In zwei Tagen zur neuen Exadata-Version – aus dem Protokoll eines DBA

Andrzej Rydzanicz, Opitz Consulting Deutschland

Der Artikel beschäftigt sich mit der Migration von der Oracle Exadata X8 in der Public Cloud auf die Version X9M, ebenfalls in der Cloud. Wir tauchen ein in ein Kundenprojekt, bei dem es darum ging, die fast 300 Terabyte große Datenbank eines Online-Händlers auf die neue Exadata-Version zu migrieren und diese in kürzester Zeit auf die neue Hardware umzuziehen. Der Artikel ist für alle interessant, die eine Exadata in der Public Cloud betreiben und sich mehr Flexibilität bei Speicher- und Rechenleistung wünschen – und natürlich für alle, die ein Exadata-System in kürzester Zeit auf die neueste Hardware umziehen wollen.

Die wichtigsten Arbeitsschritte, die wir beim Umzug der Exadata X8 zur X9M in der Cloud gegangen sind, Tools, die wir empfehlen, und Stolperfallen, die Sie vermeiden können – dies alles gehen wir in diesem Artikel mit Ihnen gemeinsam durch. Auch andere Dinge, die für die Migration wichtig sind, kommen dabei zur Sprache:

- Wie baue ich die Umgebung für die Exadata auf?
- Wie stelle ich den VM-Cluster bereit?
- Wie greife ich auf den bestehenden Object Storage zu?
- Wie skaliere ich die Exadata in den Bereichen Storage und CPU?

Bei der Konfiguration des Object Storage, wo Backups der alten Prod-Datenbank gespeichert werden, haben wir beispielsweise Dinge erlebt, die uns vorher nicht bewusst waren.

Knöpfchen drücken, und fertig?

In der letzten Zeit habe ich mit meinem Service-Team viele interessante Projekte durchgeführt. In einem von ihnen ging es um die Migration eines 300-Terabyte-Data-Warehouse von der Exadata X8 zur Exadata X9M – beide Systeme in der Public Cloud von Oracle. Wir hatten nicht viel Zeit. Zu gerne hätten wir nur ein Knöpfchen gedrückt, und fertig. Nicht nur wegen der Größe der Datenbank, sondern auch wegen der vielen Tests, die wir in Vorfeld durchführen mussten, um beim Go-live auf der sicheren Seite zu sein, brauchten wir am Ende 2 Tage und 30 Minuten.

Doch zurück zum Anfang: Das Ganze fing damit an, dass einer unserer Kunden, ein Online-Händler, mit Elan in eine beliebte Rabattwoche starten wollte. Alles war vorbereitet für die „Cyber Week" im November, als Zweifel aufkamen. Die IT befürchtete, dass die Datenbanksysteme den vermehrten Transaktionen in dieser Woche nicht gewachsen sein könnten. Genau genommen ging es um die zu geringe Plattenkapazität der Exadata. Unglücklicherweise ließ sich der Speicher der bestehenden Version nicht durch zusätzliche Storage Cells erweitern. Bei der Version X9M hingegen sind Erweiterungen überhaupt kein Problem. Hier lassen sich Storage Cells dynamisch hinzufügen. Gesagt, getan: Eine Migration sollte das Problem lösen!

Da nicht viel Zeit blieb, mussten wir einen Weg finden, um das 300-Terabyte-Data-Warehouse vor allem schnell, aber natürlich auch fehlerfrei zu migrieren. Für die Datenmigration nutzten wir Data Guard. Denn Data Guard bietet die Möglichkeit, die Funktionalität der Datenbank und die Integrität der migrierten Daten in einer Snapshot-Standby-DB zu testen. Wie sich zeigen sollte, nicht ganz ohne Hindernisse, aber dazu später mehr … Fangen wir vorne an:

Ein Umfeld für die neue Exadata

Der Aufbau der Umgebung für die Exadata X9M lief wie üblich in zwei Schritten ab:

1. Neue Hardware und Exadata-Infrastruktur verfügbar machen
2. VM-Cluster einrichten

Da das Unternehmen bereits eine Exadata X8-2 in der Public Cloud betrieb und fast alle Komponenten der Infrastruktur wie Compartement, Availability Domain oder Object- und Archive-Storage-Buckets vorhanden waren, gestaltete sich Schritt 1, also der Aufbau von Hardware und Infrastruktur, nicht

