Below is the Data Guard (DG) configuration for this document.

Primary Host name is called primary @ 10.0.XX.XXX

Standby Host name is called standby @ 10.0.XX.XXX

ORACLE SID is called TWSKAPD

Primary database name/unique name will be TWSKAPD\_P

Standby database name/unique name will be TWSKAPD\_S

Please note how they are in different subnet/AWS AZ. This script is very interactive in nature so MAKE SURE TO READ ALL THE OUTPUT.

Table of Contents

[Building Standby Database 2](#_Toc442970106)

[Data Guard Broker Setup 17](#_Toc442970107)

[Performing SWITCHOVER from Primary to standby 20](#_Toc442970108)

[Performing Switch back from standby to primary 25](#_Toc442970109)

[REBUILDING STANDBY DATABASE 29](#_Toc442970110)

[Performing Failover 38](#_Toc442970111)

[Reinstate database after Failover 43](#_Toc442970112)

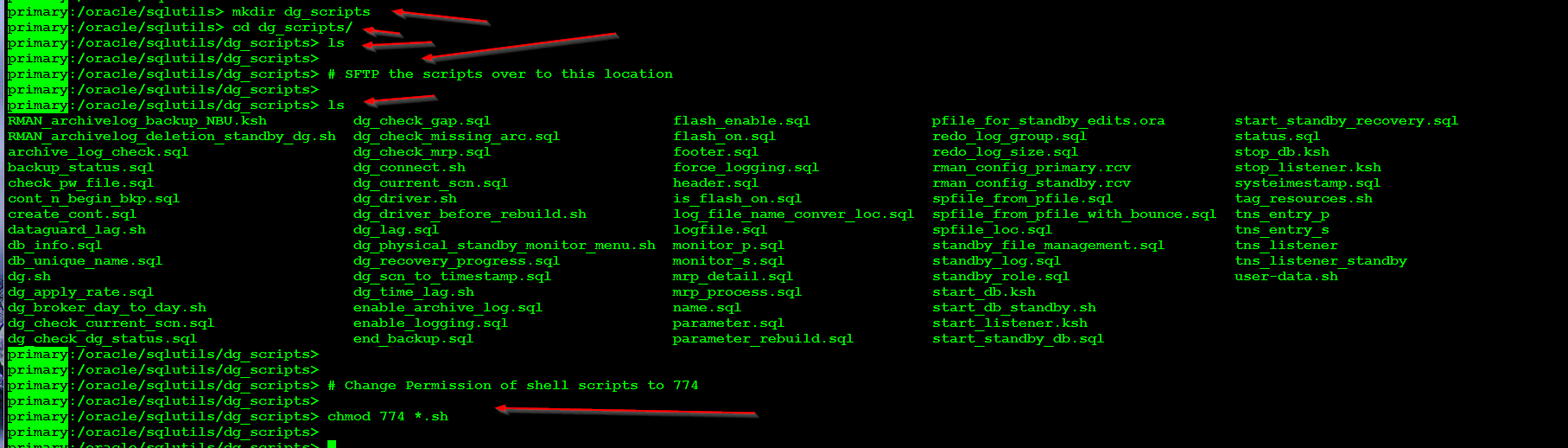
# Building Standby Database

Prerequisites –

* Copy scripts located at below location to /oracle/sqlutils/dg\_scripts. If dg\_scripts does not exist please create that directory
* Make sure standby server is NOT already pre-created for you. This script will take care of creating the standby database
* Make sure the standby database IP address/hostname is in DNS. So we can do a nslookup and ping to make sure DNS entries work and ping does not work(as server will be created by us).
* Work with app team to shut down the app services, as we will need to bounce the server for initial build of DG.
* Although the script checks to make sure /oracle/<SID>/standbylogs and /oracle/<SID>/flashrecovery are mount points and standbylogs (only) have enough storage but make sure to do a manual check.
* This script will dynamically generated 2 script(one to create AMI and one to create secondary server), so please make sure to have either sudo privileges or have someone ready to run script as ROOT user.
* Make sure to have the sys password handy before you start this process
* Secondary hostname has to start with es1aws\*, in this case I bypass that and use secondary host name as STANDBY, this will NOT will under normal circumstances.
* The script will only work for initial DG build, if you have more than one database on the host it will work for the first one and for the rest you have to do it manually. This can be added to the script at a later point if required. As most of our production database have only one DB per host this work well.
* For a successful data guard creation run through below steps from the table of contents
  + Building Standby Database
  + Data Guard Broker Setup
  + Performing SWITCHOVER from Primary to standby
  + Performing Switch back from standby to primary

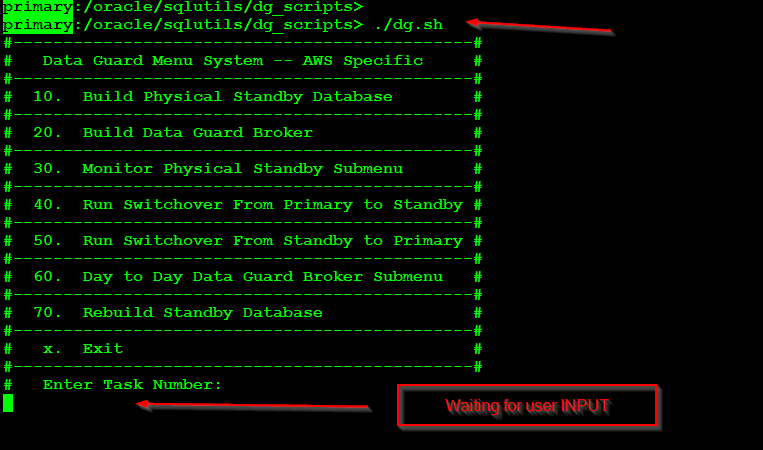
Let’s start the build by first copying scripts over to /oracle/sqlutils/dg\_scripts directory on primary host.

* Start by creating dg\_scripts directory under /oracle/sqlutils if one does not exist.
* SFTP the files to your local host
* change permission on shell script to 774(chmod 774 \*.sh)

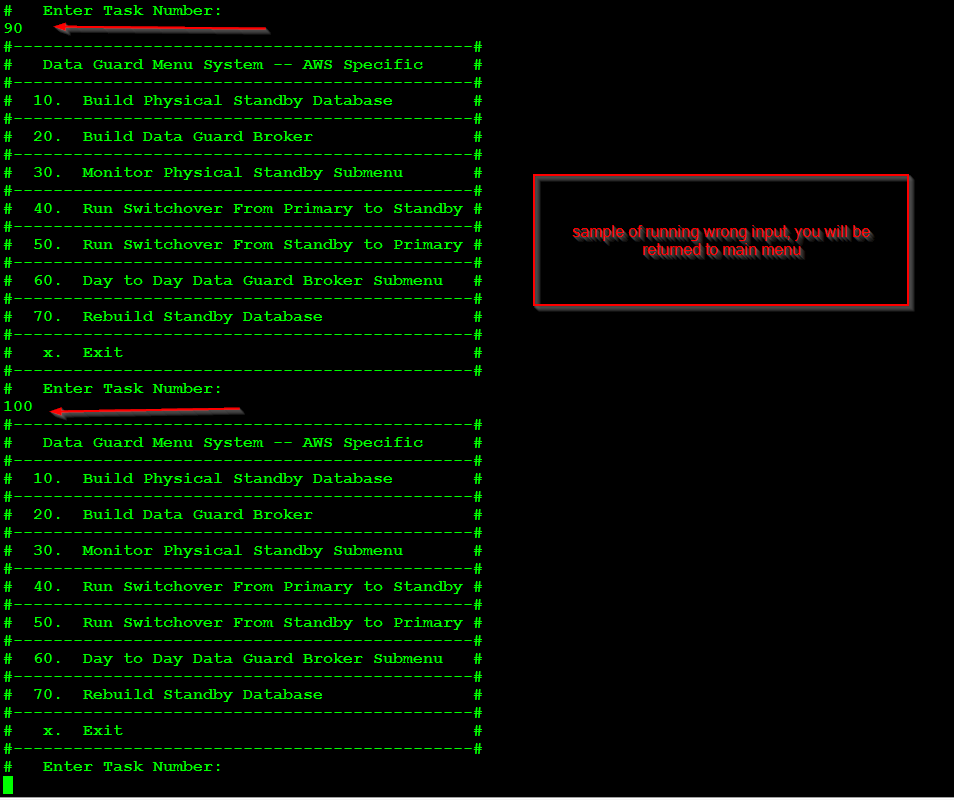


The main script is called dg\_driver.sh but this is in turn called by dg.sh so we can capture the output to a file using tee command.

Let’s start by calling dg.sh as shown below and as you can see its gives you below options

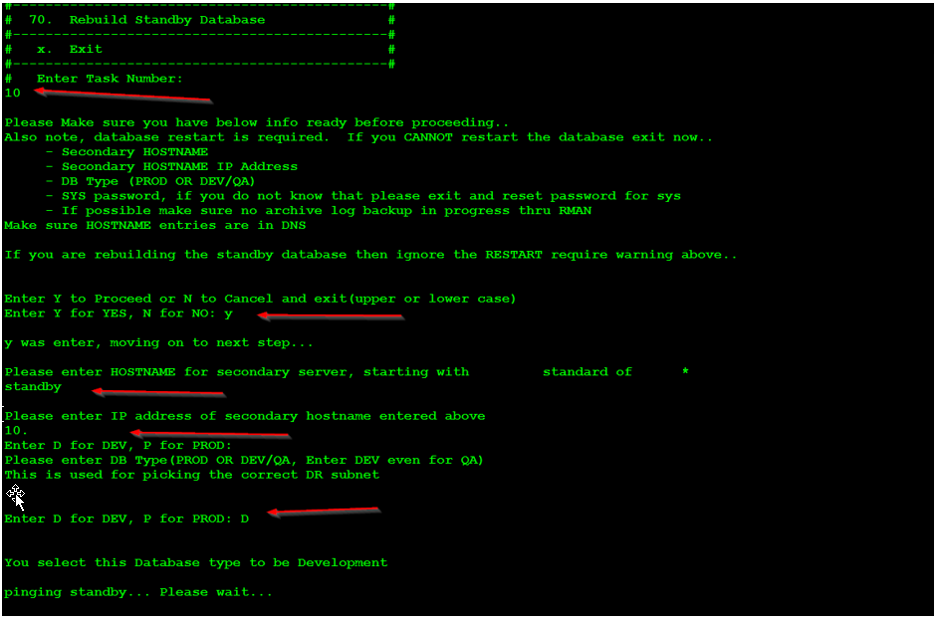


In the above screenshot if you put in an invalid number like 90 or 100(show below) you are returned to this main menu.



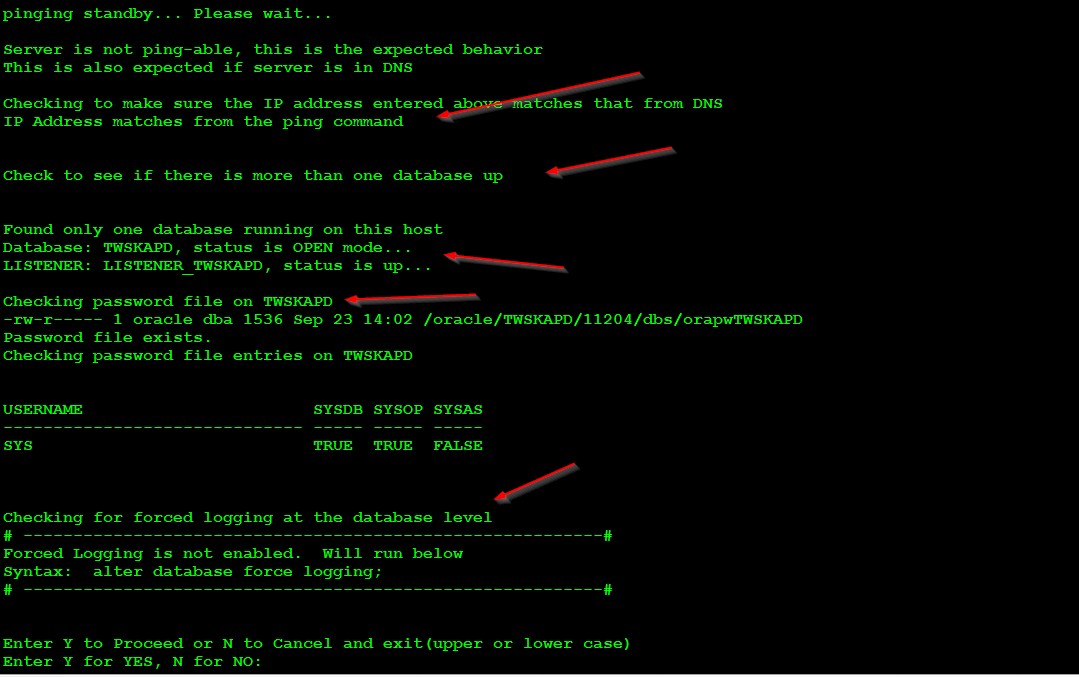
Let’s start DG build. Enter # 10, followed by entering below info

* A confirmation of “y” to make sure you are ready
* secondary hostname, standby in this case but it should be es1aws\*(I by pass this to the sake of making this document clear)
* ip address of the standby hostname
* D or P for dev/prod, this is used for selecting the proper subnet in AWS

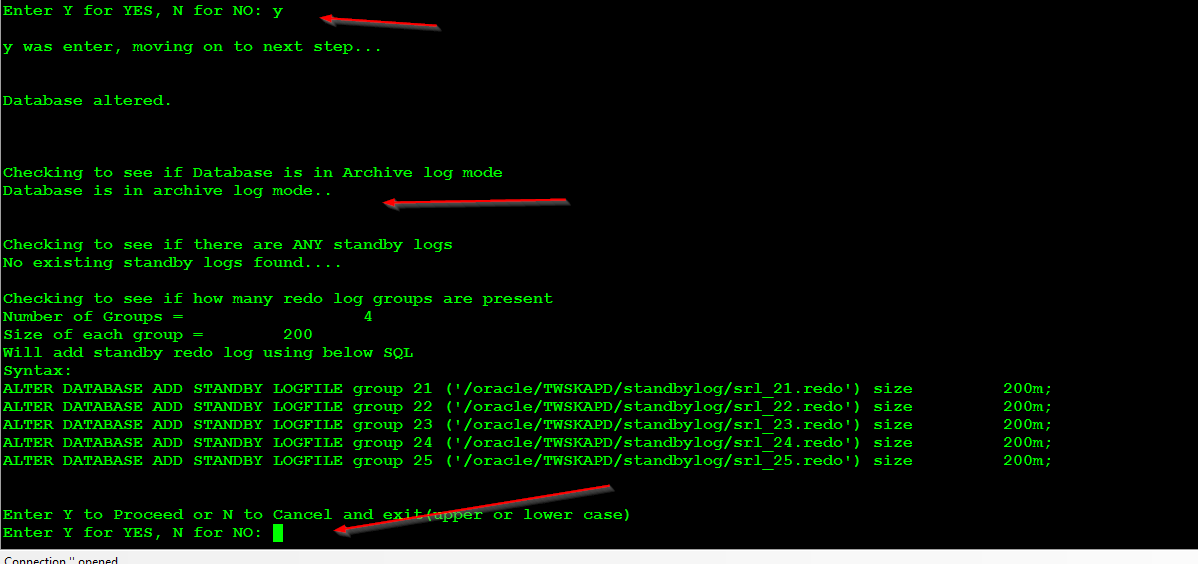


Once the checks passes it moves on the see if

* Ping status retuned proper exit code, that is to not pingable
* Next we check if there is more than one database on the host, if there is we prompt to ask which DB we want to setup DG for.
* We check if DB is up and running and same with listener
* Next we check to see if password file exist or not, if it does not we exit and ask user to manually create password file and start this again.
* Next we check if force logging is turned on or not, in this case it is not so we are asking for a confirmation. Enter “Y”

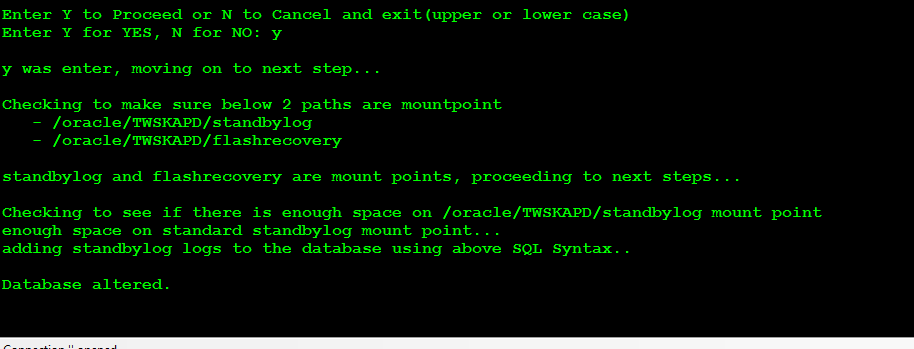


* Enter Y to enable force logging
* Next we check if archive log is turned on or not, if it is not we will ask for confirmation to start it
* Next we check if there any standby logs in place, if we do then we ask to check if we can reuse those. In this case we don’t have any standby logs so we calculate the size of the new logs and shows the SQL syntax that we can run to create those logs. Enter “Y” to proceed.

