

gDBClone

A Simple Approach to Managing Test and Development Environments
Leveraging ACFS Snapshots

ORACLE WHITE PAPER | AUGUST 2017



Table of Contents

Executive Overview	4
Database Provisioning Lifecycle and Challenges	4
Managing Test & Dev Environments Does Not Have to be Complex	5
Purpose of Database Duplication	6
Test and Dev environment	6
Database Clone vs Database Snapshot creation time	8
Supported Configurations and Features	8
gDBClone Clone	8
gDBClone Snap	10
gDBClone Convert	12
gDBClone ListDBs & DelDB	12
gDBClone ListHomes	12
gDBClone ListSnaps & DelSnap	12
gDBClone SYSPwF	12
gDBClone Command Syntax	13
gDBClone Installation	14
gDBClone deinstallation	14
Managing gDBClone Privileges and Security with SUDO	15
Allowing Root User Access Using SUDO	15
SUDO Example 1: Allow a User to Perform Any gDBClone Operation	15
SUDO Example 2: Allow a User to Perform Only Selected gDBClone Operations	15



SUDO Example 3: Allow a User to Perform Any gDBClone Operation without password request	15
Limitations & Considerations	16
Dependency	16
Source database using Transparent Data Encryption (TDE)	16
Database clone/snap overwriting SGA parameters	16
gDBClone usage example	17
1. Clone a Remote/Local database to ACFS(Gold/Image)	17
2. Clone a Remote/Local database to ASM and make it a RAC database	18
3. Clone a 12c Multitenant database to ACFS	18
4. Snapshot a gold/master database as RAC OneNode	19
5. Convert a database (SI or RAC OneNode) to RAC	19
6. Convert a non-CDB database to PDB of a given CDB	19
7. Delete database	19
8. List databases	20
9. Create an encrypted SYS password file	20
Case Studies	21
1. Managing a test & dev environment combined with Oracle Data Guard	21
2. Creating database clone using RMAN backupsets	27
3. Clone 11g Database from ASM to ACFS keeping the source running	29
4. Create a RAC snapshot database from a GOLD clone running database	30
5. Create a snapshot RAC database from a standby database	32
6. Clone a database from RMAN full backup to ACFS as standby Database	34
7. Database upgrade using Transient Logical Standby(TLS)	36



8.	Clone a database encrypted with Transparent Data Encryption (TDE)	37
9.	Using gDBClone in Oracle Public Cloud (RACDBaaS)	40
10.	Using gDBClone on ODA X6-2 S,M,L (Enterprise Edition)	44
11.	Migrate a database from OPC to BMC using gDBClone	49
12.	Test & Dev Management environment example	55
Conclusion		57
Appendix – A		58
Clone Location		58
Snap Location		59
Standby option		60

Executive Overview

As database-driven applications grow rapidly, maximizing agility, reducing management overhead and cost savings are top priority for IT organizations today. Customers must have solutions to contain data redundancy that is sprawling out of control. On the average, more than 10 full copies of a production databases are created for test, development and reporting purposes.

In addition to redundant data, customers struggle with ever increasing management overhead required for managing database life cycles. Frequent challenges are provisioning of databases and deployment in test and development environments frequently, efficiently, quickly and cost effectively.

Oracle addresses management of test and development environments with advanced products and technologies to realize:

- » Simplicity
- » Cost savings
- » Reduction of management overhead
- » Agility

Database Provisioning Lifecycle and Challenges

Managing database test and development (test & dev) environments can be challenging and costly in time and resources. Production databases often require 8-10 or more copies for varies types of test and development purposes. Each copy of a database consumes significant storage space. Database copies are typically recycled (created, deleted or refreshed) often. Conventional ways of manually managing test & dev environments can be complex, costly and time consuming. Test & dev life cycle can be defined at a high level as follows:

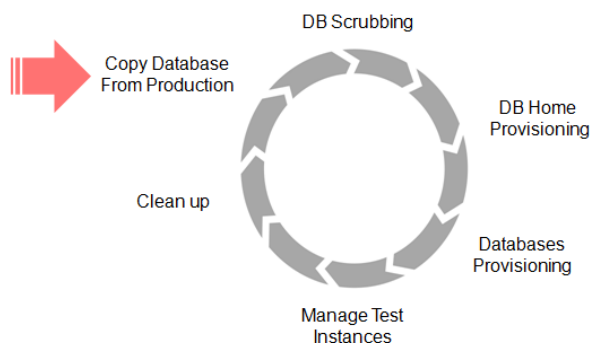



Figure 1



An initial copy of a production database is created on a test & dev cluster as a master copy. Data scrubbing may be required either on the production server or on the test cluster as well. Database or data scrubbing could mean data filtering, redaction or any other technique the user chooses to use in order to provide only the data set that is needed or authorizes for test and development purposes. A database home should be identified or provisioned in preparation for deploying a test database. Multiple copies of databases may be provisioned off the master copy on the test cluster. Database administrators manage the environments and clean up when a database copy is no longer needed.

Managing Test & Dev Environments Does Not Have to be Complex

gDBClone is a tool that was developed to provide a simple and efficient method for cloning a database for test and dev environments. gDBClone leverages Oracle Automatic Storage Management Cluster File System (ACFS) snapshot functionality to create space efficient copies of databases and manage a test and dev database life cycle. ACFS is a filesystem that's provided by Oracle on various OS platforms and really integrates into Oracle ASM (Automatic Storage Management). It's a very powerful Cluster Filesystem but it's not distributed as part of the Operating System, it's distributed with the Oracle Grid Infrastructure. The key enabling technology was introduced in Oracle Database 12.1 that allows creation of Oracle Database files directly on the ACFS file system (RDBMS 11.2.0.4 and up) and therefore benefiting from ACFS point in time snapshot functionality for sparse database provisioning efficiently.

gDBClone performs seven key functions:

- » **Clone:** Creates a clone database (as Primary or as Standby) from a production database copying the DB to the target test and dev cluster
- » **Snap:** Creates sparse snapshots of the DB to be used for test and development
- » **Convert:** Converts a given database to RAC (Real Application Cluster) OneNode, RAC or from non-CDB (non-container database) to a PDB (pluggable database) of a given CDB
- » **ListDBs:** Lists the cloned databases and its snapshots
- » **DelDB:** Deletes cloned databases and/or its snapshots
- » **ListHomes:** Lists the available oracle home
- » **SYSPwF:** Creates an encrypted password file

Purpose of Database Duplication

A duplicate database is useful for a variety of purposes, most of which involve testing & upgrade. You can perform the following tasks in a duplicate database:

- » Test backup and recovery procedures
- » Test an upgrade to a new release of Oracle Database
- » Test the effect of applications on database performance
- » Create a standby database (Dataguard)
- » Leverage on Transient Logical Standby (TLS) to perform an upgrade
- » Generate reports

For example, you can duplicate the production database on host1 to host2, and then use the duplicate database on host2 to practice restoring and recovering this database while the production database on host1 operates as usual.

Test and Dev environment

The following diagram illustrates a typical test and dev environment that can be created and managed with the gDBClone command.

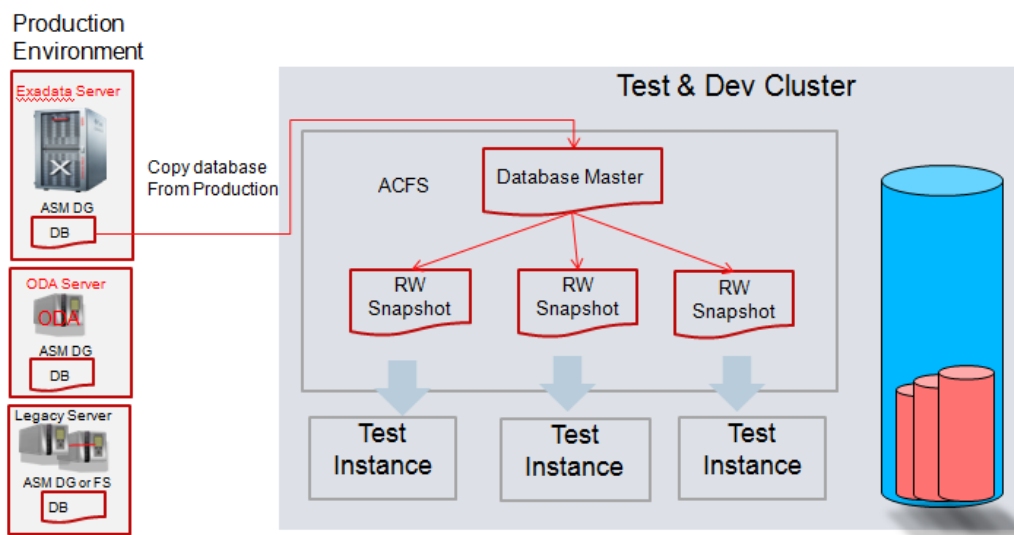



Figure 2 – Test & Dev environment

In this example, a copy of a production database is cloned into the test & dev cluster with a single gDBClone command (`gDBClone clone`). The source database may be on an Exadata Database Machine or any other legacy server and any type of file system including Oracle ASM.



gDBClone is utilized to provision sparse space efficient copies of databases (“gDBClone snap”). These copies may be deployed for test and dev purposes. Only small incremental storage is required by the snapshots after the initial creation of the master copy as illustrated by the storage capacity illustration on the right of figure 2 above.

An Oracle ACFS snapshot is an online, read only or read write, point in time copy of an Oracle ACFS file system. The snapshot copy is space efficient and uses Redirect-on-Write (ACFS ROW) functionality. Before an Oracle ACFS file extent is modified or deleted, its current value is preserved in the snapshot to maintain the point in time view of the file system. Oracle ACFS supports 1023 snapshots per file system

Database Clone vs Database Snapshot creation time

The database clone creation time depends on the database size and on the network throughput. In case of database snapshot the cloning operation is a very fast operation as it's independent on the database size and/or network speed. In the following example, we compare a clone/snap of 5Gb vs 25Gb database size:

- Database Clone

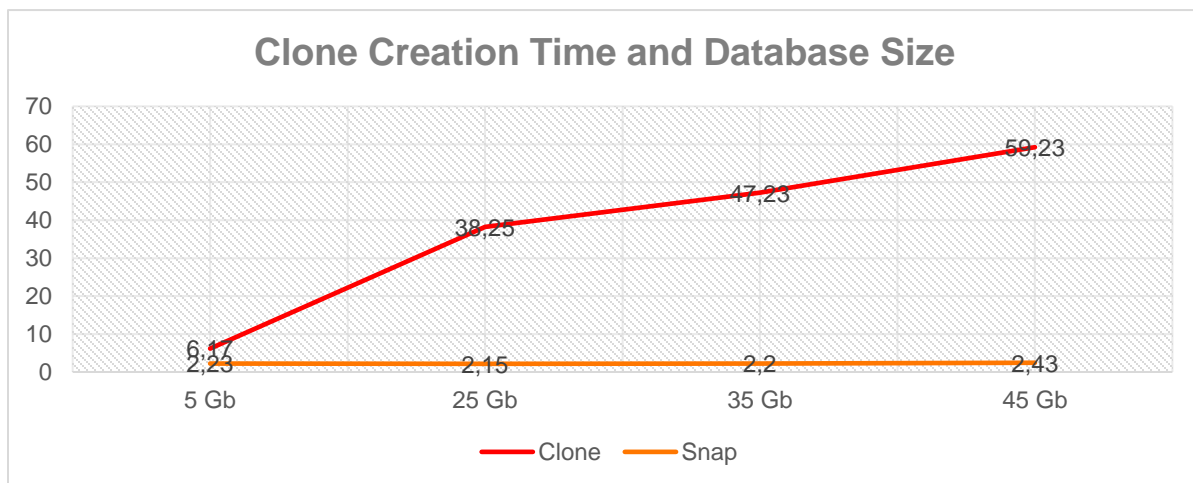
```
5Gb Database Clone Creation Time:
real 6m17.758s
user 0m12.904s
sys 0m1.028s
```

```
25Gb Database Clone Creation Time:
real 38m35.636s
user 0m13.192s
sys 0m1.001s
```

- Database Snap

```
5Gb Database Clone Creation Time:
real 2m23.723s
user 0m7.671s
sys 0m0.974s
```

```
25Gb Database Clone Creation Time:
real 2m15.842s
user 0m7.667s
sys 0m1.001s
```



Supported Configurations and Features

gDBClone Clone

Creates a clone database (as Primary or as Physical Standby) from a production database duplicating (physical copy) the DB to the Test & Dev cluster using “RMAN Duplicate from Active Database” (by default gDBClone is allocating 3 RMAN channels, you may overwrite it using “*-channels <RMAN channels number>*” command option). The source database may be on an Exadata Database Machine or any other legacy server and any type of file system including Oracle ASM. gDBClone needs to connect the remote database normally through the SCAN (Single Client Access Network) listener.

It's also possible clone a production database from a given RMAN full backup location (the full backup can be on NFS). The source database can be SI (Single Instance), RAC OneNode or RAC. The target clone database can be SI (Single Instance), RAC OneNode or RAC (by default it will be SI). The 12c Multitenant container databases are supported.

The gDBClone “clone” option can be used to instantiate a Dataguard environment and the standby can be and “Active Standby” or a “Real Time Apply”. The “gDBClone clone” is done without special impact on the source production database.

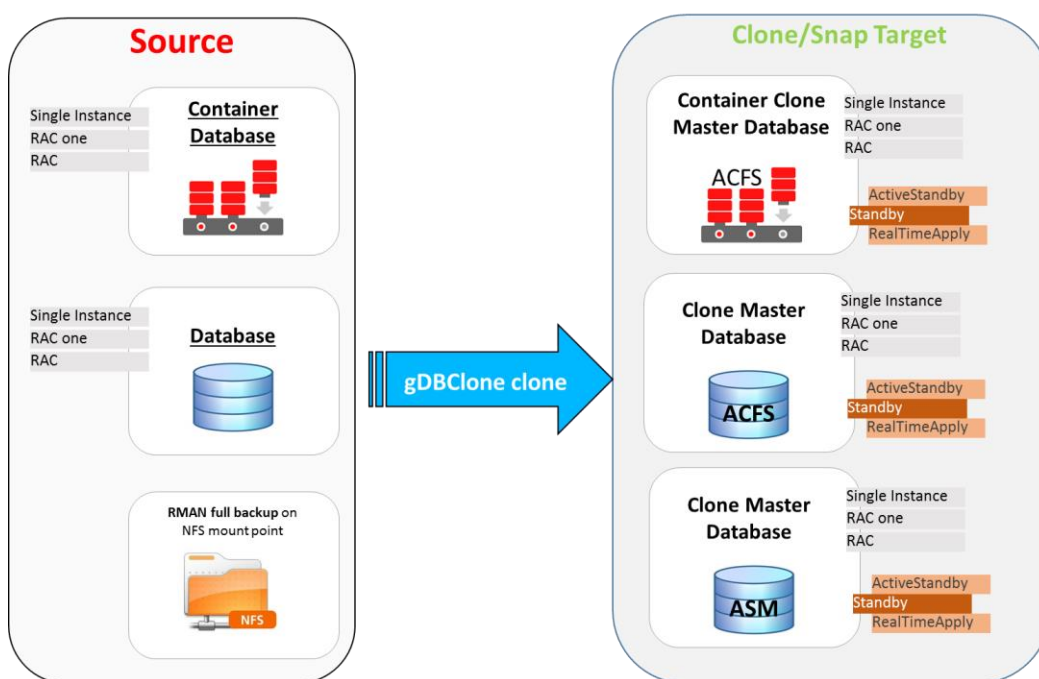


Figure 3 - gDBClone **clone** capabilities

On cloning a remote or local database 3 different ACFS mount points are possible:

- dataacfs <acfs mount point> → Database datafiles target ACFS storage
- redoacfs <acfs mount point> → Database redologs target ACFS storage (default dataacfs)
- recoacfs <acfs mount point> → Database recovery target ACFS storage (default dataacfs)

gDBClone can be used to clone a database to ASM, in such case, 3 different disk group are possible:

- datadg <ASM diskgroup> → Database datafiles target ASM disk group (default +DATA)
- redodg <ASM diskgroup> → Database redologs target ASM disk group (default +REDO)
- recodg <ASM diskgroup> → Database recovery target ASM disk group (default +RECO)

Note: cloning a database to ASM you cannot leverage later on the database gDBClone snapshot feature

gDBClone Snap

Creates sparse snapshots of the DB to be used for test and development. The source database must be stored on local Oracle ACFS filesystem.

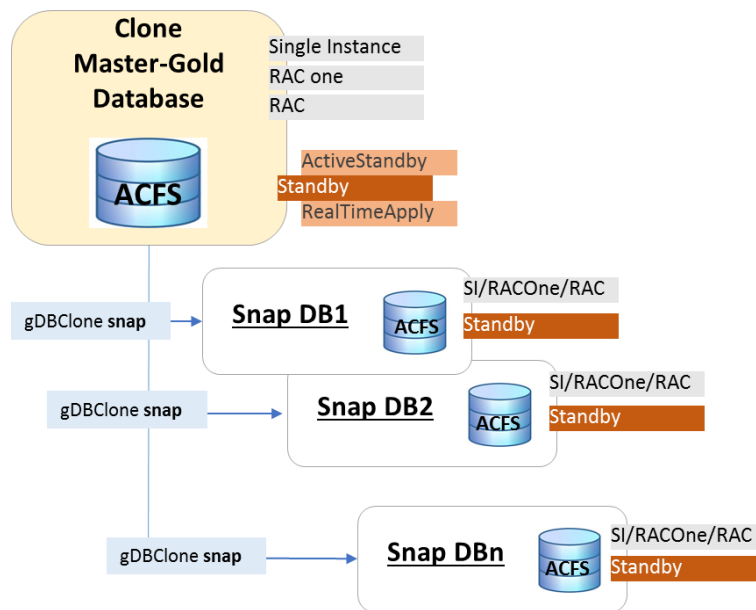


Figure 4 - gDBClone snap capabilities

The source database can be SI (Single Instance), RAC OneNode or RAC, primary or standby. The target snapshot database can be SI (Single Instance), RAC OneNode or RAC (by default it will be SI), primary or standby. The gDBClone is introducing the **“Hot Database Snapshot as Standby”** capability. Without impact over the source database and "without" storage duplication, leveraging on ACFS snapshot *redirect-on-write (ACFS ROW)* feature, gDBClone is making a snapshot of a running database and if "-standby" option is used the result will be a standby database.

Note: the possibility to get a physical standby from a running database without downtime is the key to make database upgrade leveraging on TLS (Transient Logical Standby)

The gDBClone supports the snapshot of a running Standby database without production impact leveraging on the “Snapshot Standby” database feature.