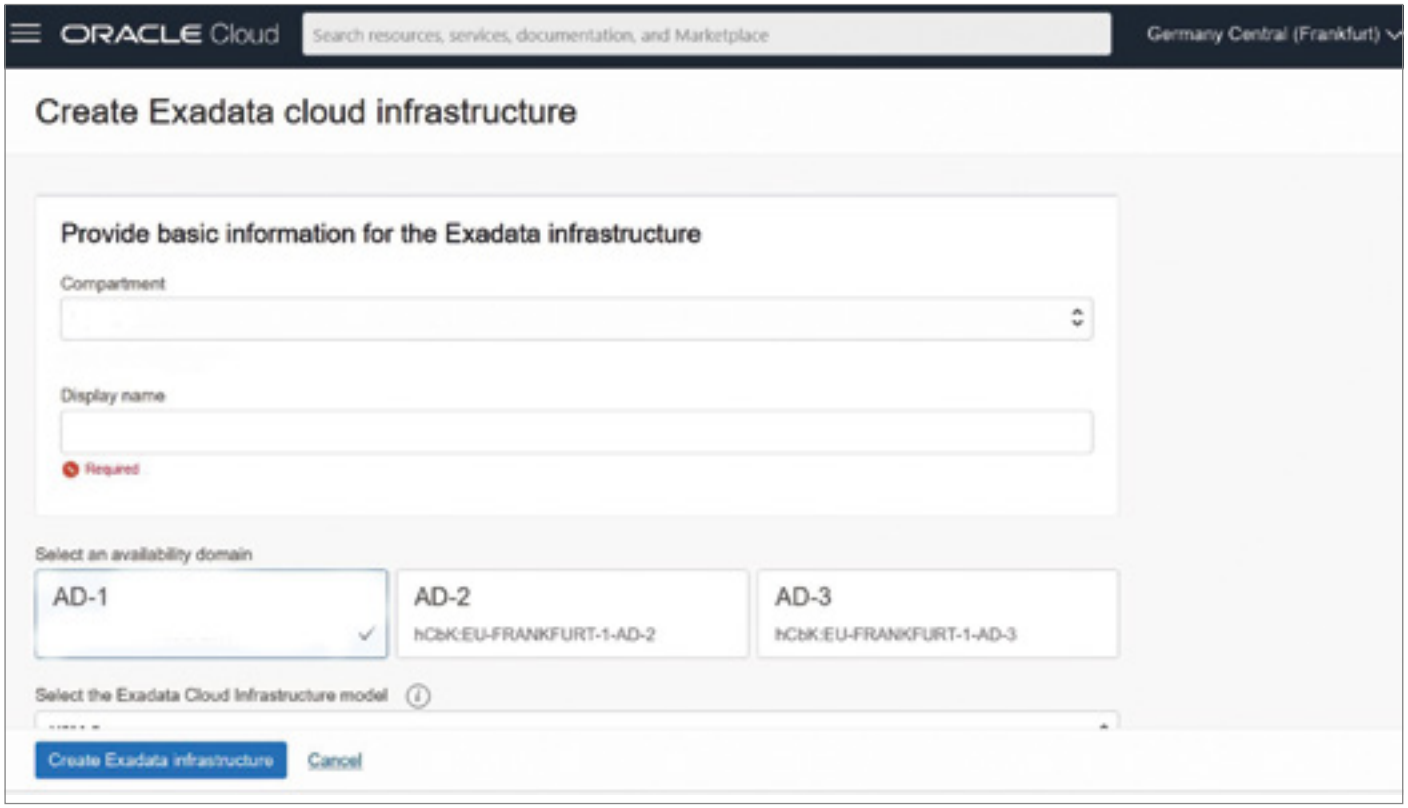

Abbildung 1: Basisinformationen für die Exadata-Infrastruktur (Quelle: Oracle)

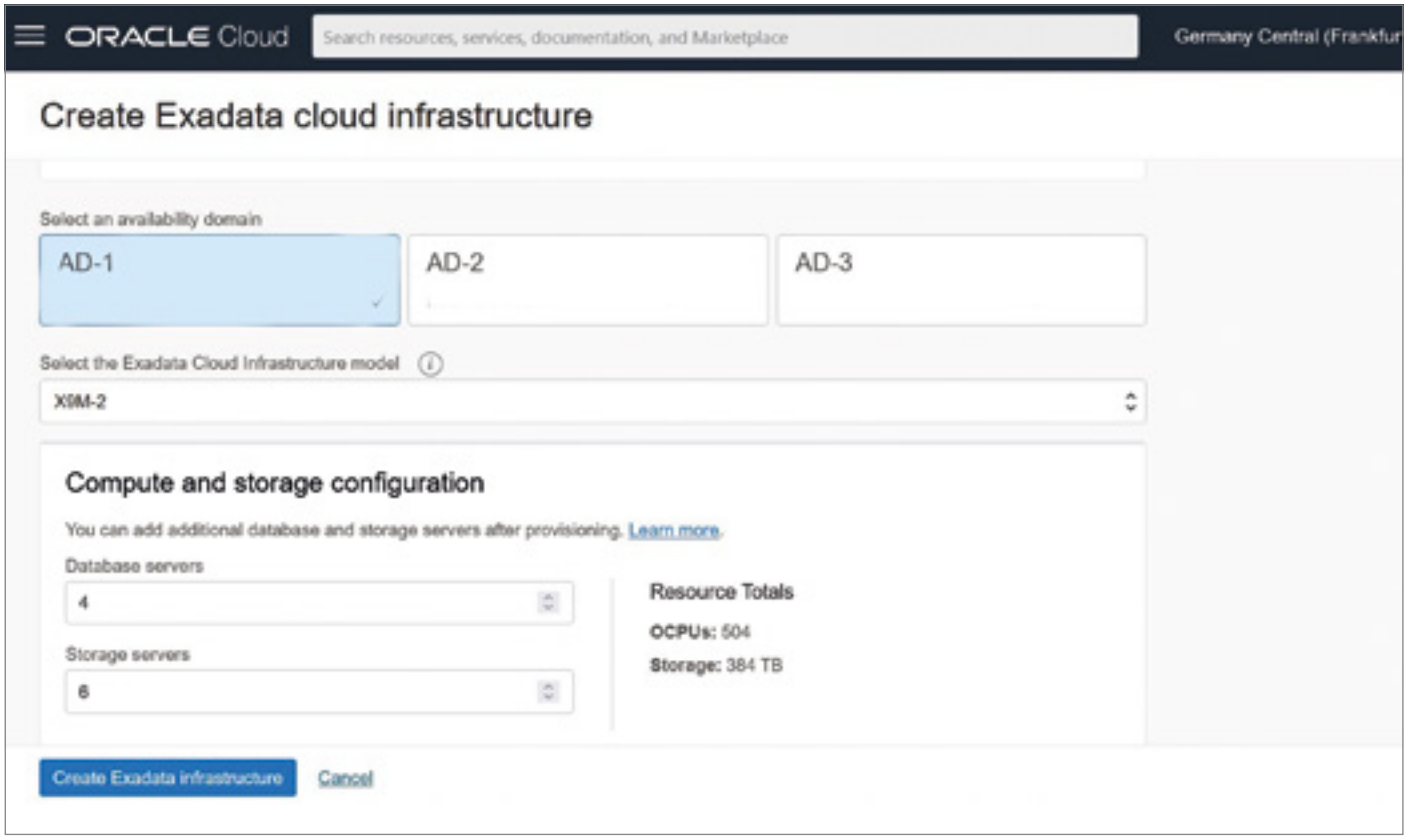

Abbildung 2: Konfiguration der Compute- und Speicher-Einheiten (Quelle: Oracle)

