java.sql.SQLRecoverableException:
Listener refused the connection with the following error:
ORA-12514, TNS:listener does not currently know of service
requested in connect descriptor
 .
[oracle@c03n01 7_1]$
```

14. In your practice environment, Oracle High Availability Services will restart the crashed database instance. Periodically re-examine the database status until you can confirm that both RAC database instances are up and running again.

```
[oracle@c03n01 7_1]$ srvctl status database -d c03orcl
Instance c03orcl_1 is running on node c03n02
Instance c03orcl_2 is running on node c03n01
[oracle@c03n01 7_1]$
```

15. Examine the status of the `actest` service. Notice that the service is running on a different node compared to what you observed earlier. This is because the service was automatically migrated when you crashed the database instance earlier in the practice.

```
[oracle@c03n01 7_1]$ srvctl status service -d c03orcl
Service actest is running on nodes: c03n01
[oracle@c03n01 7_1]$
```

16. Execute the practice application with Application Continuity enabled. You should see the same status messages as before while the application is running.

```
[oracle@c03n01 7_1]$ ./runreplay.sh
#########################################################
Connecting to
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=clus
ter03-
scan.example.com)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=actest.
example.com)))
  # of Threads             : 6
  UCP pool size            : 2
  Thread think time        : 20 ms
#########################################################

2 active connections, avg response time from db 304 ms
2 active connections, avg response time from db 45 ms
...
```

17. While the application continues to execute in the primary window, return to your ADMIN session and remind yourself about which node is now running the `actest` service. Then, crash the database instance running the `actest` service. Ensure that you crash the instance on the node running the service (which is now `c03n01` in the example shown below) and not the other database node.

```
ADMIN $ srvctl status service -d c03orcl
Service actest is running on nodes: c03n01
ADMIN $ ssh root@c03n01
root@c03n01's password: <oracle>
[root@c03n01 ~]# ps -ef|grep smon_c03orcl
root      9104  9039  0 05:00 pts/3    00:00:00 grep smon_c03orcl
oracle   32401     1  0 04:36 ?        00:00:00 ora_smon_c03orcl_2
[root@c03n01 ~]# kill -9 32401
[root@c03n01 ~]# exit
```

```
logout
Connection to c03n01 closed.
ADMIN $
```

18. As before, the service will automatically migrate in response to the instance crash. Confirm that the `actest` service is running on a different node compared to what you observed in the previous step. You may need to wait for a while for the service migration to complete, so periodically re-examine the service status if required.

```
ADMIN $ srvctl status service -d c03orcl
Service actest is running on nodes: c03n02
ADMIN $
```

19. Return to the primary window and you will see that the application continues to run in spite of crashing the database instance. You should see a brief spike in the response time, which coincides with the time when the database instance crashed. Now you have seen how Application Continuity masks the effect of database instance loss in a RAC database environment. Press Ctrl + C to abort the application.

```
[oracle@c03n01 7_1]$ ./runreplay.sh
#####################################################
Connecting to
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=clus
ter03-
scan.example.com)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=actest.
example.com)))
  # of Threads             : 6
  UCP pool size            : 2
  Thread think time        : 20 ms
#####################################################

2 active connections, avg response time from db 304 ms
2 active connections, avg response time from db 45 ms
2 active connections, avg response time from db 29 ms
2 active connections, avg response time from db 32 ms
2 active connections, avg response time from db 29 ms
2 active connections, avg response time from db 18 ms
2 active connections, avg response time from db 36 ms
2 active connections, avg response time from db 5962 ms
2 active connections, avg response time from db 98 ms
2 active connections, avg response time from db 61 ms
2 active connections, avg response time from db 72 ms
^C[oracle@c03n01 7_1]$
```

20. Exit all of your terminal sessions.

    Congratulations! You have now seen how to configure and use Application Continuity.

Practice 7-2: Environment Reconfiguration

Overview

In this practice, you will reconfigure your practice environment in preparation for the next practice.

Tasks

1. Establish a terminal session on your practice environment. Do not connect to any of the virtual machines.
2. Execute the following command to perform a series of reconfiguration steps in preparation for the next set of practices. Note that the reconfiguration process may run for more than 30 minutes. Ensure that you allow the reconfiguration process to complete and check to make sure that the VM status output includes entries for `enode01`, `enode02`, `wnode03`, and `wnode04`.

```
$ sudo /OVS/seed_pool/setup_GDS.sh
Removing current VMs...
Extracting GDS VMs...
Starting VMs...
VM status...
Name                                ID    Mem VCPUs      State
Time(s)
Domain-0                             0   1024     2     r-----  12228.3
enode01                              9   3800     1     -b----     20.1
enode02                             10   3400     1     -b----      6.3
wnode03                             11   3400     1     -b----      5.1
wnode04                             12   3400     1     -b----      4.5
$
```

Practices for Lesson 8: Oracle Global Data Services

Chapter 8

Practices for Global Data Services

Overview

In this practice, you will:

- Install Global Service Managers
- Deploy GDS framework for your data cloud
- Configure global services for the following GDS features
    - Inter-database global service failover
    - Role-based global services
    - Lag-based routing
- Obtain familiarity with the GDSCTL interface

Practice 8-1: Installing and Configuring Global Data Services

Overview

This practice will demonstrate the creation of GDS configuration that offers the global services failover/load balancing capabilities. You will create a GDS configuration that contains the following components:

- GDS catalog hosted in GDSCAT
- Two GDS Regions: EAST region and WEST region
- Two Global Services Managers (GSMs): one GSM per region (GSMEAST and GSMWEST)
- One GDS pool called SALES
- Two GDS pool RAC databases in an Active Data Guard configuration, with one database in each of the datacenters. The EASTDB database is in the EAST region and the WESTDB database is in the WEST region.

In this practice, The GSMEAST Global Service Manager, GDSCAT and EASTDB Database are hosted on CLUSTER01. The GSMWEST Global Service Manager and WESTDB Database are hosted on CLUSTER02.

1. First, you will install and deploy GDS in your environment. Establish an `ssh` connection using the `-X` option as the `oracle` user to host `enode01`.

