Converged Database Continuous Availability Workshop

Part I - Real Application Clusters

Contents

Initial requirements

- SSH private key to Access the database server in the cloud. This private key is provided along with this manual.
- SSH client app, to login to the database server
- Database server public IP

Real Application Clusters

Create database services

In the following steps we are going to create database services that will support the labs performed in this workshop. The goal is to demonstrate the TAC (Transparent Application Continuity) feature, that leverages continuous availability in situations of both planned and unplanned outages.

First, gain access to the database nodes of your two nodes RAC database. You need to use the private key provided in this workshop, along with the public IP of your database servers.

Access to the database server as "**opc**" using ssh. Then connect as user "grid" and run the following commands:

```
## Run the following commands either from node 1 or node 1

ssh -i privateKey opc@<public ip of node 1>

## Connect as "grid" user and show the cluster resources

sudo su - grid

/u01/app/19.0.0.0/grid/bin/crsctl stat res -t

## Find your database name in the Cluster Resources section with the .db. Jot this
information down, you will need it for this lab. For example:

ora.lvrac_fra3md.db
      1        ONLINE  ONLINE       lvracdb-s01-2021-11-18-17Open,HOME=/u01/app/o
                                    18421                    racle/product/19.0.0
                                                             .0/dbhome_1,STABLE
      2        ONLINE  ONLINE       lvracdb-s01-2021-11-18-17Open,HOME=/u01/app/o
                                    18422                    racle/product/19.0.0
                                                             .0/dbhome_1,STABLE

## In this particular example, database unique name is lvrac_fra3md
```

Now create database services:

```
## Exit "grid" user and connect to "oracle" user:
## Execute the following commands either from node 1 or node 2

exit
sudo su - oracle

## Get the name of the instances:

ps -ef | grep pmon | grep lvrac

oracle   72990     1  0 Nov18 ?        00:00:04 ora_pmon_lvrac1

## In that case, the two instances are named lvrac1 and lvrac2
```

```
## Create a new service called svctest

srvctl add service -d $(srvctl config database) -s svctest -preferred lvrac1 -available
lvrac2 -pdb pdb1

## Start the newly created service

srvctl start service -d $(srvctl config database) -s svctest

## Check the service status: the service is running on the preferred instance

srvctl status service -d $(srvctl config database) -s svctest

Service svctest is running on instance(s) lvrac1
```

Use the lsnrctl utility to list the services on both node 1 and node 2 as the grid user:

```
## On node 2:

sudo su - grid
export ORACLE_HOME=/u01/app/19.0.0.0/grid
$ORACLE_HOME/bin/lsnrctl services

LSNRCTL for Linux: Version 19.0.0.0.0 - Production on 19-NOV-2021 09:57:17

Copyright (c) 1991, 2021, Oracle.  All rights reserved.

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER)))
Services Summary...
Service "+APX" has 1 instance(s).
  Instance "+APX2", status READY, has 1 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
Service "+ASM" has 1 instance(s).
  Instance "+ASM2", status READY, has 1 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
Service "+ASM_DATA" has 1 instance(s).
  Instance "+ASM2", status READY, has 1 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
Service "+ASM_RECO" has 1 instance(s).
  Instance "+ASM2", status READY, has 1 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
Service "c9a7e2196ee27681e0531f02640a18c1.pub.racdblab.oraclevcn.com" has 1 instance(s).
  Instance "lvrac2", status READY, has 1 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
Service "d11546103f6d7831e0538100000a30ea.pub.racdblab.oraclevcn.com" has 1 instance(s).
  Instance "lvrac2", status READY, has 1 handler(s) for this service...
```

```
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
Service "lvracXDB.pub.racdblab.oraclevcn.com" has 1 instance(s).
  Instance "lvrac2", status READY, has 1 handler(s) for this service...
    Handler(s):
      "D000" established:0 refused:0 current:0 max:1022 state:ready
         DISPATCHER <machine: lvracdb-s01-2021-11-18-1718422, pid: 73364>
         (ADDRESS=(PROTOCOL=tcp)(HOST=lvracdb-s01-2021-11-18-
1718422.pub.racdblab.oraclevcn.com)(PORT=58310))
Service "lvrac_fra3md.pub.racdblab.oraclevcn.com" has 1 instance(s).
  Instance "lvrac2", status READY, has 1 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
Service "pdb1.pub.racdblab.oraclevcn.com" has 1 instance(s).
  Instance "lvrac2", status READY, has 1 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
The command completed successfully

$ORACLE_HOME/bin/lsnrctl status LISTENER

LSNRCTL for Linux: Version 19.0.0.0.0 - Production on 19-NOV-2021 09:57:54

Copyright (c) 1991, 2021, Oracle.  All rights reserved.

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version 19.0.0.0.0 - Production
Start Date                18-NOV-2021 17:56:30
Uptime                    0 days 16 hr. 1 min. 24 sec
Trace Level               off
Security                  ON: Local OS Authentication
SNMP                      OFF
Listener Parameter File   /u01/app/19.0.0.0/grid/network/admin/listener.ora
Listener Log File         /u01/app/grid/diag/tnslsnr/lvracdb-s01-2021-11-18-
1718422/listener/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=LISTENER)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=10.0.0.146)(PORT=1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=10.0.0.139)(PORT=1521)))
Services Summary...
Service "+APX" has 1 instance(s).
  Instance "+APX2", status READY, has 1 handler(s) for this service...
Service "+ASM" has 1 instance(s).
  Instance "+ASM2", status READY, has 1 handler(s) for this service...
Service "+ASM_DATA" has 1 instance(s).
  Instance "+ASM2", status READY, has 1 handler(s) for this service...
Service "+ASM_RECO" has 1 instance(s).
  Instance "+ASM2", status READY, has 1 handler(s) for this service...
Service "c9a7e2196ee27681e0531f02640a18c1.pub.racdblab.oraclevcn.com" has 1 instance(s).
  Instance "lvrac2", status READY, has 1 handler(s) for this service...
Service "d11546103f6d7831e0538100000a30ea.pub.racdblab.oraclevcn.com" has 1 instance(s).
  Instance "lvrac2", status READY, has 1 handler(s) for this service...
```

```
Service "lvracXDB.pub.racdblab.oraclevcn.com" has 1 instance(s).
  Instance "lvrac2", status READY, has 1 handler(s) for this service...
Service "lvrac_fra3md.pub.racdblab.oraclevcn.com" has 1 instance(s).
  Instance "lvrac2", status READY, has 1 handler(s) for this service...
Service "pdb1.pub.racdblab.oraclevcn.com" has 1 instance(s).
  Instance "lvrac2", status READY, has 1 handler(s) for this service...
The command completed successfully
```