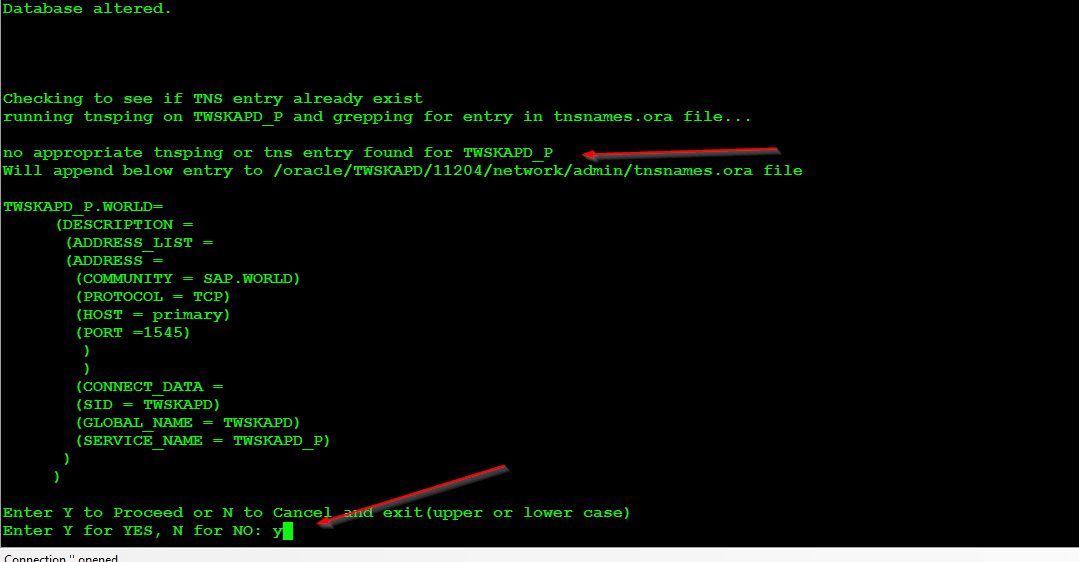


Once we enter “Y” it checks

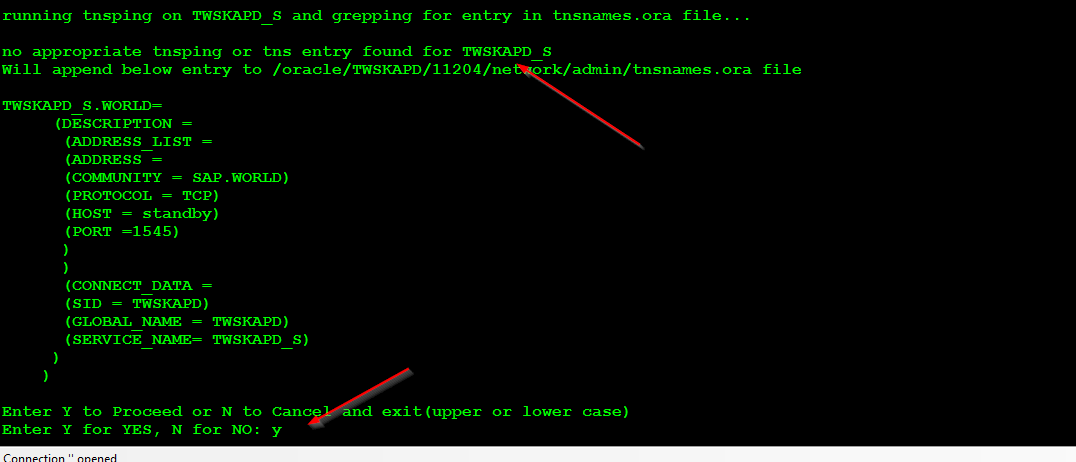
* If standbylog and flashrecovery are actual mount points and not just a directory
* Checks to see if we have standbylogs have enough storage



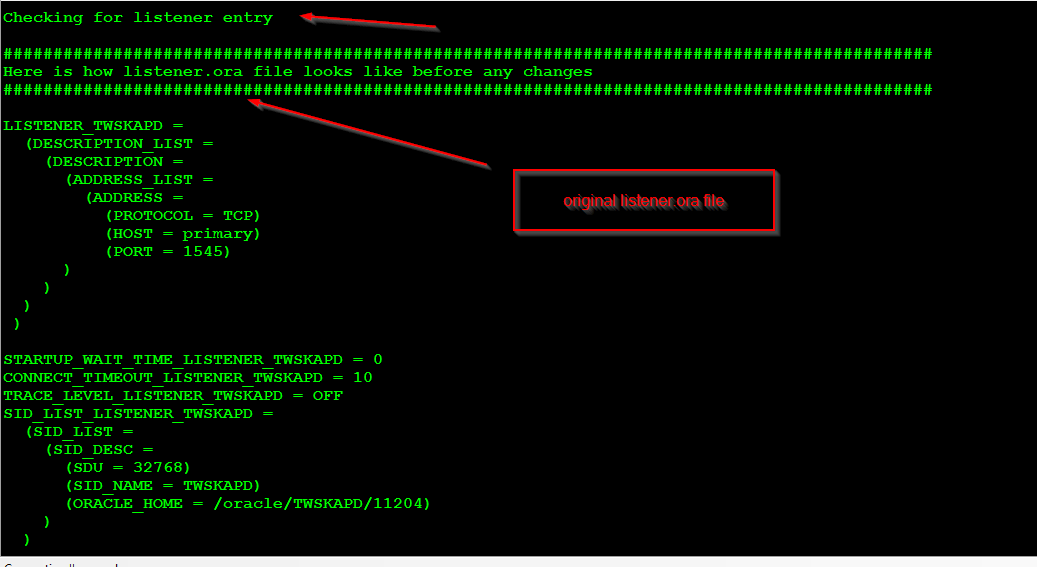
Next we check to see if TWSKAPD\_P entries exist in tnsnames.ora file. Here there is no entry so it’s asking for confirmation to add those entries to tnsnames.ora file.



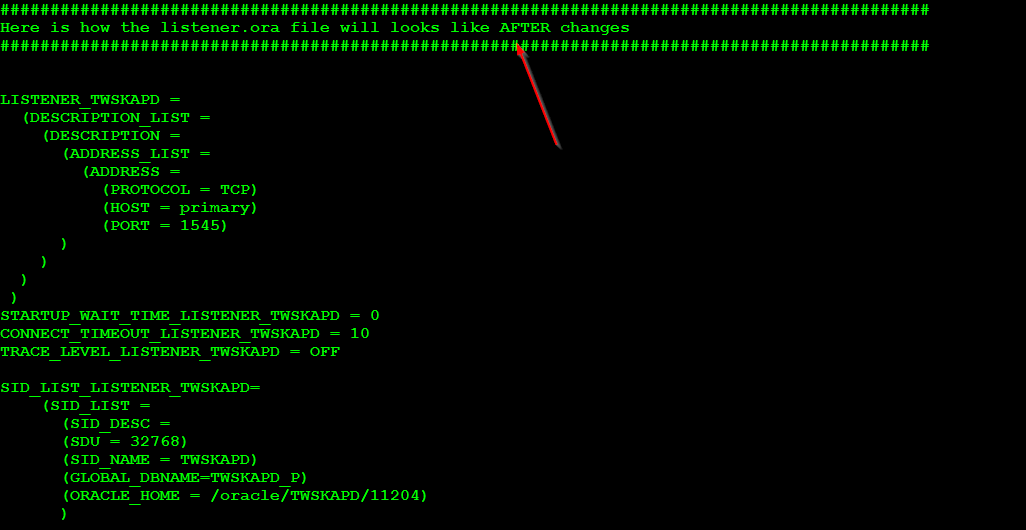
Same thing as above but this time with TWSKAPD\_S(standby entries)

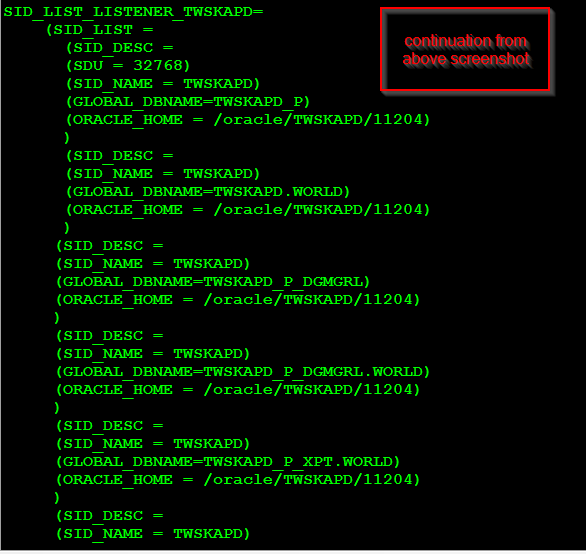


Next we add listener.ora entries. First we display the original listener.ora file followed by new entries that will be added. You will have to scroll up to see all the before/after entries.



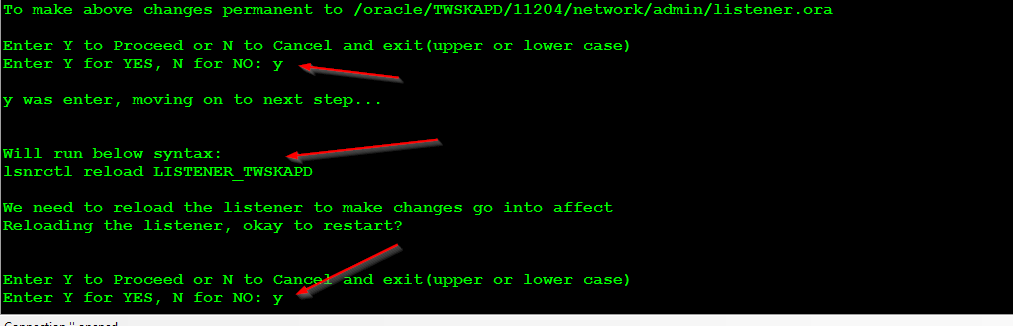
Above show how listener.ora file looks like and then below screen we show after the changes.



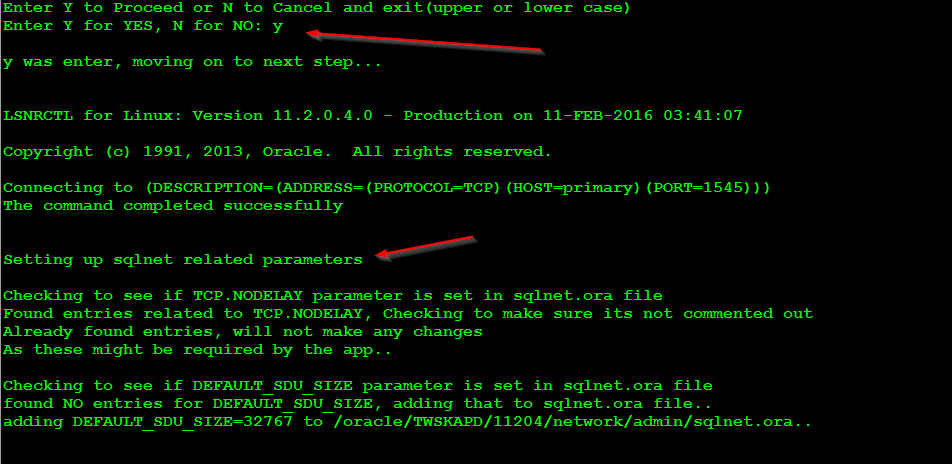




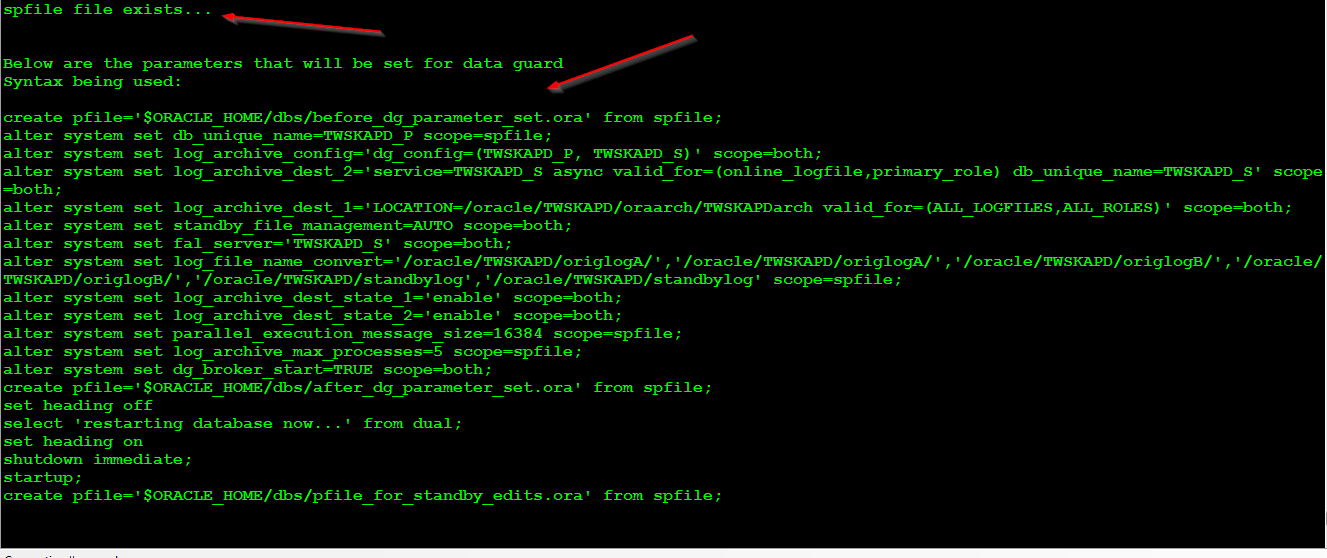
Enter Y to proceed to next step… and enter Y again to reload the listener to get changes in affect.

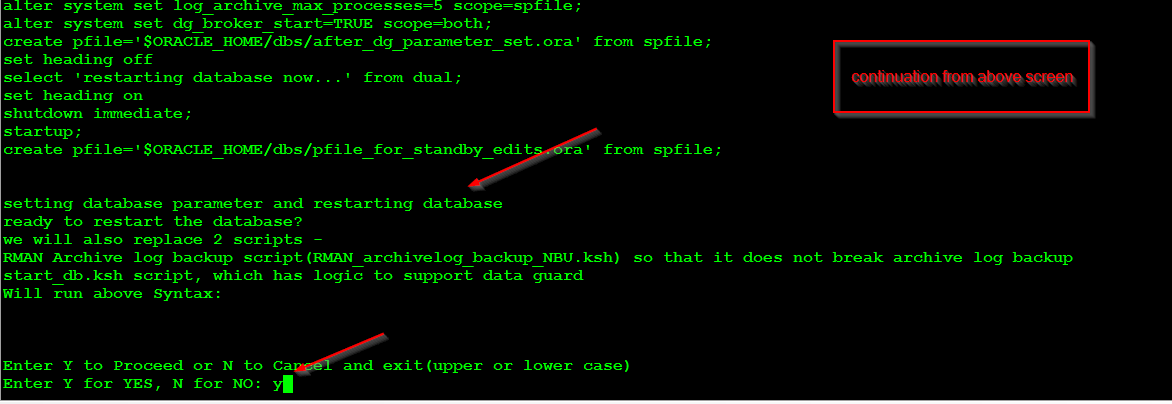


After the listener was reloaded it checks entries in sqlnet.ora file, there is nothing we need to do here. It just adds entries if none found, this is a best practice for DG setup.

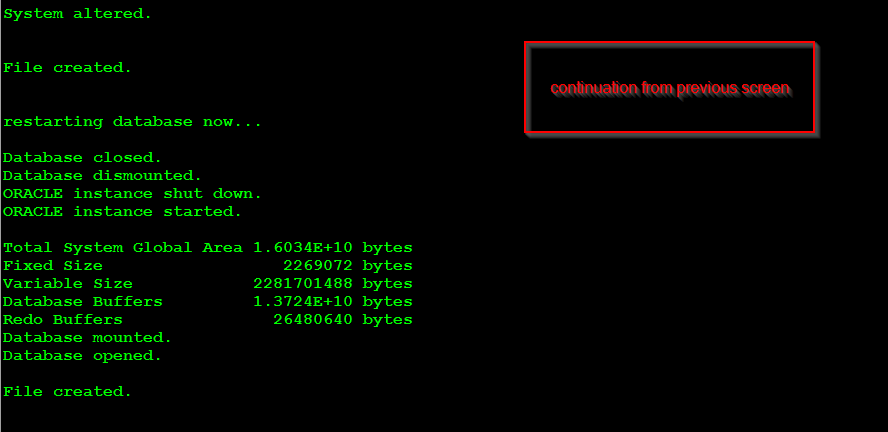


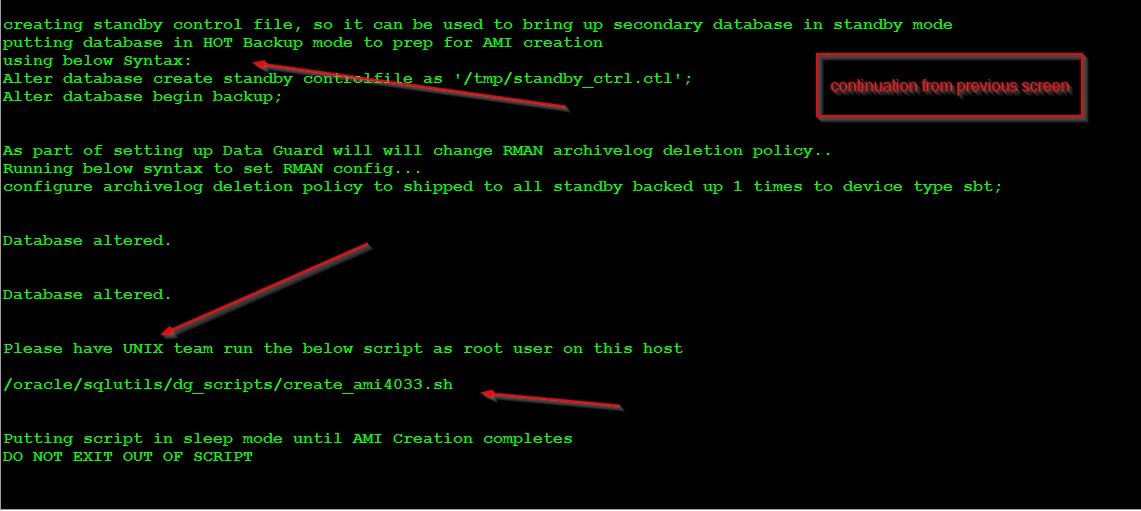
Next we check if SPFILE is in use, if not we create and then we add below listed parameters…





Once we enter “y” it will set all the above parameter and restart the database.





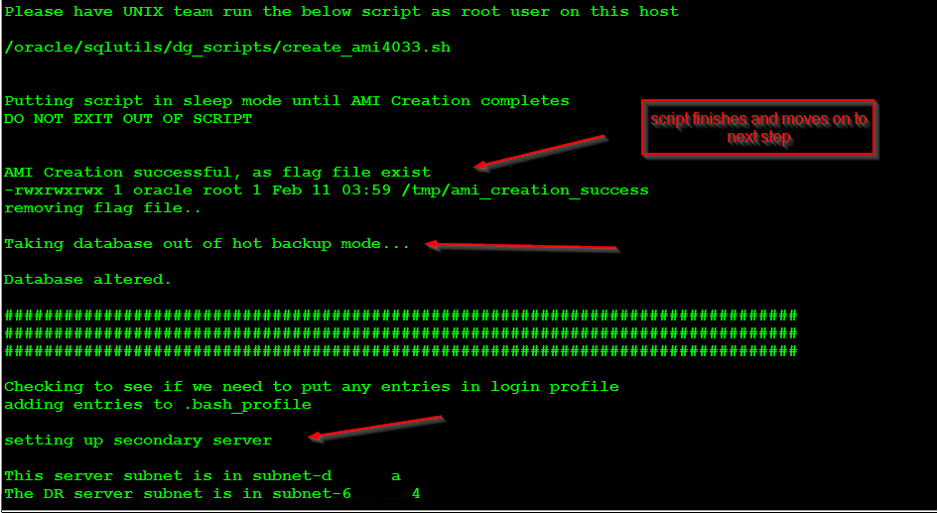
As part of the config, we put the database in hot backup mode and also create standby control file.