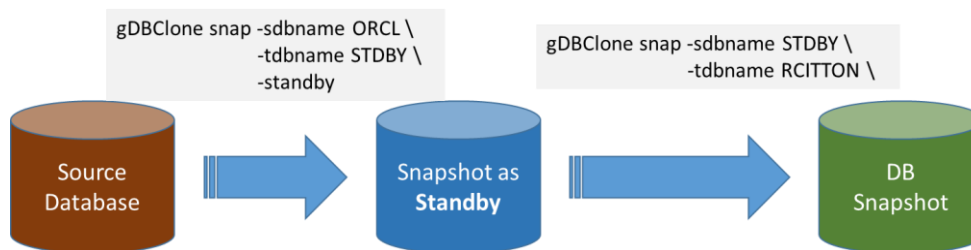


Figure 5 - Snapshot as Standby Database & Snapshot of Standby Database support

gDBClone is also supporting “Multi ACFS database file locations” and “database snapshot for different ORACLE_HOMEs

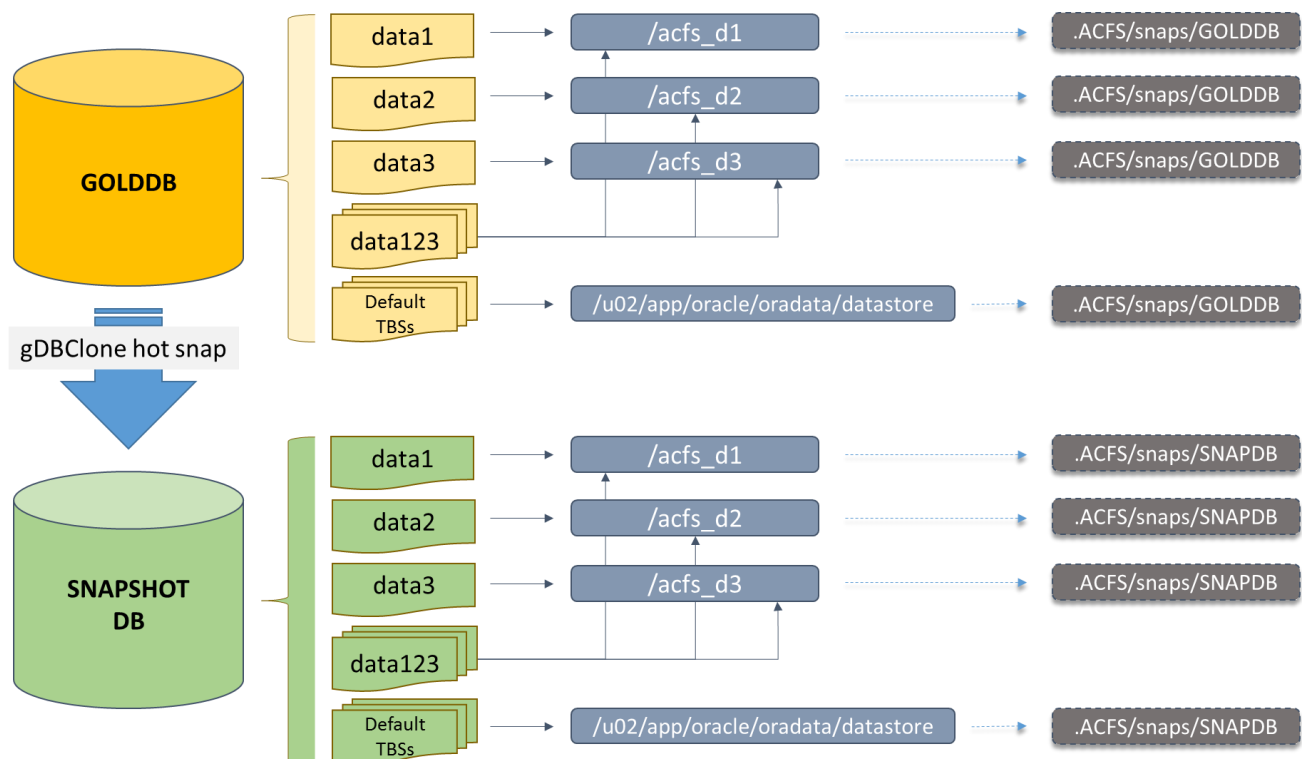



Figure 6 - Multi ACFSs Database Locations Hot Snapshot Capability Support



gDBClone Convert

Beside “clone” and “snap” features, gDBClone can be used to convert a database to a RAC or RAC OneNode database and it can be used to convert a 12c non-container database(non-CDB) to a Pluggable database (PDB) of a given CDB.

gDBClone ListDBs & DelDB

gDBClone also provides single command to verify your database environments, providing parent/child relation in case of snap-of-snap database, and delete databases that are no longer in use.

gDBClone ListHomes

With “listhomes” command option you can check which available ORACLE_HOMEs are available. The oracle home name will be used later to “attach” the clone/snap database (“-tdbhome”).

gDBClone ListSnaps & DelSnap

You could use gDBClone to list and remove ACFS snapshot

gDBClone SYSPwF

Using “gDBClone syspwf” an encrypted password file will be created. Such password is the SYS source remote database password. Doing a clone/snap with “-syspwf <sys password file>” option, gDBClone will use the encrypted password, otherwise it will request at command line. If a file with the name: “SYSpasswd_file” is present under gDBClone home (“/opt/gDBClone”), at clone/snap time, gDBClone will check for the password from that file, in such case you can avoid the “-syspwf” option and no password request are done at command line.

gDBClone Command Syntax

```
gDBClone clone -sdbname <source DB name>
               -sdbscan <source DB Host SCAN name> | -sbckloc '<backup location path>'
               -tdbname <Target Database Name> -tdbhome <Target Database Home Name>
               { -dataacfs <acfs mount point> [ -redoacfs <acfs mount point> ] [ -recoacfs <acfs mount point> ] } |
               { -datadg <asm data diskgroup> [ -redodg <asm redo diskgroup> ] [ -recodg <asm reco diskgroup> ] }
               [ -sga_max_size <size Mb> ] [ -sga_target <size Mb> ] | [ -pfile ]
               [ -channels <RMAN channels number> ]
               [ -sdbport <Source DB SCAN Listener Port> ] [ -tdbport <Target DB SCAN Listener Port> ]
               [ -standby [-pmode maxperf|maxavail|maxprot] [-activedg] [-rtapply] ]
               [ -racmod <db type> ]
               [ -opc ]
               [ -syspwf <sys password file>]
gDBClone snap -sdbname <source DB name> -tdbname <Target Database Name>
               [ -tdbhome <Target Database Home Name> ]
               [ -sga_max_size <size Mb> ] [ -sga_target <size Mb> ] | [ -pfile ]
               [ -standby [-pmode maxperf|maxavail|maxprot] [-activedg] [-rtapply] ]
               [ -sdbport <SCAN Listener Port> ]
               [ -racmod <db type> ]

gDBClone convert -sdbname <source noCDB name>
                 -racmod <1|2> | -tdbname <target CDB name> [-check] {[-copy] [-path <path>]}
                 [ -syspwf <sys password file>] [ -tsyspwf <sys password file>]

gDBClone listhomes [ -verbose ]

gDBClone listdbs [ -tree ] | [ -verbose ]
gDBClone deldb -tdbname <database name> [ -force ]

gDBClone listsnaps -dataacfs <acfs_mount_point> [ -tree ]
gDBClone delsnap -snapname <snapshot name> -dataacfs <acfs_mount_point>

gDBClone syspwf -syspwf <SYS encrypted password file path>

gDBClone OPTIONS
-sdbname      Source Database Name
-sdbscan      Source DB Host SCAN Name
-sdbport      Source SCAN Listener Port (default 1521)
-sbckloc      Source RMAN Full Backup Location
-tdbname      Target Database Name
-tdbhome      Target Database Home Name
-tdbport      Target SCAN Listener Port (default 1521)
-standby      The clone/snap will be a physical standby database
-pmode        Standby option: maxperf|maxavail|maxprot (default maxperf)
-activedg     Enable Active Dataguard
-rtapply      Enable real time apply
-racmod        0/1/2 == SINGLE/RACONE/RAC (default 0)
-dataacfs     Database datafiles target ACFS storage
-redoacfs     Database redologs target ACFS storage (default dataacfs)
-recoacfs     Database recovery target ACFS storage (default dataacfs)
-datadg       Database datafiles target ASM diskgroup (default +DATA)
-redodg       Database redologs target ASM diskgroup (default +REDO)
-recodg       Database recovery target ASM diskgroup (default +RECO)
-sga_max_size SGA Max Size (Mb)
-sga_target   SGA Target (Mb)
-pfile        Parameters file
-channels     RMAN allocate channels (default 3)
-opc          Required option on RACDBaaS environment
-syspwf       SYS encrypted password file
-tsypwf       SYS encrypted password file
-check        Will perform a CDB to PDB conversion pre-check
-copy         Will copy the source noCDB datafiles to CDB location (default: nocopy)
-path         Path where to copy the dbfiles (default CDB system dbf path)
-tree         With listdb will show the Parent/Snapshot tree
-verbose      Display OH & version on listdb
-force        With deldb will unregister the db
```

gDBClone Installation

gDBClone can be installed using the RPM (RedHat Package Manager) command as following:

```
# rpm -i gDBClone-3.0.2-X.noarch.rpm
```

(*) X=version number

or updating an installed version, issuing:

```
# rpm -Uvh gDBClone-3.0.2-X.noarch.rpm
```

(*) X=version number

Following files are created under ‘/opt/gDBClone’:

```
# tree /opt/gDBClone
/opt/gDBClone
├── gDBClone
│   └── lib
│       ├── gDBClone_AcfsUtils.pm
│       ├── gDBClone_Clone.pm
│       ├── gDBClone_DBConvert.pm
│       ├── gDBClone_GetDBConnection.pm
│       ├── gDBClone_Inventory.pm
│       ├── gDBClone_LoggingAndTracing.pm
│       ├── gDBClone_passwd.jar
│       ├── gDBClone_Queries.pm
│       ├── gDBClone_Snap.pm
│       ├── gDBClone_SqlUtils.pm
│       └── gDBClone_Utils.pm
```

1 directory, 12 files

gDBClone deinstallation

gDBClone can be removed using the RPM (RedHat Package Manager) command as following:

```
# rpm -e gDBClone-3.0.2-X.noarch
```

(*) X=version number

Managing gDBClone Privileges and Security with SUDO

gDBClone command-line utility requires root system privileges for most actions. You may want to use SUDO as part of your system auditing and security policy.

For most tasks, you need to log in as root to use the gDBClone command-line interface. If you are not logged in as root, then you cannot carry out most actions such as clone, snap.

Allowing Root User Access Using SUDO

In environments where system administration is handled by a different group than database administration, or where security is a significant concern, you may want to limit access to the root user account and password. SUDO enables system administrators to grant certain users (or groups of users) the ability to run commands as root, while logging all commands and arguments as part of your security and compliance protocol.

A SUDO security policy is configured by using the file `/etc/sudoers`. Within the sudoers file, you can configure groups of users and sets of commands to simplify and audit server administration with SUDO commands.

Caution: Configuring SUDO to allow a user to perform any operation is equivalent to giving that user root privileges. Consider carefully if this is appropriate for your security needs.

SUDO Example 1: Allow a User to Perform Any gDBClone Operation

This example shows how to configure SUDO to enable a user to perform any gDBClone operation. You do this by adding lines to the commands section in the `/etc/sudoers` file:

```
Cmnd_Alias GDBCLONE_CMD=/opt/gDBClone/gDBClone *
rcitton ALL = GDBCLONE_CMD
```

In this example, the user name is `rcitton`. The file parameter setting `ALL=GDBCLONE_CMDS` grants the user `rcitton` permission to run all gDBClone commands that are defined by the command alias `GDBCLONE_CMDS`.

SUDO Example 2: Allow a User to Perform Only Selected gDBClone Operations

To configure SUDO to allow an user to perform only selected gDBClone operations, add lines to the commands section in the `/etc/sudoers` file as follows:

```
Cmnd_Alias GDBCLONE_CMD=/opt/gDBClone/gDBClone clone
rcitton ALL = GDBCLONE_CMD
```

SUDO Example 3: Allow a User to Perform Any gDBClone Operation without password request

To configure SUDO to allow an user to perform gDBClone operations without password request, add lines to the commands section in the `/etc/sudoers` file as follows:

```
Cmnd_Alias GDBCLONE_CMD=/opt/gDBClone/gDBClone *
rcitton ALL=(root) NOPASSWD:GDBCLONE_CMD
```


Limitations & Considerations

gDBClone works on a Grid Infrastructure environment only. Source database must be in archivelog mode when cloning/snapshotting as it's executed a hot clone/snapshot. Multitenant database snapshot is not currently supported.

“gDBClone snap” needs EE (Enterprise Edition) Databases as the RMAN snapshot time recovery feature is needed and Grid Infrastructure version 12.1 or above. GI version 11g is not supported due to missing ACFS snapshot-of-snapshot feature capability.

Dependency

gDBClone may request “perl” and “perl-XML-Simple” package installation

```
# yum install -y perl perl-XML-Simple
```

Source database using Transparent Data Encryption (TDE)

Transparent Data Encryption (TDE), which functions at the column level, and tablespace encryption. If you are cloning a database with encrypted tablespaces, you must manually copy the keystore to the duplicate database. If the keystore is not an auto login (SSO) keystore, then you must convert it to an auto login keystore at the duplicate database. (See also case study at [page 37](#))

Database clone/snap overwriting SGA parameters

gDBClone is supporting the possibility to clone/snap a source database “overwriting” some SGA parameters. You can leverage on this feature using the “-pfile <parameters file>” command option. The supported parameters are the following:

```
aq_tm_processes
archive_lag_target
bitmap_merge_area_size
create_bitmap_area_size
db_block_checking
db_block_checksum
db_file_multiblock_read_count
db_files
db_lost_write_protect
fast_start_parallel_rollback
hash_area_size
job_queue_processes
log_archive_format
log_archive_max_processes
log_archive_trace
open_cursors
parallel_execution_message_size
```

```
parallel_max_servers
pga_aggregate_target
processes
recovery_parallelism
remote_login_passwordfile
sec_case_sensitive_logon
session_cached_cursors
sessions
sga_max_size
sga_target
shared_pool_reserved_size
shared_pool_size
shared_servers
sort_area_retained_size
sort_area_size
undo_management
undo_retention
```

It's possible “overwrite” just only `sga_max_size` & `sga_target`, using “-sga_max_size <size Mb>” & “-sga_target <size Mb>” gDBClone command option.

gDBClone usage example

1. Clone a Remote/Local database to ACFS(Gold/Image)

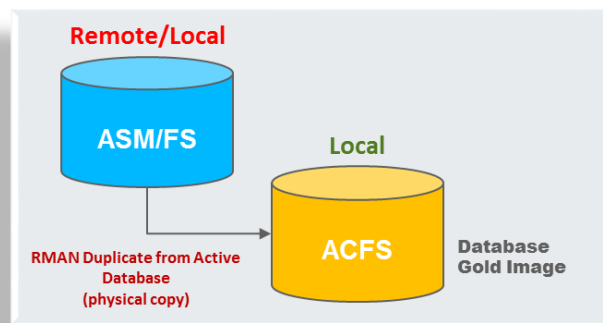


Figure 7 - Clone a remote/local Database to ACFS

gDBclone **clone** command options:

```
$ sudo /opt/gDBClone/gDBClone clone -sdbname ORCL \  
    -sdbscan exadata316-scan \  
    -tdbname GOLD \  
    -tdbhome OraDb12102_home1 \  
    -dataacfs /u02/app/oracle/oradata/datastore  
    [-redoacfs <acfs mount point>][-recoacfs <acfs mount point>]
```

You could use “-redoacfs” and/or “-recoacfs” to store redologs/archivelogs in different ACFS filesystems.

Note: If you need to decrease/increase the SGA footprint, example if your target local system cannot accommodate the source SGA, you can leverage on “-sga_max_size” and “-sga_target” gDBClone clone parameters (both expressed in Mb), or using the more comprehensive “-pfile” option.

2. Clone a Remote/Local database to ASM and make it a RAC database

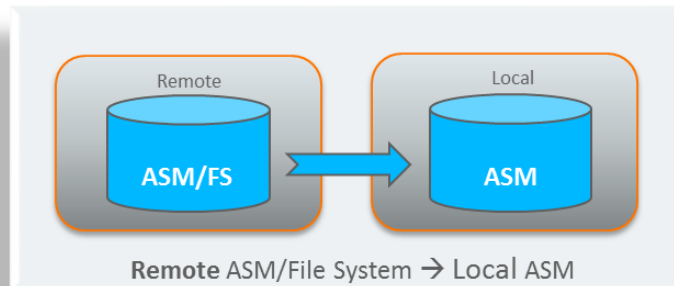


Figure 8 – Clone a remote/local database to ASM

gDBclone **clone** command options:

```
$ sudo /opt/gDBclone/gDBclone clone -sdbname ORCL \  
-sdbscan exadata316-scan \  
-tdbname GOLD \  
-tdbhome OraDb12102_home1 \  
-datadg +MYDATA \  
[-redodg <dgname>] [-recodg <dgname>]
```

Local default ASM diskgroup: +DATA, +REDO, +RECO

Local ASM diskgroup override: -datadg <dgname>, -redo <dgname>, -recodg <dgname>

3. Clone a 12c Multitenant database to ACFS

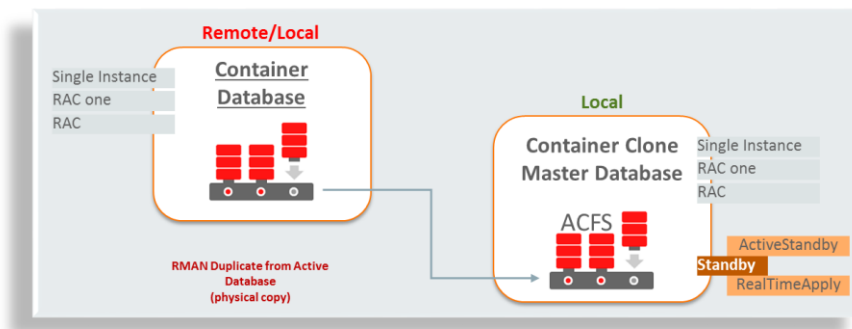


Figura 9 – Clone a 12c multitenant database to ACFS

gDBclone **clone** command options:

```
# gDBclone clone -sdbname ORCL \  
-sdbscan exadata316-scan \  
-tdbname GOLD \  
-tdbhome OraDb12102_home1 \  
-dataacfs /u02/app/oracle/oradata/datastore \  
[ -standby [-pmode maxperf|maxavail|maxprot] [-activedg] [-rtapply] ] \  
[ -racmod <db type> ]
```

Note: no extra options are needed, automatic CDB recognition

4. Snapshot a gold/master database as RAC OneNode

```
# gDBClone snap -sdbname GOLD \  
                  -tdbname SNAP \  
                  -racmod 1
```

Note: in this case as the “-tdbhome” option is not provided, the ORACLE_HOME will be the same source database’s ORACLE_HOME.

Note: If you need to decrease/increase the SGA footprint, example if your target local system cannot accommodate the source SGA, you can leverage on “-sga_max_size” and “-sga_target” gDBClone snap parameters (both expressed in Mb), or using the more comprehensive “-pfile” option.

