„Lift & Shift“ oder wie kommt meine (Pluggable) Datenbank in die Oracle-Cloud

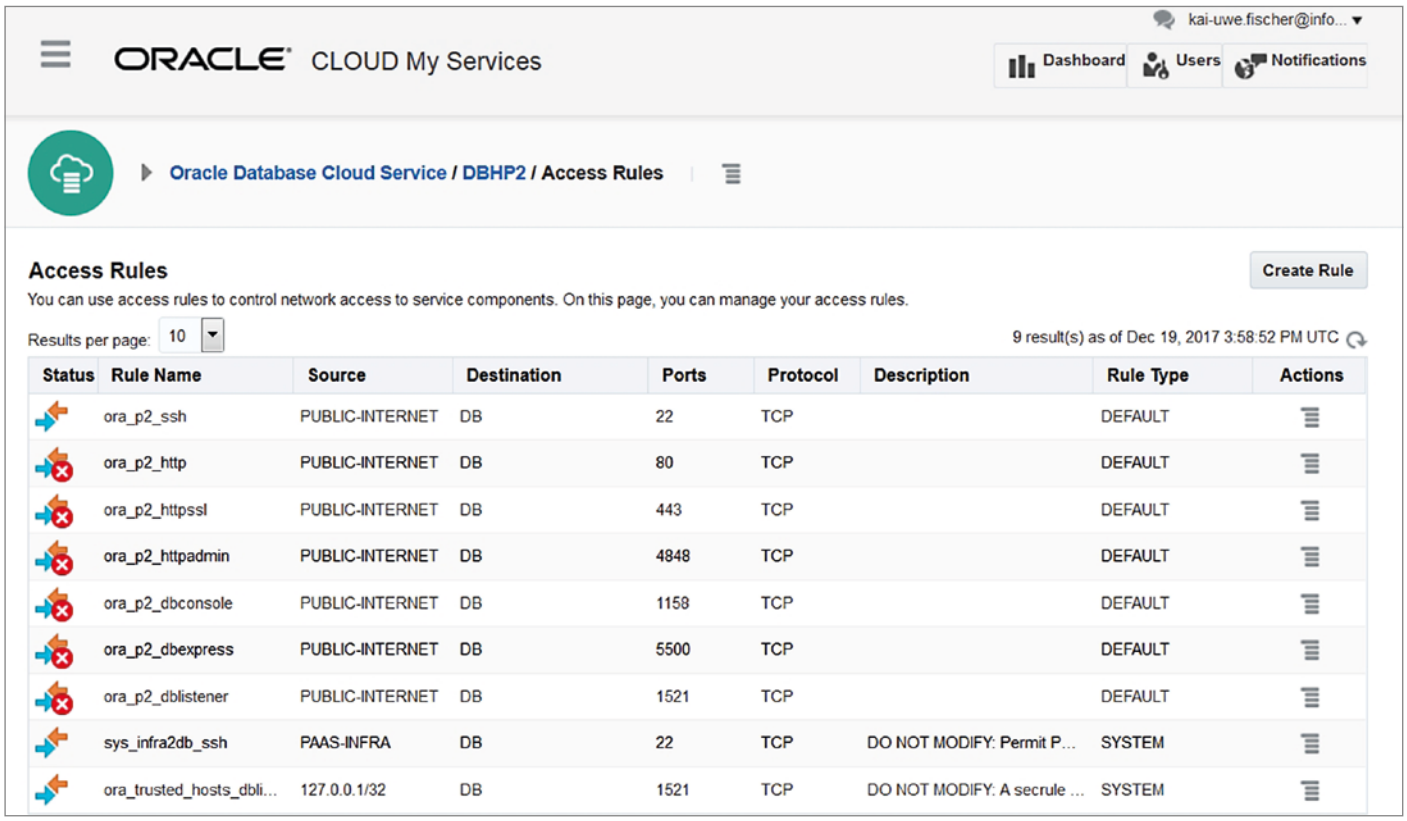
Kai Uwe Fischer, Logicalis GmbH

„Push a Button to move your Database to the Oracle-Cloud” ist einer der Slogans, mit denen Oracle für die Public Cloud wirbt. Ist es wirklich nur ein Knopfdruck? Dieser Artikel – basierend auf eigenen Test- und PoC-Erfahrungen – soll dies klären.

Bevor wir unsere Reise in die Oracle-Datenbank-Cloud beginnen, weist der Autor auf drei wichtige Dinge hin, die es auf dem Weg zu beachten gibt. Nach Erstellung einer Database-as-a-Service-Instanz über das Oracle-Database-Cloud-Service-Dashboard sind per Default alle Ports in der Firewall, außer dem SSH-Port 22, geschlossen *(siehe Abbildung 1)*. Damit die Reise nicht beendet ist, bevor sie überhaupt richtig begonnen hat, muss der SSH-Port in der Firewall des Unternehmens geöffnet sein. Ist dies der Fall, kann beispielsweise mit Putty oder dem SQL Developer mithilfe des erstellten Private-/Public-SSH-Schlüsselpaares auf den Datenbank-Server zugegriffen werden.