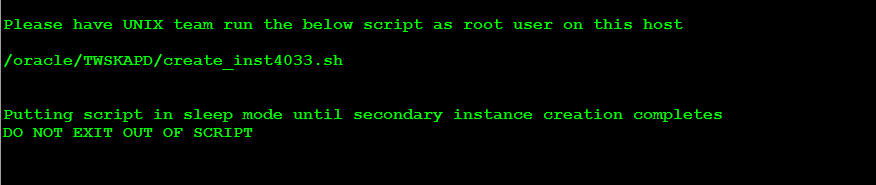
Have unix team run the below script as ROOT user on primary host.

/oracle/sqlutils/dg\_scripts/create\_ami4033.sh

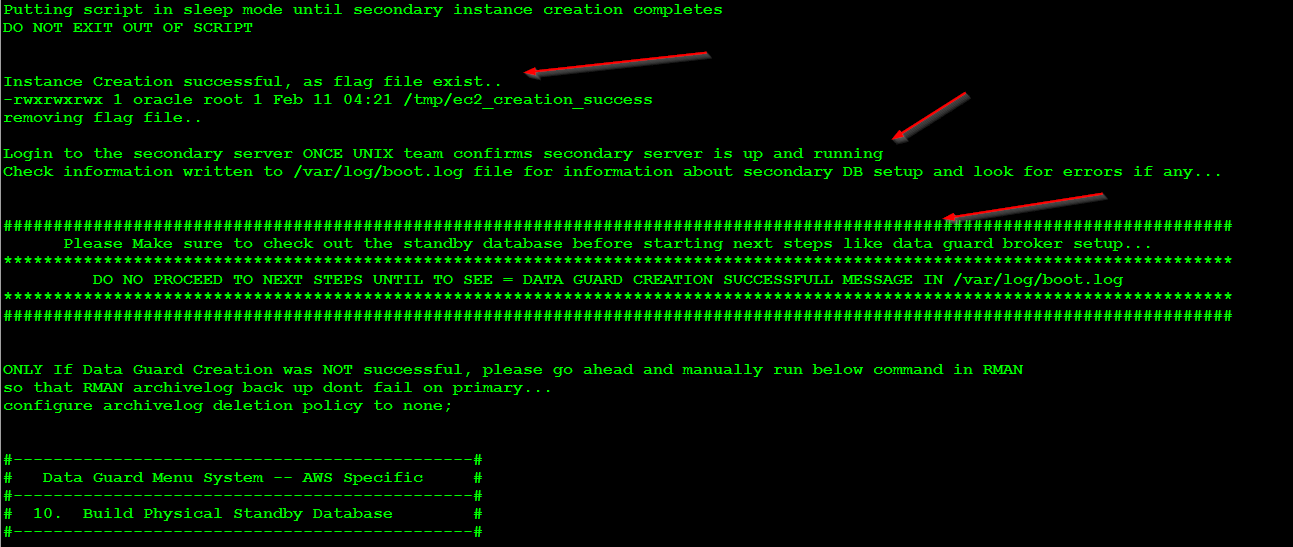
While UNIX team is running the above script “DO NOT EXIT”, the script is place in sleep mode until the root script finishes…

Once the above script finishes, it automatically goes to next step which spits out another script that we need to run as ROOT user. So pass that script to root user (show below).

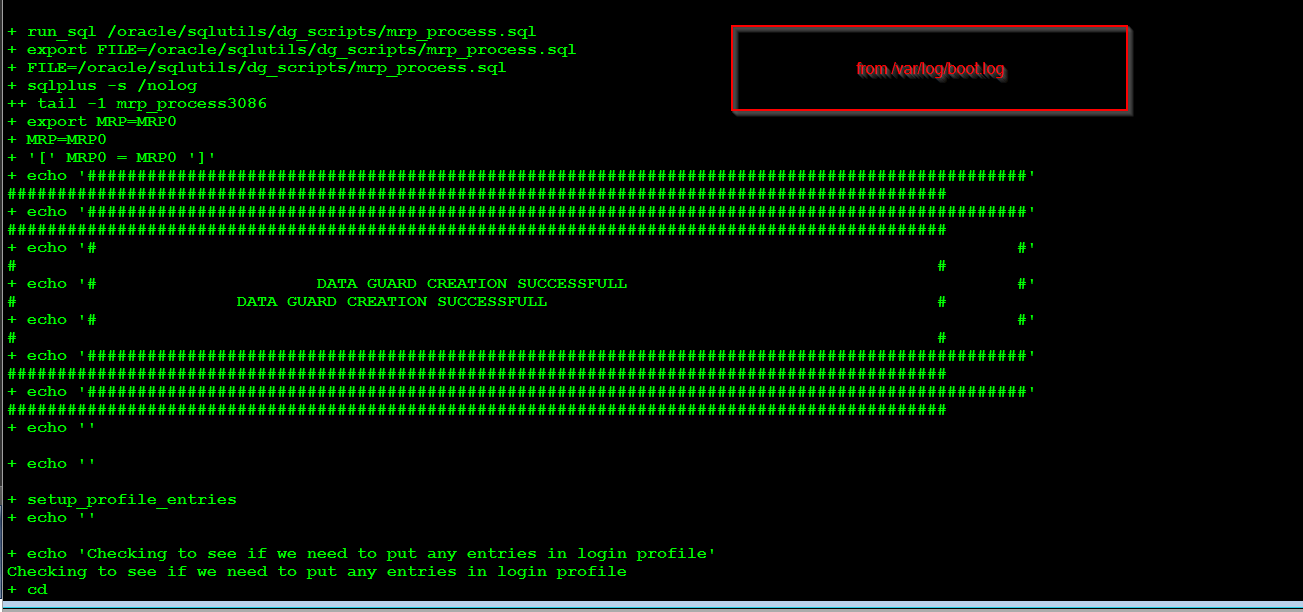




Have unix team run the second script as shown above, The script goes into sleep mode until root script finishes.



As mentioned in the above screen shot, login to secondary host (standby) and look at /var/log/boot.log to see if there is a successful message, shown below…

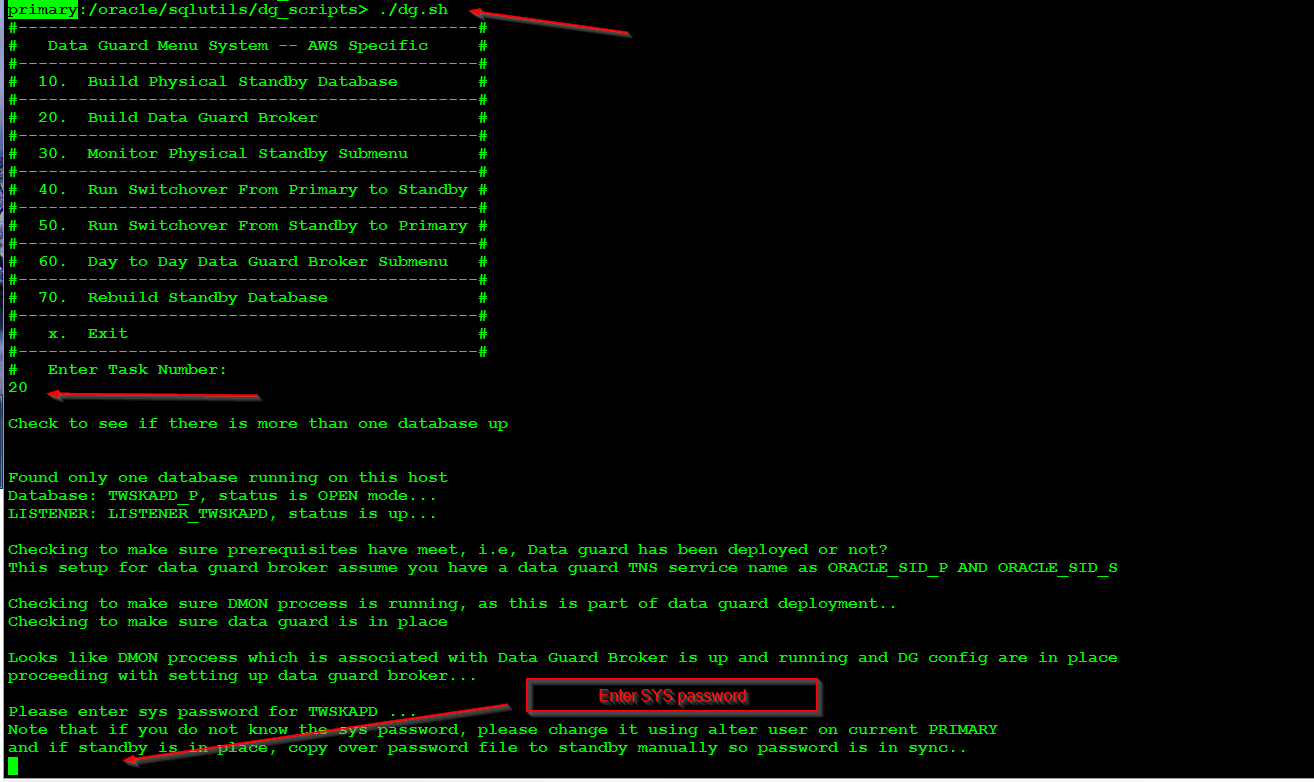


**Only when the above message appears in the standby hosts /var/log/boot.log move to next step**

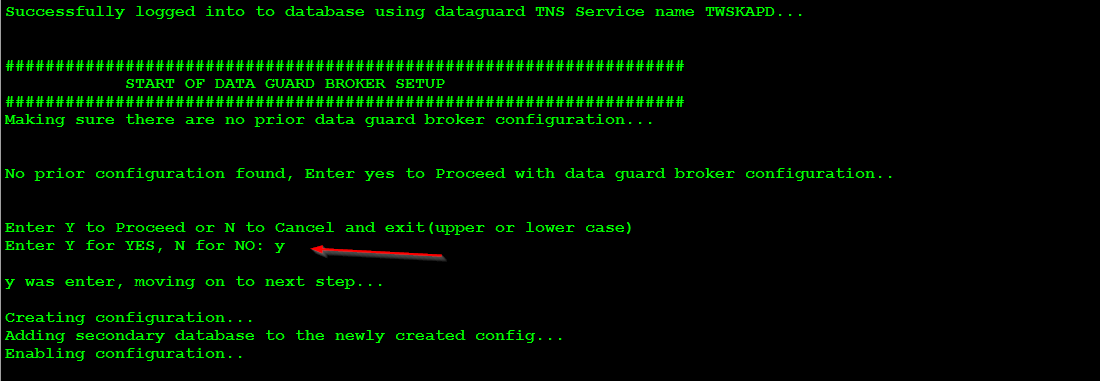
# Data Guard Broker Setup

Now back on the primary host… we need to setup data guard broker.

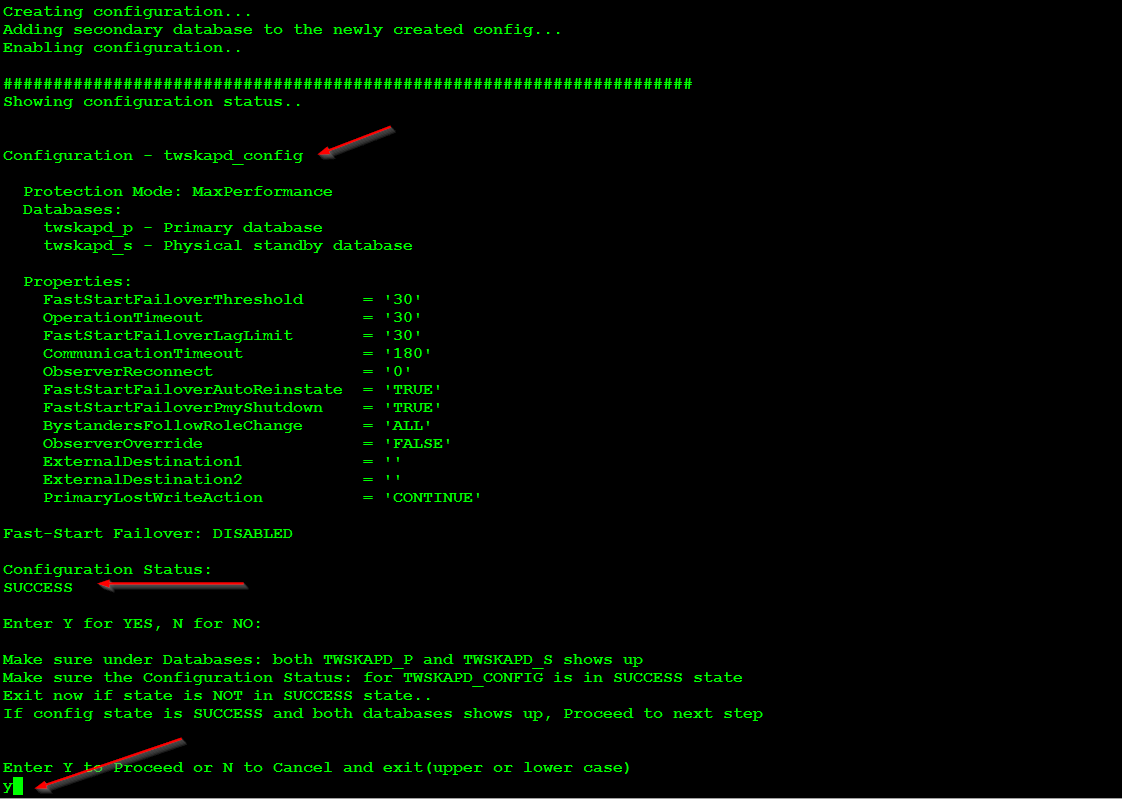
* Enter option #20
* Then enter sys password for the database and hit enter



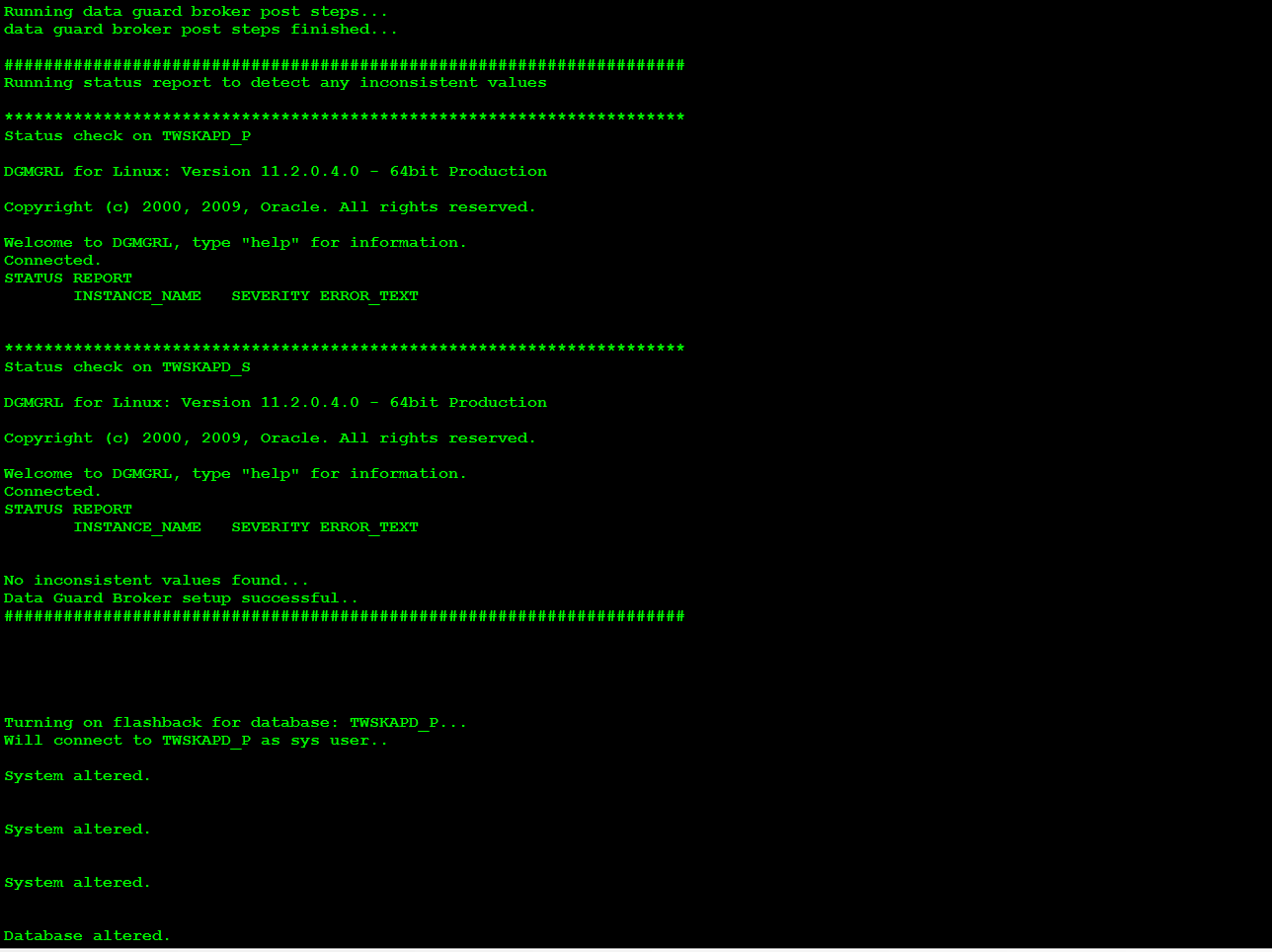
Once sys password is entered it will check to see if we can access the database with correct password. Hit “Y” to continue on

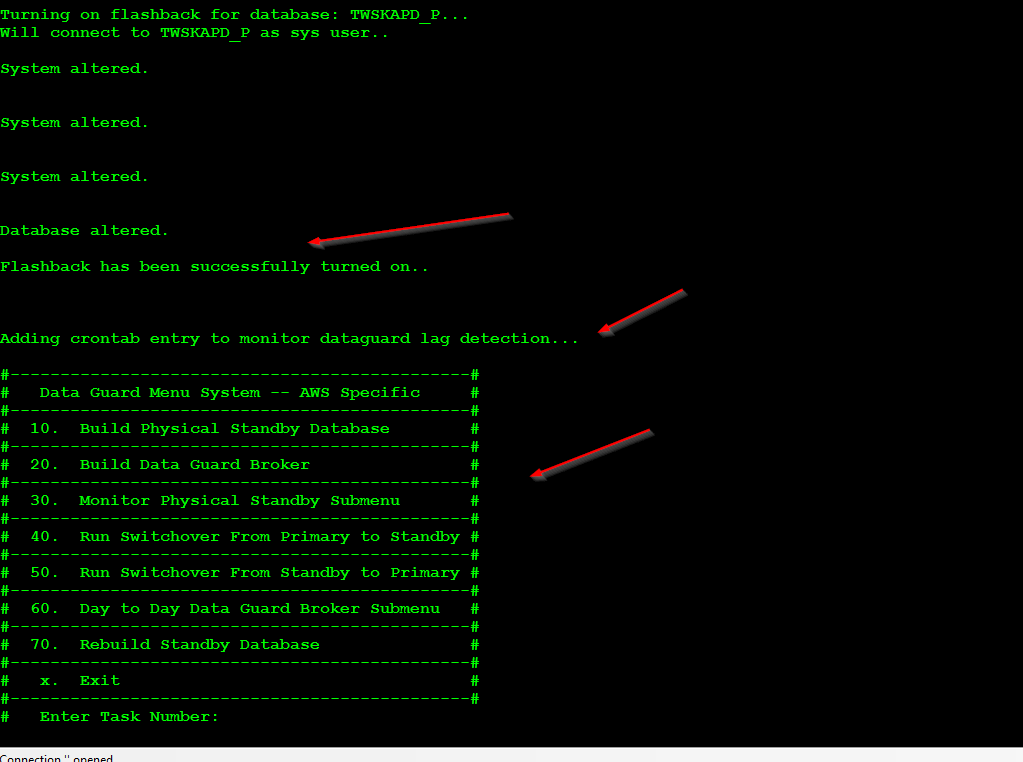


Once the data guard broker configuration is setup, we have to enter “y” again to run some post steps.