5. Convert a database (SI or RAC OneNode) to RAC

```
# gDBClone convert -sdbname SNAP \  
                  -racmod 2
```

Note: “-racmod” 0/1/2 = SINGLE/RACONE/RAC (default 0). Convert RAC OneNode, RAC to single instance is not supported.

6. Convert a non-CDB database to PDB of a given CDB

```
# gDBClone convert -sdbname <source noCDB name>  
                  -tdbname <target CDB name>  
                  [-check] [[-copy] [-path <path>]]
```

-check Will perform a CDB to PDB conversion pre-check
-copy Will copy the source noCDB datafiles to CDB location (default: nocopy)
-path Path where to copy the dbfiles (default CDB system dbf path)

Note: before the conversion you may want execute “gDBClone convert -check” to verify the conversion result in “dry mode”. Using “-check” a report with warnings and potential conversion errors is generated for your review.

7. Delete database

```
# gDBClone delldb -tdbname SNAP -force  
INFO: 2016-12-15 01:29:56: Please check the logfile  
'/opt/gDBClone/out/log/gDBClone_91617.log' for more details  
  
You are going to drop the database SNAP, are you sure (Y/N)? y  
WARNING: 2016-12-15 01:29:58: ORACLE_BASE is not set  
INFO: 2016-12-15 01:29:58: Got Oracle Base from orabase  
SUCCESS: 2016-12-15 01:30:23: ACFS snapshot 'SNAP' on  
'/u02/app/oracle/oradata/datastore' has been deleted.
```

Note: without “-force” option, dbca will be used to delete the database

8. List databases

Having the following scenario, “gDBClone listdb” will list relations and database type

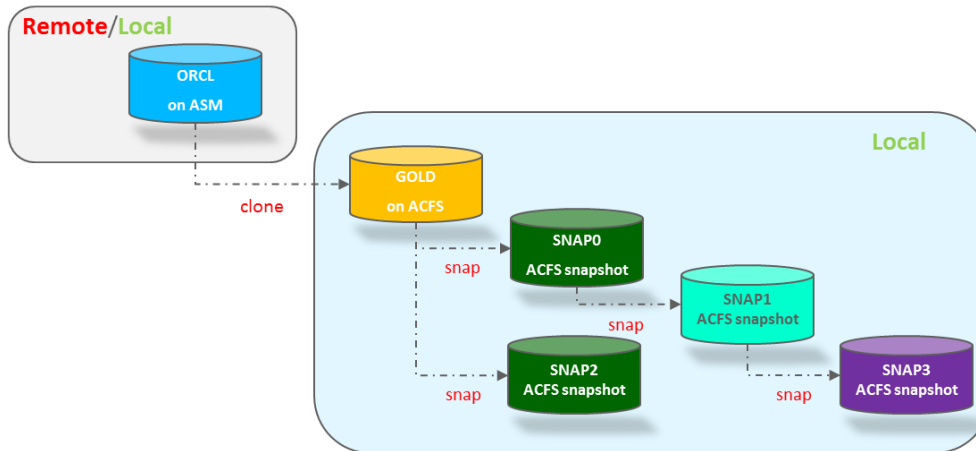


Figure 10 – gDBClone ListDB scenario

```
# ./gDBClone listdb -tree
Parent  Child
-----
GOLD
      SNAP2
      SNAP0
      SNAP1
      SNAP3
```

```
# ./gDBClone listdb -verbose
```

Database Name	Database Type	Database HomeLocation	Database Version	Database Role	Location/Parent
SNAP2	RACOneNode	/u01/app/oracle/product/12.1.0/dbhome_1	12.1.0.1.0	Snapshot	GOLD
GOLD	RAC	/u01/app/oracle/product/12.1.0/dbhome_1	12.1.0.1.0	Master	/cloudfs/.ACFS/snaps
SNAP3	RACOneNode	/u01/app/oracle/product/12.1.0/dbhome_1	12.1.0.1.0	Snapshot	SNAP1
SNAP0	SINGLE	/u01/app/oracle/product/12.1.0/dbhome_1	12.1.0.1.0	Snapshot	GOLD
SNAP1	SINGLE	/u01/app/oracle/product/12.1.0/dbhome_1	12.1.0.1.0	Snapshot	SNAP0

9. Create an encrypted SYS password file

```
# gDBClone syspwf -syspwf /opt/gDBClone/SYSpasswd_file
```

Example:

```
# gDBClone syspwf -syspwf /opt/gDBClone/SYSpasswd_file

Please enter the SYS User password :      ## Enter the remote SYS password
Please re-enter the SYS user password :    ## re-Enter the remote SYS password

SYS password file created as /opt/gDBClone/SYSpasswd_file

# cat /opt/gDBClone/SYSpasswd_file
701579ABE9D7E2C64A03332B12309E97
```

Case Studies

The gDBClone script provides flexible options and configurations to best fulfill the customer's requirements.

1. Managing a test & dev environment combined with Oracle Data Guard

Many customers deploy Oracle Data Guard as their disaster recovery solution. When you create the standby database on an ACFS file system, you will have simple options to create and manage a test & dev environment on the standby cluster. This makes better utilization of the standby resources while enabling a test and dev environment. The following diagram illustrates a test and dev environment that can easily be managed using the gDBClone commands:

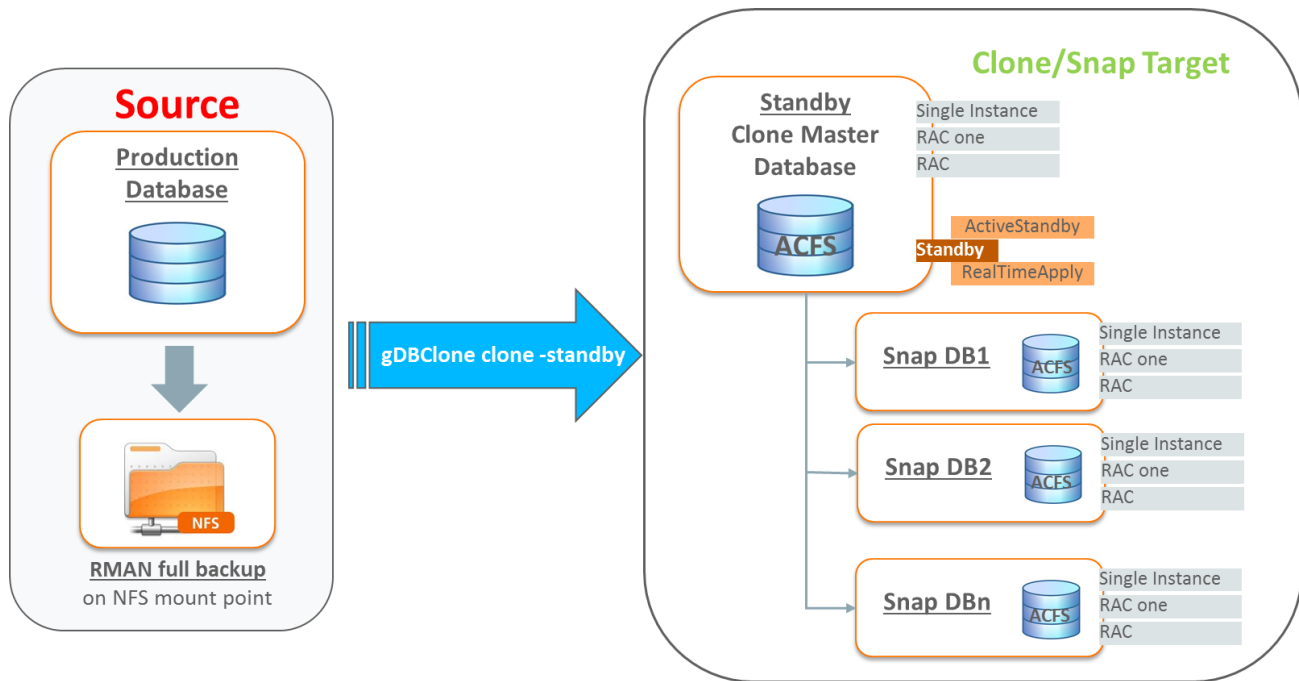


Figure 11 – Managing a test & dev environment combined with Oracle Data Guard

Using the 'gDBClone snap' function, you can either create multiple snapshots and provision for different purposes, or create a snapshot to preserve the point in time copy and create snaps of snaps to deploy identical copies of databases for test & dev as you can see from the diagram above. The advantage of this approach is that the master copy of the database (standby) is continuously refreshed by Data Guard allowing disaster recovery as well as refreshed data for testing.

Such scenario can be setup using gDBClone within few commands:

- 1) On target list the available Oracle Homes:

```
$ sudo /opt/gDBClone/gDBClone listhomes
Oracle Home Name      Home Location
-----
OraDb11204_home1      /u01/app/oracle/product/11.2.0.4/dbhome_1
OraDb12102_home1      /u01/app/oracle/product/12.1.0.2/dbhome_1
```

- 2) Create an encrypted SYS (source database) password file (optional):

```
$ sudo /opt/gDBClone/gDBClone syspwf -syspwf /opt/gDBClone/SYSpasswd_file

Please enter the SYS User password :    ## Enter the SYS source database password
Please re-enter the SYS user password : ##re-enter the SYS source DB password file

SYS password file created as /opt/gDBClone/SYSpasswd_file
```

Note: if you skip the syspwf file creation, the SYS source database password will be requested at command line

- 3) Clone the source database (ORCL) on target as Standby database (CLONE):

```
$ sudo /opt/gDBClone/gDBClone clone -sdbname ORCL \
                                     -sdbscan exadata316-scan \
                                     -tdbname CLONE \
                                     -tdbhome OraDb12102_home1 \
                                     -dataacfs /u02/app/oracle/oradata/datastore \
                                     -syspwf /opt/gDBClone/SYSpasswd_file \
                                     -standby
```

Output:

```
INFO: 2016-12-13 03:31:22: Please check the logfile '/opt/gDBClone/out/log/gDBClone_60775.log' for more details

MacroStep1 - Getting information and validating setup...
INFO: 2016-12-13 03:31:22: Validating environment
INFO: 2016-12-13 03:31:22: Checking superuser usage
INFO: 2016-12-13 03:31:22: Checking ping to host 'exadata316-scan'
INFO: 2016-12-13 03:31:22: Checking if target database name 'CLONE' is a valid name
INFO: 2016-12-13 03:31:22: Checking if target database home 'OraDb12102_home1' exists
INFO: 2016-12-13 03:31:23: Got Oracle Base from env variable: /u01/app/oracle
INFO: 2016-12-13 03:31:23: Checking if target database 'CLONE' exists
INFO: 2016-12-13 03:31:23: Checking 'CLONE' snapshot existence on '/u02/app/oracle/oradata/datastore'
INFO: 2016-12-13 03:31:23: Checking registered instance 'CLONE'
INFO: 2016-12-13 03:31:25: Checking listener on 'oda458:1521'
INFO: 2016-12-13 03:31:25: Checking source and target database version
INFO: 2016-12-13 03:31:28: Checking source log mode
INFO: 2016-12-13 03:31:29: Checking FLASHBACK mode
WARNING: 2016-12-13 03:31:29: Source database 'ORCL' is not in FLASHBACK mode
INFO: 2016-12-13 03:31:29: Checking LOG_ARCHIVE_DEST settings
INFO: 2016-12-13 03:31:30: Checking Flash Cache setting
INFO: 2016-12-13 03:31:30: Checking ACFS command options
INFO: 2016-12-13 03:31:30: Checking if '/u02/app/oracle/oradata/datastore' is an ACFS file system
SUCCESS: 2016-12-13 03:31:30: Environment validation complete
```

continue...

```
MacroStep2 - Setting up clone environment...
INFO: 2016-12-13 03:31:30: Creating local pfile
INFO: 2016-12-13 03:31:31: Creating local password file
INFO: 2016-12-13 03:31:31: Creating local Audit folder
INFO: 2016-12-13 03:31:31: Creating local auxiliary listener
INFO: 2016-12-13 03:31:31: Starting auxiliary listener
INFO: 2016-12-13 03:31:34: Sleeping 60 secs, please wait
INFO: 2016-12-13 03:32:34: Setting up ACFS storage
INFO: 2016-12-13 03:32:34: Creating dynamic scripts
INFO: 2016-12-13 03:32:37: Cloning to target ACFS from host 'exadata316-scan' as standby database
INFO: 2016-12-13 03:32:37: Creating RMAN script for spfile target to ACFS
INFO: 2016-12-13 03:32:37: Instantiating standby database
INFO: 2016-12-13 03:32:37: Enabling force logging
INFO: 2016-12-13 03:32:37: Getting standby logs on source database ORCL
INFO: 2016-12-13 03:32:38: Getting MAX redologs group on source database ORCL
INFO: 2016-12-13 03:32:38: Getting THREAD# and redolog size on source database ORCL
INFO: 2016-12-13 03:32:39: Creating standby logs on source database ORCL
SUCCESS: 2016-12-13 03:32:54: Environment setup complete

MacroStep3 - Cloning database 'ORCL'...
INFO: 2016-12-13 03:32:54: please wait (this can take a while depending on database size and/or network speed)
INFO: 2016-12-13 03:38:14: Moving spfile
INFO: 2016-12-13 03:38:37: Updating local dbs pfile/spfile
INFO: 2016-12-13 03:38:37: Register 'CLONE' database as cluster resource
INFO: 2016-12-13 03:38:39: Modifying DB instance
INFO: 2016-12-13 03:38:40: Setup ACFS dependency
INFO: 2016-12-13 03:38:42: Database 'CLONE' dependency to '/u02/app/oracle/oradata/datastore' done successfully

MacroStep5 - Standby setup...
INFO: 2016-12-13 03:39:00: Starting redo apply
INFO: 2016-12-13 03:39:06: Configuring primary database 'ORCL'
SUCCESS: 2016-12-13 03:39:07: Successfully created clone "CLONE" database as standby
INFO: 2016-12-13 03:39:07: Cleaning up the setup
```

4) Check the target database (CLONE) creation:

```
$ sudo /opt/gDBClone/gDBClone listdbs
```

Database Name	Database Type	Database Role	Master/Snapshot	Location/Parent
CLONE	SINGLE	PHYSICAL_STANDBY	Master	/u02/app/oracle/oradata/datastore/.ACFS/snaps/

5) Create a snapshot database SDB1 from the source standby CLONE database:

```
$ sudo /opt/gDBClone/gDBClone snap -sdbname CLONE \
                                     -tdbname SDB1 \
                                     -syspwf /opt/gDBClone/SYSpasswd_file
```

Note1: in order to get a snapshot database from a source standby database, the **Flashback is mandatory**, if needed, on source CLONE standby database execute:

```
alter database recover managed standby database cancel;
alter database flashback on;
alter database recover managed standby database using current logfile disconnect;
```

Note2: without “-tdbhome <Target Database Home Name>” parameter, the database home will be the same source database oracle home, in this example “OraDb12102_home1”

Output:

```
INFO: 2016-12-13 23:21:57: Please check the logfile '/opt/gDBClone/out/log/gDBClone_59587.log' for more details

MacroStep1 - Getting information and validating setup...
INFO: 2016-12-13 23:21:57: Validating environment...
INFO: 2016-12-13 23:21:57: Superuser usage check
INFO: 2016-12-13 23:21:57: Database 'CLONE' existence check
INFO: 2016-12-13 23:21:57: Database 'CLONE' running check
WARNING: 2016-12-13 23:22:00: ORACLE_BASE is not set
INFO: 2016-12-13 23:22:00: Got Oracle Base from adrci: /u01/app/oracle
INFO: 2016-12-13 23:22:00: Checking if target database name SDB1 is a valid name
INFO: 2016-12-13 23:22:00: Checking database 'CLONE' connectivity
INFO: 2016-12-13 23:22:13: Checking whether the database 'CLONE' is in ACFS snapshot
INFO: 2016-12-13 23:22:13: Checking source database 'CLONE' and target dbhome version
INFO: 2016-12-13 23:22:17: Checking if target database 'SDB1' exists
INFO: 2016-12-13 23:22:17: Checking registered instance 'SDB1'
INFO: 2016-12-13 23:22:24: Checking if SDB1 exists as snapshot in '/u02/app/oracle/oradata/datastore'
INFO: 2016-12-13 23:22:24: Checking if source database CLONE is snapable
INFO: 2016-12-13 23:22:30: ...Checking whether the database 'CLONE' is entirely on ACFS
INFO: 2016-12-13 23:22:36: ...Checking whether the database 'CLONE' is a primary/physical standby database.
INFO: 2016-12-13 23:22:40: ...Checking whether the Physical Standby database 'CLONE' is in MOUNT mode
INFO: 2016-12-13 23:22:46: ...Checking flashback on database 'CLONE'
INFO: 2016-12-13 23:22:52: ...Checking whether the database 'CLONE' is a CDB
INFO: 2016-12-13 23:23:06: ...Checking whether the database 'CLONE' is running in archive log mode
INFO: 2016-12-13 23:23:12: ...Checking if all the datafiles are available
SUCCESS: 2016-12-13 23:23:18: Environment validation complete

MacroStep2 - Getting database snapshot...
INFO: 2016-12-13 23:23:18: Cloning source database 'CLONE' using ACFS snapshot
INFO: 2016-12-13 23:23:43: Entering into SNAP database creation phase 1
INFO: 2016-12-13 23:23:51: Stopping redo apply for standby database 'CLONE'
INFO: 2016-12-13 23:23:58: Converting physical standby 'CLONE' to snapshot standby
INFO: 2016-12-13 23:24:23: ...Getting the snapshot of Database 'CLONE' at this time
INFO: 2016-12-13 23:24:23: ...Successfully took the snapshot 'SDB1' of database 'CLONE' on '/u02/app/oracle/oradata/datastore'
INFO: 2016-12-13 23:25:08: ...Converting physical standby 'CLONE' to snapshot standby
INFO: 2016-12-13 23:25:51: ...Starting redo apply for standby database 'CLONE'
INFO: 2016-12-13 23:26:03: ...Setting up storage for SNAP Database 'SDB1'
INFO: 2016-12-13 23:26:04: Entering into SNAP database creation phase 2
INFO: 2016-12-13 23:26:04: ...Creating controlfile for database 'SDB1'
INFO: 2016-12-13 23:26:54: ...Opening the database 'SDB1' with resetlogs
INFO: 2016-12-13 23:27:07: ...Setting the temporary tablespace for database 'SDB1'
INFO: 2016-12-13 23:27:34: ...Changing the Database ID
INFO: 2016-12-13 23:28:30: ...Creating spfile for SDB1
INFO: 2016-12-13 23:28:31: ...Creating password file for SDB1
INFO: 2016-12-13 23:28:31: Entering into SNAP database creation phase 3
INFO: 2016-12-13 23:28:54: ...Successfully started the database
INFO: 2016-12-13 23:29:02: ...Setting RMAN SNAPSHOT control file
INFO: 2016-12-13 23:29:38: Enabling flashback for database 'SDB1'
SUCCESS: 2016-12-13 23:29:53: Successfully created the database 'SDB1' from 'CLONE'
INFO: 2016-12-13 23:29:56: Cleaning up the setup
```

6) Check the target database (SDB1) creation:

```
$ sudo /opt/gDBClone/gDBClone listdbs
```