```
$ ssh -X oracle@enode01
oracle@'s password:
[oracle@enode01 ~]$
```

2. Change directory to `/stage/GSM/` and start the installer.

```
[oracle@enode01 ~]$ cd /stage/12.1/gsm

[oracle@enode01 Disk1]$ ./runInstaller
Starting Oracle Universal Installer...

Checking Temp space: must be greater than 500 MB.   Actual 7934
MB    Passed
Checking swap space: must be greater than 150 MB.   Actual 8632
MB    Passed
Checking monitor: must be configured to display at least 256
colors.    Actual 16777216    Passed
Preparing to launch Oracle Universal Installer from
/tmp/OraInstall2013-05-22_03-35-08PM. Please wait ...
```

3. On the Specify Installation Location screen, specify `/u01/app/oracle` as the Oracle base value and `/u01/app/oracle/product/12.1.0/gsmhome_1` as the Software location. Click Next.
4. On the Summary page, click Install.
5. When the Execute Configuration scripts dialog box appears, open a terminal window to `enode01` as the `root` user and execute the `root.sh` script.

```
[root@enode01 ~]# /u01/app/oracle/product/12.1.0/gsmhome_1/root.sh

Performing root user operation for Oracle 12c
```

```
The following environment variables are set as:
    ORACLE_OWNER= oracle
    ORACLE_HOME=  /u01/app/oracle/product/12.1.0/gsmhome_1

Enter the full pathname of the local bin directory:
[/usr/local/bin]:
The contents of "dbhome" have not changed. No need to overwrite.
The contents of "oraenv" have not changed. No need to overwrite.
The contents of "coraenv" have not changed. No need to overwrite.

Entries will be added to the /etc/oratab file as needed by
Database Configuration Assistant when a database is created
Finished running generic part of root script.
Now product-specific root actions will be performed.
You have new mail in /var/spool/mail/root

[root@enode01 ~]# exit
```

6. When the script has finished running, dismiss the dialog box by clicking OK.
7. Repeat steps 1-6 on host `wnode03` to install GDS.

GDS Setup and Configuration

8. On `enode01`, there is a precreated single instance database called GDSCAT to host the GDS catalog. Add the following TNS entries for the GDS catalog and pool databases to the GSM Home's `tnsnames.ora` file located in `/u01/app/oracle/product/12.1.0/gsmhome_1/network/admin`. If the file does not exist, create it. Set the environment and use `tnsping` to test the new connect string. Perform this task on the GSM hosts, `enode01` and `wnode03`.

```
[oracle@enode01 ~]$ vi
/u01/app/oracle/product/12.1.0/gsmhome_1/network/admin/tnsnames.o
ra

**** Add entries shown below ****

GDSCAT =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = enode01.example.com)(PORT
= 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = gdscat.example.com)
  )
)

EASTDB,EASTDB.EXAMPLE.COM =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = cluster01-scan)(PORT =
1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
```

```
      (SERVICE_NAME = eastdb.example.com)
   )
)

WESTDB,WESTDB.EXAMPLE.COM =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = cluster02-scan)(PORT =
1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = westdb.example.com)
    )
)

[oracle@enode01 admin]$ export
ORACLE_HOME=/u01/app/oracle/product/12.1.0/gsmhome_1

[oracle@enode01 admin]$ export ORACLE_BASE=/u01/app/oracle

[oracle@enode01 admin]$ export PATH=$ORACLE_HOME/bin:$PATH

[oracle@enode01 admin]$ tnsping gdscat

TNS Ping Utility for Linux: Version 12.1.0.2.0 - Production on
22-AUG-2014 16:05:26

Copyright (c) 1997, 2014, Oracle.  All rights reserved.

Used parameter files:


Used TNSNAMES adapter to resolve the alias
Attempting to contact (DESCRIPTION = (ADDRESS = (PROTOCOL =
TCP)(HOST = enode01.example.com)(PORT = 1521)) (CONNECT_DATA =
(SERVER = DEDICATED) (SERVICE_NAME = gdscat.example.com)))
OK (90 msec)

[oracle@enode01 ~]$
```

Setup GDS Administrator Accounts & Privileges

9. Make sure that the listeners and databases WESTDB, EASTDB, and GDSCAT are up. Connect to the pool database EASTDB (current Primary) and unlock the GSM user.

```
[oracle@enode01 ~]$ . oraenv
ORACLE_SID = [oracle] ? eastdb
The Oracle base has been set to /u01/app/oracle

[oracle@enode01 ~]$ sqlplus system/oracle_4U@eastdb

SQL> show parameter db_unique
NAME                                 TYPE        VALUE
------------------------------------ ----------- ----------------
```

```
db_unique_name                         string      eastdb
SQL>

SQL> alter user gsmuser account unlock;
User altered.

SQL> alter user gsmuser identified by oracle_4U;
SQL>
```

10. In this practice, you are hosting the GDS catalog in the `GDSCATdatabase`. So, connect to the GDSCAT and unlock `gsmcatuser` and grant `GSMADMIN_ROLE` to an existing user or a newly created user.

```
SQL> connect system/oracle_4U@gdscat
Connected.

SQL> show parameter db_unique
NAME                                 TYPE        VALUE
------------------------------------ ----------- ----------------
db_unique_name                       string      gdscat


SQL> alter user gsmcatuser account unlock;
User altered.

SQL> alter user gsmcatuser identified by oracle_4U;
User altered.


SQL> create user mygdsadmin identified by oracle_4U;
User created.

SQL> grant gsmadmin_role to mygdsadmin;
Grant succeeded.

SQL> exit

[oracle@enode01 ~]$
```

Configure GDS

11. From the terminal window of the EAST GSM node, `enode01`, set the following environment variables. Using the GDSCTL interface, the user with `GSMADMIN_ROLE` creates the GDS catalog.

