Oracle HA を使用して Google Cloud NetApp Volumes をデプロイする

NetApp database solutions

NetApp
June 25, 2026

目次

Oracle HA を使用して Google Cloud NetApp Volumes をデプロイする

Google Cloud NetApp Volumes 用に Google Compute Engine インスタンスをプロビジョニングする

Google Compute Engine 仮想マシンをプロビジョニングして、Google Cloud NetApp Volumes iSCSI ストレージ上で Oracle Database 26ai をホストします。この手順では、プライマリおよびスタンバイデータベースホストと Fast-Start フェールオーバー オブザーバー VM の作成、Oracle Net 用の VPC ファイアウォール ルールの構成、ホスト名解決の設定、OS の準備、および GCNV ストレージ プロビジョニング用の iSCSI イニシエータ名の取得について説明します。

ステップ1：VMを作成する

ゾーン障害分離のために、同じリージョン内の異なるゾーンに3つのGoogle Compute Engine VMを作成します。Cloud Console、 `gcloud`、Terraform、または標準のプロビジョニング ワークフローを使用します。

1. 以下の表に示す仕様で、3つの仮想マシンを作成します。

   TCOと持続可能性の観点から、レイテンシーとコンプライアンスのニーズを満たす低炭素地域を優先します（例： `us-west1`対 `us-central1`）：

VM	ゾーン	マシンタイプ	ブートディスク	Network	目的
`oracdb1`	`us-west1-a`	`n4-highmem-8`（サンプル）または `c4-standard-*`	OL 10、50 GB Hyperdisk Balanced（OSのみ）	`oracle-vpc` / `oracle-subnet`、gVNIC	プライマリDB
`oracdb2`	`us-west1-b`	プライマリと同じ	OL 10、50 GB Hyperdisk Balanced（OSのみ）	同じ	スタンバイDB
`oradg-obs`	`us-west1-c`	`e2-medium`	OL 10、20 GB Hyperdisk Balanced	同じ	FSFOオブザーバー（Instant Clientのみ）

   レイテンシまたはエグレス（>~200 GiB/月）が重要な場合は Premium ネットワーク ティアを使用し、開発とテストで TCO を低く抑える場合は Standard を使用します。

2. シールドVM機能を有効にし、ブートディスクの構成を確認します：

   3台の仮想マシンすべてで、**Secure Boot**、**vTPM**、および*Integrity Monitoring*を有効にしてください。

   ブートディスクにはOSのみが格納されています。 `/u01` Grid/DB ホーム、ステージング、およびすべての

ASM データは GCNV iSCSI ボリュームを使用します （GCNV iSCSI ボリュームをプロビジョニングするを参照）

GCEデータディスクを別途接続しないでください `/u01`。

ステップ2：VPC ファイアウォールを TCP 1521 用に構成する

Oracle Net redoトランスポートとオブザーバー接続のために、3つのVMすべて間でTCP/1521を許可するVPCファイアウォールルールを作成します。ルールが欠落していると、Data Guardのレプリケーションが失敗します。

1. 3つのVMの内部IPアドレスすべて間でTCP/1521を許可するVPCファイアウォールのイングレスルールを作成します。同じ許可リストを持つVPCファイアウォールルールまたはファイアウォールポリシーを使用します：

   Cloud Console： VPC ネットワーク → ファイアウォール → ルールの作成 `allow-oracle-net-dbhosts`オン `oracle-vpc` — Ingress、Allow、sources = 3 つの `/32` IP、TCP 1521。必要に応じて Egress をミラーリングします。

2. 各仮想マシンからの接続性を検証して、ファイアウォールルールが正しく設定されていることを確認します：

```
sudo dnf install -y nmap-ncat

for tgt in <oracdb1-ip> <oracdb2-ip> <oradg-obs-ip>; do
  nc -zv -w 5 "$tgt" 22
  nc -zv -w 5 "$tgt" 1521
done
```

ポート	想定	説明
`22`	接続済み	SSHパスが機能します
`1521`	接続が拒否されました	ファイアウォールが開いています。グリッドリスナーはステップ1：各DBホストにOracle Grid Infrastructure（Oracle Restart）をインストールします中に起動します
どちらか	Timeout	ファイアウォールまたはルーティングを修正する

3 台の VM すべてから、各ピア IP に向けて実行します。

ステップ 3: ホスト名、DNS、および `/etc/hosts`

Oracle Net、Data Guard Broker、およびObserverの正引きおよび逆引き名前解決が正しく機能するように、3つのVMすべてでホスト名とDNS解決を設定します。

1. ホスト名を設定し、3つのホストすべてに `/etc/hosts`エントリを追加します。GCEの内部IPアドレス（*Compute Engine → VMインスタンス*リストの_Internal IP_列に表示されます）を置き換えてください。

```
# Run on each VM, substituting the local short name (oracdb1, oracdb2,
oradg-obs)
sudo hostnamectl set-hostname <this-host>.example.internal

# Run on every VM (same content)
sudo tee -a /etc/hosts >/dev/null <<EOF

# Oracle DG peers + FSFO Observer
<oracdb1-ip>    oracdb1.example.internal    oracdb1
<oracdb2-ip>    oracdb2.example.internal    oracdb2
<oradg-obs-ip>  oradg-obs.example.internal  oradg-obs
EOF
```

2. 各ホストからの名前解決を検証します：

```
ping -c 1 oracdb1 && ping -c 1 oracdb2 && ping -c 1 oradg-obs
```

ステップ 4：DB ホストのみで OS を準備する

`oracdb1`および `oracdb2`上のOSをOracle Database 26ai用に準備するには、プリインストールパッケージのインストール、ユーザーとグループの作成、iSCSIおよびマルチパスパッケージのインストール、iSCSIイニシエータの設定を行います。オブザーバーのセットアップについては、xref:gcnv-oracle-data-guard.adoc#install-instant-client-on-observer[ステップ4：オブザーバーホストにOracle Instant Clientをインストールする]を参照してください。

前提条件： `yum.oracle.com`へのアウトバウンドHTTPS（プライベートサブネット上のCloud NATまたは内部ミラー）。

1. Oracle Database のプリインストール パッケージをインストールし、 `grid`ユーザーと ASM グループを作成し、 `oracle`ユーザーを ASM グループに追加します：

```
# Oracle 26ai preinstall (package name varies by repo)
sudo dnf install -y oracle-ai-database-preinstall-26ai \
  || sudo dnf install -y oracle-database-preinstall-26ai \
  || sudo dnf install -y oracle-database-preinstall-23ai

# grid user + asm groups
sudo groupadd -g 54327 asmadmin; sudo groupadd -g 54328 asmdba; sudo
groupadd -g 54329 asmoper
sudo useradd -u 54322 -g oinstall -G dba,oper,asmadmin,asmdba,asmoper
grid
sudo passwd -l grid; sudo passwd -l oracle
sudo usermod -a -G asmdba,asmadmin oracle
```

2. iSCSI、マルチパス、およびJDKパッケージをインストールし、THPと時刻同期を確認します：

```
sudo dnf install -y iscsi-initiator-utils device-mapper-multipath
sg3_utils \
  java-21-openjdk-headless libxcrypt-compat

# THP and time
cat /sys/kernel/mm/transparent_hugepage/enabled   # expect [never]
timedatectl
chronyc tracking
```

3. SELinux、ファイアウォール、iSCSI イニシエータの設定を行い、再起動します。

> *セキュリティ態勢（OL 10）：*以下のコマンドはSELinuxをpermissiveに設定し、`firewalld`を無効にします。これは最小限のラボ環境の設定です。SELinuxとファイアウォールの強化された設定については、組織のセキュリティベースラインを参照してください。

```
sudo setenforce 0
sudo sed -i 's/^SELINUX=.*/SELINUX=permissive/' /etc/selinux/config
sudo systemctl disable --now firewalld

sudo cp -n /etc/iscsi/iscsid.conf /etc/iscsi/iscsid.conf.orig
sudo sed -i '/^[#[:space:]]*node\.session\.timeo\.replacement_timeout/d'
/etc/iscsi/iscsid.conf
echo "node.session.timeo.replacement_timeout = 120" | sudo tee -a
/etc/iscsi/iscsid.conf
sudo systemctl enable --now iscsid

sudo reboot
```

ステップ 5：iSCSI イニシエータ名 IQN を取得する

再起動後、各データベースホストからiSCSIイニシエータ名（IQN）を取得します。これらの IQN を使用して、ステップ2：ホストグループを作成する で GCNV ホスト グループを作成します。

1. IQNを `oracdb1`からキャプチャして記録します：

```
sudo cat /etc/iscsi/initiatorname.iscsi
# InitiatorName=iqn.1994-05.com.redhat:abc123def456
```

2. 繰り返して `oracdb2`そのIQNを記録します。ホストごとに1つのホストグループを使用することで、1つのホストの再起動やIQNの再生成が、他のホストのGCNV iSCSIボリュームの可視性に影響を与えないようにします。

> クローンされた **VM**： 両方のホストが同じ IQN を共有している場合は、 `oracdb2`で再生成します（停止 `iscsi`、クリア `/var/lib/iscsi/nodes/*`、新しい `InitiatorName`を `/etc/iscsi/initiatorname.iscsi`に、再起動 `iscsid`）。

