Datastar

CASO REAL · PASO A PASO

Migración con Zero Downtime Migration (ZDM): de Exadata On-Premise a OCI

Migración física online (ONLINE_PHYSICAL) con Direct Data Transfer, incluyendo los errores reales que aparecieron en el camino

🎯 Nivel	🕒 Duración del job	🧰 Perfil
Avanzado	~40 minutos (migración)	DBAs Oracle con experiencia en RMAN / Data Guard

Zero Downtime Migration (ZDM) es la herramienta que usa Oracle para mover bases de datos hacia OCI con la mínima interrupción posible, apoyándose en tecnologías que cualquier DBA Oracle ya conoce: RMAN y Data Guard. La promesa es atractiva — migrar "sin downtime" — pero como toda herramienta que orquesta procesos complejos, en la práctica aparecen errores de conectividad, de wallets, de configuración de red que no siempre están documentados con claridad.

Este artículo documenta un caso real de migración de una base Oracle 19c Multitenant, desde una VM en Exadata on-premise hacia un DB System en OCI, usando ZDM con migración física online y transferencia directa de datos (Direct Data Transfer). Además del procedimiento paso a paso, incluimos los tres problemas concretos que surgieron durante la ejecución y cómo se resolvieron — la parte que normalmente no aparece en la documentación oficial.

¿Qué vas a encontrar en este artículo?

- Qué es ZDM y qué significa una migración física "online" con Direct Data Transfer.
- Cómo preparar la base de datos origen y la base de datos destino (placeholder) antes de migrar.

- Cómo dejar la conectividad SSH y de listener (puertos 22 y 1521) lista entre los tres servidores involucrados.
- Cómo ejecutar la evaluación (-eval) y la migración real con zdmcli.
- Tres problemas reales que aparecieron durante el proceso, con su causa raíz y su solución.
- Buenas prácticas para reducir el riesgo de que estos mismos errores te pasen a vos.

> 📌 **NOTA** Por tratarse de un caso real, algunos nombres de host, dominios y contraseñas del documento original fueron reemplazados por valores genéricos de ejemplo antes de publicar este artículo. La estructura de los comandos, parámetros y mensajes de error se mantiene sin cambios.

1. ¿Qué es Zero Downtime Migration (ZDM)?

Zero Downtime Migration es la herramienta de Oracle diseñada específicamente para migrar bases de datos hacia OCI (o entre distintos entornos Oracle) minimizando el tiempo de inactividad. Se ejecuta desde un servidor dedicado (ZDM Service Host) y se controla mediante la utilidad de línea de comandos zdmcli.

ZDM soporta, entre otros, dos grandes enfoques de migración:

- **Migración física (Physical):** usa RMAN y Data Guard para crear una standby en el destino, sincronizarla con el origen y hacer un switchover. Puede ejecutarse en modo ONLINE (con Data Guard real, mínimo downtime) u OFFLINE (backup/restore clásico, con una ventana de downtime mayor).
- **Migración lógica (Logical):** usa Data Pump / GoldenGate para mover los datos a nivel lógico, útil cuando origen y destino no son totalmente compatibles a nivel físico.

En este caso se usó migración física ONLINE, con el método de transferencia Direct Data Transfer: los datafiles se restauran directamente desde el origen hacia el destino a través de la red, sin pasar por un staging intermedio en Object Storage.

> 💡 **TIP PARA DBAs** La migración física ONLINE arma, por detrás, una configuración de Data Guard temporal entre el origen y el destino. Por eso las validaciones y requisitos de Data Guard (mismo character set, mismo edition, wallets correctamente configuradas, SPFILE, etc.) aplican directamente a este tipo de migración con ZDM.

2. Objetivo

El objetivo de este procedimiento es migrar una base de datos Oracle Multitenant 19c hacia OCI, utilizando como destino una base de datos placeholder (misma versión, también multitenant) que será sobrescrita durante el proceso. La migración se realiza con ZDM, método físico online, con Direct Data Transfer.

3. Entornos utilizados

Tres servidores participan del proceso: el servidor origen (on-premise), el servidor destino (OCI) y el servidor de servicio de ZDM, que orquesta la migración.

	Servidor origen (Exadata on-premise)	Servidor destino (OCI)	Servidor ZDM
Hostname	exaadm01vm03.midominio.com.ar	vmtest	exaadm02vm01.midominio.com.ar
Nombre SCAN	scan.midominio.com.ar	vmtest-scan	—
Tipo de servidor	VM en Exadata on-premise	VM — DB System en OCI	VM dedicada para ZDM
Base de datos	Oracle 19c Multitenant	Oracle 19c Multitenant	—
Nombre de la BD	DBTDRS	DBTDR_TGT	—
Storage	ASM	ASM	—
Otros datos	—	IP: 10.0.0.175	Versión: 21c (21.5) · Usuario: ZDMUSER · Herramienta: ZDMCLI

4. Preparación de la base de datos origen

En esta sección se documentan los pasos realizados en la base de datos de origen para dejarla lista para la migración.