```
[oracle@enode01 ~]$ export
ORACLE_HOME=/u01/app/oracle/product/12.1.0/gsmhome_1

[oracle@enode01 ~]$ export ORACLE_BASE=/u01/app/oracle

[oracle@enode01 ~]$ export PATH=$ORACLE_HOME/bin:$PATH

[oracle@enode01 ~]$ gdsctl
GDSCTL: Version 12.1.0.2.0
```

```
Copyright (c) 2011, 2012, Oracle. All rights reserved.
Welcome to GDSCTL, type "help" for information.
Current GSM is set to GSMORA

GDSCTL> create catalog -database enode01:1521:gdscat –user
mygdsadmin/oracle_4U

GDSCTL>
```

12. Add the East Global Service Manager (GSM).

```
GDSCTL> add gsm -gsm gsmeast -listener 1571 -catalog
enode01:1521:gdscat
"gsmcatuser" password:*****************
Create credential oracle.security.client.connect_string1
GSM successfully added

GDSCTL>
```

As the `-pwd` parameter has not been provided, GDSCTL asks for the `gsmcatuser` password. Port 1571 is specified for the GSM listener of the GSMEAST.

13. Start the East Global Service Manager (`gsmeast`) on `enode01`.

```
GDSCTL> start gsm –gsm gsmeast
GSM is started successfully

GDSCTL> config gsm -gsm gsmeast
Name: gsmeast
Endpoint 1:
(ADDRESS=(HOST=enode01.example.com)(PORT=1571)(PROTOCOL=tcp))
Local ONS port: 6123
Remote ONS port: 6234
ORACLE_HOME path: /u01/app/oracle/product/12.1.0/gsmhome_1
GSM Host name: enode01.example.com
Region: regionora
Buddy
------------------------

GDSCTL> status gsm -gsm gsmeast
Alias GSMEAST
Version 12.1.0.2.0
Start Date 21-MAR-2014 20:23:33
Trace Level off
Listener Log File
/u01/app/oracle/diag/gsm/enode01/gsmeast/alert/log.xml
Listener Trace File
/u01/app/oracle/diag/gsm/enode01/gsmeast/trace/ora_11548_46969260
647648.trc
Endpoint summary
(ADDRESS=(HOST=enode01.example.com)(PORT=1571)(PROTOCOL=tcp))
GSMOCI Version 0.1.11
Mastership N
Connected to GDS catalog Y
```

```
Process Id 11550
Number of reconnections 0
Pending tasks. Total 0
Tasks in process. Total 0
Regional Mastership FALSE
Total messages published 0
Time Zone -07:00
Orphaned Buddy Regions:
None
GDS region regionora

GDSCTL> exit

oracle@enode01 ~]$
```

14. Open a terminal window to `wnode03` as the `oracle` user, set the GSM environment, and create the West Global Service Manager (GSM) on `wnode03`.

```
$ ssh oracle@wnode03
oracle@'s password:
[oracle@wnode03 ~]$ export
ORACLE_HOME=/u01/app/oracle/product/12.1.0/gsmhome_1

[oracle@wnode03 ~]$ export ORACLE_BASE=/u01/app/oracle

[oracle@wnode03 ~]$ export PATH=$ORACLE_HOME/bin:$PATH

[oracle@wnode03 ~]$ gdsctl

GDSCTL: Version 12.1.0.2.0 - Production on Fri Aug 22 18:12:45
UTC 2014
Copyright (c) 2011, 2014, Oracle.  All rights reserved.
Welcome to GDSCTL, type "help" for information.
Current GSM is set to GSMORA

GDSCTL> add gsm -gsm gsmwest -listener 1572 -catalog
enode01:1521:gdscat
"gsmcatuser" password:*****************
Create credential oracle.security.client.connect_string1
GSM successfully added

GDSCTL>
```

As the `-pwd` parameter has not been provided, GDSCTL asks for the `gsmcatuser` password. Port 1572 is specified for the GSM listener of the GSMWEST.

15. From the GDSCTL prompt on `wnode03`, start the newly created GSM. Check the GSM configuration and status after starting GSMWEST.

```
On wnode03:

GDSCTL> start gsm -gsm gsmwest
GSM is started successfully

GDSCTL> config gsm -gsm gsmwest
```

```
Name: gsmwest
Endpoint 1:
(ADDRESS=(HOST=wnode03.example.com)(PORT=1572)(PROTOCOL=tcp))
Local ONS port: 6123
Remote ONS port: 6234
ORACLE_HOME path: /u01/app/oracle/product/12.1.0/gsmhome_1
GSM Host name: wnode03.example.com
Region: regionora
Buddy
------------------------

GDSCTL> status gsm -gsm gsmwest
Alias GSMWEST
Version 12.1.0.1.0
Start Date 21-MAR-2014 20:18:13
Trace Level off
Listener Log File
/u01/app/oracle/diag/gsm/wnode03/gsmwest/alert/log.xml
Listener Trace File
/u01/app/oracle/diag/gsm/wnode03/gsmwest/trace/ora_7493_473457111
17536.trc
Endpoint summary (ADDRESS=(HOST=
wnode03.example.com)(PORT=1572)(PROTOCOL=tcp))
GSMOCI Version 0.1.10
Mastership N
Connected to GDS catalog Y
Process Id 7495
Number of reconnections 0
Pending tasks. Total 0
Tasks in process. Total 0
Regional Mastership TRUE
Total messages published 21282
Time Zone -07:00
Orphaned Buddy Regions:
None
GDS region regionora

GDSCTL> exit

[oracle@wnode03 ~]$
```

Create the GDS regions

16. From the `enode01` terminal window, create two regions, EAST and WEST.

```
[oracle@enode01 ~]$ gdsctl
GDSCTL: Version 12.1.0.2.0

Copyright (c) 2011, 2012, Oracle. All rights reserved.
Welcome to GDSCTL, type "help" for information.
Current GSM is set to GSMEAST

GDSCTL> add region -region west, east

GDSCTL>
```

17. From the GDSCTL prompt on enode01, assign the east region to the east GSM.

```
On enode01:

GDSCTL> modify gsm -gsm gsmeast -region east
GSM modified

GDSCTL> exit

[oracle@enode01 ~]$
```

18. From the West GSM terminal (wnode03), assign the west region to the west GSM.