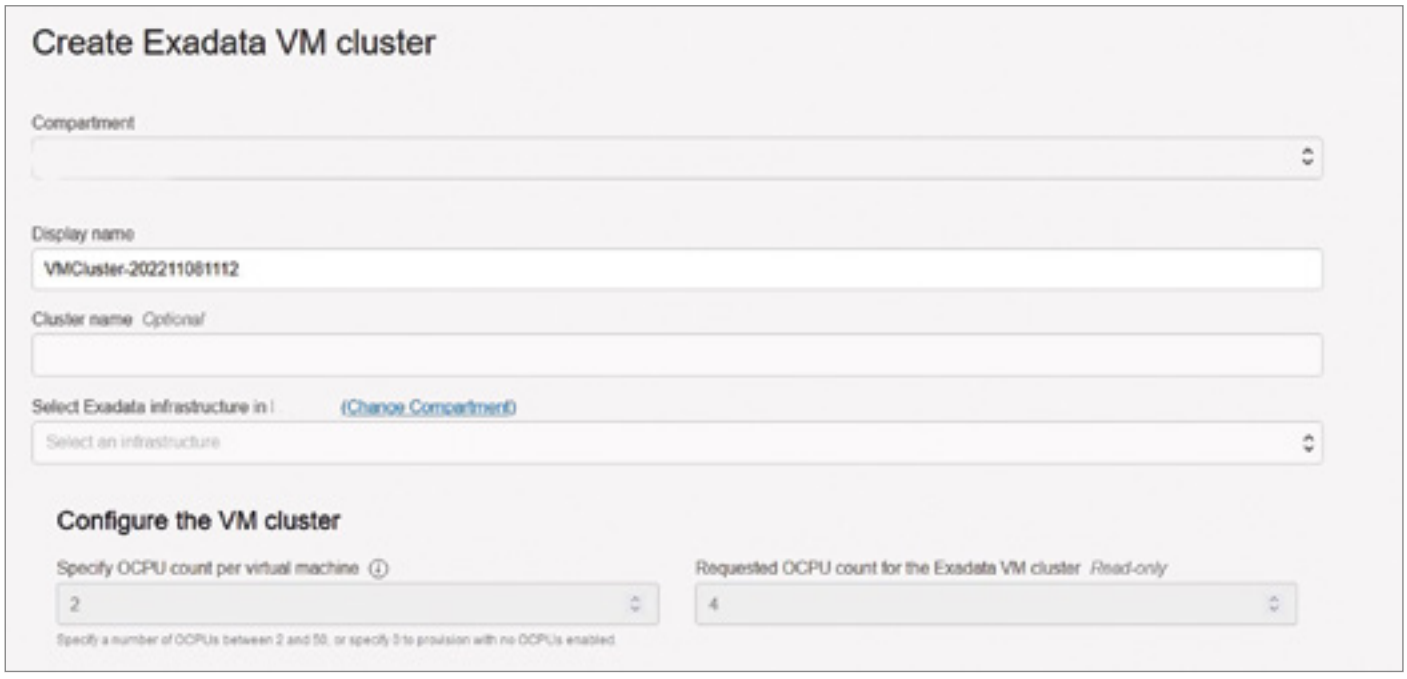

Abbildung 3: VM-Cluster konfigurieren (Quelle: Oracle)

so kompliziert. Neu hinzu kam ein Subnet im Virtual Cloud Network. Denn die IP-Adressen im bestehenden Subnet reichten für die neue Exadata-Infrastruktur nicht aus.

Aber auch Details waren zu beachten: Ein Cloud-Infrastrukturmodell musste aufgesetzt, die Rechen- und Speicher-Konfiguration musste eingerichtet und benannt werden, ebenso die Availability Domain und das in unserem Fall bereits vorhandene Netzwerk-Compartment.

Die Screenshots *(siehe Abbildungen 1 bis 7)* zeigen die wichtigsten Eingabemasken im Provisioning-Prozess:

Weiter ging es mit Schritt 2: der Erstellung eines VM-Clusters. Hier galt es festzulegen, welche Grid-Infrastruktur-Version installiert werden soll. Wer mit dem Deployment von alten Bare-Metal-Exadata-Systemen vertraut ist, findet sich hier schnell zurecht. Hier werden Dinge wie

