DB Migration to the Cloud

Key Considerations

Deciding the right migration method is a very challenging and complex task and we have to consider several factors before arriving at the right migration method.

We should focus on some point before con considerations for migrating a database to the cloud:

- **Database size:** This is the single most important factor along with available downtime.
- **Database version and edition:** Cloud providers do not support all editions or older DB versions like Oracle 10g and below, hence comparing the source DB version and edition with the allowed/available one is important.
- **ARCHIVELOG mode:** Migration methods such as RMAN hot backup, DataGuard-based migration, and many other methods require databases to be running in ARCHIVELOG mode.

This is why this information should be captured before deciding the migration method.

- **Migration level:** Migration can be at different levels, such as DB server, DB schema, or even the table level.
- **OS endianness (Little endian/ big endian):** Cloud providers do not support all OSs, hence chances of having to do a cross-platform migration is very high. For cross-platform migration, it is very important to know the OS endianness so that the appropriate migration method can be selected.
- **32-bit/64-bit:** 32-bit servers are not very common, but if you have very old legacy systems running on 32-bit, then you need to consider moving them as-is or converting them to 64-bit systems.
- **Application nature (mission-critical/performance sensitive/big data):** The application nature defines the downtime availability, hence it affects the migration method used.
- **Available downtime and available time for migration:** Some database migrations might have zero or very little downtime. This is an important factor to remember while determining the migration method Network bandwidth between on-premise and the cloud: Network bandwidth plays an important role with large database migrations.
- **Application compatibility:** While doing database migration, you need to consider whether the application is compatible with the proposed cloud deployment model.
- **Cloud provider (Oracle Cloud, AWS, or Microsoft Azure):** Different cloud vendors have different migration methods. Amazon AWS has a database migration service called DMS, which is not available in Oracle Cloud.

- **Cloud deployment model (IaaS, PaaS, DBaaS):** The migration method depends on the database deployment model, for example, you can't do a RMAN backup restore in Oracle Schema as a service deployment.
- **Cloud type (public/private):** Migration methods for a private cloud in your own datacenter will be different than for a public cloud.

Migration Lifecycle

Migration is multistep process; the different stages are like :

- **Analyse :** In the Analyze phase, you go through the customer requirements and key considerations.
- **Identify :** These inputs help in the next phase, called Identify, where you identify the migration method.
- **Prepare** : The Prepare phase is all about getting ready for migration by having things in place. For example, you prepare the target environment for DB migration.
- **Migrate**
- **Validate**

Migration Approach

At a high level, all DB migration method consists of three phases, as shown in
BAckup
Transfer
Restore

Backup
Transfer
Restore

Migrating Method

There are various ways you can do DB migration to Oracle or Amazon Cloud, including

- Export/Import,
- using SQL Developer,
- and RMAN based restore.

Other methods are:

- Create a DataGuard in the cloud
- Perform on-premise backups to the cloud and restore the cloud backup to create the DB in the cloud

- Perform a DB server VM backup and restore the machine backup to the cloud, then restore it as new cloud VM
- Use Oracle Golden Gate-based DB replication to the cloud
- For AWS, Data Migration Services (DMS) is also a very effective method

Comparison of Migration Methods

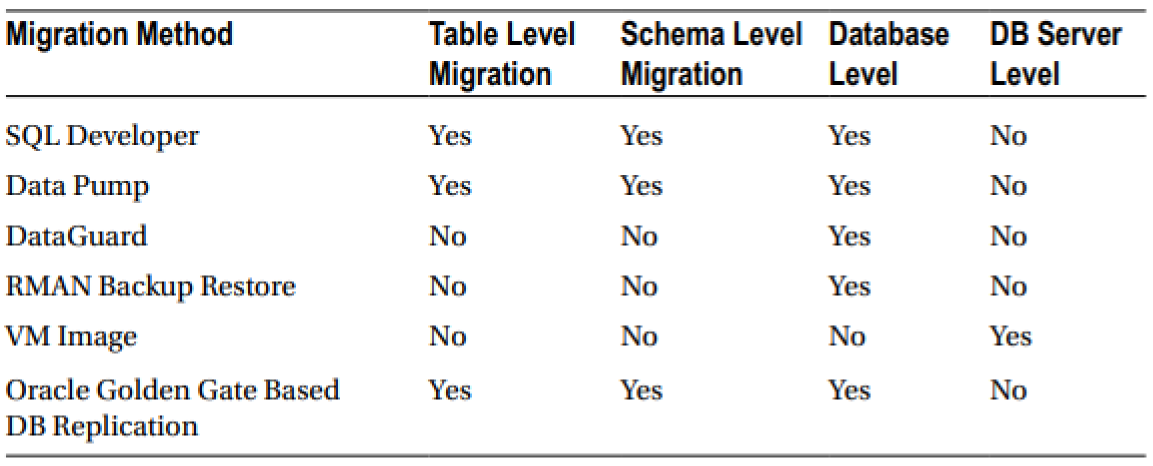


Table 7-1. *Migration Methods Comparison*

Migration Method	Table Level Migration	Schema Level Migration	Database Level	DB Server Level
SQL Developer	Yes	Yes	Yes	No
Data Pump	Yes	Yes	Yes	No
DataGuard	No	No	Yes	No
RMAN Backup Restore	No	No	Yes	No
VM Image	No	No	No	Yes
Oracle Golden Gate Based DB Replication	Yes	Yes	Yes	No

Database migration from On-premise to AWS RDS

Method: Importing using Oracle Data Pump

There are two methods.

- Importing with Data pump using S3 bucket
- Importing with Data Pump with database link

Importing with Data Pump with database link

###START###

Prerequisite on source

- You must have execute privileges on the DBMS_FILE_TRANSFER and DBMS_DATAPUMP packages.