Folgende Software-Releases stehen als Oracle Database Cloud Service (DBCS) zur Verfügung:

- 11g Release 2
- 12c Release 1
- 12c Release 2
- 18c Release 1



Status	Rule Name	Source	Destination	Ports	Protocol	Description	Rule Type	Actions
	ora_p2_ssh	PUBLIC-INTERNET	DB	22	TCP		DEFAULT	
	ora_p2_http	PUBLIC-INTERNET	DB	80	TCP		DEFAULT	
	ora_p2_httpssl	PUBLIC-INTERNET	DB	443	TCP		DEFAULT	
	ora_p2_httpadmin	PUBLIC-INTERNET	DB	4848	TCP		DEFAULT	
	ora_p2_dbconsole	PUBLIC-INTERNET	DB	1158	TCP		DEFAULT	
	ora_p2_dbexpress	PUBLIC-INTERNET	DB	5500	TCP		DEFAULT	
	ora_p2_dblistener	PUBLIC-INTERNET	DB	1521	TCP		DEFAULT	
	sys_infra2db_ssh	PAAS-INFRA	DB	22	TCP	DO NOT MODIFY: Permit P...	SYSTEM	
	ora_trusted_hosts_dbli...	127.0.0.1/32	DB	1521	TCP	DO NOT MODIFY: A secrule ...	SYSTEM	

Abbildung 1: Die Access-Rules

Zu beachten ist, dass das Software-Release 12c, im Gegensatz zur On-Premise-Welt, nur in der Single-/Multitenant-Architektur verfügbar ist. Ein weiterer Unterschied zwischen der Oracle-On-Premise-Welt und dem Database Cloud Service (DBCS) in der Oracle-Public-Cloud ist, dass alle User-Daten per Default in der Oracle-Public-Cloud per Transparent Data Encryption (TDE) verschlüsselt sind, dies gilt auch für die Standard Edition. Der Anwender muss sich um die TDE-Konfiguration im DBCS-Umfeld nicht kümmern, dies geschieht automatisch beim Anlegen des Datenbank-Cloud-Service. *Listing 1* zeigt den entsprechenden Eintrag in der Datei „sqlnet.ora".

Jede Pluggable Datenbank (PDB) benötigt dabei einen eigenen Master-Key. Mehr dazu später. Aus den genannten Punkten ist zu erkennen, dass neue Aufgaben auf einen Datenbank-Administrator zukommen werden, denn mit den Themen „Verschlüsselung" (TDE) und „Tenant-Architektur" wird er sich in Zukunft beschäftigen müssen.

Wege in die Oracle-Database-Cloud

Es gibt eine Vielzahl von Möglichkeiten, wie eine On-Premise-Oracle-Datenbank in die Cloud migriert werden kann. Die eingesetzten Tools und Technologien hängen dabei von dem eingesetzten Release und der Edition ab. Im Using-Oracle-Database-Cloud-Service-Handbuch *(siehe „https://docs.oracle.com/en/cloud/paas/database-dbaas-cloud/csdbi/mig-migrating-premises-oracle-db-cloud.html")* sind im Kapitel 9 die einzelnen Migrationsmethoden beschrieben:

- Data Pump Conventional Export/Import
- Data Pump Full Transportable
- Data Pump Transportable Tablespace
- Remote Cloning a PDB
- Remote Cloning Non-CDB
- RMAN Cross-Platform Transportable PDB
- RMAN Cross-Platform Transportable Tablespace Backup Sets
- RMAN Transportable Tablespace with Data Pump
- RMAN CONVERT Transportable Tablespace with Data Pump
- SQL Developer and INSERT Statements to Migrate Selected Objects
- SQL Developer and SQL*Loader to Migrate Selected Objects
- Unplugging/Plugging a PDB
- Unplugging/Plugging Non-CDB

Dieser Artikel geht nicht auf die klassischen Migrationsmethoden mittels Data Pump und RMAN ein, sondern stellt folgende neue Migrationswege und deren Besonderheiten vor:

- Unplug/Plug-PDB (Kommandozeile)
- PDB-Clone (Oracle Enterprise Manager, SQL Developer)
- PDB-Hot-Clone (Kommandozeile)
- NONCDB to PDB (Kommandozeile)
- PDB-Klon von Oracle 12.1 nach Oracle 12.2

PDB-Unplug/Plug mittels Kommandozeile

In der Datenbank-Version 12 Release 1 konnte eine Pluggable Datenbank (PDB)

```
ENCRYPTION_WALLET_LOCATION = (SOURCE=(METHOD=FILE)(METHOD_
DATA=(DIRECTORY=/u01/app/oracle/admin/DBHP2/tde_wallet)))
```

Listing 1

```
SQL> alter pluggable database KUF3 close;
SQL> alter pluggable database KUF3 unplug into '/u01/stage/kuf3.pdb';
```