```
[oracle@wnode03 ~]$ gdsctl

GDSCTL> connect
mygdsadmin/oracle_4U@enode01:1521/gdscat.example.com
Catalog connection is established

GDSCTL> modify gsm -gsm gsmwest -region west
GSM modified
GDSCTL>exit

[oracle@wnode03 ~]$
```

Create the GDS pools

19. As we need just one gdspool, we can use either the predefined one dbpoolora or create a new one called SALES.

```
[oracle@enode01 ~]$ gdsctl

GDSCTL: Version 12.1.0.2.0 - Production on Fri Aug 22 18:12:45
UTC 2014
Copyright (c) 2011, 2014, Oracle.  All rights reserved.
Welcome to GDSCTL, type "help" for information.
Current GSM is set to GSMEAST

GDSCTL> add gdspool -gdspool sales

GDSCTL> exit

[oracle@enode01 ~]$
```

Add Data Guard broker configuration to the GDS pool

20. From the enode01 terminal, add the Data Guard broker configuration to the SALES gdspool. Using DGMGRL, first check how the Data Guard configuration looks like from one of the database nodes. Use DGMGRL to check the databases to confirm that redo transport and apply are active.

```
[oracle@enode01 ~]$ . oraenv
ORACLE_SID = [orcl] ? eastdb
The Oracle base remains unchanged with value /u01/app/oracle
```

```
[oracle@enode01 ~]$ export ORACLE_SID=eastdb1

[oracle@enode01 ~]$ dgmgrl
DGMGRL for Linux: Version 12.1.0.2.0 - 64bit Production
Copyright (c) 2000, 2013, Oracle. All rights reserved.
Welcome to DGMGRL, type "help" for information.

DGMGRL> connect sysdg/oracle_4U
Connected as SYSDG.

DGMGRL> show configuration
Configuration - dg_config
Protection Mode: MaxPerformance
Databases:
eastdb - Primary database
 westdb - Physical standby database
Fast-Start Failover: DISABLED
Configuration Status:
SUCCESS

DGMGRL> show database eastdb

Database - eastdb

  Role:               PRIMARY
  Intended State:     TRANSPORT-ON
  Instance(s):
    eastdb1
    eastdb2

Database Status:
SUCCESS

DGMGRL> show database westdb

Database - westdb

  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: 5.00 KByte/s
  Real Time Query:    ON
  Instance(s):
    westdb1 (apply instance)
    westdb2

Database Status:
SUCCESS

DGMGRL> exit

[oracle@enode01 ~]$
```

Observe that in this configuration, EASTDB is the current Primary and WESTDB is the current Standby.

21. From the `enode01` terminal, set your environment for GSM and add the broker configuration by using GDSCTL. Always add the broker configuration connecting to the current PRIMARY database. In this case, EASTDB is the PRIMARY database.

```
[oracle@enode01 ~]$ export
ORACLE_HOME=/u01/app/oracle/product/12.1.0/gsmhome_1

[oracle@enode01 ~]$ export ORACLE_BASE=/u01/app/oracle

[oracle@enode01 ~]$ export PATH=$ORACLE_HOME/bin:$PATH

[oracle@enode01 ~]$ gdsctl
GDSCTL: Version 12.1.0.2.0 - Production on Fri Aug 22 18:39:20
UTC 2014

Copyright (c) 2011, 2014, Oracle.  All rights reserved.

Welcome to GDSCTL, type "help" for information.

Current GSM is set to GSMEAST

GDSCTL> add brokerconfig -connect enode01:1521:eastdb1 -region
east -gdspool sales
"gsmuser" password:**************
DB Unique Name: eastdb

GDSCTL>
```

22. From the GDSCTL prompt on `enode01`, assign the database WESTDB to the WEST region.

```
On enode01:

GDSCTL> modify database -database westdb -region west -gdspool
sales -connect wnode03:1521/westdb.example.com
```

23. From the GDSCTL prompt on `enode01`, check that the databases are registered to the GSM.

```
On enode01:

GDSCTL> databases
Database: "eastdb" Registered: Y State: Ok ONS: Y. Role: PRIMARY
Instances: 2 Region: east
   Registered instances:
     sales%1
     sales%2
Database: "westdb" Registered: Y State: Ok ONS: Y. Role:
PH_STNDBY Instances: 2 Region: west
   Registered instances:
     sales%11
     sales%12

GDSCTL>
```

Create and Start Global Service in a Pool

24. From the GDSCTL prompt on `enode01`, create a service called `SALES_REPORTING_SRVC`, which should always run on the standby database. In the event the standby database is not available, the service can run on the primary database. When the service has been created, use the `modify service` command to add instances to the respective databases (since they are administrator-managed). Start the new service and check the status.

```
On enode01:

GDSCTL> add service -service sales_reporting_srvc -gdspool sales
-preferred_all -role PHYSICAL_STANDBY -failover_primary

GDSCTL> modify service -gdspool sales -service
sales_reporting_srvc -database westdb -add_instances -preferred
westdb1,westdb2

GDSCTL> modify service -gdspool sales -service
sales_reporting_srvc -database eastdb -add_instances -preferred
eastdb1,eastdb2

GDSCTL> start service -service sales_reporting_srvc -gdspool
sales

GDSCTL> status service -service sales_reporting_srvc
Service "sales_reporting_srvc.sales.oradbcloud" has 2
instance(s). Affinity: ANYWHERE
   Instance "sales%1", name: "westdb1", db: "westdb", region:
"west", status: ready.
   Instance "sales%2", name: "westdb2", db: "westdb", region:
"west", status: ready.

GDSCTL>
```

25. View the configuration info of the newly created service.