Database Name	Database Type	Database Role	Master/Snapshot	Location/Parent
CLONE	SINGLE	PHYSICAL_STANDBY	Master	/u02/app/oracle/oradata/datastore/.ACFS/snaps/
SDB1	SINGLE	PRIMARY	Snapshot	CLONE

Note: you could verify the parent relationship also doing:

```
$ sudo /opt/gDBClone/gDBClone listdbs -tree

Parent Child
-----
CLONE
      SDB1

-----
Note: a branch tree will be displayed only if the parent exists!
-----
```

- 7) Create a snapshot database SDB2 as RAC(Real Application Cluster) from the source standby CLONE database:

```
$ sudo /opt/gDBClone/gDBClone snap -sdbname CLONE \
                                     -tdbname SDB1 \
                                     -syspwf /opt/gDBClone/SYSpasswd_file \
                                     -racmod 2
```

Output:

```
INFO: 2016-12-13 23:35:18: Please check the logfile '/opt/gDBClone/out/log/gDBClone_82301.log' for more details

MacroStep1 - Getting information and validating setup...
INFO: 2016-12-13 23:35:18: Validating environment...
INFO: 2016-12-13 23:35:18: Superuser usage check
INFO: 2016-12-13 23:35:18: Database 'CLONE' existence check
INFO: 2016-12-13 23:35:18: Database 'CLONE' running check
WARNING: 2016-12-13 23:35:21: ORACLE_BASE is not set
INFO: 2016-12-13 23:35:21: Got Oracle Base from adrci: /u01/app/oracle
INFO: 2016-12-13 23:35:21: Checking if target database name SDB2 is a valid name
INFO: 2016-12-13 23:35:21: Checking database 'CLONE' connectivity
INFO: 2016-12-13 23:35:33: Checking whether the database 'CLONE' is in ACFS snapshot
INFO: 2016-12-13 23:35:33: Checking source database 'CLONE' and target dbhome version
INFO: 2016-12-13 23:35:38: Checking if target database 'SDB2' exists
INFO: 2016-12-13 23:35:38: Checking registered instance 'SDB2'
INFO: 2016-12-13 23:35:47: Checking if SDB2 exists as snapshot in '/u02/app/oracle/oradata/datastore'
INFO: 2016-12-13 23:35:47: Checking if source database CLONE is snapable
INFO: 2016-12-13 23:35:54: ...Checking whether the database 'CLONE' is entirely on ACFS
INFO: 2016-12-13 23:36:00: ...Checking whether the database 'CLONE' is a primary/physical standby database.
INFO: 2016-12-13 23:36:04: ...Checking whether the Physical Standby database 'CLONE' is in MOUNT mode
INFO: 2016-12-13 23:36:10: ...Checking flashback on database 'CLONE'
INFO: 2016-12-13 23:36:16: ...Checking whether the database 'CLONE' is a CDB
INFO: 2016-12-13 23:36:29: ...Checking whether the database 'CLONE' is running in archivelog mode
INFO: 2016-12-13 23:36:35: ...Checking if all the datafiles are available
SUCCESS: 2016-12-13 23:36:41: Environment validation complete
MacroStep2 - Getting database snapshot...
INFO: 2016-12-13 23:36:41: Cloning source database 'CLONE' using ACFS snapshot
INFO: 2016-12-13 23:37:06: Entering into SNAP database creation phase 1
INFO: 2016-12-13 23:37:13: Stopping redo apply for standby database 'CLONE'
INFO: 2016-12-13 23:37:22: Converting physical standby 'CLONE' to snapshot standby
INFO: 2016-12-13 23:37:47: ...Getting the snapshot of Database 'CLONE' at this time
INFO: 2016-12-13 23:37:47: ...Successfully took the snapshot 'SDB2' of database 'CLONE' on
'/u02/app/oracle/oradata/datastore'
INFO: 2016-12-13 23:38:32: ...Converting physical standby 'CLONE' to snapshot standby
INFO: 2016-12-13 23:39:14: ...Starting redo apply for standby database 'CLONE'
INFO: 2016-12-13 23:39:27: ...Setting up storage for SNAP Database 'SDB2'
INFO: 2016-12-13 23:39:27: Entering into SNAP database creation phase 2
INFO: 2016-12-13 23:39:27: ...Creating controlfile for database 'SDB2'
```

```

INFO: 2016-12-13 23:40:28: ...Opening the database 'SDB2' with resetlogs
INFO: 2016-12-13 23:40:39: ...Setting the temporary tablespace for database 'SDB2'
INFO: 2016-12-13 23:41:08: ...Changing the Database ID
INFO: 2016-12-13 23:42:02: ...Creating spfile for SDB2
INFO: 2016-12-13 23:42:03: ...Creating password file for SDB2
INFO: 2016-12-13 23:42:03: Entering into SNAP database creation phase 3
INFO: 2016-12-13 23:42:27: ...Successfully started the database
INFO: 2016-12-13 23:42:35: ...Setting RMAN SNAPSHOT control file
INFO: 2016-12-13 23:43:10: Enabling flashback for database 'SDB2'
SUCCESS: 2016-12-13 23:43:25: Successfully created the database 'SDB2' from 'CLONE'

```

```

MacroStep3 - Converting clone database 'SDB2' to cluster mode...
WARNING: 2016-12-13 23:43:29: Database 'SDB2' was already running
INFO: 2016-12-13 23:43:29: Database conversion started, it will take some time
SUCCESS: 2016-12-13 23:54:17: Database 'SDB2' converted to RAC successfully
INFO: 2016-12-13 23:54:17: Cleaning up the setup

```

8) Check the target database (SDB2) creation:

```

$ sudo /opt/gDBClone/gDBClone listdbs

```

Database Name	Database Type	Database Role	Master/Snapshot	Location/Parent
-----	-----	-----	-----	-----
CLONE	SINGLE	PHYSICAL_STANDBY	Master	/u02/app/oracle/oradata/datastore/.ACFS/snaps/
SDB1	SINGLE	PRIMARY	Snapshot	CLONE
SDB2	RAC	PRIMARY	Snapshot	CLONE

2. Creating database clone using RMAN backupsets

Databases may be cloned using RMAN backupsets from the production server to minimize the overhead. The backupset may be exported using the NFS network protocol as the source for the gDBClone command.

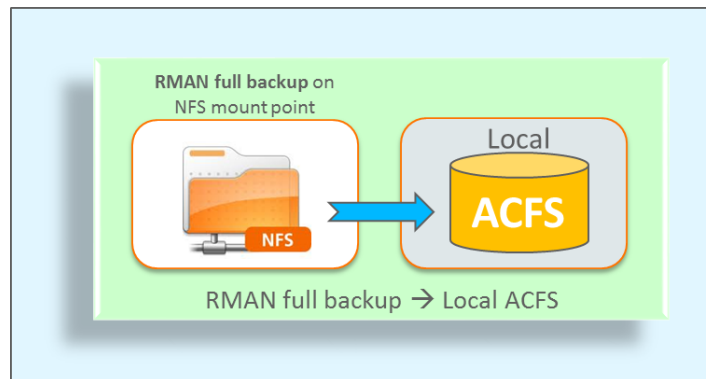


Figure 12 – Creating a database clone using RMAN backupset

The backup sets may be mounted via NFS on the test cluster. In this case, the NFS mount must be exported using the “insecure” export option on the source server for Oracle Database 12c tools to access the NFS mount properly. A step by step procedure is described later in this paper.

The following is an example of RMAN source database full backup command:

```
RMAN> RUN
{
  ALLOCATE CHANNEL disk1 DEVICE TYPE DISK FORMAT '/mnt/backup/ORCL/%U';
  BACKUP DATABASE PLUS ARCHIVELOG;
  BACKUP AS COPY CURRENT CONTROLFILE FORMAT '/mnt/backup/ORCL/control_%U';
  BACKUP SPFILE FORMAT '/mnt/backup/ORCL/spfile_%U';
}
```

The gDBClone command (example) is as following:

```
$ sudo /opt/gDBClone/gDBClone clone -sdbname ORCL \
    -sbckloc '/NFS/backup/ORCL' \
    -tdbname GOLD \
    -tdbhome OraDb12102_home1 \
    -dataacfs /u02/app/oracle/oradata/datastore \
    -syspwf /opt/gDBClone/SYSpasswd_file
```

Output:

```
INFO: 2016-12-14 03:30:36: Please check the logfile '/opt/gDBClone/out/log/gDBClone_70811.log' for more details

MacroStep1 - Getting information and validating setup...
INFO: 2016-12-14 03:30:36: Validating environment
INFO: 2016-12-14 03:30:36: Checking superuser usage
INFO: 2016-12-14 03:30:36: Checking source backup location /goldengate/tmp...
INFO: 2016-12-14 03:30:36: Checking if target database name 'GOLD' is a valid name
INFO: 2016-12-14 03:30:36: Checking if target database home 'OraDb12102_home1' exists
WARNING: 2016-12-14 03:30:37: ORACLE_BASE is not set
INFO: 2016-12-14 03:30:37: Got Oracle Base from adrci: /u01/app/oracle
INFO: 2016-12-14 03:30:37: Checking if target database 'GOLD' exists
```

```

INFO: 2016-12-14 03:30:37: Checking 'GOLD' snapshot existence on '/u02/app/oracle/oradata/datastore'
INFO: 2016-12-14 03:30:37: Checking registered instance 'GOLD'
INFO: 2016-12-14 03:30:43: Checking listener on 'slcac458:1521'
INFO: 2016-12-14 03:30:43: Checking ACFS command options
INFO: 2016-12-14 03:30:43: Checking if '/u02/app/oracle/oradata/datastore' is an ACFS file system
SUCCESS: 2016-12-14 03:30:43: Environment validation complete

MacroStep2 - Setting up clone environment...
INFO: 2016-12-14 03:30:43: Creating local pfile
INFO: 2016-12-14 03:30:43: Creating local password file
INFO: 2016-12-14 03:30:43: Creating local Audit folder
INFO: 2016-12-14 03:30:43: Creating local auxiliary listener
INFO: 2016-12-14 03:30:43: Starting auxiliary listener
INFO: 2016-12-14 03:30:43: Sleeping 60 secs, please wait
INFO: 2016-12-14 03:31:43: Setting up ACFS storage
INFO: 2016-12-14 03:31:43: Creating dynamic scripts
INFO: 2016-12-14 03:31:46: Cloning to target ACFS from backup location '/goldengate/tmp'
INFO: 2016-12-14 03:31:46: Creating RMAN script for spfile target to ACFS
INFO: 2016-12-14 03:31:46: Instantiating clone database
SUCCESS: 2016-12-14 03:31:46: Environment setup complete

MacroStep3 - Cloning database 'ORCL'...
INFO: 2016-12-14 03:31:46: please wait (this can take a while depending on database size and/or network speed)
INFO: 2016-12-14 03:35:19: Moving spfile
INFO: 2016-12-14 03:35:43: Updating local dbs pfile/spfile
INFO: 2016-12-14 03:35:44: Register 'GOLD' database as cluster resource
INFO: 2016-12-14 03:35:45: Checking database name
INFO: 2016-12-14 03:35:45: Modifying DB instance
INFO: 2016-12-14 03:35:46: Setup ACFS dependency
INFO: 2016-12-14 03:35:48: Database 'GOLD' dependency to '/u02/app/oracle/oradata/datastore' done successfully
SUCCESS: 2016-12-14 03:35:48: Successfully created clone database 'GOLD'
INFO: 2016-12-14 03:35:48: Cleaning up the setup

```

Note: if LOG_ARCHIVE_DEST_2 is set on source database backup, the clone will fail with error "ORA-16019: cannot use LOG_ARCHIVE_DEST_2 with LOG_ARCHIVE_DEST or LOG_ARCHIVE_DUPLEX_DEST". Unset LOG_ARCHIVE_DEST_2 on source database, get a full database backup and retry the clone with "-sbckloc"

Check for the new cloned database "GOLD":

```

$ sudo /opt/gDBClone/gDBClone listdbs

```

Database Name	Database Type	Database Role	Master/Snapshot	Location/Parent
GOLD	SINGLE	PRIMARY	Master	/u02/app/oracle/oradata/datastore/.ACFS/snaps/

3. Clone 11g Database from ASM to ACFS keeping the source running

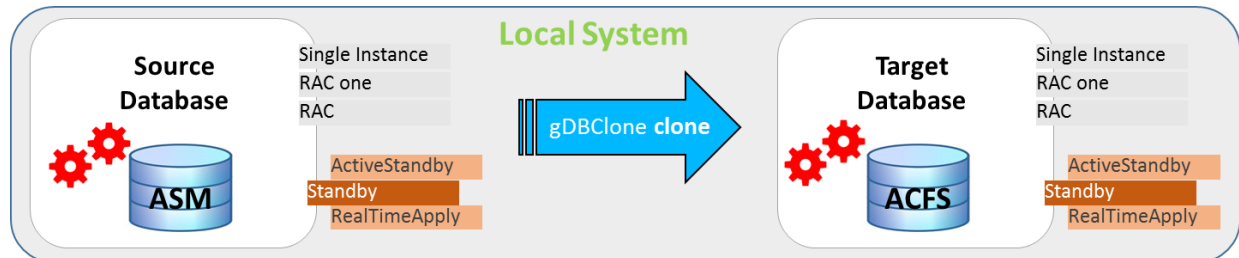


Figure 13 – Clone running 11g Database from ASM to ACFS

Prior to Oracle 12c, moving datafiles is always an offline task. Using gDBClone you can “move” (clone) a database from ASM to ACFS keeping it running. If you need to preserve the transactions during the cloning operation you may consider the Oracle GoldenGate usage.

```
$ sudo /opt/gDBClone/gDBClone clone -sdbname ORCL \  
    -sdbscan exadata316-scan \  
    -tdbname CLONE \  
    -tdbhome OraDb11204_home1 \  
    -dataacfs /u02/app/oracle/oradata/datastore  
[ -redoacfs <acfs mount point> ]  
[ -recoacfs <acfs mount point> ]  
[ -standby [-pmode maxperf|maxavail|maxprot]  
           [-activedg] [-rtapply] ]  
[ -racmod <db type> ]
```

4. Create a RAC snapshot database from a GOLD clone running database

Once you got a clone gold image (from a backupset or from a running database) of your production database you can now, using the 'gDBClone snap' function, to create multiple snapshots and provision for different purposes. The clone can be SI, RAC OneNode or RAC and the snapshot database can be SI, RAC OneNode or RAC ("–racmod"). You could convert the snapshot database later also, using the "convert" option.

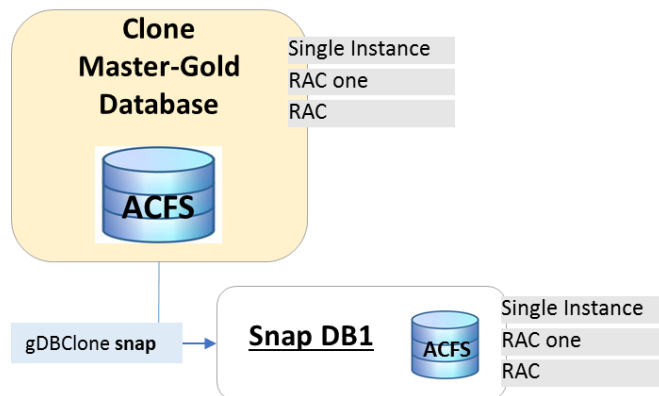


Figure 13 – Create a RAC snapshot database from a GOLD clone running database

1) Create a snapshot database SDB from the source standby CLONE database:

```
$ sudo /opt/gDBClone/gDBClone snap -sdbname GOLD \  
-tdbname SDB \  
-racmod 2
```

Output:

```
INFO: 2016-12-15 00:41:00: Please check the logfile '/opt/gDBClone/out/log/gDBClone_22829.log' for more details  
  
MacroStep1 - Getting information and validating setup...  
INFO: 2016-12-15 00:41:00: Validating environment...  
INFO: 2016-12-15 00:41:00: Superuser usage check  
INFO: 2016-12-15 00:41:00: Database 'GOLD' existence check  
INFO: 2016-12-15 00:41:00: Database 'GOLD' running check  
WARNING: 2016-12-15 00:41:03: ORACLE_BASE is not set  
INFO: 2016-12-15 00:41:03: Got Oracle Base from orabase  
INFO: 2016-12-15 00:41:03: Checking if target database name SNAP is a valid name  
INFO: 2016-12-15 00:41:03: Checking database 'GOLD' connectivity  
INFO: 2016-12-15 00:41:16: Checking whether the database 'GOLD' is in ACFS snapshot  
INFO: 2016-12-15 00:41:16: Checking source database 'GOLD' and target dbhome version  
INFO: 2016-12-15 00:41:21: Checking if target database 'SNAP' exists  
INFO: 2016-12-15 00:41:21: Checking registered instance 'SNAP'  
INFO: 2016-12-15 00:41:31: Checking if SNAP exists as snapshot in '/u02/app/oracle/oradata/datastore'  
INFO: 2016-12-15 00:41:31: Checking if source database GOLD is snapable  
INFO: 2016-12-15 00:41:37: ...Checking whether the database 'GOLD' is entirely on ACFS  
INFO: 2016-12-15 00:41:44: ...Checking whether the database 'GOLD' is a primary/physical standby database.  
INFO: 2016-12-15 00:41:47: ...Checking whether the database 'GOLD' is in READ WRITE mode  
INFO: 2016-12-15 00:41:53: ...Checking whether the database 'GOLD' is a CDB  
INFO: 2016-12-15 00:42:06: ...Checking whether the database 'GOLD' is running as backup mode  
INFO: 2016-12-15 00:42:12: ...Checking whether the database 'GOLD' is running in archivelog mode  
INFO: 2016-12-15 00:42:18: ...Checking if all the datafiles are available  
INFO: 2016-12-15 00:42:24: ...Checking if there are OFFLINE datafiles  
SUCCESS: 2016-12-15 00:42:30: Environment validation complete
```

```

MacroStep2 - Getting database snapshot...
INFO: 2016-12-15 00:42:30: Cloning source database 'GOLD' using ACFS snapshot
INFO: 2016-12-15 00:42:56: Entering into SNAP database creation phase 1
INFO: 2016-12-15 00:42:56: ...Getting required information to get consistent database snapshot
WARNING: 2016-12-15 00:43:09: Do not perform any Structural change to database 'GOLD' till SNAP database
'SNAP' is created
INFO: 2016-12-15 00:43:22: ...Getting the snapshot of Database 'GOLD' at this time
INFO: 2016-12-15 00:43:22: ...Successfully took the snapshot 'SNAP' of database 'GOLD' on
'/u02/app/oracle/oradata/datastore'
INFO: 2016-12-15 00:43:29: ...Setting up storage for SNAP Database 'SNAP'
INFO: 2016-12-15 00:44:00: Entering into SNAP database creation phase 2
INFO: 2016-12-15 00:44:00: ...Creating controlfile for database 'SNAP'
INFO: 2016-12-15 00:45:02: ...Recovering the database: SNAP, snapshot time : '2016-12-15:00:43:22',
until 'change:698612'
INFO: 2016-12-15 00:45:03: ...Opening the database with resetlogs
INFO: 2016-12-15 00:45:17: ...Setting the temporary tablespace for database 'SNAP'
INFO: 2016-12-15 00:45:46: ...Changing the Database ID
INFO: 2016-12-15 00:46:41: ...Creating spfile for SNAP
INFO: 2016-12-15 00:46:42: ...Creating password file for SNAP
INFO: 2016-12-15 00:46:42: Entering into SNAP database creation phase 3
INFO: 2016-12-15 00:47:09: ...Successfully started the database
INFO: 2016-12-15 00:47:16: ...Setting RMAN SNAPSHOT control file
INFO: 2016-12-15 00:47:25: ...Disabling the external references in the database 'SNAP' inherited from
'GOLD'
-----
Run on the database 'SNAP' the SQL script:
'/u01/app/oracle/product/12.1.0.2/dbhome_1/enable_external_refs_SNAP_D0yo.sql'
to enable these external references.
Also need to restart the database after running the SQL script.
-----
SUCCESS: 2016-12-15 00:48:35: Successfully created the database 'SNAP' from 'GOLD'

MacroStep3 - Converting clone database 'SNAP' to cluster mode...
WARNING: 2016-12-15 00:48:39: Database 'SNAP' was already running
INFO: 2016-12-15 00:48:39: Database conversion started, it will take some time
SUCCESS: 2016-12-15 00:56:40: Database 'SNAP' converted to RAC successfully
INFO: 2016-12-15 00:56:40: Cleaning up the setup

```

2) Check the new database created:

```
# gDBClone listdbs
```

Database Name	Database Type	Database Role	Master/Snapshot	Location/Parent
-----	-----	-----	-----	-----
GOLD	SINGLE	PRIMARY	Master	/u02/app/oracle/oradata/datastore/.ACFS/snaps/
SNAP	RACOneNode	PRIMARY	Snapshot	GOLD

5. Create a snapshot RAC database from a standby database

Having a running standby database, using gDBClone is possible to get a snapshot of it. The source standby can be SI, RAC OneNode or RAC and the snapshot database can be SI, RAC OneNode or RAC (“-racmod”). You could convert the snapshot database also later using the “convert” option.