- You must have write privileges to the DATA_PUMP_DIR directory on the source DB instance.
- You must ensure that you have enough storage space to store the dump file on the source instance and the target DB instance.
- Get the schema size from source database

```
set linesize 150
set pagesize 5000
col owner for a15
col segment_name for a30
col segment_type for a20
col TABLESPACE_NAME for a30
clear breaks
clear computes
compute sum of SIZE_IN_GB on report
break on report

select OWNER,sum(bytes)/1024/1024/1000 "SIZE_IN_GB" from
dba_segments where owner in ('APPLSYS') group by owner order by
owner;
```

- Get the count of object (group by object type)

```
set lines 333 pages 111
col OWNER for a19
col OBJECT_TYPE for a21

select owner, object_type, count(*) from dba_objects where owner in
('APPLSYS') and status='VALID' group by owner, object_type order by
owner;
```

- Get the user DDL, role user default tablespace and tablespace details for owner related objects.

```
set head off
set pages 0
set long 9999999

SQL> select Dbms_metadata.get_ddl('USER','APPLSYS') from dual;

SQL> SELECT
DBMS_METADATA.GET_GRANTED_DDL('ROLE_GRANT','APPLSYS
') FROM DUAL;

SQL> SELECT
DBMS_METADATA.GET_GRANTED_DDL('SYSTEM_GRANT','APPLS
YS') FROM DUAL;
```

- Get the user TABLESPACE Details.

```
SELECT owner,tablespace_name FROM dba_segments WHERE
owner='APPLSYS';

SQL> EXEC DBMS_TTS.TRANSPORT_SET_CHECK(ts_list =>
'APPS_TS_TX_DATA');

SQL> SELECT * FROM transport_set_violations;
```

- Get the TABLESPACE usage .

```
select t.tablespace, t.totalspace as " Totalspace(MB)",
round((t.totalspace-nvl(fs.freespace,0)),2) as "Used
Space(MB)", nvl(fs.freespace,0) as "Freespace(MB)",
round(((t.totalspace-nvl(fs.freespace,0))/t.totalspace)*100,2)
as "%Used", round((nvl(fs.freespace,0)/t.totalspace)*100,2) as
"% Free" from (select round(sum(d.bytes)/(1024*1024)) as
totalspace,d.tablespace_name tablespace from dba_data_files d
group by d.tablespace_name) t, (select
round(sum(f.bytes)/(1024*1024)) as freespace,f.tablespace_name
tablespace from dba_free_space f group by f.tablespace_name)
fs where t.tablespace=fs.tablespace (+) order by "% Free";
```

Prerequisite on target

- Create the required tablespaces before you import the data as per above output

  ```
  select tablespace_name from dba_data_files;

  select b.tablespace_name, tbs_size SizeMb, a.free_space FreeMb
  from    (select tablespace_name, round(sum(bytes)/1024/1024 ,2) as
  free_space
        from dba_free_space
        group by tablespace_name) a,
       (select tablespace_name, sum(bytes)/1024/1024 as tbs_size
        from dba_data_files
        group by tablespace_name) b
  where a.tablespace_name(+)=b.tablespace_name;
  ```

  Note: Increase the size of tablespace before start the import process (Reference)

- If the user account into which the data is imported doesn't exist, create the user account and grant the necessary permissions and roles. If you plan to import data into multiple user schemas, create each user account and grant the necessary privileges and roles to it. As per above output

Step 1 (Optional) :Create export user and Grant privileges to the user on the source database - It is optional , We can take the export as SYS user

Note
If the source database is an Amazon RDS instance, you can skip this step. You use your Amazon RDS master user account to perform the export.

The following commands create a new user and grant the necessary permissions.

```
CREATE USER export_user IDENTIFIED BY user123;
GRANT CREATE SESSION, CREATE TABLE, CREATE DATABASE LINK TO
export_user;
ALTER USER export_user QUOTA unlimited  on APPS_TS_TX_DATA;
GRANT READ, WRITE ON DIRECTORY data_pump_dir TO export_user;
GRANT SELECT_CATALOG_ROLE TO export_user;
GRANT EXECUTE ON DBMS_DATAPUMP TO export_user;
GRANT EXECUTE ON DBMS_FILE_TRANSFER TO export_user;
```

Step 2: Create a dump file using expdp

We have multiple option to take

1.st option if we want to take backup in a single file with unlimited size

```
expdp schemas=APPLSYS directory=DATA_PUMP_DIR
dumpfile=exp_APPLSYS.dmp logfile=expAPPLSYS.log
```

2nd option , We can take backup in multiple file with limit size

```
Create parameter File

DUMPFILE=SOURCE_SID_FULL_%u_11JULY.DMP
LOGFILE=SOURCE_SID_FULL_11JULY.LOG
SCHEMAS=SCHEMA_1,SCHEMA_2
DIRECTORY=DATA_PUMP_DIR
PARALLEL=16
COMPRESSION=ALL   << If the destination is Standard edition, don't use
this
FILESIZE=5G

Run the export backup

expdp parfile=exp_on_prem.pf
```

Step 3: Validate the TNSPING Response from Source to target

```
tnsping database-1.ccknm69zyn6v.us-east-1.rds.amazonaws.com:1521/ORCL
```

Step 4: Create Database Link on Source database

```
CREATE DATABASE LINK onprem_to_rds
CONNECT TO admin IDENTIFIED BY indiandba123
USING
'(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=database-1.ccknm6
9zyn6v.us-east-1.rds.amazonaws.com)
(PORT=1521))(CONNECT_DATA=(SID=ORCL)))';
```