```
On enode01:

GDSCTL> config service -service sales_reporting_srvc
Name: sales_reporting_srvc
Network name: sales_reporting_srvc.sales.oradbcloud
Pool: sales
Started: Yes
Preferred all: Yes
Locality: ANYWHERE
Region Failover: No
Role: PHYSICAL_STANDBY
Primary Failover: Yes
Lag: ANY
Runtime Balance: SERVICE_TIME
Connection Balance: LONG
Notification: Yes
```

```
TAF Policy: NONE
Policy: AUTOMATIC
DTP: No
Failover Method: NONE
Failover Type: NONE
Failover Retries:
Failover Delay:
Edition:
PDB:
Commit Outcome:
Retention Timeout:
Replay Initiation Timeout:
Session State Consistency:
SQL Translation Profile:
Databases
------------------------
Database Preferred Status
-------- --------- ------
westdb Yes Enabled
eastdb Yes Enabled
```

26. Display the configuration information of all the components that are part of your GDS configuration.

```
On enode01:

GDSCTL> config
Regions
------------------------
Name Buddy
---- -----
regionora <- Default GDS Region
west
east
GSMs
------------------------
gsmeast
gsmwest
GDS pools
------------------------
dbpoolora <- Default GDS Pool
sales
Databases
------------------------
eastdb
westdb
Services
------------------------
sales_reporting_srvc
GDSCTL pending requests
------------------------
Command Object Status
------- ------ ------
Global properties
```

```
------------------------
Name: oradbcloud
Master GSM: gsmeast

GDSCTL> exit
[oracle@enode01 ~]$
```

27. Now that the global service has been created and started, you can query DBA_SERVICES and V$ACTIVE_SERVICES to learn more about the global services. So, connect to the standby database where the global service is currently running:

```
[oracle@wnode03 ~]$ . oraenv
ORACLE_SID = [oracle] ? westdb
The Oracle base remains unchanged with value /u01/app/oracle

[oracle@wnode03 ~]$ sqlplus sys/oracle_4U@westdb as sysdba

SQL*Plus: Release 12.1.0.2.0 Production on Tue Sep 2 23:18:51
2014

Copyright (c) 1982, 2014, Oracle.  All rights reserved.


Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit
Production
With the Partitioning, Real Application Clusters, Automatic
Storage Management, OLAP,
Advanced Analytics and Real Application Testing options

SQL> column name format a30
SQL> column network_name format a40
SQL> column global format a10
SQL> set linesize 120
SQL> select name, network_name, global_service from dba_services;

NAME                           NETWORK_NAME                                 GLO
------------------------------ -------------------------------------------- ---
SYS$BACKGROUND                                                              NO
SYS$USERS                                                                   NO
eastdb_DGB                     eastdb_DGB                                   NO
eastdbXDB                      eastdbXDB                                    NO
eastdb.example.com             eastdb.example.com                           NO
sales_reporting_srvc           sales_reporting_srvc.sales.oradbcloud        YES
6 rows selected.
```

Note: For the sales_reporting_srvc, the value of the column GLOBAL_SERVICE is "Yes", denoting that it is a global service.

```
SQL> select name, network_name, global from v$active_services;

NAME                       NETWORK_NAME                                 GLOBAL
-------------------------- -------------------------------------------- -------
sales_reporting_srvc       sales_reporting_srvc.sales.oradbcloud        YES
SYS$BACKGROUND                                                          NO
```

```
SYS$USERS                                                          NO
3 rows selected.
```

Note: The query above shows that currently the sales_reporting_srvc global service is listed in the v$active_services because it has been started.

```
SQL> exit

[oracle@wnode03 ~]$
```

Add TNS entries for Clients

28. Add the TNS entries (based on GSM listeners) to the client's `tnsnames.ora`. Here is an example TNS entry for the "`sales_reporting_srvc`" application clients, where `SERVICE_NAME` is the name of the global service and the `REGION` is the region that the client is coming from. This entry should appear in the `tnsnames.ora` file on any host from which a client connection to the service is required. But in this example, we'll update the tnsnames.ora on `enode01` only for the sake of simplicity and originate our connections from there.

    Note: The connect descriptor in the `tnsnames.ora` in a GDS configuration will be using the GSM listener end points and not the RAC SCAN listeners or local listeners.

```
[oracle@enode01 ~]$ vi
/u01/app/oracle/product/12.1.0/dbhome_1/network/admin/tnsnames.ora

**** Add entry below ***

sales_reporting_srvc =
   (DESCRIPTION =
    (ADDRESS_LIST =
       (ADDRESS = (PROTOCOL = TCP)(HOST = enode01)(PORT = 1571)))
    (ADDRESS_LIST =
       (ADDRESS = (PROTOCOL = TCP)(HOST = wnode03)(PORT = 1572)))
     (CONNECT_DATA =
     (SERVICE_NAME = sales_reporting_srvc.sales.oradbcloud)
(REGION=WEST)
    )
   )
```

    Note: 1572 is the GSM listener port on `wnode03` and 1571 is the GSM listener port on `enode01`

29. Clients can connect to the pool databases either via the Easy Connect Naming or with the TNSNAMES. From a terminal window on `enode01`, set the environment and connect to the service as shown below.

```
[oracle@enode01 ~]$ . oraenv
ORACLE_SID = [oracle] ? eastdb
The Oracle base has been set to /u01/app/oracle
[oracle@enode01 ~]$
```

Via Easy Connect Naming Method:

```
[oracle@enode01 ~]$ sqlplus
system/oracle_4U@//enode01:1571/sales_reporting_srvc.sales.oradbc
loud

SQL> select instance_name from v$instance;

INSTANCE_NAME
----------------
westdb1


Via TNSNAMES Connect Descriptor:
SQL> connect system/oracle_4U@sales_reporting_srvc

SQL> select instance_name from v$instance;

INSTANCE_NAME
----------------
westdb2

SQL> exit

[oracle@enode01 ~]$
```

Practice 8-2: Global Service Failover

Overview

To see how the failover of global services happens, you will shut down one of the databases and observe that the services are automatically started in the other pool database.

1. Using GDSCTL, observe that the `sales_reporting_srvc` global service is currently running on the standby WESTDB per the service attributes that we have defined.