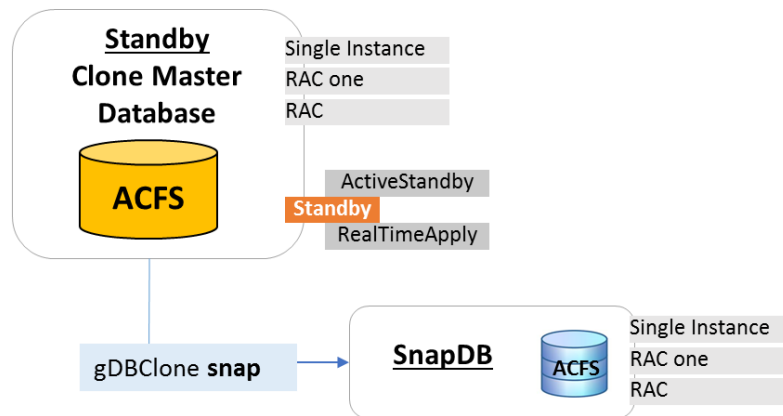


Figure 14 – Create a snapshot RAC database from a standby database

1. Check the source Standby database:

```
$ sudo /opt/gDBClone/gDBClone listdbs
```

Database Name	Database Type	Database Role	Master/Snapshot	Location/Parent
STDBY	SINGLE	PHYSICAL_STANDBY	Master	/u02/app/oracle/oradata/datastore/.ACFS/snaps/

2. Create a snapshot database SNAP from the source standby STDBY database:

```
$ sudo /opt/gDBClone/gDBClone snap -sdbname STDBY \
-tdbname SNAP \
-racmod 2
```

Output:

```
INFO: 2016-12-15 04:53:16: Please check the logfile '/opt/gDBClone/out/log/gDBClone_31709.log' for more details

MacroStep1 - Getting information and validating setup...
INFO: 2016-12-15 04:53:16: Validating environment...
INFO: 2016-12-15 04:53:16: Superuser usage check
INFO: 2016-12-15 04:53:16: Database 'STDBY'existence check
INFO: 2016-12-15 04:53:16: Database 'STDBY'running check
WARNING: 2016-12-15 04:53:19: ORACLE_BASE is not set
INFO: 2016-12-15 04:53:19: Got Oracle Base from orabase
INFO: 2016-12-15 04:53:19: Checking if target database name SNAP is a valid name
INFO: 2016-12-15 04:53:19: Checking database 'STDBY' connectivity
INFO: 2016-12-15 04:53:31: Checking whether the database 'STDBY' is in ACFS snapshot
INFO: 2016-12-15 04:53:31: Checking source database 'STDBY' and target dbhome version
INFO: 2016-12-15 04:53:36: Checking if target database 'SNAP' exists
INFO: 2016-12-15 04:53:36: Checking registered instance 'SNAP'
INFO: 2016-12-15 04:53:46: Checking if SNAP exists as snapshot in '/u02/app/oracle/oradata/datastore'
INFO: 2016-12-15 04:53:46: Checking if source database 'STDBY' is snapable
```

```

INFO: 2016-12-15 04:53:52: ...Checking whether the database 'STDBY' is entirely on ACFS
INFO: 2016-12-15 04:53:59: ...Checking whether the database 'STDBY' is a primary/physical standby
database.
INFO: 2016-12-15 04:54:02: ...Checking whether the Physical Standby database 'STDBY' is in MOUNT mode
INFO: 2016-12-15 04:54:08: ...Checking flashback on database 'STDBY'
INFO: 2016-12-15 04:54:14: ...Checking whether the database 'STDBY' is a CDB
INFO: 2016-12-15 04:54:27: ...Checking whether the database 'STDBY' is running in archivelog mode
INFO: 2016-12-15 04:54:33: ...Checking if all the datafiles are available
SUCCESS: 2016-12-15 04:54:39: Environment validation complete

```

MacroStep2 - Getting database snapshot...

```

INFO: 2016-12-15 04:54:39: Cloning source database 'STDBY' using ACFS snapshot
INFO: 2016-12-15 04:55:05: Entering into SNAP database creation phase 1
INFO: 2016-12-15 04:55:12: Stopping redo apply for standby database 'STDBY'
INFO: 2016-12-15 04:55:20: Converting physical standby 'STDBY' to snapshot standby
INFO: 2016-12-15 04:55:44: ...Getting the snapshot of Database 'STDBY' at this time
INFO: 2016-12-15 04:55:45: ...Successfully took the snapshot 'SNAP' of database 'STDBY' on
'/u02/app/oracle/oradata/datastore'
INFO: 2016-12-15 04:56:29: ...Converting physical standby 'STDBY' to snapshot standby
INFO: 2016-12-15 04:57:12: ...Starting redo apply for standby database 'STDBY'
INFO: 2016-12-15 04:57:24: ...Setting up storage for SNAP Database 'SNAP'
INFO: 2016-12-15 04:57:25: Entering into SNAP database creation phase 2
INFO: 2016-12-15 04:57:25: ...Creating controlfile for database 'SNAP'
INFO: 2016-12-15 04:58:19: ...Opening the database 'SNAP' with resetlogs
INFO: 2016-12-15 04:58:31: ...Setting the temporary tablespace for database 'SNAP'
INFO: 2016-12-15 04:58:59: ...Changing the Database ID
INFO: 2016-12-15 04:59:53: ...Creating spfile for SNAP
INFO: 2016-12-15 04:59:54: ...Creating password file for SNAP
INFO: 2016-12-15 04:59:54: Entering into SNAP database creation phase 3
INFO: 2016-12-15 05:00:25: ...Successfully started the database
INFO: 2016-12-15 05:00:32: ...Setting RMAN SNAPSHOT control file
INFO: 2016-12-15 05:01:08: Enabling flashback for database 'SNAP'
SUCCESS: 2016-12-15 05:01:22: Successfully created the database 'SNAP' from 'STDBY'

```

MacroStep3 - Converting clone database 'SNAP' to cluster mode...

```

WARNING: 2016-12-15 05:01:26: Database 'SNAP' was already running
INFO: 2016-12-15 05:01:26: Database conversion started, it will take some time
SUCCESS: 2016-12-15 05:12:18: Database 'SNAP' converted to RAC successfully
INFO: 2016-12-15 05:12:18: Cleaning up the setup

```

3) Check the new database created:

```
$ sudo /opt/gDBClone/gDBClone listdbs
```

Database Name	Database Type	Database Role	Master/Snapshot	Location/Parent
STDBY	SINGLE	PHYSICAL_STANDBY	Master	/u02/app/oracle/oradata/datastore/.ACFS/snaps/
SNAP	RAC	PRIMARY	Snapshot	STDBY

```
$ sudo /opt/gDBClone/gDBClone listdbs -tree
```

```

Parent  Child
-----  -----
STDBY
      SNAP

```

6. Clone a database from RMAN full backup to ACFS as standby Database

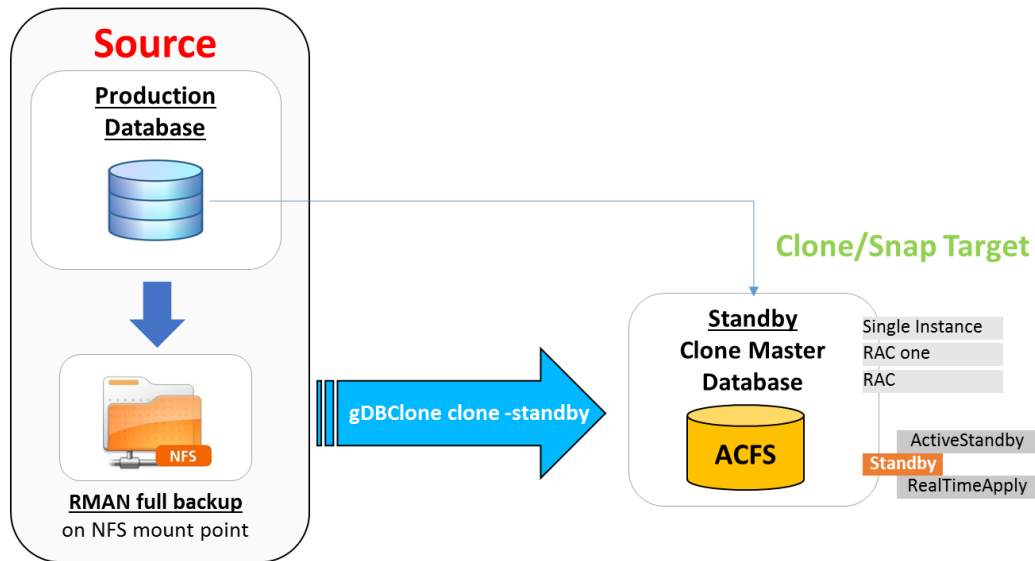


Figure 15 – Clone a database from RMAN full backup to ACFS as standby database

Create a snap database NOWK from RMAN backupset and make it as standby of source database
ORCL:

```
$ sudo /opt/gDBClone/gDBClone clone -sdbname ORCL \  
-sbckloc '/mnt/ORCL/bck' \  
-sdbscan exa316c1n1-scan \  
-tdbname NOWK \  
-tdbhome OraDb12102_home1 \  
-dataacfs /u02/app/oracle/oradata/datastore \  
-standby
```

Output:

```
INFO: 2016-12-16 00:10:47: Please check the logfile '/opt/gDBClone/out/log/gDBClone_59610.log' for more  
details  
  
MacroStep1 - Getting information and validating setup...  
INFO: 2016-12-16 00:10:47: Validating environment  
INFO: 2016-12-16 00:10:47: Checking superuser usage  
INFO: 2016-12-16 00:10:47: Checking ping to host exa316c1n1-scan'  
INFO: 2016-12-16 00:10:47: Checking if target database name 'NOWK' is a valid name  
INFO: 2016-12-16 00:10:47: Checking if target database home 'OraDb12102_home1' exists  
WARNING: 2016-12-16 00:10:47: ORACLE_BASE is not set  
INFO: 2016-12-16 00:10:47: Got Oracle Base from orabase  
INFO: 2016-12-16 00:10:47: Checking if target database 'NOWK' exists  
INFO: 2016-12-16 00:10:48: Checking 'NOWK' snapshot existence on '/u02/app/oracle/oradata/datastore'  
INFO: 2016-12-16 00:10:48: Checking registered instance 'NOWK'  
INFO: 2016-12-16 00:10:57: Checking listener on 'slcac458:1521'  
INFO: 2016-12-16 00:10:57: Checking FLASHBACK mode  
WARNING: 2016-12-16 00:10:58: Source database 'ORCL' is not in FLASHBACK mode  
INFO: 2016-12-16 00:10:58: Checking LOG_ARCHIVE_DEST settings  
INFO: 2016-12-16 00:10:58: Checking ACFS command options  
INFO: 2016-12-16 00:10:58: Checking if '/u02/app/oracle/oradata/datastore' is an ACFS file system  
SUCCESS: 2016-12-16 00:10:58: Environment validation complete
```

continue...

```
INFO: 2016-12-16 00:10:58: Creating local pfile
INFO: 2016-12-16 00:10:58: Creating local password file
INFO: 2016-12-16 00:10:58: Creating local Audit folder
INFO: 2016-12-16 00:10:58: Creating local auxiliary listener
INFO: 2016-12-16 00:10:58: Starting auxiliary listener
INFO: 2016-12-16 00:10:58: Sleeping 60 secs, please wait
INFO: 2016-12-16 00:11:58: Setting up ACFS storage
INFO: 2016-12-16 00:11:59: Creating dynamic scripts
INFO: 2016-12-16 00:12:01: Cloning to target ACFS from backup location '/mnt/ORCL/bck' as standby
database
INFO: 2016-12-16 00:12:01: Creating RMAN script for spfile target to ACFS
INFO: 2016-12-16 00:12:01: Instantiating standby database
INFO: 2016-12-16 00:12:01: Enabling force logging
INFO: 2016-12-16 00:12:01: Getting standby logs on source database ORCL
INFO: 2016-12-16 00:12:02: Standby logs exist on source database ORCL
SUCCESS: 2016-12-16 00:12:02: Environment setup complete

MacroStep3 - Cloning database 'ORCL'...
INFO: 2016-12-16 00:12:02: please wait (this can take a while depending on database size and/or network
speed)
INFO: 2016-12-16 00:14:32: Moving spfile
INFO: 2016-12-16 00:14:56: Updating local dbs pfile/spfile
INFO: 2016-12-16 00:14:56: Register 'NOWK' database as cluster resource
INFO: 2016-12-16 00:14:58: Modifying DB instance
INFO: 2016-12-16 00:14:59: Setup ACFS dependency
INFO: 2016-12-16 00:15:01: Database 'NOWK' dependency to '/u02/app/oracle/oradata/datastore' done
successfully

MacroStep5 - Standby setup...
INFO: 2016-12-16 00:15:16: Starting redo apply
INFO: 2016-12-16 00:15:23: Configuring primary database 'ORCL'
SUCCESS: 2016-12-16 00:15:23: Successfully created clone "NOWK" database as standby
INFO: 2016-12-16 00:15:23: Cleaning up the setup
```

Check the standby creation:

```
$ sudo /opt/gDBClone/gDBClone listdbs
```

Database Name	Database Type	Database Role	Master/Snapshot	Location/Parent
NOWK	SINGLE	PHYSICAL_STANDBY	Master	/u02/app/oracle/oradata/datastore/.ACFS/snaps/

7. Database upgrade using Transient Logical Standby(TLS)

Due to “*Hot Database Snapshot as Standby*” capability, without impact over the source production database and "without" storage duplication, leveraging on ACFS snapshot “copy&write” feature, gDBClone is making a snapshot of a running database as standby. Having a physical standby from a running production database without downtime is the key to make a database upgrade using the Transient Logical Standby (TLS).

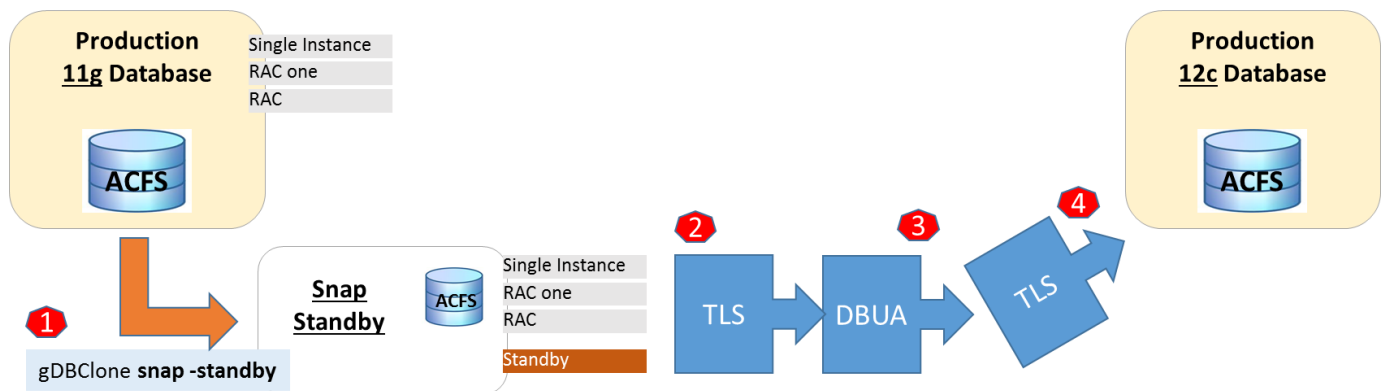


Figure 16 – Database upgrade using Transient Logical Standby (TLS)

The steps could be as following:

1. gDBClone snap as standby --> no downtime, minimal storage duplication (depend on source database production activity)
2. physru (TLS tool)
3. DBUA
4. physru (TLS tool)

Note: physru is a script to minimize downtime and simplify a database rolling upgrade using a physical standby database. A 'transient logical' herein refers to the physical standby database that has been temporarily converted to a transient logical standby database for the purpose of executing the upgrade. Also see the [MAA Best Practice Paper: Database Rolling Upgrades Made Easy](#), for additional information describing this process and [Oracle11g Data Guard: Database Rolling Upgrade Shell Script \(Doc ID 949322.1\)](#)

8. Clone a database encrypted with Transparent Data Encryption (TDE)

In this scenario we want use gDBClone to clone/snap database encrypted with the Transparent Data Encryption (TDE). Before to use gDBClone, you must manually copy the keystore to the duplicate database. If the keystore is not an auto login (SSO) keystore, then you must convert it to an auto login keystore at the duplicate database.