As "svctest" service is running only on node 1, it doesn't appear among the local listener services on node 2.
Repeat the commands on node 1, as "grid" user:

```
$ORACLE_HOME/bin/lsnrctl services

LSNRCTL for Linux: Version 19.0.0.0.0 - Production on 19-NOV-2021 10:06:51

Copyright (c) 1991, 2021, Oracle.  All rights reserved.

Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=LISTENER)))
Services Summary...
Service "+APX" has 1 instance(s).
  Instance "+APX1", status READY, has 1 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
Service "+ASM" has 1 instance(s).
  Instance "+ASM1", status READY, has 1 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
Service "+ASM_DATA" has 1 instance(s).
  Instance "+ASM1", status READY, has 1 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
Service "+ASM_RECO" has 1 instance(s).
  Instance "+ASM1", status READY, has 1 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
Service "c9a7e2196ee27681e0531f02640a18c1.pub.racdblab.oraclevcn.com" has 1 instance(s).
  Instance "lvrac1", status READY, has 1 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
Service "d11546103f6d7831e0538100000a30ea.pub.racdblab.oraclevcn.com" has 1 instance(s).
  Instance "lvrac1", status READY, has 1 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
Service "lvracXDB.pub.racdblab.oraclevcn.com" has 1 instance(s).
  Instance "lvrac1", status READY, has 1 handler(s) for this service...
    Handler(s):
      "D000" established:0 refused:0 current:0 max:1022 state:ready
         DISPATCHER <machine: lvracdb-s01-2021-11-18-1718421, pid: 8733>
```

```
         (ADDRESS=(PROTOCOL=tcp)(HOST=lvracdb-s01-2021-11-18-
1718421.pub.racdblab.oraclevcn.com)(PORT=57366))
Service "lvrac_fra3md.pub.racdblab.oraclevcn.com" has 1 instance(s).
  Instance "lvrac1", status READY, has 1 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
Service "pdb1.pub.racdblab.oraclevcn.com" has 1 instance(s).
  Instance "lvrac1", status READY, has 1 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
Service "svctest.pub.racdblab.oraclevcn.com" has 1 instance(s).
  Instance "lvrac1", status READY, has 1 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
The command completed successfully
```

On node 1 service svctest.pub.racdblab.oraclevcn.com has been registered in the local listener.

Cause the service to fail over. After identifying which instance the service is being offered on, kill that instance by removing the SMON process at the operating system level. Run this on node 1:

```
sudo su - oracle
ps -ef | grep ora_smon
oracle    8627     1  0 Nov18 ?        00:00:02 ora_smon_lvrac1
oracle   92886 92852  0 10:17 pts/0    00:00:00 grep --color=auto ora_smon

kill  -9 8627

## Check the service status:
srvctl status service -d $(srvctl config database) -s svctest

Service svctest is running on instance(s) lvrac2
```

The service failed over node 2.

Manually relocate the service on node 1:

```
srvctl relocate service -d $(srvctl config database) -s svctest -oldinst lvrac2 -newinst
lvrac1

srvctl status service -d $(srvctl config database) -s svctest

Service svctest is running on instance(s) lvrac1
```

Configure services for Application Continuity

FAN, connection identifier, TAC, AC, switchover, consumer groups, and many other features and operations are committed to the use of database services.

Do not use the default database service (the service created automatically with the same name as the database or PDB) as this service cannot be disabled, relocated, or restricted and so has no high availability support.

The services you use are associated with a specific primary or standby role in a Data Guard environment.

Attributes set on the service enable applications to use Application Continuity.

Create three services, setting the attributes failover_restore, commit_outcome, and failovertype for Application Continuity (AC) and Transparent Application Continuity (TAC), and leaving one of them without AC or TAC settings.

Replace the values for "-preferred" and "-available" with those of your system.

```
## As oracle user, create a service with AC settings
## Run the following on any node

srvctl add service -d $(srvctl config database) -s svc_ac -commit_outcome TRUE -
failovertype TRANSACTION -failover_restore LEVEL1 -preferred lvrac1 -available lvrac2 -
pdb pdb1 -clbgoal LONG -rlbgoal NONE

## Create a service named noac with no AC settings

srvctl add service -d $(srvctl config database) -s noac -commit_outcome FALSE -
failovertype NONE -failover_restore NONE -preferred lvrac1 -available lvrac2 -pdb pdb1 -
clbgoal LONG -rlbgoal NONE

## Create a service named tac_service with TAC settings

srvctl add service -d $(srvctl config database) -s tac_service -commit_outcome TRUE -
failovertype AUTO -failover_restore AUTO -preferred lvrac1 -available lvrac2 -pdb pdb1 -
clbgoal LONG -rlbgoal NONE


## Start the three services

srvctl start service -d $(srvctl config database) -s svc_ac
srvctl start service -d $(srvctl config database) -s noac
srvctl start service -d $(srvctl config database) -s tac_service


## Check their status

srvctl status service -d $(srvctl config database) -s svc_ac

Service svc_ac is running on instance(s) lvrac1

srvctl status service -d $(srvctl config database) -s noac

Service noac is running on instance(s) lvrac1

srvctl status service -d $(srvctl config database) -s tac_service

Service tac_service is running on instance(s) lvrac1
```

To enable TAC:

- commit_outcome is TRUE
- failovertype is set to AUTO
- failover_restore is AUTO.

To enable AC:

- commit_outcome is TRUE
- failovertype is set to TRANSACTION
- failover_restore is LEVEL1.

Note: The attributes **failoverretry** and **failoverdelay** are not required when RETRY_COUNT and RETRY_DELAY are set in the connect string\/URL as recommended.

This concludes the initial services setup.

Demonstrate Application Continuity

In the following steps, we will install and use an application to illustrate AC and TAC.