次の手順

OracleバイナリとASM ディスク グループに共有ストレージを提供するには、Google Cloud NetApp Volumes の iSCSI プール、ホスト グループ、およびボリュームをプロビジョニングするに移動します。

Oracle Database 26ai 向け Google Cloud NetApp Volumes iSCSI ストレージのプロビジョニング

Google Compute Engine 上の Oracle Database 26ai 高可用性向けに Google Cloud NetApp Volumes iSCSI ブロックストレージをプロビジョニングします。この手順では、GCNV Flex Unified ストレージ プールの作成、ホスト グループの定義、各データベース ホストの iSCSI ボリュームの作成、Linux iSCSI とマルチパスの構成、ASM バッキング デバイスのパーティショニング、および `/u01`ファイルシステムのマウントについて説明します。

ステップ 1：GCNV iSCSI プールを作成する

プライマリホストとスタンバイホストにiSCSIボリュームを提供するために、各データベースゾーンに1つずつ、合計2つのFlex Unifiedストレージプールを作成します。各データベースホストは、自身のローカルゾーンのプールからボリュームを使用します。

1. Cloud Console を使用して、2つのストレージ プールを作成します。下記の表の仕様を使用して、各ゾーンの作成プロセスを繰り返します。

プール名	ゾーン	使用者
`oracle-pool-a`	`us-west1-a`	`oracdb1`（プライマリ）

プール名	ゾーン	使用者
`oracle-pool-b`	`us-west1-b`	`oracdb2`（スタンバイ）

NetApp Volumes → **Storage pools** → **Create** （各プールに対して）：

- サービスレベル： Flex（Premiumではありません）
- タイプ： 統合型
- *ゾーン：*データベースVMゾーンに一致します(`us-west1-a`/ `us-west1-b`)
- **PSA:** に接続 `oracle-vpc`
- *容量：*ワークロードに合わせてサイズが設定されます。リドゥ、バックアップ、またはリストアがデフォルトの余裕を超える場合（製品制限ごとにプールあたり最大5120 MiB/sまたは160K IOPS）、カスタムプロビジョニングされたスループット/IOPSを使用します。

2. 両方のプールが `READY`ステータスになるまで待ってから続行してください。プールのサイズをデータベースのフットプリントに合わせてスケールします（ステップ3：GCNV iSCSIボリュームを作成する の値は例です）：

> デフォルトモード（このガイド）： Flex Unified プールはデフォルトモードを使用します(`--mode=default`)。Cloud Console または `gcloud netapp`でプールと iSCSI ボリュームを作成します。ボリュームのレプリケーション、スナップショット、クローンは、Google Cloud API を使用します（ステップ3：GCNVスタンバイの初期化）。

ステップ 2：ホスト グループを作成する

データベースホストごとにホストグループを1つ作成することで、各仮想マシンは自身のボリュームのみを参照できるようになります。プライマリホストとスタンバイホストは、独立したストレージを維持するために、GCNV iSCSIボリュームを共有してはなりません。

1. Cloud Console を使用して `oracdb1`のホストグループを作成します：

   NetApp Volumes → ホストグループ → 作成

   - 名前： `oracdb1-hg`
   - リージョン： `us-west1`
   - タイプ： iSCSIイニシエータ
   - **OS** タイプ： Linux
   - ホスト： `oracdb1`からIQNを貼り付けます（ `/etc/iscsi/initiatorname.iscsi`の値）
   - 概要： "Oracleプライマリホスト oracdb1"
   - 作成

2. プロセスを繰り返して `oracdb2`名前 `oracdb2-hg`と`oracdb2`のIQNを使用します。オブザーバーホストにはGCNVリソースは必要ありません。

ステップ3：GCNV iSCSI ボリュームを作成する

各データベースホスト用に 5 つの GCNV iSCSI ボリュームを作成します。1 つは `/u01`用、4 つは ASM バッキングデバイス用です。各ホストのボリュームは、対応するホストグループとともに、そのホストのローカルゾーンのストレージプール内に作成する必要があります。

1. `oracdb1`の5つのボリュームを `oracle-pool-a`でホストグループ `oracdb1-hg`を使用して作成します。以下の表の仕様を使用してください。

GCNV iSCSI ボリューム	サイズ	用途	マルチパス エイリアス
`ora_<host>_u01`	100 GiB	`/u01` GCNV iSCSI ボリューム — Grid/Oracle ホーム、ステージング	`/dev/mapper/ora_<host>_u01`
`ora_<host>_data_01`	50 GiB	ASM `+DATA`	`/dev/mapper/ora_<host>_data_01`
`ora_<host>_data_02`	50 GiB	ASM `+DATA` （ストライプ）	`/dev/mapper/ora_<host>_data_02`
`ora_<host>_arch_01`	100 GiB	ASM `+RECO`	`/dev/mapper/ora_<host>_arch_01`
`ora_<host>_fra_01`	100 GiB	ASM `+FRA`	`/dev/mapper/ora_<host>_fra_01`

ボリューム名：英字、数字、アンダースコアのみ使用可能（ハイフンは不可）。

> 最小限のレイアウト（検証のみ）： ホストあたり2つのLUN(`*_data *_reco`）と `arch_01p1→+RECO`および `arch_01p2→`+FRA`はラボでの使用には適しています。本番環境ではステップ3：GCNV iSCSIボリュームを作成するあたり5つのボリュームを使用します。

2. 同じ仕様を使用して、 `oracle-pool-b`内の `oracdb2`用にホストグループ `oracdb2-hg`を持つ5つのボリュームを作成します。各プールで、**NetApp Volumes → Volumes → Create** — iSCSI、正しいプールおよびホストグループ、Linuxを使用します。次の情報を記録します：
   - iSCSI ポータル IP → `<ISCSI_PORTAL_1>`、 `<ISCSI_PORTAL_2>`（ `oracdb1` 上のプライマリプールポータル、 `oracdb2` 上のスタンバイプールポータル。両者は異なる場合があります）
   - クラウドコンソールからのボリュームシリアル：ホストで検出されたWWIDで使用ステップ4：Google Cloud NetApp Volumes iSCSIボリューム用にLinux iSCSIとマルチパスを設定する

ステップ4：iSCSIとマルチパスの設定

各データベースホストでiSCSIとdevice-mapper-multipathを設定し、両方のストレージポータルIP経由でGCNVボリュームにアクセスできるようにします。以下の手順を `oracdb1`プライマリプールのポータルIPを使用して実行し、次に `oracdb2`スタンバイプールのポータルIPを使用して繰り返します。ホストの送信が制限されている場合、各データベースVMからGCNV iSCSIポータルIPへのTCP/3260を許可します（ステップ2：VPCファイアウォール — 3つのゾーンすべてでTCP/1521を許可リストに追加するからのVM間TCP/1521に加えて）。

1. ターゲットを検出し、ログインして、ノードの起動状態を維持する：

```
sudo iscsiadm --mode discovery --op update --type sendtargets --portal
<ISCSI_PORTAL_1>
sudo iscsiadm --mode discovery --op update --type sendtargets --portal
<ISCSI_PORTAL_2>
sudo iscsiadm --mode node --op update --name node.startup --value
automatic
sudo iscsiadm --mode node -l all
sudo systemctl enable --now iscsid iscsi multipathd
sudo iscsiadm --mode session         # expect 10 sessions (5 GCNV iSCSI
volumes × 2 portals)
sudo lsblk -o NAME,SIZE,WWN,VENDOR,MODEL
```

再起動後、Oracle を起動する前に再度確認してください：

```
sudo iscsiadm --mode session
sudo multipath -ll
```

2. デフォルト設定とブラックリストルールを使用して `device-mapper-multipath`を設定する：

```
sudo tee /etc/multipath.conf >/dev/null <<'EOF'
defaults {
    find_multipaths     yes
    user_friendly_names yes
}
blacklist {
    devnode "^(ram|raw|loop|fd|md|dm-|sr|scd|st)[0-9]*"
    devnode "^hd[a-z]"
    devnode "^cciss.*"
}
EOF

sudo systemctl enable --now multipathd
sudo multipath -ll
```

3. ホストが検出した WWID エイリアスを `/etc/multipath.conf`に追加します（推測しないでください — `multipath.conf`はシェル変数を展開*しません*）。WWID を検出：

```
sudo multipath -ll
for dev in /dev/sd*; do
  [ -b "$dev" ] || continue
  printf '%s: ' "$dev"
  sudo /usr/lib/udev/scsi_id --whitelisted --device="$dev" 2>/dev/null
|| true
  echo
done
```

そのホストの具体的なエイリアスを `/etc/multipath.conf`に追加し、次に `sudo systemctl restart multipathd`を実行します。

`oracdb1`で、次を追加します。

```
multipaths {
    multipath { wwid <host-discovered-wwid-for-u01>      alias
ora_oracdb1_u01     }
    multipath { wwid <host-discovered-wwid-for-data-01>  alias
ora_oracdb1_data_01 }
    multipath { wwid <host-discovered-wwid-for-data-02>  alias
ora_oracdb1_data_02 }
    multipath { wwid <host-discovered-wwid-for-arch-01>  alias
ora_oracdb1_arch_01 }
    multipath { wwid <host-discovered-wwid-for-fra-01>   alias
ora_oracdb1_fra_01  }
}
```