```
[oracle@enode01 ~]$ export
ORACLE_HOME=/u01/app/oracle/product/12.1.0/gsmhome_1

[oracle@enode01 ~]$ export ORACLE_BASE=/u01/app/oracle

[oracle@enode01 ~]$ export PATH=$ORACLE_HOME/bin:$PATH

[oracle@enode01 ~]$ gdsctl
GDSCTL: Version 12.1.0.2.0 - Production on Fri Aug 22 18:39:20
UTC 2014

Copyright (c) 2011, 2014, Oracle.  All rights reserved.

Welcome to GDSCTL, type "help" for information.

Current GSM is set to GSMEAST

GDSCTL> services
Service "sales_reporting_srvc.sales.oradbcloud" has 2
instance(s). Affinity: ANYWHERE
Instance "sales%11", name: "westdb1", db: "westdb", region:
"west", status: ready.
Instance "sales%12", name: "westdb2", db: "westdb", region:
"west", status: ready.

GDSCTL> databases
Database: "eastdb" Registered: Y State: Ok ONS: Y. Role: PRIMARY
Instances: 2 Region: east
  Service: "sales_reporting_srvc" Globally started: Y Started: N
           Scan: N Enabled: Y Preferred: Y
   Registered instances:
     sales%1
     sales%2
Database: "westdb" Registered: Y State: Ok ONS: Y. Role:
PH_STNDBY Instances: 2 Region: west
  Service: "sales_reporting_srvc" Globally started: Y Started: Y
```

```
            Scan: N Enabled: Y Preferred: Y
   Registered instances:
     sales%11
     sales%12
GDSCTL>
```

2. From the wnode03 terminal, stop the pool database WESTDB using SRVCTL.

```
[oracle@wnode03 ~]$ . oraenv
ORACLE_SID = [oracle] ? westdb
The Oracle base has been set to /u01/app/oracle

[oracle@wnode03 ~]$ srvctl stop database -d westdb

[oracle@wnode03 ~]$
```

3. From the GDSCTL prompt on enode01, observe that the sales_reporting_srvc global service is automatically failed over to EASTDB database as per the Service attributes that we have defined.

```
On enode01:

GDSCTL> services
Service "sales_reporting_srvc.sales.oradbcloud" has 2
instance(s). Affinity: ANYWHERE
   Instance "sales%1", name: "eastdb1", db: "eastdb", region:
"east", status: ready.
   Instance "sales%2", name: "eastdb2", db: "eastdb", region:
"east", status: ready.

GDSCTL> databases
Database: "eastdb" Registered: Y State: Ok ONS: Y. Role: PRIMARY
Instances: 2 Region: east
   Service: "sales_reporting_srvc" Globally started: Y Started: N
            Scan: N Enabled: Y Preferred: Y
   Registered instances:
     sales%1
     sales%2
Database: "westdb" Registered: N State: Ok ONS: N. Role: N/A
Instances: 0 Region: west
Service: "sales_reporting_srvc" Globally started: Y Started: N
Scan: N Enabled: Y Preferred: Y

GDSCTL>
```

4. From the GDSCTL prompt on enode01, stop and remove the service that was created for this test case.

```
On enode01:

GDSCTL> stop service -service sales_reporting_srvc -gdspool sales
-database eastdb

GDSCTL> remove service -gdspool sales -service
sales_reporting_srvc
```

```
GDSCTL> exit

[oracle@enode01 ~]$
```

5. From the wnode03 terminal, bring the WESTDB database up.

```
[oracle@wnode03 ~]$ srvctl start database -d westdb

[oracle@wnode03 ~]$
```

Practice 8-3: Role-Based Global Services

Overview

When a Data Guard role transition is performed either manually or via Fast-Start Failover, GDS automatically relocates the global services based on the role of the databases. GDS does this without the Oracle Clusterware.

To comprehend how the role-based global services function, create two global services. `sales_entry_srvc` targeted to run on the Primary (EASTDB) and `sales_adhoc_srvc` targeted to run on the standby (WESTDB). Then execute the Data Guard switchover operation and observe that upon the role change, the `sales_entry_srvc` global service automatically gets relocated to the new primary (WESTDB) and the `sales_adhoc_srvc` to the new Standby (EASTDB).

1. Using GDSCTL, create the global services `sales_entry_srvc` and `sales_adhoc_srvc`. Add instances as shown below.

```
[oracle@enode01 ~]$ export
ORACLE_HOME=/u01/app/oracle/product/12.1.0/gsmhome_1

[oracle@enode01 ~]$ export ORACLE_BASE=/u01/app/oracle

[oracle@enode01 ~]$ export PATH=$ORACLE_HOME/bin:$PATH

[oracle@enode01 ~]$ gdsctl
GDSCTL: Version 12.1.0.2.0 - Production on Fri Aug 22 18:39:20
UTC 2014

Copyright (c) 2011, 2014, Oracle.  All rights reserved.

Welcome to GDSCTL, type "help" for information.

Current GSM is set to GSMEAST

GDSCTL> add service -service sales_entry_srvc -gdspool sales
-preferred_all -role PRIMARY

GDSCTL> modify service -gdspool sales -service sales_entry_srvc
-database eastdb -add_instances -preferred eastdb1 -available
eastdb2

GDSCTL> modify service -gdspool sales -service sales_entry_srvc
-database westdb -add_instances -preferred westdb1 -available
westdb2

GDSCTL> start service -service sales_entry_srvc -gdspool sales
```

```
GDSCTL> services
Service "sales_entry_srvc.sales.oradbcloud" has 1 instance(s).
Affinity: ANYWHERE
   Instance "sales%1", name: "eastdb1", db: "eastdb", region:
"east", status: ready.