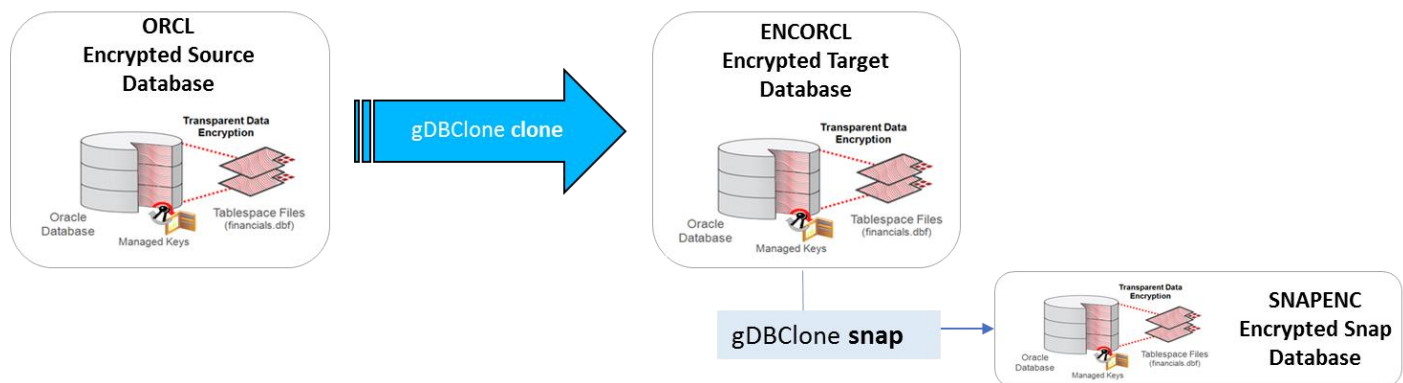


Figura 17 - gDBClone and encrypted database

In this example we are considering ORCL as source encrypted database, ENCORCL the cloned database and SNAPENC the snapshot encrypted database

1. Copy the wallet file (ewallet.p12) from source database server to target clone database server.

You can check the wallet file location on source database from sqlnet.ora file of the source database ORACLE_HOME

```
$ mkdir -p /u01/app/oracle/admin/ENCORCL/tde_wallet
$ scp oracle@prod-serv:/u01/app/oracle/admin/ORCL/tde_wallet/ewallet.p12
/u01/app/oracle/admin/ENCORCL/tde_wallet/
```

2. Modify sqlnet.ora file in target clone database ORACLE_HOME to reflect the location of the wallet file:

```
ENCRYPTION_WALLET_LOCATION =
  (SOURCE = (METHOD = FILE)
    (METHOD_DATA =
      (DIRECTORY=/u01/app/oracle/admin/ENCORCL/tde_wallet)
    )
  )
```

3. Invoke orapki utility on the target clone database server to make the wallet auto-login:

```
$ orapki wallet create -wallet /u01/app/oracle/admin/ENCORCL/tde_wallet \
  -pwd "Welcome_1" \
  -auto_login
```

4. You can now use gDBClone to clone the source encrypted database

```
$ sudo /opt/gDBClone/gDBClone clone -sdbname ORCL \  
-sdbscan exadata316-scan \  
-tdbname ENCORCL \  
-tdbhome OraDb12102_home1 \  
-dataacfs /u02/app/oracle/oradata/datastore
```

If you need also a database snapshot:

```
$ sudo /opt/gDBClone/gDBClone snap -sdbname ENCORCL -tdbname SNAPENC
```

The required wallet will be created under the expected location:

"/u01/app/oracle/admin/SNAPENC/tde_wallet"

Output:

```
INFO: 2016-12-22 08:06:38: Please check the logfile '/opt/gDBClone/out/log/gDBClone_46802.log' for more details  
  
MacroStep1 - Getting information and validating setup...  
INFO: 2016-12-22 08:06:38: Validating environment...  
INFO: 2016-12-22 08:06:38: Superuser usage check  
INFO: 2016-12-22 08:06:38: Database 'ENCORCL' existence check  
INFO: 2016-12-22 08:06:38: Database 'ENCORCL' running check  
WARNING: 2016-12-22 08:06:41: ORACLE_BASE is not set  
INFO: 2016-12-22 08:06:41: Got Oracle Base from orabase  
INFO: 2016-12-22 08:06:41: Checking if target database name SNAPENC is a valid name  
INFO: 2016-12-22 08:06:41: Checking database 'ENCORCL' connectivity  
INFO: 2016-12-22 08:06:53: Checking whether the database 'ENCORCL' is in ACFS snapshot  
INFO: 2016-12-22 08:06:53: Checking source database 'ENCORCL' and target dbhome version  
INFO: 2016-12-22 08:06:58: Checking if target database 'SNAPENC' exists  
INFO: 2016-12-22 08:06:58: Checking registered instance 'SNAPENC'  
INFO: 2016-12-22 08:07:08: Checking if SNAPENC exists as snapshot in '/u02/app/oracle/oradata/datastore'  
INFO: 2016-12-22 08:07:08: Checking if source database ENCORCL is snapable  
INFO: 2016-12-22 08:07:33: Transparent Data Encryption is enabled on 'ENCORCL'  
  
Please enter the 'WALLET' User password for the database:  
Please re-enter the 'WALLET' user password for the database:  
INFO: 2016-12-22 08:07:34: ...Checking whether the database 'ENCORCL' is entirely on ACFS  
INFO: 2016-12-22 08:07:41: ...Checking whether the database 'ENCORCL' is a primary/physical standby database.  
INFO: 2016-12-22 08:07:44: ...Checking whether the database 'ENCORCL' is in READ WRITE mode  
INFO: 2016-12-22 08:07:51: ...Checking whether the database 'ENCORCL' is a CDB  
INFO: 2016-12-22 08:08:03: ...Checking whether the database 'ENCORCL' is running as backup mode  
INFO: 2016-12-22 08:08:10: ...Checking whether the database 'ENCORCL' is running in archivelog mode  
INFO: 2016-12-22 08:08:16: ...Checking if all the datafiles are available  
INFO: 2016-12-22 08:08:22: ...Checking if there are OFFLINE datafiles  
SUCCESS: 2016-12-22 08:08:28: Environment validation complete  
  
MacroStep2 - Getting database snapshot...  
INFO: 2016-12-22 08:08:28: Cloning source database 'ENCORCL' using ACFS snapshot  
INFO: 2016-12-22 08:08:54: Entering into SNAP database creation phase 1  
INFO: 2016-12-22 08:08:54: ...Getting required information to get consistent database snapshot  
WARNING: 2016-12-22 08:09:06: Do not perform any Structural change to database 'ENCORCL' till SNAP database 'SNAPENC' is created  
INFO: 2016-12-22 08:09:20: ...Getting the snapshot of Database 'ENCORCL' at this time  
INFO: 2016-12-22 08:09:20: ...Successfully took the snapshot 'SNAPENC' of database 'ENCORCL' on '/u02/app/oracle/oradata/datastore'  
INFO: 2016-12-22 08:09:27: ...Setting up storage for SNAP Database 'SNAPENC'
```

Continue...

```
INFO: 2016-12-22 08:09:56: Entering into SNAP database creation phase 2
INFO: 2016-12-22 08:09:56: ...Creating controlfile for database 'SNAPENC'
INFO: 2016-12-22 08:10:57: ...Recovering the database: SNAPENC, snapshot time : '2016-12-22:08:09:20',
until 'change:2566598'
INFO: 2016-12-22 08:10:57: ...Opening the database with resetlogs
INFO: 2016-12-22 08:11:12: ...Setting the temporary tablespace for database 'SNAPENC'
INFO: 2016-12-22 08:11:40: ...Changing the Database ID
INFO: 2016-12-22 08:12:35: ...Creating spfile for SNAPENC
INFO: 2016-12-22 08:12:36: ...Creating password file for SNAPENC
INFO: 2016-12-22 08:12:36: Entering into SNAP database creation phase 3
INFO: 2016-12-22 08:13:01: ...Successfully started the database
INFO: 2016-12-22 08:13:09: ...Setting RMAN SNAPSHOT control file
INFO: 2016-12-22 08:13:18: ...Disabling the external references in the database 'SNAPENC' inherited from
'ENCORCL'
-----
Run on the database 'SNAPENC' the SQL script:
'/u01/app/oracle/product/12.1.0.2/dbhome_1/enable_external_refs_SNAPENC_i7tZ.sql'
to enable these external references.
Also need to restart the database after running the SQL script.
-----
SUCCESS: 2016-12-22 08:14:30: Successfully created the database 'SNAPENC' from 'ENCORCL'
INFO: 2016-12-22 08:14:32: Cleaning up the setup
```

You can avoid the “WALLET” password request if “/opt/gDBClone/WALLETpasswd_file” is present. You can create such file with the wallet password issuing:

```
$ sudo /opt/gDBClone/gDBClone syspwf -syspwf WALLETpasswd_file
```


9. Using gDBClone in Oracle Public Cloud (RACDBaaS)

In this scenario we want use gDBClone to clone/snap database running on Oracle Public Cloud (OPC) RACDBaaS.

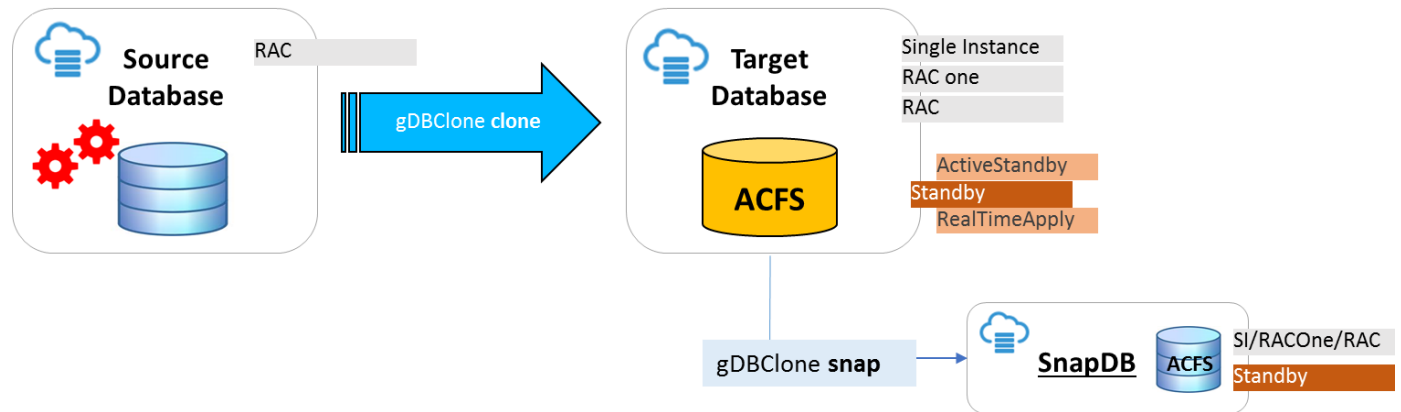


Figure 18 - gDBClone in Oracle Public Cloud (OPC) – RACDBaaS

Note as a snapshot of parent snapshot (used on getting a database snapshot: “gDBClone snap”) is possible when the Oracle ASM Dynamic Volume Manager (Oracle ADVM) compatibility attribute for the disk group is not less than 12.1. Before to use gDBClone on OPC you should verify the `compatible.advm` for DATA diskgroup and in case make it at least at 12.1

```
[grid@rcrac1 ~]$ asmcmd lsattr -G DATA -l |grep compatible
compatible.advm          11.2.0.4
compatible.asm           12.1.0.0.0
compatible.rdbms         11.2.0.4
```

As “grid” user setup the `compatible.advm` to 12.1.0.0.0 as following:

```
[grid@rcrac1 ~]$ asmcmd setattr -G DATA compatible.advm 12.1.0.0.0
```

Create a clone database GOLD from the source ORCL database (“-opc” is a required command ption):

```
$ sudo /opt/gDBClone/gDBClone clone -sdbname orcl.gboracle60892.oraclecloud.internal
-sdbscan rcrac-scan-int
-tdbname GOLD
-tdbhome OraDB11204_home1
-dataacfs /u02
-redoacfs /u04
-recoacfs /u03
-opc
```

Output:

```
MacroStep1 - Getting information and validating setup...
INFO: 2016-12-22 10:21:47: Validating environment
INFO: 2016-12-22 10:21:47: Checking superuser usage
INFO: 2016-12-22 10:21:47: Checking if target database name 'GOLD' is a valid name
INFO: 2016-12-22 10:21:47: Checking if target database home 'OraDB11204_home1' exists
WARNING: 2016-12-22 10:21:48: ORACLE_BASE is not set
INFO: 2016-12-22 10:21:48: Got Oracle Base from orabase
INFO: 2016-12-22 10:21:48: Checking if target database 'GOLD' exists
INFO: 2016-12-22 10:21:48: Checking 'GOLD' snapshot existence on '/u02'
INFO: 2016-12-22 10:21:48: Checking registered instance 'GOLD'
INFO: 2016-12-22 10:21:50: Checking listener on 'rcrac1:1521'
INFO: 2016-12-22 10:21:50: Checking source and target database version
INFO: 2016-12-22 10:21:52: Checking source log mode
INFO: 2016-12-22 10:21:52: Checking Flash Cache setting
INFO: 2016-12-22 10:21:53: Checking ACFS command options
INFO: 2016-12-22 10:21:53: Checking if '/u02' is an ACFS file system
INFO: 2016-12-22 10:21:53: Checking if '/u04' is an ACFS file system
INFO: 2016-12-22 10:21:53: Checking if '/u03' is an ACFS file system
SUCCESS: 2016-12-22 10:21:53: Environment validation complete

MacroStep2 - Setting up clone environment...
INFO: 2016-12-22 10:21:53: Creating local pfile
INFO: 2016-12-22 10:21:53: Creating local password file
INFO: 2016-12-22 10:21:53: Creating local Audit folder
INFO: 2016-12-22 10:21:53: Creating local auxiliary listener
INFO: 2016-12-22 10:21:53: Starting auxiliary listener
INFO: 2016-12-22 10:21:53: Sleeping 60 secs, please wait
INFO: 2016-12-22 10:22:53: Setting up ACFS storage
INFO: 2016-12-22 10:22:53: Creating dynamic scripts
INFO: 2016-12-22 10:22:55: Cloning to target ACFS from host 'rcrac-scan-int'
INFO: 2016-12-22 10:22:55: Creating RMAN script for spfile target to ACFS
INFO: 2016-12-22 10:22:56: Instantiating clone database
SUCCESS: 2016-12-22 10:22:56: Environment setup complete

MacroStep3 - Cloning database 'orcl.gboracle60892.oraclecloud.internal'...
INFO: 2016-12-22 10:22:56: please wait (this can take a while depending on database size and/or network speed)
INFO: 2016-12-22 10:26:20: Moving spfile
INFO: 2016-12-22 10:26:34: Updating local dbs pfile/spfile
INFO: 2016-12-22 10:26:34: Register 'GOLD' database as cluster resource
INFO: 2016-12-22 10:26:37: Checking database name
INFO: 2016-12-22 10:26:37: Modifying DB instance
INFO: 2016-12-22 10:26:38: Setup ACFS dependency
INFO: 2016-12-22 10:26:41: Database 'GOLD' dependency to '/u02,/u04,/u03' done successfully
SUCCESS: 2016-12-22 10:26:41: Successfully created clone database 'GOLD'
INFO: 2016-12-22 10:26:41: Cleaning up the setup
```

Check the database clone creation:

```
[oracle@rcrac1 gDBCclone]$ sudo /opt/gDBCclone/gDBCclone listdbs
```

Database Name	Database Type	Database Role	Master/Snapshot	Location/Parent
GOLD	SINGLE	PRIMARY	Master	/u02/.ACFS/snaps/
orcl	RAC	PRIMARY	Master	/u02/app/oracle/oradata

Get a database snapshot from clone single instance GOLD source database created above and make it as RAC:

```
$ sudo /opt/gDBClone/ gDBClone snap -sdbname GOLD \  
-tdbname S1GOLD \  
-racmod 2 \  
-opc
```

Output:

```
MacroStep1 - Getting information and validating setup...  
INFO: 2016-12-22 10:28:14: Validating environment...  
INFO: 2016-12-22 10:28:14: Superuser usage check  
INFO: 2016-12-22 10:28:14: Database 'GOLD' existence check  
INFO: 2016-12-22 10:28:14: Database 'GOLD' running check  
WARNING: 2016-12-22 10:28:16: ORACLE_BASE is not set  
INFO: 2016-12-22 10:28:16: Got Oracle Base from orabase  
INFO: 2016-12-22 10:28:16: Checking if target database name S1GOLD is a valid name  
INFO: 2016-12-22 10:28:16: Checking database 'GOLD' connectivity  
INFO: 2016-12-22 10:28:25: Checking whether the database 'GOLD' is in ACFS snapshot  
INFO: 2016-12-22 10:28:25: Checking source database 'GOLD' and target dbhome version  
INFO: 2016-12-22 10:28:29: Checking if target database 'S1GOLD' exists  
INFO: 2016-12-22 10:28:30: Checking registered instance 'S1GOLD'  
INFO: 2016-12-22 10:28:34: Checking if S1GOLD exists as snapshot in '/u02'  
INFO: 2016-12-22 10:28:34: Checking if source database GOLD is snapable  
INFO: 2016-12-22 10:28:39: ...Checking whether the database 'GOLD' is entirely on ACFS  
INFO: 2016-12-22 10:28:43: ...Checking whether the database 'GOLD' is a primary/physical standby database.  
INFO: 2016-12-22 10:28:45: ...Checking whether the database 'GOLD' is in READ WRITE mode  
INFO: 2016-12-22 10:28:50: ...Checking whether the database 'GOLD' is a CDB  
INFO: 2016-12-22 10:28:54: ...Checking whether the database 'GOLD' is running as backup mode  
INFO: 2016-12-22 10:28:58: ...Checking whether the database 'GOLD' is running in archivelog mode  
INFO: 2016-12-22 10:29:03: ...Checking if all the datafiles are available  
INFO: 2016-12-22 10:29:07: ...Checking if there are OFFLINE datafiles  
SUCCESS: 2016-12-22 10:29:11: Environment validation complete  
  
MacroStep2 - Getting database snapshot...  
INFO: 2016-12-22 10:29:11: Cloning source database 'GOLD' using ACFS snapshot  
INFO: 2016-12-22 10:29:29: Entering into SNAP database creation phase 1  
INFO: 2016-12-22 10:29:29: ...Getting required information to get consistent database snapshot  
WARNING: 2016-12-22 10:29:33: Setting the database 'GOLD' in backup mode  
INFO: 2016-12-22 10:29:48: ...Getting the snapshot of Database 'GOLD' at this time  
INFO: 2016-12-22 10:29:49: ...Successfully took the snapshot 'S1GOLD' of database 'GOLD' on '/u02'  
WARNING: 2016-12-22 10:29:59: Ending the database 'GOLD' backup mode  
INFO: 2016-12-22 10:30:04: ...Setting up storage for SNAP Database 'S1GOLD'  
INFO: 2016-12-22 10:30:27: Entering into SNAP database creation phase 2  
INFO: 2016-12-22 10:30:27: ...Creating controlfile for database 'S1GOLD'  
INFO: 2016-12-22 10:30:46: ...Recovering the database 'S1GOLD', until change:1538492  
INFO: 2016-12-22 10:30:47: ...Opening the database with resetlogs  
INFO: 2016-12-22 10:30:54: ...Setting the temporary tablespace for database 'S1GOLD'  
INFO: 2016-12-22 10:31:21: ...Changing the Database ID  
INFO: 2016-12-22 10:31:58: ...Creating spfile for S1GOLD  
INFO: 2016-12-22 10:31:59: ...Creating password file for S1GOLD  
INFO: 2016-12-22 10:31:59: Entering into SNAP database creation phase 3  
INFO: 2016-12-22 10:32:22: ...Successfully started the database  
INFO: 2016-12-22 10:32:22: ...Setting RMAN SNAPSHOT control file  
INFO: 2016-12-22 10:32:29: ...Disabling the external references in the database 'S1GOLD' inherited from 'GOLD'  
-----  
Run on the database 'S1GOLD' the SQL script:  
'/u01/app/oracle/product/11.2.0.4/dbhome_1/enable_external_refs_S1GOLD_CcSs.sql'  
to enable these external references.  
Also need to restart the database after running the SQL script.  
-----  
INFO: 2016-12-22 10:33:17: Enabling block change tracking for database 'S1GOLD'  
SUCCESS: 2016-12-22 10:33:25: Successfully created the database 'S1GOLD' from 'GOLD'
```

Continue...

```
MacroStep3 - Converting clone database 'S1GOLD' to cluster mode...
WARNING: 2016-12-22 10:33:28: Database 'S1GOLD' was already running
INFO: 2016-12-22 10:33:28: Database conversion started, it will take some time
SUCCESS: 2016-12-22 10:39:36: Database 'S1GOLD' converted to RAC succesfully
INFO: 2016-12-22 10:39:36: Cleaning up the setup
```

Check the database clone creation:

```
[oracle@rcrac1 gDBClone]$ sudo /opt/gDBClone/gDBClone listdbs
```

Database Name	Database Type	Database Role	Master/Snapshot	Location/Parent
-----	-----	-----	-----	-----
S1GOLD	RAC	PRIMARY	Snapshot	GOLD
GOLD	SINGLE	PRIMARY	Master	/u02/.ACFS/snaps/
orcl	RAC	PRIMARY	Master	/u02/app/oracle/oradata

10. Using gDBClone on ODA X6-2 S,M,L (Enterprise Edition)

At the time of writing this document, on ODA X6-2 S,M,L the snapshot database feature is not supported, then you can use gDBClone to get a database snapshot. Note as this is possible only if an Enterprise Edition deploy has been done as gDBClone is using RMAN snapshot time recovery feature available on Enterprise Edition only. Before to get a database snapshot with gDBClone you must get a source database “clone” as by default, on ODA X6-2 S,M,L the databases are stored on ACFS “root” filesystem and not on an ACFS filesystem snapshot. Once you have the clone database you could remove the source database to save storage space.