Step 5: Copy the exported dump file to the target DB instance using DBMS_FILE_TRANSFER

Checks Some point before transfer the export dump:

1. Validate the Available disk storage on target RDS to store export dump files. (Reference)

If it's just a single file

```
BEGIN
  DBMS_FILE_TRANSFER.PUT_FILE(
    source_directory_object      => 'DATA_PUMP_DIR',
    source_file_name             => 'sample.dmp',
    destination_directory_object  => 'DATA_PUMP_DIR',
    destination_file_name        => 'sample_copied.dmp',
    destination_database         => 'to_rds' );
END;
```

```
/
```

If it’s multiple export file

Depending on the number of files, edit the PL/SQL blocks below - change the numbers - several in one loop, and open multiple command prompts or terminal sessions and run several, covering all files – change the file date/names accordingly, and run it in SQLPlus:

```
This does files 01 to 09
begin
for i in 1..9 loop
DBMS_FILE_TRANSFER.PUT_FILE(
   source_directory_object      => 'DATA_PUMP_DIR',
   source_file_name             =>
'SOURCE_SID_FULL_0'||i||'_11JULY.DMP',
   destination_directory_object  => 'DATA_PUMP_DIR',
   destination_file_name        =>
'SOURCE_SID_FULL_0'||i||'_11JULY.DMP',
   destination_database         => 'onprem_to_rds'
 );
 end loop;
end;
/

This does files 10 to 15

begin
for i in 1..9 loop
DBMS_FILE_TRANSFER.PUT_FILE(
   source_directory_object      => 'DATA_PUMP_DIR',
   source_file_name             =>
'SOURCE_SID_FULL_1'||i||'_11JULY.DMP',
   destination_directory_object  => 'DATA_PUMP_DIR',
   destination_file_name        =>
'SOURCE_SID_FULL_1'||i||'_11JULY.DMP',
   destination_database         => 'onprem_to_rds'
 );
 end loop;
end;
/
```

[Yet Another Oracle DBA Blog: Migrating a database from an on prem server to Amazon RDS]

Step 6: **Validate the data_pump_dir and progress of transfer details on target**

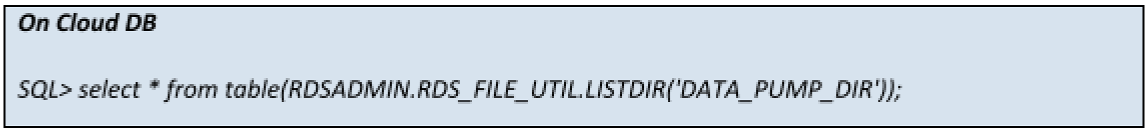
```
select * from dba_directories where directory_name='DATA_PUMP_DIR';

SELECT * FROM TABLE(rdsadmin.rds_file_util.listdir('DATA_PUMP_DIR')) ORDER BY MTIME;
```

Step 7: There are two option for import

Option 1: Run IMPORT command from Source system

1. **To know the DATA_PUMP_DIRECTORY local on RDS**



On Cloud DB

```
SQL> select * from table(RDSADMIN.RDS_FILE_UTIL.LISTDIR('DATA_PUMP_DIR'));
```

2. **Validate the TNSPING Response from Source to target using TNSNAMES.ora file entry**

```
RDS =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = database-1.ccknm69zyn6v.us-east-1.rds.amazonaws.com)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = ORCL)
    )
  )
```

Then validate the tnsping RDS response

3. **Validate the data_pump_dir details on target**

```
select * from dba_directories where directory_name='DATA_PUMP_DIR';

SELECT * FROM TABLE(rdsadmin.rds_file_util.listdir('DATA_PUMP_DIR'))
ORDER BY MTIME;
```

4. **Now import the data in the cloud DB with a user who has the required privileges. On the local DB:**

   If it's just a single file

```
impdp admin/indiandba123@rds directory=DATA_DUMP_DIR
dumpfile=exp_APPLSYS.dmp full=yes
```

   If it's just a multiple file

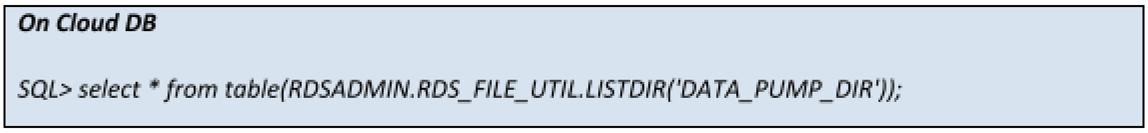
```
impdp admin/indiandba123@rds
DUMPFILE=SOURCE_SID_FULL_%u_11JULY.DMP
logfile=import_SOURCE_SID_FULL_11JULY.log  parallel=16 full=y
directory=DATA_PUMP_DIR PARTITION_OPTIONS=DEPARTITION
<< use this if the target is SE and source has partitions
```

   Issue: Import Failed with multiple dump file
   Solution: Data Pump Import, IMPDP, Fails with Errors ORA-39246 and ORA-39059 (Doc ID 1512192.1)

Option 2: Import the data file to the target DB instance using DBMS_DATAPUMP

Connect to the DB instance with the Amazon RDS master user account to perform the import.