Listing 2

```
SET SERVEROUTPUT ON
DECLARE
  l_result BOOLEAN;
BEGIN
  l_result := DBMS_PDB.check_plug_compatibility(
                pdb_descr_file  => '/u02/stage/kuf3.pdb',
                pdb_name        => 'kuf3');
  IF l_result THEN
      DBMS_OUTPUT.PUT_LINE('compatible');
  ELSE
      DBMS_OUTPUT.PUT_LINE('incompatible');
  END IF;
END;
/
SQL> SET SERVEROUTPUT ON
DECLARE
  l_result BOOLEAN;
BEGIN
  l_result := DBMS_PDB.check_plug_compatibility(
                pdb_descr_file  => '/u02/stage/kuf3.pdb',
                pdb_name        => 'kuf3');
  IF l_result THEN
      DBMS_OUTPUT.PUT_LINE('compatible');
  ELSE
      DBMS_OUTPUT.PUT_LINE('incompatible');
  END IF;
END;
/
SQL>  2   3   4   5   6   7   8   9   10  11  12  13  compatible

PL/SQL procedure successfully completed.
```

Listing 3

```
SQL> create pluggable database KUF3 using '/u02/stage/kuf3.pdb';
SQL> alter pluggable database KUF3 open;
SQL> show pdbs;
CON_ID       CON_NAME       OPEN MODE          RESTRICTED
----------   -----------    -----------        ----------
       2     PDB$SEED       READ ONLY          NO
       3     PDB1           READ WRITE         NO
       4     MIG            READ WRITE         NO
       5     MIG2           READ WRITE         NO
       6     KUF3           READ WRITE         NO
```

Listing 4

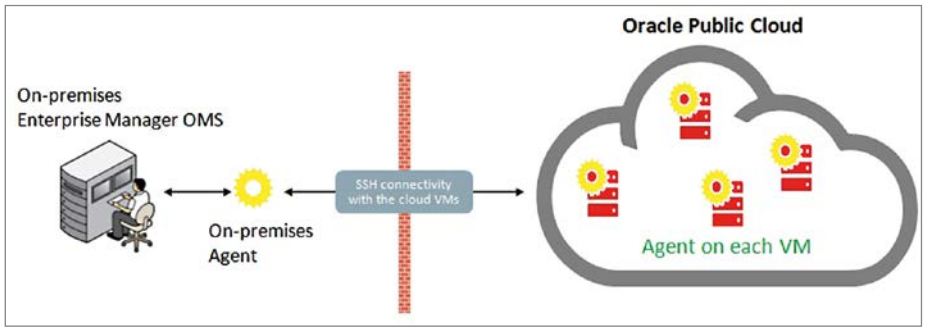


Abbildung 2: Der Hybrid-Cloud-Agent

in eine „*.xml“-Datei entladen werden. In dieser war nur die Beschreibung der PDB enthalten. Um diese PDB nun auf eine andere Datenbank umzuziehen, war es notwendig, daneben alle relevanten Datenbank-Dateien händisch zu kopieren. Dies ist in der Version 12.2 auch weiterhin möglich, allerdings gibt es nun auch die Möglichkeit, anstatt einer „*.xml“- eine komprimierte „*.pdb“-Datei zu erstellen, mit dem Vorteil, dass in diesem Dateiformat die komplette PDB enthalten ist.

Welche Schritte sind nun notwendig, um eine PDB aus einer On-Premise in eine DBCS-Umgebung zu verschieben? Im ersten Schritt muss die PDB auf der Quell-Datenbank gestoppt werden. Im Anschluss wird die gestoppte PDB ausgehängt *(siehe Listing 2)*.

Anschließend kann die Archive-Datei, in diesem Fall „kuf3.pdb“, auf den Zielhost in der Oracle-Public-Cloud kopiert werden. Danach lässt sich mit einer Prozedur überprüfen, ob der Inhalt der Archive-Datei, in diesem Fall die PDB „KUF3“, mit dem neuen Container kompatibel ist *(siehe Listing 3)*.

Sollte die PDB nicht kompatibel sein, lässt sich der Grund für die Inkompatibilität mit der View „PDB_PLUG_IN_VIOLATIONS“ abfragen. Da in diesem Fall kein Problem vorliegt, kann die PDB importiert werden *(siehe Listing 4)*.

PDB-Clone mittels Oracle Enterprise Manager 13.2

Bevor eine PDB mithilfe von Oracle Enterprise Manager (OEM) in die Cloud geklont werden kann, muss die DBCS-Instanz mit dem Hybrid-Cloud-Gateway zum OEM hinzugefügt werden. *Abbildung 2* zeigt die SSH-Verbindung zwischen den Agenten in der Oracle-Public-Cloud und dem Oracle Enterprise Manager On-Premise.

Nachdem die Hybrid-Cloud-Agent-Installation abgeschlossen ist, erfolgt die Auswahl der PDB, in diesem Fall „KUF1“, die geklont werden soll. Dazu im OEM im Menü „Oracle Database“ die Optionen „Cloning“ und „Clone to Oracle Cloud“ auswählen *(siehe Abbildung 3)*.

Im nächsten Fenster wird Folgendes definiert:

- Name der PDB auf dem Zielsystem
- PDB Administrator Credentials

- Name des Containers, in den die PDB hinzugefügt werden soll

Nun kann das Klonen beginnen, die einzelnen Schritte des Clone-Jobs können über das Fenster „Procedure Activity“ überwacht werden. Nach dem Klonen steht die PDB in der neuen CDB zur Verfügung. Leider erfolgte dieser Vorgang nicht fehlerfrei. Der Schritt „Post clone PDB creation operations“ bricht mit der Fehlermeldung „ORA-24964“ ab.