Check the source database:

```
[oracle@odas gDBClone]$ sudo /opt/gDBClone/gDBClone listdbs
```

Database Name	Database Type	Database Role	Master/Snapshot	Location/Parent
ORCL	SINGLE	PRIMARY	Master	/u02/app/oracle/oradata

Make a new ODA “DB storage” for the clone database issuing:

```
# odacli create-dbstorage --dbname CLONE
{
  "jobId" : "4c7a481c-59b0-4b70-ab30-67c24c05001a",
  "status" : "Created",
  "message" : null,
  "reports" : [ ],
  "createTimestamp" : "January 23, 2017 06:13:30 AM PST",
  "description" : "Database storage service creation with db name: CLONE",
  "updatedAtTime" : "January 23, 2017 06:13:30 AM PST"
}
```

Get a clone database from the source ‘ORCL’:

```
$ sudo /opt/gDBClone/gDBClone clone -sdbname ORCL.it.oracle.com \
-sdbscan odas-scan \
-tdbname CLONE \
-tdbhome OraDB12102_home1 \
-dataacfs /u02/app/oracle/oradata/CLONE \
-recoacfs /u03/app/oracle/fast_recovery_area \
-redoacfs /u03/app/oracle/redo
```

Output:

```
MacroStep1 - Getting information and validating setup...

Please enter the 'SYS' User password for the database ORCL.us.oracle.com:
Please re-enter the 'SYS' user password for the database ORCL.us.oracle.com:
INFO: 2017-01-23 07:49:27: Validating environment
INFO: 2017-01-23 07:49:27: Checking superuser usage
INFO: 2017-01-23 07:49:27: Checking ping to host 'rwsodas001-scan'
INFO: 2017-01-23 07:49:27: Checking if target database name 'CLONE' is a valid name
INFO: 2017-01-23 07:49:27: Checking if target database home 'Oradb12102_homel' exists
WARNING: 2017-01-23 07:49:28: ORACLE_BASE is not set
INFO: 2017-01-23 07:49:28: Got Oracle Base from orabase
INFO: 2017-01-23 07:49:28: Checking if target database 'CLONE' exists
INFO: 2017-01-23 07:49:28: Checking 'CLONE' snapshot existence on '/u02/app/oracle/oradata/CLONE'
INFO: 2017-01-23 07:49:28: Checking registered instance 'CLONE'
INFO: 2017-01-23 07:49:31: Checking listener on 'rwsodas001:1521'
INFO: 2017-01-23 07:49:31: Checking source and target database version
INFO: 2017-01-23 07:49:34: Checking source log mode
INFO: 2017-01-23 07:49:34: Checking Flash Cache setting
INFO: 2017-01-23 07:49:34: Checking ACFS command options
INFO: 2017-01-23 07:49:34: Checking if '/u02/app/oracle/oradata/CLONE' is an ACFS file system
INFO: 2017-01-23 07:49:34: Checking if '/u03/app/oracle/redo' is an ACFS file system
INFO: 2017-01-23 07:49:34: Checking if '/u03/app/oracle/fast_recovery_area' is an ACFS file system
SUCCESS: 2017-01-23 07:49:34: Environment validation complete

MacroStep2 - Setting up clone environment...
INFO: 2017-01-23 07:49:34: Creating local pfile
INFO: 2017-01-23 07:49:35: Creating local password file
INFO: 2017-01-23 07:49:35: Creating local Audit folder
INFO: 2017-01-23 07:49:35: Creating local auxiliary listener
INFO: 2017-01-23 07:49:35: Starting auxiliary listener
INFO: 2017-01-23 07:49:35: Sleeping 60 secs, please wait
INFO: 2017-01-23 07:50:35: Setting up ACFS storage
INFO: 2017-01-23 07:50:35: Creating dynamic scripts
INFO: 2017-01-23 07:50:37: Cloning to target ACFS from host 'rwsodas001-scan'
INFO: 2017-01-23 07:50:37: Creating RMAN script for spfile target to ACFS
INFO: 2017-01-23 07:50:37: Instantiating clone database
SUCCESS: 2017-01-23 07:50:37: Environment setup complete

MacroStep3 - Cloning database 'ORCL.us.oracle.com'...
INFO: 2017-01-23 07:50:37: please wait (this can take a while depending on database size and/or network speed)
INFO: 2017-01-23 07:52:05: Moving spfile
INFO: 2017-01-23 07:52:25: Updating local dbs pfile/spfile
INFO: 2017-01-23 07:52:25: Register 'CLONE' database as cluster resource
INFO: 2017-01-23 07:52:26: Checking database name
INFO: 2017-01-23 07:52:26: Modifying DB instance
INFO: 2017-01-23 07:52:27: Setup ACFS dependency
INFO: 2017-01-23 07:52:29: Database 'CLONE' dependency to
'/u02/app/oracle/oradata/CLONE,/u03/app/oracle/redo,/u03/app/oracle/fast_recovery_area' done successfully
SUCCESS: 2017-01-23 07:52:29: Successfully created clone database 'CLONE'
INFO: 2017-01-23 07:52:29: Cleaning up the setup
```

Check the new clone database:

```
[oracle@odas gDBClone]$ sudo /opt/gDBClone/gDBClone listdbs
```

Database Name	Database Type	Database Role	Master/Snapshot	Location/Parent
ORCL	SINGLE	PRIMARY	Master	/u02/app/oracle/oradata
CLONE	SINGLE	PRIMARY	Master	/u02/app/oracle/oradata/CLONE/.ACFS/snaps/

You could register the new clone database to the dcs-agent so it can be managed by the dcs-agent stack also. In order to do so you must de-register the clone db into the cluster (done by gDBClone) as “odacli register-database” will fail otherwise, the steps are as following:

1. stop the database

```
$ srvctl stop database -d CLONE
```

2. de-register the database

```
$ srvctl remove database -d CLONE
Remove the database CLONE? (y/[n]) y
```

3. startup the database using SQL*Plus

```
$ export ORACLE_SID=CLONE
$ sqlplus / as sysdba
SQL> startup
```

4. Run the ‘odacli register-database’ command:

```
# odacli register-database --dbclass OLTP --dbshape odb1 --servicename CLONE.it.oracle.com -p
Password for SYS:

{
  "jobId" : "8c99c372-f1fc-4da8-a2a6-104e45a9d4f9",
  "status" : "Created",
  "message" : null,
  "reports" : [ ],
  "createTimestamp" : "January 23, 2017 07:01:57 AM PST",
  "description" : "Database service registration with db service name: CLONE.it.oracle.com",
  "updatedAt" : "January 23, 2017 07:01:57 AM PST"
}
```

Check the new registered database:

```
# odacli list-databases
```

ID	DB Name	DB Version	CDB	Class	Shape	Storage	Status
baf88246-1538-4fee-87fe-0ede412c37ae	ORCL	12.1.0.2	false	OLTP	odb1	ACFS	Configured
589d5ac0-0556-429a-8ba6-6b3e070946a6	CLONE	12.1.0.2	false	OLTP	odb1	ACFS	Configured

You can now leverage on the **gDBClone snapshot feature**:

```
$ sudo /opt/gDBClone/gDBClone snap -sdbname CLONE \
-tdbname SNAP
```

Output:

```
MacroStep1 - Getting information and validating setup...

Please enter the 'SYS' User password for the database CLONE:
Please re-enter the 'SYS' user password for the database CLONE:
INFO: 2017-01-23 08:59:10: Validating environment...
INFO: 2017-01-23 08:59:10: Superuser usage check
INFO: 2017-01-23 08:59:10: Database 'CLONE' existence check
INFO: 2017-01-23 08:59:11: Database 'CLONE' running check
WARNING: 2017-01-23 08:59:14: ORACLE_BASE is not set
INFO: 2017-01-23 08:59:14: Got Oracle Base from orabase
INFO: 2017-01-23 08:59:14: Checking if target database name SNAP is a valid name
INFO: 2017-01-23 08:59:14: Checking database 'CLONE' connectivity
INFO: 2017-01-23 08:59:24: Checking whether the database 'CLONE' is in ACFS snapshot
INFO: 2017-01-23 08:59:24: Checking source database 'CLONE' and target dbhome version
INFO: 2017-01-23 08:59:30: Checking if target database 'SNAP' exists
INFO: 2017-01-23 08:59:30: Checking registered instance 'SNAP'
INFO: 2017-01-23 08:59:47: Checking if SNAP exists as snapshot in '/u02/app/oracle/oradata/CLONE'
INFO: 2017-01-23 08:59:47: Checking if source database CLONE is snapable
INFO: 2017-01-23 08:59:53: ...Checking whether the database 'CLONE' is entirely on ACFS
INFO: 2017-01-23 08:59:58: ...Checking whether the database 'CLONE' is a primary/physical standby database.
INFO: 2017-01-23 09:00:01: ...Checking whether the database 'CLONE' is in READ WRITE mode
INFO: 2017-01-23 09:00:07: ...Checking whether the database 'CLONE' is a CDB
INFO: 2017-01-23 09:00:17: ...Checking whether the database 'CLONE' is running as backup mode
INFO: 2017-01-23 09:00:22: ...Checking whether the database 'CLONE' is running in archivelog mode
INFO: 2017-01-23 09:00:27: ...Checking if all the datafiles are available
INFO: 2017-01-23 09:00:33: ...Checking if there are OFFLINE datafiles
SUCCESS: 2017-01-23 09:00:38: Environment validation complete

MacroStep2 - Getting database snapshot...
INFO: 2017-01-23 09:00:38: Cloning source database 'CLONE' using ACFS snapshot
INFO: 2017-01-23 09:00:59: Entering into SNAP database creation phase 1
INFO: 2017-01-23 09:00:59: ...Getting required information to get consistent database snapshot
WARNING: 2017-01-23 09:01:09: Do not perform any Structural change to database 'CLONE' till SNAP database
'SNAP' is created
INFO: 2017-01-23 09:01:21: ...Getting the snapshot of Database 'CLONE' at this time
INFO: 2017-01-23 09:01:21: ...Successfully took the snapshot 'SNAP' of database 'CLONE' on
'/u02/app/oracle/oradata/CLONE'
INFO: 2017-01-23 09:01:26: ...Setting up storage for SNAP Database 'SNAP'
INFO: 2017-01-23 09:01:48: Entering into SNAP database creation phase 2
INFO: 2017-01-23 09:01:48: ...Creating controlfile for database 'SNAP'
INFO: 2017-01-23 09:01:58: ...Recovering the database: SNAP, snapshot time : '2017-01-23:09:01:21', until
'change:1396033'
INFO: 2017-01-23 09:01:59: ...Opening the database with resetlogs
INFO: 2017-01-23 09:02:04: ...Setting the temporary tablespace for database 'SNAP'
INFO: 2017-01-23 09:02:28: ...Changing the Database ID
INFO: 2017-01-23 09:03:12: ...Creating spfile for SNAP
INFO: 2017-01-23 09:03:13: ...Creating password file for SNAP
INFO: 2017-01-23 09:03:13: Entering into SNAP database creation phase 3
INFO: 2017-01-23 09:03:32: ...Successfully started the database
INFO: 2017-01-23 09:03:38: ...Setting RMAN SNAPSHOT control file
INFO: 2017-01-23 09:03:45: ...Disabling the external references in the database 'SNAP' inherited from 'CLONE'
-----
Run on the database 'SNAP' the SQL script:
'/u01/app/oracle/product/12.1.0.2/dbhome_1/enable_external_refs_SNAP_ktnd.sql'
to enable these external references.
Also need to restart the database after running the SQL script.
-----
SUCCESS: 2017-01-23 09:04:35: Successfully created the database 'SNAP' from 'CLONE'
INFO: 2017-01-23 09:04:36: Cleaning up the setup
```


Check the SNAP database creation:

```
[oracle@odas gDBClone]$ sudo /opt/gDBClone/gDBClone listdbs
```

Database Name	Database Type	Database Role	Master/Snapshot	Location/Parent
ORCL	SINGLE	PRIMARY	Master	/u02/app/oracle/oradata
CLONE	SINGLE	PRIMARY	Master	/u02/app/oracle/oradata/CLONE/.ACFS/snaps/
SNAP	SINGLE	PRIMARY	Snapshot	CLONE

Note: DCS-Agent does not support databases created using the “gDBClone snap” feature.

11. Migrate a database from OPC to BMC using gDBClone

In this scenario, we want use gDBClone to migrate a database running on Oracle Public Cloud (OPC) to Oracle Bare Metal Cloud Service (BMC).

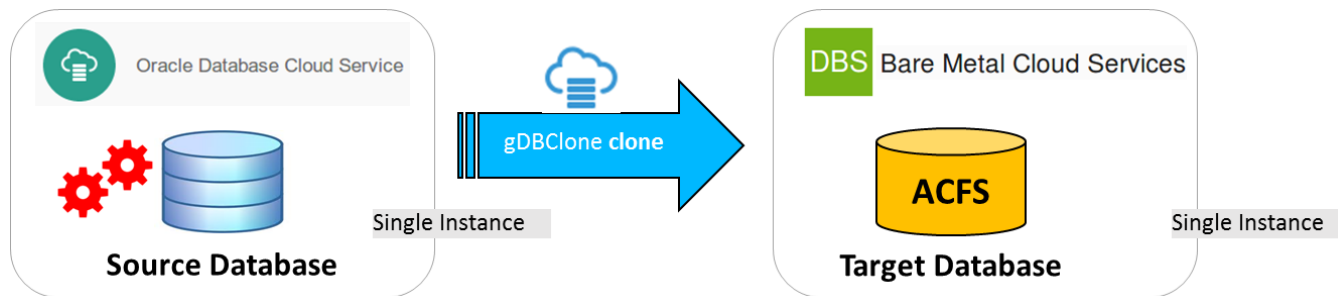


Figure 19 – OPC to BMC using gDBClone

Requirements

- Source database on OPC must have Access Rule “ora_p2_dblistener” enabled

	ora_p2_dblistener	PUBLIC-INTERNET	DB	1521	TCP	DEFAULT	
--	-------------------	-----------------	----	------	-----	---------	--

- Source database time zone must be available on target DB home (DST Patches), check the timezone in use by the database using the following query:

```
col PROPERTY_NAME format a30
col VALUE format a5
SELECT PROPERTY_NAME, SUBSTR(property_value, 1, 30) value
FROM DATABASE_PROPERTIES
WHERE PROPERTY_NAME LIKE 'DST_%' ORDER BY PROPERTY_NAME;
```

PROPERTY_NAME	VALUE
DST_PRIMARY_TT_VERSION	28
DST_SECONDARY_TT_VERSION	0
DST_UPGRADE_STATE	NONE

In this example, time zone ver.28 must be present on BMC \$ORACLE_HOME/oracore/zoneinfo:

```
[root@BMC ~]# ls -l /u01/app/oracle/product/12.1.0.2/dbhome_1/oracore/zoneinfo/*28*
-rw-r--r-- 1 oracle oinstall 53922 May 17 03:55
/u01/app/oracle/product/12.1.0.2/dbhome_1/oracore/zoneinfo/readme_28.txt
-rw-r--r-- 1 oracle oinstall 782585 May 17 03:55
/u01/app/oracle/product/12.1.0.2/dbhome_1/oracore/zoneinfo/timezlrq_28.dat
-rw-r--r-- 1 oracle oinstall 341401 May 17 03:55
/u01/app/oracle/product/12.1.0.2/dbhome_1/oracore/zoneinfo/timezone_28.dat
```

- Make on BMC a new “DB storage” for the target database issuing:

```
[root@BMC ~]# dbcli create-dbstorage --dbname RC12CBMC
{
  "jobId" : "cba4b195-80d2-413d-806e-31b2f850809a",
  "status" : "Created",
  "message" : null,
  "reports" : [ ],
  "createTimestamp" : "May 23, 2017 08:16:50 AM UTC",
  "resourceList" : [ ],
  "description" : "Database storage service creation with db name: RC12CBMC",
  "updatedAtTime" : "May 23, 2017 08:16:50 AM UTC"
}
```

- Setup the wallet file on BMC

1. Copy the wallet file (ewallet.p12) from the OPC database server to target BMC database server.

You can check the wallet file location on source database from sqlnet.ora file of the source database ORACLE_HOME

```
$ mkdir -p /opt/oracle/dcs/commonstore/wallets/tde/RC12CBMC
$ scp oracle@opc:/u01/app/oracle/admin/RC12COPC/tde_wallet/ewallet.p12
oracle@bmc:/opt/oracle/dcs/commonstore/wallets/tde/RC12CBMC/
$ chmod 600 /opt/oracle/dcs/commonstore/wallets/tde/RC12CBMC/ewallet.p12
```