`oracdb2`では、 `ora_oracdb2_*`エイリアスで同じパターンを使用し、次に：

```
sudo systemctl restart multipathd
ls -l /dev/mapper/ora_$(hostname -s)_*
```

ステップ5：ASMデバイスをパーティション分割する

4 つの ASM バッキング デバイス （ u01 を除く） をそれぞれ 1 つの GPT パーティションで ASM 消費用にパーティション分割し、グリッド所有権の udev ルールを設定します。各データベースホストで以下の手順を実行します。

1. 4つのASMバッキングデバイスをGPTでパーティション分割し、パーティションを確認します：

```
HOST=$(hostname -s)        # oracdb1 on the primary, oracdb2 on the
standby
for dev in /dev/mapper/ora_${HOST}_data_01 \
           /dev/mapper/ora_${HOST}_data_02 \
           /dev/mapper/ora_${HOST}_arch_01 \
           /dev/mapper/ora_${HOST}_fra_01; do
  sudo parted -s "$dev" mklabel gpt
  sudo parted -s "$dev" mkpart primary 0% 100%
done
sudo partprobe
sudo systemctl reload multipathd
ls /dev/mapper/ora_${HOST}_*p1     # expect 4 partitions
```

2. グリッドの所有権を割り当て、変更をトリガーするようにudevルールを設定します。

```
HOST=$(hostname -s)
sudo tee /etc/udev/rules.d/99-oracle-asm.rules >/dev/null <<'EOF'
KERNEL=="dm-*", ENV{DM_UUID}=="part?-mpath-*",
ENV{DM_NAME}=="ora_oracdb*_*p?", \
    OWNER="grid", GROUP="asmadmin", MODE="0660"
EOF

sudo udevadm control --reload-rules
for part in /dev/mapper/ora_${HOST}_*p1; do
  dm=$(readlink -f "$part" | xargs basename)
  sudo udevadm trigger --action=change --name-match="/dev/${dm}"
done
sudo udevadm settle
ls -lL /dev/mapper/ora_${HOST}_*p1    # grid:asmadmin 0660
```

ステップ6：フォーマットとマウント /u01

`ora_<host>_u01` GCNV ボリュームを XFS でフォーマットし、 `/etc/fstab` で UUID を使用して永続的にマウントします。 `/u01` ファイルシステムには、Grid ホーム、Oracle ホーム、およびステージング ファイルが格納されます。

1. マルチパスデバイスをXFSでフォーマットし、そのUUIDを取得します。

```
HOST=$(hostname -s)
U01_DEV=/dev/mapper/ora_${HOST}_u01
ls -l "$U01_DEV"

sudo mkfs.xfs -f "$U01_DEV"
U01_UUID=$(sudo blkid -s UUID -o value "$U01_DEV")
```

2. UUIDベースのマウントエントリを `/etc/fstab`に追加し、ファイルシステムをマウントします：

```
sudo mkdir -p /u01
echo "UUID=${U01_UUID} /u01 xfs defaults,_netdev,nofail,x-
systemd.requires=iscsi.service,x-systemd.requires=multipathd.service,x-
systemd.after=iscsi.service,x-systemd.after=multipathd.service 0 0" |
sudo tee -a /etc/fstab
sudo mount -a
```

3. GridおよびOracleソフトウェア用に適切な所有権を持つディレクトリ構造を作成します。

```
sudo mkdir -p /u01/app/oraInventory /u01/app/26ai/grid /u01/app/grid \
  /u01/app/oracle/product/26ai/db_1 /u01/stage
sudo chown -R grid:oinstall /u01/app/oraInventory /u01/app/26ai
/u01/app/grid
sudo chown -R oracle:oinstall /u01/app/oracle /u01/stage
sudo chmod -R 775 /u01/app /u01/stage
```

1回再起動し、Oracleソフトウェアのインストール の前に `/u01` がマウントされることを確認します。

次の手順

Oracle Grid InfrastructureおよびDatabaseバイナリを準備済みのホストにインストールするには、両方のホストでOracle Grid InfrastructureおよびOracle Databaseソフトウェアをインストールしますに移動してください。

Google Cloud NetApp Volumes に Oracle Grid Infrastructure と Oracle Database 26ai をインストールする

各データベースホスト用のGoogle Cloud NetApp Volumes iSCSIストレージに Oracle Restart と ASM を使用して Oracle Grid Infrastructure をインストールし、その後 Oracle Database 26ai ソフトウェアをインストールします。この手順には、Oracle GoldImages のステージング、応答ファイルを使用したサイレントインストールの実行、GCNV ボリューム上への ASM ディスク グループの作成、およびデータベース作成前にプライマリホストとスタンバイホストの両方に同一の Oracle ソフトウェアを準備することが含まれ

ます。

ステップ 1：各 DB ホストに Grid Infrastructure をインストールする

各データベースホストに Oracle Grid Infrastructure GoldImage をインストールして、Oracle Restart と ASM を有効にします。どちらのホストも、それぞれ独自の Grid ホーム、ASM インスタンス、およびディスク グループが必要です。Data Guard は、共有ストレージではなく、Oracle Net を介してデータを複製します。すべてのステップを `oracdb1`で完了してから、 `oracdb2`で繰り返してください。

1. Oracle GoldImages、リリースアップデート、およびOPatchバイナリを `/u01/stage`にステージングします：

```
sudo chown oracle:oinstall /u01/stage && sudo chmod 775 /u01/stage
# Upload GoldImages, RU, OPatch to /u01/stage.
```

2. ターゲットのグリッドホームでグリッド GoldImage を解凍します。26ai GoldImage は、ターゲットディレクトリに直接解凍してインストールします。

```
sudo -u grid bash -c '
cd /u01/app/26ai/grid
unzip -q /u01/stage/LINUX.X64_<RELEASE>_grid_home.zip
'
sudo chown -R grid:oinstall /u01/app/26ai/grid
```

Grid GoldImageがターゲット RU より古い場合は、 `gridSetup.sh -applyRU`フローを使用してセットアップ中に Grid ホームにパッチを適用するか、RU が同梱された GoldImage を使用してください。Grid と Database のホームを同じ意図したパッチ レベルに維持してください。

3. 各ホストで `gridSetup`レスポンスファイル `/tmp/grid.rsp`を構築します。ホスト名を置き換え、強力なパスワードを使用してください：

```
HOST=$(hostname -s)

sudo -u grid bash -c "cat > /tmp/grid.rsp <<RSP
oracle.install.responseFileVersion=/oracle/install/rspfmt_crsinstall_res
ponse_schema_v23.0.0
INVENTORY_LOCATION=/u01/app/oraInventory
installOption=HA_CONFIG
ORACLE_BASE=/u01/app/grid
clusterUsage=GENERAL_PURPOSE
OSDBA=asmdba
OSOPER=asmoper
OSASM=asmadmin
storageOption=FLEX_ASM_STORAGE
sysasmPassword=WelcomeOracle1!
asmsnmpPassword=WelcomeOracle1!
diskGroupName=DATA
redundancy=EXTERNAL
auSize=4
diskString=/dev/mapper/ora_${HOST}_*p*
diskList=/dev/mapper/ora_${HOST}_data_01p1,/dev/mapper/ora_${HOST}_data_
02p1
managementOption=NONE
RSP"
sudo -u grid chmod 600 /tmp/grid.rsp
```

4. `gridSetup.sh`をサイレントモードで実行して、バイナリをコピーし、設定を準備します。`Successfully Setup Software with warning(s).`と終了コード6（警告）または0を想定します：

```
sudo -u grid bash -c '
export ORACLE_HOME=/u01/app/26ai/grid
export ORACLE_BASE=/u01/app/grid
cd /u01/app/26ai/grid
./gridSetup.sh -silent -responseFile /tmp/grid.rsp -ignorePrereqFailure
'
```

5. `orainstRoot.sh`および `root.sh`をrootとして実行します。 `root.sh`スクリプトは`crsctl、 srvctl、および`asmcmd`ラッパーを作成し、OHASを起動します。

```
sudo /u01/app/oraInventory/orainstRoot.sh
sudo /u01/app/26ai/grid/root.sh
```

6. `gridSetup.sh -executeConfigTools`を実行して、応答ファイルに対して構成アシスタント（NETCA、ASMCA、CVU）を実行します。これにより、ASM インスタンスと `+DATA ディスク グループ

が作成されます。NETCA / ASMCA / CVU 後に `Successfully Configured Software.`が表示されます：

```
sudo -u grid bash -c '
export ORACLE_HOME=/u01/app/26ai/grid
export ORACLE_BASE=/u01/app/grid
cd /u01/app/26ai/grid
./gridSetup.sh -silent -executeConfigTools -responseFile /tmp/grid.rsp
'
```

7. `+RECO`および`+FRA`のディスク グループを `asmca`を使用して作成します。シングルショットインストールでは、 `+DATA`のみが作成されます：

```
HOST=$(hostname -s)

sudo -u grid bash -c "
export ORACLE_HOME=/u01/app/26ai/grid
export ORACLE_SID=+ASM

\$ORACLE_HOME/bin/asmca -silent -createDiskGroup \
  -diskGroupName RECO \
  -disk /dev/mapper/ora_${HOST}_arch_01p1 \
  -redundancy EXTERNAL -au_size 4

\$ORACLE_HOME/bin/asmca -silent -createDiskGroup \
  -diskGroupName FRA \
  -disk /dev/mapper/ora_${HOST}_fra_01p1 \
  -redundancy EXTERNAL -au_size 4
"
```