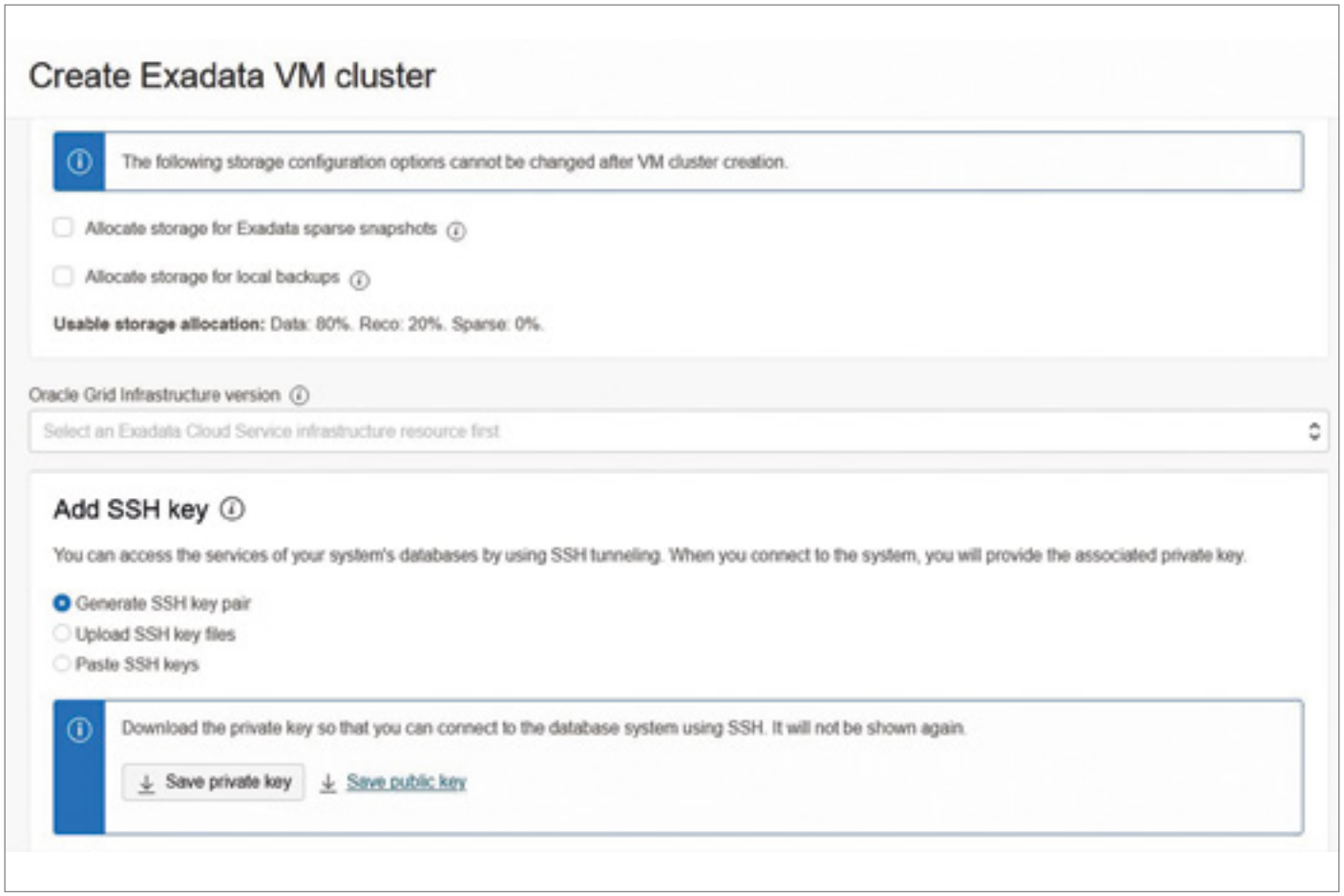

Abbildung 4: SSH-Key hinzufügen (Quelle: Oracle)