2. Modify sqlnet.ora file on target BMC database ORACLE_HOME to reflect the location of the wallet file:

```
ENCRYPTION_WALLET_LOCATION =
  (SOURCE = (METHOD = FILE)
    (METHOD_DATA =
      (DIRECTORY=/opt/oracle/dcs/commonstore/wallets/tde/RC12CBMC)
    )
  )
```

3. Invoke orapki utility on the target clone database server to make the wallet auto-login (the password is an example):

```
$ orapki wallet create -wallet /opt/oracle/dcs/commonstore/wallets/tde/RC12CBMC \
  -pwd "Welcome_1" \
  -auto_login
```

You can now execute gDBClone



Create the password file for gDBClone:

```
[root@BMC ~]# /opt/gDBClone/gDBClone syspwf -syspwf /opt/gDBClone/SYS.passwd  
  
Please enter the SYS User password :  
Please re-enter the SYS user password :  
  
SYS password file created as /opt/gDBClone/SYS.passwd
```

run gDBClone as following:

```
[root@BMC ~]# nohup /opt/gDBClone/gDBClone clone \  
-sdbname RC12COPC.gboracle888888.oraclecloud.internal \  
-sdbscan 140.85.10.81 \  
-tdbname RC12CBMC \  
-tdbhome OraDB12102_home1 \  
-dataacfs /u02/app/oracle/oradata/RC12CBMC \  
-redoacfs /u03/app/oracle/redo \  
-recoacfs /u03/app/oracle/fast_recovery_area \  
-opc \  
-syspwf /opt/gDBClone/SYS.passwd &
```

Output:

```
MacroStep1 - Getting information and validating setup...
INFO: 2017-05-23 08:59:42: Validating environment
INFO: 2017-05-23 08:59:42: Checking superuser usage
INFO: 2017-05-23 08:59:42: Checking if target database name 'RC12CBMC' is a valid name
INFO: 2017-05-23 08:59:42: Checking if target database home 'OraDB12102_home1' exists
WARNING: 2017-05-23 08:59:42: ORACLE_BASE is not set
INFO: 2017-05-23 08:59:42: Got Oracle Base from orabase
INFO: 2017-05-23 08:59:42: Checking if target database 'RC12CBMC' exists
INFO: 2017-05-23 08:59:43: Checking 'RC12CBMC' snapshot existence on '/u02/app/oracle/oradata/RC12CBMC'
INFO: 2017-05-23 08:59:43: Checking registered instance 'RC12CBMC'
INFO: 2017-05-23 08:59:43: Checking listener on 'rc12c:1521'
INFO: 2017-05-23 08:59:43: Checking source and target database version
INFO: 2017-05-23 08:59:49: Checking source log mode
INFO: 2017-05-23 08:59:52: Checking Flash Cache setting
INFO: 2017-05-23 08:59:55: Checking ACFS command options
INFO: 2017-05-23 08:59:55: Checking if '/u02/app/oracle/oradata/RC12CBMC' is an ACFS file system
INFO: 2017-05-23 08:59:55: Checking if '/u03/app/oracle/redo' is an ACFS file system
INFO: 2017-05-23 08:59:55: Checking if '/u03/app/oracle/fast_recovery_area' is an ACFS file system
SUCCESS: 2017-05-23 08:59:55: Environment validation complete

MacroStep2 - Setting up clone environment...
INFO: 2017-05-23 08:59:55: Creating local pfile
INFO: 2017-05-23 08:59:58: Creating local password file
INFO: 2017-05-23 08:59:58: Creating local Audit folder
INFO: 2017-05-23 08:59:58: Creating local auxiliary listener
INFO: 2017-05-23 08:59:58: Starting auxiliary listener
INFO: 2017-05-23 08:59:58: Sleeping 60 secs, please wait
INFO: 2017-05-23 09:00:58: Setting up ACFS storage
INFO: 2017-05-23 09:00:58: Creating dynamic scripts
INFO: 2017-05-23 09:00:59: Cloning to target ACFS from host '140.85.10.81'
INFO: 2017-05-23 09:00:59: Creating RMAN script for spfile target to ACFS
INFO: 2017-05-23 09:00:59: Instantiating clone database
SUCCESS: 2017-05-23 09:00:59: Environment setup complete

MacroStep3 - Cloning database 'RC12COPC.gboracle88888.oraclecloud.internal'...
INFO: 2017-05-23 09:00:59: please wait (this can take a while depending on database size and/or network speed)
INFO: 2017-05-23 09:12:18: Moving spfile
INFO: 2017-05-23 09:12:51: Updating local dbs pfile/spfile
INFO: 2017-05-23 09:12:51: Register 'RC12CBMC' database as cluster resource
INFO: 2017-05-23 09:12:55: Checking database name
INFO: 2017-05-23 09:12:55: Modifying DB instance
INFO: 2017-05-23 09:12:56: Setup ACFS dependency
INFO: 2017-05-23 09:12:58: Database 'RC12CBMC' dependency to
'/u02/app/oracle/oradata/RC12CBMC,/u03/app/oracle/redo,/u03/app/oracle/fast_recovery_area' done successfully
INFO: 2017-05-23 09:12:58: Starting database 'RC12CBMC'
SUCCESS: 2017-05-23 09:13:10: Successfully created clone database 'RC12CBMC'
INFO: 2017-05-23 09:13:10: Cleaning up the setup
```

Check the database creation:

```
[root@BMC ~]# gDBClone listdbs
```

Database Name	Database Type	Database Role	Master/Snapshot	Location/Parent
RC12CBMC	SINGLE	PRIMARY	Master	/u02/app/oracle/oradata/RC12CBMC/.ACFS/snaps/

You could register the new clone database to the dcs-agent so it can be managed by the dcs-agent stack also. In order to do so the “COMPATIBLE” parameter must be in the form of 4 numbers (x.y.z.w) example: “12.1.0.2” (12.1.0.2.0 is not valid) and the database password must have at least two uppercase, two lowercase, two special chars and two numbers (example: “WELcome__12”). You must de-register the clone db into the cluster (done by gDBClone) as “dbcli register-database” will fail otherwise, the steps are as following:

1. set the compatible parameter

```
SQL> alter system set compatible='12.1.0.2' scope=spfile;
```

2. Set the SYS password to support “dbcli registration”

```
SQL> alter user sys identified by "WELcome__12";
```

3. stop the database

```
$ srvctl stop database -d RC12CBMC
```

4. de-register the database

```
$ srvctl remove database -d RC12CBMC
Remove the database RC12CBMC? (y/[n]) y
```

5. startup the database using SQL*Plus

```
$ export ORACLE_SID=RC12CBMC
$ sqlplus / as sysdba
SQL> startup
```

6. Run the ‘odacli register-database’ command:

```
[root@BMC ~]# dbcli register-database \
--dbclass OLTP \
--dbshape odb2 \
--servicename RC12CBMC.gboracle60892.oraclecloud.internal \
-p
Password for SYS:

{
  "jobId" : "36a28b76-43d7-4fe8-a8ff-11f96bc08e26",
  "status" : "Created",
  "message" : null,
  "reports" : [ ],
  "createTimestamp" : "May 23, 2017 09:32:52 AM UTC",
  "resourceList" : [ ],
  "description" : "Database service registration with db service name:
RC12CBMC.gboracle60000.oraclecloud.internal",
  "updatedAt" : "May 23, 2017 09:32:52 AM UTC"
}
```



Check the new registered database:

```
[root@BMC ~]# dbcli list-databases
```

ID	DB Name	DB Version	CDB	Class	Shape	Storage	Status
19e4b52a-4ef0-4aaa-8d56-cae6dd0b069d	RC12CBMC	12.1.0.2	true	OLTP	odb2	ACFS	Configured

12. Test & Dev Management environment example

Test System Configuration

- A two-node test & dev cluster: TestCluster1
- Database on test and dev cluster
 - TestCluster1 server running
 - 12.1 RACM database 'RAC'
- Database on production server
 - ProdRAC1/2 server running 'SALES' RAC database
 - ProdRAC3/4 server running 'INV' RACOne database
- ACFS file systems: /acfs and /cloudfs

It is assumed that the user is running a 12.1 production database 'SALES' on ProdRAC1/2 cluster and wants to create a clone of that database on a test & dev cluster for testing and certification purposes. Therefore, the user needs to have two snapshots of the database for this purpose

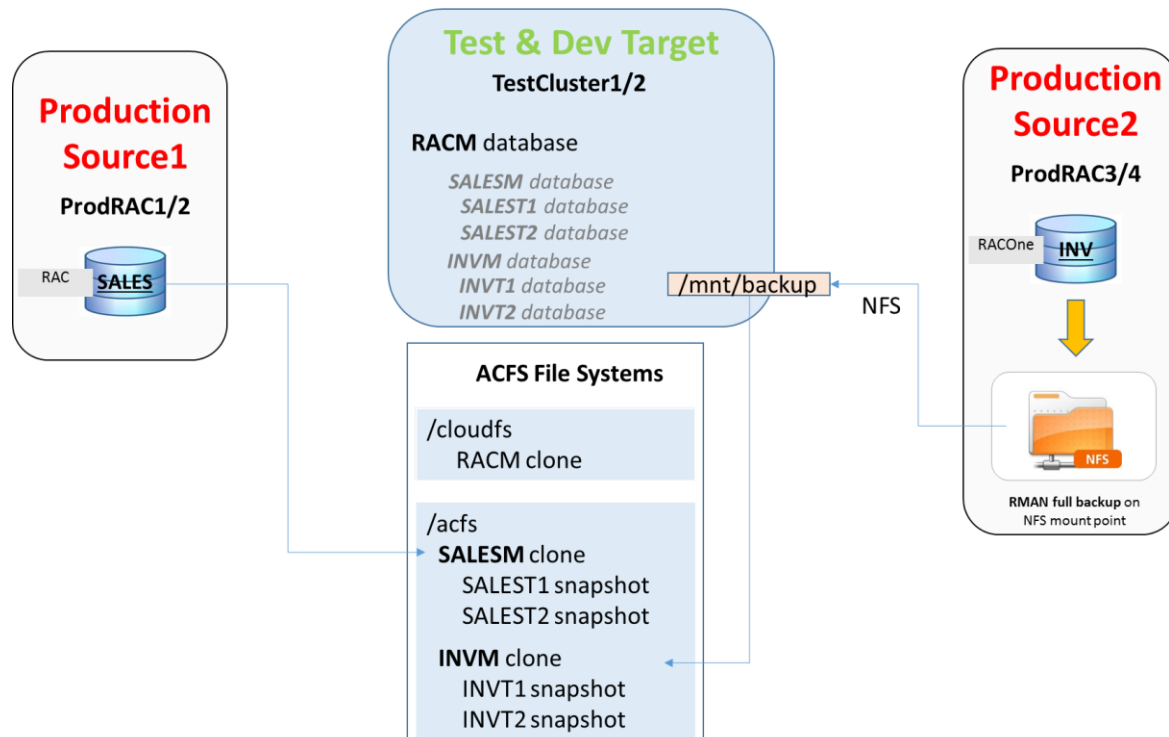


Figure 19 – Test & Dev Management environment example

The following are the steps necessary to provision two snapshot databases for test and dev purposes:

Step 1

List the current cloned databases on the test cluster.

```
[root@TestCluster1 ~]# gDBClone listdb
```

Database Name	Database Type	Database Role	Location/Parent
-----	-----	-----	-----
RACM	RAC	Master	/cloudfs

Step 2

Create a clone of RAC database 'SALES' on the test cluster and name it 'SALESM'. Create the clone on /acfs file system

```
[root@TestCluster1 ~]# gDBClone clone -sdbname SALES -sdbhost ProdRAC1 -tdbname SALESM -dataacfs /acfs -racmod 2
```

Step 3

List the current cloned databases on the test cluster.

```
[root@TestCluster1 ~]# gDBClone listdb
```

Database Name	Database Type	Database Role	Location/Parent
-----	-----	-----	-----
SALESM	RAC	Master	/acfs
RACM	RAC	Master	/cloudfs

Step 4

Create a read-write snapshot of SALESM clone database called SALEST1 and configure it as a single instance database. Also, create a read-write snapshot of SALESM clone database called SALEST2 and configure it as a RAC database.

```
[root@TestCluster1 ~]# gDBClone snap -sdbname SALESM -tdbname SALEST1
[root@TestCluster1 ~]# gDBClone snap -sdbname SALESM -tdbname SALEST2 -racmod 2
```

Step 3

List the current cloned databases on the test cluster.

```
[root@TestCluster1 ~]# gDBClone listdb
```

Database Name	Database Type	Database Role	Location/Parent
-----	-----	-----	-----
RACM	RAC	Master	/cloudfs
SALESM	RAC	Master	/acfs
SALEST1	SINGLE	Snapshot	SALESM
SALEST2	RAC	Snapshot	SALESM



Conclusion

Managing test and dev environments does not have to be complex. The Oracle Cloud File System coupled with the gDBClone script provides powerful, flexible and simple tools that ease management of test and dev servers and reduce management complexity.

You can finally contain the sprawling cost of storage by using the ACFS point-in-time snapshot technology; and therefore, realize significant storage savings. Many sparse snapshot clones can be created for parallel test and development purpose and only require a fraction of the storage. The Oracle Cloud File System is bundled with Oracle Grid Infrastructure and installs automatically on every cluster.

Refreshing and recycling test databases have never been easier. The gDBClone script allows you to create clones and snapshots, list and delete them in one simple command. This is the type of agility businesses need to adapt to changing requirements in their IT organization.

Appendix – A

Clone Location

Using the following options:

- -dataacfs Database datafiles target ACFS storage
- -redoacfs Database redologs target ACFS storage (default dataacfs)
- -recoacfs Database recovery target ACFS storage (default dataacfs)

automatically gDBClone will create:

- an ACFS snapshot under "-dataacfs <acfs mount point>" called as tdbname and the database dbfs will be stored in such place
- a folder called tdbname under "-redoacfs <acfs mount point>" and "-recoacfs <acfs mount point>" if such options are provided

```
gDBClone clone -sdbname O12C -sdbscan slcac458-scan -tdbname CO12C -tdbhome OraDb12102_home1
               -dataacfs /u02/app/oracle/oradata/datastore
               -redoacfs /u01/app/oracle/oradata/datastore
               -recoacfs /u01/app/oracle/fast_recovery_area/datastore

-->
# tree /u02/app/oracle/oradata/datastore/.ACFS/snaps/CO12C
/u02/app/oracle/oradata/datastore/.ACFS/snaps/CO12C
├── CO12C
│   ├── datafile
│   │   ├── o1_mf_sysaux_d0wd0jj3_.dbf
│   │   ├── o1_mf_system_d0wd0j6g_.dbf
│   │   ├── o1_mf_temp_d0wd2qt1_.tmp
│   │   ├── o1_mf_undotbs1_d0wd0jmb_.dbf
│   │   ├── o1_mf_undotbs2_d0wd0qt2_.dbf
│   │   └── o1_mf_users_d0wd0r1r_.dbf
│   └── spfileCO12C.ora
└── .
# tree /u01/app/oracle/oradata/datastore/CO12C/
/u01/app/oracle/oradata/datastore/CO12C/
├── CO12C
│   ├── onlinelog
│   ├── o1_mf_1_d0wd2h1n_.log
│   ├── o1_mf_2_d0wd2m7t_.log
│   ├── o1_mf_3_d0wd26qo_.log
│   ├── o1_mf_4_d0wd2boz_.log
│   └── control01.ctl
└── .
# tree /u01/app/oracle/fast_recovery_area/datastore/CO12C/
/u01/app/oracle/fast_recovery_area/datastore/CO12C/
├── CO12C
│   ├── archivelog
│   │   └── 2016_10_24
│   │       ├── o1_mf_1_4_d0wd13wv_.arc
│   │       ├── o1_mf_1_5_d0wd148z_.arc
│   │       ├── o1_mf_1_6_d0wd14jr_.arc
│   │       ├── o1_mf_2_6_d0wd14sl_.arc
│   │       └── o1_mf_2_7_d0wd153n_.arc
│   └── backupset
│       └── 2016_10_24
│           └── o1_mf_nnsnf_TAG20161024T090133_d0wd2xow_.bkp
└── .
```



Snap Location

When gDBClone snap is in use the database location is made based on the following assumptions:

Database dbf will be stored following "db_create_file_dest" source database location.

Example having:

```
db_create_file_dest='/u02/app/oracle/oradata/datastore/.ACFS/snaps/<sourceDBname>'
```

the snapshot database will store the dbf under

```
/u02/app/oracle/oradata/datastore/.ACFS/snaps/<snapDBname>/<uppercase  
snapDBname>/datafile
```

Database Redologs will follow "db_create_online_log_dest_1"

Example having:

```
db_create_online_log_dest_1='/u01/app/oracle/oradata/datastore/<sourceDBname>'
```

the snapshot database will store the dbf under

```
/u01/app/oracle/oradata/datastore/<snapDBname>
```

Database recovery area will follow "db_recovery_file_dest"

Example having:

```
db_recovery_file_dest='/u01/app/oracle/fast_recovery_area/datastore/<sourceDBname>'
```

the snapshot database will store the dbf under

```
/u01/app/oracle/fast_recovery_area/datastore/<snapDBname>
```

Standby option

The standby option (usable doing clone/snap) is as following:

```
-standby [-pmode maxperf|maxavail|maxprot] [-activedg] [-rtapply]
```

Where

-standby	The clone/snap will be a physical standby database
-pmode	Standby option: maxperf/maxavail/maxprot (default maxperf)
-activedg	Enable Active Dataguard
-rtapply	Enable real time apply

-pmode

If "-pmode" is used and ne maxperf LOG_ARCHIVE_DEST_2--> AFFIRM/ASYNC

If "-pmode" is not in use LOG_ARCHIVE_DEST_2--> NOAFFIRM/ASYNC

-activedg

Using "-activedg" the clone/snap database will be register as "-r physical_standby", "-s "READ ONLY"

Without "-activedg" the clone/snap database will be register as "-r physical_standby", "-s "mount"

-rtapply

If "-rtapply" is in use

--> "ALTER DATABASE RECOVER MANAGED STANDBY DATABASE
DISCONNECT USING CURRENT LOGFILE"

If "-rtapply" is not in use

--> "ALTER DATABASE RECOVER MANAGED STANDBY DATABASE
DISCONNECT FROM SESSION"



CONNECT WITH US



blogs.oracle.com/oracle

facebook.com/oracle

twitter.com/oracle

oracle.com

Oracle Corporation, World Headquarters

500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries

Phone: +1.650.506.7000
Fax: +1.650.506.7200

Hardware and Software, Engineered to Work Together

Copyright © 2014, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group. 0817

gDBClone - A Simple Approach to Managing Test and Development Environments Leveraging ACFS Snapshots

August 2017

Author: Ruggero Citton

Contributing Authors: Sanjay Singh, RACPack Team



Oracle is committed to developing practices and products that help protect the environment