Post steps being ran for DG broker setup. As part of it we turn on flash back for primary data and add cron entry to monitoring DG setup. Once that is done we are back to main screen.

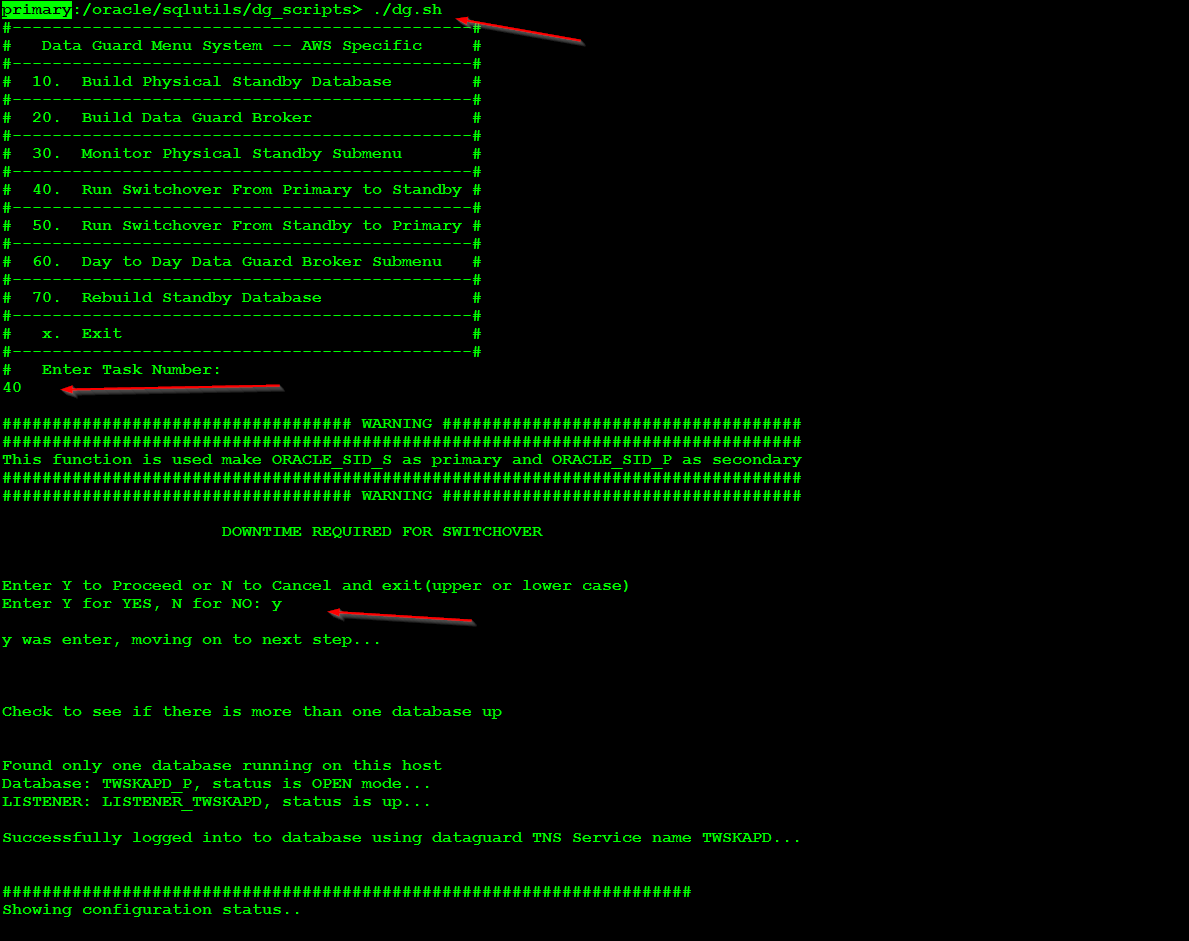


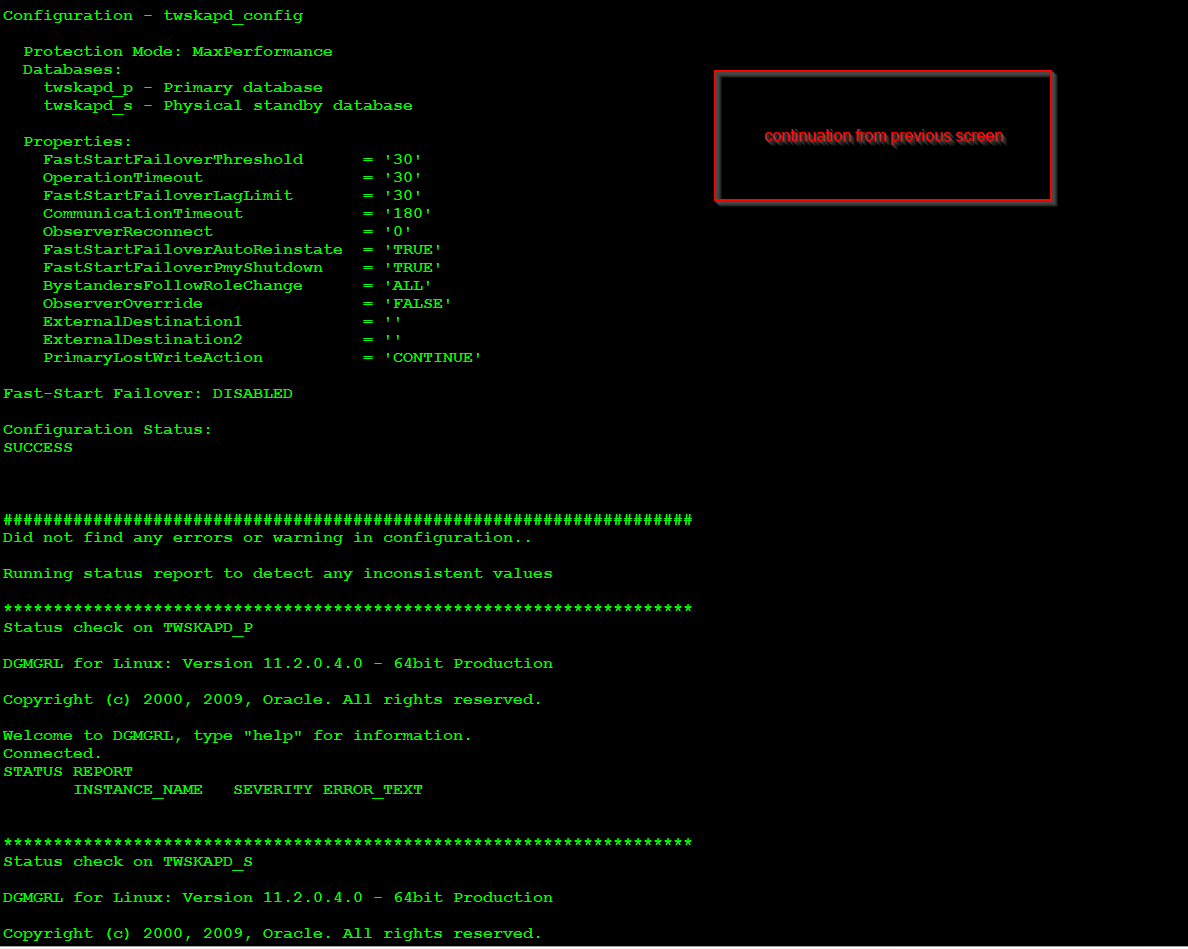


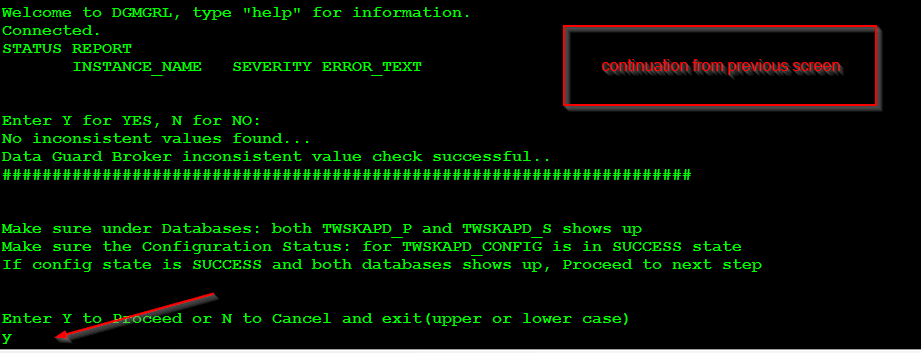
# Performing SWITCHOVER from Primary to standby

Next we will test switchover from primary (TWSKAPD\_P) TO standby (TWSKAPD\_S). We can do this directly in data guard broker or use the script option # 40(this has some built in checks before performing switchover).

Enter option# 40, and enter Y to proceed. It runs some pre-checks and makes sure configuration status for data guard broker is correct.



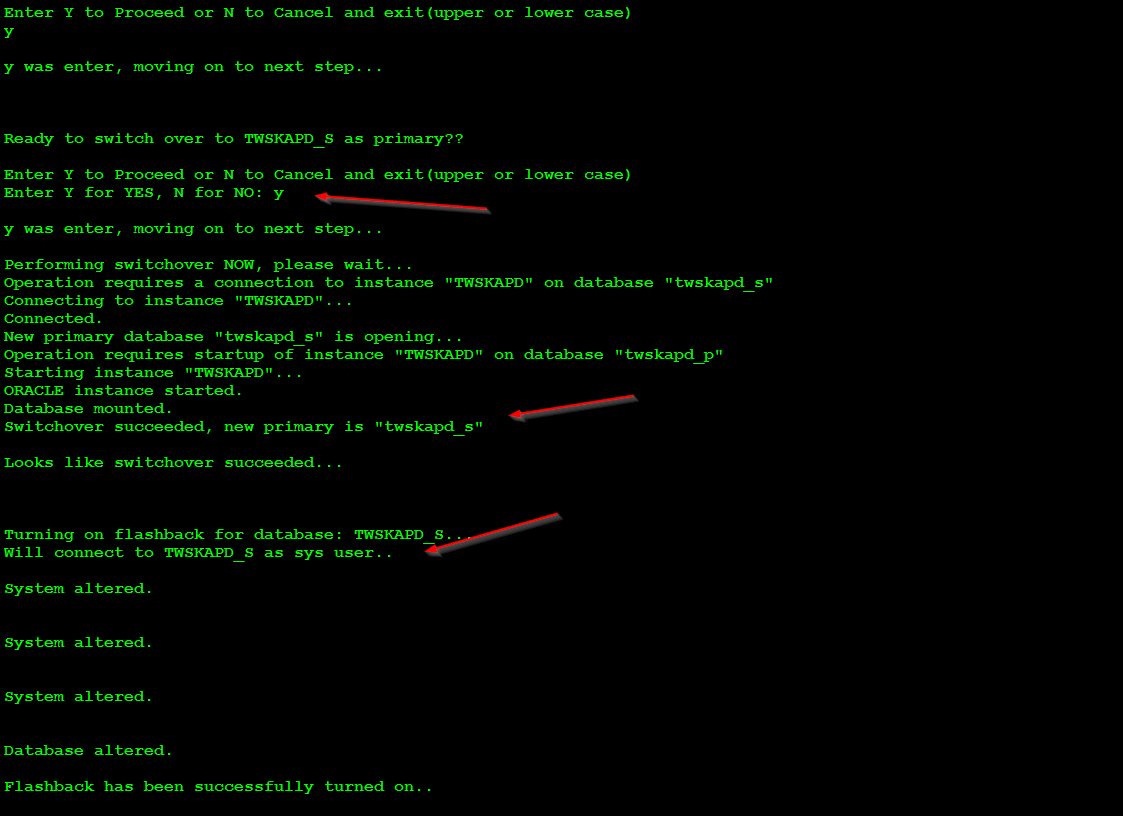




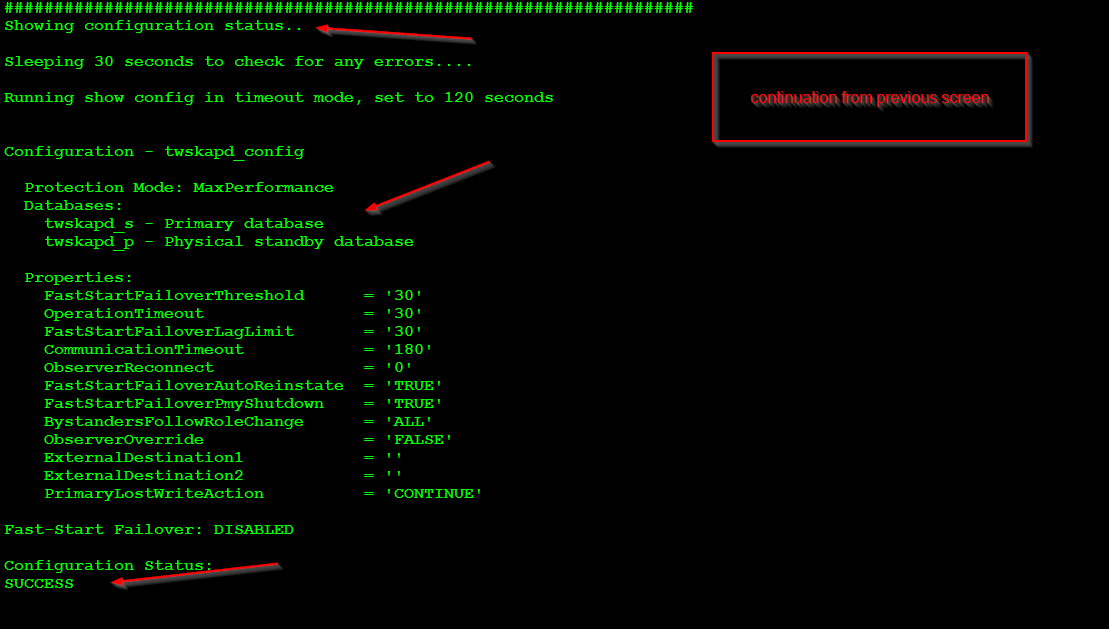
Script did not find any errors or inconsistent values, so enter Y to perform switchover.

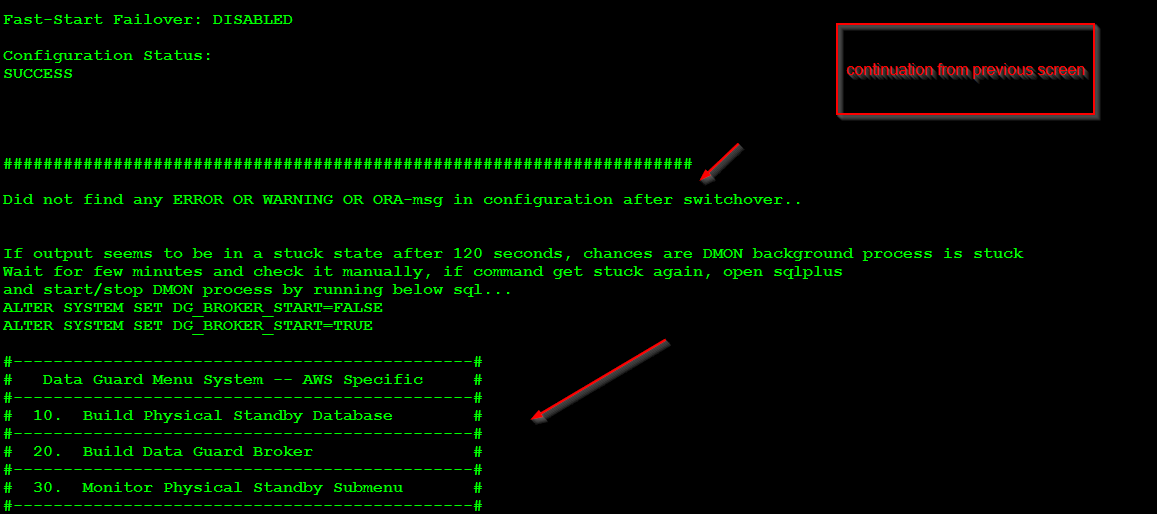
We ask for confirmation AGAIN, just to make sure.