4.1 Creación de la base de datos origen

Se creó la base Single Instance llamada DBTDRS, versión 19.0.0.0.0, con una PDB llamada PDBTDR:

```
gdbName=DBTDRS sid=DBTDRS createAsContainerDatabase=true numberOfPDBs=1 \
  pdbName=PDBTDR templateName=General_Purpose.dbc \
  characterSet=AL32UTF8 nationalCharacterSet=AL16UTF16 storageType=ASM \
  datafileDestination=+DATAC3 recoveryAreaDestination=+RECOC3 recoveryAreaSize=5120 \
  diskGroupName=DATAC3 recoveryGroupName=RECOC3 emConfiguration=NONE \
  memoryPercentage=40 automaticMemoryManagement=false \

initParams=undo_tablespace=UNDOTBS1,db_block_size=8192,sga_target=2436M,pga_aggregate_target=1024
M,use_large_pages=true \
```

```
sysPassword=<TU_PASSWORD_SEGURA> systemPassword=<TU_PASSWORD_SEGURA> \
asmsnmpPassword=<TU_PASSWORD_SEGURA> pdbAdminPassword=<TU_PASSWORD_SEGURA>
```

4.2 Configurar la base en modo ARCHIVELOG

La base de datos se configuró en modo ARCHIVELOG para garantizar la consistencia de los datos durante la migración:

```
SQL> shutdown immediate;
Database closed. Database dismounted. ORACLE instance shut down.
SQL> startup mount;
ORACLE instance started.
SQL> alter database archivelog;
Database altered.
SQL> alter database open;
Database altered.
SQL> archive log list;
Database log mode              Archive Mode
Automatic archival             Enabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
```

4.3 Apertura de las PDBs creadas

Se validó que la PDBTDR quedara en estado READ WRITE antes de iniciar el proceso:

```
SQL> show pdbs;
CON_ID CON_NAME  OPEN MODE  RESTRICTED
2      PDB$SEED  READ ONLY  NO
3      PDBTDR    MOUNTED

SQL> ALTER PLUGGABLE DATABASE ALL OPEN;
Pluggable database altered.
SQL> ALTER PLUGGABLE DATABASE ALL SAVE STATE;
Pluggable database altered.

SQL> show pdbs;
CON_ID CON_NAME  OPEN MODE   RESTRICTED
2      PDB$SEED  READ ONLY   NO
3      PDBTDR    READ WRITE  NO
```

> ⚠️ **IMPORTANTE** Este paso es más importante de lo que parece: si después de crear la base hay un reinicio del CDB y las PDBs quedan MOUNTED en lugar de READ WRITE, ZDM puede arrancar la migración igual — y fallar más adelante por falta de master key en la wallet (ver Problema 2 en la sección de Troubleshooting).

4.4 Configuración de la wallet de TDE Keystore

Si la base de datos origen no tiene habilitado TDE (Transparent Data Encryption), es obligatorio configurar la wallet antes de la migración. Consideraciones:

- No es necesario cifrar los datos en la base de datos de origen: los datos se cifran en el destino usando la configuración de la wallet definida en el origen.
- El WALLET_TYPE puede ser AUTOLOGIN (preferido) o PASSWORD.
- El STATUS de la wallet debe estar en OPEN.
- Al ser un entorno multitenant, la wallet debe estar OPEN tanto en el CDB como en todas las PDBs, y la master key debe estar establecida en cada una de ellas.

La configuración utilizada fue la siguiente:

```
SQL> SELECT wrl_parameter, status, wallet_type FROM v$encryption_wallet;
WRL_PARAMETER                                          STATUS          WALLET_TYPE
/u01/app/oracle/product/19.0.0.0/dbhome_1/network/admin/     NOT_AVAILABLE   UNKNOWN

SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE
     '/u01/app/oracle/product/19.0.0.0/dbhome_1/network/admin/'
     identified by <PASSWORD_WALLET>;
keystore altered.

SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN
     IDENTIFIED BY <PASSWORD_WALLET> container = ALL;
keystore altered.

SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY <PASSWORD_WALLET>
     with backup container = ALL;
keystore altered.

SQL> ADMINISTER KEY MANAGEMENT CREATE AUTO_LOGIN KEYSTORE FROM KEYSTORE
     '/u01/app/oracle/product/19.0.0.0/dbhome_1/network/admin/'
     IDENTIFIED BY <PASSWORD_WALLET>;
keystore altered.
```

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY <PASSWORD_WALLET>;
keystore altered.

SQL> shutdown immediate;
SQL> startup;
```

Así quedó configurada correctamente la wallet:

```
SQL> select * from v$encryption_wallet;

WRL_TYPE             WRL_PARAMETER                                                STATUS                         WALLET_TYPE          WALLET_OR KEYSTORE FULLY_BAC
    CON_ID
-------------------- ------------------------------------------------------------ ------------------------------ -------------------- --------- -------- ---------
----------
FILE                 /u01/app/oracle/product/19.0.0.0/dbhome_1/network/admin/     OPEN                           AUTOLOGIN            SINGLE    NONE     NO
         1
FILE                                                                              OPEN                           AUTOLOGIN            SINGLE    UNITED   NO
         2
FILE                                                                              OPEN                           AUTOLOGIN            SINGLE    UNITED   NO
         3
```

5. Preparación de la base de datos destino

En esta sección se documentan los pasos realizados en la base de datos destino, ubicada en el DB System vmtest.

5.1 Creación de la base de datos destino

Se creó la base Oracle 19c denominada DBTDR_TGT. Esta base actúa como placeholder: será sobrescrita durante la migración, pero su configuración general (storage, memoria, character set) queda definida desde su creación.