Install the sample program **on node 1**:

```
-- Connect to node1 as "oracle", and install a sample application:

cd /home/oracle
wget https://objectstorage.us-ashburn-
1.oraclecloud.com/p/O8AOujhwl1dSTqhfH69f3nkV6TNZWU3KaIF4TZ-XuCaZ5w-
xHEQ14ViOVhUXQjPB/n/oradbclouducm/b/LiveLabTemp/o/ACDemo_19c.zip

--2021-11-19 10:32:10--  https://objectstorage.us-ashburn-
1.oraclecloud.com/p/O8AOujhwl1dSTqhfH69f3nkV6TNZWU3KaIF4TZ-XuCaZ5w-
xHEQ14ViOVhUXQjPB/n/oradbclouducm/b/LiveLabTemp/o/ACDemo_19c.zip
Resolving objectstorage.us-ashburn-1.oraclecloud.com (objectstorage.us-ashburn-
1.oraclecloud.com)... 134.70.32.1, 134.70.24.1, 134.70.28.1
Connecting to objectstorage.us-ashburn-1.oraclecloud.com (objectstorage.us-ashburn-
1.oraclecloud.com)|134.70.32.1|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 8573765 (8.2M) [application/x-zip-compressed]
Saving to: 'ACDemo_19c.zip'

100%[=============================================================================
===============>] 8,573,765   12.2MB/s   in 0.7s

2021-11-19 10:32:11 (12.2 MB/s) - 'ACDemo_19c.zip' saved [8573765/8573765]

## Unzip the ZIP file

cd /home/oracle
unzip ACDemo_19c.zip

ls -ltr
```

```
total 8392
-rw-r--r-- 1 oracle oinstall    2717 Mar 16  2021 README.txt
drwxr-xr-x 6 oracle oinstall    4096 Sep  8 13:42 acdemo
-rw-r--r-- 1 oracle oinstall    7990 Sep  9 13:46 SETUP_AC_TEST.sh
-rw-r--r-- 1 oracle oinstall 8573765 Sep 10 04:21 ACDemo_19c.zip

## This created a "acdemo" directory
## Set the execute bit +x on the SETUP_AC_TEST.sh script

chmod +x SETUP_AC_TEST.sh

## Run the script SETUP_AC_TEST.sh. You will be prompted for INPUTS. If a default value
is shown, press ENTER to accept except for service name

./SETUP_AC_TEST.sh
Enter a value for ORACLE_HOME [/u01/app/oracle/product/19.0.0.0/dbhome_1]:
Enter the database name [lvrac_fra3qb]:
Enter the PDB to use [pdb1]:
Enter the SYSTEM user password:   <password>
Enter the SCAN name [lvracdb-s01-2021-11-28-130443-scan.pub.racdblab.oraclevcn.com]:
Enter a new service name [ac_service]:  svc_ac
Enter the tablespace name for the HR user [USERS]:

## Make the run scripts executable

cd /home/oracle/acdemo
chmod +x run*
chmod +x kill_session.sh

## Identify your service names:

srvctl status service -d $(srvctl config database)

Service ac_service is running on instance(s) lvrac1
Service noac is running on instance(s) lvrac1
Service svc_ac is running on instance(s) lvrac1
Service svctest is running on instance(s) lvrac1
Service tac_service is running on instance(s) lvrac1
Service unisrv is running on instance(s) lvrac1,lvrac2
```

Run the application in NOREPLAY mode

We are going to run the application with neither AC nor TAC.
Move to /home/oracle/acdemo directory, and check the content of " ac_noreplay.properties" file:

```
## As oracle, on node 1:
cd /home/oracle/acdemo
cat ac_noreplay.properties

#Stub file to build ac_noreplay.properties
# Use vanilla datasource
datasource=oracle.jdbc.pool.OracleDataSource

# Set verbose mode
VERBOSE=FALSE
```

```
# database JDBC URL
url=jdbc:oracle:thin:@(DESCRIPTION=(CONNECT_TIMEOUT=90)(RETRY_COUNT=50)(RETRY_DELAY=3)(T
RANSPORT_CONNECT_TIMEOUT=3)(ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp)(HOST=lvracdb-s01-2021-
11-18-171842-
scan.pub.racdblab.oraclevcn.com)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=noac.pub.racdbl
ab.oraclevcn.com)))

# database username and password
username=hr
password=W3lc0m3#W3lc0m3#

# Disable FAN
fastConnectionFailover=FALSE

#Disable connection tests
validateConnectionOnBorrow=FALSE

# number of connections in the UCP''s pool
ucp_pool_size=20

#Connection Wait Timeout for busy pool
connectionWaitTimeout=5

# number of active threads (this simulates concurrent load)
number_of_threads=10

# think time is how much time the threads will sleep before looping
thread_think_time=50
```

Note the application will use the "noac" service to connect to the database.
Now, run the application with no replay:

```
cd /home/oracle/acdemo
./runnoreplay

#########################################################
Connecting to
jdbc:oracle:thin:@(DESCRIPTION=(CONNECT_TIMEOUT=90)(RETRY_COUNT=50)(RETRY_DELAY=3)(TRANS
PORT_CONNECT_TIMEOUT=3)(ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp)(HOST=lvracdb-s01-2021-11-
18-171842-
scan.pub.racdblab.oraclevcn.com)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=noac.pub.racdbl
ab.oraclevcn.com)))
 # of Threads             : 10
 UCP pool size            : 20
FCF Enabled:  false
VCoB Enabled: false
ONS Configuration:  null
Enable Intensive Wload:  false
Thread think time        : 50 ms
#########################################################

Starting the pool now... (please wait)
Pool is started in 7438ms
2 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 682, avg response time from
db 12ms
```

```
0 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 1518, avg response time
from db 7ms
2 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 2340, avg response time
from db 8ms
[...]
```

From another terminal, kill the SMON process of the instance where "noac" service is currently running (node 1):

```
sudo su - oracle
ps -ef | grep smon

root     35442     1  1 Nov18 ?        00:17:40 /u01/app/19.0.0.0/grid/bin/osysmond.bin
grid     36088     1  0 Nov18 ?        00:00:01 asm_smon_+ASM1
oracle   39453 39393  0 10:58 pts/1    00:00:00 grep --color=auto smon
oracle   94293     1  0 10:18 ?        00:00:00 ora_smon_lvrac1

kill -9 94293
```