Die Recherche in My Oracle Support ergab, dass diese Fehlermeldung nicht bekannt ist. Der eröffnete Service Request führte zu einem Bug *(siehe Abbildung 4)*. Auch nach dem Einspielen der neusten Oracle-Enterprise-Manager-Updates und Plug-ins vom Oktober 2017 bleibt der Fehler bestehen.

PDB-Klon mittels SQL Developer

Um eine PDB in die Oracle-Public-Cloud mithilfe des SQL Developer zu klonen, muss dieser auf dem Quell-Server installiert sein. Ist dies nicht der Fall, erhält man wie der Autor beim Klonen in die Oracle-Cloud die folgende Fehlermeldung: „Quelle ist keine lokale Datenbank; führen Sie auf dem Quellrechner sqldeveloper aus.“ Ein Hinweis dazu ist im SQL-Developer-

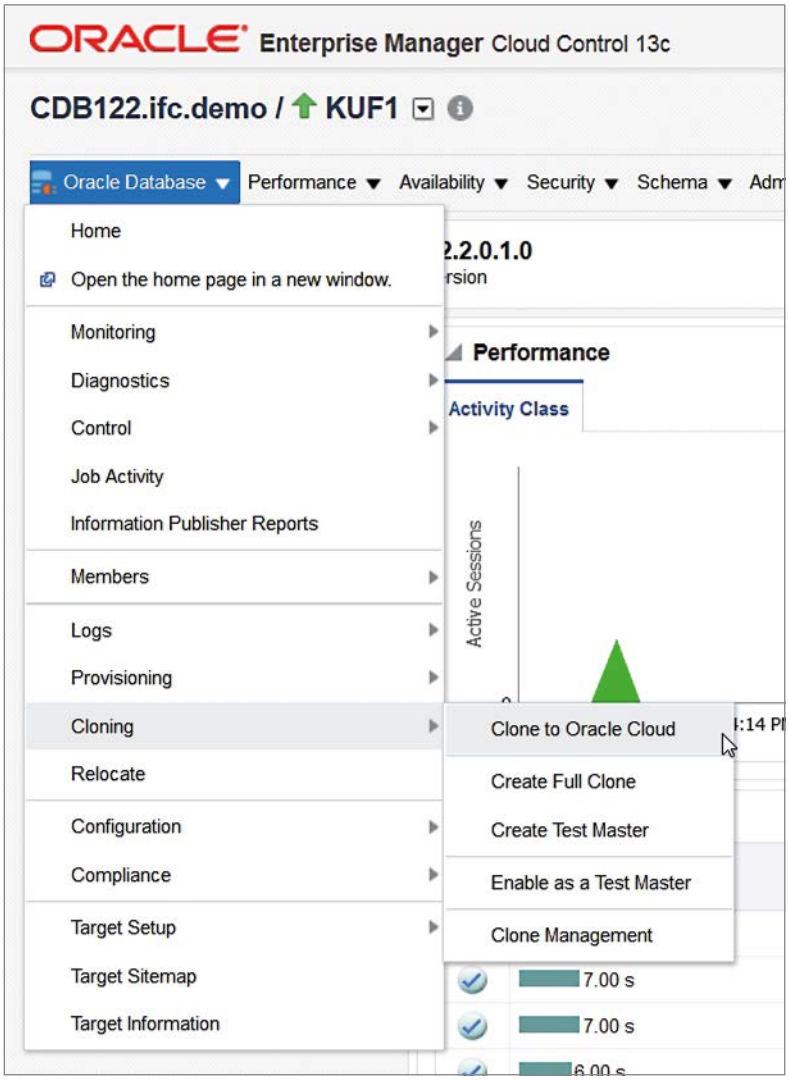


Abbildung 3: Auswahl „Clone to Oracle Cloud“

Bug 26117242 : Post clone PDB creation operation failes with ORA-24964

Bug Attributes

Type	B - Defect	**Fixed in Product Version**	
Severity	2 - Severe Loss of Service	**Product Version**	13.2.2.0.0
Status	11 - Code/Hardware Bug (Response/Resolution)	**Platform**	226 - Linux x86-64
Created	May 22, 2017	**Platform Version**	ORACLE LINUX 6
Updated	Oct 20, 2017	**Base Bug**	N/A
Database Version	12.1.0.2	**Affects Platforms**	Generic
Product Source	Oracle	Knowledge, Patches and Bugs related to this bug	

Related Products

Line	Enterprise Management	**Family**	Enterprise Manager Products
Area	Managing Databases using Enterprise Manager	**Product**	1366 - Enterprise Manager for Oracle Database

```
Hdr: 26117242 12.1.0.2 CLONEMGMT 13.2.2.0.0 DB_CLOUD PRODID-1366 PORTID-226
Abstract: Post clone PDB creation operation failes with ORA-24964
```

Abbildung 4: Fehler beim Klonen mit OEM

```
SQL> shutdown immediate;
SQL> startup mount;
SQL> alter database open upgrade;
SQL> alter database local undo on;
SQL> shutdown immediate;
SQL> startup
```

Listing 5

```
SQL> create user c##ifl identified by welcome1 container=all;
SQL> grant sysoper to c##ifl container=all;
SQL> grant create session to c##ifl container=all;
```

Listing 6

```
SQL> create public database link cdb122_link connect to c##ifl identified
by welcome1 using 'CDB122';
SQL> create pluggable database KUF2_CL2 from KUF2@cdb122_link;
SQL> alter pluggable database KUF2_CL2 open;
```

Listing 7

```
# export ORACLE_SID=NONCDB;
SQL> create user MIG identified by mig;
SQL> grant create session, create pluggable database to MIG;
SQL> shutdown immediate;
SQL> startup mount;
SQL> alter database open read only;
```

Listing 8

```
SQL> create tablespace KUF datafile size 5M;
ORA-28374: typed master key not found in wallet
```

Listing 9

Handbuch, Kapitel 6.11 „Clone PDB to Oracle Cloud“, leider nicht zu finden.