```
gdbName=DBTDR_TGT sid=DBTDR_TGT createAsContainerDatabase=true numberOfPDBs=1 \
  pdbName=PDBTDR templateName=seed_db.dbc \
  characterSet=AL32UTF8 nationalCharacterSet=AL16UTF16 storageType=ASM \
  datafileDestination=+DATA recoveryAreaDestination=+RECO recoveryAreaSize=5120 \
  diskGroupName=DATA recoveryGroupName=RECO emConfiguration=NONE \
  memoryPercentage=40 automaticMemoryManagement=false \

initParams=undo_tablespace=UNDOTBS1,db_block_size=8192,sga_target=2436M,pga_aggregate_target=1024
M,use_large_pages=true \
  sysPassword=<TU_PASSWORD_SEGURA> systemPassword=<TU_PASSWORD_SEGURA> \
  asmsnmpPassword=<TU_PASSWORD_SEGURA> pdbAdminPassword=<TU_PASSWORD_SEGURA>
```

5.2 Configuración de la wallet TDE Keystore

Al igual que en el origen, se configuró el Keystore de TDE en el destino, con las mismas consideraciones: wallet AUTOLOGIN o PASSWORD según el requerimiento, STATUS en OPEN, y wallet habilitada tanto en el CDB como en las PDBs con la master key establecida en cada una.

5.3 Consideraciones importantes antes de migrar

> ⚠️ **IMPORTANTE** Estos puntos son requisitos duros de ZDM para migraciones físicas ONLINE — si alguno no se cumple, el -eval debería detectarlo, pero conviene revisarlos manualmente antes de lanzar:

- Para migraciones físicas ONLINE (basadas en Data Guard), origen y destino deben tener el mismo sistema operativo y la misma versión de base de datos.
- ZDM no admite migración física entre ediciones distintas (por ejemplo, de Standard Edition a Enterprise Edition).
- El character set de origen y destino debe coincidir.
- Tanto el origen como el destino deben estar configurados con SPFILE (ZDM solo admite PFILE en migraciones OFFLINE; en ONLINE, el origen debe ejecutarse obligatoriamente con SPFILE).
- El archivo sqlnet.ora debe tener la misma configuración de algoritmos de encriptación en origen y destino.
- Se recomienda que la zona horaria (timezone) de origen y destino coincida. Si no se configura de antemano, se puede ajustar después, pero eso genera downtime adicional. ZDM puede automatizar este ajuste como parte del flujo de migración física.

6. Configuración de las conexiones necesarias

Para que la migración se ejecute correctamente, es necesario habilitar la comunicación entre los tres servidores involucrados:

- **Puerto 22:** habilitado para permitir la conexión SSH desde el servidor ZDM hacia los servidores origen y destino.

- **Puerto 1521:** habilitado para permitir la comunicación (listener) entre el servidor origen y el destino, necesaria para la transferencia de datos y la sincronización de Data Guard.

6.1 Comprobación y configuración del puerto 22 (SSH)

Desde el nodo de servicio de ZDM (zdmuser@exaadm02vm01), se comprobó la conectividad SSH hacia origen y destino:

```
[zdmuser@exaadm02vm01 ~]$ telnet exaadm01vm03 22
Trying 172.16.163.44...
Connected to exaadm01vm03.
Escape character is '^]'.
SSH-2.0-OpenSSH_7.4

[zdmuser@exaadm02vm01 ~]$ telnet vmtest 22
Trying 10.0.0.175...
Connected to vmtest.
Escape character is '^]'.
SSH-2.0-OpenSSH_8.0
```

Establecimiento de la conexión SSH sin contraseña

Desde el ZDM_HOST, como usuario oracle, se ejecutó ssh-copy-id para poder conectarse sin que se solicite contraseña:

```
ssh-copy-id -i ~/.ssh/id_rsa.pub oracle@host_origen1
ssh-copy-id -i ~/.ssh/id_rsa.pub oracle@host_destino1
```

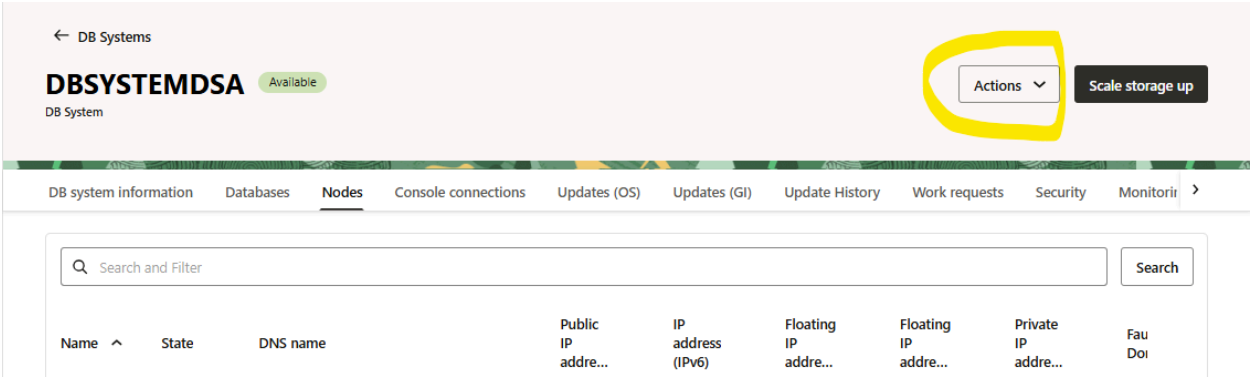
Prueba de conexión SSH desde el servidor ZDM:

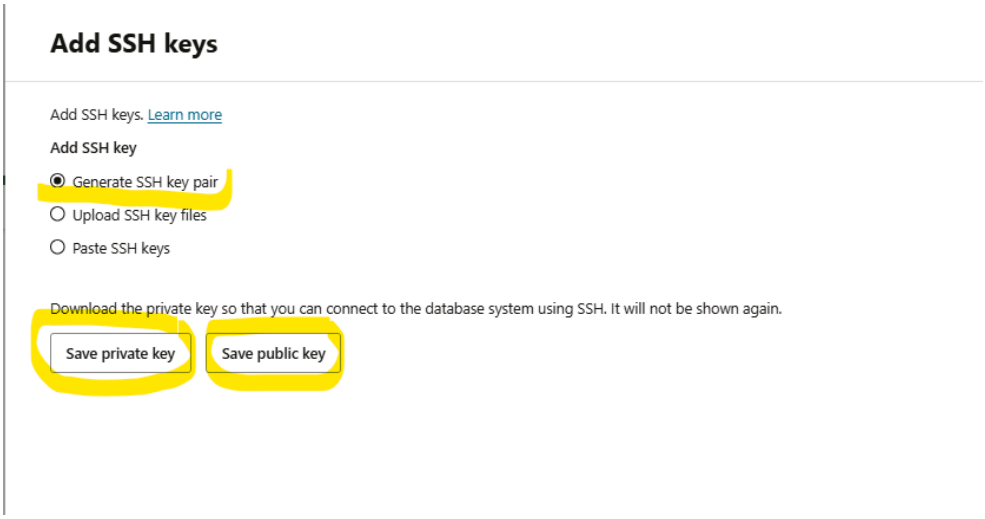
```
[oracle@exaadm02vm01 ~]$ ssh oracle@exaadm01vm03 date
Tue Aug 19 11:27:16 -03 2025

[zdmuser@exaadm02vm01 ~]$ ssh -i ~/.ssh/zdm_service_host.ppk opc@vmtest date
Tue Aug 19 14:29:20 UTC 2025
```

Configuración de la conexión SSH hacia la VM de OCI

1. Generación de claves SSH en OCI. Desde la consola de OCI se generaron las claves SSH para el acceso a la VM de destino; la descarga de la clave se obtiene en formato .key:





2. Conversión de la clave privada. La clave descargada se convirtió de formato .key a .ppk utilizando PuTTYgen — el mismo procedimiento detallado en nuestra guía anterior sobre creación de un OCI Database Service, en la sección "Convertir archivo de clave privada de formato PEM a PPK".

3. Distribución de la clave. La clave privada se envió tanto al servidor ZDM como al servidor de base de datos origen:

```
[zdmuser@exaadm02vm01 ~]$ cd .ssh
[zdmuser@exaadm02vm01 .ssh]$ ls -lrt
total 24
-rw------- 1 zdmuser zdm 1679 Aug 22  2023 id_rsa
-rw-r--r-- 1 zdmuser zdm  422 Aug 22  2023 id_rsa.pub
-rw-r--r-- 1 zdmuser zdm  422 Jul 21 12:14 zdm_service_tool.ppk
-rw-r--r-- 1 zdmuser zdm 5067 Jul 21 12:41 known_hosts
-rw------- 1 zdmuser zdm 1675 Jul 22 17:07 ssh-key-2025-07-21.ppk
```

4. Ajuste de permisos de la clave:

```
chmod 600 ssh-key-2025-07-21.ppk
```

5. Conexión vía SSH al servidor destino, usando el usuario estándar opc:

```
[zdmuser@exaadm02vm01 ~]$ ssh -i ~/.ssh/zdm_service_host.ppk opc@vmtest date
Tue Aug 19 14:29:20 UTC 2025
```

Desde el servidor origen hacia el servidor destino:

```
[oracle@exaadm01vm03 .ssh]$ ssh -i ssh-key-2025-07-21.ppk opc@vmtest date
Tue Aug 19 17:20:40 UTC 2025
```

6.2 Comprobación y configuración del puerto 1521

El puerto 1521 debe estar abierto y no bloqueado por firewall: se usa para la conectividad de Oracle Net Services entre origen y destino, lo que permite la sincronización de Data Guard.

Edición del archivo /etc/hosts

Se agregaron las entradas correspondientes a la base origen (DBTDRS) y a la base destino (DBTDR_TGT). Formato de entrada: <IP_host> <nombreHost> <aliasHost>.

Archivo /etc/hosts en el servidor origen (correspondiente a la base DBTDRS):

```
### OCI VM
10.0.0.175  sub10311921030.jsvcn.oraclevcn.com  vmtest
10.0.0.175  vmtest-scan
```

Archivo /etc/hosts en el servidor destino (correspondiente a la base DBTDR_TGT):