Go back to the terminal where the application is running, and observe the result:

```
java.sql.SQLRecoverableException: No more data to read from socket
       at
oracle.jdbc.driver.T4CMAREngineNIO.prepareForUnmarshall(T4CMAREngineNIO.java:811)
       at oracle.jdbc.driver.T4CMAREngineNIO.unmarshalUB1(T4CMAREngineNIO.java:449)
       at oracle.jdbc.driver.T4CTTIfun.receive(T4CTTIfun.java:410)
       at oracle.jdbc.driver.T4CTTIfun.doRPC(T4CTTIfun.java:269)
       at oracle.jdbc.driver.T4C8Oall.doOALL(T4C8Oall.java:655)
       at oracle.jdbc.driver.T4CPreparedStatement.doOall8(T4CPreparedStatement.java:270)
       at oracle.jdbc.driver.T4CPreparedStatement.doOall8(T4CPreparedStatement.java:91)
       at
oracle.jdbc.driver.T4CPreparedStatement.executeForDescribe(T4CPreparedStatement.java:807
)
       at
oracle.jdbc.driver.OracleStatement.executeMaybeDescribe(OracleStatement.java:983)
       at
oracle.jdbc.driver.OracleStatement.doExecuteWithTimeout(OracleStatement.java:1168)
       at
oracle.jdbc.driver.OraclePreparedStatement.executeInternal(OraclePreparedStatement.java:
3666)
       at
oracle.jdbc.driver.T4CPreparedStatement.executeInternal(T4CPreparedStatement.java:1426)
       at
oracle.jdbc.driver.OraclePreparedStatement.executeQuery(OraclePreparedStatement.java:371
3)
       at
oracle.jdbc.driver.OraclePreparedStatementWrapper.executeQuery(OraclePreparedStatementWr
apper.java:1167)
       at
oracle.ucp.jdbc.proxy.oracle$1ucp$1jdbc$1proxy$1oracle$1StatementProxy$2oracle$1jdbc$1in
ternal$1OraclePreparedStatement$$$Proxy.executeQuery(Unknown Source)
       at acdemo.Worker.databaseWorkload(Worker.java:56)
       at acdemo.Worker.run(Worker.java:137)
       at java.lang.Thread.run(Thread.java:748)
```

```
Application error handling: attempting to get a new connection No more data to read from
socket.
FCF information:
0 borrowed, 10 pending, 0ms getConnection wait, TotalBorrowed 5211, avg response time
from db 14ms
 Application driven connection retry succeeded
 Application driven connection retry succeeded
 Application driven connection retry succeeded
 Application driven connection retry succeeded
 Application driven connection retry succeeded
 Application driven connection retry succeeded
 Application driven connection retry succeeded
 Application driven connection retry succeeded
 Application driven connection retry succeeded
 Application driven connection retry succeeded
1 borrowed, 0 pending, 11ms getConnection wait, TotalBorrowed 5355, avg response time
from db 84ms
0 borrowed, 0 pending, 10ms getConnection wait, TotalBorrowed 5806, avg response time
from db 57ms
```

You observe that the application was disconnected from instance 1, and catched an error (in red). Then, as service noac was failed over node 2, the application reconnected **programmatically** to the noac service, now running on node 2 (in green). Once reconnected, the sessions started working again on node 2 (in blue).

This illustrates that without AC/TAC, you need to code your transactional applications so that they catch the exceptions that occur when an instance goes down, and the database service is failed over surviving instances.

Type CTRL-C in the app terminal to stop the sample application.

Run the application in AC mode

Now we will re-run the application in AC mode.

Examine the ac_replay.properties file to see that we are using a replay datasource oracle.jdbc.replay.OracleDataSourceImpl and we have enabled FAN, fastConnectionFailover=TRUE and connection tests validateConnectionOnBorrow=TRUE.

The URL uses the recommended format and connects to the service you created previously, which has AC attributes set.

```
cd /home/oracle/acdemo
cat ac_replay.properties

# Stub file to create ac_replay.properties
# Use replay datasource
datasource=oracle.jdbc.replay.OracleDataSourceImpl

# Set verbose mode
VERBOSE=FALSE

# database JDBC URL
```

```
url=jdbc:oracle:thin:@(DESCRIPTION=(CONNECT_TIMEOUT=90)(RETRY_COUNT=50)(RETRY_DELAY=3)(T
RANSPORT_CONNECT_TIMEOUT=3)(ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp)(HOST=lvracdb-s01-2021-
11-18-171842-scan.pub.racdblab.oraclevcn.com)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=
svc_ac.pub.racdblab.oraclevcn.com)))

# database username and password:
username=hr
password=<password>

# Enable FAN
fastConnectionFailover=TRUE

#Disable connection tests
validateConnectionOnBorrow=TRUE

# number of connections in the UCP''s pool:
ucp_pool_size=20

#Connection Wait Timeout for busy pool
connectionWaitTimeout=5

# number of active threads (this simulates concurrent load):
number_of_threads=10

# think time is how much time the threads will sleep before looping:
thread_think_time=50

## Check the service:

srvctl status service -d $(srvctl config database) -s svc_ac

Service ac_service is running on instance(s) lvrac2

## Write down the node the service is running on !!!

-- Start app in replay mode:

cd /home/oracle/acdemo
./runreplay

#########################################################
Connecting to
jdbc:oracle:thin:@(DESCRIPTION=(CONNECT_TIMEOUT=90)(RETRY_COUNT=50)(RETRY_DELAY=3)(TRANS
PORT_CONNECT_TIMEOUT=3)(ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp)(HOST=lvracdb-s01-2021-11-
18-171842-scan.pub.racdblab.oraclevcn.com)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=
svc_ac.pub.racdblab.oraclevcn.com)))
 # of Threads             : 10
 UCP pool size            : 20
FCF Enabled:  true
VCoB Enabled: true
ONS Configuration:  null
Enable Intensive Wload:  false
Thread think time        : 50 ms
#########################################################

Starting the pool now... (please wait)
Pool is started in 6286ms
```

```
4 borrowed, 0 pending, 1ms getConnection wait, TotalBorrowed 507, avg response time from
db 26ms
5 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 1186, avg response time
from db 19ms
[...]


-- Kill smon on the instance where the service is running

ps -ef | grep smon

root     35442     1  1 Nov18 ?        01:29:41 /u01/app/19.0.0.0/grid/bin/osysmond.bin
grid     36088     1  0 Nov18 ?        00:00:06 asm_smon_+ASM2
oracle   64589     1  0 08:57 ?        00:00:00 ora_smon_lvrac2
oracle   73308 62438  0 09:03 pts/1    00:00:00 grep --color=auto smon

kill -9 64589

## Observe the application output
[...]
8 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 1926, avg response time
from db 13ms
1 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 2627, avg response time
from db 17ms
2 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 3297, avg response time
from db 20ms
1 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 4003, avg response time
from db 17ms
0 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 4721, avg response time
from db 15ms
4 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 5339, avg response time
from db 26ms
4 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 6055, avg response time
from db 16ms
10 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 6491, avg response time
from db 13ms
10 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 6551, avg response time
from db 1072ms   <=    avg response time briefly increases, but no error !!!
1 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 7091, avg response time
from db 44ms
2 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 7827, avg response time
from db 14ms
4 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 8589, avg response time
from db 12ms
```

No errors occur. Application Continuity traps the error(s), re-establishes connections at a surviving instance, and replays any uncommitted transactions.
We do not progress into any of the application"s error handling routines.
Type CTRL-C in the app terminal to stop the sample application.

Run the application in TAC mode

Examine the tac_replay.properties file to see that we are using a replay datasource oracle.jdbc.replay.OracleDataSourceImpl and we have enabled FAN, fastConnectionFailover=TRUE and connection tests validateConnectionOnBorrow=TRUE.