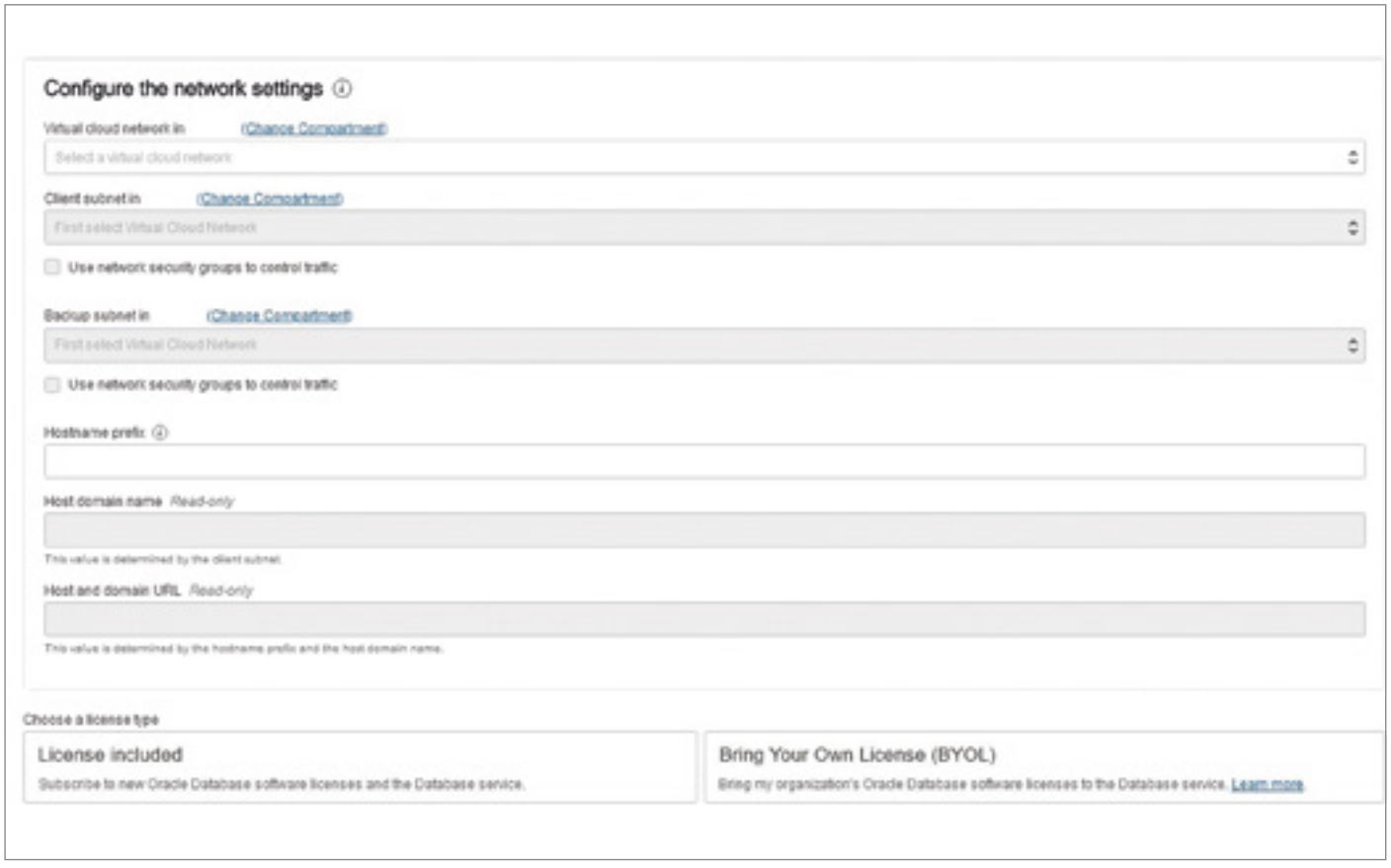

Abbildung 5: Netzwerkeinstellungen vornehmen (Quelle: Oracle)

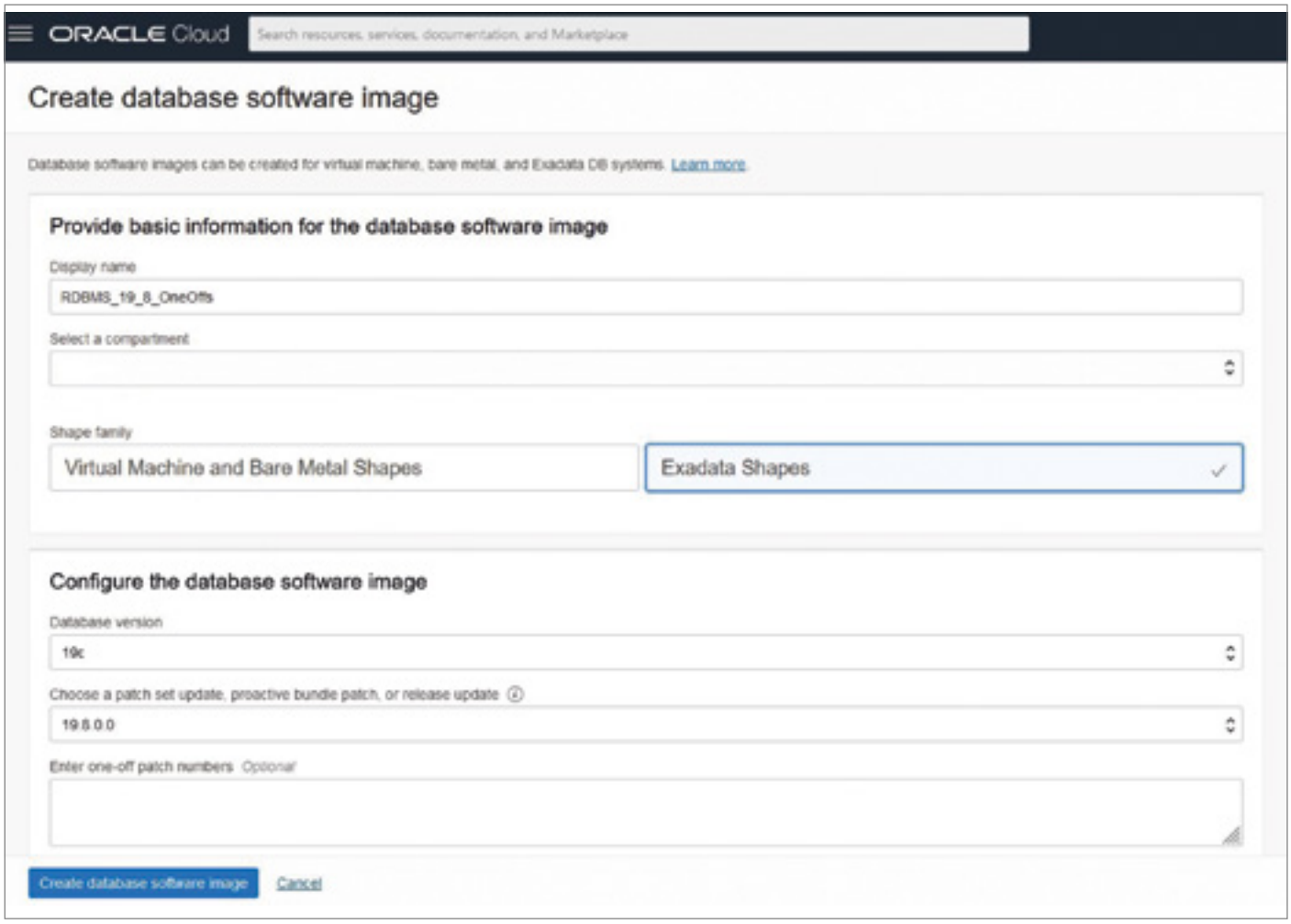

Abbildung 6: Database-Software-Image einrichten (Quelle: Oracle)

Host Prefix, Host Domain, Client Subnet oder Backup Subnet definiert.

In zwei Tagen hatten wir die Umgebung für die neue Exadata-Version installiert, ein gutes Zwischenergebnis!

RDBMS Deployment und Patches

Wenn die Exadata läuft, gibt es noch einiges zu tun. Es müssen Patches und weitere Dinge deployt und installiert werden, wie zum Beispiel das RDBMS Home. Da wir die produktive Datenbank auf RDMS 19.8 Home betrieben, mussten wir diese Version nun auf den neuen Cluster heben. Hier kam für uns eine sehr interessante Möglichkeit zum Vorschein: Wir erstellten via OCI Console ein „Gold-Image" mit allen OneOffs, RUs und CVEs, die wir brauchten. Aber Achtung: Das geht nur, wenn die Patches nicht passwortgeschützt sind!