```
#IP EXADATA vm03
172.16.163.44  exaadm01vm03.midominio.com.ar  exaadm01vm03
172.16.163.44  exa-scan3.midominio.com.ar  exa-scan3
```

> 💡 **TIP PARA DBAs** ZDM no siempre resuelve nombres usando los hostnames "de conveniencia" que agregues al /etc/hosts: para el chequeo de conectividad de listener, usa puntualmente los nombres SCAN de origen y destino. Conviene agregar también esos alias SCAN al /etc/hosts desde el principio, para evitar el problema descrito en la sección de Troubleshooting.

Capturas de las conexiones establecidas por el puerto 1521, validadas en ambos sentidos:

```
[oracle@vmtest ~]$ tnsping exaadm01vm03:1521/DBTDRS

TNS Ping Utility for Linux: Version 19.0.0.0.0 - Production on 30-JUL-2025 17:41:15

Copyright (c) 1997, 2025, Oracle.  All rights reserved.

Used parameter files:
/u01/app/oracle/product/19.0.0.0/dbhome_1/network/admin/sqlnet.ora

Used HOSTNAME adapter to resolve the alias
Attempting to contact (DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=DBTDRS))(ADDRESS=(PROTOCOL=tcp)(HOST=172.16.163.44)(PORT=1521)))
OK (90 msec)
[oracle@vmtest ~]$
```

```
[oracle@vmtest ~]$ sqlplus sys/WelcomE-01#@172.16.163.44:1521/DBTDRS AS SYSDBA

SQL*Plus: Release 19.0.0.0.0 - Production on Wed Jul 30 17:44:51 2025
Version 19.27.0.0.0

Copyright (c) 1982, 2024, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.25.0.0.0

SQL>
```

```
SQL> select instance_name, host_name from v$instance;

INSTANCE_NAME
----------------
HOST_NAME
----------------------------------------------------------------
DBTDRS
exaadm01vm03.sop.datastar.com.ar


SQL>
```

```
[oracle@exaadm01vm03 ~]$ tnsping vmtest-scan

TNS Ping Utility for Linux: Version 19.0.0.0.0 - Production on 06-AUG-2025 12:23:52

Copyright (c) 1997, 2024, Oracle.  All rights reserved.

Used parameter files:
/u01/app/oracle/product/19.0.0.0/dbhome_1/network/admin/sqlnet.ora

Used EZCONNECT adapter to resolve the alias
Attempting to contact (DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=))(ADDRESS=(PROTOCOL=tcp)(HOST=10.0.0.175)(PORT=1521)))
OK (110 msec)
[oracle@exaadm01vm03 ~]$ exit
```

```
[oracle@vmtest ~]$ /u01/app/oracle/product/19.0.0.0/dbhome_1/bin/tnsping exa-scan3:1521

TNS Ping Utility for Linux: Version 19.0.0.0.0 - Production on 06-AUG-2025 16:02:47

Copyright (c) 1997, 2025, Oracle.  All rights reserved.

Used parameter files:
/u01/app/oracle/product/19.0.0.0/dbhome_1/network/admin/sqlnet.ora

Used HOSTNAME adapter to resolve the alias
Attempting to contact (DESCRIPTION=(CONNECT_DATA=(SERVICE_NAME=))(ADDRESS=(PROTOCOL=tcp)(HOST=172.16.163.44)(PORT=1521)))
OK (70 msec)
```

Configuración de sqlnet.ora

Se configuró el archivo sqlnet.ora tanto en el servidor de la base de datos destino (DBTDR_TGT) como en el de origen (DBTDRS).

sqlnet.ora de DBTDRS (origen):

```
[oracle@exaadm01vm03 .ssh]$ cd $ORACLE_HOME/network/admin
[oracle@exaadm01vm03 admin]$ less sqlnet.ora

ENCRYPTION_WALLET_LOCATION=(SOURCE=(METHOD=FILE)
  (METHOD_DATA=(DIRECTORY=/u01/app/oracle/product/19.0.0.0/dbhome_1/network/admin/)))
```

sqlnet.ora de DBTDR_TGT (destino):

```
[opc@vmtest ~]$ cd /u01/app/oracle/product/19.0.0.0/dbhome_1/network/admin
[opc@vmtest admin]$ less sqlnet.ora

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

7. Proceso de migración con ZDM

ZDM utiliza un response file para definir todos los parámetros necesarios de la migración: rutas de los ORACLE_HOME, datos de conexión de origen y destino, y el tipo de migración a ejecutar. El archivo base se encuentra en $ZDM_HOME/rhp/zdm/template.