You can see it switchover from primary to standby. And as part of switch over we check if flashback is turned on standby (now the primary), if it is not turned on we turn it back on.



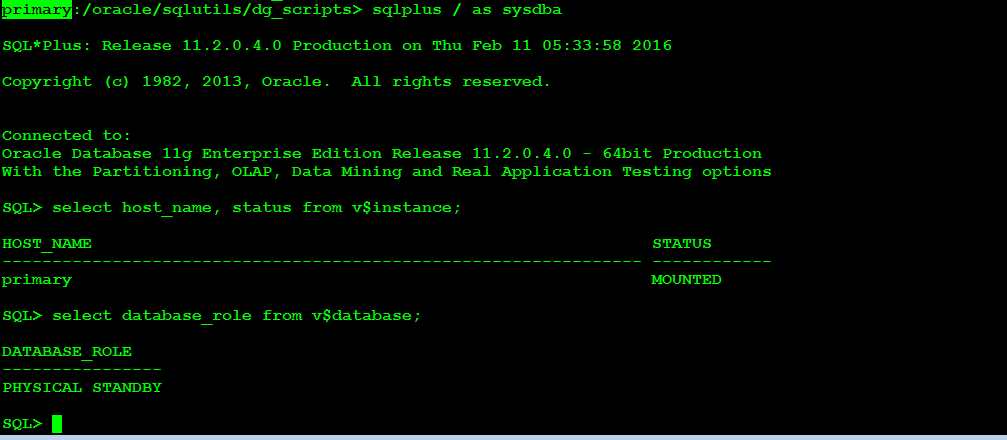
We also show config status again followed by checking for any errors or warning msg. Once that is done we go back to main menu.



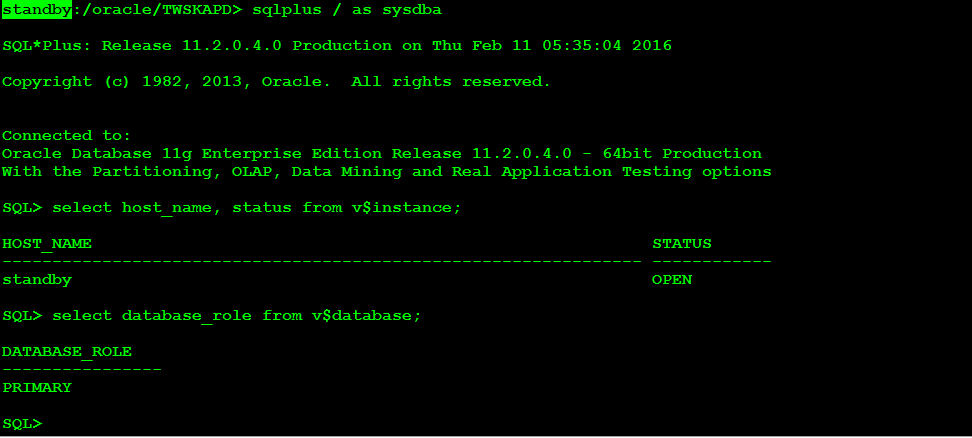


We can confirm the switchover happened successfully by logging into the database at sqlplus …

From primary (now the standby)

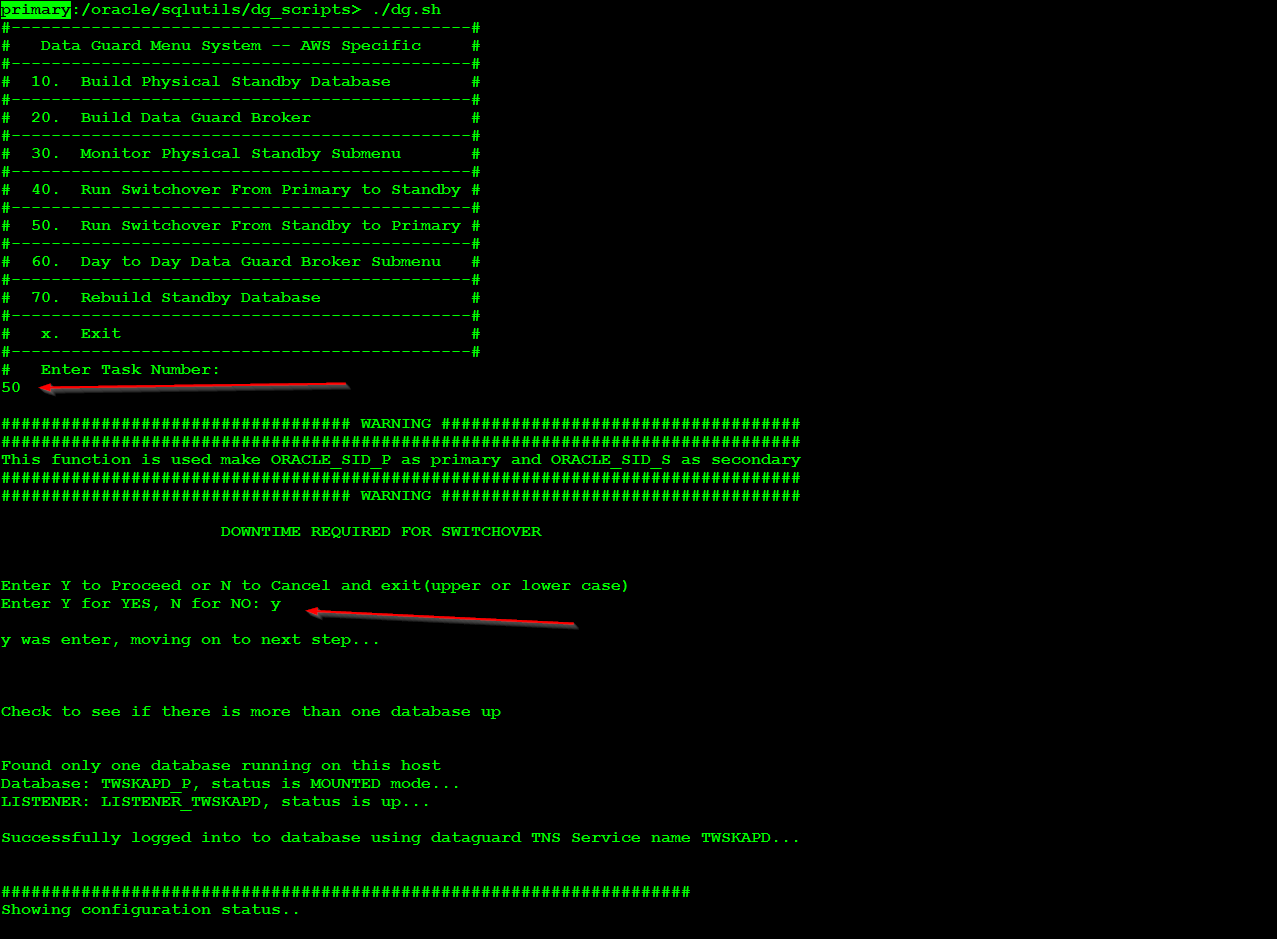


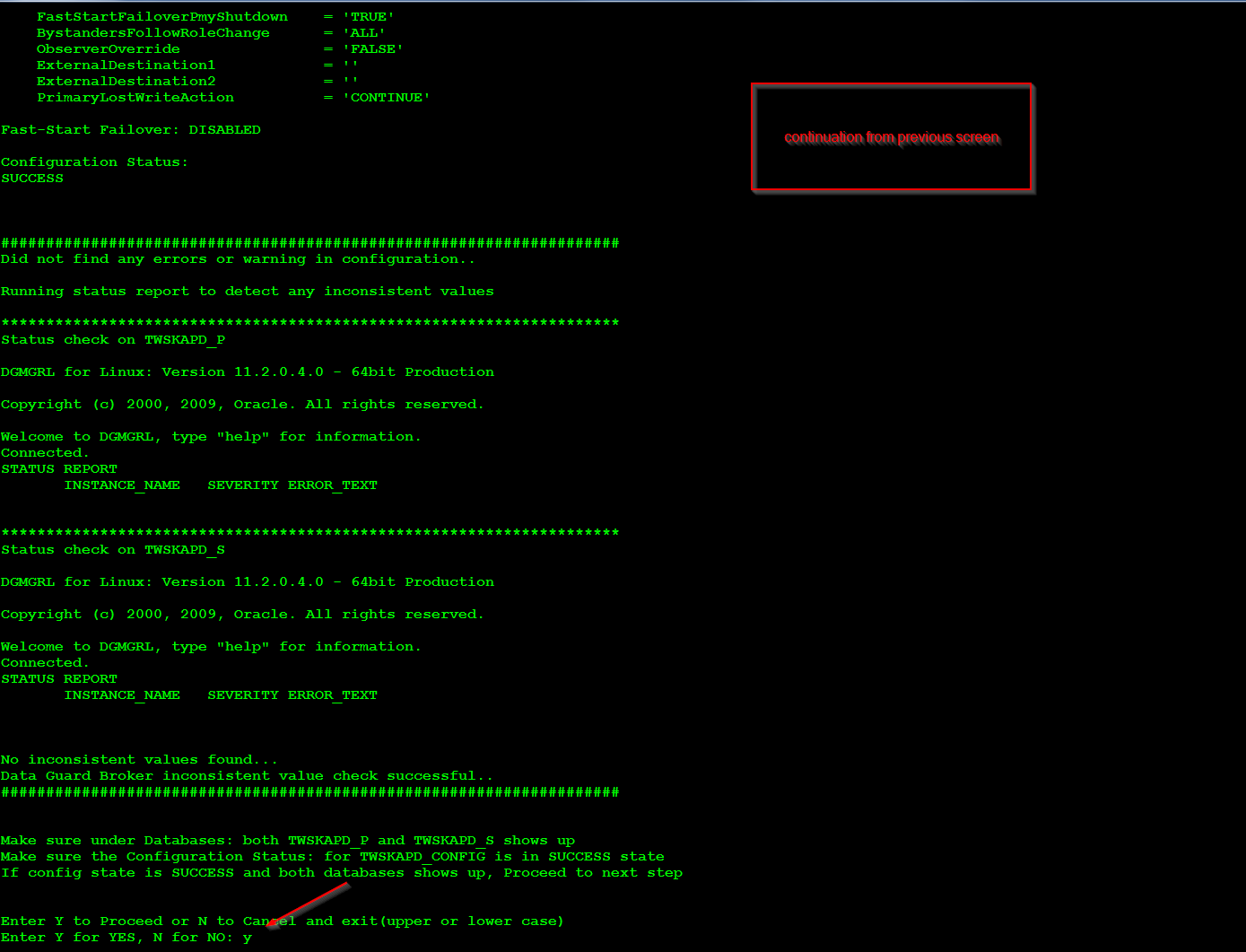
From standby (now the primary)



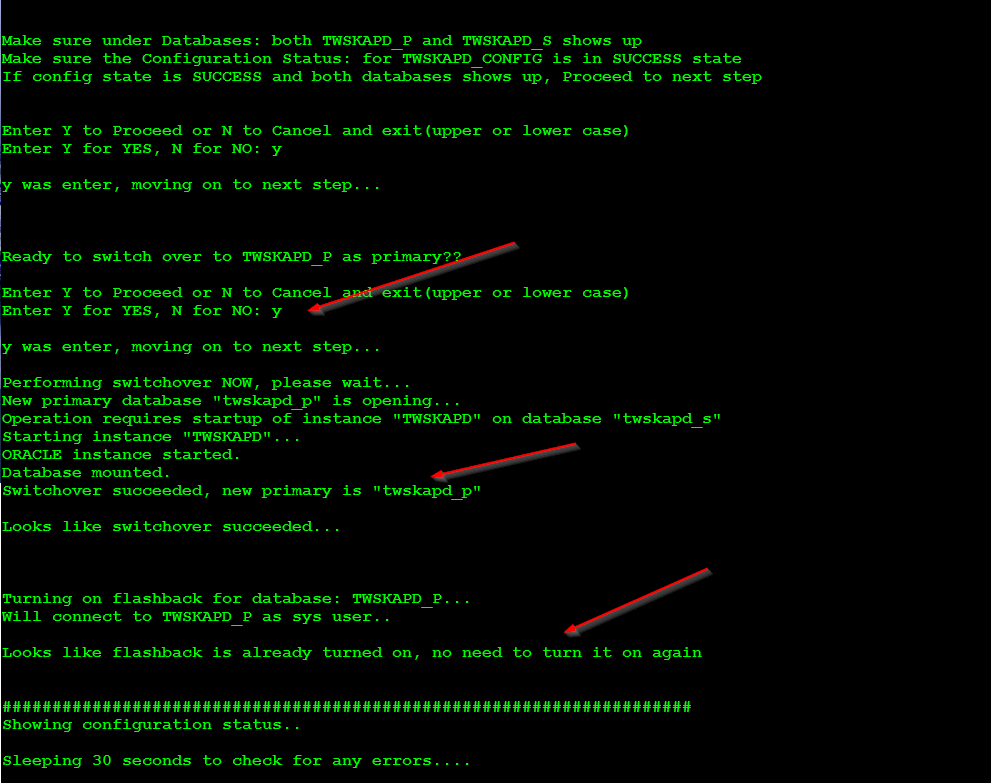
# Performing Switch back from standby to primary

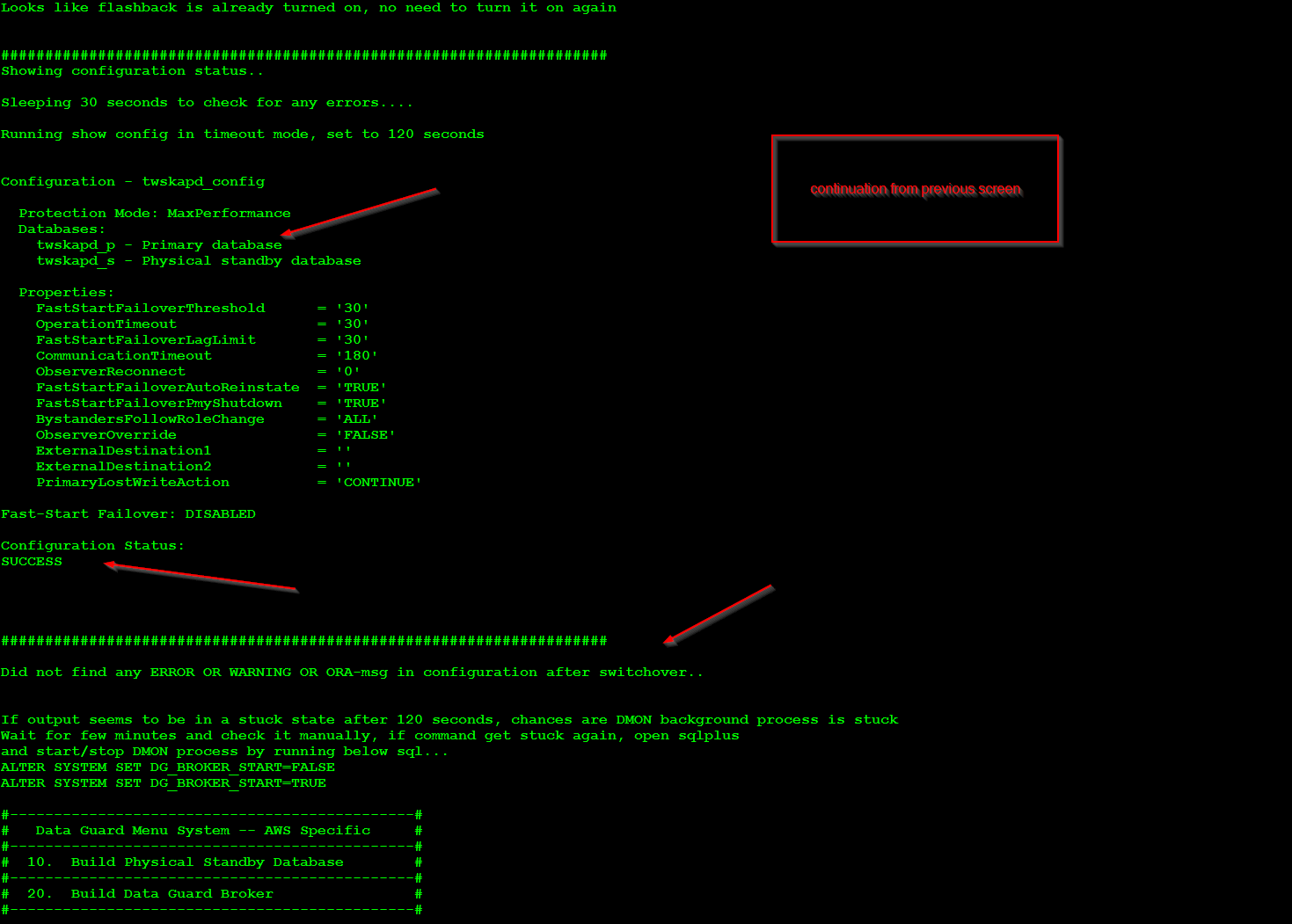
Next we Switch back from standby – twskapd\_s(now the primary) to primary – twskapd\_p (now the standby). Use option#50 for this and enter “y” to proceed





We will ask for confirmation again and it goes on to running the switch over and as you can see it succeeded and now TWSKAPD\_P is back to being primary. And we also check to see if flashback is turned on, if it’s already on we do not turn it on.