Was die Grid-Infrastruktur-Version 19.16 betrifft: Alle Patches, die hierfür zur Verfügung stehen, hatten wir schon im Zuge der VM-Cluster-Erstellung installiert. Ebenso die Patches für das OS-Image. Dies war wichtig, um später eine Downtime nach dem Switchover zu verhindern. Vor dem Switchover war das System also bereits auf dem neuesten Stand.

Warum Oracle Data Guard?

Auf welche Art migriert man eine fast 300 Terabyte schwere Datenbank voller Kundendaten? Die Entscheidung für das Tooling ist uns nicht leichtgefallen. Wir haben uns diverse Möglichkeiten für die Datenbankreplikation angeschaut. Wie schon erwähnt auch GoldenGate. Letztendlich überzeugte uns aber Data Guard. Warum?

Diese Vorteile haben uns überzeugt:

- Data Guard ist seit vielen Jahren als Technologie ausgereift.
- Es bietet viele Möglichkeiten, um Probleme im Vorfeld zu analysieren.
- Es gibt diverse Tests, die vor der tatsächlichen Migration durchgeführt werden können. Vor allem die Snapshot-Standby-Datenbank bietet hierfür viele Möglichkeiten. Dorthin kann man innerhalb von Data Guard konvertieren.
- Die Endian-Plattform, die für die Migration zur Verfügung steht, hilft Zeit zu sparen.
- Data Guard ist weniger komplex als beispielsweise GoldenGate und damit leichter zu handhaben.

Die Standby-Datenbank war mit RMAN Restore/Recovery aufgebaut worden. Unser Ziel war eine RAC-Standby-Datenbank. Die Herausforderung lag darin, den schnellsten Weg zu finden, um die enorme Datenmenge der produktiven Datenbank zu übertragen, aber auch kein Risiko einzugehen. Eine Datenbank dieses Kalibers mittels RMAN Duplicate oder Restore aufzubauen, kann bedeuten, dass während eines Speichervorgangs neue

```
run {
  ALLOCATE CHANNEL ch01 DEVICE TYPE 'SBT_TAPE' CONNECT 'sys/@<DB_UNIQUE_NAME[1-4]> PARMS 'SBT_LIBRARY=/
var/opt/oracle/dbaas_acfs/<DB_NAME>/opc/libopc.so ENV=(OPC_PFILE=/var/opt/oracle/dbaas_acfs/<DB_NAME>/opc/
opc<DB_NAME>ora)';
.
.
.
     SET NEWNAME FOR DATABASE TO '+DATAC1';
     restore database;
   }
   switch database to copy;
```

Listing 1: Restore-Skript

```
run {
  ALLOCATE CHANNEL ch01 DEVICE TYPE 'SBT_TAPE' CONNECT 'sys/@ <DB_UNIQUE_NAME>' PARMS 'SBT_LIBRARY=/var/
opt/oracle/dbaas_acfs/<DB_NAME>/opc/libopc.so ENV=(OPC_PFILE=/var/opt/oracle/dbaas_acfs/<DB_NAME>/opc/op-
c<DB_NAME>.ora)';
.
.
recover database from service '<DB_UNIQUE_NAME_PROD>' section size 512G;
}
switch database to copy;
```

Listing 2: Recovery-Skript

```
dbaascli database backup --dbName <DB_NAME> --configure --configFile /home/oracle/backup_config/bkup-<DB_
NAME>.cfg
DBAAS CLI version 22.3.1.1.0
Executing command database backup --dbName <DB_NAME> --configure --configFile /home/oracle/backup_config/
bkup-<DB_NAME>.cfg
DBaaS Backup API V1.5 @2022 Multi-Oracle home
-> Action : set_config
-> logfile: /var/opt/oracle/log/<DB_NAME>/bkup_api_log/bkup_api_3e777dfe_20225123543.717530.log
cfgfile : /home/oracle/backup_config/bkup-<DB_NAME>.cfg
Using configuration file: /home/oracle/backup_config/bkup-<DB_NAME>.cfg
API::Parameters validated.
UUID 77b3e5ae449910200990708cd for this set_config(configure-backup)
** process started with PID: 158063
** see log file for monitor progress
-------------------------------------
dbaascli execution completed
```

Listing 3: Kommando

```
dbaascli database backup --status --uuid 77b3e5ae449910200990708cd --dbname <DB_NAME>

DBaaS Backup API V1.5 @2022 Multi-Oracle home
@ STARTING CHECK STATUS 77b3e5ae449910200990708cd
[ REQUEST TICKET ]
[UUID    ->  77b3e5ae449910200990708cd
[DBNAME  ->  <DB_NAME>
[STATE   ->  success
[ACTION  ->  configure-backup
[STARTED ->  2022-10-05 10:35:45 UTC
[ENDED   ->  2022-10-05 10:38:31 UTC
[PID     ->  158563
[TAG     ->
[PCT     ->
[ END TICKET ]
dbaascli execution completed
```

Listing 4: Abruf des aktuellen Stands der Konfiguration

Abbildung 7: Anzahl der OCPUs pro Cluster erhöhen (Quelle: Oracle)

```
ALTER DATABASE FLASHBACK ON;
```

Listing 5: Anpassung einer Einstellung auf der Standby-Seite

```
ALTER SYSTEM SET db_flashback_retention_target=<Anzahl-Minuten>;
```

Listing 6: Anpassung einer zweiten Einstellung auf der Standby-Seite

Daten auf der Primary-Datenbank hinzugefügt werden. Was leicht zu Problemen bei Recovery-Vorgängen führen kann.

Um dem vorzubeugen, stellten wir den Restore auf 50 Channels per Instanz ein. Damit kamen wir auf insgesamt 200 Channels. Mit dieser Konfiguration dauerte der Restore-Vorgang ungefähr 14 Stunden. Wobei ich erwähnen muss, dass wir, um das Ganze performant zu halten, die Anzahl der CPUs auf dem VM-Cluster noch auf 18 OCPUs pro virtuelle Machine erhöht haben, auf 72 OCPUs pro Cluster *(siehe Abbildung 7)*.