Um eine Verbindung zu der DBCS-Instanz zu erstellen, muss zunächst ein neuer SSH-Host im SQL Developer hinzugefügt und konfiguriert sein. Anschließend kann über den Menüpunkt „PDB in Oracle Cloud klonen …“ das Klonen gestartet werden. Anders als beim Oracle Enterprise Manager lässt sich hier der Name der zu klonenden PDB im Zielsystem nicht ändern. Mit der Meldung „Your PDB has been successfully cloned to the Oracle Cloud“ ist der Vorgang abgeschlossen.

PDB-Hot-Clone

Der Nachteil von PDB-Clone ist, dass während des Klonvorgangs die Quell-PDB den Anwendern nur lesend zur Verfügung steht („open read only“). Mit Einführung von PDB-Hot-Clone hat sich dies ab dem Release 12.2 geändert. Nun steht die zu klonende PDB während des Klonvorgangs den Benutzern uneingeschränkt zur Verfügung. Voraussetzung dabei ist allerdings, dass sich die Quell-Datenbank im Archivelog-Modus befindet und das neue Feature „LOCAL UNDO“ verwendet wird. Mit dem Kommando „select property_value from database_properties where property_name='LOCAL_UNDO_ ENABLED';“ lässt sich prüfen, ob „LOCAL UNDO“ verwendet wird. Falls dieses nicht konfiguriert ist, sind einige Schritte notwendig, um diesen Modus zu aktivieren *(siehe Listing 5)*.

Im nächsten Schritt wird auf der Quell-CDB noch ein Common User mit den folgenden Rechten angelegt *(siehe Listing 6)*. Auf der Zielseite kann direkt nach dem Anlegen eines Datenbank-Links auf die Quell-Datenbank mit dem Klonen begonnen werden *(siehe Listing 7)*. Im Moment ist PDB-Hot-Clone in die Oracle-Public-Cloud noch nicht über die Tools SQL Developer und Oracle Enterprise Manager unterstützt.

NONCDB TO PDB

Viele Kunden-Datenbanken, die in On-Premise-Umgebungen betrieben werden, verwenden allerdings noch die klassische Architektur. Ab der Version 12 können nun auch Nicht-Container-Datenbanken mithilfe des PDB-Klonmechanismus in die Oracle-Cloud geklont werden. Dazu muss auf der Quell-Datenbank ein Benutzer existieren, der das Recht „create pluggable database“ besitzt. Dies ist auch der Grund, aus dem dieser Mechanismus erst bei Datenbanken ab Version 12 funktioniert. Anschließend müssen die Quell-Datenbanken noch im „read only“-Modus geöffnet sein *(siehe Listing 8)*.

Auf der Ziel-Datenbank wird nun mit „create database link noncdb_link connect to mig identified by mig using ‚NONCDB‘;“ ein Datenbank-Link zur Quelldatenbank erzeugt. Das Kommando „CREATE PLUGGABLE DATABASE PDBX FROM NON$CDB@noncdb_link;“ wandelt eine NONCDB in eine PDB. Die Besonderheit an diesem Befehl ist, dass anstatt eines PDB-Namens, der ja bekanntlich nicht vorhanden ist, das Pseudonym „NON$CDB“ zum Einsatz kommt. Nach dem Ende des Vorgangs muss auf der erzeugten PDB, in diesem Falle „PDBX“, noch das Skript „@$ORACLE_HOME/rdbms/admin/noncdb_to_pdb.sql“ ausgeführt werden.

Vorsicht, Falle!

Egal, ob man nun die PDB mit dem „unplug/plug clone“- oder „hot clone“-Kommando erstellt hat, die PDBs wurden erfolgreich in die Oracle-Cloud geklont und stehen den Anwendern zur Verfügung. Was passiert aber, wenn man bei einer der geklonten PDBs einen neuen Tablespace hinzufügen möchte *(siehe Listing 9)*?

Der Tablespace kann nicht erstellt werden, da der Master-Key der PDB nicht im Wallet gefunden wird. Wie schon erwähnt, werden alle User-Daten in der Oracle-Cloud verschlüsselt, dazu wurde der Datenbank-Parameter „ENCRYPT_NEW_TABLESPACES“ eingeführt. In der Oracle-Public-Cloud steht dieser auf „CLOUD_ONLY“. Neue Tablespaces werden automatisch mit

AES128 verschlüsselt. Die Abfrage der View „V$ENCRYPTION_WALLET“ auf der geklonten PDB (Version 12.2) zeigt, dass ein „AUTOLOGIN“-Wallet verwendet wird und der Status auf „OPEN_NO_MASTER_KEY“ steht *(siehe Listing 10)*.