7.1 Parámetros modificados del response file

```
TGT_DB_UNIQUE_NAME=DBTDR_TGT
```

```
MIGRATION_METHOD=ONLINE_PHYSICAL
DATA_TRANSFER_MEDIUM=DIRECT
NONCDBTOPDB_SWITCHOVER=FALSE
ZDM_USE_DG_BROKER=TRUE
ZDM_RMAN_ENCRYPT_BACKUP=FALSE
ZDM_TGT_STBY_CASCADING=FALSE
```

7.2 Ejecución de la evaluación (EVAL)

Antes de ejecutar la migración real, se corrió una evaluación (-eval) con el mismo response file. Este paso valida conectividad, parámetros y prerrequisitos sin tocar la base de datos destino:

```
$ZDM_HOME/bin/zdmcli migrate database -sourcedb DBTDRS \
  -sourcenode exaadm01vm03.midominio.com.ar -srcauth zdmauth \
  -srcarg1 user:oracle -srcarg2 identity_file:/home/zdmuser/.ssh/zdm_service_host.ppk \
  -srcarg3 sudo_location:/usr/bin/sudo -targetnode vmtest -tgtauth zdmauth \
  -tgtarg1 user:opc -tgtarg2 identity_file:/home/zdmuser/.ssh/zdm_service_host.ppk \
  -tgtarg3 sudo_location:/usr/bin/sudo \
  -targethome /u01/app/oracle/product/19.0.0.0/dbhome_1 \
  -rsp /u01/app/zdm/zdmhome/rhp/zdm/template/zdm_physical_DBTDRS_oci.rsp -eval
```

Resultado del eval:

```
exaadm02vm01: 2025-08-06T16:14:11.507Z : Oracle ZDM ONLINE PHYSICAL evaluation job completed
exaadm02vm01: 2025-08-06T16:14:11.508Z : Job duration: 6 minutes and 23 seconds
```

> 💡 **TIP PARA DBAs** Correr el -eval no es opcional: es la forma más barata de detectar problemas de conectividad, wallets o configuración antes de tocar la base destino. Todos los problemas descritos en la sección de Troubleshooting se detectaron justamente en sucesivas corridas de eval y de la migración real.

7.3 Ejecución de la migración

Una vez validado el entorno, se ejecutó la migración (mismo comando, sin el flag -eval):

```
$ZDM_HOME/bin/zdmcli migrate database -sourcedb DBTDRS \
```

```
 -sourcenode exaadm01vm03.midominio.com.ar -srcauth zdmauth \
 -srcarg1 user:oracle -srcarg2 identity_file:/home/zdmuser/.ssh/zdm_service_host.ppk \
 -srcarg3 sudo_location:/usr/bin/sudo -targetnode vmtest -tgtauth zdmauth \
 -tgtarg1 user:opc -tgtarg2 identity_file:/home/zdmuser/.ssh/zdm_service_host.ppk \
 -tgtarg3 sudo_location:/usr/bin/sudo \
 -targethome /u01/app/oracle/product/19.0.0.0/dbhome_1 \
 -rsp /u01/app/zdm/zdmhome/rhp/zdm/template/zdm_physical_DBTDRS_oci.rsp
```

El comando genera un job id, que permite hacer seguimiento del avance de la migración y consultar su log:

```
$ZDM_HOME/bin/zdmcli query job -jobid 42
```

7.4 Monitoreo del job

Siguiendo el log del job (tail -f) se puede ver el avance fase por fase — ZDM_SETUP_SRC, ZDM_SETUP_TGT, ZDM_VALIDATE_SRC/TGT, ZDM_DISCOVER_SRC, ZDM_COPYFILES, ZDM_RESTORE_TGT, ZDM_CONFIGURE_DG_SRC, ZDM_SWITCHOVER_SRC/TGT, hasta ZDM_CLEANUP_SRC/TGT:

```
[zdmuser@exaadm02vm01 template]$ tail -f /u01/app/zdm/zdmbase/chkbase/scheduled/job-42-....log
exaadm02vm01: ... Executing phase ZDM_SETUP_SRC
exaadm01vm03: ... TNS aliases successfully setup on the source node exaadm01vm03...
exaadm02vm01: ... Execution of phase ZDM_SETUP_SRC completed
exaadm02vm01: ... Executing phase ZDM_SETUP_TGT
vmtest: ... TNS aliases successfully setup on the target node vmtest...
exaadm02vm01: ... Execution of phase ZDM_SETUP_TGT completed
exaadm02vm01: ... Executing phase ZDM_DISCOVER_SRC
exaadm01vm03: ... Enabling force logging on database DBTDRS...
exaadm01vm03: ... Creating standby logs on database DBTDRS...
exaadm01vm03: ... Source environment set up successfully
  [ ... fases intermedias omitidas ... ]
exaadm02vm01: ... Executing phase ZDM_CLEANUP_TGT
vmtest: ... Unregistering Data Guard configuration for database DBTDRS from database DBTDR_TGT...
vmtest: ... Removed the Oracle Data Guard configuration successfully
vmtest: ... Dropping standby log file groups from database DBTDR_TGT...
exaadm02vm01: ... Execution of phase ZDM_CLEANUP_TGT completed
```

7.5 Resultado final

```
exaadm02vm01: 2025-08-12T19:59:50.280Z : Oracle ZDM ONLINE PHYSICAL migration completed
exaadm02vm01: 2025-08-12T19:59:50.281Z : Job duration: 40 minutes and 31 seconds
```

Con esto, la base de datos DBTDR_TGT en OCI queda como el nuevo primary, con los datos migrados desde DBTDRS y la configuración temporal de Data Guard ya desmontada por ZDM al finalizar.

8. Troubleshooting: problemas reales y cómo se resolvieron

Durante la preparación y la ejecución de la migración aparecieron tres problemas concretos. Se documentan en el orden en que se presentaron, porque cada uno llevó al siguiente.