8. ASM ディスク グループと Oracle Restart リソースのステータスを確認します。

```
sudo -u grid ORACLE_HOME=/u01/app/26ai/grid ORACLE_SID=+ASM \
  /u01/app/26ai/grid/bin/sqlplus -s / as sysasm <<'SQL'
SELECT name, total_mb, free_mb, state FROM v$asm_diskgroup ORDER BY
name;
SQL

sudo /u01/app/26ai/grid/bin/crsctl stat res -t
# Expected ONLINE: ora.DATA.dg, ora.RECO.dg, ora.FRA.dg,
ora.LISTENER.lsnr, ora.asm, ora.cssd, ora.evmd.
```

9. 上記のステップを `oracdb2`で繰り返します。ステップ3と4およびステップ7内の `HOST=$(hostname -s)`パターンにより、そのホストのGCNV iSCSIデバイスが自動的に選択されます。

同じASMディスク グループ名を使用します—Data Guardはストレージではなく Oracle Net経由でレプリケートします。

ステップ2：各DBホストにOracle Databaseをインストールする

最新のリリースアップデートを適用した上で、サイレントインストール（ソフトウェアのみのインストール）を使用して、各データベースホストにOracle Database 26aiソフトウェアホームをインストールします。すべてのステップを `oracdb1`で完了してから、 `oracdb2`で繰り返します。

1. データベースホーム、最新のOPatch、およびRUパッチをそれぞれのディレクトリに解凍します。RUディレクトリのレイアウトと `-applyRU`パスについては、Oracleのドキュメントを参照してください：

```
sudo su - oracle
cd /u01/app/oracle/product/26ai/db_1
unzip -q /u01/stage/LINUX.X64_<RELEASE>_db_home.zip
rm -rf OPatch
unzip -q /u01/stage/p6880880_<base>_Linux-x86-64.zip
# latest OPatch
unzip -q /u01/stage/p<RU_PATCH>_<base>_Linux-x86-64.zip -d /u01/stage
# latest 26ai RU
```

2. インストール応答ファイルを作成し、RUを適用したサイレント ソフトウェアのみのインストールを実行します。OL 8/9では、 `runInstaller`行から `-applyOneOffs`を省略します：

```
sudo -u oracle tee /u01/stage/dbinstall.rsp >/dev/null <<'EOF'
oracle.install.option=INSTALL_DB_SWONLY
UNIX_GROUP_NAME=oinstall
INVENTORY_LOCATION=/u01/app/oraInventory
ORACLE_HOME=/u01/app/oracle/product/26ai/db_1
ORACLE_BASE=/u01/app/oracle
oracle.install.db.InstallEdition=EE
oracle.install.db.OSDBA_GROUP=dba
oracle.install.db.OSOPER_GROUP=oper
oracle.install.db.OSBACKUPDBA_GROUP=backupdba
oracle.install.db.OSDGDBA_GROUP=dgdba
oracle.install.db.OSKMDBA_GROUP=kmdba
oracle.install.db.OSRACDBA_GROUP=racdba
oracle.install.db.rootconfig.executeRootScript=false
EOF

sudo -u oracle bash -c '
export CV_ASSUME_DISTID=OEL10    # OEL9 / OEL8.10 if cluvfy requires it
cd /u01/app/oracle/product/26ai/db_1
./runInstaller -applyRU /u01/stage/<RU_PATCH> \
  -applyOneOffs /u01/stage/39292021 \
  -silent -ignorePrereqFailure -responseFile /u01/stage/dbinstall.rsp
'
```

3. インストール後のルートスクリプトを実行します。

```
sudo /u01/app/oracle/product/26ai/db_1/root.sh
```

4. 各DBホストでOracle環境を設定します。 `ORACLE_SID=orcl`を `oracdb1`で使用し、 `ORACLE_SID=orcls`を `oracdb2`で使用します：

```
sudo -u oracle tee -a /home/oracle/.bash_profile >/dev/null <<'EOF'
export ORACLE_HOME=/u01/app/oracle/product/26ai/db_1
export ORACLE_SID=orcl                          # use 'orcls' on oracdb2
export GRID_HOME=/u01/app/26ai/grid
export PATH=$ORACLE_HOME/bin:$GRID_HOME/bin:$PATH
export TNS_ADMIN=$ORACLE_HOME/network/admin
EOF
```

スタンバイデータベースはスタンバイデータベースを作成するに作成されます。

次の手順

HAデプロイ用の本番プライマリ インスタンスを作成するには、 `oracdb1`のOracleプライマリデータベースを作成するに移動します。

Google Cloud NetApp Volumes に Oracle プライマリデータベースを作成する

Oracle Database Configuration Assistantをサイレントモードで使用して、Google Cloud NetApp VolumesのiSCSIストレージにOracleプライマリデータベースを作成します。この手順では、 `dbca`を実行して、GCNVを基盤とするASMディスク グループ上にコンテナ データベースとプラガブル データベースを作成し、アーカイブ ログの宛先を構成し、Data Guardを有効にした後の透過的なフェイルオーバーのためのロールベース アプリケーション サービスを追加する方法について説明します。

手順

Oracleコンテナデータベースとプラガブルデータベースを `oracdb1`上に作成し、 `dbca`をサイレントモードで使用して、アーカイブログの宛先を構成し、Oracle Restartの登録を確認し、透過的なクライアントフェイルオーバーのためのロールベース アプリケーション サービスを追加します。

1. サイレントモードで実行 `dbca`し、ASM ディスク グループ上にCDBとPDBを作成します：

```
sudo -u oracle bash -c '
export ORACLE_HOME=/u01/app/oracle/product/26ai/db_1
export PATH=$ORACLE_HOME/bin:$PATH

dbca -silent -createDatabase \
  -templateName General_Purpose.dbc \
  -gdbname orcl -sid orcl \
  -characterSet AL32UTF8 -nationalCharacterSet AL16UTF16 \
  -sysPassword "ChangeMe!1" -systemPassword "ChangeMe!1" \
  -emConfiguration NONE \
  -datafileDestination +DATA -storageType ASM \
  -recoveryAreaDestination +FRA -recoveryAreaSize 25000 \
  -enableArchive true -archiveLogMode AUTO \
  -memoryMgmtType AUTO_SGA -totalMemory 4096 \
  -databaseType MULTIPURPOSE \
  -createAsContainerDatabase true -numberOfPDBs 1 \
  -pdbName orclpdb -pdbAdminPassword "ChangeMe!1" \
  -ignorePreReqs
'
```

2. アーカイブログを `+RECO`にポイントし、プラグイン可能なデータベースの状態を開いて保存します。スタンバイは、ステップ 2：スタンバイ init.ora、pfile、および NOMOUNT内の一致するアーカイブログ設定を使用します：

```
sudo -u oracle bash -c '
export ORACLE_HOME=/u01/app/oracle/product/26ai/db_1
export ORACLE_SID=orcl
$ORACLE_HOME/bin/sqlplus -s / as sysdba <<SQL
ALTER SYSTEM SET log_archive_dest_1='\''LOCATION=+RECO
VALID_FOR=(ALL_LOGFILES,ALL_ROLES) DB_UNIQUE_NAME=orcl'\'' SCOPE=BOTH;
ALTER PLUGGABLE DATABASE ALL OPEN;
ALTER PLUGGABLE DATABASE ALL SAVE STATE;
EXIT
SQL
'
```

3. Oracle Restart でデータベースが実行されていることを確認します。

```
sudo /u01/app/26ai/grid/bin/srvctl status database -d orcl
# Expected: Database is running

sudo -u oracle sqlplus -s / as sysdba <<<"SELECT name, open_mode,
log_mode FROM v\$database;"
# Expected: ORCL, READ WRITE, ARCHIVELOG
```

4. ロールベース アプリケーション サービスを作成して、アプリケーションが `orclapp`経由で接続し、Data Guard が有効になっている場合にフェイルオーバーが透過的に行われるようにします：

```
sudo -u oracle bash -c '
export GRID_HOME=/u01/app/26ai/grid
export ORACLE_HOME=/u01/app/oracle/product/26ai/db_1
export PATH=$ORACLE_HOME/bin:$GRID_HOME/bin:$PATH

srvctl add service \
  -db orcl \
  -service orclapp \
  -pdb orclpdb \
  -role PRIMARY \
  -policy AUTOMATIC

srvctl start service -db orcl -service orclapp
srvctl status service -db orcl -service orclapp
'
```

Data Guard Brokerが有効になった後、 `orclapp`はPRIMARYでのみ動作します。制御ファイルをASM ディスク グループ全体に多重化し、ワークロードに合わせてメモリサイズを調整します。