Der Vorteil liegt auf der Hand: Beim Restore-Vorgang verteilt man die Arbeit auf mehrere Instanzen, um den Prozess zu beschleunigen, beim Recovery-Vorgang profitiert man allerdings nur von Channels, die innerhalb eines Datenbankknotens allokiert wurden *(siehe Listing 1 und 2)*. Hier kann jeweils nur eine Apply-Instanz aktiv sein. Der Recovery-Vorgang kostete uns etwa drei Stunden. Kein Wunder bei Redo-Logs in einer Größe von 30 Gigabyte!

Der Restore- und Recovery-Prozess hätte ohne die Konfiguration des Object-Store-Zugriffs so nicht funktioniert. Die Konfiguration des OCI-Backup-Moduls erwies sich als problematisch. Ein Workaround musste helfen:

Der Workaround bestand in der Erstellung einer Template-Datenbank (OCI „create database"). Damit konnten wir sichergehen, dass alle Verzeichnisstrukturen zum Beispiel für Wallets und Logfiles für die zukünftige Standby-Datenbank vorliegen. Außerdem wurde die Datenbank automatisch im CRS registriert. Das ersparte uns manuelle Schritte wie das Hinzufügen einer Instanz oder Ähnliches. Der Zugriff auf den Object Store wurde bereits auf der Primary-Seite konfiguriert, sonst wäre es nicht möglich gewesen, die Backups auf der Primary-Datenbank zu speichern.

Die gleiche Konfiguration führten wir auf der Standby-Datenbank durch. Dafür generierten wir eine Konfig-CFG-Datei auf der Primary-Datenbank und übertrugen sie folgendermaßen auf die Standby-Seite:

```
dbaascli database backup --getConfig --dbname <DB_NAME>
```

Es ist nicht nötig, jeden einzelnen Punkt aus dieser Datei zu besprechen, die Kommentierung ist ausreichend und selbsterklärend. Was ich aber erwähnen möchte, ist der Fakt, dass man das OSS-Passwort (oss_password) in der Datei im Plain-Text-Format angeben muss, bevor man das in *Listing 3* abgebildete Kommando ausführt.

Der aktuelle Stand der Konfiguration lässt sich mit der in *Listing 4* abgebildeten Abfrage aufrufen.

Natürlich müssen wie üblich Standby-Redo-Logs auf der Primary- und Standby-Datenbank erstellt werden. Mittels „Restore/Recover from Service" werden die Standby-Redo-Logs automatisch auf der Standby-Datenbank angelegt.

„Snapshots" für mehr Sicherheit

Mit Data Guard waren wir in der Lage, die physikalische Standby-Datenbank in eine Snaphot-Standby-Datenbank zu konvertieren.

Was ist eigentlich eine Snapshot-Standby-Datenbank?

Eine Snapshot-Standby-Datenbank ist eine geöffnete Standby-Datenbank, mit der alle Aktionen durchgeführt werden können, wie mit jeder anderen Datenbank auch. Es sind also sowohl Lese- als auch Schreiboperationen erlaubt. Das Besondere: Während die Snapshot-Standby-Datenbank geöffnet ist, unterbricht der Recovery-Vorgang. Die Snapshot-Standby-Datenbank wird also zunächst auf den Zeitpunkt zurückgesetzt, an dem sie geöffnet wurde.

Und so funktioniert es:

- Beim Öffnen der Datenbank wird ein Restore Point erstellt, auf den die Datenbank wieder zurückgesetzt werden soll.
- Alle Recovery-Aktionen, die durch das Öffnen der Snapshot-Standby-Datenbank ausgesetzt waren, werden nachgeholt.

```
DGMGRL> connect sys/password
Connected to "<standby_db_unique_name>"
Connected as SYSDBA.
DGMGRL> convert database "<standby_db_unique_name>" to snapshot standby ;
Converting database "<standby_db_unique_name>" to a Snapshot Standby database, please wait...
Database "<standby_db_unique_name>"  converted successfully
```

Listing 7: Konvertieren zur Snapshot-Datenbank

```
DGMGRL
DGMGRL> connect sys/password
Connected to "<standby_db_unique_name>"
Connected as SYSDBA.
DGMGRL> convert database <standby_db_unique_name> to physical standby;
Converting database "<standby_db_unique_name>" to a Physical Standby database, please wait...
Operation requires a connection to database "<primary_db_unique_name>"
Connecting ...
Connected to "<primary_db_unique_name>"
Connected as SYSDBA.
Oracle Clusterware is restarting database "<standby_db_unique_name> " ...
Connected to "<standby_db_unique_name>"
Connected to "<standby_db_unique_name>"
Continuing to convert database"<standby_db_unique_name>" ...
Database "<standby_db_unique_name>" converted successfully
```

Listing 8: Restore Flashback

```
Select job_name, database_role, enabled from dba_scheduler_job_roles;

JOB_NAME              DATABASE_ROLE    ENABL
--------------------- ---------------- -----
JOB_NAME_XYZ            PRIMARY        FALSE
```

Listing 9: Ansicht der Job-Konfiguration

```
SQL> select database_role from v$database;

DATABASE_ROLE
----------------
SNAPSHOT STANDBY
```

Listing 10: Abfrage der Snapshot-Datenbank

```
Dbms_scheduler.set_attribute (name => '<job_name> ', attribute => 'database_role', value => 'snapshot
standby or ALL ');
End;
```

Listing 11: Änderung der Job-Definition auf der Primary-Datenbank

```
select 'exec Dbms_scheduler.set_attribute (name => ''' || owner || '.' || job_name || '''' || ',' || 'at-
tribute => ' || '''' || 'database_role' || '''' || ', value => '|| ''''|| 'SNAPSHOT STANDBY'||''|| ''''
||');'|| ''
from dba_scheduler_jobs;
```

Listing 12: SQL-Skript

- Die Länge des Zeitraums, in dem die Snapshot-Standby-Datenbank geöffnet war, bestimmt die Dauer der Synchronisation. Die Produktivdatenbank ist in keiner Weise betroffen, es gibt also keinen Overhead.