The URL uses the recommended format and connects to the service you created previously, which has TAC attributes set.

```
cat tac_replay.properties

# Stub file to create tac_replay.properties
# Use replay datasource
datasource=oracle.jdbc.replay.OracleDataSourceImpl

# Set verbose mode
VERBOSE=FALSE

# database JDBC URL
url=jdbc:oracle:thin:@(DESCRIPTION=(CONNECT_TIMEOUT=90)(RETRY_COUNT=50)(RETRY_DELAY=3)(T
RANSPORT_CONNECT_TIMEOUT=3)(ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp)(HOST=lvracdb-s01-2021-
11-18-171842-
scan.pub.racdblab.oraclevcn.com)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=tac_service.pub
.racdblab.oraclevcn.com)))

# database username and password:
username=hr
password=W3lc0m3#W3lc0m3#

# Enable FAN
fastConnectionFailover=TRUE

#Disable connection tests
validateConnectionOnBorrow=TRUE

# number of connections in the UCP''s pool:
ucp_pool_size=20

#Connection Wait Timeout for busy pool
connectionWaitTimeout=5

# number of active threads (this simulates concurrent load):
number_of_threads=10

# think time is how much time the threads will sleep before looping:
thread_think_time=50

## Check the service

srvctl status service -d $(srvctl config database) -s tac_service

Service tac_service is running on instance(s) lvrac2

## Write down the node the service is running on !!!

## Run the application in TAC mode

cd /home/oracle/acdemo
./runtacreplay

[oracle@lvracdb-s01-2021-11-18-1718421 acdemo]$ ./runtacreplay
#########################################################
Connecting to
jdbc:oracle:thin:@(DESCRIPTION=(CONNECT_TIMEOUT=90)(RETRY_COUNT=50)(RETRY_DELAY=3)(TRANS
```

```
PORT_CONNECT_TIMEOUT=3)(ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp)(HOST=lvracdb-s01-2021-11-
18-171842-
scan.pub.racdblab.oraclevcn.com)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=tac_service.pub
.racdblab.oraclevcn.com)))
 # of Threads             : 10
 UCP pool size            : 20
FCF Enabled:  true
VCoB Enabled: true
ONS Configuration:  null
Enable Intensive Wload:  false
Thread think time        : 50 ms
#########################################################

Starting the pool now... (please wait)
Pool is started in 4744ms
1 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 619, avg response time from
db 17ms
2 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 1417, avg response time
from db 9ms
1 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 2234, avg response time
from db 7ms
5 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 3040, avg response time
from db 8ms
[...]

-- Kill smon on the instance where the service is running

ps -ef | grep smon

oracle    9036  7516  0 09:32 pts/1    00:00:00 grep --color=auto smon
root     35442     1  1 Nov18 ?        01:30:15 /u01/app/19.0.0.0/grid/bin/osysmond.bin
grid     36088     1  0 Nov18 ?        00:00:06 asm_smon_+ASM2
oracle   75842     1  0 09:04 ?        00:00:00 ora_smon_lvrac2

kill -9 75842

## Observe the application output

1 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 4642, avg response time
from db 8ms
5 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 5462, avg response time
from db 8ms
1 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 6283, avg response time
from db 7ms
3 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 7097, avg response time
from db 8ms
1 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 7922, avg response time
from db 7ms
1 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 8741, avg response time
from db 7ms
10 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 8877, avg response time
from db 6ms
2 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 9444, avg response time
from db 107ms   <=== Slight RT increment !!!
0 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 10267, avg response time
from db 7ms
1 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 11094, avg response time
from db 7ms
```

```
1 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 11917, avg response time
from db 7ms
1 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 12742, avg response time
from db 7ms
0 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 13588, avg response time
from db 6ms
0 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 14430, avg response time
from db 6ms
2 borrowed, 0 pending, 0ms getConnection wait, TotalBorrowed 15268, avg response time
from db 6ms
```

Type CTRL-C in the app terminal to stop the sample application.

TAC will protect applications that do, or do not use a connection pool. Let's try that with an Sql*Plus connection:

```
## Check the TAC service, from any node, as "oracle" user

srvctl status service -d $(srvctl config database) -s tac_service

Service tac_service is running on instance(s) lvrac1

## If tac_service is not running on instance 1, relocate it to instance 1

srvctl relocate service -d $(srvctl config database) -s tac_service -oldinst lvrac2 -
newinst lvrac1

## Connect to the database with SQL*Plus as the HR user over the TAC-enabled service.
You can copy this connection string from tac_replay.properties file.
sqlplus
hr/<password>@"(DESCRIPTION=(CONNECT_TIMEOUT=90)(RETRY_COUNT=50)(RETRY_DELAY=3)(TRANSPOR
T_CONNECT_TIMEOUT=3)(ADDRESS_LIST=(ADDRESS=(PROTOCOL=tcp)(HOST=lvracdb-s01-2021-11-18-
171842-
scan.pub.racdblab.oraclevcn.com)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=tac_service.pub
.racdblab.oraclevcn.com)))"

-- Update a row in the table EMP4AC. For example:

select empno, ename  from emp4ac where rownum < 10;

     EMPNO ENAME
---------- ----------
      5814 Bob5814
      2271 Bob2271
      3538 Bob3538
      3033 Bob3033
      7151 Bob7151
      8948 Bob8948
      5642 Bob5642
     -9208 Bob-9208
     -7790 Bob-7790

9 rows selected.

update emp4ac set empno=9999 where empno=5642 and ename='Bob5642' and rownum < 10;
```

```
1 row updated.

## Leave the transaction uncommitted !!!

## From another terminal, kill the session with the uncommitted transaction
## This MUST be run on node 1, where the application has been deployed:

cd /home/oracle/acdemo
./kill_session.sh tac_service.pub.racdblab.oraclevcn.com

SQL*Plus: Release 19.0.0.0.0 - Production on Mon Nov 22 11:48:42 2021
Version 19.12.0.0.0

Copyright (c) 1982, 2021, Oracle.  All rights reserved.

Last Successful login time: Fri Nov 19 2021 12:08:57 +00:00

Connected to:
Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 - Production
Version 19.12.0.0.0

SQL> SQL>
'ALTERSYSTEMKILLSESSION'''||SID||','||SERIAL#||'''IMMEDIATE;'
--------------------------------------------------------------------------------
ALTER SYSTEM KILL SESSION '346,24628' IMMEDIATE;

SQL> SQL> Disconnected from Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 -
Production
Version 19.12.0.0.0

SQL*Plus: Release 19.0.0.0.0 - Production on Mon Nov 22 11:48:43 2021
Version 19.12.0.0.0

Copyright (c) 1982, 2021, Oracle.  All rights reserved.

Last Successful login time: Mon Nov 22 2021 11:48:43 +00:00

Connected to:
Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 - Production
Version 19.12.0.0.0

SQL>
System altered.

SQL> Disconnected from Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 -
Production
Version 19.12.0.0.0

## Go back to your sqlplus session, and commit your transaction:

SQL> commit;

Commit complete.
```

TAC protected your Sql*Plus session, even from an explicit "kill session". This leverages "zero downtime" with planned and unplanned outages.