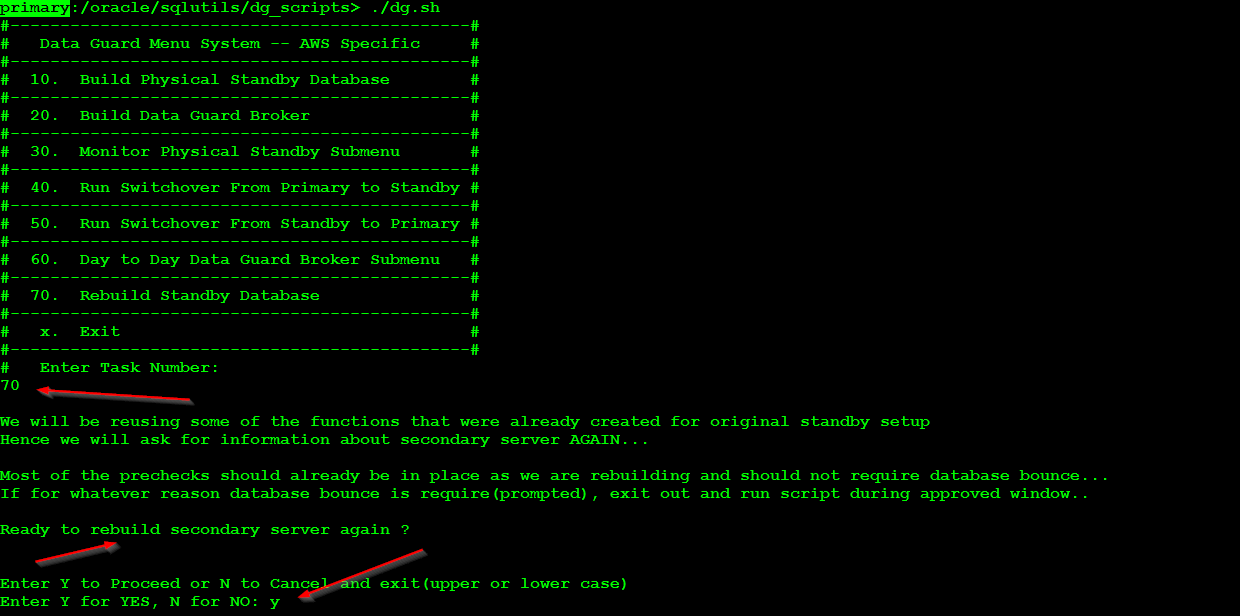




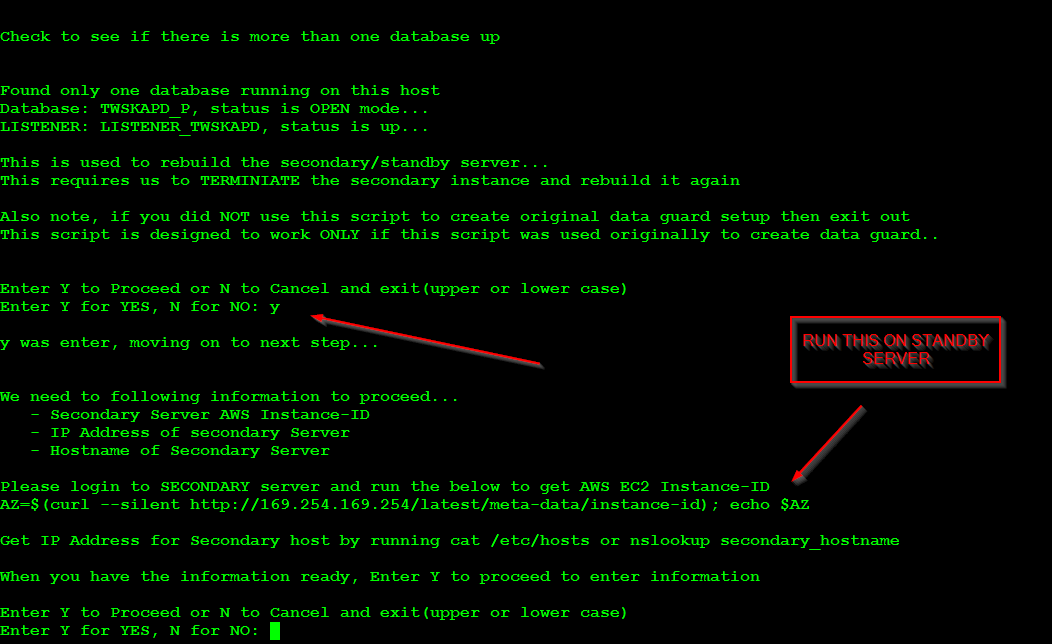
As we can see from above we are back to the original setting of twskapd\_p being primary and twskapd\_s being standby.

# REBUILDING STANDBY DATABASE

For whatever reason if you have to rebuild your standby database, let’s say if the standby redo/archive log is far behind compared to primary, we can use below steps to rebuild the standby. Also note this CAN be done online without any downtime. Use dg.sh script and use option#70 and enter “y” to proceed. **Please note that this option once executed will TERMINATE the secondary instance and create a new one.**

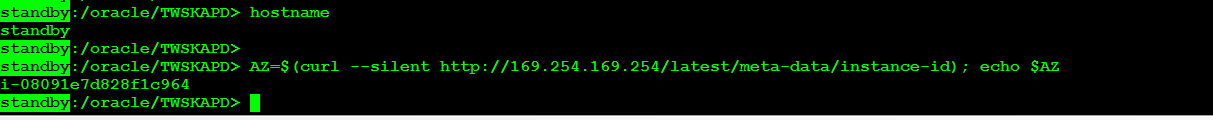


Next enter “Y” to for another confirmation. And as shown below, we need to get couple of things before we can move on to next step. First login to the standby host and run below command to get AWS secondary instance-id, then get the IP address for standby host and the name of the standby host.



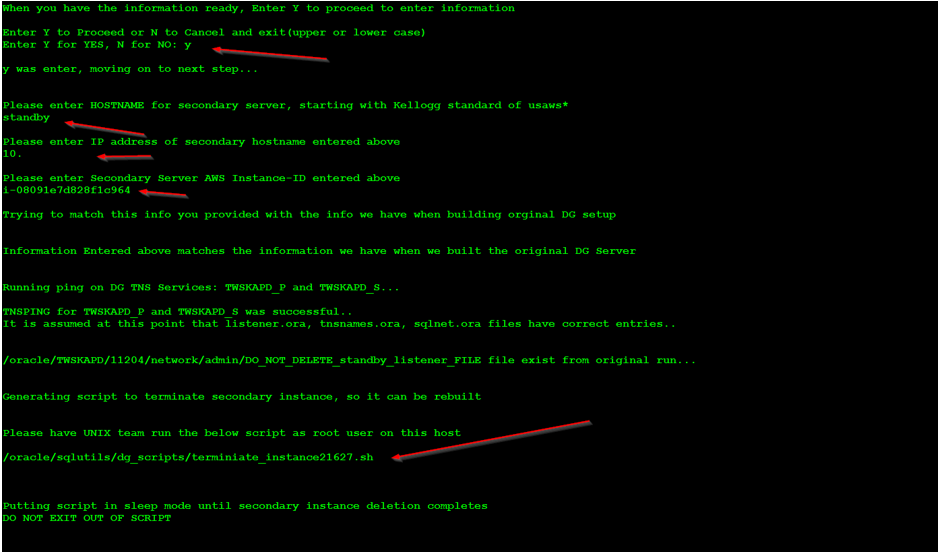
Before entering Y again, go to standby host and run below to get AWS Instance-id

FROM STANDBY SERVER we get below ID, keep this handy

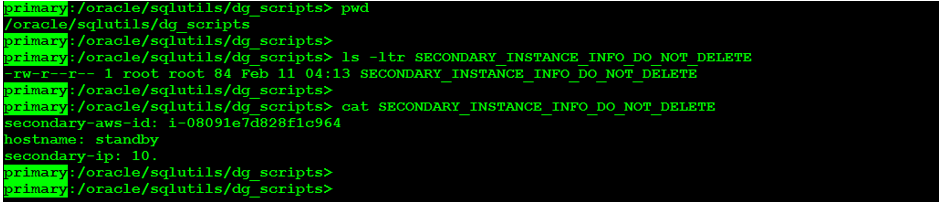


Once we have the information handy, we enter “y” and pass in the secondary host name, as shown below. That is “standby” as host name for secondary host, IP address of 10.0.XX.XXX for standby host, and the “i-08091e7d828flc964” for Instance-ID.

Once the information has been passed we get a script name generated and we need someone with “ROOT” permission to run that script on primary host. This script will TERMINATE the secondary host and the storage attached to it so we can rebuild it again.

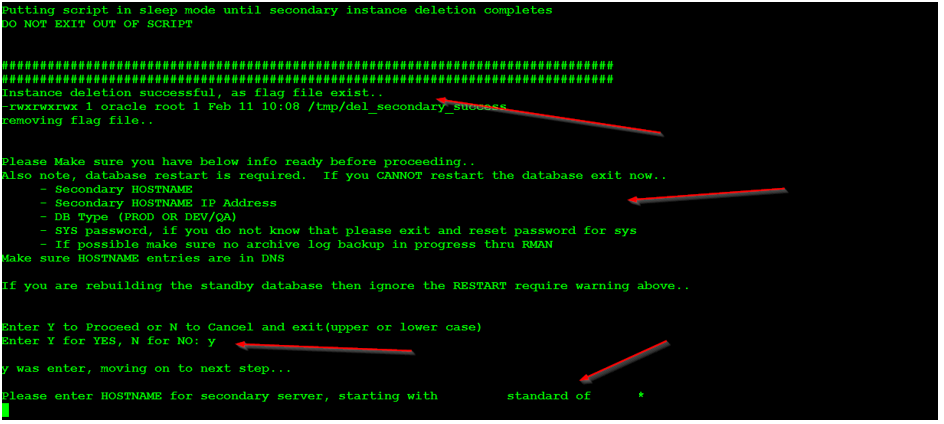


One thing to point out is, we store this information when we build/rebuild the initial standby database and we compare the info passed in above to this file. PLEASE DO NOT DELETE THIS FILE, as you can see it is owned by root.

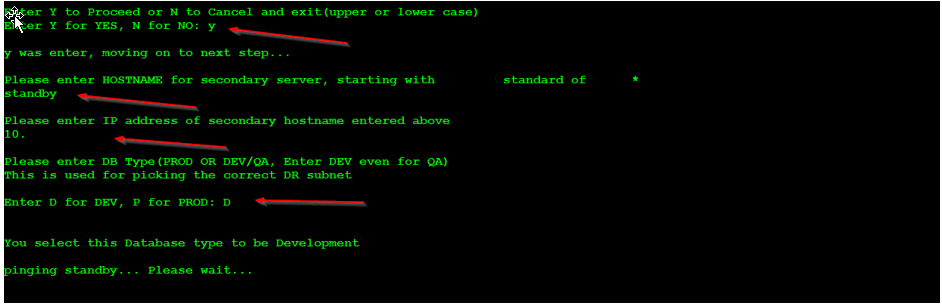


Once the above script finishes we will ask for information again, as the function to create the initial data guard (option #10) is being reused, we ask the information again.

Enter “y” to proceed

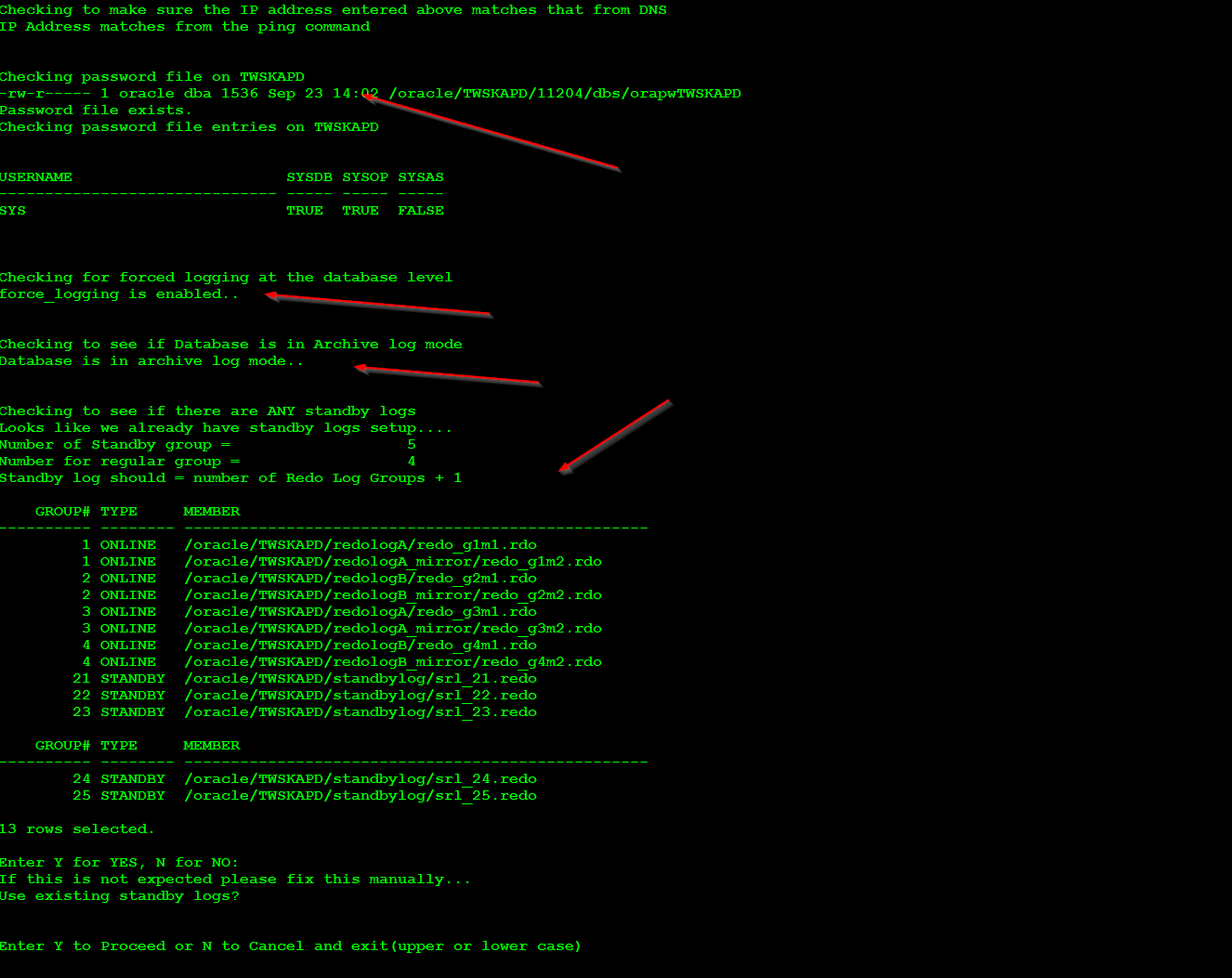


Enter secondary host name (standby in this case), the IP address and if this is Dev or Production.

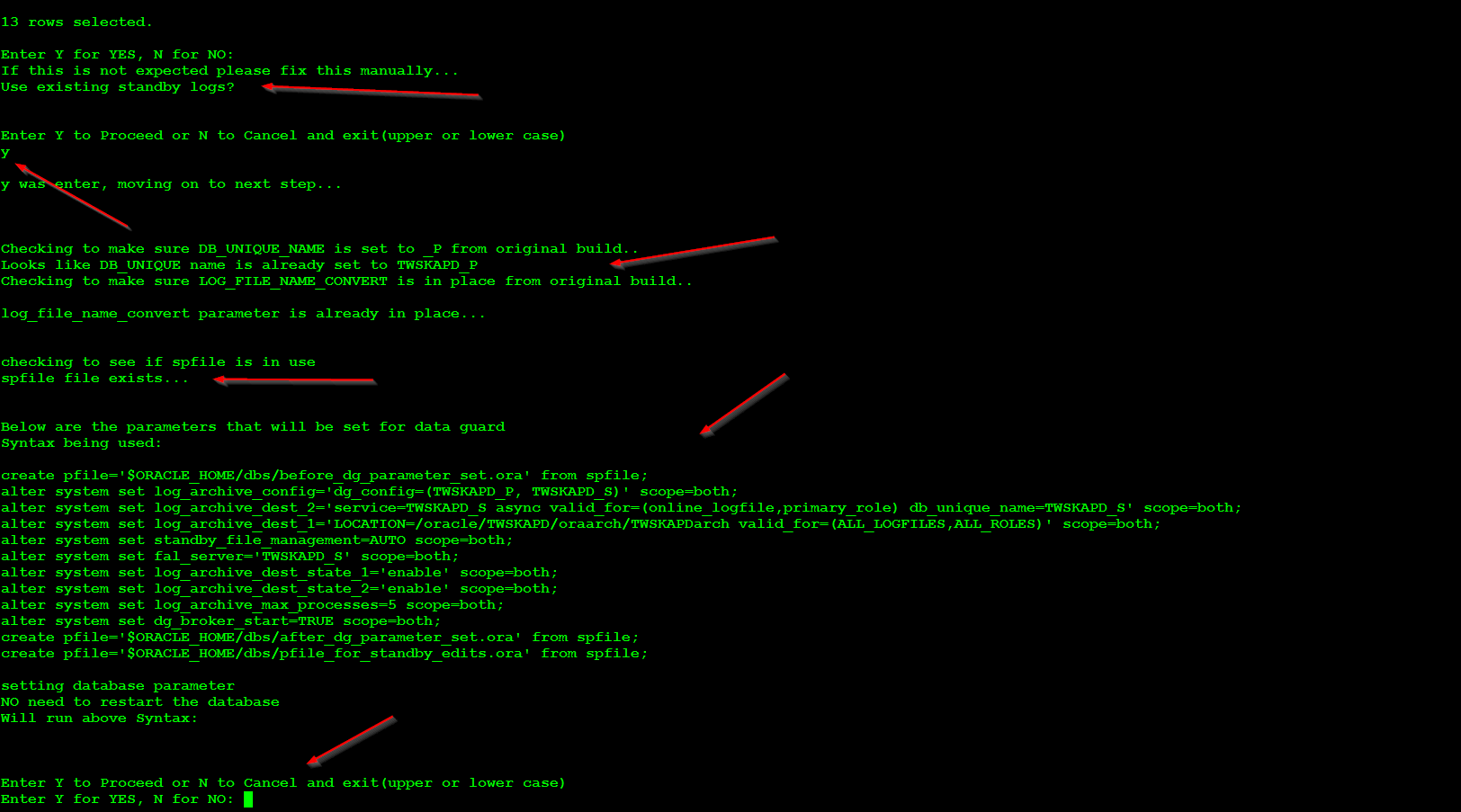


Next we recheck to make sure below are in place and they should be from the initial build, so NO restart is required