Das Problem ist also, dass für die geklonte PDB noch kein Master-Key im zentralen Wallet („ewallet.p12“) hinterlegt ist. Doch wie kann ein solcher Key neu erzeugt werden und gibt es dabei Unterschiede je nach verwendeter Datenbank-Version? Werfen wir zunächst ein Blick auf 12.1: Der Versuch, in der geklonten PDB einen neuen Master-Key mit dem SQL-Kommando „administer key management set key identified by „xxxxx“ with backup;“ zu erzeugen, schlägt mit der Meldung „ORA-46658: keystore not open in the container“ fehl. Diese Meldung macht den Anschein, dass der Keystore nicht geöffnet ist, obwohl die vorherige Abfrage „OPEN_NO_MASTER_KEY“ lieferte. In Wirklichkeit bedeutet diese Fehlermeldung, dass die Keys nicht modifiziert werden können, solange ein „AUTOLOGIN“-Wallet benutzt wird.

Wie lässt sich „AUTOLOGIN“ deaktivieren? Dazu wechselt man in das Verzeichnis „$ORACLE_BASE/admin/<SID>/tde_wallet“ und löscht beziehungsweise nennt die Datei „cwallet.sso“ um. Als Nächstes erfolgt ein Restart der Datenbank, um das Wallet zu schließen. Nun ist es möglich, mit dem SQL-Befehl „administer key management set keystore open identified by „xxxxx“ container=all;“ das Wallet manuell zu öffnen. Nach dem Wechseln in die geklonte PDB wird ein neues Wallet („ewallet.p12“) erzeugt. Das „AUTOLOGIN“-Wallet kann ab diesem Zeitpunkt nicht mehr verwendet und muss ebenfalls, auf Ebene der CDB, neu erzeugt werden *(siehe Listing 11)*. Nach einem weiteren Restart der Datenbank wird diese mit dem neuen „AUTOLOGIN“-Wallet geöffnet und auf der geklonten PDB kann nun ein neuer Tablespace erstellt werden.

Man sieht, dass der Aufwand, der pro geklonte PDB betrieben werden muss, doch erheblich ist. Wenn eine Migration von mehreren PDBs am Stück erfolgt, kann die Downtime minimiert werden, indem man die Wallet-Migration erst am Ende der PDB-Migration durchführt. Sollte sich diese allerdings über einen längeren Zeitraum erstrecken, wäre eine Downtime nach jeder PDB-Migration unter Umständen nicht mehr tragbar.

```
SQL> select status, wallet_type from v$encryption_wallet;
STATUS               WALLET_TYPE
OPEN_NO_MASTER_KEY   AUTOLOGIN
```

Listing 10

```
SQL> administer key management create auto_login keystore from
keystore '/u01/app/oracle/admin/ORCL/tde_wallet/' identified by "xxxxx";
SQL> administer key management set keystore close identified  by "xxxxx"
container=all;
```

Listing 11

```
SQL> administer key management set keystore open force    keystore
identified by "xxx";
SQL> administer key management set key force keystore    identified by
"xxx" with backup;
```

Listing 12

```
TABLESPACE_NAME              ENC
---------------------------- ---
SYSTEM                       NO
SYSAUX                       NO
UNDOTBS1                     NO
TEMP                         NO
USERS                        NO
```

Listing 13

```
TABLESPACE_NAME              ENC
---------------------------- ---
SYSTEM                       NO
SYSAUX                       NO
UNDOTBS1                     NO
TEMP                         NO
USERS                        YES
```

Listing 14

```
TABLESPACE_NAME                ENC
------------------------------ ---
SYSTEM                         NO
SYSAUX                         NO
UNDOTBS1                       NO
TEMP                           NO
USERS                          YES
```

Listing 15

Oracle hat diese Problematik erkannt und in der Version 12.2 die Klausel „FORCE KEYSTORE“ eingeführt, die nun ein Key-Management bei gleichzeitiger Benutzung des AUTOLOGIN-Wallet ermöglicht. Nach dem Anmelden an der geklonten PDB können mit den beiden folgenden Kommandos das Wallet aktualisiert und anschließend die gewünschten Tablespaces erzeugt werden *(siehe Listing 12)*.

Doch wie sieht es bei den bestehenden Tablespaces der geklonten PDBs aus? Sind dort die User Tablespaces verschlüsselt? Die SQL-Abfrage „select tablespace_name, encrypted from dba_tablespaces;“ liefert das folgende Ergebnis *(siehe Listing 13)*. *Listing 14* zeigt zum Vergleich die Abfrage auf einer bestehenden PDB.