GDSCTL> add service -service sales_adhoc_srvc -gdspool sales
-preferred_all -role PHYSICAL_STANDBY

GDSCTL> modify service -gdspool sales -service sales_adhoc_srvc
-database westdb -add_instances -preferred westdb2 -available
westdb1
Catalog connection is established

GDSCTL> modify service -gdspool sales -service sales_adhoc_srvc
-database eastdb -add_instances -preferred eastdb2 -available
eastdb1

GDSCTL> start service -service sales_adhoc_srvc -gdspool sales

GDSCTL> services
Service "sales_adhoc_srvc.sales.oradbcloud" has 1 instance(s).
Affinity: ANYWHERE
   Instance "sales%12", name: "westdb2", db: "westdb", region:
"west", status: ready.
Service "sales_entry_srvc.sales.oradbcloud" has 1 instance(s).
Affinity: ANYWHERE
   Instance "sales%1", name: "eastdb1", db: "eastdb", region:
"east", status: ready.

GDSCTL> databases
Database: "eastdb" Registered: Y State: Ok ONS: Y. Role: PRIMARY
Instances: 2 Region: east
   Service: "sales_adhoc_srvc" Globally started: Y Started: N
            Scan: Y Enabled: Y Preferred: Y
   Service: "sales_entry_srvc" Globally started: Y Started: Y
            Scan: N Enabled: Y Preferred: Y
   Registered instances:
     sales%1
     sales%2
Database: "westdb" Registered: Y State: Ok ONS: Y. Role:
PH_STNDBY Instances: 2 Region: west
   Service: "sales_adhoc_srvc" Globally started: Y Started: Y
            Scan: Y Enabled: Y Preferred: Y
   Service: "sales_entry_srvc" Globally started: Y Started: N
```

```
            Scan: N Enabled: Y Preferred: Y
   Registered instances:
     sales%11
     sales%12

GDSCTL>
```

2. On any of the database nodes, With DGMGRL, perform the Data Guard switchover. In this example, the switchover is performed from `wnode03`.

```
[oracle@wnode03 ~]$ dgmgrl "sys/oracle_4U@eastdb"
DGMGRL for Linux: Version 12.1.0.1.0 - 64bit Production
Copyright (c) 2000, 2012, Oracle. All rights reserved.
Welcome to DGMGRL, type "help" for information.
Connected as SYSDBA.

DGMGRL> show configuration
  Configuration - dg_config
  Protection Mode: MaxAvailability
  Databases:
  eastdb - Primary database
    westdb - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS

DGMGRL> switchover to westdb
Performing switchover NOW, please wait...
Operation requires a connection to instance "westdb2" on
database "westdb"
Connecting to instance "westdb2"...
Connected as SYSDBA.
New primary database "westdb" is opening...
Oracle Clusterware is restarting database "eastdb" ...
Switchover succeeded, new primary is "westdb"

*** Wait a few minutes and check the configuration again ***

DGMGRL> show configuration
Configuration - dg_config

  Protection Mode: MaxPerformance
```

```
  Members:
  westdb - Primary database
    eastdb - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS   (status updated 71 seconds ago)

DGMGRL> exit

[oracle@wnode03 ~]$
```

3. From the GDSCTL prompt on `enode01`, observe that the `sales_entry_srvc` global service is automatically relocated to the new Primary (WESTDB) and the `sales_adhoc_srvc` global service to the new standby (EASTDB).

```
From enode01:

GDSCTL> services
Service "sales_adhoc_srvc.sales.oradbcloud" has 1 instance(s).
Affinity: ANYWHERE
   Instance "sales%2", name: "eastdb2", db: "eastdb", region:
"east", status: ready.
Service "sales_entry_srvc.sales.oradbcloud" has 1 instance(s).
Affinity: ANYWHERE
   Instance "sales%11", name: "westdb1", db: "westdb", region:
"west", status: ready.

GDSCTL> databases
Database: "eastdb" Registered: Y State: Ok ONS: Y. Role:
PH_STNDBY Instances: 2 Region: east
   Service: "sales_adhoc_srvc" Globally started: Y Started: Y
            Scan: N Enabled: Y Preferred: Y
   Service: "sales_entry_srvc" Globally started: Y Started: N
            Scan: N Enabled: Y Preferred: Y
   Registered instances:
     sales%1
     sales%2
Database: "westdb" Registered: Y State: Ok ONS: Y. Role: PRIMARY
Instances: 2 Region: west
   Service: "sales_adhoc_srvc" Globally started: Y Started: N
            Scan: N Enabled: Y Preferred: Y
   Service: "sales_entry_srvc" Globally started: Y Started: Y
            Scan: N Enabled: Y Preferred: Y
```

```
   Registered instances:
     sales%11
     sales%12

GDSCTL>
```

4. At this point, you may stop and remove the services that we have created for this exercise.

```
From enode01:

GDSCTL> stop service -service sales_entry_srvc -gdspool sales
-database westdb

GDSCTL> remove service -gdspool sales -service sales_entry_srvc

GDSCTL> stop service -service sales_adhoc_srvc -gdspool sales
-database eastdb

GDSCTL> remove service -gdspool sales -service sales_adhoc_srvc

GDSCTL> exit

[oracle@enode01 ~]$
```

Practice 8-4: Replication Lag-Based Routing

Overview

Sometimes the Data Guard standby databases may lag behind the primary database due to various reasons. If the replication lag exceeds the lag limit, the global service is relocated to another available database that lags below the threshold. With the `-failover_primary` clause, you can even relocate the service to the Primary database.

Tasks

1. To understand the Lag tolerance-based routing, create a global Service `sales_reader_lag15_srvc` and set the `-lag` attribute to 15 seconds. Add instances using the `modify service` command as shown below. We will artificially create the lag by turning off the Apply process. We then will observe that once the lag exceeds the 15 seconds threshold, GDS automatically relocates the global service to the Primary (since we used the `-failover_primary` clause).

```
[oracle@enode01 ~]$ export
ORACLE_HOME=/u01/app/oracle/product/12.1.0/gsmhome_1

[oracle@enode01 ~]$ export ORACLE_BASE=/u01/app/oracle

[oracle@enode01 ~]$ export PATH=$ORACLE_HOME/bin:$PATH

[oracle@enode01 ~]$ gdsctl
GDSCTL: Version 12.1.0.2.0 - Production on Fri Aug 22 18:39:20
UTC 2014

Copyright (c) 2011, 2014, Oracle.  All rights reserved.