🔴 PROBLEMA — Error de conexión / resolución de nombres (TNS-03505)

Al ejecutar el -eval, el pre-chequeo de conectividad falló:

```
PRGZ-1132 : -eval failed for the phase ZDM_PRECHECKS_TGT with exception
PRGZ-3130 : failed to establish connection to target listener from nodes [exaadm01vm03]
PRCZ-4001 : failed to execute command ".../bin/tnsping" using the privileged
            execution plugin "zdmauth" on nodes "exaadm01vm03" within 15 seconds
PRCZ-2103 : Failed to execute command ".../bin/tnsping" on node "exaadm01vm03"
            as user "root". Detailed error:
TNS Ping Utility for Linux: Version 19.0.0.0.0
TNS-03505: Failed to resolve name.
```

No había una respuesta concreta en el log generado por ZDM al ejecutar el comando, así que se consultó la sección de Troubleshooting de la guía oficial de ZDM:

Zero Downtime Migration Service Host Log

Determine which operational phase the migration job was in at the time of failure, and whether the phase belongs to the source (phase name contains SRC) or target (phase name contains TGT) . Check the Zero Downtime Migration service host log at `$ZDM_BASE/crsdata/zdm_service_host/rhp/zdmserver.log.0`, and access the respective source or target server to check the log associated with the operational phase in `$ORACLE_BASE/zdm/zdm_db_unique_name_job-id/zdm/log`.

If the Zero Downtime Migration service does not start, then check the Zero Downtime Migration service logs for process start-up errors to determine the cause of the error reported. The Zero Downtime Migration service log can be found at `$ZDM_BASE/crsdata/zdm_service_host/rhp/zdmserver.log.0`.

Revisando el log del servicio ZDM (en /u01/app/zdm/zdmbase/crsdata/exaadm02vm01/rhp/zdmserver.log.0), se encontró el comando exacto que ZDM estaba intentando ejecutar:

```
[JSChChannel.execCommandInternal] command is cd /tmp/; /usr/bin/sudo /bin/sh -c
  '(ORACLE_HOME=/u01/app/oracle/product/19.0.0.0/dbhome_1
   /u01/app/oracle/product/19.0.0.0/dbhome_1/bin/tnsping vmtest-scan:1521 )'
...
OutputString: TNS Ping Utility for Linux: Version 19.0.0.0.0
TNS-03505: Failed to resolve name
```

Causa: Para el chequeo de conectividad por el puerto 1521, ZDM no usa los hostnames "de conveniencia" configurados manualmente en /etc/hosts, sino los nombres de servicio asociados al SCAN tanto del origen como del destino (en este caso, vmtest-scan). Ese alias SCAN no estaba resuelto.

Solución: Se agregaron los hostnames de los SCAN de origen y destino al archivo /etc/hosts en ambos servidores (ver sección 6.2). Adicionalmente, se configuró la wallet en ambas bases de datos —origen y destino—, ya que la guía oficial de ZDM indica que es obligatorio que las dos tengan la wallet correctamente configurada antes de iniciar la migración.

🔴 PROBLEMA — Error de master key ausente en la wallet

Al relanzar la migración, el proceso falló durante la restauración de datafiles:

```
exaadm02vm01: ... Oracle ZDM ONLINE PHYSICAL migration failed
exaadm02vm01: ... Failed at phase: ZDM_RESTORE_TGT
PRGZ-3163 : Restoration and encryption of data files failed
```

```
<EXCEPTION>
RMAN> RUN {
  ALLOCATE CHANNEL ZDM_JOB_39_C1 DEVICE TYPE DISK;
  ALLOCATE CHANNEL ZDM_JOB_39_C2 DEVICE TYPE DISK; ...
channel ZDM_JOB_39_C4: restoring datafile 00009 to +DATA/.../system.272....
RMAN-03009: failure of IRESTORE command on ZDM_JOB_39_C4 channel
ORA-19849: error while reading backup piece from service ZDM_SRC_JOB_39
ORA-45919: no key present in wallet
ORA-19660: some files in the backup set could not be verified
ORA-19661: datafile 9 could not be verified due to corrupt blocks
  [ ... el mismo patrón de error se repite para varios datafiles ... ]
```

Causa: Si bien las wallets habían sido configuradas antes del lanzamiento, no se tuvo en cuenta que, tras un reinicio de la base de datos CDB, las PDBs habían quedado en estado MOUNTED en lugar de READ WRITE (ver sección 4.3). Al iniciar la migración, ZDM intentó armar la configuración de Data Guard con esa configuración incompleta de wallet/master key, lo que dejó la base de datos destino inutilizable.

Solución: Se recrearon tanto la base de datos origen como la base de datos destino, y se validó explícitamente —antes de relanzar la migración— que todas las PDBs quedaran en estado READ WRITE y que la master key estuviera establecida en el CDB y en cada PDB.