次の手順

スタンバイ保護を確立し、フェイルオーバーに備えるには、 `oracdb2` の Oracleスタンバイデータベースを作成する に移動します。

Google Cloud NetApp Volumes のストレージ レイヤ シードを使用して Oracle スタンバイ データベースを作成する

Google Cloud NetApp Volumes のストレージ レイヤ レプリケーション、スナップショット、またはクローンを使用して Oracle の物理スタンバイデータベースを作成し、従来の RMAN 方式と比較してスタンバイの初期化を高速化します。この手順では、リスナーの設定、スタンバイ pfile の作成、GCNV レプリケーションを使用したスタンバイボリュームのシード処理、Oracle インスタンスの最終処理、および Oracle Restart へのスタンバイの登録について説明します。すべての HA ティアでこれらの手順を実行します。**Prod HA (Data Guard + FSFO)** ティアの場合は、データガードの最終化 を設定する前に続けてください。Data Guard Broker、Fast-Start Failover、およびObserver

ステップ 1：リスナーと Data Guard パラメーターを設定する

両方のデータベースホストでリスナーを構成して、ブローカーに必要な `_DGMGRL`サービスを含む Data Guard 接続をサポートします。プライマリデータベース上でパスワードファイルを設定し、アーカイブログのパラメータを構成します。

1. プライマリリスナーを設定し、環境を確認します `oracdb1`：

```
sudo su - oracle
. ~/.bash_profile        # ORACLE_SID=orcl, ORACLE_HOME set
```

2. スタンバイリスナーを `oracdb2`に設定して、 `orcls`および `orcls_DGMGRL`サービスを含めます：

```
GRID_HOME=/u01/app/26ai/grid
sudo -u grid tee "$GRID_HOME/network/admin/listener.ora" >/dev/null <<
'EOF'
LISTENER =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = oracdb2.example.internal)(PORT =
1521)))

SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC = (GLOBAL_DBNAME = orcls)        (ORACLE_HOME =
/u01/app/oracle/product/26ai/db_1) (SID_NAME = orcls))
    (SID_DESC = (GLOBAL_DBNAME = orcls_DGMGRL) (ORACLE_HOME =
/u01/app/oracle/product/26ai/db_1) (SID_NAME = orcls)))
EOF
```

3. 両方のホストで Oracle Restart を使用してリスナーを再起動し、 `_DGMGRL`サービスが登録されていることを確認します：

```
sudo -u grid bash -c '
export GRID_HOME=/u01/app/26ai/grid
export ORACLE_HOME=$GRID_HOME
$GRID_HOME/bin/srvctl stop listener
$GRID_HOME/bin/srvctl start listener
$GRID_HOME/bin/lsnrctl status
'
```

`lsnrctl status`をリストする必要があります `<SID>`および `<SID>_DGMGRL。

ステップ2：スタンバイpfileを準備し、NOMOUNTを実行する

スタンバイデータベースインスタンスを準備するには、プライマリからパスワードファイルをコピーし、Data Guard パラメータを含む最小限の init.ora pfile を作成し、インスタンスを NOMOUNT モードで起動します。

1. IAP を使用してプライマリパスワードファイルをスタンバイホストにコピーし、 `gcloud compute scp` ：

```
PRIMARY_ZONE=us-west1-a       # zone of oracdb1
STANDBY_ZONE=us-west1-b       # zone of oracdb2

gcloud compute scp \
  oracdb1:/u01/app/oracle/product/26ai/db_1/dbs/orapworcl ./orapworcl \
  --zone=$PRIMARY_ZONE --tunnel-through-iap

gcloud compute scp \
  ./orapworcl oracdb2:/u01/app/oracle/product/26ai/db_1/dbs/orapworcls \
  --zone=$STANDBY_ZONE --tunnel-through-iap
```

2. プライマリデータベースから `compatible`パラメータ値をクエリします：

```
# On oracdb1
sudo -u oracle sqlplus -s / as sysdba \
  <<<"SELECT value FROM v\$parameter WHERE name='compatible';"
```

3. `oracdb2`でスタンバイpfileを作成し、パスワードファイルの所有権を設定して、インスタンスをNOMOUNTモードで起動します。前のステップの `compatible`の値を `<COPY_FROM_PRIMARY>`に代入します：

```
sudo -u oracle mkdir -p /u01/app/oracle/admin/orcls/adump
sudo chown oracle:oinstall
/u01/app/oracle/product/26ai/db_1/dbs/orapworcls
sudo chmod 0600 /u01/app/oracle/product/26ai/db_1/dbs/orapworcls

sudo -u oracle tee /u01/app/oracle/product/26ai/db_1/dbs/initorcls.ora
>/dev/null <<'EOF'
*.db_name='orcl'
*.db_unique_name='orcls'
*.audit_file_dest='/u01/app/oracle/admin/orcls/adump'
*.diagnostic_dest='/u01/app/oracle'
*.compatible='<COPY_FROM_PRIMARY>'
*.sga_target=3072m
*.pga_aggregate_target=1024m
*.processes=320
*.remote_login_passwordfile='EXCLUSIVE'
*.standby_file_management='AUTO'
*.fal_server='orcl'
*.log_archive_config='DG_CONFIG=(orcl,orcls)'
*.log_archive_dest_1='LOCATION=+RECO VALID_FOR=(ALL_LOGFILES,ALL_ROLES)
DB_UNIQUE_NAME=orcls'
*.log_archive_dest_2='SERVICE=orcl AFFIRM SYNC
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE) DB_UNIQUE_NAME=orcl'
```

```
*.log_archive_dest_state_2='DEFER'
*.log_archive_format='%t_%s_%r.arc'
*.dg_broker_start=TRUE
*.undo_tablespace='UNDOTBS1'
*.open_cursors=300
*.db_create_file_dest='+DATA'
*.db_create_online_log_dest_1='+DATA'
*.db_recovery_file_dest='+FRA'
*.db_recovery_file_dest_size=25000m
EOF

echo "orcls:/u01/app/oracle/product/26ai/db_1:N" | sudo tee -a
/etc/oratab

sudo -u oracle bash -c '
export ORACLE_HOME=/u01/app/oracle/product/26ai/db_1
export ORACLE_SID=orcls
sqlplus / as sysdba <<SQL
STARTUP NOMOUNT
PFILE=/u01/app/oracle/product/26ai/db_1/dbs/initorcls.ora;
EXIT
SQL
'
```

スタンバイインスタンスは現在、ステップ3：GCNVを使用してスタンバイストレージを初期化する までデータファイルなしでNOMOUNTモードになっています。

ステップ3：GCNVを使用してスタンバイストレージを初期化する

スタンバイボリュームを `oracle-pool-b`にシードし、それらを `oracdb2`にアタッチし、ASMディスクグループをマウントして、スタンバイインスタンスをMOUNT状態に確定します。

本番環境へのシードには GCNV レプリケーションを使用し、単発のラボワークフローにはスナップショットシードを使用してください。

シードパスを選択します

環境とリカバリ要件に基づいて、スタンバイシード方式を選択してください。

- 本番環境での推奨: レプリケーションパス：レプリケーションの作成と同期 および レプリケーションパス：スタンバイボリュームのカットオーバーと接続 のレプリケーションパスを使用してください。
- ラボ向けの代替案: 代替パス：スナップショットからのシードを使用します。

すべてのパスはスタンバイASMディスクグループのマウントとスタンバイインスタンスを確定するで合流します。

前提条件を確認する

スタンバイボリュームにシードデータを送信する前に、以下の前提条件を確認してください。

- `gcloud netapp`ボリューム レプリケーションに対応しています。
- 異なる場所にある2つの*デフォルトモード*プール(`oracle-pool-a oracle-pool-b`）。
- プライマリ プール上のソース ボリュームが `oracdb1-hg`に接続され、レプリケーションによって宛先ボリュームが作成されます。
- レプリケーションは、DB VMからではなく、Cloud Shellまたはワークステーションから実行してください。
- `oracdb2`で、ステップ4、ステップ5、およびステップ6から iSCSI および ASM ホストのセットアップを完了します。

```
export PROJECT=<your-gcp-project>
export LOC_A=us-west1-a
export LOC_B=us-west1-b
export DEST_POOL="projects/${PROJECT}/locations/${LOC_B
}/storagePools/oracle-pool-b"
```

- 必要に応じてスタンバイプールを作成します。

```
gcloud netapp storage-pools create oracle-pool-b \
  --project="${PROJECT}" --location="${LOC_B}" \
  --service-level=flex --type=unified --mode=default \
  --capacity=1024 --network=name=<your-vpc>
```

レプリケーションを作成して同期する

プライマリボリュームからスタンバイボリュームへのレプリケーション関係を作成し、初期同期が完了するまで待ちます。

```
gcloud netapp volumes replications create repl-oracdb2-data \
  --project="${PROJECT}" --location="${LOC_A}" --volume=oracdb1_data \
  --replication-schedule=EVERY_10_MINUTES \
  --destination-volume-parameters="storage_pool=${DEST_POOL
},volume_id=oracdb2_data,share_name=oracdb2_data"

gcloud netapp volumes replications create repl-oracdb2-reco \
  --project="${PROJECT}" --location="${LOC_A}" --volume=oracdb1_reco \
  --replication-schedule=EVERY_10_MINUTES \
  --destination-volume-parameters="storage_pool=${DEST_POOL
},volume_id=oracdb2_reco,share_name=oracdb2_reco"
```