Use Fast Application Notification (FAN)

In the following steps, we will demonstrate FAN with Real Application Clusters. With FAN, the continuous service and continuous connections built into Real Application Clusters and Data Guard are extended to applications and application servers. When the state of database services change, (for example, up, down, or unresponsive), the new status is posted to interested subscribers through FAN events. FAN provides rapid notification about state changes for database services, instances, the databases themselves, and the nodes that form the cluster, and starting with Oracle Database 12c with Global Data Services, distributed database systems.

First we will demonstrate FAN with a simple shell script callout. Connect to both servers as "grid" user:

```
ssh -i privateKey opc@<public ip of node 1>

## Connect as "grid" user and move to FAN callouts default directory:

sudo su - grid
cd /u01/app/19.0.0.0/grid/racg/usrco/

## Use "vi" editor to create a simple callout script: callout-log.sh
## The content of this callout file must be as follows:

cat callout-log.sh

#!/usr/bin/bash
umask 022
FAN_LOGFILE=/tmp/`hostname -s`_events.log
echo $* " reported = "`date` >> ${FAN_LOGFILE} &

chmod +x callout-log.sh

### REPEAT THE PREVIOUS STEPS ON NODE2
```

Now that the simple callout script has been created on both nodes, let's generate an event that will be trapped by FAN:

```
## As grid, from node 1: list the cluster resources

crsctl stat res -t

## Stop force instance 1:

srvctl stop instance -d $(srvctl config database) -i <Name of instance 1> -o immediate -
force

## Check the database status on all nodes: confirm that instance 1 is not running

srvctl status database -d $(srvctl config database)
```

If your callout was written correctly and had the appropriate execute permissions, a file named **hostname**_events.log should be visible in the /tmp directory. Check the content of this file:

```
## For example:

[grid@sduclunode1 usrco]$ cat /tmp/sduclunode1_events.log
SERVICEMEMBER VERSION=1.0 service=ac_service.sub06221433571.skynet.oraclevcn.com
database=sdurac_dbsdu instance=SDURAC1 host=sduclunode1 status=down reason=USER
timestamp=2021-07-01 12:39:52 timezone=+00:00
db_domain=sub06221433571.skynet.oraclevcn.com  reported = Thu Jul 1 12:39:52 UTC 2021
SERVICE VERSION=1.0 service=ac_service.sub06221433571.skynet.oraclevcn.com
database=sdurac_dbsdu instance=SDURAC1 host=sduclunode1 status=down reason=USER
timestamp=2021-07-01 12:39:52 timezone=+00:00
db_domain=sub06221433571.skynet.oraclevcn.com  reported = Thu Jul 1 12:39:52 UTC 2021
SERVICEMEMBER VERSION=1.0 service=tac_service.sub06221433571.skynet.oraclevcn.com
database=sdurac_dbsdu instance=SDURAC1 host=sduclunode1 status=down reason=USER
timestamp=2021-07-01 12:39:52 timezone=+00:00
db_domain=sub06221433571.skynet.oraclevcn.com  reported = Thu Jul 1 12:39:52 UTC 2021
SERVICE VERSION=1.0 service=tac_service.sub06221433571.skynet.oraclevcn.com
database=sdurac_dbsdu instance=SDURAC1 host=sduclunode1 status=down reason=USER
timestamp=2021-07-01 12:39:52 timezone=+00:00
db_domain=sub06221433571.skynet.oraclevcn.com  reported = Thu Jul 1 12:39:52 UTC 2021
INSTANCE VERSION=1.0 service=sdurac_dbsdu.sub06221433571.skynet.oraclevcn.com
database=sdurac_dbsdu instance=SDURAC1 host=sduclunode1 status=down reason=USER
timestamp=2021-07-01 12:40:20 timezone=+00:00
db_domain=sub06221433571.skynet.oraclevcn.com  reported = Thu Jul 1 12:40:20 UTC 2021
[grid@sduclunode1 usrco]$

## Now start instance 1 and re-check the logfile:

srvctl start instance -d $(srvctl config database) -i <Name of instance 1>
srvctl status database -d $(srvctl config database)

[grid@sduclunode1 usrco]$ cat /tmp/sduclunode1_events.log
SERVICEMEMBER VERSION=1.0 service=ac_service.sub06221433571.skynet.oraclevcn.com
database=sdurac_dbsdu instance=SDURAC1 host=sduclunode1 status=down reason=USER
timestamp=2021-07-01 12:39:52 timezone=+00:00
db_domain=sub06221433571.skynet.oraclevcn.com  reported = Thu Jul 1 12:39:52 UTC 2021
SERVICE VERSION=1.0 service=ac_service.sub06221433571.skynet.oraclevcn.com
database=sdurac_dbsdu instance=SDURAC1 host=sduclunode1 status=down reason=USER
timestamp=2021-07-01 12:39:52 timezone=+00:00
db_domain=sub06221433571.skynet.oraclevcn.com  reported = Thu Jul 1 12:39:52 UTC 2021
SERVICEMEMBER VERSION=1.0 service=tac_service.sub06221433571.skynet.oraclevcn.com
database=sdurac_dbsdu instance=SDURAC1 host=sduclunode1 status=down reason=USER
timestamp=2021-07-01 12:39:52 timezone=+00:00
db_domain=sub06221433571.skynet.oraclevcn.com  reported = Thu Jul 1 12:39:52 UTC 2021
SERVICE VERSION=1.0 service=tac_service.sub06221433571.skynet.oraclevcn.com
database=sdurac_dbsdu instance=SDURAC1 host=sduclunode1 status=down reason=USER
timestamp=2021-07-01 12:39:52 timezone=+00:00
db_domain=sub06221433571.skynet.oraclevcn.com  reported = Thu Jul 1 12:39:52 UTC 2021
INSTANCE VERSION=1.0 service=sdurac_dbsdu.sub06221433571.skynet.oraclevcn.com
database=sdurac_dbsdu instance=SDURAC1 host=sduclunode1 status=down reason=USER
timestamp=2021-07-01 12:40:20 timezone=+00:00
db_domain=sub06221433571.skynet.oraclevcn.com  reported = Thu Jul 1 12:40:20 UTC 2021
```

```
INSTANCE VERSION=1.0 service=sdurac_dbsdu.sub06221433571.skynet.oraclevcn.com
database=sdurac_dbsdu instance=SDURAC1 host=sduclunode1 status=up reason=USER
timestamp=2021-07-01 12:45:45 timezone=+00:00
db_domain=sub06221433571.skynet.oraclevcn.com  reported = Thu Jul 1 12:45:45 UTC 2021
SERVICEMEMBER VERSION=1.0 service=ac_service.sub06221433571.skynet.oraclevcn.com
database=sdurac_dbsdu instance=SDURAC1 host=sduclunode1 status=up reason=USER card=1
timestamp=2021-07-01 12:45:47 timezone=+00:00
db_domain=sub06221433571.skynet.oraclevcn.com  reported = Thu Jul 1 12:45:47 UTC 2021
SERVICE VERSION=1.0 service=ac_service.sub06221433571.skynet.oraclevcn.com
database=sdurac_dbsdu instance=SDURAC1 host=sduclunode1 status=up reason=USER
timestamp=2021-07-01 12:45:47 timezone=+00:00
db_domain=sub06221433571.skynet.oraclevcn.com  reported = Thu Jul 1 12:45:47 UTC 2021
SERVICEMEMBER VERSION=1.0 service=tac_service.sub06221433571.skynet.oraclevcn.com
database=sdurac_dbsdu instance=SDURAC1 host=sduclunode1 status=up reason=USER card=1
timestamp=2021-07-01 12:45:47 timezone=+00:00
db_domain=sub06221433571.skynet.oraclevcn.com  reported = Thu Jul 1 12:45:47 UTC 2021
SERVICE VERSION=1.0 service=tac_service.sub06221433571.skynet.oraclevcn.com
database=sdurac_dbsdu instance=SDURAC1 host=sduclunode1 status=up reason=USER
timestamp=2021-07-01 12:45:47 timezone=+00:00
db_domain=sub06221433571.skynet.oraclevcn.com  reported = Thu Jul 1 12:45:47 UTC 2021
```