Für die Snapshot-Standby-Datenbank müssen zwei Einstellungen auf der Standby-Seite angepasst werden *(siehe Listing 5 und 6)*.

Das Konvertieren zur Snapshot-Datenbank ging in rund 30 Minuten über die Bühne *(siehe Listing 7)*.

Wenn man im Hinterkopf behält, dass wir die Standby-Datenbank im Snapshot-Modus zwei Tage lang intensiv getestet und viele Änderungen bei den Daten durchgeführt haben, dann sind 30 Minuten für das Zurückwandern zur physikalischen Standby-Datenbank relativ wenig. In diesen 30 Minuten fand auch der Restore Flashback statt, der die Datenbank auf die Zeit des Restore Point zurücksetzte *(siehe Listing 8)*.

Als 8.000 Scheduling-Jobs plötzlich weg waren …

Da unser Kunde über 8.000 Scheduling-Jobs für interne Zwecke benutzt, war es wichtig, diese Funktionalität auf der Standby-Seite zu testen. Hier erwarteten wir keine Probleme. Schließlich machen wir die Datenbank dafür im Read/Write-Modus auf. Hier sollte alles 1:1 identisch sein mit der Primary-Datenbank, oder?

Leider nicht so ganz: Als wir die Jobs starten wollten, waren die überhaupt nicht da. Ein select count(*) form v$scheduler_jobs zeigte nur interne Jobs, aber nicht die, die für unseren Kunden wirklich wichtig waren. Als wir uns die Job-Konfiguration anschauten, bekamen wir dies zu sehen *(siehe Listing 9)*.

Die Abfrage der Snapshot-Datenbank ergab das in *Listing 10* abgebildete Ergebnis.

Kein Wunder also, dass die Jobs auf der Standby-Datenbank nicht mehr sichtbar waren. Dafür sahen wir zwei Lösungswege:

a) Wir ändern die Job-Definition auf der Primary-Datenbank *(siehe Listing 11)*.
b) Wir passen die Job-Definitionen auf der Snapshot-Datenbank an.

Option 2 war natürlich die bessere Variante. Naja, jedenfalls, wenn es um 20 Definitionen gegangen wäre. Doch bei mehr als 8.000? Glücklicherweise kam uns ein simples SQL-Skript zu Hilfe. Mit ihm konnten wir 8.000 Definitionen so behandeln, als wären es 20 *(siehe Listing 12)*.

Ende gut, alles gut: Die Jobs waren wieder sichtbar und ausführbar.

Zwei kleine Tipps zum Schluss

Das Umswitchen der Standby- zur Primary-Datenbank haben wir übrigens über das Command-Line Interface von Data Guard ausgeführt. Dazu zwei kleine Tipps:

- ✓ Die Verbindung immer via TNS herstellen, also über User und Passwort, und nicht lokal mit OS-Authentifizierung.
- ✓ Bitte immer „Screen" benutzen.

Fazit

Rückblickend kann ich sagen, dass ich keine bessere Lösung für die Datenmigration innerhalb der Public Cloud empfehlen kann als Data Guard. Die Technologie bietet eine Vielfalt von Möglichkeiten, vor allem wenn es um Tests im Vorfeld der Migration geht. Die Snapshot-Standby-Datenbank, die man als normale Read/Write-Datenbank benutzen kann, ist unschlagbar. Dennoch gibt es Stolperfallen, auf die es aufzupassen gilt. Das mussten wir zum Beispiel bei den Scheduling-Jobs feststellen, die in ihrer Definition die Datenbankrolle beinhalteten.

Den Übergang zwischen physikalischer Standby-Funktion und Snapshot-Standby-Funktion haben wir bei unserer 300 Terabyte mächtigen Datenbank schnell hinbekommen: 2 Tage im Snapshot-Modus, 30 Minuten für den Switch zur physikalischen Datenbank.

Aber was sagte unser Kunde dazu?

Für den Online-Händler hatte sich der Umstieg auf die Oracle Exadata X9M gelohnt. Die Cyber Week wurde ein voller Erfolg. Nicht zuletzt dank der sehr ausgereiften Technologie und der Flexibilität der X9M, wenn es um die Skalierbarkeit von Ressourcen geht.

Quellen

1. *https://docs.oracle.com/en-us/iaas/exadatacloud/exacs/using-data-guard-with-exacc.html#GUID-6EBC4D6A-C58B-4721-B756-F22FC6819A45* - Data Guard
2. *https://docs.oracle.com/en-us/iaas/exadatacloud/exacs/ecs-using-dbaascli.html#GUID-232E5D26-420A-4B7A-BE30-79929C95314F* – dbascli
3. *https://docs.oracle.com/en-us/iaas/exadatacloud/exacs/ecs-create-instance.html#GUID-5E520C3B-F141-48C4-9AC2-0FE0AE8055FD* - Exadata Infrastructure & VM Cluster
4. OCI: How To Configure & Manage Database Backups On OCI EXACS DB System (Doc ID 2708469.1) - OCI Object Storage
5. Creating a Physical Standby database using RMAN restore database from service (Doc ID 2283978.1) – Restore/Recovery from service
6. Exadata Cloud Service Software Versions (Doc ID 2333222.1) – Exadata-Versionen im OCI

Andrzej Rydzanicz
andrzej.rydzanicz@opitz-consulting.com