+

`mirrorState`が
`MIRRORED`になり、各レプリケーションの初期同期が完了するまで待ちます。

切り替えてスタンバイボリュームを接続する

プライマリを静止状態にし、最終同期後にレプリケーションを停止し、宛先ボリュームをスタンバイホストグループにアタッチします。

プライマリ側で、書き込みを静止させ、リカバリメタデータを取得します。

```
ALTER DATABASE BEGIN BACKUP;
SELECT CURRENT_SCN FROM V$DATABASE;
ALTER DATABASE CREATE STANDBY CONTROLFILE AS '/tmp/orcls_stby.ctl';
```

最後に1回のレプリケーションサイクルを実行した後、レプリケーションを停止します。

```
gcloud netapp volumes replications stop repl-oracdb2-data \
  --project="${PROJECT}" --location="${LOC_A}" --volume=oracdb1_data
--force

gcloud netapp volumes replications stop repl-oracdb2-reco \
  --project="${PROJECT}" --location="${LOC_A}" --volume=oracdb1_reco
--force
```

宛先ボリュームを `oracdb2-hg`にアタッチします（複製されたLUNはソース名を保持する場合があります）：

```
HG=$(gcloud netapp host-groups describe oracdb2-hg --project="${PROJECT}"
\
  --location=us-west1 --format='value(name)')

gcloud netapp volumes update oracdb2_data --project="${PROJECT}"
--location="${LOC_B}" \
  --block-devices="name=oracdb1_data_lun,host-groups=${HG},os-type=LINUX"
```

スタンバイ制御ファイルを `oracdb2`にコピーし、プライマリのバックアップモードを終了します。

```
ALTER DATABASE END BACKUP;
```

スナップショットからのシード

連続レプリケーションが不要な場合の1回限りのラボシードには、このパスを使用してください。

一度限りのラボシードの場合、ソーススナップショットを作成し、そのスナップショットからスタンバイボリュームを `oracle-pool-b`（Cloud Console または API）で作成します。作成したボリュームを `oracdb2-hg` にアタッチし、スタンバイASMディスクグループのマウントに進みます。

スタンバイ**ASM**ディスクグループのマウント

スタンバイホストで、接続されているストレージパスを検出し、データベース リカバリの前に ASM ディスクグループをマウントします。

`oracdb2`で、スタンバイプール `iSCSI` ポータルにログインし、マルチパスデバイスを再スキャンします。`ASM` ディスクヘッダーがラボワークフローのプライマリ命名と一致する場合は、プライマリスタイルのエイリアス （例： `ora_oracdb1_data_01`、 `ora_oracdb1_arch_01`） を使用し、`asm_diskstring='/dev/mapper/ora_oracdb1_*p*'`を設定して、パーティションの所有権が `grid:asmadmin`であることを確認してから、ディスクグループをマウントします：

```
ALTER DISKGROUP DATA MOUNT FORCE;
ALTER DISKGROUP RECO MOUNT FORCE;
ALTER DISKGROUP FRA MOUNT FORCE;
```

スタンバイインスタンスを確定する

スタンバイ制御ファイルを復元し、キャプチャしたSCNにリカバリし、物理スタンバイに変換して、マネージドリカバリを開始します。

```
STARTUP NOMOUNT;
RESTORE STANDBY CONTROLFILE FROM '/tmp/orcls_stby.ctl';
ALTER DATABASE MOUNT;
RECOVER DATABASE UNTIL SCN <quiesce_scn>;
ALTER DATABASE CONVERT TO PHYSICAL STANDBY;
SHUTDOWN IMMEDIATE;
STARTUP MOUNT;
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM SESSION;
```

この時点で、スタンバイは `PHYSICAL STANDBY`および `MOUNTED`で、管理リカバリが開始された状態になっているはずです。

ティア別の次のステップ：

- **Prod HA (no Data Guard):** 直接ステップ4：Oracle Restartにスタンバイを登録するに進みます。
- **Prod HA (Data Guard + FSFO):** ステップ4：Oracle Restartにスタンバイを登録するに進み、次にData

Guardの最終処理手順に進みます。

ステップ4：Oracle Restartにスタンバイを登録する

スタンバイデータベースをOracle Restartに登録すると、再起動時にASMディスク グループの自動リカバリ、スタンバイデータベースのマウント、およびマネージドリカバリの再開が自動的に行われます。また、アプリケーション サービスを両方のデータベースリソースに追加します。

1. スタンバイデータベースからspfileの場所を取得し、 `oracdb2`のOracle Restartに登録します。 `<STANDBY_SPFILE_PATH>`をクエリから置き換えます（多くの場合、 `+DATA`の下）：

```
sudo -u oracle bash -c '
export ORACLE_SID=orcls
sqlplus -s / as sysdba <<< "SHOW PARAMETER spfile;"
'

sudo -u oracle bash -c '
export GRID_HOME=/u01/app/26ai/grid
export ORACLE_HOME=/u01/app/oracle/product/26ai/db_1
export PATH=$ORACLE_HOME/bin:$GRID_HOME/bin:$PATH

srvctl add database \
  -db orcls \
  -dbname orcl \
  -oraclehome /u01/app/oracle/product/26ai/db_1 \
  -spfile <STANDBY_SPFILE_PATH> \
  -pwfile /u01/app/oracle/product/26ai/db_1/dbs/orapworcls \
  -role PHYSICAL_STANDBY \
  -startoption MOUNT \
  -stopoption IMMEDIATE \
  -diskgroup DATA,RECO,FRA

srvctl config database -db orcls
srvctl status database -db orcls
'
```

2. `oracdb1`のプライマリデータベースリソースを確認して更新し、すべての ASM ディスク グループの依存関係を含めます：

```
sudo -u oracle bash -c '
export GRID_HOME=/u01/app/26ai/grid
export ORACLE_HOME=/u01/app/oracle/product/26ai/db_1
export PATH=$ORACLE_HOME/bin:$GRID_HOME/bin:$PATH
srvctl config database -db orcl
srvctl modify database -db orcl -diskgroup DATA,RECO,FRA
srvctl config database -db orcl
'
```

3. アプリケーション サービスを、 `oracdb2`上のスタンバイ データベース リソース（`orcls`）に追加します。スイッチオーバー後に `orclapp`を利用できるように、両側で `role PRIMARY`を使用します：

```
sudo -u oracle bash -c '
export GRID_HOME=/u01/app/26ai/grid
export ORACLE_HOME=/u01/app/oracle/product/26ai/db_1
export PATH=$ORACLE_HOME/bin:$GRID_HOME/bin:$PATH

srvctl add service \
  -db orcls \
  -service orclapp \
  -pdb orclpdb \
  -role PRIMARY \
  -policy AUTOMATIC

srvctl config service -db orcls -service orclapp
'
```

4. スタンバイデータベースリソースを確認します `oracdb2`：

```
sudo -u oracle bash -c '
export GRID_HOME=/u01/app/26ai/grid
export ORACLE_HOME=/u01/app/oracle/product/26ai/db_1
export PATH=$ORACLE_HOME/bin:$GRID_HOME/bin:$PATH
srvctl status database -db orcls
'
```

次の手順

ティア別：

- **Prod HA (データガードなし):** ストレージレプリケーションベースのリカバリターゲットを維持するために、スタンバイ初期化が完了し、スタンバイデータベースがバックアップインスタンスとしてOracle Restartに登録されます。

- **Prod HA (Data Guard + FSFO):** ブローカー管理のスイッチオーバーと高速スタートフェイルオーバーを有効にするには、Data Guard用のスタンバイデータベースを確定するに進みます。

Google Cloud NetApp Volumes 上の Data Guard のスタンバイ データベースを確定する

スタンバイREDOログファイルの作成、フラッシュバックデータベースの有効化、REDO転送の有効化、およびData Guardの状態確認によって、Google Cloud NetApp Volumes上のOracle Data Guardのスタンバイデータベースを確定します。

ティア固有： この手順は、**Prod HA (Data Guard + FSFO)** ティアでのみ必要です。

ステップ 1：スタンバイ REDO ログファイルを作成する

高速スタート・フェイルオーバーをサポートするために、両方のデータベースホストにスタンバイREDOログファイルを作成します。サイズは最大のプライマリオンラインREDOログ以上である必要があり、カウントは（スレッドあたりのオンライングループ）+ 1 と等しくなければなりません。GCNVシード処理後、レプリケートされたパスを修正するために、スタンバイ側でスタンバイREDOログを削除して再作成します。

1. プライマリデータベース上にスタンバイREDOログファイルを作成する(`orcl`）：

```
ALTER SYSTEM SET db_create_file_dest='+DATA' SCOPE=BOTH;
ALTER DATABASE ADD STANDBY LOGFILE THREAD 1 ('+DATA') SIZE 1024M;
-- repeat (online log groups + 1) times
```

2. GCNV シード後、スタンバイ データベース上のスタンバイ REDO ログ ファイルを削除して再作成します（(`orcls`）。 `+DATA/ORCL/…` 配下の複製されたパスは、再構築されるまで `ORA-19527` / `ORA-16086` を引き起こします：