1. **To know the DATA_PUMP_DIRECTORY local on RDS**



```
On Cloud DB

SQL> select * from table(RDSADMIN.RDS_FILE_UTIL.LISTDIR('DATA_PUMP_DIR'));
```

2. **Validate the data_pump_dir details on target**

```
select * from dba_directories where directory_name='DATA_PUMP_DIR';

SELECT * FROM TABLE(rdsadmin.rds_file_util.listdir('DATA_PUMP_DIR'))
ORDER BY MTIME;
```

3. **Now import the data in the cloud DB with a user who has the required privileges. On the local DB:**

```
DECLARE
  v_hdnl NUMBER;
BEGIN
  v_hdnl := DBMS_DATAPUMP.OPEN(
    operation => 'IMPORT',
    job_mode  => 'SCHEMA',
    job_name  => null);
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl,
    filename  => 'sample_copied.dmp',
    directory => 'DATA_PUMP_DIR',
    filetype  => dbms_datapump.ku$_file_type_dump_file );
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl,
    filename  => 'sample_imp.log',
    directory => 'DATA_PUMP_DIR',
    filetype  => dbms_datapump.ku$_file_type_log_file);
  DBMS_DATAPUMP.METADATA_FILTER(v_hdnl,'SCHEMA_EXPR','IN
(''SCHEMA_1'')');
  DBMS_DATAPUMP.START_JOB(v_hdnl);
END;
/
```

Step 8: Validate the object in Target RDS Instance

Verify the data in the cloud DB using SQL Developer. While copying the dump file may take more time, the import is expected to be instantaneous since it is done locally in the cloud. In case of data mismatch, specific table/objects may need to be imported again. The exact steps for fixing the partially successful migration depend on the specific issues encountered.

Step 9: Clean up

After the data has been imported, you can delete the files that you don't want to keep. You can list the files in DATA_PUMP_DIR using the following command.

```
SELECT * FROM TABLE(rdsadmin.rds_file_util.listdir('DATA_PUMP_DIR'))
ORDER BY MTIME;
```

To delete files in DATA_PUMP_DIR that you no longer require, use the following command.

```
EXEC UTL_FILE.FREMOVE('DATA_PUMP_DIR','<file name>');
```

Database Migration from On-premise to AWS EC2

Supported platforms

You can query the V$TRANSPORTABLE_PLATFORM view to see the platforms that are supported and to determine each platform's endian format (byte ordering).

```
SQL> COLUMN PLATFORM_NAME FORMAT A32
SQL> SELECT * FROM V$TRANSPORTABLE_PLATFORM;

PLATFORM_ID PLATFORM_NAME                    ENDIAN_FORMAT
----------- -------------------------------- --------------
```

```
 1 Solaris[tm] OE (32-bit)          Big
 2 Solaris[tm] OE (64-bit)          Big
 7 Microsoft Windows IA (32-bit)    Little
10 Linux IA (32-bit)                Little
 6 AIX-Based Systems (64-bit)       Big
 3 HP-UX (64-bit)                   Big
 5 HP Tru64 UNIX                    Little
 4 HP-UX IA (64-bit)                Big
11 Linux IA (64-bit)                Little
15 HP Open VMS                      Little
 8 Microsoft Windows IA (64-bit)    Little
 9 IBM zSeries Based Linux          Big
13 Linux 64-bit for AMD             Little
16 Apple Mac OS                     Big
12 Microsoft Windows 64-bit for AMD Little
17 Solaris Operating System (x86)   Little
```

If the source platform and the target platform are of different endianness, then an additional step must be done on either the source or target platform to convert the tablespace being transported to the target format. If they are of the same endianness, then no conversion is necessary and tablespaces can be transported as if they were on the same platform.

Compare Available Method

Available Method		
Using Data Pump	Data Pump Supports cross platform backup and restore.	
Using Full Transportable Export Import	● The endian should be the same on source and target for this method.	
Using Transportable Tablespace	● Source and Target database should be on same Operating System version ● Endian Format should be the same on source and target.	
Using RMAN Transport Tablespace	● Supports cross platform backup and restore. ● No need to take the source tablespaces in read only mode, which means this method can be applied to production databases	

Using RMAN Convert Transportable Tablespace	● Source and Target database can be on different Operating System version ● Endian Format can be different on source and target.	

Migrate to EC2 using Data Pump

• Oracle Data Pump is a logical backup and restore utility
• Data Pump Supports cross platform backup and restore.
• We can perform backup/restore at table/user/tablespace/database level
• Data Pump supports parallel options , which perform the backup and restore using a parallel worker process and thus helps in faster job completion.

Lab Setup Details

	Source	**Target**
OS	Window	Linux
Location	On premise	AWS
Version	19c	19c
Database Name	ONPREM	PROD
Database Type	PDB	PDB

Workflow

• In source identify the Schema to be migrated
• Create directory object to store the backup
• Take the data pump export (expdp) backup of the schema
• Create directory object on target to store the backup
• Copy the dump files to target
• Perform data pump import (impdp) to restore the backup of the schema

Prerequisite on source

- You must have execute privileges on the DBMS_FILE_TRANSFER and DBMS_DATAPUMP packages.
- You must have write privileges to the DATA_PUMP_DIR directory on the source DB instance.
- You must ensure that you have enough storage space to store the dump file on the source instance and the target DB instance.
- Get the schema size from source database

```
set linesize 150
set pagesize 5000
col owner for a15
col segment_name for a30
col segment_type for a20
col TABLESPACE_NAME for a30
clear breaks
clear computes
compute sum of SIZE_IN_GB on report
break on report

select OWNER,sum(bytes)/1024/1024/1000 "SIZE_IN_GB" from
dba_segments where owner in ('APPLSYS') group by owner order by
owner;
```

- Get the count of object (group by object type)

```
set lines 333 pages 111
col OWNER for a19
col OBJECT_TYPE for a21

select owner, object_type, count(*) from dba_objects where owner in
('APPLSYS') and status='VALID' group by owner, object_type order by
owner;
```