Now we will create a more elaborated callout. Callouts can be any shell-script or executable. There can be multiple callouts in the racg/usrco directory and all will be executed with the FAN payload as arguments.
The scripts are executed sequentially, so it is not recommended to have many scripts in this directory, as they could place a load on the system that is not desired, and there may be timeliness issues if the scripts wait for scheduling.

Create a second shell script on each node, in directory /u01/app/19.0.0.0/grid/racg/usrco/

```
cat callout_elaborate.sh

#!/usr/bin/bash
    # Scan and parse HA event payload arguments:
    #
    # define AWK
    AWK=/bin/awk
    # Define a log file to see results
    FAN_LOGFILE=/tmp/`hostname -s`.log
    # Event type is handled differently
    NOTIFY_EVENTTYPE=$1
    for ARGS in $*; do
        PROPERTY=`echo $ARGS | $AWK -F "=" '{print $1}'`
        VALUE=`echo $ARGS | $AWK -F "=" '{print $2}'`
        case $PROPERTY in
          VERSION|version) NOTIFY_VERSION=$VALUE ;;
          SERVICE|service) NOTIFY_SERVICE=$VALUE ;;
          DATABASE|database) NOTIFY_DATABASE=$VALUE ;;
          INSTANCE|instance) NOTIFY_INSTANCE=$VALUE ;;
          HOST|host) NOTIFY_HOST=$VALUE ;;
          STATUS|status) NOTIFY_STATUS=$VALUE ;;
          REASON|reason) NOTIFY_REASON=$VALUE ;;
          CARD|card) NOTIFY_CARDINALITY=$VALUE ;;
          VIP_IPS|vip_ips) NOTIFY_VIPS=$VALUE ;; #VIP_IPS for public_nw_down
```

```
          TIMESTAMP|timestamp) NOTIFY_LOGDATE=$VALUE ;; # catch event date
          TIMEZONE|timezone) NOTIFY_TZONE=$VALUE ;;
          ??:??:??) NOTIFY_LOGTIME=$PROPERTY ;; # catch event time (hh24:mi:ss)
        esac
    done

    # FAN events with the following conditions will be inserted# into the critical
trouble ticket system:
    # NOTIFY_EVENTTYPE => SERVICE | DATABASE | NODE
    # NOTIFY_STATUS => down | public_nw_down | nodedown
    #
    if (( [ "$NOTIFY_EVENTTYPE" = "SERVICE" ] ||[ "$NOTIFY_EVENTTYPE" = "DATABASE" ] ||
\
        [ "$NOTIFY_EVENTTYPE" = "NODE" ] \
        ) && \
        ( [ "$NOTIFY_STATUS" = "down" ] || \
        [ "$NOTIFY_STATUS" = "public_nw_down" ] || \
        [ "$NOTIFY_STATUS" = "nodedown " ] ) \
        ) ; then
        # << CALL TROUBLE TICKET LOGGING PROGRAM AND PASS RELEVANT NOTIFY_* ARGUMENTS >>
        echo "Create a service request as " ${NOTIFY_EVENTTYPE} " " ${NOTIFY_STATUS} "
occured at " ${NOTIFY_LOGTIME} >> ${FAN_LOGFILE}
    else
        echo "Found no interesting event: " ${NOTIFY_EVENTTYPE} " " ${NOTIFY_STATUS} >>
${FAN_LOGFILE}
    fi

chmod +x callout_elaborate.sh
## Stop the database and check its status:

srvctl stop database -d $(srvctl config database) -o immediate -force
srvctl status database -d $(srvctl config database)

## Review the logfile on /tmp, for example:

[grid@sduclunode1 usrco]$ cat /tmp/sduclunode1_elaborate.log
Found no interesting event:  SERVICEMEMBER   down
Create a service request as  SERVICE   down  occured at  12:59:13
Found no interesting event:  SERVICEMEMBER   down
Create a service request as  SERVICE   down  occured at  12:59:13
Found no interesting event:  INSTANCE   down
Create a service request as  DATABASE   down  occured at  12:59:49

## Review the logfile on node 2:

[grid@sduclunode2 usrco]$ cat /tmp/sduclunode2_elaborate.log
Found no interesting event:  SERVICEMEMBER   down
Create a service request as  SERVICE   down  occured at  12:59:13
Found no interesting event:  INSTANCE   down

## Start the database and check the logs on both nodes

srvctl start database -d $(srvctl config database)
srvctl status database -d $(srvctl config database)

[grid@sduclunode1 usrco]$ cat /tmp/sduclunode1_elaborate.log
Found no interesting event:  SERVICEMEMBER   down
Create a service request as  SERVICE   down  occured at  12:59:13
```

```
Found no interesting event:  SERVICEMEMBER   down
Create a service request as  SERVICE   down  occured at  12:59:13
Found no interesting event:  INSTANCE   down
Create a service request as  DATABASE   down  occured at  12:59:49
Found no interesting event:  INSTANCE   up

[grid@sduclunode2 usrco]$ cat /tmp/sduclunode2_elaborate.log
Found no interesting event:  SERVICEMEMBER   down
Create a service request as  SERVICE   down  occured at  12:59:13
Found no interesting event:  INSTANCE   down
Found no interesting event:  INSTANCE   up
Found no interesting event:  DATABASE   up
Found no interesting event:  SERVICEMEMBER   up
Found no interesting event:  SERVICE   up
Found no interesting event:  SERVICE   up
Found no interesting event:  SERVICEMEMBER   up
Found no interesting event:  SERVICEMEMBER   up
Found no interesting event:  SERVICE   up
```

Client side FAN events

FAN events are sent to the application mid-tier or client tier using the Oracle Notification Service (ONS). ONS is configured automatically on the cluster when you install Grid Infrastructure.
CRS manages the stop and start of the ONS daemon.