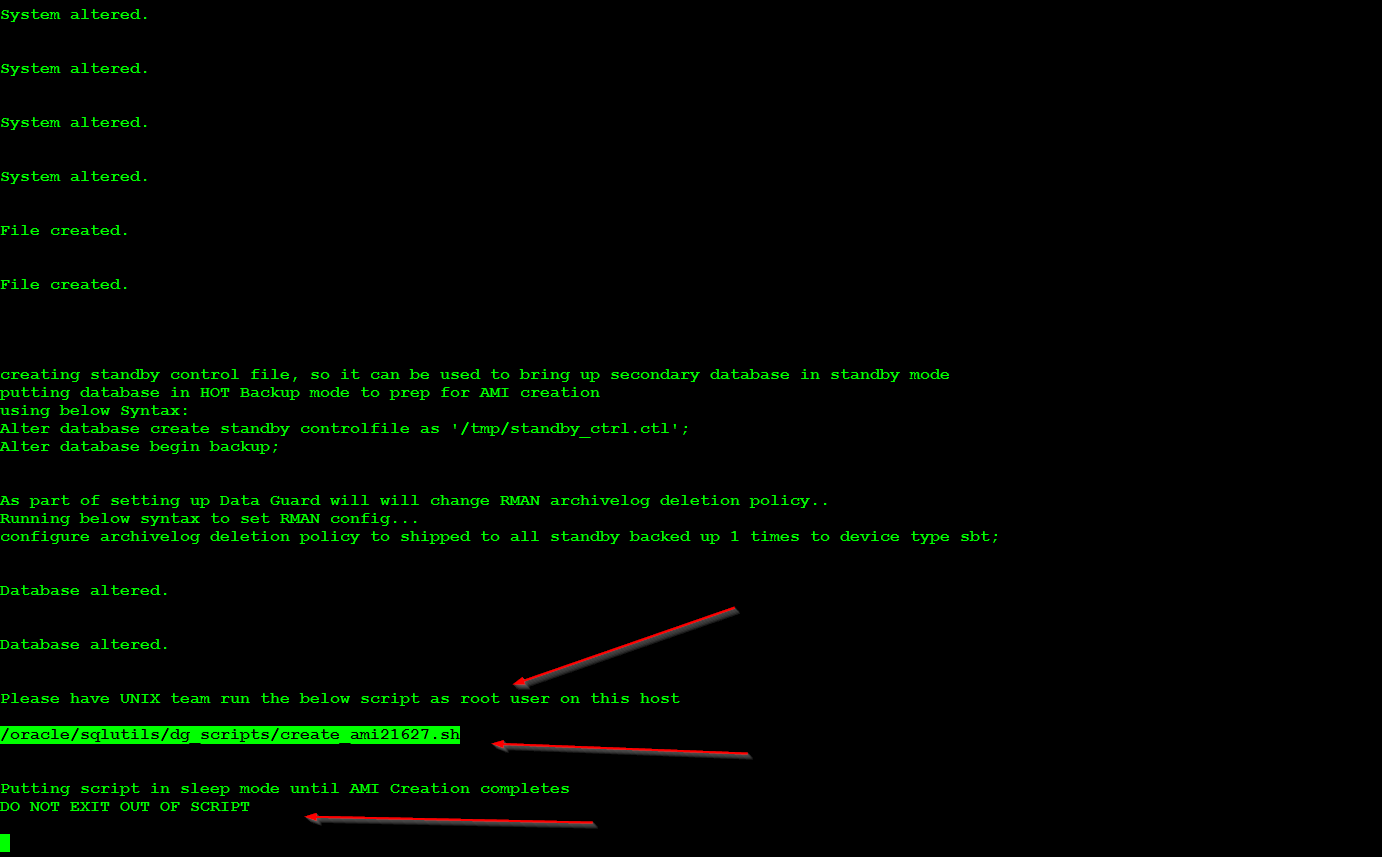
* Password file is in place
* Force logging is enabled
* Archive log is enabled
* Standby logs are in place and if we can reuse it. Enter “y” to use existing standby logs.



Once the standby logs confirmation is given, we check to see if DB\_UNIQUE\_NAME is set to <ORACLE\_SID>\_P AND LOG\_FILE\_NAME\_CONVERT is already in place. This is just a quick way to make sure that we do have those in place (as they are static and need a DB restart and we would want to avoid that).

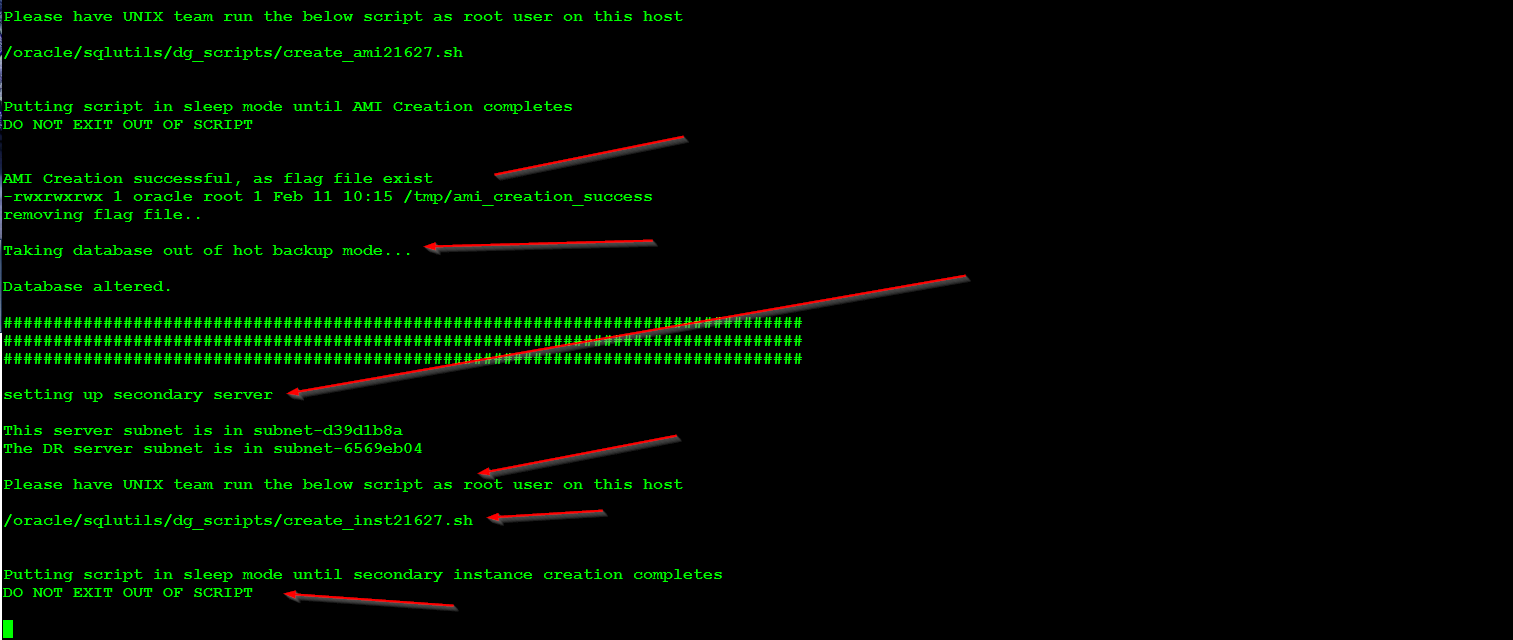


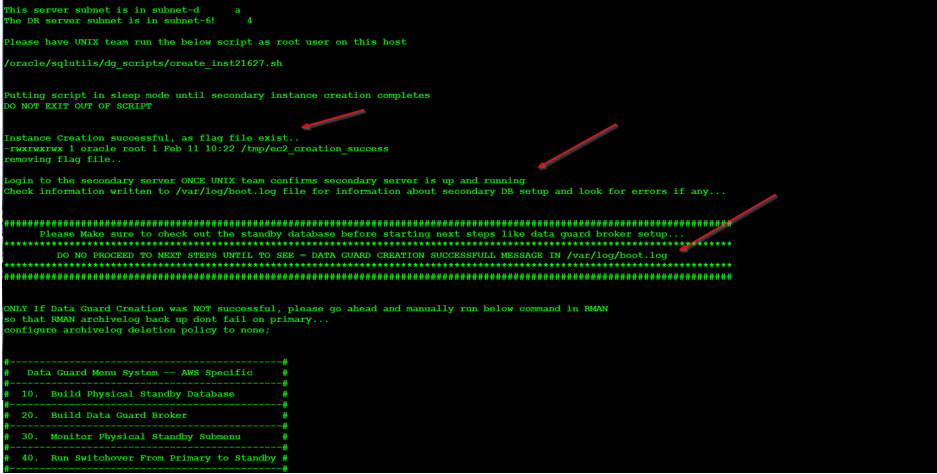
Once we hit Y, all the parameter is set and they are dynamic and no restart is needed, shown below.



Have someone from UNIX team run the above script on this server with ROOT access, and do not exit out of this script. As it will move on when the image creation finishes.

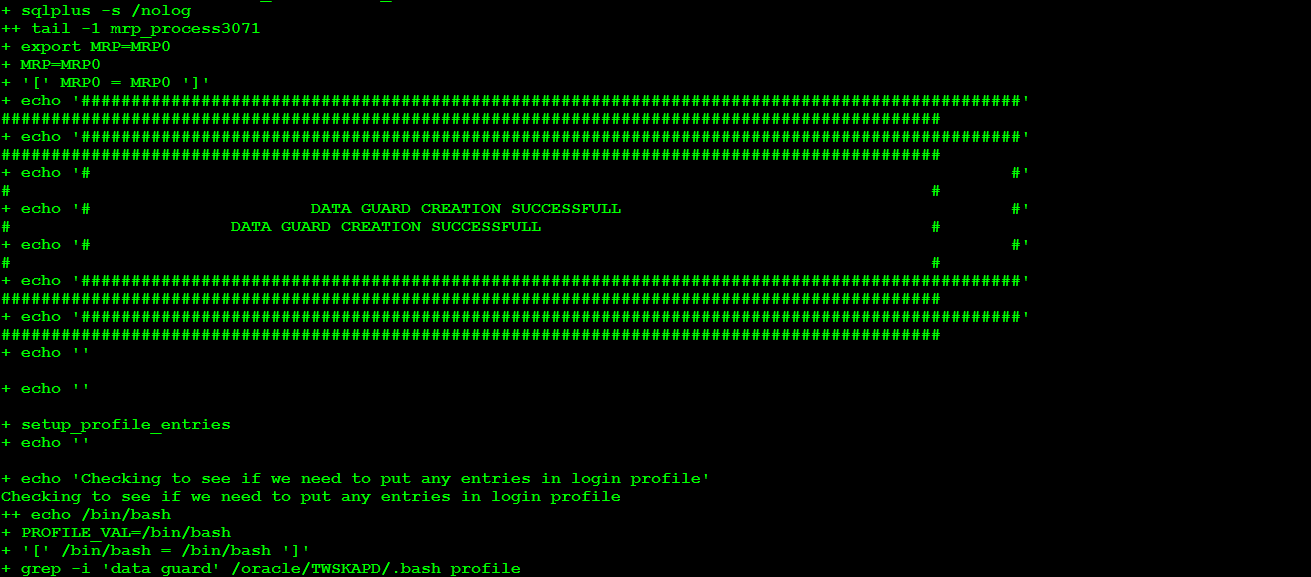
Once the Image creation finished, we move on to bring up the secondary server. Have unix team as root user run the second script on this server so the standby/secondary server can be brought online.





As shown above, once the server comes online (flag file creation from root user), we end display success message and WARN you to look at /var/log/boot.log on secondary server and look for success message there as well.

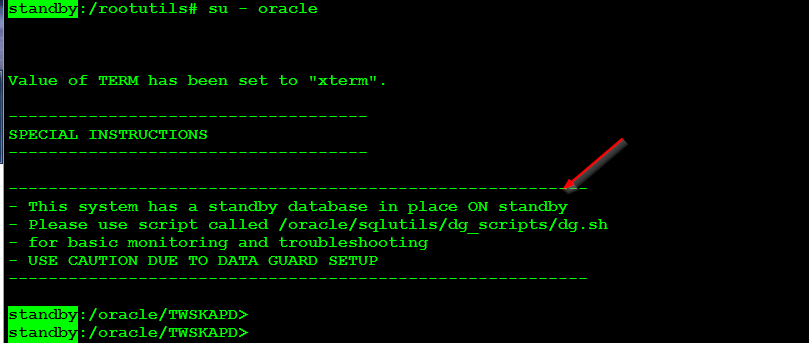
Now from the standby host if we look at /var/log/boot.log we see below entries which means the data guard REBUILD was successful



One thing to MANUALLY do after rebuild finishes is to update the ~/.bash\_profile or ~/.login with proper hostname. We reuse some of the function we created for original standby creation and hence we get this message (shown below) when we login. This should be called PRIMARY.

As shown below this message was copied over from primary (which state it’s on host standby). But in fact we are on the standby we just rebuild and that should say PRIMARY instead of STANDBY.

Update those entries manually and **have ROOT user do the same**.



# Performing Failover

In this scenario we will be doing an actual FAILOVER of production system and same can be followed for other system as well with this setup.

Primary Host is primary(AZ1), DB\_UNIQUE\_NAME is TWSKAPP\_P

Secondary Host is standby(AZ1), DB\_UNIQUE\_NAME is TWSKAPP\_S

**Please note Hostname has been redacted from screenshots below**

Here we have shutdown primary and have NO access to the server. When disaster is declared and we are ready to failover we do the following on the secondary host (primary). Login as the owner of the oracle database, usually oracle or ora<SID> (in SAP env)

1. Set your environment variable if not already set
2. Start data guard broker with dgmgrl
3. Once inside data guard broker interface, connect using “connect sys/PASSWORD;”
4. Make sure the Failover succeeded as show in below step#4
5. Login to sqlplus as sysdba
6. Select the host\_name to make sure you are connecting to proper database
7. Select status from v$ view as show below to make sure it is “OPEN”.
8. Exit

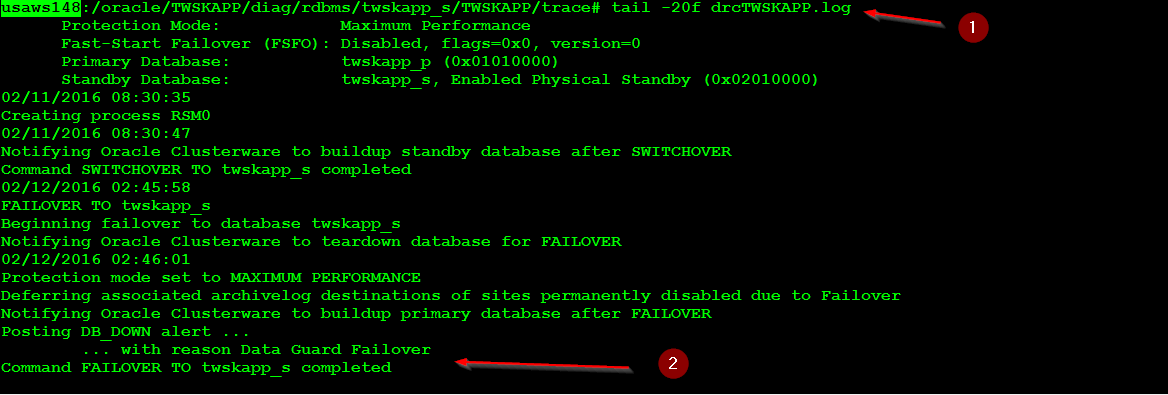


While still on secondary (secondary host), look for entries in data guard broker log file located at

/oracle/TWSKAPP/diag/rdbms/twskapp\_s/TWSKAPP/trace

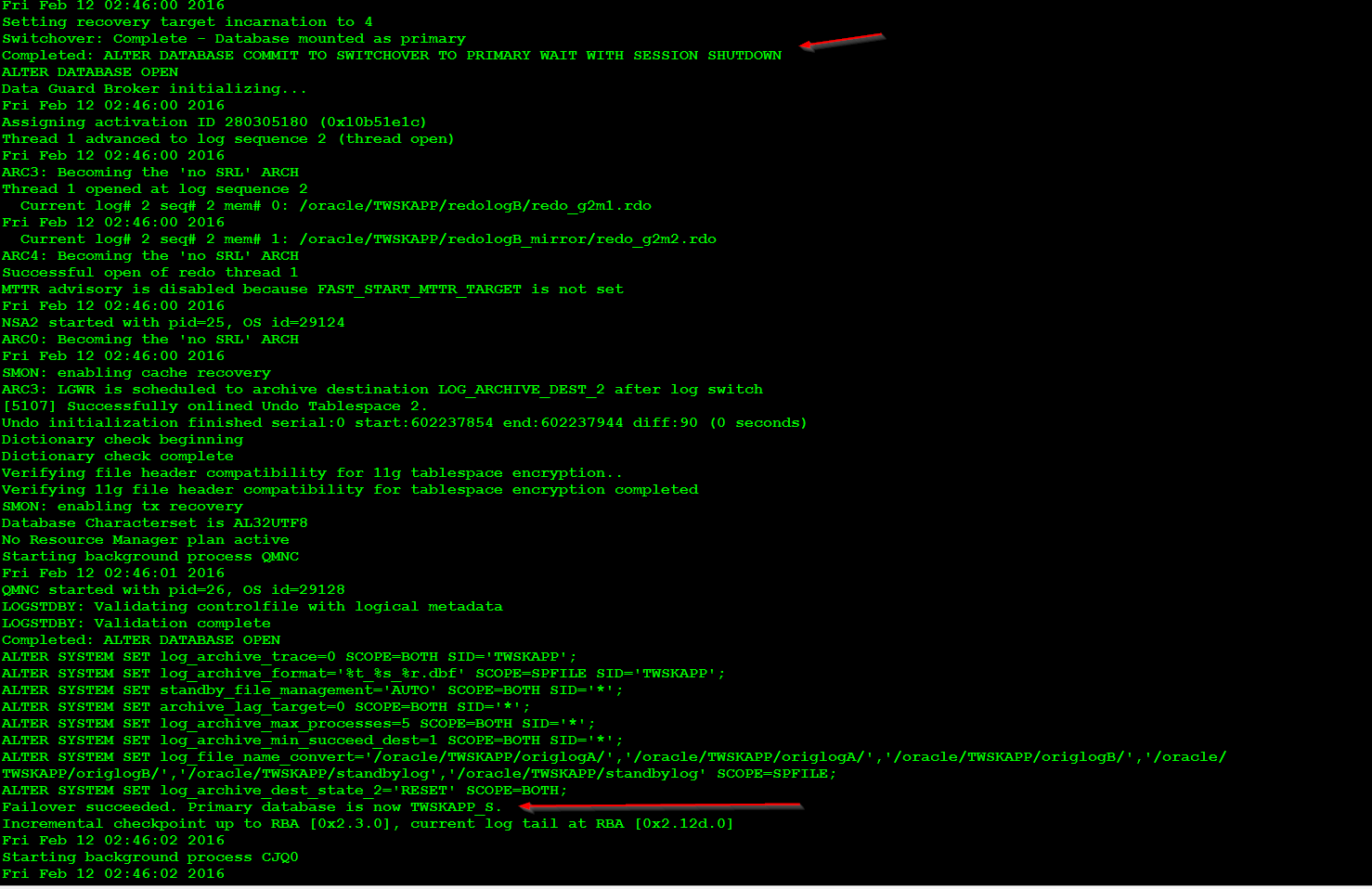
/oracle/<SID>/diag/rdbms/<sid>\_s/<SID>/trace