- Get the user DDL, role user default tablespace and tablespace details for owner related objects.

```
set head off
set pages 0
set long 9999999

SQL> select Dbms_metadata.get_ddl('USER','APPLSYS') from dual;

SQL> SELECT
DBMS_METADATA.GET_GRANTED_DDL('ROLE_GRANT','APPLSYS
') FROM DUAL;

SQL> SELECT
DBMS_METADATA.GET_GRANTED_DDL('SYSTEM_GRANT','APPLS
YS') FROM DUAL;
```

- Get the user TABLESPACE Details.

```
SELECT owner,tablespace_name FROM dba_segments WHERE
owner='APPLSYS';

SQL> EXEC DBMS_TTS.TRANSPORT_SET_CHECK(ts_list =>
'APPS_TS_TX_DATA');

SQL> SELECT * FROM transport_set_violations;
```

- Get the TABLESPACE usage .

```
select t.tablespace, t.totalspace as " Totalspace(MB)",
round((t.totalspace-nvl(fs.freespace,0)),2) as "Used
Space(MB)", nvl(fs.freespace,0) as "Freespace(MB)",
round(((t.totalspace-nvl(fs.freespace,0))/t.totalspace)*100,2)
as "%Used", round((nvl(fs.freespace,0)/t.totalspace)*100,2) as
"% Free" from (select round(sum(d.bytes)/(1024*1024)) as
totalspace,d.tablespace_name tablespace from dba_data_files d
group by d.tablespace_name) t, (select
round(sum(f.bytes)/(1024*1024)) as freespace,f.tablespace_name
tablespace from dba_free_space f group by f.tablespace_name)
fs where t.tablespace=fs.tablespace (+) order by "% Free";
```

Prerequisite on target

- Create the required tablespaces before you import the data as per above output

  select tablespace_name from dba_data_files;

  ```
  select b.tablespace_name, tbs_size SizeMb, a.free_space FreeMb
  from  (select tablespace_name, round(sum(bytes)/1024/1024 ,2) as free_space
        from dba_free_space
        group by tablespace_name) a,
       (select tablespace_name, sum(bytes)/1024/1024 as tbs_size
        from dba_data_files
        group by tablespace_name) b
  where a.tablespace_name(+)=b.tablespace_name;
  ```

  Note: Increase the size of tablespace before start the import process (Reference)

- If the user account into which the data is imported doesn't exist, create the user account and grant the necessary permissions and roles. If you plan to import data into multiple user schemas, create each user account and grant the necessary privileges and roles to it. As per above output

Create tablespace
Create user
Grant to users

Step 1 (Optional) :Create export user and Grant privileges to the user on the source database - It is optional , We can take the export as SYS user

Note
If the source database is an Amazon RDS instance, you can skip this step. You use your Amazon RDS master user account to perform the export.

The following commands create a new user and grant the necessary permissions.

```
CREATE USER export_user IDENTIFIED BY user123;
GRANT CREATE SESSION, CREATE TABLE, CREATE DATABASE LINK TO
export_user;
ALTER USER export_user QUOTA unlimited  on APPS_TS_TX_DATA;
GRANT READ, WRITE ON DIRECTORY data_pump_dir TO export_user;
GRANT SELECT_CATALOG_ROLE TO export_user;
GRANT EXECUTE ON DBMS_DATAPUMP TO export_user;
GRANT EXECUTE ON DBMS_FILE_TRANSFER TO export_user;
```

Step 2: Create a dump file using expdp

We have multiple option to take

1.st option if we want to take backup in a single file with unlimited size

```
expdp schemas=APPLSYS directory=DATA_PUMP_DIR
dumpfile=exp_APPLSYS_19092023.dmp logfile=expAPPLSYS_19092023.log
```

2nd option , We can take backup in multiple file with limit size

```
Create parameter File
```

```
DUMPFILE=SOURCE_SID_FULL_%u_11JULY.DMP
LOGFILE=SOURCE_SID_FULL_11JULY.LOG
SCHEMAS=SCHEMA_1,SCHEMA_2
DIRECTORY=DATA_PUMP_DIR
PARALLEL=16
COMPRESSION=ALL   << If the destination is Standard edition, don't use
this
FILESIZE=5G

Run the export backup

expdp parfile=exp_on_prem.pf
```

Step 3: Validate the TNSPING Response from Source to target

```
tnsping database-1.ccknm69zyn6v.us-east-1.rds.amazonaws.com:1521/ORCL
```

Step 4: Create Database Link on Source database

```
CREATE DATABASE LINK onprem_to_EC23
CONNECT TO admin IDENTIFIED BY admin123
USING
'(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=ip-10-0-11-164.ap-south-1.
compute.internal)
(PORT=1521))(CONNECT_DATA=(SID=PROD)))';
```

Step 5: We have a tree option to transfer export backup dump.

1. Through FTP
2. Through S3 Bucket
3. Copy the exported dump file to the target DB instance using DBMS_FILE_TRANSFER

Here we use DBMS_FILE_TRANSFER

Checks Some point before transfer the export dump:

1. Validate the Available disk storage on target EC2 to store export dump files.

If it's just a single file

```
BEGIN
  DBMS_FILE_TRANSFER.PUT_FILE(
    source_directory_object      => 'DATA_PUMP_DIR',
    source_file_name             => 'EXP_APPLSYS_19092023.DMP',
    destination_directory_object  => 'DATA_PUMP_DIR',
    destination_file_name         => 'EXP_APPLSYS_19092023.DMP',
    destination_database          => 'onprem_to_EC2' );
END;
/
```

If it's multiple export file

Depending on the number of files, edit the PL/SQL blocks below - change the numbers - several in one loop, and open multiple command prompts or terminal sessions and run several, covering all files – change the file date/names accordingly, and run it in SQLPlus:

This does files 01 to 09

```
begin
for i in 1..9 loop
DBMS_FILE_TRANSFER.PUT_FILE(
    source_directory_object      => 'DATA_PUMP_DIR',
    source_file_name             =>
'SOURCE_SID_FULL_0'||i||'_11JULY.DMP',
    destination_directory_object  => 'DATA_PUMP_DIR',
    destination_file_name         =>
'SOURCE_SID_FULL_0'||i||'_11JULY.DMP',
    destination_database          => 'onprem_to_rds'
  );
  end loop;
end;
/
```

```
This does files 10 to 15

begin
for i in 1..9 loop
DBMS_FILE_TRANSFER.PUT_FILE(
   source_directory_object      => 'DATA_PUMP_DIR',
   source_file_name             =>
'SOURCE_SID_FULL_1'||i||'_11JULY.DMP',
   destination_directory_object  => 'DATA_PUMP_DIR',
   destination_file_name        =>
'SOURCE_SID_FULL_1'||i||'_11JULY.DMP',
   destination_database         => 'onprem_to_rds'
  );
  end loop;
end;
/
```

Step 6: **Validate the data_pump_dir and progress of transfer details on target**

```
select * from dba_directories where directory_name='DATA_PUMP_DIR';
```

Step 7: There are two option for import

Option 1: Run IMPORT command from Source system

1. **Validate the TNSPING Response from Source to target using TNSNAMES.ora file entry**

```
EC2 =
  (DESCRIPTION =
                    (ADDRESS = (PROTOCOL = TCP)(HOST =
database-1.ccknm69zyn6v.us-east-1.rds.amazonaws.com)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = PROD)
    )
  )
```

Then validate the tnsping RDS response

2. **Validate the data_pump_dir details on target**

```
select * from dba_directories where directory_name='DATA_PUMP_DIR';

SELECT * FROM TABLE(rdsadmin.rds_file_util.listdir('DATA_PUMP_DIR'))
ORDER BY MTIME;
```

3. **Now import the data in the cloud DB with a user who has the required privileges. On the local DB:**

   If it's just a single file

   ```
   impdp admin/indiandba123@rds directory=DATA_DUMP_DIR
   dumpfile=exp_APPLSYS.dmp full=yes
   ```

   If it's just a multiple file

   ```
   impdp admin/indiandba123@rds
   DUMPFILE=SOURCE_SID_FULL_%u_11JULY.DMP
   logfile=import_SOURCE_SID_FULL_11JULY.log  parallel=16 full=y
   directory=DATA_PUMP_DIR PARTITION_OPTIONS=DEPARTITION
   << use this if the target is SE and source has partitions
   ```

Option 2: Import the data file to the target DB instance using DBMS_DATAPUMP

Connect to the DB instance with the Amazon EC2 master user account to perform the import.

1. **Validate the data_pump_dir details on target**

```
select * from dba_directories where directory_name='DATA_PUMP_DIR';
```

2. **Now import the data in the cloud DB with a user who has the required privileges. On the local DB:**

```
DECLARE
  v_hdnl NUMBER;
BEGIN
  v_hdnl := DBMS_DATAPUMP.OPEN(
    operation => 'IMPORT',
    job_mode  => 'SCHEMA',
    job_name  => null);
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl,
    filename  => 'sample_copied.dmp',
    directory => 'DATA_PUMP_DIR',
    filetype  => dbms_datapump.ku$_file_type_dump_file );
  DBMS_DATAPUMP.ADD_FILE(
    handle    => v_hdnl,
    filename  => 'sample_imp.log',
    directory => 'DATA_PUMP_DIR',
    filetype  => dbms_datapump.ku$_file_type_log_file);
  DBMS_DATAPUMP.METADATA_FILTER(v_hdnl,'SCHEMA_EXPR','IN
(''SCHEMA_1'')');
  DBMS_DATAPUMP.START_JOB(v_hdnl);
END;
/
```

Option 3: Run IMPORT command from Target system

```
impdp schemas=APPLSYS directory=DATA_PUMP_DIR
dumpfile=exp_APPLSYS_19092023.dmp logfile=impAPPLSYS_19092023.log
```

Step 8: Validate the object in Target RDS Instance

Verify the data in the cloud DB using SQL Developer. While copying the dump file may take more time, the import is expected to be instantaneous since it is done locally in the cloud. In case of data mismatch, specific table/objects may need to be imported again. The exact steps for fixing the partially successful migration depend on the specific issues encountered.

- Get the count of object (group by object type)

```
set lines 333 pages 111
col OWNER for a19
col OBJECT_TYPE for a21

select owner, object_type, count(*) from dba_objects where owner in
('APPLSYS') and status='VALID' group by owner, object_type order by
owner;
```