Wie aus den beiden Abfrage-Ergebnissen zu erkennen ist, ist der Tablespace „USERS“ der geklonten PDB nicht verschlüsselt. Dies sollte dringend nachgeholt werden. Wurde das Wallet, wie oben beschrieben, um die neue PDB ergänzt, lässt sich der Tablespace mit „SQL> alter tablespace USERS encryption online encrypt;“ verschlüsseln. Da in DBCS Oracle-Managed-Files verwendet werden, kann bei der Verschlüsselung des Datafiles auf die „FILE_NAME_CONVERT“-Option verzichtet werden *(siehe Listing 15)*.

```
SQL> create user mig identified by welcome1;
SQL> grant create session, create pluggable database to mig;
SQL> alter pluggable database close;
SQL> alter pluggable database open read only;
```

Listing 16

```
SQL> create database link pdb121_link connect to mig  identified by mig
using 'PDB121';.
SQL> create pluggable database PDB2 from pdb11@pdb121_link;
```

Listing 17

```
create pluggable database PDB2 from pdb11@pdb121_link
ERROR at line 1:
ORA-17628: Oracle error 17630 returned by remote Oracle server
ORA-17630: Mismatch in the remote file protocol version client server
```

Listing 18

```
SQL> alter session set container=PDB2;
Session altered.
SQL> alter pluggable database open;
Warning: PDB altered with errors.
SQL> select name, message from PDB_PLUG_IN_VIOLATIONS
 where status<>'RESOLVED' and CON_ID=8 order by time;
NAME      MESSAGE
--------- --------------------------------------------------
PDB2      PDB's version does not match CDB's version: PDB's
          version 12.1.0.2.0. CDB's version 12.2.0.1.0.
PDB2      CDB is using local undo, but no undo tablespace
          found in the PDB.
```

Listing 19

```
SQL> alter pluggable database PDB2 close;
Pluggable database altered.
SQL> alter pluggable database PDB2 open upgrade;
Pluggable database altered.
SQL> exit
# cd $ORACLE_HOME/bin
# ./dbupgrade -c 'PDB2'
# sqlplus / as sysbda
SQL> alter pluggable database PDB2 open;
Pluggable database altered.
SQL> exit
# /u01/app/oracle/product/12.2.0/dbhome_1/perl/bin/perl catcon.pl -n 1
-e -b utlrp -d '''.''' utlrp.sql
```

Listing 20

```
SQL> alter session set container=PDB2;
Session altered.
SQL> create undo tablespace PDB2_UNDO;
create undo tablespace PDB2_UNDO
*
ERROR at line 1:
ORA-28374: typed master key not found in wallet
```

Listing 21

PDB-Klon von 12.1 nach 12.2

Ist ein Klonen einer NONCDB/PDB von der Version 12.1 nach 12.2 möglich? Die Oracle-Dokumentation gibt zu diesem Thema leider keine Auskunft. Der Autor hat es am Beispiel einer PDB mit der Version 12.1 getestet. Dazu hat er auf der Quell-PDB einen User-MIG mit den Rechten „create session" und „create pluggable database" angelegt und die PDB im „read only"-Modus geöffnet *(siehe Listing 16)*.

Auf der Ziel-CDB wird ein Datenbank-Link auf die Quell-DB erstellt und der Klonvorgang gestartet *(siehe Listing 17)*. Das Klon-Kommando schlägt mit folgender Fehlermeldung fehl *(siehe Listing 18)*. Es handelt sich hierbei um den Bug 18633374. Weitere Informationen findet man in der Oracle-Support-Note 18633374 „Error copying across remote servers with ASMCMD CP command (ASMCMD-8016, ORA-17628, ORA-17630, ORA-6512 Doc ID 18633374.8)".

Nach dem Einspielen des Patches, der übrigens auch hilft, wenn man kein ASM im Einsatz hat, konnte der Klon der PDB erstellt werden. Wie die folgende Abfrage der View „PDB_PLUG_IN_VIOLATIONS" zeigt, ist die geklonte PDB noch von der Version 12.1 und verwendet noch kein „LOCAL UNDO" *(siehe Listing 19)*.

Der nächste Schritt besteht darin, die PDB auf die aktuelle Version zu migrieren. Dazu muss die PDB im Modus „OPEN UPGRADE" geöffnet werden. Im Anschluss kann das Upgrade der PDB mittels „DBUPGRADE"-Kommando erfolgen *(siehe Listing 20)*. Nach erfolgreicher Migration der PDB auf die Version 12.2 wird im nächsten Schritt der lokale UNDO-Tablespace erzeugt *(siehe Listing 21)*.

Das Erstellen des UNDO-Tablespace bricht mit der mittlerweile bekannten Fehlermeldung ab, da der Master-Key für die erstellte PDB im Wallet fehlt. Leider lässt sich der neue Master-Key nicht erzeugen, da die PDB im Restricted Mode geöffnet ist *(siehe Listing 22)*.