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
ALTER SYSTEM SET standby_file_management=MANUAL SCOPE=BOTH;
-- DROP STANDBY LOGFILE GROUP for each group# in v$standby_log;
ALTER SYSTEM SET db_create_file_dest='+DATA' SCOPE=BOTH;
ALTER SYSTEM SET standby_file_management=AUTO SCOPE=BOTH;
ALTER DATABASE ADD STANDBY LOGFILE THREAD 1 ('+DATA') SIZE 1024M;
-- repeat (online groups + 1) times; one member per group
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE
DISCONNECT FROM SESSION;
```

ステップ2：フラッシュバックを有効にしてリカバリを開始する

スタンバイ側でフラッシュバック データベースを有効にして、フェイルオーバー後の自動復旧をサポートし、その後、リアルタイム適用でマネージド リカバリを開始します。MRP がアクティブな間はフラッシュバックを有効にできないため、マネージド リカバリを開始する前にフラッシュバックを有効にする必要があります。

1. スタンバイデータベースをシャットダウンし、MOUNTモードで再起動し、 `oracdb2`でフラッシュバックデータベースを有効にします：

```
# On oracdb2
sudo -u oracle bash -c '
. ~/.bash_profile
export ORACLE_SID=orcls
sqlplus / as sysdba <<SQL
SHUTDOWN IMMEDIATE;
STARTUP MOUNT;
ALTER SYSTEM SET db_flashback_retention_target=1440 SCOPE=BOTH;
ALTER DATABASE FLASHBACK ON;
EXIT
SQL'
```

2. リアルタイム適用による管理リカバリの開始：

```
sudo -u oracle bash -c '
. ~/.bash_profile
export ORACLE_SID=orcls
sqlplus / as sysdba <<SQL
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE
DISCONNECT FROM SESSION;
EXIT
SQL'
```

`USING CURRENT LOGFILE`リアルタイム適用を有効にします（SRLに到着した時点でredoが適用されます）。

ステップ3：redo shippingを有効にする

スタンバイ作成中の `ORA-12154`エラーを抑制するために、スタンバイ初期化手順のステップ2で意図的に`DEFER`に設定されていた `LOG_ARCHIVE_DEST_STATE_2`を有効にすることで、プライマリからスタンバイへのREDO転送を有効にします。

1. `LOG_ARCHIVE_DEST_STATE_2`に切り替え `ENABLE`、ログスイッチを強制してREDOログの転送を開始します：

```
sudo -u oracle bash -c '
. ~/.bash_profile
sqlplus / as sysdba <<SQL
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE SCOPE=BOTH;
ALTER SYSTEM SWITCH LOGFILE;
ALTER SYSTEM ARCHIVE LOG CURRENT;
EXIT
SQL'
```

2. REDOログの転送が正しく機能していることを確認します。

```
sudo -u oracle bash -c '
. ~/.bash_profile
sqlplus / as sysdba <<SQL
SELECT dest_id, status, error FROM v\$archive_dest_status WHERE dest_id
IN (1,2);
EXIT
SQL'
# Expected: dest_id=2, STATUS=VALID, ERROR null.
```

`dest_2`が `ORA-12154`を示している場合は、プライマリをバウンスします。xref:gcnv-oracle-data-guard.adoc#enable-broker-on-both-databases[手順1：両方のデータベースでブローカーを有効にする]の後、DGMGRLを介してトランスポートを管理します。

ステップ 4：Data Guard の状態を確認する

プライマリデータベースが READ WRITE モードになっていること、およびスタンバイデータベースがマネージドリカバリでマウントされ、redo ログが適用されていることを確認してください。

1. プライマリ データベース ロールとオープン モードを確認します `oracdb1`：

```
sudo -u oracle sqlplus -s / as sysdba \
  <<<"SELECT database_role || ' | ' || open_mode FROM v\$database;"
# Expected: PRIMARY | READ WRITE
```

2. スタンバイデータベースのロール、オープンモード、および管理リカバリのステータスを `oracdb2`で確認します：

```
gcloud compute ssh oracdb2 --tunnel-through-iap --zone=us-west1-b

sudo -u oracle bash <<'BASH'
. ~/.bash_profile
export ORACLE_SID=orcls

sqlplus -s / as sysdba <<'SQL'
SELECT database_role || ' | ' || open_mode
FROM v$database;

SELECT process, status, sequence#
FROM v$managed_standby
WHERE process IN ('MRP0','RFS');

EXIT
SQL
BASH
```

スタンバイで想定される値： `PHYSICAL STANDBY | MOUNTED`; `MRP0`（ `APPLYING_LOG`を使用）。

3. スタンバイが `MOUNTED`を報告しているが apply が実行されていない場合は、 `oracdb2`でマネージドリカバリを再起動します：

```
sudo -u oracle bash -c '
. ~/.bash_profile
export ORACLE_SID=orcls
sqlplus / as sysdba <<SQL
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE
DISCONNECT FROM SESSION;
EXIT
SQL'
```

次の手順

自動ロール管理とフェイルオーバー保護を有効にするには、Oracle Data Guard Broker、Fast-Start Failover、およびObserverを設定するに進んでください。

Google Cloud NetApp Volumes 上の Oracle Database 26ai 向けに Data Guard Broker と Fast-Start フェイルオーバーを構成する

専用のオブザーバーを使用してOracle Data Guard BrokerとFast-Start Failoverを構成

し、Google Cloud NetApp Volumes上のOracle Database 26aiの自動ロール遷移を有効にします。

ティア固有： この手順は **Prod HA (Data Guard + FSFO)** ティアにのみ適用されます。

この手順では、両方のデータベースでブローカーを有効にすること、Data Guard 構成を作成すること、MaxAvailability 保護モードで FSFO を有効にすること、Observer ホストへの Oracle Instant Client のインストール、ウォレットベースの認証情報を使用した systemd サービスとしての Observer の起動、および切り替えとフェイルオーバーのテストについて説明します。後 `ENABLE CONFIGURATION`、トランスポートとロールは **DGMGRL** で管理します （アドホック `LOG_ARCHIVE_DEST_*` SQL ではなく）。

ステップ 1：Data Guard Broker を有効にする

両方のデータベースホストでData Guard Brokerを有効にし、プライマリデータベースとスタンバイデータベースを統合管理下に置くためのブローカー構成を作成します。

1. プライマリおよびスタンバイのデータベースホストで以下を設定します `dg_broker_start=TRUE`：

```
sudo -u oracle bash -c '
. ~/.bash_profile
sqlplus / as sysdba <<SQL
ALTER SYSTEM SET dg_broker_start=TRUE SCOPE=BOTH;
EXIT
SQL'
```

2. プライマリ側で、OS認証を使用してDGMGRLに接続し、ブローカー構成を作成します。

> Observer ホストでのみ、自動ログイン ウォレットが存在した後に `dgmgrl /@orcl`を使用します。 `dgmgrl`コマンド ラインにパスワードを入力しないでください。

```
sudo -u oracle bash -c '
export ORACLE_HOME=/u01/app/oracle/product/26ai/db_1
export ORACLE_SID=orcl
export PATH=$ORACLE_HOME/bin:$PATH
dgmgrl /
'
```

```
DGMGRL> CREATE CONFIGURATION 'orcl_dg' AS
        PRIMARY DATABASE IS 'orcl' CONNECT IDENTIFIER IS orcl;
DGMGRL> ADD DATABASE 'orcls' AS CONNECT IDENTIFIER IS orcls;
DGMGRL> ENABLE CONFIGURATION;
DGMGRL> SHOW CONFIGURATION;
-- Expect: Configuration Status: SUCCESS, both members SUCCESS.
```

3. 構成を検証します。 `WARNING`またはNULL以外 `ERROR`をステップ3：FSFOプロパティを設定して有効にするの前に修正してください：

```
DGMGRL> VALIDATE DATABASE 'orcls';
DGMGRL> SHOW CONFIGURATION VERBOSE;
```

ステップ2：FSFOのフラッシュバックを確認する

両方のホストでフラッシュバックデータベースが有効になっていることを確認してください。FSFOの自動復旧にはフラッシュバックが必要です。これにより、フェイルオーバー後に以前のプライマリがスタンバイとして自動的に構成に復帰できるようになります。

1. 両方のデータベースホストで `flashback_on`が `YES`であることを確認します：

```
sudo -u oracle bash -c '
. ~/.bash_profile
sqlplus -s / as sysdba <<<"SELECT flashback_on FROM v\$database;"
'
# Expected on both hosts: YES
```

2. プライマリデバイスのみで、フラッシュバック保持がまだ設定されていない場合：

```
sudo -u oracle bash -c '
. ~/.bash_profile
export ORACLE_SID=orcl
sqlplus / as sysdba <<SQL
ALTER SYSTEM SET db_flashback_retention_target=1440 SCOPE=BOTH;
EXIT
SQL'
```

ステップ3：FSFOの設定と有効化

SYNC リドゥ転送を設定し、MaxAvailability 保護モードを構成し、各データベースで FSFO ターゲットを定義して、Fast-Start Failover を有効にします。

1. 両方のデータベースでリドゥ転送モードを `SYNC`に設定し、保護モードを `MaxAvailability`に上げます：

```
DGMGRL> EDIT DATABASE 'orcl'  SET PROPERTY LogXptMode='SYNC';
DGMGRL> EDIT DATABASE 'orcls' SET PROPERTY LogXptMode='SYNC';
DGMGRL> EDIT CONFIGURATION SET PROTECTION MODE AS MaxAvailability;
```

2. 各データベースが互いをフェイルオーバーターゲットとして指定するようにFSFOターゲットを設定し、

しきい値と自動復旧動作を構成します：