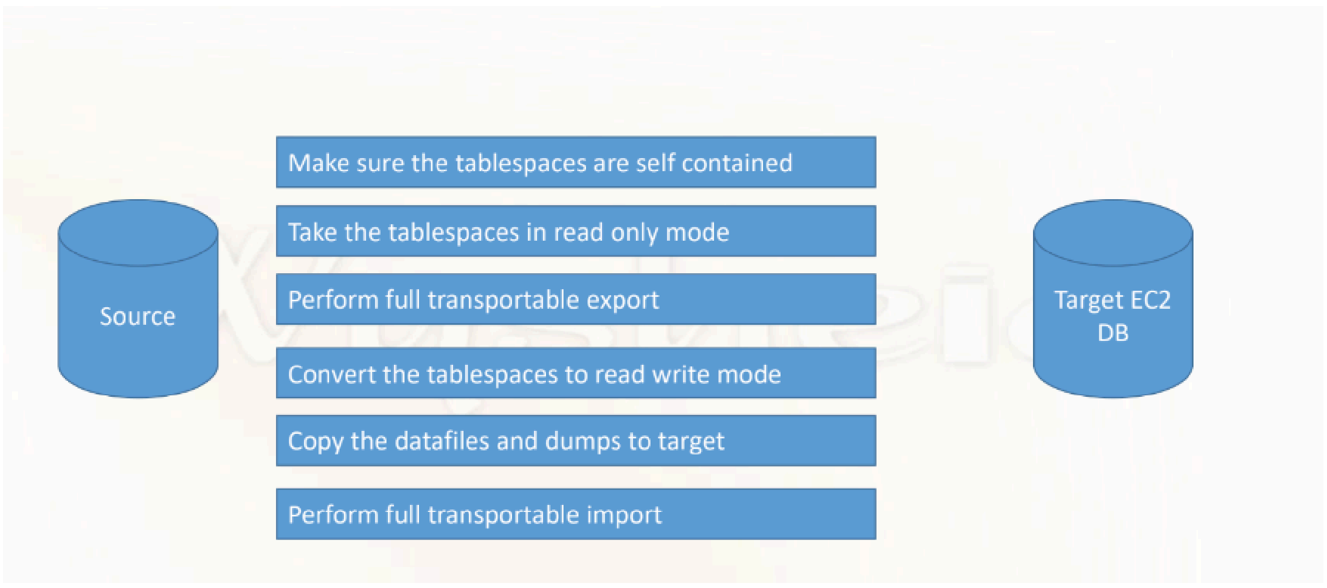
Using Full Transportable Export Import

- This method is an enhancement of "transportable tablespace" feature and more usability and easiness
- Migrate the data as data files instead of logical data as in the case of regular export
- Faster method compared to regular export as the physical files are copied from source to target
- Helps in migrating all the user defined tablespaces in single command
- The methods used is applicable for both Non-CDB as well as PDB
- Full transportable export and import can take the advantage of import over network features and migrate the tablespaces with a single command. In all cases manual copy of the data files are required

Workflow

- Create Database in AWS EC2

- Find out the endian format of the source and target.
- The endian should be the same on source and target for this method.
- In source identify the tablespaces to be migrated
- Verify that the set of tablespaces to be transported is self-contained
- Create directory object to store the backup
- Take the source tablespaces in read only mode.
- Take the full transportable export
- Make the source tablespaces into read write mode
- Copy the dump files and data files to target
- Create directory object to store the backup on target
- copy the data file to correct directories
- Perform full transportable import


Source
Make sure the tablespaces are self contained
Take the tablespaces in read only mode
Perform full transportable export
Convert the tablespaces to read write mode
Copy the datafiles and dumps to target
Perform full transportable import
Target EC2 DB

Steps:

Find The Endian Format of the source and target

• This method will work only if source and target has the same endian format

SQL> select platform_name, endian_format from v$transportable_platform where platform_id in (select platform_id from v$database);

Verify that the set of tablespaces to be transported is self-contained

```
SQL> EXECUTE
DBMS_TTS.TRANSPORT_SET_CHECK('APPS_TS_TX_DATA', TRUE);

SQL> SELECT * FROM TRANSPORT_SET_VIOLATIONS;
```

Take the tablespace read only

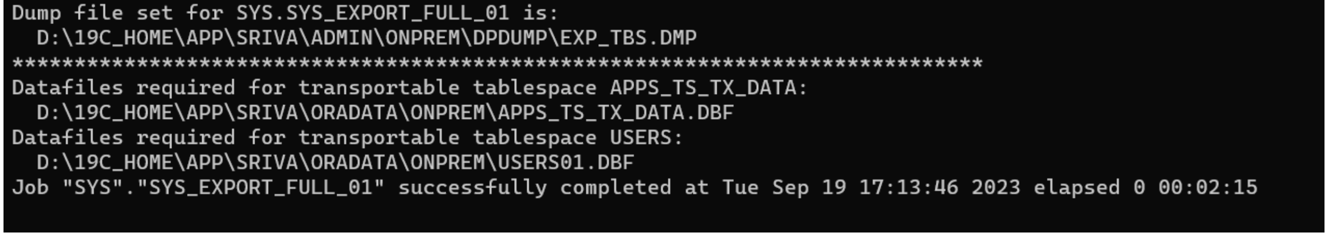
```
SQL> ALTER TABLESPACE APPS_TS_TX_DATA Read only;
```

Take the full transportable export

```
expdp  full=y transportable=always directory=DATA_PUMP_DIR
dumpfile=exp_tbs.dmp logfile=exp_tbs.log
```

• This command will take the meta data backup of all non-system tablespaces

• Transportable=ALWAYS and full=Y . These parameters tell Data pump to take full transportable export rather than conventional export Method.



```
******************************************************************************
Dump file set for SYS.SYS_EXPORT_FULL_01 is:
  D:\19C_HOME\APP\SRIVA\ADMIN\ONPREM\DPDUMP\EXP_TBS.DMP
******************************************************************************
Datafiles required for transportable tablespace APPS_TS_TX_DATA:
  D:\19C_HOME\APP\SRIVA\ORADATA\ONPREM\APPS_TS_TX_DATA.DBF
Datafiles required for transportable tablespace USERS:
  D:\19C_HOME\APP\SRIVA\ORADATA\ONPREM\USERS01.DBF
Job "SYS"."SYS_EXPORT_FULL_01" successfully completed at Tue Sep 19 17:13:46 2023 elapsed 0 00:02:15
```

Copy the files to target

- The following files to be copied.
  Full transportable export dump files
  All the data files which should be plugged in to the target database

- Any of the following methods can be used
  - Scp
  - Use s3 buckets
- Copy the data files to the correct directories to maintain the standard

Import the transportable data

- $ impdp directory=DATA_PUMP_DIR dumpfile=EXP_TBS.DMP logfile=full_tts_imp.log transport_datafiles='/oracle/oradata/PROD/APPS_TS_TX_DATA.DBF,'/oracle/oradata/PROD/USERS01.DBF'

- Import will create the users
- Cross verify if the objects and data is imported correctly

Using RMAN Convert Transportable Tablespace

- Similar to transportable tablespace using data pump
- Source and Target database can be on different Operating System version
- Endian Format can be different on source and target.