ONS is configured automatically by FAN-aware Oracle clients, which include Universal Connection Pool (UCP), ODP.Net, Weblogic Server with Active Gridlink, CMAN and others, when a particular format connect string is used (for more information on this refer to the Application Continuity checklist: https://www.oracle.com//technetwork/database/clustering/checklist-ac-6676160.pdf)

Determining if a client has received FAN events may require running your client in a debug fashion. This may be difficult to do and even more difficult to interpret.

To confirm that FAN events are being received at a particular tier, you can install a java utility called FANWatcher, that will subscribe to ONS on a cluster and display events that it receives.

We will install FANWatcher on node 1: connect to node 1 as user "oracle", and run the following steps:

```
mkdir -p /home/oracle/fANWatcher

cd /home/oracle/fANWatcher

## Download FANWatcher application:

wget https://objectstorage.uk-london-
1.oraclecloud.com/p/gKfwKKgzqSfL4A48e6lSKZYqyFdDzvu57md4B1MegMU/n/lrojildid9yx/b/labtest
_bucket/o/fanWatcher_19c.zip
--2021-12-13 11:37:17--  https://objectstorage.uk-london-
1.oraclecloud.com/p/gKfwKKgzqSfL4A48e6lSKZYqyFdDzvu57md4B1MegMU/n/lrojildid9yx/b/labtest
_bucket/o/fanWatcher_19c.zip
```

```
Resolving objectstorage.uk-london-1.oraclecloud.com (objectstorage.uk-london-
1.oraclecloud.com)... 134.70.64.1, 134.70.56.1, 134.70.60.1
Connecting to objectstorage.uk-london-1.oraclecloud.com (objectstorage.uk-london-
1.oraclecloud.com)|134.70.64.1|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 6116 (6.0K) [application/x-zip-compressed]
Saving to: 'fanWatcher_19c.zip'

100%[=======================================================================================
===========>] 6,116       --.-K/s   in 0s

2021-12-13 11:37:17 (47.6 MB/s) - 'fanWatcher_19c.zip' saved [6116/6116]

## Unzip the application:

[oracle@sduclunode1 fANWatcher]$ unzip fanWatcher_19c.zip
Archive:  fanWatcher_19c.zip
  inflating: fanWatcher.bash
  inflating: fanWatcher.class
  inflating: fanWatcher.java
[oracle@sduclunode1 fANWatcher]$ ls -ltr
total 28
-rw-r--r-- 1 oracle oinstall 6416 Jun 24  2020 fanWatcher.java
-rw-r--r-- 1 oracle oinstall 5733 Jun 24  2020 fanWatcher.class
-rw-r--r-- 1 oracle oinstall 6116 Aug 18  2020 fanWatcher_19c.zip
-rw-r--r-- 1 oracle oinstall  905 Aug 18  2020 fanWatcher.bash
[oracle@sduclunode1 fANWatcher]$

## Edit the fanWatcher.bash file, and change/check "User", "Password", "Url"
,"ORACLE_HOME" and "CLASSPATH" accordingly, for example:

[oracle@sduclunode1 fANWatcher]$ cat fanWatcher.bash
#!/usr/bin/bash
ORACLE_HOME=/u01/app/oracle/product/19.0.0.0/dbhome_1
JAVA_HOME=${ORACLE_HOME}/jdk
export ORACLE_HOME
export JAVA_HOME
# Set the credentials in the environment. If you don't like doing this,
# hardcode them into the java program
# Edit the values for password, url, user and CLASSPATH
password=XXXXX
url='jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=sduclunod
e1.sub06221433571.skynet.oraclevcn.com)(PORT=1521))(ADDRESS=(PROTOCOL=TCP)(HOST=sducluno
de2.sub06221433571.skynet.oraclevcn.com)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=noac.su
b06221433571.skynet.oraclevcn.com)))'
user=hr
export password url user
CLASSPATH="/u01/app/oracle/product/19.0.0.0/dbhome_1/jdbc/lib/ojdbc8.jar:/u01/app/oracle
/product/19.0.0.0/dbhome_1/opmn/lib/ons.jar:."
export CLASSPATH

# Compile fanWatcher with the exported classpath
#javac fanWatcher.java

# Run fanwatcher with autoons
${JAVA_HOME}/jre/bin/java fanWatcher autoons
# EOF
```

```
## Then chmod the file, and execute:

chmod +x fanWatcher.bash

[oracle@sduclunode1 fANWatcher]$ ./fanWatcher.bash
Auto-ONS configuration=maxconnections.0001=0003
nodes.0001=SDUCLUNODE1.SUB06221433571.SKYNET.ORACLEVCN.COM:6200
maxconnections.0002=0003
nodes.0002=SDUCLUNODE2.SUB06221433571.SKYNET.ORACLEVCN.COM:6200
Opening FAN Subscriber Window ...

## From another terminal, connect to node 2 as "oracle" and kill smon process of
instance 2 !!!

[oracle@sduclunode2 ~]$ ps -ef | grep smon
oracle   18581     1  0 13:02 ?        00:00:00 ora_smon_SDURAC2
root     41364     1  1 Jun30 ?        00:20:15 /u01/app/19.0.0.0/grid/bin/osysmond.bin
oracle   42272 42106  0 13:26 pts/0    00:00:00 grep --color=auto smon
grid     46853     1  0 Jun30 ?        00:00:01 asm_smon_+ASM2
[oracle@sduclunode2 ~]$ kill -9 18581

## Go back to FANWatcher terminal, and observe how it traps the events:

[...]
** Event Header **
Notification Type: database/event/service
Delivery Time: Thu Jul 01 13:26:53 UTC 2021
Creation Time: Thu Jul 01 13:26:53 UTC 2021
Generating Node: sduclunode2
Event payload:
VERSION=1.0 event_type=SERVICEMEMBER
service=tac_service.sub06221433571.skynet.oraclevcn.com instance=SDURAC2
database=sdurac_dbsdu db_domain=sub06221433571.skynet.oraclevcn.com host=sduclunode2
status=down reason=FAILURE timestamp=2021-07-01 13:26:53 timezone=+00:00
```

If fanWatcher can auto-configure with ONS and receive and display events, so can any client on the same tier.

This concludes the RAC workshop.