Der Grund für den Restricted Mode ist der fehlende lokale „UNDO"-Tablespace, dieser kann aber, wie gesehen, wegen des fehlenden Master-Keys nicht erzeugt werden. Hier liegt ein klassischer Deadlock vor. Die Lösung dieses Dilemmas besteht aus den folgenden Schritten:

- „LOCAL UNDO"-Mode ausschalten
- Master-Key für die PDB erzeugen
- „UNDO"-Tablespace für die PDB erzeugen

- User-Tablespaces, in diesem Fall „USERS“ und „KUF“ der PDB, verschlüsseln
- „LOCAL UNDO“-Mode wieder anschalten

Noch ein Hinweis zum Schluss: Die Verschlüsselungsproblematik nach dem Klonen von PDBs ist kein reines Oracle-Datenbank-Cloud-Problem, sondern tritt bei der Verwendung der Transparent-Database-Encryption-Option bei CDB-Architektur On-Premise ebenfalls auf. Trotzdem ist es schade, dass Oracle im Handbuch „Using Oracle Database Cloud Service“ auf diese Problematik nicht eingeht. Das Kapitel „Remote Cloning einer PDB“ endet, nachdem die Pluggable Datenbank geöffnet wurde. Dass zu diesem Zeitpunkt weder ein Master-Key für die PDB existiert noch die User-Tablespaces unverschlüsselt sind, wird dabei nicht erwähnt.

```
SQL> administer key management set keystore open force keystore identi-
fied by "xxxx";
keystore altered.
SQL> administer key management set key force keystore identified by
"xxxx" with backup;
administer key management set key force keystore identified by "xxxx"
with backup
*
ERROR at line 1:
ORA-28442: Rekey of the TDE master encryption key is not permitted when
the database is in restricted mode.
```

Listing 22

Fazit

Wie man sieht, ist mehr als ein Knopfdruck notwendig, um eine Datenbank beziehungsweise eine Pluggable Datenbank in die Oracle-Cloud zu klonen. Es ist jedoch auch zu erkennen, dass sich Oracle auf dem richtigen Weg befindet und mit der Version 12.2 einige Schwachstellen der vorherigen Version beseitigt sind. Zum Abschluss noch einmal die Vorteile der CDB-Architektur des aktuellen Release im Überblick:

- Mittels „Hot Clone“ nahezu ohne Downtime eine PDB klonen
- Vereinfachtes Key-Management bei Verwendung von Transparent Data Encryption (TDE)
- Unterschiedliche Zeichensätze pro PDB
- Lockdown Profile pro PDB

Kai Uwe Fischer
kai-uwe.fischer@logicalis.de

DOAG Legal Council gestartet

Juristische Aspekte und vielfältige rechtliche Herausforderungen spielen auch in der Arbeit der DOAG und der Vertretung der Mitglieder-Interessen eine immer größere Rolle. Die fachlichen Kompetenzen der DOAG Competence Center werden zukünftig durch das am 20. April 2018 gegründete DOAG Legal Council ergänzt. Dieses wird die Arbeit der DOAG-Gremien und Competence Center mit einer Expertise im IT-Recht begleiten. Ob Vertragsgestaltung, Lizenzierung oder Datenschutz – das DOAG Legal Council wird rechtliche Themen für IT-Spezialisten zugänglicher machen und der DOAG bei der Vertretung der Interessen der Mitglieder zur Seite stehen. Denn ein Anlass zu dieser Initiative sind die seit Jahren diskutierten Anliegen zu rechtlichen Fragen der DOAG, für die Oracle bisher keine befriedigenden allgemeingültigen Lösungen lieferte.

Die Gründungsversammlung mit den ersten Mitgliedern des DOAG Legal Council fand parallel zur diesjährigen Delegiertenversammlung des Vereins statt. Die zukünftig noch um weitere Mitglieder ergänzte Runde bringt fundiertes Wissen zusammen und bildet die Plattform eines Austausches fachlicher und rechtlicher Betrachtungen. Schon beim ersten Treffen wurde die Erörterung der vieldiskutierten Themen „Lizenzierung von Oracle-Produkten in virtualisierten Umgebungen“ und „Datenschutz in der Oracle-Cloud“ als prioritäre Aufgaben des Council identifiziert und mit ersten Überlegungen begonnen.

Als Koordinator des DOAG Legal Council konnte der ehemalige DOAG-Vorstandsvorsitzende Dr. Dietmar Neugebauer Rechtsanwältin Dr. Jana Jentzsch aus der Sozietät Jentzsch IT, Hamburg, und Rechtsanwalt Dr. Thomas Thalhofer aus der Sozietät Noerr, München, begrüßen. Rechtsanwalt Dr. Jan Bohnstedt von Bartsch Rechtsanwälte, Frankfurt, ist weiteres Mitglied. Der langjährige Berater der DOAG, Rechtsanwalt Carsten J. Diercks, wird in dieser Runde die juristische Seite koordinieren.

Dr. Dietmar Neugebauer betonte bei der Gründung: „Wir freuen uns, mit dem DOAG Legal Council ab sofort bei rechtlichen Fragen auf die Fachkompetenz spezialisierter Rechtsanwälte zurückgreifen zu können. Wir sind sicher, dass wir gemeinsam mehr für unsere Mitglieder erreichen können, sei es über unsere Informationsplattform, oder vor allem im Gespräch mit Oracle.“

Die Mitglieder des Council sind von der DOAG ausgewählt worden und engagieren sich ehrenamtlich. Dabei wird keine Rechtsberatung im Einzelfall erfolgen, sondern allgemeine Betrachtungen von rechtlichen Fragen. Die Ergebnisse werden unter anderem in Form von Artikeln oder Vorträgen im Rahmen der DOAG in das DOAG Netzwerk eingebracht. Für darüberhinausgehende Beratung können die Mitglieder der DOAG direkten Kontakt zu den Anwälten aufnehmen. Weitere Informationen unter „*https://www.doag.org/index.php?id=1072*“.