```
-- Each side names the other
DGMGRL> EDIT DATABASE 'orcl'  SET PROPERTY FastStartFailoverTarget =
'orcls';
DGMGRL> EDIT DATABASE 'orcls' SET PROPERTY FastStartFailoverTarget =
'orcl';

-- 30 s is the default; lower for faster RTO but more sensitive to
network blips
DGMGRL> EDIT CONFIGURATION SET PROPERTY FastStartFailoverThreshold = 30;
DGMGRL> EDIT CONFIGURATION SET PROPERTY FastStartFailoverAutoReinstate =
TRUE;
```

3. 高速フェイルオーバーを有効にして、設定を確認します。

```
DGMGRL> ENABLE FAST_START FAILOVER;
DGMGRL> SHOW FAST_START FAILOVER;
-- Expected: Threshold 30 seconds, Target orcls, Observer not yet
registered.
```

ステップ4：オブザーバーにInstant Clientをインストールする

専用のオブザーバーVM（`oradg-obs`）にOracle Instant Clientをインストールし、専用の `oracle` OSユーザーを作成して、ObserverがTCP/1521で両方のデータベースメンバーに接続できるようにOracle Net環境を構成します。

1. ObserverホストにOracle Instant Clientパッケージをインストールします（`oradg-obs`）：

```
# Use -el8 / -el9 if the Observer is on an older OL/RHEL release
sudo dnf install -y oracle-instantclient-release-el10
sudo dnf install -y oracle-instantclient-basic \
                    oracle-instantclient-sqlplus \
                    oracle-instantclient-tools
```

2. ウォレットとsystemdユニットを所有する専用の `oracle` OSユーザーを作成します：

```
sudo useradd -u 54321 -m oracle
sudo passwd -l oracle
```

3. Oracle Net 環境を構成し、両方のデータベースホストのエントリを含む `tnsnames.ora`を作成します：

```
sudo mkdir -p /etc/oracle/network/admin
sudo chown -R oracle:oracle /etc/oracle

sudo -u oracle tee /home/oracle/.bash_profile >/dev/null <<'EOF'
export ORACLE_HOME=/usr/lib/oracle/26/client64
export LD_LIBRARY_PATH=$ORACLE_HOME/lib
export PATH=$ORACLE_HOME/bin:$PATH
export TNS_ADMIN=/etc/oracle/network/admin
EOF

# tnsnames.ora — must reach both DB hosts on TCP/1521
sudo tee /etc/oracle/network/admin/tnsnames.ora >/dev/null <<'EOF'
orcl =
  (DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP)(HOST = oracdb1)(PORT = 1521))
                (CONNECT_DATA = (SERVER = DEDICATED)(SERVICE_NAME =
orcl)))
orcls =
  (DESCRIPTION =
  (ADDRESS = (PROTOCOL = TCP)(HOST = oracdb2)(PORT = 1521))
                (CONNECT_DATA = (SERVER = DEDICATED)(SERVICE_NAME =
orcls)))
EOF
sudo chown oracle:oracle /etc/oracle/network/admin/tnsnames.ora
```

ステップ5：Observerをsystemdサービスとして実行する

両方のデータベースメンバーの認証情報を含む自動ログインウォレットを作成し、Observerをsystemdサービスとして設定および起動して、再起動後も動作し、自動的に設定に再接続するようにします。

専用の Data Guard 管理アカウント（たとえば、 `SYSDG`）の資格情報は、 `SYS` ではなくウォレットに保存します。資格情報は、 `ps` や `journalctl` から確認できる状態になってしまうため、 `dgmgrl` コマンドライン上には絶対に表示させないでください。Observer では常に `/@<tns_alias>` を使用して接続してください。

1. 暗号化ウォレットを作成し、両方のデータベースメンバーの認証情報を入力します。

```
sudo -iu oracle bash <<'BASH'
mkdir -p $TNS_ADMIN/wallet
mkstore -wrl $TNS_ADMIN/wallet -create     # prompts for a wallet
password — store in your secrets manager
mkstore -wrl $TNS_ADMIN/wallet -createCredential orcl  sys ChangeMe!1
mkstore -wrl $TNS_ADMIN/wallet -createCredential orcls sys ChangeMe!1
BASH

sudo tee /etc/oracle/network/admin/sqlnet.ora >/dev/null <<'EOF'
WALLET_LOCATION = (SOURCE = (METHOD = FILE) (METHOD_DATA = (DIRECTORY =
/etc/oracle/network/admin/wallet)))
SQLNET.WALLET_OVERRIDE = TRUE
EOF
sudo chown oracle:oracle /etc/oracle/network/admin/sqlnet.ora
sudo chmod -R 0700 /etc/oracle/network/admin/wallet

sudo -iu oracle ls -l /etc/oracle/network/admin/wallet
# Expected: cwallet.sso and ewallet.p12

sudo -iu oracle bash <<'BASH'
sqlplus -L "/@orcl as sysdba" <<'SQL'
SELECT database_role FROM v$database;
EXIT
SQL
BASH

sudo -iu oracle bash <<'BASH'
sqlplus -L "/@orcls as sysdba" <<'SQL'
SELECT database_role FROM v$database;
EXIT
SQL
BASH

sudo -iu oracle dgmgrl /@orcl  'SHOW CONFIGURATION;'
sudo -iu oracle dgmgrl /@orcls 'SHOW CONFIGURATION;'
```

2. Observerのsystemdサービスがパスワードの入力プロンプトなしで起動できるように、自動ログインウォレット `cwallet.sso`）を生成します。 `mkstore`の実行後に `cwallet.sso`がない場合は、Instant Clientツールパッケージまたはデータベースホームの `orapki`を使用して作成し、保存されている資格情報を再追加します：

```
sudo -iu oracle orapki wallet create \
  -wallet /etc/oracle/network/admin/wallet \
  -auto_login
sudo -iu oracle ls -l /etc/oracle/network/admin/wallet
# Expected: cwallet.sso and ewallet.p12
```

3. systemd ユニットを作成し、サービスを有効にして、オブザーバーが接続されていることを確認します：

```
sudo tee /etc/systemd/system/dgmgrl-observer.service >/dev/null <<'EOF'
[Unit]
Description=Oracle Data Guard Fast-Start Failover Observer
After=network-online.target
Wants=network-online.target

[Service]
Type=simple
User=oracle
Group=oracle
Environment=ORACLE_HOME=/usr/lib/oracle/26/client64
Environment=LD_LIBRARY_PATH=/usr/lib/oracle/26/client64/lib
Environment=TNS_ADMIN=/etc/oracle/network/admin
Environment=PATH=/usr/lib/oracle/26/client64/bin:/usr/bin:/bin
ExecStart=/usr/lib/oracle/26/client64/bin/dgmgrl -silent /@orcl "START
OBSERVER FILE IS '/var/lib/oracle/dgmgrl-observer.dat'"
Restart=always
RestartSec=5

[Install]
WantedBy=multi-user.target
EOF
```

```
sudo install -d -o oracle -g oracle -m 0755 /var/lib/oracle
sudo install -o oracle -g oracle -m 0640 /dev/null /var/log/dgmgrl-
observer.log

sudo tee /etc/logrotate.d/dgmgrl-observer >/dev/null <<'EOF'
/var/log/dgmgrl-observer.log {
    weekly
    rotate 8
    compress delaycompress missingok notifempty
    create 0640 oracle oracle
    copytruncate
}
EOF

sudo systemctl daemon-reload && sudo systemctl enable --now dgmgrl-
observer.service
sudo systemctl status dgmgrl-observer.service
```

オブザーバーはプライマリから読み取る必要があります `CONNECTED`（`DISCONNECTED`オブザーバーは暗黙的にFSFOを停止します）：

```
DGMGRL> SHOW FAST_START FAILOVER;
DGMGRL> SHOW CONFIGURATION;       -- Configuration Status: SUCCESS,
FSFO: ENABLED
```

ステップ6：FSFOをテストする

Data Guard の構成を `VALIDATE DATABASE`で検証し、次に計画的なスイッチオーバーを実行して、テスト期間中に計画外の VM リセット フェイルオーバーを実行し、FSFO がエンドツーエンドで正常に動作していることを確認します。

1. 計画された切り替えをテストし、元のトポロジを復元します。

```
DGMGRL> VALIDATE DATABASE 'orcls';
DGMGRL> SWITCHOVER TO 'orcls';
DGMGRL> SHOW CONFIGURATION;
DGMGRL> SWITCHOVER TO 'orcl';        -- restore topology
```

2. 制御されたテストウィンドウ内で、VM のリセットを使用して計画外のフェイルオーバーをテストします。

   VMの*リセット*（クラッシュ形式のテスト）を使用してください。通常の*停止*ではFSFOがトリガーされない場合があります。 `/var/log/dgmgrl-observer.log`tail `oradg-obs`を使用してフェイルオーバーの進行状況を監視し、完了したらトポロジーを復元してください。

次の手順

このデプロイメントでは、Oracle Data Guard Broker、Fast-Start Failover、およびObserverの構成が完了しました。