Welcome to GDSCTL, type "help" for information.

Current GSM is set to GSMEAST

GDSCTL> add service -service sales_reader_lag15_srvc -gdspool
sales -preferred_all -role PHYSICAL_STANDBY -lag 15 -
failover_primary

GDSCTL> modify service -gdspool sales -service
sales_reader_lag15_srvc -database westdb -add_instances -
preferred westdb1,westdb2

GDSCTL> modify service -gdspool sales -service
sales_reader_lag15_srvc -database eastdb -add_instances -
preferred eastdb1,eastdb2
```

```
GDSCTL> start service -service sales_reader_lag15_srvc -gdspool
sales

GDSCTL> services
Service "sales_reader_lag15_srvc.sales.oradbcloud" has 2
instance(s). Affinity: ANYWHERE
   Instance "sales%1", name: "eastdb1", db: "eastdb", region:
"east", status: ready.
   Instance "sales%2", name: "eastdb2", db: "eastdb", region:
"east", status: ready.

GDSCTL> databases
Database: "eastdb" Registered: Y State: Ok ONS: Y. Role:
PH_STNDBY Instances: 2 Region: east
   Service: "sales_reader_lag15_srvc" Globally started: Y
Started: Y
            Scan: Y Enabled: Y Preferred: Y
   Registered instances:
     sales%1
     sales%2
Database: "westdb" Registered: Y State: Ok ONS: Y. Role: PRIMARY
Instances: 2 Region: west
   Service: "sales_reader_lag15_srvc" Globally started: Y
Started: N
            Scan: Y Enabled: Y Preferred: Y
   Registered instances:
     sales%11
     sales%12

GDSCTL> config service -service sales_reader_lag15_srvc
Name: sales_reader_lag15_srvc
Network name: sales_reader_lag15_srvc.sales.oradbcloud
Pool: sales
Started: Yes
Preferred all: Yes
Locality: ANYWHERE
Region Failover: No
Role: PHYSICAL_STANDBY
Primary Failover: Yes
Lag: 15
Runtime Balance: SERVICE_TIME
Connection Balance: LONG
Notification: Yes
TAF Policy: NONE
```

```
Policy: AUTOMATIC
DTP: No
Failover Method: NONE
Failover Type: NONE
Failover Retries:
Failover Delay:
Edition:
PDB:
Commit Outcome:
Retention Timeout:
Replay Initiation Timeout:
Session State Consistency:
SQL Translation Profile:


Databases
------------------------
Database                       Preferred Status
--------                       --------- ------
eastdb                         Yes       Enabled
westdb                         Yes       Enabled

GDSCTL>
```

2. On `wnode03`, start DGMGRL and stop the apply process on `enode01`.

```
[oracle@wnode03 ~]$ dgmgrl "sys/oracle_4U@eastdb"
DGMGRL for Linux: Version 12.1.0.1.0 - 64bit Production
Copyright (c) 2000, 2012, Oracle. All rights reserved.
Welcome to DGMGRL, type "help" for information.
Connected as SYSDBA.

DGMGRL> show configuration
Configuration - dg_config

  Protection Mode: MaxPerformance
  Members:
  westdb - Primary database
    eastdb - Physical standby database

Fast-Start Failover: DISABLED
```

```
Configuration Status:
SUCCESS   (status updated 14 seconds ago) S

DGMGRL> show database westdb
Database - westdb

  Role:               PRIMARY
  Intended State:     TRANSPORT-ON
  Instance(s):
    westdb1
    westdb2

DGMGRL> show database eastdb
Database - eastdb

  Role:               PHYSICAL STANDBY
  Intended State:     APPLY-ON
  Transport Lag:      0 seconds (computed 0 seconds ago)
  Apply Lag:          0 seconds (computed 0 seconds ago)
  Average Apply Rate: 9.00 KByte/s
  Real Time Query:    ON
  Instance(s):
    eastdb1
    eastdb2 (apply instance)

Database Status:
SUCCESS

DGMGRL> edit database eastdb set state='APPLY-OFF';
Succeeded.

DGMGRL> show database eastdb
Database - eastdb

  Role:               PHYSICAL STANDBY
  Intended State:     APPLY-OFF
  Transport Lag:      0 seconds (computed 1 second ago)
  Apply Lag:          33 seconds (computed 1 second ago)
  Average Apply Rate: (unknown)
  Real Time Query:    OFF
  Instance(s):
    eastdb1
```

```
    eastdb2 (apply instance)

Database Status:
SUCCESS
DGMGRL>
```

3. Wait for 15 seconds and run the services command from GDSCTL and observe that the `sales_reader_lag15_srvc` global service has failed over to primary.

```
From enode01:

GDSCTL> services
Service "sales_reader_lag15_srvc.sales.oradbcloud" has 2
instance(s). Affinity: ANYWHERE
   Instance "sales%11", name: "westdb1", db: "westdb", region:
"west", status: ready.
   Instance "sales%12", name: "westdb2", db: "westdb", region:
"west", status: ready.

GDSCTL> databases
Database: "eastdb" Registered: Y State: Ok ONS: Y. Role:
PH_STNDBY Instances: 2 Region: east
   Service: "sales_reader_lag15_srvc" Globally started: Y
Started: N
            Scan: Y Enabled: Y Preferred: Y
   Registered instances:
     sales%1
     sales%2
Database: "westdb" Registered: Y State: Ok ONS: Y. Role: PRIMARY
Instances: 2 Region: west
   Service: "sales_reader_lag15_srvc" Globally started: Y
Started: Y
            Scan: Y Enabled: Y Preferred: Y
   Registered instances:
     sales%11
     sales%12
GDSCTL>
```

4. Make sure that the Apply Process is started.

```
*** On wnode03

DGMGRL> edit database eastdb set state='APPLY-ON';
Succeeded.
```

This exercise illustrated replication lag tolerance-based routing for Active Data Guard configuration, which allows applications achieve better data quality. Instead of accessing data in the standby database that is lagging behind, you can automatically relocate the service to a database that is not lagging more than the defined threshold.