🔴 PROBLEMA — Error de configuración del path de la wallet en sqlnet.ora

Con las master keys y wallets ya correctamente configuradas, se relanzó la migración y volvió a fallar en la restauración:

```
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of restore command
RMAN-03009: failure of IRESTORE command on ZDM_JOB_39_C2 channel
ORA-19849: error while reading backup piece from service ZDM_SRC_JOB_39
ORA-45919: no key present in wallet
ORA-19660: some files in the backup set could not be verified
ORA-19661: datafile 12 could not be verified due to corrupt blocks
  [ ... mismo patrón repetido para varios datafiles y canales ... ]
```

A pesar de que las wallets y master keys ya estaban correctamente configuradas (con las PDBs abiertas), el error persistía. Revisando los logs generados por ZDM en origen y destino (ubicación: /u01/app/oracle/zdm/zdm_DBTDR_TGT_42/zdm/log), se encontró que la herramienta apuntaba a un path distinto para la wallet:

```
[jobid-42][ERROR] Keystore location
  /opt/oracle/dcs/commonstore/wallets/tde/DBTDR_TGT/tde/ does not exists
  for database 'DBTDR_TGT'
[jobid-42][ERROR] Reporting error:
```

Causa: El path de wallet que ZDM esperaba en el destino (convención propia del Base Database Service, bajo /opt/oracle/dcs/commonstore/wallets/tde/<db_unique_name>/tde/) no coincidía con la ubicación real donde se habían creado las wallets ni con lo configurado en sqlnet.ora.

Solución: Se verificó que la ruta esperada no existía en el servidor destino, y se creó manualmente:

```
[oracle@vmtest tde]$ file /opt/oracle/dcs/commonstore/wallets/tde/DBTDR_TGT/tde/
/opt/oracle/dcs/commonstore/wallets/tde/DBTDR_TGT/tde/: cannot open
  '/opt/oracle/dcs/commonstore/wallets/tde/DBTDR_TGT/tde/' (No such file or directory)

[oracle@vmtest tde]$ mkdir -p /opt/oracle/dcs/commonstore/wallets/tde/DBTDR_TGT/tde/
```

Luego se copiaron las wallets ya creadas a esa carpeta:

```
[oracle@vmtest tde]$ ls -lrt
-rw------- 1 oracle asmadmin 2553 Aug 11 14:48 ewallet_2025081114485741.p12
-rw------- 1 oracle asmadmin 3993 Aug 11 15:15 ewallet_2025081115152697.p12
-rw------- 1 oracle asmadmin 5465 Aug 11 15:15 ewallet.p12
-rw------- 1 oracle asmadmin 5510 Aug 11 15:15 cwallet.sso

[oracle@vmtest tde]$ cp -v * /opt/oracle/dcs/commonstore/wallets/tde/DBTDR_TGT/tde/
```

Y finalmente se ajustó sqlnet.ora para que ENCRYPTION_WALLET_LOCATION apuntara a ese mismo path:

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

Con estos tres ajustes resueltos, la migración se relanzó y completó exitosamente (ver resultado en la sección 7.5).

9. Buenas prácticas y recomendaciones

- **Corré siempre el -eval primero.** Es la forma más económica de encontrar problemas de conectividad, wallets o configuración antes de arriesgar la base destino.
- **Verificá el estado de las PDBs antes de migrar.** Todas deben estar READ WRITE, con la master key establecida — no solo MOUNTED.
- **Agregá los alias SCAN al /etc/hosts, no solo los hostnames.** ZDM valida conectividad de listener usando el nombre SCAN, no el hostname de conveniencia que le pongas al servidor.
- **Respetá la convención de paths de wallet del Base Database Service en el destino.** En OCI, el path esperado sigue el patrón /opt/oracle/dcs/commonstore/wallets/tde/<db_unique_name>/tde/. Si la wallet se creó en otro lugar, hay que copiarla o recrearla ahí y alinear sqlnet.ora.
- **Guardá la ubicación de los logs de ZDM antes de necesitarlos.** El log del servicio (zdmserver.log) y el log de cada job (bajo .../zdm/zdmbase/chkbase/scheduled/) son la fuente más confiable cuando el mensaje de error en pantalla no alcanza para diagnosticar.
- **Alineá character set, edición y timezone entre origen y destino antes de migrar.** Corregirlos después de la migración implica downtime adicional que ZDM no puede evitar.
- **Tené un plan si la migración falla a mitad de camino.** ZDM limpia la configuración de Data Guard temporal al finalizar (fases ZDM_CLEANUP_SRC/TGT), pero si un job queda a medio camino, revisá el estado de Data Guard en ambos lados antes de relanzar.

Conclusión y próximos pasos

ZDM cumple lo que promete: una migración física con una ventana de downtime acotada al tiempo del switchover, apoyada en herramientas ya conocidas por cualquier DBA Oracle. La parte que más tiempo insume, en la práctica, no es el comando de migración en sí, sino dejar bien alineados los prerrequisitos: wallets, PDBs abiertas, resolución de nombres SCAN y paths de configuración específicos de OCI.

Como próximos pasos, según el proyecto, puede ser útil profundizar en:

- Migraciones lógicas con ZDM (Data Pump / GoldenGate) para escenarios donde el físico no aplica.
- Migración OFFLINE con ZDM, para bases donde una ventana de downtime mayor es aceptable.
- Automatización del response file y del lanzamiento del job con scripts, para migraciones recurrentes o en lote.
- Validación post-migración: verificación de servicios, performance baseline y pruebas de aplicación contra la base ya migrada.

Recursos adicionales

- Documentación oficial de Oracle Zero Downtime Migration
- Guía de migración física con ZDM
- Documentación de Oracle Data Guard
- Documentación de Oracle Base Database Service en OCI