Workflow

- In source identify the Tablespaces to be migrated
- Make sure that the tablespaces are self contained
- Create the directory for directory object for the data pump
- Take the tablespaces to be transported into read only mode
- Take the meta data backup of the tablespaces using expdp with TRANSPORT_TABLESPACES=<list of tablespaces> parameter.
- Using RMAN , convert the tablespaces into the endian format of the target database

- Take the tablespaces to be transported back into read write mode
- Create a directory object on the target database.
- Copy the dump files and datafiles of the tablespaces to the target. Make sure to copy the dump files to the physical path of the directory object created before
- Import the transportable tablespace metadata using impdp with parameter TRANSPORT_DATAFILES.

Steps:

- Identify the tablespaces to be transported. In this example "mytbs"
- Check if the tablespaces are self contained.

  DBMS_TTS.TRANSPORT_SET_CHECK can be used for this.

  ```
  CONN / AS SYSDBA
  EXEC SYS.DBMS_TTS.TRANSPORT_SET_CHECK(ts_list => 'mytbs',
  incl_constraints => TRUE);

  PL/SQL procedure successfully completed.
  ```

- Check if any violations using transport_set_violations view

  ```
  SELECT * FROM transport_set_violations;
  ```

- Take the tablespaces to be exported into read only mode.

  ```
  SQL> alter tablespace mytbs read only;
  ```

- Create a directory object in the source database .

  ```
  SQL> create directory dp_dir as 'G:\wysheid\backup';
  ```

- Perform transportable export backup using expdp

  ```
  $ expdp system/oracle@sourcedb1 TRANSPORT_TABLESPACES=mytbs
  TRANSPORT_FULL_CHECK=YES DIRECTORY=dp_dir
  dumpfile=mytbs.dmp logfile=mytbs.log
  ```

```
RMAN> CONVERT TABLESPACE mytbs to platform='Linux x86 64-bit'
format 'G:\wysheid\backup\%U';
```

- Take the tablespaces to be exported back into read write mode.

```
SQL> alter tablespace mytbs read write;
```

- Create directory object on the target database

```
SQL> create directory dp_dir as '/wysheid/backup';
```

Copy the dump files and converted data files of the tablespaces to be transported to the EC2 Instance. Make sure to copy the dump file to the physical path of the directory object created

- S3
- SCP

Copy the data files to the standard location similar to that of the other data files.

- Import the transportable tablespace into the target database

```
impdp system/oracle@targetdb1 DIRECTORY=dp_dir
TRANSPORT_DATAFILES='/wysheid/backup/mytbs.dbf'
dumpfile=mytbs.dmp logfile=impdp_mytbs.log
```

- Take the target tablespace into read write mode

```
SQL> alter tablespace mytbs read write;
```

- Validate the Object after Migration

Using RMAN Transport Tablespace

- Similar to Data pump full transportable export and import, but much simplified
- No need to take the source tablespaces in read only mode, which means this method can be applied to production databases
- Supports cross platform backup and restore.
- Import on target can be done via PLSQL script or impdp command.
- RMAN uses auxiliary instance to perform restoration of datafiles , which means Point In Time based migration is possible
- There should be enough space to perform restoration data files on the source

Workflow

- In source identify the Tablespaces to be migrated
- Create the directory for storing data files and dump files to be transported
- Create the directory for Auxiliary Instance destination
- From RMAN connect to Non-CDB or Root container in the case of CDB
- Make sure that we have RMAN backup based on the planned time stamp
- Execute ‘transport tablespace “ command from RMAN
- Copy the dump files and data files to target
- Perform the restoration by any of the following methods
- Execute “impscrpt.sql” by connecting to the target DB
- Run the impdp utility by specifying transport_datafiles parameter.

Steps:

- Identify the tablespaces to be transported
- Create the tablespace destination and auxiliary destination

```
    $ rman target /
```

Here target should be Non-CDB database or root container of CDB

```
RMAN > transport tablespace
SOURCEDB:MYTBS,SOURCEDB:MYTBS_IDX auxiliary destination
'/wysheid/auxiliary' tablespace destination '/wysheid/backup‘ ;
```

- RMAN create and starts an automatic auxiliary instance

- Using the source database backup , restore and recover the database (control file and datafiles) to auxiliary destination in the format '<auxiliary_destination> /<DB_NAME> '

- On the auxiliary database create a directory object pointing to "tablespace destination"
- On the auxiliary database , converted the tablespaces to be transported into read only mode
- Perform transportable export of those tablespaces using the directory object created.
- Automatically Drop the directory object created
- Automatically delete the auxiliary database created.

Restore on Target

- Method 1

Copy the files to target server
Connect to target database
Execute the impscript.sql

- Method 2

- Create the directory object on the target database
- Copy the data file and dump file from source to the physical path on the target
- Restore using impdp

```
impdp system/oracle@targetdb directory=DP_DIR dumpfile= 'dmpfile.dmp'
transport_datafiles= /wysheid/backup/mytab.dbf,
/wysheid/backup/mytab_idx.dbf
```

• Make sure that target tablespaces are in read write mode.