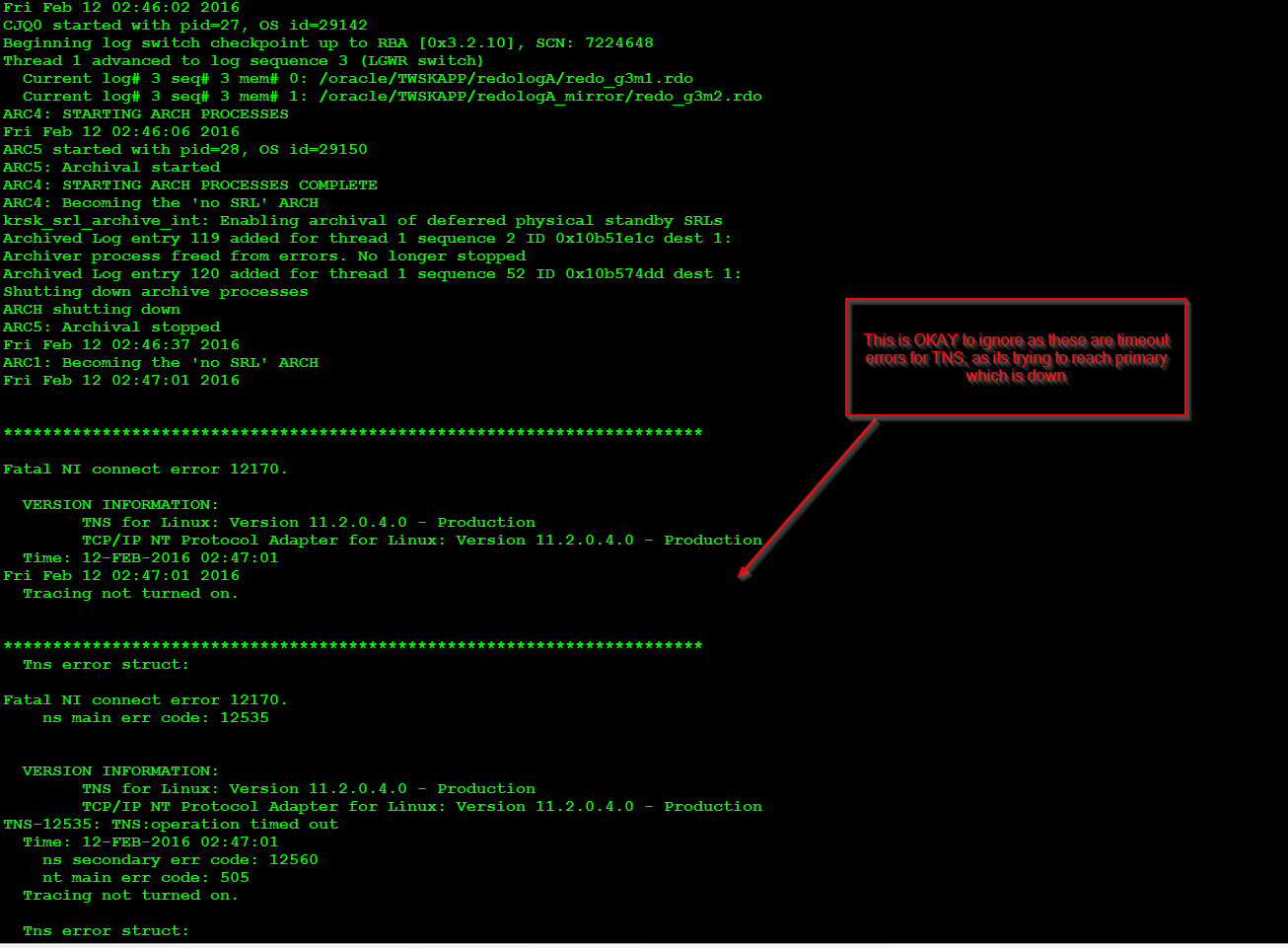
File name is drc<SID>.log as shown below. Make sure there are NO ERRORS reported.



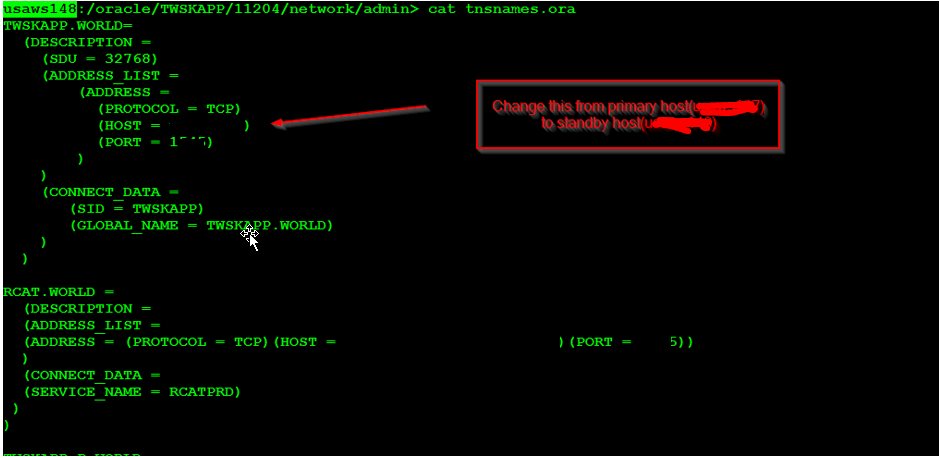
Next make sure alert logs has no errors reported and you see below messages that are pointed for a successful failover at below location

/oracle/<SID>/diag/rdbms/<sid>\_s/<SID>/trace/alert<SID>.log



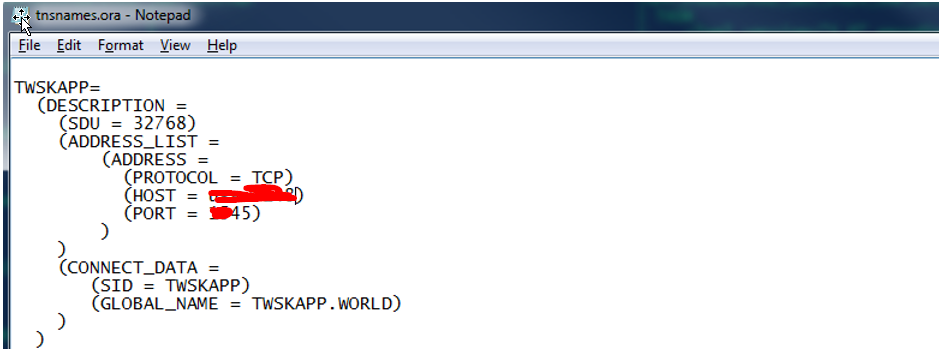


Next thing we have to do is edit the tnsnames.ora file, this can be found at $ORACLE\_HOME/network/admin/tnsnames.ora file. Only change the TNS entries that are used for making connection to database. Change the host entry from primary to secondary.



From your laptop/desktop while connected to Kellogg network make a remote connection using the TNS entries show above. Make sure connection is successful.

In your laptop/desktop make below changes or add new entry as shown below to tnsnames.ora file on your local oracle client.



Open command prompt and do below

1. Tnsping <SID>
2. Make sure connection is status of “OK”
3. Login to sqlplus as system or sys user
4. Run select query shown below



If this is works, please go ahead and release the system to application user.

# Reinstate database after Failover

Here the scenario is

Primary Host is primary(AZ1), DB\_UNIQUE\_NAME is TWSKAPP\_P

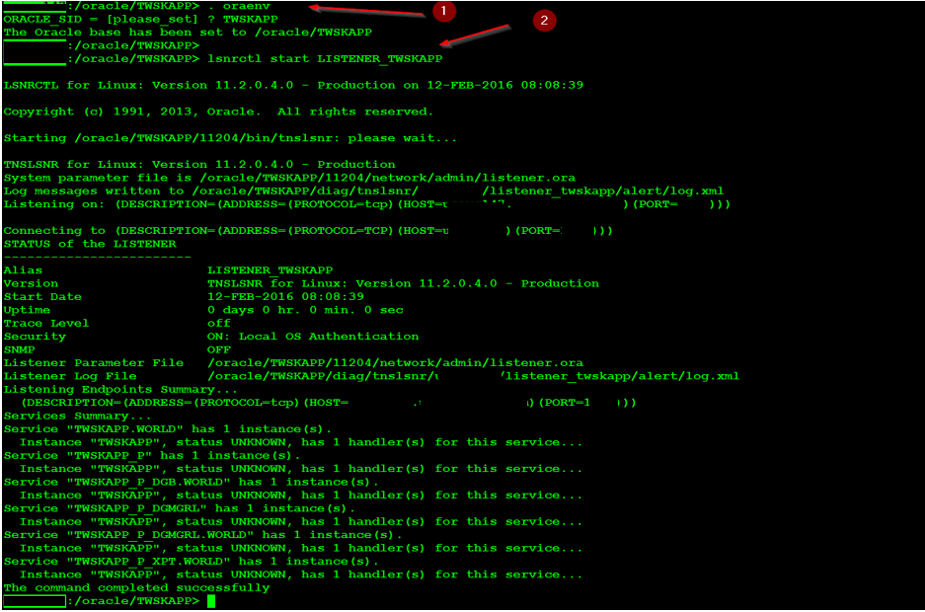
Secondary Host is standby(AZ1), DB\_UNIQUE\_NAME is TWSKAPP\_S

Primary host was down due to failures, and we had promoted the standby host (TWSKAPP\_S) as being primary. Now we want to bring everything back to normal state.

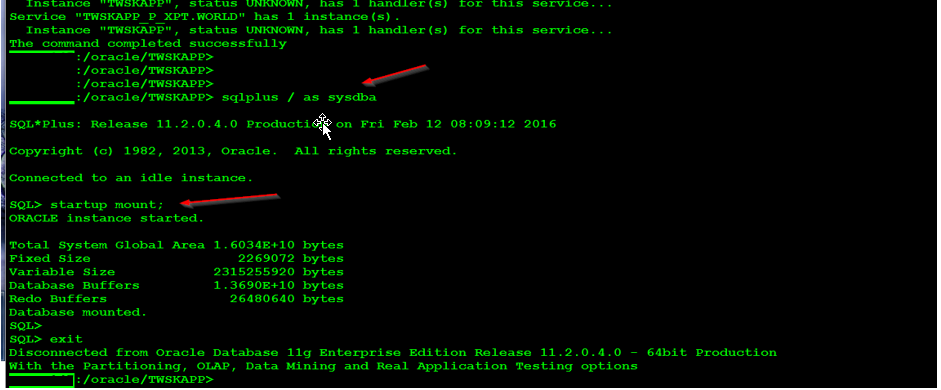
The reason why we reinstate a database is, after the above failover over; when the primary host comes back online it is out of sync with the standby host (now the primary after failover). So to get it in sync we use flashback database. **One important thing to keep in mind is the success of this solely depends on parameter “db\_flashback\_retention\_target” which is set to 12 hours. So you can only do this if you are with the 12 hour window, if you are past the 12 hours then you will have to rebuild data guard.**

Once the primary database which is on primary comes back online, login as oracle or ora<SID> user

1. Set your oracle environment
2. Start the listener

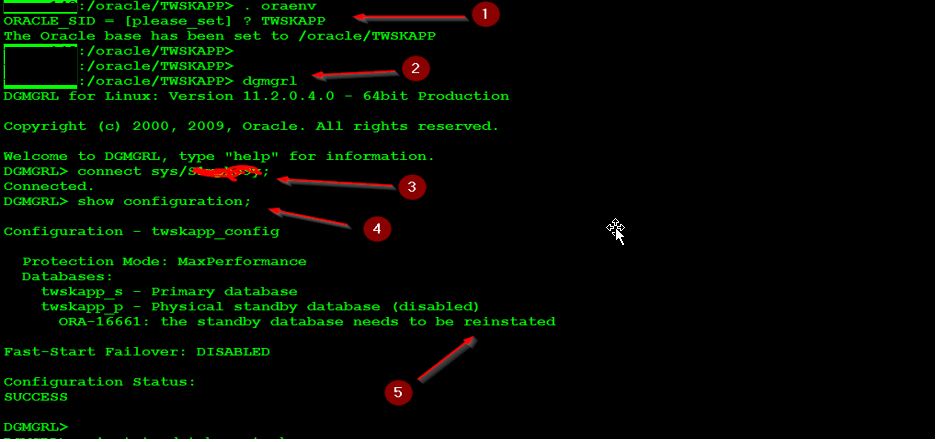


While still on the primary host (primary), login to the database and MOUNT the database, DO NOT OPEN the database for read write.



Next login to the standby host (which is now the primary after failover),

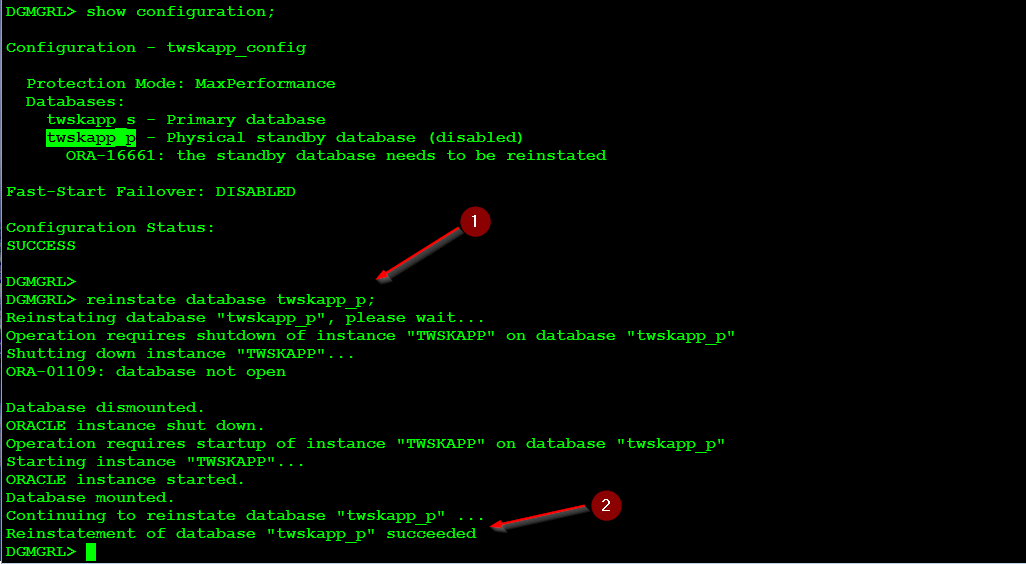
1. Set oracle environment
2. Start data guard broker session
3. Connect using sys password
4. Run “show configuration” command
5. Make a note how twskapp\_p is showing up and standby database that needs to be reinstated.



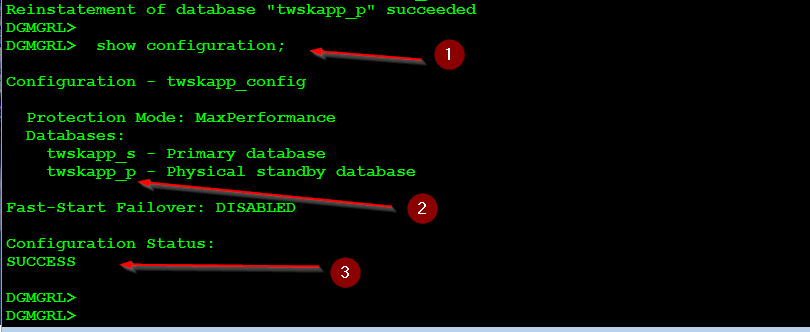
Next we issue below command

1. Reinsate database <SID>\_p;
2. Look for confirmation that reinstatement succeeded.

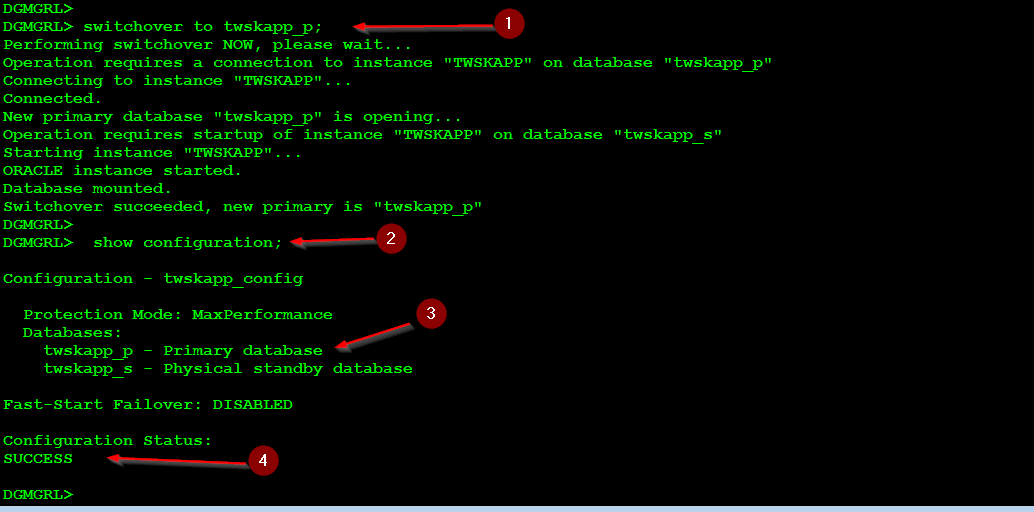
When we issue reinstate database, oracle connects to the primary host, run a flashback database command (right before failover happens) on primary host, then request then the now primary(secondary) to get it in sync with it using redo transport.



Next run show configuration again and we should see a success and <SID>\_p as being the standby database.



Only if we want to bring everything back to normal state, that is primary being primary and secondary being standby, we can issue the below command to switch back over to primary. This does require downtime.



Also looks for any errors in data guard broker trace/alert file located at below location. Note how we have confirmation of reinstate of database and a switchover.

/oracle/<SID>/diag/rdbms/<sid>\_s/<SID>/trace/drc<SID>.log

